Data Augmentation using AI Generative Methods

Petros Kafkas (mtn2204)

Contents

1 Introduction

It is a well known fact that one of the most common challenges when trying to solve a real-life classification problem using deep learning networks is the class balance problem. In such a case we have a collection of training samples belonging to two or more classes with the population of one class being vastly greater than the other. This imbalance acts as a bottleneck for the classifier's performance by limiting its discriminating power. Several techniques have been used in order to address this problem, with the most common of these being the artificial creation of knew samples for the imbalanced class by applying tranformations on the samples we allready have. Another approach is to use generative networks in order to generate entirely new samples for the training of the classifier. This latter approach is the one we utilize in this project. Therefore, it is the aim of this project to explore the effectivness of generative adversarial networks as data creation tools with the target of augmenting an inherently imbalanced dataset so as to increase the performance of a classifier.

2 Short discription of generative methods

2.1 Autoencoders

An autoencoder is a deep neural network which is comprised of two distinct parts: an encoder and a decoder (figure 1). The encoder recieves the user input and compresses it down to a lower dimension representation while the purpose of the decoder is to recieve the compressed representation of the data and then try to reconstruct the original piece of information. More specifically, the space of the compressed information is call latent, and the reduced representation is known as latent represenation in a similar fashion. If the latent space is two or three dimensional, then it permits the visualization of the previously higher dimentional data. In the case of a particular type of autoencoders known as variational, this latent space/representation

is approximated by a normal distribution. On the other side, the job of the decoder is to recieve this latent representation and produce the original data. More intrestingly, if we sample the latent representation distribution and produce new samples from it, we can then feed them to the encoder which in turn will produce "new" data. This new generated data is theoretically realistic-looking, since the basis for its generation, the latent variable distribution, is the same as the distribution learned by compressing the real world data. Lastly, in this project we make use of a special type of autoencoder called variational, which ensures that the distribution of the latent variables will resemble the shape of the normal distribution. This is achieved by including in the loss function the Kullback–Leibler divergence in order to minimize the distance between these two distributions (figure 2).

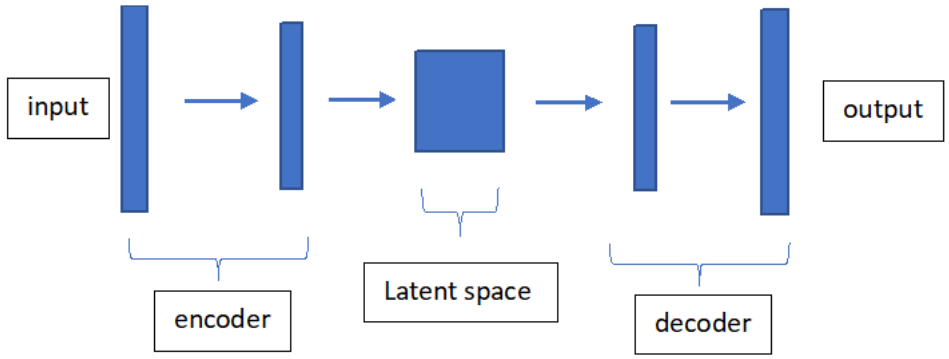


Figure 1: autoencoder

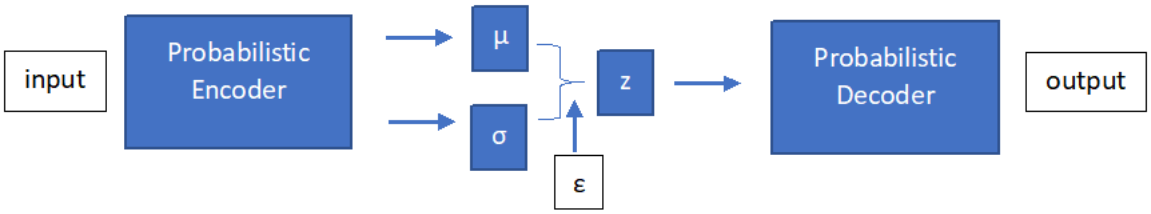


Figure 2: variational autoencoder

2.2 Generative Adversarial Networks

A generative adversarial network is created by using two distinct deep networks, one called the discriminator which has the function of identifying real data from fake data, and a generator network whose aim is to fool the discriminator by generating fake data that seem to be real in the eye of the discriminator (figure 3). Some of the most common problems of GANs are listed below:

- Mode collapse: This problem occurs when the generator "learns" to create a specific type of data that fools effectively the discriminator and from that point onwards it losses all motivation to create varied data, since it can be sure to fool the discriminator by repeatedly creating the same output. In other words, it fails to capture the true distribution of the input data.
- The generator oftentimes is proven to be more powerfull than the discriminator in which case it overpowers the ability of the critic to correct in a meaningfull way, with the result of the generator creating low quality data.

A solution to these problems is to use an alternative type of gan called wgan. The wgan makes use of a differnt kind of loss function named Wasserstein Loss. The Wasserstein loss function is essentialy using the Lipschitz norm in order to bound the gradients of the discriminator's learning function so as to remain between 0 and 1, ensuring that the gradients can at all times be properly defined and calculated. Finaly, we have implemented two other variations of the basic GAN architecture: the conditional wgan, whereby the generator is conditioned on the labels of the data, so that its output can be chosen based on the label we want. Finaly, there is also the progressive learning gan, whereby we start by generating low resolution data and progressively we increase the resolution output of the network's layers (4). All these networks implementations are uploaded on out project's github repo.

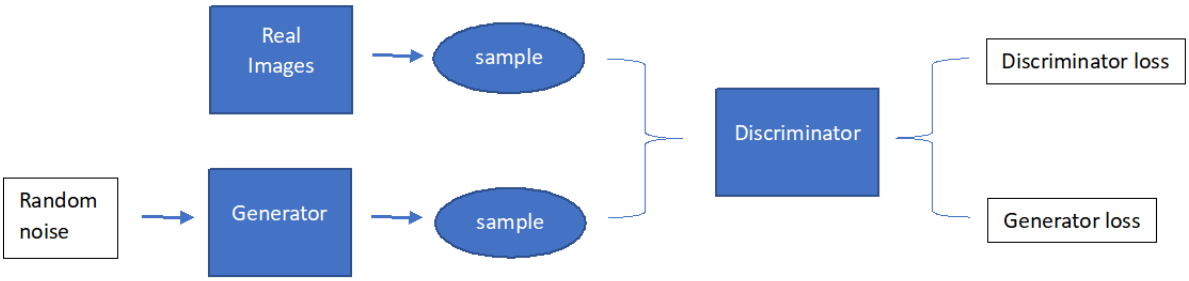


Figure 3: generative adversarial network

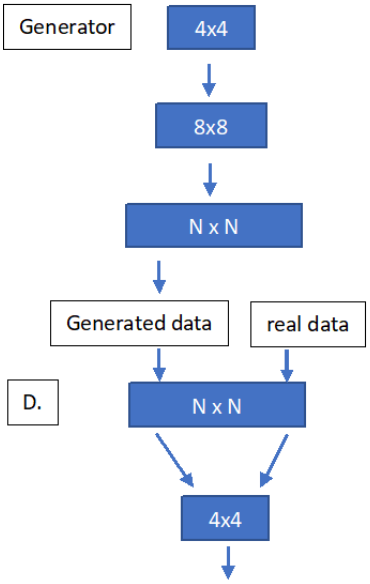


Figure 4: prograssive learning GAN

3 Experiment Discription

3.1 Dataset Discription and Visualization

For this project we have used two well known datasets: the mnist number dataset as well as the mnist fashion dataset. Because these datasets are equally balanced along their respective data

classes, we have created an "artificial" imbalance by deleting most of the samples of a particular class. For the mnist number dataset we have chosen to eliminate most of the number five (5) samples. In particular, we deleted 99 per cent of the number fives of the dataset, which resulted in a total of 54 samples remaining of that class. We have utilised a variational auto encoder in order to visualize the dataset. The results can be seen in figure 5 and figure 6.

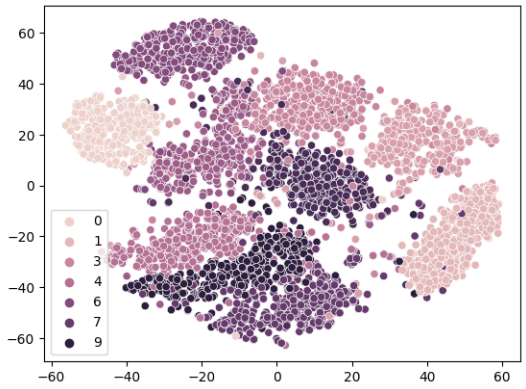


Figure 5: MNIST train data distribution

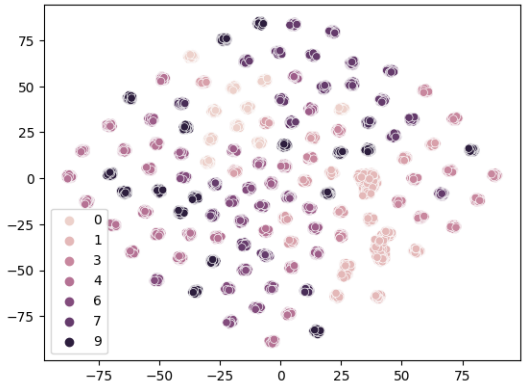


Figure 6: MNIST test data distribution

3.2 Initial Classification Results

We used a deep convolutional classifier in order to get a baseline for the accuracy that can be achieved on the imbalanced dataset without any augmentation performed. The results can be seen in the form of the confusion matrix (figure 7). We observe that the classifier's accuracy of the class corresponding to number five is very low compared to all other classes. In more detail, it seems that the classifier finds it especially difficult to discriminate between numbers three, five, eight and nine. This is to be expected since it doesn't have enough samples of that class in order to properly learn to identify it. This can also be observed in the total accuracy and loss graphs (figures 8 and 9, respectively). We note that during training we observe that as time progresses, the classifier "overfits" the training data and the discrepancy between that train data distribution and the test data distribution (the former is highly imbalanced) begin to affect the loss as well as the accuracy of the classifier.

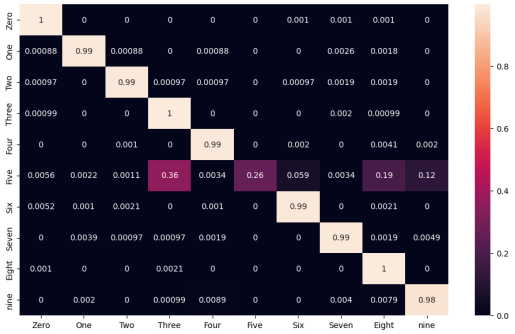


Figure 7: Mnist confusion matrix (imbalanced)

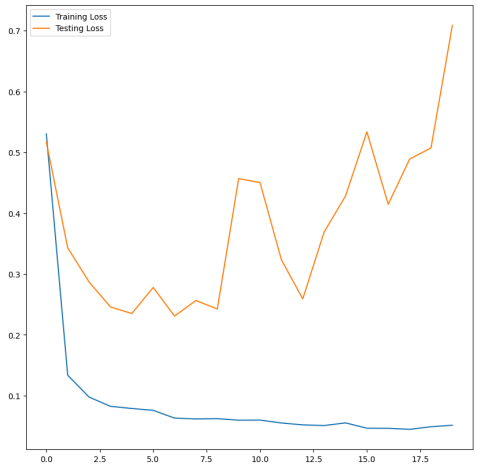


Figure 8: train/test loss (imbalanced)

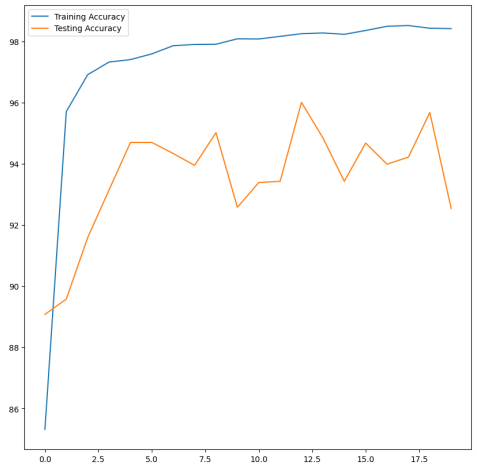
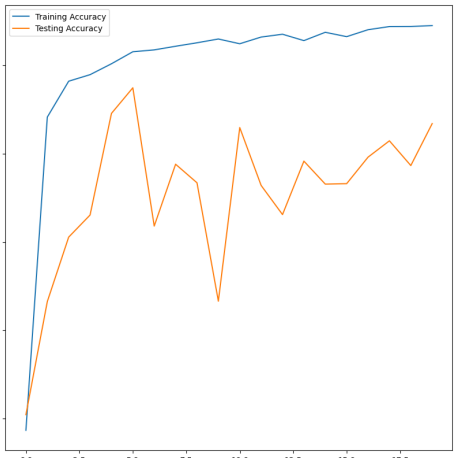


Figure 9: train/test accuracy (imbalanced)

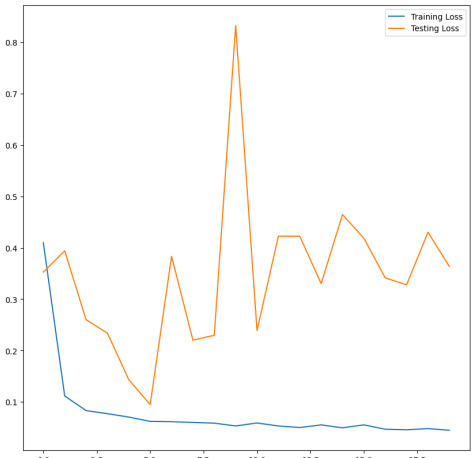
3.3 Data augmentation using w(pro)GAN

As we mentioned in the introduction during this experiment we utilised a wGAN network in order to create new "fake" data for the imbalanced class. We experimented using different

ammounts of generated data, namely 100 new samples of fives, 1000, 2000 as well as 4500 samples. The results on the accuracy can be seen in the following figures (10, 11, 12, 13). The respective confusion matrix metrics are depicted on figures 14 and 15. We observe an increase in accuracy and a decrease in the loss measured both in training and in validation phases of the experiment.

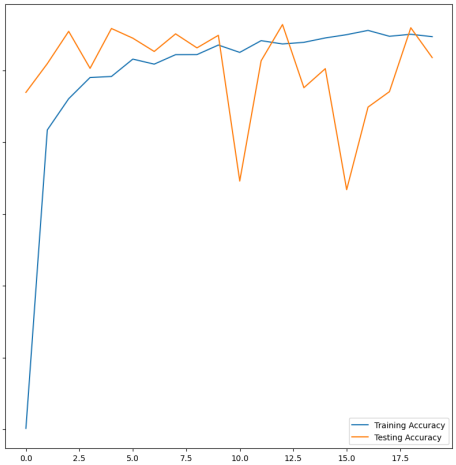


(a) train/test accuracy (100 new samples)

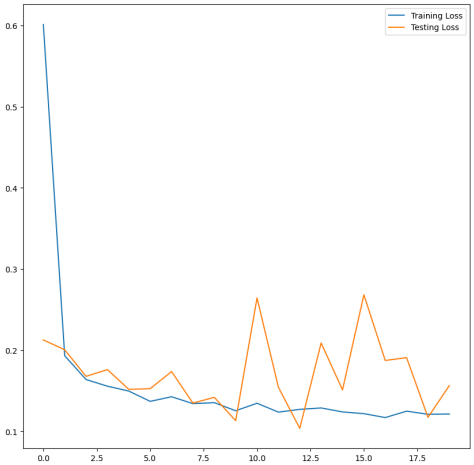


(b) Train/test loss (100 new samples)

Figure 10: accuracy and loss (100 new samples)

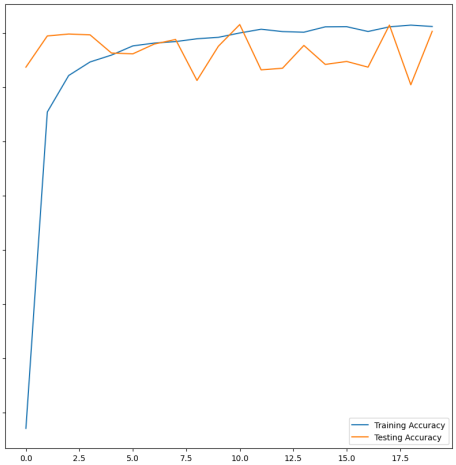


(a) train/test accuracy (1000 new samples)

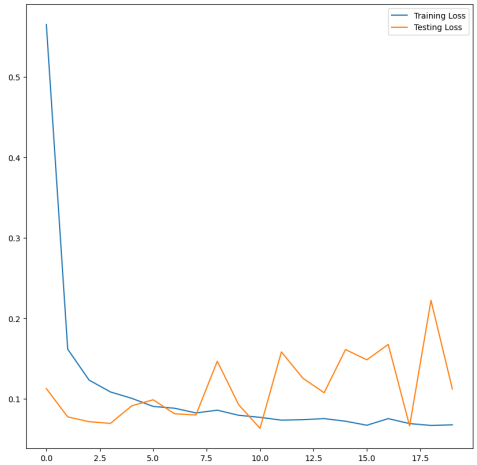


(b) Train/test loss (1000 new samples)

Figure 11: accuracy and loss (1000 new samples)

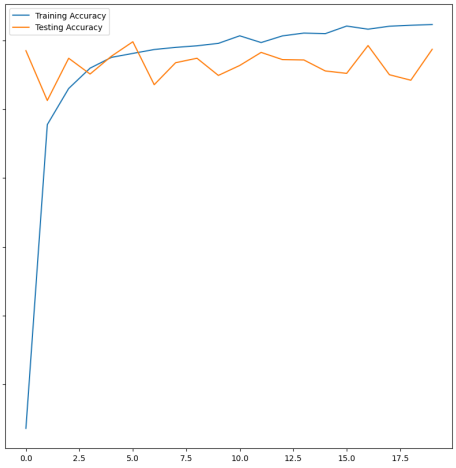


(a) train/test accuracy (2000 new samples)

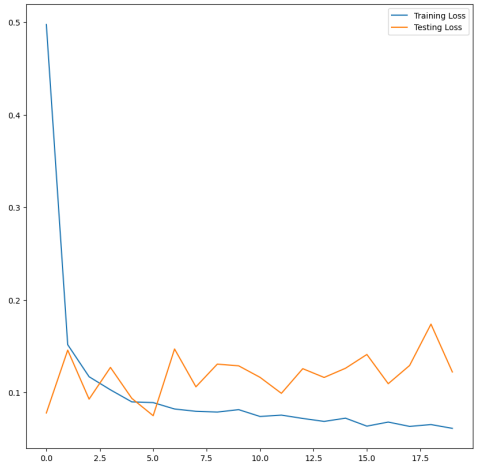


(b) Train/test loss (2000 new samples)

Figure 12: accuracy and loss (2000 new samples)

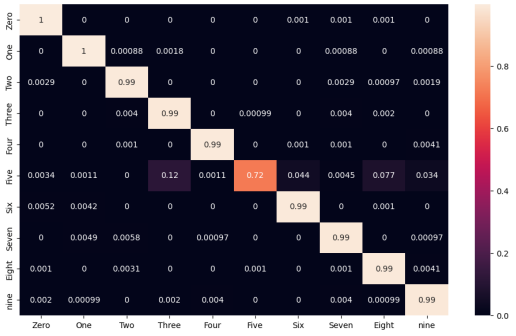


(a) train/test accuracy (4500 new samples)

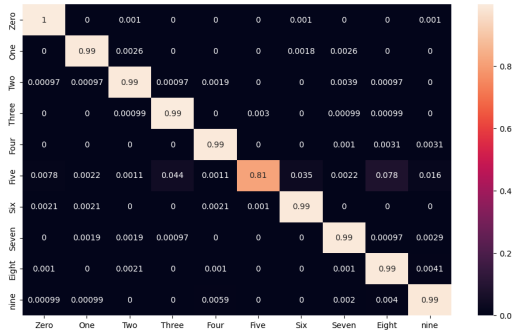


(b) Train/test loss (4500 new samples)

Figure 13: accuracy and loss (4500 new samples)

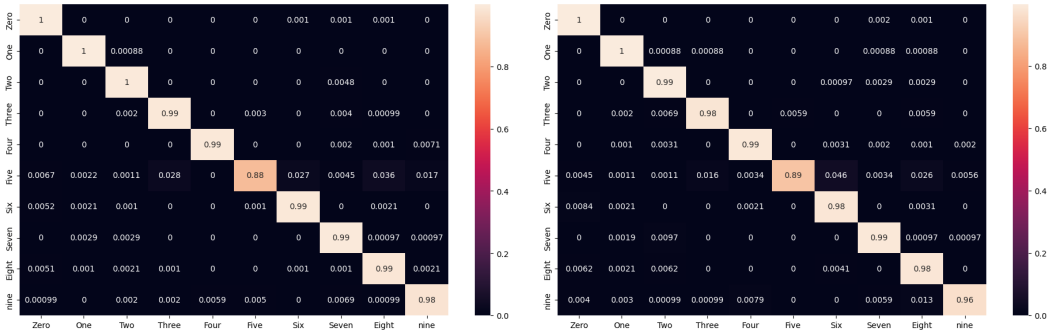


(a) confusion matrix (100 new samples)



(b) confusion matrix (1000 new samples)

Figure 14: confusion matrix (100 and 1000 new samples)

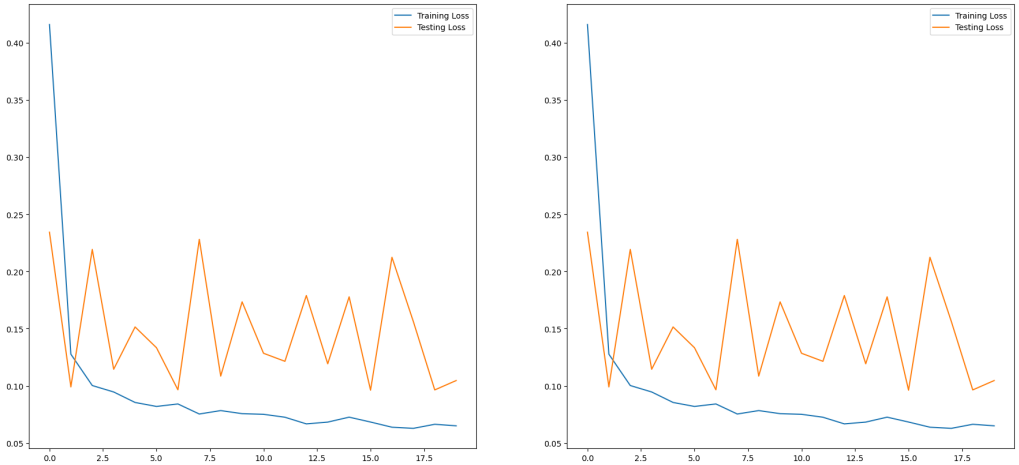


(a) confusion matrix (2000 new samples) (b) confusion matrix (4500 new samples)

Figure 15: confusion matrix (2000 and 4500 new samples)

3.4 Data augmentation using crop/rotate

For this part of the project we experimented with a more traditional way of augmented data, that is we applied some tranformations (crop and rotation += 10 deg) on the samples of the imbalanced class in order to create more samples. The results for the train/test loss and accuracy, as well as the confusion matrix are given in figures 16. Finaly, the confusion matrix is shown in figure 17.



(a) train/test loss (b) train/test accuracy

Figure 16: loss and accuracy (crop/rotation data)

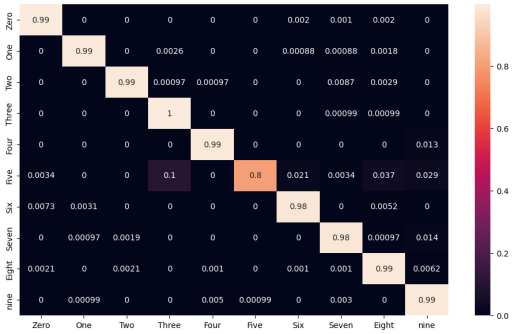


Figure 17: confusion matrix

4 Analysis and Results

First, we observed an increase on the accuracy and a decrease on the loss values during both training and testing when we used the augmented data generated from either method. In general, the accuracy reached by the fake data generated from the GAN is higher than the accuracy we observed when we augmented the data using traditional transformations. More specifically, we observed a correlation between the number of the generated data and the improvement of the accuracy of the classifier: namely, the smaller the gap between the number of samples on the imbalanced class and the rest gets, the better the perfomance of the classifier gets. This is to be expeted, since when there is virtually no imbalance (number of samples 4500) the classifier reaches its highest accuracy. Also, we were able to significatly improve the classifiers performance by just augmenting our data using random transfomations on our dataset. However, our data was relatively simple, being just numbers in grayscale, and we expect that the more complex the data gets, the less efficient this method of augmenting will get. Lastly, we observe that the classifier trained on the augmented data exhibit a somewhat erratic loss graph (albeit getting a little smoother when we use more samples). This is due to the very limited initial data we had available to train our GAN which resulted in the GAN not learning the full distribution of data that is present on the test case. We believe that the closer the training data's distribution matches the testing data's distribution, the higher will be the quality of the generated data and, subsequently, the classifier's perfomance will also increase.