Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control

Charles Dawson, Sicun Gao, and Chuchu Fan

***Abstract*—Learning-enabled control systems have demonstrated impressive empirical performance on challenging control problems in robotics, but this performance comes at the cost of reduced transparency and lack of guarantees on the safety or stability of the learned controllers. In recent years, new techniques have emerged to provide these guarantees by learning certificates alongside control policies—these certificates provide concise data-driven proofs that guarantee the safety and stability of the learned control system. These methods not only allow the user to verify the safety of a learned controller but also provide supervision during training, allowing safety and stability requirements to influence the training process itself. In this article, we provide a comprehensive survey of this rapidly developing field of certificate learning. We hope that this article will serve as an accessible introduction to the theory and practice of certificate learning, both to those who wish to apply these tools to practical robotics problems and to those who wish to dive more deeply into the theory of learning for control.**

***Index Terms*—Deep learning in robotics and automation, formal methods in robotics and automation, neural certificates, robot safety.**

I. Introduction

MANY desirable properties of dynamical systems, including stability, safety, and robustness to disturbance, can be proven via the use of *certificate functions*. Perhaps the most well known of these certificate functions are Lyapunov functions: pseudo-energy functions that, if shown to be strictly decreasing along trajectories of the system, prove the stability of the system about a fixed point [1]. Other well-known certificates include barrier functions (which prove forward invariance of a set, and thus safety [2], [3]) and contraction metrics (which prove differential stability in trajectory tracking [4], [5]). A summary of these different certificates is shown in Fig. 1.

These certificate functions can be extremely valuable to the control system designer, as they allow her to prove safety and stability properties even for complex and nonlinear control systems. However, although the theory of certificates like Lyapunov functions is more than a century old, it is not until the last decade that general numerical methods have emerged to construct certificates, and even then many proposed methods were computationally intractable (e.g., relying on solving a high-dimensional partial differential equation (PDE) numerically) [1]. Without efficient general-purpose methods, finding certificates requires spending great effort to hand-design certificates for specific systems. Even in the best case, this hand-tuning requires a good deal of intuition and luck to find an appropriate functional form (e.g., fixed-degree polynomial) and parameters (e.g., polynomial coefficients) for the certificate.

In recent years, new techniques have emerged for automatically synthesizing certificate functions. For systems with polynomial dynamics, the search for a valid certificate can be framed as a convex semidefinite optimization problem through the use of sum-of-squares (SoS) techniques [6]. Unfortunately, SoS methods are limited to polynomial systems and scale poorly to higher dimensional systems [7]. To address these shortcomings, an emerging body of work in the control theory, machine learning, and robotics communities has employed neural networks to learn approximate certificate functions [8].

Unlike traditional approaches to learning for control that search only for a control policy (such as many reinforcement learning (RL) methods), certificate-based techniques simultaneously search for a control policy and a certificate that proves the soundness of that policy. This search may be guided by a separate reward function, similarly to traditional RL methods [9], but the reward function can be omitted entirely in favor of a self-supervised training signal provided by the certificates themselves [8], [10], [11], [12]. Because they produce a verifiable correctness certificate alongside a learned control policy, these *neural certificates* provide a trustable approach to learning for control, helping to address concerns about the safety and reliability of learned controllers. Neural certificates have been successfully applied to complex nonlinear control tasks, including stable walking under parametric uncertainty [13], precision quadrotor flight through turbulence induced by propeller

Manuscript received 7 October 2022; accepted 6 December 2022. Date of publication 11 January 2023; date of current version 7 June 2023. The work of Charles Dawson was supported by the National Science Foundation Graduate Research Fellowships Program under Grant 2141064. This work was supported by the Defense Science and Technology Agency (DSTA) in Singapore. This paper was recommended for publication by Associate Editor J. Kober and Editor P. Robuffo Giordano upon evaluation of the reviewers' comments. *(Corresponding author: Charles Dawson.)*

Charles Dawson and Chuchu Fan are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: cbd@mit.edu; chuchufan1990@gmail.com).

Sicun Gao is with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093 USA (e-mail: sicung@ucsd.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3232542.

Digital Object Identifier 10.1109/TRO.2022.3232542

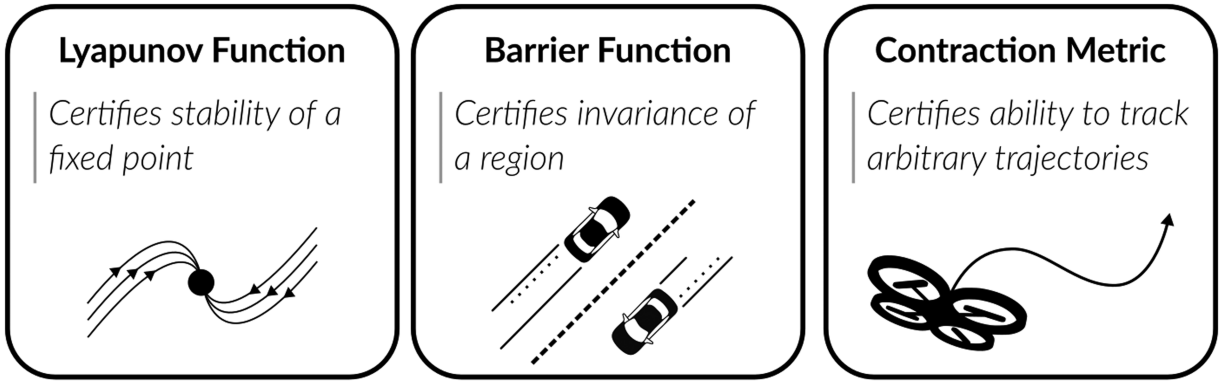

Fig. 1. Lyapunov functions, barrier functions, and contraction metrics are three common types of control certificate. Each can be used to certify different properties.

wash [14], and provably safe decentralized control of multiagent systems containing over 1000 agents [11].

A. Scope and Contributions

This is a rapidly developing field for which no comprehensive survey has yet been written. Giesl and Hafstein's 2015 review [1] covers traditional methods for Lyapunov function synthesis, focusing on techniques for finding Lyapunov functions as the solutions of PDEs or through linear and second-order cone programming, but does not cover neural representations. Ahmadi and Majumdar's 2016 paper [6] presents a relevant summary of the SoS-optimization-based approaches for synthesizing Lyapunov functions, but similarly does not cover neural representations (and only applies to systems with polynomial dynamics). The 2021 review of contraction theory by Tsukamoto et al. [5] discusses neural approaches to learning one particular type of control certificate, but does not discuss the most common certificates (Lyapunov and barrier functions). Our goal is to provide a comprehensive survey of recent developments in certificate learning to serve as an accessible introduction to these tools, both to those who might wish to apply them to practical robotics problems and to those who wish to dive more deeply into the theory of learning for control. Our survey includes theoretical discussion of all three common types of control certificate: Lyapunov functions, barrier functions, and contraction metrics. However, reflecting the prevalence of Lyapunov and barrier functions in the literature, our case studies focus mainly on these two types of certificate (readers interested in case studies involving contraction metrics are referred to the tutorial paper by Tsukamoto et al. [5]). Our review is organized as follows.

1) Section II provides the relevant background from control theory, introducing Lyapunov functions, barrier functions, and contraction metric certificates.
2) Section III discusses prior nonneural approaches to certificate synthesis.
3) Section IV discusses how neural networks can be used to synthesize certificates and their corresponding safe controllers. This section attempts to unify the various synthesis methods presented in the literature, but also includes some historical discussion.
4) Section V discusses issues that arise when implementing certificate-based controllers on hardware, and Section VI presents a number of case studies applying these techniques to practical robotic systems, both in simulation and hardware.
5) Finally, Sections VII and VIII conclude by discussing some open problems in this area.

II. Background

This section provides the necessary background from control theory by outlining the various types of certificate function and their use in controller design. Readers with a background in control theory should feel free to skim this section. We begin by defining notation and terminology and then examine the three major types of control certificates: Lyapunov functions, barrier functions, and contraction metrics.

A. Dynamical Systems

In the rest of this article, we will consider general dynamical systems of the form $\dot{x} = f(x, u)$, where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state, $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the input, and $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$ is the flow map (which we assume to be locally Lipschitz in x and u). The sets $\mathcal{X}$ and $\mathcal{U}$ represent the sets of admissible state and control inputs, respectively. Often, we will consider a restricted (but still quite general) class of dynamics known as *control affine:* $\dot{x} = f(x) + g(x)u$, where $g : \mathbb{R}^n \mapsto \mathbb{R}^{n\times m}$ is also assumed to be locally Lipschitz. For clarity, we will focus only on the continuous time case, but the theory for the discrete-time case follows similarly; readers interested in the discrete-time theory may read [5] or [15].

In the context of these dynamics, the control engineer's task is to find a feedback controller $\pi : \mathcal{X} \mapsto \mathcal{U}$ such that the control input $u = \pi(x)$ imparts the *closed-loop system* $\dot{x} = f_c(x) = f(x, \pi(x))$ with certain desirable properties (e.g., stability). In general, π may be a function of time as well as state, as in trajectory-tracking control. For clarity, we focus on the case when the state x is fully observable, but Section V discusses

methods for ensuring robustness to imperfect state measurements or observation-feedback control.

Given an initial state $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$ at time $t = 0$, the closed-loop system's behavior can be described via its trajectory map $x(t) = \xi_\pi(x_0, t) : \mathcal{X}_0 \times \mathbb{R}^+ \mapsto \mathcal{X}$, which we associate with the controller π used to close the feedback loop. When designing a controller, it is important to ensure that the system's behavior achieves objectives such as stability, safety, or contraction. In the rest of this section, we will define each of these objectives in turn and introduce certificates that can be used to prove that a controller achieves these objectives.

B. Stability and Lyapunov Certificates

Stability can be defined in a number of increasingly strict senses [16]. A point $x_g \in \mathcal{X}$ is said to be:

1) stable in the sense of Lyapunov (i.s.L.) if, for an appropriate norm, for every $\epsilon > 0$, there exists a $\delta > 0$ such that for all $t_2 \geq t_1 \geq 0$

$$||x(t_1) - x_g|| \leq \delta \Rightarrow ||\xi_\pi(x_0, t_2) - x_g|| \leq \epsilon \quad (1)$$

2) asymptotically stable if the system is stable i.s.L. and

$$\lim_{t\to\infty} ||\xi_\pi(x_0, t) - x_g|| = 0 \quad \forall x_0 \in \mathcal{X}_0 \quad (2)$$

3) exponentially stable if there exist constants $C, \lambda > 0$ such that

$$||\xi_\pi(x_0, t) - x_g|| \leq C||x_0 - x_g||e^{-\lambda t} \quad \forall\, x_0 \in \mathcal{X}_0. \quad (3)$$

The concept of stability can also be extended to the case where the goal is a set $\mathcal{X}_g \subseteq \mathcal{X}$ rather than a point, in which case the norm distance between points in the above hierarchy is replaced with the distance between a point and $\mathcal{X}_g$ (see [17, Def. 4.10]).

To prove that a closed-loop system is stable about a point, we can turn to one of the most widely known types of certificate: the Lyapunov function. A continuously differentiable function $V : \mathcal{X} \mapsto \mathbb{R}$ is a Lyapunov function if

$$V(x_g) = 0 \quad (4a)$$

$$V(x) > 0 \quad \forall x \in \mathcal{X} \setminus \{x_g\} \quad (4b)$$

$$\frac{dV}{dt} \leq 0 \quad \forall x \in \mathcal{X} \quad (4c)$$

where $\frac{dV}{dt} = \nabla V(x) f_c(x)$ is the Lie derivative of V along the closed-loop dynamics f_c (often denoted $L_{f_c} V(x)$). If a function satisfying these conditions can be found, then it serves to certify the stability of x_g via the following theorems.

Theorem 1 (see [16, Th. 4.1]): If V is a Lyapunov function (i.e., $V(x_g) = 0, V(x) > 0 \,\forall\, x \in \mathcal{X} \setminus \{x_g\}$, and $\frac{dV}{dt} \leq 0 \,\forall\, x \in \mathcal{X}$) and $f_c(x_g) = 0$, then the system has a stable i.s.L. equilibrium at x_g [with $\mathcal{X}$ forming the region of attraction (RoA)]. Moreover, if $\frac{dV}{dt} < 0$ for all $x \in \mathcal{X} \setminus \{x_g\}$, then the system has an asymptotically stable equilibrium at x_g.

The proof can be found in most control textbooks and is omitted here, but the primary insights are: 1) sublevel sets of V are forward invariant (i.e., once the system enters a sublevel set of V, it will remain inside that set for all future time), proving stability i.s.L.; and 2) if V is monotonically decreasing and bounded below, then it must eventually approach its minimum value at 0. Intuitively, it is useful to think of V as a generalized energy—if the system is strictly dissipative, then it will eventually come to rest. We can also certify exponential stability by adding additional conditions to V.

Theorem 2 (see [16, Th. 4.10]): If V is a Lyapunov function, $f(x_g) = 0$, and there exist positive constants k_1, k_2, k_3, and a such that

$$k_1||x - x_g||^a \leq V \leq k_2||x - x_g||^a \quad (5a)$$

$$\frac{dV}{dt} \leq -k_3||x - x_g||^a \quad (5b)$$

then x_g is exponentially stable.

The proof can be found in [16]; the main insight is that condition (5b) ensures that V decays exponentially to zero, while condition (5a) ensures that $V \to 0$ implies $||x - x_g|| \to 0$. Other forms of Theorem 2 exist in the literature [10], typically replacing (5b) with $L_{f_c} V \leq -k_3 V$, but these modifications do not change the essential theory.

In addition to certifying the stability of a closed-loop system, Lyapunov functions can also be applied to certify the *stabilizability* of open-loop systems (i.e., prove that there exists some controller that makes the closed-loop system stable). Restricting our view to a control-affine system, a control Lyapunov function (CLF) $V : \mathcal{X} \mapsto \mathbb{R}$ certifies asymptotic stabilizability about x_g if

$$V(x_g) = 0 \quad (6a)$$

$$V(x) > 0 \quad \forall x \in \mathcal{X} \setminus \{x_g\} \quad (6b)$$

$$\inf_{u\in\mathcal{U}} \left[L_f V(x) + L_g V(x) u\right] \leq 0 \quad \forall x \in \mathcal{X} \quad (6c)$$

where $L_f V$ and $L_g V$ denote the Lie derivatives of V along f and g, respectively. A similar definition exists for exponentially stabilizing CLFs (ESCLF) [18], where the last condition is replaced with

$$\inf_{u\in\mathcal{U}} \left[L_f V(x) + L_g V(x) u + cV(x)\right] \leq 0 \quad \forall x \in \mathcal{X} \quad (7)$$

for some positive c. Once found, a CLF (or ESCLF) defines a set of admissible control inputs for each state

$$K_{\text{CLF}} = \{u : L_f V(x) + L_g V(x) u \leq 0\} \quad (8)$$

$$K_{\text{ESCLF}} = \{u : L_f V(x) + L_g V(x) u + cV(x) \leq 0\}. \quad (9)$$

Any Lipschitz policy π that chooses control inputs from these sets will necessarily stabilize the system. Since these conditions are affine in u, a common choice is a quadratic program (QP) that finds the smallest magnitude control such that $u \in K_{\text{CLF}}$ or $u \in K_{\text{ESCLF}}$ [18], for example

$$\min_{u\in\mathcal{U}} \; ||u||^2 \quad (10a)$$

$$\text{s.t. } L_f V(x) + L_g V(x) u \leq -cV(x). \quad (10b)$$

This QP can also be used to filter a potentially unstable reference policy π_r by replacing the objective with $||u - \pi_r||^2$, in which case the QP solves for the value of u closest to the reference that still ensures stability, as shown in Fig. 2.

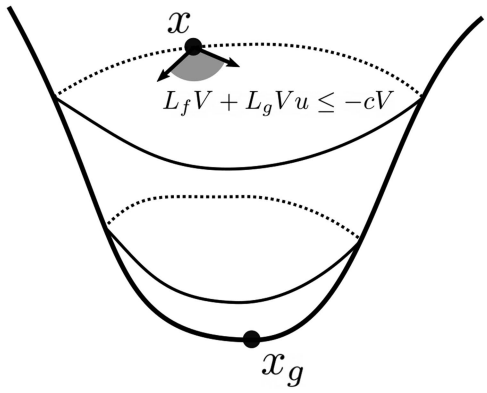

Fig. 2. CLF can be used to filter a potentially unstable control input by solving a QP. The contours illustrate how V varies with x, and the gray region indicates the potential values of $\dot{x}$ for different values of $u \in K_{\text{ESCLF}}$.

Now that we have derived a control policy from the solution of a constrained optimization problem, it is important to discuss the continuity of that policy. An important result in the theory of CLFs (also applicable to control barrier functions (CBFs), which we discuss next) is that the controller (10a) is Lipschitz continuous (see [18] and [19, Sec. 4.2]) and has a closed-form solution so long as $\mathcal{U}$ is convex [18].

C. Safety and Barrier Certificates

Given an unsafe set $\mathcal{X}_u \subseteq \mathcal{X}$ that does not intersect the set of initial conditions $\mathcal{X}_0$, a system is safe if it will never enter the unsafe region if started within $\mathcal{X}_0$. That is, it will be safe if all trajectories satisfy

$$x_0 \in \mathcal{X}_0 \Rightarrow \xi_\pi(x_0, t) \notin \mathcal{X}_u \qquad \forall t \geq 0. \tag{11}$$

To prove this property, we would need to prove that some set (containing $\mathcal{X}_0$ but not intersecting $\mathcal{X}_u$) is forward invariant. We can do this using a certificate known as a barrier function.

Concretely, consider a compact set $\mathcal{C}$, defined as the zero sublevel set of a barrier function $h : \mathcal{X} \mapsto \mathbb{R}$ (i.e., $\mathcal{C} = \{x : h(x) \leq 0\}$). If h satisfies certain properties (given in Theorem 3), then $\mathcal{C}$ will be forward invariant. Note that some references reverse the sign of h to be positive on the invariant set; however, we choose this convention to illustrate the parallels between barrier and Lyapunov certificates. In particular, it is often useful to know that every Lyapunov function implies a family of barrier functions defined via its sublevel sets.

Theorem 3 (see [20, Proposition 1]): If there exists a strictly increasing scalar function $\alpha : \mathbb{R} \to \mathbb{R}$ such that $\alpha(0) = 0$ (i.e., an extended class-$\mathcal{K}$ function) and

$$\frac{dh}{dt} \leq -\alpha(h(x)) \quad \forall x \in \mathcal{X} \tag{12}$$

then h is a barrier function and $\mathcal{C}$ is forward invariant for the closed-loop system $\dot{x} = f_c(x)$.

A common choice for $\alpha(h)$ is simply γh for some positive γ. The proof of this theorem can be found in [20]; the main insight is that h is decreasing whenever x is on the boundary of $\mathcal{C}$, where $h(x) = 0$, which prevents the system from ever exiting $\mathcal{C}$. In fact, if the system starts outside of $\mathcal{C}$, this condition ensures that it will asymptotically converge to $\mathcal{C}$. A direct consequence of Theorem 3 is that if $\mathcal{X}_0 \subseteq \mathcal{C}$ and $\mathcal{X}_u \cap \mathcal{C} = \emptyset$, then the closed-loop system is safe in the sense of the definition in (11).

Barrier functions can be extended to certify the safety of open-loop systems through the use of CBFs, which directly parallel the CLFs introduced in the previous section. A function h is a CBF if

$$\inf_{u \in \mathcal{U}} \left[L_f h(x) + L_g h(x) u + \alpha(h(x))\right] \leq 0. \tag{13}$$

As with CLFs, this condition is affine in u, allowing a CBF to be used in a QP in a similar way. Usefully, the CBF condition (13) can be combined with a relaxed version of the CLF condition (6c) to yield a single controller that combines safety and stability considerations [20]

$$\min_{u \in \mathcal{U}} \ ||u||^2 + kr \tag{14a}$$

$$\text{s.t. } L_f V(x) + L_g V(x) u + cV(x) \leq r \tag{14b}$$

$$L_f h(x) + L_g h(x) u + \alpha(h(x)) \leq 0 \tag{14c}$$

$$r \geq 0. \tag{14d}$$

In this combined CLF-CBF QP, the constraint (14c) ensures safety at all times, while the stability constraint (14b) is relaxed by some variable amount r. This relaxation allows the system to temporarily cease progress toward the goal in order to remain safe, and $k > 0$ is a tunable parameter governing the tradeoff between control effort and relaxation of the CLF condition. It is important to note that this combined QP can suffer from deadlock when there is no safe direction that moves closer to the goal, although there are proposals for switched controllers [21] and unified Lyapunov-barrier certificates [10] that alleviate this concern.

D. Contraction Metric Certificates

In our previous discussion of stability, we assumed that the control system designer is interested in stabilizing the system about a fixed point; however, in many applications, we are more interested in trajectory tracking rather than stabilization. Of course, a fixed-point-tracking controller can be used to track a trajectory if that trajectory varies slowly enough, but in order to track more general trajectories, we need a richer specification than simply stabilizing a fixed point.

In this context, the designer wishes to find a controller so that the trajectories $x(t)$ of the closed-loop system will converge to some desired trajectory, which we represent as a sequence of states and control inputs $x^*(t)$ and $u^*(t)$

$$||x(t) - x^*(t)|| \leq Re^{-\lambda t}||x(0) - x^*(0)|| \; \forall \, t \geq 0 \tag{15}$$

for some $R \geq 1$, $\lambda > 0$. Intuitively, we can think of this as a requirement that the distance between the system's trajectory and the reference we wish to track shrinks exponentially (with some potential overshoot allowed by R). Contraction theory [22], [23] allows us to formalize this requirement.

Just as a Lyapunov function defines an energy-like quantity that decays along trajectories (proving stability), contraction

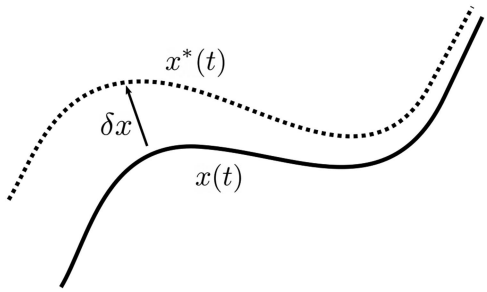

Fig. 3. Contraction metric proves that the system will exponentially converge to track any feasible reference trajectory. The error $||x(t) - x^*(t)||$ shrinks exponentially, with transient overshoot limited by the condition number of M and the steady-state tracking error is bounded proportional to the worst case disturbance.

theory defines a measure of distance between neighboring trajectories (a *contraction metric*) that decreases exponentially over time [22]. Let the metric $M(x) : \mathcal{X} \mapsto \mathbb{R}^{n\times n}$ be a function returning a positive-definite matrix (which we denote $M \succ 0$). This matrix defines a metric (a generalized distance) on virtual displacements between trajectories $\delta x^T M \delta x$. By showing that this metric is contracting (i.e., the displacement between any two neighboring trajectories is shrinking), a contraction metric allows us to certify that a closed-loop system (with some feedback tracking controller) is capable of tracking any feasible trajectory using the following theorem.

Theorem 4 (see [14, Proposition 1]): Let $\dot{M} = \sum_{i=1}^{n} \frac{\partial M}{\partial x_i}\dot{x}_i$, $A = \frac{\partial f}{\partial x}$, $B = \frac{\partial f}{\partial u}$, and $K = \frac{\partial \pi}{\partial x}$ for some feedback tracking controller $\pi(x, x^*, u^*)$. Furthermore, let $\mathrm{sym(A)} = \mathrm{A} + \mathrm{A^T}$. Then, if for all $x \in \mathcal{X}$, $x^* \in \mathcal{X}$, and $u^* \in \mathcal{U}$

$$\dot{M} + \mathrm{sym(M(A + BK))} + 2\lambda \mathrm{M} \prec 0 \tag{16}$$

then (15) holds with R equal to the square root of the condition number of M, and we say that the system is contracting with rate λ.

Contraction is a property of the closed-loop system, not of any particular trajectory, and thus, the existence of a metric satisfying condition (16) suffices to prove that a control system is capable of exponentially stabilizing any dynamically feasible nominal trajectory. In addition, if a system is contracting, then bounded disturbances will produce a bounded worst case tracking error, and this tracking error will be proportional to the magnitude of the disturbance [14]. As a result, the metric M provides a certificate that the corresponding tracking controller π successfully stabilizes the system and is robust to disturbances, as illustrated in Fig. 3.

III. Prior Work on Certificate Synthesis

In this section, we discuss prior work on certificate synthesis methods. We separate this discussion into two parts. First, we cover early methods for certificate synthesis, including numerical methods, polynomial optimization, and simulation-guided synthesis. These methods represent an important step in the history of certificate synthesis, but the poor scalability of these early methods, particularly when dealing with nonlinear dynamics, sparked the development of the neural techniques that are the focus of this survey.

Of course, neural certificates are not the only class of methods that have evolved to fill this gap. The second part of this section discusses two parallel lines of work, safe RL and reachability, that are similar in purpose to neural certificate methods but often use different vocabularies and methods. Our aim here is not to provide a comprehensive introduction to these methods, since excellent surveys exist for both Hamilton–Jacobi (HJ) reachability [24] and safe RL [25]; instead, we aim to highlight key similarities and differences between these lines of research and situate recent developments in neural certificates in this modern context.

A. Early Approaches: Numerical Solutions and Optimization

Depending on the complexity of the dynamics, several existing techniques may be used to synthesize certificate functions, although there is currently no generally applicable scalable framework [1]. When the dynamics of the closed-loop system are linear and stable, $\dot{x} = (A - BK)x$ for feedback gains K, then a quadratic Lyapunov function will exist of the form $V(x) = x^T P x$ for some symmetric positive-definite matrix P. In this case, P can be found by solving the continuous Lyapunov equation numerically (most numerical linear algebra packages provide this functionality, including MATLAB [26] and SciPy [27]).

For systems with polynomial dynamics, constraints on the sign of the certificate function's derivative can be expressed as constraints that certain polynomials be expressible as SoS. Owing to the correspondence between fixed-degree SoS polynomials and positive-semidefinite matrices (which form a convex cone), certificates encoded in the SoS framework can be synthesized using convex optimization methods. Unfortunately, the computational complexity of these techniques grows exponentially in the degree of polynomials involved, and they are limited to systems with polynomial dynamics [6]. In addition, SoS optimization is only convex when searching for either a certificate or a controller; the optimization problem becomes nonconvex and often encounters numerical issues when searching for a certificate and controller simultaneously [28]. A comprehensive review of SoS methods for control is outside the scope of this review, but the interested reader may see [6] for a survey and [28], [29], and [30] for relevant examples.

A related technique for certificate synthesis for nonlinear systems is the simulation-guided synthesis [31]. In this framework, the control engineer selects a fixed set of basis function $z(x)$ (e.g., monomials up to some degree) and constructs a Lyapunov candidate of the form $V(x) = z^T P z$. The state space is then sampled at a large number of points, and each point is simulated forward to estimate the closed-loop state derivative. After computing z for these fixed points, the Lyapunov and barrier function conditions at those points become linear in the entries of P, allowing P to be computed using linear programming (LP). After solving for P, the candidate V is verified using a satisfiability modulo theory (SMT) solver, which generalizes Boolean

TABLE I
QUALITATIVE ASSESSMENT OF EXISTING METHODS FOR CONTROLLER SAFETY CERTIFICATION

	Sum-of-Squares [6]	CEGIS [31]	HJ Reachability [24]	Neural certificates
Scalability	$O(n^{6.5d})$ for d-degree monomials [101]	SMT is NP-complete [85]	$O(\epsilon^{-n})$, grid size ϵ [24]	Requires sampling state space
Robustness to disturbance	✓	✗	✓	✓
Parametric uncertainty	✓	✗	✗	✓
Model	Known polynomial	Known	Known	May be unknown [62], [67]

satisfiability to include statements involving real numbers [32]. If the SMT solver finds a violation of the certificate conditions, then a counterexample representing that violation is added to the optimization problem and P is recomputed. This process continues until either the SMT solver verifies that V is a valid certificate or the LP becomes infeasible (in which case we can conclude nothing beyond the fact that our choice of basis is not suitable).

Owing to the poor scalability of these methods, particularly with regard to nonlinear or high-dimensional dynamics, they have not seen widespread adoption since their introduction (with the notable exception of SoS methods, although substantial concerns remain about the scalability of SoS methods in practice [10]). A qualitative summary of various prior methods is shown in Table I.

B. Parallel Approaches: Safe RL and HJ Reachability

It is important to acknowledge that certificate-based control is not the only line of research that deals with the problem of guaranteeing safety and stability for control systems. In particular, safe RL [25] and HJ reachability [24] have received a large amount of research interest in recent years. While full coverage of these methods is outside the scope of this survey, we will review the basics of each approach here, with an eye toward highlighting the underlying commonalities and differences between these methods and certificate-based control.

In the broadest possible terms, safe RL is a collection of methods that seek to optimize a control policy to achieve good performance on a specified task (i.e., by maximizing some reward) while respecting a number of constraints on the behavior of the controlled system [25]. There are three classes of safe RL methods in particular that highlight connections to certificate-based control. The first class of methods use an *a priori* given certificate to constrain the actions that the RL agent can take [33], [34], [35], possibly with some learning of model uncertainty as the controller explores [33], [34]. These methods highlight the complementary nature of certificate-based control and traditional RL; certificates enforce safety and stability constraints, while the RL agent tries to achieve good performance on nonsafety-critical metrics. Of course, these methods require a certificate to be provided *a priori*, requiring a separate certificate synthesis process, and therefore, they are not the focus of this survey. The second class of safe RL methods propose to learn certificates alongside a control policy using standard RL optimization algorithms (e.g., learning a Lyapunov function in [9], a CLF in [36], or a CLF and CBF together in [37]). We see these methods as examples of the neural certificate approach, with the main difference between these and other neural certificate learning methods being the choice of optimization algorithm (there is also a minor vocabulary difference, as the safe RL community often considers a "safety index" function that acts similarly to a barrier function [38]). The final class of safe RL methods attempts to derive certificate functions directly from the structure of the constrained partially observable Markov decision process underlying the safe control problem [39], [40]; these methods are exciting, as they suggest the future possibility of extending control-theoretic certificates to handle a more general class of constraints (as we discuss in Section VII).

In addition to the body of safe RL literature, there is also a class of methods for safe control synthesis and verification based on HJ reachability analysis. HJ methods, which are sometimes also referred to as Hamilton–Jacobi–Isaacs (HJI) methods to emphasize the connection to game theory, typically frame the safe control problem as a two-player zero-sum game between the controller and an adversary. In this game, the controller seeks to maximize some safety index (making the system as safe as possible), while the adversary seeks to minimize this index. Both a value function (indicating how unsafe the system will be if it starts at a given state) and a control policy that maximizes the safety of the system are found by approximately solving an HJI PDE under the assumption of bounded actions by the adversary [24]. There are fundamental connections between the HJ value function and certificate functions, since the super- and sublevel sets of HJ value functions and Lyapunov functions, respectively, certify the forward invariance of a safe region. The primary difference between traditional HJ and certificates is that the controller found by solving an HJ reachability problem is *optimally safe* in that it always seeks to maximize the safety of the system and can thus be conservative, but recent work in linking HJ reachability with barrier functions [41] has reduced this conservatism. Historically, HJ methods have been limited by the complexity of solving the HJI PDE in high-dimensional state spaces (which usually requires covering the state space with a discrete mesh of sufficient resolution before iteratively solving the PDE [24]). However, there has been an exciting trend toward using neural networks to approximate solutions of HJ reachability problems [42], [43], paralleling the use of neural networks for general certificate synthesis, substantially reducing the amount of computation required for HJ analysis. Based on these developments, we anticipate that HJ reachability methods and neural certificates will continue to converge over the coming years, with each community applying insights from the other to develop more scalable and flexible algorithms.

IV. LEARNING NEURAL CERTIFICATES

All of the certificates discussed above allow the control engineer to prove that her controller design is sound (or, in the case of CLFs and CBFs, derive a sound controller directly from the

certificate). However, these methods share a common drawback: there has historically been no general scalable method for finding these certificates [1]. In this section, we will discuss how we can apply *self-supervised learning* to learn these certificates using neural networks (so-called because the algorithm requires no human labeling to supervise its learning; it can generate its own labels).

First, we will discuss how a certificate can be found independent of any controller. This case is useful when a controller is known and the designer simply wishes to find a certificate for the closed-loop system, or when the certificate itself can be used to derive a controller implicitly (as with a CLF or CBF). Of course, in some applications, we wish to learn an explicit control policy as well; we will discuss this case shortly, but it is instructive to begin with the certificate-only case.

Generally speaking, the search for a certificate can be seen as an optimization over a space of continuously differentiable functions $\mathcal{V}$. If we consider a generic certificate $V : \mathcal{X} \mapsto R^q$ (where $q = 1$ for Lyapunov and barrier certificates and $q = n^2$ for contraction metrics), then the search for a generic certificate can be encoded as an optimization problem [44]

$$\text{find}_{V\in\mathcal{V}} \text{ s.t. } c_i(x, V) \leq 0 \quad \forall x \in X \quad i \in \mathcal{I} \tag{17}$$

where $\mathcal{I}$ is the set of conditions c_i that must hold for this particular certificate (with slight abuse of notation for converting between equality constraints and inequalities). For example, to search for a Lyapunov function, inequalities (4a)–(4c) must hold, creating the optimization problem

$$\text{find}_{V\in\mathcal{V}} \text{ s.t. } V(x_g) = 0 \tag{18a}$$

$$V(x) \geq 0 \quad \forall x \in \mathcal{X} \setminus \{x_g\} \tag{18b}$$

$$\frac{dV}{dt} \leq 0 \quad \forall x \in \mathcal{X}. \tag{18c}$$

Different approaches to certificate synthesis can be organized according to how they approach this optimization program. For example, the SoS and simulation-guided synthesis methods discussed in Section III restrict $\mathcal{V}$ to the span of a chosen set of basis functions (e.g., polynomials of fixed degree) and then solve (17) as a convex (or bilevel convex) program. The convexity imposed by a particular choice of $\mathcal{V}$ (e.g., the set of polynomials up to a certain degree) makes solving (17) more efficient but can be overly restrictive, resulting in (17) becoming infeasible if $\mathcal{V}$ is not rich enough.

The neural certificate framework can be seen as a natural extension of this approach by searching over a much richer function space. To avoid the limitations imposed by the choice of function space, $\mathcal{V}$ is represented as the space of neural networks of particular depth and width (chosen as hyper-parameters by the user). By increasing the size of these networks, this representation can approximate any continuous function [45], [46], alleviating limitations due to the choice of function space. Although this same universal approximation property also applies to polynomials (i.e., a polynomial of large enough degree can also approximate any continuous function on some domain), in practice, the scaling for neural networks is much more favorable (the number of parameters in the input layer of a neural network grows with $O(nd)$ with n input dimensions and d dimensions in the next layer, while the number of monomials in a d-degree polynomial in n dimensions grows with $O(n^d)$).

To adapt (17) to be solved using a neural representation for V, we must make two modifications to account for the fact that neural networks lend themselves best to unconstrained optimization problems (solved via stochastic gradient descent) rather than constrained optimization. First, we relax the constraints using a penalty method. Second, instead of imposing constraints universally (e.g., $\forall x \in \mathcal{X}$), we evaluate them at a large finite set of randomly sampled training points $\{x_1, \ldots, x_N\} \subset \mathcal{X}$. Each of these training points can be automatically labeled with a loss equal to the violation of the certificate conditions at that point, and averaging this loss across the entire training set yields the *empirical certificate loss*

$$\mathcal{L}_V = \sum_{i\in\mathcal{I}} \frac{\alpha_i}{N} \sum_{j=1}^{N} \max(c_i\left(x_j, V\right), 0) \tag{19}$$

with penalty weights α_i that are tuned empirically as hyper-parameters to encourage constraint satisfaction. *Loss* refers to the fact that all constraints are included as scalar penalties to be minimized, and *empirical loss* is distinguished from the *true* loss in that it is evaluated at a finite number of points rather than over the entire state space. This empirical loss can be minimized using stochastic gradient descent with respect to the weights and biases of the neural network used to represent V, but it is important to remember that zero empirical loss does not guarantee that the certificate is valid; it provides only statistical evidence of validity that can potentially be supplemented by one of the neural network verification strategies discussed in Section V-E.

This approach provides the foundation for many certificate-based learning works; early examples include [47], and later works include [3], [48], [49], [50], [51], [52], [53], and [54]. Examples that learn CLF or CBF certificates that imply controllers include [10], [12], [21], [55], [56], [57], [58], [59], and [60]. A related class of works assumes that a certificate is given for a nominal system and use supervised learning to adapt that certificate to the uncertain true model [2], [13], [61], [62].

A. Remarks on Certificate Learning

At this point, it is important to make two remarks. The first remark concerns the constraints that certificates impose on neural network structure. Most of certificate theory for continuous-time dynamical systems assumes that the certificate is continuously differentiable, so it is important that continuous-time certificates are learned using networks with continuously differentiable activation functions such as `tanh`, `softplus`, or the exponential linear unit. In the discrete-time case, Lyapunov and barrier certificates are required only to be continuous [63]; accordingly, works that focus on the discrete-time case have learned certificates using rectified linear unit (ReLU) activation functions, which are continuous but not differentiable at the origin [64].

Second, it is important to note that although we frame the certificate synthesis problem as a feasibility problem in (17), there are several metrics of certificate quality that might be

included as an objective for the certificate search. The first metric, applicable to Lyapunov functions, is the size of the RoA certified by the resulting certificate [8], [47], [59]. Larger regions of attraction can be encouraged either implicitly (by attempting to maximize the percentage of samples in the state space where the certificate is valid [8]) or by explicitly searching for new training examples to expand the RoA [47]. The second metric, applicable to CLFs and CBFs, is the size of the admissible control set $K(x)$, the maximizing of which will yield a less-restrictive controller. This is particularly relevant when actuator limits are present, in which case the size of the intersection of K and $\mathcal{U}$ should be maximized. This second metric is less well explored in the literature. Either objective could be incorporated into the certificate learning pipeline by adding an appropriately normalized term to the loss function (19), but care should be taken to not allow the neural certificate to overfit to this term at the expense of violating the certificate conditions.

B. Learning Certificates and Controllers Together

Learning the certificate alone is useful when a controller is already known and we seek to verify its performance, or in cases when learning a certificate such as a CLF or CBF allows us to derive a controller. When it is necessary to explicitly learn a controller alongside the certificate, we can easily extend this framework to simultaneously search for a control policy π from some family of policies Π (commonly, π is parameterized as a separate neural network).

There are three different contexts in which we might wish to learn a control policy. The first case is behavior cloning, where we know an "expert" control policy to use as a basis for the learned controller. For example, we may have a computationally expensive nonlinear model-predictive control (MPC) policy that we wish to compress into a neural network for online computation [12] or we may have a full state feedback controller that we wish to clone for a partial state feedback context (Margolis et al.[65] provide a noncertificate-based example of this context). To accomplish this certified behavior cloning, we simply augment the empirical loss in (19) with a behavior cloning term that penalizes the difference between π and the expert policy (often simply the mean of an appropriate norm)

$$\mathcal{L} = \mathcal{L}_V + \mathcal{L}_{\text{BC}}.$$

Another interesting application of this case is using a barrier certificate to learn more realistic behavior predictions for pedestrians [66]. In this application, the "expert" is not a control policy per se but observations of human actions, and the resulting policy π is not used for control but to generate more realistic predictions of the humans' future actions.

The second case where we might wish to learn a controller alongside a certificate is when we do not have an expert reference controller, but we have a reward that we seek to maximize; this leads to certificate-regularized RL. Chang and Gao [9] provide an example of RL with a Lyapunov function, where $\mathcal{L}_V$ is used alongside the reward to update the control policy. This setting brings additional complications, as RL algorithms typically assume no prior knowledge of the dynamics, and therefore, the Lie derivative of V must be estimated from observed trajectories of the system rather than computed exactly.

The final case is when there are no requirements (beyond the certificate) placed on the learned control policy. In this case, $\mathcal{L}_V$ is used to self-supervise both the certificate V and control policy π. The first example of simultaneous certificate/policy learning appeared in [8] for a Lyapunov certificate; later works learn barrier functions [11] and contraction metrics [14] alongside control policies as well, or extend to the case when dynamics are only partially known [67].

Next, we will make this discussion concrete with examples of learning a Lyapunov function, barrier function, and contraction metric and, then, discuss the history of certificate learning in the controls, robotics, and machine learning literature. Code for each example can be found at https://github.com/MIT-REALM/neural_clbf.

C. Example: Learning a Lyapunov Function

Our first example involves learning a CLF for the classic toy example: the inverted pendulum (we will progress to more complex examples shortly). This dynamical system has states $x = [\theta, \dot{\theta}]$ and control inputs $u = [\tau]$. The system is parameterized by its mass m, length L, damping b, and gravitational acceleration g, and its dynamics have control-affine form

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \frac{g}{L}\sin\theta - \frac{b}{mL^2}\dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{mL^2} \end{bmatrix} u.$$

Our goal in learning a CLF is to stabilize this system about $x_0 = [0, 0]$. To learn the CLF, we parameterize V as a neural network with two hidden layers of 64 units and minimize the following empirical loss over $N = 10^4$ training points. In all the experiments, we sample the training data uniformly from the state space, which we represent as a hyper-rectangle

$$\begin{aligned}\mathcal{L}_V = {} & \lambda_1 V(x_0) + \frac{\lambda_2}{N}\sum_{i=1}^{N} r(x_i) \\ & + \frac{\lambda_3}{N}\sum_{i=1}^{N} \max\left(L_f V(x_i) + L_g V(x_i) u_i, 0\right) \\ & + \frac{\lambda_4}{N}\sum_{i=1}^{N} \max\left(\frac{V(x_i + \Delta t \dot{x}(x_i, u_i)) - V(x_i)}{\Delta t}, 0\right)\end{aligned}$$

where $\lambda_1 = 10$, $\lambda_2 = 10^3$, $\lambda_3 = \lambda_4 = 1$, and $[r(x_i), u_i]$ are the solutions to the relaxed CLF QP at state x_i with $c = 1$ and $\lambda_5 = 10^2$

$$\begin{aligned}\min_{r,u} \ & ||u||^2 + \lambda_5 r \\ \text{s.t. } & L_f V(x) + L_g V(x) u \leq -cV(x) + r \\ & r \geq 0.\end{aligned}$$

The third and fourth terms of the empirical loss are different approximations of the Lyapunov decrease conditions; strictly speaking, only the first two terms are needed, but additional terms can help the learning process. Note that in this example, we

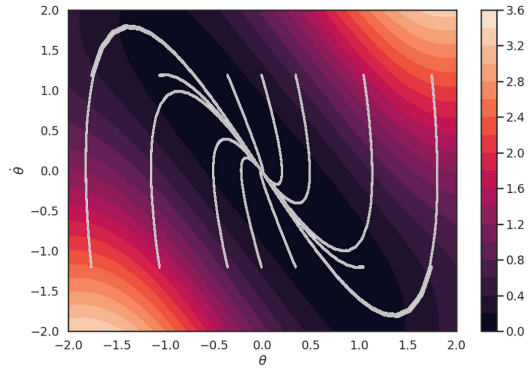

Fig. 4. CLF learned for the inverted pendulum, overlaid with trajectories of the system controlled using the CLF-QP (10a). Lighter colors indicate higher values of V; as expected, V decreases to zero at the goal point, and trajectories of the controlled system always move in a direction of decreasing V.

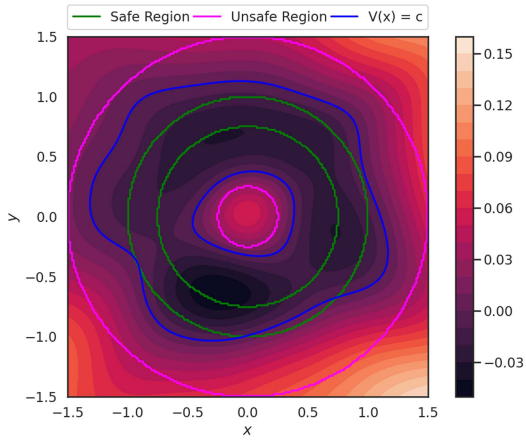

Fig. 5. 2-D cross section of the CBF learned for the satellite station-keeping task, showing how the learned barrier function successfully separates the safe and unsafe sets.

train V to be positive definite, but it is also possible to ensure positive semidefiniteness by construction by learning some function $\omega(x) : \mathbb{R} \mapsto \mathbb{R}^{n_h}$ as a neural network with n_h output units, then taking $V(x) = \omega(x)^T \omega(x)$ (this construction is helpful for more complicated examples, but the first loss term is still required to encourage a minimum at x_0). Code for this example can be found in `training/train_inverted_pendulum.py`. The results of learning a CLF for this example are shown in Fig. 4; the CLF-derived controller successfully stabilizes this simple dynamical system. In the next section, we will set our sights higher with a more complicated example of learning a CBF.

D. Example: Learning a CBF

To demonstrate learning a CBF, we use the spacecraft rendezvous problem from [68]. In this problem, we need to design a station-keeping controller to keep the "chaser" satellite near a "target," respecting both a minimum distance constraint (to avoid collision) and a maximum distance constraint (to enable observation of the target). The state of the chaser is expressed relative to the target using linearized Clohessy–Wiltshire–Hill equations, with state $x = [p_x, p_y, p_z, v_x, v_y, v_z]$ and control inputs corresponding to thrust in each direction $u = [u_x, u_y, u_z]$. With mass m and mean-motion orbital parameter n, the dynamics are given by

$$\dot{x} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & 0-n^2 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} u.$$

The station-keeping constraints define the unsafe states $\mathcal{X}_u = \{x : 0.25 \geq r \leq 1.5\}$ with $r = \sqrt{p_x^2 + p_y^2 + p_z^2}$. To learn a CBF for these constraints, we take a similar approach as in the previous example and define h as a neural network with two hidden layers of 256 units each, sample $N = 10^5$ points from the state space (labeled according to whether or not they belong to the unsafe or safe sets), and minimize the empirical loss

$$\mathcal{L}_V = \frac{\lambda_1}{N_{\text{safe}}} \sum_{i=1}^{N_{\text{safe}}} \max\left(h(x_i), 0\right) + \frac{\lambda_2}{N_{\text{unsafe}}} \sum_{i=1}^{N_{\text{unsafe}}} \max\left(-h(x_i), 0\right) + \frac{\lambda_3}{N} \sum_{i=1}^{N} r(x_i)$$

where $\lambda_1 = \lambda_2 = 100$, $\lambda_3 = 1$, and $r(x_i)$ is found by solving the CBF QP at state x_i with nominal control input u_0 [found using linear–quadratic regulator (LQR)], $c = 0.1$, and $\lambda_5 = 10^4$

$$\min_{r,u} \ ||u - u_0||^2 + \lambda_4 r$$
$$\text{s.t. } L_f h(x) + L_g h(x) u \leq -c h(x) + r$$
$$r \geq 0.$$

Here, the first and second terms of the empirical loss enforce the boundary conditions, training the h network so that its zero-level set segments the safe and unsafe states. The third term is designed to train h so that the corresponding CBF QP is feasible. Code for this example can be found in `training/train_linear_satellite.py`. The learned barrier function is shown in Fig. 5, and an example trajectory for the chaser controlled using the CBF QP is shown in Fig. 6.

E. Example: Learning a Contraction Metric

Our third example demonstrates learning a contraction metric alongside a control policy. For this example, we consider the problem of designing a nonlinear trajectory-tracking control policy to run on an autonomous ground vehicle. To motivate this problem, we look at the case where the ground vehicle has limited computational resources onboard, so running computationally expensive control policies, such as nonlinear MPC, may be infeasible. Instead of running this expensive policy onboard,

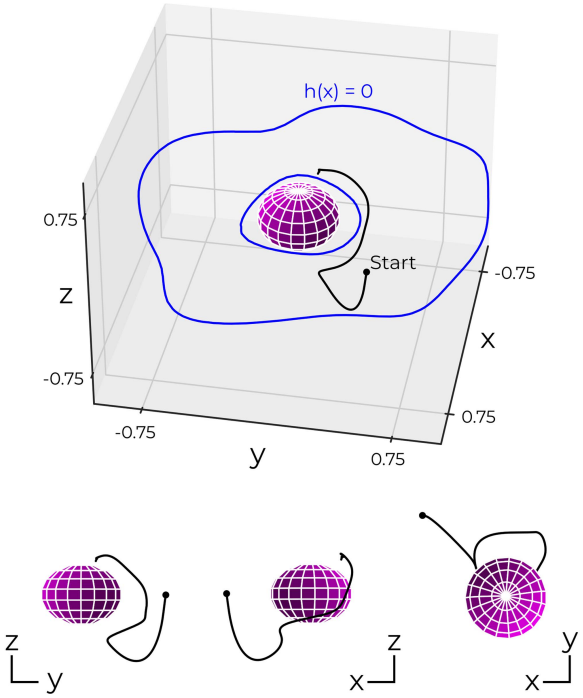

Fig. 6. Trajectory followed by chaser satellite subject to station-keeping CBF. The chaser starts by dropping below the target, then circles up to a fixed point, respecting the station-keeping constraints at all times. The blue lines show a 2-D cross section of the zero-level set of the CBF h in the xy plane.

we can use a neural network to learn a computationally inexpensive clone of the MPC policy, and we also learn a contraction metric to ensure that the cloned policy is sound.

Concretely, we consider a simple nonlinear ground vehicle model with states $x = [p_x, p_y, \theta]$ for the 2-D position and heading and control inputs $u = [v, \omega]$ for the linear and angular velocities. The system dynamics are control affine

$$\dot{x} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u.$$

We construct an expert control policy by solving a nonlinear MPC problem using CasADi [69]. Using this expert policy, we construct a training dataset of 100 random reference trajectories of length 10 s each (simulated at 100 Hz with a 10-Hz zero-order hold control signal; data points were sampled at 10 Hz). Expert demonstrations are augmented with a small amount of noise to widen the distribution of training data. We represent the contraction metric $M(x)$ and control policy $u(x, x^*, u^*)$ as neural networks with two hidden layers of 32 neurons and construct the metric $M = A + A^T$, where A is the output of the neural network to ensure that the contraction metric is symmetric (other parameterizations exist that ensure that the eigenvalues of M are lower-bounded by construction [14], but we use this construction for simplicity). We also set $u(x, x^*, u^*) = u^* + \pi(x, x - x^*) - \pi(x, 0)$ to ensure that $u(x, x^*, u^*) = u^*$ when $x = x^*$. We simultaneously optimize the parameters of the metric and policy networks to minimize the empirical loss

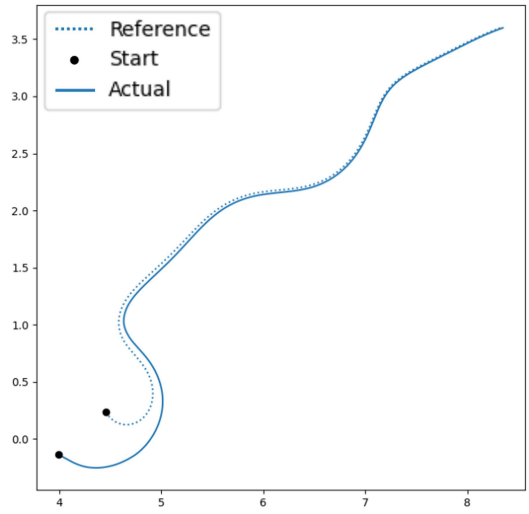

Fig. 7. Cloned policy (learned alongside a contraction metric) successfully tracks a previously unseen reference trajectory. The reference trajectory was generated using a random combination of sinusoidal control inputs.

over $N_{\text{train}} = 10^4$ tuples $(x, x^*, u^*, u_{\text{expert}})$ sampled uniformly from a user-specified hyper-rectangle

$$\mathcal{L} = \mathcal{L}_M + \mathcal{L}_u$$

$$\mathcal{L}_M = \frac{1}{N_{\text{train}}} \sum_i [M - \underline{m} I]_{\text{PD}} + \frac{1}{N_{\text{train}}} \sum_i [\bar{m} I - M]_{\text{PD}} + \frac{1}{N_{\text{train}}} \sum_i [\dot{M} + \text{sym}(M(A + BK)) + 2\lambda M]_{\text{ND}}$$

$$\mathcal{L}_u = \frac{1}{N_{\text{train}}} \sum_i ||u(x_i, x_i^*, u_i^*) - u_{\text{expert},i}||^2$$

where $[\circ]_{\text{PD}}$ and $[\circ]_{\text{ND}}$ are hinge losses encouraging positive and negative semidefiniteness, respectively. Example code for this example is provided in `training/contraction/train_cm.py`; note that the structure of this code is different from that in the previous two examples since contraction metrics must be trained using trajectory data rather than randomly sampled points. Additional examples of contraction metric training can be found in the code accompanying [14]: https://github.com/sundw2014/C3M.

An example of the trained controller tracking a (previously unseen) random reference trajectory is shown in Fig. 7, and the value of the learned contraction metric over time is shown to decrease exponentially in Fig. 8. Note that even though the neural network control policy only receives the reference state and input at the current instant in time (the original MPC policy receives a 1-s window of future reference states and inputs), it is able to successfully track a trajectory that was not part of the training dataset.

F. History of Certificate Learning

The earliest proposal for using a neural network for Lyapunov function synthesis comes from Prokhorov's 1994 paper [70],

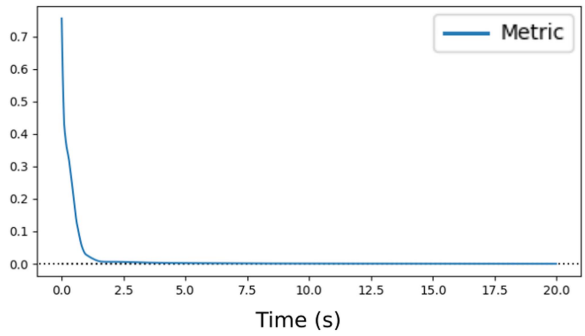

Fig. 8. Metric value $(x - x^*)^T M(x - x^*)$ over time as the learned control policy tracks the reference policy. As expected, the contraction metric value decreases exponentially as the system converges to the reference trajectory.

but this technique relies on neural networks as a model of computation rather than as a function approximator, preventing the approach from scaling. Serpen [71] in 2005 first framed the neural network certificate-representation problem as an optimization problem analogous to (19). However, neither of these works includes a demonstration of their learned certificates. Noroozi et al. [72] demonstrated learning a neural network approximation of a Lyapunov function for simple 2-D and 3-D nonlinear dynamical systems in 2008, while Petridis and Petridis [73] in 2006 employed a neural certificate trained via a genetic algorithm to verify the stability of a recurrent neural network. These early works are typically restricted to Lyapunov certificates and involved relatively small dynamical systems.

It is not until after the explosive growth in interest and computational power applied to neural networks in the mid-2010s that neural certificates were rediscovered in the context of robotics and control. One of the first practical implementations came in 2018, where Richards et al. [47] frame certificate synthesis as a classification problem, with the goal of finding the largest RoA certified by a neural network certificate (they search only for a certificate and assume that a controller is given). This approach iteratively grows the approximate certified RoA outward from a fixed point, sampling from points near the boundary of the RoA to expand its training dataset. The Lyapunov conditions are only enforced for points that lie within the current estimate of the RoA (as determined by simulating forward from those points), allowing this method to estimate the RoA for systems that are only locally stable. Richards et al. [47] demonstrate their method on a feedback-stabilized inverted pendulum in simulation, where it outperforms other methods for estimating the RoA (particularly, local linearization and SoS).

Since this initial work, several other authors have also addressed this issue of finding a certificate for a fixed predefined controller. Abate et al. [49] and Ahmed et al. [74] incorporate an SMT solver to verify the learned Lyapunov certificates and provide counterexamples for training, and Peruffo et al. [3] take a similar approach to learning barrier functions (Peruffo et al. [3], Abate et al. [49], and Ahmed et al. [74] later unified these methods in a single software framework in [50]). Singh et al. [4] learns a contraction metric in the case when the control policy is known but applies this technique for system identification rather than controller verification (the contraction metric constrains the learned dynamics model to be stabilizable).

The first work to propose jointly optimizing for a control policy and Lyapunov certificate is that of Chang et al. [8]. This work employs an SMT-based learner–verifier architecture using the dReal SMT solver and alternates between optimizing the empirical loss (19) and searching for counterexamples to guide training. Chang et al. [8] demonstrate that jointly optimizing for a controller and a certificate yields improved performance, in the sense of a larger RoA, than either optimizing for a fixed controller (using LQR) or using SoS techniques.

Since Chang et al.'s work, other authors have expanded this joint control-certificate learning approach to other certificate functions. For instance, Qin et al. [11] jointly learn a barrier function and controller, while the authors of [10], [21], [56], [57], [58], and [59] learn CLF and CBF (implicitly combining the certificate and control policy). Other authors, notably Tsukamoto et al. [55], [75] have extended the neural certificate approach to contraction metrics (see [5] for an overview of these approaches). In addition to different types of certificate, later works have expanded this framework to more challenging problems in control, such as multiagent control in [11], black-box dynamical models in [67], the RL context in [9], and the robust case in [10]. Many of these later works are particularly notable for providing theoretical and algorithmic contributions to address difficulties that arise in practice (such as control from observations), which we discuss next in Section V.

V. Implementation Considerations

Successfully deploying a control system on hardware is more challenging than demonstrating the performance of that system in simulation. In hardware, effects such as state estimation error, control frequency and delay, external disturbances, unmodeled dynamics, and actuator limits can degrade the safety and stability of the controller unless suitable steps are taken to mitigate these effects. This section will discuss a number of strategies to mitigate these effects.

A. Mitigating Measurement Uncertainty

A notable gap between simulation and real hardware is the availability of high-quality state estimates. As a result, it is natural to ask how state estimation errors can affect the guarantees of certificate-based controllers. The most directly relevant work in this vein is that by Dean et al. [76] deriving conditions under which barrier functions are robust to measurement errors. In this framework, the authors assume access to a state estimate with bounded error, i.e., $\hat{x} = x + e(x)$ with error bounded by some known function $||e(x)|| \leq \epsilon(x)$. Given this assumption and knowledge of the Lipschitz constants of the barrier function and its Lie derivatives, Dean et al. [76] derive a tightened version of condition (13) that guarantees control invariance despite state estimation uncertainty. These conditions can be used to derive quantitative requirements on the state estimation accuracy needed to enable a certificate-driven approach to control.

Although Dean et al. [76] provide the theoretical basis for barrier certificates that are robust to errors in state estimation,

these "measurement-robust" barrier functions are strictly more difficult to synthesize than standard barrier functions [due to the tightening of condition (13)]. We are not aware of any published work learning (or otherwise automatically synthesizing) measurement-robust barrier functions, but we anticipate that future work in this area will bridge this gap, since neural synthesis methods can be easily adapted to these tightened conditions.

B. Observation-Feedback Control Certificates

In Section V-A, we ask how certificates might be adapted to handle an uncertain state estimate; a natural extension of this question is how certificates might be adapted to the case where the control policy is a function of observations themselves (i.e., observation-feedback or "pixels-to-torques" control). This is an active area of research in the certificate learning setting. Early works in this direction assume that the observations are first converted to state estimates with some bounded error (e.g., [76], discussed in Section V-A). Later works have provided proofs of concept for certificates defined directly in the space of observations, without recourse to an intermediate state estimate [21]. These works typically rely on some approximate model of how control actions affect future observations. For some observations, we can construct an analytical approximate model, for example, by approximating future Lidar observations as affine transformations of previous observations [21], or modeling force measurements using spring connections to the environment [77]. For more complicated observations, particularly image feedback, there has been some work toward approximating future images using generative models [78], [79], but there is still work to be done to understand the approximation errors of deep generative models and the impact of those errors on the safety of a certificate-based controller.

C. Robustness to Disturbance and Model Uncertainty

A common goal when designing control systems is to ensure that disturbances to the nominal system dynamics do not adversely affect the safety or performance of the system. Some disturbances, such as unmodeled aerodynamic effects [80], can be represented as unknown forces added to the known system dynamics, while others, such as uncertain mass or inertia, must be modeled as potentially multiplicative disturbances [10]. Depending on the prior knowledge of the disturbance, different certificate-based control strategies (each with distinct advantages and drawbacks) may be applied to ensure safety.

When little is known about an additive disturbance beyond its bound, i.e., $\dot{x} = f(x,u) + d, d \in \mathcal{D}$ for a known disturbance set $\mathcal{D}$, then learning a contraction metric will guarantee that any bounded disturbance will lead to a bounded tracking error in the worst case. When more information about the structure of the disturbance is known, for instance when $\dot{x} = f(x,u,d)$ and f is known to be affine in d for any fixed (x,u) (as is the case for additive disturbance as well as uncertainty in many physical parameters), then robust variants of Lyapunov and barrier functions exist to guarantee safety and stability despite this uncertainty [10]. These robust methods are complementary with the adaptation techniques discussed in Section V-D, where we might wish to combine a partially learned dynamics model with robustness to error in those learned dynamics [13], [62].

The most challenging case of model uncertainty is when there is no *a priori* knowledge of the structure (e.g., additive, control-dependent, etc.) or extent of the uncertainty. In these cases, the model uncertainty may be estimated from data; Taylor et al. [81] discuss this case for control-affine systems. While Taylor et al. [81] do not include a method for synthesizing a CLF or CBF certificate in this setting, the authors provide a theoretical discussion on how to derive a control input from these certificates [extending the certificate-based QP controllers (10a) and (14a)] and when those controllers are feasible despite model uncertainty.

D. Certificate Adaptation

In Section V-C, we discussed robust certificate techniques that take a worst case approach to handling model uncertainty. This raises the natural question: instead of trying to be robust to a wide range of possible model errors, could we rely on the nominal model for the bulk of the training process and then transfer the learned certificates to the true model using a smaller amount of data from the real system, thus "adapting" the certificate to the true model?

One line of work in certificate adaptation takes inspiration from system identification: if the difference between the nominal and real models can be learned, then certificate-based controllers can adjust for this difference at runtime. For example, if CBF h is known for the nominal system (which we write as $\hat{f}$ and $\hat{g}$ to distinguish from f and g for the real control-affine system), then a safe controller can be found by solving the CBF QP [61]

$$\min_{u \in \mathcal{U}} \ ||u||^2 \tag{20}$$

$$\text{s.t. } L_f h(x) + L_g h(x) u \leq -h(x). \tag{21}$$

Unfortunately, the constraint of this QP depends on the unknown dynamics. If the nominal model differs from the true dynamics with error terms $\Delta_f(x)$ and $\Delta_g(x)$ (i.e., $\dot{x} = \hat{f}(x) + \hat{g}(x)u + \Delta_f(x) + \Delta_g(x)u$), then this introduces an error term

$$\min_{u \in \mathcal{U}} \ ||u||^2 \tag{22}$$

$$\text{s.t. } L_{\hat{f}} h(x) + L_{\hat{g}} h(x) u + a(x) + b(x) u \leq -h(x) \tag{23}$$

where $a = L_{\Delta f(x)} h$ and $b = L_{\Delta g(x)} h$. Choi et al. [13] and Castenñeda et al. [62] demonstrate that a neural network can be used to learn either the model residue (Δ_f, Δ_u) or the Lie derivative residue (a,b) via RL and Gaussian process regression, respectively. They apply their model-learning adaptation technique to a simulated bipedal walker with 14 state dimensions, demonstrating successful adaptation when the mass and inertia of each robot link are changed [13]. In a similar vein, Taylor et al. learn a and b using an episodic learning framework for both barrier [61] and Lyapunov [2] certificates applied to a Segway model in simulation.

E. Certificate Verification

Several methods for verifying learned certificates have been proposed, each with its own advantages and drawbacks. Here,

we will discuss the three most common methods for verification: probabilistic methods based on generalization error bounds, Lipschitz arguments, and optimization-based methods.

The first framework for verifying learned certificates, generalization error bounds, is the least computationally expensive but provides relatively soft guarantees. These methods typically sample a large number of points from the state space, check whether the certificate is valid with some margin, and then use this margin to extrapolate (with high probability) to claim that the certificate is valid continuously throughout the state space. These claims are typically based on either statistical learning theory (e.g. [11, Proposition 2] or [44]) or almost Lyapunov theory [82] (see [9] for an application to certificate verification). This verification strategy is perhaps the most convenient, since it requires little infrastructure in addition to that used for training the neural certificate; however, generalization bounds tend to be conservative, and in some cases, it may be desirable to provide firm (i.e., nonprobabilistic) soundness guarantees.

The second common verification framework provides such deterministic guarantees and is based on the Lipschitz constant. Recall that a scalar function $f(x) : \mathbb{R}^n \mapsto \mathbb{R}$ is said to be Lipschitz continuous with Lipschitz constant L if for any two $x_1, x_2 \in \mathbb{R}$, it holds that $|f(x_1) - f(x_2)| \leq L||x_1 - x_2||$ for an appropriate norm. Certificate learning works that make a Lipschitz argument for verification (see, e.g., [47] and [83]) compute the Lipschitz constant for each certificate constraint L_{c_i} and then check the margin of constraint satisfaction at each point on either a fixed-size [47] or adaptive [83] grid. If the constraint is satisfied with some margin (i.e., $c_i(x_j, V) \leq -L_{c_i}\tau$, where 2τ is the distance between adjacent grid points), then the constraint is guaranteed to be satisfied continuously between those points. This strategy is convenient, since it can be applied to any Lipschitz continuous constraint without much overhead. However, it has a number of drawbacks. First, the use of the Lipschitz constant necessarily introduces conservatism by assuming worst case variation between sampled points (and the estimated Lipschitz constant itself can be either difficult to estimate [84] or provide a very loose upper bound). Second, checking the constraints over a grid of points incurs the curse of dimensionality, limiting applications to higher-dimensional systems. Third, this after-the-fact verification scheme does little to guide the training process, since it is only used after training is complete (this drawback applies to both Lipschitz arguments and generalization error bounds).

The third common category of verification methods, based on optimization, fills this gap by running periodically throughout the training process to verify the certificate as it is learned (and provide feedback in the form of counterexamples where the certificate is not yet valid). These approaches typically take the form of a *learner–verifier* architecture, sometimes also referred to as counterexample-guided inductive synthesis (CEGIS [3], [8], [49], [50], [64]). In these architectures (illustrated in Fig. 9), the learner is responsible for training the certificate neural network, while the verifier periodically checks the learned certificate. If the certificate is valid, the verifier stops the training early; otherwise, it provides counterexamples to enrich the training dataset.

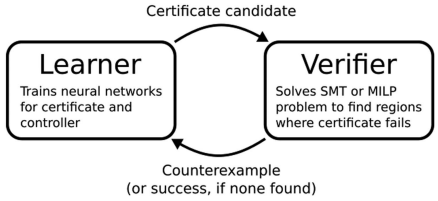

Fig. 9. Learner–verifier architecture. Several different technologies may be used for the verifier, including SMT and MILP solvers.

Depending on how the certificate and system dynamics are formulated, the verifier can be implemented using a number of different technologies. If the underlying dynamics are piecewise affine and all neural networks are encoded using ReLU activation functions, then the verification problem can be represented as a mixed-integer linear program (MILP) and solved using off-the-shelf MILP solvers, as proposed in [64]. If the underlying dynamics are nonlinear, then the verification problem can be solved using SMT. SMT problems can be solved using dedicated solvers such as dReal [85]. For examples of this approach, see [3], [8], [49], [50], and [74].

Both MILP- and SMT-based methods help address the certificate verification problem, with the added benefit of speeding the training process by providing counterexamples. However, both the methods are computationally expensive, and their complexity typically grows exponentially with the number of neurons in the certificate network. In practice, this poor scalability limits the applicability of these methods; for example, the MILP-based method in [64] is demonstrated on networks containing only 16 neurons. More advanced MILP-based neural network verification tools exist, such as MIPVerify [86], but empirical results suggest that they are currently limited to networks with less than 200 neurons [87]. SMT-based methods have been demonstrated on networks of up to 30 neurons in [3]. It is possible that recent advances in neural network verification (see [87] for a relevant survey) or adversarial analysis and training of neural networks [88] may improve the scalability of verifier architectures; however, most approaches are currently limited to small neural networks.

As a final note on the topic of verification, it is important to note that control certificates can still be valuable without exhaustive verification, for two reasons. First, even nonverified certificates can act as useful supervision for training control policies [9]. Second, it is possible to deploy a nonverified Lyapunov or barrier certificate safely using a real-time safety monitor. This monitor tracks the rate of change of the certificate and checks whether the derivative condition (4c) (for Lyapunov functions) or (12) (for barrier functions) is violated; any such violation indicates that the certificate is no longer able to guarantee stability or safety and the system should enter a fail-safe mode.

F. Effects of Actuator Limits

Similarly to constrained MPC, certificate-based controllers, such as the QP given in (10a), can easily adapt to actuator limits [20]. The control policy is simply modified to respect the

actuator limits (a CLF-based policy is shown here for example)

$$\min_{u\in\mathcal{U}} \ ||u||^2 \tag{24}$$

$$\text{s.t. } L_fV(x) + L_gV(x)u \leq -cV(x) \tag{25}$$

$$u \in \mathcal{U} \tag{26}$$

where $\mathcal{U}$ is the set of admissible controls. When $\mathcal{U}$ is a polytope, then this policy remains a QP and can be solved efficiently online. The only complication is ensuring that this QP will be feasible. A neural certificate architecture can accommodate this constraint through the use of differentiable convex programming [60], which allows backpropagation through the result of solving a QP like (24). To train the certificate network in this context, the QP policy is relaxed

$$\min_{u,r} \ ||u||^2 + r^2 \tag{27}$$

$$\text{s.t. } L_fV(x) + L_gV(x)u \leq -cV(x) + r \tag{28}$$

$$u \in \mathcal{U} \tag{29}$$

$$r \geq 0 \tag{30}$$

and the certificate network is trained with an additional loss term to minimize r; see [10] for an example. Although solving this optimization problem at training time incurs an additional offline cost, it enables a learning-enabled controller to provide assurance that it will respect actuation limits at runtime (as contrasted with black-box-learned policies such as those derived from RL, which make no such guarantees).

G. Attainable Control Frequencies

Another important consideration when deploying controllers to hardware is the computational burden of running that controller in real time. This can become an issue when the complexity of the control task increases; for example, it can be difficult to run robust MPC algorithms in real time, due to the additional complexity of considering the effect of disturbances over a long horizon [10]. In the case of MPC in particular, the need to consider the vehicle's safety over a multistep horizon imposes a significant computational cost. In contrast, certificate-based controllers effectively encode long-term system properties (like safety and stability) into local properties of the certificate function; synthesizing a certificate can be seen as *compiling* long-term behaviors into local properties. As a result, certificate-based controllers need to consider only a single-step horizon when evaluating a controller such as (10a). Empirically, this results in at least an order of magnitude increase in attainable control frequency as compared to robust MPC [10]. Because a neural certificate control framework shifts the burden of computation to the offline stage (synthesizing the certificate), it is well suited for deployment on robots with limited computational resources or in situations where high control frequencies are required.

VI. Case Studies

In this section, we present a series of instructive examples to demonstrate how the certificate learning techniques discussed in this survey can be applied to practical problems in robotics. The first example demonstrates a straightforward application of certificate learning: given a nonlinear model of an autonomous car with uncertainty, we synthesize a trajectory-tracking controller that is robust to variation in the reference trajectory. Second, we demonstrate combining a learned CLF with a learned CBF to synthesize an observation-feedback controller for a mobile robot navigating a cluttered unknown environment. We also include results from a hardware demonstration for this second example.

Source code for each of these examples, along with pretrained models, is available on the accompanying website: https://github.com/MIT-REALM/neural_clbf. This code is intended to serve as a well-documented reference implementation for many of these techniques, and we hope that they will provide a useful starting point for any readers interested in applying these techniques to their own problems.

A. Learning a CLF for Complex Dynamics

Our first example involves learning a stabilizing tracking controller for a nonlinear model of an autonomous car. We use the single-track car model from the CommonRoad benchmarks [89]. This model includes not only nonlinear steering dynamics, but also the effects of friction and load transfer between the front and rear axles as the car accelerates. The model includes seven state dimensions and two control inputs: steering angle velocity and longitudinal acceleration. More details on the dynamics can be found in [89] or as implemented in the code accompanying this article. We modify the dynamics from [89] slightly to express the car's state relative to a reference trajectory. This example builds on experiments presented in [10].

These dynamics are not amenable to traditional control synthesis techniques (such as SoS, which fails due to numerical issues even when approximating these dynamics using polynomials [10]). However, we can apply the techniques introduced in Section IV to learn a controller based on a CLF that stabilizes this system about the reference trajectory. This section will explain how we apply these techniques to this specific problem.

To simplify the learning problem, we look for a CLF, which defines the optimization-based control policy (10a) and removes the need to simultaneously search for a control policy (in effect, the control policy is parameterized by the CLF). We then instantiate the optimization problem (17) with the specific constraints governing CLFs, given in (6a)–(6c). We encode the CLF with learnable parameters θ as $V_\theta(x) = w_\theta(x)^T w_\theta(x)$, where w_θ is a fully connected neural network with $\tanh$ activation functions and size $9 \times 64 \times 64$ (with the car's seven state dimensions expanded to nine by replacing two angular dimensions with their sine and cosine). To improve the convergence of the learning process, we linearize the dynamics, construct an LQR controller for the linearized system, and find the Lyapunov function for the closed-loop linear system $V_{\text{lin}} = \frac{1}{2}x^T P x$, where P is the solution to the Lyapunov matrix equation. This linear solution is used to initialize the learned CLF by training the learned CLF to imitate V_{lin} for 11 epochs; once this initialization phase is complete, we do not reference the linearized Lyapunov function and train to satisfy the constraints (6a)–(6c) (training continues to 25 total epochs).

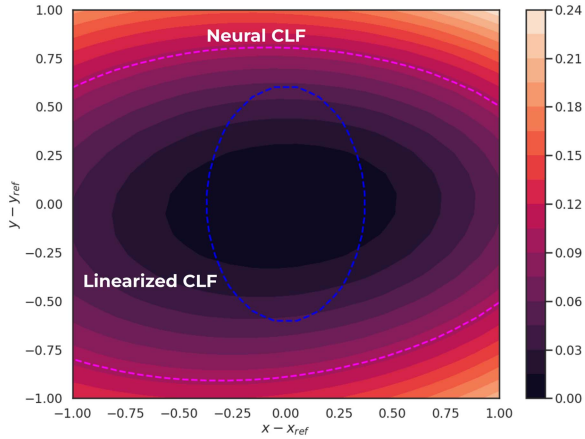

Fig. 10. CLF learned for the single-track car. Lighter colors indicate higher values of V; as expected, V decreases to zero at the origin (corresponding to zero error relative to the reference trajectory). By ensuring that trajectories flow "downhill" on this landscape, a CLF-QP control policy can robustly track the reference trajectory. The 0.1-level sets of the neural CLF and the CLF found via linearization are overlaid to highlight the difference.

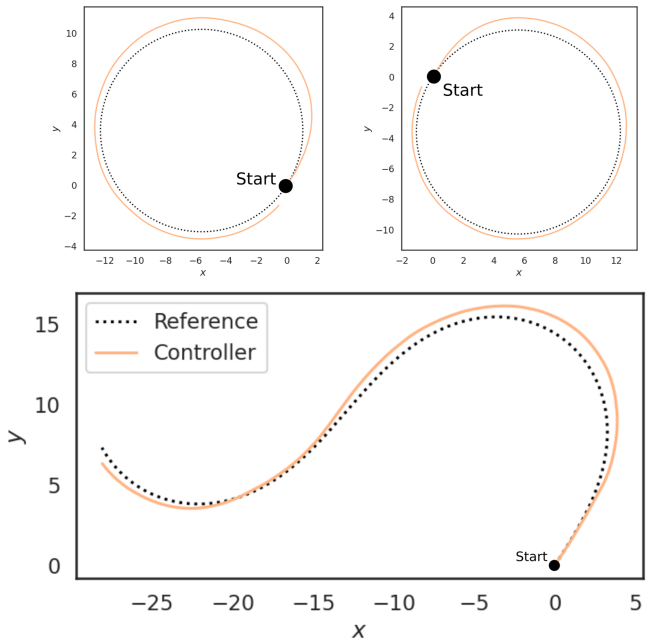

Fig. 11. Tracking performance of a controller based on a learned CLF. Note that the system is only trained on "scenarios" like the circular reference paths shown above, but it successfully generalizes to more general reference paths through the use of the CLF.

Our choice to express the vehicle's state relative to the reference trajectory introduces a new challenge to this leaning problem. We wish to ensure that our learned CLF is valid not only for the reference trajectories seen during training, but also for the range of trajectories we might expect to see in practice. To address this issue, we can draw on work creating robust variants of CLF certificates [10]: we select two "scenarios" characterizing the uncertainty in the reference trajectory (representing maximum steering effort to the right and left) and then train the CLF to be valid in both of these scenarios. Using our knowledge of the underlying model and the structure of the CLF conditions (6c) (which are affine in the model uncertainty, as the authors of [10] demonstrate), we can prove that the trained controller generalizes from these scenarios to more general trajectories.

A CLF for this system can be trained by running the code in `training/train_single_track_car.py`, and a pre-trained model can be evaluated using the code in `evaluation/eval_single_track_car.py`. The CLF learned using this method is shown in Fig. 10 and the tracking performance is shown in Fig. 11. Fig. 10 clearly shows the difference between the neural CLF and the CLF found by the continuous-time Lyapunov equation for the linearized system dynamics.

B. Safe Visual-Feedback Control With Barrier Functions

Our second example demonstrates the application of certificates to "full-stack" robotics problems that integrate perception and control. In particular, we can combine a CBF with a CLF (both learned using neural networks) to certify the safety and liveness of a perception-feedback controller capable of safely navigating previously unseen environments. At a high level, this system uses a CBF to ensure that it remains safe at all times, and it uses a CLF to guide itself to the goal. To avoid getting stuck when the CLF and CBF conflict, the controller is capable of temporarily suspending the CLF as needed to move around obstacles (this corresponds to tracing a level set of the CBF, i.e., maintaining a constant level of safety, while seeking a state from which the CLF can safely decrease). To enable generalization to new environments, the CBF and CLF are learned as functions of observations (Lidar data and range and bearing to the goal) rather than states. More details on this example can be found in [21]. This example shows how learned certificates can be used to ensure safety as part of a larger robotics architecture.

We consider an autonomous ground robot with discrete-time nonlinear Dubins car dynamics and the ability to observe its environment via local Lidar observations, represented as range o^i along each ray ($i = 1, \ldots, N_r$). We also assume that the robot can measure its range ρ and bearing ϕ relative to a beacon at the goal. We define the CBF $h(o) = h_\sigma(o) - \min_i ||o^i|| + d_c$, where h_σ is a permutation-invariant neural network with parameters σ and the second and third terms impose the prior that the barrier function should correlate with distance to the nearest obstacle. The CLF is represented as $V(\rho, \sin\phi, \cos\phi) = V_\omega(\rho, \sin\phi, \cos\phi) + \rho^2 + (1 - \cos\phi)/2$, where V_ω is a neural network and the second and third terms impose the prior that the Lyapunov function should correlate with distance to the goal. These neural networks are trained in a single randomly generated 2-D environment using the self-supervision method discussed in Section IV.

Although the state dynamics of this system are control affine, as discussed in Section II, the dynamics in observation space are not. As a result, we cannot use a simple QP control policy like (10a); instead, we must find a control input that satisfies the certificate conditions (5b) and (12) by searching over the action space directly. This is computationally feasible, e.g., by discretizing the action space, but extending this approach to higher dimensional action spaces, perhaps by applying stochastic optimization methods such as CMA-ES [90], is an interesting area

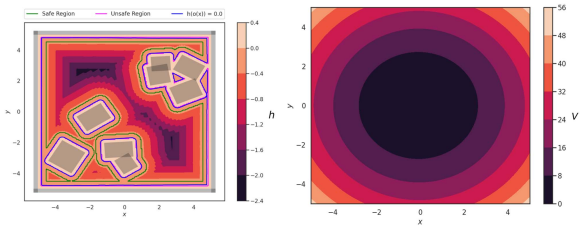

Fig. 12. CBF h (left) and CLF V (right) learned for navigating a cluttered environment (lighter colors indicate higher values). The learned CBF correlates with distance to the nearest obstacle, while the learned CLF correlates with distance from the goal. Since the CBF is learned as a function of observations, it can easily generalize to new environments.

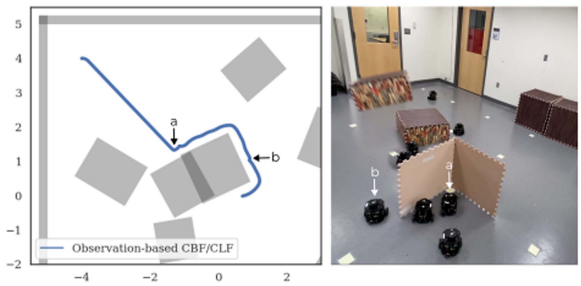

Fig. 13. Certificate-based hybrid perception-feedback controller navigating a previously unseen environment in both simulation (left) and hardware (right). In the hardware demonstration, a new obstacle is thrown in front of the robot partway through the experiment; the CBF allows the robot to gracefully avoid this obstacle. Labels (a) and (b) indicate where the policy switches from goal seeking to exploratory and vice versa, respectively.

of future work. Finally, we must handle the complication that arises when the system cannot simultaneously preserve safety by satisfying the barrier function condition (12) while moving toward the goal [satisfying the Lyapunov condition (5b)]. To do this, we can construct a simple hybrid controller that switches between goal-seeking and exploratory modes, preserving safety in all modes by enforcing the barrier function conditions but able to temporarily disable the Lyapunov conditions in order to avoid getting stuck in local minima. In addition, the controller presented in [21] defines a constrained stochastic control policy in the exploratory mode to encourage exploration of the state space. The details of this switching strategy and a proof of its liveness can be found in [21].

The learned CBF and CLF are shown in a random environment in Fig. 12, and the hybrid controller is shown navigating a previously unseen environment in Fig. 13. Even though these certificates were trained using a single randomly generated environment, the controller is able to navigate new environments safely without adapting its certificate. To demonstrate the potential of certificate-enabled controllers to handle some of the implementation challenges in Section V, we also include results deploying this controller on hardware in a laboratory environment in Fig. 12.

VII. Limitations and Future Work

In our opinion, certificate-based learning for control has great promise but a number of drawbacks, which we discuss in this section. In addition to these limitations, we also highlight promising directions for future research in this area.

A. Limitations

1) Data Requirements and Generalization: Presently, the field's understanding of the amount of data required to successfully train a neural certificate is based largely on intuition and experience; there is not yet a firm theoretical footing to predict the data required by a certificate learning framework. Some works have made some promising progress in this direction; notably, Boffi et al. [44] prove that certificate learning is *asymptotically consistent* in the sense that as more data points are added to the training set, the maximum volume where the trained certificate fails to be valid shrinks to zero, but their analysis is restricted to systems that are known *a priori* to be stable. It remains to extend this theory to the case of general nonlinear systems, or systems where the controller is learned alongside a certificate.

Closely related is the issue of generalization error, which relates a learned certificate's performance on a finite training set with its performance on the full state space (as discusses in Section V-E). Some works [11], [44] have applied statistical learning theory to provide probabilistic upper bounds on the generalization error (i.e., upper bounds that hold with some high probability), but these bounds tend to be conservative. There have been some preliminary attempts to apply almost-Lyapunov theory [82] to certificate learning [9], but it remains to formalize this connection.

2) Scalable Verification: As mentioned in Section V-E, there has yet to be an effective solution to the certificate verification problem that scales to networks involving > 100 neurons. Probabilistic guarantees based on generalization error bounds [11], [14], [44] or almost Lyapunov theory [82] have the potential to scale by replacing exhaustive verification with probabilistic guarantees, but this theory has yet to be fully developed and integrated into the certificate learning process (all existing applications of these techniques are *post hoc*).

B. Future Work

In addition to addressing these limitations in the existing theory of certificate learning, we believe that there are several promising directions to extend this theory to more complex controls and robotics problems.

1) Model-Free Certificate Learning and Theoretical Connections to RL: Although works like [67] and [9] deploy certificate learning in the black-box dynamics model and unknown-model settings, respectively, there remains much room to explore both practical and theoretical connections between certificate learning and RL. On a practical level, Chang and Gao [9] and Westenbroek et al. [36], [37] all propose to learn the parameters of a Lyapunov function [9], CLF [36], or CLF and CBF together [37] via RL.

On a theoretical level, deep connections between certificate learning and RL remain unexplored. For example, Berkenkamp et al. [34] show that it is possible to construct a reward function for specific RL problems so that the corresponding value function is also a Lyapunov function, but it is not well understood whether this is possible more generally. In particular, what conditions must the reward function satisfy so that the value function implies a Lyapunov function? Is it always possible to construct such a reward, even when nonstability objectives are

included in the reward function? Work on constrained Markov decision processes (cMDPs) suggests that it is possible to derive Lyapunov functions from a cMDP's cost metric [39], and later work [40] has extended this approach to continuous state and action spaces. We anticipate that these questions will be the basis for exciting future research at the intersection of RL and certificate learning.

2) Heterogeneous Multiagent Certificates: With an eye toward deploying large fleets of autonomous robots (e.g., in drone delivery or driving contexts), some works have applied certificate learning to systems with multiple agents [11], [66]. However, these works focus exclusively on the case when every agent in the fleet has identical dynamics and constraints, as this allows a single certificate function to be learned for all agents at once. To extend these methods to heterogeneous multiagent systems (e.g., a fleet of multiple autonomous delivery trucks, each with its own fleet of unmanned aerial vehicles), it may be possible to apply compositional verification methods [91], [92], [93] to learn a certificate for various subsets of agents and combine them with rigorous safety guarantees.

3) Distributed and Network Control: Related to the subject of heterogeneous multiagent control, we believe that there is an opportunity for fruitful research in learning certificates for distributed systems and networks. In addition to compositional verification techniques discussed above, this setting raises a number of interesting issues. The first issue is control and communication delay [94], which has been studied in the control literature for standard CBFs [95] but has not yet been applied in a certificate learning context. The second issue is fault tolerance [96], since a distributed control system should ideally be resilient to failures in individual nodes; this will likely require building on existing work in resilient control (see, e.g., [97]) to develop frameworks for resilient certificate learning. The third issue is scalability; contraction metrics in particular will likely require some decomposition techniques to avoid computing the eigenvalues of large matrices at training time. The application of certificate learning in this area also presents an interesting opportunity to apply graph neural networks [98], [99] to learn network control certificates.

VIII. Conclusion

In this survey, we reviewed an emerging suite of methods for automatically synthesizing safe controllers for nonlinear systems—the neural certificate framework. This framework builds on established control-theoretic concepts (see Section II) by applying the representational power of neural networks (see Section IV) to synthesize control certificates such as Lyapunov functions (see Section IV-C), barrier functions (see Section IV-D), and contraction metrics (see Section IV-E). We also discussed several practical and theoretical challenges within this framework, as well as proposed tools from the literature to mitigate these challenges (see Section V). Finally, we presented two case studies (see Section VI) and discussed directions for future work (see Section VII). We hope that this review provides an accessible jumping-off-point and high-level perspective on this emerging field.

A general framework for automatic certificate synthesis has long eluded control theorists. Since the development of Lyapunov functions in the late 1800s and modern control theory in the mid-to-late 1900s, a number of techniques have brought us progressively closer to this goal. LP-based simulation-guided synthesis [31] and SoS programming [6] provide important steps in this direction, each addressing some of the restrictions that applied to earlier works. In this context, neural certificates represent an important step forward for the state of the art for the computational synthesis of nonlinear controllers. Neural certificates do not require a choice of hand-designed basis, as LP-based methods do, nor are they limited to systems with polynomial dynamics, as SoS methods are. Owing to their generality, control architectures based on neural certificates have great promise for a wide range of robotics problems. We anticipate that neural certificates will see increasing adoption among practicing roboticists solving nonlinear safe control problems.

References

[1] P. Giesl and S. Hafstein, "Review on computational methods for Lyapunov functions," *Discrete Continuous Dyn. Syst.—B*, vol. 20, no. 8, pp. 2291–2331, 2015.

[2] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic learning with control Lyapunov functions for uncertain robotic systems," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2019, pp. 6878–6884.

[3] A. Peruffo, D. Ahmed, and A. Abate, "Automated and formal synthesis of neural barrier certificates for dynamical models," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2021, pp. 370–388.

[4] S. Singh, S. M. Richards, V. Sindhwani, J.-J. E. Slotine, and M. Pavone, "Learning stabilizable nonlinear dynamics with contraction-based regularization," *Int. J. Robot. Res.*, vol. 40, 2020, Art. no. 027836492094993.

[5] H. Tsukamoto, S. J. Chung, and J. J. E. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annu. Rev. Control*, vol. 52, pp. 135–169, 2021.

[6] A. A. Ahmadi and A. Majumdar, "Some applications of polynomial optimization in operations research and real-time decision making," *Optim. Lett.*, vol. 10, no. 4, pp. 709–729, 2016.

[7] M. Srinivasan, M. Abate, G. Nilsson, and S. D. Coogan, "Extent-compatible control barrier functions," *Syst. Control Lett.*, vol. 150, 2021, Art. no. 104895.

[8] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov control," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 3245–3254.

[9] Y.-C. Chang and S. Gao, "Stabilizing neural control using self-learned almost Lyapunov critics," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 51–57.

[10] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural Lyapunov-barrier functions," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 1724–1735.

[11] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, "Learning safe multi-agent control with decentralized neural barrier certificates," in *Proc. Conf. Learn. Represent.*, 2021. [Online]. Available: https://openreview.net/forum?id=P6_q1BRxY8Q

[12] W. Xiao, R. Hasani, X. Li, and D. Rus, "BarrierNet: A safety-guaranteed layer for neural networks," *IEEE Trans. Robot.*, 2022.

[13] 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," in *Proc. Robot.: Sci. Syst. Conf.*, 2020. [Online]. Available: http://www.roboticsproceedings.org/rss16/p088.html

[14] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," in *Proc. Conf. Robot Learn.*, 2020. [Online]. Available: https://corlconf.github.io/corl2020/paper_347/

[15] A. Agrawal and K. Sreenath, "Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation," in *Proc. Robot.: Sci. Syst. Conf.*, 2017. [Online]. Available: http://www.roboticsproceedings.org/rss13/p73.html

[16] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[17] W. M. Haddad and V. Chellaboina, *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton, NJ, USAPrinceton Univ. Press, 2008.

[18] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 876–891, Apr. 2014.

[19] R. A. Freeman and P. Kokotović, *Robust Nonlinear Control Design*. Boston, MA, USA: Birkhäuser, 1996.

[20] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[21] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1904–1911, Apr. 2022.

[22] W. Lohmiller and J.-J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, vol. 34, no. 6, pp. 683–696, 1998.

[23] I. R. Manchester and J.-J. E. Slotine, "Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 3046–3053, Jun. 2017.

[24] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 2242–2253.

[25] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, no. 1, pp. 411–444, 2022.

[26] MathWorks, "Continuous Lyapunov equation solution," *MATLAB Documentation*, accessed Jan. 3, 2022. [Online]. Available: https://www.mathworks.com/help/control/ref/lyap.html

[27] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[28] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 4054–4061.

[29] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *Proc. Annu. Amer. Control Conf.*, 2018, pp. 585–590.

[30] W. Tan and A. Packard, "Stability region analysis using polynomial and composite polynomial Lyapunov functions and sum-of-squares programming," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 565–571, Mar. 2008.

[31] J. Kapinski, S. Sankaranarayanan, J. V. Deshmukh, and N. Aréchiga, "Simulation-guided Lyapunov analysis for hybrid dynamical systems," in *Proc. 17th Int. Conf. Hybrid Syst.: Comput. Control*, 2014, pp. 133–142.

[32] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*, E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem Eds. Cham, Switzerland:Springer, 2018, pp. 305–343.

[33] 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 *Proc. 33rd AAAI Conf. Artif. Intell./31st Innov. Appl. Artif. Intell. Conf./9th AAAI Symp. Educ. Adv. Artif. Intell.*, 2019, vol. 33, pp. 3387–3395.

[34] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 908–919.

[35] X. Li and C. Belta, "Temporal logic guided safe reinforcement learning using control barrier functions," 2019, *arXiv:1903.09885*.

[36] T. Westenbroek, F. Castañeda, A. Agrawal, S. S. Sastry, and K. Sreenath, "Learning min-norm stabilizing control laws for systems with unknown dynamics," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 737–744.

[37] T. Westenbroek, A. Agrawal, F. Castañeda, S. S. Sastry, and K. Sreenath, "Combining model-based design and model-free policy optimization to learn safe, stabilizing controllers," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 19–24, 2021.

[38] H. Ma, C. Liu, S. E. Li, S. Zheng, and J. Chen, "Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning," in *Proc. 4th Annu. Learn. Dyn. Control Conf.*,2022, vol. 168, pp. 97–109.

[39] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in Proc. 32nd Int. Conf. Neural Inf. Process. Syst., 2018, pp. 8103–8112.

[40] Y. Chow, O. Nachum, M. Ghavamzadeh, and E. Guzman-Duenez, "Lyapunov-based safe policy optimization for continuous control," in *proc. RL4RealLife workshop Int. Conf. Mach. Learn.*, 2019. [Online]. Available: https://openreview.net/forum?id=SJgUYBVLsN

[41] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust control barrier-value functions for safety-critical control," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 6814–6821.

[42] S. Bansal and C. J. Tomlin, "DeepReach: A deep learning approach to high-dimensional reachability," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 1817–1824.

[43] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin, "A classification-based approach for approximate reachability," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 7697–7704.

[44] N. M. Boffi, S. Tu, N. Matni, J. J. E. Slotine, and V. Sindhwani, "Learning stability certificates from data," in *Proc. Conf. Robot Learn.*, 2020, pp. 1341–1350.

[45] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Netw.*, vol. 2, no. 3, pp. 183–192, 1989.

[46] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Mach. Learn.*, vol. 14, no. 1, pp. 115–133, 1994.

[47] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proc. Conf. Robot Learn.*, 2018, pp. 466–476.

[48] J. Z. Kolter and G. Manek, "Learning stable deep dynamics models," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11128–11136.

[49] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of Lyapunov neural networks," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 773–778, Jul. 2021.

[50] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks," in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–11.

[51] H. Zhao, X. Zeng, T. Chen, and Z. Liu, "Synthesizing barrier certificates using neural networks," in *Proc. 23rd Int. Conf. Hybrid Syst.: Comput. Control*, 2020, pp. 1–11.

[52] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 7139–7145.

[53] N. Gaby, F. Zhang, and X. Ye, "Lyapunov-Net: A deep neural network architecture for Lyapunov function approximation," in *Proc. IEEE Conf. Decis. Control*, to be published, *arXiv:2109.13359*.

[54] H. Yin, P. Seiler, and M. Arcak, "Stability analysis using quadratic constraints for systems with neural network controllers," *IEEE Trans. Autom. Control*, vol. 67, no. 4, pp. 1980–1987, Apr. 2022.

[55] H. Tsukamoto and S.-J. Chung, "Neural contraction metrics for robust estimation and control: A convex optimization approach," *IEEE Control Syst. Lett.*, vol. 5, no. 1, pp. 211–216, Jan. 2021.

[56] A. Robey et al., "Learning control barrier functions from expert demonstrations," in *Proc. IEEE Conf. Decis. Control*, 2020, pp. 3717–3724.

[57] L. Lindemann et al., "Learning hybrid control barrier functions from data," in *Proc. 4th Conf. Robot Learn.*, 2020, pp. 1351–1370.

[58] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning Lyapunov functions for hybrid systems," in *Proc. 24th Int. Conf. Hybrid Syst.: Comput. Control*, 2021, pp. 1–11.

[59] A. Robey, L. Lindemann, S. Tu, and N. Matni, "Learning robust hybrid control barrier functions for uncertain systems," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 1–6, 2021.

[60] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9562–9574.

[61] A. J. Taylor et al., "Learning for safety-critical control with control barrier functions," in *Proc. 2nd Conf. Learn. Dyn. Control*, 2020, vol. 120, pp. 1–10.

[62] F. Castañeda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and Dynamics," in *Proc. Amer. Control Conf.* 2021, pp. 3683–3690, doi: 10.23919/ACC50511.2021.9483420.

[63] J. Grizzle and J.-M. Kang, "Discrete-time control design with positive semi-definite Lyapunov functions," *Syst. Control Lett.*, vol. 43, no. 4, pp. 287–292, 2001.

[64] H. Dai, B. Landry, M. Pavone, and R. Tedrake, "Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems," in *Proc. 59th IEEE Conf. Decis. Control*, 2020, pp. 1274–1281.

[65] G. Margolis et al., "Learning to jump from pixels," in *Proc. Conf. Robot Learn.*, vol. 164, 2022, pp. 1025–1034. [Online]. Available: https://proceedings.mlr.press/v164/margolis22a.html
[66] Y. Meng, Z. Qin, and C. Fan, "Reactive and safe road user simulations using neural barrier certificates," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6299–6306.
[67] Z. Qin, D. Sun, and C. Fan, "SABLAS: Learning safe control for black-box dynamical systems," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1928–1935, Apr. 2022.
[68] C. Jewison and R. S. Erwin, "A spacecraft benchmark problem for hybrid control and estimation," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 3300–3305.
[69] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2018.
[70] D. V. Prokhorov, "Lyapunov machine for stability analysis of nonlinear systems," in *Proc. IEEE Int. Conf. Neural Netw.,*1994, vol. 2, pp. 1028–1031.
[71] G. Serpen, "Empirical approximation for Lyapunov functions with artificial neural nets," in *Proc. Int. Joint Conf. Neural Netw.*, 2005, vol. 2, pp. 735–740.
[72] N. Noroozi, P. Karimaghaee, F. Safaei, and H. Javadi, "Generation of Lyapunov functions by neural networks," in *Proc. World Congr. Eng.*, 2008, pp. 61–65.
[73] V. Petridis and S. Petridis, "Construction of neural network based Lyapunov functions," in *Proc. IEEE Int. Conf. Neural Netw.*, 2006, pp. 5059–5065.
[74] D. Ahmed, A. Peruffo, and A. Abate, "Automated and sound synthesis of Lyapunov functions with SMT solvers," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2020, vol. 12078, pp. 97–114.
[75] H. Tsukamoto, S. J. Chung, and J. J. E. Slotine, "Neural stochastic contraction metrics for learning-based control and estimation," *IEEE Control Syst. Lett.*, vol. 5, no. 5, pp. 1825–1830, Nov. 2021.
[76] S. Dean, A. J. Taylor, R. K. Cosner, B. Recht, and A. D. Ames, "Guaranteeing safety of learned perception modules via measurement-robust control barrier functions," in *Proc. Conf. Robot Learn.,*2020, pp. 654–670.
[77] C. Dawson, A. Garrett, F. Pollok, Y. Zhang, and C. Fan, "Barrier functions enable safety-conscious force-feedback control," 2022, *arXiv:2209.12270*.
[78] S. M. Katz, A. L. Corso, C. A. Strong, and M. J. Kochenderfer, "Verification of image-based neural network controllers using generative models," *J. Aerosp. Inf. Syst.*, vol. 19, no. 9, pp. 574–584, 2022.
[79] M. Tong, C. Dawson, and C. Fan, "Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields," 2022, *arXiv:2209.12266*.
[80] G. Shi et al., "Neural lander: Stable drone landing control using learned dynamics," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 9784–9790.
[81] A. J. Taylor, V. D. Dorobantu, S. Dean, B. Recht, Y. Yue, and A. D. Ames, "Towards robust data-driven control synthesis for nonlinear systems with actuation uncertainty," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 6469–6476.
[82] S. Liu, D. Liberzon, and V. Zharnitsky, "Almost Lyapunov functions for nonlinear systems," *Automatica*, vol. 113, 2020, Art. no. 108758.
[83] R. Bobiti and M. Lazar, "Automated-sampling-based stability verification and DOA estimation for nonlinear systems," *IEEE Trans. Autom. Control*, vol. 63, no. 11, pp. 3659–3674, Nov. 2018.
[84] M. Fazlyab, A. H. Robey Hassani, M. Morari, and G. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," in Proc. 33rd Int. Conf. Neural Inf. Process. Syst., 2019, pp. 11427–11438.
[85] S. Gao, S. Kong, and E. M. Clarke, "dReal: An SMT solver for nonlinear theories over the reals," in *Proc. 24th Int. Conf. Autom. Deduction*, 2013, pp. 208–214.
[86] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," in *Proc. Int. Conf. Learn. Represent.*, 2019.
[87] C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9354089/authors#authors
[88] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018. [Online]. Available: https://openreview.net/forum?id=rJzIBfZAb
[89] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.
[90] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. Cambridge, MA, USA:MIT Press, 2019.
[91] C. Fan, B. Qi, S. Mitra, and M. Viswanathan, "DRYVR: Data-driven verification and compositional reasoning for automotive systems," in *Computer Aided Verification*. Cham, Switzerland: Springer, 2017, pp. 441–461.
[92] R. Ivanov, K. Jothimurugan, S. Hsu, S. Vaidya, R. Alur, and O. Bastani, "Compositional learning and verification of neural network controllers," *ACM Trans. Embedded Comput. Syst.*, vol. 20, pp. 1–26, 2021.
[93] S. Shen and R. Tedrake, "Compositional verification of large-scale nonlinear systems via sums-of-squares optimization," in *Proc. Annu. Amer. Control Conf.*, 2018, pp. 4385–4392.
[94] Y.-S. Wang and N. Matni, "Localized distributed optimal control with output feedback and communication delays," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput.*, 2014, pp. 605–612.
[95] T. G. Molnar, A. K. Kiss, A. D. Ames, and G. Orosz, "Safety-critical control with input delay in dynamic environment," IEEE Trans. Control Syst. Technol., to be published, doi: 10.1109/TCST.2022.3227451.
[96] N. Matni, Y. P. Leong, Y. S. Wang, S. You, M. B. Horowitz, and J. C. Doyle, "Resilience in large scale distributed systems," *Procedia Comput. Sci.*, vol. 28, pp. 285–293, 2014.
[97] J. B. Bouvier, K. Xu, and M. Ornik, "Quantitative resilience of linear driftless systems," in *Proc. Conf. Control Appl.*, 2021, pp. 32–39.
[98] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[99] F. Yang and N. Matni, "Communication topology co-design in graph recurrent neural network based distributed control," in *Proc. 60th IEEE Conf. Decis. Control*, 2021, pp. 3619–3626.
[100] R. Y. Zhang and J. Lavaei, "Sparse semidefinite programs with near-linear time complexity," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 1624–1631.

Charles Dawson received the B.S. degree in engineering from Harvey Mudd College, Claremont, CA, USA, in 2019. He is currently working toward the Ph.D. degree with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA, supported by the National Science Foundation Graduate Research Fellowship.

He works on using tools from controls, learning, and optimization to understand safety in autonomous and cyberphysical systems.

Sicun Gao received the Ph.D. degree in logic from Carnegie Mellon University (CMU), Pittsburgh, PA, USA, in 2012.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA, USA. He works on computational methods and tools for improving automation and autonomous systems. He was a Postdoctoral Researcher with CMU and the Massachusetts Institute of Technology, Cambridge, MA, USA.

Dr. Gao is a recipient of the Air Force Young Investigator Award, the National Science Foundation CAREER Award, and a Silver Medal for the Kurt Godel Research Prize.

Chuchu Fan received the Ph.D. degree in computer engineering from the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2019.

She is currently an Assistant Professor with the Department of Aeronautics and Astronautics and Laboratory for Information and Decision Systems, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Before that, she was a Postdoctoral Researcher with the California Institute of Technology, Pasadena, CA, USA.Her group at MIT, REALM, works on using rigorous mathematics, including formal methods, machine learning, and control theory, to design, analyze, and verify safe autonomous systems.

Dr. Fan is the recipient of the AFSOR YIP Award, Innovator under 35 by MIT Technology Review, and the ACM Doctoral Dissertation Award.