Adversarial Attacks on Artificial Neural Networks: Analysis of Threats and Possible Countermeasures

Nicholaus Danispadmanaba Y – 18217028
Information System and Technology – Bandung Institute of Technology

Abstract - Seamless integration of image recognition, natural language processing, and other predictive capabilities into both mobile and web-based applications has been made possible with various forms of neural networks. Neural networks are able to generate an approximation of any functions given specific input ranges without needing humans to directly manipulate the function itself. Hence, by just providing input data within that specific input ranges, the neural networks would be able to generate a result that is close to the intended output with an error margin (i.e. supervised learning) or even categorize data on its own without needing labeled training data (i.e. unsupervised learning). Those features enable us to create a great variety of products previously deemed to be too computationally complex, e.g. personal AI assistant, self-driving car, and even automating some aspects of the development of cure for certain diseases. With society increasingly reliant on technologies powered by neural networks, attacks against them would become costly and even potentially life-threatening for humans. One method commonly used to attack neural networks is adversarial attacks. By introducing strategically placed noise on the appropriate input data, the network would generate a highly erroneous result. One example would be just how by adding certain sticker beside a banana, an image recognition model would classify the banana as a toaster! Since adversarial attacks are often computationally simple, we should expect an increase on the amount of adversarial attacks on established neural networks. Therefore, countermeasures must be devised to thwart such attacks, especially on neural network models used for mission critical systems (e.g. fraud detection on a transaction system) or systems directly related to human life (e.g. self-driving car). Therefore, this paper will explore various types of adversarial attacks. After that, various countermeasure methods will also be explored.

Index Terms - neural networks, adversarial attacks, model, noise, countermeasure

I. Introduction

I.1. Background

Machine learning along with one of its subsets, neural networks, are a staple in modern consumer products, ranging from recommendations in video-sharing web application to self-driving cars. One of the causes of the proliferation of machine learning implementation is the doubling of the number of transistors in integrated circuit [1]. Another cause would be the availability of hardware accelerators (e.g. GPU) that support computations needed by machine learning to process data and generate its model [2]. As integrated circuits become more densely packed with transistors and hardware accelerators become more readily available, the computation cost of machine learning algorithms becomes more attainable by consumer-grade hardware while its computation time is markedly reduced. Hence, access to machine learning implementation is no longer restricted for products used in academia or big enterprise but also available for products sold to the general public.

Accessibility of products containing machine learning implementation to the general public means that machine learning engineers must ensure that their models are able to fulfill consumer trust and safety through highly accurate prediction. Accuracy of prediction is especially important in mission-critical systems, e.g., credit card fraud detection, self-driving car, and medical diagnosis. Even a small amount of error in prediction generated by those mission-critical systems could potentially lead to the loss of consumer trust and life.

Unfortunately, increased exposure of systems with machine learning components to the general public means that the number of potential attackers that could compromise those systems have also increased. One of the methods that could be used by potential attackers is adversarial attack. This method of attack utilizes modification of input data by injecting noise into the aforementioned input data. The introduction of noise would then cause the model to generate a highly erroneous prediction. One example would be the classification of an image of a panda with and without noise. The model would classify the image of a panda without noise correctly, but when noise is introduced, the image classifier would classify the image as that of a gibbon [3].

Since accuracy of prediction correlates with customer trust and safety while potential attackers along with methods of attack are readily available, a set of countermeasures must be developed by machine learning engineers. Hence, this technical report would initially focus on analyzing adversarial attacks through exploration of different methods used to execute adversarial attacks and discussing mechanism of each method of adversarial attacks. After that,

this technical report would discuss about countermeasures that are available against those adversarial attacks.

I.2. Problem Identification

Below is the list of problems that would be discussed in this technical report:

1. What are the methods used to execute adversarial attacks?
2. What are the available countermeasures against adversarial attacks?

I.3. Objectives

Below is the list of objectives of this technical report:

1. To explore methods used to execute adversarial attacks.
2. To explore available countermeasures against adversarial attacks.

I.4. Methodology

This technical report is divided into four main parts that are consisted of: introduction, literature review, discussion, and conclusion and recommendations. This table would describe the overview of each of those parts.

Table 1 Methodology

No.	Chapter	Purpose	Sections
1.	Introduction	This chapter explains the motivation behind the creation of this technical report and things that this report is trying to accomplish.	a. Background b. Problem Identification c. Objectives d. Methodology
2.	Literature Review	This chapter describes the main theoretical basis for developing the contents of this technical reports.	a. Machine Learning b. Artificial Neural Network c. Adversarial Attack
3.	Discussion	This chapter is the main content of this technical report, which explains each of the points laid out in both Section I.2. and I.3.	a. Methods of Adversarial Attacks b. Countermeasures Against Adversarial Attacks

4.	Conclusion and Recommendations	This chapter concludes the discussion of the report and gives recommendation for future research in related topics.	a. Conclusion b. Recommendations

II. Literature Review

II.1. Machine Learning

Machine learning is a class of algorithms that are able to generate a set of outputs without explicit step-by-step instruction or functions defined by human programmers; i.e., an algorithm that could autonomously improve itself. This class of algorithms relies on sample data, commonly called training data, to achieve autonomous improvement by continuously generating an approximation of mathematical functions / models that are applicable to the sample data [4]. Algorithms used in machine learning could be divided into several main approaches, which consist of: supervised learning, unsupervised learning, and semi-supervised learning.

Supervised learning is a machine learning algorithm approach that uses training data which contains both inputs and the correct outputs that correspond to their respective inputs. The correct outputs of each inputs are defined by humans that create the machine learning model. Difference between inputs and outputs in the training data is identified with the use of labels. Each iteration of models generated during the training phase would be evaluated and modified based on the margin of error between the set of predicted outputs and the set of correct outputs given a set of inputs from the training data. Learning algorithms that could be categorized as supervised learning include Support Vector Machines, Decision Trees, and Neural Networks (Multilayer Perceptron) [5].

Unsupervised learning is a machine learning algorithm approach that is the exact opposite of supervised learning, in which it is fed with input data that does not contain a set of correct outputs corresponding to a set of inputs. Instead, unsupervised learning only relies on input data that is unlabeled to produce models that in turn would produce predictions. Therefore, unsupervised learning does not go through training phase at all. The advantage of unsupervised learning over supervised learning is that it is able to detect patterns that are previously undetected by human analysts within a set of input data. Learning algorithms that

could be categorized as unsupervised learning include Neural Networks (Autoencoders, Generative Adversarial Networks), Anomaly Detection, and Clustering [6].

Semi-supervised learning is a machine learning algorithm approach that combines methods of both supervised and unsupervised learning. It uses a set of input data that is partially mapped to a set of output data. Therefore, this approach makes use of training phase but with partially labeled training data. This approach is useful for modelling systems in which generating a full mapping between input and output data is costly or unfeasible. Example of such system is generating 3D structure of a protein based on its smaller components to generate correct outputs in training data. Therefore, this approach could detect previously undetected patterns while at the same time retaining the high accuracy of supervised learning. Learning algorithms that could be categorized as semi-supervised learning include Generative Models, Low-density Separation, and Graph-based Methods [7].

The aforementioned self-improving property of machine learning has led to various application of machine learning in fields where the development of conventional step-by-step instruction or functions in algorithms are deemed impossible or too difficult to implement. Examples of those fields include natural language processing [8], image classification [9], fraud detection [10], and even intrusion detection in network security [11].

II.2. Artificial Neural Network

Artificial Neural Network is a subset of machine learning that deals with a specific model of machine learning. That specific model is loosely based upon biological neurons present in brains. Broadly speaking, an Artificial Neural Network is composed of an input layer and an output layer. There could be a number of hidden layers located between an input layer and an output layer. Architectures that utilized multiple hidden layers between input layer and output layer are called Deep Neural Network [12]. It should be noted that this report would focus on Deep Neural Network since real-life implementation of ANN is usually based on DNN architecture. Below is an illustration of a Deep Neural Network.

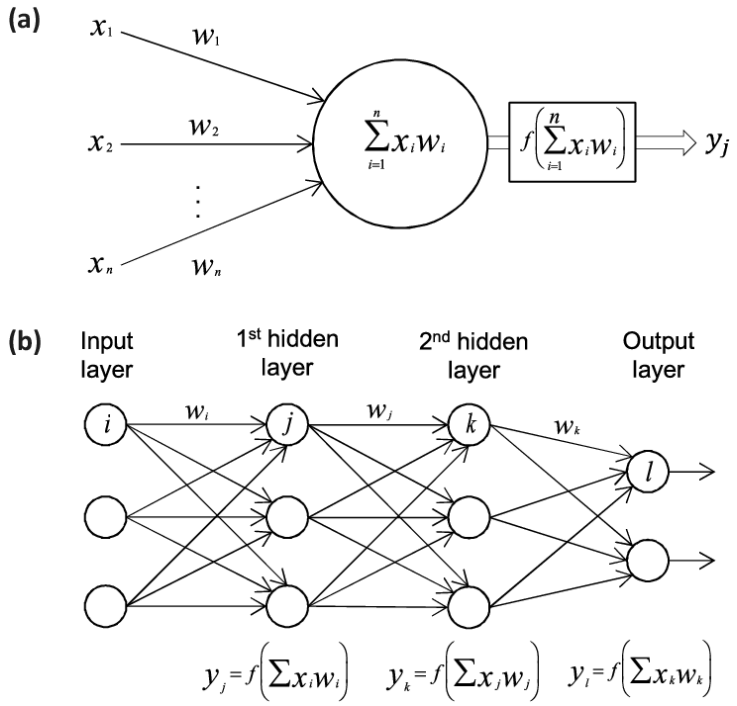

Figure 1Illustration of Deep Neural Network, (a) illustrates a single neuron, (b) illustrates the whole network [13]

Each layer consists of a number of nodes called neurons that are connected to each neuron in the next layer. Every time a number of signal passes through a neuron, the sum of each signal would be calculated with a non-linear function to generate an output of that particular neuron. That function to calculate output of a neuron is called activation function [14]. The connection between neurons in different layers are called edges. Each edge usually has weight that would increase or decrease the strength of signal that passes through it based on the value of that weight. As learning progresses, the value of the weight would be adjusted to increase the accuracy of the output. Adjustment of weight is done with the use of gradients where gradients themselves describe loss, which is the result of error calculation between predicted and correct output, over time. The error calculation is done with the use of functions called loss functions [15]. Gradients should be minimized to discover minimum loss.

II.3. Adversarial Attack

Adversarial attack is the introduction of modified input data that would cause a Machine Learning model, including Deep Neural Network, to produce incorrect prediction. Assume there is unmodified input data D that is fed into L, a ML model, resulting into L(D) = y_{true}. L(D) is the result of the prediction of L based on D, while y_{true} is the correct output. Adversarial attack would generate a modified input data called D', such that L(D') ≠ y_{true}. Differences between unmodified and modified input data are usually imperceptible in

adversarial attacks [16]. Variety of adversarial attacks would be discussed in-depth in Chapter III.

III. Discussion

III.1. Methods of Adversarial Attacks

There is a variety of methods that could be done to execute adversarial attacks on Deep Neural Network systems. Below is the list of methods that would be discussed in this section [17]:

a. Fast gradient sign method
b. One-step target class method
c. Basic iterative method
d. Iterative least-likely class method

Each of the above methods would be discussed in-depth on sub-sections below.

III.1.1. Fast Gradient Sign Method

The basic idea of fast gradient sign method (FGSM) is to generate an adversarial example by utilizing gradient ascent to maximize loss between predicted output and correct output as opposed to minimizing them [18]. It could be said that the approach used in FGSM is the opposite to the one used in training DNN. The objective of FGSM is to cause DNN to misclassify input data. Hence, FGSM is commonly used on DNN used for classification task, such as image recognition. The adversarial example itself is used on pre-trained DNN whether the attacker has access to the model; i.e., white box, or without access; i.e., black box.

Adversarial example mentioned previously is generated by adding perturbation; i.e., noise, to the input data. The perturbation itself is the result of the multiplication between a coefficient called epsilon (€) and sign function. Epsilon determines the magnitude of the perturbation. Increasing epsilon value means the perturbation would be more visible to human eyes and decreasing it means that the perturbation would be less visible. Sign function takes three parameters, which consist of: parameters of a DNN model (θ), unmodified input data (x), and correct output that corresponds to x (y_{true}). Parameter θ is a constant and could be omitted. From those three parameters, loss of the model (J) could be determined which is then derived to calculate gradient of loss (∇). Sign function then extracts the sign of the components of the calculated gradient of loss to generate a vector small enough to be imperceptible. That vector

would be the perturbation to be multiplied with the aforementioned coefficient (€). Function to generate perturbation could be mathematically formulated as below [19]:

$$\eta = \text{€}\, sign\,(\nabla x\, J(\theta, x, ytrue))$$

Perturbation would then be added to the original input data, such that the formula would be as seen below with x' as the resulting modified input data. Adding is done to maximize the value of a particular set of pixels in the input data that could sway the result away from the correct one while reducing the value of another set of pixels that could direct the result towards the correct one.

$$x' = x + \text{€}\, sign\,(\nabla x\, J(\theta, x, ytrue\,))$$

As long as the value of epsilon is small enough, the perturbation added to the original input would be imperceptible to humans as could be seen in an experiment result below. In this experiment, the experimenter intends to induce a DNN image recognition system to misclassify an image of sports car.

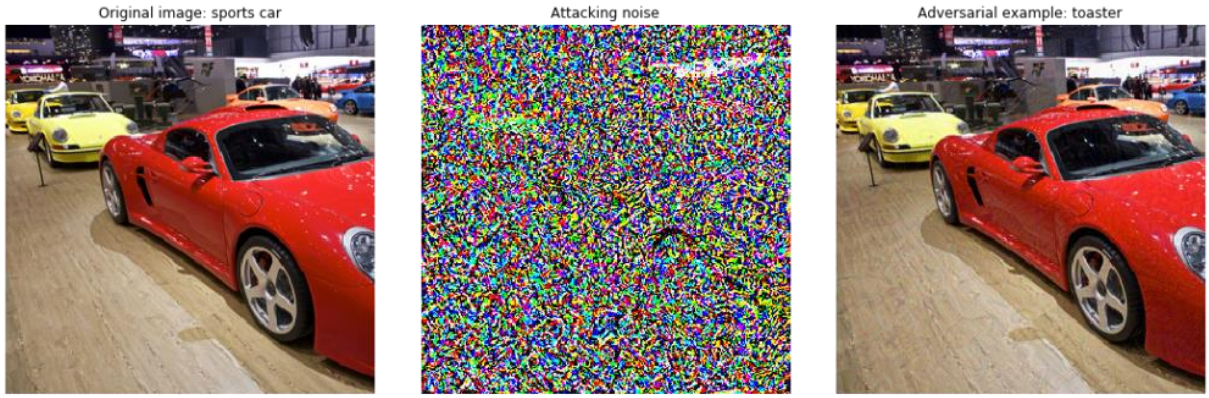

Figure 2 Demonstration of a successful FGSM attack causing erroneous image classification result [20]

Experiments done by I.J. Goodfellow, et. al., shows that epsilon value of 0.25 on a shallow softmax classifier causes it to have an error rate of 99.9 % with average confidence of 79.3% on MNIST data set. The same epsilon value with data set as above causes a maxout network to have an error rate of 89.4% with average confidence of 97.6%. Meanwhile, using epsilon value of 0.1 on convolutional maxout network with CIFAR-10 data set causes it to have an error rate of 87.15% [21].

III.1.2. One-Step Target Class Method

If the use of FGSM is limited to causing misclassification of a given data input, one-step target class method aims to induce a specific classification of a data input where the classification result is dictated by the attacker. Instead of executing gradient ascent to maximize

loss, one-step target class method is more similar to DNN learning method where it tries to maximize the probability of $P(y_{target} \mid x)$ by utilizing gradient descent to minimize loss. However, y_{target} is dictated by the attacker instead of the correct output. The usage of this adversarial attack method is also commonly directed towards DNN used for classification system [22].

This method starts by generating perturbation using the same function as in FSGM as explained in Sub-section III.1.1. The difference would be that parameter y is filled with target output intended by the attacker instead of correct output corresponding to x. The resulting function to generate perturbation could be seen below.

$$\eta = \epsilon\, sign\,(\nabla x\, J(\theta, x, ytarget))$$

However, unlike in FSGM where the sign of the components of gradient of loss is added to the original input, one-step target class method instead subtracts the sign from the original input data. Subtraction is done to reduce the value of a particular set of pixels from the input data that could sway the result away from the targeted output while increasing the value of another set of pixels that could direct the result towards the targeted output. Hence, a different function to modify input data is obtained as could be seen below [23].

$$x' = x - \epsilon\, sign\,(\nabla x\, J(\theta, x, ytarget))$$

Similar to FSGM, as long as the value of epsilon is small enough, the perturbation added to the original input would be imperceptible to humans as could be seen in an experiment result below. In this experiment, the experimenter intends to induce the DNN image recognition system to identify an image of Sylvester Stallone as belonging to "Keanu Reeves" class of images.

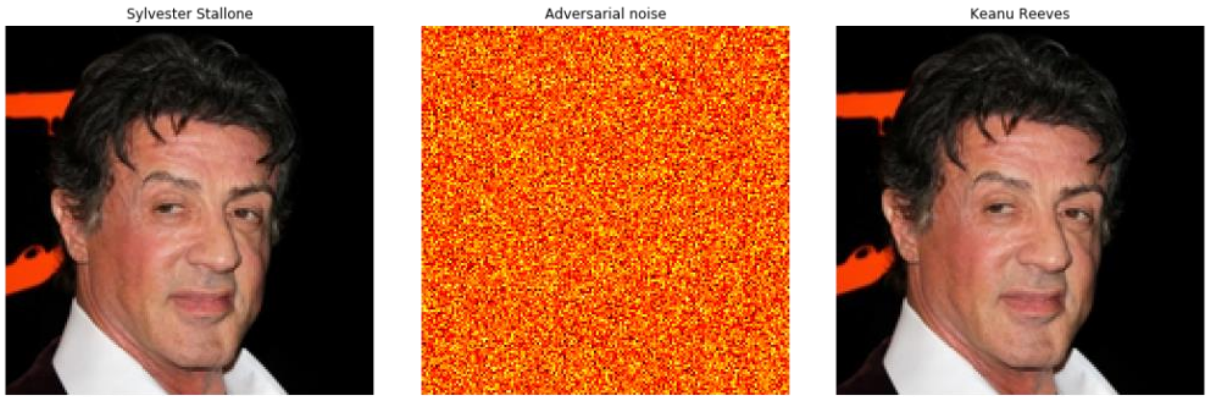

Figure 3 Demonstration of one-step target class method to induce specific but incorrect result on a DNN image recognition system [24]

III.1.3. Basic Iterative Method

This method introduces iteration on FSGM as explained on Sub-section III.1.1. Iteration is done to provide smoother adversarial input data with less visible noise and improve its performance in inducing incorrect prediction on DNN. The objective of Basic Iterative Method (BIM) is the same as FSGM, which is to induce DNN to misclassify input data.

BIM introduces two new parameters. The first one is iteration number (T). The second one is perturbation magnitude for each iteration (α). The process of determining the value of T and α must conform to a specific constraint provided by an equation that could be seen below [25].

$$\alpha T = €$$

Therefore, the value of € in this method indicates the accumulated perturbation magnitude after the final iteration. Since the value of € is divided by the number of iteration (T) in order to produce α, the amount of perturbation that is applied to input data after each iteration is significantly less when compared to FSGM. Less perturbation means a smoother modified input data and less possibility of intentional perturbation being detected by defenders.

In BIM, the iteration is initialized by assigning variable to be iterated with the value of the original input data.

$$X(0)' = X$$

After that, the value of variable would be updated in the n-th iteration with the following formula [26].

$$\mathrm{X(N+1)'} = \mathrm{Clip\ X, €\ \{X(N)'} + \alpha\ sign\ (\nabla X\ J(X(N)', ytrue\ ))\ \}$$

As could be seen from the above formula, the vector that results from sign extraction of components of gradient of loss would be multiplied with the value of α. The result of multiplication would then be added to the input data. In order to ensure that the result of each iteration is within the range of €, the pixel values would be fed into clip function. Clip function converts pixel values exceeding the value of € to that of € itself.

III.1.4. Iterative Least-Likely Class Method

This method introduces iteration to one-step target class method as explained in Sub-section III.1.2. Iteration is introduced to increase the likelihood that a DNN would classify an input data as a target data where the target data is dictated by the attacker. Iteration would also decrease the possibility of the intentional perturbation being detected by the defenders.

The target class in this method would be the least-likely class given a set of input data. The least-likely class would be determined by finding the output class with minimum

probability value of being predicted given a set of input data. This means that the attackers must either have access to a set of prediction data along with their corresponding input data or they have to generate their own. The formula to find the least-likely class is given below.

$$yLL = \arg\min\{P(\ y\ |X)\}$$

On the formula above, y_{LL} is the least-likely class to be predicted while P (y | X) indicates the probability of output class being y given input data X.

The constraint that applies to BIM in Sub-section III.1.3 also applies to the iterative least-likely class method. This means that the process of determining the value of T, α, and € must follow the equation that could be seen below.

$$\alpha T = \text{€}$$

Initializing the iteration is also the same as BIM, which means that the variable to be iterated would be assigned with the value of the original input data.

$$X(0)' = X$$

After that, the value of variable would be updated in the n-th iteration with the following formula.

$$\mathrm{X(N+1)'} = \mathrm{Clip\ X, \text{€}\{X(N)'} - \alpha\ sign\ (\nabla X\ J(X(N)', yLL\ ))\ \}$$

Since this method tries to induce DNN to produce prediction that is the same as the least-likely class, the perturbation that results from multiplication of perturbation magnitude with vector would be subtracted from the input data. This is done because of the same reason as one-step target class method, which is to decrease the value of a set of pixels swaying the result away from the least-likely class, while increasing the value of a set of pixels that directs the result towards the least-likely class. Clip function is also used to ensure that the value of the pixels within the perturbed input data would not exceed that of epsilon (€) [27].

III.2. Countermeasures Against Adversarial Attacks

A variety of countermeasures against adversarial attacks could be executed by the defenders of Deep Neural Network systems. Below is the list of approaches that could be used to counter adversarial attacks [28]:

a. Adversarial Training
b. Defensive Distillation
c. MagNet
d. Generative Adversarial Network

Each of the above approaches would be discussed in-depth on sub-sections below.

III.2.1. Adversarial Training

This approach utilizes adversarial samples on the training data for Deep Neural Networks. The adversarial samples are generated using any of the adversarial attack methods described in Section III.1. First of all, unmodified input data called clean samples would be grouped into a batch; i.e., clean batch. Clean samples would then be fed into the chosen adversarial attack algorithm. It will then result in a new batch; i.e., adversarial batch, that would later be combined with the clean batch. The neural network is then trained using a combined batch, which consists of the clean batch and the adversarial batch [29].

The DNN model would then need to utilize a loss function that could independently calculate losses of each of the two batches during the training phase, instead of a loss function that does not differentiate between clean and adversarial batch. The formulation of the loss function could be seen below [30]:

$$Loss = \frac{1}{(m-k) + \lambda k} \sum_{i \in CLEAN} L(Xi \mid yi) + \lambda \sum_{i \in ADV} L(X(i)ADV \mid yi)$$

L (X | y) is the value of prediction loss of input data X given correct output y. Variable m indicates the total number of input data within the minibatch. Variable k indicates the number of adversarial samples within the minibatch, while λ is the variable used to determine the weight of adversarial samples during the calculation of loss function.

The process of adversarial training could be described through the use of pseudocode below [31]:

1. Initialize DNN called N
2. **Repeat**
    a. Read minibatch B = {X(1), …, X(m)} from training data set
    b. Generate k number of adversarial samples {X(1) ADV, …, X(k)ADV} from clean samples {X(1), …, X(k)} using chosen adversarial attack algorithm
    c. Make new minibatch B’ = {X(1) ADV, …, X(k+1), …, X(m)}
    d. Execute 1 training step of N with minibatch B’
3. **Until** training converged

In the pseudocode above, it could be seen that the first to the k-th sample from a clean batch is replaced by adversarial samples in each training step. Meanwhile, the k+1-th sample to the m-th sample are left as-is. As could be seen above, the resulting DNN would still be vulnerable

to attacks using attack methods that are different from the one used during the adversarial training.

III.2.2. Defensive Distillation

This defensive approach utilizes distillation network, which is a network of neurons that trains itself using prediction output from the original DNN. The difference between prediction made by DNN and the one made by distillation network is that distillation network could predict multiple possible classes of an input data, instead of a singular and absolute classification. An example would be the use of an image that looks similar to both 1 and 7. Distillation network would be able to predict that the image has high possibility of belonging to both class '1' and class '7', while the original DNN would predict the image as either '1' or '7'. The type of identification done by distillation network is called class probabilities identification [32]. Usage of class probabilities identification means that the resulting distilled DNN would be less vulnerable to adversarial attacks.

III.2.3. MagNet

MagNet utilizes two components on its DNN model to thwart adversarial attacks. The first component is called detector, which is a function that determines if an input data is adversarial or not. The second component is called reformer, which is a function responsible for recreating an input data, whether adversarial or clean, so it would be the same or close to the normal data as defined by the training data. The illustration below visualizes the workflow of the MagNet defense mechanism:

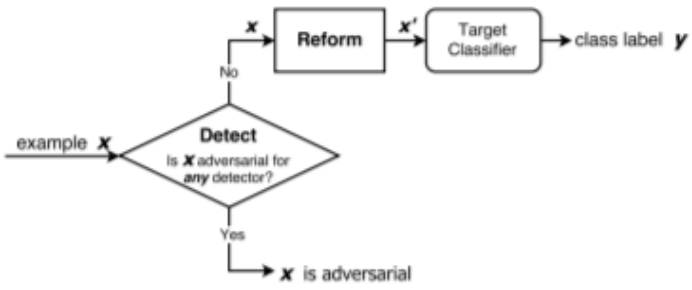

Figure 4 Visualization of MagNet workflow during testing phase [33]

From the above illustration, it could be seen that the detector component is placed before the reformer component. First of all, the detector would detect whether the input data is adversarial or not. If the input data is detected as adversarial, it means that the pixel values

underlying the image are far different from the threshold values determined by the system and therefore rejected. But, if the input data is detected as not adversarial, it means that the pixel values are within the threshold values determined by the system and therefore passed to the reformer component. The reformer component would then rebuild the input data to be closer to the normal data as given by the training data and then passed it to target classifier to generate the prediction [34].

Detector component used by MagNet could either be based on reconstruction error or probability divergence. Reconstruction error-based detector calculates reconstruction error of input data by comparing it before and after being fed into autoencoder. The resulting reconstruction error is then used to estimate distance between the input data and the boundary of classification. If the error is large, then that particular input data could be determined as adversarial. This type of detector is useful for detecting adversarial input data that has large reconstruction error. For detecting adversarial input data that has small reconstruction error, MagNet uses detector based on probability divergence. This type of detector compares the probability mass function of output when input data t is fed directly to the last layer of DNN against output of the same input data when it is fed into autoencoder; i.e., reconstructed input data. If the difference is large, then that particular input data could be determined as adversarial [35].

Reformer component used by MagNet could either be a noise-based reformer or autoencoder-based reformer. A noise-based reformer generates random noise that is then added to the input data. The disadvantage of this type of reformer is that it changes both unmodified and adversarial input data without any discrimination, which results in a distorted unmodified input data. An autoencoder-based reformer uses autoencoder to generate a representation of input data that is as close as possible to the reference data as given by the training data. This type of autoencoder allows for significant changes to be done to adversarial input data in order to make them closer to the original input data, while leaving unmodified input data without any significant changes [36].

III.2.4. Generative Adversarial Network

Generative Adversarial Network (GAN) uses two networks which consist of generative network and discriminative network. The role of generative network is to produce adversarial input data by utilizing random variables to generate its own data. The generated data would then be mixed with real input data before being passed into the discriminative network. Meanwhile, the role of discriminative network is to differentiate whether an input

data is generated or real. From that process, it could be seen that the objective of generative network is to fool the discriminative network into making wrong differentiation of real and generated input data, while the objective of the discriminative network is to build a classifier that is able to differentiate between real and generated input data [37].

The process of determining the ideal discriminator (D) given a generator (G) is done by maximizing error in classification for G, while at the same time minimizing error in classification for D. The pseudocode for the training process of GAN that implements above concept is provided below [38].

1. **for** # of training iterations
    a. **for** k steps
        i. Get minibatch of m noise samples {z(1), … , z(m)} from previous noise.
        ii. Get minibatch of m samples {x(1), … , x(m)}from G.
        iii. Update D by ascending its gradient.

    end

    b. Get minibatch of m noise samples {z(1), … , z(m)} from previous noise.
    c. Update G by descending its gradient.

Since each discriminative network and generative network has objective that contradicts each other, it could be said that GAN system simulates an engagement between adversarial attack and defense of DNN as it happens in real life.

IV. Conclusion and Recommendations

IV.1. Conclusion

Two conclusions could be drawn from this report.

1. There are 4 adversarial attack methods that could be conducted against DNN in order to induce incorrect prediction, which consist of:
    a. Fast gradient sign method; this method generates an adversarial example by utilizing gradient ascent to maximize loss between predicted and correct output.
    b. One-step target class method; this method aims to induce a specific classification of a data input where the classification result is dictated by the attacker. This is done through the utilization of gradient descent to minimize loss between predicted and targeted output.

c. Basic iterative method; this method introduces iteration to FGSM to increase accuracy and reduce detection possibility.
d. Iterative least-likely class method; this method introduces iteration to one-step target class method by marking the least-likely class as its target class.

2. There are 4 countermeasures against adversarial attacks that could be executed, which consist of:
   a. Adversarial Training; this approach utilizes adversarial samples on the training data for DNN.
   b. Defensive Distillation; this defensive approach utilizes distillation network, which is a network of neurons that trains itself using prediction output from the original DNN to produce class probabilities identification.
   c. MagNet; utilizes two components on its DNN model, which consist of detector to identify adversarial input data and reformer to recreate input data.
   d. Generative Adversarial Network; this approach uses two networks which consist of generative network with the role of generating data and maximizing classification error and discriminative network with the role of building a classifier and minimizing classification error.

IV.2. Recommendations

This report primarily focuses on exploring concepts and mechanisms of the various methods and approaches of adversarial attacks and defenses. Future research should focus on quantitative comparison on cost of implementation and effectiveness between those various adversarial attack and defense approaches. Execution of adversarial attacks and defenses on DNN already implemented on real life systems could also be analyzed.

V. References

[1] G.E. Moore, "Cramming More Components onto Integrated Circuits", *Intel.com. Electronics Magazine*, 1965.

[2] L. Du, Y. Du, "Hardware Accelerator Design for Machine Leaning", *IntechOpen*, June 2017. Accessed on: Apr. 20, 2020. [Online]. Available URL: https://www.intechopen.com/books/machine-learning-advanced-techniques-and-emerging-applications/hardware-accelerator-design-for-machine-learning

[3] Haohui, "Adversarial Attacks in Machine Learning and How to Defend Against Them", *Towards Data Science*, Dec. 2019. Accessed on: Apr. 27, 2020. [Online]. Available URL: https://towardsdatascience.com/adversarial-attacks-in-machine-learning-and-how-to-defend-against-them-a2beed95f49c

[4] J.R. Koza, F.H. Bennett, D. Andre, M.A. Keane, "Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming", *Artificial Intelligence in Design '96*, pp. 151-170, 1996.

[5] S.J. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach (3rd Edition)", *Prentice Hall*, 2010.

[6] G. Hinton, T. Sejnowski, "Unsupervised Learning: Foundations of Neural Computation", *MIT Press*, 1999.

[7] M.F. Balcan, A. Blum, "A Discriminative Model for Semi-Supervised Learning", *Journal of the ACM*, 2009.

[8] W. Daelemans, V. Hoste, "Evaluation of Machine Learning Methods for Natural Language Processing Tasks", *CNTS Language Technology Group University of Antwerp*, 2002.

[9] A. Krizhevsky, I. Sutskever, G.E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *University of Toronto*, 2012.

[10] A. Oza, "Fraud Detection using Machine Learning", *Stanford University*, 2018.

[11] M. Zamani, "Machine Learning Techniques for Intrusion Detection", *Department of Computer Science University of New Mexico*, 2013.

[12] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview", *Neural Networks: The Official Journal of the International Neural Network Society*, 2015.

[13] S. Vieira, W.H.L. Pinaya, A. Mechelli, "Using Deep Learning to Investigate the Neuroimaging Correlates of Psychiatric and Neurological Disorders: Methods and Applications", *ResearchGate*, Jan. 2017. Accessed on: Apr. 20, 2020. [Online]. Available URL: https://www.researchgate.net/profile/Sandra_Vieira5/publication/312205163/figure/fig1/AS:453658144972800@1485171938968/a-The-building-block-of-deep-neural-networks-artificial-neuron-or-node-Each-input-x.png

[14] F. Agostinelli, M. Hoffman, P. Sadowski, P. Baldi, "Learning Activation Functions to Improve Deep Neural Networks", *Cornell University*, 2014.

[15] K. Janocha, W.M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification", *Cornell University*, 2017.

[16] A. Kurakin, I. Goodfellow, S. Bengio, “Adversarial Machine Leaning at Scale”, *International Conference on Learning Representations (ICLR)*, pp. 1, 2017.

[17] A. Kurakin, I. Goodfellow, S. Bengio, “Adversarial Machine Leaning at Scale”, *International Conference on Learning Representations (ICLR)*, pp. 2-3, 2017.

[18] K. Ren, T. Zheng, Z. Qin, X. Liu, “Adversarial Attacks and Defenses in Deep Learning”, *ScienceDirect*, pp. 3, Jan. 2020. Accessed on: Apr. 7, 2020. [Online]. Available URL: https://reader.elsevier.com/reader/sd/pii/S209580991930503X?token=C372FE4CE42F6342F64D0DB6EAAACCB7BD1B088C0B56236949790AE5133DA7FEB127D4B39941B84B42B0E268EE3CBDED

[19] I.J. Goodfellow, J. Shlens, C. Szegedy, “Explaining and Harnessing Adversarial Examples”, *International Conference on Learning Representations (ICLR)*, pp. 3, 2015.

[20] E. Mikhailov, R. Trusov, “How Adversarial Attacks Work”, *YCombinator*, Nov. 2017. Accessed on: Apr. 27, 2020. [Online]. Available URL: https://blog.ycombinator.com/wp-content/uploads/2017/11/car-1024x347.png

[21] I.J. Goodfellow, J. Shlens, C. Szegedy, “Explaining and Harnessing Adversarial Examples”, *International Conference on Learning Representations (ICLR)*, pp. 3, 2015.

[22] A. Kurakin, I. Goodfellow, S. Bengio, “Adversarial Machine Leaning at Scale”, *International Conference on Learning Representations (ICLR)*, pp. 3, 2017.

[23] T. Nam, J.H. Ko, S. Mukhopadhyay, “Cascade Adversarial Machine Learning Regularized with A Unified Embedding”, *International Conference on Learning Representations (ICLR)*, pp. 2, 2018.

[24] E. Mikhailov, R. Trusov, “How Adversarial Attacks Work”, *YCombinator*, Nov. 2017. Accessed on: Apr. 27, 2020. [Online]. Available URL: https://blog.ycombinator.com/wp-content/uploads/2017/11/stallone-1024x347.png

[25] K. Ren, T. Zheng, Z. Qin, X. Liu, “Adversarial Attacks and Defenses in Deep Learning”, *ScienceDirect*, pp. 3, Jan. 2020. Accessed on: Apr. 7, 2020. [Online]. Available URL: https://reader.elsevier.com/reader/sd/pii/S209580991930503X?token=C372FE4CE42F6342F64D0DB6EAAACCB7BD1B088C0B56236949790AE5133DA7FEB127D4B39941B84B42B0E268EE3CBDED

[26] A. Kurakin, I.J. Goodfellow, S. Bengio, “Adversarial Examples in the Physical World”, *International Conference on Learning Representations (ICLR)*, pp. 4, 2017.

[27] A. Kurakin, I.J. Goodfellow, S. Bengio, “Adversarial Examples in the Physical World”, *International Conference on Learning Representations (ICLR)*, pp. 4 - 5, 2017.

[28] P. Samangouei, M. Kabkab, R. Chellappa, “Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models”, *International Conference on Learning Representations (ICLR)*, pp. 3 – 4, 2018.

[29] P. Samangouei, M. Kabkab, R. Chellappa, “Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models”, *International Conference on Learning Representations (ICLR)*, pp. 3, 2018.

[30] A. Kurakin, I. Goodfellow, S. Bengio, “Adversarial Machine Leaning at Scale”, *International Conference on Learning Representations (ICLR)*, pp. 3, 2017.

[31] A. Kurakin, I. Goodfellow, S. Bengio, “Adversarial Machine Leaning at Scale”, *International Conference on Learning Representations (ICLR)*, pp. 4, 2017.

[32] M. Plotz, “Paper Summary: Distillation As A Defense to Adversarial Perturbations Against Deep Neural Networks”, *Medium*, Nov. 2018. Accessed on: Apr. 27, 2020. [Online]. Available URL: https://medium.com/@hyponymous/paper-summary-distillation-as-a-defense-to-adversarial-perturbations-against-deep-neural-networks-b68970789bfc

[33] D. Meng, H. Chen, “MagNet: A Two-Pronged Defense Against Adversarial Examples”, *ACM Conference on Computer and Communications Security (CCS)*, pp. 5, 2017.

[34] D. Meng, H. Chen, “MagNet: A Two-Pronged Defense Against Adversarial Examples”, *ACM Conference on Computer and Communications Security (CCS)*, pp. 2, 2017.

[35] D. Meng, H. Chen, “MagNet: A Two-Pronged Defense Against Adversarial Examples”, *ACM Conference on Computer and Communications Security (CCS)*, pp. 6, 2017.

[36] D. Meng, H. Chen, “MagNet: A Two-Pronged Defense Against Adversarial Examples”, *ACM Conference on Computer and Communications Security (CCS)*, pp. 7, 2017.

[37] J. Rocca, “Understanding Generative Adversarial Networks (GANs)”, *Towards Data Science*, Jan. 2019. Accessed on: Apr. 27, 2020. [Online]. Available URL:

https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

[38] I.J. Goodfellow, J.P-Abadie, M. Mirza, B. Xu, D.W-Farley, S. Ozair, A. Courville, Y. Bengio, “Generative Adversarial Nets”, *University of Montreal*, pp. 4, 2014.