Safe control under input saturation with neural control barrier functions

Anonymous Author(s)
Affiliation
Address
email

Abstract: We propose new methods to synthesize control barrier function (CBF) based safe controllers that avoid input saturation, which can cause safety violations. In particular, our method is created for high-dimensional, general nonlinear systems, for which such tools are scarce. We leverage techniques from machine learning, like neural networks and deep learning, to simplify this challenging problem in nonlinear control design. The method consists of a learner-critic architecture, in which the critic gives counterexamples of input saturation and the learner optimizes a neural CBF to eliminate those counterexamples. We provide empirical results on a 10D state, 4D input quadcopter-pendulum system. Our learned CBF avoids input saturation and maintains safety over nearly 100% of trials.

Keywords: safety, control, learning

1 Introduction

In theory, control barrier functions are an appealing tool for safe control. However, it is difficult to make the derived controllers respect input limits, which reduces their usage in practice. CBFs target the *set invariance* class of safety problems, in which safety is defined as keeping a system's state to a prescribed region. A large part of their appeal is that they offer mathematical guarantees of safety. Such assurances are essential for safety-critical robotics applications, like collision-free drone flight [1, 2], manipulators that work safely around humans [3], and stable bipedal walking [4]. However, these safety guarantees break down when input saturation occurs, since that means the system cannot exert the force required for an evasive maneuver. The system then becomes endangered (or dangerous), with the possibility of expensive equipment failure or people getting harmed. It is therefore imperative that we account for input limits when designing CBFs.

CBFs are *energy functions* which map states to an energy value, with safe states having lower energy. In *energy function methods*, an energy function is found and then a controller is crafted that dissipates the energy. The core problem of these methods is constructing the energy function. A valid energy function has to meet complex constraints that depend on the input limits, system dynamics, and safety specification. So far, this problem of designing CBFs around input limits has been studied to a limited degree, mostly for small or simple systems. To our knowledge, nothing has been proposed which handles the nonlinear and high-dimensional systems that are more realistic in robotics. Examples of existing work include methods for designing CBFs using intuition for simple systems [5]. Other works derive CBFs that are provably non-saturating for systems with special structure, like kinematic bicycle, polynomial and Euler-Lagrange systems [6, 7, 8, 9, 10]. Another approach is to frame this as an optimization problem, computing parameters for CBFs with simple forms for use on low-dimensional systems [11, 12, 8]. A non-CBF safe control method with guarantees against saturation is HJ reachability [13]. It is generic (handles nonlinear systems), but has not been able to scale past 6D[1], except in special cases [15, 16].

[1] In general, synthesizing a safe controller for a nonlinear system with *formal guarantees* against saturation is an NP-hard problem [13]. Hence, it is intrinsically difficult to scale Hamilton-Jacobi (HJ) reachability.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

For the most part, existing works address narrow classes of systems, and some of them even require substantial human effort. In contrast, we envision a framework for *automating* CBF synthesis that has a *wide range of applicability*. Our goal is for it to be easy-to-use and generic, encompassing nonlinear systems of higher dimension. To achieve this, we borrow ideas from machine learning (ML), like neural networks, which are well suited to large-scale nonlinear optimization problems.

The related field of Lyapunov function (LF) synthesis has already taken inspiration from ML, with success. LFs are a different kind of energy function used for *certifying* stabilizing controllers, rather than constructing new controllers. Usually, it is assumed that the candidate stabilizing controllers already respect input limits, so input saturation is not considered. However, LFs are still hard to construct because they need to satisfy other complex constraints. Recent works have proposed representing LFs as neural networks (NNs), which are expressive and easy to optimize [17, 18, 19, 20]. To enforce constraint satisfaction over input sets, these neural LFs are either trained on a grid of inputs [20], or better yet, violating inputs [19]. There has also been work on neural CBFs, but they have been used for the entirely distinct problem of inferring a safety criterion from expert trajectories [21, 22].

We propose using *neural CBFs* to simplify CBF synthesis and extend its range of applicability. This choice of functional representation enables us to handle high-dimensional, nonlinear systems. In particular, NNs can easily handle systems with large state spaces (>6D), since they can be efficiently trained on much larger inputs than that (>100D). Further, for a nonlinear system, a valid CBF can be arbitrarily nonlinear, which demands the rich expressiveness of NNs. Finally, learning a non-saturating CBF for a nonlinear system involves a highly nonlinear loss function. Thus, we also need a functional representation that can be optimized for good solutions very easily, like NNs.

In the rest of the paper, we expound on our two main contributions, the first of which is our aforementioned neural CBF design (Sec. 3.1). The second contribution is our learning framework (Sec. 3.2). We begin by proposing a loss function that captures the severity of saturation. In our learner-critic paradigm, the critic finds states that maximize the loss (that is, worst saturators) and the learner updates the neural CBF to minimize saturation at those states. We also introduce a number of techniques that are required to make our learner stable and our critic efficient and powerful. Our method produces a CBF that ensures safety nearly 100% of the time for a input-limited, nonlinear, 10D quadcoptor-pendulum system (Sec. 4).

2 Preliminaries

In this section, we provide a review of CBFs, mathematically define a non-saturating CBF, and explain the premise of CBF synthesis. First, some notation: for a function $c : \mathbb{R}^n \to \mathbb{R}$, let $\mathcal{C} \triangleq \{c\}_{\leq 0}$ be its zero-sublevel set, $\partial\mathcal{C} = \{c\}_{=0}$ the boundary of this set, and $\text{Int}(\mathcal{C}) = \{c\}_{<0}$ the interior. Now, we assume the following is given: (1) a control-affine system $\dot{x} = f(x) + g(x)u$, where $x \in \mathcal{D} \subset \mathbb{R}^n, u \in \mathcal{U} \subseteq \mathbb{R}^m$ and $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous on $\mathbb{R}^n$, (2) input set $\mathcal{U}$, a bounded convex polytope, and (3) a safety specification $\rho : \mathbb{R}^n \to \mathbb{R}$, which implicitly defines the *allowable set* $\mathcal{A} \triangleq \{\rho\}_{\leq 0}$. ρ is continuous and piecewise smooth.

Typically, in CBF-based safe control, a CBF and *safe set* are constructed based on the user-defined safety specification and then a controller is formed to keep the safe set *forward invariant* (FI). A set is forward invariant if all trajectories starting within it stay within it. As such, let r be the *relative degree* from ρ to u (i.e. the first derivative of ρ where u appears) and then a standard *limit-blind CBF* is:

$$\phi = \left[\prod_{i=1}^{r-1}\left(1 + c_i\frac{\partial}{\partial t}\right)\right]\rho \qquad \text{(Limit-blind CBF)}$$

However, recent advances in HJ reachability have abandoned formal guarantees to create potential for scalability [14], which we have also done in our work. We note, however, that despite the lack of guarantees, our method is very effective in practice.

, where $c_i < 0$ [5]. This CBF has a corresponding safe set $\mathcal{S} \subseteq \mathcal{A}$; see the Appendix for its exact definition. Then, as usual, we can straightforwardly construct a safe controller. Any controller which can repel the system back into the safe set whenever it reaches the safe set boundary is a safe controller. The following theorem formalizes this idea:

Theorem 1 (Taken from [5]). *Given a CBF ϕ and safe set $\mathcal{S}$, any feedback controller $k(x) : \mathbb{R}^n \to \mathbb{R}^m$ satisfying*

$$\dot{\phi}(x, k(x)) \triangleq \underbrace{\nabla\phi(x)^\top f(x)}_{L_f\phi(x)} + \underbrace{\nabla\phi(x)^\top g(x)}_{L_g\phi(x)} k(x) \leq 0 \quad \forall x \in \partial\mathcal{S} \tag{1}$$

renders the system forward invariant over $\mathcal{S}$.

This theorem requires the controller to repel the system (decrease ϕ) from the boundary ($x \in \partial\mathcal{S}$). It allows us to follow an arbitrary nominal policy, k_{nom}, as long as we project the input to the half-space satisfying Eqn. 1. Thus, the standard CBF safe controller simply applies this projection at every timestep of control execution [5, 7]. This is implemented as a quadratic program (QP):

$$k(x) = \arg\min_u \ \frac{1}{2} \|u - k_{nom}(x)\|_2^2 \tag{CBF-QP}$$

$$\text{s.t.} \qquad L_f\phi(x) + L_g\phi(x)u \leq \begin{cases} 0 & \text{if } x \in \partial\mathcal{S} \\ \infty & \text{o.w.} \end{cases} \tag{2}$$

$$u \in \mathcal{U} \tag{3}$$

The significant problem with this paradigm is that this controller can saturate. Specifically, saturation occurs when no $u \in \mathcal{U}$ exists that satisfies Eqn. 2, causing the loss of safety guarantees.[2] This can happen because we have not yet accounted for input limits in our design of ϕ. What we propose is to craft a different CBF that does not lead to saturation down the line, which is the process of *CBF synthesis*. We will call this a *non-saturating CBF*:

Definition 1 (*Non-saturating CBF*). *A function $\phi : \mathbb{R}^n \to \mathbb{R}$ is a non-saturating CBF over a set $\mathcal{S}$ if for all $x \in \partial\mathcal{S}$:*

$$\inf_{u \in \mathcal{U}} \dot{\phi}(x, u) \leq 0 \tag{4}$$

Intuitively, this is just requiring that there exist a feasible control input to decrease ϕ (return the system to the interior of $\mathcal{S}$) at every state on the boundary, $\partial\mathcal{S}$. It is possible to find a non-saturating CBF ϕ^* that defines a smaller safe set ($\mathcal{S}^* \subseteq \mathcal{S}$) than the original saturating CBF. The idea is that a smaller set can exclude all the irrecoverable states from the original set. Formally, we want $\rho^* : \mathbb{R}^n \to \mathbb{R}$ such that $\{\rho^*\}_{\leq 0} \subseteq \{\rho\}_{\leq 0}$. Then, our new CBF is just the following:

$$\phi^* = \left[\prod_{i=1}^{r-1} \left(1 + c_i \frac{\partial}{\partial t}\right)\right] \rho - \rho + \rho^* \tag{Limit-aware CBF}$$

3 Learning Non-Saturating Control Barrier Functions

In this section, we present our neural CBF design (Sec. 3.1) and then discuss the learning framework which trains it to be non-saturating (Sec. 3.2). Learning is a min-max optimization problem of the following form, where θ denotes the parameters of ϕ^*:

$$\min_\theta \max_{x \in \partial\mathcal{S}^*} \mathcal{L}(\theta, x) \tag{5}$$

We call this loss $\mathcal{L}(\theta, x)$ the *saturation risk*. This min-max problem is solved using a learner-critic algorithm (Alg. 1), where the critic and learner alternately find where the worst saturation occurs and then update the CBF to reduce saturation there. We also propose some strategies for training stably, boosting critic efficiency, and for increasing the volume of the safe set. We visualize the training process for a simple running example (Fig. 1), where we learn a CBF to prevent an inverted pendulum from tipping.

[2] In practice, to avoid an unsolvable QP when saturation occurs, we add a slack variable to Eqn. 2. However, we will still violate safety guarantees.

Algorithm 1 Learning non-saturating CBF

function LEARN($\mathbb{X}_{ce}, \theta$)
  Set learning rate $\beta = 0.001$
  $\theta \leftarrow \theta - \beta \cdot \nabla_\theta \left[\sum_{x \in \mathbb{X}_{ce}} \text{softmax}(\mathcal{L}(\theta, x)) + \mathcal{R}(\theta)\right]$ ▷ From Eqn. 6, 7
  return θ
end function
function COMPUTECE($\mathbb{X}_{ce}, \theta$)
  Set learning rate $\alpha = 0.001$, number of gradient steps $N = 20$
  $\mathbb{X}_{ce} \leftarrow$ project $\mathbb{X}_{ce}$ to new boundary ▷ See Alg. 3
  $\mathbb{X}_{rand} \leftarrow$ uniformly sample a set on the boundary ▷ See Alg. 5
  $\mathbb{X} \leftarrow \mathbb{X}_{rand} \cup \mathbb{X}_{ce}$
  for i **in** $[0, \ldots, N]$ **do**
    $\mathbb{G} \leftarrow \nabla_{\mathbb{X}} \mathcal{L}(\theta, \mathbb{X})$ ▷ Batch gradient
    $\mathbb{P} \leftarrow$ project $\mathbb{G}$ along boundary
    $\mathbb{X} \leftarrow \mathbb{X} + \alpha \cdot \mathbb{P}$ ▷ Batch update
    $\mathbb{X} \leftarrow$ project $\mathbb{X}$ to boundary ▷ See Alg. 3
  end for
  $\mathbb{X}_{ce} \leftarrow$ worst saturating states in $\mathbb{X}$
  return $\mathbb{X}_{ce}$
end function
function MAIN( )
  Input: dynamical system $\dot{x}$, safety specification ρ
  Randomly initialize neural CBF parameters θ
  Repeat:
    $\mathbb{X}_{ce} \leftarrow$ COMPUTECE($\mathbb{X}_{ce}, \theta$) ▷ $\mathbb{X}_{ce}$ is a set of counterexamples
    $\theta \leftarrow$ LEARN($\mathbb{X}_{ce}, \theta$)
  Until convergence
  return θ
end function

3.1 Neural CBF Design

We consider the design of ρ^*. Let $\text{nn} : \mathbb{R}^n \to \mathbb{R}$ be a multilayer perceptron with $\tanh$ activations. Then, we define

$$\rho^*(x) = (\text{nn}(x) - \text{nn}(x_e))^2 + \rho(x) \tag{Design 1}$$

if there is a state, x_e, that should belong to any valid $\mathcal{S}^*$. Otherwise, let

$$\rho^*(x) = \text{softplus}(\text{nn}(x)) + \rho(x) \tag{Design 2}$$

The reason for these designs is that a valid ρ^* needs to obey three constraints: *Constraint 1*: ρ^* is piecewise smooth. This is required to preserve the piecewise smoothness of ϕ^*, which allows us to make existence and uniqueness arguments for the closed-loop system. Our ρ^* satisfies this because nn has smooth $\tanh$ activations and ρ is assumed piecewise smooth. *Constraint 2*: The 0-sublevel set of ρ^* is contained within the 0-sublevel set of ρ: $\{\rho^*\}_{\leq 0} \subseteq \{\rho\}_{\leq 0}$. The allowable set can be shrunk but not enlarged; otherwise, dangerous states may be incorporated. Our design meets this criterion because $\rho^* \geq \rho$. *Constraint 3*: $\mathcal{S}^*$ is nonempty, where $\mathcal{S}^*$ is defined by ρ^*. In some cases, it makes sense to enforce some state x_e to lie within $\mathcal{S}^*$ (Design 1), which means $\mathcal{S}^*$ will be nonempty by design. One such case is stabilization-type safety problems, where safety is defined as bounded deviation from an equilibrium. Then, the equilibrium would be a natural choice for x_e. However, if this is not the case, then we can choose Design 2 and encourage non-emptiness by regularizing the safe set volume (Sec. 3.2). Our CBF forms have the advantage of satisfying constraints 1-3 by design, while still offering the nonlinear expressiveness that we need to find a non-saturating CBF.

3.2 Learning framework

In the following sections, we present a loss function that encourages satisfaction of Eqn. 1 and the algorithm for solving the min-max problem on this loss. First, we define θ as the parameters of ϕ^*,

which include the weights of nn and the c_i coefficients. Our loss function, called *saturation risk*, is defined as:

$$\mathcal{L}(\theta, x) \triangleq \inf_{u \in \mathcal{U}} \dot{\phi}^*_\theta(x, u) \qquad \text{(Saturation risk)}$$

It is a measure of the best-case saturation at a given state x. When $\mathcal{L}(\theta, x) \leq 0$, then no saturation occurs at x; when $\mathcal{L}(\theta, x) > 0$, it measures how severe the saturation is. Thus, our min-max problem (Eqn. 5) is to minimize the *worst best-case saturation* over the boundary. When the worst best-case is negative, i.e.

$$\max_{x \in \partial \mathcal{S}^*} \mathcal{L}(\theta, x) \leq 0 \qquad \text{(Training goal)}$$

, then we have successfully found a non-saturating CBF.

To solve the min-max problem on $\mathcal{L}(\theta, x)$ (Eqn. 5), we propose a learner-critic algorithm (Alg. 1). Essentially, the algorithm alternates between the critic computing counterexamples (maximization with respect to x) and the learner updating the CBF (minimization with respect to θ). The critic uses projected gradient descent to produce an approximate maximizer, $\hat{x}^*$, and then the learner uses gradient descent to minimize the saturation loss at $\hat{x}^*$. See below for more details on the critic. Since both learner and critic perform gradient descent on $\mathcal{L}(\theta, x)$, it is useful to re-express it as an analytic function, rather than a continuous minimization. To find the analytic expression, observe that this $\mathcal{L}(\theta, x)$ a minimization where the objective is affine in the variable u (from Eqn. 1) and the variable is constrained to a convex polyhedron set $\mathcal{U}$. This means the minimizing u^* is one of the vertices $v \in \mathcal{V}(\mathcal{U})$ of the constraint set. Thus, $\mathcal{L}(\theta, x)$ can be computed as a discrete minimization, which is analytic:

$$\mathcal{L}(\theta, x) \triangleq \min_{v \in \mathcal{V}(\mathcal{U})} \dot{\phi}^*_\theta(x, v) \qquad \text{(Analytic saturation risk)}$$

In practice, we get more stable convergence if we use a *weighted average* of the loss over all counterexamples $\mathbb{X}_{ce}$ instead of the loss on the single worst counterexample. That is, the learner update becomes:

$$\theta = \theta - \alpha \cdot \nabla_\theta \left[\sum_{x \in \mathbb{X}_{ce}} \mathrm{softmax}(\mathcal{L}(\theta, x)) \right] \tag{6}$$

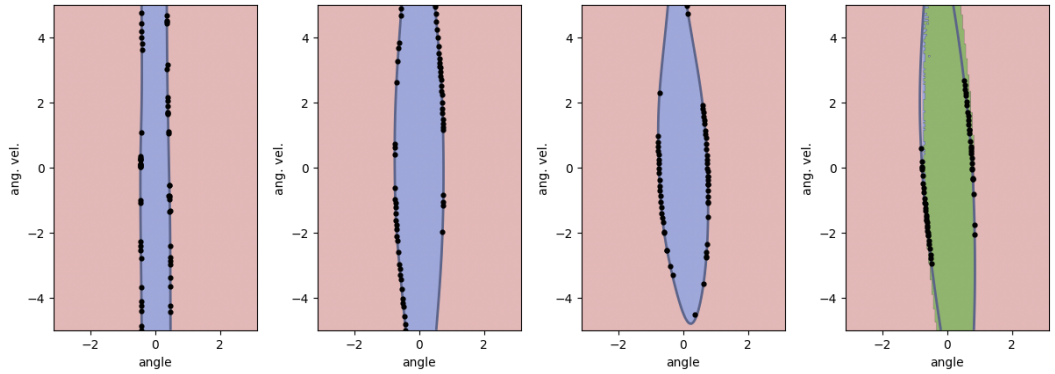
This avoids two problems which hinder convergence: (1) deadlock (when improvement at one counterexample causes saturation at another) and (2) noisy gradient ($\nabla_\theta \mathcal{L}(\theta, \hat{x}^*)$ only *approximates* the true gradient $\nabla_\theta \mathcal{L}(\theta, x^*)$, since $\hat{x}^*$ is a suboptimal maximizer). A weighted average loss both encourages progress on many counterexamples at once and averages out the effect of noisy gradients.

Design of the critic

In this section, we discuss the design of a fast and powerful critic, which is key to the success of the training algorithm (see function COMPUTECE in Alg. 1). The critic needs to return good counterexamples, otherwise the CBF will not train properly. However, it also needs to accomplish this efficiently, since it needs to recompute counterexamples repeatedly throughout training. Another challenge to speed is systems with large state spaces, since state space size is also the size of the critic's optimization space. To this end, we propose strategies for boosting critic efficacy cheaply.

The critic is in charge of finding counterexamples, which are states along the safe set boundary where saturation would occur at control-time. This is a constrained optimization problem, which we handle using projected gradient descent (PGD). We project the saturation risk's gradient along the boundary, use this to update x, and finally project x back to the boundary. However, in practice, this basic critic produces unsatisfactory counterexamples. There are many poor local optima in this optimization problem that can trap PGD. We explore two strategies for circumventing this issue: taking the best of a batch and warmstarting.

For the first strategy, we apply batch PGD to a *set* of initial states and take the best of the resulting candidate counterexamples. This batch optimization is a cheap way to improve our chances of



(a) Iter. 0 (b) Iter. 70 (c) Iter. 140 (d) Iter. 750

Figure 1: Learned safe sets for toy inverted pendulum problem, at four points during training. The black dots represent candidate counterexamples; the critic correctly identifies that the states where angle and angular velocity have the same sign are the most likely to saturate (angular velocity acting to destabilize pendulum). In (d), green denotes the largest safe set, approximated by model predictive control (MPC) (see Appendix for details). Our regularization is so effective that there is virtually no difference between our learned set and the MPC set.

evading local optima. A portion of the initial states is sampled *uniformly* on the boundary using the fast routine from [23]. The other initial states are "warmstarted": we take them to be the best counterexamples from the previous critic call. This strategy is based on the idea that old counterexamples can still be good candidates for the current call. The current call should only have a slightly different optimization objective and constraint than the previous call, due to the incremental update to θ. Overall, this reuse of work is a big boon to efficiency. Together, these two techniques help the critic perform efficiently and effectively, even for high-dimensional state spaces, as in Sec. 4.

Enlarging the safe set In this section, we introduce a regularization term that we add to the training objective to help enlarge the safe set. In general, one benefit of optimization-driven approaches to synthesis is that we can augment the objective with terms that target control *performance*. One measure of a good safe controller is the size of its safe set: a larger safe set means the controller can prevent more states from evolving toward danger. Thus, we devise a regularization term:

$$\mathcal{R}(\theta) \triangleq \sum_{x \in \mathbb{X}_{reg}} \text{sigmoid}(s^*_\theta(x)) \tag{7}$$

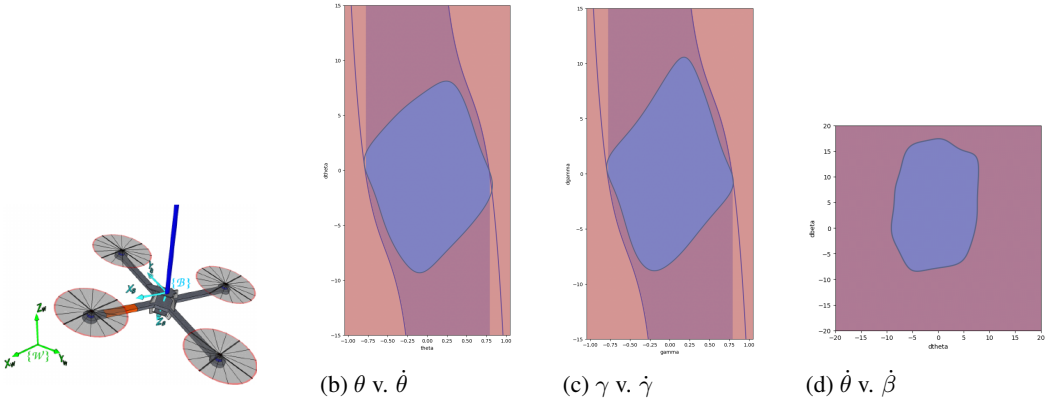
, where $s^*_\theta(x)$ defines inclusion in the safe set ($\mathcal{S}^* = \{s^*_\theta\}_{\leq 0}$) and $\mathbb{X}_{reg}$ is a set sampled uniformly in $\mathcal{D}$. The idea behind this term is that we want states near the safe set boundary to move inside the safe set (or further inside, if they are already). Mathematically, we want to decrease $s^*_\theta(x)$ toward negative values (move into $\mathcal{S}^*$) for x with $s^*_\theta(x) \approx 0$ (near boundary). In order to only affect x such that $s^*_\theta(x) \approx 0$, we apply the sigmoid transform. To test the efficacy of this term, we compare the volume of safe sets learned with and without it, at iterations where they attain similar saturation risk. It turns out that without this term, the safe set tends to shrink more than necessary during training, as smaller safe sets have a lower possibility of saturating. We find that inclusion of the term provides a nearly 200% increase in volume.

4 Experiments

In this section, we train a neural CBF on a challenging input-limited robotic system. We pose two experimental questions:

Q1. How well do we achieve our training objective?

Q2. Does the CBF-based safe controller ensure FI?



(a) (b) θ v. $\dot{\theta}$ (pend pitch, pitch vel) (c) γ v. $\dot{\gamma}$ (quad roll, roll vel) (d) $\dot{\theta}$ v. $\dot{\beta}$ (pend, quad pitch vel)

Figure 2: (Left) quadcopter-pendulum system (image from [24]). (Others) Various axis-aligned 2D slices of the 10D safe set (learned in blue, baseline in purple). Note that for each slice, the unvisualized states have been set to 0.

Our system is a pendulum on top of a quadcopter (Fig. 2). This is a coupled system, with the dynamics of both components found by the Euler-Lagrangian method [25]. The states are quadcopter position and roll-pitch-yaw orientation (x, y, z and γ, α, β) and roll-pitch pendulum orientation (ϕ_p, θ_p), as well as the first derivatives of these states. The inputs are thrust and torque ($F, \tau_\gamma, \tau_\beta, \tau_\alpha$), which are limited to a bounded convex polytope set. See the Appendix for the system dynamics. The safety specification is to prevent the pendulum from tipping and the quadcopter from rolling:

$$\rho = \gamma^2 + \beta^2 + \delta_p^2 - (\pi/4)^2 \tag{8}$$

, where $\delta_p = \arccos(\cos(\phi_p)\cos(\theta_p))$ is the pendulum's angle from the vertical. Since the quadcopter position does not impact safety and the position dynamics can be decoupled, we exclude position and consider the resulting 10D state, 4D input system. Finally, we note that we have a stabilization-type safety problem (the goal is to limit deviation from the equilibrium, $x_e = \vec{0}$), so we use Design 1 for our CBF.

Next, we propose metrics to answer each of our experimental questions:

M1. $\%$ *of non-saturating states on* $\partial\mathcal{S}^*$: we address Q1 by measuring how well we satisfy Eqn. 4. We compute this by uniformly sampling 10k states on $\partial\mathcal{S}^*$ and calculating what percentage of them are non-saturating. We also use these samples to compute the mean and variance of the severity of saturation, $\mathbb{E}[\mathcal{L}(\theta, x)]$ and $\sigma[\mathcal{L}(\theta, x)]$. To compute the severity of the worst saturation, $\mathcal{L}(\theta, x^*)$, we apply our critic and allow it to use more samples and computation time than during CBF training.

M2. $\%$ *of simulated rollouts that are FI*: Q2 considers the in-the-loop control performance of our learned CBF. For this metric, we initialize 5k rollouts uniformly randomly inside $\mathcal{S}^*$, simulate their trajectories under the safe controller (CBF-QP) until just after they reach the boundary. Then, we record whether the system exited or remained inside $\mathcal{S}^*$ after arriving at the boundary. The value of this metric depends on the choice of nominal controller, k_{nom}. We try two different ones: $k_{nom,unact}(x) = 0$, which yields an unactuated system, and $k_{nom,LQR}$, which is a linear-quadratic regulator (LQR) stabilizing controller.

We choose a non-neural CBF baseline. For nonlinear functions, the usual approach to synthesis is to hand-select a function form and then optimize the parameters. We select a simple but often used form [6, 11]:

$$\rho^*_{baseline} = (\gamma^2 + \beta^2 + \delta_p^2)^{a_1} - (\pi/4)^{2 \cdot a_1} + a_2 \tag{9}$$

, where $a_1 > 0, a_2 \geq 0$ are parameters that are optimized using our formulated loss (Saturation risk). See Appendix for details of this optimization.

Training details. The learning framework and metrics were implemented using Python and PyTorch [26]. Training took around 11 hours on a single NVIDIA GeForce RTX 2080 Ti GPU.

	Q1			Q2: % FI rollouts	
method	% non-sat. states	mean, std dev of sat.	worst sat.	$k_{nom,unact}$	$k_{nom,LQR}$
Baseline	78.68	4.99 ± 3.78	29.50	79.53	49.50
Ours	99.00	1.75 ± 2.40	15.19	98.90	97.00

Table 1: Metrics computed on our CBF and baseline CBF. The "mean, std dev of sat." is $\mathbb{E}[\mathcal{L}(\theta, x)] \pm \sigma[\mathcal{L}(\theta, x)]$ and "worst sat." is $\mathcal{L}(\theta, x^*)$.

Discussion. As we can see in Table 1, for our learned CBF, almost 100% of the boundary states are non-saturating and almost 100% of the rollouts are FI across the nominal policies. This shows that our neural CBF and learning framework are an effective combination and capable of handling systems of high complexity. We are not able to attain 100% non-saturating states and FI rollouts, which is either due to suboptimality of training or limitations of the function class, as NNs are only universal approximators when their size is taken to the infinite. The optimized non-neural CBF does not do as well: only 80% of the boundary states are non-saturating and around $50 - 80\%$ of rollouts are FI. For our learned CBF, the severity of saturation at the saturating states is not negligible, but is it still low, and the worst saturation is large, but rarely encountered.

Visualizing slices of the safe set can provide deeper insight into the results of training (Fig. 2). Recall that a safe set contains only states that are *recoverable* from danger, given our input limits. We observe that our CBF has diamond-shaped safe sets in slices B and C, which makes sense because small angles can still be recoverable at larger speeds. Additionally, there should be a maximum safe angular speed for quadcopter and pendulum; our CBF learns this, but the baseline does not. Next, our CBF has a roughly rectangular set in slice D, indicating that $\dot{\theta}, \dot{\beta}$ (pendulum, quadcopter pitch velocity) should both be limited to an interval. This makes sense, since higher angular velocities are certainly harder to pull back from. We also observe a larger range for the quadcopter's pitch than the pendulum's, which makes sense because the quadcopter is directly actuated and the pendulum is only indirectly actuated. On the other hand, the baseline does not restrict $\dot{\theta}, \dot{\beta}$ in slice D. Due to the functional form of the baseline CBF, $\dot{\theta}, \dot{\beta}$ are only restricted when $\dot{\theta}\theta > 0$ or $\dot{\beta}\beta > 0$ (that is, when angular velocity is acting to destabilize). However, this safety criterion is too lenient, since large angular velocities that are currently swinging the system to the origin can cause overshooting and toppling shortly after. Overall, we can see that the non-neural CBF does not have the right function form. In general, it can be hard to guess what the right form might be for a system of this complexity. However, our algorithmic approach, combined with the NN functional representation, removes the need for guessing.

Limitations and future work One limitation of our results is that we had to do some minor feature engineering on the inputs to the neural CBF. Namely, we changed the pendulum's angular velocity to a linear velocity and the quadcopter's angular velocity to the linear velocity of its vertical body axis. (Note that the dynamics of these states is still nonlinear). Before this change, learning was difficult. This might be because a valid CBF is a *highly* nonlinear function (possibly involving some trigonometric terms) of the original states, and such functions are challenging for NNs to learn. This problem might be able to be solved otherwise with a different network architecture. However, we note that after we made this adjustment, the rest of the synthesis required no human intervention. Additionally, in this work we have assumed known, deterministic systems, but real-world systems are often unknown and stochastic. An interesting future direction is extending our work with more realistic modeling assumptions.

5 Conclusion

In summary, we proposed a framework for facilitating CBF synthesis under input limits. Thanks to our neural CBF representation and our effective and efficient learning framework, our method scales to higher-dimensional nonlinear systems. We learned a virtually non-saturating CBF on one such system, the quadcopter-pendulum. We hope that this safe control tool will makes CBFs more accessible and of practical value to roboticists.

References

[1] B. Xu and K. Sreenath. Safe teleoperation of dynamic uavs through control barrier functions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7848–7855. IEEE, 2018.

[2] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames. Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8129–8136. IEEE, 2020.

[3] R. Liu, R. Chen, Y. Sun, Y. Zhao, and C. Liu. Jerk-bounded position controller with real-time task modification for interactive industrial robots. 2022.

[4] A. D. Ames, E. A. Cousineau, and M. J. Powell. Dynamically stable bipedal robotic walking with nao via human-inspired hybrid zero dynamics. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 135–144, 2012.

[5] C. Liu and M. Tomizuka. Control in a safe set: Addressing safety in human-robot interactions. In *Dynamic Systems and Control Conference*, volume 46209, page V003T42A003. American Society of Mechanical Engineers, 2014.

[6] W. Zhao, T. He, and C. Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

[7] 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*, pages 6271–6278. IEEE, 2014.

[8] Y. Lyu, W. Luo, and J. M. Dolan. Probabilistic safety-assured adaptive merging control for autonomous vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10764–10770. IEEE, 2021.

[9] W. S. Cortez and D. V. Dimarogonas. Safe-by-design control for euler-lagrange systems. *arXiv preprint arXiv:2009.03767*, 2020.

[10] W. S. Cortez, X. Tan, and D. V. Dimarogonas. A robust, multiple control barrier function framework for input constrained systems. *IEEE Control Systems Letters*, 6:1742–1747, 2021.

[11] T. Wei and C. Liu. Safe control with neural network dynamic models. *arXiv preprint arXiv:2110.01110*, 2021.

[12] A. Clark. Verification and synthesis of control barrier functions. *arXiv preprint arXiv:2104.14001*, 2021.

[13] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[14] S. Bansal and C. J. Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2021.

[15] M. Chen, S. Herbert, and C. J. Tomlin. Fast reachable set approximations via state decoupling disturbances. *arXiv preprint arXiv:1603.05205*, 2016.

[16] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018.

[17] S. M. Richards, F. Berkenkamp, and A. Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Conference on Robot Learning*, pages 466–476. PMLR, 2018.

[18] H. Ravanbakhsh and S. Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 43(2):275–307, 2019.

[19] Y.-C. Chang, N. Roohi, and S. Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

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

[21] 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)*, pages 3717–3724. IEEE, 2020.

[22] N. M. Boffi, S. Tu, N. Matni, J.-J. E. Slotine, and V. Sindhwani. Learning stability certificates from data. *arXiv preprint arXiv:2008.05952*, 2020.

[23] H. Narayanan and P. Niyogi. Sampling hypersurfaces through diffusion. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 535–548. Springer, 2008.

[24] R. Figueroa, A. Faust, P. Cruz, L. Tapia, and R. Fierro. Reinforcement learning for balancing a flying inverted pendulum. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 1787–1793. IEEE, 2014.

[25] M. Hehn and R. D'Andrea. A flying inverted pendulum. In *2011 IEEE International Conference on Robotics and Automation*, pages 763–770. IEEE, 2011.

[26] 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*, 32, 2019.