Neural Network Control Policy Verification with Persistent Adversarial Perturbations

Yuh-Shyang Wang¹  Tsui-Wei Weng²  Luca Daniel²

Abstract

Deep neural networks are known to be fragile to small adversarial perturbations, which raises serious concerns when a neural network policy is interconnected with a physical system in a closed loop. In this paper, we show how to combine recent works on static neural network certification tools with robust control theory to certify a neural network policy in a control loop. We give a sufficient condition and an algorithm to ensure that the closed loop state and control constraints are satisfied when the persistent adversarial perturbation is $\ell_\infty$ norm bounded. Our method is based on finding a positively invariant set of the closed loop dynamical system, and thus we do not require the continuity of the neural network policy. Along with the verification result, we also develop an effective attack strategy for neural network control systems that outperforms exhaustive Monte-Carlo search significantly. We show that our certification algorithm works well on learned models and could achieve 5 times better result than the traditional Lipschitz-based method to certify the robustness of a neural network policy on the cart-pole balance control problem.

1. Introduction

Deep neural networks have become state-of-the-arts in a variety of machine learning tasks, including control of a physical system in the reinforcement learning setting. For example, in guided policy search (Zhang et al., 2016), a neural network policy is used to replace the online model predictive control policy to directly control a physical system in a feedback loop. Yet, recent studies demonstrate that neural networks are surprisingly vulnerable to adversarial examples and attacks (Huang et al., 2017; Szegedy et al., 2013; Xie et al., 2017; Jia & Liang, 2017). For applications that are safety-critical, such as self-driving cars, the existence of adversarial examples in neural networks has raised severe and unprecedented concerns – a small perturbation on the neural network input may cause significant change in control actions, which could possibly destabilize the system or drive the system state to an unsafe region. Therefore, it is crucial to develop a verification tool that can provide useful certificate (such as safety or robustness guarantees) for a dynamical system with a neural network policy in the loop.

Recently, much research effort has been devoted to developing verification methods to quantify the robustness of neural networks against adversarial input perturbations (Katz et al., 2017; Ehlers, 2017; Kolter & Wong, 2018; Ruan et al., 2018; Weng et al., 2018; Dvijotham et al., 2018; Bunel et al., 2018; Boopathy et al., 2019; Royo et al., 2019). Specifically, given a neural network policy $u = \pi(y)$, these verifiers can certify that $u = \pi(y) \in \mathcal{U}$ for all $y \in \mathcal{Y}$ for some sets $\mathcal{U}$ and $\mathcal{Y}$. However, these verifiers are developed in a static setting, where the neural networks do not interact with a dynamical system. When a neural network policy is deployed to a dynamical system (Zhang et al., 2016; Kiumarsi et al., 2017), the neural network output $u$ can affect the neural network input $y$ in the future via a feedback loop. To certify a neural network policy in closed loop, it is necessary to extend the static neural network certification tools to the dynamic setting.

In this paper, we propose a novel framework to verify neural network policies in a closed loop system by combining the aforementioned static neural network certification tools with robust control theory. We consider a realistic and strong threat model where the adversaries can manipulate observations and states at every time step over an infinite horizon (the so-called persistent perturbations). Our key idea is to leverage the static neural network certification tools to give a tight input-output characterization of the neural network policy, consequently allowing us to find a positively invariant set of the closed loop dynamical system when combined with robust control theory. Our contributions are summarized as below:
• To our best knowledge, our work is the first one to extend the aforementioned neural network certification tools to a dynamic setting, in which we certify a neural network policy in a feedback control loop under persistent adversarial perturbation.

• Our framework can handle non-Lipschitz and discontinuous neural network policies, while the traditional Lipschitz-based robust control approach cannot. Even when the neural network policy is Lipschitz continuous, we prove theoretically and validate experimentally that our proposed framework is always better than the traditional Lipschitz-based robust control approach in terms of tighter certification bound.

• We show that our method could work on situations where the system dynamic is unknown, unstable, and nonlinear. As long as one can learn a nominal linear model and over-approximate the modeling error using a data driven approach, our certification algorithm can be applied.

• Along with the certification algorithm, we develop an effective persistent $\ell_\infty$ attack algorithm that can successfully discover the vulnerability of a neural network control system while an exhaustive Monte-Carlo simulation cannot. This observation indicates that the robustness of a neural network control system cannot be certified via exhaustive simulation, and the mathematical-based certification framework developed in this paper is necessary.

Related works. Recent works (Berkenkamp et al., 2017; Richards et al., 2018; Jin & Lavaei, 2018a;b; Ivanov et al., 2019) in the field of safe reinforcement learning can give a certificate of stability or safety properties of a neural network control system. Our work differs from these works as we assume continuous instead of discrete state space model. We also explicitly consider model uncertainty and persistent adversarial perturbation, and provide safety guarantees against worst-case perturbations to the systems. While in (Carr et al., 2019), the safety constraint is certified on nominal systems. Finally, (Carr et al., 2019) considers both policy synthesis and verification. Our work only focuses on verification of a given policy.

Notations. We use lower case letters such as $x$ to denote vectors and upper case letters such as $A$ to denote matrices. We use $x \preceq y$ to denote that $x$ is element-wise less than or equal to $y$. For a square matrix $A$, we use $\rho(A)$ to denote the largest absolute value of the eigenvalues of $A$. We use boldface letters such as $x$ and $\Phi$ to denote signals and transfer matrices in the frequency domain, respectively. For a time signal $\{x[t]\}_{t=0}^{\infty}$, the unilateral $z$-transform of the time series $\{x[t]\}_{t=0}^{\infty}$ is given by $x(z) = \sum_{t=0}^{\infty} z^{-t} x[t]$. We use $\Phi[t]$ to denote the $t$-th spectral component of a transfer matrix $\Phi$, i.e., $\Phi(z) = \sum_{t=0}^{\infty} z^{-t} \Phi[t]$. We use $\mathbf{x}$ and $\Phi$ as shorthand of $x(z)$ and $\Phi(z)$ when the context makes it clear. For a frequency domain equation $y = \Phi x$, the corresponding time domain equation is given by the convolution formula $y[t] = \sum_{\tau=0}^{\infty} \Phi[\tau] x[t - \tau]$. We use $\|x\|_{\ell_p}$ to denote the standard $\ell_p$ norm of the signal $x$ and use $\|\Phi\|_{\mathcal{L}_1}$ and $\|\Phi\|_{\mathcal{H}_\infty}$ to denote the $\mathcal{L}_1$ norm (Dahleh & Pearson, 1987) and the $\mathcal{H}_\infty$ norm (Zhou et al., 1996) of the system $\Phi$. We define the absolute operator $\text{abs}(\cdot)$ of a stable transfer matrix $\Phi$ by $\text{abs}(\Phi) = \sum_{t=0}^{\infty} |\Phi[t]|$. Note that each element of the transfer matrix $\Phi$ is absolutely summable if and only if the transfer matrix $\Phi$ is real rational and stable (Page 113-114 of (Oppenheim et al., 1996)).

2. Problem Formulation

We consider a neural network policy interconnected with a dynamical system. The model architecture is shown in Fig. 1, where $\pi(\cdot)$ is the neural network control policy, $P$ is the plant to be controlled, and $\Delta$ characterizes the uncertainty...
We assume that $P$ is a discrete time linear time invariant (LTI) system with dynamics given by

$$
\begin{align*}
  x[t + 1] &= Ax[t] + Bu[t] + B_w w[t] + B_\delta \delta[t] & \quad (1) \\
  y[t] &= Cx[t] + Dw_w w[t] & \quad (2) \\
  \alpha[t] &= C_\alpha x[t] + D_\alpha u[t] + D_\alpha w w[t]. & \quad (3)
\end{align*}
$$

where $t$ denotes the time index, $x$ the state vector, $u$ the control action, $y$ the measurement, $w$ the external perturbation, and $\alpha$ and $\delta$ the input and output of the uncertainty block $\Delta$.

In particular, $w$ is a persistent perturbation over an infinite horizon $t \geq 0$. In the rest of the paper, we assume zero initial condition unless stated otherwise:

$$
x[0] = 0.
$$

This is for ease of exposition – the method proposed in this paper can be readily extended to handle persistent perturbation with nonzero initial condition as well. We assume the pair $(A, B)$ is stabilizable and $(A, C)$ is detectable. The neural network policy is assumed to be static:

$$
u[t] = \pi(y[t]),
$$

while the uncertainty block can be dynamic, with

$$
\delta = \Delta(\alpha)
$$

in the frequency domain. We assume that $\Delta$ is stable and norm bounded. Equations (1) - (6) give a complete description of the model architecture shown in Fig. 1.

In this paper, we use $w$ to model the persistent adversarial perturbation. We assume that $w$ lies in the set $\|w\|_\infty \leq w_\infty$, or equivalently,

$$
w \in \{w \mid \|w[t]\|_\infty \leq w_\infty 1 = \bar{w}, \forall t \geq 0\}. & \quad (7)
$$

The matrices $B_\alpha$ and $D_\alpha$ in (1) - (2) control the impact of $w$ on the state $x$ and the neural network input $y$. The goal of this paper is the following:

**Goal.** Given the model equations (1) - (6), design an algorithm to certify that the following requirement is satisfied for any adversarial sequence $\{w[t]\}_{t=0}^\infty$ satisfying (7):

$$
|x[t]| \leq x_{lim}, \quad |y[t]| \leq y_{lim}, \quad |u[t]| \leq u_{lim}, \quad \forall t \geq 0.
$$

3. Closed Loop Boundedness

In this section, we derive a sufficient condition to ensure that the state $x$, measurement $y$, and control $u$ in Fig. 1 are bounded for any adversarial sequence $w$. We first assume that the plant model (1) - (3) is open loop stable, i.e., $\rho(A) < 1$, and directly apply traditional robust control theory to obtain Lemma 1 for closed loop stability. We then explain the restrictions of Lemma 1 and improve the results by combining robust control theory with neural network certification tools to obtain Theorem 1. Finally, we extend our results to unstable plants.

3.1. Stable Plant: robust control baseline

We use the frequency domain notation introduced before to rewrite (1) - (3) as

$$
\begin{bmatrix}
x \\
y \\
\alpha
\end{bmatrix}
= 
\begin{bmatrix}
\Phi_{xu} & \Phi_{xw} & \Phi_{x\delta} \\
\Phi_{yu} & \Phi_{yw} & \Phi_{y\delta} \\
\Phi_{\alpha u} & \Phi_{\alpha w} & \Phi_{\alpha \delta}
\end{bmatrix}
\begin{bmatrix}
u \\
w \\
\delta
\end{bmatrix}
= \Phi
$$

with $\Phi_{xu} = (zI - A)^{-1} B$, $\Phi_{xw} = (zI - A)^{-1} B_w$, $\Phi_{x\delta} = (zI - A)^{-1} B_\delta$, $\Phi_{yu} = C\Phi_{xu}$, $\Phi_{yw} = C \Phi_{xw}$, $\Phi_{y\delta} = C \Phi_{x\delta}$, $\Phi_{\alpha u} = C_\alpha \Phi_{xu}$, $\Phi_{\alpha w} = C_\alpha \Phi_{xw}$, and $\Phi_{\alpha \delta} = C_\alpha \Phi_{x\delta}$. To show the input-output stability of the structure in Fig. 1, we first assume that the neural network policy is locally Lipschitz continuous with a finite $\ell_\infty$ gain $\gamma_\pi$, i.e., $\|u\|_\infty \leq \gamma_\pi \|y\|_\infty$ over some range $\|y\|_\infty \leq y_\infty$. For a stable transfer matrix, the $\ell_\infty$ induced norm of the system norm (Dahleh & Pearson, 1987). The following Lemma gives the local input-output stability of Fig. 1, which directly comes from robust control theory:

**Lemma 1.** Consider the model in Fig. 1 ((1) - (7)) with $\rho(A) < 1$. Suppose that the neural network policy $u = \pi(y)$ has a finite $\ell_\infty$ to $\ell_\infty$ gain $\gamma_\pi$ for all $\|y\|_\infty \leq y_\infty$, and the uncertainty block $\Delta$ has the property $\|\delta\|_\infty \leq \gamma_\Delta \|\alpha\|_\infty$. If the following three conditions hold:

$$
\begin{align*}
  \beta_1 &= \gamma_\Delta \|\Phi_{\alpha \delta}\|_1 < 1 & \quad (10a) \\
  \beta_2 &= \gamma_\pi \left[\|\Phi_{yu}\|_1 + \gamma_\Delta \|\Phi_{y\delta}\|_1 \|\Phi_{\alpha u}\|_1 \|\Phi_{\alpha \delta}\|_1 \right] < 1 & \quad (10b) \\
  \frac{1}{1 - \beta_2} \left[\|\Phi_{y\delta}\|_1 + \gamma_\Delta \|\Phi_{y\delta}\|_1 \|\Phi_{\alpha w}\|_1 \|\Phi_{\alpha \delta}\|_1 \right] w_\infty \leq y_\infty & \quad (10c)
\end{align*}
$$

then the closed loop system in Fig. 1 is input-output stable over the region $\|y\|_\infty \leq y_\infty$ for all adversarial attack $w$ satisfying (7).
The proof of Lemma 1 is in the Supplementary. To certify the local input-output stability of the closed loop system using Lemma 1, we can iteratively search for \( y_{\infty} \) until the local Lipschitz continuous assumption and (10a) - (10c) are both satisfied, or the state constraint is violated. Note that when \( \gamma_r \) is the global \( \ell_\infty \) to \( \ell_\infty \) bound of the neural network policy, we can drop the condition (10c) because \( y_{\infty} \) can be arbitrarily large. The global version of Lemma 1, i.e., with \( y_{\infty} \) being arbitrarily large, can be found in the robust control literature in (Khammash & Pearson, 1991).

When the magnitude of the signals \( w \) and \( y \) is characterized by the \( \ell_2 \) norm, we can replace the \( L_1 \) norm in Lemma 1 by the \( H_\infty \) norm, as the \( H_\infty \) norm is equivalent to the \( \ell_2 \) to \( \ell_2 \) induced norm. We include the global and \( H_\infty \) version of Lemma 1 in the Supplementary, which is an unstructured version of the main loop theorem \(^3\) (Packard & Doyle, 1993; Zhou et al., 1996) in the robust control literature (\( H_\infty \) control and structured singular value).

### 3.2. Stable Plant: improvement

Lemma 1 has several restrictions. First, the neural network policy needs to be Lipschitz continuous, and thus Lemma 1 cannot be applied on non-differentiable or discontinuous policies due to quantization or other issues. Second, even when the given neural network policy is Lipschitz continuous, the bound of the local Lipschitz constant \( \gamma_r \) for a deep neural network policy is usually very loose. As a consequence, Lemma 1 can only be applied to certify the robustness of Fig. 1 over a small region near the stable equilibrium. Finally and most importantly, even if the given policy is Lipschitz continuous and the local Lipschitz constant \( \gamma_r \) is exact, using \( \|u\|_{\ell_{\infty}} \leq \gamma_r \|y\|_{\ell_{\infty}} \) to characterize the input-output relation of a given neural network is usually very loose. In this subsection, we improve the results of Lemma 1 and propose a more useful Theorem to certify the boundedness of the structure in Fig. 1.

Our strategy is to use the static neural network certification tool to give a tighter characterization of the input-output relation of the given neural network policy. The following Theorem gives a sufficient condition to ensure closed loop boundedness under any adversarial attack satisfying (7).

**Theorem 1.** Consider the model in Fig. 1 ((1) - (7) with \( \rho(A) < 1 \)). If we can find a quadruplet \((\bar{y}, \bar{u}, \bar{\alpha}, \bar{\delta})\) satisfying the following conditions:

1. Neural network robustness certificate: The static policy \( u = \pi(y) \) has the property \( |u| \leq \bar{u} \) for all \( |y| \leq \bar{y} \).

2. Input-output relation of the uncertainty block: \( \delta = \Delta(\alpha) \) has the property \( |\alpha[k]| \leq \bar{\alpha}, k = 0, \cdots, t \implies |\delta[t]| \leq \bar{\delta} \) for all \( t \geq 0 \).

3. Feedback condition: \( \text{abs}(\Phi_{yw}) \bar{w} + \text{abs}(\Phi_{yu}) \bar{u} + \text{abs}(\Phi_{w\delta}) \bar{\delta} \leq \bar{y} \) and \( \text{abs}(\Phi_{cw}) \bar{w} + \text{abs}(\Phi_{cu}) \bar{u} + \text{abs}(\Phi_{c\delta}) \bar{\delta} \leq \bar{\alpha} \).

then we have the following properties:

1. Bounded feedback signals: \( |y[t]| \leq \bar{y}, |u[t]| \leq \bar{u}, |\alpha[t]| \leq \bar{\alpha}, |\delta[t]| \leq \bar{\delta} \) for all \( t \geq 0 \).

2. Bounded state: \( |x[t]| \leq \bar{x}, \text{with} \bar{x} = \text{abs}(\Phi_{wx}) \bar{w} + \text{abs}(\Phi_{xu}) \bar{u} + \text{abs}(\Phi_{x\delta}) \bar{\delta} \) for all \( t \geq 0 \).

The key idea of Theorem 1 is to combine a static neural network certification algorithm (Kolter & Wong, 2018; Weng et al., 2018; Gehr et al., 2018; Dvijotham et al., 2018; Boopathy et al., 2019; Royo et al., 2019) with robust control method (Berkenkamp et al., 2017; Richards et al., 2018; Jin & Lavaei, 2018a;b) to robust control theory to certify a neural network policy in a feedback control loop. The first condition of Theorem 1 is an input-output characterization of the neural network policy, which can be obtained by a static neural network certification algorithm. The second condition is a characterization of the model uncertainty block \( \Delta \). The third condition, when combining with the first two conditions, ensures that \( \{(y, u, \alpha, \delta)\mid |y| \leq \bar{y}, |u| \leq \bar{u}, |\alpha| \leq \bar{\alpha}, |\delta| \leq \bar{\delta}\} \) is a positively invariant set of the closed loop dynamical system. The complete proof of Theorem 1 is in the Supplementary. Theorem 1 can be used as follows: if we have \( (\bar{x}, \bar{y}, \bar{u}) \leq (x_{\text{lim}}, y_{\text{lim}}, u_{\text{lim}}) \), then the requirement (8) is satisfied. We will discuss how to find a quadruplet \((\bar{y}, \bar{u}, \bar{\alpha}, \bar{\delta})\) satisfying the conditions of Theorem 1 in the next section.

Theorem 1 has several advantages over Lemma 1. First, Theorem 1 does not require the differentiability or continuity of the neural network policy \( \pi(\cdot) \). Theorem 1 is valid as long as the property \( |u| \leq \bar{u} \) for all \( |y| \leq \bar{y} \) can be certified. This is one of the key differences between our approach and the Lipschitz-based robust control method (Berkenkamp et al., 2017; Richards et al., 2018; Jin & Lavaei, 2018a;b). Second, the neural network certification tool can give a tighter input-output characterization of the neural network policy than the local Lipschitz constant. As a result, the conditions of Theorem 1 are less restricted and easier to satisfy. The following Theorem (proof in the Supplementary) claims that the conditions of Lemma 1 implies that the sufficient conditions of Theorem 1 will always hold. This means that Theorem 1 can be applied on a strictly larger class of problems than that of Lemma 1.

**Theorem 2.** The conditions of Lemma 1 imply the conditions of Theorem 1.

Note that Theorem 1 only certifies the boundedness of the closed loop system, not the stability of the closed loop system – Theorem 1 does not guarantee that \( x = 0 \) is a stable...
Neural Network Control Policy Verification with Persistent Adversarial Perturbations

equilibrium. However, in the presence of persistent adversarial perturbation, we argue that there is no significant difference between boundedness and stability because both of them will have a finite yet nonzero state deviation. In our case study section, we show that Theorem 1 is much more useful than the traditional robust control approach (Lemma 1) as it can certify the requirement (8) with persistent adversarial attack (7) that is 5 times larger.

3.3. Unstable Plant

In this subsection, we consider the case where the plant (1) - (3) is unstable. Our strategy is to extract a first order approximation of the neural network policy to stabilize the plant first, then analyze the interconnection of the stabilized plant and the residual control policy. Specifically, we rewrite the neural network policy as $u[t] = \pi(y[t]) = K_0 y[t] + \pi_0(y[t])$ for some matrix $K_0$. We call $\pi_0(\cdot)$ the residual control policy. Equations (1) - (3) then become

$$x[t + 1] = (A + BK_0C)x[t] + Bu_0[t] + (BK_0 D_o + B_w)w[t] + B_0 \delta [t]$$  \hspace{1cm} (11a)

$$y[t] = Cx[t] + D_ow[t]$$ \hspace{1cm} (11b)

$$\alpha[t] = (C_n + D_{ou} K_0 C)x[t] + D_{ou} u_0[t] + (D_{ou} K_0 D_o + D_{ou} w[t]$$ \hspace{1cm} (11c)

with the neural network policy $u_0[t] = \pi_0(y[t])$. As long as the spectral radius of the closed loop system matrix $A_{cl} = (A + BK_0C)$ is less than 1, the transfer matrix $(z I - A_{cl})^{-1}$ is stable. Theorem 1 can then be used with redefined transfer matrices: for instance, we have $\Phi_{w} = (z I - A_{cl})^{-1}(BK_0 D_o + B_w) - \text{other transfer matrices could be derived in a similar manner.}$ In the following, we propose two different ways to obtain a candidate $K_0$: (1) using the Jacobian evaluated at the origin, and (2) using neural network certification tool to find a first order approximation of the policy over a region.

If the neural network policy has the property $\pi(0) = 0$ and is differentiable at $y = 0$, we can use the Jacobian of $\pi(y)$ evaluated at $y = 0$ as a candidate for $K_0$, i.e., $K_0 = \frac{\partial \pi(y)}{\partial y} |_{y=0}$. In this case, the residual control policy $\pi_0(y)$ has the following property:

$$\pi_0(0) = 0 \text{ and } \lim_{||y|| \to 0} \frac{||\pi_0(y)||}{||y||} = 0 \hspace{1cm} (12)$$

We then have the following Lemma adopt from Theorem 4.3 in (Aström & Murray, 2010) (uncertainty $\Delta$ is ignored).

**Lemma 2.** Consider the system (11a) - (11b) with the residual control policy $\pi_0(y) = \pi(y) - K_0 y$ satisfying (12). If $\rho(A + BK_0C) < 1$, then $x = 0$ is a locally asymptotically stable equilibrium point of the system (11a) - (11b). If $\rho(A + BK_0C) > 1$, then $x = 0$ is a locally unstable equilibrium point of the system (11a) - (11b).

If the neural network policy is not differentiable or the equilibrium $x = 0$ is locally unstable according to Lemma 2, one can leverage the neural network certification tools to find a different candidate $K_0$. For instance, the method proposed in (Weng et al., 2018) provide a pair of linear lower and upper bounds on the neural network output as

$$K_L y + b_L \leq \pi(y) \leq K_U y + b_U, \hspace{1cm} \forall \|y\| \leq y_{ref} \hspace{1cm} (13)$$

A candidate $K_0$ is given by $(K_U + K_L)/2$, which is a first order approximation of the control policy over the region $\|y\| \leq y_{ref}$. As long as we can find a $K_0$ to make $\rho(A_{cl}) < 1$, Theorem 1 is a valid sufficient condition to verify the requirement (8) under persistent perturbation (7). This statement holds even when a stable equilibrium does not exist.

4. Algorithms

In this section, we propose an iterative algorithm (Algorithm 1) to find a quadruplet $(\bar{y}, \bar{u}_0, \bar{\alpha}, \bar{\delta})$ that satisfies the conditions of Theorem 1. We then propose a simple and effective attack strategy for a neural network control system.

4.1. Algorithm for Theorem 1

For an unstable plant, we assume that a matrix $K_d$ with $\rho(A + BK_d C) < 1$ is given as a default linear approximation of the neural network policy$^4$. We assume that the uncertainty block is described by $|\delta| \leq \Gamma_{\Delta} |\alpha|$ for a non-negative matrix $\Gamma_{\Delta}$. We explain why this form of uncertainty is natural when the model and uncertainty are learned using a data driven approach in (Dean et al., 2017) in the Supplementary. The following algorithm finds a quadruplet $(\bar{y}, \bar{u}_0, \bar{\alpha}, \bar{\delta})$ satisfying the conditions of Theorem 1.

Algorithm 1 can be interpreted as follows. We first extract a linear policy $K$ to make the closed loop transfer matrices stable and calculate the bounds for $\bar{u}$ and $\bar{u}_0$ over the region $|y| \leq y_{ref}$ using neural network certification tools (Kolter & Wong, 2018; Weng et al., 2018; Gehr et al., 2018). Meanwhile, we calculate the bound for $\delta$ based on the assumption of the uncertainty block $\Delta$. These two steps ensure that the quadruplet $(y_{ref}, \bar{u}_0, \alpha_{ref}, \delta)$ satisfies the first two conditions of Theorem 1. Given the range of the adversarial perturbation $\bar{w}$, the residual control action $\bar{u}_0$, and the uncertainty-induced input $\delta$, we calculate the bounds for the state $x$, measurement $y$, and $\alpha$ on Line 7. If we have $\bar{y} \leq y_{ref}$ and $\bar{\alpha} \leq \alpha_{ref}$, then $(y_{ref}, \bar{u}_0, \alpha_{ref}, \delta)$ also satisfies the third condition of Theorem 1, thus we obtain a certificate for closed loop boundedness. If $\bar{y} \nleq y_{ref}$ or

$^4$Finding a static gain $K$ to make $A + BK_d C$ a stable matrix is known as the static output feedback problem. This problem can be NP-hard (Blondel & Tsitsiklis, 1997) in worst case. For an output feedback problem, one may consider using a dynamic (recurrent) policy for control.
Algorithm 1 Certification of state and control constraints under persistent adversarial perturbation

1: **Input:** Equations (1) - (8), initial bounds $y_{ref} = 0$ and $\alpha_{ref} = 0$, default approximation $K_d$, parameter $\epsilon = 10^{-6}$, flag Success = False, Done = False, $k = 0$, MaxIter = 200
2: **Output:** flag Success, certified bounds $\bar{x}$, $\bar{y}$, $\bar{u}$
3: **while** Done == False **and** $k < $ MaxIter **do**
4: \[ u_0, \bar{u}, K = \text{NN-CERTIFICATION}(\pi, y_{ref}, K_d) \text{ from existing tools such as (13)}.\]
5: \[ \bar{\delta} = \Gamma_\Delta \alpha_{ref} \]
6: **Calculate the transfer matrix** $\Phi$ in (9).
7: \[ \{ \bar{x} \bar{y} \bar{\alpha} \}^\top = \text{abs}(\Phi) [\bar{w} \bar{u}_0 \bar{\delta}]^\top \]
8: **if** $\bar{x} \not\leq x_{lim}$ or $\bar{y} \not\leq y_{lim}$ or $\bar{u} \not\leq u_{lim}$ **then**
9: Success = False, Done = True
10: **else if** $\bar{y} \geq y_{ref}$ and $\bar{\alpha} \leq \alpha_{ref}$ **then**
11: Success = True, Done = True
12: **else**
13: \[ y_{ref} = (1 + \epsilon)\bar{y}, \alpha_{ref} = (1 + \epsilon)\bar{\alpha}, k = k + 1 \]
14: **end if**
15: **end while**
16: **Return** Success, $\bar{x}$, $\bar{y}$, $\bar{u}$

$\bar{x} \not\leq \alpha_{ref}$, we then increase the test bound $(y_{ref}, \alpha_{ref})$ and repeat the search.

The overall computational complexity of Algorithm 1 depends mainly on the chosen neural network certification tool, as Line 4 is usually more expensive than other lines within the while loop. Our case study uses the method in (Zhang et al., 2018) as the certification tool, whose complexity is $O(n^2 n^3)$ for a $m$ layer neural network with $n$ nodes per layer.

4.2. Attack Algorithm

In addition to the certification algorithm, here we propose an algorithm to attack the neural network policy in the control loop. As an example, we show how to design a perturbation sequence $\{w[t]\}_{t=0}^T$ with $\|w\|_\infty = 1$ to attack the $i$-th state $x_i$. Our idea is to follow Lines 4 - 6 of Algorithm 1 to construct the closed loop transfer matrices with the help of neural network certification tools. We then have $x[T] = \sum_{T=0}^\infty \Phi_{xw}[T-\tau]w[\tau] + \Phi_{xu}[T-\tau]u_0[\tau] + \Phi_{xa}[T-\tau]\delta[\tau]$. If we ignore the contribution of $u_0$ and $\delta$, we can maximize $x_i[T]$ using the following $\ell_\infty$ bounded perturbation sequence:

\[ w_j[t] = \text{sign}((\Phi_{xw}[T-t])_{ij}), \quad t = 0, \cdots, T \quad (14) \]

for each $j$. Note that (14) is sub-optimal because we ignore the contribution of $u_0$ and $\delta$. Nevertheless, we show in the next section that our attack (14) is extremely strong.

5. Case Study

In this section, we demonstrate our algorithm on a cart-pole example and show the following results:

- Our proposed framework (Theorem 1 and Algorithm 1) outperforms methods based on traditional robust control theory (Lemma 1). Specifically, Algorithm 1 can certify the boundedness of the closed loop system with attack level that is 5 times larger than that of a robust control approach. See the result in Fig. 2a.
- Algorithm 1 works on situations where the system dynamics is unknown, unstable, and nonlinear. We use the method in (Dean et al., 2017) to learn a nominal model $P$ with conservative over-approximation of the modeling error $\Delta$, then use Algorithm 1 to certify the robustness of the neural network policy interconnected with the learned model. The result is close to that with the knowledge of the true model. See the result in Fig. 2b.
- Algorithm 1 can be applied on a discontinuous policy where the control action is quantized into discrete levels. Lipschitz-based methods (Berkenkamp et al., 2017; Richards et al., 2018; Jin & Lavaei, 2018b,a) cannot certify the boundedness of the closed loop system in this case. See the result in Fig. 2c.
- Our proposed attack algorithm (14) is far more effective than an exhaustive Monte-Carlo attack. In particular, our model-based attack algorithm can successfully discover the internal vulnerability of the closed loop system while an exhaustive Monte-Carlo simulation cannot. See the result in Fig. 2d, 2e, and 2f.

**Experiment setup.** We use proximal policy optimization (Schulman et al., 2017) in stable baselines (Hill et al., 2018) to train a 3-layer neural network policy for cart-pole problem in OpenAI gym (Brockman et al., 2016). Our neural network has 16 neurons per hidden layer, ReLU activations, and continuous control output. To obtain a more stable policy, we modify the reward function to be

\[ r[t] = 2 - \frac{x[t]^\top Q x[t] + u[t]^\top R u[t]}{x_{lim}^\top Q x_{lim} + u_{lim}^\top R u_{lim}} \]

which will attempt to minimize the quadratic cost $x[t]^\top Q x[t] + u[t]^\top R u[t]$. The policy is trained with 2M steps. We use the neural network certification framework developed in (Weng et al., 2018; Zhang et al., 2018) for Algorithm 1. For the ease of illustrating the result, we consider an one dimensional persistent perturbation on the pole angle measurement of the cart-pole. The requirement is to certify that a single state (angle of the pole) is within the user-specified limit. Note that Algorithm 1 can be applied.
Algorithm 1 works on learned models

State deviation as output

Algorithm 1 works on non-Lipschitz policies

The safe region certified by Theorem 1 is larger than the safe region certified by the traditional robust control approach (Lemma 1, area below the red curve).

Tightness of Algorithm 1.

The unsafe region discovered by our attack algorithm (14) (area above the black curve) is much larger than the unsafe region discovered by exhaustive Monte-Carlo simulation (area above the brown curve).

Our designed attack causes a huge state deviation between 100s and 150s. Monte-Carlo based random attack is injected before 100s and after 150s.

Our designed attack is injected between 100s and 150s.

Figure 2: Summary of our result. (a): The safe region certified by our Algorithm 1 (area below the blue curve) is larger than the safe region certified by the traditional robust control approach (Lemma 1, area below the red curve). (b): Compare Algorithm 1 on various learned models (Dean et al., 2017) and the true model. (c): Compare certificates on Lipschitz and non-Lipschitz (quantized) neural network policies. (d): The unsafe region discovered by our attack algorithm (14) (area above the black curve) is much larger than the unsafe region discovered by exhaustive Monte-Carlo simulation (area above the brown curve). (e): Our designed attack causes a huge state deviation between 100s and 150s compared to the Monte-Carlo based attack.

The complete model equations are in the Supplementary.

Tightness of Algorithm 1. In this experiment, we use the linearized cart-pole model with no uncertainty ($\Gamma_\Delta = 0$) to compare the tightness of Algorithm 1 (Theorem 1) and the Lipschitz-based robust control baseline (Lemma 1). Given a user-specified state deviation limit $x_{\text{lim}}$, we use binary search to call Algorithm 1 repeatedly to obtain the largest possible $w^*_\infty$ such that the safety requirement $\|x\|_\infty \leq x_{\text{lim}}$ is satisfied for any persistent attack $\|w\|_\infty \leq w^*_\infty$. We plot $w^*_\infty$ as a function of $x_{\text{lim}}$ as the blue curve in Fig. 2a. Clearly, the area below the blue curve is the safe region certified by Theorem 1. Likewise, the area below the red curve is the safe region certified by the traditional robust control theory (Lemma 1), where the local Lipschitz constant is obtained via a sampling-based approach. Fig. 2a validates our claim in Theorem 2 – the safe region certified by Lemma 1 is always a subset of the safe region certified by Theorem 1. For $x_{\text{lim}} = 50$ (the vertical green line in Fig. 2a), Algorithm 1 can certify an attack level that is 5.3 times larger than the one using Lemma 1. Note that there is a flat dashed line at attack level $= 1$ for the robust control approach. This is because when the attack level is greater than 1, the conditions (10a) - (10c) of Lemma 1 no longer hold. In other words, the maximum attack level that can be certified by Lemma 1 is 1 regardless of the state deviation limit. Note that the local Lipschitz constant used in Lemma 1 is only a lower bound because it is obtained via a sampling-based approach. Therefore, the only reason that can explain the gap between the blue curve and the red curve is that the neural network certification algorithm (Weng et al., 2018) gives a much tighter input-output characterization of the neural network policy than the...
Lipschitz-based method (which uses $\|u\|_{\ell_{\infty}} \leq \gamma_{\pi} \|y\|_{\ell_{\infty}}$).

In Fig. 2a, the area above the black curve is the unsafe region where our attack algorithm (14) (with $T = 2500$) is able to make the state deviation exceed the user-specified limit. When the attack level is small ($< 1$), we can see from Fig. 2a that the black, blue, and red curves overlap. This means that our certification bound is tight in this region. When the attack level increases (or the state deviation limit increases), we start to see a gap between our certification algorithm and our attack algorithm. Future research will attempt to further bridge this gap.

Algorithm 1 on learned models. In many reinforcement learning applications, the mathematical model of the system dynamics (1) - (6) is not available in the first place. Therefore, we need to learn the model before performing the certification task. In this experiment, we show that Algorithm 1 works on learned models, even when the underlying system dynamics is unknown, unstable, and nonlinear.

We use the method proposed in (Dean et al., 2017) to learn the nominal model $P$ and a conservative estimate of the uncertainty $\Delta$. For each episode, we run the simulation using the black-box nonlinear cart-pole model with random control action for 30 time steps. We run $N$ episodes of simulation to collect the data, then we solve a regression problem to estimate the system matrices for the nominal model $P$. To get a conservative bound of the modeling error $\Delta$, we synthesize 100 bootstrap samples of $P$ and use the element-wise maximum deviation from the nominal model to over-approximate the model uncertainty $\Delta$. The details of our model learning procedure is in the Supplementary.

In Fig. 2b, we show the certification result of Algorithm 1 on models learned with different number of episodes $N$. As $N$ increases, the 100 bootstrap samples become more consistent and thus the size of the modeling error $\Delta$ shrinks. For $N = 100$, Fig. 2b shows that the safe region certified by Algorithm 1 is very close to that with the true linearized model with no uncertainty. This experiment shows that our method works well even when the model of the system dynamics is unknown in the first place. Given a nonlinear and unknown plant model interconnected with a neural network control policy, as long as we can find a nominal LTI plant $P$ and bound the modeling error by $\Delta$, we can use Algorithm 1 to certify the boundedness of the closed loop system under persistent adversarial perturbation.

Algorithm 1 on non-Lipschitz neural network policies. In Fig. 2c, we show that Algorithm 1 can be applied on a non-Lipschitz neural network policy, where the output of the neural network is quantized into discrete levels. This setting is common in many reinforcement learning tasks, in which the control action is chosen from a discrete set. When the neural network output is quantized into discrete levels, the neural network policy becomes discontinuous and the closed loop system may not have a stable equilibrium. Therefore, the Lipschitz-based and the stability based methods cannot be used to certify the boundedness of the closed loop system. On the other hand, our Algorithm 1 can still be applied in this case – the area below the light blue curve in Fig. 2c is the safe region certified by Algorithm 1. This is because we use the static neural network certification tools to characterize the input-output relation of a neural network policy, which works even when the policy is not Lipschitz continuous.

Our attack algorithm. In Fig. 2d, we show that our simple attack algorithm (14) is far more effective than an exhaustive Monte-Carlo simulation. Specifically, the area above the black curve is the unsafe region discovered by our attack algorithm (14) (with $T = 2500$), while the area above the brown curve is the unsafe region discovered by Monte-Carlo simulation with 1 million time steps. There is a huge gap between the unsafe region found by the two methods. This is a strong evidence that the safety of a neural network control system cannot be certified using exhaustive Monte-Carlo simulation. We need to use the certification framework developed in this paper to guarantee the safety of a neural network control system.

For example, at attack level 5 (the horizontal red line in Fig. 2d), the mean, standard deviation, and maximum of the state deviation in 1M steps Monte-Carlo simulation are 0, 1.1, and 5.2, respectively. One may conclude that the probability of seeing a state deviation greater than 5.2 is $10^{-6}$ and falsely claim that the area outside the brown region is safe. Unfortunately, this statement is not true when the perturbation sequence is adversarial – with the same attack level, the maximum state deviation found by our algorithm is 31.4, which is 29 standard deviation away. We show the perturbation sequence (input) and the state deviation (output) as functions of time in Fig. 2e and 2f. In Fig. 2e, we inject our designed attack between 100s and 150s (sampling time $= 0.02s$, thus the number of time step is $T = 2500$). We inject Monte-Carlo based random attack with the same attack level before 100s and after 150s. It is clear from Fig. 2f that the state deviation is much higher when we inject our attack.

6. Conclusions

Neural networks have shown superior performance in various control tasks in reinforcement learning, yet people have concerns using them in safety critical systems because it is hard to certify its robustness under adversarial attacks. In this paper, we extended the static neural network certification tools to the dynamic setting and developed an algorithm to certify the robustness of a neural network policy in a feedback loop under persistent adversarial attack. We showed both theoretically and empirically that our method
Neural Network Control Policy Verification with Persistent Adversarial Perturbations

outperforms the traditional Lipschitz-based robust control approach and works on situations where the model dynamics is unknown in the first place. We also developed an \( \ell_\infty \) attack algorithm and showed that it can discover the vulnerability of a neural network control system while an exhaustive Monte-Carlo simulation cannot – this suggested that a mathematical-based certification framework, like the one developed in this paper, is necessary to ensure the safety of a neural network control system.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments to improve the quality of this paper.

References


