EFFECTS OF GAUSSIAN AND ADVERSARIAL PERTURBATIONS ON A CNN'S PERFORMANCE

Kareem Arab

COMP 4107
Professor Tony White

ABSTRACT

Convolutional Neural Networks (CNN) are powerful class of Deep Neural Networks (DNN)that are used mostly for image recognition and classification. They have proven themselves as one of the best models in modern ML and are widely accepted as the golden standard for image classification. Yet sometimes these networks are susceptible to certain types of data that can be harmful. In this paper we explore two distinct types of input data noise that can interfere with a CNN's ability to perform correct classifications and to what degree these perturbations are harmful.

1 Introduction

Deep Neural Networks are a class of algorithms that have proved their worth over the years in tasks varying from inference to classification. Their success is, in-part, attributed to their highly non-linear nature and ability to map extremely complex functions. Yet, in 2013, a group of researchers at Google published a paper titled "Intriguing properties of neural networks" [1]. In this paper they claimed that they were able to come up with a way to perturb input data in a way that would allow it to become misclassified. This was an eye-opening revelation to the scientific community as the researcher consensus began shifting towards defending neural networks against these non-random targeted attacks, now most commonly known as Adversarial Attacks (AA).

Since the publishing of the 2013 Paper, many adversarial attack techniques have been developed, most notably the Fast Gradient Signed Method (FGSM), DeepFool, and Carlini-Wagner Attacks [2][6]. Generally, AA are built using additive models where a certain linear operator is used to add a non-random perturbation on to the inputs [3].

A well-trained Neural Network's performance is relatively invariant to random noise, yet the issue arises when the noise is non-random and is specifically designed to force a network to misclassify the samples given to it. Therefore in this paper I attempt to compare the effects that random and non-random noise have on a CNN's classification performance[4][5].

2 Background & Related Work

2.1 Dataset

The dataset used in this paper is the MNIST database of handwritten digits. It is comprised of 60,000 training examples and 10,000 testing examples. Each sample is a 28×28 grey-scale image of a digit between 0 and 9.

2.2 Data Preperation

To prepare the MNIST data, we began by loading the data using the Keras dataset importer module and at this point the data is already split into training and testing (inputs/labels). Each image is an 28×28 matrix that is then reshaped into a 4-dimensional tensor (explained in section 3.3). Next, we normalize all the images by dividing each pixel by 255, which results in a normalized input space $\boldsymbol{x} \in [0, 1]$ and we apply one-hot encoding to the input data as well as randomly permute it in order to reduce over-fitting and to increase generalization. Finally, the training portion of the input data is split into 90% training and 10% validation.

2.3 Libraries

A number of Python Libraries were used to implement the models and analysis.

- **TensorFlow** was used to build the different CNN and Adversarial Attack computational graphs.
- **Numpy** was used for most linear algebra related tasks.
- **Matplotlib** was used for visualization of results.
- **Keras** was used to import the MNIST data.

2.4 Environment

The environment used to produce *all* the results in this paper is as follows:

- MacBook Pro 2.2 GHz 6-Core Intel Core i7 - 16GB RAM
- Python 3.7

3 Methodology

The two main models used in this paper are the **Convolutional Neural Network** and the **Carlini-Wagner Attack**. We will shortly see how both these models are not run independantly and how the CW Attack depends on the CNN's computational graph to generate it's adversarial samples. Let's begin by defining the models used.

3.1 Gaussian noise on data

Applying random noise to the input data is relatively simple. We can achieve this by drawing random samples from a gaussian (normal) distribution $P(x)$ given a mean value of $\mu = 0$ and a standard deviation $\sigma = 0.1 \rightarrow 0.3$ in the form of the original image (28×28) and then adding both image matrices together. The result is shown in figures 1 and 2.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{1}$$

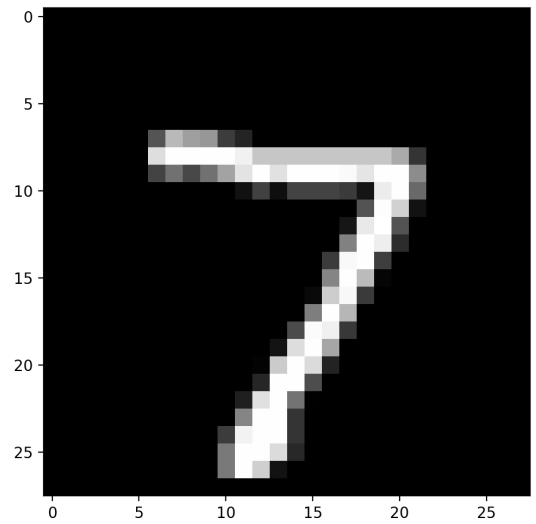

Figure 1: The original digit 7 from the MNIST database.

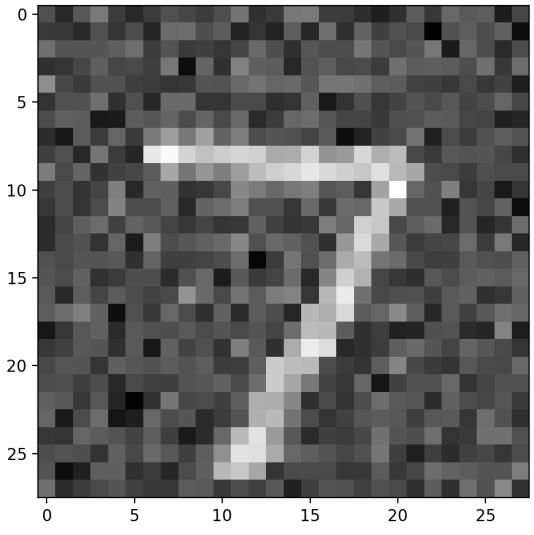

Figure 2: The digit 7 from the MNIST database with gaussian noise added to it (std dev = 0.2).

3.2 Non-random noise on data

The other type of noise we are aiming to compare in this paper is non-random noise. Non-random noise is a type of targetted, noise that has a certain goal. In our case this noise is trying to force our trained CNN classifier to mis-classify a certain input to a target class of our choice.

3.2.1 Carlini Wagner Attack

An Adversarial Attack is a technique of generating non-random targeted noise that forces a trained classifier to mis-classify. Given a clean, input sample that belongs to a certain class, $\boldsymbol{x_0} \in C_i$, the adversarial attack attempts to perturb the sample, converting it into an adversarial sample $\boldsymbol{x}$ that aims to be misclassified by the network. The misclassification portion of this technique is targeted, meaning we must specify which class we want the adversarial example to point to, therefore forcing $\boldsymbol{x_0}$ from $C_i \rightarrow C_t$.

Specifically, the Carlini-Wagner Attack does exactly that by directly minimizing the L_2 distance between the original image and it's adversarial counter-part and at the same time maximizing the probability of misclassification [6]. The attack is defined using the following optimization,

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x_0}\| + l_\Omega(\boldsymbol{x}) \tag{2}$$

where,

$$l_\Omega(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \; max_{j \neq t} \left\{g_j(\boldsymbol{x})\right\} - g_t(\boldsymbol{x}) \leq 0 \\ +\infty, & \text{otherwise} \end{cases} \tag{3}$$

This essentially states that if the constraint $max_{j \neq t} \left\{g_j(\boldsymbol{x})\right\} - g_t(\boldsymbol{x}) \leq 0$ is satisfied, then the original data point $\boldsymbol{x_0} = \boldsymbol{x}$ [6]. It is also important to note that the CW Attack operates under the assumption that the model's internal state is completely exposed.

But a very important question to ask is: *where do these adversarial 'counter-parts' come from?* And the answer is, to a certain extent, from the high dimensionality of the inputs and the internal linearity of deep neural network models. Although the reason DNNs are so powerful is that they are highly non-linear, yet they mostly rely (at least in this case) on a linear activation function: Rectified Linear Unit (ReLU). The advantage that the ReLU function has on other activations is that it provides a non-zero gradient everywhere to the right of 0, and therefore making it stable and fast to run.

3.3 Convolutional Neural Network Model

The network built was a 2-layer CNN (outlined below) that has a cross entropy objective function defined as,

$$H(p, q) = -\sum_x p(x) \log q(x) \tag{4}$$

which is minimized using the RMSProp optimizer using a learning rate of 0.001 and a decay of 0.9.

- **Input Layer**
  - $Tensor(?, 28, 28, 1)$
- **Convolutional Layer 1**
  - activation: $ReLU$
  - filters: 32
  - filter size: 3×3
  - filter padding: SAME
  - filter stride: 2
- **Convolutional Layer 2**
  - activation: $ReLU$
  - filters: 64

  - filter size: 3×3
  - filter padding: SAME
  - filter stride: 2
- **Fully-Connected Layer**
  - activation: $ReLU$
  - neurons: 128
- **Output Layer**
  - activation: $Softmax$
  - neurons: 10

4 Results and Discussion

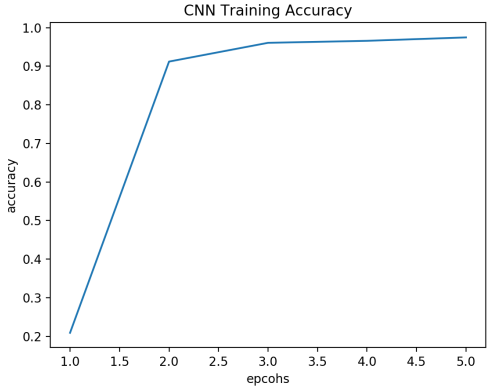

Figure 3: TRAINING // CNN classification accuracy on MNIST digits.

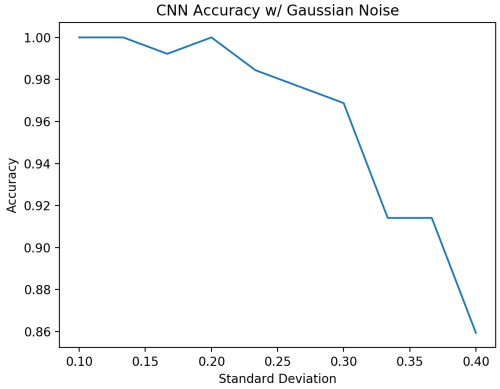

Figure 4: TESTING // CNN classification accuracy on MNIST digits using images with gaussian noise.

Now that we have outlined the techniques and models we are using, let's take a look at the results they produce and how we can interpret them.

4.1 CNN Performance

Before we look into the Performance of the CNN under the influence of the two types of noise we introduced earlier, let us asses the performance of the Convolutional Neural Network, only using the clean MNIST test set.

We can see in Figure 3 that the network's accuracy plateaus at around 98%, which is a very promising result. The high accuracy is attributed, in part, to how powerful the CNN is. Taking into account the powerful gradient decent optimizer being used, the network's spatial invariance properties and amount of data available, it becomes apparent why such a high result was attained.

4.2 Gaussian Noise

As expected, figure 4 shows that the classification results pertaining to the gaussian noise samples were above par even while using very high σ values. This is attributed to the high non-linearity of the CNN and how it is able to efficiently utilize the spatial properties of the image to classify it.

4.3 Carlini-Wagner Attack

As we can see in figure 5, the perturbations introduced by the CW attack produce a visually similar image compared to the original. Yet the trained classifier was unable to produce remotely sufficient results when fed the adversarial samples with the original labels. Table 1 shows the comparison between the CNN performance when the adversarial data is compared with the original class labels and the target class labels. The performance on the target class is much higher as expected since the samples have been optimized to for this type of misclassification.

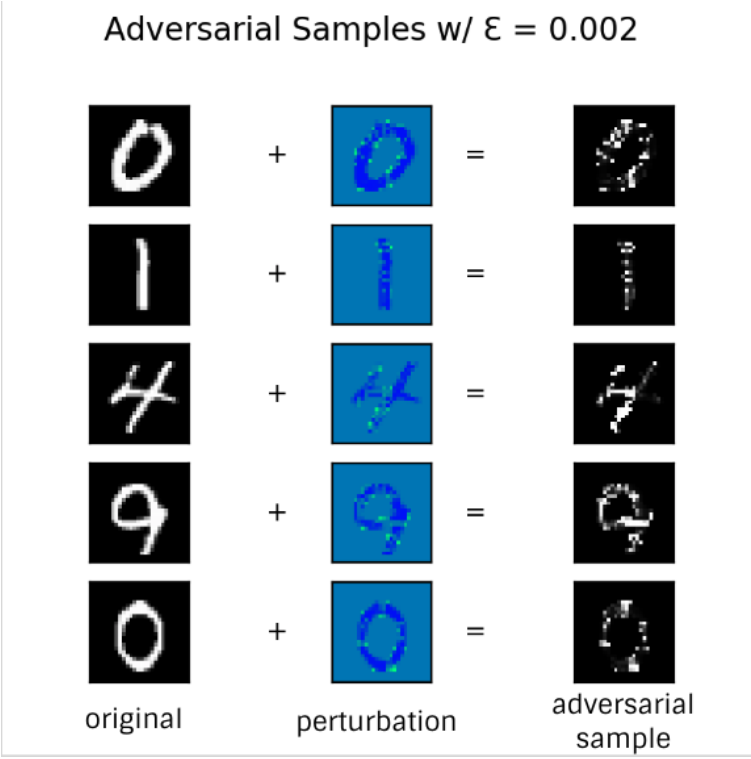

Figure 5: Display of generated Adversarial Samples using CW Attack. ϵ is the scaling factor.

Table 1: Performance Comparsion Original and Target Classes

Original Classes	Target classes
0.203	0.783

5 Conclusion and Future Work

In this paper, two types of input data perturbations were introduced: random and non-random. Both types of perturbations were compared when fed into a training CNN classifier and the results were very promising. I was able to show that there is a significant difference between the effects that random and non-random noise have on trained deep neural networks. AA were able to out-perform random noise not just because they force the classifier to misclassify but because they are also capable of forcing the classification to a *specified* target class. However, this only gives rise to the fact that deep neural networks have very dangerous vulnerabilities and lack stablity and reliablility. Many new defensive techniques were developed such as MagNet and Model Distillation, and these are routes I intend in pursuing in the future.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, Rob Fergus (2014). Intriguing properties of neural networks. arXiv:1312.6199v4 [cs.CV]. Retrieved from https://arxiv.org/abs/1312.6199

[2] Cody Marie Wild. Know Your Adversary: Understanding Adversarial Examples. Retrieved from https://towardsdatascience.com/know-your-adversary-understanding-adversarial-examples-part-1-2-63af4c2f5830

[3] Chapter 3: Adversarial Attack. Retrieved from https://engineering.purdue.edu/ChanGroup/ECE595/files/chapter3.pdf

[4] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh (2017). A EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. arXiv:1709.04114 [stat.ML]. Retrieved from https://arxiv.org/abs/1709.04114

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[6] Nicholas Carlini, David Wagner (2017). Towards Evaluating the Robustness of Neural Networks. arXiv:1608.04644v2 [cs.CR]. Retrieved from https://arxiv.org/abs/1608.04644