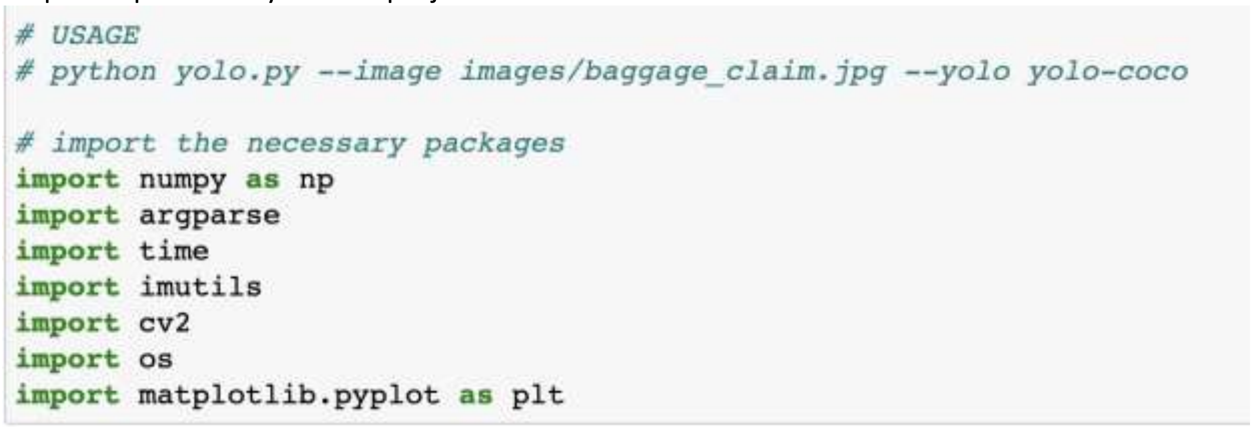
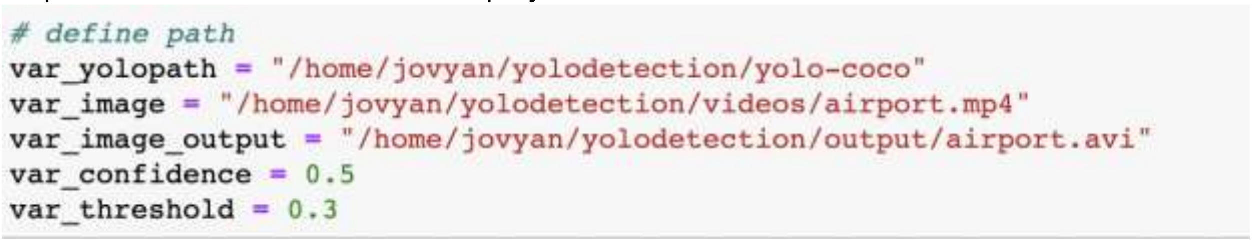
Detect objects in video clips with Yolov3

Step 1: Import library for this project:



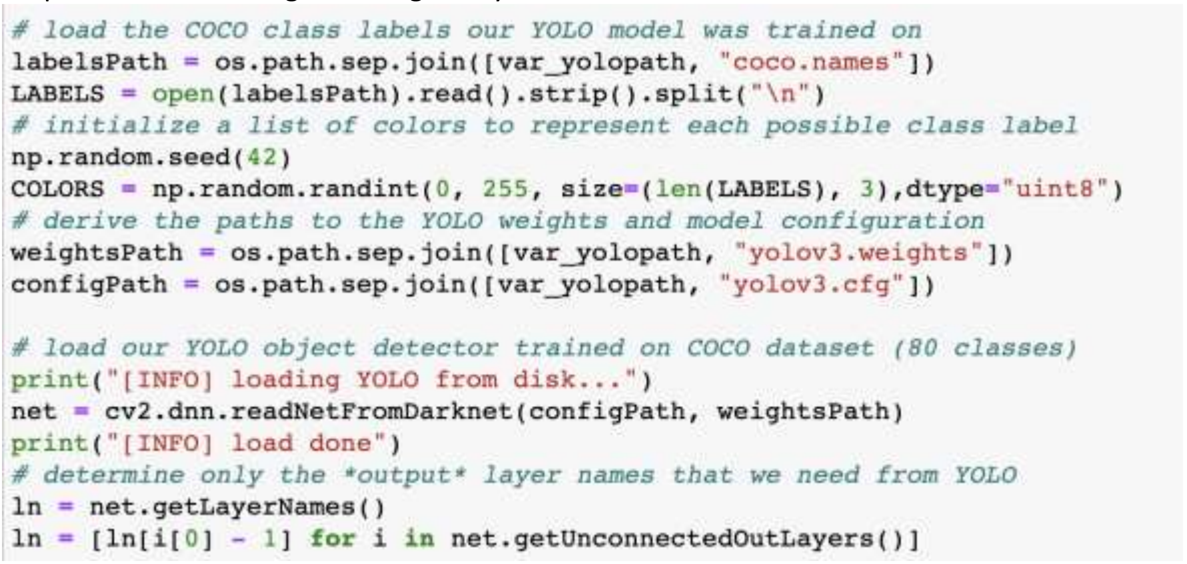
```
# USAGE
# python yolo.py --image images/baggage_claim.jpg --yolo yolo-coco

# import the necessary packages
import numpy as np
import argparse
import time
import imutils
import cv2
import os
import matplotlib.pyplot as plt
```

Step 2: Define the variable for whole project



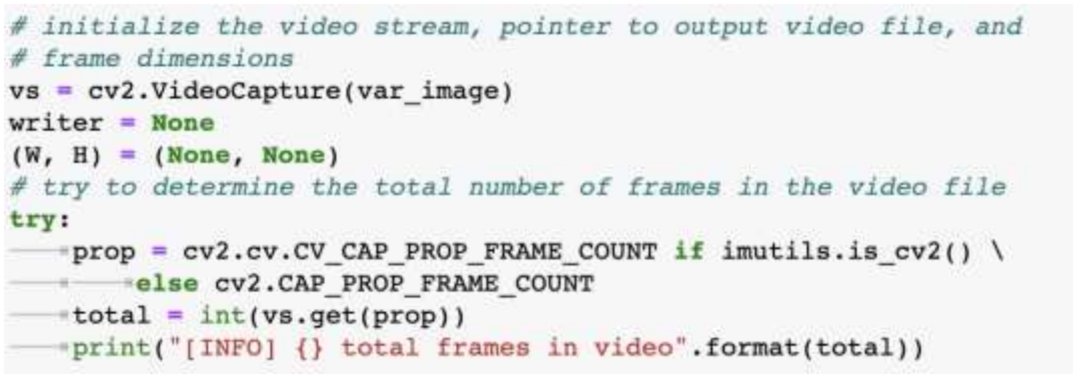
```
# define path
var_yolopath = "/home/jovyan/yolodetection/yolo-coco"
var_image = "/home/jovyan/yolodetection/videos/airport.mp4"
var_image_output = "/home/jovyan/yolodetection/output/airport.avi"
var_confidence = 0.5
var_threshold = 0.3
```

Step 3: Load the config and weight of yolov3:



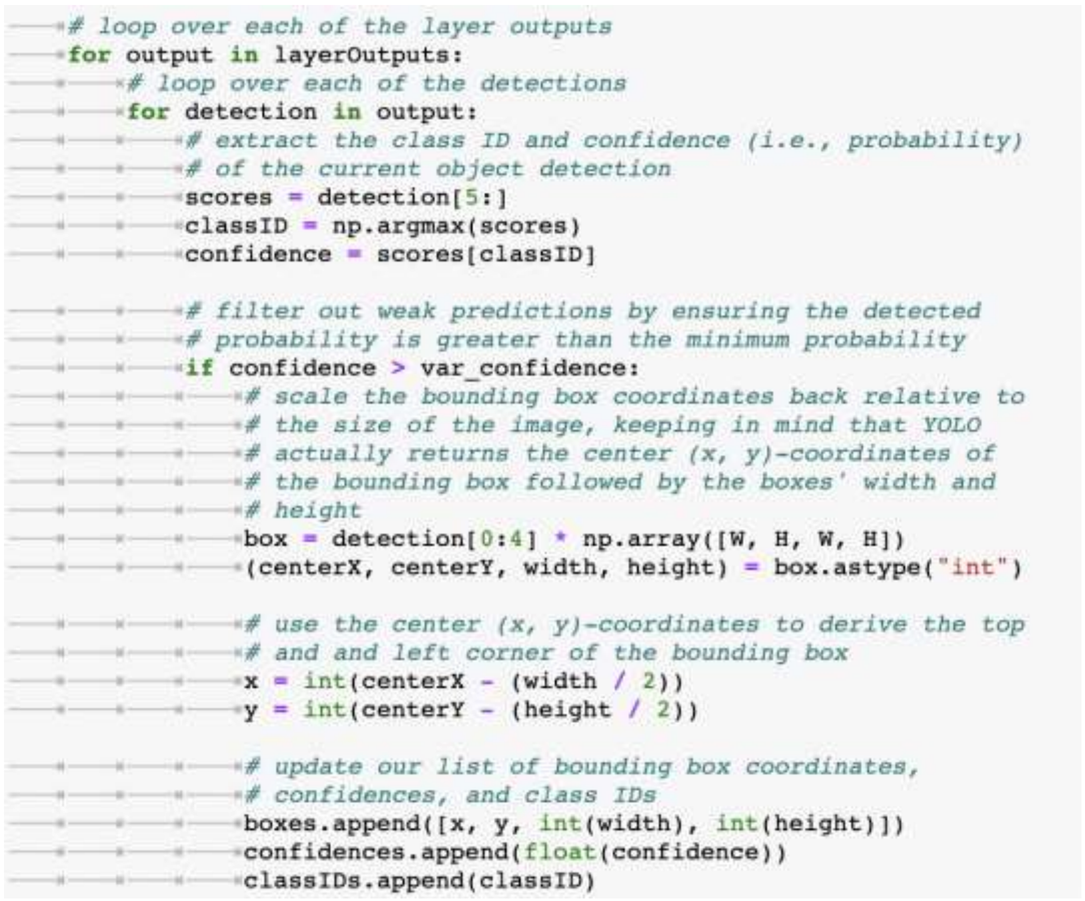
```
# load the COCO class labels our YOLO model was trained on
labelsPath = os.path.sep.join([var_yolopath, "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")
# initialize a list of colors to represent each possible class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),dtype="uint8")
# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join([var_yolopath, "yolov3.weights"])
configPath = os.path.sep.join([var_yolopath, "yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
print("[INFO] load done")
# determine only the *output* layer names that we need from YOLO
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

Step 4: Load the videos that we need to recognize the object



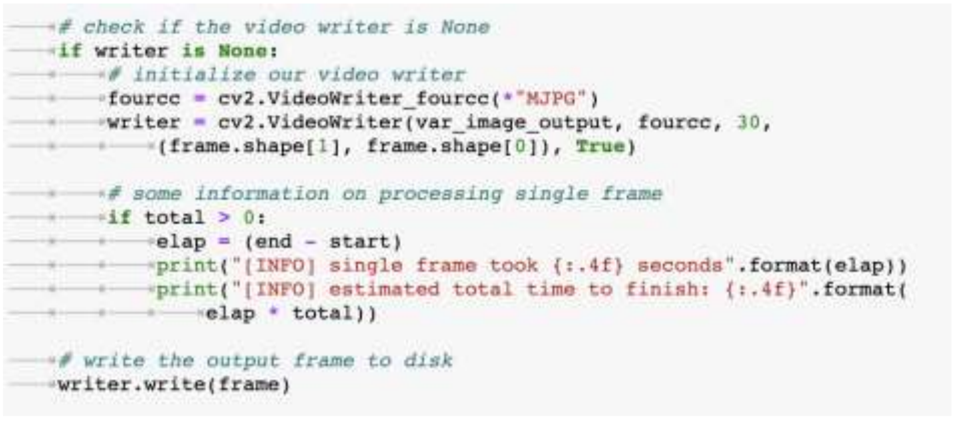
```
# initialize the video stream, pointer to output video file, and
# frame dimensions
vs = cv2.VideoCapture(var_image)
writer = None
(W, H) = (None, None)
# try to determine the total number of frames in the video file
try:
    prop = cv2.cv.CV_CAP_PROP_FRAME_COUNT if imutils.is_cv2() \
        else cv2.CAP_PROP_FRAME_COUNT
    total = int(vs.get(prop))
    print("[INFO] {} total frames in video".format(total))
```

Step 5: We loop over each the frames in the videos and try detect object in this frame by using the Yolov3



```
    # loop over each of the layer outputs
    for output in layerOutputs:
        # loop over each of the detections
        for detection in output:
            # extract the class ID and confidence (i.e., probability)
            # of the current object detection
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]

            # filter out weak predictions by ensuring the detected
            # probability is greater than the minimum probability
            if confidence > var_confidence:
                # scale the bounding box coordinates back relative to
                # the size of the image, keeping in mind that YOLO
                # actually returns the center (x, y)-coordinates of
                # the bounding box followed by the boxes' width and
                # height
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype("int")

                # use the center (x, y)-coordinates to derive the top
                # and and left corner of the bounding box
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))

                # update our list of bounding box coordinates,
                # confidences, and class IDs
                boxes.append([x, y, int(width), int(height)])
                confidences.append(float(confidence))
                classIDs.append(classID)
```

Step 6: After detect, and draw the boxes cover the object, we write the frame to the output video.



```python
    # check if the video writer is None
    if writer is None:
        # initialize our video writer
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(var_image_output, fourcc, 30,
            (frame.shape[1], frame.shape[0]), True)

        # some information on processing single frame
        if total > 0:
            elap = (end - start)
            print("[INFO] single frame took {:.4f} seconds".format(elap))
            print("[INFO] estimated total time to finish: {:.4f}".format(
                elap * total))

    # write the output frame to disk
    writer.write(frame)
```