2020 IEEE Workshop on Design Automation for CPS and IoT (DESTION)

Demo: The Neural Network Verification (NNV) Tool

Hoang-Dung Tran *Vanderbilt University*
Diego Manzanas Lopez *Vanderbilt University*
Xiaodong Yang *Vanderbilt University*
Patrick Musau *Vanderbilt University*
Luan Viet Nguyen *University of Notre Dame*
Weiming Xiang *Augusta University*
Stanley Bak *Safe Sky Analytics*
Taylor T. Johnson *Vanderbilt University*

***Abstract*—NNV (Neural Network Verification) is a framework for the verification of deep neural networks (DNNs) and learning-enabled cyber-physical systems (CPS) inspired by a collection of reachability algorithms that make use of a variety of set representations such as the star set. NNV supports exact and over-approximate reachability algorithms used to verify the safety and robustness of feed-forward neural networks (FFNNs). These two analysis schemes are also used for learning enabled CPS, i.e., closed-loop systems, and particularly in neural network control systems with linear models and FFNN controllers with piecewise-linear activation functions. Additionally, NNV supports over-approximate analysis for nonlinear plant models by combining the star set analysis used for FFNNs with the zonotope-based analysis for nonlinear plant dynamics provided by CORA. This demo paper demonstrates NNV's capabilities by considering a case study of the verification of a learning-enabled adaptive cruise control system.**

I. Introduction

Artificial intelligence has been a disruptive innovation in the way we develop complex embedded systems in the last few years. In this area, deep neural networks (DNNs) have become prevalent due to their promising performance on solving sophisticated problems in domains such as image classification, natural language processing (NLP), and autonomous vehicles.

In this paper, we present an overview demonstration of *NNV* (*N*eural *N*etwork *V*erification), which is a tool for the set-based verification of Feedforward Neural Networks (FNNs) and learning-enabled CPS, i.e., close-loop systems (NNCS). NNV offers a selection of reachability algorithms that are able to calculate the reachable sets of DNNs and NNCSs, either exact or over-approximate results, that can be represented as polyhedra [1], star-sets [2], zonotopes, and the relax polyhedron [3]. *NNV* declares a system to be safe if the reachable sets do not violate safety properties, expressed as the intersection of the reachable sets and the unsafe set is the null set (empty). We illustrate the capabilities of *NNV* with a NNCS example, an adaptive cruise control system (NNCS) with a FFNN controller.

A. Overview and Features

NNV[1] is written in MATLAB and builds on pre-existing toolboxes such as MPT [4] and CORA [5]. It also incorporates the neural network verification transformation tool[2] (nnvmt) which supports transforming DNNs models from frameworks such as Tensorflow. For plant models, we include HyST [6] , the hybrid systems model transformation and translation tool. We provide a summary of the key features *NNV* supports in Table I. Most recently, support for convolutional neural networks (CNNs) has been added [7].

Features	**NNV Suport**
FFNN and NNCS	E & O
Linear dynamics (NNCS)	E & O
Nonlinear dynamics (NNCS)	O
Discrete-time plant (NNCS)	E & O
Continuous-time plant (NNCS)	O
ReLU and other piecewise-linear activation functions	E & O
Sigmoid and tanh activation functions	O
Star representation	E & O
Polyhedron representation	E
Zonotope and polytope representation	O
Set visualization	E & O
Parallel computing	E
Safety verification and falsification	E & O
Robustness verification	E & O
Counterexample generation	E & O

Table I
Overview of the features in *NNV*, where *E* designates exact analysis and *O* designates over-approximate analysis.

II. Reachability Analysis

NNV implements several reachability algorithms for DNNs, which are computed layer-by-layer. This means that given an input set, we compute the output reachable set of the first layer. This output set is then the input set of the following layer and repeats until the last layer. For NNCS, we utilize this procedure to compute the reachable set of the FFNN controller, which is the plant's input set. Based on the current state set and the input set of the plant, we compute the reachable set of the plant, which is the input set of the controller in the next time step. This process occurs repetitively.

Polyhedron. We are able to compute the exact reachable set of a FFNN with ReLU activation functions. However, computing the exact reachable set using polyhedra is very expensive as it can split into multiple polyhedra at each operation. Moreover, they are not efficient in affine mapping.

[1]https://github.com/verivital/nnv

[2]https://github.com/verivital/nnvmt

978-1-7281-9994-8/20/$31.00 ©2020 IEEE
DOI 10.1109/DESTION50928.2020.00010

Thus, since the output set consists of a union of polyhedra, when they are propagated through a NNCS we obtain a very conservative over-approximation [8].

Star set. The star set representation is very efficient and suitable for large linear systems and reachability analysis of DNNs [2]. We can use star sets to compute exact and over-approximate analysis. Although the exact approach is very similar to the polyhedron case, it is much more efficient to compute affine mappings and intersections of stars.

Zonotope. This method is able to compute an over-approximation of the exact reachable set, similar to the over-approximate method using star sets, and it is also a fast approach. Although it produces a very conservative approximation [2], it is suitable for robustness verification when provided a very small input set.

Relax polyhedron. We implement the relax polyhedron abstract domain [3] for FFNN by specifying it as a star set and estimate output ranges using a 'back-tracking' method.

III. Demonstration

We present one experiment, an adaptive cruise control (ACC) system, which is an example of NNCS that shows the capabilities of *NNV* for both FFNNs and closed-loop systems with neural network controllers. In this experiment, we evaluate multiple ACC systems under different conditions. We consider a four controllers, where the smallest has 3 hidden layers and the largest has 10 hidden layers. All controllers have a similar layer structure with 20 neurons with ReLU activation functions and a linear output layer. The NNs have 5 inputs and one output to control the acceleration of the ego car. These controllers are combined with a nonlinear continuous-time plant and a discrete-time linear plant to form multiple ACC systems.

In Figure 1, we show an example of the ACC [8] in which we combine the smallest controller (61 neurons) and the nonlinear plant. Given the initial distance and velocities, it may not satisfy the safety condition as the relative distance intersects with the unsafe set, but this is not conclusively decided due to the over-approximate analysis. We present more details in the demonstration presentation or an upcoming tool paper [9], or one can refer to the *NNV* GitHub repository for the artifacts.

IV. Discussion

In this paper, we have demonstrated *NNV* by discussing its main features, such as the exact and over-approximate methods using different types of set representation. We have evaluated *NNV* functionalities via the safety verification of an ACC system with a continuous-time plant with nonlinear dynamics and a FFNN controller. We have shown one example, but we will explore how changing the initial conditions of the plant, discrete-time vs. continuous-time plants, linear vs. nonlinear dynamics and more, can affect the resulting reachable sets as well as the computational time in the demonstration presentation.

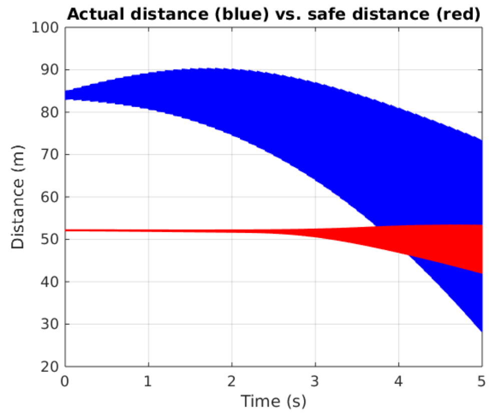

Figure 1. The ACC may not satisfy the safety property because the relative distance intersects with the unsafe region, but this computation is done with over-approximate analysis, so we cannot conclude it is necessarily unsafe.

Acknowledgment

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers SHF 1910017 and FMitF 1918450, the Air Force Office of Scientific Research (AFOSR) through contract number FA9550-18-1-0122, and the Defense Advanced Research Projects Agency (DARPA) through contract number FA8750-18-C-0089. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFOSR, DARPA, or NSF.

References

[1] H.-D. Tran, P. Musau, D. M. Lopez, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Parallelizable reachability analysis algorithms for feed-forward neural networks," in *Proceedings of the 7th International Workshop on Formal Methods in Software Engineering (FormaliSE'19)*, ser. FormaliSE '19. Piscataway, NJ, USA: IEEE Press, May 2019, pp. 31–40.

[2] H.-D. Tran, P. Musau, D. Lopez, X. Yang, L. Nguyen, W. Xiang, and T. Johnson, "Star-based reachability analysis for deep neural networks," in *23rd International Symposium on Formal Methods (FM'19)*. Springer International Publishing, October 2019.

[3] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, Jan. 2019.

[4] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *Proc. of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510.

[5] M. Althoff, D. Grebenyuk, and N. Kochdumper, "Implementation of taylor models in cora 2018," in *Proc. of the 5th International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2018, pp. 145–173.

[6] S. Bak, S. Bogomolov, and T. T. Johnson, "Hyst: A source transformation and translation tool for hybrid automaton models," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 128–133.

[7] H.-D. Tran, S. Bak, W. Xiang, and T. T. Johnson, "Verification of deep convolutional neural networks using imagestars," in *32nd International Conference on Computer-Aided Verification (CAV)*, 2020.

[8] H.-D. Tran, F. Cei, D. M. Lopez, T. T. Johnson, and X. Koutsoukos, "Safety verification of cyber-physical systems with reinforcement learning control," in *ACM SIGBED International Conference on Embedded Software (EMSOFT'19)*. ACM, October 2019.

[9] H. D. Tran, X. Yang, D. M. Lopez, P. Musau, L. V. Nguyen, W. Xiang, S. Bak, and T. T. Johnson, "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.