Formally Verifying Deep Reinforcement Learning Controllers with Lyapunov Barrier Certificates

Udayan Mandal[1], Guy Amir[2], Haoze Wu[1], Ieva Daukantas[3], Fletcher Lee Newell[1], Umberto J. Ravaioli[4], Baoluo Meng[5], Michael Durling[5], Milan Ganai[1], Tobey Shim[1], Guy Katz[2], and Clark Barrett[1]
[1]Stanford University, Stanford, United States, {udayanm, haozewu, flnewell, mganai, tshim24, barrett}@stanford.edu
[2]The Hebrew University of Jerusalem, Jerusalem, Israel, {guyam, guykatz}@cs.huji.ac.il
[3]IT University of Copenhagen, Copenhagen, Denmark, daukantas@itu.dk
[4]Google, Mountain View, United States, uravaioli@google.com
[5]GE Aerospace Research, Niskayuna, United States, {baoluo.meng, durling}@ge.com

***Abstract*—Deep reinforcement learning (DRL) is a powerful machine learning paradigm for generating agents that control autonomous systems. However, the "black box" nature of DRL agents limits their deployment in real-world safety-critical applications. A promising approach for providing strong guarantees on an agent's behavior is to use *Neural Lyapunov Barrier* (NLB) certificates, which are learned functions over the system whose properties indirectly imply that an agent behaves as desired. However, NLB-based certificates are typically difficult to learn and even more difficult to verify, especially for complex systems. In this work, we present a novel method for training and verifying NLB-based certificates for discrete-time systems. Specifically, we introduce a technique for certificate *composition*, which simplifies the verification of highly-complex systems by strategically designing a sequence of certificates. When jointly verified with neural network verification engines, these certificates provide a formal guarantee that a DRL agent both achieves its goals and avoids unsafe behavior. Furthermore, we introduce a technique for certificate *filtering*, which significantly simplifies the process of producing formally verified certificates. We demonstrate the merits of our approach with a case study on providing safety and liveness guarantees for a DRL-controlled spacecraft.**

I. Introduction

In recent years, deep reinforcement learning (DRL) has achieved unprecedented results in multiple domains, including game playing, robotic control, protein folding, and many more [22], [49], [65], [72]. However, such models have an opaque decision-making process, making it highly challenging to determine whether a DRL-based system will *always* behave correctly. This is especially concerning for safety-critical domains (e.g., autonomous vehicles), in which even a single mistake can have dire consequences and risk human lives. This drawback limits the incorporation of DRL in real-world safety-critical systems.

The formal methods community has responded to this challenge by developing automated reasoning approaches for *proving* that a DRL-based controller behaves correctly [62]. These efforts rely in part on specialized DNN verification engines (a.k.a. *DNN verifiers*), which adapt techniques from other domains such as satisfiability modulo theories, abstract interpretation, mixed integer linear programming, and convex optimization [55], [56], [67], [87]. DNN verifiers take as input a DNN and a specification of the desired property and produce either a *proof* that the property always holds, or a *counterexample* demonstrating a case where the property does not hold. While the scalability of DNN verifiers has improved dramatically in the past decade [20], they struggle when applied to *reactive* (e.g., DRL-based) systems with temporal properties which require reasoning about interactions with the environment over time. This is because a naive approach for reasoning about time requires the involved DNN to be *unrolled* (i.e., a copy made for each time step), greatly increasing the complexity of the verification task.

On the other hand, for dynamical systems, a traditional approach for guaranteeing temporal properties has been to use control certificates such as Lyapunov Barrier functions [63]. Unfortunately, standard approaches for constructing these functions are not easily applicable to DRL-based dynamical systems. Recently, however, techniques have been developed for *learning* control certificates. We call these *Neural Lyapunov Barrier* (NLB) certificates [33]. Although NLB-based approaches have been shown to work for simple, toy examples, these certificates have been, thus far, difficult to learn and verify for real-world systems, which often involve large state spaces with complex dynamics.

In this work, we present a novel framework for training and formally verifying NLB-based certificates. Our framework can verify both *liveness* and *safety* properties of interest, providing *reach-while-avoid* (RWA) guarantees. We use off-the-shelf DNN verifiers and introduce a set of novel techniques to improve scalability, including certificate *filtering* and *composition*.

We demonstrate our approach with a case study targeting a specific challenge problem, in which the goal is to verify a DRL-based spacecraft controller [78]. We show that our framework is able to generate verified NLB-based RWA certificates for a range of complex properties. These include liveness properties (e.g., *will the spacecraft eventually reach its destination?*) and complex non-linear safety properties (e.g., *the spacecraft will never violate a non-linear velocity constraint*), both of which are challenging to verify using existing techniques.

https://doi.org/10.34727/2024/isbn.978-3-85448-065-5_15

The rest of this paper is organized as follows: Sec. II gives an overview of relevant background material on property types, DNN verifiers, and NLB certificates. Related work is covered in Sec. III. In Sec. IV and V, we present our approach, and Sec. VI reports the results of our spacecraft case study.[1] Finally, Sec. VII concludes.

Note. Proofs and additional details can be found in an extended technical report [69].

II. Preliminaries

A. Property Types

This work focuses on DRL controllers that are invoked over discrete time steps. We consider both safety and liveness properties [5].

Safety. A safety property indicates that *a bad state is never reached*. More formally, let $\mathcal{X}$ be the set of system states, and let $\tau \subseteq \mathcal{X}^*$ be the set of possible system trajectories. The system satisfies a *safety* property P if and only if every state in every trajectory satisfies P:

$$\forall\, \alpha : \alpha \in \tau : (\forall\, x \in \alpha :\ x \vDash P) \tag{1}$$

A violation of a safety property is a finite trajectory ending in a "bad" state (i.e., a state in which P does not hold).

Liveness. A liveness property concerns the *eventual* behavior of a system (e.g., *a good state is eventually reached*). More formally, we say a *liveness* property P holds if and only if there exists a state x in every infinite trajectory where P holds. Letting τ^∞ be the set of infinite trajectories, we can formalize this as follows.

$$\forall\, \alpha : \alpha \in \tau^\infty :\ (\exists\, x \in \alpha :\ x \vDash P), \tag{2}$$

A violation of a liveness property is an infinite trajectory in which each state violates the property P.

B. DNNs, DNN Verification, and Dynamical Systems.

Deep Learning. Deep neural networks (DNNs) [43] consist of layers of neurons, each layer performing a (typically non-linear) transformation of its input. This work focuses on deep reinforcement learning (DRL), a popular paradigm in which a DNN is trained to realize a *policy*, i.e., a mapping from states (the DNN's inputs) to actions (the DNN's outputs), which is used to control a reactive system. For more details on DRL, we refer to [64].

DNN Verification. Given (i) a trained DNN (e.g., a DRL agent) N; (ii) a precondition P on the DNN's inputs, limiting the input assignments; and (iii) a postcondition Q on the DNN's output, the goal of DNN verification is to determine whether the property $P(x) \rightarrow Q(N(x))$ holds for any neural network input x. In many DNN verifiers (a.k.a., *verification engines*), this task is equivalently reduced to determining the satisfiability of the formula $P(x) \wedge \neg Q(N(x))$. If the formula is satisfiable (SAT), then there is an input that satisfies the pre-condition and violates the post-condition, which means the property is violated. On the other hand, if the formula is unsatisfiable (UNSAT), then the property holds. It has been shown [55] that verification of piecewise-linear DNNs is NP-complete.

Discrete Time-Step Dynamical Systems. We focus on dynamical systems that operate in a discrete time-step setting. More formally, these are systems whose trajectories satisfy the equation:

$$x_{t+1} = f(x_t, u_t), \tag{3}$$

where f is a *transition function* that takes as inputs the current state $x_t \in \mathcal{X}$ and a control input $u_t \in \mathcal{U}$ and produces the next state x_{t+1}. These systems are controlled using a feedback control policy $\pi : \mathcal{X} \rightarrow \mathcal{U}$ which, given a state $x \in \mathcal{X}$ produces control input $u = \pi(x)$. In our setting, the controller π is realized by a DNN trained using DRL. DRL-based controllers are potentially useful in many real-world settings, due to their expressivity and their ability to generalize to complex environments [85].

C. Control Lyapunov Barrier Functions

The problem of verifying a liveness or safety property over a dynamical system with a given control policy can be reduced to the task of identifying a certificate function $V : \mathcal{X} \mapsto \mathbb{R}$, whose input-output relation satisfies a particular set of constraints that imply the property. There are two fundamental types of certificate functions.

Lyapunov Functions. A Lyapunov function (a.k.a. *Control Lyapunov function*) represents the energy level at the current state: as time progresses, energy is dissipated until the system reaches the zero-energy equilibrium point [48]. Hence, such functions are typically used to provide *asymptotic stability*, i.e., adherence to a desired liveness property, or the eventual convergence of the system to some goal state. Such guarantees can be afforded by learning a function that (i) reaches a 0 value at equilibrium, (ii) is strictly positive everywhere else; and (iii) either monotonically decreases [25], [26] or decreases by a particular constant [40] with each time step.

Barrier Functions. Barrier functions [8], a.k.a. *Control Barrier Functions*, are also energy-based certificates. However, these functions are typically used for verifying safety properties. Barrier functions enforce that a system will never enter an unsafe region in the state space. This is done by assigning unsafe states a function value above some threshold and then verifying that barrier function never crosses this threshold [7], [16], [90].

Control Lyapunov Barrier Functions. In many real-world settings, it can be useful to verify both liveness properties and safety properties. In such cases, a *Control Lyapunov Barrier Function* (CLBF) can be used, which combines the properties of both Control Barrier functions and Lyapunov functions. CLBFs can provide rigorous guarantees w.r.t. a wide variety of temporal properties, including the general setting of reach-while-avoid tasks [37], which we describe next.

[1] Code for reproducing the experiments is available at: github.com/NeuralNetworkVerification/artifact-fmcad24-docking.

Reach-while-Avoid Tasks. In a *reach-while-avoid* (RWA) task, we must find a controller π for a dynamical system such that all trajectories $\{x_1, x_2...\}$ produced by this controller (i) do not include unsafe ("bad") states; and (ii) eventually reach a goal state. More formally the problem can be defined as follows:

Definition 1 (Reach-while-Avoid Task)**.**
Input: *A dynamical system with a set of initial states $\mathcal{X}_I \subseteq \mathcal{X}$, a set of goal states $\mathcal{X}_G \subseteq \mathcal{X}$, and a set of unsafe states $\mathcal{X}_U \subseteq \mathcal{X}$, where $\mathcal{X}_I \cap \mathcal{X}_U = \varnothing$ and $\mathcal{X}_G \cap \mathcal{X}_U = \varnothing$*
Output: *A controller π such that for every trajectory $\tau = \{x_1, x_2...\}$ satisfying $x_1 \in \mathcal{X}_I$:*
1) ***Reach:*** $\exists\, t \in \mathbb{N}.\, x_t \in \mathcal{X}_G$
2) ***Avoid:*** $\forall\, t \in \mathbb{N}.\, x_t \notin \mathcal{X}_U$

III. Related Work

A. Control Certificates

Control certificate-based approaches form a popular and effective class of methods for providing guarantees about complex dynamical systems in diverse application areas including robotics [33], energy management [52], and biomedical systems [35]. Control Lyapunov functions are certificates for system stability, and the closely-related control Barrier functions are certificates for safety. While such Lyapunov-based certificates have been proposed over a century ago [66], their main drawback lies in their computational intractability [42]. As a result, practitioners have mainly relied on unscalable methods for constructing certificates, such as manual design for domain-dependent certificate functions [24], [27], sum-of-squares approaches restricted to polynomial systems [53], [68], and quadratic programming [63].

1) Formal Verification of Neural Certificates: Recent methods have leveraged neural networks as verifiable models of these control certificates, forming a class of *neural certificate* approaches [33]. For a fixed controller, [80] distills the problem into solving binary classification with neural networks, but the method is limited to polynomial systems and only obtains a region of attraction, making it incompatible with most RWA problems, which have a predefined goal region.

In [2], [4], SMT solvers are employed to check whether a certificate for a specific controller satisfies the Lyapunov conditions and, if not, to return counterexamples which can be used to retrain the neural certificate. A similar approach can be used for Barrier conditions [73]. In [1], [36], the Fossil tool is introduced, which combines these methods. In [3], Fossil is used to generate training examples for barrier certificates which are used to construct overapproximations of safe reach sets. However, these methods require verifying all constraints in the certificate for the entirety of the relevant state space — a task which can be computationally prohibitive (as we show in Section VI).

In [26], a Neural Lyapunov Control (NLC) framework is proposed, which jointly learns the Lyapunov certificate and the controller. The algorithm iteratively calls the dReal SMT solver [41] to generate counterexamples and retrain both the neural certificate and the control policy. Various extensions and applications followed: [45] addresses algorithmic problems in NLC; [46] automates the design of passive fault-tolerant control laws using NLC; [97] extends NLC to unknown nonlinear systems; [88] extends NLC to discrete-time systems; [82] verifies single hidden-layer ReLU neural certificates with enumeration [77] and linear programming; and [96] develops a framework for Barrier functions when there is an existing nominal controller. However, these methods do not consider the more general reach-avoid problem.

2) Data-driven Neural Certificates: To improve scalability, a recent line of research proposes learning certificates and controllers from online and/or offline data without additional formal verification [33], following the intuition that, with increasing data, the number of violations in the trained certificate will tend toward zero [19]. [25], [40] learn Lyapunov certificates for stabilization control, and [75], [76], [86], [95] synthesize neural Barrier functions in various settings like multi-agent control, neural radiance field [71] imagery, and pedestrian avoidance. These methods (by design) cannot provide rigorous guarantees on the validity of their learned certificates.

B. Reach-Avoid methods

Solutions for tasks requiring the simultaneous verification of both liveness and safety properties, of which the RWA task is a common example, have also relied on control theoretic principles. [34] learns a combined Lyapunov and Barrier certificate to construct controllers with stabilization and safety guarantees. The Hamilton-Jacobi (HJ) reachability-based method (a verification method for ensuring optimal control performance and safety in dynamical systems [15]) has also been used to solve reach-avoid problems [38], [51], [84]. Safe reinforcement learning is closely related to reach-avoid: the goal is to maximize cumulative rewards while minimizing costs along a trajectory [21], and it has been solved with both Lyapunov/Barrier methods [28], [91] and HJ reachability methods [39], [94]. As mentioned, scalability is a crucial challenge in this context. The next section describes our approach for addressing this challenge.

IV. Reach-While-Avoid Certificates

In this section, we present our approach for scalably creating verified NLB certificates. We first describe reach-while-avoid (RWA) certificates, a popular class of existing NLB-based certificates. We next present an extension called *Filtered* RWA certificates, which significantly simplifies the learning task and enables efficient training of certificates for complex properties. We then present a *compositional* certification approach, which independently trains a series of certificates that can be jointly verified to handle even larger state spaces.

A. RWA certificates

A function $V : \mathcal{X} \mapsto \mathbb{R}$ is an RWA certificate for the Reach-Avoid task in Definition 1 if, for some $\alpha > \beta$ and $\epsilon > 0$, it satisfies the following constraints.[2]

$$\forall\, x \in \mathcal{X}_I. \qquad V(x) \le \beta \tag{4}$$

$$\forall\, x \in \mathcal{X} \setminus \mathcal{X}_G. \quad V(x) \le \beta \rightarrow V(x) - V(f(x, \pi(x))) \ge \epsilon \tag{5}$$

$$\forall\, x \in \mathcal{X}_U. \qquad V(x) \ge \alpha \tag{6}$$

Any tuple of values $(\alpha, \beta, \epsilon)$ for which these conditions hold is called a *witness for* the certificate. RWA certificates provide the following guarantees.[3]

Lemma 1. *If V is an RWA certificate for a dynamical system with witness $(\alpha, \beta, \epsilon)$, and V has a lower bound,[4] then for every infinite trajectory τ starting from a state $x \in \mathcal{X} \setminus \mathcal{X}_G$ such that $V(x) \le \beta$, τ will eventually contain a state in $\mathcal{X}_G$ without ever passing through a state in $\mathcal{X}_U$.*

Intuitively, V partitions the state space into three regions:

- a *safe region* where the value of the certificate is at most β. This region includes the initial states $\mathcal{X}_I$ and any states reachable from $\mathcal{X}_I$. Furthermore, starting from any non-goal state in the safe region, the certificate function value should decrease by *at least* ϵ at each time step.
- an *unsafe region* where the value of the certificate is at least α. This region must include the unsafe states $\mathcal{X}_U$.
- an *intermediate region*, where the value of the certificate is strictly between β and α. States in this region are not unsafe but are also not reachable from $\mathcal{X}_I$. This can also be thought of as a "buffer" region that separates the safe region from the unsafe region. These states play a role in the compositional approach described below.

B. FRWA certificates

A *neural RWA* certificate is an RWA certificate realized by a DNN. Such a DNN can be trained by following the NLC approach [26], using the constraints (4)–(6) as training objectives. Because we are also interested in formally verifying these certificates, we would like to keep the DNNs (both the controller and the certificate) small so that verification remains tractable. We have observed that this can be challenging when the system and properties are non-trivial. To help address this, we introduce an improvement called *Filtered Reach-while-Avoid* (FRWA) certificates.

The idea behind FRWA is straightforward. Often, we can describe the goal and unsafe regions using simple predicates (or filters) on the state space. We pick constants c_1, c_2 such that $c_1 \le \beta < \alpha \le c_2$ and then hard-code the implementation of V so that $x \in \mathcal{X}_G \rightarrow V(x) = c_1$ and $x \in \mathcal{X}_U \rightarrow V(x) = c_2$. Note that the latter ensures that condition (6) holds by construction. Importantly, this not only makes the training task easier, but also reduces the number of queries required to formally verify the certificate. On the other hand, hard-coding the certificate value for inputs in $\mathcal{X}_G$ makes it easier to learn constraint (5). The reason for this is more nuanced. If we randomly initialize the certificate neural network, the certificate value for some states in $\mathcal{X}_G$ could start out larger than β, making it more difficult to satisfy constraint (5) for a point x where $V(x) \le \beta$ and $f(x, \pi(x)) \in \mathcal{X}_G$. Fixing the certificate values for states in $\mathcal{X}_G$ to at most β (ideally, significantly below β) ensures that, at least for such points, condition (5) is easier to satisfy. In practice, FRWA certificates can be implemented by using a wrapper around a DNN which checks the two filters and only calls the DNN if they both fail. The practical effectiveness of FRWA certificates is demonstrated in Sec. VI.

FRWA Training. FRWA simplifies the certificate learning process, as now, only constraints 4 and 5 are relevant for training. We custom design the reinforcement learning training objective function as follows. Let $x_1, \ldots, x_N$ be the set of training points, and let $x_i' = f(x_i, \pi(x_i))$. We define:

$$O_s = c_s \sum_{i \,|\, x_i \in \mathcal{X}_I} \frac{\mathrm{ReLU}(\delta_1 + V(x_i) - \beta)}{\sum_{i \,|\, x_i \in \mathcal{X}_I} 1} \tag{7}$$

$$O_d = c_d \sum_{i \,|\, x_i \in \mathcal{X} \setminus (\mathcal{X}_U \cup \mathcal{X}_G), V(x_i) \le \beta} \frac{\mathrm{ReLU}(\delta_2 + \epsilon + V(x_i') - V(x_i))}{\sum_{i \,|\, x_i \in \mathcal{X} \setminus (\mathcal{X}_U \cup \mathcal{X}_G), V(x_i) \le \beta} 1} \tag{8}$$

$$O = O_s + O_d \tag{9}$$

Eq. (7) penalizes deviations from constraint (4), and Eq. (8) penalizes deviations from constraint (5). We incorporate parameters $\delta_1 > 0$ and $\delta_2 > 0$, which can be used to tune how strongly the certificate over-approximates adherence to each constraint. Similarly, constants c_s and c_d can be used to tune the relative weight of the two objectives. The final training objective O in (9) is what the optimizer seeks to minimize, by using stochastic gradient descent (SGD) or other optimization techniques. We note that the FRWA certificates are trained in a self-supervised, non-RL setting.

FRWA Data Sampling. From the formulation above, we see that only data points in $(\mathcal{X} \setminus (\mathcal{X}_U \cup \mathcal{X}_G)) \cup \mathcal{X}_I$ affect the objectives, and thus, only these data points need to be sampled.

FRWA Verification. We use DNN verification tools to formally verify that conditions (4)-(6) hold for our certificates. Filtering introduces a slight complication. Recall that a FRWA certificate is implemented as a wrapper around a DNN, meaning that the DNN itself can behave arbitrarily when either $x \in \mathcal{X}_G$ or $x \in \mathcal{X}_U$. Fortunately, we can adjust the verification conditions for the DNN part of the certificate as follows.

Constraint (4) can be checked as is. The filtering does not affect this property. And it is easy to see that checking the property for the DNN does indeed ensure the property holds for the full certificate.

Constraint (6) need not be checked at all, as the filtered certificate ensures this condition by construction.

[2] These constraints are similar to the ones defined in prior work [37] but are specific to discrete time-step systems and instead place constraints on the set of unsafe states instead of a compact safe set.

[3] See [69] for a proof.

[4] This is always the case if the output of V is implemented using a finite representation such as floating-point arithmetic.

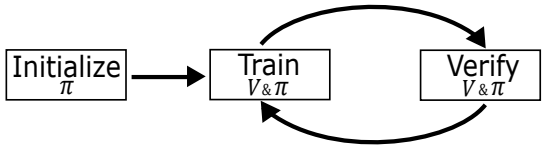

Fig. 1: The CEGIS Loop used to iteratively train and verify controller π and certificate V. Verification counterexamples are used to augment the training dataset.

Verification of constraint 5 is done by instead checking that:

$$\forall\, x \in \mathcal{X} \setminus (\mathcal{X}_U \cup \mathcal{X}_G), x' \in \mathcal{X}.$$
$$(x' = f(x, \pi(x)) \wedge V(x) \leq \beta) \rightarrow$$
$$(V(x) - V(x') \geq \epsilon \vee (x' \in \mathcal{X}_G)) \wedge (x' \notin \mathcal{X}_U) \qquad (10)$$

There are three main differences between (5) and (10). Since the filter ensures that $V(x) > \beta$ when $x \in \mathcal{X}_U$, we can safely exclude states in $\mathcal{X}_U$ from the check. Similarly, if the system ever transitions from a state x with $V(x) \leq \beta$ to an unsafe state, the filter ensures that condition (5) is violated, so it suffices to check that $x' \notin \mathcal{X}_U$ to cover this case.

The last difference is a bit more subtle. Observe that (10) is trivially true if $x' \in \mathcal{X}_G$, meaning that if we transition to a goal state, we do not enforce (5). However, it is easy to see that Lemma 1 still holds with this relaxed condition: if every transition either reduces V by at least ϵ or reaches a goal state, then clearly, we must eventually reach a goal state.

C. CEGIS loop

We use a *counterexample-guided inductive synthesis* (CEGIS) loop, shown in Fig. 1, to obtain a fully verified controller and certificate. We first train an initial controller π. Then, at each CEGIS iteration, we jointly train V and π until a loss of 0 is obtained and then use a sound and complete DNN verifier (we use *Marabou* [58] in our experiments) to identify counterexamples. If the verifier identifies a counterexample violating constraints (4) or (5) (recall that constraint (6) is satisfied by construction), we sample points in the proximity of the counterexample and use these to augment the training data. By sampling multiple nearby points, we hope to influence the training to learn smooth behavior for a localized neighborhood instead of overfitting to a specific point. This process is repeated iteratively until no counterexamples are found, at which point we are guaranteed to have produced a fully verified controller and certificate.

V. Compositional Certificates

While filtering does improve the efficiency of both training and verification, the approach outlined above still suffers from scalability challenges, especially as the system complexity or state space covered by the controller increases. In this section, we introduce *compositional certificates*, which aim to aid scalability by training multiple controller-certificate pairs, each covering different parts of the state space. The certificates are compositional in the sense that a simple meta-controller can be designed to determine which controller-certificate pair to use when in a given state, and we can formally guarantee that the meta-controller satisfies the requirements of definition 1.

CRWA. Formally, a compositional RWA certificate (CRWA) for an RWA task is composed of n RWA certificates,[5] which we denote $V_0, \ldots, V_{n-1}$, with corresponding controllers, which we denote $\pi_0, \ldots, \pi_{n-1}$, with $n \geq 2$. Furthermore, each pair (V_i, π_i) must be an RWA certificate with some witness $(\alpha_i, \beta_i, \epsilon_i)$ for an RWA task whose dynamics are that of the main RWA task, but whose parameters are $(\mathcal{X}_I^i, \mathcal{X}_G^i, \mathcal{X}_U^i)$. These parameters must satisfy the following conditions:

(*i*) $\mathcal{X}_I^0 \subseteq \mathcal{X}_I$, $\mathcal{X}_G^0 = \mathcal{X}_G$, and $\mathcal{X}_U \subseteq \mathcal{X}_U^0 \subseteq \overline{(\mathcal{X}_I^0 \cup \mathcal{X}_G^0)}$, where $\overline{S}$ denotes the complement of the set S;

(*ii*) for $0 < i < n$, $\mathcal{X}_I^{i-1} \subseteq \mathcal{X}_I^i \subseteq \mathcal{X}_I$, $\mathcal{X}_G^i = \{x \in \overline{\mathcal{X}_U^{i-1}} \mid V_{i-1}(x) \leq \beta_{i-1}\} \cup \mathcal{X}_G^0$, and $\mathcal{X}_U \subseteq \mathcal{X}_U^i \subseteq \mathcal{X}_U^{i-1}$;

(*iii*) either $\mathcal{X}_I^i \neq \mathcal{X}_I^{i-1}$ or $\mathcal{X}_U^i \neq \mathcal{X}_U^{i-1}$; and

(*iv*) $\mathcal{X}_I^{n-1} = \mathcal{X}_I$ and $\mathcal{X}_U^{n-1} = \mathcal{X}_U$.

Intuitively, the idea is as follows. We start with an initial controller capable of guiding the system from some *subset* of the initial states $\mathcal{X}_I$ to the original goal states $\mathcal{X}_G$ while avoiding some *superset* of the unsafe states $\mathcal{X}_U$. Then, for each subsequent controller, we ensure that it can guide the system either from a larger subset of the initial states $\mathcal{X}_I$ or while avoiding a smaller superset of the unsafe states $\mathcal{X}_U$, or both, to a new goal region consisting of the states considered safe by the previous controller, i.e., the states x for which $V(x) \leq \beta$. For the final controller (controller $n-1$), the set of initial and unsafe states should coincide with those of the original RWA problem. Note that the algorithm does not say how to choose of $\mathcal{X}_I^i$ and $\mathcal{X}_U^i$ for $i < n-1$ other than to specify that these sets should be monotonically increasing and decreasing, respectively. Finding good heuristics for choosing these sets in the general case is a promising direction for future work.

The meta-controller behaves as follows. Given any starting state $x \in \mathcal{X}_I$, we first check if $x \in \mathcal{X}_G$. If so, we are done. Otherwise, we determine the smallest i for which $x \in \mathcal{X}_I^i$ and guide the system using π_i until a state in $\mathcal{X}_G^i$ is reached, which will occur in some finite number of steps because of the guarantees provided by V_i. At this point, we transition to π_{i-1}, and the process repeats until a state in $\mathcal{X}_G$ is reached.

The training and verification of a CRWA certificate is described in Alg. 1 and visualized in Fig. 2.

The following lemma captures the correctness of our approach.[6]

Lemma 2. *Given a CRWA certificate for an RWA task with parameters $\mathcal{X}_I$, $\mathcal{X}_G$, and $\mathcal{X}_U$, all trajectories guided by the meta-controller starting at any point in $\mathcal{X}_I$ will reach $\mathcal{X}_G$ in a finite number of steps while avoiding $\mathcal{X}_U$. In other words, a CRWA certificate provides a correct solution for the RWA task.*

CRWA Data Sampling. When training certificate V_i, it is important that the training dataset contains sufficient states

[5] Each controller in a CRWA can make use of the FRWA technique described above.

[6] See [69] for a proof.

Algorithm 1: CRWA Training and Verification

Input : $\mathcal{X}_I$, $\mathcal{X}_G$, $\mathcal{X}_U$
Output: $\pi_0, \ldots, \pi_{n-1}, V_0, \ldots, V_{n-1}$ for some n

$\mathcal{X}_G^0 \leftarrow \mathcal{X}_G$
Choose $\mathcal{X}_I^0 \subseteq \mathcal{X}_I$ and $\mathcal{X}_U^0 \supseteq \mathcal{X}_U$, with $\mathcal{X}_U^0 \subseteq \overline{(\mathcal{X}_I^0 \cup \mathcal{X}_G^0)}$
Choose $\alpha^0 > \beta^0$ and $\epsilon^0 > 0$
Train and verify controller π_0 and certificate V_0 with witness $(\alpha^0, \beta^0, \epsilon^0)$ for the RWA task corresponding to $\mathcal{X}_I^0, \mathcal{X}_G^0$, and $\mathcal{X}_U^0$ using, e.g., the approach shown in Fig. 1
$i \leftarrow 0$
while $\mathcal{X}_I^i \subset \mathcal{X}_I$ *OR* $\mathcal{X}_U^i \supset \mathcal{X}_U$ **do**
  $i \leftarrow i + 1$
  Choose $\mathcal{X}_I^i, \mathcal{X}_U^i$ such that $\mathcal{X}_I^i \supset \mathcal{X}_I^{i-1}$ or $\mathcal{X}_U^i \subset \mathcal{X}_U^{i-1}$
  $\mathcal{X}_G^i \leftarrow \{x \in \overline{\mathcal{X}_U^{i-1}} \mid V_{i-1}(x) \leq \beta_{i-1}\} \cup \mathcal{X}_G^0$
  Choose $\alpha^i > \beta^i$ and $\epsilon^i > 0$
  Train and verify controller π_i and certificate V_i with witness $(\alpha^i, \beta^i, \epsilon^i)$ for the RWA task corresponding to $\mathcal{X}_I^i, \mathcal{X}_G^i$, and $\mathcal{X}_U^i$ using, e.g., the approach shown in Fig. 1

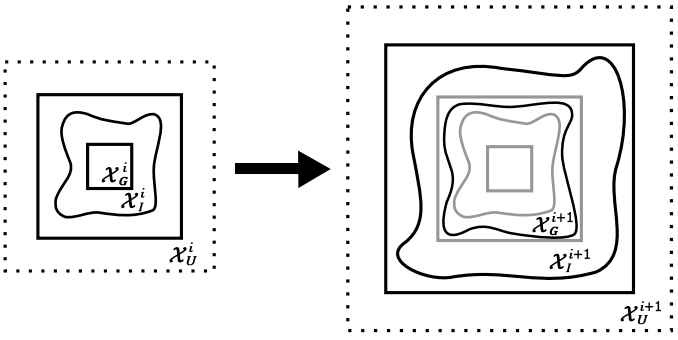

Fig. 2: Visualization of how consecutive certificates relate when building a CRWA certificate. Note that $\mathcal{X}_G$ need not be a subset of $\mathcal{X}_I$. The dotted lines indicate that the unsafe state region extends infinitely outside the solid line box. Wavy lines indicate outer boundaries for initial or goal regions.

sampled from $\mathcal{X}_I^i \setminus \mathcal{X}_G^i$. Otherwise, V_i might learn to assign values greater than β^i as much as possible in order to meet constraint (5), as opposed to appropriately assigning all states in $\mathcal{X}_I^i \setminus \mathcal{X}_G^i$ to have values less than β^i, due to an insufficient loss penalty for constraint (6). To ensure that states in the region $\mathcal{X}_I^i \setminus \mathcal{X}_G^i$ are included in the training data, we can identify states over constrained subspaces in $\mathcal{X}_I^i \setminus \mathcal{X}_G^i$, and then include in the data set those points as well as a random subset of their neighbors which likely lie in the same region.

Tradeoffs in choosing Intermediate Goals for CRWA certificates. It is possible to further reduce the state space for individual certificates in a CRWA certificate by using a more precise description of the goal states. In particular, we could set the goal states as follows:

$$\mathcal{X}_G^i = \{x \in \overline{\mathcal{X}_U^{i-1}} \mid V_{i-1}(x) \leq \beta_{i-1}\} \cup \mathcal{X}_G^{i-1}. \quad (11)$$

However, using 11 leads to a linear increase in the number of DNNs that must be included during training and verification at each iteration of Alg. 1. This quickly becomes prohibitively expensive, especially for the verification step. We thus use the simpler formulation described above.

VI. Evaluation

A. Case Study

We evaluate our approach on the 2D docking task from [78],[7] in which a spacecraft is trained using DRL to navigate to a goal. More specifically, a DRL agent maneuvers a *deputy* spacecraft, controlled with thrusters that provide forces in the x and y directions. The deputy spacecraft attempts to safely navigate until it reaches a state that is in close proximity to a designated *chief* spacecraft, while obeying a distance-dependent safety constraint. We focus on this benchmark for several reasons: (i) it has been proposed and studied as a challenge problem in the literature [78], (ii) there exist natural safety and liveness properties for it; and (iii) existing approaches have been been unable to formally verify these properties.

System Dynamics. The system is modeled using the Clohessy-Wiltshire relative orbital motion linear approximation in the non-inertial Hill's reference frame, with the *chief* spacecraft lying at the origin [29], [50]. The state of the system, $\boldsymbol{x} = [x, y, \dot{x}, \dot{y}]^T$, includes the position in (x, y) and the velocities in each direction, $(\dot{x}, \dot{y})$. The control input is $\boldsymbol{u} = [F_x, F_y]$, where F_x and F_y are the thrust forces applied along the x and y directions, respetively. Each thrust force component is allowed to range between -1 and $+1$ Newtons (enforced with standard piecewise linear clipping). As in the original scenario [78], the spacecraft's mass, m, is 12kg. The continuous time state dynamics of the system are determined by the following ordinary differential equations (ODE), with $n = 0.001027$ rad/s:

$$\dot{\boldsymbol{x}} = [\dot{x}, \dot{y}, \ddot{x}, \ddot{y}]^T \quad (12)$$

$$\ddot{x} = 2n\dot{y} + 3n^2 x + \frac{F_x}{m} \quad (13)$$

$$\ddot{y} = -2n\dot{x} + \frac{F_y}{m} \quad (14)$$

This, in turn, is converted to a discrete system (with a time-step of T) by numerically integrating the continuous time dynamics ODE:

$$\boldsymbol{x}(t_i + T) = \boldsymbol{x}(t_i) + \int_{t_i}^{t_i+T} \dot{\boldsymbol{x}}(\tau) d\tau \quad (15)$$

The discrete-time version has a closed-form solution that we use to generate successive states for the spacecraft.

Constraints and Terminal Conditions. To maintain safety, a distance-dependent constraint is imposed on the *deputy*

[7] Additional details on this case study are described in our recent related paper [70] and in the extended version of the current paper [69].

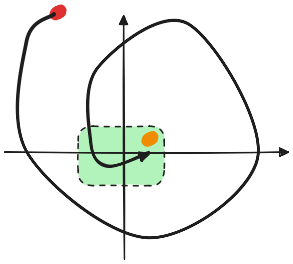

Fig. 3: A spiral trajectory: The DRL-controlled spacecraft (starting from the red point) eventually reaches the destination (orange) point within the docking region.

spacecraft's maximal velocity magnitude (while approaching the *chief*):

$$\sqrt{\dot{x}^2 + \dot{y}^2} \leq 0.2 + 2n\sqrt{x^2 + y^2} \tag{16}$$

We construct a linear over-approximation of this safety constraints (with OVERT [83]). It can then be incorporated into the description of the unsafe region.

The goal is for the deputy to successfully dock with the chief without ever violating the velocity safety constraint. In the original benchmark [78], the docking (goal) region is defined as a circle of diameter $d = 1$m centered at the origin $(0, 0)$. In our evaluation, we use this same goal region for the initial training of the DRL controller. However, during the CEGIS iteration (i.e., the "Train" and "Verify" steps in Fig. 1), we use a conservative subset of this goal region, namely a square centered at the origin whose sides have length $l = 0.7$m. The reason for this is so that the goal region can easily be described using linear inequalities.

B. DNN architecture

We explored various architectures for the DNN used to control the (deputy) spacecraft. During this exploration phase, we trained each DNN architecture using the original Proximal Policy Optimization RL algorithm implemented by Ray RLlib as described in [78] (without any CEGIS iteration).

After training, we simulated each architecture on 4,000 random trajectories. Some selected results are shown in Table I. There are two main observations to take away from these results: (i) while a robust docking capability can be achieved fairly easily, even for small architectures, safety is more difficult and appears to not be robust, even for large architectures; (ii) in all cases, it takes an average of at least 50 steps to dock. The first observation suggests that verification of the liveness (docking) property should be feasible and that training a controller that verifiably achieves both safety and liveness is challenging. The second observation suggests that even state-of-the-art DNN verifiers are unlikely to be unable to fully verify the liveness property using the naive unrolling approach [9].

We also note that the spacecraft often exhibits highly non-linear spiral trajectories (as depicted in Fig. 3), making DNN verification based on induction difficult, as it is difficult to find an inductive property over such irregular trajectories. These results help motivate the use of NLB certificates for formal verification of the desired properties. For the experiments below, we settled on a DNN architecture of two hidden layers with 20 neurons each, and a certificate architecture of two hidden layers with 30 neurons each. Both DNNs use ReLU activations for all hidden layers. The DNN sizes were chosen based on experimentation and the rough criterion that we wanted the smallest DNNs for which the CEGIS loop would converge in a reasonable amount of time.

TABLE I: Performance of various DNN architectures. Statistics are collected (per architecture) over $4,000$ trials, with a maximum trajectory length of $2,000$, initialized arbitrarily to set $x, y \in [-10, 10]$, but outside the docking region, and $\dot{x} = \dot{y} = 0$. The first column indicates the number of neurons per hidden layer.

DNN Architecture	Safety Success	Docking Success	Average Docking Steps
[4,4]	100	10	1,821
[8,8]	11	100	389
[16,16]	30	100	50
[32,32]	5	100	59
[64,64]	100	100	55
[64,64,64,64]	100	99	58
[200,200]	92	100	51

C. Implementation and Setup

The training and verification of the DRL controllers and certificates were carried out on a cluster of Intel Xeon E5-2637 machines, with eight cores of v4 CPUs, running the Ubuntu 20.04 operating system. Verification queries were dispatched using the *Marabou* DNN verifier [58], [87] (used in previous DNN safety research [10]–[14], [17], [23], [30], [61], [79]) as well as its Gurobi back end.

For training and verification of RWA, FRWA, and CRWA certificates, we use the following parameters: $\alpha = 1 + 10^{-5}$, $\beta = 1$, $\epsilon = 10^{-7}$ (the same for all certificates); $c_1 = -10$, $c_2 = 1.2$, $\delta_1 = 10^{-4} - 10^{-5}$, and $\delta_2 = 10^{-4} - 10^{-7}$. These values were determined to work well experimentally.

For weighting of the training objectives, we use $c_s = 1$ and $c_d = 10$. The rationale for this is that constraint (4) is much easier to satisfy than (5), so we use the weights to force the training to focus on (5).

In the CEGIS loop, a learning rate of 5×10^{-3} is used to train the first network iteration in the CEGIS loop, and for retraining, a learning rate of 10^{-4} is used, since we treat the incorporation of counterexamples as a "fine-tuning" step and do not want to overfit to the counterexamples. In the CEGIS loop, we train until a loss of 0 is achieved and then use the verification step to find counterexamples. We repeat this until there are no more counterexamples or a timeout (12 hours) is reached.

All of our experiments aim to solve RWA tasks, as defined in Definition 1. The system dynamics are those of the 2D spacecraft, as described in Section VI-A. RWA tasks are

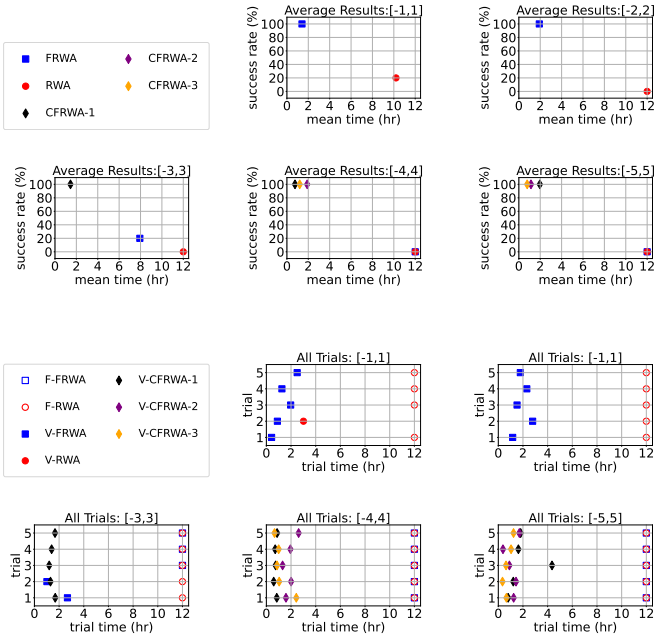

Fig. 4: The first 2 rows show average times and success rates for creating verified certificates over 5 trials. The bottom 2 rows show specific times for each trial, separated into failed ("F-") and verified ("V-") certificates. CFRWA-1, CFRWA-2, and CFRWA-3 refer to the first (up to) three CRWA tasks for the given starting region, corresponding, respectively to the first (up to) three rows for that starting region in Table II.

parameterized by $\mathcal{X}_G$, $\mathcal{X}_I$, and $\mathcal{X}_U$. For these sets of states, we typically use square regions centered at the origin. For convenience, we refer to the set $\{(x, y) \mid x, y \in [-a, a]\}$ with the abbreviation $[-a, a]$. For example, as outlined in Section VI-A, we set $\mathcal{X}_G = [-0.35, 0.35]$. We use different values for $\mathcal{X}_I$, depending on the experiment (indeed, this is the primary variable we vary in our experiments), but whenever $\mathcal{X}_I = [-a, a]$, we then set $\overline{\mathcal{X}_U} = [-(a+1), a+1]$.

D. Experimental Results

RWA vs. FRWA. In our first set of experiments, we select a set of RWA tasks and train both RWA and FRWA certificates using our CEGIS loop.[8] We select five RWA tasks, where $\mathcal{X}_I$ is set to $[-i, i]$ for the ith task. For the RWA certificates, we follow the approach of [37], whereas our FRWA certificates are constructed as described in Sec. IV. In each case, we run five independent trials for each task.

The results are summarized in Fig. 4. The first two rows show, for each starting region, the number of successful runs (a run is successful if the CEGIS loop produces a fully verified controller/certificate pair within the 12 hour time limit) and the average time required for the successful runs. Results for RWA are shown as red circles and FRWA as blue squares (we explain the diamonds later). For example, for starting region $[-2, 2]$, all five trials are successful for FWRA, with an average time of 2 hours, whereas all five trials are unsuccessful for RWA. The bottom two rows show data from the same experiments, but here we show the time taken for each of the five trials. An unfilled circle or box represents a timeout.

The results suggest that FWRA has a clear advantage over standard RWA. In fact, RWA only succeeded once in producing any verified certificate, and only for the simplest starting region. On the other hand, our FRWA approach is able to produce certificates faster and for starting regions up to $[-3, 3]$. After that, both techniques time out.

Compositional Certificates. As demonstrated above, RWA and FRWA certificates quickly run into scalability challenges on our case study problem. For example, even with 5 tries and a 12 hour timeout, neither approach could produce a verified controller for the $[-4, 4]$ or $[-5, 5]$ starting regions.

Our second set of experiments demonstrates that this scalability challenge can be addressed with compositional certificates. We train a set of compositional certificates (each composed of multiple FRWA certificates) and report the results in Table II.

Each row of the table corresponds to a compositional certificate. The first column shows the value of $\mathcal{X}_I$ for this certificate. The next columns indicate the number n of composed certificates, the values of $\mathcal{X}_I^i$ for $0 \leq i < n-1$, and the cumulative time required for all but the last certificate. The next three columns give the minimum, mean, and maximum time required to produce the controller and certificate for the last stage of the compositional certificate (recall that we run five independent trials for all CEGIS loops). The next three columns show the minimum, mean, and maximum number of CEGIS iterations used, and the last column indicates how many of the trials succeeded. Note that when $n = 1$, the row corresponds to a single FRWA certificate.

The results clearly indicate that compositional certificates greatly improve scalability. Whereas the stand-alone certificates could not scale beyond $[-3, 3]$ in 12 hours, we were able to successfully produce a formally verified 5-stage certificate for $[-11, 11]$ in a little over 5.7 hours. It is also worth noting that we do get a significant benefit by running 5 independent CEGIS loops, as both the time and the number of loops can vary significantly from the minimum to the maximum. Nearly all of the CEGIS loops eventually completed—only the initial $[3, 3]$ region failed to complete all of its trials—suggesting that the compositional approach is also more stable and robust. This can also be seen in Fig. 4: for each starting region $[-a, a]$, the diamond point labeled CFRWA-i corresponds to the ith row containing $[-a, a]$ in column 1. We can see that, compared to the stand-alone RWA and FRWA certificates, the compositional certificates can be trained faster and with fewer failures.

[8]The Fossil 2.0 tool provides an implementation for computing the RWA certificates used in [37]. However, our definition of RWA is slightly different, and we use a different DNN verification tool, so we compare with our own implementation of RWA certificates to have a more meaningful comparison and to better isolate the contribution of the filtering technique.

TABLE II: Compositional certificate results. The columns indicate: the initial set for the final certificate, the size of the compositional certificate, the initial sets for all but the final certificate, the cumulative time for all but the final certificate, the total time (min, mean, and max) for training all certificates, and statistics for training the final certificate. We note that the cumulative time column is always equal to the corresponding value in the min column corresponding to the penultimate certificate. The wall time is the total time including the final controller/certificate. The CEGIS iterations and success stats are for the final controller/certificate only.

	Compositional Certificate			Wall Time (s)			CEGIS Iterations			
$\mathcal{X}_\mathbf{I}$	$\mathbf{n}$	$\mathcal{X}_\mathbf{I}^\mathbf{0} \dots \mathcal{X}_\mathbf{I}^\mathbf{n-2}$	**Cumulative Time (s)**	**min(t)**	**mean(t)**	**max(t)**	**min(i)**	**mean(i)**	**max(i)**	**success (%)**
[-2,2]	1	N/A	0	4199	6890	8322	3	4.8	10	100
[-3,3]	1	N/A	0	3650	6644.5	9639	2	2.5	3	40
[-3,3]	2	[-2,2]	4199	8514	9421	10271	4	4.4	5	100
[-4,4]	2	[-3,3]	3650	5802	6374	6790	2	2.4	3	100
[-4,4]	2	[-2,2]	4199	8940	11026	13620	3	4.2	6	100
[-4,4]	3	[-2,2], [-3,3]	8514	10901	12829	17248	2	4.2	8	100
[-5,5]	2	[-3,3]	3650	6526	10716	19331	2	4.4	9	100
[-5,5]	3	[-3,3], [-4,4]	5802	7171	9884	11945	1	3.4	5	100
[-5,5]	4	[-2,2],[-3,3],[-4,4]	10901	12130	13710	15353	1	2.4	4	100
[-6,6]	3	[-2,2],[-4,4]	8940	13183	16384	20059	4	4.4	5	100
[-6,6]	4	[-3,3],[-4,4],[-5,5]	7171	9680	14027	32103	2	4.6	9	100
[-6,6]	5	[-2,2],[-3,3],[-4,4],[-5,5]	12130	18607	21768	24356	3	4.4	5	100
[-7,7]	3	[-3,3],[-5,5]	6526	9158	10171	10848	2	2.8	3	100
[-7,7]	5	[-3,3],[-4,4],[-5,5],[-6,6]	9680	11878	15967	23419	2	3.6	7	100
[-8,8]	4	[-2,2],[-4,4],[-6,6]	13183	16677	22623	33849	2	3.2	4	100
[-9,9]	4	[-3,3],[-5,5],[-7,7]	9158	12919	16013	18507	2	3.4	5	100
[-10,10]	5	[-2,2],[-4,4],[-6,6],[-8,8]	16677	18137	23421	30872	1	3.4	6	100
[-11,11]	5	[-3,3],[-5,5],[-7,7],[-9,9]	12919	20641	27860	32834	1	2.6	5	100

VII. Conclusion

In this work, we present a novel framework for formally verifying DRL-based controllers. Our approach leverages Neural Lyapunov Barrier certificates and demonstrates how they can be used to verify DNN-based controllers for complex systems. We use a CEGIS loop for training and formally verifying certificates, and we introduce filters for reach-while-avoid certificates, which simplify the training and verification process. We also introduce compositional certificates which use a sequence of simpler certificates to scale to large state spaces.

We demonstrate the merits of our approach on a 2D case study involving a DRL-controlled spacecraft which is required to dock in a predefined region, from any initialization point. We demonstrate that for small subdomains, our FRWA approach is strictly better than competing RWA-based certificate methods. Furthermore, we demonstrate that our compositional approach unlocks significant additional scalability.

In the future, we plan to extend our approach to be compatible with additional formal techniques (e.g., shielding against safety violations [6], [18], [31], [60], [74], [81], [89], and Scenario-Based Programming [32], [44], [47], [54], [57], [59], [92], [93]). We also plan to apply our approach to more challenging case studies with larger DRL controllers. We see this work as an important step towards the safe and reliable use of DRL in real-world systems.

VIII. Acknowledgements

This work was supported by AFOSR (FA9550-22-1-0227), the Stanford CURIS program, the NSF-BSF program (NSF: 1814369, BSF: 2017662), and the Stanford Center for AI Safety. The work of Amir was further supported by a scholarship from the Clore Israel Foundation. We thank Kerianne Hobbs (AFRL), Thomas Henzinger (ISTA), Chuchu Fan (MIT), and Songyuan Zhang (MIT) for useful conversations and advice which contributed to the success of this project.

REFERENCES

[1] 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 *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[2] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo. Formal synthesis of lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

[3] A. Abate, S. Bogomolov, A. Edwards, K. Potomkin, S. Soudjani, and P. Zuliani. Safe reach set computation via neural barrier certificates. *arXiv preprint arXiv:2404.18813*, 2024.

[4] D. Ahmed, A. Peruffo, and A. Abate. Automated and sound synthesis of lyapunov functions with smt solvers. In *Tools and Algorithms for the Construction and Analysis of Systems: 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25–30, 2020, Proceedings, Part I 26*, pages 97–114. Springer, 2020.

[5] B. Alpern and F. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2:117–126, 09 1987.

[6] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe Reinforcement Learning via Shielding. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*, pages 2669–2678, 2018.

[7] A. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *Trans. on Automatic Control*, 2017.

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *European Control Conf.*, 2019.

[9] G. Amir, D. Corsi, R. Yerushalmi, L. Marzari, D. Harel, A. Farinelli, and G. Katz. Verifying Learning-Based Robotic Navigation Systems. In *Proc. 29th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 607–627, 2023.

[10] G. Amir, Z. Freund, G. Katz, E. Mandelbaum, and I. Refaeli. veriFIRE: Verifying an Industrial, Learning-Based Wildfire Detection System. In *Proc. 25th Int. Symposium on Formal Methods (FM)*, pages 648–656, 2023.

[11] G. Amir, O. Maayan, T. Zelazny, G. Katz, and M. Schapira. Verifying Generalization in Deep Learning. In *Proc. 35th Int. Conf. on Computer Aided Verification (CAV)*, pages 438–455, 2023.

[12] G. Amir, M. Schapira, and G. Katz. Towards Scalable Verification of Deep Reinforcement Learning. In *Proc. 21st Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 193–203, 2021.

[13] G. Amir, H. Wu, C. Barrett, and G. Katz. An SMT-Based Approach for Verifying Binarized Neural Networks. In *Proc. 27th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 203–222, 2021.

[14] G. Amir, T. Zelazny, G. Katz, and M. Schapira. Verification-Aided Deep Ensemble Selection. In *Proc. 22nd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 27–37, 2022.

[15] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *Conf. on Decision and Control*, 2017.

[16] G. Basile and G. Marro. Controlled and conditioned invariant subspaces in linear system theory. *Journal of Optimization Theory and Applications*, 3:306–315, 1969.

[17] S. Bassan, G. Amir, D. Corsi, I. Refaeli, and G. Katz. Formally Explaining Neural Networks within Reactive Systems. In *Proc. 23rd Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 10–22, 2023.

[18] R. Bloem, B. Könighofer, R. Könighofer, and C. Wang. Shield Synthesis: - Runtime Enforcement for Reactive Systems. In *Proc. of the 21st Int. Conf. in Tools and Algorithms for the Construction and Analysis of Systems, (TACAS)*, volume 9035, pages 533–548, 2015.

[19] N. Boffi, S. Tu, N. Matni, J.-J. Slotine, and V. Sindhwani. Learning stability certificates from data. In *Conference on Robot Learning*, pages 1341–1350. PMLR, 2021.

[20] C. Brix, S. Bak, C. Liu, and T. T. Johnson. The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results. *arXiv preprint arXiv:2312.16760*, 2023.

[21] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

[22] Y. Cao, H. Zhao, Y. Cheng, T. Shu, G. Liu, G. Liang, J. Zhao, and Y. Li. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods, 2024.

[23] M. Casadio, E. Komendantskaya, M. Daggitt, W. Kokke, G. Katz, G. Amir, and I. Refaeli. Neural Network Robustness as a Verification Property: A Principled Case Study. In *Proc. 34th Int. Conf. on Computer Aided Verification (CAV)*, pages 219–231, 2022.

[24] 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. *arXiv*, Nov 2020.

[25] Y.-C. Chang and S. Gao. Stabilizing neural control using self-learned almost lyapunov critics. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1803–1809, 2021.

[26] Y.-C. Chang, N. Roohi, and S. Gao. Neural lyapunov control. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[27] 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 *Robotics: Science and Systems*. Robotics: Science and Systems, Apr 2020.

[28] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8092–8101. Curran Associates, Inc., 2018.

[29] W. Clohessy and R. Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the aerospace sciences*, 27(9):653–658, 1960.

[30] D. Corsi, G. Amir, G. Katz, and A. Farinelli. Analyzing Adversarial Inputs in Deep Reinforcement Learning, 2024. Technical Report. https://arxiv.org/abs/2402.05284.

[31] D. Corsi, G. Amir, A. Rodriguez, C. Sanchez, G. Katz, and R. Fox. Verification-Guided Shielding for Deep Reinforcement Learning, 2024. Technical Report. http://arxiv.org/abs/2406.06507.

[32] D. Corsi, R. Yerushalmi, G. Amir, A. Farinelli, D. Harel, and G. Katz. Constrained Reinforcement Learning for Robotics via Scenario-Based Programming, 2022. Technical Report. https://arxiv.org/abs/2206.09603.

[33] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

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

[35] J. L. C. B. de Farias and W. M. Bessa. Intelligent control with artificial neural networks for automated insulin delivery systems. *Bioengineering*, 9(11):664, 2022.

[36] A. Edwards, A. Peruffo, and A. Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. *arXiv preprint arXiv:2311.09793*, 2023.

[37] A. Edwards, A. Peruffo, and A. Abate. A general verification framework for dynamical and control models via certificate synthesis, 2023.

[38] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Proceedings of the 18th international conference on hybrid systems: computation and control*, pages 11–20, 2015.

[39] M. Ganai, Z. Gong, C. Yu, S. L. Herbert, and S. Gao. Iterative reachability estimation for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[40] M. Ganai, C. Hirayama, Y.-C. Chang, and S. Gao. Learning stabilization control from observations by learning lyapunov-like proxy models. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2913–2920, 2023.

[41] S. Gao, S. Kong, and E. M. Clarke. dreal: An smt solver for nonlinear theories over the reals. In *International conference on automated deduction*, pages 208–214. Springer, 2013.

[42] P. Giesl and S. Hafstein. Review on computational methods for lyapunov functions. *Discrete and Continuous Dynamical Systems-B*, 20(8):2291–2331, 2015.

[43] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[44] M. Gordon, A. Marron, and O. Meerbaum-Salant. Spaghetti for the Main Course? Observations on the Naturalness of Scenario-Based Programming. In *Proc. 17th ACM Annual Conf. on Innovation and Technology in Computer Science Education (ITCSE)*, pages 198–203, 2012.

[45] D. Grande, E. Anderlini, A. Peruffo, and G. Salavasidis. Augmented neural lyapunov control. *IEEE Access*, 2023.

[46] D. Grande, D. Fenucci, A. Peruffo, E. Anderlini, A. B. Phillips, G. Thomas, and G. Salavasidis. Systematic synthesis of passive fault-tolerant augmented neural lyapunov control laws for nonlinear systems. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 5851–5856. IEEE, 2023.

[47] J. Greenyer, D. Gritzner, G. Katz, and A. Marron. Scenario-Based Modeling and Synthesis for Reactive Systems with Dynamic System Structure in ScenarioTools. In *Proc. 19th ACM/IEEE Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*, pages 16–23, 2016.

[48] W. Haddad and V. Chellaboina. Nonlinear dynamical systems and control: A lyapunov-based approach. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*, 01 2008.

[49] T. Hester, M. Quinlan, and P. Stone. A real-time model-based reinforcement learning architecture for robot control, 2011.

[50] G. W. Hill. Researches in the lunar theory. *American journal of Mathematics*, 1(1):5–26, 1878.

[51] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. In *Proceedings of Robotics: Science and Systems*, Virtual, 7 2021.

[52] T. Huang, S. Gao, and L. Xie. A neural lyapunov approach to transient stability assessment of power electronics-interfaced networked microgrids. *IEEE transactions on smart grid*, 13(1):106–118, 2021.

[53] Z. Jarvis-Wloszek, R. Feeley, Weehong Tan, Kunpeng Sun, and A. Packard. Some controls applications of sum of squares programming. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 5, pages 4676–4681 Vol.5, Dec 2003.

[54] G. Katz. Guarded Deep Learning using Scenario-Based Modeling. In *Proc. 8th Int. Conf. on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 126–136, 2020.

[55] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Proc. 29th Int. Conf. on Computer Aided Verification (CAV)*, pages 97–117, 2017.

[56] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. Reluplex: a Calculus for Reasoning about Deep Neural Networks. *Formal Methods in System Design (FMSD)*, 2021.

[57] G. Katz and A. Elyasaf. Towards Combining Deep Learning, Verification, and Scenario-Based Programming. In *Proc. 1st Workshop on Verification of Autonomous and Robotic Systems (VARS)*, pages 1–3, 2021.

[58] G. Katz, D. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. Dill, M. Kochenderfer, and C. Barrett. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *Proc. 31st Int. Conf. on Computer Aided Verification (CAV)*, pages 443–452, 2019.

[59] G. Katz, A. Marron, A. Sadon, and G. Weiss. On-the-Fly Construction of Composite Events in Scenario-Based Modeling Using Constraint Solvers. In *Proc. 7th Int. Conf. on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 143–156, 2019.

[60] B. Könighofer, F. Lorber, N. Jansen, and R. Bloem. Shield Synthesis for Reinforcement Learning. In *Proc. Int. Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, pages 290–306, 2020.

[61] O. Lahav and G. Katz. Pruning and Slicing Neural Networks using Formal Verification. In *Proc. 21st Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 183–192, 2021.

[62] M. Landers and A. Doryab. Deep reinforcement learning verification: A survey. *ACM Comput. Surv.*, 55(14s), jul 2023.

[63] B. Li, S. Wen, Z. Yan, G. Wen, and T. Huang. A survey on the control lyapunov function and control barrier function for nonlinear-affine control systems. *IEEE/CAA Journal of Automatica Sinica*, 10(3):584–602, 2023.

[64] Y. Li. Deep Reinforcement Learning: An Overview, 2017. Technical Report. http://arxiv.org/abs/1701.07274.

[65] J. Lu. Protein folding structure prediction using reinforcement learning with application to both 2d and 3d environments. In *Proceedings of the 5th International Conference on Computer Science and Software Engineering*, CSSE '22, page 534–542, New York, NY, USA, 2022. Association for Computing Machinery.

[66] A. M. Lyapunov. The general problem of motion stability. *Annals of Mathematics Studies*, 17(1892), 1892.

[67] Z. Lyu, C. Y. Ko, Z. Kong, N. Wong, D. Lin, and L. Daniel. Fastened Crown: Tightened Neural Network Robustness Certificates. In *Proc. 34th AAAI Conf. on Artificial Intelligence (AAAI)*, pages 5037–5044, 2020.

[68] A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[69] U. Mandal, G. Amir, H. Wu, I. Daukantas, F. Newell, U. Ravaioli, B. Meng, M. Durling, M. Ganai, T. Shim, G. Katz, and C. Barrett. Formally Verifying Deep Reinforcement Learning Controllers with Lyapunov Barrier Certificates, 2024. Technical Report. https://arxiv.org/abs/2405.14058.

[70] U. Mandal, G. Amir, H. Wu, I. Daukantas, F. Newell, U. Ravaioli, B. Meng, M. Durling, K. Hobbs, M. Ganai, T. Shim, G. Katz, and C. Barrett. Safe and Reliable Training of Learning-Based Aerospace Controllers, 2024. Technical Report. http://arxiv.org/abs/2407.07088.

[71] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[72] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013.

[73] A. Peruffo, D. Ahmed, and A. Abate. Automated and formal synthesis of neural barrier certificates for dynamical models. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 370–388. Springer, 2021.

[74] S. Pranger, B. Könighofer, M. Tappler, M. Deixelberger, N. Jansen, and R. Bloem. Adaptive Shielding under Uncertainty. In *American Control Conference, (ACC)*, pages 3467–3474, 2021.

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

[76] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan. Learning safe multi-agent control with decentralized neural barrier certificates. In *ICLR*, 2021.

[77] M. Rada and M. Cerny. A new algorithm for enumeration of cells of hyperplane arrangements and a comparison with avis and fukuda's reverse search. *SIAM Journal on Discrete Mathematics*, 32(1):455–473, 2018.

[78] U. J. Ravaioli, J. Cunningham, J. McCarroll, V. Gangal, K. Dunlap, and K. L. Hobbs. Safe reinforcement learning benchmark environments for aerospace control systems. In *2022 IEEE Aerospace Conference (AERO)*, pages 1–20. IEEE, 2022.

[79] I. Refaeli and G. Katz. Minimal Multi-Layer Modifications of Deep Neural Networks, 2021. Technical Report. https://arxiv.org/abs/2110.09929.

[80] S. M. Richards, F. Berkenkamp, and A. Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 466–476, 29–31 Oct 2018.

[81] A. Rodriguez, G. Amir, D. Corsi, C. Sanchez, and G. Katz. Shield Synthesis for LTL Modulo Theories, 2024. Technical Report. http://arxiv.org/abs/2406.04184.

[82] P. Samanipour and H. A. Poonawala. Stability analysis and controller synthesis using single-hidden-layer relu neural networks. *IEEE Transactions on Automatic Control*, 2023.

[83] C. Sidrane, A. Maleki, A. Irfan, and M. J. Kochenderfer. Overt: An algorithm for safety verification of neural network control policies for nonlinear systems. *Journal of Machine Learning Research*, 23(117):1–45, 2022.

[84] O. So and C. Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. In *Proceedings of Robotics: Science and Systems*, 2023.

[85] V. Talpaert, I. Sobh, B. R. Kiran, P. Mannion, S. Yogamani, A. El-Sallab, and P. Perez. Exploring applications of deep reinforcement learning for real-world autonomous driving systems, 2019.

[86] M. Tong, C. Dawson, and C. Fan. Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10511–10517. IEEE, 2023.

[87] H. Wu, O. Isac, A. Zeljić, T. Tagomori, M. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, et al. Marabou 2.0: A versatile formal analyzer of neural networks. *arXiv preprint arXiv:2401.14461*, 2024.

[88] J. Wu, A. Clark, Y. Kantaros, and Y. Vorobeychik. Neural lyapunov control for discrete-time systems. *Advances in Neural Information Processing Systems*, 36:2939–2955, 2023.

[89] M. Wu, J. Wang, J. Deshmukh, and C. Wang. Shield Synthesis for Real: Enforcing Safety in Cyber-Physical Systems. In *Proc. 19th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD)*, pages 129–137, 2019.

[90] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames. Robustness of control barrier functions for safety critical control. *Int. Federation of Automatic Control*, 2015.

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

[92] R. Yerushalmi, G. Amir, A. Elyasaf, D. Harel, G. Katz, and A. Marron. Scenario-Assisted Deep Reinforcement Learning. In *Proc. 10th Int. Conf. on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 310–319, 2022.

[93] R. Yerushalmi, G. Amir, A. Elyasaf, D. Harel, G. Katz, and A. Marron. Enhancing Deep Reinforcement Learning with Scenario-Based Modeling. *SN Computer Science*, 4(2):156, 2023.

[94] D. Yu, H. Ma, S. Li, and J. Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning*, pages 25636–25655. PMLR, 2022.

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

[96] H. Zhang, J. Wu, Y. Vorobeychik, and A. Clark. Exact verification of relu neural control barrier functions. *Advances in Neural Information Processing Systems*, 36, 2024.

[97] R. Zhou, T. Quartz, H. De Sterck, and J. Liu. Neural lyapunov control of unknown nonlinear systems with stability guarantees. *Advances in Neural Information Processing Systems*, 35:29113–29125, 2022.