D:\myprojects\ultralytics\yolov8_detect.py

```python
#************************************************************
# Object Detecction using YOLO8
#************************************************************
# https://docs.ultralytics.com/modes/predict/#working-with-results
# https://encord.com/blog/yolo-object-detection-guide/          # YOLO8 Explanation

# conda create -n yolov8_detect python=3.10 tk
# conda activate yolov8_detect
# conda install -c conda-forge ultralytics
# conda install pytorch torchvision torchaudio cpuonly -c pytorch
# conda install  pyyaml matplotlib tqdm
# pip install opencv-python

'''
[1] For Korean character output, copy "NanumGothic.ttf" file to c:/windows/fonts folder
[2] Include the following at line 14 of utlralytics/ultralytics/metrics.py file
        import matplotlib as mpl
        from matplotlib import font_manager as fm

        plt.rcParams['font.family'] = 'NanumGothic'
        mpl.rcParams['axes.unicode_minus'] = False
'''

from ultralytics import YOLO
import os, sys, torch
import cv2, tkinter, yaml
import numpy as np
from PIL import ImageFont, ImageDraw, Image
from datetime import datetime

YOLO_MODEL = "yolov8m.yaml"
YOLO_WEIGHT = "yolov8m.pt"

INPUT_IMAGE_SIZE = 640
BATCH_SIZE = 16

#============================================================
# Get Screen Info.
#============================================================
def get_screen_resolution():
    app = tkinter.Tk()
    screenWidth = int(round(app.winfo_screenwidth() * 0.9))
    screenHeight = int(round(app.winfo_screenheight() * 0.9))
    app.destroy()

    return screenWidth, screenHeight

#============================================================
# SHow OpenCV-style Image Window
#============================================================
def show_image(image,windowTitle="",x=0,y=0,showMode="manual",destroyMode='destroy') :
    if windowTitle == "" : windowTitle = "Result Image"

    sizeInfo = image.shape
    imgHeight=sizeInfo[0];  imgWidth = sizeInfo[1]
    heightRatio = float(screenHeight / imgHeight)
    showWidth = int(round(heightRatio * imgWidth)); showHeight = screenHeight
```

```
    resizedImage = cv2.resize(image, (showWidth, showHeight), interpolation=
cv2.INTER_CUBIC)

    cv2.namedWindow(windowTitle,cv2.WINDOW_NORMAL)
    cv2.moveWindow(windowTitle,x,y)
    cv2.resizeWindow(windowTitle, showWidth, showHeight)
    cv2.imshow(windowTitle,resizedImage)

    miliSecond = 0 if showMode == 'manual' else 2

    inKey = cv2.waitKey(miliSecond)
    if destroyMode == 'destroy' :  cv2.destroyWindow(windowTitle)

    return inKey

#=========================================================
# Make Prediction File List
#=========================================================
def get_prediction_file_list(dataFolder) :
    argFullPath =  os.path.join(os.getcwd(), dataFolder)
    print("data_path: ",argFullPath)
    if os.path.isfile(argFullPath) :
        imgDataList.append(argFullPath)
    elif os.path.isdir(argFullPath) :
        fileList = os.listdir(argFullPath)
        imgDataList = [os.path.join(argFullPath,file) for file in fileList
                       if (file.endswith(".jpg") or file.endswith(".png") or
                           file.endswith(".JPG") or file.endswith(".PNG"))]

    # make save image extention
    originalExt = imgDataList[0][-4:]
    originalExtUpper = originalExt.upper()
    if originalExtUpper == '.JPG' :
        newExt = '.png'
    elif originalExtUpper == '.PNG' :
        newExt = '.jpg'
    else :
        print("\n\n Image Format Mismatch !!! [.jpg or .png needed]")
        sys.exit[1]

    return imgDataList, originalExt, newExt

#=========================================================
# Train the YOLO model with the Custom Data
#=========================================================
def train_custom_data(dataYaml="data-car.yaml", cfgYaml="conf-car.yaml",  epochNo = 100 )
:
    # remove previous training result folder
    '''
    resultPath =  os.path.join(os.getcwd(), resultFolder) + "/train"
    if os.path.exists(resultPath) :
        os.remove(resultPath)           '''

    # load a pretrained model
    model = YOLO(YOLO_MODEL)    # build a new model from scratch
    model = YOLO(YOLO_WEIGHT)     # load a pretrained model (recommended for training)
    #model = YOLO("best.pt")     # load a pretrained model

    # train the model with the custom data
    startTime = datetime.now()
```

```
    model.train(data=dataYaml, cfg=cfgYaml, imgsz=INPUT_IMAGE_SIZE, batch=BATCH_SIZE, epochs=epochNo, verbose=False)
    endTime = datetime.now()
    print(f"Training Time: ",{(endTime - startTime)})

    # save the model
    metrics = model.val()  # evaluate model performance on the validation set

    print("\n","**** Training Ended !!!... [best.pt] saved under the runs/detect/train** folder. ***")

#============================================================
# Get DataSet File(s), Predict and show the Result Images
#============================================================
def predict_data(weightToUse = "trainResult/car/weights/best.pt", dataFolder = "datasets/car/test/",
                 showMode = 'manual', saveMode = 'save' ) :
    imgDataList = []

    imgDataList, originalExt, newExt = get_prediction_file_list(dataFolder)

    # load a yolo model and weights
    model = YOLO(YOLO_MODEL)    # build a new model from scratch
    model = YOLO(weightToUse)     # load a pretrained model

    #-------------------------------------------------------------------------------------------
    for imgSeq, imgFileName in enumerate(imgDataList) :
        # pridict for each image
        colorImage = cv2.imread(imgFileName,cv2.IMREAD_COLOR)
        results = model(imgFileName, max_det = 30,  verbose=False)#, conf = 0.25) # predict maximum 30 objects from an image

        #----------------------------
        # Drawing recognition results for all found objects (mbr, conf, className)
        #----------------------------
        for result in results:
            boxes = result.boxes  # Boxes object(tensor) for bbox outputs

            mbrLeft =[];   mbrRight=[];   mbrTop=[]; mbrBottom=[]   # MBR positional Values
            foundConfList = [];   foundClassNdxList = []; foundClassNamesList = []

            no=0
            #-------------------------------------------------------------------------
            for l,t,r,b in boxes.xyxy :
                # [1] Get MBR values
                left = round(l.item()); right = round(r.item()); top = round(t.item()); bottom = round(b.item())
                mbrLeft.append(left);  mbrTop.append(top); mbrRight.append(right); mbrBottom.append(bottom)
                # [2] Get Confidence Values
                confValue = round(boxes[no].conf.item(),2);
                foundConfList.append(confValue)
                # [3] Get Class Values
                classNdx = int(boxes[no].cls.item());
                foundClassNdxList.append(classNdx)
                foundClassNamesList.append(labelStr[classNdx])

                # [4] Draw a rectangle for an onject
                THICKNESS = 1
```

```
                boxColor = cGreen if classNdx == 0 else cMagenta # plate and character color
                cv2.rectangle(colorImage,(left,top),(right,bottom),boxColor, thickness=THICKNESS)

                # Write confidence value
                colorConf = cRed if confValue < 0.5 else cCyan
                cv2.putText(colorImage, str(int(confValue*100)) , (mbrLeft[no],mbrTop[no]),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.4, colorConf)
                no+= 1

            # Print class name (using pillow for Korean character output)
            pillowFrame = Image.fromarray(colorImage)
            draw = ImageDraw.Draw(pillowFrame)
            for ndx in range(no):
                posX = int(round((mbrLeft[ndx]+mbrRight[ndx])/2))
                posY = mbrBottom[ndx] -10
                if foundClassNdxList[ndx] == 0 :
                    posY = mbrTop[ndx]-20
                # Write Class Name
                draw.text(xy=(posX,posY), text = foundClassNamesList[ndx], font=font, fill=cYellow)
            colorImage = np.array(pillowFrame)

        resultFileName = ''
        if saveMode == 'save':
            resultFileName = imgFileName.replace(originalExt,"_result"+newExt,1)
            cv2.imwrite(resultFileName,colorImage)

        if show_image(colorImage,resultFileName,0,0,showMode,'destroy') == ord('q'):
            cv2.destroyAllWindows()
            break

        print("...... [%d] of %d predicted."%(imgSeq, len(imgDataList)), end='\r')

    if saveMode == 'save':
        print(f'\nRecognized-images are saved at {dataFolder} folder')
    if showMode == 'auto': cv2.waitKey(0)
    print("\n","*** Prediction Ended(Stopped) !!! ***")

#================================================================================
def show_program_usage():
    print("*"*80)
    print("USAGE[train]:   python yolov8_detect.py  train  data-car.yaml  conf-car.yaml 100 ")
    print("USAGE[predict]: python yolov8_detect.py  predict trainResult/car/weights/best.pt  datasets/car/valid/images  manual  nosave")
    print("*"*80)

#================================================================================
# main
#================================================================================
if __name__ == "__main__":
    os.environ['KMP_DUPLICATE_LIB_OK']='True'   # Library Collision issue
    font =ImageFont.truetype('./MapoBackpacking.ttf',20)

    cBlue=(255,0,0); cRed=(0,0,255); cOrange=(0,128,255); cDarkGray=(80,80,80); cMagenta=(255,0,255)
```

```
    cGreen=(0,255,0); cYellow=(0,255,255); cLightGray=(160,160,160); cCyan=(255,255,0)

    # read class label names from the yaml file
    with open('./ultralytics/cfg/datasets/data-car.yaml', 'r',encoding='utf-8') as dataFile:
        yamlData = yaml.safe_load(dataFile)
        labelStr = yamlData['names']

    # Check Parameters
    if len(sys.argv) > 3  :
        screenWidth, screenHeight = get_screen_resolution()
        # Check Training/Prediction Mode
        if sys.argv[1] == 'train' :
            dataYaml = sys.argv[2]; configYaml=sys.argv[3];
            epochNo = int(sys.argv[4]);
            train_custom_data(dataYaml, configYaml, epochNo)
        elif sys.argv[1] == 'predict' :
            bestWeight = sys.argv[2];   dataFolder = sys.argv[3];
            showMode = sys.argv[4]; saveMode = sys.argv[5]
            predict_data(bestWeight, dataFolder, showMode, saveMode)
        else :
            show_program_usage()
    else :
        show_program_usage()

#-------------------------end of the source ----------------------------------------
```