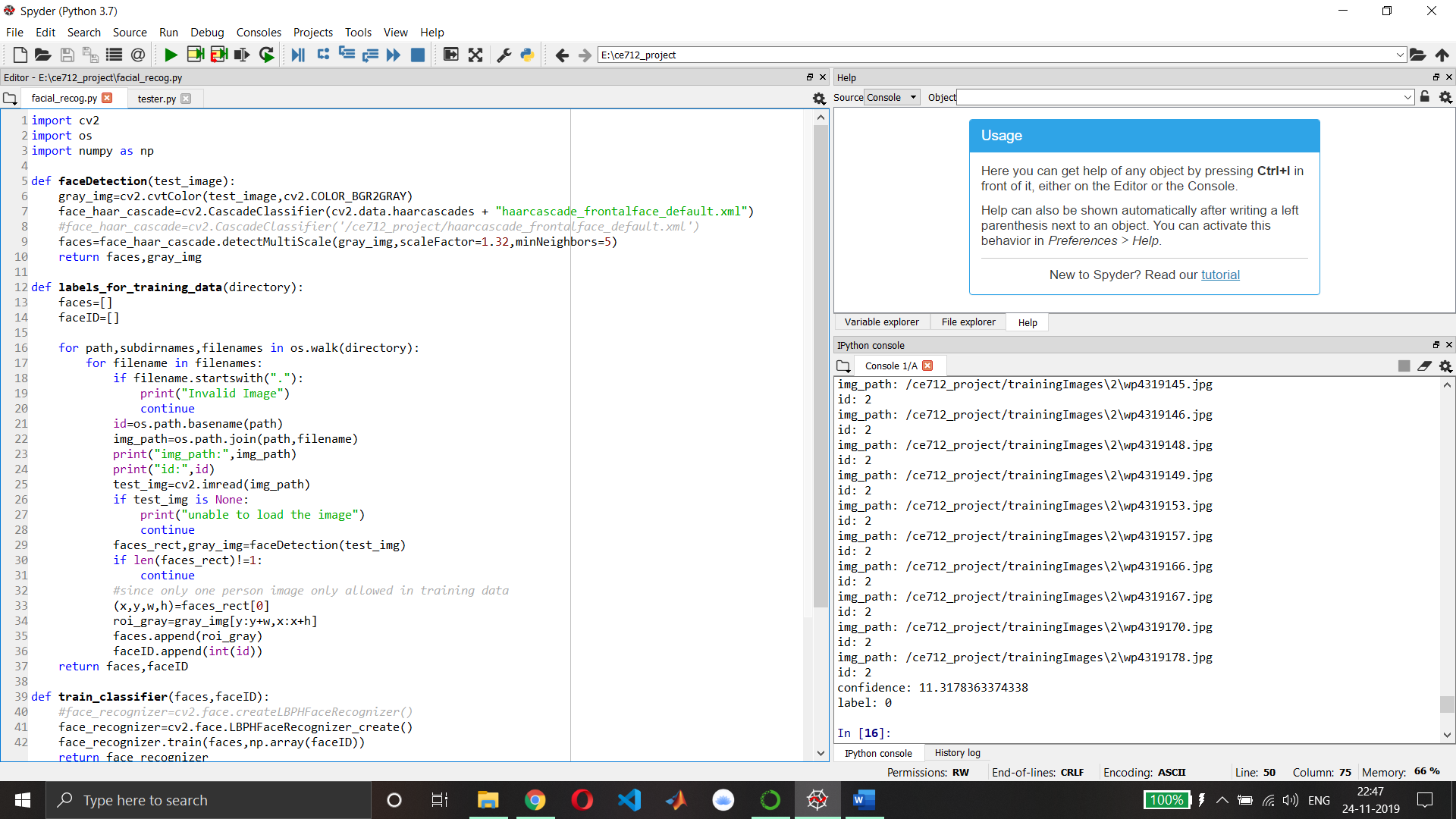
Facial Recognition

Kanishk Jadhav (170040004) ||**CE-712 COURSE PROJECT**||Jay Thakkar (170040006)

Introduction: Detecting the face in the test image and later recognizing which class image belongs with. The probability of a test image of being related to a particular class in training data is calculated by Machine Learning Techniques or Statistical Methods. In our case we have used Linear Binary Pattern Histogram (LBPH). The code is presented in Python using OpenCV library. We also use different classifiers to create the boundaries within the given data points.

**Insync with Code and Logic Flow:**



**For, face\_recog.py:**

Initially in the ***faceDetection*** function we are passing the image for which we want to find the results. The faceDetection function detects the faces (can be multiple) in the particular test image and returns the array of coordinates of detected faces. In faceDetection function we are also converting the test image into greyscale so that the feature extraction becomes easy in the greyscale image. In this function we are using Haar Cascade classifier which is basically an opencv tool trained to detect the faces with the help of a pre-developed XML file. This Haar Cascade derives its result from superimposing the positive and negative images. Scale factor= 1.32 means decrease the size of image by 32% because our classifier was trained to detect for certain fixed size only, so to increase the accuracy we rescale the image. Mineighbors tells whether the face detected is true positive or negative on the basis of the neighbouring points.

Next, we construct the ***labels\_for\_training\_data*** function. In this function we pass the directory of the training data. Accordingly using the folder name of classes (for eg 0,1,2) we label the training data and also each class folder must contain a same persons training data. We use the OS library of python for accessing the directories, sub-directories and paths. So, here we label all the images of all the classes using our function. Images in training data which are null or having 2 or more than 2 faces are ignored by us. Also, we call faceDetection function on each and every training image and return the coordinates of detected face and its converted greyscale image. Later we crop the grey image according to the coordinates of face detected and call it is as our region of interest. This new grey region of interest is appended than into “faces” array for each training image and label of each image is appended into “faceID” array for each training image. We return the faces and faceID arrays from this function

In the ***train classifier*** function, we pass the faces and faceID arrays of training data images. Later we use the cv library package functions to train our model. Here we use Local Binary Pattern Histogram (LBPH) to do the statistical analysis of each class (i.e. person). It uses the faces and faceID arrays to train the data using train function of OpenCV library. Here we return facerecognizer variable which later can be use predict the class of the input/test image. In LBPH, a pixel is selected in an image to analyse and surrounding 8 pixels around it are also selected simultaneously. After selecting the block of 9 pixels, the central pixel greyscale value is than compared with the neighbouring pixel and their new values are set accordingly. If the initial neighbour pixel value is greater than central pixel value then the new neighbour pixel value becomes 1 and if it is less than the central pixel value it becomes 0. This neighbouring number are later arranged in single series with staring from top left corner in clockwise. From this we obtain a binary number from above process and later convert this binary number to a decimal number. This decimal number will be the new value of the central pixel. Repeat the process for all the pixels in the image and calculate the histogram of the image. Average out the histogram of all training data sets into a single histogram. Later when test image is passed find the distance between the histogram of test image and training data sets histogram and accordingly find a class.

***draw\_rect*** function draws the border around the faces detected in given test or input image

put\_text function puts text around the coordinates of the faces detected in given test or input image with custom input text.

**FOR test.py:**

We import the required packages from the respective libraries. We load an input image which is to be classified. Then we use faceDetection function on the input image to get the coordinates of detected face in input image and the converted grey image.

Next, we label the training data images using the labels\_for\_training\_data function. Then we train our model or classifier using train classifier function

Initially we crop the faces detected in grey images. For each face of test/input image, we compare it with all of our training image faces using the predict function of OpenCV package. This return us the with the label and confidence value for each face in test/input image. Lower the confidence value, higher is it rate of accuracy. Confidence value and actual confidence of predicting are inversely proportional. Then using the predicted label, we find the predicted name from dictionary defined above for names: labels. For confidence value < 40, we discard the criteria for classification into that particular class. We use the function draw\_rect and put\_text later on the test/input image.

Resizing the image ensures that it is completely visible on our screens. Later imshow we display the image with the labelled name and rectangle box drawn around the face.

**Results/Conclusions:** Using Linear Binary Pattern Histogram method, the accuracy of classifying or rather recognizing the image increases even there is low lighting and other distortions. It is robust against monotonic grey scale transformations. Also, our model is not reliable when the number of training images are very less. It may start predicting wrongly with high confidence value. Hence, we have detected and recognized the faces given for an input/test image.

Explanation for Code: