National University of Singapore
Electrical & Computer Engineering

EE5907 Pattern Recognition
CA2

Chen Kean
A0229502L

Date: 30th OCT 2021

1. Introduction

In this assignment, we should construct a face recognition system. We are required to apply Principal Component Analysis (PCA) to perform data dimensionality reduction and visualization in order to understand underlying data distribution. Also, we are required to train and apply three classification models, which are Linear Discriminative Analysis (LDA), Support Vector Machine (SVM) and Convolutional Neural Network (CNN) to classify the face images.

Here, the programming language I choose is MATLAB. This project is conducted on the CMU PIE dataset and the face photos taken by myself which includes 10 images. There are in total 68 different subject and I choose 25 out of them randomly for PIE dataset. For each chosen subject, I use 70% of the provided images for training and use the remaining 30% for testing. Also, my 10 selfie photos are converted to grayscale images and resized into the same resolution as the CMU PIE images. And I split them into 7 for training and 3 for testing.

2. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is explored in this assignment for compact feature extraction based on data variations. Firstly, we should make the training and testing images down to lower dimension feature spaces and perform classification with nearest neighbor classifier.

In order to implement PCA, we should calculate the covariance matrix of data matrix. Firstly, the centered data matrix $X_{d,n}$ is shown below:

$$X_{d,n} = [(x_1 - \bar{x})(x_2 - \bar{x}) \cdots (x_n - \bar{x})] \tag{2.1}$$

Next, with centered data matrix, we can compute its SVD, whose D is a diagonal matrix, U and V are orthogonal matrices.

$$X = U_{d,d} D_{d,n} (V_{n,n})^T \tag{2.2}$$

Then we can extract the eigenvector out of the orthogonal matrices. By this way, we can construct the low-dim space especially for PCA

$$x_i = \bar{x} + U_{d,p} {U_{d,p}}^T (x_i - \bar{x}) \tag{2.3}$$

$$U_{d,p} = U(:, 1:p) \tag{2.4}$$

We can obtain the eigenvectors of covariance matrix corresponding to the N features and select the largest eigenvalue by this way. And the dimensional reduction of data features is successfully implemented.

2.1 PCA based data distribution visualization

In this part, I will show the visibility of PCA. Here, we should show the 2D, 3D plot and 3 eigenfaces based on the requirement for this CA. In the figures shown below, the yellow points are belonging to my own selfie photos and I implement PCA to reduce the dimension from 1024 to 2 and 3 respectively. You can see from the visualization that points for my selfie photos are closed to each other, so it is pretty easy for us to distinguish them from others.

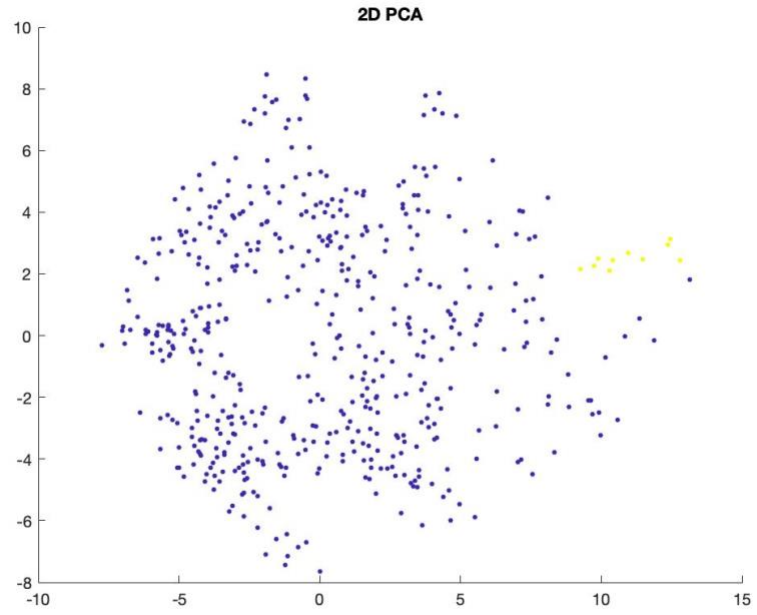

Fig 2.1 Visualize training set data vectors in 2D for PCA

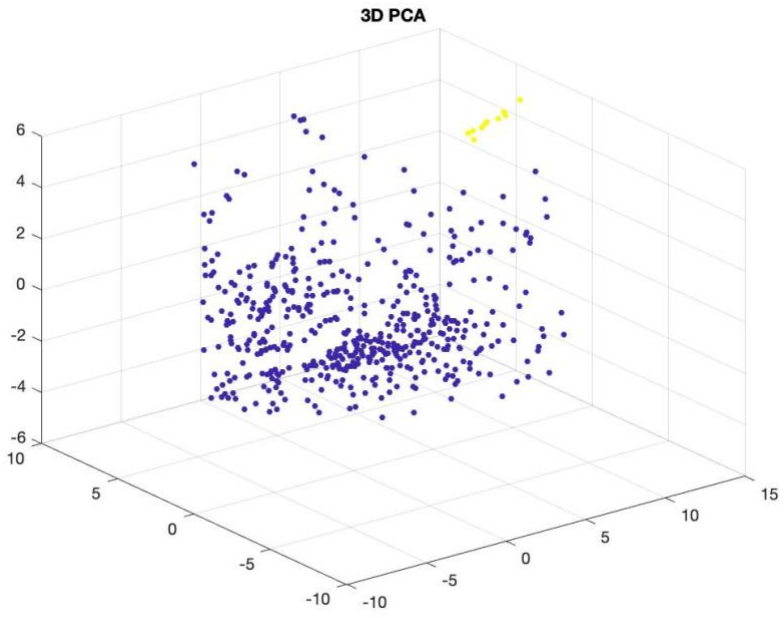

Fig 2.2 Visualize training set data vectors in 3D for PCA

In the next part, I will show 3 eigenfaces as requirement for this CA.

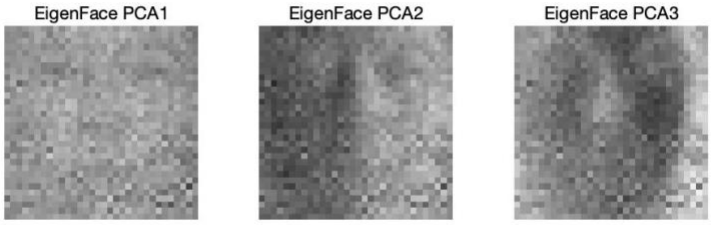

Fig 2.3 Visualize top 3 eigenfaces

2.2 PCA plus nearest neighbor classification results

Nearest neighbor classifier is implemented to classify the test images and output classification accuracy. Here we should compute the Euclidean distance between each training and testing dataset and we should consider the smallest distance as the belonging class. The accuracy when we apply PCA to reduce the dimensionality is shown below Table 2.1.

	40	80	200
Accuracy of PIE	90.745%	91.765%	92.784%
Accuracy of Self	100%	100%	100%

Table 2.1 Accuracy for reducing the dimensionality with PCA

From Table 2.1, we can know that the accuracy for selfie images is 100%. I think the reason why this phenomenon happens is that every selfie images are not different between each other. Also in Fig 2.1 and 2.2, all the yellow points represented for my selfie images are very closed to each other. Therefore, it is reasonable to achieve that high accuracy.

For PIE dataset, the overall accuracy is more than 90%. The accuracy becomes higher when the dimensions are increasing, which can be seen from Table 2.1. Also, when we have larger number of training datasets, the accuracy becomes higher.

In conclusion, we can conclude that PCA with nearest neighbor classifier can classify the faces very well is we have enough training set to put them into the algorism.

3. Linear Discriminative Analysis (LDA)

Linear Discriminative Analysis (LDA) can find most discriminative projection by maximizing between-class distance and minimizing within-class distance. It is a supervised learning algorithm which can consider the data class membership for feature extractions. Therefore, LDA perform better than PCA for feature classification.

In order to implement LDA, we should calculate the class-specific mean vector μ_i and covariance matrix S_i by following formulas.

$$\mu_i = \frac{1}{n_i}\sum_{x\epsilon C_i} x \tag{3.1}$$

$$S_i = \sum_{x\epsilon C_i}(x-\mu_i)(x-\mu_i)^T \tag{3.2}$$

Based on the class-specific mean vector μ_i and covariance matrix S_i above, we can find the within-class scatter matrix S_W and between-class scatter matrix S_B by formulas below.

$$S_W = \sum_{i=1}^{C}\frac{n_i}{N}S_i = \sum_{i=1}^{C}P_i S_i \tag{3.3}$$

$$S_B = \sum_{i=1}^{C}P_i\,(\mu-\mu_i)(\mu-\mu_i)^T \tag{3.4}$$

Finally, we can generate the eigenvalue and use SVD to find the eigenvector.

$$[U] = SVD({S_W}^{-1}S_B) \tag{3.5}$$

From the calculating process we can tell the differences between PCA and LDA. The LDA explicitly attempts to model the difference between the classes of data, while PCA cannot tell any differences between classes.

3.1 LDA based data distribution visualization

In this part, I will show the visibility of LDA. Here, we should show the 2D, 3D plot based on the requirement for this CA. In the figures shown below, the yellow points are belonging to my own selfie photos and I implement LDA to reduce the dimension from 1024 to 2 and 3 respectively. You can see from the visualization that the yellow points is very far away from blue points. So I think my selfie images can be distinguished easily from the whole dataset and accuracy of my selfie images can be very high.

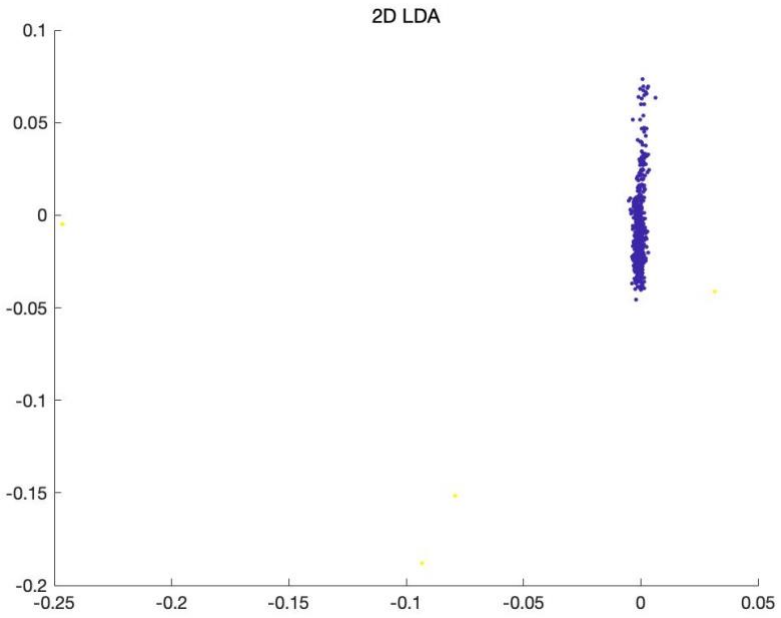

Fig 3.1 Visualize training set data vectors in 2D for LDA

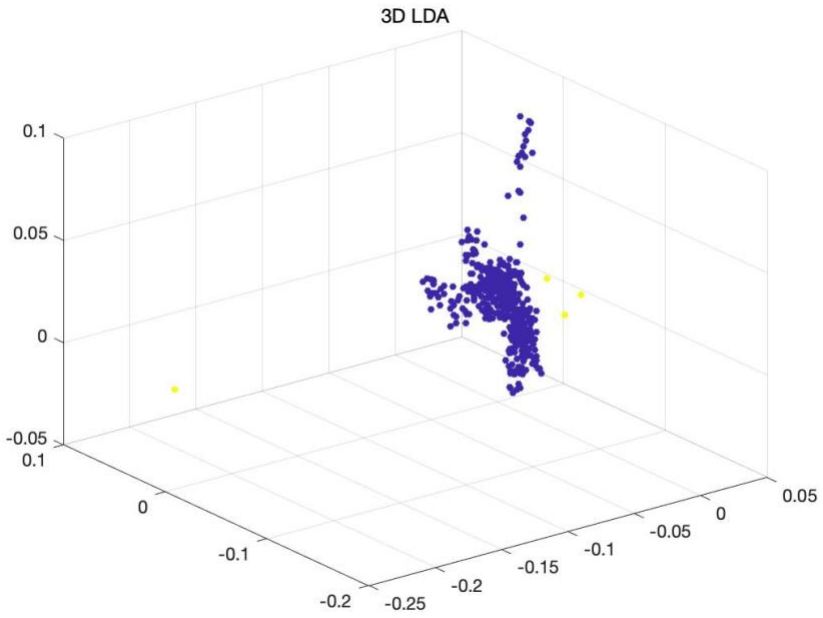

Fig 3.2 Visualize training set data vectors in 3D for LDA

And we can see the difference between PCA and LDA clearly that LDA can cluster the points more perfectly than PCA. But we cannot just judge that LDA is better than PCA. I think the performance between these 2 methods is based on the type of data and task requirement.

3.2 LDA plus nearest neighbor classification results

Nearest neighbor classifier is implemented to classify the test images and output classification accuracy. Here we should compute the Euclidean distance between each training and testing dataset and we should consider the smallest distance as the belonging class. The accuracy when we apply LDA to reduce the dimensionality is shown below Table 3.1.

	2	3	9
Accuracy of PIE	25.0196%	48.0000%	96.9412%
Accuracy of Self	100%	100%	100%

Table 3.1 Accuracy for reducing the dimensionality with LDA

From Table 3.1, we can know that the accuracy for selfie images is 100% which is similar to PCA. With more dimension, we can classify PIE images with higher accuracy. Comparing with PCA, LDA have less dimension but the accuracy between 2 methods is pretty close. So for some special tasks, LDA have more advantage of classification because of the difference of algorism.

4. GMM

In this part, the assignment requires me to visualize the test result of GMM with 3 kinds of training data, which are raw face images data and 2 set of pre-processed data by PCA.

4.1 GMM visualization

The Fig 4.1, 4.2 and 4.3 show the clustering results based on raw face image data, data processed by PCA with dimension of 80 and data processed by PCA with dimension of 200 respectively.

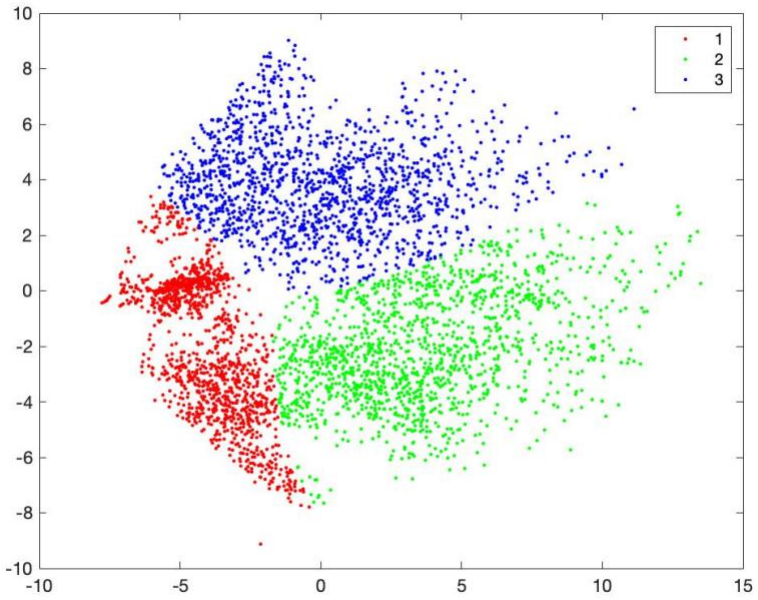

Fig 4.1 The clustering result of the raw data

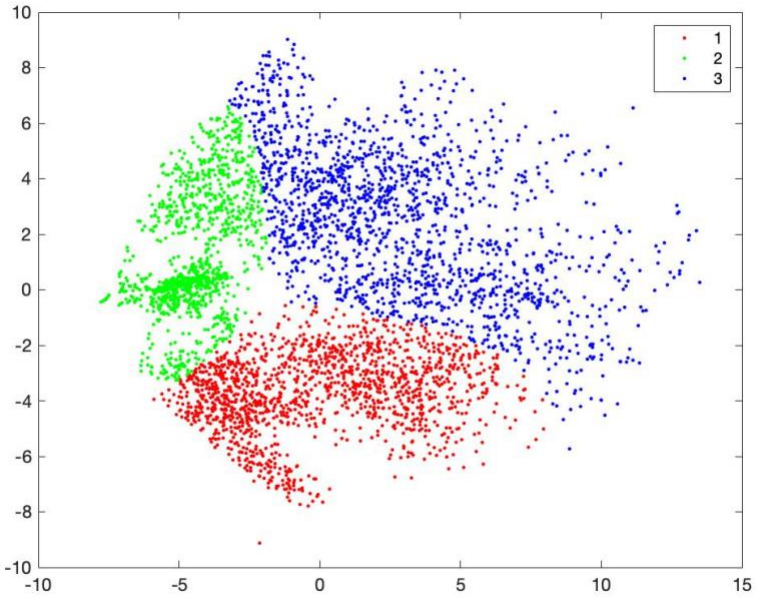

Fig 4.2 The clustering result of the data with 80 dimensionalities

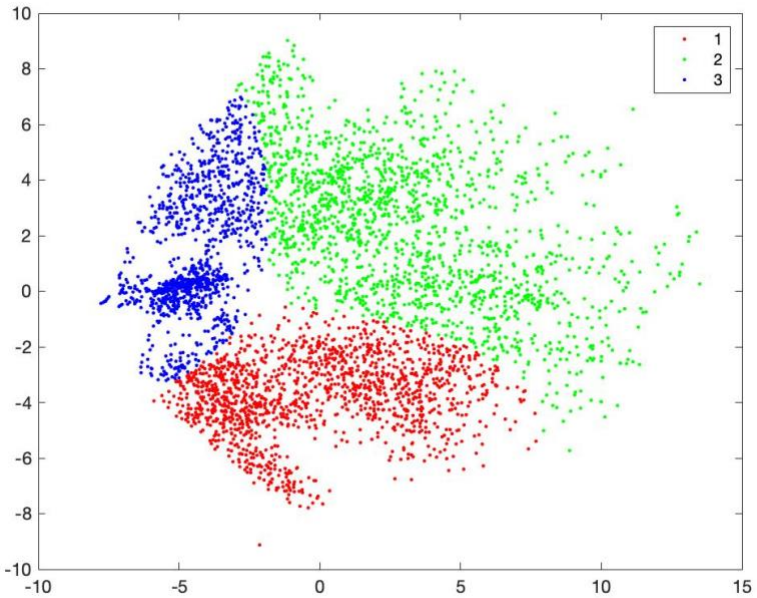

Fig 4.3 The clustering result of the data with 200 dimensionalities

In each figure, different color represents different cluster. With different training data, the results of them are not entirely different. Also, you can see that all these 3 results of clustering are all very good.

5. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised classifier, which can increase the margin between different classes and can increase the width between the boundary. In this case, the margin can guarantee the optimal classification results.

SVM can generate the optimal superplane based on edge distance, which can be seen in the formular below.

$$W_* = \min_{W,b} \frac{1}{2} W^T W \tag{5.1}$$

We can consider it as Quadratic Program (QP), for which efficient global solution algorism exists. If the samples are linearly separable, We can use the Penalty parameter to avoid the misclassification.

$$W_* = \min_{W,b} \frac{1}{2} W^T W + C \sum_{i=1}^{N} \epsilon_i \tag{5.2}$$

$$y_i(W^T x_i + b) \geq 1 - \epsilon_i \tag{5.3}$$

5.1 SVM classification results with different parameter values

Here, I implement SVM to classify the raw face image data, data processed by PCA with dimension of 80 and data processed by PCA with dimension of 200 respectively. Also I will try 3 different value of penalty parameter C, which are 0.01, 0.1 and 1.

First I use linear kernel function with different training sets and different penalty parameters. All the results are shown in Table 5.1.

	1	0.1	0.01
Accuracy of raw data	98.7425%	96.6462%	80.9466%
Accuracy of 80 dimension	97.9608%	96.2353%	76.3137%
Accuracy of 200 dimension	98.1176%	97.098%	79.451%

Table 5.1 The accuracy of different penalty parameter and different dimension

From above table, we can see that when the value of the penalty parameter is larger, the accuracy will be higher. But if the penalty is big enough, the problem of classification can be overfitting. So we should achieve the balance between interval size and accuracy of classification by choosing a suitable penalty parameter.

If we increase the dimensions, the accuracy for classification can be higher. Here, more dimension means we need to consider more information of classification. When reducing the dimension, the problem becomes easier but contains less information. So that is why the accuracy of raw data can be highest.

Then, we implement different kernel function by setting penalty parameter equals to 1 and make comparison between them, as seen in Table 5.2.

	RBF kernel function	Linear kernel function
Accuracy of raw data	45.3841%	98.7425%
Accuracy of 80 dimension	88.549%	97.9608%
Accuracy of 200 dimension	82.2745%	98.1176%

Table 5.2 The accuracy of different kernel function and different training sets

The results of RBF kernel function and linear kernel function is pretty different. If we want to implement RBF kernel function of SVM, we need to use PCA to reduce the dimension to increase the accuracy of classification.

6. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a class of deep, feed-forward artificial neural network and commonly applied to analyzing visual imagery.

In this report, I have applied CNN to CMU PIE dataset (25 classes) and my selfie images (1 class) with the architecture, which the number of nodes is 20-50-500-26. In this part, I use the toolbox in the MATLAB called Deep Network Designer to build it. This is a very friendly toolbox with GUI interface. The dataset arrangement can be seen in Fig 6.1.

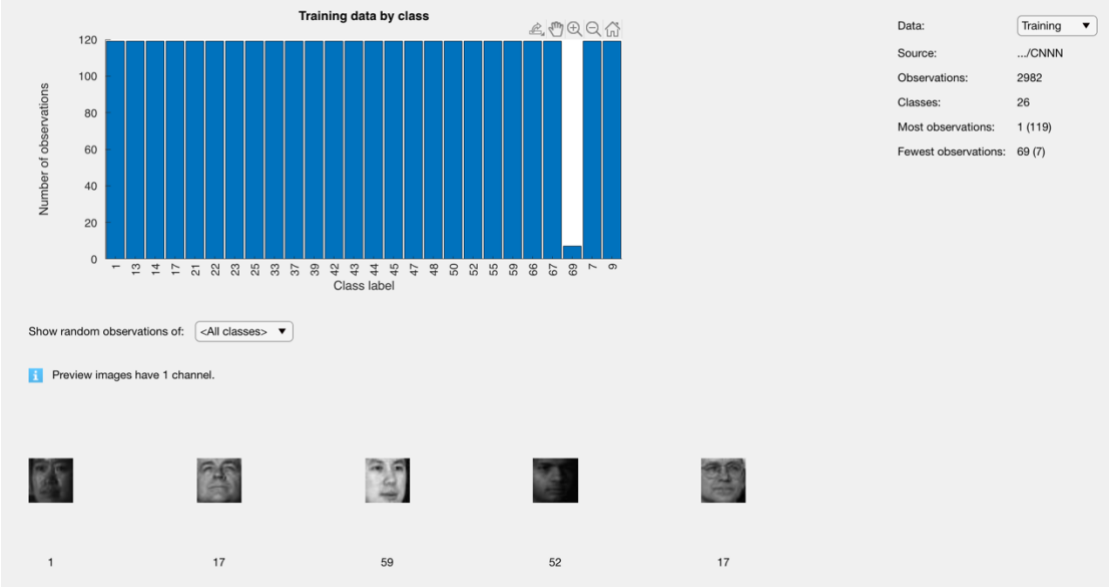

Fig 6.1 The dataset arrangement of CNN

6.1 CNN classification results with different network architectures

Here, the convolutional kernel sizes are set as 5 and each convolution layer is followed by a max pooling layer with a kernel size of 2 and stride of 2. Then the fully connected layer is followed by ReLU. The network architecture of CNN is shown in Fig 6.2. And we can see the detail of this CNN network in Deep Learning Network Analyzer, shown in Fig 6.3.

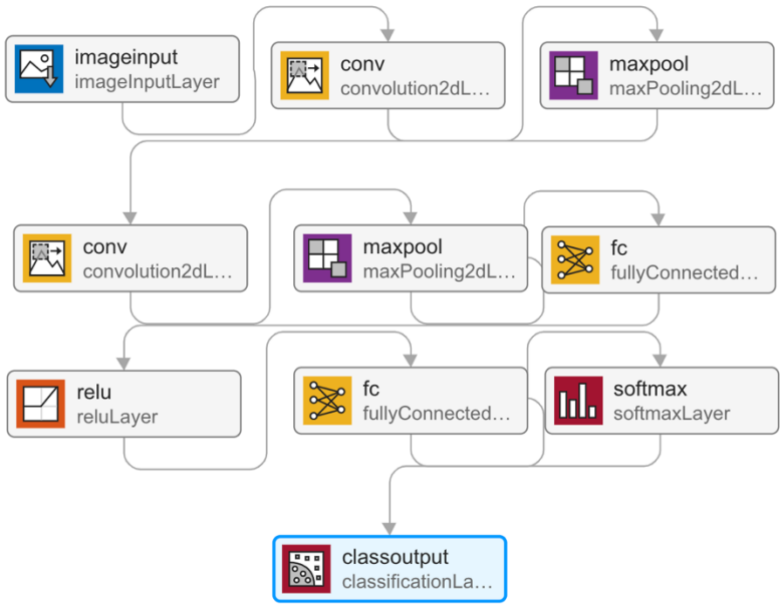

Fig 6.2 The network architecture of CNN

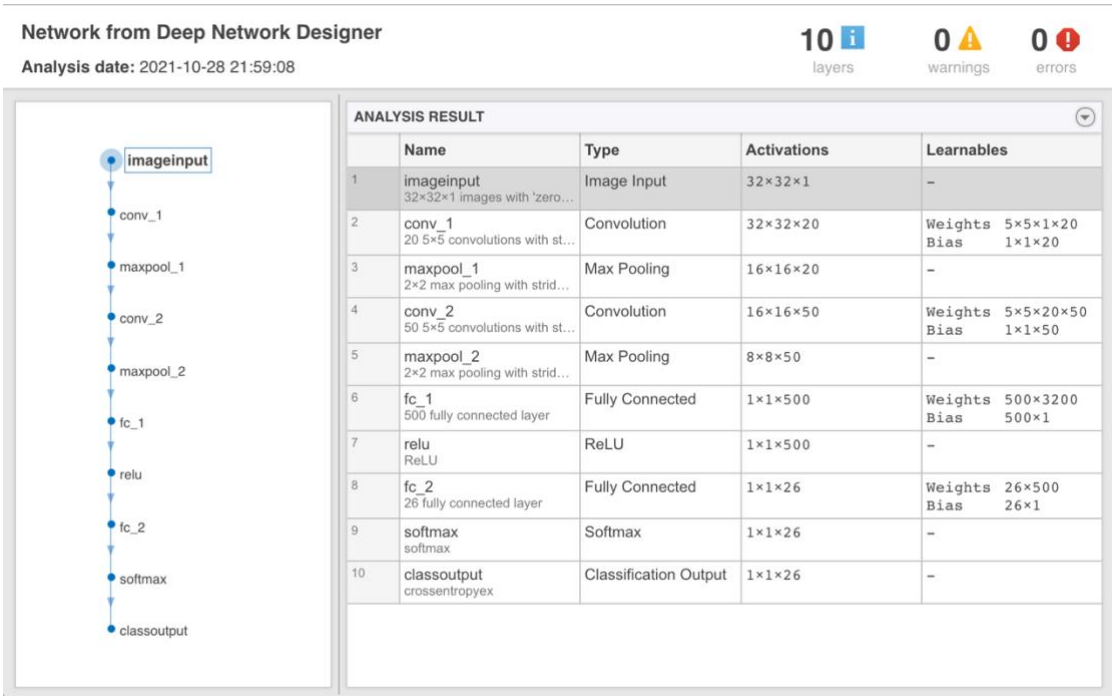

Fig 6.3 The detail of the structure

With the structure above, I start my training and finally get the accuracy of this architecture which is 88.42%. Here the learning rate is equal to 0.001. The outcome shows in Fig 6.4.

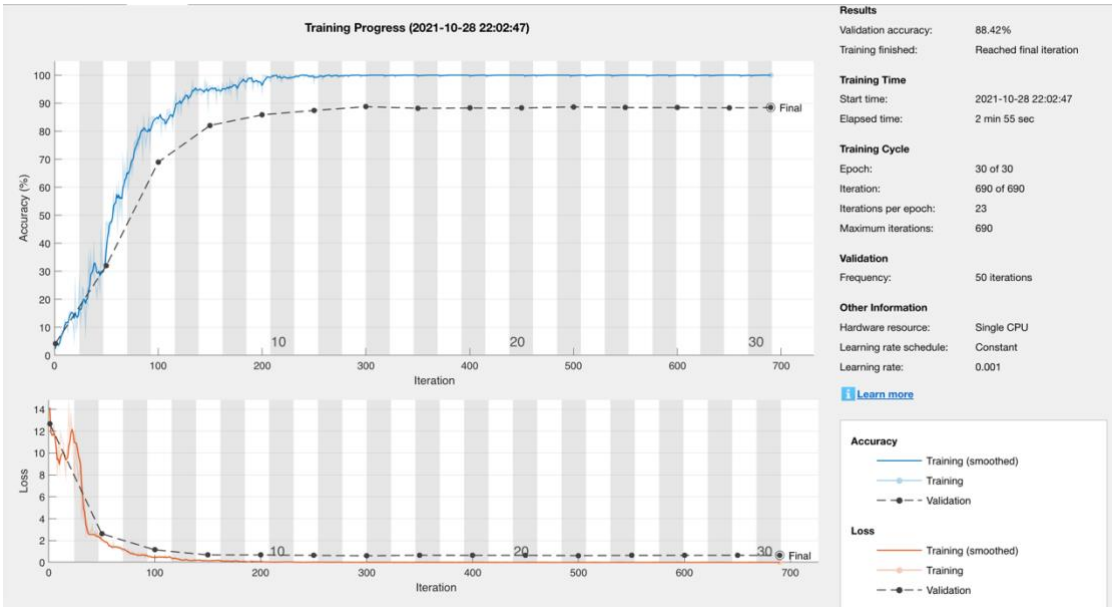

Fig 6.4 The result of the structure

Then we add padding into it and the structure is changed, as shown in Fig. 6.5.

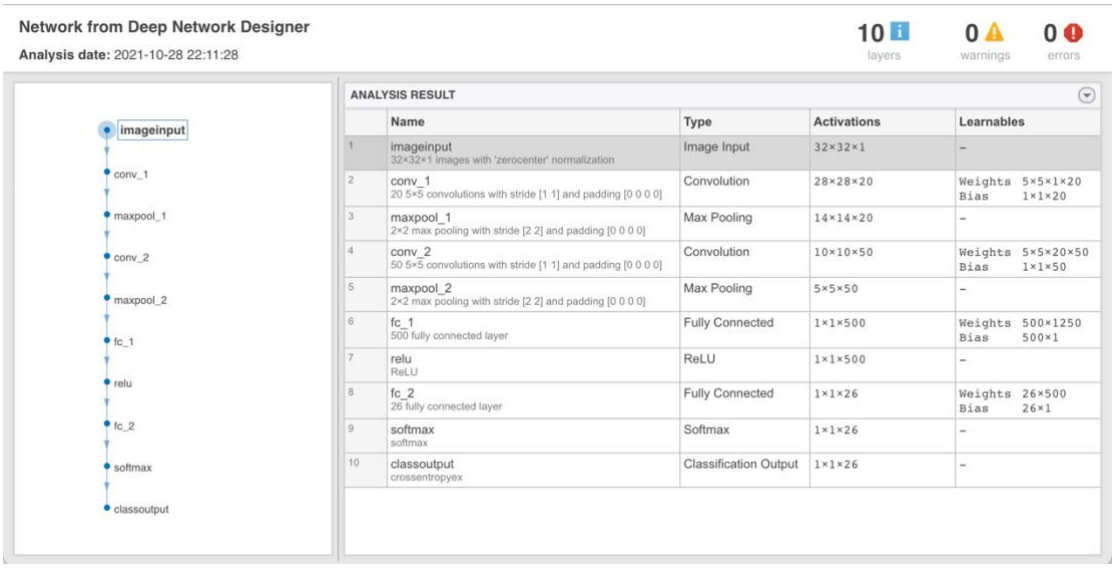

Fig 6.5 The detail of the structure with padding

With the structure above, I start my training and finally get the accuracy of this architecture which is 88.03%, which is very similar to the structure above. Here the learning rate is equal to 0.0005. The outcome shows in Fig 6.6.

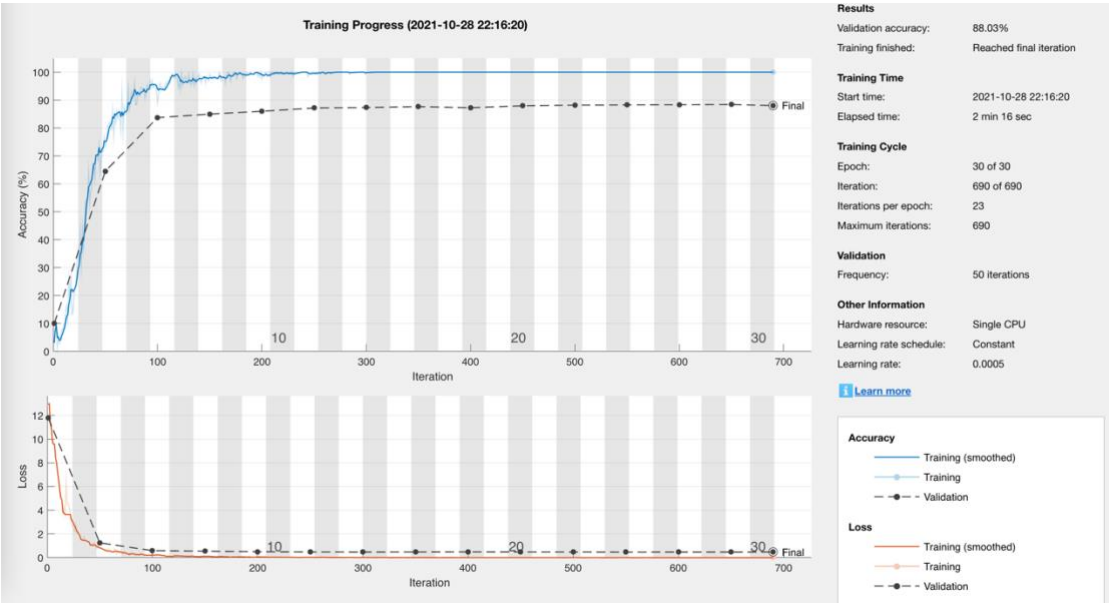

Fig 6.6 The result of the structure with padding

Finally, we can see that all the 2 structures have a good ability to distinguish the faces images of PIE and myself.