On the Probabilistic Analysis of Neural Networks

Corina Păsăreanu
Carnegie Mellon University, NASA Ames
corina.s.pasareanu@nasa.gov

Hayes Converse
University of Texas at Austin
hayesconverse@utexas.edu

Antonio Filieri
Imperial College London
a.filieri@imperial.ac.uk

Divya Gopinath
KBR Inc., NASA Ames
divya.gopinath@nasa.gov

ABSTRACT

Neural networks are powerful tools for automated decision-making, seeing increased application in safety-critical domains, such as autonomous driving. Due to their black-box nature and large scale, reasoning about their behavior is challenging. Statistical analysis is often used to infer probabilistic properties of a network, such as its robustness to noise and inaccurate inputs. While scalable, statistical methods can only provide probabilistic guarantees on the quality of their results and may underestimate the impact of low probability inputs leading to undesired behavior of the network.

We investigate here the use of symbolic analysis and constraint solution space quantification to precisely quantify probabilistic properties in neural networks. We demonstrate the potential of the proposed technique in a case study involving the analysis of ACAS-Xu, a collision avoidance system for unmanned aircraft control.

KEYWORDS

Neural networks, symbolic execution, probabilistic analysis.

1 INTRODUCTION

Neural networks are used sucessfully in solving many complex tasks, including applications in safety-critical and mission-critical domains. Numerous techniques have been proposed to reason about properties of neural networks [4, 6, 7, 11, 13, 18].

The focus of many current techniques is on the synthesis of adversarial inputs that *fool* the network into producing wrong classifications [17]. Specialized tools can provide point-wise adversarial robustness guarantees [14], but they only check within a limited region of the input domain. A counter-example to the robustness check acts as evidence that the network is not robust, but beyond this there is no further information regarding the number of such adversaries or probability of such an adversarial input to occur given a realistic probability distribution over the input domain.

Furthermore, current techniques have trouble scaling to large networks due to the large input spaces and large number of paths that need to be analyzed. As a result, probabilistic analysis of neural networks is often performed using statistical methods [16, 19].

SEAMS '20, October 7–8, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7962-5/20/05.
https://doi.org/10.1145/3387939.3391594

Statistical methods are attractive as they offer both speed and information regarding the number of adversaries around a point. However, given that robustness analysis frequently depends on finding rare inputs, a sampling campaign's bias towards common inputs can lead such conclusions towards inaccuracy regarding the number of adversaries at best and false robustness declarations at worst.

In this work, we investigate the use of symbolic analysis techniques to precisely quantify the confidence of probabilistic arguments about neural networks. Specifically, we use a form of *concolic* execution to collect symbolic constraints characterizing the inputs that would lead to the same activation pattern and decision. We then use solution space quantification techniques to estimate the probability mass concentrated around the concrete execution. This allows us to compute the *probability* (or likelihood) of executing different paths through the network. The observed activation patterns are stored in a compact data structure allowing for early rejection of inputs that bring redundant information. An *amplification* procedure is adopted for a targeted analysis of the possible decision space resulting from an observed activation pattern, beyond the decision taken for the concrete input that initially triggered it.

We implemented our approach and evaluated its effectiveness in performing robustness analysis for ACAS-Xu, a safety-critical collision avoidance system for unmanned aircraft control [12]. Our preliminary experimental evaluation shows that our approach can provide robustness measures with precise confidence guarantees, as it is desirable for the analysis of safety-critical systems.

2 APPROACH

The key insight of the proposed approach is the use of solution space quantification techniques to compute the probability of executing a path through the neural network. Each path corresponds to a specific activation pattern – i.e., which neurons get activated when processing the input.

An activation pattern in a ReLU network can be characterized by the conjunction of a set of linear inequalities over the input domain, one for each activated node. Each inequality represents a halfplane splitting the domain into two parts, and their conjunction defines a polytope in the input domain. We will refer to this polytope as an *activation condition* (*AC*).

The set of all activation conditions induces a partition of the input domain. Therefore, determining the activation pattern of one concrete input allows us to infer information about the entire equivalence class of inputs satisfying the corresponding activation condition. In particular, we can use the volume of the AC polytope

as a measure of *how much* of the input domain would lead to the same activation. Furthermore, by adding to AC the constraints imposed by the logits layer for the desired decision, referred to as a *decision condition* (*DC*), we can use the volume computation to quantify the portion of the input space that leads to the desired final decision.

2.1 Path Exploration

The path exploration can be done systematically, as typically performed in symbolic execution, or heuristically, for instance we can collect the activation conditions from inputs generated from a user-defined population model. We perform the latter in our experiments. The use of more advanced generation strategies, e.g., aiming at maximizing the diversity of the inputs, is orthogonal to the contributions of this work and left for future investigation.

In general, an exhaustive collection of all the feasible activation conditions may be infeasible in practice due to the large size of most realistic neural networks. Our analysis provides an *anytime* approach and it gradually improves its results with more paths explored. Furthermore the analysis is able to precisely quantify how much of the input space has been covered, providing a measure of *confidence* in the results.

When the network is executed on a given input, we compute for each node the condition on the input values that leads to its observed state, be that activated or deactivated. Notably, a ReLU node is activated when an affine function of its inputs evaluates to a positive value. If activated, it produces as output the value of the affine function, otherwise, it produces as output 0. Since the output of each node in the network is a (conditional) analytical expression of its inputs, the activation condition of a node can always be expressed as an affine condition on the initial inputs of the network, resulting in a constraint of the form $\mathbf{c} \cdot \mathbf{x} + b > 0$ (where $\mathbf{x}$ is the input vector, $\mathbf{c}$ is a vector of coefficients specific to the node, and b is a scalar bias value). If the node is not activated, its respective activation condition is naturally of the form $\mathbf{c} \cdot \mathbf{x} + b \leq 0$.
Decision conditions. After an input is executed all the way through the network, the activation condition that accompanies it is finished with a *decision condition* (DC), a small set of constraints that represent the network's decision in the final layer to output desired class $\mathcal{D}$. For example, a network with n nodes at its output layer representing n possible decisions that selects the greatest value among its output nodes would have a decision constraint of the form $\bigwedge_{i \neq k} o_i > o_k$, for desired class $\mathcal{D} = i$. Note that given an AC, we append to it a DC as described earlier for every node of the output layer even if the original input led to only one of the classes, thus creating n different constraints. In this way we **amplify** the effect of the input, by searching for all possible classes within the input region that satisfy the same activation constraint.

2.2 Probabilistic analysis

Each activation pattern discovered during the path exploration can be seen as the representative of an equivalence class on the input values satisfying the corresponding activation condition. Such an activation condition, being the conjunction of affine constraints on the input, identifies a polytope in $\mathbb{R}^d$, where d is the dimension of the input vector over the Reals.

Given a distribution on the input domain, the probability for the network to produce a decision $\mathcal{D}$ can be formalized as a function of the activation conditions AC that lead to the decision $\mathcal{D}$ ($AC \rightsquigarrow \mathcal{D}$).

Assume first that the input distribution is uniform. In this case, this probability would be the ratio between the volume of the polytope and the size of the input domain $Pr(\mathcal{D}) = Vol(AC)/Vol(\mathbb{D}_x)$.

To handle non-uniform input distributions, we follow an approach similar to [2] requiring the input distribution to be discretized in an arbitrarily accurate histogram distribution. A histogram distribution is a map from a partition of the input domain to a probability value representing the cumulative probability mass of the element set of the partition. Partitions can be arbitrarily fine, allowing us to strike a trade-off between the accuracy of the discretization and the computational cost of the probabilistic analysis. Given a histogram distribution $H : s_i \rightarrow p_i$, mapping a subset of the input domain $s_i \subseteq \mathbb{D}_x$ to its probability value p_i (with $\bigcup_i s_i = \mathbb{D}_x$ and $i \neq j \implies s_i \cap s_j = \emptyset$; we assume w.l.o.g. s_i expressed as a boolean constraint on $\mathbf{x}$), the probability can be written as:

$$Pr(\mathcal{D}) = \sum_{s_i} p_i \cdot \sum_{AC \rightsquigarrow \mathcal{D}} \frac{Vol(AC \wedge s_i)}{Vol(\mathbb{D}_x)} \tag{1}$$

In other words, the complexity of the computation grows linearly with the size of the support of the histogram distribution. If the sets s_i are constructed as boxes (i.e., hyper-rectangles in the input domain), the computation of their probability reduces to the evaluation of the cumulative distribution function of each input of the network in the vertices of each box, we adopted this discretization approach for the evaluation of this work. How to compute an optimal discretization, where the sets s_i can be arbitrary constraints, is specific to each input distribution and orthogonal to our approach.
Volume of a polytope. Equation (1) reduces the problem of computing the probability of a network producing a specific decision $\mathcal{D}$ for a given input distribution to the quantification of the volume of polytopes. A variety of algorithms have been studied for this calculation, including both exact and approximate solutions [3, 5, 10].
Exact solutions. To analyze networks with up to tens of distinct inputs we can use efficient off-the-shelf implementations of exact volume computation methods [3]. In particular, we adopt the implementation of Lasserre's method [15] provided by the Vinci solver [3]. This method applies directly to the representation of a polytope as a conjunction of halfplanes. In our experiments, Vinci required minutes of computation for polytopes defined by up to 200 halfplanes on 5 input variables. More efficient exact methods can be used after computing the vertices of the polytope explicitly. However, the number of vertices of a polytope may be exponential in the number of halfplanes, in the worst case, making their computation expensive for deeper networks, where the number of halfplanes grows linearly with the number of nodes.
Sound intervals. To analyze networks with a larger number of nodes, instead of directly computing the exact volume of the resulting polytopes, we compute an arbitrarily small interval guaranteed to contain the exact solution. For this task, we use a set of branch-and-bound and interval constraint propagation techniques implemented in Realpaver [9]. Starting from the initial input domain, Realpaver produces a set of boxes whose union is guaranteed to contain all the solutions of a given set of constraints. The set of

boxes is iteratively refined to produce tighter approximations of the solution space, up to an arbitrary target accuracy (or a predefined timeout). The resulting boxes are labeled as either inner or outer boxes. Inner boxes are guaranteed to contain only solutions. Outer boxes may instead contain also points that do not satisfy the constraints (i.e., are outside the polytope). The volume of a box can be computed efficiently as the product of the lengths of its sides. By construction, the cumulative volume of the inner boxes is a sound lower bound of the volume of the polytopes provided as constraint, while the total cumulative volume of inner and outer boxes is a sound upper bound of the volume of the polytope.

In our experiments, Realpaver required approximately 12 minutes per problem for problems with over 300 constraints and 5 input variables, and a time on the order of seconds for similar problems with only 2 input variables. The number of such problems for a given experiment ranges depending on how many paths are identified during the exploration phase, but as the calls to Realpaver can be parallelized, the computation time can be reduced to a scale of tens of minutes using a larger number of cores.

Approximate solutions. Input spaces with higher dimensionality may challenge the scalability of our bounding method, requiring the use of statistical volume estimation methods. While state of the art statistical methods can scale on high-dimensional polytopes [5], their results are confidence intervals, which are only probabilistically correct.

We also remark that while in this work we focus on affine activation functions, the method described in this section can be applied to more complex activation functions, provided the adoption of suitable methods for quantifying their solution space. If an interval approximation is acceptable, several nonlinear activation functions can be analyzed using Realpaver as described before in this section. Other interval constraint propagation or branch and bound tools (e.g., Ibex [1]) may support a different set of constraints or perform more efficiently.

Confidence measures. When the volume of the activation constraints is computed exactly, the probabilistic symbolic analysis presented in this section produces as a byproduct the quantification of the volume of each of the activation patterns AC_i covered during the sampling phase. The ratio between the cumulative volume of the covered activation patterns and the volume of the valid input domain ($\sum_i Vol(AC_i)/Vol(\mathbb{D}_x)$) provides a natural confidence measure for the result of the analysis, quantifying how much of the input domain has been covered. If combined with an input distribution, a straightforward adaptation of Equation (1) allows to quantify the total probability mass from the input population that has been accounted for by the analysis.

3 APPLICATION

We applied our technique to perform robustness and sensitivity analysis for the ACAS-Xu network. ACAS-Xu is a safety-critical collision avoidance system for unmanned aircraft control [12]. It receives sensor information regarding the drone (the *ownship*) and any nearby intruder drones, and then issues horizontal turning advisories aimed at preventing collisions.

The FAA is exploring an implementation of ACAS-Xu that uses an array of 45 deep neural networks, from which we selected one network for our analysis.

The ACAS-Xu network has been the subject of adversarial robustness analysis in the past [8, 14]. In [14], the Reluplex solver was used to prove local adversarial robustness around randomly chosen inputs. The network was said to be ϵ-locally-robust at input point x if for every x', s.t. $||x - x'||_\infty \leq \epsilon$ the network assigns the same label for x and x'. For a given ϵ the analysis either returns UNSAT indicating robustness, or returns SAT indicating the presence of an adversary. We employ our analysis to not only determine adversarial robustness with high confidence (when the desired label has a 100% probability), but also provide insight into the sensitivity of the network at that point with respect to other labels (when there is a non-zero distribution of probabilities across labels). Specifically, given an input x and an ϵ, we generate inputs from within the region defined by $||x - x'|| \leq \epsilon$. We then apply our probabilistic analysis to determine the probability distribution across all the labels. For the ACAS-Xu network, we only apply the ϵ perturbation to the first two dimensions (range and θ). This is based on feedback we received from the domain experts on the ACASXU application, that inputs can be considered truly adversarial to the network only if they share the values of the last three dimensions with nearby correctly classified inputs and only differ in the first two dimensions.

We took a set of five distinct inputs for each label (for a total of 25 inputs), relaxed the first two dimensions by 5% (ϵ = 0.05) in both directions and generated 10,000 inputs from within that region. With this input set, we applied our analysis to determine the distribution of probabilities. We rejected an average of 91% of these inputs using our technique. This indicates that most of the inputs followed the same set of paths and that our technique is able to efficiently identify and process those paths. We appended decision constraints corresponding to the five different labels to each path constraint, as described above, to identify the volume for every label within the region covered by the path. We fed the resulting path condition set to Realpaver. From this analysis we obtained the inner and outer boxes corresponding to each path and label and computed the respective volumes and probabilities (Section 2).

Table 1 highlights the results of our technique on these inputs. We display the probabilities obtained based on both the inner boxes (under-approximation) and the inner+outer boxes (over-approximation) in the "P" columns under each label. The confidence value is calculated as the sum of the probabilities obtained from the inner boxes for each of the labels. This corresponds to the total amount of the input space covered by the precise calculation of the probabilities by our technique. We observed that for some inputs, like most of those with label 0, the network is highly robust around the input point with high confidence. This is not always the case: input 2 belonging to label 0 has a 21% and 22% chance of being mis-classified to labels 1 or 2, respectively. This gives us an indication of the vulnerability of the network around these inputs. We also obtain a quantification of this sensitivity and an identification of the labels that the network may mis-classify to. The inputs from labels 1, 3, and 4 appeared from our observations to be more susceptible to adversarial perturbations. However, they

Table 1: ACASXU 5% Perturbation Probabilities
(10,000 Samples; P: min, max percentages / SP: percentage ± confidence interval, ± 0 when only one label has been sampled)

Label	Input #	Label 0		Label 1		Label 2		Label 3		Label 4		Confidence
		P	SP	P	SP	P	SP	P	SP	P	SP	P
0	0	98.58, 100	100 ± 0	0, 0	0 ± 0	0, 0	0 ± 0	0, 0	0 ± 0	0, 0	0 ± 0	98.58
0	1	99.13, 100	100 ± 0	0, 0	0 ± 0	0, 0	0 ± 0	0, 0	0 ± 0	0, 0	0 ± 0	99.13
0	2	52.94, 63.70	53.27 ± 0.98	21.64, 22.05	22.5 ± 0.82	22.78, 23.29	23.5 ± 0.83	0, 0	0 ± 0	0.69, 0.70	0.73 ± 0.17	98.04
1	0	24.94, 25.26	25.15 ± 0.85	23.68, 24.06	24.97 ± 0.85	26.70, 27.09	27.92 ± 0.88	5.43, 5.60	6.55 ± 0.48	13.14, 13.54	15.41 ± 0.71	93.90
1	3	34.46, 34.97	34.9 ± 0.93	36.54, 37.53	37.9 ± 0.95	0, 0	0 ± 0	23.84, 25.02	27.2 ± 0.87	*1.03e-03, 1.04e-03*	*0 ± 0*	94.84
1	4	50.63, 51.62	51.16 ± 0.98	14.31, 14.77	15.18 ± 0.70	23.40, 24.39	24.12 ± 0.84	8.99, 9.14	9.54 ± 0.58	0, 0	0 ± 0	97.32
2	0	9.17, 9.28	9.18 ± 0.57	0, 0	0 ± 0	42.31, 42.89	43.67 ± 0.97	0, 0	0 ± 0	41.80, 42.68	47.15 ± 0.98	93.28
2	1	14.52, 14.67	14.5 ± 0.69	0, 0	0 ± 0	39.39, 39.87	40.36 ± 0.96	0, 0	0 ± 0	40.66, 41.46	45.14 ± 0.98	94.57
3	0	0, 0	0 ± 0	74.73, 77.56	77.65 ± 0.82	0.20, 0.25	0.23 ± 0.09	17.38, 19.16	19.6 ± 0.78	2.07, 2.35	2.52 ± 0.31	94.39
3	1	2.13, 2.15	2.15 ± 0.28	31.18, 31.66	32.4 ± 0.92	0, 0	0 ± 0	57.07, 57.85	62.56 ± 0.95	2.17, 2.20	2.89 ± 0.33	92.55
4	0	0, 0	0 ± 0	28.93, 29.33	29.76 ± 0.90	34.10, 34.55	34.3 ± 0.93	10.64, 10.91	11.39 ± 0.62	23.24, 23.69	34.55 ± 0.84	96.90
4	1	0.99, 1.00	1.03 ± 0.20	0, 0	0 ± 0	38.59, 39.00	39.68 ± 0.96	8.16, 8.30	9.43 ± 0.57	48.18, 48.85	49.86 ± 0.98	95.92
4	2	*3.46e-7, 3.51e-7*	*0 ± 0*	0, 0	0 ± 0	20.17, 20.42	21.78 ± 0.81	18.55, 19.07	21.06 ± 0.80	52.23, 53.02	57.16 ± 0.97	90.95

are not equally likely to be mis-classified to the other labels, indeed, some examples have targeted robustness against mis-classification to certain other labels. In a more general pattern, inputs with label 2 appear to have consistent targeted robustness against labels 1 and 3. This is useful information in terms of the network application, since it means that what should be a "weak right" advisory will only ever be either "strong right" or "clear of conflict" in the event of noise within our specified boundary, never advising the drone to turn entirely the wrong way.

We also performed pure statistical analysis based on the same 10,000 sample set to determine the probabilities for the different labels for the same set of inputs. Table 1 shows the results in the "SP" columns under each label. Each probability is shown in terms of its statistical confidence interval. We observed that the base probabilities obtained from statistical analysis matched those obtained from our white-box analysis, however, ours are more dependable as they are based on precise volume calculation. Our analysis also has the ability to find non-zero probabilities for labels for which the statistical analysis counterpart finds a probability of zero, indicating potential adversaries that would be difficult to otherwise discover. When observed, these probabilities tended to be small. However, false robustness guarantees of any kind can be a serious issue for safety-critical applications.

REFERENCES

[1] Ignacio Araya, Gilles Trombettoni, Bertrand Neveu, and Gilles Chabert. 2014. Upper bounding in inner regions for global optimization under inequality constraints. *Journal of Global Optimization* 60, 2 (01 Oct 2014), 145–164. https://doi.org/10.1007/s10898-014-0145-7

[2] Mateus Borges, Antonio Filieri, Marcelo d'Amorim, Corina S. Păsăreanu, and Willem Visser. 2014. Compositional Solution Space Quantification for Probabilistic Software Analysis. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. ACM, New York, NY, USA, 123–132. https://doi.org/10.1145/2594291.2594329

[3] Benno Büeler, Andreas Enge, and Komei Fukuda. 2000. *Exact Volume Computation for Polytopes: A Practical Study*. Birkhäuser Basel, Basel, 131–154. https://doi.org/10.1007/978-3-0348-8438-9_6

[4] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. 2018. Output Range Analysis for Deep Feedforward Neural Networks. In *NASA Formal Methods - 10th International Symposium, NFM 2018, Newport News, VA, USA, April 17-19, 2018, Proceedings*.

[5] Ioannis Z. Emiris and Vissarion Fisikopoulos. 2018. Practical Polytope Volume Approximation. *ACM Trans. Math. Softw.* 44, 4, Article 38 (June 2018), 21 pages. https://doi.org/10.1145/3194656

[6] Matteo Fischetti and Jason Jo. 2017. Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study. *CoRR* abs/1712.06174 (2017).

[7] 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*. 3–18. https://doi.org/10.1109/SP.2018.00058

[8] Divya Gopinath, Guy Katz, Corina S. Pasareanu, and Clark Barrett. 2017. DeepSafe: A Data-driven Approach for Checking Adversarial Robustness in Neural Networks. https://arxiv.org/abs/1710.00486.

[9] Laurent Granvilliers and Frédéric Benhamou. 2006. Algorithm 852: Realpaver: an interval solver using constraint satisfaction techniques. *ACM Transactions on Mathematical Software (TOMS)* 32, 1 (2006), 138–156.

[10] Peter Gritzmann and Victor Klee. 1994. On the complexity of some basic problems in computational convexity: I. Containment problems. *Discrete Mathematics* 136, 1 (1994), 129 – 174. https://doi.org/10.1016/0012-365X(94)00111-U

[11] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In *CAV*.

[12] K. Julian, J. Lopez, J. Brush, M. Owen, and M. Kochenderfer. 2016. Policy Compression for Aircraft Collision Avoidance Systems. In *Proc. 35th Digital Avionics System Conf. (DASC)*. 1–10. https://doi.org/10.1109/DASC.2016.7778091

[13] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *CAV*.

[14] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*. 97–117. https://doi.org/10.1007/978-3-319-63387-9_5

[15] J. B. Lasserre. 1983. An analytical expression and an algorithm for the volume of a convex polyhedron inRn. *Journal of Optimization Theory and Applications* 39, 3 (01 Mar 1983), 363–377. https://doi.org/10.1007/BF00934543

[16] Ravi Mangal, Aditya V. Nori, and Alessandro Orso. 2019. Robustness of neural networks: a probabilistic and practical approach. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results, ICSE (NIER) 2019, Montreal, QC, Canada, May 29-31, 2019*. 93–96. https://doi.org/10.1109/ICSE-NIER.2019.00032

[17] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2013. Intriguing Properties of Neural Networks. Technical Report. http://arxiv.org/abs/1312.6199.

[18] 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*.

[19] Lily Weng, Pin-Yu Chen, Lam M. Nguyen, Mark S. Squillante, Akhilan Boopathy, Ivan V. Oseledets, and Luca Daniel. 2019. PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 6727–6736. http://proceedings.mlr.press/v97/weng19a.html