NNV 2.0: The Neural Network Verification Tool

Diego Manzanas Lopez[1](✉), Sung Woo Choi[2], Hoang-Dung Tran[2], and Taylor T. Johnson[1]

[1] Vanderbilt University, Nashville, USA
`diego.manzanas.lopez@vanderbilt.edu`
[2] University of Nebraska, Lincoln, USA

Abstract. This manuscript presents the updated version of the Neural Network Verification (NNV) tool. NNV is a formal verification software tool for deep learning models and cyber-physical systems with neural network components. NNV was first introduced as a verification framework for feedforward and convolutional neural networks, as well as for neural network control systems. Since then, numerous works have made significant improvements in the verification of new deep learning models, as well as tackling some of the scalability issues that may arise when verifying complex models. In this new version of NNV, we introduce verification support for multiple deep learning models, including neural ordinary differential equations, semantic segmentation networks and recurrent neural networks, as well as a collection of reachability methods that aim to reduce the computation cost of reachability analysis of complex neural networks. We have also added direct support for standard input verification formats in the community such as VNNLIB (verification properties), and ONNX (neural networks) formats. We present a collection of experiments in which NNV verifies safety and robustness properties of feedforward, convolutional, semantic segmentation and recurrent neural networks, as well as neural ordinary differential equations and neural network control systems. Furthermore, we demonstrate the capabilities of NNV against a commercially available product in a collection of benchmarks from control systems, semantic segmentation, image classification, and time-series data.

Keywords: neural networks · cyber-physical systems · verification · tool

1 Introduction

Deep Learning (DL) models have achieved impressive performance on a wide range of tasks, including image classification [13,24,44], natural language processing [15,25], and robotics [47]. Recently, the usage of these models has expanded into many other areas, including safety-critical domains, such as autonomous vehicles [9,10,85]. However, deep learning models are opaque systems, and it has been demonstrated that their behavior can be unpredictable when small changes are applied to their inputs (i.e., adversarial attacks) [67].

Therefore, for safety-critical applications, it is often necessary to comprehend and analyze the behavior of the whole system, including reasoning about the safety guarantees of the system. To address this challenge, many researches have been developing techniques and tools to verify Deep Neural Networks (DNN) [4,6,22,39,40,48,55,64,65,77,83,84,86,87], as well as learning-enabled Cyber-Physical Systems (CPS) [3,8,12,23,26,34,35,38,50,51]. It is worth noting that despite the growing research interest, the verification of deep learning models still remains a challenging task, as the complexity and non-linearity of these models make them difficult to analyze. Moreover, some verification methods suffer from scalability issues, which limits the applicability of some existing techniques to large-scale and complex models. Another remaining challenge is the extension of existing or new methods for the verification of the extensive collection of layers and architectures existing in the DL area, such as Recurrent Neural Networks (RNN) [37], Semantic Segmentation Neural Networks (SSNN) [58] or Neural Ordinary Differential Equations (ODE) [11].

This work contributes to addressing the latter challenge by introducing version 2.0 of NNV[1] (*N*eural *N*etwork *V*erification)[2], which is a software tool that supports the verification of multiple DL models as well as learning-enabled CPS, also known as Neural Network Control Systems (NNCS) [80]. NNV is a software verification tool with the ability to compute exact and over-approximate reachable sets of feedforward neural networks (FFNN) [75,77,80], Convolutional Neural Networks (CNN) [78], and NNCS [73,80]. In NNV 2.0, we add verification support of 3 main DL models: 1) RNNs [74], 2) SSNNs (encoder-decoder architectures) [79], and 3) neural ODEs [52], as well as several other improvements introduced in Sect. 3, including support for The Verification of Neural Networks Library (VNNLIB) [29] and reachability methods for MaxUnpool and Leaky ReLU layers. Once the reachability computation is completed, NNV is capable of verifying a variety of specifications such as safety or robustness, very commonly used in learning-enabled CPS and classification domains, respectively [50,55]. We demonstrate NNV capabilities through a collection of safety and robustness verification properties, which involve the reachable set computation of feedforward, convolutional, semantic segmentation and recurrent neural networks, as well as neural ordinary differential equations and neural network control systems. Throughout these experiments, we showcase the range of the existing methods, executing up to 6 different star-based reachability methods that we compare against MATLAB's commercially available verification tool [69].

2 Related Work

The area of DNN verification has increasingly grown in recent years, leading to the development of standard input formats [29] as well as friendly competitions [50,55], that help compare and evaluate all the recent methods and tools proposed in the community [4,6,19,22,31,39–41,48,55,59,64,65,77,83,84,

[1] Code available at: https://github.com/verivital/nnv/releases/tag/cav2023.

[2] Archival version: https://doi.org/10.24433/CO.0803700.v1.

86,87]. However, the majority of these methods focus on regression and classification tasks performed by FFNN and CNN. In addition to FFNN and CNN verification, Tran et al. [79] introduced a collection of star-based reachability analysis that also verify SSNNs. Fischer et al. [21] proposed a probabilistic method for the robustness verification of SSNNs based on randomize smoothing [14]. Since then, some of the other recent tools, including Verinet [31], α,β-Crown [84,87], and MN-BaB [20] are also able to verify image segmentation properties as demonstrated in [55]. A less explored area is the verification of RNN. These models have unique "memory units" that enable them to store information for a period of time and learn complex patterns of time-series or sequential data. However, due to their memory units, verifying the robustness of RNNs is challenging. Recent notable state-of-the-art methodologies for verifying RNNs include unrolling the network into an FFNN and then verify it [2], invariant inference [36,62,90], and star-based reachability [74]. Similar to RNNs, neural ODEs are also deep learning models with "memory", which makes them suitable to learn time-series data, but are also applicable to other tasks such as continuous normalizing flows (CNF) and image classification [11,61]. However, existing work is limited to a stochastic reachability approach [27,28], reachability approaches using star and zonotope reachability methods for a general class of neural ODEs (GNODE) with continuous and discrete time layers [52], and GAINS [89], which leverages ODE-solver information to discretize the models using a computation graph that represent all possible trajectories from a given input to accelerate their bound propagation method. However, one of the main challenges is to find a framework that is able to verify several of these models successfully. For example, α,β-Crown was the top performer on last year's NN verification competition [55], able to verify FFNN, CNN and SSNNs, but it lacks support for neural ODEs or NNCS. There exist other tools that focus more on the verification of NNCS such as Verisig [34,35], Juliareach [63], ReachNN [17,33], Sherlock [16], RINO [26], VenMas [1], POLAR [32], and CORA [3,42]. However, their support is limited to NNCS with a linear, nonlinear ODE or hybrid automata as the plant model, and a FFNN as the controller.

Finally, for a more detailed comparison to state-of-the-art methods for the novel features of NNV 2.0, we refer to the comparison and discussion about neural ODEs in [52]. For SSNNs [79], there is a discussion on scalability and conservativeness of methods presented (approx and relax star) for the different layers that may be part of a SSNN [79]. For RNNs, the approach details and a state-of-the-art comparison can be found in [74]. We also refer the reader to two verification competitions, namely VNN-COMP [6,55] and AINNCS ARCH-COMP [38,50], for a comparison on state-of-the-art methods for neural network verification and neural network control system verification, respectively.

3 Overview and Features

NNV is an object-oriented toolbox developed in MATLAB [53] and built on top of several open-source software, including CORA [3] for reachability analysis of nonlinear ordinary differential equations (ODE) [73] and hybrid automata, MPT toolbox [45] for polytope-based operations [76], YALMIP [49] for some optimization problems in addition to MATLAB's Optimization Toolbox [53] and GLPK [56], and MatConvNet [82] for some convolution and pooling operations. NNV also makes use of MATLAB's deep learning toolbox to load the Open Neural Network Exchange (ONNX) format [57,68], and the Hybrid Systems Model Transformation and Translation tool (HyST) [5] for NNCS plant configuration.

NNV consists of two main modules: a *computation engine* and an *analyzer*, as illustrated in Fig. 1. The computation engine module consists of four components: 1) *NN constructor*, 2) *NNCS constructor*, 3) *reachability solvers*, and 4) *evaluator*. The NN constructor takes as an input a neural network, either as a DAGNetwork, dlnetwork, SeriesNetwork (MATLAB built-in formats) [69], or as an ONNX file [57], and generates a NN object suitable for verification. The NNCS constructor takes as inputs the NN object and an ODE or Hybrid Automata (HA) file describing the dynamics of a system, and then creates an NNCS object. Depending on the task to solve, either the NN (or NNCS) object is passed into the reachability solver to compute the reachable set of the system from a given set of initial conditions. Then, the computed set is sent to the analyzer module to verify/falsify a given property, and/or visualize the reachable sets. Given a specification, the verifier can formally reason whether the specification is met by computing the intersection of the define property and the reachable sets. If an exact (sound and complete) method is used, (e.g., exact-star), the analyzer can determine if the property is satisfied or unsatisfied. If an over-approximate (sound and incomplete) method is used, the verifier may also return "*uncertain*" (unknown), in addition to satisfied or unsatisfied.

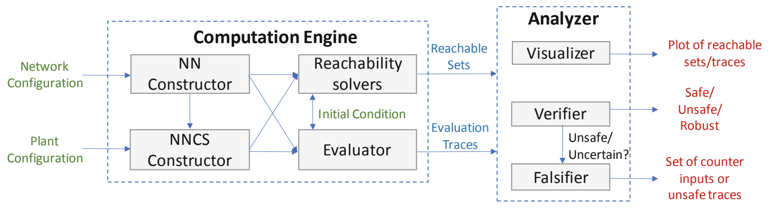


Fig. 1. An overview of NNV and its major modules and components.

3.1 NNV 2.0 vs NNV

Since the introduction of NNV [80], we have added to NNV support for the verification of a larger subset of deep learning models. We have added reachability methods to verify SSNNs [79], and a collection of relax-star reachability methods [79], reachability techniques for Neural ODEs [52] and RNNs [74]. In addition, there have been changes that include the creation of a common NN class that encapsulates previously supported neural network classes (FFNN and CNN) as well as Neural ODEs, SSNNs, and RNNs, which significantly reduces the software complexity and simplifies user experience. We have also added direct support for ONNX [57], as well as a parser for VNN-LIB [29], which describes properties to verify of any class of neural networks. We have also added flexibility to use one of the many solvers supported by YALMIP [49], GLPK [56] or linprog [70]. Table 1 shows a summary of the major features of NNV, highlighting the novel features.

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.

Feature	Supported (NNV 2.0 additions in blue)
Neural Network Type	FFNN, CNN, NeuralODE, SSNN, RNN
Layers	MaxPool, Conv, BN, AvgPool, FC, MaxUnpool, TC, DC, NODE
Activation functions	ReLU, Satlin, Sigmoid, Tanh, Leaky ReLU, Satlins
Plant dynamics (NNCS)	Linear ODE, Nonlinear ODE, HA, Continuous & Discrete Time
Set Representation	Polyhedron, Zonotope, Star, ImageStar
Star Reach methods	exact, approx, abs-dom, relax-range, relax-area, relax-random, relax-bound
Reachable set visualization	Yes, exact and over-approximation
Verification	Safety, Robustness, VNNLIB
Miscellaneous	Parallel computing, counterexample generation, ONNX*

*ONNX was partially supported for feedforward neural networks through NNVMT. Support has been extended to other NN types without the need for external libraries.

Semantic Segmentation [79]**.** Semantic segmentation consists on classifying image pixels into one or more classes which are semantically interpretable, like the different objects in an image. This task is common in areas like perception for autonomous vehicles, and medical imaging [71], which is typically accomplished by neural networks, referred to as semantic segmentation neural networks (SSNNs). These are characterized by two major portions, the encoder, or sequence of down-sampling layers to extract important features in the input, and the decoder, or sequence of up-sampling layers, to scale back the data information and classify each pixel into its corresponding class. Thus, the verification of these models is rather challenging, due to the complexity of the layers, and the output space dimensionality. We implement in NNV the collection of reachability methods introduced by Tran et al. [79], that are able to verify the robustness of a SSNNs. This means that we can formally guarantee the robustness value for each pixel, and determine the percentage of pixels that are correctly classified despite the adversarial attack. This was demonstrated using several architectures on two datasets: MNIST and M2NIST [46]. To achieve this, additional support for transposed and dilated convolutional layers was added [79].

Neural Ordinary Differential Equations [52]**.** Continuous deep learning models, referred to as Neural ODEs, have received a growing consideration over the last few years [11]. One of the main reasons for their popularity is due to their memory efficiency and their ability to learn from irregularly sampled data [61]. Similarly to SSNNs, despite their recent popularity, there is very limited work on the formal verification of these models [52]. For this reason, we implemented in NNV the first deterministic verification approach for a general class of neural ODEs (GNODE), which supports GNODEs to be constructed with multiple continuous layers (neural ODEs), linear or nonlinear, as well as any discrete-time layer already supported in NNV, such as ReLU, fully-connected or convolutional layers [52]. NNV demonstrates its capabilities in a series of time-series, control systems and image classification benchmarks, where it significantly outperforms any of the compared tools in the number of benchmarks and architectures supported [52].

Recurrent Neural Networks [74]**.** We implement star-based verification methods for RNNs introduced in [74]. These are able to verify RNNs without unrolling, reducing accumulated over-approximation error by optimized relaxation in the case of approximate reachability. The star set is an efficient technique in the computation of RNN reachable sets due to its advantages in computing affine mapping, the intersection of half-spaces, and Minkowski summation [74]. A new star set representing the reachable set of the current hidden state can be directly and efficiently constructed based on the reachable sets of the previous hidden state and the current input set. As proposed in verifying FFNNs [7,77,78], CNNs [72], and SSNNs [79], tight and efficient over-approximation reachability can be applied to the verification of ReLU RNNs. The triangular over-approximation of ReLU enables a tight over-approximation of the exact reachable set, preventing exponentially increasing the number of star sets during splitting. Estimation of the state bound required for over-approximation can compute state bounds without solving LPs. Furthermore, the relaxed approximate reachability estimates the triangle over-approximation areas to optimize the ranges of state by solving LP optimization. Consequently, the extended exact reachability method is 10× faster, and the over-approximation method is 100× to 5000× faster than existing state-of-the-art methods [74].

Zonotope Pre-filtering Star Set Reachability [78]**.** The star-based reachability methods are improved by using the zonotope pre-filtering approach [7,78]. This improvement consists on equipping the star set with an outer-zonotope, on the reachability analysis of a ReLU layer, to estimate quickly the lower and upper bounds of the star set at each specific neuron to establish if splitting may occur at this neuron without the need to solve any LP problems. The reduction of LP optimizations to solve is critical for the scalability of star-set reachability methods [77]. For the exact analysis, we are able to avoid the use of the zonotope pre-filtering, since we can efficiently construct the new output set with one star, if the zero point is not within the set range, or the union of 2 stars, if the zero point is contained [78]. In the over-approximation star, the range information is required to construct the output set at a specific neuron if and only if the range contains the zero point.

Relax-Star Reachability [79]**.** To tackle some of the scalability problems that may arise when computing the reachable set of complex neural networks such as SSNNs, a collection of four relaxed reachability methods were introduced [79]. The main goal of these methods is to reduce the number of Linear Programming (LP) problems to solve by quickly estimating the bounds or the reachable set, and only solving a fraction of the LP problems, while over-approximating the others. The LPs to solve are determined by the heuristics chosen, which can be random, area-based, bound-based, or range-based. The number of LPs is also determined by the user, who can choose from 0% to 100%. The closer to 100%, the larger number of LPs are skipped and over-approximated, thus the reachable set tends to be a larger over-approximation of the output, which significantly reduces the computation time [79].

Other Updates. In addition to the previous features described, there is a set of changes and additions included in the latest NNV version:

- *Activation Functions.* The star set method is extended to other classes of piecewise activation functions such as saturating linear layer (satlin), saturating linear symmetric layer (satlins), and leaky ReLU. The reachability analysis of each of these functions can be performed similarly to ReLU layers using the zonotope pre-filtering method to find where splits happen.

- *LP solver.* We generalize the use of LP solvers across all methods and optimizations. We allow the user to select the solver to use, which can choose between GLPK [56], linprog [70] (MATLAB's Optimization Toolbox) or any of the solvers supported by YALMIP [49]. We select *linprog* as the default solver, while keeping GLPK as a backup. However, if a different solver is selected that is supported by YALMIP, our implementation of the LP solver abstraction also supports this selection for any reachability method.

- *Standard Input Formats.* In the past few years, the verification community has been working to standardize formats across all tools to facilitate comparison among them. We have improved NNV by replacing the NNVMT tool [81] with a module to load ONNX [57] networks directly from MATLAB, as well as adding support for VNNLIB [29] files to define NN properties.

4 Evaluation

The evaluation is divided into 4 sections: 1) Comparison of FFNN and CNN to MATLAB's commercial toolbox [53,69], 2) Reachability analysis of Neural ODEs [52], 3) Robustness Verification of RNNs [74], and 4) Robustness Verification of SSNNs [79]. The results presented were all performed on a desktop with the following configuration: AMD Ryzen 9 5900X @3.7GHz 12-Core Processor, 64 GB Memory, and 64-bit Microsoft Windows 10 Pro.

4.1 Comparison to MATLAB's Deep Learning Verification Toolbox

In this comparison, we make use of a subset of the benchmarks and properties evaluated in last year's Verification of Neural Network (VNN) [55] competition, in which we demonstrate the capabilities of NNV with respect to the latest commercial product from MATLAB for the verification of neural networks [69].

We compared them on a subset of benchmarks from VNN-COMP'22 [55]: *ACAS Xu*, *Tllverify*, *Oval21* (CIFAR10 [43]), and *RL* benchmarks, which consists on verifying 90 out of 145 properties of the ACAS Xu, where we compare

Table 2. Verification of ACAS Xu properties 3 and 4.

		matlab	approx	relax 25%	relax 50%	relax 75%	relax 100%	exact (8)
prop 3 (45)	*SAT*	3	3	3	2	0	0	3
	UNSAT	10	29	8	2	1	0	42
	time (s)	0.1383	0.6368	0.6192	0.5714	0.3843	0.0276	521.9
prop 4 (45)	*SAT*	1	3	3	2	0	0	3
	UNSAT	2	32	6	1	1	0	42
	time (s)	0.1387	0.6492	0.6420	0.5682	0.3568	0.0261	89.85

Table 3. Verification results of the RL, tllverify and oval21 benchmarks. We selected 50 random specifications from the RL benchmarks, 10 from tllverify and all 30 from oval21. **-** means that the benchmark is not supported.

	RL (50)			Tllverify (10)			Oval21 (30)		
	SAT	*UNSAT*	*time (s)*	*SAT*	*UNSAT*	*time (s)*	*SAT*	*UNSAT*	*time (s)*
matlab	20	11	0.0504	0	0	0.1947	–	–	–
NNV	32	14	0.0822	0	0	13.57	0	11	136.5

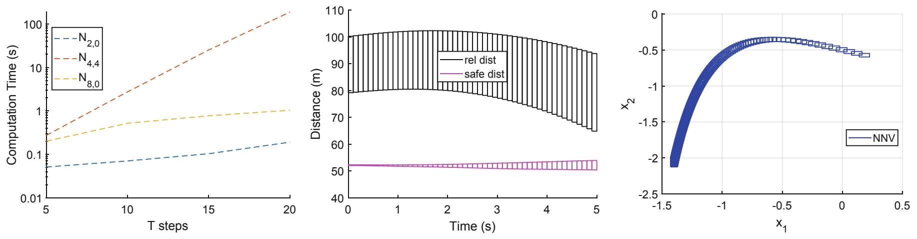
MATLAB's methods, approx-star, exact (parallel, 8 cores) and 4 relax-star methods. From the other 3 benchmarks, we select a total of 90 properties to verify, from which we limit the comparison to the approx-star and MATLAB's method. In this section, we demonstrate NNV is able to verify fully-connected layers, ReLU layers, flatten layers, and convolutional layers. The results of this comparison are described in Table 2. We can observe that MATLAB's computation time is faster than NNV star methods, except for the relax star with 100% relaxation. However, NNV's exact and approx methods significantly outperform MATLAB's framework by verifying 100% and 74% of the properties respectively, compared to 18% from MATLAB's. The remainder of the comparison is described in Table 3, which shows a similar trend: MATLAB's computation is faster, while NNV is able to verify a larger fraction of the properties.

4.2 Neural Ordinary Differential Equations

We exhibit the reachability analysis of GNODEs with three tasks: dynamical system modeling of a Fixed Point Attractor (FPA) [52,54], image classification of MNIST [46], and an adaptive cruise control (ACC) system [73].

Dynamical Systems. For the FPA, we compute the reachable set for a time horizon of 10 s, given a perturbation of $\pm$ 0.01 on all 5 input dimensions. The results of this example are illustrated in Fig. 2c, with a computation time of 3.01 s. The FPA model consists of one nonlinear neural ODE, no discrete-time layers are part of this model [52].

Classification. For the MNIST benchmark, we evaluate the robustness of two GNODEs with convolutional, fully-connected, ReLU and neural ODE layers, corresponding to CNODE$_S$ and CNODE$_M$ models introduced in [52]. We verify the robustness of 5 random images under an L$_\infty$ attack with a perturbation



(a) RNN Computation Times (b) Neural ODE, NNCS Nonlinear ACC (c) Neural ODE FPA

Fig. 2. Verification of RNN and neural ODE results. Figure 2a shows the verification time of the 3 RNNs evaluated. Figure 2b depicts the safety verification of the ACC, and Fig. 2c shows the reachability results of the FPA benchmark.

value of ± 0.5 on all the pixels. We are able to prove the robustness of both models on 100% of images, with an average computation time of 16.3 s for the CNODE_S, and 119.9 s for the CNODE_M.

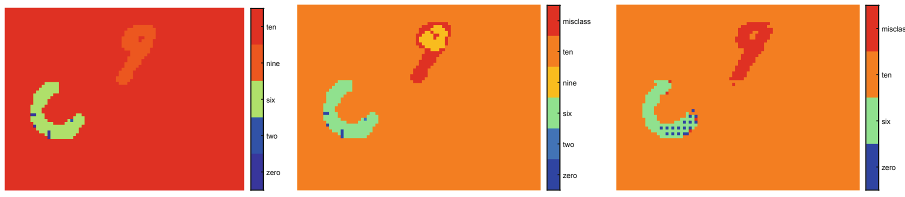
Control Systems. We verify an NNCS of an adaptive cruise control (ACC) system, where the controller is a FFNN with 5 ReLU layers with 20 neurons each, and one output linear layer, and the plant is a nonlinear neural ODE [52]. The verification results are illustrated in Fig. 2b, showing the current distance between the ego and lead cars and the safety distance allowed. We can observe that there is no intersection between the two, guaranteeing its safety.

4.3 Recurrent Neural Networks

For the RNN evaluation, we evaluate of three RNNs trained on the speaker recognition VCTK dataset [88]. Each network has an input layer of 40 neurons, two hidden layers with 2,4, or 8 memory units, followed by 5 ReLU layers with 32 neurons, and an output layer of 20 neurons. For each of the networks, we use the same 5 input points (40-dimensional time-independent vectors) for comparison. The robustness verification consists on proving that the output label after $\text{T} \in \{5, 10, 15, 20\}$ steps in the sequence is still the same, given an adversarial attack perturbation of $\epsilon = \pm\, 0.01$. We compute the reachable sets of all reachability instances using the approx-star method, which was able to prove the robustness of *19* out of 20 on $\text{N}_{2,0}$, and $\text{N}_{4,4}$ networks, and 18 for the $\text{N}_{8,0}$ network. We show the average reachability time per T value in Fig. 2a.

4.4 Semantic Segmentation

We demonstrate the robustness verification of two SSNNs, one with dilated convolutional layers and the other one with transposed convolutional layers, in addition to average pooling, convolutional and ReLU layers, which correspond to N_4 and N_5 introduced in Table 1 by Tran et al. [79]. We evaluate them on one random image of M2NIST [18] by attacking each image using an UBAA brightening attack [79]. One of the main differences of this evaluation with respect to the robustness analysis of other classification is the evaluation metrics used. For these networks, we evaluate the average robustness values (percentage of pixels correctly classified), sensitivity (number of not robust pixels over number of attacked pixels), and IoU (intersection over union) of the SSNNs. The computation time for the dilated example, shown in Fig. 3, is 54.52 s, with a robustness value of 97.2%, a sensitivity of 3.04, and a IoU of 57.8%. For the equivalent example with the transposed network, the robustness value is 98.14%, sensitivity of 2, IoU of 72.8%, and a computation time of 7.15 s.



(a) Target Image (b) Transposed SSNN (c) Dilated SSNN

Fig. 3. Robustness verification of the dilated and transposed SSNN under a UBAA brightening attack to 150 random pixels in the input image.

5 Conclusions

We presented version 2.0 of NNV, the updated version of the Neural Network Verification (NNV) tool [80], a software tool for the verification of deep learning models and learning-enabled CPS. To the best of our knowledge, NNV is the most comprehensive verification tool in terms of the number of tasks and neural networks architectures supported, including the verification of feedforward, convolutional, semantic segmentation, and recurrent neural networks, neural ODEs and NNCS. With the recent additions to NNV, we have demonstrated that NNV can be a one-stop verification tool for users with a diverse problem set, where verification of multiple neural network types is needed. In addition, NNV supports zonotope, polyhedron based methods, and up to 6 different star-based reachability methods to handle verification tradeoffs for the verification problem of neural networks, ranging from the exact-star, which is sound and complete, but computationally expensive, to the relax-star methods, which are significantly faster but more conservative. We have also shown that NNV outperforms a commercially available product from MATLAB, which computes the reachable sets of feedforward neural networks using the zonotope reachability method presented in [66]. In the future, we plan to ensure support for other deep learning models such as ResNets [30] and UNets [60].

Acknowledgments. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) through grant numbers 1910017, 2028001, 2220418, 2220426 and 2220401, and the NSF Nebraska EPSCoR under grant OIA-2044049, the Defense Advanced Research Projects Agency (DARPA) under contract number FA8750-18-C-0089 and FA8750-23-C-0518, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-22-1-0019 and FA9550-23-1-0135. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of AFOSR, DARPA, or NSF.

References

1. Akintunde, M.E., Botoeva, E., Kouvaros, P., Lomuscio, A.: Formal verification of neural agents in non-deterministic environments. In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), IFAAMAS 2020, . ACM, Auckland (2020)
2. Akintunde, M.E., Kevorchian, A., Lomuscio, A., Pirovano, E.: Verification of rnn-based neural agent-environment systems. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6006–6013 (2019)
3. Althoff, M.: An introduction to cora 2015. In: Proceedings of the Workshop on Applied Verification for Continuous and Hybrid Systems (2015)
4. Bak, S.: nnenum: verification of ReLU neural networks with optimized abstraction refinement. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NFM 2021. LNCS, vol. 12673, pp. 19–36. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76384-8_2
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., Liu, C., Johnson, T.T.: The second international verification of neural networks competition (VNN-COMP 2021): Summary and results. CoRR abs/2109.00498 (2021)
7. Bak, S., Tran, H.-D., Hobbs, K., Johnson, T.T.: Improved geometric path enumeration for verifying ReLU neural networks. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12224, pp. 66–96. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_4
8. Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., Schilling, C.: Juliareach: a toolbox for set-based reachability. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pp. 39–44 (2019)
9. Bojarski, M., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
10. 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)
11. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. Adv. Neural Inf. Process. Syst. (2018)
12. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_18

13. Cireşan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. arXiv preprint arXiv:1202.2745 (2012)
14. Cohen, J., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 1310–1320. PMLR (2019)
15. 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)
16. Dutta, S., Jha, S., Sankaranarayanan, S., Tiwari, A.: Output range analysis for deep feedforward neural networks, pp. 121–138 (2018)
17. Fan, J., Huang, C., Chen, X., Li, W., Zhu, Q.: ReachNN*: a tool for reachability analysis of neural-network controlled systems. In: Hung, D.V., Sokolsky, O. (eds.) ATVA 2020. LNCS, vol. 12302, pp. 537–542. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_30
18. (farhanhubble), F.A.: M2NIST, MNIST of semantic segmentation. https://www.kaggle.com/datasets/farhanhubble/multimnistm2nist
19. Ferlez, J., Khedr, H., Shoukry, Y.: Fast BATLLNN: fast box analysis of two-level lattice neural networks. In: Bartocci, E., Putot, S. (eds.) HSCC '22: 25th ACM International Conference on Hybrid Systems: Computation and Control, Milan, Italy, 4–6 May 2022. pp. 23:1–23:11. ACM (2022)
20. Ferrari, C., Müller, M.N., Jovanovic, N., Vechev, M.T.: Complete verification via multi-neuron relaxation guided branch-and-bound. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, 25–29 April 2022. OpenReview.net (2022)
21. Fischer, M., Baader, M., Vechev, M.: Scalable certified segmentation via randomized smoothing. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 3340–3351. PMLR (2021)
22. Fischer, M., Sprecher, C., Dimitrov, D.I., Singh, G., Vechev, M.: Shared certificates for neural network verification. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification, pp. 127–148. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13185-1_7
23. Frehse, G., et al.: SpaceEx: scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_30
24. 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)
25. Goldberg, Y.: A primer on neural network models for natural language processing. J. Artif. Intell. Res. **57**, 345–420 (2016)
26. Goubault, E., Putot, S.: Rino: Robust inner and outer approximated reachability of neural networks controlled systems. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification, pp. 511–523. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13185-1_25
27. Gruenbacher, S., Hasani, R.M., Lechner, M., Cyranka, J., Smolka, S.A., Grosu, R.: On the verification of neural odes with stochastic guarantees. In: AAAI (2021)
28. Gruenbacher, S., et al.: Gotube: scalable stochastic verification of continuous-depth models (2021)
29. Guidotti, D., Demarchi, S., Tacchella, A., Pulina, L.: The Verification of Neural Networks Library (VNN-LIB) (2022). https://www.vnnlib.org

30. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
31. Henriksen, P., Hammernik, K., Rueckert, D., Lomuscio, A.: Bias field robustness verification of large neural image classifiers. In: Proceedings of the 32nd British Machine Vision Conference (BMVC21). BMVA Press (2021)
32. Huang, C., Fan, J., Chen, X., Li, W., Zhu, Q.: Polar: a polynomial arithmetic framework for verifying neural-network controlled systems (2021). https://doi.org/10.48550/ARXIV.2106.13867
33. Huang, C., Fan, J., Li, W., Chen, X., Zhu, Q.: Reachnn: reachability analysis of neural-network controlled systems. arXiv preprint arXiv:1906.10654 (2019)
34. Ivanov, R., Carpenter, T., Weimer, J., Alur, R., Pappas, G., Lee, I.: Verisig 2.0: verification of neural network controllers using taylor model preconditioning. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 249–262. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_11
35. 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)
36. Jacoby, Y., Barrett, C., Katz, G.: Verifying recurrent neural networks using invariant inference. In: Hung, D.V., Sokolsky, O. (eds.) ATVA 2020. LNCS, vol. 12302, pp. 57–74. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59152-6_3
37. Jain, L.C., Medsker, L.R.: Recurrent neural networks: design and applications (1999)
38. Johnson, T.T., et al.: Arch-comp21 category report: artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In: Frehse, G., Althoff, M. (eds.) 8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21). EPiC Series in Computing, vol. 80, pp. 90–119. EasyChair (2021). https://doi.org/10.29007/kfk9
39. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: an efficient SMT solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 97–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_5
40. Katz, G., et al.: The marabou framework for verification and analysis of deep neural networks. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 443–452. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_26
41. Khedr, H., Ferlez, J., Shoukry, Y.: PEREGRiNN: penalized-relaxation greedy neural network verifier. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 287–300. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_13
42. Kochdumper, N., Schilling, C., Althoff, M., Bak, S.: Open- and closed-loop neural network verification using polynomial zonotopes (2022)
43. Krizhevsky, A.: Learning multiple layers of features from tiny images, pp. 32–33 (2009)
44. 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)
45. Kvasnica, M., Grieder, P., Baotić, M., Morari, M.: Multi-parametric toolbox (MPT). In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 448–462. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24743-2_30
46. LeCun, Y.: The mnist database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist/

47. Lenz, I.: Deep learning for robotics. Ph.D. thesis, Cornell University (2016)
48. Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M.J.: Algorithms for verifying deep neural networks. Found. Trends Optim. **4**(3–4), 244–404 (2021). https://doi.org/10.1561/2400000035
49. Löfberg, J.: Yalmip : A toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference, Taipei, Taiwan (2004). http://users.isy.liu.se/johanl/yalmip
50. Lopez, D.M., et al.: Arch-comp22 category report: artificial intelligence and neural network control systems (ainncs) for continuous and hybrid systems plants. In: Frehse, G., Althoff, M., Schoitsch, E., Guiochet, J. (eds.) Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22). EPiC Series in Computing, vol. 90, pp. 142–184. EasyChair (2022)
51. Lopez, D.M., Johnson, T.T., Bak, S., Tran, H.D., Hobbs, K.: Evaluation of neural network verification methods for air to air collision avoidance. AIAA J. Air Transp. (JAT) (2022)
52. Manzanas Lopez, D., Musau, P., Hamilton, N., Johnson, T.: Reachability analysis of a general class of neural ordinary differential equation. In: Proceedings of the 20th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2022), Co-Located with CONCUR, FMICS, and QEST as part of CONFEST 2022, Warsaw, Poland (2022)
53. MATLAB: Update 3, (R2022b). The MathWorks Inc., Natick, Massachusetts (2022)
54. Musau, P., Johnson, T.T.: Continuous-time recurrent neural networks (ctrnns) (benchmark proposal). In: 5th Applied Verification for Continuous and Hybrid Systems Workshop (ARCH), Oxford, UK (2018). https://doi.org/10.29007/6czp
55. Müller, M.N., Brix, C., Bak, S., Liu, C., Johnson, T.T.: The third international verification of neural networks competition (vnn-comp 2022): Summary and results (2022)
56. Oki, E.: Glpk (gnu linear programming kit) (2012)
57. (ONNX), O.N.N.E.: https://github.com/onnx/
58. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. CoRR abs/1511.08458 (2015). http://dblp.uni-trier.de/db/journals/corr/corr1511.html#OSheaN15
59. Prabhakar, P., Rahimi Afzal, Z.: Abstraction based output range analysis for neural networks. Adv. Neural Inf. Process. Syst. **32** (2019)
60. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
61. Rubanova, Y., Chen, R.T.Q., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32. Curran Associates, Inc. (2019)
62. Ryou, W., Chen, J., Balunovic, M., Singh, G., Dan, A., Vechev, M.: Scalable polyhedral verification of recurrent neural networks. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 225–248. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_10
63. Schilling, C., Forets, M., Guadalupe, S.: Verification of neural-network control systems by integrating Taylor models and zonotopes. In: AAAI, pp. 8169–8177. AAAI Press (2022). https://doi.org/10.1609/aaai.v36i7.20790

64. Shriver, D., Elbaum, S., Dwyer, M.B.: DNNV: a framework for deep neural network verification. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 137–150. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_6
65. 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)
66. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang. **3**(POPL), 41 (2019)
67. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
68. The MathWorks, I.: Deep Learning Toolbox Converter for ONNX Model Format. Natick, Massachusetts, United State (2022). https://www.mathworks.com/matlabcentral/fileexchange/67296-deep-learning-toolbox-converter-for-onnx-model-format
69. The MathWorks, I.: Deep Learning Toolbox Verification Library. Natick, Massachusetts, United State (2022). https://www.mathworks.com/matlabcentral/fileexchange/118735-deep-learning-toolbox-verification-library
70. The MathWorks, I.: Optimization Toolbox. Natick, Massachusetts, United State (2022). https://www.mathworks.com/products/optimization.html
71. Thoma, M.: A survey of semantic segmentation (2016)
72. Tran, H.-D., Bak, S., Xiang, W., Johnson, T.T.: Verification of deep convolutional neural networks using imagestars. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12224, pp. 18–42. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_2
73. 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 2019). ACM (2019)
74. 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) (2023)
75. Tran, H.D., et al.: Parallelizable reachability analysis algorithms for feed-forward neural networks. In: Proceedings of the 7th International Workshop on Formal Methods in Software Engineering (FormaliSE 2019), pp. 31–40. IEEE Press, Piscataway (2019). https://doi.org/10.1109/FormaliSE.2019.00012
76. Tran, H.D., et al.: Parallelizable reachability analysis algorithms for feed-forward neural networks. In: 7th International Conference on Formal Methods in Software Engineering (FormaliSE2019), Montreal, Canada (2019)
77. Tran, H.-D., et al.: Star-based reachability analysis of deep neural networks. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM 2019. LNCS, vol. 11800, pp. 670–686. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30942-8_39
78. Tran, H.D., et al.: Verification of piecewise deep neural networks: a star set approach with zonotope pre-filter. Formal Asp. Comput. **33**(4), 519–545 (2021)
79. Tran, H.-D., et al.: Robustness verification of semantic segmentation neural networks using relaxed reachability. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 263–286. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_12
80. Tran, H.D., et al.: 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) (2020)

81. Transformation, N.N.V.M.: https://github.com/verivital/nnvmt
82. Vedaldi, A., Lenc, K.: Matconvnet: convolutional neural networks for matlab. In: Proceedings of the 23rd ACM International Conference on Multimedia, pp. 689–692. ACM (2015)
83. 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 2018), pp. 1599–1614 (2018)
84. Wang, S., et al.: Beta-CROWN: efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. Adv. Neural Inf. Process. Syst. **34** (2021)
85. 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)
86. Xiang, W., Tran, H.D., Johnson, T.T.: Output reachable set estimation and verification for multilayer neural networks. IEEE Trans. Neural Netw. Learn. Syst. **29**(11), 5777–5783 (2018)
87. Xu, K., et al.: Fast and Complete: enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=nVZtXBI6LNn
88. Yamagishi, J., Veaux, C., MacDonald, K.: Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92). In: University of Edinburgh. The Centre for Speech Technology Research (CSTR) (2019). https://doi.org/10.7488/ds/2645
89. Zeqiri, M., Mueller, M.N., Fischer, M., Vechev, M.: Efficient robustness verification of neural ordinary differential equations. In: The Symbiosis of Deep Learning and Differential Equations II (2022)
90. Zhang, H., Shinn, M., Gupta, A., Gurfinkel, A., Le, N., Narodytska, N.: Verification of recurrent neural networks for cognitive tasks via reachability analysis. In: ECAI 2020, pp. 1690–1697. IOS Press (2020)