# EE5907 Assignment 1 Report

Lee Si En, A0167775L

October 2020

## 1 Principal Component Analysis

**Introduction:**

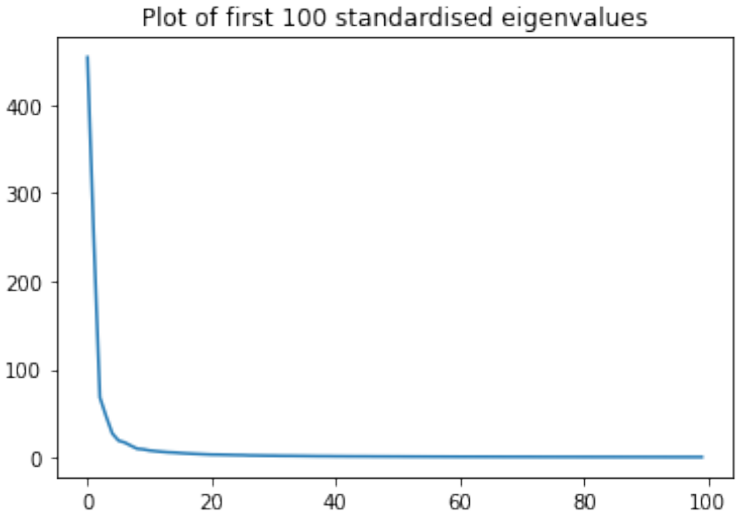
Principal component analysis (PCA) is a unsupervised feature extraction method that is commonly used is exploratory data analysis. The general idea behind PCA is to compute principal components of a dataset, which are directions in n-dimensional space where the greatest variability is observed. After that, the dataset can be expressed with its most significant principal components.

**Outline of process:**

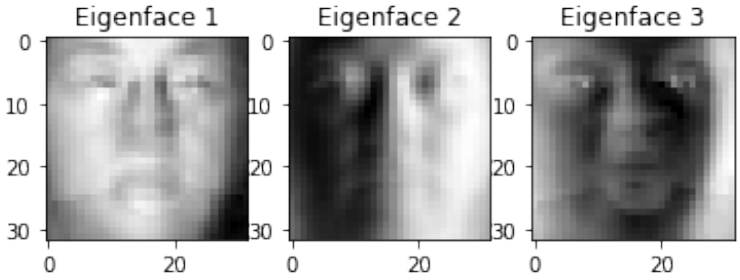
1. Find the covariance matrix of the dataset.
2. Find the eigenvectors and eigenvalues of the matrix of the covariance matrix. The index of the largest eigenvalue corresponds to the index of the eigenvector of the first principal component.
3. Apply the desired principal component(s) to our dataset to reduce the dimensionality of the dataset.

**Results:**

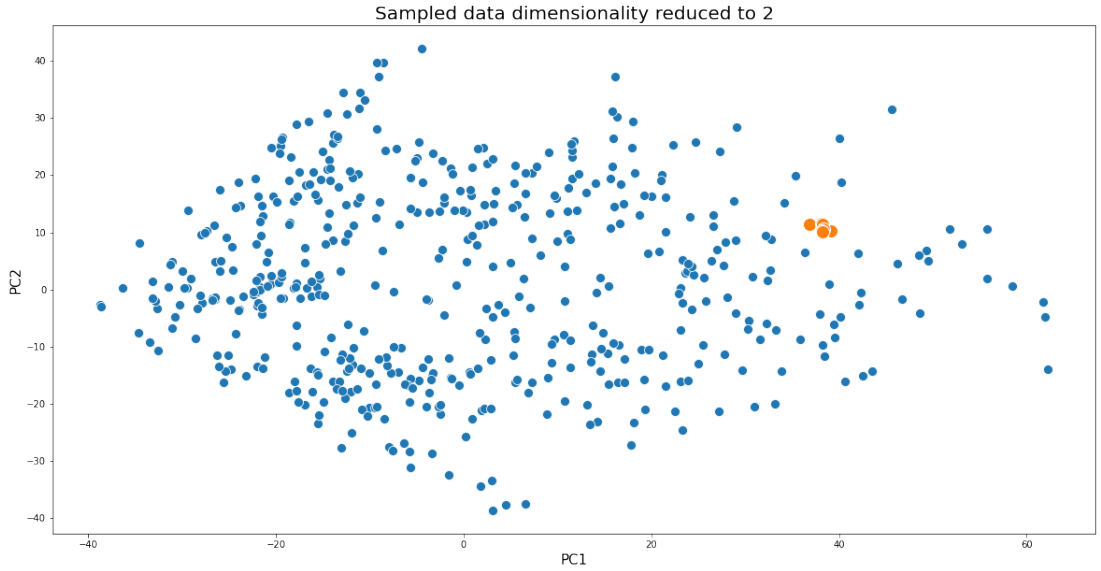
Through the aforementioned process, we plot the first 100 eigenvalues to make sure it is in descending order, as well as to get a sense of how much each eigenvalues contributed to the total energy.

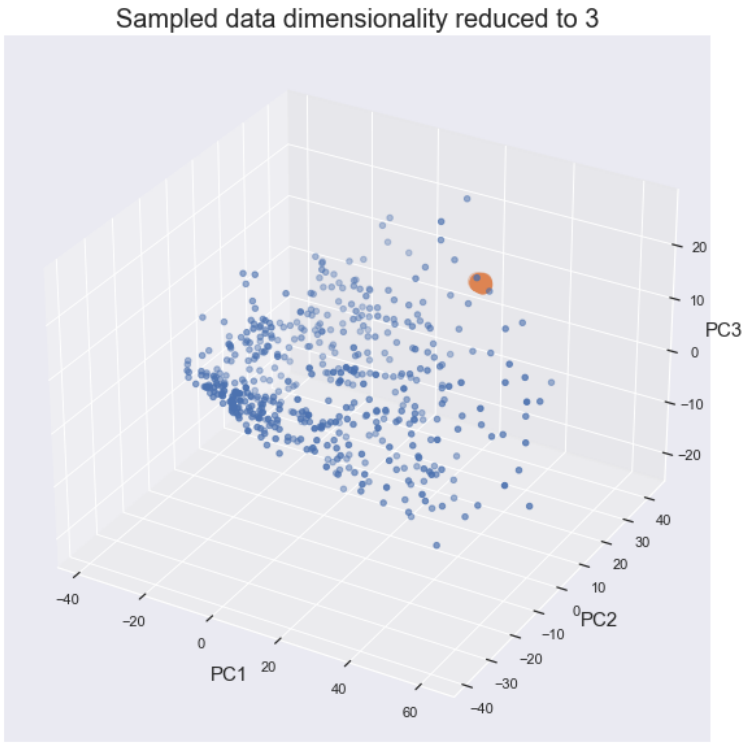



After that, the first 3 eigenfaces are visualised.




Finally, the 2 dimensional and 3 dimensional representation of the dimensionally-reduced images are plotted below. The orange points are the selfie images.


Sampled data dimensionality reduced to 2
PC1
PC2


Sampled data dimensionality reduced to 3
PC1
PC2
PC3

**Classification Accuracy for PCA Vectors:**

| Number of dimensions | 40 | 80 | 200 |
|---|---|---|---|
| CMU PIE Training Accuracy (%) | 100 | 100 | 100 |
| CMU PIE Testing Accuracy (%) | 93.56 | 94.27 | 94.74 |
| Selfie Training Accuracy (%) | 100 | 100 | 100 |
| Selfie Testing Accuracy(%) | 100 | 100 | 100 |

It is unsurprising that the training accuracy is always 100%. This is because the classifier being used is K Nearest Neighbour, and closest neighbour of the training set is itself. Something worth noting is that the testing accuracy increases with the number of dimensions used to represent the dataset. Intuitively, this makes sense because more energy is being captured as dimensions increase, leading to a more "accurate" representation of the original dataset.

## 2 Linear Discriminant Analysis

**Introduction:**

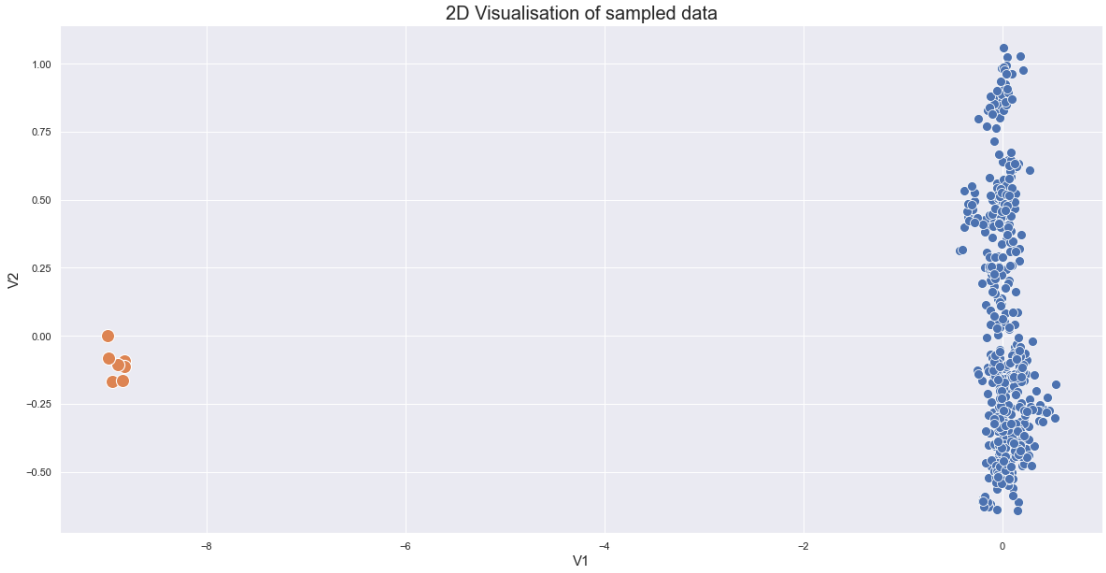
Linear discriminant analysis (LDA) is a method that finds a linear combination of features the separates two or more classes. Ideally, LDA aims to maximise the distane between points belonging to different classes, and minimise distance between points belonging to the same class.

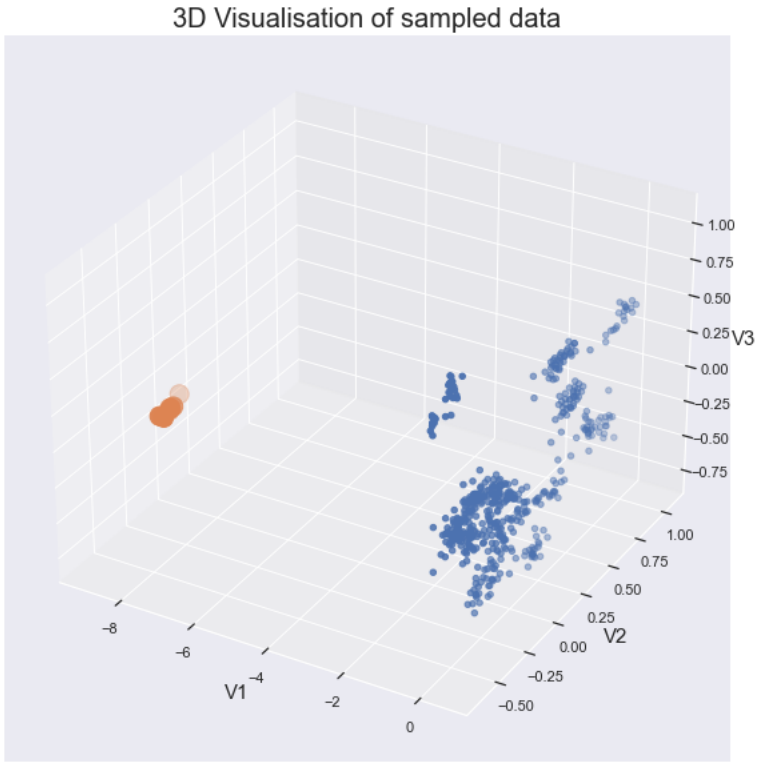
**Outline of process:**

1. Find the mean vector of each class, as well as the overall mean vector of the dataset.
2. With those, we can calculate the within-class and between-class scatter matrices.
3. Following that, we obtain the eigenvectors and eigenvalues of the product of the inverse of within-class scatter matrix, and the between-class scatter matrix.
4. Sort these eigenvectors and eigenvalues by descending order, and the eigenvectors can be applied to the original dataset to reduce data dimensionality.

**Results:**

Data dimensionality of the dataset is reduced to 2, 3 and 9. The two-dimensional and 3-dimensional representation is visualised below. The orange points are the selfie images.


2D Visualisation of sampled data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
V2
−8
−6
−4
−2
0
V1


3D Visualisation of sampled data
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
V3
−8
−6
−4
−2
0
V1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
V2

**Classification Accuracy for LDA Vectors:**

| Number of dimensions | 2 | 3 | 9 |
|---|---|---|---|
| CMU PIE Training Accuracy (%) | 100 | 100 | 100 |
| CMU PIE Testing Accuracy (%) | 24.15 | 46.20 | 89.96 |
| Selfie Training Accuracy(%) | 100 | 100 | 100 |
| Selfie Testing Accuracy(%) | 100 | 100 | 100 |

Through the plots, LDA appears to be highly effective in separating the selfie images from the CMU PIE set. One final thing worthy of note: in my initial attempt to build the function, I used np.linalg.inv to calculate the inverse of the within-class scatter matrix. Even though its classification accuracy appeared normal, the plots did not resemble sklearn.LDA's plot. Therefore, I experimented with np.linalg.pinv instead, and obtained a plot highly similar to sklearn's.

# 3 Gaussian Mixture Model

**Introduction:**

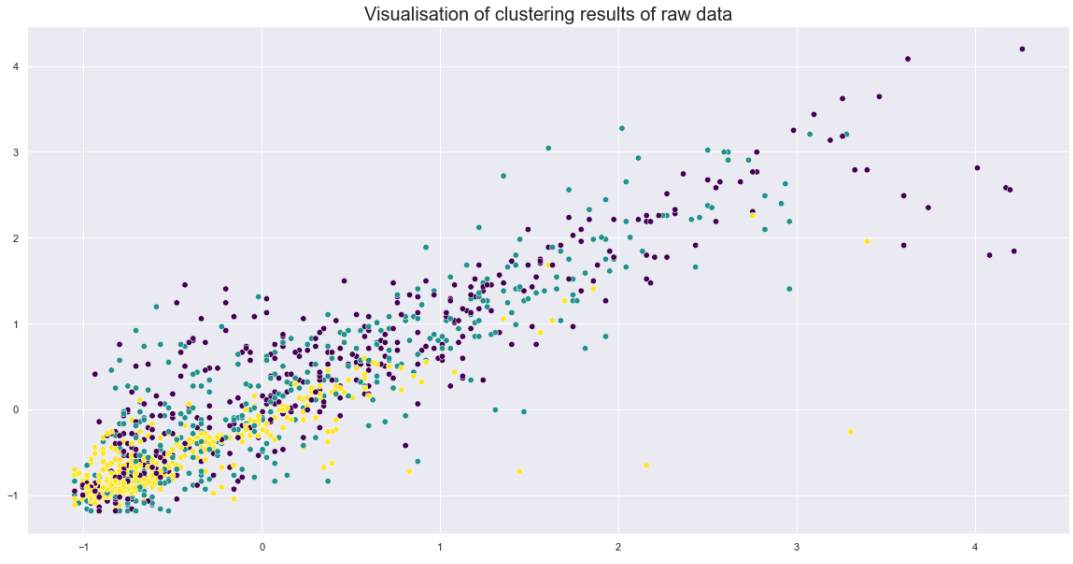
Gaussian mixure models are probabilistic models that assume that all data points are generated from a finite number of Gaussian distributions. To determine the parameters of the Gaussian distributions, the expectation-maximisation algorithm is employed.

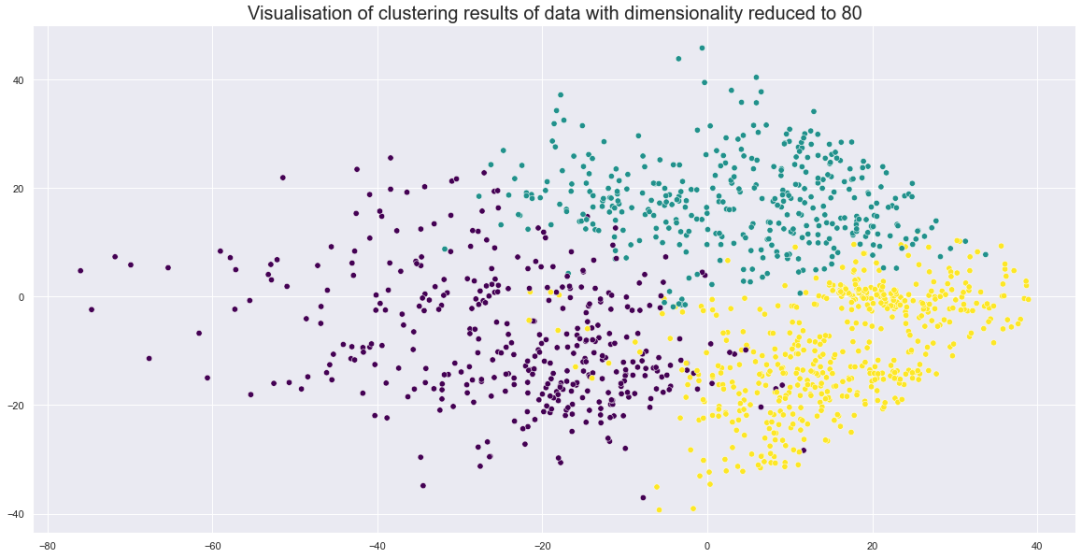
**Outline of process:**

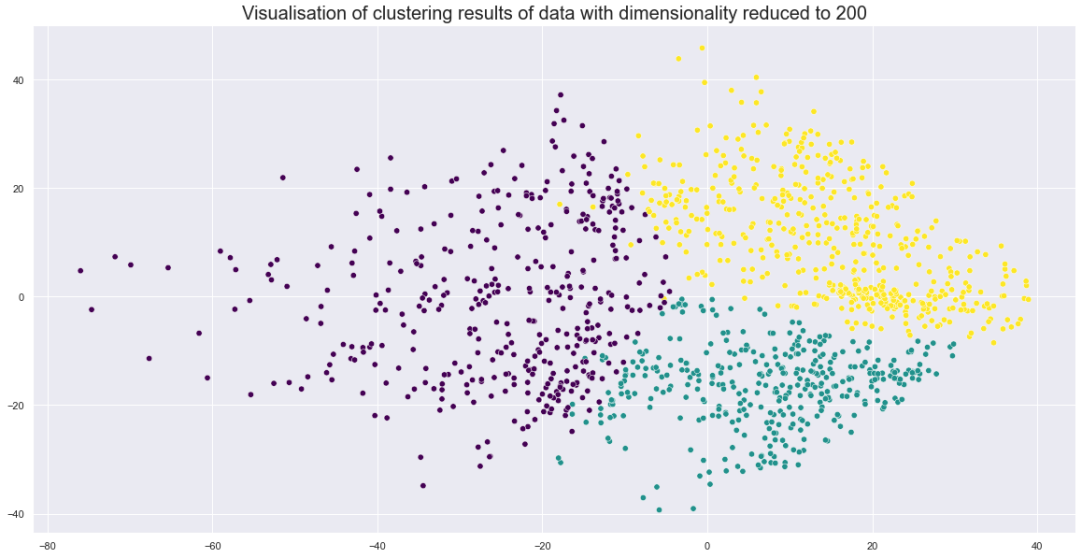
1. First, initialise $\theta$.
2. After that, calculate the probability of Z, a set of unobserved latent data, using $\theta$.
3. Then, use Z to find an improved estimate of $\theta$.
4. Steps 2 and 3 are repeated recursively until convergence.

**Results:**

A GMM model with 3 Gaussian components was trained on the following training data: raw face images, face vectors after PCA pre-processing, reduced to 80 and 200 dimensions. This model is applied to the testing data, which are then visualised on two-dimensional plots.




Visualisation of clustering results of data with dimensionality reduced to 80
40
20
0
−20
−40
−80
−60
−40
−20
0
20
40


Visualisation of clustering results of data with dimensionality reduced to 200
40
20
0
−20
−40
−80
−60
−40
−20
0
20
40

# 4 Support Vector Machines for Classification

**Introduction:**

Support vector machine (SVM) is a supervised learning model that performs classification. SVM is based around the idea of finding a separating hyperplane that best divides the dataset into its respective classes.

**Classification Accuracy for Linear SVMs:**

| C | 0.01 | 0.1 | 1 |
|---|---|---|---|
| Training 80 (%) | 97.69 | 99.50 | 100 |
| Testing 80 (%) | 98.85 | 96.95 | 96.71 |
| Training 200 (%) | 99.66 | 100 | 100 |
| Testing 200 (%) | 98.89 | 97.97 | 97.73 |
| Training unreduced (%) | 99.93 | 100 | 100 |
| Testing unreduced (%) | 97.81 | 97.50 | 97.50 |

The C parameter determines how much the SVM model avoids misclassifying each data point. When C is large, a smaller-margin hyperplane is chosen to better classify data points. On the other hand, when C is small, a larger-margin hyperplane is chosen, even at the cost of mislassifying more points. Therefore, for smaller values of C, we should expect accuracy to worsen.

However, this is not consistent in our results. Increasing C improves the training accuracy, but worsens the testing accuracy of all 3 datasets. Furthermore, there is not clear trend for changing dimensionality. Arguably, increasing the dimensionality should yield a better accuracy, because more dimensions are preserved from the PCA transformation, meaning total energy should increase. Therefore, the model can train on a larger amount of the original data. but this is not explicitly show in the results.

Interestingly, larger values of C seem to require more iterations to converge. Whenever C increases, there is a tendency for sklearn to give the "convergence warning". Therefore, the maximum iterations was increased tenfold to allow convergence, but at the cost of computation speed.

# 5 Neural Networks

**Introduction:**

A neural network is a highly interconnected system with input, output, in-between layers. Each layer comprises of many nodes, and neurons are successively connected. During forward propagation, every node contains a function of the values in the nodes in the previous layer, and the weights of each connection. A cost function is recomputed during the learning phase, while back-propagation updates the weights of the previous connections.

CNN is a class of deep neural networks most commonly applied to analyzing visual imagery.The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication.

**Outline of process:**

The following neural network was modelled. One modification I made was to increase the number of nodes in the final layer from 21 to 26. This is because the assignment required us to sample 25+1 faces in total, not 21.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 28, 28, 20)        520
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 20)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 50)        25050
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 50)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1250)              0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               625500
_________________________________________________________________
dense_3 (Dense)              (None, 26)                13026
=================================================================
Total params: 664,096
Trainable params: 664,096
Non-trainable params: 0
_________________________________________________________________
```

**Results:**

Minimal optimisation was performed. However, a few things were worth noting:

1. 'Adam' was used an optimiser, and 'categorical_cross_entropy' was required because we had more than two classes.

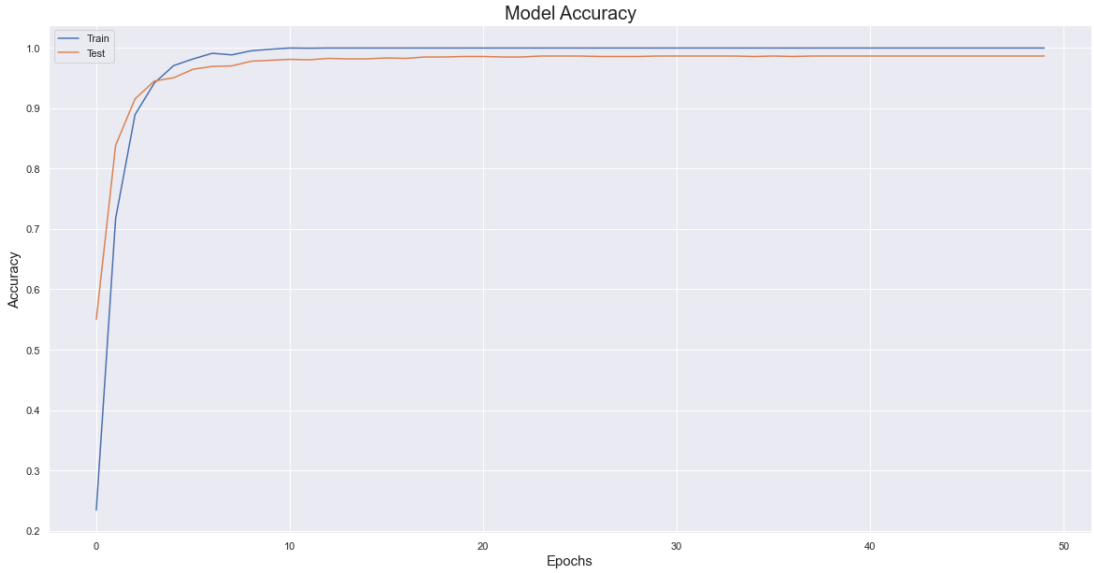
2. Technically, one hot encoding is not required because our target is non-categorical. However, because the integers in our target are not evenly spaced, converting them to binary form seemed more suitable for the task

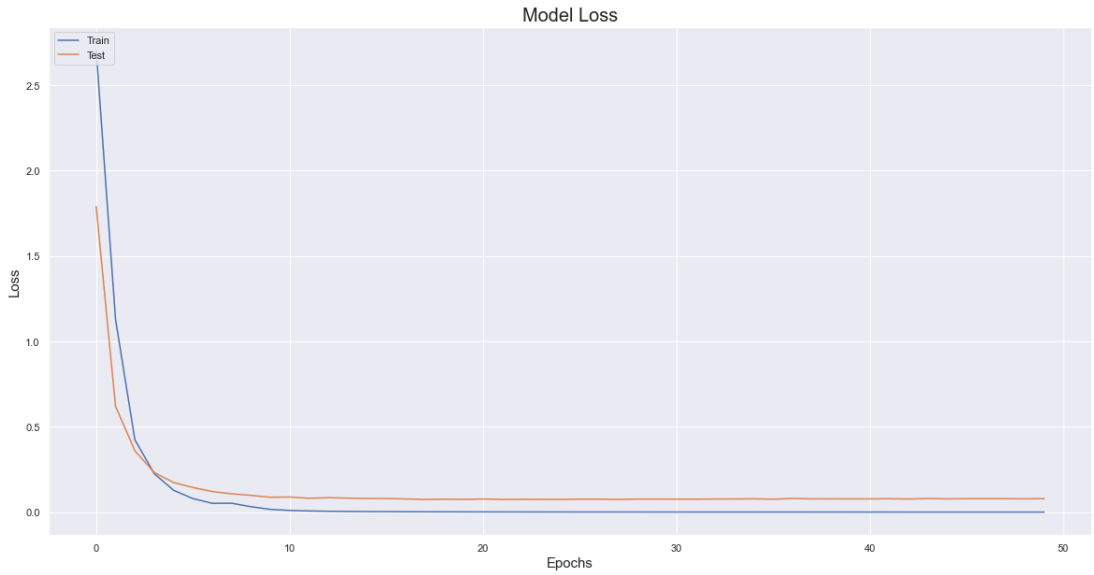
3. 'Softmax' was used at the final dense layer because we had more than two classes.

The following results were obtained:

**Classification Accuracy for Neural Networks:**

| | Accuracy | Loss |
|---|---|---|
| Training (%) | 100 | 0.023 |
| Testing (%) | 98.44 | 8.71 |






There is slight overfitting, but overall, the convolutional neural network appeared to perform extremely well in classifying our images.

However, one problem with neural networks is the lack of interpretability. Few people can fully understand the inner workings of a neural network, making it hard to persuade people to employ neural networks for important tasks, such as interpreting an MRI scan. Because of that, new methods for model intepretability need to be looked into.

# 6 Reflections

Even though this part is not required, I would like to include it anyway. This assignment was actually harder than the first one in my opinion, even though less code was actually written. It's probably because the second half of the module requires a different kind of understanding of mathematics, mainly to do with linear algebra, which is something I am weak in.

That being said, I think I spent roughly 5-10 hours less than the first assignment, but time taken is not representative of its difficulty. However, it has been a fruitful experience.