

|  |  |
| --- | --- |
| **Module** | EE5907 |
|  |  |
| **Name** | Zavier Ong Jin Jie |
|  |  |
| **Matriculation No.** | A0138993L |

# **Dataset**

For this assignment, out of the 68 different subjects, the following folders are chosen:

1, 4, 7, 8, 10, 13, 16, 17, 21, 22, 23, 28, 31, 34, 36, 43, 45, 48, 53, 55, 57, 61, 64, 67, 68

10 selfie photos of myself is also included in the folder ‘mine’ and they are converted to gray-scale, resized to 32x32 pixels split into 7 for training and 3 for testing. Likewise, 70% of each subject is also chosen as the train dataset and 30% of the remaining is used as the test dataset.

# **PCA**

## **Data Distribution Visualization**

### **2D**

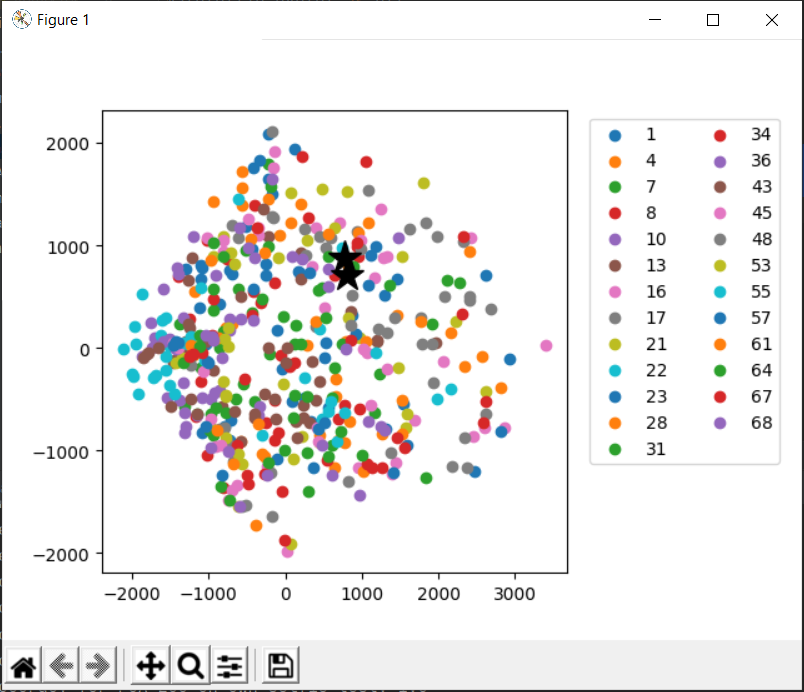


Figure 1. Projected Data Vector (2D)

In Figure 1, we can see the projected data points of the 500 images from the CMU PIE training set. My own selfie photos are marked with a black star instead of the colored circle.

### **3D**

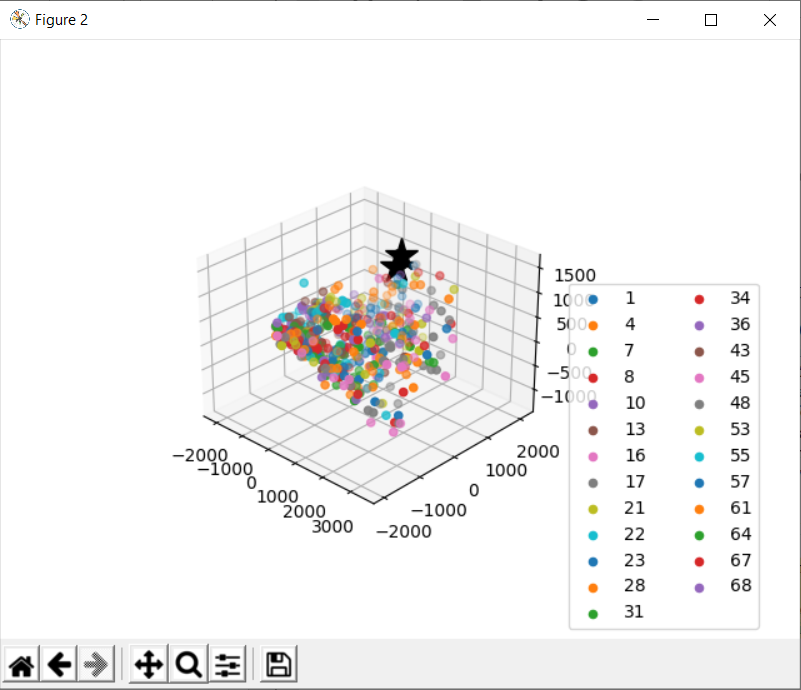


Figure 2. Projected Data Vector (3D)

In Figure 2 above, is the 3D representation of the projected data vector. Similar to Figure 1, my selfie photos are marked as the black star.

### **Eigenfaces**

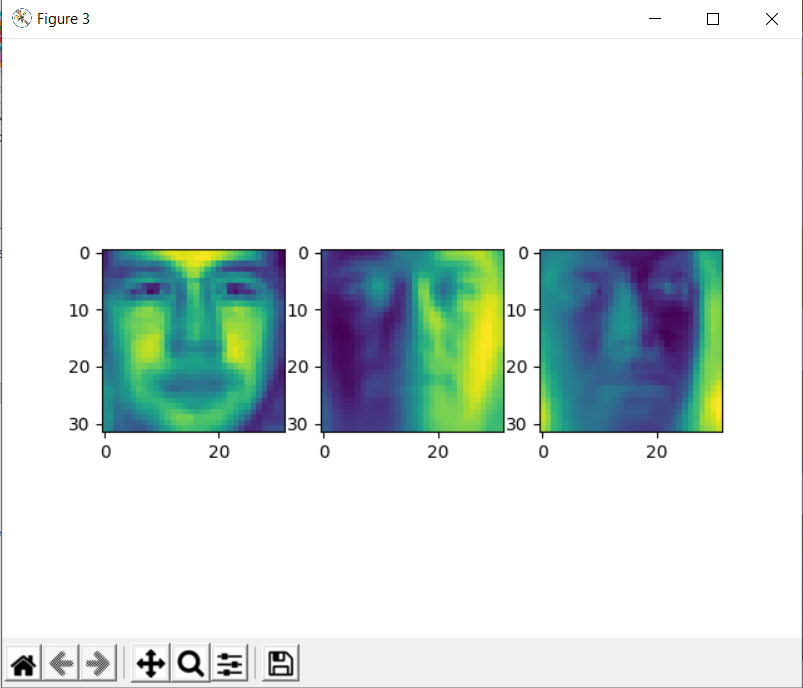


Figure 3. Eigenfaces

The eigenfaces shown in Figure 3 have the greatest variance which is used for dimensionality reduction to 3 features. As the eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images, the eigenfaces resembles faces as they form a basis set of the dataset.

## **Classification Results**

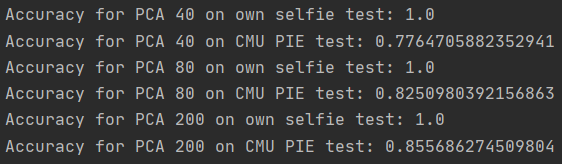


Figure 4. Classification Results (PCA)

As seen in Figure 4, the accuracy of my own selfie tests at PCA 40, 80 and 200 all have the same accuracy of 100% while the accuracy of the CMU PIE data is 77.6%, 82.5% and 85.6% for PCA 40, 80 and 200 respectively. The reason my own selfie tests have 100% accuracy could be because my selfie points are separated from the other test/training points. As for the CMU PIE data, we can observe that at higher dimensions, the accuracy increases.

# **LDA**

## **Data Distribution Visualization**

### **2D**

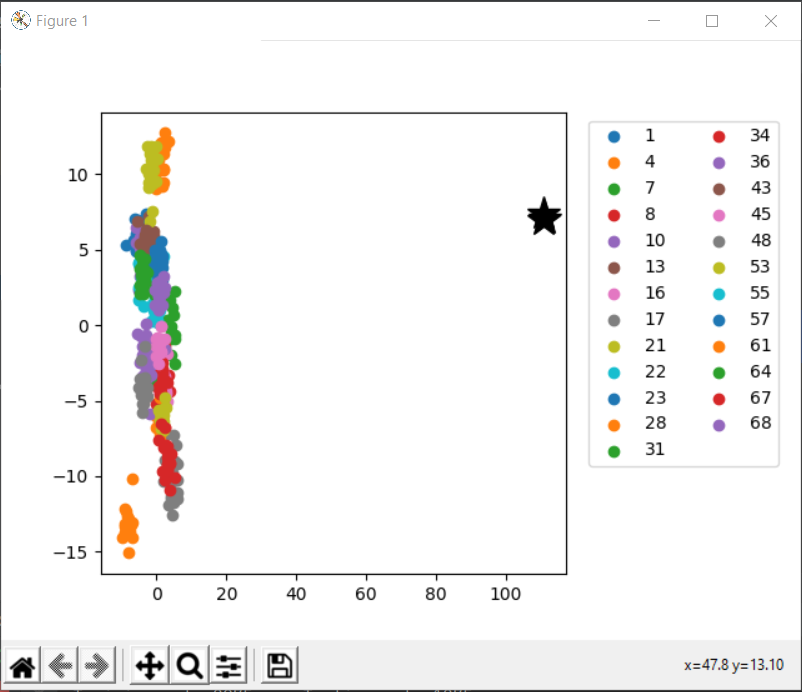


Figure 5. Distribution of sampled data (2D)

Figure 5 above depicts the 2D plot of the projected data using LDA. Like PCA, my selfie photos are marked as the black star. Based on observation, we notice that my selfie images are distinctly separated from all the other datasets. As LDA achieves distinction between classes by maximizing “between class distance” and minimizing “within class distance” through the use of class-specific covariance, my selfie photos, that contains portions of the environment and different lighting makes it much easier to separated as compared to the dataset.

### **3D**

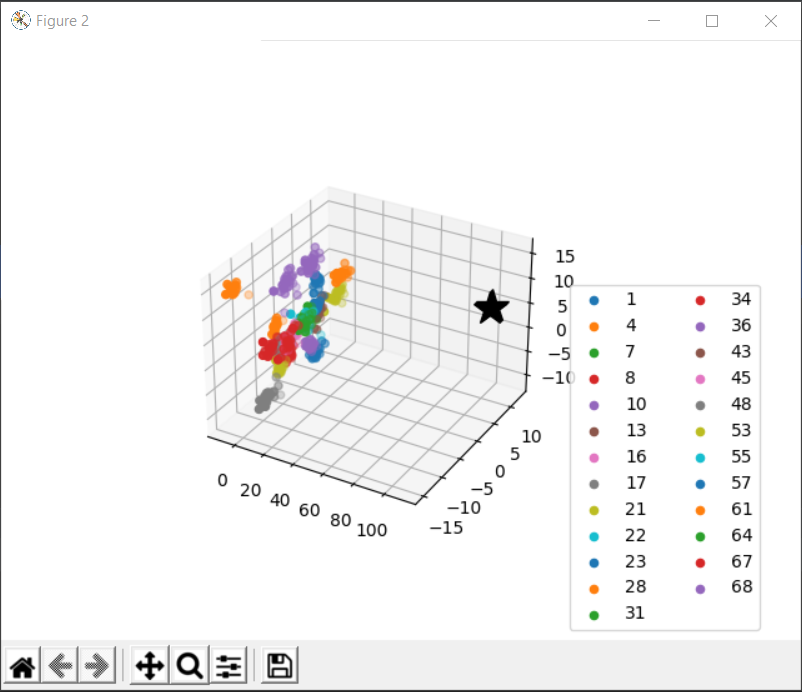


Figure 6. Distribution of sampled data (3D)

Similar to Figure 5, we can observe that my selfie images are clearly separated from the training clusters.

## **Classification Results**

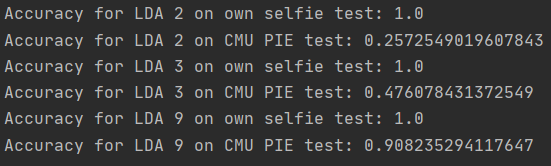


Figure 7. Classification Results (LDA)

As shown in Figure 7, classification accuracy of my selfie photos at 2, 3 and 9 dimensions are 100% while CMU PIE tests accuracy increases drastically from 25% to 47% and finally 90% for dimension 2,3 and 9 respectively. The increase in accuracy for CMU PIE dataset could be because with more dimensions, the data can be projected more distinctly with results in a more accurate classification.

# **GMM**

## **Clustering Results**

### Raw Image

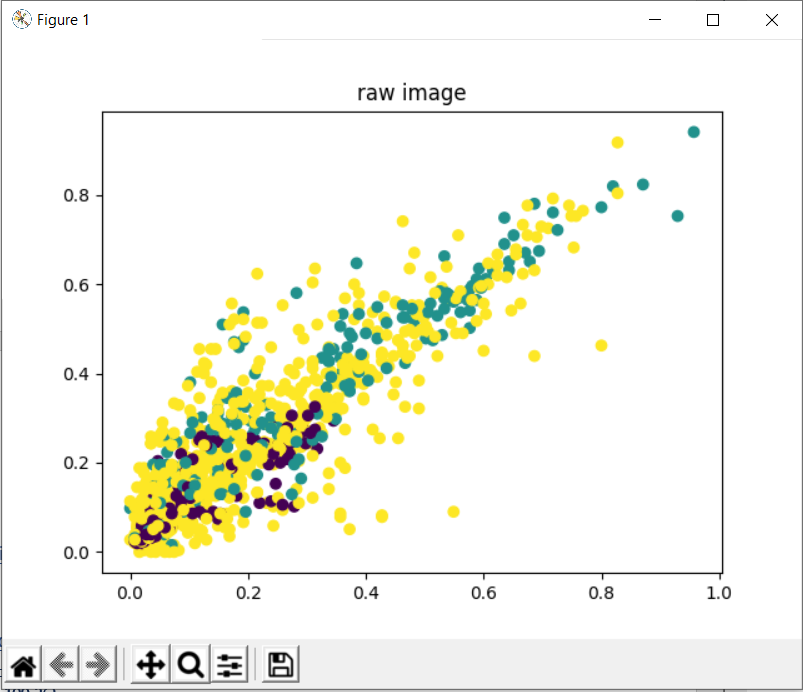


Figure 8. GMM (Raw

### PCA 80

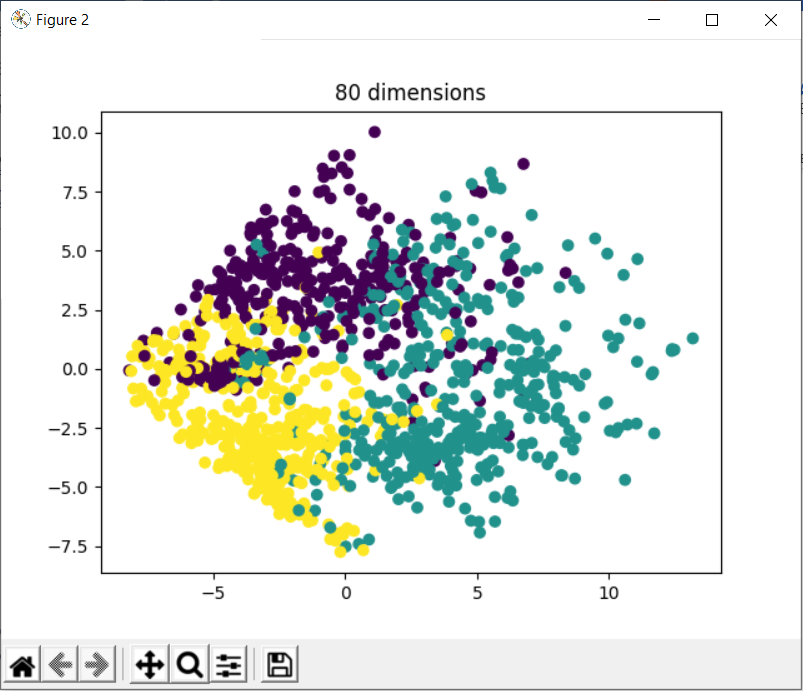


Figure 9. GMM (PCA 80)

### PCA 200

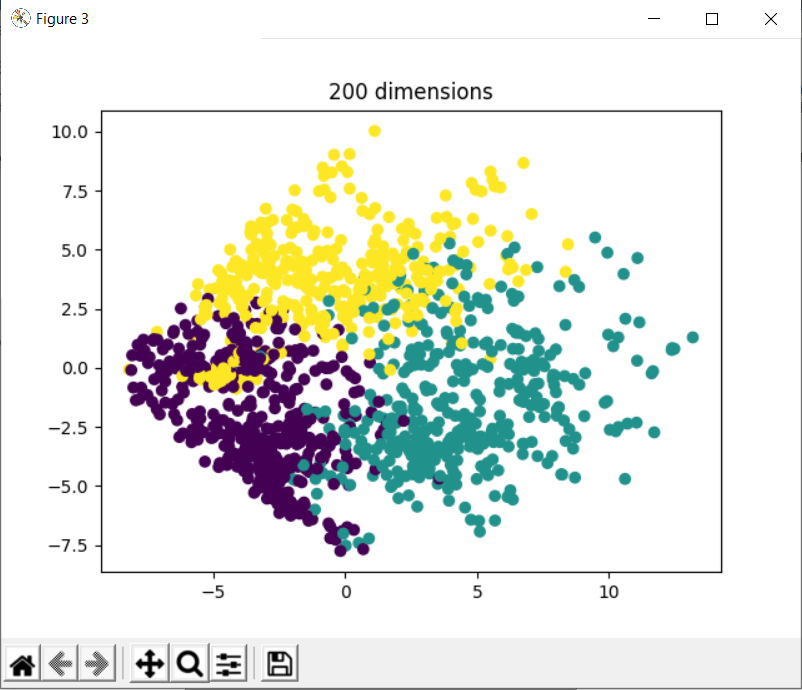


Figure 10. GMM (PCA 200)

By observing figure 8, 9 and 10, we can see that using GMM clustering on the raw image file is not as usual as compared to applying GMM clustering after reducing the image dimension. As seen in Figure 9 and 10, thought the points are still somewhat clustered together, we can observe 3 distinct clusters.

# **SVM**

## **Classification Results**

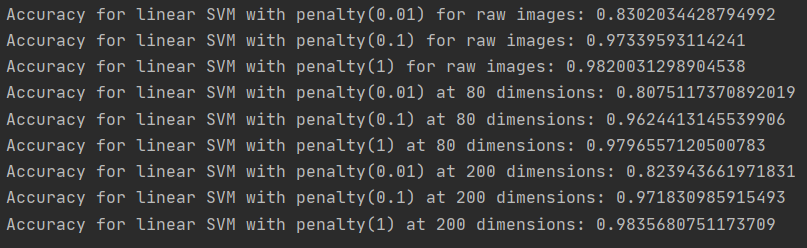


Figure . Accuracy for Linear SVM console output

In Figure 11 above, we observe the console outputs for apply linear SVM on different dimensions as well as penalty parameters C. Table 1 below compiles this information for clarity and ease of comparison. Note that the values are in terms of percentage and truncated to 2 decimal points.

|  |  |  |  |
| --- | --- | --- | --- |
| Dimensionality \ C (%) | 0.01 | 0.1 | 1 |
| Raw | 83.02 | 97.33 | 98.20 |
| PCA 80 | 80.75 | 96.24 | 97.96 |
| PCA 200 | 82.39 | 97.18 | 98.35 |

Table . Compiled Accuracy Results

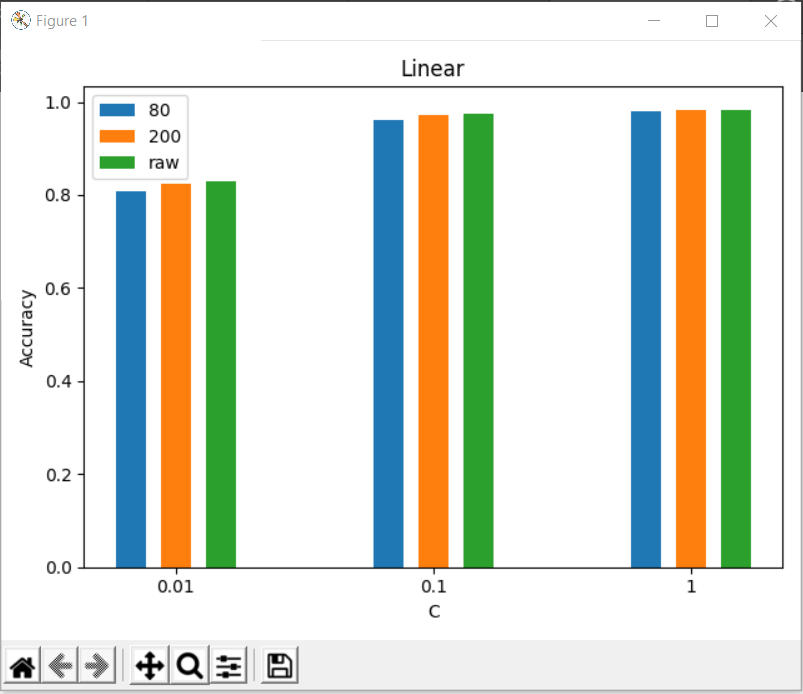


Figure . Bar chart of SVM accuracy vs penalty parameter (C)

Based on Table 1, Figure 12 is derived for better visualization. As observed in Figure 12, we have applied SVM on 3 different dimensions (raw, PCA 80 and PCA 200) as well as different penalty parameters (0.01, 0.1 and 1).

## **Discuss effect of data dimension and penalty parameter C**

From Figure 12, we can see that applying Linear SVM for the raw images as well as reduced dimensions of 80 and 200 does not make a large difference to the result. This could be because the performance bounds on which the maximal margin classifier are independent of the dimension of the feature space, but instead depending on the margin. As such, I believe that Linear SVM is robust to the curse of dimensionality and does not severely affect the accuracy when doing a prediction.

The penalty parameter C tells the linear SVM optimization how much to avoid misclassifying each training data. For larger values of C, the optimization would choose a smaller-margin hyperplane if it can do better at getting the training points classified correctly. Conversely, a smaller value of C would cause the optimizer to look for a larger-margin hyperplane at the cost of possibly misclassifying more points. As shown in Figure 12, a smaller penalty parameter of 0.01 does not yield better results than a penalty parameter of 1.

With the high accuracy results, we should be able to conclude that the current training and test dataset is a linear problem, and we are indeed using the correct kernel and penalty parameter.

# **CNN**

The CNN is trained with two convolutional layers and one fully connected layer with the architecture as follows: number of nodes: 20-50-500-26. The number of nodes in the last layer is fixed at 26 as we are performing 26-category (25 CMU PIE faces plus 1 selfie) classification. The convolutional kernel sizes are set as 5. Each convolutional layer us followed by a max pooling layer with a kernel size of 2 and stride of 2. The fully connected layer is followed by ReLU. The model is also iterated over 20 epochs.

## **Final Classification Performance**

For the final classification performance, we will be classification 3 different network architectures. The first architecture would be number of nodes are 20-50-500-26 as required in the original question. Second, we would adjust the dense layer from 500 to 100. The third change would have the same number of nodes as the first but we will change the max pooling pool size of 10 instead of 2.

### **Network Architecture 1(20-50-500-26)**

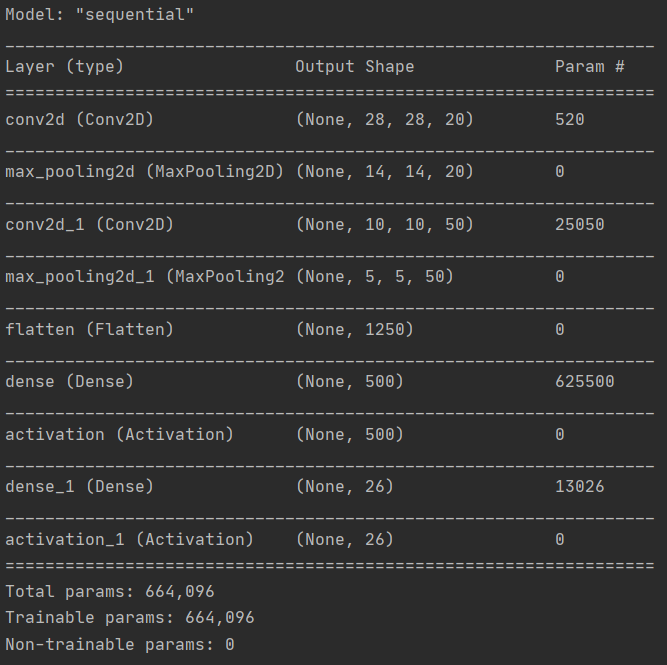


Figure . Model summary (1)

Figure 13 above shows the model summary of the CNN model. As shown in Figure 14 below, this current model is capable of yielding 98.35% accuracy and 5.6% of loss with the test data.



Figure . Classification Performance (1)

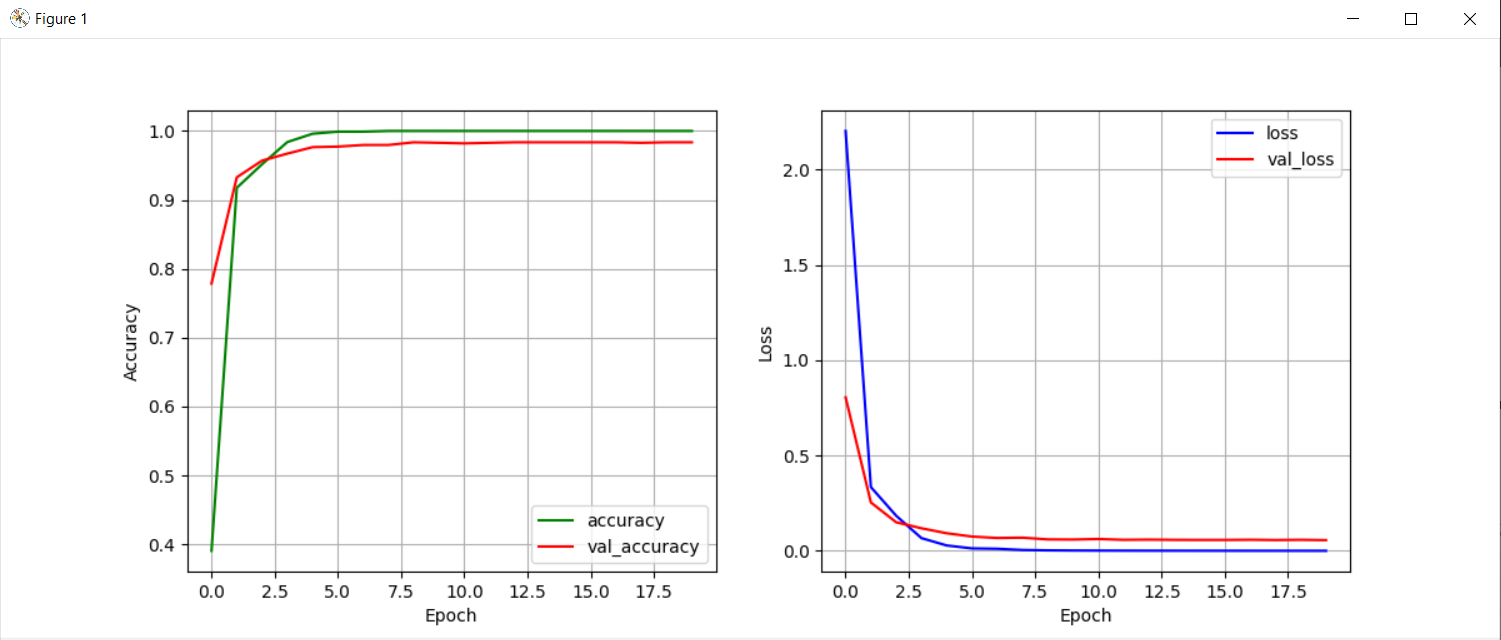


Figure . Accuracy and loss graphs against epochs (1)

By observing Figure 15, we can see that as epoch increases, the accuracy of the model increases and the loss of the model decreases. This is because the neural network learns via back propagation and updates the weights in each layer. As observed in Figure 15, the validation loss is already saturated at around 7-8 epochs, which may be the ideal epoch number for this model. If we increase the number of epochs and observe an increase in loss, it would signify an overfitting of the model.

### **Network Architecture 1(20-50-100-26)**

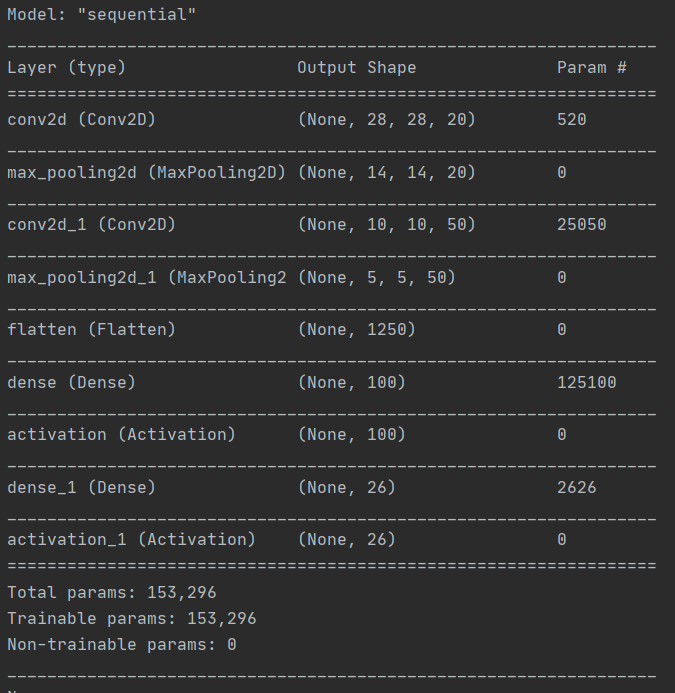


Figure . Model summary (2)

As shown in Figure 16, we have changed the Dense layer from 500 to 100. As observed in Figure 17, the test accuracy has decreased slightly while the test loss increased slightly by 2%. The dense layer feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer. By decreasing the number of neurons, the model gets less adaptive and becomes less capable of learning the smaller details in the images, which may explain the small increase in losses and decrease in accuracy.



Figure . Classification Performance (2)

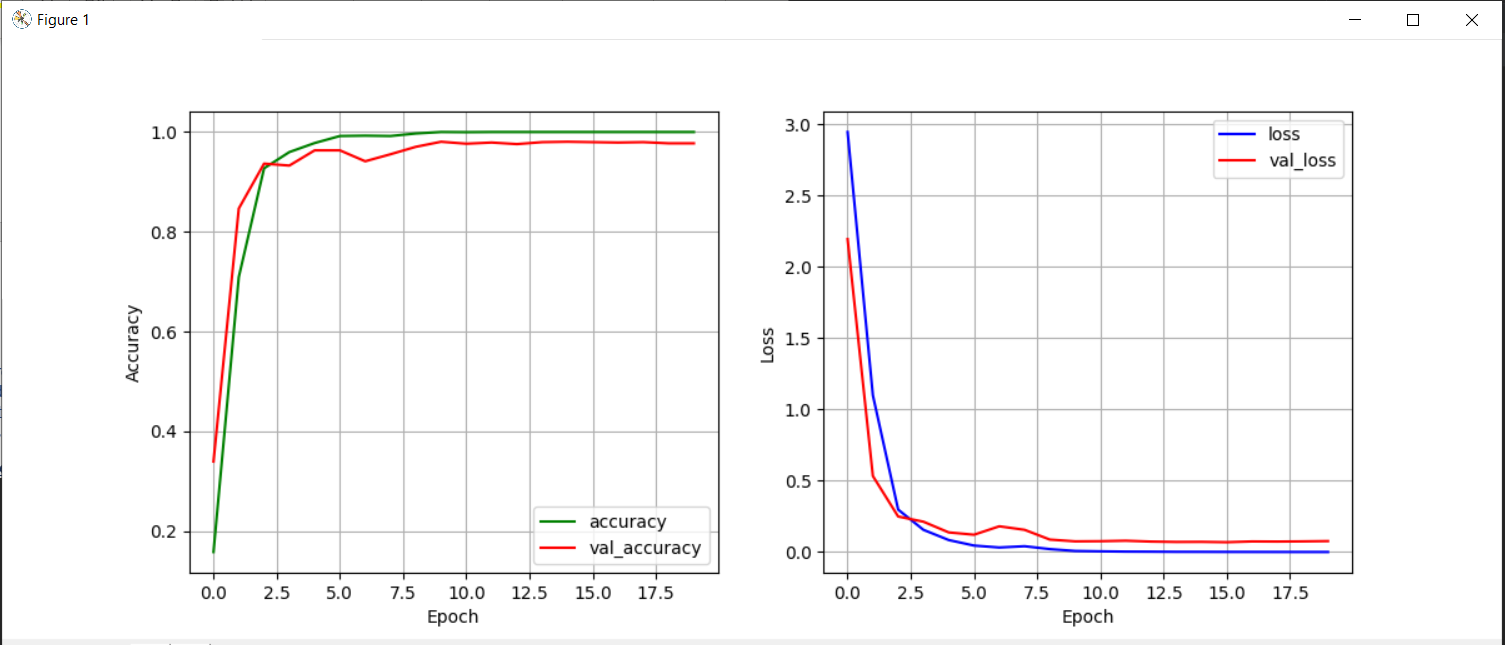


Figure . Accuracy and loss graphs against epochs (2)

Figure 18 depicts the accuracy and loss graphs against the epochs. When compared to Figure 15, we can observe that it is more unstable between the fifth and eighth epoch but eventually saturates at its loss and accuracy.

### **Network Architecture 1(20-50-500-26, Pool size=10)**

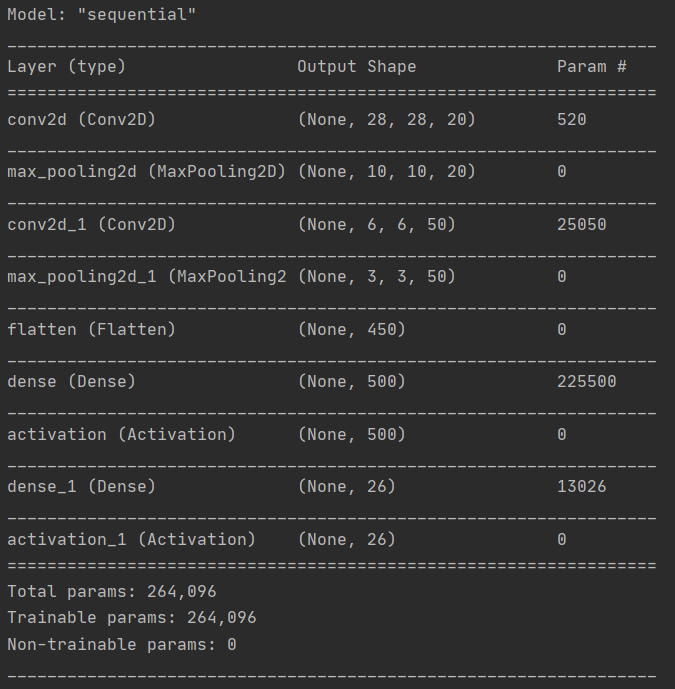


Figure . Model Summary (3)

As shown in Figure 19, the max\_pooling2d shaped has changed after changing the pool size from 2 to 10. By observing the results in Figure 20, we observed a lower accuracy as well as higher loss.



Figure . Classification Performance (3)

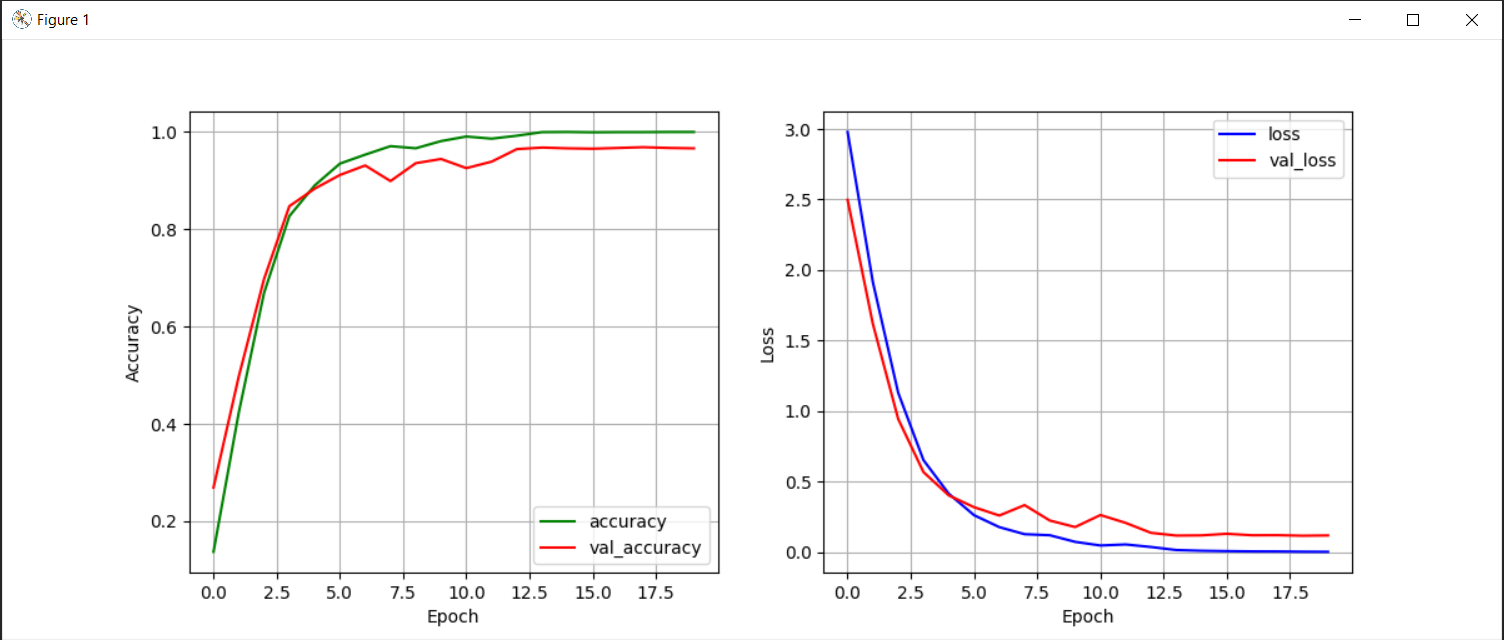


Figure . Accuracy and loss graphs against epochs (3)

Figure 21 shows that model requires even more epochs to stabilize while having small increases in losses as several epochs.