Step 4: Writing the code to detect faces in the video stream

What Is Face Detection?

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images.

The goal of face detection is to determine if there are any faces in the image or video. If multiple faces are present, each face is enclosed by a bounding box and thus we know the location of the faces.

Human faces are difficult to model as there are many variables that can change for example facial expression, orientation, lighting conditions . The result of the detection gives the face location parameters and it could be required in various forms, for instance, a rectangle covering the central part of the face, eye centres or landmarks including eyes, nose and lips, mouth corners, eyebrows, nostrils, etc.

How face detection works

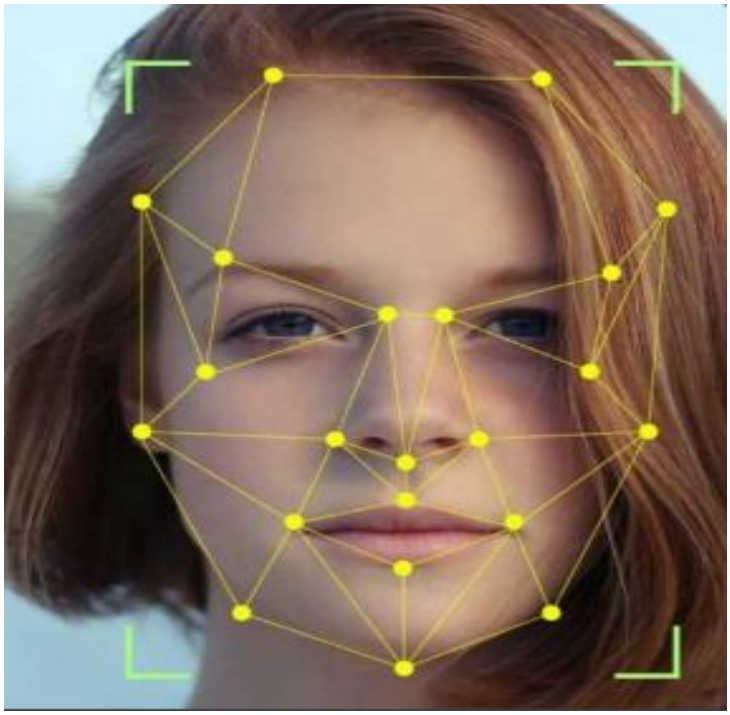
Face detection applications use algorithms and machine learning to find human faces within images and video , which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes -- one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face

To make algorithms as accurate as possible, they must be trained with huge data sets that contain hundreds of thousands of images. Some of these images contain faces, while others do not. The training procedures help the algorithm’s ability to decide whether an image contains faces, and where those facial regions are located.

To detect the faces we used tensor flow with cv2 to do the images/video treatment.we used the pretrained model haarscascade

```
cv2.namedWindow('window_frame')

video_capture = cv2.VideoCapture(0)

while True:

    bgr_image = video_capture.read()[1]

    gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)

    rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)

    faces = detect_faces(face_detection, gray_image) # decte face
```

Writing The Code To Find The Region Of Interest



ROI or region of interest is the face region that contains the expression, we will feed the face detection algorithm the ROI in order to predict the face expression.

```
for face_coordinates in faces:



        x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
```

```
    gray_face = gray_image[y1:y2, x1:x2]
    try:
        gray_face = cv2.resize(gray_face, (emotion_target_size))
    except:
        continue


    gray_face = preprocess_input(gray_face, True)
    gray_face = np.expand_dims(gray_face, 0)
    gray_face = np.expand_dims(gray_face, -1)
```