Exploring the Trade-offs to Train Robust Neural Models

Pedro Mendes
pgmendes@andrew.cmu.edu
Carnegie Mellon University
Institute for Software Research
Pittsburgh, USA

ABSTRACT

Adversarial training aims at generating and introducing adversarial examples while training a DNN model and it is one of the few defenses against adversarial attacks. Unfortunately, creating strong adversarial examples comes with an extremely large cost that makes adversarial training techniques impractical on large-scale and distributed models and problems. Therefore, in this survey, I first revisit the state-of-the-art in training robust models and adversarial training, second explore and investigate the different trade-offs found during training (e.g., standard accuracy, adversarial accuracy, training and inference time, time to generate perturbations, and the ratio of cleaned and perturbed inputs while training), and at last discuss how (or if) these techniques can be leveraged in real systems in production. This survey presents some of the results obtained to explore the trade-offs and conclude that although adversarial training is one of the best defenses against adversarial and targeted attacks, its high cost hinders its adoption by the industry.

1 INTRODUCTION

Machine Learning (ML) is now ubiquitous. A broad variety of applications exploits Artificial Intelligence (AI) and ML techniques as a part of their product, for example, to add new recommendation features or model the users' behaviors. Furthermore, ML models represent a vital component of many security or sensitive apps [10, 20, 26, 32]. Recent works [16, 31] have exposed several security concerns when exploiting Deep Neural Network models (DNN) that highlight the importance of developing new methods to cope with and obey security requirements. Although a vast spectrum of works focus on DNN, a wider range of modeling techniques also present similar vulnerabilities [7, 8]. More in detail, those works show how vulnerable several modeling techniques can be against adversarial attacks.

Adversarial attacks [16, 31] aim at misclassifying a model prediction by introducing a small perturbation in the input. In such attacks, malicious perturbations are determined based on the model and added to the clean data. Then, those are fed into the model with the goal to mislead it. On the other hand, targeted attacks [3, 30] are a similar type of attack but instead of only aiming at creating a misclassification, the attacker's goal is to target a certain class and yield a misclassification in one specific class.

There are several examples of sensitive applications that exploit ML techniques that also need to cope with these vulnerabilities and ensure that the model will not misclassify if the attacker introduces a small perturbation to the clean input. For example, autonomous driving cars leverage several AI and ML techniques (e.g., image recognition models) as a fundamental component of the interaction with the environment/world. Furthermore, a model's misclassification can have catastrophic consequences and, in the worst case, lead to mortal car accidents. Thus, these models should cope with possible adversarial examples that might surge in the real world. For example, Eykholt et al. [14] showed that the introduction of some stickers in a predefined position in a stop sign can mislead the model, and instead of being classified as a "stop sign", the model will predict a speed limit sign (see Figure 1).

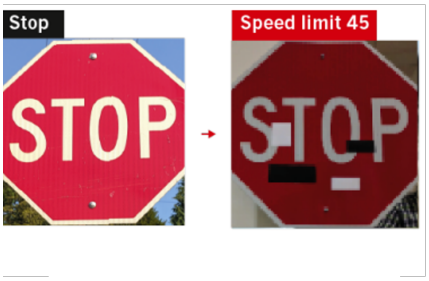

Figure 1: Example of an adversarial attack: adding perturbations in a Stop sign will mislead the model changing its prediction into a speed limit sign (from Eykholt et al. [14]).

Another example of applications vulnerable to adversarial or target attacks arises from the credit card fraud detection industries where ML is a core component [1, 6]. In this case, a misclassification can lead to large losses of capital and investments. These systems should be able to detect and cope with small changes in the input/feature space created by attackers and classified those as fraud. Furthermore, there are examples of adversarial attacks in several different domains like image recognition [16], text classification [12, 13], malware detection [22], and speech recognition [5].

In the recent years, several works [9, 16, 23, 24, 26] investigated and developed different techniques to mitigate these vulnerabilities by training robust models. Those works can be divided in four main areas: i) adversarial training [16, 23, 27, 34], ii) detection of adversarial attacks [4, 15, 18], iii) pre-processing techniques [2, 17, 29], and iv) verification and provable defenses [28, 33]. Moreover, these different methods take different assumptions about the model (e.g, white or black box model), the attacker capabilities, and optimization algorithms [9, 16, 23, 24, 26]. Despite all these different approaches, adversarial training is still the one that yields better defenses against adversarial and target attacks [21], and thus, in this survey, I will only focus on those methods. For example, the detection of adversarial attacks/examples is an extremely difficult task because the attacker can mislead two or more models in a single attack at the same time [4].

Although training robust ML models gained significant attention from the scientific community, it is still not commonly used in production mainly due to its high cost [21]. Moreover, the research in the area is mostly focused on robust DNN within the image recognition domain. Thus, in this survey, I will first revisit and dive into the literature on training robust ML models and adversarial training, second explore the different trade-offs presented in the literature (e.g., standard accuracy, adversarial accuracy, training and inference time, time to generate perturbations, and the ratio of cleaned and perturbed inputs while training), and at last, studied how (or if) these robust models can be really trained and deployed in production to ensure some level of security to a system.

AndrewID: pgmendes, April 2022, ISR, CMU, Pittsburgh, USA

2 ADVERSARIAL TRAINING

The goal of adversarial training is to train the model using adversarial examples generated and injected into the data while training. The perturbations are computed based on the current model and the clean inputs. Thus, instead of feeding the model directly with clean data and determining the new models' parameters θ by minimizing the loss function between the model's prediction $f_\theta(x)$ for the clean input x and the original class y, i.e.,

$$\min_\theta \{ \mathop{\mathbf{E}}_{x,y\sim D} [L(f_\theta(x), y)] \}, \tag{1}$$

adversarial training aims at determining adversarial examples and injecting them into the data used to train the model. In other words, adversarial training aims at finding a small perturbation δ (smaller than a maximum pre-defined bound ϵ) that will mislead the current model and train it with that adversarial input (the clean data input plus the perturbation), i.e.,

$$\min_\theta \{ \mathop{\mathbf{E}}_{x,y\sim D} [\max_{\|\delta\|<\epsilon} L(f_\theta(x+\delta), y)] \} \tag{2}$$

The robustness of the model trained adversarially will depend on the bound used to produce the adversarial examples and the strength of the method used to compute those examples.

There are three main strategies to train robust DNN models that are related to lower bounds, exact solutions, and upper bounds. Moreover, the most efficient methods in the literature rely on lower bounds to solve the optimization problem and generate adversarial examples (namely Fast Gradient Sign Method (FGSM) [16] and Project Gradient Descent (PGD) [23]). Both techniques solve first the inner maximization problem to find the perturbation and then the outer minimization problem to determine the models' parameters. However, these techniques require the computation of the gradient, which is not always possible.

2.1 Fast Gradient Sign Method

In order to solve the inner optimization problem, we can rely on Gradient Descent (or Ascent) methods. Moreover, for a given clean input x, this method adjusts the perturbation δ in the direction of its gradient. The gradient $g = \nabla_\delta L(f_\theta(x+\delta), y)$ can be easily computed using backpropagation technique. The perturbation δ is then adjusted according to $\delta = \delta + \alpha g$, for a given α, and then projected into the norm ball defined by $\|\delta\| \leq \epsilon$. However, if we select a value of α big enough and considering the projection into the ball, the value of g will not matter (only the sign of g, i.e., gradient direction) and the second term above equation can only take the values of $+\epsilon$ or $-\epsilon$. Thus, the perturbation can be easily obtained by computing $\delta = \epsilon \cdot \text{sign}(g)$.

FGSM was probably the first method to generate adversarial examples being proposed. However, the FGSM only allows the definition of L_∞ norms, and it was proven that stronger methods can mislead models trained adversarially using FGSM [21, 23, 34].

2.2 Projected Gradient Descent

The PGD method [23] does not make the assumption that α is large enough, and solves the inner maximization problem through the project gradient descent with a smaller step size and iteratively projects the result into the ball, i.e.,

$$\text{Repeat:} \quad \delta = \mathcal{P}(\delta + \alpha \nabla_\delta L(f_\theta(x+\delta), y)) \tag{3}$$

where $\mathcal{P}$ is the projection onto the ball of radius ϵ. PGD allows more choices of the hyper-parameters (in particular, the step size α and number of iterations) and different norms (e.g., L_1, L_2, L_∞ norms).

In spite of PGD presenting a significantly higher computational cost than the FGSM (see Section 3), it is considered one of the strongest attacks and defenses. Moreover, Madry et al. [23] show that PGD attacks can mislead robust models trained via FGSM and yield models with higher adversarial accuracy.

Hyper-parameter	Algorithm	Values
ϵ	FGSM, PGD	{0.01, 0.05, 0.1, 0.2}
α	PGD	$\{10^{-2}, 10^{-3}\}$
No. of Iterations	PGD	{5, 10, 20}
Ratio [%]	FGSM, PGD	{20, 40, 60, 80, 100}

Table 1: Hyper-parameters considered

There are several algorithms that extend both FGSM and PGD in order to improve their efficiency and performance. For example, Free Adversarial Training [27] considers a mini-batch of samples where it computes the perturbations via FGSM but using a non-zero initialization. In this method, the value of the previous perturbation in the mini-batch is used to initialize the perturbation of a new sample. On the other hand, Fast Adversarial Training [34] considers a non-zero random initialization and shows improvements in the adversarial accuracy of FGSM comparable to those achieved by PGD.

There are also other methods in the literature that are not so commonly used as for example, DeepFool [24] where the algorithm projects the clean input onto the closest decision hyperplane using a L_∞ norm, or Zero-th Order Optimization [9] that considers a black-box approach and just queries the real model without calculating the gradients and adapts the perturbation accordingly with the output. There is also another class of attacks and defenses developed particularly for robust decision trees, where the model is not differentiable. More in detail, Papernot et al. [25] present a method that reconstructs the path in the tree until a leaf node where the prediction is outputted, searches in the neighbor leaves to find a new path that will result in a different class/prediction, and change the input features accordingly to misclassify the model. It is also possible to solve a Mixed Integer Linear Programming optimization problem to determine small perturbations that mislead the model [19]. However, this last method presents a significant additional cost to solve the optimization problem.

All these methods to train robust models have their own trade-offs that I will explore in the next section. However, due to time and resource constraints, I will only consider the two most famous adversarial training techniques (i.e., FGSM and PGD).

Moreover, adversarial training introduces new hyper-parameters to the optimization problem. Thus, to train robust models, we need not only to perform the standard training to determine the models' parameters but also to compute the perturbations for the clean data. Thus, the difficulty of the problem increases considerably when compared with standard training with clean data. Furthermore, it is still under investigation in my current research if it is better and faster to train first the model only with clean data and then at the end inject adversarial examples or perform adversarial training from the beginning of the learning process, and if we need to jointly optimize the model's and the adversarial training hyper-parameters or if we can simplify the search space and consider two optimization problems. We aim at investigating those in the next steps of my research.

3 TRADE-OFFS OF ADVERSARIAL TRAINING

There are several different qualities that should be considered when training a robust model. In this section, we explore some of these trade-offs that we consider the most important while training a robust model using FGSM and PGD, namely i) standard accuracy, ii) adversarial accuracy, iii) training and iv) inference/testing time, and v) the ratio between clean and adversarial inputs while training.

To investigate these trade-offs, we consider a convolutional neural network with five layers that was trained using the MNIST [11] dataset. In our experiments, we used machines equipped with Intel Xeon Gold 6138 CPU, 64GB of memory, and two GPUs Nvidia GeForce GTX 1080. The training was performed by resorting to GPUs to speed up the experiments.

Furthermore, we consider several different bounds ϵ for the perturbation and different hyper-parameters of the adversarial training algorithms. The values considered are described in Table 1. To test the model's robustness, we

Algorithm	ϵ	Accuracy [%]	Adversarial Accuracy [%]	Training Time [s]	Inference Time [s]
Standard training	-	98.91	59.10	73.69	8.26
FGSM	0.01	99.11	60.48	102.56	8.51
	0.05	99.13	92.87	101.16	8.58
	0.1	99.09	95.73	103.97	8.57
	0.2	98.62	95.63	104.64	8.58
PGD	0.01	99.08	59.39	559.60	9.06
	0.05	99.17	93.75	556.47	9.08
	0.1	99.02	96.91	559.57	9.05
	0.2	97.60	94.79	540.71	8.79

Table 2: Results using standard and adversarial training with FGSM and PGD (20 iterations and $\alpha = 0.01$)

Algorithm	ϵ	Iterations	α	Accuracy [%]	Adversarial Accuracy [%]	Training Time [s]	Inference Time [s]
PGD	0.1	5	0.01	99.17	92.99	193.16	10.34
		10		99.16	95.34	346.16	9.57
		20		99.02	96.91	559.57	9.05
			0.001	99.37	79.94	570.95	9.71

Table 3: Results using adversarial training with PGD and varying the number f iterations and α

Algorithm	Ratio	Accuracy [%]	Adversarial Accuracy [%]	Training Time [s]	Inference Time [s]
FGSM	0.2	99.04	90.54	80.25	8.51
	0.4	99.20	92.56	83.08	8.58
	0.6	99.14	93.18	86.86	9.54
	0.8	99.17	94.13	97.54	9.02
	1	99.09	95.73	103.97	9.17
PGD	0.2	99.18	79.39	180.61	9.76
	0.4	99.01	93.75	283.47	9.18
	0.6	98.87	96.91	363.01	9.25
	0.8	99.09	95.73	468.17	9.37
	1	99.02	96.91	559.57	9.05

Table 4: Results using adversarial training with FGSM and PGD (20 iterations and $\alpha = 0.01$), fixing $\epsilon = 0.1$, and varying the ratio between clean vs. adversarial inputs

evaluate the adversarial accuracy. For that, we need to generate adversarial examples, attack the model, and measure the accuracy under attack. We resort to PGD since it is considered the strongest method in the literature, using a fixed bound ϵ of 0.1, 20 iterations, α equal to 0.01, and a ratio of 1 (i.e., for all the inputs in the testing set, we compute the perturbation and then attack the model).

Table 2 reports the main results obtained in this study using the metrics mentioned above. The model trained with clean data achieves a standard accuracy of 98.91% but only an adversarial accuracy of 59.1%. Although several works [16, 23, 27] highlight the fact that adversarial training yields a reduction in the final standard accuracy of the model, we only verified this fact when the bound ϵ is larger than 0.3 (these experiments were not included in the results reported in this survey). Moreover, the adversarial accuracy of a model trained only with clean data is significantly smaller, and the reduction in the standard accuracy when training robust models is just approximately 2%.

The training time increases on average by 7.5× and 1.4× when using PGD and FGSM compared to standard training, respectively. Thus, although PGD was proven to yield better robustness, there is an extremely high impact on the computational performance of this algorithm that cannot be neglected. On the other hand, the testing and inference times are not impacted since they were performed in the same conditions (and the model architecture is the same in all experiments). Furthermore, as expected the time necessary to generate a perturbation is just dependent on the technique used and the respective hyper-parameters. FGSM presents a constant time to generate each perturbation, while the same time in PGD depends mainly on two hyper-parameters (the number of iterations and α). However, the number of iterations presents the main contribution for this time and increases linearly. We also conclude that a higher number of iterations on PGD yields better robust models with higher adversarial accuracy (see Table 3).

At last, we evaluate the impact of varying the ratio between clean and adversarial inputs while training and report those results in Table 4. In FGSM, we verified an increase in training time by up to 1.3× when varying the ratio from 20% of adversarial inputs to 100%, while the standard accuracy remains practically the same and the adversarial accuracy increased by 5.4%. Using PGD, we obtained similar results for the standard and adversarial accuracy. Moreover, there is a linear increase in the training time with the ratio.

To sum up, although PGD is the method that yields better robust models with higher adversarial accuracy, there is a significant overhead while computing the perturbations with this method that harms the adoption of this method.

4 ADVERSARIAL TRAINING IN PRODUCTION

As we saw in the previous section, adversarial training comes with a significant cost that cannot be neglected. Moreover, the values of adversarial accuracy reported are based on the MNIST dataset using a small model and,

for example, the adversarial accuracy values reported by Kolter et al. [21] using the Cifar10 benchmark are significantly smaller than the obtained with MNIST.

Furthermore, other selection procedures before and while training (such as selecting the architecture, regularization, or hyper-parameter tuning) are not yet kept into attention while training robust models. All these decisions were done based on standard training, but there is no evidence that the optimal values obtained for standard training are still valid in the context of adversarial training.

It is still very costly to apply the techniques discussed in this work in large and distributed models which harms the adoption of these methods by industries and companies to be used in production. Thus, there is still a long path to go through to optimize the adversarial training procedures and be possible to use those in real systems in production.

5 CONCLUSION AND FUTURE WORK

In this survey, I presented an overview of the literature on adversarial training and investigate the different trade-offs that we face when training robust models (e.g., standard accuracy, adversarial accuracy, training and inference time, time to generate perturbations, and the ratio of cleaned and perturbed inputs while training).

We conclude that although adversarial training yields robust models and increases the level of security in a system, there is a huge impact on the training time that harms the adoption of these methods.

Thus, in the future work, we aim at developing optimization techniques to automatically tune all the hyper-parameters of adversarial training while exploiting cheap evaluations (e.g. by using cheaper methods to generate the adversarial examples, or reducing the training time, dataset size, or model size) to decrease the optimization cost while exploiting this new knowledge to extrapolate good quality configurations for adversarial training when using the entire datasets and more complex models.

REFERENCES

[1] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana S. C. Almeida, João Tiago Ascensão, and Pedro Bizarro. 2020. Interleaved Sequence RNNs for Fraud Detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

[2] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. https://openreview.net/pdf?id=S18Su--CW

[3] Prasanth Buddareddygari, Travis Zhang, Yezhou Yang, and Yi Ren. 2021. Targeted Attack on Deep RL-based Autonomous Driving with Learned Visual Patterns.

[4] Nicholas Carlini and David Wagner. 2017. *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods.*

[5] Nicholas Carlini and David Wagner. 2018. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. In *2018 IEEE Security and Privacy Workshops (SPW).*

[6] Francesco Cartella, Orlando Anunciação, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. 2021. Adversarial Attacks for Tabular Data: Application to Fraud Detection and Imbalanced Data. *ArXiv* abs/2101.08030 (2021).

[7] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. 2019. Robust Decision Trees Against Adversarial Examples. In *Proceedings of the 36th International Conference on Machine Learning*.

[8] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2019. Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach. In *Proceedings of the 7th International Conference on Learning Representations*.

[9] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. 2019. Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach. *ArXiv* abs/1807.04457 (2019).

[10] George E. Dahl, Jack W. Stokes, Li Deng, and Dong Yu. 2013. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*.

[11] Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research [Best of the Web]. In *IEEE Signal Processing Magazine*, Vol. 29. IEEE.

[12] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On Adversarial Examples for Character-Level Neural Machine Translation. In *Proceedings of the 27th International Conference on Computational Linguistics*.

[13] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

[14] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[15] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. 2017. Detecting Adversarial Samples from Artifacts. *ArXiv* abs/1703.00410 (2017).

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

[17] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2017. Countering Adversarial Images using Input Transformations. *ArXiv* abs/1711.00117 (2017).

[18] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. *arXiv* abs/1702.04267 (2017).

[19] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. 2016. Evasion and Hardening of Tree Ensemble Classifiers. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*.

[20] Eric Knorr. 2015. How PayPal beats the bad guys with machine learning. https://www.infoworld.com/article/2907877/how-paypal-reduces-fraud-with-machine-learning.html

[21] Zico Kolter and Aleksander Madry. 2018. Adversarial Robustness - Theory and Practice. https://adversarial-ml-tutorial.org/

[22] Keane Lucas, Mahmood Sharif, Lujo Bauer, Michael K. Reiter, and Saurabh Shintre. 2021. Malware Makeover: Breaking ML-Based Static Analysis by Modifying Executable Bytes. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*.

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations*.

[24] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2574–2582.

[25] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*.

[26] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*. United States, 582–597.

[27] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free!. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*.

[28] Aman Sinha, Hongseok Namkoong, and John C. Duchi. 2018. Certifying Some Distributional Robustness with Principled Adversarial Training. In *Proceedings of the 6th International Conference on Learning Representations*.

[29] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2018. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. *ArXiv* abs/1710.10766 (2018).

[30] Jacob M. Springer, Melanie Mitchell, and Garrett T. Kenyon. 2021. A Little Robustness Goes a Long Way: Leveraging Robust Features for Targeted Transfer Attacks. In *Advances in Neural Information Processing Systems*.

[31] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[32] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness May Be at Odds with Accuracy. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SyxAb30cY7

[33] Eric Wong and J. Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the 35th International Conference on Machine Learning*.

[34] Eric Wong, Lesli Rice, and Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. In *ICLR*.