# Toward Scalable Verification for Safety-Critical Deep Networks

Lindsey Kuper
Parallel Computing Lab, Intel Labs
lindsey.kuper@intel.com

Guy Katz
Stanford University
guyk@cs.stanford.edu

Justin Gottschlich
Parallel Computing Lab, Intel Labs
justin.gottschlich@intel.com

Kyle Julian
Stanford University
kjulian3@stanford.edu

Clark Barrett
Stanford University
barrett@cs.stanford.edu

Mykel J. Kochenderfer
Stanford University
mykel@stanford.edu

## ABSTRACT

The increasing use of deep neural networks for safety-critical applications, such as autonomous driving and flight control, raises concerns about their safety and reliability. Formal verification can address these concerns by guaranteeing that a deep learning system operates as intended, but the state of the art is limited to small systems. In this work-in-progress report we give an overview of our work on mitigating this difficulty, by pursuing two complementary directions: devising scalable verification techniques, and identifying design choices that result in deep learning systems that are more amenable to verification.

**ACM Reference Format:**
Lindsey Kuper, Guy Katz, Justin Gottschlich, Kyle Julian, Clark Barrett, and Mykel J. Kochenderfer. 2018. Toward Scalable Verification for Safety-Critical Deep Networks. In *Proceedings of SysML Conference (SysML)*. ACM, New York, NY, USA, 3 pages.

## 1 INTRODUCTION

Machine learning systems, and, in particular, deep neural networks (DNNs), are becoming a widely used and effective means for tackling complex, real-world problems [4]. However, a major obstacle to the use of DNNs in safety-critical systems, such as autonomous driving or flight control systems, is the great difficulty in providing formal guarantees about their behavior.

A powerful technique for formal verification of properties of a software artifact is to encode the artifact and the property one wishes to prove about it as a *satisfiability modulo theories* (SMT) formula, and then use an SMT solver to prove that the property holds or find a counterexample showing that it does not. While it is possible to verify properties of neural networks using SMT solvers, until recently the technique only scaled to toy-sized networks of fewer than ten neurons [12].

Yet, for the practical adoption of SMT-based DNN verification, we must be able to verify properties of DNNs of up to thousands (or more) of neurons. To do this, we advocate a two-pronged approach. First, we propose the development of specialized, efficient SMT solvers that are well-suited for DNN verification problems. Second, we propose designing DNNs in ways that make them more amenable to SMT-based verification. These two approaches complement each other, and we observe that design choices that make a DNN more amenable to verification are also desirable for other reasons, such as improved speed of inferencing, smaller memory requirements, and reduced power footprint.

*SysML, February 15-16, 2018, Stanford, California, USA*

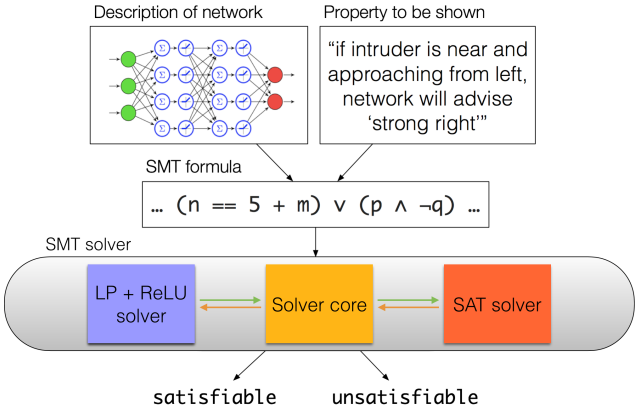


**Figure 1: Overview of the Reluplex architecture. Reluplex takes as input a network description and a property we wish to prove about the network's behavior, both expressed as an SMT formula. The SMT solver incorporates a domain-specific linear programming (LP) + ReLU theory solver that interacts with an underlying SAT solver and determines whether the formula is satisfiable.**

## 2 SCALING UP SMT-BASED VERIFICATION OF NEURAL NETWORKS

A primary focus of this work is in extending the capabilities of automated verification tools such as SMT solvers to formally verify properties of DNNs used for safety-critical systems. A major challenge of verifying properties of DNNs with SMT solvers is in handling the networks' activation functions.

Each neuron of a neural network computes a weighted sum of its inputs according to learned weights. It then passes that sum through an activation function to produce the neuron's final output. Typically, the activation functions (e.g., sigmoid) introduce nonlinearity to the network, making DNNs capable of learning arbitrarily complex functions, but also making the job of automated verification tools much harder, in some cases moving the problem from P to NP.

Using SMT solvers to verify properties of neural networks involves encoding the network and the property in question as formulas in some *theory*, such as the theory of linear real arithmetic. Our work leverages the observation that, apart from their activation functions, neural networks can be expressed using conjunctions of linear real arithmetic formulas, which are straightforward to handle using standard linear programming (LP) solving algorithms. It is also possible to express *piecewise-linear* activation functions, such

as rectified linear units (ReLUs), as part of linear arithmetic formulas, but every ReLU in a network then introduces a disjunction to the formula. These disjunctions quickly cause an exponential increase in the state space that the SMT solver must explore to prove properties about the network, thus limiting the applicability and scalability of the approach.

In a recent paper [8], we proposed an improved SMT-based algorithm, called Reluplex, capable of verifying properties of networks that are an order of magnitude larger than previously possible. Reluplex mitigates the difficulty posed by activation functions through a *lazy* approach, which often makes it possible to eliminate many activation functions from the problem without changing the result. It extends the theory of linear real arithmetic by introducing a new "ReLU" predicate that can be split into disjuncts lazily, making it possible to avoid exploring large parts of the state space. As a result, the Reluplex solver can verify networks that are notably larger than what was previously possible. For example, we used Reluplex to verify safety properties of a DNN used as the controller for a prototype of the ACAS Xu aircraft collision avoidance system [7].

The lazy-ReLU-splitting technique that Reluplex uses is an example of the general problem-solving strategy of *exploiting high-level domain-specific abstractions for efficiency* that has proven fruitful in a variety of areas. For example, in the setting of high-performance domain-specific languages, a high-level representation of programmer intent enables compiler optimizations and smart scheduling choices that would be difficult or impossible otherwise [1]. The use of high-level abstractions not only does not compromise high performance, but actually enables it. Reluplex's lazy ReLU splitting is another such optimization, made possible by the addition of the high-level ReLU predicate to the theory used by the solver. The higher-level representation makes it possible to determine the satisfiability of a formula more efficiently than if the problem were expressed at a lower level.

An important lesson here for scalable verification is that we have much to gain by not treating SMT solvers as black boxes, but instead developing *domain-specific theory solvers* like Reluplex that are uniquely suited to the verification task at hand. We are currently working on extending Reluplex to handle piecewise-linear approximations of other commonly used activation functions to be able to handle a wider variety of networks.

## 3 DESIGNING VERIFICATION-FRIENDLY NEURAL NETWORKS

In addition to improving the scalability of verification tools, a complementary direction for scaling DNN verification is to design the networks themselves in a way that makes them more amenable to verification. When designing a neural network, some of the obvious design decisions are related to the topology of the network, such as the number of hidden layers and their dimensions. It is not surprising that, from a verification point of view, smaller networks are generally easier to handle. Developers of neural networks may opt to use a smaller network, perhaps achieving lower accuracy, in order to enjoy the benefits of verification. On the other hand, recent work [5] suggests that it is possible to significantly reduce the storage requirements of neural networks without compromising accuracy. Although the motivation for this work was ease of deployment of neural networks in resource-constrained settings rather than ease of verification, these pruned, quantized networks may also be easier for verification tools to handle than uncompressed networks.

Our initial experiments suggest that the size of the network is not necessarily the only factor to consider; the network topology is also important. We have observed that networks with many layers with a few neurons each are generally easier for the solver to handle than networks with few layers, but many neurons in each layer.

An extreme way of applying this principle is by discretizing parts of the neural network in question, effectively turning it into a family of smaller networks. The ACAS Xu network [7] used this approach due to considerations that did not include verification — rather, due to hardware constraints, the developers found that many smaller networks were preferable to one large one. However, the discretization step also made it easier to verify properties of each of the smaller networks. A similar approach could facilitate the verification of other systems as well.

Another decision with consequences for the scalability of verification is activation function selection. These choices can have far-reaching effects. For example, some of the more successful verification efforts thus far [3, 8] have focused on piecewise-linear activation functions, such as ReLUs or max-pooling layers, while attempts to verify networks with sigmoid activation functions have proved far less scalable.

In addition to network topology and activation functions, there are several other potential avenues to explore. One example is low-precision DNN arithmetic, which is an increasingly popular way to accelerate DNN training and inferencing [6]. The simplicity and smaller size of low-precision networks will make them more amenable to verification than full-precision networks [2, 11], as well as more suitable for use on low-power edge devices [10]. It may even be the case that hardware accelerator techniques that optimize inference on low-precision networks could be used to speed up verification of those same networks.

## 4 CONCLUSION

Verifying that neural networks behave as intended may soon become a limiting factor in their applicability to real-world, safety-critical systems such as those used to control autonomous vehicles and aircraft. Recent work revealing neural networks' vulnerability to adversarial inputs [13], including in physical-world attacks [9], makes meeting this challenge more urgent.

Verification is a promising avenue for mitigating this difficulty, but additional work is required to scale up verification techniques to be practically applicable to modern DNNs. Initial work by us and others points to two complementary avenues that could achieve the sought-after scalability: first, the design of verification algorithms tailored for neural networks (e.g., by enriching the theories used by SMT solvers); and second, the creation and use of design principles for neural networks that produce DNNs that are more amenable to verification (e.g., model topology and activation function selection). We believe that through additional work in these directions, verification could be successfully applied to many real-world deep learning systems.

## REFERENCES


[1] Hassan Chafi, Arvind K. Sujeeth, Kevin J. Brown, HyoukJoong Lee, Anand R. Atreya, and Kunle Olukotun. 2011. A Domain-Specific Approach to Heterogeneous Parallelism. In *Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming (PPoPP '11)*. ACM, New York, NY, USA, 35–46. https://doi.org/10.1145/1941553.1941561

[2] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. 2017. Verification of Binarized Neural Networks. *CoRR* abs/1710.03107 (2017). arXiv:1710.03107 http://arxiv.org/abs/1710.03107

[3] Rüdiger Ehlers. 2017. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. *CoRR* abs/1705.01320 (2017). arXiv:1705.01320 http://arxiv.org/abs/1705.01320

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

[5] Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015). arXiv:1510.00149 http://arxiv.org/abs/1510.00149

[6] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *CoRR* abs/1609.07061 (2016). arXiv:1609.07061 http://arxiv.org/abs/1609.07061

[7] Kyle Julian, Jessica Lopez, Jeffrey S. Brush, Michael Owen, and Mykel J. Kochenderfer. 2016. Policy Compression for Aircraft Collision Avoidance Systems. In *Digital Avionics Systems Conference (DASC)*. https://doi.org/10.1109/DASC.2016.7778091

[8] 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

[9] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016). arXiv:1607.02533 http://arxiv.org/abs/1607.02533

[10] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar. 2016. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 1–12. https://doi.org/10.1109/IPSN.2016.7460664

[11] N. Narodytska, S. P. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh. 2017. Verifying Properties of Binarized Deep Neural Networks. *CoRR* abs/1709.06662 (2017). arXiv:1709.06662 http://arxiv.org/abs/1709.06662

[12] Luca Pulina and Armando Tacchella. 2010. An Abstraction-refinement Approach to Verification of Artificial Neural Networks. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV'10)*. Springer-Verlag, Berlin, Heidelberg, 243–257. https://doi.org/10.1007/978-3-642-14295-6_24

[13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199 (2013). arXiv:1312.6199 http://arxiv.org/abs/1312.6199