```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 19 12:44:52 2019

@author: ATHARV S DESAI
"""

import numpy as np
import cv2
import os
import time
import argparse


ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True,
        help="path to input image")
args = vars(ap.parse_args())

YOLO_PATH = "cfg"
CONF = 0.5 # minimum probability to filter weak detections
THRESH = 0.3 # threah when applying non-maxima suppresion
```

```
# loading the coco labels
labels_path = YOLO_PATH + "/coco.names"
LABELS = open(labels_path).read().strip().split('\n')

# init list of colors to represent each class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
        dtype='uint8')

# reading weights and cfg files
weights_path = YOLO_PATH + "/yolov3.weights"
cfg_path = YOLO_PATH + "/yolov3.cfg"

print("Loading the yolo cfg files...")
net = cv2.dnn.readNetFromDarknet(cfg_path, weights_path)

# load the given image
image = cv2.imread(args['image'])
(h, w) = image.shape[:2]

# determine only the output layer names
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

```
# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416),
        swapRB=True, crop=False)
net.setInput(blob)
start = time.time()

# perform forward pass of the YOLO detector and get the bounding boxes
layer_outputs = net.forward(ln)
end = time.time()

print("Forward Pass took {:.6f} seconds".format(end-start))

# init boxes, confidences, class_ids
boxes, confidences, class_ids = [], [], []

for output in layer_outputs:
        for detection in output:
                # extract the class_ids and the confidence
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence =  scores[class_id]
```

```
                # filter out weak predictions
                if confidence > CONF:
                        # scaling the bounding box coordinates to the
                        # size of the image
                        box = detection[0:4] * np.array([w, h, w, h])
                        (center_x, center_y, width, height) = box.astype('int')

                        x = int(center_x - (width / 2))
                        y = int(center_y - (height / 2))

                        boxes.append([x, y, int(width), int(height)])
                        confidences.append(float(confidence))
                        class_ids.append(class_id)

# apply npn-maxima suppresion
idxs = cv2.dnn.NMSBoxes(boxes, confidences, CONF, THRESH)

if len(idxs) > 0:
        for i in idxs.flatten():
                # extracting the box coordinates
                x, y = boxes[i][0], boxes[i][1]
                w, h = boxes[i][2], boxes[i][3]
```

```
                # drawing the box
                color = [int(c) for c in COLORS[class_ids[i]]]
                cv2.rectangle(image, (x, y), (x+w, y+h), color, 2)
                text = "{} : {:.4f}".format(LABELS[class_ids[i]], confidences[i])
                cv2.putText(image, text, (x, y-5), cv2.FONT_HERSHEY_SIMPLEX,
0.5, color, 2)


# display the image
cv2.imshow("Image", image)
cv2.waitKey(0)
```