



Formal Verification Techniques for Vision-Based Autonomous Systems – A Survey

Sayan Mitra¹, Corina Păsăreanu^{2,3(✉)}, Pavithra Prabhakar⁴, Sanjit A. Seshia⁵,
Ravi Mangal⁶, Yangge Li¹, Christopher Watson⁷, Divya Gopinath³,
and Huafeng Yu⁸

¹ University of Illinois, Urbana-Champaign, Urbana, USA

² Carnegie Mellon University, Pittsburgh, USA
pcorina@cmu.edu

³ KBR, NASA Ames, Mountain View, USA

⁴ Kansas State University, Manhattan, KS, USA

⁵ University of California, Berkeley, Berkeley, USA

⁶ Colorado State University, Fort Collins, USA

⁷ University of Pennsylvania, Philadelphia, USA

⁸ Department of Transportation, Washington, D.C., USA

Abstract. Providing safety guarantees for autonomous systems is difficult as these systems operate in complex environments that require the use of learning-enabled components, such as deep neural networks (DNNs), for visual perception. DNNs are hard to formally verify due to their size (they can have billions of parameters), lack of formal specifications, and sensitivity to slight changes in the surrounding environment. Furthermore, the high-dimensional inputs to the DNNs come from sensors such as high-fidelity cameras that are themselves complex and hard to model – they bear complex relationships to the system states and are subject to random environmental perturbations. We present a survey of verification techniques that aim to provide quantitative or qualitative formal guarantees for such autonomous systems.

1 Introduction

Complex autonomous systems, such as autonomous aircraft taxiing systems [26, 49] and autonomous cars [29, 32, 42, 73], need to perceive and reason about their environments using high-dimensional data streams (such as images) generated by rich sensors (such as cameras). Machine learned components, especially deep neural networks (DNNs), are particularly capable of the required high-dimensional inference and classification; hence they are increasingly used for perception in these systems. Formal analysis of the safety of these systems is highly desirable due to their safety-critical operational settings and the error-prone nature of learned components. However, in practice this is very challenging because of the

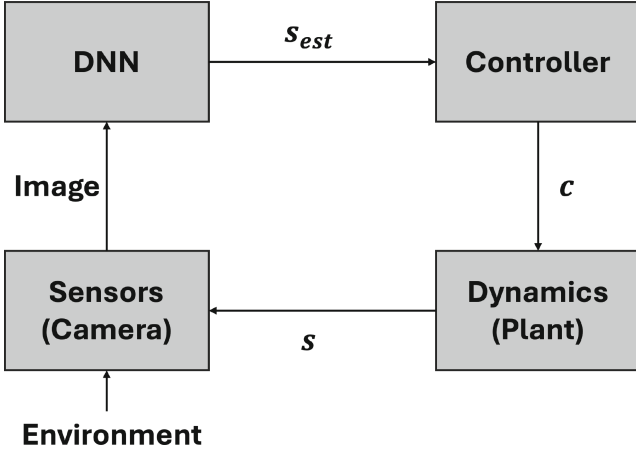


Fig. 1. Example of autonomous system using a DNN for visual perception

complexity of the system components, including the high complexity of the neural networks (which may have millions or billions of parameters), the complexity of the camera capture process, and the random and hard to characterize nature of the environment in which the system operates (i.e., the world itself).

Typical assurance techniques for such autonomous systems include simulations and in-field testing, which have the advantage of being applicable to full-scale systems, but do not provide guarantees, unlike formal verification. In this work, we survey recently-proposed techniques that use formal methods with the goal of providing formal system-level guarantees.

While there may be other papers and surveys on techniques and challenges for safe artificial intelligent (AI) systems, see [70, 84] for prominent examples, we specifically focus here on formal verification techniques for autonomous systems that incorporate DNNs for perception from high-dimensional data. To the best of our knowledge, such a survey has not been performed before. It is our hope that this survey can serve as a quick reference to the techniques in this rapidly evolving space and the guarantees that they provide. To scope this work, we focus on verification techniques, while related approaches, such as synthesis of safe vision-based controllers, are not covered in depth in this study.

The rest of the paper is organized as follows. In the next section we present a schematic description of an autonomous system with a DNN for visual perception and discuss properties for such a system. Section 3 gives details about the surveyed verification techniques. Section 4 surveys related verification techniques that also target autonomous systems, albeit with DNNs that have low-dimensional input space, while Sect. 5 outlines some future challenges and conclusion.

2 A Vision-Based Autonomous System

Let us consider an autonomous system consisting of four components; systems with more components can be treated similarly. The system contains a

Perception component (i.e., a DNN) which processes images (img) and produces estimates (s_{est}) of the system state, a *Controller* that sends commands (c) to the physical system (the plant) being controlled in order to maneuver it based on these state estimates, the *Dynamics* modeling the evolution of the actual physical system states (s) in response to control signals, and the *Sensor*, e.g., a high-definition camera, that captures images representing the current state of the system and its surrounding environment. There may be other sensors (radar, LiDAR, GPS) that we do not consider here for simplicity. The environment may have its own dynamics which may or may not depend on the system under verification and may also be subject to environment perturbations, which we ignore here for simplicity.

Although simple, the type of system we consider resembles (semi-)autonomous mechanisms that are already deployed in practice, such as adaptive cruise controllers and lane-keeping assist systems, which similarly use a DNN for visual perception to provide guidance to the software controlling electrical and mechanical subsystems of modern vehicles. An example of such a system designed for autonomous taxiing of airplanes on taxiways is illustrated in Fig. 1. For this system, the *state* captures the position of the airplane on the surface and the role of the DNN is to estimate this position from images captured by a camera placed on the airplane.

System-Level Properties. We aim to check that the overall system satisfies a system-level property P . In the autonomous taxiing example, an example property is that *the airplane shall never leave the taxiway*, which can be written as $|cte| \leq d$ meters, where cte is the cross-track error (one of the values that the DNN aims to estimate) and d is the dimension of the runway [63]. See also previous works [78, 79] for several examples of specifications for autonomous vehicles. Such properties can be expressed in various flavors of *temporal logics* depending on the formalism used to model the system. For instance, the above property can be encoded in PCTL [14] as follows.

$$P = ?[F(|cte| > d)]$$

Here $P = ?$ indicates that we want to calculate the probability that eventually (F) the system reaches an error state.

In order to check this property, one could run many simulations using a high-fidelity simulator such as CARLA [20] or X-Plane [54]. However, best-effort simulation alone may not be enough to achieve the high degree of confidence in the correctness of the system necessary for deployment in a safety-critical setting. The goal is therefore to formally verify the property. To this end, one needs to formally model the autonomous system operating in the environment (denoted *System*) and to check that $System \models P$ holds with a provable guarantee.

Verification Challenges. Formally verifying *System* presents serious scalability challenges, even ignoring the learning-enabled aspect, since the conventional components (*Controller* and *Dynamics* in our simplified case) can be quite complex; nevertheless they can be tackled with previous formal verifica-

Table 1. Summary of surveyed techniques.

Technique	Modelling for Perception	Analysis (Techniques and Tools)	Environment	System-level Guarantees	Case-studies
[63] (Sect. 3.1)	Probabilistic Abstraction of Env, Sensor, DNN	Probabilistic Model Checking PRISM, STORM, Incorporates DNN Verification	Stochastic	Probabilistic with Confidence Intervals	Airplane Center-line Tracking
[7] (Sect. 3.1)	Probabilistic Abstraction of Env, Sensor, DNN	Probabilistic Model Checking STORM	Stochastic	Probabilistic	Urban Driving Car-Pedestrian Scenario
[39, 55] (Sect. 3.2)	Non-deterministic Approximation (contract) of Env, Sensor, DNN	Constrained Optimization Problem Reachability Analysis Data-driven construction of contracts	Stochastic	Non-probabilistic	Lane Keeping, Vision-based Landing, Formation flight systems
[64] (Sect. 3.3)	Non-deterministic Abstraction of Env, Sensor, DNN	Model Checking Extended with Assumption Generation	Non-deterministic	Non-probabilistic	Airplane Center-line Tracking
[16, 33, 68] (Sect. 3.4)	Geometric, Mathematical Models of Env and Sensor	Decision Procedures, Z3, PPL AI/VERIF DNN Verification	Deterministic	Non-probabilistic	Auto Drone Vision-based Landing
[4, 10, 51] (Sect. 3.5)	GAN Model for Env and Sensor	DNN Verification Reachability Analysis Continuous-time Reachability analysis VerifyGAN	Stochastic	Non-probabilistic	Airplane Center-line Tracking
[81] (Sect. 3.5)	Canonical Env Model with Data-driven Noise Models, Hybrid System Model of DNN	Hybrid Model Checking Verisig	Deterministic	Non-probabilistic	Mountain Car, F1/10 Auto Racing Car
[23] (Sect. 3.6)	Scenic program for Env Connected to Simulation	Statistical Model Checking VerifAI	Stochastic	Statistical	Autonomous Driving, Airplane Taxiing, UAVs, Indoor Robotics
[76] (Sect. 3.6)	Scenic program for Env	Run-time Monitoring VerifAI	Stochastic	Run-Time Safety	Autonomous Driving, Airplane Taxiing

tion techniques (such as software, probabilistic, or hybrid model checking) possibly involving abstraction to reduce the size of the state space. The perception components, including the DNN and the camera, make the scalability problem extremely severe. A DNN can have billions of parameters, which precludes formal verification for most realistic, state-of-the-art networks. Furthermore, the high-dimensional inputs to the DNN come from sensors—themselves complex and hard to model—and bear complex relationships to the system states while also being subject to random perturbations from the external environment that are difficult to model precisely.

In the next section we describe formal methods techniques that address the main challenge of reasoning about the perception which involves: the complex DNN, the high-definition cameras, and the changes from the environment.

3 Verification Techniques

Table 1 briefly summarizes the main techniques (with the corresponding citations) that we survey in this section. We briefly describe the modeling approach taken by different techniques when tackling the perception challenges, the particular verification techniques and tools that are employed to conduct the verification of the autonomous system, the assumptions made about the environment conditions, and the system-level guarantees that are provided by each of the surveyed techniques. We also highlight the various case studies reported in the respective papers.

3.1 Verification with Probabilistic Abstractions for Perception

We begin our survey with verification techniques that are *probabilistic*, which are perhaps the most natural since the camera images capturing the state of

the world are subject to randomness due to the environment; further DNNs are learnt from data and are not guaranteed to be 100% accurate. To render the complexities introduced by sensor noise and DNN-based perception tractable, [6] and [63] build a *probabilistic abstraction* of the perception components. At a high level, the probabilistic abstraction maps each (discrete) system state to a distribution over the state estimates that the DNN-based perception component would output.

These distributions are based on *confusion matrices* computed for the DNN-based perception component on a representative dataset. A confusion matrix is a commonly used performance metric for classification problems which succinctly summarizes the accuracy of a network for each output class. These accuracy values can be translated directly into probabilities on transitions connecting actual and estimated system states.

Importantly, the representative dataset must accurately reflect the operational conditions of the system. Composing the probabilistic abstraction with a known model of the system dynamics and controller yields a discrete-time Markov model that can then be analyzed with a probabilistic model checker such as PRISM [53] or STORM [37].

As the probabilities are estimated based on empirical data, and thus are subject to error, [63] computes *confidence intervals* in addition to point estimates for these probabilities to strengthen the soundness of the analysis. Further, [63] shows how to leverage local, DNN-specific analyses as run-time guards to filter out misbehaving inputs and increase the safety of the overall system. The use of DNN-specific analyses builds on the authors' prior work [11, 12] that uses probabilistic abstractions of perception to synthesize controllers with safety guarantees.

In the special case where the perception module is an object detector [7] introduces *distance-parameterized* and *proposition-labeled* confusion matrices to help construct a suitable probabilistic abstraction. Each of these augments the traditional notion of a confusion matrix with class divisions that are not explicitly predicted by the object detector but may be correlated with the object detector's performance or system-level behavior of the closed-loop system.

Also related is the verification approach of PACTI [44] which provides an algebra of assume-guarantee contracts along with algorithms to infer component contracts given a system-level contract. For example, PACTI can take the *quotient* of a system-level contract and the controller's contract to obtain a contract that the perception component needs to uphold. Since strong (non-probabilistic) guarantees may be too stringent in realistic settings, PACTI allows the user to supply a minimum satisfaction probability alongside the system level contract. In this case, PACTI computes a contract for the perception component that is strong enough to guarantee the minimum satisfaction probability while occasionally allowing errors that lead to system-level violations. The probabilistic perception contract is formalized in terms of a proposition-labeled confusion matrix. To check whether the contract is upheld, one can compare the contract

against a probabilistic abstraction of perception constructed using the techniques surveyed earlier in this section.

3.2 Symbolic Model Checking with Perception Contracts

An alternative approach for verification of learning-enabled autonomous systems centers around **perception contracts** that serve as the specifications for the machine learning (ML) models (e.g., DNNs for perception) that can be used to verify system-level properties [5, 39]. The methods was used to assure the safety of lane-keeping systems for simulated vehicles and agricultural robots, and more recently, vision-based automated landing [55, 71] and formation flight systems [38].

As outlined in Sect. 2, the ML model in the autonomous system serves as a state estimator for the plant state. Suppose s_{est} is the DNN’s output of the ground truth estimate and s is the ground truth state. The correctness of the DNN module can be specified *in terms of some relation* $M(s, s_{est})$ relating the quality of the estimate s_{est} with respect to ground truth s . For example, although it is unrealistic to formally specify the “lane markings” in terms of color-density of pixels in an image, it is easy to specify that a vision-based lane detection system should, ultimately, estimate the distance s between the center of the ego car and the center of the lane. The correctness specification of such a lane detector could then be stated in terms of the raw error $s_{est} - s$, and one can just choose $M(s, s_{est})$ to be $= |s - s_{est}|^2 - r \leq 0$ with r being the perception error bound. In auto-landing, adaptive cruise control, pedestrian detection and avoidance, ML models are similarly used to estimate pose of the ego aircraft (relative to runway), separation between vehicles, distance and intent of pedestrians, etc. Getting access to ground truth-labeled data is straightforward in simulations, and albeit more expensive it is also reasonably accessible for instrumented and controlled real world experiments.

It is noteworthy that perception contracts are *not* input-output contracts for DNNs. They cannot hence be formally verified (say using DNN verification techniques). Rather, they are specifications that relate the *implicit ground truth* associated with an image to the ground truth estimated by the DNN.

In this sense they are similar to the probabilistic maps developed in [63] and the nondeterministic maps used in [64] which map ground-truth states to (sets of) estimated states. As we shall see below, perception contracts are constructed so that they verify the system-level safety property—i.e., the car does not go off the road—with M substituting the actual perception components in the autonomous system. Furthermore, unlike a traditional uniform estimation error bounding $|s - s_{est}|$, M can be a much more expressive, nonuniform, nonconvex, algorithmically synthesized relationship between s and s_{est} .

The key benefit of perception contracts is that we can stand on the foundations laid by decades of prior work on verification of control system and hybrid systems that *do not* use ML components. If we assume for a moment that the state estimation s_{est} provided by the DNN conforms to the perception contract, then the safety assurance problem becomes amenable, to a battery of

existing techniques from formal verification, control theory, and program analysis [3, 9, 57, 59, 65–67, 87]. This is achieved with the following assume-guarantee strategy:

- **Conformance (A). Synthesize a perception contract:** Construct a *perception contract* (PC) for the ML module, using input/output perception data and available system information, that captures the estimation quality via a relation $M(s, s_{est})$ under *all* operating environments Env .
- **Correctness (B). Prove safety under perception contract:** Formally verify the composition of the above contract M and the rest of the system (Dynamics, Controller,...) against the system safety requirement using one of the above techniques.

It is conceptually easier to think of two steps: **conformance (A)** and **correctness (B)** sequentially, however, more precise contracts are obtained by solving for M jointly. In [5, 39] the contracts are computed using constrained optimization problems that always satisfies correctness and maximizes conformance with respect to collected data. In [55], hybrid reachability analysis (Verse tool [56]) is used for checking the correctness condition of the constructed contracts.

If we set $M(s, s_{est}) = |s - s_{est}|^2 - r \leq 0$, then the above approach reduces to certifying the safety with gross error bounds. This idea works in relatively stationary environments [15, 19], but is too restrictive as environment variations increase. In contrast, perception contracts are more flexible and are typically represented by logical formulas, decision trees, simpler neural networks, and with other expressive data structures.

Perception contracts have been used to discover safe *Operational Design Domains (ODDs)*, that is, the environmental conditions under which the autonomous system can be expected to work safely. Start with a large set of environmental conditions E , compute the corresponding perception contract M_E , if M_E satisfies the conformance and correctness conditions, then it can be declared as a safe environment, otherwise, shrink E to a smaller environment, and iteratively repeat the process. For guaranteed convergence, the shrink step should keep at least one environmental condition (nominal environment) in which the system works safely. In [55], this procedure is used to find ranges of ambient lighting conditions and sun angles that are safe for automated landing.

We close this section by remarking that perception contracts are useful for offline verification of ML-enabled autonomous systems, but they are limited for online or runtime monitoring. It is tempting to think of $M(s)$ as a set of estimated states that are provably safe, and therefore, could be used to raise an alarm or take corrective actions when the actual s_{est} at runtime is outside this set. However, to use $M(s)$ we need the actual ground truth state at runtime which is of course not available. In [71], the authors address this issue by computing the preimage of the perception M^{-1} contract and using that for online recovery.

3.3 Compositional Verification with Assumption Generation for Perception

Another approach builds upon the technique in [63] described earlier, to provide *qualitative*, i.e., non-probabilistic, guarantees for system-level safety properties. This follow-up work [64] describes a compositional verification approach which uses a form of *abductive reasoning*, where the autonomous system, modeled as a labeled transition system, is analyzed *in the absence of the DNN, the camera, and the environment*, using instead a nondeterministic abstraction mapping actual states to all the possible estimated states, and thus assuming worst-case behaviour for perception. The approach then automatically derives conditions, in the form of *assumptions*, on DNN behaviour that guarantee the safety of the overall system.

They build on previous work on automated assume-guarantee compositional verification [8, 31, 61], to automatically generate these *assumptions* in the form of labeled transition systems, encoding sequences of DNN predictions that guarantee system-level safety. The assumptions are the *weakest* in the sense that they characterize the output sequences of all the possible DNNs that, plugged into the autonomous system, satisfy the property. They further propose to mine the assumptions to extract local properties on DNN behavior, which in turn can be used for the separate testing and training of the DNNs. These local properties relate the ground truth with the estimates and are thus similar to the perception contracts described above.

The approach can be applied at different development phases for the autonomous system. At design time, the approach can be used to uncover problems in the autonomous system *before* deployment. The automatically generated assumptions and the extracted local properties can be seen as *safety requirements* for the development of neural networks. At run time, the assumptions can be deployed as safety monitors over the DNN outputs to *guarantee the safety behaviour* of the overall system.

A key challenge in building the monitors is that the system does not have access to the ground truth states, but only to the estimated states as output by the DNN and therefore subject to error. The way they solve this challenge is by restricting the alphabet of the automatically generated assumptions to only refer to the estimated states. Further, although the work provides qualitative guarantees, the work also leverage quantitative, probabilistic model checking (PRISM) to quantify the permissiveness of the generated assumptions.

In general, early works that apply compositional verification to closed-loop systems with DNN-based perception start by defining component-level contracts, then apply compositional reasoning to yield system-level guarantees. For example, the 2018 white paper [62] first uses clustering to discover regions of the perception component’s input space that yield predictable behavior, then uses Reluplex [50] to prove input-output properties for these regions, and finally encapsulates these properties as an input-output contract on the perception component. Although the resulting contract is guaranteed to hold, it is not necessarily sufficient to prove system-level safety. Another approach is to handcraft a contract

before verifying that it is upheld by the perception component, as is done in the *robustness contracts* framework [60]. Although robustness contracts offer an expressive formalism for contract-based design, it may be difficult for a human to design contracts that are both amenable to existing verification tools and are strong enough to ensure safety.

3.4 Verification with Mathematical Models of the Cameras and the Environment

Considering more accurate models of perception camera and environment can provide a higher level of assurance about the correctness and safety. Instead of abstracting away perception components, the following works [16, 33, 68] incorporate (geometric) mathematical models of the camera and the environment alongside the perception DNN and the conventional components (such as controller and vehicle dynamics) in their analysis.

NNLander-VeriF [16, 68] considers a geometric model of the pinhole camera to capture the relation between the system states (e.g., position of the aircraft) and the images processed by the DNN that represents a combined perception/control module. The camera model depends of the environment, that is, the runway parameters in the aircraft landing case study considered in the paper. To address the scalability issues of the closed-loop verification, the proposed approach remodels the camera as a neural network and constructs a single neural network by augmenting the DNNs of the camera model and the given perception/control module, and use neural network model-checkers to verify this augmented neural network in the closed-loop verification.

The camera model in [16, 68] is specific to an environment, requiring remodeling and retraining for different environment models. In [33], the authors consider a more decoupled approach by modeling the camera and the environment separately. A synthetic 3-D environment E (created using a 3-D-design tool like Blender) is considered where the environment is represented as a collection of triangles that represent the triangulated faces of objects in the environment (e.g. obstacles which the vehicle must avoid when traveling). The camera model C specifies the parameters of the camera, including the canvas details and the focal length of the lens. A central modeling of the paper is a mathematical function $img_C(E, p)$ which takes as input the environment E , camera model parameters C and the camera/vehicle position p and outputs the image captured by the pinhole camera on its canvas from that position. The paper then uses these models to do a symbolic reachability analysis and falsification.

One of the important steps in the symbolic verification process is to compute the set of output images of the camera for a given set of positions. This is accomplished using the notion of an image invariant region, which corresponds to a set of positions in the 3-D space from which the camera captures identical images. More precisely, the given region - set of positions - is split into image invariant regions, and each image invariant region and the corresponding image is propagated through the loop independently. The computation of the image invariant regions of a given set of positions is computationally expensive as the number

of such regions can be huge (upper bounded by the number of different possible images on the canvas). Hence, an abstraction-based approach to speed-up the propagation through the loop is proposed. The experimental results point to the benefits in proving safety and falsification for an autonomous vehicle with several synthetic road environments with up to 800 triangle edges in the synthetic environment; however, scalability still remains a challenge as the technique already takes about an hour for the largest environments considered. An abstraction-refinement algorithm based on an abstract data structure for representing a set of images called interval image is explored in [34] which substantially scales the verification.

3.5 Verification Using Generative Models for Perception

Another related work [51] proposes a method to address the challenges of modeling the perception by training a generative adversarial network (GAN) to map states to plausible input images. Concatenating the generator network with the control network results in a network with a low-dimensional input space, which allows for the use of existing closed-loop verification tools to obtain formal guarantees on the performance of image-based controllers. This approach is applied to provide safety guarantees for an image-based neural network controller for an autonomous aircraft taxi problem. The resulting guarantees are with respect to the set of input images modeled by the generator network, and so a recall metric is provided to evaluate how well the generator captures the space of plausible images. The resulting closed-loop systems are analyzed with traditional (non-probabilistic) formal verification techniques. The recent work [10] builds on this approach by composing the system dynamics with the generative model and unrolling multiple steps of this composition into a large neural network to reduce the overapproximation that occurs at each timestep. An orthogonal approach to mitigating compounding overapproximation is to use a continuous time model; another recent work [4] builds on the approach of [51] by incorporating a piecewise linear approximation of the neural controller to permit the use of continuous-time reachability techniques.

In contrast to the aforementioned techniques that learn a monolithic noise model, [81] takes a compositional approach. The authors first obtain a canonical environment model (from first principles or from a simulator) that represents perfect perception under nominal conditions. Then, they use real-world data to learn a noise model that augments the canonical model with realistic observations. The authors distinguish two kinds of noise: *benign noise* (e.g. image blur) that can be accurately modeled by a generative model and *adversarial noise* (e.g. reflected LiDAR rays) for which they train instead a classifier that predicts which component of the observation should be perturbed. Once a noise models is learned, it can be composed with the canonical environment model, resulting in a hybrid system that can be analyzed using Verisig [48]. The resulting safety guarantees are only meaningful if the data driven noise models faithfully model real-world conditions; in future work, the authors plan to incorporate a PAC bound on the modeling error into the verification problem.

3.6 Compositional Statistical Model Checking, Simulation-Based Verification, and Run-Time Verification

An important class of formal verification techniques are those that integrate with simulation and can be performed not just at design time, but also at run time. Simulation presents important advantages: high fidelity simulators contain more details of complex dynamical systems than the high-level models used in verification, simulation can be used on full-scale software and hardware implementations of autonomous systems, and simulations are widely used even for highly complicated systems. The main disadvantage is that, in general, simulation does not provide formal guarantees. Additionally, traditional simulation is not coupled with high-level formal models; traditional simulation models (e.g., Matlab/Simulink code) are executable models that do not capture the stochasticity and non-determinism present in operating environments. In this section, we describe multiple inter-related contributions that address the limitations of traditional simulation and show how to perform scalable, compositional formal analysis that builds upon simulation methods.

Stochastic Formal Modeling with Probabilistic Programming: Learning-enabled autonomous systems operate in environments about which there is typically a great deal of uncertainty. Indeed, learning is introduced to reason about and handle this uncertainty. One needs formal modeling languages that can capture this uncertainty as well as the stochastic behavior of the objects and agents in the environment. As argued in [70], probabilistic programming languages are well-suited to modeling environments of complex learning-enabled autonomous systems. One such language is Scenic [27,28]. Scenic is a formal modeling language, with well-defined semantics, and can express not only common formalisms such as Markov chains and Markov Decision Processes, but also more complex stochastic models. Importantly, for learning-enabled vision-based autonomy, Scenic can also be interfaced to simulators such as CARLA [20], X-Plane [54], and Webots [83] which implement high-fidelity models of system dynamics and sensors. A basic unit of modeling in Scenic is a *scenario*, a distribution over configurations of objects and agents in the physical world and their dynamics. Scenic allows one to build up complex scenarios from simpler ones using compositional modeling constructs.

Compositional Falsification and Abstract Perception Models: Simulation-based falsification has found widespread industrial adoption in conventional cyber-physical systems. However, the complexity of neural networks used in vision-based autonomy makes these systems beyond the reach of conventional falsification tools. Additional methods are needed to scale up to industrial-scale systems. One approach is to use compositional reasoning coupled with abstractions of perception components. The first work on this topic was by Dreossi et al. [21,22], showing how to perform simulation-based falsification of temporal logic specifications by first abstracting away the DNN based perception compo-

ment with a simpler function approximation and then computing a set of environment states where the output of the perception component, if wrong, can flip the outcome of the safety property from safe to unsafe. This set is then used to generate a pre-condition over the input space of the DNN with respect to which a localized adversarial analysis of the perception component can be performed. This method was demonstrated to scale up to falsify autonomous driving systems using neural networks with hundreds of millions of parameters. In subsequent work, Ghosh et al. [30] gave a counterexample-guided approach to generating abstractions of perception components and using those abstractions to provide formal guarantees on control of autonomous systems. The abstractions take the form of non-deterministic over-approximate relations between the state of the environment being perceived and the output of the perception component (classification, object detection, distance estimation, etc.). In essence, these are a variant of perception contracts as introduced earlier in this paper. One can also perform correct-by-construction controller synthesis with respect to such abstract models of perception components, where one synthesizes a controller that keeps the system safe even when the perception component makes an error.

Temporally Compositional Statistical Model Checking (SMC) and Falsification: Another source of complexity for simulation-based verification arises from long-horizon scenarios, where even running a single simulation can take hours or even days. To deal with this temporal dimension of complexity, we can take a model-based approach. Specifically, one can use Scenic to model a long-horizon scenario as a composition (sequential, parallel, random, etc.) of simpler scenarios. Yalcinkaya et al. [85] show how this compositional structure of a Scenic program can be exploited to split up a statistical model checking (or falsification) problem for a Markovian safety specification into several simpler SMC (or falsification) sub-problems. Simulations of shorter sub-scenarios can be performed in parallel and the results can be composed to provide the same guarantee as one would achieve with monolithic SMC or falsification. These algorithms are implemented in VerifAI [23], an open-source toolkit accompanying Scenic that can perform several verification, debugging, and synthesis tasks including SMC and falsification. Scenic and VerifAI have been together used on several industrial-scale autonomous systems, including a DNN-based autonomous taxiing system [26] and a full-scale autonomous vehicle evaluated both in simulation and on the road [29].

Runtime Monitoring of Operational Design Domains: Scenic is also useful for run-time verification, which is crucial for vision-based autonomy as real-world environments can deviate from the models assumed at design time. In this context, an important notion is that of an *operational design domain*, a specification of operating environments in which the autonomous system is designed to perform safely. Scenic provides a formalism for ODDs; i.e., a Scenic program can precisely represent the ODD of a system [75, 76]. Importantly, ODDs must be *monitorable* at run time: it is important to be able to tell at run time whether

you are within the ODD or might exit the ODD. However, not all variables of the Scenic model can be monitored accurately at run time; for example, one may not know ground truth values of positions or other attributes of environment vehicles. Torfah et al. [76] provide an algorithmic method to learn a runtime monitor from a Scenic program over a subset of monitorable variables so as to predictively monitor whether a system is in its ODD or not. This capability has been demonstrated on autonomous driving systems and avionics systems including for automated taxiing and landing.

4 Other Related Techniques

Formal proofs of closed-loop safety have been obtained for systems with low-dimensional sensor readings in the absence of sensor noise [24, 25, 40, 41, 45, 46, 48, 52, 69, 72, 77, 82]. For example, Verisig [46, 48] models both the system dynamics *and* the neural component as a hybrid system, permitting the use of the existing hybrid verification tool Flow* [13]. Verisig has successfully verified safety and liveness for simulated autonomous racecars that use LiDAR sensors and an end-to-end neural controller with sigmoid activations [47] but has not been scaled up to handle the convolutional neural networks commonly used to process visual input. Support for CNNs is provided by the NNV 2.0 tool [58], however additional treatment of hard-to-model sensor behavior is needed to apply any of the aforementioned techniques to realistic settings with high dimensional perception.

An alternate approach for scaling closed-loop safety proofs is to construct (or learn) inductive invariants in the form of control barrier functions (CBFs) [2, 3, 17]. One only needs to prove that, over a single step of the system dynamics, the invariant is inductive and implies safety. This is no small task in settings with realistic perception, as safety and system dynamics are typically defined in terms of the system’s low-dimensional state. To bridge the gap between LiDAR observations and system-level safety, LOCUS [18] learns an *observation-space CBF* defined in terms of observations rather than state estimates. This requires a model of system and sensor dynamics that can predict the observation one step into the future. Such single-step foresight may be practical for LiDAR observations, but is much harder for high dimensional camera observations. To address this challenge, the successor work [74] uses a neural radiance field (NeRF) to provide visual foresight. The authors of [74] identify both reducing computational overhead and incorporating bounds on the NeRF’s reconstruction error into the CBFs (similarly to [19]) as directions for future work.

Measurement-robust control barrier functions [15, 19] enforce safety in the presence of state estimates that suffer from bounded approximation error. This approximation error is represented as a function that can be learned from labeled data to capture the characteristics of the sensor and perception. To provide probabilistic safety guarantees in the presence of imperfect perception, the authors of [86] use conformal prediction [80] to quantify estimate uncertainty during CBF construction. Conformal prediction allows the authors to use a representative calibration dataset to augment the imperfect perception map’s predictions with regions that contain the true state of the system with high probability.

The authors demonstrate that their approach can guarantee 75% safety for a simulated racecar that receives noisy LiDAR observations; the technique could also be applied to camera observations assuming a readily available source of calibration data (e.g., from simulation).

In the context of safe reinforcement learning, shielding [1] augments the controller with a *shield* that disables actions that might lead to safety violations. Although shielding typically assumes direct access to the symbolic state of the environment, VSRL [43] treats the case of safe reinforcement learning from high-dimensional visual inputs. To make enforcing safety tractable, VSRL assumes access to an object detector that localizes the position of each safety-relevant object to within an ϵ Euclidean distance of the object’s actual position.

5 Conclusion and Future Directions

We surveyed techniques for the verification of autonomous systems that use DNNs for perception. While these techniques open the door to analyzing autonomous systems with state-of-the-art DNNs, they can still suffer from the well-known scalability issues associated with formal verification techniques, such as model checking. We believe these can be addressed via judicious use of abstraction and compositionality.

Since the abstractions of perception, such as the probabilistic abstractions or the perception contracts, are often constructed with respect to an offline dataset of states and observations, the safety assurances they afford are only valid when the deployed system stays within a narrow operational design domain. This is especially important for bridging the sim-to-real gap, as the labeled data needed to build the abstractions is readily available in simulation but expensive to collect in the real world. Furthermore, the simulation environment may not be reflective of the actual deployment. Future work could investigate ways to combine large amounts of simulation data with small amounts of labelled real-world data to practically construct useful abstractions for perception. Another interesting direction would be to integrate probabilistic abstractions of perception with out-of-distribution detectors during deployment to qualify safety guarantees in novel scenarios.

Some of the techniques discussed attempt a mathematical modeling of the camera and the environment. Models are, in general, only approximations of the real-world; specifically, the models need to incorporate the uncertainties in the environment, the camera rendering process and the dynamics. Developing frameworks that consider uncertainties, possibly through stochastic models, and corresponding analysis techniques will be an important future direction to explore.

Existing techniques consider a fully trained neural network for certain components of the closed-loop such as perception and control. In practice, these networks are continuously evolving due to retraining as more and more data become available. Techniques that deal with evolving neural network components in a vision-based closed-loop system is an important open question. Specifically, one needs to investigate how proofs can be transferred between different versions of the neural networks in the closed-loop. Some preliminary work in this direction

has been explored in the context of traditional closed-loop systems with evolving neural networks [35, 36].

Many techniques surveyed treat stateless models of perception that directly map single-timestep observations to state estimates. Future work could investigate ways to build probabilistic abstractions of perception that incorporate the time-series nature of observations and perception systems.

References

1. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
2. Alur, R.: Formal verification of hybrid systems. In: Proceedings of the Ninth ACM International Conference on Embedded Software, pp. 273–278 (2011)
3. Ames, A.D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., Tabuada, P.: Control barrier functions: Theory and applications. In: 2019 18th European Control Conference (ECC), pp. 3420–3431 (2019). <https://doi.org/10.23919/ECC.2019.8796030>
4. ArjomandBigdeli, A., Mata, A., Bak, S.: Verification of neural network control systems in continuous time. In: 7th Symposium on AI Verification (SAIV) (2024)
5. Astorga, A., Hsieh, C., Madhusudan, P., Mitra, S.: Perception contracts for safety of ML-enabled systems. Proc. ACM Program. Lang. **7**(OOPSLA2), 2196–2223 (2023). <https://doi.org/10.1145/3622875>
6. Badithela, A., Wongpiromsarn, T., Murray, R.M.: Leveraging classification metrics for quantitative system-level analysis with temporal logic specifications. In: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 564–571. IEEE (2021)
7. Badithela, A., Wongpiromsarn, T., Murray, R.M.: Evaluation metrics of object detection for quantitative system-level analysis of safety-critical autonomous systems. In: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8651–8658. IEEE (2023)
8. Bogomolov, S., et al.: Assume-guarantee abstraction refinement meets hybrid systems. In: Yahav, E. (ed.) Hardware and Software: Verification and Testing - 10th International Haifa Verification Conference, HVC 2014, Haifa, Israel, 18–20 November 2014, Proceedings. LNCS, vol. 8855, pp. 116–131. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13338-6_10
9. Bullo, F.: Contraction Theory for Dynamical Systems, 1.1 edn. Kindle Direct Publishing (2023)
10. Cai, F., Fan, C., Bak, S.: Scalable surrogate verification of image-based neural network control systems using composition and unrolling (2024)
11. Calinescu, R., Imrie, C., Mangal, R., Păsăreanu, C., Santana, M.A., Vázquez, G.: Discrete-event controller synthesis for autonomous systems with deep-learning perception components. arXiv preprint [arXiv:2202.03360](https://arxiv.org/abs/2202.03360) (2022)
12. Calinescu, R., et al.: Controller synthesis for autonomous systems with deep-learning perception components. IEEE Trans. Softw. Eng. 1–22 (2024). <https://doi.org/10.1109/TSE.2024.3385378>
13. Chen, X., Abraham, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_18

14. Ciesinski, F., Größer, M.: On probabilistic computation tree logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) *Validation of Stochastic Systems*. LNCS, vol. 2925, pp. 147–188. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24611-4_5
15. Cosner, R.K., Singletary, A.W., Taylor, A.J., Molnar, T.G., Bouman, K.L., Ames, A.D.: Measurement-robust control barrier functions: certainty in safety with uncertainty in state. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6286–6291 (2021). <https://doi.org/10.1109/IROS51168.2021.9636584>
16. Cruz, U.S., Shoukry, Y.: Certified vision-based state estimation for autonomous landing systems using reachability analysis. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 6052–6057 (2023). <https://doi.org/10.1109/CDC49753.2023.10384107>
17. Dawson, C., Gao, S., Fan, C.: Safe control with learned certificates: a survey of neural Lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762* (2022)
18. Dawson, C., Lowenkamp, B., Goff, D., Fan, C.: Learning safe, generalizable perception-based hybrid control with certificates. *IEEE Robot. Autom. Lett.* **7**(2), 1904–1911 (2022)
19. Dean, S., Taylor, A., Cosner, R., Recht, B., Ames, A.: Guaranteeing safety of learned perception modules via measurement-robust control barrier functions. In: *Conference on Robot Learning*, pp. 654–670. PMLR (2021)
20. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16 (2017)
21. Dreossi, T., Donze, A., Seshia, S.A.: Compositional falsification of cyber-physical systems with machine learning components. In: *Proceedings of the NASA Formal Methods Conference (NFM)*, pp. 357–372, May 2017
22. Dreossi, T., Donzé, A., Seshia, S.A.: Compositional falsification of cyber-physical systems with machine learning components. *J. Autom. Reason.* **63**(4), 1031–1053 (2019)
23. Dreossi, T., et al.: VerifAI: a toolkit for the formal design and analysis of artificial intelligence-based systems. In: *31st International Conference on Computer Aided Verification (CAV)*, July 2019
24. Dutta, S., Chen, X., Sankaranarayanan, S.: Reachability analysis for neural feedback systems using regressive polynomial rule inference. In: *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019*, pp. 157–168. Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3302504.3311807>
25. Fan, J., Huang, C., Chen, X., Li, W., Zhu, Q.: ReachNN*: a tool for reachability analysis of neural-network controlled systems. In: Hung, D.V., Sokolsky, O. (eds.) *Automated Technology for Verification and Analysis*, pp. 537–542. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_30
26. Fremont, D.J., Chiu, J., Margineantu, D.D., Osipychiev, D., Seshia, S.A.: Formal analysis and redesign of a neural network-based aircraft taxiing system with VerifAI. In: *32nd International Conference on Computer Aided Verification (CAV)*, July 2020
27. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., Seshia, S.A.: Scenic: a language for scenario specification and scene generation. In: *Proceedings of the 40th Annual ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, June 2019

28. Fremont, D.J., et al.: Scenic: a language for scenario specification and data generation. *Mach. Learn. J.* (2022)
29. Fremont, D.J., et al.: Formal scenario-based testing of autonomous vehicles: from simulation to the real world. In: 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC), September 2020
30. Ghosh, S., Pant, Y.V., Ravanbakhsh, H., Seshia, S.A.: Counterexample-guided synthesis of perception models and control. In: American Control Conference (ACC), pp. 3447–3454. IEEE (2021)
31. Giannakopoulou, D., Pasareanu, C.S., Barringer, H.: Assumption generation for software component verification. In: 17th IEEE International Conference on Automated Software Engineering (ASE 2002), 23–27 September 2002, Edinburgh, Scotland, UK, pp. 3–12. IEEE Computer Society (2002). <https://doi.org/10.1109/ASE.2002.1114984>
32. Grigorescu, S.M., Trasnea, B., Cocias, T.T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. *CoRR* **abs/1910.07738** (2019)
33. Habeeb, P., Deka, N., D’Souza, D., Lodaya, K., Prabhakar, P.: Verification of camera-based autonomous systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **42**(10), 3450–3463 (2023). <https://doi.org/10.1109/TCAD.2023.3240131>
34. Habeeb, P., D’Souza, D., Lodaya, K., Prabhakar, P.: Interval image abstraction for verification of camera-based autonomous systems. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2024)
35. Habeeb, P., Gupta, L., Prabhakar, P.: Approximate conformance checking for closed-loop systems with neural network controllers. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* (2024)
36. Habeeb, P., Prabhakar, P.: Approximate conformance verification of deep neural networks. *NASA Formal Methods* (2024)
37. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transfer* **24**(4), 589–610 (2022)
38. Hsieh, C., Koh, Y., Li, Y., Mitra, S.: Assuring safety of vision-based swarm formation control. In: American Control Conference (ACC) (2024)
39. Hsieh, C., Li, Y., Sun, D., Joshi, K., Misailovic, S., Mitra, S.: Verifying controllers with vision-based perception using safe approximate abstractions. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **41**(11), 4205–4216 (2022). <https://doi.org/10.1109/TCAD.2022.3197508>
40. Huang, C., Fan, J., Chen, X., Li, W., Zhu, Q.: POLAR: a polynomial arithmetic framework for verifying neural-network controlled systems. In: Automated Technology for Verification and Analysis: 20th International Symposium, ATVA 2022, Virtual Event, 25–28 October 2022, Proceedings, pp. 414–430. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-19992-9_27
41. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: ReachNN: reachability analysis of neural-network controlled systems. *ACM Trans. Embed. Comput. Syst.* **18**(5s), 1–22 (2019). <https://doi.org/10.1145/3358228>
42. Huang, X., et al.: A survey of safety and trustworthiness of deep neural networks: verification, testing, adversarial attack and defence, and interpretability. *Comput. Sci. Rev.* **37**, 100270 (2020)
43. Hunt, N., Fulton, N., Magliacane, S., Hoang, T.N., Das, S., Solar-Lezama, A.: Verifiably safe exploration for end-to-end reinforcement learning. In: Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control, HSCC 2021. Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3447928.3456653>

44. Incer, I., et al.: Pacti: scaling assume-guarantee reasoning for system analysis and design. arXiv preprint [arXiv:2303.17751](https://arxiv.org/abs/2303.17751) (2023)
45. Ivanov, R., Carpenter, T., Weimer, J., Alur, R., Pappas, G., Lee, I.: Verisig 2.0: verification of neural network controllers using Taylor model preconditioning. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 249–262. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_11
46. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verifying the safety of autonomous systems with neural network controllers. ACM Trans. Embed. Comput. Syst. (TECS) **20**(1), 1–26 (2020)
47. Ivanov, R., Jothimurugan, K., Hsu, S., Vaidya, S., Alur, R., Bastani, O.: Compositional learning and verification of neural network controllers. ACM Trans. Embed. Comput. Syst. (TECS) **20**(5s), 1–26 (2021)
48. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pp. 169–178 (2019)
49. Kadron, I.B., Gopinath, D., Pasareanu, C.S., Yu, H.: Case study: analysis of autonomous center line tracking neural networks. In: Bloem, R., Dimitrova, R., Fan, C., Sharygina, N. (eds.) Software Verification - 13th International Conference, VSTTE 2021, New Haven, CT, USA, 18–19 October 2021, and 14th International Workshop, NSV 2021, Los Angeles, CA, USA, 18–19 July 2021, Revised Selected Papers. LNCS, pp. 104–121. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-95561-8_7
50. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčák, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
51. Katz, S.M., Corso, A.L., Strong, C.A., Kochenderfer, M.J.: Verification of image-based neural network controllers using generative models. J. Aerosp. Inf. Syst. **19**(9), 574–584 (2022)
52. Kochdumper, N., Schilling, C., Althoff, M., Bak, S.: Open- and closed-loop neural network verification using polynomial zonotopes. In: Rozier, K.Y., Chaudhuri, S. (eds.) NASA Formal Methods, pp. 16–36. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33170-1_2
53. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
54. Laminar Research: X-Plane (2023). <https://www.x-plane.com>
55. Li, Y., Yang, B.C., Jia, Y., Zhuang, D., Mitra, S.: Refining perception contracts: case studies in vision-based safe auto-landing (2023)
56. Li, Y., Zhu, H., Braught, K., Shen, K., Mitra, S.: Verse: a Python library for reasoning about multi-agent hybrid system scenarios. In: Enea, C., Lal, A. (eds.) Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, 17–22 July 2023, Proceedings, Part I. LNCS, vol. 13964, pp. 351–364. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37706-8_18
57. Lohmiller, W., Slotine, J.J.E.: On contraction analysis for non-linear systems. Automatica (1998)
58. Lopez, D.M., Choi, S.W., Tran, H.D., Johnson, T.T.: NNV 2.0: the neural network verification tool. In: Enea, C., Lal, A. (eds.) Computer Aided Verification, pp. 397–412. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37703-7_19

59. Mitra, S.: Verifying Cyber-Physical Systems: A Path to Safe Autonomy. The MIT Press, Cambridge (2021)
60. Naik, N., Nuzzo, P.: Robustness contracts for scalable verification of neural network-enabled cyber-physical systems. In: 2020 18th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE), pp. 1–12 (2020). <https://doi.org/10.1109/MEMOCODE51338.2020.9315118>
61. Pasareanu, C.S., Giannakopoulou, D., Bobaru, M.G., Cobleigh, J.M., Barringer, H.: Learning to divide and conquer: applying the I^* algorithm to automate assume-guarantee reasoning. *Formal Methods Syst. Des.* **32**(3), 175–205 (2008). <https://doi.org/10.1007/s10703-008-0049-6>
62. Pasareanu, C.S., Gopinath, D., Yu, H.: Compositional verification for autonomous systems with deep learning components. *CoRR* abs/1810.08303 (2018). <http://arxiv.org/abs/1810.08303>
63. Păsăreanu, C.S., et al.: Closed-loop analysis of vision-based autonomous systems: a case study. In: Enea, C., Lal, A. (eds.) *Computer Aided Verification*, pp. 289–303. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37706-8_15
64. Pasareanu, C.S., Mangal, R., Gopinath, D., Yu, H.: Assumption generation for learning-enabled autonomous systems. In: Katsaros, P., Nenzi, L. (eds.) *Runtime Verification - 23rd International Conference, RV 2023, Thessaloniki, Greece, 3–6 October 2023, Proceedings. LNCS*, vol. 14245, pp. 3–22. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-44267-4_1
65. Podelski, A., Wagner, S.: Model checking of hybrid systems: from reachability towards stability. In: Hespanha, J.P., Tiwari, A. (eds.) *HSCC 2006. LNCS*, vol. 3927, pp. 507–521. Springer, Heidelberg (2006). https://doi.org/10.1007/11730637_38
66. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: Alur, R., Pappas, G.J. (eds.) *HSCC 2004. LNCS*, vol. 2993, pp. 477–492. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24743-2_32
67. Qin, Z., Zhang, K., Chen, Y., Chen, J., Fan, C.: Learning safe multi-agent control with decentralized neural barrier certificates. In: *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=P6_q1BRxY8Q
68. Santa Cruz, U., Shoukry, Y.: NNlander-VerIF: a neural network formal verification framework for vision-based autonomous aircraft landing. In: Deshmukh, J.V., Havelund, K., Perez, I. (eds.) *NASA Formal Methods*, pp. 213–230. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06773-0_11
69. Schilling, C., Forets, M., Guadalupe, S.: Verification of neural-network control systems by integrating Taylor models and zonotopes. *Proc. AAAI Conf. Artif. Intell.* **36**(7), 8169–8177 (2022). <https://doi.org/10.1609/aaai.v36i7.20790>
70. Seshia, S.A., Sadigh, D., Sastry, S.S.: Toward verified artificial intelligence. *Commun. ACM* **65**(7), 46–55 (2022)
71. Sun, D., Yang, B., Mitra, S.: Learning-based inverse perception contracts and applications. In: *International Conference on Robotics and Automation* (2024)
72. Sun, X., Khedr, H., Shoukry, Y.: Formal verification of neural network controlled autonomous systems. In: *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019*, pp. 147–156. Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3302504.3311802>
73. Tabernik, D., Skocaj, D.: Deep learning for large-scale traffic-sign detection and recognition. *CoRR* abs/1904.00649 (2019)

74. Tong, M., Dawson, C., Fan, C.: Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields. arXiv preprint [arXiv:2209.12266](https://arxiv.org/abs/2209.12266) (2022)
75. Torfah, H., Junges, S., Fremont, D.J., Seshia, S.A.: Formal analysis of AI-based autonomy: from modeling to runtime assurance. In: Feng, L., Fisman, D. (eds.) RV 2021. LNCS, vol. 12974, pp. 311–330. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88494-9_19
76. Torfah, H., Xie, C., Junges, S., Vazquez-Chanlatte, M., Seshia, S.A.: Learning monitorable operational design domains for assured autonomy. In: Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA), October 2022
77. Tran, H.D., et al.: NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: Lahiri, S.K., Wang, C. (eds.) Computer Aided Verification, pp. 3–17. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_1
78. Viswanadha, K., et al.: Addressing the IEEE AV test challenge with Scenic and VerifAI. In: IEEE International Conference on Artificial Intelligence Testing (AITest), pp. 136–142. IEEE (2021)
79. Viswanadha, K., Kim, E., Indaheng, F., Fremont, D.J., Seshia, S.A.: Parallel and multi-objective falsification with SCENIC and VERIFAI. In: Feng, L., Fisman, D. (eds.) RV 2021. LNCS, vol. 12974, pp. 265–276. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88494-9_15
80. Vovk, V., Gammerrman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer, Cham (2022). <https://doi.org/10.1007/b106715>
81. Waite, T., Robey, A., Hamed, H., Pappas, G.J., Ivanov, R.: Data-driven modeling and verification of perception-based autonomous systems (2023)
82. Wang, Y., et al.: POLAR-express: efficient and precise formal reachability analysis of neural-network controlled systems. Trans. Comp.-Aided Des. Integr. Circuits Sys. **43**(3), 994–1007 (2023). <https://doi.org/10.1109/TCAD.2023.3331215>
83. Webots: <http://www.cyberbotics.com>, open-source Mobile Robot Simulation Software
84. Wing, J.M.: Trustworthy AI. Commun. ACM **64**(10), 64–71 (2021)
85. Yalcinkaya, B., Torfah, H., Fremont, D.J., Seshia, S.A.: Compositional simulation-based analysis of AI-based autonomous systems for Markovian specifications. In: Katsaros, P., Nenzi, L. (eds.) Runtime Verification - 23rd International Conference, RV 2023, Thessaloniki, Greece, 3–6 October 2023, Proceedings. LNCS, vol. 14245, pp. 191–212. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-44267-4_10
86. Yang, S., Pappas, G.J., Mangharam, R., Lindemann, L.: Safe perception-based control under stochastic sensor uncertainty using conformal prediction. In: 2023 62nd IEEE Conference on Decision and Control (CDC), pp. 6072–6078. IEEE (2023)
87. Zutshi, A., Sankaranarayanan, S., Tiwari, A.: Timed relational abstractions for sampled data control systems. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 343–361. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_27