

Learning a Formally Verified Control Barrier Function in Stochastic Environment

Manan Tayal¹, Hongchao Zhang², Pushpak Jagtap¹, Andrew Clark², and Shishir Kolathaya¹

Abstract—Safety is a fundamental requirement of control systems. Control Barrier Functions (CBFs) are proposed to ensure the safety of the control system by constructing safety filters or synthesizing control inputs. However, the safety guarantee and performance of safe controllers rely on the construction of valid CBFs. Inspired by universal approximability, CBFs are represented by neural networks, known as neural CBFs (NCBFs). This paper presents an algorithm for synthesizing formally verified continuous-time neural Control Barrier Functions in stochastic environments in a single step. The proposed training process ensures efficacy across the entire state space with only a finite number of data points by constructing a sample-based learning framework for Stochastic Neural CBFs (SNCBFs). Our methodology eliminates the need for post hoc verification by enforcing Lipschitz bounds on the neural network, its Jacobian, and Hessian terms. We demonstrate the effectiveness of our approach through case studies on the inverted pendulum system and obstacle avoidance in autonomous driving, showcasing larger safe regions compared to baseline methods.

I. INTRODUCTION

In the rapidly evolving landscape of control theory, ensuring safety in real-world applications has emerged as a paramount concern. With the pervasive integration of automated systems such as self-driving cars, the ability to guarantee their safety becomes increasingly critical. Current methods, including traditional optimal control approaches and Hamilton-Jacobi reachability analysis, have been instrumental in addressing safety requirements by casting them as constraints or generating safe controls offline. However, the scalability limitations of these methods, coupled with the challenge of swift reactions in complex environments, underscore the need for alternative approaches to safety assurance. With the rise in popularity of learning-based methods in control synthesis, learning-based methods have also been used to synthesize controllers for safety-critical tasks. However, most learning-based methods require a significant amount of unsafe interactions to learn a safe controller, which might be costly or impossible to obtain.

One promising avenue in addressing safety concerns is through the utilization of Control Barrier Functions (CBFs) [1]. CBFs serve as a practical method for synthesizing safe control for control affine systems [2] [3] as well as stochastic control systems [4], [5]. By formulating controllers through

Quadratic Programs (QPs), solvable at high frequencies with modern optimization solvers, CBFs have found applications in various safety-critical tasks, including adaptive cruise control [1], aerial maneuvers [6], [7] and legged locomotion [3], [8]. In these applications, the performance and safety guarantees depend on the CBF that is used. While control synthesis techniques such as sum-of-squares [9], [10] have been used to construct polynomial control barrier functions, they have been limited to systems with low-dimensional state spaces.

In recent years, the emergence of neural network-based barrier function (NCBF) synthesis has garnered considerable attention due to the universal approximation property of neural networks. A variety of methods have been proposed for training NCBFs, including learning from expert demonstrations [11], SMT-based techniques [12]–[15], mixed-integer programs [16], and nonlinear programs [17]. Loss functions for training NCBFs were proposed in [18]–[22], while [23] synthesizes CBFs by learning a value function of a nominal policy and demonstrating that the maximum-over-time cost serves as a CBF. In these existing works, the trained CBFs must be verified as a postprocessing step and recomputed if verification fails, resulting in a potentially time-consuming trial-and-error process.

More recently, a data-driven approach for training NCBFs was proposed in [24]. The proposed training process also bounds the Lipschitz constant of the trained NCBF. By exploiting these Lipschitz bounds, the safety of the continuous state space can be verified using only a discrete set of sample points. This approach, however, only considers discrete-time deterministic systems. In a continuous-time setting, the safety guarantees also depend on the Lie derivative of the NCBF, and hence Lipschitz bounds on the NCBF alone are insufficient to guarantee safety. Moreover, the abovementioned works on NCBFs do not consider the presence of stochastic system noise, making formally verifiable synthesis of stochastic NCBFs an open research problem.

In this paper, we propose an algorithm to synthesize a formally verified continuous-time neural CBF in stochastic environments in a single step. We construct a sample-based learning framework to train Stochastic Neural CBF (SNCBF) and prove the trained SNCBF ensuring efficacy across the entire state space with only a finite number of data points. To summarize, this paper makes the following contributions.

- We propose a training framework to synthesize provably correct Control Barrier Functions (CBFs) parameterized as neural networks for continuous-time, stochastic systems, eliminating any need for post hoc verification.

*This work was supported by the ARTPARK, PMRF, Indian Army Design Bureau

¹ Center for Cyber-Physical Systems, Indian Institute of Science (IISc), Bengaluru. {manantayal, pushpak, shishirk}@iisc.ac.in.

² Electrical & Systems Engineering, Washington University in St. Louis. {hongchao, andrewclark}@wustl.edu.

- Our methodology establishes completeness guarantees by deriving a validity condition, ensuring efficacy across the entire state space with only a finite number of data points. We train the network by enforcing Lipschitz bounds on the neural network, its Jacobian and (trace of) Hessian terms.
- We evaluate our approach using two case studies, namely, the inverted pendulum system and the obstacle avoidance of an autonomous driving system. We show that our training framework successfully constructs an SNCBF to differentiate safe and unsafe regions. We demonstrate our proposed method ensures a larger safe region compared with the baseline method in [22].

The rest of this paper is organized as follows. Preliminaries explaining the concept of control barrier functions (CBFs) and safety filter designs are introduced in Section II. Problem formulation is discussed in Section III. The method proposed to synthesize the CBFs using neural networks, the construction of loss functions and the algorithm to train it, are discussed in Section IV. The simulation results will be discussed in Section V. Finally, we present our conclusion in Section VI.

II. PRELIMINARIES

In this section, we will formally introduce Control Barrier Functions (CBFs) and their importance for real-time safety-critical control.

A. Notations

We first present the definitions of class- \mathcal{K} and extended class- \mathcal{K} functions. A continuous function $\kappa : [0, d) \rightarrow [0, \infty)$ for some $d > 0$ is said to belong to *class- \mathcal{K}* if it is strictly increasing and $\kappa(0) = 0$. Here, d is allowed to be ∞ . The same function can be extended to the interval $\kappa : (-b, d) \rightarrow (-\infty, \infty)$ with $b > 0$ (which is also allowed to be ∞), in this case we call it the *extended class- \mathcal{K} function*. Given a square matrix A , the trace representing the sum of its diagonal elements is denoted by $\text{tr}(A)$, and the determinant is denoted by $\det(A)$. Given a function $\phi(x)$, we represent the derivative and double derivative of $\phi(x)$ with respect to the input x as $\phi'(x)$ and $\phi''(x)$, respectively. If $m = [m_1, \dots, m_n]^T \in \mathbb{R}^n$ is a $n \times 1$ matrix, then

$$\text{diag}(m) = \begin{bmatrix} m_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & m_n \end{bmatrix}.$$

B. System Description

We consider a continuous time stochastic control system with state $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ and input $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ at time $t \geq 0$. The state dynamics is described by the stochastic differential equation as:

$$dx(t) = (f(x(t)) + g(x(t))u(t)) dt + \sigma dW(t), \quad (1)$$

where functions $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ are locally Lipschitz, $\sigma \in \mathbb{R}^{n \times n}$, W is an n -dimensional Brownian motion.

Consider a set \mathcal{C} defined as the *super-level set* of a continuously differentiable function $h : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ yielding,

$$\mathcal{C} = \{x \in \mathcal{X} \subseteq \mathbb{R}^n : h(x) \geq 0\} \quad (2)$$

$$\mathcal{X} - \mathcal{C} = \{x \in \mathcal{X} \subseteq \mathbb{R}^n : h(x) < 0\}. \quad (3)$$

We further let the interior and boundary of \mathcal{C} be $\text{Int}(\mathcal{C}) = \{x \in \mathcal{X} \subseteq \mathbb{R}^n : h(x) > 0\}$ and $\partial\mathcal{C} = \{x \in \mathcal{X} \subseteq \mathbb{R}^n : h(x) = 0\}$, respectively.

We define a Lipschitz continuous control policy $\mu : \mathbb{R}^n \rightarrow \mathbb{R}^m$ to be a mapping from the sequence of states $x(t)$ to a control input $u(t)$ at each time t . The safety of a controlled system is defined as follows.

Definition 1 (Safety): A set $\mathcal{C} \subseteq \mathcal{X} \subseteq \mathbb{R}^n$ is positive invariant under dynamics (1) and control policy μ if $x(0) \in \mathcal{C}$ and $u(t) = \mu(x(t)) \forall t \geq 0$ imply that $x(t) \in \mathcal{C}$ for all $t \geq 0$. If \mathcal{C} is positive invariant, then the system satisfies the safety constraint with respect to \mathcal{C} .

C. Control Barrier Functions (CBFs)

The control policy needs to guarantee the robot satisfies a safety constraint, which is specified as the positive invariance of a given safety region \mathcal{C} . The Control Barrier Function (CBF) are widely used to synthesize a control policy with positive invariance guarantees. We next present the definition of CBFs as discussed in [3] for nonstochastic systems.

Definition 2 (Control barrier function (CBF)): Given a control-affine system $\dot{x} = f(x) + g(x)u$, the set \mathcal{C} defined by (2), with $\frac{\partial h}{\partial x}(x) \neq 0$ for all $x \in \partial\mathcal{C}$, the function h is called the control barrier function (CBF) defined on the set \mathcal{X} , if there exists an extended *class- \mathcal{K}* function κ such that for all $x \in \mathcal{X}$:

$$\sup_{u \in \mathcal{U}} \left[\underbrace{\mathcal{L}_f h(x) + \mathcal{L}_g h(x)u}_{\dot{h}(x,u)} + \kappa(h(x)) \right] \geq 0, \quad (4)$$

where $\mathcal{L}_f h(x) = \frac{\partial h}{\partial x} f(x)$ and $\mathcal{L}_g h(x) = \frac{\partial h}{\partial x} g(x)$ are the Lie derivatives.

By [2] and [3], we have that any Lipschitz continuous control law $\mu(x)$ satisfying the inequality: $\dot{h} + \kappa(h) \geq 0$ ensures safety of \mathcal{C} if $x(0) \in \mathcal{C}$, and asymptotic convergence to \mathcal{C} if $x(0)$ is outside of \mathcal{C} .

We next present the safety lemma to ensure the safety of continuous-time stochastic systems.

Lemma 1 ([5]): The set $\mathcal{C} \subset \mathbb{R}^n$ be a set defined on the super-level set of a continuously differentiable function $h : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$. The function h defined on the set \mathcal{X} is a CBF for stochastic system in (1), if there exists an extended *class- \mathcal{K}* function κ such that for all $x \in \mathcal{X}$:

$$\sup_{u \in \mathcal{U}} \left[\mathcal{L}_f h(x) + \mathcal{L}_g h(x)u + \frac{1}{2} \text{tr} \left(\sigma^T \frac{\partial^2 h(x)}{\partial x^2} \sigma \right) + \kappa(h(x)) \right] \geq 0. \quad (5)$$

D. Controller Synthesis for Real-time Safety

Having described the CBF and its associated formal results, we now discuss its Quadratic Programming (QP) formulation. CBFs are typically regarded as *safety filters* which take the desired input (reference controller input) $u_{ref}(x, t)$ and modify this input in a minimal way:

$$\begin{aligned} u^*(x, t) = \min_{u \in \mathbb{U} \subseteq \mathbb{R}^m} \|u - u_{ref}(x, t)\|^2 \\ \text{s.t. } \mathcal{L}_f h(x) + \mathcal{L}_g h(x)u + \frac{1}{2} \text{tr} \left(\sigma^\top \frac{\partial^2 h(x)}{\partial x^2} \sigma \right) \\ + \kappa(h(x)) \geq 0. \end{aligned} \quad (6)$$

This is called the Control Barrier Function based Quadratic Program (CBF-QP).

III. PROBLEM FORMULATION

In this section, we begin by formally defining the problem of synthesis of CBF in a stochastic environment and highlighting the challenges associated with its direct solution. Subsequently, we propose a reformulation of the problem and derive the conditions, that provide formal guarantees on the correctness of the solution over the entire state set despite using finitely many data samples.

To leverage the universal approximation property of neural networks, we represent the Stochastic Control Barrier Function with a feed-forward neural network. Denoted as $\tilde{h}(x | \theta)$, where θ signifies the trainable parameters of the neural network, this representation is termed the Stochastic Neural CBF (SNCBF). Due to the absence of an SNCBF, we lack access to the safe set \mathcal{C} . Hence, we start with initial safe and unsafe sets $\mathcal{X}_s \subseteq \mathcal{C}$ and $\mathcal{X}_u \subseteq \mathcal{X} - \mathcal{C}$, respectively, such that any trajectory starting in \mathcal{X}_s never enters \mathcal{X}_u . We next formulate the problem.

Problem 1: Given a continuous-time stochastic control system defined as (1), state set \mathcal{X} , initial safe and unsafe sets \mathcal{X}_s and \mathcal{X}_u , respectively, the objective is to devise an algorithm to synthesize SNCBF $\tilde{h}(x | \theta)$ using a Lipschitz continuous controller u such that

$$\begin{aligned} \tilde{h}(x | \theta) \geq 0, \forall x \in \mathcal{X}_s, \\ \tilde{h}(x | \theta) < 0, \forall x \in \mathcal{X}_u, \\ \frac{\partial \tilde{h}(x | \theta)}{\partial x} (f(x) + g(x)u(x)) + \frac{1}{2} \text{tr} \left(\sigma^\top \frac{\partial^2 \tilde{h}(x | \theta)}{\partial x^2} \sigma \right) \\ + \kappa(\tilde{h}(x | \theta)) \geq 0, \forall x \in \mathcal{X}. \end{aligned} \quad (7)$$

In order to enforce conditions (7) in Problem 1, we first cast our problem as the following robust optimization problem (ROP):

$$\text{ROP} : \begin{cases} \min_{\psi} & \psi \\ \text{s.t.} & \max(q_k(x)) \leq \psi, k \in \{1, 2, 3\} \\ & \forall x \in \mathcal{X}, \psi \in \mathbb{R}, \end{cases} \quad (8)$$

where

$$\begin{aligned} q_1(x) &= (-\tilde{h}(x | \theta)) \mathbb{1}_{\mathcal{X}_s}, \\ q_2(x) &= (\tilde{h}(x | \theta) + \delta) \mathbb{1}_{\mathcal{X}_u}, \\ q_3(x) &= -\frac{\partial \tilde{h}(x | \theta)}{\partial x} (f(x) + g(x)u(x)) \\ &\quad - \frac{1}{2} \text{tr} \left(\sigma^\top \frac{\partial^2 \tilde{h}(x | \theta)}{\partial x^2} \sigma \right) - \kappa(\tilde{h}(x | \theta)), \end{aligned} \quad (9)$$

where δ is a small positive value to ensure the strict inequality. If the optimal solution of the ROP (ψ_{ROP}^*) ≤ 0 , then the conditions (7) are satisfied and the SNCBF is valid.

However, the proposed ROP in (8) has infinitely many constraints since the state of the system in a continuous set. This motivates us to employ data-driven approaches and the scenario optimization program of ROP. Given $\bar{\epsilon}$, suppose we sample N data points: $x_i \in \mathcal{X}, i \in \{1, \dots, N\}$, such that $\|x - x_i\| \leq \bar{\epsilon}$. Instead of solving the ROP in (8), we employ the following scenario optimization problem (SOP):

$$\text{SOP} : \begin{cases} \min_{\psi} & \psi \\ \text{s.t.} & q_1(x_i) \leq \psi, \forall x_i \in \mathcal{S}, \\ & q_2(x_i) \leq \psi, \forall x_i \in \mathcal{U}, \\ & q_3(x_i) \leq \psi, \forall x_i \in \mathcal{D}, \\ & \psi \in \mathbb{R}, i \in \{1, \dots, N\}, \end{cases} \quad (10)$$

where $q_k(x), k \in \{1, 2, 3\}$ are defined as in (9). The data sets \mathcal{S}, \mathcal{U} and \mathcal{D} corresponding to points sampled from the initial safe set $\mathcal{X}_s \subseteq \mathcal{C}$, initial unsafe set $\mathcal{X}_u \subseteq \mathcal{X} - \mathcal{C}$, and state set \mathcal{X} , respectively.

Given the finite number of data samples x_i , and considering SOP as a linear program in relation to the decision variable ψ , it becomes feasible to find a solution for the SOP. Let us denote the optimal solution of the SOP as ψ^* . We now derive conditions under which the SNCBF $\tilde{h}(x | \theta)$ satisfies conditions (7). The following theorem shows that solving (8) can be achieved by finding a solution to (10).

Theorem 1: Consider a continuous time stochastic control system (1), and initial safe and unsafe sets $\mathcal{X}_s \subseteq \mathcal{X}$ and $\mathcal{X}_u \subseteq \mathcal{X}$, respectively. Let $\tilde{h}(x | \theta)$ be the neural network-based CBF with trainable parameters θ . For the SOP (10) constructed by utilizing N samples, let ψ^* be the optimal value, with the assumption that functions $q_k(x), k \in \{1, 2, 3\}$ in equation (9) are Lipschitz continuous. Then $\tilde{h}(x | \theta)$ is a valid SCBF, i.e., it satisfies conditions (7), if the following condition holds:

$$L_{\max} \bar{\epsilon} + \psi^* \leq 0, \quad (11)$$

where L_{\max} is maximum of the Lipschitz constants of $q_k(x), k \in \{1, 2, 3\}$ in (9).

Hence, the original Problem 1 can be efficiently addressed by solving the problem reformulated as follows.

Problem 2: Given a continuous-time stochastic control system defined as (1) and the data sets \mathcal{S}, \mathcal{U} and \mathcal{D} , the objective is to devise an algorithm to synthesize SNCBF $\tilde{h}(x | \theta)$ and consequently, SNCBF – QP based controller

u , such that they satisfy the conditions required in SOP (10) (functions $q_k(x) \leq \psi^*$, $k \in \{1, 2, 3\}$ in equation (9)) and ψ^* satisfies condition (11).

IV. METHOD

In this section, we propose an algorithmic approach to solve the problem formulated in Section III. The structure of this section is as follows. We first present the method to synthesize SNCBF to solve Problem 2 and then demonstrate the training process.

A. Synthesis of SNCBF

Following the problem formulation in Section III, we now describe the construction of suitable loss functions for the training SNCBF $\tilde{h}(x | \theta)$ such that its minimization leads to the solution of Problem 2. As described in the previous section, SNCBF is a feed-forward neural network, with trainable weight parameters θ . To address the aforementioned problem, it is imperative to compute $\frac{\partial \tilde{h}(x|\theta)}{\partial x}$ and $\text{tr}(\frac{\partial^2 \tilde{h}(x|\theta)}{\partial x^2})$. This necessitates the selection of a neural network with a smooth activation function, thereby facilitating the derivation of smooth Jacobian and Hessian values upon differentiation. Interested readers are directed to [25], which analytically computes the Jacobian and Hessian using the chain rule.

Now, let us consider the following loss functions satisfying the conditions required by SOP (10) over the training data sets $\mathcal{S}, \mathcal{U}, \mathcal{D}$ as follows:

$$\begin{aligned}\mathcal{L}_1(\theta) &= \frac{1}{N} \sum_{x_i \in \mathcal{S}} \max(0, q_1(x_i) - \psi), \\ \mathcal{L}_2(\theta) &= \frac{1}{N} \sum_{x_i \in \mathcal{U}} \max(0, q_2(x_i) - \psi), \\ \mathcal{L}_3(\theta) &= \frac{1}{N} \sum_{x_i \in \mathcal{D}} \max(0, q_3(x_i) - \psi)\end{aligned}\quad (12)$$

with \mathcal{L}_1 representing the loss for safe states, \mathcal{L}_2 representing the loss for unsafe states and \mathcal{L}_3 representing the loss for lie derivative conditions over the entire state set, respectively. Using the terms defined above, the loss function is given by

$$\mathcal{L}_\theta(\theta) = \mathcal{L}_1 + \lambda_1 \mathcal{L}_2 + \lambda_2 \mathcal{L}_3, \quad (13)$$

with $\lambda_1, \lambda_2 \in \mathbb{R}_+$ weighting the importance of the individual loss terms.

To ensure the fulfillment of the assumptions in Theorem 1, it is imperative to verify the Lipschitz boundedness of the functions $q_k(x)$, where $k \in \{1, 2, 3\}$. This necessitates the Lipschitz boundedness of $\tilde{h}(x | \theta)$, $\frac{\partial \tilde{h}(x|\theta)}{\partial x}$, and $\text{tr}(\sigma^\top \frac{\partial^2 \tilde{h}(x|\theta)}{\partial x^2} \sigma)$, with corresponding Lipschitz bounds denoted as L_h , L_{dh} , and L_{d^2h} , respectively. To train neural networks with Lipschitz bounds, we have the following lemma:

Lemma 2 ([26]): Suppose f_θ is a l -layered feed-forward neural network with θ as a trainable parameter, then a certificate for L -Lipschitz continuity of the neural network

is given by the semi-definite constraint $M(\theta, \Lambda) :=$

$$\begin{bmatrix} A \\ B \end{bmatrix}^T \begin{bmatrix} 2\alpha\beta\Lambda & -(\alpha + \beta)\Lambda \\ -(\alpha + \beta)\Lambda & 2\Lambda \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} + \begin{bmatrix} L^2\mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\theta_l^T \\ 0 & 0 & -\theta_l & \mathbf{I} \end{bmatrix} \succeq 0,$$

where

$$A = \begin{bmatrix} \theta_0 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \theta_{l-1} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & \mathbf{I} \end{bmatrix},$$

($\theta_0, \dots, \theta_l$) are the weights of the neural network, $\Lambda \in \mathbb{D}_+^n$, $i = 1, \dots, l$ and α and β are the minimum and maximum slopes of the activation functions, respectively.

Remark 1: For a single-layer case, the matrix M reduces to:

$$M(\theta, \Lambda) = \begin{bmatrix} L^2\mathbf{I} + 2\alpha\beta\theta_0^T\Lambda\theta_0 & -(\alpha + \beta)\theta_0^T\Lambda & 0 \\ -(\alpha + \beta)\Lambda\theta_0 & 2\Lambda & -\theta_1 \\ 0 & -\theta_1 & \mathbf{I} \end{bmatrix} \succeq 0.$$

The lemma above addresses the certification of the L -Lipschitz bound for a neural network. However, our scenario necessitates ensuring not only the Lipschitz boundedness of the SNCBF \tilde{h} , but also of $\frac{\partial \tilde{h}(x_i|\theta)}{\partial x}$ and $\sigma^\top \frac{\partial^2 \tilde{h}(x_i|\theta)}{\partial x^2} \sigma$. Therefore, we must explore the relationship between the network weights and the semi-definite matrix M to guarantee the boundedness of the aforementioned terms. To address this issue, we introduce the following theorem.

Theorem 2: Consider a 1-layered feedforward neural network f_θ , with output dimension 1×1 , where θ represents the trainable weight parameters. Let y denote the output of the neural network, x denote the input of the neural network, θ_i , $i \in 0, 1$, denote the weight parameters of each layer with $\theta = \{\theta_0, \theta_1\}$ and ϕ denote the activation. The certificate for L -Lipschitz continuity of the derivative of the neural network ($\frac{\partial y}{\partial x}$) is given by $M_{\hat{\phi}}(\hat{\theta}, \Lambda) \succeq 0$, where $\hat{\phi} = \phi'$ and $\hat{\theta} = (\theta_0, \hat{\theta}_1)$, $\hat{\theta}_1$ is defined as:

$$\hat{\theta}_1 = \theta_0^T \text{diag}(\theta_1). \quad (14)$$

Similarly, the certificate for the L -Lipschitz continuity of $\text{tr}(\sigma^\top \frac{\partial^2 y}{\partial x^2} \sigma)$ is expressed as $M_{\bar{\phi}}(\bar{\theta}, \Lambda) \succeq 0$, where $\bar{\phi} = \phi''$ and $\bar{\theta} = (\theta_0, \bar{\theta}_1)$ and $\bar{\theta}_1$ is defined as:

$$\bar{\theta}_1 = \left[\sum_{j=0}^r \sigma_j^2 \theta_1^{j1} \theta_0^{1j} \quad \dots \quad \sum_{j=0}^r \sigma_j^2 \theta_1^{jn} \theta_0^{nj} \right]. \quad (15)$$

Proof: The proof can be found in Appendix of [27]. ■

We address a constrained optimization problem aiming to minimize loss $\mathcal{L}(f_\theta)$ subject to $M_j(\theta, \Lambda) \succeq 0$ for $j = 1, 2, 3$, where f_θ is a given neural network. By employing a log-det barrier function, we convert this into an unconstrained optimization problem:

$$\min_{\theta, \Lambda} \mathcal{L}(f_\theta) + \mathcal{L}_M(\theta, \Lambda),$$

where $\mathcal{L}_M(\theta, \Lambda) = -\sum_{j=0}^q \rho_j \log \det(\mathbf{M}_j(\theta, \Lambda))$ and $\rho_j > 0$ are barrier parameters. Ensuring that the loss function

$\mathcal{L}_M(\theta, \Lambda) \leq 0$, guarantees that the linear matrix inequalities $\mathcal{M}_j(\theta, \Lambda) \succeq 0, j = 1, 2, 3$ hold true. Let us consider the loss functions characterizing the satisfaction of Lipschitz bound as

$$\mathcal{L}_M(\theta, \Lambda, \hat{\Lambda}, \bar{\Lambda}) = -c_{l_1} \log \det(M_1(\theta, \Lambda)) - c_{l_2} \log \det(M_2(\hat{\theta}, \hat{\Lambda})) - c_{l_3} \log \det(M_3(\bar{\theta}, \bar{\Lambda})), \quad (16)$$

where $c_{l_1}, c_{l_2}, c_{l_3}$ are positive weight coefficients for the sub-loss functions, M_1, M_2, M_3 are the semi-definite matrices corresponding to the Lipschitz bounds L_h, L_{dh}, L_{d2h} respectively, $\Lambda, \hat{\Lambda}, \bar{\Lambda}$ are trainable parameters and $\theta, \hat{\theta}, \bar{\theta}$ are the weights mentioned in Theorem 2.

Finally, let us consider the following loss function to satisfy validity condition (11):

$$\mathcal{L}_v(\psi) = \max(0, L_{\max} \bar{\epsilon} + \psi), \quad (17)$$

where L_{\max} is maximum of the Lipschitz constants of $q_k(x), k \in \{1, 2, 3\}$ in (9), or $L_{\max} = \max(L_h, L_h + L_{dh}L_x + L_{d2h})$.

B. Training with Safety Guarantee

Algorithm 1 Learning Formally Verified Stochastic Neural Control Barrier Functions

Require: Data Sets: $\mathcal{S}, \mathcal{U}, \mathcal{D}$, Dynamics: f, g, σ , Lipschitz Bounds: $L_h, L_{dh}, L_x, L_{d2h}$
 Initialise($\theta, \psi, \Lambda, \Lambda', \Lambda''$)
 $x_i \leftarrow \text{sample}(\mathcal{S}, \mathcal{U}, \mathcal{D})$
 $L_{\max} \leftarrow L_h, L_{dh}, L_x, L_{d2h}$
while $\mathcal{L}_\theta > 0$ or $\mathcal{L}_M \not\leq 0$ or $\mathcal{L}_v > 0$ **do**
 $\tilde{h} \leftarrow \theta$
 $u_i \leftarrow \text{SNCBF-QP}(\tilde{h}, f, g, \sigma, x_i, \psi) \triangleright$ From eq. (18)
 $\mathcal{L}_\theta \leftarrow (\tilde{h}, f, g, \sigma, x_i, u_i, \psi) \triangleright$ From eq. (13)
 $\theta \leftarrow \text{Learn}(\mathcal{L}_\theta, \theta)$
 $\mathcal{L}_M \leftarrow (\theta, \Lambda, \Lambda', \Lambda'') \triangleright$ From eq. (16)
 $\theta, \Lambda, \Lambda', \Lambda'' \leftarrow \text{Learn}(\mathcal{L}_M)$
 $\mathcal{L}_v \leftarrow (L_{\max}, \psi) \triangleright$ From eq. (17)
 $\psi \leftarrow \text{Learn}(\mathcal{L}_v)$
end while

The overall training procedure is as follows. We first fix all the hyper-parameters required for the training, including $\bar{\epsilon}, \mathcal{L}_h, \mathcal{L}_{dh}, \mathcal{L}_{d2h}, \lambda_1, \lambda_2, c_{l_1}, c_{l_2}, c_{l_3}$, and the maximum number of epochs considered. At each time step, the controller intakes the current state x and reference control input $u_{\text{ref}}(x)$. The basic idea is to construct safe control input by filtering out unsafe actions based on the condition derived from learned SNCBF. Specifically, the controller minimizes the modification on the control input u compared to u_{ref} such that the SNCBF safety condition holds by solving SNCBF-QP defined as follows.

$$\begin{aligned} \min_{u \in \mathcal{U}} & \|u - u_{\text{ref}}(x)\| \\ \text{s.t.} & \frac{\partial \tilde{h}(x_i | \theta)}{\partial x} (\mathbf{f}(x_i) + \mathbf{g}(x_i)u) + \frac{1}{2} \text{tr} \left(\sigma^\top \frac{\partial^2 \tilde{h}(x_i | \theta)}{\partial x^2} \sigma \right) \\ & \geq -\kappa(\tilde{h}(x_i)). \end{aligned} \quad (18)$$

Finally, the control action u can be used in computing loss functions in the next iteration. The training data sets are randomly shuffled into several batches, and the loss is calculated for each batch at a time. Then, using a stochastic gradient descent algorithm, e.g., ADAM, the trainable parameters $\theta, \Lambda, \hat{\Lambda}, \bar{\Lambda}$ and ψ are updated. This procedure is repeated until the network converges or if a predefined maximum episode number is reached. For our proposed approach, all of the learning is done offline in a simulation environment. After the learning process converges, the SNCBF-QP based controller u can then be deployed to the intended system. The overall algorithm is summarised in Algorithm 1

We next present the following theorem that provides formal guarantees of safety for the continuous time stochastic control system by utilizing the trained SNCBF $\tilde{h}(x | \theta)$.

Theorem 3: Consider a continuous time stochastic control system (1), and safe and unsafe sets $X_s \subseteq X$ and $X_u \subseteq X$, respectively. Let $\tilde{h}(x | \theta)$ be the trained SNCBF, such that $\mathcal{L}_\theta = 0, \mathcal{L}_M \leq 0$ and $\mathcal{L}_v = 0$. Then, the control action u obtained by solving the SNCBF-QP (18), makes the system safe.

V. SIMULATIONS

In this section, we assess the efficacy of our proposed methodology through two distinct case studies: the inverted pendulum model and the obstacle avoidance of an autonomous mobile robot [28]. Both case studies are conducted on a computing platform equipped with an Intel i7-12700H CPU, 32GB RAM, and NVIDIA GeForce RTX 3090 GPU. For the class \mathcal{K} function in the CBF inequality (5), we chose $\kappa(h) = \gamma h$, where $\gamma = 1$.

A. Inverted Pendulum

We consider a continuous time stochastic inverted pendulum dynamics given as follows:

$$d \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} = \left(\begin{bmatrix} \dot{\theta} \\ \frac{g}{l} \sin(\theta) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix} u \right) dt + \sigma dW_t, \quad (19)$$

where $\theta \in \mathbb{R}$ denotes the angle, $\dot{\theta} \in \mathbb{R}$ denotes the angular velocity, $u \in \mathbb{R}$ denotes the controller, m denotes the mass and l denotes the length of the pendulum. We let the mass $m = 1\text{kg}$, length $l = 10\text{m}$ and the disturbance $\sigma = \text{diag}(0.1, 0.1)$. The inverted pendulum operate in a state space given as $\mathcal{X} = [-\frac{\pi}{4}, \frac{\pi}{4}]^2$. The system is required to stay in a limited safe stable region. The initial safe region is given as $\mathcal{X}_s = [-\frac{\pi}{15}, \frac{\pi}{15}]^2$ and the unsafe region is given as $\mathcal{X}_u = \mathcal{X} \setminus [-\frac{\pi}{6}, \frac{\pi}{6}]^2$.

We train the SNCBF, assuming knowledge of the model. The SNCBF \tilde{h} consists of one hidden layer of 20 neurons, with Softplus activation function ($\log(1 + \exp(x))$).

We set the training hyper-parameters to $\bar{\epsilon} = 0.00016$, $L_h = 0.01$, $L_{dh} = 0.4$ and $L_{d2h} = 2$ yielding $L_{\max} = 2.4$. We perform the training to simultaneously minimize the loss functions $\mathcal{L}_\theta, \mathcal{L}_M$, and \mathcal{L}_v . The training algorithm then converges to obtain the SNCBF $\tilde{h}(x | \theta)$ with $\psi^* =$

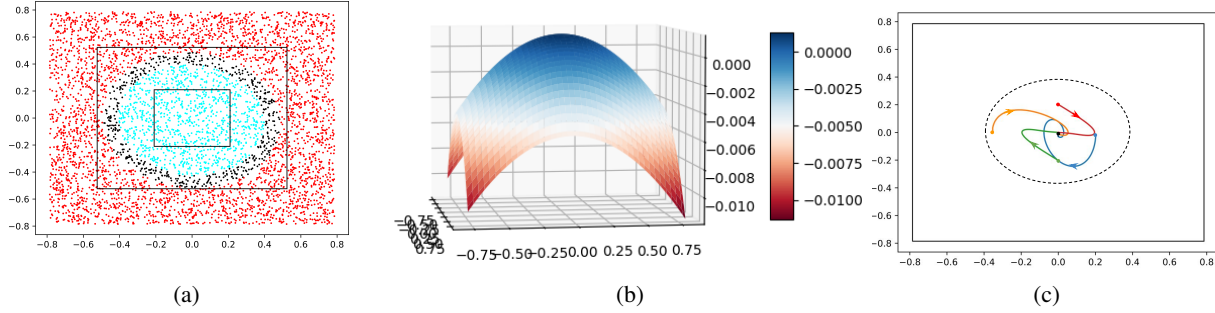


Fig. 1: This figure presents the experimental results on the inverted pendulum system. Fig. 1a visualizes of $\tilde{h}(x | \theta)$ over X . Blue and red regions denote the safe region ($\tilde{h} \geq -\psi^*$) and the unsafe region ($\tilde{h} < \psi^*$), respectively. The initial safety region boundary and unsafe region boundary is denoted by black boxes. We observe that the boundary of trained SNCBF (black dots) successfully separate the unsafe and safe region. Fig. 1b shows the 3D plot of $\tilde{h}(x | \theta)$ over X . Fig. 1c presents trajectories initiating inside the safe set following SNCBF-QP, following different reference controllers

-0.00042 . Thus, using Theorem 1, we can verify that the SNCBF thus obtained is valid, thus ensuring safety.

Visualizations of the trained SNCBF are presented in both 2D with sample points (Fig. 1a) and in 3D function value heat map (Fig. 1b). These visualizations demonstrate the successful separation of the initial safety region boundary from the unsafe region boundary. We validate our SNCBF-QP based safe controller on an inverted pendulum model on PyBullet. Fig 1c shows the trajectories initiating inside the safe set (with different reference controllers) and never leaving the safe set, thus validating our approach.

B. Obstacle Avoidance

We consider an autonomous mobile robot navigating on a road following the dynamics [29] given below

$$d \begin{bmatrix} x_1 \\ x_2 \\ \psi \end{bmatrix} = \left(\begin{bmatrix} v \cos \psi \\ v \sin \psi \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u \right) dt + \sigma dW_t \quad (20)$$

where $[x_1, x_2, \psi]^T \in \mathcal{X} \subseteq \mathbb{R}^3$ is the state consisting of the location (x_1, x_2) of the robot and its orientation ψ , with v representing the robot's speed and u controlling its orientation. We set the speed $v = 1$ and the disturbance $\sigma = \text{diag}(0.1, 0.1, 0.1)$.

The mobile robot is required to stay on the road while avoiding pedestrians sharing the field of activities. The unicycle operates in a state space given as $\mathcal{X} = [-2, 2]^3$. The initial safe region is given as $\mathcal{X}_s = \mathcal{X} \setminus [-1.5, 1.5]^2 \times [-2, 2]$ and the unsafe region is given as $\mathcal{X}_u = [-0.2, 0.2]^2 \times [-2, 2]$.

We train the SNCBF, assuming knowledge of the model with the same architecture as before (one hidden layer of 20 neurons, with Softplus activation function ($\log(1 + \exp(x))$)). The training hyper-parameters are set to $\bar{\epsilon} = 0.01$, $L_h = 1$, $L_{dh} = 1$ and $L_{d2h} = 2$ resulting in $L_{\max} = 4$. We perform the training to simultaneously minimize the loss functions \mathcal{L}_θ , \mathcal{L}_M , and \mathcal{L}_v . The training algorithm then converges to obtain the SNCBF $\tilde{h}(x | \theta)$ along with $\psi^* = -0.04002$.

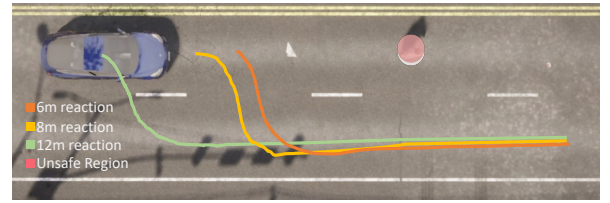


Fig. 2: Proposed safe control comparison among different reaction distance. We let the vehicle to adjust its orientation to maneuver in its lane. We show three trajectories to demonstrate our proposed SNCBF-based controller under different initial state, namely, 6, 8 and 12 meter away from the pedestrian, respectively. Three trajectories of the vehicle under control shows our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

We validate our safe control in the CARLA simulation environment, in which the vehicle maneuvers its orientation to conduct obstacle avoidance. If the control policy guides the vehicle to another lane with no obstacle ahead, then it switches to a baseline autonomous driving algorithm. The trajectory of the vehicle is shown in Fig. 2. We show the trajectory of the proposed SNCBF-based safe control in different reaction distances, namely, 6, 8, and 12 meters, respectively. All three trajectories of the vehicle under control show our proposed method succeeds in maneuvering the vehicle to avoid the pedestrian.

We next compare our proposed method with SNCBF trained with baseline proposed in [22] assuming given function $h(x)$. The synthesized SNCBF are visualized in 3. We observe that the synthesized SNCBF \tilde{h} successfully separates the safe and the unsafe region, since both zero-level sets (in yellow) do not overlap with the unsafe region in red color. We also found that the proposed method ensures a larger subset compared with the baseline. The baseline ensures that 11.8% of the state space is safe, while the proposed method ensures that 77.6% of the state space is safe.

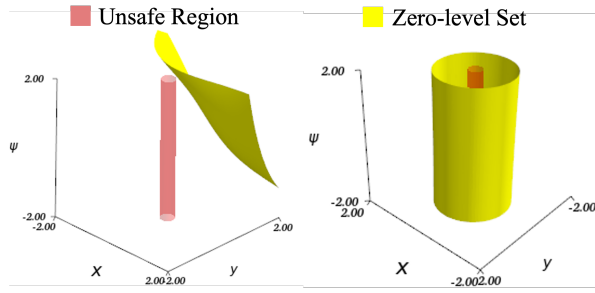


Fig. 3: The left and right figures show the unsafe region and the zero-level sets of the SNCBF \tilde{h} trained by baseline and the proposed method, respectively. Both zero-level sets (in yellow) do not overlap with the unsafe region in red color. The SNCBF trained by the baseline ensures a guaranteed safe subset over the state space with the ratio 11.8% only, whereas the proposed method ensures a guaranteed safe subset with a ratio up to 77.6%.

VI. CONCLUSIONS

In this paper, we proposed an algorithmic approach to learn a valid continuous-time neural CBF with formal guarantees in a stochastic environment. We constructed a sample-based training framework to train SNCBF and proved that the efficacy of learned SNCBF by enforcing Lipschitz bounds on the neural network, its Jacobian and (trace of) Hessian terms. We further derived the sufficient condition for the safety of SNCBF-based system. The effectiveness of our proposed approach was demonstrated using the simulation study on inverted pendulum and obstacle avoidance. As a part of future work, we plan to extend this framework to account for unknown dynamics and control bounds. We also plan to improve the algorithm in order to iterate on the improving the size of the safe region and making the learnt CBF less conservative.

REFERENCES

- [1] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.
- [2] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.
- [3] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.
- [4] P. Jagtap, S. Soudjani, and M. Zamani, "Formal synthesis of stochastic systems via control barrier certificates," *IEEE Transactions on Automatic Control*, vol. 66, no. 7, pp. 3097–3110, 2020.
- [5] A. Clark, "Verification and synthesis of control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 6105–6112.
- [6] G. Wu and K. Sreenath, "Safety-critical control of a planar quadrotor," in *2016 American Control Conference (ACC)*, 2016, pp. 2252–2258.
- [7] M. Tayal and S. Kolathaya, "Control barrier functions in dynamic uavs for kinematic obstacle avoidance: a collision cone approach," *arXiv preprint arXiv:2303.15871*, 2023.
- [8] Q. Nguyen and K. Sreenath, "Safety-critical control for dynamical bipedal walking with precise footstep placement," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 147–154, 2015.

- [9] A. Papachristodoulou and S. Prajna, "A tutorial on sum of squares techniques for systems analysis," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 2686–2700 vol. 4.
- [10] U. Topcu, A. Packard, and P. Seiler, "Local stability analysis using simulations and sum-of-squares programming," *Automatica*, vol. 44, no. 10, pp. 2669–2675, 2008.
- [11] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3717–3724.
- [12] H. Zhao, X. Zeng, T. Chen, and Z. Liu, "Synthesizing barrier certificates using neural networks," in *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, 2020, pp. 1–11.
- [13] 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 *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.
- [14] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.
- [15] A. Peruffo, D. Ahmed, and A. Abate, "Automated and formal synthesis of neural barrier certificates for dynamical models," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer International Publishing, 2021, pp. 370–388.
- [16] Q. Zhao, X. Chen, Z. Zhao, Y. Zhang, E. Tang, and X. Li, "Verifying neural network controlled systems using neural networks," in *25th ACM International Conference on Hybrid Systems: Computation and Control*, 2022, pp. 1–11.
- [17] H. Zhang, J. Wu, Y. Vorobeychik, and A. Clark, "Exact verification of relu neural control barrier functions," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 5685–5705.
- [18] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural Lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.
- [19] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, 2023.
- [20] S. Liu, C. Liu, and J. Dolan, "Safe control under input limits with neural control barrier functions," in *Conference on Robot Learning*. PMLR, 2023, pp. 1970–1980.
- [21] B. Dai, P. Krishnamurthy, and F. Khorrami, "Learning a better control barrier function," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 945–950.
- [22] H. Zhang, L. Niu, A. Clark, and R. Poovendran, "Fault tolerant neural control barrier functions for robotic systems under sensor faults and attacks," *arXiv preprint arXiv:2402.18677*, 2024.
- [23] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan, "How to train your neural control barrier function: Learning safety filters for complex input-constrained systems," *arXiv preprint arXiv:2310.15478*, 2023.
- [24] M. Anand and M. Zamani, "Formally verified neural network control barrier certificates for unknown systems," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 2431–2436, 2023.
- [25] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," in *International Conference on Learning Representations*, 2019.
- [26] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2021.
- [27] M. Tayal, H. Zhang, P. Jagtap, A. Clark, and S. Kolathaya, "Learning a formally verified control barrier function in stochastic environment," *arXiv preprint arXiv:2403.19332*, 2024.
- [28] A. J. Barry, A. Majumdar, and R. Tedrake, "Safety verification of reactive controllers for UAV flight in cluttered environments using barrier certificates," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 484–490.
- [29] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.