Robustness Metrics for Evasion Attacks on Neural Networks

Miranda Overstreet
Department of Computer and Information Science and Engineering
University of Florida, USA

***Abstract*—We use a wide variety of IoT (Internet of Things) devices to perform our daily activities. IoT devices powered by neural networks play an increasingly important role in enabling smart and enjoyable interactions with the computing world. Neural networks are often used to classify data, such as text or image, so that a machine can make a decision based on this classification. With the growing importance and application of neural networks, the security of this technology is an increasing concern. This is especially true in safety-critical systems, such as in biomedical applications or autonomous vehicles, where the correct functioning of neural networks is critical. The field of adversarial machine learning has led to many promising ideas in developing robust neural networks. Unfortunately, there are very limited studies in evaluating the robustness of machine learning models. This thesis investigates suitable metrics for evaluating the robustness of neural networks to evasion attacks. Specifically, I explore fast heuristics, simulation, and Lipschitz continuity to develop a useful robustness metric. Experimental results demonstrate that the proposed metrics are suitable for popular machine learning data sets.**

I. Introduction

Many innovations in technology have led to the rise of the "Internet of Things" (IoT) which refers to the pervasive presence of interconnected technologies in every day life [1]. Today, we have not only smart phones and laptops, but also smart refrigerators and self-driving cars. With this increase of smart devices powered by machine learning, there is also a growing security concern as more and more attack surfaces are introduced [1].

One area of particular concern within IoT is the increase in self-driving cars. Many such cars are now under development and many more already possess full or partial self-driving abilities. These cars depend on images that are captured of the world around them and the software that makes decisions based on these images. Many autonomous driving systems rely on neural networks to support this computer vision and pattern recognition [2]. If the system gets infected by an attacker, it could have catastrophic consequences. Similarly, many neural networks are now being used to assist medical practitioners in the diagnosis of illnesses based on images [3]. Here also an adversarial system could cause devastating results by suggesting an incorrect diagnosis and causing a delay in the correct treatment of a patient. Clearly, there is a need for improved security for defending against such attacks on neural networks.

This security risk has not gone unnoticed by the machine learning community. Many attacks on neural networks have been explored and many defense mechanisms have been developed. However, identification of insecure neural networks still remains underdeveloped. The focus of this thesis is to develop effective metrics to measure the robustness of neural networks against evasion attacks. An evasion attack happens when a neural network is fed with an "adversarial" example — a carefully perturbed input that looks and feels exactly the same as its untampered copy to a human but that leads to incorrect classification. In this honors thesis, I make the following contributions:

1) Develop three metrics to measure the robustness of neural networks to evasion attacks. These metrics provide a trade-off between evaluation time and accuracy.
2) Train neural networks on two popular data sets
3) Evaluate the effectiveness of the developed metrics using popular evasion attacks

The remainder of this thesis is organized as follows. Section II provides an overview of neural networks and adversarial machine learning. Section III describes related work in the area of measuring the robustness of neural networks. Section IV discusses the three metrics that I have developed. Section V presents the experimental results. Finally, Section VI concludes the thesis and discusses future work.

II. Background

Neural networks are a form of supervised machine learning which are often used for classification tasks. One of the most common classification tasks that use neural networks is the classification of images. Many neural networks have been shown to very successfully classify images in labeled data sets such as MNIST [4] and CIFAR-10 [5]. However, it has also been shown that many of these networks suffer from a vulnerability to evasion attacks. In this section, I will discuss neural networks, adversarial machine learning, and evasion attacks for neural networks.

Neuron: A neural network consists of layers of neurons. Each neuron in a neural network has an activation function. This activation function takes the various inputs for that neuron and returns some output based on these inputs. Figure 1 shows an example of a single neuron in a neural network. Each input x_i is typically multiplied by some weight w_i and then a bias b_i is added to this value. These values are then summed and then the activation function f is applied to this summed value as shown in Figure 1. Typical examples of activation functions are Sigmoid [6], Tanh [7], and ReLU [8]. All three of these activation functions add non linearity to the network.

Neural Network: A neural network's structure contains several kinds of layers mainly an input layer, hidden layers, and an output layer. In the case of images, the pixels of the

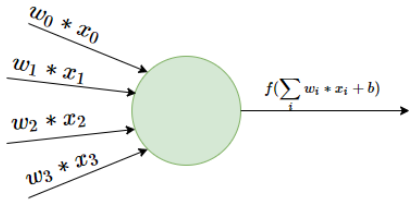

Fig. 1: An illustrative example of a single neuron

image are the values for the input layer. The values from the input layers are sent to one or more hidden layers. Each hidden layer includes a certain number of neurons that apply some activation function to the input values and output the result of the activation function to the next layer. Finally, the output layer contains the final classification values usually ranging from 0 to 1 for each class in the labeled data set. A simple example of a neural network is shown in Figure 2. This neural network has an input layer, one hidden fully connected layer, and an output layer.

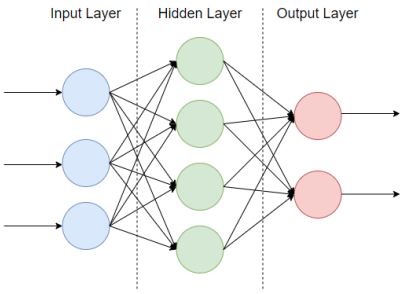

Fig. 2: An example neural network with three layers

Convolutional Neural Network: Neural networks have several variations that are more effective on specific domains. In the case of images, a convolutional neural network is often used for classification. A convolutional neural network introduces a new kind of hidden layer called a convolutional layer. A convolutional layer consists of a set of filters with some width and height of pixels and depth of color channels. These filters are convolved across an image and the dot products are calculated between the filters and the section of image. This process can be seen in Figure 3. In this way, larger features can be identified. For example, a filter that identifies the triangular shape of a dog's ear might be convolved across the image to identify if a picture contains a dog or not.

Adversarial Machine Learning: With the rise of neural networks in technology there has been a growing concern of the security of neural networks. Many attacks on neural networks have been demonstrated to have high effectiveness [10]–[15]. A type of attack that has been shown to be particularly viable is an evasion attack [16]. The goal of an evasion attack is to change an input image to a network some

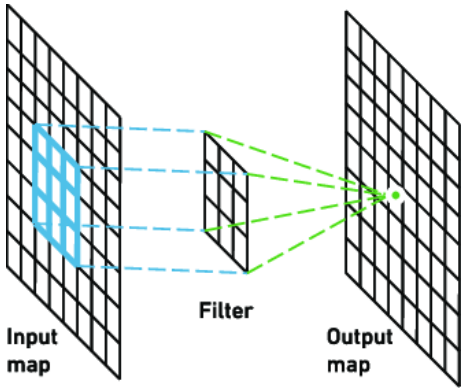

Fig. 3: An example of a convolutional layer [9]

amount in order to cause the network to classify the image incorrectly. This is often done so that the perturbation in the image is not noticeable to the human eye. These attacks often use the gradients of an output of a neural network to determine what pixels to perturb so that the smallest changes in pixel values can cause the largest changes in the classification. One example of an evasion algorithm that does this is the Fast Gradient Sign Method [13]. This method uses the following equation to develop an adversarial image.

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y))$$

In this equation, θ is the parameters of the model, x is the input image to the model, and y is the output labels. $J(\theta, x, y)$ is the cost function used during the training of the neural network, and $\nabla_x J(\theta, x, y)$ is the gradient of this cost function given the input image. The gradient of this cost function can be easily calculated using backpropagation, an already necessary step in network training. An example of an adversarial image created using this method for an evasion attack is shown in Figure 4.

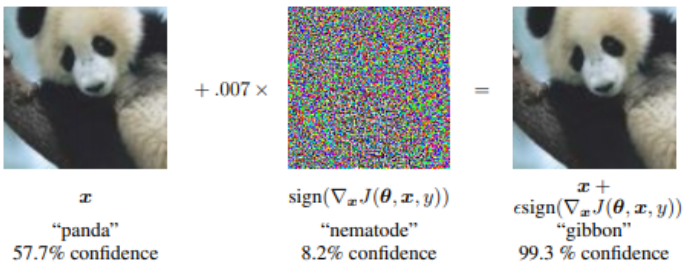

Fig. 4: An example of an adversarial image for an evasion attack using Fast Gradient Sign Method [13]

From this attack method, it can be seen that a network with higher gradients at any given point will be more susceptible to an attack and that any attacks for such a network would be hard to identify. There are several methods that have been developed to counter such evasion attacks. One method is simply training a neural network on adversarial images that are generated. For example, in the case of Figure 4, one could retrain the network with the second picture of a panda labeled as a panda added to the labeled data set. This has been shown

to be successful in reducing the effectiveness of evasion attacks for a given network [11], [13], [16]. Another method is to change the structure of the network itself by doing something like smoothing out the gradients [17]–[19].

III. Related Work

This section discusses research work that suggests a method of evaluating neural networks in terms of robustness. I will discuss two different kinds of evaluation methods: attack-specific and attack-agnostic.

A. Attack-Specific Robustness Evaluation

Some methods evaluates the robustness of a neural network by evaluating it against a specific evasion attack [11], [13]. An example of this is presented in [11] where the DeepFool attack is used to create a robustness metric [11]. In this case, the metric for the average robustness $\hat{p}_{adv}(f)$ of a classifier f is defined by the following equation:

$$\hat{p}_{adv}(f) = \frac{1}{|D|} \sum_{x \epsilon D} \frac{||\hat{r}(x)||_2}{||x||_2}$$

where $\hat{r}(x)$ is the minimal perturbation calculated by DeepFool and D is the test set [11]. Although attack-specific metrics are a good estimate of the robustness of a system to a specific attack, they might not extend well to other attacks [20]. In this case, an attack-agnostic method would be preferred.

B. Attack-Agnostic Robustness Evaluation

Some methods for evaluation the robustness of a neural network rely solely on the network itself without using a specific attack [20], [21], and therefore, might be more representative to overall robustness of a neural network. For example, [20] suggests a method that uses the encoding of robustness as a linear program. This is achievable through the piecewise linearity of neural networks with rectified-linear units. The metrics that I propose in this thesis are also attack-agnostic.

IV. Proposed Robustness Metrics

As discussed in the previous section, the gradient is often used in evasion attacks to find the optimal directions to perturb an image. In order to develop a metric for the robustness of a neural network, I decided to use measurements on the neural network to reflect the amount of change in the the classification given a change in the input image. In this section, I discuss three proposed metrics to measure robustness: heuristic, simulation method, and Lipschitz method.

A. Heuristic

The heuristic is the simplest method of evaluating the robustness of a neural network against evasion attacks. The heuristic measures the magnitude of the gradient for each pixel value in a given image. When taken with a comprehensive subsection of images, this can provide a good robustness estimate about a network.

Because the equations involved in a neural network are differentiable, backpropagation can be used to find the gradients of the classification values given the input values. These gradients can then be used to test the robustness of a neural network. The formula for the heuristic is defined as:

$$h = \left\| \frac{df}{dx} \right\|$$

Here, the f is the final output of the neural network, x is an image, and $\frac{df}{dx}$ is the gradient of the output given a particular image.

B. Simulation Method

The simulation method requires significantly more computation than the heuristic. However, it gives a better idea of the values of the neighborhood around the picture in question. Given some picture x, new images x' are generated within a certain pixel distance. The classification is then calculated for this x', and the slope is found with the change of the classification value over the change in pixel value. The highest slope value is kept as the value for the simulation method. Algorithm 1 outlines the major steps.

Algorithm 1 Simulation Method

```
metric = 0
for each image x do
    for each image sample x' do
        x' = x
        for each pixel in x' do
            sample random pixel value
        end for
        Compute f(x') where f is network
        Compute ||Δf(x)||/||Δx||
        metric = maximum(metric, ||Δf(x)||/||Δx||)
    end for
end for
```

This method can provide a good idea of robustness given a comprehensive subsection of images.

C. Lipschitz Method

The Lipschitz method approximates the Lipschitz constant for a given image. The Lipschitz continuity is denoted by the following equation where f is some function, x is some input, x' is some input sufficiently close to x, and L is a constant:

$$\|f(x') - f(x)\| \leq L \|x' - x\|$$

The Lipschitz constant L is the smallest constant such that the above equation holds. Let C_n be some output node in a neural network. Then the Lipschitz continuity for this output node where the input are the values for the inputs I for some image, J can be expressed as:

$$\|C_n(I_{J'}) - C_n(I_J)\| \leq L_n \|I'_J - I_J\|$$

Let the image inputs I_J be the pixels of that image $p_1, p_2, p_3, ..., p_q$. For computations in neural networks, the

partial derivative for the output given any input p_i can be calculated as $\frac{\delta C_n}{\delta p_i}$. Let $e_i = max_{p_i}\{C_n * \frac{\delta C_n}{\delta p_i}\}$. Assuming that the second derivative for C_n exists, the change in C_n can be written in terms of e_i using the maxima of the partial derivatives, the Mean Value Theorem and the Triangle Inequality.

$$
\begin{aligned}
&|C_n(p_1, p_2, p_3) - C_n(p'_1, p'_2, p'_3)| \\
&= |C_n(p_1, p_2, p_3) - C_n(p'_1, p_2, p_3) + C_n(p'_1, p_2, p_3) \\
&-C_n(p'_1, p'_2, p_3) + C_n(p'_1, p'_2, p_3) - C_n(p'_1, p'_2, p'_3)| \\
&\leq |C_n(p_1, p_2, p_3) - C_n(p'_1, p_2, p_3)| + |C_n(p'_1, p_2, p_3) \\
&-C_n(p'_1, p'_2, p_3)| + |C_n(v'_1, p'_2, p_3) - C_n(p'_1, p'_2, p'_3)| \\
&\leq (max_{p1} \frac{\delta C_n}{\delta p_1} C_n(p''_1, p_2, p_3))|p'_1 - p_1| + \\
&(max_{p2} \frac{\delta C_n}{\delta p_2} C_n(p'_1, p''_2, p_3))|p'_2 - p_2| + \\
&(max_{p3} \frac{\delta C_n}{\delta p_3} C_n(p'_1, p'_2, p''_3))|p'_3 - p_3| \\
&\leq e_1|p'_1 - p_1| + e_2|p'_2 - p_2| + e_3|p'_3 - p_3| \\
&\leq \sqrt{e_1^2 + e_2^2 + e_3^2}|(p'_1, p'_2, p'_3) - (p_1, p_2, p_3)|
\end{aligned}
$$

Therefore, the Lipschitz constant is bounded by the sum of the squares of the maxima of the product of the partial derivatives and the function C_n for a given neuron. This process can be repeated for each neuron so that the entire output is bounded by the sum of the Lipschitz constant for each neuron.

In order to calculate this bound, the maximum value of $\frac{\delta C_n}{\delta p_i}$ must be found. This value is estimated by sampling around the given image for pixel p_i within a given range and finding the pixel value for the highest value of $\frac{\delta C_n}{\delta p_i}$. The method for estimating the Lipschitz constant is shown in Algorithm 2.

Algorithm 2 Lipschitz Continuity

```
for each image x do
    metric = 0
    for each node C_n in network do
        total = 0
        for each pixel p_i in x do
            max = 0
            for p'_i in pixel range do
                Find δC_n/δp_i where C_n is network
                max = maximum(max, δC_n/δp_i) * C_n
            end for
            total = total + max^2
        end for
        metric = metric + sqrt(total)
    end for
end for
```

D. Complexity Analysis

These three metrics provide a clear trade-off. If a designer is interested in finding the robustness value of a neural network as quickly as possible, the *heuristic* method is the choice because it has the lowest computation time. However, it has experimentally been shown to lead to inferior accuracy compared to the other two metrics. If a designer is interested in the highest accuracy and willing to spend the required evaluation time, the *Lipschitz* method is the best choice. It has been shown through my experiments to have both the highest accuracy and the highest computation time for most cases. The *simulation* based method provides moderate accuracy with an evaluation time between heuristic and simulation method.

V. Experiments

This section describes various experiments that were run to test the effectiveness of the three proposed metrics developed as indicators of the robustness of neural networks. The section is organized as follows. First, I describe the experimental setup including the architecture used to run the experiments and the software used to develop the experiments. Next, I present the results of the experiments using two data sets and discuss their significance.

A. Experimental Setup

The following experiments were performed on a host machine with AMD Ryzen 5 1600X Six-Core Processor 3.60 GHz CPU, 8.00 GB RAM, and NVIDIA 1070 256-bit GPU. The operating system used was a Windows 10 64-bit operating system. I developed code using Python for model training using Keras as the machine learning library. The attacks were generated using the *Adversarial-Machine-Learning* repository [22]. Two popular machine learning data sets were used in this experimental setup:

- **CIFAR-10** - A data set of 60,000 32x32 pixel color images with 10 classes [23].
- **MNIST** - A data set of 70,000 28x28 pixel grey scale images of hand drawn digits [24].

For each data set the following steps were followed to evaluate the robustness metrics:

1) Train a neural network on the data set for 10 epochs
2) Generate adversarial samples and test accuracy of these samples using three attacks: Fast Gradient Method [13], DeepFool [11], and NewtonFool [15]
3) Retrain three new networks using the adversarial samples generated in the previous step
4) Generate adversarial samples for three new networks and test network accuracy on adversarial samples
5) Calculate heuristic, simulation-based metric, and Lipschitz metric for original network and for each retrained network
6) Calculate the correlation coefficient between the success of the attacks and the robustness metrics

For step 1, the data set was split into a training and testing data set. For CIFAR-10 50,000 images were used for the training data set and 10,000 images were used for the testing set. For MNIST, 60,000 images were used for the trainind data set and 10,000 images were used for the testing set. Let the original training data be referred to as x_{train} and the original testing data be referred to as x_{test}. A network built with Keras was then trained on these data sets.

For step 2, the three attacks described were run on the trained network. Each of these attacks generated a set of adversarial examples for x_{train} and x_{test}. Let these adversarial

examples for each attack be denoted as $x_{train-adversarial}$ and $x_{test-adversarial}$. I then had the network classify $x_{test-adversarial}$ and recorded the accuracy of the network for each attack.

For step 3, I retrained three new networks with the adversarial samples generated by the three attack methods so that the new training data set for each was $x_{train} + x_{train-adversarial}$. Therefore, the first network was trained on the original training set plus the adversarial samples generated by the Fast Gradient Method [13], the second was trained on the original training set plus the adversarial samples generated by the DeepFool [11] and so on. These new networks were trained in an identical way to the original training.

For step 4, I generated new adversarial samples for x_{test} using the three new networks and the same three attacks. I then find the accuracy of these three networks on the new adversarial samples.

For step 5, I calculated the values for the heuristic, the simulation method, and the Lipschitz method for the original network and the three new networks trained on $x_{train-adversarial}$ for each attack method. In order to do this, I had to choose some subsection of images to develop the metrics. I simply used the first 500 images in the test set for the data set for both the heuristic and the simulation method. However, due to the high computation time of the Lipschitz method I only used the first 45 images from the test set for the Lipschitz method. Future work could be done to ensure a comprehensive selection of images are selected for these three metrics. The heuristic was easily calculated for each of the images. The values for the simulation and Lipschitz method, however, required designation of the parameters involved. For the simulation, I decided to sample 100 perturbed images around the given input image and used a maximum pixel distance of .01 (out of 1). For the Lipschitz method I decided to use a pixel range of .01. Once again, further work could be done in this area in order to optimize the parameters for these methods.

For step 6, I calculated the correlation coefficient between the success of the attacks on a given network and the three metrics. The original network that is trained on a data set should be the least robust because the other three networks have the adversarial examples generated by the Fast Gradient Method [13], DeepFool [11], and NewtonFool [15] attacks as part of their training data. Training on adversarial samples has been shown to substantially increase the robustness of a neural network [11], [13], [16]. This difference in robustness should be observable in the success of the attacks. If the metrics are also a good indicator of robustness, then there should be a strong correlation between the metrics and the success of the attacks on the neural networks. Given two data sets X and Y with data points x and y and averages $\bar{x}$ and $\bar{y}$, the correlation coefficient between the two data sets is calculated as follows:

$$Correlation(X,Y) = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\sqrt{\Sigma(x-\bar{x})^2\Sigma(y-\bar{y})^2}}$$

B. CIFAR-10 Results

In this section, I describe the experimental results for the CIFAR-10 data set. As can be seen in Figure 5, the success of all three attacks was much higher on the original neural network that was trained on the CIFAR-10 data set than on the three retrained networks. This is expected because training with adversarial examples increases the robustness of a neural network.

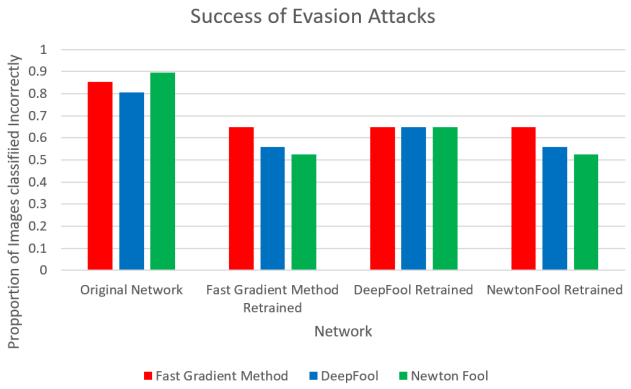

Fig. 5: The success of the three attacks on the original and retrained networks

It can be seen from Figure 6 that the heuristic value follows the same pattern as the success of the evasion attacks. The highest heuristic value is for the original network trained on x_{train}. The heuristic values for the three retrained networks are noticeably smaller than the original heuristic value. This suggests that the heuristic is indicating the increase in robustness after the retraining.

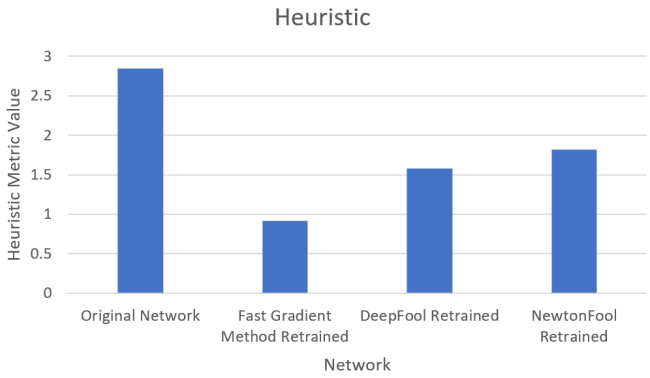

Fig. 6: Resulting metric values for the heuristic

It can be seen from Figure 7 that the simulation value follows the same pattern as the success of the evasion attacks. The highest simulation value is for the original network trained on x_{train}. The simulation values for the three retrained networks are noticeably smaller than the original simulation value. This suggests that the simulation is also indicating the increase in robustness after the retraining.

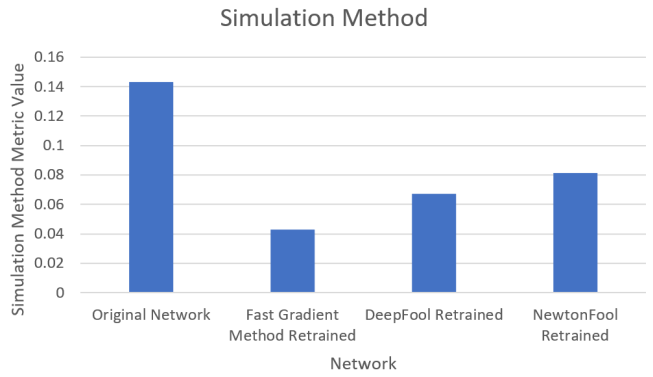

Fig. 7: Resulting metric values for the simulation method

It can be seen in Figure 8 that the Lipschitz method metric follows the same pattern as the heuristic and simulation method values. It has the highest value for the original network and lower values for the retrained networks. This demonstrates that the Lipschitz method is also a suitable metric for measuring robustness.

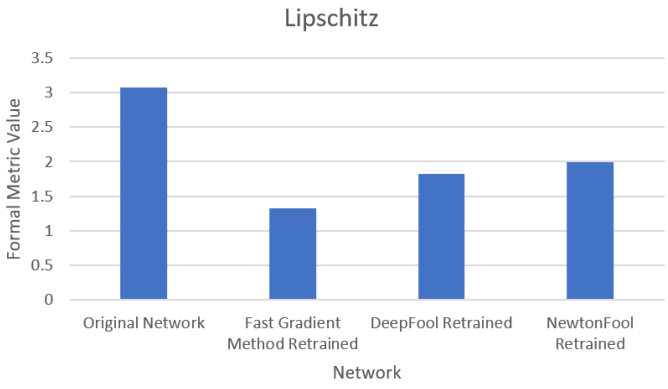

Fig. 8: Resulting metric values for the Lipschitz method

As discussed previously, a high correlation between the success of the attacks on a given network and the metrics developed in this paper suggests that the metrics developed are a good indicator of the robustness of a neural network. Figure 9 shows the correlation coefficient between the success of each attack and the metrics. This figure shows that the correlation coefficient suggests a strong positive linear relationship between the success of the attacks and the metrics. The correlation coefficient for the Lipschitz method is the highest followed closely by the simulation method and then the heuristic for most cases. This is as was expected for the three methods. However, for the first attack simulation had the higher correlation. One reason for this might be that the Lipschitz method was evaluated on less of a sample of images. If more images were tested, the Lipschitz method would likely have the highest correlation for all three attacks. The correlation for the heuristic is less for all three metrics but still relatively high. This suggests that all three metrics are good indicators of robustness.

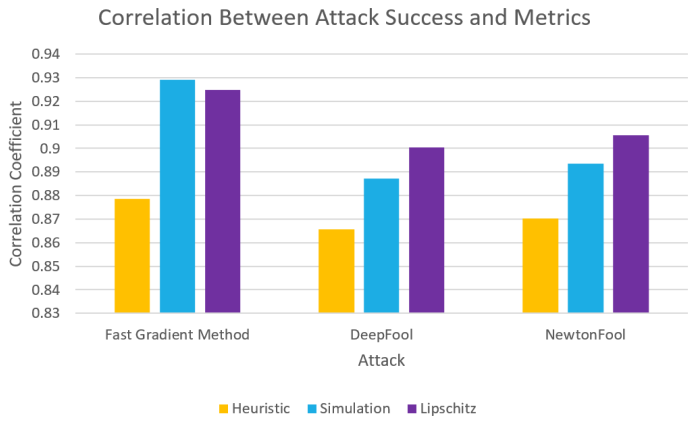

Fig. 9: Correlation Coefficients Between the Success of the Attacks and the Metrics

C. MNIST Results

In this section, I describe the experimental results for the MNIST data set. As can be seen in Figure 10, the success of all three attacks was much higher on the original neural network that was trained on the MNIST data set than on the three retrained networks. This follows the same trend as the CIFAR-10 data set.

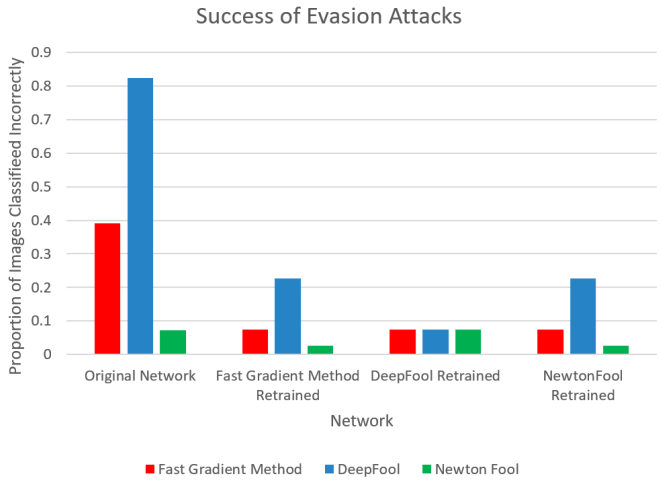

Fig. 10: The success of the three attacks on the original and retrained networks

The results for the three metrics are similar to the results for CIFAR-10. It can be seen that for all three metrics the highest value is for the original network trained on x_{train}. This suggests that all three metrics are reasonable indicators of robustness.

Figure 14 shows the correlation coefficient between the success of each attack and the metrics. This figure shows

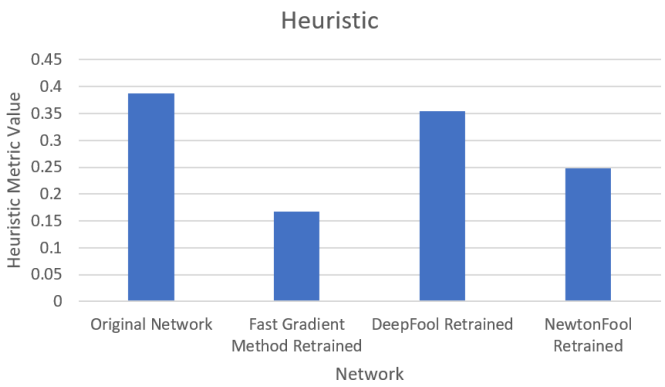

Fig. 11: Resulting metric values for the heuristic

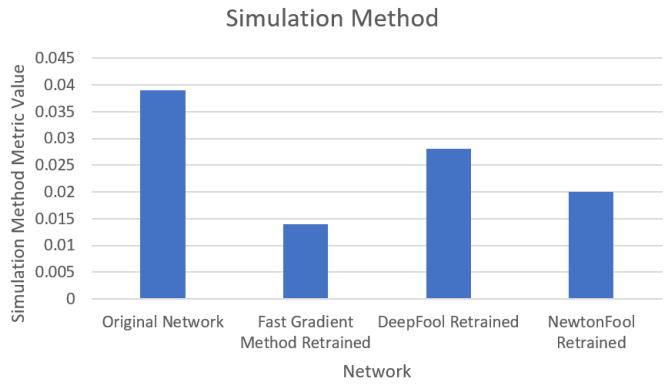

Fig. 12: Resulting metric values for the simulation method

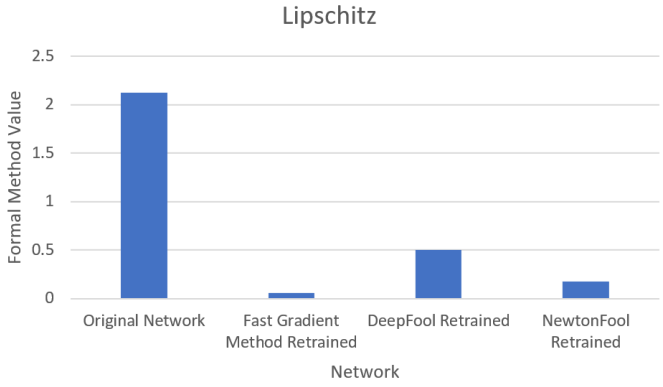

Fig. 13: Resulting metric values for the Lipschitz method

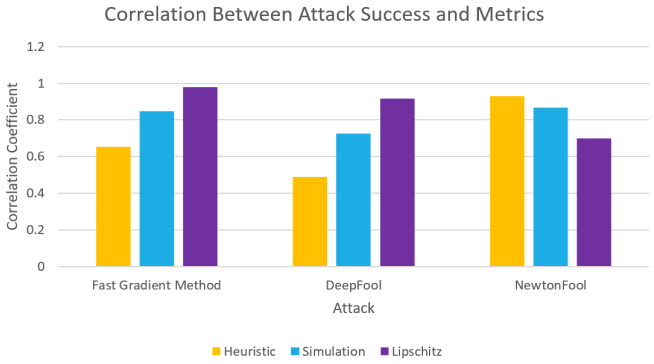

Fig. 14: Correlation coefficients between the success of the attacks and the metrics

that the correlation coefficient suggests a strong positive linear relationship between the success of the attacks and the metrics. For the first two networks the Lipschitz method performed better than the simulation method and the heuristic. This pattern is consistent with the previous data set and the expected results. The last, however, was inverse. This is a somewhat surprising result because it seems like the Lipschitz method and simulation method should perform better than the heuristic in all cases. This result might be an incentive for further research to discover the cause of this inconsistency.

VI. Conclusion

Neural networks are widely used today for designing smart systems. Due to their widespread use, an important emerging concern is to secure the neural networks from adversarial attacks. The field of adversarial machine learning has led to many promising ideas in developing robust neural networks. Unfortunately, there are very limited studies in evaluating the robustness of machine learning models. This thesis investigated suitable metrics for evaluating the robustness of neural networks to evasion attacks. Specifically, I explored fast heuristics, simulation, and Lipschitz methods to develop useful robustness metrics. I have performed a detailed evaluation of accuracy of these metrics on two popular machine learning data sets.

The results from the three metrics are promising in their ability to judge the robustness of a neural network to an evasion attack. This is because all three metrics have a relatively high correlation coefficient to the success of the attacks on a given network. The simulation method provides the best metric out of the three based on the experimental data because it seems to have the best trade off between accuracy and computational complexity. The simulation method results had a strong correlation while remaining relatively cheap to compute. The Lipschitz method seemed to be the most accurate as it had the highest correlation metrics with a limited data set (only 45 images). However, it had a very high computational complexity that would most likely be prohibitive in most circumstances. In cases where very low computational complexity is desired, the heuristic would likely be most desirable because it is much cheaper computationally than the simulation method and Lipschitz method while still providing a good approximation of the robustness. This method would most likely be best used to give a quick approximation of the robustness of a neural network.

Future work can refine the individual metrics to be more accurate in their measure of neural network robustness. Given

the computationally expensive nature of Lipschitz method, more work is needed in refining the Lipschitz method and reducing its computational complexity. Future work could also be done by experimenting in incorporating these metrics into the training of neural networks. This could help improve the robustness of the neural network during the initial training.

REFERENCES

[1] G. M. Luigi Atzori, Antonio Lera, "The internet of things: A survey," *Computer Networks*, vol. 54, 2010.

[2] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. J. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *CoRR*, vol. abs/1902.06705, 2019. [Online]. Available: http://arxiv.org/abs/1902.06705

[3] "Application of neural networks in medicine - a review," 1998.

[4] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, 2012.

[5] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *14th Internation Conference on Artificial Intelligence and Statistics*, 2011.

[6] N. Kang, "Multi-layer neural networks with sigmoid function - deep learning for rookies (2)," Available at https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f (2017/06/27).

[7] S. I. Serengil, "Hyperbolic tangent as neural network activation function," Available at https://sefiks.com/2017/01/29/hyperbolic-tangent-as-neural-network-activation-function/ (2017/01/29).

[8] J. Brownlee, "A gentle introduction to the rectified linear unit (relu," Available at https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/ (2019/01/09).

[9] H. Yakura, S. Shinozaki, R. Nishimura, Y. Oyama, and J. Sakuma, "Malware analysis of imaged binary samples by convolutional neural network with attention mechanism," in *The 8th ACM Conference on Data and Application Security and Privacy*, 2018.

[10] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," *CoRR*, vol. abs/1608.04644, 2016.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," *CVPR*, 11 2016.

[12] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations*, 2015.

[14] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations*, 2017.

[15] "Objective metrics and gradient descent algorithms for adversarial examples in machine learning," in *33rd Annual Computer Security Application Conference*, 2017.

[16] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations*, 2014.

[17] Y. Yoshida and T. Miyato, "Spectral norm regularization for improving the generalizability of deep learning," *CoRR*, vol. abs/1705.10941, 2017.

[18] "A unified gradient regulatization family for adversarial examples," in *IEE International Conference on Data Mining*, 2015.

[19] "Improcing the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients," in *AAI*, 2018.

[20] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. V. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," in *30th Conference on Neural Information Processing Systems (NIPS)*, 2016.

[21] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," in *6th International Conference on Learning Representation*, 2018.

[22] "Adversarial robustness toolbox." [Online]. Available: https://github.com/IBM/adversarial-robustness-toolbox

[23] "The cifar-10 dataset." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[24] "The mnist database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/