IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Safe and Stable Neural Network Dynamical Systems for Robot Motion Planning

Allen Emmanuel Binny, Mahathi Anand, Hugo T.M. Kussaba, Lingyun Chen, *Graduate Student Member, IEEE*, Shreenabh Agrawal, Fares J. Abu-Dakka, *Senior Member, IEEE*, and Abdalla Swikir, *Senior Member, IEEE*

***Abstract*—Learning safe and stable robot motions from demonstrations remains a challenge, especially in complex, nonlinear tasks involving dynamic, obstacle-rich environments. In this letter, we propose Safe and Stable Neural Network Dynamical Systems S^2-NNDS, a learning-from-demonstration framework that simultaneously learns expressive neural dynamical systems alongside neural Lyapunov stability and barrier safety certificates. Unlike traditional approaches with restrictive polynomial parameterizations, S^2-NNDS leverages neural networks to capture complex robot motions, providing probabilistic guarantees through split conformal prediction in learned certificates. Experimental results in various 2D and 3D datasets—including LASA handwriting and demonstrations recorded kinesthetically from the Franka Emika Panda robot—validate the effectiveness of S^2-NNDS in learning robust, safe, and stable motions from potentially unsafe demonstrations.**

***Index Terms*—Learning from demonstration, motion and path planning, formal methods in robotics and automation.**

I. Introduction

LEARNING from Demonstration (LfD) is a paradigm in robotics that teaches robots the essence of a task through multiple demonstrations [1]. This enables a robot to generalize a learned task to new environments, allowing safe navigation under perturbations. A key advantage of LfD is its accessibility to end-users without technical expertise, as it requires minimal time and effort to adapt robot's capabilities to diverse environments. A prominent approach in LfD is to encode demonstrations as stable Dynamical System (DS) [2]. This involves recording the robot's end effector or joint positions and velocities and then solving an optimization problem to find a stable DS that accurately replicates these demonstrations. Although the demonstrations capture the essence of the task, the inherent stability of DS ensures generalization and robust performance under perturbations, enhancing data efficiency by reducing the need for additional demonstrations.

However, a critical limitation of many LfD methods is that the optimization problem generally does not consider the obstacles that may be placed in the environment. Consequently, the resulting robot motion may not be safe. This highlights the need to integrate safety constraints directly into the LfD framework alongside stability guarantees.

In this letter, we propose a novel framework for learning a DS that is not only stable, ensuring convergence to a target equilibrium point under perturbations, but also safe, guaranteeing avoidance of *static obstacles* in the environment. Our learning-based algorithm simultaneously identifies neural network-based DS that mimics the provided demonstrations while co-synthesizing formally verified neural stability and safety certificates, characterized by Lyapunov and barrier functions, respectively. The entire DS is learned offline, avoiding the need for any real-time modification of DS.

A. Related Work

Dynamical properties like stability and safety can be certified without explicit trajectory computation utilizing the so-called *certificates*. Particularly, *Lyapunov functions* can prove the asymptotic stability of equilibrium points of a system [3], while *barrier functions* can ensure that a system avoids undesirable states [4]. A common strategy is to synthesize these certificates concurrently with the DS [5], [6], [7], [8], [9].

A standard approach for this synthesis is Sum-Of-Squares (SOS) optimization [10], which reformulates the stability and/or safety constraints as SOS constraints [11], [12], [13]. However, SOS techniques are generally limited to systems with polynomial dynamics and can be overly conservative. In fact, there

Received 13 July 2025; accepted 16 December 2025. Date of publication 19 January 2026; date of current version 28 January 2026. This article was recommended for publication by Associate Editor T. Berrueta and Editor S. J. Chung upon evaluation of the reviewers' comments. This work was supported in part by German Research Foundation, in part by Deutsche Forschungsgemeinschaft (DFG) through Germany's Excellence Strategy – EXC 2050/1 – under Project 390696704, in part by the Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop", in part by (CeTI) of Technische Universität Dresden, in part by the Bavarian State Ministry for Economic Affairs, and in part by Regional Development and Energy (StMWi) through the Lighthouse Initiative KI.FABRIK Bayern Phase 1: Aufbau Infrastruktur. *(Allen Emmanuel Binny and Mahathi Anand contributed equally to this work.) (Corresponding Author: Mahathi Anand.)*

Allen Emmanuel Binny is with the Indian Institute of Technology Kharagpur, Kharagpur 721302, India.

Mahathi Anand and Lingyun Chen are with the Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, 80333 Munich, Germany (e-mail: mahathi.anand@tum.de).

Hugo T.M. Kussaba is with the University of Brasilia, Brasília 70910-900, Brazil.

Shreenabh Agrawal is with Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Fares J. Abu-Dakka is with Mechanical Engineering Program at New York University Abu Dhabi, Abu Dhabi, UAE.

Abdalla Swikir is with the Mohamed bin Zayed University of Artificial Intelligence (MBZUAI), Abu Dhabi, UAE.

The source code, supplementary material and experiment videos can be accessed via https://github.com/allemmbinn/S2NNDS.

Digital Object Identifier 10.1109/LRA.2026.3655207

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

are globally asymptotically stable polynomial DSs that lack a polynomial Lyapunov function [14]. Furthermore, high-degree polynomials in SOS programs are prone to numerical problems, such as floating point errors and poorly conditioned problem formulations, which can undermine the reliability of the results [15].

Neural network (NN)-based certificate synthesis offers a scalable alternative to the limitations of SOS-based approaches [16], [17], [18], [19]. By parameterizing certificates as NNs, these methods take advantage of the universal approximation properties of NNs [20] to find less conservative certificates for more complex motions [21]. However, learning-based certificate synthesis introduces intrinsic errors that can compromise formal guarantees. Thus, formal verification of the learned certificates is necessary, e.g., using Satisfiability Modulo Theories (SMT) solvers [16], [17], [22] or using Lipschitz continuity arguments [18]. However, these approaches are either not scalable to deep neural networks or are highly conservative and require dense grid-based sampling of the domains of interest. Recently, *conformal prediction* [23] has emerged as a promising, sample-efficient technique for providing Provably Approximately Correct (PAC) guarantees for the formal verification of learning-based algorithms, including applications in safe planning [24], formal abstractions [25], and reachability [26]. Conformal prediction was also used in the context of control barrier functions in [27]. This approach can quantify the validity of stability and safety constraints with high sample efficiency, which positions it as a promising method for the formal verification of learned certificates.

Despite the progress in NN-based certificate learning, few works have explored integrating it with the LfD framework. Most existing works in LfD either address only stability [5], [6], [7], [28], [29] or consider obstacles in an online setting [9], [30], [31], [32]. These approaches modulate DS in real-time to deal with dynamic obstacles; however, altering the DS can potentially deviate the robot's motions from initial demonstrations. Furthermore, while [9] considers an NN-based approach to learn safe and stable DS, they do not synthesize any certificates and consider them fixed and known, resulting in conservatism. Moreover, their approach is affected by potential conflicts between stability and safety constraints, resulting in compromise between safety and stability. Finally, because robots operate under strict real-time control frequency constraints, incorporating online optimization within the control loop significantly increases the computational burden. The most closely related work to our letter is [8], where the authors proposed obstacle Avoidance with Barrier-Certified polynomial Dynamical Systems (ABC-DS), an SOS-based approach to identify safe and stable DS against static obstacles.

B. Contributions

This letter makes the following contributions:

1) We propose S^2-NNDS, a novel *offline* approach to generate time-invariant, safe, and stable DSs directly from demonstrations without requiring online modifications, alleviating the aforementioned challenges. S^2-NNDS jointly learns the DS and its corresponding neural certificates, even from unsafe demonstrations.
2) Our approach uses expressive neural *network representations* and incorporates *split conformal prediction* to provide probabilistic guarantees on the correctness of the learned stability and safety certificates. An overview is illustrated in Fig. 1.
3) We validate S^2-NNDS through experiments in 2D and 3D environments, including a publicly available handwriting dataset and kinesthetic demonstrations collected using a Franka Emika Panda robot and compare them with [8]. The results show that our method reliably learns safe and stable motions with minimal restrictions on the obstacle configurations, and is competitive with existing approaches.

II. Preliminaries

A. Notations

Let $\mathbb{R}$ and $\mathbb{N}$ denote the sets of real and non-negative integers, respectively. The subscripts specify subsets of $\mathbb{R}$ and $\mathbb{N}$; e.g., $\mathbb{R}_{\geq 0}$ is the set of non-negative reals. We use $\mathbb{R}^n$ for a real space of n-dimension and $\mathbb{R}^{m\times n}$ for the space of $m \times n$ real matrices. The cardinality of a finite set X is denoted by $|X|$. For a vector (denoted in lowercase) $x \in \mathbb{R}^n$, x_i is its i^{th} element, $1 \leq i \leq n$. The Euclidean norm of x is $\|x\|$. For a differentiable function $f : X \to Y$, where sets $X, Y \subseteq \mathbb{R}^n$, and ∇f denotes its gradient, i.e., $\nabla f(\mathbf{x}) = (\partial f/\partial x_1(x), \ldots, \partial f/\partial x_n(x))^\top$. The notation $\circ$ and id denote the composition of functions and the identity function, respectively. The indicator function $\mathbb{I}_X(x)$ is 1 if $x \in X$ and 0 otherwise. The ceiling and floor functions are $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$, respectively. The regularized incomplete Beta function is $\mathcal{I}_c(a,b) = \frac{\int_0^c t^{a-1}(1-t)^{b-1}\mathrm{d}t}{B(a,b)}$, where $B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}\mathrm{d}t$ is the beta function. Lastly, $\mathbf{x}$ (denoted in bold letters) represents both continuous trajectories and distinct sequences. The distinction will become clear from the context.

B. DS for Motion Planning

The LfD framework for robot motion planning aims to learn a DS described by the following differential equation

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)), \tag{1}$$

where $\mathbf{x}(t) \in X$ is the state variable (e.g., Cartesian position of the robot's end-effector within its workspace X) at $t \in \mathbb{R}_{\geq 0}$. $f : X \to X$ is a continuous non-linear function that describes the state velocity of the robot. We assume that (1) has a unique solution for every initial condition $\mathbf{x}(0) \in X$.

Given a set of N demonstrations, $\mathcal{D} = \{(x_{ij}, \dot{x}_{ij})\}_{i=1,j=1}^{N,M}$, where each demonstration consists of M time-discretized positions $\mathbf{x}_i = (\mathbf{x}_i(t_1), \mathbf{x}_i(t_2), \ldots, \mathbf{x}_i(t_M))$ and corresponding velocities $\dot{\mathbf{x}}_i = (\dot{\mathbf{x}}_i(t_1), \dot{\mathbf{x}}_i(t_2), \ldots, \dot{\mathbf{x}}_i(t_M))$, $t_0 = 0, t_M > 0, i \in \{1, \ldots, N\}$. The goal of this work is to learn a function f such that the resultant DS in (1) not only approximates the demonstrations $\mathcal{D}$ but also converges to a goal state $x^\star$ while avoiding obstacles in X. Under nominal conditions, i.e., in the absence of external disturbances or obstacles, one can learn a DS

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

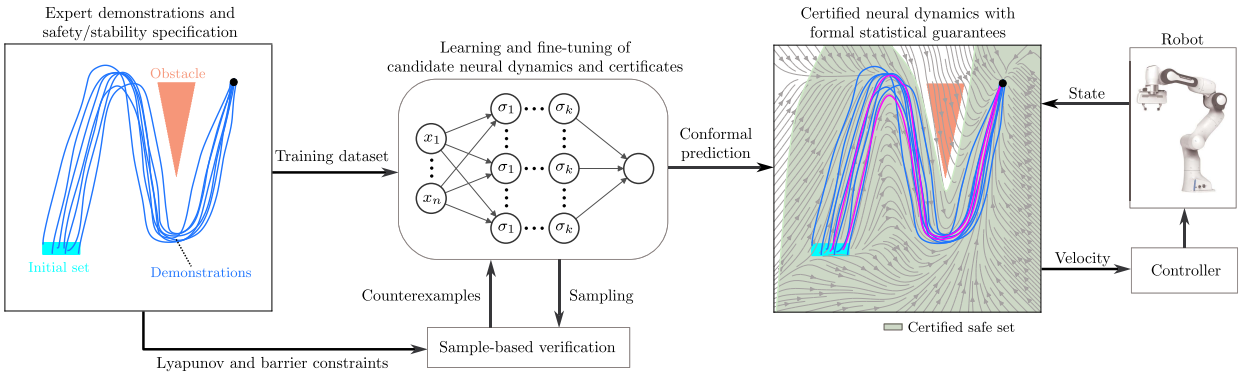

Fig. 1. Overview of Safe and Stable Neural Network Dynamical Systems (S^2-NNDS) Framework. Neural dynamics are first learned from expert demonstrations and iteratively refined to satisfy Lyapunov and barrier constraints using counterexamples. Verification with conformal prediction then provides formal statistical guarantees on safety and stability.

simply by minimizing the Mean Square Error (MSE) between f and $\dot{\mathbf{x}}$, i.e.,

$$\min_f \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \|\dot{x}_{ij} - f(x_{ij})\|^2 \tag{2}$$

However, since f is learned using only a finite number of demonstrations, the DS is not robust, i.e., under any external perturbations during deployment, the trajectory may not converge to the required goal $x^\star$. In addition, since obstacles are not considered during training, the resulting trajectories may not be safe. Therefore, to ensure robust convergence and obstacle avoidance, we must incorporate formal certificates of stability and safety. It is well-established that the stability of a DS can be certified using Lyapunov functions [3], which are formulated below.

Proposition 1 (Asymptotic Stability): Consider the DS in (1) with an isolated equilibrium point $x^\star \in X$. A continuously differentiable real-valued function $V : X \to \mathbb{R}$ is a *Lyapunov function*, where $V(x^\star) = 0, \dot{V}(x^\star) = 0$, if for all $x \in X \setminus \{x^\star\}$ the following conditions are satisfied:

$$V(x) > 0, \quad \dot{V}(x) = \nabla V(x)^T f(x) < 0. \tag{3}$$

The existence of a Lyapunov function guarantees that the system is locally asymptotically stable within X and $\exists X_i \subseteq X$ s.t. $\forall \mathbf{x}(0) \in X_i, \exists t \in \mathbb{R}_{\geq 0}$ with $\mathbf{x}(t) = x^\star$.
The proof is standard and a direct application of the Lyapunov theory [3, Theorem 4.1], and is thus omitted. Similarly, barrier functions were first introduced in [4] to certify the safety of DSs.

Proposition 2 (Safety): Consider the DS in (1) with an initial set $X_0 \subseteq X$, i.e., $\mathbf{x}(0) \in X_0$ and an unsafe set $X_u \subseteq X$. A continuously differentiable real-valued function $\mathrm{B} : X \to \mathbb{R}$ is a *barrier function* if $\exists\, \varepsilon > 0$ such that:

$$\mathrm{B}(x) \leq 0, \qquad \forall x \in X_0, \tag{4}$$

$$\mathrm{B}(x) > 0, \qquad \forall x \in X_u, \tag{5}$$

$$\nabla B(x)^\top f(x) \leq 0, \qquad \forall x \in \{x \in X \mid |B(x)| \leq \varepsilon\}. \tag{6}$$

The existence of a barrier function guarantees that the system is safe, i.e., $\mathbf{x}(t) \notin X_u$ for all $t \in [0, T]$ for any $T \in \mathbb{N}$.
The proof of Proposition 2 can be found in [8].

Learning a DS while guaranteeing stability and safety can, therefore, be expressed by minimizing the MSE in (2) subject to the stability and safety conditions defined by Propositions (3)-(6). Specifically, this can be formulated as follows[1]:

$$\begin{aligned} \min_f \; & \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \|\dot{x}_{ij} - f(x_{ij})\|^2 \\ \text{s.t. } & f(0) = 0, \\ & \text{conditions from Prop. (3)}-\text{(6) hold.} \end{aligned} \tag{7}$$

Finding a solution to (7) is computationally intractable, since it requires optimizing in infinite-dimensional function spaces to find functions f, V, and B. In practice, one needs to obtain suitable parameterizations of the functions. Conventional approaches consider linear parameter-varying dynamics with quadratic Lyapunov functions [5], [6], [7] or polynomial dynamics with polynomial certificate functions [8]. However, these methods have significant drawbacks. The rigid structure of these parameterizations limits the flexibility needed to represent highly nonlinear and complex motions. They also lack the versatility to handle obstacles with complex shapes, e.g., [8] only considers semi-algebraic sets. Furthermore, these parameterizations often lead to highly complex, possibly nonconvex, optimization problems that are computationally demanding to solve. To alleviate these challenges, we propose a neural network parameterization of the above optimization problem. We characterize the functions f, V, and B as NNs and learn them simultaneously by constructing an appropriate loss functions to enforce (7).

III. Proposed Methodology

In this section, we introduce our Safe and Stable Neural Network Dynamical Systems (S^2-NNDS) framework to solve the optimization problem in (7). The core idea is to represent the DS and the certificate functions using NNs and train them concurrently using carefully designed loss functions. Once the

[1]Without loss of generality, we assume $x^\star = 0$, which can be achieved by translating the coordinate system. We also assume the demonstrations end at the equilibrium, i.e., $(\boldsymbol{x}_{iM}, \dot{\boldsymbol{x}}_{iM}) = (0, 0)$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

networks are trained, we use conformal prediction to validate the learned certificates.

A. Neural Network Parameterizations

We parameterize the function f in (1) with a fully connected neural network of K layers, denoted $f_\theta(x)$. The NN consists of $n_0 = n$ inputs, $n_k = n$ outputs, and $K-2$ hidden layers, with each hidden layer $i \in \{1, \dots, K-2\}$ containing n_i hidden neurons. More precisely, consider

$$f_\theta(x) = l_{K-1} \circ \dots \circ l_0(x), \tag{8}$$

where each layer $l_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ is defined as

$$l_i(x) = \sigma_i(W_i x + b_i), \quad \forall i \in \{0, \dots, K-1\}.$$

Here, $W_i \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b_i \in \mathbb{R}^{n_{i+1}}$ are the weight matrix and the bias vector, respectively, collectively denoted by the parameters $\theta = [W; b]$. $\sigma_i : \mathbb{R}^{n_{i+1}} \to \mathbb{R}^{n_{i+1}}$ represents the element-wise application of an appropriate activation function (e.g., ReLU, Tanh, ELU, etc.). To ensure $f_\theta(0) = 0$ as required in (7), we set all bias terms $b_i = 0, \forall i \in \{0, \dots, K-1\}$.

The certificate functions V and B are also parameterized as NNs, $V_{\theta'}$ and $\mathrm{B}_{\theta''}$, similarly to (8), with parameters θ' and θ'', respectively. However, some distinctions are made to enforce certain properties on these functions. For example, to ensure $V_{\theta'}(0) = 0$ we set the bias terms in $V_{\theta'}$ to 0 according to Proposition 1. No such restriction is placed on $\mathrm{B}_{\theta''}$. Moreover, to ensure continuous differentiability of the certificates (as required for conditions (3) or (6)), differentiable activation functions are considered for both $V_{\theta'}$ and $\mathrm{B}_{\theta''}$. Finally, we set $\sigma_{K-1} = \mathrm{id}$ for these networks.

To train these neural networks, we define a set of loss functions. The primary loss is the MSE loss from (2), augmented by the incorporation of the constraints (3)–(6). Given a demonstration set $\mathcal{D}$ consisting of $|\mathcal{D}| = MN$ elements of robot's positions and velocities $(x, \dot{x}) \in \mathcal{D}$ (possibly shuffled), the loss function corresponding to the objective function is formally defined as the MSE between the demonstrated velocities $\dot{x}$ and the predicted velocities of the neural network function $f_\theta(x)$:

$$L_{MSE}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,\dot{x}) \in \mathcal{D}} \|\dot{x} - f_\theta(x)\|^2. \tag{9}$$

To construct a candidate Lyapunov function $V_{\theta'}$, we first sample a set of i.i.d samples from the workspace X as $\mathcal{S} = \{x_i \in X \mid i \in \{1, \dots, S\}\}$. We then define a hinge-like loss using a leaky ReLU function in [16]. Then, the loss function corresponding to condition (3) is obtained as

$$\begin{aligned} L_{lyap}(\theta, \theta') = \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} & (\lambda_{l1} \mathrm{LR}_\alpha(\delta_{l1} - V_{\theta'}(x))) \\ & + \lambda_{l2} \mathrm{LR}_\alpha(\nabla V_{\theta'}(x) f_\theta(x) - \delta_{l2}), \end{aligned} \tag{10}$$

where LR_α is the leaky ReLU function parameterized by $\alpha \geq 0$, $\lambda_{l1}, \lambda_{l2}$ are the loss weights and δ_{l1}, δ_{l2} are small positive tolerances for numerical stability. This loss $L_{lyap}(\theta, \theta')$ penalizes violations of the Lyapunov conditions (3) while promoting them to take on values where (3) is satisfied with a large margin.

Similarly, for the barrier function $\mathrm{B}_{\theta''}$, we sample points from the initial $\mathcal{S}_0 = \mathcal{S} \cap X_0$ and the unsafe set $\mathcal{S}_u = \mathcal{S} \cap X_u$. Moreover, to enforce condition (6), an additional term is included in the loss function.[2] Consequently, the loss function for the barrier certificate is defined as

$$\begin{aligned} L_{bar}(\theta, \theta'') = & \frac{1}{|\mathcal{S}_0|} \sum_{x \in \mathcal{S}_0} \lambda_{b2} \mathrm{LR}_\alpha(\mathrm{B}_{\theta''}(x) + \delta_{b2}) \\ & + \frac{1}{|\mathcal{S}_u|} \sum_{x \in \mathcal{S}_u} \lambda_{b3} \mathrm{LR}_\alpha(\delta_{b3} - \mathrm{B}_{\theta''}(x)) \\ & + \frac{1}{|\mathcal{S}|} \sum_{x \in \mathcal{S}} \lambda_{b1} \mathrm{LR}_\alpha(\nabla \mathrm{B}_{\theta''}(x) f_\theta(x) - \delta_{b1}), \end{aligned} \tag{11}$$

where $\lambda_{b1}, \lambda_{b2}, \lambda_{b3}$ represent the weighting coefficients for each term, and $\delta_{b1}, \delta_{b2}, \delta_{b3}$ are the corresponding tolerance parameters. Note that these weights, tolerance parameters, and α are hyperparameters that must be predetermined.

Minimizing the loss functions described in (9)–(11) aims to yield a suitable DS along with candidate stability and safety certificates for the optimization problem (7). However, minimization of losses such as (3) and (11) does not inherently guarantee the satisfaction of conditions (3)–(6) throughout the entire continuous domain X. Furthermore, even over the sampled dataset, strict satisfaction guarantees are not achieved, as positive loss terms can be canceled out by larger negative terms. Consequently, the functions obtained through this training process offer neither empirical nor formal guarantees on their validity. To address this limitation, we propose to leverage conformal prediction techniques to obtain formal PAC-like guarantees on the validity of safety and stability certificates.

B. Validation of Certificates via Conformal Prediction

Conformal prediction [33] is an uncertainty quantification technique that is predominantly used in the context of machine learning to obtain statistically rigorous confidence intervals for model predictions. Its core principle involves assigning nonconformity scores to a calibration set of independently drawn data samples. These scores are then used to obtain reliable predictions for new test data drawn from the same underlying distribution. In this letter, inspired by [25], [26], [27], we propose a split conformal prediction-based algorithm. This algorithm aims to establish probabilistic lower bounds on the satisfaction of conditions (3)–(6) by the learned candidate certificates $V_{\theta'}$ and $\mathrm{B}_{\theta''}$ across the entire workspace X.

To formalize this, consider a learned DS f_θ as in (1), alongside the certificate functions $V_{\theta'}$ and $\mathrm{B}_{\theta''}$. We define a calibration set $\mathcal{C} = \{x_i \in X \mid i \in \{1, \dots, N_{ver}\}$ consisting of N_{ver} i.i.d samples. For each $x \in \mathcal{C}$, we compute a nonconformity score function as $s(x) = \max_{q \in \{1,\dots,5\}} \rho_q(x)$, which is defined as the maximum violation across all relevant conditions; where $\rho_q(x)$ represents the violation of the q-th condition. Specifically, $\rho_1(x), \rho_2(x)$ correspond to the stability conditions in (3)

[2]The last loss term is applied over the entire domain X rather than just the region $\{x \in X \mid |B(x)| \leq \varepsilon\}$ to enhance robustness during training.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Algorithm 1: Certificate Verification using Conformal Prediction.

function CP$f_\theta, V_{\theta'}, \mathrm{B}_{\theta''}, N_{ver}, \epsilon$
  verified $\leftarrow$ **false**
  $\mathcal{C} \leftarrow$ Sample N_{ver} i.i.d states from X
  $S = \emptyset$
  for $x \in \mathcal{C}$ **do**
    S.insert(s(x))
  $S \leftarrow$ Sort S in non-decreasing order
  $\alpha, \beta \leftarrow$ Solve $\mathcal{I}_{1-\epsilon}(N_{ver} - l + 1, l) \leq \beta$
  $p \leftarrow \frac{\lceil (N_{ver+1}+1)(1-\alpha) \rceil}{N_{ver}}{}^{th}$ quantile of S
  if $p \leq 0$ **then**
    verified $\leftarrow$ **true**
  return $p, 1-\beta$, verified

formulated such that $\rho_q(x) \leq 0$ indicates satisfaction. Similarly, $\rho_q(x)$ for $q \in \{3,4,5\}$ can be expressed in an analogous form to represent violations of the barrier conditions (6). For instance, $\rho_3(x) = \mathrm{B}(x)\mathbb{I}_{X_0}(x)$ could represent a violation related to the initial set. A higher value of $\rho_q(x)$ directly signifies a greater violation of the corresponding condition (3)–(6). Consequently, obtaining $s(x) \leq 0$ for any randomly drawn test point $x \in X$ would rigorously validate the certificate functions. Building upon this intuition, we present the following theorem, adapted from [26], [27], which provides a quantifiable measure of satisfaction for conditions (3)–(6).

Theorem 3 (Verification via Conformal Prediction): Let f_θ be a learned DS and $V_{\theta'}$ and $\mathrm{B}_{\theta''}$ be the corresponding candidate Lyapunov and barrier functions, respectively. Consider a calibration set $\mathcal{C} = \{x_1, \ldots, x_{N_{ver}}\}$ consisting of N_{ver} i.i.d. samples. Given confidence levels $\alpha, \beta, \epsilon \in (0,1)$ such that the following condition holds for the regularized incomplete beta function $\mathcal{I}_{1-\epsilon}(N_{ver} - l + 1, l) \leq \beta$, where $l = \lfloor (N_{ver})(\alpha) \rfloor$. Then, with a confidence of at least $1-\beta$, the probability that a randomly drawn test point $x \in X$ satisfies $s(x) \leq p$ is at least $1-\epsilon$:

$$\mathbb{P}_{x \in X}(s(x) \leq p) \geq 1 - \epsilon, \tag{12}$$

where p is the quantile, $\frac{\lceil (N_{ver+1}+1)(1-\alpha) \rceil}{N_{ver}}{}^{th}$, of the scores $s(x), \forall x \in \mathcal{C}$.

The proof of Theorem 3 is analogous to the methodology presented in [26]. $p \leq 0$ indicates the satisfaction of conditions (3)–(6) for some sufficiently high confidence levels $(1-\beta)$ and $(1-\epsilon)$, while $p > 0$ indicates the presence of safety violations [27]. Therefore, to obtain a statistically significant guarantee on the validity of the safety and stability certificates, it is imperative to ensure that $p \leq 0$. For a more intuitive explanation of formal verification via conformal prediction, we refer the readers to [26]. The general algorithm to verify the validity of these safety conditions is detailed in Algorithm 1.

C. The S^2-NNDS Algorithm

The S^2-NNDS algorithm for learning safe and stable DS is structured in two distinct phases: training and verification. The training phase begins with an initial training of f_θ using the MSE loss (9) on the demonstration dataset $\mathcal{D}$. However, this initial dynamics model does not inherently guarantee safety or stability, as the corresponding certificates may not yet exist. Consequently, f_θ is further fine-tuned within a counterexample guided scheme, concurrently learning the Lyapunov and barrier certificate functions. This is done by training networks with respect to composite loss functions (9)–(11) with the aim of achieving verification guarantees based on samples with minimal data. Initially, a small finite sample dataset $\mathcal{S} \subseteq X$ is established, and NNs $f_\theta, V_{\theta'}$ and $\mathrm{B}_{\theta''}$ are trained simultaneously for a specified number of epochs. Following this, a few counterexamples from a randomly generated larger finite set $\mathcal{S}_{cex} \subset X$ that violate conditions (3)–(6) are iteratively added to $\mathcal{S}$ and the networks are re-trained until no further counterexamples are found. At this stage, only statistical verification guarantees are available for the $|\mathcal{S}_{cex}|$ samples. To obtain stronger formal statistical guarantees on the trained candidate certificates, Algorithm 1 is applied as a posterior verification step. The general algorithm is formalized in Algorithm 2.

Algorithm 2: The S^2-NNDS Algorithm.

Require: $X, X_0, X_u, \mathcal{D}, \mathcal{S}$,
N_{cex}, N_{ver}, epochs, iters, ϵ
$\mathcal{S}_0, \mathcal{S}_u \leftarrow \mathcal{S} \cap X_0, \mathcal{S} \cap X_u$
Train f_θ subject to loss (9)
Initialize $V_{\theta'}, \mathrm{B}_{\theta''}$
for $i \leq iters$ **do**
  for $j \leq epochs$ **do**
    Train $f_\theta, V_{\theta'}, \mathrm{B}_{\theta''}$ w.r.t. losses (9)–(11)
  $\mathcal{S}_{cex} \leftarrow$ Sample N_{cex} i.i.d states from X
  $no_{cex}, \mathcal{S}_{viol} \leftarrow$ number and set of counterexamples violating conditions (3)–(6)
  if $no_{cex} = 0$ **then**
    break
  else
    $\mathcal{S}_0 \leftarrow \mathcal{S}_0 \wedge \mathcal{S}_{viol}, \mathcal{S}_u \leftarrow \mathcal{S}_u \wedge \mathcal{S}_{viol}, \mathcal{S} \leftarrow \mathcal{S} \wedge \mathcal{S}_{viol}$
if $no_{cex} = 0$ **then**
  $p, 1-\beta$, verified $\leftarrow$ CP$(f_\theta, V_{\theta'}, \mathrm{B}_{\theta''}, N_{ver}, \epsilon)$
  return $f_\theta, V_{\theta'}, \mathrm{B}_{\theta''}, p, 1-\beta$, verified
return none

IV. Simulations and Experimental Results

Our experimental evaluation of the S^2-NNDS algorithm is organized into the following categories:

1) The performance of our algorithm is illustrated on a representative subset of the 2D LASA handwriting datasets[3] as well as a 3D dataset[4] in an environment with single obstacles,
2) We provide a detailed performance analysis and benchmark comparisons of our algorithm with the ABC-DS algorithm [8] for the case of 2D handwriting datasets,

[3]Publicly available at https://bitbucket.org/khansari/lasahandwritingdataset
[4]Available at https://github.com/nbfigueroa/ds-opt

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

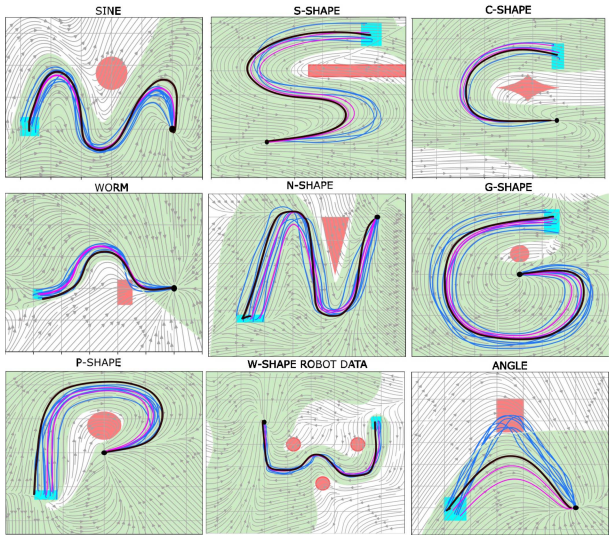

Fig. 2. Neural DS generated by our proposed approach with obstacles for the LASA handwriting and robot demonstration datasets. Five demonstrations (blue) were used. The learned trajectories (pink) are simulated for two initial conditions within the initial set, and the robot path (brown) is obtained for another initial point. The region in green describes the safe set, while the arrows indicate the flow of the DS.

3) We evaluate the performance of demonstrations recorded kinesthetically by the Franka Emika Panda robots [34] in the presence of multiple obstacles.

a) 2D and 3D Datasets: First, we consider 8 datasets corresponding to different shapes in the LASA handwriting dataset, which is a standard benchmarking dataset used in the literature [7], [8], [9], and offers a simplification to several industrial tasks such as cutting, carving, and welding [35]. The data is normalized to reside within $X = [-1, 1]^2$, and without loss of generality, it is assumed that the attractor point coincides with the origin. The neural networks corresponding to $f_\theta, V_{\theta'}$ and $\mathrm{B}_{\theta''}$ are trained and certified using Algorithm 2 with high confidence levels, specifically $1-\epsilon, 1-\beta \geq 0.99$. The simulation results presented in Fig. 2 show multiple key advantages of the S^2-NNDS approach.[5] From the evaluation: (*i*) our approach can handle arbitrarily complex obstacle configurations without imposing any restrictions on the shape's convexity, star-shapedness, or semialgebraicity, as required in [8], [30]. This is illustrated by a diamond-like obstacle that is considered for the C-Shape dataset. In particular, our algorithm only requires an analytical closed-form expression for the unsafe set. (*ii*) S^2-NNDS can generate safe trajectories even with potentially unsafe demonstrations, as demonstrated by the Angle and Worm datasets. Here, the learned trajectories closely resemble the shape of the original demonstrations for all the initial conditions, though the dynamics have been modified to ensure safety. (*iii*) the sub-zero-level set $\{x \in \mathbb{R}^n \mid \mathrm{B}(x) \leq 0\}$ of the barrier function, which defines the *safe set*, outlined by the green regions in the figures, fit tightly between the obstacles and the demonstrations, showing that the certificates learned by our approach are non-conservative. As long as the robot lands in the green region when perturbed, safe navigation is formally guaranteed.

[5]The models used in the figures were obtained on an Ubuntu 20.04 LTS system with 16GB RAM equipped with NVIDIA GeForce RTX 4050 - 6GB GPU.

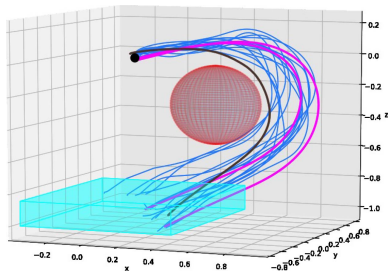

Fig. 3. Neural DS generated by our proposed approach with obstacles for the 3D C-shaped motion. Ten demonstrations were used. The legend follows from Fig. 2.

Furthermore, we extend our evaluation to a 3D C-shaped motion. Consistent with the 2D results, and with very high confidence levels of $1-\epsilon, 1-\beta \geq 0.99$, we find that the learned trajectories are not only safe and stable, but also closely mimic the demonstrations, as shown in Fig. 3. However, as is evident from the figure, a larger number of demonstrations are required to learn safe, generalizable trajectories due to the curse of dimensionality.

b) Performance Analysis and Benchmark Comparisons: The performance of our S^2-NNDS algorithm is evaluated by measuring the mean, the standard deviation (SD) of the mean square error (MSE), and dynamic time warping (DTW) distance between the demonstrations and the learned trajectories (i.e., the value of our objective function (2)), on a set of test data corresponding to the LASA handwriting sets. While MSE measures the average squared, point-point distance between two trajectories, DTW measures the optimal alignment between two trajectories [36]. Therefore, MSE provides a good measure for analyzing how close the DS predicts motion w.r.t to the demonstrations, while DTW measures the similarity between the path shapes. First, we compare the results obtained in Fig. 2 with ABC-DS in [8], which characterizes the dynamical system as well as certificates as polynomial functions and solves a bilinear optimization problem [8] using PENBMI, a proprietary, commercial solver to solve bilinear problems. However, since we observed that PENBMI performs reliably only with specifically crafted semi-algebraic sets with at most one constraint (e.g. a hyper-rectangular initial set $a_i \leq x_i \leq b_i,\ i \in \{1, 2\}$ requires four semi-algebraic constraints), we utilized ellipsoidal approximations for any non-ellipsoidal sets. The obtained results are provided in Table I.[6] Then, we also consider the LASA handwriting datasets and the obstacle configurations utilized in [8]. The obtained metrics are provided in Table II.

The results show that the performance of our approach is competitive to ABC-DS, especially in cases involving highly nonlinear motion shapes with high rate of change in motion, and complex obstacle configurations, where it demonstrates superior

[6]The ABC-DS results were obtained after executing the algorithm for a maximum of 100 iterations and validated with numerical tolerance (primal residual) of 0.001, as the code consistently terminated with a maximum iteration error.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE I
MSE, SD AND DTW OF THE LEARNED TRAJECTORIES FOR THE LASA HANDWRITING DATASETS CORRESPONDING TO FIG. 2, COMPARED ALSO WITH ABC-DS [8]. NOTE THAT ELLIPSOIDAL APPROXIMATIONS FOR THE INITIAL AND OBSTACLE CONFIGURATIONS WERE UTILIZED FOR ABC-DS, AND −− DENOTES CASES WHERE NO SATISFACTORY RESULTS WERE OBTAINED.

	Sine	S-Shape	C-Shape	Worm	Angle	G-Shape	P-Shape	N-Shape
MSE: S^2-NNDS	0.015	0.016	0.019	0.026	0.044	0.019	0.008	0.084
MSE: ABC-DS	0.023	0.014	0.029	−−	−−	0.041	0.027	0.020
SD: S^2-NNDS:	0.065	0.063	0.077	0.074	0.109	0.080	0.045	0.1937
SD: ABC-DS:	0.0762	0.061	0.096	−−	−−	0.108	0.0705	0.081
DTW: S^2-NNDS:	0.062	0.234	0.288	0.071	0.341	0.737	0.667	0.157
DTW: ABC-DS:	0.132	0.336	0.381	−−	−−	0.8617	0.95	0.190

TABLE II
COMPARISONS OF MSE, SD AND DTW OF THE LEARNED TRAJECTORIES FOR THE OBSTACLE CONFIGURATIONS IN [8]

	Sine	S-Shape	Worm	P-Shape
MSE: S^2-NNDS	0.035	0.014	0.014	0.016
MSE: ABC-DS	0.025	0.012	0.066	0.048
SD: S^2-NNDS	0.11	0.067	0.068	0.06
SD: ABC-DS	0.078	0.060	0.116	0.109
DTW: S^2-NNDS	0.0686	0.119	0.068	0.845
DTW: ABC-DS	0.101	0.521	0.089	0.532

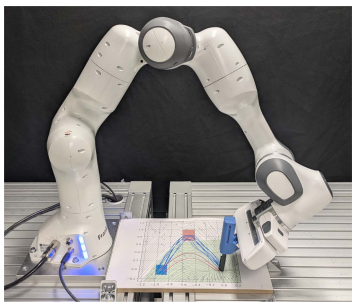

Fig. 4. Drawing platform with Franka Panda Robot.

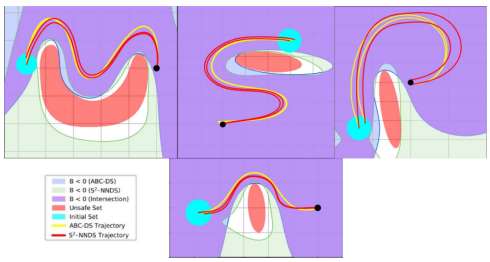

Fig. 5. Comparison of the learned trajectories and barrier functions obtained via S^2-NNDS and ABC-DS, respectively. Barrier functions learned via our approach generally fit more tightly around the obstacles and offer less conservative results, as can be seen from the area computed in Table III.

TABLE III
QUANTITATIVE AREA COMPARISONS OF THE SAFE REGIONS GENERATED BY S^2-NNDS AND ABC-DS FOR OBSTACLE CONFIGURATIONS USED IN [8]

	Sine	S-Shape	Worm	P-Shape
Safe Area: S^2-NNDS	2.02	3.871	3.865	3.59
Safe Area: ABC-DS	2.68	3.768	3.101	3.13

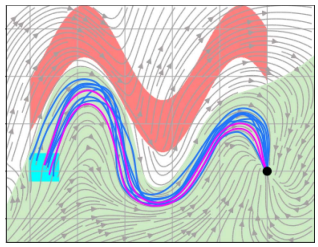

Fig. 6. Sine-like motion learned in a cluttered environment with a large sine-shaped obstacle. While S^2-NNDS produces reasonable motion, ABC-DS leads to infeasibility as any computationally tractable semi-algebraic approximation (e.g. ellipsoid) leads to an intersection between the demonstrations and the obstacles.

performance with lower MSEs as well as DTW. Moreover, ABC-DS fails to provide any results when demonstrations are unsafe. Additionally, we observed that the S^2-NNDS algorithm may provide less conservative barrier functions, as illustrated in Fig. 5 and Table III, by the larger safe sets that lie tightly between the demonstrations and obstacles. However, a certain trade-off between MSE, DTW, and conservatism of the barrier function is reasonable. Note that, in general, ABC-DS can only handle semi-algebraic obstacle configurations, whereas our approach is more general and can handle non-semi-algebraic sets. For example, one cannot use diamond-like obstacles with ABC-DS, but our approach indeed handles this case well, as demonstrated by the C-Shaped motion in Fig. 2. We also show a scenario in Fig. 6 where S^2-NNDS succeeds and ABC-DS fails due to the fact that any semi-algebraic approximation (e.g. ellipsoid) leads to infeasibility as demonstrations become unsafe. Thus, S^2-NNDS is more suitable for complex obstacle configurations in tightly-spaced, cluttered environments. However, note that while ABC-DS provides *absolute* guarantees on *global* asymptotic stability, S^2-NNDS only provides *statistical* formal guarantees for *local* asymptotic stability (c.f. Section V).

c) Demonstrations from Franka Emika Robot: To further evaluate the validity of our approach, we tested our approach on kinesthetically recorded demonstrations from the Franka Emika robot [34]. To test the adaptability of S^2-NNDS, we introduced 3 obstacles around the robot's intended path. Once again, the learned trajectories closely follow the demonstrations while avoiding the obstacles, as shown in Fig. 2.

Robot Experiments: To assess the practical feasibility of the proposed framework in a real setup, we utilize the robot drawing system depicted in Fig. 4 to draw shapes from the LASA dataset.[7]

[7]The drawing robot in the drawing system is a 7-DoF Franka Emika robot [34], controlled via the Franka Control Interface at a frequency of 1 kHz. The control loop is executed on a computer equipped with an Intel Core i5-12600 K CPU running Ubuntu 22.04 LTS with a real-time kernel version 5.15.55-rt48.

The robot Z-axis and orientation are governed by an impedance controller to achieve consistent pen-strokes while preventing excessive force on the pen and paper. Additionally, passive interaction control from [37] is utilized to follow the integral curves of the DS generated by our method, enabling drawing along the X and Y axes while maintaining a passive relation between external forces and the robot velocity. The 3D C-shaped motion is also validated on the robot, with the translation axes being controlled by the DS generated by our method and orientation governed by an impedance controller. The resultant motion of the robot for the considered datasets (2D and 3D datasets) is recorded and plotted in Figs. 2 and 3, respectively.

V. Discussion

Despite the strong performance of S^2-NNDS for safe LfD, several limitations remain. Since NNs are trained on bounded domains, S^2-NNDS ensures only local asymptotic stability in a domain X, unlike prior work guaranteeing global stability [5], [8]. Thus, convergence to the origin is only guaranteed from some (unknown) region of attraction; starting or drifting outside this region may break convergence. Nevertheless, enforcing radial unboundedness in the Lyapunov function or careful NN tuning can yield strong empirical stability. Moreover, S^2-NNDS is trained offline and currently handles only static obstacles, although this still covers many practical scenarios such as warehouses. Finally, like most NN-based methods, its performance depends on hyperparameters and on the initial training of f_θ. Addressing these challenges is left for future work.

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2008, pp. 1371–1394.

[2] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Cambridge, MA, USA: MIT Press, 2022.

[3] H. K. Khalil, *Nonlinear System*. Upper Saddle River, NJ, USA: Prentice Hall, 2013.

[4] S. Prajna and A. Jadbabaie, "Safety verification of hybrid systems using barrier certificates," in *Proc. Int. Workshop Hybrid Syst., Comput. Control*, 2004, pp. 477–492.

[5] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian mixture models," *IEEE Trans. Robot.*, vol. 27, no. 5, pp. 943–957, Oct. 2011.

[6] N. Figueroa and A. Billard, "A physically-consistent Bayesian non-parametric mixture model for dynamical system learning," in *Proc. Conf. Robot Learn.*, 2018, pp. 927–946.

[7] N. Figueroa and A. Billard, "Locally active globally stable dynamical systems: Theory, learning, and experiments," *Int. J. Robot. Res.*, vol. 41, no. 3, pp. 312–347, 2022.

[8] M. Schonger et al., "Learning barrier-certified polynomial dynamical systems for obstacle avoidance with robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 17201–17 207.

[9] F. Nawaz, T. Li, N. Matni, and N. Figueroa, "Learning complex motion plans using neural ODEs with safety and stability guarantees," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 17216–17222.

[10] G. Blekherman, P. A. Parrilo, and R. R. Thomas, *Semidefinite Optim. and Convex Algebr. Geometry*. Philadelphia, PA, USA: SIAM, 2012.

[11] P. Giesl and S. Hafstein, "Review on computational methods for Lyapunov functions," *Discrete Contin. Dyn. Syst. – B*, vol. 20, no. 8, pp. 2291–2331, 2015.

[12] A. Papachristodoulou and S. Prajna, "A tutorial on sum of squares techniques for systems analysis," in *Proc. Amer. Control Conf.*, 2005, vol. 4, pp. 2686–2700.

[13] A. Clark, "Verification and synthesis of control barrier functions," in *Proc. IEEE Conf. Decis. Control*, 2021, pp. 6105–6112.

[14] A. A. Ahmadi, M. Krstic, and P. A. Parrilo, "A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf.*, 2011, pp. 7579–7580.

[15] P. Roux, Y.-L. Voronin, and S. Sankaranarayanan, "Validating numerical semidefinite programming solvers for polynomial invariants," *Form. Methods Syst. Des.*, vol. 53, pp. 286–312, 2018.

[16] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of Lyapunov neural networks," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 773–778, Jul. 2021.

[17] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks," in *Proc. Int. Conf. Hybrid Syst, Comput*. Control, 2021, pp. 1–11.

[18] M. Anand and M. Zamani, "Formally verified neural network control barrier certificates for unknown systems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 2431–2436, Jan. 2023.

[19] J. Liu, Y. Meng, M. Fitzsimmons, and R. Zhou, "Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification," *Automatica*, vol. 175, 2025, Art. no. 112193.

[20] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Mach. Learn.*, vol. 14, no. 1, pp. 115–133, 1994.

[21] F. B. Mathiesen, S. C. Calvert, and L. Laurenti, "Safety certification for stochastic systems via neural barrier functions," *IEEE Control Syst. Lett.*, vol. 7, pp. 973–978, 2023.

[22] H. Zhao, N. Qi, L. Dehbi, X. Zeng, and Z. Yang, "Formal synthesis of neural barrier certificates for continuous systems via counterexample guided learning," *ACM Trans. Embedded Comput. Syst.*, vol. 22, no. 5s, pp. 1–21, 2023.

[23] V. Vovk, "Conditional validity of inductive conformal predictors," in *Proc. Asian Conf. Mach. Learn.*, 2012, pp. 475–490.

[24] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas, "Safe planning in dynamic environments using conformal prediction," *IEEE Robot. Automat. Lett.*, vol. 8, no. 8, pp. 5116–5123, Aug. 2023.

[25] R. Coppola, A. Peruffo, L. Lindemann, and M. Mazo, "Scenario approach and conformal prediction for verification of unknown systems via data-driven abstractions," in *Proc. Eur. Control Conf.*, 2024, pp. 558–563.

[26] A. Lin and S. Bansal, "Verification of neural reachable tubes via scenario optimization and conformal prediction," in *Proc. 6th Annu. Learn. Dyn. Control Conf.*, 2024, pp. 719–731.

[27] M. Tayal, A. Singh, P. Jagtap, and S. Kolathaya, "CP-NCBF: A conformal prediction-based approach to synthesize verified neural control barrier functions," 2025, *arXiv:2503.17395*.

[28] S. Agrawal et al., "Scalable learning of high-dimensional demonstrations with composition of linear parameter varying dynamical systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2025, pp. 9917–9923.

[29] D. Boetius, A. Abdelnaby, A. Kumar, S. Leue, A. Swikir, and F. J. Abu-Dakka, "Stable robot motions on manifolds: Learning lyapunov-constrained neural manifold ODEs," 2025, *arXiv:2510.05707*.

[30] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Auton. Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[31] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1462–1469, Apr. 2019.

[32] L. Huber, J.-J. Slotine, and A. Billard, "Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3113–3132, Oct. 2022.

[33] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," 2021, *arXiv:2107.07511*.

[34] S. Haddadin et al., "The Franka Emika robot: A reference platform for robotics research and education," *IEEE Robot. Autom. Mag.*, vol. 29, no. 2, pp. 46–64, Jun. 2022.

[35] Y. Liu and Y. Zhang, "Toward welding robot with human knowledge: A remotely-controlled approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 769–774, Apr. 2015.

[36] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2007.

[37] K. Kronander and A. Billard, "Passive interaction control with dynamical systems," *IEEE Robot. Automat. Lett.*, vol. 1, no. 1, pp. 106–113, Jan. 2016.