**Practical Assignment**

**Objective: - Human Face Detection System**

The face detection took a major leap with deep learning techniques. We can build models with high accuracy in detecting the bounding boxes of the human face. This project will get you started with object detection and you will learn how to detect any object in an image.

Face recognition technology is a subset of Object Detection that focuses on observing the instance of semantic objects. It is designed to track and visualize human faces within digital images.

**Dataset Link: -**

Lot of them are available use any one of your choice.

**Task: -** Create a Web Application using FASTAPI. Use the end user should be able to upload an image and get results with the prediction face and bounding box coordinates. Use any 3 different algorithms. The end user should have the choice to select the algorithm options via a dropdown in the webpage.

**Deployment: -** Any Free Platform (Try to look out for free options.)

**Assignment Submission: -** Only submit the hosted app link. OR GitHub Link

pip install opencv-python

**import** cv2

image = cv2.imread("image.jpg")

face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

faces = face\_cascade.detectMultiScale(image, scaleFactor=1.1, minNeighbors=5)

**Once the faces are detected, we can draw rectangles around them using the rectangle function:**

**for** (x, y, w, h) **in** faces:

cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0),2)

Finally, we can display the image with the detected faces:

// displaying the image //

cv2.imshow("Faces", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

Implementation using OpenCV

**import** cv2

// load the cascade classifier **for** face detection //

face\_cascade = cv2.CascadeClassifier('path/to/haarcascade\_frontalface\_default.xml')

// load the image //

img = cv2.imread('path/to/image.jpg')

// convert the image to grayscale //

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

// detect faces **in** the image //

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

// draw a rectangle around the detected faces //

**for** (x, y, w, h) **in** faces:

    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

// display the image //

cv2.imshow('img', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Face Recognition using DLIB

**Implementation of the Dlib Method:-**

**import** dlib

// Create a face detector //

detector = dlib.get\_frontal\_face\_detector()

// Load an image //

img = dlib.load\_rgb\_image("image.jpg")

// Detect faces **in** the image //

faces = detector(img)

// Print the number of faces detected //

**print**("Number of faces detected: ", len(faces))

// Loop through the faces **and** **print** their coordinates //

**for** face **in** faces:

**print**("Left: ", face.left())

**print**("Top: ", face.top())

**print**("Right: ", face.right())

**print**("Bottom: ", face.bottom())

## Face Recognition using Face Recognition Library

**Implementation of the code :-**

**import** face\_recognition

// load the image //

image = face\_recognition.load\_image\_file("image.jpg")

// detect faces **in** the image //

face\_locations = face\_recognition.face\_locations(image)

// loop through the face locations **and** draw rectangles around the faces //

**for** face\_location **in** face\_locations:

    top, right, bottom, left = face\_location

    cv2.rectangle(image, (left, top), (right, bottom), (0, 0, 255), 2)

// display the image //

cv2.imshow("Faces", image)

cv2.waitKey(0)

### MTCCN Method for Face Recognition

**from** mtcnn **import** MTCNN

**import** cv2

// load image **from** file //

filename = "image.jpg"

pixels = cv2.imread(filename)

// we need to create a detector by default weights //

detector = MTCNN()

// to find faces **in** an img //

faces = detector.detect\_faces(pixels)

// display faces on the original image //

**for** face **in** faces:

    x, y, width, height = face['box']

    cv2.rectangle(pixels, (x, y), (x+width, y+height), (0, 0, 255), 2)

    cv2.imshow('Face', pixels)

    cv2.waitKey(0)

### Face Recognition in TensorFlow Method

Example of Tenserflow method

**import** tensorflow as tf

// Load the Caffe model //

model = tf.keras.models.load\_model('path/to/caffe\_model.h5')

// Define the input image //

img = tf.keras.Input(shape=(224, 224, 3))

//Preprocess the image **for** the model //

x = tf.keras.layers.Lambda(**lambda** x: x / 255.0)(img)

// Run the image through the model //

predictions = model(x)

// Find the location of the face **in** the image //

face\_locations = tf.where(predictions > 0.5)

// Print the location of the face **in** the image //

**print**(face\_locations)