CROP: CERTIFYING ROBUST POLICIES FOR REINFORCEMENT LEARNING THROUGH FUNCTIONAL SMOOTHING

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a framework of Certifying Robust Policies (CROP) for reinforcement learning against adversarial state perturbations, which provides state level robustness certification and the *first* certification for cumulative rewards. We propose two particular types of robustness certification criteria: *robustness of per-state actions* and *lower bound of cumulative rewards*. Specifically, we develop a local smoothing algorithm that uses a policy derived from Q-functions smoothed with Gaussian noise over each encountered state to guarantee the robustness of actions taken along this trajectory. Next, we develop a global smoothing algorithm for certifying the robustness of a finite-horizon cumulative reward under adversarial state perturbations. Finally, we propose a local smoothing approach that makes use of adaptive search in order to obtain tight certification bounds for the reward. We use the proposed RL robustness certification framework to evaluate methods that have previously been shown to yield empirically robust RL, including adversarial training and several forms of regularization, on three representative Atari games. We show that RegPGD, RegCVX, and RadialRL achieve high certified robustness among these. Furthermore, by evaluating these algorithms against adversarial attacks, we demonstrate that our certifications are often tight. All experiment results are available at website `https://crop-leaderboard.me`.

1 INTRODUCTION

Reinforcement learning (RL) has been widely applied to different applications, such as robotics (Kober et al., 2013; Deisenroth et al., 2013; Polydoros & Nalpantidis, 2017), autonomous driving vehicles (Shalev-Shwartz et al., 2016; Sallab et al., 2017), and trading (Deng et al., 2016; Almahdi & Yang, 2017; Ye et al., 2020). However, recent studies have shown that learning algorithms are vulnerable to adversarial attacks (Goodfellow et al., 2014; Kurakin et al., 2016; Moosavi-Dezfooli et al., 2016; Jia & Liang, 2017; Eykholt et al., 2018), and a range of attacks have also been proposed against the input states and trained policies of RL (Huang et al., 2017; Kos & Song, 2017; Lin et al., 2017; Behzadan & Munir, 2017a). As more and more safety-critical applications are being deployed in real-world (Christiano et al., 2016; Fisac et al., 2018; Cheng et al., 2019; eop), how to test and improve their robustness before massive production is of great importance.

To defend against adversarial attacks in RL, different *empirical defenses* have been proposed (Mandlekar et al., 2017; Behzadan & Munir, 2017b; Pattanaik et al., 2018; Fischer et al., 2019; Zhang et al., 2020; Oikarinen et al., 2020; Donti et al., 2020; Shen et al., 2020; Eysenbach & Levine, 2021). In particular, adversarial training (Kos & Song, 2017; Behzadan & Munir, 2017b; Pattanaik et al., 2018) and regularized RL algorithms by enforcing the smoothness of the trained models (Shen et al., 2020; Zhang et al., 2020) have been studied to improve the robustness of trained policies. However, several strong adaptive attacks have been proposed against these empirical defenses (Gleave et al., 2019; Hussenot et al., 2019; Russo & Proutiere, 2019) and it is important to provide *robustness certification* for a given learning algorithm to end such repeated game between attackers and defenders.

To provide *robustness certification*, several studies have been conducted on classification. For instance, both deterministic (Ehlers, 2017; Katz et al., 2017; Cheng et al., 2017; Tjeng et al., 2017; Weng et al., 2018; Zhang et al., 2018; Singh et al., 2019; Gehr et al., 2018; Wong & Kolter, 2018; Raghunathan et al., 2018) and probabilistic approaches (Lecuyer et al., 2019; Cohen et al., 2019; Lee et al., 2019;

Salman et al., 2019; Carmon et al., 2019; Jeong & Shin, 2020) have been explored to provide a lower bound of classification accuracy given bounded adversarial perturbation. Considering the sequential decision making property of RL, which makes it more challenging to be directly certified compared to classification, in this paper we ask: *How to provide efficient and effective robustness certification for RL algorithms? What criteria should be used to certify the robustness of RL algorithms?*

Different from classification which involves one-step prediction only, RL algorithms provide both action prediction and reward feedback, making *what to certify* and *how to certify* robustness of RL challenging. In this paper we focus on Q-learning and propose two certification criteria: *per-state action stability* and *lower bound of perturbed cumulative reward*. In particular, to certify the *per-state action stability*, we propose the *local smoothing* on each input state and therefore derive the certified radius for perturbation at each state, within which the action prediction will not be altered. To certify the *lower bound of cumulative reward*, we propose both *global smoothing* over the finite trajectory to obtain the expectation or percentile bounds given trajectories smoothed with sampled noise sequences; and *local smoothing* to calculate an absolute lower bound based on our adaptive search algorithm.

We leverage our framework to test six empirically robust RL algorithms on three representative Atari games. We show that the certified robustness depends on both the algorithm and the game properties. For instance, RadialRL (Oikarinen et al., 2020) is the most certifiably robust method on Freeway. In addition, based on the per-state certification, we observe that for some games such as Pong, some states are more certifiably robust and such pattern is periodic. This means given the information of which states are more vulnerable, it is possible to design robust algorithms to specifically focus on these vulnerable states. Based on the lower bound of perturbed cumulative reward, we show that our certification is tight by comparing our bounds with adversarial attacks for the games.

Technical Contributions. In this paper, we take an important step towards providing robustness certification for Q-learning. We make contributions on both theoretical and empirical fronts.

- We propose a framework for certifying the robustness of Q-learning algorithms, which is notably the *first* that provides the robustness certification w.r.t. the cumulative reward.
- We propose two robustness certification criteria for Q-learning algorithms, together with corresponding certification algorithms based on global and local smoothing strategies.
- We theoretically prove the certification radius for input state and lower bound of perturbed cumulative reward under bounded adversarial state perturbations.
- We conduct extensive experiments to provide certification for six empirically robust RL algorithms on three Atari games. We provide several interesting observations which would further inspire the development of robust RL algorithms.

2 PRELIMINARIES

Q-learning and Deep Q-Networks (DQNs). Markov decision processes (MDPs) are at the core of RL. Our focus is on discounted discrete-time MDPs, which are defined by tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma, d_0)$, where $\mathcal{S}$ is a set of states (each with dimensionality N), $\mathcal{A}$ represents a set of discrete actions, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function with $\mathcal{P}(\cdot)$ defining the set of probability measures, $\gamma \in [0, 1]$ is the discount factor, and $d_0 \in \mathcal{P}(\mathcal{S})$ is the distribution over the initial state. At time step t, the agent is in the state $s_t \in \mathcal{S}$. After choosing action $a_t \in \mathcal{A}$, the agent transitions to the next state $s_{t+1} \sim P(s_t, a_t)$ and receives reward $R(s_t, a_t)$. The goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that maximizes the expected cumulative reward $\mathbb{E}[\sum_t \gamma^t r_t]$.

Q-learning (Watkins & Dayan, 1992) learns an action-value function (Q-function), $Q^\star(s, a)$, which is the maximum expected cumulative reward the agent can achieve after taking action a in state s: $Q^\star(s, a) = R(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} [\max_{a'} Q^\star(s', a')]$. In deep Q-Networks (DQNs) (Mnih et al., 2013), $Q^\star$ is approximated using a neural network parametrized by θ, *i.e.*, $Q^\pi(s, a; \theta) \approx Q^\star(s, a)$. Let $\rho \in \mathcal{P}(\mathcal{S} \times \mathcal{A})$ be the observed distribution defined over states s and actions a, the network can be trained via minimizing loss function $\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{(s,a)\sim\rho, s'\sim P(s,a)} \left[\left(R(s, a) + \gamma \max_{a'} Q^\pi(s', a'; \theta) - Q^\pi(s, a; \theta) \right)^2 \right]$. The greedy policy π is defined as taking the action with highest Q^π value in each state s: $\pi(s) = \arg\max_{a\in\mathcal{A}} Q^\pi(s, a)$.

Certified Robustness for Classifiers via Randomized Smoothing. Randomized smoothing (Cohen et al., 2019) has been proposed to provide probabilistic certified robustness for classification models. It achieves state-of-the-art certified robustness on large-scale datasets such as ImageNet under ℓ_2-bounded constraints (Salman et al., 2019; Yang et al., 2020). In particular, given a base

model and a test instance, a smoothed model is constructed by outputting the most probable prediction over different Gaussian perturbed inputs.

3 ROBUSTNESS CERTIFICATION IN Q-LEARNING

In this section, we first introduce the threat model, followed by two robustness certification criteria for the Q-learning algorithm: *per-state action* and *cumulative reward*. We consider the standard adversarial setting in Q-learning (Huang et al., 2017; Kos & Song, 2017; Zhang et al., 2020), where the adversary can apply ℓ_2-bounded perturbation $\mathcal{B}^\varepsilon = \{\delta \in \mathbb{R}^n \mid \|\delta\|_2 \leq \varepsilon\}$ to input state observations of the agent during decision (test) time to cause the policy to select suboptimal actions. The agent observes the perturbed state and takes action $a' = \pi(s + \delta)$, following policy π. Following the Kerckhoff's principle (Shannon, 1949), we consider a *worst-case* adversary who applies adversarial perturbations to *every* state at decision time. Our analysis and methods are generalizable to other ℓ_p norms following (Yang et al., 2020; Lecuyer et al., 2019).

3.1 ROBUSTNESS CERTIFICATION FOR Q-LEARNING WITH DIFFERENT CRITERIA

To provide the robustness certification for Q-learning, we propose two certification criteria: *per-state action robustness* and *lower bound of the cumulative reward*.

Robustness Certification for Per-State Action. We first aim to explore the robustness (stability/consistency) of the per-state action given adversarially perturbed input states.

Definition 1 (Certification for per-state action)**.** Given a trained network Q^π with policy π, we define the robustness certification for per-state action as the *maximum perturbation magnitude* $\bar{\varepsilon}$, such that for any perturbation $\delta \in \mathcal{B}^{\bar{\varepsilon}}$, the predicted action under the perturbed state will be the same as the action taken in the clean environment, *i.e.*, $\pi(s + \delta) = \pi(s)$, $\forall \delta \in \mathcal{B}^{\bar{\varepsilon}}$.

Robustness Certification for Cumulative Reward. Given that the cumulative reward is important for RL, here in addition to the per-state action, we also define the robustness certification regarding the cumulative reward under input state perturbation.

Definition 2 (Cumulative reward)**.** Let $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ be the transition function of the environment with $\mathcal{P}(\cdot)$ defining the set of probability measures. Let $R, d_0, \gamma, Q^\pi, \pi$ be the reward function, initial state distribution, discount factor, a given trained Q-network, and the corresponding greedy policy as introduced in Section 2. $J(\pi)$ represents the *cumulative reward* and $J_\varepsilon(\pi)$ represents the *perturbed cumulative reward* under perturbations $\delta_t \in \mathcal{B}^\varepsilon$ at each time step t:

$$J(\pi) := \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)), \quad \text{where } s_{t+1} \sim P(s_t, a_t), s_0 \sim d_0, \qquad \text{and} \qquad J_\varepsilon(\pi) := \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t + \delta_t)), \quad \text{where } s_{t+1} \sim P(s_t, \pi(s_t + \delta_t)), s_0 \sim d_0. \tag{1}$$

The randomness of $J(\pi)$ arises from the environment dynamics, while that of $J_\varepsilon(\pi)$ includes additional randomness from the perturbations $\{\delta_t\}$. Since for a sufficiently large finite horizon H we can approximate $J(\pi)$ and $J_\varepsilon(\pi)$ to arbitrary precision, we henceforth focus on a finite horizon H.

Definition 3 (Robustness certification for cumulative reward)**.** The robustness certification for cumulative reward is the *lower bound* of *perturbed cumulative reward* $\underline{J}$ such that $\underline{J} \leq J_\varepsilon(\pi)$ under perturbation in $\mathcal{B}^\varepsilon = \{\delta \in \mathbb{R}^n \mid \|\delta\|_2 \leq \varepsilon\}$ applied to all time steps.

We will provide details on the certification of per-state action in Section 4 and the certification of cumulative reward in Section 5 based on different smoothing strategies and certification methods.

4 ROBUSTNESS CERTIFICATION STRATEGIES FOR PER-STATE ACTION

In this section, we discuss the robustness certification for *per-state action*, aiming to calculate a lower bound of *maximum perturbation magnitude* $\bar{\varepsilon}$ in Definition 1.

4.1 CERTIFICATION FOR PER-STATE ACTION VIA ACTION-VALUE FUNCTIONAL SMOOTHING

Let Q^π be the action-value function given by the trained network Q with policy π. We derive a smoothed function $\widetilde{Q}^\pi$ through per-state *local smoothing*. Specifically, at each time step t, for each action $a \in \mathcal{A}$, we draw random noise from a Gaussian distribution $\mathcal{N}(0, \sigma^2 I_N)$ to smooth $Q^\pi(\cdot, a)$.

$$\widetilde{Q}^\pi(s_t, a) := \mathop{\mathbb{E}}_{\Delta_t \sim \mathcal{N}(0, \sigma^2 I_N)} Q^\pi(s_t + \Delta_t, a) \quad \forall s_t \in \mathcal{S}, a \in \mathcal{A}, \text{ and } \tilde{\pi}(s_t) := \operatorname*{argmax}_a \widetilde{Q}^\pi(s_t, a) \quad \forall s_t \in \mathcal{S}. \tag{2}$$

Lemma 1 (Lipschitz continuity of the smoothed value function)**.** *Given the action-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to [V_{\min}, V_{\max}]$, the smoothed function $\widetilde{Q}^\pi$ with smoothing parameter σ is L-Lipschitz continuous with $L = \frac{V_{\max} - V_{\min}}{\sigma}\sqrt{2/\pi}$ w.r.t. the state input.*
The proof is given in Appendix A.1. Leveraging the Lipschitz continuity in Lemma 1, we derive the following theorem for certifying the robustness of per-state action.

Theorem 1. *Let $Q^\pi : \mathcal{S} \times \mathcal{A} \to [V_{\min}, V_{\max}]$ be a trained value network, $\widetilde{Q}^\pi$ be the smoothed function with (2). At time step t with state s_t, we can compute the lower bound r_t of maximum perturbation magnitude $\bar{\varepsilon}(s_t)$ (i.e., $r_t \leq \bar{\varepsilon}(s_t)$, $\bar{\varepsilon}$ defined in Definition 1) for locally smoothed policy $\tilde{\pi}$:*

$$r_t = \frac{\sigma}{2}\left(\Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t, a_1) - V_{\min}}{V_{\max} - V_{\min}}\right) - \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t, a_2) - V_{\min}}{V_{\max} - V_{\min}}\right)\right), \tag{3}$$

where Φ^{-1} is the inverse CDF function, a_1 is the action with the highest $\widetilde{Q}^\pi$ value at state s_t, and a_2 is the runner-up action. We name the lower bound r_t as certified radius for the state s_t.

The proof is omitted to Appendix A.2. The theorem provides a certified radius r_t for per-state action given smoothed policy: As long as the perturbation is bounded by r_t, *i.e.*, $\|\delta_t\|_2 \leq r_t$, the action does not change: $\tilde{\pi}(s_t + \delta_t) = \tilde{\pi}(s_t)$. To achieve high certified robustness for per-state action, Theorem 1 implies a tradeoff between value function smoothness and the margin between the values of top two actions: If a larger smoothing parameter σ is applied, the action-value function would be smoother and therefore more stable; however, it would shrink the margin between the top two action values leading to smaller certified radius. Thus, there exists a proper smoothing parameter to balance the tradeoff, which depends on the actual environments and algorithms.

4.2 CROP-LoAct: Local Randomized Smoothing for Certifying Per-State Action

Next we introduce the algorithm to achieve the certification for per-state action. Given a Q^π network, we apply (2) to derive a smoothed network $\widetilde{Q}^\pi$. At each state s_t, we obtain the greedy action $\tilde{a}_t$ w.r.t. $\widetilde{Q}^\pi$, and then compute the certified radius r_t. We present the complete algorithm in Appendix B.1.

There are some additional challenges in smoothing the value function and computing the certified radius in Q-learning compared with the standard classification task (Cohen et al., 2019). **Challenge 1:** In classification, the output range of the confidence $[0, 1]$ is known a priori; however, in Q-learning, for a given Q^π, its range $[V_{\min}, V_{\max}]$ is unknown. **Challenge 2:** In the classification task, the lower and upper bounds of the top two classes' prediction probabilities can be directly computed via the confidence interval base on multinomial proportions (Goodman, 1965). For Q-networks, the outputs are not probabilities and calculating the multinomial proportions becomes challenging.

Pre-processing. To address **Challenge 1**, we estimate the output range $[V_{\min}, V_{\max}]$ of a given network Q^π based on a finite set of valid states $\mathcal{S}_{\text{sub}} \subseteq \mathcal{S}$. In particular, we craft a sufficiently large set $\mathcal{S}_{\text{sub}}$ to estimate $V_{\min}$ and $V_{\max}$ for Q^π on $\mathcal{S}$, which can be used later in per-state smoothing. The details for estimating $V_{\min}$ and $V_{\max}$ are deferred to Appendix B.1.

Certification. To smooth a given state s_t, we use Monte Carlo sampling (Cohen et al., 2019) to sample noise applied to s_t, and then estimate the corresponding smoothed value function $\widetilde{Q}^\pi$ at s_t with (2). In particular, we sample m Gaussian noise $\Delta_i \sim \mathcal{N}(0, \sigma^2 I_N)$, clip the Q-network output to ensure that it falls within the range $[V_{\min}, V_{\max}]$, and then take the average of the output to obtain the smoothed action prediction based on $\widetilde{Q}^\pi$. We then employ Theorem 1 to compute the certified radius r_t. We omit the detailed inference procedure to Appendix B.1. To address **Challenge 2**, we leverage Hoeffding's inequality (Hoeffding, 1994) to compute a lower bound of $\widetilde{Q}^\pi(s_t, a_1)$ and an upper bound of $\widetilde{Q}^\pi(s_t, a_2)$ with one-sided confidence level parameter α given the top two actions a_1 and a_2. When the former is higher than the latter, we can certify a positive radius for the given state s_t.

5 Robustness Certification Strategies for the Cumulative Reward

In this section, we present robustness certification strategies for the cumulative reward. The goal is to provide the lower bounds for the *perturbed cumulative reward* in Definition 2. In particular, we propose both *global smoothing* and *local smoothing* strategies to certify the perturbed cumulative reward. In the global smoothing, we view the whole state trajectory as a function to smooth, which

would lead to relatively loose certification bound. We then propose the local smoothing by smoothing each state individually to obtain the absolute lower bound.

5.1 Certification of Cumulative Reward based on Global Smoothing

In contrast to Section 4 where we perform per-state smoothing to achieve the certification for per-state action, here, we aim to perform *global smoothing* on the state trajectory by viewing the entire trajectory as a function. In particular, we first derive the *expectation bound* of the cumulative reward based on global smoothing by estimating the Lipschitz constant for the cumulative reward w.r.t. the trajectories. Since the Lipschitz estimation in the expectation bound is algorithm agnostic and could lead to loose estimation bound, we subsequently propose a more practical and tighter *percentile bound*.

Definition 4 (σ-randomized trajectory and σ-randomized policy)**.** Given a state trajectory $(s_0, s_1, \ldots, s_{H-1})$ of length H where $s_{t+1} \sim P(s_t, \pi(s_t))$, $s_0 \sim d_0$, with π the greedy policy of the action-value function Q^π, we derive a *σ-randomized trajectory* as $(s'_0, s'_1, \ldots, s'_{H-1})$, where $s'_{t+1} \sim P(s'_t, \pi(s'_t + \Delta_t))$, $\Delta_t \sim \mathcal{N}(0, \sigma^2 I_N)$, and $s'_0 = s_0 \sim d_0$. We correspondingly define a *σ-randomized policy* π' based on π in the following form: $\pi'(s_t) := \pi(s_t + \Delta_t)$ where $\Delta_t \sim \mathcal{N}(0, \sigma^2 I_N)$.

Let the operator $\oplus$ concatenates given input states or noise that are added to each state. The sampled noise sequence is denoted by $\Delta = \oplus_{t=0}^{H-1} \Delta_t$, where $\Delta_t \sim \mathcal{N}(0, \sigma^2 I_N)$.

Definition 5 (Perturbed return function)**.** Let R, P, γ, d_0 be the reward function, transition function, discount factor, and initial state distribution in Definition 2. We define a bounded *perturbed return function* $F_\pi : \mathbb{R}^{H \times N} \to [J_{\min}, J_{\max}]$ representing cumulative reward with potential perturbation δ:

$$F_\pi\left(\oplus_{t=0}^{H-1} \delta_t\right) := \sum_{t=0}^{H} \gamma^t R(s_t, \pi(s_t + \delta_t)), \quad \text{where } s_{t+1} \sim P(s_t, \pi(s_t + \delta_t)), s_0 \sim d_0. \tag{4}$$

We can see when there is no perturbation ($\delta_t = \mathbf{0}$), $F_\pi(\oplus_{t=0}^{H-1} \mathbf{0}) = J(\pi)$; when there are adversarial perturbations $\delta_t \in \mathcal{B}^\varepsilon$ at each time step, $F_\pi(\oplus_{t=0}^{H-1} \delta_t) = J_\varepsilon(\pi)$, *i.e.*, *perturbed cumulative reward*.

Mean Smoothing: Expectation bound. Here we propose to sample noise sequences Δ to perform *global smoothing* for the entire state trajectory, and calculate the lower bound of the expected perturbed cumulative reward $\mathbb{E}_\Delta\left[J_\varepsilon(\pi')\right]$ under all possible ℓ_2-bounded perturbations within magnitude ε. The expectation is over the noise sequence Δ involved in the σ-randomized policy π' in Definition 4.

Lemma 2 (Lipschitz continuity of smoothed perturbed return function)**.** *Let F be the perturbed return function function defined in (4), the smoothed perturbed return function $\widetilde{F}_\pi$ is $\frac{(J_{\max} - J_{\min})}{\sigma}\sqrt{2/\pi}$-Lipschitz continuous, where $\widetilde{F}_\pi\left(\oplus_{t=0}^{H-1} \delta_t\right) := \mathop{\mathbb{E}}_{\Delta \sim \mathcal{N}(0, \sigma^2 I_{H \times N})} F_\pi\left(\oplus_{t=0}^{H-1} (\delta_t + \Delta_t)\right)$.*

Theorem 2 (Expectation bound)**.** *Let $\underline{J_E} = \widetilde{F}_\pi\left(\oplus_{t=0}^{H-1} \mathbf{0}\right) - L\varepsilon\sqrt{H}$, where $L = \frac{(J_{\max} - J_{\min})}{\sigma}\sqrt{2/\pi}$. Then $\underline{J_E} \leq \mathbb{E}\left[J_\varepsilon(\pi')\right]$.*

Proof Sketch. We first derive the equality between expected perturbed cumulative reward $\mathbb{E}\left[J_\varepsilon(\pi')\right]$ and the smoothed perturbed return function $\widetilde{F}_\pi(\oplus_{t=0}^{H-1} \delta_t)$. Thus, to lower bound the former, it suffices to lower bound the latter, which can be calculated leveraging the Lipschitz continuity of $\widetilde{F}$ in Lemma 2 (proved in Appendix A.3), noticing that the distance between $\oplus_{t=0}^{H-1} \mathbf{0}$ and the adversarial perturbations $\oplus_{t=0}^{H-1} \delta_t$ is bounded by $\varepsilon\sqrt{H}$. The complete proof is omitted to Appendix A.4.

We obtain $J_{\min}$ and $J_{\max}$ in Lemma 2 from game specifications which can be loose in practice. Thus the Lipschitz constant L estimation is coarse and mean smoothing is usually loose. We next present a method that circumvents estimating the Lipschitz constant and provides a tight percentile bound.

Percentile Smoothing: Percentile bound. We now propose to apply *percentile smoothing* to smooth the *perturbed cumulative reward* and obtain the lower bound of the p-th percentile of $J_\varepsilon(\pi')$, where π' is a σ-randomized policy defined in Definition 4.

$$\widetilde{F}_\pi^p\left(\oplus_{t=0}^{H-1} \delta_t\right) = \sup_y \left\{ y \in \mathbb{R} \mid \mathbb{P}\left[F_\pi\left(\oplus_{t=0}^{H-1} (\delta_t + \Delta_t)\right) \leq y\right] \leq p \right\}. \tag{5}$$

Theorem 3 (Percentile bound)**.** *Let $\underline{J_p} = \widetilde{F}_\pi^{p'}\left(\oplus_{t=0}^{H-1} \mathbf{0}\right)$, where $p' := \Phi\left(\Phi^{-1}(p) - \varepsilon\sqrt{H}/\sigma\right)$. Then $\underline{J_p} \leq$ the p-th percentile of $J_\varepsilon(\pi')$.*

The proof is provided in Appendix A.5 based on Chiang et al. (2020). There are several other advantages of percentile smoothing over mean smoothing. First, the certification given by percentile smoothing is among the cumulative rewards of the sampled σ-randomized trajectories and is therefore

achievable by a real-world policy, while the expectation bound is less likely to be achieved in practice given the loose Lipschitz bound. Second, for a discrete function such as perturbed return function, the output of mean smoothing is continuous w.r.t. σ, while the results given by percentile smoothing remain discrete. Thus, the percentile smoothed function preserves properties of the base function before smoothing, and shares similar interpretation, *e.g.*, the number of rounds that the agent wins. Third, taking $p = 50\%$ in percentile smoothing leads to the *median smoothing* which achieves additional properties such as robustness to outliers (Manikandan, 2011). The detailed algorithm CROP-GRE, including the inference procedure, the estimation of $\widetilde{F}_\pi$, the algorithm for computing the empirical order statistics, and the configuration of algorithm parameters $J_{\min}, J_{\max}$, are deferred to Appendix B.2.

5.2 Certification of Cumulative Reward based on Local Smoothing

Though global smoothing provides efficient and practical bounds for the perturbed cumulative reward, such bounds are still loose as they involve smoothing the entire trajectory at once. In this section, we aim to provide a tighter lower bound for $J_\varepsilon(\tilde{\pi})$ by performing *local smoothing*.

Given a trajectory of H time steps which is guided by the locally smoothed policy $\tilde{\pi}$, we can compute the certified radius at each time step according to Theorem 1, which can be denoted as $r_0, r_1, \ldots, r_{H-1}$. Recall that when the perturbation magnitude $\varepsilon < r_t$, the optimal action a_t at time step t will remain unchanged. This implies that when $\varepsilon < \min_{t=0}^{H-1} r_t$, none of the actions in the entire trajectory will be changed, and therefore the lower bound of the cumulative reward when $\varepsilon < \min_{t=0}^{H-1} r_t$ is the return of the current trajectory in a deterministic environment. Increasing ε has two effects. First, the *total* number of time steps where the action is susceptible to change will increase; second, at *each* time step, the action can change from the best to the runner-up or the rest. We next introduce an extension of certified radius r_t to characterize the two effects.

Theorem 4. *Let $(r_t^1, \ldots, r_t^{|\mathcal{A}|-1})$ be a sequence of certified radii for state s_t at time step t, where r_t^k denotes the radius such that if $\varepsilon < r_t^k$, the possible action at time step t will belong to the actions corresponding to top k action values of $\widetilde{Q}$ at state s_t. The definition of r_t in Theorem 1 is equivalent to r_t^1 here. The radii can be computed similarly as follows:*

$$r_t^k = \frac{\sigma}{2}\left(\Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t, a_1) - V_{\min}}{V_{\max} - V_{\min}}\right) - \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t, a_{k+1}) - V_{\min}}{V_{\max} - V_{\min}}\right)\right), \quad 1 \le k < |\mathcal{A}|,$$

where a_1 is the action of the highest $\widetilde{Q}$ value at state s_t and a_{k+1} is the $(k+1)$-th best action. We additionally define $r_t^0(s_t) = 0$, which is also compatible with the definition above.

We defer the proof in Appendix A.6. With Theorem 4, for any given ε, we can compute all possible actions under perturbations in $\mathcal{B}^\varepsilon$. This allows an exhaustive search to traverse all trajectories satisfying that all certified radii along the trajectory are smaller than ε. Then, we can conclude that $J_\varepsilon(\tilde{\pi})$ is lower bounded by the minimum return over all these possible trajectories.

5.2.1 CROP-LoRe: Local smoothing for Certified Reward

Given a policy π in a deterministic environment, let the initial state be s_0, we propose CROP-LoRe to certify the lower bound of $J_\varepsilon(\tilde{\pi})$. At a high-level, CROP-LoRe *exhaustively* explores new trajectories leveraging Theorem 4 with priority queue and *effectively* updates the lower bound of cumulative reward $\underline{J}$ by expanding a trajectory tree dynamically. The algorithm returns a collection of pairs $\{(\varepsilon_i, \underline{J}_{\varepsilon_i})\}_{i=1}^{|C|}$ sorted in ascending order of ε_i, where $|C|$ is the length of the collection. For any $(\varepsilon_i, \underline{J}_{\varepsilon_i})$ in the collection, our algorithm ensures that as long as the perturbation magnitude $\varepsilon \le \varepsilon_i$, the cumulative reward $J_\varepsilon(\tilde{\pi}) \ge \underline{J}_{\varepsilon_i}$. The algorithm is shown in Algorithm 3 in Appendix B.3.

Algorithm Description. The method starts from the base case: when perturbation magnitude $\varepsilon = 0$, the lower bound of cumulative reward $\underline{J}$ is exactly the benign reward. The method then gradually increases the perturbation magnitude ε (later we will explain how a new ε is determined). Along the increase of ε, the perturbation may cause the policy π to take different actions at some time steps, thus resulting in new trajectories. Thanks to the local smoothing, the method leverages Theorem 4 to figure out the exhaustive list of possible actions under current perturbation magnitude ε, and effectively explore these new trajectories by formulating them as expanded branches of a trajectory tree. Once all new trajectories are explored, the method examines all leaf nodes of the tree and figures out the minimum reward among them, which is the new lower bound of cumulative reward $\underline{J}$ under this new ε. To mitigate the explosion of branches, CROP-LoRe proposes several *optimization* tricks. In the following, we will briefly introduce the *trajectory exploration and expansion* and *growth of*

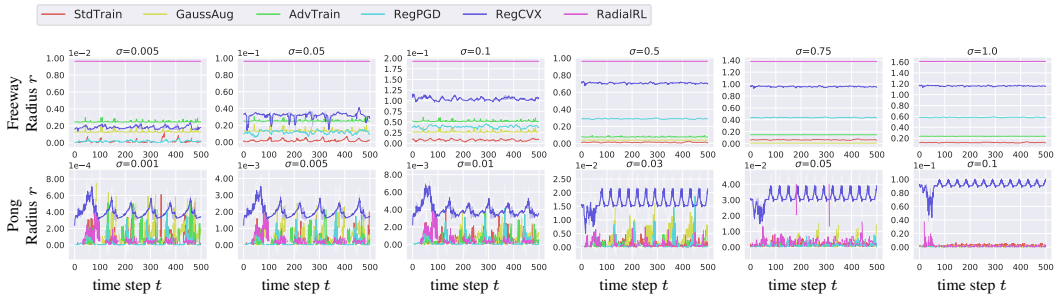


Figure 1: Robustness certification for *per-state action* in terms of certified radius r at all time steps. Each column corresponds to one smoothing variance σ. The shaded area represents the standard deviation which is small. RadialRL is the most certifiably robust method on Freeway, while RegCVX is the most robust on Pong.

perturbation magnitude steps, as well as the *optimization* tricks. More algorithm details are deferred to Appendix B.3, where we also provide an analysis on the time complexity of the algorithm.

In **trajectory exploration and expansion**, CROP-LoRe organizes all possible trajectories in the form of search tree and progressively grows it. For each node (representing a game state), leveraging Theorem 4, we compute a non-decreasing sequence $\{r^k(s)\}_{k=0}^{|\mathcal{A}|-1}$ representing the required perturbation radii for π to choose each alternative action. Suppose the current ε satisfies $r^i(s) \leq \varepsilon < r^{i+1}(s)$, we can grow $(i+1)$ branches from current state s corresponding to the original action and i alternative actions since $\varepsilon \geq r^j(s)$ for $1 \leq j \leq i$. We expand the tree branches using depth-first search (Tarjan, 1972). In **perturbation magnitude growth**, when all trajectories for perturbation magnitude ε are explored, we increase ε to seek certification under larger perturbations. This is achieved by preserving a priority queue (van Emde Boas, 1977) of the critical ε's that we will expand on. Concretely, along the trajectory, at each tree node, we search for the possible actions and store actions corresponding to $\{r^k(s)\}_{k=i+1}^{|\mathcal{A}|-1}$ into the priority queue, since these actions are exactly those need to be explored when ε grows. We repeat the procedure of *perturbation magnitude growth* (*i.e.*, popping out the head element from the queue) and *trajectory exploration and expansion* (*i.e.*, exhaustively expand all trajectories given the perturbation magnitude) until the priority queue becomes empty or the perturbation magnitude ε reaches the predefined threshold. Additionally, we adopt a few **optimization** tricks commonly used in search algorithms to reduce the complexity of the algorithm, such as pruning and the memorization technique (Michie, 1968). More potential improvements are discussed in Appendix B.3.

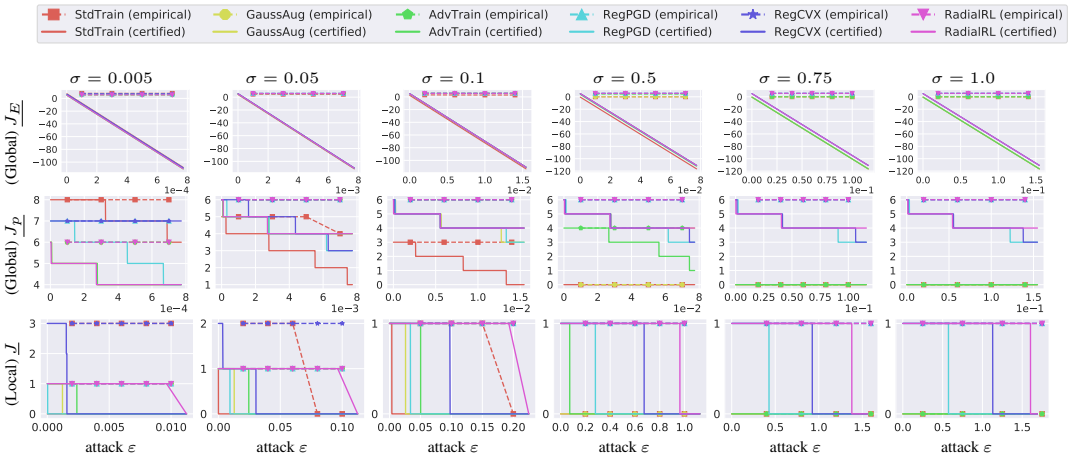
6 Experiments

In this section, we present evaluation for the proposed robustness certification framework CROP. Concretely, we apply our three certification algorithms (CROP-LoAct, CROP-GRe, and CROP-LoRe) to certify six RL methods (baseline methods **StdTrain** (Mnih et al., 2013), **GaussAug** (Kos & Song, 2017), **AdvTrain** (Behzadan & Munir, 2017b), and SOTA methods **RegPGD**, **RegCVX** (Zhang et al., 2020), **RadialRL** (Oikarinen et al., 2020)) on two high-dimensional games (Pong and Freeway) and one low dimensional game (CartPole). As a summary, we find that (1) RegCVX, RegPGD, and RadialRL achieve high certified robustness in different games; (2) Large smoothing variance can help to improve certified robustness significantly on Freeway, while a more careful selection of the smoothing parameter is needed in Pong; (3) For methods that demonstrate high certified robustness, our certification of the cumulative reward is tight. We defer the detailed descriptions of the game environments to Appendix D.1 and the introduction to the RL methods and implementation details to Appendix D.2. More interesting results and discussions are omitted to Appendix E and our leaderboard, including results on CartPole (Appendix E.6) and Breakout (Appendix E.7) and three more algorithms (CARRL, NoisyNet, and GradDQN). As a *foundational* work providing robustness certification for RL, we expect more RL algorithms and games will be certified under our framework in future work.

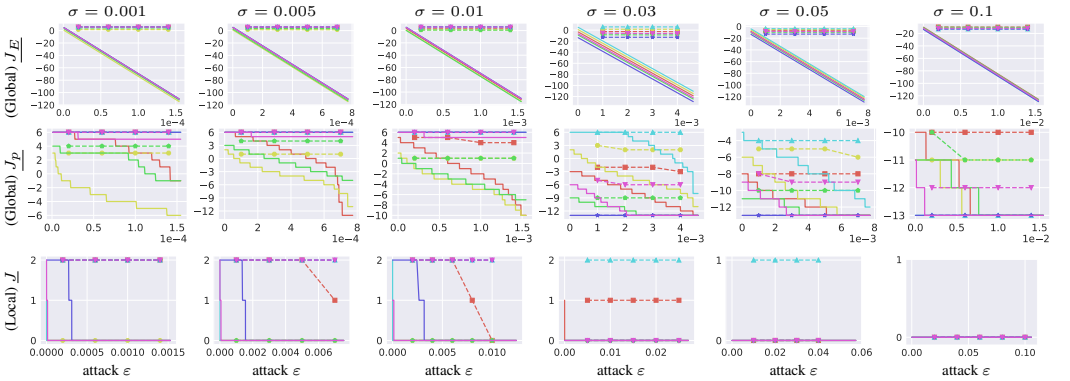
6.1 Evaluation of Robustness Certification for Per-state Action

In this subsection, we provide the robustness certification evaluation for per-state action.

Experimental Setup and Metrics. We evaluate the *locally smoothed policy* $\tilde{\pi}$ derived in (2). We follow Theorem 1 and report the *certified radius* r_t at each step t. We additionally compute *certified ratio* η *at radius* r, representing the percentage of time steps that can be certified, formally denoted as $\eta_r = {}^1\!/_H \sum_{t=1}^{H} \mathbb{1}_{\{r_t \geq r\}}$. More details are omitted to Appendix D.4.



(a) Freeway



(b) Pong

Figure 3: Robustness certification as cumulative reward, including *expectation bound* $\underline{J_E}$, *percentile bound* $\underline{J_p}$ ($p = 50\%$), and *absolute lower bound* $\underline{J}$ on (a) Freeway and (b) Pong. Each column corresponds to one smoothing variance. Solid lines represent the certified reward bounds of different methods, and dashed lines show the empirical performance under PGD.

Evaluation Results of CROP-LoAct. We present the certification comparison in Figure 1 and the benign performance under different smoothing variances in Figure 2; we omit details of certified ratio to Appendix E.1.

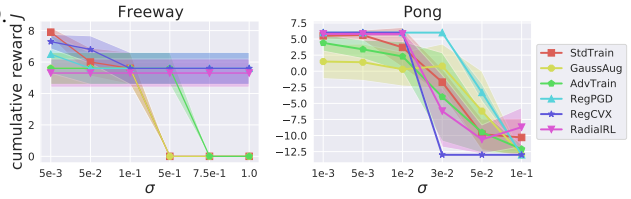


Figure 2: Benign performance of locally smoothed policy $\tilde{\pi}$ under different smoothing variance σ with clean state observations.

On *Freeway*, we evaluate with a large range of smoothing parameter σ up to 1.0, since Freeway can tolerate large noise as shown in Figure 2. Results under even larger σ are deferred to Appendix E.2, showing that the certified robustness can be further improved for the empirically robust methods. From Figure 1, we see that *RadialRL consistently achieves the highest certified radius across all σ's.* This is because RadialRL explicitly optimizes over the worst-case perturbations. StdTrain, GaussAug, and AdvTrain are not as robust, and increasing σ will not make a difference. In *Pong*, RegCVX is the most certifiably robust, which may be due to the hardness of optimizing the worst-case perturbation in Pong. More interesting, all methods present similar periodic patterns for the certified radius on different states, which would inspire further robust training methods to take the "confident state" (*e.g.*, when the ball is flying towards the paddle) into account. We illustrate the frames with varying certified radius in Appendix E.3. Overall, the certified robustness of these methods largely matches empirical observations (Behzadan & Munir, 2017b; Oikarinen et al., 2020; Zhang et al., 2020).

6.2 Evaluation of Robustness Certification for Cumulative Reward

Here we will discuss the evaluation for the robustness certification regarding cumulative reward in Section 5. We show the evaluation results for both CROP-GRe and CROP-LoRe.

Evaluation Setup and Metrics. We evaluate the *σ-randomized policy* π' derived in Definition 4 for CROP-GRE and the *locally smoothed policy* $\tilde{\pi}$ derived in (2) for CROP-LORE. We compute the expectation bound $\underline{J_E}$, percentile bound $\underline{J_p}$, and absolute lower bound $\underline{J}$ following Theorem 2, Theorem 3, and Section 5.2.1. To validate the tightness of the bounds, we additionally perform empirical attacks. More rationales see Appendix D.3. The detailed parameters are omitted to Appendix D.4.

Evaluation of CROP-GRE. We present the certification of reward in Figure 3 and mainly focus on analyzing percentile bound. We present the bound w.r.t. different attack magnitude ε under different smoothing parameter σ. For each σ, there exists an upper bound of ε that can be certified, which is positively correlated with σ. Details see Appendix B.2. We observe similar conclusion as in the per-state action certification that RadialRL is most certifiably robust for Freeway, while RegCVX and RegPGD are most robust for Pong although the highest robustness is achieved at different σ.

Evaluation of CROP-LORE. In Figure 3, we note that in *Freeway*, RadialRL is the most robust, followed by RegCVX, then RegPGD, AdvTrain, and GaussAug. As for *Pong*, RegCVX outperforms RadialRL and RegPGD. The remaining methods are not clearly ranked.

6.3 Discussion on Evaluation Results

Impact of Smoothing Parameter σ. We draw similar conclusions regarding the impact of smoothing variance from the results given by CROP-LOACT and CROP-GRE. In *Freeway*, as σ increases, the robustness of StdTrain, GaussAug, and AdvTrain barely increases, while that of RegPGD, RegCVX, and RadialRL steadily increases. In *Pong*, a σ in the range 0.01-0.03 shall be suitable for almost all methods. More explanations from the perspective of benign performance and certified results will be provided in Appendix E.4. For CROP-LORE, there are a few different conclusions. On *Freeway*, it is clear that under larger attack magnitude ε, larger smoothing parameter σ always secures higher lower bound $\underline{J}$. In *Pong*, CROP-LORE can only achieve non-zero certification with small σ under a small range of attack magnitude, implying the difficulty of establishing non-trivial certification for these methods on Pong. We defer details on the selection of σ to Appendix E.4.

Tightness of the certification $\underline{J_E}$, $\underline{J_p}$, and $\underline{J}$. We compare the empirical cumulative rewards achieved under PGD attacks with our certified lower bounds. First, the empirical results are consistently lower bounded by our certifications, validating the correctness of our bounds. Regarding the tightness, the improved $\underline{J_p}$ is much tighter than the loose $\underline{J_E}$, supported by discussions in Section 5.1. The tightness of $\underline{J}$ can be reflected by the zero gap between the certification and the empirical result under a wide range of attack magnitude ε. Furthermore, the certification for RegCVX and RegPGD are quite tight for Freeway under large attack magnitudes, and RadialRL demonstrates its superiority on Freeway. Additionally, different methods may achieve the same empirical results under attack, yet their certifications differ tremendously, indicating the importance of the robustness *certification*.

Game Properties. The certified robustness of any method on Freeway is much higher than that on Pong, indicating that Freeway is a more stable game than Pong, also shown in Mnih et al. (2015).

Comparison of Different Certification Criteria. Conclusions given by our different certification criteria are largely consistent. We discuss the *advantages*, *limitations*, and *extensions* of our certification methods in Appendix C, and provide more *detailed analysis* to help with understanding.

7 Related Work

We briefly review several categories of RL methods that demonstrate empirical robustness. *Randomization methods* (Tobin et al., 2017; Akkaya et al., 2019) were first proposed to encourage exploration. Under the *adversarial training* framework, Kos & Song (2017) and Behzadan & Munir (2017b) show that re-training with random noise and FGSM perturbations increases the agent's resilience. SA-DQN (Zhang et al., 2020) is a *regularization* based method that encourages the top-1 action to stay unchanged under perturbation. Radial-RL (Oikarinen et al., 2020) minimizes an adversarial loss function that incorporates the upper bound of the perturbed loss. CARRL (Everett et al., 2021) computes the lower bounds of Q values under perturbation for action selection, but it is only suitable for low-dimensional environments. Most of these works only provide empirical robustness, with Zhang et al. (2020) and Fischer et al. (2019) additionally provide robustness certificates at state level. To the best of our knowledge, this is the *first* work that provides robustness certification for the cumulative reward of RL methods. Broader discussions and comparisons of related work are in Appendix F.

Conclusions. To provide the robustness certification for RL methods, we propose a general framework CROP, aiming to provide certification based on two criteria. Our evaluations show that certain empirically robust RL methods are certifiably robust for specific games.

Ethics Statement. In this paper, we prove the first framework for certifying the robustness of RL algorithms against evasion attacks. We aim to certify and therefore potentially improve the trustworthiness of machine learning models, and we do not expect any ethics issues raised by our work.

Reproducibility Statement. All theorem statements are substantiated with rigorous proofs in our Appendix. We have upload the source code as the supplementary material for reproducibility purpose.

REFERENCES

Promoting the use of trustworthy artificial intelligence in the federal government. https://www.federalregister.gov/documents/2020/12/08/2020-27065/promoting-the-use-of-trustworthy-artificial-intelligence-in-the-federal-government. Accessed: 2021-05-20.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Saud Almahdi and Steve Y Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017.

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

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

Vahid Behzadan and Arslan Munir. Mitigation of policy manipulation attacks on deep q-networks with parameter-space noise. In *International Conference on Computer Safety, Reliability, and Security*, pp. 406–417. Springer, 2018.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. *arXiv preprint arXiv:1905.13736*, 2019.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 251–268. Springer, 2017.

Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3387–3395, 2019.

Ping-yeh Chiang, Michael Curry, Ahmed Abdelkader, Aounon Kumar, John Dickerson, and Tom Goldstein. Detection as regression: Certified object detection with median smoothing. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1275–1286. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0dd1bc593a91620daecf7723d2235624-Paper.pdf.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

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

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

Priya L Donti, Melrose Roderick, Mahyar Fazlyab, and J Zico Kolter. Enforcing robust control guarantees within neural network policies. *arXiv preprint arXiv:2011.08105*, 2020.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286. Springer, 2017.

Michael Everett, Björn Lütjens, and Jonathan P How. Certifiable robustness to adversarial state uncertainty in deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

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

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2018.

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

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Leo A Goodman. On simultaneous confidence intervals for multinomial proportions. *Technometrics*, 7(2):247–254, 1965.

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

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pp. 409–426. Springer, 1994.

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

Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Copycat: Taking control of neural policies with constant attacks. *arXiv preprint arXiv:1905.12282*, 2019.

Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2): 257–280, 2005.

Jongheon Jeong and Jinwoo Shin. Consistency regularization for certified robustness of smoothed classifiers. *arXiv preprint arXiv:2006.04062*, 2020.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

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

Aounon Kumar, Alexander Levine, Soheil Feizi, and Tom Goldstein. Certifying confidence via randomized smoothing. 2020.

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

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

Guang-He Lee, Yang Yuan, Shiyu Chang, and Tommi S Jaakkola. Tight certificates of adversarial robustness for randomly smoothed classifiers. *arXiv preprint arXiv:1906.04948*, 2019.

Linyi Li, Xiangyu Qi, Tao Xie, and Bo Li. Sok: Certified robustness for deep neural networks. *arXiv preprint arXiv:2009.04131*, 2020.

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

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

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

S Manikandan. Measures of central tendency: Median and mode. *Journal of pharmacology and pharmacotherapeutics*, 2(3):214, 2011.

Donald Michie. "memo" functions and machine learning. *Nature*, 218(5136):19–22, 1968.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586. PMLR, 2018.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

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

Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *17th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2018*, pp. 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2018.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

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

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/3a24b25a7b092a252166a1641ae953e7-Paper.pdf.

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

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28 (4):656–715, 1949.

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

Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019.

Charles Stein et al. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*. The Regents of the University of California, 1972.

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

Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2): 146–160, 1972.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Peter van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information processing letters*, 6(3):80–82, 1977.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pp. 5276–5285. PMLR, 2018.

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

Kaidi Xu, Zhouxing Shi, Huan Zhang, Minlie Huang, K Chang, Bhavya Kailkhura, Xue Lin, and C Hsieh. Automatic perturbation analysis on general computational graphs. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2020.

Greg Yang, Tony Duan, J Edward Hu, Hadi Salman, Ilya Razenshteyn, and Jerry Li. Randomized smoothing of all shapes and sizes. In *International Conference on Machine Learning*, pp. 10693–10705. PMLR, 2020.

Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1112–1119, 2020.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31:4939–4948, 2018.

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

Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=sCZbhBvqQaU.

The appendices are organized as follows:

- In Appendix A, we present the detailed proofs for lemmas and theorems in Section 4 and Section 5, laying down the basis for the design of our certification strategies.
- In Appendix B, we present some additional details of our three certification strategies: CROP-LOACT to certify the per-state action, and CROP-GRE and CROP-LORE to certify the cumulative reward. We include the detailed algorithm description and the complete pseudocode for each algorithm. We will make our implementation public upon acceptance.
- In Appendix C, we provide a discussion on the *advantages* and *limitations* of different certification methods, and also present several promising direct *extensions* for future work. We also provide more *detailed analysis* that help with the understanding of our algorithms.
- In Appendix D, we show the experimental details, including the game environment, details descriptions of the RL methods we evaluate, rationales for our evaluation settings and experimental designs, detailed evaluation setup for each of our certification algorithms in the main paper, as well as the setup for a new game CartPole.
- In Appendix E, we present additional evaluation results and discussions for the games and algorithms evaluated in the main paper, from the perspectives of different metrics and different ranges of parameters. We also provide the running time statistics and specifically illustrate the periodic patterns in the Pong game. In addition, we present the evaluation results on the game CartPole, and the results for CROP-LORE on a more complicated Atari game Breakout.
- In Appendix F, we provide a broader discussion of the related works, spanning the evasion attacks in RL, Robust RL, as well as robustness certification for RL.

A PROOFS

A.1 PROOF OF LEMMA 1

We recall Lemma 1:

Lemma 1 (Lipschitz continuity of the smoothed value function)**.** *Given the action-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to [V_{\min}, V_{\max}]$, the smoothed function $\widetilde{Q}^\pi$ with smoothing parameter σ is L-Lipschitz continuous with $L = \frac{V_{\max} - V_{\min}}{\sigma}\sqrt{2/\pi}$ w.r.t. the state input.*

Proof. To prove Lemma 1, we leverage the technique in the proof for Lemma 1 of Salman et al. (2019) in their Appendix A.

For each action $a \in \mathcal{A}$, our smoothed value function is

$$\widetilde{Q}^\pi(s,a) := \mathop{\mathbb{E}}_{\Delta \sim \mathcal{N}(0,\sigma^2 I_N)} Q^\pi(s+\Delta, a) = \frac{1}{(2\pi)^{N/2}\sigma^N}\int_{\mathbb{R}^n} Q^\pi(t,a)\exp\left(-\frac{1}{2\sigma^2}\|s-t\|^2\right)\,dt.$$

Taking the gradient w.r.t. s, we obtain

$$\nabla_s \widetilde{Q}^\pi(s,a) = \frac{1}{(2\pi)^{N/2}\sigma^N}\int_{\mathbb{R}^n} Q^\pi(t,a)\frac{1}{\sigma^2}(s-t)\exp\left(-\frac{1}{2\sigma^2}\|s-t\|^2\right)\,dt.$$

For any unit direction u, we have

$$\begin{aligned}
u \cdot \nabla_s \widetilde{Q}^\pi(s,a) &\le \frac{1}{(2\pi)^{N/2}\sigma^N}\int_{\mathbb{R}^n} \frac{V_{\max}-V_{\min}}{\sigma^2}|u(s-t)|\exp\left(-\frac{1}{2\sigma^2}\|s-t\|^2\right)\,dt \\
&= \frac{V_{\max}-V_{\min}}{\sigma^2}\cdot\int_{\mathbb{R}^n}\frac{1}{(2\pi)^{1/2}\sigma}|s_i-t|\exp\left(-\frac{1}{2\sigma^2}|s_i-t|^2\right)\,dt \\
&\quad\cdot\prod_{j\neq i}\int_{\mathbb{R}^n}\frac{1}{(2\pi)^{1/2}\sigma}\exp\left(-\frac{1}{2\sigma^2}|s_j-t|^2\right)\,dt \\
&= \frac{V_{\max}-V_{\min}}{\sigma}\sqrt{\frac{2}{\pi}}
\end{aligned}$$

Thus, $\widetilde{Q}^\pi$ is L-Lipschitz continuous with $L = \frac{V_{\max}-V_{\min}}{\sigma}\sqrt{2/\pi}$ w.r.t. the state input.

A.2 PROOF OF THEOREM 1

We recall Theorem 1:

Theorem 1. *Let $Q^\pi : \mathcal{S} \times \mathcal{A} \to [V_{\min}, V_{\max}]$ be a trained value network, $\widetilde{Q}^\pi$ be the smoothed function with (2). At time step t with state s_t, we can compute the lower bound r_t of maximum perturbation magnitude $\bar{\varepsilon}(s_t)$ (i.e., $r_t \leq \bar{\varepsilon}(s_t)$, defined in Definition 1) for locally smoothed policy $\widetilde{\pi}$:*

$$r_t = \frac{\sigma}{2}\left(\Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t,a_1) - V_{\min}}{V_{\max} - V_{\min}}\right) - \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t,a_2) - V_{\min}}{V_{\max} - V_{\min}}\right)\right) \tag{6}$$

where Φ^{-1} is the inverse CDF function, a_1 is the action with the highest $\widetilde{Q}^\pi$ value at state s_t, and a_2 is the runner-up action. We name the lower bound r_t as certified radius for the state s_t.

We first present a lemma that can help in the proof of Theorem 1.

Lemma 3. *Let Φ be the CDF of a standard normal distribution, the mapping $\eta_a(s) := \sigma \cdot \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s,a) - V_{\min}}{V_{\max} - V_{\min}}\right)$ is 1-Lipschitz continuous.*

The lemma can be proved following the same technique as the proof for Lemma 1 in Appendix A.1. The detailed proof can be referred to in the proof for Lemma 2 of Salman et al. (2019) in their Appendix A. We next show how to leverage Lemma 3 to prove Theorem 1.

Proof. [Proof for Theorem 1] Let the perturbation be δ_t, based on the Lipschitz continuity of the mapping η, we have

$$\eta_{a_1}(s_t) - \eta_{a_1}(s_t + \delta_t) \leq \|\delta_t\|_2, \tag{7}$$

$$\eta_{a_2}(s_t + \delta_t) - \eta_{a_2}(s_t) \leq \|\delta_t\|_2. \tag{8}$$

Suppose that under perturbation δ_t, the action selection would be misled in the sense that the smoothed value for the original action a_1 is lower than that of another action a_2, *i.e.*, $\widetilde{Q}^\pi(s_t + \delta_t, a_1) \leq \widetilde{Q}^\pi(s_t + \delta_t, a_2)$. Then, based on the monotonicity of η, we have

$$\eta_{a_1}(s_t + \delta) \leq \eta_{a_2}(s_t + \delta). \tag{9}$$

Summing up (7), (8), and (9), we obtain

$$\begin{aligned}\|\delta_t\|_2 &\geq \frac{1}{2}\left(\eta_{a_1}(s_t) - \eta_{a_2}(s_t)\right) \\ &= \frac{\sigma}{2}\left(\Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t,a_1) - V_{\min}}{V_{\max} - V_{\min}}\right) - \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t,a_2) - V_{\min}}{V_{\max} - V_{\min}}\right)\right)\end{aligned}$$

which is a lower bound of the *maximum perturbation magnitude* $\bar{\varepsilon}(s_t)$ that can be tolerated at state s_t. Hence, when r_t takes the value of the computed lower bound, it satisfies the condition that $r_t \leq \bar{\varepsilon}(s_t)$.

A.3 PROOF OF LEMMA 2

We recall Lemma 2:

Lemma 2 (Lipschitz continuity of smoothed perturbed return function)**.** *Let F be the perturbed return function function defined in (4), the smoothed perturbed return function $\widetilde{F}_\pi$ is $\frac{(J_{\max} - J_{\min})}{\sigma}\sqrt{2/\pi}$-Lipschitz continuous, where $\widetilde{F}_\pi\left(\oplus_{t=0}^{H-1}\delta_t\right) := \mathop{\mathbb{E}}_{\Delta \sim \mathcal{N}(0, \sigma^2 I_{H \times N})} F_\pi\left(\oplus_{t=0}^{H-1}(\delta_t + \Delta_t)\right)$.*

Proof. The proof can be done in a similar fashion as the proof for Lemma 1 in Appendix A.1. Compared with the smoothed value network where the expectation is taken over the sampled states, here, similarly, the smoothed perturbed return function is derived by taking the expectation over sampled σ-randomized trajectories. The difference is that the output range of the Q-network is $[V_{\min}, V_{\max}]$, while the output range of the perturbed return function is $[J_{\min}, J_{\max}]$. Thus, the smoothed perturbed return function $\widetilde{F}$ is $\frac{(J_{\max} - J_{\min})}{\sigma}\sqrt{2/\pi}$-Lipschitz continuous.

A.4 PROOF OF THEOREM 2

We recall the definition of $\widetilde{F}_\pi$ as well as Theorem 2:

$$\widetilde{F}_\pi\left(\oplus_{t=0}^{H-1}\delta_t\right) := \mathop{\mathbb{E}}_{\zeta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})} F_\pi\left(\oplus_{t=0}^{H-1}(\delta_t+\zeta_t)\right).$$

Theorem 2 (Expectation bound)**.** *Let* $\underline{J_E} = \widetilde{F}_\pi\left(\oplus_{t=0}^{H-1}\mathbf{0}\right) - L\varepsilon\sqrt{H}$*, where* $L = \frac{(J_{\max}-J_{\min})}{\sigma}\sqrt{2/\pi}$*. Then* $\underline{J_E} \leq \mathbb{E}\left[J_\varepsilon(\pi')\right]$*.*

Proof. We note the following equality

$$\widetilde{F}_\pi(\oplus_{t=0}^{H-1}\delta_t) \overset{(a)}{=} \mathbb{E}\left[F_\pi\left(\oplus_{t=0}^{H-1}(\delta_t+\zeta_t)\right)\right] \overset{(b)}{=} \mathbb{E}\left[F_{\pi'}\left(\oplus_{t=0}^{H-1}\delta_t\right)\right] \overset{(c)}{=} \mathbb{E}\left[J_\varepsilon^H(\pi')\right], \tag{10}$$

where (a) comes from the definition of the smoothed perturbed return function $\widetilde{F}_\pi$, (b) is due to the definition of the σ-randomized policy π', and (c) arises from the definition of the perturbed cumulative reward J_ε^H. Thus, the expected perturbed cumulative reward $\mathbb{E}\left[J_\varepsilon^H(\pi')\right]$ is equivalent to the smoothed perturbed return function $\widetilde{F}_\pi\left(\oplus_{t=0}^{H-1}\delta_t\right)$. Furthermore, since the distance between the all-zero $\oplus_{t=0}^{H-1}\mathbf{0}$ and the adversarial perturbations $\oplus_{t=0}^{H-1}\delta_t$ is bounded by $\varepsilon\sqrt{H}$, leveraging the Lipschitz smoothness of $\widetilde{F}$ in Lemma 2, we obtain the lower bound of the expected perturbed cumulative reward $\mathbb{E}\left[J_\varepsilon^H(\pi')\right]$ as $\widetilde{F}_\pi\left(\oplus_{t=0}^{H-1}\mathbf{0}\right) - L\varepsilon\sqrt{H}$.

A.5 Proof of Theorem 3

We recall the definition of $\widetilde{F}_\pi^p$ as well as Theorem 3:

$$\widetilde{F}_\pi^p\left(\oplus_{t=0}^{H-1}\delta_t\right) := \sup_y\left\{y\in\mathbb{R} \mid \mathbb{P}\left[F_\pi\left(\oplus_{t=0}^{H-1}(\delta_t+\zeta_t)\right)\leq y\right]\leq p\right\}. \tag{11}$$

Theorem 3 (Percentile bound)**.** *Let* $\underline{J_p} = \widetilde{F}_\pi^{p'}\left(\oplus_{t=0}^{H-1}\mathbf{0}\right)$*, where* $p' := \Phi\left(\Phi^{-1}(p) - {}^{\varepsilon\sqrt{H}}/_{\sigma}\right)$*. Then* $\underline{J_p} \leq$ *the* p*-th percentile of* $J_\varepsilon(\pi')$*.*

Proof. To prove Theorem 3, we leverage the technique in the proof for Lemma 2 of Chiang et al. (2020) in their Appendix B.

For brevity, we abbreviate $\delta := \oplus_{t=0}^{H-1}\delta_t$ and redefine the plus operator such that $\delta+\zeta := \oplus_{t=0}^{H-1}(\delta_t+\zeta_t)$. Then, similar to Lemma 3, we have the conclusion that

$$\delta \mapsto \sigma\cdot\Phi^{-1}\left(\mathbb{P}\left[F_\pi(\delta+\zeta)\leq \widetilde{F}_\pi^{p'}(\mathbf{0})\right]\right)$$

is 1-Lipschitz continuous, where $\zeta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})$.

Thus, under the perturbations $\delta_t\in\mathcal{B}^\varepsilon$ for $t=0\ldots H-1$, we have

$$\begin{aligned}
\Phi^{-1}\left(\mathbb{P}\left[F_\pi(\delta+\zeta)\leq \widetilde{F}_\pi^{p'}(\mathbf{0})\right]\right) &\leq \Phi^{-1}\left(\mathbb{P}\left[F_\pi(\zeta)\leq \widetilde{F}_\pi^{p'}(\mathbf{0})\right]\right) + \frac{\|\delta\|_2}{\sigma} \\
&\leq \Phi^{-1}\left(\mathbb{P}\left[F_\pi(\zeta)\leq \widetilde{F}_\pi^{p'}(\mathbf{0})\right]\right) + \frac{\varepsilon\sqrt{H}}{\sigma} && \textit{(Since } \|\delta\|_2\leq\varepsilon\sqrt{H}\textit{)} \\
&= \Phi^{-1}(p') + \frac{\varepsilon\sqrt{H}}{\sigma} && \textit{(By definition of } \widetilde{F}_\pi^p\textit{)} \\
&= \Phi^{-1}(p) && \textit{(By definition of } p'\textit{).}
\end{aligned}$$

Since Φ^{-1} monotonically increase, this implies that $\mathbb{P}\left[F_\pi(\delta+\zeta)\leq \widetilde{F}_\pi^{p'}(\mathbf{0})\right]\leq p$. According to the definition of $\widetilde{F}_\pi^p$ in (11), we see that $\widetilde{F}_\pi^{p'}(\mathbf{0})\leq \widetilde{F}_\pi^p(\delta)$, *i.e.*, $\underline{J_p}\leq$ the p-th percentile of $J_\varepsilon(\pi')$. Hence, the theorem is proved.

A.6 Proof of Theorem 4

We recall Theorem 4:

Theorem 4. *Let* $(r_t^1,\ldots,r_t^{|\mathcal{A}|-1})$ *be a sequence of certified radii for state* s_t *at time step* t*, where* r_t^k *denotes the radius such that if* $\varepsilon < r_t^k$*, the possible action at time step* t *will belong to the actions corresponding to top* k *action values of* $\widetilde{Q}$ *at state* s_t*. The definition of* r_t *in Theorem 1 is equivalent to* r_t^1 *here. The radii can be computed similarly as follows:*

$$r_t^k = \frac{\sigma}{2}\left(\Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t, a_1) - V_{\min}}{V_{\max} - V_{\min}}\right) - \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s_t, a_{k+1}) - V_{\min}}{V_{\max} - V_{\min}}\right)\right), \quad 1 \le k < |\mathcal{A}|,$$

where a_1 is the action of the highest $\widetilde{Q}$ value at state s_t and a_{k+1} is the $(k+1)$-th best action. We additionally define $r_t^0(s_t) = 0$, which is also compatible with the definition above.

Proof. Replacing a_2 with a_{k+1} in the proof for Theorem 1 in Appendix A.2 directly leads to Theorem 4.

B Additional Details of Certification Strategies

In this section, we cover the concrete details regarding the implementation of our three certification strategies, as a complement to the high-level ideas introduced back in Section 4 and Section 5.

B.1 Detailed Algorithm of CROP-LoAct

Algorithm 1: CROP-LoAct: Local smoothing for certifying per-state action

Input: state s, trained value network Q^π with range $[V_{\min}, V_{\max}]$; parameters for smoothing: sampling times m, smoothing variance σ^2, one-sided confidence parameter α

Output: smoothed value network $\widetilde{Q}^\pi$, selected action a, certification indicator $cert$, certified radius r constant L

▷ `Step 1: smoothing`

Generate noise samples $\delta_i \sim \mathcal{N}(0, \sigma^2 I)$ for $1 \le i \le m$

for *each action* $a \in \mathcal{A}$ **do**

  ▷ `clipping and averaging`

  $\widetilde{Q}^\pi(s, a) \leftarrow \frac{1}{m}\sum_{i=1}^{m} \texttt{clip}(Q^\pi(s + \delta_i, a), \min = V_{\min}, \max = V_{\max})$

$a_1, a_2 \leftarrow$ best action and runner-up action given by $\widetilde{Q}^\pi$

▷ `Step 2: certification`

$\Delta = (V_{\max} - V_{\min})\sqrt{\frac{1}{2m}\ln\frac{1}{\alpha}}$

  ▷ `confidence interval`

if $\widetilde{Q}^\pi(s, a_1) \ge \widetilde{Q}^\pi(s, a_2) + 2\Delta$ **then**

  ▷ `certification success`

  $cert \leftarrow$ `True`

  $r \leftarrow \frac{\sigma}{2}\left(\Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s,a_1) - \Delta - V_{\min}}{V_{\max} - V_{\min}}\right) - \Phi^{-1}\left(\frac{\widetilde{Q}^\pi(s,a_2) + \Delta - V_{\min}}{V_{\max} - V_{\min}}\right)\right)$

else

  ▷ `certification failure`

  $cert \leftarrow$ `False`

  $r \leftarrow$ `undefined`

return $\widetilde{Q}^\pi, a_1, cert, r$

We present the concrete algorithm of CROP-LoAct in Algorithm 1 for the procedures introduced in Section 4.2. For each given state s_t, we first perform Monte Carlo sampling (Cohen et al., 2019; Lecuyer et al., 2019) to achieve local smoothing. Based on the smoothed value function $\widetilde{Q}^\pi$, we then compute the robustness certification for per-state action, *i.e.*, the certified radius r_t at the given state s_t, following Theorem 1.

Detailed Inference Procedure. During inference, we invoke the model with m samples of Gaussian noise at each time step, and use the averaged Q value on these m noisy samples to obtain the greedy action selection and compute the certification. This procedure is similar to Predict in Cohen et al. (2019), but since RL involves multiple step decisions, we do not have the "abstain" decision as in Predict in Cohen et al. (2019), *i.e.*, CROP-LoAct will take the greedy action at all steps no matter whether the action can be certified or not.

Estimation of the Algorithm Parameters $V_{\min}, V_{\max}$. Our estimate is obtained via sampling the trajectories and calculating the Q values associated with the state-action pairs along these trajectories. Since we perform clipping using the obtained bounds, the certification is sound. The cost for estimating $V_{\min}$ and $V_{\max}$ is not high given that they can be directly obtained via sampling as concretely described above. Furthermore, we can balance the trade-off between the accuracy and efficiency of the estimation, since for any configuration of $V_{\min}$ and $V_{\max}$, the certification will invariantly be sound (reason above).

Algorithm 2: CROP-GRE: Global smoothing for certifying cumulative reward

Input: initial state distribution d_0, trained value network Q^π, game cumulative reward range $[J_{\min}, J_{\max}]$, number of steps in an episode H, perturbation magnitude at each state ε, percentile p; parameters for smoothing: sampling times m, smoothing variance σ^2, one-sided confidence parameter α

Output: Expectation bound $\underline{J_E}$, p-th percentile bound $\underline{J_p}$

▷ Step 1: smoothing

1 **for** $i = 1$ *to* m **do**

2 $\quad s_0 \sim d_0, J_i^C \leftarrow 0$ ▷ initialization

3 $\quad$ Generate macro-state noise $\delta_t \sim \mathcal{N}(0, \sigma^2 I)$ for $0 \leq t < H$

4 $\quad$ **for** $t = 0$ *to* $H-1$ **do**

5 $\quad\quad a_t \leftarrow \text{argmax}_a Q(s_t + \delta_t, a)$

6 $\quad\quad$ Execute action a_t and observe reward re_t and next state s_{t+1} ▷ take a step

7 $\quad\quad J_i^C \leftarrow J_i^C + re_t$ ▷ accumulate the reward

▷ Step 2.1: certifying expectation bound

8 $\widetilde{F} \leftarrow \frac{1}{m}\sum_{i=1}^m J_i^C$ ▷ smoothed actual reward

9 $\Delta_{\text{conf}} \leftarrow (R_{\max} - R_{\min})\sqrt{\frac{\ln(1/\alpha)}{2m}}$ ▷ confidence interval

10 $\Delta_{\text{lip}} \leftarrow \frac{R_{\max} - R_{\min}}{\sigma}\sqrt{\frac{2}{\pi}} \cdot \varepsilon\sqrt{H}$ ▷ bound given by Lipschitz continuity

11 $\underline{J_E} \leftarrow \widetilde{F} - \Delta_{\text{conf}} - \Delta_{\text{lip}}$

12

▷ Step 2.2: certifying percentile bound

13 $k \leftarrow \textsc{ComputeOrderStats}(\varepsilon, \sigma, p, m, H)$

14 $\underline{J_p} \leftarrow k$-th smallest value in $\{J_i^C\}_{i=1}^m$

15

16 **return** $\underline{J_E}, \underline{J_p}$

B.2 Detailed Algorithm of CROP-GRE

We present the concrete algorithm of CROP-GRE in Algorithm 2 for the procedures introduced in Section 5.1. Similarly to CROP-LoAct, the algorithm also consists of two parts: performing smoothing and computing certification, where we compute both the expectation bound $\underline{J_E}$ and the percentile bound $\underline{J_p}$.

Step 1: Global smoothing. We adopt Monte Carlo sampling (Cohen et al., 2019; Lecuyer et al., 2019) to estimate the smoothed perturbed return function $\widetilde{F}$ by sampling multiple σ-randomized trajectories via drawing m noise sequences. For each noise sequence $\zeta \sim \mathcal{N}(0, \sigma^2 I_{H\times N})$, we apply noise ζ_t to the input state s_t sequentially, and obtain the sum of the reward $J_i^C = \sum_{t=0}^{H-1} re_t$ as the return for this σ-randomized trajectory. We then aggregate the smoothed perturbed return values $\{J_i^C\}_{i=1}^m$ via mean smoothing and percentile smoothing.

Step 2: Certification for perturbed cumulative reward. First, we compute the *expectation bound* $\underline{J_E}$ using Theorem 2. Since the smoothed perturbed return function $\widetilde{F}$ is obtained based on m sampled noise sequences, we use Hoeffding's inequality (Hoeffding, 1994) to compute the lower bound of the random variable $\widetilde{F}\left(\oplus_{t=0}^{H-1}\mathbf{0}\right)$ with a confidence level α. We then calculate the lower bound of $\widetilde{F}\left(\oplus_{t=0}^{H-1}\delta_t\right)$ under all possible ℓ_2-bounded perturbations $\delta_t \in \mathcal{B}^\varepsilon$ leveraging the smoothness of $\widetilde{F}$.

We then compute the *percentile bound* $\underline{J_p}$ using Theorem 3. We let J_i^C be sorted increasingly, and perform normal approximations (Stein et al., 1972) to compute the largest empirical order statistic J_k^C such that $\mathbb{P}\left[\underline{J_p} \geq J_k^C\right] \geq 1-\alpha$. The empirical order statistic J_k^C is then used as the proxy of $\underline{J_p}$ under α confidence level. We next provide detailed explanations for $\textsc{ComputeOrderStats}$, which aims to compute the order k using binomial formula plus normal approximation.

Estimation of $\widetilde{F}_\pi$ (defined in Lemma 2). For computing the expectation bound (*i.e.*, the lower bound of $\mathbb{E}_\Delta\left[J_\varepsilon(\pi')\right]$), an intermediate step is to estimate $\widetilde{F}_\pi$. However, the *accuracy* of the estimation does not influence the *soundness* of the lower bound calculation. The reason is given below. The number of sampled randomized trajectories m controls the trade-off between the estimation *efficiency* and *accuracy*, as well as the *tightness* of the derived lower bound. Concretely, a small m would provide high efficiency, low estimation accuracy, and loose lower bound; but the lower bound is always *sound*, since our algorithm explicitly accounts for the *inaccuracy* associated with m via leveraging the Hoeffding's inequality in line 9 in Algorithm 2.

Details of ComputeOrderStats. We consider the sorted sequence $J_1^C \leq J_2^C \leq \cdots \leq J_m^C$. We additionally set $J_0^C = -\infty$ and $J_{m+1}^C = \infty$. Our goal is to find the largest k such that $\mathbb{P}\left[\underline{J_p} \geq J_k^C\right] \geq 1-\alpha$. We evaluate the probability explicitly as follows:

$$\mathbb{P}\left[\underline{J_p} \geq J_k^C\right] = \sum_{i=k}^{m} \mathbb{P}\left[J_i^C \leq \underline{J_p} < J_{i+1}^C\right] = \sum_{i=k}^{m} \binom{m}{i} (p')^i (1-p')^{m-i}.$$

Thus, the condition $\mathbb{P}\left[\underline{J_p} \geq J_k^C\right] \geq 1-\alpha$ is equivalent to

$$\sum_{i=0}^{k-1} \binom{m}{i} (p')^i (1-p')^{m-i} \leq \alpha. \tag{12}$$

Given large enough m, the LHS of (12) can be approximated via a normal distribution with mean equal to mp' and variance equal to $mp'(1-p')$. Concretely, we perform binary search to find the largest k that satisfies the constraint.

We finally explain the upper bound of ε that can be certified for each given smoothing variance. In practical implementation, for a given sampling number m and confidence level parameter α, if p' is too small, then the condition (12) may not be satisfied even for $k = 1$. This implies that the existence of an upper bound of ε that can be certified for each smoothing parameter σ, recalling that $p' := \Phi\left(\Phi^{-1}(p) - {}^{\varepsilon\sqrt{H}}/_{\sigma}\right)$.

Detailed Inference Procedure. Instead of sampling m noisy states per time step as in CROP-LoAct, we sample m randomized trajectories in CROP-GRe. For each sampled randomized trajectory, at each time step in the trajectory, we invoke the model with 1 sample of Gaussian noise. Given the m randomized trajectories and the cumulative reward for each of them, we compute the certification using Theorem 2 and Theorem 3.

The Algorithm Parameters $J_{\min}, J_{\max}$. We use the default values of $J_{\min}, J_{\max}$ in the game specifications rather than estimate them, which are independent with the trained models. Thus there is no computational cost at all for obtaining the two parameters. As mentioned under Theorem 2 in Section 5.1, using these default values may induce a loose bound in practice, so we further propose the percentile smoothing that aims to eliminate the dependency of our certification on $J_{\min}$ and $J_{\max}$, thus achieving a tighter bound.

B.3 Detailed Algorithms of CROP-LoRe

In the following, we will explain the *trajectory exploration and expansion*, the *growth of perturbation magnitude*, and *optimization tricks* in details.

Trajectory Exploration and Expansion. CROP-LoRe organizes all possible trajectories in the form of a search tree and progressively grows it. Each node of the tree represents a state, and the depth of the node is equal to the time step of the corresponding state in the trajectory. The root node (at depth 0) represents the initial state s_0. For each node, leveraging Theorem 4, we compute a non-decreasing sequence $\{r^k(s)\}_{k=0}^{|\mathcal{A}|-1}$ corresponding to required perturbation radii for π to choose each alternative action (the subscript t is omitted for brevity). Suppose the current ε satisfies $r^i(s) \leq \varepsilon < r^{i+1}(s)$. We grow $(i+1)$ branches from current state s corresponding to the original action and i alternative actions since $\varepsilon \geq r^j(s)$ for $1 \leq j \leq i$. For nodes on the newly expanded branch, we repeat the same procedure to expand the tree with depth-first search (Tarjan, 1972) until the terminal state of the game is reached or the node depth reaches H. As we expand, we keep the record of cumulative reward for each trajectory and update the lower bound $\underline{J}$ when reaching the end of the trajectory if necessary.

Perturbation Magnitude Growth. When all trajectories for perturbation magnitude ε are explored, we need to increase ε to seek for certification under larger perturbations. Luckily, since the action space is discrete, we do not need to examine every $\varepsilon \in \mathbb{R}^+$ (which is infeasible) but only need to examine the next ε where the chosen action in some step may change. We leverage *priority queue* (van Emde Boas, 1977) to effectively find out such next "critical" ε. Concretely, along the trajectory exploration, at each tree node, we search for the possible actions and store actions corresponding to $\{r^k(s)\}_{k=i+1}^{|\mathcal{A}|-1}$ into the priority queue, since these actions are exactly those need to be explored when ε grows. After all trajectories for ε are fully explored, we pop out the head element from the priority queue as the next node to expand and the next perturbation magnitude ε to grow. We repeat

Algorithm 3: CROP-LoRe: Adaptive search for certifying cumulative reward

Input: Enivronment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, R, \Gamma, d_0)$, trained value network Q^π with range $[V_{\min}, V_{\max}]$; parameters for randomized smoothing: sampling times m, smoothing variance σ^2, one-sided confidence parameter α

Output: a map M that maps an attack magnitude ε to the corresponding certified lower bound of reward J

▷ Initialize global variables

$p_que \leftarrow \emptyset$ ▷ initialize an empty priority queue containing tuples of (state s, action a, radius r, reward J), sorted by increasing r

$M \leftarrow \emptyset$

$J_{\text{global}} \leftarrow \infty$ ▷ initialize global minimum reward

$\Delta = (V_{\max} - V_{\min}) \sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}}$ ▷ confidence bound

Function GetActions $(s, \varepsilon_{\text{lim}}, J_{\text{cur}})$:
  Generate noise samples $\delta_i \sim \mathcal{N}(0, \sigma^2 I)$ for $1 \le i \le m$
  for *each action* $a \in \mathcal{A}$ **do**
    $\widetilde{Q}^\pi(s,a) \leftarrow \frac{1}{m} \sum_{i=1}^{m} \text{clip}(Q^\pi(s + \delta_i, a), \min = V_{\min}, \max = V_{\max})$
  $a^\star \leftarrow \text{argmax}_{a \in \mathcal{A}} \widetilde{Q}^\pi(s,a)$
  $a_list \leftarrow \emptyset$
  for *each action* $a \in \mathcal{A}$ **do**
    if $\Gamma(s,a) = \perp$ **then**
      continue
    if $\widetilde{Q}^\pi(s,a^\star) \ge \widetilde{Q}^\pi(s,a) + 2\Delta$ **then**
      $r \leftarrow \frac{\sigma}{2}(\Phi^{-1}(\frac{\widetilde{Q}^\pi(s,a^\star) - \Delta - V_{\min}}{V_{\max} - V_{\min}}) - \Phi^{-1}(\frac{\widetilde{Q}^\pi(s,a) + \Delta - V_{\min}}{V_{\max} - V_{\min}}))$
    else
      $r \leftarrow 0$
    if $r \le \varepsilon_{\text{lim}}$ **then**
      ▷ take possible actions
      $a_list \leftarrow a_list \cup \{a\}$
    else
      ▷ store impossible actions in queue for later expansion
      $p_que.\text{push}((s, a, r, J_{\text{cur}}))$
  return a_list

Procedure Expand $(s, \varepsilon_{\text{lim}}, J_{\text{cur}})$:
  if $J_{\text{cur}} \ge J_{\text{global}}$ **then**
    return 0 ▷ pruning
  $a_list \leftarrow$ GetActions$(s, \varepsilon_{\text{lim}}, J_{\text{cur}})$
  if $a_list = \emptyset$ **then**
    $J_{\text{global}} \leftarrow \min(J_{\text{global}}, J_{\text{cur}})$
    return 0
  for $a \in a_list$ **do**
    $s' \leftarrow \Gamma(s,a)$
    $ret \leftarrow$ Expand$(s', \varepsilon_{\text{lim}}, J_{\text{cur}} + R(s,a))$

$s_0 \sim d_0$ ▷ initialize initial state

Expand $(s_0, \varepsilon_{\text{lim}} = 0, J_{\text{cur}} = 0)$ ▷ expand initial trajectory

while *True* **do**
  if $p_que = \emptyset$ **then**
    break
  ▷ pop out the first element
  $(s, a, r, J) \leftarrow p_que.\text{pop}()$
  ▷ examine the next first element
  $(_, _, r', _) \leftarrow p_que.\text{top}()$
  ▷ derive the critical ε's
  $\varepsilon \leftarrow r, \varepsilon' \leftarrow r'$
  ▷ obtain a pair of mapping
  $M[\varepsilon] \leftarrow J_{\text{global}}$
  ▷ expand the tree from the new node
  Expand $(\Gamma(s,a), \varepsilon', J + R(s,a))$

this process until the priority queue becomes empty or the perturbation magnitude ε reaches the predefined threshold.

Additional Optimization. We adopt some additional optimization tricks to reduce the complexity of the algorithm. First, for games with no negative reward, we perform pruning to limit the tree size—if the cumulative reward leading to the current node already reaches the recorded lower bound, we can perform pruning, since the tree that follows will not serve to update the lower bound. This largely reduces the potential search space. We additionally adopt the memorization (Michie, 1968) technique which is commonly applied in search algorithms. We point out a few more potential improvements. *First*, with more specific knowledge of the game mechanisms, the search algorithm can be further optimized. Take the Pong game as an example, given the horizontal speed of the ball, we can compress the time steps where the ball is flying between the two paddles, thus reducing the computation. *Second*, empirical attacks may be efficiently incorporated into the algorithm framework to provide upper bounds that help with pruning.

Detailed Modules in Algorithm 3. The function GetAction computes the possible actions at a given state s under the limit ε, while the procedure Expand accomplishes the task of expanding upon a given node/state. The main part of the algorithm involves a loop that repeatedly selects the next element from the priority queue, *i.e.*, a node associated with an ε value, to expand upon.

Time Complexity. The time complexity of CROP-LoRe is $O(H|S_{\text{explored}}| \times (\log |S_{\text{explored}}| + |A|T))$, where $|S_{\text{explored}}|$ is the number of explored states throughout the search procedure, which is no larger than cardinality of state set, H is the horizon length, $|A|$ is the cardinality of action set, and T is the time complexity of performing local smoothing. The main bottleneck of the algorithm is the

large number of possible states, which is in the worst case exponential to state dimension. However, to provide a sound worst-case certification agnostic to game properties, exploring all possible states may be inevitable.

Detailed Inference Procedure and Additional Clarifications. In CROP-LoRe, the inference procedure per step is exactly the same as in CROP-LoAct. We obtain the certification using the adaptive search algorithm and Theorem 4. We clarify that the algorithm only requires access to the environment such that it can obtain the reward and next state via *interacting* with the environment $\mathcal{E}$ (*i.e.*, taking action a at state s and obtain next action s' and reward r), but does not require access to an oracle transition function Γ. We further emphasize that access to the environment that supports back-tracking is already sufficient for conducting our adaptive search algorithm.

C Discussion on the Certification Methods

We discuss the *advantages* and *limitations* of our certification methods, as well as possible direct *extensions*, hoping to pave the way for future research along similar directions. We also provide more *detailed analysis* that help with the understanding of our algorithms.

CROP-LoAct. The algorithm provides state-wise robustness certification in terms of the stability/consistency of the per-state action. It treats each time step *independently*, smooths the given state at the given time step, and provides the corresponding certification. Thus, one potential *extension* is to expand the time window from one time step to a consecutive sequence of several time steps, and provide certification for the sequences of actions for the given window of states.

CROP-GRe. As explained in Section 5, the expectation bound $\underline{J_E}$ is too loose to have any practical usage. In comparison, percentile bound $\underline{J_p}$ is much tighter and practical. However, one limitation of $\underline{J_p}$ is that there exists an upper bound of the attack magnitude ε that can be certified for each σ, as explained in Appendix B.2. For attack magnitudes that exceed the upper bound, we can obtain no useful information via this certification (though the upper bound is usually sufficiently large).

One drawback of CROP-GRe. Lemma 2 requires knowing the noise added to each step beforehand so as to generate m randomized trajectories given the known noise sequence $\{\zeta_t\}_{t=0}^{H-1}$. Thus, it cannot be applied against an adaptive attacker. We note that our CROP-LoAct and CROP-LoRe does not have such limitation.

CROP-LoRe. The algorithm provides the absolute lower bound $\underline{J}$ of the cumulative reward for any finite-horizon trajectory with a given initial state. The advantage of the algorithm is that $\underline{J}$ is an absolute lower bound that bounds the worst-case situation (apart from the exceptions due to the probabilistic confidence α), rather than the statistical lower bounds $\underline{J_E}$ and $\underline{J_p}$ that characterize the statistical properties of the random variable J_ε.

One potential pitfall in understanding $\underline{J}$. What CROP-LoRe certifies is the lower bound for the trajectory with *a given initial state* $s_0 \sim d_0$, rather than all possible states in d_0. This is because our search starts from a root node of the tree, which is set to the fixed given state s_0.

Confidence of CROP-LoRe. Regarding the *confidence* of our probabilistic certification, we clarify that we consider the independent multiple-test. Concretely, due to the independence of decision making errors, the confidence is $(1-\alpha)^N$, where N is the maximum number of possible attacked states explored by CROP-LoRe. Formally,

$$\begin{aligned}
\Pr[\text{CROP-LoRe certification holds}] &= \prod_{\substack{\text{all } s \text{ explored} \\ \text{by CROP-LoRe}}} \Pr[\text{not make error on } s | \text{not make error on } s_{pre}] \\
&\overset{\text{(i)}}{=} \prod_{\text{all attackable } s} (1-\alpha) \times \prod_{\text{all unattackable } s} 1 \\
&\overset{\text{(ii)}}{=} (1-\alpha)^N. \qquad (13)
\end{aligned}$$

In the above equation, we can see that (i) leverages the independence which in turn gives a bound of form $(1-\alpha)^N$. This is because in each step, the event of "certification does not hold" is *independent* since we sample Gaussian noise *independently*. Leveraging such independence, we obtain a confidence lower bound of $(1-\alpha)^N$. From (ii), we see that the confidence is only related to

the number of possible attacked states N. This is because for unattackable states, the certification deterministically holds since there is no attack at current step. Therefore, we only need to count the confidence intervals from all possibly attackable steps instead of all states explored. We remark that in practice, the attacker usually has the ability to perturb only a limited number of steps, *i.e.*, N is typically small. Therefore, the confidence is non-trivial.

Main limitation of CROP-LORE. The main *limitation* is the high time complexity of the algorithm (Details see Appendix B.3). The algorithm has exponential time complexity in worst case despite the existence of several optimizations. Therefore, it is not suitable for games with a large action set, or when the horizon length is set too long.

Extension to Policy-based Methods. Though we specifically study the robustness certification in Q-learning in this paper, our two certification criteria and three certification strategies can be readily extended to policy-based methods. The intuition is that, instead of smoothing the value function in the Q-learning setting, we directly smooth the policy function in policy-based methods. With the smoothing module replaced and the theorems updated, other technical details in the algorithms for certifying the per-state action and the cumulative reward would then be similar.

D ADDITIONAL EXPERIMENTAL DETAILS

D.1 DETAILS OF THE GAME ENVIRONMENT

We experiment with two Atari-2600 games in OpenAI Gym (Brockman et al., 2016) on top of the Arcade Learning Environment (Bellemare et al., 2013). The *states* in the games are high dimensional color images ($210 \times 160 \times 3$) and the *actions* are discrete actions that control the agent to accomplish certain tasks. Concretely, we use the `NoFrameskip-v4` version for our experiments, where the randomness that influences the environment dynamics can be fully controlled by setting the random seed of the environment at the beginning of an episode.

D.2 DETAILS OF THE RL METHODS

We introduce the details of the six RL methods we evaluated. Specifically, for each method, we first discuss the algorithm, and then present the implementation details for it.

StdTrain (Mnih et al., 2013): StdTrain (naturally trained DQN model) is an algorithm based on Q-learning, in which DQN is used to reprent Q-value function and TD-loss is used to optimize DQN. Our StdTrain model is implemented with Double DQN (Van Hasselt et al., 2016) and Prioritized Experience Replay (Schaul et al., 2015).

GaussAug (Behzadan & Munir, 2017b): The algorithm of GaussAug is similar with AtdTrain, in which appropriate Gaussian random noises added to states during training. As for implementation, our GaussAug model is based on the same architecture with the same set of *basic* training techniques of StdTrain, and adds Gaussian random noise with $\sigma = 0.005$, which has the best testing performance under attacks compared with other σ, to all the frames during training.

AdvTrain (Behzadan & Munir, 2017b): Instead of using the original observation to train Double DQN (Van Hasselt et al., 2016), AdvTrain generates adversarial perturbations and applies the perturbations to part of or all of the frames when training. In our case, we add the perturbations generated by 5-step PGD attack to 50% of the frames to make a balance between between stable training and effectiveness of adversarial training.

RegPGD and RegCVX (Zhang et al., 2020): Instead of utilizing adversarial noise to train models directly like AdvTrain, RegPGD and RegCVX regularize the loss function with the help of adversarial noise during training in order to train empirically robust reinforcement learning agents. In our experiment settings, which is the same as RegPGD and RegCVX (Zhang et al., 2020), models consider adversarial noise with ℓ_∞-norm $\varepsilon = 1/255$ when solving the maximization for the regularizer and minimize the original TD-loss concurrently.

RadialRL (Oikarinen et al., 2020): With the similar idea of adding regularization during training of RegPGD and RegCVX, RadialRL compute the worst-case loss instead of regularizing of the bound of the difference of policy distribution under original states and adversarial states. In our case, the

RadialRL model uses a linearly increasing perturbation magnitude ε (from 0 to $1/255$) to compute the adversarial loss.

CARRL (Everett et al., 2021): Unlike other methods which try to improve robustness of RL-agents during training, CARRL enhances RL-agents' robustness during testing time. It computes the lower bound of each action's Q value at each step and take the one with the highest lower bound conservatively. Our CARRL model evaluated in the environment of CartPole uses $\varepsilon = 0.1$ when computing the lower bound of Q value, which has the best empirical performance under noises and attacks.

NoisyNet (Fortunato et al., 2017): Instead of the conventional exploration heuristics for DQN and Dueling agents which is ε-greedy algorithms, NoisyNet adds parametric noise to DQN's weights, which is claimed to aid efficient exploration and yield higher scores than StdTrain. Our NoisyNet with the same basic architecture of StdTrain model adds Gaussian random noise with $\sigma = 0.5$ to its weights and samples during training and testing time.

GradDQN (Pattanaik et al., 2018): GradDQN is a variation of AdvTrain, which utilizes conditional value of risk (CVaR) as optimization criteria for robust control instead standard expected long term return. This method maximizes the expected return over worst α percentile of returns, which can prevent the adversary from possible bad states. In our case, we generate the adversarial states with 10-step attack by calculating the direction of gradient and sampling in that direction from $beta1, 1$ distribution in each step.

For StdTrain, RegPGD, RegCVX, RadialRL, we directly evaluate the trained models provided by the authors[1].

D.3 Rationales for Settings and Experimental Designs

Finite Horizon Setting. In this paper, we focus on a finite horizon setting, mainly for the evaluation purpose—we would need to compute the (certified and empirical) cumulative rewards from finite rollouts. We clarify that our criteria, theorem, and algorithm are applicable to the infinite horizon case as well.

The Role of Practical Attacks in Our Evaluation. The goal of this paper is to provide certification that comes with theoretical guarantees, and therefore following the standard certified robustness literatures (Cohen et al., 2019; Li et al., 2020; Salman et al., 2019; Jeong & Shin, 2020) there is no need to evaluate the empirical attacks, since the certification always holds as long as the attacks satisfy certain conditions regardless of the actual attack algorithms. The attacks we evaluated in our paper were only for the demonstration purpose and in the hope of providing an example.

D.4 Detailed Evaluation Setup

Evaluation Setup for CROP-LoAct. We report results averaged over 10 episodes and set the length of the horizon $H = 500$. At each time step, we sample $m = 10,000$ noisy states for smoothing. When applying Hoeffding's inequality, we adopt the confidence level parameter $\alpha = 0.05$. Since the input state observations for the two Atari games are in image space, we rescale the input states such that each pixel falls into the range $[0, 1]$. When adding Gaussian noise to the rescaled states, we sample Gaussian noise of zero mean and different levels of variances. Concretely, the standard deviation σ is selected among $\{0.001, 0.005, 0.01, 0.03, 0.05, 0.1, 0.5, 0.75, 1.0, 1.5, 2.0, 4.0\}$. We evaluate different parameters for different games.

Evaluation Setup for CROP-GRe. We sample $m = 10,000$ σ-randomized trajectories, each of which has length $H = 500$, and conduct experiments with the same set of smoothing parameters as in the setup of CROP-LoAct. When accumulating the reward, we set the discount factor $\gamma = 1.0$ with no discounting. We take $\alpha = 0.05$ as the confidence level when applying Hoeffding's inequality in the expectation bound $\underline{J_E}$, and $\alpha = 0.05$ as the confidence level for ComputeOrderStats in the percentile bound $\underline{J_p}$. For the validation of tightness, concretely, we carry out a 10-step PGD attack with ℓ_2 radius within the perturbation bound at all time steps during testing to evaluate the tightness of our certification.

[1] StdTrain, RegPGD, RegCVX from `https://github.com/chenhongge/SA_DQN` and RadialRL from `https://github.com/tuomaso/radial_rl` under Apache-2.0 License

Evalution Setup for CROP-LoRe. We set the horizon length as $H = 200$ and sample $m = 10,000$ noisy states with the same set of smoothing variance as in the setup of CROP-LoAct. Similarly, we adopt $\gamma = 1.0$ as the discount factor and $\alpha = 0.05$ as the confidence level when applying Hoeffding's inequality. When evaluating Freeway, we modify the reward mechanism so that losing one ball incurs a zero score rather than a negative score. This modification enables the pruning operation and therefore facilitates the search algorithm, yet still respects the goal of the game. We also empirically attack the policy $\tilde{\pi}$ to validate the tightness of the certification via the same PGD attack as described above. For each set of experiments, we run the attack one time with the same initial state as used for computing the lower bound in Algorithm 3.

D.5 Experimental Setup for CartPole

We additionally experiment with CartPole-v0 in OpenAI Gym (Brockman et al., 2016) on top of the Arcade Learning Environment (Bellemare et al., 2013). We introduce the experimental setup below.

Details of the CartPole Game Environment. The *state* in the CartPole game is a low dimensional vector of length 4 and the *action* $\in \{$*move right*, *move left*$\}$. The goal of the game is to balance the rod on the cart, and one reward point is earned at each timestep when the rod is upright.

Implementation Details of the RL Methods on CartPole. In addition to the previous 8 RL Methods in Pong and Freeway, we evaluate another RL algorithm CARRL (Everett et al., 2021), which is claimed to be robust in low dimensional games like CartPole. This method relies on linear bounds output by the Q network and selects the action whose lower bound of the Q value is the highest.

Evaluation Setup for CROP-LoAct on CartPole. We report results averaged over 10 episodes and set the length of the horizon $H = 200$. At each time step, we sample $m = 10,000$ noisy states for smoothing. When applying Hoeffding's inequality, we adopt the confidence level parameter $\alpha = 0.05$. We do not performing rescaling on the state observations. When adding Gaussian noise to states,we sample Gaussian noise of zero mean and different levels of variances. Concretely, the standard deviation σ is selected among $\{0.001, 0.005, 0.01, 0.03, 0.05, 0.1\}$.

Evaluation Setup for CROP-GRe on CartPole. We sample $m = 10,000$ σ-randomized trajectories, each of which has length $H = 200$, and conduct experiments with the same set of smoothing parameters as in the setup of CROP-LoAct. All other experiment settings are the same as Pong and Freeway in Appendix D.4.

Evalution Setup for CROP-LoRe on CartPole. Since the reward of CartPole game is quite dense, we set the horizon length as $H = 10$ and sample $m = 10,000$ noisy states with the same set of smoothing variance as in the setup of CROP-LoAct. All other experiment settings are the same as Pong and Freeway in Appendix D.4.

E Additional Evaluation Results and Discussions

E.1 Robustness Certification for Per-state Action – Certified Ratio

We present the robustness certification for per-state action in terms of the certified ratio η_r w.r.t. the certified radius r in Figure 4. From the figure, we see that RadialRL is the most certifiably robust method on Freeway, followed by RegCVX and RegPGD; while on Pong, RegCVX is the most robust. These conclusions are highly consistent with those in Section 6.1 as observed from Figure 1. Comparing the curves for Freeway and Pong, we note that Freeway not only achieves larger certified radius *overall*, it also more *frequently* attains large certified radius.

E.2 Results of Freeway under Larger Smoothing Parameter σ

Since the robustness of the methods RadialRL, RegCVX, and RegPGD on Freeway have been shown to improve with the increase of the smoothing parameter σ in Section 6, we subsequently evaluate their performance under even larger σ values. We present the certification results in Figure 5 and their benign performance in Figure 6.

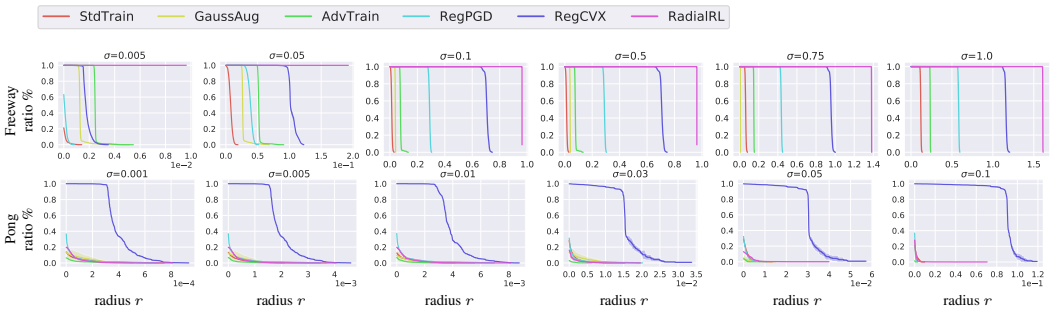
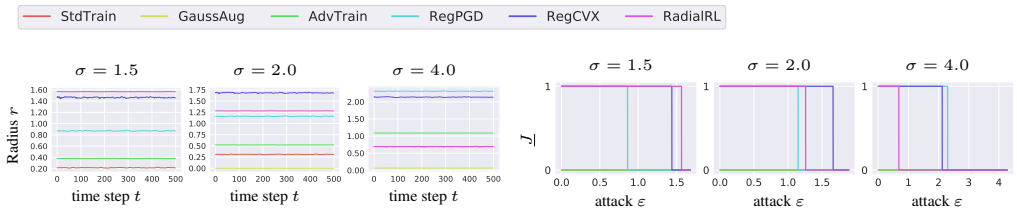


Figure 4: Robustness certification for *per-state action* in terms of certified ratio η_r w.r.t. certified radius r. Each column corresponds to one smoothing parameter σ. The shaded area represents the standard deviation.



(a) Certified radius R_t along time steps

(b) Absolute lower bound $\underline{J}$ w.r.t. attack ε

Figure 5: Robustness certification for (a) *per-state action* in terms of certified radius r at all time steps, and (b) *cumulative reward* in terms of absolute lower bound $\underline{J}$ w.r.t. the attack magnitude ε. The results are reported under large smoothing parameter σ compared with those evaluated in Figure 1 and Figure 3 in the main paper.

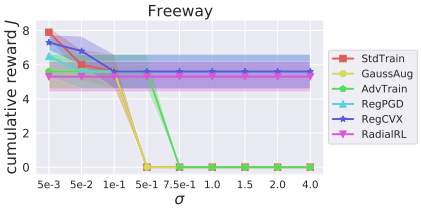


Figure 6: Benign performance of locally smoothed policy $\tilde{\pi}$ under a larger range of smoothing parameter σ with clean state observations.

First of all, as Figure 6 reveals, RadialRL, RegCVX, and RegPGD can tolerate quite large noise—the benign performance of these methods does not drop even for σ up to 4.0. For the remaining three methods, the magnitude of noise they can tolerate is much smaller, as already presented in Figure 2 in the main paper. On the other hand, although all the three methods RadialRL, RegCVX, and RegPGD invariably attain good benign performance as σ grows, their certified robustness does not equally increase. As Figure 5 shows, the certified robustness of RegPGD increases the fastest, while that of RadialRL drops a little. The observation corresponds exactly to our discussion regarding the tradeoff between value function smoothness and the margin between the values to the top two actions in Section 4.

E.3 Periodic Patterns for Per-State Robustness

We specifically study the periodic patterns for per-state robustness in Pong and present the results in Figure 7. In the game frames, our agent controls the green paddle on the right. Whenever the opponent (the orange paddle on the left) misses catching a ball, we earn a point. We present two periods (frame 320-400 and frame 400-480) where our agent each earns a point in the figure above. The total points increase from 2 to 3, and from 3 to 4, respectively.

We note that Pong is a highly periodic game, with each period consisting of the following four game stages corresponding to distinct stages of the *certified robustness*: 1) A ball is initially fired and set flying towards the opponent. In this stage, the certified radius remains low, since our agent does not need to take any critical action. 2) The opponent catches the ball and reverses the direction of the ball. In this stage, the certified radius gradually increases, since the moves of our agent serve as the preparation to catch the ball. 3) The ball bounces from the ground, flies towards our agent, gets caught, and bounces back. In this stage, the certified radius increases rapidly, since every move is critical in determining whether we will be able to catch the ball, *i.e.*, whether our point (action value) will increase. 4) The opponent misses the ball, and the ball disappears from the game view. In this

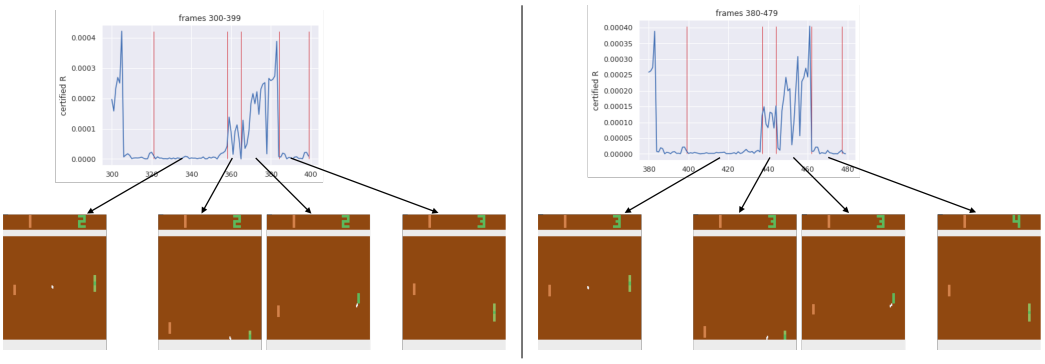


Figure 7: Periodic patterns in Pong. The figure shows two periods (left and right), including the certified radius r w.r.t. the time steps (above), and the selected game frames corresponding to different stages in each period (below). Different periods are highly similar.

stage, the certified radius drastically plummets to a low value, since there is no clear signal (*i.e.*, ball) in the game for the agent to take any determined action.

The illustration not only helps us understand the semantic meaning of the certified radius, but can further provide guidance on designing empirically robust RL algorithms that leverage the information regarding the high/low certified radius of the states.

E.4 Detailed Discussions and Selection of Smoothing Parameter σ

We first take Freeway as an example to concretely explain the impact of smoothing variance from the perspective of benign performance and certified results, and then introduce the guidelines for the selection of σ based on different certification criteria.

On *Freeway*, as σ increases, the *benign performance* of the locally smoothed policy $\tilde{\pi}$ of StdTrain, GaussAug, and AdvTrain decrease rapidly in Figure 2. Also, the *certified radius* of these methods barely gets improved and remains low even as σ increases. For the percentile bound $\underline{J_p}$, these three methods similarly suffer a drastic decrease of *cumulative reward* as σ grows. This is however not the case for RegPGD, RegCVX, and RadialRL, for which not only the *benign performance* do not experience much degradation for σ as large as 1.0, but the *certified radius* even steadily increases. In addition, in terms of the percentile bound $\underline{J_p}$, as σ grows, even we increase the attack ε simultaneously, $\underline{J_p}$ almost does not decrease. This indicates that larger smoothing variance can bring more robustness to RegPGD, RegCVX, and RadialRL.

CROP-LoAct. We discuss how to find the sweet spot that enables high benign performance and robustness simultaneously. As σ increases, the benign performance of different methods generally decreases due to the noise added, while not all methods decrease at the same rate. On *Freeway*, StdTrain and GaussAug decrease the fastest, while RegPGD, RegCVX, and RadialRL do not degrade much even for σ as large as 4.0. Referring back to Figure 1 and Figure 5, we see that the certified radius of StdTrain and GaussAug remains low even though σ increases, thus the best parameters for these two methods are both around 0.1.

For RegPGD and RegCVX, their certified radius instead steadily increases as σ increases to 4.0, indicating that larger smoothing variance can bring more robustness to these two methods without sacrificing benign performance, offering certified radius larger than 2.0. As to RadialRL, its certified radius reaches the peak at $\sigma = 1.0$ and then decreases under larger σ. Thus, 1.0 is the best smoothing parameter for RadialRL which offers a certified radius of 1.6. Generally speaking, it is feasible to select an appropriate smoothing parameter σ for all methods to increase their robustness, and robust methods will benefit more from this by admitting larger smoothing variances and achieving larger certified radius.

CROP-GRe. We next discuss how to leverage Figure 3 to assist with the selection of σ. For a given ε or a known range of ε, we can compare between the lower bounds corresponding to different σ and select the one that gives the highest lower bound. For example, if we know that ε is small in advance, we can go for a relatively small σ which retains a quite high lower bound. If we know that ε will be large, we will instead choose among the larger σ's, since a larger ε will not fall in the

certifiable ranges of smaller σ's. According to this guideline, we are able to obtain lower bounds of quite high value for RadialRL, RegCVX, and RegPGD on Freeway under a larger range of attack ε.

CROP-LoRe. Apart from the conclusion that larger smoothing parameter σ often secures higher lower bound $\underline{J}$, we point out that smaller smoothing variances may be able to lead to a higher lower bound than larger smoothing variances for a certain range of ε. This is almost always true for very small ε since smaller σ is sufficient to deal with the weak attack without sacrificing much empirical performance, *e.g.*, in Figure 3, when $\varepsilon = 0.001$, RegCVX on Freeway can achieve $\underline{J} = 3$ at $\sigma = 0.005$ while only $\underline{J} = 1$ for large σ. This can also happen to robust methods at large ε, *e.g.*, when $\varepsilon = 1.2$, RadialRL on Freeway achieves $\underline{J} = 2$ at $\sigma = 0.75$ while only $\underline{J} = 1$ at $\sigma = 1.0$. Another case is when σ is large enough such that further increasing σ will not bring additional robustness, *e.g.*, in Figure 5, when $\varepsilon = 1.5$, RadialRL on Freeway achieves $\underline{J} = 1$ at $\sigma = 1.5$ while only $\underline{J} = 0$ at $\sigma = 4.0$.

E.5 Computational Cost

Our experiments are conducted on GPU machines, including GeForce RTX 3090, GeForce RTX 2080 Ti, and GeForce RTX 1080 Ti. The running time of $m = 10000$ forward passes with sampled states for per-state smoothing ranges from 2 seconds to 9 seconds depending on the server load. Thus, for one experiment of CROP-LoAct with trajectory length $H = 500$ and 10 repeated runs, the running time ranges from 2.5 hours to 12.5 hours. CROP-GRe is a little more time-consuming than CROP-LoAct, but the running time is still in the same magnitude. For trajectory length $H = 500$ and sampling number $m = 10000$, most of our experiments finish within 3 hours. For CROP-LoRe specifically, the running time depends on the eventual size of the search tree, and therefore differs tremendously for different methods and different smoothing variances, ranging from a few hours to 4 to 5 days.

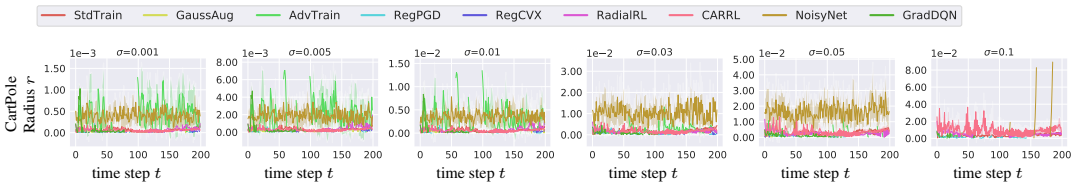
E.6 Discussion on Evaluation Results of CartPole

We provide the evaluation results for CROP on CartPole, comparing nine RL algorithms: the six algorithms evaluated in our main paper (StdTrain, GaussAug, AdvTrain, RegPGD, RegCVX, and RadialRL), as well as three additional algorithms (CARRL, NoisyNet, and GradDQN). The detailed descriptions of these algorithms are provided in Appendix D.2. We present the evaluation results in Figure 8.

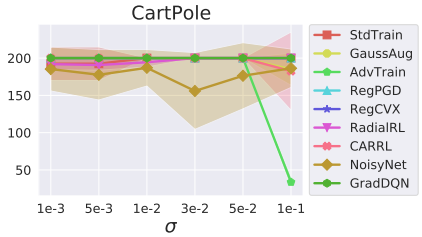
Impact of Smoothing Parameter σ. In CartPole game, we draw similar conclusions regarding the impact of smoothing variance from the results given by CROP-LoAct, CROP-GRe and CROP-LoRe. For small σ, different methods are indistinguishable. As σ increases, CARRL demonstrates its advantage as σ increases, but AdvTrain and GradDQN tolerate large noise. One of differences when evaluating with CROP-GRe and CROP-LoRe is that RadialRL can hold a high performance and even sometimes can be better than CARRL. Another one is that the lower bound for GradDQN is lower than all other methods in CROP-LoRe.

Tightness of the certification $\underline{J_E}$, $\underline{J_p}$ and $\underline{J}$. In CartPole, we also compare the empirical cumulative rewards achieved under PGD attacks with our certified lower bounds $\underline{J_E}$, $\underline{J_p}$ and $\underline{J}$. Demonstrated by Figure 8c, first of all, the correctness of our bounds is validated because the empirical results are consistently lower bounded by our certifications. Compared with the loose expectation bound $\underline{J_E}$, the improved percentile bound $\underline{J_p}$ is much tighter. Compared with these two methods, the absolute lower bound $\underline{J}$ is even tighter, especially noticing the zero gap between the certification and the empirical result when the attack magnitude is not too large. Finally, methods with the same empirical results under attack may achieve very different certified lower bounds $\underline{J_E}$, $\underline{J_p}$ and $\underline{J}$, showing the importance of certification.

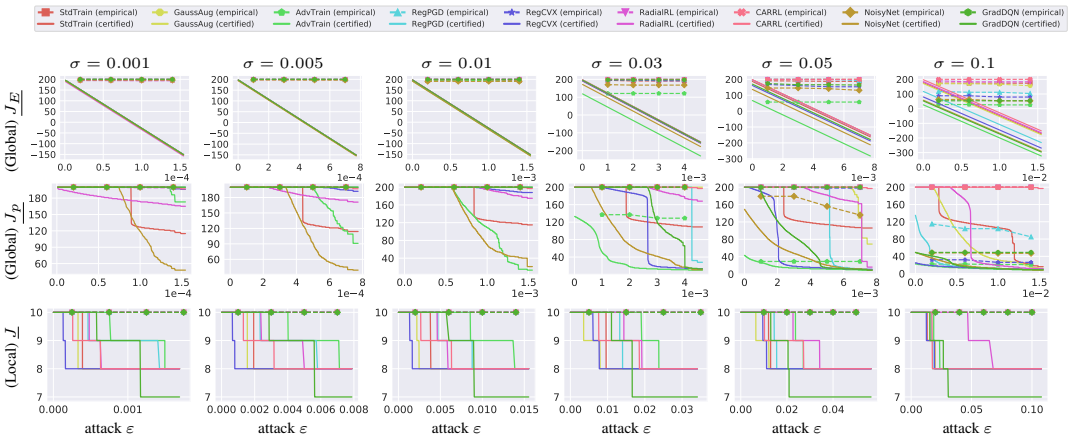
Game Properties. Different from Pong and Freeway, CartPole is a game with low dimension *states*, which leads to some different observations. First, StdTrain can tolerate noise well in CartPole. Second, CARRL demonstrates its advantage in low dimensional game, which is consistent with the empirical observations in Everett et al. (2021).



(a) Robustness certification for *per-state action* in terms of certified radius r at all time steps. The shaded area represents the standard deviation.



(b) Benign performance of locally smoothed policy $\tilde{\pi}$ under different smoothing variance σ.



(c) Robustness certification as cumulative reward, including *expectation bound* $\underline{J_E}$, *percentile bound* $\underline{J_p}$ ($p = 50\%$), and *absolute lower bound* $\underline{J}$. Solid lines represent the certified reward bounds of different methods, and dashed lines show the empirical performance under PGD attacks.

Figure 8: **Robustness certification on CartPole** in terms of (a-b): *robustness of per-state action* and (c): *lower bound of cumulative rewards*. In (a) and (c), each column corresponds to one smoothing variance.

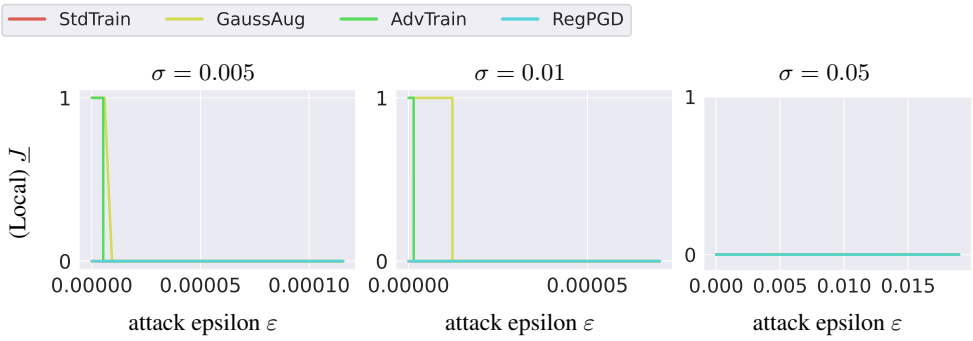


Figure 9: **Robustness certification for cumulative reward lower bound $\underline{J}$ on *Breakout***. We evaluate the trajectory length $H = 200$. Each column corresponds to one smoothing parameter σ. Different lines represent the certified reward bounds of different RL methods.

Table 1: **Empirical cumulative reward obtained under attacks of different magnitudes.** We follow Zhang et al. (2020) and Oikarinen et al. (2020) to conduct PGD attacks. We evaluate the trajectory length $H = 200$. Each column group corresponds to one smoothing parameter σ. For each smoothing parameter σ, we evaluate four attack magnitudes ε. Different rows show the achieved empirical cumulative reward of different methods.

σ	0.005				0.01				0.05			
ε	$5e-3$	$1e-2$	$5e-2$	$1e-1$	$5e-3$	$1e-2$	$5e-2$	$1e-1$	$5e-3$	$1e-2$	$5e-2$	$1e-1$
StdTrain	6	1	0	0	1	1	0	0	0	0	0	0
GaussAug	6	5	5	1	6	5	5	1	0	0	0	0
AdvTrain	6	6	0	0	7	7	0	0	0	0	0	0
RegPGD	2	0	0	0	0	0	0	0	0	0	0	0

E.7 More Results for CROP-LoRe on Atari Game Breakout

We provide the results for CROP-LoRe on a more complicated Atari game Breakout. Concretely, we present the *certified* cumulative reward lower bound computed via our CROP-LoRe algorithm in Figure 9, as well as the *empirical* cumulative reward obtained under practical attacks in Table 1.

We obtain similar conclusions based on the new game and will describe the detailed observations as below. **First**, Figure 9 shows that GaussAug is the most certifiably robust, followed by AdvTrain; while StdTrain and RegPGD are shown to be non-robust for this game, whose lower bounds remain 0 for all smoothing parameters σ we evaluate. **Second**, the robustness ranking given by our certification is aligned with the ranking under practical attacks. From Table 1, we see that GaussAug can achieve much higher cumulative reward under larger perturbation magnitudes ε compared with other RL methods, which is consistent with the conclusion drawn from Figure 9. **Third**, our certification is relatively tight, especially for larger perturbation magnitudes. Notably, when $\sigma = 0.05$, our lower bounds exactly match the attack results.

F A Broader Discussion on Related Work

F.1 Evasion Attacks in RL

We consider the adversarial attacks on state observations, where the attacker aims to add perturbations to the state during test time, so as to achieve certain **adversarial goals**, such as misleading the *action* selection and minimizing the *cumulative reward*. We discuss the related works that fall into these two categories below.

Misleading the Action Selection. It has been shown that different methods—applying *random noise* to simply interfere with the action selection (Kos & Song, 2017), or adopting *adversarial attacks* (*e.g.*, FGSM (Goodfellow et al., 2014) and CW attacks (Carlini & Wagner, 2017)) to deliberately alter the probability of action selection—are effective to different degrees. More concretely, when manipulating the action selection probability, some works aim to reduce the probability of selecting the optimal action (Huang et al., 2017; Kos & Song, 2017; Behzadan & Munir, 2017b), while others try to increase the probability of selecting the worst action (Lin et al., 2017; Pattanaik et al., 2018).

Minimizing the Cumulative Reward. ATLA (Zhang et al., 2021) and PA-AD (Sun et al., 2021) consider an *optimal adversary* under the SA-MDP framework (Zhang et al., 2020), which aims to lead to minimal value functions under bounded state perturbations. To find this *optimal adversary* (*i.e.*, the optimal adversarial state perturbation), ATLA (Zhang et al., 2021) proposes to train an adversary whose action space is the perturbation set in the state space, while PA-AD (Sun et al., 2021) further decouples the problem of finding state perturbations into finding policy perturbations plus finding the state that achieves the lowest value policy, thus addressing the challenge of large state space.

F.2 Robust RL

Distributionally Robust RL. Nilim & El Ghaoui (2005) and Iyengar (2005) consider the problem of distributionally robust RL, where there is normally an uncertainty set with prior information

regarding the distribution of the uncertain parameters. Iyengar (2005) directly put constraints on environmental dynamics and reward functions by assuming that they take values in an uncertainty set. Another line of research assumes the environmental dynamics and reward function as the uncertain parameters and are sampled from an uncertain distribution in the uncertain set. The uncertainty set is in general state-wise independent, *i.e.*, "s-rectangularity" in Nilim & El Ghaoui (2005). Both types aim to derive a policy by solving a max-min problem, with the former taking the minimum directly over the parameters, while the latter taking the minimum over the distribution.

Empirically Robust RL against Evasion Attacks. We have briefly introduced related work on empirically robust RL in Section 7 in the main paper. We emphasize two main differences between the contributions of these works and our CROP. **First**, most of these robust RL methods only provide empirical robustness against perturbed state inputs during test time, but cannot provide theoretical guarantees for the performance of the trained models under any bounded perturbations; while our CROP framework can provide practically computable *certified robustness* w.r.t. two robustness certification criteria (per-state action stability and cumulative reward lower bound), *i.e.*, for a given RL algorithm, we can compute certifications for it that indicate its robustness. (We will discuss a few more related work that can provide certified robustness in Appendix F.3.) **Second**, most of these Robust RL methods focus on only the per-state decision, while we additionally consider the *certification of trajectories* to obtain the lower bound of cumulative rewards. Below, we provide a more comprehensive review of the categories of RL methods that demonstrate empirical robustness against evasion attacks.

Randomization methods (Tobin et al., 2017; Akkaya et al., 2019) were first proposed to encourage exploration. This type of method was later systematically studied for its potential to improve model robustness. NoisyNet (Fortunato et al., 2017) adds parametric noise to the network's weight during training, providing better resilience to both training-time and test-time attacks (Behzadan & Munir, 2017b; 2018), also reducing the transferability of adversarial examples, and enabling quicker recovery with fewer number of transitions during phase transition.

Under the *adversarial training* framework, Kos & Song (2017) and Behzadan & Munir (2017b) show that re-training with random noise and FGSM perturbations increases the resilience against adversarial examples. Pattanaik et al. (2018) leverage attacks using an engineered loss function specifically designed for RL to significant increase the robustness to parameter variations. RS-DQN (Fischer et al., 2019) is an *imitation learning* based approach that trains a robust student-DQN in parallel with a standard DQN in order to incorporate the constrains such as SOTA adversarial defenses (Madry et al., 2017; Mirman et al., 2018).

SA-DQN (Zhang et al., 2020) is a *regularization* based method that adds regularizers to the training loss function to encourage the top-1 action to stay unchanged under perturbation.

Built on top of the *neural network verification* algorithms (Gowal et al., 2018; Weng et al., 2018), Radial-RL (Oikarinen et al., 2020) proposes to minimize an adversarial loss function that incorporates the upper bound of the perturbed loss, computed using certified bounds from verification algorithms. CARRL (Everett et al., 2021) aims to compute the lower bounds of action-values under potential perturbation and select actions according to the worst-case bound, but it relies on linear bounds (Weng et al., 2018) and is only suitable for low-dimensional environments.

F.3 Robustness Certification for RL

Despite the abundant literature in robustness certification in supervised learning, there is almost no work on robustness certification for RL, given the unclear criteria and the intrinsic difficulty of the task. In this part, we first introduce another two works that can achieve robustness certification at state-level, and then another concurrent work Kumar et al. (2021) which also aims to provide provable robustness regarding the cumulative reward for RL.

Robustness Certification on State Level. Fischer et al. (2019) and Zhang et al. (2020) have discussed the state level robustness certification as well, but the way they obtain the certification is different from our CROP. Concretely, we achieve robustness by probabilistic certification via randomized smoothing (Cohen et al., 2019), while Fischer et al. (2019) and Zhang et al. (2020) directly achieve robustness by deterministic certification via leveraging the neural network verification techniques (Zhang et al., 2018; Xu et al., 2020; Mirman et al., 2018).

Despite these works on state-level robustness certification for RL, we are the *first* to provide the certification of cumulative rewards. We emphasize that the certification of lower bound of the cumulative reward in RL is more challenging than the per-state certification considering the dynamic nature of RL. Our main contribution of the paper indeed mainly focuses on providing an efficient tree search based algorithm CROP-LoRe to certify the lower bound of the cumulative reward together with rigorous analysis of the certification.

Robustness Certification for Cumulative Rewards. We compare our robustness certification for cumulative rewards with another concurrent work Kumar et al. (2021).

In terms of the *threat model*, both Kumar et al. (2021) and CROP consider the type of adversary that can perturb the state observations of the agent during test time. In our work, we consider a perturbation budget per time step following previous works (Huang et al., 2017; Behzadan & Munir, 2017b; Kos & Song, 2017; Pattanaik et al., 2018; Zhang et al., 2020), while they assume a budget for the entire episode. The two perspectives are closely related.

Regarding the *certification criteria*, we certify both *per-state action stability* and cumulative reward lower bound. Although they also consider the cumulative reward in their certification goal, they formulate the certification as a classification problem—certifying whether the cumulative reward is above a threshold or not, in contrast to directly certifying the lower bound as in our case.

For the *certification technique*, both works are developed based on randomized smoothing proposed in supervised learning (Cohen et al., 2019). We propose a global smoothing technique (CROP-GRe), as well as an adaptive search algorithm coupled with local smoothing (CROP-LoRe) to achieve the certification of cumulative reward; while they propose an adaptive version of the Neyman-Pearson lemma, which is similar with our global smoothing. One distinction between our CROP-GRe and Kumar et al. (2020) is that they can defend against an adaptive adversary while our CROP-GRe cannot (as concretely stated in Appendix C). However, we would like to further emphasize that our adaptive search algorithm CROP-LoRe does not have such limitation regarding the adaptive attacker, and we show that CROP-LoRe is much more sophisticated and tight than the global smoothing one, which is an important contribution in our work.