POLARNET: SINGLE-MINIMA NEURAL NETWORK FOR MODELING LYAPUNOV FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning control strategies with provable stability guarantees continues to be a challenging problem. In this work, we examine a family of training-time behaviors exhibited by existing neural Lyapunov control methods under specific conditions, which can hinder the synthesis of a provably stable controller. We identify the root cause as the lack of neural network architectural guarantees on the learned Lyapunov function, and propose PolarNet, a network architecture that provably addresses these issues by structurally guarantee to have a single critical point. We provide theoretical guarantee regarding the properness and universality of PolarNet for modeling Lyapunov functions, and show that using it as a drop-in replacement in existing neural Lyapunov control methods can effectively circumvent particular difficulties in training. We conduct a set of numerical experiments to verify that PolarNet consistently maintains a single critical point and, when used as a drop-in replacement in existing neural Lyapunov control methods, successfully avoids training failures caused by the lack of architectural guarantees.

1 INTRODUCTION

Lyapunov functions are a cornerstone of modern control theory, serving as the primary tool for analyzing the stability of dynamic systems modeled in state space. Despite their near-dominant role, the search for a Lyapunov function has, for over a century, relied heavily on the physical insights and mathematical ingenuity of domain experts. Constructing such functions remains a longstanding open problem in both control theory and mathematics (Alfarano et al. (2024)). Although various analytical methods have been developed within the control community to assist in constructing Lyapunov functions, such as solving the algebraic Riccati equation (Lewis et al. (2012)) and sum-of-squares programming (Papachristodoulou & Prajna (2002)), none of these approaches are universally applicable. Even for relatively simple systems, these methods may fail to produce a valid Lyapunov function or may incorrectly conclude that no such function exists. Recent research (e.g., Chang et al. (2019); Wei et al. (2023; 2025); Zhou et al. (2022); Wu et al. (2023); Mehrjou et al. (2020)) has explored an intriguing learning-based alternative that automatically synthesizes provably stable controllers by jointly learning a control strategy and a Lyapunov function, with the latter serving as the stability certificate. Specifically, by embedding the Lyapunov conditions into the learning objective, these methods, collectively known as neural Lyapunov control, are applicable to complex uncertain nonlinear systems without sacrificing a theoretical stability guarantee. This line of work has achieved remarkable empirical success on a wide range of challenging control tasks, demonstrating a promising direction beyond conventional analytical approaches.

However, the reliance on flexible neural networks to learn Lyapunov functions introduces a new challenge that is absent in traditional analytical methods. We identify a family of training-time behaviors that can arise under specific conditions in existing neural Lyapunov control methods, that can potentially prevent the learning process from properly converging and thus blocking the synthesis of a provably stable controller. We trace this issue to the absence of a suitable structural guarantee in the neural networks used to learn Lyapunov functions. Specifically, we show that for a certain class of general nonlinear systems, any valid Lyapunov function must possess exactly one critical point, yet the network architectures commonly adopted in existing methods do not enforce this property structurally. This issue is specific to learning-based paradigms and has therefore been overlooked in the recent shift from analytical methods to learning-based controller synthesis.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To fill this gap, we propose PolarNet, a neural network architecture that provides a rigorous structural guarantee by exploiting an interesting mathematical fact. A key theoretical result is that PolarNet always possesses exactly one critical point, thereby circumventing the training difficulty described above. Furthermore, we show that under a mild condition, PolarNet is capable of constructing Lyapunov functions for any asymptotically stable equilibrium of general dynamic systems. We validate the effectiveness of PolarNet through numerical experiments on both synthetic function approximation tasks and a challenging nonlinear control problem. The results demonstrate that PolarNet consistently maintains a single critical point and, when used as a drop-in replacement in existing neural Lyapunov control methods, successfully avoids training failures caused by the lack of architectural guarantees. To the best of our knowledge, PolarNet is the first architecture that structurally enforces the single-critical-point property for Lyapunov function learning, offering a principled and practical solution to a previously overlooked failure mode in neural Lyapunov control.

2 PRELIMINARY

2.1 PROBLEM FORMULATION

Consider the following general dynamic system, which is an ordinary differential equation defined as:

$$\dot{x} = f(x, u), \qquad x(t_0) = x_0, \tag{1}$$

where $x \in \mathcal{X} \subset \mathbb{R}^m$ is the system state, $u \in \mathcal{U} \subset \mathbb{R}^n$ is the control input, $\dot{x}$ denotes the derivative of x w.r.t. t, and $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ is an unknown nonlinear function that is locally Lipschitz continuous in $\mathcal{X}$. Suppose the system equation 1 is controllable, and $x_e \in \mathcal{X}$ is an equilibrium point of equation 1, that is, $f(x_e, 0) = 0$. The goal of the task is to synthesize an appropriate input $u(x)$, such that the closed-loop system is asymptotically stable in the sense of Definition 1.

Definition 1 (Asymptotic stability (Khalil (2002)))**.** Let x_e be an equilibrium point of the closed-loop autonomous system $\dot{x} = f(x, u(x))$. Then x_e is said to be asymptotically stable if there exists $\delta > 0$ such that for every initial condition satisfying $\|x_0 - x_e\| < \delta$, we have $\lim_{t\to\infty} x(t) = x_e$.

Lyapunov functions $V(x) : \mathbb{R}^m \to \mathbb{R}$ are special functions whose existence imply that a dynamic system is asymptotically stable. Formally:

Definition 2 (Lyapunov stability criteria (Khalil (2002)))**.** Without loss of generality, let $x = 0$ be an equilibrium point of $\dot{x} = f(x, u(x))$. Suppose $V : \mathbb{R}^m \to \mathbb{R}$ is a radially unbounded $\mathcal{C}^1$ function satisfying: *i)* $V(0) = 0$, *ii)* $V(x) > 0, \forall\, x \neq 0$, and $\dot{V}(x) < 0, \forall\, x \neq 0$, then $x = 0$ is asymptotically stable. In this case, $V(x)$ is called a Lyapunov function of this system.

Throughout this work, we consider the case where $x_e = 0$ is the only desired stable equilibrium.

Recent research (e.g., Chang et al. (2019); Wei et al. (2023; 2025)) has found that, by directly learning to satisfy the Lyapunov stability criteria, it is possible to synthesize a controller u that stabilizes a general dynamic system. The general idea is to parameterize the controller and a corresponding Lyapunov function with two neural networks u_θ and V_θ, and to train them simultaneously by minimizing the empirical risk of violating the Lyapunov stability criteria. This empirical risk (Chang et al. (2019)) is defined as:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\max(0, -V_\theta(x_i)) + \max(0, \dot{V}_\theta(x_i)) + V_\theta^2(0)\right), \tag{2}$$

where $\{x_i \in \mathcal{X} | i \in \{1, 2, \ldots, N\}\}$ is a set of points in $\mathcal{X}$, called the training set, and each x_i is an m-dimensional state vector sampled from $\mathcal{X}$.

2.2 LIMITATION OF EXISTING METHODS

For a proper Lyapunov function V_θ that satisfies $\dot{V}_\theta(x) < 0, \forall x \neq 0$, its derivative along the system trajectory is given by the chain rule:

$$\dot{V}_\theta(x) = \langle \nabla V_\theta(x), \dot{x} \rangle, \qquad \forall x \in \mathcal{X}, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. If there exists a point $x^* \neq 0$ such that $\langle \nabla V_\theta(x), x^* \rangle = 0$, then we would have $\dot{V}_\theta(x^*) = 0$, contradicting $\dot{V}_\theta(x) < 0$ for all $x \neq 0$. Therefore, such a point cannot exist. This gives the following necessary condition for valid Lyapunov function V_θ:

$$\nabla V_\theta(x) \neq 0, \qquad \forall x \neq 0. \tag{4}$$

Recent works of neural Lyapunov control (e.g., Wei et al. (2025); Gaby et al. (2022)) typically model Lyapunov functions with specialized networks $V_\theta(x) : \mathbb{R}^m \to \mathbb{R}$ that are always positive definite, and pursue the $\dot{V}_\theta(x) < 0, \forall x \neq 0$ condition by minimizing its empirical risk function as defined in (2). With such network that provides structural guarantee of $V_\theta(0) = 0$ and $V_\theta(x) > 0, \forall x \neq 0$, (2) can be reduced to:

$$\frac{1}{N} \sum_{i=1}^{N} \max\left(0, \dot{V}_\theta(x_i)\right). \tag{5}$$

While these methods have demonstrated the ability to synthesize controllers that can be effective in practice, their network architectures do not inherently guarantee that (4) stays true. As a result, training may encounter difficulties in certain scenarios. Consider, for example, a setup where we train u_θ and V_θ by minimizing the empirical risk in (5) over uniformly sampled $x_i \in \mathcal{X}$ with scalar state and input signals. Under this training strategy, the situation depicted in Fig. 1 may occur: at some point during training, V_θ learned a shape with multiple local minima (see Fig. 1(a)). Fig. 1(b) shows the partial derivative $\partial V_\theta / \partial x$ of this learned function, where the existence of points with zero derivative (other than the origin) clearly violates the necessary condition (4). In addition, since (5) is computed with the current V_θ and u_θ that are still being trained, this loss term is essentially encouraging u_θ to steer the states along the descending directions of V_θ, regardless of whether V_θ itself is a valid Lyapunov function or not. Consequently, the closed-loop dynamics determined by f and u_θ may take a shape similar to that shown in Fig. 1(c). In this case, $\dot{V}_\theta < 0$ is not always true, and $\dot{V}_\theta = 0$ occurs at the origin, P and Q (see Fig. 1(d)). However, the empirical risk in (5) remains zero, even if the Lyapunov stability criteria is clearly violated at points P and Q. Furthermore, training of V_θ and u_θ is performed on a finite number of samples $\{x_i\}$ in practice. So even when the roots of $\dot{x} = 0$ do not perfectly coincide with P and Q, the empirical risk can still vanish to zero at this stage, providing little useful gradient information for continued learning. Consequently, u_θ may fail to eliminate the spurious equilibria (e.g., P and Q), and more importantly, the training may stall at these points, offering no guarantee that the learned controller achieves asymptotic stability of the desired equilibrium $x = 0$.

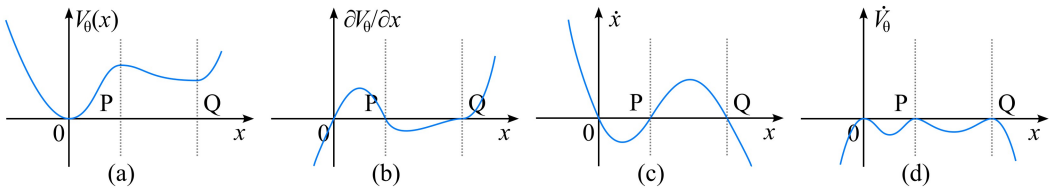

Figure 1: A possible scenario that may arise when using existing neural network architectures (e.g., Chang et al. (2019); Wei et al. (2023; 2025); Gaby et al. (2022)), shown as functions of the state x along the horizontal axis. As a numerical example, consider a scalar system $\dot{x} = u - x(x - P)(x - Q)$. The learned Lyapunov candidate function could take the form $V_\theta(x) = x^2 + \sin^2(\pi x)$, and P, Q are the smallest two positive roots of equation $\partial V_\theta / \partial x = 0$. If at $x = P$ (or $x = Q$), the learned controller satisfies $u_\theta = 0$, then $\dot{x} = 0$ and hence the system trajectory will remain at $x(t) = P$ (or $x(t) = Q$), and the true equilibrium $x = 0$ is not globally asymptotically stable, which hinders the successful synthesis of a provably stable controller. An experimental demonstration of such a scenario is provided in Section 4.3.

Existing methods adopt various strategies to mitigate this issue. One common approach adds a small positive constant ϵ to the loss term in (5), making it non-zero unless $\dot{V}_\theta < -\epsilon$, thereby expanding the region where the loss provides useful gradients (Wei et al. (2023; 2025)). Another approach actively seeks out points $x_i \neq 0$ where $\dot{V}_\theta(x_i) \geq 0$ via numerical optimization and re-construct the training set to include these points, allowing the model to focus on eliminating these violations (Chang et al. (2019)). Similarly, a third strategy simulates closed-loop trajectories to assess the stability of initial states, constructing the training set around the current region of attraction (RoA) to maintain and

gradually expand it during training (Wei et al. (2023); Mehrjou et al. (2020)). While these strategies may alleviate the issue during training, they do not completely eliminate it. We demonstrate this with an experiment in Section 4.3. To the best of our knowledge, no existing method fundamentally resolves this issue by enforcing a network structural guarantee on the learned Lyapunov function itself.

It should also be noted that this issue is unique to learning-based methods that use flexible neural networks to model Lyapunov function, and is not present in analytic methods (Khalil (2002)).

3 METHOD

3.1 NETWORK ARCHITECTURE DESIGN

To address the issue described in Section 2.2, we propose PolarNet, a network architecture that guarantees a single-critical-point shape in addition to positive-definiteness. By construction, this architecture strictly enforces (4) to be true, regardless of how its parameters are set. Without multiple local minima, the problem investigated in Section 2.2 is fundamentally resolved, as its precondition would never be true. Formally speaking, PolarNet models a single pole function, which we define as:

Definition 3 (Single pole function)**.** A single pole function is a function $V(x) : \mathcal{X} \to \mathbb{R}, \mathcal{X} \subset \mathbb{R}^m, 0 \in \mathcal{X}$, such that it:

$i)$ is $\mathcal{C}^\infty$ smooth $\forall x \in \mathcal{X}$;

$ii)$ is positive definite: $V(0) = 0; V(x) > 0, \forall x \neq 0$;

$iii)$ has exactly one critical point at $x = 0$: $\nabla V \neq 0, \forall x \neq 0$;

$iv)$ is proper on $\mathcal{X}$, i.e., every sublevel set $\{x \in \mathcal{X} | V(x) \leq c\}$ of V is compact.[1]

We additionally show in Section 3.2 that, there exists a Lyapunov function, which is also a single pole function, for the equilibrium of the general dynamic system (1).

We first reveal an interesting connection between a single pole function and a smoothly invertible mapping $\Psi(x) : \mathbb{R}^m \to \mathbb{R}^m$. Exploiting this connection, by integrating invertible neural networks (i.e., neural networks that are invertible by design, e.g., Dinh et al. (2017); Kingma & Dhariwal (2018); Kingma et al. (2016); Papamakarios et al. (2017)) into the model architecture, we can naturally construct a neural network that guarantees to be a single pole function.

We prove that, under a mild condition, any single pole function can be expressed in the form $V = \|\Psi(x)\|^2$, where $\Psi(x) : \mathbb{R}^m \to \mathbb{R}^m$ is a $\mathcal{C}^\infty$ smooth, invertible and zero-crossing function, and $\|\cdot\|$ is the standard Euclidean norm. Geometrically, this means V can always be smoothly deformed to match an Euclidean ball $\|\cdot\|^2$, as illustrated in Fig. 2:

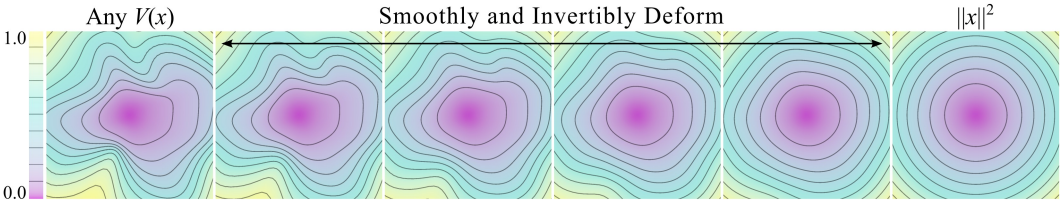

Figure 2: Graphical illustration of Theorem 1: The case of $V(x) : \mathbb{R}^m \to \mathbb{R}, m = 2$. As illustrated, a single pole function $V(x)$ can be spatially distorted by a smoothly invertible mapping $\Psi(x) : \mathbb{R}^m \to \mathbb{R}^m$, such that it matches an Euclidean ball $\|x\|$ after distortion. In other words, $V(x)$ can always be reformulated as the squared Euclidean norm $\|z\|^2$ in a smoothly distorted space $z = \Psi(x)$. For visual clarity, we visualize $\mathbb{R}^2 \to \mathbb{R}$ functions as contour lines.

Theorem 1. For all single pole functions $V(x) : \mathbb{R}^m \to \mathbb{R}$, if the origin is a non-degenerate critical point of V (i.e., the Hessian matrix $D^2V(0)$ is invertible), then there exists a $\mathcal{C}^\infty$ smooth,

[1] When $x \in \mathbb{R}^m$ and V is continuous, $V(x)$ being proper is equivalent to $V(x)$ being radially unbounded.

invertible and zero-crossing vector-valued function $\Psi(x) : \mathbb{R}^m \to \mathbb{R}^m$, such that $V = \|\Psi(x)\|^2$ holds throughout $\mathcal{X}$.

Proof. The idea of the proof is to directly construct a diffeomorphism Ψ between the sublevel sets $\{x \in \mathcal{X} | V(x) \leq c\}$ and standard spheres. This is done by smoothly identifying level sets $L_c = \{x \in \mathcal{X} | V(x) = c\}$ to each other with the flow (Lee (2012)) of $V(x)$, thus constructing a diffeomorphism between the product structure $L_c \times (0, \infty)$ and $\mathcal{X} \setminus \{0\}$. Then we use the Morse lemma (Milnor (1963); Nirenberg (2001)) to obtain a diffeomorphism between a particular level set of $V(x)$ and a standard sphere. By composing these two diffeomorphism, we can construct an eligible Ψ such that $V(x) = \|\Psi(x)\|^2$. For a detailed and rigorous proof, see Appendix A. □

Remark 1. While Theorem 1 requires a non-degenerate critical point, this is a mild condition, as non-degenerancy is a common situation, thus any degenerate critical point can be perturbed by an arbitrarily small perturbation to become a non-degenerate critical point (see Chapter 1.7 of Guillemin & Pollack (2010)).

We construct PolarNet V_θ exactly as $V_\theta(x) = \|\Psi(x)\|^2$. We use the affine coupling flow architecture (Dinh et al. (2017); Kingma & Dhariwal (2018); Kingma et al. (2016); Papamakarios et al. (2017)) to implement Ψ. This architecture $h_{cf}(x) : \mathbb{R}^m \to \mathbb{R}^m$ is defined as:

$$h^{(l)} \begin{pmatrix} y_{\leq k} \\ y_{>k} \end{pmatrix} = \begin{pmatrix} y_{\leq k} \\ \exp(f_s^{(l)}(y_{\leq k})) \odot y_{>k} + f_t^{(l)}(y_{\leq k}) \end{pmatrix}, l \in \{1, 2, \ldots, L\}, y = h^{(l-1)}, \tag{6a}$$

$$h^{(0)}(x) = x, \tag{6b}$$

$$h_{cf}(x) = h^{(L)} \circ h^{(L-1)} \circ \cdots \circ h^{(1)}(x), \tag{6c}$$

where $k \in \{1, \ldots, m-1\}$, $\odot$ is the Hadamard product (i.e., element-wise product), $\exp$ is applied in an element-wise manner, and $\circ$ is the function composition operator $a \circ b(y) = a(b(y))$, $a \circ b \circ c = a \circ (b \circ c)$. Both f_s and f_t are arbitrary neural networks. For each layer $h^{(l)}$, its input $y = (y_1, \cdots, y_m)^\intercal$ is partitioned into 2 parts $y_{\leq k} = (y_1, \cdots, y_k)^\intercal$ and $y_{>k} = (y_{k+1}, \cdots, y_m)^\intercal$. After this layer, $y_{\leq k}$ is kept unchanged, and $y_{>k}$ is being transformed in an affine manner, with scale $\exp f_s(y_{\leq k})$ and translation $f_t(y_{\leq k})$ derived from $y_{\leq k}$. Both $f_s(\cdot)$ and $f_t(\cdot)$ are arbitrary neural networks. This architecture guarantees that, if $f_s(\cdot)$ is finite, the inverse of $h^{(l)}$ exists, and can be formulated as follows:

$$h^{-1(l)} \begin{pmatrix} y_{\leq k} \\ y_{>k} \end{pmatrix} = \begin{pmatrix} y_{\leq k} \\ \dfrac{y_{>k} - f_t(y_{\leq k})}{\exp f_s(y_{\leq k})} \end{pmatrix}. \tag{7}$$

In order to ensure that the entire network V_θ is $\mathcal{C}^\infty$ smooth, we only use smooth activation functions (tanh) throughout the network. To achieve $V_\theta(0) = 0$, we additionally ensure that $\Psi(0) = 0$. Specifically, we construct the translation function f_t in each affine layer $h^{(l)}$ using a multi-layer perceptron (MLP) (Rumelhart et al. (1986)) without bias (and using tanh activation, which is zero-crossing). This guarantees that $\exp f_s(0) \odot 0 + f_t(0) = 0$. In other words, each layer is zero-crossing, so does their composition Ψ.

3.2 Theoretical guarantee

Theorem 1 has shown that all single pole functions $V(x)$ can be formulated as $V(x) = \|\Psi(x)\|^2, \forall x \in \mathcal{X} \setminus \{0\}$. As has been proved in (Teshima et al. (2020)), the affine coupling flow architecture is a universal approximator of $\mathcal{C}^2$ diffeomorphisms in $\mathbb{R}^m$. In other words, all $\mathcal{C}^2$ differentiably invertible mappings in $\mathbb{R}^m$ can be approximated by the affine coupling flow architecture to arbitrary precision. And by restricting all components of this architecture to be $\mathcal{C}^\infty$, we can obtain a universal approximator for Ψ. Hence, a single pole function can always be approximated by V_θ. Now we also show that $V_\theta(x) = \|\Psi(x)\|^2$ is always a single pole function, regardless of how its parameters are set (as long as they are finite):

Theorem 2. If $\Psi : \mathbb{R}^m \to \mathbb{R}^m$ is a $\mathcal{C}^\infty$ smooth, zero-crossing invertible mapping, then $\|\Psi(x)\|^2$ is a single pole function.

Proof. We prove for each property:

$i)$ Since both $\|\cdot\|^2$ and Ψ are $\mathcal{C}^\infty$ smooth, their composition $\|\Psi(\cdot)\|^2$ is also $\mathcal{C}^\infty$ smooth.

$ii)$ Invertibility of Ψ implies that $\Psi(x) \neq \Psi(0), \forall x \neq 0$, hence $\Psi(x) \neq \Psi(0) = 0, \forall x \neq 0$. By definition of the squared norm $\|\cdot\|^2$, we have $\|\Psi(x)\|^2 \neq 0, \forall x \neq 0$. Since $\|\cdot\|^2 \geq 0$, $\|\Psi(x)\|^2$ is positive definite.

$iii)$ Since Ψ is invertible, it has no critical point. $\|\cdot\|^2$ has exactly one critical point at 0, $\Psi(x) = 0$ only at $x = 0$. By the chain rule of derivative computation, $\|\Psi(x)\|^2$ has exactly one critical point at $x = 0$.

$iv)$ $N(y) = \|y\|^2$ is proper on $\mathbb{R}^m$ (with codomain $[0, +\infty)$), hence for any compact set $K \subset [0, +\infty)$, its preimage $N^{-1}(K)$ is also compact. Invertibility of Ψ implies continuity of Ψ^{-1}, and the continuous image of any compact set is also a compact set. Thus $\Psi^{-1}(N^{-1}(K))$ is also compact. Hence $\|\Psi(x)\|^2$ is proper on $\mathbb{R}^m \supset \mathcal{X}$.

We have thus shown that $\|\Psi(x)\|^2$ is always a single pole function. □

We also show that, for all asymptotically stable equilibrium of a general dynamic system (1), there exists a single pole function V, such that V is a Lyapunov function of this equilibrium:

Theorem 3. Let $x = 0$ be an asymptotically stable equilibrium of a dynamic system $\dot{x} = f(x, u(x))$, where $f(x, u(x))$ is locally Lipschitz. Let $\mathcal{R}_A \in \mathcal{X}$ be the region of attraction of $x = 0$. Then there exists a Lyapunov function $V(x)$ for this equilibrium, that is also a single pole function.

Proof. By Kurzweil's converse Lyapunov theorem (*Theorem 4.17* in Khalil (2002)), and by the conclusion shown in (4), for an asymptotically stable equilibrium $x = 0$ of a dynamic system, there exists a Lyapunov function $V(x) : \mathcal{X} \to \mathbb{R}$ that is: $i)$ $\mathcal{C}^\infty$ smooth, $ii)$ positive definite, $iii)$ has exactly one critical point at $x = 0$, and $iv)$ satisfies $\forall c > 0, \{x \in \mathcal{X} | V(x) \leq c\}$ is compact, i.e., $V(x)$ is proper in $\mathcal{R}_A$. In other words, it is a single pole function (By Definition 3, letting $\mathcal{X} = \mathcal{R}_A$, one can conclude that V is a single pole function). □

Theorem 3 shows that for any controller synthesis task demanding an asymptotically stable equilibrium, there exists a Lyapunov function that can be accurately modeled by PolarNet. Combining with Theorem 2, this provides a principled method to circumvent training difficulties caused by the lack of architectural guarantees in existing neural Lyapunov control methods.

4 Experiments

4.1 Network architecture implementation

In all experiments, we implement PolarNet with 4 affine coupling flow layers, (i.e., use the architecture described in (6) with $L = 4$), evenly partition the dimension of input vectors, and alternate the part being preserved in adjacent layers. In other words, we implement Ψ as:

$$\Psi(x) = \begin{pmatrix} y_{\leq k} \\ g^{(4)}(y_{\leq k}, y_{>k}) \end{pmatrix} \circ \begin{pmatrix} g^{(3)}(y_{>k}, y_{\leq k}) \\ y_{>k} \end{pmatrix} \circ \begin{pmatrix} y_{\leq k} \\ g^{(2)}(y_{\leq k}, y_{>k}) \end{pmatrix} \circ \begin{pmatrix} g^{(1)}(y_{>k}, y_{\leq k}) \\ y_{>k} \end{pmatrix}, \quad (8)$$

where $g^{(l)}(a, b) = \exp(f_s^{(l)}(a)) \odot b + f_t^{(l)}(a)$ and $y^0 = x$. All $f_s^{(l)}$ and $f_t^{(l)}$ are MLPs with two 12-dimensional hidden layers. And $f_t^{(l)}$ does not have a bias term. Each linear layer in these MLPs are followed by a tanh activation function, with the last layer (of each MLP) as an exception, which uses an identity activation.

4.2 Studying properties of PolarNet

In this experiment, we verify that PolarNet is always a single pole function. Particularly, we show that it would never produce critical point other than 0, even when directly trained to fit a function with multiple critical points. Specifically, we manually design several scalar fields $V(x) : \mathbb{R}^2 \to \mathbb{R}, x \in (-1, +1) \times (-1, +1)$ and train a neural network V_θ to fit them by uniformly sample $x_i \in (-1, +1) \times (-1, +1)$, and minimize a mean-squared-error (MSE) loss function:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(V_\theta(x_i) - V(x_i)\right)^2. \quad (9)$$

We compare PolarNet against several existing and representative network architectures for modeling Lyapunov functions. Namely: i) MLP (Chang et al. (2019); Rumelhart et al. (1986)): A multi-layer perceptron with three 64-dimensional hidden layers and tanh activation. ii) Lyapunov Net (Gaby et al. (2022)): defined as $|h(x) - h(0)| + \gamma\|x\|^2$, where $h(x)$ is an MLP with three 64-dimensional hidden layers, $\gamma > 0$ is a hyperparameter that is set to 1e-2, as in Gaby et al. (2022). And iii) Wei et al. (2025): defined as $\frac{1}{2}\beta x^\intercal x + \frac{1}{2}\phi_\theta(x)^\intercal \phi_\theta(x)$, where $\phi_\theta(x)$ is an MLP with three 64-dimensional hidden layers, no bias term, and using tanh activation, β is a hyperparameter that is set to 1e-6, as in Wei et al. (2025). The computation resource and training configuration being used to conduct this experiment is described in Appendix B.

The results are presented in Fig. 3. We found that, when V is a single pole function, all methods have comparable performance. However, when V is not a single pole function, existing architectures would still fit it without problem, thus producing multiple local minima. As a consequence, they may learn to exhibit multiple local minima when being used to model Lyapunov functions in controller synthesis tasks, thus hindering the successful synthesis of a globally stable controller. On the other hand, PolarNet remains a single pole function in all cases. Hence it can always guarantee that (4) stays true throughout training process, avoiding this family of potential fail cases.

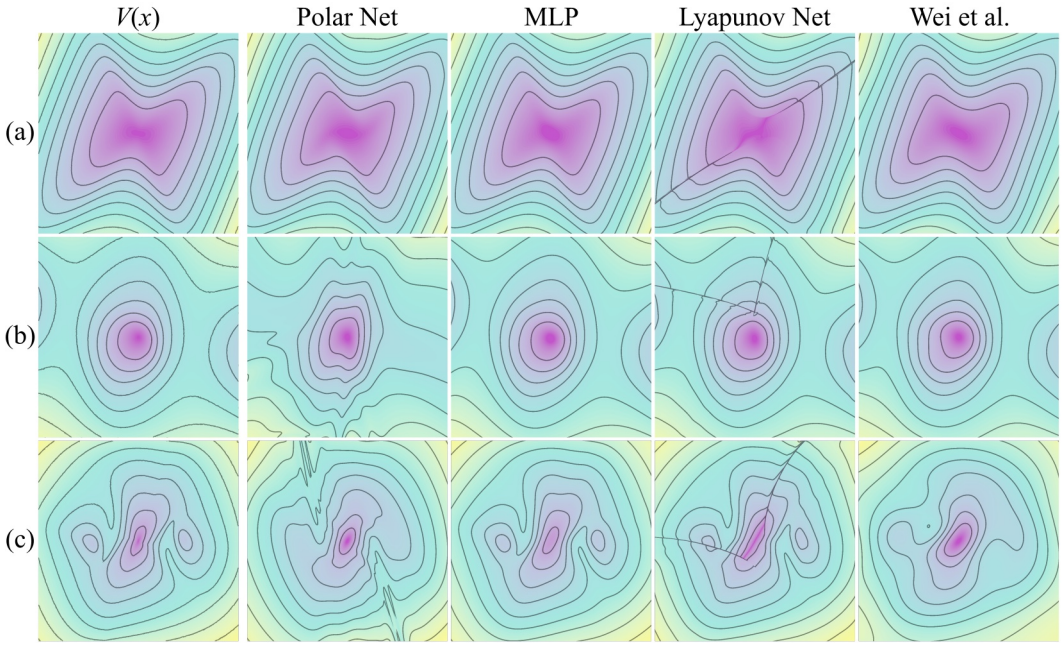

Figure 3: Results of fitting four different $V(x) : \mathbb{R}^2 \to \mathbb{R}$. In (a), V is a single pole function. In (b) and (c), V has multiple local minima and thus is not a single pole function. Note that a network that guarantees to be a single pole function should fail to faithfully represent V in cases (b), (c). We visualize $\mathbb{R}^2 \to \mathbb{R}$ as contour lines for visual clarity, using the same scheme as in Fig. 2.

4.3 Controller Synthesis

We verify the effectiveness of PolarNet in a controller synthesis task using numerical simulation. Consider the following nonlinear dynamic system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \frac{1}{2}\sin(\pi x_1) + (20\exp(-2x_1^2 - \frac{1}{2}x_2^2) - 10)u_1 + 50g(x_1)u_2 - \frac{1}{10}u_2 \end{cases}, \tag{10a}$$

$$g(x_1) = \begin{cases} \frac{k}{2}\left(\frac{x_1}{k}\right)^2 & \text{if } |x_1| < k \\ |x_1| - \frac{k}{2} & \text{otherwise} \end{cases}, \tag{10b}$$

where $k = 0.02$, $x = (x_1\ x_2)^\intercal, x \in \mathcal{X} = (-1, +1) \times (-1, +1)$ is the system state, $u = (u_1\ u_2)^\intercal \in \mathbb{R}^2$ is the input. Note that $g(x_1)$ is $\mathcal{C}^1$ and globally Lipschitz, as the values and first order derivatives coincide at $|x_1| = k$, and the derivative is bounded between $[-1, +1]$. This system is also a control-affine system (i.e., it can be formulated as $\dot{x} = f_1(x) + f_2(x)u$, this matches the assumption made in Wei et al. (2023)). Note that this system model is unknown to the controller.

We verify the effectiveness of PolarNet with a recent neural Lyapunov control method proposed in Wei et al. (2023). We first show that, when using an existing neural network architecture as in Wei et al. (2023), this method could fail to stabilize the dynamic system (10). And when we duplicate the same setting, except that we replace the network for Lyapunov function by PolarNet, we can successfully stabilize the dynamic system. We use a learning rate of 5e-6 for both the controller and Lyapunov function networks, and use a batch size of 32. Training is performed for 200 iterations (100 gradient update steps for each iteration). Further technical details used in this experiment are given in Appendix C. The baseline architecture for comparison is implemented exactly as described in Wei et al. (2023). The results are as shown in Fig. 4.

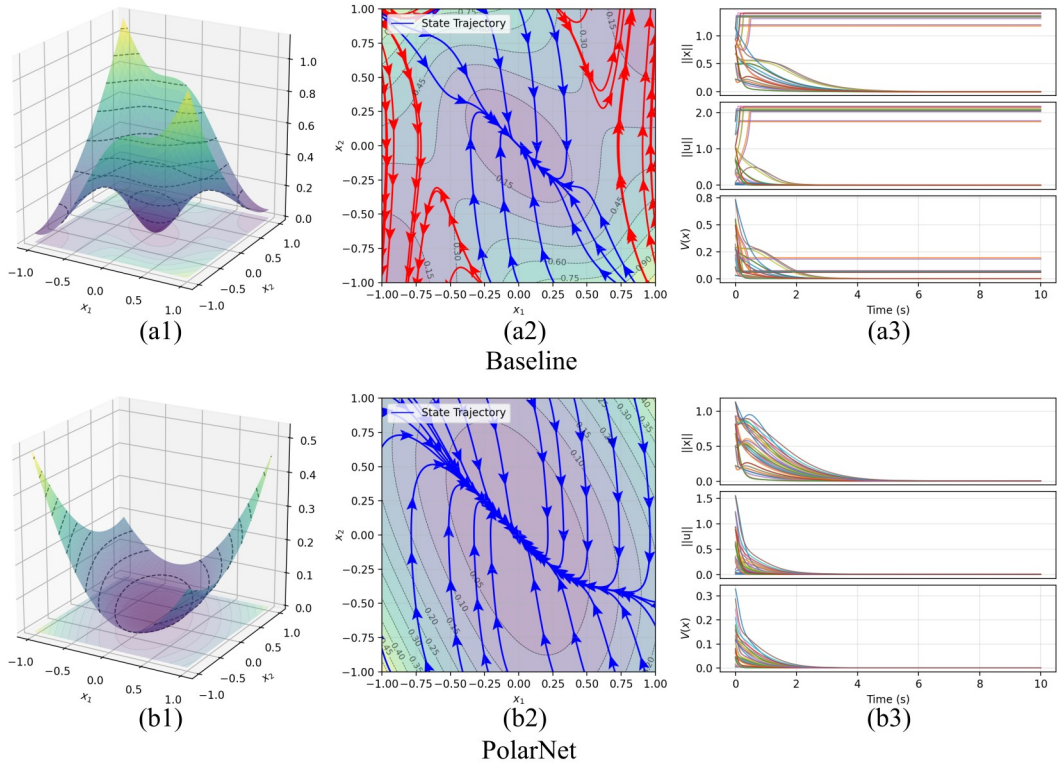

Figure 4: Controller synthesis result with different network architectures: (a1), (b1): Lyapunov function. (a2), (b2): Phase portrait, unstable trajectories are marked in red. (a3), (b3): 36 numerically simulated trajectories. (a): Using the network architecture as in Wei et al. (2023). (b): Using PolarNet. $V(x)$ and $\|x\|$ trajectories are collected by uniformly sample $x_0 \in [-0.8, +0.8]^2$, and if the state leaves the valid region $\mathcal{X} = (-1, +1)^2$ of the system dynamics, simulation is terminated for that trajectory. Notice how states evolve along descending directions of V_θ for $x \in (-1, +1)^2$ in (a) but do not converge to the origin, as analyzed in Section 2.2.

As shown in Fig. 4, in this experimental setup, existing neural network architecture, that does not guarantee (4), may learn a shape with multiple local minima, causing controller synthesis to fail. When directly replacing the network with PolarNet, V_θ is guaranteed to have a unique minima at 0, and a controller that stabilize the dynamic system in the entire $\mathcal{X}$ can be successfully synthesized. We conduct this experiment across 20 different random seeds, and consistently obtained 62% larger RoA compared to baseline. More details can be found in Appendix C.

We also conduct experiments with another method (Wei et al. (2025)) and obtained similar results. Details can be found in Appendix D.

It should be noted that this experiment is designed to verify both the existence of the issue described in Section 2.2 and the effectiveness of PolarNet in circumventing it. Our method is at the principle level and thus applies universally to any system, whether synthetic or real world, provided the system satisfies the underlying mathematical assumptions.

5 Limitations and future directions

The theoretical guarantee provided by our work assumes a locally Lipschitz dynamic system. When the system is not Lipschitz, PolarNet may still be applied, but does not guarantee that it can always represent a valid Lyapunov function for the system (i.e., Theorem 3 does not cover this). Future works may seek to generalize PolarNet to work with a broader continuum of dynamic systems.

Our method also only focus on continuous time dynamic system and Lyapunov stability, while discrete systems and other stability definitions are also of interest in many cases. Our work does not consider the case in discrete systems and stability definitions other than the Lyapunov stability.

It should also be noted that PolarNet is specifically designed for controller synthesis task with a single desired equilibrium. For out-of-scope tasks such as finding stability certificate of existing dynamic systems (Boffi et al. (2021)), PolarNet does not have clear advantages over existing architectures, and may even be less expressive if the intention is to find multiple stable equilibriums. Care should be taken when trying to apply PolarNet in any of these out-of-scope tasks.

6 Related works

Neural Lyapunov control methods commonly learn a controller u_θ and its Lyapunov function V_θ simultaneously, yielding provable stability guarantees. Several works (Chang et al. (2019); Zhou et al. (2022); Wu et al. (2023)) adopt a learner-falsifier architecture: a falsifier searches the state space for Lyapunov stability violations (counterexamples), which are added to the training set; the learner minimizes empirical violation risk via gradient descent. If no counterexample is found, u_θ is deemed stable and V_θ certifies it. Among these, Chang et al. (2019); Zhou et al. (2022) formulate counterexample generation as a *satisfiability modulo theories* (SMT) problem (Barrett & Tinelli, 2018), solved by numerical SMT solvers. In contrast, Wu et al. (2023) propose a *projected gradient descent* (PGD) algorithm to approximately find counterexamples, improving training efficiency.

Other methods, such as Wei et al. (2023); Mehrjou et al. (2020), depart from learner-falsifier. They iteratively evaluate the region of attraction (RoA) of u_θ and construct training sets covering the current RoA and a surrounding region, maintaining contiguity while expanding it. Mehrjou et al. (2020) determines the RoA solely by checking Lyapunov criteria with the current u_θ and V_θ, whereas Wei et al. (2023) also collects numerically simulated trajectories to assess initial-state stability more reliably. Most target continuous-time systems, whereas Wu et al. (2023) addresses discrete-time systems. Wei et al. (2025) instead trains u_θ and V_θ on a finite state-space grid without sophisticated training sets, reporting empirical success. Wei et al. (2023; 2025); Gaby et al. (2022) use positive-definite-by-construction networks for V_θ, eliminating a positive-definiteness loss. To our knowledge, PolarNet is the first architecture that structurally enforces the single-critical-point property for Lyapunov function learning.

7 Conclusion

In this work, we analyzed the cause of a family of training-time difficulties that may present in existing neural Lyapunov control methods, identifying their root cause as lack of neural network architectural guarantees, and propose a new network architecture, PolarNet, that addresses this issue by construction. We derived theoretical guarantees that PolarNet possess the desired mathematical properties by construction, and can be applied to all asymptotically stable equilibrium of locally Lipschitz dynamic systems. We then verified the effectiveness of PolarNet with numerical experiments, demonstrating that PolarNet indeed possesses the desired properties in practice, and can circumvent certain difficulties encountered in the training process of existing neural Lyapunov control method, as we have analyzed. We also noted on the theoretical assumptions we made. Future works may seek to generalize PolarNet to be provably effective in a wider range of scenarios.

AI USE STATEMENT

In this work, we used generative AI tools to aid or polish writing, find related academic materials. We have not used generative AI tools for data generation, method design, conduct experiments, implement methods, interpret results or proving claims (except for finding existing results that are used in the proof of theorems).

We have reviewed all AI-assisted work. We manually verified that the manuscript after being polished by LLM properly reflects our intent. We also manually verified that all materials, including related works and existing results, are real and are correctly interpreted in this work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

For theoretical results, the proof of Theorem 1 is provided in the appendix (with a sketch in the main paper), while the proof of all other theorems are either immediately given (for theorems we derived) or cited from references (for existing theorems).

For all our experiments, we provide detailed information in Section 4.1, Appendices C and D. We also provide code and instructions in the supplemental material to reproduce the results.

REFERENCES

Alberto Alfarano, François Charton, and Amaury Hayat. Global lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. In *Advances in Neural Information Processing Systems*, volume 37, pp. 93643–93670. Curran Associates, Inc., 2024. doi: 10.52202/079017-2969.

Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pp. 305–343. Springer, 2018.

Nicholas Boffi, Stephen Tu, Nikolai Matni, Jean-Jacques Slotine, and Vikas Sindhwani. Learning stability certificates from data. In Jens Kober, Fabio Ramos, and Claire Tomlin (eds.), *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1341–1350. PMLR, 16–18 Nov 2021.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Nathan Gaby, Fumin Zhang, and Xiaojing Ye. Lyapunov-net: A deep neural network architecture for lyapunov function approximation. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2091–2096, 2022. doi: 10.1109/CDC51059.2022.9993006.

Victor Guillemin and Alan Pollack. *Differential Topology*. American Mathematical Society, August 2010. ISBN 9781470411350. doi: 10.1090/chel/370.

Hassan K Khalil. *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

John M. Lee. *Introduction to Smooth Manifolds*. Springer New York, 2012. ISBN 9781441999825. doi: 10.1007/978-1-4419-9982-5.

Frank L. Lewis, Draguna L. Vrabie, and Vassilis L. Syrmos. *Optimal Control*. Wiley, January 2012. ISBN 9781118122631. doi: 10.1002/9781118122631.

Arash Mehrjou, Mohammad Ghavamzadeh, and Bernhard Schölkopf. Neural lyapunov redesign. *arXiv preprint arXiv:2006.03947*, 2020.

John Milnor. *Morse Theory. (AM-51)*. Princeton University Press, December 1963. ISBN 9781400881802. doi: 10.1515/9781400881802.

Louis Nirenberg. *Topics in Nonlinear Functional Analysis*. American Mathematical Society, April 2001. ISBN 9781470431082. doi: 10.1090/cln/006.

A. Papachristodoulou and S. Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pp. 3482–3487, 2002. doi: 10.1109/CDC.2002.1184414.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. In *Nature*, volume 323(6088), pp. 533–536. Springer Science and Business Media LLC, October 1986. doi: 10.1038/323533a0.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. In *Advances in Neural Information Processing Systems*, volume 33, pp. 3362–3373. Curran Associates, Inc., 2020.

Shiqing Wei, Prashanth Krishnamurthy, and Farshad Khorrami. Neural lyapunov control for nonlinear systems with unstructured uncertainties. In *2023 American Control Conference (ACC)*, pp. 1901–1906, 2023.

Shiqing Wei, Prashanth Krishnamurthy, and Farshad Khorrami. Robust neural lyapunov control for nonlinear systems with quadratically bounded disturbances. In *International Journal of Robust and Nonlinear Control*, volume 36(4), pp. 2299–2311, 2025. doi: https://doi.org/10.1002/rnc.70279.

Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural lyapunov control for discrete-time systems. In *Advances in Neural Information Processing Systems*, volume 36, pp. 2939–2955. Curran Associates, Inc., 2023.

Ruikun Zhou, Thanin Quartz, Hans De Sterck, and Jun Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. In *Advances in Neural Information Processing Systems*, volume 35, pp. 29113–29125. Curran Associates, Inc., 2022.

A Proof of Theorem 1

Proof. The proof goes by first showing that we can smoothly identify different regular level sets (i.e., level sets that do not contain the critical point 0) of V, and construct a diffeomorphism $\Phi : L_c \times (-c, \infty) \to \mathcal{X} \setminus \{0\}$. We then smoothly map a regular level set to the standard sphere using Morse lemma, thus obtaining Ψ_0 such that $V(x) = \|\Psi_0(x)\|^2, \forall x \in \mathcal{X} \setminus \{0\}$. Finally we extend Ψ_0 smoothly to 0 to obtain a suitable Ψ.

A.1 GRADIENT FLOW AND A GLOBAL PRODUCT STRUCTURE

Since V is $\mathcal{C}^\infty$ and has no critical points in $\mathcal{X} \setminus \{0\}$, its gradient ∇V is also $\mathcal{C}^\infty$ and never zero on this set. Hence the *normalised gradient vector field*

$$X_{\text{grad}}(x) = \frac{\nabla V(x)}{\|\nabla V(x)\|^2}, \qquad x \in \mathcal{X} \setminus \{0\}, \tag{11}$$

is well defined and of class $\mathcal{C}^\infty$. Let $\phi_t(x)$ denote the *flow* of X_{grad}, i.e. the unique maximal solution of the ordinary differential equation $\frac{d}{dt}\phi_t(x) = X_{\text{grad}}(\phi_t(x))$ with $\phi_0(x) = x$. For each x, the flow is a function of t defined on a maximal open interval $(T_-(x), T_+(x))$ containing 0 (by the *Fundamental Theorem on Flows* in Lee (2012)). If $T_+(x) < \infty$, then the trajectory must leave every compact subset of $\mathcal{X}$ as $t \to T_+(x)$ (by the *Escape Lemma* in Lee (2012)).

Now compute the derivative of V along the flow:

$$\frac{\mathrm{d}}{\mathrm{d}t} V(\phi_t(x)) = \big\langle \nabla V(\phi_t(x)),\, \dot{\phi}_t(x) \big\rangle = \big\langle \nabla V(\phi_t(x)),\, \frac{\nabla V(\phi_t(x))}{\|\nabla V(\phi_t(x))\|^2} \big\rangle = 1. \tag{12}$$

Therefore, $V(\phi_t(x)) = V(x) + t$ as long as the flow is defined. Because V is proper on $\mathcal{X}$ and vanishes only at the origin, by definition of properness, the sublevel set $S_c = \{x \in \mathcal{X} : V(x) \le c\}$ is compact for every $c \ge 0$.

For any $T > 0$ and any starting point $x \neq 0$, the forward segment $\{\phi_t(x) : 0 \le t \le T\}$ is contained in the compact sublevel set $S_{V(x)+T}$, which lies inside $\mathcal{X} \setminus \{0\}$ because $V \ge V(x) > 0$ there. The Escape Lemma then implies that the flow cannot grow to infinity in finite forward time, hence $T_+(x) = +\infty$. Similarly, moving backwards, V decreases linearly until it reaches 0. The trajectory approaches the origin but never escapes $\mathcal{X}$ because V is proper. Consequently the lower endpoint is exactly $T_-(x) = -V(x)$, and the flow exists for all $t \in (-V(x),\, +\infty)$.

For a regular value $c > 0$, consider the level set $L_c = V^{-1}(c)$, which is a smooth $(m-1)$-dimensional manifold (by the *Regular Level Set Theorem* in Lee (2012)). The map

$$\Phi : L_c \times (-c, \infty) \to \mathcal{X} \setminus \{0\}, \qquad \Phi(p,t) = \phi_t(p), \qquad \forall p \in L_c, t \in (-c, \infty), \tag{13}$$

is a bijection, because every $x \neq 0$ can be written uniquely as $\phi_t(p)$ with $p = \phi_{-(V(x)-c)}(x) \in L_c$ and $t = V(x) - c$. The inverse is $x \mapsto \big(\phi_{-(V(x)-c)}(x), V(x) - c\big)$. Both Φ and its inverse are $\mathcal{C}^\infty$ (the flow of a $\mathcal{C}^\infty$ vector field is $\mathcal{C}^\infty$), so Φ is a $\mathcal{C}^\infty$ diffeomorphism. Thus we obtain a global product structure: every level set of V is diffeomorphic to L_c, and the punctured domain $\mathcal{X} \setminus \{0\}$ is diffeomorphic to $L_c \times (0, \infty)$ (with diffeomorphism Φ).

A.2 MORSE COORDINATES NEAR THE ORIGIN

Lemma 1 (Morse Lemma (Milnor (1963)))**.** Let p be a non-degenerate critical point for f (where f is a smooth real-valued function), then there is a local coordinate system $(y^1, \cdots, y^n)$ in a neighbourhood $\mathcal{U}$ of p with $y^i(p) = 0$ for all i and such that the identity $f = f(p) - (y^1)^2 - \cdots - (y^\lambda)^2 + (y^{\lambda+1})^2 + \cdots + (y^n)^2$ holds throughout $\mathcal{U}$, where λ is the index of f at p.

Corollary 1. There exists an open neighbourhood $\mathcal{U}_0 \subset \mathcal{X}$ of 0 and a $\mathcal{C}^\infty$ diffeomorphism

$$\varphi : \mathcal{U}_0 \to B_\delta(0), \qquad B_\delta(0) = \{x \| \|x\| < \delta\}, \tag{14}$$

such that $V(x) = \|\varphi(x)\|^2$ for all $x \in \mathcal{U}_0$. And $\mathcal{U}_0$ can be chosen such that $\mathcal{U}_0 \subset \mathcal{X}$.

Proof. Since 0 is a non-degenerate critical point, its index λ is 0. Let $p = 0$ and $\lambda = 0$, by applying Lemma 1, we can conclude that there exists a coordinate system $(y^1, \cdots, y^n)$ in a neighbourhood $\mathcal{U}$ of 0, such that $f = (y^1)^2 + \cdots + (y^n)^2$ holds throughout $\mathcal{U}$.

We take $\mathcal{U}_0$ as the intersection $\mathcal{U}_0 = \mathcal{U} \cap \mathcal{X}$. By definition of open sets, and consider the fact that $0 \in \mathcal{U}, 0 \in \mathcal{X} \implies 0 \in \mathcal{U}_0, \mathcal{U}_0$ is also an open neighbourhood of $p = 0$. Thus, we have obtained $\mathcal{U}_0$ and a coordinate system $(y^1, \cdots, y^n)$, such that the identity $f = (y^1)^2 + \cdots + (y^n)^2$ holds throughout $\mathcal{U}_0$.

By taking $\varphi(x)$ to be the coordinate transform, i.e., $\varphi(x) = (y^1, \cdots, y^n)^\intercal$, we have $V(x) = \|\varphi(x)\|^2$.

Moreover, by theorem 3.1.1 (Morse Lemma) given in Nirenberg (2001), there exists such a $\varphi(x)$ that is also $\mathcal{C}^\infty$ smooth. This completes the proof of Corollary 1. □

Remark 2. Note that Corollary 1 only shows that there exists some $\mathcal{U}_0 \in \mathcal{X}$ where $V(x) = \|\varphi(x)\|^2$ holds, hence letting $\Psi = \varphi$ does not complete our proof (which demands $V(x) = \|\Psi(x)\|^2, \forall x \in \mathcal{X}$).

A.3 Smoothly map a single level set to a sphere

Recall that, by corollary 1, we have $V(x) = \|\varphi(x)\|^2, \forall x \in \mathcal{U}_0$, and $\mathcal{U}_0$ is smoothly deformed by φ into an open ball $B_\delta(0)$ of radius δ. In other words, $V(x) < \delta^2, \forall x \in \mathcal{U}_0$.

Choose $a = \delta^2/4$. Then $L_a = V^{-1}(a)$ is contained in the Morse neighbourhood $\mathcal{U}_0$. The diffeomorphism φ maps L_a onto the sphere $\partial B_{\sqrt{a}}(0)$, consequently

$$h : L_a \to S^{m-1}, \qquad h(x) = \frac{\varphi(x)}{\|\varphi(x)\|} \tag{15}$$

is a $\mathcal{C}^\infty$ diffeomorphism onto the unit sphere S^{m-1}.

A.4 Global construction of Ψ

Using the product structure (13) with $c = a$, define

$$F : \mathcal{X} \setminus \{0\} \to S^{m-1} \times (0, \infty), \tag{16a}$$

$$F(x) = \Big(h\big(\Phi(x, -(V(x) - a))\big), V(x)\Big) = \Big(h\big(\phi_{-(V(x)-a)}(x)\big), V(x)\Big). \tag{16b}$$

Note that $\phi_{-(V(x)-a)}(x) \in L_a$ because $V(\phi_{-(V(x)-a)}(x)) = V(x) - (V(x) - a) = a$. The map F is the composition of the diffeomorphism Φ (with the second parameter re-expressed) and the diffeomorphism h on the first factor, hence F is a $\mathcal{C}^\infty$ diffeomorphism.

Remark 3. In other words, we use ϕ_t to diffeomorphically map every level set L_c to L_a via ϕ_{a-c}, and then smoothly map L_a onto the unit sphere S^{m-1} with φ obtained from the Morse coordinates. We also use $V(x)$ as coordinate along an extra dimension in the product $S^{m-1} \times (0, \infty)$, so that we have a diffeomorphism from $\mathcal{X} \setminus \{0\}$ to the product $S^{m-1} \times (0, \infty)$.

Now compose F with map $P : S^{m-1} \times (0, \infty) \to \mathbb{R}^m \setminus \{0\}, P(u, r) = \sqrt{r}\, u$:

$$\Psi_0(x) = P(F(x)) = \sqrt{V(x)} h\big(\phi_{-(V(x)-a)}(x)\big), \qquad \forall x \neq 0. \tag{17}$$

Set $\Psi_0(0) = 0$. For $x \neq 0$ we immediately obtain

$$\|\Psi_0(x)\| = \sqrt{V(x)} \quad \Longrightarrow \quad V(x) = \|\Psi_0(x)\|^2, \qquad \forall x \neq 0. \tag{18}$$

Note that this is $\mathcal{C}^\infty$ on $\mathcal{X} \setminus \{0\}$.

A.5 Extension to the origin

To obtain a global diffeomorphism (that additionally includes $x = 0$), we modify the normalized gradient field $X(x)$ near the origin so that its flow coincides with the radial flow coming from φ. Choose numbers r_1, r_2 with $\delta/2 < r_1 < r_2 < \delta$ and a smooth function $\eta : [0, \infty) \to [0, 1]$ such that

$$\eta(s) = 0 \ \ (s \leq r_1), \qquad \eta(s) = 1 \ \ (s \geq r_2). \tag{19}$$

On $\mathcal{U}_0 \setminus \{0\}$, define:

$$X_{\mathrm{rad}}(x) = (D\varphi(x))^{-1}\varphi(x), \tag{20}$$

By the chain rule,

$$\langle \nabla V(x),\, X_{\mathrm{rad}}(x) \rangle = \big\langle \nabla(\|\varphi(\cdot)\|^2)(x),\, (D\varphi(x))^{-1}\varphi(x) \big\rangle = 2\|\varphi(x)\|^2 = 2V(x). \tag{21}$$

Normalize it to $Y_{\mathrm{rad}}(x) = X_{\mathrm{rad}}(x)/(2V(x))$, then $\langle \nabla V(x), Y_{\mathrm{rad}} \rangle = 1$ and Y_{rad} is $\mathcal{C}^\infty$ on $\mathcal{U}_0 \setminus \{0\}$. Now we define the blended vector field as:

$$\widetilde{X}(x) = \begin{cases} \big(1 - \eta(\|\varphi(x)\|)\big)\, Y_{\mathrm{rad}}(x) + \eta(\|\varphi(x)\|)\, X_{\mathrm{grad}}(x), & x \in \mathcal{U}_0 \setminus \{0\}, \\ X_{\mathrm{grad}}(x), & x \in \mathcal{X} \setminus \mathcal{U}_0. \end{cases} \tag{22}$$

Since $\langle \nabla V(x), Y_{\mathrm{rad}} \rangle = \langle \nabla V(x), X_{\mathrm{grad}} \rangle = 1$, and $\widetilde{X}$ is a linear blend of Y_{rad} and X_{grad}, we have:

$$\langle \nabla V(x), \widetilde{X}(x) \rangle = 1, \qquad \forall x \neq 0. \tag{23}$$

$\widetilde{X}$ is $\mathcal{C}^\infty$ on $\mathcal{X} \setminus \{0\}$ and coincides with Y_{rad} on the region $\{0 < \|\varphi(x)\| \leq r_1\}$.

Let $\widetilde{\phi}_t$ be the flow of $\widetilde{X}$. By the same properness argument as in Appendix A.1, $\widetilde{\phi}_t(x)$ exists for all $t \in (-V(x), \infty)$ and stays inside $\mathcal{X} \setminus \{0\}$, and $V(\widetilde{\phi}_t(x)) = V(x) + t$. Because $\widetilde{X} = Y_{\mathrm{rad}}$ near the origin, the flow there is exactly the radial flow of the Morse chart φ. In other words, for $0 < \|\varphi(x)\| \leq r_1$, let $y(t) = \varphi(\widetilde{\phi}_t(x))$. Using $\dot{\widetilde{\phi}}_t = \widetilde{X}(\widetilde{\phi}_t)$(by definition) and the equality $\widetilde{X} = Y_{\mathrm{rad}}$ on the region in question, we obtain

$$\dot{y}(t) = \frac{y(t)}{2\|y(t)\|^2}, \qquad y(0) = \varphi(x). \tag{24}$$

The solution to (24) is:

$$y(t) = \frac{\sqrt{\|\varphi(x)\|^2 + t}}{\|\varphi(x)\|}\, \varphi(x). \tag{25}$$

Setting $t = a - V(x)$ and using $V(x) = \|\varphi(x)\|^2$ gives

$$y(a - V(x)) = \frac{\sqrt{a}}{\|\varphi(x)\|}\, \varphi(x) = \sqrt{a}\, \frac{\varphi(x)}{\|\varphi(x)\|}, \tag{26}$$

hence

$$\widetilde{\phi}_{a - V(x)}(x) = \varphi^{-1}\left(\sqrt{a}\, \frac{\varphi(x)}{\|\varphi(x)\|} \right). \tag{27}$$

Now repeat the construction of Ψ_0 using the new flow. Let

$$\Psi(x) = \sqrt{V(x)}\; h\big(\widetilde{\phi}_{a - V(x)}(x)\big), \qquad \forall x \neq 0,\ \Psi(0) = 0. \tag{28}$$

For x with $\|\varphi(x)\| \leq r_1$, by (27), we have:

$$\Psi(x) = \|\varphi(x)\|\, \frac{\varphi(x)}{\|\varphi(x)\|} = \varphi(x), \qquad \forall x \in \{x \in \mathcal{X} | V(x) \leq a\}. \tag{29}$$

Thus Ψ coincides with the $\mathcal{C}^\infty$ diffeomorphism φ on the whole ball $\{x : \|\varphi(x)\| < r_1\}$. In particular, Ψ is $\mathcal{C}^\infty$ on that neighbourhood, $\Psi(0) = 0$, and $D\Psi(0) = D\varphi(0)$ is invertible.

Outside that neighbourhood, Ψ is the composition of the flow diffeomorphism $\widetilde{\Phi}(p, t) = \widetilde{\phi}_t(p)$, the diffeomorphism h, and the polar map $P(u, r) = \sqrt{r}\, u$; hence it is also a $\mathcal{C}^\infty$ diffeomorphism. Additionally, by construction in (28), we have:

$$\|\Psi(x)\|^2 = V(x), \qquad \forall\, x \in \mathcal{X}. \tag{30}$$

This completes the proof. □

B Computation resource of function fitting experiment

This experiment is conducted on a laptop with Intel i9-13900HX CPU, 32G RAM and NVIDIA RTX4060 GPU (8GB VRAM). Each individual case (out of 16) displayed in Fig. 3 takes around 1 minute to run. A discrete GPU with at least 2GB VRAM is recommended for running this experiment.

We train the models for 10000 steps using the Adam optimizer (Kingma & Ba (2015)), with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon =$ 1e-8, 5e-3 learning rate with 400 steps warm-up, and a batch size of 65536.

The manually designed scalar fields used in this experiments are saved as 2D images with 1024 pixels in both height and width, and are provided in the supplemental material alongside the code.

Table 1: Hyperparameters of controller synthesis experiment. Hyperparameters not listed here are kept the same as in the inverted pendulum experiment in Wei et al. (2023) (this includes the learning rate and batch size).

Name	Value	Remarks
roa_adaptive_level_multiplier	False	Whether to decrease RoA level multiplier over time
cutoff_radius	0.1	Radius of initial safe set

C Details of controller synthesis experiment

To ensure correctness of the implementation, we implement the experiment by adapting code publicly released by Wei et al. (https://github.com/shiqingw/Neural-Lyapunov-Uncertainties, eg1_inverted_pendulum_2d/inv_pend_2d_sum4_nn_controller.py). A full list of all relevent hyperparameters is given in Table 1.

Note that, the authors of Wei et al. (2023) applied a loose saturation function to the output of the controller only to limit its capability and to demonstrate that their method could indeed gradually expand the RoA of the equilibrium (Wei et al. (2023)). In our experiment, we removed this function and use the output of the controller network as-is. Also, (Wei et al. (2023)) took uncertainty of the system dynamics into account, and used both an accurate "true" dynamics, and a nominal dynamics that is allowed to be slightly inaccurate in their algorithm. We consider a simplified case where the nominal system dynamics is accurate by setting the nominal system to be exactly the same as the true system.

It should also be noted that this method starts by first synthesizing an LQR (Lewis et al. (2012)) controller for linearization of the full nonlinear system dynamics at $x = 0$. For this, we also give the linearization of the system described in (10) as:

$$\begin{cases} \dot{x_1} = x_2 \\ \\ \dot{x_2} = \dfrac{\pi}{2} x_1 + 10 u_1 - \dfrac{1}{2} u_2. \end{cases} \tag{31}$$

Note that this method is also effective for uncertain nonlinear systems, this linearization is only used as part of the method, see Wei et al. (2023) for further details.

This experiment is performed on a laptop with Intel i9-13900HX CPU, 32G RAM for around 40 minutes. This experiment is performed on CPU without using CUDA.

We also report the RoA sizes obtained with baseline and PolarNet in Table 2 (averaged over 20 runs using different random seeds).

Table 2: RoA sizes achieved by baseline (Wei et al., 2023) and PolarNet, relative to area of $\mathcal{X}$. Computed by simulating 100×100 trajectories uniformly sampled on $\mathcal{X}$, until convergence.

Network Architecture	Roa area relative to $\mathcal{X}$
Baseline	59.83%
PolarNet	96.95%

D Controller synthesis experiment with an alternative method

This experiment follows the same design as in Section 4.3, but with another neural Lyapunov control method proposed in Wei et al. (2025). The dynamic system being considered is:

$$\begin{cases} \dot{x}_1 = 5 \exp\left(-40(x_1 - 0.4)^2 - 3x_2^2\right) + u_1 \\ \\ \dot{x}_2 = \dfrac{x_1 - x_2}{10} + u_2 \end{cases}, \tag{32}$$

where $x = (x_1\ x_2)^\intercal, x \in \mathcal{X} = (-1, +1)^2$ is the state, $u = (u_1\ u_2)^\intercal \in (-5, +5)^2$ is the input.

To ensure correctness, we implement this experiment by adapting code publicly released by Wei et al. (https://github.com/shiqingw/Robust-Lyapunov-Prob, MIT License). This experiment is conducted on a laptop with Intel i9-13900HX CPU, 32G RAM and NVIDIA RTX4060 GPU (8GB VRAM) for 200 epochs, with 10000 samples in each epoch, following the practice of the authors. Each run takes around 5 minutes. The results are shown in Fig. 5.

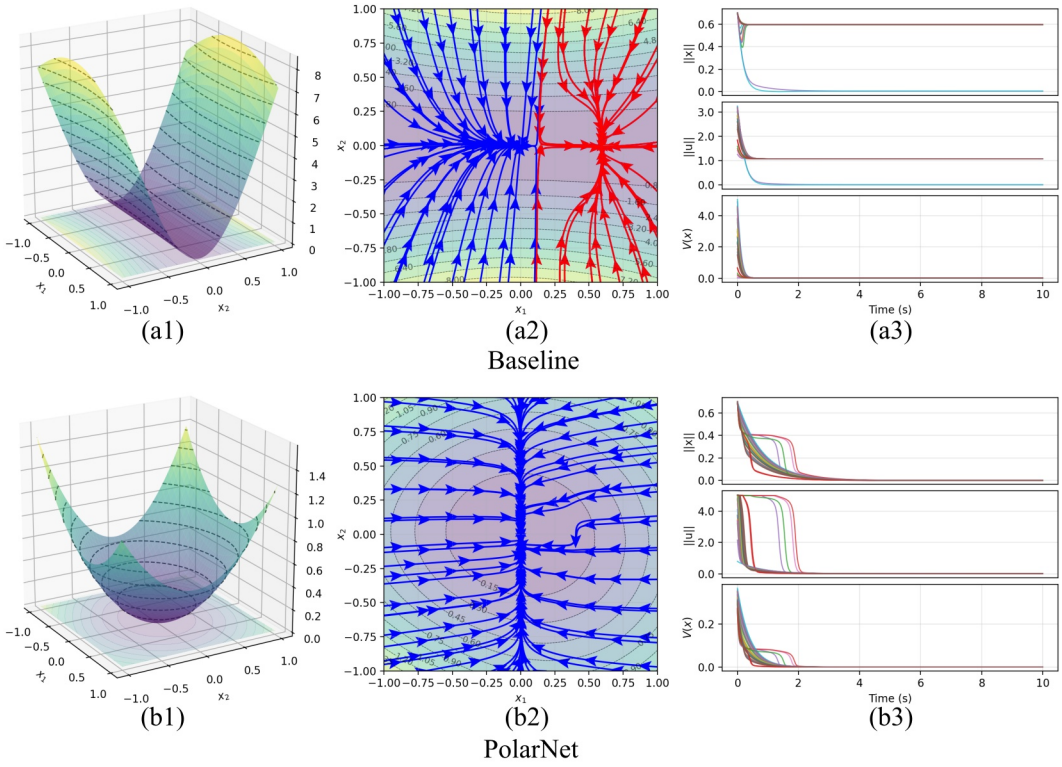

Figure 5: Controller synthesis result with different network architectures: (a1), (b1): Lyapunov function. (a2), (b2): Phase portrait, unstable trajectories are marked in red. (a3), (b3): 36 numerically simulated trajectories. (a): Using the network architecture as in Wei et al. (2023). (b): Using PolarNet. $V(x)$ and $\|x\|$ trajectories are collected by uniformly sample x_0 on a circle of 0.7 radius centering at the origin. Notice how states evolve along descending directions of V_θ for $x \in (-1, +1) \times (-1, +1)$ in (a) but do not converge to the origin, as analyzed in Section 2.2.

Note that we are adapting the code provided in eg1_inverted_pendulum/train_nominal.py of the repository (https://github.com/shiqingw/Robust-Lyapunov-Prob), and our statespace $\mathcal{X}$ has values in $(-1, +1)$ in both dimensions, rather than $(-\pi, +\pi)$. We limit the controller output to $(-5, +5)$ following the practice of in Wei et al. (2025). All hyperparameters are kept the same as in Wei et al. (2025), except that the cutoff radius (region around the origin where the trajectory is considered converged) is set to 0.1 rather than 0.3, so that it matches the smaller size of valid state space in our experiment. We also report the RoA sizes obtained with baseline and PolarNet in Table 3 (averaged over 20 runs using different random seeds).

Table 3: RoA sizes achieved by baseline (Wei et al., 2025) and PolarNet, relative to area of $\mathcal{X}$. Computed by simulating 100×100 trajectories uniformly sampled on $\mathcal{X}$, until convergence.

Network Architecture	**Roa area relative to $\mathcal{X}$**
Baseline	58.17%
PolarNet	99.78%