Efficient Adversarial Training without Attacking: Worst-Case-Aware Robust Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent studies reveal that a well-trained deep reinforcement learning (RL) policy can be particularly vulnerable to adversarial perturbations on input observations. Therefore, it is crucial to train RL agents that are robust against any attacks with a bounded budget. Existing robust training methods in deep RL either treat correlated steps separately, ignoring the robustness of long-term reward, or train the agents and RL-based attacker together, doubling the computational burden and sample complexity of the training process. In this work, we propose a strong and efficient robust training framework for RL, named Worst-case-aware Robust RL (WocaR-RL), that directly estimates and optimizes the worst-case reward of a policy under bounded ℓ_p attacks without requiring extra samples for learning an attacker. Experiments on multiple environments show that WocaR-RL achieves state-of-the-art performance under various strong attacks, and obtains significantly higher training efficiency than prior state-of-the-art robust training methods.

1 Introduction

Deep reinforcement learning (DRL) has achieved impressive results by using deep neural networks (DNN) to learn complex policies in high-dimensional environments. However, well-trained DNNs may drastically fail under adversarial perturbations of the input [1, 5]. Therefore, before deploying DRL policies to real-life applications, there is a crucial need to improve the robustness of deep policies against adversarial attacks, especially worst-case attacks that maximally depraves the performance of trained agents [41].

A line of regularization-based robust methods [52, 32, 39] focuses on improving the robustness of the DNN itself and regularizes the policy network to output similar actions under bounded state perturbations. However, different from supervised learning problems, the vulnerability of a deep policy comes not only from the DNN approximator, but also from the dynamics of the RL environment [50]. These regularization-based methods neglect the intrinsic vulnerability of policies under the environment dynamics, and thus may still fail under strong attacks [41]. For example, in the go-home task shown in Figure 1, both the green policy and the red policy arrive home without rock collision, when there is no attack. However, although regularization-based methods may ensure a minor action change under a state perturbation, the red policy may still be susceptible to a low reward under attacks, as a very small divergence can lead it to the bomb. On the contrary, the green policy is more robust to adversarial attacks since it stays away from the bomb. Therefore, besides promoting the robustness of DNN approximators (such as the policy network), it is also important to learn a policy with stronger intrinsic robustness.

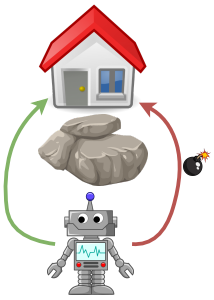


Figure 1: Policies have different vulnerabilities.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

There is another line of work considering the long-term robustness of a deep policy under strong adversarial attacks. In particular, it is theoretically proved [52, 41] that the strongest (worst-case) attacker against a policy can be learned as an RL problem, and training the agent under such a learned attacker can result in a robust policy. Zhang et al. [50] propose the *Alternating Training with Learned Adversaries (ATLA)* framework, which alternately trains an RL agent and an RL attacker. Sun et al. [41] further propose PA-ATLA, which alternately trains an agent and the proposed more efficient PA-AD RL attacker, obtaining state-of-the-art robustness in many MuJoCo environments. However, training an RL attacker requires extra samples from the environment, and the attacker's RL problem may even be more difficult and sample expensive to solve than the agent's original RL problem [50, 41], especially in large-scale environments such as Atari games with pixel observations. Therefore, although ATLA and PA-ATLA are able to achieve high long-term reward under attacks, they double the computational burden and sample complexity to train the robust agent.

The above analysis of existing literature suggests two main challenges in improving the adversarial robustness of DRL agents: (1) correctly characterizing the long-term reward vulnerability of an RL policy, and (2) efficiently training a robust agent without requiring much more effort than vanilla training. To tackle these challenges, in this paper, we propose a generic and efficient robust training framework named *Worst-case-aware Robust RL (WocaR-RL)* that estimates and improves the long-term robustness of an RL agent.

WocaR-RL has 3 key mechanisms. First, WocaR-RL introduces a novel *worst-attack Bellman operator* which uses existing off-policy samples to estimate the lower bound of the policy value under the worst-case attack. Compared to prior works [50, 41] which attempt to learn the worst-case attack by RL methods, WocaR-RL does not require any extra interaction with the environment. Second, using the estimated worst-case policy value, WocaR-RL optimizes the policy to select actions that not only achieve high natural future reward, but also achieve high worst-case reward when there are adversarial attacks. Therefore, WocaR-RL learns a policy with less intrinsic vulnerability. Third, WocaR-RL regularizes the policy network with a carefully designed state importance weight. As a result, the DNN approximator tolerates state perturbations, especially for more important states where decisions are crucial for future reward. The above 3 mechanisms can also be interpreted from a geometric perspective of adversarial policy learning, as detailed in Appendix B.

Our **contributions** can be summarized as below. **(1)** We provide an approach to estimate the worst-case value of any policy under any bounded ℓ_p adversarial attacks. This helps evaluate the robustness of a policy without learning an attacker which requires extra samples and exploration. **(2)** We propose a novel and principled robust training framework for RL, named *Worst-case-aware Robust RL (WocaR-RL)*, which characterizes and improves the worst-case robustness of an agent. WocaR-RL can be used to robustify existing DRL algorithms (e.g. PPO [38], DQN [31]). **(3)** We show by experiments that WocaR-RL achieve **improved robustness** against various adversarial attacks as well as **higher efficiency**, compared with state-of-the-art (SOTA) robust RL methods in many MuJoCo and Atari games. For example, compared to the SOTA algorithm PA-ATLA-PPO [41] in the Walker environment, we obtain 20% more worst-case reward (under the strongest attack algorithm), with only about 50% training samples and 50% running time. Moreover, WocaR-RL learns **more interpretable "robust behaviors"** than PA-ATLA-PPO in Walker as shown in Figure 2.

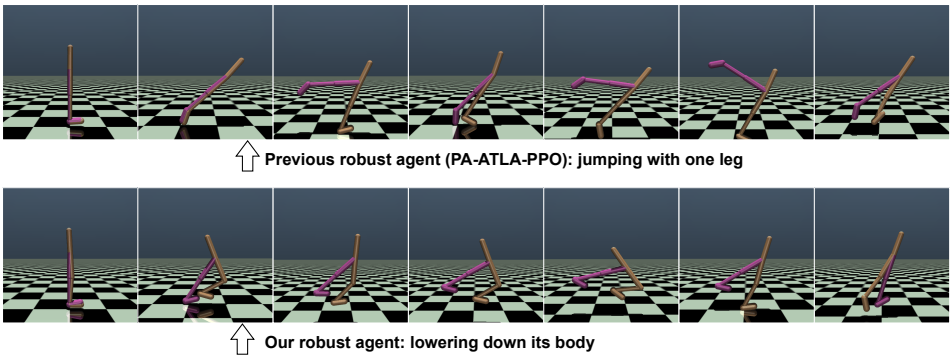


Figure 2: The robust Walker agents trained with **(top)** the state-of-the-art method PA-ATLA-PPO [41] and **(bottom)** our WocaR-RL. Although PA-ATLA-PPO agent also achieves high reward under attacks, it learns to jump with one leg, which is counter-intuitive and may indicate some level of overfitting to a specific attacker. In contrast, our WocaR-RL agent learns to lower down its body, which is more intuitive and interpretable. The full agent trajectories in Walker and other environments are provided in supplementary materials as GIF figures.

2 Related Work

Defending against Adversarial Perturbations on State Observations. **(1)** *Regularization-based methods* [52, 39, 32] enforce the policy to have similar outputs under similar inputs, which achieves certifiable performance for DQN in some Atari games. But in continuous control tasks, these methods may not reliably improve the worst-case performance. A recent work by Korkmaz [20] points out that these adversarially trained models may still be sensible to new perturbations. **(2)** *Attack-driven methods* train DRL agents with adversarial examples. Some early works [21, 4, 28, 33] apply weak or strong gradient-based attacks on state observations to train RL agents against adversarial perturbations. Zhang et al. [50] propose Alternating Training with Learned Adversaries (ATLA), which alternately trains an RL agent and an RL adversary and significantly improves the policy robustness in continuous control games. Sun et al. [41] further extend this framework to PA-ATLA with their proposed more advanced RL attacker PA-AD. Although ATLA and PA-ATLA achieve strong empirical robustness, they require training an extra RL adversary that can be computationally and sample expensive. **(3)** There is another line of work studying *certifiable robustness* of RL policies. Several works [26, 32, 8] computed lower bounds of the action value network Q^π to certify robustness of action selection at every step. However, these bounds do not consider the distribution shifts caused by attacks, so some actions that appear safe for now can lead to extremely vulnerable future states and low long-term reward under future attacks. Moreover, these methods cannot apply to continuous action spaces. Kumar et al. and Wu et al.[22, 48] both extend randomized smoothing [6] to derive robustness certificates for trained policies. But these works mostly focus on theoretical analysis, and effective robust training approaches rather than robust training.

Adversarial Defenses against Other Adversarial Attacks. Besides observation perturbations, attacks can happen in many other scenarios. For example, the agent's executed actions can be perturbed [49, 43, 44, 23]. Moreover, in a multi-agent game, an agent's behavior can create adversarial perturbations to a victim agent [12]. Pinto et al. [34] model the competition between the agent and the attacker as a zero-sum two-player game, and train the agent under a learned attacker to tolerate both environment shifts and adversarial disturbances. We point out that although we mainly consider state adversaries, our WocaR-RL can be extended to action attacks as formulated in Appendix C.5. Note that we focus on robustness against test-time attacks, different from poisoning attacks which alter the RL training process [3, 19, 40, 54, 35].

Safe RL and Risk-sensitive RL. There are several lines of work that study RL under safety/risk constraints [17, 10, 9, 2, 45] or under intrinsic uncertainty of environment dynamics [25, 29]. However, these works do not deal with adversarial attacks, which can be adaptive to the learned policy. More comparison between these methods and our proposed method is discussed in Section 4.

3 Preliminaries and Background

Reinforcement Learning (RL). An RL environment is modeled by a Markov Decision Process (MDP), denoted by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a stochastic dynamics model[1], $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function and $\gamma \in [0, 1)$ is a discount factor. An agent takes actions based on a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. For any policy, its *natural performance* can be measured by the value function $V^\pi(s) := \mathbb{E}_{P,\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s]$, and the action value function $Q^\pi(s, a) := \mathbb{E}_{P,\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a]$. We call V^π the *natural value* and Q^π the *natural action value* in contrast to the values under attacks, as will be introduced in Section 4.

Deep Reinforcement Learning (DRL). In large-scale problems, a policy can be parameterized by a neural network. For example, value-based RL methods (e.g. DQN [31]) usually fit a Q network and take the greedy policy $\pi(s) = \mathrm{argmax}_a Q(s, a)$. In actor-critic methods (e.g. PPO [38]), the learner directly learns a policy network and a critic network. In practice, an agent usually follows a stochastic policy during training that enables exploration, and executes a trained policy deterministically in test-time, e.g. the greedy policy learned with DQN. Throughout this paper, we use π_θ to denote the training-time stochastic policy parameterized by θ, while π denotes the trained deterministic policy that maps a state to an action.

Test-time Adversarial Attacks. After training, the agent is deployed into the environment and executes a pre-trained fixed policy π. An attacker/adversary, during the deployment of the agent, may

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

perturb the state observation of the agent/victim at every time step with a certain attack budget ϵ. Note that the attacker only perturbs the inputs to the policy, and the underlying state in the environment does not change. This is a realistic setting because real-world observations can come from noisy sensors or be manipulated by malicious attacks. For example, an auto-driving car receives sensory observations; an attacker may add imperceptible noise to the camera, or perturb the GPS signal, although the underlying environment (the road) remains unchanged. In this paper, we consider the ℓ_p *thread model* which is widely used in adversarial learning literature: at step t, the attacker alters the observation s_t into $\tilde{s}_t \in \mathcal{B}_\epsilon(s_t)$, where $\mathcal{B}_\epsilon(s_t)$ is a ℓ_p norm ball centered at s_t with radius ϵ. The above setting (ℓ_p constrained observation attack) is the same with many prior works [18, 33, 52, 50, 41].

4 Worst-case-aware Robust RL

In this section, we present *Worst-case-aware Robust RL (WocaR-RL)*, a generic framework that can be fused with any DRL approach to improve the adversarial robustness of an agent. We will introduce the three key mechanisms in WocaR-RL: worst-attack value estimation, worst-case-aware policy optimization, and value-enhanced state regularization, respectively. Then, we will illustrate how to incorporate these mechanisms into existing DRL algorithms to improve their robustness.

Mechanism 1: Worst-attack Value Estimation

Traditional RL aims to learn a policy with the maximal value V^π. However, in a real-world problem where observations can be noisy or even adversarially perturbed, it is not enough to only consider the natural value V^π and Q^π. As motivated in Figure 1, two policies with similar natural rewards can get totally different rewards under attacks. To comprehensively evaluate how good a policy is in an adversarial scenario and to improve its robustness, we should be aware of the lowest possible long-term reward of the policy when its observation is adversarially perturbed with a certain attack budget ϵ at every step (with an ℓ_p attack model introduced in Section 3).

The worst-case value of a policy is, by definition, the cumulative reward obtained under the optimal attacker. As justified by prior works [52, 41], for any given victim policy π and attack budget $\epsilon > 0$, there exists an optimal attacker, and finding the optimal attacker is equivalent to learning the optimal policy in another MDP. We denote the optimal (deterministic) attacker's policy as h^*. However, learning such an optimal attacker by RL algorithms requires extra interaction samples from the environment, due to the unknown dynamics. Moreover, learning the attacker by RL can be hard and expensive, especially when the state observation space is high-dimensional.

Instead of explicitly learning the optimal attacker with a large amount of samples, we propose to directly estimate the worst-case cumulative reward of the policy by characterizing the vulnerability of the given policy. We first define the *worst-attack action value* of policy π as $\underline{Q}^\pi(s,a) := \mathbb{E}_P[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, \pi(h^*(s_t))\right) \mid s_0 = s, a_0 = a]$. The *worst-attack value* $\underline{V}^\pi$ can be defined using h^* in the same way, as shown in Definition A.1 in Appendix A. Then, we introduce a novel operator $\underline{\mathcal{T}}^\pi$, namely the *worst-attack Bellman operator*, defined as below.

Definition 4.1 (Worst-attack Bellman Operator)**.** For MDP $\mathcal{M}$, given a fixed policy π and attack radius ϵ, define the worst-attack Bellman operator $\underline{\mathcal{T}}^\pi$ as

$$\left(\underline{\mathcal{T}}^\pi Q\right)(s,a) := \mathbb{E}_{s' \sim P(s,a)}[R(s,a) + \gamma \min_{a' \in \mathcal{A}_{\text{adv}}(s',\pi)} Q\left(s', a'\right)], \tag{1}$$

where $\forall s \in \mathcal{S}$, $\mathcal{A}_{\text{adv}}(s,\pi)$ is defined as

$$\mathcal{A}_{\text{adv}}(s,\pi) := \{a \in \mathcal{A} : \exists \tilde{s} \in \mathcal{B}_\epsilon(s) \text{ s.t. } \pi(\tilde{s}) = a\}. \tag{2}$$

Here $\mathcal{A}_{\text{adv}}(s',\pi)$ denotes the set of actions an adversary can mislead the victim π into selecting by perturbing the state s' into a neighboring state $\tilde{s} \in \mathcal{B}_\epsilon(s')$. This hypothetical perturbation to the *future* state s' is the key for characterizing the worst-case long-term reward under attack. The following theorem associates the worst-attack Bellman operator and the worst-attack action value.

Theorem 4.2 (Worst-attack Bellman Operator and Worst-attack Action Value)**.** *For any given policy π, $\underline{\mathcal{T}}^\pi$ is a contraction whose fixed point is $\underline{Q}^\pi$, the worst-attack action value of π under any ℓ_p observation attacks with radius ϵ.*

Theorem 4.2 proved in Appendix A suggests that the lowest possible cumulative reward of a policy under bounded observation attacks can be computed by worst-attack Bellman operator. The corresponding worst-attack value $\underline{V}^\pi$ can be obtained by $\underline{V}^\pi(s) = \min_{a \in \mathcal{A}_{\text{adv}}(s,\pi)} \underline{Q}^\pi(s,a)$.

How to Compute $\mathcal{A}_{\text{adv}}$. To obtain $\mathcal{A}_{\text{adv}}(s,\pi)$, we need to identify the actions that can be the outputs of the policy π when the input state s is perturbed within $\mathcal{B}_\epsilon(s)$. This can be solved by

commonly-used convex relaxation of neural networks [14, 53, 47, 51, 13], where layer-wise lower and upper bounds of the neural network are derived. That is, we calculate $\overline{\pi}$ and $\underline{\pi}$ such that $\overline{\pi}(s) \geq \pi(\hat{s}) \geq \underline{\pi}(s), \forall \hat{s} \in \mathcal{B}_\epsilon(s)$. With such a relaxation, we can obtain a superset of $\mathcal{A}_{\text{adv}}$, namely $\hat{\mathcal{A}}_{\text{adv}}$. Then, the fixed point of Equation (1) with $\mathcal{A}_{\text{adv}}$ being replaced by $\hat{\mathcal{A}}_{\text{adv}}$ becomes a lower bound of the worst-attack action value. For a continuous action space, $\hat{\mathcal{A}}_{\text{adv}}(s, \pi)$ contains actions bounded by $\overline{\pi}(s)$ and $\underline{\pi}(s)$. For a discrete action space, we can first compute the maximal and minimal probabilities of taking each action, and derive the set of actions that are likely to be selected. The computation of $\hat{\mathcal{A}}_{\text{adv}}$ is not expensive, as there are many efficient convex relaxation methods [30, 51] which compute $\overline{\pi}$ and $\underline{\pi}$ with only constant-factor more computations than directly computing $\pi(s)$. Experiment in Section 5 verifies the efficiency of our approach. More implementation details and explanations are provided in Appendix C.1.

Estimating Worst-attack Value. Note that the worst-attack Bellman operator $\underline{\mathcal{T}}^\pi$ is similar to the optimal Bellman operator $\mathcal{T}^*$, although it uses $\min_{a \in \mathcal{A}_{\text{adv}}}$ instead of $\max_{a \in \mathcal{A}}$. Therefore, once we identify $\mathcal{A}_{\text{adv}}$ as introduced above, it is straightforward to compute the worst-attack action value using Bellman backups. To model the worst-attack action value, we train a network named *worst-attack critic*, denoted by $\underline{Q}^\pi_\phi$, where ϕ is the parameterization. Concretely, for any mini-batch $\{s_t, a_t, r_t, s_{t+1}\}_{t=1}^N$, $\underline{Q}^\pi_\phi$ is optimized by minimizing the following estimation loss:

$$\mathcal{L}_{\text{est}}(\underline{Q}^\pi_\phi) := \frac{1}{N} \sum_{t=1}^{N} (\underline{y}_t - \underline{Q}^\pi_\phi(s_t, a_t))^2, \text{where } \underline{y}_t = r_t + \gamma \min_{\hat{a} \in \mathcal{A}_{\text{adv}}(s_{t+1}, \pi)} \underline{Q}^\pi_\phi(s_{t+1}, \hat{a}). \quad (3)$$

For a discrete action space, $\mathcal{A}_{\text{adv}}$ is a discrete set and solving $\underline{y}_t$ is straightforward. For a continuous action space, we use gradient descent to approximately find $\hat{a}$ that minimizes $\underline{Q}^\pi_\phi(s_{t+1}, \hat{a})$. Since $\mathcal{A}_{\text{adv}}$ is in general small, this minimization is usually easy to solve. In our experiments in MuJoCo, we find that 50-step gradient descent already converges to a good solution with little computational cost.

Differences with Worst-case Value Estimation in Related Work. Our proposed worst-attack Bellman operator is different from the worst-case Bellman operator in the literature of risk-sensitive RL [17, 10, 42, 9, 2, 45], whose goal is to avoid unsafe trajectories under the intrinsic uncertainties of the MDP. These inherent uncertainties of the environment are independent of the learned policy. In contrast, our focus is to defend against adversarial perturbations created by malicious attackers that can be *adaptive* to the policy. The GWC reward proposed by [32] also estimates the worst-case reward of a policy under state perturbations. But their evaluation is based on a greedy strategy and requires interactions with the environment, which is different from our estimation.

Mechanism 2: Worst-case-aware Policy Optimization

So far we have introduced how to evaluate the worst-attack value of a policy by learning a worst-attack critic. Inspired by the actor-critic framework, where the actor policy network π_θ is optimized towards a direction that the critic value increases the most, we can regard worst-attack critic as a special critic that directs the actor to increase the worst-attack value. That is, we encourage the agent to select an action with a higher worst-attack action value, by minimizing the worst-attack policy loss below:

$$\mathcal{L}_{\text{wst}}(\pi_\theta; \underline{Q}^\pi_\phi) := -\frac{1}{N} \sum_{t=1}^{N} \sum_{a \in \mathcal{A}} \pi_\theta(a|s_t) \underline{Q}^\pi_\phi(s_t, a), \quad (4)$$

where $\underline{Q}^\pi_\phi$ is the worst-attack critic learned via $\mathcal{L}_{\text{est}}$ introduced in Equation (3). Note that $\mathcal{L}_{\text{wst}}$ is a general form, while the detailed implementation of the worst-attack policy optimization can vary depending on the architecture of π_θ in the base RL algorithm (e.g. PPO has a policy network, while DQN acts using the greedy policy induced by a Q network). In Appendix C.2 and Appendix C.3, we illustrate how to implement $\mathcal{L}_{\text{wst}}$ for PPO and DQN as two examples.

The proposed worst-case-aware policy optimization has several **merits** compared to prior ATLA [50] and PA-ATLA [41] methods which alternately train the agent and an RL attacker. **(1)** Learning the optimal attacker h^* requires collecting extra samples using the current policy (on-policy estimation). In contrast, $\underline{Q}^\pi_\phi$ can be learned using off-policy samples, e.g., historical samples in the replay buffer, and thus is more suitable for training where the policy changes over time. ($\underline{Q}^\pi_\phi$ depends on the current policy via the computation of $\mathcal{A}_{\text{adv}}$.) **(2)** We properly exploit the policy function that is being trained by computing the set of possibly selected actions $\hat{\mathcal{A}}_{\text{adv}}$ for any state. In contrast, ATLA [50] learns an attacker by treating the current policy as a black box, ignoring the intrinsic properties of the policy. PA-ATLA [41], although assumes white-box access to the victim policy, also needs to explore and learn from extra on-policy interactions. **(3)** The attacker trained with DRL methods, namely $\hat{h}^*$, is

not guaranteed to converge to an optimal solution, such that the performance of π estimated under $\hat{h}^*$ can be overly optimistic. Our estimation, as mentioned in Mechanism 1, computes a lower bound of $\underline{Q}^\pi$ and thus can better indicate the robustness of a policy.

Mechanism 3: Value-enhanced State Regularization

As discussed in Section 1, the vulnerability of a deep policy comes from both the policy's intrinsic vulnerability with the RL dynamics and the DNN approximator. The first two mechanisms of WocaR-RL mainly focus on the policy's intrinsic vulnerability, i.e., let the policy select actions that are less vulnerable to possible attacks in all future steps. However, if a bounded state perturbation can cause the network to output a very different action, then the $\mathcal{A}_{\text{adv}}$ set will be large and $\underline{Q}^\pi$ can thus be low. Therefore, it is also important to encourage the trained policy to output similar actions for the clean state s and any $\tilde{s} \in \mathcal{B}_\epsilon(s)$, as is done in prior work [52, 39, 8].

But different from these prior methods, we note that different states should be treated differently. Some states are "critical" where selecting a bad action will result in catastrophic consequences. For example, when the agent gets close to the bomb in Figure 1, we should make the network more resistant to adversarial state perturbations. To differentiate states based on their impacts on future reward, we propose to measure the importance of states with Definition 4.3 below.

Definition 4.3 (State Importance Weight)**.** Define state importance weight of $s \in \mathcal{S}$ for policy π as

$$w(s) = \max_{a_1 \in \mathcal{A}} Q^\pi(s, a_1) - \min_{a_2 \in \mathcal{A}} Q^\pi(s, a_2). \tag{5}$$

To justify whether Definition 4.3 can characterize state importance, we train a DQN network in an Atari game Pong, and show the states with the highest weight and the lowest weight in Figure 3, among many state samples. We can see that the state with higher weight in Figure 3(left) is indeed crucial for the game, as the green agent paddle is close to the ball. Conversely, a less-important state in Figure 3(right) does not have significantly different future rewards under different actions. Computing $w(s)$ is easy in a discrete action space, while in a continuous action space, one can use gradient descent to approximately find the maximal and the minimal Q values for a state.

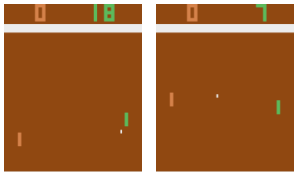


Figure 3: States in Pong with **(left)** high weight $w(s)$ and **(right)** low weight $w(s)$.

By incorporating the state importance weight $w(s)$, we regularize the policy network and let it pay more attention to more crucial states, by minimizing the following loss:

$$\mathcal{L}_{\text{reg}}(\pi_\theta) = \frac{1}{N} \sum_{t=1}^{N} w(s_t) \max_{\tilde{s}_t \in \mathcal{B}_\epsilon(s_t)} \text{Dist}(\pi_\theta(s_t), \pi_\theta(\tilde{s}_t)), \tag{6}$$

where Dist can be any distance measure between two distributions (e.g., KL-divergence). Minimizing $\mathcal{L}_{\text{reg}}$ can result in a smaller $\mathcal{A}_{\text{adv}}$, and thus the worst-attack value will be closer to the natural value.

WocaR-RL: A Generic Robust Training Framework

So far we have introduced three key mechanisms and their loss functions, $\mathcal{L}_{\text{est}}$ in Equation (3), $\mathcal{L}_{\text{wst}}$ in Equation (4) and $\mathcal{L}_{\text{reg}}$ in Equation (6). Then, our robust training framework WocaR-RL combines these losses with any base RL algorithm. To be more specific, as shown in Figure 4, for any base RL algorithm that trains policy π_θ using loss $\mathcal{L}_{\text{RL}}$, we learn an extra worst-attack critic network $\underline{Q}^\pi_\phi$ by minimizing

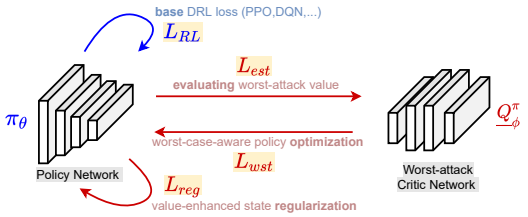


Figure 4: Training architecture of WocaR-RL. (Components proposed in this paper are colored as red.)

$$\mathcal{L}_{\underline{Q}^\pi_\phi} := \mathcal{L}_{\text{est}}(\underline{Q}^\pi_\phi), \tag{7}$$

and combine $\mathcal{L}_{\text{wst}}$ and $\mathcal{L}_{\text{reg}}$ with $\mathcal{L}_{\text{RL}}$ to optimize π_θ by minimizing

$$\mathcal{L}_{\pi_\theta} := \mathcal{L}_{\text{RL}}(\pi_\theta) + \kappa_{\text{wst}} \mathcal{L}_{\text{wst}}(\pi_\theta; \underline{Q}^\pi_\phi) + \kappa_{\text{reg}} \mathcal{L}_{\text{reg}}(\pi_\theta), \tag{8}$$

where κ_{wst} and κ_{reg} are hyperparameters balancing between natural performance and robustness. Note that $\underline{Q}^\pi_\phi$ is trained together but independently with π_θ using historical transition samples, so WocaR-RL does not require extra samples from the environment. WocaR-RL can also be interpreted from a geometric perspective based on prior RL polytope theory [7, 41] as detailed in Appendix B.

Our WocaR-RL is a generic robust training framework that can be used to robustify existing DRL algorithms. We provide two case studies: **(1)** combining WocaR-RL with a policy-based algo-

rithm PPO [38], namely *WocaR-PPO*, and **(2)** combining WocaR-RL with a value-based algorithm DQN [31], namely *WocaR-DQN*. The pseudocodes of WocaR-PPO and WocaR-DQN are illustrated in Appendix C.2 and Appendix C.3. The application of WocaR-RL to other DRL methods is then straightforward, since most DRL methods are either policy-based or value-based. Next, we show by experiments that WocaR-PPO and WocaR-DQN achieve state-of-the-art robustness with superior efficiency, in various continuous control tasks and video game environments. We also empirically verify the effectiveness of each of the 3 mechanisms of WocaR-RL in Section 5.2.

5 Experiments and Discussion

In this section, our experimental evaluations on various MuJoCo and Atari environments aim to study the following questions: **(1)** Can WocaR-RL learn policies with better **robustness** under existing strong adversarial attacks? **(2)** Can WocaR-RL maintain **natural performance** when improving robustness? **(3)** Can WocaR-RL learn more **efficiently** during robust training? **(4)** Is each mechanism in WocaR-RL **effective**? Problem (1), (2) and (3) are answered in Section 5.1 with detailed empirical results, and problem (4) is studied in Section 5.2 via ablation experiments.

5.1 Experiments and Evaluations

Environments. Following most prior works [52, 50, 32] and the released implementation, we apply our WocaR-RL to PPO [38] on 4 MuJoCo tasks with continuous action spaces, including Hopper, Walker2d, Halfcheetah and Ant, and to DQN [31] agents on 4 Atari games including Pong, Freeway, BankHeist and RoadRunner, which have high dimensional pixel inputs and discrete action spaces.

Baselines and Implementation. We compare our algorithm with several state-of-the-art robust training methods, including (1) *SA-PPO/SA-DQN* [52]: regularizing policy networks by convex relaxation. (2) *ATLA-PPO* [50]: alternately training an agent and an RL attacker. (3) *PA-ATLA-PPO* [41]: alternately training an agent and a more advanced RL attacker PA-AD. (4) *RADIAL-PPO/RADIAL-DQN* [32]: optimizing policy network by designed adversarial loss functions based on robustness bounds. SA and RADIAL have both PPO and DQN versions, which are compared with our WocaR-PPO and WocaR-DQN. But ATLA and PA-ATLA do not provide DQN versions, since alternately training on DQN can be expensive as explained in the original papers [41]. (PA-ATLA has an A2C version, which we compare in Appendix D.2.) Therefore, we reproduce their ATLA-PPO and PA-ATLA-PPO results and compare them with our WocaR-PPO. More implementation and hyperparameter details are provided in Appendix D.1.

Case I: Robust PPO for MuJoCo Continuous Control

Evaluation Metrics. To reflect both the natural performance and robustness of trained agents, we report the average episodic rewards under no attack and against various attacks. For a comprehensive robustness evaluation, we attack the trained robust models with multiple existing attack methods, including: (1) *MaxDiff* [52] (maximal action difference), (2) *Robust Sarsa (RS)* [52] (attacking with a robust action-value function), (3) *SA-RL* [52] (finding the optimal state adversary) and (4) *PA-AD* [41] (the *existing strongest attack* by learning the optimal policy adversary with RL). For a clear comparison, we use the same attack radius ϵ as in most baselines [52, 50, 41].

Performance and Robustness of WocaR-PPO Figure 5 (left four columns) shows performance curves during training under four different adversarial attacks. Among all four attack algorithms, WocaR-PPO converges much faster than baselines, and often achieves the best asymptotic robust performance, especially under the strongest PA-AD attack. It is worth emphasizing that since we train a robust agent without explicitly learning an RL attacker, our method not only obtains stronger robustness and much higher efficiency, but also a more general defense: WocaR-PPO obtains comprehensively superior performance against a variety of attacks compared against existing SOTA algorithms based on learned attackers (ATLA-PPO, PA-ATLA-PPO). Additionally, in our experiments, WocaR-PPO learns relatively more universal defensive behaviors as shown in Figure 2, which can physically explain why our algorithm can defend against diverse attacks. We provide policy demonstrations in multiple tasks in our supplementary materials.
The comparison of natural performance and the worst-case performance appears in Figure 5 (right). We see that WocaR-PPO maintains competitive natural rewards under no attack compared with other baselines, which demonstrates that our algorithm gains more robustness without losing too much natural performance. The full results of baselines and our algorithm under different attack evaluations are provided by Table 2 in Appendix D.2 (including performance under random attacks).

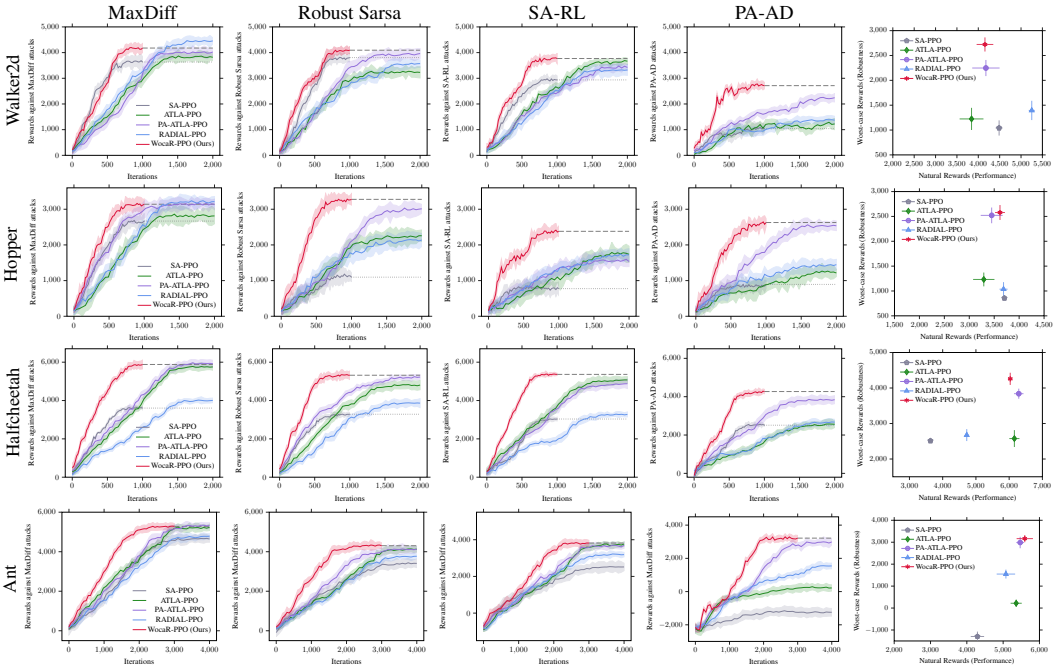


Figure 5: Robustness, Efficiency and High Natural Performance of WocaR-PPO. (Left four columns) Learning curves of rewards under MaxDiff, Robust Sarsa, SA-RL and PA-AD *(the strongest)* attacks during training on four environments. **(Rightmost column)** Average episode natural rewards v.s. average worst rewards under attacks. Each row shows the performance of baselines and WocaR-PPO on one environment. Shaded regions are computed over 20 random seeds. Results under more attack radius ϵ's are in Appendix D.3.1.

Efficiency of Training WocaR-PPO. The learning curves in Figure 5 (left) directly show the sample efficiency of WocaR-PPO. Following the optimal settings provided in [52, 50, 32], our method takes *50% training steps* required by RADIAL-PPO and ATLA methods on Hopper, Walker2d, and Halfcheetah because RADIAL-PPO needs more steps to ensure convergence and ATLA methods require additional adversary training steps. When solving high dimensional environments like Ant, WocaR-PPO only requires *75% steps* compared with all other baselines to converge. We also provide additional results of baselines using the same training steps as WocaR-PPO in Appendix D.3.2.
In terms of time efficiency, WocaR-PPO saves *50% training time* for convergence on Hopper, Walker2d, and Halfcheetah, and *32% time* on Ant compared with the SOTA method. Therefore, *WocaR-PPO achieves both higher computational efficiency and higher sample efficiency than SOTA baselines.* Detailed costs in time and sampling are in Appendix D.3.3.

Case II: Robust DQN for Atari Video Games

Evaluation Metrics. Since Atari games have pixel state spaces and discrete action spaces, the applicable attacking algorithms also differ from those in MuJoCo tasks. We include the following common attacks: (1) 10-step untargeted *PGD* (projected gradient descent) attack, (2) *MinBest* [18], which minimizes the probability of choosing the "best" action, (3) *PA-AD* [41], as the state-of-the-art RL-based adversarial attack algorithm.

Performance and Robustness of WocaR-DQN. Table 1 presents the results on four Atari games under attack radius $\epsilon = 3/255$, while results and analysis under smaller attack radius $1/255$ are in Appendix D.2. We can see that *our WocaR-DQN consistently outperforms baselines under MinBest and PA-AD attacks in all environments, with a significant advance under the strongest (worst-case) PA-AD attacks compared with other robust agents.* Under PGD attacks, WocaR-DQN performs comparably with the state-of-the-art in Freeway and Pong (which are simpler games) and gains higher rewards than other agents in BankHeist and Roadrunner. Since SA-DQN and RADIAL-DQN focus on bounding and smoothing the policy network and do not consider the policy's intrinsic vulnerability, they are robust under the PGD attack but still vulnerable against the stronger PA-AD attack.

Efficiency of Training WocaR-DQN. The total training time for SA-DQN, RADIAL-DQN, and our WocaR-DQN are roughly 35, 17, and 18 hours, respectively. All baselines are trained for 6 million frames on the same hardware. Therefore, WocaR-DQN is 49% faster (and is more robust) than SA-DQN. Compared to the more advanced baseline RADIAL-DQN, although WocaR-DQN is 5% slower, it achieves better robustness (539% higher reward than RADIAL-DQN in RoadRunner).

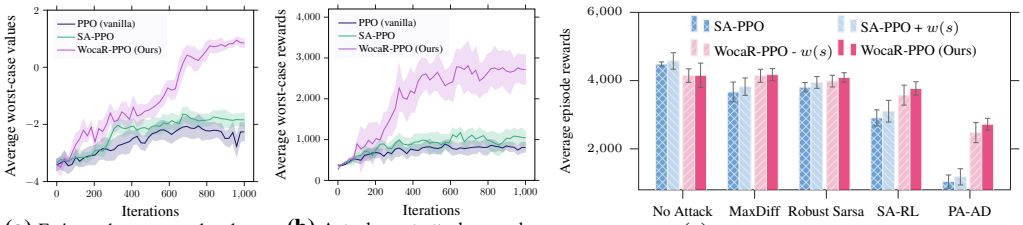
Model	Pong				BankHeist			
	Natural Reward	PGD	MinBest	PA-AD	Natural Reward	PGD	MinBest	PA-AD
		ϵ= 3/255				ϵ= 3/255		
DQN	21.0 ± 0.0	-21.0 ± 0.0	-9.7 ± 4.0	-19.0 ± 2.2	**1308 ± 24**	0 ± 0	119 ± 65	102 ± 92
SA-DQN	21.0 ± 0.0	21.0 ± 0.0	20.6 ± 3.5	18.7 ± 2.6	1245 ± 14	1176 ± 63	1024 ± 31	489 ± 106
RADIAL-DQN	21.0 ± 0.0	21.0 ± 0.0	19.5 ± 2.1	13.2 ± 1.8	1178 ± 4	1176 ± 63	928 ± 113	508 ± 85
WocaR-DQN (Ours)	**21.0 ± 0.0**	**21.0 ± 0.0**	**20.8 ± 3.3**	**19.7 ± 2.4**	1220 ± 12	**1214 ± 7**	**1045 ± 20**	**754 ± 102**

Model	Freeway				RoadRunner			
	Natural Reward	PGD	MinBest	PA-AD	Natural Reward	PGD	MinBest	PA-AD
		ϵ= 3/255				ϵ= 3/255		
DQN	**34.0 ± 0.1**	0.0 ± 0.0	5.5 ± 1.8	4.7 ± 2.9	**45527 ± 4894**	0 ± 0	2985 ± 1440	203 ± 65
SA-DQN	30.0 ± 0.0	30.0 ± 0.0	18.3 ± 3.0	9.5 ± 3.8	44638 ± 2367	20678 ± 1563	4214 ± 2587	5516 ± 4684
RADIAL-DQN	33.1 ± 0.2	**33.2 ± 0.2**	16.4 ± 2.3	10.8 ± 3.6	44675 ± 5854	38576 ± 1960	8476 ± 3964	1290 ± 4015
WocaR-DQN (Ours)	31.2 ± 0.4	31.4 ± 0.3	**19.8 ± 3.8**	**12.3 ± 3.2**	44156 ± 2279	**38720 ± 1765**	**10545 ± 2984**	**8239 ± 2766**

Table 1: Robustness and High Natural Performance of WocaR-DQN. Average episode rewards ± standard deviation over 50 episodes on three baselines and WocaR-DQN on four Atari environments. Best results (natural reward of under attacks for each column) on each environment boldfaced. WocaR-DQN outperforms all the baselines in most cases or gains similar performance in the other metrics. We highlight the most robust agent as gray. Each result is obtained with 10 random seeds.

5.2 Verifying Effectiveness of WocaR-RL

Now we dive deeper into the algorithmic design and verify the effectiveness of WocaR-RL by ablation studies on WocaR-PPO.



(a) Estimated worst-attack values **(b)** Actual worst-attack rewards **(c)** Ablation study for $w(s)$

Figure 6: (a)&(b) Comparison of the estimated worst-attack action value $\underline{Q}^{\pi}_{\phi}$ and the worst rewards under strongest attack during training on Walker2d; **(c)** Ablation evaluations for state importance weight $w(s)$ under no attack and four different attacks on Walker2d.

(1) Worst-attack value estimation. We show the learned worst-attack value estimation, $\underline{Q}^{\pi}_{\phi}$, during the training process in Figure 9c, in comparison with the actual reward under the strongest attack (PA-AD [41]) in Figure 9b. The pink curves in both plots suggest that *our worst-attack value estimation matches the trend of actual worst-case reward under attacks,* although the network estimated value and the real reward have different scales due to the commonly-used reward normalization for learning stability. Therefore, the proposed worst-attack value estimation ($\mathcal{L}_{\text{est}}$) is effective in the experiments.

(2) Worst-case-aware policy optimization. Compared to vanilla PPO and SA-PPO, we can see that *WocaR-PPO improves the worst-attack value and the worst-case reward during training*, suggesting the effectiveness of our worst-attack value improvement ($\mathcal{L}_{\text{wst}}$). The comparison of natural rewards, as well as curves in other environments, are provided in Appendix D.3.4. Moreover, the adjustable weight κ_{wst} in Equation (8) controls the trade-off between natural value and worst-attack value in policy optimization. When κ_{wst} is high, the policy pays more attention to its worst-attack value. Appendix D.3.5 verifies that *WocaR-RL, with different values of weight κ_{wst}, produces different robustness and natural performance while consistently dominating other robust agents.*

(3) Value-enhanced state regularization. We compare the performance of the original WocaR-PPO to variants of WocaR-PPO without the state importance weight $w(s)$ in Figure 6c. The results of SA-PPO with $w(s)$ also indicate the effectiveness of the state importance. Detailed ablation studies for more environments are shown in Appendix D.3.6, which verify that the weighted state regularization is beneficial to enhancing the robustness of RL agents. Without $w(s)$, our algorithm also achieves similar or better performance than baselines, but including this inexpensive technique $w(s)$ gives WocaR-RL a greater advantage, especially under learned strong attacks SA-RL and PA-AD.

6 Conclusion

This paper proposes a robust RL training framework, WocaR-RL, that evaluates and improves the long-term robustness of a policy via worst-attack value estimation, worst-case-aware policy optimization, value-enhanced state regularization. Compared with state-of-the-art robust RL approaches, WocaR-RL achieves better robustness, efficiency and interpretability in experiments.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[2] Sarah Bechtle, Yixin Lin, Akshara Rai, Ludovic Righetti, and Franziska Meier. Curious ilqr: Resolving uncertainty in model-based rl. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 162–171. PMLR, 30 Oct–01 Nov 2020.

[3] 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.

[4] Vahid Behzadan and Arslan Munir. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017.

[5] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

[6] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[7] Robert Dadashi, Adrien Ali Taiga, Nicolas Le Roux, Dale Schuurmans, and Marc G Bellemare. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pages 1486–1495. PMLR, 2019.

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

[9] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[10] Chris Gaskett. Reinforcement learning under circumstances beyond its control. In *International Conference on Computational Intelligence for Modelling Control and Automation*, 2003.

[11] Saul B Gelfand and Sanjoy K Mitter. Recursive stochastic algorithms for global optimization in rˆd. *SIAM Journal on Control and Optimization*, 29(5):999–1018, 1991.

[12] 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.

[13] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[14] Sven Gowal, Krishnamurthy Dj Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4842–4851, 2019.

[15] Arthur Guez and H Van Hasselt. Deep reinforcement learning with double q-learning. *Association for the Advancement of Artificial Intelligence*, 2015.

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

[17] Matthias Heger. Consideration of risk in reinforcement learning. In *International Conference on Machine Learning*, 1994.

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

[19] Yunhan Huang and Quanyan Zhu. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International Conference on Decision and Game Theory for Security*, pages 217–237. Springer, 2019.

[20] Ezgi Korkmaz. Investigating vulnerabilities of deep neural policies. In *Uncertainty in Artificial Intelligence*, pages 1661–1670. PMLR, 2021.

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

[22] Aounon Kumar, Alexander Levine, and Soheil Feizi. Policy smoothing for provably robust reinforcement learning. *arXiv preprint arXiv:2106.11420*, 2021.

[23] 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.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[25] Shiau Hong Lim, Huan Xu, and Shie Mannor. Reinforcement learning in robust markov decision processes. *Advances in Neural Information Processing Systems*, 26:701–709, 2013.

[26] 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.

[27] 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.

[28] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3932–3939. IEEE, 2017.

[29] Daniel J. Mankowitz, Nir Levine, Rae Jeong, Abbas Abdolmaleki, Jost Tobias Springenberg, Yuanyuan Shi, Jackie Kay, Todd Hester, Timothy Mann, and Martin Riedmiller. Robust reinforcement learning for continuous control with model misspecification. In *International Conference on Learning Representations*, 2020.

[30] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3578–3586. PMLR, 10–15 Jul 2018.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[32] Tuomas Oikarinen, Wang Zhang, Alexandre Megretski, Luca Daniel, and Tsui-Wei Weng. Robust deep reinforcement learning through adversarial loss. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[33] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

[34] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[35] Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pages 7974–7984, 2020.

[36] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[37] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

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

[39] Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.

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

[41] Yanchao Sun, Ruijie Zheng, Yongyuan Liang, and Furong Huang. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087*, 2021.

[42] Aviv Tamar, Huan Xu, and Shie Mannor. Scaling up robust mdps by reinforcement learning. *arXiv preprint arXiv:1306.6189*, 2013.

[43] 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.

[44] 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.

[45] Garrett Thomas, Yuping Luo, and Tengyu Ma. Safe reinforcement learning by imagining the near future. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[46] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[47] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[48] Fan Wu, Linyi Li, Zijian Huang, Yevgeniy Vorobeychik, Ding Zhao, and Bo Li. Crop: Certifying robust policies for reinforcement learning through functional smoothing. *arXiv preprint arXiv:2106.09292*, 2021.

[49] 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.

[50] 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.

[51] Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.

[52] 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.

[53] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4944–4953, Red Hook, NY, USA, 2018. Curran Associates Inc.

[54] Xuezhou Zhang, Yuzhe Ma, Adish Singla, and Xiaojin Zhu. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See our abstract and Section 1.
   (b) Did you describe the limitations of your work? [Yes] We discuss the limitations in Appendix E.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We discuss the potential negative societal impacts in Appendix E.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our paper conforms to the guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix A.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] For each empirical result, we report error bars.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See efficiency analysis in Appendix D.3.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]