EE5907 Face Recognition CA2

Yutong Xia
yutong.xia@u.nus.edu

Abstract

This report is coursework submitted to EE5907. The aim of this project is to construct a face recognition system via Principal Component Analysis (PCA), Linear Discriminative Analysis (LDA), Support Vector Machine (SVM) and Convolution Neural Network (CNN). PCA and LDA are used to perform data dimensionality reduction and visualization, in order to understand underlying data distribution. Then lower-dimensional data are classified based on the nearest neighbour classifier. In addition, SVM and CNN are also used to classify face images. It is found that CNN outperforms the other three approaches with the highest accuracy of 100%.

1 Dataset and Code

The dataset used in this project contains the CMU PIE dataset [1] and some selfie photos. There are in total of 68 different subjects in the CMU PIE dataset and **I selected the first 25 out of them**. For each chosen subject, 70% for training and the remaining 30% for testing. Besides the CMU PIE images, there are 10 selfie photos used as samples after being converted to grey-scale images and resized into the same resolution with provided PIE image data (i.e., 32×32). Table 1 shows the detial of the dataset. The code for this project can be found here: `https://github.com/yutong-xia/EE5907`.

Table 1: The detail of the dataset used in this project. # means 'numbers of'.

		CMU PIE dataset	Selfie Photo
# Samples	Training set	2982	7
	Testing set	1278	3
# Objects		25	1

2 Principal Component Analysis (PCA)

This section shows the experimental results of using PCA for feature extraction, visualization and classification. Then dataset used to calculate the eigenvectors is a 500-sample dataset generated by randomly sampling 493 images from the CMU PIE training set and adding 7 selfie photos[2].

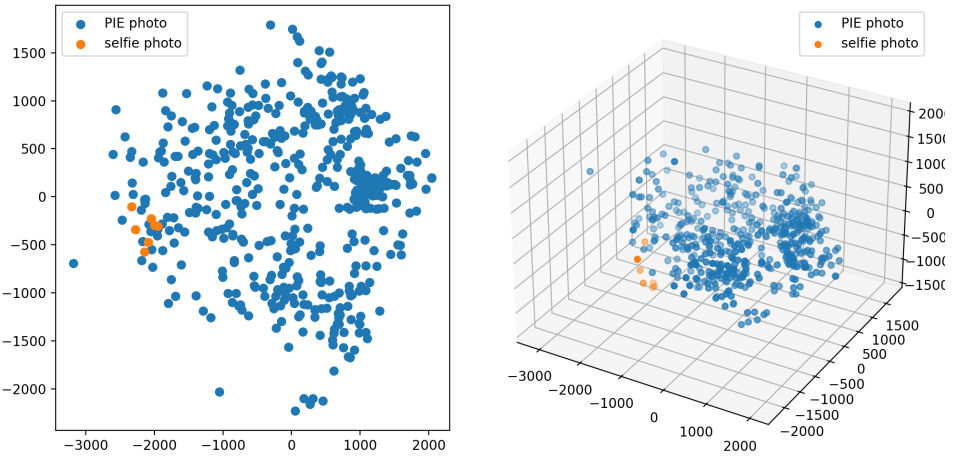
2.1 Data Distribution in 2D and 3D

PCA is applied to reduce the dimensionality of vectorized images from 1024 (i.e., 32×32) to 2 and 3, respectively. Figure 1 visualised the distribution of the projected data points. According to Figure 1a, when reducing the dimensions to 2, the data points of selfie photos are relatively close in two-dimensional space. However, it is hard to cluster them in this space, because there are several data points does not belong to this class among these orange points (i.e., selfie photos data points).

[1] https://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie/Multi-Pie/Home.html

[2] To avoid the situation when selfie photos do not include in the randomly sampled 500 images from the training set, I add the selfie photo manually instead of randomly sampling all 500 images.

While the situation is much better in three-dimensional space (see Figure 1b), in which fewer blue points are among orange points. This indicates that when reducing the dimensionality to 2, it may lose some features that play a significant role in classifying different data points. Figure 2 shows the corresponding 3 eigenfaces used for the dimensionality reduction.



(a) Two dimensions (b) Three dimensions

Figure 1: Data distribution visualization of the projected data point after PCA processing with the dimensionality of 2 and 3, respectively. The selfie photo data points are highlighted in orange colour.

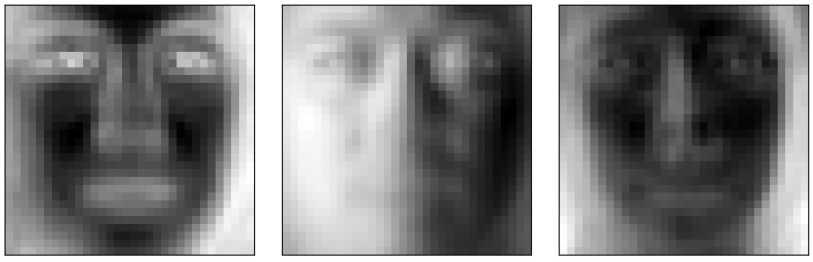


Figure 2: Corresponding 3 eigenfaces used for the dimensionality reduction.

2.2 Nearest Neighbor Classification Results

After exploring the data distribution, the face images are pre-processed by applying PCA with the reduced dimensionality of 40, 80 and 200, respectively, and then classified via the rule of K-Nearest Neighbors Neighbour (KNN). The upper part of Table 2 shows the classification accuracy on both the PIE test set and the selfie test set. According to the results, samples from the selfie photo class have 100% in all three cases. Such high accuracy may be caused by two possible reasons: (1) the data from this class are generated in a different way from other PIE samples (such as different locations and zooms of the facial features in the image), which leads to they located far away from other classes in a high-dimensional space and easy to be classified, and (2) there are just 3 samples in the selfie test samples, which makes it easier to get a high classification performance.

3 Linear Discriminant Analysis (LDA)

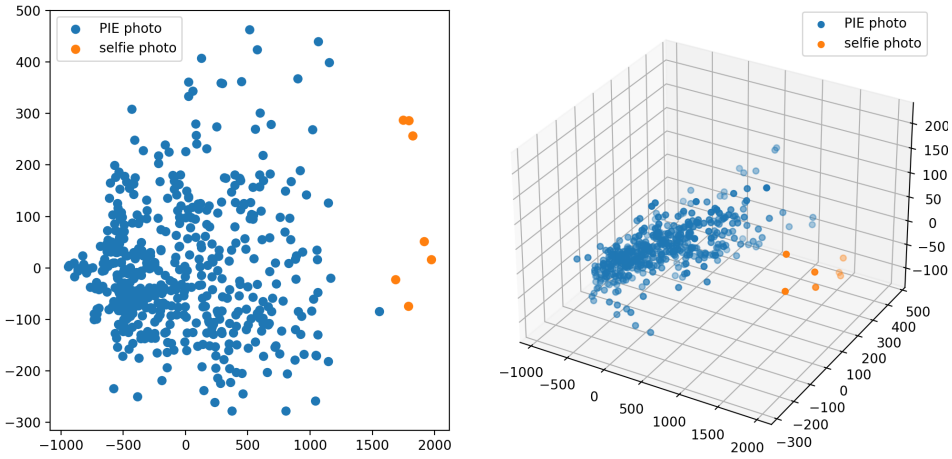
Compared to PCA, LDA is a supervised learning approach, which finds directions of maximum class separability. This section shows the data distribution when using LDA to reduce dimensionality and the performance of the classification accuracy.

Table 2: KNN classification accuracy after PCA pre-processing with the dimensionality of 40, 80 and 200 (*upper*) and LDA pre-processing with the dimensionality of 2, 3 and 9, trained on randomly sampled 500 images (*middle*) and all 2982 images in total (*lower*). The best k values are searched in [1, 3, 5, 7, 9, 11, 13].

Method	Dimension	k	PIE acc (%)	Selfie photo acc (%)
PCA	40	1	54.12	100
	80	1	57.65	100
	200	1	61.57	100
LDA-500	2	1	15.69	100
	3	1	27.45	100
	9	1	45.96	100
LDA-all	2	1	20.31	100
	3	1	33.02	100
	9	1	83.92	100

3.1 Data Distribution in 2D and 3D

Similarly to Section 2.1, 500 samples are generated for 2d and 3d data distribution visualization after reducing the dimension via LDA. Figure 3 shows the distributions in 2d and 3d, respectively, where selfie photo data are highlighted in orange. Compared to the results of PCA pre-processing (Figure 1), via LDA, data from different classes are more likely to separate from each other, as indicated by there being no blue data point between the orange ones.



(a) Two dimensions (b) Three dimensions

Figure 3: Data distribution visualization of the projected data point after LDA processing with the dimensionality of 2 and 3, respectively. The selfie photo data points are highlighted in orange colour.

3.2 Nearest Neighbor Classification Results

The results of classification based on the nearest neighbour rule can be found in the middle and lower part of Table 2. When using randomly sampled 500 images as the training set (i.e., LDA-500 in the middle part of the table), the performance of classification is not comparable with PCA. This may caused the reason that when reducing to a very low dimension (i.e., 2, 3 and 9), the information or the features of the original data has an unbearable loss for classification. However, this undesirable performance may be improved by adding more samples to the training set (i.e., LDA-all in the lower part of the table). Though it still has not very outstanding performances when reducing the dimension to 2 and 3, when using 9-dimensional projected vectors to do the classification task, the accuracy can reach 83.92%, which is 36.3% higher than the best performance of PCA in Section 2.1.

4 Support Vector Machine (SVM)

The SVM is also a supervised learning method. Table 3 shows the classification accuracy of SVM models with different penalty parameters C (i.e., 0.01, 0.1 and 1) and inputs (i.e., raw images, 40- and 80-dimensional vectors after PCA pre-processing). According to the results, SVM for this task is not sensitive to the parameter C. On the other hand, after PCA pre-processing, the data may lose some features that can beneficial for classification, leading to relatively lower accuracy compared with directly using raw images as input. However, overall the performances of SVM (i.e., over 99% accuracy) considerably outperforms the nearest neighbour classification after PCA and LDA pre-processing (see Table 2).

Table 3: The classification accuracy of SVM models with different penalty parameters C and inputs. *Raw* means using the raw face images as the input of SVM. *PCA-200* and *PCA-80* represent the input data that are PCA pre-processed with the dimensionality of 200 and 80, respectively.

Input	C	Accuracy (%)
Raw	0.01	99.92
	0.1	99.92
	1	99.92
PCA-200	0.01	99.77
	0.1	99.77
	1	99.77
PCA-80	0.01	99.69
	0.1	99.69
	1	99.69

5 Convolution Neural Networks (CNN)

CNN is a very common and popular deep learning model for image classification task. In this report, a CNN with two convolutional layers and one fully connected layer is trained for classifying the face images. The kernel sizes are 5, and a max pooling layer with a kernel size of 2 and a stride of 2 follows each convolutional layer. ReLU is selected as the activation function after the fully connected layer. To make the training of CNN faster and more stable, batch normalization layers are added to re-centring and re-scaling the output of each layer. The detail of the CNN architecture can be found in the code as follows.

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=20,
                               kernel_size=5, stride=1, padding=1)
        self.conv2 = nn.Conv2d(in_channels=20, out_channels=50,
                               kernel_size=5, stride=1, padding=1)
        self.bn1 = nn.BatchNorm2d(20)
        self.bn2 = nn.BatchNorm2d(50)
        self.pool = nn.MaxPool2d(2,2)
        self.fc1 = nn.Linear(50 * 6 * 6, 500)
        self.out = nn.Linear(500, 26)

    def forward(self, x):
        output = F.relu(self.conv1(x))
        output = self.pool(self.bn1(output))
        output = F.relu(self.conv2(output))
        output = self.pool(self.bn2(output) )
        output = output.view(-1, 50 * 6 * 6)
        output = F.relu(self.fc1(output))
        output = F.relu(self.out(output))
        return output
```

Listing 1: The code for CNN architecture.

Figure 4 and Table 4 show the training loss and the accuracy on the test set. According to the results, although with some fluctuations, **the accuracy has already been 100% since 470 epochs**, indicating the trained CNN model performs perfectly on this classification task, outperforming other three appoarches.

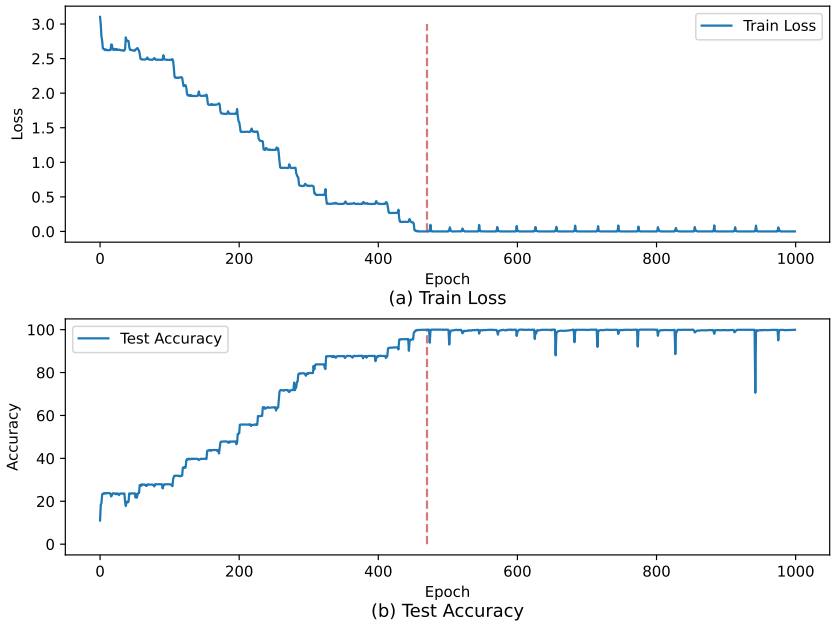


Figure 4: The training loss and test accuracy of the CNN during the training stage. The red dashed line indicates epoch 470 when the model first reaches 100% classification accuracy.

Table 4: Training loss and classification accuracy on testing set on selective epochs. The bolded epoch 470 is the first time when the model reaches 100% accuracy.

	Epoch 0	Epoch 100	Epoch 300	**Epoch 470**	Epoch 600	Epoch 800	Epoch 900
Loss	3.1031	2.4795	0.6634	**0.0001**	0.0156	0.0002	0.0001
Accuracy	11.03	27.93	79.66	**100**	98.83	99.92	99.77