2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Learning Region of Attraction for Nonlinear Systems

Shaoru Chen, Mahyar Fazlyab, Manfred Morari, George J. Pappas, Victor M. Preciado

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9682880

***Abstract*—Estimating the region of attraction (ROA) of general nonlinear autonomous systems remains a challenging problem and requires a case-by-case analysis. Leveraging the universal approximation property of neural networks, in this paper, we propose a counterexample-guided method to estimate the ROA of general nonlinear dynamical systems provided that they can be approximated by piecewise linear neural networks and that the approximation error can be bounded. Specifically, our method searches for robust Lyapunov functions using counterexamples, i.e., the states at which the Lyapunov conditions fail. We generate the counterexamples using Mixed-Integer Quadratic Programming. Our method is guaranteed to find a robust Lyapunov function in the parameterized function class, if exists, after collecting a finite number of counterexamples. We illustrate our method through numerical examples.**

I. Introduction

Certifying the stability of nonlinear dynamical systems described by differential or difference equations is a long-standing fundamental problem in control theory. A general method for proving stability is the method of Lyapunov whereby one attempts to find, from a hypothesis class, a positive definite function that decreases along the trajectories of the dynamical system [1, 2]. Once a Lyapunov function is found, its level sets can be used to estimate the region of attraction (ROA) of the equilibrium, which is a useful tool in constructing safety certificates [3, 4].

In general, in order to find Lyapunov functions for nonlinear systems, one needs to solve a non-convex optimization problem. To avoid intractable non-convex optimization problems, there has been an increasing interest in developing data-driven and machine-learning inspired methods for stability analysis. For example, the authors in [5] parameterize the Lyapunov function as a neural network and train it through stochastic gradient descent to learn the largest ROA of a nonlinear system. Giesl et al. [6] learn a Lyapunov function as a reproducing kernel Hilbert space predictor from noisy trajectories of a nonlinear system. Boffi et al. [7] treat Lyapunov function synthesis as a machine learning problem and provide generalization error bounds for the learned Lyapunov function.

Although success examples have been demonstrated for the learning-based methods, the guarantees on the stability certificates are often of probabilistic nature since only sampled trajectory data are used. On the other hand, feedforward neural network or recurrent neural network models obtained from learning-based system identification are shown to have a strong predictive power [8] and can approximate complex nonlinear dynamics with small errors. This offers an opportunity to obtain deterministic stability guarantees on nonlinear systems by analyzing their neural network approximations with the model errors taken into account. However, how to analyze the stability of uncertain neural network dynamics remains a challenge with only limited work available [9, 10].

In this paper, we propose a sampling-based method for constructing robust Lyapunov functions (see Theorem 1 for definition) for a class of uncertain dynamical systems with feedforward ReLU networks modeling the nominal dynamics over a compact region of interest (ROI). Our method consists of a learner, which is responsible for generating Lyapunov function candidates, and a verifier, which validates or rejects the proposed candidates with counterexamples to guide the search. More importantly, if a Lyapunov function exists in the parameterized Lyapunov function class, our method is guaranteed to find one in a finite number of steps. These properties are particularly desirable for analysis of uncertain neural network dynamical systems. For comparison, the convex relaxation based methods [10] easily become excessively conservative as the size of the neural network or the model uncertainty increases. The counterexample-guided Lyapunov function synthesis methods in [11–13] do not have finite-step termination guarantees which means that the number of calls to the computationally expensive verifier cannot be bounded. Our method is inspired by [9] which proposes a sampling-based method for stability analysis of autonomous hybrid systems. We extend the method in [9] to handle uncertain dynamical systems and demonstrate the potential of our proposed method to automate Lyapunov function synthesis for nonlinear systems through their neural network dynamics approximations.

We state the assumption on the uncertain neural network dynamical system and the problem formulation in Section II. In Section III, the parameterization of robust Lyapunov function is introduced and a sampling-based method to synthesize a robust Lyapunov function is presented in Section IV. The practical use of the proposed framework is discussed in Section V with numerical examples given in Section VI to demonstrate our method. Section VII concludes the paper.

Shaoru Chen, Manfred Morari, George J. Pappas, and Victor M. Preciado are with the Department of Electrical and Systems Engineering, University of Pennsylvania. Email: srchen, morari, pappasg, preciado@seas.upenn.edu. Mahyar Fazlyab is with the Mathematical Institute for Data Science, Johns Hopkins University. Email: mahyarfazlyab@jhu.edu.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

II. Problem Statement

Consider a discrete-time nonlinear dynamical system

$$x_+ = f(x) \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the state, x_+ denotes the state at the next time instance, and $f : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_x}$ is a locally Lipschitz continuous function. We assume that system (1) has an equilibrium $x = 0$, i.e., $0 = f(0)$, and is asymptotically stable at the origin. Verifying that system (1) is locally stable around the origin can be done by analyzing the linearization of the system at $x = 0$ if it exists. Of more practical use is estimating the region of attraction of the origin.

Definition 1 (Region of attraction)**.** *The region of attraction for system* (1) *at the origin is* $\mathcal{O} = \{x_0 | \lim_{k\to\infty} x_k = 0\}$ *where* x_k *denotes the state of the system at time* $k = 0, 1, 2, \cdots$.

The ROA characterizes how much the system can be perturbed from $x = 0$ without diverging. Therefore, the ROA is often used to describe the safety of nonlinear systems. In this work, we are also interested in the region where the convergence of system trajectories to a small neighborhood of the origin can be verified since this relaxed notion of ROA also suffices to show safety of nonlinear systems in practical applications.

Computing the exact ROA $\mathcal{O}$ for general nonlinear systems is infeasible and hence, one seeks to find an estimate $\tilde{\mathcal{O}}$ of ROA for specific classes of dynamical systems. In this work, we assume that the map f can be approximated by a piecewise linear multi-layer neural network f_{NN} and that the approximation error $f - f_{NN}$ can be bounded. Specifically, consider a neural network with ReLU[1]activation:

$$\begin{aligned} z_0 &= x \\ z_{\ell+1} &= \max(W_\ell z_\ell + b_\ell, 0), \quad \ell = 0, \cdots, L-1 \\ f_{NN}(x) &= W_L z_L + b_L \end{aligned} \tag{2}$$

where $z_0 = x \in \mathbb{R}^{n_0}$ $(n_0 = n_x)$ is the input to the neural network, $z_{\ell+1} \in \mathbb{R}^{n_{\ell+1}}$ is the the output vector of the $(\ell+1)$-th hidden layer with $n_{\ell+1}$ neurons, $f_{NN}(x) \in \mathbb{R}^{n_{L+1}}$ $(n_{L+1} = n_x)$ is the output of the neural network, and $W_\ell \in \mathbb{R}^{n_{\ell+1}\times n_\ell}, b_\ell \in \mathbb{R}^{n_{\ell+1}}$ are the weight matrix and the bias vector of the $(\ell+1)$-th hidden layer.

Assumption 1. *For a given Schur stable matrix* $A \in \mathbb{R}^{n_x \times n_x}$ *and a compact region of interest (ROI)* $\mathcal{X} \subset \mathbb{R}^{n_x}$ *with* $0 \in$ *int*$(\mathcal{X})$*, where int*$(\mathcal{X})$ *denotes the interior of* $\mathcal{X}$*, the map* $f(x)$ *can be decomposed as*

$$f(x) = Ax + f_{NN}(x) + w(x), \quad \forall x \in \mathcal{X} \tag{3}$$

where the approximation error $w(x) \in \mathbb{R}^{n_x}$ *satisfies*

$$\|w(x)\|_\infty \le \gamma \|x\|_\infty + \delta, \quad \forall x \in \mathcal{X}, \tag{4}$$

for some known positive constants γ, δ.

[1]ReLU (rectified linear unit) activation function is defined by $\phi(x) = \max(0, x)$ for $x \in \mathbb{R}^n$ where the maximum is applied entry-wise.

Now consider the uncertain dynamical system

$$x_+ = \underbrace{Ax + f_{NN}(x)}_{:=\hat{f}(x)} + w, \quad w \in \mathcal{W}(x) \tag{5}$$

where $\mathcal{W}(x) = \{w \in \mathbb{R}^{n_x} \mid \|w\|_\infty \le \gamma\|x\|_\infty + \delta\}$ is a state dependent set that contains the "disturbance" w. Under Assumption 1, the nonlinear dynamics $f(x)$ is contained in the uncertain dynamics class described by (5) which enjoys a specific structure that can be exploited for stability analysis. The main problem considered in this paper is stated as follows.

Definition 2 (Robust ROA)**.** *Let* x_k*,* w_k *denote the state and disturbance of the uncertain system* (5) *at time* $k = 0, 1, \cdots$*, respectively, and* $B \in \mathcal{X}$ *be a small neighborhood of the origin with* $0 \in$ *int*(B)*. The robust ROA of the uncertain system* (5) *is defined as* $\mathcal{O} = \{x_0 | \lim_{k\to\infty} x_k \in B, \forall w_k \in \mathcal{W}(x_k)\}$.

Problem 1. *Find an estimate of the robust ROA of the uncertain neural network dynamical system* (5)*.*

In the Section III and IV, we propose a sampling-based method to synthesize a robust Lyapunov function which identifies an estimate of the robust ROA $\tilde{\mathcal{O}}$ for the uncertain system (5). By virtue of Assumption 1, $\tilde{\mathcal{O}}$ would be a valid inner estimate of ROA for system (1). We postpone the discussion on the applicability of Assumption 1 to Section V.

III. Parameterization of Robust Lyapunov functions

For the uncertain system (5), we aim to find a robust Lyapunov function that monotonically decreases for all possible realization of uncertainty. Once a robust Lyapunov function is found, its sublevel sets encode an inner estimate of the robust ROA for the uncertain system (5), and hence the original system (1).

A. Robust Lyapunov function

For the uncertain system (5) with the uncertainty set $\mathcal{W}(x)$, we aim to show that the system trajectories converge to a neighborhood $\mathcal{B}$ of the origin. The neighborhood $\mathcal{B}$ is chosen according to the practical need of verification. Let $B_r = \{x \in \mathbb{R}^{n_x} | \|x\|_\infty \le r\} \subset \mathcal{X}$ denote an ℓ_∞ norm ball with radius $r > 0$ contained inside the ROI. The robust Lyapunov function is defined in the following theorem.

Definition 3 (Successor set)**.** *For the uncertain dynamics* (5)*, we denote the successor set from a set* $\mathcal{B}$ *as* $succ(\mathcal{B}) = \{y \in \mathbb{R}^{n_x} | \exists x \in \mathcal{B}, w \in \mathcal{W}$ *s.t.* $y = \hat{f}(x) + w\}$.

Theorem 1. *Consider the discrete-time uncertain system* (5) *and assume there exists a continuous function* $V(x) : \mathcal{X} \mapsto \mathbb{R}$ *satisfying*

$$V(x) > 0, \forall x \in \mathcal{X} \setminus B_r, \tag{6a}$$

$$V(\hat{f}(x) + w) - V(x) < 0, \forall x \in \mathcal{X} \setminus B_r, \forall w \in \mathcal{W}(x). \tag{6b}$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

Denote $\underline{\Omega}_{succ(B_r)}(V)$ the smallest sublevel set of $V(x)$ that contains the successor set of B_r and $\bar{\Omega}_{\mathcal{X}}(V)$ the largest sublevel set of $V(x)$ that is contained in $\mathcal{X}$. Then for all initial states $x_0 \in \bar{\Omega}_{\mathcal{X}}(V)$ of the uncertain system (5)*, we have* $\lim_{k\to\infty} x_k \in \underline{\Omega}_{succ(B_r)}(V)$.

Proof. All trajectories starting from $x_0 \in \bar{\Omega}_{\mathcal{X}}(V)$ reach B_r in a finite number of steps since $V(x)$ is upper bounded in $\mathcal{X}$ and its decrease in each step is lower bounded by a positive number as long as $x \notin B_r$. Once the state x_k reaches B_r, it is mapped to $x_{k+1} \in \text{succ}(B_r) \subset \underline{\Omega}_{\text{succ}(B_r)}(V)$ in the next step. Whether $x_{k+1} \notin B_r$ or $x_{k+1} \in B_r$, the trajectory starting from x_{k+1} will always stay in $\underline{\Omega}_{\text{succ}(B_r)}(V)$. □

We call any $V(x)$ satisfying constraints (6a) and (6b) a *robust Lyapunov function* and those satisfying constraint (6a) a robust Lyapunov function candidate. Theorem 1 states that a robust Lyapunov function $V(x)$ certifies the convergence of system trajectories under uncertainty to a neighborhood of the origin whose size and shape is jointly decided by $V(x)$ and B_r. Correspondingly, the largest sublevel set of $V(x)$ contained in the ROI $\mathcal{X}$ comprises an estimate of robust ROA for the uncertain system, and $\tilde{\mathcal{O}} = \bar{\Omega}_{\mathcal{X}}(V)$ is also an inner-approximation of ROA of the original nonlinear system (1). In the rest of the paper, we refer to robust Lyapunov functions simply as Lyapunov functions.

B. Lyapunov function parameterization

To make the search for a Lyapunov function $V(x)$ tractable, we parameterize the Lyapunov function candidate as a quadratic function using predicted nominal trajectories of the uncertain system (5) for a finite horizon as the basis. For a horizon of $k \geq 0$, the basis is given by

$$z_k(x) = [x^\top \ \hat{f}^{(1)}(x)^\top \ \hat{f}^{(2)}(x)^\top \ \cdots \ \hat{f}^{(k)}(x)^\top]^\top$$

where $\hat{f}^{(i+1)}(x) = \hat{f}(\hat{f}^{(i)}(x))$ for $i \geq 0$ and $\hat{f}^{(0)}(x) = x$. By denoting $\mathbb{S}^n$ the set of $n \times n$-dimensional symmetric matrices, and $\mathbb{S}^n_+$ ($\mathbb{S}^n_{++}$) the set of $n \times n$-dimensional positive semidefinite (definite) matrices, the Lyapunov function candidate of *order* k is defined as

$$V_k(x; P) = z_k^\top(x) P z_k(x), \quad P \in \mathbb{S}^{n_x}_{++}. \tag{7}$$

This parameterization is inspired by the non-monotonic Lyapunov function [14] and finite-step Lyapunov function [15] methods which construct Lyapunov function candidates using system states several steps ahead. By construction, $V_k(x; P)$ is positive definite and readily satisfies constraint (6a). In addition, the parameterization of $V_k(x; P)$ has two desirable properties: (i) $V_k(x; P)$ is a linear function in P when the state x is fixed, and (ii) for $k \geq 1$, $V_k(x; P)$ is a piecewise quadratic function in x and its complexity can be easily tuned by the order k.

The first property indicates that the set of valid Lyapunov function candidates is convex. To show this, define the Lyapunov difference function as

$$\Delta V_k(x, w, P) = V_k(\hat{f}(x) + w; P) - V_k(x; P). \tag{8}$$

From the first property of $V_k(x; P)$, we know that $\Delta V_k(x, w, P)$ is also a linear function in P when the state x and uncertainty w are fixed. We describe the set of valid Lyapunov functions in the parameter space as

$$\begin{aligned}\mathcal{F} = \{P \in \mathbb{S}^{n_x} | 0 \prec P \preceq I, \Delta V_k(x, w, P) < 0, \\ \forall x \in \mathcal{X} \setminus B_r, \forall w \in \mathcal{W}(x)\}\end{aligned} \tag{9}$$

and denote $\mathcal{F}$ the *target set*. It follows that $\mathcal{F}$ is a convex set since it is defined by two linear matrix inequalities and infinitely many linear inequalities in P. The convexity of $\mathcal{F}$ is essential for the design of our main algorithm to search for Lyapunov functions.

The second property follows from the fact that the ReLU network $\hat{f}(x)$ is a piecewise affine function and so are $\hat{f}^{(i)}(x)$. By tuning the order k, we can parameterize functions with high complexity, i.e., piecewise quadratic functions with a large amount of partitions, by using a relatively small number of parameters.

After identifying the set of Lyapunov functions $\mathcal{F}$, our goal becomes finding a feasible point in $\mathcal{F}$ or proves that $\mathcal{F}$ is empty with the given parameters. Although the set $\mathcal{F}$ is convex, finding a feasible point in it is challenging due to the infinitely many constraints in (9) and the complexity of the underlying uncertain neural network dynamics (5).

IV. Sampling-based Lyapunov function synthesis

In this section, we extend the sampling-based method in [9] to synthesize a Lyapunov function for the uncertain system (5) with neural network nominal dynamics. Specifically, the proposed method is guaranteed to find a Lyapunov function in the target set $\mathcal{F}$ in a finite number of iterations when $\mathcal{F}$ is full-dimensional, and is capable of providing an infeasibility certificate of $\mathcal{F}$ when $\mathcal{F}$ is empty. Similar to [9], we obtain this guarantee by sampling according to the analytic center cutting-plane method [16, 17] from convex optimization. A step-by-step explanation is given below.

A. Localization of the target set

To handle the infinitely many linear constraints in (9), we first observe that an over-approximation $\tilde{\mathcal{F}}$ of the target set $\mathcal{F}$ can be obtained by relaxing the Lyapunov difference constraint in (9) to hold only on a sample set with a finite number of state-disturbance pairs $\mathcal{S} = \{(x_i^S, w_i^S)\}_{i=1}^N$, i.e.,

$$\tilde{\mathcal{F}} = \{P \in \mathbb{S}^{n_x} | 0 \prec P \preceq I, \Delta V_k(x, w, P) < 0, \forall (x, w) \in \mathcal{S}\}$$

with samples $x_i^S \in \mathcal{X} \setminus B_r, w_i^S \in \mathcal{W}(x_i^S)$ for $i = 1, \cdots, N$. We call $\tilde{\mathcal{F}}$ a localization set of $\mathcal{F}$ and $\mathcal{F} \subseteq \tilde{\mathcal{F}}$ holds by definition. Since the localization set $\tilde{\mathcal{F}}$ is given by a finite number of convex inequalities, finding a feasible point in $\tilde{\mathcal{F}}$ or showing infeasibility of $\tilde{\mathcal{F}}$ can be done by solving a convex program efficiently. If for a sample set $\mathcal{S}$, the localization set $\tilde{\mathcal{F}}$ is empty, then we have a certificate that the target set $\mathcal{F}$ is empty. When the target set is nonempty, we can find a feasible point in $\mathcal{F}$ by iteratively expanding the sample set $\mathcal{S}$ and refining the localization set $\tilde{\mathcal{F}}$. However, expanding the sample set in an arbitrary or random way is

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

not efficient or may not improve the over-approximation of $\mathcal{F}$; in order to obtain performance guarantees, we need to add samples that are informative at each iteration. This is done by alternating between a learner and a verifier to select the samples.

B. Construction of the learner

Given a sample set $\mathcal{S}$ and the corresponding localization set $\tilde{\mathcal{F}}$, the learner proposes a Lyapunov function candidate $V_k(x; P_{ac})$ with P_{ac} as the analytic center of $\tilde{\mathcal{F}}$.

Definition 4 (Analytic center [18])**.** *The analytic center x_{ac} of a set of convex inequalities and linear equalities $h_i(x) \leq 0, i = 1, \cdots, m,\ Fx = g$, is defined as the solution of the convex problem*

$$\underset{x}{\text{minimize}} \quad -\sum_{i=1}^{m} \log(-h_i(x)) \quad \textit{subject to} \quad Fx = g.$$

By the definition of analytic center, P_{ac} is given by

$$\begin{aligned} P_{ac} := \underset{P}{\text{argmin}} \ & -\sum_{(x,w)\in\mathcal{S}} \log(-\Delta V_k(x, w, P)) \\ & - \log\det(I - P) - \log\det(P). \end{aligned} \tag{10}$$

where $-\log\det(I-P)-\log\det(P)$ is the barrier function of the set $\{P | 0 \prec P \prec I\}$. If problem (10) is infeasible, the localization set $\tilde{\mathcal{F}}$ is empty and so is the target set $\mathcal{F}$. If problem (10) is feasible, $V_k(x; P_{ac})$ is proposed by the learner as the Lyapunov function candidate based on the localization set $\tilde{\mathcal{F}}$ and is passed to the verifier.

C. Construction of the verifier

For the given Lyapunov function candidate $V_k(x; P_{ac})$ proposed by the learner, the verifier either validates $V_k(x; P_{ac})$ or rejects it by giving a counterexample (x^*, w^*) at which $V_k(x; P_{ac})$ violates the Lyapunov difference condition (6b), i.e., $\Delta V_k(x^*, w^*, P_{ac}) \geq 0$. This is done by solving the global optimization problem:

$$p^* = \underset{x\in\mathcal{X}\setminus B_r, w\in\mathcal{W}(x)}{\text{maximize}} \quad \Delta V_k(x, w, P_{ac}) \tag{11}$$

which is nonconvex. Denote p^* the optimal value of (11) and (x^*, w^*) the optimal solution. The verifier validates $V_k(x; P_{ac})$ if $p^* < 0$ and rejects $V_k(x; P_{ac})$ if $p^* \geq 0$ with (x^*, w^*) as the counterexample. In the former case, we already have a valid Lyapunov function $V_k(x; P_{ac})$ for the uncertain system (5); in the latter case, the counterexample (x^*, w^*) is added to the sample set $\mathcal{S}$ and the localization set $\tilde{\mathcal{F}}$ is updated. Then the learner generates a new Lyapunov function candidate based on the updated localization set. The alternation between the learner and the verifier is repeated until either a Lyapunov function in $\mathcal{F}$ is found or $\mathcal{F}$ is certified to be empty. We summarize our method in Algorithm 1. Solving a nonconvex optimization problem to global optimality is in general intractable, but for the uncertain neural network dynamical system (5), we are able to formulate problem (11) as a nonconvex mixed-integer quadratic program (MIQP) for which global optimization algorithms exists [19, 20]. In practice, Algorithm 1 is easy to implement since solving nonconvex MIQPs to global optimality can be done in off-the-shell solvers such as Gurobi v9.0 [21]. Interested readers are referred to [9] for more discussions on the complexity and solvability of MIQPs. Next, we show how to formulate problem (11) as an MIQP by exploiting the mixed-integer linear formulation of ReLU networks.

Algorithm 1: Learning robust Lyapunov function

Data: Initial sample set $\mathcal{S}$
Result: Infeasibility certificate or Lyapunov function
while *True* **do**
  Update localization set $\tilde{\mathcal{F}}$
  Call the learner
  if (10) *is feasible* **then**
    Propose candidate $V_k(x; P_{ac})$
  else
    Return infeasibility certificate
  Call the verifier
  if *max* (11) < 0 **then**
    Return Lyapunov function
  else
    Generate counterexample (x^*, w^*)
    Update sample set $\mathcal{S} = \mathcal{S} \cup \{(x^*, w^*)\}$

The formulation of the MIQP is based on the fact that the feedforward ReLU network $f_{NN}(x)$ can be described by a set of mixed-integer linear (MIL) constraints. For the $(\ell+1)$-th activation layer in the ReLU network described in (2), let $\underline{m}^\ell$ and $\bar{m}^\ell$ be the element-wise lower and upper bounds on the input, i.e., $\underline{m}_\ell \leq W_\ell z_\ell + b_\ell \leq \bar{m}_\ell$. Then the ReLU activation function is equivalent to the following set of mixed-integer linear constraints [22]:

$$\begin{aligned} & z_{\ell+1} = \max(W_\ell z_\ell + b_\ell, 0) \iff \\ & \begin{cases} z_{\ell+1} \geq W_\ell z_\ell + b_\ell \\ z_{\ell+1} \leq W_\ell z_\ell + b_\ell - \text{diag}(\underline{m}_\ell)(\mathbf{1} - t_\ell) \\ z_{\ell+1} \geq 0 \\ z_{\ell+1} \leq \text{diag}(\bar{m}_\ell) t_\ell, \end{cases} \end{aligned} \tag{12}$$

where $t_\ell \in \{0, 1\}^{n_{\ell+1}}$ is a vector of binary variables for the $(\ell+1)$-th activation layer. Since the input to the ReLU network is bounded in our considered problem, various methods are available to find the element-wise pre-activation bounds $\{\underline{m}_\ell, \bar{m}_\ell\}$ such as interval bound propagation [23] and linear programming [24]. By assembling the mixed-integer linear constraints (12) for each layer, we obtain the mixed-integer linear representation of the ReLU network and also the nominal dynamics $x_+ = \hat{f}(x)$ in (5). Consequently, the composition $\hat{f}^{(i)}(x)$ for $i = 1, \cdots, k$ can be described by mixed-integer linear constraints as well.

With the given Lyapunov function candidate $V_k(x; P_{ac})$, problem (11) can be rewritten as

$$\underset{x,y,w}{\text{maximize}} \quad z_k^\top(y) P_{ac} z_k^\top(y) - z_k^\top(x) P_{ac} z_k^\top(x) \tag{13a}$$

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

$$\text{subject to} \quad y = Ax + f_{NN}(x) + w \tag{13b}$$
$$\|w\|_\infty \le \gamma \|x\|_\infty + \delta \tag{13c}$$
$$x \in \mathcal{X} \setminus B_r \tag{13d}$$

where the MIL representation of constraint (13b) and basis $z_k(x), z_k(y)$ follows from that of the ReLU network, and the MIL representation of constraints (13c) and (13d) follow from that of the ℓ_∞ norm. Since the objective function (13a) is quadratic and indefinite, problem (13) is a nonconvex MIQP which can be solved by Gurobi to global optimality.

D. Termination guarantee

Algorithm 1 is guaranteed to find a Lyapunov function in the target set under the following assumption:

Assumption 2. *The target set $\mathcal{F}$ defined in* (9) *is full-dimensional, i.e., there exists $P_c \in \mathbb{S}^{(k+1)n_x}$ and $\epsilon > 0$ such that $\{P \in \mathbb{S}^{(k+1)n_x} | \|P - P_c\|_F \le \epsilon\} \subset \mathcal{F}$ where $\|\cdot\|_F$ is the Frobenius norm.*

Theorem 2. *Under Assumption 2, Algorithm 1 finds a Lyapunov function in the target set $\mathcal{F}$ in at most $O(((k+1)n_x)^3/\epsilon^2)$ iterations.*

Theorem 2 states that when the target set is full-dimensional, Algorithm 1 is guaranteed to find a Lyapunov function in a finite number of steps. Such a guarantee follows from the termination guarantee of the analytic center cutting-plane method considered in [25] and this is the main reason we require the learner to propose the analytic center of the localization sets as Lyapunov function candidates. From an optimization point of view, the sampling-based method described in Algorithm 1 is equivalent to finding a feasible solution in the target set through the analytic center cutting-plane method with the verifier serving as the cutting-plane oracle. The proof of Theorem 2 directly follows from that of [9, Theorem 2] since our extension of the Lyapunov function synthesis method in [9] to the uncertain dynamical system (5) does not change the convexity of the target set.

Remark 1. *When the target set is empty, we do not have a finite-step termination guarantee for the proposed algorithm. Therefore, Algorithm 1 is semi-complete and in practice, we can add additional stopping criteria, e.g., maximum number of iterations, to Algorithm 1.*

To summarize, the sampling-based Lyapunov function synthesis method in [9] looks for Lyapunov functions simply based on the mixed-integer representation of hybrid systems. This makes it suitable to verify the stability of neural network dynamical systems since modeling the behavior of ReLU network over a compact set through MIL constraints is straightforward. In this paper we extend this method to handle uncertain dynamical systems and apply it in a novel setting to estimate the ROA of nonlinear dynamical systems through their neural network approximations. The non-conservativeness and the finite-step termination guarantee provided by Theorem 1 makes this method favorable to analyze uncertain dynamical systems. However, this is obtained at the cost of the worst-case exponential complexity of the MIQP (13).

V. Approximation of dynamical systems by neural networks

In this section, we investigate the applicability of Assumption 1 and show how to estimate the error bound parameters (γ, δ).

A. Neural network training

The decomposition of the nonlinear map $f(x) = Ax + f_{NN}(x) + w(x)$ in (3) is motivated by two facts: first, stability verification tools [9, 10] for neural network dynamical systems have been developed; second, powerful and easy-to-use machine learning tools are available to train $f_{NN}(x)$ to high accuracy and keep the approximation error $w(x)$ small. In training $f_{NN}(x)$, we want to obtain a neural network model that not only achieves a small approximation error but also is easy to verify. In light of the latter objective, it is desirable to increase the number of *stable* ReLUs in $f_{NN}(x)$, i.e., ReLUs that always output zero or one. For these ReLUs, we do not need to introduce binary variables in the MIQP (13). As a result, any training method that can increase the number of stable ReLUs can significantly improve the complexity of the verification. As shown in [26], applying ℓ_1-regularization to the training objective generally works well in reducing the complexity of verification. Other training methods with the same goal are presented in [26] as well.

B. Identification of error bound

For a given neural network $f_{NN}(x)$ which approximates the nonlinear map $f(x)$, we estimate an error bound in the form of (4) based on dense sampling of the ROI $\mathcal{X}$. This method is not meant to compete with the modeling error quantification methods in the literature. Instead, we use it to provide a reasonable error bound to set up the simulation.

By assumption $f(x)$ is locally Lipschitz continuous. Let L_f (L_N) be an upper bound on the Lipschitz constant of $f(x)$ ($Ax + f_{NN}(x)$) over the ROI $\mathcal{X}$. Since $w(x) = f(x) - Ax - f_{NN}(x)$, we can upper bound the Lipschitz constant of $w(x)$ as $L_w = L_f + L_N$. Then we sample densely from the ROI $\mathcal{X}$ and in particular the samples $\mathcal{S}_\mathcal{X} = \{x_i^S\}_{i=0}^N \subset \mathcal{X}$ form a ϵ-net of $\mathcal{X}$, i.e., for all $x \in \mathcal{X}$, there exists a sample $y \in \mathcal{S}_\mathcal{X}$ such that $\|x - y\| \le \epsilon$. We define the admissible set of (γ, δ) as

$$\Gamma = \{(\gamma, \delta) | \|w(x)\|_\infty \le \gamma \|x\|_\infty + \delta, \forall x \in \mathcal{S}_\mathcal{X}\}$$

and any pair $(\gamma', \delta') \in \Gamma$ gives an estimate of the upper bound in (4). Since we want to keep (γ', δ') small, we let $\delta' = \min \delta$ s.t. $(\gamma', \delta) \in \Gamma$. Note that there are infinitely many pairs in the admissible set and decreasing one parameter would increase the other one.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

Given any admissible pair (γ', δ'), for all $x \in \mathcal{X}$ we have

$$\begin{aligned}\|w(x)\|_\infty &\le \|w(x) - w(y)\|_\infty + \|w(y)\|_\infty \\ &\le L_w\|x - y\|_\infty + \gamma'\|y\|_\infty + \delta' \\ &\le L_w\|x - y\|_\infty + \gamma'\|y - x\|_\infty + \gamma'\|x\|_\infty + \delta' \\ &\le \gamma'\|x\|_\infty + \delta' + (L_w + \gamma')\epsilon\end{aligned} \tag{14}$$

where $y \in \mathcal{S}_\mathcal{X}$ is chosen such that $\|x - y\| \le \epsilon$. Therefore, $\gamma = \gamma', \delta = \delta' + (L_w + \gamma')\epsilon$ gives a valid upper bound for $\|w(x)\|_\infty$ over the ROI $\mathcal{X}$ if we assume L_f is known. Otherwise, inequality (14) indicates that by choosing ϵ small enough, the admissible pair (γ', δ') obtained from sampling gives a good estimate of a deterministic error bound.

Since every pair from the admissible set Γ generates an error bound estimate, we can combine K admissible (γ', δ') pairs to obtain a tighter error bound as

$$\|w\|_\infty \le \gamma'_i\|x\|_\infty + \delta'_i, \quad i = 1, \cdots, K \tag{15}$$

from the sampling. One special, interesting choice is setting $\gamma' = 0$ since in this case constraint (13c) in the MIQP becomes $\|w\|_\infty \le \delta'$ and is convex.

VI. Numerical Examples

We consider a 2-dimensional and 3-dimensional nonlinear system from the literature and train ReLU networks to approximate their dynamics. The approximation errors are estimated from sampling to generate the uncertain neural network dynamical systems, for which our proposed algorithm is called to synthesize robust Lyapunov functions. All experiments are run on an Intel Core i7-6700K CPU.

A. 2-D rational system

Consider the 2 dimensional discrete-time rational system adapted from [27, Example 2]

$$x_1^+ = x_1 - \frac{x_1 + x_2^3}{1 + x_1^2}, \quad x_2^+ = x_2 + \frac{x_1^3 - 0.25x_2}{1 + x_2^2}. \tag{16}$$

and denote the dynamics in (16) as $x_+ = f(x)$. For the uncertain dynamics decomposition (5), we choose the Jacobian of $f(x)$ at the origin as matrix $A = [0\ 0; 0\ 0.75]$. Then we sample 22500 states uniformly from the box $\mathcal{R} = \{x \in \mathbb{R}^2 | \|x\|_\infty \le 1.5\}$ and train a fully-connected ReLU network with the structure $2 - 100 - 100 - 100 - 2$ (three hidden layers with 100 neurons each) with Keras [28] and the Adam optimizer [29] in 100 epochs to approximate the error dynamics $f(x) - Ax$. Since the approximation error bound (4) is written in terms of the ℓ_∞ norm, the training loss function is set to the mean absolute error. Furthermore, in order to alleviate the practical computational cost of the MIQP (13) in the analysis phase, we add ℓ_1 regularization with weight 10^{-4} during the training.

We construct the ROI $\mathcal{X}$ by uniformly sampling initial states from the region $\mathcal{R}$ and simulate the trajectories of the system (16) for 50 steps. Then, the convex hull of the initial condition samples from which the trajectories converge to the origin is chosen as the reference ROI $\mathcal{X}_{\text{ref}} := \{x | A_{\text{ref}} x \le b_{\text{ref}}\}$. We tune the ROI by $\mathcal{X} = \tau\mathcal{X}_{\text{ref}} := \{x | A_{\text{ref}} \le \tau b_{\text{ref}}\}$ with $0 < \tau \le 1.0$. In this example, we set the scaling parameter $\tau = 0.9$. The sampled convergent initial states are marked as the black dots in Fig. 2 where the ROI is also plotted.

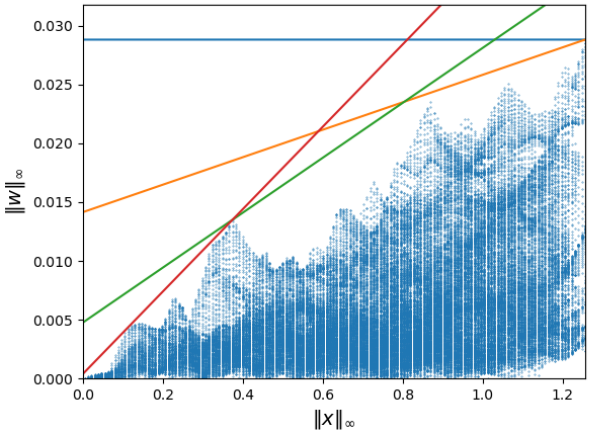


Figure 1: The neural network approximation errors are marked by the blue dots for a dense sample set over the ROI. The affine error bounds are denoted by the blue, orange, green, and red lines corresponding to $(\gamma, \delta) = (0, 0.0288)$, $(1.167, 0.0141)$, $(0.0233, 4.767 \times 10^{-3})$, $(0.0350, 4.026 \times 10^{-4})$, respectively.

After training the neural network and fixing the ROI, we estimate the neural network approximation error by sampling a uniformly spaced grid from the ROI. The grid consists of 72306 state samples which form an ϵ-net of the ROI with $\epsilon \approx 0.005$. We plot the $(\|x\|_\infty, \|w\|_\infty)$ pairs for all the samples in Fig. 1 and our estimated error bound with (γ, δ) gives an affine upper bound over all the sampled graph of $(\|x\|_\infty, \|w\|_\infty)$. Selected values of (γ, δ) and their corresponding affine bounds are plotted in Fig. 1. Although each pair (γ, δ) gives an estimate on the bound of the neural network approximation error, formulating a concave piecewise affine upper bound by combining different values of (γ, δ) is tighter than using a single one as discussed in Section V. As shown in (14), if the sampling is dense enough, the estimated error bounds from samples is close to the true bounds. In this section, we apply estimated error bounds from sampling for simplicity.

Finally, we run Algorithm 1 to find a robust Lyapunov function for the considered neural network approximation system with the error bound given by the concave bound shown in Fig. 1 using four pairs of (γ, δ) values. Let B denote the neighborhood that is excluded from the search space in place of the ℓ_∞ norm ball B_r in (6). For this example, we choose $B = \{x \in \mathbb{R}^2 | -0.05 \le x_1 \le 0.05, -0.2 \le x_2 \le 0.2\}$ as a rectangle according to the simulated uncertain dynamical system (5). Then we set the order of the Lyapunov function candidates to be $k = 1$ in (7) and initialize Algorithm 1 with an empty sample set. Algorithm 1 terminates in 6 iterations with a total

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

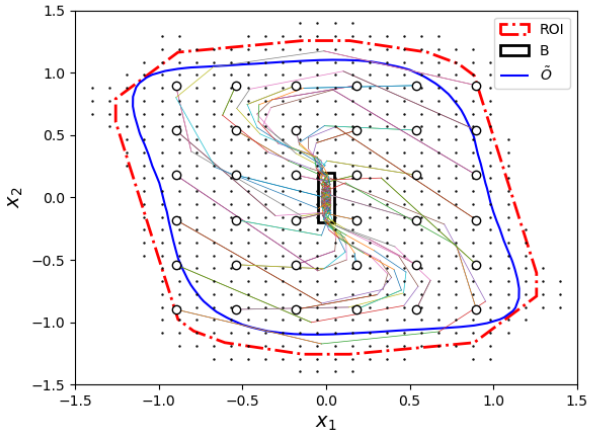


Figure 2: Estimate ROA of system (16) through neural network approximation and synthesizing robust Lyapunov function. The estimate ROA $\tilde{O}$ is marked by the blue curve and the ROI is given by the red polytope with dashed lines. The set B is a rectangle around the origin which is excluded from the search space of the verifier.

solver time of 748 seconds. It successfully finds a robust Lyapunov function $V_1(x;P)$ with $P \in \mathbb{S}^4_{++}$. The resulting estimate of the ROA $\tilde{\mathcal{O}}$ which is the largest sublevel set of $V_1(x;P)$ is plotted in Fig. 2 together with several simulated trajectories of the uncertain system (5). We observe that the estimate ROA provided by $V_1(x;P)$ is close to the true ROA estimated by the samples.

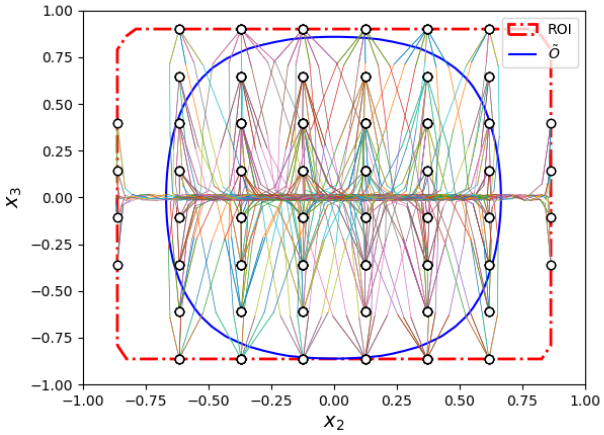
As a comparison, we run Algorithm 1 with error bound $(\gamma,\delta) = (0, 0.0288)$ which corresponds to the blue line in Fig 1. Since $\gamma = 0$, the constraint (13c) becomes convex at the cost of looser characterization of the approximation errors. With the same setup, Algorithm 1 finds a robust Lyapunov function in 17 iterations with the total solver time 3162 seconds. The synthesized robust Lyapunov function generates an estimate ROA similar to that shown in Fig 2 with about 4 times the running time.

B. 3-D polynomial system

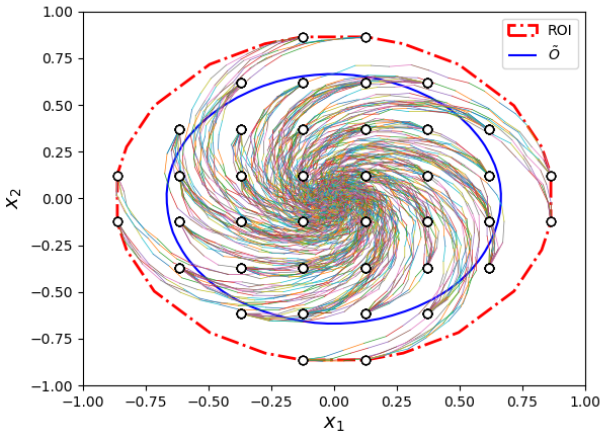
Consider the 3-dimensional continuous-time polynomial system in [30, Example B]

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{bmatrix} = \begin{bmatrix}x_1(x_1^2+x_2^2-1)-x_2(x_3^2+1)\\ x_2(x_1^2+x_2^2-1)+x_1(x_3^2+1)\\ 10x_3(x_3^2-1)\end{bmatrix}. \tag{17}$$

We apply the Euler discretization method with sampling time $dt = 0.1$ seconds to discretize the dynamics and choose the Jacobian of the discretized dynamics at the origin as the A matrix. Then we apply a $3-100-100-100-3$ fully-connected ReLU network to approximate the discrete-time dynamics over the box $\mathcal{R} = \{x \in \mathbb{R}^3 | \|x\|_\infty \le 1\}$. The neural network is trained with 125000 samples uniformly spaced in $\mathcal{R}$ by the same training method described in the 2-dimensional example. The ROI is obtained similarly through simulation and we estimate the neural network approximation error bound from sampling. The samples form an ϵ-net in the ROI with $\epsilon \approx 0.01$ and the error bound $(\gamma,\delta) = (0, 0.0165)$ is estimated from sampling.



(a) Slice at $x_1 = 0$



(b) Slice at $x_3 = 0$

Figure 3: Estimated ROA $\tilde{\mathcal{O}}$ (blue curve) given by the robust Lyapunov function and the ROI (red polytope with dashed lines) on slices $x_1 = 0$ and $x_3 = 0$. Simulated trajectories of the uncertain neural network dynamical system and their initial states (white dots) are projected to the corresponding slices for plotting.

We run Algorithm 1 for the uncertain neural network dynamical system with the exclusion of $B = \{x \in \mathbb{R}^3 | \|x\|_\infty \le 0.05\}$ from the ROI. The sample set is initialized as an empty set. With order $k = 2$, the algorithm terminates in one iteration with a total solver time of 10 seconds. Upon termination, a robust Lyapunov function $V_2(x;P)$ is found whose largest sublevel set is shown in Fig 3 together with the ROI at slices $x_1 = 0$ and $x_3 = 0$. Simulated trajectories of the uncertain neural network dynamical system starting from uniformly spaced initial conditions inside the ROI are plotted in Fig. 3 to illustrate the system dynamics.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.

VII. Conclusion

Finding Lyapunov functions for nonlinear dynamical systems and estimating their region of attraction relies on deep insights from experts and requires a case-by-case analysis. In this paper, we move a step towards automating Lyapunov function synthesis for nonlinear systems. Our method starts with approximating the map of a dynamical system by a piecewise linear neural network. Assuming that the resulting approximation error can be bounded, we then develop a counterexample-guided method to synthesize a Lyapunov function that is robust to the approximation error.

While in practice the approximation error can be made arbitrarily small, computing deterministic bounds on it can be challenging, especially in high dimensions. In future work, we will explore efficient ways to bound the approximation error.

VIII. Acknowledgment

This work was partially supported by the AFOSR grant FA9550-19-1-0265 (Assured Autonomy in Contested Environments).

References

[1] W. M. Haddad and V. Chellaboina, *Nonlinear dynamical systems and control: a Lyapunov-based approach*. Princeton university press, 2011.

[2] P. Giesl and S. Hafstein, "Review on computational methods for Lyapunov functions," *Discrete & Continuous Dynamical Systems-B*, vol. 20, no. 8, p. 2291, 2015.

[3] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 4661–4666.

[4] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 585–590.

[5] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Conference on Robot Learning*. PMLR, 2018, pp. 466–476.

[6] P. Giesl, B. Hamzi, M. Rasmussen, and K. N. Webster, "Approximation of Lyapunov functions from noisy data," *arXiv preprint arXiv:1601.01568*, 2016.

[7] N. M. Boffi, S. Tu, N. Matni, J.-J. E. Slotine, and V. Sindhwani, "Learning stability certificates from data," *arXiv preprint arXiv:2008.05952*, 2020.

[8] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, "Nonlinear systems identification using deep dynamic neural networks," *arXiv preprint arXiv:1610.01439*, 2016.

[9] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for hybrid systems," in *Proceedings of the 24nd ACM International Conference on Hybrid Systems: Computation and Control*, 2021.

[10] H. Yin, P. Seiler, and M. Arcak, "Stability analysis using quadratic constraints for systems with neural network controllers," *arXiv preprint arXiv:2006.07579*, 2020.

[11] D. Ahmed, A. Peruffo, and A. Abate, "Automated and sound synthesis of Lyapunov functions with smt solvers," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2020, pp. 97–114.

[12] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of Lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.

[13] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov control," *arXiv preprint arXiv:2005.00611*, 2020.

[14] A. A. Ahmadi and P. A. Parrilo, "Non-monotonic Lyapunov functions for stability of discrete time nonlinear and switched systems," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 614–621.

[15] R. Bobiti and M. Lazar, "A sampling approach to finding Lyapunov functions for nonlinear discrete-time systems," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 561–566.

[16] Y. Nesterov, "Cutting plane algorithms from analytic centers: efficiency estimates," *Mathematical Programming*, vol. 69, no. 1, pp. 149–176, 1995.

[17] D. S. Atkinson and P. M. Vaidya, "A cutting plane algorithm for convex programming that uses analytic centers," *Mathematical Programming*, vol. 69, no. 1-3, pp. 1–43, 1995.

[18] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[19] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, p. 1, 2013.

[20] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter, "Branching and bounds tightening techniques for non-convex minlp," *Optimization Methods & Software*, vol. 24, no. 4-5, pp. 597–634, 2009.

[21] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[22] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.

[23] C.-H. Cheng, G. Nührenberg, and H. Ruess, "Maximum resilience of artificial neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2017, pp. 251–268.

[24] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International Conference on Machine Learning*, 2018, pp. 5286–5295.

[25] J. Sun, K.-C. Toh, and G. Zhao, "An analytic center cutting plane method for semidefinite feasibility problems," *Mathematics of Operations Research*, vol. 27, no. 2, pp. 332–346, 2002.

[26] K. Y. Xiao, V. Tjeng, N. M. Shafiullah, and A. Madry, "Training for faster adversarial robustness verification via inducing relu stability," *arXiv preprint arXiv:1809.03008*, 2018.

[27] D. Coutinho and C. E. de Souza, "Local stability analysis and domain of attraction estimation for a class of uncertain nonlinear discrete-time systems," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 13, pp. 1456–1471, 2013.

[28] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] A. I. Doban and M. Lazar, "Computation of Lyapunov functions for nonlinear differential equations via a massera-type construction," *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1259–1272, 2017.

Authorized licensed use limited to: University of Pennsylvania. Downloaded on October 29,2026 at 15:03:05 UTC from IEEE Xplore. Restrictions apply.