NNV: The Neural Network Verification Tool for Deep Neural Networks and Learning-Enabled Cyber-Physical Systems

Hoang-Dung Tran[1], Xiaodong Yang[1], Diego Manzanas Lopez[1], Patrick Musau[1], Luan Viet Nguyen[2], Weiming Xiang[1], Stanley Bak and Taylor T. Johnson[1]

[1] Institute for Software Integrated Systems, Vanderbilt University, TN, USA
[2] Department of Computer and Information Science, University of Pennsylvania, PA, USA

Abstract. This paper presents the Neural Network Verification (NNV) software tool, a set-based verification framework for deep neural networks (DNNs) and learning-enabled cyber-physical systems (CPS). The crux of NNV is a collection of reachability algorithms that make use of a variety of set representations, such as polyhedra, star sets, zonotopes, and abstract-domain representations. NNV supports both exact (sound and complete) and over-approximate (sound) reachability algorithms for verifying safety and robustness properties of feed-forward neural networks (FFNNs) with various activation functions. For learning-enabled CPS, such as closed-loop control systems incorporating neural networks, NNV provides exact and over-approximate reachability analysis schemes for linear plant models and FFNN controllers with piecewise-linear activation functions, such as ReLUs. For similar neural network control systems (NNCS) that instead have nonlinear plant models, NNV supports over-approximate analysis by combining the star set analysis used for FFNN controllers with zonotope-based analysis for nonlinear plant dynamics building on CORA. We evaluate NNV using two real-world case studies: the first is safety verification of ACAS Xu networks, and the second deals with the safety verification of a deep learning-based adaptive cruise control system.

1 Introduction

Deep neural networks (DNNs) have quickly become one of the most widely used tools for dealing with complex and challenging problems in numerous domains, such as image classification [10, 16, 25], function approximation, and natural language translation [11, 18]. Recently, DNNs have been used in safety-critical cyber-physical systems (CPS), such as autonomous vehicles [8, 9, 47] and air traffic collision avoidance systems [21]. Although utilizing DNNs in safety-critical applications can demonstrate considerable performance benefits, assuring the safety and robustness of these systems is challenging because DNNs possess complex non-linear characteristics. Moreover, it has been demonstrated that

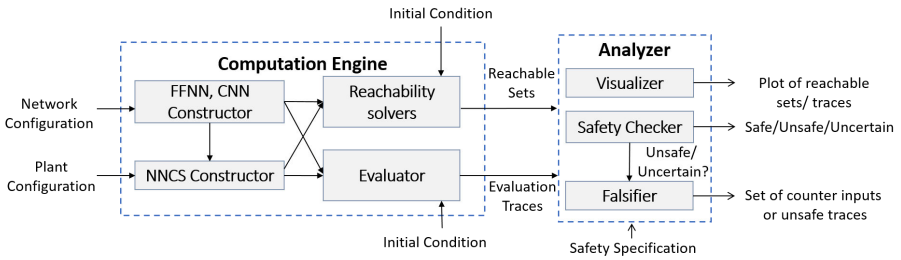

Fig. 1: An overview of NNV and its major modules and components.

their behavior can be unpredictable due to slight perturbations in their inputs (i.e., adversarial perturbations) [35].

In this paper, we introduce the NNV (**N**eural **N**etwork **V**erification) tool, which is a software framework that performs set-based verification for DNNs and learning-enabled CPS, known colloquially as neural network control systems (NNCS) as shown in Figure 2[3]. NNV provides a set of reachability algorithms that can compute both the exact and over-approximate reachable sets of DNNs and NNCSs using a variety of set representations such as polyhedra [38, 48–51], star sets [28,36,37,39], zonotopes [31], and abstract domain representations [32]. The reachable set obtained from NNV contains all possible states of a DNN from bounded input sets or of a NNCS from sets of initial states of a plant model. NNV declares a DNN or a NNCS to be safe if, and only if, their reachable sets do not violate safety properties (i.e., have a non-empty intersection with any state satisfying the negation of the safety property). If a safety property is violated, NNV can construct a complete set of counter-examples demonstrating the set of all possible unsafe initial inputs and states by using the star-based exact reachability algorithm [36, 39]. To speed up computation, NNV uses parallel computing, as the majority of the reachability algorithms in NNV are more efficient when executed on multi-core platforms and clusters.

NNV has been successfully applied to safety verification and robustness analysis of several real-world DNNs, primarily feedforward neural networks (FFNNs) and convolutional neural networks (CNNs), as well as learning-enabled CPS. To highlight NNV's capabilities, we present brief experimental results from two case studies. The first compares methods for safety verification of the ACAS Xu networks [21], and the second presents safety verification of a learning-based adaptive cruise control (ACC) system.

[3] The source code for NNV is publicly available: https://github.com/verivital/nnv/. A CodeOcean capsule is also available: https://doi.org/10.24433/CO.1314285.v1, which will be updated with a new DOI and the latest reproducibility results if accepted. The latest version of the CodeOcean capsule with all aspects described in this paper is available at: https://codeocean.com/capsule/1314285/, which requires a username (taylor.johnson@uta.edu) and password (cav2020ae) to access. This account has read-only permission, so to rerun the results shown in the capsule, you can select Capsule then Duplicate from the menu bar, which will clone the capsule to allow rerunning and editing if desired. Detailed instructions for the artifact evaluation are available at: https://github.com/verivital/run_nnv_comparison/blob/cav2020/README_AE.md

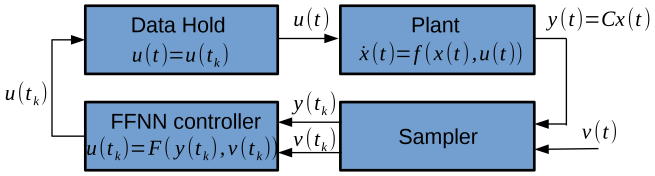

Fig. 2: Architecture of a typical neural network control system (NNCS).

2 Overview and Features

NNV is an object-oriented toolbox written in Matlab, which was chosen in part due to the prevalence of Matlab/Simulink in the design of CPS. It uses the MPT toolbox [26] for polytope-based reachability analysis and visualization [38], and makes use of CORA [3] for zonotope-based reachability analysis of nonlinear plant models [36]. NNV also utilizes the Neural Network Model Transformation Tool (NNMT) for transforming neural network models from Keras and Tensorflow into Matlab using the Open Neural Network Exchange (ONNX) format, and the Hybrid Systems Model Transformation and Translation tool (HyST) [5] for plant configuration.

The NNV toolbox contains two main modules: a *computation engine* and an *analyzer*, shown in Figure 1. The computation engine module consists of four subcomponents: 1) the *FFNN constructor*, 2) the *NNCS constructor*, 3) *the reachability solvers*, and 4) *the evaluator*. The FFNN constructor takes a network configuration file as an input and generates a FFNN object. The NNCS constructor takes the FFNN object and the plant configuration, which describes the dynamics of a system, as inputs and then creates an NNCS object. Depending on the application, either the FFNN (or NNCS) object will be fed into a reachability solver to compute the reachable set of the FFNN (or NNCS) from a given initial set of states. Then, the obtained reachable set will be passed to the analyzer module. The analyzer module consists of three subcomponents: 1) a *visualizer*, 2) a *safety checker*, and 3) a *falsifier*. The visualizer can be called to plot the obtained reachable set. Given a safety specification, the safety checker can reason about the safety of the FFNN or NNCS with respect to the specification. When an exact (sound and complete) reachability solver is used, such as the star-based solver, the safety checker can return either "safe," or "unsafe" along with a set of counterexamples. When an over-approximate (sound) reachability solver is used, such as the zonotope-based scheme or the approximate star-based solvers, the safety checker can return either "safe" or "*uncertain*" (unknown). In this case, the falsifier automatically calls the evaluator to generate simulation traces to find a counterexample. If the falsifier can find a counterexample, then NNV returns unsafe. Otherwise, it returns unknown. A summary of NNV's major features is given in Table 1.

3 Set Representations and Reachability Algorithms

NNV implements a set of reachability algorithms for *sequential* FFNNs and CNNs, as well as NNCS with FFNN controllers as shown in Figure 2. The reachable set of a sequential FFNN is computed layer-by-layer. The output reachable set of a layer is the input set of the next layer in the network.

Feature	Exact Analysis	Over-approximate Analysis
Components	FFNN, CNN, NNCS	FFNN, CNN, NNCS
Plant dynamics (for NNCS)	Linear ODE	Linear ODE, Nonlinear ODE
Discrete/Continuous (for NNCS)	Discrete Time	Discrete Time, Continuous Time
Activation functions	ReLU, Satlin	ReLU, Satlin, Sigmoid, Tanh
CNN Layers	MaxPool, Conv, BN, AvgPool, FC	MaxPool, Conv, BN, AvgPool, FC
Reachability methods	Star, Polyhedron, ImageStar	Star, Zonotope, Abstract-domain, ImageStar
Reachable set/Flow-pipe Visualization	Yes	Yes
Parallel computing	Yes	Partially supported
Safety verification	Yes	Yes
Falsification	Yes	Yes
Robustness verification (for FFNN/CNN)	Yes	Yes
Counterexample generation	Yes	Yes

Table 1: Overview of major features available in NNV. Links refer to relevant files/classes in the NNV codebase. BN refers to batch normalization layers, FC to fully-connected layers, AvgPool to average pooling layers, Conv to convolutional layers, and MaxPool to max pooling layers.

3.1 Polyhedron [38]

The polyhedron reachability algorithm computes the exact polyhedron reachable set of a FFNN with ReLU activation functions. The exact reachability computation of layer L in a FFNN is done as follows. First, we construct the affine mapping $\bar{I}$ of the input polyhedron set I, using the weight matrix W and the bias vector b, i.e., $\bar{I} = W \times I + b$. Then, the exact reachable set of the layer R_L is constructed by executing a sequence of stepReLU operations, i.e., $R_L = stepReLU_n(stepReLU_{n-1}(\cdots(stepReLU_1(\bar{I}))))$. Since a $stepReLU$ operation can split a polyhedron into two new polyhedra, the exact reachable set of a layer in a FFNN is usually a union of polyhedra. The polyhedron reachability algorithm is computationally expensive because computing affine mappings with polyhedra is costly. Additionally, when computing the reachable set, the polyhedron approach extensively uses the expensive conversion between the H-representation and the V-representation. These are the main drawbacks that limit the scalability of the polyhedron approach. Despite that, we extend the polyhedron reachability algorithm for NNCSs with FFNN controllers. However, the propagation of polyhedra in NNCS may lead to a large degree of conservativeness in the computed reachable set [36].

3.2 Star Set [36, 39] (code)

The star set is an efficient set representation for simulation-based verification of large linear systems [6,7,40] where the superposition property of a linear system can be exploited in the analysis. It has been shown in [39] that the star set is also suitable for reachability analysis of FFNNs. In contrast to polyhedra, the affine mapping and intersection with a half space of a star set is more easily computed. NNV implements an enhanced version of the exact and over-approximate reachability algorithms for FFNNs proposed in [39] by minimizing the number of LP optimization problems that need to be solved in the computation. The exact algorithm that makes use of star sets is similar to the polyhedron method

that makes use of *stepReLU* operations. However, it is much faster and more scalable than the polyhedron method because of the advantage that star sets have in affine mapping and intersection. The approximate algorithm obtains an over-approximation of the exact reachable set by approximating the exact reachable set after applying an activation function, e.g., ReLU, Tanh, Sigmoid. We refer readers to [39] for a detailed discussion of star-set reachability algorithms for FFNNs.

We note that NNV implements enhanced versions of earlier star-based reachability algorithms [39]. Particularly, we minimize the number of linear programming (LP) optimization problems that must be solved in order to construct the reachable set of a FFNN by quickly estimating the ranges of all of the states in the star set using only the ranges of the predicate variables. Additionally, the extensions of the star reachability algorithms to NNCS with linear plant models can eliminate the explosion of conservativeness in the polyhedron method [36,37]. The reason behind this is that in star sets, the relationship between the plant state variables and the control inputs is preserved in the computation since they are defined by a unique set of predicate variables. We refer readers to [36,37] for a detailed discussion of the extensions of the star-based reachability algorithms for NNCSs with linear/nonlinear plant models.

3.3 Zonotope [31] (code)

NNV implements the zonotope reachability algorithms proposed in [31] for FFNNs. Similar to the over-approximate algorithm using star sets, the zonotope algorithm computes an over-approximation of the exact reachable set of a FFNN. Although the zonotope reachability algorithm is very fast and scalable, it produces a very conservative reachable set in comparison to the star set method as shown in [39]. Consequently, zonotope-based reachability algorithms are usually only more efficient for very small input sets. As an example it can be more suitable for robustness certification.

3.4 Abstract Domain [32]

NNV implements the abstract domain reachability algorithm proposed in [32] for FFNNs. NNV's abstract domain reachability algorithm specifies an abstract domain as a star set and uses a "back-tracking" approach to estimate the *over-approximate ranges* of the states. The abstract domain is more conservative than the star set method.

3.5 ImageStar Set (code)

NNV recently introduced a new set representation called the ImageStar for use in the verification of deep convolutional neural networks (CNNs). Briefly, the ImageStar is a generalization of the star set where the anchor and generator vectors are replaced by multi-channel images. The ImageStar is efficient in the analysis of convolutional layers, average pooling layers, and fully connected layers,

whereas max pooling layers and ReLU layers consume most of computation time in reachability analysis of CNNs. NNV implements exact and over-approximate reachability algorithms using the ImageStar for serial CNNs. Since the ImageStar method has not been published yet, we defer its evaluation in our experimental evaluation. In short, using the ImageStar, we can analyze the robustness under adversarial attacks of the real-world VGG16 and VGG19 deep perception networks [30] that consist of $>$ 100 million parameters.

4 Evaluation

The experiments presented in this section were performed on a desktop with the following configuration: Intel Core i7-6700 CPU @ 3.4GHz 8 core Processor, 64 GB Memory, and 64-bit Ubuntu 16.04.3 LTS OS.

4.1 Safety verification of ACAS Xu networks

We evaluate NNV in comparison to Reluplex [22], Marabou [23], and ReluVal [44], by considering the verification of safety property ϕ_3, and ϕ_4 of the ACAS Xu neural networks [21] for all 45 networks.[4] All the experiments were done using 4 cores for the computation. The verification results are summarized in Table 2 where (SAT) denotes that the networks are safe, (UNSAT) denotes unsafe, and (UNK) denotes unknown. We note that (UNK) may occur due to the conservativeness of the reachability analysis scheme. Detailed verification results are presented in the appendix. For a fast comparison with other tools, we also tested a subset input of Property 1-4 on all the 45 networks. The results are also shown in the appendix. We note that the polyhedron method [38] achieves a timeout on most of networks, and therefore, we neglect this method in the comparison.

Verification time. For property ϕ_3, our exact-star method is about 20.7$\times$ faster than Reluplex, 14.2$\times$ faster than Marabou, 81.6$\times$ faster than Marabou-DnC (i.e., divide and conquer method). The approximate star method is 547$\times$ faster than Reluplex, 374$\times$ faster than Marabou, 2151$\times$ faster than Marabou-DnC, and 8$\times$ faster than ReluVal. For property ϕ_4, our exact-star method is 25.3$\times$ faster than Reluplex, 18.0$\times$ faster than Marabou, 53.4$\times$ faster than Marabou-DnC, while the approximate star method is 625$\times$ faster than Reluplex, 445$\times$ faster than Marabou, 1321$\times$ faster than Marabou-DnC.

Conservativeness. The approximate star method is much less conservative than the zonotope and abstract domain methods. This is illustrated since it can verify more networks than the zonotope and abstract domain methods, and is because it obtains a tighter over-approximate reachable set. For property ϕ_3, the zonotope and abstract domain methods can prove the safety of 2/45 networks, (4.44%) and 19/45 networks, (42.22%) respectively, while our approximate star

[4] We omitted properties ϕ_1 and ϕ_2 for space and due to their long runtimes, but they can be reproduced in the artifact evaluation if desired.

ACAS XU ϕ_3	**SAT**	**UNSAT**	**UNK**	**TIMEOUT**			**TIME(s)**
				1h	**2h**	**10h**	
Reluplex	3	42	0	2	0	0	28454
Marabou	3	42	0	1	0	0	19466
Marabou DnC	3	42	0	3	3	1	111880
ReluVal	3	42	0	0	0	0	416
Zonotope	0	2	43	0	0	0	3
Abstract Domain	0	10	35	0	0	0	72
NNV Exact Star	3	42	0	0	0	0	1371
NNV Appr. Star	0	29	16	0	0	0	52
ACAS XU ϕ_4							
Reluplex	3	42	0	0	0	0	11880
Marabou	3	42	0	0	0	0	8470
Marabou DnC	3	42	0	2	2	0	25110
ReluVal	3	42	0	0	0	0	27
Zonotope	0	1	44	0	0	0	5
Abstract Domain	0	0	45	0	0	0	7
NNV Exact Star	3	42	0	0	0	0	470
NNV Appr. Star	0	32	13	0	0	0	19

Table 2: Verification results of ACAS Xu networks.

method can prove the safety of 29/45 networks, (64.4%). For property ϕ_4, the zonotope and abstract domain method can prove the safety of 1/45 networks, (2.22%) and 0/45 networks, (0.00%) respectively while the approximate star method can prove the safety of 32/45, (71.11%).

4.2 Safety Verification of Adaptive Cruise Control System

To illustrate how NNV can be used to verify/falsify safety properties of learning-enabled CPS, we analyze a learning-based ACC system depicted in Figure 3, in which the ego vehicle has a radar sensor to measure the distance to the lead vehicle in the same lane, D_{rel}, as well as the relative velocity of the lead vehicle, V_{rel}. The ego vehicle has two control modes. In speed control mode, it travels at a driver-specified set speed $V_{set} = 30$, and in spacing control mode, it maintains a safe distance from the lead vehicle, D_{safe}. We train a neural network with 5 layers, 20 neurons per layer utilizing the ReLU activation function to control the ego vehicle with a control period of 0.1 seconds.

We investigate safety of the learning-based ACC system with two types of plant dynamics: 1) a discrete linear plant, and 2) a nonlinear continuous plant governed by the following differential equations:

$$\dot{x}_{lead}(t) = v_{lead}(t),\ \dot{v}_{lead}(t) = \gamma_{lead}, \quad \dot{\gamma}_{lead}(t) = -2\gamma_{lead}(t) + 2a_{lead} - \mu v_{lead}^2(t),$$
$$\dot{x}_{ego}(t) = v_{ego}(t),\ \dot{v}_{ego}(t) = \gamma_{ego}, \qquad \dot{\gamma}_{ego}(t) = -2\gamma_{ego}(t) + 2a_{ego} - \mu v_{ego}^2(t),$$

where $x_{lead}(x_{ego})$, $v_{lead}(v_{ego})$ and $\gamma_{lead}(\gamma_{ego})$ are the position, velocity and acceleration of the lead (ego) vehicle respectively. $a_{lead}(a_{ego})$ is the acceleration control input applied to the lead (ego) vehicle, and $\mu = 0.0001$ is a friction parameter. To obtain a discrete linear model of the plant, we let $\mu = 0$ and

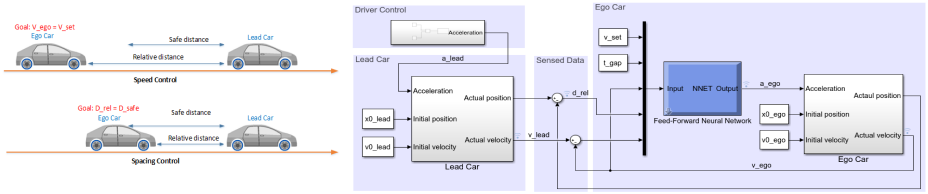

Fig. 3: Learning-based Adaptive Cruise Control System [1].

discretize the corresponding linear continuous model using a zero-order hold on the inputs with a sample time of 0.1 seconds (i.e., the control period).

Verification Problem. The scenario we are interested in is when the two vehicles are operating at a safe distance between them and the ego vehicle is in speed control mode. In this state the lead vehicle driver suddenly decelerates with $a_{lead} = -5$ to reduce the speed. We want to verify if the neural network controller on the ego vehicle will also de-accelerate to maintain a safe distance between the two vehicles. To guarantee safety, we require that $D_{rel} = x_{lead} - x_{ego} \geq D_{safe} = D_{default} + T_{gap} \times v_{ego}$ where $T_{gap} = 1.4$ seconds and $D_{default} = 10$. Our analysis investigates if the safety requirement holds in the 5 seconds after the lead vehicle decelerates. We consider the safety of the system under the following initial conditions: $x_{lead}(0) \in [90, 92]$, $v_{lead}(0) \in [20, 30]$, $\gamma_{lead}(0) = \gamma_{ego}(0) = 0$, $v_{ego}(0) \in [30, 30.5]$, $x_{ego} \in [30, 31]$.

Verification results. For linear dynamics, NNV can compute both the exact and over-approximate reachable sets of the ACC system in bounded time steps, while for nonlinear dynamics, NNV constructs an over-approximation of the exact reachable sets and uses it for safety verification. The verification results for linear and nonlinear models using the over-approximate star method are presented in Table 3, which shows that, the safety of the ACC system depends on the initial velocity of the lead vehicle. When the initial velocity of the lead vehicle is smaller than $27(m/s)$, the ACC system with the discrete plant model is unsafe. Using the exact star method, NNV can construct a *complete* set of counter-example inputs. When the over-approximate star method is used, if there is a potential safety violation, NNV simulates the system with 1000 random inputs from the input set to find counter examples. If a counterexample is found, the system is *UNSAFE*, otherwise, NNV returns a safety result of *UNKNOWN*. Figure 4 visualizes the reachable sets of the relative distance D_{rel} between two vehicles versus the required safe distance D_{safe} over time for two cases of initial velocities of the lead vehicle: $v_{lead}(0) \in [29, 30]$ and $v_{lead}(0) \in [24, 25]$. We can see that in the first case, $D_{ref} \geq D_{safe}$ for all 50 time steps stating that the system is safe. In the second case, $D_{ref} < D_{safe}$ in some control steps which means that the system is unsafe. NNV supports a *reachLive* method to perform analysis and reachable set visualization on-the-fly to help the user observe the behavior of the system during verification.

The verification results for ACC system with the nonlinear model are all $UNSAFE$, which is surprising. Since the neural network controller of the ACC system was trained with the linear model, it works quite well for the linear model. However, when a small friction term is added to the linear model to form a

v_lead(0)	Linear Plant		Nonlinear Plant	
	$Safety$	$VT(s)$	$Safety$	$VT(s)$
[29, 30]	SAFE	9.60	UNSAFE	346.62
[28, 29]	SAFE	9.45	UNSAFE	277.50
[27, 28]	SAFE	9.82	UNSAFE	289.70
[26, 27]	UNSAFE	17.80	UNSAFE	315.60
[25, 26]	UNSAFE	19.24	UNSAFE	305.56
[24, 25]	UNSAFE	18.12	UNSAFE	372.00

Table 3: Verification results for ACC system with different plant models, where VT is the verification time (in seconds).

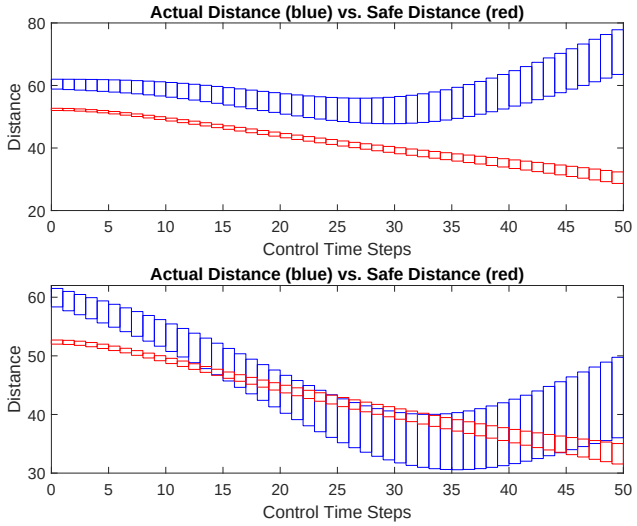

Fig. 4: Two scenarios of the ACC system. In the first (top) scenario ($v_{lead}(0) \in [29,\ 30]m/s$), safety is guaranteed, $D_{rel} \geq D_{safe}$. In the second scenario (bottom) ($v_{lead}(0) \in [24,\ 25]m/s$), safety is violated since $D_{ref} < D_{safe}$ in some control steps.

nonlinear model, the neural network controller's performance, in terms of safety, is significantly reduced. This problem raises an important issue in training neural network controllers using simulation data, and these schemes may not work in real systems since there is always a mismatch between the plant model in the simulation engine and the real system.

Verification times. As shown in Table 3, the approximate analysis of the ACC system with discrete linear plant model is very fast. It can be done in 84 seconds. We note that NNV also supports exact analysis, which is computationally expensive since it constructs all reachable sets of the system. Because there are splits in the reachable sets of the neural network controller, the number of star sets in the reachable set of the plant increases quickly over time [36]. In contrast, the over-approximate method computes the interval hull of all reachable star sets at each time step. It maintains a single reachable set of the plant throughout the computation. Therefore, the over-approximate method is much

faster than the exact method. In terms of plant models, the nonlinear model requires more computation time than the linear one. As shown in Table 3, the verification for linear model using the over-approximate method is 22.7× faster on average than the verification of the nonlinear model.

5 Related Work

NNV was inspired by many insightful research works in the emerging fields of neural network and machine learning verification. For the "open-loop" verification problem (verification of DNNs), many efficient techniques have been proposed, such as SMT-based methods [22, 23, 29], mixed-integer linear programming methods [14, 24, 27], set-based methods [4, 17, 31, 32, 43, 45, 49], and optimization methods [46,52]. For the "closed-loop" verification problem (NCCS verification), we note that the Verisig approach [20] is very efficient for NNCS with nonlinear plants and with Sigmoid and Tanh activation functions. Additionally, the recent regressive polynomial rule inference approach [33] is very fast for the safety verification of NNCS with nonlinear plant models and ReLU activation functions. The satisfiability modulo convex (SMC) approach [34] is also very promising for NNCS with discrete linear plants as it provides both soundness and completeness properties in verification. ReachNN [19] is a recent approach that can efficiently control the conservativeness in the reachability analysis of NNCS with nonlinear plants and ReLU, Sigmoid and Tanh activation functions in the controller. In other learning-enabled systems, falsification and testing-based approaches [12,13,41] have shown a significant promise in enhancing the safety of systems where perception components and neural network controllers interact with the physical world. Finally, there is significant related work in the domain of safe reinforcement learning [2,15,42,53] and combining guarantees from NNV with those provided in these methods would be interesting to explore.

6 Conclusion and Future Work

We have presented NNV, a toolbox for the verification of DNNs and learning-enabled CPS. Our tool provides a collection of reachability algorithms that can be used to verify the safety (and robustness) of real-world DNNs as well as learning-enabled CPS, such as the ACC system. Our method is comparable to existing methods such as Reluplex and Marabou when dealing with the open-loop verification problem. For closed-loop systems, NNV can compute the exact and over-approximate reachable sets of a NNCS with linear plant models. For a NNCS with a nonlinear plant, NNV can obtain an over-approximate reachable set and use it to verify the safety, but can also automatically falsify the system to construct/find counterexamples (using exact analysis) or randomized simulations (in over-approximate analysis).

References

1. 2019b, M.: Model Predictive Control Toolbox. The MathWorks Inc., Natick, Massachusetts (2019), https://www.mathworks.com/help/mpc/ug/adaptive-cruise-control-using-model-predictive-controller.html
2. Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., Topcu, U.: Safe reinforcement learning via shielding. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
3. Althoff, M.: An introduction to cora 2015. In: Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems (2015)
4. Anderson, G., Pailoor, S., Dillig, I., Chaudhuri, S.: Optimization and abstraction: A synergistic approach for analyzing neural network robustness. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. p. 731744. PLDI 2019, Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3314221.3314614
5. Bak, S., Bogomolov, S., Johnson, T.T.: Hyst: a source transformation and translation tool for hybrid automaton models. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control. pp. 128–133. ACM (2015)
6. 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)
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. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
9. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: Learning affordance for direct perception in autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2722–2730 (2015)
10. Cireşan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745 (2012)
11. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th international conference on Machine learning. pp. 160–167. ACM (2008)
12. Dreossi, T., Donzé, A., Seshia, S.A.: Compositional falsification of cyber-physical systems with machine learning components. In: NASA Formal Methods Symposium. pp. 357–372. Springer (2017)
13. Dreossi, T., Fremont, D.J., Ghosh, S., Kim, E., Ravanbakhsh, H., Vazquez-Chanlatte, M., Seshia, S.A.: Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In: Dillig, I., Tasiran, S. (eds.) Computer Aided Verification. pp. 432–442. Springer International Publishing, Cham (2019)
14. Dutta, S., Jha, S., Sanakaranarayanan, S., Tiwari, A.: Output range analysis for deep neural networks. arXiv preprint arXiv:1709.09130 (2017)
15. Fulton, N., Platzer, A.: Verifiably safe off-model reinforcement learning. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 413–430. Springer International Publishing, Cham (2019)
16. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2414–2423 (2016)

17. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai 2: Safety and robustness certification of neural networks with abstract interpretation. In: Security and Privacy (SP), 2018 IEEE Symposium on (2018)
18. Goldberg, Y.: A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research 57, 345–420 (2016)
19. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: Reachnn: Reachability analysis of neural-network controlled systems. arXiv preprint arXiv:1906.10654 (2019)
20. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Hybrid Systems: Computation and Control (HSCC) (2019)
21. Julian, K.D., Lopez, J., Brush, J.S., Owen, M.P., Kochenderfer, M.J.: Policy compression for aircraft collision avoidance systems. In: Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th. pp. 1–10. IEEE (2016)
22. 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)
23. 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)
24. Kouvaros, P., Lomuscio, A.: Formal verification of cnn-based perception systems. arXiv preprint arXiv:1811.11373 (2018)
25. 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)
26. Kvasnica, M., Grieder, P., Baotić, M., Morari, M.: Multi-parametric toolbox (mpt). In: International Workshop on Hybrid Systems: Computation and Control. pp. 448–462. Springer (2004)
27. Lomuscio, A., Maganti, L.: An approach to reachability analysis for feed-forward relu neural networks. arXiv preprint arXiv:1706.07351 (2017)
28. Lopez, D.M., Musau, P., Tran, H.D., Johnson, T.T.: Verification of closed-loop systems with neural network controllers. In: Frehse, G., Althoff, M. (eds.) ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems. EPiC Series in Computing, vol. 61, pp. 201–210. EasyChair (April 2019), https://easychair.org/publications/paper/ZmnC
29. Pulina, L., Tacchella, A.: An abstraction-refinement approach to verification of artificial neural networks. In: International Conference on Computer Aided Verification. pp. 243–257. Springer (2010)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
31. Singh, G., Gehr, T., Mirman, M., Püschel, M., Vechev, M.: Fast and effective robustness certification. In: Advances in Neural Information Processing Systems. pp. 10825–10836 (2018)
32. 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)
33. Souradeep Dutta, Xin Chen, S.S.: Reachability analysis for neural feedback systems using regressive polynomial rule inference. In: Hybrid Systems: Computation and Control (HSCC) (2019)

34. Sun, X., Khedr, H., Shoukry, Y.: Formal verification of neural network controlled autonomous systems. In: Hybrid Systems: Computation and Control (HSCC) (2019)
35. 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)
36. 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)
37. Tran, H.D., Cei, F., Lopez, D.M., Johnson, T.T., Koutsoukos, X.: Safety verification of cyber-physical systems with reinforcement learning control (July 2019)
38. Tran, H.D., Musau, P., Lopez, D.M., Yang, X., Nguyen, L.V., Xiang, W., Johnson, T.T.: Parallelizable reachability analysis algorithms for feed-forward neural networks. In: 7th International Conference on Formal Methods in Software Engineering (FormaliSE2019), Montreal, Canada (2019)
39. 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)
40. Tran, H.D., Nguyen, L.V., Hamilton, N., Xiang, W., Johnson, T.T.: Reachability analysis for high-index linear differential algebraic equations (daes). In: 17th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'19). Springer International Publishing (August 2019)
41. Tuncali, C.E., Fainekos, G., Ito, H., Kapinski, J.: Simulation-based adversarial test generation for autonomous vehicles with machine learning components. arXiv preprint arXiv:1804.06760 (2018)
42. Verma, A., Murali, V., Singh, R., Kohli, P., Chaudhuri, S.: Programmatically interpretable reinforcement learning. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 5045–5054. PMLR, Stockholmsmssan, Stockholm Sweden (10–15 Jul 2018), http://proceedings.mlr.press/v80/verma18a.html
43. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. In: Advances in Neural Information Processing Systems. pp. 6369–6379 (2018)
44. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Formal security analysis of neural networks using symbolic intervals. In: 27th USENIX Security Symposium (USENIX Security 18). USENIX Association, Baltimore, MD (2018), https://www.usenix.org/conference/usenixsecurity18/presentation/wang-shiqi
45. 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)
46. Weng, T.W., Zhang, H., Chen, H., Song, Z., Hsieh, C.J., Boning, D., Dhillon, I.S., Daniel, L.: Towards fast computation of certified robustness for relu networks. arXiv preprint arXiv:1804.09699 (2018)
47. Wu, B., Iandola, F.N., Jin, P.H., Keutzer, K.: Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: CVPR Workshops. pp. 446–454 (2017)
48. Xiang, W., Tran, H.D., Johnson, T.T.: Reachable set computation and safety verification for neural networks with relu activations. arXiv preprint arXiv:1712.08163 (2017)

49. Xiang, W., Tran, H.D., Johnson, T.T.: Output reachable set estimation and verification for multilayer neural networks. IEEE transactions on neural networks and learning systems (99), 1–7 (2018)
50. Xiang, W., Tran, H.D., Johnson, T.T.: Specification-guided safety verification for feedforward neural networks. AAAI Spring Symposium on Verification of Neural Networks (2019)
51. Xiang, W., Tran, H.D., Rosenfeld, J.A., Johnson, T.T.: Reachable set estimation and safety verification for piecewise linear systems with neural network controllers. arXiv preprint arXiv:1802.06981 (2018)
52. 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)
53. Zhu, H., Xiong, Z., Magill, S., Jagannathan, S.: An inductive synthesis framework for verifiable reinforcement learning. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. p. 686701. PLDI 2019, Association for Computing Machinery, New York, NY, USA (2019), https://doi.org/10.1145/3314221.3314638

A Appendix: Additional Evaluation Details

Figures 5 and 6 show detailed comparisons between NNV's approximate and exact star methods relative to the zonotope, abstract domain, Reluplex, Marabou, and Marabou divide-and-conquer (DnC) methods, summarized earlier in Table 2.

ID	Zonotope		Abstract Domain		Reluplex		Marabou		Marabou DnC		ReluVal		NNV Exact Star			NNV Appr. Star	
	VT	V	VT	V	VT	V	VT	V	VT	V	VT	V	N_p	VT	V	VT	V
N_{11}	0.05	UNK	1.96	UNSAT	6156	UNSAT	3637	UNSAT	73560	UNSAT	96.83	UNSAT	70631	398.50	UNSAT	1.89	UNK
N_{12}	0.06	UNK	2.31	UNSAT	4942	UNSAT	3418	UNSAT	3221	UNSAT	1.87	UNSAT	39529	205.88	UNSAT	1.86	UNK
N_{13}	0.07	UNK	2.42	UNSAT	1134	UNSAT	1795	UNSAT	10858	UNSAT	3.22	UNSAT	12274	45.62	UNSAT	2.16	UNK
N_{14}	0.06	UNK	1.73	UNK	528	UNSAT	187	UNSAT	186	UNSAT	0.27	UNSAT	4275	12.29	UNSAT	1.04	UNSAT
N_{15}	0.07	UNK	1.68	UNK	317	UNSAT	124	UNSAT	193	UNSAT	0.11	UNSAT	4744	16.45	UNSAT	0.96	UNSAT
N_{16}	0.06	UNK	0.99	UNSAT	64	UNSAT	37	UNSAT	23	UNSAT	0.04	UNSAT	1266	5.70	UNSAT	0.79	UNSAT
N_{17}	0.07	UNK	0.73	UNK	1	SAT	5	SAT	6	SAT	0.03	SAT	500	2.76	SAT	0.64	UNK
N_{18}	0.05	UNK	0.62	UNK	3	SAT	2	SAT	6	SAT	0.03	SAT	393	2.39	SAT	0.45	UNK
N_{19}	0.06	UNK	0.52	UNK	2	SAT	2	SAT	6	SAT	0.03	SAT	290	2.18	SAT	0.42	UNK
N_{21}	0.06	UNK	2.36	UNK	1208	UNSAT	1001	UNSAT	813	UNSAT	18.46	UNSAT	16179	54.05	UNSAT	1.79	UNK
N_{22}	0.06	UNK	1.64	UNK	653	UNSAT	480	UNSAT	2106	UNSAT	7.22	UNSAT	6861	20.62	UNSAT	1.39	UNK
N_{23}	0.06	UNK	1.93	UNK	1043	UNSAT	1103	UNSAT	1163	UNSAT	2.73	UNSAT	10614	32.19	UNSAT	1.69	UNK
N_{24}	0.05	UNK	1.01	UNSAT	41	UNSAT	22	UNSAT	8	UNSAT	0.48	UNSAT	345	2.43	UNSAT	0.55	UNSAT
N_{25}	0.06	UNK	1.82	UNK	235	UNSAT	58	UNSAT	13	UNSAT	0.31	UNSAT	2456	8.38	UNSAT	1.17	UNSAT
N_{26}	0.07	UNK	0.82	UNSAT	79	UNSAT	23	UNSAT	8	UNSAT	0.03	UNSAT	253	2.45	UNSAT	0.69	UNSAT
N_{27}	0.06	UNK	1.53	UNSAT	116	UNSAT	57	UNSAT	9	UNSAT	0.20	UNSAT	1199	5.29	UNSAT	1.01	UNSAT
N_{28}	0.06	UNK	1.03	UNSAT	83	UNSAT	13	UNSAT	8	UNSAT	0.04	UNSAT	326	2.58	UNSAT	0.50	UNSAT
N_{29}	0.05	UNSAT	0.38	UNSAT	27	UNSAT	2	UNSAT	7	UNSAT	0.01	UNSAT	188	2.13	UNSAT	0.36	UNSAT
N_{31}	0.06	UNK	2.41	UNSAT	177	UNSAT	97	UNSAT	240	UNSAT	2.58	UNSAT	5967	19.97	UNSAT	1.13	UNSAT
N_{32}	0.07	UNK	2.32	UNK	1561	UNSAT	1973	UNSAT	1564	UNSAT	5.27	UNSAT	36967	183.86	UNSAT	1.91	UNK
N_{33}	0.08	UNK	2.09	UNSAT	1105	UNSAT	344	UNSAT	891	UNSAT	0.26	UNSAT	7848	30.29	UNSAT	1.76	UNSAT
N_{34}	0.06	UNK	2.89	UNK	209	UNSAT	106	UNSAT	34	UNSAT	0.49	UNSAT	2077	9.35	UNSAT	1.96	UNK
N_{35}	0.06	UNK	1.50	UNK	83	UNSAT	41	UNSAT	10	UNSAT	5.01	UNSAT	1034	5.08	UNSAT	1.16	UNSAT
N_{36}	0.07	UNK	2.50	UNK	255	UNSAT	120	UNSAT	194	UNSAT	138.73	UNSAT	1857	8.04	UNSAT	1.86	UNK
N_{37}	0.05	UNK	0.76	UNSAT	35	UNSAT	8	UNSAT	7	UNSAT	0.10	UNSAT	107	1.95	UNSAT	0.56	UNSAT
N_{38}	0.07	UNK	1.34	UNK	179	UNSAT	57	UNSAT	25	UNSAT	3.51	UNSAT	654	3.66	UNSAT	1.04	UNSAT
N_{39}	0.08	UNK	1.54	UNK	118	UNSAT	55	UNSAT	35	UNSAT	2.36	UNSAT	1201	5.50	UNSAT	0.75	UNSAT
N_{41}	0.07	UNK	1.34	UNK	201	UNSAT	94	UNSAT	914	UNSAT	7.29	UNSAT	2276	8.68	UNSAT	1.20	UNK
N_{42}	0.05	UNK	2.13	UNK	2882	UNSAT	1724	UNSAT	11861	UNSAT	90.93	UNSAT	17894	68.22	UNSAT	1.67	UNK
N_{43}	0.07	UNK	2.05	UNK	1767	UNSAT	1257	UNSAT	1014	UNSAT	2.13	UNSAT	20746	98.76	UNSAT	1.79	UNK
N_{44}	0.07	UNK	1.51	UNK	86	UNSAT	19	UNSAT	8	UNSAT	0.11	UNSAT	560	3.38	UNSAT	0.94	UNSAT
N_{45}	0.06	UNK	0.90	UNSAT	35	UNSAT	14	UNSAT	8	UNSAT	0.09	UNSAT	351	2.34	UNSAT	0.77	UNSAT
N_{46}	0.05	UNK	2.78	UNSAT	300	UNSAT	118	UNSAT	55	UNSAT	0.14	UNSAT	2496	11.33	UNSAT	2.60	UNSAT
N_{47}	0.06	UNK	1.45	UNK	126	UNSAT	54	UNSAT	43	UNSAT	0.52	UNSAT	944	4.68	UNSAT	0.70	UNSAT
N_{48}	0.07	UNK	1.33	UNSAT	142	UNSAT	36	UNSAT	41	UNSAT	1.65	UNSAT	576	3.46	UNSAT	0.81	UNSAT
N_{49}	0.06	UNK	2.41	UNSAT	143	UNSAT	33	UNSAT	22	UNSAT	0.07	UNSAT	611	5.07	UNSAT	1.02	UNSAT
N_{51}	0.06	UNK	1.65	UNK	1131	UNSAT	764	UNSAT	1910	UNSAT	19.33	UNSAT	9461	29.50	UNSAT	1.47	UNK
N_{52}	0.05	UNK	1.15	UNK	151	UNSAT	120	UNSAT	635	UNSAT	1.94	UNSAT	2104	8.17	UNSAT	1.01	UNSAT
N_{53}	0.17	UNK	1.73	UNSAT	341	UNSAT	146	UNSAT	64	UNSAT	0.13	UNSAT	2874	9.08	UNSAT	1.41	UNSAT
N_{54}	0.06	UNK	1.53	UNK	62	UNSAT	42	UNSAT	10	UNSAT	0.13	UNSAT	758	4.50	UNSAT	1.06	UNSAT
N_{55}	0.07	UNK	1.83	UNK	86	UNSAT	27	UNSAT	18	UNSAT	0.41	UNSAT	1305	5.59	UNSAT	1.06	UNSAT
N_{56}	0.07	UNK	2.14	UNK	283	UNSAT	81	UNSAT	47	UNSAT	0.86	UNSAT	1143	5.73	UNSAT	1.12	UNSAT
N_{57}	0.07	UNSAT	0.43	UNSAT	35	UNSAT	5	UNSAT	8	UNSAT	0.02	UNSAT	88	1.81	UNSAT	0.34	UNSAT
N_{58}	0.05	UNK	1.58	UNSAT	310	UNSAT	157	UNSAT	22	UNSAT	0.06	UNSAT	2368	9.10	UNSAT	0.92	UNSAT
N_{59}	0.06	UNK	0.72	UNK	19	UNSAT	8	UNSAT	8	UNSAT	0.07	UNSAT	107	2.03	UNSAT	0.58	UNSAT

Fig. 5: Detailed verification results for ϕ_3 of ACAS Xu networks.

ID	Zonotope		Abstract		Reluplex		Marabou		Marabou DnC		ReluVal		NNV Exact Star			NNV Appr. Star	
	VT	*V*	*VT*	*V*	*VT*	*V*	*VT*	*V*	*VT*	*V*	*VT*	*V*	N_p	*VT*	*V*	*VT*	*V*
N_{11}	0.11	UNK	0.15	UNK	1291	UNSAT	1929	UNSAT	11379	UNSAT	1.08	UNSAT	19143	63.78	UNSAT	0.52	UNK
N_{12}	0.10	UNK	0.15	UNK	1267	UNSAT	2012	UNSAT	8629	UNSAT	1.13	UNSAT	13143	40.60	UNSAT	0.64	UNK
N_{13}	0.11	UNK	0.16	UNK	1150	UNSAT	956	UNSAT	2806	UNSAT	0.34	UNSAT	9837	31.32	UNSAT	0.71	UNK
N_{14}	0.10	UNK	0.16	UNK	107	UNSAT	79	UNSAT	93	UNSAT	0.18	UNSAT	1184	4.86	UNSAT	0.30	UNK
N_{15}	0.11	UNK	1.15	UNK	352	UNSAT	299	UNSAT	267	UNSAT	0.41	UNSAT	6608	22.23	UNSAT	0.33	UNK
N_{16}	0.11	UNK	0.15	UNK	219	UNSAT	201	UNSAT	82	UNSAT	0.18	UNSAT	4443	13.19	UNSAT	0.41	UNSAT
N_{17}	0.14	UNK	0.15	UNK	1	SAT	2	SAT	6	SAT	0.02	SAT	642	3.21	SAT	0.18	UNK
N_{18}	0.11	UNK	0.15	UNK	3	SAT	2	SAT	6	SAT	0.02	SAT	397	2.64	SAT	0.20	UNK
N_{19}	0.11	UNK	0.13	UNK	3	SAT	1	SAT	6	SAT	0.02	SAT	471	2.68	SAT	0.16	UNK
N_{21}	0.11	UNK	0.17	UNK	330	UNSAT	239	UNSAT	112	UNSAT	1.01	UNSAT	5066	14.80	UNSAT	0.51	UNK
N_{22}	0.11	UNK	0.15	UNK	415	UNSAT	273	UNSAT	210	UNSAT	1.85	UNSAT	4500	14.40	UNSAT	0.57	UNK
N_{23}	0.11	UNK	0.15	UNK	243	UNSAT	47	UNSAT	35	UNSAT	0.93	UNSAT	1087	4.58	UNSAT	0.27	UNSAT
N_{24}	0.11	UNK	0.14	UNK	86	UNSAT	23	UNSAT	16	UNSAT	0.12	UNSAT	913	4.65	UNSAT	0.37	UNSAT
N_{25}	0.11	UNK	0.16	UNK	151	UNSAT	95	UNSAT	46	UNSAT	0.31	UNSAT	3419	11.43	UNSAT	0.52	UNSAT
N_{26}	0.11	UNK	0.15	UNK	118	UNSAT	71	UNSAT	70	UNSAT	0.26	UNSAT	1462	6.33	UNSAT	0.60	UNSAT
N_{27}	0.10	UNK	0.16	UNK	34	UNSAT	25	UNSAT	8	UNSAT	0.04	UNSAT	555	3.38	UNSAT	0.31	UNSAT
N_{28}	0.11	UNK	0.17	UNK	549	UNSAT	121	UNSAT	33	UNSAT	0.06	UNSAT	1805	8.85	UNSAT	1.15	UNK
N_{29}	0.11	UNK	0.13	UNK	52	UNSAT	6	UNSAT	7	UNSAT	0.02	UNSAT	157	2.30	UNSAT	0.15	UNSAT
N_{31}	0.12	UNK	0.15	UNK	478	UNSAT	220	UNSAT	283	UNSAT	1.12	UNSAT	4281	13.82	UNSAT	0.52	UNSAT
N_{32}	0.13	UNK	0.16	UNK	107	UNSAT	237	UNSAT	102	UNSAT	0.44	UNSAT	8708	23.53	UNSAT	0.34	UNSAT
N_{33}	0.11	UNK	0.14	UNK	116	UNSAT	36	UNSAT	12	UNSAT	0.06	UNSAT	1201	5.13	UNSAT	0.26	UNSAT
N_{34}	0.11	UNK	0.14	UNK	75	UNSAT	32	UNSAT	13	UNSAT	0.15	UNSAT	1214	5.34	UNSAT	0.27	UNSAT
N_{35}	0.11	UNK	0.15	UNK	206	UNSAT	212	UNSAT	52	UNSAT	0.91	UNSAT	3630	15.78	UNSAT	0.80	UNSAT
N_{36}	0.14	UNK	0.17	UNK	141	UNSAT	50	UNSAT	47	UNSAT	1.42	UNSAT	1495	6.52	UNSAT	0.68	UNSAT
N_{37}	0.11	UNK	0.16	UNK	304	UNSAT	49	UNSAT	10	UNSAT	0.22	UNSAT	862	5.18	UNSAT	0.25	UNSAT
N_{38}	0.11	UNK	0.15	UNK	131	UNSAT	48	UNSAT	20	UNSAT	0.23	UNSAT	542	4.31	UNSAT	0.43	UNK
N_{39}	0.11	UNK	0.17	UNK	621	UNSAT	144	UNSAT	57	UNSAT	1.36	UNSAT	2684	11.72	UNSAT	0.56	UNSAT
N_{41}	0.11	UNSAT	0.13	UNK	49	UNSAT	30	UNSAT	40	UNSAT	2.73	UNSAT	848	3.74	UNSAT	0.19	UNSAT
N_{42}	0.11	UNK	0.16	UNK	244	UNSAT	77	UNSAT	42	UNSAT	1.96	UNSAT	1348	6.16	UNSAT	0.45	UNSAT
N_{43}	0.11	UNK	0.15	UNK	243	UNSAT	163	UNSAT	100	UNSAT	1.10	UNSAT	3725	10.62	UNSAT	0.50	UNSAT
N_{44}	0.11	UNK	0.15	UNK	206	UNSAT	52	UNSAT	31	UNSAT	1.31	UNSAT	1253	6.04	UNSAT	0.72	UNK
N_{45}	0.14	UNK	0.16	UNK	217	UNSAT	53	UNSAT	32	UNSAT	1.49	UNSAT	1255	5.30	UNSAT	0.44	UNSAT
N_{46}	0.11	UNK	0.15	UNK	177	UNSAT	18	UNSAT	8	UNSAT	0.02	UNSAT	2366	8.64	UNSAT	0.57	UNSAT
N_{47}	0.13	UNK	0.15	UNK	47	UNSAT	16	UNSAT	10	UNSAT	0.17	UNSAT	216	2.43	UNSAT	0.29	UNSAT
N_{48}	0.11	UNK	0.17	UNK	195	UNSAT	68	UNSAT	25	UNSAT	0.18	UNSAT	1591	6.87	UNSAT	0.47	UNSAT
N_{49}	0.11	UNK	0.17	UNK	422	UNSAT	51	UNSAT	19	UNSAT	0.08	UNSAT	2566	9.05	UNSAT	0.43	UNSAT
N_{51}	0.11	UNK	0.15	UNK	513	UNSAT	174	UNSAT	193	UNSAT	2.52	UNSAT	6932	17.71	UNSAT	0.40	UNSAT
N_{52}	0.11	UNK	0.15	UNK	210	UNSAT	48	UNSAT	31	UNSAT	0.86	UNSAT	4361	12.81	UNSAT	0.30	UNSAT
N_{53}	0.11	UNK	0.15	UNK	124	UNSAT	39	UNSAT	41	UNSAT	0.09	UNSAT	1662	6.54	UNSAT	0.43	UNSAT
N_{54}	0.11	UNK	0.15	UNK	144	UNSAT	43	UNSAT	12	UNSAT	0.12	UNSAT	1088	5.18	UNSAT	0.36	UNSAT
N_{55}	0.14	UNK	0.15	UNK	114	UNSAT	61	UNSAT	30	UNSAT	0.48	UNSAT	1549	6.27	UNSAT	0.30	UNSAT
N_{56}	0.10	UNK	0.15	UNK	160	UNSAT	55	UNSAT	45	UNSAT	0.33	UNSAT	677	4.56	UNSAT	0.33	UNSAT
N_{57}	0.11	UNK	0.18	UNK	38	UNSAT	8	UNSAT	8	UNSAT	0.03	UNSAT	157	2.12	UNSAT	0.24	UNSAT
N_{58}	0.11	UNK	0.16	UNK	111	UNSAT	48	UNSAT	22	UNSAT	0.06	UNSAT	502	4.20	UNSAT	0.59	UNSAT
N_{59}	0.11	UNK	0.15	UNK	116	UNSAT	57	UNSAT	14	UNSAT	0.09	UNSAT	992	5.20	UNSAT	0.35	UNSAT

Fig. 6: Detailed verification results for ϕ_4 of ACAS Xu networks.