# Comparison of two adversarial libraries and Defense Model using Conditional Variational Autoencoder

Arun Sivadasan (2581838)[1], Aakash Rajpal (2581266)[2], and Vikram Vashisth (2581724)[3]

[1]University of Saarland, Germany

## I. Abstract

Deep neural network methods for machine learning tasks have been widely adopted in recent years including classification. These have been shown to be prone to adversarial perturbations which are carefully designed small perturbations that may cause legitimate image miss-classification. In this project, first we compare the use of two Adversarial Libraries - FoolBox and IBM's Adversarial Robustness Toolbox (referred to henceforth as 'IBM ART'). And later introduce a Defense model against these attacks using a Conditional Variational Autoencoder. Our proposed defense approach can be used with any classification model and does not alter the layout of the classifier or the training procedure.

## II. Introduction

For the first part of this project, We would compare the efficiency of the two libraries in terms of how accurate and efficient (time taken) they are to generate adversarial entities.

## III. Motivation

Many organizations may be using adversarial techniques to reinforce their models and one of the questions they would like to answer would be - which libraries should be used. There are multiple libraries that are now available for practitioners to use and selection of libraries would be one decision that they would be making. The most common adversarial libraries we found were:

- FoolBox
- IBM's Adversarial Robustness Toolbox (IBM ART)
- Cleverhans
- SecML
- AdLib
- Deep-Pwning

We felt that there would be some value in providing a guidance to industry practitioners who are looking at these libraries and need to take a decision as to which one to use in their project. To this end and to limit the scope of the project, we have covered only FoolBox and IBM ART and decided to arrive at some metrics that would be helpful to these practitioners. Our rationale for the selection of these libraries were the following:

- We had worked on FoolBox[1] on the previous project and were quite familiar with how it works
- IBM's Adversarial Robustness Toolbox[2] has more publications on Google Scholar than Foolbox or CleverHans

Another objective of our project is to explore VAE based defense techniques to defend against adversarial inputs.

## IV. Dataset used

MNIST dataset using Lenet5 model. We will generate some sample adversarial images using both models. We have used Pytorch as the standard library to execute the attacks.

## V. About Foolbox and IBM ART

Both IBM ART and Foolbox are a Python based open-source libraries that can be used to create adversarial examples that fool neural networks.

## VI. Approach with library comparison

The approach we used to compare the libraries is as follows:

1) Identify common attacks in both
2) Generate images from MNIST in the format accepted by the respective libraries
3) Generate adversarial for each attack
4) Calculate detection accuracy of the LeNet model on the adversarials

## VII. Comparison of Foolbox and IBM ART

**Attacks common to both**

Based on our analysis we found 12 attacks that are common in both libraries. Out of these 12 attacks, 10 were relevant to our project and we proceeded to the experiments.

Table: Count of attacks in both

| | **Foolbox** | **IBM ART** |
|---|---|---|
| Common | 12 | 12 |
| Exclusively available | 24 | 10 |
| **Total** | **36** | **22** |

The common attacks used in our project are:

1) FGSM
2) Basic Iterative Method (BIM)
3) ProjectedGradientDescent
4) DeepFool
5) NewtonFool
6) Spatial Transformations Attack
7) SaliencyMap
8) Carlini Wagner Attack
9) Elastic Net Attack (EAD)
10) HopSkipJump
11) VirtualAdversarialAttack*
12) Boundary Attack*

* These two attacks were not relevant to our dataset and the model used. Hence they were not part of the experiment.

After this, we started our experiments as per the methodology described.

In all the 10 attacks, we found that the model's accuracy on the adversarial images were 0% except for Foolbox's implementation of Saliency map. In that case the LeNet model was able to correctly detect 1% of the adversarial images correctly.

The next metric we looked at is how much time the two libraries took to generate adversarial. This is because in an organization that may have to such a library in their ML engineering cycle would prefer a more efficient (faster) one.

Overall, in terms of time taken (measured in seconds), the performance of the two libraries against 100, 1000 and 5000 samples are given below:

Table: Time to generate 5000 samples

| Attacks | IBM | Foolbox |
|---|---:|---:|
| DeepFool | 103.68 | 0.52 |
| NewtonFool | 44.96 | 1.27 |
| Elastic Net Attack | 723.64 | 60.11 |
| FGSM | 0.51 | 3.71 |
| Spatial Attack | 0.4 | 1.18 |
| CarliniWagner | 112.02 | 256.82 |
| BIM | 33.7 | 11.88 |
| Projected Gradient Descent | 32.92 | 46.28 |
| HopSkipJump Attack | 1447.42 | 640.14 |
| SaliencyMap* | 43.87 | 25.67 |

Table: Time to generate 1000 samples

| Attacks | IBM | Foolbox |
|---|---:|---:|
| DeepFool | 102.73 | 0.53 |
| NewtonFool | 45 | 1.27 |
| Elastic Net Attack | 731.19 | 60.6 |
| FGSM | 0.49 | 3.63 |
| Spatial Attack | 0.38 | 1.16 |
| CarliniWagner | 111.59 | 258.16 |
| BIM | 32.18 | 11.68 |
| Projected Gradient Descent | 32.08 | 45.9 |
| HopSkipJump Attack | 1442.42 | 644.38 |
| SaliencyMap* | 44.85 | 24.93 |

All the experiments were performed on a Google Colab instance with CPU. The specifications were as follows:

- Model name : Intel(R) Xeon(R) CPU @ 2.00GHz
- CPU MHz : 2000.146
- Cache size : 39424 KB
- CPU cores : 1
- RAM: 13 GB

## VIII. Discussion about the results on library comparison

### A. Reason for successful adversarials

The reason for adversarials' success ( and hence 0 % success of LeNet to classify correctly) in evading the model could be because the libraries have been internally programmed to manipulate the images until a successful adversarial is found.

### B. Difference in the time taken to generate adversarials

In terms of the turnaround time to generate adversarials, Foolbox proved faster on the below:

Table: Attacks where FoolBox was faster

| Attack | Comparison with IBM ART |
|---|---|
| DeepFool | 190 times faster than IBM |
| NewtonFool | 34 times faster than IBM |
| Elastic Net Attack | 12 times faster than IBM |
| BIM | Almost 3 times faster than IBM |
| HopSkipJump Attack | Almost 2 times faster than IBM |
| Saliency Map | 1.3 times faster |

In terms of the turnaround time to generate adversarials, IBM ART proved faster on four of them.

Table: Attacks where IBM ART was faster

| Attack | Comparison with Foolbox |
|---|---|
| FGSM | 7 times faster |
| Spatial Attack | 3 times faster |
| Carlini Wagner | 2.3 times faster |
| Projected Gradient Descent | 1.4 times faster |

As evident from the results, it seems that in some attacks Foolbox's speed to generate adversarial is significantly faster than IBM ART.

### C. Conclusion from library comparison

With various caveats and the limitations of our project, we could propose the following from the available information:

In organizations looking to deploy one of these libraries to generate adversarial images to strengthen their model, we recommend Foolbox for the below types of attack as a more time efficient alternative:

- DeepFool
- NewtonFool
- Elastic Net Attack
- BIM
- HopSkipJump Attack

Similarly, we recommend IBM ART for:

- FGSM
- Spatial Attack
- Carlini Wagner
- Projected Gradient Descent

**Detection of Adversarial Images Using Conditional Variational Autoencoders**

## IX. Introduction

Susceptibility of Deep Neural Networks to Adversarial attacks is one of the most researched areas in the recent years. Although these networks achieve state-of-the-art performance, sometimes exceeding human-level performance on the test data used for various tasks, their vulnerability is a concern when implemented in real-life applications, particularly in areas such as health care, autonomous vehicles etc.

Here, we propose a Defense Mechanism using a Conditional Variational Autoencoder (VAE) to act as an enhancement to any existing Neural Network and detect whether the incoming test data is an Adversarial image or not. VAE is a generative model - it estimates the Probability Density Function (PDF) of the training data. The model can also sample examples from the learned PDF which is what we enhance to classify whether an image is adversarial or not.[4]

## X. Motivation

Various defenses have been introduced to mitigate the effects of adversarial assaults. Such defenses can be grouped under three different approaches: (1) modification of training data to make the classifier more resilient against attacks, e.g. adversarial training which increases the classifier's training data with adversarial examples . 2) alteration of the classifier's training technique to minimize gradient amplitude, e.g. protective distillation (3) attempting to remove adverse noise from the input samples. All of these methods have drawbacks in the sense that either white box attacks or black box attacks are successful against them but not both.

The proposed Defense Mechanism works against both Black-box or White-box Adversarial attacks and is trained along with the actual Neural Network.

## XI. RELATED WORK AND BACKGROUND INFORMATION

A few pieces of work on defense against adversarial attacks in the existing literature have attempted to use generative models in various ways. Samangouei et al. (2018) propose training a Generative Adversarial Network (GAN) [7] on the training data of a classifier, and use this network to project every test sample on to the data manifold by iterative optimization. However, there are a couple of downsides to using a GAN in a defense state. First, the images are generated off some arbitrary noise. If you wanted to generate a picture with specific features, there's no way of determining which initial noise values would produce that picture, other than searching over the entire distribution. Second, a generative adversarial model only discriminates between "real" and "fake" images. There's no constraints that an image of a cat has to look like a cat. This leads to results where there's no actual object in a generated image, but the style just looks like picture. Thus, this method does not try to detect adversarial samples, and this defense technique has been recently shown to be ineffective (2018). .

MagNet [3] uses autoencoders to detect adversarial inputs but this defense method does not claim security in the white box setting. Further, the technique has also been broken in the grey box setting by recently proposed attack methods. Traditional autoencoders do not limit the latent representation to have a particular distribution.Our use of variational autoencoders helps us to simultaneously protect against adversarial and fooling inputs unlike MagNet. (2017a).

## XII. Variational Auto Encoder Method

We are working with the standard MNIST dataset and hence we know every image of a digit should contain, well, a single digit. An input in $R^{28*28}$ doesn't explicitly contain that information. But it must reside somewhere... That somewhere is the latent space. You can think of the latent space as $R^k$ where every vector contains pieces of essential information needed to draw an image.[6] Let's say the first dimension contains the number represented by the digit. The second dimension can be the width. The third - the angle. And so on. We can think of the process that generated the images as a two steps process. First the person decides - consciously or not - all the attributes of the digit he's going to draw. Next, these decisions transform into brushstrokes. VAE tries to model this process: given an image x, we want to find at least one latent vector which is able to describe it; one vector that contains the instructions to generate x. Formulating it using the law of total probability, we get P(x)=

$$\int P(x|z)P(z)dz.$$

The VAE training objective is to maximize P(x). We'll model P(x$\|z) using a multivariate Gaussian N(f(z),\sigma^2$ I). f(z) will be modeled using a neural network. $\sigma$ is a hyper-parameter that multiplies the identity matrix I. You should keep in mind that f is what we'll be using when generating new images using a trained model. Imposing a Gaussian distribution serves for training purposes only [5].

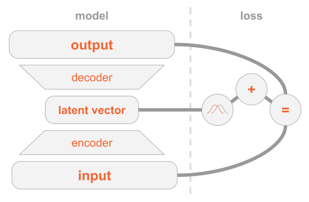

Fig. 1. A Sample VAE Model

On the left side we have the model definition:

- An input image is passed through an encoder network.
- The encoder outputs parameters of a distribution $Q(z\|x)$.
- A latent vector z is sampled from $Q(z\|x)$. If the encoder learned to do its job well, most chances are z will contain the information describing x.
- The decoder decodes z into an image.

On the right side we have the loss:

- Reconstruction error: the output should be similar to the input.

- $Q(z\|x)$ should be similar to the prior (multivariate standard Gaussian).

In order to generate new images, you can directly sample a latent vector from the prior distribution, and decode it into an image.

## XIII. Defense Conditional VAE MODEL

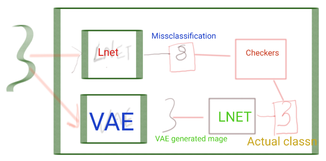

Fig. 2. Defense Model

During the training phase, the Lenet is trained to classify the handwritten digits whereas the VAE is trained together with the same training dataset. The Encoder of the VAE learns to generate latent space for each digit image and the decoder learns to decode this latent vector back to an image.

During the testing phase, we pass an image ( clean or adversarial) to our model. The image is fed to both the Lenet MNIST Model and the VAE model. The Lenet model classifies it accordingly. Whereas the VAE generates an image more suitable to the original data distribution.When we feed the VAE generated image pack to the Lenet model it gives a classification. If the images was an adversarial we would have a difference in classification by the Lenet model for the original image and the VAE generated image. And we use this to detect whether the Original image was an adversarial or real.

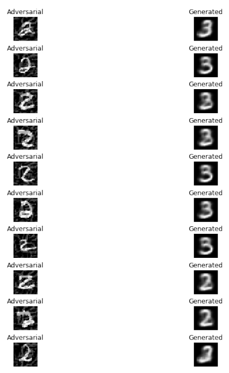

Fig. 3. VAE GENERATED IMAGES
[5]

## XIV. Experimental Observations

We have conducted our training on MNIST Dataset using the Lenet model and the VAE implemented in Pytorch. For the adversarial attacks we are using IBM ART Toolbox and implementing FGSM,Deepfool,CWG attacks. We have 30000 samples for each of the attack method mentioned above. All the below results are being run on NVidia Tesla GPU

Table:Accuracy on adversarial images

| | FGSM | Deepfool | CWG |
|---|---|---|---|
| Lenet model | 9% | 6.7% | 8.9% |
| Lenet + VAE Defense | 91.345% | 91.333% | 91.65% |

As we can see we get an accuracy of around 91% for detection of adversarial images irrespective of the attack method used. This proves our robustness and improvement.

## XV. Potential future directions

We believe our project has a lot of potential for application in the industry and for its unique approach for detecting adversarial. As this was a very limited study, we propose the following potential future directions which can be evaluated to the current limitations and to provide further evidence to validate the conclusions we have made. Future projects can consider the following:

- Replicate the experiments with more data sets and models, with potentially a dataset that will result in creation of a large number of adversarial (in millions) and GPU which will simulate a real ML engineering environment.
- Explore the code used by both and investigate reason for deviation (for example the variation seen in Foolbox and IBM's implementation of DeepFool algorithm).
- Improving the Decoder of the Defense Conditional VAE to generate cleaner images and thus improve accuracy

## References

[1] Jonas Rauber Wieland Brendel. Foolbox documentation site. Available at https://foolbox.readthedocs.io/en/latest/index.html.
[2] IBM Corporation. Ibm art documentation site. Available at https://adversarial-robustness-toolbox.readthedocs.io/.
[3] Hao Chen Dongyu Meng. Magnet: a two-pronged defense against adversarial examples. Available at https://arxiv.org/abs/1705.09064.
[4] Raviraja G. Conditional variational autoencoder (vae) in pytorch. Available at https://graviraja.github.io/conditionalvae/.
[5] Raviraja G. Using autoencoders to prevent adversarial inputs. Available at https://colab.research.google.com/drive/1ky8foTDlb2OeQ1ckgxuydBEAgkex2Ir2.
[6] Arpan Losalka. Resisting adversarial attacks using gaussian mixture variational autoencoders. Available at https://towardsdatascience.com/resisting-adversarial-attacks-using-gaussian-mixture-variational-autoencoders-be98e69b5070.
[7] Rama Chellappa Pouya Samangouei, Maya Kabkab. Defense-gan. Available at https://arxiv.org/abs/1805.06605/.