EE5907 Pattern Recognition

CA2 Face Recognition Programming

Project Report

Zhou Ying - A0229575R

8 Nov. 2021

Department of Electrical Engineering

1. Data Prepressing

1. Data Set

The project uses the CMU PIE dataset which contains 68 different subjects representing 68 different people's face.

For the training and testing data, I choose subjects named from 1 to 25 from the CMU PIE dataset and label them by their file name. And take 10 selfie photos as class 26.

Convert the self-images to grayscale and resize them into same resolution, which is 32 x 32 pixels as the images in CMU PIE.

Figure 1 Processed Images

All the images data will be converted to a 1024-dimensional vector and form a dataset.

2. Spilt Data Set

Randomly select 70% of the image-data as the training set and the remaining 30% as the testing set. And same to their corresponding labels.

The rough description of train_PIE_images can be obtained in figure2

	0	1	2	3	4	5	6	7	8	9	...	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023
1	74	63	86	84	85	99	81	75	82	71	...	23	28	32	25	15	10	8	9	8	7
1	14	10	23	65	75	101	81	76	77	59	...	17	20	18	18	18	17	17	17	16	16
1	14	24	28	28	34	44	43	33	39	39	...	39	37	28	36	49	45	64	4	16	5
1	88	38	16	22	21	24	28	16	21	20	...	174	167	156	154	151	147	142	133	118	105
1	9	11	14	15	14	14	15	17	11	23	...	175	179	152	150	134	155	174	3	13	32
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25	81	115	111	92	123	145	135	149	146	158	...	8	8	11	34	71	61	42	50	46	42
25	119	91	58	63	53	76	70	64	67	59	...	56	60	67	52	65	55	59	74	43	17
25	63	77	69	56	74	100	106	101	123	127	...	44	38	29	27	24	22	21	20	19	18
25	117	128	129	146	155	159	185	197	186	197	...	78	67	96	152	20	35	9	24	38	7
25	51	49	56	65	67	64	71	84	85	94	...	163	166	168	231	23	26	17	9	40	23

2975 rows × 1024 columns

Figure 2 Data Description

The number of examples in training set and test set data are showed in figure3.

```
Number of train PIE images: 2975
Number of test PIE images: 1275
Number of train self images: 7
Number of test self images: 3
```

Figure 3 Data Set

2. PCA for Feature Extraction, Visualization and Classification

2.1 PCA based data distribution visualization

Randomly choose 500 samples from the PIE training data set and add 7 self-images to the data set.

Import PCA package from “sklearn.decomposition” to reduce the dimensionality of vectorized images to 2 and 3 respectively.

Visualize the projected data vector in 2D and 3D figures. The outcomes are as follows.

The projected points corresponding to selfie photos are the reddest points, i.e. “Class 26” and be highlighted by a “star” background.

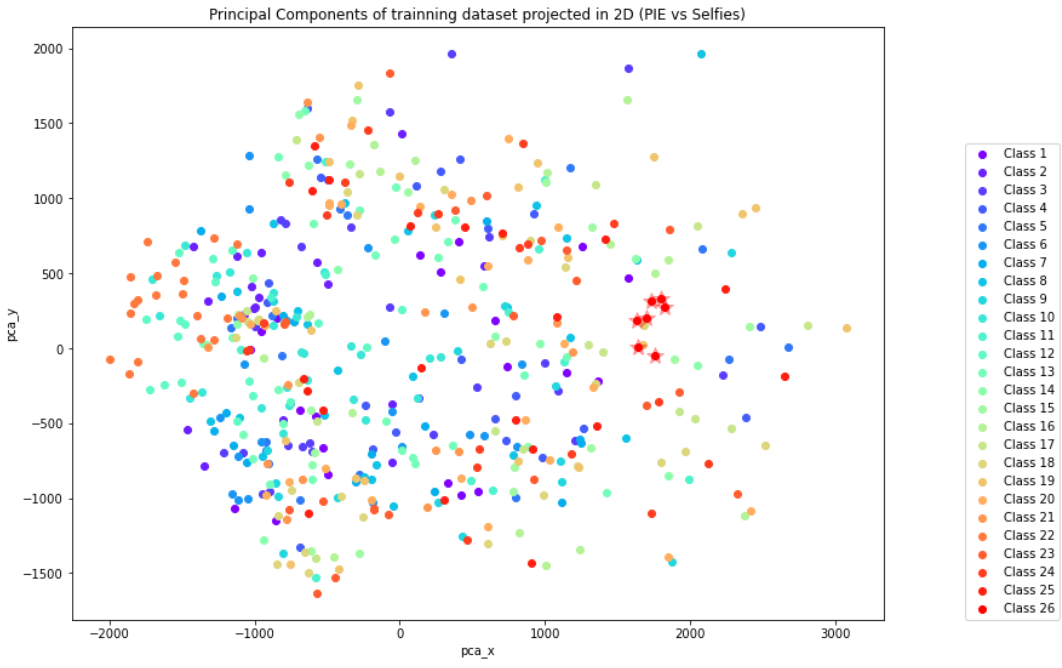

Figure 4 2D Scatter Plot

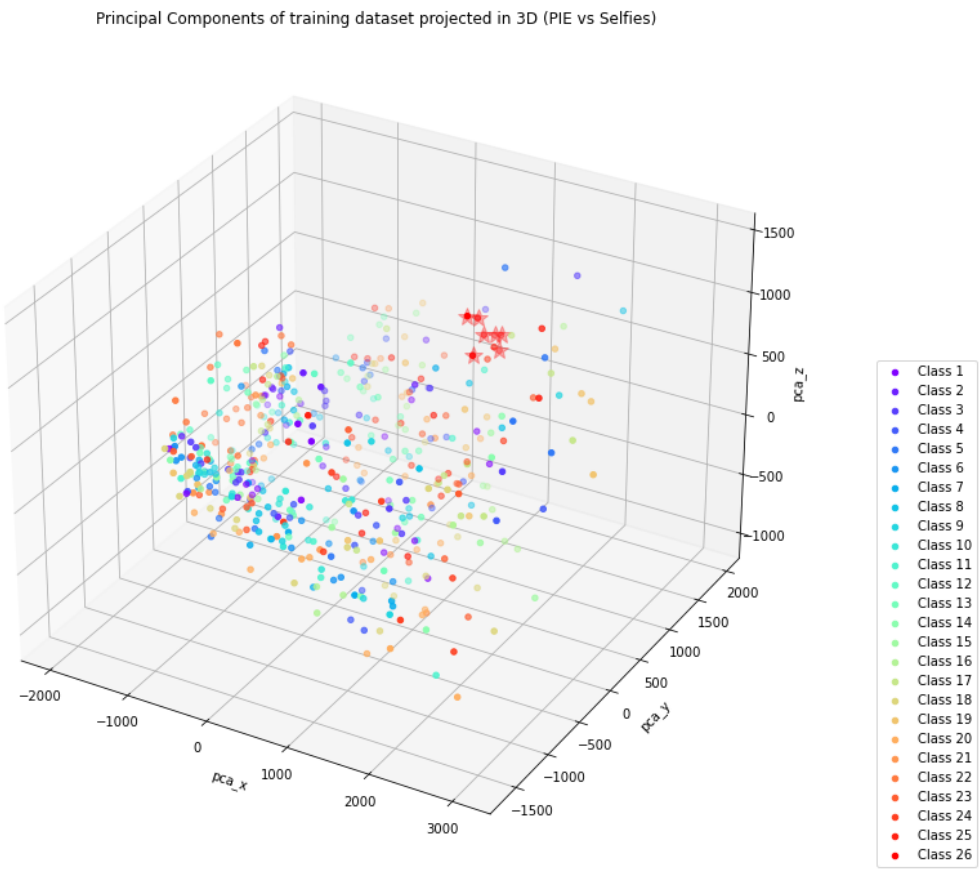

Figure 5 3D Scatter Plot

It can be seen that the 25 classes PIE images in 2D and 3D plots are classified, but it seems hard to divide them separately. However, separate selfie images from all seems easier because there is a clustering among the selfie images data.

2.2 Eigenfaces of the 3D PCA model

Visualize the corresponding 3 eigenfaces used for the dimensionality reduction.

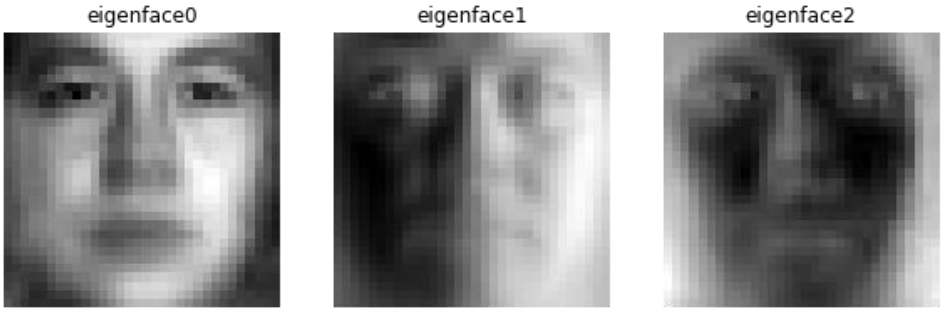

Figure 6 Eigenfaces

2.3 PCA Classification

Apply PCA to reduce dimensionality of face images to 40, 80 and 200 respectively and classify the test images using the rule of KNN.

The result of classification accuracy is shown in table 1.

Dimension	CMU PIE	Selfie
40	0.7803921568627451	1.0
80	0.8219607843137255	1.0
200	0.8454901960784313	1.0

Table 1 Classification Accuracy

From the classification results above, the greater the dimensions (PCA components) of images are, the better the accuracy of classification. But for the selfie images, there seems no distinction between different PCA components. I think it may because the dataset is too small to has pervasiveness.

3. LDA for Feature Extraction, Visualization and Classification

3.1 LDA based data distribution visualization

Similar with PCA above, apply LDA to dataset to reduce the dimensionality of data set to 2 and 3 dimensions respectively.

Visualize the projected data vector in 2D and 3D figures. The figures are showed in figure 6 and figure 7.

The projected points corresponding to selfie photos are the reddest points. i.e. “Class 26”, which are very conspicuous and separately from the PIE images data in figure 6, but the PIE data set is much narrow to distinguish different classes.

In figure 7, the PIE images clustering phenomenon is much clear while the selfie images data is still conspicuous.

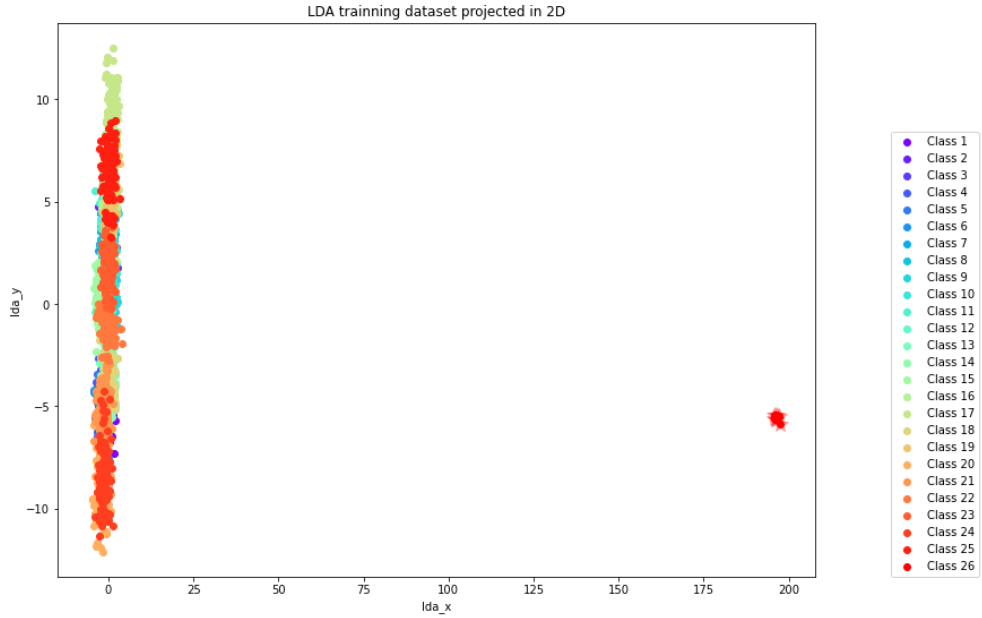

Figure 7 2D Scatter Plot

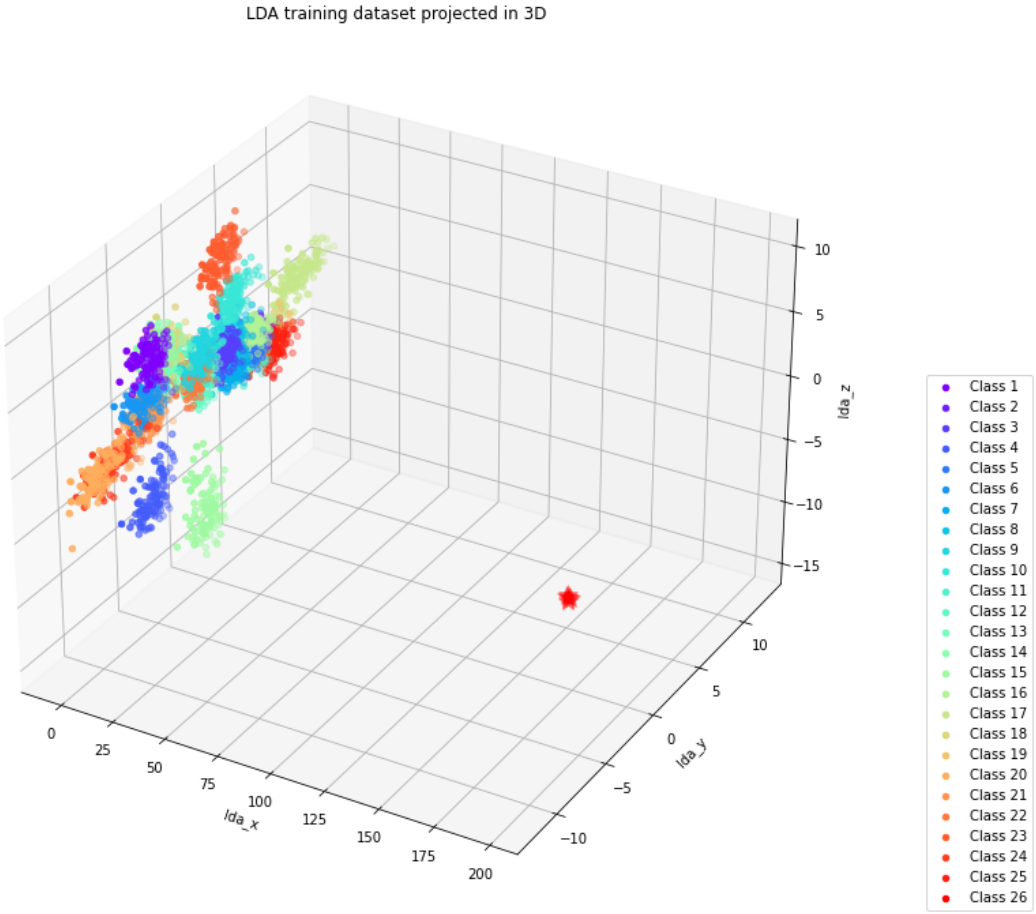

Figure 8 3D Scatter Plot

3.2 LDA Classification

Apply LDA to reduce data dimensionality from to 2, 3 and 9 and use KNN to classify the PIE test images and selfie test images separately.

The test accuracy results are shown in table 2.

Dimension	PIE	selfie
2	0.2384313725490196	0
3	0.44	0
9	0.9215686274509803	0

Table 2 Classification Accuracy

The classification accuracy results of selfie images are all zero, which is quite different from what the 2D and 3D plots show.

The reason may be that the test selfie image data is much smaller than training data.

When we increased the number of test data, the accuracy increased.

```
The accuracy of LDA with 2 components plus KNN classification on all
the test images is: 0.2378716744913928
The accuracy of LDA with 3 components plus KNN classification on all
the test images is: 0.43896713615023475
The accuracy of LDA with 9 components plus KNN classification on all
the test images is: 0.9194053208137715
```

Besides, from Google and “Stack overflow”, we know it may because the training data is much larger than selfie testing data (only three examples in testing data set while the training data has 2982.). Therefore, we changed the scale of training dataset to 507 samples (select data used in PCA plots). Then, get the accuracy result below.

```
The accuracy of LDA with 2 components plus KNN classification on the
SELFIE test images is: 1.0
The accuracy of LDA with 3 components plus KNN classification on the
SELFIE test images is: 1.0
The accuracy of LDA with 9 components plus KNN classification on the
SELFIE test images is: 1.0
```

4. Gaussian Mixture Model (GMM)

4.1 Use raw face images as training data

The raw face images data set is in size of (2982,1024), and the model is a 3-Gaussian components GMM model.

The GMM classifier will cluster and separate the data set into three parts and label each part with 0,1 or 2.

As the raw data has a large number of PCs, we only choose the first and the second eigenvalues, and visualizing them in a 2D spaces scatterplots in Figures 9.

Besides, there is a number label representing their classes on each scatter.

colors = {0: 'blue', 1: 'orange', 2: 'green'}.

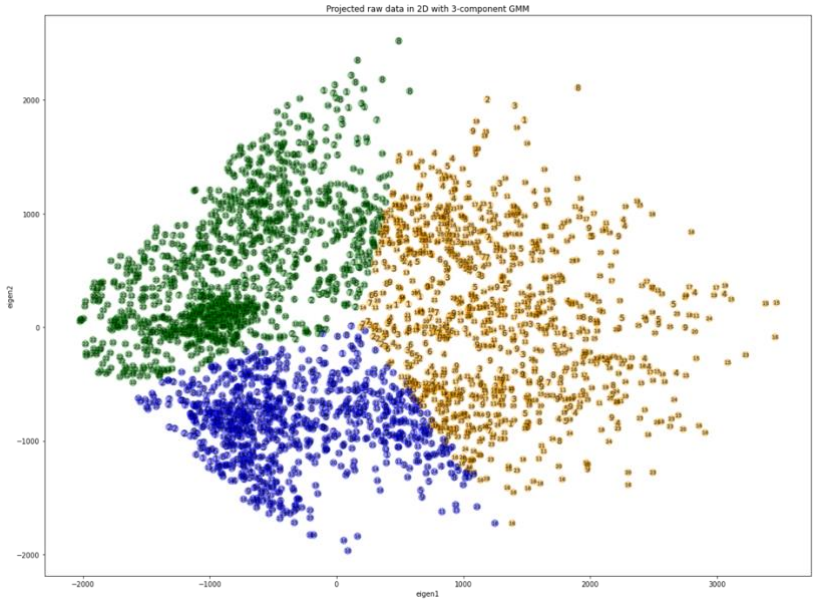

Figure 9 Projected of raw data with 3-Components GMM

4.2 Use face vectors after PCA pre-processing with dimensionality of 200

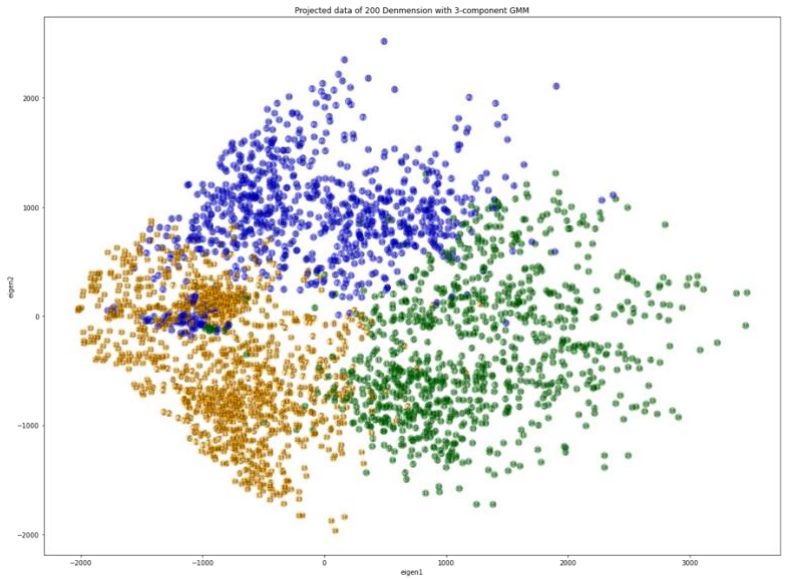

Figure 10 Projected of 80-Dimension PCA data with 3-Components GMM

4.3 Use face vectors after PCA pre-processing with dimensionality of 80

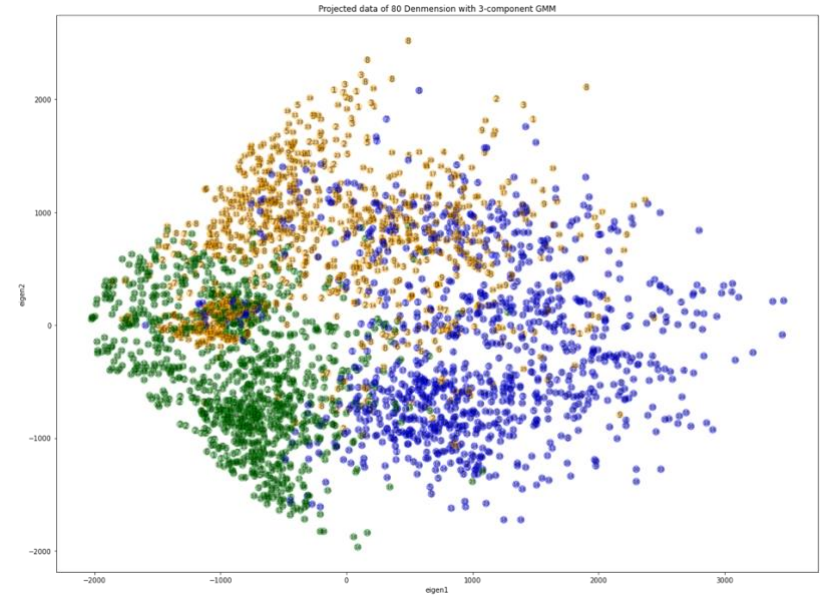

Figure 11 Projected of 80-Dimension PCA data with 3-Components GMM

Form the figures above, it seems that raw data has a better perform in GMM classifier. After deducted the dimension of data to 200-dimension or smaller 80-dimension through PCA, the clustering phenomenon has some cross and overlap in the margin.

The reason may be that when the dimension increase, the covariance matrix of one point to the component will become smaller, and effect the importance of different dimension, then this data point will be clustered to the component.

Besides, it may because the choose of components numbers that equals to 3 is small.

5. Support Vector Model (SVM)

5.1 SVM Model

From sklearn.svm import LinearSVC package and train the model.

Set the penalty parameters C = 0.01, 0.1 and 1.0, after training the linear SVM using training data, we can get the accuracy of the classifier applying PIE testing data set and selfie testing data set.

The accuracy results are shown in table 3

5.2 Accuracy Analysis

The accuracy results are shown in table 3

From the table, we can see the accuracy increased after applying PCA, but when the dimension deducted to 80, the performance of classifier becomes not much well.

For the penalty parameters, when C = 0.1, the accuracy is largest. As we know, for soft linear SVM, when C increase, it becomes more like hard margin SVM, who is harder to tolerate mistakes that may cause overfitting.

	C = 0.01	C = 0.1	C = 1.0
raw data	0.9796557120500783	0.9796557120500783	0.9773082942097027
PCA 200-component	0.9874804381846636	0.9906103286384976	0.986697965571205
PCA 80-component	0.9624413145539906	0.9671361502347418	0.9694835680751174

Table 3

6. Convolutional Neural Network (CNN)

6.1 CNN Model Building

In this part, I used TensorFlow framework and Keras for easier model building. Keras is a deep learning framework based on TensorFlow which is much easier for CNN building.

I build a CNN model with 2 convolution layers, 2 pooling layers, 1 fully connected layer with 500 nodes and 1 output layer with 26 notes, i.e. 26 classes.

The data format used in TensorFlow is: tf-format/channels-last-format, and in this experiment, it is (32, 32, 1).

The CNN model is shown in figure 12.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 20)        520

 max_pooling2d (MaxPooling2D  (None, 14, 14, 20)       0
 )

 conv2d_1 (Conv2D)           (None, 10, 10, 50)        25050

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 50)         0
 2D)

 flatten (Flatten)           (None, 1250)              0

 dense (Dense)               (None, 500)               625500

 dense_1 (Dense)             (None, 26)                13026

=================================================================
Total params: 664,096
Trainable params: 664,096
Non-trainable params: 0
_________________________________________________________________
```

Figure 12 CNN Model

When run the code at the first time, the accuracy might be extremely low (<20%) and loss doesn't reduce over epochs, suggesting that there are tendencies for the model to get stuck at some local minimum.

After several times try, the accuracy becomes much bigger, which means that the model needs repeat training to avoid stuck in local minimum.

After 24 epochs and three times repeat training, it get a good result.

```
Test score: 0.658921480178833
Test accuracy: 0.8552425503730774
```

6.2 Accuracy Analysis

The accuracy and loss history per epoch is shown in figure 13.

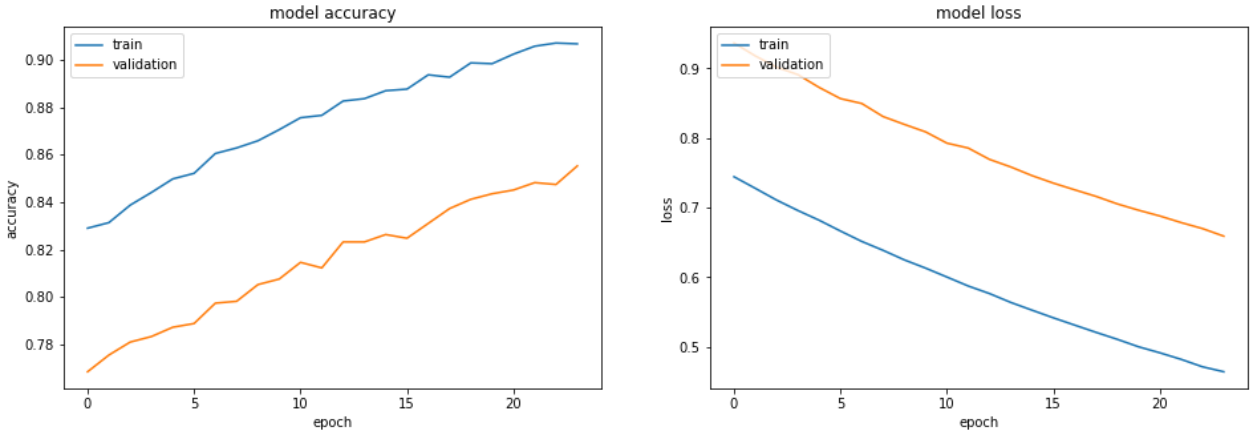

Figure 13 Training and Testing Accuracy of CNN

With the epoch increasing, the accuracy of training and testing model are both increasing, and don't get saturation point, which means the accuracy can be higher and the performance of the model can be better with more times of repeat training.

7. Conclusion

Among the 5 classification ways above, I think svm has the best performance with the accuracy all higher than 95%. And for some classification and clustering ways, PCA and LDA seem doesn't get much effect, GMM for example. But for svm, PCA do some help with proper dimension.