# A Dual Approach to Verify and Train Deep Networks*

**Sven Gowal**, **Krishnamurthy Dvijotham**, **Robert Stanforth**, **Timothy Mann** and **Pushmeet Kohli**

DeepMind

{sgowal, dvij, stanforth, timothymann, pushmeet}@google.com

## Abstract

This paper addressed the problem of formally verifying desirable properties of neural networks, *i.e.*, obtaining provable guarantees that neural networks satisfy specifications relating their inputs and outputs (*e.g.*, robustness to bounded norm adversarial perturbations). Most previous work on this topic was limited in its applicability by the size of the network, network architecture and the complexity of properties to be verified. In contrast, our framework applies to a general class of activation functions and specifications. We formulate verification as an optimization problem (seeking to find the largest violation of the specification) and solve a Lagrangian relaxation of the optimization problem to obtain an upper bound on the worst case violation of the specification being verified. Our approach is *anytime*, *i.e.*, it can be stopped at any time and a valid bound on the maximum violation can be obtained. Finally, we highlight how this approach can be used to train models that are amenable to verification.

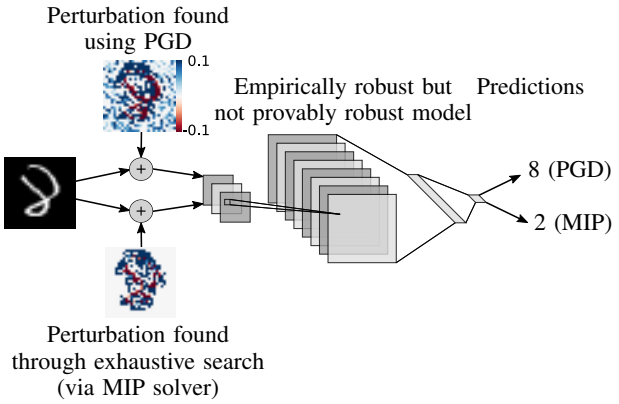



Figure 1: Example motivating why robustness to projected gradient descent (PGD) attacks is not a true measure of robustness. Given a seemingly robust neural network, the worst-case perturbation of size $\epsilon = 0.1$ found using 200 PGD iterations and 10 random restarts (shown at the top) is correctly classified as an "eight". However, a worst case perturbation classified as a "two" can be found through exhaustive search (shown at the bottom).

## 1 Introduction

Despite the successes of deep learning [Goodfellow *et al.*, 2016], it is well-known that neural networks are not robust. In particular, it has been shown that the addition of small but carefully chosen deviations to the input, called adversarial perturbations, can cause the neural network to make incorrect predictions with high confidence [Carlini and Wagner, 2017a; Carlini and Wagner, 2017b; Goodfellow *et al.*, 2014; Kurakin *et al.*, 2016; Szegedy *et al.*, 2013]. Starting with Szegedy *et al.* [2013], there has been a lot of work on understanding and generating adversarial perturbations [Carlini and Wagner, 2017b; Athalye and Sutskever, 2017], and on building models that are robust to such perturbations [Goodfellow *et al.*, 2014; Papernot *et al.*, 2015; Madry *et al.*, 2017; Kannan *et al.*, 2018]. Unfortunately, many of the defense strategies proposed in the literature are targeted towards a specific adversary (*e.g.*, obfuscating gradients against projected gradient attacks), and as such they are easily broken by stronger adversaries [Uesato *et al.*, 2018; Athalye *et al.*, 2018]. Robust optimization techniques, like the one developed by Madry *et al.* [2017], overcome this problem by trying to find the worst-case adversarial examples at each training step and adding them to the training data. While the resulting models show strong empirical evidence that they are robust against many attacks, we cannot yet guarantee that a different adversary (for example, one that does brute-force enumeration to compute adversarial perturbations) cannot find inputs that cause the model to predict incorrectly.

In fact, Figure 1 provides an example that motivates why projected gradient descent (PGD) – the technique at the core of Madry *et al.*'s method – does not always find the worst-case attack (a phenomenon also observed in [Tjeng *et al.*, 2017]). This has driven the need for *formal verification*: a provable guarantee that neural networks are consistent with a *specification* for all possible inputs to the network.

*Originally published under the title "A Dual Approach to Scalable Verification of Deep Networks" at the Conference on Uncertainty in Artificial Intelligence (UAI) 2018, where it received the best paper award.

**Complete methods.** Verification of neural networks has seen significant research interest in recent years. In the formal verification community, Satisfiability Modulo Theory (SMT) solvers have been adapted for verification of neural networks [Ehlers, 2017; Katz *et al.*, 2017]. More recently, researchers also proposed a set of approaches that make use of branch-and-bound algorithms either directly or via Mixed-Integer Programming (MIP) solvers [Bunel *et al.*, 2017; Cheng *et al.*, 2017; Tjeng *et al.*, 2017]. These approaches rely heavily on the piecewise linear structure of neural networks; and, while they achieve strong results on smaller networks, scaling them to large networks remains an open challenge (since they perform exhaustive enumeration in the worst case).

**Scalable incomplete methods.** This has led researchers to study incomplete verification algorithms. Typically, incomplete algorithms work by computing upper bounds on the worst case violation of the specification being verified. If the upper bound is smaller than zero, the property being verified is indeed true. However, if it is not, the property may still be true but the algorithm could not prove it. Algorithms have been derived based on ideas from abstract interpretation [Mirman *et al.*, 2018], propagating bounds through the network [Gowal *et al.*, 2018; Weng *et al.*, 2018], analyzing the Lipschitz properties of the network [Weng *et al.*, 2018] and using convex optimization and duality theory [Raghunathan *et al.*, 2018; Kolter and Wong, 2017]. While significant progress has been made and impressive results obtained, the search for efficient *tight* verification procedures (*i.e.*, algorithms that compute an upper bound close to the true maximum violation) is ongoing.

**Contributions.** The method proposed in this paper is an incomplete method based on optimization and duality. At the time of writing [Dvijotham *et al.*, 2018b], our results improved upon prior work in the following ways:

1. Our verification approach applies to *arbitrary* feed-forward neural networks with *any architecture* and *any activation function* and our framework recovers previous results [Ehlers, 2017] when applied to the special case of piecewise linear activation functions.

2. We can handle verification of systems with discrete inputs and combinatorial constraints on the input space, including cardinality constraints.

3. The computation involved only requires solving an unconstrained convex optimization problem (of size linear in the number of neurons in the network), which can be done using a sub-gradient method efficiently. Further, our approach is *anytime*, in the sense that the computation can be stopped at any time and a valid bound on the verification objective can be obtained.

4. We attain state-of-the-art verified bounds on adversarial error rates on image classifiers trained on MNIST and CIFAR-10 under adversarial perturbations in the infinity norm.

## 2 Verification as an Optimization

**Neural network.** We focus on feed-forward neural networks. During training, the network is fed pairs of input $x^{\text{nom}}$ and correct output label $y^{\text{true}}$, and trained to minimize a loss, such as cross-entropy for classification tasks or squared error for regression tasks.

For clarity of presentation, we assume that the neural network is defined by a sequence of transformations $h_k$ for each of its $K$ layers. That is, for an input $x^0$ (which we define formally in the next paragraph), we have

$$x^k = h^k(x^{k-1}) \quad k = 1, \ldots, K. \tag{1}$$

The output of the network is $x^K$.

**Verification problem.** We are interested in verifying that neural networks satisfy a specification by generating a proof that this specification holds. We consider specifications that require that for all inputs in some set $\mathcal{S}^{\text{in}}(x^{\text{nom}})$ around $x^{\text{nom}}$, the network output satisfies a linear relationship

$$c^\mathsf{T} x^K + d \leq 0 \quad \forall x^0 \in \mathcal{S}^{\text{in}}(x^{\text{nom}}) \tag{2}$$

where $c$ and $d$ are a vector and a scalar that may depend on the nominal input $x^{\text{nom}}$ and label $y^{\text{true}}$. As shown in [Dvijotham *et al.*, 2018b], many useful verification problems fit this definition (*e.g.*, robustness to adversarial attacks, monotonic predictors, cardinality constraints).

**Optimization problem.** Verifying a specification like (2) can be done by searching for a counter-example that violates the specification constraint:

$$\begin{aligned} \max_{x^0 \in \mathcal{S}^{\text{in}}(x^{\text{nom}})} \quad & c^\mathsf{T} x^K + d \\ \text{subject to} \quad & x^k = h^k(x^{k-1}) \quad k = 1, \ldots, K \end{aligned} \tag{3}$$

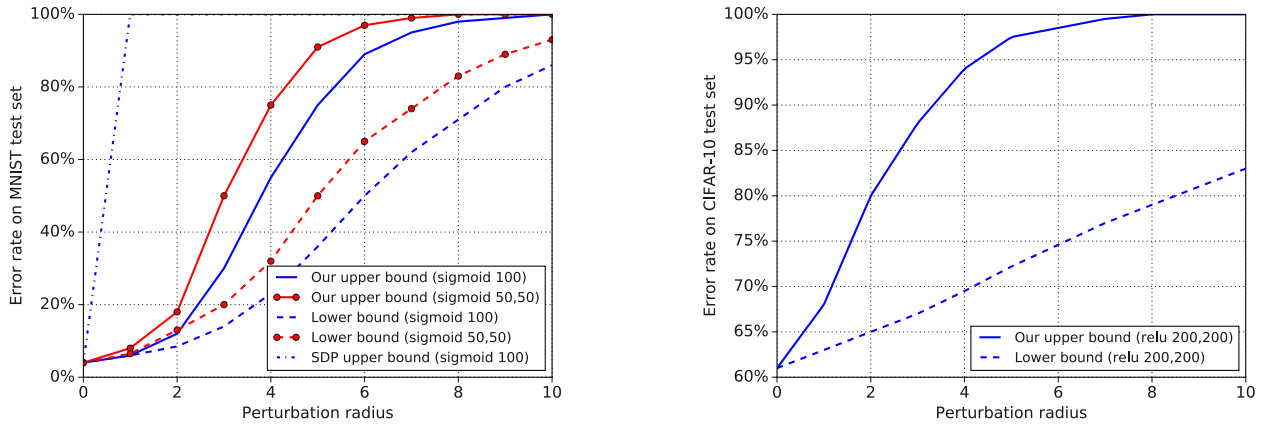
If the optimal value of the above optimization problem is smaller than 0, the specification (2) is satisfied.

## 3 Lagrangian Relaxation

We assume that bounds on the activations are available, such that $\underline{x}^k \leq x^k \leq \overline{x}^k$ for all $k = 1, \ldots, K$. Given the input constraints $x^0 \in \mathcal{S}^{\text{in}}(x^{\text{nom}})$, such bounds can be computed using interval arithmetic [Ehlers, 2017], symbolic intervals [Wang *et al.*, 2018] or tightened bounds [Dvijotham *et al.*, 2018b]. Noticing that $x^K = h^K(x^{K-1})$, we can bound the optimal value of (3) using a Lagrangian relaxation of the constraints:

$$\begin{aligned} \max_{x^0, \ldots, x^{K-1}} \quad & c^\mathsf{T} h^K(x^{K-1}) + d \\ & + \sum_{k=1}^{K-1} \left(\lambda^k\right)^\mathsf{T} \left(x^k - h^k(x^{k-1})\right) \\ \text{subject to} \quad & \underline{x}^k \leq x^k \leq \overline{x}^k \quad k = 0, \ldots, K-1 \end{aligned} \tag{4}$$

Note that any feasible solution of the original problem (3) is feasible for the above problem too, and for any such solution, the terms involving $\lambda^k$ become 0 (since the terms multiplying $\lambda^k$ are $\mathbf{0}$ for every feasible solution). Thus, for any choice of dual variables $\boldsymbol{\lambda} = \{\lambda^1, \ldots, \lambda^{K-1}\}$, the above optimization problem provides a valid upper bound on the optimal value of (3). This property is known as weak duality.




(a) (b)

Figure 2: Panels (a) shows three curves per model on MNIST where the dashed lines are the lower bounds computed from the LBFGS attack, the solid lines are our verified upper bound, and the dash-dot lines are the SDP verified upper bound from [Raghunathan *et al.*, 2018] (each color represents a different network). Panel (b) shows the verification of a robustly trained model on CIFAR-10 as a function of the perturbation radius $\epsilon$ (again solid lines are verified upper bounds computed by our method and dashed lines are the attack lower bounds).

Since the objective and constraints are separable in the layers, the variables in each layer can be optimized independently. The above problem becomes:

$$\underbrace{\sum_{k=0}^{K-1} \max_{\underline{x}^k \le x^k \le \overline{x}^k} \left\{ \left(\lambda^k\right)^\mathsf{T} x^k - \left(\lambda^{k+1}\right)^\mathsf{T} h^{k+1}(x^k) \right\} + d}_{\xi(\boldsymbol{\lambda}; x^{\text{nom}}, \boldsymbol{\theta})} \tag{5}$$

with $\lambda^0 = \mathbf{0}$ and $\lambda^K = -c$

where $\boldsymbol{\theta}$ represents the parameters of the underlying neural network (such as its weights and biases). Thankfully each individual maximization problem can be solved in closed-form for a wide range of transformations $h^k$.

**Affine layers.** For affine layers (e.g., fully connected layers, convolutions) that can be represented by $h^{k+1}(x^k) = Wx^k + b$, we have

$$\begin{aligned} &\max_{\underline{x}^k \le x^k \le \overline{x}^k} \left\{ \left(\lambda^k\right)^\mathsf{T} x^k - \left(\lambda^{k+1}\right)^\mathsf{T} (Wx^k + b) \right\} \\ &= \left[\lambda^k - W^\mathsf{T}\lambda^{k+1}\right]_+^\mathsf{T} \overline{x}^k + \left[\lambda^k - W^\mathsf{T}\lambda^{k+1}\right]_-^\mathsf{T} \underline{x}^k - b^\mathsf{T}\lambda^{k+1} \end{aligned} \tag{6}$$

where $[x]_+ = \max(x, 0)$ and $[x]_- = \min(x, 0)$ denote the positive and negative coordinates of $x$.

**Activation functions.** For component-wise non-linearity, each coordinate of $x^k$ can be optimized independently. For the i-th coordinate, we obtain:

$$\max_{\underline{x}_i^k \le x_i^k \le \overline{x}_i^k} \left\{ \lambda_i^k x_i^k - \lambda_i^{k+1} h^{k+1}(x_i^k) \right\}. \tag{7}$$

This is a one-dimensional optimization problem and can be solved easily – for common activation functions (ReLU, tanh, sigmoid, maxpool), it can even be solved analytically. [Dvijotham *et al.*, 2018b] provides additional details.

**Dual optimization problem.** Once these individual optimization problems are solved, we can construct the dual optimization problem:

$$\min_{\boldsymbol{\lambda}} \xi(\boldsymbol{\lambda}; x^{\text{nom}}, \boldsymbol{\theta}) \tag{8}$$

This seeks to choose the values of $\boldsymbol{\lambda}$ so as to minimize the upper bound on the verification objective, thereby obtaining the tightest bound. This optimization can be solved using a sub-gradient method on $\boldsymbol{\lambda}$. We note that (8) is a convex optimization problem.

## 4 Experiments

While the methodology presented is general, we now focus on measuring robustness to adversarial attacks. In the context of $\ell_\infty$ norm-bounded attacks of size $\epsilon$, we want to verify that for each class $y \neq y^{\text{true}}$

$$\begin{aligned} &(e_y - e_{y^{\text{true}}})^\mathsf{T} x^K \le 0 \\ &\forall x^0 \in \mathcal{S}^{\text{in}}(x^{\text{nom}}) = \{x \mid \|x - x^{\text{nom}}\|_\infty < \epsilon\} \end{aligned} \tag{9}$$

where $e_i$ is the standard $i^{\text{th}}$ basis vector. The output $x^K \in \mathbb{R}^N$ has $N$ logits corresponding to $N$ classes, and $y$ and $y^{\text{true}}$ are in $\{1, \ldots, N\}$.

We are interested in the *adversarial error rate*, which is the ratio of test examples for which there exists an attack. Computing this quantity precisely requires solving the NP-hard problem (3) for each test example and target class. However, we can obtain upper bounds on this value by using incomplete methods and lower bounds using a fixed attack algorithm[1]. We compare our approach with the SDP formulation from [Raghunathan *et al.*, 2018] (note that this approach only works for single hidden layer networks).

[1]We used an algorithm similar to [Carlini and Wagner, 2017b]

On MNIST, we train different models of various sizes with the sigmoid activation function.[2] Figure 2a shows that our approach is able to compute nearly tight bounds (bounds that match the upper bound) for small perturbation radii (up to 2 pixel units) and our bounds significantly outperform those from the SDP approach [Raghunathan *et al.*, 2018]. On CIFAR-10, we train a robust model with two hidden layers of 200 units each and ReLU activations using the adversarial training method from [Madry *et al.*, 2017]. Figure 2b shows that we were able to obtain the first non-trivial verification bounds on CIFAR-10 (to the best of our knowledge). While the model quality is rather poor, the results indicate that our approach can scale to more complicated models.

## 5 Learning to Verify

The formulation (8) can be exploited to train verifiable models [Dvijotham *et al.*, 2018a]. The most important properties that we use are that *(i) any* choice of $\boldsymbol{\lambda}$ provides a valid upper bound on (3), and *(ii)* if that upper bound is smaller than zero then that $\boldsymbol{\lambda}$ is a valid certificate to verify that the property holds. If the dual variables were required to satisfy constraints to be valid (this is the case with the formulation in [Kolter and Wong, 2017]), or if they needed to exactly optimize an objective to be a certificate (as in the case of exhaustive search methods), then a neural network, which is approximate by its very nature, would not be able to produce them. Further, the dual objective $\xi$ is (sub-)differentiable with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$, which allows backpropagation to be used to train neural networks to produce near-optimal dual variables.

**Predictor-verifier training.** The key idea is to exploit the fact that the dual optimization problem shares a lot of structure across training examples, so that the solution of the optimization problem $\boldsymbol{\lambda}^\star(x, \boldsymbol{\theta}) = \operatorname{argmin}_{\boldsymbol{\lambda}} \xi(\boldsymbol{\lambda}; x^{\text{nom}}, \boldsymbol{\theta})$ can be "learned", *i.e.*, a neural network can be trained to approximate the optimal solution $\boldsymbol{\lambda}^\star$ given $x, \boldsymbol{\theta}$. This alleviates the burden of solving the above optimization problem within each iteration in the training loop. This is done by using a verifier network to predict the dual variables given the input example. As the network trains, the verifier learns to produce dual variables that approximately minimize the upper bound and the predictor adjusts its weights so that the violation of the specification is minimized. The entire process is amenable to backpropagation and training with standard stochastic gradient algorithms (or variants thereof). Concretely, we train two networks simultaneously:

1. A predictor $P$, that takes as input the data to be classified $x^0$ and produces logits $x^K$ as output. The predictor is parameterized by $\boldsymbol{\theta}$.
2. A verifier $V$, that takes the activations $\boldsymbol{x} = \{x^0, \ldots, x^K\}$ produced by the predictor and the corresponding label $y^{\text{true}}$ as input and produces as output the dual variables $\boldsymbol{\lambda}$. The verifier is parameterized by $\boldsymbol{\rho}$.

[2] Results for more models and other activation functions are available in [Dvijotham *et al.*, 2018b].

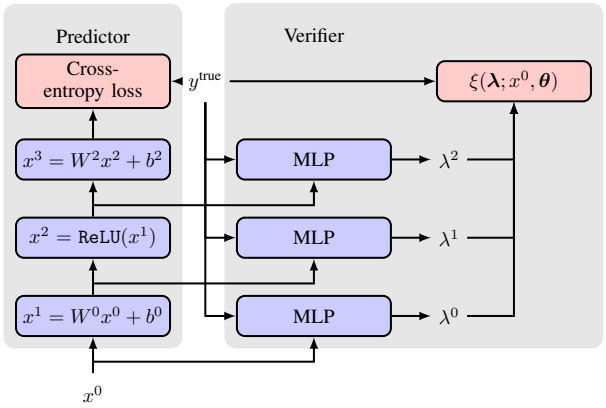



Figure 3: Example of predictor-verifier network architecture.

The training objective can be stated as follows:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\rho}} \mathbb{E}[\, \ell(P(x^0; \boldsymbol{\theta}), y^{\text{true}}) + \kappa \log(1 + e^{\xi(V(\boldsymbol{x}, y^{\text{true}}; \boldsymbol{\rho}); x^0, \boldsymbol{\theta})}]. \tag{10}$$

where $\ell$ is the supervised learning loss function (e.g., cross-entropy), and $\kappa$ is a hyperparameter that governs the relative weight of satisfying the specification versus fitting the data. Figure 3 show a possible predictor-verifier architecture.

**Results.** We compare predictor-verifier training (PVT) with interval bound propagation (which is equivalent to setting $\boldsymbol{\lambda}$ to zero) in the context of $\ell_\infty$ norm-bounded attacks of size $\epsilon$. On MNIST with $\epsilon = 0.1$, CIFAR-10 with $\epsilon = 0.03$ and SVHN with $\epsilon = 0.01$, PVT improves the verified error rate from 5% to 4.44%, 72.21% to 70.79% and 45.92% to 41.52%, respectively. We observe that the performance gap, even if significant, is surprisingly small. This phenomenon has been investigated further in [Gowal *et al.*, 2018], where we demonstrate that interval bound propagation with careful tuning can achieve state-of-the-art verified accuracy and can scale to even larger models. Currently, the jury is still out on whether training with tighter bounds is necessary for obtaining models with lower verified error rates. We believe that this is an important avenue for future research.

## 6 Conclusion

Today, the prevailing practice in machine learning is to train a system on a training data set, and then test it on another set. While this reveals the average-case performance of models, it is also crucial to ensure robustness, or acceptably high performance even in the worst case. Deployment of machine learning presents unique challenges, and requires the development of evaluation techniques that reliably detect unlikely failure modes. More broadly, we believe that learning consistency with specifications can provide large efficiency improvements over approaches where specifications only arise implicitly from training data. We are excited about ongoing research into adversarial evaluation, learning robust models, and verification of formal specifications.

## Acknowledgments

We would like to thank Chongli Qin, Brendan O'Donoghue, Rudy Bunel, Jonathan Uesato and Relja Arandjelovic.

## References

[Athalye and Sutskever, 2017] Anish Athalye and Ilya Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017.

[Athalye *et al.*, 2018] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

[Bunel *et al.*, 2017] Rudy Bunel, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Piecewise linear neural network verification: a comparative study. *arXiv preprint arXiv:1711.00455*, 2017.

[Carlini and Wagner, 2017a] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017.

[Carlini and Wagner, 2017b] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57. IEEE, 2017.

[Cheng *et al.*, 2017] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

[Dvijotham *et al.*, 2018a] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. Training verified learners with learned verifiers. *arXiv preprint arXiv:1805.10265*, 2018.

[Dvijotham *et al.*, 2018b] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. *arXiv preprint arXiv:1803.06567*, 2018.

[Ehlers, 2017] Ruediger Ehlers. Formal verification of piecewise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

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

[Gowal *et al.*, 2018] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[Kannan *et al.*, 2018] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[Katz *et al.*, 2017] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[Kolter and Wong, 2017] J Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

[Kurakin *et al.*, 2016] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[Madry *et al.*, 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[Mirman *et al.*, 2018] Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3578–3586, 2018.

[Papernot *et al.*, 2015] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

[Raghunathan *et al.*, 2018] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018.

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

[Tjeng *et al.*, 2017] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

[Uesato *et al.*, 2018] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

[Wang *et al.*, 2018] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. *arXiv preprint arXiv:1804.10829*, 2018.

[Weng *et al.*, 2018] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. *arXiv preprint arXiv:1804.09699*, 2018.