Robust Classifier using GAN

Capstone Project

Machine Learning Engineer Nanodegree

Prity Roy

June 27th, 2017

I. Definition

Project Overview

Artificial Neural Networks are machine learning model based on the neural structure of the brain. A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers.

Deep Neural Networks (DNN) are known to perform exceptionally well on many categories of machine learning problems, particularly input classification. These networks efficiently learn highly accurate models from a large datasets of training samples, and thereafter classify unseen samples with great accuracy. As a result, DNNs are used in many settings such as automatic speech recognition, image recognition, natural language processing, drug discovery and toxicology and many more of which system are increasingly security-sensitive, such as large scale malware classification, credit card fraud identification, biometric authentication, etc.

However, studies have shown that these algorithms, like in any other machine learning techniques, are susceptible to adversarial examples, small input perturbations in input data leading to incorrect predictions [1]. Such attacks can seriously undermine the security of the system supported by the DNN, sometimes with devastating consequences.

Problem Statement

As more and more applications are using DNN and other machine learning models it is meaningful and urgent to increase the local stability towards adversarial examples. It is necessary to decrease the gap between the machines and the human perception, and devise safety mechanism for security–sensitive applications.

These adversarial examples are either inputs that an attacker has designed to specifically fool the model or data occurring naturally. The vulnerability caused by their existence is somewhat disturbing. In the case of visual data, for example, it would be expected that images that are perceivable to “human eye” be predicted to the same accuracy.

Several works have been proposed to use adversarial examples during training of neural networks, and reported increase in classification accuracy on test data [1], [2]. The goal of this project is to make deep neural network model robust to adversarial examples using such training [3], [4].

Using a generative adversarial network (GAN), first introduced by Ian Goodfellow and others in Yoshua Bengio's lab in 2014 [6], the classical deep neural network is made robust to adversarial examples.

A Deep Neural Network capable of predicting highly accurately is trained under adversarial settings so has to make it robust to adversarial examples. The adversarial setting used in this project is the most improved and highly innovative method used in Generative adversarial networks. It is the Wasserstein GAN[7], which has empirically shown remarkable results.

As Wasserstein GAN does not spit classification as is required in the solution a separate classifier is trained along with it. The structure of this classier will be of a highly accurate model using Deep Neural Network. After the training the classifier now robust to adversarial examples should be able to identify adversarial inputs to a better accuracy than the one not trained in the same adversarial setting. This robust classifier is the expected model.

Metrics

The accuracy of the model is evaluated using the cross entropy loss function, which goes well with back-propagation required to train neural networks for classification problems.

$$\mathcal{L}(X, Y) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)} \ln a(x^{(i)}) + \left(1 - y^{(i)}\right) \ln\left(1 - a(x^{(i)})\right)$$

Here, $X = \left\{x^{(1)}, \ldots, x^{(n)}\right\}$ is the set of input examples in the training dataset,

and $Y = \left\{y^{(1)}, \ldots, y^{(n)}\right\}$ is the corresponding set of labels for those input examples.

a(x) represents the output of the neural network given input x.

As the purpose is to evaluate the robustness of the Classifier, that is, successful identification of digits from a generated image, the above cross entropy loss is used.

GANs are trained to minimize the distance between the generated and true data distributions. Wasserstein GAN uses Wasserstein distance, or the Earth Mover (EM) distance, instead of Jensen-Shannon (JS) divergence used in other GAN training, as the final cost function. The loss function for a WGAN is constructed by applying the Kantorovich-Rubinstein duality (Villani, 2008) to obtain

$$\min_{G} \max_{D \in \mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x})\right] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \left[D(\tilde{\boldsymbol{x}})\right]$$

where D is the set of 1-Lipschitz functions and Pg is once again the model distribution implicitly defined by x˜= G(z), z ∼ p(z).

Sadly, EM distance cannot be computed directly from the definition; however as an optimization problem there exists a dual problem corresponding to this, which is the Kantorovich-Rubinstein duality

Finally, as the objective of this project is a robust classifier, against adversarial examples, the classifier is also tested against the generated images from the WGAN network. The two classifiers, one with adversarial training (*Robust Classifier*) and other without adversarial training (*Classic Classifier*) are tested for their performance against samples generated by the WGAN at different epochs. The generated samples slowly resemble the original dataset and the accuracy of the classifiers are recorded and analyzed as the final metric.

II. Analysis

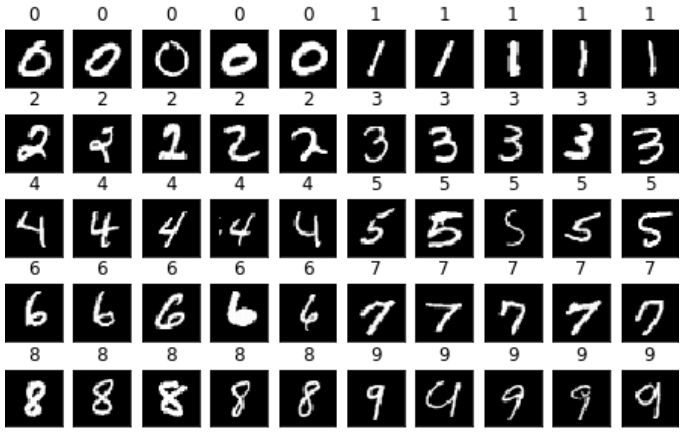
Data Exploration

Datasets and Inputs

In the experiment, MNIST database is used, a subset of a larger set available from NIST. The MNIST, handwritten digits, dataset is split into three parts: 55,000 examples of training dataset, 10,000 examples of test dataset, and 5,000 examples of validation dataset.

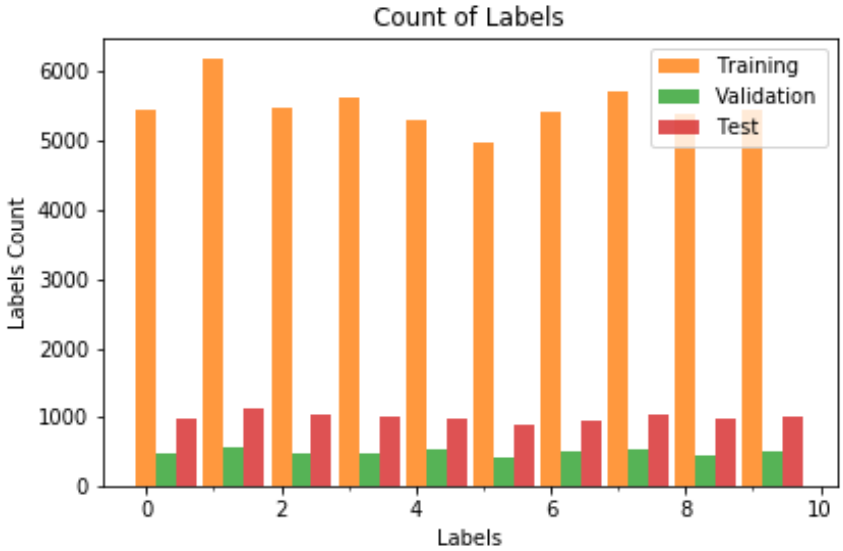
Every MNIST data point has two parts: an image of a handwritten digit and a corresponding label. The MNIST data is hosted on Yann LeCun's website [5].

The digits have been size-normalized and centered in a fixed-size image. Each image is 28 pixels by 28 pixels. We can interpret this as a big array of numbers. The random digits from the MNIST dataset in the figure below.

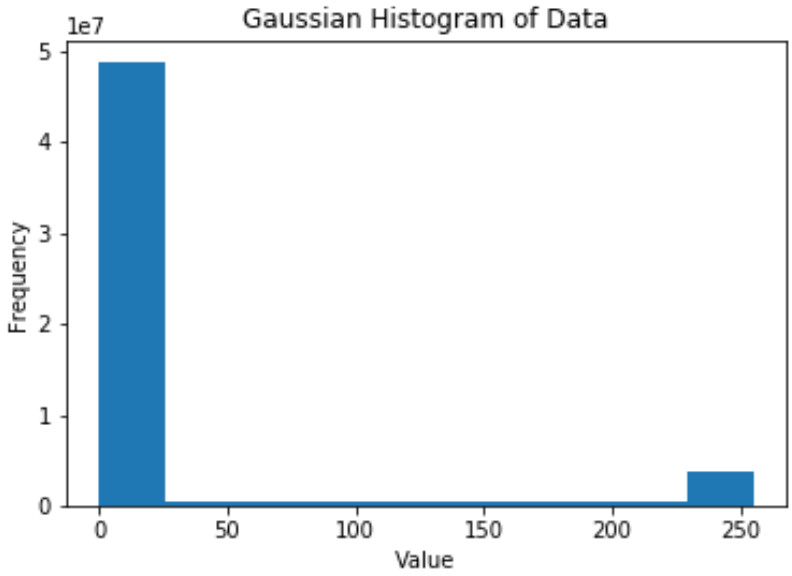


MNIST Dataset

Exploratory Visualization



From the above figure it is evident that there are even counts of digits in the different datasets of the MNIST dataset. This will allow us to train and test our model without any biasness towards any digit.



As it is evident from the above figure, the image contains primarily pixels of value zeros, indicating that most part of the image is dark and does not contain any features. While a one-tenth of it contains non-zero values.

Algorithms and Techniques

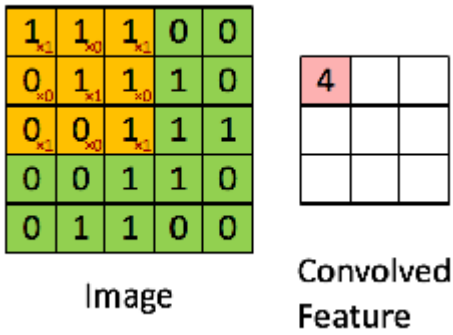
A set of Convolution Network with multiple layers, also known as deep neural networks, are the building blocks of the solution to this problem.

Each of the models are Deep Neural Networks which are either trained independently or together to achieve the objective of the project.

A model comprising only of a deep neural network forms the *Classic Classifier*, and another network containing multiple networks modeling in a game between the different players also known as **Generative Adversarial Network** is implemented, to output a deep neural network called the *Robust Classifier* in this setting.

Convolutional Networks

Convolutional networks are similar to artificial neural networks as it also has learnable weights and biases. But the special characteristic of such network is that it does not learn the entire feature in one setting but learns it in parts developing different levels of feature extraction for complicated input features such as the image. The features are extracted in multiple layers forming many activation maps. These extracted features may be different forms of the feature that distinguishes it from the other feature.



A filter of particular size less than the dimension of the input feature is used to convolve around the input feature to extract details. These extractions form outlines in case the input features are images. Consequently these networks are capable of identifying objects without spatial restriction, that is, a cat can be identified even if it is in the top left corner of the image in one and in the center in the

other image. This remarkable characteristic of Convolutional networks has made them a winner in input identification. It was first used by Yann LeCun in an Imagenet Challenge.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a powerful class of generative models that cast the generative modeling problem as a game between two adversary networks: the generator network produces synthetic data given some noise source and the discriminator network discriminates between the generator's output and true data.

The generator network generates an adversarial perturbation that can easily fool the classifier network by using a gradient of each image. Simultaneously, the discriminator network is trained to classify correctly both original and adversarial images generated by the generator. These procedures help the classifier network to become more robust to adversarial perturbations.

However, GANs are difficult to optimize and the training is unstable. Mode dropping is another typical issue in GANs, and with near-meaningless learning curves making it difficult to debug.

The recent introduction of Wasserstein GAN has shown remarkable performance. It has been known to resolve the GANs mode collapse problem and also considerably requires reduced training time.

Thus, in this project Improved Wasserstein GAN [8], technique will be used to train the classifier to make it robust to adversarial examples.

Wasserstein GAN

GANs are trained to minimize the distance between the generated and true data distributions. Initially, the Jensen-Shannon divergence was used as this distance metric. However, in Wasserstein GAN (wGAN) the cost function used is Earth Mover's (EM) distance.

Improved training of Wasserstein GANs enables very stable GAN training by penalizing the norm of the gradient of the critic with respect to its input instead of clipping weights. This 'gradient penalty' is simply added to the Wasserstein distance described above for the total loss. The training time of the GAN is also less.

The classifiers are trained with and without adversarial training and evaluated against original and generated images.

Benchmark

In the beginning the *Classic Classifier* and the *Robust Classifier*, after their training, are tested against both the MNIST testing dataset.

The classifiers are further tested against the generated samples from the Generator. As, the intention of the project is to obtain a robust classifier, both the classifiers are tested against the images generated by the network at different stages of the training. The Generator gradually learns to imitate the dataset and in the goal to finally output a near to original image.

The output at separate intervals from the gradually learning Generator are hard to predict even for a human eye and acts as a perfect testing ground for the two classifier. The classifiers are test against these samples generated at different training epochs. The accuracy of the classification of the digits from both the classifier is compared.

III. Methodology

Data Preprocessing

As the MNIST dataset is a preprocessed dataset, widely used for training and testing in the field of machine learning, the dataset is already split into training, validation and test datasets. The images are size-normalized in a fixed square, with 28 X 28 pixels. The images are grayscale with a single channel. The color channel has the values of range 0 to 255.

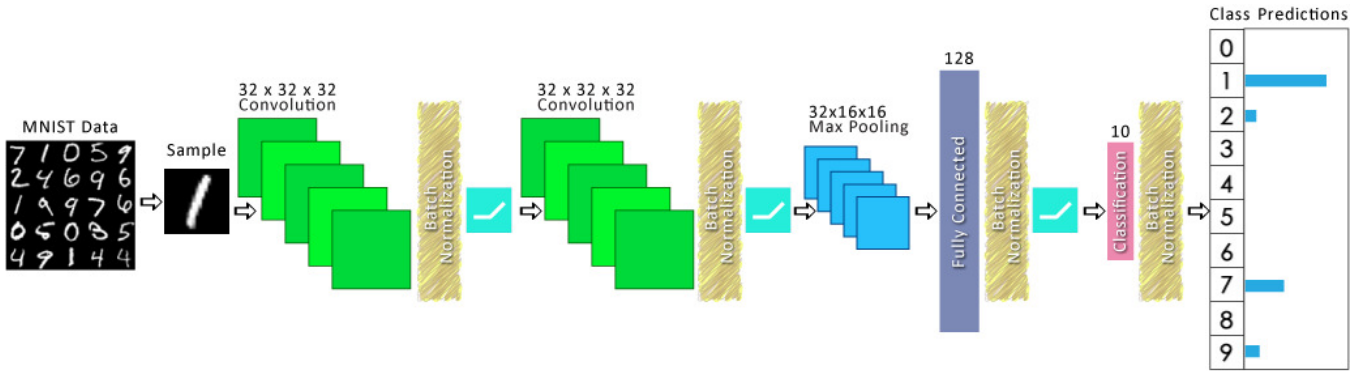
For the purpose of this project each image is again padded on all sides by 2 pixels. So, the size of the images used and generated is 32 pixels by 32 pixels and the values are normalized in the range -1 to 1.

Implementation

The DNN Classifier is trained against two settings. Initially the DNN classifier dubbed *Classic Classifier* is trained using a preprocessed training dataset of the MNIST dataset. And, then the DNN Classifier of the structure similar to that of *Classic Classifier* is trained against adversarial settings, using Improved Wasserstein GAN Techniques, making it the *Robust Classifier*.

Classic Classifier

The architecture of the Deep Neural Network for *Classic Classifier* is exhibited by the figure below.



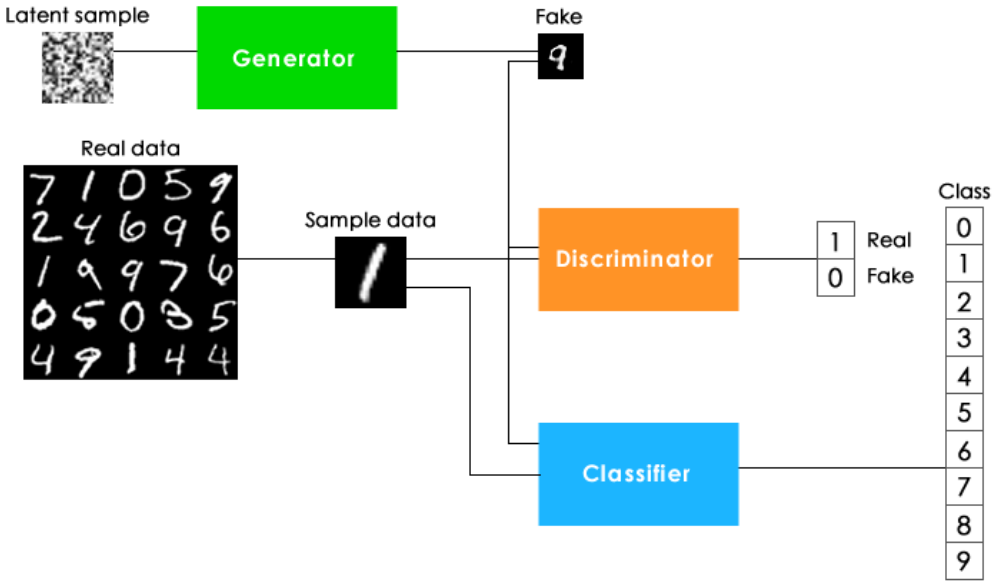
There are two convolution layers used with relu activation and batch normalization. Then the layers are sub-sampled using max pooling, which is later passed through a fully connected layer of size 128. This layer is batch normalized to another fully connected layer of size 10, again batch normalized. The output of this convolution layer is our classification.

- SAMPLE [32x32x1] containing the pixel values of the MNIST grayscale image, of 32x32
- The two Convolution layer converts the input given to it into 32x32x32, which means, it creates 32 layers of size 32x32 using kernel (3,3) and stride (1,1)
- The layers are then Batch Normalized with an activation function of RELU
- After the two layer of Convolution, a Max pool layer is applied which down-samples the dimensions (width, height), resulting in a [32x16x16].
- There are two fully-connected layer applied with sizes 128 and 10 respectively. Each is batch normalized as done with pervious two layers of Convolution Layer. The last fully connected layer computes the class scores, resulting in size of [1x10], where each of the 10 numbers correspond to the scores of the classes, that is the digits prediction.

Adversarial Setting:

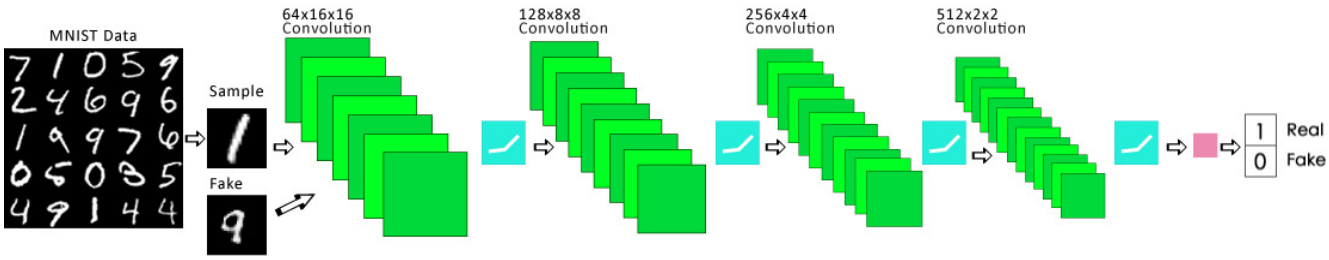
Generative Adversarial Network also uses Deep Neural Network, known as DCGAN or Deep Convolution Generative Adversarial Networks.

The Generative Adversarial Network along with the Classifier:



The above figure is the structure of the **GAN** architecture used in the project. **Wasserstein GAN** is used to train the *Robust Classifier* to make it robust to adversarial examples. As the Discriminator or also called Critic, only spits binary output, that is, if the input data is real or fake and does not classify the digits, a separate classifier is trained on the real and fake data to classify on the handwritten digits. The classifier uses the same structure as the *Classic Classifier* and becomes robust to adversarial examples after training. The different models used in this particular Generative Adversarial Networks are explained below.

Discriminator Model



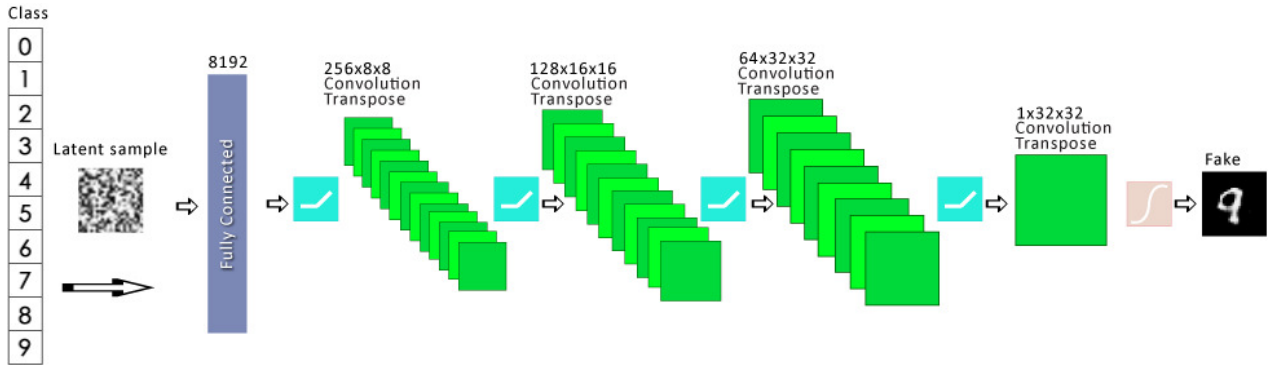
The Discriminator takes in either the real data, here MNIST data samples, or the fake data, here samples generated from the Generator, and outputs a single value. This value allows us to calculate the loss function.

- Input of real data or fake data is passed through the discriminator during the training process, of size (1x32x32)
- As in Wasserstein GAN the *Discriminator* or critic does not use *Batch normalization* and used *leaky relu* as the activation function, there are four consecutive layers of convolution of

sizes (64x16x16), (128x8x8), (256x4x4) and (512x2x2) respectively, without batch normalization.

- The output is then passed through a fully connected layer of size 1 which forms the classification and spits binary output.

Generator Model



The *Generator* takes in a latent variable and random class as its input and spits out a grayscale image.

- Random class and a latent sample of size (1x32x32) is passed through the generator.
- A fully connected layer is used to increase the input size to 8192 which is then passed to series of four Convolution transpose
- It uses *Convolution transpose*, which upsamples the data to larger size so as to achieve a size of the image to be outputted.
- As with *Discriminator* in **Wasserstein GAN** the Generator also does not use *Batch normalization*, but uses *relu* as the activation fuction.
- There are four consecutive layers of *convolution transpose* are of sizes (256x8x8), (128x16x16), (64x32x32) and (1x32x32) respectively.
- *relu* activation is used after the all the layers except the last layer which uses *tanh* to spit data which is the generated image.
- The last layer outputs the generated image.

The above models are implemented in three stages:

Firstly, the *Classic Classifier*, using the above architecture, is trained on MNIST dataset with a goal of attaining very high accuracy in its test dataset.

Secondly, another classifier using the same architecture as the *Classic Classifier* is trained in adversarial setting. The above Wasserstein Generative adversarial network (GAN) setting is used to train the classifier to make it robust. This classifier trained on the real and fake data to classify on the handwritten digits becomes the *Robust Classifier*.

When the GAN network is being trained the Generator learns to imitate original images. At intervals of thousand epochs the Generator generates images based on its learning. These generated images form the basis of our testing in the final stage of the implementation.

Finally, the two networks are evaluated against both the MNIST test dataset and generated samples by the Generator and reported. This model trained under adversarial training is robust to perturbations.

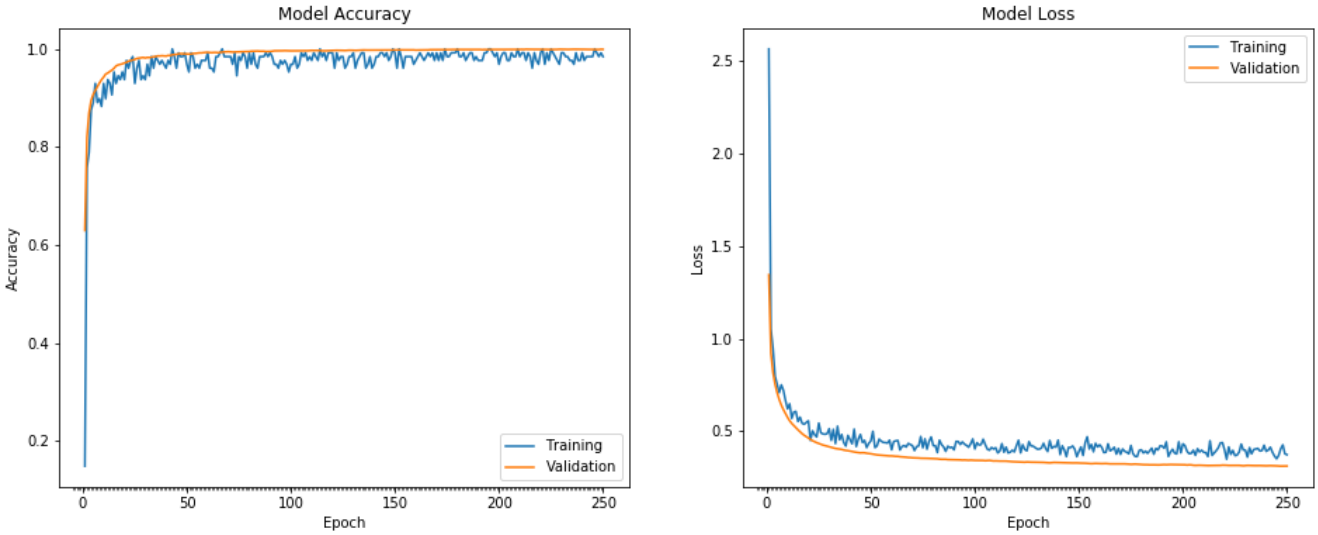
Refinement

- The Classifier is modified using Batch Normalization to get higher accuracy, from the Drop out used earlier in the project. There was a change of 1% in accuracy however the training time reduced to half.
- After the use of Batch Normalization in the Classifier the convolutions was made shallower so as not to overfit.
- Earlier a weight initialize of random normal between 0 and 0.04 was used, which was changed to Xavier initialization. The epoch for the GAN training reduced considerably.
- The scopes of the variables were defined clearly as it did not allow the two models, *Classic Classifier* and the *Robust Classifier* to be loaded in one session.
- Changed in the model restoration process so as to call it only once during the ipython session.

IV. Results

Model Evaluation and Validation

Training of Classic Classifier



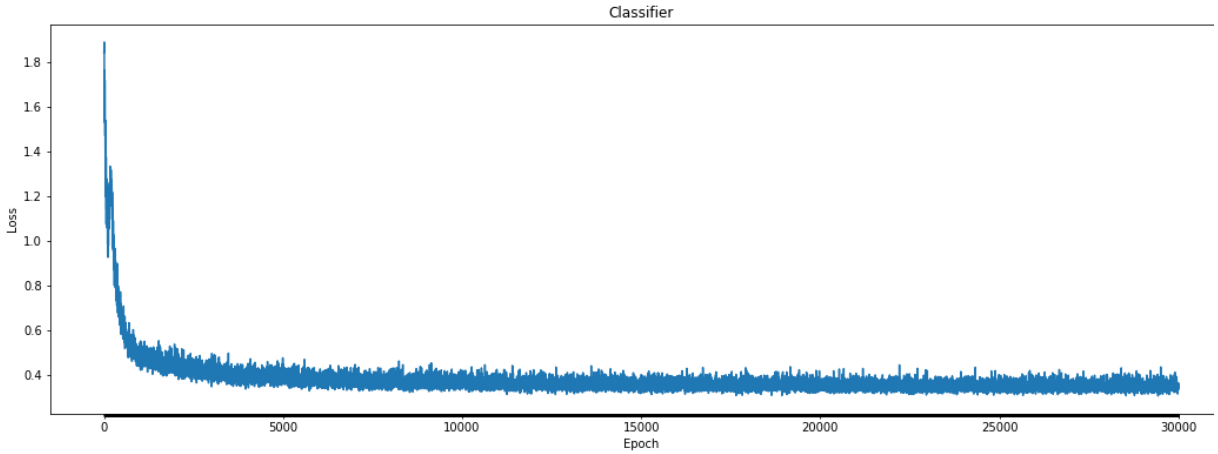
From the above curves it can be perceived that the model has been trained with high accuracy. The training accuracy reaches 100% in less than 50 epochs, while the validation accuracy hovers around the 99% mark. This understandably suggests that the model is highly accurate and that ***Deep Neural Networks*** perform with great accuracy as is known to us.

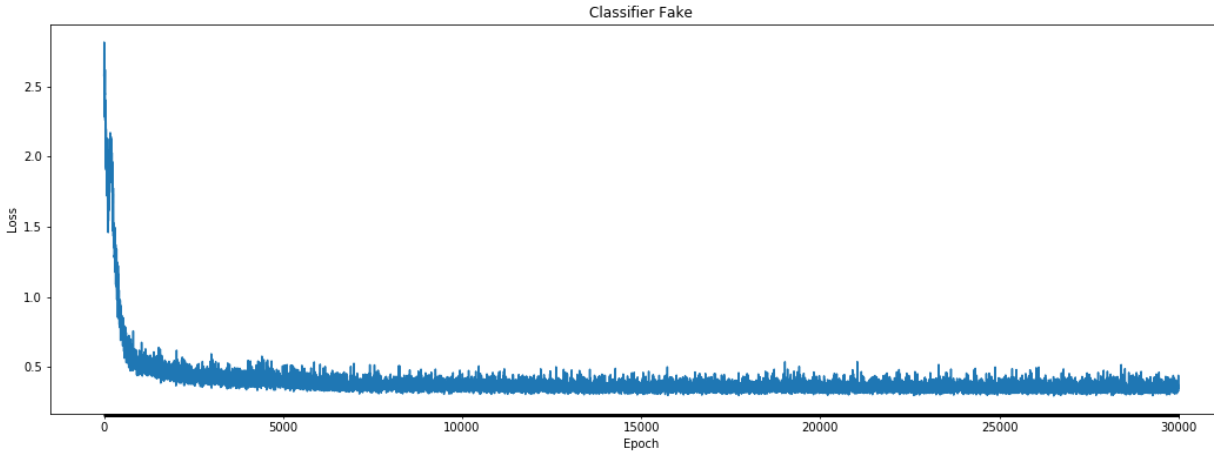
Training the Robust Classifier

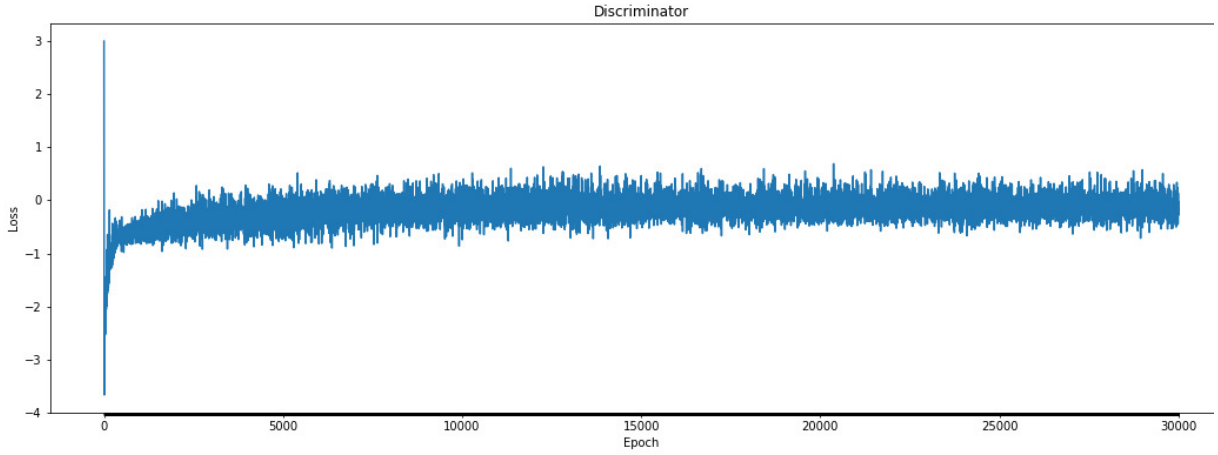
The chart below demonstrates the training curves of the **Wasserstein GAN**. The loss of the different models at the epochs can be seen.

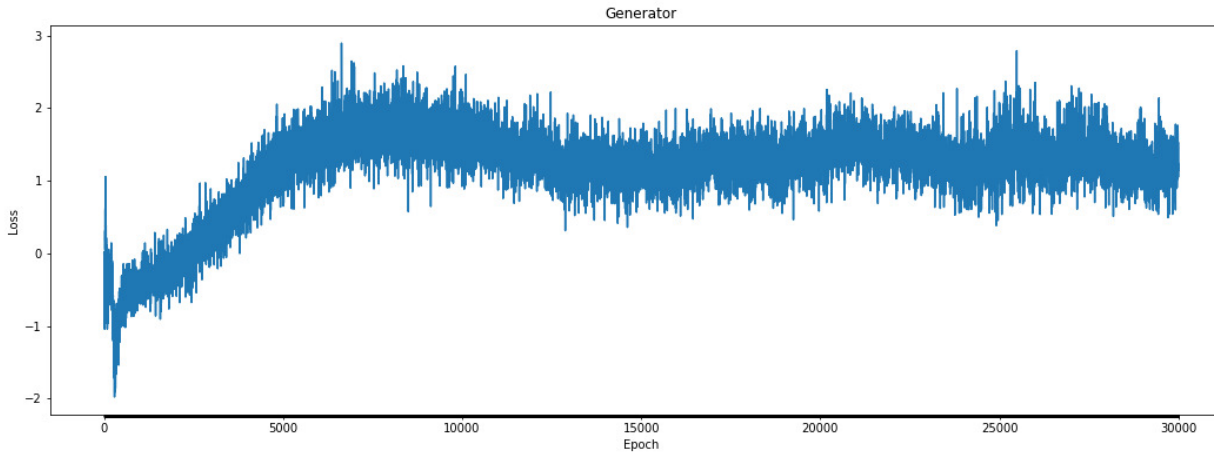
The *Classifier* and the *Discriminator* reaches a plateau after few thousand epochs while the *generator* requires several thousand epochs as the loss starts to increase in the beginning and then falls later to plateau at the end of the epoch.

The curves suggest that the **Improved Wasserstein GAN** implemented in the project required very little epochs to train, than other GANs to achieve near to original image generation.


Classifier
Loss
Epoch
0
5000
10000
15000
20000
25000
30000
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8


Classifier Fake
Loss
Epoch
0
5000
10000
15000
20000
25000
30000
0.5
1.0
1.5
2.0
2.5


Discriminator
Loss
Epoch
0
5000
10000
15000
20000
25000
30000
−4
−3
−2
−1
0
1
2
3

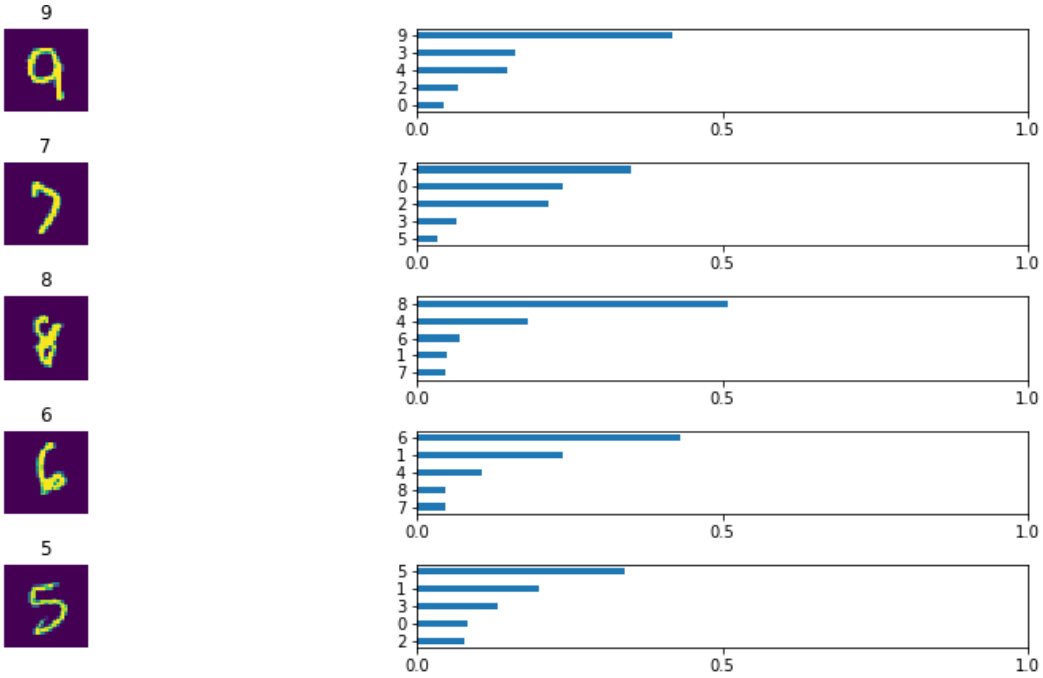

Generator
Loss
Epoch
0
5000
10000
15000
20000
25000
30000
−2
−1
0
1
2
3

The Discriminator oscillates at zero after a few thousand epochs because in Improved Wasserstein GAN, it is trained five times more than the *Generator*. The *Classifier* is trained half times than the *Generator*.

The *Generator* however demonstrates a curve which is not intuitive as it is with every GAN. Since, the loss of the *Generator* increases as soon as it reaches zero, it becomes necessary to train it further increasing the epochs, even when the loss of *Discriminator* and the *Classifier* plateaus. After around 13000 epochs the Generator starts to oscillate between 0 and 2 and remains there however more epochs are added, evident from the graph below. Thus the training is ended after 30000 epochs. The result of the GAN is, however, very promising with near to original imitation.

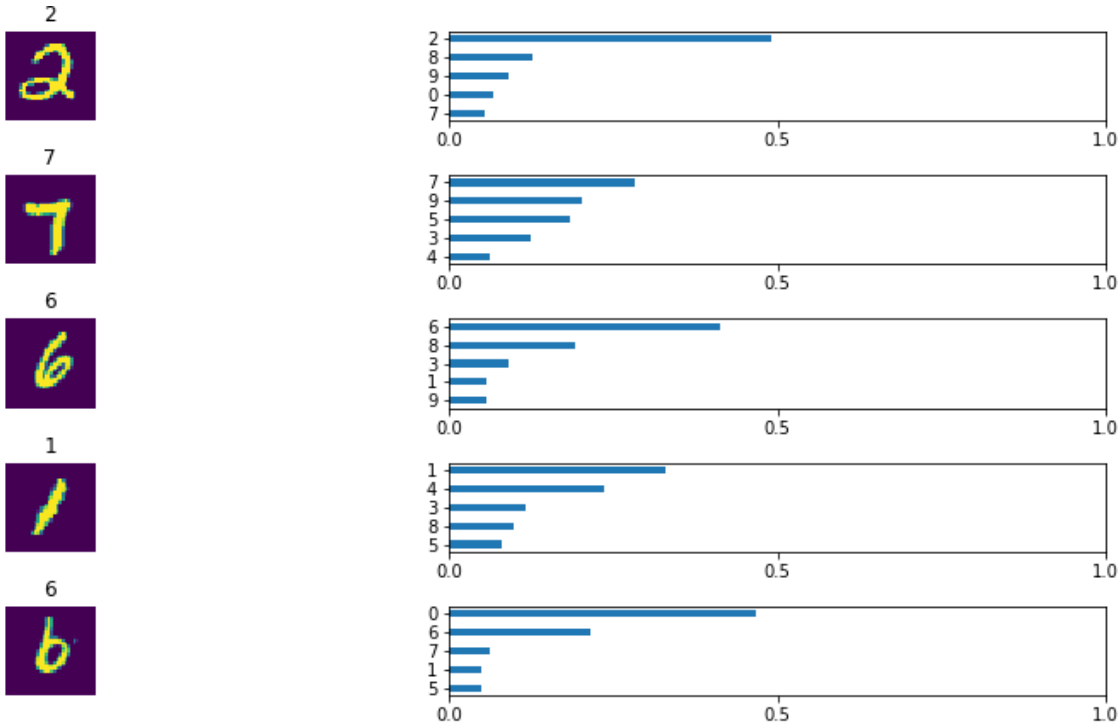
The *Classifier* trained along with the *Generator* and *Discriminator* also becomes robust with very high performance.

Testing Classic Classifier Against MNIST Dataset



The model when tested against gives accuracy of 98.7%. The top 5 predictions of randomly selected images gives us 100% correct predictions, again, demonstrating a highly accurate model.

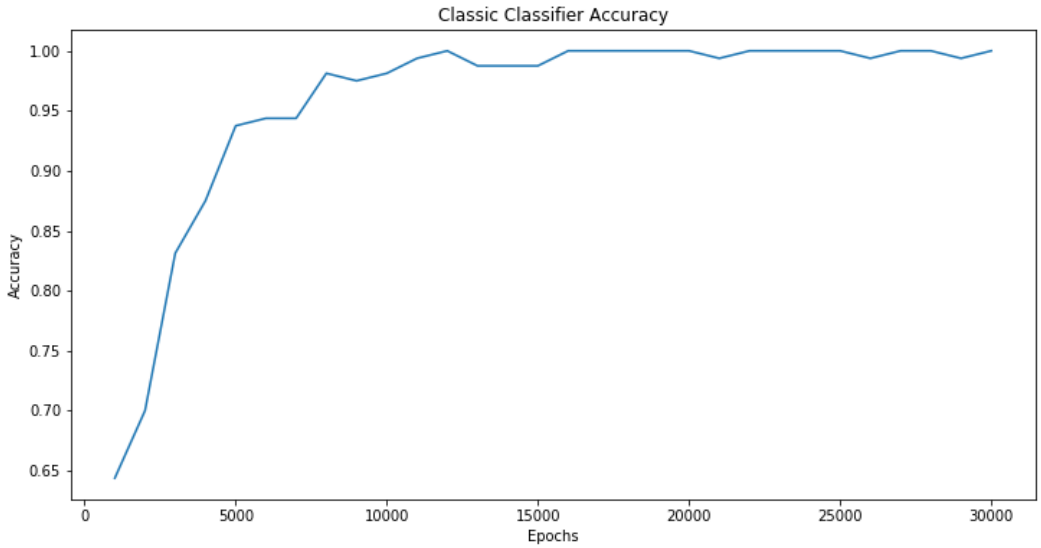
Testing Robust Classifier against MNIST Dataset



The model when tested against gives accuracy of 99.16%. The top 5 predictions of randomly selected images give us good predictions. This suggests that the *Robust Classifier* is better and has better accuracy than the *Classic Classifier*

Justification

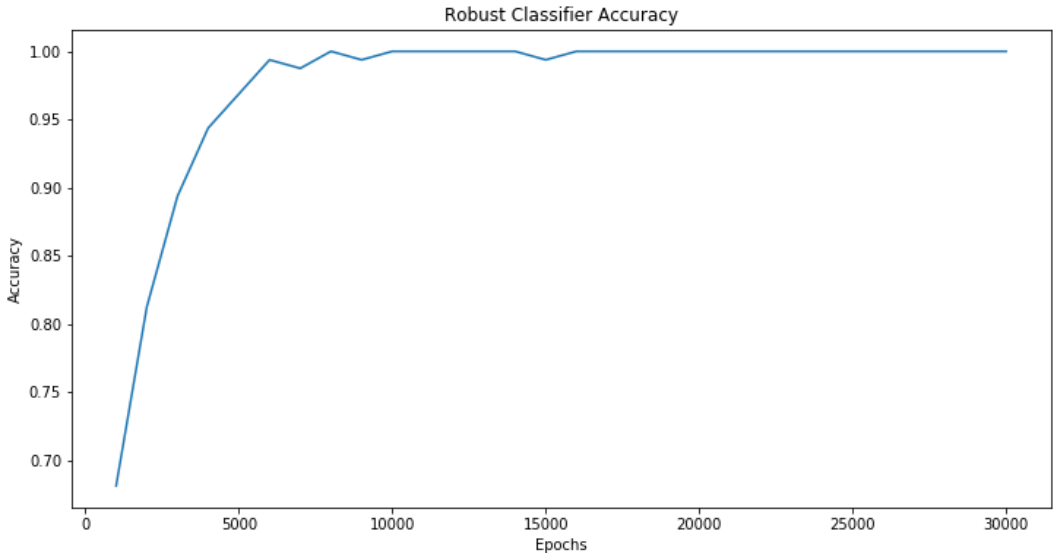
Testing Classic Classifier Against Generated Images



The classifier is tested against different images generated at the development stages of the Generator. The classifer is not capable of predicting the disfigured digits generated by the Generator in its initial stages of the training. However, due to highly accurate nature of Deep Neural Networks, the *Classic Classifier* predicts at a higher accuracy at later stages. At later stages the generated images similar to the original. The classifier takes some time to identify the generated

images and its accuracy gets better 5000th epoch, advocating the fact that these are susceptible to perturbation.

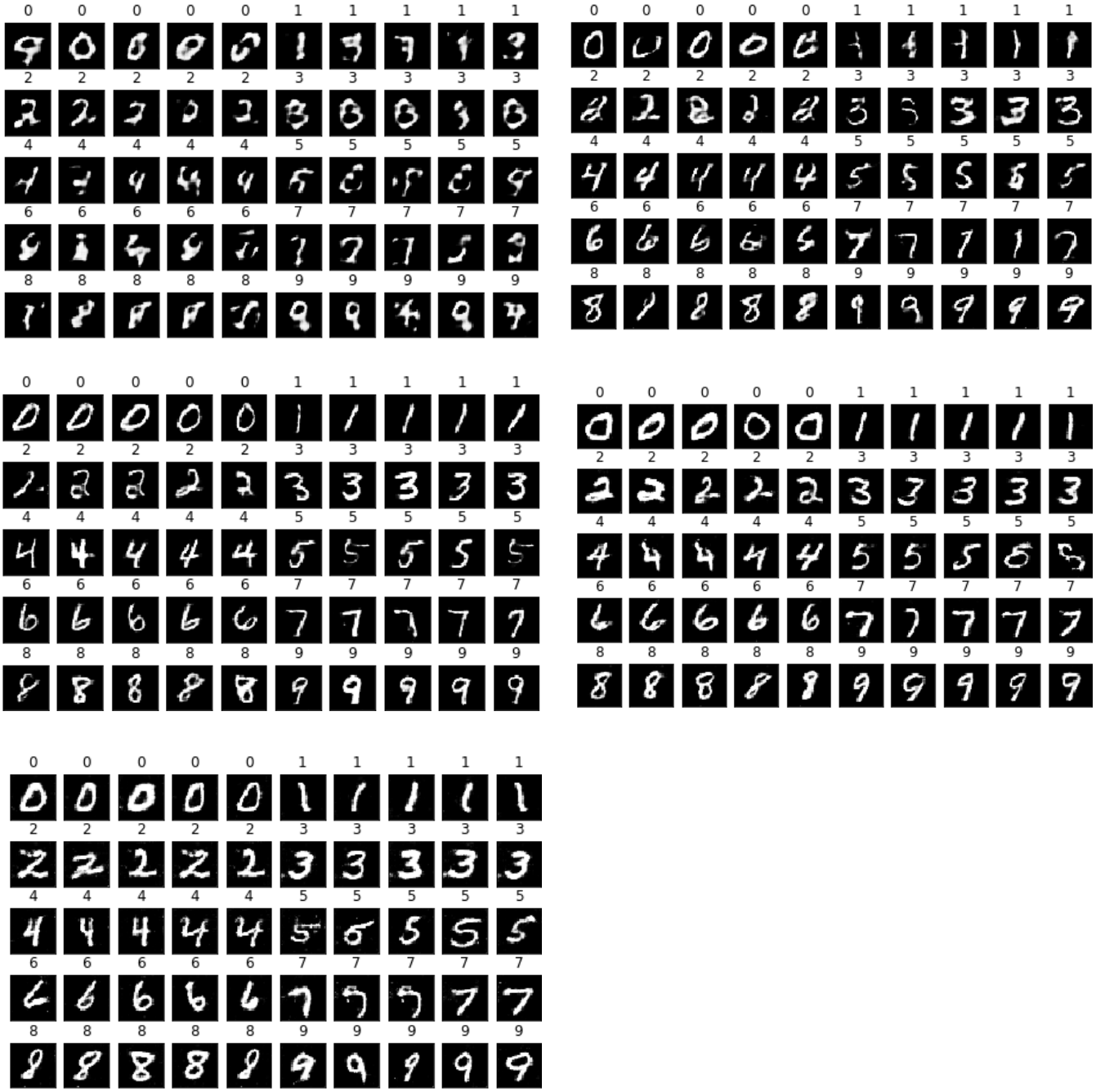
Testing Robust Classifier Against Generated Images



The *Robust Classifier* is capable of identifying digits with better than human accuracy and is showing remarkable performance even on images which are distorted and not properly formed. The image generated on the 1000th epoch is hardly decipherable and hence has a low accuracy of 68%, which sharply increases to 81% in the next 1000 epoch and then to 90% and nearing 100%.

V. Conclusion

Free-Form Visualization



The above images portray the different stages of the development of the *Generator*. The *Generator* outputs the images near to original during the end of its development. This enunciates that the GAN is trained without the earlier issues such as mode collapse and hard training. The highly accurate generation of images by the *Generator* implies that the same can be done for other types of dataset with multiple channels and larger images. However, there has been very little success.

Reflection

A deep neural network capable of high accuracy on the MNIST dataset is trained against adversarial settings. The trained model, *Robust Classifier*, is capable of identifying the digits with higher accuracy than the generic model, *Classic Classifier*.

Each of the models are trained in their respective settings, one with no adversarial training and another along with the Wasserstein GAN Network. The resultant model becomes robust to adversarial examples and is the *Robust Classifer*. The performance of this classifier is demonstrated by the figure provided in the Result section.

During the adversarial training process, interestingly the network also generates images near to original. The *Generator* quite accurately imitates the dataset.

Improvement

The major improvement in the project is to use a more complicated dataset with many features such as the CIFAR-10 or the Street View House Number (SVHN). However, due to the lower performance of GAN in generating images of multiple channels is discouraging.

Adversarial training methods are immune to adversarial examples, that is, if the model is tested against adversarial examples generated by other training methods then the model is still vulnerable. An improved adversarial training method is required so that the model is robust to universal adversarial examples.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[3] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. arXiv preprint arXiv:1507.00677, 2015.

[4] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 582–597. IEEE, 2016.

[5] Yann LeCun's website http://yann.lecun.com/exdb/mnist/

[6] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. arXiv preprint arXiv:1406.2661, 2014.

[7] Arjovsky, Martin, Chintala, Soumith, and Bottou, Léon. Wasserstein gan (WGAN). arXiv preprint arXiv:1701.07875, 2017.

[8] Ishaan Gulrajani , Faruk Ahmed, Martin Arjovsky , Vincent Dumoulin, Aaron Courville Improved Training of Wasserstein GANs (WGAN-GP). arXiv preprint arXiv:1406.2661, 2014.