Bounded Model Checking of Deep Neural Network Controllers

MATTHEW SOTOUDEH, University of California, Davis, USA

We consider the problem of verifying the safety of trained, piecewise-linear neural network controllers. Prior work, including Z3 [9], PPL [4], and ReluPlex [19], proves to be poorly suited for the task. We present an alternative approach based on a *symbolic representation* of the network to efficiently compute strongest post-conditions over infinite input regions, extending our earlier work [24] to higher dimensions. We demonstrate significant performance improvements on BMC of three networks from prior work.

Author Information: Undergraduate category. Advised by Professor Aditya V. Thakur.

1 MOTIVATION, BACKGROUND, AND OVERVIEW

Deep Neural Network Controllers The past decade has seen the rise of deep neural networks (DNNs) [14] to solve a variety of problems, including image recognition [21, 25], natural-language processing [10], and autonomous vehicle control [18]. One major use of DNNs is as controllers for robotic devices such as satellites. Verifying that these networks are safe and correct is a challenging and important area of research as these networks become used in safety-critical applications.

In this work, we focus on the class of *piecewise-linear (PWL) neural networks*, which can be decomposed into a set of affine functions. In fact, the majority of modern neural networks are piecewise-linear [1, 15, 20], and those that are not can usually be modified to use piecewise-linear approximations with little-to-no effect on performance in practice.

Symbolic Representations for PWL Neural Networks In Sotoudeh and Thakur [24] we introduced a *one-dimensional symbolic representation* for piecewise-linear neural networks. This symbolic representation can *precisely characterize* the network's behavior on a one-dimensional subset of its input domain by partitioning the subset into regions where the network's behavior is affine. With this representation, any analysis on the one-dimensional subset can be translated into the union of such analyses on a finite set of affine functions, one for each of the partitions. In that work, we showed that efficient algorithms exist for the one-dimensional case and demonstrated that there are a number of worthwhile analyses that can be performed on such one-dimensional subsets.

In this work, we have extended the one-dimensional representation to two dimensions. We have designed an efficient algorithm to compute this representation based on a counter-clockwise vertex representation of high-dimensional polytopes. We provide a high-performance implementation [2] of our algorithm in C++ parallelized with Intel TBB, accessible via a user-friendly Python frontend.

Bounded Model Checking One important approach to verifying the safety of a controller is *bounded model checking*. With this approach, given a controller represented as a transition system, a set of *initial* states, a set of *unsafe* states, and a bound K, we want to determine whether an unsafe state is reachable from an initial state in at most K steps.

Bounded Model Checking of DNNs with Prior Work As long as the underlying theory can encode the DNN, the state transition, and the sets of initial and unsafe states, in principle any SMT solver can perform BMC on a DNN by repeatedly querying the SMT solver for each timestep in question and requesting an assignment to the inputs which is within the initial set and results in an output after some specific number of steps in the unsafe set. We performed initial tests using the state-of-the-art SMT solver Z3 [9] as well as the polyhedra-specific library PPL [4], both of which had untenable performance and accuracy issues on any non-toy network (eg. more than 2-unit).

Author's address: Matthew Sotoudeh, Computer Science, University of California, Davis, USA, masotoudeh@ucdavis.edu.

Table 1. Maximum number of steps verified by our algorithm compared to ReluPlex before timeout. Timeout was set to 1 hour. "*" indicates a counter example was found after that many steps.

Model	Our Algorithm Steps	ReluPlex Steps
Pendulum	51*	2
Quadcopter	25	6
Satelite	13	4

This is to be expected, as DNNs present special challenges because they are both highly non-linear (eg. the commonly-used ReLU node introduces branching that grows exponentially with the size of each layer) and high-dimensional (eg. common neural networks have states in their execution with many thousands of components). Prior work such as ReluPlex [19] have recognized this issue and built special-purpose SMT solvers optimized for common DNN architectures. We found that ReluPlex was performant enough to verify a small number of steps, but that it quickly timed out as we verified later steps because the transition function increases in size for each subsequent step.

Our Approach In contrast to the SMT-solver based approaches, we build our algorithm around a *symbolic representation* which allows us to quickly and efficiently compute the reach set of the system given any set of initial states. In this way, we can iteratively compute the reach set at each step, re-using the results computed at the previous iteration instead of "starting from scratch" as in the ReluPlex approach.

2 TWO-DIMENSIONAL SYMBOLIC REPRESENTATION ALGORITHM

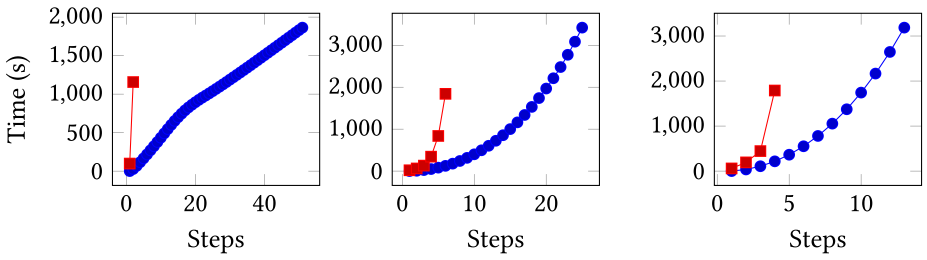
Our BMC algorithm relies on a symbolic representation of the neural network. The fundamental algorithm used to compute this symbolic representation takes as input a convex polytope along with a PWL function described by its partitions (i.e., within each partition the function is linear), then returns a partitioning of the polytope such that the function is linear within each partition.

The key idea of the algorithm is to start at some vertex of the polytope and "trace" the boundary of the polytope until a boundary of one of the function partitions is encountered. Once such a boundary is encountered we partition the polytope in two such that each partition lies on exactly one side of the function boundary. We recursively continue this process on each of the new partitions. The algorithm terminates when each partition lies entirely on one side of each function boundary, i.e. within a single linear partition of the function. In practice, we further optimize the performance of this algorithm by allowing function boundaries to be bounded (instead of just infinite planes).

In the best-case scenario, the algorithm simply iterates over all of the n input partitions, checks their v vertices, and appends to the resulting set (for a best-case complexity of $O(nv)$). In the worst case, it splits each polytope in the queue on each face, resulting in exponential worst-case complexity. In practice, however, we find that this algorithm is very efficient and can be applied to real-world networks effectively (see Section 3).

3 EXPERIMENTAL EVALUATIONS

Tests performed on a dedicated Amazon EC2 c5.metal instance, using Benchexec [7] to limit the number of CPU cores to 16 and RAM to 16GB. Our code is available at SyR [2]. We took three neural network controllers from Zhu et al. [31] with two-dimensional states, modified them to use only PWL activations (which did not impact test accuracy of the network), and performed BMC on the models using the same initial and unsafe sets from that work. Figure 1 shows the time taken to perform the analysis for each network up to a given timestep of the model, while Table 1 summarizes these results.



(a) Pendulum (b) Quadcopter (c) Satelite

Fig. 1. Performance of bounded model checking (BMC) for three neural-network controllers using our algorithm (blue line) and ReluPlex (red line). The x-axis is the number of steps used in BMC, y-axis is the time taken in seconds. A timeout of 1 hour was used. For the Pendulum model, our algorithm approach terminated after finding a counter-example on the 51st step (after approximately 30 minutes).

As we can see, bounded model checking with our algorithm is significantly more efficient than using ReluPlex. This is primarily because our algorithm *directly* computes the strongest postcondition on network output after each step, which can then be re-used efficiently in the strongest-postcondition computation for the next timestep. By contrast, ReluPlex is a conjunctive decision procedure which must consider each timestep separately, making verification of additional timesteps progressively more challenging.

4 RELATED WORK

Xiang et al. [30] solve the problem of exactly computing the reach set of a neural network given an arbitrary convex input polytope. However, the authors use an algorithm that relies on explicitly enumerating all exponentially-many (2^n) possible signs at each ReLU layer. By contrast, our algorithm adapts to the actual input polytopes, efficiently restricting its consideration to activations that are actually possible. Thrun [26] presents an early approach for extraction of if-then-else rules from artificial neural networks. Bastani et al. [6] learn decision tree policies guided by a DNN policy that was learned via reinforcement learning. This decision tree could be seen as a particular form of symbolic representation of the underlying DNN. Scheibler et al. [22] verify the safety of a machine-learning controller with BMC using the SMT-solver iSAT3, but support small unrolling depths and basic safety properties. Zhu et al. [31] use a synthesis procedure to generate a safe deterministic program that can enforce safety conditions by monitoring the deployed DNN and preventing potentially unsafe actions. The presence of adversarial and fooling inputs for DNNs as well as applications of DNNs in safety-critical systems has led to efforts to verify and certify DNNs [3, 5, 8, 12, 13, 17, 19, 23, 28]. *Approximate reachability analysis* for neural networks safely overapproximates the set of possible outputs [11, 13, 27–30].

5 CONCLUSION AND FUTURE WORK

We extended our *symbolic neural network representation* from Sotoudeh and Thakur [24] to two-dimensional input regions and presented an algorithm for computing the representation. We then applied this symbolic representation to the problem of *bounded model checking* of DNN controllers, and discussed how the representation allows us to more efficiently solve the problem than prior work using ReluPlex. For future work, we would like to scale our algorithms to larger networks, incorporate over-approximations to handle non-PWL networks, and extend our approach to support unbounded model checking [16].

REFERENCES

[1] 2019. Stanford CS231n Convolutional Neural Networks for Visual Recognition: Commonly used activation functions. http://cs231n.github.io/neural-networks-1/#actfun. Accessed: 2019-11-15.

[2] 2019. SyReNN: Symbolic Representations for Neural Networks. https://github.com/95616ARG/SyReNN. Accessed: 2019-11-15.

[3] Greg Anderson, Shankara Pailoor, Isil Dillig, and Swarat Chaudhuri. 2019. Optimization and Abstraction: A Synergistic Approach for Analyzing Neural Network Robustness. *CoRR* abs/1904.09959 (2019).

[4] Roberto Bagnara, Patricia M Hill, and Enea Zaffanella. 2008. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming* 72, 1-2 (2008), 3–21.

[5] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya V. Nori, and Antonio Criminisi. 2016. Measuring Neural Net Robustness with Constraints. In *Advances in Neural Information Processing Systems*.

[6] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* 2499–2509.

[7] Dirk Beyer. 2016. Reliable and reproducible competition results with benchexec and witnesses (report on SV-COMP 2016). In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Springer, 887–904.

[8] Rudy R. Bunel, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and Pawan Kumar Mudigonda. 2018. A Unified View of Piecewise Linear Neural Network Verification. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* 4795–4804.

[9] L. de Moura and N. Bjørner. 2008. Z3: An Efficient SMT Solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018).

[11] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. 2018. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*. Springer, 121–138.

[12] Ruediger Ehlers. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*.

[13] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

[16] Kryštof Hoder and Nikolaj Bjørner. 2012. Generalized property directed reachability. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 157–171.

[17] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification (CAV)*.

[18] Kyle D Julian, Mykel J Kochenderfer, and Michael P Owen. 2018. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics* 42, 3 (2018), 598–608.

[19] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification (CAV)*.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.

[22] Karsten Scheibler, Leonore Winterer, Ralf Wimmer, and Bernd Becker. 2015. Towards Verification of Artificial Neural Networks. In *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, MBMV 2015, Chemnitz, Germany, March 3-4, 2015.* 30–40.

[23] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. 2019. An abstract domain for certifying neural networks. *PACMPL* 3, POPL (2019), 41:1–41:30.

[24] Matthew Sotoudeh and Aditya V. Thakur. 2019. Computing Linear Restrictions of Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS), 2019.*

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

(CVPR).

[26] Sebastian Thrun. 1994. Extracting Rules from Artifical Neural Networks with Distributed Representations. In *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*. 505–512.

[27] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. 1599–1614.

[28] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane S. Boning, and Inderjit S. Dhillon. 2018. Towards Fast Computation of Certified Robustness for ReLU Networks. In *International Conference on Machine Learning, (ICML)*.

[29] Weiming Xiang, Hoang-Dung Tran, Joel A. Rosenfeld, and Taylor T. Johnson. 2018. Reachable Set Estimation and Safety Verification for Piecewise Linear Systems with Neural Network Controllers. In *2018 Annual American Control Conference, (ACC)*.

[30] Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. 2017. Reachable set computation and safety verification for neural networks with ReLU activations. *arXiv preprint arXiv:1712.08163* (2017).

[31] He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. 2019. An inductive synthesis framework for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019*. 686–701.