Certified Adversarial Robustness for Deep Reinforcement Learning

Michael Everett*, Björn Lütjens*, and Jonathan P. How

***Abstract*—Deep Neural Network-based systems are now the state-of-the-art in many robotics tasks, but their application in safety-critical domains remains dangerous without formal guarantees on network robustness. Small perturbations to sensor inputs (from noise or adversarial examples) are often enough to change network-based decisions, which was recently shown to cause an autonomous vehicle to swerve into another lane. In light of these dangers, numerous algorithms have been developed as defensive mechanisms from these adversarial inputs, some of which provide formal robustness guarantees or certificates. This work leverages research on certified adversarial robustness to develop an online certified defense for deep reinforcement learning algorithms. The proposed defense computes guaranteed lower bounds on state-action values during execution to identify and choose a robust action under a worst-case deviation in input space due to possible adversaries or noise. The approach is demonstrated on a Deep Q-Network policy and is shown to increase robustness to noise and adversaries in pedestrian collision avoidance scenarios and a classic control task. This work extends our previous paper with new performance guarantees, expanded results aggregated across more scenarios, an extension into scenarios with adversarial behavior, comparisons with a more computationally expensive method, and visualizations that provide intuition about the robustness algorithm.**

***Index Terms*—Adversarial Attacks, Reinforcement Learning, Collision Avoidance, Robustness Verification**

I. Introduction

Deep reinforcement learning (RL) algorithms have achieved impressive success on robotic manipulation [2] and robot navigation in pedestrian crowds [3], [4]. Many of these systems utilize black-box predictions from deep neural networks (DNNs) to achieve state-of-the-art performance in prediction and planning tasks. However, the lack of formal robustness guarantees for DNNs currently limits their application in safety-critical domains, such as collision avoidance. In particular, even subtle perturbations to the input, known as *adversarial examples*, can lead to incorrect (but highly-confident) decisions by DNNs [5]–[7]. Furthermore, several recent works have demonstrated the danger of adversarial examples in real-world situations [8], [9], including causing an autonomous vehicle to swerve into another lane [10]. The work in this paper addresses the lack of robustness against adversarial examples and sensor noise by proposing an online certified defense to add to existing deep RL algorithms during execution.

Existing methods to defend against adversaries, such as adversarial training [11]–[14], defensive distillation [15], or model ensembles [16] do not provide theoretical guarantees for reliably improving the robustness and are often ineffective against more advanced adversarial attacks [17]–[20]. Several methods provide formal guarantees on the robustness of a given network, but finding the guarantees is an NP-complete problem and is computationally intractable to solve in real-time for applications like robot manipulation or navigation [21]–[26]. *Robustness certification* methods relax the problem to make it tractable. Given an adversarial distortion of a nominal input, instead of finding exact bounds on the worst-case output deviation, these methods efficiently find certified lower bounds [27]–[29]. In particular, the work by [29] runs in real-time for small networks (33 to $14,000$ times faster than verification methods), its bound has been shown to be within 10% of the true bound, and the idea compatible with many activation functions [30] and neural network architectures [31]. These certification methods were previously applied on computer vision/supervised learning tasks.

This work extends the tools for robustness certification against adversaries to deep RL tasks. Previous works in robust computer vision address the *verification problem*: what is the maximum input perturbation for which the network returns the same decision? In RL, this analysis would simply *mark* a nominal action as non-robust if the input perturbation exceeds a robustness threshold (e.g., the system's known level of uncertainty), but would not reason about alternative actions. Hence, we focus on the *robust decision making problem*: given a known bound on the input perturbation, what is the best action to take? This aligns with the requirement that an agent *must* select an action at each step of an RL problem.

As a motivating example, consider the collision avoidance setting in Fig. 1, in which an adversary perturbs an agent's (orange) observation of an obstacle (blue). An agent following a nominal/standard deep RL policy would observe $\boldsymbol{s}_{adv}$ and select an action, a^*_{nom}, that collides with the obstacle's true position, $\boldsymbol{s}_0$, thinking that the space is unoccupied. Our proposed approach assumes a worst-case deviation of the observed input, $\boldsymbol{s}_{adv}$, bounded by $\boldsymbol{\epsilon}$, and takes the robust-optimal action, a^*_{adv}, under that perturbation, to safely avoid the true obstacle. Nominal robustness certification algorithms assume ϵ is a scalar, which makes sense for image inputs (all pixels have same scale, e.g., $0-255$ intensity). A key challenge in direct application to RL tasks is that the observation vector (network input) could have elements with substantially different scales (e.g., position, angle, joint torques) and associated measurement uncertainties, motivating our extension with vector $\boldsymbol{\epsilon}$.

Authors are with the Aerospace Controls Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 02139 USA e-mail: mfe@mit.edu

Preliminary version of paper presented at Conference on Robot Learning (CoRL) on Nov. 1, 2019 [1]. This manuscript submitted Mar. 28, 2020.

* indicates equal contributions

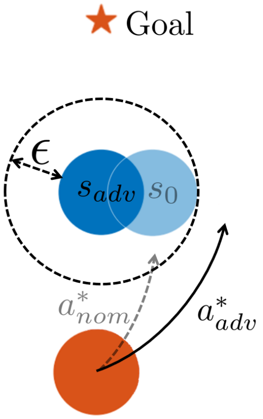


Fig. 1: Intuition. An adversary distorts the true position, $\boldsymbol{s}_0$, of a dynamic obstacle (blue) into an adversarial observation, $\boldsymbol{s}_{adv}$. The agent (orange) only sees the adversarial input, so nominal RL policies would take a^*_{nom} to reach the goal quickly, but would then collide with the true obstacle, $\boldsymbol{s}_0$. The proposed defensive strategy considers that $\boldsymbol{s}_0$ could be anywhere inside the $\boldsymbol{\epsilon}$-ball around $\boldsymbol{s}_{adv}$, and selects the action, a^*_{adv}, with the best, worst-case outcome as calculated by a guaranteed lower bound on the value network output, which cautiously avoids the obstacle while reaching the goal. Note this is different from simply inflating the obstacle radius, since the action values contain information about environment dynamics, e.g., blue agent's cooperativeness.

This work contributes (i) the first formulation of robustness certification in deep RL problems, (ii) an extension of existing robustness certification algorithms to variable scale inputs, (iii) an optimal action selection rule under worst-case state perturbations, and (iv) demonstrations of increased robustness to adversaries and sensor noise on cartpole and a pedestrian collision avoidance simulation.

We extend our previously published conference paper [1] with new performance guarantees (Section IV-D), expanded results aggregated across more scenarios (Sections V-A and V-B), an extension of the algorithm into scenarios with adversarial behavior (Section V-D), comparisons with a more computationally expensive method (Section V-E), and visualizations that provide intuition about the robustness algorithm (Section V-F).

II. Related work

The lack of robustness of DNNs to real-world uncertainties [5] has motivated the study of adversarial attacks (i.e., worst-case uncertainty realizations) in many learning tasks. This section summarizes adversarial attack and defense models in deep RL (see [32] for a thorough survey) and describes methods for formally quantifying DNN robustness.

A. Adversarial Attacks in Deep RL

An adversary can act against an RL agent by influencing (or exploiting a weakness in) the observation or transition models of the environment.

Observation model: While many of the techniques for attacking supervised learning networks through small image perturbations [33] could be used to attack image-based deep RL policies, recent works show how to specifically craft adversarial attacks (in the input image space) against a Deep Q-Network (DQN) in RL [34], [35]. Another work applies both adversarial observation and transition perturbations [36].

Transition model: Several approaches attack an RL agent by changing parameters of the physics simulator, like friction coefficient or center of gravity, between episodes [37], [38]. Other approaches change the transition model between steps of episodes, for instance by applying disturbance forces [39], [40], or by adding a second agent that is competing against the ego agent [41]. In [42], the second agent unexpectedly learns to visually distract the ego agent rather than exerting forces and essentially becomes an observation model adversary. Thus, adversarial behavior of a second agent (the topic of *multiagent games* [43]) introduces complexities beyond the scope of this work (except a brief discussion in Section V-D), which focuses on robustness to adversarial observations.

B. Empirical Defenses to Adversarial Attacks

Supervised Learning: To mitigate the impact of adversarial attacks, several works detect or defend against adversarial examples for supervised learning tasks. Adversarial training or retraining augments the training dataset with adversaries [11]–[14] to increase robustness during testing (empirically). Other works increase robustness through distilling networks [15], comparing the output of model ensembles [16], or detect adversarial examples through comparing the input with a binary filtered transformation of the input [44]. Although these approaches show impressive empirical success, they are often ineffective against more sophisticated adversarial attacks [17]–[20].

Deep RL: Many ideas from supervised learning were transferred over to deep RL to provide empirical defenses to adversaries (e.g., training in adversarial environments [36]–[40], using model ensembles [37], [38]). Moreover, because adversarial observation perturbations are a form of measurement uncertainty, there are close connections between Safe RL [45] and adversarial robustness. Many Safe RL (also called risk-sensitive RL) algorithms optimize for the reward under *worst-case* assumptions of environment stochasticity, rather than optimizing for the expected reward [46]–[48]. The resulting policies are more risk-sensitive (i.e., robust to stochastic deviations in the input space, such as sensor noise), but could still fail on algorithmically crafted adversarial examples.

Rather than modifying the RL training process, this work adds a defense layer on top of an already trained DQN, by assuming a worst-case deviation of the input space inside some bounds and taking the action with maximum expected reward. To guarantee that the add-on defense tool leads to robustness at test time, this work leverages methods to propagate known DNN input bounds to guaranteed output bounds.

C. Formal Robustness Methods

Methods that formally solve the verification problem provide these desirable theoretical guarantees for DNN outputs, but have not yet been formulated to solve the robust decision making problem, relevant for RL tasks. *Exact methods* find

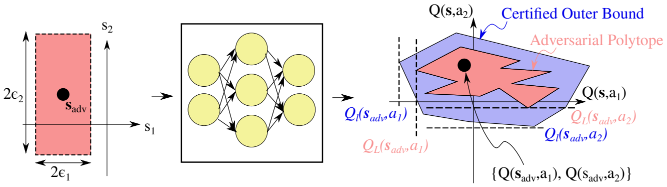


Fig. 2: State Uncertainty Propagated Through Deep Q-Network. The red region (left) represents bounded state uncertainty (an L_∞ $\boldsymbol{\epsilon}$-ball) around the observed state, $\boldsymbol{s}_{adv}$. A neural network maps this set of possible inputs to a polytope (red) of possible outputs (Q-values in RL). This work's extension of [29] provides a certified outer bound (blue) on that polytope. This work then modifies the RL action-selection rule by considering lower bounds, Q_l, on the blue region for each action. In this 2-state, 2-action example, our algorithm would select action 2, since $Q_l(\boldsymbol{s}_{adv}, a_2) > Q_l(\boldsymbol{s}_{adv}, a_1)$, i.e. the worst possible outcome from action 2 is better than the worst possible outcome from action 1, given an $\boldsymbol{\epsilon}$-ball uncertainty and a pre-trained DQN.

bounds on the maximum output deviation, given a bounded input perturbation. These methods rely on satisfiability modulo theory (SMT) [21]–[23], linear programming (LP) or mixed-integer LP solvers [24], [25], or zonotopes [26], to propagate constraints on the input space through to the output space (exactly). The difficulty in this propagation arises through DNNs with ReLU or other nonlinear activation functions – in fact, the problem of finding the exact output bounds is NP-complete [22], [29] and thus currently infeasible to be run online on a robot.

The formal robustness approaches can be visualized in Fig. 2 in terms of a 2-state, 2-action deep RL problem. For a given state uncertainty (red, left), the exact methods reason over the exact adversarial polytope (red, right), i.e., all outputs the network could produce. A convex relaxation [27]–[29] of the nonlinear activations provides convex, guaranteed outer bounds, Q_{LP}, on the adversarial polytope. Even though this LP can be solved more efficiently than the exact forms (such as [24] mentioned above), solving the relaxed LP is still computationally intensive. Fortunately, finding a feasible solution to the dual problem is computationally lightweight and provides a certified (i.e., guaranteed, but not necessarily tight; see blue region in Fig. 2) outer bound, Q_l, on the LP solution and thus, also on the adversarial polytope. One of these methods, Fast-Lin [29] (which this work extends), certifies image classification (MNIST) networks in $< 200ms$ with provable guarantees.

Each of [21]–[29] solves the verification problem: determine whether the calculated set of possible network outputs crosses a decision hyperplane (a line of slope 1 through the origin, in this example) – if it does cross, the classifier is deemed "not robust" to the input uncertainty. Our approach instead solves the robust decision making problem: determine the best action considering worst-case outcomes, $Q_l(\boldsymbol{s}_{adv}, a_1), Q_l(\boldsymbol{s}_{adv}, a_2)$, denoted by the dashed lines in Fig. 2.

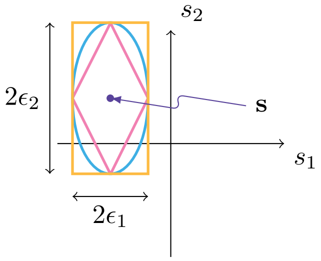


Fig. 3: Illustration of $\boldsymbol{\epsilon}$-Ball, $\mathcal{B}_p(\boldsymbol{s}, \boldsymbol{\epsilon})$, for $n = 2$. Let $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2]$. Depending on the choice of L_p norm (e.g., L_1, L_2, and L_∞), $\mathcal{B}_p(\boldsymbol{s}, \boldsymbol{\epsilon})$ is the set of points inside the corresponding colored outline. The adversary can perturb nominal observation $\boldsymbol{s}$ to any point inside $\mathcal{B}_p(\boldsymbol{s}, \boldsymbol{\epsilon})$. The values of $\{n, \boldsymbol{\epsilon}, p\}$ are application-specific choices and the components of $\boldsymbol{\epsilon}$ need not be equal.

III. Background

A. Preliminaries

In RL problems[1], the state-action value (or, "Q-value") $Q(\boldsymbol{s}, a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(t) | \mathrm{s}(t{=}0) = \boldsymbol{s}, \mathrm{a}(t{=}0) = a\right]$ expresses the expected accumulation of future reward, r, discounted by γ, received by starting in state $\boldsymbol{s} \in \mathbb{R}^n$ and taking one of d discrete actions, $a \in \{a_0, a_1, \ldots, a_{d-1}\}$.

Let $\boldsymbol{\epsilon} \in \mathbb{R}^n_{\geq 0}$ be the maximum element-wise deviation of the state vector, and let $1 \leq p \leq \infty$ parameterize the L_p-norm. We define the set of states within this deviation as the $\boldsymbol{\epsilon}$-Ball,

$$\mathcal{B}_p(\boldsymbol{s}_0, \boldsymbol{\epsilon}) = \{\boldsymbol{s} : \lim_{\boldsymbol{\epsilon}' \to \boldsymbol{\epsilon}^+} ||(\boldsymbol{s} - \boldsymbol{s}_0) \oslash \boldsymbol{\epsilon}'||_p \leq 1\}, \qquad (1)$$

where $\oslash$ denotes element-wise division, and the $\lim$ is only needed to handle the case where $\exists i \ \epsilon_i = 0$ (e.g., when the adversary is not allowed to perturb some component of the state, or the agent knows some component of the state vector with zero uncertainty).

An $\boldsymbol{\epsilon}$-Ball is illustrated in Fig. 3 for the case of $n = 2$, highlighting that different elements of the state, s_i, might have different perturbation limits, ϵ_i, and that the choice of L_p-norm affects the shape of the ball. The L_p-norm is defined as $||\boldsymbol{x}||_p = (|x_1|^p + \ldots + |x_n|^p)^{1/p}$ for $\boldsymbol{x} \in \mathbb{R}^n, 1 \leq p \leq \infty$.

B. Robustness certification

This work aims to find the action that maximizes state-action value under a worst-case perturbation of the observation by sensor noise or an adversary. This section explains how to obtain the certified lower bound on the DNN-predicted Q, given a bounded perturbation in the state space from the true state. The derivation is based on [29], re-formulated for RL.

The adversary perturbs the true state, $\boldsymbol{s}_0$, to another state, $\boldsymbol{s}_{adv} \in \mathcal{B}_{p_{adv}}(\boldsymbol{s}_0, \boldsymbol{\epsilon}_{adv})$, within the $\boldsymbol{\epsilon}_{adv}$-ball. The ego agent only observes the perturbed state, $\boldsymbol{s}_{adv}$. As displayed in Fig. 2, let the worst-case state-action value, Q_L, for a given action, a_j, be

$$Q_L(\boldsymbol{s}_{adv}, a_j) = \min_{\boldsymbol{s} \in \mathcal{B}_{p_{adv}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{adv})} Q(\boldsymbol{s}, a_j), \qquad (2)$$

[1] This work considers problems with a continuous state space and discrete action space.

for all states $\boldsymbol{s}$ inside the $\boldsymbol{\epsilon}_{adv}$-Ball around the observation, $\boldsymbol{s}_{adv}$.

The goal of the certification process is to compute a guaranteed lower bound, $Q_l(\boldsymbol{s}, a_j)$, on the minimum state-action value, that is, $Q_l(\boldsymbol{s}, a_j) \leq Q_L(\boldsymbol{s}, a_j)$. The key idea is to pass interval bounds[2] $[\boldsymbol{l}^{(0)}, \boldsymbol{u}^{(0)}] = [\boldsymbol{s}_{adv} - \boldsymbol{\epsilon}_{adv}, \boldsymbol{s}_{adv} + \boldsymbol{\epsilon}_{adv}]$ from the DNN's input layer to the output layer, where $\boldsymbol{l}^{(k)}$ and $\boldsymbol{u}^{(k)}$ denote the lower and upper bounds of the pre-activation term, $\boldsymbol{z}^{(k)}$, i.e., $l_i^{(k)} \leq z_i^{(k)} \leq u_i^{(k)} \forall i \in 1, ..., u_k$, in the k-th layer with u_k units of an m-layer DNN. When passing interval bounds through a ReLU activation[3], $\sigma(\cdot)$, the upper and lower pre-ReLU bounds of each element could either both be positive ($l_r^{(k)}, u_r^{(k)} > 0$), negative ($l_r^{(k)}, u_r^{(k)} < 0$), or positive and negative ($l_r^{(k)} < 0, u_r^{(k)} > 0$), in which the ReLU status is called *active, inactive* or *undecided*, respectively. In the active and inactive case, bounds are passed directly to the next layer. In the undecided case, the output of the ReLU is bounded linearly above and below:

$$\sigma(z_r^{(k)})_{|l_r^{(k)}, u_r^{(k)}} = \begin{cases} [z_r^{(k)}, z_r^{(k)}] \\ \qquad \text{if } l_r^{(k)}, u_r^{(k)} > 0, \text{ ``active''} \\ \\ [0, 0] \\ \qquad \text{if } l_r^{(k)}, u_r^{(k)} < 0, \text{ ``inactive''} \\ \\ [\frac{u_r^{(k)}}{u_r^{(k)} - l_r^{(k)}} z_r^{(k)}, \frac{u_r^{(k)}}{u_r^{(k)} - l_r^{(k)}} (z_r^{(k)} - l_r^{(k)})] \\ \qquad \text{if } l_r^{(k)} < 0, u_r^{(k)} > 0, \text{ ``undecided''}, \end{cases} \tag{3}$$

for each element, indexed by r, in the k-th layer.

The identity matrix, D, is introduced as the ReLU status matrix, H as the lower/upper bounding factor, W as the weight matrix, b as the bias in layer (k) with r or j as indices, and the pre-ReLU-activation, $z_r^{(k)}$, is replaced with $W_{r,:}^{(k)} \boldsymbol{s} + b_r^{(k)}$. The ReLU bounding is then rewritten as

$$\begin{aligned} & D_{r,r}^{(k)} (W_{r,j}^{(k)} s_j + b_r^{(k)}) \\ \leq \quad & \sigma(W_{r,j}^{(k)} s_j + b_r^{(k)}) \\ \leq \quad & D_{r,r}^{(k)} (W_{r,j}^{(k)} s_j + b_r^{(k)} - H_{r,j}^{(k)}), \end{aligned} \tag{4}$$

where

$$D_{r,r}^{(k)} = \begin{cases} 1 & \text{if } l_r^{(k)}, u_r^{(k)} > 0; \\ 0 & \text{if } l_r^{(k)}, u_r^{(k)} < 0, \\ \frac{u_r^{(k)}}{u_r^{(k)} - l_r^{(k)}} & \text{if } l_r^{(k)} < 0, u_r^{(k)} > 0; \end{cases} \tag{5}$$

$$H_{r,j}^{(k)} = \begin{cases} l_r^{(k)} & \text{if } l_r^{(k)} < 0, u_r^{(k)} > 0, A_{j,r}^{(k)} < 0; \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Using these ReLU relaxations, a guaranteed lower bound of the state-action value for a single state $\boldsymbol{s} \in \mathcal{B}_p(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon})$ (based on [29]) is:

$$\bar{Q}_l(\boldsymbol{s}, a_j) = A_{j,:}^{(0)} \boldsymbol{s} + b_j^{(m)} + \sum_{k=1}^{m-1} A_{j,:}^{(k)} (\boldsymbol{b}^{(k)} - H_{:,j}^{(k)}), \tag{7}$$

where the matrix A contains the network weights and ReLU activation, recursively for all layers: $A^{(k-1)} = A^{(k)} W^{(k)} D^{(k-1)}$, with identity in the final layer: $A^{(m)} = \mathbb{1}$.

Unlike the exact DNN output, the bound on the relaxed DNN's output in Eq. (7) can be minimized across an $\boldsymbol{\epsilon}$-ball in closed form (as described in Section IV-C), which is a key piece of this work's real-time, robust decision-making framework.

IV. Approach

This work develops an add-on certified defense for existing Deep RL algorithms to ensure robustness against sensor noise or adversarial examples during test time.

A. System architecture

In an offline training phase, an agent uses a deep RL algorithm, here DQN [49], to train a DNN that maps non-corrupted state observations, $\boldsymbol{s}_0$, to state-action values, $Q(\boldsymbol{s}_0, a)$. Action selection during training uses the nominal cost function, $a_{nom}^* = \text{argmax}_{a_j} Q(\boldsymbol{s}_0, a_j)$.

Figure 4 depicts the system architecture of a standard model-free RL framework with the added-on certification. During online execution, the agent only receives corrupted state observations from the environment. The certification node uses the DNN architecture, DNN weights, W, and robustness hyperparameters, $\boldsymbol{\epsilon}_{rob}, p_{rob}$, to compute lower bounds on possible Q-values for robust action selection.

B. Optimal cost function under worst-case perturbation

We assume that the training process causes the network to converge to the optimal value function, $Q^*(\boldsymbol{s}_0, a)$ and focus on the challenge of handling perturbed observations during execution. Thus, we consider robustness to an adversary that perturbs the true state, $\boldsymbol{s}_0$, within a small perturbation, $\boldsymbol{\epsilon}_{adv}$, into the worst-possible state observation, $\boldsymbol{s}_{adv}$. The adversary assumes that the RL agent follows a nominal policy (as in, e.g., DQN) of selecting the action with highest Q-value at the current observation. A worst possible state observation, $\boldsymbol{s}_{adv}$, is therefore any one which causes the RL agent to take the action with lowest Q-value in the true state, $\boldsymbol{s}_0$:

$$\begin{aligned} \boldsymbol{s}_{adv} \in \{ \boldsymbol{s} : \boldsymbol{s} \in \mathcal{B}_{p_{adv}}(\boldsymbol{s}_0, \boldsymbol{\epsilon}_{adv}) \text{ and } & \\ \underset{a_j}{\text{argmax}}\, Q(\boldsymbol{s}, a_j) = \underset{a_j}{\text{argmin}}\, Q(\boldsymbol{s}_0, a_j) \}. & \end{aligned} \tag{8}$$

This set could be computationally intensive to compute and/or empty – an approximation is described in Section IV-F.

After the agent receives the state observation picked by the adversary, the agent selects an action. Instead of trusting the observation (and thus choosing the worst action for the true state), the agent could leverage the fact that the true state, $\boldsymbol{s}_0$, must be somewhere inside an $\boldsymbol{\epsilon}_{adv}$-Ball around $\boldsymbol{s}_{adv}$ (i.e., $\boldsymbol{s}_0 \in \mathcal{B}_{p_{adv}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{adv})$).

However, in this work, the agent assumes $\boldsymbol{s}_0 \in \mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})$), where we make a distinction between the adversary and robust agent's parameters. This distinction gives the CARRL algorithm hyperparameters $\boldsymbol{\epsilon}_{rob}, p_{rob}$ that provide further flexibility in the defense

[2] Element-wise $\pm \boldsymbol{\epsilon}_{adv}$ can cause overly conservative categorization of ReLUs for $p < \infty$. p is accounted for later in the Algorithm in Eq. (12).

[3] Although this work considers DNNs with ReLU activations, the formulation could be extended to general activation functions via more recent certification process [30].

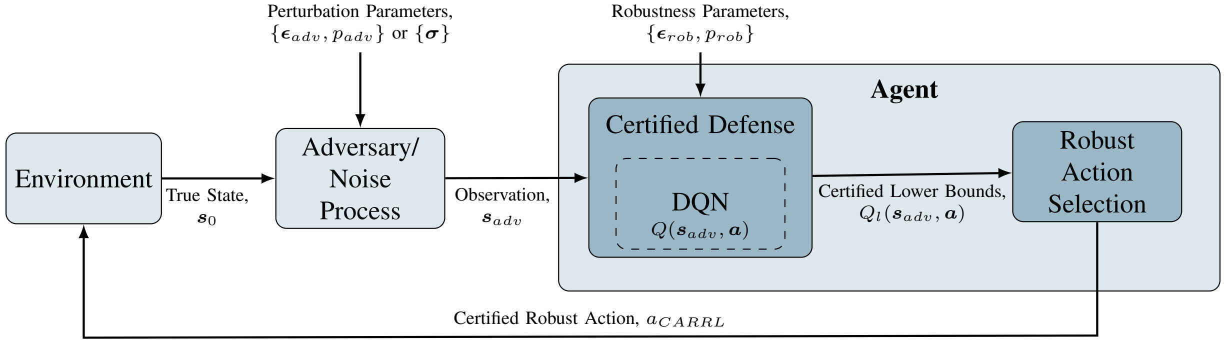

Perturbation Parameters,
$\{\boldsymbol{\epsilon}_{adv}, p_{adv}\}$ or $\{\boldsymbol{\sigma}\}$
Robustness Parameters,
$\{\boldsymbol{\epsilon}_{rob}, p_{rob}\}$
Agent
Environment
True State,
$\boldsymbol{s}_0$
Adversary/
Noise
Process
Observation,
$\boldsymbol{s}_{adv}$
Certified Defense
DQN
$Q(\boldsymbol{s}_{adv}, \boldsymbol{a})$
Certified Lower Bounds,
$Q_l(\boldsymbol{s}_{adv}, \boldsymbol{a})$
Robust
Action
Selection
Certified Robust Action, a_{CARRL}

Fig. 4: System Architecture. During online execution, an agent observes a state, $\boldsymbol{s}_{adv}$, corrupted by an adversary or noise process (constrained to $\boldsymbol{s}_{adv} \in \mathcal{B}_{p_{adv}}(\boldsymbol{s}_0, \boldsymbol{\epsilon}_{adv})$. A trained Deep RL algorithm, e.g., Deep Q-Network (DQN) [49], predicts the state-action values, Q. The certified defense module computes a lower bound of the network's predicted state-action values of each discrete action: Q_l, w.r.t. a robustness threshold $\boldsymbol{\epsilon}_{rob}$ and $L_{p_{rob}}$-norm in the input space. The agent takes the action, a_{CARRL}, that maximizes the lower bound, i.e., is best under a worst-case deviation in the input space.

algorithm's conservatism, that do not necessarily have to match what the adversary applies. Nonetheless, for the sake of providing guarantees, the rest of Section IV assumes $\boldsymbol{\epsilon}_{rob} = \boldsymbol{\epsilon}_{adv}$ and $p_{rob} = p_{adv}$ to ensure $\mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})) = \mathcal{B}_{p_{adv}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{adv}))$. Empirical effects of tuning $\boldsymbol{\epsilon}_{rob}$ to other values are explored in Section V.

The agent evaluates each action by calculating the worst-case Q-value under all possible true states. In accordance with the robust decision making problem, the robust-optimal action, a^*, is defined here as one with the highest Q-value under the worst-case perturbation,

$$a^* = \underset{a_j}{\operatorname{argmax}} \underbrace{\min_{\boldsymbol{s} \in \mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})} Q(\boldsymbol{s}, a_j)}_{Q_L(\boldsymbol{s}_{adv}, a_j)}. \tag{9}$$

As described in Section II, computing $Q_L(\boldsymbol{s}_{adv}, a_j)$ exactly is too computationally intensive for real-time decision-making.

Thus, this work proposes the algorithm **C**ertified **A**dversarial **R**obustness for Deep **RL** (**CARRL**). In CARRL, the action, a_{CARRL}, is selected by approximating $Q_L(\boldsymbol{s}_{adv}, a_j)$ with $Q_l(\boldsymbol{s}_{adv}, a_j)$, its guaranteed lower bound across all possible states $\boldsymbol{s} \in \mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})$, so that:

$$a_{CARRL} = \underset{a_j}{\operatorname{argmax}}\, Q_l(\boldsymbol{s}_{adv}, a_j), \tag{10}$$

where Eq. (16) below defines $Q_l(\boldsymbol{s}_{adv}, a_j)$ in closed form. Conditions for optimality ($a^* = a_{CARRL}$) are described in Section IV-D.

C. Robustness certification with vector-$\boldsymbol{\epsilon}$-ball perturbations

To solve Eq. (10) when $Q(\boldsymbol{s}, a_j)$ is represented by a DNN, we adapt the formulation from [29]. Most works in adversarial examples, including [29], focus on perturbations on image inputs, in which all channels have the same scale (e.g., grayscale images with pixel intensities in $[0, 255]$). More generally, however, input channels could be on different scales (e.g., joint torques, velocities, positions). Existing robustness certification methods require choosing a scalar ϵ_{rob} that bounds the uncertainty across all input channels; in general, this could lead to unnecessarily conservative behavior, as some network inputs might be known with zero uncertainty, or differences in units could make uncertainties across channels incomparable. Hence, this work computes certified bounds on the network output under perturbation bounds specific to each network input channel, as described in a vector $\boldsymbol{\epsilon}_{rob}$ with the same dimension as $\boldsymbol{s}$.

To do so, we minimize $\bar{Q}_l(\boldsymbol{s}, a_j)$ across *all* states in $\mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})$, where $\bar{Q}_l(\boldsymbol{s}, a_j)$ was defined in Eq. (7) as the lower bound on the Q-value for a *particular* state $\boldsymbol{s} \in \mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})$. This derivation uses a vector $\boldsymbol{\epsilon}_{rob}$ (instead of scalar ϵ_{rob} as in [29]):

$$\begin{aligned}
Q_l(\boldsymbol{s}_{adv}, a_j) &= \min_{\boldsymbol{s} \in \mathcal{B}_p(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})} \left(\bar{Q}_l(\boldsymbol{s}, a_j) \right) && (11) \\
&= \min_{\boldsymbol{s} \in \mathcal{B}_p(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})} \Bigg(A^{(0)}_{j,:} \boldsymbol{s} + \\
&\qquad \underbrace{b^{(m)}_j + \sum_{k=1}^{m-1} A^{(k)}_{j,:} (\boldsymbol{b}^{(k)} - H^{(k)}_{:,j})}_{=:\Gamma} \Bigg) && (12) \\
&= \min_{\boldsymbol{s} \in \mathcal{B}_p(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})} \left(A^{(0)}_{j,:} \boldsymbol{s} \right) + \Gamma && (13) \\
&= \min_{\boldsymbol{y} \in \mathcal{B}_p(\boldsymbol{0}, \boldsymbol{1})} \left(A^{(0)}_{j,:} (\boldsymbol{y} \odot \boldsymbol{\epsilon}_{rob}) \right) + A^{(0)}_{j,:} \boldsymbol{s}_{adv} + \Gamma && (14) \\
&= \min_{\boldsymbol{y} \in \mathcal{B}_p(\boldsymbol{0}, \boldsymbol{1})} \left((\boldsymbol{\epsilon}_{rob} \odot A^{(0)}_{j,:}) \boldsymbol{y} \right) + A^{(0)}_{j,:} \boldsymbol{s}_{adv} + \Gamma && (15) \\
&= -||\boldsymbol{\epsilon}_{rob} \odot A^{(0)}_{j,:}||_q + A^{(0)}_{j,:} \boldsymbol{s}_{adv} + \Gamma, && (16)
\end{aligned}$$

with $\odot$ denoting element-wise multiplication. From Eq. (11) to Eq. (12), we substitute in Eq. (7). From Eq. (12) to Eq. (13), we introduce the placeholder variable Γ that does not depend on $\boldsymbol{s}$. From Eq. (13) to Eq. (14), we substitute $\boldsymbol{s} := \boldsymbol{y} \odot \boldsymbol{\epsilon}_{rob} + \boldsymbol{s}_{adv}$, to shift and re-scale the observation to within the unit ball around zero, $\boldsymbol{y} \in \mathcal{B}_p(\boldsymbol{0}, \boldsymbol{1})$. The maximization in Eq. (15) is equivalent to a L_q-norm in Eq. (16) by the definition of the dual norm $||\boldsymbol{z}||_q = \{\sup\ \boldsymbol{z}^T \boldsymbol{y} : ||\boldsymbol{y}||_p \leq 1\}$

and the fact that the L_q norm is the dual of the L_p norm for $p, q \in [1, \infty)$ (with $1/p + 1/q = 1$). Equation (16) is inserted into Eq. (10) to calculate the CARRL action in closed form.

Recall from Section II that the certified bound, Q_l, is the solution of the dual LP that describes a DNN with linearly relaxed ReLU activations. In this work, we refer to the solution of the primal as Q_{LP} (see [28] for the explicit form of the primal/dual LPs for supervised learning).

To summarize, the relationship between each of the Q-value terms is:

$$Q(\boldsymbol{s}_{adv}, a_j) \geq Q_L(\boldsymbol{s}_{adv}, a_j) \geq Q_{LP}(\boldsymbol{s}_{adv}, a_j) \geq Q_l(\boldsymbol{s}_{adv}, a_j) \tag{17}$$

$$Q(\boldsymbol{s}_{adv}, a_j) \geq \bar{Q}_l(\boldsymbol{s}_{adv}, a_j) \geq Q_l(\boldsymbol{s}_{adv}, a_j). \tag{18}$$

D. Guarantees on Action Selection

1) Avoiding a Bad Action: Unlike the nominal DQN rule, which could be tricked to take an arbitrarily bad action, the robust-optimal decision-rule in Eq. (9) can avoid bad actions provided there is a better alternative in the $\boldsymbol{\epsilon}_{rob}$-Ball.

Claim 1: If for some q', $\exists \boldsymbol{s}' \in \mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob}), a'$ s.t. $Q(\boldsymbol{s}', a') \leq q'$ and $\exists a''$ s.t. $\forall \boldsymbol{s} \in \mathcal{B}_{p_{rob}}(\boldsymbol{s}_{adv}, \boldsymbol{\epsilon}_{rob})$ $Q(\boldsymbol{s}, a'') > q'$, then $a^* \neq a'$.

In other words, if action a' is sufficiently bad for some nearby state, $\boldsymbol{s}'$, and at least one other action a'' is better for all nearby states, the robust-optimal decision rule will not select the bad action a' (but DQN might). This is because $Q_L(\boldsymbol{s}_{adv}, a') \leq q'$, $Q_L(\boldsymbol{s}_{adv}, a'') > q' \Rightarrow \text{argmax}_{a_j} Q_L(\boldsymbol{s}_{adv}, a_j) \neq a'$.

2) Matching the Robust-Optimal Action: While Claim 1 refers to the robust-optimal action, the action returned by CARRL is the same as the robust-optimal action (returned by a system that can compute the exact lower bounds) under certain conditions,

$$\underbrace{\underset{a_j}{\text{argmax}}\, Q_l(\boldsymbol{s}_{adv}, a_j)}_{a_{CARRL}} \overset{?}{=} \underbrace{\underset{a_j}{\text{argmax}}\, Q_L(\boldsymbol{s}_{adv}, a_j)}_{a^*}. \tag{19}$$

Claim 2: CARRL selects the robust-optimal action if the certification process satisfies any of the following conditions:

i) $Q_l = Q_L + c_1$, where $c_1 = \text{const.}$
ii) $Q_l = c_2 Q_L$, where $c_2 = \text{const.}, c_2 > 0$
iii) $Q_l = g(Q_L)$, where g is a strictly monotonic function,

where Q_l, Q_L are written without their arguments, $\boldsymbol{s}_{adv}, a_j$. A special case of any these conditions is a tight certification process, i.e., $Q_l(\boldsymbol{s}_{adv}, a_j) = Q_L(\boldsymbol{s}_{adv}, a_j)$. Using the Fast-Lin-based approach in this work, for a particular observation, confirmation that all of the ReLUs are "active" or "inactive" in Eq. (3) would provide a tightness guarantee.

In cases where i)-iii) are not fulfilled, but Claim 1 is fulfilled, CARRL is not guaranteed to select the robust-optimal action, but will still reason about all possible outcomes, and empirically selects a better action than a nominal policy across many settings explored in Section V.

Note that when $\boldsymbol{\epsilon}_{rob} = \boldsymbol{0}$, no robustness is applied, so both the CARRL and robust-optimal decisions reduce to the DQN decision, since $Q_l(\boldsymbol{s}_{adv}, a_j) = Q_L(\boldsymbol{s}_{adv}, a_j) = Q(\boldsymbol{s}_{adv}, a_j)$.

E. Probabilistic Robustness

The discussion so far considered cases where the state perturbation is known to be bounded (as in, e.g., many adversarial observation perturbation definitions [33], stochastic processes with finite support. However, in many other cases, the observation uncertainty is best modeled by a distribution with infinite support (e.g., Gaussian). To be fully robust to this class of uncertainty (including very low probability events) CARRL requires setting $\epsilon_{rob,i} = \infty$ for the unbounded state elements, $\boldsymbol{s}_i$.

For instance, for a Gaussian sensor model with known standard deviation of measurement error, $\boldsymbol{\sigma}_{sensor}$, one could set $\boldsymbol{\epsilon}_{rob} = 2\boldsymbol{\sigma}_{sensor}$ to yield actions that account for the worst-case outcome with 95% confidence. In robotics applications, for example, $\boldsymbol{\sigma}_{sensor}$ is routinely provided in a sensor datasheet. Implementing this type of probabilistic robustness only requires a sensor model for the observation vector and a desired confidence bound to compute the corresponding $\boldsymbol{\epsilon}_{rob}$.

F. Adversaries

To evaluate the learned policy's robustness to deviations of the input in an $\boldsymbol{\epsilon}_{adv}$-Ball, we pass the true state, $\boldsymbol{s}_0$, through an adversarial/noise process to compute $\boldsymbol{s}_{adv}$, as seen in Fig. 4. Computing an adversarial state $\boldsymbol{s}_{adv}$ exactly, using Eq. (8), is computationally intensive. Instead, we use a fast gradient sign method with targeting (FGST) [8] to approximate the adversary from Eq. (8). FGST chooses a state $\hat{\boldsymbol{s}}_{adv}$ on the L_∞ $\boldsymbol{\epsilon}_{adv}$-Ball's perimeter to encourage the agent to take the nominally worst action, $\text{argmin}_{a_j} Q(\boldsymbol{s}_0, a_j)$. Specifically, $\hat{\boldsymbol{s}}_{adv}$ is picked along the direction of sign of the lowest cross-entropy loss, $\mathcal{L}$, between $\boldsymbol{y}_{adv}$, a one-hot encoding of the worst action at $\boldsymbol{s}_0$, and $\boldsymbol{y}_{nom}$, the softmax of all actions' Q-values at $\boldsymbol{s}_0$. The loss is taken w.r.t. the proposed observation $\boldsymbol{s}$:

$$\boldsymbol{y}_{adv} = [\mathbb{1}\{a_i = \text{argmin}_{a_j} Q(\boldsymbol{s}_0, a_j)\}] \in \mathbb{U}^d \tag{20}$$

$$Q_{nom} = [Q(\boldsymbol{s}_0, a_j) \forall j \in \{0, \ldots, d-1\}] \in \mathbb{R}^d \tag{21}$$

$$\boldsymbol{y}_{nom} = \text{softmax}(Q_{nom}) \in \Delta^d \tag{22}$$

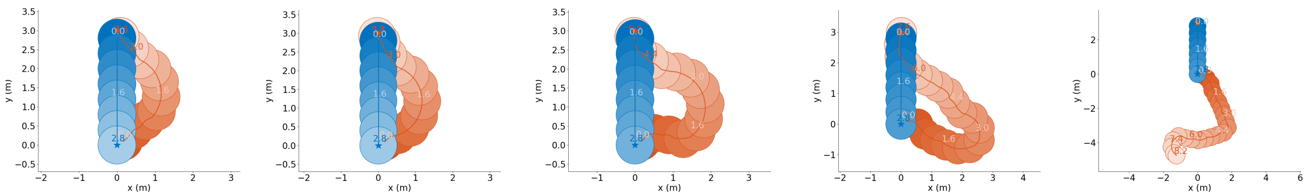
$$\hat{\boldsymbol{s}}_{adv} = \boldsymbol{s}_0 - \boldsymbol{\epsilon}_{adv} \odot \text{sign}(\nabla_{\boldsymbol{s}} \mathcal{L}(\boldsymbol{y}_{adv}, \boldsymbol{y}_{nom})), \tag{23}$$

where $\mathbb{U}^d$ denotes a one-hot vector of dimension d, and Δ^d denotes the standard d-simplex.

In addition to adversarial perturbations, Section V also considers uniform noise perturbations, where $\boldsymbol{s}_{adv} \sim \texttt{Unif}\left([\boldsymbol{s}_0 - \boldsymbol{\sigma}, \boldsymbol{s}_0 + \boldsymbol{\sigma}]\right)$.

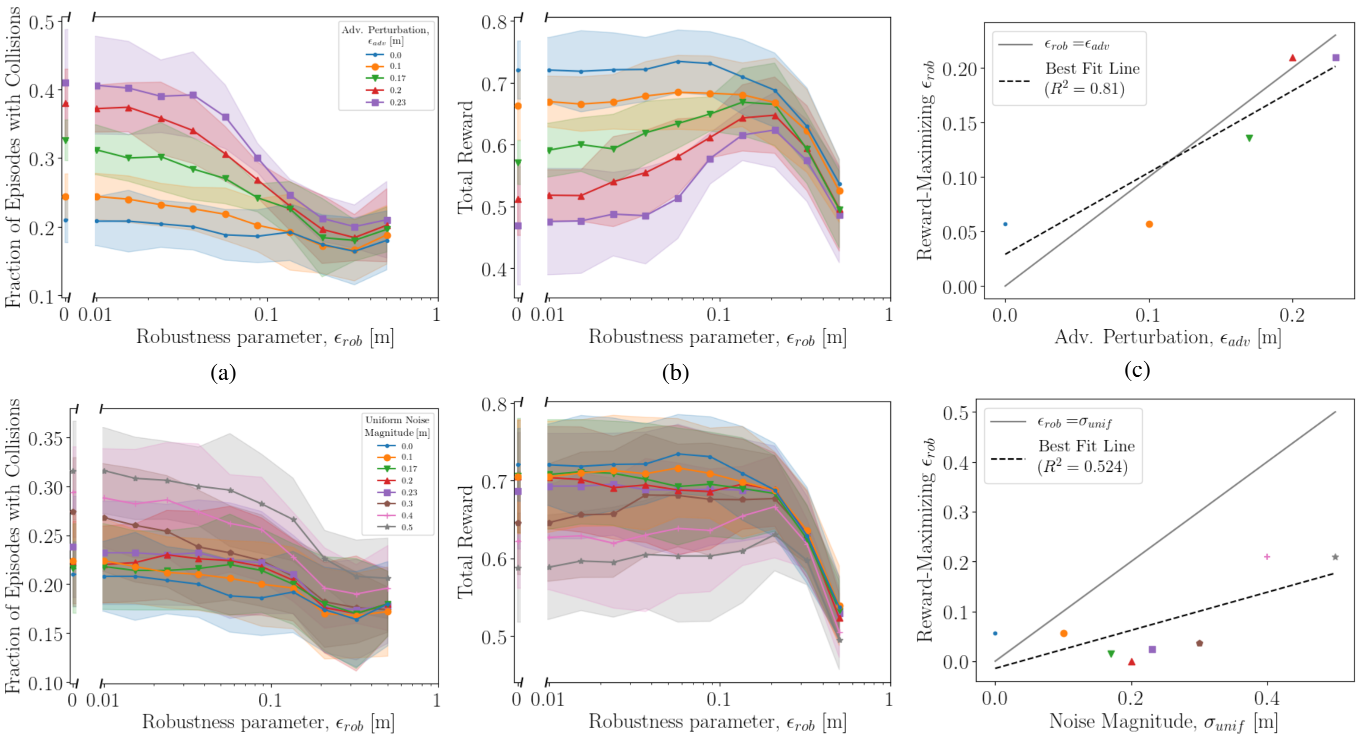
V. Experimental Results

The key result is that while imperfect observations reduce the performance of a nominal deep RL algorithm, our proposed algorithm, CARRL, recovers much of the performance by adding robustness during execution. Robustness against perturbations from an adversary or noise process during execution is evaluated in two simulated domains: collision avoidance among dynamic, decision-making obstacles [50] and cartpole [51]. This section also describes the impact of the $\boldsymbol{\epsilon}_{rob}$ hyperparameter, the ability to handle behavioral adversaries, a comparison with another certification process, and provides intuition on the tightness of the lower bounds.



(a) $\epsilon_{rob} = 0.0$ (b) $\epsilon_{rob} = 0.1$ (c) $\epsilon_{rob} = 0.46$ (d) $\epsilon_{rob} = 1.0$ (e) $\epsilon_{rob} = 2.0$

Fig. 5: Increase of conservatism with ϵ_{rob}. An agent (orange) following the CARRL policy avoids a dynamic, non-cooperative obstacle (blue) that is observed without noise. An increasing robustness parameter ϵ_{rob} (left to right) increases the agent's conservatism, i.e., the agent avoids the obstacle with a greater safety distance.



(a) (b) (c)

(d) (e) (f)

Fig. 6: Robustness against adversarial attacks (top row) and noise (bottom row). Increasing the robustness parameter, ϵ_{rob}, decreases the number of collisions in the presence of adversarial attacks (a), or noise (d) for various perturbation magnitudes, $\epsilon_{adv}, \sigma_{\text{unif}}$. CARRL also recovers substantial amounts of reward lost due to imperfect observations as ϵ_{rob} increases (b,e). The choice of ϵ_{rob} to maximize reward in the presence of a particular adversary or noise process can be guided by the models in (c,f).

A. *Collision Avoidance Domain*

Among the many RL tasks, a particularly challenging safety-critical task is collision avoidance for a robotic vehicle among pedestrians. Because learning a policy in the real world is dangerous and time consuming, this work uses a *gym*-based [51] kinematic simulation environment [50] for learning pedestrian avoidance policies. In this work, the RL policy controls one of two agents with 11 discrete actions: change of heading angle evenly spaced between $[-\pi/6, +\pi/6]$ and constant velocity $v = 1$ m/s. The environment executes the selected action under unicycle kinematics, and controls the other agent from a diverse set of fixed policies (static, non-cooperative, ORCA [52], GA3C-CADRL [4]). The sparse reward is 1 for reaching the goal, -0.25 for colliding (and 0 otherwise, i.e., zero reward is received in cases where the agent does not reach its goal in a reasonable amount of time). The observation vector includes the CARRL agent's goal, each agent's radius, and the other agent's x-y position and velocity, with more detail in [4]. In this domain, robustness and perturbations are applied only on the measurement of the other agent's x-y position (i.e., $\epsilon_{rob} = \epsilon_{rob} \cdot [0 \ \ldots \ 0 \ 1 \ 1 \ 0 \ \ldots 0]$) – an example of CARRL's ability to handle uncertainties of varying scales.

A non-dueling DQN policy was trained with 2, 64-unit layers with the following hyperparameters: learning rate 2.05e−4, ϵ-greedy exploration ratio linearly decaying from 0.5 to 0.05, buffer size 152e3, 4e5 training steps, and target network update frequency, 10e3. The hyperparameters were found by running 100 iterations of Bayesian optimization with Gaussian Processes [53] on the maximization of the sparse training reward.

After training, CARRL is added onto the trained DQN policy. Intuition on the resulting CARRL policy is demonstrated in Fig. 5. The figure shows the trajectories of the CARRL agent (orange) for increasing ϵ_{rob} values in a scenario with unperturbed observations. With increasing ϵ_{rob} (toward right), the CARRL agent accounts for increasingly large worst-case perturbations of the other agent's position. Accordingly, the agent avoids the dynamic agent (blue) increasingly conservatively, i.e., selects actions that leave a larger safety distance. When ϵ_{rob}=2.0, the CARRL agent is overly conservative – it "runs away" and does not reach its goal, which is explained more in Section V-F.

Figure 6 shows that the nominal DQN policy is not robust to the perturbation of inputs. In particular, increasing the magnitude of adversarial, or noisy perturbation, $\epsilon_{adv}, \sigma_{\text{unif}}$, drastically 1) increases the average number of collisions (as seen in Figs. 6a and 6d, respectively, at $\epsilon_{rob} = 0$) and 2) decreases the average reward (as seen in Figs. 6b and 6e). The results in Fig. 6 are evaluated in scenarios where the other agent is non-cooperative (i.e., travels straight to its goal position at constant velocity) and each datapoint represents the average of 100 trials across 5 random seeds that determine the agents' initial/goal positions and radii (shading represents ± 1 standard deviation of average value per seed).

Next, we demonstrate that CARRL recovers performance. Increasing the robustness parameter ϵ_{rob} decreases the number of collisions under varying magnitudes of noise, or adversarial attack, as seen in Figs. 6a and 6d. Because collisions affect the reward function, the received reward also increases with an increasing robustness parameter $\epsilon_{rob}{<}{\sim}0.1$ under varying magnitudes of perturbations. As expected, the effect of the proposed defense is highest under large perturbations, as seen in the slopes of the curves ϵ_{adv}=0.23 (violet) and σ_{unif}=0.5 (gray).

Since the CARRL agent selects actions more conservatively than a nominal DQN agent, it is able to successfully reach its goal instead of colliding like a nominal DQN agent does under many scenarios with noisy or adversarial perturbations. However, the effect of overly conservative behavior seen in Figs. 5d and 5e appears in Fig. 6 for $\epsilon_{rob}{>}\sim 0.2$, as the reward drops significantly. This excessive conservatism for large $\boldsymbol{\epsilon}_{rob}$ can be partially explained by the fact that highly conservative actions may move the agent's position observation into states that are far from what the network was trained on, which breaks CARRL's assumption of a perfectly learned Q-function. Further discussion about the conservatism inherent in the certified lower bounds is discussed in Section V-F.

Figures 6c and 6f illustrate further intuition on choosing ϵ_{rob}. Figure 6c demonstrates a strong correlation between the attack magnitude ϵ_{adv} and the best (i.e., reward-maximizing) robustness hyperparameter ϵ_{rob} under that attack magnitude from Fig. 6b. In the case of uniform noise, the correlation between ϵ_{rob} and σ_{unif} is weaker, because the FGST adversary chooses an input state on the perimeter of the $\boldsymbol{\epsilon}_{adv}$-Ball, whereas uniform noise samples lie inside the $\boldsymbol{\sigma}_{unif}$-Ball.

The flexibility in setting ϵ_{rob} enables CARRL to capture uncertainties of various magnitudes in the input space, e.g., ϵ_{rob} could be adapted on-line to account for a perturbation magnitude that is unknown a priori, or to handle time-varying sensor noise.

B. Cartpole Domain

In the cartpole task [51], [54], the reward is the number of time steps (capped at 200) that a pole remains balanced upright (±12° from vertical) on a cart that can translate along a horizontal track. The state vector, $\boldsymbol{s} = \left[p_{cart}, v_{cart}, \theta_{pole}, \dot{\theta}_{pole}\right]^T$ and action space, $\boldsymbol{a} \in \{\text{push cart left}, \text{push cart right}\}$ are defined in [51]. A 2-layer, 4-unit network was trained in an environment without any observation perturbations using an open-source DQN implementation [55] with Bayesian Optimization used to find training hyperparameters. The trained DQN is evaluated against perturbations of various magnitudes, shown in Fig. 7. In this domain, robustness and perturbations are applied to all states equally, i.e., $\boldsymbol{\epsilon}_{adv} = \epsilon_{adv} \cdot \mathbb{1} \in \mathbb{R}^4$, $\boldsymbol{\epsilon}_{rob} = \epsilon_{rob} \cdot \mathbb{1} \in \mathbb{R}^4$ and $\boldsymbol{\sigma}_{unif} = \sigma_{unif} \cdot \mathbb{1} \in \mathbb{R}^4$.

Each curve in Figs. 7a and 7c corresponds to the average reward received under different magnitudes of adversarial, or uniform noise perturbations, respectively. The reward of a nominal DQN agent drops from 200 under perfect measurements to 105 under adversarial perturbation of magnitude $\epsilon_{adv} = 0.075$ (blue and red reward at x-axis, $\epsilon_{rob} = 0$ in Fig. 7a) or to 145 under uniform noise of magnitude $\sigma_{\text{unif}} = 0.5$ (Fig. 7c). For $\epsilon_{rob} > 0$ (moving right along x-axis), the CARRL algorithm considers an increasingly large range of states in its worst-case outcome calculation. Accordingly, the algorithm is able to recover some of the performance lost due to imperfect observations. For example, with $\epsilon_{adv} = 0.075$ (red triangles), CARRL achieves 200 reward using CARRL with $\epsilon_{rob} = 0.1$. The ability to recover performance is due to CARRL selecting actions that consider the worst-case state (e.g., a state in which the pole is closest to falling), rather than fully trusting the perturbed observations.

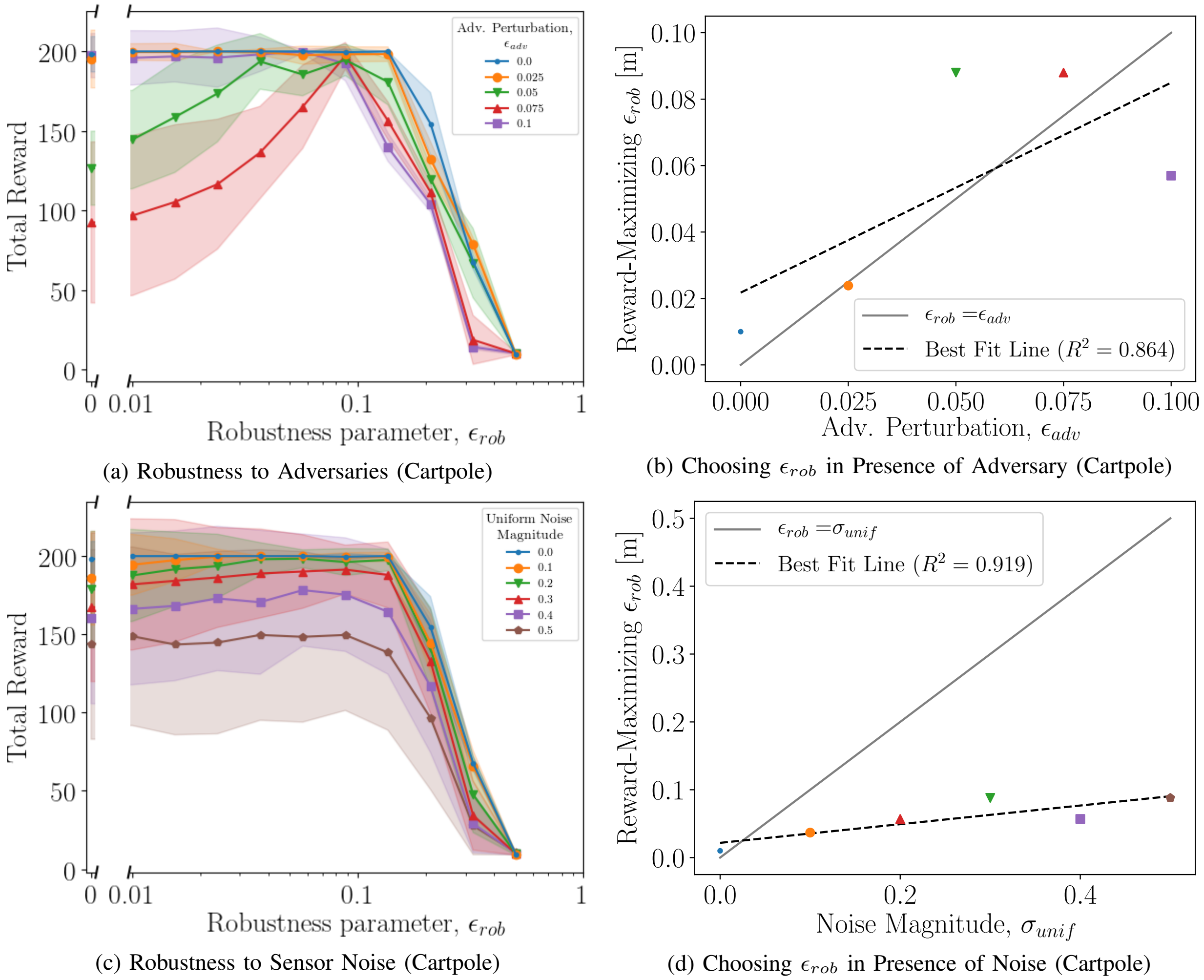
However, there is again tradeoff between robustness and conservatism. For large values of ϵ_{rob}, the average reward declines steeply for all magnitudes of adversaries and noise, because CARRL considers an excessively large set of worst-case states (more detail provided in Section V-F).

The reward-maximizing choice of ϵ_{rob} for a particular class and magnitude of perturbations is explored in Figs. 7b and 7d. Similarly to the collision avoidance domain, the best choice of ϵ_{rob} is close to ϵ_{adv} under adversarial perturbations and less than σ_{unif} under uniform noise perturbations.

These results use 200 random seeds causing different initial conditions.

C. Computational Efficiency

For the cartpole task, one forward pass with certified bounds takes on average 0.68 ± 0.06ms, which compares to a forward pass of the same DQN (nominal, without certified bounds) of 0.24 ± 0.03ms; for collision avoidance, it takes 1.85 ± 1.62ms (CARRL) and 0.30 ± 0.04ms (DQN), all on one i7-6700K CPU. In our implementation, the certified bound of all actions (i.e., 2 or 11 discrete actions for the cartpole and collision avoidance domain, respectively) are computed in parallel. While the DQNs used in this work are relatively small, [29] shows that the runtime of Fast-Lin scales linearly with the network size, and a recent GPU implementation offers faster



(a) Robustness to Adversaries (Cartpole)

(b) Choosing ϵ_{rob} in Presence of Adversary (Cartpole)

(c) Robustness to Sensor Noise (Cartpole)

(d) Choosing ϵ_{rob} in Presence of Noise (Cartpole)

Fig. 7: Results on Cartpole. CARRL recovers performance (measured by reward received) by adding robustness under various magnitudes of adversarial (a) and uniform noise (c) perturbations. Each curve in (a,c) correspond to a different magnitude of perturbation, and $\epsilon_{rob} = 0$ corresponds to zero robustness, i.e., DQN. For all adversary/noise magnitudes, CARRL becomes overly conservative for large ϵ_{rob}, and the performance degrades. Thus, choosing the best ϵ_{rob} for a particular perturbation magnitude can be guided by the curves in (b,d).

performance [56], suggesting CARRL could be implemented in real-time for higher-dimensional RL tasks, as well.

D. Robustness to Behavioral Adversaries

In addition to observational perturbations, many real-world domains also require interaction with other agents, whose *behavior* could be adversarial. In the collision avoidance domain, this can be modeled by an environment agent who actively tries to collide with the CARRL agent, as opposed to the various cooperative or neutral behavior models seen in the training environment described earlier. Although the CARRL formulation does not explicitly consider behavioral adversaries, one can introduce robustness to this class of adversarial perturbation through robustness in the observation space, namely by specifying uncertainty in the other agent's position. In other words, requiring the CARRL agent to consider worst-case positions of another agent while selecting actions causes the CARRL agent to maintain a larger spacing, which in turn prevents the adversarially behaving agent from getting close enough to cause collisions.

In the collision avoidance domain, we parameterize the adversarial "strength" by a collaboration coefficient, λ, where $\lambda = 0.5$ corresponds to a nominal ORCA agent (that does half the collision avoidance), $\lambda = 0$ corresponds to a non-cooperative agent that goes straight toward its goal, and $\lambda \in [-1, 0)$ corresponds to an adversarially behaving agent. Adversarially behaving agents sample from a Bernoulli distribution (every 1 second) with parameter $|\lambda|$. If the outcome is 1, the adversarial agent chooses actions directly aiming into the CARRL agent's projected future position, otherwise, it chooses actions moving straight toward its goal position. Thus, $\lambda = -1$ means the adversarial agent is always trying to collide with the CARRL agent.

The idea of using observational robustness to protect against behavioral uncertainty is quantified in Fig. 8, where each curve corresponds to a different behavioral adversary. Increasing the magnitude of $\lambda < 0$ (increasingly strong adversaries) causes collisions with increasing frequency, since the environment transition model is increasingly different from what was seen during training. Increasing CARRL's robustness parameter

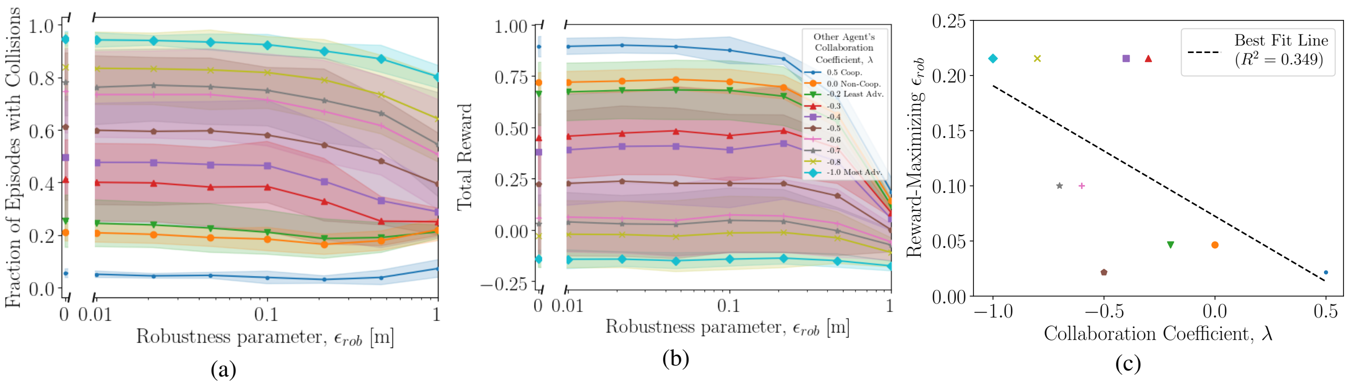


Fig. 8: Robustness to Adversarial Behavior. Each curve shows a different magnitude of adversarial *behavior* of another agent in the collision avoidance task. The adversarially behaving other agents (negative collaboration coefficient) are able to cause many collisions with a CARRL agent trained among cooperative and non-cooperative agents. Although CARRL was not explicitly designed to handle behavioral adversaries, CARRL can reduce the number of collisions by providing robustness in the other agent's position measurement. CARRL's effect on reward (b) is not as strong as in Fig. 6, but the reward-maximizing $\epsilon_{rob} > 0$ against all adversaries (c), and there is a trend of larger ϵ_{rob} working well against stronger adversaries.

ϵ_{rob} leads to a reduction in the number of collisions, as seen in Fig. 8a. Accordingly, the reward received increases for certain values of $\epsilon_{rob} > 0$ (seen strongest in red, violet curves; note y-axis scale is wider than in Fig. 6b). Although the impact on the reward function is not as large as in the observational uncertainty case, Fig. 8c shows that the reward-maximizing choice of ϵ_{rob} has some negative correlation the adversary's strength (i.e., the defense should get stronger against a stronger adversary).

The same trade-off of over-conservatism for large ϵ_{rob} exists in the behavioral adversary setting. Because there are perfect observations in this experiment (just like training), CARRL has minimal effect when the other agent is cooperative (blue, $\lambda = 0.5$). The non-cooperative curve (orange, $\lambda = 0$) matches what was seen in Fig. 6 with $\epsilon_{adv} = 0$ or $\sigma_{\text{unif}} = 0$. 100 test cases with 5 seeds were used.

E. Comparison to LP Bounds

As described in Section II, the convex relaxation approaches (e.g., Fast-Lin) provide relatively loose, but fast-to-compute bounds on DNN outputs. Equation (17) relates various bound tightnesses theoretically, which raises the question: how much better would the performance of an RL agent be, given more computation time to better approximate the worst-case outcome, Q_L?

In Fig. 9, we compare the performance of an agent following the CARRL decision rule, versus one that approximates $Q_L(\boldsymbol{s}_{adv}, \boldsymbol{a})$ with the primal LP (in the collision avoidance domain with observational perturbations). The key takeaway is that there is very little difference: the CARRL curves are solid and the LP curves are dashed, for various settings of adversarial perturbation, ϵ_{adv}. The small difference in the two algorithms is explained by the fact that CARRL provides extra conservatism versus the LP (CARRL accounts for a worse worst-case than the LP), so the CARRL algorithm performs slightly better when ϵ_{rob} is set too small, and slightly worse when ϵ_{rob} is too large for the particular observational adversary.

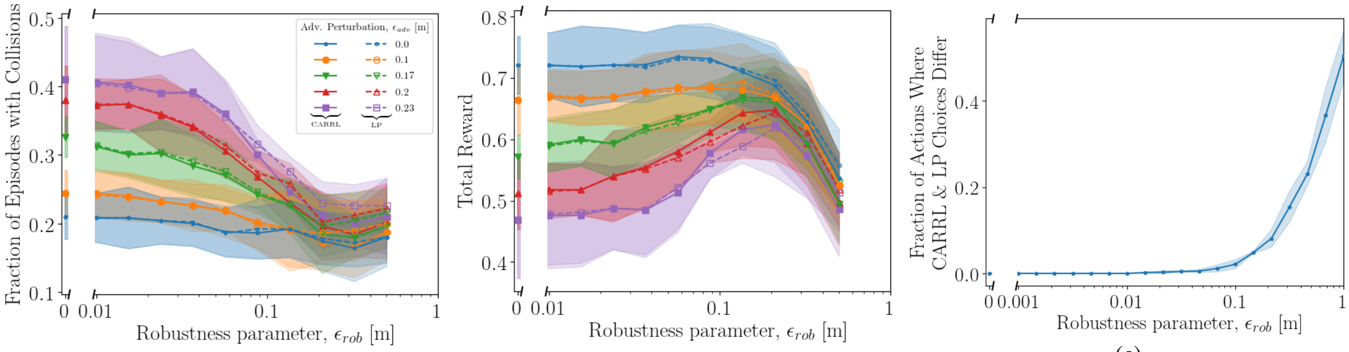
This result is further explained by Fig. 9c, where it is shown that CARRL and LP-based decision rules choose the same action $> 99\%$ of the time for $\epsilon < 0.1$, meaning in this experiment, the extra time spent computing the tighter bounds has little impact on the RL agent's decisions.

F. Intuition on Certified Bounds

Visual inspection of actual adversarial polytopes and the corresponding certified bounds provides additional intuition into the CARRL algorithm. The state uncertainty's mapping into an adversarial polytope in the Q-value space is visualized in Figs. 10 and 11. In Fig. 10a, a CARRL agent observes another agent (of the same size) positioned somewhere in the concentric, colored regions. The state uncertainty, drawn for various values of $\boldsymbol{\epsilon}_{rob}$ with L_∞ norm manifests itself in Fig. 10b as a region of possible $Q(\boldsymbol{s}_{adv}, a_j)$ values, for each action, a_j. Because there are only two dimensions of state uncertainty, we can exhaustively sample Q-values for these states. To visualize the corresponding Q-value region in 2D, consider just two actions, a_0, a_5 (right-most and straight actions, respectively).

The certified lower bounds, Q_l for each action are shown as dotted lines for each of the $\boldsymbol{\epsilon}_{rob}$-Balls. Note that for small $\boldsymbol{\epsilon}_{rob}$ (orange) the bounds are quite tight to the region of possible Q-values. A larger $\boldsymbol{\epsilon}_{rob}$ region (blue) leads to looser (but still useful) bounds, and this case also demonstrates $\boldsymbol{\epsilon}_{rob}$ with non-uniform components ($\epsilon_4 = 0.1, \epsilon_5 = 0.2$, where $(4, 5)$ are the indices of the other agent's position in the state vector) For large $\boldsymbol{\epsilon}_{rob}$ (green), the bounds are very loose, as the lowest sampled Q-value for a_0 is 0.3, but $Q_l(\boldsymbol{s}_{adv}, a_0) = 0.16$.

This increase in conservatism with ϵ_{rob} is explained by the formulation of Fast-Lin, in that linear bounds around an "undecided" ReLU become looser for larger input uncertainties. That is, as defined in Eq. (3), the lower linear bound of an undecided ReLU in layer k, neuron i is $\frac{u_i^{(k)} \cdot l_i^{(k)}}{u_i^{(k)} + l_i^{(k)}} < 0$ for $z_i^{(k)} = l_i^{(k)}$, whereas a ReLU can never output a negative number. This under-approximation gets more extreme for large



(a) (b) (c)

Fig. 9: Certified Bounds vs. LP. Using certified bounds on the adversarial polytope, CARRL's action-selection produces similar performance to that of action-selection using the primal LP (solid vs. dashed curves in each color). In (c), CARRL and LP select identical actions on $> 99\%$ of timesteps (across 60 episodes) for small ϵ_{rob}, but the two methods diverge as the certified bounds become overly conservative. Thus, for small-to-moderate ϵ_{rob}, the extra computation required for the LP does not have substantial effect on decision-making.

$\epsilon_{rob,i}$, since $u_i^{(k)}, l_i^{(k)}$ become further apart in the input layer, and this conservatism is propagated through the rest of the network. Expanding the possible uncertainty magnitudes without excessive conservatism and while maintaining computational efficiency could be an area of future research. Nonetheless, the lower bounds in CARRL were sufficiently tight to be beneficial across many scenarios.

In addition to $\boldsymbol{\epsilon}_{rob}$, Fig. 11a considers the impact of variations in $\boldsymbol{s}_{adv}$: when the other agent is closer to the CARRL agent (purple), the corresponding Q-values in Fig. 11b are lower for each action, than when the other agent is further away (red). Note that the shape of the adversarial polytope (region of Q-value) can be quite complicated due to the high dimensionality of the DNN (inset of Fig. 11b). Furthermore, the $\times$ symbols correspond to the Q-value if the agent simply inflated the other agent's radius to account for the whole region of uncertainty. This heuristic is highly conservative (and domain-specific), as the $\times$'s have lower Q-values than the certified lower bounds for each action in these examples.

VI. Future Directions

In laying a foundation for certified adversarial robustness in deep RL, this work offers numerous future research directions in connecting deep RL algorithms and real-world, safety-critical systems.

For example, extensions of CARRL to other RL tasks will raise the question: how does the certification process extend to continuous action spaces? When $\boldsymbol{\epsilon}_{adv}$ is unknown, as in our empirical results, how could $\boldsymbol{\epsilon}_{rob}$ be tuned efficiently online while maintaining robustness guarantees? How can the online robustness estimates account for uncertainties from the training process (e.g., unexplored states) in a guaranteed manner?

The key factor in the performance of robustness algorithms is the ability to precisely describe the uncertainty over which to be robust. This work provides the $\boldsymbol{\epsilon}_{rob}$, p_{rob} hyperparameters to describe various shapes and sizes of uncertainties in different dimensions of the observation space and shows how this description could be applied under non-uniform, probabilistic uncertainties or a model of behavioral uncertainty. However, real-world systems also include other types of environmental uncertainties. For instance, observational uncertainties beyond L_p balls could be studied for certified defenses in deep RL (e.g., [57], [58] in supervised learning). Moreover, how can one protect against an adversary that is allowed to plan n timesteps into the future (e.g., extending [59] to model-free RL)?

Understanding each of these areas of extension will be crucial in providing both performance and robustness guarantees for deep RL algorithms deployed on real-world systems.

VII. Conclusion

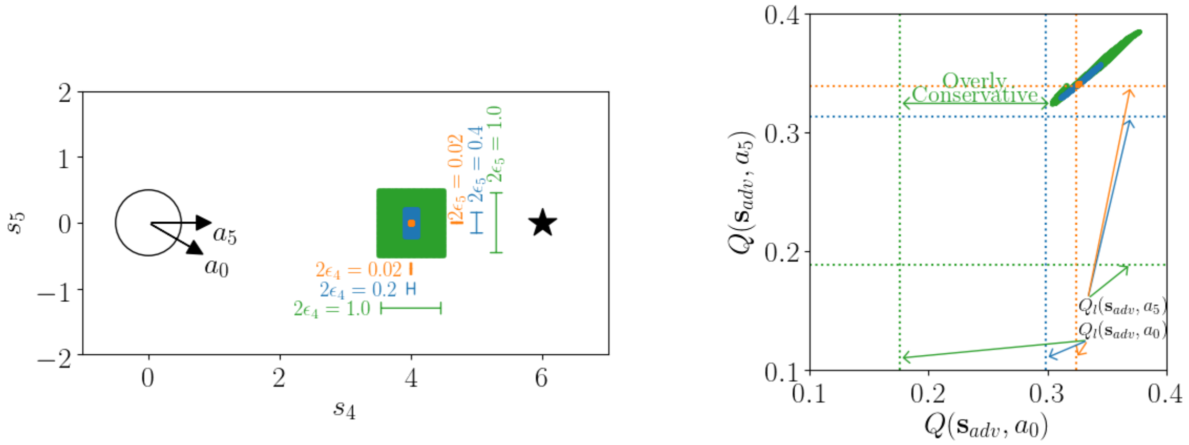
This work adapted deep RL algorithms for application in safety-critical domains, by proposing an add-on certified defense to address existing failures under adversarially perturbed observations and sensor noise. The proposed extension of robustness certification tools from the verification literature into a deep RL formulation enabled efficient calculation of a lower bound on Q-values, given the observation perturbation/uncertainty. These guaranteed lower bounds were used to modify the action selection rule to provide maximum performance under worst-case observation perturbations. The resulting policy (added onto trained DQN networks) was shown to improve robustness to adversaries and sensor noise, causing fewer collisions in a collision avoidance domain and higher reward in cartpole. Furthermore, the proposed algorithm was demonstrated in the presence of adversaries in the behavior space, compared against a more time-intensive alternative, and visualized for particular scenarios to provide intuition on the algorithm's conservatism.

Acknowledgment

This work is supported by Ford Motor Company. The authors greatly thank Tsui-Wei (Lily) Weng for providing code for the Fast-Lin algorithm and insightful discussions, as well as Rose Wang and Dr. Kasra Khosoussi.

References

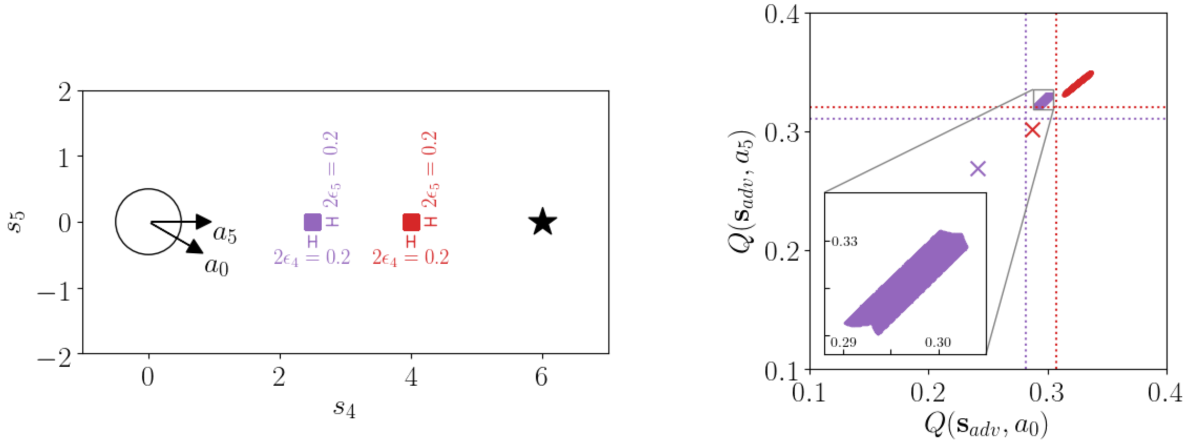
[1] B. Lütjens, M. Everett, and J. P. How, "Certified adversarial robustness for deep reinforcement learning," in *2019 Conference on Robot Learning (CoRL)*, Osaka, Japan, October 2019. [Online]. Available: https://arxiv.org/pdf/1910.12908.pdf



(a) Position Uncertainty of Another Agent [m]

(b) Corresponding Certified Bounds & Sampled DQN Outputs

Fig. 10: Influence of ϵ_{rob} on Q-Values. In (a), the CARRL agent (◯) has a goal (★) at (6,0) and decides between two actions, a_0 and a_5. A second agent could be centered somewhere in the colored regions, corresponding to different values of ϵ_{rob}. In (b) are the corresponding Q-values for those possible states (orange, blue green regions in top-right), exhaustively sampled in each ϵ_{rob}-Ball. As ϵ_{rob} increases, the spread of possible Q-values increases. CARRL's lower bounds on each action, $Q_l(\boldsymbol{s}_{adv}, a_0)$, $Q_l(\boldsymbol{s}_{adv}, a_5)$, are depicted by the dotted lines. Conservatism is measured by the gap between the dashed line and the left/bottom-most sampled point. For small ϵ_{rob} (orange), the bounds are tight; for moderate ϵ_{rob} (blue) are moderately conservative, and for large ϵ_{rob} (green), the linear approximation of ReLU degrades, causing excessive conservatism.



(a) Position Uncertainty of Another Agent [m]

(b) Corresponding Certified Bounds & Sampled DQN Outputs

Fig. 11: Influence of $\boldsymbol{s}_{adv}$ on Q-Values. For the same ϵ_{rob}, the spread of Q-values are shown for two examples of $\boldsymbol{s}_{adv}$. When the other agent is close (purple), the Q-values are lower than when the other agent is far (red). A closer look at one of the non-convex adversarial polytopes is inset in (b). Moreover, if one instead used the heuristic of simply inflating the other agent's radius, the Q-values would lie at the $\times$'s – in both cases, radius inflation is more conservative (further toward bottom-left) than CARRL.

[2] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[3] T. Fan, X. Cheng, J. Pan, P. Long, W. Liu, R. Yang, and D. Manocha, "Getting robots unfrozen and unlost in dense pedestrian crowds," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1178–1185, 2019.

[4] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2018.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[6] N. Akhtar and A. S. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[7] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, 2019.

[8] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *International Conference on Learning Representation (ICLR) (Workshop)*, 2017.

[9] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1528–1540.

[10] Tencent Keen Security Lab, "Experimental security research of Tesla Autopilot," 03 2019. [Online]. Available: https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf

[11] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learn-

ing at scale," in *International Conference on Learning Representations (ICLR)*, 2017.

[12] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[13] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," in *International Conference on Learning Representations (ICLR) (Workshop)*, 2017.

[14] M. Mirman, M. Fischer, and M. Vechev, "Distilled agent DQN for provable adversarial robustness," 2019. [Online]. Available: https://openreview.net/forum?id=ryeAy3AqYm

[15] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 582–597.

[16] F. Tramr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *International Conference on Learning Representations (ICLR)*, 2018.

[17] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. New York, NY, USA: ACM, 2017, pp. 3–14.

[18] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defenses: Ensembles of weak defenses are not strong," in *Proceedings of the 11th USENIX Conference on Offensive Technologies*, ser. WOOT'17. Berkeley, CA, USA: USENIX Association, 2017, pp. 15–15.

[19] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 274–283.

[20] J. Uesato, B. O'Donoghue, P. Kohli, and A. van den Oord, "Adversarial risk and the dangers of evaluating against weak attacks," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5025–5034.

[21] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *ATVA*, 2017.

[22] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, 2017, pp. 97–117.

[23] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, pp. 3–29.

[24] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward relu neural networks," *CoRR*, vol. abs/1706.07351, 2017. [Online]. Available: http://arxiv.org/abs/1706.07351

[25] V. Tjeng, K. Y. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *International Conference on Learning Representations (ICLR)*, 2019.

[26] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 3–18.

[27] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018.

[28] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80, 2018, pp. 5283–5292.

[29] T. Weng, H. Zhang, H. Chen, Z. Song, C. Hsieh, L. Daniel, D. Boning, and I. Dhillon, "Towards fast computation of certified robustness for relu networks," in *International Conference on Machine Learning (ICML)*, 2018.

[30] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 4939–4948.

[31] A. Boopathy, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel, "Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks," in *AAAI Conference on Artificial Intelligence (AAAI)*, Jan 2019.

[32] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *arXiv preprint arXiv:2001.09684*, 2020.

[33] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[34] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," 2017.

[35] V. Behzadan and A. Munir, "Vulnerability of deep reinforcement learning to policy induction attacks," in *International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM)*. Springer, 2017, pp. 262–275.

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

[37] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," in *International Conference on Learning Representations (ICLR)*, 2017.

[38] F. Muratore, F. Treede, M. Gienger, and J. Peters, "Domain randomization for simulation-based policy optimization with transferability assessment," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, 2018, pp. 700–713.

[39] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 2817–2826.

[40] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[41] W. Uther and M. Veloso, "Adversarial reinforcement learning," In Proceedings of the AAAI Fall Symposium on Model Directed Autonomous Systems, Tech. Rep., 1997.

[42] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," 2020.

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

[44] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," in *Network and Distributed Systems Security Symposium (NDSS)*. The Internet Society, 2018.

[45] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, pp. 1437–1480, 2015.

[46] M. Heger, "Consideration of risk in reinforcement learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds. San Francisco (CA): Morgan Kaufmann, 1994, pp. 105 – 111.

[47] A. Tamar, "Risk-sensitive and efficient reinforcement learning algorithms," Ph.D. dissertation, Technion - Israel Institute of Technology, Faculty of Electrical Engineering, 2015.

[48] P. Geibel, "Risk-sensitive approaches for reinforcement learning," Ph.D. dissertation, University of Osnabrück, 2006.

[49] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," in *Nature*. Nature Publishing Group, a division of Macmillan Publishers Limited., 2015, vol. 518.

[50] M. Everett and J. How, "Gym: Collision avoidance," https://github.com/mit-acl/gym-collision-avoidance, 2020.

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

[52] J. P. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *International Symposium on Robotics Research (ISRR)*, 2009.

[53] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems (NeurIPS) 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959.

[54] A. G. Barto, R. S. Sutton, and C. W. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems,” *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.

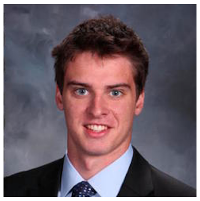
[55] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, “Stable baselines,” https://github.com/hill-a/stable-baselines, 2018.

[56] H. Zhang, H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh, “Crown-ibp: Towards stable and efficient training of verifiably robust neural networks,” https://github.com/huanzhang12/CROWN-IBP, 2020.

[57] T. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, “Adversarial patch,” *Conference on Neural Information Processing Systems (NeurIPS) Workshop*, 2017.

[58] E. Wong, F. R. Schmidt, and J. Z. Kolter, “Wasserstein adversarial examples via projected sinkhorn iterations,” vol. 97, 2019.

[59] Y.-S. Wang, T.-W. Weng, and L. Daniel, “Verification of neural network control policy under persistent adversarial perturbation,” *NeurIPS Workshop on Safety and Robustness in Decision Making*, 2019.



Michael Everett is a Ph.D. Candidate at MIT and conducts research in the Aerospace Controls Laboratory. He received the SM degree (2017) and the SB degree (2015) from MIT in Mechanical Engineering. His research addresses fundamental gaps in the connection of machine learning and real mobile robotics, with recent emphasis on developing the theory of safety/robustness of learned modules. He was an author of works that won the Best Paper Award on Cognitive Robotics at IROS 2019, the Best Student Paper Award and finalist for the Best Paper Award on Cognitive Robotics at IROS 2017, and finalist for the Best Multi-Robot Systems Paper Award at ICRA 2017. He has been interviewed live on the air by BBC Radio and his team’s robots were featured by Today Show, Reuters, and the Boston Globe.

Björn Lütjens is currently a Ph.D. Candidate in the Human Systems Laboratory of the Department of Aeronautics and Astronautics at MIT. He has received the S.M. degree in Aeronautics and Astronautics from MIT in 2019 and the B.Sc. degree in Engineering Science from Technical University of Munich in 2017. His research interests include deep reinforcement learning, bayesian deep learning, and climate and ocean modeling.

Jonathan P. How is the Richard C. Maclaurin Professor of Aeronautics and Astronautics at the Massachusetts Institute of Technology. He received a B.A.Sc. (aerospace) from the University of Toronto in 1987, and his S.M. and Ph.D. in Aeronautics and Astronautics from MIT in 1990 and 1993, respectively, and then studied for 1.5 years at MIT as a postdoctoral associate. Prior to joining MIT in 2000, he was an assistant professor in the Department of Aeronautics and Astronautics at Stanford University. Dr. How was the editor-in-chief of the IEEE Control Systems Magazine (2015-19) and is an associate editor for the AIAA Journal of Aerospace Information Systems and the IEEE Transactions on Neural Networks and Learning Systems. He was an area chair for International Joint Conference on Artificial Intelligence (2019) and will be the program vice-chair (tutorials) for the Conference on Decision and Control (2021). He was elected to the Board of Governors of the IEEE Control System Society (CSS) in 2019 and is a member of the IEEE CSS Technical Committee on Aerospace Control and the Technical Committee on Intelligent Control. He is the Director of the Ford-MIT Alliance and was a member of the USAF Scientific Advisory Board (SAB) from 2014-17. His research focuses on robust planning and learning under uncertainty with an emphasis on multiagent systems, and he was the planning and control lead for the MIT DARPA Urban Challenge team. His work has been recognized with multiple awards, including the 2020 AIAA Intelligent Systems Award. He is a Fellow of IEEE and AIAA.