On the Optimality, Stability, and Feasibility of Control Barrier Functions: An Adaptive Learning-Based Approach

Alaa Eddine Chriat[1] and Chuangchuang Sun[1]

***Abstract*— Safety has been a critical issue for the deployment of learning-based approaches in real-world applications. To address this issue, control barrier function (CBF) and its variants have attracted extensive attention for safety-critical control. However, due to the myopic one-step nature of CBF and the lack of principled methods to design the class-$\mathcal{K}$ functions, there are still fundamental limitations of current CBFs: optimality, stability, and feasibility. In this paper, we proposed a novel and unified approach to address these limitations with Adaptive Multi-step Control Barrier Function (AM-CBF), where we parameterize the class-$\mathcal{K}$ function by a neural network and train it together with the reinforcement learning policy. Moreover, to mitigate the myopic nature, we propose a novel *multi-step training and single-step execution* paradigm to make CBF farsighted while the execution remains solving a single-step convex quadratic program. Our method is evaluated on the first and second-order systems in various scenarios, where our approach outperforms the conventional CBF both qualitatively and quantitatively.**

I. INTRODUCTION

While (deep) learning-based approaches have become pervasive nowadays, safety issues limit their deployment in real-world applications, especially those involving humans in the loop. For example, autonomous driving vehicles should guarantee the safety of the drivers and other entities by following the driving rules. Other safety-critical applications can be found in industrial, medical, and household scenarios. Therefore, learning-enabled models should rigorously guarantee safety, and failing to do so can result in undesirable or even disastrous outcomes.

In recent years, the control barrier function (CBF [1]) has attracted extensive attention due to its forward invariance property and scalability of solving a convex quadratic programming (QP) such that many variants have been developed in different settings and application scenarios. Additionally, the combination of reinforcement learning (RL) and control barrier functions [1–9] attracts much attention for safety assurance and explorations by using CBF as the safety shield. Specifically, work in [10] integrates the CBF into the utility function of RL, while others have used neural networks to parameterize and learn the barrier function parameters [11, 12]. Moreover, some other works integrated model predictive control with CBF as a predictive safety filter for reinforcement learning[13]. However, while control barrier functions are widely investigated and studied, there are still major issues to address as follows. (i) The one-step forward nature, while rendering simplicity and scalability, also makes it myopic and fails to yield an optimal path. (ii)

[1]The authors are with the Aerospace Engineering Department, Mississippi State University, Starkville, MS 39759, USA. Emails: aec652@msstate.edu, csun@ae.msstate.edu.

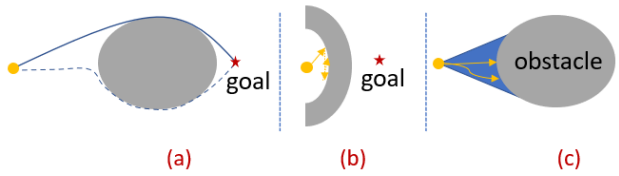

Fig. 1. Limitations of CBF. (a) One step is often myopic and thus generates an overall sub-optimal path (dashed) instead of the ideal/optimal one (solid). (b) When marching towards the goal driven by the control Lyapunov functions, the CBF agent gets stuck into the trap. (c) Limited translational/angular control input fails to avoid the obstacle for high-order systems.

The goal-reaching and safety guarantee, driven by control Lyapunov functions (CLF) and CBF, can often conflict with each other. (iii) The barrier function $\kappa(\bullet)$ is often manually designed (such as linear and quadratic candidates) and thus lacks expressiveness and adaptivity. Furthermore, such issues lead to the following concrete limitations; see the illustrations in Fig. 1. (1) It can often lead to an overall sub-optimal controller design, with "greedy" single-step control synthesis. (2) Because of the one-step planning nature to minimize the control Lyapunov function, it can easily get trapped in a concave safety set. For example, when an autonomous vehicle tries to go through an intersection of two convex obstacles, it can get stuck there due to the objective to minimize the CLF. (3) It can often encounter infeasibility [14] due to control limitations in high-order systems. In other words, the CBF constraint conflicts with the control constraints. A common example is the adaptive cruise control scenario when it is "too late to brake" when deceleration is limited such that collision cannot be avoided. We aim to address those fundamental challenges in CBF via a learning-based approach.

Learning and control approaches have been closely combined to mitigate their respective disadvantages while retaining their advantages. Modern control theory has rigorous guarantees of stability and constraint satisfaction with accurate dynamics models given. Such guarantees are often missing in partial-observable environments with pervasive noise and uncertainty. Moreover, the design of proper metrics, such as Lyapunov functions, is often case-by-case and requires expert knowledge. A principled way to design such metrics is desirable. As a result, data-driven learning-based control has attracted much attention in recent years. Methods are developed to learn the unmodelled dynamics and quantify the uncertainty, such as the Gaussian process [15–17]. Lyapunov function [18–20] and (neural) contraction metric [21–24] based methods are developed to guarantee the stability of the dynamical systems. As a result, a learning-based adaptive multi-step control barrier function method is proposed to

improve expressiveness, optimality, and feasibility for the control of safety-critical autonomous systems. Specifically, we propose to learn a class-$\mathcal{K}$ function in a principled way. Moreover, for the myopic nature of CBF, we propose a novel *muli-step training and single-step execution* paradigm. Intuitively, in training it considers a long horizon (instead of one step) and in execution/ inference, the advantage of single-step QP is kept. This, to the best of our knowledge, is the first systematic and unified approach to mitigate those fundamental issues.

II. PRELIMINARIES

A. High-order CBF

Control barrier functions are used in control theory to guarantee that a dynamical system can achieve some desired goals while remaining within safe constraints. A CBF is a function that quantifies the system's safety measurements. Hence, we aim to find a control input that keeps the system within its safe set measured by CBF. Mathematically, consider the nonlinear control-affine system:

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

where f and g are globally Lipschitz, $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the states and control inputs, respectively, constrained in closed sets, with initial condition $x(t_0) = x_0$.

Definition 1: [1]$h : \mathbb{R}^n \to \mathbb{R}$ is a barrier function for the set $C = \{x \in \mathbb{R}^n : h(x) \geqslant 0\}$ if $\exists$ an extended class-$\mathcal{K}$ function $\alpha(\bullet)$ such that:

$$\begin{aligned} &\sup_{u \in U}[L_f h(x) + L_g h(x)u + \alpha(h(x))] \geqslant 0 \\ &\inf_{\text{int}(C)}[\alpha(h(x))] \geqslant 0 \quad \text{and} \quad \lim_{\partial C} \alpha(h(x)) = 0 \end{aligned} \tag{2}$$

Because not all systems are first-order in inputs, we can use higher-order control barrier functions to constrain higher-order systems.

Definition 2: [25]For the nonlinear system (1) with the m^{th} differentiable function $h(x)$ as a constraint, we define a sequence of functions ψ_i with $i \in \{1, 2, ..., m\}$, starting from $\psi_0 = h(x)$:

$$\psi_i(x,t) = \dot{\psi}_{i-1}(x,t) + \alpha_i\left(\psi_{i-1}(x,t)\right) \tag{3}$$

and define $C_i(t)$ sequence of safe sets associated with each ψ_i:

$$C_i(t) = \{x \in \mathbb{R}^n : \psi_{i-1}(x,t) \geqslant 0\} \tag{4}$$

the function $h(x)$ is a high order control barrier function if there exist extended class-$\mathcal{K}$ functions $\alpha_i(\bullet)$ such that:

$$\psi_m(x,t) \geqslant 0 \tag{5}$$

CBFs have great potential in designing safe and robust systems, and they have been applied to various applications such as robotics, and autonomous vehicles.

B. Reinforcement learning

Reinforcement learning (RL) is to learn a policy for sequential decision-making from active interaction with the dynamic systems [26]. Such dynamic systems are often defined as Markov decision processes (MDP) that can either be fully or partially observable. An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, R, \gamma, P_0 \rangle$, where $\mathcal{S}$ is a set of agent states in the environment, $\mathcal{A}$ is a set of agent actions, $\mathcal{O}$ is a set of observations in partially observable case, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, R is the reward function, $\gamma \in [0, 1]$ is the discount factor and $P_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution. In the partially observable case, the agent receives an observation o_i correlated with the state s_i as $\mathcal{S} \mapsto \mathcal{O}$. A policy $\pi : \mathcal{S} \mapsto P(\mathcal{A})$ is a mapping from the state space to probability over actions. $\pi_\theta(a|s)$ denotes the probability of taking action a under state s following a policy parameterized by θ. The objective is to maximize the cumulative reward: $J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t \gamma^t R(s_t, a_t)]$, where τ are the trajectories sampled under $\pi_\theta(a|s)$. In order to optimize the policy that maximizes $J(\theta)$, the policy gradient with respect to θ can be computed as $\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) G(\tau)]$, where $G(\tau) = \sum_t \gamma^t R(s_t, a_t)$ [26]. The Q-function of a policy π is defined as $Q^\pi : \mathcal{S} \times \mathcal{A} \to R$ at any state action pair (s, a). Mathematically, for a policy π, $Q^\pi(s_0, a_0) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$ denotes the expected return of the trajectory. The policy can be deterministic in the form as $\mu_\theta : \mathcal{S} \mapsto \mathcal{A}$. As the objective gradient depends on the differentiation over actions, it requires continuous action space. With the policy parameters θ as deep neural networks (DNN), it is termed as *deep deterministic policy gradient* (DDPG) and can be used as a suitable instantiation of the RL algorithm for continuous control.

C. Differentiable convex programming

Differentiable convex programming is a technique that allows computation of the gradients of an optimization problem objective function with respect to the parameters of the problem, by taking matrix differentiation of the Karush-Kuhn-Tucker (KKT) conditions. One example of a differentiable optimization method is OPTNET [27], which has differentiable optimization problems within the architecture of the neural network. During training, the gradients of the objective function are back-propagated through the neural network. In general, we can use this method to differentiate through any disciplined convex program [28], by mapping it into a cone program first [29], computing the gradients, and mapping back to the original problem. A common example of differentiable programming is learning the constraints of the optimization problem such as convex polytopes or ellipsoid projections, through supervised learning. The advantage of differentiable optimization methods like OPTNET is that they can be used to optimize a wide range of convex objectives that are difficult to optimize using traditional optimization methods.

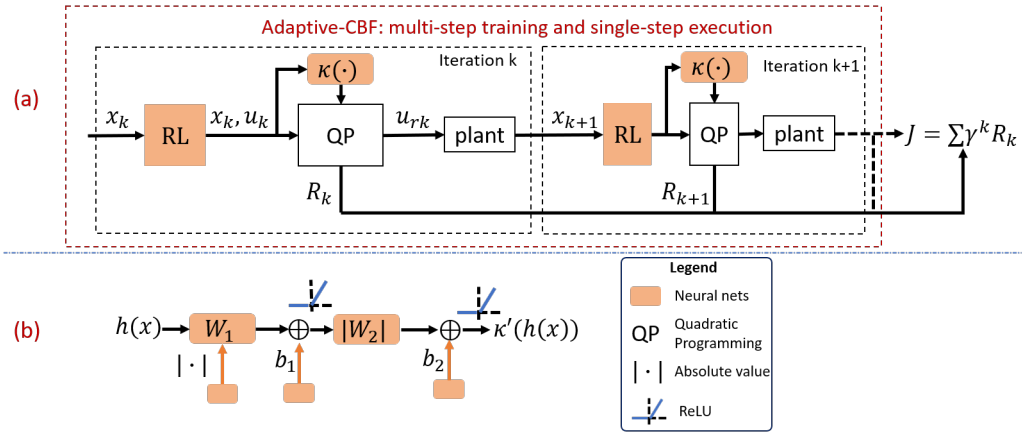

Fig. 2. Overview of the adaptive multi-step control barrier function (AM-CBF). (a) An end-to-end trainable multi-step CBF via differentiable programming. Back propagating through all the learnable modules, including the $\kappa(\bullet)$ within the quadratic programming, the return $J(\theta)$ will be maximized. (b) The neural network architecture to learn an adaptive class-$\mathcal{K}$ function.

III. APPROACH: ADAPTIVE MULTI-STEP CONTROL BARRIER FUNCTION

A. Learning-based CBF: a multi-step training and single-step execution paradigm

Control barrier functions have been used to enforce safety constraints in control systems. In reinforcement learning, CBFs can be used to ensure that an agent's actions satisfy safety constraints while maximizing a reward function. In general, CBFs can be used as a safety shield that projects an unsafe action into a safe one via the CBF conditions [2–4]. However, non-learning-based CBFs suffer the limitations described in Section I, which we aim to address here with its learning-based counterpart.

Consider the nonlinear system (1), the objective of safe reinforcement learning is to generate a policy/control u_r to achieve certain goals characterized by the reward function in the MDP while satisfying safety constraints. The typical way is to drive a potential function $V(x)$ to be zero, such as goal-reaching with $V(x) = \|x - x_f\|_2^2$. The RL policy will generate an action without safety guarantee first as $u_{\text{RL}}(t) = \mu\left(x_t \mid \theta^\mu\right) + \mathcal{N}_t$, where $\mu(\bullet \mid \theta^\mu)$ is a policy parameterized by deep neural networks θ^μ and $\mathcal{N}$ is a random process for promoting exploration. [1] Then the barrier function method [1] ensures that the controller complies with the safety constraint by solving the following convex quadratic program for control synthesis

$$\begin{aligned} &\min_{u_r \in [\underline{u}, \bar{u}]} && ||u_r - u_{\text{RL}}||^2 \\ &\text{s.t.} && \tfrac{\partial h(x)}{\partial x}\left(f(x) + g(x)u_r\right) \geq -\kappa(h(x)) \end{aligned} \tag{6}$$

where $\alpha > 0$ and $\kappa(\bullet)$ is an extended class-$\mathcal{K}$ function (strictly increasing and $\kappa(0) = 0$). Then like typical RL trajectory rollout, such process will be repeated for an episode length T; see Fig. 2(a). Unlike existing works in the literature using a manually designed class-$\mathcal{K}$ function, we propose to learn an extended class-$\mathcal{K}$ function parameterized by a neural network; see the illustration in Fig. 2(b). First, the class-$\mathcal{K}$ function is made expressive and adaptive with the parameterization of DNNs. Moreover, it should keep the property of a class-$\mathcal{K}$ function. (a) To make sure that $\kappa(\bullet)$ is monotonically increasing, the weights (excluding the bias) of the DNNs should be non-negative [30, 31], which is achieved by the absolute value (or exponential) activation function to guarantee $W_1 \geq 0$ and $W_2 \geq 0$. (b) By setting $\kappa(z) := \kappa'(z) - \kappa'(0)$, we guarantee that $\kappa(0) = 0$. Then the learned function $\kappa(\bullet)$ is guaranteed to be a class-$\mathcal{K}$ function. Moreover, we consider multi-steps of CBF in the rolling-out and training process of RL policies to address the infeasibility and sub-optimality issues. The intuition is that with the learning-based multiple-step planning, 1) it can have a more global view (instead of myopic) to achieve optimality, 2) it can be more foresighted and thus avoid getting stuck into the concave trap (stability), and (3) avoid the conflicts between CBF condition and control limitations (infeasibility). Hence, with a learned class-$\mathcal{K}$ function and a *multi-step training and single-step execution* paradigm, we mitigate the three fundamental issues of CBF described in section I and the overall of the AM-CBF is illustrated in Fig. 2.

Following the RL formalism, the policy $\mu(\bullet \mid \theta^\mu)$ and the class-$\mathcal{K}$ function $\kappa(\bullet)$ will be updated to maximize the cumulative reward function as

$$J(\theta) = \sum_{k=1}^{T} \gamma^k R(s_k, a_k). \tag{7}$$

Moreover, the temporal difference loss function used to train the critic network is as follows [32]

$$\begin{aligned} &\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left(\left(y - Q(s,a|\theta^Q)\right)^2\right. \\ &\text{where} \quad y = R + \gamma Q'\left(s, \mu'(s \mid \theta^{\mu'}) \mid \theta^{Q'}\right), \end{aligned} \tag{8}$$

where $\theta^{\mu'}$ and $\theta^{Q'}$ are the target networks of the actor and critic, respectively. Gradient descent-type algorithms are used to update the parameters θ^μ, $\theta^{\mathcal{K}}$, and θ^Q.

[1]The state x and s, the control/action u and a, terminologies in control theory and reinforcement learning, are used interchangeably here.

B. Gradient evaluation of the class-$\mathcal{K}$ function within QP via differentiable convex programming

To update the class-$\mathcal{K}$ function, it requires to differentiate through the QP in (6) to get the derivative of the loss function regarding $\theta^{\mathcal{K}}$. Note that the QP in (6) is convex and can be differentiated via the KKT conditions [27], which are *equivalent* conditions for (global) optimality. The KKT conditions state that at the optimal solution, the gradient of the Lagrangian function with respect to the program's input and parameters must be zero. Hence, by taking the partial derivative of the Lagrangian function with respect to the input and extending it via the chain rule to the program's parameters, we obtain all the gradients needed for training. Therefore it can be integrated seamlessly into the end-to-end training framework. We have integrated differentiable optimization using the cvxpylayers package [2] which is an extension to the cvxpy package with an affine-solver-affine (ASA) approach. The ASA consists of taking the optimization problem's objective and constraints and mapping them to a cone program. For a generalized QP

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x + q^T x \\ \text{s.t.} \quad & Ax = b \\ & Gx \leq h, \end{aligned} \tag{9}$$

we can write the Lagrangian of the problem as:

$$L(z, \nu, \lambda) = \frac{1}{2} z^T Q z + q^T z + \nu^T (Az - b) + \lambda^T (Gz - h) \tag{10}$$

where ν are the dual variables on the equality constraints and $\lambda \geq 0$ are the dual variables on the inequality constraint. Using the KKT conditions for stationarity, primal feasibility, and complementary slackness.

$$\begin{aligned} Qz^\star + q + A^T \nu^\star + G^T \lambda^\star &= 0 \\ Az^\star - b &= 0 \\ D(\lambda^\star)(Gz^\star - h) &= 0 \end{aligned} \tag{11}$$

By differentiating these conditions, we can shape the Jacobian of the problem as follows.

$$\begin{bmatrix} d_z \\ d_\lambda \\ d_\nu \end{bmatrix} = - \begin{bmatrix} Q & G^T D(\lambda^\star) & A^T \\ G & D(Gz^\star - h) & 0 \\ A & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \left(\frac{\partial \ell}{\partial z^\star}\right)^T \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

Furthermore, via chain rule, the derivatives of the loss function regarding any of the parameters in the QP, including the class-$\mathcal{K}$ function, are available [27]. This will enable end-to-end training for any learnable modules in this framework. This differentiable programming module is integrated into DDPG training process[3]. Moreover, during training, multiple tasks will be encountered and thus the resulting controller can be adaptive to different or even unseen tasks. Note that in execution, only one step of the QP in (6) is needed to solve (the same as normal CBF). As a result, this AM-CBF can address the critical limitations of existing CBF-based approaches and can lead to more adaptive, reliable, and safe controllers. Algorithm 1 summarizes the overall framework with the DDPG [32] and the learnable AM-CBF.

Algorithm 1 Safe reinforcement learning with AM-CBF

Require: Environment setting, learning rates α, β, discount factor γ, and target network update rate τ
Initialize critic network $Q\left(s, a \mid \theta^Q\right)$, actor $\mu\left(s \mid \theta^\mu\right)$ and $\mathcal{K}$-function network with weights θ^Q and θ^μ and $\theta^{\mathcal{K}}$
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $\mathcal{R}$
for episode $= 1, \ldots, M$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for $t = 1, \ldots, T$ **do**
    Select action $a_t = \mu\left(s_t \mid \theta^\mu\right) + \mathcal{N}_t$ according to the current policy and exploration noise
    Rectify the action via (6) for safe exploration
    Execute action a_{t_R} and observe reward r_t and new state s_{t+1}
    Store transition $(s_t, a_t, a_{t_R}, R_t, s_{t+1})$ in $\mathcal{R}$
    Sample a random mini-batch of N transitions $(s_t, a_t, a_{t_R}, R_t, s_{t+1})$ from $\mathcal{R}$
    Update critic by minimizing the loss in (8) with learning rate β
    Update the actor θ^μ and $\mathcal{K}$-function $\theta^{\mathcal{K}}$ using the gradient ascent with the sampled gradient of the return in (7)
    $\theta^\mu \leftarrow \theta^\mu + \alpha \nabla_{\theta^\mu} J(\theta)$
    $\theta^{\mathcal{K}} \leftarrow \theta^{\mathcal{K}} + \alpha \nabla_{\theta^{\mathcal{K}}} J(\theta)$
    Update the target networks with rate τ
    $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
  end for
end for
Return: $\theta^\mu, \theta^{\mathcal{K}}, \theta^Q$.

Remark. We now provide some thoughts on how our AM-CBF framework can address the limitations of vanilla CBF. (i) The joint training of policy and the safety shield (i.e., the $\mathcal{K}$ function) can collaboratively accomplish tasks while remaining safe, instead of purely depending on CBF. That means the unshielded action from RL can often be somewhat safe, with the jointly learned CBF to further guarantee safety. (ii) The multi-step training makes the safe policy far-sighted and predictive, to better avoid getting stuck in the local optima of the potential function (i.e., stability), to divert before exceeding the limit of the controller (i.e., feasibility), and eventually improve the performance of the whole trajectory (i.e., optimality). (iii) The learning-based $\mathcal{K}$ function allows for adaptiveness such that the trade-off between safety and conservatism is balanced.

IV. SIMULATIONS AND RESULTS

In this section, we evaluate the AM-CBF performance in two cases of a Dubin's car environment, a first-order

[2] https://github.com/cvxgrp/cvxpylayers

[3] https://github.com/philtabor/Youtube-Code-Repository/blob/master/ReinforcementLearning/PolicyGradient/DDPG/pytorch/lunar-lander/ddpg_torch.py

and a second-order system, and also a quadcopter system. The objective is to address the three research questions that have emerged from the limitations of the current CBFs. We compare our approach, which incorporates learning-based techniques, with non-learning-based CBFs. The comparison is carried out under the condition that all other settings and parameters are kept identical to ensure a consistent assessment. By examining these specific cases and answering the research questions, we aim to display the effectiveness and potential advantages of the AM-CBF approach in dealing with the inherent challenges of traditional CBFs.

A. Optimality

1) ***Dubins Car****:* To evaluate the AM-CBF performance on the optimality of trajectory, we used the first-order Dubins car environment with the following kinematics(13).

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix}, \tag{13}$$

where v_x is the velocity along the x axis of the car's frame, v_y is the sideways velocity, and ω is the angular velocity. To reach its final destination x_f from an initial state x_o, we designed a reward that penalizes the squared distance between the car and the goal state multiplied by a coefficient as $d\left\|x - x_f\right\|_2^2$, and penalizes every time step by a constant s for minimum time goal-reaching. Hence, the reward is defined as:

$$R = -d\left\|x - x_f\right\|_2^2 - s, \tag{14}$$

with $d > 0$ and $s \geq 0$. The discount factor γ, learning rates for training the actor and critic, and the update rates for the target networks are summarized in Tables I and II in the appendix.

Fig. 3 presents the trajectories from the AM-CBF class-$\mathcal{K}$ function, the linear class-$\mathcal{K}$ function, and the quadratic class-$\mathcal{K}$ function. It is shown that the linear CBF follows a myopic trajectory where it avoids the obstacle only after reaching it resulting in a sub-optimal path. The quadratic functions exhibit a similar behavior as it only avoids the obstacle after reaching it, but leaves a safer margin resulting in a moderately optimal path. Comparatively, the AM-CBF starts the avoidance from the initial state and clears the obstacle more optimally in terms of the shortest path.

Quantitatively, the reward functions of AM-CBF, linear $\mathcal{K}$-function CBF, and quadratic $\mathcal{K}$-function CBF cases are plotted in Fig. 4, where we can see that the non-learning-based CBFs approach have lower training time compared to the AM-CBF, with the quadratic function achieving slightly higher return than the linear one. However, the AM-CBF reaches the highest return value, which indicates the optimality of the trajectory and the shorter time to reach the final destination.

The final trained class-$\mathcal{K}$ function for the Dubins car is plotted alongside the linear and quadratic functions used in the CBF in Fig. 5. Intuitively, the learned function represents a piecewise affine function in the form of an increasing quadratic function, this result can be explained by relating

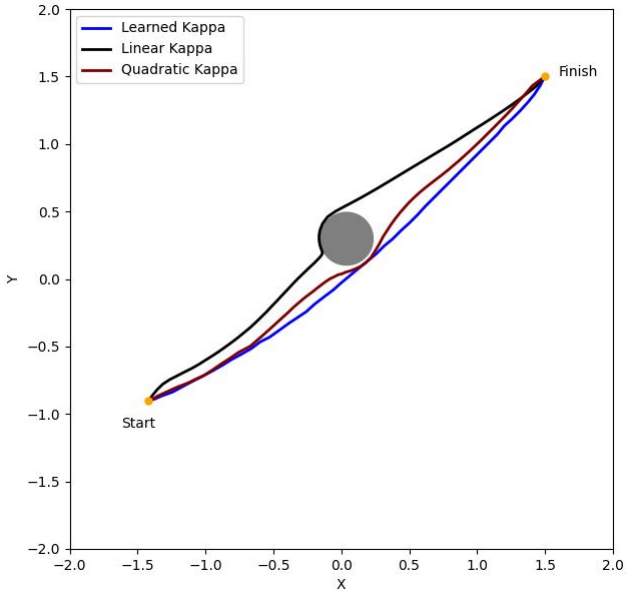

Fig. 3. Dubins car trajectories for learning based AM-CBF, the linear $\mathcal{K}$-function CBF, and the quadratic $\mathcal{K}$-function CBF.

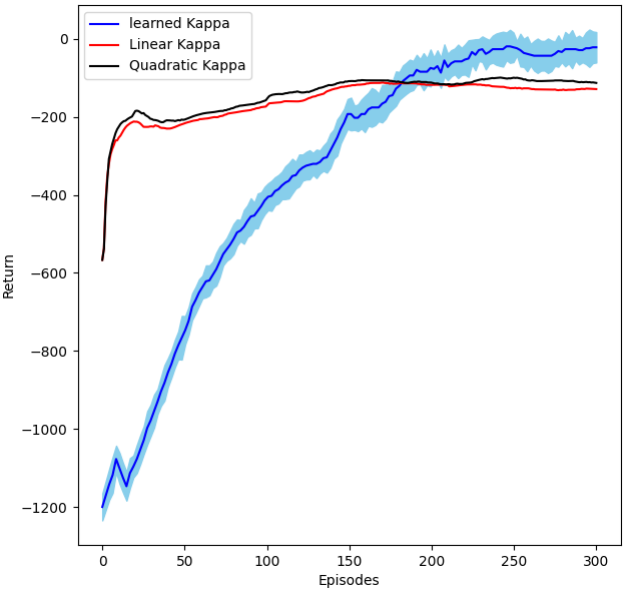

Fig. 4. Return comparison for Dubins car between AM-CBF, linear $\mathcal{K}$-function CBF, and quadratic $\mathcal{K}$-function CBF. The shadowed area denotes the variance from three runs with different random seeds.

the slope of the linear function to the safety degree desired. In the Linear case, the slope is constant for all values of the safety constraints operating the system on one safety level, while the learned function provides different slopes, allowing the safety degree to change depending on the state of the system.

2) ***Quadcopter****:* To inspect the performance of the AM-CBF on a system with more complex dynamics and safety constraints, we design high-level controllers for a heading-locked quadcopter environment with the following dynamics

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} T\sin\theta \\ T\cos\theta\sin\phi \\ T\cos\theta\cos\phi - g \end{pmatrix} \tag{15}$$

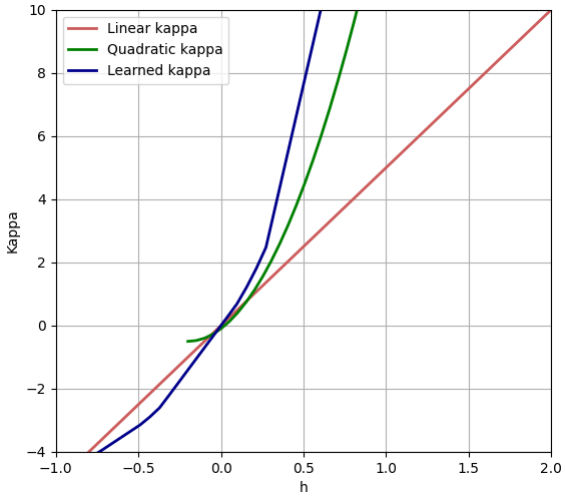

Fig. 5. $\mathcal{K}$-function learned from AM-CBF, the linear $\mathcal{K}$-function, and the quadratic $\mathcal{K}$-function.

where x, y, z are the inertial displacements of the quadcopter and θ, ϕ, T are the pitch, roll, and total thrust, respectively. While training, the quadcopter is penalized by the distance between the initial and terminal positions:

$$R = -\left\|x - x_f\right\|_2^2 \tag{16}$$

In this section, we simulate a landing scenario of the quadcopter within a prescribed glide slope defined by a cone with a half angle δ. The safety constraint can be written concisely as:

$$r_I^T M_{gs} r_I \geqslant 0 \tag{17}$$

where $M_{gs} = [[-\cot(\delta_{gs})^2, 0, 0]^T, [0, -\cot(\delta_{gs})^2, 0]^T, [0, 0, 1]^T]$ and $r_I = [x, y, z]$ is the inertial position of the quadcopter.

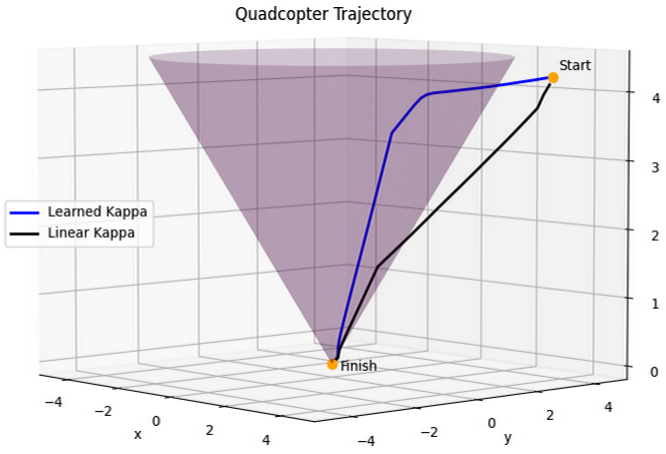

Fig. 6. Quadcopter trajectories for learning based AM-CBF and linear $\mathcal{K}$-function CBF.

Fig.6 presents the trajectories generated using the AM-CBF and the linear $\mathcal{K}$-function CBF. We can see that although both trajectories start from an unsafe position, the AM-CBF converges toward the safety region faster than the linear one while keeping a wider safety margin. Meanwhile, the linear CBF follows a steeper descent slope and slowly converges to the safety region inside the cone.

B. Stability

To evaluate the AM-CBF performance when encountering a concave obstacle, we created two overlapped circular obstacles to create a local minimum of the Lyapunov function that can possibly trap the car. Fig. 7 shows how the linear CBF gets attracted to the contact point and gets stuck there, while the AM-CBF adapts and learns how to avoid the obstacle and the trap. The return for the AM-CBF concave obstacle is plotted in Fig. 8, we can see some instability at the beginning of the learning but it smoothes out and reaches the optimal reward. The linear CBF has no reward profile due to the failure for reaching and thus the truncation of episodes.

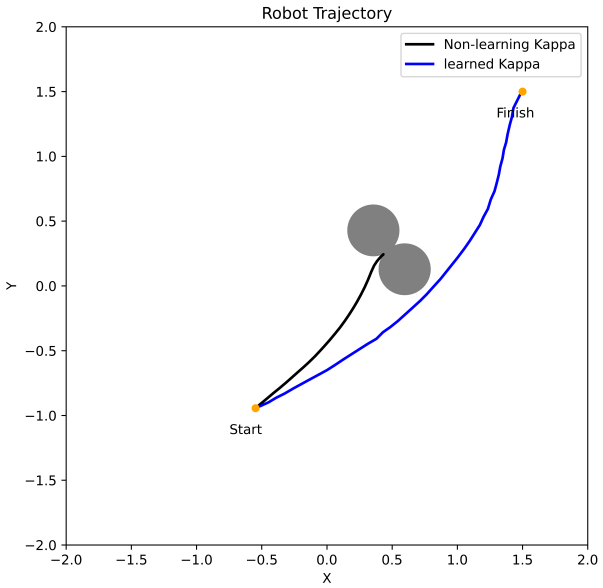

Fig. 7. The AM-CBF reaches its destination, while linear $\mathcal{K}$-function getting stuck in the trap.

While stability is more related to CLF, the one-step myopic CLF-CBF can often get stuck in the local optima of the CLF and cannot recover; see Figure 7. In this case, our far-sighted AM-CBF framework that jointly trains the policy and $\mathcal{K}$ function can divert early before getting stuck due to the exploration to achieve high cumulative rewards, which cannot be achieved if the agent gets stuck.

C. Feasibility

Infeasibility only happens in high-order systems with control constraints (e.g., upper/ lower bound). Hence, to create the infeasibility case, we use a second-order Dubin's car with the following kinematics

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} u_x \\ u_y \\ \tau_c \end{pmatrix}, \tag{18}$$

with norm constraint of the control input as $\|u\| \leq u_{\max}$. We also have the following adjusted reward function to penalize the velocities at the final destination for learning to brake as well

$$R = -d\left\|x - x_f\right\|_2^2 - b\left\|v - v_f\right\|_2^2 - s. \tag{19}$$

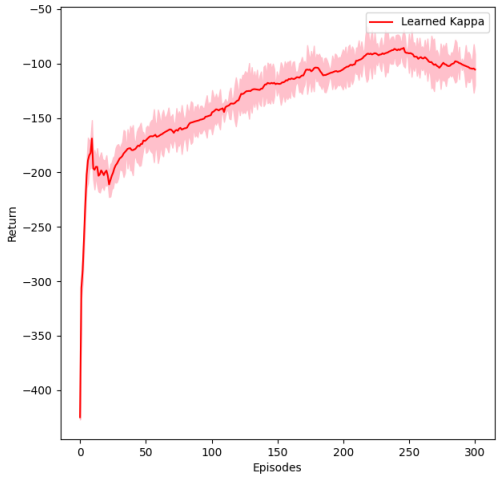

Fig. 8. The return profile from AM-CBF for the stability case. The shadowed area denotes the variance from three runs with different random seeds.

In Fig. 9, it is observed that the AM-CBF avoids the obstacle by diverging earlier with constrained input, while the linear CBF only tries to avoid the obstacle after reaching it, which results in infeasibility due to constrained inputs. The side zoom-in figure shows the direction of the car when the infeasibility arises in magenta, where the translational/ rotational control inputs (u_x, u_y, τ_c) are insufficient to brake/ turn enough to avoid the collision with the obstacle. Fig. 10 shows the reward profile from the AM-CBF, where we can see some oscillations at the start of the learning, smoothing out as the training progresses.

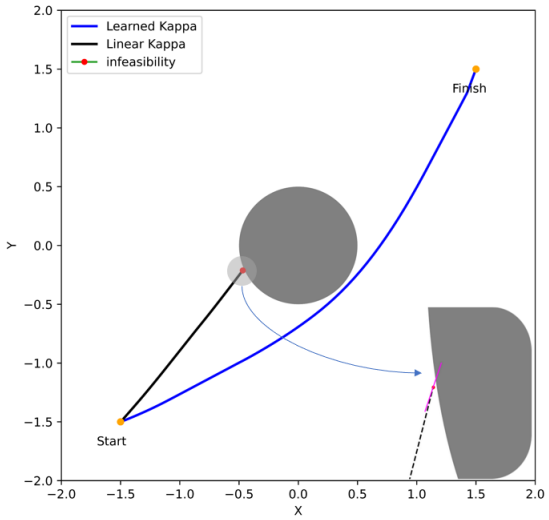

Fig. 9. The AM-CBF reaching its destination in blue, and linear $\mathcal{K}$-function violates the safety constraints next to the obstacle due to the conflicts between CBF conditions and the control constraints. The magnified magenta line shows the car's direction while encountering collision.

V. CONCLUSIONS

In this paper, we proposed a novel approach to address the optimality, stability, and feasibility of control barrier functions. Our approach is called the Adaptive Multi-step Control Barrier Function (AM-CBF), where we parameterize the class-$\mathcal{K}$ function by a neural network and train it together with the reinforcement learning policy. We evaluate our method on the first and second-order Dubin's car in various scenarios, where our approach outperforms the conventional linear class-$\mathcal{K}$ function both qualitatively and quantitatively. For future work, we plan to explore the generalization of our approach to meta-learning settings for fast adaptation to new tasks and also work on distributionally robust learning under distributional shift.

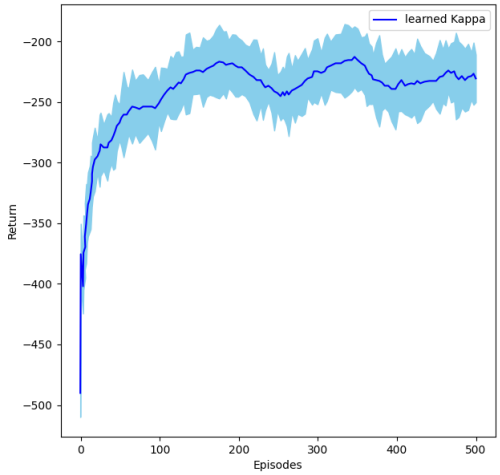

Fig. 10. Return profile from AM-CBF for the feasibility case. The shadowed area denotes the variance from three runs with different random seeds.

APPENDIX

We show the hyper-parameters in learning here in Tables I and II.

TABLE I

THE PARAMETERS USED IN THE DUBINS CAR

Parameter	description	Value
x_o	initial state	$-1.5 + \text{rand}, -1.5 + \text{rand}, \frac{\pi}{4}$
x_f	final state	$1.5 + \text{rand}, 1.5 + \text{rand}, \frac{\pi}{4}$
d	distance penalty	0.6
b	velocity penalty	0.1
s	step penalty	1
γ	discount factor	0.99

TABLE II

THE HYPER-PARAMETERS FOR TRAINING THE NEURAL NETWORKS

Parameters	Value
Actor-Critic networks hidden layers	$(128, 64)$
$\mathcal{K}$-function hidden layers	$(7, 7)$
batch size	64
Critic learning rate (β)	0.01
Actor learning rate (α)	0.001
Target update rate (τ)	0.7

REFERENCES

[1] 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, 2016.

[2] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3387–3395.

[3] L. Zheng, Y. Shi, L. J. Ratliff, and B. Zhang, "Safe reinforcement learning of control-affine systems with vertex networks," *arXiv preprint arXiv:2003.09488*, 2020.

[4] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," *arXiv preprint arXiv:2004.07584*, 2020.

[5] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," *arXiv preprint arXiv:1708.08611*, 2017.

[6] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods: Toward safe control through proof and learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[7] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal, "Safe reinforcement learning via curriculum induction," *arXiv preprint arXiv:2006.12136*, 2020.

[8] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[9] Y. Emam, G. Notomista, P. Glotfelter, Z. Kira, and M. Egerstedt, "Safe reinforcement learning using robust control barrier functions," *IEEE Robotics and Automation Letters*, no. 99, pp. 1–8, 2022.

[10] R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare, "Safe and efficient off-policy reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[11] Y. Yang, Y. Jiang, Y. Liu, J. Chen, and S. E. Li, "Model-free safe reinforcement learning through neural barrier certificate," *IEEE Robotics and Automation Letters*, 2023.

[12] 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*, 2023.

[13] K. P. Wabersich and M. N. Zeilinger, "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control," *IEEE Transactions on Automatic Control*, 2022.

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

[15] I. D. J. Rodriguez, U. Rosolia, A. D. Ames, and Y. Yue, "Learning unstable dynamics with one minute of data: A differentiation-based gaussian process approach," *arXiv preprint*, 2021.

[16] M. Khan, T. Ibuki, and A. Chatterjee, "Safety uncertainty in control barrier functions using gaussian processes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6003–6009.

[17] C. Peng and Y. Yang, "Trajectory tracking of a quadrotor based on gaussian process model predictive control," in *2021 33rd Chinese Control and Decision Conference (CCDC)*. IEEE, 2021, pp. 4932–4937.

[18] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Advances in neural information processing systems*, 2018, pp. 8092–8101.

[19] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, 2017, pp. 908–918.

[20] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," *arXiv preprint arXiv:1808.00924*, 2018.

[21] H. Tsukamoto, S.-J. Chung, and J.-J. Slotine, "Learning-based adaptive control via contraction theory," *arXiv*, 2021.

[22] H. Tsukamoto and S.-J. Chung, "Neural contraction metrics for robust estimation and control: A convex optimization approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 211–216, 2020.

[23] ——, "Learning-based robust motion planning with guaranteed stability: A contraction theory approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6164–6171, 2021.

[24] H. Tsukamoto, S.-J. Chung, J.-J. Slotine, and C. Fan, "A theoretical overview of neural contraction metrics for learning-based control with guaranteed stability," *arXiv preprint arXiv:2110.00693*, 2021.

[25] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3655–3662, 2021.

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[27] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," *arXiv preprint arXiv:1703.00443*, 2017.

[28] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[29] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi, "Differentiating through a cone program," *arXiv preprint arXiv:1904.09043*, 2019.

[30] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating functional knowledge in neural networks." *Journal of Machine Learning Research*, vol. 10, no. 6, 2009.

[31] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, 2018.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.