Verification Approaches for Learning-Enabled Autonomous Cyber–Physical Systems

Hoang-Dung Tran
University of Nebraska–Lincoln, Lincoln, NE, USA

Taylor T. Johnson
Vanderbilt University, Nashville, TN, USA

Weiming Xiang
Augusta University, Augusta, GA, USA

Editor's notes:
Neural network control systems are often at the heart of autonomous systems. The authors classify existing verification methods for these systems and advocate the necessity of integrating verification techniques in the training process to enhance robustness.
—Selma Saidi, TU Dortmund

Artificial intelligence (AI) is in a renaissance, and AI methods, such as machine learning (ML), are now at a level of accuracy and performance to be competitive or better than humans for many tasks. Deep neural networks (DNNs), in particular, are increasingly effective at recognition and classification tasks. For instance, much of the sensing, estimation, and fusion of data that enables applications such as autonomous driving [1], aircraft collision avoidance [2], and other autonomous cyber–physical systems (CPSs) [3] increasingly relies on DNNs and related ML techniques. However, this progress comes at a significant risk when these methods are deployed in operational safety-critical systems, especially those without direct human supervision. Notably, it has been observed that neural networks can react in unexpected and incorrect ways to even slight perturbations of their inputs [4]. Therefore, there is a need for methods beyond testing [5] that can provide formal guarantees about the behavioral properties and specifications of autonomous CPS with learning-enabled components (LECs), especially for the purpose of safety assurance [6]. There are two main streams of verification for intelligent systems with ML components. The first one focuses on the correctness of only the learning components while the second one aims at proving the correctness of the whole system in which the learning-based component interacts with the physical dynamics of the system. As one of the most important and prevalent AI/ML techniques, this article specifically surveys the current state-of-the-art for safety verification of autonomous CPS with neural network controllers.

Artificial neural networks are used in systems that introduce ML components to resolve complex problems. This can be attributed to the impressive ability of neural networks to approximate complex functions as shown by the universal approximation theorem [7]. Neural networks are trained over a finite amount of input and output data, and are expected

Digital Object Identifier 10.1109/MDAT.2020.3015712
Date of publication: 12 August 2020; date of current version: 9 February 2022.

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

to generalize the said data and produce desirable outputs for the given inputs and even for previously unseen inputs. The data-driven nature and lack of efficient methods for analysis of neural networks leads to, in most cases, the treatment of neural networks as black boxes with no assurance in safety. However, due to the rapid development AI-inspired applications, neural networks have recently been deployed in several safety-critical systems, such as autonomous systems [1], [3], and aircraft collision avoidance procedures [2]. Regrettably, it has been observed that neural networks can react in unexpected and incorrect ways to even slight perturbations of their inputs [4]. Thus, methods are needed that can provide formal guarantees about the behavioral properties of neural networks, especially for the purpose of safety assurance [6].[1]

Verification of neural networks

Verifying neural networks is a difficult problem, and it has been demonstrated that validating even simple properties about their behavior is NP-complete [10]. The intuition of many approaches is shown in Figure 1. The difficulties encountered in verification mainly arise from the presence of activation functions and complex structures (i.e., an interconnected group of nodes, inspired by biological neurons) of neural networks. Moreover, neural networks are large-scale, nonlinear, nonconvex, and often incomprehensible to humans so that traditional verification tools are not applicable for neural networks. The action of a neuron is described by the activation function in the form of $y_i = f\left(\sum_{j=1}^{n} w_{ij}x_j + \theta_i\right)$, where x_j is the jth input of the ith neuron, w_{ij} is the weight from the jth input to the ith neuron, θ_i is called the bias of the ith neuron, y_i is the output of the ith neuron, and $f(\cdot)$ is the activation function. Typically, the activation function is either the rectified linear unit (ReLU), logistic sigmoid, hyperbolic tangent, the exponential linear unit, or another linear function. In general, existing methods for neural network verification can be categorized into geometric (reachability) methods, mixed-integer linear programming (MILP) methods, satisfiability (SAT)-based and SAT modulo theory (SMT)-based methods, optimization-based methods, and others.

[1] This article summarizes an extended survey of work in this area [8]. A related survey in the area is [9]. Source code for many of the methods described are available in our NNV tool (https://github.com/verivital/nnv), and other methods corresponding to [9] are also available (https://github.com/sisl/NeuralVerification.jl/).

Geometric and reachability methods

To circumvent the difficulties brought by the nonlinearities present in the neural networks, many recent results focus on activation functions of the form $f(x) = \max(0, x)$, known as ReLUs. Taking advantage of the piecewise linear feature of ReLUs and considering the input as polyhedra or special classes of polyhedra, such as zonotopes or hyper-rectangles, the verification process can be turned into a sequence of operations on polyhedra. For instance, in [11], the computation process involves standard polytope operations, such as intersection and projection, and all of these can be computed by employing sophisticated computational geometry tools, such as MPT3 [12]. The essence of the approach is to be able to obtain an exact output set with respect to the input set. However, the number of polytopes involved in the computation process increases exponentially with the number of neurons in its worst case performance which makes the method not scalable to neural networks with a large number of neurons. Within the polyhedra computation framework, further developments have been made in the recent tool NNV [13]–[17] by using star sets, an efficient representation for convex sets, to enhance the scalability with respect to polyhedral operations. Due to the parallelizability of the approach, parallel computing techniques can be employed to speed up the computation to some extent.

In the framework of zonotopes, a verification engine for ReLU neural networks called AI^2 was proposed in [18]. In this approach, perturbed inputs and safety specifications are abstracted as zonotopes, and reasoning about their behaviors use operations over zonotopes. The framework AI^2 is capable of handling neural networks of size up to 50,000 neurons and, in particular, their approach has had success dealing with convolutional neural networks (CNNs). Figure 2 shows a representative reachability computation. The star set approach within NNV has also recently been extended to CNNs using an extension of star sets to the computer vision domain called ImageStars [15], applied to large classification networks such as VGG16 and VGG19 that have tens-to-hundreds of millions of parameters [19]. Another special class of polyhedra that are called interval sets or hyper-rectangles is also considered for verification problems. These interval-based methods perform reachability analysis as the propagation of interval sets across hidden layers and eventually derive the output intervals. Specification-guided methods have been developed to provide an adaptive partitioning method for the

input space [20]. By making use of the information of specification, unnecessary partitioning can be avoided so that the computational complexity can be reduced significantly. In [21], an interval symbolic method is developed to compute rigorous bounds for outputs of neural networks. Their approach is easily parallelizable and makes use of symbolic interval analysis to minimize overestimation (conservativeness). The authors implement their approach as part of ReluVal, a system for checking the security properties of ReLU-based neural networks.

MILP methods

The use of binary variables to encode piecewise linear functions is standard in optimization [22]. In [23], the constraints of ReLU functions are encoded as an MILP. Combining output specifications that are expressed in terms of linear programming (LP), the verification problem for output set eventually turns to the feasibility problem of MILP. For layer i, the MILP encoding is given as

$$\begin{aligned} C_i = \Big\{ & x_j^{[i]} \geq W_j^{[i]} x^{[i-1]} + \theta_j^i \\ & x_j^{[i]} \leq W_j^{[i]} x^{[i-1]} + \theta_j^i + M\delta_j^{[i]} \\ & x_j^{[i]} \geq 0, \\ & x_j^{[i]} \leq M\left(1 - \delta_j^{[i]}\right) \mid j = 1, \ldots, \mid L^{[i]} \mid \Big\} \end{aligned} \tag{1}$$

where M is sufficiently large so that it is larger than the maximum possible output at any node. A similar MILP problem is formulated in [24], where the authors conduct a robustness analysis and search for adversarial examples in ReLU neural networks. It is well known that MILP is an NP-hard problem and, Dutta et al. [25] and [26] elucidate significant efforts for solving MILP problems efficiently to make the approach scalable. Their methods combine MILP solvers with a local search yielding a more efficient solver for range estimation problems of ReLU neural networks than several other approaches. Basically, a local search is conducted using a gradient search and then a global search is formulated as MILP. Instead of finding the global optimum directly, it performs the search seeking values greater/smaller than the upper/lower bound obtained in the preceding local search. This is the primary reason for the computational complexity reduction. This MILP-based approach is integrated in a tool called Sherlock [27]. In [28], an MILP encoding scheme is used for a class of neural networks whose input spaces are encoded as binaries. This MILP encoding has a similar flavor to the other encodings present in the research literature for nonbinarized networks. In their framework, since all the inputs are integer values, the real-valued variables can be rounded so that they can be safely removed, resulting in a reformulated integer LP (ILP) problem that is smaller in comparison to the original MILP encoding. With the ILP encoding, an SAT solver is utilized to reason about the behavior of a binarized neural network of hundreds of neurons.

Satisfiability and SMT methods

In [10], an SMT solver called Reluplex is developed. An algorithm, that stems from the Simplex algorithm for linear functions, for ReLU functions is proposed. Due to the piecewise linear feature of ReLU functions, each node is divided into two nodes. Thus, in their formulation, each node consists of a forward-facing and backward-facing node. If the ReLU semantics are not satisfied, two additional update functions are given to fix the mismatching pairs. Thus, the search process is similar to the Simplex algorithm that pivots and updates the basic and nonbasic variables with the addition of a fixing process for ReLU activation pairs. This method is applied on a DNN implementation of a next-generation airborne collision avoidance system for unmanned aircraft (ACAS-X), which has been used as a benchmark for a number of successive works. Scheibler et al. [29] used bounded model checking (BMC) to create formulas that are solved using the SMT-solver iSAT3, which is able to deal with transcendental functions, such as exp and cos (that exist in various activation functions) that frequently appear in neural network controllers and plant models. Although the verification framework is rigorously developed, the verification problem suffers scalability barriers due to the curse of dimensionality and state–space explosion problems. An approach for finding adversarial inputs using SMT solvers that relies on a layer-by-layer analysis is presented in [30]. The work focuses on the robustness of a neural network where safety is defined in terms of classification invariance within a small neighborhood of one individual input. An exhaustive search of the region is conducted by employing discretization and propagating the analysis layer by layer. In a similar manner, a recent paper, proposed by Ruan et al. [31], generalizes the local

robustness criterion into a global notion on a set of test examples.

An early software tool in this area, called Planet, was developed based on the MILP verification approaches [32]. This LP-based framework combine SAT solving and linear over-approximation of piecewise linear functions to verify ReLU neural networks against convex specifications. Given the output of a ReLU denoted by d and the input $c \in [l, u]$, the relationship between c and d can be approximated by the linear constraints $d \geq 0, d \geq c$, and $d \geq u\,((c-l)/(u-l))$. Based on the LP problem formulation, additional heuristic algorithms were developed to detect infeasibility and imply phase inference faster. Pulina and Tacchella [33] presented an abstraction-refinement and SMT-based tool for verifying feed-forward neural networks. Their scheme is based on encoding the network into a Boolean satisfaction problem over linear arithmetic constraints.

Other optimization-based methods

As some of the earliest papers for neural network verification, in [34] and [35], a piecewise-linearization of the nonlinear activation functions is used to reason about their behavior. In this framework, the authors replace the activation functions with piecewise constant approximations and use the bounded model checker hybrid satisfiability (HySAT) [36] to analyze various properties. The authors highlight the difficulty of scaling this technique and, currently, are only able to tackle small networks with at most 20 hidden nodes.

In [37], a simulation-based approach was developed, which used a finite number of simulations/computations to estimate the reachable set of multilayer neural networks in a general form. Despite this success, the approach lacks the ability to resolve the reachable set computation problem for neural networks that are large-scale, nonconvex, and nonlinear. Still, simulation-based approaches, like the one developed in [37], present a plausibly practical and efficient way of reasoning about neural network behavior. The critical step in improving simulation-based approaches is bridging the gap between finitely many simulations and the essentially infinite number of inputs that exist in the continuity set. A critical concept that is introduced in the work is called maximal sensitivity, which measures the maximal deviation of outputs for a set of inputs suffering disturbances in a bounded cell. The output set of the neural network can be over-approximated by the union of a finite number of reachtubes computed using a union of individual cells that cover the input set. Thus, verification of a network can be done by checking the existence of intersections of the estimated reachable set and safety regions. This approach has been extended to allow for the reachable set estimation and verification of nonlinear autoregressive-moving average (NARMA) models in the form of neural networks [38] as well as closed-loop system verification with the help of the state-of-the-art reachability tool for hybrid systems dealing with the plant dynamics [39].

In a recent result [40], an improved simulation-guided method is developed to reduce computational complexity. Unnecessary input partitions are avoided as the corresponding partition behaviors upon input space are guided by simulations instead of uniform partition. In particular, it is applicable to a variety of neural networks regardless of the specific form of the activation functions. Given a neural network, there is a tradeoff between the precision of the reachable set estimation and the number of simulations used to execute the procedure. In addition, since the approach executes in a layer-by-layer manner, the approximation error will accumulate as the number of layers present in the network increases. In this case, more simulations are required at the expense of increasing the computational cost. A novel approach for neural network verification based on optimization duality has been developed [41]. The verification problem is posed as an optimization problem that tries to find the largest violation of a property related to the output of the network.

Other methods

There exists a rich literature of other methods for neural network verification [8], [9], but we highlight a few. A comparison of the verification approaches mentioned above can be found in [42]. Additionally, the authors present a novel approach for neural network verification called branch and bound optimization. This approach adds one more layer behind the output layer $cy - b$ to represent the linear property $cy > b$ that we wish to verify. If $cy - b > 0$, it means that the property is satisfied, otherwise it is unsatisfiable. Thus, the verification problem is converted into a computation of the minimum or maximum value of the output of the neural network. By treating the neural network as a nonlinear function,

model-free optimization methods are utilized to find optimal solutions. To have a global optimum, the input space is also discretized into subregions. This approach is not only applicable to ReLU neural networks, but the model-free method allows the approach to be applied to neural networks with more general activation functions. However, despite its generalization capabilities, in the model-free framework, there is no guarantee that the algorithm will converge to a solution.

Cheng et al. [43] studied the verification of binarizied neural networks (BNNs). The forward propagation of input signals is reduced to bit arithmetic. The authors argue that the verification of BNNs can be reduced to hardware verification and represents a more scalable problem than traditional neural network verification. A randomized approach for rigorously verifying neural networks in safety-critical applications has been developed [44]. In an effort to mitigate challenges related to the curse of dimensionality, the authors make use of Monte Carlo methods to estimate the probability of neural network failure. However, although Monte Carlo methods are more efficient than methods that deterministically search through hyper-rectangular input spaces, they are probabilistic in nature. The authors further demonstrate that although the number of samples needed to guarantee this may be large, it is not as prohibitive as other methods.

In addition to neural network verification, there are also results on falsification and testing of neural networks. Several ideas for integrating semantics into adversarial learning have been explored, including a semantic modification space and the use of more detailed information about the outputs produced by ML models [45]. In work by Weng et al. [46], an attack independent robustness metric against adversarial examples for neural networks is described. Their approach converts the robustness analysis into a local Lipschitz constant estimation problem and uses extreme value theory for efficient solving. In [47], an automatic test case generator is presented that leverages real-world changes in driving conditions like rain, fog, lighting conditions, etc. The tool, called DeepTest, systematically explores different parts of the DNN logic by generating test inputs that maximize the number of activated neurons. An improved version of the tool, called DeepXplore, is proposed in [48], which is the first efficient whitebox testing framework for large-scale deep learning systems.

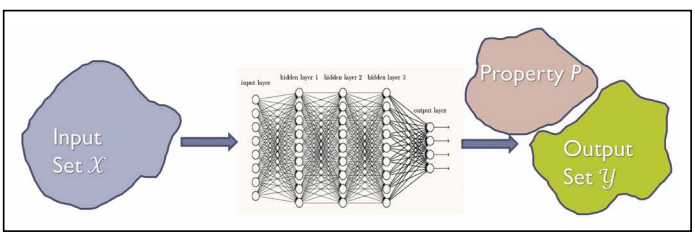

Figure 1. Illustration of neural network reachability, where the output reachable set of a mathematical function $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ representing the neural network's behavior under a set of inputs $\mathcal{X}$ is defined and computed in an exact or over-approximative manner. If $F(\mathcal{X}) = \mathcal{Y} \subseteq \neg\mathcal{P}$, then safety property $\neg P$ holds, while if $P \cap \mathcal{Y} \neq \emptyset$, then unsafe states may be reached.

Verification and fasification of neural network control systems

Verification and falsification of feedback neural network control systems (NNCSs) have become an emerging research topic recently. Unlike verification of a neural network where the specifications of interest are usually defined as predicates over the outputs of the network, in NNCS, the specification are usually defined based on the states of the plant controlled by a neural network controller. Notably, the behavior of the whole feedback control system depends not only on the behavior of the neural network controller but also the system's physical dynamics which is usually described in terms of ordinary differential equations (ODEs). The interaction between the nonlinear neural network controller and the physical dynamics makes the behavior of the whole system complicated and difficult to analyze. To overcome this challenge, several methods have been proposed recently to verify system-level safety properties of NNCS with *feedforward neural network controllers*.

The polyhedron-based approach [11] has been extended for safety verification of NNCS with *linear and discrete* dynamics [49]. Recently, a hybridization approach has been proposed in the Verisig tool [50] that transforms an NNCS to an equivalent nonlinear hybrid system that can be verified using Flow* [55], a verification tool for nonlinear hybrid systems. This approach applies for neural network

Table 1. Reachability and bounded-model checking approaches for NNCS verification.

Approaches	Plant Dynamics	Discrete/Continuous	Activation Function
Polyhedron-based [49]	Linear	Discrete	ReLU
Verisig [50]	Linear, Nonlinear	Discrete, Continuous	Sigmoid, Tanh
SMC-based [51]	Linear	Discrete	ReLU
Sherlock [52]	Linear, Nonlinear	Discrete, Continuous	ReLU
NNV [14], [16], [53]	Linear, Nonlinear	Discrete, Continuous	ReLU, Satlin, Sigmoid, Tanh
ReachNN [54]	Nonlinear	Continuous	ReLU, Sigmoid, Tanh

controllers with smooth activation functions, such as sigmoid and hyperbolic tangent (Tanh). Sound and complete satisfiability modulo convex (SMC)-based approaches for formal verification of NNCS have been developed, in which the closed-loop NNCS with *linear and discrete* dynamics is encoded as monotone SMC formulas that were formally verified by SMC decision procedures [51]. In [52], a new abstraction method has been proposed for NNCS verification in which an "local" Taylor model over-approximation of neural network controller was obtained and integrated in Flow* [55] to compute a tight over-approximation reachable set of NNCS. The advantage of this method is that it is fast and scalable and more importantly, it is proposed to reduce significantly over-approximation errors in reachable set computation process. In [54], a new reachability approach based on Bernstein polynomials has been proposed to verify NNCS with more general of activation functions. This approach can control the over-approximation error in the analysis; however, the cost of being more accurate is increased computation time.

An extension of the star-based reachability analysis method for neural networks has been implemented in NNV [53] to verify safety properties of NNCS. The star set method can deal with different activation functions, such as ReLU, Satlin, Sigmoid, and Tanh, as well as different types of dynamics, i.e., linear or nonlinear in discrete or continuous time domains. The star set method can perform exact and complete analysis of NNCS with linear discrete dynamical plants and neural network controllers with ReLU/Satlin activation functions. The extended star set method has successfully verified safety properties of advanced emergency braking systems (AEBSs) and adaptive cruise control systems (ACCSs), in which the size of the neural network controller ranges from fifty to two hundreds neurons.

A summary of recent verification methods is given in Table 1, which focuses on reachability methods that can provide finite time-horizon guarantees, although there also exist approaches based on barrier certificates (a "continuous" form of the classical inductive invariance proof rule) that may provide infinite time-horizon guarantees [56]. Although verification for NNCS provides sound guarantees for safety, it is usually computationally expensive and suffers from scalability challenges. Importantly, due to scalability limitations, current state-of-art verification techniques cannot deal with NNCS with perception components. In this case, falsification approach plays an important role since it is more scalable and applicable than the verification approaches. Particularly, in [57], a compositional falsification framework for CPS with ML components has been developed. In this framework, a temporal logic falsifier cooperates efficiently with an ML analyzer to find falsifying executions of the system. The effectiveness of the proposed framework was shown via the falsification of AEBS.

Challenges and future directions

Scalability Versus conservativeness: Scalability is still a major challenge for most existing verification techniques. It has been shown that the verification time using exact analysis increases exponentially [10], [14]. Particularly, besides the size of the network, the input set is an important factor affecting the verification time of the exact analysis method. Generally, a large network or a large input set requires more verification time. To improve scalability, a large body of research in neural network verification relies on over-approximation methods. Some recent approaches [58], [59] are optimistic

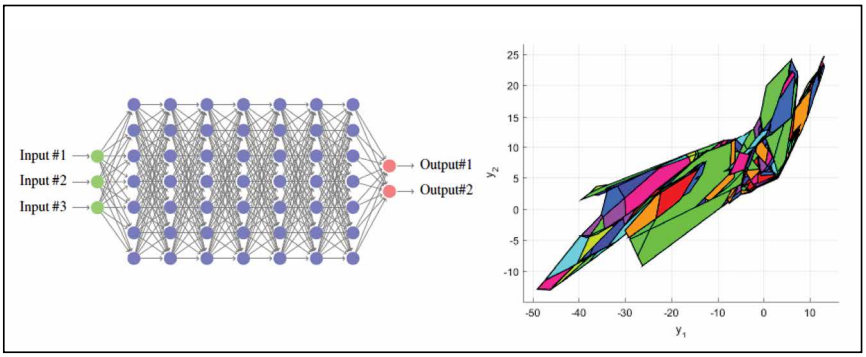

Figure 2. Example output reachable set computation for a neural network with three inputs, two outputs, and seven hidden layers with seven neurons each, where all activation functions are ReLUs and all parameters of the network (weights, biases) are chosen randomly. The input set $I = \{x \mid \|x\|_\infty \leq 1, x \in \mathbb{R}^3\}$ is a cube and convex, while the output set shown is nonconvex, represented as the union of the different colored polygons.

about the scalability of their methods. However, it has been shown that these methods only can deal with a small input set due to the explosion of over-approximation error in the analysis, which leads to conservative reachable sets [14]. In the future, we believe that new hybrid techniques that can combine the advantages of exact and over-approximate analyses are needed to improve both scalability and conservativeness in neural network verification.

Formal specifications and compositional verification: While a large body of research focuses on verifying neural networks and NNCS, fewer works investigate specification formalization for such systems [60]–[62]. For neural network verification, most current methods investigate safety and robustness properties, which can be specified as input to output relations of neural networks as illustrated in Figure 1 [60], [62]. For NNCS verification, existing approaches deal with safety specifications defined as predicates over the states of the plant model. In the real-world, learning-enabled CPS are complex in which several LECs, such as perception components and neural network controllers, interact with each other and the physical world, such as between a physical plant and its environment.

Defining meaningful system-level specifications for the whole system is relatively straightforward (such as collision avoidance), but the implications and constraints such system-level specifications place on LECs, especially those for perception, is nontrivial and needs to be investigated deeply. New specification languages for learning-enabled CPS are crucial to formally define the behavior of the systems and their subcomponents, and equally important, is defining libraries of specifications for meaningful perception problems, such as classification, semantic segmentation, and object detection/localization. One promising direction is to utilize hyperproperties for specifying robustness to adversarial perturbations [60], [63]. A further challenge, particularly related to perception, is not only in defining specifications, but in evaluating specifications with respect to meaningful environmental scenarios and data. This challenge is fundamentally different than the typical approach for verification of closed-loop systems, where a plant model generates new inputs for a controller, and instead requires verification with respect to prerecorded environmental data (such as images/video) or generation thereof. This is partly because it is unreasonable to expect formal models for the environment in which an NNCS operates, and at best, generative models such as GANs and realistic simulators may exist, beyond prerecorded real-world data. Altogether, it is unclear under what circumstances compositional specification and verification for learning-enabled CPS are achievable, such as by verifying individual LECs and attempting to compose guarantees of individual components into system-level guarantees [64].

Runtime verification for NNCS: Existing verification techniques for NNCS primarily operate offline

and are to be performed during the design-time of a system. In practice, it is useful to have techniques that can monitor, if not verify, NNCS specifications online. Based on the online verification information, a system can perform some intelligent actions to avoid upcoming difficult or catastrophic circumstances, such as hitting an obstacle or colliding with another system, for instance, with Simplex architecture approaches [65].

Robust and safe learning: All techniques surveyed in this article deal with an existing network or NNCS. In the future, new learning methods that integrate verification techniques in the training process to enhance the robustness of a network or the safety of an NNCS are essential for applying neural networks in safety-critical applications, such as in recent approaches for safe reinforcement learning.

Benchmarking and standardization: A major limiting factor in the development of this area is a lack of standardization for formal models and specifications. There are several ongoing initiatives that aim to address this shortcoming to enable easier, fairer, and more scientific comparisons between the existing verification methods, as well as future ones. Due to this lack of standardization, this article does not make specific claims relating to which methods are most appropriate or scalable, as such answers are not yet known. For the open-loop verification problem (the "Verification of neural networks" section), the verification of neural networks (VNNs) workshop hosted the first competition on neural network verification (VNN-COMP) in 2020.[2] For the closed-loop verification problem (e.g., for NNCS as in the "Verification and fasification of neural network control systems" section), the Applied Verification of Continuous and Hybrid Systems (ARCH) workshop hosted the first AINNCS category verification competition in 2019 [66]. Other efforts, such as standardization of models [e.g., in open neural network exchange (ONNX)[3]], specifications, etc., are emerging, such as through the usage of HyST hybrid automata for ARCH-COMP [67], and the development of the VNN-LIB,[4] an effort like that of SMT-LIB [68] for satisfiability and SMT problems.

This article has surveyed recent approaches for verifying ML components, specifically neural networks, that are crucial to enabling autonomy in CPS, but that suffer from well-known robustness problems. Numerous avenues for future work exist, ranging from runtime verification and assurance approaches to assure autonomy during system operation, to expanding the types of LECs beyond feedforward neural networks for which most existing verification approaches target. ■

[2] https://sites.google.com/view/vnn20/
[3] https://onnx.ai/
[4] http://www.vnnlib.org/

Acknowledgments

This work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Contract FA9550-18-1-0122 and in part by the Defense Advanced Research Projects Agency (DARPA) under Contract FA8750-18-C-0089. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR or DARPA.

■ References

[1] M. Bojarski et al., "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: http://arxiv.org/abs/1604.07316

[2] K. D. Julian et al., "Policy compression for aircraft collision avoidance systems," in *Proc. IEEE/AIAA 35th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2016, pp. 1–10.

[3] K. D. Julian and M. J. Kochenderfer, "Neural network guidance for UAVs," in *Proc. AIAA Guid., Navigat., Control Conf. (GNC)*, Jan. 2017.

[4] C. Szegedy et al., "Intriguing properties of neural networks," *CoRR*, vol. abs/1312.6199, pp. 1–10, Dec. 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[5] J. M. Zhang et al., "Machine learning testing: Survey, landscapes and horizons," *IEEE Trans. Softw. Eng.*, early access, Feb. 2020, doi: 10.1109/TSE.2019.2962027.

[6] F. Leofante et al., "Automated verification of neural networks: Advances, challenges and perspectives," May 2018, *arXiv:1805.09938*. [Online]. Available: https://arxiv.org/abs/1805.09938

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[8] W. Xiang et al., "Verification for machine learning, autonomy, and neural networks survey," *CoRR*,

vol. abs/1810.01989, pp. 1–51, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.01989

[9] C. Liu et al., "Algorithms for verifying deep neural networks," *CoRR*, vol. abs/1903.06758, pp. 1–76, Mar. 2019. [Online]. Available: http://arxiv.org/abs/1903.06758

[10] G. Katz et al., "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. Int. Conf. Comput. Aided Verification*. Berlin, Germany: Springer-Verlag, 2017, pp. 97–117.

[11] W. Xiang, H.-D. Tran, and T. T. Johnson, "Reachable set computation and safety verification for neural networks with ReLU activations," 2017, *arXiv:1712.08163*. [Online]. Available: http://arxiv.org/abs/1712.08163

[12] M. Herceg et al., "Multi-parametric toolbox 3.0," in *Proc. Eur. Control Conf.*, Zürich, Switzerland, Jul. 2013, pp. 502–510. [Online]. Available: http://control.ee.ethz.ch/?mpt

[13] H.-D. Tran et al., "Parallelizable reachability analysis algorithms for feed-forward neural networks," in *Proc. IEEE/ACM 7th Int. Conf. Formal Methods Softw. Eng. (FormaliSE)*, May 2019, pp. 51–60.

[14] H.-D. Tran et al., "Star-based reachability analysis of deep neural networks," in *Proc. 23rd Int. Symp. Formal Methods (FM)*, Oct. 2019, pp. 670–686.

[15] H.-D. Tran et al., "Verification of deep convolutional neural networks using imagestars," in *Proc. 32nd Int. Conf. Comput.-Aided Verification (CAV)*, 2020, pp. 18–42.

[16] H.-D. Tran et al., "NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems," in *Proc. 32nd Int. Conf. Comput.-Aided Verification (CAV)*, 2020, pp. 3–17.

[17] S. Bak et al., "Improved geometric path enumeration for verifying ReLU neural networks," in *Proc. 32nd Int. Conf. Comput.-Aided Verification (CAV)*, Jul. 2020, pp. 66–96.

[18] T. Gehr et al., "AI2: Safety and robustness certification of neural networks with abstract interpretation," in *Proc. IEEE Symp. Secur. Privacy (SP)*, vol. 39, May 2018, pp. 3–18.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.

[20] W. Xiang, H.-D. Tran, and T. T. Johnson, "Specification-guided safety verification for feedforward neural networks," 2018, *arXiv:1812.06161*. [Online]. Available: http://arxiv.org/abs/1812.06161

[21] S. Wang et al., "Formal security analysis of neural networks using symbolic intervals," in *Proc. 27th USENIX Secur. Symp. (USENIX Security)*, 2018, pp. 1599–1614.

[22] R. J. Vanderbei, *Linear Programming: Foundations & Extensions*, 2nd ed. Berlin, Germany: Springer-Verlag, 2001.

[23] A. Lomuscio and L. Maganti, "An approach to reachability analysis for feed-forward ReLU neural networks," *CoRR*, vol. abs/1706.07351, pp. 1–10, Jun. 2017. [Online]. Available: http://arxiv.org/abs/1706.07351

[24] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," 2017, *arXiv:1711.07356*. [Online]. Available: http://arxiv.org/abs/1711.07356

[25] S. Dutta et al., "Output range analysis for deep neural networks," 2017, *arXiv:1709.09130*. [Online]. Available: http://arxiv.org/abs/1709.09130

[26] S. Dutta et al., "Output range analysis for deep feedforward neural networks," in *NASA Formal Methods*, A. Dutle, C. Muñoz, and A. Narkawicz, Eds. Cham, Switzerland: Springer-Verlag, 2018, pp. 121–138.

[27] S. Dutta et al., "Sherlock—A tool for verification of neural network feedback systems: Demo abstract," in *Proc. 22nd ACM Int. Conf. Hybrid Systems: Comput. Control*, Apr. 2019, pp. 262–263.

[28] N. Narodytska et al., "Verifying properties of binarized deep neural networks," 2017, *arXiv:1709.06662*. [Online]. Available: http://arxiv.org/abs/1709.06662

[29] K. Scheibler et al., "Towards Verification of Artificial Neural Networks," in *Proc. MBMV*, 2015, pp. 30–40.

[30] X. Huang et al., "Safety verification of deep neural networks," *CoRR*, vol. abs/1610.06940, pp. 1–31, May 2016. [Online]. Available: http://arxiv.org/abs/1610.06940

[31] W. Ruan et al., "Global robustness evaluation of deep neural networks with provable guarantees for the L_0 norm," Apr. 2018, *arXiv:1804.05805*. [Online]. Available: https://arxiv.org/abs/1804.05805

[32] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," *CoRR*, vol. abs/1705.01320, pp. 1–15, May 2017. [Online]. Available: http://arxiv.org/abs/1705.01320

[33] L. Pulina and A. Tacchella, "NeVer: A tool for artificial neural networks verification," *Ann. Math. Artif. Intell.*, vol. 62, nos. 3–4, pp. 403–425, Jul. 2011.

[34] L. Pulina and A. Tacchella, "An abstraction-refinement approach to verification of artificial neural networks," in *Computer Aided Verification*, T. Touili, B. Cook, and P.

Jackson, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 243–257.

[35] L. Pulina and A. Tacchella, "Challenging SMT solvers to verify neural networks," *AI Commun.*, vol. 25, no. 2, pp. 117–135, 2012.

[36] M. Fränzle and C. Herde, "HySAT: An efficient proof engine for bounded model checking of hybrid systems," *Formal Methods Syst. Design*, vol. 30, no. 3, pp. 179–198, Apr. 2007.

[37] W. Xiang, H.-D. Tran, and T. T. Johnson, "Output reachable set estimation and verification for multilayer neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5777–5783, Nov. 2018.

[38] W. Xiang et al., "Reachable set estimation and verification for neural network models of nonlinear dynamic systems," in *Safe, Autonomous and Intelligent Vehicles*. Berlin, Germany: Springer-Verlag, 2019, pp. 123–144.

[39] W. Xiang and T. T. Johnson, "Reachability analysis and safety verification for neural network control systems," 2018, *arXiv:1805.09944*. [Online]. Available: http://arxiv.org/abs/1805.09944

[40] W. Xiang et al., "Reachable set estimation for neural network control systems: A simulation-guided approach," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, May 14, 2020, doi: 10.1109/TNNLS.2020.2991090.

[41] D. Krishnamurthy et al., "A dual approach to scalable verification of deep networks," 2018, *arXiv:1803.06567*. [Online]. Available: http://arxiv.org/abs/1803.06567

[42] R. Bunel et al., "A unified view of piecewise linear neural network verification," 2017, *arXiv:1711.00455*. [Online]. Available: http://arxiv.org/abs/1711.00455

[43] C.-H. Cheng et al., "Verification of binarized neural networks via inter-neuron factoring," *CoRR*, vol. abs/1710.03107, pp. 1–15, Oct. 2017. [Online]. Available: http://arxiv.org/abs/1710.03107

[44] R. R. Zakrzewski, "Randomized approach to verification of neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 4, Jul. 2004, pp. 2819–2824.

[45] T. Dreossi, S. Jha, and S. A. Seshia, "Semantic adversarial deep learning," Apr. 2018, *arXiv:1804.07045*. [Online]. Available: http://arxiv.org/abs/1804.07045

[46] T.-W. Weng et al., "Evaluating the robustness of neural networks: An extreme value theory approach," Jan. 2018, *arXiv:1801.10578*. [Online]. Available: http://arxiv.org/abs/1801.10578

[47] Y. Tian et al., "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," *CoRR*, vol. abs/1708.08559, pp. 1–12, Aug. 2017. [Online]. Available: http://arxiv.org/abs/1708.08559

[48] K. Pei et al., "DeepXplore: Automated whitebox testing of deep learning systems," *CoRR*, vol. abs/1705.06640, pp. 1–18, May 2017. [Online]. Available: http://arxiv.org/abs/1705.06640

[49] W. Xiang et al., "Reachable set estimation and safety verification for piecewise linear systems with neural network controllers," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 1574–1579.

[50] R. Ivanov et al., "Verisig: Verifying safety properties of hybrid systems with neural network controllers," in *Proc. 22nd ACM Int. Conf. Hybrid Systems: Comput. Control*, Apr. 2019, pp. 169–178.

[51] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems," in *Proc. 22nd ACM Int. Conf. Hybrid Syst. Comput. Control*, Apr. 2019, pp. 147–156.

[52] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *Proc. 22nd ACM Int. Conf. Hybrid Syst. Comput. Control*, Apr. 2019, pp. 157–168.

[53] H.-D. Tran et al., "Safety verification of cyber-physical systems with reinforcement learning control," in *Proc. ACM SIGBED Int. Conf. Embedded Softw. (EMSOFT)*, Oct. 2019, Art. no. 105.

[54] C. Huang et al., "ReachNN: Reachability analysis of neural-network controlled systems," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5s, pp. 1–22, Oct. 2019.

[55] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Proc. Int. Conf. Comput. Aided Verification*, 2013, pp. 258–263.

[56] C. E. Tuncali et al., "Reasoning about safety of learning-enabled components in autonomous cyber-physical systems," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[57] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," in *Proc. NASA Formal Methods Symp*, 2017, pp. 357–372.

[58] G. Singh et al., "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, pp. 1–30, Jan. 2019.

[59] G. Singh et al., "Fast and effective robustness certification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10802–10813.

[60] S. A. Seshia et al., "Formal specification for deep neural networks," in *Proc. Int. Symp. Autom. Technol. Verification Anal.*, 2018, pp. 20–34.

[61] D. J. Fremont et al., "Scenic: A language for scenario specification and scene generation," in *Proc. 40th ACM SIGPLAN Conf. Program. Lang. Design Implement.*, 2019, pp. 63–78.

[62] T. Dreossi et al., "A formalization of robustness for deep neural networks," 2019, *arXiv:1903.10033*. [Online]. Available: http://arxiv.org/abs/1903.10033

[63] L. V. Nguyen et al., "Hyperproperties of real-valued signals," in *Proc. 15th ACM-IEEE Int. Conf. Formal Methods Models for Syst. Design (MEMOCODE)*, Sep. 2017, Art. no. 104113.

[64] S. A. Seshia, "Compositional verification without compositional specification for learning-based systems," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2017-164, 2017, pp. 1–8.

[65] S. Bak et al., "Real-time reachability for verified simplex design," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 2014, pp. 138–148.

[66] D. M. Lopez et al., "ARCH-COMP19 category report: Artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants," in *Proc. 6th Int. Nat. Workshop Appl. Verification Continuous Hybrid Syst. (ARCH EPiC)*, vol. 61, G. Frehse and M. Althoff, Eds. Manchester, U.K.: EasyChair, Apr. 2019, pp. 103–119.

[67] S. Bak, S. Bogomolov, and T. T. Johnson, "HYST: A source transformation and translation tool for hybrid automaton models," in *Proc. 18th Int. Conf. Hybrid Syst., Comput. Control (HSCC)*, 2015, Art. no. 128133.

[68] C. Barrett, A. Stump, and C. Tinelli, "The SMT-LIB standard—Version 2.0," in *Proc. 8th Int. Workshop Satisfiability Modulo Theories (SMT)*, Edinburgh, U.K., 2010, pp. 1–85.

Hoang-Dung Tran is currently an Assistant Professor with the Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE, USA. His research interests include in formal verification of autonomous cyber–physical systems with learning-enabled components, safe artificial intelligence, hybrid, and switching systems, distributed control systems, robust control, stability analysis of nonlinear control systems, and networked control systems. Tran has a PhD in computer science from Vanderbilt University, Nashville, TN, USA.

Weiming Xiang is an Assistant Professor with the School of Computer and Cyber Sciences, Augusta University, Augusta, GA, USA. His research interests include developing formal synthesis and verification techniques and software tools for cyber–physical systems (CPSs), especially focusing on on formal methods on safety, security, and reliability of learning-enabled CPSs. He is also broadly interested in methods and applications across CPS domains, such as control synthesis, stability analysis, reachable set computation, hybrid systems, power and energy, transportation, fuzzy logic, and neural networks. Xiang has a PhD from Southwest Jiaotong University, Chengdu, China (2014). He is a Senior Member of IEEE.

Taylor T. Johnson is an Assistant Professor of electrical engineering and computer science with Vanderbilt University, Nashville, TN, USA. His research interests include developing algorithmic techniques and software tools to improve the reliability of cyber–physical systems. Johnson has an MSc and a PhD in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA. He is a Member of IEEE.

■ Direct questions and comments about this article to Taylor T. Johnson, Department of Electrical and Computer Science, Vanderbilt University, Nashville, TN 37212 USA; taylor.johnson@vanderbilt.edu.