# The Relationship between Neural Network (NN) Architecture and Robustness–Towards the Development of both Robust and Accurate NNs

*Team Grey Matter*

Alex Bai
Applied Mathematics and Statistics
Johns Hopkins University

Patrick Herbert
Applied Mathematics and Statistics
Johns Hopkins University

Annie Liang
Biomedical Engineering
Johns Hopkins University

Steven Witkin
Applied Mathematics and Statistics
Johns Hopkins University

Joseph Yu
Biomedical Engineering
Johns Hopkins University

May 1, 2020

**Abstract**

While trained neural networks (NNs) have been successful in image classification tasks in computer vision systems, they have also been shown to be vulnerable to adversarial examples. These examples, generated by slight image perturbations sometimes so insignificant that they are imperceptible to a human, can result in misclassification by the NNs. To address this, adversarial training–whereby these adversarial examples are integrated into NN training–has been developed to reduce the susceptibility of NNs to adversarial attacks. While adversarially trained NNs do appear to learn improved representations more aligned with human perception, it comes at the apparent cost of accuracy. This trade-off and its relationship to NN architecture remains unexplored. We explore this in an adversarially-trained NN and implement mixed integer programming (MIP) to identify non-critical neurons with respect to accuracy and robustness. Specifically, we iteratively prune non-critical neurons and adversarially train the reduced NN classifiers on the original data set. Subsequently, we use MIPs to quantify the robustness of both the reduced and original NN classifiers and determine the relationship between the reduced and original NN classifier to robustness.

## 1 Introduction

While neural networks (NNs) have demonstrated immense promise in various challenging pattern recognition tasks (Russakovsky et al. 2014), they are often trained and compared against one another with respect to testing and training accuracy. This has given rise to brittle classifiers that are sensitive to the uncertainty and variability of tomorrow's data as well as constructed adversarial attacks (Evtimov et al. 2017, Carlini and Wagner 2017, Carlini and Wagner 2016). These crafted examples generated by slight image perturbations such as pixel noise (Carlini and Wagner 2016), rotations (Engstrom et al. 2019), and translations (Engstrom et al. 2019) can result in misclassification despite being recognizable by a human.

To address this, adversarial training–whereby these adversarial examples are integrated into NN training–has been developed to reduce the susceptibility of NNs to adversarial attacks (Kurakin, Goodfellow, and Bengio 2016; Madry et al. 2017; Tsipras et al. 2018; Khalil, Gupta, and Dilkina 2018; Ilyas et al. 2019). Additionally, various methods of adversarial attacks have been developed to both certify robustness bounds for a trained classifier as well as improve our theoretical understanding of DNNs. These attacks typically use one of two approaches: craft adversarial attacks to demonstrate an upper bound on robustness (i.e., heuristic) or attempt to prove lower bound (i.e., complete). Additionally, adversarial attacks can be classified as either white (Kurakin, Goodfellow, and Bengio 2016, Evtimov et al. 2017) or black box (Carlini and Wagner 2016, Carlini and Wagner 2017); in the former, adversarial examples are crafted with the NN architecture and parameters known but unknown in the latter.

Unlike heuristic approaches such as projected gradient descent (Madry et al. 2017), mixed-integer programming (MIP) approaches have been developed to more exactly quantify robustness and determine lower accuracy bounds of certain types of adversarial attacks (Tjeng and Tedrake 2017; Fischetti and Jo 2017). This has been accomplished using the following robustness metrics: determining the minimum adversarial distortion, or the minimum distance to the closest adversarial example for a specific test input (Carlini and Wagner 2017), or (2) computing the adversarial test accuracy, which is the fraction of the test set for which no adversarial examples can be identified given a level $\epsilon$ perturbation (Bastani et al. 2016). Such metrics, in the context of possible real-world adversarial attacks, provide the necessary framework for complete robustness verification of trained NNs critical for the deployment of computer vision systems to the real world.

The relationship between NN architecture and robustness remains underexplored. To-date, NN architecture design remains mostly manual as practical theory relating NN architecture and their capacity/expressiveness with respect to joint distribution of the data and its corresponding label in the pattern recognition task remains incomplete. Previous research has been limited to the exploration of NN architecture and accuracy to various pattern recognition tasks (ElAraby, Wolf, and Carvalho 2020). In initial computer vision work by Kurakin, Goodfellow, and Bengio 2016, robustness appeared to be positively correlated to the number of convolutions for a specific base NN model. Subsequently, Tsipras et al. 2018 showed that robustness appears to come at the cost of accuracy despite learning representations more aligned with human perception. Both of these studies provided broad initial results regarding NN architecture and robustness but leave several important questions unexplored regarding the exact relationship between NN architecture with respect to classifier robustness and accuracy.

### 1.1 Our contributions

We seek to answer the following questions:

1. Which set of nodes in a given NN architecture are critical for accuracy? Which set of nodes are responsible for robustness?

2. Do they remain consistent before and after NN reduction via pruning?

3. Following adversarial training, is robustness of the reduced NN altered? If so, what is the relationship between the adversarial examples generated for original and reduced NNs? Are adversarial attacks transferable between original and reduced NNs? Can this be formalized into a notion of NN expressiveness/capacity?

4. How are the accuracy and robustness bounds of NNs related to its architecture? Having established such a relationship, can one identify the proper NN architecture expansion using an optimization formulation?

## 2 Related Work

General observations regarding robustness and NN architecture have been published by Madry et al. 2017 and Kurakin, Goodfellow, and Bengio 2016. Both observed that the number of convolutions correlates positively with robustness. However, a more detailed quantification and exploration remain unexplored. We refer the reader to Appendix F of Tsipras et al. 2018 and Section 6 of Madry et al. 2017 for a broader reference of related work in NN robustness in the field of computer vision.

Mixed integer linear programming (MILP) has been recently implemented as a method for complete verification of robustness. However, efficient solving still remains difficult given a $\epsilon$ perturbation threshold and NN architecture complexity. Long solving times are often truncated so that verification is incomplete. To address this, most MILP solvers only consider rectified linear (ReLU) NNs; see Bunel et al. 2017 for a comprehensive review regarding ReLU NN robustness verification. We refer the reader to Section 2 of Tjeng and Tedrake 2017 for additional details regarding MILP in robustness verification on a broader class of NN using nonlinear activation functions.

## 3 Experiments

Working off an existing NN architecture (ElAraby, Wolf, and Carvalho 2020), we implemented the ElAraby, Wolf, and Carvalho 2020 MILP pruning algorithm to identify non-important nodes. Following training, we verified robustness of both the original and pruned models using the MILP model described below.

### 3.1 Formulating the MILP model

To generate adversarial examples and evaluate the robustness of our NNs, we the network was exactly modeled using a Mixed-Integer Linear Program (MILP) developed by Fischetti and Jo 2017 and implemented in Python by Kolter and Madry 2019.

$$\underset{z_{1,\ldots,d+1},v_{1,\ldots,d-1}}{minimize} \quad (e_y - e_{y_{\text{targ}}})^T z_{d+1} \tag{1}$$

$$z_{i+1} \geq W_i z_i + b_i, \;\; i = 1 \ldots, d-1 \tag{2}$$

$$z_{i+1} \geq 0, \;\; i = 1 \ldots, d-1 \tag{3}$$

$$u_i \cdot v_i \geq z_{i+1}, \;\; i = 1 \ldots, d-1 \tag{4}$$

$$W_i z_i + b_i \geq z_{i+1} + (1 - v_i) l_i, \;\; i = 1 \ldots, d-1 \tag{5}$$

$$z_{d+1} = W_d z_d + b_d \tag{6}$$

$$v_i \in \{0,1\}^{|v_i|}, \;\; i = 1 \ldots, d-1 \tag{7}$$

$$z_1 \leq x + \delta \tag{8}$$

$$z_1 \geq x - \delta \tag{9}$$

This MILP formulation emulates a NN with $d$ ReLU layers. In it, $W_i$ and $b_i$ are respectively the weights and biases of layer $i$; these are constant because the network is already trained. The variables $z_i$ are continuous, and represent the values of the neurons at layer $i$. $v_i$ are binary variables, and control the ReLU activation functions at each layer. Taken together, constraints (2-7) of the above MILP represent the calculations made by the neural network for input image $z_1$, accounting for the ReLU activation functions at each neuron and propagation of values throughout the model, resulting in variable $z_{d+1}$. Where the MILP formulation differs in capability from the original NN, and the reason for introducing a minimization, is that this MILP is set up to take as an input a vector $x$, which is in our case a MNIST image, and create a new vector (image) $z_1$ that within the context of the NN minimizes an objective function. The objective function shown above in (1) is a general form which places a positive weight of 1 on the $y^{th}$ element of the output vector $z_{d+1}$ and a negative weight of 1 on a target element of the output vector. So, this objective function serves to maximize the output of the NN for the target classification, while minimizing the current label, thus aiming to make an input vector which is more likely to be classified by the model as the target. The objective function also has the property that if the objective function value at the optimal solution is negative, then the MILP has found an input vector that will be classified by the model as the target instead of the original (the magnitude of the output index for the target is greater than the output index for the original classification). Thus, an adversarial example has been found. Additionally, given an image $x$, the MILP allows us to set a parameter $\delta$, which in constraints (8-9) limit the range of values that the model can choose for the input vector $z_1$. So we are able to limit the $l_\infty$ norm of the adversarial attack generated by the MILP, so that for small $\delta$ values, the adversarial example will still look like the original image $x$.

### 3.2 Pruning of DNN

We use the pruning algorithm developed by ElAraby, Wolf, and Carvalho 2020 to obtain our pruned NN. For each neuron, an importance score ($s_i$) is computed based on an MILP formulation. The score is added onto constraint (2) from the Fischetti MIP above to give constraint (10).

$$z_{i+1} \geq W_i z_i + b_i - (1 - s_i) u_i \tag{10}$$

Each layer is given a sum of importance scores $I_i = \sum_{j \in L_i}(s_{i,j} - 2)$ which is ultimately minimized in the objective. Neurons below a threshold value of 0.001 are removed from the NN thus generating the pruned version. Subsequently, the pruned NN was trained on the same training MNIST images as the original NN.

### 3.3 Performance comparison

As a first pass, NN robustness was quantified in several ways including (1) the number of adversarial examples generated, (2) the likelihood of obtaining an adversarial example, and (3) the objective function value (OVF).

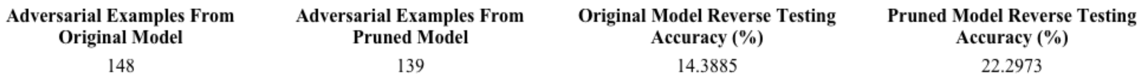
### 3.4 Data sets

Initial experiments were run using the publically available MNIST data set.

## 4 Computational results

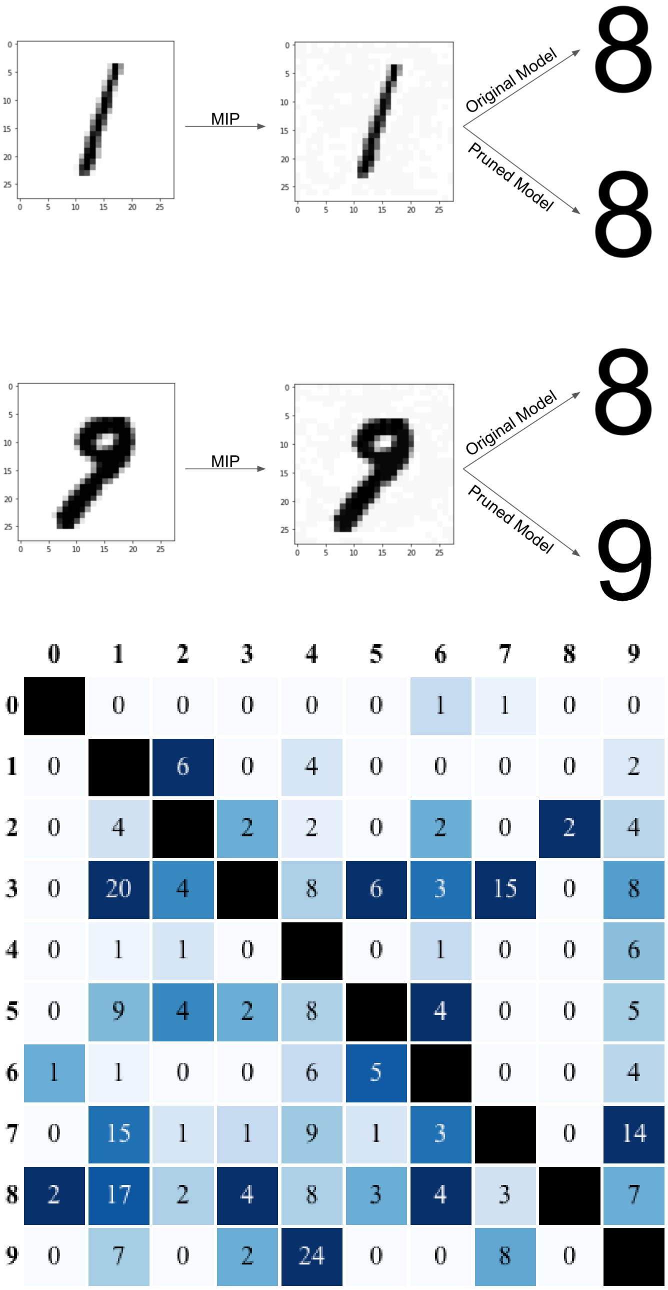
The accuracy of the trained original and pruned NNs were 97.05% and 97.11%, respectively. The elevated accuracy of the pruned model could have arisen because of (1) overfitting by the original NN or (2) the consequence of adversarial training improving the learned representation for a smaller number of training samples (Tsipras et al. 2018).

When NNs were tested for robustness, the pruned NN surprisingly appeared to be more robust (Figure 2). The adversarial MILP found fewer adversarial examples (139 vs. 148). Moreover, the pruned NN had higher accuracy when tested with adversarial examples generated from the original NN (22.3% vs 14.4%). We believe additional experimentation is required to the exact cause of this and it's exact relationship compared to previous published work (Tsipras et al. 2018, Madry et al. 2017, Kurakin, Goodfellow, and Bengio 2016). When we looked at the adversarial examples, they remained distinctly recognizable (Figure 2a) with only the addition of background pixel noise. Moreover, more adversarial examples could be generated for certain image labels and for specific target image labels (Figure 2b). For example, many adversarial examples of {3, 7, 8} could be found for image label {1}. These results are further summarized in Figure 3. Lastly, MILP performance statistics are shown in Figure 4.



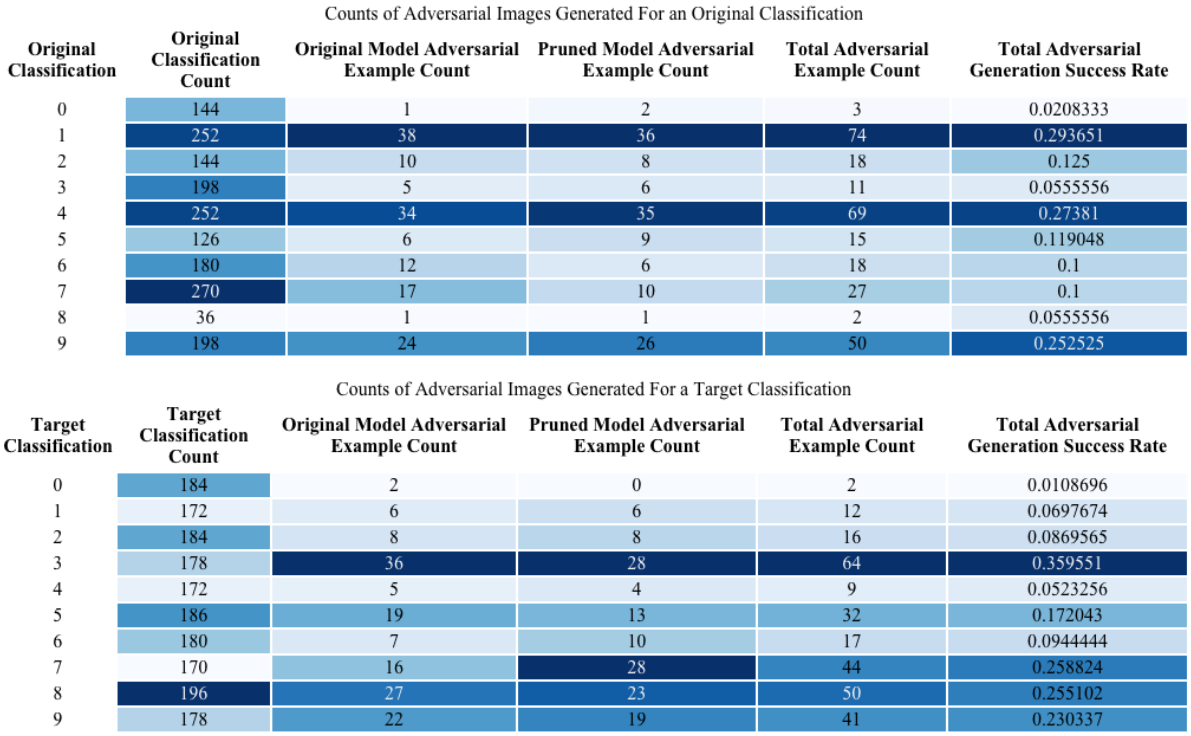
| Adversarial Examples From Original Model | Adversarial Examples From Pruned Model | Original Model Reverse Testing Accuracy (%) | Pruned Model Reverse Testing Accuracy (%) |
|---|---|---|---|
| 148 | 139 | 14.3885 | 22.2973 |

Figure 1: Number of adversarial examples generated for each NN along with (left two columns) and NN accuracy when tested upon the other model (i.e., original on pruned examples vs pruned on original examples).



|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** |  | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **1** | 0 |  | 6 | 0 | 4 | 0 | 0 | 0 | 0 | 2 |
| **2** | 0 | 4 |  | 2 | 2 | 0 | 2 | 0 | 2 | 4 |
| **3** | 0 | 20 | 4 |  | 8 | 6 | 3 | 15 | 0 | 8 |
| **4** | 0 | 1 | 1 | 0 |  | 0 | 1 | 0 | 0 | 6 |
| **5** | 0 | 9 | 4 | 2 | 8 |  | 4 | 0 | 0 | 5 |
| **6** | 1 | 1 | 0 | 0 | 6 | 5 |  | 0 | 0 | 4 |
| **7** | 0 | 15 | 1 | 1 | 9 | 1 | 3 |  | 0 | 14 |
| **8** | 2 | 17 | 2 | 4 | 8 | 3 | 4 | 3 |  | 7 |
| **9** | 0 | 7 | 0 | 2 | 24 | 0 | 0 | 8 | 0 |  |

Figure 2: Adversarial example details. (a) Representative images of generated adversarial examples. (b) Confusion matrix showing the number of adversarial examples (a particular cell) for each image label (column) given a particular target label (row)



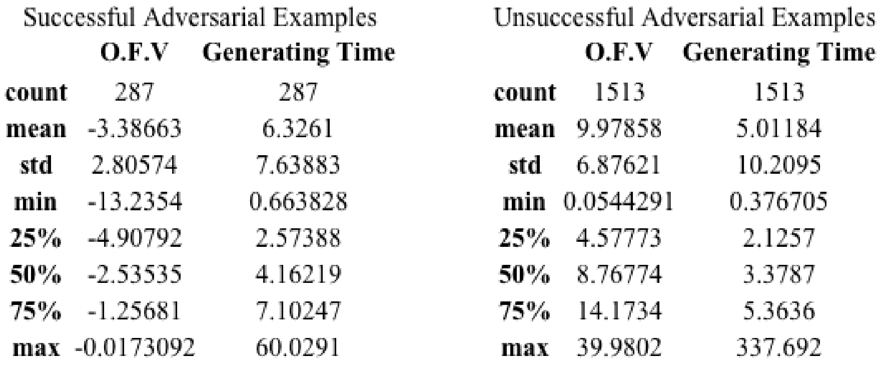
Counts of Adversarial Images Generated For an Original Classification

| Original Classification | Original Classification Count | Original Model Adversarial Example Count | Pruned Model Adversarial Example Count | Total Adversarial Example Count | Total Adversarial Generation Success Rate |
|---|---|---|---|---|---|
| 0 | 144 | 1 | 2 | 3 | 0.0208333 |
| 1 | 252 | 38 | 36 | 74 | 0.293651 |
| 2 | 144 | 10 | 8 | 18 | 0.125 |
| 3 | 198 | 5 | 6 | 11 | 0.0555556 |
| 4 | 252 | 34 | 35 | 69 | 0.27381 |
| 5 | 126 | 6 | 9 | 15 | 0.119048 |
| 6 | 180 | 12 | 6 | 18 | 0.1 |
| 7 | 270 | 17 | 10 | 27 | 0.1 |
| 8 | 36 | 1 | 1 | 2 | 0.0555556 |
| 9 | 198 | 24 | 26 | 50 | 0.252525 |

Counts of Adversarial Images Generated For a Target Classification

| Target Classification | Target Classification Count | Original Model Adversarial Example Count | Pruned Model Adversarial Example Count | Total Adversarial Example Count | Total Adversarial Generation Success Rate |
|---|---|---|---|---|---|
| 0 | 184 | 2 | 0 | 2 | 0.0108696 |
| 1 | 172 | 6 | 6 | 12 | 0.0697674 |
| 2 | 184 | 8 | 8 | 16 | 0.0869565 |
| 3 | 178 | 36 | 28 | 64 | 0.359551 |
| 4 | 172 | 5 | 4 | 9 | 0.0523256 |
| 5 | 186 | 19 | 13 | 32 | 0.172043 |
| 6 | 180 | 7 | 10 | 17 | 0.0944444 |
| 7 | 170 | 16 | 28 | 44 | 0.258824 |
| 8 | 196 | 27 | 23 | 50 | 0.255102 |
| 9 | 178 | 22 | 19 | 41 | 0.230337 |

Figure 3: Additional details about adversarial examples shown in Fig 2. Details are shown for original classification label (top) and target adversarial classification label (bottom).



Successful Adversarial Examples

| | O.F.V | Generating Time |
|---|---|---|
| **count** | 287 | 287 |
| **mean** | -3.38663 | 6.3261 |
| **std** | 2.80574 | 7.63883 |
| **min** | -13.2354 | 0.663828 |
| **25%** | -4.90792 | 2.57388 |
| **50%** | -2.53535 | 4.16219 |
| **75%** | -1.25681 | 7.10247 |
| **max** | -0.0173092 | 60.0291 |

Unsuccessful Adversarial Examples

| | O.F.V | Generating Time |
|---|---|---|
| **count** | 1513 | 1513 |
| **mean** | 9.97858 | 5.01184 |
| **std** | 6.87621 | 10.2095 |
| **min** | 0.0544291 | 0.376705 |
| **25%** | 4.57773 | 2.1257 |
| **50%** | 8.76774 | 3.3787 |
| **75%** | 14.1734 | 5.3636 |
| **max** | 39.9802 | 337.692 |

Figure 4: MILP algorithm performance statistics for successful (left) and unsuccessful (right) adversarial examples; objective function value (OFV) and generation time are shown for each. This value represents the difference in a model's confidence that a given example belongs to the Original Classification or the Target Classification. A negative OFV indicates an instance of successful adversarial example generation, as it means the model is more confident that the example belongs to the Target Classification than the Original Classification.

## 5 Future Work

In this class project, we started to explore the relationship between NN architecture and robustness in the context of computer vision with the broader goal of identifying the appropriate NN architecture that can maximize both robustness and accuracy for a particular classification task. To accomplish this, we hope to develop a MILP approach that will enable us to expand the NN architecture in a principled approach before or after pruning. In the near term, we hope to run future experiments exploring alternative MIP pruning formulations that identify specific NN architectural components important for robustness (i.e., are there additional factors to consider aside from the number of hidden layers?). In addition to this, we hope to also explore various lower dimensional embeddings of the data (learning-task dependent) that could be more robust across the broader class of adversarial perturbations. While there are guarantees of an existence of distribution with which your classifier will have poor performance (Devroye, Györfi, and Lugosi 1996, Ch. 1, p. 5). Can one flip this guarantee – good performance for a class of distribution that can represent human visual perception?

## References

[1] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

[2] Ivan Evtimov et al. "Robust Physical-World Attacks on Machine Learning Models". In: *CoRR* abs/1707.08945 (2017). arXiv: 1707.08945. URL: http://arxiv.org/abs/1707.08945.

[3] Nicholas Carlini and David A. Wagner. "Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods". In: *CoRR* abs/1705.07263 (2017). arXiv: 1705.07263. URL: http://arxiv.org/abs/1705.07263.

[4] Nicholas Carlini and David A. Wagner. "Towards Evaluating the Robustness of Neural Networks". In: *CoRR* abs/1608.04644 (2016). arXiv: 1608.04644. URL: http://arxiv.org/abs/1608.04644.

[5] Logan Engstrom et al. "Exploring the Landscape of Spatial Robustness". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, Sept. 2019, pp. 1802–1811. URL: http://proceedings.mlr.press/v97/engstrom19a.html.

[6] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. "Adversarial Machine Learning at Scale". In: *CoRR* abs/1611.01236 (2016). arXiv: 1611.01236. URL: http://arxiv.org/abs/1611.01236.

[7] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2017. arXiv: 1706.06083 [stat.ML].

[8] Dimitris Tsipras et al. *Robustness May Be at Odds with Accuracy*. 2018. arXiv: 1805.12152 [stat.ML].

[9] Elias B. Khalil, Amrita Gupta, and Bistra Dilkina. “Combinatorial Attacks on Binarized Neural Networks”. In: *CoRR* abs/1810.03538 (2018). arXiv: 1810.03538. URL: http://arxiv.org/abs/1810.03538.

[10] Andrew Ilyas et al. *Adversarial Examples Are Not Bugs, They Are Features.* 2019. arXiv: 1905.02175 [stat.ML].

[11] Vincent Tjeng and Russ Tedrake. “Verifying Neural Networks with Mixed Integer Programming”. In: *CoRR* abs/1711.07356 (2017). arXiv: 1711.07356. URL: http://arxiv.org/abs/1711.07356.

[12] Matteo Fischetti and Jason Jo. “Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study”. In: *CoRR* abs/1712.06174 (2017). arXiv: 1712.06174. URL: http://arxiv.org/abs/1712.06174.

[13] Osbert Bastani et al. “Measuring Neural Net Robustness with Constraints”. In: *CoRR* abs/1605.07262 (2016). arXiv: 1605.07262. URL: http://arxiv.org/abs/1605.07262.

[14] Mostafa ElAraby, Guy Wolf, and Margarida Carvalho. *Identifying Critical Neurons in ANN Architectures using Mixed Integer Programming.* 2020. arXiv: 2002.07259 [cs.LG].

[15] Rudy Bunel et al. “Piecewise Linear Neural Network verification: A comparative study”. In: *CoRR* abs/1711.00455 (2017). arXiv: 1711.00455. URL: http://arxiv.org/abs/1711.00455.

[16] Zico Kolter and Aleksander Madry. *Adversarial Robustness - Theory and Practice.* 2019. URL: https://adversarial-ml-tutorial.org/.

[17] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition.* Vol. 31. Stochastic Modelling and Applied Probability. Springer, 1996, pp. 1–638. ISBN: 978-1-4612-0711-5.