Defense Against Adversarial Machine Learning

Jesus Alejandro Noguera Ballen[1], Jorge E. Camargo,[2]

National University of Colombia

[1]Systems and Computing Engineering Student,janoguerab@unal.edu.co

[2]PhD Systems and Computing Engineering, jecamargom@unal.edu.co

Abstract

In recent years it has been shown that many systems of image recognition (Neural networks) are susceptible to adversary attacks, these are intentional modifications of the images, are imperceptible to the human eye, and lead to bad classification by the system, in other words they are like Optical illusions for systems. So it is extremely important to give robustness to the learning algorithms so that they can respond effectively against attacks, in this paper we seek to show some methods that can give robustness to the learning algorithms and can therefore be more effective in the process of classification.

Keywords: Adversarial example, algorithm, image recognition, machine learning, neural network

Defense Against Adversarial Machine Learning

Introduction

According to Alpaydin (Alpaydin, 2014) Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both.

Machine learning uses the theory of statistics in building mathematical models, because the core task is making inference from a sample. The role of computer science is twofold: First, in training, we need efficient algorithms to solve the optimization problem, as well as to store and process the massive amount of data we generally have. Second, once a model is learned, its representation and algorithmic solution for inference needs to be efficient as well. In certain applications, the efficiency of the learning or inference algorithm, namely, its space and time complexity, may be as important as its predictive accuracy.

In recent years machine learning has been used to detect new malware, while malware authors have strong motivation to attack such algorithms. Malware authors usually have no access to the detailed structures and parameters of the machine learning models used by malware detection systems, and therefore they can only perform black-box attacks (Hu & Tan, 2017).

Certain type of machine learning algorithms are Artificial neural networks (ANNs) these are mathematical models originally inspired by the idea of reproducing the functioning of human brain. In particular, from their biological counterpart, they have inherited the feature that data processing is distributed through a large network of processing units (Dasgupta, Liu, & Siegelmann, 2007). Each neuron maps a set of inputs to output using an activation function. The learning governs the weights and activation function so as to be able to correctly determine the output. Weights in a multi-layered feed

forward are updated by the back-propagation algorithm. Neuron was first introduced by McCulloch-Pitts, followed by Hebb's learning rule, eventually giving rise to multi-layer feed-forward perceptron and backpropagation algorithm. ANNs deal with supervised (CNN, DNN) and unsupervised network models (self organizing maps) and their learning rules.

A deep neural network (DNN) enables feature learning using raw data as input. Multiple hidden layers and its interconnections extract the features from unprocessed input and thus enhances the performance by finding latent structures in unlabeled, unstructured data. A typical DNN architecture, graphically depicted in Figure 1, consists of multiple successive layers (at least 2 hidden layers) of neurons. Each processing layer can be viewed as learning a different, more abstract representation of the original multidimensional input distribution. As a whole, a DNN can be viewed as a highly complex function that is capable of nonlinearly mapping original high-dimensional data points to a lower dimensional space.

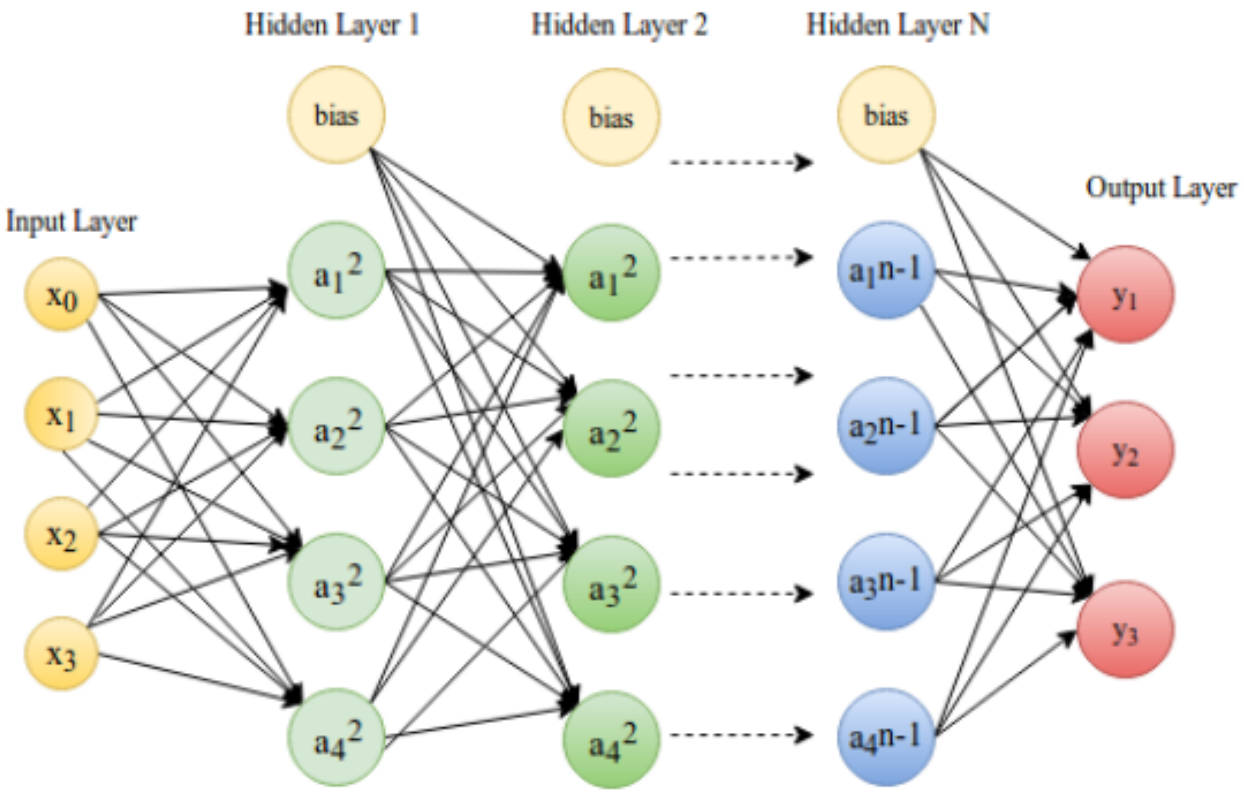

Figure 1. Deep Neural Network.

This work focuses on classifying correctly against an adversarial example, this 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. In many cases, these modifications can be so subtle that a human observer does not even notice the modification at all, yet the classifier still makes a mistake. Adversarial examples pose security concerns because they could be used to perform an attack on machine learning systems, even if the adversary has no access to the underlying model (Kurakin, Goodfellow, & Bengio, 2016).

Problem definition

Most existing machine learning classifiers are highly vulnerable to adversarial examples. Up to now, all previous work have assumed a threat model in which the adversary can feed data directly into the machine learning classifier. This is not always the

case for systems operating in the physical world, for example those which are using signals from cameras and other sensors as an input (Kurakin et al., 2016).

As we saw, adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake; they're like optical illusions for machines.

To get an idea of what adversarial examples looks like, consider this demonstration (Goodfellow, Shlens, & Szegedy, 2015): starting with an image of a panda, the attacker adds a small perturbation that has been calculated to make the image be recognized as a gibbon with high confidence Figure 2.

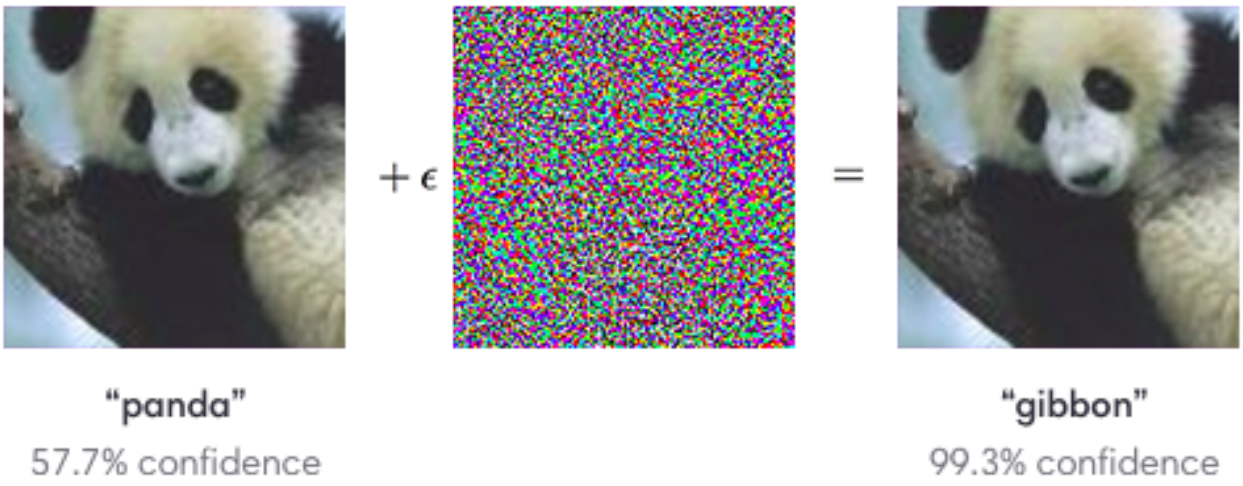

Figure 2. A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

As we see, an adversarial input, overlaid on a typical image, can cause a classifier to miscategorize a panda as a gibbon. The algorithms are also useful as a way of speeding up adversarial training or even just analysis of trained networks.

Adversarial examples have the potential to be dangerous. For example, attackers could target autonomous vehicles by using stickers or paint to create an adversarial stop sign that the vehicle would interpret as a 'yield' or other sign, as discussed in (Papernot et al., 2016)

Goals

Attempted defenses against adversarial examples

There are a lot of traditional techniques for making machine learning models more robust, such as weight decay and dropout (Helmbold & Long, 2016), generally do not provide a practical defense against adversarial examples. So far, only two methods have provided a significant defense.

Adversarial training. This is a brute force solution where we simply generate a lot of adversarial examples and explicitly train the model not to be fooled by each of them.

Defensive distillation. This is a strategy where we train the model to output probabilities of different classes, rather than hard decisions about which class to output. The probabilities are supplied by an earlier model, trained on the same task using hard class labels. This creates a model whose surface is smoothed in the directions an adversary will typically try to exploit, making it difficult for them to discover adversarial input tweaks that lead to incorrect categorization (Papernot, McDaniel, Wu, Jha, & Swami, 2015).

We will focus on Adversarial training, performing a new training but without the need to create Adversarial examples, instead, the same images used to train the neural network will be used, but applying different techniques of images transformation, such as, reflection, dilation, and rotation, as Kong proposed (Wu, Hu, & Kong, 2015).

Methodology

Adversarial examples are hard to defeat because it is difficult to construct a theoretical model of the adversarial example crafting process. Adversarial examples are

solutions to an optimization problem that is non-linear and non-convex for many Machine Learning models, including neural networks. Because we don't have good theoretical tools for describing the solutions to these complicated optimization problems, it is very hard to make any kind of theoretical argument that a defense will rule out a set of adversarial examples.

Adversarial examples are also hard to defeat because they require machine learning models to produce good outputs for every possible input. Most of the time, machine learning models work very well but only work on a very small amount of all the many possible inputs they might encounter.

The defense strategy is implemented using a two-step procedure.

1. Apply several transformations to the input image to make the attack perturbations less effective (Figure 3).

2. Classify using networks trained with adversarial examples. This kind of training helps the networks generalize better to images that are not just outside the training set, but do not even occur in "nature".

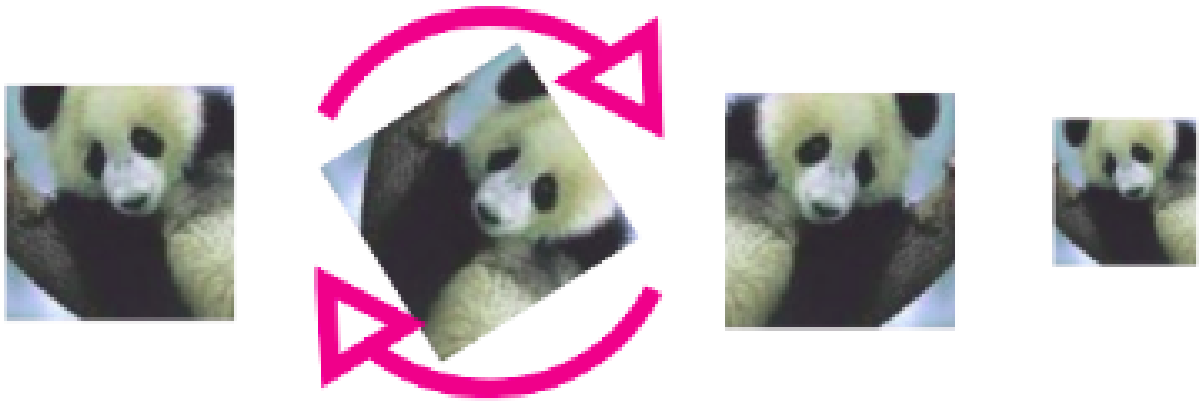

Figure 3. Different kinds of image transformation, rotation, reflection and dilation.

These steps improve the robustness of a classifier. Then retrain the network with the images, Figure 4.

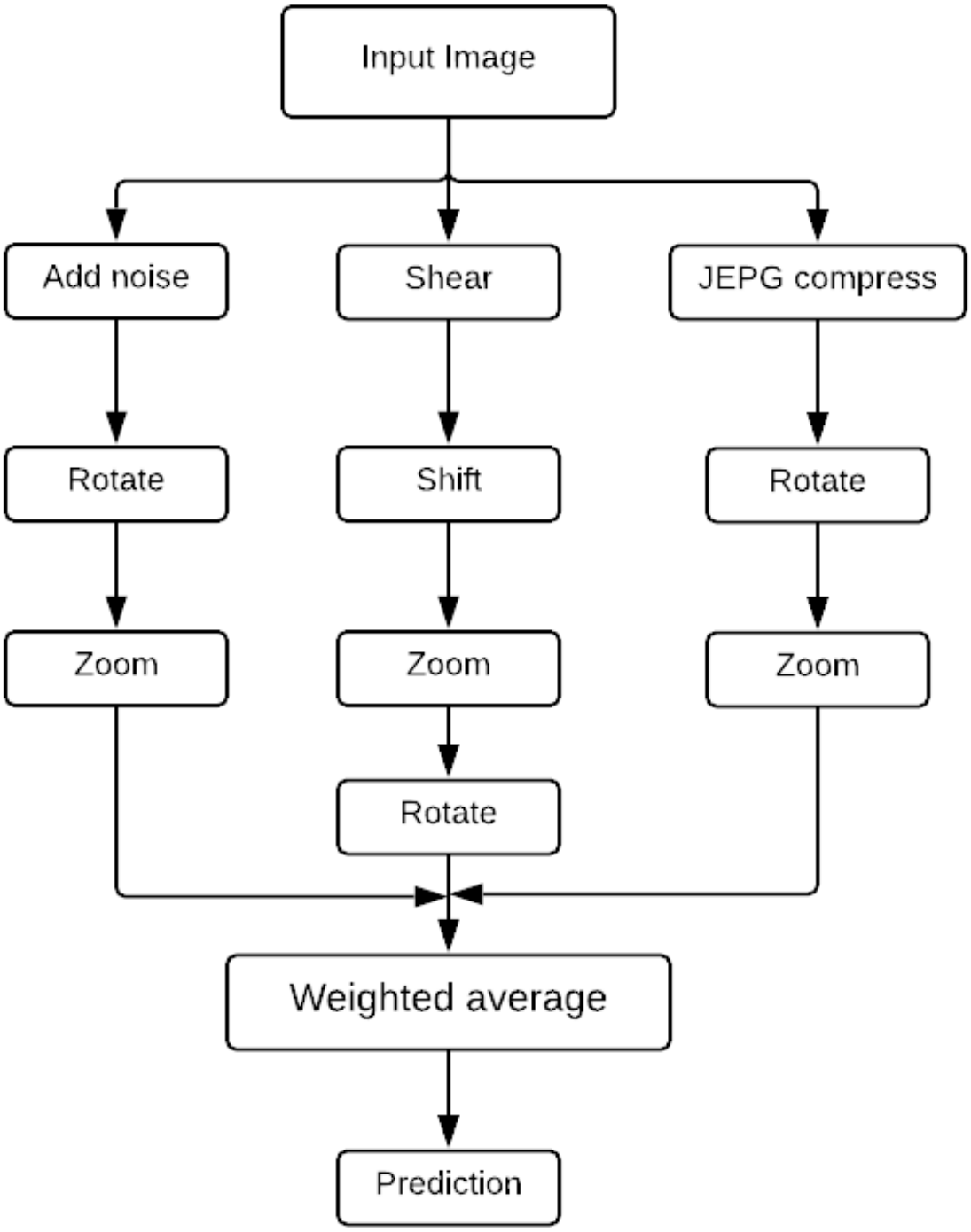

Figure 4. Complete pipeline.

The defense outlined here was able to do well against some attacks by using a

combination of image transformations and adversarially trained models. However, it should be noted that it would be harder to defend against a newly created attack that has intimate knowledge of this defense. The defense could be strengthened by using it to produce gradients that can be used in new attacks and then fine-tuning the model parameters by training with new adversarial examples thus created (Kurakin et al., 2018).

References

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Dasgupta, B., Liu, D., & Siegelmann, H. T. (2007). Neural networks. In *Handbook of approximation algorithms and metaheuristics*. doi:10.1201/9781420010749

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and Harnessing. *ICLR*. arXiv: arXiv:1412.6572v3

Helmbold, D. P. & Long, P. M. (2016). Dropout versus weight decay for deep networks. *CoRR*, *abs/1602.04484*. arXiv: 1602.04484. Retrieved from http://arxiv.org/abs/1602.04484

Hu, W. & Tan, Y. (2017). Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN. arXiv: 1702.05983. Retrieved from http://arxiv.org/abs/1702.05983

Kurakin, A., Goodfellow, I. J., Bengio, S., Dong, Y., Liao, F., Liang, M., ... Abe, M. (2018). Adversarial attacks and defences competition. *CoRR*, *abs/1804.00097*. arXiv: 1804.00097. Retrieved from http://arxiv.org/abs/1804.00097

Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial examples in the physical world. (100), 1–14. arXiv: 1607.02533. Retrieved from http://arxiv.org/abs/1607.02533

Papernot, N., McDaniel, P. D., Wu, X., Jha, S., & Swami, A. (2015). Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, *abs/1511.04508*. arXiv: 1511.04508. Retrieved from http://arxiv.org/abs/1511.04508

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2016). Practical Black-Box Attacks against Machine Learning. (November). doi:10.1145/3052973.3053009. arXiv: 1602.02697

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., ... Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, *abs/1409.4842*. arXiv: 1409.4842. Retrieved from http://arxiv.org/abs/1409.4842

Wu, F., Hu, P., & Kong, D. (2015). Flip-rotate-pooling convolution and split dropout on convolution neural networks for image classification. *CoRR*, *abs/1507.08754*. arXiv: 1507.08754. Retrieved from http://arxiv.org/abs/1507.08754