Spectacle Recommendation

Team members:

P. Bhavesh – CB.EN.U4CSE21609

Mohamed Aasil S – CB.EN.U4CSE21638

P. Joel – CB.EN.U4CSE21649

M. Shenthan – CB.EN.U4CSE21657

Y. Sri Bhargav – CB.EN.U4CSE21670

Simple
Neural Network

How does it work?

- The network learns by adjusting the weights and biases associated with each neuron during training. Training involves feeding the network with labeled data and minimizing the difference between predicted and actual outputs. Neurons in hidden layers use activation functions to introduce non-linearity into the network.

- Here, we first create classes (shapes) and then pass them through the neurons, which would create certain weights and intermediary variables, using which they'd capture the relationship between the members of a certain class.

- When passed a new image, the model would then measure the weights and then match them to the closest class

About the Dataset?

- There are about 8000 images for training (all labelled) and 1000 for testing (here also we're using labelling, to calculate performance metrics later)

- Pre-processing:

Image to be at least 224x224 pixels while maintaining its aspect ratio; would be upscaled if ends up smaller than original;

We normalize the arrays in which images are stored to scale the pixel values to the range of -1 to 1. normalized by dividing each pixel value by 127.5 and subtracting 1.

Each image is a 4-dimensional array with dimensions (1, 224, 224, 3) denoting (N.f images, H, W, N.f channels (3 for RGB)) respectively; and also the datatype of the arrays is float32, which is a 32-bit floating-point number.

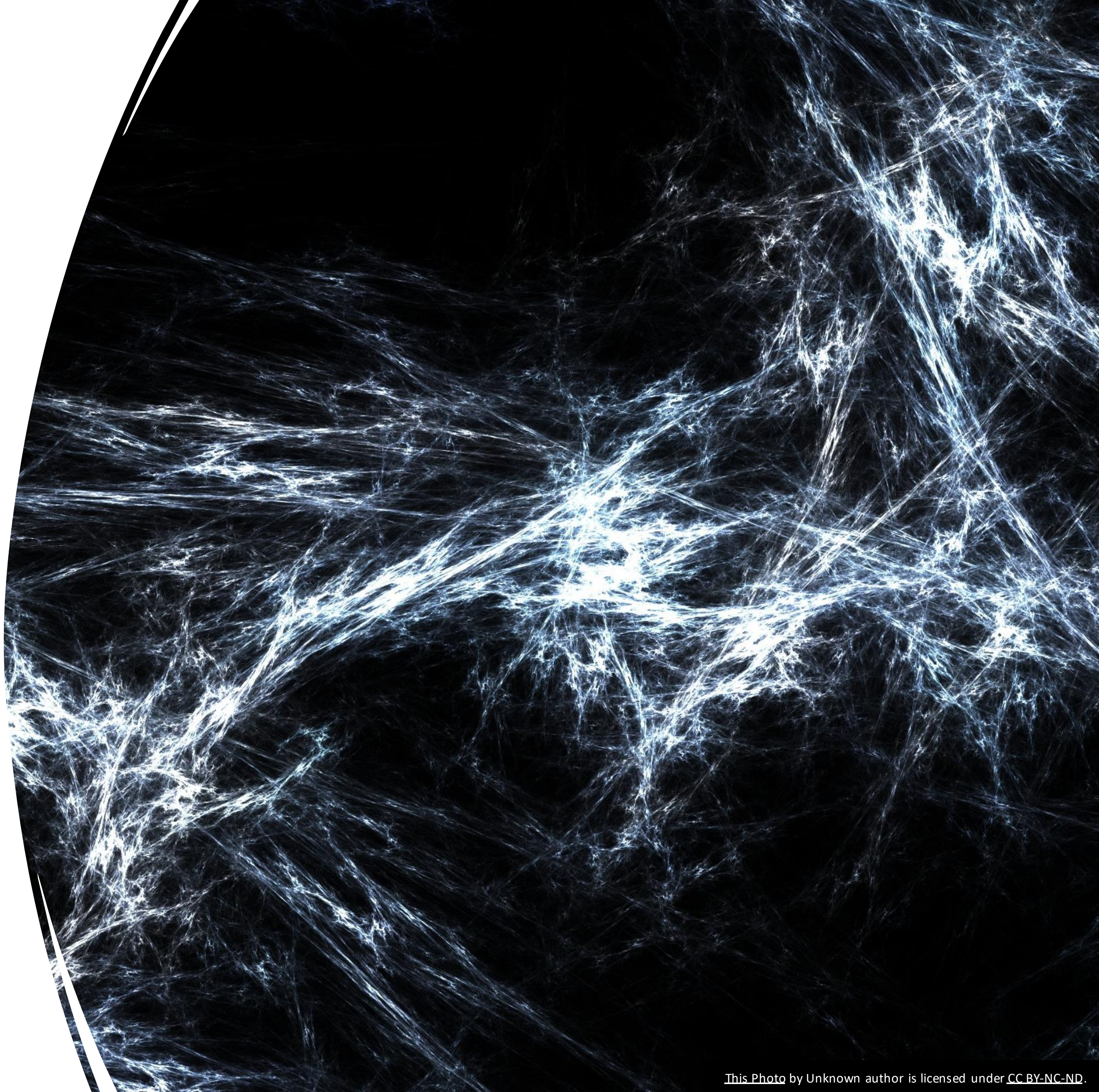
Specifics about the Model:

- There are in total 3 hidden layers: 2 Convolutional and 1 Dense Layer.

convolutional layers with 32 and 64 filters respectively, followed by max pooling layers. The flattened output is then connected to a dense layer with 5 units and a softmax activation function for multi-class classification. This model architecture is commonly used for image recognition tasks.

We do maxpooling in between each layers in order to down sample the images being passed among the layers

Sequential CNN 1



About the dataset:
(We have a total of 2140 input datapoints)

Moreover, the input dataset is in such a way that the X(features) are images but, represented in numerical values.(These numerical values represent the intensity of each pixel ranging from 0 - 255) and the image resolutions are of size 96 x 96.

So, first we try to store all these images in a single 3-D dimensional array (2140,96,96).

Then we split the datapoints into train and test in the ratio 90-10.

About the model:

1. A sequential CNN is where the layers are stacked one upon the another with only one input tensor and one output tensor. (A tensor is a multi-dimensional array that is used to store input values, model learning parameters, or intermediate values).

2. So, here we are using a sequential CNN to predict the x & y co-ordinates of different face landmarks.(face landmarks are like points on your face that can be used for differentiating human beings)

3. Here, we use these parameters to calculate distance between eyes, distance between the cheek bones etc. So, that they can be furtherly used for recommending spectacles for a person.

NOTE:
Even though there is already a built-in module for detecting face landmarks we have built the model from scratch and are using it. But, in further use cases we will try to increase the model accuracy.(Right now the accuracy is around 90).

More about the model:

We have used an initializer for biases.(Initializers act as starting point for the model parameters)

We used 7 layers (4 - convolutional layers, 3 - Dense layers).

Filters help in recognizing the underling patterns in the image and they learn something from the inputs.

The convolutional layer - 1 has 256 filters each of size (1, 1). A Rectified Linear Unit has been used as activation function

The convolutional layer - 2 has 128 filters each of size (2, 2). =The convolutional layer - 1 has 256 filters each of size (1, 1).

The convolutional layer - 3 has 64 filters each of size (3, 3). Similarly, for layer 4 32 filters are used each of size (4, 4).

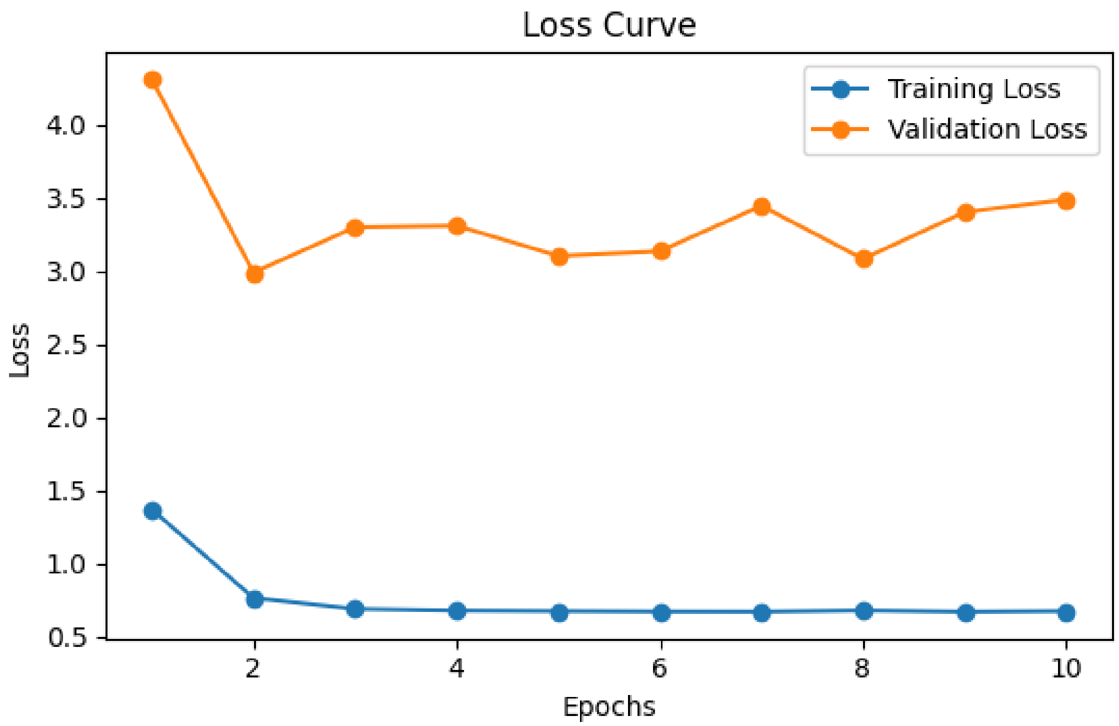
Max Pooling technique has been used to down sample the images passing from one Convolutional layer to the other

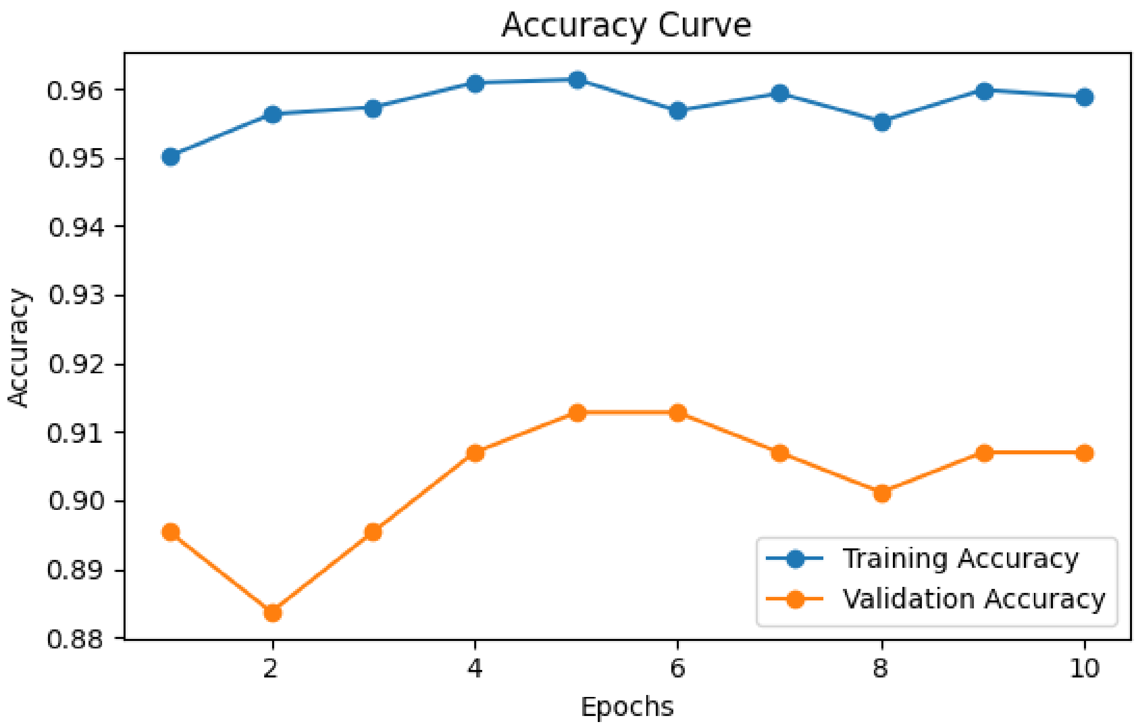
Dense layers are fully connected neural networks that means each layer is connected to every other layer and three Dense layers are used to predict the final outputs.

INFERENCE:

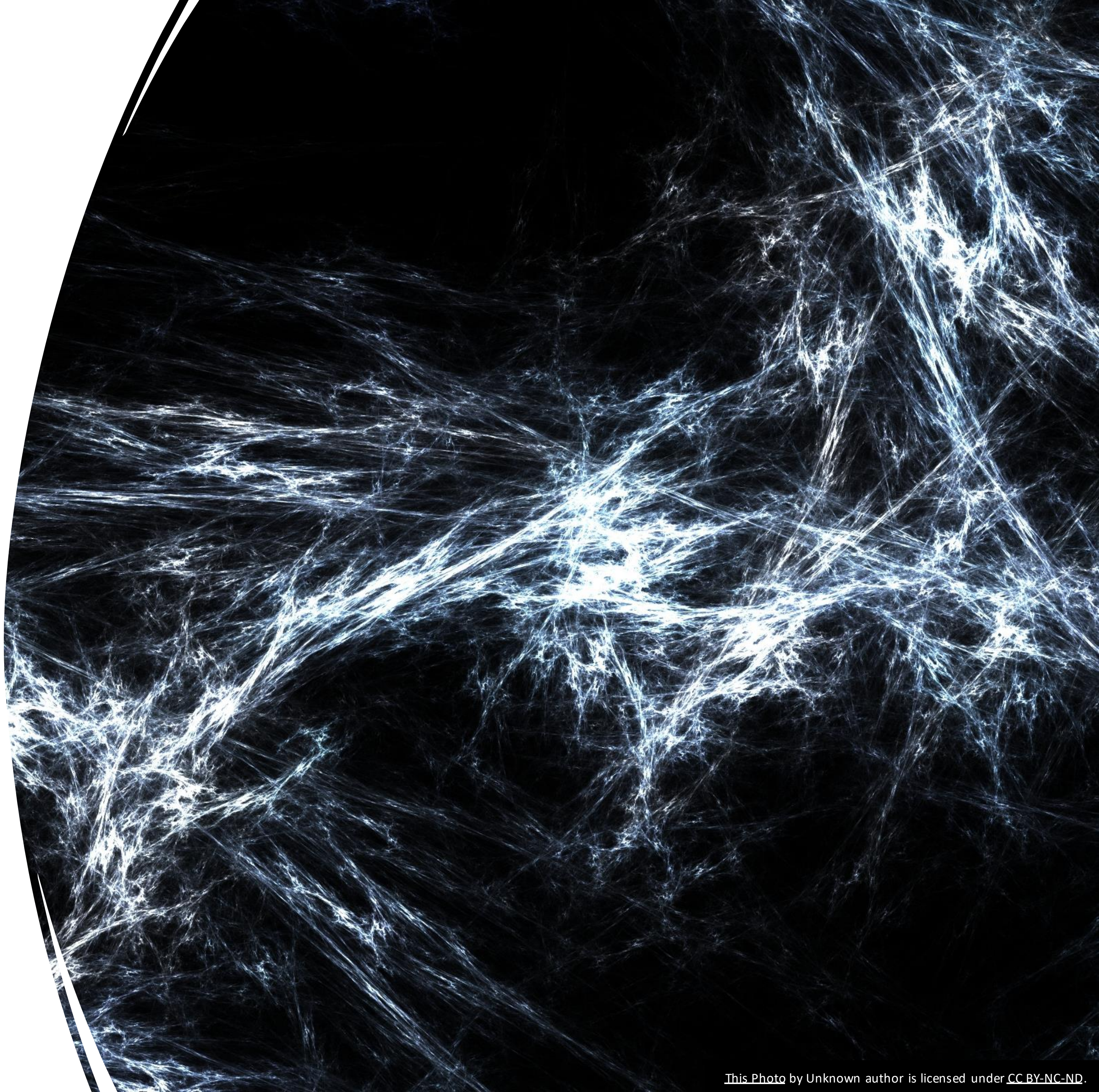
If you look at the loss curve (left side) we can infer if there is any overfitting. Generally, as training loss decreases if validation loss increases then there might be a good chance that your model is overfitting. Here it is not the case

- If we look at the accuracy curve as epochs increase, model accuracy is just fluctuating between 90 – 96 which means it is not reaching a final optimal solution which can be due to many reasons.(that means there is overfitting)





Sequential CNN 2



Preprocessing:

The preprocessing function combines the following steps for the input image:

Grayscale Conversion:

- The grayscale function converts the input image from color (RGB) to grayscale using OpenCV's cvtColor function with the cv2.COLOR_BGR2GRAY conversion.

Histogram Equalization:

- The equalize function applies histogram equalization to the grayscale image using OpenCV's equalizeHist function. Histogram equalization enhances the contrast of the image.
-

Normalization:

- After grayscale conversion and histogram equalization, the pixel values of the image are normalized to the range [0, 1]. This is done by dividing the pixel values by 255.

Model Architecture:

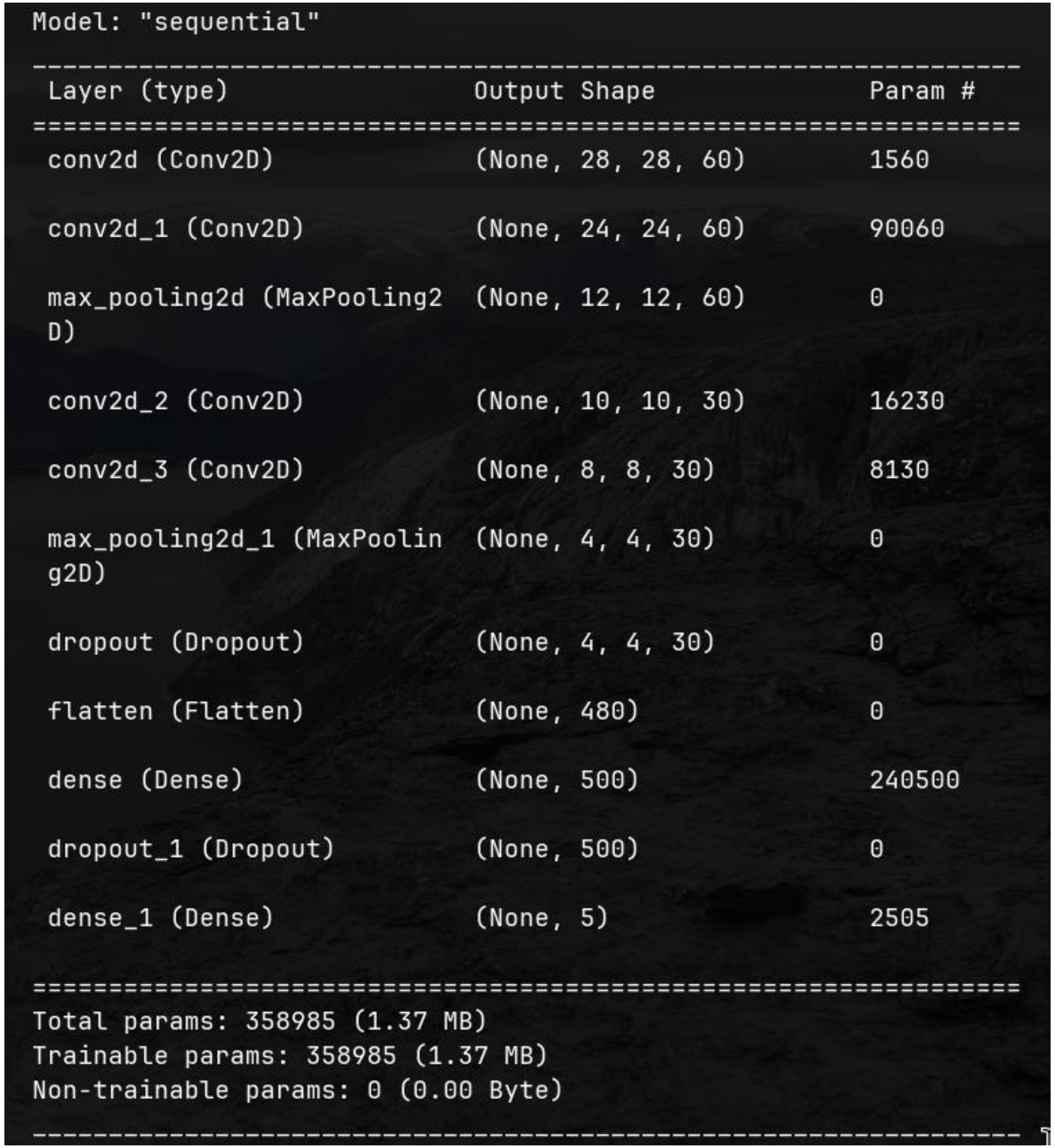
- The CNN model is defined using the Sequential API from Keras.
- It consists of convolutional layers with rectified linear unit (ReLU) activation, max pooling layers, and dropout layers for regularization.
- The model architecture is designed to capture hierarchical features from the images.

Model Training:

- The model is compiled using the Adam optimizer and categorical crossentropy loss.
- It is trained using the fit method on the training data, with validation data for monitoring the model's performance during training.

Training Parameters:

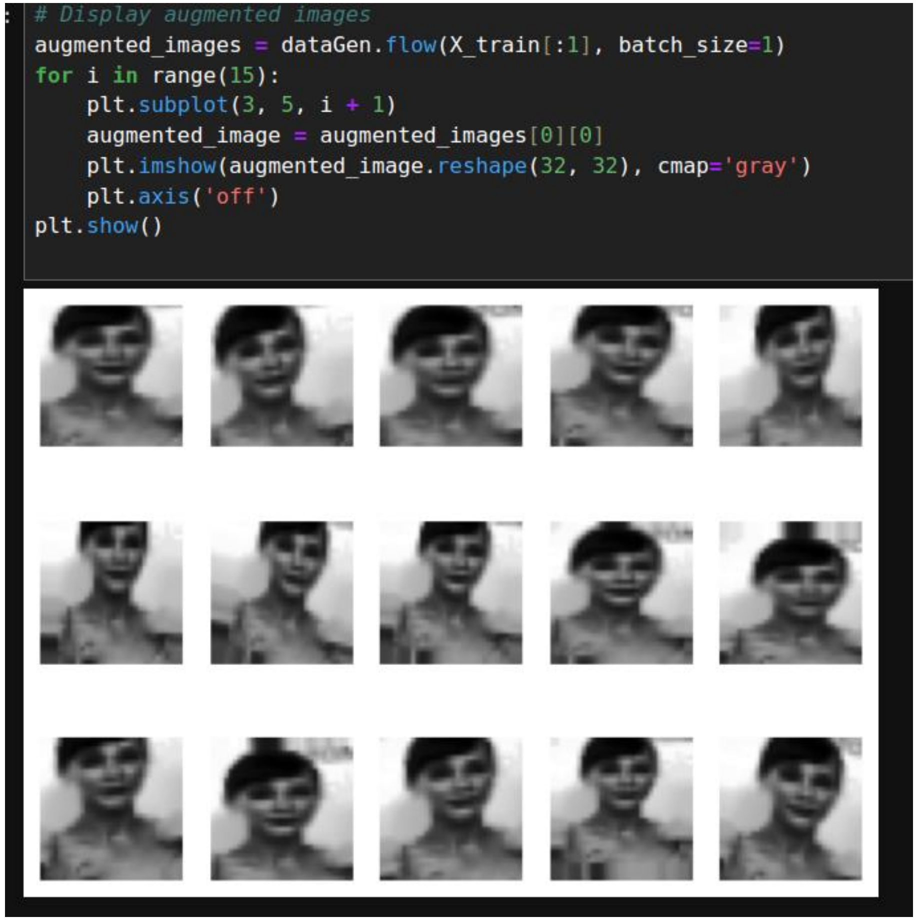
- The model is trained with a batch size of 50.
- Training occurs over 30 epoch.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 60)        1560

 conv2d_1 (Conv2D)           (None, 24, 24, 60)        90060

 max_pooling2d (MaxPooling2  (None, 12, 12, 60)        0
 D)

 conv2d_2 (Conv2D)           (None, 10, 10, 30)        16230

 conv2d_3 (Conv2D)           (None, 8, 8, 30)          8130

 max_pooling2d_1 (MaxPoolin  (None, 4, 4, 30)          0
 g2D)

 dropout (Dropout)           (None, 4, 4, 30)          0

 flatten (Flatten)           (None, 480)               0

 dense (Dense)               (None, 500)               240500

 dropout_1 (Dropout)         (None, 500)               0

 dense_1 (Dense)             (None, 5)                 2505

=================================================================
Total params: 358985 (1.37 MB)
Trainable params: 358985 (1.37 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

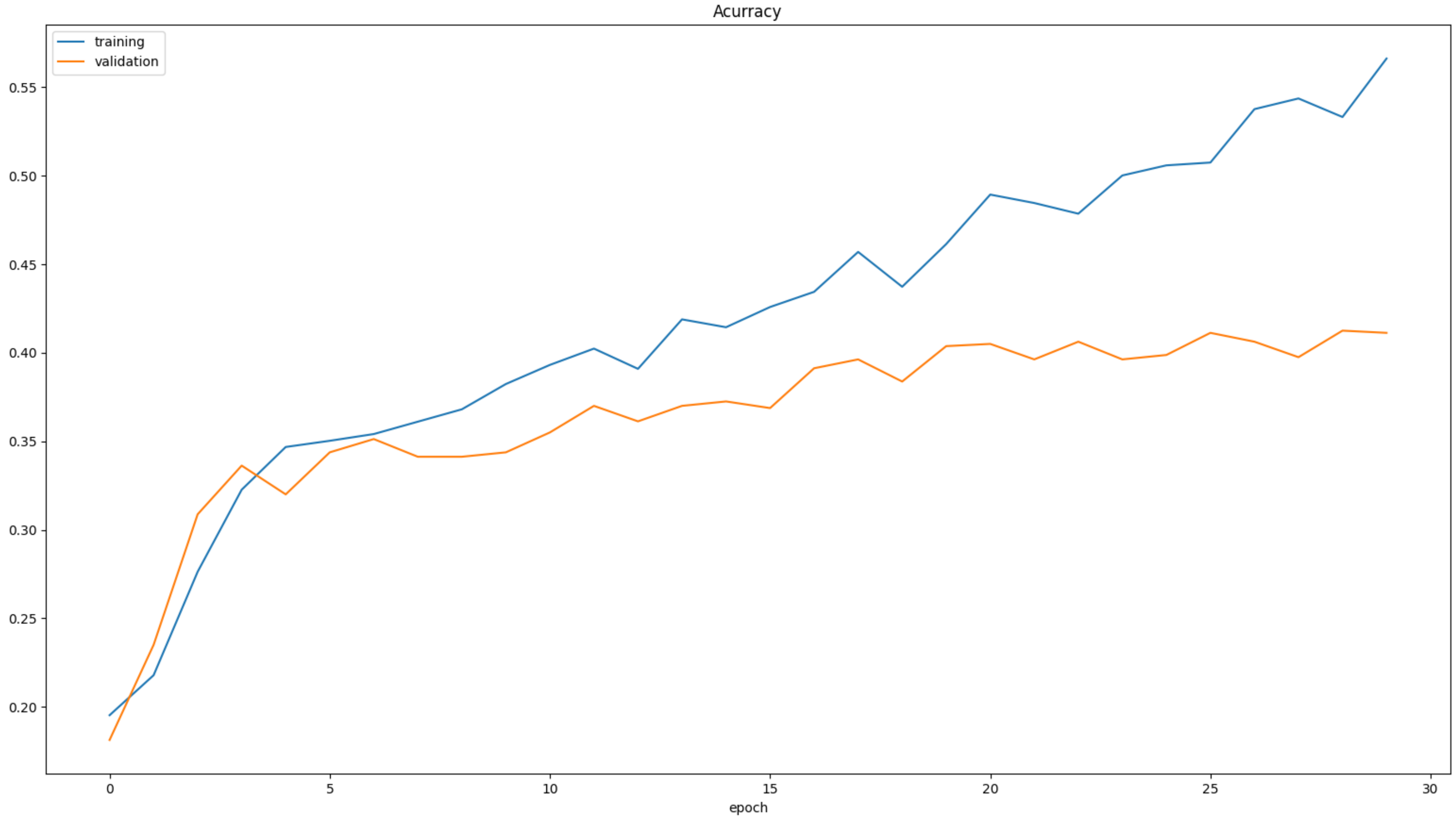
Model summary

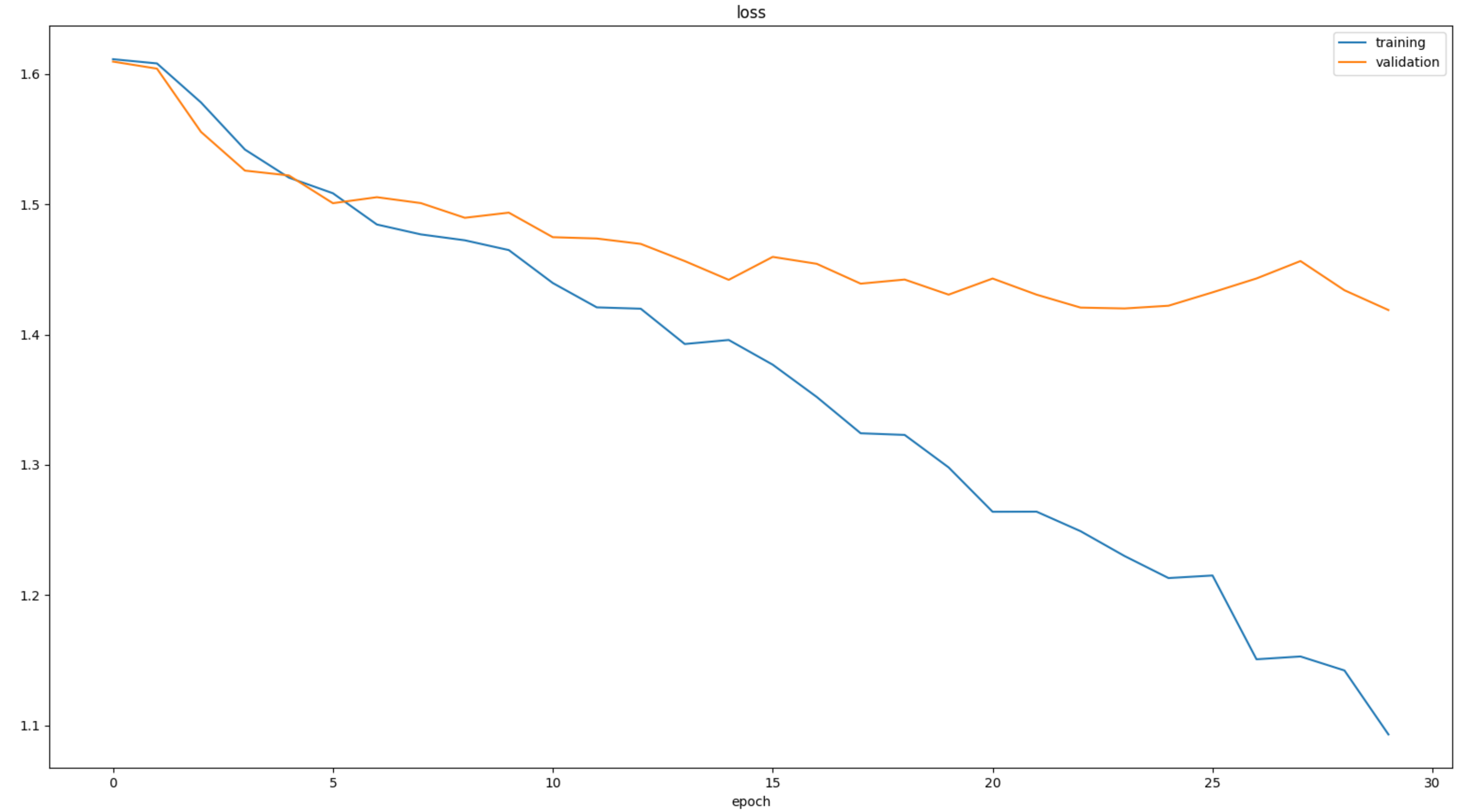
Augumentation

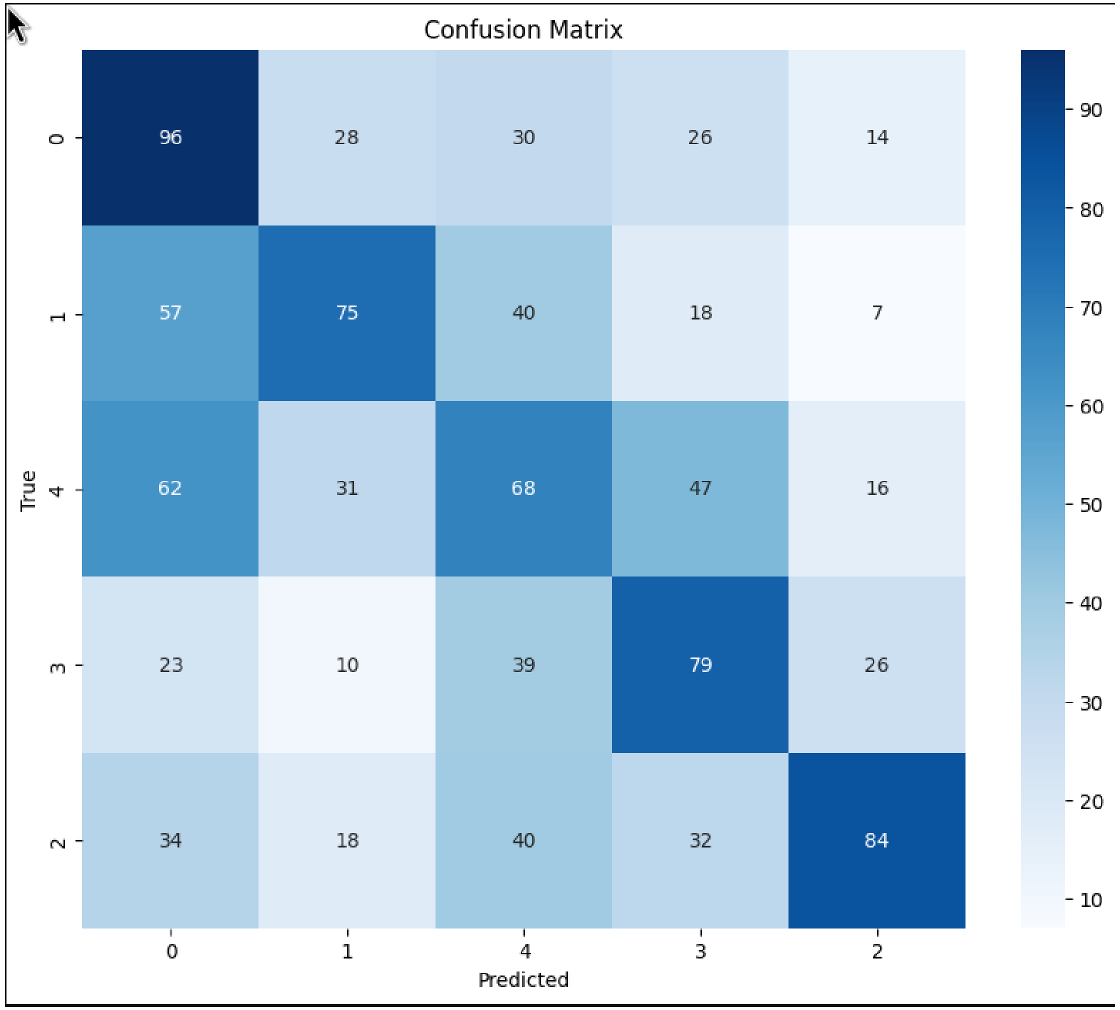


```
# Display augmented images
augmented_images = dataGen.flow(X_train[:1], batch_size=1)
for i in range(15):
    plt.subplot(3, 5, i + 1)
    augmented_image = augmented_images[0][0]
    plt.imshow(augmented_image.reshape(32, 32), cmap='gray')
    plt.axis('off')
plt.show()
```

Visualization of training and validation

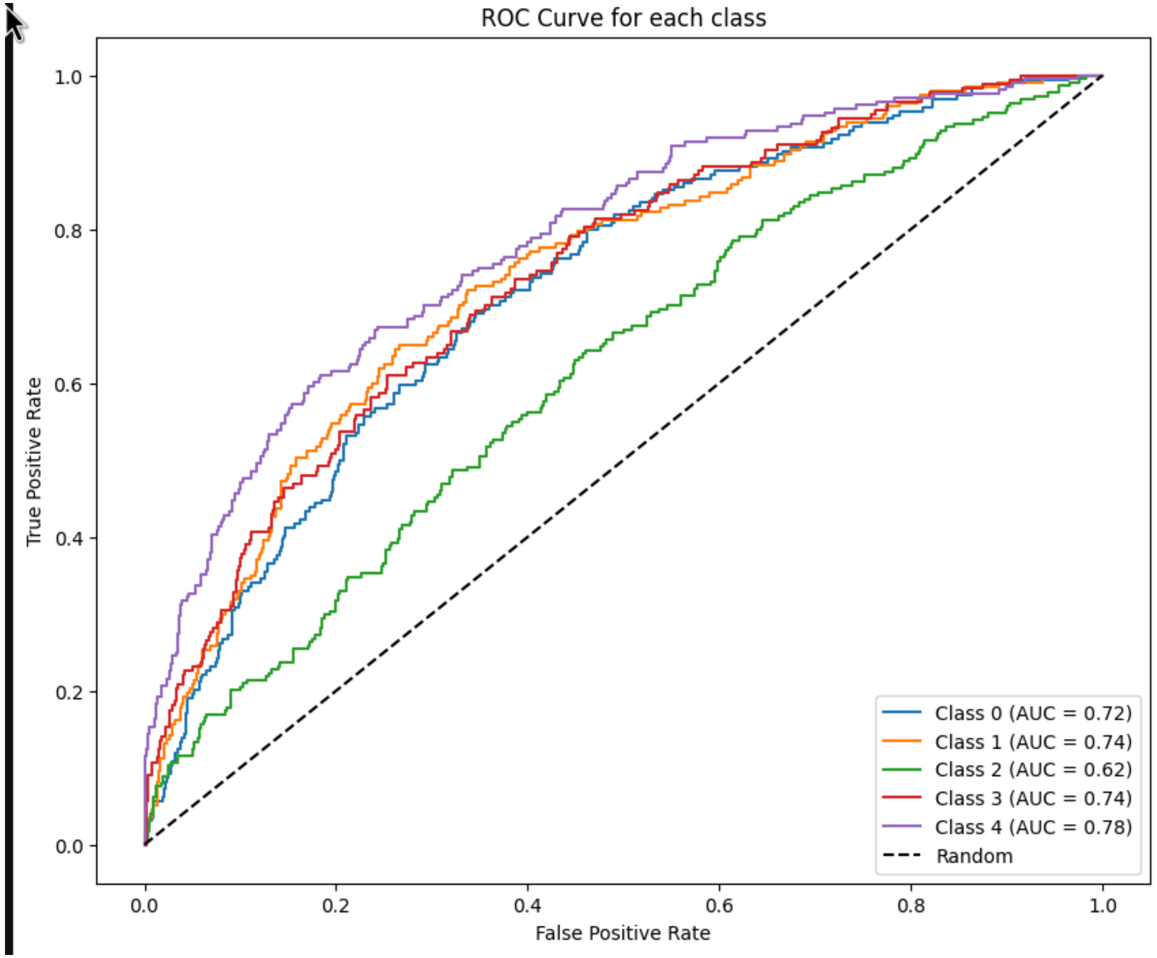



loss
training
validation
1.6
1.5
1.4
1.3
1.2
1.1
0
5
10
15
20
25
30
epoch

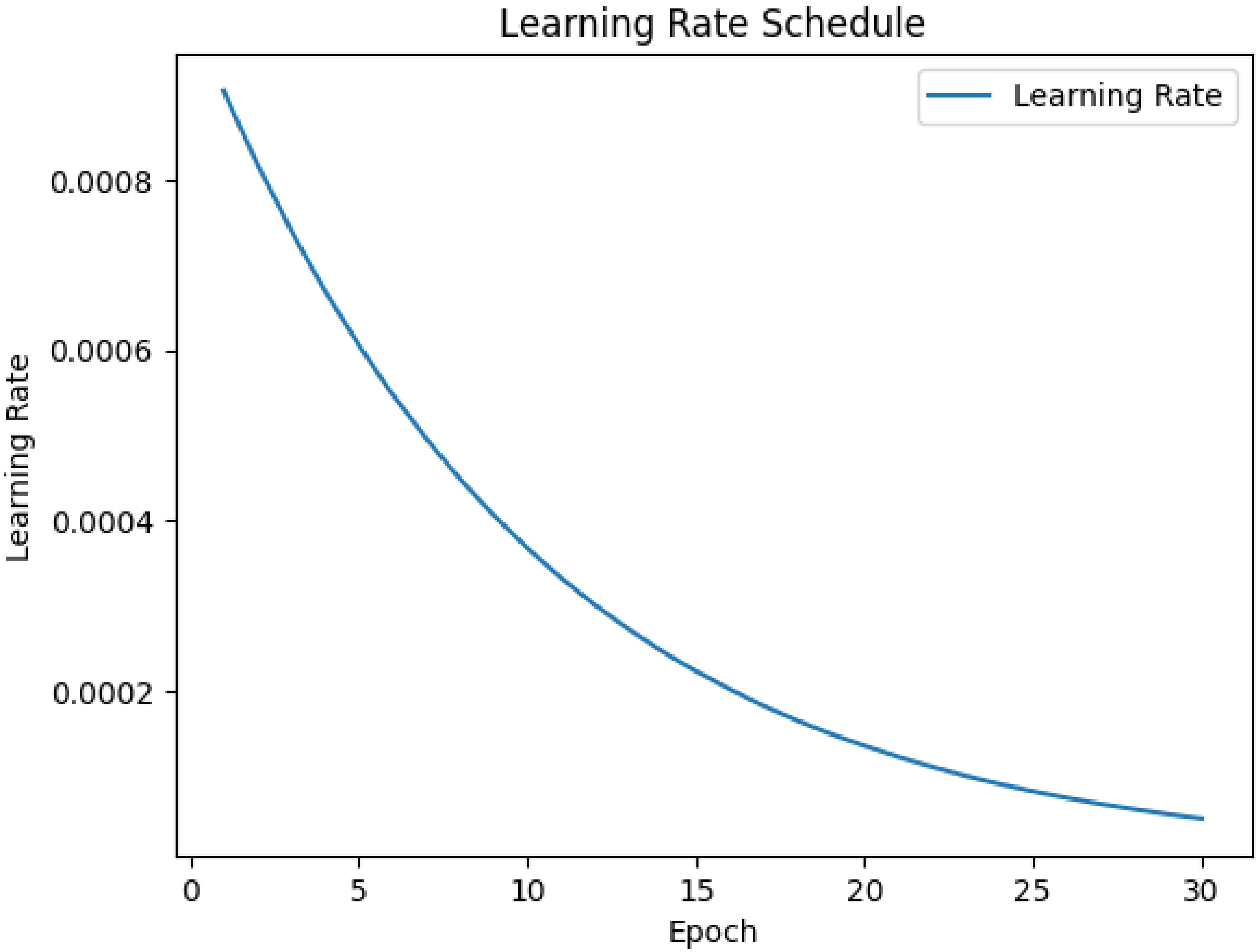


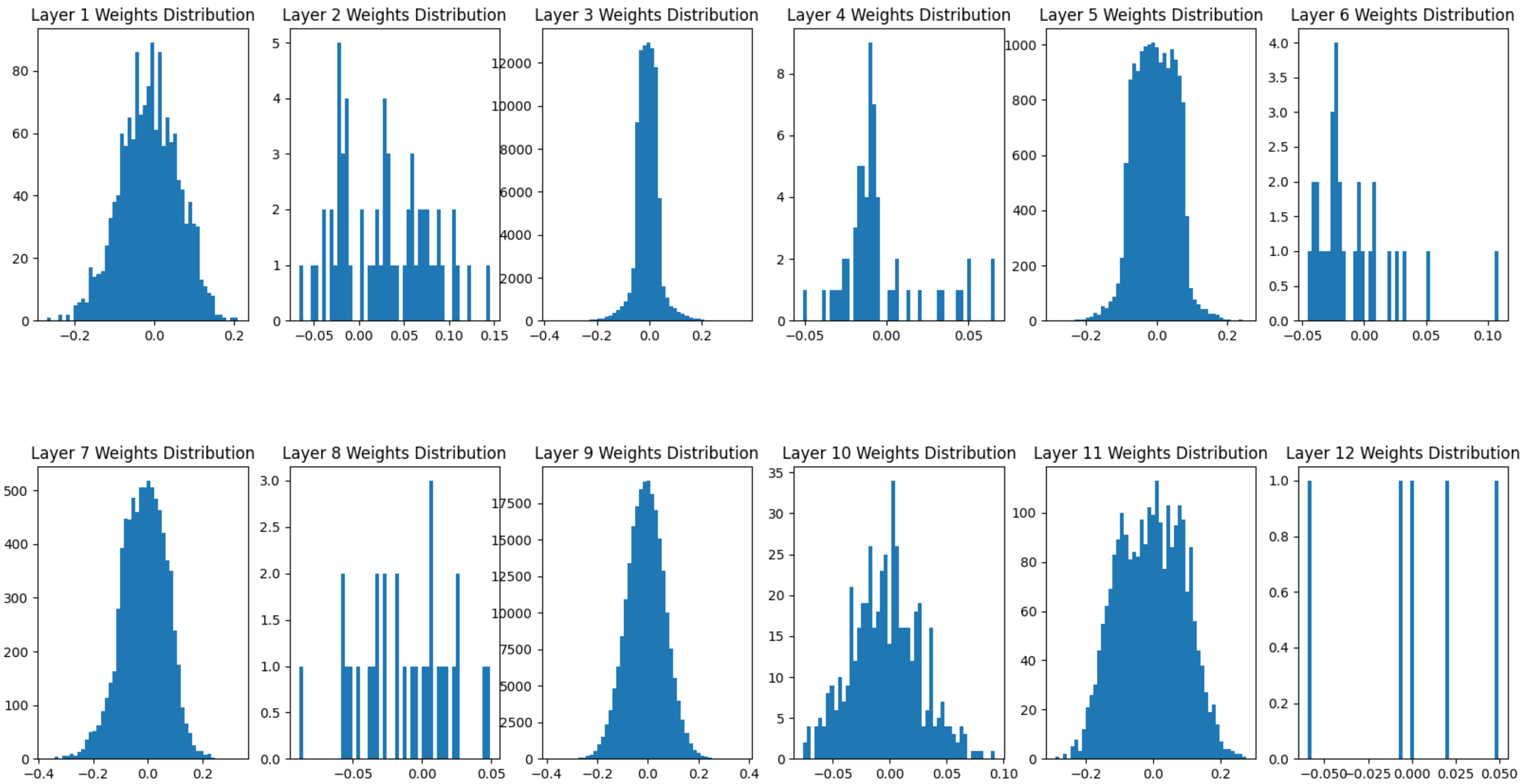
Confusion Matrix

True \ Predicted	0	1	4	3	2
0	96	28	30	26	14
1	57	75	40	18	7
4	62	31	68	47	16
3	23	10	39	79	26
2	34	18	40	32	84



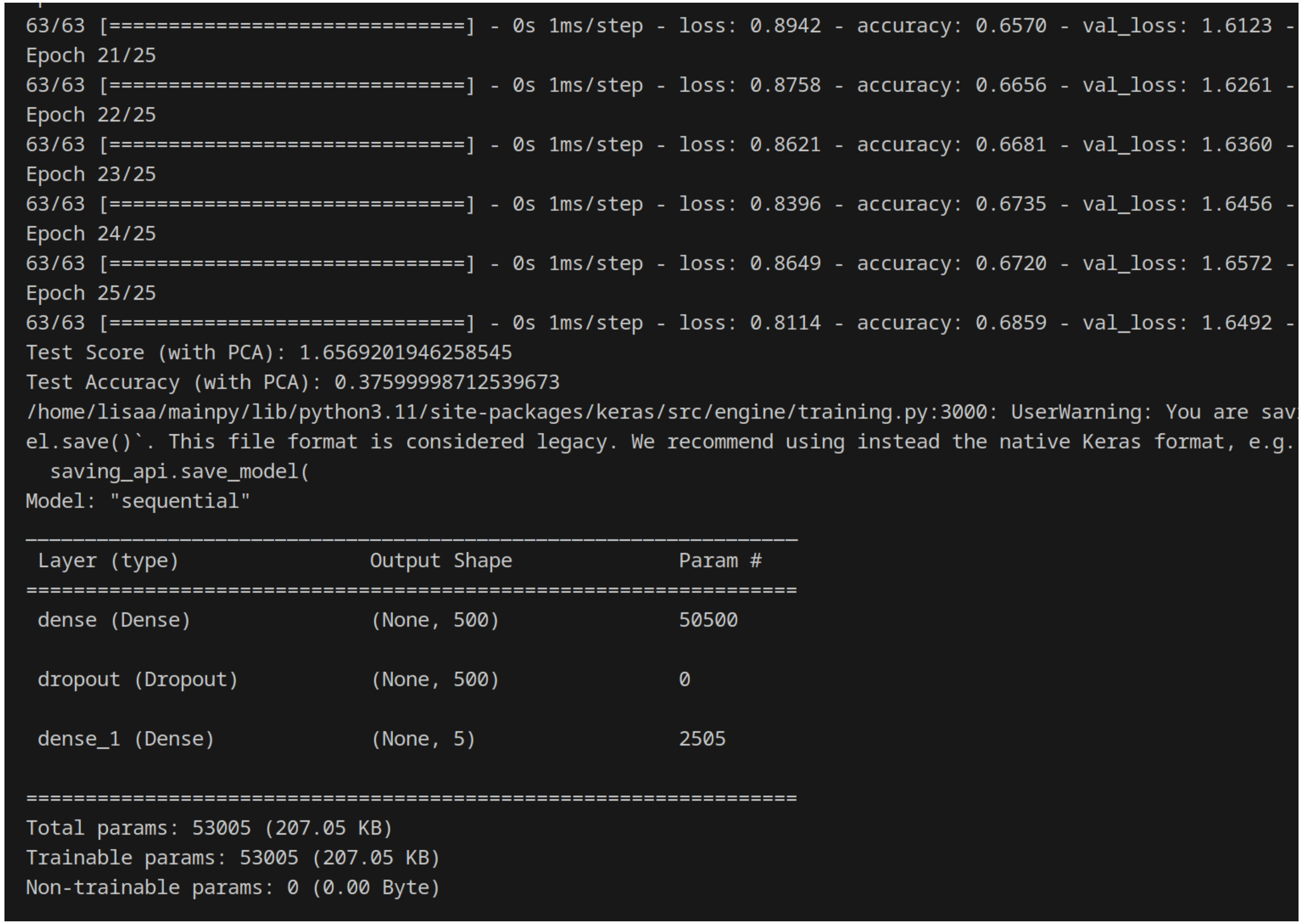
ROC Curve for each class


Learning Rate Schedule
Learning Rate
0.0008
0.0006
0.0004
0.0002
Learning Rate
0
5
10
15
20
25
30
Epoch


Layer 1 Weights Distribution
0
20
40
60
80
−0.2
0.0
0.2
Layer 2 Weights Distribution
0
1
2
3
4
5
−0.05
0.00
0.05
0.10
0.15
Layer 3 Weights Distribution
0
2000
4000
6000
8000
10000
12000
−0.4
−0.2
0.0
0.2
Layer 4 Weights Distribution
0
2
4
6
8
−0.05
0.00
0.05
Layer 5 Weights Distribution
0
200
400
600
800
1000
−0.2
0.0
0.2
Layer 6 Weights Distribution
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
−0.05
0.00
0.05
0.10
Layer 7 Weights Distribution
0
100
200
300
400
500
−0.4
−0.2
0.0
0.2
Layer 8 Weights Distribution
0.0
0.5
1.0
1.5
2.0
2.5
3.0
−0.05
0.00
0.05
Layer 9 Weights Distribution
0
2500
5000
7500
10000
12500
15000
17500
−0.4
−0.2
0.0
0.2
0.4
Layer 10 Weights Distribution
0
5
10
15
20
25
30
35
−0.05
0.00
0.05
0.10
Layer 11 Weights Distribution
0
20
40
60
80
100
−0.2
0.0
0.2
Layer 12 Weights Distribution
0.0
0.2
0.4
0.6
0.8
1.0
−0.050
−0.025
0.000
0.025
0.050

CNN with PCA

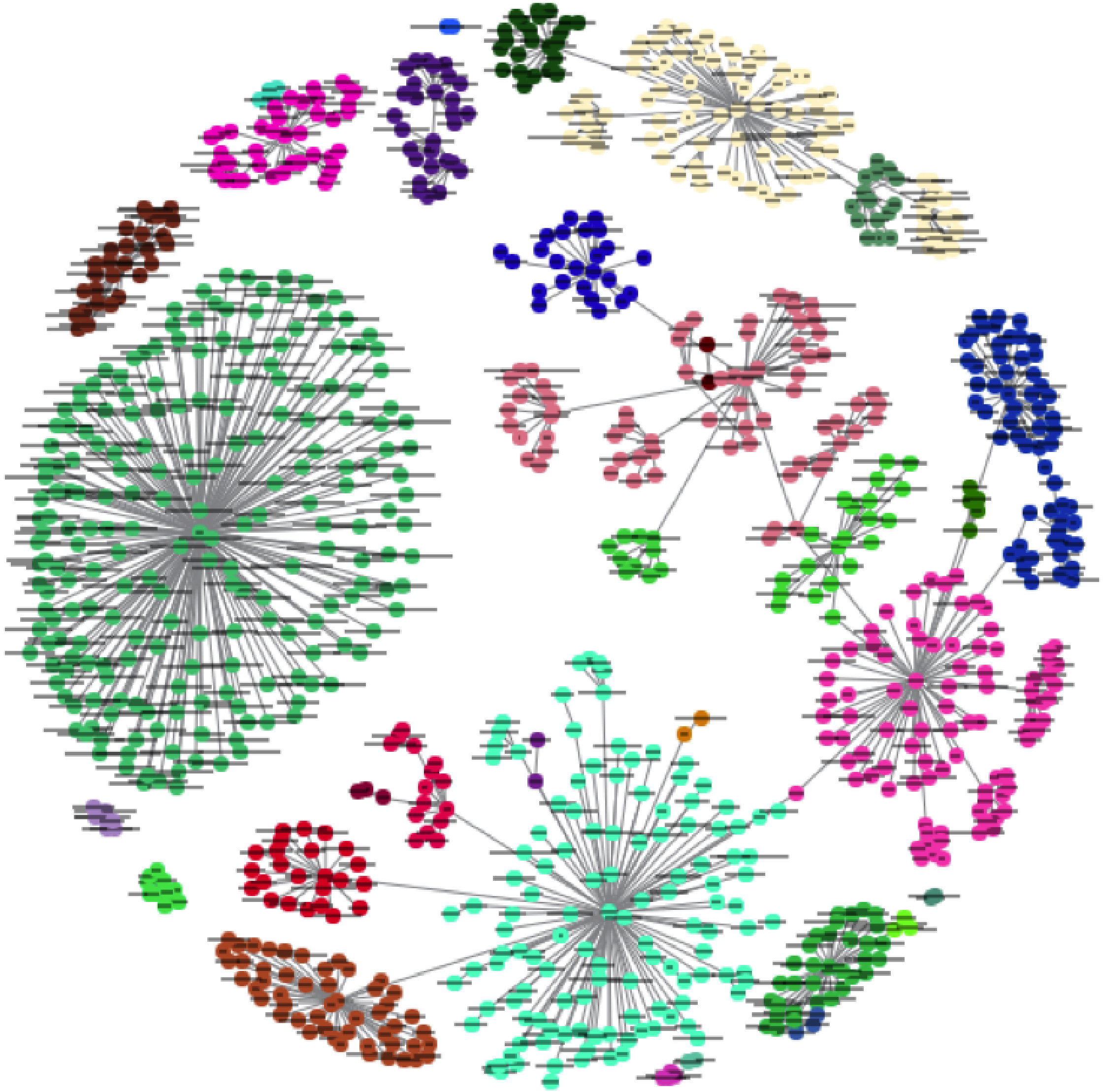
- Less accurate in our case with accuracy of 37% only



```
63/63 [==============================] - 0s 1ms/step - loss: 0.8942 - accuracy: 0.6570 - val_loss: 1.6123 -
Epoch 21/25
63/63 [==============================] - 0s 1ms/step - loss: 0.8758 - accuracy: 0.6656 - val_loss: 1.6261 -
Epoch 22/25
63/63 [==============================] - 0s 1ms/step - loss: 0.8621 - accuracy: 0.6681 - val_loss: 1.6360 -
Epoch 23/25
63/63 [==============================] - 0s 1ms/step - loss: 0.8396 - accuracy: 0.6735 - val_loss: 1.6456 -
Epoch 24/25
63/63 [==============================] - 0s 1ms/step - loss: 0.8649 - accuracy: 0.6720 - val_loss: 1.6572 -
Epoch 25/25
63/63 [==============================] - 0s 1ms/step - loss: 0.8114 - accuracy: 0.6859 - val_loss: 1.6492 -
Test Score (with PCA): 1.6569201946258545
Test Accuracy (with PCA): 0.375999987125396 73
/home/lisaa/mainpy/lib/python3.11/site-packages/keras/src/engine/training.py:3000: UserWarning: You are sav
el.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
  saving_api.save_model(
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 500)               50500

 dropout (Dropout)           (None, 500)               0

 dense_1 (Dense)             (None, 5)                 2505

=================================================================
Total params: 53005 (207.05 KB)
Trainable params: 53005 (207.05 KB)
Non-trainable params: 0 (0.00 Byte)
```

K-NEAREST NEIGHBOR MODEL

- KNN is a Machine Learning Supervised Non-parametric Model which calculates the distance classify the new individual data point.

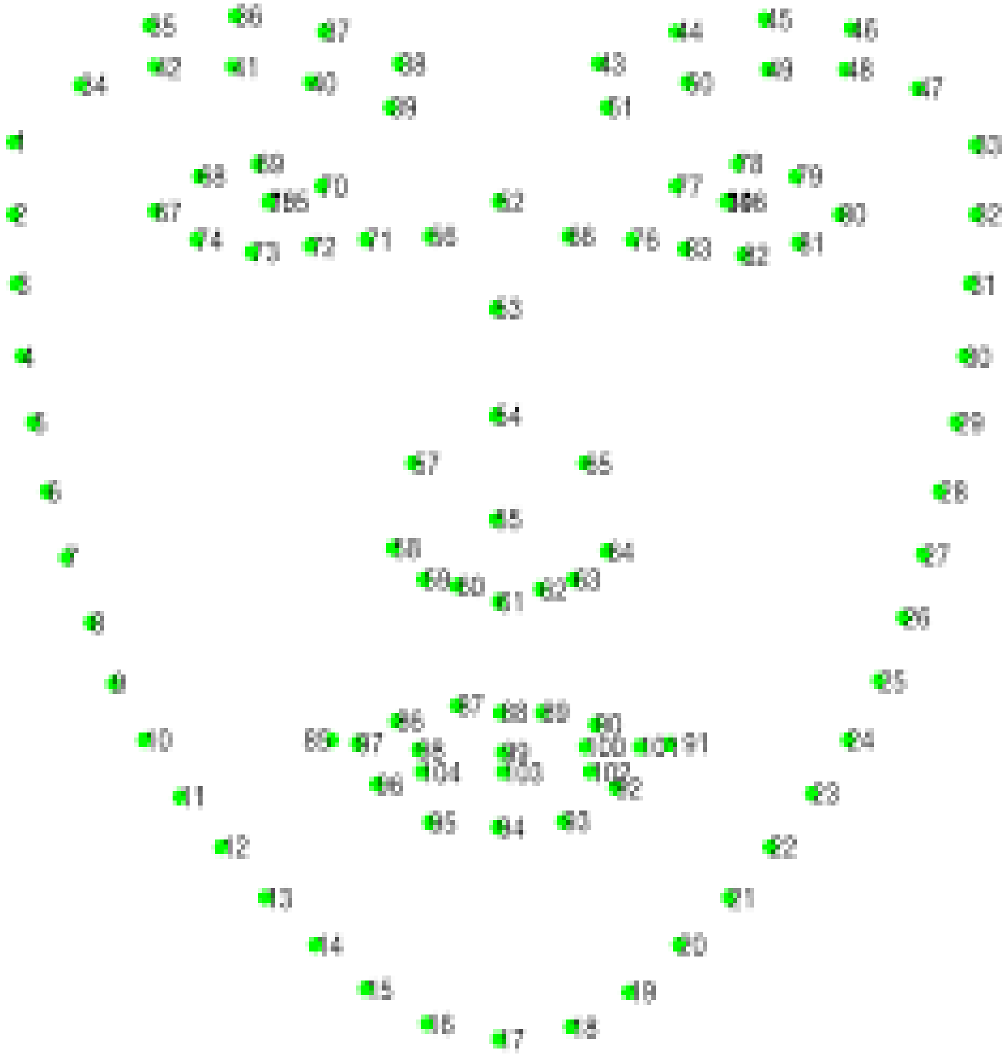


Working of KNN model

- KNN is a non-parametric algorithm, which means that it does not make any assumption on underlying data.
- KNN is a lazy learner algorithm, which does not learn or get trained from the training dataset, instead it stores the dataset and uses it at the time of each and every classification.
- It uses Distance parameter to classify the new individual data point. It calculates the distance between the new datapoint and the existing datapoints, and selects k nearest neighbors which have least distance. And uses majority voting out of the selected neighbors classify the new datapoints.
- It uses Minkowski metrics to calculate the distance like Manhattan distance, Euclidean distance, etc…

FACE SHAPE DETECTION

- Our primary goal is to detect the shape of the face to recommend the best fitting glasses.
- So we are considering the Landmark Features on the face to detect its shape.
- We are extracting 68 points from the Landmark features, in which each point consists of x and y coordinates that represents the landmark point on the face.



KNN Model For Face Shape Detection

- We have the images data set which contains 4000 images to train and test the model. There are 5 labelled classes named ‘heart’, ‘oval’, ‘oblong’, ‘round’, ‘square’.
- For each image we extract the landmark features and its corresponding label.
- We split the landmark features dataset and labels dataset to train and test the model.
- Knn model first stores all the training dataset and does not get trained on the training dataset.
- When we test the model with new images, it extracts the landmark features of that image and compares these features with each landmark features of existing training images.
- It calculates the distance between the new image datapoints and the existing image’s datapoints and finds the k-nearest neighbors with least distance, and uses majority voting to classify the new image as one of the existing labelled classes.

HyperParameter Tuning:

- Used different values for n_neighbors, weights, and power, to find the best fit parameters using Grid Search method.

Feature selection and engineering:

- Used Facial Landmark features to detect the shape of the face in the image.

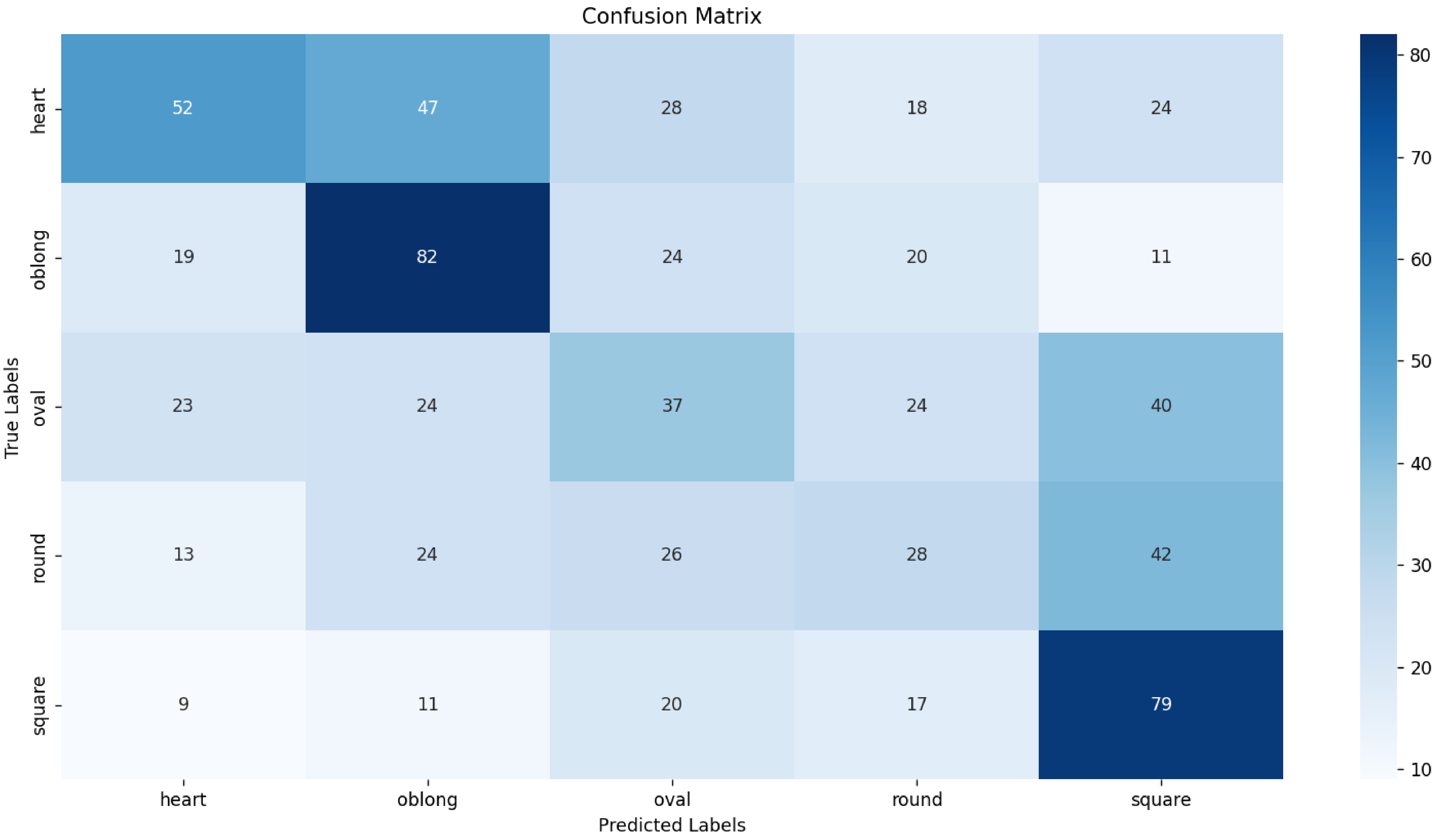
Performance metrics:

- Used accuracy_score, Recall_score, Precision_score, F1_score to check the the performance of the model.

Visualization:

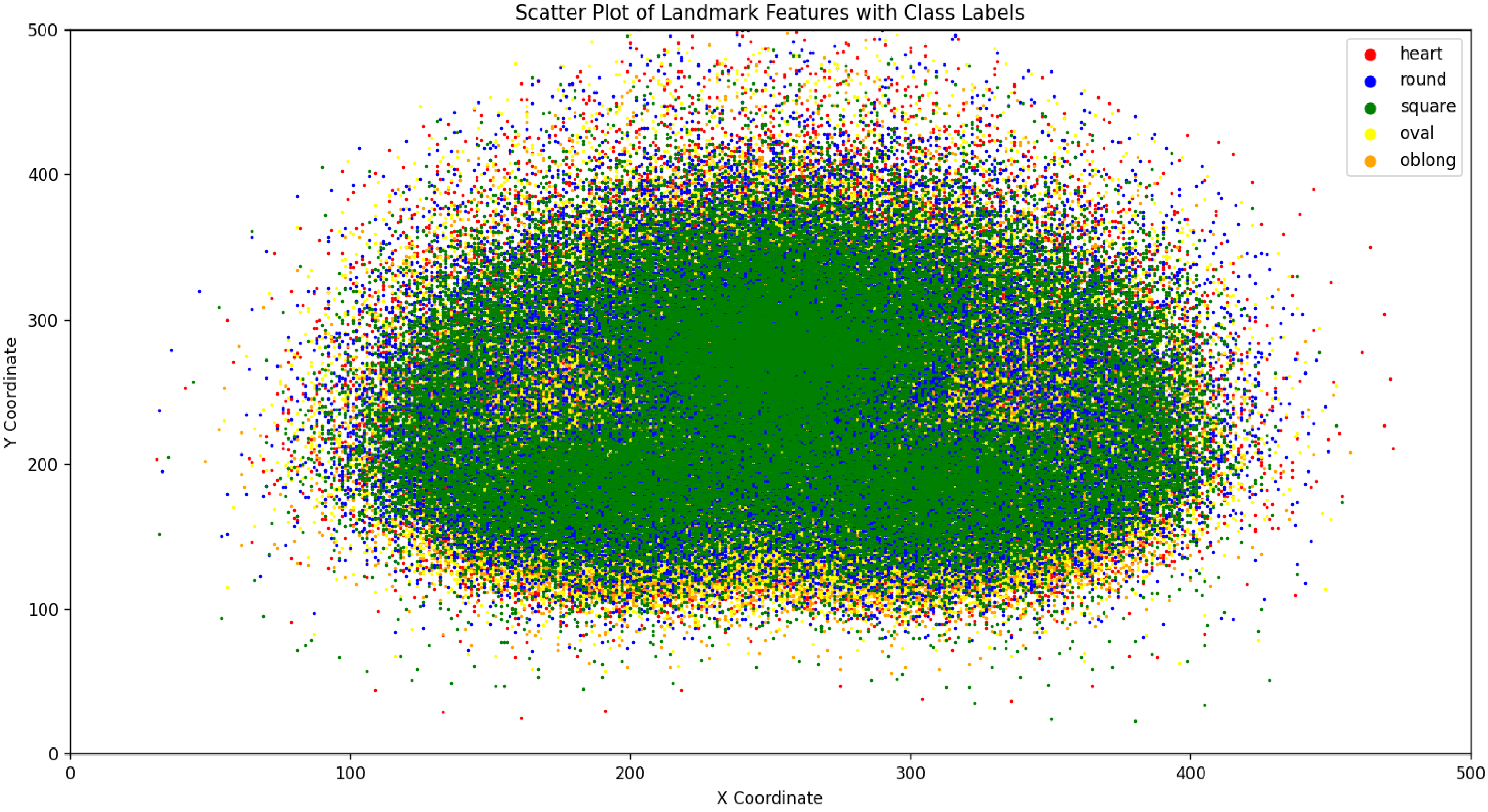
- Used Confusion Matrix, Classes distribution, ROC Curve to represent the dataset on which the model got trained.

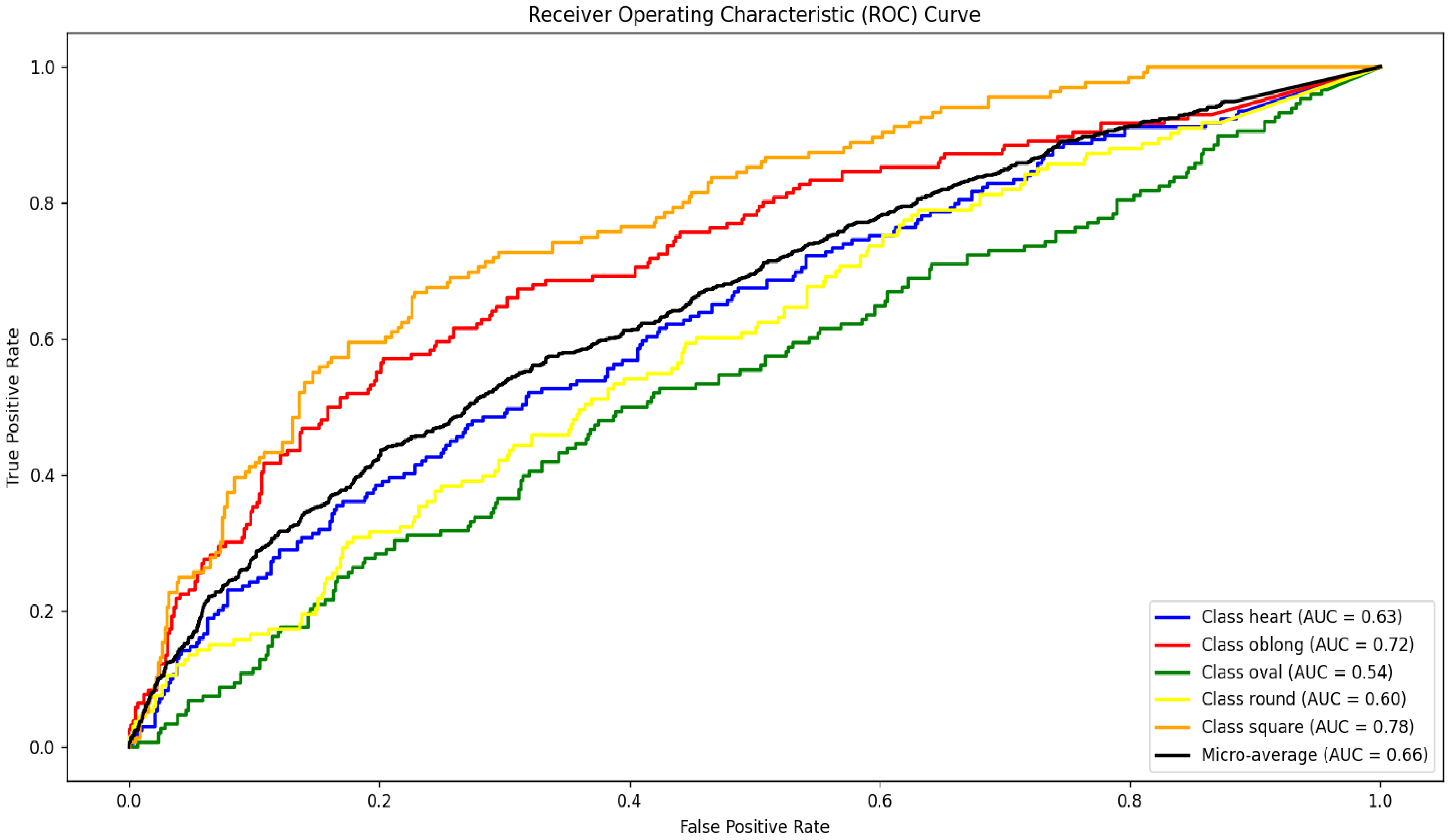
- Accuracy: 37.466%
- Precision: 36.925%
- Recall: 37.466%
- F1 Score: 36.455%
- Hyperparameters:
- K_neighbhors: 15
- Power: 2
- Weights: Distance
- Confidence Interval: 0.2-0.6



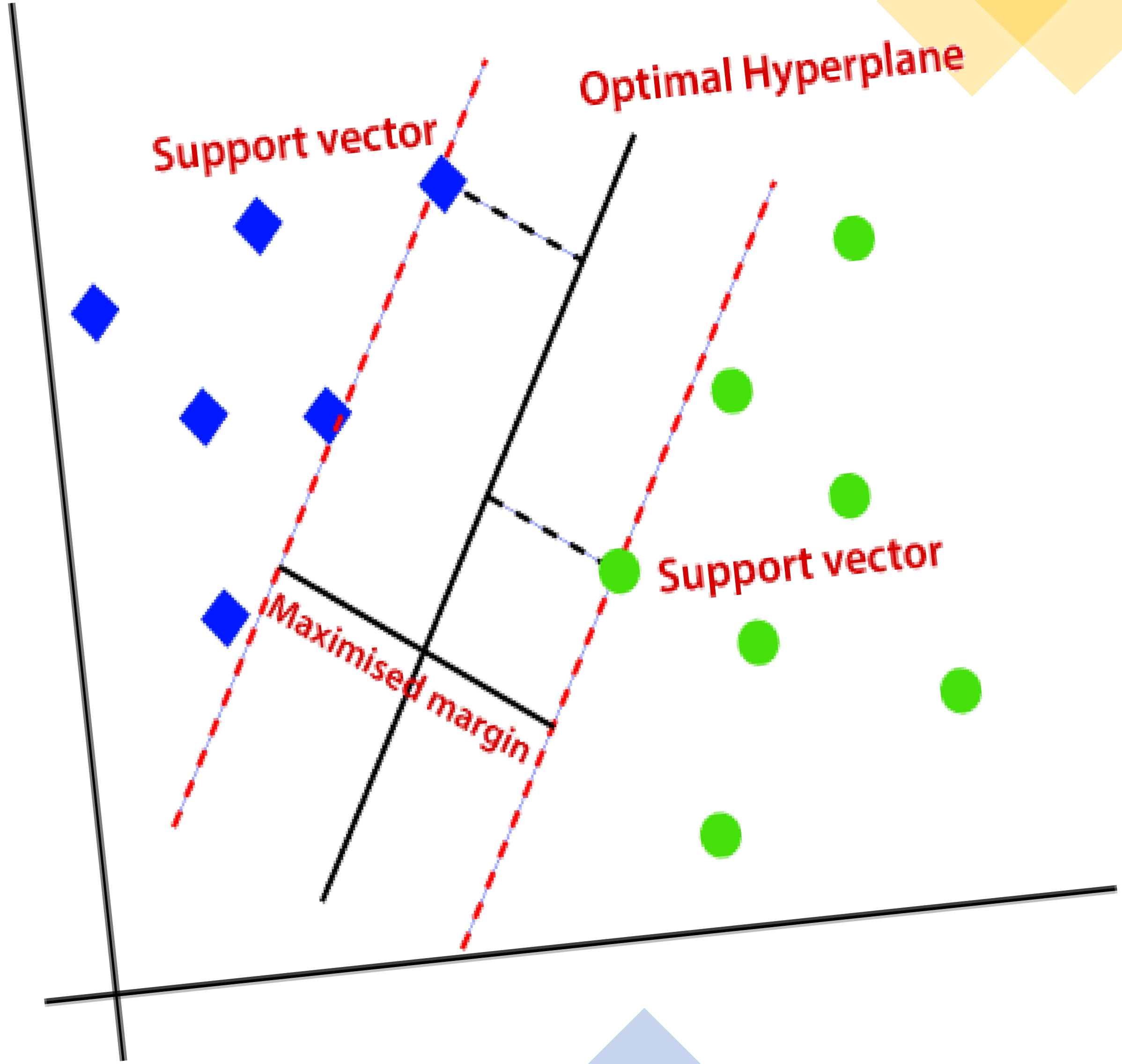
Confusion Matrix

True Labels \ Predicted Labels	heart	oblong	oval	round	square
heart	52	47	28	18	24
oblong	19	82	24	20	11
oval	23	24	37	24	40
round	13	24	26	28	42
square	9	11	20	17	79


Scatter Plot of Landmark Features with Class Labels
heart
round
square
oval
oblong
Y Coordinate
X Coordinate
0
100
200
300
400
500
0
100
200
300
400
500


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Class heart (AUC = 0.63)
Class oblong (AUC = 0.72)
Class oval (AUC = 0.54)
Class round (AUC = 0.60)
Class square (AUC = 0.78)
Micro-average (AUC = 0.66)

SVM (Support Vector Machine)



Data Preprocessing

- The dataset the is used for the purpose of training and testing the dataset contains 4000 images of celebrities with their faces, along with the labels (i.e., the face shapes which are Heart, Oblong, Oval, Round, Squared).
- The initial step for building an effective model is good data preprocessing, therefore we process these images such that our task of extracting features from them to classify the face shapes becomes easier.
- Firstly, we convert these RGB images into Grayscale for easier computation and equalize the images for luminance distribution of images. After the completion of this step the images are stored in a folder where the directory structure is same as the directory structure from where the input images are taken from.
- Other preprocessing techniques such as squaring, blurring, smoothening , edge enhancing, sharpening, cropping, flipping, expanding etc., can also be done.
- After the completion of preprocessing step, we then move onto feature selection and feature engineering.

Feature Selection

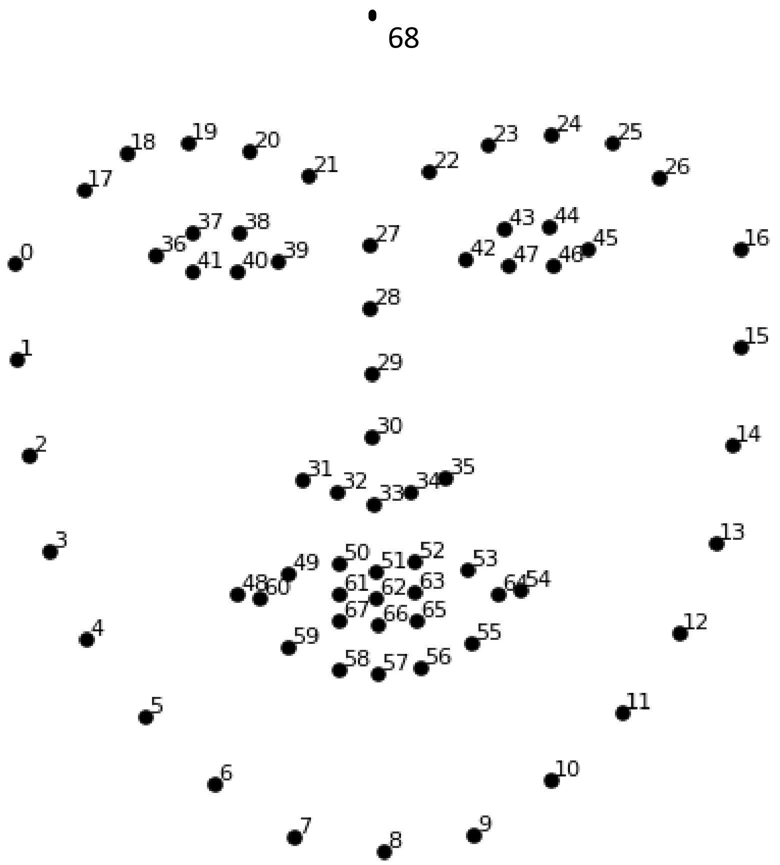
- Based on the information gathered from different sources, the face shapes are classified accordingly,
  - Heart-Shaped: The width of the forehead is greater than the width of the jawline and the chin may be pointy.
  - Oblong-Shaped: The ratio of the height of the face is very long compared to the width.
  - Oval-Shaped Face: The ratio of the height to width is approximately 1.5.
  - Round-Shaped Face: the widths of the forehead, cheekbones, and jaw are equal and the jaw is slightly round as opposed to angular.
  - Square-shaped face: The widths of the forehead, cheekbones, and jaw are equal and the jawline is sharp.

- Based on the information gathered from the various sources of face shapes are classified, we finalized the following features which are very helpful for detecting the face shapes.
  - The facial height to facial width ratio.
  - Distance between chin and mouth to distance ratio.
  - Ratio of jawline width to facial width.
  - Set of widths measuring from the cheekbones to the chin.
  - Angles made by different outline points on both sides of the face with respect to the vertical line (i.e., facial height line).

Feature Engineering

- After selecting the necessary features, we then build these features. These features are to be built by identifying the face from the processed image and detecting the landmarks.
- The identification of face is done using the cv2 along with "haar-cascade" .xml file and the inbuilt "dlib" frontal face detector. The detection of landmarks on the face are done using 68 landmark detection .dat file which is used with the "dlib" functions.
- In case if no or multiple faces are detected in the image, then the parameters of the function used for face detection are tuned in such a way that only one face is detected at the end. And even after a considerable amount of tuning the face is not detected then we are manually detecting the face by giving the boundaries which are closer to the size of the image.
- After all the 68 landmarks are detected, we are then using certain points to build the features necessary to train the model are to be engineered.
- Now these points are named from 0 to 67 based on the image shown in the next slide.
- For building the features,
    - For facial height, we are using point 27 and reducing its y-coordinate until we reach the hair which has a considerable amount of change in luminosity compared to the skin and marking the point at the boundary between the skin and hairline which is vertically above the p27. Then we calculate distance between this point (which will be point 68) and the point 8.
    - For facial width, we are calculating the distance between point 0 and point 16.
    - For jawline width, we are calculating the distance between point 4 and point 12.
    - For angles made by the outline point to the vertical, the angles between the point 8 and all the other points from 0 to 16 are calculated.
    - For different facial widths, the distances between the kth point and the 16-kth point are calculated where 0 <= k < 8
- We need to get these features along with the labels into a list for both training and testing data.
- For easier purpose, I am appending label as the first element for every images' features list

THE 68 LANDMARK POINTS



Model

- The SVM model generally finds the SVC by identifying the relationships between the data points as if they were at higher dimensions.
- The type of the SVM model used here is an SVC [Support Vector Classifier] model which uses the RBF[Radial Basis Function] kernel. This kernel calculates the relationship between the data points and the support vector classifiers for infinite dimensions.
- The radial basis function is as follows:

$$\mathrm{Exp}(-\gamma(a - b)^2)$$

where a and b represent any two data points, γ represents the scale the squared distances between two points, thus it scales the influence of any two points have on each other

- In the model used there are two parameters C and γ which are tuned accordingly using the hyperparameter tuning and the best values of both are chosen as parameters.
- Here C is the parameter for soft margin cost function
- The main basis of this function is to linearly separate the data points at higher dimensions which are not linearly seperable at lower dimensions.

Hyperparameter Tuning

- As discussed earlier, the two hyperparameters C and γ needs to be fine-tuned in order to get higher accuracy scores.
- The tuning of these hyperparameters is completely based upon the bias-variance trade-off.
- If the value of C is large, the bias will be low because of penalizing and will be high and similarly a smaller C value gives a higher bias and lower variance.
- If the value of γ is small, then the influence the two points have on each other will be high and if the value of γ increases the influence the two points have on each other will decrease.
- In the model, these parameters are being tuned using grid_serachCV where 15 values of C and γ are taken between the ranges -5 and 5.
- Based on these tuning the model will be trained on the best parameters which produces higher accuracy and then with that model, the testing takes place.

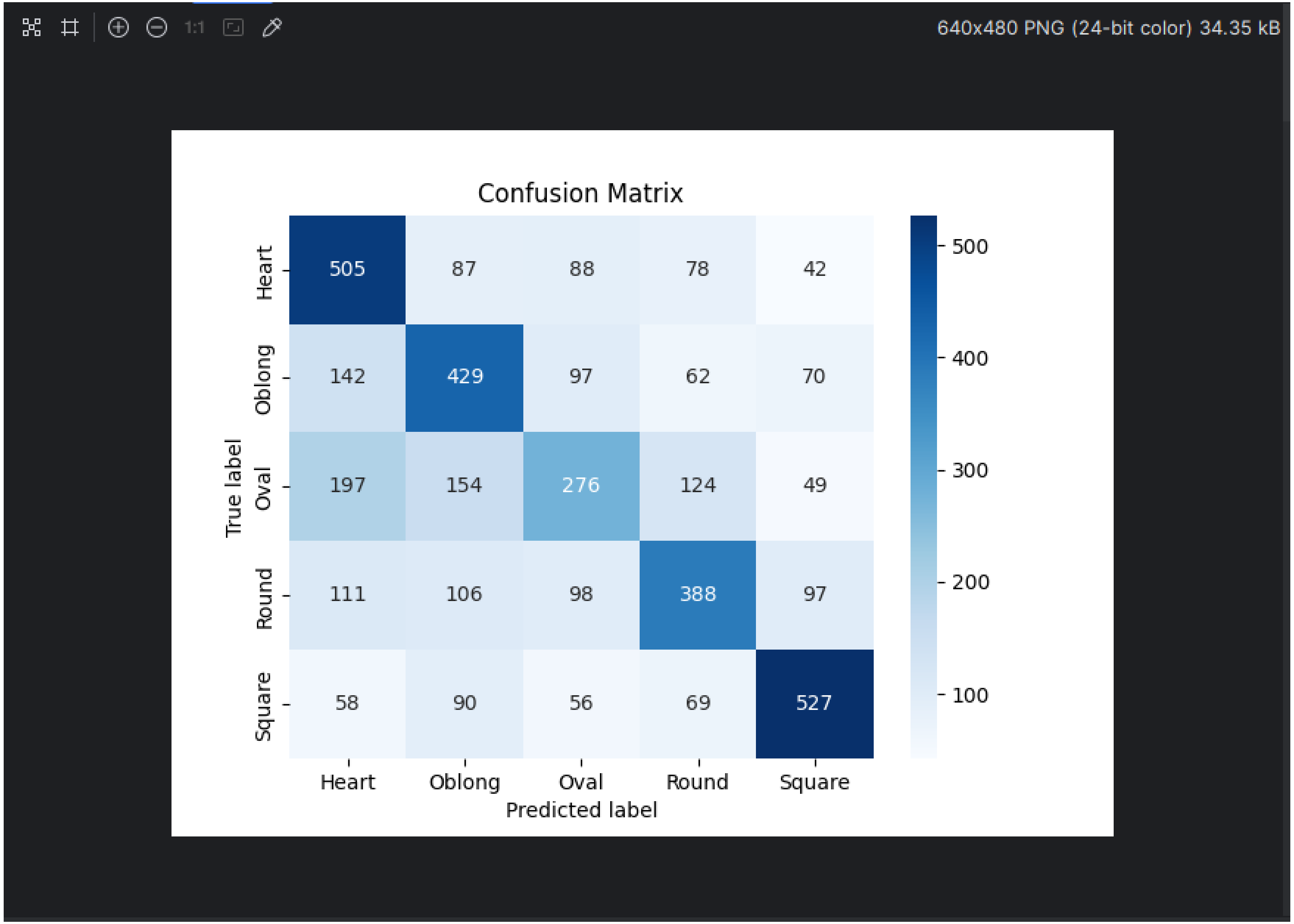
- Performance Metrics:

```
Accuracy:  0.53125
Recall:  0.53125
Precision:  0.5304319699423767
```

- Classification_report:

```
            precision    recall  f1-score   support

      Heart       0.50      0.63      0.56       800
     Oblong       0.50      0.54      0.52       800
       Oval       0.45      0.34      0.39       800
      Round       0.54      0.48      0.51       800
     Square       0.67      0.66      0.66       800

   accuracy                           0.53      4000
  macro avg       0.53      0.53      0.53      4000
weighted avg       0.53      0.53      0.53      4000
```

Visualizations

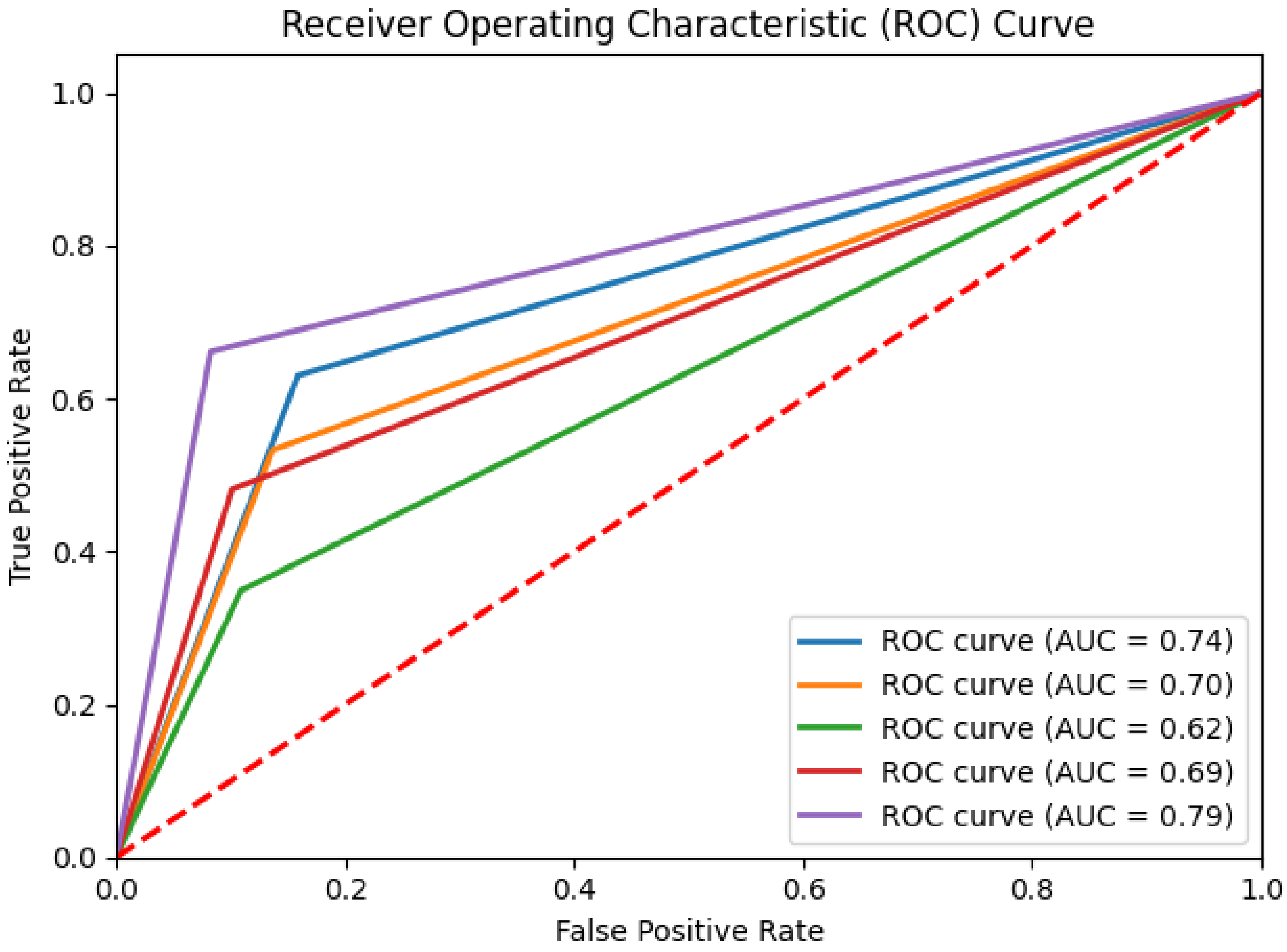


640x480 PNG (24-bit color) 34.35 kB

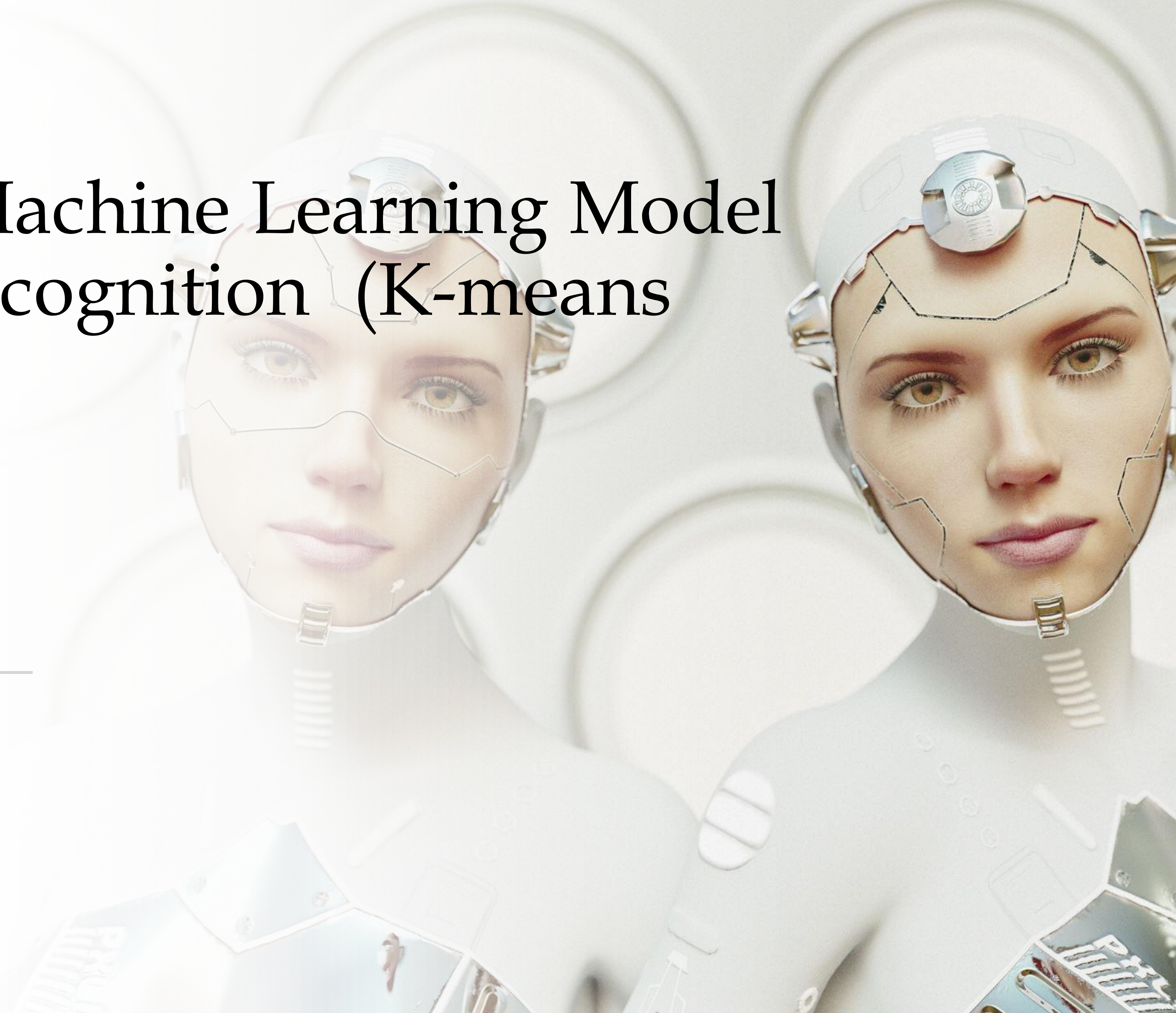
Confusion Matrix

True label \ Predicted label	Heart	Oblong	Oval	Round	Square
Heart	505	87	88	78	42
Oblong	142	429	97	62	70
Oval	197	154	276	124	49
Round	111	106	98	388	97
Square	58	90	56	69	527

Visualizations



Unsupervised Machine Learning Model for face shape recognition (K-means clustering)



The 81 facial landmarks:

The 81 facial landmarks helps us in identifying the critical landmark points on our face that will be helpful in predicting the face shape.

-> Here we are specifically choosing the facial points 2,14, 69, 75, 79, 72, 8, 12, 4, 6, 10, 7, 9.

-> which will help us in calculating or more like extracting distances, ratios and angles from face.

The distances are:

Landmarks numbered 2 and 14 were used for getting the distance D1 between the left and right ear.

Landmarks numbered 75 and 79 were used for getting the distance D2 between the left and right forehead.

Landmarks numbered 69 and 72 for the left and right-forehead and 8 for the chin. These landmarks were used for getting the distance D3 from the hairline to the chin.

- Landmarks numbered 8 and 12 were used for getting the distance D4 of the jawline.
- Landmarks numbered 4 and 12 were used for getting the distance D5.
- Landmarks numbered 6 and 10 were used for getting the distance D6.
- Landmarks numbered 7 and 9 were used for getting the distance D7.

The ratios:

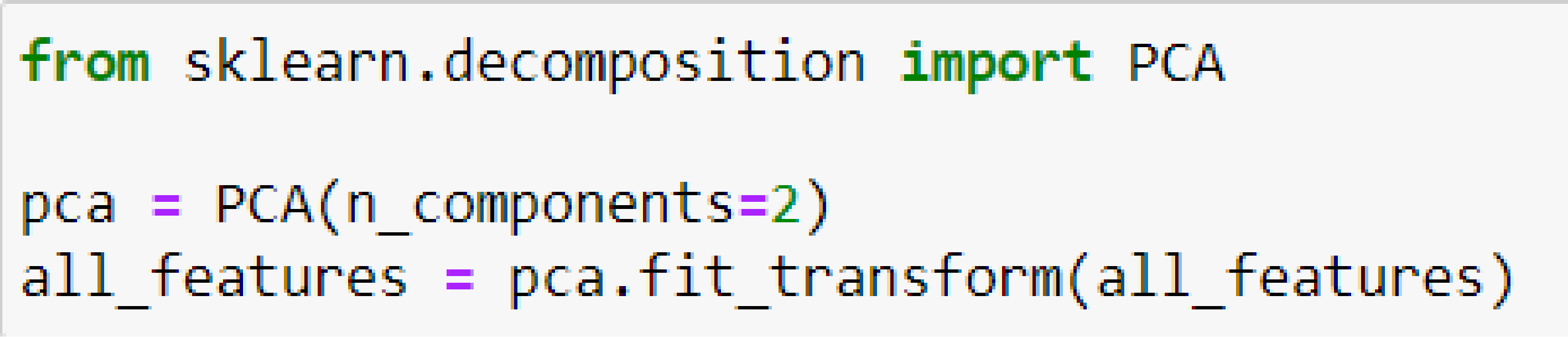
D2 / D1, D1 / D3, D2 / D3, D1 / D5, D6 / D5, D4 / D6, D6 / D1, D5 / D2

The angles:

- Angle 1 between the line from hairline to chin and line from chin to landmark numbered 10.
- Angle 2 between the line from hairline to chin and line from chin to landmark numbered 12.
- Angle 3 between the line from landmark numbered 2 to landmark numbered 14 and line from landmark numbered 14 to landmark numbered 12.

Furtherly.....

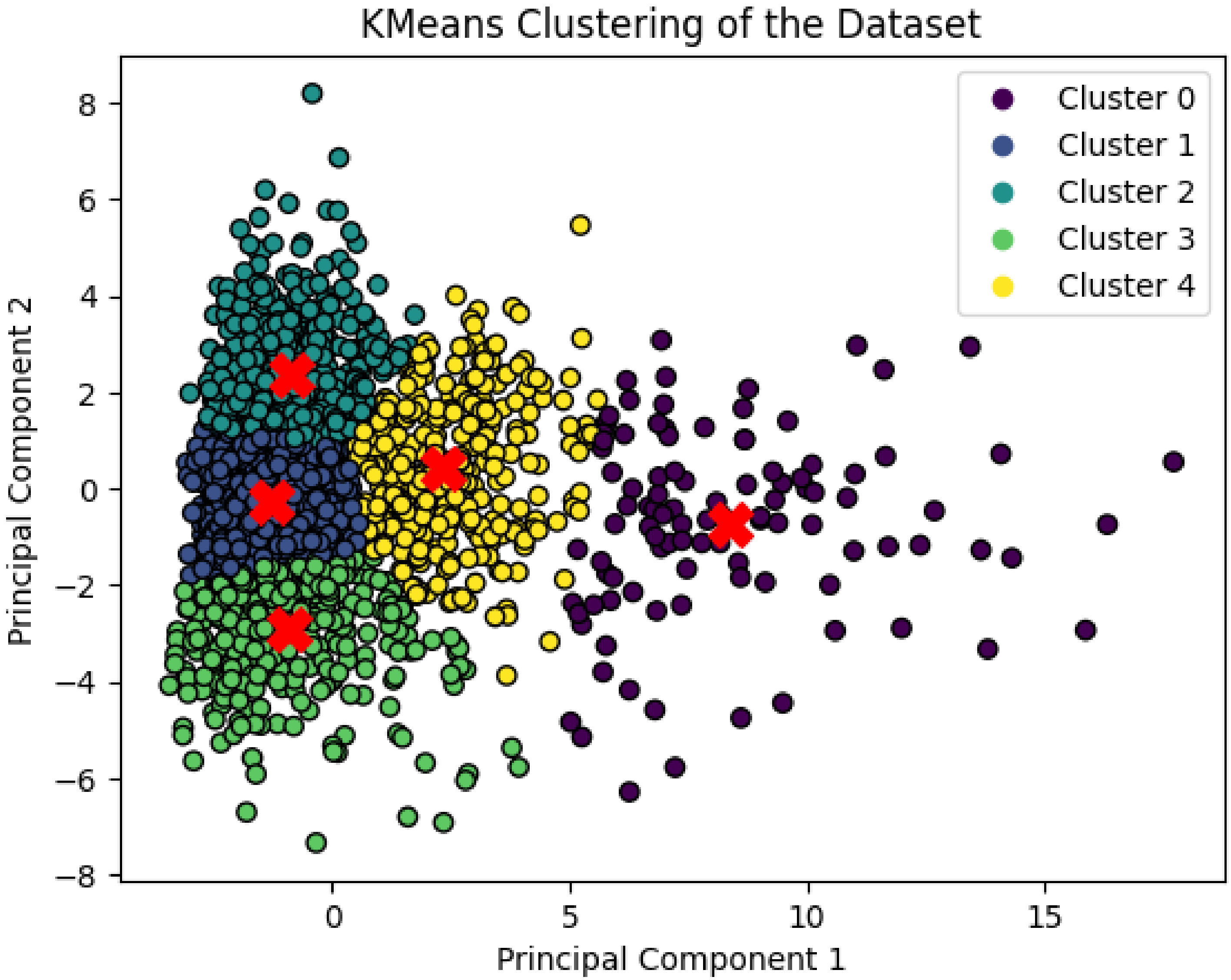
- We then, standardize these values using StandardScaler from sklearn.preprocessing
- And then apply PCA to reduce the dimensions



```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
all_features = pca.fit_transform(all_features)
```

Then the inputs are passed to KMeans model for training.

- The cluster plot of the model:



Dimensionality Reduction

- The preprocess_SVM_with_PCA function begins by extracting facial landmarks and computing a set of features from the input image data (represented as a high-dimensional feature vector).

- This leads to increased computational complexity and may result in overfitting PCA is applied to the feature vector to transform it into a lower-dimensional space while retaining the essential information that contributes to the variance in the data.

- This transformed input not only reduces the computational burden but also helps in mitigating the curse of dimensionality

ENSEMBLE LEARNING

- used ENSEMBLE LEARNING technique to combine the results of all the models and applied majority voting to the acquired results to get the best result that is having the highest confidence score.
- Machine Learning Models used: CNN,SNN,KNN,SVM,KMEANS models for the prediction of face shape
- All the above models have been trained with the data after applying PCA reduction to it.
- In Ensemble learning model, all these models are loaded. There exists a folder with images which are not there in the original dataset on which the models got trained, to validate the performance of the models.
- From the validation dataset, many features of each image have been extracted according to the requirement of each of the models.

- As per the requirement of each model, the extracted features have been sent into the model for the prediction of images in the validation dataset.
- Then combined the predictions of each image and stored it along with its confidence scores.
- At last, majority voting has been performed to get the best result having the highest votes, that is the predictions predicted by most of the models.
- In case there is a tie between 2 or more labels, we used average confidence score of each label. Then the label with the highest confidence score has been selected as the best result.
- This result is then sent into Augmented Reality function to recommend the FRAME that is best suitable to that person's face.

Gaussian Mixture Model

- A Gaussian Mixture Model (GMM) is a probabilistic model that represents a mixture of multiple Gaussian (normal) distributions. It is commonly used for modeling complex probability distributions in data analysis and machine learning applications.

- Gaussian Mixture Models (GMMs) are commonly used in unsupervised learning scenarios. In unsupervised learning, the algorithm is given a dataset without explicit labels or categories, and its goal is to discover patterns, structures, or clusters within the data.

- GMMs are particularly well-suited for unsupervised learning tasks such as clustering. The model can automatically identify underlying clusters in the data without prior knowledge of the class labels.

- Features are based upon the 68 landmarks points detected using dlib library. Those points are considered to engineer the features required for the model.

- The following features are engineered from the detected landmark points:
    - facial height
    - facial width
    - Jawline width
    - angles made by the outline point to the vertical
    - the distances between the kth point and the 16-kth point are calculated where 0 <= k < 8

- Since this is an unsupervised model and our purpose is classification, we need to reduce the dimension of this data to two, such that it can be plotted on x-y plane.

- This plotting of the points are essential, because we can't identify which classes belong to which clusters.

- Therefore, we manually select a point whose class is already known and plot it in the plane. By plotting it, we can identify to which cluster it belongs to and thus differentiate those clusters based on the classes.

THANKYOU