Article:
Zhao, Hengjun, Zeng, Xia, Chen, Taolue et al. (2 more authors) (2021) Learning safe neural network controllers with barrier certificates. Formal Aspects of Computing. pp. 437-455. ISSN: 1433-299X

https://doi.org/10.1007/s00165-021-00544-5

Learning Safe Neural Network Controllers with Barrier Certificates*

Hengjun Zhao[1], Xia Zeng[1], Taolue Chen[2], Zhiming Liu[1], and Jim Woodcock[1,3]

[1] School of Computer and Information Science, Southwest University
{zhaohj2016, xzeng0712, zhimingliu88}@swu.edu.cn
[2] Department of Computer Science, University of Surrey
taolue.chen@surrey.ac.uk
[3] Department of Computer Science, University of York
jim.woodcock@york.ac.uk

Abstract. We provide a novel approach to synthesize controllers for nonlinear continuous dynamical systems with control against safety properties. The controllers are based on neural networks (NNs). To certify the safety property we utilize barrier functions, which are also represented by NNs. We train controller-NN and barrier-NN simultaneously, achieving verification-in-the-loop synthesis. We provide a prototype tool nncontroller with a number of case studies. Preliminary experiment results confirm the feasibility and efficacy of our approach.

Keywords: Continuous dynamical systems · Controller synthesis · Neural networks · Safety verification · Barrier certificates

1 Introduction

Controller design and synthesis is the most fundamental problem in control theory. In recent years, especially with the boom of deep learning, there have been considerable research activities in the use of neural networks (NNs) for control of nonlinear systems [15, 8]. NNs feature versatile representational abilities of nonlinear maps and fast computation, making them an ideal candidate for sophisticated control tasks [16]. Typical examples include self-driving cars, drones, and smart cities. It is noteworthy that many of these applications are safety-critical systems, where safety refers to, in a basic form, that the system cannot reach a dangerous or unwanted state. For control systems in a multitude of Cyber-Physical-System domains, designing *safe* controllers which can guarantee

* H. Zhao is supported partially by the National Natural Science Foundation of China (No. 61702425, 61972385); X. Zeng is the corresponding author and supported partially by the National Natural Science Foundation of China (No. 61902325) and "Fundamental Research Funds for the Central Universities" (SWU117058); T. Chen is partially supported by NSFC grant (No. 61872340), Guangdong Science and Technology Department grant (No. 2018B010107004), and the Overseas Grant of the State Key Laboratory of Novel Software Technology (No. KFKT2018A16), the Natural Science Foundation of Guangdong Province of China (No. 2019A1515011689); Z. Liu is supported partially by the National Natural Science Foundation of China (No. 61672435, 61732019, 61811530327) and Capacity Development Grant of Southwest University (SWU116007); J. Woodcock is partially supported by the research grant from Southwest University.

safety behaviors of the controlled system is of paramount importance [19, 2, 20, 9, 23, 4, 5, 26, 12, 24, 1, 10].

Typically, when a controller is given, formal verification is usually required to certify its safety. In our previous work [27], we have dealt with the verification of continuous dynamical systems by the aid of neural networks. In a nutshell, we follow a deductive verification methodology therein by synthesizing a barrier function, the existence of which suffices to show the safety of the controlled dynamical system. The crux was to use neural networks to represent the barrier functions. In this work, we follow a correctness-by-design methodology by considering synthesizing controllers which can guarantee that the controlled system is safe, which is considerably more challenging and perhaps more interesting. In a nutshell, we learn two neural networks simultaneously: one is to represent the controller (henceforth referred to as controller-NN), and the other is to represent the barrier function (henceforth referred to as barrier-NN). The synergy of the two neural networks, supported by an additional verification procedure to make sure the learned barrier-NN is indeed a barrier certificate, provides the desired safety guarantee for the application. The advantages of our approach are threefold: (1) the approach is data-driven, requiring considerably less control theory expertise; (2) the approach can support non-linear control systems and safety properties, owing to the representation power of neural networks; and (3) the approach can achieve verification-in-the-loop synthesis, owing to the co-synthesis of controllers and barrier functions, which can be seamlessly integrated to provide a correctness-by-design controller as well as its certification.

This short paper reports the general methodology, including the design of the NNs, the generation of training data, the training process, as well as the associated verification. We shall also report a prototype implementation and some preliminary experiment results. For space restriction, we will leave the discussion of related work (e.g., [3, 25, 7]), and further improvement of the performance of the learned controller (e.g., bounded and asymptotically stable controller), as well as more extensive experiments, to the full version. ([ref-arxiv], Hengjun)

2 Preliminaries

A *constrained continuous dynamical system* (CCDS) is represented by $\Gamma = (\mathbf{f}, X_D, X_I, X_U)$, where $\mathbf{f} : \Omega \to \mathbb{R}^n$ is the vector field, $X_D \subseteq \Omega$ is the system domain, $X_I \subseteq X_D$, and $X_U \subseteq X_D$. The system dynamics is governed by first-order ordinary differential equations $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ for $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_n(\mathbf{x}))$, where $\dot{\mathbf{x}}$ denotes the derivative of $\mathbf{x}$ w.r.t the time variable t. We assume that $\mathbf{f}$ satisfies the *local Lipschitz condition*, which ensures that given $\mathbf{x} = \mathbf{x}_0$, there exists a time $\mathcal{T} > 0$ and a unique trajectory $\mathbf{x}(t) : [0, \mathcal{T}) \to \mathbb{R}^n$ such that $\mathbf{x}(0) = \mathbf{x}_0$, which is denoted by $\mathbf{x}(t, \mathbf{x}_0)$. In this paper, we consider *controlled* CCDS

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{u} = \mathbf{g}(\mathbf{x}) \end{cases} \tag{1}$$

where $\mathbf{u} \in U \subseteq \mathbb{R}^m$ are the feedback control inputs, and $\mathbf{f} : \mathbb{R}^{n+m} \to \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^m$ are locally Lipschitz continuous.

The problem we considered in this paper is defined as follows.

Definition 1 (Safe Controller Synthesis). *Given a controlled CCDS* $\Gamma = (\mathbf{f}, X_D, X_I, X_U)$ *with* $\mathbf{f}$ *defined by (1), design a locally continuous feedback control law* $\mathbf{g}$ *such that the closed-loop system* Γ *with* $\mathbf{f} = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}))$ *is safe, that is, system trajectory from* X_I *never reaches* X_U *as long as it remains in* X_D*.*

Barrier Certificate. Given a system Γ, a barrier certificate is a real-valued function $B(\mathbf{x})$ over the states of the system satisfying the condition that $B(\mathbf{x}) \leq 0$ for any reachable state $\mathbf{x}$ and $B(\mathbf{x}) > 0$ for any state in the unsafe set X_U. If such a function $B(\mathbf{x})$ exists, one can easily deduce that the system can *not* reach a state in the unsafe set from the initial set [17, 18]. In this paper, we will certify the safety of a synthesized controller by generating barrier certificates.

There are several different formulations of barrier certificates without explicit reference to the solutions of the ODEs [17, 14, 6, 22]. In this paper, we will adopt what are called *strict barrier certificate* [21] conditions.

Theorem 1 (Strict barrier certificate). *Given a system* $\Gamma = (\mathbf{f}, X_D, X_I, X_U)$*, if there exists a continuously differentiable function* $B : X_D \to \mathbb{R}$ *such that (1)* $B(\mathbf{x}) \leq 0$ *for* $\forall \mathbf{x} \in X_I$*, (2)* $B(\mathbf{x}) > 0$ *for* $\forall \mathbf{x} \in X_U$*, and (3)* $\mathcal{L}_{\mathbf{f}} B(\mathbf{x}) < 0$ *for all* $\mathbf{x} \in X_D$ *s.t.* $B(\mathbf{x}) = 0$*. Then the system* Γ *is safe, and such* B *is a barrier certificate. (Note that* $\mathcal{L}_{\mathbf{f}} B$ *is the* Lie derivative *of* B *w.r.t.* $\mathbf{f}$ *defined as* $\mathcal{L}_{\mathbf{f}} B(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \cdot \nabla B = \sum_{i=1}^{n} (f_i(\mathbf{x}) \cdot \frac{\partial B}{\partial x_i}(\mathbf{x}))$*.)*

3 Methodology

The framework of our safe controller learning approach is demonstrated in Fig. 1. Given a controlled CCDS $\Gamma = (\mathbf{f}, X_D, X_I, X_U)$, the basic idea of the proposed

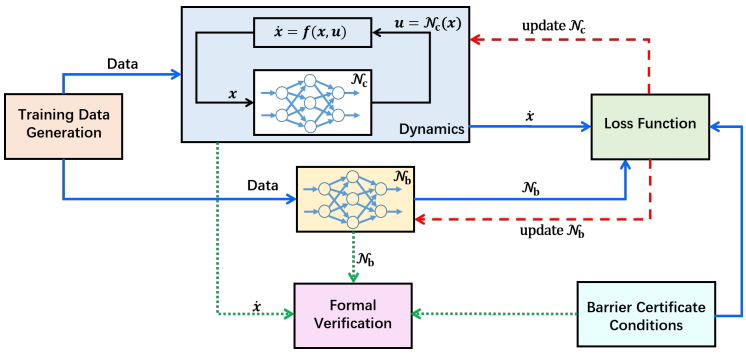

Fig. 1. The framework of safe neural network controller synthesis

approach is to represent the controller function $\mathbf{g}$ as well as the safety certificate function B by two NNs, i.e. $\mathcal{N}_c$ and $\mathcal{N}_b$ respectively. Then we formulate the

barrier certificate conditions w.r.t. $\mathcal{N}_\mathrm{b}$ and the closed-loop dynamics $\mathbf{f}(\mathbf{x}, \mathcal{N}_\mathrm{c}(\mathbf{x}))$ into a loss function, and then train the two NNs together on a generated training data set until the loss is decreased to 0. The resulting two NNs are the controller and barrier certificate candidates, the correctness of which is guaranteed by formal verification (SMT solver in this paper). The blue (solid), red (dashed), and green (dotted) arrows in Fig. 1 shows the information flow of forward propagation, backward propagation, and formal verification, respectively. Next, before giving more detailed steps of our approach, we first introduce a running example.

Example 1 (Dubins' car [25, 7]). The control objective is to steer a car with constant velocity 1 to track a path, here the X-axis in the postive direction. The states of the car are the x, y position and the driving direction θ, which can be transformed to the distance error d_e and angle error θ_e between the current position and the target path (cf. the left part of Fig. 2). The controlled CCDS is:

$$\mathbf{f}: \begin{bmatrix} \dot{d}_\mathrm{e} \\ \dot{\theta}_\mathrm{e} \end{bmatrix} = \begin{bmatrix} \sin(\theta_\mathrm{e}) \\ -u \end{bmatrix}, \quad \text{where } u \text{ is the scalar control input}$$

- X_D: $\{(d_\mathrm{e}, \theta_\mathrm{e}) \in \mathbb{R}^2 | -6 \leq d_\mathrm{e} \leq 6, -7\pi/10 \leq \theta_\mathrm{e} \leq 7\pi/10\}$;
- X_I: $\{(d_\mathrm{e}, \theta_\mathrm{e}) \in \mathbb{R}^2 | -1 \leq d_\mathrm{e} \leq 1, -\pi/16 \leq \theta_\mathrm{e} \leq \pi/16\}$;
- X_U: the complement of $\{(d_\mathrm{e}, \theta_\mathrm{e}) \in \mathbb{R}^2 | -5 \leq d_\mathrm{e} \leq 5, -\pi/2 \leq \theta_\mathrm{e} \leq \pi/2\}$.

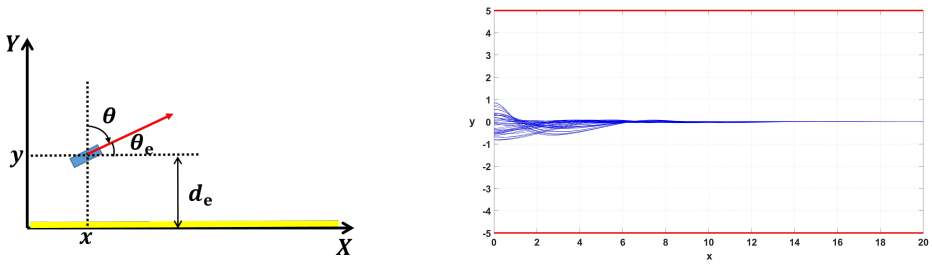

Fig. 2. Left: the car states; Right: simulated car trajectories with learned NN controller.

The right of Fig. 2 shows simulated trajectories on the x-y plane from 50 random initial states in X_I using our learned NN controller u. The two red horizontal lines are the safety upper and lower bounds (± 5) for y (the same bounds as d_e).

3.1 The Structure of $\mathcal{N}_\mathrm{c}$ and $\mathcal{N}_\mathrm{b}$

We first fix the structure of $\mathcal{N}_\mathrm{c}$ and $\mathcal{N}_\mathrm{b}$ as follows, assuming that in Γ, $\mathbf{x}$ and $\mathbf{u}$ are of n and m dimension respectively, e.g. $n = 2, m = 1$ for Example 1.

- **Input layer** has n neurons for both $\mathcal{N}_\mathrm{c}$ and $\mathcal{N}_\mathrm{b}$;
- **Output layer** has m neurons for $\mathcal{N}_\mathrm{c}$ and one single neuron for $\mathcal{N}_\mathrm{b}$;
- **Hidden layer**: there is no restriction on the number of hidden layers or the number of neurons in each hidden layer; for Example 1, $\mathcal{N}_\mathrm{c}$ has one hidden layer with 5 neurons, and $\mathcal{N}_\mathrm{b}$ has one hidden layer with 10 neurons.

– **Activation function**: considering both the simplicity and the inherent requirement of local Lipschitz continuity [13] and differentiability, we adopt ReLU, i.e. $a(x) = \max(0, x)$ for $\mathcal{N}_\mathrm{c}$, and *Bent-ReLU* [27], i.e., $a(x) = 0.5 \cdot x + \sqrt{0.25 \cdot x^2 + 0.0001}$ for $\mathcal{N}_\mathrm{b}$ respectively for hidden layers; the activation of the output layer is the identity map for both $\mathcal{N}_\mathrm{c}$ and $\mathcal{N}_\mathrm{b}$.

3.2 Training Data Generation

In our training algorithm, training data are generated by sampling points from the domain, initial set, and unsafe region of the considered system. No simulation of the continuous dynamics is needed. The simplest sampling method is to grid the super-rectangles bounding X_D, X_I, X_U with a fixed mesh size, and then filter those points not satisfying the constraints of X_D, X_I, X_U. For example, we generate a mesh with $2^8 \times 2^8$ points from X_D for Example 1. The obtained three finite data sets are denoted by S_D, S_I, and S_U.

3.3 Loss Function Encoding

Given S_I, S_U, and S_D, the loss function can be expressed as

$$L(S_D, S_I, S_U) = \sum_{\mathbf{x} \in S_I} L_1(\mathbf{x}) + \sum_{\mathbf{x} \in S_U} L_2(\mathbf{x}) + \sum_{\mathbf{x} \in S_D} L_3(\mathbf{x}) \quad \text{with} \tag{2}$$

$$\begin{aligned} L_1(\mathbf{x}) &= \mathrm{ReLU}(\mathcal{N}_\mathrm{b}(\mathbf{x}) + \varepsilon_1) \quad \text{for } \mathbf{x} \in S_I\,, \\ L_2(\mathbf{x}) &= \mathrm{ReLU}(-\mathcal{N}_\mathrm{b}(\mathbf{x}) + \varepsilon_2) \quad \text{for } \mathbf{x} \in S_U\,, \\ L_3(\mathbf{x}) &= \mathrm{ReLU}\big(\mathcal{L}_\mathbf{f}\mathcal{N}_\mathrm{b}(\mathbf{x}) + \varepsilon_3\big) \text{ for } \mathbf{x} \in \{\mathbf{x} \in S_D : |\mathcal{N}_\mathrm{b}(\mathbf{x})| \leq \varepsilon_4\} \end{aligned} \tag{3}$$

where the sub-loss functions L_1-L_3 encode the three conditions of Theorem 1. The basic idea is to give a positive (resp., zero) penalty to those sampled points that violate (resp., satisfy) barrier certificate conditions. $\varepsilon_1, \varepsilon_2, \varepsilon_3$ are three small non-negative tolerances, and ε_4 is a small positive constant approximating the zero-level set of $\mathcal{N}_\mathrm{b}$. Note that in the expression $\mathcal{L}_\mathbf{f}\mathcal{N}_\mathrm{b}$ above, $\mathbf{f}$ is $\mathbf{f}(\mathbf{x}, \mathcal{N}_\mathrm{c}(\mathbf{x}))$.

3.4 The Training Process

We adopt a modified *stochastic gradient descent* (SGD) [11] optimization technique for training the two NNs. That is, we partition the training data set into mini-batches and shuffle the list of batches, rather than the whole training set, to gain some randomness effect. To start the training, we specify ε_1 to ε_4 in the loss function, as well as hyper-parameters such as learning rate, number of epochs, number of mini-batches, etc. The training algorithm terminates when the loss is decreased to 0 or exceeds the maximum number of restarts.

3.5 Formal Verification

The rigorousness of the NNs resulting from 0 training loss is not guaranteed since our approach is data-driven. Therefore we resort to formal verification to guarantee the correctness of our synthesized controllers. To preform the verification, we replace the $\mathbf{f}$ and B in the conditions of Theorem 1 by $\mathbf{f}(\mathbf{x}, \mathcal{N}_\mathrm{c}(\mathbf{x}))$ and

$\mathcal{N}_\mathrm{b}$, and the negation of the resulting constraints are sent to the interval SMT solver iSAT3[1] for satisfiability checking. To reduce the degree of nonlinearity in the constraints, we compute piece-wise linear approximations of the Bent-ReLU function and its derivative.

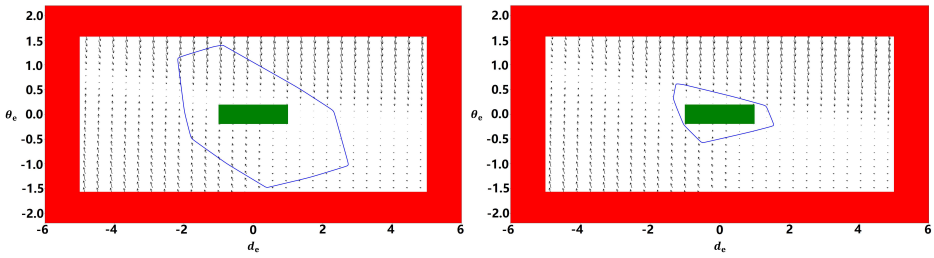

Fig. 3. Learned and verified NN controller and barrier certificate for Example 1. Left: $\varepsilon_1 = \varepsilon_2 = 0$, $\varepsilon_3 = \varepsilon_4 = 0.01$, the inner (green) and outer (red) shaded areas are the initial and unsafe regions, black arrows in the white area are the closed-loop vector fields $\mathbf{f}(\mathbf{x},\mathcal{N}_\mathrm{c}(\mathbf{x}))$, and the blue curve surrounding the inner shaded box is the zero-level set of $\mathcal{N}_\mathrm{b}$; Right: $\varepsilon_1 = 0.02$, $\varepsilon_2 = 0.8$, $\varepsilon_3 = 0.01$, $\varepsilon_4 = 0.05$.

Pre-training and Fine-tuning. The success of synthesis and formal verification heavily relies on the choices of the four constants ε_1 to ε_4 in (2) and (3). In practice, we adopt a pre-training and fine-tuning combination strategy. That is, we start with small positive ε_4 and zero ε_1 to ε_3 to perform the training and verification, and gradually increase the values when formal verification fails. For Example 1, the first controller and barrier are synthesized with $\varepsilon_4 = 0.01$ and $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 0$, and the fine-tuned controller and barrier are successfully verified when ε_3 was increased to 0.01 (cf. the left part of Fig. 3). Further fine-tuning gives a controller with larger safety margin (cf. the right part of Fig. 3).

4 Implementation and Experiment

We have implemented a prototype tool nncontroller[2] based on PyTorch[3]. We apply nncontroller to a number of case studies in the literature [25, 7, 28]. All experiments are performed on a laptop running Ubuntu 18.04 with Intel i7-8550u CPU and 32GB memory. The average time costs of pre-training for Example 1 over 5 different runs is 26.35 seconds. Formal verification of the fine-tuned controller for Example 1 (cf. the left part of Fig. 3) with iSAT3 costs 8.27 seconds. The other experiment results will be reported in the full version. [ref-arxiv]

Acknowledgements. We thank anonymous reviewers for their valuable comments.

[1] https://projects.informatik.uni-freiburg.de/projects/isat3/
[2] Publicly available at: https://github.com/zhaohj2017/FAoC-tool
[3] https://pytorch.org/

References

1. Ahmadi, M., Singletary, A., Burdick, J.W., Ames, A.D.: Safe policy synthesis in multi-agent POMDPs via discrete-time barrier functions. In: 2019 IEEE 58th Conference on Decision and Control (CDC). pp. 4797–4803. IEEE (2019)
2. Berkenkamp, F., Turchetta, M., Schoellig, A.P., Krause, A.: Safe model-based reinforcement learning with stability guarantees. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 908–919. NIPS17, Curran Associates Inc., Red Hook, NY, USA (2017)
3. Chang, Y.C., Roohi, N., Gao, S.: Neural Lyapunov control. In: Advances in Neural Information Processing Systems 32, pp. 3245–3254. Curran Associates, Inc. (2019)
4. Cheng, R., Orosz, G., Murray, R.M., Burdick, J.W.: End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. pp. 3387–3395. AAAI Press (2019)
5. Choi, J., Castaeda, F., Tomlin, C.J., Sreenath, K.: Reinforcement learning for safety-critical control under model uncertainty, using control Lyapunov functions and control barrier functions (2020), https://arxiv.org/abs/2004.07584
6. Dai, L., Gan, T., Xia, B., Zhan, N.: Barrier certificates revisited. Journal of Symbolic Computation **80**, 62–86 (2017)
7. Deshmukh, J.V., Kapinski, J., Yamaguchi, T., Prokhorov, D.: Learning deep neural network controllers for dynamical systems with safety guarantees: Invited paper. In: 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). pp. 1–7 (2019)
8. Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016. JMLR Workshop and Conference Proceedings, vol. 48, pp. 1329–1338. JMLR.org (2016)
9. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Learning and verification of feedback control systems using feedforward neural networks. IFAC-PapersOnLine **51**(16), 151 – 156 (2018), 6th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2018
10. Fulton, N., Platzer, A.: Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 6485–6492. AAAI Press (2018)
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press (2016)
12. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Case study: verifying the safety of an autonomous racing car with a neural network controller. In: HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020. pp. 28:1–28:7. ACM (2020)
13. Jordan, M., Dimakis, A.G.: Exactly computing the local Lipschitz constant of ReLU networks (2020), https://arxiv.org/abs/2003.01219
14. Kong, H., He, F., Song, X., Hung, W.N., Gu, M.: Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In: Proceedings of the 25th International Conference on Computer Aided Verification (CAV). pp. 242–257. Springer (2013)

15. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016)
16. Poznyak, A., EN, S., Yu, W.: Differential Neural Networks for Robust Nonlinear Control. World Scientific (2001)
17. Prajna, S., Jadbabaie, A., Pappas, G.J.: A framework for worst-case and stochastic safety verification using barrier certificates. IEEE Transactions on Automatic Control **52**(8), 1415–1429 (2007)
18. Ratschan, S.: Converse theorems for safety and barrier certificates. IEEE Transactions on Automatic Control **63**(8), 2628–2632 (2018)
19. Ray, A., Achiam, J., Amodei, D.: Benchmarking safe exploration in deep reinforcement learning, https://cdn.openai.com/safexp-short.pdf
20. Richards, S.M., Berkenkamp, F., Krause, A.: The Lyapunov neural network: Adaptive stability certification for safe learning of dynamic systems. CoRR **abs/1808.00924** (2018), http://arxiv.org/abs/1808.00924
21. Sloth, C., Pappas, G.J., Wisniewski, R.: Compositional safety analysis using barrier certificates. In: Proc. of the Hybrid Systems: Computation and Control (HSCC). pp. 15–24. ACM (2012)
22. Sogokon, A., Ghorbal, K., Tan, Y.K., Platzer, A.: Vector barrier certificates and comparison systems. In: Formal Methods. pp. 418–437 (2018)
23. Taylor, A., Singletary, A., Yue, Y., Ames, A.: Learning for safety-critical control with control barrier functions (2019), https://arxiv.org/abs/1912.10099
24. Tran, H.D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: Computer Aided Verification. pp. 3–17. Springer International Publishing (2020)
25. Tuncali, C.E., Kapinski, J., Ito, H., Deshmukh, J.V.: Invited: Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC). pp. 1–6 (2018)
26. Yaghoubi, S., Fainekos, G., Sankaranarayanan, S.: Training neural network controllers using control barrier functions in the presence of disturbances (2020), https://arxiv.org/abs/2001.08088
27. Zhao, H., Zeng, X., Chen, T., Liu, Z.: Synthesizing barrier certificates using neural networks. In: HSCC '20. pp. 25:1–25:11. ACM (2020)
28. Zhu, H., Xiong, Z., Magill, S., Jagannathan, S.: An inductive synthesis framework for verifiable reinforcement learning. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 686–701. PLDI 2019, Association for Computing Machinery, New York, NY, USA (2019)