Safe Neural Control for Non-Affine Control Systems with Differentiable Control Barrier Functions

Wei Xiao, Ross Allen, and Daniela Rus

Abstract— **This paper addresses the problem of safety-critical control for non-affine control systems. It has been shown that optimizing quadratic costs subject to state and control constraints can be sub-optimally reduced to a sequence of quadratic programs (QPs) by using Control Barrier Functions (CBFs). Our recently proposed High Order CBFs (HOCBFs) can accommodate constraints of arbitrary relative degree. The main challenges in this approach are that it requires affine control dynamics and the solution of the CBF-based QP is sub-optimal since it is solved point-wise. To address these challenges, we incorporate higher-order CBFs into neural ordinary differential equation-based learning models as differentiable CBFs to guarantee safety for non-affine control systems. The differentiable CBFs are trainable in terms of their parameters, and thus, they can address the conservativeness of CBFs such that the system state will not stay unnecessarily far away from safe set boundaries. Moreover, the imitation learning model is capable of learning complex and optimal control policies that are usually intractable online. We illustrate the effectiveness of the proposed framework on LiDAR-based autonomous driving and compare it with existing methods.**

I. INTRODUCTION

Optimal control problems with safety requirements are central to increasingly widespread safety critical autonomous and cyber physical systems. Control barrier functions enforcing safety have received increased attention in recent years [1] [2] [3] due to their high computational efficiency in dealing with affine-control nonlinear systems.

Barrier functions (BFs) are Lyapunov-like functions [4], [5], whose use can be traced back to optimization problems [6]. More recently, they have been employed to prove set invariance [7], [8], [9] and for multi-objective control [10]. Tee et al. [4] proved that if a BF for a given set satisfies Lyapunov-like conditions, then the set is forward invariant. A less restrictive form of a BF, which is allowed to decrease when far away from the boundary of the set, was proposed by Ames et al. [1]. Control BFs (CBFs) are extensions of BFs for control systems, and are used to map a constraint defined over system states to a constraint on the control input. The CBFs

Wei Xiao and Daniela Rus are with CSAIL, MIT, Ross Allen is with the MIT Lincoln Lab. weixy@mit.edu, ross.allen@ll.mit.edu, rus@csail.mit.edu.

proposed by Ames et al. [1] and Glotfelter et al. [2] work for constraints that have relative degree one with respect to the system dynamics. Exponential CBFs [11] for arbitrarily high relative degree constraints employ input-output linearization and find a pole placement controller with negative poles. The high order CBF (HOCBF) proposed by Xiao and Belta [3] is simpler (to define) and more general than the exponential CBF [11].

Most works using CBFs to enforce safety are based on the assumption that the control system is affine in controls and the cost is quadratic in controls. The time domain is discretized, and the state is assumed to be constant over each time interval. The optimal control problem is sub-optimally reduced to a quadratic program (QP) in each time interval and the control is kept constant for the whole interval. Using this approach, the original optimal control problem is reduced to a (possibly large) sequence of QPs - one for each interval. There are two main challenges in the aforementioned CBF-based QP formulation: (i) the dynamics must be affine in control. Otherwise, the CBF-based optimization would be a sequence of nonlinear programs (NLPs) that are inefficient and hard to solve [12] [13]. (ii) the solution of the CBF-based QP is sub-optimal since the problem is solved point-wise.

In order to reformulate constrained optimal control problems as CBF-based QPs for non-affine control systems, one can augment the system with auxiliary dynamics such that the augmented dynamics would be affine in controls [14] [15]. This is achieved at the cost of higher-relative degree CBFs, and the eventual CBFs become integral CBFs [16] since the integral solution of the CBF-based QP is the control for the original non-affine control system. However, there is no formulaic procedure to define such auxiliary dynamics. The second challenge mentioned above can be addressed using the nonlinear model predictive control (NMPC) method [17] or the inverse optimal method [18]. However, this may lead to NLPs with computationally expensive solutions. Although linearization is possible in NMPC to decrease the complexity, it may come at the cost of loss of safety guarantees. Another way to improve the optimality is to employ imitation learning [19] [20]. This approach learns complex control policies that are hard to solve online, and maps the learned policies to system observations, such as the front-view RGB images in driving. The limitation of imitation learning is that there is no guaranteed safety. BarrierNet [21] has been proposed to equip learning systems with guarantees, but it does not work for non-affine control systems. CBFs have been used in neural Ordinary Differential Equations (ODEs)-based learning models [22] to equip them with guarantees [23] [24]. However,

these CBFs are either used to modify the model trainable parameters such that CBFs are to be considered during training [23] or not trainable in the model [24].

To address the problem of safety-critical control for non-affine control systems and improve the optimality of the solution, this paper contributes a continuous optimization learning method that employs imitation learning with guarantees. The proposed learning model is based on neural ODEs that excel in learning control policies [22]. We define CBFs enforcing safety for non-affine control systems to higher-order CBFs such that the eventual CBF constraints would be linear in decision variables. Then, we incorporate these higher-order CBFs into the neural ODEs as differentiable CBFs (in terms of the CBF parameters) that are trainable to address the conservativeness of CBFs. Finally, we show that this method works efficiently for non-affine control systems with safety guarantees. We illustrate our approach and compare with other methods on a LiDAR-based autonomous driving problem.

II. PRELIMINARIES

Definition 1: (*Class $\mathcal{K}$ function* [25]) A continuous function $\alpha : [0, a) \to [0, \infty), a > 0$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. A continuous function $\beta : \mathbb{R} \to \mathbb{R}$ is said to belong to extended class $\mathcal{K}$ if it is strictly increasing and $\beta(0) = 0$.

Consider an affine control system (assumed to be affine in control only in this section) of the form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} \tag{1}$$

where $\boldsymbol{x} \in X \subset \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times q}$ are Lipschitz continuous, and $\boldsymbol{u} \in U \subset \mathbb{R}^q$ is the control constraint set defined as ($\boldsymbol{u}_{min}, \boldsymbol{u}_{max} \in \mathbb{R}^q$):

$$U := \{\boldsymbol{u} \in \mathbb{R}^q : \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max}\}, \tag{2}$$

where the inequalities are interpreted element-wise.

A. Control Barrier Functions and BarrierNet

We first introduce the concept of control barrier functions that are control synthesis tools for safe autonomous systems, and then briefly introduce BarrierNet that enables end-to-end safe learning.

Definition 2: A set $C \subset \mathbb{R}^n$ is forward invariant for system (1) if its solutions for some $\boldsymbol{u} \in U$ starting at any $\boldsymbol{x}(0) \in C$ satisfy $\boldsymbol{x}(t) \in C, \forall t \geq 0$.

Definition 3: (*Relative degree* [25]) The relative degree of a (sufficiently many times) differentiable function $b : \mathbb{R}^n \to \mathbb{R}$ with respect to system (1) is the number of times it needs to be differentiated along its dynamics until any component of the control $\boldsymbol{u}$ explicitly shows in the corresponding derivative.

In this paper, we assume that if there exists $\boldsymbol{x}$ such that the control shows up in the derivative of b, then it shows up for all $\boldsymbol{x}$. Since function b is used to define a constraint $b(\boldsymbol{x}) \geq 0$, we will also refer to the relative degree of b as the relative degree of the constraint. For a constraint $b(\boldsymbol{x}) \geq 0$ with relative degree m, $b : \mathbb{R}^n \to \mathbb{R}$, and $\psi_0(\boldsymbol{x}) := b(\boldsymbol{x})$, we define a sequence of functions $\psi_i : \mathbb{R}^n \to \mathbb{R}, i \in \{1, \dots, m\}$:

$$\psi_i(\boldsymbol{x}) := \dot{\psi}_{i-1}(\boldsymbol{x}) + \alpha_i(\psi_{i-1}(\boldsymbol{x})), i \in \{1, \dots, m\}, \tag{3}$$

where $\alpha_i(\cdot), i \in \{1, \dots, m\}$ denotes a $(m-i)^{th}$ order differentiable class $\mathcal{K}$ function.

We further define a sequence of sets $C_i, i \in \{1, \dots, m\}$ associated with (3) in the form:

$$C_i := \{\boldsymbol{x} \in \mathbb{R}^n : \psi_{i-1}(\boldsymbol{x}) \geq 0\}, i \in \{1, \dots, m\}. \tag{4}$$

Definition 4: (*High Order Control Barrier Function (HOCBF)* [3]) Let $C_1, \dots, C_m$ be defined by (4) and $\psi_1(\boldsymbol{x}), \dots, \psi_m(\boldsymbol{x})$ be defined by (3). A function $b : \mathbb{R}^n \to \mathbb{R}$ is a High Order Control Barrier Function (HOCBF) of relative degree m for system (1) if there exist $(m-i)^{th}$ order differentiable class $\mathcal{K}$ functions $\alpha_i, i \in \{1, \dots, m-1\}$ and a class $\mathcal{K}$ function α_m such that

$$\sup_{\boldsymbol{u} \in U}[L_f^m b(\boldsymbol{x}) + L_g L_f^{m-1} b(\boldsymbol{x})\boldsymbol{u} + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x}))] \geq 0, \tag{5}$$

for all $\boldsymbol{x} \in C_1 \cap, \dots, \cap C_m$. In (5), the left part is actually $\psi_m(\boldsymbol{x})$, L_f^m (L_g) denotes Lie derivatives along f (g) m (one) times, and $O(b(\boldsymbol{x})) = \sum_{i=1}^{m-1} L_f^i(\alpha_{m-i} \circ \psi_{m-i-1})(\boldsymbol{x})$.

The HOCBF is a general form of the relative degree one CBF [1], [2], i.e., setting $m = 1$ reduces the HOCBF to the common CBF form:

$$L_f b(\boldsymbol{x}) + L_g b(\boldsymbol{x})\boldsymbol{u} + \alpha_1(b(\boldsymbol{x})) \geq 0, \tag{6}$$

and it is also a general form of the exponential CBF [11].

Theorem 1: ([3]) Given an HOCBF $b(\boldsymbol{x})$ from Def. 4 with the associated sets $C_1, \dots, C_m$ defined by (4), if $\boldsymbol{x}(0) \in C_1 \cap, \dots, \cap C_m$, then any Lipschitz continuous controller $\boldsymbol{u}(t) \in U$ that satisfies the constraint in (5), $\forall t \geq 0$ renders $C_1 \cap, \dots, \cap C_m$ forward invariant for system (1).

Many existing works [1], [11] combine CBFs for systems with relative degree one with quadratic costs to form optimization problems. Time is discretized and an optimization problem with constraints given by the CBFs (inequalities in (5)) is solved at each time step. Note that these constraints are linear in control since the state value is fixed at the beginning of the interval, therefore, each optimization problem is a quadratic program (QP) if the cost is quadratic in the control. The optimal control obtained by solving each QP is applied at the current time step and held constant for the whole interval. The state is updated using dynamics (1), and the procedure is repeated. Formally, suppose we wish to minimize a cost function $u^T H \boldsymbol{u}$ for system (1), where $H \in \mathbb{R}^{q \times q}$ is positive definite, the CBF-based QP is defined as follows. We partition a time interval $[0, T]$ into a set of equal time intervals $\{[0, \Delta t), [\Delta t, 2\Delta t), \dots\}$, where $\Delta t > 0$. In each interval $[\omega \Delta t, (\omega + 1)\Delta t)$ ($\omega = 0, 1, 2, \dots$), we assume the control is constant (i.e., the overall control will be piece-wise constant). Then at $t = \omega \Delta t$, we solve the QP:

$$\min_{\boldsymbol{u}(\omega \Delta t)} \boldsymbol{u}^T(\omega \Delta t) H \boldsymbol{u}(\omega \Delta t)$$

$$\text{s.t. } \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max},$$

$$L_f^m b(\boldsymbol{x}) + [L_g L_f^{m-1} b(\boldsymbol{x})]\boldsymbol{u} + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x})) \geq 0. \tag{7}$$

This method works conditioned on the fact that the dynamics (1) are affine in control. Otherwise, the above optimization would be a sequence of NLPs that are hard and inefficient to

solve. However, there are a lot of systems whose dynamics are not affine in control, such as the bicycle and drone models. In this paper, we show how we can efficiently guarantee safety for non-affine control systems, as well as directly infer safe control from high dimensional observations using imitation learning.

BarrierNet. A BarrierNet [21] is based on the CBF-based QP (7), and it incorporates the optimization as a trainable layer in the neural network. We can find the loss of the output (solution) of the optimization layer with respect to all the optimization hyper parameters using the Karush–Kuhn–Tucker conditions. Thus. all the parameters in the cost, such as H, and the class $\mathcal{K}$ functions in the HOCBF can be trained by the data instead of by hand-tuning. In this way, we can make the HOCBF adaptive to the observation of the system, as well as addressing the conservativeness introduced by HOCBFs. The HOCBF is differentiable in terms of its parameters in a BarrierNet, and thus, we call it a differentiable CBF. Similar to existing CBF methods, one significant limitations of the BarrierNet is that it cannot work for non-affine systems.

B. Neural ODEs

A neural ordinary differential equation (ODE) [22] is defined as :
$$\dot{\boldsymbol{x}}(t) = f_{\vartheta}(\boldsymbol{x}(t)), \tag{8}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state and $\dot{\boldsymbol{x}}$ denotes the time derivative of $\boldsymbol{x}$, $n \in \mathbb{N}$ is state dimension, $f_{\vartheta} : \mathbb{R}^n \to \mathbb{R}^n$ is a neural network model parameterized by ϑ. The output of the neural ODE is the integral solution of (8). It can also include external input (e.g. observation vector), where the model is defined as:
$$\dot{\boldsymbol{x}}(t) = f'_{\vartheta}(\boldsymbol{x}(t), \mathbf{I}(t)), \tag{9}$$

where $\mathbf{I}(t) \in \mathbb{R}^d$, $d \in \mathbb{N}$ is external input dimension, $f'_{\vartheta} : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ is a neural network model parameterized by ϑ.

The limitation of neural ODEs is that they have no safety guarantees for control systems, which prevents their applications for safety-critical systems. In this work, we address this issue using the proposed learning framework.

III. PROBLEM FORMULATION AND APPROACH

We consider a non-affine control system whose dynamics are defined as:
$$\dot{\boldsymbol{x}} = h(\boldsymbol{x}, \boldsymbol{u}), \tag{10}$$

where $h : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^n$ is locally Lipschitz continuous.

Objective: (Minimizing cost) Consider an optimal control problem for system (10) with the cost:

$$\min_{\boldsymbol{u}(t)} \int_0^T \mathcal{C}(||\boldsymbol{u}(t)||)dt + p_0||\boldsymbol{x}(T) - \boldsymbol{K}||^2 \tag{11}$$

where $T > 0, p_0 > 0, \boldsymbol{K} \in \mathbb{R}^n$, $|| \cdot ||$ denotes the 2-norm of a vector, $\mathcal{C}(\cdot)$ is a strictly increasing function of its argument. We may consider penalizing the errors of all the states with respect to $\boldsymbol{K}$ in the above to make it more general.

Safety requirements: System (10) should always satisfy a safety requirement:
$$b(\boldsymbol{x}(t)) \geq 0, \forall t \in [0, T], \tag{12}$$

where $b : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and has relative degree $m \in \mathbb{N}$ with respect to system (10).

Control constraints: The control of the real system should always satisfy control bounds in the form of (2).

Problem 1: Given a real-time observation $\boldsymbol{I} \in \mathbb{R}^d$ ($\boldsymbol{I}$ could be the sensor information or the state $\boldsymbol{x}$, $d \in \mathbb{N}$ is the dimension of the observation) for system (10), find an *online* control policy for system (10) such that the cost (11) is minimized, and constraints (12) and (2) are satisfied.

Approach: Our approach to solve Problem 1 is based on the proposed safety-guaranteed machine learning method. Specifically, we employ the Nonlinear Model Predictive Control (NMPC) method to estimate the optimal control $\boldsymbol{u}^*$ of Problem 1 *offline* given an observation $\boldsymbol{I}$. In this way, we collect labeled training data set in the form of $(\boldsymbol{x}, \boldsymbol{I}, \boldsymbol{u}^*)$ where $\boldsymbol{u}^*$ is the optimal control label corresponding to $(\boldsymbol{x}, \boldsymbol{I})$. Then, we construct a continuous optimization learning model using neural ODEs and BarrierNet that is trained by the collected data, and that can be deployed for online control. We provably show the safety guarantees of the learning-based control for the non-affine control system (10).

IV. SAFE NEURAL CONTROL

In this section, we show how we can solve Problem 1 using a machine learning-based method that can guarantee system safety. We start with a motivation example showing why existing CBF methods may fail to work for system (10).

A. Motivating Example

Consider a bicycle model defined as:
$$\begin{aligned} \dot{x} &= v \cos \theta, & \dot{y} &= v \sin \theta, \\ \dot{\theta} &= \frac{v}{l} \tan u_1, & \dot{v} &= u_2, \end{aligned} \tag{13}$$

where $\boldsymbol{x} = (x, y, \theta, v)$, $(x, y) \in \mathbb{R}^2$ denotes the location of the vehicle, $v \in \mathbb{R}$ denotes its linear speed, $\theta \in \mathbb{R}$ denotes its heading, $u_1 \in \mathbb{R}, u_2 \in \mathbb{R}$ are the two controls corresponding to steering wheel angle and acceleration, respectively. $l > 0$ denotes the distance between front and rear wheels.

Suppose we have a safety constraint for system (13) defined as:
$$(x - x_0)^2 + (y - y_0)^2 \geq r^2, \tag{14}$$

where $(x_0, y_0) \in \mathbb{R}^2$ denotes the location of the circular obstacle, and $r > 0$ denotes its size.

The relative degree of the safety constraint (14) is two. Thus, we may use a HOCBF with $m = 2$ as in Def. 4 to enforce it. Choosing the class $\mathcal{K}$ functions α_1, α_2 as linear functions, the corresponding HOCBF constraint in (5) in this case is:

$$(-2(x - x_0) \sin \theta + 2(y - y_0) \cos \theta) \frac{v^2}{l} \tan u_1$$
$$+ (2(x - x_0) \cos \theta + 2(y - y_0) \sin \theta) u_2 + 2\dot{b}(\boldsymbol{x}) + b(\boldsymbol{x}) \geq 0, \tag{15}$$

where $b(\boldsymbol{x}) = (x - x_0)^2 + (y - y_0)^2 - r^2$ and $\dot{b}(\boldsymbol{x}) = 2(x - x_0)v\cos\theta + 2(y - y_0)v\sin\theta$.

Note that the HOCBF constraint (15) is a nonlinear function of u_1. Therefore, the eventual CBF-based optimization would be a sequence of NLPs instead of QPs, which makes it hard to solve. One may argue that we can take $\tan u_1$ as a decision variable instead of u_1 to make the HOCBF constraint linear in decision variables. However, part of the cost function is to minimize u_1^2 instead of $\tan^2 u_1$. As a result, we may have a nonlinear cost function that still makes it become NLPs. Moreover, in some systems, like the quadrotor, the dynamics may include both the control and its quadratic term, which makes the decision variable transformation method further intractable. We show how we may efficiently guarantee safety for non-affine control systems in this work.

B. Continuous Optimization Learning Model

Given a safety constraint $b(\boldsymbol{x}) \geq 0$ whose relative degree is m for non-affine control system (10), we use a HOCBF to enforce it. We still define a sequence of $\psi_i(\boldsymbol{x}), i \in \{1, \ldots, m\}$ functions as in (3), where $\psi_m(\boldsymbol{x})$ is involved with the control $\boldsymbol{u}$ when combining it with system (10). Therefore, we rewrite $\psi_m(\boldsymbol{x})$ as $\psi_m(\boldsymbol{x}, \boldsymbol{u})$, where

$$\psi_m(\boldsymbol{x}, \boldsymbol{u}) = \dot{\psi}_{m-1}(\boldsymbol{x}, \boldsymbol{u}) + \alpha_m(\psi_{m-1}(\boldsymbol{x})). \quad (16)$$

The above constraint corresponds to the HOCBF constraint in (5) for affine-control system (1). The difference is that the constraint (16) is a nonlinear function of $\boldsymbol{u}$, while the HOCBF constraint in (5) is a linear function of $\boldsymbol{u}$ given the state $\boldsymbol{x}$.

In order to address the issue that the constraint (16) is a nonlinear function of $\boldsymbol{u}$, we may take an additional derivative of $\psi_m(\boldsymbol{x}, \boldsymbol{u})$ such that the derivative of $\psi_m(\boldsymbol{x}, \boldsymbol{u})$ would be linear in $\dot{\boldsymbol{u}}$ following the chain rule. Formally, we introduce a higher order CBF based on (16) in the form:

$$\psi_{m+1}(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}) = \dot{\psi}_m(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}) + \alpha_{m+1}(\psi_m(\boldsymbol{x}, \boldsymbol{u})), \quad (17)$$

where $\alpha_{m+1}(\cdot)$ is also a class $\mathcal{K}$ function. The above function is linear in $\dot{\boldsymbol{u}}$.

In (17), $\dot{\boldsymbol{u}}$ is undefined. We use a neural ODE to model it:

$$\dot{\boldsymbol{u}} = \pi_\vartheta(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{I}), \quad (18)$$

where $\pi_\vartheta : \mathbb{R}^n \times \mathbb{R}^q \times \mathbb{R}^d \to \mathbb{R}^q$ is any neural network, such as multi-layer perception (MLP), convolutional neural network (CNN), long-short term memory (LSTM) network, parameterized by ϑ. Recall that $\boldsymbol{I}$ is the observation of system (10).

We can then use training data set to train the neural ODE (18) that makes (17) non-negative. Thus, we can provably show the safety guarantees of this neural ODE controller. However, the neural ODE is not guaranteed to make (17) non-negative due to the uncertainties in training or the model generalization issue. In order to address this, we incorporate (17) into the neural ODE using a BarrierNet, and obtain what we call the continuous optimization learning model:

$$\begin{aligned}\dot{\boldsymbol{u}} &= BarrierNet(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}), \\ \boldsymbol{y} &= \pi_\vartheta(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{I}),\end{aligned} \quad (19)$$

where $\boldsymbol{y} \in \mathbb{R}^q$ is a latent variable. The above model is continuous in the sense that the number of layers could be interpreted as infinite and we could solve it in continuous time, which is usually compared with discrete learning models (such as CNNs, LSTMs, etc.) that have finite number of layers. For more discussions about continuous versus discrete learning models, please refer to [22]. $BarrierNet(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y})$ is defined as

$$BarrierNet(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}) = \arg\min_{\hat{\boldsymbol{y}}} ||\hat{\boldsymbol{y}} - \boldsymbol{y}||^2,$$

s.t.

$$\begin{aligned}\psi_{m+1}^{\vartheta_p}(\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{y}}) &\geq 0, \\ b_{min}(\boldsymbol{u}, \hat{\boldsymbol{y}}) &\geq 0, \\ b_{max}(\boldsymbol{u}, \hat{\boldsymbol{y}}) &\geq 0,\end{aligned} \quad (20)$$

where $\hat{\boldsymbol{y}} = \dot{\boldsymbol{u}}$ and $\psi_{m+1}^{\vartheta_p}(\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{y}}) = \dot{\psi}_m(\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{y}}) + \vartheta_p \alpha_{m+1}(\psi_m(\boldsymbol{x}, \boldsymbol{u}))$ corresponds to the CBF (17) with an additional trainable parameter $\vartheta_p > 0$ that can address the conservativeness of the CBF method [21]. In addition, we can make ϑ_p dependent on the observation $\boldsymbol{I}$ such that it is adaptive to the observation. We would also add trainable parameters to all the class $\mathcal{K}$ functions $\alpha_i, i \in \{1, \ldots, m\}$ in $\psi_m(\boldsymbol{x}, \boldsymbol{u})$. Since the CBF is differentiable in terms of the parameter ϑ_p in the BarrierNet, we call it a differentiable CBF (dCBF). The model structure is shown in Fig. 1.

Further, in the BarrierNet (20), $b_{min}(\boldsymbol{u}, \hat{\boldsymbol{y}}) = \hat{\boldsymbol{y}} + \boldsymbol{u} - \boldsymbol{u}_{min}$ and $b_{max}(\boldsymbol{u}, \hat{\boldsymbol{y}}) = -\hat{\boldsymbol{y}} + \boldsymbol{u}_{max} - \boldsymbol{u}$ are CBF constraints that enforce the lower control bound $\boldsymbol{u} \geq \boldsymbol{u}_{min}$ and upper control bound $\boldsymbol{u} \leq \boldsymbol{u}_{max}$ as given in (2), respectively. The inequalities are interpreted component-wise.

Let $odeint(\cdot)$ denote an ODE solver. Now, we provably show the safety guarantees of the continuous optimization learning model (19).

Theorem 2: Given an initial state $\boldsymbol{x}(0)$ that is safe to system (10) (i.e., $b(\boldsymbol{x}(0)) > 0$), and an initial control $\boldsymbol{u}(0)$ such that $\psi_m(\boldsymbol{x}(0), \boldsymbol{u}(0)) \geq 0$ as given in (16), any control $\boldsymbol{u}(t), t \geq 0$ with

$$\boldsymbol{u}(t) = odeint(\text{equation (19)}), \quad (21)$$

guarantees the safety (i.e., $b(\boldsymbol{x}(t)) \geq 0, \forall t \geq 0$) and control bound satisfaction of system (10).

Proof: We first show that the BarrierNet (20) is a sequence of QPs given $\boldsymbol{x}$ and $\boldsymbol{u}$. This is important as the ODE solver is intractable during training or inference if it is a sequence of NLPs. The $\psi_m(\boldsymbol{x}, \boldsymbol{u})$ in (16) is a nonlinear function of $\boldsymbol{u}$ for system (10). However, when taking a derivative of $\psi_m(\boldsymbol{x}, \boldsymbol{u})$, we have that $\dot{\psi}_m(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}})$ is a linear function of $\dot{\boldsymbol{u}}$ following the chain rule. Therefore, $\psi_{m+1}^{\vartheta_p}(\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{y}}) \geq 0$ in (20) is a linear function of $\hat{\boldsymbol{y}}$ given $\boldsymbol{x}, \boldsymbol{u}$. The remaining constraints in (20) are also linear functions of $\hat{\boldsymbol{y}}$, and the cost is quadratic in $\hat{\boldsymbol{y}}$. Therefore, the BarrierNet (20) is a sequence of QPs given $\boldsymbol{x}$ and $\boldsymbol{u}$.

Since $\psi_m(\boldsymbol{x}(0), \boldsymbol{u}(0)) \geq 0$, $\psi_{m+1}^{\vartheta_p}(\boldsymbol{x}, \boldsymbol{u}, \hat{\boldsymbol{y}}) = \dot{\psi}_m(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}) + \vartheta_p \alpha_{m+1}(\psi_m(\boldsymbol{x}, \boldsymbol{u})) \geq 0$ and $\vartheta_p > 0$, following Thm. 1, we have that $\psi_m(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq 0, \forall t \geq 0$. Further, $\psi_m(\boldsymbol{x}, \boldsymbol{u})$ is defined as in (16) with $\psi_i(\boldsymbol{x}), i \in$

A Safety-Guaranteed and Continuous Learning System

Fig. 1. A continuous optimization learning system with safety guarantees. We may define more trainable parameters, such as cost weights, in the BarrierNet (20). The model can guarantee safety that is enforced by dCBFs during training or inference in a non-overly-conservative way.

$\{1, \ldots, m-1\}$ recursively defined as in (3). Since $b(\boldsymbol{x}(0)) > 0$, we can always find class $\mathcal{K}$ functions $\alpha_i(\cdot), i \in \{1, \ldots, m-1\}$ such that $\psi_i(\boldsymbol{x}(0)) \geq 0$ [3]. Since $\psi_m(\boldsymbol{x}(t), \boldsymbol{u}(t)) \geq 0, \forall t \geq 0$ and $\psi_{m-1}(\boldsymbol{x}(0)) \geq 0$, we have that $\psi_{m-1}(\boldsymbol{x}(t)) \geq 0, \forall t \geq 0$ by Thm. 1. Recursively, we can can show that $\psi_i(\boldsymbol{x}(t)) \geq 0, \forall t \geq 0$ from $i = m-1$ to $i = 0$ using the same argument. Since $b(\boldsymbol{x}) = \psi_0(\boldsymbol{x})$, we have that system (10) is safety guaranteed (i.e., $b(\boldsymbol{x}(t)) \geq 0, \forall t \geq 0$). Along the same line, since the remaining constraints in the BarrierNet (20) are CBFs enforcing the control bound (2), we have that the control bound (2) is also satisfied. ∎

The initial control $\boldsymbol{u}(0)$ usually takes a zero vector. The selection of $\boldsymbol{u}(0)$ depends on the generalization of the model.

Remark 1 (Feasibility guarantees and robustness): The BarrierNet may become infeasible at a certain time due to the conflict between the CBFs for safety and control bound. This is still a challenging problem in the CBF method. One possible solution is to find an analytical feasibility constraint that is added to the optimization [26]. The system dynamics become affine in $\dot{\boldsymbol{u}}$ inside the neural ODEs. Thus, this feasibility constraint method may still work. Neural ODEs are generally robust to model uncertainties if the training data is reasonably sampled [22].

Example revisited. Now, let's reconsider the example in Sec. IV-A. Within the model (19), we introduce a higher order of CBF $\psi_2(\boldsymbol{x}, \boldsymbol{u}) := (-2(x - x_0)\sin\theta + 2(y - y_0)\cos\theta)\frac{v^2}{l}\tan u_1 + (2(x - x_0)\cos\theta + 2(y - y_0)\sin\theta)u_2 + 2\dot{b}(\boldsymbol{x}) + b(\boldsymbol{x}) \geq 0$. The HOCBF constraint corresponding to (17) in this case would take another derivative of $\psi_2(\boldsymbol{x}, \boldsymbol{u})$, and it is linear in $\dot{u}_1, \dot{u}_2$ that we choose as decision variables in (20), i.e., $\hat{\boldsymbol{y}} = (\dot{u}_1, \dot{u}_2)$. Therefore, the CBF-based optimization becomes a sequence of QPs within the neural ODE (19).

C. Training of the Model

In this subsection, we introduce how we may train the continuous optimization learning model (19). In this work, we focus on the imitation learning method.

In imitation learning, we need to have a nominal controller (such as nonlinear MPC) to generate optimal controls as training labels. The objective is to make the model imitate the nominal controller. A nominal controller for non-affine control system (10) usually involves solving NLPs that are computationally expensive. However, the trained model can

be implemented efficiently. The model input includes the system observation $\boldsymbol{I}$, state $\boldsymbol{x}$, and control $\boldsymbol{u}$, as shown in (19). The inclusion of the control $\boldsymbol{u}$ in the input is to equip the model with short memory. For each of the observation data $\boldsymbol{I}$ with the corresponding state $\boldsymbol{x}$, we can find an optimal control using the nominal controller that label this data. As a result, we can collect training data set in a diverse set of scenarios that system (10) may encounter.

The training of the model (19) involves solving the following optimization problem

$$(\vartheta^*, \vartheta_p^*) = \arg\min_{\vartheta, \vartheta_p} \mathbb{E}_{\boldsymbol{x}}(\ell(\boldsymbol{u}, \boldsymbol{u}_{nominal})) \quad (22)$$

where $\boldsymbol{u}$ is from the solution of (19)—parameterized by ϑ, ϑ_p in (20)—and $\boldsymbol{u}_{nominal}$ is the optimal control from the nominal controller. $\ell(\cdot, \cdot)$ is a similarity measurement, and $\mathbb{E}_{\boldsymbol{x}}$ is the expectation over all the training data. It has been shown that both the neural ODE and the BarrierNet could be trained using gradient descent in [22] and [21], respectively, which eventually enables end-to-end (from observation $\boldsymbol{I}$ to control $\boldsymbol{u}$) training of the model.

Synthesized Model. Existing CBF methods assume piecewise constant controls across discretized time intervals. Our proposed model (19) could relax this piece-wise constant assumption. This is achieved by incorporating the dynamics (10) into the model (19):

$$\begin{aligned} \dot{\boldsymbol{u}} &= BarrierNet(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{y}), \\ \boldsymbol{y} &= \pi_\vartheta(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{I}), \quad (23) \\ \dot{\boldsymbol{x}} &= h(\boldsymbol{x}, \boldsymbol{u}), \end{aligned}$$

Then, the neural ODE solver could export safe controls corresponding to each state. The resolution of the control depends on the solver. Specifically, adaptive ODE solvers, such as *adaptive_adams, dopri5* [22], could ensure the solution to be within certain error bounds.

Remark 2 (Complexity of Training): The complexity of the BarrierNet, i.e., the CBF-based QP, is $\mathcal{O}(q^3)$, where $q \in \mathbb{N}$ is the dimension of the decision variable. Therefore, the training complexity of the model (19) is higher than neural ODEs and other traditional neural networks. The complexity of the BarrierNet can be reduced using the batch QP solving method. Due to the existence of CBFs that guarantee safety during training, the model is harder to train. To simplify the training, We may train the model using two stages: (i) we

train the neural ODE without BarrierNet, and (ii) we train the BarrierNet in the model (19) while fixing all other training parameters. For simple tasks, like the overtaking in driving considered in this paper, we can train the model all at once.

V. CASE STUDIES

In this section, we consider a high-way driving scenario where the *ego vehicle* overtakes a vehicle in front of it referred to as the *preceding vehicle*. We use *fmincon* to solve the NLP for NMPC, use *QPFunction* [27] to solve the BarrierNet, and use *torchdiffeq* [22] with method *dopri5* to solve the neural ODE. All the code is implemented in *PyTorch* [28] and runs on a *AMD Ryzen Threadripper PRO 3975WX 32-Cores* computer.

Problem setup. The ego vehicle dynamics are non-affine in control, as defined in (13). The ego vehicle has onboard LiDAR observation I (100 distance points around the ego vehicle, as shown in Fig. 2(a)) to detect its preceding vehicle. The preceding vehicle is assumed to have a constant moving speed. The safety constraint between the ego and its preceding vehicle is obtained by a disk-covering approach. In other words, we use an off-the-center disk to cover the preceding vehicle, as shown in Fig. 2(a). The disk is designed such that no collisions will happen when the center of the ego vehicle stays outside the disk, and the corresponding safety constraint is $b(\boldsymbol{x}) = (x - x_0)^2 + (y - (y_0 - y_{\mathrm{off}})) - r^2 \geq 0$, where $(x_0, y_0) \in \mathbb{R}^2$ is the location of the preceding vehicle, and $y_{\mathrm{off}} \in \mathbb{R}$ is the offset of the disk. The objective of the ego vehicle is to achieve a desired speed while overtaking its preceding vehicle.

(a) Highway overtaking. (b) Open-loop validation.

Fig. 2. Highway overtaking problem setup and end-to-end learning model validation with testing data set.

Training data generation. We randomly sample initially safe positions with random heading and speed for the ego vehicle around the preceding vehicle. Then, we use a NMPC to find optimal controls for the ego vehicle that can make it safely overtake the preceding vehicle. We collect 201 different initial states, and each initial state will further generate 100 trajectory points using NMPC. The sampling time is 0.1s. Thus, each trajectory corresponds to 10s driving. In summary, the data includes 200 trajectories with optimal control labels as training data set and 1 trajectory as validation data set during training.

Model and training setup. The model π_ϑ in (19) is defined as a 6-layer MLP with shape (104, 128, 256, 64, 16, 2) followed

(a) Closed-loop control profiles. (b) Closed-loop safety profiles.

Fig. 3. Highway overtaking closed-loop testing comparisons between NMPC, neural ODE and our models. Safety is guaranteed if $b(\boldsymbol{x}(t)) \geq 0, \forall t$

by a BarrierNet and a neural ODE integration layer, as shown in Fig. 1. In addition to the LiDAR I and control $\boldsymbol{u}$, we take the heading θ and speed v of the ego vehicle as input for the model while ignoring its location since this is captured by the LiDAR. We use the *RMSprop* in the package *torch.optim* as our optimizer during training [28]. The training batch size is 20, and each batch has 10 time sequence trajectory points.

Results. We first compare the open-loop (Fig. 2(b)) and closed-loop (Fig. 3) testing results. The open-loop testing is based on the test data set without feedback states from dynamics, and it is actually not a good indicator for the performance. In the closed-loop testing, the trajectory from our model can stay close to the NMPC one, while the trajectory from neural ODE could easily violate the safety constraint, as shown in Fig. 3(b). Both our model and NMPC can guarantee safety.

We further present quantitative comparisons in Table I. The testing results are based on noisy LiDAR (40% noise magnitude of the LiDAR range), and are from 100 driving overtaking scenarios. Only the neural ODE model fails to guarantee safety, although it is very efficient. The NMPC is very computationally expensive, preventing it from real-time applications. Although linearization is possible in NMPC to decrease its complexity, it may lose guarantees. Our model is computationally efficient under both adaptive (*dopri5*) and fixed-time-step (*fixed_adams*) ODE solvers, which is tractable for online safe control. Moreover, our model will not make the system conservative (measured by the average value of the minimum distance with respect to the obstacle among all testing trajectories). The snapshots of one overtaking example is shown in Fig. 4, and it shows that collision happens under the neural ODE model, but not in our model.

VI. CONCLUSION & FUTURE WORK

This paper proposes a continuous optimization learning method with safety guarantees for safety-critical systems. The proposed method leverages differentiable control barrier functions and neural ordinary differential equations, and it works efficiently even for non-affine control systems. In the future, we will study the simultaneous modelling of the system dynamics, safety constraints, and control policy using the proposed framework.

Fig. 4. Snapshots of an overtaking simulation (with our proposed model). The trajectory from the neural ODE makes the ego vehicle collide with the preceding vehicle, and the ego vehicle in the snapshots is with safety guarantees as it is controlled by our proposed model.

TABLE I

SELF-DRIVING COMPARISONS BETWEEN NMPC, NEURAL ODE AND OUR MODEL. ITEMS ARE SHORT FOR CONSERVATIVENESS MEASUREMENT (CONSER.) THAT IS DEFINED BY THE AVERAGE VALUE OF THE MINIMUM DISTANCE WITH RESPECT TO THE OBSTACLE AMONG ALL TESTING TRAJECTORIES, SAFETY MEASUREMENT (SAFETY), COMPUTATION TIME AT EACH ITERATION UNDER ADAPTIVE/FIXED-TIME-STEP SOLVERS (COMPUTATION TIME), RESPECTIVELY.

METHOD	CONSER. (≥ 0 & $\downarrow$)	SAFETY (≥ 0)	COMPUTATION TIME (S)
NMPC	4.8	4.8	1.235
NEURAL ODE	-42.4 ± 4.0	-51.6	0.007/0.001
OURS	1.3 ± 0.2	0.7	0.038/0.006

REFERENCES

[1] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. of 53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[2] P. Glotfelter, J. Cortes, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE control systems letters*, vol. 1, no. 2, pp. 310–315, 2017.

[3] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *Proc. of 58th IEEE Conference on Decision and Control*, Nice, France, 2019, pp. 474–479.

[4] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier lyapunov functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009.

[5] P. Wieland and F. Allgower, "Constructive safety using control barrier functions," in *Proc. of 7th IFAC Symposium on Nonlinear Control System*, 2007.

[6] S. P. Boyd and L. Vandenberghe, *Convex optimization*. New York: Cambridge university press, 2004.

[7] J. P. Aubin, *Viability theory*. Springer, 2009.

[8] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Trans. on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[9] R. Wisniewski and C. Sloth, "Converse barrier certificate theorem," in *Proc. of 52nd IEEE Conference on Decision and Control*, Florence, Italy, 2013, pp. 4713–4718.

[10] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris, "Multi-objective control for multi-agent systems using lyapunov-like barrier functions," in *Proc. of 52nd IEEE Conference on Decision and Control*, Florence, Italy, 2013, pp. 1478–1483.

[11] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. of the American Control Conference*, 2016, pp. 322–328.

[12] T. D. Son and Q. Nguyen, "Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 7623–7628.

[13] J. Seo, J. Lee, E. Baek, R. Horowitz, and J. Choi, "Safety-critical control with nonaffine control inputs via a relaxed control barrier function for an autonomous vehicle," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1944–1951, 2022.

[14] J.-J. E. Slotine, W. Li *et al.*, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[15] W. Xiao, C. G. Cassandras, C. A. Belta, and D. Rus, "Control barrier functions for systems with multiple control inputs," in *2022 American Control Conference (ACC)*, 2022, pp. 2221–2226.

[16] A. D. Ames, G. Notomista, Y. Wardi, and M. Egerstedt, "Integral control barrier functions for dynamically defined control laws," *IEEE control systems letters*, vol. 5, no. 3, pp. 887–892, 2020.

[17] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2018.

[18] M. Krstic, "Inverse optimal safety filters," *IEEE Transactions on Automatic Control*, 2023.

[19] T.-H. Wang, A. Amini, W. Schwarting, I. Gilitschenski, S. Karaman, and D. Rus, "Learning interactive driving policies via data-driven simulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7745–7752.

[20] 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.

[21] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2289–2307, 2023.

[22] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 6572–6583.

[23] W. Xiao, T.-H. Wang, R. Hasani, M. Lechner, Y. Ban, C. Gan, and D. Rus, "On the forward invariance of neural odes," in *International conference on machine learning*. PMLR, 2023, pp. 38 100–38 124.

[24] Y. Huang, I. D. J. Rodriguez, H. Zhang, Y. Shi, and Y. Yue, "Fi-ode: Certified and robust forward invariance in neural odes," *arXiv preprint arXiv:2210.16940*, 2022.

[25] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, third edition, 2002.

[26] W. Xiao, C. Belta, and C. G. Cassandras, "Sufficient conditions for feasibility of optimal control problems using control barrier functions," *Automatica*, vol. 135, p. 109960, 2022.

[27] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a

layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.