Breaking the Barrier: Enhanced Utility and Robustness in Smoothed DRL Agents

Chung-En Sun [1] **Sicun Gao** [1] **Tsui-Wei Weng** [1]

Abstract

Robustness remains a paramount concern in deep reinforcement learning (DRL), with randomized smoothing emerging as a key technique for enhancing this attribute. However, a notable gap exists in the performance of current smoothed DRL agents, often characterized by significantly low clean rewards and weak robustness. In response to this challenge, our study introduces innovative algorithms aimed at training effective smoothed robust DRL agents. We propose S-DQN and S-PPO, novel approaches that demonstrate remarkable improvements in clean rewards, empirical robustness, and robustness guarantee across standard RL benchmarks. Notably, our S-DQN and S-PPO agents not only significantly outperform existing smoothed agents by an average factor of $2.16\times$ under the strongest attack, but also surpass previous robustly-trained agents by an average factor of $2.13\times$. This represents a significant leap forward in the field. Furthermore, we introduce Smoothed Attack, which is $1.89\times$ more effective in decreasing the rewards of smoothed agents than existing adversarial attacks. Our code is available at: https://github.com/Trustworthy-ML-Lab/Robust_HighUtil_Smoothed_DRL

1. Introduction

Deep Reinforcement Learning (DRL) has achieved remarkable performance, surpassing human-level capabilities in various game environments (Mnih et al., 2013; Silver et al., 2016). However, recent studies have unveiled a significant vulnerability within DRL – its susceptibility to adversarial perturbations (Huang et al., 2017; Lin et al., 2017; Weng et al., 2020). As a result, it is imperative to enhance the robustness of DRL agents before deploying them in real-world applications, especially those involving safety-critical tasks.

In response to this need, researchers have adapted techniques from robust classifier training to bolster DRL agents' resilience (Pattanaik et al., 2018; Zhang et al., 2020; Oikarinen et al., 2021). This includes employing adversarial training strategies (Pattanaik et al., 2018) and introducing methods that enhance robustness through the use of robustness verification bounds (Zhang et al., 2020; Oikarinen et al., 2021). Additionally, a focus has shifted towards enabling certifiable robustness in DRL agents using Randomized Smoothing (RS) (Wu et al., 2022; Kumar et al., 2022), transforming agents into their "smoothed" counterparts. However, this transformation traditionally occurs only during testing, without additional training.

Unfortunately, despite the progress in enhancing DRL robustness, we found that existing smoothed agents (Wu et al., 2022; Kumar et al., 2022) demonstrate a notable deficiency: they yield substantially lower clean reward and show little improvement in robustness compared to their non-smoothed counterparts. This critical gap, which we will discuss in Section 2 "*Failure in existing smoothed DRL agents*", has been largely overlooked in previous research. This highlights the need for more effective strategies. Furthermore, previous attack evaluations are ineffective at reducing the rewards of smoothed agents as discussed in section 3.1 Table 1, potentially creating an illusion of empirical robustness.

Driven by these challenges, our work aims to significantly enhance the clean reward, robust reward, and robustness guarantee of smoothed DRL agents. We also address the overestimation of robustness in previous studies by introducing a novel smoothing strategy and a more effective attack method. As a result, we present two innovative agents, S-DQN and S-PPO, designed for both discrete and continuous action spaces. Our proposed agents not only achieve high clean rewards but also provide robustness certification, setting new state-of-the-art across various standard RL environments, including Atari games (Mnih et al., 2013) and continuous control tasks (Brockman et al., 2016).

[1]UC San Diego. Correspondence to: Chung-En Sun <cesun@ucsd.edu>, Tsui-Wei Weng <lweng@ucsd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

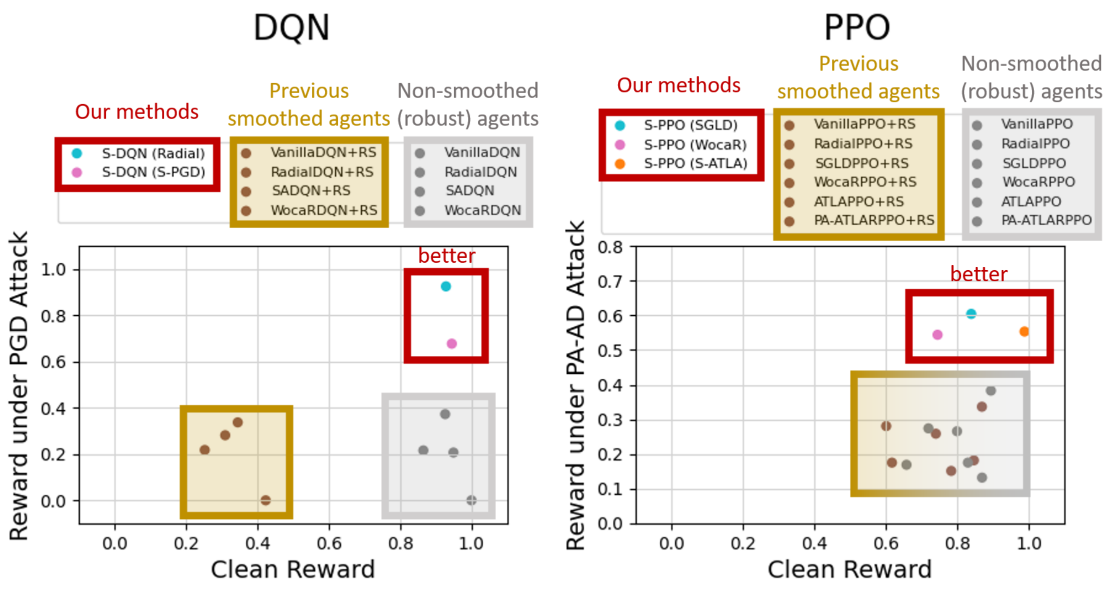

Figure 1. The clean reward and reward under attack for DQN and PPO agents. The presented reward is normalized and averaged across environments. Our S-DQN and S-PPO agents (in the Red boxes) exhibit significantly improved clean reward and robustness in comparison to the previous smoothed agents (in the Brown boxes) and the non-smoothed robust agents (in the Gray boxes).

Our contributions are two-fold:

1. We identify and address the shortcomings in existing smoothed DRL agents, particularly concerning their low clean rewards and limited robustness. To address the limitation, we introduce the first robust DRL training algorithms utilizing Randomized Smoothing (RS) for both discrete actions (S-DQN) and continuous actions (S-PPO). Additionally, we introduce new smoothing strategies and a new attack (Smoothed Attack) to fix the overestimation of robustness in the previous works.

2. Our agents establish a new state-of-the-art record on both robust reward and clean reward. Our S-DQN and S-PPO achieve a $2.52\times$ and $1.80\times$ increase in reward respectively, outperforming existing best **smoothed agents** under the strongest attack. Notably, our S-DQN and S-PPO also surpass previous **best (non-smoothed) robust agents** by $2.70\times$ and $1.58\times$ increase in reward respectively.

We structure our paper as follows: In Section 2, we discuss the issue of low clean reward in existing smoothed DRL agents. In Section 3, we introduce the main algorithms of S-DQN and S-PPO. In Section, 4, we derive the robustness certification for S-DQN and S-PPO. In Section 5, we evaluate the performance of S-DQN and S-PPO in terms of both robust reward and robustness guarantee. In Section 6, we provide background information relevant to our work. Finally, in Section 7, we summarize our work and discuss potential future directions.

2. Failure in existing Smoothed DRL Agents

Randomized Smoothing (RS) is a known technique for enhancing robustness in Deep Reinforcement Learning (DRL). However, our analysis reveals a critical drawback: **the clean reward of all previously studied smoothed agents is notably low with no improvement on the robust reward compared to the non-smoothed agents, as demonstrated with the yellow boxes in Figure 1**. In the DQN framework, previous smoothed agents experience notable reward degradation due to the noise from RS. This degradation persists even under attack scenarios, where no improvement in robust reward is observed. The same pattern is evident with PPO agents: the previous smoothed agents display diminished clean rewards compared to their non-smoothed versions, with only marginal enhancements on the robust reward. For further context on these previous studies, please refer to Section 6.

In contrast, our proposed S-DQN and S-PPO, highlighted in Figure 1 with red boxes, outperform all the previous smoothed agents (Wu et al., 2022; Kumar et al., 2022) and non-smoothed robust agents (Zhang et al., 2020; Oikarinen et al., 2021; Liang et al., 2022; Zhang et al., 2021; Sun et al., 2022) in both robustness and clean reward. This suggests the feasibility of mitigating the adverse effects of randomized smoothing while significantly enhancing robustness. In the following section, we introduce our novel approaches: S-DQN for discrete actions and S-PPO for continuous actions.

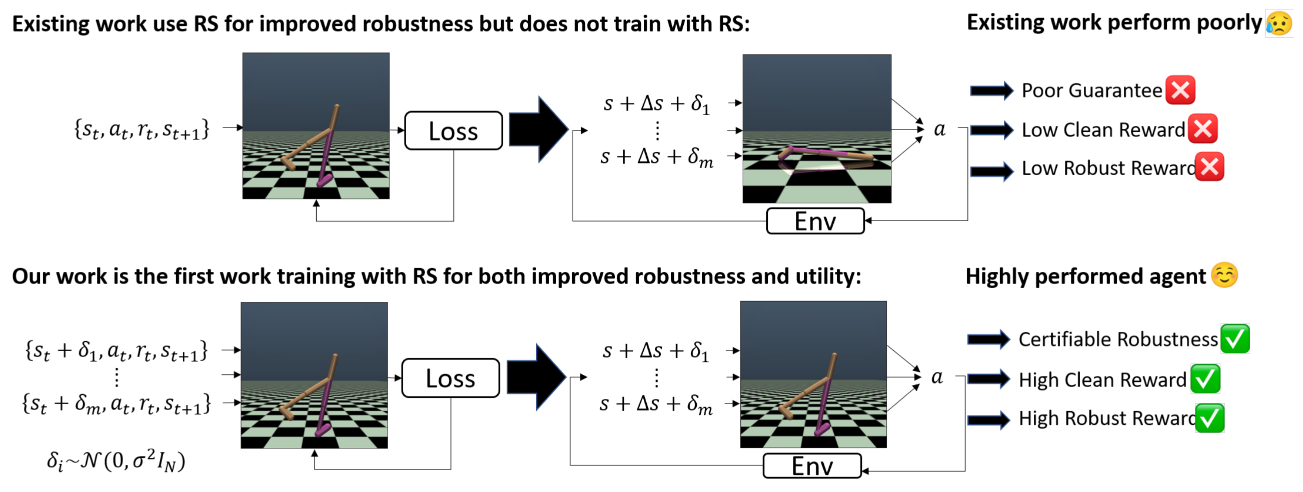
Existing work use RS for improved robustness but does not train with RS:
Existing work perform poorly
$\{s_t, a_t, r_t, s_{t+1}\}$
Loss
$s + \Delta s + \delta_1$
$s + \Delta s + \delta_m$
a
Env
Poor Guarantee
Low Clean Reward
Low Robust Reward
Our work is the first work training with RS for both improved robustness and utility:
Highly performed agent
$\{s_t + \delta_1, a_t, r_t, s_{t+1}\}$
$\{s_t + \delta_m, a_t, r_t, s_{t+1}\}$
$\delta_i \sim \mathcal{N}(0, \sigma^2 I_N)$
Certifiable Robustness
High Clean Reward
High Robust Reward

Figure 2. The overview of our framework. We propose new DRL training algorithms leveraging Randomized Smoothing, achieving strong certifiable robustness, high clean reward, and high robust reward simultaneously.

3. Learning Robust DRL Agents with Randomized Smoothing

In this section, we propose first training algorithms leveraging Randomized Smoothing (RS) to achieve certifiably robust agents, solving the issues mentioned in Section 2 and effectively boosting the robustness as shown in Figure 1. The overview of our framework is shown in Figure 2. Our primary focus centers on two representative RL algorithms: DQN for discrete action space, and PPO for continuous action space, which are the focus of prior works in robust DRL literature (Zhang et al., 2020; Oikarinen et al., 2021; Liang et al., 2022; Zhang et al., 2021; Sun et al., 2022; Wu et al., 2022; Kumar et al., 2022).

3.1. S-DQN (Smoothed - Deep Q Network)

We describe the details of training, testing, and evaluating S-DQN in the following paragraphs.

Training and loss function. The training process of S-DQN is shown in Figure 3 (a), which involves two main steps: collecting transitions and updating the networks. First, we collect the transitions $\{s_t, a_t, r_t, s_{t+1}\}$ with noisy states, which can be formulated as follows:

$$a_t = \begin{cases} \arg\max_a Q(D(\tilde{s}_t;\theta), a), & \text{with probability } 1-\epsilon \\ \text{Random Action}, & \text{with probability } \epsilon \end{cases} \tag{1}$$

where $\tilde{s}_t$ is the state with noise $\tilde{s}_t = s_t + \mathcal{N}(0, \sigma^2 I_N)$, D is the denoiser, Q is the pretrained Q-network, and σ is the standard deviation of the Gaussian distribution. Here, we introduce a denoiser D before the Q-network, aiming to alleviate the side effects of the low clean reward resulting from the noisy states. After collecting the transitions, they are stored in the replay buffer. In the second stage, we sample some transitions from the replay buffer and update the parameters of the denoiser D. The entire loss function is designed with two parts, reconstruction loss $\mathcal{L}_\text{R}$ and temporal difference loss $\mathcal{L}_\text{TD}$:

$$\mathcal{L} = \lambda_1 \mathcal{L}_\text{R} + \lambda_2 \mathcal{L}_\text{TD}, \tag{2}$$

where λ_1 and λ_2 are the hyperparameters. Suppose the sampled transition is $\{s, a, r, s'\}$, the reconstruction loss $\mathcal{L}_\text{R}$ is defined as:

$$\mathcal{L}_\text{R} = \frac{1}{N} ||D(\tilde{s};\theta) - s||_2^2, \tag{3}$$

where $\tilde{s} = s + \mathcal{N}(0, \sigma^2 I_N)$, and N is the dimension of the state. The reconstruction loss is the mean square error (MSE) between the original state and the output of the denoiser. This loss aims to train the denoiser D to effectively reconstruct the original state. The temporal difference loss $\mathcal{L}_\text{TD}$ is defined as:

$$\mathcal{L}_\text{TD} = \begin{cases} \frac{1}{2\zeta}\eta^2, & \text{if } |\eta| < \zeta \\ |\eta| - \frac{\zeta}{2}, & \text{otherwise} \end{cases} \tag{4}$$
$$\eta = r + \gamma \max_{a'} Q(s', a') - Q(D(\tilde{s};\theta), a),$$

where ζ is set to 1. Our designed $\mathcal{L}_\text{TD}$ is different from the common temporal difference loss in the DQN learning: the current Q-value is estimated with the denoised state (the output of D) and the target Q-value remains clean without noisy input. Note that the pretrained Q-network Q can be replaced with robust agents such as RadialDQN (Oikarinen et al., 2021) and our S-DQN framework can also be combined with adversarial training to further improve the robustness. We will discuss this later in Section 5. The full training algorithm can be found in Appendix A.1.1 Algorithm 1.

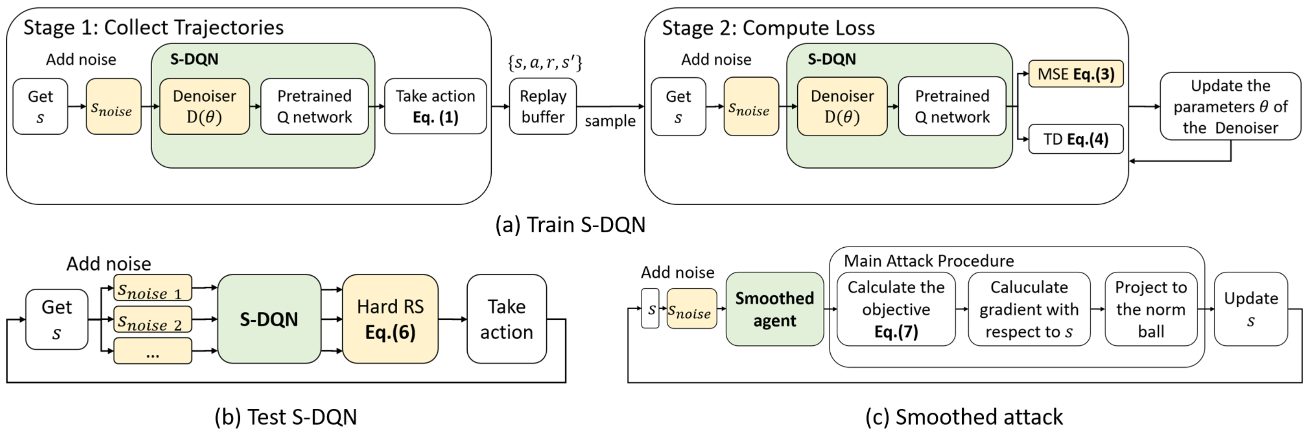

Figure 3. The flow chart of: (a) training process of S-DQN, (b) testing process of S-DQN, (c) our Smoothed Attack pipeline for smoothed agents, which is much more effective than non-smoothed attack.

Testing with hard randomized smoothing. The testing process of S-DQN is shown Figure 3 (b). In the testing stage, we need to obtain the smoothed Q-values of S-DQN. We leverage the hard Randomized Smoothing (hard RS) strategy to enhance robustness, which will be further discussed in Section 4. We first define the hard Q-value Q_h as follows:

$$Q_h(s, a) = \mathbb{1}_{\{a = \arg\max_{a'} Q(s, a')\}} \tag{5}$$

Note that the hard Q-value Q_h is always in $[0, 1]$. Then, we define the hard RS for S-DQN as follows:

$$\widetilde{Q}(s, a) = \mathbb{E}_{\delta \sim \mathcal{N}(0, \sigma^2 I_N)} Q_h(D(s + \delta), a). \tag{6}$$

In practice, we need to estimate the expectation to get $\widetilde{Q}$, which can be done by using Monte Carlo sampling. The action is then selected by taking $\arg\max_a \widetilde{Q}(s, a)$. The full algorithm is in Appendix A.1.2 Algorithm 2.

New attack framework: Smoothed attack. In (Wu et al., 2022), they evaluated all the smoothed DQN agents with the classic Projected Gradient Descent (PGD) attack. However, we found that the classic PGD attack is ineffective in decreasing the reward of the smoothed DQN agents as shown in Table 1. Hence, we propose a new attack framework named Smoothed Attack, which is specifically designed for the smoothed agents to evaluate our S-DQN. The pipeline of Smoothed Attack is shown in Figure 3 (c). The objective of Smoothed Attack is as follows:

$$\min_{\Delta s} \log \frac{\exp Q(D(\tilde{s} + \Delta s), a^*)}{\Sigma_a \exp Q(D(\tilde{s} + \Delta s), a)}, \text{ s.t. } ||\Delta s||_p \leq \epsilon, \tag{7}$$

where $a^* = \arg\max_a \widetilde{Q}(s, a)$, $\widetilde{Q}(s, a)$ is defined in Eq.(6), $\tilde{s} = s + \mathcal{N}(0, \sigma^2 I_N)$, ϵ is the attack budget, and $p = 2$ or ∞ in our setting. In our Smoothed Attack, the state with perturbation is added with a noise sampled from Gaussian distribution with the corresponding smoothing variance σ.

Table 1. The comparison between our smoothed attacks (S-PGD and S-PA-AD) and the existing attacks. A lower reward means the attack is stronger. Our S-PGD attack reduces 61.8% of the reward of S-DQN on average, which is over 2.62× stronger than 23.6% of the classic PGD attack. Our S-PA-AD attack reduces 55.4% of the reward of S-DQN on average, which is over 1.15× stronger than 48.1% of the original version of PA-AD attack. The ℓ_∞ budget is set to $\epsilon = 0.05$ in all the attacks.

Agents	Environments	No Attack	classic PGD Attack	**S-PGD Attack (Ours)**
S-DQN	Pong	20.4 ± 0.5	19.4 ± 2.1	$\mathbf{18.4 \pm 2.1}$
	Freeway	34.0 ± 0.0	32.0 ± 1.4	$\mathbf{6.6 \pm 2.2}$
	RoadRunner	47480 ± 8807	17740 ± 3718	$\mathbf{0 \pm 0}$
Agents	Environments	No Attack	PA-AD	**S-PA-AD (Ours)**
S-DQN	Pong	20.4 ± 0.5	19.4 ± 0.8	$\mathbf{18.6 \pm 1.2}$
	Freeway	34.0 ± 0.0	19.8 ± 1.5	$\mathbf{13.0 \pm 2.1}$
	RoadRunner	47480 ± 8807	$\mathbf{0 \pm 0}$	$\mathbf{0 \pm 0}$

This setting can be integrated with various existing attacks, such as PGD attack and PA-AD (Sun et al., 2022), by replacing the objective with the Smoothed Attack objective in Eq.(7). The comparison of our Smoothed Attack (S-PGD and S-PA-AD) against the PGD attack and PA-AD attack is in Table 1. The full algorithm of our smoothed attack is in Appendix A.1.3 Algorithm 3.

3.2. S-PPO (Smoothed - Proximal Policy Optimization)

The specifics of training, testing, and evaluating our proposed S-PPO are outlined in the following paragraphs.

Training and loss function. PPO agents demonstrate enhanced tolerance to Gaussian noise in contrast to DQN agents. This attribute allows us to directly employ RS for training the PPO agents. The training process of S-PPO is shown in Figure 4. Initially, we gather trajectories using the smoothed policy and subsequently update both the value network and the policy network. In the trajectory collection phase, We use the Median Smoothing (Chiang et al., 2020)

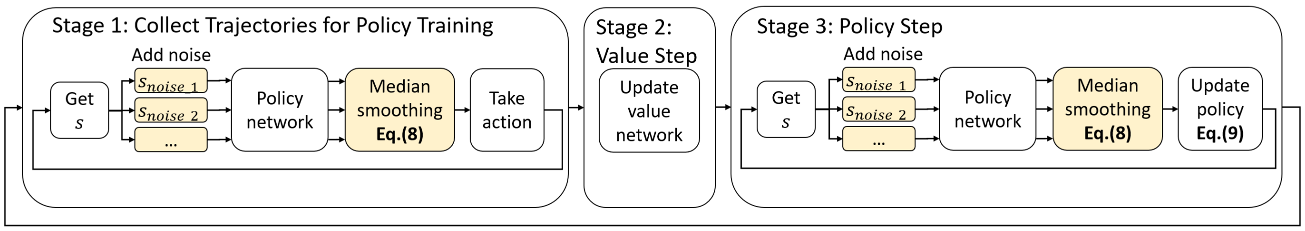

Figure 4. The training process of S-PPO.

strategy to smooth our agents. The median value has a nice property: it is almost unaffected by the outliers. Hence, Median Smoothing can give a better estimation of the expectation than mean smoothing when the number of samples is small. The smoothed policy of S-PPO is defined as follows:

$$\tilde{\pi}_i(a|s) = \mathcal{N}(\widetilde{M}_i, \widetilde{\Sigma}_i^2),\ \forall i \in \{1, ..., N_{\text{action}}\} \tag{8}$$

where $\widetilde{M}_i = \sup\{M \in \mathbb{R} | \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I_N)}[a_i^{\text{mean}} \leq M] \leq p\}$, $\widetilde{\Sigma}_i = \sup\{\Sigma \in \mathbb{R} | \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I_N)}[a_i^{\text{std}} \leq \Sigma] \leq p\}$, $(a_i^{\text{mean}}, a_i^{\text{std}})$ is the output of policy network given a state with noise $s + \delta$ as input, which represents the mean and standard deviation of the i-th coordinate of the action, N_{action} is the dimension of the action, and p is the percentile.

Now, we define the loss function for S-PPO as follows:

$$\mathcal{L}_{\tilde{\pi}}(\theta) = -\mathbb{E}_t[\min(\mathcal{R}_{\tilde{\pi}}\hat{A}_t, \text{clip}(\mathcal{R}_{\tilde{\pi}}, 1 - \epsilon_c, 1 + \epsilon_c)\hat{A}_t)],$$
$$\mathcal{R}_{\tilde{\pi}} = \frac{\tilde{\pi}(a_t|s_t;\theta)}{\tilde{\pi}(a_t|s_t;\theta_{\text{old}})}, \tag{9}$$

where $\hat{A}_t$ is the advantage, and ϵ_c is the clipping hyperparameter. This is the loss of the classic PPO algorithm combined with RS. Note that our S-PPO can also be combined with robust PPO algorithms such as SGLD (Zhang et al., 2020), Radial (Oikarinen et al., 2021), or WocaR (Liang et al., 2022).

Adversary training for S-PPO. In ATLA-PPO (Zhang et al., 2021) and PA-ATLA-PPO (Sun et al., 2022), they jointly train a policy network and an adversarial network to robustify the PPO agents. Our S-PPO can also be combined with these adversarial training methods by modifying the adversarial policy and objective to align with the smoothed one. The smoothed adversarial policy is defined as follows:

$$\tilde{\mathcal{A}}_i(\Delta p|s) = \mathcal{N}(\widetilde{M}_i, \widetilde{\Sigma}_i^2),\ \forall i \in \{1, ..., N_{\Delta p}\} \tag{10}$$

where $\mathcal{A}$ is the adversary, Δp is the attack direction, $\widetilde{M}_i = \sup\{M \in \mathbb{R} | \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I_N)}[\Delta p_i^{\text{mean}} \leq M] \leq p\}$, $\widetilde{\Sigma}_i = \sup\{\Sigma \in \mathbb{R} | \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I_N)}[\Delta p_i^{\text{std}} \leq \Sigma] \leq p\}$, $(\Delta p_i^{\text{mean}}, \Delta p_i^{\text{std}})$ is the output of the adversarial network given a state with noise $s + \delta$ as input, which represents the mean and standard deviation of the i-th coordinate of the perturbation, and $N_{\Delta p}$ is the dimension of Δp.

The loss of training smoothed adversarial policy is defined as follows, which is designed to minimize the surrogate reward:

$$\mathcal{L}_{\tilde{\mathcal{A}}}(\theta) = \mathbb{E}_t[\min(\mathcal{R}_{\tilde{\mathcal{A}}}\hat{A}_t, \text{clip}(\mathcal{R}_{\tilde{\mathcal{A}}}, 1 - \epsilon_c, 1 + \epsilon_c)\hat{A}_t)],$$
$$\mathcal{R}_{\tilde{\mathcal{A}}} = \frac{\tilde{\mathcal{A}}(\Delta p_t|s_t;\theta)}{\tilde{\mathcal{A}}(\Delta p_t|s_t;\theta_{\text{old}})}. \tag{11}$$

In ATLA, Δp represents the direction of the state change Δs used to perturb the state of the PPO agents. On the other hand, in PA-ATLA, Δp represents the direction of the action change Δa. To induce the PPO agents to undergo the specified action change Δa, a Fast Gradient Sign Method (FGSM) attack is then executed to perturb the state. We use S-FGSM, which is the Smoothed Attack, while using the PA-ATLA algorithm to perform adversarial training for S-PPO.

The full algorithm of training S-PPO is in Appendix A.2.1 Algorithm 4 and 5.

Testing. We also use Median Smoothing during testing to obtain the smoothed policy. However, we use the smoothed deterministic policy as follows:

$$\tilde{\pi}_{i,\text{det}}(s) = \widetilde{M}_i,\ \forall i \in \{1, ..., N_{\text{action}}\}, \tag{12}$$

where $\widetilde{M}_i = \sup\{M \in \mathbb{R} | \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I_N)}[a_i^{\text{mean}} \leq M] \leq p\}$, and a_i^{mean} is the output of policy network given a state with noise $s + \delta$ as input ($a_i^{\text{mean}} = \pi_{i,\text{det}}(s + \delta)$) representing the mean of the i-th coordinate of the action. Here we only use the a^{mean} value of the output of the policy network for smoothing.

Attack. To evaluate the performance of our S-PPO, we use the Maximal Action Difference (MAD) Attack and Minimum Robust Sarsa (Min-RS) Attack proposed in Zhang et al. (2020). Furthermore, We also evaluate our S-PPO under the two strongest optimal adversaries (Zhang et al., 2021; Sun et al., 2022). (Zhang et al., 2021) proposed the Optimal Attack, employing an adversarial agent to perturb the states. (Sun et al., 2022) proposed the state-of-the-art PA-AD attack, where an adversarial agent determines a direction and uses FGSM to perturb the states based on the specified direction. In the PPO setting, we did not find a significant difference between the smoothed attack and the non-smoothed attack (see Table 16 in Appendix A.14), and hence, we used the original setting for every attack.

4. Robustness certification

The strength of the smoothed agents lies in their certifiable robustness. However, previous literature (Wu et al., 2022; Kumar et al., 2022) fails to give a good expression for the certified radius of DQN agents and has not derived the action bound for PPO agents. To make the study of certifiable robustness more complete, we formally formulate the certified radius, action bound, and reward lower bound of our S-DQN and S-PPO agents.

Certified Radius for S-DQN. The certified radius for our S-DQN is defined as follows:

$$R_t = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \Phi^{-1}(\widetilde{Q}(s_t, a_2))), \quad (13)$$

where a_1 is the action with the largest Q-value among all the other actions, a_2 is the "runner-up" action, R_t is the certified radius at time t, Φ is the CDF of normal distribution, σ is the smoothing variance, and $\widetilde{Q}(s, a)$ is defined in Eq.(6). As long as the ℓ_2 perturbation is bounded by R_t, the action is guaranteed to be the same.

Note that our expression of the certified radius is different from the one proposed in CROP (Wu et al., 2022) since we use hard RS. In CROP, they took the average of the output samples, which is the mean smoothing strategy. However, this might not lead to a precise estimation of the certified radius since it requires estimating the output range $[V_{\text{min}}, V_{\text{max}}]$ of the Q-network. The certified radius proposed in CROP is shown as follows:

$$\begin{aligned} R_t =& \frac{\sigma}{2}(\Phi^{-1}(\frac{\widetilde{Q}_{\text{CROP}}(s_t, a_1) - \Delta - V_{\text{min}}}{V_{\text{max}} - V_{\text{min}}}) \\ &- \Phi^{-1}(\frac{\widetilde{Q}_{\text{CROP}}(s_t, a_2) + \Delta - V_{\text{min}}}{V_{\text{max}} - V_{\text{min}}})), \end{aligned} \quad (14)$$

where R_t is the certified radius at time step t, $Q_{\text{CROP}} : \mathcal{S} \times \mathcal{A} \to [V_{\text{min}}, V_{\text{max}}]$, $\widetilde{Q}_{\text{CROP}}(s, a) = \frac{1}{m}\Sigma_{i=1}^{m} Q_{\text{CROP}}(s + \delta_i, a)$, $\delta_i \sim \mathcal{N}(0, \sigma^2 I_N), \forall i \in \{1, ..., m\}$, a_1 is the action with the largest Q-value, a_2 is the "runner-up" action, $\Delta = (V_{\text{max}} - V_{\text{min}})\sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}}$, Φ is the CDF of standard normal distribution, m is the number of the samples, and α is the one-side confidence parameter. Based on this expression, the output range of the Q-network $[V_{\text{min}}, V_{\text{max}}]$ can significantly affect the certified radius. The certified radius is small when the output range of the Q-network $[V_{\text{min}}, V_{\text{max}}]$ is large (e.g. Suppose $\widetilde{Q}_{\text{CROP}}(s_t, a_1) = 3$, $\widetilde{Q}_{\text{CROP}}(s_t, a_2) = -3$, $\sigma = 0.1$, $m = 100$, and $\alpha = 0.05$. The certified radius is only 0.007 under $[V_{\text{min}}, V_{\text{max}}] = [-10, 10]$. Instead, if we narrow down the interval to $[V_{\text{min}}, V_{\text{max}}] = [-3.5, 3.5]$, the certified radius grows to 0.086). CROP estimated $[V_{\text{min}}, V_{\text{max}}]$ by sampling some trajectories and finding the maximum and the minimum of the Q-values. However, if the actual interval is much larger than the estimation (which is likely to happen in practice since it is impossible to go over all the states), the certified radius can be significantly overestimated.

In contrast, our hard RS strategy eliminates the need for estimating $[V_{\text{min}}, V_{\text{max}}]$, resulting in a more precise estimation of the certified radius. Moreover, based on Eq.(13), the certified radius of our S-DQN is not influenced by the out range of the Q-network $[V_{\text{min}}, V_{\text{max}}]$, which gives a more stable guarantee. Detailed experiments for the certified radius of our S-DQNs versus the CROP agents are provided in Appendix A.8, demonstrating that our S-DQNs achieve a larger radius. The proof of the certified radius for S-DQN can be found in Appendix A.5.

Action Bound for S-PPO. Unfortunately, unlike the discrete action setting, there is no guarantee that the action will not change under a certain radius in the continuous action setting. Hence, we derive the **Action Bound**, which bounds the policy of S-PPO agents in a close region:

$$\tilde{\pi}_{\text{det},\underline{p}}(s_t) \preceq \tilde{\pi}_{\text{det},p}(s_t + \Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p}}(s_t), \text{ s.t. } ||\Delta s||_2 \leq \epsilon, \quad (15)$$

where $\tilde{\pi}_{i,\text{det},p}(s) = \sup\{a_i \in \mathbb{R} | \mathbb{P}_{\delta \sim \mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s + \delta) \leq a_i] \leq p\}, \forall i \in \{1, ..., N_{\text{action}}\}$, $\underline{p} = \Phi(\Phi^{-1}(p) - \frac{\epsilon}{\sigma})$, $\overline{p} = \Phi(\Phi^{-1}(p) + \frac{\epsilon}{\sigma})$, and p is the percentile. We designed a metric based on this action bound to evaluate the certified robustness of S-PPO agents. See Appendix A.9 for more details. The proof of the action bound can be found in Appendix A.6.

Reward lower bound for smoothed agents. By viewing the whole trajectory as a function F_π, we define $F_\pi : \mathbb{R}^{H \times N} \to \mathbb{R}$ that maps the vector of perturbations for the whole trajectory $\boldsymbol{\Delta s} = [\Delta s_0, ..., \Delta s_{H-1}]^\top$ to the cumulative reward. Then, the reward lower bound is defined as follows:

$$\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) \geq \widetilde{F}_{\pi,\underline{p}}(\boldsymbol{0}), \text{ s.t. } ||\boldsymbol{\Delta s}||_2 \leq B, \quad (16)$$

where $\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) = \sup\{r \in \mathbb{R} | \mathbb{P}_{\boldsymbol{\delta} \sim \mathcal{N}(0,\sigma^2 I_{H \times N})}[F_\pi(\boldsymbol{\delta} + \boldsymbol{\Delta s}) \leq r] \leq p\}$, $\widetilde{F}_{\pi,\underline{p}}(\boldsymbol{0}) = \sup\{r \in \mathbb{R} | \mathbb{P}_{\boldsymbol{\delta} \sim \mathcal{N}(0,\sigma^2 I_{H \times N})}[F_\pi(\boldsymbol{\delta}) \leq r] \leq \underline{p}\}$, $\boldsymbol{\delta} = [\delta_0, ..., \delta_{H-1}]^\top$, $\underline{p} = \Phi(\Phi^{-1}(p) - \frac{B}{\sigma})$, H is the length of the trajectory, and B is the ℓ_2 attack budget for the entire trajectory. If the attack budget of each state is ϵ, then $B = \epsilon\sqrt{H}$. This bound ensures that the reward will not fall below a certain value while given any ℓ_2 perturbation with budget B. We will discuss the reward lower bound for all the smoothed agents in Section 5. The proof of the reward lower bound can be found in Appendix A.7.

In practice, it is necessary to introduce the confidence interval, which can change the bounds based on the sample number, while estimating all the bounds introduced above. The details of estimating the bounds are provided in Appendix A.4.

5. Experiment

Setup. We follow the previous robust DRL literature to conduct experiments on Atari (Mnih et al., 2013) and Mujoco (Brockman et al., 2016) benchmarks. In our DQN settings, the evaluations are done in three Atari environments — Pong, Freeway, and RoadRunner. We train the denoiser D with different base agents and with adversarial training. Our methods are listed as follows:

- S-DQN ({*Base agent*}): S-DQN combined with a certain base agents. {*base agent*} can be Radial (Oikarinen et al., 2021) or Vanilla (simple DQN).
- S-DQN (S-PGD): S-DQN (Vanilla) adversarially trained with our proposed S-PGD.

We compare our S-DQN with the following baselines:

- Non-smoothed robust agents: WocaRDQN (Liang et al., 2022), RadialDQN (Oikarinen et al., 2021), SADQN (Zhang et al., 2020).
- Previous smoothed agents (Wu et al., 2022; Kumar et al., 2022): WocaRDQN+RS, RadialDQN+RS, SADQN+RS. We use {*base agent*}+RS to denote them.

In our PPO settings, the evaluations are done on two continuous control tasks in the Mujoco environments — Walker and Hopper. We train each agent 15 times and report the median performance as suggested in Zhang et al. (2020) since the training variance of PPO algorithms is high. Our methods are listed as follows:

- S-PPO ({*base algorithm*}): S-PPO combined with a certain base algorithms. {*base algorithm*} can be SGLD (Zhang et al., 2020), Radial (Oikarinen et al., 2021), WocaR (Liang et al., 2022), or Vanilla (simple PPO).
- S-PPO (S-ATLA), S-PPO (S-PA-ATLA): S-PPO with smoothed adversarial training described in Section 3.2 ”*Adversary training for S-PPO*”.

We compare our S-PPO with the following baselines:

- Non-smoothed robust agents: WocaRPPO (Liang et al., 2022), PA-ATLAPPO (Sun et al., 2022), ATLAPPO (Zhang et al., 2021), RadialPPO (Oikarinen et al., 2021), SGLDPPO (Zhang et al., 2020).
- Previous smoothed agents: WocaRPPO+RS, PA-ATLAPPO+RS, ATLAPPO+RS, RadialPPO+RS, SGLDPPO+RS.

See Appendix A.3 for more details about our setting.

Robust reward and lower bound for S-DQN. The robust reward of our S-DQN under ℓ_∞ PGD attack and PA-AD attack (Sun et al., 2022) is shown in Table 2. The presented rewards are first normalized and then averaged across the three environments. Note that we use our stronger S-PGD and S-PA-AD introduced in Section 3.1 to evaluate all the smoothed agents. Our S-DQN (Radial), S-DQN (S-PGD), and S-DQN (Vanilla) exhibit superior performance compared to the state-of-the-art robust RadialDQN and WocaRDQN. Notably, our S-DQN (Vanilla) already demonstrates greater robustness than RadialDQN without further combining with other robust agents. The poor performance of rows (c) suggests that the previous smoothed agents struggle to tolerate the Gaussian noise introduced by RS and fail to enhance the reward under attack. More detailed experiment results and discussion about the robust reward for S-DQN can be found in Appendix A.10.

Figure 5 shows the reward lower bound of our S-DQNs. Our S-DQNs exhibit high reward lower bounds compared to the previous smoothed agents, indicating that our method can enhance not only the empirical robustness but also the robustness guarantee. More detailed experiment results for the reward lower bound can be found in Appendix A.11.

Robust reward and lower bound for S-PPO. The robust reward of our S-PPO under attacks is shown in Table 3. The presented rewards are first normalized and then averaged across the two environments. Our S-PPO agents constantly outperform their counterparts (previous smoothed agents and the SOTA robust agents) for all robust training algorithms. Through comparing rows (b) and (c), the previous smoothed agents exhibit lower clean reward and only marginal improvement on the reward under attacks, suggesting that naively applying RS during the test time cannot improve the robustness of PPO agents. In addition, our S-PPO agents receive a much higher clean reward on average, showing that our RS training approach can further boost performance in the non-adversarial setting. More detailed experiment results and discussion about the robust reward for S-PPO can be found in Appendix A.12.

Our S-PPOs also exhibit higher reward lower bounds than the previous smoothed PPO agents, which is shown in Figure 6. More detailed experiment results for the reward lower bound can be found in Appendix A.13.

Table 2. The average normalized reward of DQN agents under ℓ_∞ PGD attack and PA-AD attack. Our S-DQNs achieve the highest robust reward, especially under a large attack budget ϵ.

Avg normalized reward	Clean	PGD attack					PA-AD attack
$\epsilon(\ell_\infty)$		0.01	0.02	0.03	0.04	0.05	0.05
(a) Ours:							
S-DQN (Radial)	0.929	0.928	**0.932**	**0.830**	**0.788**	**0.735**	**0.669**
S-DQN (S-PGD)	0.945	0.945	0.886	0.775	0.700	0.450	0.552
S-DQN (Vanilla)	0.989	0.818	0.660	0.601	0.498	0.377	0.442
(b) SOTA robust agents:							
RadialDQN	0.926	**0.947**	0.770	0.337	0.206	0.210	0.248
SADQN	0.949	0.825	0.302	0.205	0.207	0.185	0.224
WocaRDQN	0.865	0.617	0.218	0.204	0.208	0.216	0.210
VanillaDQN	**1.000**	0.000	0.000	0.000	0.000	0.000	0.000
(c) Previous smoothed agents:							
RadialDQN+RS	0.310	0.295	0.281	0.264	0.271	0.240	0.265
SADQN+RS	0.345	0.316	0.331	0.231	0.219	0.227	0.230
WocaRDQN+RS	0.253	0.222	0.218	0.218	0.218	0.218	0.214
VanillaDQN+RS	0.424	0.000	0.000	0.000	0.000	0.000	0.000

Table 3. The average normalized reward of PPO agents under ℓ_∞ attack. Our S-PPO ({*base algorithm*}) constantly achieves a much higher worst reward compared to {*base algorithm*}PPO (row (b)) and {*base algorithm*}PPO+RS (row (c)), where {*base algorithm*} represents various robust training algorithms.

Avg normalized reward	Clean Reward	MAD	Min-RS	Optimal	PA-AD	Worst Reward
(a) Ours:						
S-PPO (SGLD)	0.840	**0.837**	**0.745**	0.617	**0.604**	**0.604**
S-PPO (Radial)	0.709	0.641	0.263	0.262	0.336	0.262
S-PPO (WocaR)	0.745	0.726	0.566	0.531	0.544	0.531
S-PPO (S-ATLA)	**0.989**	0.784	0.449	**0.844**	0.553	0.449
S-PPO (S-PA-ATLA)	0.935	0.753	0.481	0.234	0.296	0.234
S-PPO (Vanilla)	0.929	0.804	0.459	0.226	0.265	0.226
(b) SOTA robust agents:						
SGLDPPO	0.800	0.760	0.384	0.418	0.266	0.266
RadialPPO	0.658	0.628	0.284	0.133	0.169	0.133
WocaRPPO	0.895	0.788	0.342	0.438	0.383	0.342
ATLAPPO	0.830	0.454	0.232	0.237	0.175	0.175
PA-ATLAPPO	0.720	0.609	0.206	0.220	0.274	0.206
VanillaPPO	0.870	0.595	0.166	0.136	0.132	0.132
(c) Previous smoothed agents:						
SGLDPPO+RS	0.740	0.728	0.420	0.302	0.259	0.259
RadialPPO+RS	0.617	0.569	0.195	0.163	0.175	0.163
WocaRPPO+RS	0.869	0.797	0.280	0.466	0.336	0.280
ATLAPPO+RS	0.847	0.531	0.251	0.263	0.182	0.182
PA-ATLAPPO+RS	0.601	0.600	0.224	0.279	0.281	0.224
VanillaPPO+RS	0.783	0.585	0.181	0.138	0.151	0.138

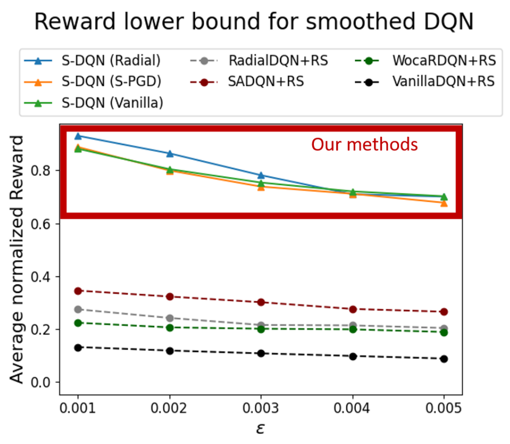

Figure 5. The certified reward lower bound of smoothed DQN agents. Our S-DQNs achieve a much higher lower bound than all the previous smoothed agents.

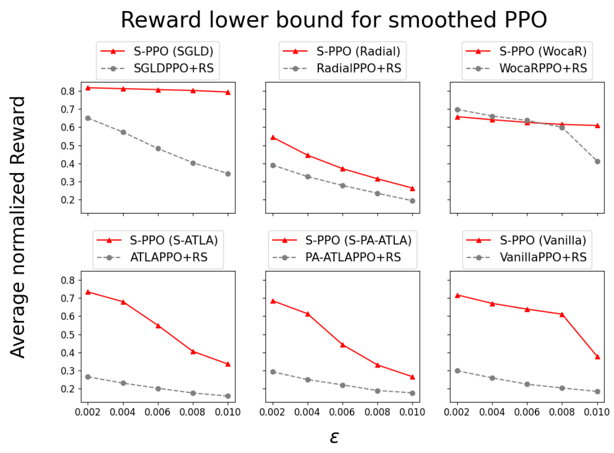

Figure 6. The certified reward lower bound of smoothed PPO agents. Our S-PPOs demonstrate a much higher lower bound compared to previous smoothed agents.

6. Background and related works

Randomized smoothing (RS). Randomized Smoothing (Cohen et al., 2019) has been proved to provide a robustness guarantee to a *smoothed* classifier under ℓ_2 perturbation on input examples. The idea is to transform an arbitrary base classifier into an L-Lipschitz smoothed classifier by adding Gaussian noises to the input. This transformation facilitates *black-box* robustness verification on the smoothed classifier, which ensures the classification result remains unchanged within the certified radius without the need to know the model parameters. This can be formulated as below. Given a base classifier $f : \mathbb{R}^d \to \mathcal{Y}$, and let $\tilde{f} : \mathbb{R}^d \to \mathcal{Y}$ be the smoothed classifier (i.e., f after RS), $\tilde{f}$ can be expressed as $\tilde{f}(x) = \arg\max_{c\in\mathcal{Y}} \mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[f(x+\delta)=c]$, where δ is a random vector following Gaussian distribution $\mathcal{N}(0,\sigma^2 I)$. The smoothed classifier $\tilde{f}$ predicts class c_A with probability p_A, and predicts the "runner-up" class c_B with probability p_B. The certified radius of $\tilde{f}$ is denoted as R such that $\tilde{f}(x+\Delta)=\tilde{f}(x),\ \forall ||\Delta||_2 \le R$. R can be derived as $R = \frac{\sigma}{2}(\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$, where Φ^{-1} is the inversed Gaussian CDF. There have been techniques improving the limitations of RS. For example, (Salman et al., 2020) proposed to add a denoiser before the original image classifier to remove the Gaussian noises introduced by RS. This approach gives the classifier the ability to tolerate large noises. Our method is the first work leveraging Denoised Smoothing in the DRL setting.

Learning Robust DRL agents. There are several existing works of learning robust DRL agents through robust training. These agents are non-smoothed DRL agents and their performance is shown in Figure 1 (the grey boxes). **SA-RL (SADQN and SGLDPPO) (Zhang et al., 2020)**

Table 4. The comparison between our methods and other DRL agents. Our methods are desirable in both empirical robustness and robustness guarantee.

Methods	Empirical Robustness		Robustness Guarantee		
	Clean Reward↑	Reward under Attack↑	Certified Radius (for DQN)	Action bound (for PPO)	Reward lower bound↑
Our methods:					
S-DQN & S-PPO	**Highest**	**Highest**	**Yes**	**Yes**	**Highest**
SOTA robust agents:					
SA-RL & RADIAL-RL & WocaR-RL	High	High	No	No	No
ATLAPPO & PA-ATLAPPO (PPO only)	High	High	No DQN implementation	No	No
Previous smoothed agents:					
CROP (DQN only)	Low	Medium	Yes	No PPO implementation	Low
Policy Smoothing (PPO only)	Medium	Medium	No DQN implementation	No derivation	Low

trained robust agents using a robust regularizer based on the total variation distance and KL-divergence between the perturbed policies and the original policies. **RADIAL-RL** (**Oikarinen et al., 2021**) used the adversarial loss based on the robustness verification bounds as a regularizer. **WocaR-RL** (**Liang et al., 2022**) robustify agents through improving the worst-case reward. **ATLAPPO** (**Zhang et al., 2021**) proposed to use the optimal adversary for adversarial training. **PA-ATLAPPO** (**Sun et al., 2022**) improved ATLA by separating the adversary into a RL-based director and a non-RL actor.

Previous smoothed DRL agents. Recently, two works proposed to smooth DRL agents in the test-time. **CROP** (**Wu et al., 2022**) proposed the first framework using RS to study the robustness certification of DRL agents. They showed that the certified radius of a smoothed robustly trained agent is generally larger compared to the smoothed vanilla agents. **Policy Smoothing** (**Kumar et al., 2022**) demonstrated that the robustness guarantee in the Supervised Learning setting cannot directly transfer to the RL setting due to the non-static nature of RL. They provided an alternative proof for the reward lower bound in the RL setting. However, both approaches perform poorly as shown in Figure 1 (the yellow boxes), suggesting that the previous smoothed agents are not usable in practice, emphasizing the necessity of applying our proposed methods.

The detailed comparison among our methods, the robust DRL agents, and previous smoothed agents is shown in Table 4.

7. Conclusion and future works

In this work, we have shown with extensive experiments that our proposed S-DQN and S-PPO agents outperform previous robust agents and smoothed agents in terms of both robustness certificates and robust reward against the current strongest attack, establishing the new state-of-the-art in the field. In future work, we are planning to investigate the idea of leveraging robustness certificates into training to further strengthen the robustness of DRL agents.

Impact Statement

This paper investigates certifiably robust Deep Reinforcement Learning (DRL) agents, with a close connection to safety-critical continuous control domains. We hold the belief that our proposed method has the potential to serve as a foundation for constructing more reliable RL tools, thereby positively influencing the broader societal landscape.

Acknowlegements

This work is supported in part by National Science Foundation (NSF) awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, OAC-2112167, CNS-2100237, CNS-2120019, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute. Thanks to CENIC for the 100Gbps networks. C. Sun and T.-W. Weng are supported by National Science Foundation under Grant No. 2107189 and 2313105. T.-W. Weng also thanks the Hellman Fellowship for providing research support.

References

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chiang, P., Curry, M. J., Abdelkader, A., Kumar, A., Dickerson, J. P., and Goldstein, T. Detection as regression: Certified object detection by median smoothing. *CoRR*, 2020.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. In *ICLR*, 2017.

Kumar, A., Levine, A., and Feizi, S. Policy smoothing for provably robust reinforcement learning. In *ICLR*, 2022.

Liang, Y., Sun, Y., Zheng, R., and Huang, F. Efficient adversarial training without attacking: Worst-case-aware robust reinforcement learning. In *NeurIPS*, 2022.

Lin, Y., Hong, Z., Liao, Y., Shih, M., Liu, M., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. In *ICLR*, 2017.

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

Oikarinen, T. P., Zhang, W., Megretski, A., Daniel, L., and Weng, T. Robust deep reinforcement learning through adversarial loss. In *NeurIPS*, 2021.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. In *AAMAS*, 2018.

Salman, H., Li, J., Razenshteyn, I. P., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. In *NeurIPS*, 2019.

Salman, H., Sun, M., Yang, G., Kapoor, A., and Kolter, J. Z. Denoised smoothing: A provable defense for pretrained classifiers. In *NeurIPS*, 2020.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nat.*, 2016.

Sun, Y., Zheng, R., Liang, Y., and Huang, F. Who is the strongest enemy? towards optimal and efficient evasion attacks in deep RL. In *ICLR*, 2022.

Weng, T., Dvijotham, K. D., Uesato, J., Xiao, K., Gowal, S., Stanforth, R., and Kohli, P. Toward evaluating robustness of deep reinforcement learning with continuous control. In *ICLR*, 2020.

Wu, F., Li, L., Huang, Z., Vorobeychik, Y., Zhao, D., and Li, B. CROP: certifying robust policies for reinforcement learning through functional smoothing. In *ICLR*, 2022.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D. S., and Hsieh, C. Robust deep reinforcement learning against adversarial perturbations on state observations. In *NeurIPS*, 2020.

Zhang, H., Chen, H., Boning, D. S., and Hsieh, C. Robust reinforcement learning on state observations with learned optimal adversary. In *ICLR*, 2021.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.*, 2017.

A. Appendix

Contents

A.1. Detailed algorithms of S-DQN

A.1.1. Training algorithm of S-DQN

The training algorithm of S-DQN is shown in Algorithm 1. The algorithm includes all the details of the training procedure introduced in Section 3.1. We first add a noise to the current state and take action with ϵ-greedy strategy, Then, store the transitions $\{s_t, a_t, r_t, s_{t+1}\}$ into the replay buffer. Note that the state s_t we stored here is the clean state without noise. When updating the denoiser D, we sample a batch of transitions from the replay buffer, add noise to the state again, and compute the loss.

Algorithm 1 Train S-DQN

```
 1: Input: smoothing variance $\sigma$, steps $T$, replay buffer $\mathcal{B}$, Denoiser $D$, pretrained Q network $Q$
 2: for $t = 1$ to $T$ do
 3:     Sample a noise from the normal distribution and add to the state $\tilde{s}_t = s_t + \mathcal{N}(0, \sigma^2 I_N)$
 4:     Select a random action $a_t$ with probability $\epsilon_t$, otherwise $a_t = \arg\max_a Q(D(\tilde{s}_t; \theta), a)$
 5:     Store the transition $\{s_t, a_t, r_t, s_{t+1}\}$ in $\mathcal{B}$
 6:     Sample a batch of samples $\{s, a, r, s'\}$ from $\mathcal{B}$
 7:     Sample a noise from the normal distribution and add to the state $\tilde{s} = s + \mathcal{N}(0, \sigma^2 I_N)$
 8:     Compute the reconstruction loss $\mathcal{L}_{\text{R}} = \text{MSE}(D(\tilde{s}; \theta), s)$
 9:     Compute the temporal difference loss $\mathcal{L}_{\text{TD}} = \text{Huber}(r + \gamma \max_{a'} Q(s', a') - Q(D(\tilde{s}; \theta), a))$
10:     Total loss $\mathcal{L} = \lambda_1 \mathcal{L}_{\text{R}} + \lambda_2 \mathcal{L}_{\text{TD}}$
11:     Perform gradient descent to minimize loss $\mathcal{L}$ and update the parameters $\theta$ of the denoiser $D$
12: end for
```

A.1.2. Testing algorithm of S-DQN

The testing algorithm of S-DQN is shown in Algorithm 2. The algorithm includes all the details of the testing procedure introduced in Section 3.1. We use the hard randomized smoothing strategy to smooth our agent and do Monte Carlo sampling to estimate the expectation. The definition of Q_h is in Eq.(5).

Algorithm 2 Test S-DQN

```
 1: Input: smoothing variance $\sigma$, number of samples $M$, number of the actions $N$, Denoiser $D$, pretrained Q network $Q$
 2: while not end game do
 3:     Get state $s$ from the environment
 4:     for $m = 1$ to $M$ do
 5:         Sample a noise from the normal distribution and add to the state $\tilde{s}_m = s_m + \mathcal{N}(0, \sigma^2 I_N)$
 6:         Store the $Q_h$ value of all the actions $[Q_h(D(\tilde{s}_m), a_1), ..., Q_h(D(\tilde{s}_m), a_N)]$ to the list
 7:     end for
 8:     Take the mean of the $Q_h$ value of each action $\widetilde{Q}(s, a_n) = \frac{1}{M} \Sigma_{m=1}^{M} Q_h(D(\tilde{s}_m), a_n)$
 9:     Choose the action with the maximum $\widetilde{Q}$ value $a^* = \arg\max_{a_n} \widetilde{Q}(s, a_n)$
10:     Take action and get the reward
11: end while
12: Return the total reward
```

A.1.3. Attack algorithm of smoothed attack

The algorithm of our Smoothed Attack (S-PGD) is shown in Algorithm 3. The algorithm includes all the details of the attack procedure introduced in Section 3.1. Note that our Smoothed Attack considers the noise introduced by randomized smoothing.

Algorithm 3 Smoothed Attack (S-PGD)

```
1: Input: number of iterations $T$, attack budget $\epsilon$, smoothing variance $\sigma$, number of samples $M$, Denoiser $D$, pretrained Q
   network $Q$
2: Get state $s$ from the environment
3: $\hat{s} = s$
4: for $t = 1$ to $T$ do
5:     Sample a noise from the normal distribution and add to the state $\tilde{\hat{s}} = \hat{s} + \mathcal{N}(0, \sigma^2 I_N)$
6:     Compute the cross-entropy loss
       $\mathcal{L} = -\log \frac{\exp(Q(D(\tilde{\hat{s}}), a^*))}{\Sigma_a \exp(Q(D(\tilde{\hat{s}}), a))}$,
       where $a^*$ is the original optimal action decided by the agent
7:     Calculate the gradient with respect to $\hat{s}$, and project to the $\ell_2$ or $\ell_\infty$ norm ball
8:     Update $\hat{s}$ by adding the gradient
9: end for
10: Return the perturbed state $\hat{s}$
```

A.2. Detailed algorithms of S-PPO

A.2.1. Training algorithm of S-PPO

The training algorithm of S-PPO is shown in Algorithm 4 and 5. The algorithm includes all the details of the training procedure introduced in Section 3.2. The algorithm of CollectTrajectories function used in step 1 of Algorithm 4 is shown in Algorithm 5.

Algorithm 4 Train S-PPO

```
1:  Input: smoothing variance $\sigma$, attack budget $\epsilon$, number of samples $M$, iterations $T$, Policy network $\pi$, Value network $V$
2:  for $t = 1$ to $T$ do
3:      // Step 1: Collect trajectories for policy training
        $\{\tau_k\} =$ CollectTrajectories()
4:      Compute cumulative reward $\hat{R}_{k,i}$ for each step $i$ in episode $k$ with discount factor $\gamma$
5:      // Step 2: Update the value network with loss
        $\mathcal{L}_V(\theta) = \frac{1}{\Sigma_k |\tau_k|} \Sigma_{\tau_k} \Sigma_i (V(s_{k,i}) - \hat{R}_{k,i})^2$
6:      // Step 3: Update the policy network
7:      for $m = 1$ to $M$ do
8:          Sample a noise from the normal distribution and add to the state $\tilde{s}_{k,i,m} = s_{k,i,m} + \mathcal{N}(0, \sigma^2 I_N)$
9:          Store the output of the policy network $(a^{\text{mean}}_{k,i,m}, a^{\text{std}}_{k,i,m})$ to the list, where $\mathcal{N}(a^{\text{mean}}_{k,i,m}, a^{\text{std}}_{k,i,m}) = \pi(a_{k,i,m} | \tilde{s}_{k,i,m})$
10:     end for
11:     Take the median and obtain the smoothed policy
        $\tilde{\pi}(a_{k,i} | s_{k,i}) = \mathcal{N}(\text{median}(a^{\text{mean}}_{k,i,1}, ..., a^{\text{mean}}_{k,i,M}), \text{median}(a^{\text{std}}_{k,i,1}, ..., a^{\text{std}}_{k,i,M}))$
12:     Update the policy network with the S-PPO loss
        $\mathcal{L}(\theta) = -\frac{1}{\Sigma_k |\tau_k|} \Sigma_{\tau_k} \Sigma_i \min(\frac{\tilde{\pi}(a_{k,i}|s_{k,i};\theta)}{\tilde{\pi}(a_{k,i}|s_{k,i};\theta_{\text{old}})} \hat{A}_{k,i}, \text{clip}(\frac{\tilde{\pi}(a_{k,i}|s_{k,i};\theta)}{\tilde{\pi}(a_{k,i}|s_{k,i};\theta_{\text{old}})}, 1 - \epsilon_{\text{clip}}, 1 + \epsilon_{\text{clip}}) \hat{A}_{k,i})$,
        where $\hat{A}_{k,i}$ is the advantage
13: end for
```

Algorithm 5 CollectTrajectories function

```
1:  Input: number of trajectories $K$, smoothing variance $\sigma$, number of samples $M$, Policy network $\pi$
2:  for $k = 1$ to $K$ do
3:      while not end game do
4:          Get state $s$ from the environment
5:          for $m = 1$ to $M$ do
6:              Sample a noise from the normal distribution and add to the state $\tilde{s}_m = s_m + \mathcal{N}(0, \sigma^2 I_N)$
7:              Store the mean and standard deviation of the action $(a^{\text{mean}}_m, a^{\text{std}}_m)$ to the list, where $\mathcal{N}(a^{\text{mean}}_m, a^{\text{std}}_m) = \pi(a|\tilde{s}_m)$
8:          end for
9:          Take the median and obtain the smoothed policy $\tilde{\pi}(a|s) = \mathcal{N}(\text{median}(a^{\text{mean}}_1, ..., a^{\text{mean}}_M), \text{median}(a^{\text{std}}_1, ..., a^{\text{std}}_M))$
10:         Take action with the smoothed policy and collect the reward
11:     end while
12:     Store the trajectory $\tau_k$
13: end for
14: Return the set of the trajectories $\{\tau_k\}$
```

A.3. Detailed settings for DQN and PPO

A.3.1. SETTINGS FOR DQN

Our DQN implementation is based on the SADQN (Zhang et al., 2020) and CROP (Wu et al., 2022). We use the DnCNN structure proposed in Zhang et al. (2017) as the denoiser to train S-DQN. We train our S-DQN for $300,000$ frames in Pong, Freeway, and RoadRunner. The training time of S-DQN is roughly 12 hours on our hardware, which is much faster than 40 hours of SADQN and 17 hours of RadialDQN. For WocaRDQN, the training is initialized with RadialDQN as we found the training is unstable. The smoothing variance σ for S-DQN is set to 0.1 in Pong, 0.1 in Freeway, and 0.05 in RoadRunner. All the experiment results under attack are obtained by taking the average of 5 episodes.

A.3.2. SETTINGS FOR PPO

Our PPO implementation is based on the SAPPO (Zhang et al., 2020), RadialPPO (Oikarinen et al., 2021), ATLAPPO (Zhang et al., 2021), and PA-ATLAPPO (Sun et al., 2022). We train S-PPO for 2000000 steps in Walker and Hopper. We use a simple MLP network for all the PPO algorithms. For the PA-ATLAPPO, we do not combine with SGLD unlike the original paper, as we want to evaluate the true robustness of PA-ATLA algorithm. Note that there is high a variance between the performance of each agent trained with the same algorithm. To get a fair and comparable result, we trained each agent 15 times and reported the median of the performance as suggested in Zhang et al. (2020). The median agent is selected by considering the median of clean reward, reward under MAD attack, and reward under Min-RS attack from a pool of 15 agents. Subsequently, we conduct further evaluations on the median agents under the Optimal Attack and the PA-AD attack since these evaluations involve high computational costs and are impractical to perform on the entire set of 15 agents. The smoothing variance σ for S-PPO is set to 0.2 in all environments. The ℓ_∞ attack budget for all the attacks for PPO (MAD, Min-RS, Optimal Attack, PA-AD attack) is set to 0.075. All the experiment results under attack are obtained by taking the average of 50 episodes.

A.4. Details of estimating bounds

A.4.1. Estimating the certified radius for S-DQN

In practice, we use Monte Carlo sampling to estimate $\widetilde{Q}$, which denotes as $\widetilde{Q}_{\text{est}}$. The estimation of the Certified Radius is formulated as follows:

$$R_{\text{est},t} = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}_{\text{est}}(s_t, a_1) - \Delta) - \Phi^{-1}(\widetilde{Q}_{\text{est}}(s_t, a_2) + \Delta)), \tag{17}$$

where $\widetilde{Q}_{\text{est}}(s, a) = \frac{1}{m}\Sigma_{i=1}^{m} Q_h(D(s + \delta_i), a)$, $\delta_i \sim \mathcal{N}(0, \sigma^2 I_N), \forall i \in \{1, ..., m\}$, $\Delta = \sqrt{\frac{1}{2m}\ln\frac{1}{\alpha}}$, m is the number of the samples ($m = 100$ in our setting), and α is the one-side confidence parameter ($\alpha = 0.05$ in our setting). The proof of this estimation can be found in Appendix A.5.

A.4.2. Estimating the action bound for S-PPO

In practice, we use Monte Carlo sampling to estimate $\tilde{\pi}_{\text{det},p}$, which denotes as $\tilde{\pi}_{\text{det},p_{\text{est}}}$. The estimation of the Action Bound is formulated as follows:

$$\tilde{\pi}_{\text{det},\underline{p_{\text{est}}}}(s_t) \preceq \tilde{\pi}_{\text{det},p_{\text{est}}}(s_t + \Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p_{\text{est}}}}(s_t),\ s.t\ ||\Delta s||_2 \le \epsilon, \tag{18}$$

where $\tilde{\pi}_{i,\text{det},p_{\text{est}}}(s) = max\{a_i \in \mathbb{R}|\ \ |\{x \in S_i | x \le a_i\}| \le \lceil m p_{\text{est}} \rceil\}, S_i = \{\pi_{i,\text{det}}(s + \delta_1), ..., \pi_{i,\text{det}}(s + \delta_m)\}, \forall i \in \{1, ..., N_{\text{action}}\}$, $\delta_j \sim \mathcal{N}(0, \sigma^2 I_N), \forall j \in \{1, ..., m\}$, $\underline{p_{\text{est}}} = \Phi(\Phi^{-1}(p_{\text{est}} - \Delta) - \frac{\epsilon}{\sigma})$, $\overline{p_{\text{est}}} = \Phi(\Phi^{-1}(p_{\text{est}} + \Delta) + \frac{\epsilon}{\sigma})$, $\Delta = \sqrt{\frac{1}{2m}\ln\frac{1}{\alpha}}$, m is the number of the samples ($m = 100$ in our setting), and α is the one-side confidence parameter ($\alpha = 0.05$ in our setting). The proof of this estimation can be found in Appendix A.6.

A.4.3. Estimating the reward lower bound for smoothed agents

In practice, we use Monte Carlo sampling to estimate $\widetilde{F}_{\pi,p}$, which denotes as $\widetilde{F}_{\pi,p_{\text{est}}}$. The estimation of the Reward Lower Bound is formulated as follows:

$$\widetilde{F}_{\pi,p_{\text{est}}}(\boldsymbol{\Delta s}) \ge \widetilde{F}_{\pi,\underline{p_{\text{est}}}}(\boldsymbol{0}),\ \text{s.t.}\ ||\boldsymbol{\Delta s}||_2 \le B, \tag{19}$$

where $\widetilde{F}_{\pi,p_{\text{est}}}(\boldsymbol{\Delta s}) = max\{r \in \mathbb{R} || \{x \in S | x \le r\}| \le \lceil m_\tau p_{\text{est}} \rceil\}, S = \{F_\pi(\boldsymbol{\delta}_1 + \boldsymbol{\Delta s}), ..., F_\pi(\boldsymbol{\delta}_{m_\tau} + \boldsymbol{\Delta s})\}$, $\boldsymbol{\delta}_i \sim \mathcal{N}(0, \sigma^2 I_{H\times N}), \forall i \in \{1, ..., m_\tau\}$, $\underline{p_{\text{est}}} = \Phi(\Phi^{-1}(p_{\text{est}} - \Delta) - \frac{B}{\sigma})$, $\Delta = \sqrt{\frac{1}{2m_\tau}\ln\frac{1}{\alpha}}$, m_τ is the number of sample trajectories ($m_\tau = 1000$ in our setting), and α is the one-side confidence parameter ($\alpha = 0.05$ in our setting). Note that in this setting, each state is added with a noise. Therefore, $m = 1$. The proof of this estimation can be found in Appendix A.7.

A.5. Proof of the certified radius for S-DQN

In this section, we give the formal proof of the certified radius introduced in Section 4. Our proof is based on the proof proposed by Salman et al. (2019) in Appendix A. Recall that we have:

$$R_t = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \Phi^{-1}(\widetilde{Q}(s_t, a_2))), \tag{20}$$

where a_1 is the action with the largest Q-value among all the other actions, a_2 is the "runner-up" action, R_t is the certified radius at time t, Φ is the CDF of normal distribution, σ is the smoothing variance, and $\widetilde{Q}(s, a)$ is defined in Eq.(6).

We first go over the lemma needed for proof.

Lemma 1 For the function $Q_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the function $\widetilde{Q}$ is $\frac{1}{\sigma}\sqrt{\frac{2}{\pi}}$-Lipschitz.

Proof. From the definition of $\widetilde{Q}$, we have

$$\widetilde{Q}(s, a) = (Q_h * \mathcal{N}(0, \sigma^2 I_n))(D(s), a) = \frac{1}{(2\pi)^{n/2}\sigma^n} \int_{\mathbb{R}_n} Q_h(D(t), a) \exp\Big(-\frac{1}{2\sigma^2}||s - t||_2^2\Big) dt. \tag{21}$$

Take the gradient w.r.t. s, we have

$$\nabla_s \widetilde{Q}(s, a) = \frac{1}{(2\pi)^{n/2}\sigma^n} \int_{\mathbb{R}_n} \frac{1}{\sigma^2}(s - t) Q_h(D(t), a) \exp\Big(-\frac{1}{2\sigma^2}||s - t||_2^2\Big) dt. \tag{22}$$

For any unit direction u, we have

$$\begin{aligned}
u \cdot \nabla_s \widetilde{Q}(s, a) &\leq \frac{1}{(2\pi)^{n/2}\sigma^n} \int_{\mathbb{R}_n} \frac{1}{\sigma^2} |u \cdot (s - t)| \exp\Big(-\frac{1}{2\sigma^2}||s - t||_2^2\Big) dt \\
&= \frac{1}{\sigma^2} \int_{\mathbb{R}_n} \frac{1}{\sqrt{2\pi}\sigma} |u \cdot (s - t)| \exp\Big(-\frac{1}{2\sigma^2}||s - t||_2^2\Big) dt \\
&= \frac{1}{\sigma^2} \int_{-\infty}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} |z| \exp\Big(-\frac{1}{2\sigma^2} z^2\Big) dz \\
&= \frac{1}{\sigma^2} \mathbb{E}_{z\sim\mathcal{N}(0,\sigma^2)}[|z|] \\
&= \frac{1}{\sigma}\sqrt{\frac{2}{\pi}}.
\end{aligned} \tag{23}$$

In fact, there is a stronger smoothness property for $\widetilde{Q}$.

Lemma 2 For the function $Q_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$, the mapping $s \mapsto \sigma\Phi^{-1}(\widetilde{Q}(s, a))$ is 1-Lipschitz.

Proof. Take the gradient of $\Phi^{-1}(\widetilde{Q}(s, a))$ w.r.t. s, we have

$$\nabla \Phi^{-1}(\widetilde{Q}(s, a)) = \frac{\nabla \widetilde{Q}(s, a)}{\Phi'(\Phi^{-1}(\widetilde{Q}(s, a)))}. \tag{24}$$

We intend to show that for any unit direction u,

$$\begin{gathered}
u \cdot \sigma \nabla \Phi^{-1}(\widetilde{Q}(s, a)) \leq 1 \\
u \cdot \sigma \nabla \widetilde{Q}(s, a) \leq \Phi'(\Phi^{-1}(\widetilde{Q}(s, a))) \\
u \cdot \sigma \nabla \widetilde{Q}(s, a) \leq \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{1}{2}(\Phi^{-1}(\widetilde{Q}(s, a)))^2\Big).
\end{gathered} \tag{25}$$

The left-hand side can be written as

$$\frac{1}{\sigma}\mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I_n)}[Q_h(D(s + \delta), a)\delta \cdot u]. \tag{26}$$

We claim that the supremum of the above quantity over all functions $Q_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$, subject to $\mathbb{E}[Q_h(D(s+\delta), a)] = \widetilde{Q}(s, a)$, is equal to

$$\frac{1}{\sigma}\mathbb{E}[(\delta \cdot u)\mathbb{1}\{\delta \cdot u \geq -\sigma\Phi^{-1}(\widetilde{Q}(s, a))\}] = \frac{1}{\sqrt{2\pi}}\exp\Big(-\frac{1}{2}(\Phi^{-1}(\widetilde{Q}(s, a)))^2\Big). \tag{27}$$

To prove the claim is true, note that $h : \delta \mapsto \mathbb{1}\{\delta \cdot u \geq -\sigma\Phi^{-1}(\widetilde{Q}(s, a))\}$ achieves equality. Assume by contradiction that the maximum is reached by some function $f : \delta \to [0, 1]$. Consider the set $\Omega^+ = \{\delta | h(\delta) > f(\delta)\}$ and the set $\Omega^- = \{\delta | h(\delta) < f(\delta)\}$. Now construct the new function $f' = f + (h - f)\mathbb{1}\{\Omega^+\} - (f - h)\mathbb{1}\{\Omega^-\}$, which takes value in $[0, 1]$. Since both h and f integrate to $\widetilde{Q}(s, a)$, we have $\int_{\Omega^+}(h - f)d\delta = \int_{\Omega^-}(f - h)d\delta$. This gives that f' also integrates to $\widetilde{Q}(s, a)$. By the definition of h, for any $\delta_1 \in \Omega^+$ and $\delta_2 \in \Omega^-$, we have $\delta_1 \cdot u > \delta_2 \cdot u$, and since $\int_{\Omega^+}(h - f)d\delta = \int_{\Omega^-}(f - h)d\delta$, we have

$$\begin{gathered}
\int_{\Omega^+}(\delta \cdot u)(h - f)(\delta)d\delta > \int_{\Omega^-}(\delta \cdot u)(f - h)(\delta)d\delta \\
\int(\delta \cdot u)f(\delta)d\delta < \int(\delta \cdot u)f(\delta)d\delta + \int_{\Omega^+}(\delta \cdot u)(h - f)(\delta)d\delta - \int_{\Omega^-}(\delta \cdot u)(f - h)(\delta)d\delta \\
\int(\delta \cdot u)f(\delta)d\delta < \int(\delta \cdot u)f'(\delta)d\delta
\end{gathered} \tag{28}$$

Hence, the maximum is obtained at h. The claim holds, and hence, we have

$$u \cdot \sigma\nabla\Phi^{-1}(\widetilde{Q}(s, a)) \leq 1. \tag{29}$$

Now, we can prove the certified radius in Eq.(20).

Theorem 1 Let $Q_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and $\widetilde{Q}(s, a) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}Q_h(D(s + \delta), a)$. At time step t with state s_t, the certified radius is

$$R_t = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \Phi^{-1}(\widetilde{Q}(s_t, a_2))), \tag{30}$$

where a_1 is the action with the largest Q-value among all the other actions, a_2 is the ”runner-up” action, R_t is the certified radius at time t, Φ is the CDF of normal distribution, and σ is the smoothing variance. The certified radius gives a lower bound on the minimum ℓ_2 adversarial perturbation required to change the policy from a_1 to a_2.

Proof. Let the perturbation be Δs and able to change the action from a_1 to a_2. By lemma 2, we have

$$\sigma\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \sigma\Phi^{-1}(\widetilde{Q}(s_t + \Delta s, a_1)) \leq ||\Delta s||_2 \tag{31}$$

Since the perturbation can change the action, we have $\widetilde{Q}(s_t + \Delta s, a_1) \leq \widetilde{Q}(s_t + \Delta s, a_2)$, which leads to

$$\sigma\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \sigma\Phi^{-1}(\widetilde{Q}(s_t + \Delta s, a_2)) \leq ||\Delta s||_2 \tag{32}$$

By lemma 2 and $\widetilde{Q}(s_t + \Delta s, a_2) \geq \widetilde{Q}(s_t, a_2)$, we have

$$\sigma\Phi^{-1}(\widetilde{Q}(s_t + \Delta s, a_2)) - \sigma\Phi^{-1}(\widetilde{Q}(s_t, a_2)) \leq ||\Delta s||_2 \tag{33}$$

Combine Eq.(32) and Eq.(33), we have

$$||\Delta s||_2 \geq \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \Phi^{-1}(\widetilde{Q}(s_t, a_2))), \tag{34}$$

which gives us the certified radius

$$R_t = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}(s_t, a_1)) - \Phi^{-1}(\widetilde{Q}(s_t, a_2))). \tag{35}$$

Now, we prove the practical version of the certified radius introduced in Appendix A.4.1:

Theorem 2 Let $Q_h : \mathcal{S} \times \mathcal{A} \to [0, 1]$, and $\widetilde{Q}_{\text{est}}(s, a) = \frac{1}{m}\Sigma_{i=1}^{m} Q_h(D(s + \delta_i), a), \delta_i \sim \mathcal{N}(0, \sigma^2 I_N), \forall i \in \{1, ..., m\}$. At time step t with state s_t, the certified radius is

$$R_{\text{est},t} = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}_{\text{est}}(s_t, a_1) - \Delta) - \Phi^{-1}(\widetilde{Q}_{\text{est}}(s_t, a_2) + \Delta)), \tag{36}$$

where $\Delta = \sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}}$, m is the number of the samples, α is the one-side confidence parameter, a_1 is the action with the largest Q-value among all the other actions, a_2 is the "runner-up" action, R_t is the certified radius at time t, Φ is the CDF of normal distribution, and σ is the smoothing variance.

Proof. By *Hoeffding's Inequality*, for any $t \geq 0$, we have

$$P(\widetilde{Q}_{\text{est}} - \widetilde{Q} \geq t) \leq \exp^{-2mt^2}. \tag{37}$$

Rearrange the inequality

$$P(\widetilde{Q}_{\text{est}} - \widetilde{Q} \geq \sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}}) \leq \alpha. \tag{38}$$

Hence, a $1 - \alpha$ confidence lower bound $\underline{\widetilde{Q}}$ of $\widetilde{Q}$ is

$$\underline{\widetilde{Q}} = \widetilde{Q}_{\text{est}} - \sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}} = \widetilde{Q}_{\text{est}} - \Delta. \tag{39}$$

Similarly, we have $1 - \alpha$ confidence upper bound $\overline{\widetilde{Q}}$ of $\widetilde{Q}$

$$\overline{\widetilde{Q}} = \widetilde{Q}_{\text{est}} + \Delta. \tag{40}$$

Substitute $\widetilde{Q}(s_t, a_1)$ with the lower bound and $\widetilde{Q}(s_t, a_2)$ with the upper bound, we have

$$R_{\text{est},t} = \frac{\sigma}{2}(\Phi^{-1}(\widetilde{Q}_{\text{est}}(s_t, a_1) - \Delta) - \Phi^{-1}(\widetilde{Q}_{\text{est}}(s_t, a_2) + \Delta)) \tag{41}$$

A.6. Proof of the action bound for S-PPO

In this section, we give the formal proof of the action bound introduced in Section 4. Our proof is based on the proof proposed by Chiang et al. (2020) in Appendix B. Recall that we have:

$$\tilde{\pi}_{\text{det},\underline{p}}(s_t) \preceq \tilde{\pi}_{\text{det},p}(s_t+\Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p}}(s_t),\ s.t\ ||\Delta s||_2 \leq \epsilon, \tag{42}$$

where $\tilde{\pi}_{i,\text{det},p}(s) = \sup\{a_i \in \mathbb{R} | \mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[\pi_{i,\text{det}}(s+\delta) \leq a_i] \leq p\}, \forall i \in \{1,...,N_{\text{action}}\}$, $\underline{p} = \Phi(\Phi^{-1}(p) - \frac{\epsilon}{\sigma})$, $\overline{p} = \Phi(\Phi^{-1}(p) + \frac{\epsilon}{\sigma})$, Φ is the CDF of normal distribution, and σ is the smoothing variance.

Theorem 3 Let $\pi : \mathcal{S} \rightarrow \mathcal{A}$ be the policy network, and $\tilde{\pi}_{i,\text{det},p}(s) = \sup\{a_i \in \mathbb{R} | \mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I)}[\pi_{i,\text{det}}(s+\delta) \leq a_i] \leq p\}, \forall i \in \{1,...,N_{\text{action}}\}$. At time step t with state s_t, the action bound is

$$\tilde{\pi}_{\text{det},\underline{p}}(s_t) \preceq \tilde{\pi}_{\text{det},p}(s_t+\Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p}}(s_t),\ s.t\ ||\Delta s||_2 \leq \epsilon, \tag{43}$$

where $\underline{p} = \Phi(\Phi^{-1}(p) - \frac{\epsilon}{\sigma})$, $\overline{p} = \Phi(\Phi^{-1}(p) + \frac{\epsilon}{\sigma})$, Φ is the CDF of a normal distribution, and σ is the smoothing variance.

Proof. Let $\mathcal{E}_i(s_t) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\mathbb{1}\{\pi_{i,\text{det}}(s_t+\delta) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)\}]$, and we have $\mathcal{E}_i : \mathbb{R}^N \rightarrow [0,1], \forall i \in \{1,...,N_{\text{action}}\}$. The mapping $s_t \mapsto \sigma\Phi^{-1}(\mathcal{E}_i(s_t))$ is 1-Lipschitz, which can be proved by the similar technique used in Lemma 2. Since $\mathcal{E}_i(s_t) = \mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)]$, given the perturbation Δs, we have

$$\begin{aligned}&\sigma\Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta+\Delta s) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)])- \\ &\sigma\Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)]) \leq ||\Delta s||_2.\end{aligned} \tag{44}$$

Rearrange the inequality, we have

$$\begin{aligned}&\Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta+\Delta s) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)]) \\ &\leq \Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)]) + \frac{||\Delta s||_2}{\sigma} \\ &\leq \Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)]) + \frac{\epsilon}{\sigma} \\ &= \Phi^{-1}(\underline{p}) + \frac{\epsilon}{\sigma} \\ &= \Phi^{-1}(p).\end{aligned} \tag{45}$$

By the monotonicity of Φ, we have

$$\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta+\Delta s) \leq \tilde{\pi}_{i,\text{det},\underline{p}}(s_t)] \leq p. \tag{46}$$

Recall that $\tilde{\pi}_{i,\text{det},p}(s_t+\Delta s) = \sup\{a_i \in \mathbb{R} | \mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_N)}[\pi_{i,\text{det}}(s_t+\delta+\Delta s) \leq a_i] \leq p\}, \forall i \in \{1,...,N_{\text{action}}\}$, we have

$$\tilde{\pi}_{\text{det},\underline{p}}(s_t) \preceq \tilde{\pi}_{\text{det},p}(s_t+\Delta s). \tag{47}$$

We can show that $\tilde{\pi}_{\text{det},p}(s_t+\Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p}}(s_t)$ for all $||\Delta s||_2 \leq \epsilon$ with the similar technique. Combine the two bounds we have

$$\tilde{\pi}_{\text{det},\underline{p}}(s_t) \preceq \tilde{\pi}_{\text{det},p}(s_t+\Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p}}(s_t). \tag{48}$$

Now, we prove the practical version of the action bound introduced in Appendix A.4.2:

Theorem 4 Let $\pi : \mathcal{S} \rightarrow \mathcal{A}$ be the policy network, and $\tilde{\pi}_{i,\text{det},p_{\text{est}}}(s) = max\{a_i \in \mathbb{R}|\ |\{x \in S_i | x \leq a_i\}| \leq \lceil mp_{\text{est}} \rceil\}$, $S_i = \{\pi_{i,\text{det}}(s+\delta_1), ..., \pi_{i,\text{det}}(s+\delta_m)\}, \forall i \in \{1,...,N_{\text{action}}\}$, $\delta_j \sim \mathcal{N}(0,\sigma^2 I_N), \forall j = 1,...,m$. At time step t with state s_t, the action bound is

$$\tilde{\pi}_{\text{det},\underline{p_{\text{est}}}}(s_t) \preceq \tilde{\pi}_{\text{det},p_{\text{est}}}(s_t+\Delta s) \preceq \tilde{\pi}_{\text{det},\overline{p_{\text{est}}}}(s_t),\ s.t\ ||\Delta s||_2 \leq \epsilon, \tag{49}$$

where $\underline{p_{\text{est}}} = \Phi(\Phi^{-1}(p_{\text{est}} - \Delta) - \frac{\epsilon}{\sigma})$, $\overline{p_{\text{est}}} = \Phi(\Phi^{-1}(p_{\text{est}} + \Delta) + \frac{\epsilon}{\sigma})$, $\Delta = \sqrt{\frac{1}{2m}\ln\frac{1}{\alpha}}$, m is the number of the samples, α is the one-side confidence parameter, Φ is the CDF of normal distribution, and σ is the smoothing variance.

Proof. By *Hoeffding's Inequality*, for any $t \geq 0$, we have

$$P(p_{\text{est}} - p \geq t) \leq \exp^{-2mt^2}. \tag{50}$$

Rearrange the inequality

$$P(p_{\text{est}} - p \geq \sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}}) \leq \alpha. \tag{51}$$

Hence, a $1-\alpha$ confidence lower bound $\underline{p}$ of p is

$$\underline{p} = p_{\text{est}} - \sqrt{\frac{1}{2m} \ln \frac{1}{\alpha}} = p_{\text{est}} - \Delta. \tag{52}$$

Similarly, we have $1-\alpha$ confidence upper bound $\overline{p}$ of $\underline{p}$

$$\overline{p} = p_{\text{est}} + \Delta. \tag{53}$$

Substitute $\Phi(\Phi^{-1}(p) - \frac{\epsilon}{\sigma})$ with the lower bound, and $\Phi(\Phi^{-1}(p) + \frac{\epsilon}{\sigma})$ with the upper bound, we have

$$\big[\Phi(\Phi^{-1}(p_{\text{est}} - \Delta) - \frac{\epsilon}{\sigma}),\ \ \Phi(\Phi^{-1}(p_{\text{est}} + \Delta) + \frac{\epsilon}{\sigma}\big], \tag{54}$$

which is the new upper bound and lower bound in the expression.

A.7. Proof of the reward lower bound for smoothed agents

In this section, we give the formal proof of the reward lower bound introduced in Section 4. Our proof is based on the proof proposed by Chiang et al. (2020) in Appendix B. Recall that we have:

$$\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) \geq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0}), \text{ s.t. } ||\boldsymbol{\Delta s}||_2 \leq B, \tag{55}$$

where $\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) = \sup\{r \in \mathbb{R} | \mathbb{P}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq r] \leq p\}$, $\underline{p} = \Phi(\Phi^{-1}(p) - \frac{B}{\sigma})$, and B is the ℓ_2 attack budget of the entire trajectory.

Theorem 5 Let $F_\pi : \mathbb{R}^{H\times N} \to \mathbb{R}$ be the function mapping the perturbation to the total reward, and $\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) = \sup\{r \in \mathbb{R} | \mathbb{P}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq r] \leq p\}$. The reward lower bound is

$$\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) \geq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0}), \text{ s.t. } ||\boldsymbol{\Delta s}||_2 \leq B, \tag{56}$$

where $\underline{p} = \Phi(\Phi^{-1}(p) - \frac{B}{\sigma})$, B is the ℓ_2 attack budget of the entire trajectory, Φ is the CDF of normal distribution, and σ is the smoothing variance.

Proof. Let $\mathcal{E}(\boldsymbol{\Delta s}) = \mathbb{E}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[\mathbb{1}\{F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})\}]$, and we have $\mathcal{E} : \mathbb{R}^{H\times N} \to [0,1]$. The mapping $\boldsymbol{\Delta s} \mapsto \sigma\Phi^{-1}(\mathcal{E}(\boldsymbol{\Delta s}))$ is 1-Lipschitz by Lemma 2. Since $\mathcal{E}(\boldsymbol{\Delta s}) = \mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})]$, given the perturbation $\boldsymbol{\Delta s}$, we have

$$\begin{aligned}&\sigma\Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})]) - \sigma\Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})])\\ &\leq ||\boldsymbol{\Delta s}||_2.\end{aligned} \tag{57}$$

Rearrange the inequality, we have

$$\begin{aligned}&\Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})])\\ &\leq \Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})]) + \frac{||\boldsymbol{\Delta s}||_2}{\sigma}\\ &\leq \Phi^{-1}(\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})]) + \frac{B}{\sigma}\\ &= \Phi^{-1}(\underline{p}) + \frac{B}{\sigma}\\ &= \Phi^{-1}(p).\end{aligned} \tag{58}$$

By the monotonicity of Φ, we have

$$\mathbb{P}_{\delta\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0})] \leq p. \tag{59}$$

Recall that $\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) = \sup\{r \in \mathbb{R} | \mathbb{P}_{\boldsymbol{\delta}\sim\mathcal{N}(0,\sigma^2 I_{H\times N})}[F_\pi(\boldsymbol{\delta}+\boldsymbol{\Delta s}) \leq r] \leq p\}$, we have

$$\widetilde{F}_{\pi,p}(\boldsymbol{\Delta s}) \geq \widetilde{F}_{\pi,\underline{p}}(\mathbf{0}). \tag{60}$$

Now, we prove the practical version of the reward lower bound introduced in Appendix A.4.3:

Theorem 6 Let $F_\pi : \mathbb{R}^{H\times N} \to \mathbb{R}$ be the function mapping the perturbation to the total reward, and $\widetilde{F}_{\pi,p_{\text{est}}}(\boldsymbol{\Delta s}) = max\{r \in \mathbb{R} | |\{x \in S | x \leq r\}| \leq \lceil m_\tau p_{\text{est}} \rceil\}, S = \{F_\pi(\boldsymbol{\delta}_1 + \boldsymbol{\Delta s}), ..., F_\pi(\boldsymbol{\delta}_{m_\tau} + \boldsymbol{\Delta s})\}$, $\boldsymbol{\delta}_i \sim \mathcal{N}(0, \sigma^2 I_{H\times N}), \forall i = \{1, ..., m_\tau\}$. The reward lower bound is

$$\widetilde{F}_{\pi,p_{\text{est}}}(\boldsymbol{\Delta s}) \geq \widetilde{F}_{\pi,\underline{p_{\text{est}}}}(\mathbf{0}), \text{ s.t. } ||\boldsymbol{\Delta s}||_2 \leq B, \tag{61}$$

where $\underline{p_{\text{est}}} = \Phi(\Phi^{-1}(p_{\text{est}} - \Delta) - \frac{B}{\sigma})$, $\Delta = \sqrt{\frac{1}{2m_\tau} \ln \frac{1}{\alpha}}$, m_τ is the number of sample trajectories, α is the one-side confidence parameter, Φ is the CDF of normal distribution, and σ is the smoothing variance.

Proof. By *Hoeffding's Inequality*, for any $t \geq 0$, we have

$$P(p_{\text{est}} - p \geq t) \leq \exp^{-2m_\tau t^2}. \tag{62}$$

Rearrange the inequality

$$P(p_{\text{est}} - p \geq \sqrt{\frac{1}{2m_\tau} \ln \frac{1}{\alpha}}) \leq \alpha. \tag{63}$$

Hence, a $1 - \alpha$ confidence lower bound $\underline{p}$ of p is

$$\underline{p} = p_{\text{est}} - \sqrt{\frac{1}{2m_\tau} \ln \frac{1}{\alpha}} = p_{\text{est}} - \Delta. \tag{64}$$

Substitute $\Phi(\Phi^{-1}(p) - \frac{B}{\sigma})$ with the lower bound, we have

$$\Phi(\Phi^{-1}(p_{\text{est}} - \Delta) - \frac{B}{\sigma}), \tag{65}$$

which is the new lower bound in the expression.

A.8. The certified radius of smoothed DQN agents

Table 5 presents the Certified Radius of our S-DQNs and CROP's agents. Our S-DQN agents generally achieve higher Certified Radius. It's important to note that while the CROP framework used a sample number of $m = 10000$ for estimating the Certified Radius, we used $m = 100$ here. Although a larger m can enhance confidence in estimating and result in a larger Certified Radius, $m = 10000$ is not a practical setting. Our hard randomized smoothing strategy demonstrates the capability to provide a large Certified Radius even with a small m.

Table 5. The Certified Radius of different smoothed DQN agents.

Methods	Certified Radius (larger is better)		
	Pong	Freeway	RoadRunner
Ours (using hard randomized smoothing):			
S-DQN (Radial)	$\mathbf{0.1044}$	$\mathbf{0.1134}$	$\mathbf{0.0576}$
S-DQN (S-PGD)	0.0502	0.0766	0.0520
S-DQN (Vanilla)	0.0619	0.0774	0.0502
CROP (Wu et al., 2022) (using mean smoothing):			
RadialDQN+RS	0.0000	0.0000	0.0000
SADQN+RS	0.0615	0.0665	0.0000
VanillaDQN+RS	0.0000	0.0000	0.0000

A.9. The Action Divergence of smoothed PPO agents

We designed a metric based on the action bound in Section 4 to evaluate the certified robustness of the smoothed PPO agents. We define the **Action Divergence** as follows:

$$\text{ADIV} = \mathbb{E}_{s,\epsilon}\Big[\frac{||\tilde{\pi}_{\text{det},\overline{p_{\text{est}}}}(s) - \tilde{\pi}_{\text{det},\underline{p_{\text{est}}}}(s)||_2}{2\epsilon}\Big], \tag{66}$$

where ϵ is the ℓ_2 attack budget used in estimating the action bound, and the definition of $\overline{p_{\text{est}}}$ and $\underline{p_{\text{est}}}$ is in Appendix A.4.2. We found that the ℓ_2 norm of the difference between the upper and lower bound of the actions is proportional to the magnitude of the ℓ_2 budget ϵ, which makes $\frac{||\tilde{\pi}_{\text{det},\overline{p_{\text{est}}}}(s) - \tilde{\pi}_{\text{det},\underline{p_{\text{est}}}}(s)||_2}{2\epsilon}$ almost unchanged under different ϵ setting. Hence, we take the expectation over the state s and the budget ϵ to estimate this fraction, which is the ADIV proposed here. We estimate the ADIV by taking the average of 50 trajectories with three different ϵ settings ($\epsilon = 0.1$, $\epsilon = 0.2$, and $\epsilon = 0.3$).

ADIV describes the worst-case stability of the actions of a smoothed PPO agent under any ℓ_2 perturbation. The more this value is, the more unstable the smoothed agent is under the ℓ_2 attack. The result is shown in Table 6. Our S-PPO agents exhibit lower ADIV compared to their naively smoothed counterparts. Notably, S-PPO (SGLD) and S-PPO (WocaR) have the lowest ADIV, and they also demonstrate higher robustness under attacks compared to the others in our study.

Table 6. The Action Divergence of different smoothed PPO agents.

Methods	Action Divergence (lower is better)	
	Walker	Hopper
Ours:		
S-PPO (SGLD)	1.401	**0.656**
S-PPO (Radial)	8.665	2.305
S-PPO (WocaR)	**1.125**	0.778
S-PPO (S-ATLA)	4.218	8.964
S-PPO (S-PA-ATLA)	3.899	8.432
S-PPO (Vanilla)	2.926	1.618
Previous smoothed agents:		
SGLDPPO+RS	2.221	1.375
RadialPPO+RS	7.964	2.766
WocaRPPO+RS	2.431	1.466
ATLAPPO+RS	6.062	16.994
PA-ATLAPPO+RS	5.595	11.165
VanillaPPO+RS	5.030	4.187

A.10. Detailed experiment results of robust reward for S-DQN

Table 7 shows the reward of DQN agents under ℓ_∞ PGD attack and PA-AD attack. Note that we used our stronger S-PGD attack and S-PA-AD to evaluate all the smoothed agents. Our S-DQN (Vanilla) already outperformed the state-of-the-art robust agent, RadialDQN, in most of the settings except for RoadRunner. This problem was solved by introducing S-DQN (Radial) and S-DQN (S-PGD). S-DQN (Radial) performs especially well under all attacks across various environments, which suggests that our S-DQN can be further boosted by changing the base model to a robust agent.

Table 7. The reward of DQN agents under ℓ_∞ PGD attack and PA-AD attack. The smoothing variance σ for the smoothed agents is set to 0.1 in Pong, 0.1 in Freeway, and $\sigma = 0.05$ in RoadRunner.

Pong	Clean reward	PGD or S-PGD					PAAD or S-PAAD
$\epsilon(\ell_\infty)$		0.01	0.02	0.03	0.04	0.05	0.05
Ours:							
S-DQN (Radial)	$\mathbf{21.0{\pm}0.0}$	$\mathbf{21.0{\pm}0.0}$	$\mathbf{21.0{\pm}0.0}$	$\mathbf{21.0{\pm}0.0}$	$\mathbf{21.0{\pm}0.0}$	$\mathbf{20.8{\pm}0.4}$	$14.0{\pm}2.1$
S-DQN (S-PGD)	$20.6{\pm}0.5$	$20.8{\pm}0.4$	$20.0{\pm}1.1$	$15.6{\pm}4.3$	$13.8{\pm}4.8$	$1.6{\pm}4.2$	$11.0{\pm}2.6$
S-DQN (Vanilla)	$20.4{\pm}0.5$	$\mathbf{21.0{\pm}0.0}$	$20.4{\pm}0.8$	$20.2{\pm}0.8$	$16.6{\pm}4.4$	$18.4{\pm}2.1$	$\mathbf{18.6{\pm}1.2}$
SOTA robust agents:							
RadialDQN	$\mathbf{21.0{\pm}0.0}$	$\mathbf{21.0{\pm}0.0}$	$20.0{\pm}2.0$	$-20.2{\pm}0.4$	$-20.6{\pm}0.5$	$-21.0{\pm}0.00$	$-21.0{\pm}0.00$
SADQN	$\mathbf{21.0{\pm}0.0}$	$\mathbf{21.0{\pm}0.0}$	$-19.4{\pm}0.8$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$
WocaRDQN	$20.0{\pm}0.9$	$19.6{\pm}1.4$	$-20.4{\pm}0.8$	$-20.8{\pm}0.4$	$-21.0{\pm}0.00$	$-21.0{\pm}0.00$	$-21.0{\pm}0.00$
VanillaDQN	$\mathbf{21.0{\pm}0.0}$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$
Previous smoothed agents:							
RadialDQN+RS	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$
SADQN+RS	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$
WocaRDQN+RS	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$
VanillaDQN+RS	$-20.8{\pm}0.4$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$	$-21.0{\pm}0.0$
Freeway							
Ours:							
S-DQN (Radial)	$33.0{\pm}0.0$	$\mathbf{33.0{\pm}0.0}$	$\mathbf{32.6{\pm}0.5}$	$\mathbf{32.6{\pm}0.5}$	$\mathbf{31.6{\pm}0.5}$	$\mathbf{32.0{\pm}1.1}$	$\mathbf{32.0{\pm}1.1}$
S-DQN (S-PGD)	$32.6{\pm}1.4$	$32.6{\pm}1.0$	$32.0{\pm}1.3$	$30.2{\pm}0.8$	$28.2{\pm}1.5$	$25.6{\pm}0.5$	$30.4{\pm}1.0$
S-DQN (Vanilla)	$\mathbf{34.0{\pm}0.0}$	$\mathbf{33.0{\pm}0.9}$	$31.4{\pm}1.0$	$28.0{\pm}1.4$	$20.4{\pm}1.9$	$6.6{\pm}2.2$	$13.0{\pm}2.1$
SOTA robust agents:							
RadialDQN	$32.6{\pm}0.5$	$\mathbf{33.0{\pm}0.0}$	$28.4{\pm}1.2$	$22.8{\pm}1.9$	$20.0{\pm}1.1$	$21.0{\pm}0.6$	$22.8{\pm}1.7$
SADQN	$30.0{\pm}0.0$	$30.0{\pm}0.0$	$27.2{\pm}1.2$	$20.4{\pm}0.5$	$20.8{\pm}1.0$	$18.8{\pm}1.3$	$21.0{\pm}1.8$
WocaRDQN	$32.2{\pm}1.2$	$29.0{\pm}1.3$	$21.8{\pm}2.0$	$20.6{\pm}0.8$	$21.2{\pm}1.0$	$22.0{\pm}0.0$	$21.4{\pm}1.6$
VanillaDQN	$\mathbf{34.0{\pm}0.0}$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$
Previous smoothed agents:							
RadialDQN+RS	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$21.8{\pm}1.2$
SADQN+RS	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.9$	$21.6{\pm}1.7$	$22.8{\pm}0.8$	$22.6{\pm}1.2$
WocaRDQN+RS	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$22.2{\pm}2.2$	$21.8{\pm}1.2$
VanillaDQN+RS	$22.2{\pm}2.2$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$	$0.0{\pm}0.0$
RoadRunner							
Ours:							
S-DQN (Radial)	$39380{\pm}4579$	$39360{\pm}4566$	$\mathbf{40480{\pm}8076}$	$25640{\pm}3232$	$21060{\pm}2286$	$\mathbf{13020{\pm}4935}$	$\mathbf{11220{\pm}4324}$
S-DQN (S-PGD)	$42780{\pm}6316$	$42620{\pm}3953$	$35740{\pm}5420$	$\mathbf{27380{\pm}8896}$	$\mathbf{21360{\pm}9340}$	$2840{\pm}1756$	$0{\pm}0$
S-DQN (Vanilla)	$47480{\pm}8807$	$23320{\pm}3932$	$3460{\pm}5924$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$
SOTA robust agents:							
RadialDQN	$39620{\pm}4821$	$\mathbf{43520{\pm}4081}$	$24160{\pm}2604$	$15500{\pm}6466$	$1020{\pm}937$	$620{\pm}492$	$3560{\pm}488$
SADQN	$46680{\pm}7742$	$28580{\pm}2584$	$3240{\pm}1544$	$780{\pm}840$	$420{\pm}523$	$100{\pm}200$	$2640{\pm}1317$
WocaRDQN	$32480{\pm}5096$	$1580{\pm}2108$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$20{\pm}40$
VanillaDQN	$\mathbf{48320{\pm}5989}$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$
Previous smoothed agents:							
RadialDQN+RS	$13420{\pm}1955$	$11260{\pm}2504$	$9220{\pm}3080$	$6680{\pm}1705$	$7780{\pm}1900$	$3180{\pm}1326$	$7420{\pm}2604$
SADQN+RS	$18520{\pm}2510$	$14240{\pm}6013$	$16440{\pm}3817$	$1960{\pm}1323$	$1040{\pm}1074$	$560{\pm}973$	$1180{\pm}922$
WocaRDQN+RS	$5120{\pm}3319$	$560{\pm}647$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$
VanillaDQN+RS	$29640{\pm}5271$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$	$0{\pm}0$

A.11. Detailed experiment results of reward lower bound for S-DQN

Table 8 shows the details of the reward lower bound for smoothed DQN agents under different ℓ_2 budget ϵ. We use the same budget ϵ for every state, and hence, the total budget is $B = \epsilon\sqrt{H}$, where H is the length of the trajectory. We set $H = 2500$ in Pong, Freeway, and RoadRunner. The reward lower bound of S-DQN (Vanilla) is comparable with the bound of S-DQN (Radial) and S-DQN (S-PGD), suggesting that our S-DQN already achieves a high robustness guarantee without further combining with other robust agents or leveraging adversarial training.

Table 8. The reward lower bound of smoothed DQN agents under different ℓ_2 attack budgets. The smoothing variance σ for all the agents is set to 0.1 in Pong, 0.1 in Freeway, and $\sigma = 0.05$ in RoadRunner.

Pong	ℓ_2 attack budget				
$\epsilon(\ell_2)$	0.001	0.002	0.003	0.004	0.005
Ours:					
S-DQN (Radial)	**20.0**	**20.0**	**19.0**	**18.0**	**18.0**
S-DQN (S-PGD)	18.0	17.0	16.0	14.0	11.0
S-DQN (Vanilla)	18.0	17.0	16.0	15.0	14.0
Previous smoothed agents:					
RadialDQN+RS	−21.0	−21.0	−21.0	−21.0	−21.0
SADQN+RS	−21.0	−21.0	−21.0	−21.0	−21.0
WocaRDQN+RS	−21.0	−21.0	−21.0	−21.0	−21.0
VanillaDQN+RS	−21.0	−21.0	−21.0	−21.0	−21.0
Freeway					
Ours:					
S-DQN (Radial)	**31.0**	**30.0**	**29.0**	28.0	**28.0**
S-DQN (S-PGD)	**31.0**	**30.0**	**29.0**	28.0	27.0
S-DQN (Vanilla)	**31.0**	**30.0**	**29.0**	**29.0**	**28.0**
Previous smoothed agents:					
RadialDQN+RS	20.0	20.0	20.0	20.0	19.0
SADQN+RS	20.0	20.0	20.0	20.0	19.0
WocaRDQN+RS	20.0	20.0	20.0	20.0	19.0
VanillaDQN+RS	13.0	12.0	11.0	10.0	9.0
RoadRunner					
Ours:					
S-DQN (Radial)	**36200**	**29400**	**21612**	14163	14001
S-DQN (S-PGD)	33000	24483	19295	**19104**	**19100**
S-DQN (Vanilla)	32215	25097	21123	18067	18001
Previous smoothed agents:					
RadialDQN+RS	9400	5497	2295	2104	2100
SADQN+RS	17900	15197	12623	9567	9501
WocaRDQN+RS	3300	1200	593	306	300
VanillaDQN+RS	500	100	0	0	0

A.12. Detailed experiment results of robust reward for S-PPO

Table 9 shows the reward of PPO agents under different ℓ_∞ attacks. Note that we trained each agent 15 times and reported the median of the performance as suggested in Zhang et al. (2020) to get a fair and comparable result. Our S-PPO exhibits high clean reward and robust reward under attacks in all environments, while the previous smoothed agents only achieve similar performance compared to the original robust agents.

Table 9. The reward of PPO agents under different attacks. The smoothing variance σ for all the smoothed agents is set to 0.2 in Walker and Hopper. The ℓ_∞ attack budget is set to 0.075 in both environments.

Walker	Clean reward	MAD attack	Min-RS attack	Optimal attack	PA-AD attack
Ours:					
S-PPO (SGLD)	4566	**4537**	**4241**	4085	**4026**
S-PPO (Radial)	2117	2160	1028	915	689
S-PPO (WocaR)	4363	4360	3907	3920	3867
S-PPO (S-ATLA)	**4897**	4460	2170	**5010**	2980
S-PPO (S-PA-ATLA)	4407	4045	2379	144	372
S-PPO (Vanilla)	4552	4386	3203	944	1077
SOTA robust agents:					
SGLDPPO	4329	4177	2376	2747	718
RadialPPO	2221	2230	1270	132	152
WocaRPPO	4110	3918	1950	2916	2067
ATLAPPO	3564	2567	672	818	263
PA-ATLAPPO	2548	1717	591	183	298
VanillaPPO	4301	2806	551	437	275
Previous smoothed agents:					
SGLDPPO+RS	4290	4124	2739	1615	717
RadialPPO+RS	1804	1883	610	145	208
WocaRPPO+RS	4013	4160	1362	3211	1765
ATLAPPO+RS	4129	3348	894	1090	322
PA-ATLAPPO+RS	1325	1990	427	322	332
VanillaPPO+RS	3582	2892	592	440	401
Hopper					
Ours:					
S-PPO (SGLD)	2894	2896	**2428**	1579	1523
S-PPO (Radial)	3756	**3205**	1212	1285	**2015**
S-PPO (WocaR)	2335	2194	1328	1053	1189
S-PPO (S-ATLA)	**3770**	2557	1752	**2595**	1927
S-PPO (S-PA-ATLA)	3737	2631	1839	1655	1950
S-PPO (Vanilla)	3583	2765	1049	995	1190
SOTA robust agents:					
SGLDPPO	2772	2587	1107	1087	1463
RadialPPO	3291	3056	1182	900	1161
WocaRPPO	3652	2993	1111	1112	1331
ATLAPPO	3577	1493	1245	1172	1124
PA-ATLAPPO	3508	3297	1110	1518	1842
VanillaPPO	3321	2375	834	695	789
Previous smoothed agents:					
SGLDPPO+RS	2354	2386	1106	1059	1411
RadialPPO+RS	3298	2876	1011	1122	1165
WocaRPPO+RS	3535	2878	1084	1095	1208
ATLAPPO+RS	3278	1485	1220	1161	1129
PA-ATLAPPO+RS	3537	3027	1365	1861	1866
VanillaPPO+RS	3211	2238	920	707	840

A.13. Detailed experiment results of reward lower bound for S-PPO

Table 10 shows the details of the reward lower bound for smoothed PPO agents under different ℓ_2 budget ϵ. We use the same budget ϵ for every state, and hence, the total budget $B = \epsilon\sqrt{H}$, where H is the length of the trajectory. We set $H = 1000$ in Walker and Hopper. Our S-PPOs exhibit higher reward lower bounds compared to their naively smoothed counterparts.

Table 10. The reward lower bound of smoothed PPO agents under different ℓ_2 attack budgets. The smoothing variance σ for all the agents is set to 0.2 in all environments.

Walker	ℓ_2 attack budget				
$\epsilon(\ell_2)$	0.002	0.004	0.006	0.008	0.01
Ours:					
S-PPO (SGLD)	4496	4478	4460	**4440**	**4420**
S-PPO (Radial)	1648	1413	1159	817	550
S-PPO (WocaR)	4345	4333	4322	4308	4296
S-PPO (S-ATLA)	**4781**	**4556**	3571	2287	1746
S-PPO (S-PA-ATLA)	4364	4017	2526	1573	1100
S-PPO (Vanilla)	4585	4531	**4476**	4368	2189
Previous smoothed agents:					
SGLDPPO+RS	4159	3703	2886	2236	1839
RadialPPO+RS	1160	987	821	654	420
WocaRPPO+RS	4235	4195	4130	3969	2178
ATLAPPO+RS	935	735	568	378	307
PA-ATLAPPO+RS	606	512	455	416	385
VanillaPPO+RS	1263	979	853	748	657
Hopper					
Ours:					
S-PPO (SGLD)	2783	**2758**	**2732**	**2710**	**2661**
S-PPO (Radial)	**2865**	2294	1925	1760	1574
S-PPO (WocaR)	1691	1573	1470	1397	1360
S-PPO (S-ATLA)	1935	1700	1456	1338	1217
S-PPO (S-PA-ATLA)	1883	1603	1438	1309	1176
S-PPO (Vanilla)	1959	1646	1447	1321	1206
Previous smoothed agents:					
SGLDPPO+RS	1773	1534	1464	1361	1212
RadialPPO+RS	2073	1724	1479	1278	1146
WocaRPPO+RS	2076	1832	1696	1533	1473
ATLAPPO+RS	1293	1183	1095	1041	966
PA-ATLAPPO+RS	1750	1500	1319	1114	1040
VanillaPPO+RS	1300	1218	1046	970	895

A.14. Additional experiments

Table 11. The reward of our S-PPO (Vanilla) under Humanoid, Ant, and Halfcheetah environments. Our S-PPO (Vanilla) outperforms the previous smoothed agents significantly without further combining other robust training algorithms. The attack budget is set to 0.075 for Humanoid and 0.15 for HalfCheetah and Ant.

Humanoid	Clean reward	MAD attack	Min-RS attack	Optimal attack	PA-AD attack
S-PPO (Vanilla)	**6956**	**6336**	**4620**	**6785**	**265**
VanillaPPO+RS	4875	1581	1014	3350	153
VanillaPPO	4913	1766	1040	3074	153
Ant					
S-PPO (Vanilla)	5654	**4466**	**1437**	**871**	**474**
VanillaPPO+RS	6106	942	378	-1560	-1817
VanillaPPO	**6141**	710	338	-1555	-1817
Halfcheetah					
S-PPO (Vanilla)	5140	**4171**	**3577**	**2703**	**2648**
VanillaPPO+RS	5272	560	327	-490	-382
VanillaPPO	**5371**	527	207	-489	-412

Table 12. The reward of our S-DQN (Vanilla) with different smoothing variance σ. A higher σ usually leads to more robust S-DQN agents but with a trade-off of decreasing the clean reward.

Pong	Clean reward	S-PGD				
$\epsilon(\ell_\infty)$		0.01	0.02	0.03	0.04	0.05
S-DQN (Vanilla) $\sigma = 0.01$	**21.0±0.0**	8.0±4.0	-20.8±0.4	-20.8±0.4	-20.8±0.4	-20.8±0.4
S-DQN (Vanilla) $\sigma = 0.05$	**21.0±0.0**	20.8±0.4	**20.6±0.5**	18.6±2.2	-11.0±3.4	-20.6±0.5
S-DQN (Vanilla) $\sigma = 0.1$	20.4±0.5	**21.0±0.0**	20.4±0.8	**20.2±0.8**	**16.6±4.4**	**18.4±2.1**
S-DQN (Vanilla) $\sigma = 0.15$	18.8±1.5	19.6±1.2	17.8±3.2	17.6±1.9	14.6±3.2	14.4±3.0
Freeway						
S-DQN (Vanilla) $\sigma = 0.01$	**34.0±0.0**	16.6±1.9	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
S-DQN (Vanilla) $\sigma = 0.05$	33.6±0.5	**33.8±0.4**	**31.6±1.5**	6.8±1.7	0.0±0.0	0.0±0.0
S-DQN (Vanilla) $\sigma = 0.1$	**34.0±0.0**	33.0±0.9	31.4±1.0	**28.0±1.4**	20.4±1.9	6.6±2.2
S-DQN (Vanilla) $\sigma = 0.15$	26.4±1.0	26.6±1.6	26.8±1.0	25.2±1.9	**24.0±2.5**	**20.2±1.3**
RoadRunner						
S-DQN (Vanilla) $\sigma = 0.01$	45180±8944	840±869	0±0	0±0	0±0	0±0
S-DQN (Vanilla) $\sigma = 0.05$	**47480±8807**	**23320±3932**	3460±5924	0±0	0±0	0±0
S-DQN (Vanilla) $\sigma = 0.1$	39200±6156	19640±2263	**11160±5644**	620±1040	0±0	0±0
S-DQN (Vanilla) $\sigma = 0.15$	16860±1334	16540±671	**11160±993**	**4680±5629**	**940±1830**	**20±40**

Table 13. The reward of our S-PPO (Vanilla) with different smoothing variance σ. The best σ settings for Walker and Hopper are 0.2 and 0.3 respectively. However, we use $\sigma = 0.2$ in every environment for simplicity.

Walker	Clean reward	MAD attack	Min-RS attack	Optimal attack	PA-AD attack
S-PPO (Vanilla) $\sigma = 0.1$	**4798**	4316	1598	**2853**	822
S-PPO (Vanilla) $\sigma = 0.2$	4552	**4386**	**3203**	944	**1077**
S-PPO (Vanilla) $\sigma = 0.3$	4207	4218	2098	744	915
Hopper					
S-PPO (Vanilla) $\sigma = 0.1$	3392	2653	1014	569	918
S-PPO (Vanilla) $\sigma = 0.2$	3583	2765	1049	995	1190
S-PPO (Vanilla) $\sigma = 0.3$	**3642**	**2864**	**1135**	**1366**	**2083**

Table 14. testing time cost and clean reward of S-DQN (Vanilla) and S-PPO (Vanilla) under different sample numbers m. We can see that $m = 5$ is already sufficient to achieve high clean reward and the time cost is not high even with $m = 100$.

Pong	$m = 100$	$m = 10$	$m = 5$	$m = 1$
S-DQN (Vanilla) test time (sec/step)	0.1154	0.0106	0.0092	0.0042
S-DQN (Vanilla) clean reward	21.0±0.0	20.6±0.5	20.4±0.5	18.2±3.2
Walker				
S-PPO (Vanilla) test time (sec/step)	0.0094	0.0026	0.0022	0.0019
S-PPO (Vanilla) clean reward	4552±65	4442±86	4593±92	4654±168

Table 15. The ablation study of S-DQN (Vanilla) without Denoiser. It is hard to learn S-DQN agents without Denoiser.

Pong	Clean reward	S-PGD				
$\epsilon(\ell_\infty)$		0.01	0.02	0.03	0.04	0.05
S-DQN (Vanilla)	**20.4±0.5**	**21.0±0.0**	**20.4±0.8**	**20.2±0.8**	**16.6±4.4**	**18.4±2.1**
S-DQN (Vanilla) w/o Denoiser	−21.0±0.0	−21.0±0.0	−21.0±0.0	−21.0±0.0	−21.0±0.0	−21.0±0.0
Freeway						
S-DQN (Vanilla)	**34.0±0.0**	**33.0±0.9**	**31.4±1.0**	**28.0±1.4**	**20.4±1.9**	**6.6±2.2**
S-DQN (Vanilla) w/o Denosier	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
RoadRunner						
S-DQN (Vanilla)	**47480±8807**	**23320±3932**	**3460±5924**	0±0	0±0	0±0
S-DQN (Vanilla) w/o Denoiser	960±0	0±0	0±0	0±0	0±0	0±0

Table 16. The comparison between the smoothed attack and the non-smoothed attack for the PPO setting. We use the prefix "S-" to denote the Smoothed Attack. Unlike the DQN setting, we did not observe a significant difference between the smoothed attack and the non-smoothed attack.

Agents	Environments	MAD attack	Min-RS attack	Optimal attack	PA-AD attack
S-PPO (Vanilla)	Walker	**4386**	**3203**	**944**	**1077**
	Hopper	**2765**	**1049**	995	1190
Agents	Environments	S-MAD attack	S-Min-RS attack	S-Optimal attack	S-PA-AD attack
S-PPO (Vanilla)	Walker	4637	3225	949	1224
	Hopper	2910	1057	**979**	**1114**

Table 17. Addtional results for S-DQN (SADQN) and S-DQN (WocaR). Our S-DQN can also use SADQN and WocaRDQN as base agents.

Pong	Clean reward	S-PGD				
$\epsilon(\ell_\infty)$		0.01	0.02	0.03	0.04	0.05
S-DQN (SADQN)	**21.0±0.0**	**21.0±0.0**	**20.6±0.8**	20.0±1.1	17.0±5.0	13.4±2.1
S-DQN (WocaR)	19.8±1.6	19.8±1.2	19.0±2.6	**20.6±0.5**	**20.0±0.9**	**15.6±4.9**
Freeway						
S-DQN (SADQN)	30.0±0.0	30.0±0.0	29.6±0.5	26.4±1.6	26.6±1.9	26.8±1.0
S-DQN (WocaR)	**31.2±1.3**	**31.4±1.0**	**31.8±1.2**	**30.2±1.2**	**29.6±1.9**	**28.6±1.4**
RoadRunner						
S-DQN (SADQN)	**44560±8724**	**41180±5618**	**37920±6478**	**36600±4994**	**32480±3803**	**27160±3287**
S-DQN (WocaR)	39120±6430	36980±6978	18880±9335	7520±9212	20±40	2100±2417