Estimating Control Barriers from Offline Data

Hongzhan Yu[1], Seth Farrell[1], Ryo Yoshimitsu[2], Zhizhen Qin[1], Henrik I. Christensen[1] and Sicun Gao[1]

***Abstract*— Learning-based methods for constructing control barrier functions (CBFs) are gaining popularity for ensuring safe robot control. A major limitation of existing methods is their reliance on extensive sampling over the state space or online system interaction in simulation. In this work we propose a novel framework for learning neural CBFs through a fixed, sparsely-labeled dataset collected prior to training. Our approach introduces new annotation techniques based on out-of-distribution analysis, enabling efficient knowledge propagation from the limited labeled data to the unlabeled data. We also eliminate the dependency on a high-performance expert controller, and allow multiple sub-optimal policies or even manual control during data collection. We evaluate the proposed method on real-world platforms. With limited amount of offline data, it achieves state-of-the-art performance for dynamic obstacle avoidance, demonstrating statistically safer and less conservative maneuvers compared to existing methods.**

I. Introduction

Control Barrier Functions (CBFs) provide an effective framework for safe robot control [1], [2].The recent development of learning-based CBF methods exploit the expressiveness of neural networks and data-driven approaches to handle systems with complex dynamics and high uncertainty, with promising results [3], [4], [5], [6], [7], [8], [9].

However, the scalability of learning-based methods has been a major bottleneck. The typical approach for learning neural CBFs requires sampling over the entire state space to enforce constraints from the standard CBF conditions [10]. While such methods ensure comprehensive coverage, they are impractical for high-dimensional domains due to the need of exponentially many samples. Online methods [11] have been proposed to mitigate this issue by interacting with the system during learning, allowing the state space to be gradually explored. However, the methods still rely on the availability of high-fidelity simulations and online interactions for learning at arbitrary system states, and can not operate with a fixed pre-collected dataset.

Consequently, *offline* training for constructing CBFs from a pre-collected dataset is an important and mostly open problem. Existing methods that consider the offline setting impose restrictive assumptions on the training data, such as only utilizing successful trajectories [7], [8], [10], or relying on a fixed performative controller for data collection [8], [12], [13], [14]. In contrast, leveraging unlabeled trajectories collected by untrained controllers, which is the more realistic setting of data collection, is key to achieving offline learning in practice. Thus, the core question is how to harness a small set of labeled data to train CBFs from a larger set of unlabeled suboptimally-collected behavior data of the robot (Figure 1). Note that given the learning-based setting of the problem, our focus is not to derive complete guarantee of safety, but to achieve better control performance using CBFs compared to existing methods.

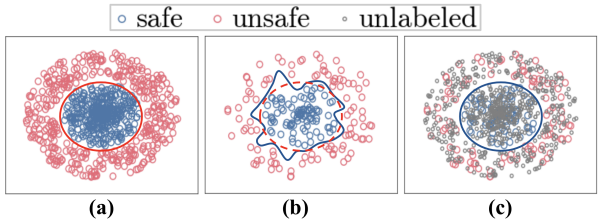

Fig. 1: Visualizations on toy datasets, to illustrate the motivation for utilizing unlabeled data. **(a)** With sufficient labeled data, the model can accurately capture the safety boundary. **(b)** When labeled data is limited, the learned boundary often misclassifies the safe and unsafe regions of the system. **(c)** Unlabeled data is generally more accessible than labeled data. Our approach leverages unlabeled data, along with the limited labeled data, to capture the CBF landscape that best adhere to the constraints inherent in the data.

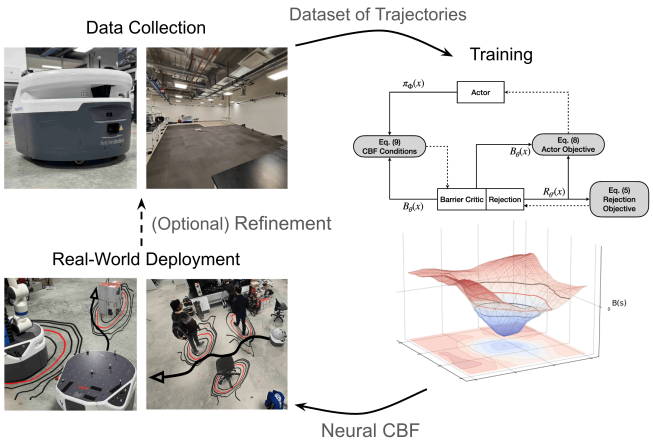

Fig. 2: Overall learning pipeline of the proposed method. Our method utilize offline demonstrations - even directly collected from real-world platforms - to construct neural control barriers, ensuring their zero-superlevel sets are control-invariant. The red contours denote the learned zero level set, serving as the safety boundary the ego-robot must not cross. Optionally, we gather additional demonstrations to further refine the barrier.

Overall, we introduce a novel offline learning framework for constructing neural CBFs (Figure 2). First, we propose new learning procedures that leverage out-of-distribution analysis [15] of trained neural models to propagate insights from labeled data, thereby maximizing the utility of

This paper has been accepted to ICRA 2025.

[1] Affiliation: 1. UCSD, 2. IHI Japan. Email: hoy021@ucsd.edu

unlabeled data. Second, we remove the reliance on high-performance controllers during data collection, enabling sub-optimal controllers to gather data while still achieving successful offline training. To achieve this, we improve the learning of CBFs to more closely align with their theoretical Lie-derivative condition. We evaluate the proposed method for obstacle avoidance with autonomous mobile robots on both simulation and real-world platforms. With limited amount of offline data, the proposed methods can achieve state-of-the-art performance in dynamic obstacle avoidance. The paper is organized as follows. Sections II and III cover the related work and the background necessary to support our theory. In Section IV, we introduce the proposed algorithm and explain the rationale behind each component. Section V presents the experimental results, and Section VI concludes the paper.

II. Related Work

Control Barrier Function. Control barrier functions (CBF) [16] aim to ensure control safety in dynamical systems by imposing value-landscapes to render the safe set forward invariant. The key point is to enforce the derivative of the CBF to satisfy Lyapunov conditions [17]. Traditional CBFs are manually designed based on domain-specific knowledge of the system, making them unsuitable for systems with complex dynamics or high uncertainty [1], [2].

Learning-based methods have been introduced to construct data-driven CBF candidates [3], [4], [5], [6], [8], [12], [18], [19] from data. Online algorithms learn CBFs by interacting with, or sampling from, the controlled system. In [3], the authors learn barrier certificates to derive the safe region of an unknown control-affine system. They propose an adaptive sampling algorithm to iteratively refine the CBF candidate on the states that have high uncertainty. [5] studies the multi-agent control problem. They jointly learn the barrier certificates alongside the multi-agent control policy, while regulating the policy based on CBF. [6] develops a model-based approach to learn control Lyapunov barrier functions based on stability and safety specifications. The training state are sampled uniformly from the state space. Offline algorithms learn CBFs without new data during the learning. [7] proposes an incremental learning of a set of linear parametric CBFs from human demonstrations. In [8], the authors present an approach to synthesize local valid CBFs for control-affine systems with known but nonlinear dynamics. The expert demonstrations contain only safe trajectories collected with a fixed nominal controller.

Out-of-distribution Analysis. Out-of-distribution (OOD) analysis is an emerging topic of machine learning that examines the distribution shifts where test data diverges from the training data distribution [15]. Unsupervised representation learning methods focus on learning domain-agnostic features from unlabeled data [20], [21], [22], [23]. However, these methods can introduce bias, if the OOD domain distributions overlap with the unlabeled data distribution [24]. Supervised learning methods incorporate implicit domain labels from both in-distribution and OOD data [25], [26]. While these methods are often more accurate due to the additional information, they may not generalize well to OOD examples that differ significantly from those seen during training.

III. Preliminary

We consider ego-robots with underlying dynamics $\dot{x}(t) = f(x(t), u(t))$ where $x(t)$ takes values in an n-dimensional state space $\mathcal{X} \subseteq \mathbb{R}^n$, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control vector, and $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is a Lipschitz-continuous vector field. We allow the dynamics function f to be generally nonlinear and not control-affine. Consider an unsafe region of the state space $\mathcal{X}_u \subset \mathcal{X}$ where safety constraints are violated. We say the system is safe if none of its trajectories intersects with $\mathcal{X}_u$. To enforce safety properties of a system, the controller needs to find a *control invariant* set for the system that is disjoint from the unsafe set. A subset of the state space $\mathsf{Inv} \subseteq \mathcal{X}$ is control invariant, if for any initial state $x(0) \in \mathsf{Inv}$ and any $t > 0$, we have $x(t) \in \mathsf{Inv}$. Namely, any trajectory that starts in the invariant set Inv stays in Inv forever. CBFs are scalar functions whose zero-superlevel set is a control invariant set within the safe region of the system, and whose spatial gradients can be used to enforce the invariance.

Definition 1 (Control Barrier Functions [16]): Consider a dynamical system defined by vector field $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. Let $B : \mathcal{X} \to \mathbb{R}$ be a continuously differentiable function. The Lie derivative of B over f is defined as:

$$L_{f,u}B(x) = \sum_{i=1}^{n} \frac{\partial B}{\partial x_i} \cdot \frac{\partial x_i}{\partial t} = \langle \nabla_x B(x), f(x,u) \rangle, \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. The Lie derivative measures the change of B over time along the direction of system dynamics under control u. If the zero-superlevel set of B, i.e. $\mathcal{C} = \{x \in \mathcal{X} : B(x) \geq 0\}$, is disjoint from the unsafe region of the system, i.e. $\mathcal{C} \cap \mathcal{X}_u = \emptyset$. And if for any safe state $x \in \mathcal{C}$ and an extended class-$\mathcal{K}_\infty$ function $\alpha(\cdot)$ [27]:

$$\max_{u \in \mathcal{U}} L_{f,u}B(x) \geq -\alpha(B(x)). \qquad (2)$$

Then B is a control barrier function (CBF), and its zero-superlevel set $\mathcal{C}$ is control invariant.

Out-of-distribution (OOD) algorithms study whether an input to the neural model follows the training distribution, being *in-distribution*, or deviates from the model's training set, making it *out-of-distribution*. In our work, we use the unsupervised algorithm [23] to implement OOD detection. Consider the input space $\mathcal{X}$ and the binary output space $\mathcal{Y} = \{-1, +1\}$. For a threshold $c \in (0, 1) \subseteq \mathbb{R}$, a binary classifier $P : \mathcal{X} \to (0, 1)$ is optimized to capture the correct labels, where the classification decision $f_{c,P}(x)$ is set to $+1$ only if $P(x) > c$. To achieve binary classification with rejection, [23] proposes learning two binary classifiers, denoted as P_1 and P_2, with thresholds c and $1 - c$, respectively, to satisfy Chow's rule [28]. The two models share all the model weights except for the last layer. If the two classifiers disagree on the classification of a given input x, i.e. $f_{c,P_1}(x) \neq f_{1-c,P_2}(x)$, the input is rejected as OOD.

IV. Offline Learning of Barrier Critic

Consider that we have a well-defined CBF $B : \mathcal{X} \to \mathbb{R}$, and a discrete control system $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. For an arbitrary unlabeled state $o \in \mathcal{X}$ that does not violate safety, i.e. $o \notin \mathcal{X}_u$, if there exist controls at o that can lead the system to be in the zero-superlevel set of B:

$$\exists u \in \mathcal{U} \ s.t. \ B(f(o,u)) > 0, \tag{3}$$

then the state o satisfies the control invariant property, and thus, assigning a safe label to it must be correct. However, for the data-driven neural CBF models, following (3) can lead to incorrect annotations of the unlabeled. This is because if an unlabeled state o is uncovered by the training set, it is likely that neither is its one-step reachable set, i.e. $\mathcal{X}^{'} = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} \ s.t. \ f(o,u) = x\}$, covered fully. Thus, the model predictions on $\mathcal{X}^{'}$ are not reliable in determining the safety of o. This phenomenon is critical for offline methods, as we cannot interact with the system to acquire on-policy data to refine on the wrongly-annotated state regions.

In our work, we propose to label an unlabeled state o as safe if there exists a control $u \in \mathcal{U}$ such that:

$$B_\theta(x') > 0 \ \textbf{\textit{and}} \ x' \text{ is } \textit{in-distribution} \text{ w.r.t. } \theta, \tag{4}$$

where $x' = f(o,u)$ and θ represents the parameters of neural CBF model. Intuitively, if we can derive such a control that leads the system to a *seen & safe* state, then there arises no concern about undermining the annotation steps due to the OOD samples. In the following sections, we describe the components for achieving the proposed idea.

A. Rejection-based Out-of-distribution Analysis

We employ [23] to determine whether a given input is in-distribution. Let the rejection model be denoted by $R_\Phi : X \to \mathbb{R}^2$, which outputs two-dimensional rejection scores for the given state input. Denote the rejection threshold by $c \in (0,1) \subseteq \mathbb{R}$. Over the safe set $\mathbb{X}_s$ and the unsafe set $\mathbb{X}_u$, we optimize R_Φ to minimize the following objective:

$$\begin{aligned} L_\Phi(\mathbb{X}_s, \mathbb{X}_u) &= L_{\Phi_1,c}(\mathbb{X}_s, \mathbb{X}_u) + L_{\Phi_2,1-c}(\mathbb{X}_s, \mathbb{X}_u), \\ L_{\Phi_i,(\cdot)}(\mathbb{X}_s, \mathbb{X}_u) &= \frac{1}{|\mathbb{X}_s|} \sum_{x \in \mathbb{X}_s} [-R_{\Phi_i}(x) + (\cdot)]_+ \\ &\quad + \frac{1}{|\mathbb{X}_u|} \sum_{x \in \mathbb{X}_u} [R_{\Phi_i}(x) - (\cdot)]_+, \end{aligned} \tag{5}$$

where $R_{\Phi_i}(x)$ denotes the i^{th} score from $R_\Phi(x)$ and $[\cdot]_+ = \max(\cdot, 0)$. When the rejection model is well trained, we say that the state $x \in \mathcal{X}$ is an in-distribution sample if:

$$R_{\Phi_1}(x) > c \ \textbf{\textit{and}} \ R_{\Phi_2}(x) > 1 - c, \tag{6}$$

implying no disagreement between the two rejection scores.

B. Actor Model Learning

The rejection model enables us to classify if a given state is in-distribution or not. However, to realize the annotation steps proposed in (4), we must be able to efficiently determine what controls to attempt at one unlabeled state.

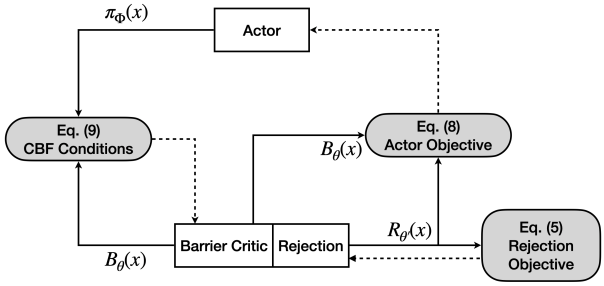

Fig. 3: Model component visualization with the proposed training flows. The optimization of the CBF requires the actor to derive the maximally-safe controls over which we enforce Lie derivative condition. The actor is optimized based on both CBF and rejection models, capturing the control that leads to the *safest in-distribution* state. Rejection model's training does not rely on other models.

We achieve this by learning an actor model $A_\Theta : \mathcal{X} \to \mathcal{U}$ that captures the maximally-safe, in-distribution control for the given state. The term 'maximally-safe' is with respect to the CBF landscape, accounting for the maximal increase to the learned CBF score led by the control. Denote the CBF model by $B_\theta : \mathcal{X} \to \mathbb{R}$. With the rejection model R_Φ and the parameter c, we aim at solving the following optimization problem with the actor at an arbitrary state $x \in \mathcal{X}$:

$$\begin{aligned} &\arg\max_{u \in \mathcal{U}} \ B_\theta(f(x,u)), \\ &\textit{s.t. } R_{\Phi_1}(f(x,u)) > c \textit{ and } R_{\Phi_2}(f(x,u)) > 1 - c. \end{aligned} \tag{7}$$

Consider that we obtain the control u^* by solving (7) at an unlabeled state o. If following u^* at o cannot satisfy (4), then no less safe control can satisfy it either. Therefore, we can label o as unsafe without evaluating any other controls. Given a training batch $\mathbb{X}$, we optimize Θ by minimizing:

$$\begin{aligned} L_\Theta(\mathbb{X}) = \frac{1}{|\mathbb{X}|} \sum_{x \in \mathbb{X}} \Bigg[&- B_\theta\Big(f(x, \pi_\Theta(x))\Big) \\ &+ [-R_{\Phi_1}\Big(f(x, \pi_\Theta(x))\Big) + c]_+ \\ &+ [-R_{\Phi_2}\Big(f(x, \pi_\Theta(x))\Big) + 1 - c]_+ \Bigg]. \end{aligned} \tag{8}$$

Unlike Reinforcement Learning (RL) methods [29], [30], we do not rely on the actor to generate controls at execution time. Instead, the actor is used soly as an auxiliary model to shape the barrier landscape during training.

C. Overall Pipeline

We now discuss the learning pipeline of the CBF model. We write the CBF model as $B_\theta : \mathcal{X} \to \mathbb{R}$. Given a safe batch $\mathbb{X}_s$ and an unsafe batch $\mathbb{X}_u$, we optimize θ by minimizing:

$$\begin{aligned} &L_\theta(\mathbb{X}_s, \mathbb{X}_u) \\ &= \left(\frac{1}{|\mathbb{X}_s|} \sum_{x \in \mathbb{X}_s} [-B_\theta(x)]_+ \right) + \left(\frac{1}{|\mathbb{X}_u|} \sum_{x \in \mathbb{X}_u} [B_\theta(x)]_+ \right) \\ &+ \frac{1}{|\mathbb{X}_s|} \sum_{x \in \mathbb{X}_s} \Big[- \Big\langle \nabla_x B_\theta(x), \nabla_x f(x, \pi_\Theta(x)) \Big\rangle - \alpha\Big(B_\theta(x)\Big) \Big]_+. \end{aligned} \tag{9}$$

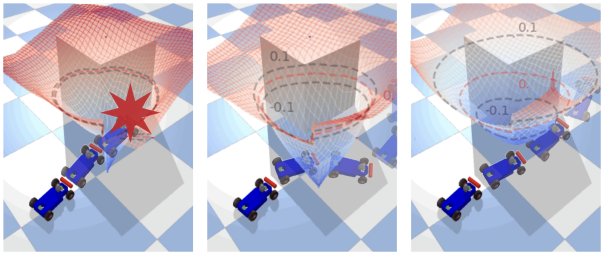

Fig. 4: Visualization of the learned CBF landscapes. Early in training, the safety boundary is primarily informed by the collision states from data, leading the robot to collide. As training progresses, the CBF model begins to approximate the safe region of the system. However, jittery robot motions are exhibited due to incomplete training. Once learning converges, the CBF model satisfies the Lyapunov condition over its Lie-derivatives, enabling the selection of more aggressive yet still safe controls.

The first two terms enforce $B_\theta(x)$ to take positive values on safe states and negative values on unsafe states, respectively. The third term optimizes the model to satisfy the Lie derivative condition of CBF in (2). Unlike prior work [8], [12], [19], which optimizes the Lie derivative condition over the safe controls from data, we optimize it over the controls generated by the actor π_Θ. In fact, optimizing with maximally-safe controls more closely follows the original CBF definition (2) which applies a $\max$ operator over the control space on the Lie derivative. In Section V-A, we show that incorporating the actor allows for training data collected with diverse controllers without imposing any performance assumption on them. Figure 3 illustrates the overall training flow of the proposed models.

We present the full procedures in Algorithm 1. Early in training, unlabeled data remain unannotated until a sufficient number of iterations have been completed (Line 4). This is to prevent false model estimations at the outset. To annotate an unlabeled state x, we unroll the dynamics function using the actor's control output to obtain the next state $\bar{x}$ (Line 14). We then label x as safe if and only if $\bar{x}$ is deemed safe with respect to the CBF model B_θ **and** in-distribution with respect to the rejection model R_Φ (Line 15).

D. Optimization Regularization

As the learning objective of the CBF model (9) enforces inequality constraints on the estimated landscape, the training process can suffer from the collapse problem similar to that reported in self-supervised learning [31]: as training proceeds, the magnitude of the learned landscape may shrink towards near-zero values while still violating the inequality constraints. This issue is especially pronounced for the offline methods, as they cannot acquire new data to mitigate the problem unlike the online methods.

To alleviate the collapse issue, we optimize (9) using the surrogate CBF values $\bar{B}_\theta$ defined as follows:

$$\bar{B}_\theta(x) = B_\theta(x) / \mathbb{E}_{x \in \bar{\mathbb{X}}}\big[B_\theta(x)\big], \tag{10}$$

where $\bar{\mathbb{X}} \subseteq \mathbb{X}_s$ is a subset of the safe set sampled in advance. We do not detach the gradient of the denominator in (10) with respect to model parameters θ, which helps to elevate the overall magnitude of barrier landscape whenever it begins to collapse toward zero.

Algorithm 1 Neural CBF with Barrier Critic (NCBF-BC)

Input: labeled sets D_s and D_u, unlabeled set D_{ul}, training iteration T, annotation start iteration T_a

1: *Initialize* the models and data buffer
2: **for** $t = 1$ ... T **do**
3: Sample labeled batches $\mathbb{X}_s \subseteq D_s$ and $\mathbb{X}_u \subseteq D_u$
4: **if** $(t \geq T_a)$ **then**
5: Sample an unlabeled batch $\mathbb{X}_{ul} \subseteq D_{ul}$
6: $\mathbb{X}_{ul,s}, \mathbb{X}_{ul,u} \leftarrow$ *Annotate*$(\mathbb{X}_{ul})$
7: $\mathbb{X}_s \leftarrow \mathbb{X}_s \cup \mathbb{X}_{ul,s}$, $\mathbb{X}_u \leftarrow \mathbb{X}_u \cup \mathbb{X}_{ul,u}$
8: **end if**
9: Model *updates* over $\mathbb{X}_s$ and $\mathbb{X}_u$: R_Φ with (5), π_Θ with (8), B_θ with (9)
10: **end for**
11: **return** B_θ, R_Φ, π_Θ

Function *Annotate*$(\mathbb{X})$:
12: $\mathbb{X}_s \leftarrow \{\}$, $\mathbb{X}_u \leftarrow \{\}$
13: **for** x in $\mathbb{X}$ **do**
14: $\bar{x} \leftarrow f(x, \pi_\Theta(x))$
15: **if** $B_\theta(\bar{x}) > 0$ **and** $R_\Phi(\bar{x})$ satisfies (6) **then**
16: $\mathbb{X}_s \leftarrow \mathbb{X}_s \cup \{x\}$
17: **else**
18: $\mathbb{X}_u \leftarrow \mathbb{X}_u \cup \{x\}$
19: **end for**
20: **return** $\mathbb{X}_s$, $\mathbb{X}_u$
EndFunction

V. Experiments

We evaluate the proposed algorithm in both simulation and real-world experiments. To derive safety-critical controls from the learned models, we follow Algorithm 2 which requires a heuristic goal-driven metric to rank controls according to task completion progress.

A. Simulation Experiments

We focus the simulation evaluations (Figure 4) on the task of static obstacle avoidance. We generate training trajectories using potential-field controllers with either *fixed* or *randomized* parameters. A trajectory is considered safe if no collision occurs. If a trajectory ends in a collision, we add the collision state to the unsafe set, and its preceding segment (of unlabel horizon τ) to the unlabeled set. In our simulation environments, where the time-step is discretized at $\triangle t = 0.2$ second, we set the unlabel horizon $\tau = 9$.

We consider three different ego-robot dynamics including Double Integrator, the Dubins, and the Bicycle models. For each dynamics model, the system is of 5-dimensions consisting of vehicle linear & angular velocities and yaw angles besides the coordinates. All the neural network models are 2-layer Tanh networks with 128 hidden neurons per layer. We use rejection parameter $c = 0.1$, and perform optimization

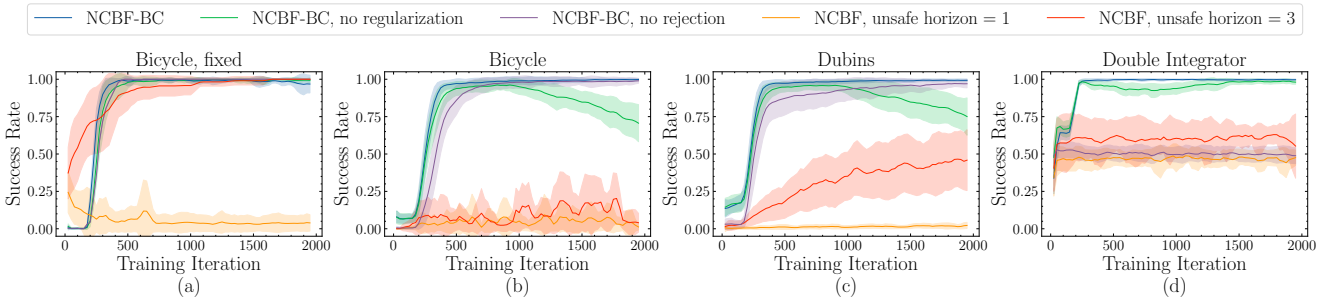

Fig. 5: Simulation experiments for static obstacle avoidance with different dynamics models of ego-robot. Evaluation metric is the mean success rate where we follow Algorithm 2 to derive the controls based on trained models, and perform evaluations over 100 randomized scenarios. When collecting the safe trajectories, we employ the potential-field controller(s) with **(a)** *fixed* and **(b)-(d)** *randomized* parameters.

regularization with subset size 1000. We employ the linear mapping $\alpha(x) = \kappa \cdot x$ with $\kappa = 0.1$. The algorithm runs for 2000 iterations in total, while the annotation of unlabeled data starts at the 200^{th} iteration. We use the orientation towards the goal as the heuristic goal-driven metric.

Comparisons with Baseline Methods. Figure 5 demonstrates the experiment results. The baseline is the standard method for learning CBFs as in [6], [8], [10], [12], [19], denoted by *Neural CBF* (NCBF). The *Unsafe horizon* defines the number of states near the end of failure trajectories which we label as unsafe, but only when training the baseline.

First, we show that the baseline underperforms when training trajectories are generated by multiple controller polices. When there are multiple controls provided at a state, it is incorrect to optimize the Lie derivative condition (2) of CBF over all the provided safe controls. Instead, we should optimize the condition only along the maximally-safe control, while those less conservative controls can be allowed to violate the inequality constraint in (2). In Figure 5(b)-(d), we show that our method can handle training sets collected by a diverse range of sub-optimal controllers.

Across all experiments, the proposed method incorporating both regularization and rejection-based annotation performs the best in terms of learning rates and training stability. For the Bicycle and Dubins models, the regularization technique effectively prevents collapse, thereby avoiding divergence during training. The Double Integrator does not exhibit collapse issues, regardless of whether regularization is applied. Because it has simpler dynamics, the optimization for satisfying the CBF conditions converges before collapse can occur. Meanwhile, disabling the rejection-based annotation noticeably slows convergence. In particular, for the Double Integrator, the CBF quickly overfits to the available labeled data, weakening the effectiveness of the annotation process that relies only on the learned CBF scores.

Ratio of Labeled Data. We conduct ablation experiments on the Bicycle model to investigate how the ratio of training set size impacts the proposed method. In Fig. 6(a), we vary the size of unlabeled set while fixing the labeled size, showing that the learning can be quickly stimulated even with small amounts of unlabeled data. This is because the labeled states that are certainly safe may be derived from conservative policies whose safety rules deviate from the optimal safety boundary. Meanwhile, the unlabeled trajectories with uncertain safety often involve aggressive controls that may exceed the optimal safety boundary, and thus carry more useful information. In Fig. 6(b), we fix the unlabeled size while varying the size of labeled set. With sufficient unlabeled data provided, there appears to be a threshold to the labeled size beyond which the learning rates become indifferent.

B. Hardware Experiments

In this section, we discuss hardware experiments on dynamic obstacle avoidance (Figure 8). The experiments on static obstacles (Figure 7 Right) are showcased in supplementary video but are not discussed in the paper. The

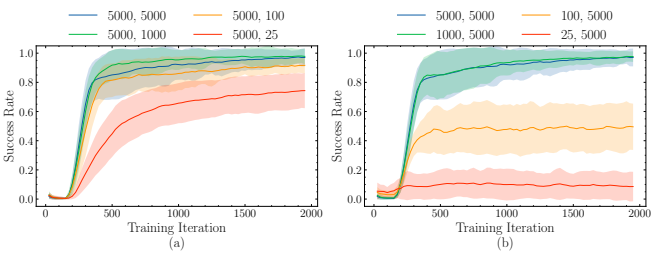

Fig. 6: Ablation experiments on the ratio between training labeled and unlabeled set sizes for Bicycle. The two numbers in each label are the sizes of labeled and unlabeled demonstration sets, respectively. For instance, the blue curve in (a) refers to the setting with 5000 labeled safe & unsafe states, and 5000 unlabeled states.

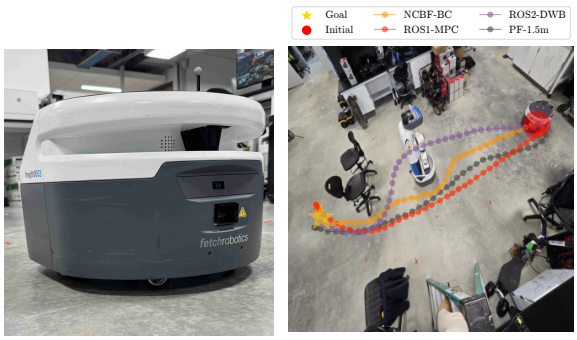

Fig. 7: **Left**: Freight robot. **Right**: Trajectories generated by different controllers.

	Success Rate (%)	Mean Path Length (meter)	Mean Completion Time (sec)	Mean Velocity (meter/sec)	Minimal Distance to Obstacles (meter)
NCBF-BC (ours)	**93.3**	7.23	36.46	**0.21**	**0.578**
NCBF	46.7	7.60	35.49	**0.21**	0.610
PF-1.5m	80.0	7.30	**35.06**	0.20	0.668
ROS1-MPC	80.0	7.75	44.65	0.17	0.697
ROS2-DWB	86.7	**6.68**	40.26	0.17	0.677

TABLE I: Real-world experiments for dynamic obstacle avoidance over 30 runs. Statistics are computed over the successful runs only.

platform utilized in our experiments is the Freight (Figure 7 Left), a research variant from Fetch Robotics. We cap the velocity of the robot at 0.22 m/s. To train our model, we collected 40-minutes of demonstrations by manually driving the robot around pedestrians, deliberately splitting the data into roughly 15-minutes of successful and 25-minutes of failure trajectories. For applying the proposed method, we discretize the time-step to be $\Delta t = 0.15$ second, and employ unlabel horizon $\tau = 9$.

The system state space for dynamic obstacle avoidance is 11-dimensional, consisting of robot coordinates, yaw and velocity information, and 3-step past state history of individual pedestrian. All the neural models are 2-layer Tanh networks with 256 hidden neurons per layer. We perform the learning for 5000 iterations, initiating the annotation steps over unlabeled data at the 500^{th} iteration. All other training parameters match those used in simulation experiments. When optimizing (9) to enforce the Lie derivative condition of CBF, we only leverage the derivative of ego-robot dynamics, while taking from data the pedestrian movements at the future timestamps.

Table I presents the quantitative results. Besides *NCBF*, we include one potential-field controller using a repulsive range of 1.5 meter. We further compare against both the ROS1 MoveBase (MPC-based) navigation stack and the ROS2 Nav2 stack. Offline Deep RL algorithms cannot be directly applied, as the reward labels that can accurately reproduce manual controls (via RL) are typically unavailable in real-world data. First, the proposed method achieves the highest success rate. The failures with our method are always due to unfamiliar pedestrian movements that deviate from the training data. Second, we show that the proposed method completes the scenarios with the highest mean velocity, while maintaining the lowest distance to the pedestrians without violating safety. This showcases the robustness of the learned safety boundary which allows us to select the controls that are performative, or even aggressive, yet safe. Third, the performance of potential-field controller degrades when there involve more pedestrians surrounding the robot. Oscillation behaviors are observed with potential-field controller in our experiments. Last, the NCBF baseline shows sub-optimal performance, producing strange looping behaviors and frequently taking unnecessarily long paths when pedestrians are present. Since data were collected via manual control, states could be reached with controls of varying levels of conservativeness. Consequently, the baseline's training objective forces the CBF to accommodate the most conservative control among the provided examples. Moreover, the NCBF baseline under-utilize failure trajectories, especially the uncertain states preceding the collisions, thereby limiting the amount of data it can effectively leverage.

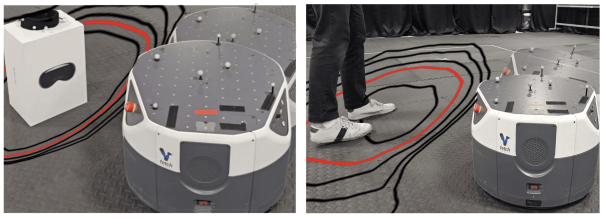

Fig. 8: Visualization of the learned CBF landscapes. Note that the CBF model trained for dynamic obstacle avoidance exhibits a wider gap between level sets, which reflects the need to initiate collision avoidance further from dynamic obstacles compared to static ones.

Algorithm 2 Control using NCBF-BC

Input: state x, CBF model B_θ, rejection model R_Φ, goal-driven metric $\mathcal{G} : X \to \mathbb{R}$, sample size N

Sample control candidates $\boldsymbol{a} = [a_1, a_2, ..., a_N]$
$\mathbf{g} \leftarrow []$
for a_i in $\boldsymbol{a}$ **do**
  Unroll the dynamics $\bar{x} = f(x, a_i)$
  if $B_\theta(\bar{x} < 0)$ **or** $R_\Phi(\bar{x})$ does not satisfy (6) **then**
    Remove a_i from $\boldsymbol{a}$
  else
    Evaluate the goal-driven score, $\mathbf{g} = \mathbf{g} \cup \{\mathcal{G}(\bar{x})\}$
  end if
end for
if $\boldsymbol{a}$ is now empty **then**
  return *Error - no safe control found*
end if
return control candidate from $\boldsymbol{a}$ with the maximal score

VI. Conclusion

This paper presents a novel learning-based approach for constructing neural control barriers from offline demonstrations. Experiment results show that the proposed algorithm outperforms the existing methods for offline data-driven CBFs construction. The results also highlight the benefits of utilizing unlabeled demonstrations to further refine the learning of control barriers Future directions include extending the proposed method to higher-dimensional systems, such as those incorporating Lidar sensory data or images.

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*, pp. 6271–6278, IEEE, 2014.

[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, 2016.

[3] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2460–2465, IEEE, 2018.

[4] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*, pp. 708–717, PMLR, 2020.

[5] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," *arXiv preprint arXiv:2101.05436*, 2021.

[6] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*, pp. 1724–1735, PMLR, 2022.

[7] M. Saveriano and D. Lee, "Learning barrier functions for constrained motion planning with dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 112–119, IEEE, 2019.

[8] 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)*, pp. 3717–3724, IEEE, 2020.

[9] H. Zhao, X. Zeng, T. Chen, Z. Liu, and J. Woodcock, "Learning safe neural network controllers with barrier certificates," *Formal Aspects of Computing*, vol. 33, pp. 437–455, 2021.

[10] Z. Qin, T.-W. Weng, and S. Gao, "Quantifying safety of learning-based self-driving control using almost-barrier functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12903–12910, IEEE, 2022.

[11] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[12] H. Yu, C. Hirayama, C. Yu, S. Herbert, and S. Gao, "Sequential neural barriers for scalable dynamic obstacle avoidance," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11241–11248, IEEE, 2023.

[13] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," in *Conference on Robot Learning*, pp. 1519–1539, PMLR, 2021.

[14] 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.

[15] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *International Journal of Computer Vision*, pp. 1–28, 2024.

[16] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[17] A. M. Lyapunov, "The general problem of the stability of motion," *International journal of control*, vol. 55, no. 3, pp. 531–534, 1992.

[18] S. Zhang, K. Garg, and C. Fan, "Neural graph control barrier functions guided distributed collision-avoidance multi-agent control," in *Conference on Robot Learning*, pp. 2373–2392, PMLR, 2023.

[19] C. Yu, H. Yu, and S. Gao, "Learning control admissibility models with graph neural networks for multi-agent navigation," in *Conference on Robot Learning*, pp. 934–945, PMLR, 2023.

[20] D. Mahajan, S. Tople, and A. Sharma, "Domain generalization using causal matching," in *International conference on machine learning*, pp. 7313–7324, PMLR, 2021.

[21] X. Zhang, L. Zhou, R. Xu, P. Cui, Z. Shen, and H. Liu, "Towards unsupervised domain generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4910–4920, 2022.

[22] S. Harary, E. Schwartz, A. Arbelle, P. Staar, S. Abu-Hussein, E. Amrani, R. Herzig, A. Alfassy, R. Giryes, H. Kuehne, *et al.*, "Unsupervised domain generalization by learning a bridge across domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5280–5290, 2022.

[23] N. Charoenphakdee, Z. Cui, Y. Zhang, and M. Sugiyama, "Classification with rejection based on cost-sensitive classification," in *International Conference on Machine Learning*, pp. 1507–1517, PMLR, 2021.

[24] H. Yu, X. Zhang, R. Xu, J. Liu, Y. He, and P. Cui, "Rethinking the evaluation protocol of domain generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21897–21908, 2024.

[25] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[26] Y.-X. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua, "Discovering invariant rationales for graph neural networks," *arXiv preprint arXiv:2201.12872*, 2022.

[27] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002. The book can be consulted by contacting: PH-AID: Wallet, Lionel.

[28] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on information theory*, vol. 16, no. 1, pp. 41–46, 1970.

[29] 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.

[30] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[31] L. Jing, P. Vincent, Y. LeCun, and Y. Tian, "Understanding dimensional collapse in contrastive self-supervised learning," *arXiv preprint arXiv:2110.09348*, 2021.