Face Recognition

EE5907 CA2

Student Name: Lin Yi Student ID: A0244307L

Dataset Preparation

In this project, the CMU PIE dataset and 10 selfies photos are used as the dataset.

For the PIE open access dataset, 25 subjects were selected in a random manner. The 25 selected subjects are: ['29', '42', '12', '2', '44', '6', '38', '17', '26', '35', '50', '11', '10', '24', '37', '5', '43', '15', '13', '60', '53', '52', '36', '21', '28']. For each chosen subject, 70% of the images are sampled from the subject's folder as the training set, and the remaining 30% are used as the testing set. In total, there are **2971** training samples and **1273** testing samples from the PIE dataset.

For the 10 selfie photos for myself, these photos are first converted to gray-scale images and resized into the size of (32, 32). Among them, **7** selfie-photos are selected and include into the training set and **3** are used as the testing set.

The following figures show an example face image from the PIE dataset and a resized gray-scale selfie image from my own dataset.

(a) A face image from the PIE dataset

(b) A face image from my own pictures

Figure. 1 Examples of face images used in this project

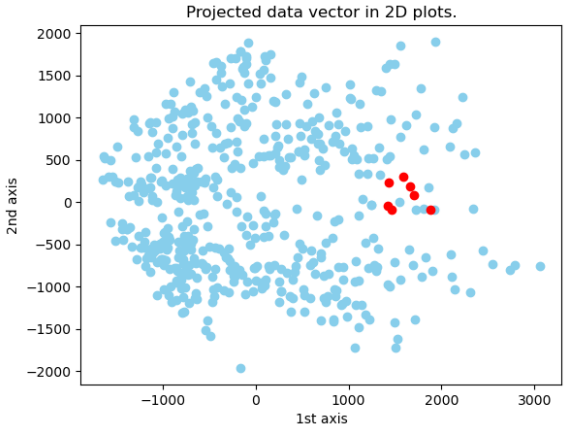
1. Principle Component Analysis (PCA)

1.1 PCA based data distribution visualization

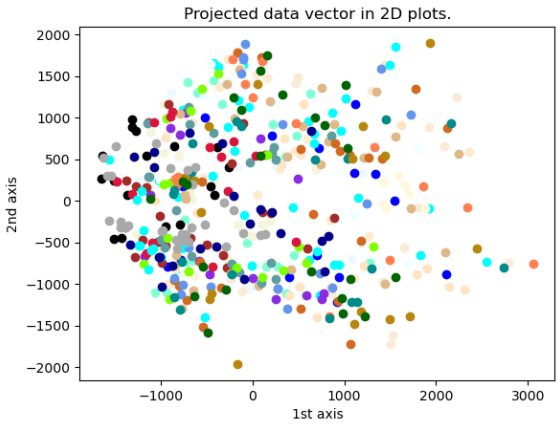
In this part, PCA was applied to reduce the dimensionality of data and for data visualization. The raw face images are first vectorized into 1024-dimensional vectors. And 500 images are

randomly sampled from the training dataset consisting of the PIE training set and my own training photos. Then PCA is implemented and applied on these 500 images to reduce the dimensionality of the vectorized images to 2 and 3, respectively.

The projected data vector after applying PCA transform in 2d plots are shown in the figures below. The left image (a) shows the projected data points where the points colored in red corresponds to my own selfie photos. The right image (b) shows the projected data points from all 500 samples and each color represents the sample points from the same subject.



(a) 500 sample points with highlighted points from my own pictures

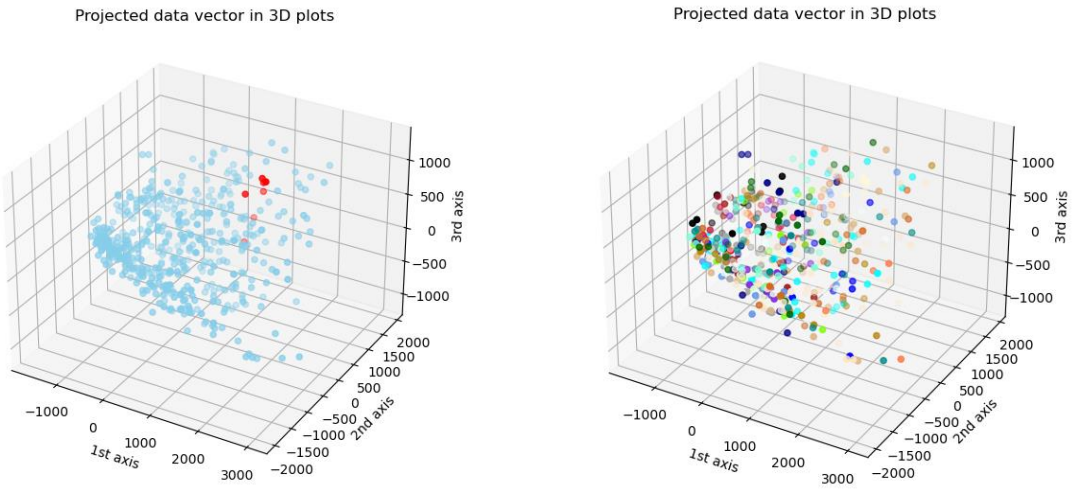


(b) 500 sample points from all 26 subjects

Figure 2. The 2D plot of the projected data after PCA

As can be observed in figure (b) above, when the dimension of data points is reduced to 2 using PCA, the data points are relatively scattered among different classes. However, from figure (a), it can be observed that the red points are relatively concentrated and aggregated together, which indicates that the features of my own photos are different from the features of the PIE dataset to some extent, and thus, the corresponding data points are more clearly separated from the data points from the PIE dataset.

Similarly, the 3d plots of the projected data after applying PCA to reduce the data dimensionality from 1024 to 3 are shown in the figures below. The left image (a) shows the projected data points where the red points correspond to my own selfie photos. The right image (b) shows the projected data points from all 500 samples and each color represents the sample points from a specified subject.



(a) 500 sample points with highlighted points from my own pictures

(b) 500 sample points from all 26 subjects

Figure 3. The 3D plot of the projected data after PCA

As shown in the figure above, when the points are reduced to dimensionality of 3, the data points are still relatively scattered between different classes.

The corresponding 3 eigenfaces used for the dimensionality reduction are shown in the figure below.

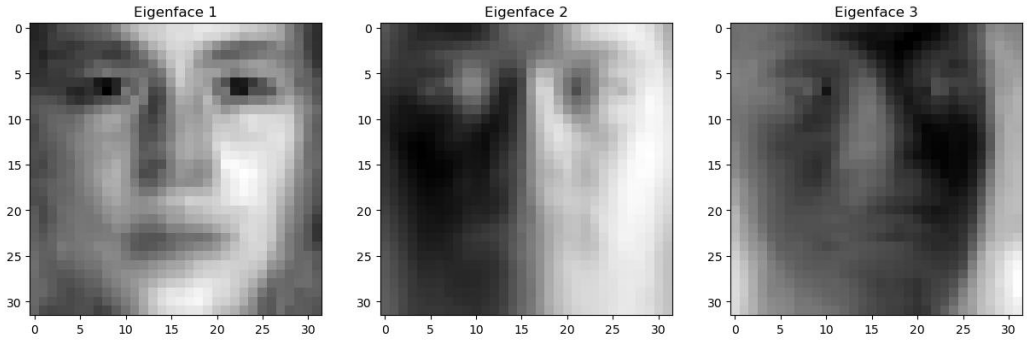


Figure 4. The three eigenfaces

It can be observed that these eigenfaces indeed contain some significant information about the face images. And as its name suggested, these eigenfaces themselves form a set of basis image that can be used to construct or represent the original face image dataset.

1.2 PCA plus nearest neighbor classification results

In this part, PCA is applied to reduce the dimensionality of face images to 40, 80 and 200, respectively. Then, K-Nearest Neighbors (KNN) is applied on these projected data to classify the images into different subjects. In this task, I chose the number of nearest neighbors to be 5 in KNN. The following table shows the classification results of the cases when the dimensionality is 40, 80, 200, and the case when dimension is not reduced, i.e., 1024.

Accuracy	Dimensionality			
Dataset	40	80	200	1024
PIE test images	84.52%	86.41%	87.67%	91.52%
My test images	100.00%	100.00%	100.00%	100.00%

Table 1. Classification results from PCA plus KNN

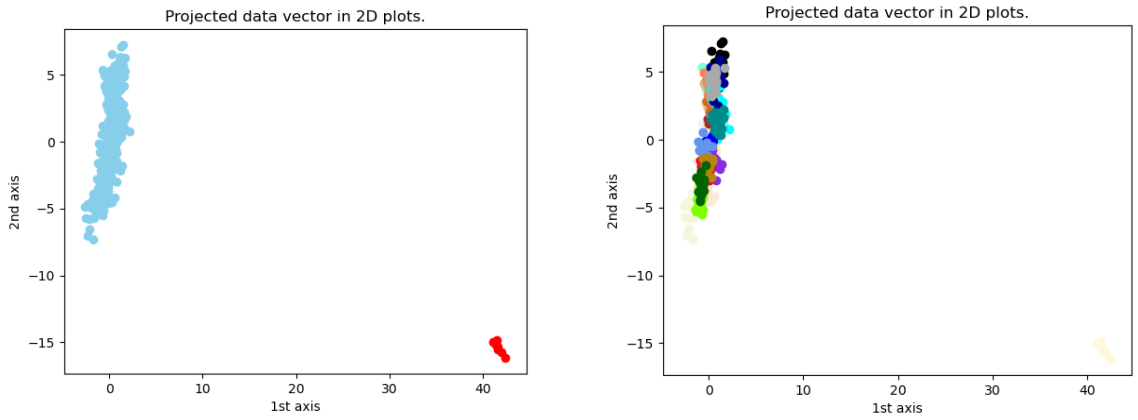
As can be seen from table 1, the classification accuracy of KNN on my own test images is 100 percent in any dimension. However, for the PIE test set, as the dimensionality increases, the accuracy on the PIE test set gradually increases.

2. Linear Discriminator Analysis (LDA)

2.1 LDA based data distribution visualization

In this part, LDA was applied to reduce the dimensionality of data and to visualize the data distribution after projection. Similarly, 500 images are randomly sampled from the training dataset consisting of the PIE training set and my own training photos. Then LDA is implemented and applied on these 500 images to reduce the dimensionality of the vectorized images to 2 and 3 respectively.

The 2D plots of the projected data points with dimensionality of 2 are shown in the figures below. The left image (a) shows the projected data points where the red points correspond to my own selfie photos. The right image (b) shows the projected data points from all 500 samples and each color represents the sample points from a specified subject.



(a) 500 samples points with highlighted points from my own pictures

(b) 500 sample points from all 26 subjects

Figure 5. The 2D plot of the projected data after LDA

As shown in the figure above, when these 500 sample points include my own pictures, after

visualizing the data distribution it can be observed that the features of my own pictures are quite different from the features of PIE's pictures --- along the first axis, the data points from my own pictures (the red points) are all concentrated around 40, while the data points from PIE (the blue points) are all concentrated around 0. Because of this gap between the features, all the data points of PIE are "compressed" to the left side of the image.

To see more clearly the differences in characteristics between different subjects within the PIE dataset, I sampled 500 samples from the PIE dataset only, as shown in the following figure.

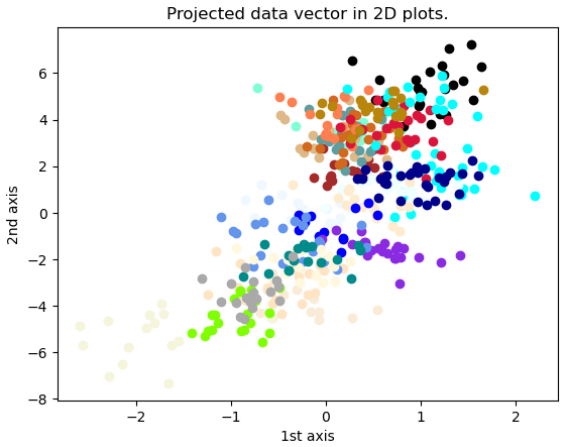
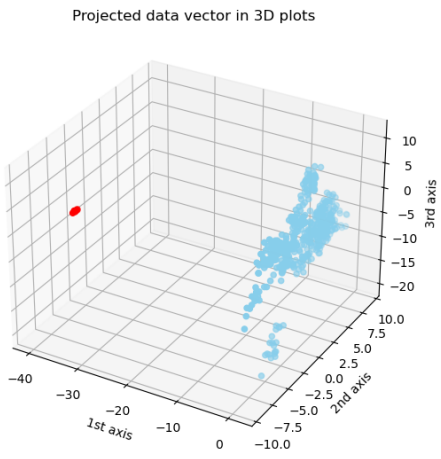


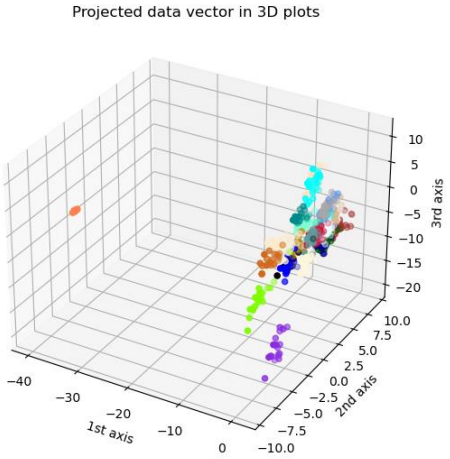
Figure 6. The 2D plot of the projected data after LDA (PIE data samples only)

As shown in the figure above, it can be observed that the data points from different subjects are located in different clusters. This indicates that: after applying LDA to reduce the data dimensionality to 2, the data points may be easier to classify since they are in different clusters. However, there still exists some overlapping between these clusters.

The 3D plots of the projected data points with dimensionality of 3 are shown in the figures below. The left image (a) shows the projected data points where the projected points corresponding to my own selfie photos. The right image (b) shows the projected data points from all 500 samples and each color represents the sample points from a specified subject.



(c) 500 sample points with highlighted



(d) 500 sample points from all 26

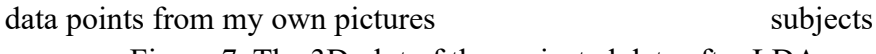


Figure 7. The 3D plot of the projected data after LDA

Similar to the 2D plots, when the dimension of the data points is reduced to 3, it can be observed that data samples from different subjects are concentrated in different clusters.

2.2 LDA plus nearest neighbor classification results

In this part, LDA is applied to reduce the dimensionality of face images to 2, 3 and 9, respectively. Then K-Nearest Neighbors is applied on these projected data to classify the images. The following table shows the classification results when using 5 nearest neighbors and the dimensionality is 2, 3, 9, and the case when dimension is not reduced, respectively.

Accuracy	**Dimensionality**			
Dataset	2	3	9	1024 (no LDA)
PIE test images	25.53%	51.37%	91.04%	91.52%
My test images	33.33%	0.00%	0.00%	100.00%

Table 2. Classification results from LDA plus KNN

As we can see from the Table 2, when applying LDA, as the dimension increases, the classification accuracy on the PIE testing set will increase. If no LDA is apply, i.e., the dimensionality of the data remains 1024, then the classification accuracy can achieve the highest. However, when testing on my own testing images, I noticed that the classification accuracy is not so stable in different testing trials. But in general, the classification accuracy is very low when the training and testing samples are projected to a lower dimension. This may be due to the fact that the number of my own pictures in the training set is only 7, also the feature difference between my own pictures and the PIE dataset is very larger. As a result, when applying KNN for classification, the decision boundary is more likely to be closer to the PIE data points. Thus, my testing samples are more likely to be wrongly classified.

3. Support Vector Machine (SVM)

In this part, face images are vectorized and pre-processed using PCA to reduce the dimensionality into 80 and 200. Then, linear SVM is applied to classify the testing data point. In this part, I combined the testing set from the PIE dataset and my own dataset and report the classification accuracy on this combined testing set. Different values of the penalty parameter C are set and experimented (i.e., C=1 × 10−2, 1 × 10−1, 1).

The following tables show the SVM classification results with different dimensionality and different parameter values.

	Dimensionality		
penalty	80	200	1024 (No PCA)
0.01	99.45%	99.84%	99.53%
0.1	99.45%	99.84%	99.53%
1	99.45%	99.84%	99.53%

Table 3. Classification results from PCA plus SVM on PIE dataset

	Dimensionality		
penalty	80	200	1024 (No PCA)
0.01	100.00%	100.00%	100.00%
0.1	100.00%	100.00%	100.00%
1	100.00%	100.00%	100.00%

Table 4. Classification results from PCA plus SVM on my own dataset

From Table 3, it can be observed that, as dimensionality increases, the classification accuracy on the PIE testing dataset will first increase then decrease. That is, when dimensionality equals 200, the classification accuracy is higher than the accuracy when dimensionality equals 80. However, when PCA is not applied, that is when dimensionality equals 1024, the accuracy is slightly lower than the case when dimensionality equals 200. The possible reason for that may be when the dimensionality is too larger, the features space of the data points is too complex for the linear SVM classifier, so the classifier may be "overfit" and the classification performance on the testing set will be worse.

From Table 4, it can be observed that, no matter what dimensionality it is, the classification accuracy on my own testing dataset is all 100%.

However, as we can see from both Table 3 and Table 4, the penalty parameter doesn't have much effect to the classification accuracy. The possible reason for this maybe we use linear SVM classifier instead of other more complex type of SVM as our classifier.

4. Convolutional Neural Network (CNN)

In this part, I implemented the Convolutional Neural Network for classification. In order to fine tune the model and find the optimal model's hyperparameters and configurations, I first split the original training dataset into training set and validation set in a ratio of 7:3, resulting in a training set with 2085 samples and a validation set with 893 samples.

The structure of the network is designed as follows. I used two 2D convolutional layers, each of which is followed by an activation function ReLU and a max-pooling layer. After these two convolutional layers, the feature maps will go through a fully connected layer. Finally, the Linear layer will map feature maps into raw probability logits with a dimensionality of 26.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 20, 28, 28]             520
              ReLU-2          [-1, 20, 28, 28]               0
         MaxPool2d-3          [-1, 20, 14, 14]               0
            Conv2d-4          [-1, 50, 10, 10]          25,050
              ReLU-5          [-1, 50, 10, 10]               0
         MaxPool2d-6           [-1, 50, 5, 5]               0
           Flatten-7                [-1, 1250]               0
            Linear-8                 [-1, 500]         625,500
              ReLU-9                 [-1, 500]               0
           Linear-10                  [-1, 26]          13,026
================================================================
Total params: 664,096
Trainable params: 664,096
Non-trainable params: 0
```

Figure 8. CNN structure

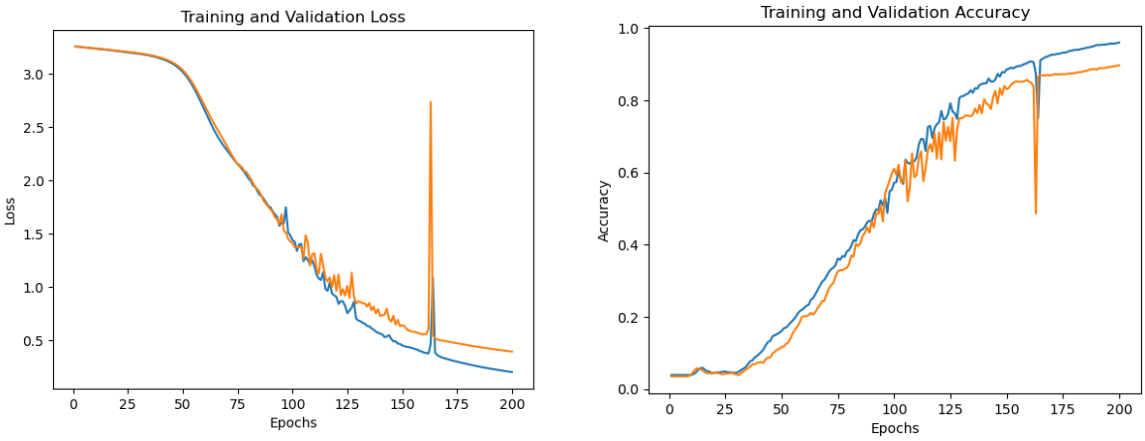
4.1 Experiment with different hyperparameters

To find out the optimal hyperparameters, I tried different set of the hyperparameters and did some experiments. The initial hyperparameters are set as follows:

- Number of epochs: 200
- Learning rate: 0.01
- Batch size: 128
- Optimizer: Stochastic Gradient Descent (SGD)
- Loss function: Cross Entropy Loss

Number of epochs

One of the major challenges in training a neural networks is how long it will take to train a model that has good enough performance on the testing set. Thus, I first test the network using different number of epochs. The following figures show the results when the number of epochs equal 200.

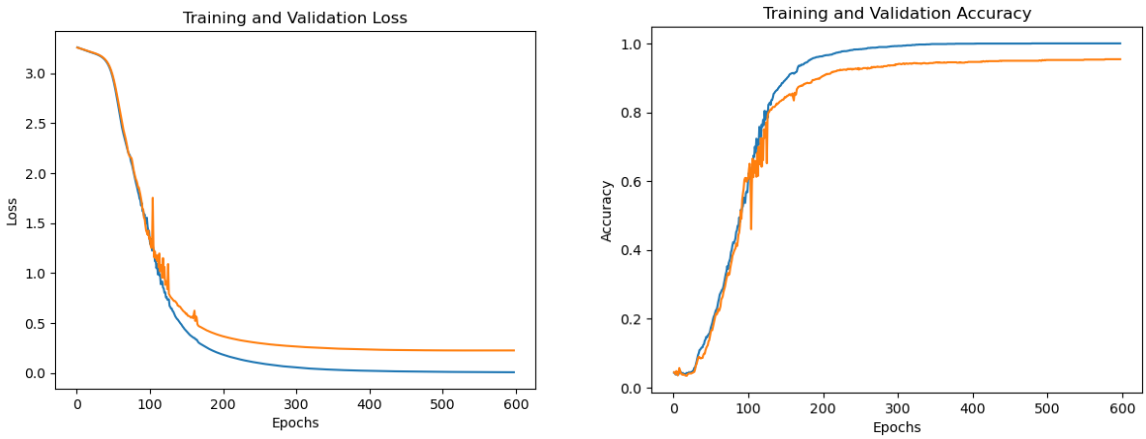


(a) Training vs. validation loss (b) Training vs. validation accuracy

Figure 9. Training Curve when number of epochs equal 200

After training with 200 epochs, the final classification accuracy on the PIE test dataset is **92.30%**, while the testing accuracy on my own testing dataset is **100.00%**. Although the classification accuracy is satisfactory enough, from Fig.9, it can be observed that the training loss and the validation loss have not converged yet at the 200th epoch and still has a trend of decreasing.

Therefore, I increase the number of epochs to 600 and use the model "**early stopping**" strategy to find the best model. Early sopping means training the model on the training dataset but stop the training at the point when the validation loss starts to stagnate or degrade. Here, I set the value of "patience" to be 50, which means when the validation loss does not have any decrease in more than 50 epochs, model training process will be stopped and the model at the lowest validation loss will be saved for testing in the test dataset. The following figures show the loss curve and the accuracy curve of both training and validation.



(a) Training vs. validation loss (b) Training vs. validation Accuracy

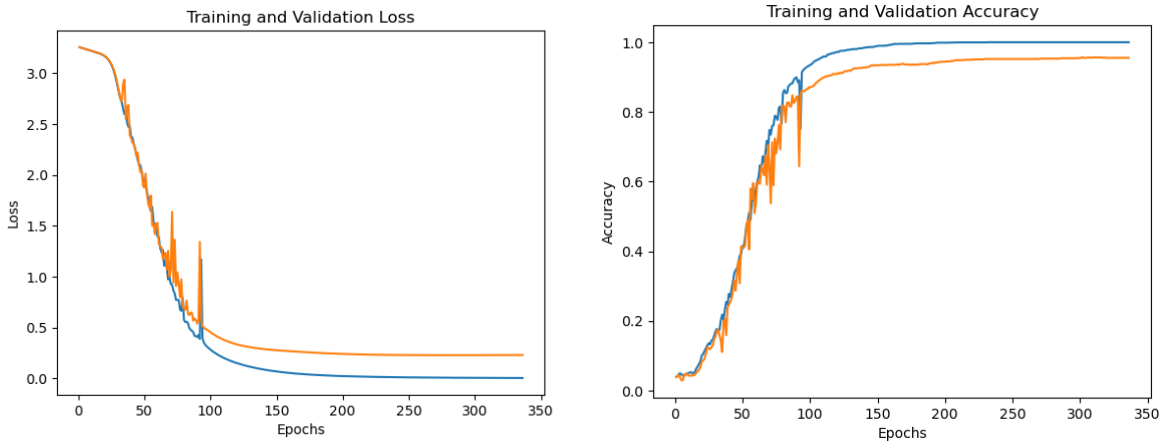
Figure 10. Training Curve using the early stopping strategy

From the figure above, it can be noticed that, after 600 epochs of training, the training and validation loss tends to be converged and the validation accuracy is very close to 1. Besides, with the early stopping strategy, the best model is saved at epoch 545. The final classification accuracy of this best model on the PIE testing set is **96.34%** and **100.00%** on my own testing set.

Number of learning rate

To see how learning rate will affect the training process as well as the final classification performance, I try different values of learning rates.

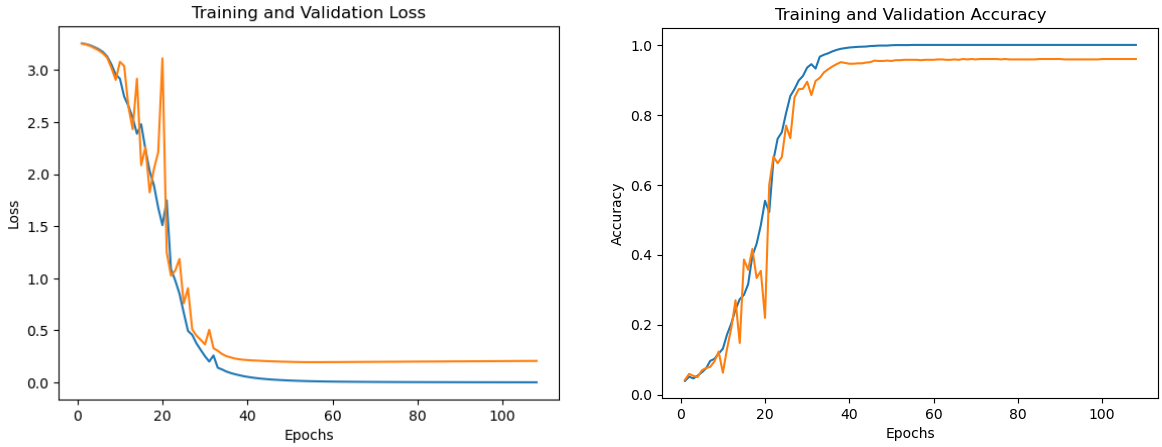
(1) The following figures shows the training process when **learning rate = 0.02.** In this case, the best model is saved at epoch 285.



(a) Training vs. validation loss (b) Training vs. validation accuracy

Figure 11. Training Curve when learning rate = 0.02

(2) The following figures shows the training process when **learning rate = 0.1.** In this case, the best model is saved at epoch 56.



(a) Training vs. validation loss (b) Training vs. validation accuracy

Figure 12. Training Curve when learning rate = 0.1

The classification accuracy of this best model on the PIE testing set and on my own testing set is summarized in the table below.

Accuracy	Learning rate=0.01	Learning rate=0.02	Learning rate=0.1
PIE testing set	96.34%	96.47%	97.56%
My testing set	100.00%	100.00%	100.00%

Table 5: CNN Classification Accuracy with different learning rates

From table 5, it can be noticed that, under different conditions of learning rates, the classification accuracy didn't show very significant difference. But still, when learning rate is set to 0.1, CNN can achieve the highest classification accuracy on the PIE testing set; the training process of CNN is also capable of the fastest convergence.

Besides, on my own testing set, the classification accuracy can be as high as 100%. This is possibly because the features extracted by the CNN inner layers show significant difference

with the features from the PIE dataset. As a results, testing samples from my own dataset would be easier to classified by the network.

Overall, this shallow CNN network I used in this experiment was able to obtain decent accuracy on the testing set under suitable hyperparameters.