

Machine learning

Project Report



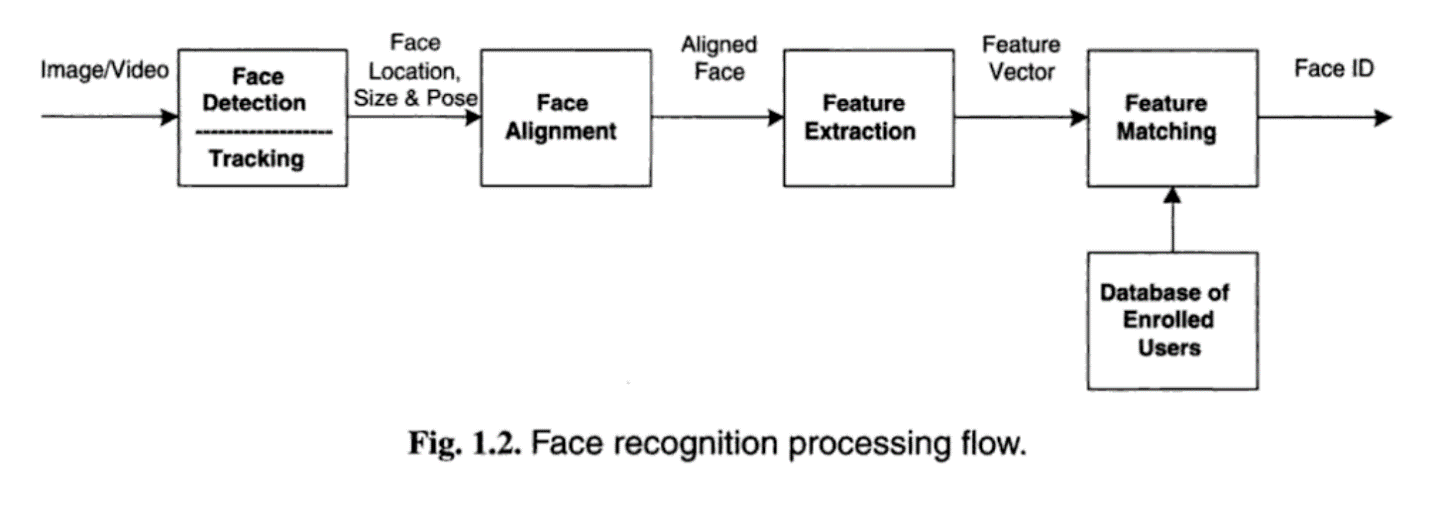
January 8, 2021

task-03

# Altamash Yar Khan Syed Abu Turab

INTRODUCTION

Our Project was based on Face recognition. Facial recognition is a biometric technology that uses distinguishable facial features to identify a person. Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Due to the advancements in face detection technology, it is now possible to detect faces in an image or video, regardless of head pose, lighting conditions, and skin color. Face detection applications use algorithms that determine whether images are positive images (i.e. images with a face) or negative images (i.e. images without a face). To be able to do this accurately, the algorithms must be trained on huge datasets containing hundreds of thousands of face images and non-face images.



APPROACH

There are multiple approaches in order to perform face recognition:

1. Deep Learning
2. FaceNet
3. OpenCV
4. Principal Component Analysis.

**Task-01 :PCA approach**.:

The Principal Component Analysis (PCA) is one of the most successful techniques that have been used in image recognition and compression. PCA is a statistical method under the broad title of factor analysis. The purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables.

The jobs which PCA can do are prediction, redundancy removal, feature extraction, data compression, etc. Because PCA is a classical technique which can do something in the linear domain, applications having linear models are suitable, such as signal processing, image processing, system and control theory, communications.

PCA computes the basis of a space which is represented by its training vectors. These basis vectors, actually are eigenvectors, computed by PCA are in the direction of the largest variance of the training vectors.  
  
Once the eigenfaces have been computed, several types of decision can be made depending on the application. What we call face recognition is a broad term which may be further specified to one of following tasks:

• identification where the labels of individuals must be obtained

• recognition of a person, where it must be decided if the individual has already been seen

• categorization where the face must be assigned to a certain class.

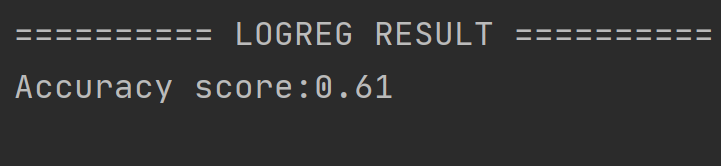
These eigenvectors are obtained from covariance matrix of a training image set. The weights are found out after selecting a set of most relevant Eigenfaces. Recognition is performed by projecting a test image onto the subspace spanned by the eigenfaces and then classification is done by measuring minimum Euclidean distance.

Applied Logistic Regression to test and train the data set and also get the accuracy of the output.

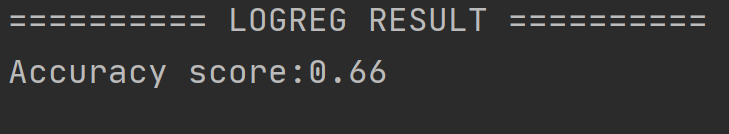
* **Python Outputs:**

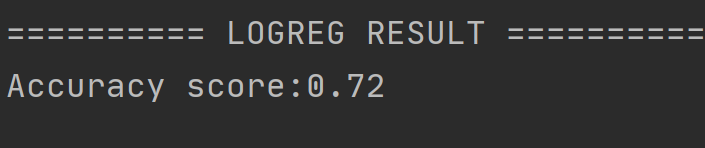
N components are basically the number of Eigen Faces Selected from the Covariance with reduced dimensionality images.

n components value = 5

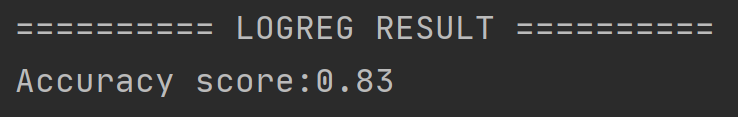


n components value = 10

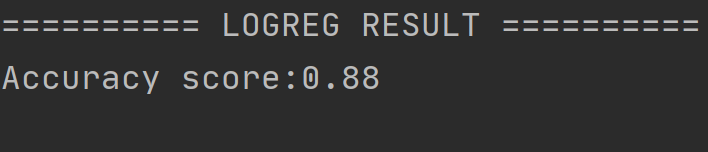


N component=20  
  


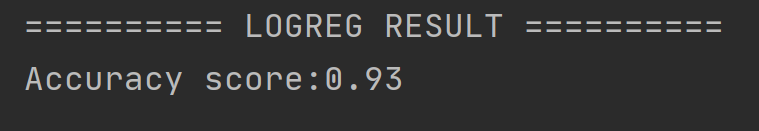
N component=30



N component =40

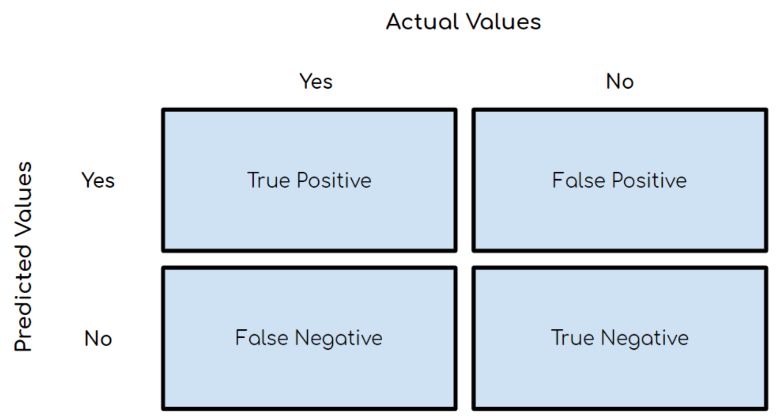


N component =50



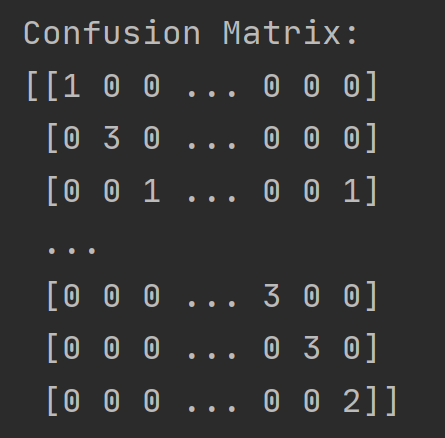
N component selected =50

Confusion Matrix

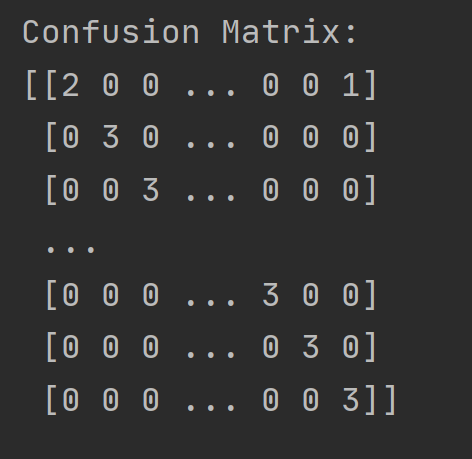


* **True Positive (TP)**: Outcome where the model correctly predicts the positive class.
* **True Negative (TN)**: Outcome where the model correctly predicts the negative class.
* **False Positive (FP)**: Also called a **type 1 error**, an outcome where the model incorrectly predicts the positive class when it is actually negative.
* **False Negative (FN)**: Also called a **type 2 error**, an outcome where the model incorrectly predicts the negative class when it is actually positive.
* Accuracy = (TP+TN)/total

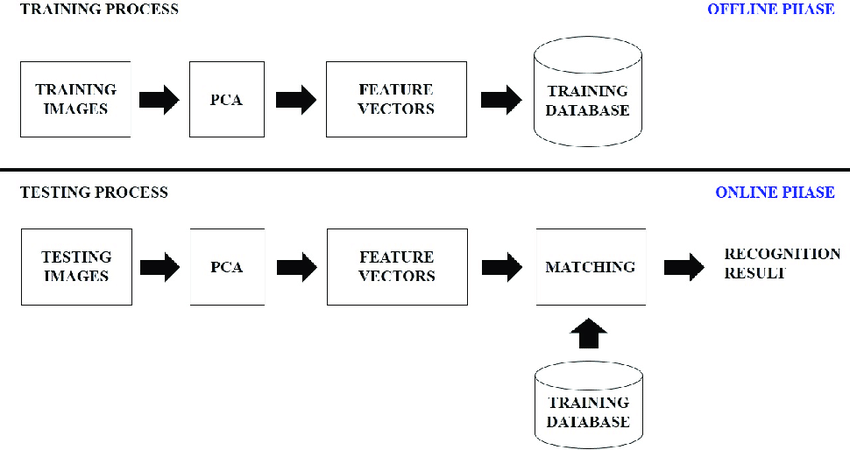
------------------------------------------------------------------------------------------------------------------------------------------

Output: **confusion matrix for n components =10**

Accuracy: 66%

Output: **confusion matrix for n components =50**

Accuracy: 93%



**Task-02 :Deep Learning approach**.

For deep learning we used a Multi-task Cascaded Convolutional Networks (MTCNN)

MTCNN is a framework developed as a solution for both face detection and face alignment. The process consists of three stages of convolutional networks that are able to recognize faces and landmark location such as eyes, nose, and mouth.

The network uses a cascade structure with three networks; first the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Network or P-Net) proposes candidate facial regions by making bounding boxes, the second model (Refine Network or R-Net) filters the bounding boxes ( non-maximum suppression (NMS) is used to filter the candidate bounding boxes proposed by the first-stage P-Net prior to providing them to the second stage R-Net model. ), and the third model (Output Network or O-Net) proposes facial landmarks

Side note:

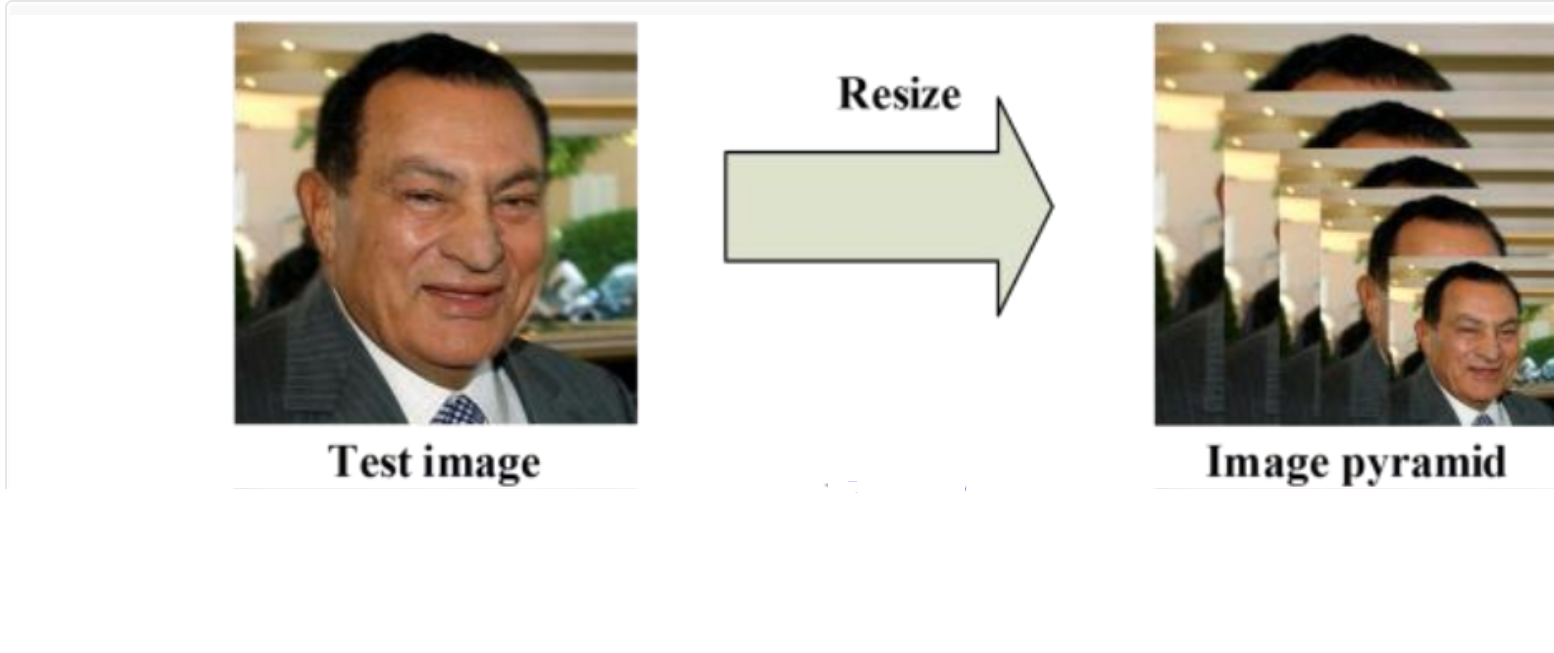
* There are five landmarks: left eye, right eye, nose, left mouth corner and right mouth corner
* NMS: technique/ a class of algorithms to select one entity (e.g. bounding boxes) out of many overlapping entities

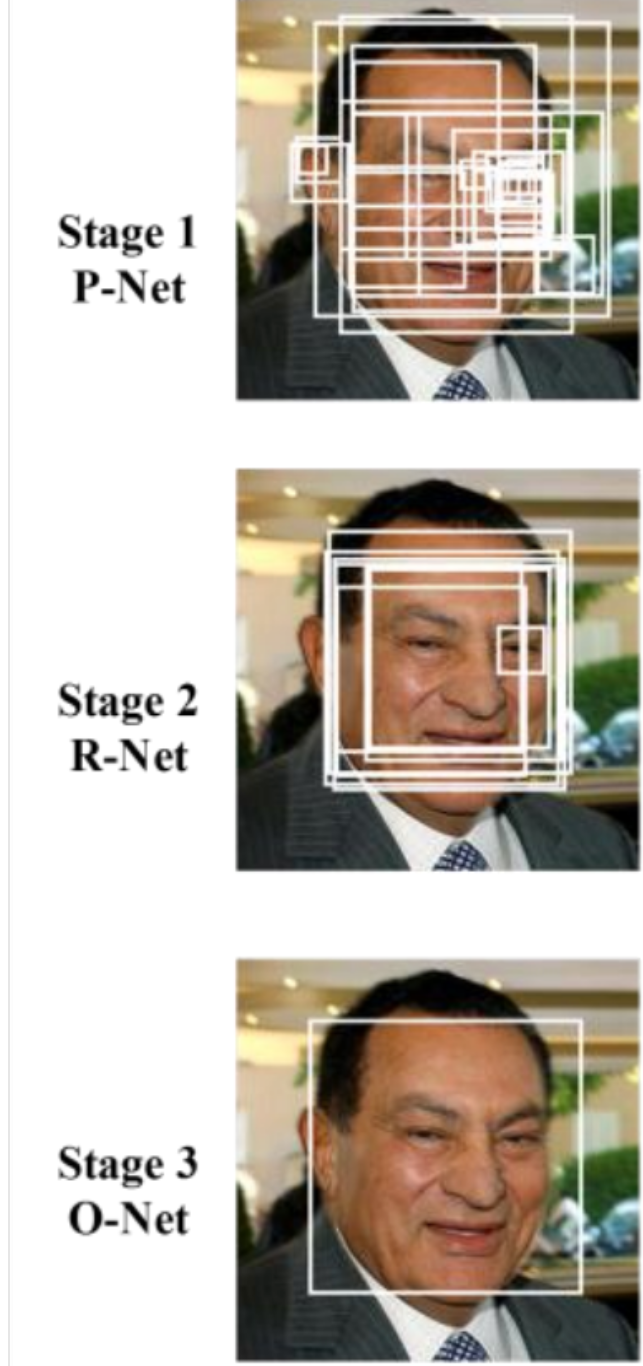
PNET:

This Proposal Network is used to obtain candidate windows and their bounding box regression vectors.

R-Net:   
further reduces the number of candidates, performs calibration with bounding box regression and employs non-maximum suppression (NMS) to merge overlapping candidates.

O NET:  
this Output Network aims to describe the face in more detail and output the five facial landmarks’ positions for eyes, nose and mouth.





The model is called a multi-task network because each of the three models in the cascade (P-Net, R-Net and O-Net) are trained on three tasks, e.g. make three types of predictions; they are: face classification, bounding box regression, and facial landmark localization.

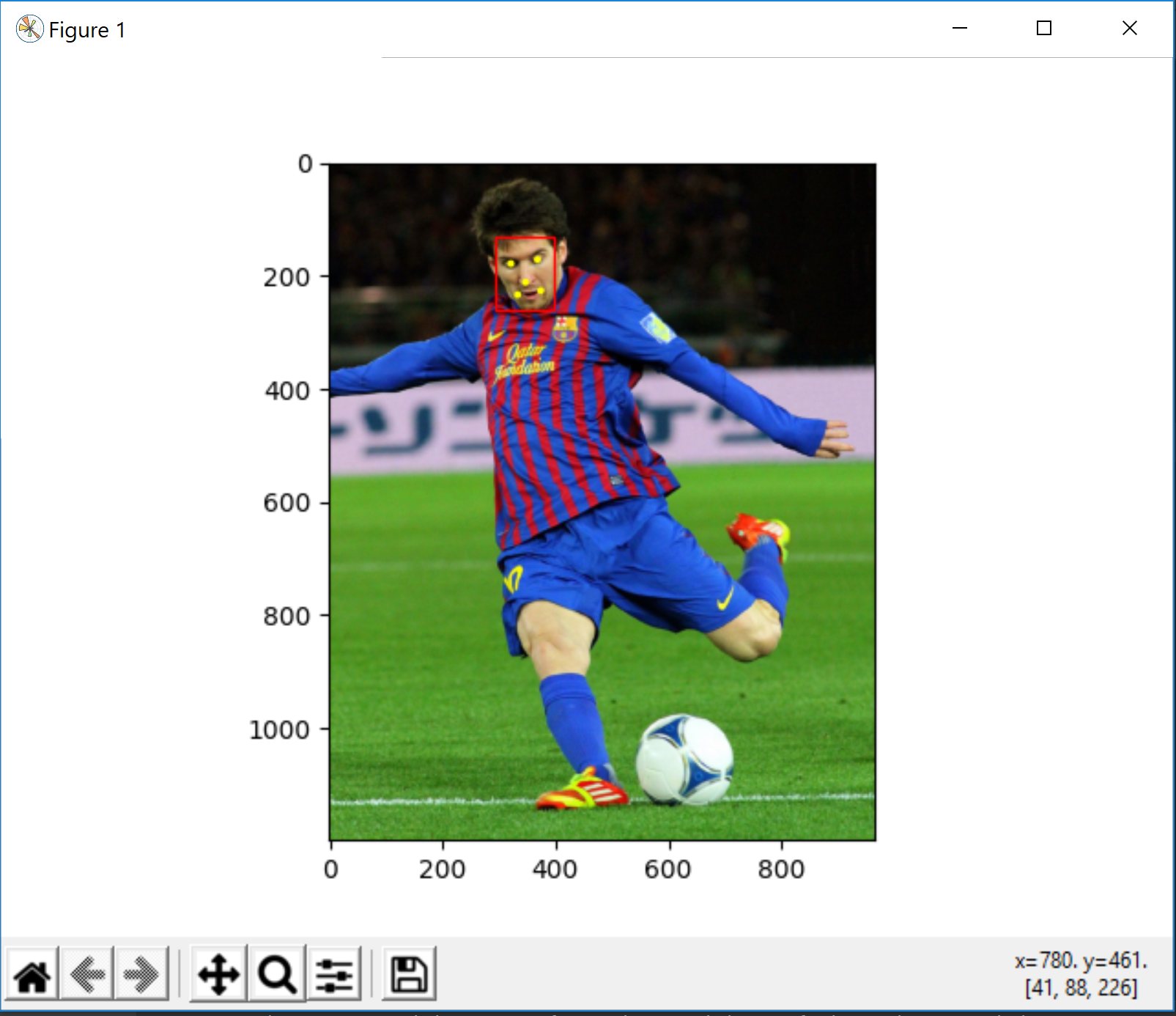
But this was the predefined model that is used to recognize the face and its features.

* **Pyhton Functions for MTCNN:**

|  |  |
| --- | --- |
| Imread() | **imread()** Function: The **imread()** function in **pyplot** module of **matplotlib** library is used to read an image from a file into an array |
| **imshow()** | in pyplot module of matplotlib library is used to display data as an image |
| Gca() | Get the current [Axes](https://matplotlib.org/3.1.1/api/axes_api.html#matplotlib.axes.Axes) instance on the current figure matching the given keyword args, or create one. |
| Detect.faces() | it basically returns coordinates of a rectangle where the MTCNN algorithm detected faces. The “box” value above returns the location of the whole face, followed by a “confidence” level and key points. |

* *box*‘: Providing the *x*, *y* of the bottom left of the bounding box, as well as the *width* and *height* of the box.
* ‘*confidence*‘: The probability confidence of the prediction.
* ‘*keypoints*‘: Providing a dict with dots for the ‘*left\_eye*‘, ‘*right\_eye*‘, ‘*nose*‘, ‘*mouth\_left*‘, and ‘*mouth\_right*‘.
* {'box': [186, 71, 87, 115], 'confidence': 0.9994562268257141, 'keypoints': {'left\_eye': (207, 110), 'right\_eye': (252, 119), 'nose': (220, 143), 'mouth\_left': (200, 148), 'mouth\_right': (244, 159)}}
* {'box': [368, 75, 108, 138], 'confidence': 0.998593270778656, 'keypoints': {'left\_eye': (392, 133), 'right\_eye': (441, 140), 'nose': (407, 170), 'mouth\_left': (388, 180), 'mouth\_right': (438, 185)}}

OUTPUT:



**For Training the model using Deep Learning:**

To Train The Model We had to Use TensorFlow and Keras library.

1. It creates an image classifier using a **keras.Sequential()** model
2. loads data using **preprocessing.image\_dataset\_from\_directory**()
3. for compiling the model **model.compile()**
4. for testing/training it **model .fit()**.

**The Objectives was to** :

1. Examine and understand data
2. Build an input pipeline
3. Build the model
4. Train the model
5. Test the model
6. Improve the model and repeat the process.

**Issues:**

The issues I faced during training the model through deep learning were quite a few.

* I couldn’t properly finish the task, couldn’t get the resulting output showing face classification/recognition. Because:
* I did not had the version that could actually install tensor flow and run my program.
* Had an error, Which I couldn’t resolve.
* Since program wasn’t running successfully couldn’t find the accuracy aswell.
* Used a predefined model to recognize the face using mtcnn.

CONCLUSION

With PCA we were able to properly test and train the data set in order to recognize the face by extracting features through eigen vectors and forming average face. In addition to test and train the models, the logistic regression library also helped us determine the accuracy of the output.  
Whereas, The facial recognition through mtcnn we were able to use predefined models to test the outputs. But we weren’t able to train the model before hand using tensorflow and keras library due to some issues of the version of the python script that I was using.