Internship at IIIT

<u>From:</u> August 2020 – October 2020

<u>Name:</u> Samriddh Gupta

<u>University:</u> American University

<u>Topic:</u> Classification of images using CNN and Generation of Images using GANs and Autoencoders

Convolutional Neural Network

Convolutional neural networks (CNNs) are a group of neural networks that belong to a wider family of methods known as deep learning. The secret for their success lies in their carefully designed architecture capable of considering the local and global characteristics of the input data. Initially, CNNs have been designed to process image data efficiently, and for this, they were developed with properties such as local connectivity, spatial invariance, and hierarchical features. With these properties, CNNs have propelled breakthroughs across several research areas and have recently been applied in psychiatry and neurology to investigate brain disorders. In this chapter, we will present a theoretical introduction to CNNs. We will then illustrate their use in brain disorders by reviewing exemplary applications to neuroimaging and electroencephalogram data. The most popular use of CNN is that it could learn anything after you have provided enough data. The most common method which they are implied is that of object detection, image scanner, face detection, etc. While interning at the IIIT I did multiple projects on them and I will be explaining them in this paper.

CNN with MNIST Dataset

The study that we perform is that of detecting the number presented is the actual number on its label. I used two hidden layers. We took a batch size of 150 and an epoch of 50. After compiling the data, I have a value of loss function to be 0.4 and have an accuracy of 98.62%.

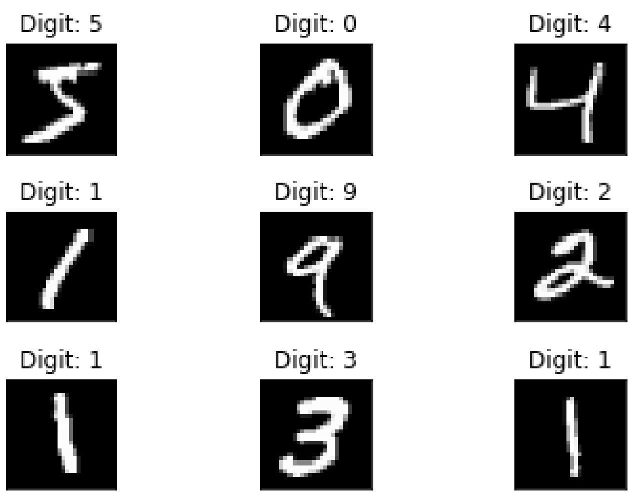
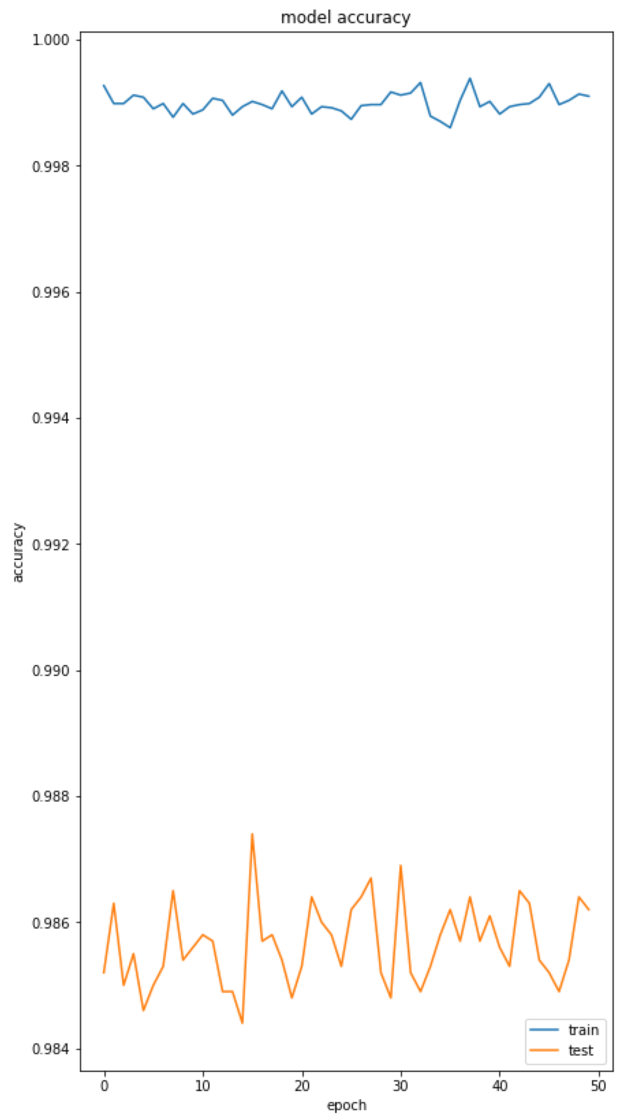

Digit: 5
Digit: 0
Digit: 4
Digit: 1
Digit: 9
Digit: 2
Digit: 1
Digit: 3
Digit: 1

Fig: MNIST dataset



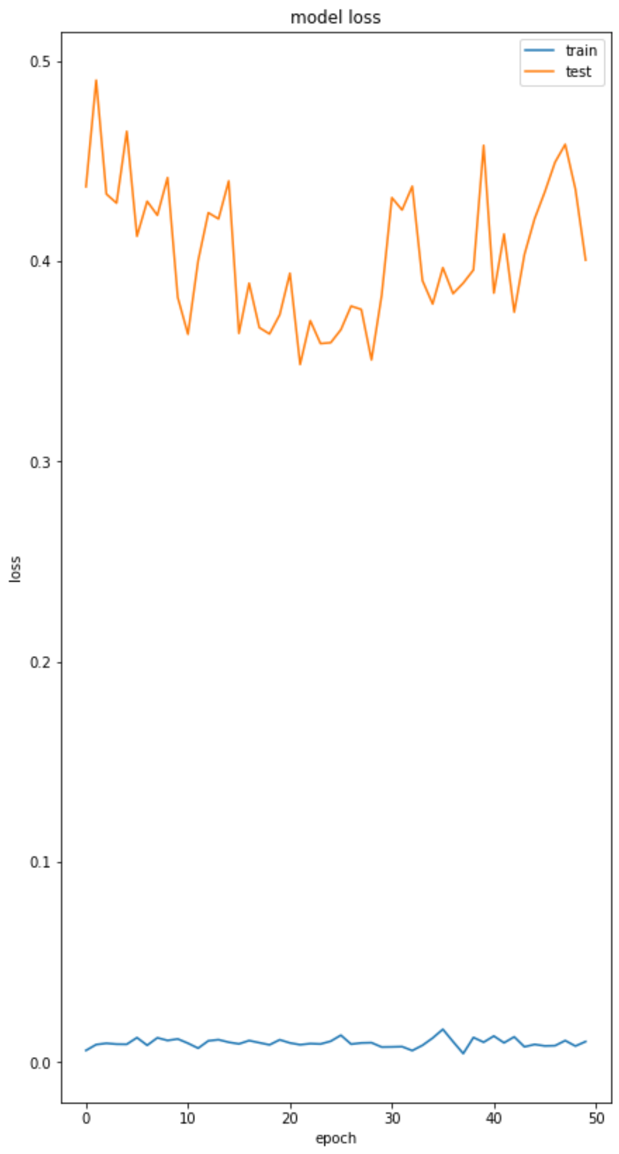
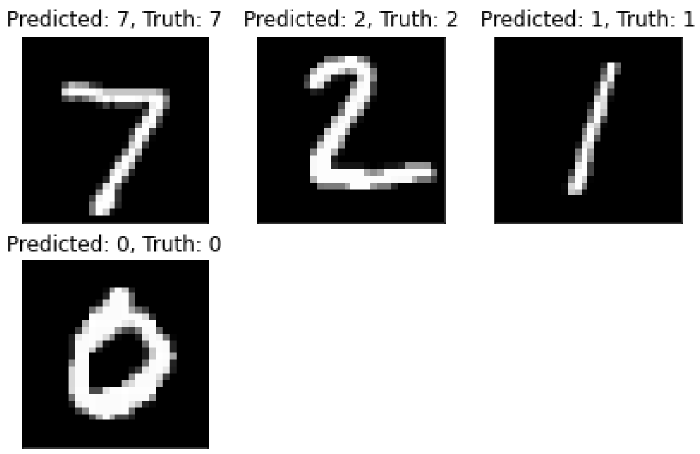


Fig: Accuracy Graph and Model Loss



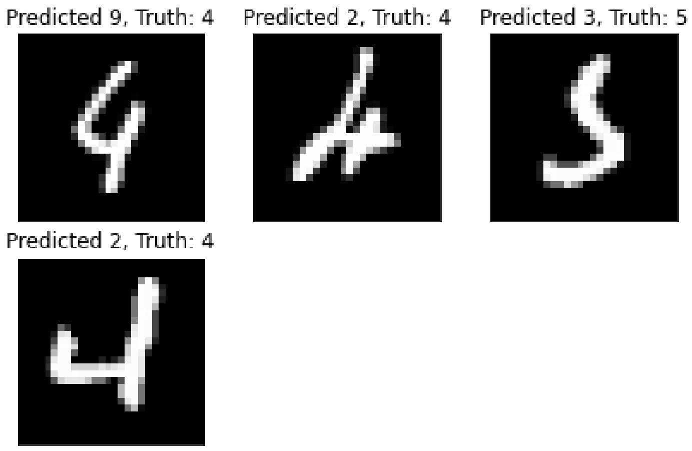
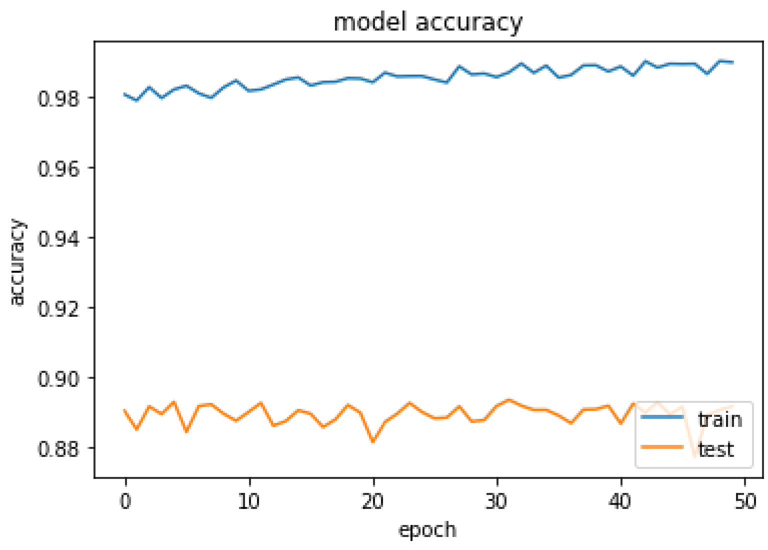


Fig: CNN Model Example

CNN with MNIST Fashion Dataset

MNIST fashion data set have clothes label on them and we have a total of 10 unique labels which are T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. For this CNN we also have two hidden layers. For this model, while compiling we have a batch size of 100 and epochs of 50. But for this, we have a high loss function value of 0.96, and accuracy was only 89.17%.

Fig: MNIST Fashion Dataset



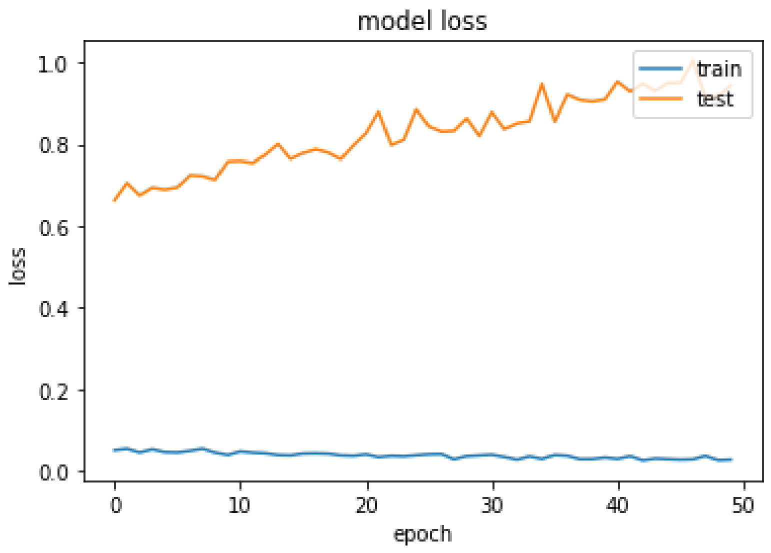


Fig: Accuracy Graph and Model Loss

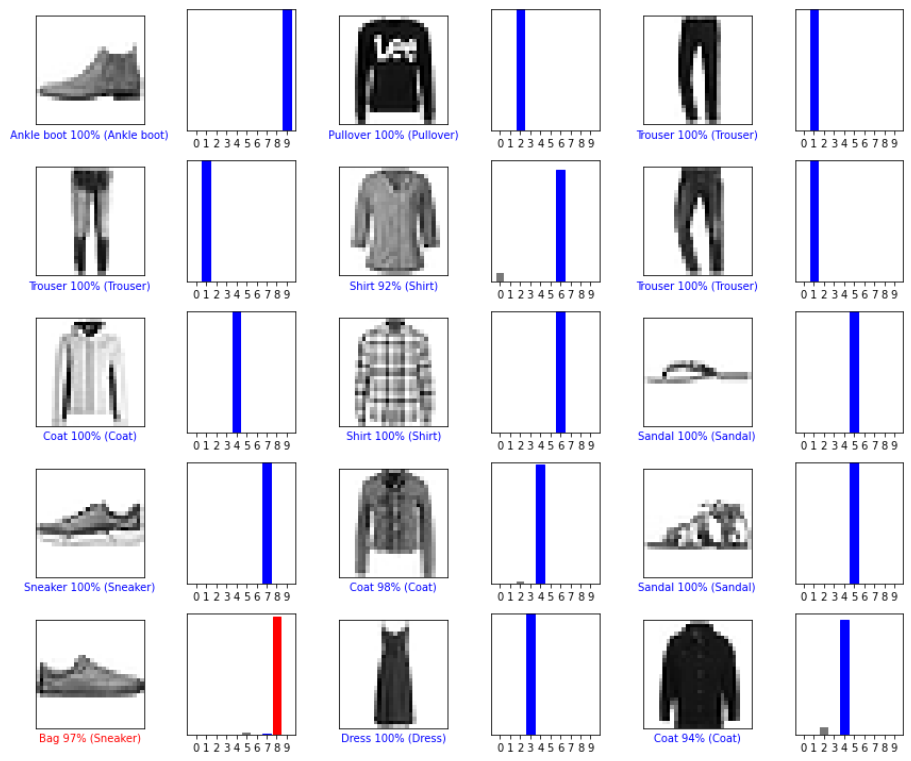
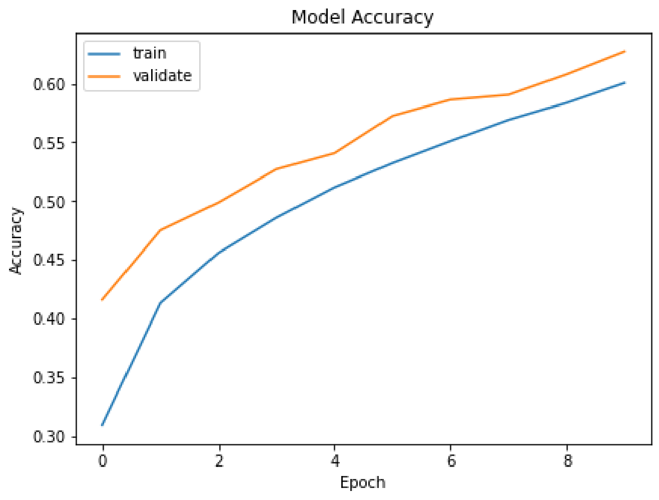


Fig: CNN Model Example

CNN with CIFAR-10

CIFAR-10 is a dataset that has 10 different labels of data such as a tree, airplane, etc. For this CNN we have 3 hidden layers. For this model, while compiling we have



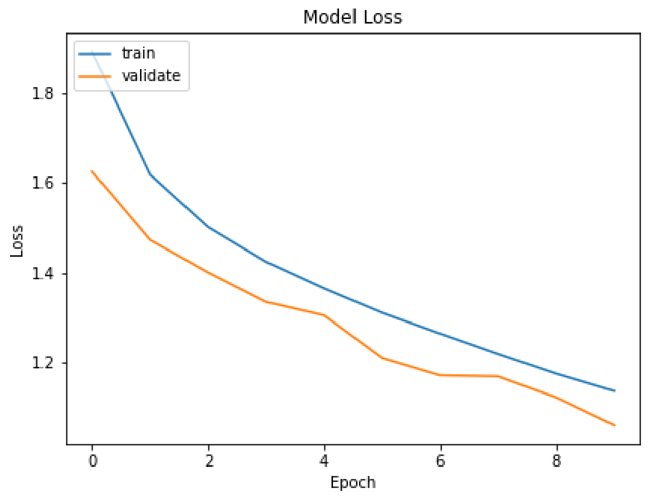


Fig: Accuracy Graph and Model Loss

a batch size of 100 and epochs of 10. We have a high loss function of 1.06 and accuracy was of 62.73%

CNN with Dogs VS Cats Dataset

Dogs VS Cats Dataset were images of both dog and cat. We 12500 images of both dogs and cats. The dataset was taken from Kaggle. We have 3 hidden layers and while compiling the model we have we have a batch size of 15 and epochs of 50. But for this, we have a high loss function value of 1.12, and accuracy was only 63%.

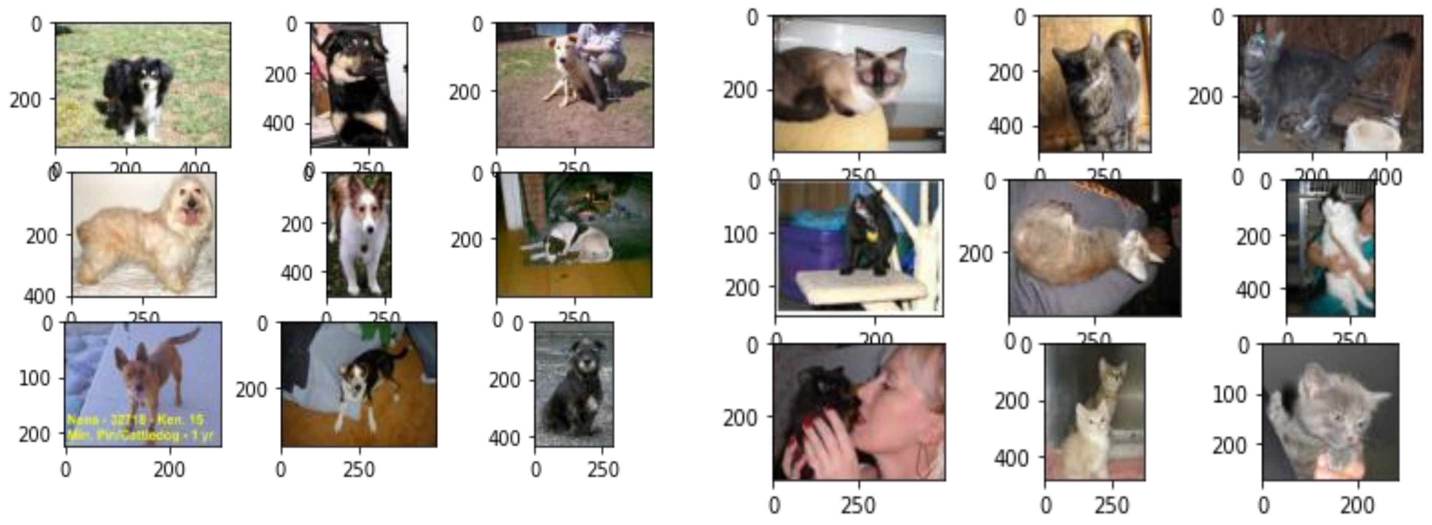
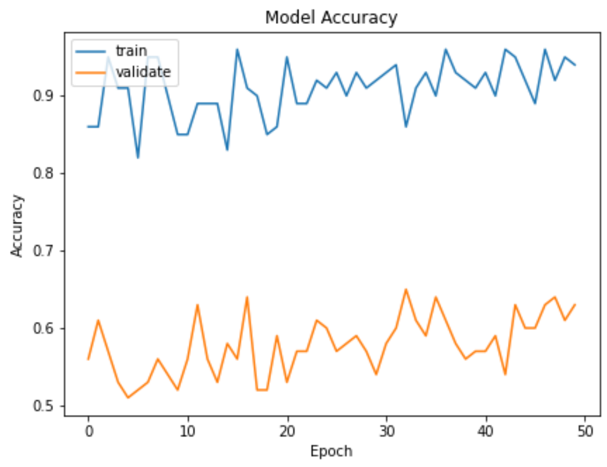


Fig: Cat Vs Dog Data set



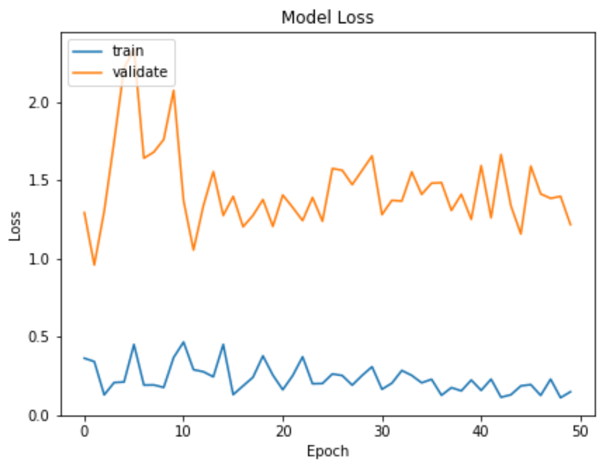


Fig: Accuracy Graph and Model Loss

Gan Descriptions

Introductions

Gan or generative adversarial network is the latest machine learning technique that could be used to generate new data. These are set of two neural nets: one is a Generator and the other one is Discriminator. These two neural nets which help others to learn and generate new data based on it.

Generative Method

This neural net tries to generate new data from the training data set. Its main objective is to make a discriminator neural net believe that the image which was generated by it is real and the other is fake. If the discriminator is assuming its image to be real, then we have new data generated and if it fails, the generator uses backpropagation to make a new image. The Generator has one input that gets a random sample from the population and it generates images or data based on it.

Discriminative Method

This neural net tries to generate the result of whether the image that it has received from the generator is real or not. The objective of this neural net is to see whether the image which it has gotten from the generator is real or not. If it determines that the image is fake, then it is a win for Discriminator otherwise its losses. The discriminator takes two values as input which are the training dataset and the other one is the image generated by the generator. It gives two outputs as well which is 0 or 1(being the result was not a particular image and 1 being that that was the generated image) as well as the loss function which can be used to backpropagate the Generative method to give a better image.

Applications for the Gans

Gans can be used anywhere the image needs to be created. It could be used from creating new images to changing the background of the old image. Since it has multiple uses, it is lightly in demand in the industry.

Face Generation using Gan

In Face generation project, we were tasked to make facial images with the help of Gans using the celebrity images which is available on Kaggle website. The discriminator in our model have four hidden layer and was using the sigmoid function to get the output. The generator has four hidden layers as well and was using the tangent function to get the output. The dataset has around 10000 images of different people and the images were structured from them. In the making of images we train the model 9 times and we have a batch size of 32. We must limit the number of epochs because one epoch was taking approximately two hours to run in google colab. We got the following as our result.

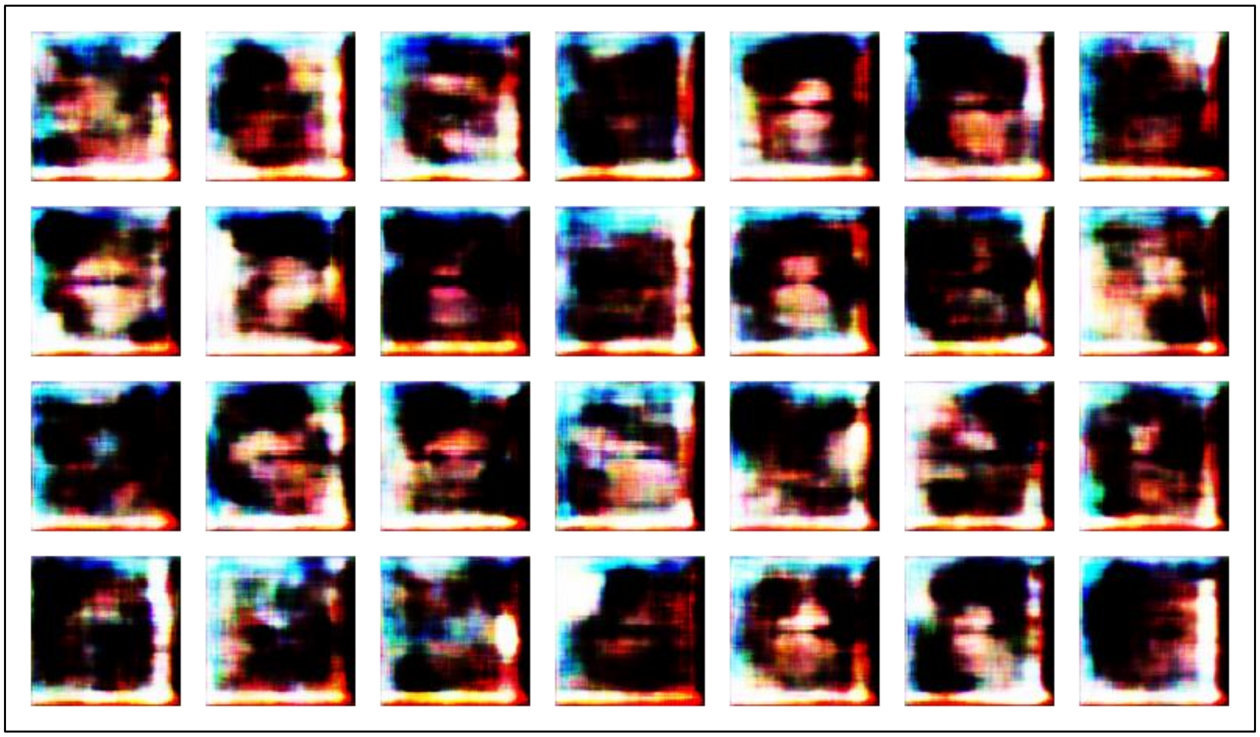


Fig: Training Image 1

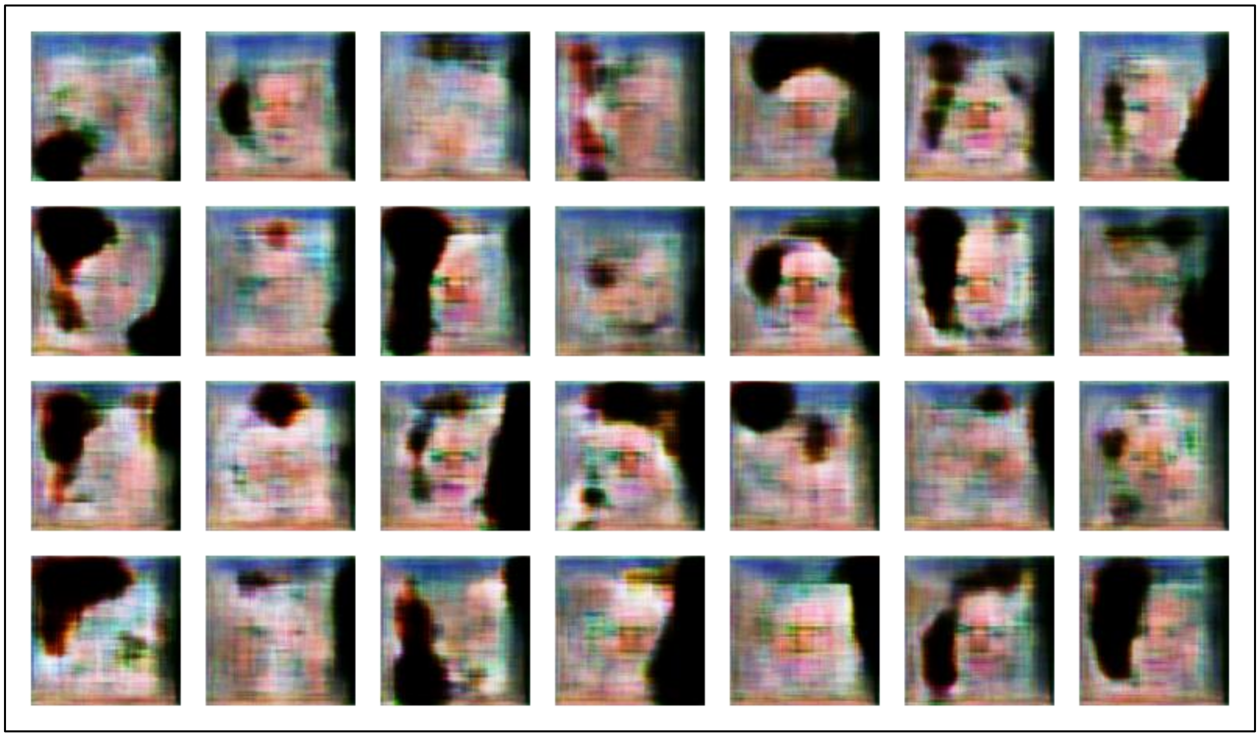


Fig: Training Image 9

The images in the training set 1 does not have that much clear images but training in the set 9, we can make out the outline of the images and we can also see detailed explaining od facial features like eyes, nose and lips. Considering we have somewhat facial features as training Image 9, we could get a face by training image 50.

MNIST Data on Gans

The MNIST data was used on gans to find out how do they work and there were two different ways that it was implemented. We will be discussing the first one here in our paper. The Gan which we make have 2 hidden layers both in generator and discriminator. We have a batch size of 256 and 13 epochs. Again, we did comparatively less epoch because of the time. Here is the end training result at end of 13th epoch.

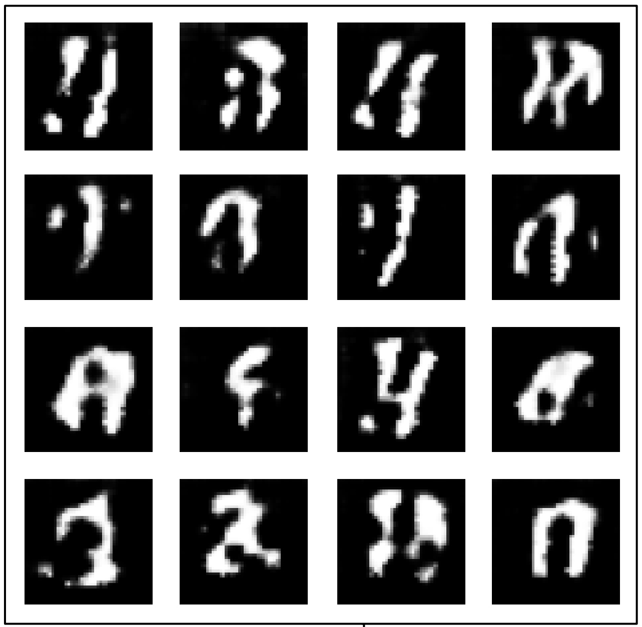
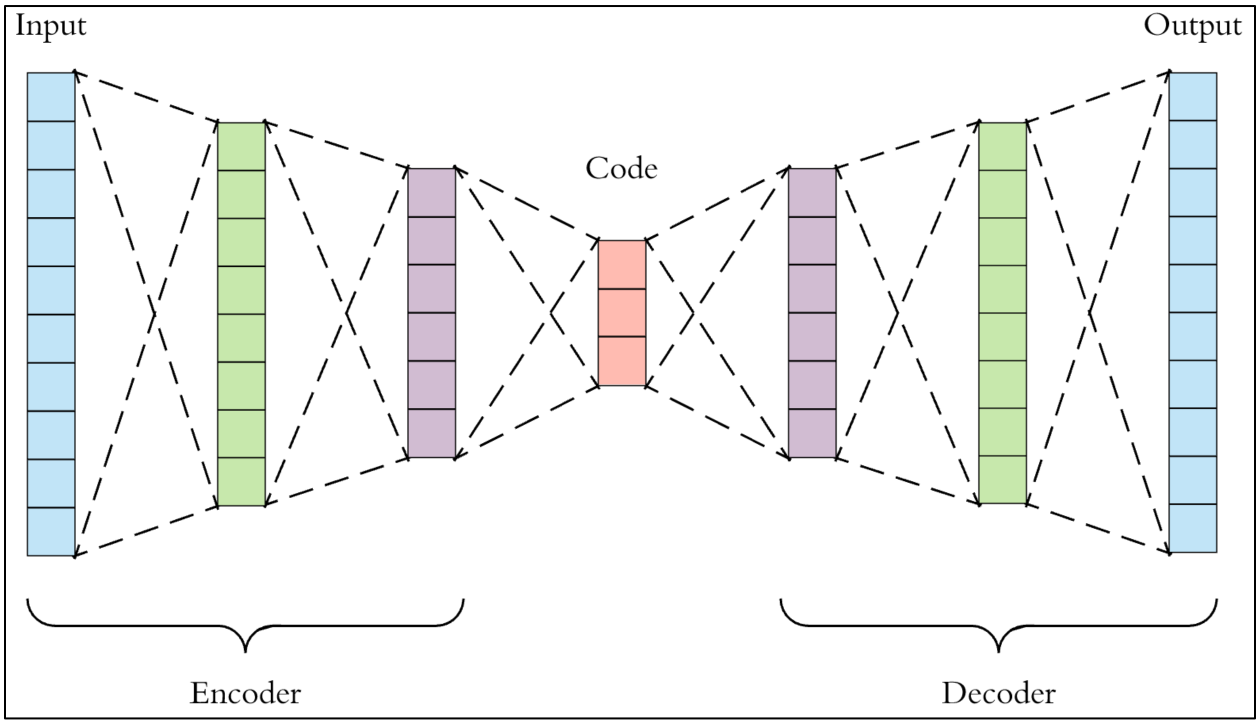


Fig: End of 13th Epoch

Autoencoders

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. Autoencoders consists of 4 main parts:



1- Encoder: In which the model learns how to reduce the input dimensions and compress the input data into an encoded representation.

2- Bottleneck: which is the layer that contains the compressed representation of the input data. This is the lowest possible dimensions of the input data.

3- Decoder: In which the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.

4- Reconstruction Loss: This is the method that measures measure how well the decoder is performing and how close the output is to the original input.

MNIST on Autoencoders

MNIST was used to get the starting for autoencoders. The Task was to create similar image of after the autoencoder has been used. We have 3 layers of encoders and decoders. We used the number of units to be multiple of 2 so that we can easily backpropagate to get our results. Our epochs value was 60 by that point we get really good results.

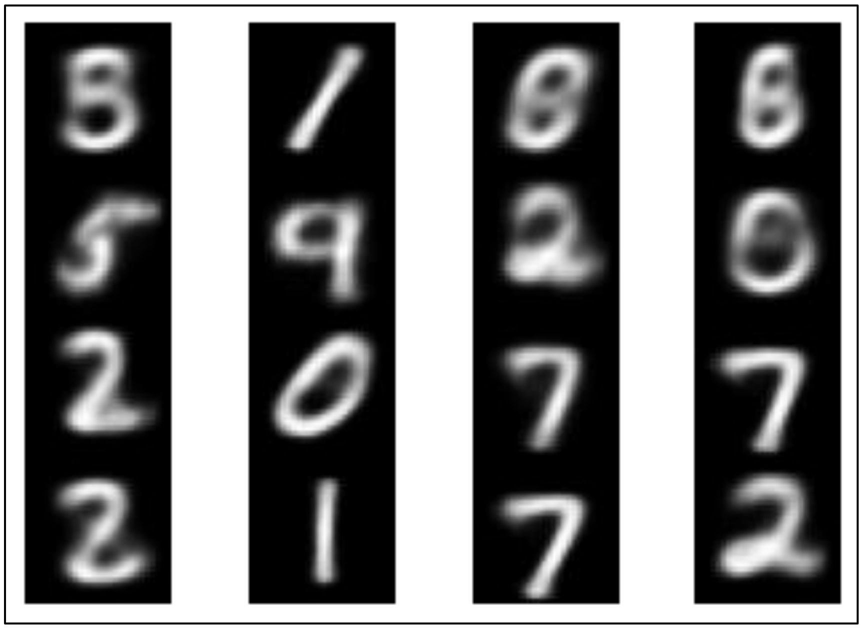


Fig: End of 60th Epoch

Face Construction using Autoencoder

In this project we tried to reconstruct a face using the autoencoders. The data which we consist of approximately 200,000 images of different people and we tried to reconstruct them using the autoencoding method. In the autoencoder, we have only one hidden layer. We run the code for 25 epochs and got optimum result. The loss function was low for both testing and training dataset.

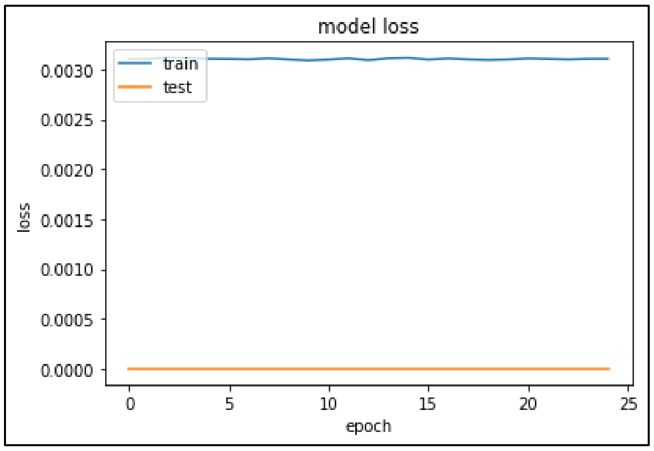


Fig: Loss Function Graph

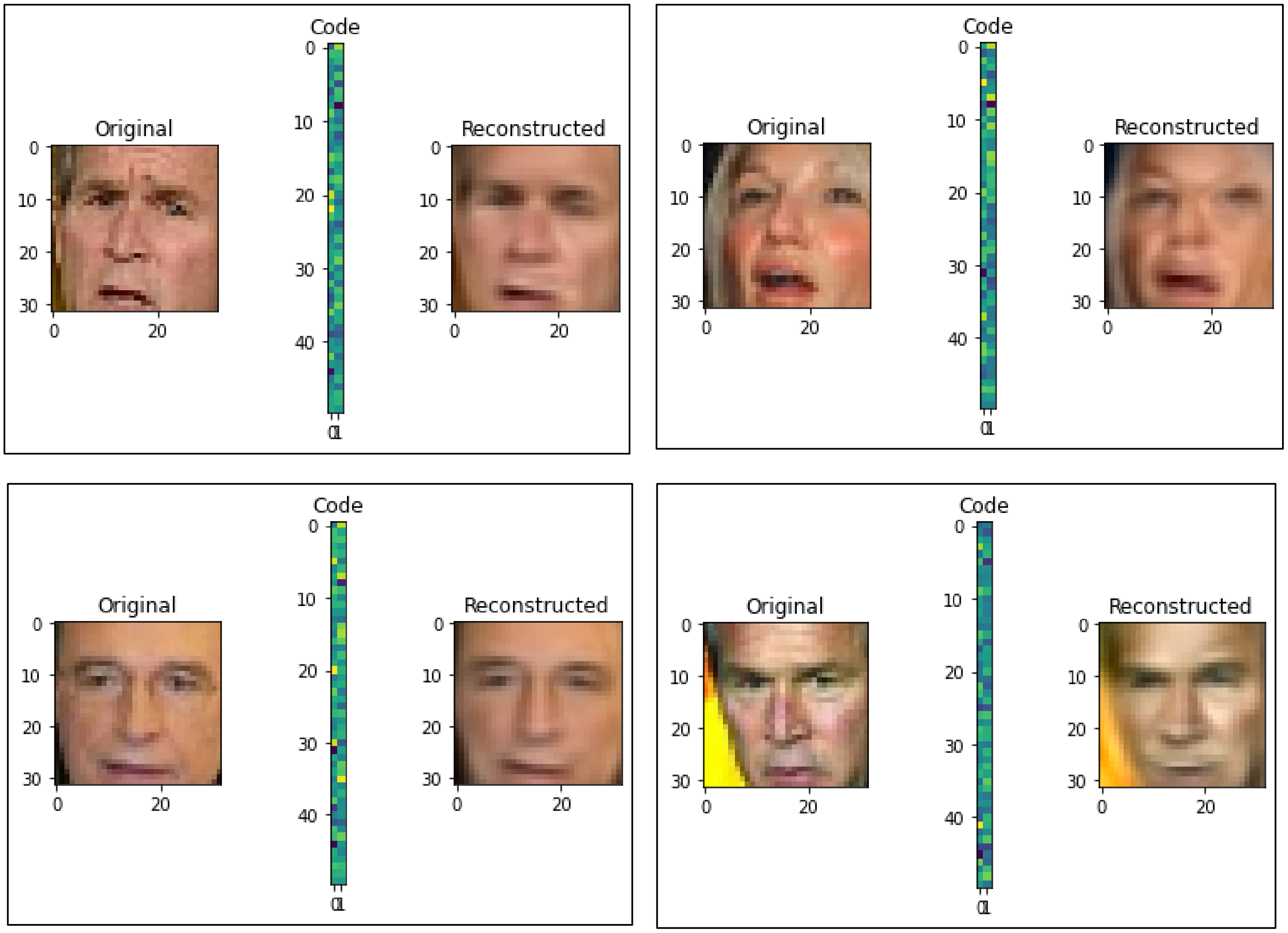


Fig: Result of 25th Epoch

After these numerous projects that I have done in the internship, I was able to get a deep understanding of Machine learning and appreciate them for taking out time from there busy schedule for this.

References

A. Mechelli, S. Vieira (2020), Machine Learning: Methods and Applications to Brain Disorders

Dogs vs. Cats | Kaggle. (2020). Retrieved 6 October 2020, from https://www.kaggle.com/c/dogs-vs-cats

Advanced Topics in GANs. (2020). Retrieved 7 October 2020, from https://towardsdatascience.com/comprehensive-introduction-to-turing-learning-and-gans-part-2-fd8e4a70775

Understanding of Convolutional Neural Network (CNN) — Deep Learning. (2020). Retrieved 7 October 2020, from https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Auto-Encoder: What Is It? And What Is It Used For? (Part 1). (2020). Retrieved 7 October 2020, from https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

Building Autoencoders in Keras. (2020). Retrieved 7 October 2020, from https://blog.keras.io/building-autoencoders-in-keras.html

Advanced Topics in GANs. (2020). Retrieved 14 October 2020, from https://towardsdatascience.com/comprehensive-introduction-to-turing-learning-and-gans-part-2-fd8e4a70775

Brownlee, J. (2020). How to Develop a CNN From Scratch for CIFAR-10 Photo Classification. Retrieved 14 October 2020, from https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/

Keras Autoencoders: Beginner Tutorial. (2020). Retrieved 14 October 2020, from https://www.datacamp.com/community/tutorials/autoencoder-keras-tutorial