Fall 2109

Synthesizing Adversarial Examples for Neural Networks

Hasitha Rasineni
hasithar@iastate.edu

Synthesizing Adversarial Examples for Neural Networks

by

Hasitha Rasineni

A creative component report submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Electrical Engineering

Program of Study Committee:
Zhengdao Wang, Major Professor

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University
Ames, Iowa
2019

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Zhengdao Wang for introducing me to the topic of Adversarial Examples in the field of Machine Learning. I really appreciate his guidance and support during my creative component project. His suggestions and expertise helped me to successfully complete my project.

In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at the Department of Electrical Engineering, and Iowa State University a wonderful experience.

ABSTRACT

As machine learning is being integrated into more and more systems, such as autonomous vehicles or medical devices, they are also becoming entry points for attacks. Many state-of-the-art Neural Networks have been proved to be consistently misclassifying adversarial examples. These failures of machine learning models demonstrate that even simple algorithms can behave very differently from what their designers intend. In order to close this gap between what designers intend and how algorithms behave, there is a huge need for preventing adversarial examples to improve credibility of the model.

In this creative component, I have synthesized adversarial examples using two different white-box methods – 'Fast Gradient Sign Method (FGSM)[1]' and 'Expectation Over Transformation (EOT)[2]'. And used the synthesized adversarial examples to see how vulnerable the neural networks are to them.

CHAPTER 1. OVERVIEW

1.1 Introduction

Adversarial examples have been an interesting topic in the world of deep neural networks recently. Adversarial examples challenge the robustness of state-of-the-art machine learning/ deep learning models. It's no surprise that Deep Learning is very effective and powerful in solving many complex computer vision problems. However, despite being extremely accurate, these Deep Learning models are highly vulnerable to attacks. One such attacks are adversarial examples. Recent studies by Google Brain have shown that any machine learning classifier can be tricked to give incorrect predictions, and with a little bit of skill, we can get them to give pretty much any result we wish. As many different fields are starting to use deep learning in critical systems, it is important to bring awareness on how neural networks can be fooled to result in strange and potentially dangerous behaviors, as most of them are crucial for our safe and comfortable life. Banks, surveillance systems, ATMs, face recognition on your laptop and very soon, self-driving cars.

The authors in the paper – 'Intriguing properties of neural networks' [3] discovered a worrying fact about the models they were experimenting with - that is we could often induce a network to change the predicted class of the label, while not changing that how that image is perceived by humans. A well-trained model should be relatively invariant to small amounts of noise. And, when it came to random noise, this was in fact generally the case, experiments have typically confirmed that adding true white noise to an image typically doesn't impact the predictions of well-performing models. But when it comes to non-random noise, noise specifically engineered to fool the network, a surprisingly small amount of such noise, much less than is perceptible by a human eye, can meaningfully shift the network's final output.

Synthesizing adversarial examples has also been proved to be fairly easy in the case where the attacker has knowledge about the model architecture and the underlying parameters and a bit difficult without that knowledge. Many algorithms have been put forward by researchers in order to generate adversarial examples based on the training set. The paper – Synthesizing robust adversarial examples[2] presents an algorithm to synthesize 3D adversarial examples which remain adversarial in real life.

1.2 Goal

The main goal of this creative component is to generate adversarial examples based on two different methods - 'Fast Gradient Sign Method (FGSM)[1]' and 'Expectation Over Transformation (EOT)[2]'. And to use the synthesized adversarial examples to test the robustness and reliability of the networks.

CHAPTER 2. REVIEW OF LITERATURE

2.1 Adversarial Examples

An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a Machine Learning classifier to misclassify it. [1] An attacker intentionally designs inputs to machine learning models to cause the model to make a mistake. Adversarial examples are referred to as optical illusions for machines. It is easier to get a sense of this phenomenon thinking about it in a computer vision setting, in computer vision, these are small perturbations to input images that result in an incorrect classification by the models.

Why is it important?

Deep neural networks are highly expressive models that have recently achieved state of the art performance on speech and visual recognition tasks. While their expressiveness is the reason they succeed, it also causes them to learn uninterpretable solutions that could have counter-intuitive properties.[3] Several machine learning models, including neural networks, have consistently misclassified adversarial examples, inputs formed by applying small but intentionally worst-case perturbations to examples from the dataset, such that the perturbed input results in the model outputting an incorrect answer with high confidence.

Adversarial attacks are an important topic of research and consideration because it has been shown that adversarial examples transfer from one model to another. In other words, adversarial examples generated to fool one model can also fool other models using a different architecture or trained using different data sets for the same task. AI Security is one of the areas that needs to quickly evolve to keep up with the advancements in deep learning technology. Following are some of the examples where adversarial examples make machine learning models vulnerable to attacks:

- A self-driving car crashes into another car because it ignores a stop sign. Someone had placed a picture over the sign, which looks like a stop sign with a little dirt for humans, but was designed to look like a unlimited speed sign for the sign recognition software of the car.
- A spam detector fails to classify an email as spam. The spam mail has been designed to resemble a normal email, but with the intention of cheating the recipient.
- A machine-learning powered scanner scans suitcase for weapons at the airport. A knife was developed to avoid detection by making the system think it is an umbrella.

2.2 Types of Adversarial Attacks

There are two different types of adversarial attacks based on the knowledge of the attacker – White Box attacks and Black Box attacks. White-box attacks are where the adversary has complete knowledge and access to the model, including architecture, inputs, outputs, and weights. Whereas, in the Black box attacks, the attacker only has access to the inputs and outputs of the model, and knows nothing about the underlying architecture or weights. It is fairly simple to synthesize adversarial examples for a model with known system architecture and underlying parameters, as in the case of white box attacks. Generating adversarial examples for black box attacks, where the adversary has no knowledge about the model being attacked, is an extremely difficult task which involves a lot of trial and error. One of the most well-known black box attack strategies is to train a local substitute network with a synthetic dataset: the inputs are synthetic and generated by the adversary, while the outputs are labels assigned by the target network and observed by the adversary. Adversarial examples crafted using the substitute parameters, are not only misclassified by the substitute but also by the target network, because both models have similar decision boundaries. Now this becomes a huge security risk as attackers can develop local models for the same task as the target model, generate adversarial examples for the local model and use them for attacking the target. This puts a whole lot of mainstream or soon to be mainstream applications like facial recognition, self-driving cars, biometric recognition etc. that leverage machine learning based computer vision models at risk.

2.3 Types of Misclassification

There are also several types of goals for an attacker, which are non-targeted misclassification and targeted misclassification. The goal of adversary in non-targeted misclassification is to only change the output classification from the true label and does not really worry about the new classification. Targeted misclassification is where the attacker wants to modify an image that is originally of a specific source class to be classified as a specific target class.

2.4 Non-Targeted Misclassification

One idea is to generate some image that is designed to make the neural network have a certain output. Let the goal label be y_{goal}. We like to synthesize an image such that the neural network's output is y_{goal}. We can formulate this as an optimization problem in much the same way we train a network.

$$C = \frac{1}{2}\|y_{goal} - \hat{y}(\vec{x})\|_2^2$$

The output of the neural network given our image is $y(\vec{x})$. You can see that if the output of the network given the generated image $\vec{x}$ is very close to the goal label $y_{\_goal}$, then the corresponding cost is low. If the output of the network is very far from our goal then the cost is high. Therefore, finding a vector $\vec{x}$ that minimizes the cost $\boldsymbol{C}$ results in an image that the neural network predicts as the assigned goal label. The goal is to find this vector $\vec{x}$. This problem is incredibly similar to how we train a neural network, where we define a cost function and then choose weights and biases that minimize the cost function. In the case of adversarial example generation, instead of choosing weights and biases that minimize the cost, we hold the weights and biases constant and choose an $\vec{x}$ input that minimizes the cost.

Figure 1, *Figure 2* and *Figure 3* are non-targeted adversarial examples for each class along with the neural network's predictions, depicted below. They are adversarial examples generated for MNIST dataset. Though they look like images with random noise to human eyes, the model classifies them as numbers, in *Figure 3*, the model only predicts the image to be number 0, but also has a very high confidence of more than 80%.

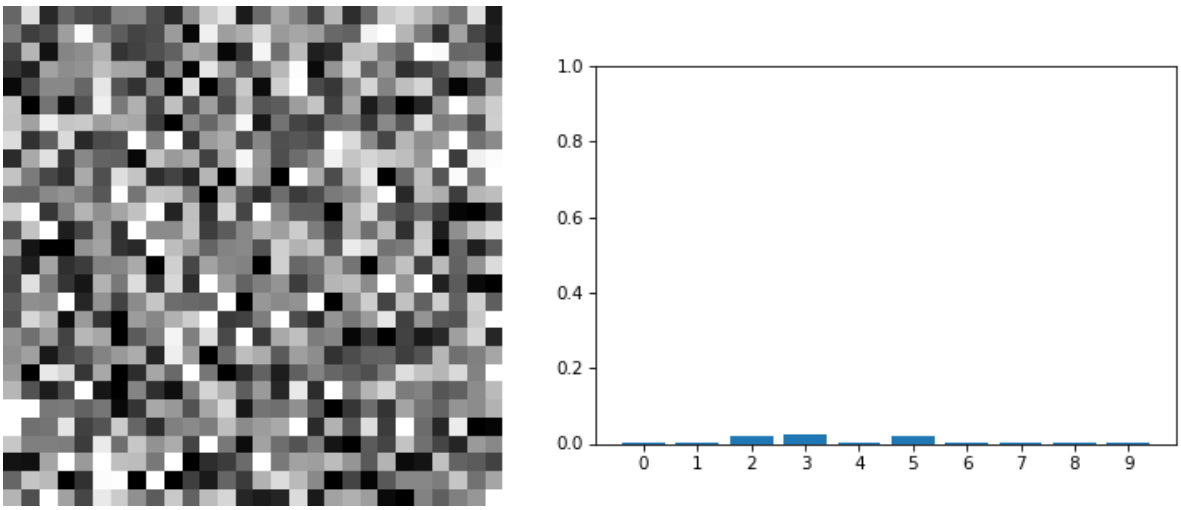

Figure 1. *Left side is the non-targeted adversarial example (a 28 X 28 pixel image). The right side plots the activations of the network when given the image.* [4]

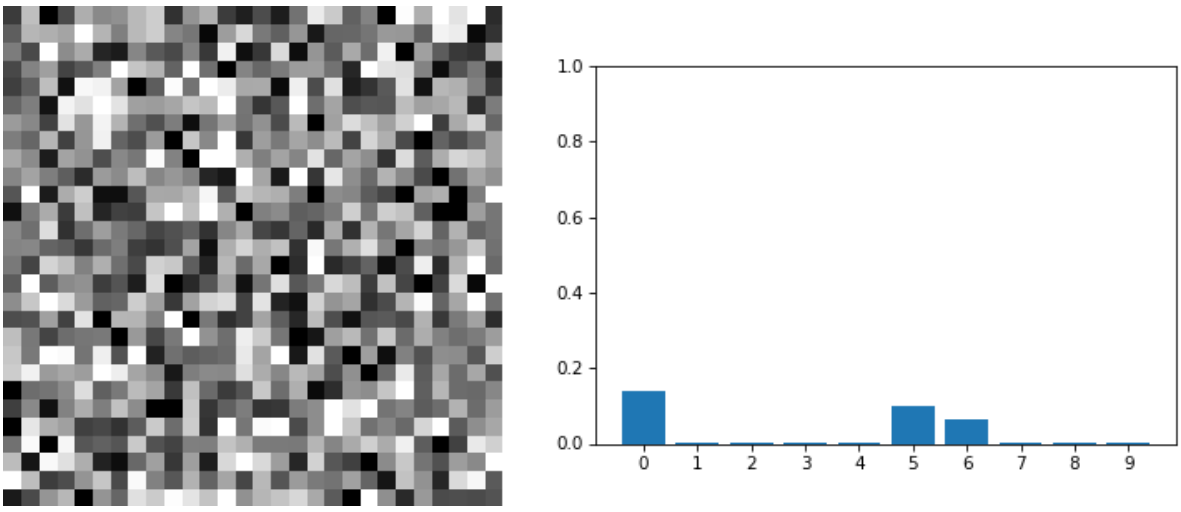

Figure 2. *Left side is the non-targeted adversarial example (a 28 X 28 pixel image). The right side plots the activations of the network when given the image* [4]

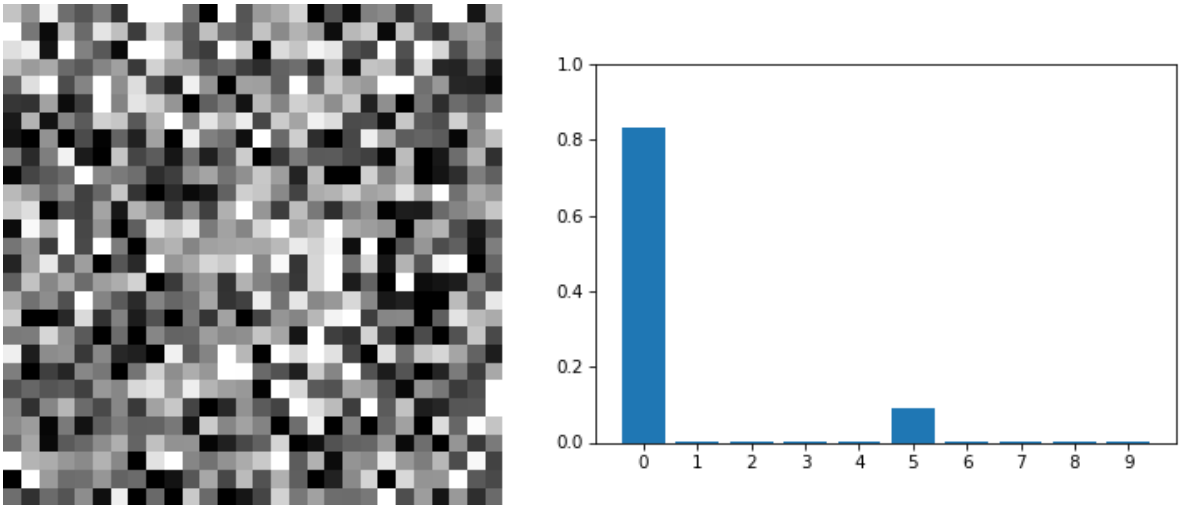

Figure 3. *Left side is the non-targeted adversarial example (a 28 X 28 pixel image). The right side plots the activations of the network when given the image* [4]

2.5 Targeted Misclassification

Non-targeted adversarial examples are generated by adding a term to the cost function that has been mentioned above and minimizing it. The cost function to synthesize non-targeted adversarial examples will be,

$$C = \frac{1}{2}\|y_{goal} - \hat{y}(\vec{x})\|_2^2 + \lambda\|\vec{x} - x_{target}\|_2^2$$

This is where we want x_{target} to look almost similar to x. Minimizing the first term will make the neural network output *y_goal* when given $\vec{x}$. Minimizing the second term will try to force our adversarial image x to be as close as possible to x_{target} as possible. λ out front is a hyperparameter that dictates which of the terms is more important. As with most hyperparameters we have to tune λ to find the best possible value to get the desired result.

Figure 4, *Figure 5* and *Figure 6* are targeted adversarial examples. They have been generated by adding very small calculated noise to the original MNIST images, thus making the model misclassify the images as desired target lables.

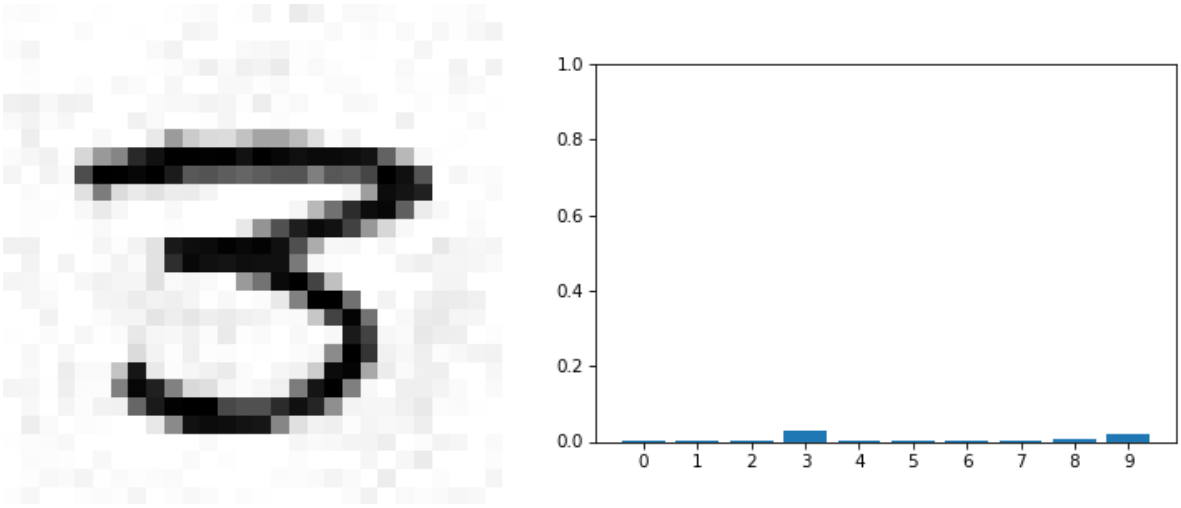

Figure 4. *Left side is the non-targeted adversarial example (a 28 X 28 pixel image). The right side plots the activations of the network when given the image*[4]

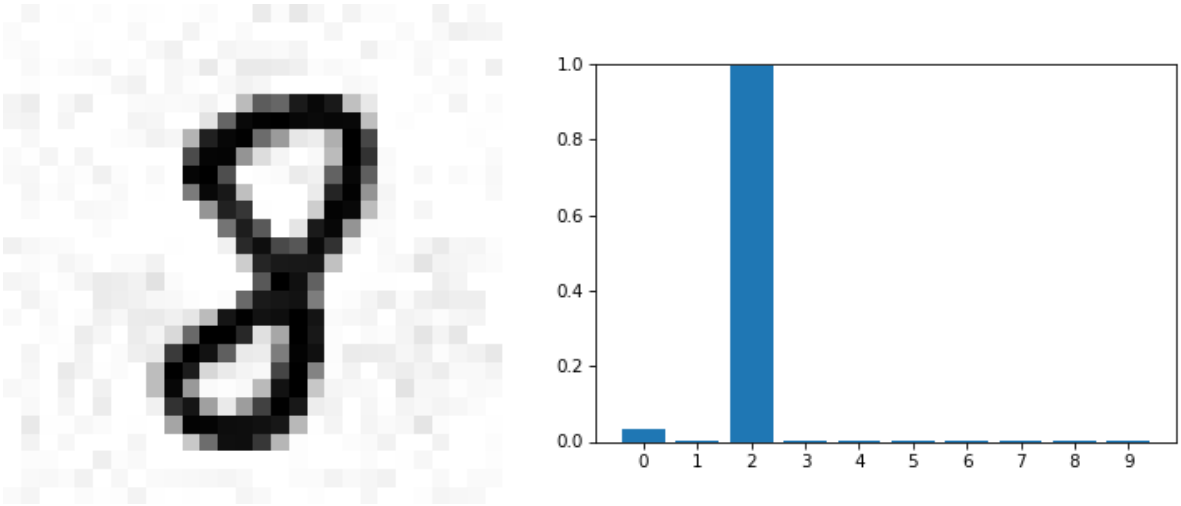

Figure 5. *Left side is the non-targeted adversarial example (a 28 X 28 pixel image). The right side plots the activations of the network when given the image*[4]

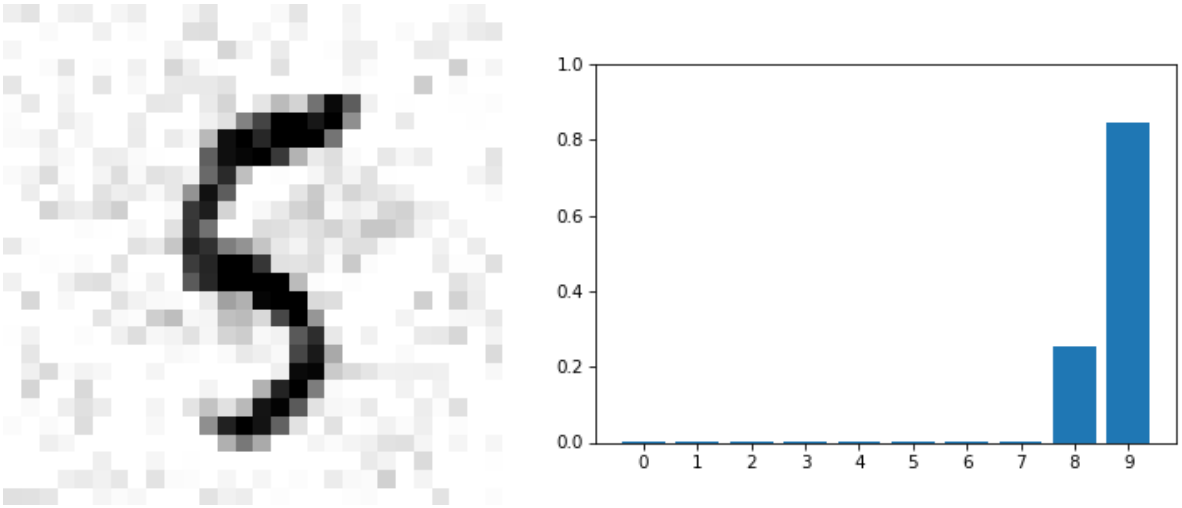

Figure 6. *Left side is the non-targeted adversarial example (a 28 X 28 pixel image). The right side plots the activations of the network when given the image*[4]

One of the most popular examples of targeted adversarial attack is from the paper – 'Explaining and Harnessing Adversarial Examples'[1] where a small amount of calculated noise had been added to the original 'panda' image and the target label has been set to 'gibbon'. *Figure 6* shows that, not only the image which looks like a panda to human eye has been classified as gibbon, we can also notice how the model has very high confidence in the misclassification which is more than 99% in comparison with the correct classification which is just 57.7%.

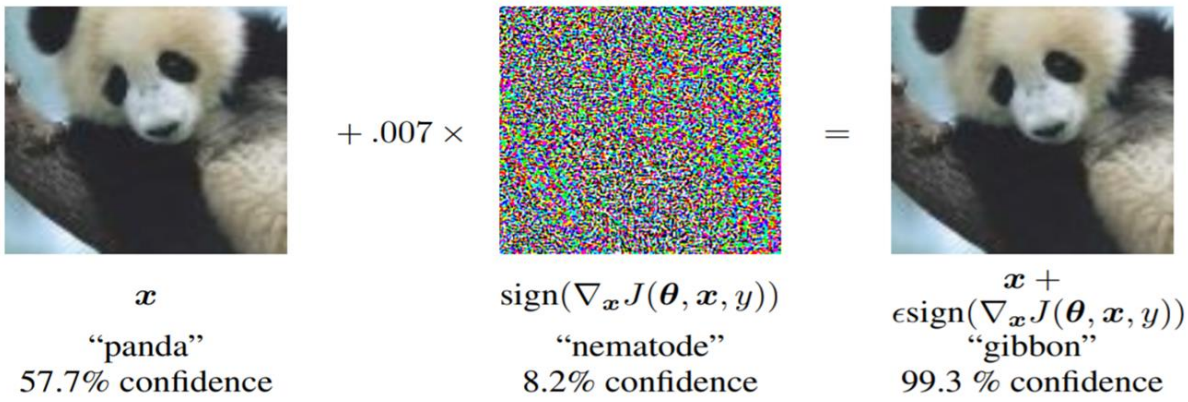

Figure 7. *Targeted Adversarial example of a panda misclassified as a gibbon*[1]

CHAPTER 3. METHODS AND RESULTS

My focus for this creative component is to synthesize adversarial examples using two different white box attacks - Fast Gradient Sign Method (FGSM) *by Goodfellow et. al*.[1] and Expectation Over Transformation (EOT) Method – by A. Athalye et. al.[2] Many techniques have been put forward to generate adversarial examples. Most approaches suggest minimizing the distance between the adversarial example and the object to be altered, while changing the prediction to the desired label. Some methods require access to the gradients of the model, which of course only works with gradient based models such as neural networks, other methods only require access to the prediction function, which makes these methods model-agnostic[5].

3.1 Fast Gradient Sign Method (FGSM)

The fast gradient sign method uses the gradient of the underlying model to find adversarial examples. The original image x is altered by adding or subtracting a small noise ϵ to each pixel of the image. Whether we add or subtract ϵ depends on the sign of the gradient for a pixel, positive or negative. Adding error in the direction of the gradient means that the image is intentionally modified so that the model misclassifies the image. It is designed to attack neural networks by leveraging the way they learn, *gradients*. The idea is to, rather than working to minimize the loss by adjusting the weights based on the backpropagated gradients, the attacker adjusts the input data to maximize the loss based on the same backpropagated gradients. This method computes an adversarial image by adding some weak noise on every step of optimization, drifting towards the desired class or away from the correct one. The adversarial example is generated using the following formula,

$$x_{adv} = x + \epsilon \cdot sign(\nabla x J(\theta, x, y))$$

- x is the original image, y the class of x, θ the weights of the network and $J(\theta, x, y)$ the loss function used to train the network.
- sign of the slopes to know if we should increase or decrease the pixel values to maximize the loss.
- ϵ to ensure that the perturbation will be imperceptible**.**

The paper also talks about how linear behavior in high-dimensional spaces is sufficient to cause adversarial examples and that enabled them to design a fast method for synthesizing adversarial examples that makes adversarial training practical. The authors also suggest that generic regularization strategies such as dropout, pretraining, and model averaging do not benefit a significant reduction in a model's vulnerability to adversarial examples. They suggest changing to nonlinear models such as RBF networks can reduce the vulnerability of the models to adversarial examples. There is a tradeoff between linearity of the models (easy to train) and their vulnerability to adversarial examples. They project that it may be possible to escape this tradeoff in the future by designing more powerful optimization methods that can successfully train more nonlinear models.

I have used fast gradient method to synthesize adversarial examples for MNIST dataset trained on a Convolutional Neural Network with the architecture as shown in ***Figure 7.***

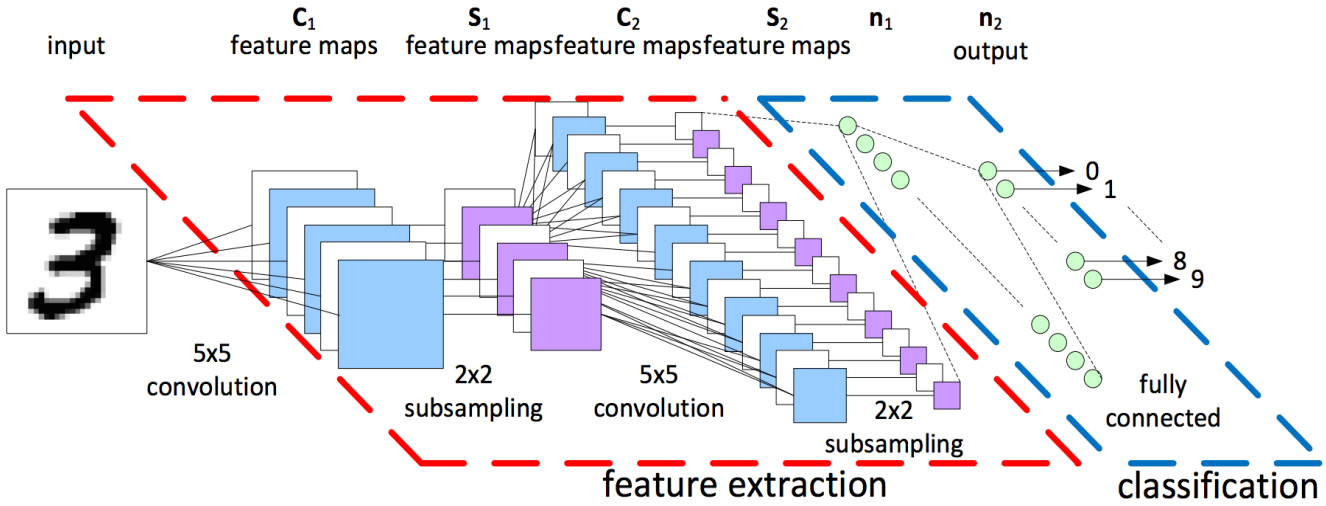

Figure 8. *CNN architecture for Classification of MNIST Dataset*[6]

Figure 8 shows a snippet of code for the implementation of Fast Gradient Sign Method. I have generated adversarial examples using FGSM using different values of ϵ. It can be observed from *Figure 9*, that as the value of ϵ increases, the accuracy of the CNN model decreases. This is because increasing ϵ implies increasing the noise added to each pixel of the original image, which results in increasing the probability of classifying the image into desired target class.

```
In [21]: #Targeted attack using Fast gradient sign method
adversarial_img = original_images.copy()
for i in range(0, steps):
    gradient = img_gradient.eval({x: adversarial_img, y_: target_labels, keep_prob: 1.0})
    #Update using value of gradient
    adversarial_img = adversarial_img - step_size * gradient
    prediction = tf.argmax(y,1)
    prediction_val = prediction.eval(feed_dict={x: adversarial_img, keep_prob: 1.0}, session=sess)
    print("predictions", prediction_val)
    probabilities = y
    probabilities_val = probabilities.eval(feed_dict={x: adversarial_img, keep_prob: 1.0}, session=sess)
    print('Confidence 2:', probabilities_val[:, 2])
    print('Confidence 6:', probabilities_val[:, 6])
```

Figure 9. *Code snippet to implement Fast Gradient Sign Method*

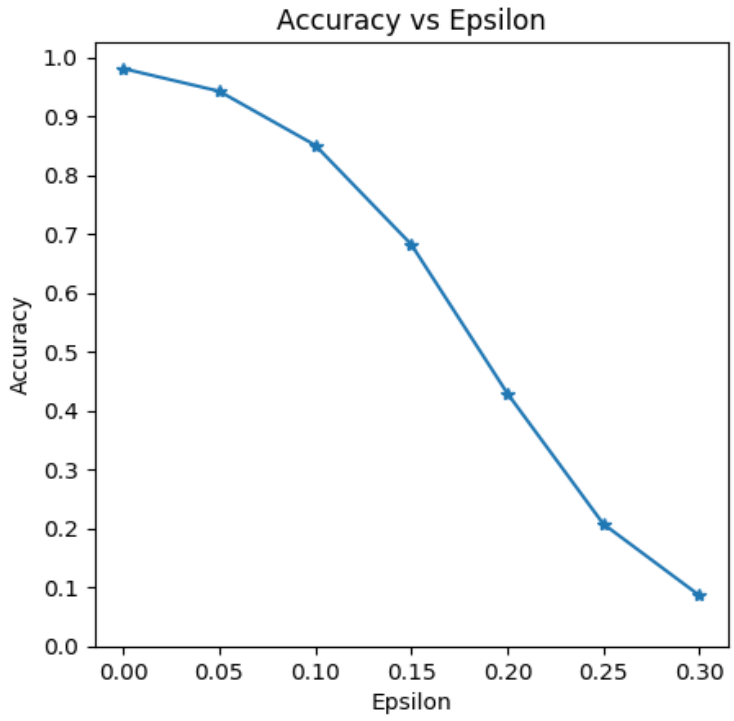

Figure 10. *Accuracy of the model vs Epsilon*

$\epsilon = 0$ implies that no noise has been added to the pixels of the image, hence the model has an accuracy of nearly 99%. As the value of ϵ increases, the accuracy of the model decreases. When $\epsilon = 0$ the model has the least accuracy of less than 10%.

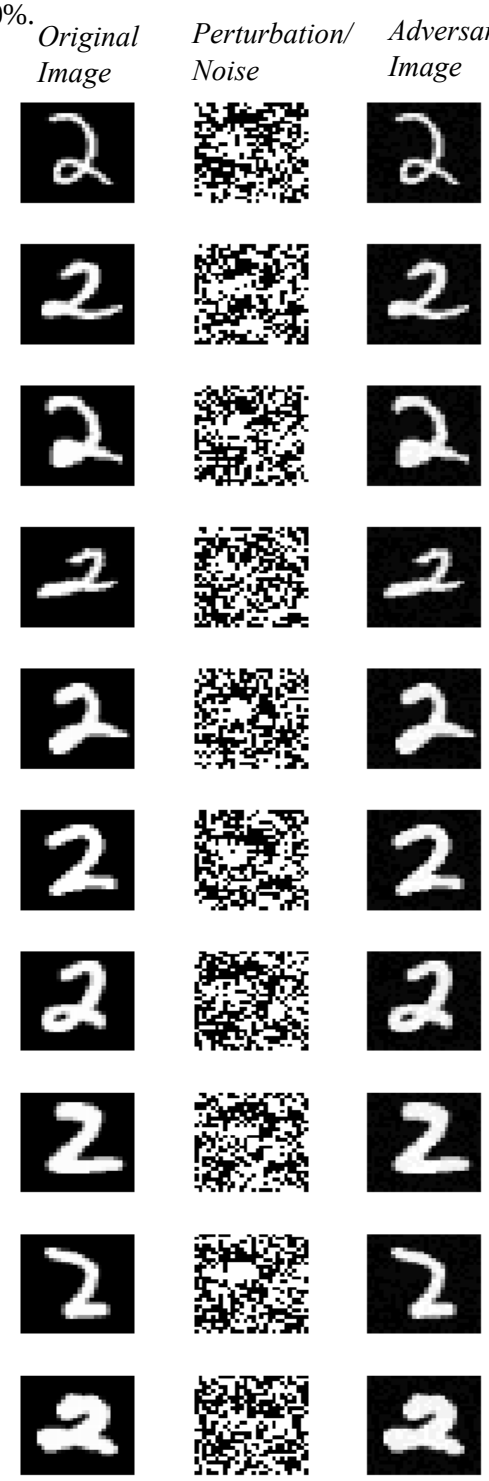

Figure 11. *Adversarial example for the image 2 when $\epsilon = 0.02$*

Figure 10 shows the set of adversarial examples generated for the image with number 2 when $\epsilon = 0.02$. As the value of ϵ is very small, the adversarial image looks exactly similar to the original image. It is because only a very tiny amount of noise has been added to each pixel of the original image.

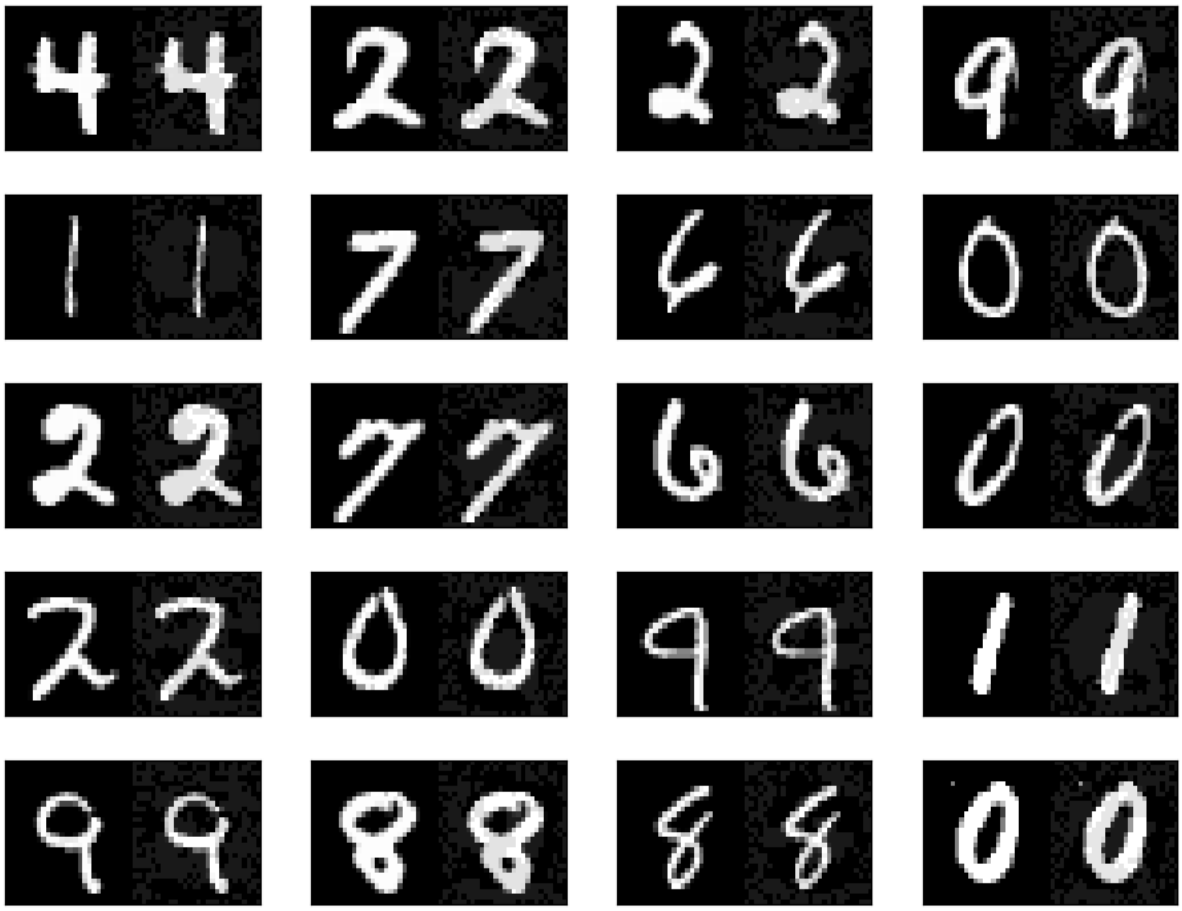

Figure 12. *Adversarial examples for MNIST images when $\epsilon = 0.3$*

From *Figure 11*, we can observe that there is a visible difference between the original and adversarial images, this is because the values of epsilon is much greater. As we are adding a significant amount of noise to each pixel of the original image, the difference is not imperceptible.

3.2 Expectation Over Transformation (EOT) Method

EOT uses a chosen distribution T of transformation functions t taking an input x' controlled by the adversary to the true input t(x') perceived by the classifier. Instead of simply taking the norm of x' - x to constrain the solution space, given a distance function d(.,.), EOT aims to constrain the expected effective distance between the adversarial and original inputs, which is defined as[2]:

$$\delta = E_{t\sim T}\,[d(t(x'), t(x))]$$

Rather than optimizing the log-likelihood of a single example, EOT uses a chosen distribution T of transformation functions t taking an input x 0 controlled by the adversary to the "true" input t(x 0) perceived by the classifier[2]. Thus the optimization problem in EOT is as follows:

$$\begin{aligned} \arg\max_{x'} \quad & \mathbb{E}_{t\sim T}[\log P(y_t|t(x'))] \\ \text{subject to} \quad & \mathbb{E}_{t\sim T}[d(t(x'), t(x))] < \epsilon \\ & x \in [0,1]^d \end{aligned}$$

In practice, the distribution T can be any transformation such as random rotation, translation, or addition of noise. However, the method generalizes beyond simple transformations; transformations in T can perform operations such as 3D rendering of a texture. The authors have proved that a 3D printed turtle has been classified as a rifle in all possible angles. The authors also talk about how the methods to generate adversarial examples before have not been able to generate adversarial examples that are robust in real life, and hence they came up with Expectation Over Transformation method which can be used to synthesize adversarial examples which will be adversarial in real life scenarios as well.

I have generated adversarial examples for Inception v3 network trained on ImageNet using expectation over transformation method. Synthesized a single adversarial input that's robust to rotation by $\theta \in [-\pi/4, \pi/4]$.

```
In [26]: num_samples = 10
         average_loss = 0
         for i in range(num_samples):
             rotated = tf.contrib.image.rotate(
                 image, tf.random_uniform((), minval=-np.pi/4, maxval=np.pi/4))
             rotated_logits, _ = inception(rotated, reuse=True)
             average_loss += tf.nn.softmax_cross_entropy_with_logits(
                 logits=rotated_logits, labels=labels) / num_samples
```

Figure 13. *Code snippet to implement Expectation Over Transformation Method*

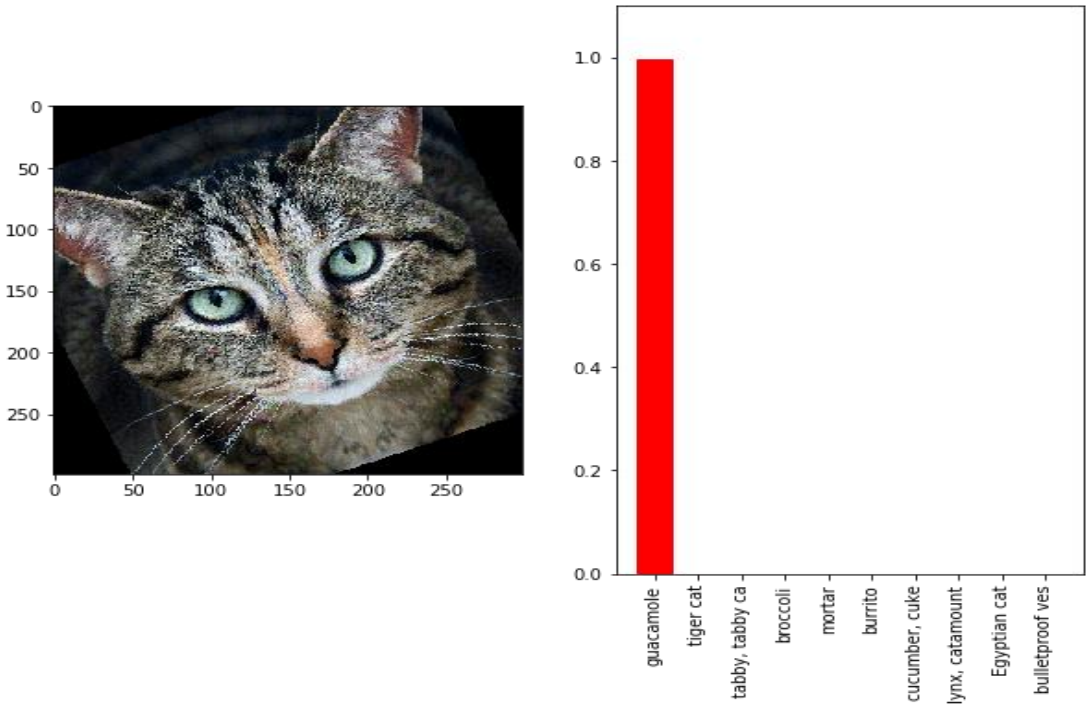

Figure 14. *Tabby Cat misclassified as Guacamole*

I have set my target class as *guacamole.* As shown in *Figure 13*, 'Tabby Cat' has been misclassified as 'guacamole', the targeted class with 100% accuracy. The original image has been rotated by an angle of $\pi/8$.

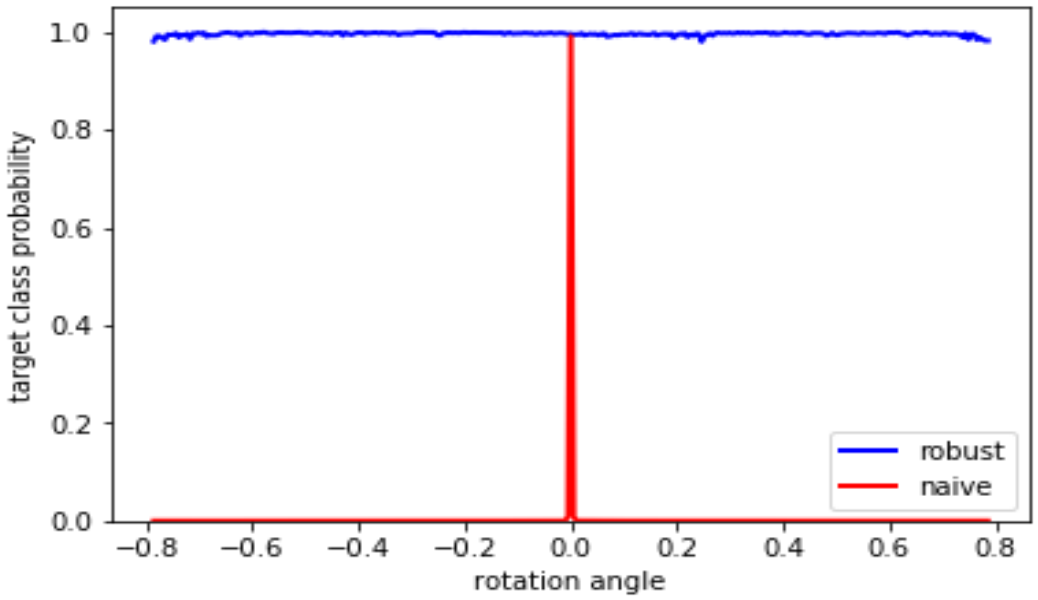

Figure 15. *Rotation-invariance of the adversarial example*

Figure 14 shows that the image remained adversarial throughout the transformation range.

The authors have also been able to generate 3D adversarial examples by modeling the 3D rendering as a transformation under EOT. They optimized the texture of a 3D object such that the result is adversarial from any point of view. They considered a distribution that incorporates different camera distances, lighting conditions, translation and rotation of the object, and solid background colors.

4. SUMMARY AND CONCLUSION

It has been observed that adversarial examples from excessive linearity of models. As many of the modern Neural Networks are piece-wise linear, it will be an easy task for the attacker to synthesize adversarial examples for them. One of the reasons could be, linear models usually assign unusual confidences as we move away from the decision boundary, even if there isn't any data in those regions. There might be data points in those regions which do not belong to that class. There also might be a case where the distance between two samples in a certain subspace is almost negligible but they belong to different classes. As linear models do not consider all these possibilities, they are extremely susceptible to adversarial examples.

As there is a tradeoff between linearity of the model and its vulnerability to adversarial attacks, a lot of work must be done in order to come up with better optimization techniques for non-linear models, which may help in reducing the vulnerability of models to adversarial attacks.

As mentioned before, sometimes the model has more confidence in the incorrect prediction than for the true one which indicates how powerful adversarial examples can be in challenging the robustness and reliability of the model. Even though the adversary might not be well aware of the model and its underlying parameters, Cross Model Generalization for Black Box Attacks makes it highly possible for the attacker to generate adversarial examples for unknown model architectures as well.

Not only images but researchers have shown that adversarial examples can be generated for Natural Language Processing models too. In an example of Speech to Text model, changing two words completely changed the sentiment of the sentence.

As machine learning is integrated into more and more systems, such as autonomous vehicles or medical devices, they are also becoming entry points for attacks. Although may defense techniques such as

Adversarial Training, Autoencoders, Ensemble methods and more have been proposed by researchers, there is no method that has been accepted universally, yet.

Adversarial attacks pose a very real threat to AI security. Ultimately, while it's important to look forward towards the future of AI development, we always need to be aware of the potential problems it brings.

REFERENCES

[1] Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In Proceedings of the International Conference on Learning Representations (ICLR), 2015. URL https://arxiv.org/abs/1412.6572

[2] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. URL https://arxiv.org/abs/1707.07397

[3] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, d., Goodfellow, I., Fergus, R. Intriguing properties of neural networks. URL https://arxiv.org/abs/1312.6199

[4] Tricking Neural Networks: Create your own Adversarial Examples by Daniel Geng and Rishi Veerapaneni.

[5] Interpretable Machine Learning by Christoph Molnar. URL https://christophm.github.io/interpretable-ml-book/adversarial.html

[6] URL https://www.tensorflow.org/get_started/mnist/pros