Checking Probabilistic Properties of Neural Networks via Symbolic Methods and Sampling

Ravi Mangal Aditya V. Nori Alessandro Orso

Georgia Institute of Technology
{rmangal3, orso}@gatech.edu

Microsoft Research
adityan@microsoft.com

***Abstract*—The study of verification and testing techniques for neural networks is in its infancy. The formal problem statement as well as the techniques for solving the problem are constantly evolving. We forward two informal hypotheses that, if true, can help guide future research. First, correctness properties of neural network properties need to be probabilistic in order to capture the practical correctness semantics of neural networks. Second, checking such properties will require a synergistic combination of sampling and symbolic techniques.**

The theory of neural networks is riddled with open questions such as, (i) Why do gradient descent based learning algorithms efficiently learn neural networks that show low rates of training error? (ii) Why are such trained networks able to generalize to unseen data? (iii) Given a trained neural network, how can we ensure that it is adversarially robust, fair, and secure? While the first two questions require an analysis of the algorithms used for training neural networks, the third question requires analyzing trained neural networks. Since a trained neural network is essentially a computational object representing a function from a real-valued vector to a real-valued vector ($\mathbb{R}^n \rightarrow \mathbb{R}^m$), it is possible to employ tools from program verification and testing literature to analyze these objects.

The starting point of any verification or testing technique is a logical specification of program correctness. Neural networks, however, are used exactly when such logical correctness specifications are unavailable. The only available specification is in the form of input-output data. In this context, besides accuracy with respect to the available training data, what property of neural networks can we possibly check? Recent works have revealed a number of desirable properties about the structure of the function represented by a neural network. For instance, the presence of adversarial examples [1] has revealed that it is desirable for a neural network to represent a Lipschitz function, such that small changes in the input are guaranteed to result in bounded changes in the output. Other proposed properties of neural networks include fairness [2], and privacy [3], [4]. In general, while defining full functional correctness specifications is not possible for neural networks, partial specifications have found popularity.

Although neural networks represent a deterministic mapping from inputs to outputs, the real-world processes that they model have inherent randomness. A real-world process might be envisaged as follows: The inputs are generated according to some distribution D. Given an input x from D, there is a deterministic function mapping x to the output y. The goal of the neural network learning algorithm is to infer this mapping given (x_i, y_i) pairs as training data. It is natural then to reason about the properties of a neural network with respect to the input distribution. Although the input distribution is typically unknown, it can be approximated based on the available data. In such a scenario, probabilistic properties can more effectively capture practical correctness semantics of neural networks. For instance, if a neural network exhibits lack of robustness only at very unlikely inputs, it might be reasonable to declare such a network as robust. Another argument in favor of probabilistic properties is based on the fact that learning algorithms are themselves stochastic, and so, it is highly unlikely that a trained neural network would exhibit correct behavior over the entire input space. This leads to our first hypothesis.

Hypothesis One. *The inherent stochastic nature of neural network training algorithms strongly suggests expressing global correctness properties in a probabilistic form.*

Standard program verification techniques are geared towards finding proofs of program correctness, i.e., proving that the program satisfies the property of interest on all inputs. Standard testing techniques are geared towards finding counter-examples that exhibit property violation. Neither of these perspectives is suitable for checking probabilistic properties. Checking correctness on all inputs is unnecessary, while finding a single property violating input is not sufficient for showing probabilistic violation. A simple approach for checking probabilistic properties is to sample inputs from the input distribution, execute the neural network on each sample, and check if the property is violated by the sample. The ratio of the number of samples that violate the property to the total number of samples drawn gives an estimate of the probability of property satisfaction. Although simple, in order to compute accurate estimates for the potentially unlikely event of property violation, a prohibitively large number of samples might need to be drawn. We believe that a symbolic analysis of the neural networks can help reduce redundant sampling, and improve the sampling efficiency. Examples of such symbolic techniques that might be helpful include abstract interpretation, symbolic execution, and weakest precondition computation. This leads to our second hypothesis.

Hypothesis Two. *Probabilistic properties can be checked by reduction to a sampling problem. To improve sample efficiency, we can employ symbolic techniques for analyzing the neural networks.*

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013. [Online]. Available: http://arxiv.org/abs/1312.6199

[2] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12, 2012.

[3] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, 2016.

[4] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, 2015.