Arunava Manna (A0232541R)

Module EE5907

CA 2 Report

Contents

DataSet Creation:

Within the main dataset folder, there are a total of 67 folders. We must choose 25 folders from them, including the selfie dataset, which is the 68th folder.

Dimensions of the actual image: 4250 total samples were collected. 32-inch height; 32-pound weight

Training Data Dimension: (2975,1024), where 2975 represents 70% of the total sample size and 1024 represents the number of pixels.

Testing Data Dimension: (1275,1024), where 1275 is the number of test samples (30% of total sample size) and 1024 is the number of pixels.

Because the number of selfie picture training samples is 7, the training data dimension becomes (2982,1024) after incorporating selfie photos.

The dataset changes with each iteration. The folderlist (31, 61, 45, 2, 12, 49, 20, 34, 15, 4, 40, 64, 13, 57, 54, 43, 38, 6, 42, 47, 46, 41, 7 22, 19) and selfie folder 68 are used to get the results shown below.

Manipulation of Data for CNN:

The training and test data are downsized into (sample size, 32, 32, 1) format, which is recognised by tensorflow.

Furthermore, using a dictionary and the sklearn MultilabelBinarizer function, all of the distinct labels are converted into a range of 0 to 25 classes.

PCA based data distribution visualization

2D Plot with 500 random sample data from 2982 sample with 2 PCA components:

Library Used: Matplotlib has been used for the below graph

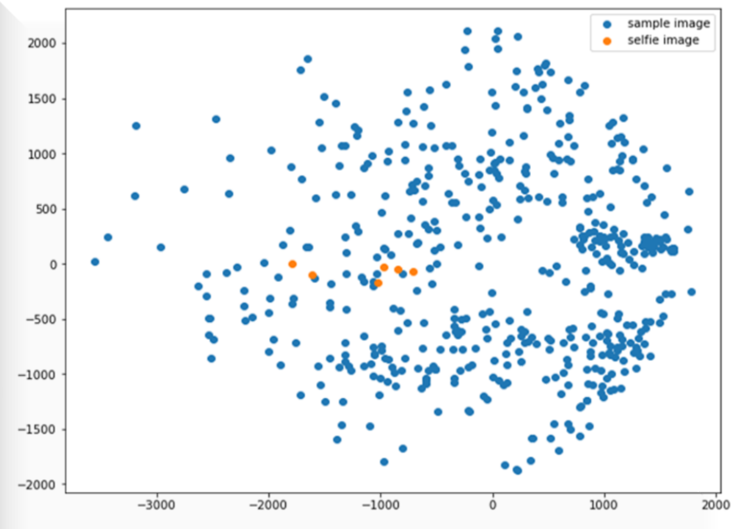


Fig 1: 2D Plot of 500 random sample

3D Plot with 500 random sample data from 2982 samples with 3 PCA components:

Library used: Plotly to plot the 3D plot in python

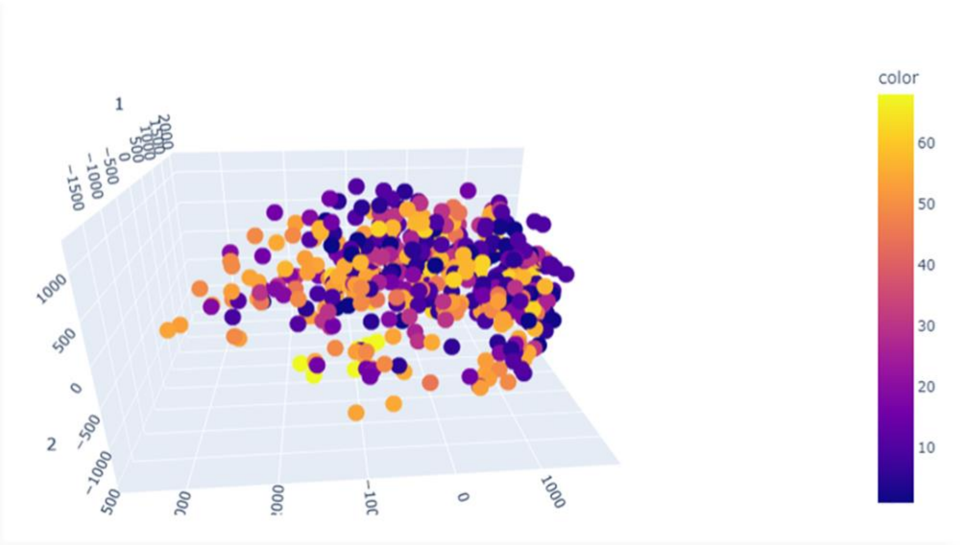


Fig 2: 3D plot of 500 random sample

Visualization of the corresponding 3 eigenfaces used for the dimensionality reduction.

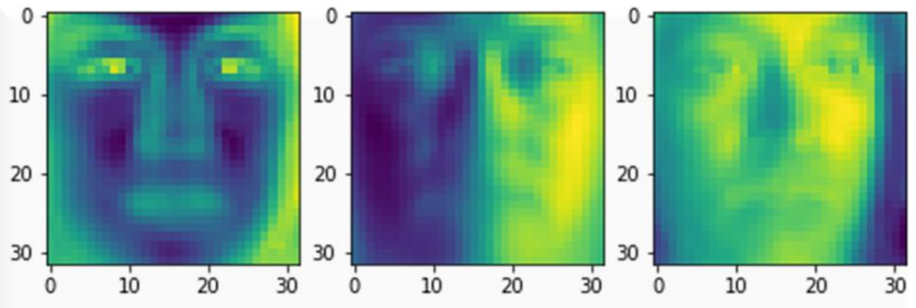


Fig 3. Eigen Faces

PCA plus nearest neighbor classification results

Table 1

Dimension	Accuracy from CMU test Image (%)	Accuracy From Selfie Image (%)
40	93.647	100
80	95.450	100
200	96.0	100

Explanation:

- Figures 1 and 2 depict how the data is spread along the 2 and 3 PCA axes, respectively, and the selfie image distribution is also visible via suitable markers. Figure 2 depicts each class with a different color.
- Figure 3 depicts the reconstructed faces from the PCA dimensionality reduction approach.
- According to table 1, the accuracy of the nearest neighbor classifier increases with the increase in dimension. The accuracy of the selfie image, on the other hand, is always the same.

My conclusion on the PCA method is that we can get high accuracy even when the dimension of the data is reduced and the image can be restored to initial condition again. With each iteration, the accuracy pattern can be seen similar.

LDA based data distribution visualization

2D plot with 2982 Data sample for 2 LDA components:

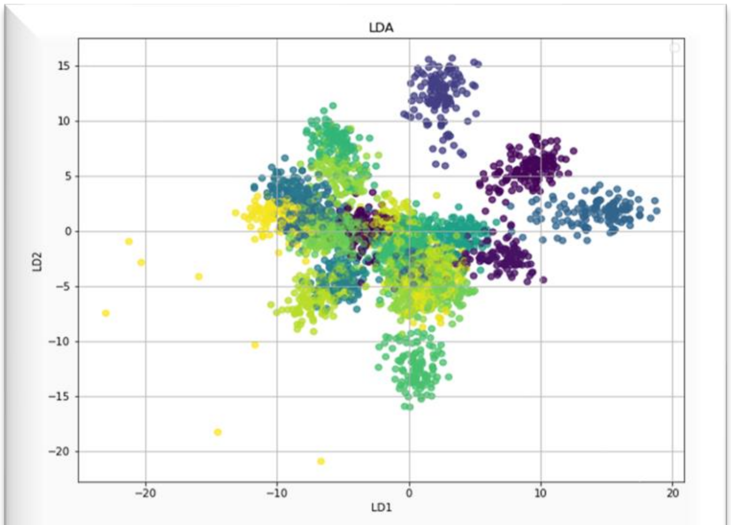


Fig 4: 2D plot with 2982 Data Sample

Explanation:

- From figure 1 the scatteredness of the data can be visualized for the 2 LDA components.
- Each color is representing one label or one face class.
- The selfie data points can be easily distinguished from other data points

3D plot with 2982 Data sample for 3 LDA components

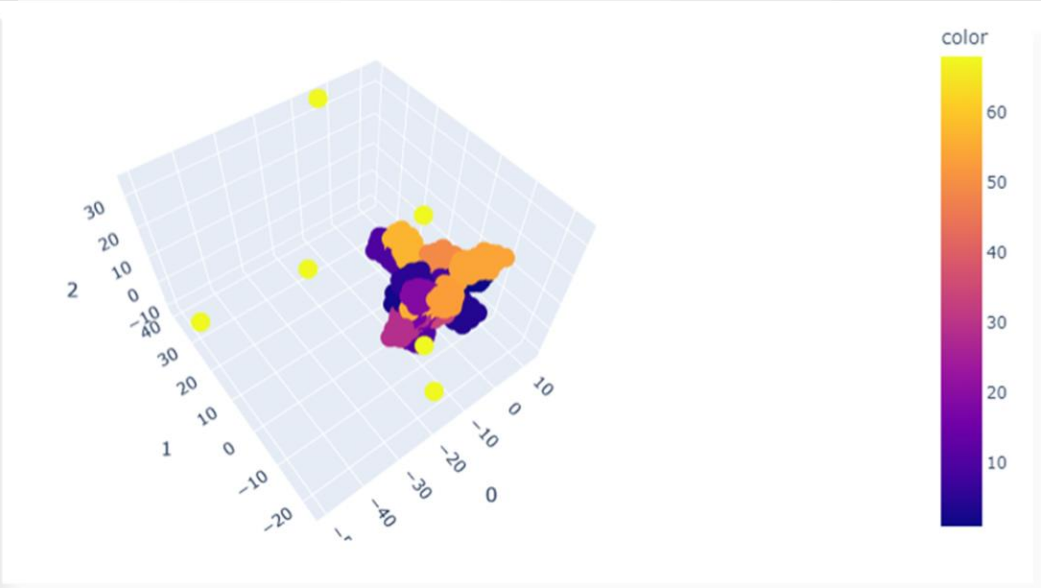


Fig 5: 3D plot of 2982 samples

LDA plus nearest neighbor classification results

Table 2

LDA Component	Accuracy from CMU test Image (%)	Accuracy From Selfie Image (%)
2	46.745	33.33
3	61.803	66.66
9	91.686	33.33

Explanation:

- Figure 3 helps us to visualize how the data is scattered along the 3 LDA components. The color bar is representing different class.
- In LDA, the accuracy is increasing with the increase in components while the accuracy for the face images is different for different LDA components.

As a feature extraction tool, LDA performs admirably for the provided data set, in my opinion. With each iteration, the accuracy pattern can be seen similar.

SVM classification results with different parameter value

Library Used: **libsvm**

Table 3

PCA Dimension	C value	Accuracy from CMU test Image (%)
80	0.01	97.1765
	1	97.1765
	0.1	97.1765
200	0.01	97.7255
	1	97.7255
	0.1	97.7255
1024	0.01	97.6471
	1	97.6471
	0.1	97.6471

Explanation:

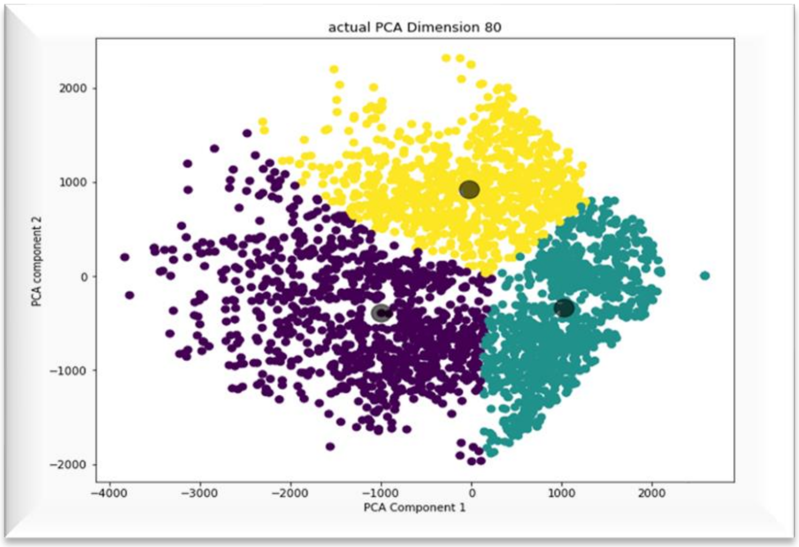
- From the above table, it can be concluded that the accuracy is almost similar for different dimension of PCA.
- Choosing a low c will provide a large minimum margin while a large c will not neglect outliers and give a much smaller margin. Hence choosing a low c is best. In our test, different c produces the same accuracy for a particular PCA dimension. Moreover, it also depends on how the data would look like. And the accuracy will change based on what type of dataset we have.

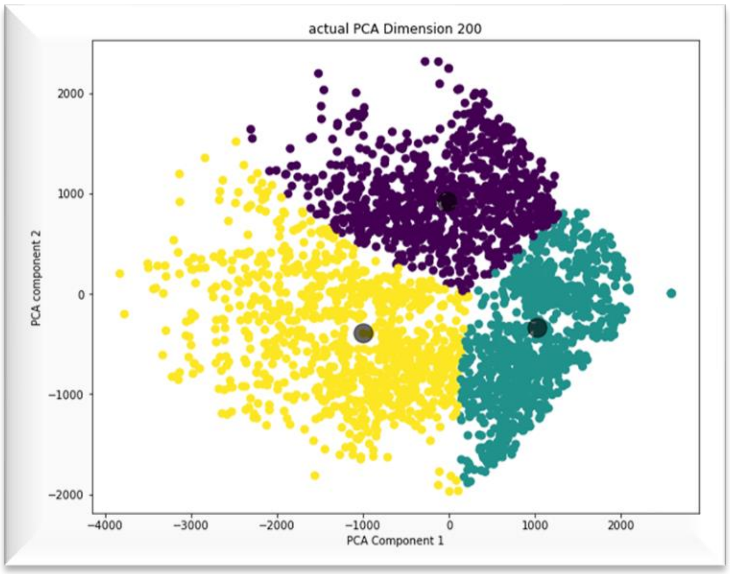
In my conclusion, the SVM works very well for the given data set as a classifier. With each iteration, the accuracy pattern can be seen similar.

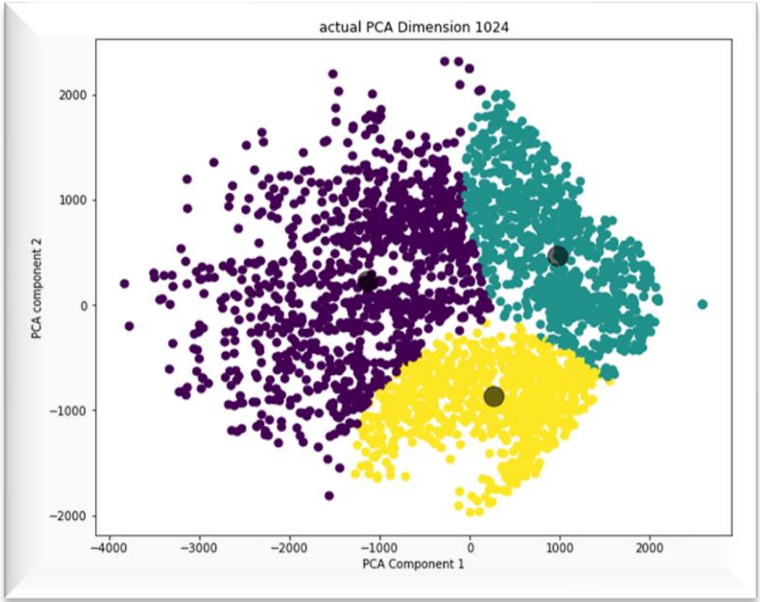
Gaussian Mixture Model:

Visualize the clustering results from GMM

Library Used: GaussianMixtrure from Sklearn.mixture library

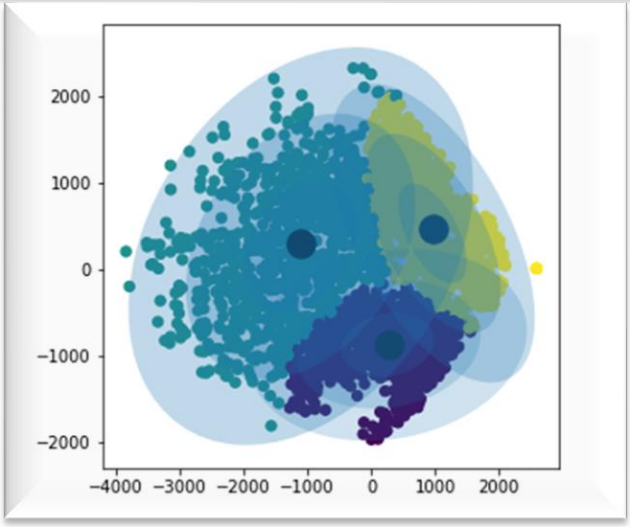






After applying GMM on 80, 200 PCA reduced dataset and the raw data, it can be seen that the data belonging to a certain contour is differing from one plot to other. But the difference is not so much. The last plot is showing the probability density plot for the three clusters.

GMM contour plot:



CNN classification results with different network architectures

Architechture:

Library Used: **Tensorflow Keras**

Layers Used: Conv2D, MaxPool2D and Dense

Layer Description:

Layer 1. First convolution layer has 20 filters, stride dimension: (1,1), kernel size: (5,5) , and input shape: (32,32,1)
Layer 2. Maxpool layer has pool size: (2,2), strides: (2,2)
Layer 3. Convolution layer has 50 filters, stride size: (1,1), kernel size(5,5)
Layer 4. Maxpool layer has pool size: (2,2), strides: (2,2)
Layer 5. Flatten Layer
Layer 6. Fully connected layer has 500 filters and activation function as ‘relu’
Layer 7. Output layer has 26 classes (25 CMU PIE faces and 1 selfie) and activation function as ‘softmax’

Hyper parameter Used:

- Optimizer: Adam optimizer
- Loss function: Sparse Categorical Cross entropy
- Epoch: 5
- Batch size: default
- Using above architecture, the below result is received:
- Learning rate: Default

Accuracy: Test Set Accuracy: 97.098%

Selfie Image Accuracy: 66.667%

With l2 regularization on the first Dense Layer**:**

Accuracy: Test Set Accuracy: 96.549%

Selfie Image Accuracy: 66.667%

This accuracy pattern is similar for different dataset.