EE5907 Pattern Recognition CA2 Report

Fan Ruiming A0279678Y

November 9, 2023

Abstract

This report is a course project submitted to EE5907. The aim of this project is to construct a face recognition system via Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), and Convolution Neural Network (CNN). PCA and LDA are used to perform data dimensionality reduction and visualization to understand the underlying data distribution. Then lower-dimensional data are classified based on the nearest-neighbor classifier. In addition, SVM and CNN are also used to classify face images.

1 Dataset

This project is conducted on the CMU PIE dataset and selfie photos taken by myself. There are in total 68 different subjects and **I randomly selected 25 subjects from it**. For each chosen subject, use 70% of the images provided for training and use the remaining 30% for testing. In addition to the provided CMU PIE images, I took 10 selfie photos for myself, converted them to grayscale images, resized them into the same resolution as the CMU PIE images (i.e., 32×32) and split them into 7 for training and 3 for testing. The size of the raw face image is 32×32 pixels, resulting in a 1024 dimensional vector for each image. The selected subjects and information of dataset are shown in Table. 1.

Table 1: Selected subjects and detail of dataset

	SELECTED SUBJECT			
data_idx	[3 4 6 7 9 10 13 14 20 21 25 26 35 37 41 49 50 53 54 57 59 62 63 64 67]			
		CMU PIE	**SELFIES**	**TOTAL**
#Samples	**Training set**	2975	7	2982
	Testing set	1275	3	1278
	TOTAL	4250	10	4260
#Objects		25	1	26

To avoid duplicating the data processing, I stored the data and labels of the training and test sets in four files formatted as Numpy binary files (.npy) and named in 'train_x.npy', 'train_y.npy', 'test_x.npy', and 'test_y.npy', respectively.

2 Principal Component Analysis (PCA)

Requirements: Randomly sample 500 images from the CMU PIE training set and your own photos. Apply PCA to reduce the dimensionality of vectorized images to 2 and 3 respectively. Visualize the projected data vector in 2d and 3d plots. Highlight the projected points corresponding to your photo. Also visualize the corresponding 3 eigenfaces used for the dimensionality reduction. Then apply PCA to reduce the dimensionality of face images to 40, 80 and 200 respectively. Classifying the test images using the rule of the nearest neighbor. Report the classification accuracy on the CMU PIE test images and your own photo separately.

2.1 Dimensionality Reduction

To ensure that the 500 samples selected contained my selfies, instead of choosing to randomly select all 500 samples directly from the training set, I randomly selected 493 samples from the CMU PIE and 7 of my selfies respectively in the training set to combine into the dataset used for PCA.

PCA is utilized to reduce the dimensionality of vectorized images from 1024 to 2 and 3, respectively. The projected data distribution after PCA processing with the dimensionality of 2 and 3 are shown in Fig. 1. When examining the left image and reducing the dimension to 2, it becomes apparent that the data points attributed to the selfie (highlighted in orange) are blended with those of the CMU PIE image, and therefore cannot be distinguished with clarity. The situation is enhanced when projecting the data into three-dimensional space. A clearer distinction can be observed between the data points that correspond to the selfie and those that correspond to the image from the CMU PIE. In contrast, it is evident that the reduction to a two-dimensional space results in the loss of critical features crucial to image classification due to a dearth of dimensions. Fig. 2 visualizes the corresponding 3 eigenfaces used for dimensionality reduction.

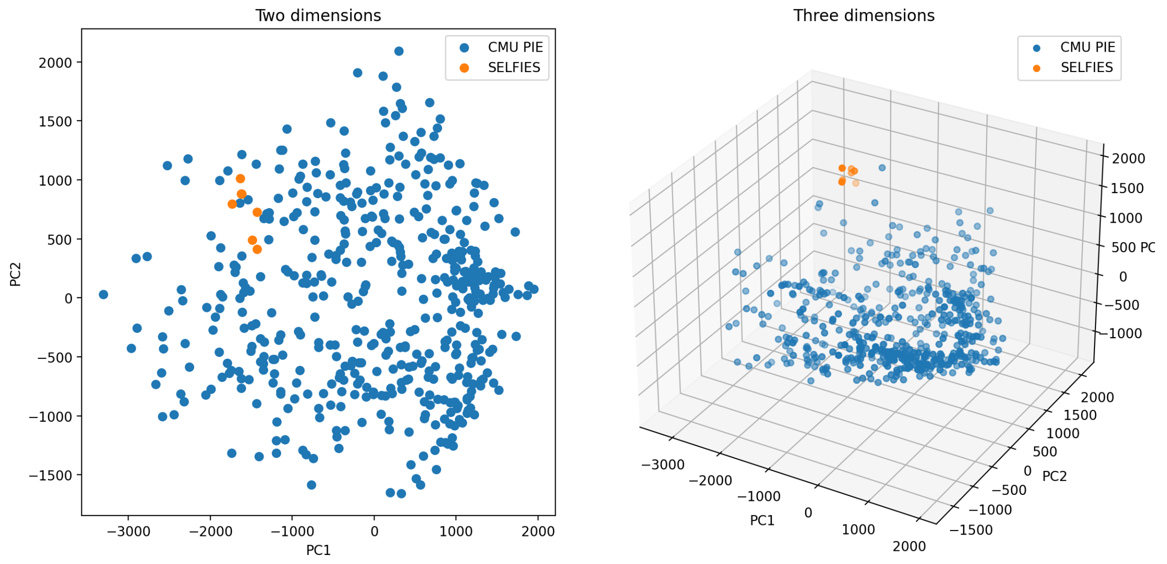


Figure 1: Data distribution visualization of the projected data after PCA processing with the dimensionality of 2 (left) and 3 (right).

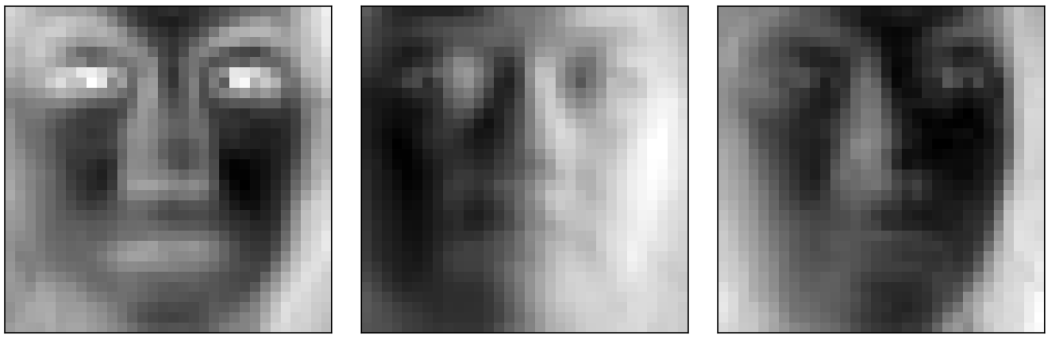


Figure 2: Visualization of the corresponding 3 eigenfaces used for dimensionality reduction

2.2 Classification

After comprehending the distribution of data, the dimensionality was decreased to 40, 80, and 200 through the implementation of PCA. Subsequently, the data labels of the test dataset were anticipated and classified employing K-Nearest Neighbors Neighbour (KNN). I designated the value of k as 1, which is tantamount to assigning the label corresponding to the data point with the closest Euclidean distance. Table. 2 exhibits the accuracy of the classification in three dimensions. The data points 'acc@CMU PIE', 'acc@SELFIES', and 'acc@TOTAL' correspond to the CMU PIE dataset and the accuracy of selfie data points and all data points, respectively. The findings demonstrate that the selfie category samples achieved a classification accuracy of 100% in all three cases. This performance may be attributed to the fact that, in higher-dimensional space, data points corresponding to selfies exhibit a clearer demarcation from other data points. This distinction is already relatively apparent in three-dimensional space, facilitating straightforward classification. However, the accuracy of classifying the data from CMU PIE is relatively low because the data points corresponding to the 25 different labels cannot be well distinguished in such a high-dimensional space, resulting in a high degree of clustering.

Table 2: Classification accuracy of PCA

Dim	acc@CMU PIE	acc@SELFIES	acc@TOTAL
40	0.4400	1.0000	0.4413
80	0.4714	1.0000	0.4726
200	0.4839	1.0000	0.4851

3 Linear Discriminant Analysis (LDA)

Requirements: Apply LDA to reduce data dimensionality to 2, 3, and 9. Visualize distribution of the sampled data (as in the PCA section) with dimensionality of 2 and 3 respectively (similar to PCA). Report the classification accuracy for data with dimensions of 2, 3, and 9, respectively, based on nearest neighbor classifier. Report the classification accuracy on the CMU PIE test images and your own photo separately.

3.1 Dimensionality Reduction

The core idea of LDA is to maximize the between-class mean and minimise the within-class variance. That is, after the data is projected in lower dimensions, the projection points of the same class of data are as close as possible after projection, and the centres of the projection points of different classes of data are as far away as possible.

A training set containing photos from the CMU PIE as well as selfies was generated using the methodology mentioned in Sec. 2.1. Fig. 3 shows the results after LDA dimensionality reduction to two and three dimensions, where the data points corresponding to selfies are marked using red asterisks. From the results presented, the data points corresponding to selfies are well differentiated. Upon comparing the results obtained from the PCA dimensionality reduction showcased in Fig. 1, it becomes apparent that data points belonging to the same category exhibit a higher degree of clustering following low-dimensional mapping. Additionally, observations made within the three-dimensional space suggest the existence of discernible within-class distances between certain categories, allowing for easy distinction.

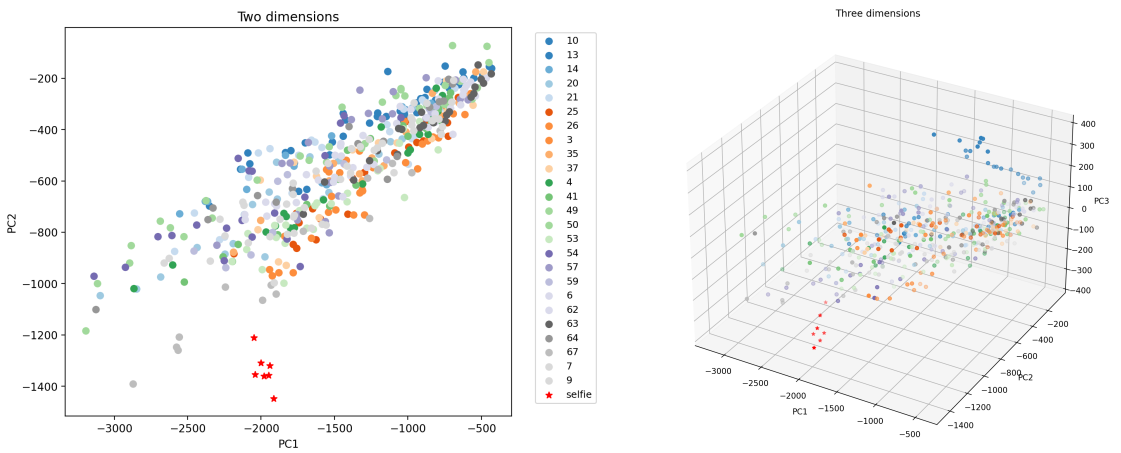


Figure 3: Data distribution visualization of the projected data after LDA processing with the dimensionality of 2 (left) and 3 (right) where the data points corresponding to selfies are marked with a red asterisk.

3.2 Classification

Following the implementation of LDA, the dimensions were reduced to 2, 3, and 9, as stated in sec. 2.2. Afterwards, KNN was utilized to predict and classify the data labels from the test dataset, where k was set to 1. Table. 3 exhibits the accuracy of the classification in three dimensions. The initial training on the dataset of 500 samples yielded unsatisfactory results. Although the classification of easily distinguishable selfies was accurate, the classification of CMU PIE samples mapped to 2D-3D was only 20% accurate. This is attributed to the intolerable losses in classification prediction due to low-dimensionality reduction during the process. Thus, I endeavoured to expand the training set and selected the training set stated in sec. 1 for the purpose of training. The testing results indicate that the outcomes are inadequate when the data is mapped to two dimensions. Nevertheless, as the dimensionality increases, the enhancement becomes increasingly evident when contrasted with the dataset featuring a sample size of 500.

Table 3: Classification accuracy of LDA

#Training Set		Dim	acc@CMU PIE	acc@SELFIES	acc@TOTAL
CMU PIE	493	2	0.1200	1.0000	0.1221
SELFIES	7	3	0.1937	1.0000	0.1956
TOTAL	500	9	0.3498	1.0000	0.3513
CMU PIE	2975	2	0.1082	1.0000	0.1103
SELFIES	7	3	0.3106	1.0000	0.3122
TOTAL	2982	9	0.6322	1.0000	0.6330

4 Support Vector Machines (SVM)

Requirements: Use the raw face images (vectorized) and the face vectors after PCA pre-processing (with dimensionality of 80 and 200) as inputs to *linear* SVM. Try values of the penalty parameter C in $\{1\times10^{-2}, 1\times10^{-1}, 1\}$. Report the classification accuracy with different parameters and dimensions. Discuss the effect of data dimension and parameter C on the final classification accuracy.

Table. 4 shows the classification accuracy of SVM models with different input and penalty parameter C. The inputs are categorised into the following cases: input raw face images and input the face vectors after PCA pre-processing with the dimensionality of 80 and 200. The penalty parameter C is set to $\{0.01, 0.1, 1\}$

Table 4: Classification accuracy of SVM

Input	Dim	C	Accuracy
raw face images	$\sim$	0.01	0.9163
		0.1	0.9163
		1	0.9163
face vectors after PCA pre-processing	80	0.01	0.8905
		0.1	0.8905
		1	0.8905
	200	0.01	0.8905
		0.1	0.8905
		1	0.8905

Theoretically, when the value of C is high, there is less allowance for samples that fall between the boundaries, leading to fewer misclassifications and better alignment with the samples. However, this might not always result in better predictions. Conversely, when C is low, the number of samples that fall between the two boundaries increases, resulting in a greater likelihood of misclassification and reduced fit with the samples. Nevertheless, this may be more reasonable as the samples are often noisy with interferences. The penalty parameter C considers the balance between the model's fit to the training samples and its ability to predict outcomes for the test samples. A larger value for C increases the likelihood of over-fitting. Conversely, a smaller C results in a less complex model.

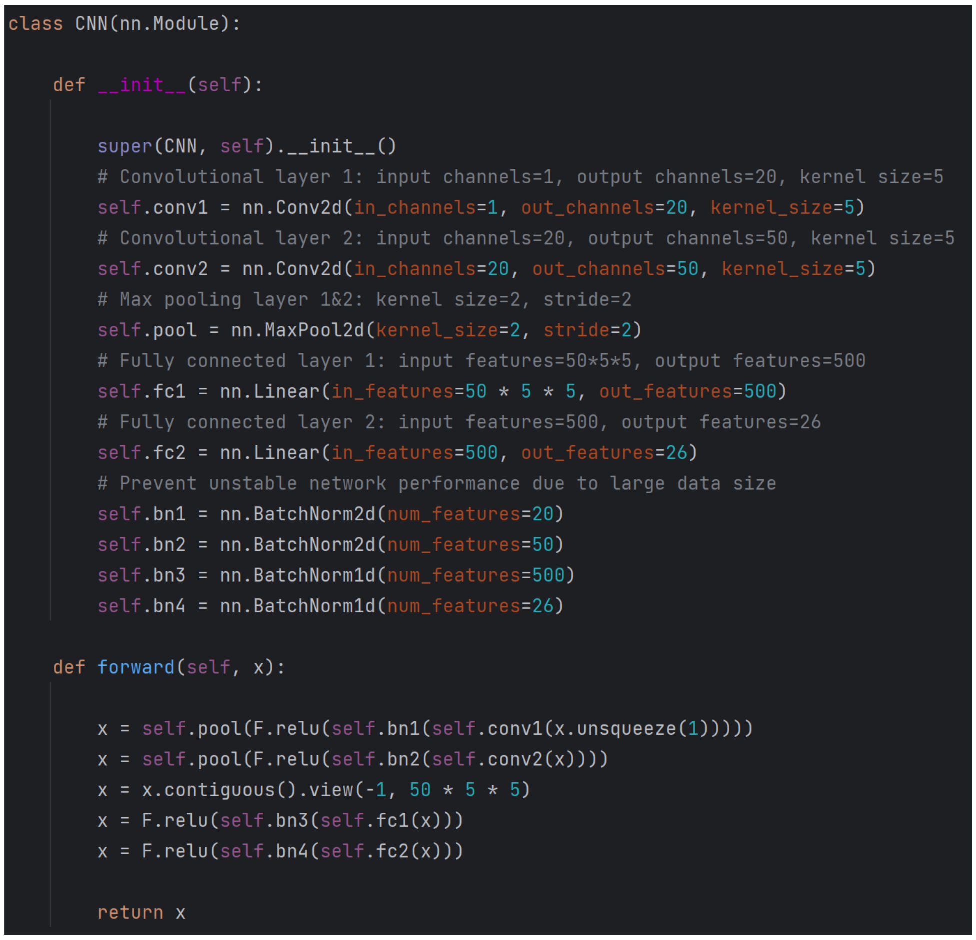
However, from the results shown in Table. 4, the change in the C-value did not have an effect on the results of the experiment. Perhaps this is because this task was not sensitive to the penalty parameter C.

The classification accuracy achieved by inputting the raw graph is higher compared to that achieved by inputting the sample pre-processed by PCA. This could be due to the loss of some dimensions and information critical for classification during sample processing.

5 Convolution Neural Networks (CNN)

Requirements: Train a CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-26. The number of the nodes in the last layer is fixed as 26 as we are performing 26-category (25 CMU PIE faces plus 1 for yourself) classification. Convolutional kernel sizes are set as 5. Each convolutional layer is followed by a max pooling layer with a kernel size of 2 and stride of 2. The fully connected layer is followed by ReLU. Train the network and report the final classification performance.

5.1 Architecture



```
class CNN(nn.Module):

    def __init__(self):

        super(CNN, self).__init__()
        # Convolutional layer 1: input channels=1, output channels=20, kernel size=5
        self.conv1 = nn.Conv2d(in_channels=1, out_channels=20, kernel_size=5)
        # Convolutional layer 2: input channels=20, output channels=50, kernel size=5
        self.conv2 = nn.Conv2d(in_channels=20, out_channels=50, kernel_size=5)
        # Max pooling layer 1&2: kernel size=2, stride=2
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        # Fully connected layer 1: input features=50*5*5, output features=500
        self.fc1 = nn.Linear(in_features=50 * 5 * 5, out_features=500)
        # Fully connected layer 2: input features=500, output features=26
        self.fc2 = nn.Linear(in_features=500, out_features=26)
        # Prevent unstable network performance due to large data size
        self.bn1 = nn.BatchNorm2d(num_features=20)
        self.bn2 = nn.BatchNorm2d(num_features=50)
        self.bn3 = nn.BatchNorm1d(num_features=500)
        self.bn4 = nn.BatchNorm1d(num_features=26)

    def forward(self, x):

        x = self.pool(F.relu(self.bn1(self.conv1(x.unsqueeze(1)))))
        x = self.pool(F.relu(self.bn2(self.conv2(x))))
        x = x.contiguous().view(-1, 50 * 5 * 5)
        x = F.relu(self.bn3(self.fc1(x)))
        x = F.relu(self.bn4(self.fc2(x)))

        return x
```

Figure 4: The code of the CNN architecture

The provided code shows in Fig. 4 outlines the architecture of a Convolutional Neural Network (CNN) implemented using PyTorch's neural network module. This CNN structure contains two convolutional layers, two pooling layers, two fully connected layers.

The network begins with two convolutional layers. The first convolutional layer takes a single-channel input (e.g., a grayscale image) and applies 20 filters of size 5x5, producing 20 feature maps. The second convolutional layer takes these 20 feature maps as input and applies 50 filters of the same size, resulting in 50 output feature maps. Both layers are designed to capture spatial hierarchies in the data by learning from the local regions of the input.

After each convolutional layer, a max pooling layer with a 2x2 kernel and a stride of 2 is used. This pooling operation reduces the spatial dimensions by half and helps in making the representation size smaller and manageable, which also contributes to the network being less sensitive to the exact location of features.

Following the convolutional and pooling layers, the network has two fully connected layers. The first fully connected layer transforms the pooled feature maps into a vector with 500 elements. The second fully connected layer then maps these 500 features to 26 outputs, which could correspond to a classification task with 26 classes.

The network includes batch normalization after each convolutional and fully connected layer, before applying the activation function. Batch normalization is used to normalize the input layer by adjusting and scaling the activations, which helps to stabilize learning and reduce the number of training epochs required to train deep networks. The ReLU (Rectified Linear Unit) activation function is applied after each batch normalization step, introducing non-linearities into the model which allows the network to learn more complex patterns.

5.2 Classification

Fig.5 and Fig. 6 illustrate the training loss and the classification accuracy on the test set, respectively. Some more specific epochs were selected and the corresponding training losses as well as test accuracies are shown in the table. 5. The training loss, which measures how well the CNN is fitting the training data. The loss drops sharply initially, reflecting rapid learning in the early stages, and then it plateaus, demonstrating diminishing gains as the network converges to a solution. The accuracy swiftly ascends to above 90% from the onset of training and maintains a high level throughout, with slight fluctuations indicating stable generalization capabilities on unseen data. The red dashed line in Fig. 6 corresponds to epoch 231, the moment when the model has the highest classification accuracy on the test set.

Table 5: The training loss and classification accuracy on the test set on some specific epoch

Epoch	50	100	200	**231**	300	400	500
LOSS@train	0.2097	0.1213	0.0574	0.0468	0.0405	0.0294	0.0244
ACC@test	96.5571	98.5915	89.5149	**99.0610**	97.1049	97.4961	98.2003

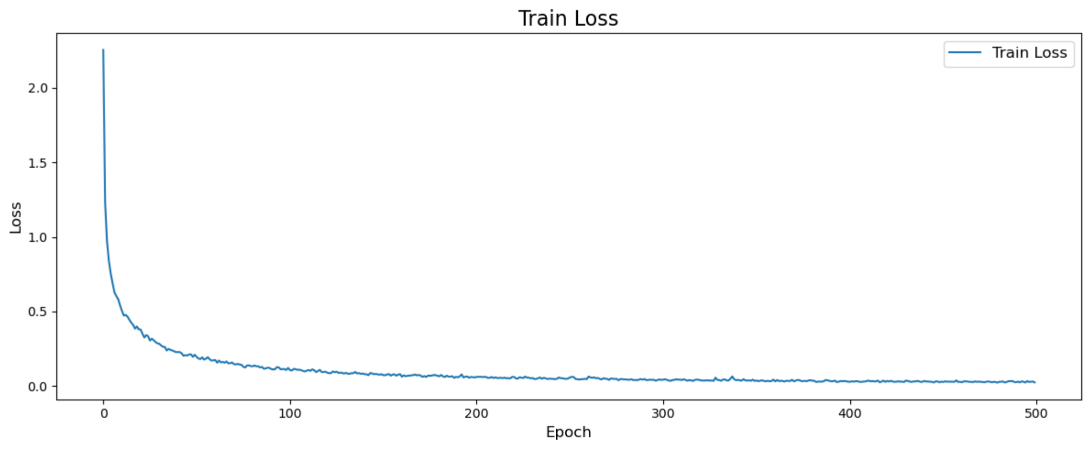


Figure 5: The training loss of the CNN during the training stage.

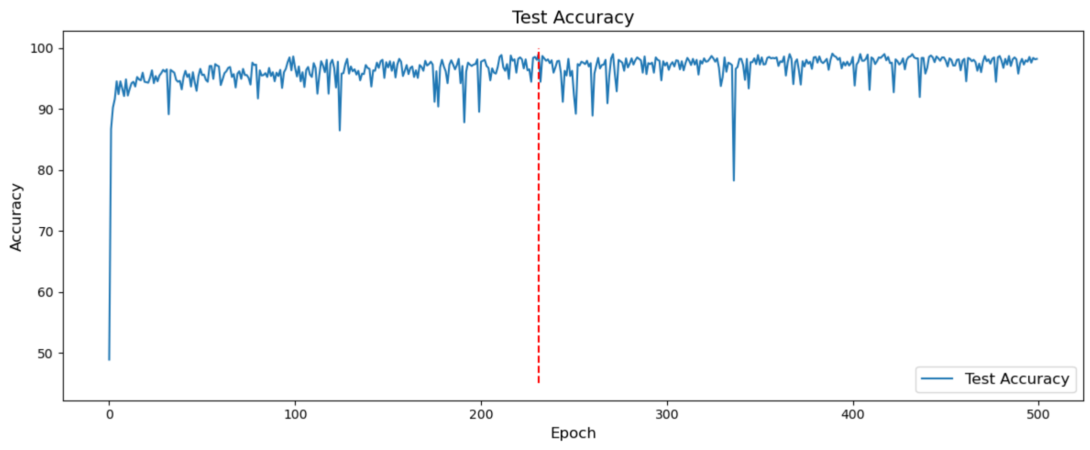


Figure 6: The test accuracy of the CNN.