**A PROJECT REPORT ON**

**FACIAL RECOGNITION USING**

**OPEN CV**

Submitted in partial fulfillment for the requirement of the award of

TRAINING IN

Data Analytics, Machine Learning and AI using Python



*Submitted By*

**Ch.Tarun kumar reddy (**Sphoorthy Engineering college, Hyderabad**)**

*Under the guidance of*

**Mr. Bipul Shahi**

**ABSTRACT**

Abstract Efficient and accurate facial detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for face detection has increased drastically. The project aims to incorporate state-of-the-art technique for face detection with the goal of achieving high accuracy with a real-time performance. A major challenge in many of the face detection systems is the dependency on other computer vision techniques for helping the deep learning based approach, which leads to slow and non-optimal performance. In this project, we use a completely deep learning based approach to solve the problem of face detection in an end-to-end fashion. The network is trained on the most challenging publicly available dataset (Haarcascade).The resulting system is fast and accurate, thus aiding those applications which require face detection.

**INTRODUCTION**

Face detection is a type of application classified under “computer vision” technology. It is the process in which algorithms are developed and trained to properly locate faces or objects (in object detection, a related system), in images. These can be in real time from a video camera or from photographs. An example where this technology is used are in airport security systems. In order to recognize a face, the camera software must first detect it and identify the features before making an identification. Likewise, when Facebook makes tagging suggestions to identify people in photos it must first locate the face. On social media apps like Snapchat, face detection is required to augment reality which allows users to virtually wear dog face masks using fancy filters. Another use of face detection is in smartphone face ID security.

In this project, I implemented a system for locating faces in digital images. These are in JPEG format only. Before we continue, we must differentiate between face recognition and face detection. They are not the same, but one depends on the other. In this case face recognition needs face detection for making an identification to “recognize” a face. I will only cover face detection using Haar cascade.

**Technology and Concepts**

**Machine Learning**

Learning algorithms are widely used in computer vision applications. Before considering image related tasks, we are going to have a brief look at basics of machine learning.

Machine learning has emerged as a useful tool for modelling problems that are otherwise difficult to formulate exactly. Classical computer programs are explicitly programmed by hand to perform a task. With machine learning, some portion of the human contribution is replaced by a learning algorithm. As availability of computational capacity and data has increased, machine learning has become more and more practical over the years, to the point of being almost ubiquitous.

It can be used in two ways:

* *Supervised Learning*
* *Unsupervised Learning*

**Computer vision**

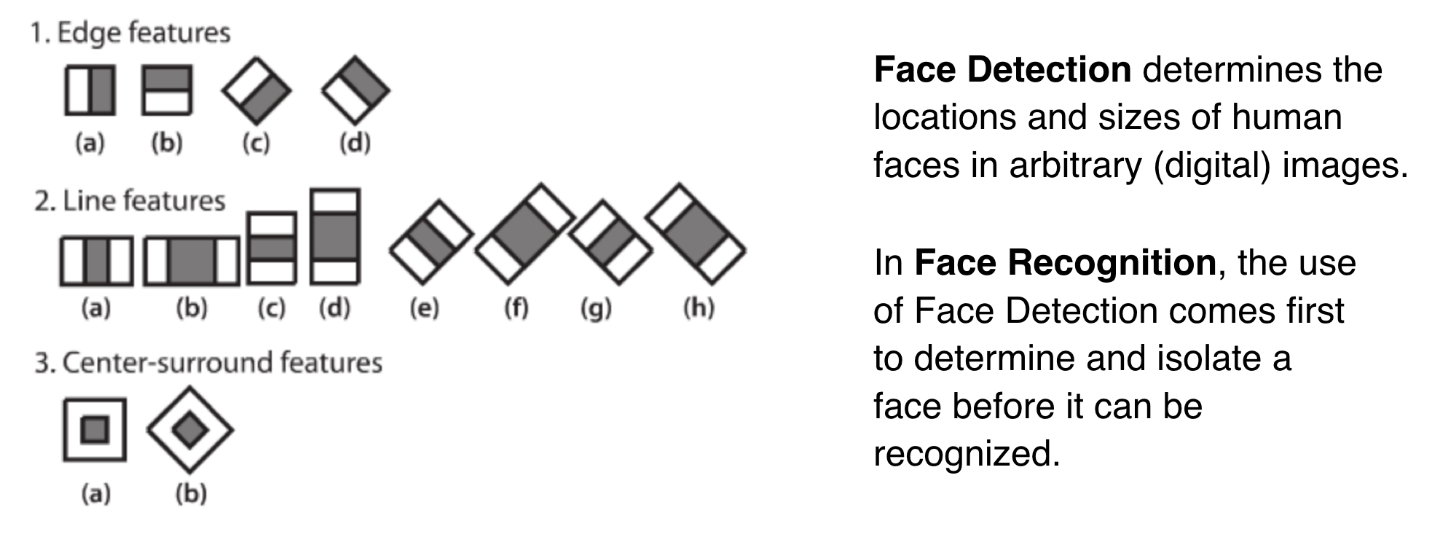
Computer vision deals with the extraction of meaningful information from the contents of digital images or video. This is distinct from mere image processing, which involves manipulating visual information on the pixel level. Applications of computer vision include image classification, visual detection, 3D scene reconstruction from 2D images, image retrieval, augmented reality, machine vision and traffic automation.

2.1 Description of the dataset

A *Haar Cascade*is based on “Haar Wavelets” which Wikipedia defines as:

***A sequence of rescaled “square-shaped” functions which together form a wavelet family or basis.***

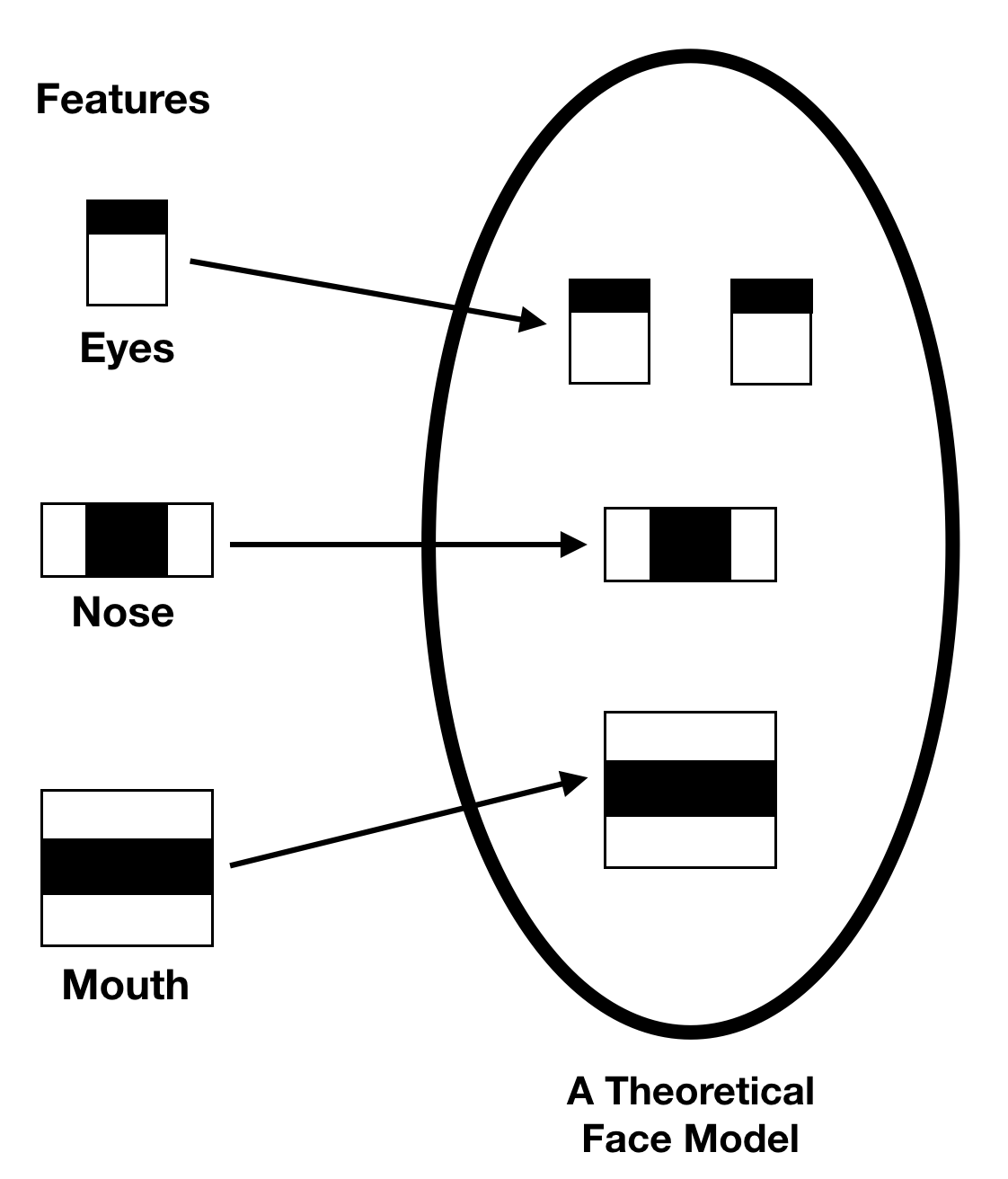
It is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses machine learning techniques to get a high degree of accuracy from what is called “training data”. This uses “integral image” concepts to compute the “features” detected. Haar Cascades use the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



This is a brief illustration of Features Extraction and the difference between Face Detection and Face Recognition. Face detection is about locating, while face recognition is about identifying.

**Feature Extraction**

As I mentioned earlier, Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images. This process in the algorithm is feature extraction.



The training data used in this project is an XML file called:

haarcascade\_frontalface\_default.xml

## Running OpenCV

For this project I prepared a directory where I dumped all the files needed. You will need to put in this directory the following:

* detect\_face\_video.py (the name I gave to the Python program that contains code. This name can be changed.)
* haarcascade\_frontalface\_default.xml (Haar Cascade training data).

We are going to use the **detectMultiscale** module from OpenCV. What this does is create a rectangle with coordinates (x,y,w,h) around the face detected in the image. This contains code parameters that are the most important to consider.I used the numpy and opencv-python modules for this.

**scaleFactor:** The value indicates how much the image size is reduced at each image scale. A lower value uses a smaller step for downscaling. This allows the algorithm to detect the face. It has a value of x.y, where x and y are arbitrary values you can set.

**minNeighbors:** This parameter specifies how many “neighbors” each candidate rectangle should have. A higher value results in less detections but it detects higher quality in an image. You can use a value of X that specifies a finite number.

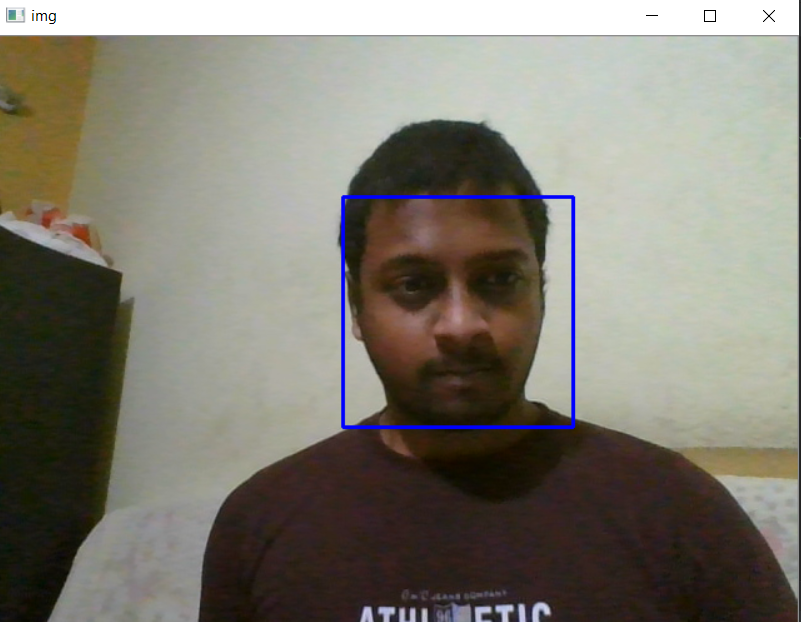
**minSize:** The minimum object size. By default it is (30,30). The smaller the face in the image, it is best to adjust the minSize value lower.

To run OpenCV, I executed it from the command prompt under the working directory where i dumped all your files.



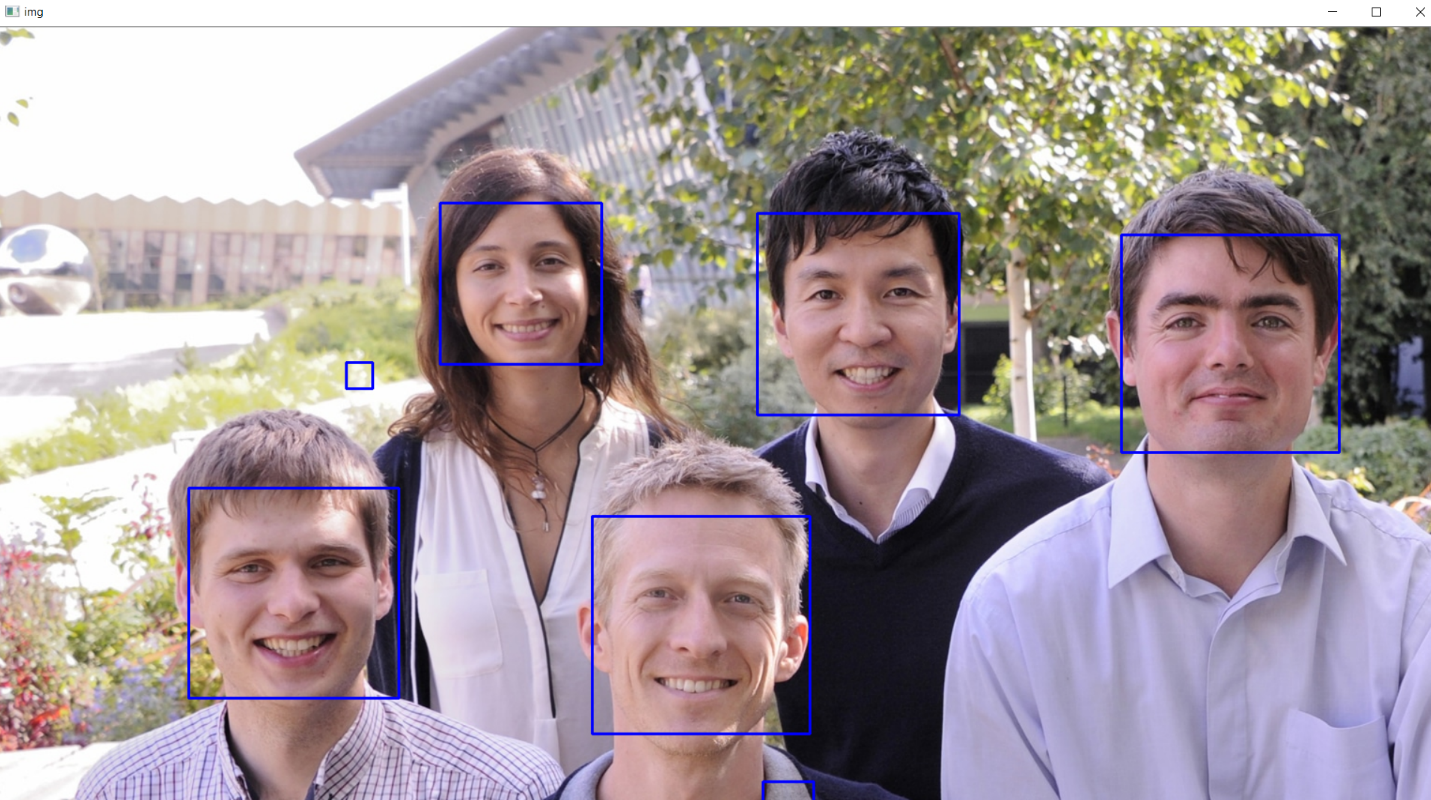
This runs on the webcam whereas detect\_face\_image.py asks an input image to scan and show the result.

**Result**

****

**Result while running detect\_face\_image.py on many faces**

Running this on the test image in the project folder the output is:



**Scope of improvement:**

More adjusting of weights can be done and can be improved by balancing and changing the weights in the cascade file.

**Bibliography**

* [https://www.kaggle.com](https://www.kaggle.com/)