

Certifying Safety in Reinforcement Learning under Adversarial Perturbation Attacks

Anonymous submission

Abstract

Function approximation has enabled remarkable advances in applying reinforcement learning (RL) techniques in environments with high-dimensional inputs, such as images, in an end-to-end fashion, mapping such inputs directly to low-level control. Nevertheless, these have proved vulnerable to small adversarial input perturbations. A number of approaches for improving or certifying robustness of end-to-end RL to adversarial perturbations have emerged as a result, focusing on cumulative reward. However, what is often at stake in adversarial scenarios is the violation of fundamental properties, such as safety, rather than the overall reward that combines safety with efficiency. Moreover, properties such as safety can only be defined with respect to true state, rather than the high-dimensional raw inputs to end-to-end policies. To disentangle nominal efficiency and adversarial safety, we situate RL in deterministic partially-observable Markov decision processes (POMDPs) with the goal of maximizing cumulative reward subject to safety constraints. We then propose a *partially-supervised reinforcement learning (PSRL)* framework that takes advantage of an additional assumption that the true state of the POMDP is known at *training* time. We present the first approach for certifying safety of PSRL policies under adversarial input perturbations, and two adversarial training approaches that make direct use of PSRL. Our experiments demonstrate both the efficacy of the proposed approach for certifying safety in adversarial environments, and the value of the PSRL framework coupled with adversarial training in improving certified safety while preserving high nominal reward and high-quality predictions of true state.

1 Introduction

Recent years have seen remarkable advances in reinforcement learning (RL) techniques using deep neural networks to represent either value and Q functions, or policies, with applications ranging from autonomous driving to healthcare (Panesar 2019; Kiran et al. 2021). As we aspire to transition advances to practice in high-stakes applications, however, safety becomes a critical concern. Indeed, a series of demonstrations have shown that small perturbations to observations that constitute inputs into the deep neural networks can successfully subvert learned policies to nearly arbitrary adversarial ends (Pattanaik et al. 2017; Sun et al. 2020). In response, a number of approaches have been proposed to augment traditional reinforcement learning algorithms in order to improve empirical robustness (Lütjens,

Everett, and How 2020; Oikarinen et al. 2021; Wu and Vorobeychik 2022; Zhang et al. 2020). In parallel, several recent efforts proposed techniques for *certifying* robustness of learned policies to adversarial perturbations (Kumar, Levine, and Feizi 2022; Wu et al. 2022).

What seems notably missing in prior work on robustness of RL to adversarial perturbations is any explicit consideration of safety, except as abstractly embedded in the reward signal along with other important, but conceptually orthogonal, aspects such as efficiency. Conflating these two issues is consequential: in practice, adversarial events are quite rare, and when they occur, we are most concerned about safety, rather than efficiency. On the other hand, safety verification is a central concern in formal analysis of dynamical systems and control (Duggirala et al. 2015; Fan et al. 2017; Chen, Ábrahám, and Sankaranarayanan 2013; Bayen, Cruck, and Tomlin 2002; Huang et al. 2014). An important feature in the latter is that state is relatively low-dimensional and interpretable, so that safety properties can be specified with respect to it. In contrast, robust reinforcement learning and robustness certification methods have been designed for end-to-end approaches that map high-dimensional uninterpretable perceptual inputs directly into low-level controls. However, as safety properties cannot be meaningfully specified over such inputs, it is not explicitly considered.

To bridge the gap between robustness certification approaches in adversarial RL and verification methods in dynamical systems and control, we begin by explicitly modeling the environment as a deterministic partially-observable Markov decision process (POMDP). States in this POMDP are low-dimensional and interpretable (and, therefore, amenable to safety specification); observations are high-dimensional and uninterpretable. Thus, conventional control would verify properties of the true state of this POMDP (usually, with additional assumptions about its structure), whereas conventional end-to-end RL would map observations directly to actions, effectively treating observations as state. Like conventional verification, we define safety with respect to the true state, and set the goal of the agent as maximizing cumulative reward over a finite horizon subject to safety constraints. In addition, although the true state of a POMDP is not observable at decision time, we assume that it can be observed during *training*. For any safety-critical application, this assumption is natural, as one

must carefully log and annotate the training phase just for the purposes of testing and debugging.

Our first contribution is a partially-supervised reinforcement learning framework (PSRL) that couples supervised learning used to predict state from observations with reinforcement learning that learns a control policy over a low-dimensional *predicted* system state. Supervision can in principle be done independently from RL (e.g., one can use a pretrained prediction model, e.g., mapping an image to locations of obstacles on the road), but it would typically be best to at least fine-tune such models during RL to better match performance to the policy-induced input distribution. The PSRL framework can be viewed as a simplified model of the far more complex compositions of perception, learning and control, that are commonly adopted in safety-critical environments, such as autonomous driving.

Our second contribution is an approach for adversarial safety certification of PSRL policies over a finite time horizon. Specifically, we develop an algorithm that computes a lower bound on the greatest magnitude of adversarial perturbations to *observations* such that safety constraints (specified with respect to true *states*) are not violated.

Finally, we propose adversarial training approaches that make use of the PSRL framework. An important tension in this context is between end-to-end adversarial training updates, which is most directly tuned to decisions, and adversarial training updates to the supervised and RL components, which most leverages the available supervised signal. We investigate several variations of adversarial training that balance these considerations in alternative ways, and find that a hybrid approach that combines end-to-end and supervised updates yields a good balance of high nominal reward, adversarial certified safety, and accurate state predictions.

Related Work Our work is situated in the broad area of adversarial machine learning, which is concerned with adversarial tampering with data used for either learning or predictions (Huang et al. 2011; Vorobeychik and Kantarcioglu 2018). The most significant progress in this field has been in the context of supervised learning, with a broad array of approaches proposed to improve robustness of learning to adversarial perturbations to either training data (Kearns and Li 1993; Long and Servedio 2011; Liu et al. 2017) or to inputs at prediction time (Cai, Liu, and Song 2018; Madry et al. 2018). The most prominent among these have been variations of adversarial training, which aims to improve robustness to attacks at prediction time by including adversarially perturbed inputs during training (Madry et al. 2018).

While many of the approaches for improving ML robustness focus on empirical robustness measures, a number aspire to formally verify robustness to adversarial perturbations (Chiang et al. 2020; Cohen, Rosenfeld, and Kolter 2019; Wang et al. 2021; Xu et al. 2021), and to train models that exhibit improved *verified* or *certified*, rather than merely *empirical* robustness (Salman et al. 2019; Wong and Kolter 2018). Our work is most connected to this latter line of work.

Just as in the case of supervised learning, modern reinforcement learning approaches have also been shown to suffer from adversarial perturbations (Pattanaik et al. 2017; Sun et al. 2020). A number of approaches have been proposed

to improve RL robustness (Lütjens, Everett, and How 2020; Oikarinen et al. 2021; Wu and Vorobeychik 2022; Zhang et al. 2020), although the efficacy of these varies a great deal by domain. Several recent efforts address the robustness certification problem in reinforcement learning by adapting the randomized smoothing technique from supervised to reinforcement learning (Kumar, Levine, and Feizi 2022; Wu et al. 2022). However, these are limited to certifying either policy invariance or reward lower bounds, rather than safety.

There is a rich literature in control theory and formal methods tackling the settings of adversarial perturbations of states and observations of dynamical systems. Research problems such as stability, reachability, robustness, formal safety verification, and correct-by-construction control synthesis in such settings have been extensively studied in the past few decades resulting in novel theoretical complexity results, algorithms, tools, and successful applications to a wide range of models (Mitra 2021; Tabuada 2009; Duggirala et al. 2015; Fan et al. 2017; Chen, Ábrahám, and Sankaranarayanan 2013; Frehse et al. 2011; Bak and Duggirala 2017; Althoff 2015; Meyer, Devonport, and Arcak 2019). However, this line of work mostly focused on the challenges in verification and certification arising from the complexity of the dynamics. For simplicity, the resulting algorithms commonly assumed that the dynamical systems under analysis are closed, i.e., have no inputs, or with low dimensional ones. More recently, a promising line of research resulted in methods for verifying safety properties input-output inclusion properties of various neural networks (NN) and for dynamical systems with machine-learned, usually NN, controllers (Ivanov et al. 2019; Dutta, Chen, and Sankaranarayanan 2019; Tran et al. 2020; Huang et al. 2019). Similar to the non-learned case, these works assumed ML models with low-dimensional inputs and relatively small number of parameters. There is an emerging line of research focused on assuring the robustness and safety of control systems with perception modules with high-dimensional sensor inputs (Dean et al. 2020b,a; Dean and Recht 2021; Santa Cruz and Shoukry 2022; Katz et al. 2022). So far, the results are restricted to scenarios where the map from sensor inputs to semantic outputs of the perception module is smooth, the state remains near training data of the perception module, an accurate generative model mapping semantic states to realistic high-dimensional sensor inputs is available, or an accurate model of the environment is known.

2 Model

We build on the formal model of deterministic finite-horizon *partially observable Markov decision processes (POMDPs)*. In Section 6, we partially relax the determinism assumption. A POMDP is defined by a tuple (S, A, O, r, τ, f, d) , where S is the state space, A is the set of actions, O is the set of observations, $r(s, a)$ is the reward function, which is a function of the current state s and action a , $\tau(s, a)$ is the (deterministic) transition function that returns the next state $s' = \tau(s, a)$, $f(s)$ is the deterministic observation function, with $o = f(s)$ for observation o and state s , and d is the probability distribution over the initial states. Moreover, we

consider the setting where the agent must also abide by a *safety constraint* that certain unsafe states should never be visited. Let U denote the set of all unsafe states s .

We assume that d, P, f , and r are unknown, and the agent must therefore learn how to act in this POMDP environment. In addition, we assume that the state space $S \subseteq \mathbb{R}^m$ and the observation space $O \subseteq \mathbb{R}^n$, where $n \gg m$. This setup captures settings in which the true state s is comprised of a collection of semantically meaningful variables, while the observations $o = f(s)$ are based on high-dimensional perceptual inputs, such as images, LiDAR point clouds, etc.

At any point in time t , the agent knows the history of observations, $h_t = \{o_0, \dots, o_t\}$. In general, a policy π depends on the full history of observations. If the state and observation spaces are finite, one can use *belief state* as a sufficient statistic. However, this is difficult in our setting since observations are high-dimensional. Consequently, a common approach in applying deep reinforcement learning to partially-observable environments with high-dimensional observations is to condition policies only on a finite sequence of preceding observations. To simplify exposition, we assume that policies depend only on the latest observation o ; extension of our approaches to dependence on finite histories is straightforward. We denote a policy by $\pi(o)$, mapping an observation o to an action a . By treating o effectively as a proxy for the true state s , we can now apply any standard reinforcement learning method directly; and, indeed, this has been done with considerable success (Borjarski et al. 2016; Peng et al. 2021; Zhang 2019).

Define $H_T(\pi) = \{o_0, s_0, o_1, s_1, \dots, o_T, s_T\}$ to be the sequence of observations and states induced by a policy π , and suppose that $S_T(\pi) \subset H_T$ be the set of states in $H_T(\pi)$. The goal of the agent is to maximize the total expected sum of rewards over a finite time horizon T , subject to the constraint that no unsafe states are visited along the way. Formally, the agent’s goal is to solve

$$\max_{\pi} \mathbb{E} \left[\sum_{t=0}^T r_t | \pi \right] \quad \text{s.t. :} \quad U \cap S_T(\pi) = \emptyset, \quad (1)$$

where the expectation is with respect to any randomness in the agent’s decisions and initial states.

Our primary focus is robustness to adversarial perturbations to observations o . Specifically, let δ be the adversarial perturbation which results in the observed state $o' = o + \delta$. As is common (Vorobeychik and Kantarcioglu 2018), we assume that the magnitude of δ is constrained to be in an ℓ_p ϵ -ball, i.e., $\|\delta\|_p \leq \epsilon$, for an exogenously specified $\epsilon \geq 0$ and norm $p \in \mathbb{N} \cup \{\infty\}$. The adversary’s goal is to cause the violation of the safety constraint over the finite POMDP horizon. To formalize, let $\Delta = \{\delta_0, \dots, \delta_{T-1}\}$ be the sequence of adversarial perturbations, $H_T(\pi, \Delta)$ be the history of the perturbed sequence of observations and states, and $S_T(\pi, \Delta)$ be the associated subset of states. The adversary’s goal is to identify a perturbation sequence Δ such that $\|\delta_t\|_p \leq \epsilon$ for all t , and $U \cap S_T(\pi, \Delta) \neq \emptyset$.

Note that we need not take the “adversarial” part of adversarial perturbations literally, as this model can capture both malicious tampering, as well as unanticipated errors in observations that do not fit an assumed distributional model.

3 Partially-Supervised Reinforcement Learning

In modeling the underlying environment as a deterministic POMDP, we have introduced an explicit way to refer to (true/semantic) states s over which safety constraints U are defined. One can still, in principle, make use of conventional deep RL techniques to learn policies π that map observed inputs to low-level control $a \in A$. However, there are a host of reasons why this is undesirable in safety-critical domains. First, when things go wrong, it is difficult to use the policy $\pi(o)$ to yield actionable insights into why, as it does not provide sufficient semantic information. Second, there are often large datasets from complementary domains that allow us to learn semantically meaningful mappings $g(o)$ from sensory inputs, such as images, to states, such as the location of the ego vehicle, as well as other objects and vehicles in the scene. Consequently, it is common in such domains to eschew end-to-end RL in favor of approaches that compose perceptual learning and reasoning with learning how to act.

We propose a *partially-supervised reinforcement learning (PSRL)* framework as a stylized way to capture the typically far more complex compositional approaches for learning to act in high-stakes domains. The key idea in PSRL is to take advantage of many situations in which the true state s is *known at training time*, and only unknown at decision time. For example, if we are to train an autonomous vehicle using simulations, the true state is actually available. It is only as we transition a working vehicle to a physical environment that it must navigate with only perceptual information.

In PSRL, we construct a policy $\pi(o)$ as a composition of perceptual prediction module $g(o)$ and a semantic policy $\pi_s(s)$ which maps the predicted state s to an action. Thus, the goal in PSRL is to learn both g and π . At decision time, we implement $\pi(o) = \pi_s(g(o))$. Significantly, we can apply supervised learning to learn $g(o; \theta_g)$, which maps high-dimensional observations o to relatively low-dimensional states s , and can apply standard RL to learn $\pi_s(s; \theta_s)$ over the low-dimensional predicted states s . Below, we make use of standard regression loss for g , with l_2 loss $l(g(o), s) = \|g(o) - s\|_2^2$, and use DDQN (Van Hasselt, Guez, and Silver 2016) deep Q-learning approach to learn an action-value function $Q(s, a)$, with $\pi_s(s) = \arg \max_a Q(s, a)$. However, any standard approach for these would work. During each training iteration, we update both g using the supervised loss and Q using the standard DDQN approach.

In the sequel, we address two key problems in using PSRL to obtain high-quality policies that remain safe even under adversarial perturbations to high-dimensional inputs o :

1. Formally certify safety properties of learned policies at decision time with respect to adversarial perturbations.
2. Learn policies with strong certified safety guarantees under adversarial perturbations.

4 Certifying Safety under Adversarial Perturbations

We begin with the problem of certifying safety of PSRL policies, $\pi(o) = \pi_s(g(o))$. The goal of robustness certi-

cation in supervised learning is to either ascertain whether a prediction is robust (i.e., invariant) under a given perturbation budget ϵ , or to certify a budget ϵ so that no perturbation under this budget constraint can possibly cause the prediction to change. Two extensions of this idea to reinforcement learning involve either certifying that a policy is invariant under adversarial perturbations, or certifying a lower bound on total reward. In our context, however, our sole concern is to certify *safety* under adversarial perturbations. The reason is that we view adversarial encounters as exceedingly rare, and expect the nominal policy to work well in most cases when it is deployed (as otherwise, it would not even be deployed). However, when these do occur, we are typically far less concerned with issues such as efficiency (which is what rewards most naturally capture) and far more concerned with issues of safety (that the vehicle, for example, doesn't crash).

We formally define the *adversarial safety certification* (ASC) problem as follows:

Given a policy π , identify the highest $\epsilon^(\pi)$ such that $U \cap S_T(\pi, \Delta) = \emptyset$ for all Δ such that $\|\delta_t\|_p \leq \epsilon^*(\pi)$ and for all $0 \leq t \leq T - 1$.*

In practice, it will typically be infeasible to actually find $\epsilon^*(\pi)$, and our goal will be to return a provable lower bound $\bar{\epsilon} \leq \epsilon^*(\pi)$. If even the nominal (that is, unperturbed) policy is unsafe, $\epsilon^*(\pi) \equiv 0$. Clearly, the ASC problem definition requires the knowledge of whether a state s encountered is unsafe. We assume that this information is easy to obtain when one actually enters an unsafe state (e.g., when the car crashes, it is not difficult to verify that it has crashed).¹

We assume that we can simulate transition $\tau(s, a)$ for any state s and action a . This is a significantly weaker assumption than those commonly made in verification of dynamical systems literature, where either the dynamical model is assumed to be fully known, or one assumes specific structure (such as discrepancy functions) (Fan et al. 2017; Hsieh et al. 2021; Santa Cruz and Shoukry 2022; Sun, Khedr, and Shoukry 2019). Next, we present our algorithm for adversarial safety certification, TASC (tree-based adversarial safety certification algorithm).

Consider a given observation o , policy $\pi_s(s)$, state prediction function $g(o)$, and a bound on adversarial perturbation ϵ . The key building block in the certification procedure is to return a set of actions $A_\epsilon(o)$ with the property that the adversary cannot cause the policy to select an action not in that set $A \setminus A_\epsilon(o)$. In the PSRL framework, this entails two steps: first, we certify a set of states $S_\epsilon(o)$ such that $g(o) \in S_\epsilon(o)$, and second, we certify a set of actions $A_\epsilon(o) = A(S_\epsilon(o))$ as the only actions that any state $s \in S_\epsilon(o)$ can induce. Our approach is to treat these two steps as separate robustness certification problems under ℓ_p -bounded input perturbations. Specifically, our certification of the policy π_s takes the smallest ℓ_∞ -norm γ such $S_\epsilon(o) \subseteq B_\gamma(s)$, where $B_\gamma(s)$

is a γ -ball around the predicted state $s = g(o)$. We can then use, e.g., the α, β -CROWN approach (Xu et al. 2021; Wang et al. 2021) to calculate the upper/lower bound of Q values, $\bar{Q}(s, a)$ and $\underline{Q}(s, a)$ for each action a for $s = g(o)$, on which π_s is based (recall that we assume the use of Q -learning). We can then use the bounds on the Q values to determine the action set $A_\epsilon(o)$ as the set of all actions a for which $\bar{Q}(s, a) \geq \max_{a'} \underline{Q}(s, a')$.

For the problem of certifying $g(o)$ above, we appeal to two distinct approaches depending on the norm ℓ_p . Specifically, if we consider adversarial perturbations to observations o to be bounded in ℓ_∞ norm, we can apply, e.g., the α, β -CROWN approach. On the other hand, if perturbations are bounded in ℓ_2 norm, we can make use of median smoothing (Chiang et al. 2020) to obtain interval bounds for each coordinate of s , which is now the median of the smoothed prediction. Either approach in fact directly yields an ℓ_∞ -ball $B_\gamma(s)$ that we can then use to certify the policy π_s as described above.

Armed with this setup for certifying an action set for a given observation and ϵ , we can now describe our full ASC algorithm. ASC proceeds through a pre-defined discrete set $\{0, \epsilon_1, \epsilon_2, \dots, \epsilon_N\}$ (e.g., by discretizing ϵ between 0 and 1 if inputs are normalized to the $[0, 1]$ interval). Let T_v be the verification horizon (which may be different from the reward horizon T), and consider the nominal history $H_{T_v}(\pi)$ and associated set of states $S_{T_v}(\pi)$ generated by the policy π we are verifying from a known initial state s_0 .² The first step is to check whether $U \cap S_{T_v}(\pi) = \emptyset$; if not, we simply return $\epsilon^*(\pi) = 0$. Otherwise, we proceed through a series of iterations of certification.

In each iteration i , we know to have verified $\bar{\epsilon} = \epsilon_{i-1}$, and we attempt to verify ϵ_i . In addition, through iterations $0, \dots, i-1$, we have expanded a tree defined by sequences of actions beginning from the initial state s_0 (root node of the tree). This tree is typically a very sparse subtree of the full tree defined by all possible T_v -long sequences of actions starting with s_0 . Each node r in this tree corresponds to a state s_r (equivalently, a sequence of actions taken starting with s_0), as well as the associated observation o_r . In addition, for each node r we have previously verified a set of actions A_r^{i-1} . Since we know the initial state, we begin with the node that follows, and proceed along the nominal path $H_T(\pi)$. For each node r thereby encountered, we certify the set of actions A_r^i under ϵ_i . If $A_r^i = A_r^{i-1}$, we can update the certificate for node r to ϵ_i (and do not need any additional exploration of the subtree rooted at r). If $A_r^i \neq A_r^{i-1}$, let $A_{r,i}^i$ be the set of new actions added by ϵ_i as compared to ϵ_{i-1} . Each action $a \in A_{r,i}^i$ induces a nominal history $H_{T(r)}(\pi, r)$, where $T(r)$ is the remaining verification horizon starting at node r , with associated set of states $S_{T(r)}(\pi, r)$. If $U \cap S_{T(r)}(\pi, r) \neq \emptyset$, we immediately return $\bar{\epsilon} = \epsilon_{i-1}$, since we can no longer verify safety for ϵ_i . Otherwise, we expand the subtree rooted at r one node at a time in a depth-first fashion, certifying an action set for

¹In the case where we only simulate potential future during decision-time certification, we can alternatively leverage supervised learning to predict whether we have entered an unsafe state. There is no concern about adversarial perturbations here, since this is done only for the purposes of certification and does not involve actual interaction with the environment.

²The assumption that the initial state is known is without loss of generality, since we can always define a dummy initial state before the beginning of the scenario.

each node under ϵ_i , and return as soon as any unsafe state is encountered. If the entire subtree is expanded without observing an unsafe state, we verify ϵ_i for node r , and continue along the nominal trajectory. If we successfully complete T_v steps along the nominal trajectory in this fashion, we can now verify $\bar{\epsilon} = \epsilon_i$, and we continue with the next iteration. Algorithm 1 provides the full details of TASC.

Algorithm 1: TASC.

Input: $g, \pi_s, T_v, s_0, \{\epsilon_i\}$

Output: $\bar{\epsilon}$ // certified adversarial safety bound

- 1: Build and initialize the clean tree.
 - 2: $i \leftarrow 0, \epsilon_0 \leftarrow 0$
 - 3: **while** True **do**
 - 4: $\bar{\epsilon} \leftarrow \epsilon_i, i \leftarrow i + 1$ // when exit successfully, output $\bar{\epsilon}$
 - 5: CHECKTREE(RootNode, ϵ_i)
 - 6: **end while**
-

Algorithm 2: CHECKTREE.

Input: Node r, ϵ_i

- 1: **if** Depth(r) is $T_v - 1$ **then**
 - 2: $A_{r,i}^i \leftarrow \text{CERT}(\pi, o_r, \epsilon_i)$
 - 3: **if** any $a \in A_{r,i}^i$ leads to an unsafe state **then**
 - 4: **exit** // successfully terminate program
 - 5: **end if**
 - 6: **return**
 - 7: **else**
 - 8: **for** r' in r .Children **do**
 - 9: CHECKTREE(r', ϵ_i)
 - 10: **end for**
 - 11: $A_{r,i}^i \leftarrow \text{CERT}(\pi, o_r, \epsilon_i)$
 - 12: **if** $A_{r,i}^i$ is not empty **then**
 - 13: GROWSUBTREE($r, A_{r,i}^i, \epsilon_i$)
 - 14: **end if**
 - 15: **end if**
-

TASC has three important properties. First, it is straightforward to make it “real-time”, that is, to have it return a certified adversarial safety bound $\bar{\epsilon}$ after an arbitrary time limit. Second, as long as the procedure for certifying a set of actions $A_\epsilon(o)$ is sound, TASC is also sound. Third, if $A_\epsilon(o)$ is also complete, then TASC is complete. Next, we formalize the latter two properties.

Let $\text{CERT}(\pi, o, \epsilon)$ return a set of action $A_\epsilon(o)$ that are certified robust for policy π in observation o under a given bound on adversary’s budget ϵ for some given norm ℓ_p .

Definition 4.1. $\text{CERT}(\pi, o, \epsilon)$ is sound if there does not exist an action $a' \notin A_\epsilon(o)$ where $\exists \delta$ with $\|\delta\|_p \leq \epsilon$ such that $\pi(o + \delta) = a'$. It is complete if $\forall a' \in A, \exists \delta : \pi(o + \delta) = a'$ and $\|\delta\|_p \leq \epsilon \Rightarrow a' \in A_\epsilon(o)$. An ASC algorithm is sound if it returns $\bar{\epsilon} \leq \epsilon^*(\pi)$. It is γ -complete if $\epsilon^*(\pi) - \bar{\epsilon} \leq \gamma$.

Proposition 1. If $\text{CERT}(\pi, o, \epsilon)$ is sound, then TASC is sound. If $\text{CERT}(\pi, o, \epsilon)$ is also complete, then TASC is γ -complete, where $\gamma = \max_{i \geq 1} (\epsilon_i - \epsilon_{i-1})$ for the finite discrete schedule of adversarial noise bounds $\{\epsilon_i\}$, where ϵ_N is the absolute upper bound.

Algorithm 3: GROWSUBTREE.

Input: Node $r, A_{r,i}^i, \epsilon_i$

- 1: ExpandQueue $\leftarrow []$ // initialize queue
 - 2: **for** action $a \in A_{r,i}^i$ **do**
 - 3: Take action a to reach next state s'_r (observation o'_r).
 - 4: **if** s'_r is an unsafe state **then**
 - 5: **exit**
 - 6: **end if**
 - 7: Create node r_{new} with state s'_r .
 - 8: $r_{\text{curr}}.appendChildren(r_{\text{new}})$
 - 9: ExpandQueue.enqueue(r_{new})
 - 10: **end for**
 - 11: **while** ExpandQueue is not empty **do**
 - 12: $r_{\text{curr}} \leftarrow \text{ExpandQueue.dequeue}()$ // FIFO
 - 13: $A_{r_{\text{curr}},i}^i \leftarrow \text{CERT}(\pi, o_{r_{\text{curr}}}, \epsilon_i)$
 - 14: **if** Depth(r_{curr}) is $T_v - 1$ **then**
 - 15: **if** any $a \in A_{r_{\text{curr}},i}^i$ leads to an unsafe state **then**
 - 16: **exit**
 - 17: **end if**
 - 18: **else**
 - 19: **for** action $a \in A_{r_{\text{curr}},i}^i$ **do**
 - 20: Take action a to reach next state $s'_{r_{\text{curr}}}$.
 - 21: **if** $s'_{r_{\text{curr}}}$ is an unsafe state **then**
 - 22: **exit**
 - 23: **end if**
 - 24: Create node r_{new} with state $s'_{r_{\text{curr}}}$.
 - 25: $r_{\text{curr}}.appendChildren(r_{\text{new}})$
 - 26: ExpandQueue.enqueue(r_{new})
 - 27: **end for**
 - 28: **end if**
 - 29: **end while**
-

Proof Sketch. Suppose $\text{CERT}(\pi, o, \epsilon)$ is sound, and let TASC returns $\bar{\epsilon} = \epsilon_i$ for some i in the discrete sequence of adversarial noise bounds. Consider the tree expanded before the algorithm moves forward with ϵ_{i+1} . Since $\text{CERT}(\pi, o, \epsilon)$ is sound, the tree has expanded all actions in $A_\epsilon(o)$ for every node o through depth T_v , and there can be no more actions to expand. Consequently, the tree contains all nodes that the adversary could reach with $\bar{\epsilon}$. Since TASC did not identify any of these to be unsafe, soundness follows.

For completeness, consider the same setting as above, and note that either $i = N$, in which case it’s complete since ϵ_N is the absolute upper bound, or $i < N$, in which case a subtree generated for ϵ_{i+1} was found to contain an unsafe state. Since $\text{CERT}(\pi, o, \epsilon)$ is complete, it follows that no actions were expanded unnecessarily. Thus, there exists a path from the starting state to an unsafe state that an adversary can induce with perturbations up to ϵ_{i+1} . This means that $\epsilon^*(\pi) \in [\epsilon_i, \epsilon_{i+1})$, and the result follows. $\square$

5 Learning Robust Policies

A natural complement to certification is the problem of bolstering robustness through learning. As mentioned above, the goal of conventional learning is to maximize total reward subject to the safety constraints (Equation (1)). This learning problem, however, is focused solely on nominal re-

ward, and only avoids safety violations on the nominal (that is, prior to adversarial perturbations) sequence of states. In certified robust learning, we add a third goal: to maximize the certification bound $\epsilon^*(\pi)$ of the learned policy π . Thus, the combined problem is

$$\max_{\pi: U \cap S_T(\pi, \Delta) = \emptyset} \alpha \mathbb{E} \left[\sum_{t=0}^T r_t | \pi \right] + (1 - \alpha) \epsilon^*(\pi), \quad (2)$$

where $\Delta = \{\delta_t\}$ with $\|\delta_t\|_p \leq \epsilon^*$.

Note that Problem (2) is substantively different from the typical goal of robust reinforcement learning, which is to maximize worst-case sum of rewards, where certification would simply play the role of constructing a lower bound on the worst-case cumulative reward. We maintain, rather, that it is crucial to separate the primary goal operative *under normal circumstances*, which is to maximize the sum of nominal rewards, and robustness pertains primarily to critical considerations, such as violation of safety constraints, in the *rare* adversarial encounters.

One of the most basic techniques for boosting robustness of machine learning techniques—including reinforcement learning—is adversarial training (Madry et al. 2018; Oikarinen et al. 2021). In DDQN, adversarial training augments the DDQN loss with an adversarial counterpart. There are a number of ways to implement the adversarial loss, but the most basic is to replace the actor Q function, $Q_{\text{actor}}(o, a)$ (using observation o as a proxy for state, as is common) with its adversarial counterpart $Q_{\text{actor}}(o + \delta, a)$, where the adversarial perturbation δ is generated by solving $\min_{\delta} \sum_a \pi_a(o) Q_{\text{actor}}(o + \delta, a)$ using projected gradient descent, where $\pi_a(o)$ is the policy induced by the actor Q function by passing the Q function through a softmax layer (Wu and Vorobeychik 2022). We refer to this baseline version of adversarial training as simply AT.

An alternative, proposed by Oikarinen et al. (2021), is to use provably bounds on the actor Q -function obtained using interval bound propagation (IBP) (Gowal et al. 2018) to maximize the difference the lower bound on the Q function value for the action prescribed by the nominal policy and the upper bound Q value for every other action. We refer to this approach as RADIAL. Note that both AT and RADIAL effectively collapse the PSRL architecture into a single Q function (and, consequently, policy) architecture.

In addition, we propose two novel approaches to adversarial training that take advantage of the PSRL compositional architecture. The first, PSRL-AT, makes use of IBP in both the loss for the policy π_s (or, rather, the associated Q function), as well as the supervised loss for g . For the former, the approach is analogous to RADIAL. For the latter, the adversarial counterpart of the ℓ_2 loss becomes $\|g(o) - s\|_2^2 + \|\bar{g}(o) - s\|_2^2$, where g and $\bar{g}$ are the upper and lower bounds on g , respectively, which can be computed by IBP in the case of ℓ_∞ loss. In the case of ℓ_2 loss, we take an approach similar to Salman et al. (2019), combining randomized smoothing with the approach for generating bounds on the actor Q function described above.

Our second proposed approach is a hybrid of RADIAL and PSRL-AT. Specifically, we first use the RADIAL loss

function to update the full composite Q function (backpropagating all the way through the supervised component g). In addition, we use *only the adversarial supervised loss* g to perform a second update over the same minibatch. We refer to this approach as PSRL-HYBRID.

6 Generalizations

Our system model is a POMDP with a deterministic transition τ and observation f functions. We now show how our approach can be applied even when neither of these assumptions hold. The simpler case is deterministic transitions $o = f(s)$: in fact, nothing in the proposed PSRL, certification, or adversarial training needs this assumption. Rather, it suffices for our purposes that every observation $o \in O$ corresponds to a unique state s , but observations can indeed be stochastic given a state s . Our assumption of deterministic transition τ can also be relaxed. Specifically, it suffices for our purposes to assume that for each state s and action a , the probability distribution $\tau(s, a) = \Pr\{s' | s, a\}$ has a finite support $D(s, a)$. Then, whenever we expand an action a at a node corresponding to state s during verification, we expand the entire set $D(s, a)$, and reason over the induced subtree for each $s' \in D(s, a)$.

7 Experiments

Experiment Setup Our experiments use the *Highway* autonomous driving environment (Leurent 2018), which we customized to our setting (see the Supplement for details). We experiment on three scenarios within this environment: *Highway*, *Two-way*, and *Exit*. In our experiments, we assume that the position of the ego vehicle is known to slightly simplify the setup. In *Highway* scenario, all vehicles are heading in the same direction on a three-lane road, with vehicles other than the ego vehicle traveling at constant speed. The ego vehicle receives the reward of 0.1 for staying in the rightmost lane, 0.5 for high speed (between 20-30), and -1 for reaching an unsafe state, which we define as either collision or being within 1 unit of distance of other vehicles. In the *Two-way* scenario, the vehicles drive on a two-lane road in opposite directions. There are 3 vehicles at the same lane as the ego vehicle, and 2 in the opposite lane. The ego vehicle receives the reward of 0.5 for high speed, with the same penalty for unsafe states and collision as above. The *Exit* scenario is similar to the *Highway* scenario, except there is an exit lane appearing on the right, with a reward of 1 for a successful exit. We assume we know the distance to the exit.

Our safety certification uses $T_v \in \{5, 10, 15\}$. PSRL in our context involves two neural networks: $g(o)$, which maps the image input into the positions of other vehicles in the observed scene, and the Q network, which takes the predicted state s (i.e., predicted positions of two vehicles closest to ego vehicle in the scene) as input. However, we always use the true state to check for safety violation and collision. First, we use standard DDQN (Van Hasselt, Guez, and Silver 2016) to train the Q network. We train for 2000 episodes in the *Highway* scenario, for 3000 episodes for the *Two-way* scenario, and 50,000 steps for the *Exit* scenario. Next, we use the policy learned by DDQN to sequentially train g , using a γ -

greedy policy (i.e., following the learned policy $\pi_s(s)$ with probability $1 - \gamma$, and taking a random action otherwise), with parameter γ decreasing over time as in DDQN. We train g for 500 episodes in the *Highway* scenario, for 1500 episodes in the *Twoway* scenario, and for 1000 episodes in the *Exit* scenario.

For all our variants of adversarial training, we train for 40,000 steps. The ϵ for adversarial training is increased linearly over the first 25,000 steps and then remains at the target level thereafter. We consider three values for a target ϵ for adversarial training: $\{1, 2, 3\}/255$. Here we present the results for ℓ_∞ perturbations; results for ℓ_2 , and full details of the experiment setup, are provided in the Supplement.

Efficacy of TASC We begin by evaluating the efficacy of the proposed TASC adversarial safety certification algorithm. We set the maximum total number of nodes allowed to be explored to $M = 500$; however, this was saturated in only a few cases in our experiments, and does not affect the medians reported below. The median results for $T_v = 5$ are presented in Table 1. As we can see, even for a relatively

	Highway	Twoway	Exit
Vanilla	0.0423	0.0064	0.0064
AT	0.0154	0.0064	0.0102
PSRL-AT	0.0256	0.0102	0.0115
RADIAL	0.0384	0.0115	0.0077
PSRL-Hybrid	0.0371	0.0166	0.0090

Table 1: TASC: fraction of nodes explored ($T_v = 5$).

short horizon, TASC only needs to explore a small fraction of the search tree. This becomes even more pronounced as the horizon increases. For $T_v = 10$, TASC explores fewer than 0.0001 fraction of all nodes in all cases in all scenarios, while when $T_v = 15$, it explores fewer than 0.00001 fraction of nodes. The details are provided in the Supplement.

Efficacy of Adversarial Training Next, we evaluate the efficacy of the adversarial training approaches described in Section 5. We compare *PSRL* with no adversarial training, which we refer to as *Vanilla*, *AT* (standard adversarial training), *RADIAL* (Oikarinen et al. 2021) adapted to our setting by backpropagating adversarial error through the combined $\pi(o) = \pi_s(g(o))$ policy, *PSRL-AT*, and *PSRL-Hybrid*. We consider four measures of efficacy: nominal reward, certified safety, false action rate (fraction of time $\pi(o) \neq \pi_s(s)$), and three measures of mean squared error (MSE) of state prediction: prediction based on actual observation o (*avg err*), and upper (*avg err (ub)*) and lower (*avg err (lb)*) bounds of prediction error after adversarial perturbation with $\epsilon = 1/255$, computed by β -CROWN (Wang et al. 2021).

Figure 1 presents the results for nominal reward (left column) and certified safety (right column). *PSRL* has a reliably high nominal reward, but neither *PSRL* nor *AT* yield meaningful safety certification, as we would expect, with *AT* often also suffering in terms of nominal reward. *PSRL-Hybrid* consistently performs well on both nominal reward and safety certification, while both *RADIAL* and *PSRL-AT* vary in this regard, scoring the best in some cases, and rel-

atively poorly in others. Thus, the hybrid approach indeed appears to yield the best balance of nominal reward and certified safety across the three scenarios.

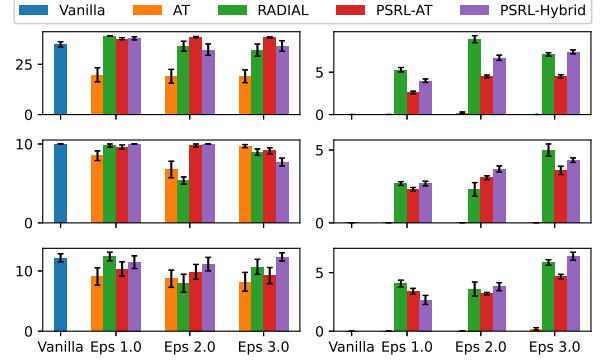


Figure 1: Averaged reward (left) and certified safety (right) for Highway (top), Twoway (middle) and Exit (bottom).

When it comes to false action rate and MSE, both *PSRL-AT* and *PSRL-Hybrid* typically outperform other adversarial training alternatives, and *PSRL-AT* in particular is far better than all alternatives. *PSRL*-style adversarial training methods thus yield the additional advantage of consistency in performance and interpretability (in correctly predicting state based on observations) over end-to-end approaches.

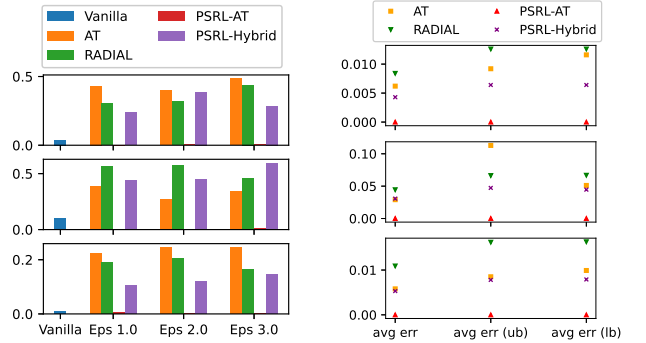


Figure 2: False action rate (left) and MSE of state prediction error (right) for Twoway (middle) and Exit (bottom).

8 Conclusion

We presented the first framework for adversarial safety certification in reinforcement learning in the context of a POMDP system model. We present a highly effective adversarial safety certification algorithm, TASC, as well as several adversarial training methods, several of which rely on a novel composition of supervised and reinforcement learning, *PSRL*, which leverages observations of true state at training time. We show that making use of *PSRL* in adversarial training can yield strong nominal performance, high adversarially certified safety, and also obtain high-quality predictions of state, enabling a high level of interpretability.

References

- Althoff, M. 2015. An Introduction to CORA 2015. In *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*.
- Bak, S.; and Duggirala, P. S. 2017. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 173–178. ACM.
- Bayen, A. M.; Cruck, E.; and Tomlin, C. 2002. Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: solving the Hamilton-Jacobi equation using viability techniques. In Tomlin, C.; and Greenstreet, M. R., eds., *HSCC*, volume 2289 of *LNCS*, 90–104. Springer. ISBN 3-540-43321-X.
- Bojarski, M.; Firner, B.; Flepp, B.; Jackel, L.; Muller, U.; Zieba, K.; and Testa, D. D. 2016. End-to-End Deep Learning for Self-Driving Cars. *NVIDIA Developer Technical Blog*.
- Cai, Q.-Z.; Liu, C.; and Song, D. 2018. Curriculum Adversarial Training. In *International Joint Conference on Artificial Intelligence*, 3740–3747.
- Chen, X.; Abraham, E.; and Sankaranarayanan, S. 2013. Flow*: An Analyzer for Non-linear Hybrid Systems. In Sharygina, N.; and Veith, H., eds., *Computer Aided Verification*, volume 8044 of *Lecture Notes in Computer Science*, 258–263. Springer Berlin Heidelberg.
- Chiang, P.-y.; Curry, M.; Abdelkader, A.; Kumar, A.; Dickerson, J.; and Goldstein, T. 2020. Detection as regression: Certified object detection with median smoothing. *Neural Information Processing Systems*, 33: 1275–1286.
- Cohen, J. M.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *International Conference on Machine Learning*.
- Dean, S.; Matni, N.; Recht, B.; and Ye, V. 2020a. Robust Guarantees for Perception-Based Control. In Bayen, A. M.; Jadbabaie, A.; Pappas, G.; Parrilo, P. A.; Recht, B.; Tomlin, C.; and Zeilinger, M., eds., *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, 350–360. PMLR.
- Dean, S.; and Recht, B. 2021. Certainty Equivalent Perception-Based Control. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, 399–411. PMLR.
- Dean, S.; Taylor, A. J.; Cosner, R. K.; Recht, B.; and Ames, A. 2020b. Guaranteeing Safety of Learned Perception Modules via Measurement-Robust Control Barrier Functions. *ArXiv*, abs/2010.16001.
- Duggirala, P. S.; Mitra, S.; Viswanathan, M.; and Potok, M. 2015. C2E2: A Verification Tool for Stateflow Models. In Baier, C.; and Tinelli, C., eds., *Tools and Algorithms for the Construction and Analysis of Systems*, 68–82. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Dutta, S.; Chen, X.; and Sankaranarayanan, S. 2019. *Reachability Analysis for Neural Feedback Systems Using Regressive Polynomial Rule Inference*, 157–168. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362825.
- Fan, C.; Qi, B.; Mitra, S.; and Viswanathan, M. 2017. DryVR: Data-Driven Verification and Compositional Reasoning for Automotive Systems. In *Computer Aided Verification*, 441–461. Springer International Publishing.
- Frehse, G.; Guernic, C. L.; Donzé, A.; Cotton, S.; Ray, R.; Lebeltel, O.; Ripado, R.; Girard, A.; Dang, T.; and Maler, O. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*.
- Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2018. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*.
- Hsieh, C.; Joshi, K.; Misailovic, S.; and Mitra, S. 2021. Verifying controllers with convolutional neural network-based perception: a case for intelligible, safe, and precise abstractions. *arXiv preprint arXiv:2111.05534*.
- Huang, C.; Fan, J.; Li, W.; Chen, X.; and Zhu, Q. 2019. ReachNN: Reachability Analysis of Neural-Network Controlled Systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s).
- Huang, L.; Joseph, A. D.; Nelson, B.; Rubinstein, B. I.; and Tygar, J. D. 2011. Adversarial machine learning. In *ACM workshop on Security and Artificial Intelligence*, 43–58.
- Huang, Z.; Fan, C.; Mereacre, A.; Mitra, S.; and Kwiatkowska, M. Z. 2014. Invariant Verification of Non-linear Hybrid Automata Networks of Cardiac Cells. In *CAV*, volume 8559 of *Lecture Notes in Computer Science*, 373–390. Springer.
- Ivanov, R.; Weimer, J.; Alur, R.; Pappas, G. J.; and Lee, I. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *ACM HSCC*.
- Katz, S. M.; Corso, A. L.; Strong, C. A.; and Kochenderfer, M. J. 2022. Verification of Image-Based Neural Network Controllers Using Generative Models. *Journal of Aerospace Information Systems*, 0(0): 1–11.
- Kearns, M.; and Li, M. 1993. Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22(4): 807–837.
- Kiran, B. R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A. A.; Yogamani, S.; and Pérez, P. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*.
- Kumar, A.; Levine, A.; and Feizi, S. 2022. Policy smoothing for provably robust reinforcement learning. In *International Conference on Learning Representations*.
- Leurent, E. 2018. An Environment for Autonomous Driving Decision-Making. <https://github.com/eleurent/highway-env>.
- Liu, C.; Li, B.; Vorobeychik, Y.; and Oprea, A. 2017. Robust linear regression against training data poisoning. In *ACM Workshop on Artificial Intelligence and Security*, 91–102.
- Long, P.; and Servedio, R. 2011. Learning large-margin half-spaces with more malicious noise. In *Neural Information Processing Systems*.

- Lütjens, B.; Everett, M.; and How, J. P. 2020. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*, 1328–1337.
- Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.
- Meyer, P.-J.; Devonport, A.; and Arcak, M. 2019. TIRA: Toolbox for Interval Reachability Analysis. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, 224–229. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362825.
- Mitra, S. 2021. *Verifying Cyber-Physical Systems: A Path to Safe Autonomy*. MIT Press. ISBN 978-0-262-04480-6.
- Oikarinen, T.; Zhang, W.; Megretski, A.; Daniel, L.; and Weng, T.-W. 2021. Robust deep reinforcement learning through adversarial loss. In *Neural Information Processing Systems*, 26156–26167.
- Panesar, A. 2019. *Machine learning and AI for healthcare*. Springer.
- Pattanaik, A.; Tang, Z.; Liu, S.; Bommannan, G.; and Chowdhary, G. 2017. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*.
- Peng, B.; Sun, Q.; Li, S. E.; Kum, D.; Yin, Y.; Wei, J.; and Gu, T. 2021. End-to-End autonomous driving through dueling double deep Q-network. *Automotive Innovation*, 4(3): 328–337.
- Salman, H.; Li, J.; Razenshteyn, I.; Zhang, P.; Zhang, H.; Bubeck, S.; and Yang, G. 2019. Provably robust deep learning via adversarially trained smoothed classifiers. In *Neural Information Processing Systems*.
- Santa Cruz, U.; and Shoukry, Y. 2022. NNlander-VeriF: A Neural Network Formal Verification Framework for Vision-Based Autonomous Aircraft Landing. In *NASA Formal Methods: 14th International Symposium, NFM 2022, Pasadena, CA, USA, May 24–27, 2022, Proceedings*, 213–230. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-031-06772-3.
- Sun, J.; Zhang, T.; Xie, X.; Ma, L.; Zheng, Y.; Chen, K.; and Liu, Y. 2020. Stealthy and efficient adversarial attacks against deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 34, 5883–5891.
- Sun, X.; Khedr, H.; and Shoukry, Y. 2019. Formal verification of neural network controlled autonomous systems. In *ACM International Conference on Hybrid Systems: Computation and Control*, 147–156.
- Tabuada, P. 2009. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer Publishing Company, Incorporated, 1st edition. ISBN 1441902236.
- Tran, H.-D.; Yang, X.; Manzanar Lopez, D.; Musau, P.; Nguyen, L. V.; Xiang, W.; Bak, S.; and Johnson, T. T. 2020. NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems. In Lahiri, S. K.; and Wang, C., eds., *CAV*. ISBN 978-3-030-53288-8.
- Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double Q-learning. In *AAAI Conference on Artificial Intelligence*.
- Vorobeychik, Y.; and Kantarcioglu, M. 2018. *Adversarial Machine Learning*. Morgan and Claypool.
- Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Neural Network Robustness Verification. In *Neural Information Processing Systems*.
- Wong, E.; and Kolter, J. Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*.
- Wu, F.; Li, L.; Huang, Z.; Vorobeychik, Y.; Zhao, D.; and Li, B. 2022. Crop: Certifying robust policies for reinforcement learning through functional smoothing. In *International Conference on Learning Representations*.
- Wu, J.; and Vorobeychik, Y. 2022. Robust Deep Reinforcement Learning through Bootstrapped Opportunistic Curriculum. In *International Conference on Machine Learning*, 24177–24211.
- Xu, K.; Zhang, H.; Wang, S.; Wang, Y.; Jana, S.; Lin, X.; and Hsieh, C.-J. 2021. Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers. In *International Conference on Learning Representations*.
- Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Liu, M.; Boning, D.; and Hsieh, C.-J. 2020. Robust deep reinforcement learning against adversarial perturbations on state observations. In *Neural Information Processing Systems*, volume 33, 21024–21037.
- Zhang, J. 2019. *End-to-End Learning for Autonomous Driving*. Ph.D. thesis, New York University, USA.

Supplementary Materials

A Results for l_2 Norm

For l_2 norm, we perform training with noise added to the image. More specifically, for $g(o)$ (mapping visual inputs to state in our experiments), we add noise $\eta \sim \sigma N(0, 1)$ to the image, where σ is the exogenously specified standard deviation, and push both the clean image and image with noise to the buffer. For adversarial training, every time we sample from the buffer and obtain (o, a, r, o') , where o is the current observation and o' is the next observation, we add the noise $\sigma N(0, 1)$ to o and then perform training either using IBP (in the case of RADIAL, PSRL-AT, and PSRL-Hybrid), or FGSM with a random start in the case of AT (Wong, Rice, and Kolter 2020).

For certifying $g(o)$, we use median smoothing in Lemma 1 (Chiang et al. 2020) to calculate the upper and lower bound for each feature of the predicted state s , and then use β -CROWN for certifying the policy π_s (since feature-by-feature median smoothing certificates effectively turn input perturbations to π_s into l_∞ -norm perturbations). We reproduce the median smoothing lemma here for completeness.

Lemma 1 (Chiang et al. (2020)). *A percentile-smoothed function h_p with adversarial perturbation δ can be bounded as*

$$h_{\underline{p}}(x) \leq h_p(x + \delta) \leq h_{\bar{p}}(x) \quad \forall \|\delta\|_2 < \epsilon, \quad (3)$$

where $h_p(x) = \{y \in \mathbb{R} \mid \mathbb{P}[f(x + G) \leq y] = p\}$, $G \sim N(0, \sigma^2 I)$ (the p -th percentile), $\underline{p} := \Phi(\Phi^{-1}(p) - \frac{\epsilon}{\sigma})$ and $\bar{p} := \Phi(\Phi^{-1}(p) + \frac{\epsilon}{\sigma})$, with Φ being the standard Gaussian CDF.

B Efficacy of l_2 Adversarially Trained Models

Figure 3 (left) presents nominal reward for both the PSRL (referred to as Vanilla) and the adversarial training methods for the l_2 -norm perturbations. The results are largely consis-

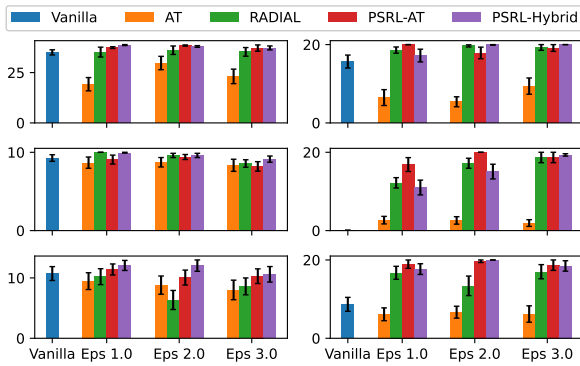


Figure 3: Averaged reward (left) and certified safety (right) for Highway (top), Twoway (middle) and Exit (bottom) for l_2 norm.

tent with what we presented in the main paper for l_∞ -norm perturbations: AT fails rather poorly, while the remaining

three methods typically yield near-optimal reward in most cases, with PSRL-Hybrid typically the best or nearly best of the methods.

Figure 3 (right) presents adversarial certified safety for $T_v = 5$. Here again we see a similar pattern: AT performs quite poorly, while both PSRL-AT and PSRL-Hybrid generally yield the best certificates, with RADIAL usually, but not always, performing comparably to at least one of these.

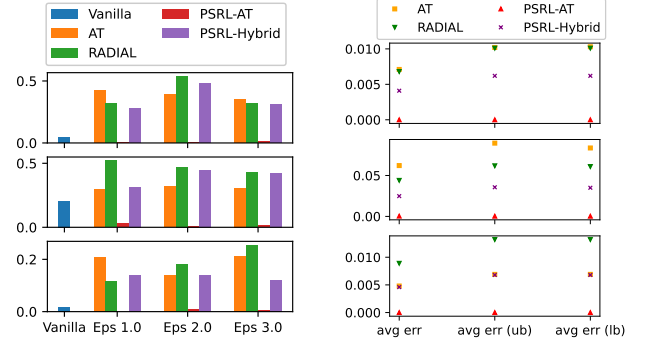


Figure 4: False action rate (left) and MSE of state prediction error (right) for Twoway (middle) and Exit (bottom) for l_2 norm.

Figure 4 presents the results for false action rate (left) and MSE (right), and the results again echo what we had observed for l_∞ norm, with PSRL-AT typically yielding far better performance in both compared to all others.

C Certification Results for 10 and 15 Steps

The certification results for 10 and 15 steps are shown in Figure 6 (l_∞ norm) and Figure 6 (l_2 norm). Here there is again broad consistency with our earlier discussion, but with several interesting exceptions. In particular, we see that—particularly for the l_∞ norm—RADIAL exhibits high variation in performance, and occasionally rather poor robustness certificates compared to other methods (e.g., for $T_v = 15$).

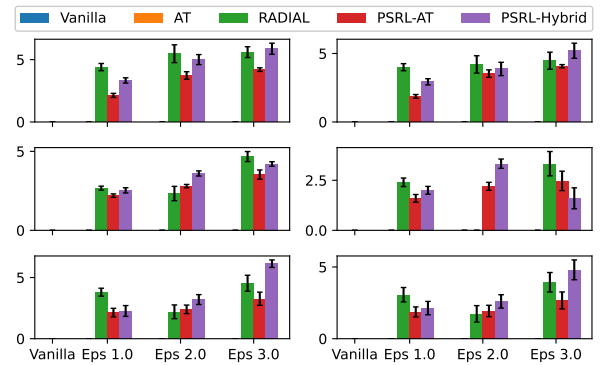


Figure 5: Results for l_∞ norm. 10 steps (left) and 15 steps (right) for Highway (top), Twoway (middle) and Exit (bottom).

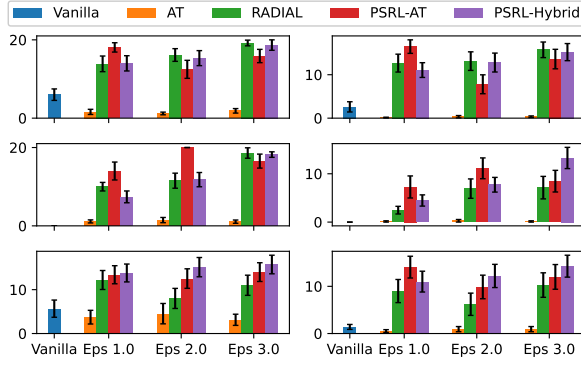


Figure 6: Results for l_2 norm. 10 steps (left) and 15 steps (right) for Highway (top), Twoway (middle) and Exit (bottom).

D Efficacy of TASC

Here we present the total median number of nodes explored. Note that the number of nodes possible is 781 for 5 steps, 2,441,406 for 10 steps and 7,629,394,531 for 15 steps. We cap the total number of nodes allowed to be explored to $M = 500$, however, as we presented below, it does not impact our median results for l_∞ norm, and only impact Vanilla and AT result for *Highway* environment under l_2 norm. For l_∞ norm, it is presented in Table 2 and for l_2 norm it is presented in Table 3.

Highway			
	5 step	10 step	15 step
Vanilla	33	12	19
AT	12	10	19
RADIAL	30	116	379
PSRL-AT	20	52	174
PSRL-Hybrid	29	154	307

Twoway			
	5 step	10 step	15 step
Vanilla	5	11	15
AT	5	11	15
RADIAL	9	25	27
PSRL-AT	8	47	51
PSRL-Hybrid	13	62	73

Exit			
	5 step	10 step	15 step
Vanilla	5	10	15
AT	8	17	17
RADIAL	6	21	19
PSRL-AT	9	13	15
PSRL-Hybrid	7	22	15

Table 2: TASC: total number of nodes explored (l_∞ norm).

E Environment Setting

We take two consecutive frames as the image input. The ego vehicle is located at the right side of the image (10%). See

Highway			
	5 step	10 step	15 step
Vanilla	41	500	500
AT	16	150	500
RADIAL	5	10	20
PSRL-AT	5	10	17
PSRL-Hybrid	5	10	20

Twoway			
	5 step	10 step	15 step
Vanilla	20	63	20
AT	10	44	22
RADIAL	6	13	18
PSRL-AT	5	13	15
PSRL-Hybrid	8	15	19

Exit			
	5 step	10 step	15 step
Vanilla	8	13	15
AT	7	14	15
RADIAL	5	10	10
PSRL-AT	5	10	11
PSRL-Hybrid	5	10	12

Table 3: TASC: total number of nodes explored (l_2 norm).

Figure 7 for demonstration. We take the grayscale image between the rightmost and left most lanes. Note that the input image for *Exit* environment is similar to *Highway*, in the figure below we demonstrate the exit lane. The input image for *Highway* and *Exit* is of size (100, 25), and for *Twoway* it is (150, 17).

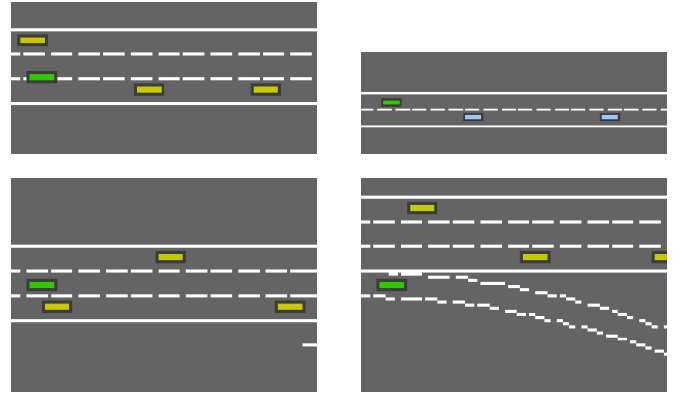


Figure 7: Images illustrating the experiment environment. Highway (top left), Twoway (top right), and Exit (bottom). Ego vehicle is colored as green, and other vehicles are colored as yellow or blue.

F Hyperparameters and Experiment Details

Hyperparameters Our reinforcement learning model implementation is based on Leurent (2018b). To ensure fair comparison, we use the same set of hyperparameters across all experiments and the same neural network architecture

within the same environment (please refer to our attached code for the neural network architecture details). For learning $g(o)$, we use buffer size 10000 and batch size 32, and predict positions of two vehicles closest to the ego vehicle in the scene. In case there are less than two vehicles in the scene, the positions default to -1 for *Highway* and *Twoway*, and defaults to 0 for *Exit*. For reinforcement learning, most of the hyperparameters are consistent with the original setting in Leurent (2018b) under the *Highway* environment. For replay buffer we tested under the *Highway* environment and choose the best among {5000, 10000, 15000}. For our final model, we use buffer size 10000, batch size 32 and discount factor 0.8 across all environments and experiments (for both l_2 and l_∞ norms). In terms of certification, there is no hyperparameter needed for l_∞ norm, and for l_2 norm with median smoothing, we use $N = 2000$ samples with $\sigma = 0.05$, and take the cap of the certified robustness as 20/255. That is, once we certify robustness against 20/255, we do not explore further and the algorithm returns 20/255.

Evaluation Details For each experiment (Vanilla, AT, RADIAL, PSRL-AT and PSRL-Hybrid) and each environment (*Highway*, *Twoway* and *Exit*), we perform three independent runs. For evaluation, we use five seeds for nominal reward and certification. That means, for nominal and certification we report the mean and SEM (standard error of the mean) across 15 evaluation results for each method under each target training epsilon ($\{1.0, 2.0, 3.0\}/255$). For the MSE error over states prediction, we further take average over three target training epsilon, meaning that the result is averaged over 45 evaluation runs. The duration of the simulation during evaluation is 40 steps for *Highway*, 20 steps for *Twoway* and 15 steps for *Exit*.

Computational Cost The training time for $g(o)$ and reinforcement learning varies among different environments, but is in general within 1 hours. The running time for adversarial training is 4 hours for each run on a single GeForce RTX 3090 GPU. The certification time varies based on the total number of nodes explored. The maximum running time for a run is 13 minutes where 500 nodes are explored; the majority of the running time is within 5 minutes.