HERITAGE INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

AND

ENGINEERING

B.Tech FINAL YEAR PROJECT

to obtain the title of

Bachelor In Technology

SPECIALISATION: COMPUTER SCIENCE

Guided By:

Prof. DINABANDHU BHANDARI

APPLICATIONS OF GENERATIVE ADVERSARIAL NETWORKS(GANs)

Written By:

DEBANGSHU BHATTACHARYA(12615001056)

SOUMYADIP ROY(12615003155)

SRIJON SARKAR(12615001159)

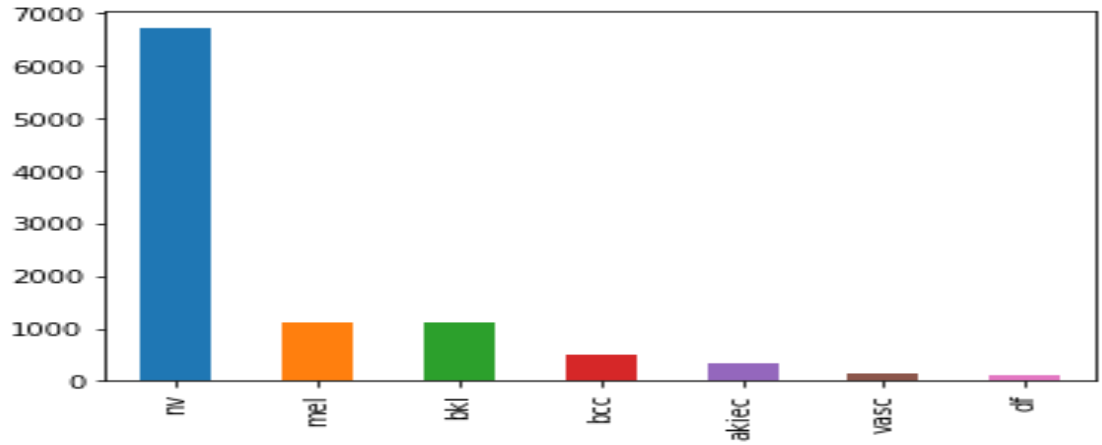
VARUN KUMAR MAHANOT(12615001183)

ABSTRACT

In this project, we are going to show some applications of Generative Adversarial Networks(GANs) in the domain of Computer Vision and Deep Learning. We are going to demonstrate how GANs are used in data augmentation techniques to improve the performance of machine learning models, especially where the dataset is so imbalanced that a classical image classification model like Convolutional Neural Networks(CNNs) tends to overfit to the dominant class. We are going to perform two different GAN-based augmentations (online and offline) and compare the results of these experiments. We found that the CNN model trained with the GAN based data augmentations (both of them) perform better than the only CNN model.

Dataset and Preprocessing

We used a part of the HAM10000 dataset, a Multi-Source Dermatoscopic images of common Pigmented Skin Lessions, for our experiment. The original dataset comprises of 10015 images of 7 classes. The data distribution of the original dataset is given below.

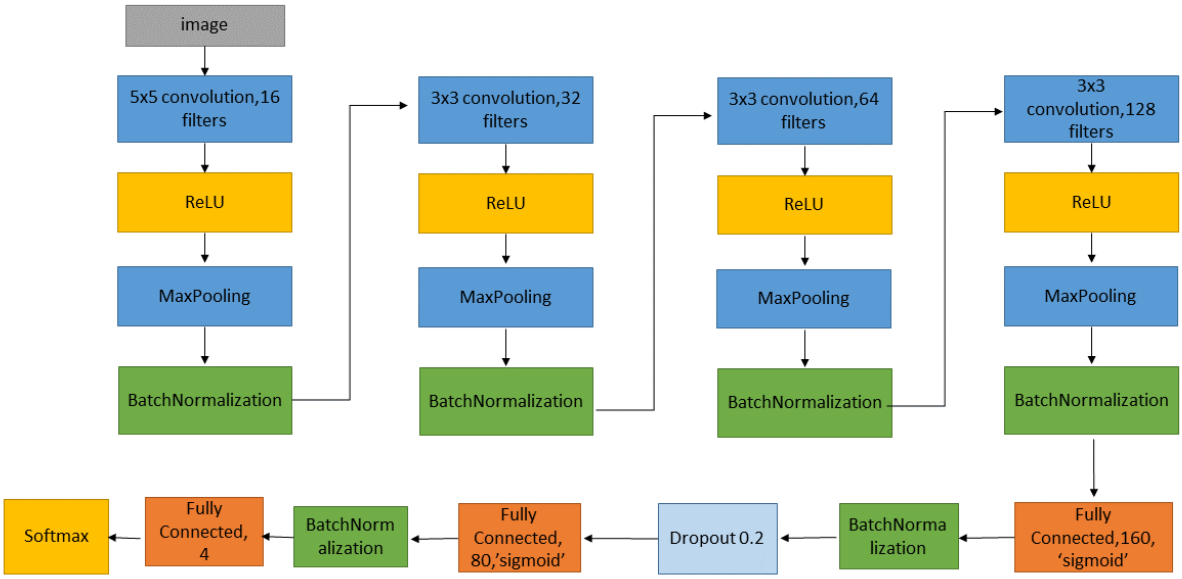


Since some of the classes have extremely low number of data, we are using only the top 4 classes of “nv”, “mel”, “bkl”, and “bcc” with 6705, 1113, 1099 and 514 number of examples each. We can see that this dataset contains a huge imbalance with the class “nv” having more examples than all the other classes combined.

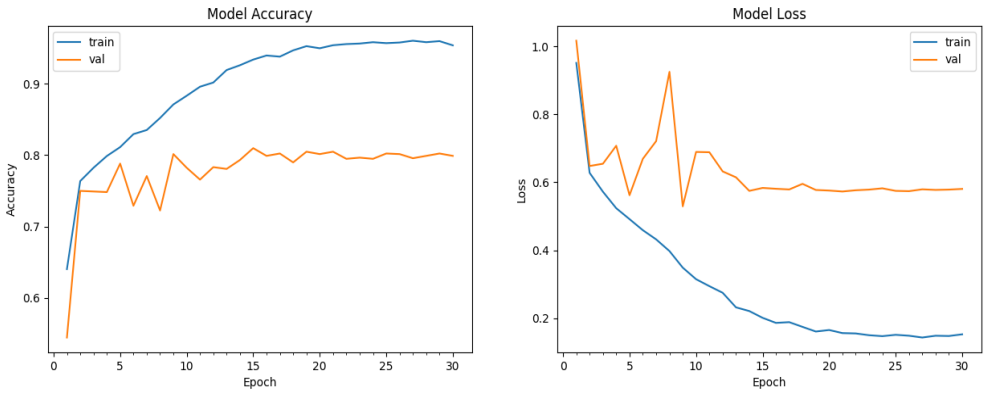
We perform a two level stratified split on this data. In the first split we take 85% of the data for training and validation sets and the remaining 15% is taken for test set. In the second split, we take 85% of the data from training and validation set for our train set and the remaining 15% for our validation set. We re-sized the images to (64,64,3), normalized them in the range of 0 to 1 and performed a one hot encoding of our labels as our pre-processing step.

Experiments and Results

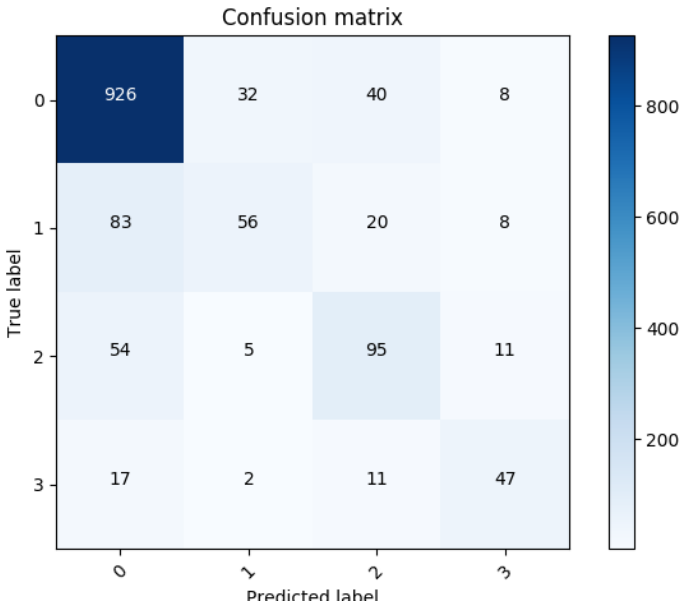
We used the following CNN model for our experiment.



We trained the dataset on this model and the results are given below:



Model History of **Classical CNN**

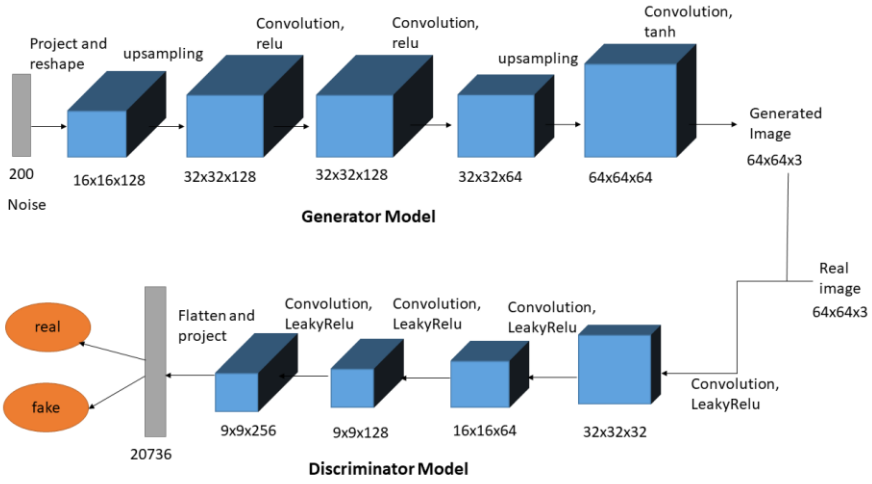


Confusion Matrix

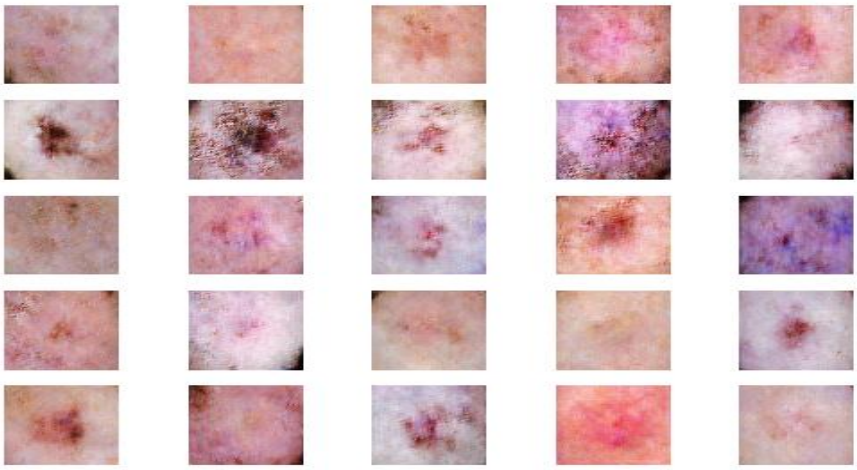
```
Test Set accuracy = 79.434629%
0.7943462897526502
             precision    recall  f1-score   support

          0       0.86      0.92      0.89      1006
          1       0.59      0.34      0.43       167
          2       0.57      0.58      0.57       165
          3       0.64      0.61      0.62        77

avg / total       0.78      0.79      0.78      1415
```

Normalized Confusion Matrix

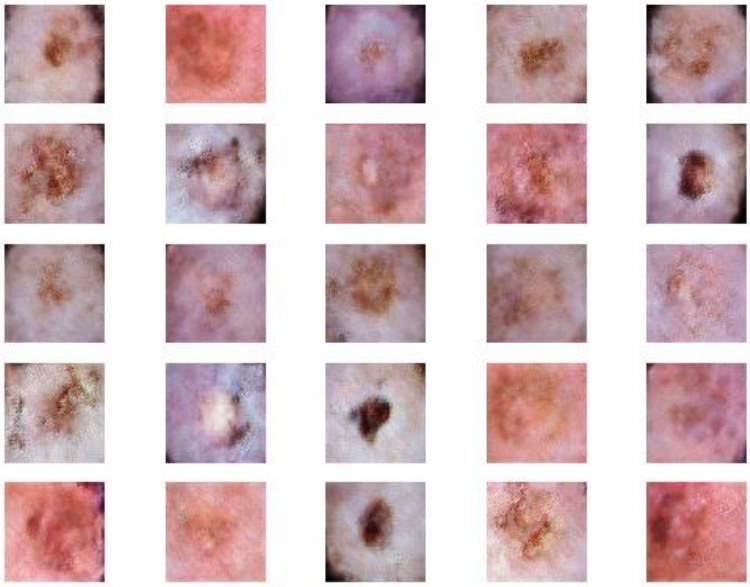
We can clearly see that the model is overfitting to the dominant class “nv”. To combat this, we use data augmentation. We train a Deep Convolutional Generative Adversarial Network (DCGAN) network to train on our data. We use the following DCGAN architecture:



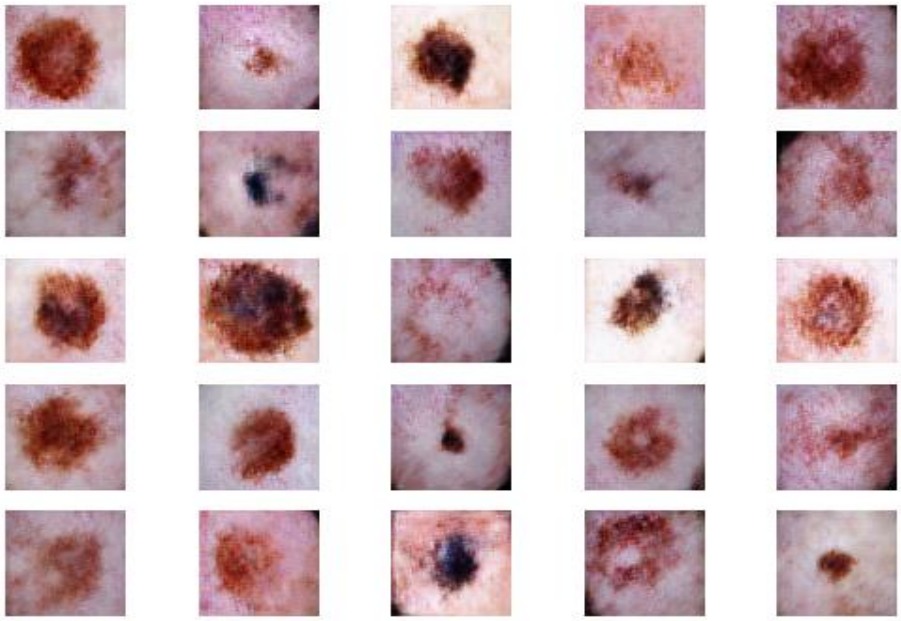
After training the DCGAN architecture these are the samples of synthetic images generated for each of the classes:



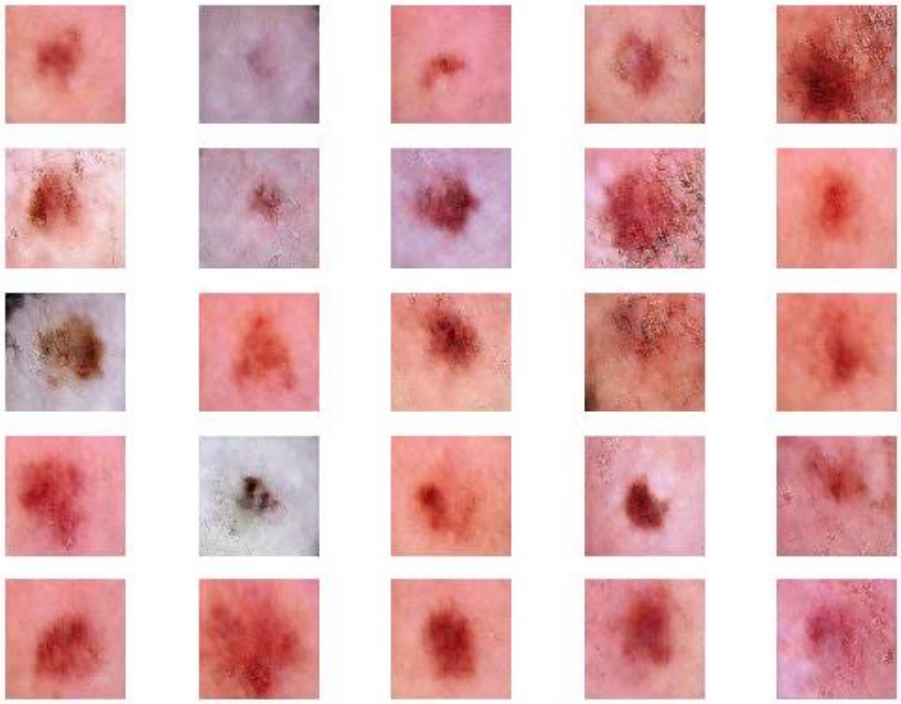
Generated images for class **bcc**



Generated images for class **bkl**



Generated images for class **mel**



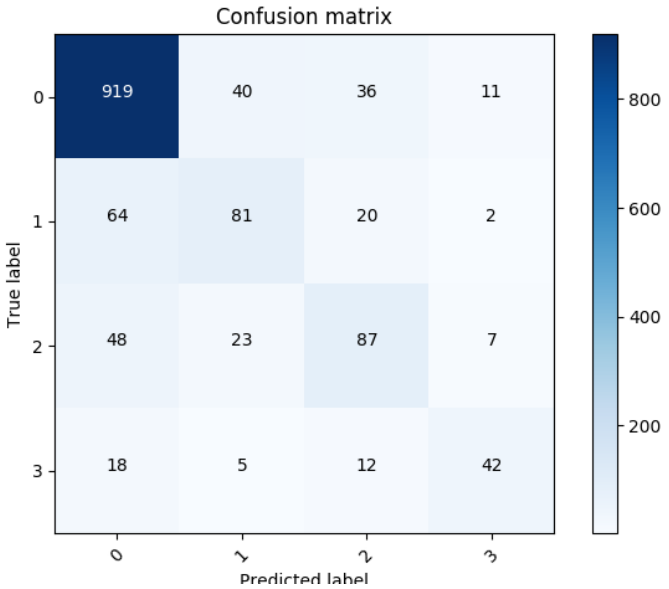
Generated images for class **nv**

We use two kinds of augmentation techniques using these GAN based Generated images to enhance the performance of our CNN model.

1. **Online Augmentation**

   In this augmentation technique we keep a part of our original dataset and replace the rest with our synthetic generated images. This replacement policy is based on a keep_probability hyper parameter which we tuned to the value of 0.7, i.e, we kept 70% of our original data and replaced the rest 30% with generated data. The reason behind this technique is to introduce some noise to our data so that the model does not overfit to the dominant class and can generalize better.

   The results of this experiment is show below:



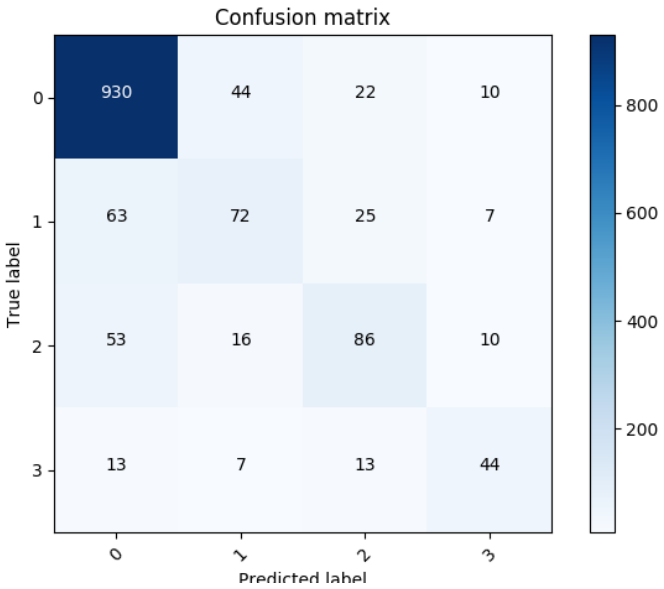
Confusion Matrix

```
Test Set accuracy = 79.787986%
0.7978798586572439
              precision    recall  f1-score   support

           0       0.88      0.91      0.89      1006
           1       0.54      0.49      0.51       167
           2       0.56      0.53      0.54       165
           3       0.68      0.55      0.60        77

avg / total       0.79      0.80      0.79      1415
```

Normalized Confusion Matrix

2. **Offline Augmentation**

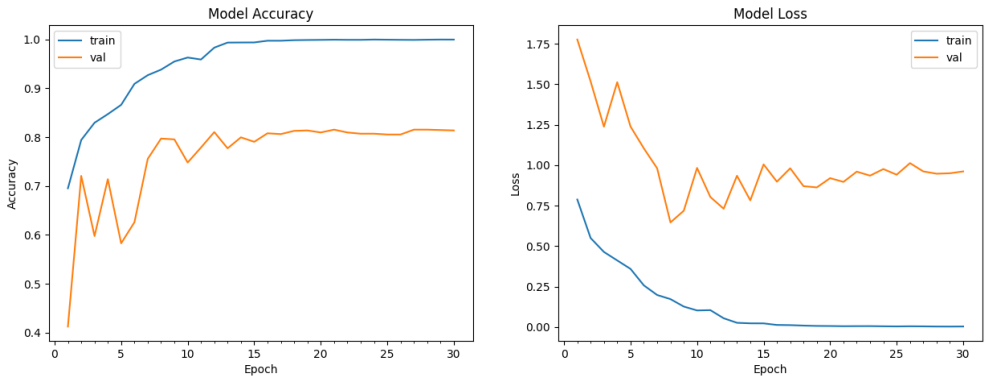
In this technique we add to our existent dataset with our generated synthetic images. The idea is to add more examples for the classes with less number of examples such that the model can generalize better. For our experiment we add 500, 1000, 1000 and 1500 generated images for the classes “nv”, ”bkl”, ”mel” and “bcc” respectively. The results are shown below:



Confusion Matrix

```
Test Set accuracy = 80.000000%
0.8
             precision    recall  f1-score   support

          0       0.88      0.92      0.90      1006
          1       0.52      0.43      0.47       167
          2       0.59      0.52      0.55       165
          3       0.62      0.57      0.59        77

avg / total       0.79      0.80      0.79      1415
```

Normalized Confusion Matrix

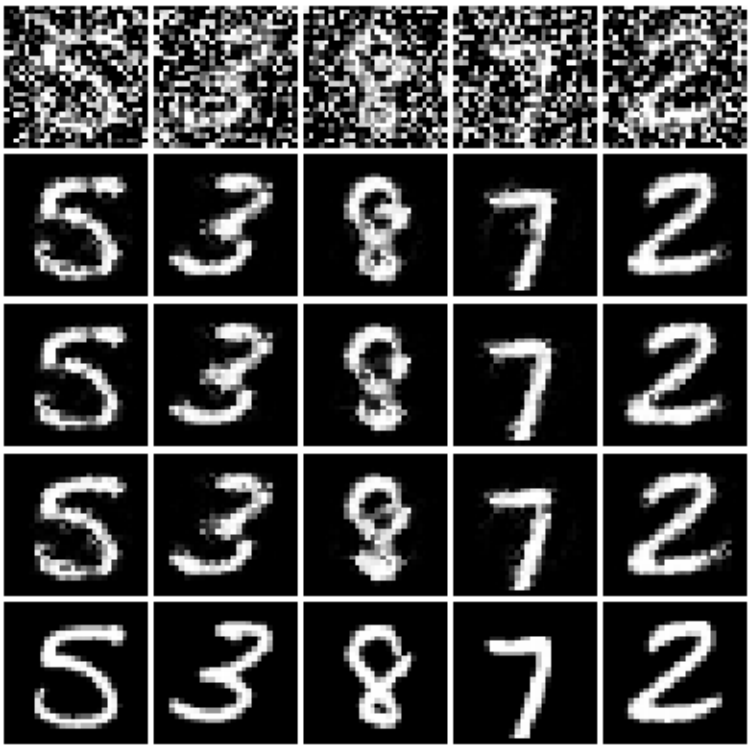


Model history for **offline augmentation**

Finding Out Numbers From A Noisy Sample Of MNIST Dataset

In this experiment we demonstrate the application of Generative Adversarial Imputation Nets (GAIN), that generalizes the well-known GAN (Goodfellow et al., 2014) and is able to operate successfully even when complete data is unavailable. In GAIN, the generator's goal is to accurately impute data, and the discriminator's goal is to distinguish between observed and imputed components. The discriminator is trained to minimize the classification loss (when classifying which components were observed and which have been imputed), and the generator is trained to maximize the discriminator's misclassification rate. Thus, these two networks are trained using an adversarial process. To achieve this goal, GAIN builds on and adapts the standard GAN architecture. To ensure that the result of this adversarial process is the desired target, the GAIN architecture provides the discriminator with additional information in the form of "hints". This hinting ensures that the generator generates samples according to the true underlying data distribution.

Results:



Here we have tested with the MNIST dataset. We at first trained the model with the dataset and then we blurred the samples with a white noise randomly. Our system then takes the blurred image as input and through a number of epoch achieves the required pictures.

Conclusion

We saw that both of our GAN based augmentation techniques performed better than the CNN classification with an improvement of 1% in precision and 1% in recall respectively.
In our second experiment we have seen that using blurred images or noisy images as input we have found the accurate figures as output.