Using GANs and Image Hash to Defend Against Adversarial Attacks

Atishay Jain
16110024
IIT Gandhinagar

Ritik Dutta
16110053
IIT Gandhinagar

Varun Gohil
16110059
IIT Gandhinagar

Shreyas Singh
16110150
IIT Gandhinagar

ABSTRACT

Recent advances in deep learning (DL) are bringing computer vision and speech recognition techniques to the centre of security-critical systems. Some use cases include computer vision for autonomous cars and face recognition. Many recent works have demonstrated the vulnerability of conventionally trained Deep Neural Networks (DNN) to adversarial examples. To a human observer, adversarial examples are indistinguishable from natural data. However, they can force conventionally trained neural networks to output a wrong prediction on samples that the network otherwise predicts correctly. In this paper, we present two novel strategies using Generative Adversarial Networks and image hashing to defend against adversarial attacks. The GAN-based approach achieves an accuracy of 80% on adversarial samples while the image hashing technique achieves a maximum accuracy of 32.5%, both of which beat the accuracy of a conventionally trained Convolutional Neural Network (CNN). Our experiments suggest that simpler models, while performing slightly worse on non-adversarial inputs are much more robust against adversarial examples compared to more complex networks.

KEYWORDS

Adversarial attacks, neural networks, computer vision, security

1 INTRODUCTION

Deep learning techniques have demonstrated outstanding performance and have achieved near human-like accuracy in many complex learning tasks such as facial recognition. These techniques are considered as black-box models, since it is difficult to interpret how these networks work. However, this has not stopped practitioners from using DL in human-centered systems such as medical imaging, automatic speech recognition and more recently in autonomous navigation. Despite their success, several recent works have shown that such systems can be easily broken by *adversarial examples* [1]. Adversarial examples are generated by adding small perturbations to the inputs that leads the DNNs to predict incorrectly [2]. In case of image-related tasks, the perturbations are small enough to be imperceptible to the human eye, but are sufficient to fool the network on input samples that it would have otherwise predicted correctly. Figure 1 is an example of such an adversarial example.

Adversarial attacks hence pose a risk to the wide-spread adaptation of deep learning models in human-centered systems. A variety of defenses have been proposed to mitigate the risk posed by such adversarial attacks. These defenses can be grouped broadly under four different approaches: (1) *adversarial training*, which involves including adversarial examples while training the networks to make them more robust against such attacks, (2) modifying the training procedure of the classifier to reduce the magnitude of gradients, (3) removing adversarial noise from the input data. [3], and (4) reducing the dimensionality of the input to make it harder for small perturbations to fool the network.

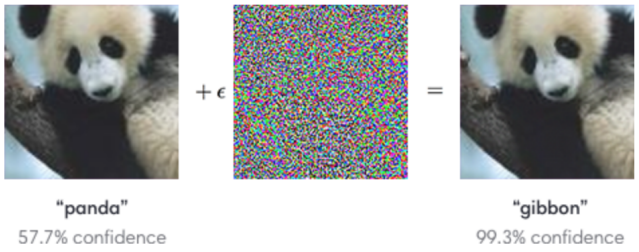

Figure 1: An adversarial Example. Source: OpenAI Blog

In this report, we present two methods to defend against adversarial attacks. The first method uses a GAN to denoise the input images, which is then fed to a classifier. The second method involves perceptual image hashing, which produces a *fingerprint* of the image input, which can then be used to evaluate the similarity between images. Perceptual hashes have the advantage of being tolerant to some amount of noise in the input image. For example, the below two images will produce the exact same hash:

Figure 2: Two images, one of which has a watermark, and their hash values. Source: Towards Data Science Black Box Attacks on Perceptual Image Hashes with GANs

This report is organised as follows. We summarise the theory pertaining to adversarial attacks, and discuss some of the common

attack techniques in section II. A few research works related to defending against such adversarial attacks have been summarised in section III. In section IV, we describe the two methods that we implemented. In section V and VI we describe the experimental setting and the results we obtained from them. In section VII we discuss the insights that we gained from the experimental results. Finally, the conclusions are drawn in section VIII.

2 ADVERSARIAL ATTACKS

Szegedy et al [1] discovered that state of the art deep learning methods were susceptible to misclassifying examples that were only slightly different from the correctly classified examples drawn from the data distribution [4]. These examples are called *adversarial attacks.*

2.1 Measuring Distortions

The common theme across all the attacks that have been proposed so far is the addition of 'infinitesimal' perturbations to the pixel values which end up causing a change that is large enough to fool trained models. That is why it is imperative to understand what metrics are used to measure these changes.

In the image domain, the most common distance of measure used is the family of L_p norms. They are used to quantify the similarity between two images.

The L_p distance is written as $\|x - x'\|_p$ where the p-norm $\|.\|_p$ is defined as:

$$\|v\|_p = \left(\sum_{i=1}^{N} \mid v_i \|^p \right)^{\frac{1}{p}} \tag{1}$$

More specifically:

(1) L_0 distance: gives a measure of the number of pixels differnet in both the images.
(2) L_1 distance: gives the sum of the absolute differences of all pixels.
(3) L_2 distance: gives the sum of the squared differences of all pixels.
(4) L_∞ distance: gives the maximum pixel difference.

The basic intuition behind the attacks is to add noise to each of the pixel values, in the direction where a small change in the pixel value causes the largest change in the prediction of the network.

2.2 Types of Attacks

Adversarial attacks can be classified into three broad classes based on the amount of information that the adversary has about network that is being attacked.

(1) Black-Box Attacks: The adversary only has access to the output $O(x)$ of the network model for any given input. The strategy behind this approach is to learn a substitute for the target model using a synthetic dataset (and the labels obtained by querying the network model), and then using the substitute model to craft the adversarial examples. The adversarial examples for the proxy model are expected to fool the original model as well due to the transferability of attacks between different architectures. [5]
(2) White-Box Attacks: In white box attacks the attacker has access to the model's parameters. It is assumed that the attacker knows everything about the model, including the specific algorithm used, the feature set, the training data, etc.[6]
(3) Grey-Box Attacks: Attacks in which the attacker has more knowledge than a black-box attack, but has lesser knowledge than a white-box attack fall under the category of grey-box attacks.

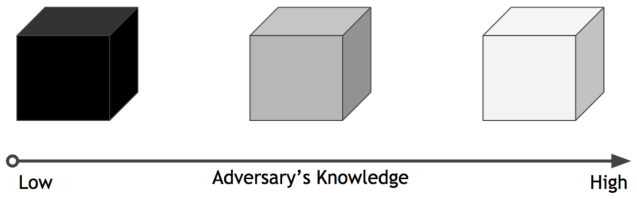

Figure 3: Types of Adversarial Attacks

For this project we assume that the adversary has complete access to the target neural network, including it's network parameters such as the number of layers, objective function used, etc. Hence, our method falls under the white-box scheme.

2.3 Implemented Attack Models

For this project we experiment with the following three models:

(1) **L-BFGS:** Proposed by Szegedy et al. [1] adversarial examples are generated using box-constrained L-BFGS which is an optimisation algorithm. The first step in the method is to calculate the gradient of the cost with respect to the input pixels and the goal is to maximize the loss holding the parameters constant. After doing this iteratively for a fixed number of times, the obtained gradients can be used to obtain new values for all the pixels, producing an image that can cause the model to missclassify it.
(2) **FGSM:** Proposed by Goodfellow et al. [4], this is similar to *L-BFGS* with the key difference being that only one step is taken along the gradient. As a result, this method is significantly faster.
(3) **CW-Attacks:** Proposed by Carlini et al. [7] is a method that directly optimizes for having the minimal distance from the original example while having extremely high missclassification rates. This is a costly attack, since it involves solving a nontrivial optimization problem.

3 RELATED WORK

DefenseGAN by Chellappa et. al. [3] proposed using a GAN to model the distribution of unperturbed images. At inference time, it finds a close output to the given input image which doesn't have the adversarial changes. This output is then fed to a classifier. Szegedy et. al. [1] provided a linear explanation for the existence of adversarial examples, where they describe how making many small infinitesimal changes to the input can lead to one large change in the output. In [5], Papernot et. al. introduced the first practical demonstration of a black-box attack. In [9] the authors use an

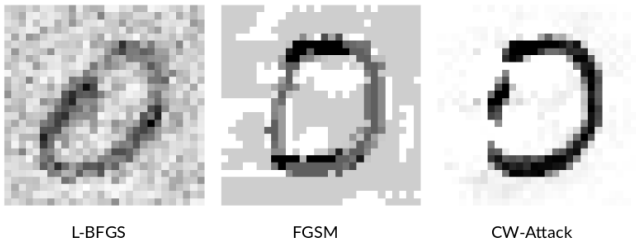

Figure 4: Effect of different Attack Models

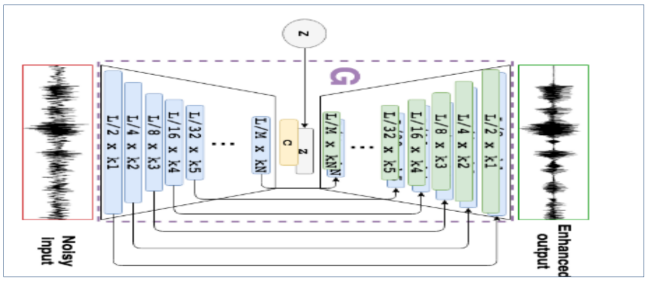

Figure 5: The IEGAN Architecture

autoencoder to perform the function of denoising the adversarial inputs. [19] introduced a defence mechanism called defensive distillation, which instead of using hard labels to train the classifier, uses the softmax output labels of one network to train another network. The authors claimed that this helped the model to learn a more generalised representation of the input space. [20, 21, 22] are survey papers that explores some of the recent attacks and defense techniques.

4 DEFENSE METHODS

4.1 IEGAN - Image Enhancing Generative Adversarial Network

One way of looking at defense against adversaries is as if it were a denoising problem. However, they are slightly different, since the perturbations depend on the sample that they are being added to (in our case the MNIST dataset). Image Enhancing GAN (IEGAN) is our exploratory experiment to see how well an adaptation of a denoising network called Speech Enhancing GAN (SEGAN) [10] would work in removing adversarial perturbations.

SEGAN is used for denoising speech signals. It has two components:

- Generator - For this network, the generator is an encoder-decoder model. An input speech signal of size S is passed through the encoder which maps it to a $\frac{S}{M}$ size space. For every subsequent layer the vector size is halved. Let N be the number of such convolution layers ($N = log_2 M$). Let the $\frac{S}{M}$ size encoded output be called the 'thought' vector c. A latent representation z of size $\frac{S}{M}$ is appended to c. z is a random vector which represents the unit, zero-mean Gaussian distribution. This concatenated vector is then passed through the decoder and the output obtained is the denoised signal. To every layer of the decoder, the input to the layer is concatenated to the output of the corresponding encoder layer i.e. the last decoder layer's input is concatenated to the first encoder layer's output, the second last decoder layer's input to the second encoder layer's and so on. The number of convolution layers in the decoder part is also N. The structure of the generator is depicted in the figure Fig 5
- Discriminator - The structure is similar to the encoder part of the Generator wherein it takes in an input of size S and maps it to a size $\frac{S}{M}$. This lower-dimensional vector is passed through a fully connected layer to get a size 1 output (which represents whether the discriminator thinks that the given input is a real denoised input or the generator's output). Just like the encoder and decoder parts of the generator, the number of convolution layers is also N.

For the SEGAN, $S = 8192$, $M = 1028$, $N = 10$.
In our example, the input is an image i.e. a 2-D input. Therefore, we flatten the input and pad zeros so that the size becomes divisible by 2. The images of MNIST are of size 28×28, and therefore equal to 784 on flattening. We pad zeros equally on both sides such that the size becomes 1028. Just like SEGAN, we also want to keep the size of the 'thought' vector (and hence the latent representation) to be equal to 8. Therefore, IEGAN has the following dimensions : $S = 1024$, $M = 128$, $N = 7$.

4.2 Image Hashing

Perceptual image hash use special hash functions that are used to produce a snippet or 'fingerprint' of various forms of multimedia. Perceptual hash functions are somewhat tolerant to noise in the data, which is unlike conventional cryptographic hash functions, in which even a small change in the input creates a drastic change in the output value. [16]

For our experiments, we used the ImageHash Python library [11], which provides implementations for four different hash functions which have been described below:

(1) aHash - Convert the input to an 8×8 grayscale image, and calculate the mean pixel value for the entire image. If a particular pixel's value is larger than the mean value, assign its position 1, otherwise assign it a 0.
(2) pHash - it is similar to aHash, but first it performs a Discrete Cosine Transform (DCT). However, unlike aHash, it work with the frequencies instead of the pixel value.
(3) dHash - It computes the difference between adjacent pixels. A bit is set to 1 if the pixel to the left of it is more brighter than itself.
(4) wHash - It stands for waveletHash. Just like pHash it works in the frequency domain but instead of using DCT, it uses Discrete Wavelet Transform.

[12] and [13] provide a more detailed explanation for the hash functions.

The basic intuition behind using this method is that by reducing the dimensionality of the input, making small perturbations along each of the dimensions is less likely to add up to a large overall change, and hence, reduces the effectiveness of adversarial attacks.

5 EXPERIMENTAL SETTINGS

We use the MNIST handwritten digits dataset [17] for our classification task.

We used DefenseGAN, Defensive Distillation, a normal Convolutional Neural Network (CNN), and a normal Fully-Connected Neural Network (FCNN) as our baselines. For DefenseGAN, we used the implementation by the authors of the paper which can be found here [3]. We decided to implement our own version of Defensive Distillation using PyTorch (the original version is written using Keras and TensorFlow). However, later on we had to drop this method since we couldn't find an exact equivalent for a function used by the authors in their implementation.

We used three attack techniques in our experiments: L-BFGS, FGSM and CW. We implemented our own version of L-BFGS based on the explanation provided HERE, while for FGSM and CW attacks we used the CleverHans library [8].

The hash functions used for the image hashing method were obtained from [11]. The hashed images were then used as inputs for three different approaches - K-Means Clustering, FCNN and a CNN. Since the inputs were 1-0 binary vectors, we noticed that while using the CNN, the values would saturate to 1's or 0's, thus the overall accuracy was very low. As a result we dropped it from the final comparisons.

The implementation for IEGAN is based on the implementation of SEGAN which can be found here [18]

We implement the *L-BFGS* attack from scratch while the examples for the other two attacks are made by implementing the attack models described in the *Cleverhans* [8] library.

Across all the papers that we read, the only metric used was the classification accuracy. The experimental setup is as follows: first we train and test the classifcation accuracy of our models on the original data. Next, we test the accuracy of the models on the adversarial inputs. Finally, we train the models on a mix of adversarial and non-adversarial inputs and check the accuracy on adversarial and non-adversarial inputs.

Attack Model	Underlying Learning Model	Accuracy of underlying model on Adversarial inputs
L-BFGS	FCNN ($784 \times 30 \times 10$)	45.9 %
FGSM	CNN	4.5 %
C&W	CNN	0 %

Table 1: Accuracy of Attacks on the models used to generate them

Attack Model	Classification Model	Misclassification rate for FCNN
L-BFGS	FCNN	45.9 %
FGSM	FCNN	5.1 %
C&W	FCNN	80 %

Table 2: Accuracy of Generated Examples on FCNN

5.1 Approach

5.1.1 IEGAN. Training - First the IEGAN was trained to remove noise. For training, the input used was an adversary while the ground truth was the corresponding non-perturbed image. After training, the model was used for denoising adversaries of similar images and the same attack.
Testing: - The IEGAN was only used to remove noise from the input images (which might or might not have adversarial noise) and not to classify the images, which was done separately using a classifier. Adversarial images were flattened and padded with an appropriate number of zeros. The reshaped adversaries are then passed through an IEGAN trained on the same dataset. The denoised output of IEGAN was stripped of the padded zeros and resized to the original image size. This output was then passed through the classifier.
To see whether the model worked, we compared the accuracy of the classifier under three conditions:

- When there was no attack
- When there was an attack but no defense
- When there was an attack but the adversaries were first passed through the trained IEGAN model

For training, we used 50,000 MNIST images altered by the FGSM attack and their corresponding un-altered images. The model was trained for 25 epochs. the optimiser used for both the generator and discriminator was the Adam optimiser with a learning rate of 0.0001.
For testing, we used 10,000 MNIST images altered by the FGSM attack and their corresponding un-altered images. We used two classifiers:

- A 3 layered Convolutional Neural Network followed by a fully connected layer of size 10
- A 4 layered Fully Connected Neural Network with size: [724, 200, 200, 10]

Figure 6: From left to right: the original image, image after FGSM attack, denoised output from IEGAN

5.1.2 Image Hashing. We used the same train and test set as used for IEGAN. Images in both the set are first converted to their hash values using the four hash functions.

For the k-means algorithm, we used the Hamming distance as the distance function. The k-means algorithm was run for 25 iterations. In the testing phase, the distance of the test image to the centroid of all the ten classes was calculated. The test image was assigned the class whose centroid had the minimum distance to the test image. Apart from testing all the hash functions individually, we also used a majority voting scheme where the votes were based upon the centroids obtained using all the four hash functions separately. The results in the table 4 correspond to the majority voting scheme.

Classifier Model	Accuracy on normal input	Accuracy on adversarial inputs with no defense	Accuracy with IEGAN	Accuracy DefenseGAN
FCNN based	95.5 %	17.8 %	80.9 %	98.8 %
CNN based	96.3 %	8.3 %	81.5 %	98.0 %

Table 3: Effect of IEGAN on Adversarial Inputs generated by the FGSM Attack

Input Sample	Accuracy on using Hashed inputs	Accuracy on using unhashed inputs
Normal Inputs	85.80 %	94.90 %
Adversarial Samples	32.50 %	54.10 %
IEGAN Samples	65.00 %	80.00 %

Table 4: Experimental results on using hashed images with k-means clustering

Input Sample	Accuracy on using Hashed inputs
Normal Inputs	61 %
Adversarial Samples	36 %
IEGAN Samples	45 %

Table 5: Experimental results on using hashed images with a neural network

In the second approach, we used a 2 layer neural network to classify the test images. The training data was the 1-0 binary vectors obtained from the hash function of size 100 (padded in case the size was smaller than 100). The network architecture was [100, 40, 10]. The table 5 shows the accuracy obtained using this method.

6 RESULTS AND DISCUSSIONS

From the experimental results, one distinct inference that can be drawn is that even though more complex deep learning models have a very high accuracy on normal inputs, they have the largest drop in accuracy when faced with adversarial inputs. On the other hand, much simpler methods such as k-means clustering, had a much smaller drop in accuracy, although their accuracy on the normal inputs was lower compared to the deep learning models.

There are different view points on the transferability of adversarial attacks, with [14] claiming that adversarial attacks are indeed transferable, while [15] reported that this wasn't the case in most of the attacks they tested out. Our own experiments seem to support the latter conclusion. From our experiments, it seems that the easier attack methods, in our case FGSM, are more likely to be transferable, while more sophisticated attacks such as C&W are much harder to be transferable.

Even though the methods we implemented couldn't beat the state of the art, their accuracy on adversarial inputs was much higher than for normal CNNs. The superior performance of our methods could perhaps be explained by the fact that since the number of parameters that our models had to learn were extremely low, they were better able to generalise over the training examples.

From table 4 and 5 we can infer that using the output of the IEGAN as an input to the image hashing method improves its performance, which tells that the IEGAN did a good job in denoising the images from the adversarial noise.

7 CONCLUSIONS

In this paper we explored two methods to act as defenses against adversarial attacks. To the best of our knowledge, image hashing has so far not been used as a defense strategy. Our experiments show that simpler models are more effective in defending against adversarial attacks as compared to much more complex, deep learning methods. Our experiments also suggest that not all kinds of attacks are transferable. The effectiveness of image hashing and other similar dimensionality reduction techniques could further be explored as future work.

8 REFERENCES

(1) Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013).

(2) Li, Xiang, and Shihao Ji. "Defense-VAE: A Fast and Accurate Defense against Adversarial Attacks." arXiv preprint arXiv:1812.06570 (2018).

(3) Samangouei, Pouya, Maya Kabkab, and Rama Chellappa. "Defense-gan: Protecting classifiers against adversarial attacks using generative models." arXiv preprint arXiv:1805.06605 (2018).

(4) Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." arXiv preprint arXiv:1412.6572 (2014).

(5) Papernot, Nicolas, et al. "Practical black-box attacks against machine learning." Proceedings of the 2017 ACM on Asia conference on computer and communications security. ACM, 2017.

(6) "Class 1: Intro to Adversarial Machine Learning; SecML." SecML, 26 Jan. 2018, secml.github.io/class1/.

(7) Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." 2017 IEEE Symposium on Security and Privacy (SP). IEEE, 2017.

(8) Papernot, Nicolas, et al. "Technical report on the cleverhans v2. 1.0 adversarial examples library." arXiv preprint arXiv:1610.00768 (2016).

(9) Sahay, Rajeev, Rehana Mahfuz, and Aly El Gamal. "Combatting Adversarial Attacks through Denoising and Dimensionality Reduction: A Cascaded Autoencoder Approach." 2019 53rd Annual Conference on Information Sciences and Systems (CISS). IEEE, 2019.

(10) Pascual, Santiago, Antonio Bonafonte, and Joan Serra. "SEGAN: Speech enhancement generative adversarial network." arXiv preprint arXiv:1703.09452 (2017).
(11) JohannesBuchner. "Imagehash." GitHub, 7 Dec. 2017, github.com/JohannesBuchner/imagehash.
(12) "Kind of Like That - The Hacker Factor Blog." http://www.hackerfactor.com/, 21 Jan. 2013, www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html.
(13) "Looks Like It - The Hacker Factor Blog." http://www.hackerfactor.com/, 26 May 2011, www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html.
(14) Papernot, Nicolas, Patrick McDaniel, and Ian Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples." arXiv preprint arXiv:1605.07277 (2016).
(15) Barni, Mauro, et al. "On the Transferability of Adversarial Examples against CNN-based Image Forensics." ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019.
(16) "PHash" PHash.org: The Open Source Perceptual Hash Library, www.phash.org/.
(17) Deng, Li. "The MNIST database of handwritten digit images for machine learning research [best of the web]." IEEE Signal Processing Magazine 29.6 (2012): 141-142.
(18) Santi-Pdp. "Segan." GitHub, 11 July 2017, github.com/santi-pdp/segan.
(19) Papernot, Nicolas, et al. "Distillation as a defense to adversarial perturbations against deep neural networks." 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 2016.
(20) Yuan, Xiaoyong, et al. "Adversarial examples: Attacks and defenses for deep learning." IEEE transactions on neural networks and learning systems (2019).
(21) Liu, Qiang, et al. "A survey on security threats and defensive techniques of machine learning: A data driven view." IEEE access 6 (2018): 12103-12117.
(22) Madry, Aleksander, et al. "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083 (2017).