The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems

Spencer M. Richards
Department of Mechanical and Process Engineering
ETH Zürich
spenrich@stanford.edu

Felix Berkenkamp
Department of Computer Science
ETH Zürich
befelix@inf.ethz.ch

Andreas Krause
Department of Computer Science
ETH Zürich
krausea@ethz.ch

Abstract: Learning algorithms have shown considerable prowess in simulation by allowing robots to adapt to uncertain environments and improve their performance. However, such algorithms are rarely used in practice on safety-critical systems, since the learned policy typically does not yield any safety guarantees. That is, the required exploration may cause physical harm to the robot or its environment. In this paper, we present a method to learn accurate safety certificates for nonlinear, closed-loop dynamical systems. Specifically, we construct a neural network Lyapunov function and a training algorithm that adapts it to the shape of the largest safe region in the state space. The algorithm relies only on knowledge of inputs and outputs of the dynamics, rather than on any specific model structure. We demonstrate our method by learning the safe region of attraction for a simulated inverted pendulum. Furthermore, we discuss how our method can be used in safe learning algorithms together with statistical models of dynamical systems.

Keywords: Lyapunov stability, Safe learning, Reinforcement learning

1 Introduction

Safety is among the foremost open problems in robotics and artificial intelligence [1]. Many autonomous systems, such as self-driving cars and robots for palliative care, are safety-critical due to their interaction with human life. At the same time, learning is necessary for these systems to perform well in *a priori* unknown environments. During learning, they must *safely explore* their environment by avoiding dangerous states from which they cannot recover. For example, consider an autonomous robot in an outdoor environment affected by rough terrain and adverse weather conditions. These factors introduce uncertainty about the relationship between the robot's speed and maneuverability. While the robot should learn about its capabilities in such conditions, it must not perform a maneuver at a high speed that would cause it to crash. Conversely, traveling at only slow speeds to avoid accidents is not conducive to learning about the extent of the robot's capabilities.

To ensure *safe learning*, we must verify a *safety certificate* for a state before it is explored. In control theory, a set of states is safe if system trajectories are bounded within it and asymptotically converge to a fixed point under a fixed control policy. Within such a *region of attraction (ROA)* [2], the system can collect data during learning and can always recover to a known safe point. In this paper, we leverage Lyapunov stability theory to construct provable, neural network-based safety certificates, and *adapt* them to the size and shape of the largest ROA of a general nonlinear dynamical system.

Related work Lyapunov functions are convenient tools for stability (i.e., safety) certification of dynamical systems [2] and for ROA estimation [3, 4, 5]. These functions encode long-term behaviour of state trajectories in a scalar value [6], such that a ROA can be encoded as a level set of the Lyapunov function. However, Lyapunov functions for general dynamical systems are difficult to find;

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.

computational approaches are surveyed in [7]. A Lyapunov function can be identified efficiently via a semi-definite program (SDP, [8]) when the dynamics are polynomial and the Lyapunov function is restricted to be a sum-of-squares (SOS) polynomial [9]. Other methods to compute ROAs include maximization of a measure of ROA volume over system trajectories [10], and sampling-based approaches that generalize information about stability at discrete points to a continuous region [11].

This paper is particularly concerned with safety certificates for dynamical systems with uncertainties in the form of *model errors*. In robust control [12], the formulation of SDPs with SOS Lyapunov functions is used to compute ROA estimates for uncertain linear dynamical systems with the assumption of a worst-case linear perturbation from a known bounded set [13, 14]. Learning-based control methods with a Gaussian process (GP, [15]) model of the system instead consider uncertainty in a Bayesian manner, where model errors are reduced in regions where data has been collected. The methods in [16, 17] estimate a ROA with Lyapunov stability certificates computed on a discretization of the state space, which is used for safe reinforcement learning (RL, [18]). The Lyapunov function is assumed to be given in [16], while [17] uses the negative value (i.e., cost) function from RL with a quadratic reward. Ultimately, this approach is limited by a *shape mismatch* between level sets of the Lyapunov function and the true largest ROA. For example, a quadratic Lyapunov function has ellipsoidal level sets, which cannot characterize a non-ellipsoidal ROA, while the SOS approach is restricted to fixed monomial features. To improve safe exploration for general nonlinear dynamics, we want to *learn* these features to determine a Lyapunov function with suitably shaped level sets.

Contributions In this paper, we present a novel method for learning accurate safety certificates for general nonlinear dynamical systems. We construct a neural network Lyapunov candidate and, unlike past work in [19, 20], we structure our candidate such that it *always* inherently yields a provable safety certificate. Then, we specify a training algorithm that adapts the candidate to the shape of the dynamical system's trajectories via classification of states as safe or unsafe. We do not depend on any specific structure of the dynamics for this. We show how our construction relates to SOS Lyapunov functions, and compare our approach to others on a simulated inverted pendulum benchmark. We also discuss how our method can be used to make safe learning more effective.

2 Problem Statement and Background

We consider a discrete-time, time-invariant, deterministic dynamical system of the form

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t), \tag{1}$$

where $t \in \mathbb{N}$ is the time step index, and $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^d$ and $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^p$ are the state and control inputs respectively at time step t. The system is controlled by a feedback policy $\pi \colon \mathcal{X} \to \mathcal{U}$ and the resulting closed-loop dynamical system is given by $\mathbf{x}_{t+1} = f_\pi(\mathbf{x}_t)$ with $f_\pi(\mathbf{x}) = f(\mathbf{x}, \pi(\mathbf{x}))$. We assume this policy is given, but it can, for example, be computed online with RL or optimal control. This policy π is safe to use within a subset $\mathcal{S}_\pi$ of the state space $\mathcal{X}$. The set $\mathcal{S}_\pi$ is a ROA for f_π, i.e., every system trajectory of f_π that begins at some $\mathbf{x} \in \mathcal{S}_\pi$ also remains in $\mathcal{S}_\pi$ and asymptotically approaches an *equilibrium point* $\mathbf{x}_\mathrm{O} \in \mathcal{S}_\pi$ where $f_\pi(\mathbf{x}_\mathrm{O}) = \mathbf{x}_\mathrm{O}$ [2]. We assume $\mathbf{x}_\mathrm{O} = \mathbf{0}$ without loss of generality. Hereafter, we use $\mathcal{S}_\pi$ to denote the true largest ROA in $\mathcal{X}$ under the policy π.

A reliable estimate of $\mathcal{S}_\pi$ is critical to online learning systems, since we need to ensure that a policy is safe to use on the real system before it can be deployed. The goal of this paper is to estimate the largest safe set $\mathcal{S}_\pi$. We must also ensure safety by never overestimating $\mathcal{S}_\pi$, i.e., we must not identify unsafe states as safe. For this to be feasible, we make a regularity assumption about the closed-loop dynamics; we assume f_π is Lipschitz continuous on $\mathcal{X}$ with Lipschitz constant $L_{f_\pi} \in \mathbb{R}_{>0}$. This is a weak assumption and is even satisfied when a neural network policy is used [21].

2.1 Safety Certification with Lyapunov Functions

One way to estimate the safe region $\mathcal{S}_\pi$ is by using a Lyapunov function. Given a suitable Lyapunov function v, a safe region for the closed-loop dynamical system $\mathbf{x}_{t+1} = f_\pi(\mathbf{x}_t)$ can be determined.

Theorem 1 (Lyapunov's stability theorem [6]): *Suppose f_π is locally Lipschitz continuous and has an equilibrium point at $\mathbf{x}_\mathrm{O} = \mathbf{0}$. Let $v : \mathcal{X} \to \mathbb{R}$ be locally Lipschitz continuous on $\mathcal{X}$. If there exists a set $\mathcal{D}_v \subseteq \mathcal{X}$ containing $\mathbf{0}$ on which v is positive-definite and $\Delta v(\mathbf{x}) \coloneqq v(f_\pi(\mathbf{x})) - v(\mathbf{x}) < 0$, $\forall \mathbf{x} \in \mathcal{D}_v \setminus \{\mathbf{0}\}$, then $\mathbf{x}_\mathrm{O} = \mathbf{0}$ is an asymptotically stable equilibrium. In this case, v is known as a Lyapunov function for the closed-loop dynamics f_π, and $\mathcal{D}_v$ is the Lyapunov decrease region for v.*

(a) Shape mismatch with a fixed Lyapunov function.

(b) Shape match with a parameterized Lyapunov function.

Figure 1. Fig. 1a illustrates a shape mismatch between the largest level set $\mathcal{V}(c)$ (blue ellipsoid) of a quadratic Lyapunov function v contained within the decrease region $\mathcal{D}_v$ (green dashes), and the safe region $\mathcal{S}_\pi$ (black). We cannot certify all of $\mathcal{S}_\pi$ with v, which limits exploration in safe learning. Instead, we train a Lyapunov candidate $v_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ to match $\mathcal{S}_\pi$ with a level set $\mathcal{V}_{\boldsymbol{\theta}}(c_\mathcal{S})$, as in Fig. 1b, via classification of sampled states as "safe" with ground-truth label $y = +1$ (i.e., $\mathbf{x} \in \mathcal{S}_\pi$) or "unsafe" with $y = -1$ (i.e., $\mathbf{x} \notin \mathcal{S}_\pi$).

Theorem 1 states that a Lyapunov function v characterizes a "basin" of safe states where trajectories of f_π "fall" towards the origin $\mathbf{x}_\mathrm{O} = \mathbf{0}$. If we can find a positive-definite v such that the dynamics always map downwards in the value of $v(\mathbf{x})$, then trajectories eventually reach $v(\mathbf{x}) = \mathbf{0}$, thus $\mathbf{x} = \mathbf{0}$. To find a ROA, rather than checking if v decreases along entire trajectories, it is sufficient to verify the *one-step decrease condition* $\Delta v(\mathbf{x}) < 0$ for every state $\mathbf{x}$ in a level set of v.

Corollary 1 (Safe level sets [6]): *Every level set* $\mathcal{V}(c) := \left\{\mathbf{x} \mid v(\mathbf{x}) \leq c\right\}, c \in \mathbb{R}_{>0}$ *contained within the decrease region* $\mathcal{D}_v$ *is invariant under* f_π. *That is,* $f_\pi(\mathbf{x}) \in \mathcal{V}(c), \forall \mathbf{x} \in \mathcal{V}(c)$. *Furthermore,* $\lim_{t\to\infty} \mathbf{x}_t = \mathbf{0}$ *for every* $\mathbf{x}_t$ *in these level sets, so each one is a ROA for* f_π *and* $\mathbf{x}_\mathrm{O} = \mathbf{0}$.

Intuitively, if $v(\mathbf{x})$ decreases everywhere in the level set $\mathcal{V}(c_1)$, except at $\mathbf{x}_\mathrm{O} = \mathbf{0}$ where it is zero, then $\mathcal{V}(c_1)$ is invariant, since the image of $\mathcal{V}(c_1)$ under f_π is the smaller level set $\mathcal{V}(c_2)$ with $c_2 < c_1$. If v is also positive-definite, then this ensures trajectories that start in a level set $\mathcal{V}(c)$ contained in the decrease region $\mathcal{D}_v$ remain in $\mathcal{V}(c)$ and converge to $\mathbf{x}_\mathrm{O} = \mathbf{0}$. To identify safe level sets, we must check if a given *Lyapunov candidate* v satisfies the conditions of Theorem 1. However, the decrease condition $\Delta v(\mathbf{x}) < 0$ is difficult to verify throughout a continuous subset $\mathcal{D}_v \subseteq \mathcal{X}$. It is sufficient to verify the tightened safety certificate $\Delta v(\mathbf{x}) < -L_{\Delta v}\tau$ at a finite set of points that cover $\mathcal{D}_v$, where $L_{\Delta v} \in \mathbb{R}_{>0}$ is the Lipschitz constant of Δv and $\tau \in \mathbb{R}_{>0}$ is a measure of how densely the points cover $\mathcal{D}_v$ [17]. We can even couple this with bounds on f_π from a statistical model to certify high-probability safe sets with the certificate $\Delta \hat{v}(\mathbf{x}) < -L_{\Delta v}\tau$, where $\Delta \hat{v}(\mathbf{x})$ is an upper confidence bound on $\Delta v(\mathbf{x})$. A GP model of f_π is used for this purpose in [17].

2.2 Computing SOS Lyapunov Functions

In general, a suitable Lyapunov candidate v is difficult to find. Computational methods often restrict v to a particular function class for tractability. The SOS approach restricts $v(\mathbf{x})$ to be polynomial, but is limited to polynomial dynamical systems, i.e., when $f_\pi(\mathbf{x})$ is a vector of polynomials in the elements of $\mathbf{x}$ [9, 22, 23]. In particular, the SOS approach enforces $v(\mathbf{x}) = m(\mathbf{x})^\top \mathbf{Q} m(\mathbf{x})$, where $m(\mathbf{x})$ is a vector of *a priori* fixed monomial features in the elements of $\mathbf{x}$, and $\mathbf{Q}$ is an unknown positive-semidefinite matrix. This makes $v(\mathbf{x})$ a quadratic function on a monomial *feature space*. A SDP can be efficiently solved to yield a $\mathbf{Q}$ that *simultaneously* guarantees that v satisfies the assumptions of Theorem 1 and has the largest possible level set in its decrease region $\mathcal{D}_v$. That is, the positive-definiteness of v and the negative-definiteness of Δv in $\mathcal{D}_v$ are enforced as constraints in the SDP. This contrasts the more general approach described in Sec. 2.1, where a Lyapunov candidate v is given and then the assumptions of Theorem 1 are verified by checking discrete points.

With the SOS approach and a suitable choice of $m(\mathbf{x})$, $\mathcal{S}_\pi$ can be estimated well with a level set $\mathcal{V}(c)$ of v, since the monomial features allow Lyapunov functions with shapes beyond simple ellipsoids to be found. However, the SOS approach requires polynomial dynamics, and the best choice of $m(\mathbf{x})$ can be difficult to determine. Without a suitable Lyapunov function, we face the problem of a *shape mismatch* between $\mathcal{V}(c)$ and $\mathcal{S}_\pi$. This is exemplified in Fig. 1a, where level sets of quadratic v are ellipsoidal while $\mathcal{S}_\pi$ is not, which limits the region of the state space that is certifiable as safe by v.

3 Learning Lyapunov Candidates

In this section, we establish a more flexible class of parameterized Lyapunov candidates that can satisfy the assumptions on v in Theorem 1 by virtue of their structure and gradient-based parameter training. In particular, we show how a binary classification problem based on whether each state $\mathbf{x}$ lies within the safe region $\mathcal{S}_\pi$ can be formulated to train the parameterized Lyapunov candidate.

3.1 Construction of a Neural Network Lyapunov Function

We take the SOS approach in Sec. 2.2 as a starting point to construct a neural network Lyapunov candidate. The SOS Lyapunov candidate $v(\mathbf{x}) = m(\mathbf{x})^\top \mathbf{Q} m(\mathbf{x})$ is a Euclidean inner product on the transformed space $\mathcal{Y} \coloneqq \{\phi(\mathbf{x}),\, \forall \mathbf{x} \in \mathcal{X}\}$ with $\phi(\mathbf{x}) \coloneqq \mathbf{Q}^{1/2} m(\mathbf{x})$. The ability of the SOS Lyapunov candidate v to certify safe states for f_π depends on the choice of monomials in $m(\mathbf{x})$. We interpret these choices as engineered features that define the expressiveness of v in delineating the decision boundary between safe and unsafe states. Rather than choose such features manually and parameterize $\phi(\mathbf{x})$ with $\mathbf{Q}$ only, we propose the Lyapunov candidate $v_{\boldsymbol{\theta}}(\mathbf{x}) = \phi_{\boldsymbol{\theta}}(\mathbf{x})^\top \phi_{\boldsymbol{\theta}}(\mathbf{x})$ to *learn* the requisite features, where $\phi_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^D$ is a feed-forward neural network with parameter vector $\boldsymbol{\theta}$. Feed-forward neural networks are expressive in that they can approximate any continuous function on compact subsets of $\mathbb{R}^d$ with a finite number of parameters [24, 25]. In Sec. 3.2, we exploit this property together with gradient-based parameter training to closely match the true ROA $\mathcal{S}_\pi$ with a level set of the candidate $v_{\boldsymbol{\theta}}$ without the need to engineer individual features of ϕ.

We cannot use an arbitrary feed-forward neural network $\phi_{\boldsymbol{\theta}}$ in our Lyapunov candidate, since the conditions of Theorem 1 must be satisfied. Otherwise, the resulting candidate $v_{\boldsymbol{\theta}}$ cannot provide any safety information. In general, $\phi_{\boldsymbol{\theta}}$ is a sequence of function compositions or layers. Each layer has the form $\mathbf{y}_\ell(\mathbf{x}) = \varphi_\ell(\mathbf{W}_\ell \mathbf{y}_{\ell-1}(\mathbf{x}))$, where $\mathbf{y}_\ell(\mathbf{x})$ is the output of layer ℓ for state $\mathbf{x} \in \mathcal{X}$, φ_ℓ is a fixed element-wise activation function, and $\mathbf{W}_\ell \mathbf{y}_{\ell-1}(\mathbf{x})$ is a linear transformation parameterized by $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$. To satisfy the assumptions of Theorem 1, $v_{\boldsymbol{\theta}}$ must be Lipschitz continuous on $\mathcal{X}$ and positive-definite on some subset of $\mathcal{X}$ around $\mathbf{x}_\mathrm{O} = \mathbf{0}$. To this end, we restrict $v_{\boldsymbol{\theta}}$ to be positive-definite and Lipschitz continuous on $\mathcal{X}$ for *all* values of $\boldsymbol{\theta} \coloneqq \{\mathbf{W}_\ell\}_\ell$ with a suitable choice of structure for $\phi_{\boldsymbol{\theta}}$.

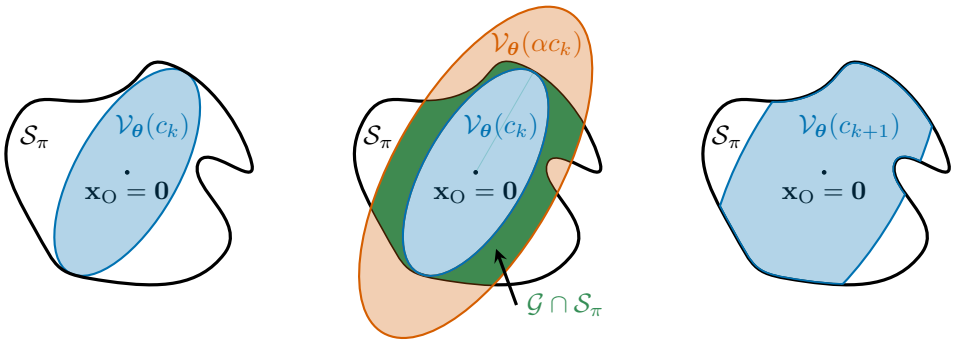
Theorem 2 (Lyapunov neural network): *Consider $v_{\boldsymbol{\theta}}(\mathbf{x}) = \phi_{\boldsymbol{\theta}}(\mathbf{x})^\top \phi_{\boldsymbol{\theta}}(\mathbf{x})$ as a Lyapunov candidate function, where $\phi_{\boldsymbol{\theta}}$ is a feed-forward neural network. Suppose, for each layer ℓ in $\phi_{\boldsymbol{\theta}}$, the activation function φ_ℓ and weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ each have a trivial nullspace. Then $\phi_{\boldsymbol{\theta}}$ has a trivial nullspace, and $v_{\boldsymbol{\theta}}$ is positive-definite with $v_{\boldsymbol{\theta}}(\mathbf{0}) = 0$ and $v_{\boldsymbol{\theta}}(\mathbf{x}) > 0$, $\forall \mathbf{x} \in \mathcal{X} \setminus \{\mathbf{0}\}$. Furthermore, if φ_ℓ is Lipschitz continuous for each layer ℓ, then $v_{\boldsymbol{\theta}}$ is locally Lipschitz continuous.*

We provide a formal proof of Theorem 2 in Appendix A and briefly outline it here. As an inner product, $v_{\boldsymbol{\theta}}(\mathbf{x}) = \phi_{\boldsymbol{\theta}}(\mathbf{x})^\top \phi_{\boldsymbol{\theta}}(\mathbf{x})$ is already positive-definite for any neural network output $\phi_{\boldsymbol{\theta}}(\mathbf{x})$, and thus is *at least* nonnegative for any state $\mathbf{x} \in \mathcal{X}$. The step from nonnegativity to positive-definiteness of $v_{\boldsymbol{\theta}}$ on $\mathcal{X}$ now only depends on how the origin $\mathbf{0} \in \mathcal{X}$ is mapped through $\phi_{\boldsymbol{\theta}}$. If $\phi_{\boldsymbol{\theta}}$ maps $\mathbf{0} \in \mathcal{X}$ *uniquely* to the zero output $\phi_{\boldsymbol{\theta}}(\mathbf{0}) = \mathbf{0}$, i.e., if $\phi_{\boldsymbol{\theta}}$ has a trivial nullspace, then $v_{\boldsymbol{\theta}}$ is positive-definite. For this, it is sufficient that each layer of $\phi_{\boldsymbol{\theta}}$ has a trivial nullspace, i.e., that each layer "passes along" $\mathbf{0} \in \mathcal{X}$ to its zero output $\mathbf{y}_\ell(\mathbf{0}) = \mathbf{0}$ until the final output $\phi_{\boldsymbol{\theta}}(\mathbf{0}) = \mathbf{0}$.

In Theorem 2, each layer ℓ has a trivial nullspace as long as its weight matrix $\mathbf{W}_\ell$ and activation function φ_ℓ have trivial nullspaces. Consequently, this requires that $d_\ell \geq d_{\ell-1}$ for each layer ℓ, where d_ℓ is the output dimension of layer ℓ. That is, $\mathbf{W}_\ell$ must not decrease the dimension of its input. To ensure that $\mathbf{W}_\ell$ has a trivial nullspace, we structure it as

$$\mathbf{W}_\ell = \begin{bmatrix} \mathbf{G}_{\ell 1}^\top \mathbf{G}_{\ell 1} + \varepsilon \mathbf{I}_{d_{\ell-1}} \\ \mathbf{G}_{\ell 2} \end{bmatrix}, \tag{2}$$

where $\mathbf{G}_{\ell 1} \in \mathbb{R}^{q_\ell \times d_{\ell-1}}$ for some $q_\ell \in \mathbb{N}_{\geq 1}$, $\mathbf{G}_{\ell 2} \in \mathbb{R}^{(d_\ell - d_{\ell-1}) \times d_{\ell-1}}$, $\mathbf{I}_{d_{\ell-1}} \in \mathbb{R}^{d_{\ell-1} \times d_{\ell-1}}$ is the identity matrix, and $\varepsilon \in \mathbb{R}_{>0}$ is a constant. The top partition $\mathbf{G}_{\ell 1}^\top \mathbf{G}_{\ell 1} + \varepsilon \mathbf{I}_{d_{\ell-1}}$ is positive-definite for $\varepsilon > 0$, thus $\mathbf{W}_\ell$ always has full rank and a trivial nullspace. Otherwise, $\mathbf{W}_\ell$ would have a non-empty nullspace of dimension $d_{\ell-1} - \min(d_\ell, d_{\ell-1}) = d_{\ell-1} - d_\ell > 0$ by the rank-nullity theorem. With this choice of structure for $\mathbf{W}_\ell$, the parameters of the neural network $\phi_{\boldsymbol{\theta}}$ are given by $\boldsymbol{\theta} \coloneqq \{\mathbf{G}_{\ell 1}, \mathbf{G}_{\ell 2}\}_\ell$. Finally, we choose activation functions that have trivial nullspaces and that are Lipschitz continuous in $\mathcal{X}$, such as $\tanh(\cdot)$ and the leaky ReLU. We can then compute a Lipschitz constant for $\phi_{\boldsymbol{\theta}}$ [21].



(a) Current safe level set. (b) Simulate gap states forward. (c) Re-shape safe level set.

Figure 2. Illustration of training the parameterized Lyapunov candidate $v_{\boldsymbol{\theta}}$ to expand the safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ (blue ellipsoid) towards the true largest ROA $\mathcal{S}_\pi$ (black). States in the gap $\mathcal{G}$ between $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ and $\mathcal{V}_{\boldsymbol{\theta}}(\alpha c_k)$ (orange ellipsoid) are simulated forward to determine regions (green) towards which we can expand the safe level set. This information is used in Algorithm 1 to iteratively adapt safe level sets of $v_{\boldsymbol{\theta}}$ to the shape of $\mathcal{S}_\pi$.

3.2 Learning a Safe Set via Classification

Previously, we constructed a neural network Lyapunov candidate $v_{\boldsymbol{\theta}}$ in Theorem 2 that satisfies the positive-definiteness and Lipschitz continuity requirements in Theorem 1. As a result, we can always use the one-step decrease condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) \coloneqq v_{\boldsymbol{\theta}}(f_\pi(\mathbf{x})) - v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ as a provable safety certificate to identify safe level sets that are subsets of the largest safe region $\mathcal{S}_\pi$. Now, we design a training algorithm to adapt the parameters $\boldsymbol{\theta}$ such that the resulting Lyapunov candidate $v_{\boldsymbol{\theta}}$ satisfies $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ throughout as large of a decrease region $\mathcal{D}_{v_{\boldsymbol{\theta}}} \subseteq \mathcal{X}$ as possible. This also makes $v_{\boldsymbol{\theta}}$ a valid Lyapunov function for the closed-loop dynamics f_π.

For now, we assume the entire safe region $\mathcal{S}_\pi$ is known. We want to use a level set $\mathcal{V}_{\boldsymbol{\theta}}(c)$ of $v_{\boldsymbol{\theta}}$ to certify the entire set $\mathcal{S}_\pi$ as safe. According to Theorem 1, this requires the Lyapunov decrease condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ to be satisfied for each state $\mathbf{x} \in \mathcal{S}_\pi$. We formally state this problem as

$$\max_{\boldsymbol{\theta}, c} \mathrm{Vol}\big(\mathcal{V}_{\boldsymbol{\theta}}(c) \cap \mathcal{S}_\pi\big) \text{ , s.t. } \Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0, \forall \mathbf{x} \in \mathcal{V}_{\boldsymbol{\theta}}(c), \tag{3}$$

where $\mathrm{Vol}(\cdot)$ is some measure of set volume. Thus, we want to find the largest level set of $v_{\boldsymbol{\theta}}$ that is contained in the true largest ROA $\mathcal{S}_\pi$; see Fig. 2a. We fix $c = c_\mathcal{S}$ with some $c_\mathcal{S} \in \mathbb{R}_{>0}$, as it is always possible to rescale $v_{\boldsymbol{\theta}}$ by a constant, and focus on optimizing over $\boldsymbol{\theta}$. We can then interpret (3) as a classification problem. Consider Fig. 1b, where we assign the ground-truth label $y = +1$ whenever a state $\mathbf{x}$ is contained in $\mathcal{S}_\pi$, and $y = -1$ otherwise. We use $v_{\boldsymbol{\theta}}$ together with Theorem 1 to classify states by their membership in the level set $\mathcal{V}(c_\mathcal{S})$. This is described by the decision rule

$$\hat{y}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathrm{sign}\left(c_\mathcal{S} - v_{\boldsymbol{\theta}}(\mathbf{x})\right). \tag{4}$$

That is, each state within the level set $\mathcal{V}(c_\mathcal{S})$ obtains the label $y = +1$. However, we must also satisfy the Lyapunov decrease condition imposed by Theorem 1. This can be written as the constraint

$$y = +1 \implies \Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0, \tag{5}$$

which means that we can assign the label $y = +1$ only if the decrease condition is also satisfied. The decision rule (4) together with the constraint (5) ensures that the resulting estimated safe set $\mathcal{V}(c_\mathcal{S})$ satisfies all of the conditions in Theorem 1. We want to select the neural network parameters $\boldsymbol{\theta}$ so that this rule can perfectly classify $\mathbf{x} \in \mathcal{S}_\pi$ as "safe" with $\hat{y}_{\boldsymbol{\theta}}(\mathbf{x}) = +1$ (i.e., $c_\mathcal{S} - v_{\boldsymbol{\theta}}(\mathbf{x}) > 0$) or $\mathbf{x} \notin \mathcal{S}_\pi$ as "unsafe" with $\hat{y}_{\boldsymbol{\theta}}(\mathbf{x}) = -1$ (i.e., $c_\mathcal{S} - v_{\boldsymbol{\theta}}(\mathbf{x}) \leq 0$). To this end, the decision boundary $v_{\boldsymbol{\theta}}(\mathbf{x}) = c_\mathcal{S}$ must exactly delineate the boundary of $\mathcal{S}_\pi$. Furthermore, the value of $\boldsymbol{\theta}$ must ensure (5) holds, such that $v_{\boldsymbol{\theta}}$ satisfies the decrease condition of Theorem 1 on $\mathcal{S}_\pi$.

Since we have rewritten the optimization problem in (3) as a classification problem, we can use ideas from the corresponding literature [26]. In particular, we define a loss function $\ell(y, \mathbf{x}; \boldsymbol{\theta})$ that penalizes misclassification of the true label y at a state $\mathbf{x}$ under the decision rule (4) associated with $\boldsymbol{\theta}$. Many common choices for the loss function are possible; for simplicity, we use the perceptron loss, which penalizes misclassifications more when they occur far from the decision boundary.

Algorithm 1 ROA Classifier Training

Input: closed-loop dynamics f_π; initialized parametric Lyapunov candidate $v_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}_{\geq 0}$; Lagrange multiplier $\lambda \in \mathbb{R}_{>0}$; level set "expansion" multiplier $\alpha \in \mathbb{R}_{>1}$; forward-simulation horizon $T \in \mathbb{N}_{\geq 1}$.
$c_0 \leftarrow \max_{\mathbf{x} \in \mathcal{X}} v_{\boldsymbol{\theta}}(\mathbf{x}),\ \text{s.t. } \mathcal{V}_{\boldsymbol{\theta}}(c_0) \subseteq \mathcal{D}_{v_{\boldsymbol{\theta}}}$. ▷ compute the initial safe level set (e.g., use a discretization, as described in Sec. 2.1)
repeat
  Sample a finite batch $\mathcal{X}_b \subset \mathcal{V}_{\boldsymbol{\theta}}(\alpha c_k)$.
  $\mathcal{S}_b \leftarrow \left\{\mathbf{x} \in \mathcal{X}_b \mid f_\pi^{(T)}(\mathbf{x}) \in \mathcal{V}_{\boldsymbol{\theta}}(c_k)\right\}$. ▷ forward-simulate the batch with f_π over T steps
  Update $\boldsymbol{\theta}$ with (7) via batch SGD on $\mathcal{X}_b$ and labels $\{y_i\}_i$ for points in $\mathcal{S}_b$.
  $c_{k+1} \leftarrow \max_{\mathbf{x} \in \mathcal{X}} v_{\boldsymbol{\theta}}(\mathbf{x}),\ \text{s.t. } \mathcal{V}_{\boldsymbol{\theta}}(c_{k+1}) \subseteq \mathcal{D}_{v_{\boldsymbol{\theta}}}$.
until convergence

We choose not to use the "maximum margin" objective of the hinge loss, since it may be unsuitable for us to accurately delineate $\mathcal{S}_\pi$, where states can lie arbitrarily close to the decision boundary in the continuous state space $\mathcal{X}$. Since we use the level set $\mathcal{V}_{\boldsymbol{\theta}}(c_{\mathcal{S}})$ in our classification setting, this corresponds to $\ell(y, \mathbf{x}; \boldsymbol{\theta}) = \max\big(0, -y \cdot \big(c_{\mathcal{S}} - v_{\boldsymbol{\theta}}(\mathbf{x})\big)\big)$. Here, $c_{\mathcal{S}} - v_{\boldsymbol{\theta}}(\mathbf{x})$ is the signed distance from the decision boundary $v_{\boldsymbol{\theta}}(\mathbf{x}) = c_{\mathcal{S}}$, which separates the safe set $\mathcal{S}_\pi$ from the rest of the state space $\mathcal{X} \setminus \mathcal{S}_\pi$. This *classifier loss* has a magnitude of $\big|c_{\mathcal{S}} - v_{\boldsymbol{\theta}}(\mathbf{x})\big|$ in the case of a misclassification, and zero otherwise. This ensures that decisions far from the decision boundary, such as those near the origin, are considered more important than the more difficult decisions close to the boundary.

Ideally, we would like to minimize this loss throughout the state space with $\min \int_{\mathcal{X}} l(y, \mathbf{x}; \theta)\, \mathrm{d}\mathbf{x}$ subject to the constraint (5). Since this problem is intractable, we use gradient-based optimization together with mini-batches instead, as is typically done in machine learning. To this end, we sample states $\mathcal{X}_b = \{\mathbf{x}_i\}_i$ from the state space $\mathcal{X}$ at random and assign the ground-truth labels $\{y_i\}_i$ to them. Based on this finite set, the optimization objective can be written as

$$\min_{\boldsymbol{\theta}} \sum_{\mathbf{x} \in \mathcal{X}_b} \ell(y, \mathbf{x}; \boldsymbol{\theta}),\ \text{s.t. } y = +1 \implies \Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0, \tag{6}$$

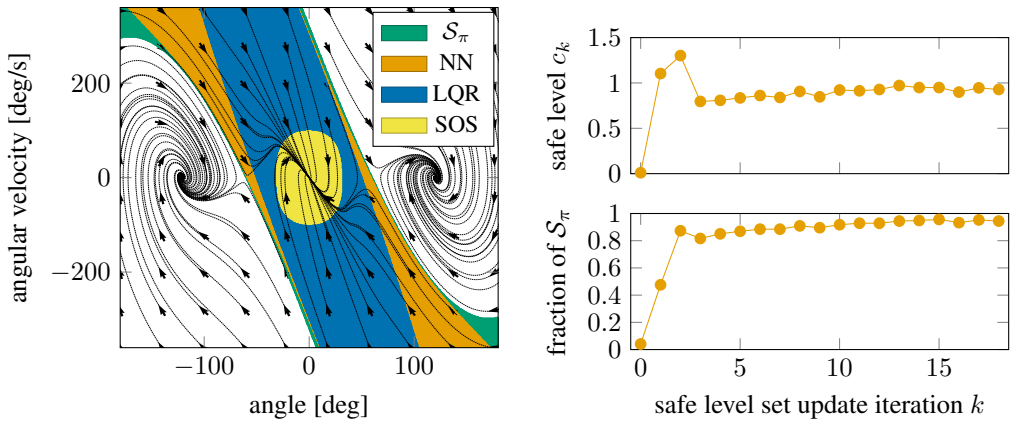
where the batch $\mathcal{X}_b$ is re-sampled after every gradient step. We can apply a Lagrangian relaxation

$$\min_{\boldsymbol{\theta}} \sum_{\mathbf{x} \in \mathcal{X}_b} \ell(y, \mathbf{x}; \boldsymbol{\theta}) + \lambda \left(\frac{y+1}{2}\right) \max\big(0,\ \Delta v_{\boldsymbol{\theta}}(\mathbf{x})\big) \tag{7}$$

in order to make the problem tractable. Here, $\lambda \in \mathbb{R}_{>0}$ is a Lagrangian multiplier and the term $\lambda((y+1)/2)\max\big(0, \Delta v_{\boldsymbol{\theta}}(\mathbf{x})\big)$ is the *Lyapunov decrease loss*, which penalizes violations of (5). The decrease condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ only needs to be enforced within the safe region $\mathcal{S}_\pi$, so we do not want to incur a loss if it is violated at a state where $y = -1$. Thus, we use the multiplier $(y+1)/2$ to map $\{+1, -1\}$ to $\{1, 0\}$, such that the Lyapunov decrease loss is zeroed-out if $y = -1$.

However, there are two issues when this formulation is compared to the exact problem in (3). Firstly, the objective (7) only penalizes violations of the decrease condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$, rather than constraining $\boldsymbol{\theta}$ to enforce it. Thus, while $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ is *always* a provable safety certificate, we must *verify* that it holds over some level set whenever we update $\boldsymbol{\theta}$. Secondly, ground-truth labels of $\mathcal{S}_\pi$ are not known in practice. To address these issues, we can use any method to check Lyapunov safety certificates over continuous state spaces to certify a level set $\mathcal{V}_{\boldsymbol{\theta}}(c)$ as safe, and then use $\mathcal{V}_{\boldsymbol{\theta}}(c)$ to estimate labels y from $\mathcal{S}_\pi$. For this work, we check the tightened certificate $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < -L_{\Delta v_{\boldsymbol{\theta}}} \tau$ on a discretization of $\mathcal{X}$, as described in Sec. 2.1. This method exposes the Lipschitz constant $L_{\Delta v_{\boldsymbol{\theta}}}$ of $\Delta v_{\boldsymbol{\theta}}$, which can conveniently be used for regularization in practice [21]. Possible alternatives to this safety verification method include the use of an adaptive discretization for better scaling to higher-dimensional state spaces [11], and formal verification methods for neural networks [27, 28].

Since such an estimate of $\mathcal{S}_\pi$ is limited by the largest safe level set of $v_{\boldsymbol{\theta}}$, we propose Algorithm 1 to iteratively "grow" an estimate of $\mathcal{S}_\pi$. We initialize $v_{\boldsymbol{\theta}}$, then use it to identify the largest safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_0)$ by verifying the condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$. At first, we use $\mathcal{V}_{\boldsymbol{\theta}}(c_0)$ to estimate $\mathcal{S}_\pi$. At iteration $k \in \mathbb{N}_{\geq 0}$, we consider the safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ and the expanded level set $\mathcal{V}_{\boldsymbol{\theta}}(\alpha c_k)$ for some $\alpha \in \mathbb{R}_{>1}$; see Fig. 2b. Then, states in the "gap" $\mathcal{G} \coloneqq \mathcal{V}_{\boldsymbol{\theta}}(\alpha c_k) \setminus \mathcal{V}_{\boldsymbol{\theta}}(c_k)$ are forward-simulated



(a) Safe Lyapunov candidate level sets.

(b) Training behaviour of neural network candidate.

Figure 3. Results for training the neural network (NN) Lyapunov candidate $v_{\boldsymbol{\theta}}$ for an inverted pendulum. In Fig. 3a, system trajectories (black) converge to the origin only within the largest safe region $\mathcal{S}_\pi$ (green). The NN candidate (orange) characterizes $\mathcal{S}_\pi$ with a level set better than both the LQR (blue ellipsoid) and SOS (yellow) candidates, as it adapts to the shape of $\mathcal{S}_\pi$. In Fig. 3b, the safe level c_k of $v_{\boldsymbol{\theta}}$ converges non-monotonically towards the fixed boundary $c_{\mathcal{S}} = 1$, and the safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ grows to cover most of $\mathcal{S}_\pi$. However, as discussed at the end of Sec. 3, convergence of $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ to $\mathcal{S}_\pi$ is not guaranteed in general by Algorithm 1.

with the dynamics f_π for $T \in \mathbb{N}_{\geq 1}$ time steps. States that fall in $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ before or after forward-simulation form a new estimate of $\mathcal{S}_\pi$, since trajectories become "trapped" in $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ and converge to the origin. We use this estimate of $\mathcal{S}_\pi$ to identify labels y for classification, then apply SGD with the objective (7) to update $\boldsymbol{\theta}$ and encourage $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ to grow. Finally, we certify the new largest safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_{k+1})$. These steps are repeated until a choice of stopping criterion is satisfied.

In general, Algorithm 1 does not guarantee convergence of the safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ to $\mathcal{S}_\pi$, nor that $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ monotonically grows in volume. Furthermore, it is not guaranteed that the iterated safe level $c_k \in \mathbb{R}_{>0}$ approaches the safe level $c_{\mathcal{S}}$ that is prescribed to delineate $\mathcal{S}_\pi$. This is typical of gradient-based parameter training, since the parameters $\boldsymbol{\theta}$ can become "stuck" in local optima. However, since the Lyapunov candidate $v_{\boldsymbol{\theta}}$ is guaranteed to satisfy the positive-definiteness and Lipschitz continuity conditions of Theorem 1 by its construction in Theorem 2, $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ is *always* a provable safety certificate for identifying safe level sets. Thus, we can *always* use $v_{\boldsymbol{\theta}}$ to identify at least a subset of $\mathcal{S}_\pi$, without ever identifying unsafe states as safe.

4 Experiments and Discussion

In the previous section, we developed Algorithm 1 to train the parameters $\boldsymbol{\theta}$ of a neural network Lyapunov candidate $v_{\boldsymbol{\theta}}$ constructed according to Theorem 2. This construction ensures the positive-definiteness and Lipschitz continuity assumptions on $v_{\boldsymbol{\theta}}$ in Theorem 1 are satisfied. Algorithm 1 encourages $v_{\boldsymbol{\theta}}$ to satisfy the decrease condition and match the true largest ROA $\mathcal{S}_\pi$ for the closed-loop dynamics f_π with a level set $\mathcal{V}_{\boldsymbol{\theta}}(c_{\mathcal{S}})$. In this section, we present details for the implementation of Algorithm 1 to learn the largest safe region of a simulated inverted pendulum system, and experimental results in a comparison to other methods of computing Lyapunov functions.

Inverted Pendulum Benchmark The inverted pendulum is governed by the differential equation $m\ell^2\ddot{\theta} = mg\ell\sin\theta - \beta\dot{\theta} + u$ with state $\mathbf{x} := (\theta, \dot{\theta})$, where θ is the angle from the upright equilibrium $\mathbf{x}_{\mathrm{O}} = \mathbf{0}$, u is the input torque, m is the pendulum mass, g is the gravitational acceleration, ℓ is the pole length, and β is the friction coefficient. We discretize the dynamics with a time step of $\Delta t = 0.01\,\mathrm{s}$ and enforce a saturation constraint $u \in [-\bar{u}, \bar{u}]$, such that the pendulum falls over past a certain angle and cannot recover. For a linear policy $u = \pi(\mathbf{x}) = \mathbf{K}\mathbf{x}$, this yields the safe region $\mathcal{S}_\pi$ in Fig. 3 around the upright equilibrium for the closed-loop dynamics f_π. In particular, we fix $\mathbf{K}$ to the linear quadratic regulator (LQR) solution for the discretized, linearized, unconstrained form of the dynamics [29]. Outside of $\mathcal{S}_\pi$, the pendulum falls down without the ability to recover and the system trajectories diverge away from $\mathbf{x}_{\mathrm{O}} = \mathbf{0}$.

Practical Considerations To train the parameters of the Lyapunov candidate $v_{\boldsymbol{\theta}}$ to adapt to the shape of $\mathcal{S}_\pi$, we use Algorithm 1 with SGD. To certify the safety of continuous level sets of $v_{\boldsymbol{\theta}}$ whenever $\boldsymbol{\theta}$ is updated, we check the stricter decrease condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < -L_{\Delta v_{\boldsymbol{\theta}}}\tau$ at a discrete set of points that cover $\mathcal{X}$ in increasing order of the value of $v_{\boldsymbol{\theta}}(\mathbf{x})$, as in [17]. Algorithm 1 does not guarantee that the safe level set estimate $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ grows monotonically in volume towards $\mathcal{S}_\pi$ with each iteration k. In fact, the estimate $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ may shrink if $v_{\boldsymbol{\theta}}$ initially succeeds and then fails to satisfy the decrease condition $\Delta v_{\boldsymbol{\theta}}(\mathbf{x}) < 0$ in some regions of the state space. This tends to occur near the origin, where $v_{\boldsymbol{\theta}}(\mathbf{0}) = \Delta v_{\boldsymbol{\theta}}(\mathbf{0}) = 0$ and the "basin of attraction" characterized by $v_{\boldsymbol{\theta}}$ "flattens". To alleviate this, we use a large Lagrange multiplier $\lambda = 1000$ in the SGD objective (7) to strongly "push" $\boldsymbol{\theta}$ towards values that ensure $v_{\boldsymbol{\theta}}$ continues to satisfy the decrease condition. In addition, we normalize the Lyapunov decrease loss $\lambda((y+1)/2)\max\left(0, \Delta v_{\boldsymbol{\theta}}(\mathbf{x})\right)$ in (7) by $v_{\boldsymbol{\theta}}(\mathbf{x})$. This more heavily weighs sampled states near the origin, i.e., where $v_{\boldsymbol{\theta}}(\mathbf{x})$ is small.

Results We implement Algorithm 1 on the inverted pendulum benchmark with the Python code available at `https://github.com/befelix/safe_learning`, which is based on TensorFlow [30]. For the neural network Lyapunov candidate $v_{\boldsymbol{\theta}}$, we use three layers of 64 $\tanh(\cdot)$ activation units each. We prescribe $\mathcal{V}_{\boldsymbol{\theta}}(c_{\mathcal{S}})$ with $c_{\mathcal{S}} = 1$ as the level set that delineates the safe region $\mathcal{S}_\pi$. Fig. 3 shows the results of training $v_{\boldsymbol{\theta}}$ with Algorithm 1, and the largest safe level set $\mathcal{V}_{\boldsymbol{\theta}}(c_{18})$ with 10 SGD iterations per update. Fig. 3a visualizes how this level set has "moulded" to the shape of $\mathcal{S}_\pi$. Fig. 3b shows how the safe level c_k converges towards the prescribed level $c_{\mathcal{S}} = 1$ that delineates $\mathcal{S}_\pi$, and how the fraction of $\mathcal{S}_\pi$ covered by $\mathcal{V}_{\boldsymbol{\theta}}(c_k)$ approaches 1. The true largest ROA $\mathcal{S}_\pi$ is estimated by forward-simulating all of the states in a state space discretization, and set volume is estimated by counting discrete states. Fig. 3a also shows the largest safe sets for a LQR Lyapunov candidate and a SOS Lyapunov candidate. The LQR candidate $v_{\mathrm{LQR}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{P}\mathbf{x}$ is computed in closed-form for the same discretized, linearized, unconstrained form of the dynamics used to determine the LQR policy $\pi(\mathbf{x}) = \mathbf{K}\mathbf{x}$ [29]. The SOS Lyapunov candidate $v_{\mathrm{SOS}}(\mathbf{x}) = m(\mathbf{x})^\top \mathbf{Q} m(\mathbf{x})$ uses up to third-order monomials in $\mathbf{x}$, thus it is a sixth-order polynomial. It is computed with the toolbox SOSTOOLS [31] and the SDP solver SeDuMi [32] in MATLAB for the unconstrained nonlinear dynamics with a Taylor polynomial expansion of $\sin\theta$. While the SOS approach is a powerful specialized method for polynomial dynamical systems, it cannot account for the non-differentiable nonlinearity introduced by the input saturation, which drastically alters the closed-loop dynamics. As a result, while v_{SOS} is optimized for the system without saturation, it is ill-suited to the true closed-loop dynamics and yields a small safe level set. Overall, our neural network Lyapunov candidate $v_{\boldsymbol{\theta}}$ performs the best at certification of as much of $\mathcal{S}_\pi$ as possible, since it only relies on inputs and outputs of f_π, and adapts to the shape of $\mathcal{S}_\pi$.

Comments on Safe Learning Fig. 3a demonstrates that a neural network Lyapunov candidate $v_{\boldsymbol{\theta}}$ can certify more of the true largest safe region $\mathcal{S}_\pi$ than other common Lyapunov candidates. This has important implications for safe exploration during learning for dynamical systems; with more safe states available to visit, an agent can better learn about itself and its environment under a wider range of operating conditions. For example, our method is applicable in the safe reinforcement learning framework of [17]. This past work provides safe exploration guarantees for a GP model of the dynamics f_π with confidence bounds on the Lyapunov stability certificate, but these guarantees are limited by the choice of Lyapunov function. As our results have shown, certain Lyapunov candidates may poorly characterize the shape of the true largest safe region $\mathcal{S}_\pi$. Since our neural network Lyapunov candidate can adapt to the shape of $\mathcal{S}_\pi$ during learning by using, for example, the mean estimate of f_π from the GP model, we could enlarge the estimated safe region more quickly as data is collected. Our method is also applicable to exploration algorithms within safe motion planning that depends on knowledge of a safe region, such as in [33]. Overall, our method strongly warrants consideration for use in safe learning methods that leverage statistical models of dynamical systems.

5 Conclusion

We have demonstrated a novel method for learning safety certificates for general nonlinear dynamical systems. Specifically, we developed a flexible class of parameterized Lyapunov candidate functions and a training algorithm to adapt them to the shape of the largest safe region for a closed-loop dynamical system. We believe that our method is appealing due to its applicability to a wide range of dynamical systems in theory and practice. Furthermore, it can play an important role in improving safe exploration during learning for real autonomous systems in uncertain environments.

Acknowledgments

This research was supported in part by SNSF grant 200020_159557, the Vector Institute, and a fellowship by the Open Philantropy Project.

References

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in AI safety. Technical report, 2016. arXiv:1606.06565v2 [cs.AI].

[2] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2002.

[3] A. Vannelli and M. Vidyasagar. Maximal Lyapunov functions and domains of attraction for autonomous nonlinear systems. *Automatica*, 21(1):69–80, 1985.

[4] D. J. Hill and I. M. Y. Mareels. Stability theory for differential/algebraic systems with application to power systems. *IEEE Transactions on Circuits and Systems*, 37(11):1416–1423, 1990.

[5] J. M. G. da Silva Jr. and S. Tarbouriech. Antiwindup design with guaranteed regions of stability: An LMI-based approach. *IEEE Transactions on Automatic Control*, 50(1):106–111, 2005.

[6] R. Kalman and J. Bertram. Control system analysis and design via the "second method" of Lyapunov II: Discrete-time systems. *Transactions of the American Society of Mechanical Engineers (ASME): Journal of Basic Engineering*, 82(2):394–400, 1960.

[7] P. Giesl and S. Hafstein. Review on computational methods for Lyapunov functions. *Discrete and Continuous Dynamical Systems, Series B*, 20(8):2291–2331, 2016.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2009.

[9] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[10] D. Henrion and M. Korda. Convex computation of the region of attraction of polynomial control systems. *IEEE Transactions on Automatic Control*, 59(2):297–312, 2014.

[11] R. Bobiti and M. Lazar. A sampling approach to finding Lyapunov functions for nonlinear discrete-time systems. In *Proc. of the European Control Conference (ECC)*, pages 561–566, 2016.

[12] K. Zhou and J. C. Doyle. *Essentials of Robust Control*. Prentice Hall, Upper Saddle River, NJ, 1998.

[13] A. Trofino. Robust stability and domain of attraction of uncertain nonlinear systems. In *Proc. of the American Control Conference (ACC)*, pages 3707–3711, 2000.

[14] U. Topcu, A. K. Packard, P. Seiler, and G. J. Balas. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137–142, 2010.

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[16] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 4661–4666, 2016.

[17] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. Safe model-based reinforcement learning with stability guarantees. In *Proc. of the Conference on Neural Information Processing Systems (NIPS)*, pages 908–918, 2017.

[18] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 2 edition, 2018. *(draft)*.

[19] V. Petridis and S. Petridis. Construction of neural network based Lyapunov functions. In *Proc. of the IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065, 2006.

[20] N. Noroozi, P. Karimaghaee, F. Safaei, and H. Javadi. Generation of Lyapunov functions by neural networks. In *Proc. of the World Congress on Engineering (WCE)*, volume 1, pages 61–65, 2008.

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

[22] A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 3482–3487, 2002.

[23] A. Papachristodoulou. *Scalable analysis of nonlinear systems using convex optimization*. PhD thesis, California Institute of Technology, 2005.

[24] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[25] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6(8):1069–1072, 2001.

[26] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, NY, 2006.

[27] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. Technical report, 2017. arXiv:1610.06940v3 [cs.AI].

[28] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. of the International Conference on Computer Aided Verification (CAV)*, 2017.

[29] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos. *Optimal Control*. John Wiley & Sons, Inc., Hoboken, NJ, 3 edition, 2012.

[30] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *Proc. of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–283, 2016.

[31] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 741–746, 2002.

[32] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1–4):625–653, 1999.

[33] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. Learning-based model predictive control for safe exploration. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2018. *(to appear)*.

A Proofs

Theorem 2 (Lyapunov neural network): *Consider $v_{\boldsymbol{\theta}}(\mathbf{x}) = \phi_{\boldsymbol{\theta}}(\mathbf{x})^\top \phi_{\boldsymbol{\theta}}(\mathbf{x})$ as a Lyapunov candidate function, where $\phi_{\boldsymbol{\theta}}$ is a feed-forward neural network. Suppose, for each layer ℓ in $\phi_{\boldsymbol{\theta}}$, the activation function φ_ℓ and weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ each have a trivial nullspace. Then $\phi_{\boldsymbol{\theta}}$ has a trivial nullspace, and $v_{\boldsymbol{\theta}}$ is positive-definite with $v_{\boldsymbol{\theta}}(\mathbf{0}) = 0$ and $v_{\boldsymbol{\theta}}(\mathbf{x}) > 0$, $\forall \mathbf{x} \in \mathcal{X} \setminus \{\mathbf{0}\}$. Furthermore, if φ_ℓ is Lipschitz continuous for each layer ℓ, then $v_{\boldsymbol{\theta}}$ is locally Lipschitz continuous.*

Proof. We begin by showing that $\phi_{\boldsymbol{\theta}}$ has a trivial nullspace in $\mathcal{X}$ by induction, and then use this to prove that $v_{\boldsymbol{\theta}}$ is positive-definite on $\mathcal{X}$. Recall that a feed-forward neural network is a successive composition of its layer transformations, such that the output $\mathbf{y}_\ell(\mathbf{x})$ of layer ℓ for the state $\mathbf{x} \in \mathcal{X}$ is the input to layer $\ell + 1$. Consider $\ell = 0$ with the input $\mathbf{y}_0(\mathbf{x}) \coloneqq \mathbf{x}$, and the first layer output $\mathbf{y}_1(\mathbf{x}) = \varphi_1(\mathbf{W}_1 \mathbf{y}_0(\mathbf{x}))$. Clearly $\mathbf{y}_0$ has a trivial nullspace in $\mathcal{X}$, since it is just the identity function. Since $\mathbf{W}_1$, φ_1, and $\mathbf{y}_0$ each have a trivial nullspace in their respective input spaces, the sequence of logical statements

$$\mathbf{x} = \mathbf{0} \iff \mathbf{y}_0(\mathbf{x}) = \mathbf{0} \iff \mathbf{W}_1 \mathbf{y}_0(\mathbf{x}) = \mathbf{0} \iff \varphi_1(\mathbf{W}_1 \mathbf{y}_0(\mathbf{x})) = \mathbf{0} \tag{8}$$

holds. Thus, $\mathbf{x} = \mathbf{0} \iff \varphi_1(\mathbf{W}_1 \mathbf{y}_0(\mathbf{x})) = \mathbf{0}$ holds, and $\mathbf{y}_1$ has a trivial nullspace in $\mathcal{X}$. If we now assume $\mathbf{y}_\ell$ has a trivial nullspace in $\mathcal{X}$, it is clear that $\mathbf{y}_{\ell+1}$ has a trivial nullspace in $\mathcal{X}$, since

$$\mathbf{x} = \mathbf{0} \iff \mathbf{y}_\ell(\mathbf{x}) = \mathbf{0} \iff \mathbf{W}_{\ell+1} \mathbf{y}_\ell(\mathbf{x}) = \mathbf{0} \iff \varphi_{\ell+1}(\mathbf{W}_{\ell+1} \mathbf{y}_\ell(\mathbf{x})) = \mathbf{0} \tag{9}$$

holds in a similar fashion. As a result, $\mathbf{y}_\ell$ has a trivial nullspace for each layer ℓ by induction. Since $\phi_{\boldsymbol{\theta}}$ is a composition of a finite number of layers, $\phi_{\boldsymbol{\theta}} = \mathbf{y}_L$ for some $L \in \mathbb{N}_{\geq 0}$, thus $\phi_{\boldsymbol{\theta}}$ has a trivial nullspace in $\mathcal{X}$.

We now use this property of $\phi_{\boldsymbol{\theta}}$ to prove that the Lyapunov candidate $v_{\boldsymbol{\theta}}(\mathbf{x}) = \phi_{\boldsymbol{\theta}}(\mathbf{x})^\top \phi_{\boldsymbol{\theta}}(\mathbf{x})$ is positive-definite on $\mathcal{X}$. As an inner product, $\phi_{\boldsymbol{\theta}}(\mathbf{x})^\top \phi_{\boldsymbol{\theta}}(\mathbf{x})$ is positive-definite on the transformed space $\mathcal{Y} \coloneqq \{\phi_{\boldsymbol{\theta}}(\mathbf{x}),\, \forall \mathbf{x} \in \mathcal{X}\}$. Thus, $v_{\boldsymbol{\theta}}(\mathbf{x}) = 0 \iff \phi_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{0}$ and $v_{\boldsymbol{\theta}}(\mathbf{x}) > 0$ otherwise. Since we have already proven $\phi_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{0} \iff \mathbf{x} = \mathbf{0}$, combining these statements shows that $v_{\boldsymbol{\theta}}(\mathbf{x}) = 0 \iff \mathbf{x} = \mathbf{0}$ and $v_{\boldsymbol{\theta}}(\mathbf{x}) > 0$ otherwise. As a result, $v_{\boldsymbol{\theta}}(\mathbf{x})$ is positive-definite on $\mathcal{X}$.

Finally, we need to show that if every activation function φ_ℓ is Lipschitz continuous, then $v_{\boldsymbol{\theta}}$ is locally Lipschitz continuous. If the neural network $\phi_{\boldsymbol{\theta}}$ is Lipschitz continuous, then clearly $v_{\boldsymbol{\theta}}$ is locally Lipschitz continuous, since it is quadratic and thus differentiable with respect to $\phi_{\boldsymbol{\theta}}$. To show that $\phi_{\boldsymbol{\theta}}$ is Lipschitz continuous, it is sufficient to show that each layer is Lipschitz continuous. This is due to the fact that any function composition $f(g(\mathbf{x}))$ is Lipschitz continuous with Lipschitz constant $L_f L_g$ if f has Lipschitz constant L_f and g has Lipschitz constant L_g. This fact can be seen from $\big\|f(g(\mathbf{x})) - f(g(\mathbf{x}'))\big\| \leq L_f \big\|g(\mathbf{x}) - g(\mathbf{x}')\big\| \leq L_f L_g \big\|\mathbf{x} - \mathbf{x}'\big\|$, for each pair $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. By the Lipschitz continuity of function composition and the linearity of $\mathbf{W}_\ell \mathbf{y}_{\ell-1}$, each layer transformation $\mathbf{y}_\ell = \varphi_\ell(\mathbf{W}_\ell \mathbf{y}_{\ell-1})$ is Lipschitz continuous if φ_ℓ is Lipschitz continuous. As a result, the neural network $\phi_{\boldsymbol{\theta}}$ is Lipschitz continuous, and the Lyapunov candidate $v_{\boldsymbol{\theta}}$ is locally Lipschitz continuous. ■

Remark 1: *In (2), we ensured each weight matrix $\mathbf{W}_\ell$ has a trivial nullspace with the structure*

$$\mathbf{W}_\ell = \begin{bmatrix} \mathbf{G}_{\ell 1}^\top \mathbf{G}_{\ell 1} + \varepsilon \mathbf{I}_{d_{\ell-1}} \\ \mathbf{G}_{\ell 2} \end{bmatrix},$$

where $\mathbf{G}_{\ell 1} \in \mathbb{R}^{q_\ell \times d_{\ell-1}}$ for some $q_\ell \in \mathbb{N}_{\geq 1}$, $\mathbf{G}_{\ell 2} \in \mathbb{R}^{(d_\ell - d_{\ell-1}) \times d_{\ell-1}}$, $\mathbf{I}_{d_{\ell-1}} \in \mathbb{R}^{d_{\ell-1} \times d_{\ell-1}}$ is the identity matrix, and $\varepsilon \in \mathbb{R}_{>0}$ is a constant. To minimize the number of free parameters required by our neural network Lyapunov candidate, we choose q_ℓ to be the minimum integer such that each entry in $\mathbf{G}_{\ell 1}^\top \mathbf{G}_{\ell 1} \in \mathbb{R}^{d_{\ell-1} \times d_{\ell-1}}$ is independent from the others. Since $\mathbf{G}_{\ell 1}^\top \mathbf{G}_{\ell 1}$ is symmetric, it has $\sum_{j=1}^{d_{\ell-1}} j = d_{\ell-1}(d_{\ell-1}+1)/2$ free parameters, thereby requiring $q_\ell d_{\ell-1} \geq d_{\ell-1}(d_{\ell-1}+1)/2$ or $q_\ell \geq (d_{\ell-1}+1)/2$. For this, we choose $q_\ell = \lceil (d_{\ell-1}+1)/2 \rceil$.