MACHINE LEARNING WITH PYTHON
FACE DETECTION

REPORT

Prepared by :

Shanmukha 1419
Srimayee Chilagani
Rakesh Ganji
Ameeth Hariprasad
Sudhesh Holla
Rajeev Kashetti
Nikhil Lingadhal
Nisha Rao
Nidhi Lisa
Lalith Mohan
Bryan Paes
Bhargav Raikwar
Bharathi Pravallika Sagireddy
Darshann Sringeri

FACE DETECTION:

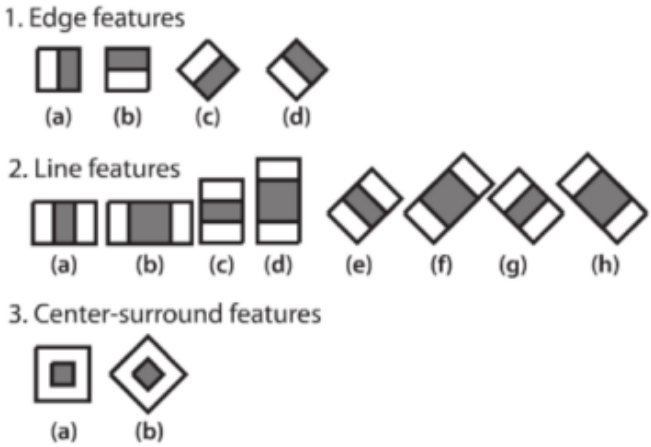
Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc.

In this project, we applied face detection to the image captured from the webcam using OpenCV with Python.

Face detection uses *classifiers*, which are algorithms that detects what is either a face or not a face in an image. Classifiers have been trained to detect faces using thousands to millions of images in order to get more accuracy. OpenCV uses two types of classifiers, LBP (Local Binary Pattern) and Haar Cascades. We have made use of Haar Cascades classifier.

Haar Cascades technique is based on Haar Wavelets . A sequence of rescaled "square-shaped" functions which together form a wavelet family or basis is what is stated by the Haar Wavelet.

It is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses machine learning techniques to get a high degree of accuracy from what is called "training data". This uses "integral image" concepts to compute the "features" detected. Haar Cascades use the **Adaboost** learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers.

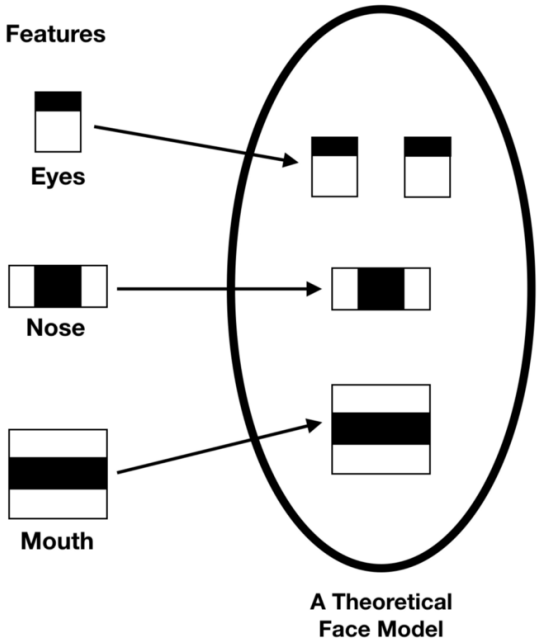


Face Detection determines the locations and sizes of human faces in arbitrary (digital) images.

In **Face Recognition**, the use of Face Detection comes first to determine and isolate a face before it can be recognized.

Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images.

We are using the module from OpenCV to coordinates (x,y,w,h) different features



detectMultiscale create a rectangle with around the face and detected in the image.

CODE -

```
import cv2
mouth_cascade = cv2.CascadeClassifier('haarcascade_mcs_mouth.xml')
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
nose_cascade = cv2.CascadeClassifier('haarcascade_mcs_nose.xml')
leftEar_cascade = cv2.CascadeClassifier('haarcascade_mcs_leftear.xml')
cap = cv2.VideoCapture(0)
while 1:
ret, img = cap.read()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:
# To draw a rectangle in a face
cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
roi_gray = gray[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]

# Detects eyes of different sizes in the input image
eyes = eye_cascade.detectMultiScale(roi_gray)
nose = nose_cascade.detectMultiScale(roi_gray)
mouth = mouth_cascade.detectMultiScale(roi_gray)
leftEar = leftEar_cascade.detectMultiScale(roi_gray)
#To draw a rectangle in eyes

for (ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)
for (ex,ey,ew,eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)
```

```
for (nx,ny,nw,nh) in nose:
cv2.rectangle(roi_color,(nx,ny),(nx+nw,ny+nh),(0,127,255),4)
for (mx,my,mw,mh) in mouth:
cv2.rectangle(roi_color,(mx,my),(mx+mw,my+mh),(0,127,255),2)
for (le_x,le_y,le_w,le_h) in leftEar:
cv2.rectangle(roi_color,(le_x,le_y),(le_x+le_w,le_y+le_h),(0,127,255),2)

# Display an image in a window
cv2.imshow('img',img)

# Wait for Esc key to stop
k = cv2.waitKey(30) & 0xff
if k == 27:
break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```