# StarV: A Qualitative and Quantitative Verification Tool for Learning-enabled Systems

Hoang-Dung Tran[1,⋆], Sung Woo Choi[1,⋆], Yuntao Li[1,⋆], Qing Liu[1], Hideki Okamoto[2], Bardh Hoxha[2], and Georgios Fainekos[2]

[1] University of Florida
{dungtran, sungwoo.choi, yli17, qliu1}@ufl.edu
[2] Toyota NA R&D

**Abstract.** This paper presents StarV, a new tool for verifying deep neural networks (DNNs) and learning-enabled Cyber-Physical Systems (Le-CPS) using the well-known star reachability. Distinguished from existing star-based verification tools such as NNV and NNENUM and others, StarV not only offers qualitative verification techniques using Star and ImageStar reachability analysis but is also the first tool to propose using ProbStar reachability for quantitative verification of DNNs with piecewise linear activation functions and Le-CPS. Notably, it introduces a novel ProbStar Temporal Logic formalism and associated algorithms, enabling the quantitative verification of DNNs and Le-CPS's temporal behaviors. Additionally, StarV presents a novel SparseImageStar set representation and associated reachability algorithm that allows users to verify deep convolutional neural networks and semantic segmentation networks with more memory efficiency. StarV is evaluated in comparison with state-of-the-art in many challenging benchmarks. The experiments show that StarV outperforms existing tools in many aspects, such as timing performance, scalability, and memory consumption.

**Keywords:** DNNs · cyber-physical systems · verification · tool

## 1 Introduction

Deep learning (DL) models have been adopted to tackle many real-world challenging problems such as image classification [1, 41, 91], natural language processing [50], and robotics [63]. However, it is well-known that deep learning models are vulnerable to adversarial attacks where a slightly change in the inputs may lead to unexpected output results [65]. Therefore, verification for learning-enabled systems (LES) built on DL technology becomes crucial to enable the use of DL models in safety-critical domains such as autonomous vehicles [48] and cancer diagnosis [79]. Extensive research efforts have been made in the last few years to deep neural network verification challenge [30, 42] as well as neural network control system verification [76] in which qualitative verification, i.e., providing SAT or UNSAT or UNKNOWN results, is the primary focus. While

⋆ Equal Contribution.

qualitative verification is crucial, quantitative verification with probabilistic results provides more information about the system's safety under probabilistic uncertainties. For example, if a system is unsafe, then what is the probability of safety violation? Additionally, physical uncertainties, such as those in sensing or actuating, are more naturally modeled in a probabilistic manner.

**To fulfill the need for both qualitative and quantitative verification for LES, this paper introduces StarV, the first verification tool for deep learning models and learning-enabled cyber-physical systems (Le-CPS) that provides both qualitative and quantitative verification methods.** For qualitative verification, StarV reimplements and optimizes star-based verification approaches [6, 12, 13, 34, 66, 68, 69, 74, 75]. **Notably, StarV introduces a new set representation called SparseImageStar**, a substantial improvement of ImageStar [66], and associated memory-efficient reachability algorithms to verify the robustness of very deep models like VGG networks [60] under thousand pixels attacks. Thanks to the SparseImageStar approach, StarV is currently the only tool that can verify the robustness of the VGG16 network with up to 3000-pixel attacks on a local computer. **For quantitative verification, StarV introduces ProbStar set representation** [11, 55, 72], a new variant of star set [7, 14], that allows the probabilistic modeling of uncertainties and the associated reachability algorithms for of deep neural networks with piecewise linear activation functions, e.g., ReLU, LeakyReLU, and Satlin and neural network control systems with linear plant dynamics. **Notably, StarV also introduces ProbStar Temporal Logic (ProbStarTL) [70], a set-based formalism that enables the verification of LES's temporal properties using reachability analysis.** Thanks to ProbStarTL and associated verification algorithms, StarV is currently the only tool that can verify quantitatively the temporal properties of Le-CPS, such as a learning-based adaptive cruise control and emergency braking systems [68].

In summary, this paper provides a comprehensive overview of StarV and new features that users cannot find in individually published papers. It improves usability via multiple examples, case studies, and tutorials (in the user manual). This tool paper helps users quickly recognize all key features that can be used for their applications. Compared to individual published papers, there are many novelties and new technical contributions on 1) data structure, 2) reachability algorithms, 3) new activation functions support, and 4) new neural network architectures support. We also restructured the tool to make it more efficient and easier to use for future extensions. We highlight these novelties and new technical contributions in the following.

- **New data structures**: we implement new memory-efficient SparseStar and SparseImageStar data structures compared to the well-known Star and ImageStar.
- **New reachability algorithms**: we implemented new reachability algorithms on SparseStar and SparseImageStar data structures. We also implemented new quantitative reachability algorithms for massive linear systems using Krylov subspace method and probstar.

- **New activation functions**: For quantitative verification, we support new activation functions such as LeakyReLU, Satlin, and Satlins.
- **New network architectures**: StarV supports LSTM and GRU verification using new SparseStar reachability.
- **Improve Usability**: Installation instructions for local and Docker environments; Tutorials demonstrating key verification workflows; Example scripts for each supported network type; API documentation for extending the framework.

## 2 Related Work

**Qualitative Verification of LES.** Qualitative verification of learning-enabled systems (LES) has attracted the great attention of many researchers in recent years. In the area of open-loop LES (i.e., neural networks) verification, multiple approaches have been proposed, utilizing theories such as Satisfiability Modulo Theory (SMT) [2, 37, 38, 86], optimization using mixed integer linear programming encoding [43], star reachability [6, 74, 75], facet-vertex incidence matrix [89, 90], symbolic interval [82], semidefinite programming [19], abstract interpretation [54, 61, 93], input quantization [35], constraint-based [81], tree-based decomposition for incremental verification [80], certificate reuse [21] and relaxed convex programming [39], to name a few. There has also been significant effort in the development of efficient tools [6,15,20,22,38,46,78,87,89] (see [9] for more details). Closed-loop LES verification focuses on the safety of closed-loop neural network control systems under bounded input conditions, involving complex interactions between the neural network controller and the physical plant model [16,17,26–29,31,33,40,44,56,57,59,64,78,92]. Representative closed-loop LES verification frameworks include VeriSig [33], ReachNN [29], Sherlock [16], and NNV [67]. Recently, verification of perception-based control systems has attracted significant attention [47] and has been proven to be significantly challenging and time-consuming due to their large input space [24,26,32,52,56,64,90].

StarV reimplements and optimizes star-based reachability and qualitative verification algorithms for checking safety and robustness of a wide range of neural network architectures, such as feedforward neural networks (FFNNs) [74], recurrent neural networks (RNNs) [73], convolutional neural networks (CNNs) [66], and semantic segmentation networks (SSNs) [75], as well as neural network control systems (NNCS) [68]. Importantly, StarV tackles the memory consumption problem, a main bottleneck that reduces the scalability of reachability analysis and verification of very deep neural networks such as VGG16. To do that, we introduce SparseImageStar, a new set representation and associated reachability algorithms that allow memory-efficient verification of deep neural networks. Using SparseImageStar method, we can verify the robustness of VGG16 with up to 3000-pixel attack on a local computer where popular tools like CROWN [93], Marabou [38], DeepPoly [61], and NNENUM [3] cannot.

**Quantitative Verification of LES.** Although important, quantitative verification for LES has attracted less attention from the community. Some quantitative verification methods verify *binary neural networks* with *quantized finite* *discrete inputs space* [8, 58, 94]. Some methods are proposed for the popular ReLU DNNs [18,23,45,51,53,84,85] with *continuous input space*. The first effort focuses on improving the sampling-based methods to certify neural networks' robustness under probabilistic uncertain inputs [45, 53, 84, 85]. Although fast and scalable, these approaches do not provide a guarantee of output estimation for certification. The second effort focuses on probabilistic safety verification of ReLU networks with formal guarantee [18, 23, 71]. In the work by Fazlyab et al. [18], the authors investigate an ellipsoidal input space characterized by Gaussian random variables. They develop a method for propagating a confidence ellipsoid of the input through the neural network. By employing affine and quadratic constraints to approximate the nonlinear ReLU activation functions, they derive a confidence ellipsoid for the output. Furthermore, the safety of the original network can be established by analyzing the abstracted network using semidefinite programming techniques. In the work by Eric Goubault et al. [23], the authors introduce a new probabilistic abstraction named Zonotopic Dempster Shafer to construct tight overapproximation of the probabilistic outputs of a ReLU network using Interval Dempster Shafer arithmetic and probabilistic affine arithmetic. This approach can handle much more general classes of input uncertainties and provide guaranteed results.

StarV introduces ProbStar, a new set representation and associated quantitative reachability and verification algorithms for networks [71] with piecewise activation functions and neural network control systems (NNCS) with linear plant dynamics. It precisely estimates the safety violation probability of an LES under truncated Gaussian inputs distribution.

**Verification of LES's Temporal Properties.** The state-of-the-art techniques focus on safety, robustness, and fairness properties. There is a shortage of quantitative verification methods for complex temporal properties of Learning-enabled Systems (LES), specifically those that involve timing information. The neurosymbolic approach [25] is the first method developed to verify closed-loop LES with Signal Temporal Logic (STL) properties. This approach introduces a novel transformation technique, making the verification of temporal properties in closed-loop LES equivalent to performing reachability analysis of large feedforward neural networks.

StarV implements a method that is similar to the neurosymbolic approach [25] in that it also focuses on verifying temporal properties. However, we differentiate ourselves by introducing a new logic framework called ProbStarTL [70]. The semantics of this framework involve satisfying constraints on random variables, enabling us to precisely compute the probability of satisfaction. Additionally, our approach is more direct, as it does not require the transformation from STL to neural networks. This allows us to bypass the need for reachability analysis of large networks altogether. We highlight that neuroSymbolic approach uses the well-known signal temporal logic quantitative semantics to compute the robustness value of satisfaction (which can be any real number), while our ProbStarTL has its own quantitative semantics defined based on reachable set signals to compute the probability of satisfaction (which is between 0 and 1).

## 3 Conceptual Overview and Core Features

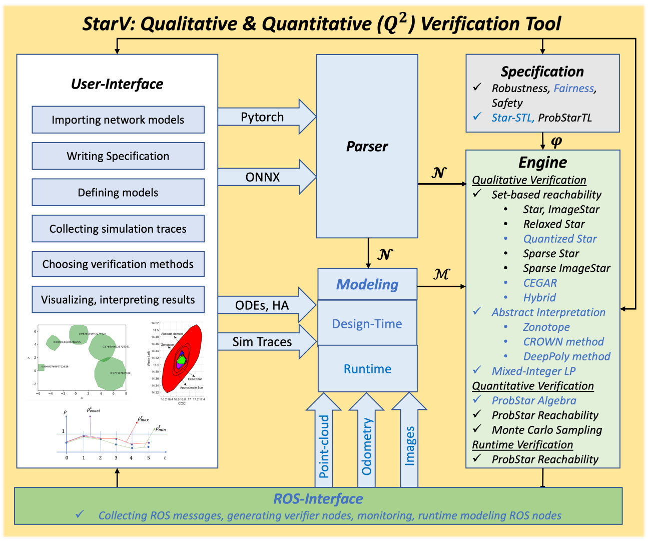

Fig. 1: A conceptual overview of StarV and core features (Features in blue color are under development).

### 3.1 Conceptual Overview

StarV is an object-oriented toolbox developed in Python. Its conceptual overview and major features are depicted in Figure 1. StarV aims to integrate popular and scalable verification approaches such as reachability-based [78], abstract interpretation [61, 93], and mixed integer linear programming [43] so that users can choose appropriate techniques for their particular applications. It also aims to facilitate the adoption of formal methods in real robotic applications built on Robotic Operating Systems (ROS) [11, 55].

Conceptually, StarV contains five modules, including a user interface, parser, specification and engine, and a ROS interface (under development). The user interface allows users to import a network model, write specifications, define a model, collect simulation traces, choose verification methods, and visualize and interpret verification results. The parser module currently supports automatic parsing of Pytorch and ONNX [49] models. StarV will construct corresponding internal models from these models for verification purposes. Users can also define their (StarV) models for verification using all supported layers or plant objects in StarV. The specification module allows users to specify the requirement for a LES. We fully support safety, robustness, and temporal properties (using ProbStarTL). The fairness and Star temporal logic (StarTL) for deterministic input sets are under development. The modeling module supports users to construct a neural network control system at design time or runtime. At design time, users can construct an NNCS with a neural network controller and a physical plant model (ODEs or Hybrid Automaton). We are working on supporting the modeling of a complex learning-enabled CPS with multiple neural network components interacting with each other and with the physical world. At runtime, we plan to support perception-based runtime modeling in which the linear physical plant motion dynamics, e.g., human motion [55] or vehicle [11] will be obtained using only perception data in ROS, i.e., LiDAR and Camera Point-Cloud data. These works have been done but have not yet been fully integrated into StarV. The ROS interface will be developed in the future to support automatic generation of verifier, runtime modeling, and monitoring ROS nodes for robotic applications. The engine module is the core of StarV, which contains different set representations and multiple verification and reachability algorithms.

**Verification Workflow and StarV's User Manual.** The general verification workflow for using StarV is as follows. Firstly, the users construct an LES model object, whether a neural network using a generic neural network object or an NNCS object in StarV. Secondly, the users then specify the property as a safety, or robustness or temporal property using ProbStarTL. Thirdly, the users specify the input conditions as an input set, e.g., Star, ImageStar, or ProbStar, and provide verification parameters such as verification methods, the number of cores used for verification, the linear programming solver, and the number of time steps (for NNCS verification). Finally, the users execute the verification using the methods in the StarV LES object. Users can interpret and visualize reachable sets and verification results using the supported plot functions in StarV. **The detailed workflows for different LES can be found in StarV's user manual**.

### 3.2 Core Features

Table 1 summarizes StarV's core features, highlighting its novel features compared to the state-of-the-art. As we mentioned above, StarV reimplements and optimizes star-based verification approaches that have also been implemented in NNV [44,77] and NNENUM [3]. Therefore, in this section, we only highlight core features that are StarV's novel contributions compared to the state-of-the-art, specifically in addressing quantitative verification and memory and scalability problems in verifying deep CNNs.

**ProbStar Reachability [71].** StarV implements probstar reachability, which is built on a new set representation named probabilistic star (or shortly ProbStar), a variant of the well-known star set used in DNNs [6,66,74,75], and linear dynamical and hybrid systems verification [4,5,14]. A ProbStar is an affine mapping of a truncated multivariate Gaussian distribution that can be used to model probabilistic inputs and efficiently propagate them through the network to construct the reachable output set. The reachable output set (a union of probstars) is then used to verify a user-defined safety property in which the violation probability can be obtained efficiently. StarV supports exact quantitative verification, where the precise probability of safety violation is computed. It also supports approximate verification by filtering out reachable intermediate sets (in the layers) with probabilities lower than a user-predefined threshold. The exact verification algorithm is expensive as it explores all paths in reachability analysis. The

| **Feature** | **Supported** |
|---|---|
| Neural Network Type | FFNN, CNN, SSN, Vanilla RNN, LSTM, GRU |
| Layers | MaxPool, Conv, BN, AvgPool, FC, TC, DC, |
| Activation functions | ReLU, Satlin, Sigmoid, Tanh, Leaky ReLU, Satlins |
| Plant dynamics (NNCS) | Linear ODE, Massive Linear ODE, Continuous & Discrete Time |
| Set Representation | Star, ImageStar, SparseStar, SparseImageStar, ProbStar |
| Qualitative Reach methods | exact, approx, relax, abs-dom |
| Quantitative Reach methods | exact, approx |
| Reachable set visualization | exact and over-approximation |
| Specification | Safety, Robustness, ProbStarTL Temporal Properties |
| Miscellaneous | Parallel computing, counterexample generation |
| Solver | Gurobi, GLPK |
| Import | Pytorch, ONNX |

Table 1: Overview of core features available in StarV. BN refers to batch normalization layers, FC to fully-connected layers, AvgPool to average pooling layers, Conv to convolutional layers, MaxPool to max-pooling layers, TC to transpose convolutional layers, and DC to dilated convolutional layers.

over-approximate verification algorithm focuses on exploring paths with large probabilities, thus reducing the number of reachable sets involved and memory consumption in verification. Nevertheless, the over-approximate verification algorithm must filter intermediate ProbStars, which may be costly.

**ProbStar Temporal Logic Verification** [70]**.** StarV implements Probstar Temporal Logic (ProbStarTL), a formalism enabling quantitative verification of temporal properties of LES using probstar reachability analysis. ProbStarTL is defined on a (bounded-time) ProbStar signal (or ProbStar trace), a sequence of discrete, timed probstar reachable sets. The interpretation of ProbStarTL captures a symbolic representation of the set of LES traces that satisfy the specification. ProbStarTL supports two basic temporal operators: *always* ($\Box$) and *eventually* ($\Diamond$). Since ProbStarTL is defined only over discrete-time and bounded-time intervals, the *until* ($\mathcal{U}$) operator is evaluated using the equivalent formula composed of the *always* ($\Box$) and *eventually* ($\Diamond$) operators. The ProbStar traces are constructed using exact or approximate ProbStar reachability algorithms, focusing on NNCS reachability with a feedforward neural network controlling a discrete linear plant model. In the future, we will extend the approach to verify temporal behaviors of networks handling time-series data, such as recurrent neural networks [73]. The verification of LES's temporal properties is done in two steps. First, we transform the user-defined ProbStar specification into a *abstract disjunctive normal form (ADNF)*, which is realized on the constructed reachable set traces to construct *computable disjunctive normal form (CDNF)*. The exact verification algorithm, while computationally expensive, calculates the exact satisfaction probability from the constructed CDNF. In contrast, the approximate verification algorithm, less expensive than the exact one, estimates only the lower and upper bounds of satisfaction probability from the CDNF.

**ProbStar Reachability and Verification for Massive Linear Systems.** StarV implements an efficient simulation-based probstar reachability method for massive linear systems based on the Krylov-subspace method proposed in [7]. However, with ProbStarTL [70], StarV allows users to quantitatively verify temporal behaviors of massive linear systems, a novel feature compared to the state-of-the-art, which supports only safety verification [4,5,7]. Our reachability analysis leverages state-space projection techniques to enhance memory efficiency and employs the Krylov subspace method in numerical simulation to optimize computation time at each discrete-time step. Additionally, our approach develops a robust quantitative verification algorithm based on ProbStarTL, which efficiently calculates the probability of satisfaction on a ProbStarTL specification, providing a way to quantify the system's temporal behaviors. We demonstrate the scalability and efficiency of our approach by successfully verifying nine large-scale linear systems, each with up to 10,000 dimensions.
**SparseImageStar Reachability for CNNs.** Verification of large CNNs, such as VGG16, with high-dimensional input like ImageNet presents significant challenges, particularly when computing resources are limited. To address these challenges, we develop SparseImageStar, a highly memory-efficient representation designed to handle pixel-level attacks involving up to 3000 pixels while ensuring scalability. This is accomplished by transforming multiple 3D RGB images into a column stack of flattened images in sparse matrix formats such as Coordinate (COO) and Compressed Sparse Row (CSR). However, this transformation disrupts the original spatial relationships. To mitigate this issue, we propose the indices-shifting technique that restores these spatial relationships without the need to revert the images to their original representation. Additionally, this technique enables SparseImageStar to operate at the feature map level rather than the pixel level. Leveraging this approach, we implement novel SpGEMM convolution and average pooling operations that directly operate on SparseImageStar, eliminating the need for feature extraction from the input, all while preserving both memory efficiency and scalability.

## 4 Evaluation

### 4.1 Qualitative Verification of LES

**LSTM and GRU Verification** We evaluate MNIST LSTM, $\mathcal{L}_{15}$ and GRU, $\mathcal{G}_{15}$ RNNs with SparseStar against infinity norm attack such that $\bigwedge_{t=1}^{T_{max}} \|x_i'^t - x_i^t\|_\infty \leq \epsilon, T_{max} = 2$. SparseStar minimizes memory consumption by eliminating dependent basis vectors and storing linear constraints matrix in Compressed Sparse Column (CSC) format. SparseStar has the unique feature of reducing the number of predicates by their depth level, i.e., DR. We over-approximate LSTM and GRU layers as the combined activation operations in a new 3D geometric approach. In Figure 2, SparseStar with LP solver('LP') proves the most robust cases for both networks, whereas with 'DR = 1' predicate reduction and 'EST' with estimate ranges, SparseStar proves the fewest cases due to the convex relation in return for improved scalability and memory consumption. The experiment is conducted on a computer with Intel Core i7-10700 CPU, 63.7 GiB Memory, 64-bit Ubuntu 18.04.6 LTS OS.

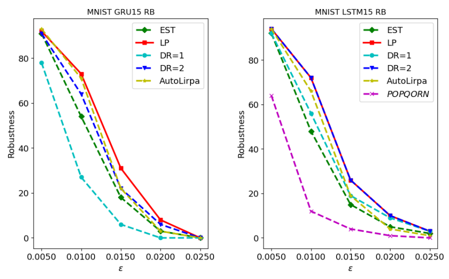

Fig. 2: $\mathcal{L}_{15}$ and $\mathcal{G}_{15}$ verification results. *'LP' proves the most robust cases.* AutoLirpa [87] proves as many robust cases as 'LP,' but it becomes more conservative as the epsilon becomes bigger. For $\mathcal{L}_{15}$, 'EST' proves more than AutoLirap for $\epsilon = 0.025$. POPQORN is the most conservative approach.

| | | Robustness results (%) | | | | | | | | | Verification time (sec) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Small | | | Medium | | | Large | | | Small | | | Medium | | | Large | | |
| | | $\delta = 0.005$ | $\delta = 0.01$ | $\delta = 0.015$ | $\delta = 0.005$ | $\delta = 0.01$ | $\delta = 0.015$ | $\delta = 0.005$ | $\delta = 0.01$ | $\delta = 0.015$ | $\delta = 0.005$ | $\delta = 0.01$ | $\delta = 0.015$ | $\delta = 0.005$ | $\delta = 0.01$ | $\delta = 0.015$ | $\delta = 0.005$ | $\delta = 0.01$ | $\delta = 0.015$ |
| $d = 250$ | IM | 87 | 87 | 87 | 99 | 99 | 99 | 99 | 99 | 99 | 0.135 | 0.195 | 0.195 | 0.143 | 0.220 | 0.345 | 0.190 | 0.284 | 0.395 |
| | SIM_csr | 87 | 87 | 87 | 99 | 99 | 99 | 99 | 99 | 99 | 0.143 | 0.177 | 0.206 | 0.236 | 0.297 | 0.362 | 0.534 | 0.628 | 0.727 |
| | SIM_coo | 87 | 87 | 87 | 99 | 99 | 99 | 99 | 99 | 99 | 0.166 | 0.194 | 0.216 | 0.274 | 0.375 | 0.454 | 0.693 | 0.865 | 1.032 |
| | NNV | 87 | 87 | 87 | 99 | 99 | 99 | 99 | 99 | 99 | 0.207 | 0.327 | 0.469 | 0.433 | 1.210 | 2.179 | 0.546 | 1.208 | 2.178 |
| $d = 245$ | IM | 78 | 78 | 78 | 95 | 95 | 95 | 100 | 100 | 99 | 0.143 | 0.194 | 0.269 | 0.188 | 0.328 | 0.503 | 0.240 | 0.396 | 0.604 |
| | SIM_csr | 78 | 78 | 78 | 95 | 95 | 95 | 100 | 100 | 99 | 0.168 | 0.221 | 0.255 | 0.281 | 0.387 | 0.542 | 0.619 | 0.770 | 0.965 |
| | SIM_coo | 78 | 78 | 78 | 95 | 95 | 95 | 100 | 100 | 99 | 0.169 | 0.222 | 0.293 | 0.331 | 0.570 | 0.666 | 0.846 | 1.163 | 1.472 |
| | NNV | 78 | 78 | 77 | 95 | 95 | 95 | 100 | 100 | 99 | 0.253 | 0.461 | 0.721 | 0.670 | 1.759 | 3.449 | 0.739 | 1.880 | 3.725 |
| $d = 240$ | IM | 73 | 73 | 73 | 90 | 90 | 88 | 99 | 99 | 99 | 0.149 | 0.242 | 0.372 | 0.236 | 0.424 | 0.863 | 0.288 | 0.496 | 0.784 |
| | SIM_csr | 73 | 73 | 73 | 90 | 90 | 88 | 99 | 99 | 99 | 0.211 | 0.271 | 0.361 | 0.292 | 0.459 | 0.639 | 0.691 | 0.893 | 1.182 |
| | SIM_coo | 73 | 73 | 73 | 90 | 90 | 88 | 99 | 99 | 99 | 0.222 | 0.248 | 0.322 | 0.370 | 0.602 | 0.963 | 1.008 | 1.353 | 1.842 |
| | NNV | 73 | 72 | 71 | 90 | 89 | 88 | 99 | 99 | 99 | 0.335 | 0.642 | 0.989 | 0.906 | 2.653 | 5.400 | 0.942 | 2.333 | 5.445 |

Table 2: Verification results of the MNIST CNN [66]. SIMs are up to 3.07 ×, 8.45 ×, 4.60 × faster than NNV in Small, Medium, Large networks, respectively.

**StarV-ImageStar vs. NNV-ImageStar** We evaluate SparseImageStar in CSR (SIM$_{CSR}$) and COO (SIM$_{COO}$) formats and ImageStar in StarV to compare with NNV for verifying the robustness of MNIST CNNs under brightening attacks. In Table 2, ImageStar in StarV is up to 2.66 ×, 6.85 ×, 6.94 × faster than NNV in Small, Medium, Large MNIST CNN networks, respectively. SIM$_{CSR}$ is up to 2.82 ×, 8.45 ×, and 4.60 × faster than NNV in Small, Medium, Large networks, respectively, while SIM$_{COO}$ is up to 3.07 ×, 5.60 ×, 2.95 × faster than NNV. The substantial improvement is achieved due to the *indices-shifting technique* that allows our SparseImageStar and ImageStar to operate at the feature map level instead of the unscalable pixel level. The experiment is conducted on a computer with Intel Core i7-6950X CPU, 125.7 GB Memory, RTX 3090 GPU, and Ubuntu 20.04 LTS OS.

**Robustness Verification of VGG16** We compare our SparseImageStar with NNV [77], DeepPoly [62], NNENUM [3], $\alpha, \beta$-Crown [83,87,88,93], Marabou [86] by verifying the robustness of the VGG16 network from vnncomp2023 [10] under infinity norm attack with randomly selected $e \in [200, 3000]$ pixel attacks on the corn image with $\epsilon = 0.001/255$ perturbation. SIM$_{CSR}$ and SIM$_{COO}$ are the only two methods to verify all specifications without memory and scalability issues. SIM$_{CSR}$ is the fastest method: 1.15 ×, 1.24 × faster than NNENUM for $c_0$, $c_2$, respectively. For the spec 11 image, SIM methods require up to 38.91 MB, which is 18× memory efficient than ImageStar and NNV, which require up to

0.59 GB. The experiment is conducted on a computer with Intel Core i7-6950X CPU, 125.7 GB Memory, RTX 3090 GPU, and Ubuntu 20.04 LTS OS.

| | | | Original Network (seconds) | | | | | | | Relufied Network (seconds) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Specs | e | Result | m | IM | SIM$_{CSR}$ | SIM$_{COO}$ | NNV | DeepPoly | Marabou | IM | NNV | NNENUM | $\alpha,\beta$-CROWN | $\beta$-CROWN |
| $c_0$ | 200 | UNSAT | 358 | O/M | 642.5 | 871.2 | O/M | O/M | T/O | O/M | O/M | 744.02 | T/O | 26782.3 |
| $c_2$ | 400 | UNSAT | 642 | O/M | 1089.4 | 1543.2 | O/M | O/M | T/O | O/M | O/M | 1354.75 | T/O | T/O |
| $c_4$ | 1000 | UNSAT | 1613 | O/M | 2512.1 | 4076.5 | O/M | O/M | T/O | O/M | O/M | T/O | T/O | T/O |
| $c_6$ | 3000 | UNSAT | 4871 | O/M | 7708.7 | 17362.4 | O/M | O/M | T/O | O/M | O/M | O/M | T/O | T/O |

Table 3: *SIM$_{CSR}$ and SIM$_{COO}$ are the only methods that verify all specifications without memory and scalability issues, with SIM$_{CSR}$ being the fastest for VGG16 verification.* 'O/M' and 'T/O' denote out-of-memory and time-out (12 hours), respectively; $e$ is the number of attacked pixels, and $m$ is the number of predicates in the output reachable set (IM: StarV ImageStar; SIM: SparseImageStar).

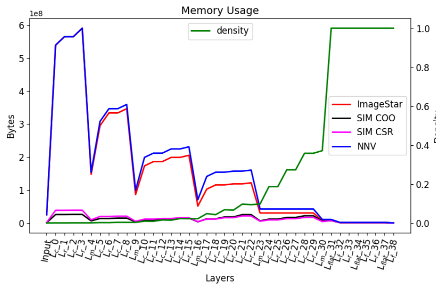

Fig. 3: *SIM methods require up to 38.91 MB, while ImageStar and NNV require up to 0.59 GB. SIM methods are 18 $\times$ more memory efficient* in verifying the VGG16 with spec 11 image, which has $l_\infty$ norm attack on 20 pixels.

## 4.2 Quantitative Verification of LES

**Quantitative Verification for ACASXu Networks.** In this experiment, we assess our quantitative verification on unsafe ACASXu networks (properties $P_2$, $P_3$, and $P_4$) [6]) using 16 cores. Our approach computes violation bounds for 90 queries in roughly 24 hours (query times: 5s–2hrs). Partial results are shown in Table 4, the approximate scheme ($p_f = 10^{-5}$) can speed up verification by up to 10$\times$, though its upper bounds are sometimes conservative. Our results are consistent with NNV [78], Marabou [86], and NNenum [3]. While our method and NNV perform exhaustive counterexample searches and are slower, Marabou and NNenum stop after the first counterexample. Moreover, Monte Carlo sampling requires many samples, potentially leading to memory issues for low violation probabilities, highlighting the efficiency of our ProbStar approach. The experiment is conducted on a computer with Intel Core i7-6950X CPU, 125.7 GiB Memory, 64-bit Ubuntu 20.04.6 LTS OS.

**Quantitative Verification of Temporal Properties of Le-ACC.** We evaluate our approach to the learning-based adaptive cruise control (Le-ACC) system from the ARCH competition [36, 78] in comparison with the NeuroSymbolic [25] approach with the network controller $N_{5\times20}$ for property $\varphi_3$ in [25]. Our ProbStarTL framework achieves safety guarantees similar to those of the NeuroSymbolic method but with substantially lower verification times. By decoupling reachability from property checking and computing satisfaction probabilities directly, our approach is significantly faster on the Le-ACC system, as shown in Table 5. The experiment is executed on an iMAC 3.8 GHz 8-Core Intell Core i7 with 128GB memory with a virtual 64-bit Ubuntu 20.04.4 LTS system.

| Quantitative Verification | | | | | | | | | | | | Monte Carlo | | Qualitative Verification | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Prop** | **Net** | $p_f$ | $\mathcal{O}$ | $\mathcal{US}-\mathcal{O}$ | $\mathcal{C}$ | **US-Prob-LB** | **US-Prob-UB** | **US-Prob-Min** | **US-Prob-Max** | **I-Prob** | **VT** | **US-Prob** | **VT** | **NNV** | **Marabou** | **NNenum** |
| 2 | 1-6 | 0 | 376352 | 80621 | 80621 | 9.07123e-06 | 9.07123e-06 | 9.07123e-06 | 0.0134354 | 0.986574 | 1287.45 | 4e-07 | 223.223 | 13739.970 | 166.58 | 1.5938 |
| 2 | 1-6 | 1e-05 | 4874 | 2662 | 2662 | 3.88239e-06 | 0.0538327 | 3.88239e-06 | 0.067259 | 0.986574 | 228.114 | | | | | |
| 3 | 1-7 | 0 | 500 | 500 | 500 | 0.986574 | 0.986574 | 0.986574 | 1 | 0.986574 | 7.52849 | 1 | 223.804 | 0.943 | 0.25 | 0.86683 |
| 3 | 1-7 | 1e-05 | 190 | 190 | 190 | 0.984972 | 0.985307 | 0.984972 | 0.998733 | 0.986574 | 6.635 | | | | | |
| 4 | 1-9 | 0 | 471 | 471 | 471 | 0.989244 | 0.989244 | 0.989244 | 1 | 0.989244 | 7.24032 | 1 | 209.705 | 1.176 | 0.31 | 0.86635 |
| 4 | 1-9 | 1e-05 | 142 | 142 | 142 | 0.989244 | 0.989244 | 0.989244 | 1 | 0.989244 | 5.38665 | | | | | |

Table 4: Quantitative verification results for unsafe ACASXu networks compared with MC sampling ($10^7$ samples) and qualitative tools. Notations: $p_f$ (filtering probability), $\mathcal{O}$ (total output sets), $\mathcal{US}-\mathcal{O}$ (unsafe output sets), $\mathcal{C}$ (counter input sets), US-Prob-LB/US-Prob-UB (lower/upper bounds of violation probability), US-Prob-Min/US-Prob-Max (min/max unsafe probability over the infinite input space), $I$-Prob (input probability), and $VT$ (verification time in seconds).

| | | $T = 10$ | | $T = 20$ | | $T = 30$ | | $T = 50$ | |
|---|---|---|---|---|---|---|---|---|---|
| CtrlNet | Method | RS | VT (sec) | RS | VT (sec) | RS | VT (sec) | RS | VT (sec) |
| $N_{5\times 20}$ | ProbStarTL | [0.9512, 0.9512] | 1.7683 | [0.9512, 0.9512] | 5.892 | [0.9512, 0.9512] | 9.141 | [0.9512, 0.9512] | 29.0227 |
| | NeuroSymbolic | [26.9067, 48.2244] | 13.4654 | [26.9067, 48.2244] | 76.0396 | [26.9067, 48.2244] | 137.227 | [24.0621, 44.3187] | 356.74 |

Table 5: Verification results (robustness intervals) of NeuroSymbolic [25] are consistent with the proposed ProbStarTL verification results (probabilities of satisfaction). The proposed ProbStarTL verification approach is significantly faster than the NeuroSymbolic. $RS$ is the verification result, $VT$ is the verification time (in seconds), and $T$ is the number of time steps.

## 5 Conclusions

We present StarV, a new qualitative and quantitative verification tool for LES. Our new tool reimplements and optimizes qualitative verification methods of the state-of-the-art for a wide range of network architectures. Notably, StarV includes a new set representation named SparseImageStar and SparseStar, as well as reachability and verification algorithms to improve memory efficiency and enhance the scalability of qualitative verification methods. Additionally, StarV introduces novel quantitative reachability and verification algorithms using ProbStar and ProbStarTL for neural networks with piecewise linear activation functions and NNCS. In the future, we will enhance StarV in the following aspects. Firstly, we will build a graphical, user-friendly interface and ROS interface for robotic applications. Secondly, we will develop Star Temporal Logic for verifying the temporal behaviors of LES with deterministic input sets. Thirdly, we will support fairness verification by developing hybrid and CEGAR reachability algorithms for LES. Fourthly, we will support the verification of complex LES with multiple neural network components and Large Language Model Verification by developing a Star/ProbStar algebra foundation and generic graph-based reachability algorithms. The key challenge in verifying complex LES with multiple neural network components is the complicated information flow (under uncertainties) inside the system. Therefore, tracking the dependencies of this information (represented as reachable sets) in multiple network components interacting with each other is crucial to constructing precise reachable sets for verification. Additionally, scalability is also a main challenge for verifying complex Le-CPS (even in design time). Addressing these challenges is the focus of our future work. Finally, we will extend our integration to CROWN, DeepPoly, and MILP approaches.

## Acknowledgments

This work is supported by the National Science Foundation (NSF) under grants NSF-CAREER-2441334, NSF-SLES-2331937, and NSF-FMitF-2220418. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF.

## References

1. Affonso, C., Rossi, A.L.D., Vieira, F.H.A., de Leon Ferreira, A.C.P., et al.: Deep learning for biological image classification. Expert systems with applications **85**, 114–122 (2017)
2. Amir, G., Wu, H., Barrett, C., Katz, G.: An smt-based approach for verifying binarized neural networks. In: Tools and Algorithms for the Construction and Analysis of Systems: 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27–April 1, 2021, Proceedings, Part II 27. pp. 203–222. Springer (2021)
3. Bak, S.: nnenum: Verification of relu neural networks with optimized abstraction refinement. In: NASA Formal Methods Symposium. pp. 19–36. Springer (2021)
4. Bak, S., Duggirala, P.S.: Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In: Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control. pp. 173–178. ACM (2017)
5. Bak, S., Duggirala, P.S.: Simulation-equivalent reachability of large linear systems with inputs. In: International Conference on Computer Aided Verification. pp. 401–420. Springer (2017)
6. Bak, S., Tran, H.D., Hobbs, K., Johnson, T.T.: Improved geometric path enumeration for verifying relu neural networks. In: Proceedings of the 32nd International Conference on Computer Aided Verification. Springer (2020)
7. Bak, S., Tran, H.D., Johnson, T.T.: Numerical verification of affine systems with up to a billion dimensions. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control. pp. 23–32. ACM (2019)
8. Baluta, T., Shen, S., Shinde, S., Meel, K.S., Saxena, P.: Quantitative verification of neural networks and its security applications. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1249–1264 (2019)
9. Brix, C., Bak, S., Liu, C., Johnson, T.T.: The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results. arXiv preprint arXiv:2312.16760 (2023)
10. Brix, C., Bak, S., Liu, C., Johnson, T.T.: The fourth international verification of neural networks competition (vnn-comp 2023): Summary and results. ArXiv **abs/2312.16760** (2023), https://api.semanticscholar.org/CorpusID:266572985
11. Brown, R., Nguyen, L.V., Xiang, W., Wolf, M., Tran, H.D.: Perception-based quantitative runtime verification for learning-enabled cyber-physical systems. In: 2025 the 16th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS). pp. –. ACM/IEEE (2025)
12. Choi, S.W., Ivashchenko, M., Nguyen, L.V., Tran, H.D.: Reachability analysis of sigmoidal neural networks. ACM Transactions on Embedded Computing Systems (2023)

13. Choi, S.W., Li, Y., Yang, X., Yamaguchi, T., Hoxha, B., Fainekos, G., Prokhorov, D., Tran, H.D.: Reachability analysis of recurrent neural networks. Nonlinear Analysis: Hybrid Systems (2025)
14. Duggirala, P.S., Viswanathan, M.: Parsimonious, simulation based verification of linear systems. In: International Conference on Computer Aided Verification. pp. 477–494. Springer (2016)
15. Duong, H., Li, L., Nguyen, T., Dwyer, M.: A dpll (t) framework for verifying deep neural networks. arXiv preprint arXiv:2307.10266 (2023)
16. 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. pp. 157–168 (2019)
17. Everett, M., Habibi, G., Sun, C., How, J.P.: Reachability analysis of neural feedback loops. IEEE Access **9**, 163938–163953 (2021)
18. Fazlyab, M., Morari, M., Pappas, G.J.: Probabilistic verification and reachability analysis of neural networks via semidefinite programming. In: 2019 IEEE 58th Conference on Decision and Control (CDC). pp. 2726–2731. IEEE (2019)
19. Fazlyab, M., Morari, M., Pappas, G.J.: Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Transactions on Automatic Control (2020)
20. Ferlez, J., Khedr, H., Shoukry, Y.: Fast batllnn: fast box analysis of two-level lattice neural networks. In: Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control. pp. 1–11 (2022)
21. Fischer, M., Sprecher, C., Dimitrov, D.I., Singh, G., Vechev, M.: Shared certificates for neural network verification. In: International Conference on Computer Aided Verification. pp. 127–148. Springer (2022)
22. Girard-Satabin, J., Alberti, M., Bobot, F., Chihani, Z., Lemesle, A.: Caisar: A platform for characterizing artificial intelligence safety and robustness. arXiv preprint arXiv:2206.03044 (2022)
23. Goubault, E., Putot, S.: A zonotopic dempster-shafer approach to the quantitative verification of neural networks. In: International Symposium on Formal Methods. pp. 324–342. Springer (2024)
24. Habeeb, P., D'Souza, D., Lodaya, K., Prabhakar, P.: Interval image abstraction for verification of camera-based autonomous systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **43**(11), 4310–4321 (2024)
25. Hashemi, N., Hoxha, B., Yamaguchi, T., Prokhorov, D., Fainekos, G., Deshmukh, J.: A neurosymbolic approach to the verification of temporal logic properties of learning-enabled control systems. In: Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023). pp. 98–109 (2023)
26. Hsieh, C., Li, Y., Sun, D., Joshi, K., Misailovic, S., Mitra, S.: Verifying controllers with vision-based perception using safe approximate abstractions. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **41**(11), 4205–4216 (2022)
27. Hu, H., Fazlyab, M., Morari, M., Pappas, G.J.: Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In: 2020 59th IEEE conference on decision and control (CDC). pp. 5929–5934. IEEE (2020)

28. Huang, C., Fan, J., Chen, X., Li, W., Zhu, Q.: Polar: A polynomial arithmetic framework for verifying neural-network controlled systems. In: International Symposium on Automated Technology for Verification and Analysis. pp. 414–430. Springer (2022)
29. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: Reachnn: Reachability analysis of neural-network controlled systems. arXiv preprint arXiv:1906.10654 (2019)
30. Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., Yi, X.: A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Computer Science Review **37**, 100270 (2020)
31. 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: International Conference on Computer Aided Verification. pp. 249–262. Springer (2021)
32. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Case study: verifying the safety of an autonomous racing car with a neural network controller. In: Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control. pp. 1–7 (2020)
33. 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)
34. Ivashchenko, M., Choi, S.W., Nguyen, L.V., Tran, H.D.: Verifying binary neural networks on continuous input space using star reachability. In: 2023 IEEE/ACM 11th International Conference on Formal Methods in Software Engineering (FormaliSE). pp. 7–17. IEEE (2023)
35. Jia, K., Rinard, M.: Verifying low-dimensional input neural networks via input quantization. In: International Static Analysis Symposium. pp. 206–214. Springer (2021)
36. Johnson, T.T., Manzanas Lopez, D., Musau, P., Tran, H.D., Botoeva, E., Leofante, F., Maleki, A., Sidrane, C., Fan, J., Huang, C.: Arch-comp20 category report: Artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. EPiC Series in Computing **74** (2020)
37. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: International Conference on Computer Aided Verification. pp. 97–117. Springer (2017)
38. Katz, G., Huang, D.A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al.: The marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification. pp. 443–452. Springer (2019)
39. Khedr, H., Ferlez, J., Shoukry, Y.: Peregrinn: Penalized-relaxation greedy neural network verifier. In: International Conference on Computer Aided Verification. pp. 287–300. Springer (2021)
40. Kochdumper, N., Schilling, C., Althoff, M., Bak, S.: Open-and closed-loop neural network verification using polynomial zonotopes. In: NASA Formal Methods Symposium. pp. 16–36. Springer (2023)
41. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

42. Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M.J.: Algorithms for verifying deep neural networks. Foundations and Trends in Optimization **4**(3-4), 244–404 (2021). https://doi.org/10.1561/2400000035
43. Lomuscio, A., Maganti, L.: An approach to reachability analysis for feed-forward relu neural networks. arXiv preprint arXiv:1706.07351 (2017)
44. Lopez, D.M., Choi, S.W., Tran, H.D., Johnson, T.T.: Nnv 2.0: the neural network verification tool. In: International Conference on Computer Aided Verification. pp. 397–412. Springer (2023)
45. Mangal, R., Nori, A.V., Orso, A.: Robustness of neural networks: A probabilistic and practical approach. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER). pp. 93–96. IEEE (2019)
46. Matthew Sotoudeh, Z.T., Thakur, A.V.: Syrenn: A tool for analyzing deep neural networks (2023). https://doi.org/10.1007/s10009-023-00695-1
47. Mitra, S., Păsăreanu, C., Prabhakar, P., Seshia, S.A., Mangal, R., Li, Y., Watson, C., Gopinath, D., Yu, H.: Formal verification techniques for vision-based autonomous systems–a survey. In: Principles of Verification: Cycling the Probabilistic Landscape: Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III, pp. 89–108. Springer (2024)
48. Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., de Albuquerque, V.H.C.: Deep learning for safe autonomous driving: Current challenges and future directions. IEEE Transactions on Intelligent Transportation Systems **22**(7), 4316–4336 (2020)
49. (ONNX), O.N.N.E.: `https://github.com/onnx/`
50. Otter, D.W., Medina, J.R., Kalita, J.K.: A survey of the usages of deep learning for natural language processing. IEEE transactions on neural networks and learning systems **32**(2), 604–624 (2020)
51. Păsăreanu, C., Converse, H., Filieri, A., Gopinath, D.: On the probabilistic analysis of neural networks. In: Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 5–8 (2020)
52. Păsăreanu, C.S., Mangal, R., Gopinath, D., Getir Yaman, S., Imrie, C., Calinescu, R., Yu, H.: Closed-loop analysis of vision-based autonomous systems: A case study. In: International conference on computer aided verification. pp. 289–303. Springer (2023)
53. Pautov, M., Tursynbek, N., Munkhoeva, M., Muravev, N., Petiushko, A., Oseledets, I.: Cc-cert: A probabilistic approach to certify general robustness of neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 7975–7983 (2022)
54. Prabhakar, P., Rahimi Afzal, Z.: Abstraction based output range analysis for neural networks. Advances in Neural Information Processing Systems **32** (2019)
55. Pramanik, A., Choi, S.W., Li, Y., Nguyen, L.V., Kim, K., Tran, H.D.: Perception-based runtime monitoring and verification for human-robot construction systems. In: 2024 22nd ACM-IEEE International Symposium on Formal Methods and Models for System Design (MEMOCODE). pp. 124–134. IEEE (2024)
56. Santa Cruz, U., Shoukry, Y.: Nnlander-verif: A neural network formal verification framework for vision-based autonomous aircraft landing. In: NASA Formal Methods Symposium. pp. 213–230. Springer (2022)
57. Schilling, C., Forets, M., Guadalupe, S.: Verification of neural-network control systems by integrating taylor models and zonotopes. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 8169–8177 (2022)

58. Shih, A., Darwiche, A., Choi, A.: Verifying binarized neural networks by angluin-style learning. In: SAT. pp. 354–370 (2019), https://doi.org/10.1007/978-3-030-24258-9_25
59. Sidrane, C., Maleki, A., Irfan, A., Kochenderfer, M.J.: Overt: An algorithm for safety verification of neural network control policies for nonlinear systems. Journal of Machine Learning Research **23**(117), 1–45 (2022)
60. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
61. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages **3**(POPL), 41 (2019)
62. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang. **3**(POPL) (Jan 2019). https://doi.org/10.1145/3290354
63. Soori, M., Arezoo, B., Dastres, R.: Artificial intelligence, machine learning and deep learning in advanced robotics, a review. Cognitive Robotics **3**, 54–70 (2023)
64. 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. pp. 147–156 (2019)
65. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
66. Tran, H.D., Bak, S., Xiang, W., Johnson, T.T.: Verification of deep convolutional neural networks using imagestars. In: 32nd International Conference on Computer-Aided Verification (CAV). Springer (July 2020)
67. Tran, H.D., Cai, F., Diego, M.L., Musau, P., Johnson, T.T., Koutsoukos, X.: Safety verification of cyber-physical systems with reinforcement learning control. ACM Transactions on Embedded Computing Systems (TECS) **18**(5s), 1–22 (2019)
68. Tran, H.D., Cei, F., Lopez, D.M., Johnson, T.T., Koutsoukos, X.: Safety verification of cyber-physical systems with reinforcement learning control. In: ACM SIGBED International Conference on Embedded Software (EMSOFT'19). ACM (October 2019)
69. Tran, H.D., Choi, S.W., Yang, X., Yamaguchi, T., Hoxha, B., Prokhorov, D.: Verification of recurrent neural networks with star reachability. In: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control. pp. 1–13 (2023)
70. Tran, H.D., Choi, S., Li, Y., Okamoto, H., Hoxha, B., Fainekos, G.: Probstar temporal logic for verifying complex behaviors of learning-enabled systems. In: Proceedings of the 28th ACM International Conference on Hybrid Systems: Computation and Control (2025)
71. Tran, H.D., Choi, S., Okamoto, H., Hoxha, B., Fainekos, G., Prokhorov, D.: Quantitative verification for neural networks using probstars. In: Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control. pp. 1–12 (2023)
72. Tran, H.D., Choi, S., Yamaguchi, T., Hoxha, B., Prokhorov, D.: Verification of recurrent neural networks using star reachability. In: The 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC) (May 2023)
73. Tran, H.D., Choi, S., Yamaguchi, T., Hoxha, B., Prokhorov, D.: Verification of recurrent neural networks using star reachability. In: The 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC) (May 2023)

74. Tran, H.D., Musau, P., Lopez, D.M., Yang, X., Nguyen, L.V., Xiang, W., Johnson, T.T.: Star-based reachability analsysis for deep neural networks. In: 23rd International Symposisum on Formal Methods (FM'19). Springer International Publishing (October 2019)
75. Tran, H.D., Pal, N., Musau, P., Lopez, D.M., Hamilton, N., Yang, X., Bak, S., Johnson, T.T.: Robustness verification of semantic segmentation neural networks using relaxed reachability. In: International Conference on Computer Aided Verification. pp. 263–286. Springer (2021)
76. Tran, H.D., Xiang, W., Johnson, T.T.: Verification approaches for learning-enabled autonomous cyber-physical systems. IEEE Design & Test (2020)
77. Tran, H.D., Yang, X., Lopez, D.M., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: 32nd International Conference on Computer-Aided Verification (CAV) (July 2020)
78. Tran, H.D., Yang, X., Manzanas Lopez, D., Musau, P., Nguyen, L.V., Xiang, W., Bak, S., Johnson, T.T.: Nnv: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In: International Conference on Computer Aided Verification. pp. 3–17. Springer (2020)
79. Tran, K.A., Kondrashova, O., Bradley, A., Williams, E.D., Pearson, J.V., Waddell, N.: Deep learning in cancer diagnosis, prognosis and treatment selection. Genome Medicine **13**, 1–17 (2021)
80. Ugare, S., Banerjee, D., Misailovic, S., Singh, G.: Incremental verification of neural networks. Proceedings of the ACM on Programming Languages **7**(PLDI), 1920–1945 (2023)
81. Usman, M., Gopinath, D., Sun, Y., Noller, Y., Păsăreanu, C.S.: Nn repair: Constraint-based repair of neural network classifiers. In: Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I 33. pp. 3–25. Springer (2021)
82. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. arXiv preprint arXiv:1804.10829 (2018)
83. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, Z.: Beta-crown: efficient bound propagation with per-neuron split constraints for neural network robustness verification. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21, Curran Associates Inc., Red Hook, NY, USA (2024)
84. Webb, S., Rainforth, T., Teh, Y.W., Kumar, M.P.: A statistical approach to assessing neural network robustness. arXiv preprint arXiv:1811.07209 (2018)
85. Weng, L., Chen, P.Y., Nguyen, L., Squillante, M., Boopathy, A., Oseledets, I., Daniel, L.: Proven: Verifying robustness of neural networks with a probabilistic approach. In: International Conference on Machine Learning. pp. 6727–6736. PMLR (2019)
86. Wu, H., Isac, O., Zeljić, A., Tagomori, T., Daggitt, M., Kokke, W., Refaeli, I., Amir, G., Julian, K., Bassan, S., et al.: Marabou 2.0: a versatile formal analyzer of neural networks. In: International Conference on Computer Aided Verification. pp. 249–264. Springer (2024)
87. Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.W., Huang, M., Kailkhura, B., Lin, X., Hsieh, C.J.: Automatic perturbation analysis for scalable certified robustness and beyond. Advances in Neural Information Processing Systems **33**, 1129–1141 (2020)

88. Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., Hsieh, C.J.: Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers (2021), https://arxiv.org/abs/2011.13824
89. Yang, X., Yamaguchi, T., Tran, H.D., Hoxha, B., Johnson, T.T., Prokhorov, D.: Neural network repair with reachability analysis. In: International Conference on Formal Modeling and Analysis of Timed Systems. pp. 221–236. Springer International Publishing Cham (2022)
90. Yang, X., Yamaguchi, T., Tran, H.D., Hoxha, B., Johnson, T.T., Prokhorov, D.: Reachability analysis of convolutional neural networks. arXiv preprint arXiv:2106.12074 (2021)
91. Yang, X., Ye, Y., Li, X., Lau, R.Y., Zhang, X., Huang, X.: Hyperspectral image classification with deep learning models. IEEE Transactions on Geoscience and Remote Sensing **56**(9), 5408–5423 (2018)
92. Zhang, C., Ruan, W., Xu, P.: Reachability analysis of neural network control systems. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 15287–15295 (2023)
93. Zhang, H., Weng, T.W., Chen, P.Y., Hsieh, C.J., Daniel, L.: Efficient neural network robustness certification with general activation functions. In: Advances in Neural Information Processing Systems. pp. 4944–4953 (2018)
94. Zhang, Y., Zhao, Z., Chen, G., Song, F., Chen, T.: BDD4BNN: a BDD-based quantitative analysis framework for binarized neural networks. In: International Conference on Computer Aided Verification. pp. 175–200. Springer (2021)