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

Yongyuan Liang [1] * **Yanchao Sun** [2] * **Ruijie Zheng** [2] **Furong Huang** [2]

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 (Akhtar & Mian, 2018; Chakraborty et al., 2018). 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 (Sun et al., 2021b).

A line of regularization-based robust methods (Zhang et al., 2020b; Oikarinen et al., 2021; Shen et al., 2020) 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 (Zhang et al., 2021). These regularization-based methods ignore the intrinsic vulnerability of policies under the environment dynamics, and thus may still fail under strong attacks (Sun et al., 2021b), as an example we provided in Appendix C.2. 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.

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 (Zhang et al., 2020b; Sun et al., 2021b) 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. (2021) propose the *Alternating Training with Learned Adversaries (ATLA)* framework, which alternately trains an RL agent and an RL attacker. Sun et al. (2021b) further propose PA-ATLA trained with a more efficient PA-AD RL attacker, obtaining state-of-the-art robustness on 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 (Zhang et al., 2021; Sun et al., 2021b), especially with high-dimensional observations such as images.

The above analysis of existing literature suggests two main challenges in improving the adversarial robustness of DRL agents: (1) correctly characterizing the long-term vulnerability of an RL policy, and (2) efficiently training a robust agent without requiring much more effort than vanilla training. To tackle these challenges, 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.

Our **contributions** can be summarized as below. **(1)** We provide an approach to estimate the worst-case value of any

*Equal contribution [1]Sun Yat-sen University, China [2]University of Maryland, College Park, USA. Correspondence to: Yongyuan Liang <liangyy58@mail2.sysu.edu.cn>.

ICML 2022 Workshop on Responsible Decision Making in Dynamic Environments, Baltimore, Maryland, USA, 2022.

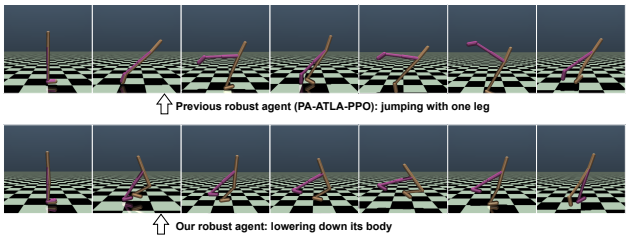


Figure 1. The robust Walker agents trained with **(top)** the state-of-the-art method PA-ATLA-PPO (Sun et al., 2021b) and **(bottom)** our WocaR-RL. Although PA-ATLA-PPO agent also achieves high reward under attacks, it learns to jump with one leg, which is counter-intuitive and may indicate some level 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 are provided in supplementary GIF figures.

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 (Schulman et al., 2017), DQN (Mnih et al., 2013)). **(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 (Sun et al., 2021b) 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 1.

2. Preliminaries and Background

Adversarial Reinforcement Learning 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 $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. 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. An attacker/adversary, during the deployment of the agent, may 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. We consider the ℓ_p *threat 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 as many prior works (Huang et al., 2017; Pattanaik et al., 2017; Zhang et al., 2020b; 2021; Sun et al., 2021b).

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

3. Worst-case-aware Robust RL

In this section, we present *Worst-case-aware Robust RL (WocaR-RL)*, a generic framework that consists of three key mechanisms and can be combined with any DRL approach to improve the adversarial robustness of an agent.

Mechanism 1: Worst-attack Value Estimation

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 2).

The worst-case value of a policy is, by definition, the cumulative reward obtained under the optimal attacker. As justified by prior works (Zhang et al., 2020b; Sun et al., 2021b), 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^*.

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

Definition 3.1 (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

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

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

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

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 3.2 (Worst-attack Bellman Operator and Worst-attack Action Value)**.** *For any given policy π, $\underline{\mathcal{T}}^\pi$ is a con-*

traction whose fixed point is $\underline{Q}^{\pi}$, the worst-attack action value of π under any ℓ_p observation attacks with radius ϵ.

Theorem 3.2 proved in Appendix B 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 $\underline{V}^{\pi}(s) = \min_{a \in \mathcal{A}_{\text{adv}}(s,\pi)} \underline{Q}^{\pi}(s,a)$. we discuss the differences with worst-case values in risk-sensitive RL in Appendix D.6. **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 TD errors. We train a network to model the worst-attack action value, with the standard Q network architecture (Mnih et al., 2013). This network is named as the *worst-attack critic*, denoted by $\underline{Q}^{\pi}_{\phi}$, where ϕ is the parameterization. Similar to the critic function that models the natural Q value of the current behavior policy π, our $\underline{Q}^{\pi}_{\phi}$ models the worst-attack action value of π. 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:

$$\mathcal{L}_{\text{est}}(\underline{Q}^{\pi}_{\phi}) := \frac{1}{N} \sum_{t=1}^{N} (\underline{y}_t - \underline{Q}^{\pi}_{\phi}(s_t, a_t))^2 \quad (3)$$

$$\text{where } \underline{y}_t = r_t + \gamma \min_{\hat{a} \in \mathcal{A}_{\text{adv}}(s_{t+1},\pi)} \underline{Q}^{\pi}_{\phi}(s_{t+1}, \hat{a}). \quad (4)$$

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. That is, we calculate $\overline{\pi}$ and $\underline{\pi}$ such that $\overline{\pi}(s) \geq \pi(\hat{s}) \geq \underline{\pi}(s), \forall \hat{s} \in \mathcal{B}_{\epsilon}(s)$. Let $\hat{\mathcal{A}}_{\text{adv}}$ be the approximation of $\mathcal{A}_{\text{adv}}$. 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 derive the maximal and minimal probabilities of taking action a under $\hat{s} \in \mathcal{B}_{\epsilon}(s)$ by convex relaxation, and then $\hat{\mathcal{A}}_{\text{adv}}$ contains actions that is likely to be selected. More explanations are provided in Appendix D.1.

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. To encourage the agent to select an action with a higher worst-attack action value, we minimize the worst-attack policy loss below:

$$\mathcal{L}_{\text{wst}}(\pi_\theta; \underline{Q}^{\pi}_{\phi}) := -\frac{1}{N} \sum_{t=1}^{N} \sum_{a \in \mathcal{A}} \pi_\theta(a|s_t) \underline{Q}^{\pi}_{\phi}(s_t, a), \quad (5)$$

where $\underline{Q}^{\pi}_{\phi}$ is the worst-attack critic learned via $\mathcal{L}_{\text{est}}$ introduced in Equation (4). 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). We also compare worst-case-aware policy optimization with ATLA methods in Appendix D.7.

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. Different from prior regularization methods, we note that some states are "critical" where selecting a bad action will result in catastrophic consequences. To differentiate states based on their impacts on future reward, we propose to measure the importance of states with Definition 3.3 below.

Definition 3.3 (State Importance Weight)**.** Define state importance weight of $s \in \mathcal{S}$ for policy π as

$$w(s) = \max_{a_1 \in \mathcal{A}} Q^{\pi}(s, a_1) - \min_{a_2 \in \mathcal{A}} Q^{\pi}(s, a_2). \quad (6)$$

To justify why Definition 3.3 can characterize state importance, we provide an example on an Atari game Pong in Appendix C.3. 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:

$$\mathcal{L}_{\text{reg}}(\pi_\theta) = \frac{1}{N} \sum_{t=1}^{N} w(s_t) \max_{\tilde{s}_t \in \mathcal{B}_{\epsilon}(s_t)} \text{Dist}(\pi_\theta(s_t), \pi_\theta(\tilde{s}_t)), \quad (7)$$

where Dist can be any distance measure between two probability 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 (4), $\mathcal{L}_{\text{reg}}$ in Equation (5) and $\mathcal{L}_{\text{reg}}$ in Equation (7). Then, our robust training framework WocaR-RL combines these losses with any base RL algorithm. To be more specific, as shown in Figure 2, 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

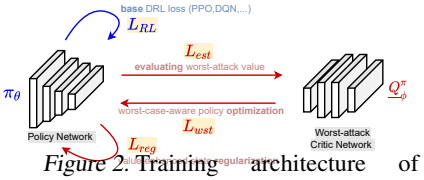


Figure 2. Training architecture of WocaR-RL. (Components proposed in this paper are colored as red.)

$$\mathcal{L}_{\underline{Q}^{\pi}_{\phi}} := \mathcal{L}_{\text{est}}(\underline{Q}^{\pi}_{\phi}), \quad (8)$$

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

$$\mathcal{L}_{\pi_\theta} := \mathcal{L}_{\text{RL}}(\pi_\theta) + \kappa_{\text{wst}} \mathcal{L}_{\text{wst}}(\pi_\theta; \underline{Q}^{\pi}_{\phi}) + \kappa_{\text{reg}} \mathcal{L}_{\text{reg}}(\pi_\theta), \quad (9)$$

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 be interpreted from a geometric perspective based on the previous RL polytope theory (Dadashi et al., 2019; Sun et al., 2021b) as in Ap-

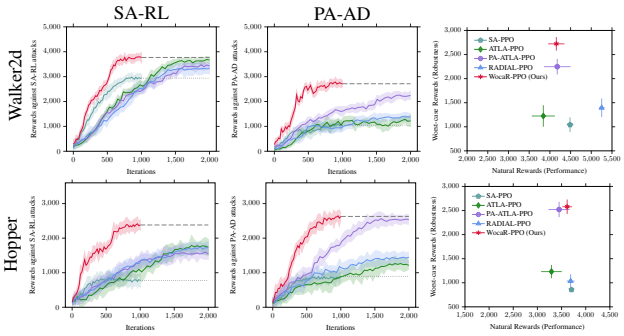


Figure 3. **Robustness, Efficiency and High Natural Performance of WocaR-PPO. (Left two columns)** Learning curves of rewards under SA-RL and PA-AD *(the strongest)* attacks during training. **(Rightmost column)** Average episode natural rewards v.s. average worst rewards under attacks. Each row shows 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 E.3.1.

pendix C. Our WocaR-RL is a generic robust training framework that can be used to robustify existing DRL algorithms: combining with policy-based PPO, namely *WocaR-PPO*, and combining WocaR-RL with value-based DQN, namely *WocaR-DQN*. The pseudocodes of WocaR-PPO and WocaR-DQN are illustrated in Appendix D.2 and Appendix D.3.

4. 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 4.1 and problem (4) is studied in Appendix E.3 via ablation studies on WocaR-PPO. The empirical analysis for Atari games are in Appendix E.2.2.

4.1. Experiments and Evaluations

Environments. Following most prior works (Zhang et al., 2020b; 2021; Oikarinen et al., 2021) and the released implementation, we apply our WocaR-RL to PPO (Schulman et al., 2017) on 4 MuJoCo tasks with continuous action spaces, including Hopper, Walker2d, Halfcheetah and Ant.

Baselines and Implementation. We compare our algorithm with several state-of-the-art robust training methods, including *SA-PPO* (Zhang et al., 2020b), *ATLA-PPO* (Zhang et al., 2021), *PA-ATLA-PPO* (Sun et al., 2021b) and *RADIAL-PPO* (Oikarinen et al., 2021). To reflect both the natural performance and robustness of trained agents, we report the average episodic rewards under no attack and against various attacks including: *Random*, *MaxDiff* (Zhang et al., 2020b), *Robust Sarsa (RS)* (Zhang et al., 2020b), *SA-RL* (Zhang et al., 2020b) and *PA-AD* (Sun et al., 2021b). For a clear comparison, we use the same attack radius ϵ as in most baselines (Zhang et al., 2020b; 2021; Sun et al., 2021b). Baseline, implementation and hyperparameter details are provided in Appendix E.1.

Performance and Robustness of WocaR-PPO Figure 3 (left two columns) shows performance curves during training under the RL-based strongest attacks. WocaR-PPO converges much faster than baselines, and often achieves the best asymptotic robust performance. 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, WocaR-PPO learns relatively more universal defensive behaviors as shown in Figure 1, which can physically explain why our algorithm can defend against diverse attacks.
The comparison of natural performance and the worst-case performance appears in Figure 3 (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 on four environments are provided in Appendix E.2.1.

Efficiency of Training WocaR-PPO. The learning curves in Figure 7 (left) directly show the sample efficiency of WocaR-PPO. Following the optimal settings provided in (Zhang et al., 2020b; 2021; Oikarinen et al., 2021), our method takes *50% steps* required by RADIAL-PPO and ATLA methods, because RADIAL-PPO needs more steps to ensure convergence and ATLA methods require additional adversary training steps. In terms of time efficiency, WocaR-PPO saves more than *50% training time* for convergence 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 and additional results of baselines using the same training steps as WocaR-PPO are in Table 4.

5. 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

Akhtar, N. and Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018. doi: 10.1109/ACCESS.2018.2807385.

Bechtle, S., Lin, Y., Rai, A., Righetti, L., and Meier, F. Curious ilqr: Resolving uncertainty in model-based rl. In Kaelbling, L. P., Kragic, D., and Sugiura, K. (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 162–171. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/bechtle20a.html.

Behzadan, V. and Munir, A. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 262–275. Springer, 2017a.

Behzadan, V. and Munir, A. Whatever does not kill deep reinforcement learning, makes it stronger. *arXiv preprint arXiv:1712.09344*, 2017b.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Dadashi, R., Taiga, A. A., Le Roux, N., Schuurmans, D., and Bellemare, M. G. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pp. 1486–1495. PMLR, 2019.

Fischer, M., Mirman, M., Stalder, S., and Vechev, M. Online robustness training for deep reinforcement learning. *arXiv preprint arXiv:1911.00887*, 2019.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

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

Gelfand, S. B. and Mitter, S. K. Recursive stochastic algorithms for global optimization in r^d. *SIAM Journal on Control and Optimization*, 29(5):999–1018, 1991.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Gowal, S., Dvijotham, K. D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4842–4851, 2019.

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

Hado Van Hasselt, Arthur Guez, D. S. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Heger, M. Consideration of risk in reinforcement learning. In *International Conference on Machine Learning*, 1994.

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

Huang, Y. and Zhu, Q. Deceptive reinforcement learning under adversarial manipulations on cost signals. In *International Conference on Decision and Game Theory for Security*, pp. 217–237. Springer, 2019.

Korkmaz, E. Investigating vulnerabilities of deep neural policies. In *Uncertainty in Artificial Intelligence*, pp. 1661–1670. PMLR, 2021.

Kos, J. and Song, D. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

Kumar, A., Levine, A., and Feizi, S. Policy smoothing for provably robust reinforcement learning. *arXiv preprint arXiv:2106.11420*, 2021.

Lee, X. Y., Esfandiari, Y., Tan, K. L., and Sarkar, S. 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, pp. 87–97, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383530.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lim, S. H., Xu, H., and Mannor, S. Reinforcement learning in robust markov decision processes. *Advances in Neural Information Processing Systems*, 26:701–709, 2013.

Lütjens, B., Everett, M., and How, J. P. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*, pp. 1328–1337. PMLR, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Mandlekar, A., Zhu, Y., Garg, A., Fei-Fei, L., and Savarese, S. Adversarially robust policy learning: Active construction of physically-plausible perturbations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939. IEEE, 2017.

Mankowitz, D. J., Levine, N., Jeong, R., Abdolmaleki, A., Springenberg, J. T., Shi, Y., Kay, J., Hester, T., Mann, T., and Riedmiller, M. Robust reinforcement learning for continuous control with model misspecification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgC60EtwB.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3578–3586. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/mirman18b.html.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Oikarinen, T., Zhang, W., Megretski, A., Daniel, L., and Weng, T.-W. Robust deep reinforcement learning through adversarial loss. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=eaAM_bdW0Q.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826. PMLR, 2017.

Rakhsha, A., Radanovic, G., Devidze, R., Zhu, X., and Singla, A. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, pp. 7974–7984, 2020.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shen, Q., Li, Y., Jiang, H., Wang, Z., and Zhao, T. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pp. 8707–8718. PMLR, 2020.

Sun, Y., Huo, D., and Huang, F. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. In *International Conference on Learning Representations*, 2021a.

Sun, Y., Zheng, R., Liang, Y., and Huang, F. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep rl. *arXiv preprint arXiv:2106.05087*, 2021b.

Tamar, A., Xu, H., and Mannor, S. Scaling up robust mdps by reinforcement learning. *arXiv preprint arXiv:1306.6189*, 2013.

Tan, K. L., Esfandiari, Y., Lee, X. Y., Sarkar, S., et al. Robustifying reinforcement learning agents via action space adversarial training. In *2020 American control conference (ACC)*, pp. 3959–3964. IEEE, 2020.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pp. 6215–6224. PMLR, 2019.

Thomas, G., Luo, Y., and Ma, T. Safe reinforcement learning by imagining the near future. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=vIDBSGl3vzl.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2018.

Wu, F., Li, L., Huang, Z., Vorobeychik, Y., Zhao, D., and Li, B. Crop: Certifying robust policies for reinforcement learning through functional smoothing. *arXiv preprint arXiv:2106.09292*, 2021.

Xiao, C., Pan, X., He, W., Peng, J., Sun, M., Yi, J., Liu, M., Li, B., and Song, D. Characterizing attacks on deep reinforcement learning. *arXiv preprint arXiv:1907.09470*, 2019.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020a. URL https://openreview.net/forum?id=Skxuk1rFwB.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., and Hsieh, C.-J. Robust deep reinforcement learning against adversarial perturbations on state observations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21024–21037. Curran Associates, Inc., 2020b. URL https://proceedings.neurips.cc/paper/2020/file/f0eb6568ea114ba6e293f903c34d7488-Paper.pdf.

Zhang, H., Chen, H., Boning, D., and Hsieh, C.-J. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452*, 2021.

Zhang, X., Ma, Y., Singla, A., and Zhu, X. Adaptive reward-poisoning attacks against reinforcement learning. In *International Conference on Machine Learning*, 2020c.

A. Related Work

Defending against Adversarial Perturbations on State Observations. **(1)** *Regularization-based methods* (Zhang et al., 2020b; Shen et al., 2020; Oikarinen et al., 2021) 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 (Korkmaz, 2021) 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 (Kos & Song, 2017; Behzadan & Munir, 2017b; Mandlekar et al., 2017; Pattanaik et al., 2017) apply weak or strong gradient-based attacks on state observations to train RL agents against adversarial perturbations. Zhang et al. (Zhang et al., 2021) 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. (Sun et al., 2021b) 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 (Lütjens et al., 2020; Oikarinen et al., 2021; Fischer et al., 2019) 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.(Kumar et al., 2021; Wu et al., 2021) both extend randomized smoothing (Cohen et al., 2019) 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 (Xiao et al., 2019; Tan et al., 2020; Tessler et al., 2019; Lee et al., 2021). Moreover, in a multi-agent game, an agent's behavior can create adversarial perturbations to a victim agent (Gleave et al., 2020). Pinto et al. (Pinto et al., 2017) 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 D.5. Note that we focus on robustness against test-time attacks, different from poisoning attacks which alter the RL training process (Behzadan & Munir, 2017a; Huang & Zhu, 2019; Sun et al., 2021a; Zhang et al., 2020c; Rakhsha et al., 2020).

Safe RL and Risk-sensitive RL. There are several lines of work that study RL under safety/risk constraints (Heger, 1994; Gaskett, 2003; Garcıa & Fernández, 2015; Bechtle et al., 2020; Thomas et al., 2021) or under intrinsic uncertainty of environment dynamics (Lim et al., 2013; Mankowitz et al., 2020). 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 3.

B. Theoretical Analysis

Similar to the worst-attack action value, we can define the worst-attack value as below

Definition B.1 (Worst-attack Value)**.** For a given policy π, define the worst-attack value of π as

$$\underline{V}^{\pi}(s) := \mathbb{E}_P[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, \pi(h^*(s_t))\right) \mid s_0 = s], \tag{10}$$

where h^* is the optimal attacker which minimizes the victim's cumulative reward under the ϵ constraint.

Proof of Theorem 3.2. First, we show that $\underline{\mathcal{T}}^{\pi}$ is a contraction.

For any two Q functions $Q_1 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $Q_2 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, we have

$$\begin{aligned}
&\|\underline{\mathcal{T}}^{\pi} Q_1 - \underline{\mathcal{T}}^{\pi} Q_2\|_{\infty} \\
&= \max_{s,a} \left| \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \left[R(s,a) + \gamma \min_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} Q_1\left(s', a'\right) - R(s,a) + \gamma \min_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} Q_2\left(s', a'\right) \right] \right| \\
&= \gamma \max_{s,a} \left| \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \left[\min_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} Q_1\left(s', a'\right) - \min_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} Q_2\left(s', a'\right) \right] \right| \\
&\leq \gamma \max_{s,a} \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \left| \min_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} Q_1\left(s', a'\right) - \min_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} Q_2\left(s', a'\right) \right| \\
&\leq \gamma \max_{s,a} \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \max_{a' \in \mathcal{A}_{\mathrm{adv}}(s', \pi)} \left| Q_1\left(s', a'\right) - Q_2\left(s', a'\right) \right| \\
&= \gamma \max_{s,a} \sum_{s' \in \mathcal{S}} P\left(s' \mid s, a\right) \|Q_1 - Q_2\|_{\infty} \\
&= \gamma \|Q_1 - Q_2\|_{\infty}
\end{aligned}$$

The second inequality comes from the fact that,

$$\left| \min_{x_1} f(x_1) - \min_{x_2} g(x_2) \right| \leq \max_{x} |f(x) - g(x)|$$

The operator $\underline{\mathcal{T}}^{\pi}$ satisfies,

$$\|\underline{\mathcal{T}}^{\pi} Q_1 - \underline{\mathcal{T}}^{\pi} Q_2\|_{\infty} \leq \gamma \|Q_1 - Q_2\|_{\infty}$$

so it is a contraction in the sup-norm.

Recall the definition of worst-attack action value:

$$\underline{Q}^{\pi}(s,a) := \mathbb{E}_P[\sum_{t=0}^{\infty} \gamma^t R\left(s_t, \pi(h^*(s_t))\right) \mid s_0 = s, a_0 = a], \tag{11}$$

where h^* is the optimal attacker which minimizes the victim's cumulative reward under the ϵ constraint. That is, the optimal attacker h^* lets the agent select the worst possible action among all achievable actions in $\mathcal{A}_{\mathrm{adv}}$. Hence, we have $\underline{Q}^{\pi}(s,a) = \underline{\mathcal{T}}^{\pi} \underline{Q}^{\pi}(s,a)$. Therefore, $\underline{Q}^{\pi}(s,a)$ is the fixed point of the Bellman operator $\underline{\mathcal{T}}^{\pi}$.

□

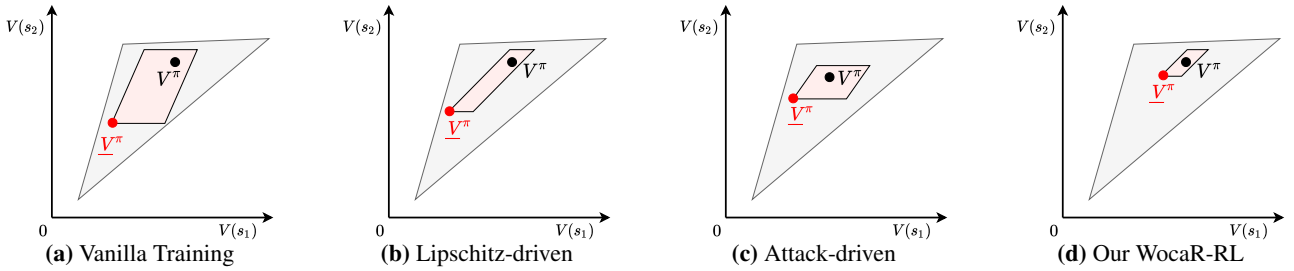
C. Geometric Understanding of WocaR-RL

C.1. A Closer Look at Robust RL

In real-world applications where observations may be noisy or perturbed, it is important to ensure that the agent not only makes good decisions, but also makes safe decisions.

Existing Robust RL Approaches. There are many existing robust training methods for RL, and we summarize the common ideas as the following two categories.

(1) *Lipschitz-driven methods:* encourage the policy to output similar actions for any pair of clean state and perturbed state, i.e., $\min_{\theta} \max_{s \in \mathcal{S}, \tilde{s} \in \mathcal{B}_{\epsilon}(s)} \mathsf{Dist}(\pi_{\theta}(s), \pi_{\theta}(\tilde{s}))$, where Dist can be any distance metric. Therefore, the policy function (network) has small local Lipschitz constant at each clean state. Note that this idea is similar to many certifiable robust



(a) Vanilla Training **(b)** Lipschitz-driven **(c)** Attack-driven **(d)** Our WocaR-RL

Figure 4. Geometric understanding of different training methods following the polytope theory by (Dadashi et al., 2019) and (Sun et al., 2021b). x, y axes represent the policy value for $s_1 \in \mathcal{S}$ and $s_2 \in \mathcal{S}$. The grey polytope depicts the value space of all policies, while the pink polytope (referred to as value perturbation polytope) contains the values of policy π under all attacks with given constraint (ϵ-radius ℓ_p perturbations on the state input to the policy). V^π denotes the value of a learned policy, and $\underline{V}^\pi$ stands for the worst-attack value of this policy π (located at the bottom leftmost vertex of the value perturbation polytope).
Two relations between the value perturbation polytope and policy robustness: The more distant the pink value perturbation polytope's bottom leftmost vertex is from the origin, the higher worst-attack value π has. The smaller the pink value perturbation polytope is, the less vulnerable the policy is (i.e., an ϵ-bounded state perturbation can not lead to a drastic change of the policy value).
Our method: WocaR-RL makes a policy more robust via worst-attack value estimation, worst-case-aware policy optimization and value-enhanced state regularization, which shrink the value perturbation polytope and move the value perturbation polytope's bottom leftmost vertex away from the origin.

training methods (Gowal et al., 2018) in supervised learning. For example, Fischer et al. (Fischer et al., 2019) achieve provable robustness for DQN by applying the DiffAI (Mirman et al., 2018) approach, so that the DQN agent selects the same action for any element inside $\mathcal{B}_\epsilon(s)$. Zhang et al. (Zhang et al., 2020b) propose to minimize the total variance between $\pi(s)$ and $\pi(\tilde{s})$ using convex relaxations of NNs. Although Lipschitz-driven methods are relatively efficient in training, they usually treat all states equally, and do not explicitly consider long-term rewards. Therefore, it is hard to obtain a non-vacuous reward certification, especially in continuous-action environments.

(2) *Attack-driven methods:* train the agent under adversarial attacks, which is analogous to Adversarial Training (AT) (Madry et al., 2018). However, different from AT, a PGD attacker may not induce a robust policy in an RL problem due to the uncertainty and complexity of the environment. Zhang et al. (Zhang et al., 2021) propose to alternately train an agent and an RL-based "optimal" adversary, so that the agent can adapt to the worst-case input perturbation. Therefore, attack-driven method can be formulated as $\max_\theta \underline{V}^{\pi_\theta}$. Zhang et al. (Zhang et al., 2021) and a follow-up work by Sun et al. (Sun et al., 2021b) apply the alternate training approach and obtain state-of-the-art robust performance. However, learning the optimal attacker using RL algorithms doubles the learning complexity and the required samples, making it hard to apply these methods to large-scale problems. Moreover, although these attack-driven methods improve the worst-case performance of an agent, the natural reward can be sacrificed.

Note that we discuss methods that improve the robustness of deep policies during training. Therefore, the focus is different from some important works (Lütjens et al., 2020; Wu et al., 2021; Kumar et al., 2021) that directly use non-robust policies and execute them in a robust way.

Our Motivation: Geometric Understanding of Robust RL. The robustness of a learned RL policy can be understood from a geometric perspective. Dadashi et al. (Dadashi et al., 2019) point out that the value functions of all policies in a finite MDP form a polytope, as shown by the grey area in Figure 4. Sun et al. (Sun et al., 2021b) further find that $V^{\tilde{\pi}}$, possible values of a policy π under all ϵ-constrained ℓ_p perturbations, also form a polytope (pink area in Figure 4), which we refer to as the *value perturbation polytope*. Recall that in robust RL, we pursue a high natural value V^π, and a high worst-case value $\underline{V}^\pi$ which is the lower leftmost vertex of the value perturbation polytope. A vulnerable policy that outputs a different action for a perturbed state as a larger value perturbation polytope. Lipschitz-driven methods, as Figure 4(a) shows, attempts to shrink the size of the value perturbation polytope, but does not necessarily result in a high $\underline{V}^\pi$. Attack-driven methods, as Figure 4 shows, improves $\underline{V}^\pi$, but have no control over the size of the value perturbation polytope, and may not obtain a high natural value V^π.

Our Proposed Robust RL Principle. In contrast to prior Lipschitz-driven methods and Attack-driven methods, we propose to both "lift the position" and "shrink the size" of the value perturbation polytope. To achieve the above principle in an efficient way, we propose to (1) directly estimate and optimize the worst-case value of a policy without training the optimal attacker (worst-attack value estimation and worst-case-aware policy optimization mechanisms of WocaR-RL), and

(2) regularize the local Lipschitz constants of the policy with value-enhanced weights (value-enhanced state regularization mechanism of WocaR-RL). See Section 3 for more details of the proposed algorithm.

C.2. An Example of Policy Vulnerabilities

In the go-home task shown in Figure 5, 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.

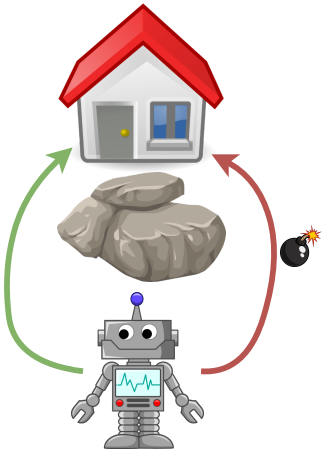
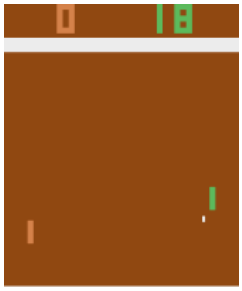


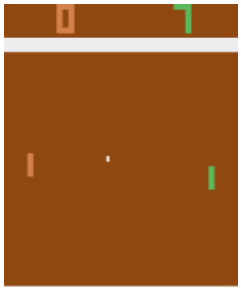
Figure 5. Policies have different vulnerabilities.

C.3. An Example of State Importance

To justify whether Definition 3.3 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 6, among many state samples. We can see that the state with higher weight in Figure 6a is indeed crucial for the game, as the green agent paddle is close to the ball. Conversely, a less-important state in Figure 6b 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.



(a) A state with high weight $w(s)$



(b) A state with low weight $w(s)$

Figure 6. Examples of different states and their importance weights $w(s)$ in Atari game Pong.

D. Algorithm Details

D.1. Computing $\mathcal{A}_{\text{adv}}$ by Network Bounding Techniques

Recall that $\mathcal{A}_{\text{adv}}(s,\pi) = \{a \in \mathcal{A} : \exists \tilde{s} \in \mathcal{B}_\epsilon(s) \text{ s.t. } \pi(\tilde{s}) = a\}$ is the set of actions that π may be misled to select in state s. Computing the exact $\mathcal{A}_{\text{adv}}$ is difficult due to the complexity of neural networks, so we use relaxations of network such as Interval Bound Propagation (IBP) (Wong & Kolter, 2018; Gowal et al., 2019) to approximately calculate $\mathcal{A}_{\text{adv}}$.

A Brief Introduction to Convex Relaxation Methods. Convex relaxation methods are techniques to bound a neural network that provide the upper and lower bound of the neural network output given a bounded l_p perturbation to the input. In particular, we take l_∞ as an example, which has been studied extensively in prior works. Formally, let f_θ be a real-valued function parameterized by a neural network θ, and let $f_\theta(s)$ denote the output of the neural network with the input s. Given an l_∞ perturbation budget ϵ, convex relaxation method outputs $(\underline{f_\theta(s)}, \overline{f_\theta(s)})$ such that

$$\underline{f_\theta(s)} \leq \min_{\|s'-s\|_\infty \leq \epsilon} f_\theta(s') \leq \max_{\|s'-s\|_\infty \leq \epsilon} f_\theta(s') \leq \overline{f_\theta(s)}$$

Recall that we use π_θ to denote the parameterized policy being trained that maps a state observation to a distribution over the action space, and π denotes the deterministic policy refined from π_θ with $\pi(s) = \text{argmax}_{a \in \mathcal{A}} \pi_\theta(a|s)$. $\mathcal{A}_{\text{adv}}(s, \pi)$ contains actions that could be selected by π (with the highest probability in π_θ's output) when s is perturbed within a ϵ-radius ball. Our goal is to approximately identify a superset of $\mathcal{A}_{\text{adv}}(s, \pi)$, i.e., $\hat{\mathcal{A}}_{\text{adv}}(s, \pi)$, via the convex relaxation of networks introduced above.

Computing $\mathcal{A}_{\text{adv}}$ in Continuous Action Space. The most common policy parameterization in a continuous action space is through a Gaussian distribution. Let $\mu_\theta(s)$ be the mean of Gaussian computed by $\pi_\theta(s)$, then $\pi = \mu(s)$. Therefore, we can use network relaxation to compute an upper bound and a lower bound of μ_θ with input $\mathcal{B}_\epsilon(s)$. Then, $\hat{\mathcal{A}}_{\text{adv}}(s, \pi) = [\underline{\mu_\theta(s)}, \overline{\mu_\theta(s)}]$, i.e., a set of actions that are coordinate-wise bounded by $\underline{\mu_\theta(s)}$ and $\overline{\mu_\theta(s)}$. For other continuous distributions, e.g., Beta distribution, the computation is similar, as we only need to find the largest and smallest actions. In summary, we can compute $\hat{\mathcal{A}}_{\text{adv}}(s, \pi) = [\underline{\pi_\theta(s)}, \overline{\pi_\theta(s)}]$.

Computing $\mathcal{A}_{\text{adv}}$ in Discrete Action Space. For a discrete action space, the output of π_θ is a categorical distribution, and π selects the action with the highest probability. Or equivalently, in value-based algorithms like DQN, the Q network (can be regarded as π_θ) outputs the Q estimates for each action, and π selects the action with the highest Q value. In this case, we can compute the upper and lower bound of π_θ in every dimension (corresponding to an action), denoted as $\overline{a}_i, \underline{a}_i$, $\forall 1 \leq i \leq |\mathcal{A}|$. Then, an action $a_i \in \mathcal{A}$ is in $\hat{\mathcal{A}}_{\text{adv}}$ if for all $1 \leq j \leq |\mathcal{A}|, j \neq i$, we have $\overline{a}_i > \underline{a}_j$.

Implementation details of $\mathcal{A}_{\text{adv}}$ For a continuous action space, interval bound propagation (IBP) is the cheapest method to implement convex relaxation. We use IBP+Backward relaxation provided by *auto_LiRPA* library, following (Zhang et al., 2020b) to efficiently produce tighter bounds $\mathcal{A}_{\text{adv}}$ for the policy networks π_{theta}. For a discrete action space, we compute the layer-wise output bounds for the Q-network by applying robustness verification algorithms from (Oikarinen et al., 2021).

D.2. Worst-case-aware Robust PPO (WocaR-PPO)

In policy-based DRL methods (Schulman et al., 2015; Lillicrap et al., 2015; Schulman et al., 2017) such as PPO, the actor policy π_θ is optimized so that it increases the probability of selecting actions with higher critic values. Therefore, we combine our worst-attack critic and the original critic function, and optimize π_θ such that both the natural value ($\mathcal{L}_{\text{RL}}$) and the worst-attack action value $\underline{Q^\pi_\phi}$ ($\mathcal{L}_{\text{wst}}$) can be increased ($\mathcal{L}_{\text{RL}}$ and $\mathcal{L}_{\text{wst}}$). At the same time, π_θ is also regularized by $\mathcal{L}_{\text{reg}}$.

We provide the full algorithm of WocaR-PPO in Algorithm 1 and highlight the differences with the prior method SA-PPO. WocaR-PPO needs to train an additional worst-attack critic $\underline{Q^\pi_\phi}$ to provide the robust-PPO-clip objective. The perturbation budget ϵ_t increases slowly during training. The implementation of $\mathcal{L}_{\text{reg}}$ is the same as the SA-regularizer (Zhang et al., 2020b). For computing the state importance weight w_{s_t}, because there is no Q-value network in PPO, we provide a different formula to measure the state importance without extra calculation (Line 11 in Algorithm 1).

D.3. Worst-case-aware Robust DQN (WocaR-DQN)

For value-based DRL methods (Mnih et al., 2013; Guez & Van Hasselt, 2015; Wang et al., 2016) such as DQN, a Q network is learned to evaluate the natural action value. Although the policy is not directly modeled by a network, the Q network induces a greedy policy by $\pi(s) = \text{argmax}_a Q(s, a)$. To distinguish the acting policy and the natural action value, we keep the original Q network, and learn a new Q network that serves as a robust policy. This new Q network is called a *robust Q network*, denoted by Q_r, which is used to take greedy actions $a = \pi(s) := \text{argmax}_a Q_r(s, a)$ In addition to the original vanilla Q network Q_v and the robust Q network Q_r, we learn the worst-attack critic network $\underline{Q^\pi_\phi}$, which evaluates the worst-attack action value of the greedy policy induced by Q_r. Then, we update Q_r by assigning higher values for actions with both high natural Q value and high worst-attack action value ($\mathcal{L}_{\text{RL}}$ and $\mathcal{L}_{\text{wst}}$), while enforcing the network to output the same action under bounded state perturbations ($\mathcal{L}_{\text{reg}}$).

Algorithm 1 Worst-case-aware Robust PPO (WocaR-PPO). We highlight the difference compares with SA-PPO (Zhang et al., 2020b) in blue.

Input: Number of iterations T, a schedule ϵ_t for the perturbation radius ϵ, weights $\kappa_{\text{wst}}, \kappa_{\text{reg}}$

1: Initialize policy network $\pi_{\theta_\pi}(a \mid s)$, value network $V_{\theta_V}(s)$ and worst-attack critic network $\underline{Q}^{\pi}_{\phi}(s,a)$ with parameters θ_π, θ_V and ϕ

2: **for** $k = 0, 1, ..., T$ **do**

3: Collect a set of trajectories $\mathcal{D} = \{\tau_k\}$ by running π_{θ_π} in the environment, each trajectory τ_k contains $\tau_k := \{(s_t, a_t, r_t, s_{t+1})\}, t \in [|\tau_k|]$

4: Compute rewards-to-go $\hat{R}_t$ for each step t in every trajectory k with discount factor γ

5: Compute advantage estimation $\hat{A}_t$ based on the current value function $V_{\theta_V}(s_t)$ and cumulative reward $\hat{R}_t$ for each step t

6: Update parameters of value function θ_V by regression on mean-squared error:

$$\theta_V \leftarrow \arg\min_{\theta_V} \frac{1}{|\mathcal{D}|\,|\tau_k|} \sum_{\tau_k \in \mathcal{D}} \sum_{t=0}^{|\tau_k|} \left(V_{\theta_V}(s_t) - \hat{R}_t\right)^2$$

7: Use IBP to compute bounds of current policy network π:
Find the upper bound $\overline{\pi}(s_{t+1}, \epsilon; \theta)$ and lower bound $\underline{\pi}(s_{t+1}, \epsilon; \theta)$ of the policy network π_{θ_π}

8: Select the worst action for next states:
Calculate the action satisfied $\hat{a}_{t+1} = \arg\min_{a \in [\underline{\pi}, \overline{\pi}]} \underline{Q}^{\pi}_{\phi}(s_{t+1}, a)$ with the worst-attack critic network $\underline{Q}^{\pi}_{\phi}$ using gradient descent.

9: Compute next worst-case value:
Set $\underline{y}_t = \begin{cases} r_t & \text{for terminal } s_{t+1} \\ r_t + \gamma \underline{Q}^{\pi}_{\phi}(s_{t+1}, \hat{a}_{t+1}) & \text{for non-terminal } s_{t+1} \end{cases}$

10: Update parameters of worst-attack critic network ϕ by minimizing the TD-error ($\mathcal{L}_{\text{est}}$):

$$\phi \leftarrow \arg\min_{\phi} \frac{1}{|\mathcal{D}|\,|\tau_k|} \sum_{\tau_k \in \mathcal{D}} \sum_{t=0}^{|\tau_k|} (\underline{y}_t - \underline{Q}^{\pi}_{\phi}(s_t, a_t))^2$$

11: For each state s_t, calculate a state importance weight w_{s_t} by $V_{\theta_V}(s_t) - \min_a \underline{Q}^{\pi}_{\phi}(s_t, a)$ for s_t

12: Solve the value-enhanced state regularization loss by SGLD (Stochastic gradient Langevin dynamics (Gelfand & Mitter, 1991)) (from SA-PPO (Zhang et al., 2020b)):

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

13: Update the policy network by minimizing the Robust-PPO-Clip objective (via ADAM):

$$\theta_\pi \leftarrow \arg\min_{\theta'_\pi} \frac{1}{|\mathcal{D}||\tau_k|} \left[\sum_{\tau_k \in \mathcal{D}} \sum_{t=0}^{|\tau_k|} \min\left(\rho_{\theta'_\pi}(a_t \mid s_t)(\hat{A}_t + \kappa_{\text{wst}} \underline{Q}^{\pi}_{\phi}(s_t, a_t)), g(\rho_{\theta'_\pi}(a_t \mid s_t)) \right) (\hat{A}_t + \kappa_{\text{wst}} \underline{Q}^{\pi}_{\phi}(s_t, a_t))) + \kappa_{\text{reg}} w(s_i) \mathcal{L}_{\text{reg}}(\pi_\theta) \right]$$

where $\rho_{\theta'_\pi}(a_t \mid s_t) := \frac{\pi_{\theta'_\pi}(a_t|s_t)}{\pi_{\theta_\pi}(a_t|s_t)}$, $g(\rho) := \text{clip}\left(\rho_{\theta'_\pi}(a_t \mid s_t), 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}\right)$

14: **end for**

WocaR-DQN is presented in Algorithm 2. WocaR-DQN trains three Q-value functions including a vanilla Q network, a worst-case Q network, and a robust Q network. The worst-case Q $\underline{Q}^{\pi}_{\phi}$ is learned to estimate the worst-case performance and the robust Q is updated using the vanilla value and worst-case value together. Moreover, a target Q network is used as the original DQN implementation, to compute the target value when updating the vanilla Q network (Line 8 to 10 in Algorithm 2). To learn the worst-case critic $\underline{Q}^{\pi}_{\phi}$, we select the worst-attack action from the estimated possible perturbed action set $\hat{\mathcal{A}}_{\text{adv}}$ to compute the worst-case TD loss $\mathcal{L}_{\text{est}}$ (Line 11 to 15). The implementation of $\mathcal{L}_{\text{reg}}$ is the same as the SA-regularizer (Zhang et al., 2020b), where the robust Q network is regularized. To update the robust Q, we use a special y_i^r

which combines the target Q $Q_{v'}$ and Q_r for the next state to compute the TD loss, and minimize the $\mathcal{L}_{\text{reg}}$ weighted by the state importance $w(s_i)$ (Line 16 to 17). In WocaR-DQN, we use an increasing ϵ_t schedule and a more slowly increasing worst-case schedule $\kappa_{wst}(t)$ for robust Q training.

Algorithm 2 Worst-case-aware Robust DQN (WocaR-DQN). We highlight the difference compares with SA-DQN (Zhang et al., 2020b) in blue.

Input: Number of iterations T, target network update coefficient τ, a schedule ϵ_t for the perturbation radius ϵ, a worst-case schedule $\kappa_{\text{wst}}(t)$ for weight κ_{wst}, regularization weight κ_{reg}

1: Initialize a vanilla Q network $Q_v(s,a)$, target Q network $Q_{v'}(s,a)$, a robust Q network $Q_r(s,a)$, and a worst-attack critic $\underline{Q}^{\pi}_{\phi}(s,a)$ with parameters θ_{Q_v}, $\theta_{Q_{v'}}$, θ_{Q_r}, and ϕ

2: Initialize replay buffer $\mathcal{B}$

3: **for** $k = 0, 1, ..., T$ **do**

4: With probability β select random action a_t, otherwise select $a_t = \arg\max_a Q_r(s_t, a|\theta_{Q_r})$

5: Execute action a_t in environment and observe reward r_t and the next state s_{t+1}.

6: Store transition $\{s_t, a_t, r_t, s_{t+1}\}$ in $\mathcal{B}$

7: Sample random a minibatch of N transitions $\{s_i, a_i, r_i, s_{i+1}\}$ from $\mathcal{B}$

8: Set $y_i = \begin{cases} r_i & \text{for terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q_{v'}(s_{i+1}, a'; \theta) & \text{for non-terminal } s_{i+1} \end{cases}$

9: Compute TD-loss for the vanilla Q network: $L(s_i, a_i, s_{i+1}; \theta) = (y_i - Q_v(s_i, a_i; \theta))^2$ and optimize θ_{Q_v}

10: Soft update the target action-value network: $\theta_{Q_{v'}} \leftarrow \tau\theta_{Q_v} + (1-\tau)\theta_{Q_{v'}}$

11: Computing bounds of robust action-value function:
For each action a in action space $\mathcal{A}$, calculate the output bounds of robust action-value function Q_r under ϵ_t-bounded perturbations using IBP to input s_{i+1}: $Q_l(s_{t+1}, a, \epsilon_t)$ and $Q_u(s_{t+1}, a, \epsilon_t)$.

12: Find the possible perturbed action set:
For every action $a \in \mathcal{A}$, if $Q_u(s_{t+1}, a, \epsilon_t) > Q_l(s_{t+1}, a', \epsilon_t), \forall a' \in \mathcal{A}$, then add a in the perturbed action set $\hat{\mathcal{A}}_{\text{adv}}$

13: Calculate the worst-attack action: $\hat{a}_{i+1} = \arg\min_{a \in \hat{\mathcal{A}}_{\text{adv}}} \underline{Q}^{\pi}_{\phi}(s_{i+1}, a)$.

14: Set $\underline{y}_i = \begin{cases} r_i & \text{for terminal } s_{i+1} \\ r_i + \gamma \underline{Q}^{\pi}_{\phi}(s_{i+1}, \hat{a}_{i+1}; \theta) & \text{for non-terminal } s_{i+1} \end{cases}$

15: Compute TD-loss for worst-attack critic: $\mathcal{L}_{\text{est}} = (\underline{y}_i - \underline{Q}^{\pi}_{\phi}(s_i, a_i; \phi))^2$ and perform a gradient descent step with respect to the parameters ϕ

16: Calculate the state importance w_{s_i} for each s_i by normalizing $\max_a Q_v(s_t, a) - \min_a Q_v(s_t, a)$

17: Update the robust Q function Q_r based on the modified TD-Loss and value-enhanced state regularization:

$$L(s_i, a_i, s_{i+1}; \theta_{Q_r}) = (y_i^r - Q_r(s_i, a_i; \theta))^2 + \kappa_{\text{reg}} w(s_i) \mathcal{L}_{\text{reg}}(\theta_{Q_r})$$

where $y_i^r = r_i + \gamma \max_{a'} \left[\kappa_{\text{wst}}(t) Q_{v'}(s_{i+1}, a'; \theta) + (1 - \kappa_{\text{wst}}(t)) \underline{Q}^{\pi}_{\phi}(s_{i+1}, a'; \theta) \right]$ if s_{i+1} is a non-terminal state, otherwise $y_i^r = r_i$

18: **end for**

D.4. Worst-case-aware Robust A2C (WocaR-A2C)

We also provide WocaR-A2C based on A2C implementation in Algorithm 3. Differ from the original A2C, WocaR-A2C needs to learn an additional $\underline{Q}^{\pi}_{\phi}$ similar to WocaR-PPO. To learn $\underline{Q}^{\pi}_{\phi}$, we compute the output bounds for the policy network π_{θ_π} under ϵ-bounded perturbations and then select the worst action $\hat{a}_{t+1}$ to calculate the TD-loss $\mathcal{L}_{\text{est}}$ (Line 6 to 9). The solutions for state importance weight $w(s_t)$ and regularization $\mathcal{L}_{\text{reg}}$ are same as WocaR-PPO (Line 10-11). To learn the policy network π_{θ_π}, we minimize the $\underline{Q}^{\pi}_{\phi}$ value together with the original actor loss (Line 12).

D.5. Extension to Action Attacks

Although our paper mainly focuses on attacks on state observations, our proposed techniques and algorithms based on the worst-attack Bellman operator can be easily extended to action attack, which is another threat model studied in previous

Algorithm 3 Worst-case-aware Robust A2C (WocaR-A2C). We highlight the difference compares with SA-A2C (Zhang et al., 2020b) in blue.

Input: Number of iterations T, a schedule ϵ_t for the perturbation radius ϵ, weights $\kappa_{\text{wst}}, \kappa_{\text{reg}}$

1: Initialize policy network $\pi_{\theta_\pi}(a \mid s)$, value network $V_{\theta_V}(s)$ and worst-attack critic network $\underline{Q}^{\pi}_{\phi}(s,a)$ with parameters θ_π, θ_V and ϕ

2: **for** $k = 0, 1, ..., T$ **do**

3: Collect a set of trajectories $\mathcal{D} = \{\tau_k\}$ by running π_{θ_π} in the environment, each trajectory τ_k contains $\tau_k := \{(s_t, a_t, r_t, s_{t+1})\}, t \in [|\tau_k|]$

4: Compute advantage function A_t by

$$A_t = r_t + \gamma V_{\theta_V}(s_{t+1}) - V_{\theta_V}(s_t)$$

5: Update parameters of value function θ_V by regression on mean-squared error:

$$\theta_V \leftarrow \arg\min_{\theta_V} \frac{1}{|\mathcal{D}|\,|\tau_k|} \sum_{\tau_k \in \mathcal{D}} \sum_{t=0}^{|\tau_k|} A_t^2$$

6: Use IBP to compute bounds of current policy network π:
Find the upper bound $\overline{\pi}(s_{t+1}, \epsilon; \theta)$ and lower bound $\underline{\pi}(s_{t+1}, \epsilon; \theta)$ of the policy network π_{θ_π}

7: Select the worst action for next states:
Calculate the action satisfied $\hat{a}_{t+1} = \arg\min_{a \in [\underline{\pi}, \overline{\pi}]} \underline{Q}^{\pi}_{\phi}(s_{t+1}, a)$ with the worst-attack critic network $\underline{Q}^{\pi}_{\phi}$ using gradient descent.

8: Compute next worst-case value:

$$\text{Set } \underline{y}_t = \begin{cases} r_t & \text{for terminal } s_{t+1} \\ r_t + \gamma \underline{Q}^{\pi}_{\phi}(s_{t+1}, \hat{a}_{t+1}) & \text{for non-terminal } s_{t+1} \end{cases}$$

9: Update parameters of worst-attack critic network ϕ by minimizing the TD-error ($\mathcal{L}_{\text{est}}$):

$$\phi \leftarrow \arg\min_{\phi} \frac{1}{|\mathcal{D}|\,|\tau_k|} \sum_{\tau_k \in \mathcal{D}} \sum_{t=0}^{|\tau_k|} (\underline{y}_t - \underline{Q}^{\pi}_{\phi}(s_t, a_t))^2$$

10: For each state s_t, calculate a state importance weight $w(s_t)$ by $V_{\theta_V}(s_t) - \min_a \underline{Q}^{\pi}_{\phi}(s_t, a)$ for s_t

11: Solve the value-enhanced state regularization loss (Zhang et al., 2020a) by SGLD (Stochastic gradient Langevin dynamics (Gelfand & Mitter, 1991)):

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

12: Update the policy network by (via ADAM)

$$\theta_\pi \leftarrow \arg\min_{\theta'_\pi} \frac{1}{|\mathcal{D}||\tau_k|} \left[\sum_{\tau_k \in \mathcal{D}} \sum_{t=0}^{|\tau_k|} (A_t log \pi_{\theta_\pi}(s_t) + \kappa_{\text{wst}} \underline{Q}^{\pi}_{\phi}(s_t, a_t)) \right]$$

13: **end for**

works (Pinto et al., 2017; Tan et al., 2020; Tessler et al., 2019). In fact, for action attack, we even do not need to apply IBP for the worst-attack Bellman backup. We could just simply replace $\mathcal{A}_{\text{adv}}$ with the set of actions that the agent could take under attack, then the rest of the algorithms will follow the exact same as the ones presented here.

D.6. Differences with Worst-case Value in Risk-sensitive RL

Note that the proposed worst-attack Bellman operator is different from the worst-case Bellman operator in the literature of risk-sensitive RL (Heger, 1994; Gaskett, 2003; Tamar et al., 2013). The main goal of risk-sensitive RL is to avoid unsafe trajectories under the intrinsic uncertainties of RL environments (Garcıa & Fernández, 2015; Bechtle et al., 2020; Thomas

et al., 2021). 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.

D.7. Comparison between worst-case-aware policy optimization and ATLA

The proposed worst-case-aware policy optimization has several **merits** compared to prior ATLA (Zhang et al., 2021) and PA-ATLA (Sun et al., 2021b) 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 (Zhang et al., 2021) learns an attacker by treating the current policy as a black box, ignoring the intrinsic properties of the policy. PA-ATLA (Sun et al., 2021b), 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, although is also an approximation due to the convex relaxation, computes a lower bound of $\underline{Q}^{\pi}$ and thus can better indicate the robustness of a policy.

E. Experiment Details and Additional Results

E.1. Implementation Details

E.1.1. Baseline introductions

We compare our algorithm with several state-of-the-art robust training methods, including (1) *SA-PPO/SA-DQN* (Zhang et al., 2020b): regularizing policy networks by convex relaxation. (2) *ATLA-PPO* (Zhang et al., 2021): alternately training an agent and an RL attacker. (3) *PA-ATLA-PPO* (Sun et al., 2021b): alternately training an agent and a more advanced RL attacker PA-AD. (4) *RADIAL-PPO/RADIAL-DQN* (Oikarinen et al., 2021): 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 (Sun et al., 2021b). (PA-ATLA has an A2C version, which we compare in Appendix E.2.) Therefore, we reproduce their ATLA-PPO and PA-ATLA-PPO results and compare them with our WocaR-PPO. For a comprehensive robustness evaluation on MuJoCo, we attack the trained robust models with multiple existing attack methods, including: (1) *MaxDiff* (Zhang et al., 2020b) (maximal action difference), (2) *Robust Sarsa (RS)* (Zhang et al., 2020b) (attacking with a robust action-value function), (3) *SA-RL* (Zhang et al., 2020b) (finding the optimal state adversary) and (4) *PA-AD* (Sun et al., 2021b) (the *existing strongest attack* by learning the optimal policy adversary with RL). For Atari environments, we include the following common attacks: (1) 10-step untargeted *PGD* (projected gradient descent) attack, (2) *MinBest* (Huang et al., 2017), which minimizes the probability of choosing the "best" action, (3) *PA-AD* (Sun et al., 2021b), as the state-of-the-art RL-based adversarial attack algorithm on Atari.

For reproducibility, the reported results are selected from 30 agents for different training methods with medium performance due to the high variance in RL training.

E.1.2. PPO in MuJoCo

(a) PPO Baselines

Vanilla PPO We use the optimal hyperparameters from (Zhang et al., 2020b) with the original fully connected (MLP) structure as the policy network for vanilla PPO training on all environments. On Hopper, Walker2d and Halfcheetah, we train for 2 million steps (976 iterations) , and 10 million steps (4882 iterations) on Ant to ensure convergence, which are consistent with other baselines (except ATLA methods).
SA-PPO We use the hyperparameters using a grid search and solve the regularizer using convex relaxation with the IBP+Backward scheme to solve the regularizer. The regularization parameter $kappa$ is chosen in $\{0.01, 0.03, 0.1, 0.3, 1.0\}$.
ATLA-PPO The hyperparameters for both policy and adversary are tuned for vanilla PPO with LSTM models. A larger entropy bonus coefficient is set to allow sufficient exploration. We set $N_v = N_\pi = 1$ for all experiments. We train 2441 iterations for Hopper, Walker2d, and Halfcheetah as well as 4882 iterations for Ant.

PA-ATLA-PPO We use the hyperparameters similar to ATLA-PPO and conduct a grid search for a part of adversary hyperparameters including the learning rate and the entropy bonus coefficient.
RADIAL-PPO RADIAL-PPO applies the same value of hyperparameters from (Oikarinen et al., 2021). We train agents with the same iterations aligning vanilla PPO for fair comparison.

(b) PPO Attackers

For **Random** and **MaxDiff** attack, we directly use the implementation from (Zhang et al., 2021). The reported rewards under RS attack are from 30 trained robust value function, which is used to attack agents.
For **SA-RL** attack, a grid search of the optimal hyperparameters for each robust agents is conducted to find the strongest attacker. The strength of the regularization κ is set as 1×10^{-6} to 1.
For **PA-AD** attack, the adversaries are trained by PPO with a grid search of hyperparameters to obtain the strongest adversary. For different types of RL-based attacks, we respectively train 100 adversaries and report the worst rewards among all trained adversaries.

(c) WocaR-PPO We use the same LSTM structure (single layer with 64 hidden neurons as in vanilla PPO agents. With a grid search experiment, we find the optimal hyperparameters for WocaR-PPO. Specially, we use PGD to compute bounds for the policy network and convex relaxation to solve the state regularization. The number of WocaR-PPO training steps in all environments are the same as those in vanilla PPO. We tune the adjustable weight κ_{wst} and increase κ_{wst} from 0 to the target value. For Hopper, Walker2d and Halfcheetah, κ_{wst} is linearly increasing and we set the target value as 0.8. For Ant, we choose the exponential increase and the target value as 0.5.

E.1.3. DQN IN ATARI

(a) DQN Baselines

Vanilla DQN We follow (Zhang et al., 2020b) and (Oikarinen et al., 2021) in hyperparameters and network structures for vanilla DQN training. The implementation of all our baselines applies Double DQN (Hado Van Hasselt, 2016) and Prioritized Experience Replay (Schaul et al., 2015). For each Atari environment without framestack, we normalize the pixel values to $[0, 1]$ and clip rewards to $[-1, +1]$. For reliably convergence, we run 6×10^6 steps for all baselines on all environments. Additionally, we use a replay buffer with a capacity of 5×10. During testing, we evaluate agents without epsilon greedy exploration for 1000 episodes.
SA-DQN SA-DQN use the same settings of network structures and hyperparameters as in vanilla DQN. The regularization parameter κ is chosen from $0.005, 0.01, 0.02$ and the schedule of ϵ during training also follows (Zhang et al., 2020b).
RADIAL-DQN Following the original implementation from (Oikarinen et al., 2021), we reproduce the results of RADIAL-DQN with our environment settings.

(b) DQN Attackers

For **PGD** attacks, we apply 10-step untargeted PGD attacks. We also try 50-step PGD attacks, but we find that the rewards of robust agents do not further reduce.
For **MinBest** attacks, we use FGSM to compute state perturbations following (Huang et al., 2017).
For **PA-AD** attacks, the PA-AD attackers are learned with the ACKTR algorithm. We use a learning rate 0.0001 and train the attackers for 5 million frames.

(c) WocaR-DQN For WocaR-DQN, we keep the same network architectures and hyperparameters as in vanilla DQN agents. During training, we set the adjustable weight κ_{wst} as 0 for the first 2×10^6 steps, and then exponentially increase it from 0 to 0.5 for 4×10^6 steps.

E.2. Additional Experiment Results on Robustness Performance

E.2.1. MUJOCO EXPERIMENTS

We reported all results in Table 1 and Figure 7 including episode rewards and training curves of well-trained robust models under various adversarial attacks. Under this full adversarial evaluation, we provide a robustness comparison between baselines and our algorithm from a comprehensive angle. We report the attack performance under a common chosen perturbation budget ϵ following (Zhang et al., 2020b; 2021). Results in all four MuJoCo environments show that our WocaR-PPO is the most robust method. We emphasize that Table 1 reports the final performance of all robust training baselines after convergence, but some baselines takes much more steps than our WocaR-PPO. Table 4 in Appendix E.3.2

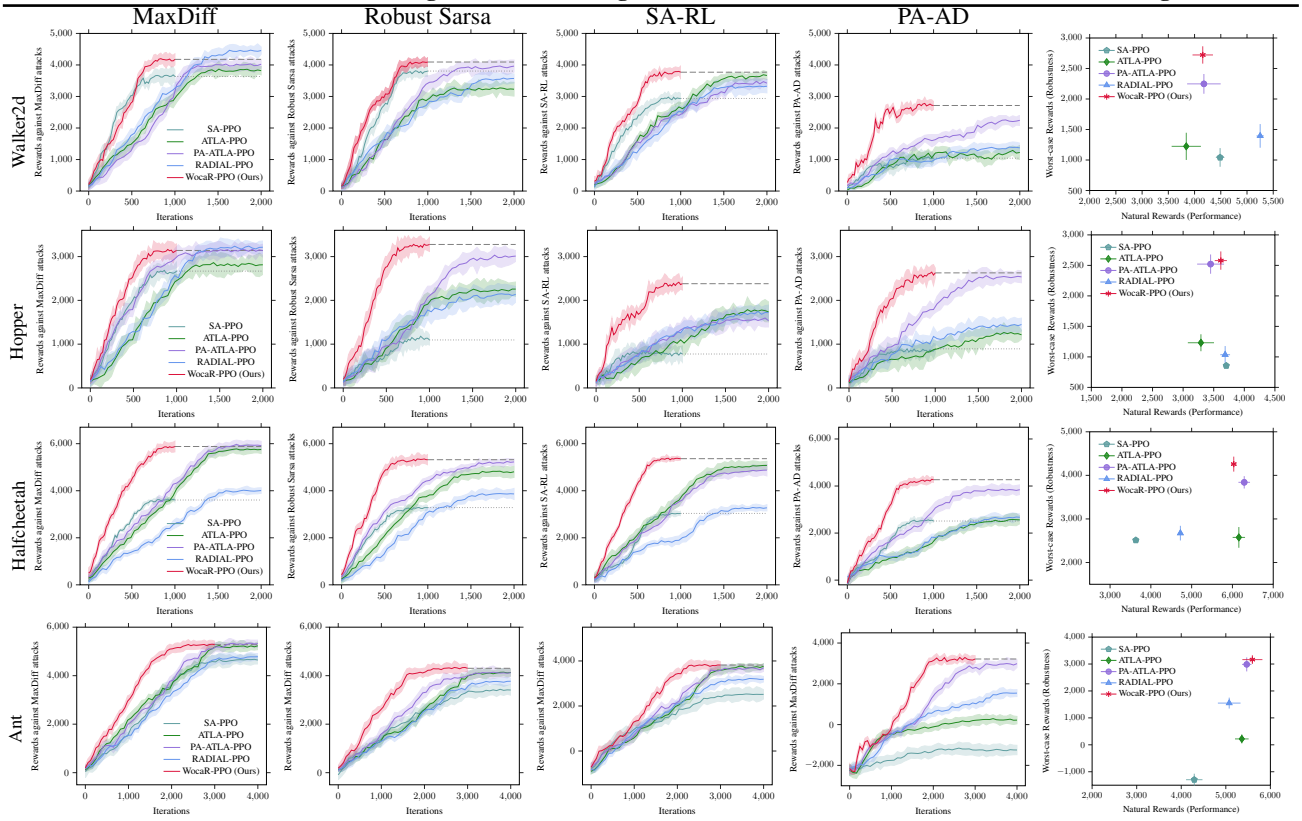


Figure 7. **Robustness, Efficiency and High Natural Performance of WocaR-PPO. (Left four columns)** Learning curves of rewards under MaxDiff, Robust Sarsa, SA-RL and PA-AD *(the strongest)* attacks during training on four environments. **(Rightmost column)** Average episode natural rewards v.s. average 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 E.3.1. compares all methods under the same number of training steps, where WocaR-PPO outperforms baselines more significantly.

E.2.2. Atari Experiments

In Table 2, we present performance based on DQN on four Atari environments under 1/255 and 3/255 ϵ attack. Under ϵ of 1/255, our WocaR-DQN achieves competitive performance under PGD attacks and outperforms all baselines under MinBest and PA-AD attacks, which shows better robustness of WocaR-DQN under weaker attacks. 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.

Based on vanilla A2C, we implement SA-A2C(Zhang et al., 2020b) and PA-ATLA-A2C(Sun et al., 2021b) as robust baselines. We implement WocaR-A2C to compare with ATLA methods on Atari. In Table 3, under any ϵ value, our WocaR-A2C outperforms other robust baselines across different attacks. We can conclude that our method considerably enhance more robustness than ATLA methods on Atari.

E.3. Additional Evaluation and Ablation Studies

E.3.1. Robustness Evaluation Using Multiple ϵ

To study how WocaR-PPO performs under attacks with different value of ϵ, Figure 8 shows the evaluation of our algorithms under different ϵ attacks compared with the baselines in Hopper and Walker2d. We can conclude that our robustly trained model universally and significantly outperforms other robust agents considering various attack budget ϵ.

E.3.2. Additional Evaluation on Sample Efficiency

In Table 4, we report the performance of WocaR-PPO and all robust PPO baselines using the same training steps. We find that under limited training steps, ATLA-PPO, PA-ATLA-PPO and RADIAL-PPO obtain sub-optimal robustness, which suggests that these methods are more sample-hungry. In contrast, WocaR-PPO converges under fewer steps and achieves

Environment	Model	Natural Reward	Random	MAD	RS	SA-RL	PA-AD
Halfcheetah state-dim: 17 ϵ=0.15	PPO (vanilla)	**7117 ± 98**	5486 ± 1378	1836 ± 866	489 ± 758	-660 ± 218	-356 ± 407
	SA-PPO	3632 ± 20	3619 ± 18	3624 ± 23	3283 ± 20	3028 ± 23	2512 ± 16
	ATLA-PPO	6157 ± 852	6164 ± 603	5790 ± 174	4806 ± 392	5058 ± 418	2576 ± 548
	PA-ATLA-PPO	6289 ± 342	**6215 ± 346**	**5961 ± 253**	5226 ± 114	4872 ± 379	3840 ± 273
	RADIAL-PPO	4724 ± 14	4731 ± 42	3994 ± 156	3864 ± 232	3253 ± 131	2674 ± 168
	WocaR-PPO (Ours)	6032 ± 68	5969 ± 149	5850 ± 228	**5319 ± 220**	**5365 ± 54**	**4269 ± 172**
Hopper state-dim: 11 ϵ=0.075	PPO (vanilla)	3167 ± 542	2101 ± 793	1410 ± 655	794 ± 238	636 ± 9	160 ± 136
	SA-PPO	3705 ± 2	2710 ± 801	2652 ± 835	1130 ± 42	1076 ± 791	856 ± 21
	ATLA-PPO	3291 ± 600	3165 ± 576	2814 ± 725	2244 ± 618	1772 ± 802	1232 ± 350
	PA-ATLA-PPO	3449 ± 237	3325 ± 239	3145 ± 546	3002 ± 329	1529 ± 284	2521 ± 325
	RADIAL-PPO	**3740 ± 44**	**3729 ± 100**	3214 ± 142	2141 ± 232	1722 ± 186	1439 ± 204
	WocaR-PPO (Ours)	3616 ± 99	3633 ± 30	**3541 ± 207**	**3277 ± 159**	**2390 ± 145**	**2579 ± 229**
Walker2d state-dim: 17 ϵ=0.05	PPO (vanilla)	4472 ± 635	3007 ± 1200	2869 ± 1271	1336 ± 654	1086 ± 516	804 ± 130
	SA-PPO	**4487 ± 61**	**4465 ± 39**	3668 ± 689	3808 ± 138	2908 ± 336	1042 ± 353
	ATLA-PPO	3842 ± 475	3927 ± 368	3836 ± 492	3239 ± 294	3663 ± 707	1224 ± 770
	PA-ATLA-PPO	4178 ± 529	4129 ± 78	4024 ± 272	3966 ± 307	3450 ± 178	2248 ± 131
	RADIAL-PPO	5251 ± 12	5184 ± 42	**4494 ± 150**	3572 ± 239	3320 ± 245	1395 ± 194
	WocaR-PPO (Ours)	4156 ± 495	4244 ± 157	4177 ± 176	**4093 ± 138**	**3770 ± 196**	**2722 ± 173**
Ant state-dim: 111 ϵ=0.15	PPO (vanilla)	**5687 ± 758**	5261 ± 1005	1759 ± 828	268 ± 227	-872 ± 436	-2580 ± 872
	SA-PPO	4292 ± 384	4986 ± 452	4662 ± 522	3412 ± 1755	2511 ± 1117	-1296 ± 923
	ATLA-PPO	5359 ± 153	5366 ± 104	5240 ± 170	4136 ± 149	3765 ± 101	220 ± 338
	PA-ATLA-PPO	5469 ± 106	5496 ± 158	**5328 ± 196**	4124 ± 291	3694 ± 188	2986 ± 364
	RADIAL-PPO	5076 ± 254	5031 ± 142	4777 ± 156	3731 ± 177	3188 ± 115	1544 ± 194
	WocaR-PPO (Ours)	5596 ± 225	**5558 ± 241**	5284 ± 182	**4339 ± 160**	**3822 ± 185**	**3164 ± 163**

Table 1. Average episode rewards ± standard deviation over 50 episodes on five baselines and WocaR-PPO on Hopper, Walker2d, Halfcheetah, and Ant. Natural reward and rewards under five types of attacks are reported. Under each column corresponding to an evaluation metric, we bold the best results. And the row for the most robust agent is highlighted as gray. Note that *ATLA-PPO, PA-ATLA-PPO and RADIAL-PPO are trained with more than* 2× *steps than WocaR-PPO*, as reported in Table 5.

best performance with a large advantage, which shows the higher efficiency of WocaR-PPO.

E.3.3. Additional Results of Time Efficiency

Efficiency of Training WocaR-PPO. We show the training efficiency of WocaR-PPO from three aspects including time, training iterations, and sampling in MuJoCo environments by comparing with SA-PPO and state-of-the-art methods ATLA-PPO, PA-ATLA-PPO, and RADIAL-PPO in Table 5. For a fair comparison, we use the same *GeForce RTX 1080 Ti GPUs* to train all the robust agents.

Without training with an adversary, *our algorithm requires much less (only 50% or 75%) steps to reliably converge.* WocaR-PPO only takes less than half of time for low-dimensional environments to converge compared to ATLA methods and RADIAL-PPO. In high-dimensional environments like Ant, we only need 4 hours for training, while ATLA methods require at least 7 hours. When solving harder tasks, the efficiency advantage of WocaR-PPO is more obvious.

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).

Environment	Model	Natural Reward	PGD (10 steps)		MinBest		PA-AD	
			ϵ=1/255	ϵ=3/255	ϵ=1/255	ϵ=3/255	ϵ=1/255	ϵ=3/255
Pong	DQN	21.0 ± 0.0	-21.0 ± 0.0	-21.0 ± 0.0	-7.4 ± 2.8	-9.7 ± 4.0	-18.2 ± 2.3	-19.0 ± 2.2
	SA-DQN	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	20.6 ± 3.5	20.4 ± 1.8	18.7 ± 2.6
	RADIAL-DQN	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	21.0 ± 0.0	19.5 ± 2.1	20.3 ± 2.5	13.2 ± 1.8
	WocaR-DQN (Ours)	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**	**21.0 ± 0.0**	**20.8 ± 3.3**	**21.0 ± 0.2**	**19.7 ± 2.4**
Freeway	DQN	**34.0 ± 0.1**	0.0 ± 0.0	0.0 ± 0.0	9.5 ± 3.0	5.5 ± 1.8	9.3 ± 2.7	4.7 ± 2.9
	SA-DQN	30.0 ± 0.0	30.0 ± 0.0	30.0 ± 0.0	27.2 ± 3.4	18.3 ± 3.0	20.1 ± 4.0	9.5 ± 3.8
	RADIAL-DQN	33.1 ± 0.2	**33.1 ± 0.2**	**33.2 ± 0.2**	22.6 ± 3.3	16.4 ± 2.3	18.5 ± 4.2	10.8 ± 3.6
	WocaR-DQN (Ours)	31.2 ± 0.4	31.2 ± 0.5	31.4 ± 0.3	**29.6 ± 2.5**	**19.8 ± 3.8**	**24.9 ± 3.7**	**12.3 ± 3.2**
BankHeist	DQN	**1308 ± 24**	54 ± 20	0 ± 0	210 ± 79	119 ± 65	213 ± 111	102 ± 92
	SA-DQN	1245 ± 14	**1245 ± 10**	1176 ± 63	1148 ± 36	1024 ± 31	1054 ± 11	489 ± 106
	RADIAL-DQN	1178 ± 4	1178 ± 4	1176 ± 63	1049 ± 27	928 ± 113	1035 ± 46	508 ± 85
	WocaR-DQN (Ours)	1220 ± 12	1220 ± 3	**1214 ± 7**	**1192 ± 12**	**1045 ± 20**	**1096 ± 19**	**754 ± 102**
RoadRunner	DQN	**45527 ± 4894**	0 ± 0	0 ± 0	14962 ± 6431	2985 ± 1440	842 ± 41	203 ± 65
	SA-DQN	44638 ± 2367	43970 ± 975	20678 ± 1563	39736 ± 2315	4214 ± 2587	38432 ± 3574	5516 ± 4684
	RADIAL-DQN	44675 ± 5854	**44605 ± 1094**	38576 ± 1960	38060 ± 1799	8476 ± 3964	36310 ± 9149	1290 ± 4015
	WocaR-DQN (Ours)	44156 ± 2279	44079 ± 2154	**38720 ± 1765**	**40758 ± 3369**	**10545 ± 2984**	**38954 ± 3647**	**8239 ± 2766**

Table 2. Average episode rewards ± standard deviation over 1000 episodes on baselines and WocaR-DQN on Pong, Freeway, BankHeist, and RoadRunner. Natural reward and rewards under different attacks with ϵ of 1/255 and 3/255 are reported. We bold the best results for each evaluation metric. And the row for the most robust agents on all environments are highlighted by gray.

Environment	Model	Natural Reward	PGD (10 steps)		MinBest		PA-AD	
			ϵ=1/255	ϵ=3/255	ϵ=1/255	ϵ=3/255	ϵ=1/255	ϵ=3/255
BankHeist	A2C	**1228 ± 93**	67 ± 14	0 ± 0	972 ± 99	697 ± 153	636 ± 74	314 ± 116
	SA-A2C	1029 ± 152	1029 ± 156	976 ± 54	902 ± 89	786 ± 52	836 ± 70	644 ± 153
	PA-ATLA-A2C	1076 ± 56	1075 ± 79	1013 ± 69	957 ± 78	842 ± 154	862 ± 106	757 ± 132
	WocaR-A2C (Ours)	1089 ± 34	**1089 ± 78**	**1035 ± 102**	**1043 ± 29**	**937 ± 65**	**1004 ± 94**	**879 ± 128**

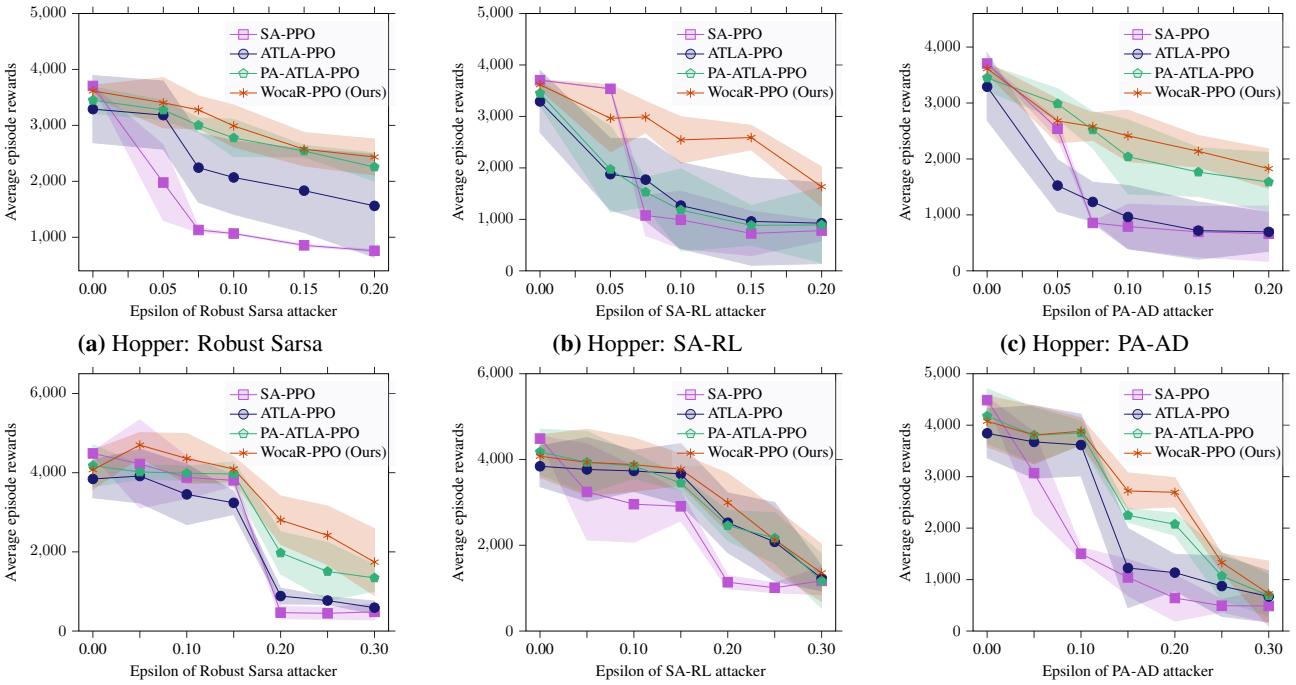
Table 3. Average episode rewards ± standard deviation over 1000 episodes on baselines and VaR-A2C on BankHeist. Natural reward and rewards under different attacks with ϵ of 1/255 and 3/255 are reported. We bold the best results for each evaluation metric. And the row for the most robust agents on all environments are highlighted by gray.

E.3.4. Effectiveness Verifications

we show the learning curves in Walker2d and Ant in Figure 9 to verify the effectiveness of worst-attack value estimation and worst-case-aware policy optimization. Figure 9(a) and (d) show the natural rewards of agents during training without attacks. The actual worst-attack rewards in Figure 9(b) and (e) refer to the the reward obtained by the agents under PA-AD attack (Sun et al., 2021b) which is the existing strongest attacking algorithm. To study the worst-case performance during training, We evaluate PPO, SA-PPO and WocaR-PPO agents after every 20 iterations using all types of attacks and report the worst-case rewards for each checkpoint. We also present the trend of the estimated worst-case values during training in Figure 9(c) and (f), which are tested by the trained worst-attack value functions $\underline{Q}^{\pi}_{\phi}$.

(1) Worst-case-aware 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 (Sun et al., 2021b)) 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 ($\mathcal{L}_{\text{wst}}$). Moreover, the adjustable weight κ_{wst} in Equation (9) 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.



(a) Hopper: Robust Sarsa **(b)** Hopper: SA-RL **(c)** Hopper: PA-AD

(d) Walker2d: Robust Sarsa **(e)** Walker2d: SA-RL **(f)** Walker2d: PA-AD

Figure 8. Comparisons under different attacks w.r.t. different budget ϵ's on Hopper and Walker2d.

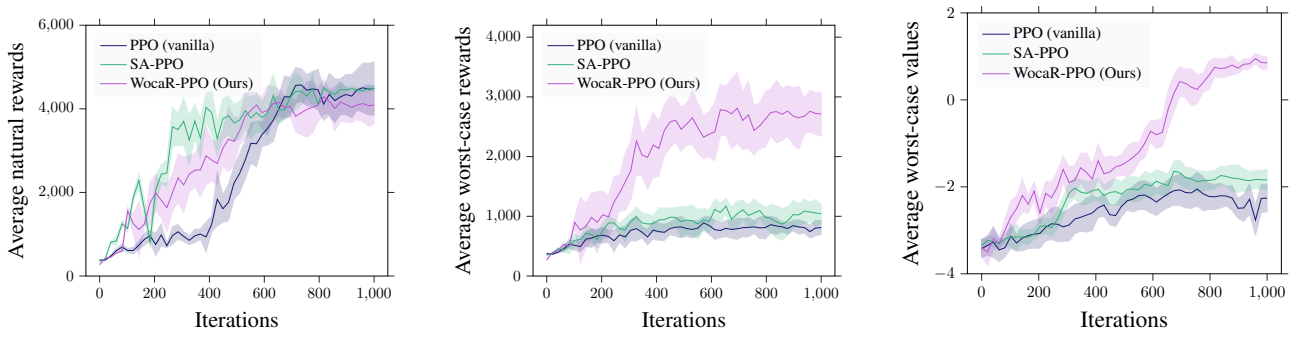
Environment	Model	Natural Reward	Random	MAD	RS	SA-RL	PA-AD
Halfcheetah state-dim: 17 ϵ=0.15	ATLA-PPO	4817 ± 277	4809 ± 186	4584 ± 100	4074 ± 285	4129 ± 348	1856 ± 294
	PA-ATLA-PPO	5023 ± 282	5076 ± 149	4720 ± 334	4392 ± 158	4159 ± 248	3085 ± 295
	RADIAL-PPO	4683 ± 97	4625 ± 190	3674 ± 222	3529 ± 173	2893 ± 165	2197 ± 251
	WocaR-PPO (Ours)	**6032 ± 68**	**5969 ± 149**	**5850 ± 228**	**5319 ± 220**	**5365 ± 54**	**4269 ± 172**
Hopper state-dim: 11 ϵ=0.075	ATLA-PPO	3265 ± 342	3195 ± 275	2675 ± 332	2098 ± 398	1542 ± 639	1135 ± 289
	PA-ATLA-PPO	3429 ± 196	3455 ± 315	3072 ± 478	2889 ± 258	1458 ± 274	2032 ± 244
	RADIAL-PPO	**3687 ± 80**	3627 ± 106	2952 ± 126	1094 ± 248	1243 ± 187	1036 ± 142
	WocaR-PPO (Ours)	3616 ± 99	**3633 ± 30**	**3541 ± 207**	**3277 ± 159**	**2390 ± 145**	**2579 ± 229**
Walker2d state-dim: 17 ϵ=0.05	ATLA-PPO	2664 ± 366	2695 ± 320	2547 ± 210	2439 ± 174	2092 ± 144	1544 ± 280
	PA-ATLA-PPO	3047 ± 223	3112 ± 111	2865 ± 230	2742 ± 177	2450 ± 229	1987 ± 246
	RADIAL-PPO	2143 ± 153	2231 ± 89	2095 ± 121	1680 ± 193	1078 ± 115	1274 ± 117
	WocaR-PPO (Ours)	**4156 ± 495**	**4244 ± 157**	**4177 ± 176**	**4093 ± 138**	**3770 ± 196**	**2722 ± 173**
Ant state-dim: 111 ϵ=0.15	ATLA-PPO	4249 ± 243	4218 ± 161	4036 ± 173	3391 ± 158	2045 ± 203	-349 ± 175
	PA-ATLA-PPO	4533 ± 238	4492 ± 190	4232 ± 203	3579 ± 261	2762 ± 152	1765 ± 185
	RADIAL-PPO	4379 ± 230	4194 ± 52	3278 ± 138	2348 ± 232	1380 ± 145	157 ± 124
	WocaR-PPO (Ours)	**5596 ± 225**	**5558 ± 241**	**5284 ± 182**	**4339 ± 160**	**3822 ± 185**	**3164 ± 163**

Table 4. Average episode rewards ± standard deviation over 50 episodes on baselines and WocaR-PPO trained for 2 million steps on Hopper, Walker2d, Halfcheetah and 7.5 million steps on Ant (less than the best settings). **Bold** numbers indicate the best results under each attack. The gray rows are the most robust agents.

Appendix E.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.*

Model	Hopper		Ant	
	Time (h)	Steps(m)	Time (h)	Steps (m)
SA-PPO	3.0	2.0	8.9	10.0
ATLA-PPO	5.6	5.0	12.8	10.0
PA-ATLA-PPO	5.2	5.0	12.3	10.0
RADIAL-PPO	3.2	4.0	10.2	10.0
WocaR-PPO (Ours)	**2.3**	**2.0**	**8.7**	**7.5**

Table 5. Efficiency comparison of state-of-the-art robust training methods and WocaR-PPO in Hopper and Ant. For Walker2d and Halfcheetah, the sampling steps are same as for Hopper and the training time is also extremely similar. We highlight the most efficient method as gray.



(a) Walker2d: Natural Rewards **(b)** Walker2d: Worst-case Rewards **(c)** Walker2d: Worst-case Values

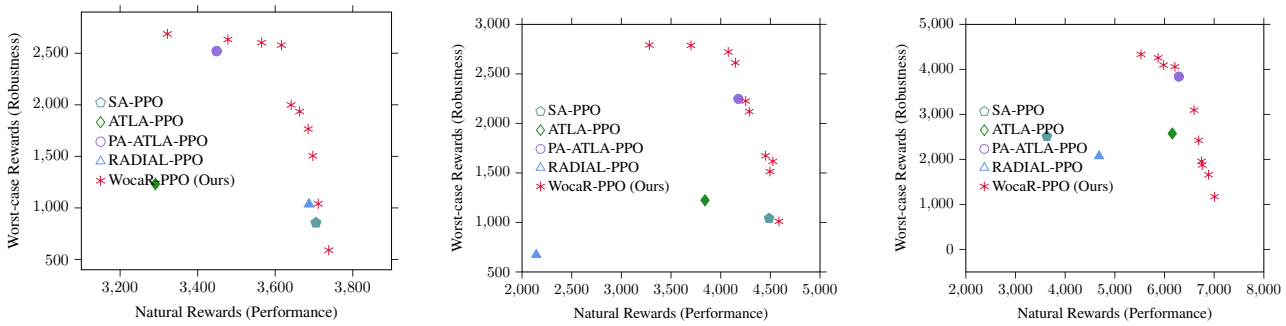
(d) Ant: Natural Rewards **(e)** Ant: Worst-case Rewards **(f)** Ant: Worst-case Values

Figure 9. Learning curves (mean ± standard deviation) of natural rewards, worst-case rewards under attacks and estimated worst-case values during training on Walker2d and Ant for vanilla PPO (blue), SA-PPO (green) and WocaR-PPO (purple).

E.3.5. Trade-off between Natural Performance and Robustness

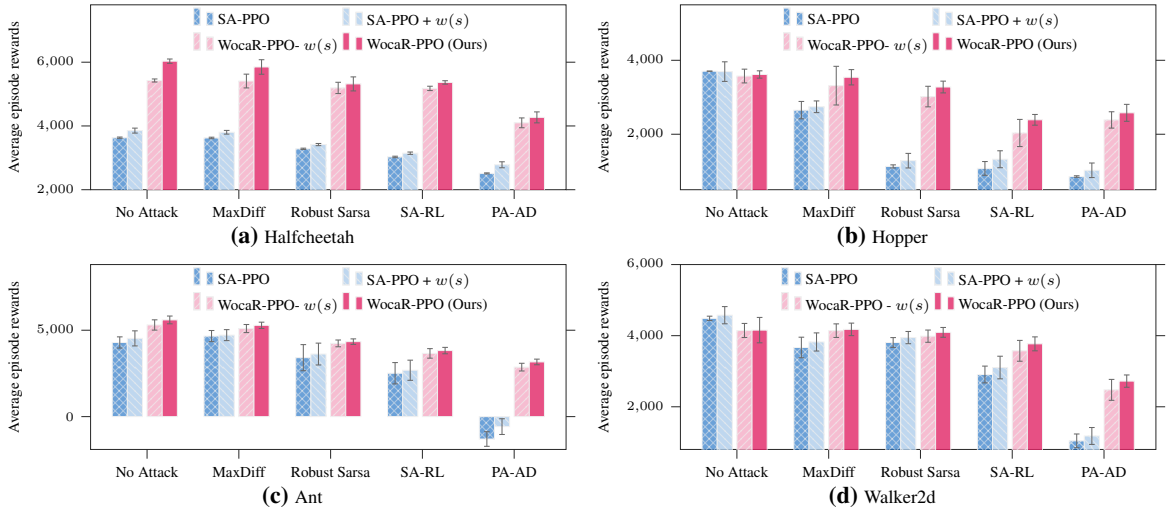
As mentioned in Section 3, the adjustable weight κ_{wst} controls the trade-off between natural performance and robustness. To discuss the effect of κ_{wst}, we train agents using WocaR-PPO in Hopper, Walker2d, and Halfcheetah with uniformly sampled 40 different values of weight κ_{wst} in range $(0, 1]$.

Figure 10 plots the worst-case performance and natural performance of robust training baselines and 10 agents trained by WocaR-PPO. We can see that when reward under worst-case perturbations increases, it leads to a reduction of the natural reward. Using different κ_{wst}'s, we balance the robustness and performance and report the results in Table 1 with significant better worst-case robustness and comparable natural performance compared with baselines. WocaR-PPO can always find policies which dominate other robust agents.



(a) Hopper **(b)** Walker2d **(c)** Halfcheetah

Figure 10. Average natural rewards and worst-case rewards of WocaR-PPO with different κ_{wst} and other baselines on Hopper, Walker2d, and Halfcheetah.



(a) Halfcheetah **(b)** Hopper **(c)** Ant **(d)** Walker2d

Figure 11. Ablation performance for the state importance weight $w(s)$ under no attack and different attacks on Hopper, Walker2d, Halfcheetah, and Ant.

E.3.6. Additional Ablation Studies

We conduct ablation experiments to analyze the effect of two techniques: our proposed state importance weight $w(s)$ and the state regularization loss $\mathcal{L}_{\text{reg}}$ (Zhang et al., 2020b).

For the state importance weight $w(s)$, we compare the performance of the original WocaR-PPO and WocaR-PPO without $w(s)$ in Figure 11. In addition, since SA-PPO (Zhang et al., 2020b) also uses a state regularization technique, we also equip SA-PPO with the state importance with $w(s)$, to demonstrate the universal applicability of this design. In all four MuJoCo environments, we can see that with $w(s)$, both WocaR-PPO and SA-PPO get boosted robustness, verifying the effectiveness of the state importance weight. 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.

For the state regularization loss $\mathcal{L}_{\text{reg}}$, Figure 12 verifies that $\mathcal{L}_{\text{reg}}$ enhances the robustness of WocaR-PPO, since the performance of WocaR-PPO drops without $\mathcal{L}_{\text{reg}}$.

On the other hand, Figure 12 also compares the performance of ATLA methods and our algorithm without $\mathcal{L}_{\text{reg}}$ (note that ATLA methods also regularizes the PPO policies during training). The results indicate that *the decisive contribution of WocaR-PPO to robustness improving comes from the worst-attack-aware policy optimization.*

These ablation studies demonstrate that all the techniques are beneficial for robustness improvement and further show that our worst-case-aware training performs better than training with attackers.

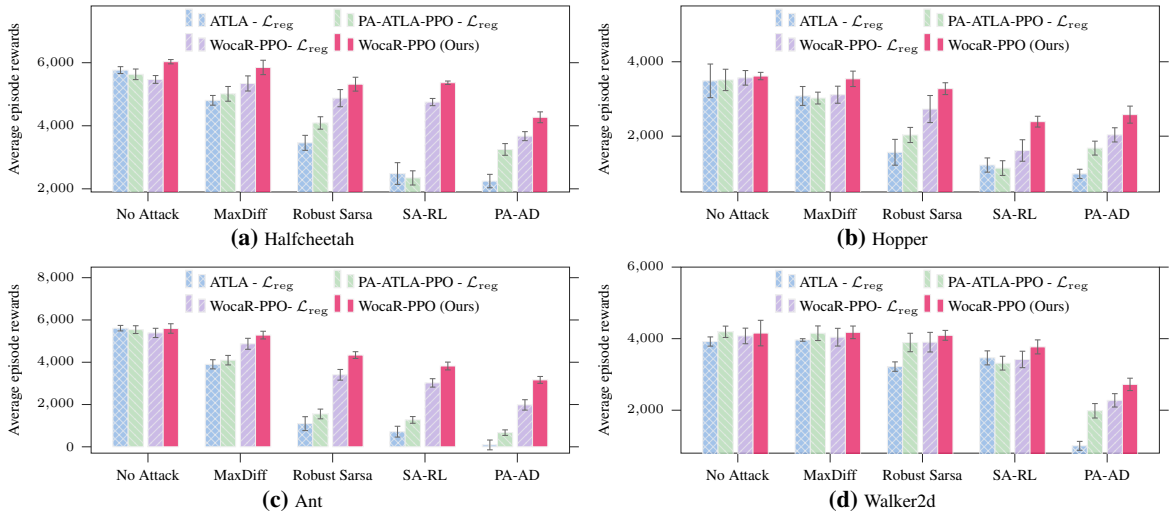


Figure 12. Ablation performance for the state regularization loss $\mathcal{L}_{\text{reg}}$ under no attack and different attacks on Hopper, Walker2d, Halfcheetah, and Ant.