VISUAL RECOGNITION

ASSIGNMENT-3

FACE DETECTION AND RECOGNITION

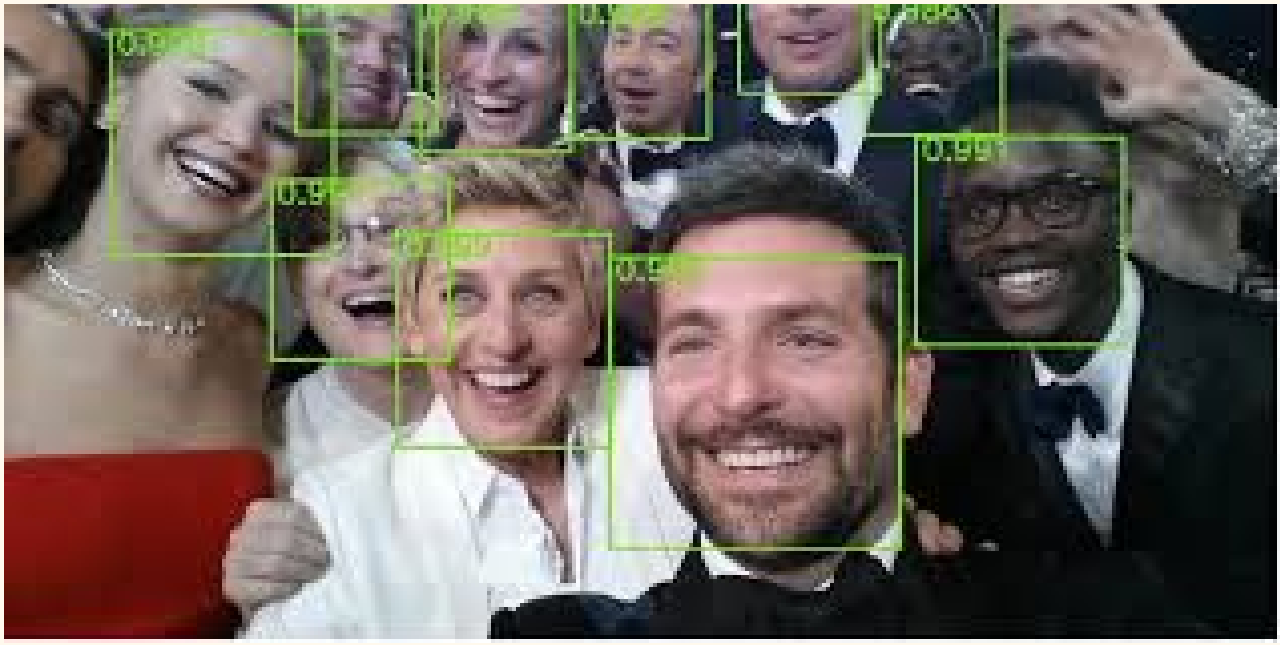
—

Prepared By:

Durga Yasasvi Yalamarthy - IMT2016060

Srujan Swaroop Gajula - IMT2016033

Siddarth Reddy Desu - IMT2016037



INTRODUCTION

In this project, the primary tasks involved are as follows:

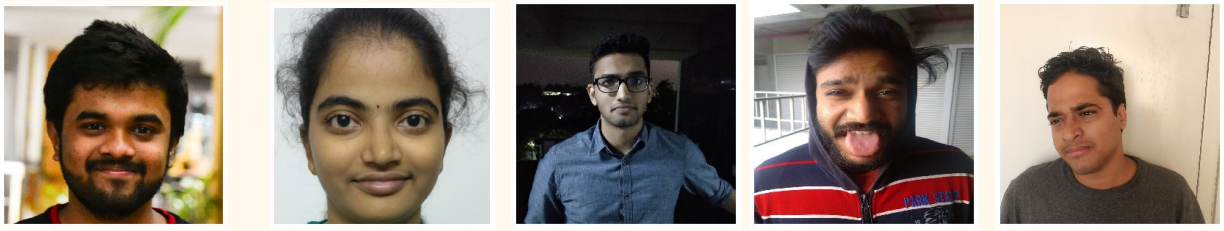
1. Detection of the face.
2. Performing pre-processing on the detected face.
3. Recognizing the detected face.

PART-1 (Dataset)

DATA COLLECTION

Data contains images of a total of 43 persons in different illuminations, pose variations, expressions, etc constituting a total of 858 images.

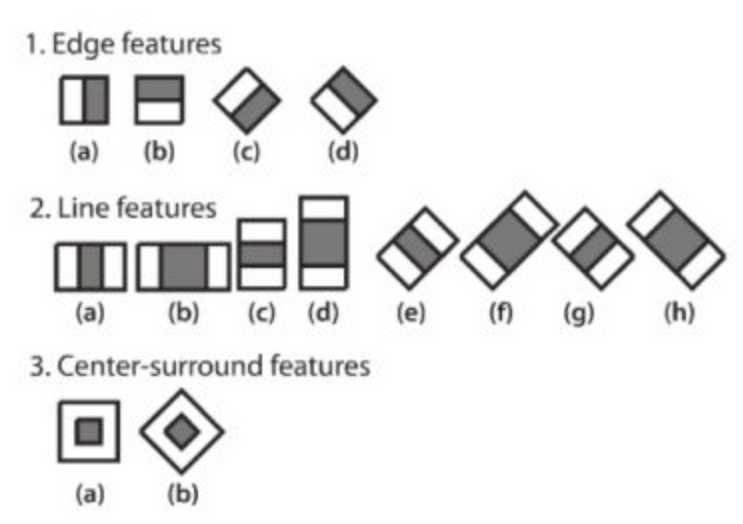
DATA SAMPLE USED TO DESCRIBE THE PROCESS:



DETECTION OF FACE

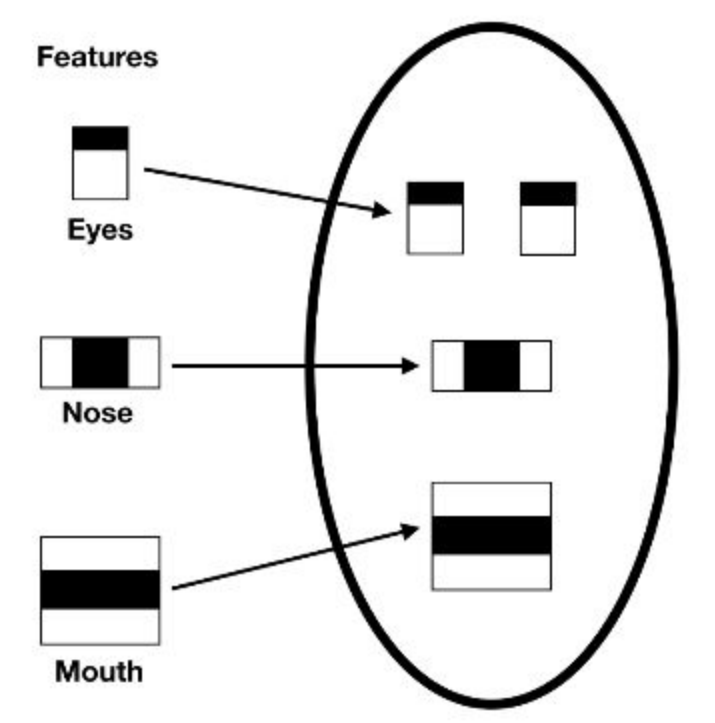
Detection of face essentially means to distinguish face region from non-face region, and specify the face with a geometrical structure usually square. Here the process of face detection follows the method proposed by Paul Viola and Michael Jones and uses the **Haar-feature based Cascade Classifiers**.

The approach is machine-learning based where it exploits the characteristic features of the face by applying various convolutional kernels on the image which produces a variety of features called **Haar features.**



These Haar features are tested on face images and non-face images using Ada boost and the features with minimum error rate are selected.

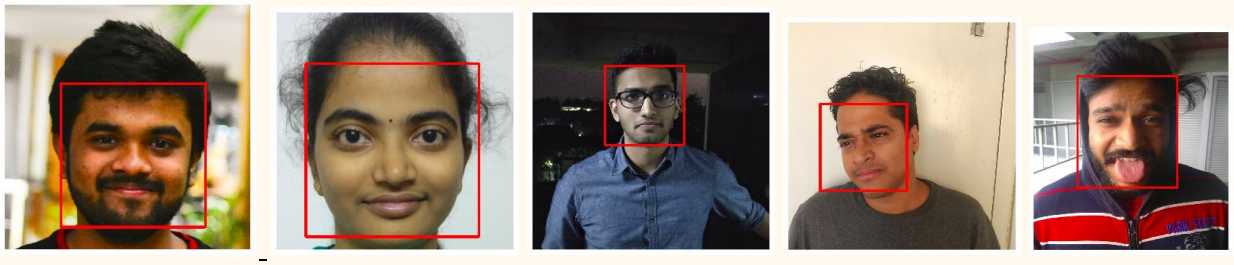
Intuitionally the features that might get selected after filtering of are of the type:



OpenCV constitutes of an inbuilt trainer as well as a detector and supports pre-trained classifiers stored in XML files. Hence we directly utilized its functionality.

Conventionally for a 24x24 image, the number of appropriately selected features are nearly 6000. In reality, the percentage of non-face region is more that of face region hence applying all 6000 features on every sub-image of size 24x24 is very inefficient,hence the features are grouped into different cascades and each cascade is applied sequentially so that non-face regions are discarded in first run itself, hence it is called as **Haar Cascade Classifiers.**

Images after Face Detection:



PART-2 (Pre-process)

PREPROCESSING OF DETECTED IMAGES:

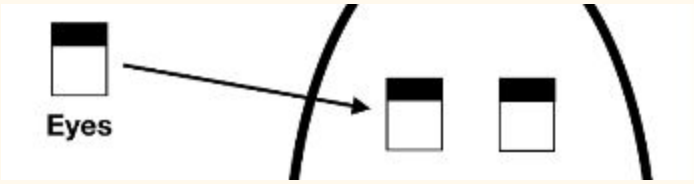
In order to achieve robust performance and efficient results for face recognition the pre-processing of the face data is mandatory so that only relevant and meaningful features of the face are extracted and these features are treated as input to Recognition task subsequently.

There are several pre-processing techniques that are supposedly meant to be performed to achieve expected efficiency, they are as follows:

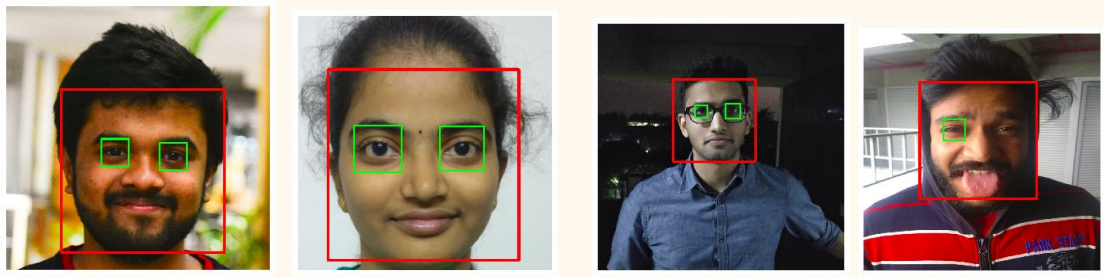
STEP-1: LOCATING THE CENTER OF EYE

This process, in turn, involves two subsequent steps:

- Detection of the eye using Haar Cascade feature as mentioned in the above description:

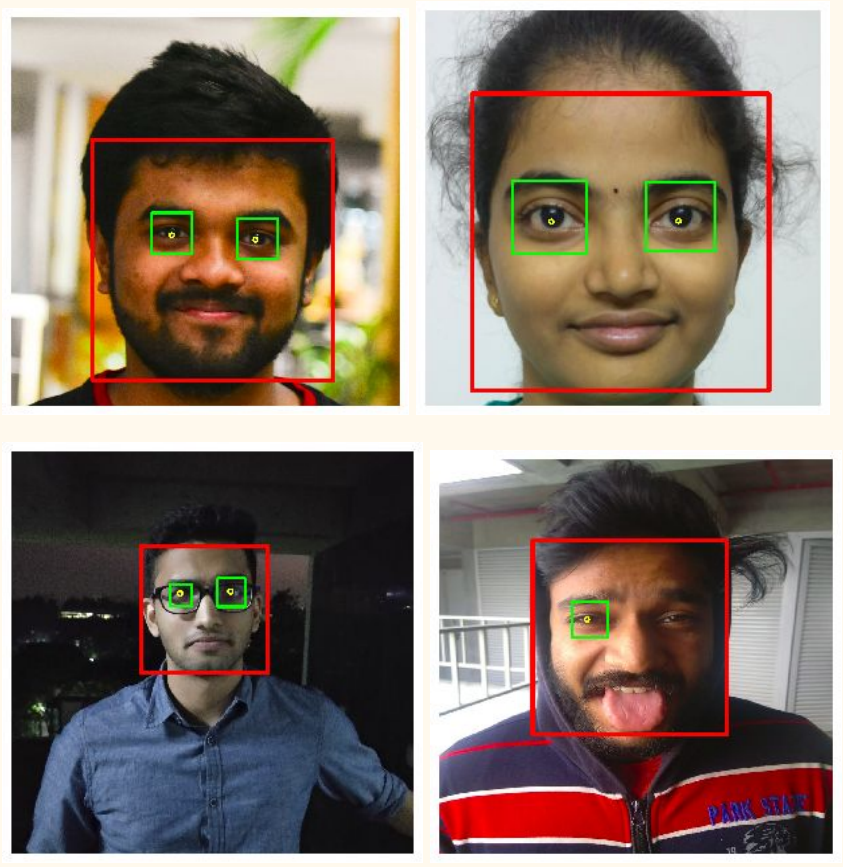


The result of the detection of eyes using Haar Cascade on sample data:



- The second step after detecting the eye is to find the centroid of the eyeball which involves two further steps:
  - ➔ Finding the eyeball by binarizing the eye images by setting a definite threshold.
  - ➔ This generates a binary image consisting of the only eyeball with a black background.
  - ➔ Now find the centroid of this black circle (using moments in OpenCV), these centroids are nothing but centres of each eye.

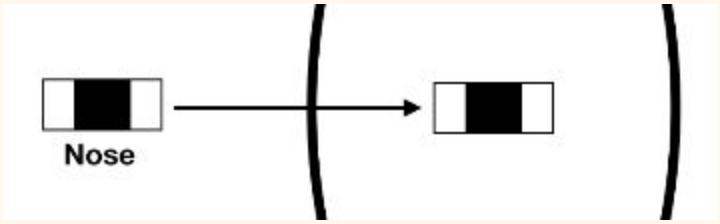
The Result of Detection of eyes using Haar Cascade and above-mentioned steps on sample data:



STEP-2: LOCATING THE TIP OF NOSE

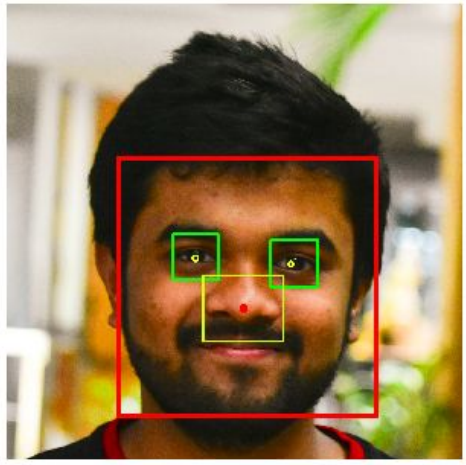
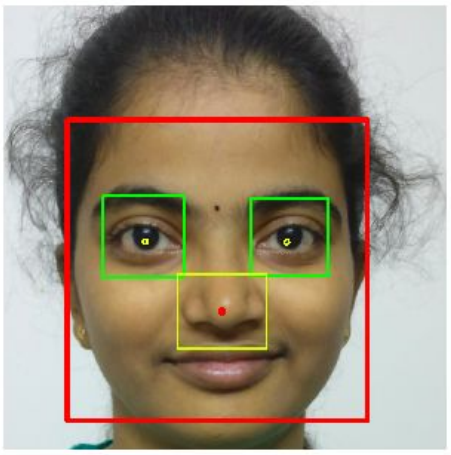
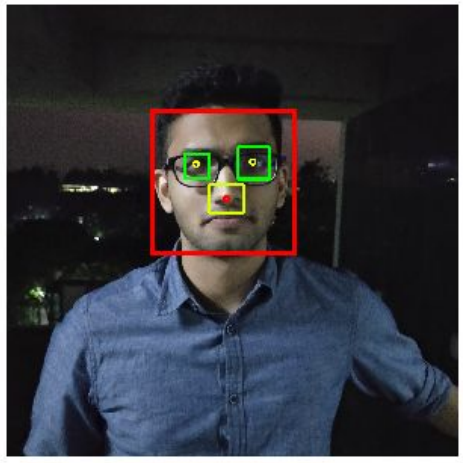
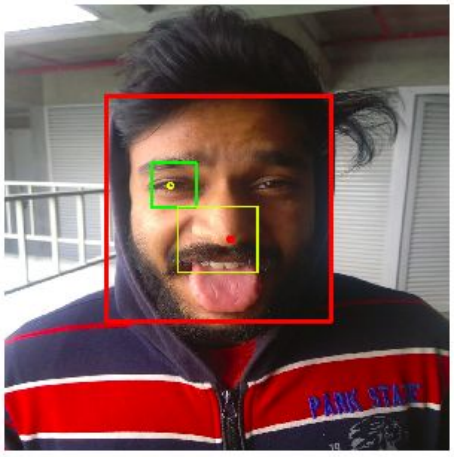
Again this involves a similar method of approach as:

➔ Detection of the nose using Haar feature Cascade as mentioned above:



➔ Applying the identical approach as specified to find the centroid of the eye, similarly, the tip of the nose is found.

Display of Data samples featuring tip of the Nose:

STEP-3: EXTRACTING THE OVAL REGION OF FACE

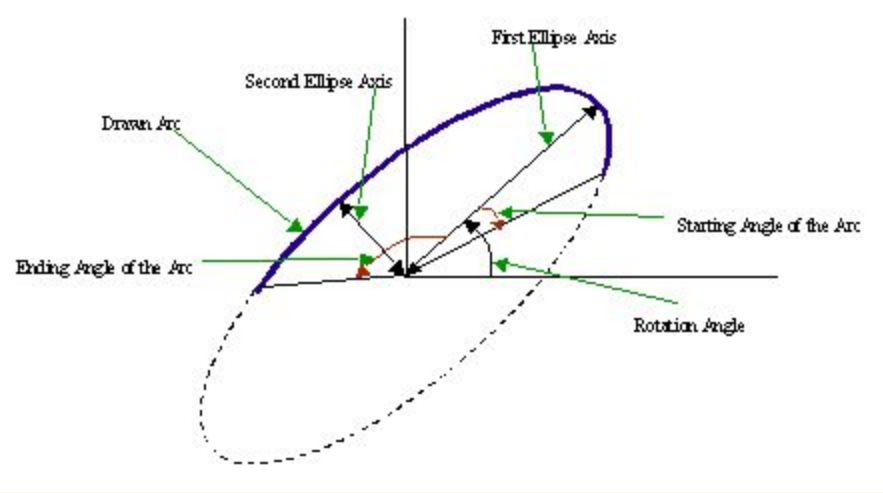
1. Creating Oval region corresponding to the face and resetting all pixels outside the region.
2. The centre of the eclipse initialized as the centroid of Nose.
3. The minor and major axis is calculated using the relative distance between the centroid and the respective centres of the eye.

4. Also, the angle of inclination is decided such that the major axis is vertical and the minor axis is horizontal.

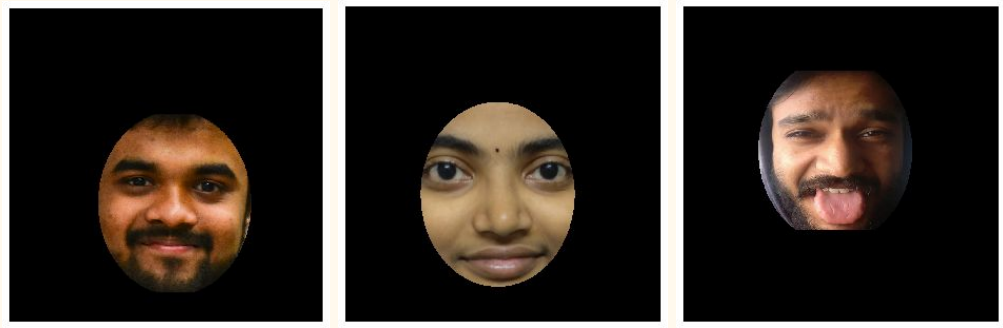
Parameter initializations:

- **Angle**(Ellipse rotation angle in degrees) -**0°**.
- **startAngle** – Starting angle of the elliptic arc in degrees-**0°**.
- **endAngle** – Ending angle of the elliptic arc in degrees-**360°**.

Explanation of Parameters:



Oval images formed using the above-mentioned process:



PART-3 (Face-Recognition)

FACE-RECOGNITION:

Face Recognition is based on geometric features of a face. Traditionally, Face recognition can be achieved through the following techniques:

1. Principle Component Analysis.
2. Linear Discriminant Analysis.
3. Local Binary Patterns.

PCA(PRINCIPLE COMPONENT ANALYSIS):

1. It is one of the ancient techniques proposed for Face Recognition.
2. This approach is based on EigenFaces. A facial image comes from a high dimensional image space and then the lower dimension space is found. The lower dimension is found through the PCA technique.
3. PCA method is achieved by performing Eigenvalue decomposition of the covariance of the data.
4. PCA best explains the variance present in the data. Given the visualization of features of the data, PCA chooses the direction of the maximum variance.
5. It involves the following key 5 steps:

. Let $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ be the training set, where each $\mathbf{x}_i$ represents a d-dimensional column vector. Compute the average of the training set as

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i. \tag{3.3}$$

Define the data matrix $\mathbf{X}$ as follows: $\mathbf{X} = [(\mathbf{x}_1 - \boldsymbol{\mu})\ (\mathbf{x}_2 - \boldsymbol{\mu})\ \cdots\ (\mathbf{x}_N - \boldsymbol{\mu})]$.

Calculate the data covariance matrix as

$$\mathbf{C} = \mathbf{X}\mathbf{X}^T, \tag{3.4}$$

where $\mathbf{X}^T$ is the transpose of matrix $\mathbf{X}$. Since $\mathbf{X}$ is a $d \times N$ dimensional matrix, the size of the covariance matrix $\mathbf{C}$ is $d \times d$.

Compute the Eigen vectors of the covariance matrix $\mathbf{C}$ by solving the following Eigen system.

$$\mathbf{C}\mathbf{E} = \lambda\mathbf{E}. \tag{3.5}$$

Any data vector $\mathbf{x}$ can be represented as a weighted sum of the Eigen vectors and these weights can be computed as $\boldsymbol{\omega} = \mathbf{E}^T\mathbf{x}$, where $\mathbf{E}^T$ is the transpose of $\mathbf{E}$. Note that $\boldsymbol{\omega} = [\omega_1, \omega_2, \cdots, \omega_d]^T$ is a d-dimensional column vector, where ω_j is the weight associated with Eigen vector $\mathbf{e}_j$, for $j = 1, 2, \cdots, d$. These weights are also known as Eigen coefficients.

PCA IMPLEMENTATION:

- The above-provided way was implemented from scratch and is named 'PCA_scratchFR.ipynb'.
- Previously computed pre-processed images were taken as the input into this PCA.
- Following are the functions implemented in the PCA:
  - ❖ **face_detection, oval:**
    - These functions are part of pre-processing.
    - They take an image as an input and returns a preprocessed image.
    - Face Detection is implemented using pre-trained model. And this implementation is based on Haar Wavelets.
  - ❖ **loadDataLabels:**
    - Parameters: Dataset path.
    - This function loads the training and testing data from the dataset.

- Returns: the train data, test data and unique_labels (these labels are encoded using Label_Encoder).

❖ **mean, X, covMat, eigenVectors, eigenWeights:**

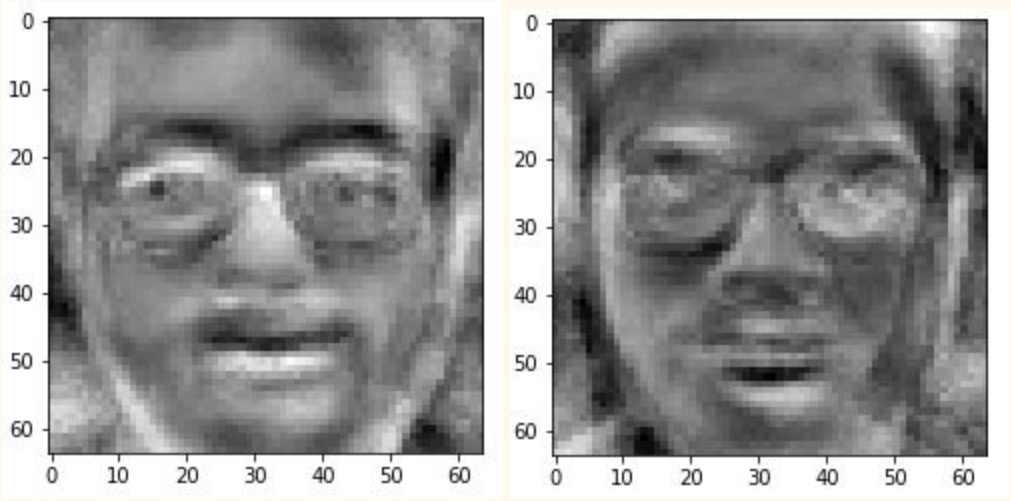
- Parameters: Flatten trained Data (flattened using FlattenData function).
- These functions take train data as input and end of it returns an Eigenvector.
- Returns: EigenVector (Later, this is used as EigenFaces).
- These are the above mentioned 5 steps that are implemented as mean, X, covMat, eigenVectors, eigenWeights.

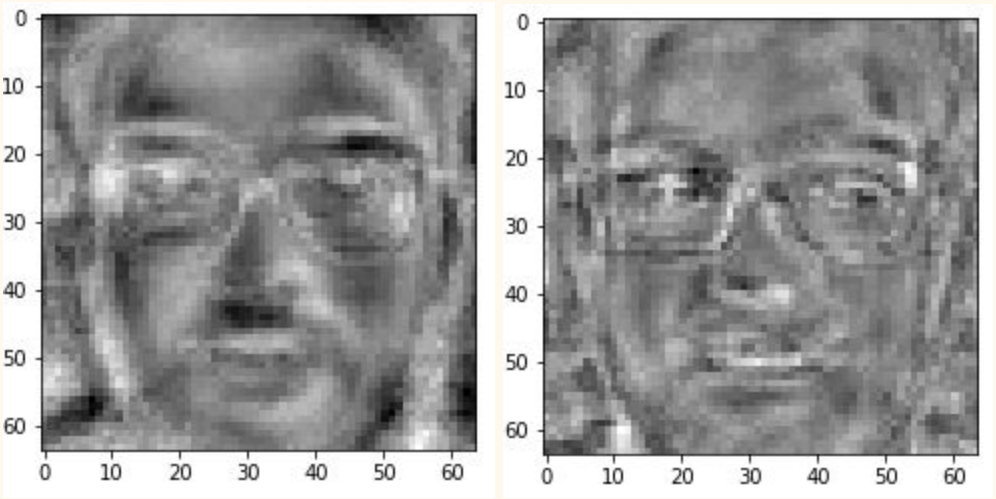
❖ **PCAfamilyEigen:**

- Takes training data as input and the above 5 steps are written down under a single function. This method returns EigenVectors.
- drawEigenFace:
  - Parameters: index, eigenvectors
  - Given the above draws an eigenFace.

Following are the sample Eigen Faces generated:

-





-

❖ **eigenWeights:**

- Parameters: EigenVectors.
- Given a set of eigenvectors, this computes a matrix that contains weights of the training data.
- Returns: Eigen Coefficients/weight matrix.

FlattenData: Takes image data as input and just converts the whole 2-D image into a single dimensional vector.

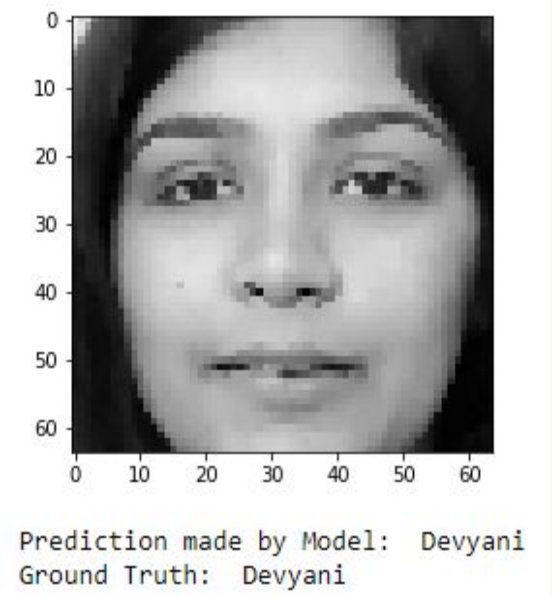
❖ **PCA_predictions:**

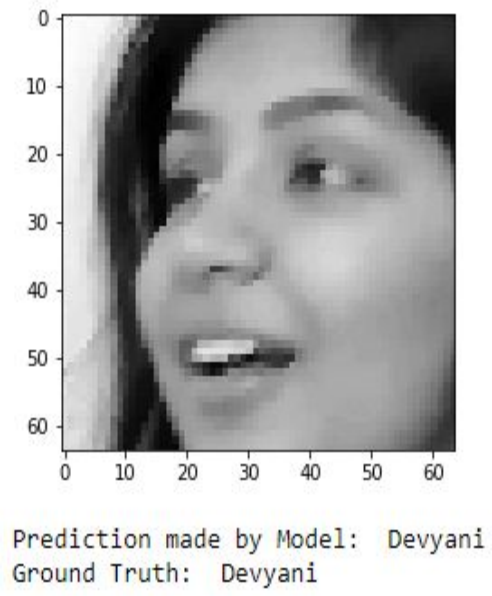
- Parameters: train data, test data, and set of Eigen Vectors that are generated from the training data.
- Computes train and test Eigen Coefficients (these are generated using the step-5 mentioned above, and both are computed based on the same eigenvectors obtained).
- Having computed the Eigen Coefficients/weight matrix of the training data and testing data, each of the testing data weights is compared with the train eigencoefficient matrix. Comparisons are made based on Euclidean distance. The lesser the Euclidean distance a test-Eigen

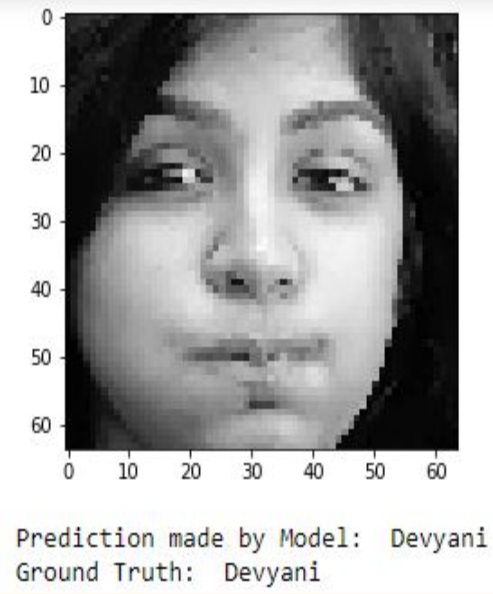
weights get matched with a train-Eigen weight that particular train label is assigned to the test data.

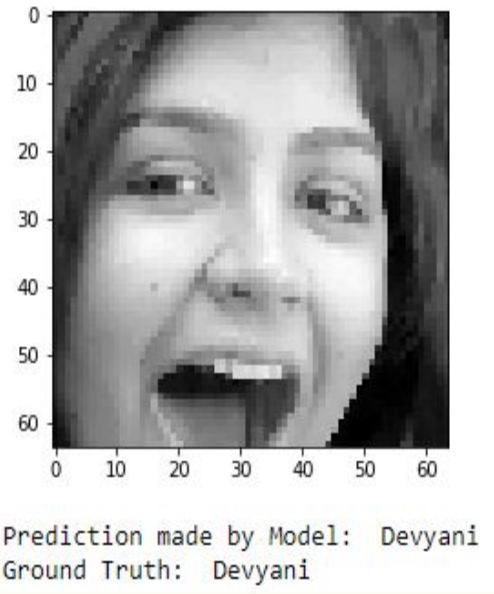
- Returns: List of predictions made.

Given an image, the following are the image predictions made:

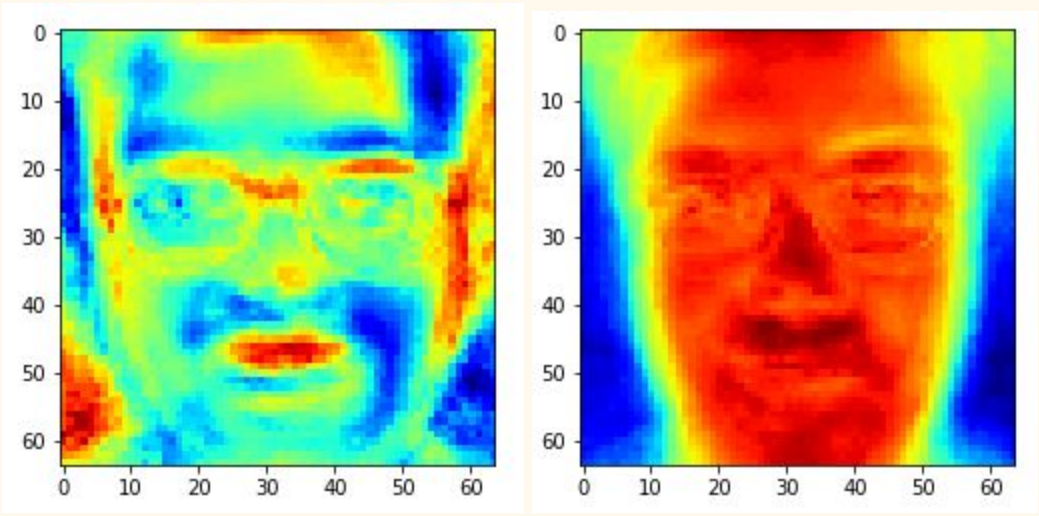




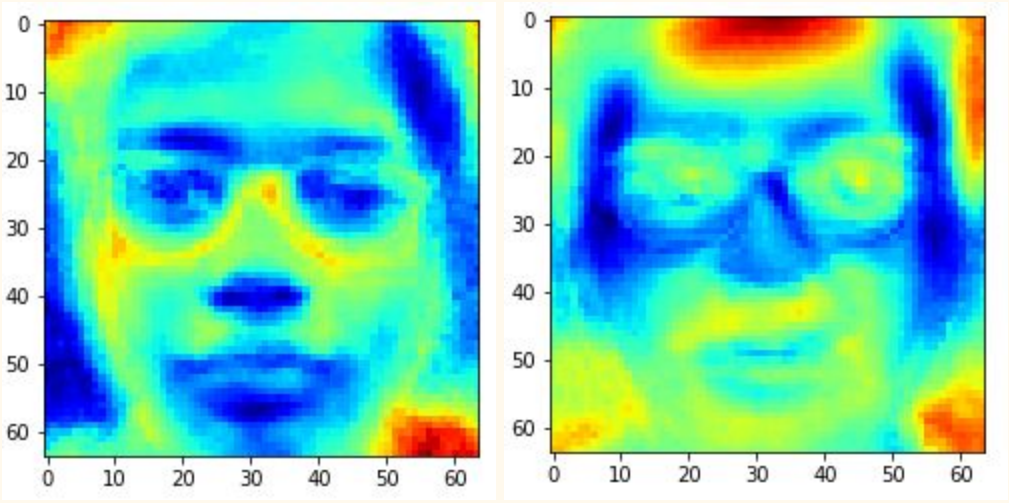




- Above are the images that are resized into 64 * 64 (We cannot use 256 * 256 due to memory constraints).
- **Following are the sample jet colour Eigen Faces generated:**
-



-



- From the above, we can observe that eigenfaces not only encode facial features but also illumination in the images.

PCA ACCURACY:

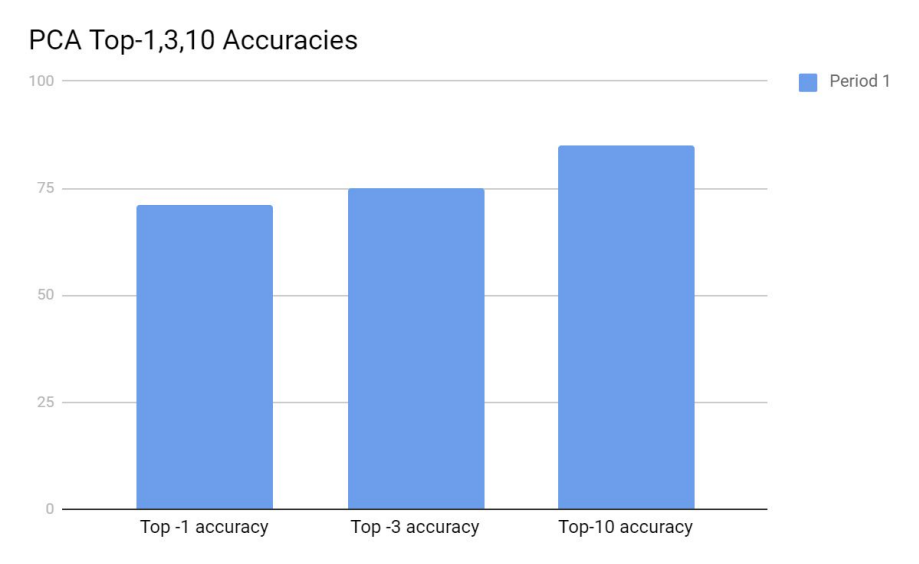
Computes the accuracy by just counting the number of Predictions that were correct wrt y_test.

The accuracy obtained: 71%.

```
PCA_accuracy(predictions, y_test)
Example(100, predictions, y_test, unique_labels)
```

```
PCA implementation Accuracy:  70.34482758620689
```

- **Below are the Top-1, 3, 10 accuracies obtained for PCA implementation:**
  - **Top-1 Accuracy: 71%.**
  - **Top-3 Accuracy: 76%.**
  - **Top-10 Accuracy: 86%.**



```
predictions_top1 = PCA_predictionN(X_train, y_train, X_test, eigenVect, 1)
PCA_accuracyN(predictions_top1, y_test, 1)
predictions_top3 = PCA_predictionN(X_train, y_train, X_test, eigenVect, 3)
PCA_accuracyN(predictions_top3, y_test, 3)
predictions_top10 = PCA_predictionN(X_train, y_train, X_test, eigenVect, 10)
PCA_accuracyN(predictions_top10, y_test, 10)
```

```
PCA implementation top- 1 Accuracy:  70.34482758620689
PCA implementation top- 3 Accuracy:  75.86206896551724
PCA implementation top- 10 Accuracy:  85.51724137931035
```

LDA(LINEAR DISCRIMINANT ANALYSIS):

- In general, this method is used to find a linear combination of features that classify into two or more classes.
- It attempts to model the difference between the classes of the data.
- This method is based on FisherFaces.
- Unlike PCA, LDA has the direction towards the better class separation of the data.
- It involves the following key 4 steps:

i.

1. Let $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \cdots, (\mathbf{x}_N, y_N)$ be the training set, where each $\mathbf{x}_i$ represents a d-dimensional column vector, $y_i \in \{1, 2, \cdots, c\}$ is the corresponding class label, and c is the number of classes. Compute the mean of each class as

$$\boldsymbol{\mu}_j = \frac{1}{N_i} \sum \mathbf{x}_i, \tag{3.6}$$

where N_j is the number of samples from class j and $j = 1, 2, \cdots, c$.

ii.

Define the within and between class scatter matrixes ($\mathbf{S}_w$ and $\mathbf{S}_b$, respectively) for the given training data as

$$\mathbf{S}_w = \sum_{j=1}^{c} \sum_{y_i=j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T, \text{ and} \tag{3.7}$$

$$\mathbf{S}_b = \sum_{j=1}^{c} N_j (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T, \tag{3.8}$$

where $\boldsymbol{\mu} = (1/N) \sum_{i=1}^{N} \mathbf{x}_i$.

iii. The LDA subspace is constructed such that it minimizes S_w and maximizes S_b, which is achieved simultaneously by maximizing $S_w^{-1}S_b$. The Eigen vectors maximizing $S_w^{-1}S_b$ can be calculated by following a similar approach as in PCA, i.e., by solving the following Eigen system.

$$S_w^{-1}S_bE = \lambda E. \quad (3.9)$$

iv. Any data vector x can be represented as a weighted sum of the above Eigen vectors and these weights can be computed as $\omega = E^T x$, where E^T is the transpose of E.

- The above-provided way was implemented from scratch and is named 'LDA_LBP Built-In.ipynb'.
- Used existing implementations for the LDA model (using the cv2 library, FaceRecognizer).

LDA ACCURACY:

The accuracy obtained for this model:~69%.

```
c2 = 0
for i in range(len(y_test)):
    if (predictionsLDA[i][0] == y_test[i]):
        c2 = c2 + 1
# print(c)
# print(c/len(y_test))
print("LDA Accuracy: ", (c2/len_y_test)*100)          #LDA Accuracy:  69
```

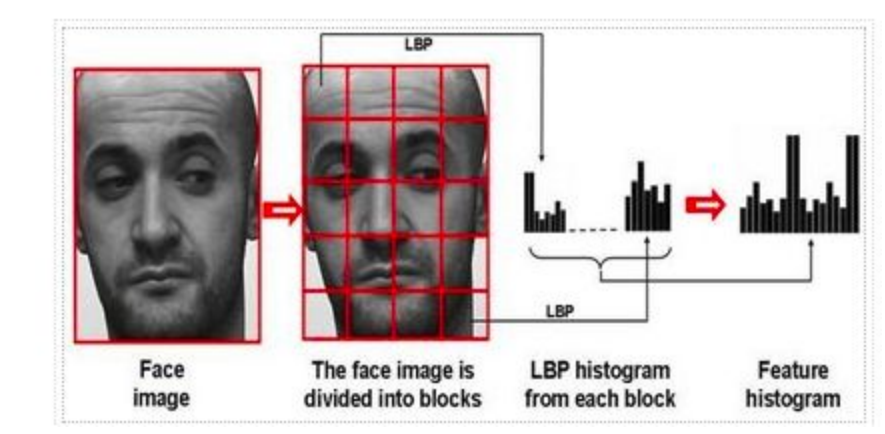
```
LDA Accuracy:  68.93939393939394
```

LBP(LOCAL BINARY PATTERNS)

LBP(Local Binary Patterns) is texture classification approach w the occurrences of the pattern codes in an image are collected into a histogram, and then classification is performed by computing simple similarities of the computed

histograms. But approach with slight modification has to be performed, as it results in a loss of spatial information .therefore the modification is that locations should be retained.

Hence facial image is divided into local regions and LBP texture descriptors are extracted from each local region independently. The descriptors are then concatenated to form a global description of the face, as shown below:



We used existing implementations for the LBP model (using the cv2 library, FaceRecognizer).

LBP ACCURACY:

The accuracy obtained for this model is: ~ 84%

```
c3 = 0
for i in range(len(y_test)):
    if (predictionsLBP[i][0] == y_test[i]):
        c3 = c3 + 1
# print(c)
# print(c/len(y_test))
print("LBP Accuracy: ", (c3/len_y_test)*100)          #LBP Accuracy:  84.0
```

```
LBP Accuracy:  84.35374149659864
```

PART-4

OPEN FACE-RECOGNITION:

Approach:

- Used face_recognition(an OpenFace library) to generate face encodings from an image.
- Facial landmarks were found for an image. Below is the image with all the landmarks drawn on it:

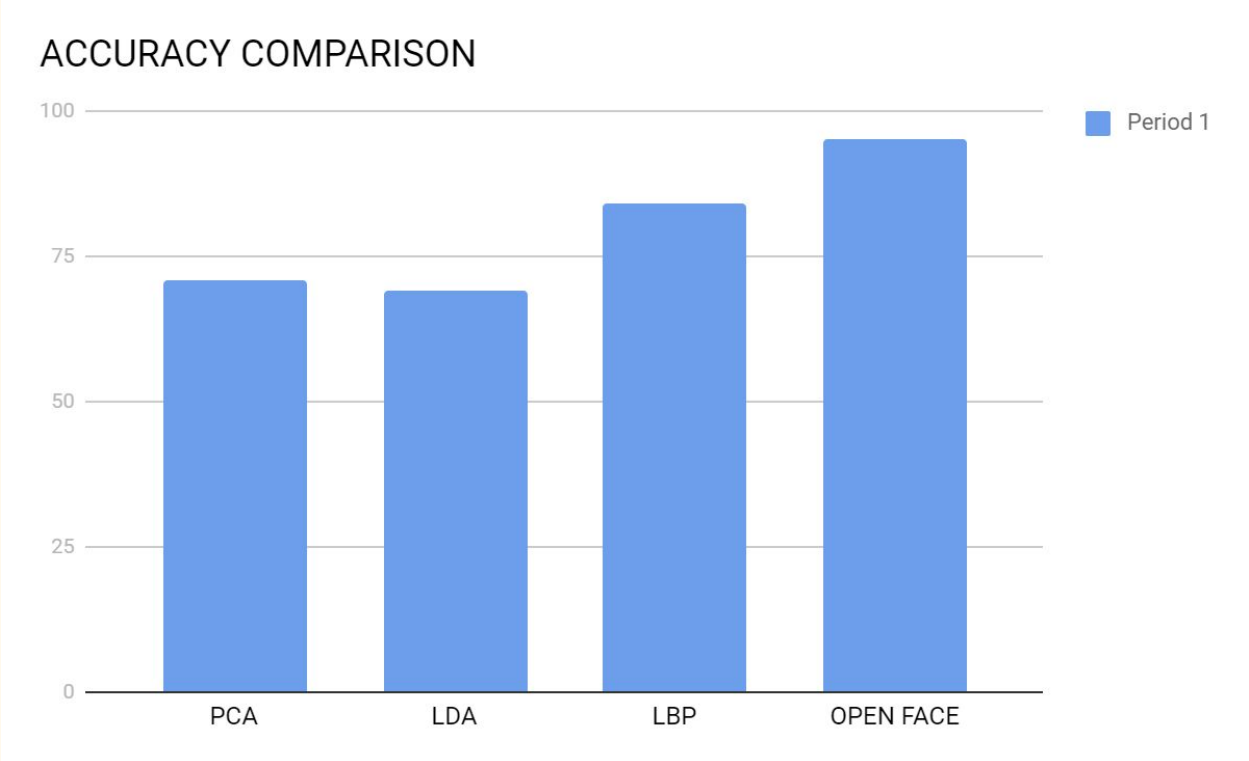
- Initially found face locations from an image and then used it as input into face encodings. And then, using inbuilt functions generated a 128-dimensional feature vector.
- This 128-dimensional feature vector is generated using previously trained Neural Networks.
- And then, image_labels along with this 128-dimensional feature vector is trained with SVM (any classification is preferred) and predicted.
- The accuracy obtained through this process: ~ 95%.

```
modelSVM = svm.SVC(kernel = 'linear')
modelSVM.fit(faceEncodings, faceEncodingLabels)
predictionsSVM = modelSVM.predict(faceEncodings_test)
print("SVM Accuracy: {}%".format(modelSVM.score(faceEncodings_test, faceEncodingLabels_test) * 100 ))
```

```
SVM Accuracy: 94.6524064171123%
```

-

Performance Comparison between OpenFace and PCA, LDA, LBP:



- Following are the accuracies obtained:
    - **PCA: 71%.**
    - **LDA: 69%.**
    - **LBP: 84%.**
    - **Open-Face: 95%**
- From the accuracies obtained we can clearly see that Open-Face outperforms all the remaining traditional approaches.
- Unlike traditional approaches like PCA, LDA, and LBP, Open-Face uses Neural-Networks. So, it stands out to be the best modern Face-Recognition technique.

PART-5

GENDER-RECOGNITION:

Proposal for a solution to Gender Recognition:

Overview of our solution to solve Gender-Recognition problem is as follows:

- Conversion of the image into contour image, contours are detected using Image gradients(using Sobel or Laplacian kernels)
- Contours provide the structural information, which generally is the characteristic to distinguish gender, also contours represent the edge fragments, curves which provides meaningful geometrical information.
- Feature extraction of the contour image is done by a descriptor called **Centrist Visual Descriptor.**
- Here transform from Image to feature vector is called Census Transform(CT).

- Here the transformation emphasizes on local image structure which maps the pixel values in the specified window to a bit string.
- Transform focusses on the relative ordering of pixel intensities in the local region and not on the magnitude of the actual intensities.

Calculation of Centrist-Visual Descriptor:

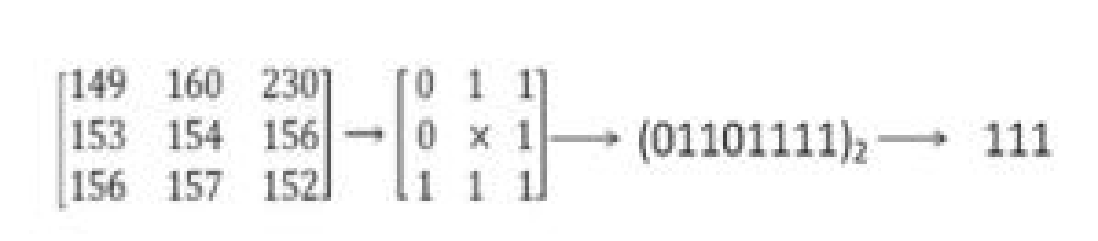
It basically gives the relative signs of comparison with the neighbours.

The calculation of descriptor is done as follows:

- **Mapping Each pixel to the corresponding n-bit string:**

Here the central pixel value is compared with eight neighbouring pixels. If the centre pixel intensity value is larger than any of its neighbours, the corresponding position value is replaced with a bit '1'.Otherwise, its value is replaced with a bit ' 0'. The bit is left shifted in each comparison, thus forming 8-bit string for a window of 3×3 size and a 32-bit string for a census window of 5×5 size.

Evaluation of transform value:



- **Calculating the image Transform:**

Each pixel in the image transform is the decimal value representation (base 10 representation) of the binary string.

- **Calculating centrist visual descriptor using created transform image:**

Histogram of the occurrence of values in transform image is the **Centrist Visual Descriptor.**

The

classical classificationn Model (like SVM)is trained with **Centrist Visual Descriptor** as a feature vector with train data and subsequently testing it on test data.Hence this Classification Model is our proposed model to solve the gender classification problem.

Reason for selection of Centrist Visual Descriptor as a feature vector:

Since the primary difference between genders is hair(including facial hair), as the Centrist Visual Descriptor signifies the relative comparison of intensities and the hair can be visually characterized by its intensity magnitude from its neighboring regions because there will be huge variance intensity gradient.Hence Centrist Visual Descriptor can be considered as a good choice for feature vector for Gender Recognition.

Prepared By:

Srujan Swaroop Gajula-(IMT2016033)

Siddarth Reddy Desu-(IMT2016037)

Durga Yasasvi Yalamarthy-(IMT2016060)

****