Face Recognition -- EE5907 CA2 Report

Gu Jiapan A0186538N

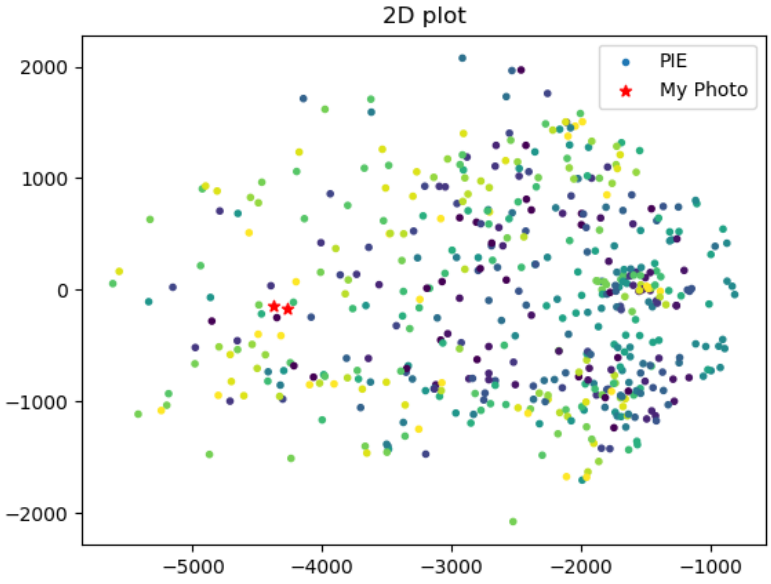
e0320776@u.nus.edu

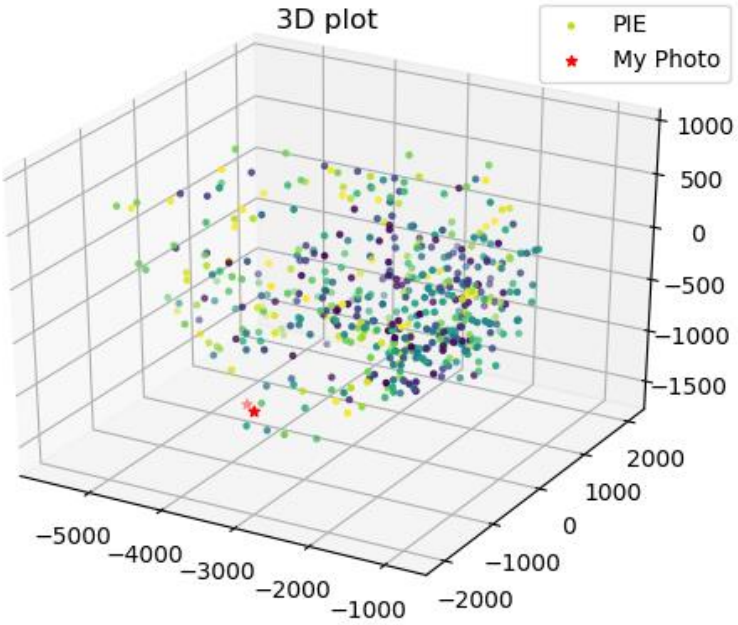
Data Indication:

I choose the folders numbered 1 to 20 from CMU PIE dataset and label them correspondingly as class 1 to class 20. In addition, I put 10 of my photos in the folder numbered 21 and label them as class 21.

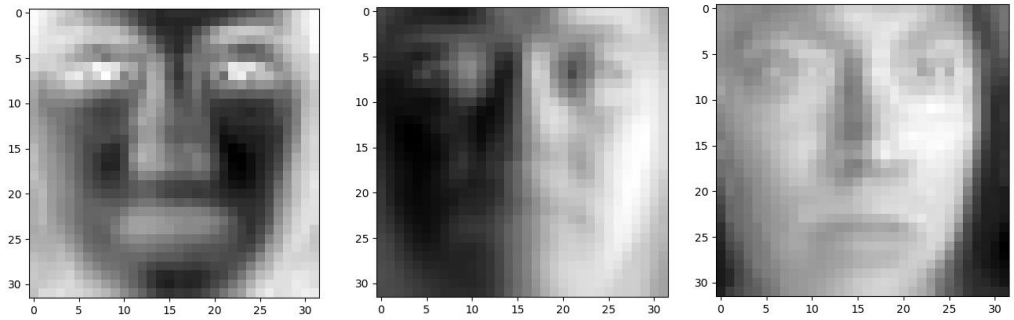
Part1: PCA

1. Visualize the projected data vector in 2d and 3d plots.





2. Visualize the corresponding 3 eigenfaces used for the dimensionality reduction.



3. Analysis on PCA based data distribution visualization.

From the 2D and 3D projection figures, we can find that all the face images are distributed within a certain range after dimensionality reduction. Since PCA is a kind of unsupervised representation learning, there is no obvious distinction between different classes in a low-dimensional space.

From the figures of eigenfaces, we can find that their distinction is obvious. These three eigenfaces showed different characteristics of the face images

4. Classification accuracy on the CMU PIE test images and my own photos.

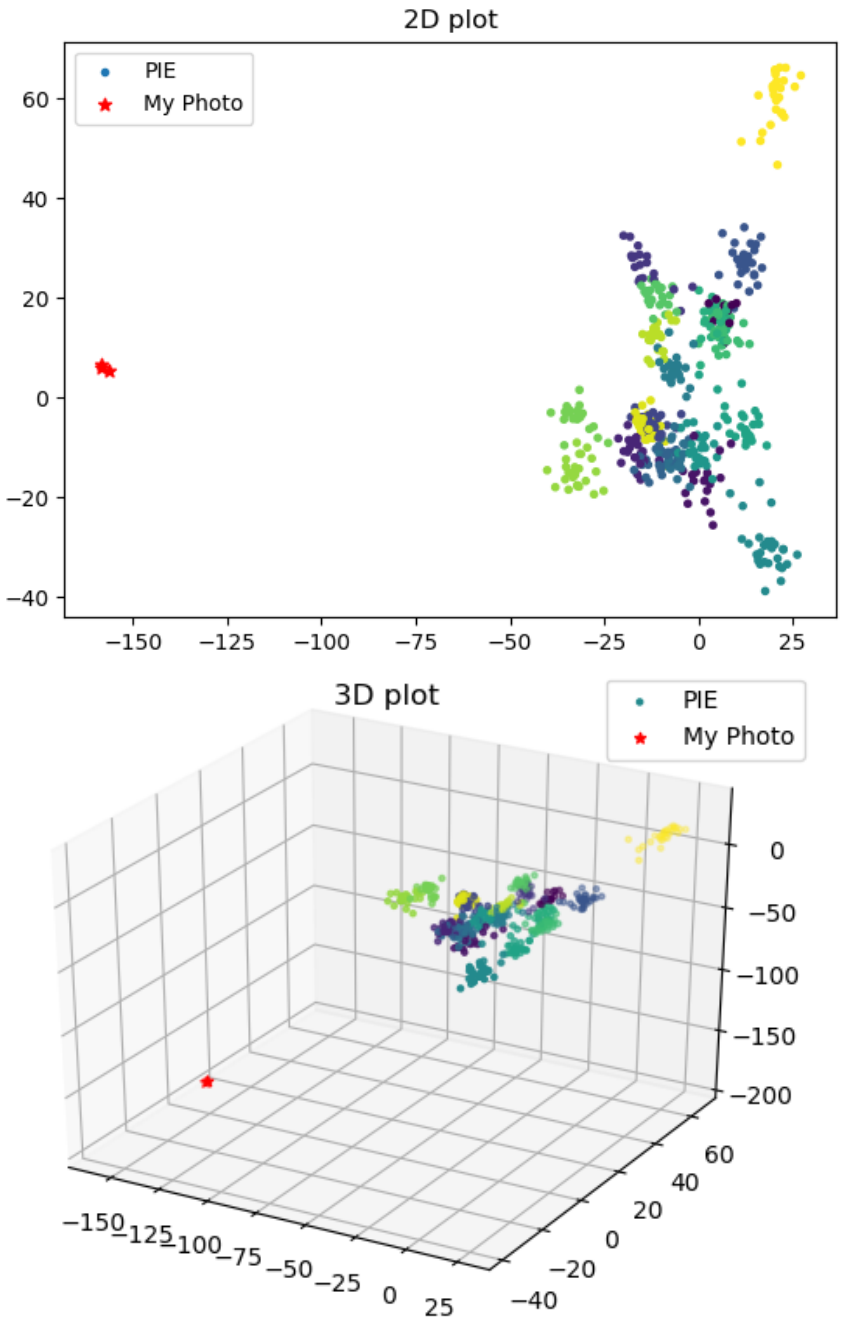
Dimensionality	40	80	200
Accuracy on PIE test images	95.29%	96.47%	96.57%
Accuracy on my own photos	100%	100%	100%

5. Analysis on PCA plus nearest neighbor classification results.

From the table shown above, we can find that all the three sets of experiments achieve good performance. With the increase of dimensionality, the classification accuracy on the PIE test images is also increases. It tells us that the higher the dimensionality, the more information the model learns, which helps the model learn and predict better. But the higher dimensionality costs more computational time. As for the accuracy on my own photos, since they are clearly different from PIE dataset, the classification accuracy can reach 100%.

Part2: LDA

1. Visualize the projected data vector in 2d and 3d plots.



2. Analysis on LDA based data distribution visualization.

From the 2D and 3D projection figures, we can find that each class of the images is clustered into a small pile. There are still overlaps among some classes. It indicates that the characteristics of each class are initially revealed in a low-dimensional space. In addition, due to the specialty of my photos, they are relatively far from all classes of PIE images.

3. Classification accuracy on the CMU PIE test images and my own photos.

Dimensionality	2	3	9
Accuracy on PIE test images	34.51%	56.76%	92.45%
Accuracy on my own photos	33.33%	33.33%	33.33%

4. Analysis on LDA plus nearest neighbor classification results.

From the table shown above, we can find that with the increase of dimensionality, the accuracy on the PIE test images increases rapidly. In a low-dimensional space, each additional dimension can greatly help improve the performance of the model. When the dimensionality reaches 9, the model has been able to achieve a decent accuracy on PIE test images. However, since the amount of my photos is very small, the accuracy on them is just 33.33% in a low-dimensional space.

Part3: SVM

1. Classification accuracy with different parameters and dimensions.

Dimensionality	80	200	Raw images
C = 0.00001	Train: 99.87% Test: 99.02%	Train: 99.96% Test: 99.12%	Train: 100% Test: 99.02%
C = 0.01	Train: 100% Test: 99.02%	Train: 100% Test: 99.32%	Train: 100% Test: 99.02%
C = 0.1	Train: 100% Test: 99.02%	Train: 100% Test: 99.32%	Train: 100% Test: 99.02%
C = 1	Train: 100% Test: 99.02%	Train: 100% Test: 99.32%	Train: 100% Test: 99.02%
C = 10000	Train: 100% Test: 99.02%	Train: 100% Test: 99.32%	Train: 100% Test: 99.02%

2. Analysis on SVM classification results with different parameter values.

From the table shown above, we can find that, since the LibSVM package is very powerful and our dataset is small, all the experiments have an excellent performance. The adjustable parameter C refers to penalty parameter. A hard-margin SVM can be approximated by a soft-margin SVM with a very large C value.

Due to the same results among C = 0.01, 0.1 and 1, I added two more sets

of experiments with C = 0.00001 and 10000. Thus, we can find that when the penalty parameter C is very small, the accuracy has a slight decrease. Combining the results with theory, we can indicate that, with the increase of penalty parameter C, the accuracy on both training set and testing set increases.

Part4: CNN

1. Classification performance by training neural network.

```
Epoch 10/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.1168 - acc: 0.9732 - val_loss: 0.1436 - val_acc: 0.9629
Epoch 11/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0943 - acc: 0.9811 - val_loss: 0.1220 - val_acc: 0.9658
Epoch 12/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0729 - acc: 0.9858 - val_loss: 0.1365 - val_acc: 0.9629
Epoch 13/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0643 - acc: 0.9870 - val_loss: 0.0968 - val_acc: 0.9736
Epoch 14/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0498 - acc: 0.9933 - val_loss: 0.1047 - val_acc: 0.9707
Epoch 15/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0442 - acc: 0.9958 - val_loss: 0.0850 - val_acc: 0.9736
Epoch 16/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0370 - acc: 0.9950 - val_loss: 0.0793 - val_acc: 0.9765
Epoch 17/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0267 - acc: 0.9971 - val_loss: 0.0841 - val_acc: 0.9736
Epoch 18/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0233 - acc: 0.9996 - val_loss: 0.0795 - val_acc: 0.9765
Epoch 19/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0203 - acc: 0.9983 - val_loss: 0.0855 - val_acc: 0.9746
Epoch 20/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0199 - acc: 0.9987 - val_loss: 0.0834 - val_acc: 0.9736
Epoch 21/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0174 - acc: 0.9983 - val_loss: 0.0821 - val_acc: 0.9765
Epoch 22/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0142 - acc: 0.9992 - val_loss: 0.0689 - val_acc: 0.9795
Epoch 23/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0106 - acc: 0.9996 - val_loss: 0.0680 - val_acc: 0.9804
Epoch 24/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0093 - acc: 1.0000 - val_loss: 0.0623 - val_acc: 0.9795
Epoch 25/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0090 - acc: 1.0000 - val_loss: 0.0698 - val_acc: 0.9804
Epoch 26/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0066 - acc: 1.0000 - val_loss: 0.0594 - val_acc: 0.9804
Epoch 27/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0056 - acc: 1.0000 - val_loss: 0.0609 - val_acc: 0.9834
Epoch 28/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0061 - acc: 1.0000 - val_loss: 0.0625 - val_acc: 0.9804
Epoch 29/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0053 - acc: 1.0000 - val_loss: 0.0685 - val_acc: 0.9785
Epoch 30/30
2387/2387 [==============================] - 3s 1ms/sample - loss: 0.0049 - acc: 1.0000 - val_loss: 0.0578 - val_acc: 0.9844
Accuracy of training set: 1.0 loss: 0.003946725987387602
Accuracy of testing set: 0.98435974 loss: 0.05784656985035623
```

2. Analysis on CNN classification performance.

The above figure shows the training record and results. Finally, the accuracy on the training set achieves 100% and the accuracy on the testing set achieves 98.44%. During the training process, the general trend for accuracy of both training set and validation set is rising. The general trend for loss value is decreasing. Thus, it shows a really good performance.

My network is based on the framework of “TensorFlow.Keras”. In order to improve the efficiency and accuracy of training, after several experiments, I made the following parameter settings: learning rate = 1e-3, epochs = 30, batch size = 256 and the optimizer is “Adam”. If the learning rate is too small, it will converge slowly and maybe get stuck in false local minima. If it is too large, it may overshoot and become unstable. Therefore, a stable learning rate is so important that the model can converge smoothly and avoid local minima.