AI Image Detection using python

How to detect Face in Python

Step 1

The first one is the getting the database which contain a crapload of faces which help us to detect what the face features are like to be in the picture
that the machine learning thing

Step 2

- Make them all black and white
- Because the algorithm not understand what the picture is and can understand only the black and white colour format that is grayscale so that's why we are converting all the pictures in the database into black and white colour

Step 3

- Train the algorithm to detect faces
- This algorithm will detect a difference basis whether they wear glass ,have mouth closed ,mouth open ,are sad ,are neutral in their faces

If you train that data really enough then you can find the difference between any body recognise babies females meals even monkeys

Contain all the pretrained face detection data set
Only front face

Haarcascade_frontface_alt_tree.xml
https://github.com/opencv/opencv/tree/master/data/haarcascades

Load some pretrained data on face frontals from open CV Haar cascade algorithm

```
# machine learning pretrained model
# classifier means detect in this case face
trained_face_data                                                    =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Choose image to detect face

```
# choose an image to detect faces in
# reading image array like bigmatrix with bunch of numbers pixels
img = cv2.imread('RDJ.jpg')
# img = cv2.imread('JT.jpg')
```

Convert to Grayscale

```
# beacuse algo require grayscale (range from black to white instead of
rgb)
# written backwards bgr=rgb
```

```
grayscaled_img=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Detect face

```
detect faces
# detect objects of diff size in input image.Detected objects are returned as a list of rectangles
#                                   CascaseClassifier::detectMultiScale
trained_face_data.detectMultiScale   (detect   all   the   faces   with multiscale thing)
# detect whether img is small or big it'll check reln bn eyes to nose to mouth
# whether its smaller or multiple of its type  detect all of them

face_coordinates = trained_face_data.detectMultiScale(grayscaled_img)
print(face_coordinates)
```

OP
x1,y1=top left
W,h =width height
[[101 168 394 394]]=[x1,y1;w , h]

Also we can detect for the cars dogs etc
Return coordinates of those green rectangles And once we have those coordinates we are able to draw rectangle on our image

Draw rectangle around the face
cv2.rectangle(image , (x1,y1),(x2,y2),(B,G,R), width of rectangle)(0,255,0) = green

```
# Draw rectangles around the faces for 1 face
for( x, y ,w ,h) in face_coordinates:
     cv2.rectangle(img, (x ,y), (x+w ,y+h), (0,255,0), 2)
```

1 Face detect numerical

```
# for 1 face
cv2.rectangle(img, (101 ,168), (394+101 ,394+168), (0,255,0), 2)
```

For n faces

```
# (x,y,w,h)=face_coordinates[0] # for 1 person image
# (x,y,w,h)=face_coordinates[1] # for 2 person image
# cv2.rectangle(img, (101 ,168), (394+101 ,394+168), (0,255,0), 2)
```

Randrange for using random color

- **128 above color are brighter**
- **Till range-1 means randrange(256-1) in system**

```
From random import randrange
cv2.rectangle(img,(x,y),(x+w,y+h),(randrange(128,256),randrange(256),ra
ndrange(256)), 2)
```

Show image

```
# show the image (waitkey() :  to pause the screen\)
# imshow('TITLE',image)
cv2.imshow('Face Detector : ',grayscaled_img)
cv2.waitKey()
```

For webcam

- Video capture is used to connect with the webcam
- 0 value indicating that the default camera in the system
- Also we cannot add **videoname.MP4** as a parameter instead of 0

```

webcam=cv2.VideoCapture(0)
```

If we don't iterate over frames forever times then it will only give us One Frame which Goa so just like static images

```
# Iterate forever over frames
while True:
```

Read function => return two values

- one is if the frame is successfully return that is Boolean value (True/False)
- the other one is the image itself which is coming from the webcam

```
    # read the current frame
    successful_frame_read,frame = webcam.read()
```

Body same as previous

```
   # convert to grayscale
    grayscaled_img=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces
                                              face_coordinates           =
trained_face_data.detectMultiScale(grayscaled_img)

    # Draw rectangles around the faces
    for ( x, y ,w ,h) in face_coordinates:
    # (x , y, w , h) in face_coordinates[0]
      cv2.rectangle(frame,(x,y),(x+w,y+h),(randrange(128,256),randrange
(256),randrange(256)), 2)
        # cv2.rectangle(frame, (x ,y), (x+w ,y+h), (0,255,0), 2)
```

To get out from The Infinite loop you have to use something

- In this case we are using Q / q key which when pressed by the user then it will quit from loop with the help of if condition
- If I don't press something then it will return none
- Q ASCII value is 81
- q ASCII value is 113

```
key = cv2.waitKey(1)
    # wait for 1 millisec and hit key by itself

    #stop if Q key is pressed
    if key==81 or key==113:
        break
```

- After streaming close the webcam data
- That would be outside of the loop

```
    # Relase the videocaptures
    webcam.release()
```

None in python

Haar cascade algo

- Haar cascade machine algorithm is chain of machine algorithm
- The images is being passed through And funnel down to its final position until it find the exact square where the face is in the image
- Looks every square and every size of images passes through this in machine learning model
- If you pass all the cascade and you get to the bottom then you can tell that this is Close to face

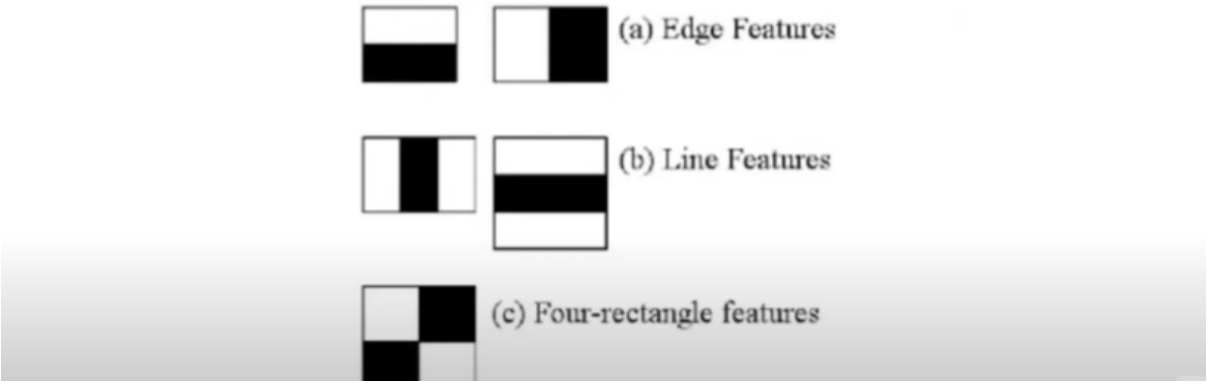
Haar features

a. **Edge feature**
b. **line feature**
c. **four rectangle feature**

First features of haar algorithm

- Haar features as rudimentary building blocks That is all the block in a diagram haar features
- Place these diagram block in the certain portion of image which have face in it or doesn't have face in it and find the relationship between the black and white region/portion
- This allowed us to approximate the relationship of this pixel within this block
- From the block we overlay to the image and we can find the relationship
- Line features edge feature and the last for the most elementary one four rectangle feature by which it find the relation between the region
- By using the block we can relate of the left have some relationship with the right ; you do the three things we can also to the horizontal line feature which is in four rectangle features
- If you chain these Together over and over then combination of this five blocks will give you a template for the face

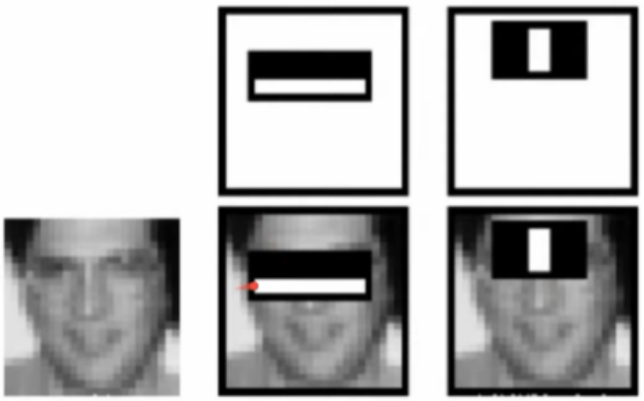
Haar Features (aka rudimentary building blocks)



- you can see in the fourth image we can see that the eyes have more darker colour as compared to its cheeks or nose
- also in the 5 image we can see that the Eyes images is black and the middle portion between the eyes is white it define the relationship between dark place as well as light place in faces
- it means that it only detect the darker as well as the lighter part in the faces

Working on 👉 Python Face Detection 🚀

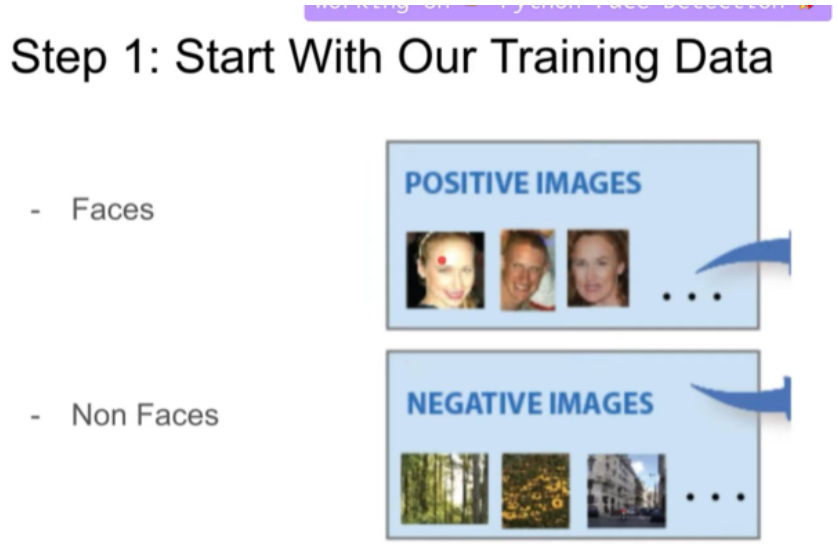
How A Haar Feature Is Applied To An Image



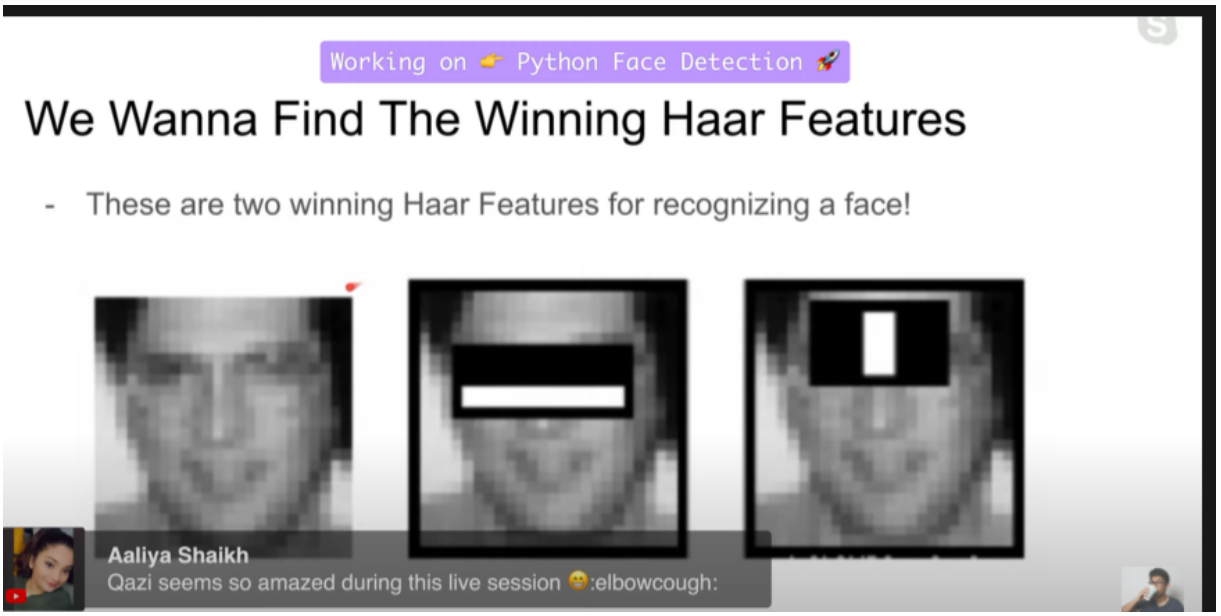
- Like this we layer thousands of Haar cascade matches together then add up a conclusion define its face
- For that reason we are using grayscale because colour is irrelevant when we are just looking for brightness

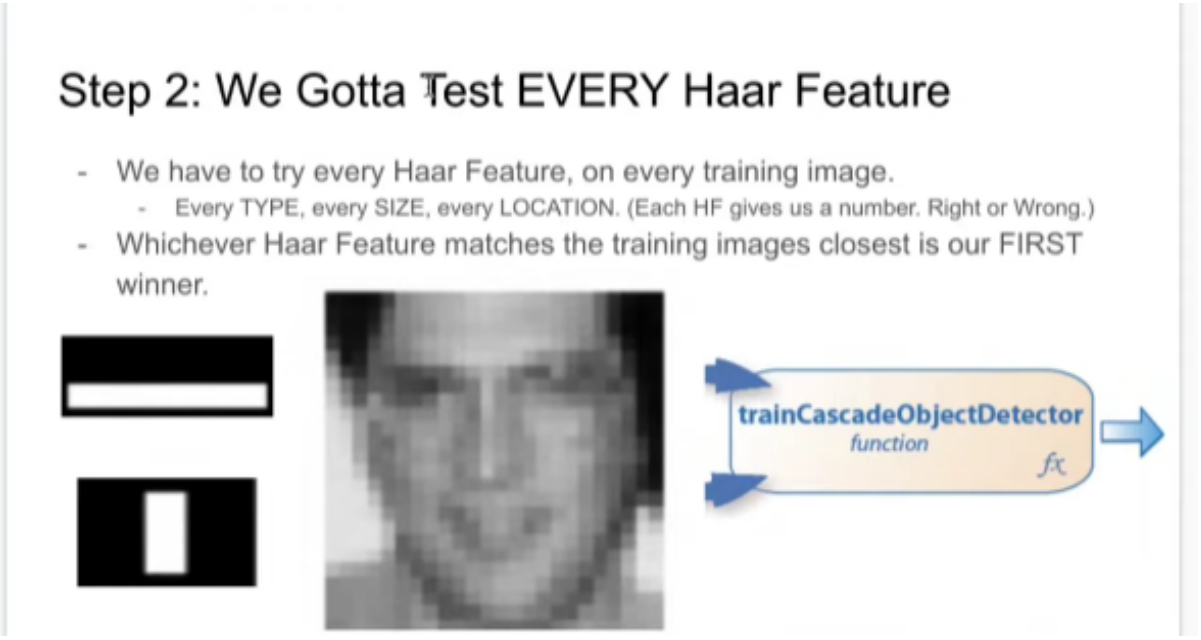
Lets Train a face detectors

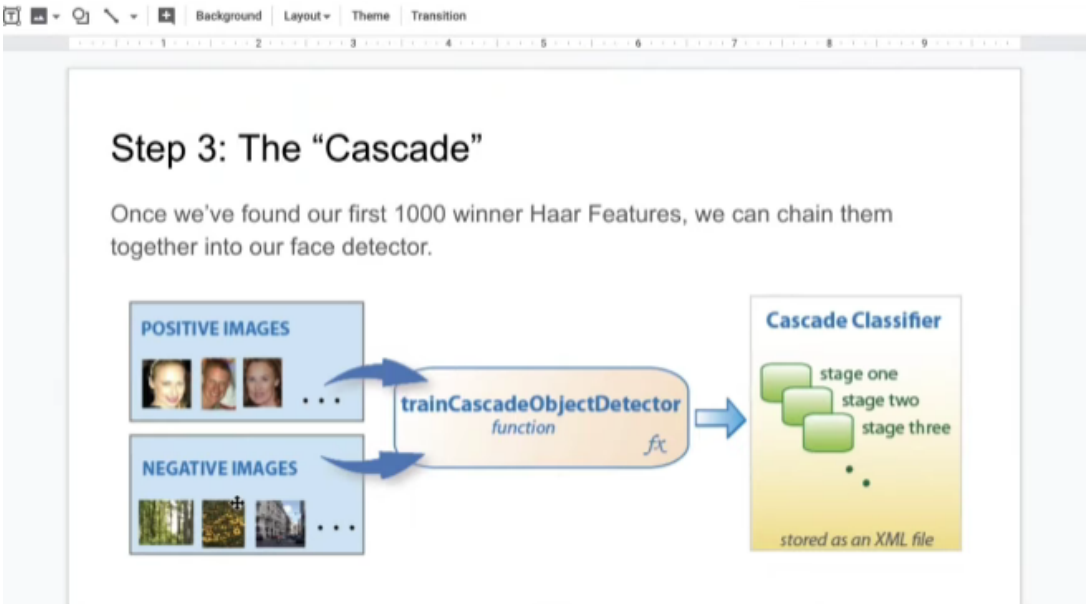
Positive measures are those which have spaces and the negative images are those in which there is no human object face



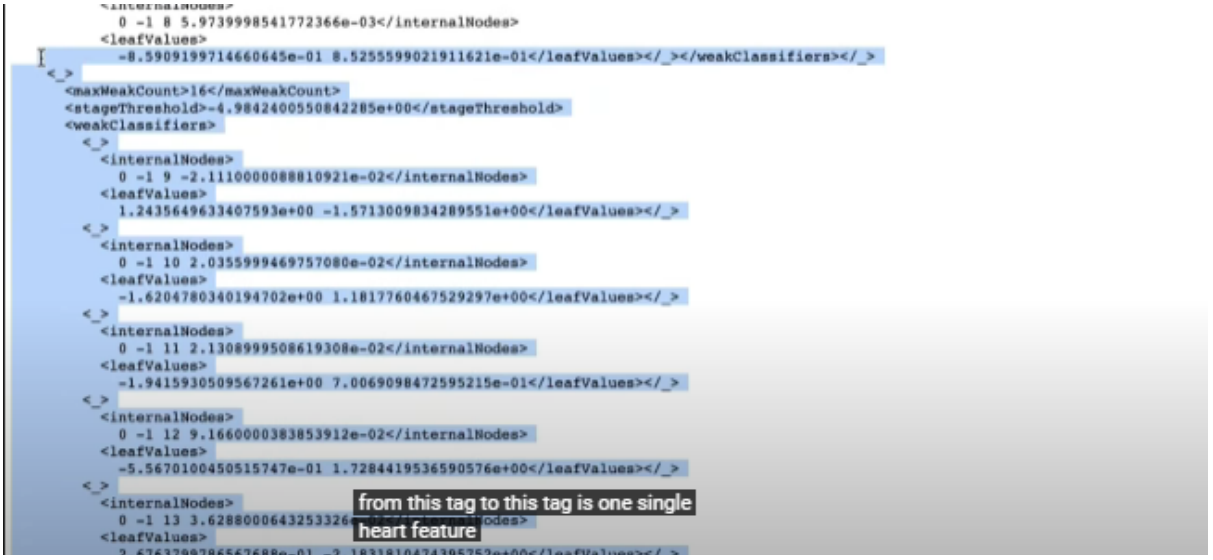
Find all the winning haar feature and add them up

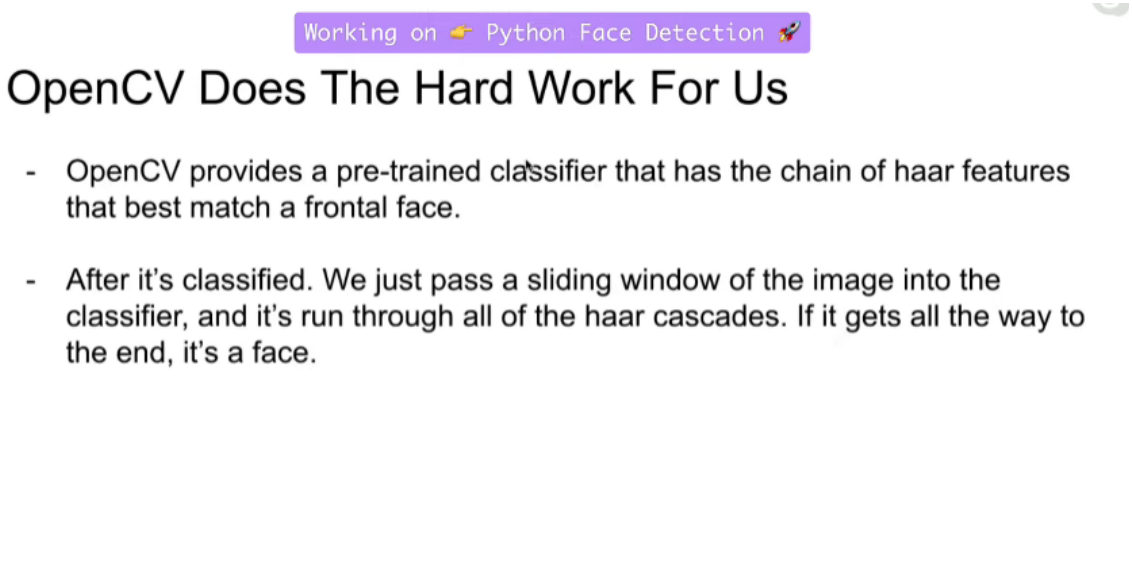






From this highlighted area we can say that this can be one haar features





https://www.youtube.com/watch?v=hPCTwxF0qf4