Bounding Perception Neural Network Uncertainty for Safe Control of Autonomous Systems

Zhilu Wang*, Chao Huang*, Yixuan Wang*, Clara Hobbs†, Samarjit Chakraborty†, Qi Zhu*
*Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL
Emails: {zhilu.wang@u., chao.huang@, yixuanwang2024@u., qzhu@}northwestern.edu
†Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC
Emails: {cghobbs, samarjit}@cs.unc.edu

***Abstract*—Future autonomous systems will rely on advanced sensors and deep neural networks for perceiving the environment, and then utilize the perceived information for system planning, control, adaptation, and general decision making. However, due to the inherent uncertainties from the dynamic environment and the lack of methodologies for predicting neural network behavior, the perception modules in autonomous systems often could not provide deterministic guarantees and may sometimes lead the system into unsafe states (e.g., as evident by a number of high-profile accidents with experimental autonomous vehicles). This has significantly impeded the broader application of machine learning techniques, particularly those based on deep neural networks, in safety-critical systems. In this paper, we will discuss these challenges, define open research problems, and introduce our recent work in developing formal methods for quantitatively bounding the output uncertainty of perception neural networks with respect to input perturbations, and leveraging such bounds to formally ensure the safety of system control. Unlike most existing works that only focus on either the perception module or the control module, our approach provides a holistic end-to-end framework that bounds the perception uncertainty and addresses its impact on control.**

I. Introduction

Future autonomous systems, such as self-driving cars, unmanned aerial vehicles, and industrial robots, are poised to fundamentally change our everyday life and provide transformative societal and economic benefits. These systems will rely on advanced sensors and deep neural networks for sensing and perceiving the dynamic environment, analyze the perceived information to reason about the situation, and make decisions accordingly for system planning, control and other functions.

The design and runtime operation of these autonomous systems, however, face significant technical challenges. In particular, while many of them are safety-critical systems, ensuring the correctness of their safety-related properties is a very challenging task [1]. This is partly due to the increasing complexity of system functionality and architectural platforms, with more functional features, growing software size, and usage of GPUs and multicore CPUs, but very importantly, also because of the various *uncertainties* in the system. These uncertainties could come from the inherently-dynamic environment, the sensing and actuation noises, the disturbances to system operations due to environment interference, transient faults, and malicious attacks, as well as the current lack of methodologies for predicting the behavior of machine learning components (especially those based on deep neural networks) [2], [3].

In this paper, we consider the uncertainty in perception and its impact on system safety. Nowadays deep neural networks are being used prevalently for perception due to their improvement on average performance. However, it is often difficult to predict the behavior of those neural networks and offer any deterministic guarantees such as bounds on the perception inaccuracy. While robust controllers may be able to maintain stability in an asymptotic manner, the unbounded perception inaccuracy could lead the system into unsafe states (defined as in reach-avoid problems for control safety [4]) and cause disastrous consequences, as evidenced by a number of high-profile accidents of autonomous systems in automotive and avionics domains.

In the literature, there has been a large body of work studying the uncertainty of perception neural networks from the perspective of robustness under adversarial attacks [5], [6], i.e., how much the neural network output may change under a small (and often intentional) perturbation to its input. Such work has been mostly focused on the perception result itself, for tasks such as image classification and object detection [7], [8]. There has only been limited study recently on how those attacks may eventually affect system-level safety. For instance, in [9] it is demonstrated that putting an intentionally-designed patch on the road may lead to vehicles driving out of lane. The work in [10] shows adversarial examples that can continuously mislead the vehicle steering. However, the approaches still largely focus on the perception module itself, and do not provide any formal bounds or guarantees at the system level.

On the other hand, there has been extensive work on formally verifying control safety for autonomous systems. This includes safety verification methods for classical model-based controllers, e.g., those based on barrier certificates [11], [12] and Taylor models [13], [14]; emerging neural network-based controllers [4], [15]–[17]; and adaptive systems that switch among model-based and neural-network-based controllers [18], [19]. However, these approaches do not explicitly consider the perception module and its impact, particularly for neural network based perception.

We argue that to address the safety challenge of autonomous systems under uncertainty from perception neural networks, it is important to develop approaches that consider both perception and control in a *holistic* and *end-to-end* manner. That is, we will need to address open problems such as:

- *Runtime adversarial safety*: At runtime, under bounded perturbations to the perception neural network input, can the system remain safe for a finite time horizon?
- *Design-time safety assurance*: Given an error bound to the entire input space of the perception neural network, can the system always remain safe?
- *Safety-assured input error bound analysis*: Can we derive the maximum error bound on the input of the perception neural network, so that the system always remains safe?
- *Safety-driven perception or control design*: Given a perception neural network and an input error bound, can we design a controller (model-based or neural-network-based) to ensure system safety, and vice versa?
- *Safety-driven perception and control co-design*: Given an input error bound, can we co-design a perception neural network and a model-based or neural-network-based controller to ensure system safety?

In this paper, we will start with formally defining the above problems in Section II, discuss possible directions for addressing them, introduce our solution to the first problem (runtime adversarial safety) in Section III and a case study for it in Section IV, and then conclude the paper in Section V.

II. Problem Formulations

In this work, we consider autonomous systems that employ neural networks for perceiving the physical environment (plant) and then perform either model-based or neural network-based control accordingly. We call such systems neural network perception based autonomous systems, or NNP-AS.

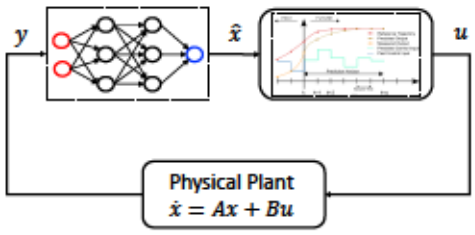

Fig. 1. Illustration of the system model for a neural network perception based autonomous system (NNP-AS).

Fig. 1 illustrates the system model for an NNP-AS. The plant model captures a physical system or process. For linear systems, it can be defined by a linear ordinary differential equation (ODE)

$$\dot{x} = Ax + Bu, \tag{1}$$

where $x \in \mathbb{R}^n$ is the n-dimensional system state and $u \in \mathbb{R}^m$ is the m-dimensional control input. The observation y is defined as a function of the system state x:

$$y = g(x).$$

The neural network-based perception module can be viewed as a function κ that maps the values of the observation (image) y to an estimated state $\hat{x}$. We then use $\Delta x = \hat{x} - x$ to denote the state estimation error.

The controller can be viewed as a function π that maps the values of the estimated state $\hat{x}$ to the control input u. Note that for neural-network-based controllers, it is often too complex to explicitly capture such a function, and approximation techniques could be used to facilitate analysis [4], [17].

The above NNP-AS works in the following way. Given a sample/control time step of size $\delta_c > 0$, at time $t = k\delta_c$ $(k = 0, 1, 2, \dots)$, the neural network perception module takes the observation $y(k\delta_c)$ as input and derives the estimated state $\hat{x}(k\delta_c)$. The controller then computes the control input $u(k\delta_c)$ for the next time step based on the the estimated state $\hat{x}(k\delta_c)$ and feeds it back to the plant. More precisely, the plant ODE becomes $\dot{x} = Ax + Bu(k\delta_c)$ in the time period of $[k\delta_c, (k+1)\delta_c)$ for $k = 0, 1, 2, \dots$. We assume that the state space is $X \subseteq \mathbb{R}^n$.

For safety-critical systems, the system state is required to be maintained within a safe pipe $\mathcal{P}$, which can be either time-invariant or time-variant, i.e., $\mathcal{P} = \mathcal{P}(t)$. Formally, system safety of an NNP-AS with a given perception module and a controller can be defined as:

Definition 1 (System Safety): Given a perception module κ and a controller π, an NNP-AS is safe if and only if $x(t) \in \mathcal{P}(t),\ \forall t \geq 0$.

In the following, we formally define the research problems stated in Section I that address the safety of an NNP-AS.

Runtime adversarial safety: During the operation of an NNP-AS, adversarial attacks or environment interference could introduce observation error Δy to the perception input y, which in turn may cause inaccuracy in state estimation and affect control safety. Thus, it is important to verify the system safety under such adversarial input at runtime. Formally, we define the following problem.

Problem 1 (Runtime Safety Verification under Adversarial Input): At time t, given the observation $y(t) = g(x(t)) + \Delta y(t)$, an observation error bound $\|\Delta y\| \leq \delta$ for a finite time horizon $[t, t+T]$, a perception module $\hat{x} = \kappa(y(t))$, and a controller $u = \pi(\hat{x})$, determine whether the system is safe, i.e., $x(\tau) \in \mathcal{P}(\tau)$, during the time horizon $\tau \in [t, t+T]$.

One possible solution to this problem is to first find an estimation error bound η where $\Delta x = \|\hat{x} - x\| \leq \eta$ (as a type of local robustness analysis), and then verify the controller safety under the estimation error, i.e., considering $u = \pi(\hat{x})$. Note that the state estimation error Δx in fact consists of two parts: one is caused by the observation error Δy, and the other is due to the algorithm inaccuracy of the perception module. In this work, we will propose an approach to address Problem 1 in Section III under the assumption that the perception module is ideally accurate, i.e., $\Delta x = 0$ when $\Delta y = 0$. Our approach can be easily extended to the cases where the algorithm inaccuracy can be bounded, although deriving such bound is in general very challenging.

Remark 1: In Problem 1, the estimation error bound η could be defined either over the entire observation space Y or a subset of Y that is around $y(t)$. It is typically much less pessimistic to define η on a smaller set $Y' \subseteq Y$ that is around $y(t)$, and then the verification needs to ensure that $y(\tau) \in Y',\ \forall \tau \in [t, t+T]$.

Design-time safety assurance: During design time, we could try to verify system safety over an infinite time horizon and the

entire observation space. Formally, this is defined as follows.

Problem 2 (Design-Time Safety Assurance): Given an input observation space Y and an observation error bound δ, such that at runtime any observation $y(t) \in Y \bigoplus \delta$ (where $\bigoplus$ denotes the Minkowski summation operator), a perception module $\hat{x} = \kappa(y(t))$, and a controller $u = \pi(\hat{x})$, determine whether the system always remains safe, i.e., $x(\tau) \in \mathcal{P}(\tau), \forall \tau \in [0, \infty)$.

To solve this problem, we could consider to derive the estimated state space error bound η with $Y \bigoplus \delta$, and then verify Problem 1 with $T \to \infty$.

Remark 2: The entire observation space Y is typically much larger than the subset Y' considered in Problem 1 around $y(t)$. Deriving the state estimation error bound in this case is equivalent to the global robustness analysis of the neural network, where we consider the local area robustness of the neural network for any possible point within the input space.

Definition 2 (Global Robustness): A neural network N is (δ', ϵ)-globally robust in input region D iff

$$\forall x_1, x_2 \in D, ||x_1 - x_2|| \leq \delta', \implies ||N(x_1) - N(x_2)|| \leq \epsilon.$$

Here $N(x_i)$ is the neural network output of x_i. Note that the global robustness analysis of neural networks is NP-hard [20] and much harder than the local robustness analysis [21]. The approaches in the literature [20], [21] are limited to small and simple networks.

Safety-assured input error bound analysis: In some cases, we are interested in deriving a bound on how much observation error (due to either adversarial input or noise) the system can sustain to ensure its safety over a finite time horizon. The problem can be defined as follows.

Problem 3 (Safety-Assured Input Error Bound Analysis): Derive an observation error bound δ, such that for any observation error $\|\Delta y\| \leq \delta$, a perception module $\hat{x} = \kappa(y(t))$, and a controller $u = \pi(\hat{x})$, the system is safe over a finite time horizon, i.e., $x(\tau) \in \mathcal{P}(\tau), \forall \tau \in [t, t+T]$.

This problem could be addressed by sequentially solving the following two subproblems.

Problem 4 (Input Error Bound Analysis for Control): Given a controller $u = \pi(\hat{x})$, derive a state estimation error bound η, such that for any state estimation error $\|\Delta x\| \leq \eta$, the system is safe over a finite time horizon, i.e., $x(\tau) \in \mathcal{P}(\tau), \forall \tau \in [t, t+T]$.

For model-based controllers, Problem 4 could be solved by directly leveraging finite-time safety tools such as Flow* [13]. For neural-network-based controllers, it can be addressed by approximation techniques, e.g., using Bernstein polynomials [4].

Problem 5 (Input Error Bound Analysis for Perception): Given a perception module $\hat{x} = \kappa(y(t))$ and an output error bound η, derive an input bound δ, such that for any observation error $\|\Delta y\| \leq \delta$, the output $\hat{x}$ stays within the neighbourhood of x with respect to η, i.e., $\|\Delta x\| \leq \eta$.

Problem 5 is generally difficult to solve, since the inverse of a neural network is not necessarily a function. A naïve approach is to leverage the bisection method: Give a value of δ and compute the output range. Bisect δ, until $\Delta x > \eta$.

Note that the above three problems can also be defined over an infinite time horizon and the entire observation space.

Safety-driven perception or control design: We could explore the design of the perception module or the controller for system safety, as defined below.

Problem 6 (Safety-Driven Perception Design): Given an input observation space Y, an observation error bound δ (i.e., $y(t) \in Y \bigoplus \delta$), and a controller $u = \pi(\hat{x})$, design a perception module $\hat{x} = \kappa(y(t))$, so that the system always remains safe, i.e., $x(\tau) \in \mathcal{P}(\tau), \forall \tau \in [0, \infty)$.

Problem 7 (Safety-Driven Control Design): Given an input observation space Y, an observation error bound δ (i.e., $y(t) \in Y \bigoplus \delta$), and a perception module $\hat{x} = \kappa(y(t))$, design a controller $u = \pi(\hat{x})$, so that the system always remains safe, i.e., $x(\tau) \in \mathcal{P}(\tau), \forall \tau \in [0, \infty)$.

Safety-driven perception and control co-design: We could also co-design the perception module and the controller for system safety, as defined below.

Problem 8 (Safety-Driven Perception and Control Co-design): Given an input observation space Y and an observation error bound δ (i.e., $y(t) \in Y \bigoplus \delta$), design a perception module $\hat{x} = \kappa(y(t))$ and a controller $u = \pi(\hat{x})$, so that the system always remains safe, i.e., $x(\tau) \in \mathcal{P}(\tau), \forall \tau \in [0, \infty)$.

III. Our Approach for Runtime Adversarial Safety

In this section, we propose an approach for addressing the Problem 1 defined in Section II. To formally analyze the impact of perception uncertainty on system safety, our approach combines output range analysis of neural networks [22] and reachability analysis for control safety verification. More specifically, we first derive the (over-approximated) output range of the perception neural network under input uncertainty (considering observation error). Such range captures the state estimation error bound, which can then be leveraged to compute the finite-state reachable set for safety verification. This approach is detailed below.

A. Bounding State Estimation Error via Output Range Analysis

Given an observation error bound δ, at each time step t, the observation $y(t)$ falls into the set $Y' = \{y \mid g(x(t)) + \Delta y, \|\Delta y\| \leq \delta\}$. We can then compute the estimated system state set $\hat{X} = \kappa(Y')$, with estimation error bounded by $\eta = \max_{\hat{x} \in \hat{X}} \|\hat{x} - x\|$, via output range analysis as defined below.

Problem 9 (Output Range Analysis of Neural Networks): Given a neural network κ and an input range Y', compute $\hat{X} = \kappa(Y')$ or its overapproximation.

A more precise estimation of the output range for the perception neural network κ reduces the state estimation error and facilitates the verification of control safety. Thus we adopt the state-of-the-art refinement-based approach in [22] for more accurately estimating the output range. The main idea is to use multiple polytopes for approximating a nonlinear activation function in the neural network with a mixed integer linear programming (MILP) formulation (Fig. 2 shows an illustrating example on ReLU). The detailed steps are as follows.

- Step 1: Compute the interval relaxation for each neuron with a propagation-based method;

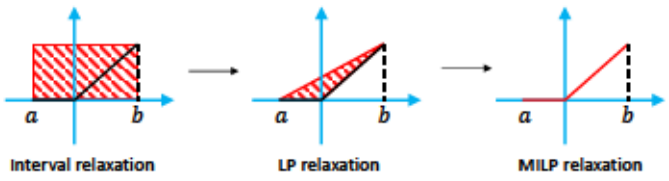

Fig. 2. Approximating a ReLU function [22]: In interval relaxation, ReLU is coarsely approximated as a rectangle. In LP relaxation with a linear polytope, a much tighter approximation is used to capture the input-output relation. In MILP relaxation with multiple polytopes (in this case two), the approximation is further refined (in this case equivalent transformtion is achieved).

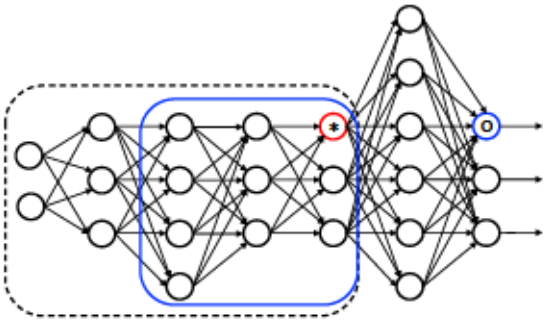

Fig. 3. Refinement-based output range analysis [22]: Refine the approximated value range of neurons in hidden layers (e.g., the red neuron with a star symbol) will help refine the output range approximation (e.g. the blue neuron with a circle symbol). To efficiently refine the red neuron, we consider the constraints of the previous two layers (within the blue solid rectangle), rather than all the previous layers (within the black dashed rectangle).

- Step 2: Construct a basic linear programming (LP) relaxation for each neuron;
- Step 3: Select important neurons to conduct layer-by-layer refinement based on an LP or MILP formulation that only encodes the constraints from several previous layers, rather than all the previous layers, for efficiency purposes;
- Step 4 (optional): The above refinement step can be repeated multiple times.

Step 1 acts as an initialization step, computing a basic sound guess of the input interval for each neuron. Step 2 constructs an LP relaxation that is tighter than the interval abstraction. Step 3 further tightens the intervals with LP or MILP. For efficiency, we only consider a sliding window of several previous layers in encoding the constraints for LP/MILP. An illustrating example is shown in Fig. 3. An optional Step 4 decides how many times the refinement in Step 3 should be repeated. This refinement-based output range analysis approach provides a balanced tradeoff between estimation precision and analysis complexity.

B. Safety Verification via Reachability Analysis

The output range analysis of the perception neural network provides a bounded estimation of system state, i.e., $x(t) \in \hat{X}$. We can then compute the reachable set $R_{T,\hat{X}}$ for a finite-horizon T to verify the system safety. The definition of reachable set and how it is computed is explained below.

Definition 3: A reachable set $R_{T,\hat{X}}$ is defined as all the states that can possibly be visited within time T (or T steps in a discretized system) from the current system set $\hat{X}$, i.e.,

$$R_{T,\hat{X}} = \{x(t) \mid \dot{x} = Ax + Bu, x(0) \in \hat{X}\}, 0 \leq t \leq T.$$

Remark 3: Such a reachable set $R_{T,\hat{X}}$ is typically an over-approximation of the underlying true system trajectory. That is, the system can never go out of $R_{T,\hat{X}}$ for the next T steps. Thus, if the reachable set is verified to be safe, i.e., $R_{T,\hat{X}} \cap \neg\mathcal{P} = \emptyset$, then the system must be safe within the next T steps.

During system operation, we may iteratively compute the reachable set $R_{T,\hat{X}}$ to perform online verification on system safety with $\hat{X}$ updated accordingly. Next, we introduce how this can be done for a linear time-invariant (LTI) system.

Reachability analysis for an LTI system under external disturbance. We address the LTI system defined in Equation (1), with added consideration of other external disturbances (beyond adversarial input) or algorithm/model inaccuracy, captured by a disturbance vector w on system state, i.e.,

$$\dot{x} = Ax + Bu + w.$$

As the sample/control time step is δ_c, and the control input $u(t) = u(k\delta_c)$ for $t \in [k\delta_c, (k+1)\delta_c)$, the LTI ODE can be discretized as:

$$x[k+1] = A_d x[k] + B_d u[k] + Ew[k],$$

where $A_d = e^{A\delta_c}$, $B_d = \int_0^{\delta_c} e^{At} B dt$, and $E = \int_0^{\delta_c} e^{At} dt$. The additional disturbance is bounded as $w \in W$, and the system is controlled by a linear feedback controller $u[k] = K\hat{x}[k]$. From the output range analysis of the perception module, the state estimation error can be bounded by $\Delta x = x[k] - \hat{x}[k] \in [-\eta, \eta]$. Assuming the state estimation error $\hat{X}$ and disturbance error W are both polyhedrons, given the initial/current estimated state $\hat{x}[0]$, the reachable set for the next T steps can be derived via recursion. Specifically, given $x[k] \in X_k$, the range of $x[k+1]$, namely X_{k+1} can be derived by

$$X_{k+1} = (A_d + B_d K)X_k \oplus (-B_d K)[-\eta, \eta] \oplus EW,$$

where $\oplus$ is the Minkowski sum operation, and the initial state space $X_0 = \hat{X}$.

IV. Case Studies

In this section, we illustrate the proposed approach with two case studies that focus on each of the two steps, respectively. In the first case study, we demonstrate the feasibility of bounding state estimation error via output range analysis of the perception neural network. In the second case study, we demonstrate the verification of system safety under a given state estimation error bound and the analysis on how different state estimation error bounds affect the system safety. Note that while our proposed approach in Section III is a holistic end-to-end approach, the two case studies are separated due to the current limitations on examples and approach efficiency, which we plan to address in our future work.

A. Output Range Analysis for Perception Neural Networks

For the perception module, we consider a small neural network that classifies the parity of the digit on the MNIST dataset. The neural network takes a 28×28 grayscale image that contains a handwritten digit as input. The output shows

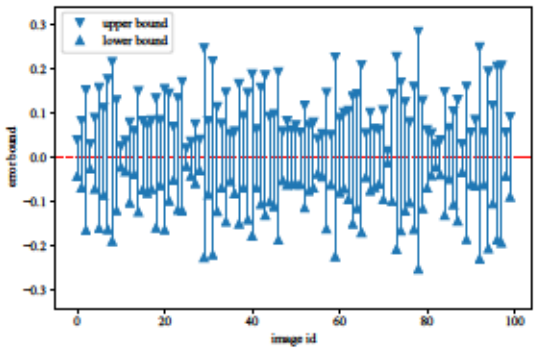

Fig. 4. Error bounds for NN-I (no activation function for final layer).

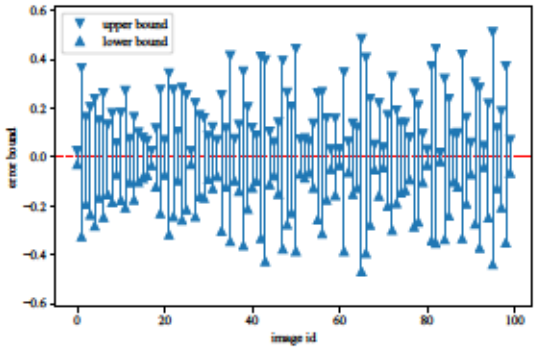

Fig. 5. Error bounds for NN-II (ReLU activation function for final layer).

whether the digit is odd or even, i.e., the digit is classified to be even if the output is close to 0 and odd if close to 1. The neural network contains two convolution layers that are followed by two fully-connected layers. ReLU is used as the activation function for convolution layers. The first fully-connected layer is nonlinearized by Sigmoid, while the second fully-connected layer has three possible forms: I) having no activation function, II) using ReLU activation, or III) using Sigmoid activation. This results in three neural networks for our experiments, called NN-I, NN-II, and NN-III, respectively.

The output range is analyzed by our LayR tool [22] using the refinement-based approach introduced in Section III. We choose the output of the last fully-connected layer, a continuous value before classification, as the system state to evaluate (to mimic the typical continuous system state in control). We assume an observation error bound of $[-0.01, 0.01]$ for each input pixel, and evaluate the average and maximum state estimation error bounds for 100 images in the MNIST test set. Figs. 4, 5, and 6 show the estimation bounds for each image from the test set for the three perception neural networks. These results are summarized in Table I, showing the range of states, the average state estimation error bound η_{ave}, and the maximum error bound η_{max} across 100 testing images for each network.

These results demonstrate the feasibility of our approach in bounding state estimation error for perception neural networks. Note that for a well-trained perception neural network, η_{max} could still be quite large. This is likely due to the pessimistic over-approximation for the state estimation error bound, which

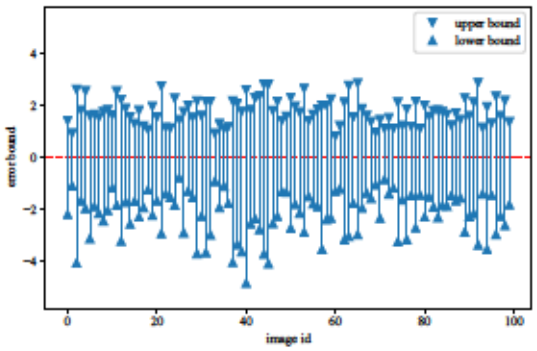

Fig. 6. Error bounds for NN-III (Sigmoid activation function for final layer).

TABLE I
STATE SPACE, AVERAGE AND MAXIMUM STATE ESTIMATION ERROR BOUNDS FOR 100 IMAGES IN MNIST FOR THREE PERCEPTION NETWORKS.

Networks	state range	η_{ave}	η_{max}
NN-I	$[-0.09, 1.19]$	$[-0.10, 0.11]$	$[-0.25, 0.29]$
NN-II	$[-2.12, 1.23]$	$[-0.19, 0.19]$	$[-0.47, 0.51]$
NN-III	$[-15.9, 14.7]$	$[-2.23, 1.80]$	$[-4.88, 2.90]$

should be further addressed in future work.

B. Safety Verification under Bounded State Estimation Error

In this case study, we demonstrate our reachability analysis for verifying system control safety under a given state estimation error bound η (which could be from the output range analysis of the perception neural network) and other bounded external disturbance w. We consider an LTI system that controls a DC motor [23]. The system state $x = [\omega, i]^\top$ includes the shaft rotational speed ω and the current i on the armature circuit. The control input u is the voltage of the armature circuit supplied by a voltage source. The ODE of the system is:

$$\dot{x} = \begin{bmatrix} -10 & 1 \\ -0.02 & -2 \end{bmatrix} x + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u + w.$$

In this work, we consider the safe space of the DC motor as $-0.5 \leq \omega \leq 0.5$, $-0.5 \leq i \leq 0.5$. The sample/control step size is set to $\delta_c = 20$ ms. We choose four different linear feedback controllers $u = K^i \hat{x}$ $(i = 1, 2, 3, 4)$, where $K^1 = [-1.3150, -2.5255]$, $K^2 = [1.9249, -0.2608]$, $K^3 = [-3.9975, 0.0501]$, and $K^4 = [1.1112, -1.7833]$, which are derived from various pole placements.

First, we consider that the state estimation error bound is given as $[-0.05, 0.05]$ for both dimensions of the system state, and the external disturbance w is also bounded by $[-0.05, 0.05]$. At time $t = 0$, the estimated state is $\hat{x}[0] = [0.45, 0.45]$. The reachability sets of each controller can then be computed by our approach, and are shown in Fig. 7. We can observe that even though all controllers are asymptotically stable, one of them (controller 2) cannot be verified to be safe within the next 15 steps, while the others' safety can be verified. This shows the effectiveness of our approach in verifying system safety under a bounded state estimation error and the importance of choosing a controller based on such safety verification.

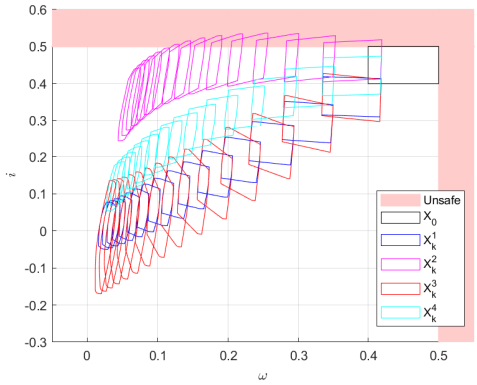

Fig. 7. Reachable sets for the four different controllers. The initial state is estimated to be $[0.45, 0.45]$. The state estimation error bound is $[-0.05, 0.05]$ for both system states, resulting in a rectangle area for estimated state space. X_0 is the initial state space and X_k^i is the reachable set of controller i at the k-th step. We can see that the safety of controller 2 cannot be verified, while the safety of the others can be verified.

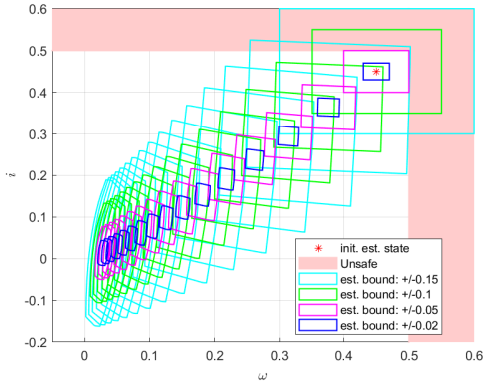

Fig. 8. Reachable sets for different state estimation error bounds, ranging from $[-0.02, 0.02]$ to $[-0.15, 0.15]$. The initial estimated state is $[0.45, 0.45]$.

In Fig. 8, we further demonstrate how different state estimation error bounds may affect the system safety. Here, the reachable set of controller 1 is analyzed for a state estimation error bound ranging from $[-0.02, 0.02]$ to $[-0.15, 0.15]$. We can see that the system safety can be guaranteed when $\eta \leq 0.05$, but cannot when $\eta \geq 0.1$. Such analysis could be very helpful for choosing the right perception neural network design.

V. Conclusion

In this paper, we discussed the importance of considering perception neural network uncertainty in addressing the safety of autonomous systems. We formally defined a number of open research problems, and proposed an end-to-end approach for addressing one of them, i.e., runtime safety verification under adversarial input to the perception neural network. This is a small first step in tackling the safety of neural network perception based autonomous systems (NNP-AS).

Acknowledgments: We gratefully acknowledge the support from NSF grants 1834701, 1834324, 1839511, 1724341, 2038960 and ONR grant N00014-19-1-2496.

References

[1] Q. Zhu and A. Sangiovanni-Vincentelli, "Codesign methodologies and tools for cyber–physical systems," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1484–1500, Sep. 2018.

[2] Q. Zhu, W. Li, H. Kim, Y. Xiang, K. Wardega, Z. Wang, Y. Wang, H. Liang, C. Huang, J. Fan, and H. Choi, "Know the unknowns: Addressing disturbances and uncertainties in autonomous systems : Invited paper," in *ICCAD*, 2020, pp. 1–9.

[3] Q. Zhu, C. Huang, R. Jiao, S. Lan, H. Liang, X. Liu, Y. Wang, Z. Wang, and S. Xu, "Safety-assured design and adaptation of learning-enabled autonomous systems," *ASP-DAC*, 2021.

[4] C. Huang, J. Fan, W. Li, X. Chen, and Q. Zhu, "Reachnn: Reachability analysis of neural-network controlled systems," *TECS*, vol. 18, no. 5s, pp. 1–22, 2019.

[5] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *CVPR*, 2018, pp. 1625–1634.

[8] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *ECML-PKDD*. Springer, 2018, pp. 52–68.

[9] T. Sato, J. Shen, N. Wang, Y. J. Jia, X. Lin, and Q. A. Chen, "Hold tight and never let go: Security of deep learning based automated lane centering under physical-world attack," *preprint arXiv:2009.06701*, 2020.

[10] Z. Kong, J. Guo, A. Li, and C. Liu, "Physgan: Generating physical-world-resilient adversarial examples for autonomous driving," in *CVPR*, 2020, pp. 14 254–14 263.

[11] C. Huang, X. Chen, W. Lin, Z. Yang, and X. Li, "Probabilistic safety verification of stochastic hybrid systems using barrier certificates," *TECS*, vol. 16, no. 5s, p. 186, 2017.

[12] Z. Yang, C. Huang, X. Chen, W. Lin, and Z. Liu, "A linear programming relaxation based approach for generating barrier certificates of hybrid systems," in *FM*. Springer, 2016, pp. 721–738.

[13] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *CAV*. Springer, 2013, pp. 258–263.

[14] E. Goubault and S. Putot, "Forward inner-approximated reachability of non-linear continuous systems," in *HSCC*. ACM Press, 2017, pp. 1–10.

[15] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *HSCC*. ACM Press, 2019, pp. 169–178.

[16] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *HSCC*. ACM Press, 2019, pp. 157–168.

[17] J. Fan, C. Huang, W. Li, X. Chen, and Q. Zhu, "Reachnn*: A tool for reachability analysis ofneural-network controlled systems," in *ATVA*, 2020.

[18] C. Huang, S. Xu, Z. Wang, S. Lan, W. Li, and Q. Zhu, "Opportunistic intermittent control with safety guarantees for autonomous systems," *DAC*, 2020.

[19] Y. Wang, C. Huang, and Q. Zhu, "Energy-efficient control adaptation with safety guarantees for learning-enabled cyber-physical systems," in *ICCAD*. IEEE, 2020, pp. 1–9.

[20] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, and M. Kwiatkowska, "Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance." IJCAI, 2019.

[21] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *CAV*. Springer, 2017, pp. 97–117.

[22] C. Huang, J. Fan, X. Chen, W. Li, and Q. Zhu, "Divide and slide: Layer-wise refinement for output range analysis of deep neural networks," *TCAD*, vol. 39, no. 11, pp. 3323–3335, 2020.

[23] "Control tutorials for matlab and simulink - DC motor speed: System modeling." [Online]. Available: https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling