Quantitative Robustness Analysis of Neural Networks*

Mara Downing
maradowning@cs.ucsb.edu
University of California Santa Barbara
USA

ABSTRACT

Neural networks are an increasingly common tool for solving problems that require complex analysis and pattern matching, such as identifying stop signs or processing medical imagery. Accordingly, verification of neural networks for safety and correctness is of great importance, as mispredictions can have catastrophic results in safety critical domains. One metric for verification is robustness, which answers whether or not a misclassified input exists in a given input neighborhood. I am focusing my research at quantitative robustness—finding not only if there exist misclassified inputs within a given neighborhood but also how many exist as a proportion of the neighborhood size. My overall goal is to expand the research on quantitative neural network robustness verification and create a variety of quantitative verification tools geared towards expanding our understanding of neural network robustness.

CCS CONCEPTS

• **Software and its engineering** → **Formal software verification**; **Software testing and debugging**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Neural Network Verification, Robustness, Quantitative Verification

ACM Reference Format:
Mara Downing. 2023. Quantitative Robustness Analysis of Neural Networks. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '23), July 17–21, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3597926.3605231

1 INTRODUCTION

With the growing prevalence of neural networks, especially in safety critical domains like self-driving cars or medical diagnoses, evaluating the correctness of these networks is essential. While accuracy measures on a previously unseen dataset give one metric for this correctness, neural networks have been shown to be vulnerable to adversarial attacks in which a correctly classified input is perturbed in some small manner [35] to make the network produce a misclassification that a human analyzing the same information would not. Thus, analyzing the **robustness** of neural networks against these types attacks is an important part of assuring safety.

*This material is based on research supported by NSF under Award #2124039

ISSTA '23, July 17–21, 2023, Seattle, WA, USA

ACM ISBN 979-8-4007-0221-1/23/07.
https://doi.org/10.1145/3597926.3605231

In this paper, I describe my work up to this point in quantitative robustness, my plan for an upcoming project in quantitative robustness, and then my further research plans in the field.

Problem Description. There exist a number of verifiers for the robustness of full-precision networks [3, 5, 6, 8–11, 14, 17, 20, 21, 23, 25, 26, 29–32, 34, 36, 38–40]. However, the majority of these verifiers look at a traditional (non-quantitative) verification problem—asking whether or not a misclassified input exists in a perturbation region. A more in-depth analysis with quantitative verification asks instead how many misclassified inputs exist within the radius, to give a more detailed analysis of how vulnerable a network is to adversarial attacks—how likely it is to encounter one of these misclassifications. Within this quantitative realm, there exist a handful of full-precision quantitative verifiers [3, 25, 36, 39]. Of these verifiers, [25, 36] describe methods for quantitative verification based on symbolic analysis, which can produce reasonably precise results but are difficult to scale. Alternatively, sampling-based quantitative verifiers [3, 39] are more scalable and able to handle very large networks, but struggle to make a conclusive distinction between fully robust and almost fully robust regions.

Moving from full-precision networks, there exist a class of neural networks called quantized networks, where the weights, biases, and computations use generally smaller, fixed-point values to save storage space and allow for faster computation [2, 15]. These can be limited as far as 1 bit, which is a special class called binary networks. There exist traditional and quantitative verifiers for these binary networks [1, 4, 19, 24, 41], but to the best of my knowledge there only exist traditional verifiers [13, 16, 18, 22, 28] for quantized networks with greater than binary precision. Here I see another area in which the current state of the art is insufficient to solve a problem—traditional verification cannot always provide a full picture of the robustness of a network, but for quantized networks no quantitative robustness verification currently exists.

With this current state of the art, in Section 2 I give a brief overview of feedforward neural networks and robustness, in Section 3 I introduce my research in creating a quantitative verifier for quantized neural networks, in Section 4 I lay out my plan for more scalable quantitative analysis of full-precision networks using interval analysis, and finally in Section 5 I briefly go over future plans beyond my upcoming experiments.

2 BACKGROUND

To explain my research contributions and future plans I will briefly summarize how a feedforward neural network with the ReLU (Rectified Linear Unit) activation function is used for classification tasks. I also give a brief definition of quantitative robustness within a perturbation region, and a quick explanation of why this allows for greater analysis of networks than traditional verification methods.

2.1 Feedforward Neural Networks with ReLU Activations

In Figure 1, I show a small neural network with two input features x_0 and x_1, one hidden layer with two nodes, and two possible output classifications y_0 and y_1. Weights and biases are marked along their edges, or above the nodes where they are added.

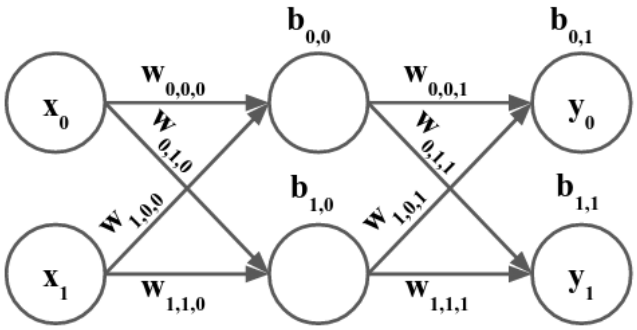

Figure 1: Small example network with 2 input features, 1 hidden layer with 2 nodes, and 2 output nodes.

To compute the value of a node, the values of nodes in the previous layer are multiplied by their respective weights (shown along the arrows), and then added together as a sum with the bias value. For the output nodes, this is the total computation. However, for internal nodes the ReLU activation function is applied—if the result of the weights and bias computation is less than 0, the node is set to 0, otherwise the node is set to the result of the weights and bias computation.

Once values have been computed for all output nodes, the output node with the greatest value determines the classification that the network returns.

2.2 Robustness within a Perturbation Region

Robustness for a neural network, at a high level, is the ability of the network to withstand small adversarial attacks—essentially, to be able to make correct predictions even in the presence of noise. If a human observer could ignore the noise and correctly interpret the data, then a neural network should also be able to do so as well. To evaluate robustness, a common approach is to choose a region around a correctly classified input and find if any misclassifications exist within the region. I will call this the perturbation region.

Selecting this region is a matter of choosing small modifications that could be made to the correctly classified input that should not produce meaningful differences, and thus should not produce a different classification.

Traditional robustness asks a yes/no question—does a misclassified input exist in a given perturbation region. However, a yes/no answer to a verification query about robustness does not give any information about how many of the perturbations change the output. For example in Fig. 2, both of these examples would be determined not robust by a traditional verifier. As both neural networks fail the robustness test, we cannot determine which one misclassifies fewer perturbed inputs. Alternatively, with quantitative verification the number of misclassified inputs is counted, and thus a distinction can be made. A network with a higher number of misclassified inputs in a given perturbation region is less robust (and, thus, more prone to adversarial attacks) than a network with fewer misclassified inputs in the same region.

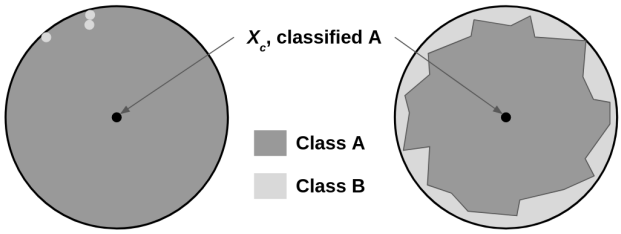

Figure 2: Image of two perturbation regions about an input, with different numbers of adversaries within the radius.

For a given neural network $\mathcal{N}$, an input $X_c = \langle x_0, \cdots, x_{N-1} \rangle$ (the center of the perturbation), and a perturbation limit $\Delta^{lim} = \langle \delta_0^{lim}, \cdots, \delta_{n-1}^{lim} \rangle$ (denoting the perturbation limit value per feature), the quantitative robustness measure $R(\mathcal{N}, X_c, \Delta^{lim})$ is as follows:

$$R(\mathcal{N}, X_c, \Delta^{lim}) = |S_{RobustSet}|/|S_{PerturbRegion}| \quad \text{where}$$
$$S_{RobustSet} = \{\tilde{X_c} \mid \arg\max \mathcal{N}(\tilde{X_c}) = \arg\max \mathcal{N}(X_c) \wedge x_i - \delta_i^{lim} \leq \tilde{x_i} \leq x_i + \delta_i^{lim}\}$$
$$S_{PerturbRegion} = \{\tilde{X_c} \mid x_i - \delta_i^{lim} \leq \tilde{x_i} \leq x_i + \delta_i^{lim}\}$$

In the definition above, $S_{PerturbRegion}$ denotes the set of all perturbed inputs within the perturbation region Δ^{lim}, and $S_{RobustSet}$ denotes the set of all perturbed inputs within the radius where the output of $\mathcal{N}$ does not change.

This is a general form of robustness definition that can represent attacks such as one or two-pixel attacks, or general L_∞ ball constraints over the input [33, 34].

3 QUANTITATIVE VERIFICATION FOR QUANTIZED NEURAL NETWORKS

The goal of this direction of research is to produce a working quantitative verifier for quantized neural networks and show some improvement over a simple sampling-based robustness evaluation. I created a tool which uses probabilistic symbolic execution [12] to count the number of incorrectly classified inputs within a perturbation region around a correctly classified input.

Briefly, symbolic execution is a verification technique in which a program is evaluated not by testing individual inputs, but rather by constructing a symbolic execution tree of all of the possible paths through the program. Constraints are recorded at each node to mark what must be true of the input to reach that node.

At a branch point, one or more of the possible paths may be infeasible, and if a path is infeasible there is no need to traverse further down. The feasibility of a path can be checked using a constraint solver, which checks to see if the path constraints at each node are satisfiable (if there is at least one assignment of the variables in the constraint that makes the constraint true).

Beyond satisfiability, probabilistic symbolic execution draws on model counting, which returns the number of solutions to a satisfiable constraint, or 0 if the constraint is unsatisfiable.

The first key aspect involved in the creation of this tool was finding translations between fixed-point constraints and equivalent integer constraints, so that I could use an integer model counter to count misclassified inputs at the end of the symbolic execution tree. This allowed me in turn to evaluate the performance of multiple linear integer arithmetic model counters to find which one was best equipped for counting the constraints generated by neural networks.

The next key aspect involved creating optimization strategies to improve symbolic execution time, specifically to reduce the time taken by a constraint solver during symbolic execution. Through experiments, I was able to find two successful optimization strategies which resulted in a notable decrease in verification time.

I currently have a paper in submission at Automated Software Enginnering (ASE) 2023 with the results of this research, including the quantitative verification tool. Briefly, I will summarize the results of one experiment from the paper.

Figure 3 shows the results of a direct comparison between my symbolic quantitative verifier and a concrete enumeration scheme where distinct inputs are sampled from the perturbation region until a time limit, run concretely with the network, and incorrect classifications are counted. The networks tested here all contained between 20 and 70 internal nodes. First, I ran the symbolic verifier on a set of networks and perturbations with a 10 minute timeout, then for each of those experiments I ran the concrete enumeration for the same exact time that symbolic analysis had taken (for example, if for an individual test the symbolic method took 52 seconds, concrete enumeration was given 52 seconds for the same test). This chart shows that the symbolic verifier I created performs better on over half of the tested cases, either by producing an exact result (which in these experiments would have taken 5.42 hours, the time for all inputs to be evaluated concretely), or by finding more incorrectly classified inputs in the same time as concrete enumeration.

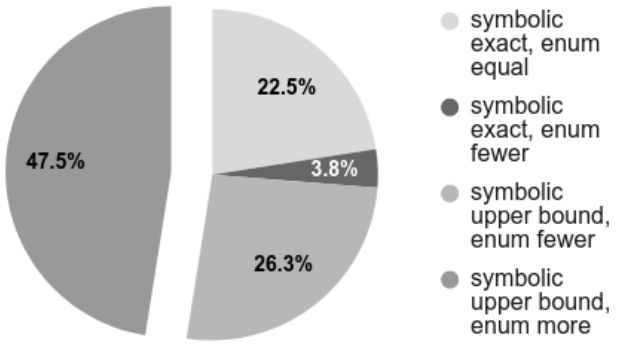

Figure 3: Comparison between my symbolic quantitative verifier and concrete enumeration for evaluating robustness. The three slices on the right of the graph are cases where symbolic verification performed better.

4 INTERVAL ABSTRACTION FOR QUANTITATIVE ROBUSTNESS OF FULL-PRECISION NETWORKS

Interval analysis, and other forms of abstraction, have been used to great effect in traditional neural network verification [10, 11, 32]. This technique can also be used for finding a bound on the minimum adversarial distortion, as shown in [40], by running the abstract analysis multiple times with different perturbation regions.

My goal with this research is to leverage the success of interval analysis to produce an estimate of the quantitative robustness when full robustness cannot be proven, for a more scalable quantitative verification approach than the current state of the art symbolic verifiers, but with the ability to guarantee full robustness unlike sampling-based verifiers.

For a brief overview of interval analysis, given a network and a known range by which each input feature ($x_0 \cdots x_{N-1}$) is perturbed, it is possible to propagate these ranges through the network and find an upper and lower limit for the value of each internal node, followed by an upper and lower limit for each of the output nodes ($y_0 \cdots y_{J-1}$).

With concrete values, the output classification of the network is decided by finding the greatest output node value. With intervals, we have two notable options: the correct (expected) output classification has an interval where the lowest value is greater than all other output node interval maximums, or there is some overlap. With traditional verification, this corresponds respectively to two outcomes—either the network is robust on all inputs within the perturbation region, or the robustness cannot be determined conclusively (it is also technically possible for the correct output interval to be fully below all other output node intervals, but with a robustness query there should be one "central" input which is known to be classified correctly, so I do not explore this outcome). In traditional verification, these two outcomes are the end of this analysis. However, it is intuitive that a greater overlap between intervals indicates lower robustness than a slight overlap, so I seek to add more analysis to the unknown result of traditional interval analysis and estimate quantitative robustness for cases where full robustness cannot be proven.

For this goal, I will start by adding a volume computation on the output intervals using the tool Vinci [7]. With this volume computation I can find the proportion of the interval space for which the output node indicating the correct classification is greater than all other output nodes. I do this by dividing the volume of the space where that node is greatest by the volume of the output interval space without node comparison. This, on its own, gives a measure of quantitative robustness. However, in the interval analysis leading up to this volume computation a great deal of information has been lost, so I plan to provide further refinement through a splitting technique loosely based on the PReach-I variant of the PReach tool [27].

Going back to the beginning of the interval analysis, after computing the possible interval of values for the first hidden node h_0, it is possible to split this interval by whether or not the value is positive or negative—whether or not the ReLU sets the value to 0 or not. At this point, we can estimate the proportion of that interval that takes each branch of that condition, and begin to construct

a tree of intervals rather than a single path through the program. At the leaves of the tree, there will still be a set of intervals for each output node, but there will also be a probability attached—the likelihood (based on an assumption of uniform input distribution) that the necessary set of decisions was taken to reach this specific leaf node. The sum of these probabilities across all leaf nodes must equal 1. Using this probability and the same volume computation as before, I can construct a more accurate estimation of the quantitative robustness.

This approach leads to two key research questions:

RQ1: How should the probability of each path be decided at a split?

RQ2: What is the best balance between accuracy and scalability to produce usable results?

For **RQ1**, there are two simple probability strategies I plan to analyze: uniform and Gaussian splitting. Figure 4 shows this distinction—assuming a uniform distribution and interval bounds $[-1, 2]$ gives a probability of 33.33% that the node will be set to 0, whereas assuming a Gaussian distribution with $\sigma = 0.17 * interval\_len$, $\mu = (interval\_max + interval\_min)/2$, and the same bounds gives a probability of 16.34% that the node will be set to 0. I chose 0.17 of the interval length for σ as this will make the distance between $-\sigma$ and σ about one third of the interval length.

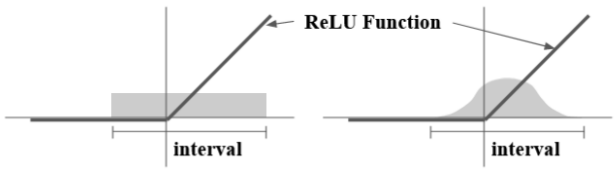

Figure 4: ReLU activation functions with a uniform and Gaussian possible distribution of input values overlaid.

My hypothesis is that the Gaussian distribution will produce more accurate results, as the weights and bias computation used to produce each input to the ReLU will include a sum of multiple values which are not directly dependent on each other.

Next, for **RQ2**, I plan to set up a tunable input variable which determines how far into the network the splitting strategy is used, before the traditional single interval strategy is applied on each branch of the tree for any remaining nodes. This will allow me to analyze the trade-off between scalability and accuracy with more refinement than just splitting or not splitting. I expect that splitting for a higher proportion of nodes will increase accuracy, but it will also cause exponential blowup in the size of the tree, so the main focus of this research will be finding a good balance so that results can be obtained quickly with meaningful accuracy.

For my evaluation, I plan to first produce a "ground truth" for the robustness of a set of perturbation queries by sampling and testing randomly selected inputs from the perturbation region for a long period of time (at least 2 hours). I will do a brief statistical analysis on these results to evaluate the probabilistic likelihood that they are within 5% of the actual robustness.

Once I have this ground truth, I will evaluate the same robustness queries using interval analysis as described above, varying the parameters of path probability function and proportion of evaluation with splitting. I will analyze both the time taken for each experiment and the accuracy of the results with relation to the ground truth determined by sampling.

Additionally, I plan to evaluate direct comparisons with sampling given the same time (the same approach as is shown in Section 3), as well as a comparison with the existing full precision quantitative verification tool described in [36]. With these experiments I hope to show that this interval analysis can scale better than more precise methods for quantitative analysis and also produce more accurate results than sampling in a short time frame.

At the end of this study I aim to produce a interval-based quantitative verification tool, with results showing its advantages over the current state of the art quantitative verification approaches.

5 FUTURE PLANS

For more distant future work, I plan to investigate how quantitative analysis can be applied to NLP models, as their current successes and drawbacks are coming to the forefront with ChatGPT. This and similar tools have made great strides in language processing, but have also shown some notable and dangerous drawbacks. One of these is the creation and citation of fake sources, which has shown to have concerning consequences such as accusing people of crimes they could not have committed [37]. This I believe is a clear case where traditional verification will not suffice. We already know a tool like ChatGPT can be incorrect or produce false citations, but for adequate analysis of the risk we need to know **how often** this can occur—which necessitates quantitative analysis.

6 CONCLUSION

In sum, my goal for my dissertation is to show a variety of ways in which quantitative verification can be performed on neural networks to improve our current knowledge about these networks. I have worked already to create a quantitative verifier for the robustness quantized networks, a type of network that quantitative verification has not been applied to yet. Moving forward, I plan to explore options for more scalable quantitative robustness verification via interval analysis. Further into the future, I aim to look at specific domains (such as NLP) to analyze using quantitative techniques. Overall, I seek to employ quantitative verification in a variety of ways and on a variety of types of networks, to expand upon current quantitative verification options.

REFERENCES

[1] Guy Amir, Haoze Wu, Clark Barrett, and Guy Katz. 2021. An SMT-based approach for verifying binarized neural networks. In *Tools and Algorithms for the Construction and Analysis of Systems: 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27–April 1, 2021, Proceedings, Part II 27*. Springer, 203–222. https://doi.org/10.26226/morressier.604907f41a80aac83ca25cda

[2] Pascal Bacchus, Robert Stewart, and Ekaterina Komendantskaya. 2020. Accuracy, training time and hardware efficiency trade-offs for quantized neural networks on fpgas. In *International symposium on applied reconfigurable computing*. Springer, 121–135. https://doi.org/10.1007/978-3-030-44534-8_10

[3] Teodora Baluta, Zheng Leong Chua, Kuldeep S Meel, and Prateek Saxena. 2021. Scalable quantitative verification for deep neural networks. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 312–323. https://doi.org/10.1109/icse43902.2021.00039

[4] Teodora Baluta, Shiqi Shen, Shweta Shinde, Kuldeep S Meel, and Prateek Saxena. 2019. Quantitative verification of neural networks and its security applications. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1249–1264. https://doi.org/10.1145/3319535.3354245

[5] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. 2019. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3240–3247. https://doi.org/10.1609/aaai.v33i01.33013240

[6] Elena Botoeva, Panagiotis Kouvaros, Jan Kronqvist, Alessio Lomuscio, and Ruth Misener. 2020. Efficient verification of relu-based neural networks via dependency analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3291–3299. https://doi.org/10.1609/aaai.v34i04.5729

[7] Benno Büeler, Andreas Enge, and Komei Fukuda. 2000. Exact volume computation for polytopes: a practical study. In *Polytopes—combinatorics and computation*. Springer, 131–154. https://doi.org/10.1007/978-3-0348-8438-9_6

[8] Rudy Bunel, P Mudigonda, Ilker Turkaslan, P Torr, Jingyue Lu, and Pushmeet Kohli. 2020. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research* 21, 2020 (2020). https://doi.org/10.23919/acc.2018.8431048

[9] Shaoru Chen, Eric Wong, J Zico Kolter, and Mahyar Fazlyab. 2022. Deepsplit: Scalable verification of deep neural networks via operator splitting. *IEEE Open Journal of Control Systems* 1 (2022), 126–140. https://doi.org/10.1109/ojcsys.2022.3187429

[10] Mahyar Fazlyab, Manfred Morari, and George J Pappas. 2020. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Trans. Automat. Control* (2020). https://doi.org/10.1109/tac.2020.3046193

[11] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3–18. https://doi.org/10.1109/sp.2018.00058

[12] Jaco Geldenhuys, Matthew B. Dwyer, and Willem Visser. 2012. Probabilistic symbolic execution. In *International Symposium on Software Testing and Analysis, ISSTA 2012, Minneapolis, MN, USA, July 15-20, 2012*, Mats Per Erik Heimdahl and Zhendong Su (Eds.). ACM, 166–176. https://doi.org/10.1145/2338965.2336773

[13] Mirco Giacobbe, Thomas A Henzinger, and Mathias Lechner. 2020. How many bits does it take to quantize your neural network?. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Vol. 12079. https://doi.org/10.1007/978-3-030-45237-7_5

[14] Divya Gopinath, Kaiyuan Wang, Mengshi Zhang, Corina S Pasareanu, and Sarfraz Khurshid. 2018. Symbolic execution for deep neural networks. *arXiv preprint arXiv:1807.10439* (2018).

[15] Yunhui Guo. 2018. A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752* (2018).

[16] Thomas A Henzinger, Mathias Lechner, and Đorđe Žikelić. 2021. Scalable verification of quantized neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3787–3795. https://doi.org/10.1609/aaai.v35i5.16496

[17] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *International conference on computer aided verification*. Springer, 3–29. https://doi.org/10.1007/978-3-319-63387-9_1

[18] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713. https://doi.org/10.1109/cvpr.2018.00286

[19] Kai Jia and Martin Rinard. 2020. Efficient exact verification of binarized neural networks. *Advances in neural information processing systems* 33 (2020), 1782–1795.

[20] 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*. Springer, 97–117. https://doi.org/10.1007/978-3-319-63387-9_5

[21] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. 2019. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 443–452. https://doi.org/10.1007/978-3-030-25540-4_26

[22] Haowen Lin, Jian Lou, Li Xiong, and Cyrus Shahabi. 2021. Integer-arithmetic-only Certified Robustness for Quantized Neural Networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7828–7837. https://doi.org/10.1109/iccv48922.2021.00773

[23] Wang Lin, Zhengfeng Yang, Xin Chen, Qingye Zhao, Xiangkun Li, Zhiming Liu, and Jifeng He. 2019. Robustness verification of classification deep neural networks via linear programming. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11418–11427. https://doi.org/10.1109/cvpr.2019.01168

[24] Nina Narodytska, Shiva Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. 2018. Verifying properties of binarized deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. https://doi.org/10.1609/aaai.v32i1.12206

[25] Corina Păsăreanu, Hayes Converse, Antonio Filieri, and Divya Gopinath. 2020. On the probabilistic analysis of neural networks. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 5–8.

[26] Luca Pulina and Armando Tacchella. 2010. An abstraction-refinement approach to verification of artificial neural networks. In *International Conference on Computer Aided Verification*. Springer, 243–257. https://doi.org/10.1007/978-3-642-14295-6_24

[27] Seemanta Saha, Mara Downing, Tegan Brennan, and Tevfik Bultan. 2022. PReach: a heuristic for probabilistic reachability to identify hard to reach statements. In *Proceedings of the 44th International Conference on Software Engineering*. 1706–1717. https://doi.org/10.1145/3510003.3510227

[28] Luiz Sena, Xidan Song, Erickson Alves, Iury Bessa, Edoardo Manino, and Lucas Cordeiro. 2021. Verifying Quantized Neural Networks using SMT-Based Model Checking. *arXiv preprint arXiv:2106.05997* (2021).

[29] Luiz H Sena, Iury V Bessa, Mikhail R Gadelha, Lucas C Cordeiro, and Edjard Mota. 2019. Incremental Bounded Model Checking of Artificial Neural Networks in CUDA. In *2019 IX Brazilian Symposium on Computing Systems Engineering (SBESC)*. IEEE, 1–8. https://doi.org/10.1109/sbesc49506.2019.9046094

[30] Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T Vechev. 2018. Fast and Effective Robustness Certification. *NeurIPS* 1, 4 (2018), 6.

[31] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2018. Boosting robustness certification of neural networks. In *International Conference on Learning Representations*.

[32] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 1–30. https://doi.org/10.1145/3290354

[33] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23, 5 (2019), 828–841. https://doi.org/10.1109/tevc.2019.2890858

[34] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. 2018. Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 109–119. https://doi.org/10.1145/3238147.3238172

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[36] Muhammad Usman, Divya Gopinath, and Corina S Păsăreanu. 2021. QuantifyML: How Good is my Machine Learning Model? *arXiv preprint arXiv:2110.12588* (2021).

[37] Pranshu Verma and Will Oremus. 2023. ChatGPT invented a sexual harassment scandal and named a real law prof as the accused. *The Washington Post* (Apr 2023). Retrieved May 24, 2023 from https://www.washingtonpost.com/technology/2023/04/05/chatgpt-lies/

[38] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Efficient formal safety analysis of neural networks. *Advances in neural information processing systems* 31 (2018).

[39] Stefan Webb, Tom Rainforth, Yee Whye Teh, and M Pawan Kumar. 2018. A statistical approach to assessing neural network robustness. *arXiv preprint arXiv:1811.07209* (2018).

[40] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. 2018. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems* 31 (2018).

[41] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, and Taolue Chen. 2021. BDD4BNN: A BDD-based Quantitative Analysis Framework for Binarized Neural Networks. *arXiv preprint arXiv:2103.07224* (2021). https://doi.org/10.1007/978-3-030-81685-8_8

Received 2023-05-24; accepted 2023-06-07