import cv2

import numpy as np

import argparse

from matplotlib import pyplot as plt

ap = argparse.ArgumentParser()

ap.add\_argument('-i', '--image', required=True, help = 'Path to input image')

ap.add\_argument('-c', '--config', required=True, help = 'Path to yolo config files')

ap.add\_argument('-w', '--weights', required=True, help = "Path to weigth files")

ap.add\_argument('-c1', '--classes', required=True, help = "path to text file containing classes")

#args = ap.parse\_args()

def get\_output\_layers(net):

    layer\_names = net.getLayerNames()

    output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

    return output\_layers

def draw\_prediction(img, class\_id, confidence, x, y, x\_plus\_w, y\_plus\_h):

    label = str(classes[class\_id])

    color = COLORS[class\_id]

    cv2.rectangle(img, (x,y), (x\_plus\_w,y\_plus\_h), color, 2)

    cv2.putText(img, label, (x-10,y-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, color, 2)

image = cv2.imread(r"C:\Users\User\Downloads\Copy of YOLO\YOLO\img\city.jpg")

plt.imshow(image)

plt.show()

Width = image.shape[1]

Height = image.shape[0]

scale = 0.00392

classes = None

with open(r"C:\Users\User\Downloads\Copy of YOLO\YOLO\yolov3.txt", 'r') as f:

    classes = [line.strip() for line in f.readlines()]

COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

''

net = cv2.dnn.readNet(r"C:\Users\User\Downloads\Copy of YOLO\YOLO\yolov3.weights",

                      r"C:\Users\User\Downloads\Copy of YOLO\YOLO\yolov3.cfg")

blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), swapRB=True, crop=False)

net.setInput(blob)

outs = net.forward(get\_output\_layers(net))

class\_ids = []

confidences = []

boxes = []

conf\_threshold = 0.5

nms\_threshold = 0.4

print(image.shape)

for out in outs:

    for detection in out:

        scores = detection[5:]

        class\_id = np.argmax(scores)

        confidence = scores[class\_id]

        if confidence > 0.5:

            center\_x = int(detection[0] \* Width)

            center\_y = int(detection[1] \* Height)

            w = int(detection[2] \* Width)

            h = int(detection[3] \* Height)

            x = center\_x - w / 2

            y = center\_y - h / 2

            class\_ids.append(class\_id)

            confidences.append(float(confidence))

            boxes.append([x, y, int(w), int(h)])

indices = cv2.dnn.NMSBoxes(boxes, confidences, conf\_threshold, nms\_threshold)

for i in indices:

    box = boxes[i]

    x = box[0]

    y = box[1]

    w = box[2]

    h = box[3]

    draw\_prediction(image, class\_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))

cv2.imshow("object detection Innomatics", image)

cv2.waitKey()

cv2.imwrite("object-detection.jpg", image)

cv2.destroyAllWindows()

plt.imshow(image)

plt.show()