Neural Control and Certificate Repair via Runtime Monitoring

Emily Yu[1], **Đorđe Žikelić**[2], **Thomas A. Henzinger**[1]

[1]Institute of Science and Technology Austria, Klosterneuburg, Austria
[2]Singapore Management University, Singapore, Singapore
emily.yu@ist.ac.at, dzikelic@smu.edu.sg, tah@ist.ac.at

Abstract

Learning-based methods provide a promising approach to solving highly non-linear control tasks that are often challenging for classical control methods. To ensure the satisfaction of a safety property, learning-based methods jointly learn a control policy together with a certificate function for the property. Popular examples include barrier functions for safety and Lyapunov functions for asymptotic stability. While there has been significant progress on learning-based control with certificate functions in the white-box setting, where the correctness of the certificate function can be formally verified, there has been little work on ensuring their reliability in the black-box setting where the system dynamics are unknown. In this work, we consider the problems of certifying and repairing neural network control policies and certificate functions in the black-box setting. We propose a novel framework that utilizes runtime monitoring to detect system behaviors that violate the property of interest under some initially trained neural network policy and certificate. These violating behaviors are used to extract new training data, that is used to re-train the neural network policy and the certificate function and to ultimately repair them. We demonstrate the effectiveness of our approach empirically by using it to repair and to boost the safety rate of neural network policies learned by a state-of-the-art method for learning-based control on two autonomous system control tasks.

Introduction

The rapid advance of machine learning has sparked interest in using it to solve hard problems across various application domains, and autonomous robotic systems control is no exception. Learning-based control methods obtain data through repeated interaction with an unknown environment, which is then used to learn a control policy. A popular example is (deep) reinforcement learning algorithms, which learn neural network policies (Mnih et al. 2015; Sutton and Barto 2018). However, the complex and often uninterpretable nature of learned policies poses a significant barrier to their safe and trustworthy deployment in safety-critical applications such as self-driving cars or healthcare devices (Amodei et al. 2016; García and Fernández 2015).

In control theory, a classical method to certify the correctness of a control policy concerning some property of interest is to compute a certificate function for that property (Dawson, Gao, and Fan 2023). A *certificate function* is a mathematical object which proves that the system under the control policy indeed satisfies the property. Common examples of certificate functions include Lyapunov functions for stability (Khalil 2002) and barrier functions for safety set invariance (Prajna and Jadbabaie 2004). While certificate functions are a classical and well-established concept in control theory, earlier methods for their computation are based either on polynomial optimization or rely on manually provided certificates, which makes them restricted to polynomial systems and typically scalable to low-dimensional environments (Ahmadi and Majumdar 2016; Dawson, Gao, and Fan 2023; Srinivasan et al. 2021).

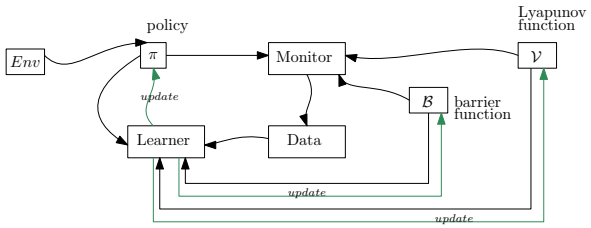

Figure 1: The monitor-learner framework.

Recent years have seen significant progress in overcoming these limitations, by utilizing machine learning to compute certificates for *non-polynomial environments* with *neural network control policies* (Abate et al. 2021b; Chang, Roohi, and Gao 2019; Dawson et al. 2021; Qin, Sun, and Fan 2022; Qin et al. 2021; Richards, Berkenkamp, and Krause 2018). The key idea behind these methods is to *jointly learn* a policy together with a certificate function, with both parametrized as neural networks. This is achieved by defining a loss function that encodes all defining properties of the certificate function. The learning algorithm then incurs loss whenever some of the defining certificate conditions are violated at any system state explored by the learning algorithm. Hence, the training process effectively guides the learning algorithm towards a control policy that admits a correct certificate function. A survey of the existing approaches and recent advances can be found in (Dawson, Gao, and Fan 2023).

While the above approach *guides* the learning algorithm towards a correct certificate, it does not guarantee that the

output control policy or the certificate function is correct. To address this issue, a number of recent works have considered the *white-box setting* which assumes complete knowledge of the environment dynamics and formally verifies the correctness of the certificate function. Notable examples include methods for learning and verifying neural Lyapunov functions (Abate et al. 2021b; Chang, Roohi, and Gao 2019; Richards, Berkenkamp, and Krause 2018) and neural barrier functions (Abate et al. 2021a; Zhang et al. 2023; Zhao et al. 2020), also extended to stochastic settings (Abate et al. 2023; Ansaripour et al. 2023; Chatterjee et al. 2023; Lechner et al. 2022; Mathiesen, Calvert, and Laurenti 2023; Mazouz et al. 2022; Zhi et al. 2024; Zikelic et al. 2023a,b). While there are several methods that learn control policies together with certificates in the black-box setting, e.g. (Dawson et al. 2021; Qin, Sun, and Fan 2022; Qin et al. 2021), to the best of our knowledge no prior method considers the problem of analyzing and certifying their correctness in the *black-box setting*, without assuming any knowledge on the dynamics. In this setting, one cannot guarantee the correctness of certificate functions at every system state as we do not have access to the system model at each state. However, we may still use more lightweight techniques to evaluate the control policy and the certificate and to repair them in cases when they are found to be incorrect.

Our contributions. In this work, we propose a method for analyzing and repairing neural network policies and certificates in the black-box setting. Our method uses *runtime monitoring* to detect system behaviors that violate the property of interest or the certificate defining conditions. These behaviors are used to extract additional training data, which is then used by the training algorithm to re-train and ultimately *repair* the neural policy and the certificate. Our method consists of two modules – called the *monitor* and the *learner*, which are composed in a loop as shown in Fig. 1. The loop is executed until the monitor can no longer find property or certificate condition violating behaviors, upon which our algorithm concludes that the neural policy and the certificate have been repaired.

While the idea of using runtime monitors to identify incorrect behaviors is natural, it introduces several subtle challenges whose overcoming turns out to be highly non-trivial. The first and obvious approach to runtime monitoring of a control policy in isolation is to simply test it, flag runs that violate the property, and add visited states to the training set. However, such a simple monitor suffers from two significant limitations. First, it can only be used to monitor violations of properties that can be observed from finite trajectories such as safety, but not of properties that are defined by the limiting behavior such as stability. Second, the simple monitor detects property violations only *after* they happen, meaning that it cannot identify states and actions that do not yet violate the property but inevitably lead to property violation. In contrast, in practice we are often able to predict in advance when some unsafe scenario may occur. For instance, if we drive a car in the direction of a wall at a high speed, then even before hitting the wall we can predict that a safety violation would occur unless we change the course of action.

To overcome these limitations, we advocate the runtime monitoring of neural network policies *together* with certificate functions and propose two monitoring algorithms. The first algorithm, called *Certificate Policy Monitor (CertPM)*, issues a warning whenever either the property or some of the defining conditions of the certificate are violated. The second algorithm, called *Predictive Policy Monitor (PredPM)*, goes a step further by estimating the remaining time until the property or some certificate condition may be violated, and issues a warning if this time is below some tolerable threshold. Experimental results demonstrate the ability of CertPM and PredPM to detect the property or certificate condition violating behaviors. Furthermore, we show that CertPM and PredPM allow our method to *repair* and significantly improve neural policies and certificates computed by a state-of-the-art learning-based control theory method.

Our contributions can be summarized as follows:

1. *Runtime monitoring of policies and certificates.* We design two novel algorithms for the runtime monitoring of control policies and certificates in the *black-box setting*.
2. *Monitoring-based policy and certificate repair.* By extracting additional training data from warnings issued by our monitors, we design a novel method for *automated repair* of neural network control policies and certificates.
3. *Empirical evaluation.* Our prototype successfully repairs neural network control policies and certificates computed by a state-of-the-art learning-based control method.

Related work. Related works on learning-based control with certificate functions, as well as on formal verification of learned control policies and certificates in the white-box setting, have been discussed above, so we omit repetition. In what follows, we discuss some other related lines of work.

Constrained reinforcement learning (RL) algorithms consider constrained Markov decision processes (CMDPs) (Altman 1999; Geibel 2006) and are also concerned with learning policies under safety constraints. Notable examples of algorithms for solving CMDPs include Constrained Policy Optimization (CPO) (Achiam et al. 2017) or the method of (Chow et al. 2018) which uses a Lyapunov based approach. While these algorithms perform well, they are empirical in nature and do not provide any mechanism for checking correctness of learned policies.

Shielding (Bloem et al. 2015; Alshiekh et al. 2018) is a runtime enforcement approach (Könighofer et al. 2020; Pranger et al. 2021; Carr et al. 2023). They use a monitor to detect if the system is close to reaching an unsafe set of states and thus violating the safety property. In such cases, the system switches to a safe *back-up policy*. However, there are two fundamental differences between our method and shielding. First, we do not assume a backup policy. Second, our method can also be used to *repair* a control policy, in cases when it is determined to be incorrect. Apart from shielding, runtime monitoring has been extensively used in the field of runtime verification as a more lightweight alternative to formal verification or when errors are only triggered at runtime (Falcone and Pinisetty 2019; Renard et al. 2019; Zhou et al. 2020). Runtime monitoring and repair of neural network policies, but without using certificates, was considered in (Bauer-Marquart et al. 2022; Lyu et al. 2023).

Preliminaries

We consider a (deterministic, continuous-time) dynamical system $\Sigma = (X, U, f)$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is the state space, $\mathcal{U} \subseteq \mathbb{R}^m$ is the control action space and $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the system dynamics, assumed to be Lipschitz continuous. The dynamics of the system are defined by $\dot{x}(t) = f(x(t), u(t))$, where $t \in \mathbb{R}_{\geq 0}$ is time, $x : \mathbb{R}_{\geq 0} \to \mathcal{X}$ is the state trajectory, and $u : \mathbb{R}_{\geq 0} \to \mathcal{U}$ is the control input trajectory. The control input trajectory is determined by a control policy $\pi : \mathcal{X} \to \mathcal{U}$. We use $\mathcal{X}_0 \subseteq \mathcal{X}$ to denote the set of initial states of the system. For each initial state $x_0 \in \mathcal{X}_0$, the dynamical system under policy π gives rise to a unique trajectory $x(t)$ with $x(0) = x_0$. Control tasks are concerned with computing a control policy $\pi : \mathcal{X} \to \mathcal{U}$ such that, under the control input $u(t) = \pi(x(t))$, the state trajectory induced by $\dot{x}(t) = f(x(t), \pi(x(t))$ satisfies a desired property for every initial state in $\mathcal{X}_0$. In this work, we consider two of the most common families of control properties:

1. *Safety.* Given a set of unsafe states $\mathcal{X}_u \subseteq \mathcal{X}$, the dynamical system satisfies the *safety* property under policy π with respect to $\mathcal{X}_u$, if it never reaches $\mathcal{X}_u$, i.e., if for each initial state in $\mathcal{X}_0$ we have $x(t) \notin \mathcal{X}_u$ for all $t \geq 0$.
2. *Stability.* Given a set of goal states $\mathcal{X}_g \subseteq \mathcal{X}$, the dynamical system satisfies the *(asymptotic) stability* property under policy π with respect to $\mathcal{X}_g$, if it asympotically converges to $\mathcal{X}_g$, i.e., if $\lim_{t\to\infty} \inf_{x_g \in \mathcal{X}_g} ||x(t) - x_g|| = 0$ for each initial state in $\mathcal{X}_0$.

We also consider the stability-while-avoid property, which is a logical conjunction of stability and safety properties. To ensure that the dynamical system satisfies the property of interest, classical control methods compute a *certificate function* for the property. In this work, we consider barrier functions (Ames, Grizzle, and Tabuada 2014) for proving safety and Lyapunov functions (Khalil 2002) for proving stability. The stability-while-avoid property is then proved by computing both certificate functions.

Proposition 1 (Barrier functions) *Suppose that there exists a continuously differentiable function $\mathcal{B} : \mathcal{X} \to \mathbb{R}$ for the dynamical system Σ under a policy π with respect to the unsafe set $\mathcal{X}_u$, that satisfies the following conditions:*

1. Initial condition. $\mathcal{B}(x) \geq 0$ *for all* $x \in \mathcal{X}_0$.
2. Safety condition. $\mathcal{B}(x) < 0$ *for all* $x \in \mathcal{X}_u$.
3. Non-decreasing condition. $L_f\mathcal{B}(x) + \mathcal{B}(x) \geq 0$ *for all* $x \in \{x \mid \mathcal{B}(x) \geq 0\}$, *where* $L_f\mathcal{B} = \frac{\partial \mathcal{B}}{\partial x} f(x, u)$ *is the Lie derivative of* $\mathcal{B}$ *with respect to* f.

Then, Σ satisfies the safety property under π with respect to $\mathcal{X}_u$, and we call $\mathcal{B}$ a barrier function.

Proposition 2 (Lyapunov functions) *Suppose that there exists a continuously differentiable function $\mathcal{V} : \mathcal{X} \to \mathbb{R}$ for the dynamical system Σ under policy π with respect to the goal set $\mathcal{X}_g$, that satisfies the following conditions:*

1. Zero upon goal. $\mathcal{V}(x_g) = 0$ *for all* $x_g \in \mathcal{X}_g$.
2. Strict positivity away from goal. $\mathcal{V}(x) > 0$ *for all* $x \in \mathcal{X} \backslash \mathcal{X}_g$.
3. Decreasing condition. $L_f\mathcal{V} < 0$ *for all* $x \in \mathcal{X} \backslash \mathcal{X}_g$.

Then, Σ satisfies the stability property under π with respect to $\mathcal{X}_g$, and we call $\mathcal{V}$ a Lyapunov function.

Assumptions. We consider the black-box setting, meaning that the state space $\mathcal{X}$, the control action space $\mathcal{U}$ as well as the goal set $\mathcal{X}_g$ and/or the unsafe set $\mathcal{X}_u$ (depending on the property of interest) are *known*. However, the system dynamics function f is *unknown* and we only assume access to an engine which allows us to execute the dynamics function.

Problem statement. Consider a dynamical system Σ defined as above. Suppose we are given one of the above properties, specified by the unsafe set $\mathcal{X}_u$ and/or the goal set $\mathcal{X}_g$. Moreover, suppose that we are given a control policy π and a certificate function $\mathcal{B}$ for the property.

1. *Policy certification and repair.* Determine whether the system Σ under policy π satisfies the property. If not, repair the policy π such that the property is satisfied.
2. *Certificate certification and repair.* Determine whether $\mathcal{B}$ is a good certificate function which shows that the dynamical system Σ under policy π satisfies the property. If not, repair the certificate function $\mathcal{B}$ such that it becomes a correct certificate function.

Runtime Monitoring Policies and Certificates

We now present our algorithms for runtime monitoring of a policy by monitoring it together with a certificate function. These algorithms present the first step in our solution to the two problems defined above. The second step, namely policy and certificate repair, will follow in the next section. The runtime monitoring algorithms apply to general policies and certificate functions, not necessarily being neural networks.

Motivation for certificate monitoring. Given a dynamical system $\Sigma = (\mathcal{X}, \mathcal{U}, f)$ and a property of interest, a *monitor* is a function $\mathcal{M} : \mathcal{X}^+ \to \{0, 1\}$ that maps each finite sequence of system states to a verdict on whether the sequence violates the property. This means that the monitor can only detect violations with respect to properties whose violations can be observed from finite state trajectories. This includes safety properties where violations can be observed upon reaching the unsafe set, but *not* infinite-time horizon properties like stability which requires the state trajectory to asymptotically converge to the goal set *in the limit*. In contrast, monitoring both the control policy and the certificate function allows the monitor to issue a verdict on either (1) property violation, or (2) certificate violation, i.e. violation of one of the defining conditions in Proposition 1 for barrier functions or Proposition 2 for Lyapunov functions. By Propositions 1 and 2, we know that the barrier function or the Lyapunov function being correct provides a formal proof of the safety or the stability property. Hence, in order to show that there are no property violations and that the property of interest is satisfied, it suffices to show that there are no certificate violations which can be achieved by monitoring both the control policy and the certificate function.

Certificate Policy Monitor

We call our first monitor the *certificate policy monitor (CertPM)*. For each finite sequence of observed states $x_0, x_1, \ldots, x_n$, the monitor $\mathcal{M}_{\mathsf{CertPM}}$ issues a verdict on whether a property or a certificate violation has been observed. If we are considering a safety property, the property

violation verdict is issued whenever $x_n \in \mathcal{X}_u$, and the certificate violation verdict is issued whenever one of the three defining conditions in Proposition 1 is violated at x_n. If we are considering a stability property, the property violation verdict cannot be issued, however, the certificate violation verdict is issued whenever one of the defining conditions in Proposition 2 is violated at x_n. In the interest of space, in what follows we define the monitor $\mathcal{M}_{\mathsf{CertPM}}$ for a safety property specified by the unsafe set of states $\mathcal{X}_u$. The definition of $\mathcal{M}_{\mathsf{CertPM}}$ for a stability property is similar, see the extened versio. Let π be a policy and $\mathcal{B}$ be a barrier function:

- The safety violation verdict is issued if $x_n \in \mathcal{X}_u$. We set $\mathcal{M}_{\mathsf{CertPM}}(x_0, x_1, \ldots, x_n) = 1$.
- The certificate violation verdict for the Initial condition in Proposition 1 is issued if $x_n \in \mathcal{X}_0$ is an initial state but $\mathcal{B}(x_n) < 0$. We set $\mathcal{M}_{\mathsf{CertPM}}(x_0, x_1, \ldots, x_n) = 1$.
- The certificate violation verdict for the Safety condition in Proposition 1 is issued if $\mathcal{B}(x_n) < 0$. We set $\mathcal{M}_{\mathsf{CertPM}}(x_0, x_1, \ldots, x_n) = 1$.
- Checking the Non-decreasing condition in Proposition 1 is more challenging since the dynamical system evolves over continuous-time and the non-decreasing condition involves a Lie derivative. To address this challenge, we approximate the non-decreasing condition by considering the subsequent observed state x_{n+1} and approximating the Lie derivative via
$$\widehat{L_f\mathcal{B}}(x_n) = \frac{\mathcal{B}(x_{n+1}) - \mathcal{B}(x_n)}{t_{n+1} - t_n}.$$
This requires the monitor to wait for at least one new observation before issuing the verdict. The approximation satisfies $|\widehat{L_f\mathcal{B}}(x) - L_f\mathcal{B}(x)| \leq \epsilon$ for all $x \in \mathcal{X}$ where $\epsilon = \frac{1}{2}\Delta_t(\mathcal{C}_{\mathcal{B}}\mathcal{L}_f + \mathcal{C}_f\mathcal{L}_{\mathcal{B}})\mathcal{C}_f$, with $\mathcal{L}_{\mathcal{B}}$ and $\mathcal{L}_f$ being the Lipschitz constants of $\mathcal{B}$ and f bounded by constants $\mathcal{C}_{\mathcal{B}}, \mathcal{C}_f \in \mathbb{R}_{>0}$ (Nejati et al. 2023). This suggests we can achieve good precision by using sufficiently small time intervals for monitoring. The certificate violation verdict for the Non-decreasing condition in Proposition 1 is issued if $\mathcal{B}(x_n) \geq 0$ but $\widehat{L_f\mathcal{B}}(x_n) + \mathcal{B}(x_n) < 0$. We set $\mathcal{M}_{\mathsf{CertPM}}(x_0, x_1, \ldots, x_n) = 1$.
- Otherwise, no violation verdict is issued. We set $\mathcal{M}_{\mathsf{CertPM}}(x_0, x_1, \ldots, x_n) = 0$.

Predictive Policy Monitor

CertPM checks if the property or one of the certificate defining conditions is violated at the states observed so far. Our second monitor, which we call the *predictive policy monitor (PredPM)*, considers the case of safety properties and *estimates the remaining time* before the property or the certificate violation may occur in the future. PredPM then issues a verdict based on whether any of the estimated remaining times are below some pre-defined thresholds. Thus, our second monitor aims to predict future behavior and issue property and certificate violation verdicts *before they happen*.

Since PredPM is restricted to safety properties, let $\mathcal{X}_u$ be the set of unsafe states. Let π be a policy and $\mathcal{B}$ be a barrier function. For each finite sequence of observed states $x_0, x_1, \ldots, x_n$, PredPM computes three quantitative assessments $[v_U, v_S, v_N]$, where:

1. v_U is an estimate of the time until $\mathcal{X}_u$ is reached and hence the safety property is violated (or, if $x_n \in \mathcal{X}_u$ already, v_U is an estimate of the time until the safe part $\mathcal{X}\backslash\mathcal{X}_u$ is reached).
2. v_S is an estimate of the time until the set $\{x \in \mathcal{X} \mid \mathcal{B}(x) < 0\}$ may be reached and hence the Safety condition in Proposition 1 may be violated;
3. v_N is an estimate of the time until the Non-dec. condition in Proposition 1 may be violated.

Note that these three values are estimates on the remaining time until some set $S \subseteq \mathcal{X}$ is reached, where $S = \mathcal{X}_u$ (or $S = \mathcal{X}\backslash\mathcal{X}_u$ if $x_n \in \mathcal{X}_u$) for computing v_U, $S = \{x \in \mathcal{X} \mid \mathcal{B}(x) < 0\}$ for computing v_S, and $S = \{x \in \mathcal{X} \mid \widehat{L_f\mathcal{B}}(x) + \mathcal{B}(x) < 0\}$ for computing v_N. PredPM computes each of the three values by solving the following problem:

$$\min_a \mathcal{T} \text{ s.t. } \frac{dx}{dt} = v(t),\ \frac{dv}{dt} = a(t),\ |a(t)| \leq a_{max},$$
$$\forall t \in [0, \mathcal{T}];\ x(\mathcal{T}) \in S,\ x(0) = x_n,\ v(0) = v_0.$$

Here, $v(t)$ is the velocity and $a(t)$ is the acceleration of the trajectory $x(t)$, with a_{max} being the maximum allowed acceleration. Intuitively, solving this optimization problem results in the acceleration that the controller should use at each time such that the set S is reached in the shortest time possible. To make the computation physically more realistic, we assume a physical bound a_{max} on the maximum acceleration that the controller can achieve at any time. To compute an approximate solution to this problem and hence estimate v_U, v_S and v_N, PredPM discretizes the time $[0, \mathcal{T}]$ and for each discrete time point it uses stochastic gradient descent to select the next acceleration within the range $[0, a_{max}]$.

Once PredPM computes v_U, v_S, v_N, it checks if any of the values exceed the thresholds ξ_U, ξ_S, ξ_N that are assumed to be provided by the user. The monitor issues the verdict $\mathcal{M}_{\mathsf{PredPM}}(x_0, x_1, \ldots, x_n) = 1$ if either $v_U > \xi_U$ or $v_S > \xi_S$ or $v_N > \xi_N$. Otherwise, it issues the verdict 0. Having quantitative assessments allows the user to specify how fault-tolerant the monitor should be. For highly safety-critical systems, one can set the thresholds to be positive such that the monitor signals a warning *in advance*, i.e., when it predicts a dangerous situation and before it happens.

Neural Policy and Certificate Repair

Our monitors in the previous section are able to flag finite state trajectories that may lead to property or certificate violations. We now show how the verdicts issued by our monitors can be used to extract additional training data that describe the property or certificate violations. Hence, if the control policy and the certificate are both learned as neural networks, we show how they can be retrained on this new data and *repaired*. In the interest of space, we consider the case of safety properties where certificates are barrier functions. Suppose that $\mathcal{X}_u$ is a set of unsafe states in a dynamical system $\Sigma = (\mathcal{X}, \mathcal{U}, f)$. The pseudocode of our monitoring-based repair algorithm is shown in Algorithm 1. Our method can be easily modified to allow Lyapunov function repair and we provide this extension in the extended version (Yu, Zikelic, and Henzinger 2024).

Algorithm 1: Policy Repair. Monitoring-based neural network policy and certificate repair for safety properties.

Input: policy π, barrier function $\mathcal{B}$, number D, time points $t_0 = 0 < t_1 < \cdots < t_N$, (optional) thresholds ξ_U, ξ_S ξ_N for PredPM

1: $\mathcal{M} \leftarrow$ BUILDMONITOR($\mathcal{X}$) ▷ initializing a monitor, either CertPM or PredPM
2: $D_{\text{New-data}} \leftarrow \emptyset$ ▷ new training data collected
3: $\tilde{\mathcal{X}}_0 \leftarrow D$ states randomly sampled from $\mathcal{X}_0$
4: **for** $x_0 \in \tilde{\mathcal{X}}_0$ **do**
5: $x(t) \leftarrow$ state trajectory from $x(0) = x_0$
6: **for** $n \in \{0, 1, \ldots, N\}$ **do**
7: $x_n \leftarrow$ observed state at time point t_n
8: **if** $\mathcal{M}(x_0, x_1, \ldots, x_n) = 1$ **then**
9: $D_{\text{New-data}} \leftarrow D_{\text{New-data}} \cup \{x_n\}$
10: **end if**
11: **end for**
12: **end for**
13: $D^{\text{repair}}_{\text{Init}}, D^{\text{repair}}_{\text{Safe}}, D^{\text{repair}}_{\text{Non-dec}} \leftarrow D_{\text{New-data}} \cap \mathcal{X}_0, D_{\text{New-data}} \cap \mathcal{X}_u, D_{\text{New-data}} \cap \{x \mid \mathcal{B}(x) \geq 0\}$
14: **if** policy certification and repair **then**
15: $\pi, \mathcal{B} \leftarrow$ repair with loss function in eq. (1) and training data $D^{\text{repair}}_{\text{Init}}$, $D^{\text{repair}}_{\text{Safe}}$, $D^{\text{repair}}_{\text{Non-dec}}$
16: **else if** certificate certification and repair **then**
17: $\mathcal{B} \leftarrow$ repair with loss function in eq. (1) and training data $D^{\text{repair}}_{\text{Init}}$, $D^{\text{repair}}_{\text{Safe}}$, $D^{\text{repair}}_{\text{Non-dec}}$
18: **end if**

Learning-based control with certificates. Before showing how to extract new training data and use it for policy and certificate repair, we first provide an overview of the general framework for jointly learning neural network policies and barrier functions employed by existing learning-based control methods (Zhao et al. 2020). Two neural networks are learned simultaneously, by minimizing a loss function which captures each of the defining conditions of barrier functions in Proposition 1. That way, the learning process is guided towards learning a neural control policy that admits a barrier function and hence satisfies the safety property. The loss function contains one loss term for each defining condition:

$$\mathcal{L}(\theta, \nu) = \mathcal{L}_{\text{Init}}(\theta, \nu) + \mathcal{L}_{\text{Safe}}(\theta, \nu) + \mathcal{L}_{\text{Non-dec}}(\theta, \nu), \quad (1)$$

where θ and ν are vectors of parameters of neural networks π_θ and $\mathcal{B}_\nu$, respectively, and

$$\mathcal{L}_{\text{Init}}(\theta, \nu) = \frac{1}{|D_{\text{Init}}|} \sum_{x \in D_{\text{Init}}} \max(-\mathcal{B}_\nu(x), 0);$$

$$\mathcal{L}_{\text{Safe}}(\theta, \nu) = \frac{1}{|D_{\text{Safe}}|} \sum_{x \in D_{\text{Safe}}} \max(\mathcal{B}_\nu(x), 0);$$

$$\mathcal{L}_{\text{Non-dec}}(\theta, \nu) = \frac{1}{|D_{\text{Non-dec}}|} \cdot \sum_{x \in D_{\text{Non-dec}}} \max\Big(-\widehat{L_{f_\theta} \mathcal{B}_\nu}(x) - \mathcal{B}_\nu(x), 0\Big).$$

In words, $D_{\text{Init}}, D_{\text{Safe}}, D_{\text{Non-dec}} \subseteq \mathcal{X}$ are the training sets of system states used for each loss term that incurs loss whenever the defining condition in Proposition 1 is violated. The term $\widehat{L_{f_\theta} \mathcal{B}_\nu}(x)$ in $\mathcal{L}_{\text{Non-dec}}$ is approximated by executing the system dynamics from state x for a small time interval. Since we are interested in repair and not the initialization of π_θ and $\mathcal{B}_\nu$, we omit the details on how this training data is collected and refer the reader to (Qin, Sun, and Fan 2022).

Monitoring-based neural policy and certificate repair. We now show how the verdicts of our monitors are used to obtain new training data that describes the property or certificate violations, and how this new training data is used for the policy and certificate repair. Algorithm 1 takes as input the neural network control policy π and neural network barrier function $\mathcal{B}$, trained as above. It also takes as input a finite set of time points $t_0 = 0 < t_1 < \cdots < t_N$ at which the monitor observes new states, as well as the number D of state trajectories that it monitors. In addition, if the PredPM monitor is to be used, the algorithm also takes as input the three thresholds ξ_U, ξ_S, and ξ_N.

Algorithm 1 first initializes the monitor $\mathcal{M}$ by constructing either the CertPM or the PredPM (line 1), the set of new training data $D_{\text{New-data}}$ which is initially empty (line 2) and a set $\tilde{\mathcal{X}}_0 \subseteq \mathcal{X}_0$ of D initial states obtained via sampling from the initial set $\mathcal{X}_0$ (line 3). Then, for each initial state $x_0 \in \tilde{\mathcal{X}}_0$, it executes the dynamical system to obtain a state trajectory $x(t)$ from $x(0) = x_0$ (line 5). For each time point t_n, a new state x_n is observed (line 7) and the monitor verdict $\mathcal{M}(x_0, x_1, \ldots, x_n)$ is computed (line 8). If $\mathcal{M}(x_0, x_1, \ldots, x_n) = 1$, i.e. if the monitor issues a property or certificate violation verdict, the new state x_n is added to the new training dataset $D_{\text{New-data}}$ (line 8). Once the new data is collected, it is used to initialize the new training datasets $D^{\text{repair}}_{\text{Init}} = D_{\text{New-data}} \cap \mathcal{X}_0$, $D^{\text{repair}}_{\text{Safe}} = D_{\text{New-data}} \cap \mathcal{X}_u$ and $D^{\text{repair}}_{\text{Non-dec}} = D_{\text{New-data}} \cap \{x \mid \mathcal{B}(x) \geq 0\}$ (line 9). Finally, the neural network policy π and the neural network barrier function $\mathcal{B}$ are repaired by being retrained on the loss function in eq. (1) with the new training datasets (line 11). If we are only interested in repairing the barrier function $\mathcal{B}$ for a fixed control policy π, only the network $\mathcal{B}$ is repaired while keeping the parameters of π fixed (line 13).

Experimental Evaluation

We implemented a prototype of our method in Python 3.6 as an extension of the SABLAS codebase (Qin, Sun, and Fan 2022). SABLAS is a state-of-the-art method and tool for learning-based control of neural network control policies with certificate functions. In order to evaluate our method, we consider the benchmarks from the SABLAS codebase, together with policies and certificate functions learned by the SABLAS method for these benchmarks. We then apply our method to these control policies and certificate functions in order to experimentally evaluate the ability of our method to detect incorrect behaviors and to repair them. Our goal is to answer the following three research questions (RQs):

- **RQ1:** Is our method able to detect violating behavior and repair neural network policies and certificate functions? This RQ pertains to Problem 1.
- **RQ2:** Is our method able to detect incorrect behavior and repair neural network certificate functions? This RQ pertains to Problem 2 in the Preliminaries Section.

	$\#D_{\text{NEW}}$	**SR** (%)	**BR** (%)	**NDR** (%)
Initialized	-	93.99	87.03	45.38
Baseline	878	96.61	-	-
CertPM	548	**99.13**	**100.00**	90.66
PredPM [0,0,-1]	146	99.06	**100.00**	90.11
PredPM [0,1,-5]	355	98.67	**100.00**	90.12
PredPM [2,2,0]	**1000**	99.09	**100.00**	**91.67**

Table 1: Results on repairing the control policy and the barrier function for DroneEnv. For PredPM, we evaluate it with three threshold configurations $[\xi_U, \xi_S, \xi_N]$ (we chose three configurations for which we observed different numbers D_{NEW} of property or certificate violations; see the extended version (Yu, Zikelic, and Henzinger 2024) for results on more thresholds configurations). Column D_{NEW} shows the total number of property or certificate violations found by each method. The safety rate (SR) is the proportion of time during which the agent stays away from the unsafe set calculated by $\frac{1}{T}\int_0^T [x(t) \notin \mathcal{X}_u]\, dt$, and BR is the proportion of time during which the system is within the invariant set $\{x \mid \mathcal{B}(x) \geq 0\}$, and NDR is the proportion of time during which the barrier function satisfies the non-decreasing condition. The best results for each column are in bold. All results are averaged over 50 further executions.

- **RQ3:** How strong is predictive power of PredPM monitor, that is, can it predict safety violations *ahead of time*?

Benchmarks. We consider two benchmarks that are available in the SABLAS codebase (Qin, Sun, and Fan 2022), originally taken from (Fossen 2000; Qin et al. 2021). The first benchmark concerns a parcel delivery drone flying in a city (called the active drone), among 1024 other drones that are obstacles to be avoided. Only the active drone is controlled by the learned policy, whereas other drones move according to pre-defined routes. The state space of this environment is 8-dimensional and states are defined via 8 variables $x = [x, y, z, v_x, v_y, v_z, \theta_x, \theta_y]$: three coordinate variables in the 3D space, three velocity variables, and two variables for row and pitch angles. The actions produced by the policy correspond to angular accelerations of θ_x, θ_y, as well as the vertical thrust. The drone is completing a delivery task at a set destination, hence the property of interest in this task is a stability-while-avoid property.

The second benchmark ShipEnv concerns a ship moving in a river among 32 other ships. The state space of this environment is 6-dimensional and states are defined via 6 variables $[x, y, \theta, u, v, w]$. The first two variables specify the 2D coordinates of the ship, θ is the heading angle, u, v are the velocities in each direction, and w is the angular velocity of the heading angle. The property of interest in this task is also a stability-while-avoid property with obstacles being collisions with other ships.

Experimental setup. We consider the black-box setting, hence the dynamics are unknown to the monitor and repair algorithm. For each monitored execution, $N = 2000$ and $N = 1200$ states are observed for ShipEnv and DroneEnv, respectively, spaced out at time intervals of $\Delta_t = 0.1s$. In each environment, the states of the eight nearest obstacles (i.e. drones or ships) are given as observations to the policy, as well as the certificate functions. In our implementation of the PredPM monitor, we employ Adam (Kingma and Ba 2014) for approximating the assessments $[v_U, v_S, v_N]$.

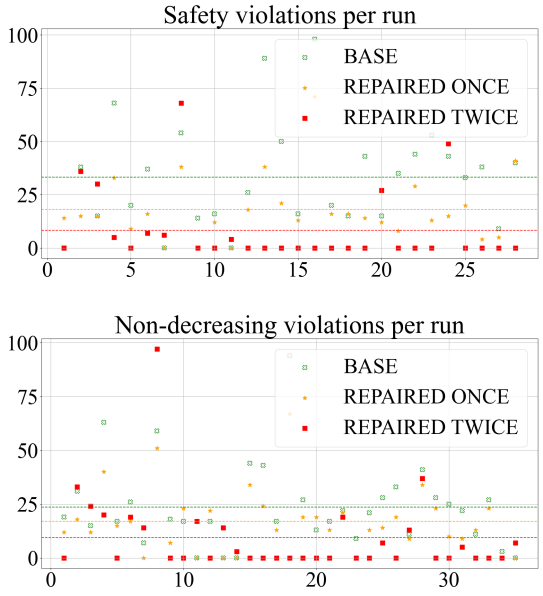

Figure 2: The change in the number of certificate violations for the Safety condition and the Non-decreasing conditions of barrier functions, for the ship benchmark after one round of repair and after a second round of repair. The x-axis represents the number of runs. The first round monitors $D = 15000$ system executions. The second round monitors an additional $D = 20000$ executions. For better readability, we plot only the executions (out of 50) for which at least one certificate violation is detected.

Results: RQ1. We observed that the control policy learned by SABLAS for DroneEnv does not satisfy the stability-while-avoid property on all runs – its safety rate is 93.99% whereas it leads to collision with other drones 6.01% of the time, initialized with 10^4 state samples. We applied our repair method to the control policy and the barrier function learned by SABLAS with $D = 1000$. To evaluate the importance of monitoring both neural policies and certificates, we also compare our method against the *baseline* approach. The baseline is the simple monitor described in the Introduction, which only monitors a neural policy, flags traces that reach an unsafe state and adds these states to re-training data. Our results are summarized in Table 1. As we can see, both CertPM and PredPM monitors are able to effectively repair the control policy and the barrier function and lead to significantly higher safety rates (SR), reaching 99.13%. The proportions of time at which the Safety and the Non-decreasing conditions of barrier functions are satisfied (BR and NDR in Table 1) go up from 87.03% to 100.00%, and from 45.38% to 91.67%. This demonstrates the advantage of monitoring both the control policy and the certificate to-

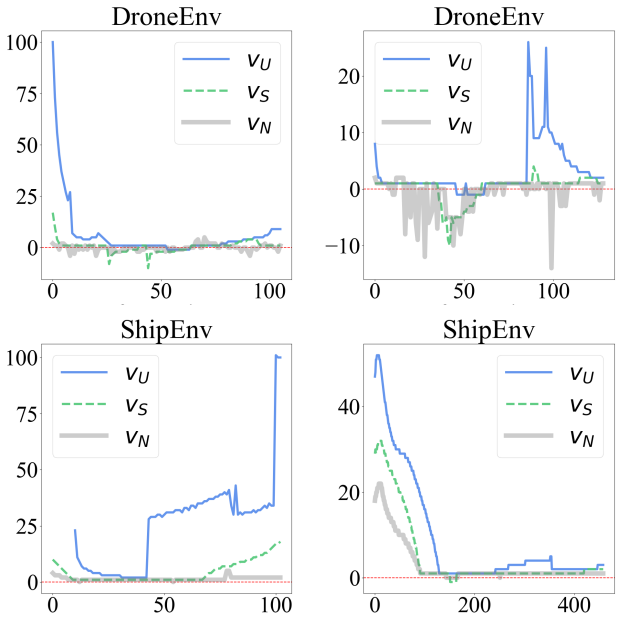

Figure 3: Number of excution steps (x-axis) vs. verdicts (y-axis). Estimates v_U, v_S, and v_N for two different systems executions computed by the PredPM in both environments.

ward effective and successful repair. Finally, the comparison between CertPM and PredPM shows that CertPM is slightly better in repairing the control policy, but PredPM is more effective for repairing the barrier function. Due to its predictive nature, PredPM identifies a larger number of property and certificate violations with the right choice of thresholds, resulting in larger re-training data $D_{\text{NEW-DATA}}$. We conducted an analogous experiment with a certificate function consisting both of a barrier and a Lyapunov function and again observed significant improvements upon repair. Further details, including results on additional threshold configurations of PredPM beyond the three configurations discussed in Table 1, can be found in the extended version (Yu, Zikelic, and Henzinger 2024).

Results: RQ2. We observed that the policy learned by SABLAS for the ship benchmark already achieves SR close to 100.00%, however, the learned barrier function provides significantly lower BR and NDR. Hence, we use the ship benchmark to answer RQ2 which is concerned with the repair of a certificate function for a *given* control policy. In this case, a good certificate function acts as a proof of correctness that allows more trustworthy policy deployment. Figure 2 shows what the number of certificate violations for the barrier function looks like before and after repair, for the ship benchmark with CertPM used as a monitor. The results demonstrate that there are significantly fewer certificate violations upon repair. Additionally, we conducted the same experiment for the drone benchmark, and also observed significant level of improvement in BR and NDR upon repair. We refer to the extened version for more results.

Results: RQ3. Recall that PredPM considers safety properties and monitors the control policy together with the barrier function. Upon each new observation, it computes an estimate v_U on the remaining time before the safety property may be violated, v_S on the remaining time before the Safety condition of barrier functions may be violated, and v_N on the remaining time before the Non-decreasing condition of barrier functions may be violated. Figure 3 shows how these estimates change along two different executions for DroneEnv and ShipEnv. It can be seen that v_S becomes negative before v_U, meaning the system is estimated to violate the Safety condition of barrier functions before it reaches the unsafe region. Hence, by tracking v_S, PredPM can predict unsafe behaviors and raise warnings *before* they happen. In comparison to the others, there are more estimated violations of the Non-decreasing condition. This means that ξ_N can be set to a lower value, as we tend to consider the other two violations to be more severe. Overall, the experiments suggest PredPM can be particularly well-suited for runtime use, in addition to repairing neural networks.

Summary of results. Our experimental results empirically justify the following claims: (i) Our method is able to successfully repair neural network control policies and certificate functions. Using either CertPM or PredPM for repair leads to significant improvements over the initial policy; (ii) Our method is able to successfully repair neural network certificate functions in the setting where the control policies are fixed; (iii) Using PredPM allows predicting safety property violations *before they happen*, hence showing potential for practical safety deployment even in the runtime setting.

Practical considerations and limitations. To conclude, we also discuss two practical aspects that one should take into account before the deployment of our method: (i) As highlighted in the Introduction, our method provides no guarantees on the correctness of repaired policies. This means that, in principle, one could end up with a policy whose performance is suboptimal compared to the initial policy. However, we did not observe a single case of such a behavior in our experiments. (ii) It was observed by (Zikelic et al. 2022) that methods for jointly learning policies and certificates rely on a good policy initialization. Hence, our repair method is also best suited for cases when the policy is well initialized by some off-the-shelf method (e.g. with SR at least 90%).

Conclusion

In this work, we propose a method for determining the correctness of neural network control policies and certificate functions and for repairing them by utilizing runtime monitoring. Our method applies to the black-box setting and does not assume knowledge of the system dynamics. We present two novel monitoring algorithms, CertPM and PredPM. Our experiments demonstrate the advantage of monitoring policies together with certificate functions and are able to repair neural policies and certificates learned by a state-of-the-art learning-based control method. Interesting directions of future work would be to consider the repair problem for stochastic systems and multi-agent systems. Another interesting direction would be to explore the possibility of deploying predictive monitors towards enhancing the safety of learned controllers upon deployment, i.e., at runtime.

Acknowledgments

This work was supported in part by the ERC project ERC-2020-AdG 101020093.

References

Abate, A.; Ahmed, D.; Edwards, A.; Giacobbe, M.; and Peruffo, A. 2021a. FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In Bogomolov, S.; and Jungers, R. M., eds., *HSCC '21: 24th ACM International Conference on Hybrid Systems: Computation and Control, Nashville, Tennessee, May 19-21, 2021*, 24:1–24:11. ACM.

Abate, A.; Ahmed, D.; Giacobbe, M.; and Peruffo, A. 2021b. Formal Synthesis of Lyapunov Neural Networks. *IEEE Control. Syst. Lett.*, 5(3): 773–778.

Abate, A.; Edwards, A.; Giacobbe, M.; Punchihewa, H.; and Roy, D. 2023. Quantitative Verification with Neural Networks. In Pérez, G. A.; and Raskin, J., eds., *34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium*, volume 279 of *LIPIcs*, 22:1–22:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International Conference on Machine Learning*, 22–31. PMLR.

Ahmadi, A. A.; and Majumdar, A. 2016. Some applications of polynomial optimization in operations research and real-time decision making. *Optim. Lett.*, 10(4): 709–729.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2669–2678. AAAI Press.

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Ames, A. D.; Grizzle, J. W.; and Tabuada, P. 2014. Control barrier function based quadratic programs with application to adaptive cruise control. In *CDC*, 6271–6278. IEEE.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P. F.; Schulman, J.; and Mané, D. 2016. Concrete Problems in AI Safety. *CoRR*, abs/1606.06565.

Ansaripour, M.; Chatterjee, K.; Henzinger, T. A.; Lechner, M.; and Zikelic, D. 2023. Learning Provably Stabilizing Neural Controllers for Discrete-Time Stochastic Systems. In *ATVA (1)*, volume 14215 of *Lecture Notes in Computer Science*, 357–379. Springer.

Bauer-Marquart, F.; Boetius, D.; Leue, S.; and Schilling, C. 2022. SpecRepair: Counter-Example Guided Safety Repair of Deep Neural Networks. In Legunsen, O.; and Rosu, G., eds., *Model Checking Software - 28th International Symposium, SPIN 2022, Virtual Event, May 21, 2022, Proceedings*, volume 13255 of *Lecture Notes in Computer Science*, 79–96. Springer.

Bloem, R.; Könighofer, B.; Könighofer, R.; and Wang, C. 2015. Shield Synthesis: - Runtime Enforcement for Reactive Systems. In *TACAS*, volume 9035 of *Lecture Notes in Computer Science*, 533–548. Springer.

Carr, S.; Jansen, N.; Junges, S.; and Topcu, U. 2023. Safe Reinforcement Learning via Shielding under Partial Observability. In *AAAI*, 14748–14756. AAAI Press.

Chang, Y.; Roohi, N.; and Gao, S. 2019. Neural Lyapunov Control. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 3240–3249.

Chatterjee, K.; Henzinger, T. A.; Lechner, M.; and Zikelic, D. 2023. A Learner-Verifier Framework for Neural Network Controllers and Certificates of Stochastic Systems. In *TACAS (1)*, volume 13993 of *Lecture Notes in Computer Science*, 3–25. Springer.

Chow, Y.; Nachum, O.; Duéñez-Guzmán, E. A.; and Ghavamzadeh, M. 2018. A Lyapunov-based Approach to Safe Reinforcement Learning. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 8103–8112.

Dawson, C.; Gao, S.; and Fan, C. 2023. Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control. *IEEE Trans. Robotics*, 39(3): 1749–1767.

Dawson, C.; Qin, Z.; Gao, S.; and Fan, C. 2021. Safe Nonlinear Control Using Robust Neural Lyapunov-Barrier Functions. In Faust, A.; Hsu, D.; and Neumann, G., eds., *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, 1724–1735. PMLR.

Falcone, Y.; and Pinisetty, S. 2019. On the Runtime Enforcement of Timed Properties. In *RV*, volume 11757 of *Lecture Notes in Computer Science*, 48–69. Springer.

Fossen, T. I. 2000. A survey on nonlinear ship control: From theory to practice. *IFAC Proceedings Volumes*, 33(21): 1–16.

García, J.; and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16: 1437–1480.

Geibel, P. 2006. Reinforcement Learning for MDPs with Constraints. In Fürnkranz, J.; Scheffer, T.; and Spiliopoulou, M., eds., *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, 646–653. Springer.

Khalil, H. K. 2002. *Control of nonlinear systems*. Prentice Hall, New York, NY.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Könighofer, B.; Lorber, F.; Jansen, N.; and Bloem, R. 2020. Shield Synthesis for Reinforcement Learning. In *ISoLA (1)*, volume 12476 of *Lecture Notes in Computer Science*, 290–306. Springer.

Lechner, M.; Zikelic, D.; Chatterjee, K.; and Henzinger, T. A. 2022. Stability Verification in Stochastic Control Systems via Neural Network Supermartingales. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 7326–7336. AAAI Press.

Lyu, D.; Song, J.; Zhang, Z.; Wang, Z.; Zhang, T.; Ma, L.; and Zhao, J. 2023. AutoRepair: Automated Repair for AI-Enabled Cyber-Physical Systems under Safety-Critical Conditions. *CoRR*, abs/2304.05617.

Mathiesen, F. B.; Calvert, S. C.; and Laurenti, L. 2023. Safety Certification for Stochastic Systems via Neural Barrier Functions. *IEEE Control. Syst. Lett.*, 7: 973–978.

Mazouz, R.; Muvvala, K.; Ratheesh, A.; Laurenti, L.; and Lahijanian, M. 2022. Safety Guarantees for Neural Network Dynamic Systems via Stochastic Barrier Functions. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nat.*, 518(7540): 529–533.

Nejati, A.; Lavaei, A.; Jagtap, P.; Soudjani, S.; and Zamani, M. 2023. Formal Verification of Unknown Discrete- and Continuous-Time Systems: A Data-Driven Approach. *IEEE Trans. Autom. Control.*, 68(5): 3011–3024.

Prajna, S.; and Jadbabaie, A. 2004. Safety Verification of Hybrid Systems Using Barrier Certificates. In *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, 477–492. Springer.

Pranger, S.; Könighofer, B.; Tappler, M.; Deixelberger, M.; Jansen, N.; and Bloem, R. 2021. Adaptive Shielding under Uncertainty. In *ACC*, 3467–3474. IEEE.

Qin, Z.; Sun, D.; and Fan, C. 2022. Sablas: Learning Safe Control for Black-Box Dynamical Systems. *IEEE Robotics Autom. Lett.*, 7(2): 1928–1935.

Qin, Z.; Zhang, K.; Chen, Y.; Chen, J.; and Fan, C. 2021. Learning Safe Multi-agent Control with Decentralized Neural Barrier Certificates. In *ICLR*. OpenReview.net.

Renard, M.; Falcone, Y.; Rollet, A.; Jéron, T.; and Marchand, H. 2019. Optimal enforcement of (timed) properties with uncontrollable events. *Math. Struct. Comput. Sci.*, 29(1): 169–214.

Richards, S. M.; Berkenkamp, F.; and Krause, A. 2018. The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, 466–476. PMLR.

Srinivasan, M.; Abate, M.; Nilsson, G.; and Coogan, S. 2021. Extent-compatible control barrier functions. *Syst. Control. Lett.*, 150: 104895.

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

Yu, E.; Zikelic, D.; and Henzinger, T. A. 2024. Neural Control and Certificate Repair via Runtime Monitoring. arXiv:2412.12996.

Zhang, H.; Wu, J.; Vorobeychik, Y.; and Clark, A. 2023. Exact Verification of ReLU Neural Control Barrier Functions. In *NeurIPS*.

Zhao, H.; Zeng, X.; Chen, T.; and Liu, Z. 2020. Synthesizing barrier certificates using neural networks. In *HSCC*, 25:1–25:11. ACM.

Zhi, D.; Wang, P.; Liu, S.; Ong, C. L.; and Zhang, M. 2024. Unifying Qualitative and Quantitative Safety Verification of DNN-Controlled Systems. In Gurfinkel, A.; and Ganesh, V., eds., *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II*, volume 14682 of *Lecture Notes in Computer Science*, 401–426. Springer.

Zhou, W.; Gao, R.; Kim, B.; Kang, E.; and Li, W. 2020. Runtime-Safety-Guided Policy Repair. In *RV*, volume 12399 of *Lecture Notes in Computer Science*, 131–150. Springer.

Zikelic, D.; Lechner, M.; Chatterjee, K.; and Henzinger, T. A. 2022. Learning Stabilizing Policies in Stochastic Control Systems. *CoRR*, abs/2205.11991.

Zikelic, D.; Lechner, M.; Henzinger, T. A.; and Chatterjee, K. 2023a. Learning Control Policies for Stochastic Systems with Reach-Avoid Guarantees. In *AAAI*, 11926–11935. AAAI Press.

Zikelic, D.; Lechner, M.; Verma, A.; Chatterjee, K.; and Henzinger, T. A. 2023b. Compositional Policy Learning in Stochastic Control Systems with Formal Guarantees. In *NeurIPS*.