! pip install ultralytics!

! pip install pycocotools

import pycocotools

from ultralytics import YOLO

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import cv2

import os

import yaml

import torch

import shutil

import wandb

%matplotlib inline

train\_imgs\_dir="/kaggle/input/car-object-detection/data/training\_images"train\_labels="/kaggle/input/car-object-detection/data/train\_solution\_bounding\_boxes (1).csv"

test\_imgs\_dir="/kaggle/input/car-object-detection/data/testing\_images"

imgs\_list=list(sorted(os.listdir(train\_imgs\_dir)))idxs=list(range(len(imgs\_list)))np.random.shuffle(idxs)

train\_idx=idxs[:int(0.8\*len(idxs))]val\_idx=idxs[int(0.8\*len(idxs)):]

!mkdir "/kaggle/working/data"

*# images directory* !mkdir "/kaggle/working/data/images"

*# train and test subdirectories with image directory*!mkdir "/kaggle/working/data/images/train"!mkdir "/kaggle/working/data/images/val"

*# labels directory*!mkdir "/kaggle/working/data/labels"

*# train and test subdirectories with labels directory*!mkdir "/kaggle/working/data/labels/train"!mkdir "/kaggle/working/data/labels/val"

root\_dir="/kaggle/working/data"labels\_dir="/kaggle/working/data/labels"images\_dir="/kaggle/working/data/images"

df=pd.read\_csv(train\_labels)

width=676height=380

df["class"]=0df.rename(columns={'image':'img\_name'}, inplace=True)

df["x\_centre"]=(df["xmin"]+df["xmax"])/2df["y\_centre"]=(df["ymin"]+df["ymax"])/2df["width"]=(df["xmax"]-df["xmin"])df["height"]=(df["ymax"]-df["ymin"])

*#normalizing bounding box coordinates*df["x\_centre"]=df["x\_centre"]/widthdf["y\_centre"]=df["y\_centre"]/heightdf["width"]=df["width"]/widthdf["height"]=df["height"]/height

df\_yolo=df[["img\_name","class","x\_centre","y\_centre","width","height"]]df\_yolo.head()

for idx,img\_name **in** enumerate(imgs\_list):

subset="train"

if idx **in** val\_idx:

subset="val"

if np.isin(img\_name,df\_yolo["img\_name"]):

columns=["class","x\_centre","y\_centre","width","height"]

img\_bbox=df\_yolo[df\_yolo["img\_name"]==img\_name][columns].values

label\_file\_path=os.path.join(labels\_dir,subset,img\_name[:-4]+".txt")

with open(label\_file\_path,"w+") as f:

for row **in** img\_bbox:

text=" ".join(row.astype(str))

f.write(text)

f.write("**\n**")

old\_image\_path=os.path.join(train\_imgs\_dir,img\_name)

new\_image\_path=os.path.join(images\_dir,subset,img\_name)

shutil.copy(old\_image\_path,new\_image\_path)

yolo\_format=dict(path="/kaggle/working/data",

train="/kaggle/working/data/images/train",

val="/kaggle/working/data/images/val",

nc=1,

names={0:"car"})

with open('/kaggle/working/yolo.yaml', 'w') as outfile:

yaml.dump(yolo\_format, outfile, default\_flow\_style=False)

model=YOLO('yolov8m.pt')model.train(data="/kaggle/working/yolo.yaml",epochs=5,patience=5,batch=8,

lr0=0.0005,imgsz=640)

path\_best\_weights="/kaggle/working/runs/detect/train/weights/best.pt"model = YOLO(path\_best\_weights)

metrics = model.val()

print(f"Mean Average Precision @.5:.95 : **{**metrics.box.map**}**") print(f"Mean Average Precision @ .50 : **{**metrics.box.map50**}**") print(f"Mean Average Precision @ .70 : **{**metrics.box.map75**}**")

with torch.no\_grad():

results=model.predict(source=test\_imgs\_dir,conf=0.50,iou=0.75)

!mkdir "/kaggle/working/predictions"prediction\_dir="/kaggle/working/predictions"

test\_img\_list=[]for result **in** results:

if len(result.boxes.xyxy):

name=result.path.split("/")[-1].split(".")[0]

boxes=result.boxes.xyxy.cpu().numpy()

scores=result.boxes.conf.cpu().numpy()

test\_img\_list.append(name)

label\_file\_path=os.path.join(prediction\_dir,name+".txt")

with open(label\_file\_path,"w+") as f:

for score,box **in** zip(scores,boxes):

text=f"**{**score**:**0.4f**}** "+" ".join(box.astype(str))

f.write(text)

f.write("**\n**")

def show\_bbox(img,boxes,scores,axis,color=(0,255,0)):

boxes=boxes.astype(int)

scores=scores

img=img.copy()

for i,box **in** enumerate(boxes):

score=f"**{**scores[i]**:**.4f**}**"

cv2.rectangle(img,(box[0],box[1]),(box[2],box[3]),color,2)

y=box[1]-10 if box[1]-10>10 else box[1]+10

cv2.putText(img,score,(box[0],y),cv2.FONT\_HERSHEY\_SIMPLEX,0.5,color,2)

axis.imshow(img)

axis.axis("off")

fig,axes=plt.subplots(5,3,figsize=(12,12))plt.subplots\_adjust(wspace=0.1,hspace=0.1)ax=axes.flatten()

imgs\_name=np.random.choice(test\_img\_list,15)

for i,img\_name **in** enumerate(imgs\_name):

img\_file\_path=os.path.join(test\_imgs\_dir,img\_name+".jpg")

img=cv2.imread(img\_file\_path)

img=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

label\_file\_path=os.path.join(prediction\_dir,img\_name+".txt")

label=pd.read\_csv(label\_file\_path,sep=" ",header=None).values

scores=label[:,0]

boxes=label[:,1:]

show\_bbox(img,boxes,scores,axis=ax[i])

plt.savefig("car.png")

OutPut:-

