# Deep learning

For this project I chose to use the MNIST dataset as I wanted to focus on computer vision, and this seemed like a good starting point. I have used three deep learning methods which are a multi-layered perceptron, a Convoluted Neural Network and a Generative Adversarial Network.

## The data

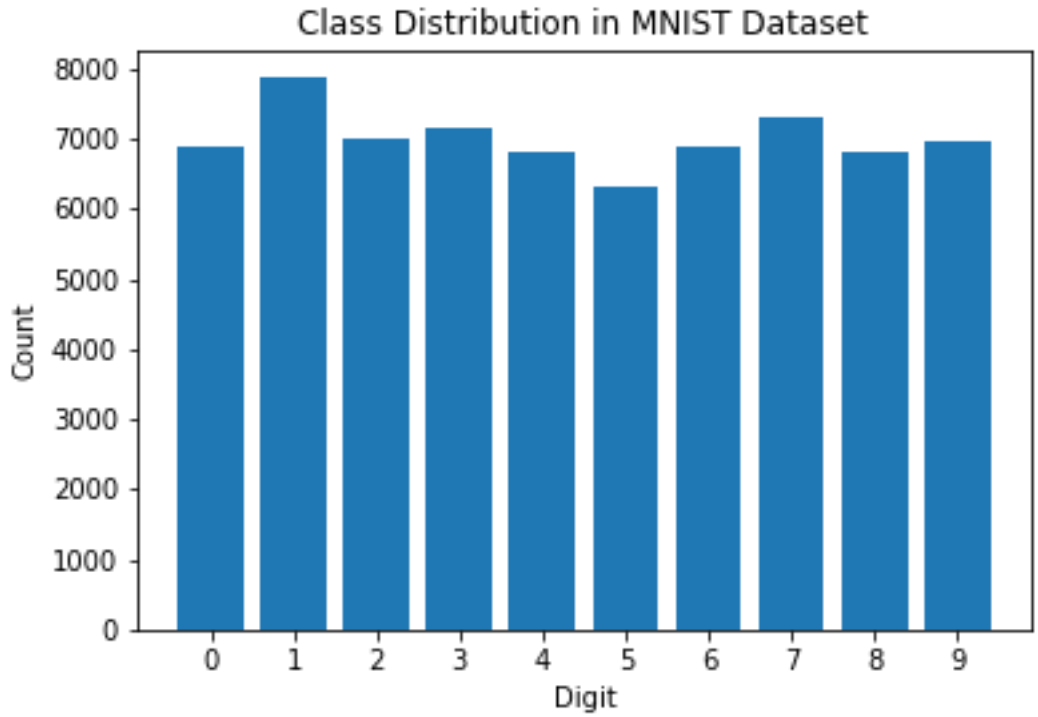


Figure 1 – The class distribution of the dataset, indicating that the class are roughly equal; therefore, no up or down sampling was required.

This dataset also has no null values, and due to the well-maintained nature of the data I started with the project very quickly.

## The Multi-Layered Perceptron

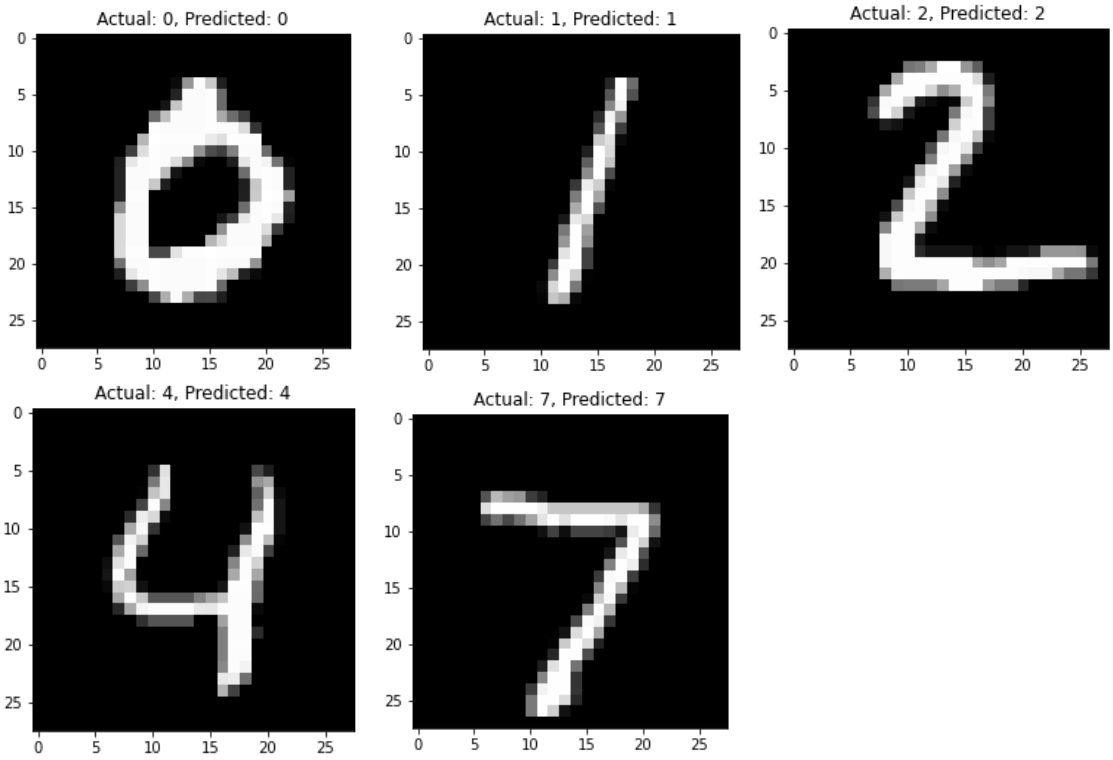


Figure 2 – Five of the test cases correctly identified.

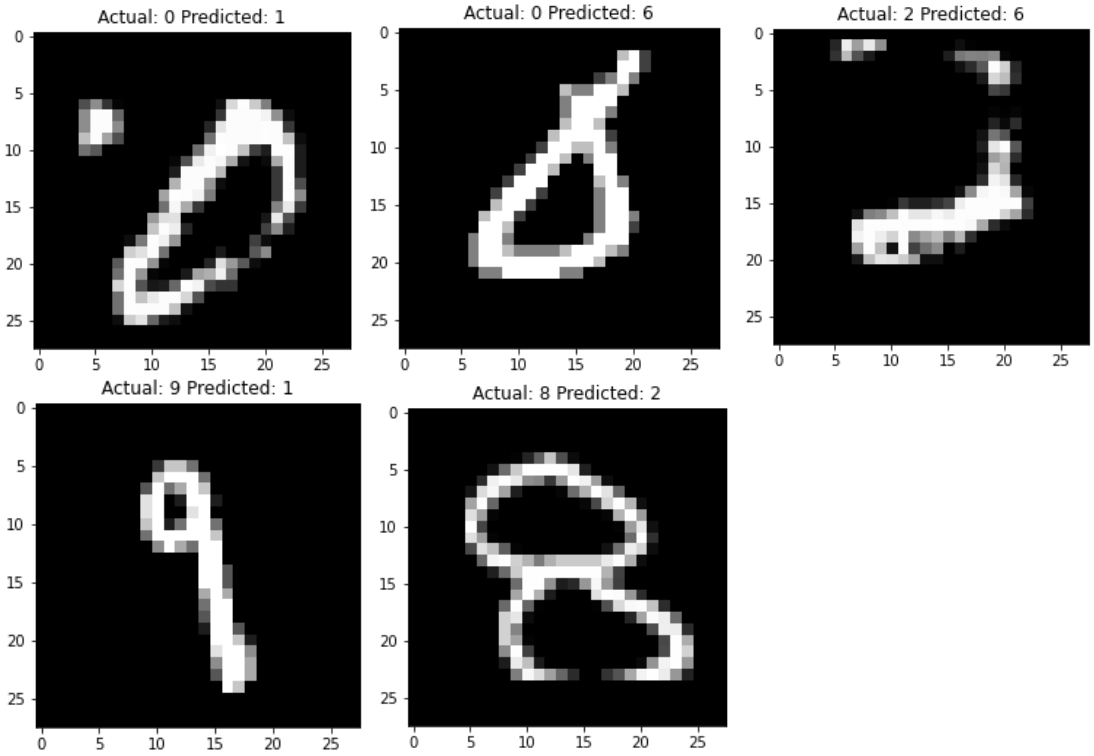


Figure 3 – Five of the test cases incorrectly identified.

Figure 3 shows that the model struggled with both clearly identifiable numbers, but primarily seemed to struggle with poor handwriting.

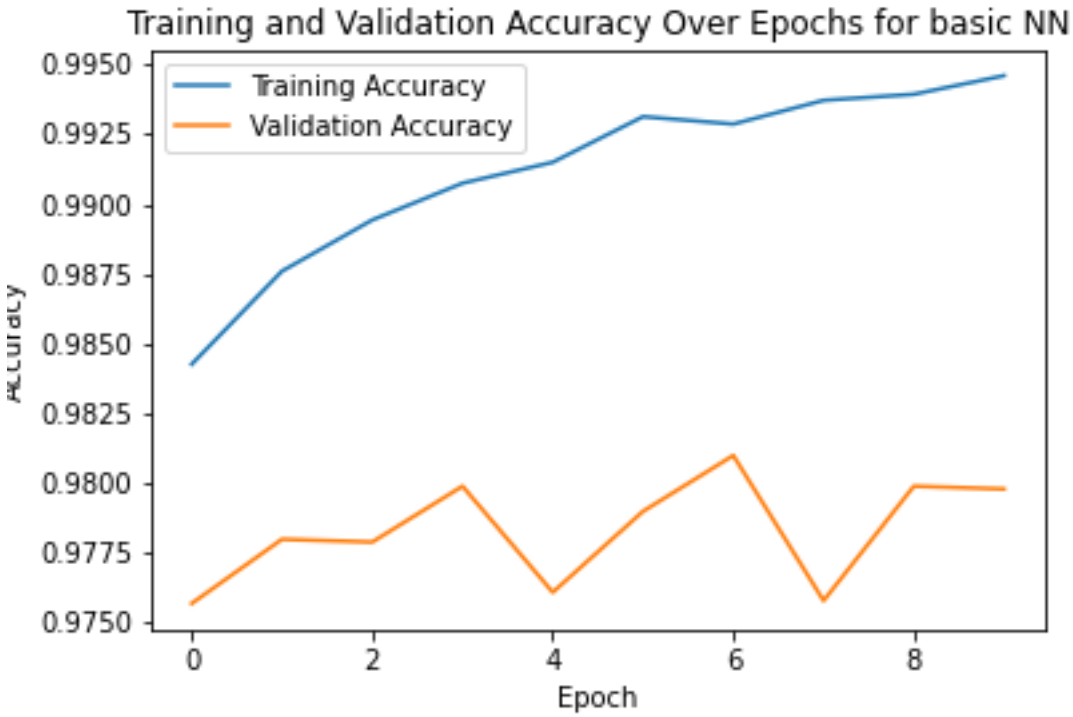


Figure 4 – The accuracy of the multi-layered perceptron over the epochs.

The above figure indicates that the model may have had insufficient training as both the validation and test accuracy were trending upwards even towards the end of the training period. I did not continue further testing of this model as I had hoped to use it as a baseline for comparisons with the more complex models.

## The Convoluted Neural Network

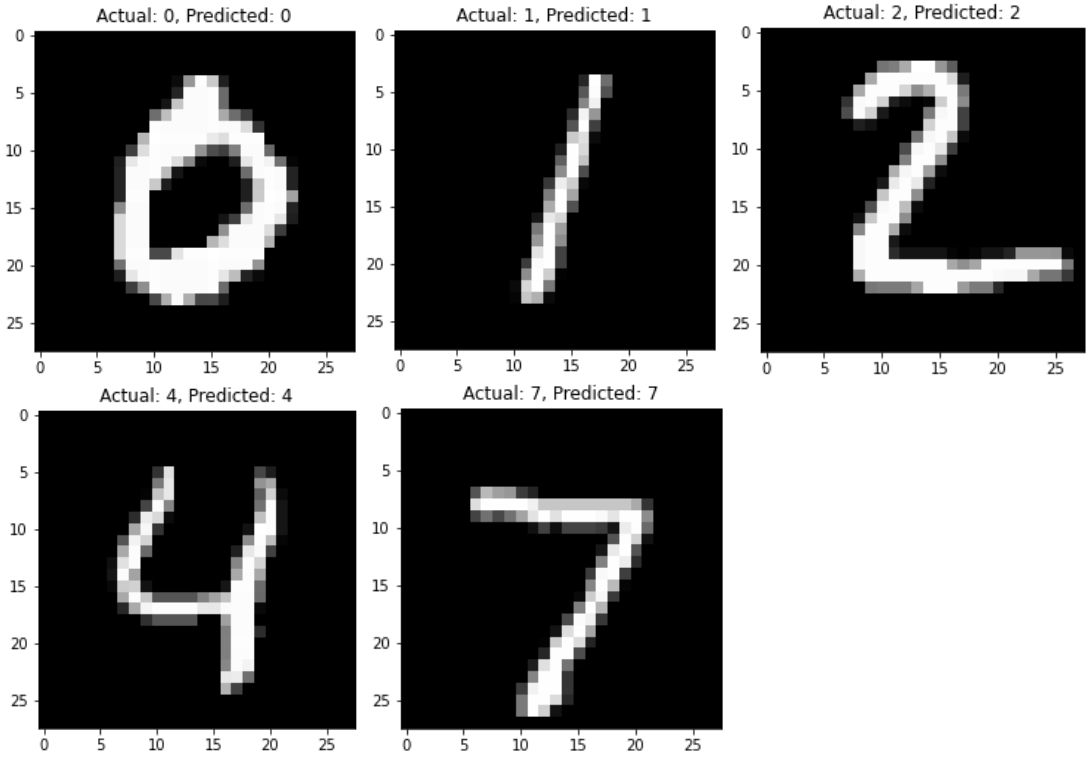


Figure 5 – A sample of five images that were correctly identified by the CNN.

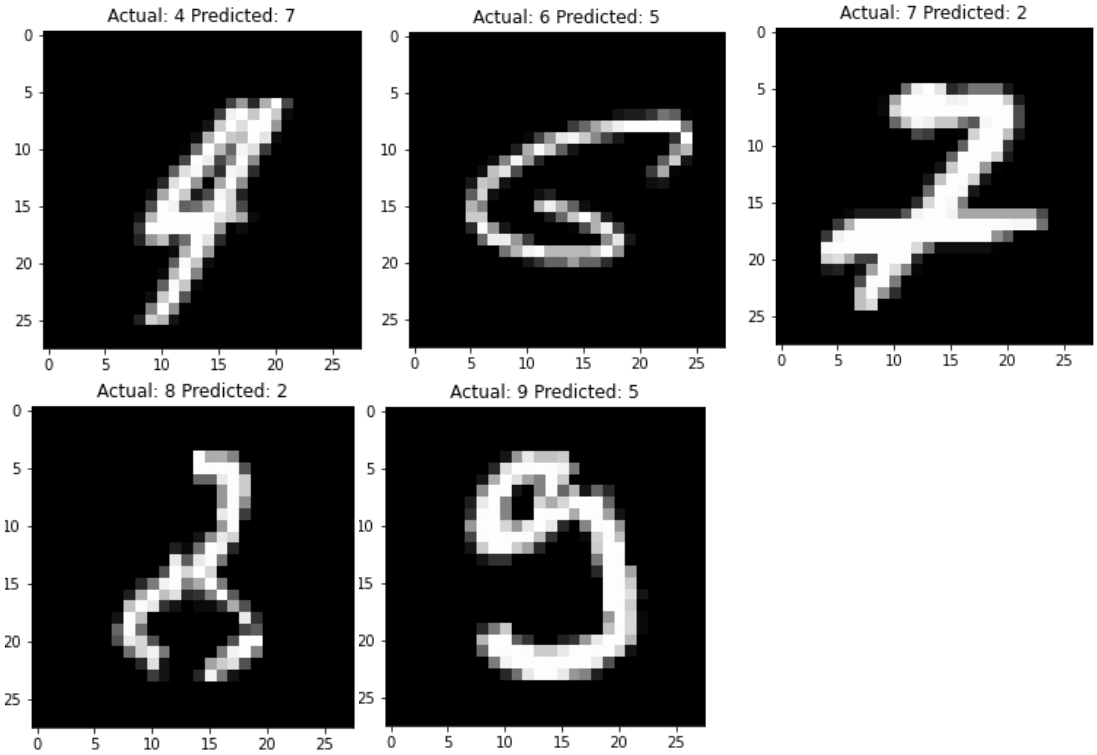


Figure 6 – A sample of five images incorrectly identified by the CNN.

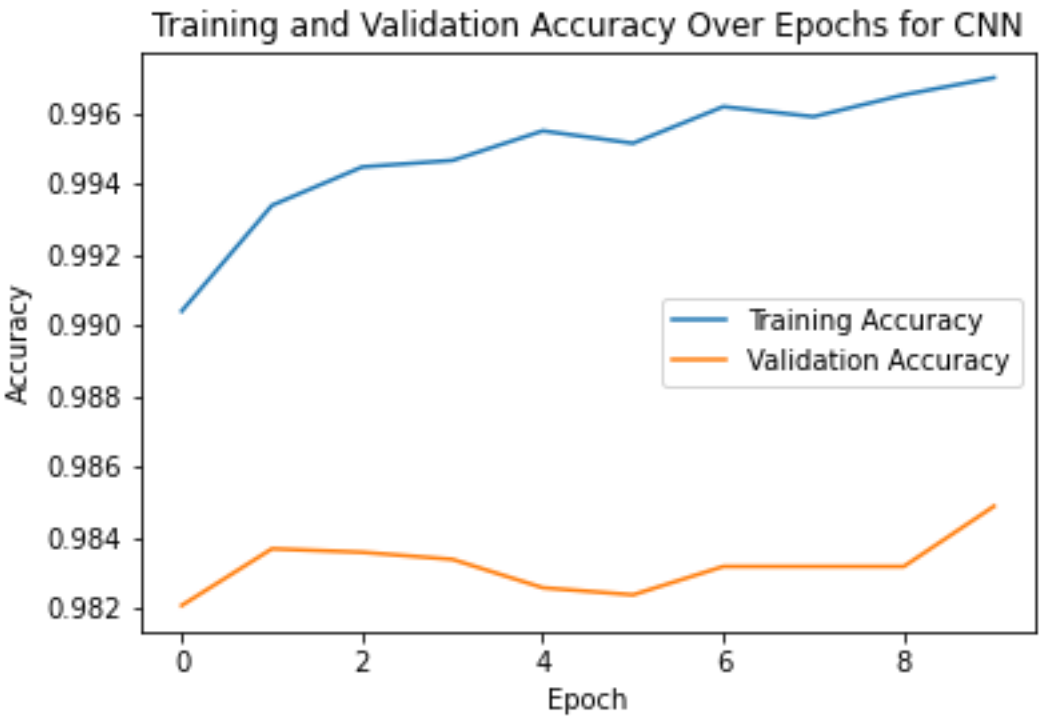


Figure 7 – The accuracy of the CNN over the epochs.

## The GAN

I had planned to train a GAN and then take the discriminator and have it differentiate between the numbers; however, this sadly did not work as expected.

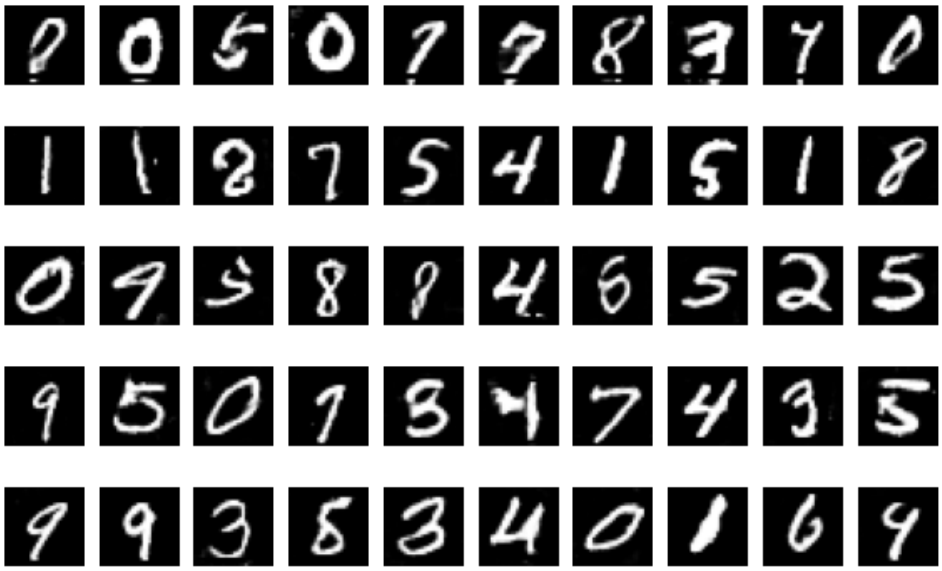

Figure 8 – The images generated by the GAN throughout its training, from epochs: 20, 40, 60, 80, and 100.

Unfortunately, while the GAN was able to be built, and the numbers generated were realistic; however, the kernel died, and I doubt my laptop could have retrained the model, nor did I have the patience to rerun the training.

## Results

Based off what I have, in terms of accuracy and training time, the CNN was the best model for this project with an impressive validation set accuracy of 98%, and taking maybe 10 minutes to train, it is the best option to tune hyperparameters and would allow exceptional results to be achieved the fastest.

## Future Work

I think that both the CNN and the MLP could both have used an increased training period, whereas the GAN would simply need to be retrained; however, to retrain the GAN I would need improved hardware as the training took almost a full day and has resulted in a significant decrease to my laptop's battery life.