Experiment-9 & 10.py

```
# 18BCS6201-CV Practical-9 & 10 (Gauri Prabhakar) (AI-ML-2)(B)
# Aim: To study the YOLO (You Only Look Once) state of the art object detection algorithm. Write various phases of implementation.
#      To study Pedestrian, car, bus, truck, etc. detection from a video (category n- problem) using YOLO.

# Importing necessary modules.
import cv2
import numpy as np

# Creating a variable to store the video using the '.VideoCapture()' function.
cap = cv2.VideoCapture(r"C:\Users\gauri\Desktop\OpenCV Media\Tenet.mp4")
#cap = cv2.VideoCapture(r"C:\Users\gauri\Desktop\OpenCV Media\Pedestrian.mp4")

# Defining parameters.
whT=320
classFile=[]

# Writing 'coco.names' to the list 'classFile'.
with open("coco.names","rt") as f: classFile = f.read().strip('\n').split('\n')

# Returning the length of the list.
print(len(classFile))

# Loading the configuration files and weights.
net=cv2.dnn.readNetFromDarknet("yolov3.cfg","yolov3.weights")

# Asking network to use specific computation back-end where it is supported.
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
net.setPreferableBackend(cv2.dnn.DNN_TARGET_CPU)

# Defining a function to implement YOLO.
def findObjects(outputs,img):

    # Loading the shape of the image into the variables.
    hT,wT,cT=img.shape

    # Printing the shape of the image.
    print(img.shape)

    # Declaring the confidence.
    confTh=0.5
    nmsTh=0.3

    # Declaring the list to store classIDs.
    classIDs=[]
    # Declaring the list to store confidence.
    confidences=[]
    bbox=[]

    # Looping to implement YOLO.
    for output in outputs:
        for det in output:
            #print(det)
            score=det[5:]
            classID=np.argmax(score)
            confidence=score[classID]
            if confidence>confTh:
                w,h=int(det[2]*wT),int(det[3]*hT)
                x,y=int((det[0]*wT)-w/2), int((det[1]*hT)-h/2)
                bbox.append([x,y,w,h])
                classIDs.append(classID)
                confidences.append(float(confidence))
                #print(bbox)
                #print(classIDs)
                #print(confidences)
    indices = cv2.dnn.NMSBoxes(bbox,confidences,confTh,nmsTh)
    #print(indices)

    # Looping through indices.
    for i in indices:
        i=i[0]
        box=bbox[i]
        #print(box)
        x,y,w,h=box[0],box[1],box[2],box[3]
        cv2.rectangle(img,(x,y),(x+w,y+h),(60,20,220),2)
        #print(cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,255),2))
        cv2.putText(img, f'{classFile[classIDs[i]].upper()} {int(confidences[i]*100)}%',
                    (x,y-10),cv2.FONT_HERSHEY_SIMPLEX,0.6,(0,252,154),2)

# Setting condition to receive input and apply YOLO.
while True:
    # Capturing the video frame by frame using the '.read()' method.
    success, frm = cap.read()
    blobs= cv2.dnn.blobFromImage(frm,1/255,(whT,whT),[0, 0, 0],1,crop=False)
    net.setInput(blobs)
    layer_names=net.getLayerNames()
    outNames=[layer_names[i[0]-1] for i in net.getUnconnectedOutLayers()]
    outputs=net.forward(outNames)
    print(type(outputs[0]))
    print(outputs[0].shape)

    findObjects(outputs,frm)

    # Rendering the video with effective YOLO Object Detection to the console by using the function '.imshow()'.
    cv2.imshow('Implementation of YOLO Object Detection',frm)

    # Setting up '.waitkey()' to wait for a specific time until any key is pressed and break the loop.
    # '.waitkey(1)' displays a frame for 1ms after which it moves to the next frame in the video.
    # Setting 'x' as the quitting button.
    if cv2.waitKey(5) & 0xFF == ord('x'):
      break

# Releasing the variable/object 'cap'.
cap.release()
```