Adversarial Attacks and Defenses on Neural Networks for Image Classification

Jafar Badour
Innopolis University
Innopolis, Russia
j.badour@innopolis.ru

Adil Khan
Innopolis University
Innopolis, Russia
a.khan@innopolis.ru

1 INTRODUCTION

1.1 What is an adversarial attack?

An adversarial attack is a procedure to fool DNNs into producing false classification to an adversarial example. The "adversarial example" is a modified sample that -before manipulations- would be classified correctly. Manipulating its parameters would invoke the DNN to misclassify it. The adversarial example to the naked eye-/ear would look benign. It is hard to detect the difference between adversarial and benign examples. An example of an adversarial and benign input is captured in figure 1.

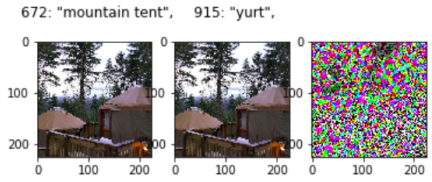

Figure 1: Example of benign (left), adversarial (middle) and magnified difference (right)

1.2 Adversarial attacks types

There are multiple types of adversarial attacks. Attacks are grouped by three main classes:

- White box: White box attacks exploit the inner structure and known weights of the target model. To build a white box attack one should have the weights and structure of the model.
- Score Based Black Box: Score Based Black Box attacks do not require full access to the weights and structure of a model. Instead, they require to make constant queries to the target model. These queries should return the score for each provided input.
- Limited Black Box: These attacks do not require weights nor scores of the inputs. They just require the class of which the target model predicted.

1.3 Adversarial attacks impact on input spaces

1.3.1 Input Space. Since we are working with images. Input space of an image with $(512, 256)$ dimensions and 1 color channel will be massive. It will have $512.256 = 131072$ dimensions. Figure 2 is a simplification of the input space. It is crucial to mention that the vast majority of regions of the input space are noise areas.

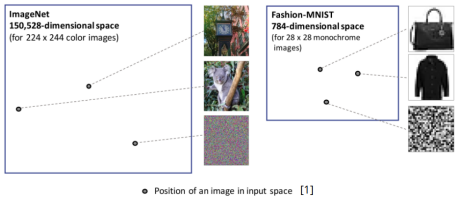

Figure 2: Input space

1.3.2 Adversary's Goal. The adversary's goal is to exploit the classification borders between classes. Its main purpose is to shift the benign example from point $v_b \in B$ to $v_a \notin B$. Initially we want to cause a mis-classification. However, It is quite feasible to cause the model to predict a predefined goal class. Formally: moving a point $v_b \in B$ to $v_a \in A$. It is also in the adversary's best interest to keep perturbations small. $p = v_b - v_a$, the adversary also wants to minimize $l_1, l_2, ..., l_\infty$ norms of p depending on the attack.

1.4 Perturbation vs Patch attacks

Adversarial attacks are carried as perturbation or patch attacks. In this report, I will focus on perturbation white-box attacks. However, the concepts are similar regardless of the type or way of attack. Patch attacks are imperative if the attacker cannot input the adversarial example in its digital form. For example, if we have a face detection system deployed at a police station. The attacker cannot access the inner system, therefore a patch is required. A patch is a physical world gadget that contains some perturbed shapes. The patch would fool the system using the same concepts as the perturbation attack, to shift the benign example vector in the input space outside of its classification region.

Figure 3 captures a patch attack while figure 1 represents a perturbation attack. The question remains: "Why can't we use perturbation attacks in the real-world by printing an image of the adversarial example?". A short answer would be: the resolution of the camera is not taken into account when building the perturbation. The image taken by a camera undergoes multiple pre-processing stages. For instance, resizing a perturbed image would drastically impact the adversarial example.

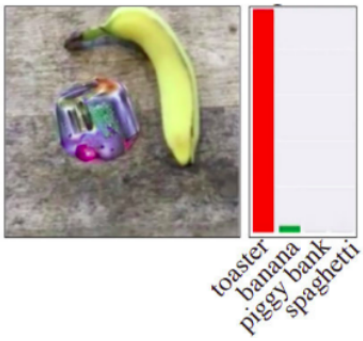

Figure 3: Patch attack: causing a mis-classification: from banana to toaster

1.5 Salient Features

A salient feature is (from image input space perspective) is a pixel that contributes to the classification greatly. For example, ellipsoids would be salient in a face detection model. Because they look like human eyes. In figure 4 one can see how the model relies on salient features to make a classification.

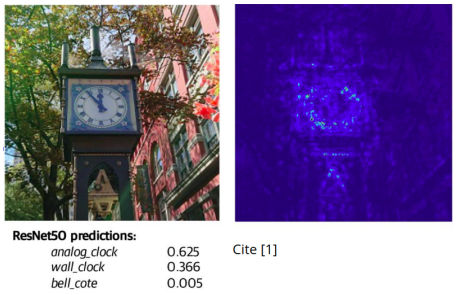

Figure 4: Salient Features of a clock

1.6 DNNs mathematical vulnerability

We know that the DNN model is a mathematical function $y = f(x)$ where x is the input (an image in our case) and the output of the function is the predicted class y. The function consists of many parameters such as weights and biases of the network. These weights and biases transform a huge-dimension input x to a uni-dimension output which is the class type. Formally:

$$y = f(x, \theta)$$

Where $theta$ represents all the weights and biases of the network. The cost function can be expressed as follows:

$$C(f(x, \theta), L)$$

Where $f(x, \theta)$ is the prediction that the network made in response to the training example, and L is the correct label. These loss functions are used to transform the classification problem into an optimization problem for this loss function.

To solve this optimization problem; One exploits the gradients that the loss function produces. The gradients give insight about the effects that a change dx_i in pixel i would impact the loss function, therefore we can, for instance, set all dx_i to some values in a special way. The pixel which impacts the loss function with positive gradient to $-\alpha$ and the others with $+\alpha$.

$$x' = DXx$$

now we can deduce that $C(f(x', \theta) < C(f(x, \theta)$. With multiple iterations we will get to a local minima.

The adversary uses the same concept of gradients. However, not to minimize the loss function, but to increase it, and cause a misclassification subject to some to constrain of l_z norm of the perturbation.

2 ATTACKED MODEL

I have experimented with two models. These models' weights are trained as a part of denoising autoencoder. I will explain why I chose DAE in the defenses section. The encoder of both models is retrieved from the encoder of the DAE.

- Model 1: DAE encoder and CNN.
- Model 2: DAE encoder, decoder and CNN.

2.1 DAE structure

The denoising-autoencoder structure is highlighted in figure 5. Multiple convolution layers and max-pooling and then deconvolution layers and upsampling.

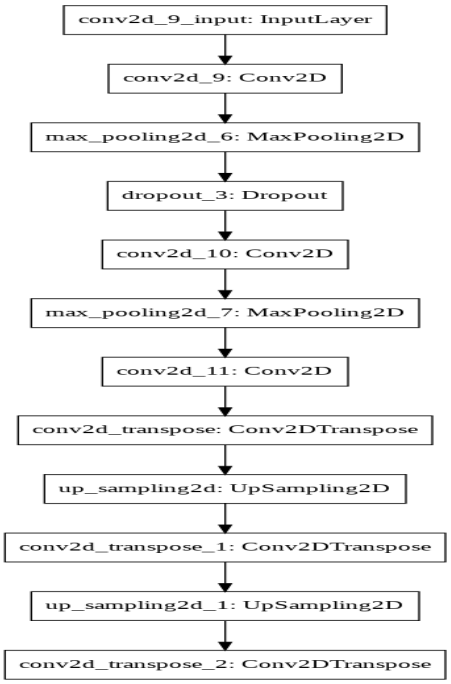

Figure 5: DAE structure

2.2 Model 1 structure

I transferred the weights of the DAE's to be as an input to a CNN the structure is described in figure 6.

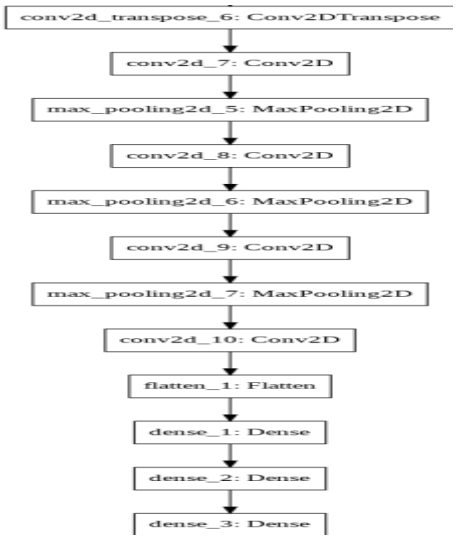

Figure 6: Model 1 structure, first part was omitted out because of similarity to DAE layers and weights

2.3 Model 2 structure

Transferred only the DAE's encoder layers and then plugged in a CNN.

2.4 DAE performance

We can check DAE loss versus epochs in figure 7. We can deduce that DAE did not experience overfitting going through 10 epochs.

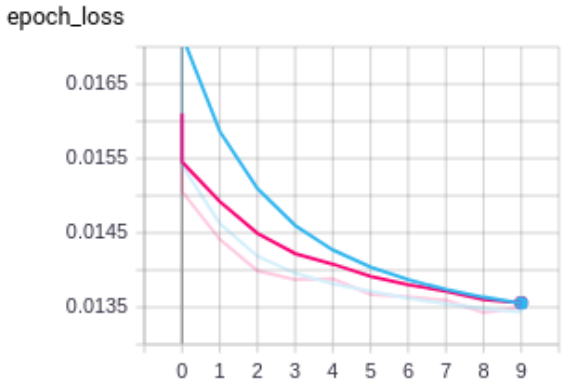

Figure 7: Model 1 loss function (MAE) values against epochs. We can see that overfitting is non-present. Blue on training, Red on validation.

3 ATTACKS IMPLEMENTED

Three types of attacks were required at the beginning of the work. Which are:

- Fast Gradient Sign Method (FGSM)
- Jacobian-based Saliency Map Attack
- DeepFool

3.1 FGSM

Fast Gradient Sign Method is an attack that reduces l_∞ norm. Meaning that $\forall p_{i,j} \in x$ for any pixel in a sample there should be x' such that.

$$p'_{i,j} \in x'$$

and

$$M = min_{x'}(max_{i,j}(|p'_{i,j} - p_{i,j}|)) : f(x', \theta) \neq f(x, \theta)$$

Where M is the l_∞ norm, and it should be minimal so that FGSM can cause a misclassification.

I ran FGSM on model 1 resulting in samples in figure number 8, and on model 2 resulting in the samples in figure number 9.

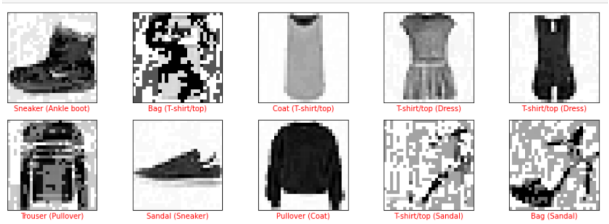

Figure 8: Adversarial examples using FGSM on model 1

Figure 9: Adversarial examples using FGSM on model 2

Confidence of a prediction is the probability of which a model has predicted an example. Confidence of model 1 prediction of adversarial vs benign examples is shown in figure 10.

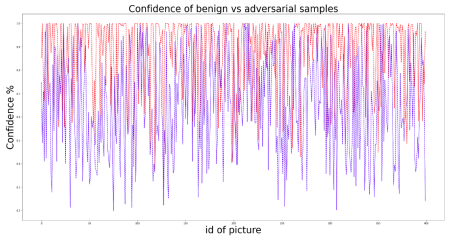

Figure 10: Confidence in adversarial-examples predictions by model 1 (Red for benign examples and blue for FGSM produced examples)

We can see that model 1's confidence of benign and adversarial examples is overlapping therefore the model is susceptible to attacks. A direct cut in confidence cannot be made without a huge loss in accuracy. Whilst confidence in benign and adversarial examples in model 2 is separable to some extent. Check figure 11.

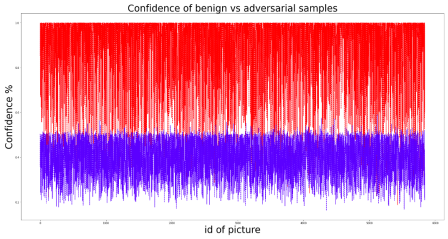

Figure 11: Confidence in adversarial-examples predictions by model 2 (Red for benign examples and blue for FGSM produced examples)

According to figure 11. It is possible to cut at $p = 0.6$ and yield good robustness while losing a small percentage of accuracy.

Running on 6000 examples of the FashoinMnist dataset. The following figure (number 12) represents the average norms and standard deviation of perturbation norms.

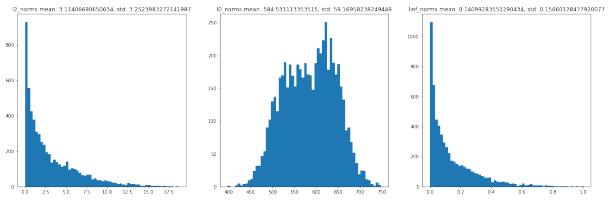

Figure 12: Norms of perturbations from 6000 FashoinMnist pictures when FGSM is run to attack model 1. (left l_2, middle l_0 and right l_∞ norms.

Values in the figure are shown below as $l_{norm} = (\mu, \sigma)$ where μ is the mean across all pixels and σ is the standard deviation across all pixels. $l_2 = (3.11, 3.25), l_0 = (583.53, 59.16), l_\infty = (0.14, 0.15)$ We can see the minimization of l_∞ norm on model 1 having $l_\infty = (0.14, 0.15)$.

For norms statistics when FGSM is run to attack model 2 we can check figure 13.

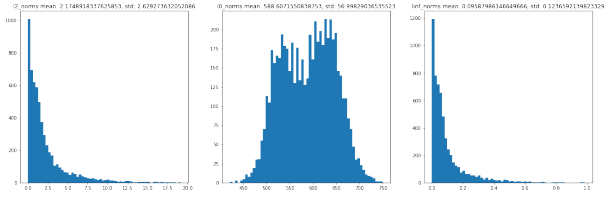

Figure 13: Norms of perturbations from 6000 FashoinMnist pictures when FGSM is run to attack model 2. (left l_2, middle l_0 and right l_∞ norms.

$l_2 = (2.17, 2.62), l_0 = (588.60, 56.99), l_\infty = (0.09, 0.12)$ It is crucial to point out that model 1 forced FGSM to increase M (0.09,0.15) for model 2 and (0.14,0.15) for model 1. Which is a good property to construct defenses. However, the confidence overlapping makes it harder.

3.2 DeepFool

DeepFool minimizes l_2 norm. Samples after running DeepFool on the models are on figure 10 for model 1 and figure 11 for model 2. Meaning that $\forall p_{i,j} \in x$ for any pixel in a sample there should be x' such that.

$$p'_{i,j} \in x'$$

and

$$M = min_{x'}(max_{i,j}(|p'_{i,j} - p_{i,j}|_2)) : f(x', \theta) \neq f(x, \theta)$$

Where M is the l_2 norm, and it should be minimal so that DeepFool can cause a mis-classification.

I ran DeepFool on model 1 resulting in samples in figure number 14, and on model 2 resulting in the samples in figure number 15.

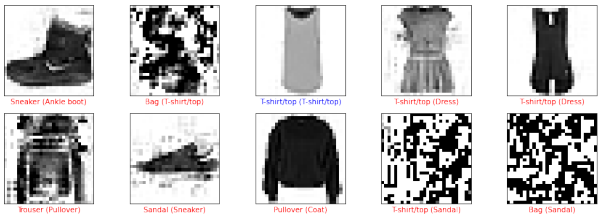

Figure 14: Adversarial examples using DeepFool on model 1

Figure 15: Adversarial examples using DeepFool on model 2

We can see that the confidence of DeepFool produced examples on model 1 -as for FGSM- are overlapping with the confidence in benign examples. Differently than FGSM in model 2, there is no strict threshold that separates confidence in benign and adversarial examples.

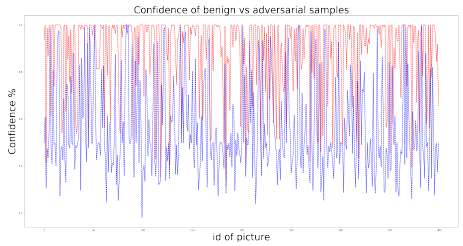

Figure 16: Confidence in adversarial examples predictions by model 1 (Red for benign examples and blue for DeepFool produced examples)

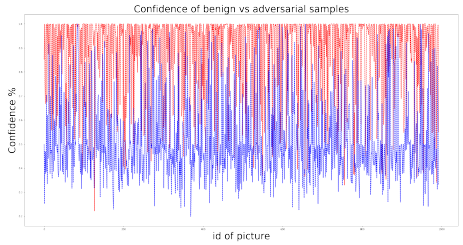

Figure 17: Confidence in adversarial examples predictions by model 2 (Red for benign examples and blue for DeepFool produced examples)

After running DeepFool on both models, the confidence graph was overlapping therefore we can deduce that both models are susceptible to attacks. You can check in this **notebook**

Lets check the norms statistics when DeepFool is run to attack model 1.

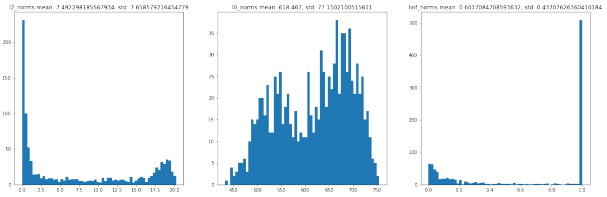

Figure 18: Norms of perturbations from 1000 FashoinMnist pictures when DeepFool is run to attack model 2. (left l_2, middle l_0 and right l_∞ norms.

$l_2 = (7.49, 7.65), l_0 = (613.64, 77.15), l_\infty = (0.60, 0.43)$ for model 2 values were $l_2 = (6.33, 7.41), l_0 = (626.70, 68.53), l_\infty = (0.55, 0.43)$. Model 1 forced DeepFool to increase M.

3.3 JSMA

Jacobian-based Saliency Map Attack minimizes the l_0 norm, which is the number of pixels changed because of the attack. It means that JSMA will choose some pixels and will change them to cause the attack while keeping the number of such pixels at a minimum.

Check figure 16 to see the impact of JSMA on samples to attack model 1 and figure 17 to attack model 2. The change is not clear because JSMA minimizes the number of pixels changed.

Figure 19: Samples produced by JSMA when run to attack model 1

Confidence intervals in JSMA examples are highlighted in figure 18 (model 1) and 19 (model 2).

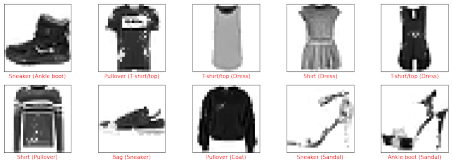

Figure 20: Samples produced by JSMA when run to attack model 2

Confidence of model 1: JSMA was run to produce 100 adversarial examples. The confidence in these examples is captured in the following figure 21 (model 1) and figure 22 (model 2)

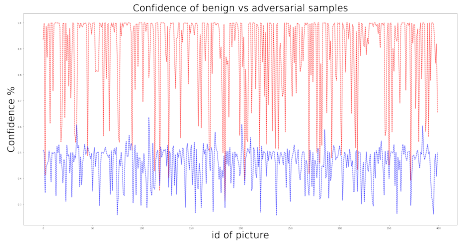

Figure 21: Confidence in adversarial examples predictions by model 2 (Red for benign examples and blue for JSMA produced examples)

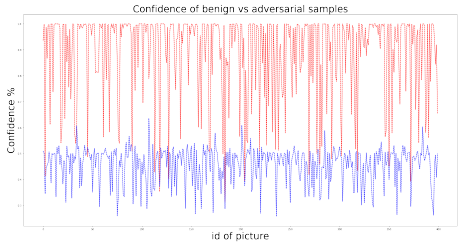

Figure 22: Confidence in adversarial examples predictions by model 2 (Red for benign examples and blue for JSMA produced examples)

We can set a certain threshold for both models to detect adversarial examples. There is a trade-off between accuracy and robustness in all models.

Norms statistics: we know that JSMA reduces l_0 norm. However, It takes so much time to produce an adversarial example. Let's check the norms for models 1 and 2.

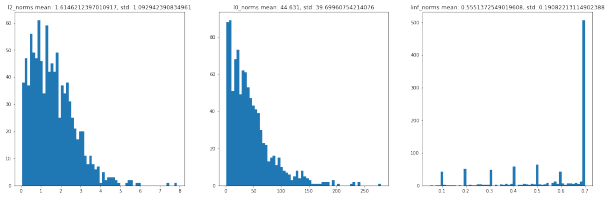

Figure 23: Norms of perturbations from 100 FashoinMnist pictures when JSMA is run to attack model 1. (left l_2, middle l_0 and right l_∞ norms.

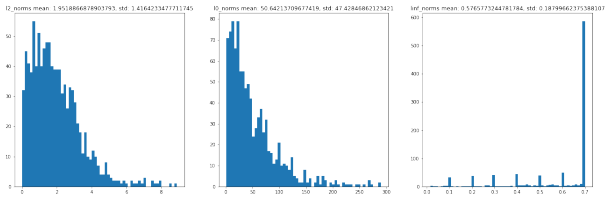

Figure 24: Norms of perturbations from 100 FashoinMnist pictures when JSMA is run to attack model 2. (left l_2, middle l_0 and right l_∞ norms.

$l_2 = (1.61, 7.65), l_0 = (44.63, 39.69), l_\infty = (0.55, 0.19)$ for model 2 values were $l_2 = (1.95, 1.41), l_0 = (50.64, 47.42), l_\infty = (0.57, 0.13)$. Model 2 forced JSMA to increase M (number of pixels here).

4 DEFENSES

Defenses are methods that are applied to the target model to strengthen it. So it becomes either robust or less susceptible to attacks. I examined three types of defenses:

- Adversarial Training: Which is to train on the adversarial examples. The model classification borders would be extended to cover blind spots and prevent attacks.
- Randomization: To add a dropout layer in the training, and to randomize gradients flow in the evaluation so that the attacker cannot possibly produce an example that can capture all combinations of the gradient flows through the neural network.
- Network Distillation: This is to smooth out gradients' borders of classes by using a student model. The student model will learn the probabilities (continuous spectrum) of the teacher model (our target model). This proved out to make gradient-based attacks job harder and increased the robustness of the target model.

4.1 Adversarial Training

Set a threshold and then do iterative training on adversarial examples until the following inequity doesn't hold.

$$|x_{adv} - x|_{norm} < \epsilon$$

Adversarial training without exposing the new weights to the attacker increased the robustness from 0.2 % to 98 %. Iterative training as mentioned before takes a lot of time to train. The amount of classification borders to be shaped is enormous. I spent 10 hours on iterative adversarial training. The models' robustness increased in training examples. However, when a new example (one model didn't see in the training phase) was introduced the attacker model found an adversarial example easily. It is because of the huge amount of classification blindspot where the border is not formed correctly in the neural network.

4.2 Network Distillation

Check Notebook "Adversarial Defenses Iteration 5" there is a description of the student and teacher models. Accuracy of the student model dropped from (87% for teacher) to (77 % for student). However, the attacker ability (FGSM attack being able to fool the model) dropped from 99.8 % to 83.2 %. Robustness has risen from 0.2 % to 16.8 %. Increased by a factor of 84.

Model 1 was, in fact, an example of network distillation because its encoder was fetched from a DAE. Knowing that a DAE was trained on continuous data (the labels were the pictures themselves). It forced FGSM and DeepFool to increase their ϵ while it was in a counter effect in JSMA. The student model in the aforementioned notebook performed very well to increase the robustness.

4.3 Randomization

The main idea is to ignore the activations of a subset of neurons randomly. Suppose we have $\gamma = 0.2$ meaning that 20 % of the activations in some layer would be ignored. The attacker has to take into account $C = \binom{n}{\gamma n}$ where n is the number of neurons. A common network would have 100 neurons. Therefore $\binom{100}{0.2*100} = \binom{100}{20} = 5.3510^{20}$. It is a huge amount of combinations that are not feasible for modern-day computations. Regardless of the ability to have a solution that satisfies all the gradient flows and would minimize a norm so that the adversarial property holds.

Taking different gradient flow each time in a random fashion made the work of the network hard. I think the randomization defense would force white-box attacks out of this type. Mainly because the gradient flow is different every time. It is the same as using an attack on a modified version of the model with slightly different weights. The white box category is debatable here. However, it is great if one wants to use an open-source model without changing its weights. Randomization increased the robustness to 99.8 %.

5 ROBUSTNESS VS ACCURACY

It has been established that there is a certain trade-off between accuracy and robustness. The following figure shows the accuracy and robustness vs threshold of confidence.

Threshold τ ranging between $[0, 1[$ samples at a rate of 0.01.

$$\forall x : f(x, \theta) < \tau \rightarrow x = x_{adv}$$

Any sample which has low confidence below τ would be considered adversarial. Now let's look at the chart.

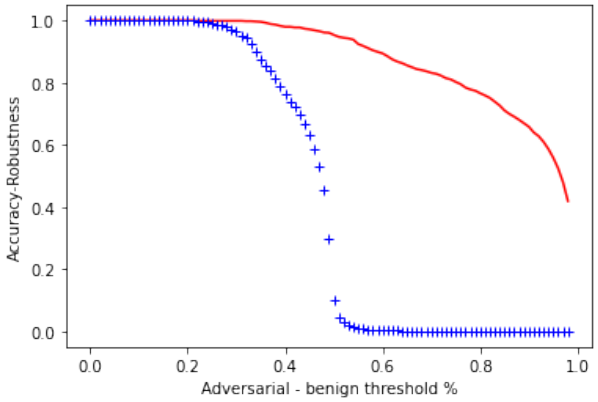

Figure 25: Benign accuracy in blue, Adversarial robustness in red

It is clear that we need a threshold $\tau > 0.5$ to maintain good robustness but for a 20% decline in the model's accuracy. (note accuracy is normalized).

6 FUTURE WORK

Further experiments could be made on white-box attacks. The whole field of limited black-box and score-based black-box attacks could be investigated. Thesis work could be to move from images to NLP and produce attacks to target spam detection neural nets.

7 ABBREVIATIONS INDEX AND NOTES

7.1 Index

Here are some of the abbreviations used in this document:

DNN	Deep Neural Network
ANN	Feed Forward Neural Network
RNN	Recurrent Neural Network
DAE	Denoising AutoEncoder
CNN	Convolutions Neural Network
FGSM	Fast Gradient Sign Method
JSMA	Jacobian-based Saliency Map Attack
NLP	Natural Language Provessing

Table 1: Abbreviations Index

7.2 L_i norms

L_i is a norm defined in the expression below:

$$L_i = \sqrt[i]{\sum_{j=0}^{j=N-1} |x_j|^i}$$

Where N is the number of pixels in a perturbation and x_j is the jth pixel.

7.3 Notebooks

Experiments on FGSM attack:
FGSM and experiments Notebook
Experiments on DeepFool attack:
DeepFool and experiments Notebook
Experiments on JSMA attack:
JSMA and experiments Notebook
Experiments on Defenses
Defenses Notebook

REFERENCES

[1] Simple Black-Box Adversarial Attacks on Deep Neural Networks ieeexplore.ieee.org/abstract/document/8014906
[2] Deep learning book www.deeplearningbook.org
[3] Maximal Jacobian-based Saliency Map Attack https://arxiv.org/pdf/1808.07945.pdf
[4] Git repo https://github.com/JafarBadour/Adversarial-Attacks-on-Neural-Networks
[5] Open source code for attacks and defenses implementations https://github.com/bethgelab/foolbox
[6] Strengthening Deep Neural Networks: Making AI Less Susceptible to Adversarial Trickery ISBN: 1492044903, 9781492044901
[7] Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning https://www.coursera.org/account/accomplishments/records/VMUNW8J4Y6L4
[8] Full project on adversarial attacks with published papers by Xin Lee and Fuxin Lee https://github.com/ya332/Adversarial-Attacks-Neural-Networks
[9] Simple python notebook to try to make adversarial example to fool classifier on MNIST dataset https://github.com/dangeng/Simple_Adversarial_Examples
[10] CNN and GAN implementation from last semester https://colab.research.google.com/drive/17NJH78-dKEyeDS5tas5baN5VTMFgyS59
[11] Distilling the Knowledge in a Neural Network https://arxiv.org/abs/1503.02531
[12] The Limitations of Deep Learning Adversarial Settings https://arxiv.org/pdf/1511.07528.pdf
[13] Adversarial Examples: Attacks and Defenses forDeep Learning https://arxiv.org/pdf/1712.07107.pdf
[14] Strengthening Deep Neural Networks: Making AI Less Susceptible to Adversarial Trickery ISBN: 1492044903, 9781492044901
[15] Deep Learning Specialization by deeplearning.ai https://www.coursera.org/specializations/deep-learning
[16] Open source code for attacks and defenses implementations https://github.com/bethgelab/foolbox
[17] Introduction to TensorFlow for Artificial Intelligence, Machine Learning, and Deep Learning https://www.coursera.org/account/accomplishments/records/VMUNW8J4Y6L4