Experiment No. 6
To study Detecting and Recognizing Faces
Name of Student :- 09_Amruta Poojary
Date of Performance: 23/8/2023
Date of Submission: 6/9/2023

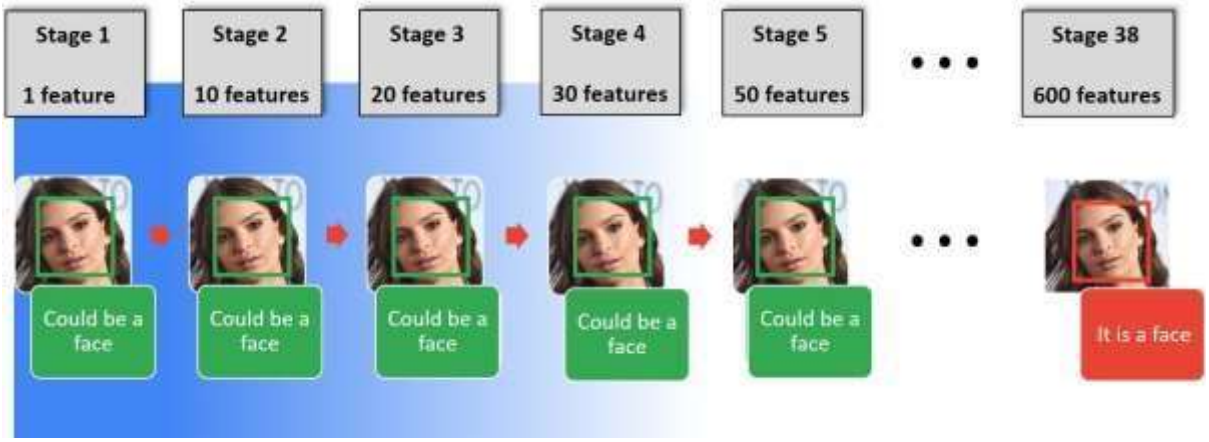
Aim: To study Detecting and Recognizing Faces

Objective: To Conceptualizing Haar Cascades Getting Haar cascade data Using Open CV to Perform face detections performing face detection on still images

Theory:

Introduction

Discover object detection with the Haar Cascade algorithm using OpenCV. Learn how to employ this classic method for detecting objects in images and videos. Explore the underlying principles, step-by-step implementation, and real-world applications. From facial recognition to vehicle detection, grasp the essence of Haar Cascade and OpenCV's role in revolutionizing computer vision. Whether you're a novice or an expert, this article will equip you with the skills to harness the potential of object detection in your projects.



Why Use Haar Cascade Algorithm for Object Detection?

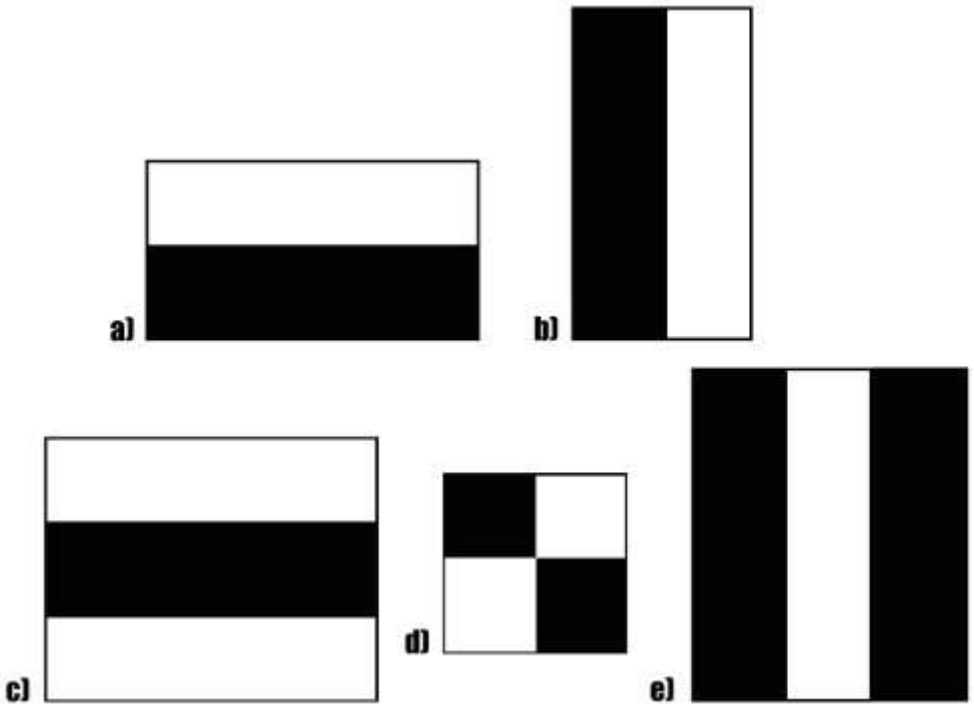
Identifying a custom object in an image is known as object detection. This task can be done using several techniques, but we will use the haar cascade, the simplest method to perform object detection in this article.

What is Haar Cascade Algorithm?

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location.

This algorithm is not so complex and can run in real-time. We can train a haarcascade detector to detect various objects like cars, bikes, buildings, fruits, etc.

Haar cascade uses the cascading window, and it tries to compute features in every window and classify whether it could be an object.



Haar cascade works as a classifier. It classifies positive data points → that are part of our detected object and negative data points → that don't contain our object.

- Haar cascades are fast and can work well in real-time.
- Haar cascade is not as accurate as modern object detection techniques are.
- Haar cascade has a downside. It predicts many false positives.
- Simple to implement, less computing power required.

Code:

```
import dlib
import cv2
#from google.colab.patches import cv2_imshow

# Load the pre-trained face detection model from dlib
detector = dlib.get_frontal_face_detector()

# Load an image or capture it from a camera
# Replace 'your_image.jpg' with the path to your image or use 0 for the
default camera (webcam)
input_image = cv2.imread('image.png')
cv2_imshow(input_image)
# Alternatively, you can capture video from a webcam
# cap = cv2.VideoCapture(0)

# Convert the image to grayscale for face detection
gray = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = detector(gray)

# Draw rectangles around detected faces
for face in faces:
    x, y, w, h = face.left(), face.top(), face.width(), face.height()
    cv2.rectangle(input_image, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Display the image with faces highlighted
cv2_imshow(input_image)
```

Input Image:

Output:

Conclusion:

Face detection is an important task in many computer vision or machine vision applications, Face detection can be carried by using the Haar Cascade algorithm that can detect objects in images, irrespective of their scale in image and location. OpenCV allows to use to Haar cascade algorithm since the OpenCV library maintains a repository of pre-trained Haar cascades thus enabling us to perform face detection tasks easily