Robust Deep Reinforcement Learning against Adversarial Perturbations on Observations

Huan Zhang [*1] **Hongge Chen** [*2] **Chaowei Xiao** [3] **Bo Li** [4] **Duane Boning** [2] **Cho-Jui Hsieh** [1]

Abstract

Deep Reinforcement Learning (DRL) is vulnerable to small adversarial perturbations on state observations. These perturbations do not alter the environment directly, but can mislead the agent into making suboptimal decisions. We analyze the Markov Decision Process (MDP) under this threat model and utilize tools from the neural network verification literature to enable robust training for DRL under observational perturbations. Our techniques are general and can be applied to both Deep Q Networks (DQN) and Deep Deterministic Policy Gradient (DDPG) algorithms for discrete and continuous action control problems. We demonstrate that our proposed training procedure significantly improves the robustness of DQN and DDPG agents under a suite of strong white box attacks on observations, including a few novel attacks we specifically craft. Additionally, our training procedure can produce provable certificates for the robustness of a Deep RL agent.

1. Introduction

With deep neural networks (DNNs) as powerful function approximators, deep reinforcement learning (DRL) has achieved great success on many complex tasks (Mnih et al., 2015; Lillicrap et al., 2015; Silver et al., 2016; Gu et al., 2016). Among them, Deep Q Networks (DQN) (Mnih et al., 2015) and Deep Deterministic Policy Gradient algorithms (DDPG) (Lillicrap et al., 2015) have become the roots of many enhanced algorithms (Wang et al., 2015; Hessel et al., 2018; Lowe et al., 2017; Fujimoto et al., 2018; Barth-Maron et al., 2018). Despite achieving super-human level performance on many tasks, the existence of adversarial examples (Szegedy et al., 2013) in DNNs and many successful attacks to DRL (Huang et al., 2017; Lin et al., 2017; Pattanaik et al., 2018; Xiao et al., 2019) have motivated us to study robust deep reinforcement learning algorithms under adversarial settings.

Since each element of RL (observations, actions, transition dynamics and rewards) can contain uncertainty, robust reinforcement learning has been studied from different perspectives. For example, Robust Markov Decision Process (RMDP) (Iyengar, 2005) considers the worst case transition probability from the environment, and theory developed in RMDP has inspired robust deep Q-learning (Shashua & Mannor, 2017) and policy gradient algorithms (Mankowitz et al., 2018; Derman et al., 2018; Mankowitz et al., 2019) that are robust against small environment changes. Adversarial multi-agent learning (Pinto et al., 2017; Li et al., 2019) and minimax games (Littman, 1994) studied the case where the agent interacts with an opponent agent in the same environment and they learn together. Tessler et al. (2019) considered adversarial perturbations on action space. Fu et al. (2017) investigated how to learn a robust reward.

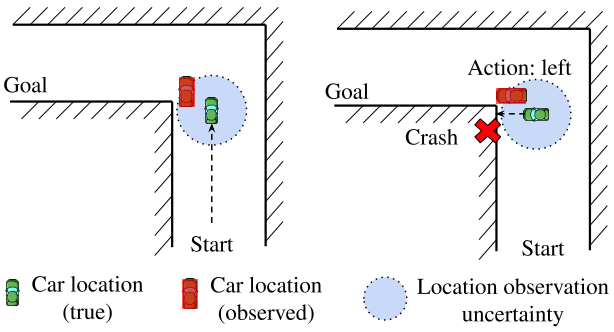


Figure 1. A car observes its location through sensors (e.g., GPS) and plans its route to the goal. Without considering the uncertainty in observed location (e.g., error of GPS coordinates), an unsafe policy may crash into the wall as the observed location and true location differ.

In our paper, we focus on the situation where the state observation of the agent contains uncertainty, where the uncertainty can origin from observation errors (Figure 1) or even adversarial noises. To ensure that an agent perform safely even under the worst case uncertainty, we consider the adversarial setting where the observation of an agent is adversarially perturbed from s to $\nu(s)$, yet the underlying true environment state s is unchanged. This setting is

[*]Equal contribution [1]UCLA [2]MIT [3]University of Michigan [4]UIUC. Correspondence to: Huan Zhang and Hongge Chen <huan@huan-zhang.com, chenhg@mit.edu>.

particularly aligned with many adversarial attacks on state observations (e.g., Huang et al. (2017); Lin et al. (2017)). To obtain a better theoretical understanding, we first consider a Markov Decision Process (MDP) under the existence of an optimal adversary for perturbing state observations (named SA-MDP). We derive the basic policy evaluation algorithm and discuss a few unusual properties of SA-MDP: an optimal policy may not exist for SA-MDP, however we can bound the loss of expected reward for a given MDP policy π. From the analysis of SA-MDP, we provide connections to DRL in both discrete and continuous action spaces. We derive theoretically principled robustness regularizers for DQN and DDPG which are related to an upper bound on performance loss.

We conduct experiments on 11 environments ranging from Atari games with discrete actions to complex robotics control tasks in continuous action space. We show that our proposed method leads to significantly better robustness under strong white-box adversarial attacks on state observations, including two novel attacks we design, the Robust Sarsa attack (RS attack) and Maximal Action Difference attack (MAD attack) for DDPG. Additionally, we demonstrate that we can achieve better performance than baselines under environment parameter changes (e.g., changes in mass and friction). Moreover, since our method utilizes certifiable neural network training as a keystone, we can obtain certain robustness guarantees for the learned policy.

2. Related Works

Robust Markov Decision Process Robust Markov Decision Process (RMDP), first proposed by Iyengar (2005) and Nilim & El Ghaoui (2004), considers the worst case perturbation from transition probabilities. The agent can observe the original true state from the environment and act accordingly, but the environment can choose from a set of transition probabilities that minimizes rewards. Under some mild assumptions, Iyengar (2005) showed that an optimal policy for RMDP can be solved using robust value iteration. RMDP is useful to model environmental uncertainty, and has been extended to distributional settings (Xu & Mannor, 2010) and partially observed MDPs (Osogami, 2015), and applied to Deep Q learning (Shashua & Mannor, 2017) and robust policy optimization (Mankowitz et al., 2018; Derman et al., 2018; Mankowitz et al., 2019). Many of these works consider environment perturbations (e.g., changes in physical parameters like mass and length in the simulation environment), and desire a robust model to perform well when the environment parameters differ from the training environment. In contrast, our threat model considers adversarial perturbations on observations and does not change the environment dynamics; we will also show that our state-adversarial MDP (SA-MDP) has many properties different from MDP and RMDP.

Adversarial Multi-agent Learning Several works consider the adversarial setting of multi-agent reinforcement learning (Tan, 1993; Bu et al., 2008). In the 2-player case, an agent can learn together with an opponent (e.g., kick and defend), which can be formulated as a two-player zero-sum Markov game (Littman, 1994). In this situation, we consider two agents where each chooses an action at each step, and the environment transits based on both actions. The regular action-value function $Q(s, a)$ can then be extended to $Q(S, a, o)$ where o is the opponent's action. Fortunately, Q-learning is still convergent under this setting, and it can be extended to deep Q learning and policy gradient algorithms (Li et al., 2019; Pinto et al., 2017). Pinto et al. (2017) show that learning an opponent agent simultaneously using TRPO (Schulman et al., 2015) can improve the agent's performance as well as its robustness against environment turbulence and test conditions (e.g., change in mass and friction). Gu et al. (2019) carried out real-world experiments on the two-player adversarial learning game. The minimax Markov game and adversarial multi-agent learning settings are different from ours as our adversary only manipulates the observations, but does not change the underlying environment directly.

Adversarial Examples and Defense in DRL Huang et al. (2017) first evaluates the robustness of deep reinforcement learning policies through an FGSM based attack on Atari games learned by A3C (Mnih et al., 2016), TRPO (Schulman et al., 2015) and DQN algorithms. Kos & Song (2017) proposed to use the value function to guide adversarial perturbation search. Lin et al. (2017) considered a more complicated case where the adversary is allowed to only attacking at a small subset of time steps, and used a generative model to generate attack plans that can lure the agent to a designated target state. Pattanaik et al. (2018) further enhanced these attacks with multi-step gradient descent and better engineered loss functions. Many recent works studied attacking an RL agent under different settings and we refer the reader to a recent survey (Xiao et al., 2019) which has thoroughly characterized adversarial attacks in deep reinforcement learning.

Several works studied adversarial defense under state perturbations. Mandlekar et al. (2017) used a weak FGSM based attack with policy gradient to adversarially train a few simple RL tasks and demonstrated that they can become more resistant to attacks. Pattanaik et al. (2018) used stronger multi-step gradient based attacks, however their evaluation focused on the robustness against environment changes rather than state perturbations. Behzadan & Munir (2017) applied adversarial training to more complicated tasks (Atari games with DQN), however they found it is

much harder to train the Q network adversarially under strong attacks, and rollout performance can drop significantly. To obtain better test time performance, Mirman et al. (2018a); Fischer et al. (2019) used imitation learning to build a robustly trained student policy network which mimics the behavior of an existing DQN agent, which outperforms naive adversarial training. Havens et al. (2018) proposed a meta online learning procedure with a master agent detecting the presence of the adversary and switch between a few sub-policies, but did not focus on how to train a single agent robustly. Lütjens et al. (2019) considered the worst-case scenario during roll-outs for an existing DQN agents to ensure safety, but it relies on an existing policy and does not include a training procedure. We will compare against the state-of-the-art (Fischer et al., 2019) in our experiments.

Certified defenses against adversarial examples Certified defenses, first developed in (Wong & Kolter, 2018; Raghunathan et al., 2018), can provide provable certificates on classification error under adversarial perturbations. Many of these kind of methods require a convex or linear relaxation of neural networks (Salman et al., 2019). Given a neural network function $f_\theta(x)$, these methods can provide its lower and upper bounds: $l_\theta(x) \leq f_\theta(x) \leq u_\theta(x)$ when x is perturbed within a small neighbourhood (e.g. a ℓ_∞ norm ball). Efficient tools (Xu et al., 2020) are available to compute these bounds. Since $l_\theta(x)$ and $u_\theta(x)$ are also functions of θ, they can be trained to give tight robustness guarantees. Many certified defenses have followed this methodology (Mirman et al., 2018b; Wong et al., 2018; Wang et al., 2018; Balunovic & Vechev, 2020). Empirical defenses like adversarial training (Kurakin et al., 2016; Madry et al., 2018), however, cannot provide any guaranteed upper or lower bounds when $f_\theta(x)$ is perturbed and thus their test accuracy under adversarial attacks cannot be mathematically guaranteed. In this paper, we use CROWN-IBP (Zhang et al., 2020), a state-of-the-art certified defense method as a building block. CROWN-IBP combines a fast interval bound propagation method in (Mirman et al., 2018b; Gowal et al., 2018) with a tight convex relaxation, CROWN (Zhang et al., 2018), allowing efficient training of tight certificates.

3. Methodology

3.1. State-Adversarial Markov Decision Process

A Markov Decision Process (MDP) is a 4-tuple, $(\mathcal{S}, \mathcal{A}, R, p)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, and $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ represents the transition probability of environment, where $\mathcal{P}(\cdot)$ defines the set of all possible probability measures on a the given set. The transition probability is $p(s'|s, a) = \Pr(s_{t+1} = s'|s_t = s, a_t = a)$, where t is the time step. A discount factor is defined as $0 < \gamma < 1$. We denote a stationary policy as $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, the set of all stochastic and Markovian policies as Π_{MR} and the set of all deterministic and Markovian policies as Π_{MD}.

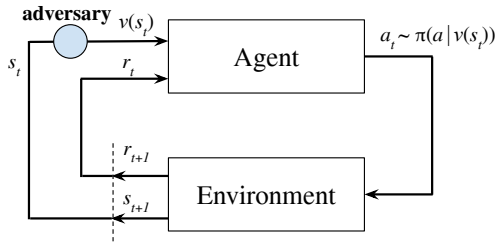


Figure 2. Reinforcement learning with perturbed state observations. The agent observes a perturbed state $\nu(s_t)$ produced by an adversary rather than the true environment state s_t.

In state-adversarial MDP (SA-MDP), we introduce an adversary $\nu(s) : \mathcal{S} \to \mathcal{S}$. The role of the adversary is to perturb the observation of the agent, such that the action is taken as $\pi(a|\nu(s))$, however the environment still transits from state s rather than $\nu(s)$ to the next state. Since $\nu(s)$ may be different from s, the agent's action from $\pi(a|\nu(s))$ may be sub-optimal and thus the adversary is able to reduce the reward earned by the agent. In real world control problems, the adversary can be reflected as the worst case noise in measurement or state estimation uncertainty. Note that this scenario is different from the two-player Markov game (Littman, 1994) where both players interact with the environment directly. For the adversary ν we have the following assumptions:

Assumption 1 (Stationary, Deterministic and Markovian Adversary)**.** *The adversary $\nu(s)$ is a deterministic function $\nu : \mathcal{S} \to \mathcal{S}$ which only depends on the current state s, and does not change over time.*

This Markovian assumption holds for many realistic attacks as in test time, many gradient based attacks only depend on the current state input and the policy or Q network; the network parameters are frozen in test time, so given the same s the adversary will generate the same perturbation, which satisfies Assumption 1. Also, we allow the adversary to perturb the observation at every step.

Assumption 2 (Bounded Adversary Power)**.** $\nu(s) \in B(s)$ *where $B(s)$ is a small set of states and $s \in B(s)$.*

Assumption 2 restricts the adversary to perturb a state s only to a predefined set of states $B(s)$. Under the context of adversarial examples, $B(s)$ is usually a set of task-specific "neighbouring" states of s. This assumption constrains the power of the adversary, which makes the observation still meaningful (yet not accurate) even with perturbations. The

definitions of adversarial value and action-value functions under ν is similar to the definition of regular MDP:

$$\tilde{V}^{\pi}_{\nu}(s) = \mathbb{E}_{\pi\circ\nu}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s\right]$$
$$\tilde{Q}^{\pi}_{\nu}(s,a) = \mathbb{E}_{\pi\circ\nu}\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t=s, a_t=a\right],$$

where the reward at step-t is defined as r_t and $\pi\circ\nu$ is the policy under observation perturbations: $\pi(a|\nu(s))$. Based on these two assumptions, we state our theorems for the State-Adversarial Markov Decision Process. The proofs of our theorems are provided in Appendix.

Theorem 1 (Bellman Equations for fixed π and ν)**.** *Given $\pi:\mathcal{S}\to\mathcal{P}(\mathcal{A})$ and $\nu:\mathcal{S}\to\mathcal{S}$, we have*

$$\tilde{V}^{\pi}_{\nu}(s) = \sum_{a\in\mathcal{A}}\pi(a|\nu(s))\sum_{s'\in\mathcal{S}}p(s'|s,a)\left[R(s,a,s')+\gamma\tilde{V}^{\pi}_{\nu}(s')\right]$$
$$\tilde{Q}^{\pi}_{\nu}(s,a) = \sum_{s'\in\mathcal{S}}p(s'|s,a)\left[R(s,a,s')+\gamma\sum_{a'\in\mathcal{A}}\pi(a'|\nu(s'))\tilde{Q}^{\pi}_{\nu}(s',a')\right].$$

We first consider the optimal adversary $\nu^*(\pi)$ that minimizes the total expected reward for a given π, and define the optimal adversarial value and action-value functions:

$$\tilde{V}^{\pi}_{\nu^*}(s) = \min_{\nu}\tilde{V}^{\pi}_{\nu}(s)$$
$$\tilde{Q}^{\pi}_{\nu^*}(s,a) = \min_{\nu}\tilde{Q}^{\pi}_{\nu}(s,a).$$

Theorem 2 (Bellman Contraction for Optimal Adversary)**.** *Define the Bellman operator $\mathscr{L}:\mathbb{R}^{|\mathcal{S}|}\to\mathbb{R}^{|\mathcal{S}|}$,*

$$(\mathscr{L}\tilde{V}^{\pi})(s) = \min_{s_\nu\in B(s)}\sum_{a\in\mathcal{A}}\pi(a|s_\nu)\sum_{s'\in\mathcal{S}}p(s'|s,a)\left[R(s,a,s')+\gamma\tilde{V}^{\pi}(s')\right]. \quad (1)$$

The Bellman equation for optimal adversary ν^* can be written as:

$$\tilde{V}^{\pi}_{\nu^*} = \mathscr{L}\tilde{V}^{\pi}_{\nu^*} \quad (2)$$

Additionally, $\mathscr{L}$ is a contraction that converges to $\tilde{V}^{\pi}_{\nu^*}$.

Based on Theorem 2 we have a policy evaluation algorithm for SA-MDP (Algorithm 1 in the appendix), that computes the $V^{\pi}_{\nu^*}(s)$ for each $s\in\mathcal{S}$. Given a π, value functions for MDP and SA-MDP can be vastly different; in Appendix A, we give an example where its optimal MDP policy has 0 (lowest) reward in SA-MDP.

Following the standard results in MDP, one may hope to find the *optimal* policy π^* for SA-MDP such that

$$\tilde{V}^{\pi^*}_{\nu^*(\pi^*)}(s) \geq \tilde{V}^{\pi}_{\nu^*(\pi)}(s) \quad \text{for } \forall s\in\mathcal{S} \text{ and } \forall\pi. \quad (3)$$

where the subscript $\nu^*(\pi)$ explicitly indicates that ν^* is the optimal adversary for π. Unfortunately, we show the following surprising negative results in Theorem 3 and Theorem 4:

Theorem 3. *There exists a SA-MDP and some stochastic policy $\pi\in\Pi_{MR}$ such that we cannot find a better deterministic policy $\pi'\in\Pi_{MD}$ satisfying $\tilde{V}^{\pi'}_{\nu^*(\pi')}(s)\geq\tilde{V}^{\pi}_{\nu^*(\pi)}(s)$ for all $s\in\mathcal{S}$.*

Contrarily, in classical MDP, for any stochastic policy we can find a deterministic policy that is at least as good as the stochastic policy. With an optimal adversary, this does not hold anymore. This conclusion also aligns with many works using randomization to defend against adversarial examples – randomization can help under strong adversaries.

Theorem 4. *Under the optimal adversary ν^*, an optimal policy $\pi^*\in\Pi_{MR}$ does not necessarily exist for SA-MDP.*

The optimal policy π^* requires to have $\tilde{V}^{\pi^*}_{\nu^*(\pi^*)}(s)\geq\tilde{V}^{\pi}_{\nu^*(\pi)}(s)$ for all s and any π. In a SA-MDP, surprisingly, sometimes we have to make a trade-off between the value of two states and there is no policy that can maximize the values of all states simultaneously.

However, not all hopes are lost and we show that under certain assumptions, even under an optimal adversary, the loss in performance can be bounded:

Theorem 5. *Given a policy π for a non-adversarial MDP. Under the optimal adversary ν in SA-MDP, for all $s\in\mathcal{S}$ we have*

$$\max_{s\in\mathcal{S}}\left\{V^{\pi}(s)-\tilde{V}^{\pi}_{\nu^*}(s)\right\}\leq\alpha\max_{s\in\mathcal{S}}\max_{\hat{s}\in B(s)}\mathrm{D}_{TV}(\pi(\cdot|s),\pi(\cdot|\hat{s})) \quad (4)$$

where $\mathrm{D}_{TV}(\pi(\cdot|s),\pi(\cdot|\hat{s}))$ is the total variation distance between $\pi(\cdot|s)$ and $\pi(\cdot|\hat{s})$, and

$$\alpha := 2[1+\frac{\gamma}{(1-\gamma)^2}]\max_{(s,a,s')\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}}|R(s,a,s')|$$

is a constant that does not depend on π.

Theorem 5 shows that as long as $\mathrm{D}_{TV}(\pi(a|s),\pi(a|\hat{s}))$ is not too large for any $\hat{s}$ close to s (within the power of adversary), the performance gap between $\tilde{V}^{\pi}_{\nu^*}(s)$ (SA-MDP) and $V^{\pi}(s)$ (regular MDP) can be bounded. This motivates us to add a regularization term for $\pi(a|s)$ during training to obtain a policy that is robust even under optimal attacks. We will discuss details on how to regularize $\pi(a|s)$ below in two popular settings, DQN and DDPG.

3.2. State-Adversarial Deep Q Networks (SA-DQN)

In Deep Q Network training, a neural network is used to learn the action-value function $Q(s,a)$. The action space is assumed to be finite. The input of the network is s and the number of outputs equals to the number of actions. Each output represents the Q function value $Q(s,a)$ at a given action a with the input state s. $Q(s,a)$ is typically learned using a temporal-difference (TD) loss, and the learned deterministic

policy $\pi(a|s)$ is chosen such that

$$\pi(a|s) = \begin{cases} 1 & a = \arg\max_{a'} Q(s, a'), \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Inspired by Theorem 5, we try to reduce the total variation distance together with the ordinary temporal-difference (TD) loss in training. In DQN the policy π is assumed deterministic, and probability 1 is assigned to the action with the largest logit value. When π is deterministic, we have

$$\mathrm{D}_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s})) = \begin{cases} 0 & \arg\max_a \pi(a|s) = \arg\max_a \pi(a|\hat{s}) \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

Then we aim to encourage π to satisfy the condition $\arg\max_a \pi(a|s) = \arg\max_a \pi(a|\hat{s})$ for all $\hat{s} \in B(s)$ by simply adding a regularization term in our loss. In our empirical evaluation, we consider the case where $B(s)$ is defined by interval ranges around s, i.e.,

$$B(s) := \{\hat{s} : s_l \le \hat{s} \le s_u, s_u \ge s, s_l \le s\}.$$

Here the inequalities are defined as element-wise when s is a vector. For two arbitrary actions a^* and a, we denote

$$Q_\theta^-(\hat{s}, a, a^*) := Q_\theta(\hat{s}, a^*) - Q_\theta(\hat{s}, a).$$

$Q_\theta^-(\hat{s}, a, a^*)$ can also be regarded as a neural network parametrized by θ with one more fixed linear layer (subtraction) after the original Q network. We can use the linear relaxation based neural network verification tools (which are frequently used in certified defenses), as introduced in Section 2 to obtain its lower bound (denoted as $l_{Q_\theta, a^*, a}(s)$):

$$Q_\theta^-(\hat{s}, a, a^*) := Q_\theta(\hat{s}, a) - Q_\theta(\hat{s}, a) \ge l_{Q_\theta, a^*, a}(s) \quad \forall a,\ a^* \in \mathcal{A}, \quad \forall \hat{s} \in B(s). \tag{7}$$

Then we let $a^* = \arg\max_a Q_\theta(s, a)$ and if $l_{Q_\theta, a^*, a}(s) \ge 0$, we know that the top-1 of Q_θ does not change in $B(s)$, which means the action chosen by the policy will not change for $\forall s_l \le \hat{s} \le s_u$. Note that $l_{Q_\theta, a^*, a}(s)$ is actually a differentiable function of θ. So we can design loss functions to encourage positive $l_{Q_\theta, a^*, a}(s)$ value to ensure that $Q_\theta(\hat{s}, \cdot)$ and $Q_\theta(s, \cdot)$ have the same top-1 logit for all $\hat{s} \in B(s)$. Thus we define a hinge loss to encourage a positive $l_{Q_\theta, a^*, a}(s)$:

$$\tilde{L}(s; \theta) := \max\big\{-c, -\min_{a \ne a^*} l_{Q_\theta, a^*, a}(s)\big\} \tag{8}$$

where $c > 0$ is a small confidence constant. Note that in this loss a^* is the top-1 action chosen by Q and thus this hinge loss does not directly depend on actions and state observations sampled from the replay buffer. The total training loss for DQN is given by

$$L(s, a, s'; \theta) = \text{TD-}L(s, a, s'; \theta) + \kappa \tilde{L}(s; \theta), \tag{9}$$

where TD-$L(\cdot)$ is the regular TD loss of DQN training and κ is a regularization parameter. The full algorithm is presented in Algorithm 2 in Appendix C.

3.3. State-Adversarial Deep Deterministic Policy Gradient (SA-DDPG)

Deep Deterministic Policy Gradient (DDPG) extends DQN to continuous control problems with an actor-critic structure. When the action space is continuous, it can be challenging to learn a probability distribution of actions given a state directly, so DDPG learns a deterministic policy $\pi(s) \in \mathbb{R}^{|\mathcal{A}|}$ instead. In this situation, the total variation distance $\mathrm{D}_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s}))$ is malformed, as the densities at different states s and $\hat{s}$ are very likely to be completely non-overlapping. To address this issue, we define a smoothed version of policy, $\bar{\pi}(a|s)$, in DDPG, where we add independent Gaussian noise with variance σ^2 to each action:

$$\bar{\pi}(a|s) \sim \mathcal{N}(\pi(s), \sigma^2 I_{|\mathcal{A}|})$$

Then we can compute $\mathrm{D}_{TV}(\bar{\pi}(\cdot|s), \bar{\pi}(\cdot|\hat{s}))$ using the following theorem:

Theorem 6. $\mathrm{D}_{TV}(\bar{\pi}(\cdot|s), \bar{\pi}(\cdot|\hat{s})) = \sqrt{\frac{2}{\pi}} \frac{d}{\sigma} + O(d^3)$, *where* $d = \|\pi(s) - \pi(\hat{s})\|_2$.

Thus, as long as we can penalize $\|\pi(s) - \pi(\hat{s})\|_2$, the total variation distance between the two smoothed distributions can be bounded. In DDPG, we parameterize the policy as a policy network π_{θ_π} and the critic as a critic network Q_{θ_Q}. According to Theorem 5, for each state we need to find $\max_{\hat{s} \in B(s)} \mathrm{D}_{TV}(\bar{\pi}(\cdot|s), \bar{\pi}(\cdot|\hat{s}))$, and we use $\max_{\hat{s} \in B(s)} \|\pi_{\theta_\pi}(s) - \pi_{\theta_\pi}(\hat{s})\|_2$ as a surrogate. Note that the smoothing procedure can be done completely in test time, and during training time our goal is to keep $\max_{\hat{s} \in B(s)} \|\pi_{\theta_\pi}(s) - \pi_{\theta_\pi}(\hat{s})\|_2$ small.

Similar to DQN, for $B(s) = \{\hat{s} : s_l \le \hat{s} \le s_u, s_u \ge s, s_l \le s\}$, we use linear relaxation based perturbation analysis tools to give linear upper and lower bounds of $\pi_{\theta_\pi}(\hat{s})$

$$l_\pi(s) \le \pi_{\theta_\pi}(\hat{s}) \le u_\pi(s), \quad \forall \hat{s} \in B(s) \tag{10}$$

where the bounds l_π and u_π are differentiable functions of θ_π. So we can obtain the upper bound of the norm of the difference as follows:

$$\max_{\hat{s} \in B(s)} \|\pi_{\theta_\pi}(s) - \pi_{\theta_\pi}(\hat{s})\|_2^2 \le \sum_{i=1}^{|\mathcal{A}|} \max\left\{(\pi_{\theta_\pi}(s)_i - l_\pi(s))^2, (\pi_{\theta_\pi}(s)_i - u_\pi(s))^2\right\}. \tag{11}$$

We then add (11) into the regular policy gradient loss in DDPG to regularize $\pi_{\theta_\pi}(s)$ with a regularization term κ. The full algorithm is presented in Algorithm 3 in Appendix D.

3.4. Robustness certificates

Since we use neural network verification tools to train our networks (sometimes referred to as a "certified defense"), it can produce robustness certificates for our task. However in RL tasks the certificates have different interpretations, as discussed in details below.

Robustness Certificates for DQN. In DQN, the action space is finite, so we have a robustness certificate on the actions taken at each state. More specifically, at each state s, policy π's action is certified if its corresponding Q function satisfies

$$\arg\max_a Q_\theta(s,a) = \arg\max_a Q_\theta(\hat{s},a) = a^* \tag{12}$$

for all $\hat{s} \in B(s)$. As mentioned in Section 3.2 if $l_{Q_\theta,a^*,a} \geq 0$ holds for all $\hat{s} \in B(s)$, we have

$$Q_\theta^-(\hat{s},a,a^*) := Q_\theta(\hat{s},a^*) - Q_\theta(\hat{s},a) \geq 0 \tag{13}$$

is guaranteed for all $a \in \mathcal{A}$, which means that the agent's action will not change when the state observation is in $B(s)$. When the agent's action is not changed under adversarial perturbation, its reward and transition at current step will not change in the DQN setting, either.

Robustness Certificates for DDPG. In DDPG, the action space is continuous, hence it is not possible to certify that actions do not change under adversary. We instead seek for a different type of guarantee, where we can upper bound the change in action given a norm bounded input perturbation:

$$U_s \geq \max_{\hat{s}\in B(s)} \|\pi_{\theta_\pi}(\hat{s}) - \pi_{\theta_\pi}(s)\|$$

Given a state s, we can use neural network verification tools to compute a U_s. Generally speaking, if $B(s)$ is small, a robust policy desires to have a small U_s, otherwise it can be possible to find an adversarial state perturbation that greatly changes $\pi_{\theta_\pi}(\hat{s})$ and cause the agent to misbehave. We will discuss an attack of this type, the maximal action difference (MAD) attack, in Section 4.

It is worth to mention that, our robustness certificates above only depend on the neural network and $B(s)$, but do not rely on the adversarial attacks used. So these certificates hold even if the attacks do not satisfy Assumption 1, as long as Assumption 2 is satisfied. In other word, the proposed SA-DQN and SA-DDPG also work when the attacks are non-stationary, stochastic and non-Markovian, as long as the their output $\hat{s}$ is in $B(s)$, a set of "neighbouring" states of s.

4. Robustness Evaluation via Attacks

Strong white-box attacks are essential for evaluating the effectiveness of any defense methods. Here we discuss a few attacks for DDPG and DQN, including our two novel attacks, Robust Sarsa and Maximal Action Difference. All these attacks satisfy Assumption 1 and 2.

4.1. Attack for DQN

For DQN, we use the regular untargeted Projected Gradient Decent (PGD) attack in the literature (Lin et al., 2017; Pattanaik et al., 2018; Xiao et al., 2019). The untargeted PGD attack with K iterations updates the state K times as follows:

$$\begin{aligned} s^{k+1} &= s^k + \eta\mathcal{P}[\nabla_{s^k}\mathcal{H}(Q_\theta(s^k,\cdot),a^*)], \\ s^0 &= s, \quad k = 0,\ldots,K-1 \end{aligned} \tag{14}$$

where $\mathcal{H}(Q_\theta(s^k,\cdot),a^*)$ is the cross-entropy loss between the output logits of $Q_\theta(s^k,\cdot)$ and the onehot-encoded distribution of $a^* := \arg\max_a Q_\theta(s,a)$. $\mathcal{P}$ is a projection operator depending on the norm constraint of $B(s)$ and η is the learning rate. A successful untargeted PGD attack will then perturb the state to lead the Q network to output an action other than the maximum probability action a^* chosen at the original state s. To guarantee that the final state obtained by the attack is within an ℓ_∞ ball around s ($B_\epsilon(s) = \{\hat{s} : s-\epsilon \leq \hat{s} \leq s+\epsilon\}$), the projection $\mathcal{P}$ is a sign operator and η is typically set to $\epsilon = \frac{\epsilon}{K}$.

4.2. Attacks for DDPG

Weakness of traditional gradient based attack on actor and critic. Pattanaik et al. (2018) and many other works use the gradient of critic network $Q_{\theta_Q}(s,a)$ to provide the direction to update states (in K steps) adversarially:

$$\begin{aligned} s^{k+1} &= s^k - \eta\mathcal{P}[\nabla_{s^k}Q_{\theta_Q}(s^k,\pi_{\theta_\pi}(s^k))], \\ s^0 &= s, \quad k = 0,\ldots,K-1 \end{aligned} \tag{15}$$

Here $\pi(s)$ is a deterministic policy, $\mathcal{P}$ is a projection operator depending on the norm constraint of $B(s)$, η is the learning rate, and s is the state under attack and initially $s^0 \leftarrow s$. The attack attempts to find a state s^K that minimizes the action-value. If Q is a perfect action-value function, $\hat{s}$ leads to the worst possible action that minimizes the value. However, the strength of attack strongly depends the quality of the critic; if Q is poorly learned, or if Q itself is not robust against small perturbations, or if Q has obfuscated gradients, this attack can fail as no right update direction is given. When we evaluate the robustness of a policy, we desire it to be independent of a specific critic network to avoid these problems. We thus propose two novel *critic independent* attacks to DDPG below.

Robust Sarsa (RS) attack. Since the agent uses a fixed π during evaluation, it is not hard to learn its corresponding $Q^\pi(s,a)$ values using temporal-difference learning algorithms without knowing the critic network used for training. We propose to use the *on-policy* TD-learning algorithm, Stateactionrewardstateaction (Sarsa) (Rummery & Niranjan, 1994) algorithm to learn $Q^\pi(s,a)$ to facilitate our attack. We follow the classical Sarsa update rule where the TD loss for the RS "critic" is:

$$L_{RS}(\theta_{RS}) = [r_{t+1} + \gamma Q^\pi_{RS}(s_{t+1}, a_{t+1}) - Q^\pi_{RS}(s_t, a_t)]^2$$

Q^π is approximated by a neural network $Q^\pi_{\theta_{RS}}$. Since π does not change over-time and Sarsa is on-policy, we found that we can learn a reasonably good Q^π pretty quickly (we only run 30,000 steps). We found that the robustness of $Q^\pi_{\theta_{RS}}$ is very important; if $Q^\pi_{\theta_{RS}}$ is not robust against small perturbations in actions, it cannot provide a good signal for finding an attack direction. Thus, we again to use CROWN-IBP to train the $Q^\pi_{\theta_{RS}}$ network robustly. Then, we use $Q^\pi_{\theta_{RS}}$ as a critic to perform critic-based attacks as in (15). We found that this robust Sarsa attack can sometimes significantly outperform the attack using the critic trained along with the policy network; moreover, its attack strength does not depend on the quality of an existing critic function.

Although beyond the scope of this paper, RS attack can also be used as a blackbox attack when perturbing the actions, as $Q^\pi_{\theta_{RS}}$ can be learned by observing the environment and the agent without any internal information of the agent. Then, using the robust critic we learned, black-box attacks can be performed on action space by solving $\min Q^\pi_{\theta_{RS}}(s,a)$ with a norm constrained a.

Maximal Action Difference (MAD) attack. We propose a second attack which does not depend on a critic, by going to the opposite direction of Theorem 5 and 6 and maximize the difference between $\pi(s)$ and $\pi(\hat{s})$:

$$L_{\text{MAD}}(\hat{s}) = -\max_{\hat{s}\in B(s)} \|\pi(s) - \pi(\hat{s})\|_2^2.$$

$L_{\text{MAD}}(\hat{s})$ is the loss function for this attack which we use projected gradient descent to minimize. We find that this attack sometimes outperforms critic based attacks.

Hybrid RS+MAD attack. We found that RS and MAD attack can achieve best results on different tasks. We thus combine them to form a hybrid attack, which minimizes both the robust critic predicted value and maximizes action differences by minimizing this loss function:

$$L_{\text{Hybrid}}(\hat{s}) = Q_{\theta_Q}(s, \pi_{\theta_{RS}}(\hat{s})) + \lambda L_{\text{MAD}}(\hat{s})$$

We try different values of λ and report the best attack (lowest reward) as the reward under attack.

5. Experiments

In this section we show our empirical results of SA-DQN and SA-DDPG. Our source code is available at `https://github.com/chenhongge/StateAdvDRL`.

5.1. Evaluation of robustly trained DQN (SA-DQN)

In this section we evaluate our proposed algorithm for DQN on four Atari games and one classic control problem, Acrobot, in OpenAI Gym (Brockman et al., 2016). Among many techniques to improve DQN training, we implemented DoubleDQN (Van Hasselt et al., 2016) and Prioritized Experience Replay (Schaul et al., 2015) in our experiments. Detailed parameter and training information can be found in the appendix. For Atari games, we normalize the pixel values from $[0, 255]$ to $[0, 1]$ in preprocessing while no normalization is applied to Acrobot environment. We choose $B(s) = B_\epsilon(s) = \{s' : s - \epsilon \le s' \le s + \epsilon\}$ in our experiments. In SA-DQN, for Atari games we use $\epsilon = 1/255$. For Acrobot, since there does not exist a limit on the state value and each state feature has different range, we first run a well-trained natural agent and collect the standard deviation (std) of each state feature value over 100 episodes. Then perturbation range ϵ on each state feature is determined individually, depending on the standard deviation of that feature. We choose $\epsilon = 0.2$std as our budget. The same ϵ value is used for both training and attack at test time.

We evaluate our proposed method under PGD attack introduced in Section 4.1. In all PGD attacks, we set number of iterations $K = 10$ and use the same ϵ value as in the training. As a comparison, we also train a regular DQN with the same model hyperparameters (detailed in Appendix C). For each agent, we run 50 episodes and report the average score it achieves with and without adversarial attacks. When tested in clean environments without adversarial attacks, we also report the percentage of actions that can be certified among all actions an agent takes. The results are presented in Table 1. As a comparison, we list the best results in Fischer et al. (2019). Note that in Fischer et al. (2019), the authors only used $k = 4$ in their PGD attack, which is weaker than our PGD attack with $k = 10$. We can see that our proposed method achieves much higher average reward under attacks in most environments. Also, the robust models' action certification rate is close to 1. The higher the action certification rate, the fewer actions can be changed under *any* norm bounded adversarial attacks. In Atari environments, there are so few actions that cannot be certified that the agents' reward virtually remains the same even under strong PGD attacks.

Environment	ϵ	DQN Natural Reward	DQN Attack Reward	DQN Action Cert. Rate	SA-DQN Natural Reward	SA-DQN Attack Reward	SA-DQN Action Cert. Rate	Fischer et al. (2019) Natural Reward	Fischer et al. (2019) Attack Reward
Pong	1/255	20.7 ± 0.5	-21.0±0.0	0.0	21.0 ± 0.0	**20.1 ± 0.0**	1.000	19.73	18.13
Freeway	1/255	32.9 ± 0.7	0.0 ± 0.0	0.0	30.78 ± 0.5	30.36 ± 0.7	0.995	32.93	**32.53**
BankHeist	1/255	1308.4 ± 24.1	56.4 ± 21.2	0.0	1041.4 ± 12.3	**1043.6 ± 9.5**	0.997	238.66	190.67
RoadRunner	1/255	36946.0 ± 6089.0	0.0 ± 0.0	0.0	15172.0 ± 791.7	**15280.0 ± 827.7**	0.969	12106.67	5753.33
Acrobot	0.2 std	-67.5 ± 8.8	-349.68 ± 178.0	0.735	-79.4 ± 17.4	**-86.7 ± 19.9**	0.856	-	-

Table 1. Average test reward values and action certification rates over 50 episodes with and without test time PGD attacks on regular DQN and SA-DQN models. Natural reward is the reward in clean environment without adversarial attacks. Attack reward is the reward under PGD attacks on state observations. Action Cert. Rate is the proportion of the actions that are guaranteed and cannot be changed by any attacks within given ϵ. We also include the state-of-the-art results from Fischer et al. (2019).

Environment	ϵ	State Space	Method	Natural Reward	Attack Reward: Critic Attack	MAD Attack	RS Attack	RS+MAD	Best Attack
Ant	0.2	111	DDPG	1750.3±522.3	1897.7±623.4	**181.4±223.6**	834.4±427.7	370.4±271.7	181.4
			SA-DDPG	**2697.2±412.6**	2816.0±112.9	2119.4±871.9	2342.3±518.9	**1994.3±885.5**	**1994.3**
HalfCheetah	0.5	17	DDPG	**7602.6±235.7**	2039.5±866.7	698.4±402.8	1056.6±739.4	**628.8±547.8**	628.8
			SA-DDPG	7299.6±124.4	3401.0±147.8	**1482.3±441.3**	2633.3±136.1	1615.7±418.7	**1482.3**
Hopper	0.075	11	DDPG	3261.2±33.3	3092.3±747.3	**1121.8±678.8**	1512.1±5.2	1313.8±716.5	1121.8
			SA-DDPG	**3618.5±9.1**	3613.8±10.5	3283.8±764.1	2096.8±601.0	**1989.1±626.6**	**1989.1**
Inverted Pendulum	0.75	4	DDPG	1000±0.0	1000±0.0	102.7±83.3	832.7±35.3	**77.8±29.869**	77.8
			SA-DDPG	1000±0.0	1000±0.0	1000±0.0	1000±0.0	**900.8±297.6**	**900.8**
Reacher	1.5	11	DDPG	**-4.4±1.1**	-10.6±3.2	-26.1±4.9	-10.9±1.9	**-28.5±4.2**	-28.5
			SA-DDPG	-5.5±1.3	-9.3±1.4	**-11.7±1.8**	-9.7±1.4	-10.7±1.5	**-11.7**
Walker2d	0.15	17	DDPG	1904.1±975.1	2191.5±746.0	489.3±446.7	**54.3±29.8**	641.8±904.6	54.3
			SA-DDPG	**4042.3±869.8**	4353.8±1000.4	**3345.5±1105.6**	3792.7±658.0	4081.6±927.6	**3345.5**

Table 2. Rewards on 6 Mujoco environments using policies trained by DDPG and SA-DDPG. Natural reward is the reward in clean environment without adversarial attacks. The "Best Attack" column reports the lowest reward over all four attacks, and this lowest reward is used for robustness evaluation.

Environment	ϵ	DDPG ℓ_2	DDPG Range	SA-DDPG ℓ_2	SA-DDPG Range
Ant	0.2	4.19	1.45	0.49	0.16
HalfCheetah	0.5	4.73	1.92	3.22	1.06
Hopper	0.075	2.96	1.68	0.58	0.30
InvertedPendulum	0.75	1.05	1.05	0.23	0.23
Reacher	1.5	1.50	1.06	0.31	0.21
Walker2d	0.15	4.57	1.85	0.57	0.19

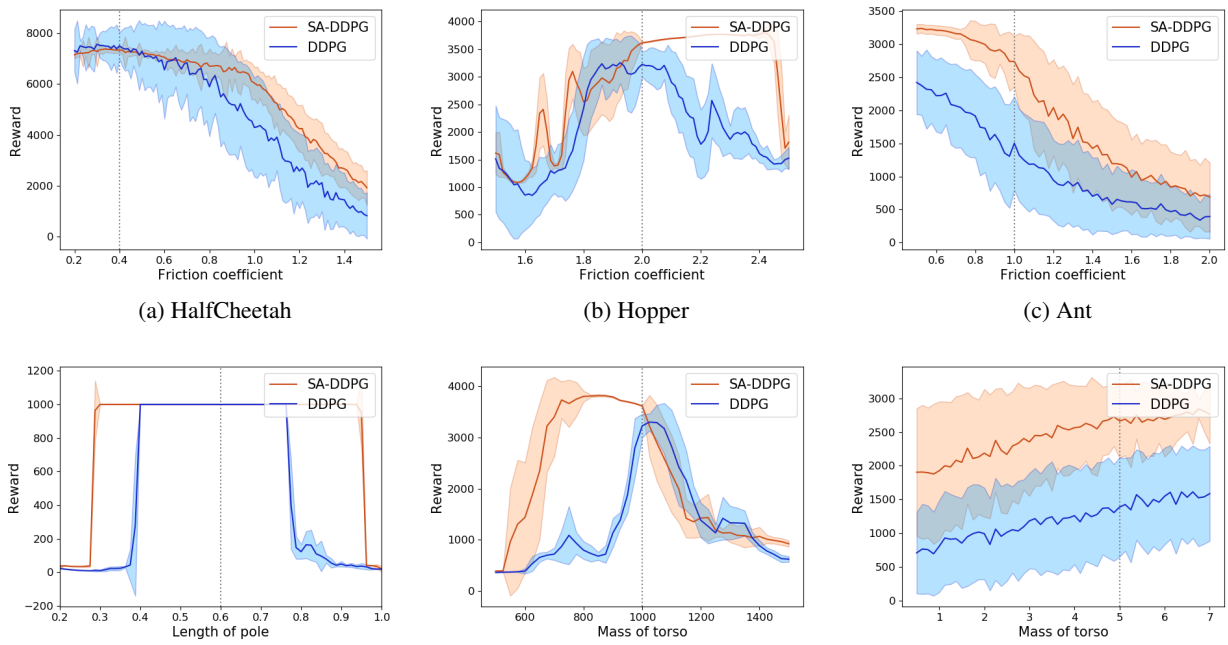
Table 3. Certificates of action change given bounded state perturbations. Numbers are averaged over all steps in 10 rollouts. Smaller numbers indicate better certificates.

5.2. Evaluation of robustly trained DDPG (SA-DDPG)

In this section we evaluate our robust SA-DDPG algorithm using six Mujoco (Todorov et al., 2012) continuous control environments in OpenAI Gym. SA-DDPG is trained using the same hyperparameters and the same number of training steps as DDPG (detailed in Appendix D), except for an additional regularization parameter on the actor regularization term (11) which is searched in $\{0.3, 1, 3, 10, 30\}$. In Table 2, we show the average and standard deviation of reward for baseline DDPG and SA-DDPG under different attack settings.

Model performance without adversary. Our DDPG baseline rewards are similar to numbers reported in previous works (Lillicrap et al., 2015; Fujimoto et al., 2018). With an action space regularizer (11), we found that the performance of SA-DDPG can sometimes significantly improve (Ant, Hopper and Walker2d environments). This indicates that the robustness of policy play an important rule on improving DRL performance even for the non-adversarial setting.

Model performance under state adversarial attacks. We consider a ℓ_∞ norm-like bounded perturbation, where each state variable is perturbed individually within a predefined $\pm\epsilon$ range. Since each state variable can have greatly different range (e.g., the range of position and velocity variables can be quite different), we rescale ϵ by the standard deviations of each state variable. The standard deviations are calculated using data collected on baseline policy without adversary. We investigate model performance under four adversarial attacks on state observations: critic based attack (Pattanaik et al., 2018) and our proposed Robust Sarsa (RS) attack, Maximal Action Difference (MAD) attack and RS+MAD attack. We found that it is important to use strong attacks to evaluate the robustness of DRL methods. The basic critic-based white-box attack (Pattanaik et al., 2018) is weaker in many settings than our proposed



(a) HalfCheetah (b) Hopper (c) Ant

(d) InvertedPendulum (e) Hopper (f) Ant

Figure 3. Comparison between DDPG and SA-DDPG on the robustness to changing environment dynamics. The dotted line on each figure represents the original environment parameter (friction, length or mass) used during training.

attacks. For example, for InvertedPendulum, critic-based attack cannot decrease the reward of DDPG, but RS+MAD can reduce reward from 1000 to 77.8. We use the lowest reward from all attacks as the final evaluation score. We found that the performance of DDPG drops significantly under strong adversaries like RS, MAD and RS+MAD. SA-DDPG demonstrates high robustness under attacks and outperforms DDPG by a large margin.

Model performance under environment dynamics changes. Although SA-DDPG is proposed to improve robustness in state observations, we found that it also improves general robustness of the policy against environment dynamics changes. In Figure 3, we evaluate the average reward during 100 rollouts for DDPG and SA-DDPG agents under different environment parameters (friction, mass and length). In HalfCheetah, the performance of SA-DDPG is slightly lower than DDPG baseline, however when the sliding friction increases, SA-DDGP's performance only slowly drops, and becomes better than DDPG when friction coefficient becomes larger. For Hopper, although SA-DDPG and DDPG perform similar when the friction and mass are the same as the training environment, SA-DDPG demonstrate superior performance under friction and mass changes. For InvertedPendulum, SA-DDPG learns a policy that remains valid for pole length in a much wider range. We omit Walker2d environment in comparison since DDPG performs much worse than SA-DDPG even without environment dynamic changes. The Reacher environment is relatively insensitive to dynamic changes so we do not include it.

Robustness certificates. In SA-DDPG, we can obtain robustness certificates that give bounds on actions in the presence of bounded perturbation on state inputs. Given an input state s, we use CROWN-IBP to obtain the upper and lower bounds for each action: $l_i(s) \leq \pi_i(\hat{s}) \leq u_i(s), \forall \hat{s} \in B(s)$. We consider the following certificates on $\pi(s)$: the average output range $\frac{\|u(s)-l(s)\|_1}{|\mathcal{A}|}$ which reflect the tightness of bounds, and the ℓ_2 distance defined in (11). Note that bounds on other ℓ_p norms can also be computed given $l_i(s)$ and $u_i(s)$. Since the Mujoco action space is normalized within $[-1, 1]$, the worst case range is 2. We report both certificates for all six environments in Table 3. DDPG without action regularizer usually cannot obtain certificates (range is close to 2). SA-DDPG can provide robustness certificates (bounded inputs guarantee bounded outputs) which can be useful for further study of DRL robustness.

6. Conclusions

In our paper, we consider the uncertainty in state observations in reinforcement learning and formulate this problem as a state-adversarial Markov decision process (SA-MDP). Based on our analysis of SA-MDP, we propose theoretically principled regularizers for DQN and DDPG and borrow techniques from neural network verification literature to

train neural network polices with robustness certificates. To thoroughly evaluate policy robustness, we propose two novel attacks under the state perturbation setting which complements existing critic-based attacks for continuous control tasks. Our proposed algorithms, SA-DQN and SA-DDPG, are demonstrated on 11 environments with both discrete and continuous action spaces, and significantly improve model performance under strong adversaries on state observations.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Balunovic, M. and Vechev, M. Adversarial training and provable defenses: Bridging the gap. In *ICLR*, 2020.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Tb, D., Muldal, A., Heess, N., and Lillicrap, T. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

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

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Bu, L., Babu, R., De Schutter, B., et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Derman, E., Mankowitz, D. J., Mann, T. A., and Mannor, S. Soft-robust actor-critic policy-gradient. *arXiv preprint arXiv:1803.04848*, 2018.

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

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Fujimoto, S., Van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

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

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pp. 2829–2838, 2016.

Gu, Z., Jia, Z., and Choset, H. Adversary a3c for robust reinforcement learning. *arXiv preprint arXiv:1912.00330*, 2019.

Havens, A., Jiang, Z., and Sarkar, S. Online robust policy learning in the presence of unknown adversaries. In *Advances in Neural Information Processing Systems*, pp. 9916–9926, 2018.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

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

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

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

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., and Russell, S. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4213–4220, 2019.

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

Lin, Y.-C., Hong, Z.-W., Liao, Y.-H., Shih, M.-L., Liu, M.-Y., and Sun, M. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pp. 6379–6390, 2017.

Lütjens, B., Everett, M., and How, J. P. Certified adversarial robustness for deep reinforcement learning. *arXiv preprint arXiv:1910.12908*, 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ICLR*, 2018.

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

Mankowitz, D. J., Mann, T. A., Bacon, P.-L., Precup, D., and Mannor, S. Learning robust options. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Mankowitz, D. J., Levine, N., Jeong, R., Abdolmaleki, A., Springenberg, J. T., Mann, T., Hester, T., and Riedmiller, M. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516*, 2019.

Mirman, M., Fischer, M., and Vechev, M. Distilled agent dqn for provable adversarial robustness. 2018a.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3575–3583, 2018b.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Nilim, A. and El Ghaoui, L. Robustness in markov decision problems with uncertain transition matrices. In *Advances in neural information processing systems*, pp. 839–846, 2004.

Osogami, T. Robust partially observable markov decision process. In *International Conference on Machine Learning*, pp. 106–115, 2015.

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., and Chowdhary, G. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. Safe policy iteration. In *International Conference on Machine Learning*, pp. 307–315, 2013.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. In *NIPS*, 2018.

Rummery, G. A. and Niranjan, M. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems 32*, pp. 9832–9842. 2019.

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

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

Shashua, S. D.-C. and Mannor, S. Deep robust kalman filter. *arXiv preprint arXiv:1703.02310*, 2017.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2013.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Tessler, C., Efroni, Y., and Mannor, S. Action robust reinforcement learning and applications in continuous control. *arXiv preprint arXiv:1901.09184*, 2019.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

Wang, S., Chen, Y., Abdou, A., and Jana, S. Mixtrain: Scalable training of formally robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5283–5292, 2018.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, J. Z. Scaling provable adversarial defenses. In *NIPS*, 2018.

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

Xu, H. and Mannor, S. Distributionally robust markov decision processes. In *Advances in Neural Information Processing Systems*, pp. 2505–2513, 2010.

Xu, K., Shi, Z., Zhang, H., Huang, M., Chang, K.-W., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis on general computational graphs. *arXiv preprint arXiv:2002.12920*, 2020.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *NIPS*, 2018.

Zhang, H., Chen, H., Xiao, C., Li, B., Boning, D., and Hsieh, C.-J. Towards stable and efficient training of verifiably robust neural networks. *ICLR*, 2020.

A. An example of SA-MDP

We first show a simple environment and solve it under different policies for MDP and SA-MDP. The environment have three states $\mathcal{S} = \{S_1, S_2, S_3\}$ and 2 actions $\mathcal{A} = \{A_1, A_2\}$. The transition probabilities and rewards are defined as (unmentioned probabilities and rewards are 0):

$$\begin{aligned}
&\Pr(s' = S_1 | s = S_1, a = A_1) = 1.0 \\
&\Pr(s' = S_2 | s = S_1, a = A_2) = 1.0 \\
&\Pr(s' = S_2 | s = S_2, a = A_2) = 1.0 \\
&\Pr(s' = S_3 | s = S_2, a = A_1) = 1.0 \\
&\Pr(s' = S_1 | s = S_3, a = A_2) = 1.0 \\
&\Pr(s' = S_2 | s = S_3, a = A_1) = 1.0 \\
&R(s = S_1, a = A_2, s' = S_2) = 1.0 \\
&R(s = S_2, a = A_1, s' = S_2) = 1.0 \\
&R(s = S_3, a = A_1, s' = S_3) = 1.0
\end{aligned}$$

The environment is illustrated in Figure 4. For the power of adversary, we allow ν to perturb one state to any other two neighbouring states:

$$B_\nu(S_1) = B_\nu(S_2) = B_\nu(S_3) = \{S_1, S_2, S_3\}$$

Now we evaluate various policies for MDP and SA-MDP for this environment. We use $\gamma = 0.99$ as the discount factor.

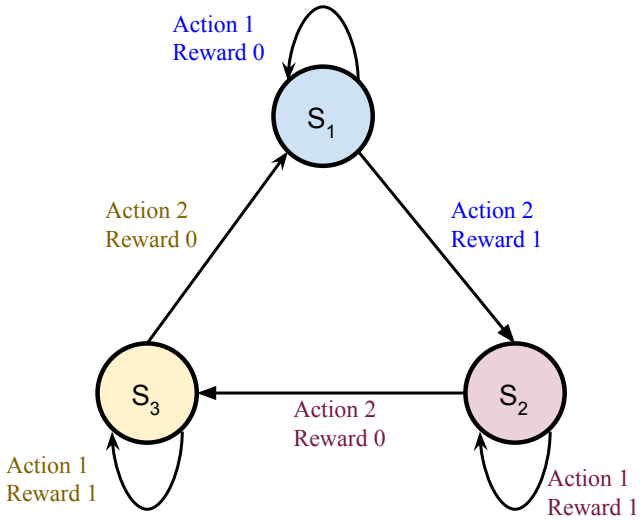


Figure 4. A simple 3-state environment.

A policy in this environment can be described by 3 parameters p_{11}, p_{21}, p_{31} where $p_{ij} \in [0, 1]$ denotes the probability $\Pr(a = A_j | s = S_i)$.

- **Optimal Policy for MDP.** For a regular MDP, the optimal solution is $p_{11} = 0$, $p_{21} = 1$, $p_{31} = 1$. We take A_2 to receive reward and leave S_1, and then keep doing A_1 in S_2 and S_3. The values for each state are $V(S_1) = V(S_2) = V(S_3) = \frac{1}{1-\gamma} = 100$, which is optimal. However, this policy obtains $V(S_1) = V(S_2) = V(S_3) = 0$ for SA-MDP, because we can set $\nu(S_1) = S_2$, $\nu(S_2) = S_1$, $\nu(S_3) = S_1$ and consequentially we always take the wrong action receiving 0 reward.
- **A Stochastic Policy for MDP and SA-MDP.** We consider a stochastic policy where $p_{11} = p_{21} = p_{31} = 0.5$. Under this policy, we randomly stay or move in each state, and has a 50% probability of receiving a reward. The adversary ν

has no power because π is the same for all states. In this situation, $V(S_1) = V(S_2) = V(S_3) = \frac{0.5}{1-0.99} = 50$ for both MDP and SA-MDP.

- **Deterministic Policies for SA-MDP.** Now we consider all $2^3 = 8$ possible deterministic policies for SA-MDP. Note that if for any state S_i we have $p_{i1} = 0$ and another state S_j we have $p_{j1} = 1$, we always have $V(S_1) = V(S_2) = V(S_3) = 0$. This is because we can set $\nu(S_1) = S_j$, $\nu(S_2) = S_i$ and $\nu(S_3) = S_i$ and always receive a 0 reward. Thus the only two possible other policies are $p_{11} = p_{21} = p_{31} = 0$ and $p_{11} = p_{21} = p_{31} = 1$, respectively. For $p_{11} = p_{21} = p_{31} = 1$ we have $V(S_1) = 0, V(S_2) = V(S_3) = 100$ as we always take A_1 and never transit to other states; for $p_{11} = p_{21} = p_{31} = 0$, we circulate through all three states and only receive a reward when we leave A_1. We have $V(S_1) = \frac{1}{1-\gamma^3} \approx 33.67$, $V(S_2) = \frac{\gamma^2}{1-\gamma^3} \approx 33.00$ and $V(S_3) = \frac{\gamma}{1-\gamma^3} \approx 33.33$.

Figure 5, 6, 7 give the graph of $V(S_1)$, $V(S_2)$ and $V(S_3)$ under three different settings of p_{11}. The figures are generated using Algorithm 1.

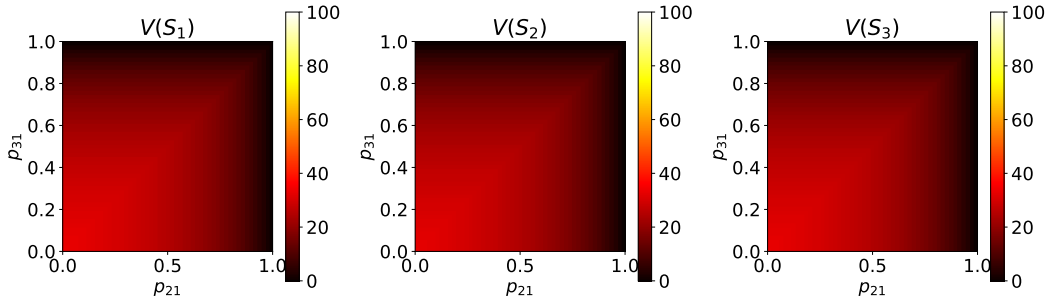


Figure 5. Value functions when $p_{11} = 0$

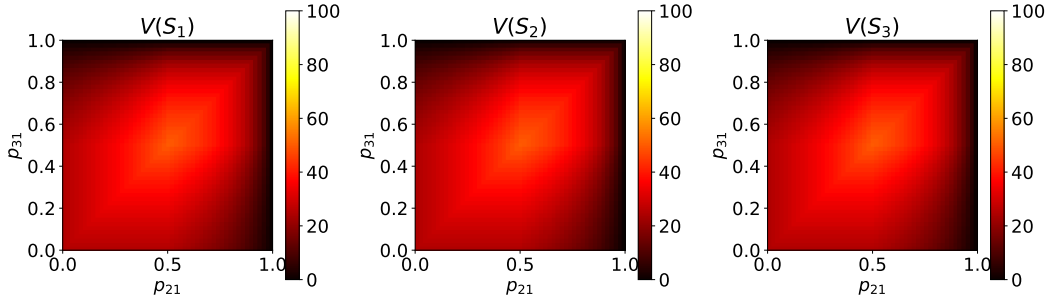


Figure 6. Value functions when $p_{11} = 0.5$

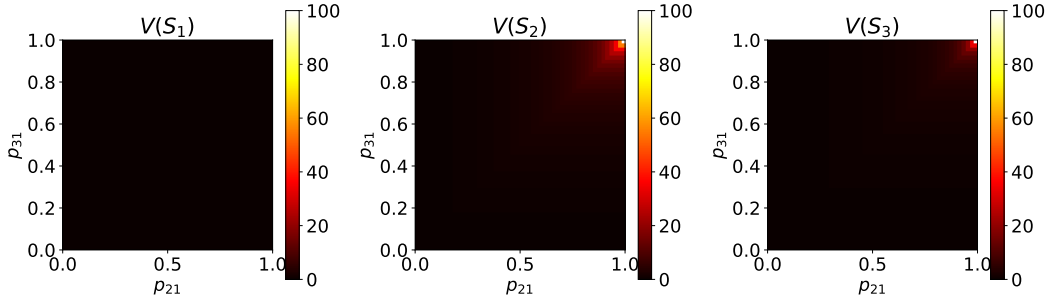


Figure 7. Value functions when $p_{11} = 1.0$

B. Proofs for State-Adversarial Markov Decision Process

Theorem 1 (Bellman Equations for fixed π and ν)**.** *Given $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ and $\nu : \mathcal{S} \to \mathcal{S}$, we have*

$$\tilde{V}_\nu^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p(s'|s,a) \left[R(s,a,s') + \gamma \tilde{V}_\nu^\pi(s')\right]$$
$$\tilde{Q}_\nu^\pi(s,a) = \sum_{s'\in\mathcal{S}} p(s'|s,a) \left[R(s,a,s') + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|\nu(s'))\tilde{Q}_\nu^\pi(s',a')\right].$$

Proof. Based on the definition of $\tilde{V}_\nu^\pi(s)$:

$$\begin{aligned}
\tilde{V}_\nu^\pi(s) &= \mathbb{E}_{\pi,\nu}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right] \\
&= \mathbb{E}_{\pi,\nu}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right] \\
&= \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r_{t+1} + \gamma \mathbb{E}_{\pi,\nu}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right]\right] \\
&= \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[R(s,a,s') + \gamma \tilde{V}_\nu^\pi(s')\right]
\end{aligned} \tag{16}$$

The recursion for $\tilde{Q}_\nu^\pi(s,a)$ can be derived similarly. Additionally, we note the following useful relationship between $\tilde{V}_\nu^\pi(s)$ and $\tilde{Q}_\nu^\pi(s,a)$:

$$\tilde{V}_\nu^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \tilde{Q}_\nu^\pi(s,a) \tag{17}$$

□

First we show that finding the optimal adversary ν^* given a fixed π for a SA-MDP can be cast into the problem of finding an optimal policy in a regular MDP.

Lemma 1 (Equivalence of finding optimal adversary in SA-MDP and finding optimal policy in MDP)**.** *Given a SA-MDP $M = (\mathcal{S}, \mathcal{A}, R, p)$ and a fixed policy π, there exists a MDP $\hat{M} = (\mathcal{S}, \hat{\mathcal{A}}, \hat{R}, \hat{p})$ such that the optimal policy of $\hat{M}$ is the optimal adversary ν for SA-MDP given the fixed π.*

Proof. For a SA-MDP $M = (\mathcal{S}, \mathcal{A}, R, p)$ and a fixed policy π, we define a regular MDP $\hat{M} = (\mathcal{S}, \hat{\mathcal{A}}, \hat{R}, \hat{p})$ such that $\hat{\mathcal{A}} = \mathcal{S}$, and ν is the policy for $\hat{M}$. At each state s, our policy ν gives a probability distribution $\nu(\cdot|s) \in \mathcal{P}(\hat{\mathcal{A}}) = \mathcal{P}(\mathcal{S})$ indicating that we perturb a state s to $\hat{s}$ with probability $\nu(\hat{s}|s)$ in the SA-MDP M.

For $\hat{M}$, the reward function is defined as:

$$\hat{R}(s,\hat{a},s') = \begin{cases} -\frac{\sum_{a\in\mathcal{A}} \pi(a|\hat{a}) p(s'|s,a) R(s,a,s')}{\sum_{a\in\mathcal{A}} \pi(a|\hat{a}) p(s'|a,s)} & \text{for } s,s' \in \mathcal{S} \text{ and } \hat{a} \in B(s) \subset \hat{\mathcal{A}} = \mathcal{S}, \\ -\infty & \text{for } s,s' \in \mathcal{S} \text{ and } \hat{a} \notin B(s). \end{cases} \tag{18}$$

The above definition is based on the following conditional probability which marginalizes π:

$$\begin{aligned}
p(r|s,\hat{a},s') &= \frac{p(r,s'|s,\hat{a})}{p(s'|s,\hat{a})} \\
&= \frac{\sum_a p(r,s'|a,s,\hat{a})\pi(a|s,\hat{a})}{\sum_a p(s'|a,s,\hat{a})\pi(a|s,\hat{a})} \\
&= \frac{\sum_a p(r,s'|a,s)\pi(a|\hat{a})}{\sum_a p(s'|a,s)\pi(a|\hat{a})} \\
&= \frac{\sum_a p(r|s',a,s)p(s'|a,s)\pi(a|\hat{a})}{\sum_a p(s'|a,s)\pi(a|\hat{a})}
\end{aligned}$$

Considering that $p(r = R(S, A, S')|s' = S', a = A, s = S) = 1.0$ and 0 otherwise, and taking an expectation over r yields the first case in (18). For $\hat{a} \notin B(s)$, we simply use a negative infinity reward to prevent the adversary taking that action.

The transition probability $\hat{p}$ is defined as

$$\hat{p}(s'|s, \hat{a}) = \sum_{a \in \mathcal{A}} \pi(a|\hat{a}) p(s'|s, a) \quad \text{for } s, s' \in \mathcal{S} \text{ and } \hat{a} \in \hat{\mathcal{A}} = \mathcal{S}.$$

Then we can get the value function $\hat{V}_\nu^\pi$ of this MDP for any policy ν not obtaining $-\infty$ reward (never taking an action $\hat{a} \notin B(s)$):

$$\begin{aligned}
\hat{V}_\nu^\pi(s) &:= \mathbb{E}_{\hat{p},\nu}\Big[\sum_{k=0}^{\infty} \gamma^k \hat{r}_{t+k+1} | s_t = s\Big] \\
&= \mathbb{E}_{\hat{p},\nu}\left[\hat{r}_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k \hat{r}_{t+k+2} | s_t = s\right] \\
&= \sum_{\hat{a} \in \mathcal{S}} \nu(\hat{a}|s) \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, \hat{a}) \left[\hat{R}(s, \hat{a}, s') + \gamma \mathbb{E}_{\hat{p},\nu}\left[\sum_{k=0}^{\infty} \gamma^k \hat{r}_{t+k+2} | s_{t+1} = s'\right]\right] \\
&= \sum_{\hat{a} \in \mathcal{S}} \nu(\hat{a}|s) \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, \hat{a}) \left[\hat{R}(s, \hat{a}, s') + \gamma \hat{V}_\nu^\pi(s')\right]
\end{aligned} \tag{19}$$

According to MDP theory (Puterman, 2014), we know that the $\hat{M}$ has an optimal policy ν^*, which satisfies $\hat{V}_{\nu^*}^\pi(s) \geq \hat{V}_\nu^\pi(s)$ for $\forall s, \forall \nu$. We also know that this ν^* is deterministic and assigns a unit mass probability for the optimal action in $B(s)$, because if a is not in $B(s)$ the reward is $-\infty$, and this policy cannot be an optimal policy.

So from now on in this proof we only study policies in $N := \{\nu : \forall s, \exists \hat{a} \in B(s), \nu(\hat{a}|s) = 1\}$. Note that all policies in N are deterministic and this class of policies consists ν^*. Also, N is consistent with the class of policies studied in Theorem 1. We denote the deterministic action $\hat{a}$ chosen by a $\nu \in N$ at s as $\nu(s)$. Then for $\forall \nu \in N$, we have

$$\begin{aligned}
\hat{V}_\nu^\pi(s) &= \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, \nu(s)) \left[\hat{R}(s, \hat{a}, s') + \gamma \hat{V}_\nu^\pi(s')\right] \\
&= \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a|\hat{a}) p(s'|s, a) \left[-\frac{\sum_{a \in \mathcal{A}} \pi(a|\hat{a}) p(s'|s, a) R(s, a, s')}{\sum_{a \in \mathcal{A}} \pi(a|\hat{a}) p(s'|a, s)} + \gamma \hat{V}_\nu^\pi(s')\right] \\
&= \sum_{a \in \mathcal{A}} \pi(a|\nu(s)) \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[-R(s, a, s') + \gamma \hat{V}_\nu^\pi(s')\right],
\end{aligned} \tag{20}$$

or

$$-\hat{V}_\nu^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|\nu(s)) \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[R(s, a, s') + \gamma(-\hat{V}_\nu^\pi(s'))\right]. \tag{21}$$

Comparing (21) and (16), we know that $-\hat{V}_\nu^\pi = \tilde{V}_\nu^\pi$ for any $\nu \in N$. Then the optimal value function $\hat{V}_{\nu^*}^\pi$ satisfies:

$$\begin{aligned}
\hat{V}_{\nu^*}^\pi(s) &= \max_{\hat{a} \in B(s)} \sum_{s' \in \mathcal{S}} \hat{p}(s'|s, \hat{a}) \left[\hat{R}(s, \hat{a}, s') + \gamma \hat{V}_\nu^\pi(s')\right] \\
&= \max_{s_\nu \in B(s)} \sum_{a \in \mathcal{A}} \pi(a|s_\nu) \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[-R(s, a, s') + \gamma \hat{V}_{\nu^*}^\pi(s')\right],
\end{aligned} \tag{22}$$

where we denote the action $\hat{a}$ taken at s as s_ν. So for ν^*, since $-\hat{V}_{\nu^*}^\pi = \tilde{V}_{\nu^*}^\pi$, we have

$$\tilde{V}_{\nu^*}^\pi(s) = \min_{\hat{a} \in B(s)} \sum_{a \in \mathcal{A}} \pi(a|\hat{a}) \sum_{s' \in \mathcal{S}} p(s'|s, a) \left[R(s, a, s') + \gamma \tilde{V}_{\nu^*}^\pi(s')\right], \tag{23}$$

and $\tilde{V}_{\nu^*}^\pi(s) \leq \tilde{V}_\nu^\pi(s)$ for $\forall s, \forall \nu \in N$. Hence ν^* is also the optimal ν for $\tilde{V}_\nu^\pi$. □

Lemma 1 gives many good properties for the optimal adversary. First, an optimal adversary always exists. Second, we do not need to consider stochastic adversaries as there always exists an optimal deterministic adversary. Additionally, showing Bellman contraction for finding the optimal adversary can be done similarly as in obtaining the optimal policy in a regular MDP, as shown in the proof of Theorem 2.

Theorem 2 (Bellman Contraction for Optimal Adversary)**.** *Define the Bellman operator* $\mathscr{L} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$,

$$(\mathscr{L}\tilde{V}^{\pi})(s) = \min_{s_\nu \in B(s)} \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a) \left[R(s,a,s') + \gamma \tilde{V}^{\pi}(s')\right]. \tag{24}$$

The Bellman equation for optimal adversary ν^* *can be written as:*

$$\tilde{V}^{\pi}_{\nu^*} = \mathscr{L}\tilde{V}^{\pi}_{\nu^*} \tag{25}$$

Additionally, $\mathscr{L}$ *is a contraction that converges to* $\tilde{V}^{\pi}_{\nu^*}$.

Proof. Based on Lemma 1, this proof is technically similar to the proof of "optimal Bellman equation" in regular MDPs, where $\max$ over π is replaced by $\min$ over ν. By the definition of $\tilde{V}^{\pi}_{\nu^*}(s)$,

$$\begin{aligned}
\tilde{V}^{\pi}_{\nu^*}(s) &= \min_{\nu} \tilde{V}^{\pi}_{\nu}(s) \\
&= \min_{\nu} \mathbb{E}_{\pi,\nu}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right] \\
&= \min_{\nu} \mathbb{E}_{\pi,\nu}\left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\right] \\
&= \min_{\nu} \sum_{a\in\mathcal{A}} \pi(a|\nu(s)) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r_{t+1} + \gamma \mathbb{E}_{\pi,\nu}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right]\right] \\
&= \min_{s_\nu \in B_\nu(s)} \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r_{t+1} + \gamma \min_{\nu} \mathbb{E}_{\pi,\nu}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\right]\right] \\
&= \min_{s_\nu \in B_\nu(s)} \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[r_{t+1} + \gamma \tilde{V}^{\pi}_{\nu^*}(s')\right]
\end{aligned}$$

This is the Bellman equation for the optimal adversary ν^*; ν^* is a fixed point of the Bellman operator $\mathscr{L}$.

Now we show the Bellman operator is a contraction. We have, if $\mathscr{L}\tilde{V}^{\pi}_{\nu_1}(s) \geq \mathscr{L}\tilde{V}^{\pi}_{\nu_2}(s)$,

$$\begin{aligned}
&\mathscr{L}\tilde{V}^{\pi}_{\nu_1}(s) - \mathscr{L}\tilde{V}^{\pi}_{\nu_2}(s) \\
&\leq \max_{s_\nu \in B_\nu(s)} \Big\{ \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[R(s,a,s') + \gamma \tilde{V}^{\pi}_{\nu_1}(s')\right] \\
&- \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a)\left[R(s,a,s') + \gamma \tilde{V}^{\pi}_{\nu_2}(s')\right] \Big\} \\
&= \gamma \max_{s_\nu \in B_\nu(s)} \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a)[\tilde{V}^{\pi}_{\nu_1}(s') - \tilde{V}^{\pi}_{\nu_2}(s')] \\
&\leq \gamma \max_{s_\nu \in B_\nu(s)} \sum_{a\in\mathcal{A}} \pi(a|s_\nu) \sum_{s'\in\mathcal{S}} p(s'|s,a) \|\tilde{V}^{\pi}_{\nu_1} - \tilde{V}^{\pi}_{\nu_2}\|_\infty \\
&= \gamma \|\tilde{V}^{\pi}_{\nu_1} - \tilde{V}^{\pi}_{\nu_2}\|_\infty
\end{aligned}$$

The first inequality comes from the fact that

$$\min_{x_1} f(x_1) - \min_{x_2} g(x_2) \leq f(x_2^*) - g(x_2^*) \leq \max_{x}(f(x) - g(x)),$$

where $x_2^* = \arg\min_{x_2} g(x_2)$. Similarly, we can prove $\mathscr{L}\tilde{V}^{\pi}_{\nu_2}(s) - \mathscr{L}\tilde{V}^{\pi}_{\nu_1}(s) \leq \|\tilde{V}^{\pi}_{\nu_1} - \tilde{V}^{\pi}_{\nu_2}\|_\infty$ if $\mathscr{L}\tilde{V}^{\pi}_{\nu_2}(s) > \mathscr{L}\tilde{V}^{\pi}_{\nu_1}(s)$. Hence

$$\|\mathscr{L}\tilde{V}^{\pi}_{\nu_1}(s) - \mathscr{L}\tilde{V}^{\pi}_{\nu_2}(s)\|_\infty = \max_{s} |\mathscr{L}\tilde{V}^{\pi}_{\nu_1}(s) - \mathscr{L}\tilde{V}^{\pi}_{\nu_2}(s)| \leq \gamma \|\tilde{V}^{\pi}_{\nu_1} - \tilde{V}^{\pi}_{\nu_2}\|_\infty.$$

Then according to the Banach fixed-point theorem, since $0 < \gamma < 1$, $\tilde{V}^{\pi}_{\nu}$ converges to a unique fixed point, and this fixed point is $\tilde{V}^{\pi}_{\nu^*}$.

□

Algorithm 1 Policy Evaluation for SA-MDP

Input: Policy π, convergence threshold ε
Output: Values for policy π, $V^{\pi}_{\nu^*}(s)$
  Initialize $V(s) \leftarrow 0$ for all $s \in \mathcal{S}$
  repeat
    $\Delta \leftarrow 0$
    for all $s \in \mathcal{S}$ **do**
      $v \leftarrow \infty, v_0 \leftarrow V(s)$
      for all $s_\nu \in B(s)$ **do**
        $v' \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s_\nu) \sum_{s' \in \mathcal{S}} p(s'|s,a) \cdot \Big[R(s,a,s') + \gamma \tilde{V}^{\pi}(s') \Big]$
        $v \leftarrow \min(v, v')$
      $V(s) \leftarrow v$
      $\Delta \leftarrow \max(\Delta, |v_0 - V(s)|)$
  until $\Delta < \varepsilon$
  $V^{\pi}_{\nu^*}(s) \leftarrow V(s)$

A direct consequence of Theorem 2 is the policy evaluation algorithm (Algorithm 1) for SA-MDP, which obtains the values for each state under optimal adversary for a fixed policy π. For both Lemma 1 and Theorem 2, we only consider a fixed policy π, and in this setting finding an optimal adversary is not difficult. However, finding an optimal π under the optimal adversary is more challenging, as we can see in Section A, given the white-box attack setting where the adversary knows π and can choose optimal perturbations accordingly, an optimal policy for MDP can only receive zero rewards under optimal adversary. We now show two intriguing properties for optimal policies in SA-MDP:

Theorem 3. *There exists a SA-MDP and some stochastic policy $\pi \in \Pi_{MR}$ such that we cannot find a better deterministic policy $\pi' \in \Pi_{MD}$ satisfying $\tilde{V}^{\pi'}_{\nu^*(\pi')}(s) \geq \tilde{V}^{\pi}_{\nu^*(\pi)}(s)$ for all $s \in \mathcal{S}$.*

Proof. Proof by giving a counter example that no deterministic policy can be better than a random policy. The SA-MDP example in section A provided such a counter example: all 8 possible deterministic policies are no better than the stochastic policy $p_{11} = p_{21} = p_{31} = 0.5$. □

Theorem 4. *Under the optimal adversary ν^*, an optimal policy $\pi^* \in \Pi_{MR}$ does not necessarily exist for SA-MDP.*

Proof. The SA-MDP example in section A does not have an optimal policy. For π_1 where $p_{11} = p_{21} = p_{31} = 1$ we have $V^{\pi_1}(S_1) = 0, V^{\pi_1}(S_2) = V^{\pi_1}(S_3) = 100$. This policy is not an optimal policy since we have π_2 where $p_{11} = p_{21} = p_{31} = 0.5$ that can achieve $V^{\pi_2}(S_1) = V^{\pi_2}(S_2) = V^{\pi_2}(S_3) = 50$ and $V^{\pi_2}(S_1) > V^{\pi_1}(S_1)$.

An optimal policy π, if exists, must be better than π_1 and have $V^{\pi}(S_1) > 0, V^{\pi}(S_2) = V^{\pi}(S_3) = 100$. In order to achieve $V^{\pi}(S_2) = V^{\pi}(S_3) = 100$, we must set $p_{21} = p_{31} = 1$ since it is the only possible way to start from S_2 and S_3 and receive +1 reward for every step. We can still change p_{11} to probabilities other than 1, however if $p_{11} < 1$ the adversary can set $\nu(S_2) = \nu(S_3) = S_1$ and reduce $V^{\pi}(S_2)$ and $V^{\pi}(S_3)$. Thus, no policy better than π_1 exists, and since π_1 is not an optimal policy, no optimal policy exists. □

Theorem 3 and Theorem 4 show that the classic definition of optimality is probably not suitable for SA-MDP. Further works can study how to obtain optimal policies for SA-MDP under some alternative definition of optimality.

Theorem 5. *Given a policy π for a non-adversarial MDP. Under the optimal adversary ν in SA-MDP, for all $s \in \mathcal{S}$ we have*

$$\max_{s \in \mathcal{S}} \left\{ V^{\pi}(s) - \tilde{V}^{\pi}_{\nu^*}(s) \right\} \leq \alpha \max_{s \in \mathcal{S}} \max_{\hat{s} \in B(s)} \mathrm{D}_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s})) \tag{26}$$

where $\mathrm{D}_{TV}(\pi(\cdot|s), \pi(\cdot|\hat{s}))$ *is the total variation distance between* $\pi(\cdot|s)$ *and* $\pi(\cdot|\hat{s})$*, and*

$$\alpha := 2[1 + \frac{\gamma}{(1-\gamma)^2}] \max_{(s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}} |R(s,a,s')|$$

is a constant that does not depend on π*.*

Proof. Our proof is based on Theorem 1 in Achiam et al. (2017). In fact, many works in the literature have proved similar results under different scenarios (Kakade & Langford, 2002; Pirotta et al., 2013). For an arbitrary starting state s_0 and two arbitrary policies π and π', Theorem 1 in Achiam et al. (2017) gives an upper bound of $V^{\pi}(s_0) - V^{\pi'}(s_0)$. The bound is given by

$$V^{\pi}(s_0) - V^{\pi'}(s_0) \leq -\mathbb{E}_{\substack{s \sim d^{\pi}_{s_0} \\ a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|a,s)}} \Big[\big(\frac{\pi'(a|s)}{\pi(a|s)} - 1 \big) R(s,a,s') \Big] + \frac{2\gamma}{(1-\gamma)^2} \max_s \Big\{ \mathbb{E}_{\substack{a \sim \pi'(\cdot|s) \\ s' \sim p(\cdot|a,s)}} \big[R(s,a,s') \big] \Big\} \mathbb{E}_{s \sim d^{\pi}_{s_0}} \big[\mathrm{D}_{TV}(\pi(\cdot|s), \pi'(\cdot|s)) \big], \tag{27}$$

where $d^{\pi}_{s_0}$ is the discounted future state distribution from s_0, defined as

$$d^{\pi}_{s_0}(s) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}(s_t = s | \pi, s_0). \tag{28}$$

Note that in Theorem 1 of Achiam et al. (2017), the author proved a general form with an arbitrary function f and we assume $f \equiv 0$ in our proof. We also assume the starting state is deterministic, so J^{π} in Achiam et al. (2017) is replaced by $V^{\pi}(s_0)$. Then we simply need to bound both terms on the right hand side of (27).

For the first term we know that

$$\begin{aligned}
-\mathbb{E}_{\substack{s \sim d^{\pi}_{s_0} \\ a \sim \pi(\cdot|s) \\ s' \sim p(\cdot|a,s)}} \Big[\big(\frac{\pi'(a|s)}{\pi(a|s)} - 1 \big) R(s,a,s') \Big] &= \sum_s d^{\pi}_{s_0}(s) \sum_a \big[\pi(a|s) - \pi'(a|s) \big] \sum_{s'} p(s'|s,a) R(s,a,s') \\
&\leq \sum_s d^{\pi}_{s_0}(s) \sum_a \big| \pi(a|s) - \pi'(a|s) \big| \Big| \sum_{s'} p(s'|s,a) R(s,a,s') \Big| \\
&\leq \max_{s,a,s'} |R(s,a,s')| \max_s \Big\{ \sum_a \big| \pi(a|s) - \pi'(a|s) \big| \Big\} \\
&= 2 \max_{s,a,s'} |R(s,a,s')| \max_s \mathrm{D}_{TV}(\pi(\cdot|s), \pi'(\cdot|s))
\end{aligned} \tag{29}$$

The second term is bounded by

$$\frac{2\gamma}{(1-\gamma)^2} \max_s \Big\{ \mathbb{E}_{\substack{a \sim \pi'(\cdot|s) \\ s' \sim p(\cdot|a,s)}} \big[R(s,a,s') \big] \Big\} \mathbb{E}_{s \sim d^{\pi}_{s_0}} \big[\mathrm{D}_{TV}(\pi(\cdot|s), \pi'(\cdot|s)) \big] \leq \frac{2\gamma}{(1-\gamma)^2} \max_{s,a,s'} |R(s,a,s')| \max_s \mathrm{D}_{TV}(\pi(\cdot|s), \pi'(\cdot|s)) \tag{30}$$

Therefore, the RHS of (27) is bounded by $\alpha \max_s \mathrm{D}_{TV}(\pi(\cdot|s), \pi'(\cdot|s))$, where

$$\alpha = 2[1 + \frac{\gamma}{(1-\gamma)^2}] \max_{s,a,s'} |R(s,a,s')| \tag{31}$$

Finally, we simply let $\pi'(\cdot|s) := \pi(\cdot|\nu^*(s))$ and the proof is complete. □

Before proving 6 we first give a technical lemma about the total variation distance between two multi-variate Gaussian distributions with the same variance.

Lemma 2. *Given two multi-variate Gaussian distributions* $X_1 \sim \mathcal{N}(\mu_1, \sigma^2 I_n)$ *and* $X_2 \sim \mathcal{N}(\mu_2, \sigma^2 I_n)$*,* $\mu_1, \mu_2 \in \mathbb{R}^n$*, define* $d = \|\mu_2 - \mu_1\|_2$*. We have* $\mathrm{D}_{TV}(X_1, X_2) = \sqrt{\frac{2}{\pi}} \frac{d}{\sigma} + O(d^3)$.

Proof. Denote probability density of X_1 and X_2 as f_1 and f_2, and denote $a = \frac{\mu_2-\mu_1}{d}$ as the normal vector of the perpendicular bisector line between μ_1 and μ_2. Due to the symmetry of Gaussian distribution, $f_1(x) - f_2(x)$ is positive for all x where $a^\top x - a^\top \mu_1 - \frac{d}{2} > 0$ and negative for all x on the other symmetric side. When $a^\top x - a^\top \mu_1 - \frac{d}{2} > 0$, $\int_{x\in\mathbb{R}^n}[f_1(x) - f_2(x)]\mathrm{d}x = \Phi(\frac{d}{2\sigma}) - (1 - \Phi(\frac{d}{2\sigma})) = 2\Phi(\frac{d}{2\sigma}) - 1$. Thus,

$$\begin{aligned} D_{TV}(X_1, X_2) &= \int_{x\in\mathbb{R}^n} |f_1(x) - f_2(x)|\mathrm{d}x \\ &= 2\int_{a^\top x - a^\top \mu_1 - \frac{d}{2} > 0} (f_1(x) - f_2(x))\mathrm{d}x \\ &= 2(\Phi(\frac{d}{2\sigma}) - (1 - \Phi(\frac{d}{2\sigma}))) \\ &= 2(2\Phi(\frac{d}{2\sigma}) - 1) \end{aligned}$$

Then we use the Taylor series for $\Phi(x)$ at $x = 0$:

$$\Phi(x) = \frac{1}{2} + \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2^n n!(2n+1)}$$

Since we consider the case where d is small, we only keep the first order term and obtain:

$$D_{TV}(X_1, X_2) = \sqrt{\frac{2}{\pi}} \frac{d}{\sigma} + O(d^3)$$

□

Theorem 6. $\mathrm{D}_{TV}(\bar{\pi}(\cdot|s), \bar{\pi}(\cdot|\hat{s})) = \sqrt{\frac{2}{\pi}} \frac{d}{\sigma} + O(d^3)$, *where* $d = \|\pi(s) - \pi(\hat{s})\|_2$.

Proof. This theorem is a special case of Lemma 2 where $X_1 = \bar{\pi}(\cdot|s)$, $X_2 = \bar{\pi}(\cdot|s')$ and $X_1 \sim \mathcal{N}(\pi(s), \sigma^2 I)$, $X_2 \sim \mathcal{N}(\pi(s'), \sigma^2 I)$. □

C. Additional details for SA-DQN

Algorithm We present the SA-DQN training algorithm in Algorithm 2. The main difference comparing to regular DQN is the additional hinge loss term $\tilde{L}$, which encourage the network not to change its output under perturbations on the state observation.

Hyperparameters for Regular DQN training. For Atari games, the deep Q networks have 3 CNN layers followed by 2 fully connected layers. The first CNN layer has 32 channels, a kernel size of 8, and stride 4. The second CNN layer has 64 channels, a kernel size of 4, and stride 2. The third CNN layer has 64 channels, a kernel size of 3, and stride 1. The fully connected layers have 512 hidden neurons. For Atari games, we run each environment for 6×10^6 steps without framestack. For Acrobot, the deep Q network is a 3-layer MLP with [128,128] hidden neorons and we run 6×10^5 steps. The learning rate is 1×10^{-3} for Acrobot, 1×10^{-5} for BankHeist, 2×10^{-5} for RoadRunner, and 6.25×10^{-5} for Pong and Freeway. For all environments, no reward scaling is used, and discount factor is set to 0.99. For all Atari environments, we use a replay buffer with a capacity of 2×10^5 and for Acrobot, the capacity is reduced to 2×10^4. Prioritized replay buffer sampling is used with $\alpha = 0.5$ and β increased from 0.4 to 1 linearly in 6×10^5 for Acrobot and 4×10^6 steps for Atari games. A batch size of 32 is used and the target network is updated every 2000 steps. We use Huber loss in TD-loss.

Hyperparameters for SA-DQN training. SA-DQN uses the same network structure and hyperparameters as in DQN training, except that for Freewaay and BankHeist, we increase the schedule length of buffer's β to 6×10^6. For the additional regularization parameter κ for robustness, we choose $\kappa \in \{0.005, 0.01, 0.02\}$. The total number of SA-DQN training steps in all environments are the same as those in DQN. For Pong and RoadRunner, we train the Q network without regularization

Algorithm 2 State-Adversarial Deep Q-Learning (SA-DQN)

Initialize current Q network $Q(s,a)$ with parameters θ.
Initialize target Q network $Q'(s,a)$ with parameters $\theta' \leftarrow \theta$.
Initial replay buffer $\mathcal{B}$
for $t = 1$ to T **do**
  With probability ϵ_t select a random action at a_t, otherwise select $a_t = \arg\max_a Q_\theta(s_t, a; \theta)$
  Execute action a_t in environment and observe reward r_t and state s_{t+1}
  Store transition $\{s_t, a_t, r_t, s_{t+1}\}$ in $\mathcal{B}$.
  Randomly sample a minibatch of N samples $\{s_i, a_i, r_i, s'_i\}$ from $\mathcal{B}$.
  For all s_i, compute $a_i^* = \arg\max_a Q_\theta(s_i, a; \theta)$.
  Set $y_i = r_i + \gamma \max_{a'} Q'_{\theta'}(s'_i, a'; \theta)$ for non-terminal s_i, and $y_i = r_i$ for terminal s_i.
  Compute TD-loss for each transition: TD-$L(s_i, a_i, s'_i; \theta) = \text{Huber}(y_i - Q_\theta(s_i, a_i^*; \theta))$
  For all s_i and all $a_i \neq a_j^*$, obtain lower bounds on $Q_\theta(s, a_i^*; \theta) - Q_\theta(s, a_i; \theta)$ using CROWN-IBP: $Q_\theta(s, a_i^*; \theta) - Q_\theta(s, a_i; \theta) \geq l_{Q_\theta, a_i^*, a_i}(s_i)$ for all $s \in B(s_i)$.
  Compute hinge loss for each s_i: $\tilde{L}(s_i) = \max\left\{c, -\min_{a_i \neq a_i^*}\{l_{Q_\theta, a_i^*, a_i}(s_i)\}\right\}$
  Perform a gradient descent step to minimize $\frac{1}{N}\sum_i$ TD-$L(s_i, a_i, s'_i; \theta) + \kappa \tilde{L}(s_i; \theta)$.
  Update Target Network every M steps: $\theta' \leftarrow \theta$.

for the first 1.5×10^6 steps, then increase ϵ from 0 to the target value in 2×10^6 steps, and then keep training at the target ϵ for 2.5×10^6 steps. For Freeway and BankHeist, this ϵ schedule starts at 1×10^6th step with a length of 4×10^6 steps. For Acrobot, this ϵ schedule starts at 2×10^4th step with a length of 1×10^5 steps. The hyper-parameter β in CROWN-IBP is scheduled as suggested in (Zhang et al., 2020): β starts from 1 when ϵ schedule starts and linearly decays to 0 when ϵ reaches the target value. The confidence constant of hinge loss is 1 in Atari environments and is 0.01 in Acrobot.

D. Additional details for SA-DDPG

Algorithm We present the SA-DDPG training algorithm in Algorithm 3. The main difference comparing to regular DDPG is the additional loss term $L_{SA}(\theta_\pi)$, which provides an upper bound on $\max_{s \in B(s_i)} \|\pi(s) - \pi(s_i)\|_2^2$. If this term is small, according to Theorem 6 and Theorem 5 we can bound the performance loss under adversary.

Algorithm 3 State-Adversarial Deep Deterministic Policy Gradient (SA-DDPG)

Initialize actor network $\pi(s)$ and critic network $Q(s,a)$ with parameter θ_π and θ_Q
Initialize target network $\pi'(s)$ and critic network $Q'(s,a)$ with weights $\theta_{\pi'} \leftarrow \theta_\pi$ and $\theta_{Q'} \leftarrow \theta_Q$
Initial replay buffer $\mathcal{B}$
for $t = 1$ to T **do**
  Initial a random process $\mathcal{N}$ for action exploration
  Choose action $a_t \sim \pi(s_t) + \epsilon$, $\epsilon \sim \mathcal{N}$
  Observe reward r_t, next state s_{t+1} from environment
  Store transition $\{s_t, a_t, r_t, s_{t+1}\}$ into $\mathcal{B}$
  Sample a mini-batch of N samples $\{s_i, a_i, r_i, s'_i\}$ from $\mathcal{B}$
  $y_i \leftarrow r_i + \gamma Q'(s'_i, \pi'(s'_i))$ for all $i \in [N]$
  Update θ_Q by minimizing loss $L(\theta_Q) = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i))^2$
  Obtain upper and lower bounds on $\pi(s_i)$ using CROWN-IBP: $l_{\theta_\pi}(s_i) \leq \pi(s) \leq u_{\theta_\pi}(s_i)$ for $s \in B(s_i)$
  Upper bound on ℓ_2 distance: $L_{SA}(\theta_\pi) := \frac{1}{N}\sum_i \|u_{\theta_\pi}(s_i) - l_{\theta_\pi}(s_i)\|_2^2$
  Update θ_π using deterministic policy gradient and gradient of $L_{SA}(\theta_\pi)$:
  $\nabla_{\theta_\pi} J(\theta_\pi) = \frac{1}{N}\sum_i \left[\nabla_a Q(s,a)|_{s=s_i, a=\pi(s_i)} \nabla_{\theta_\pi} \pi(s)|_{s=s_i} + \kappa \nabla_{\theta_\pi} L_{SA}(\theta_\pi)\right]$
  Update Target Network:
  $\theta_{Q'} \leftarrow \tau\theta_Q + (1-\tau)\theta_{Q'}$
  $\theta_{\pi'} \leftarrow \tau\theta_\pi + (1-\tau)\theta_{\pi'}$

Hyperparameters for Regular DDPG training. Both actor and critic networks are 3-layer MLPs with $[400, 300]$ hidden neurons. We run each environment for 2×10^6 steps. Actor network learning rate is 1×10^{-4} and critic network learning rate is 1×10^{-3} (except that for Hopper-v2 the learning rate is reduced to 1×10^{-4} due to the larger values of rewards); both networks are optimized using Adam optimizer. No reward scaling is used, and discount factor is set to 0.99. We use a replay buffer with a capacity of 1×10^6 items and we do not use prioritized replay buffer sampling. For the random process $\mathcal{N}$ used for exploration, we use a Ornstein-Uhlenbeck process with $\theta = 0.15$ and $\sigma = 0.2$. The mixing parameter of current and target actor and critic networks is set to $\tau = 0.001$.

Hyperparameters for SA-DDPG training. SA-DDPG uses the same hyperparameters as in DDPG training. For the additional regularization parameter κ for $\pi(s)$, we choose $\kappa \in \{0.3, 1.0, 3.0, 10.0, 30.0\}$. We train the actor network without actor regularization for the first 1×10^6 steps, then increase ϵ from 0 to the target value in 5×10^5 steps, and then keep training at the target ϵ for 5×10^5 steps. The hyper-parameter β in CROWN-IBP is scheduled as suggested in (Zhang et al., 2020): β starts from 1 when ϵ schedule starts and linearly decays to 0 when ϵ reaches the target value. The total number of training steps is thus 2×10^6, the same as regular DDPG training. The target ϵ values for each task is listed in Table 2. Note that we rescale ϵ by the standard deviations of each state variable. The standard deviations are calculated using data collected on baseline policy without adversary.