

D:\myprojects\ultralytics\yolov8_detect.py

```
1  #*****
2  # Object Detecction using YOLO8
3  #*****
4  # https://docs.ultralytics.com/modes/predict/#working-with-results
5  # https://encord.com/blog/yolo-object-detection-guide/          # YOLO8 Explanation
6
7  # conda create -n yolov8_detect python=3.10 tk
8  # conda activate yolov8_detect
9  # conda install -c conda-forge ultralytics
10 # conda install pytorch torchvision torchaudio cpuonly -c pytorch
11 # conda install pyyaml matplotlib tqdm
12 # pip install opencv-python
13
14 '''
15 [1] For Korean character output, copy "NanumGothic.ttf" file to c:/windows/fonts folder
16 [2] Include the following at line 14 of utlralytics/ultralytics/metrics.py file
17     import matplotlib as mpl
18     from matplotlib import font_manager as fm
19
20     plt.rcParams['font.family'] = 'NanumGothic'
21     mpl.rcParams['axes.unicode_minus'] = False
22 '''
23
24 from ultralytics import YOLO
25 import os, sys, torch
26 import cv2, tkinter, yaml
27 import numpy as np
28 from PIL import ImageFont, ImageDraw, Image
29 from datetime import datetime
30
31 YOLO_MODEL = "yolov8m.yaml"
32 YOLO_WEIGHT = "yolov8m.pt"
33
34 INPUT_IMAGE_SIZE = 640
35 BATCH_SIZE = 16
36
37 #=====
38 # Get Screen Info.
39 #=====
40 def get_screen_resolution():
41     app = tkinter.Tk()
42     screenWidth = int(round(app.winfo_screenwidth() * 0.9))
43     screenHeight = int(round(app.winfo_screenheight() * 0.9))
44     app.destroy()
45
46     return screenWidth, screenHeight
47
48 #=====
49 # SHow OpenCV-style Image Window
50 #=====
51 def show_image(image, windowTitle="", x=0, y=0, showMode="manual", destroyMode='destroy') :
52     if windowTitle == "" : windowTitle = "Result Image"
53
54     sizeInfo = image.shape
55     imgHeight=sizeInfo[0]; imgWidth = sizeInfo[1]
56     heightRatio = float(screenHeight / imgHeight)
57     showWidth = int(round(heightRatio * imgWidth)); showHeight = screenHeight
```

```

58     resizedImage = cv2.resize(image, (showWidth, showHeight), interpolation=
cv2.INTER_CUBIC)
59
60     cv2.namedWindow(windowTitle,cv2.WINDOW_NORMAL)
61     cv2.moveWindow(windowTitle,x,y)
62     cv2.resizeWindow(windowTitle, showWidth, showHeight)
63     cv2.imshow(windowTitle,resizedImage)
64
65     miliSecond = 0 if showMode == 'manual' else 2
66
67     inKey = cv2.waitKey(miliSecond)
68     if destroyMode == 'destroy' : cv2.destroyWindow(windowTitle)
69
70     return inKey
71
72 #=====
73 # Make Prediction File List
74 #=====
75 def get_prediction_file_list(dataFolder) :
76     argFullPath = os.path.join(os.getcwd(), dataFolder)
77     print("data_path: ",argFullPath)
78     if os.path.isfile(argFullPath) :
79         imgDataList.append(argFullPath)
80     elif os.path.isdir(argFullPath) :
81         fileList = os.listdir(argFullPath)
82         imgDataList = [os.path.join(argFullPath,file) for file in fileList
83                        if (file.endswith(".jpg") or file.endswith(".png") or
84                           file.endswith(".JPG") or file.endswith(".PNG"))]
85
86     # make save image extention
87     originalExt = imgDataList[0][-4:]
88     originalExtUpper = originalExt.upper()
89     if originalExtUpper == '.JPG' :
90         newExt = '.png'
91     elif originalExtUpper == '.PNG' :
92         newExt = '.jpg'
93     else :
94         print("\n\n Image Format Mismatch !!! [.jpg or .png needed]")
95         sys.exit[1]
96
97     return imgDataList, originalExt, newExt
98
99 #=====
100 # Train the YOLO model with the Custom Data
101 #=====
102 def train_custom_data(dataYaml="data-car.yaml", cfgYaml="conf-car.yaml", epochNo = 100 )
:
103     # remove previous training result folder
104     '''
105     resultPath = os.path.join(os.getcwd(), resultFolder) + "/train"
106     if os.path.exists(resultPath) :
107         os.remove(resultPath)          '''
108
109     # load a pretrained model
110     model = YOLO(YOLO_MODEL)      # build a new model from scratch
111     model = YOLO(YOLO_WEIGHT)     # load a pretrained model (recommended for training)
112     #model = YOLO("best.pt")      # load a pretrained model
113
114     # train the model with the custom data
115     startTime = datetime.now()

```

```

116     model.train(data=dataYaml, cfg=cfgYaml, imgsz=INPUT_IMAGE_SIZE, batch=BATCH_SIZE,
epochs=epochNo, verbose=False)
117     endTime = datetime.now()
118     print(f"Training Time: ",{(endTime - startTime)})
119
120     # save the model
121     metrics = model.val() # evaluate model performance on the validation set
122
123     print("\n", "**** Training Ended !!!... [best.pt] saved under the runs/detect/train**
folder. ***")
124
125     #=====
126     # Get DataSet File(s), Predict and show the Result Images
127     #=====
128     def predict_data(weightToUse = "trainResult/car/weights/best.pt", dataFolder = "
datasets/car/test/",
129                     showMode = 'manual', saveMode = 'save' ) :
130         imgDataList = []
131
132         imgDataList, originalExt, newExt = get_prediction_file_list(dataFolder)
133
134         # load a yolo model and weights
135         model = YOLO(YOLO_MODEL) # build a new model from scratch
136         model = YOLO(weightToUse) # load a pretrained model
137
138         #-----
139         ---
140         for imgSeq, imgFileName in enumerate(imgDataList) :
141             # pridict for each image
142             colorImage = cv2.imread(imgFileName,cv2.IMREAD_COLOR)
143             results = model(imgFileName, max_det = 30, verbose=False)#, conf = 0.25) #
predict maximum 30 objects from an image
144
145             #-----
146             # Drawing recognition results for all found objects (mbr, conf, className)
147             #-----
148             for result in results:
149                 boxes = result.boxes # Boxes object(tensor) for bbox outputs
150
151                 mbrLeft =[]; mbrRight=[]; mbrTop=[]; mbrBottom=[] # MBR positional
Values
152                 foundConfList = []; foundClassNdxList = []; foundClassNamesList = []
153
154                 no=0
155                 #-----
156                 for l,t,r,b in boxes.xyxy :
157                     # [1] Get MBR values
158                     left = round(l.item()); right = round(r.item()); top = round(t.item());
bottom = round(b.item())
159                     mbrLeft.append(left); mbrTop.append(top); mbrRight.append(right);
mbrBottom.append(bottom)
160                     # [2] Get Confidence Values
161                     confValue = round(boxes[no].conf.item(),2);
162                     foundConfList.append(confValue)
163                     # [3] Get Class Values
164                     classNdx = int(boxes[no].cls.item());
165                     foundClassNdxList.append(classNdx)
166                     foundClassNamesList.append(labelStr[classNdx])
167
168                     # [4] Draw a rectangle for an onject
169                     THICKNESS = 1

```

```

169         boxColor = cGreen if classNdx == 0 else cMagenta # plate and character
170         color
171         cv2.rectangle(colorImage,(left,top),(right,bottom),boxColor, thickness=
172         THICKNESS)
173
174         # Write confidence value
175         colorConf = cRed if confValue < 0.5 else cCyan
176         cv2.putText(colorImage, str(int(confValue*100)) , (mbrLeft[no],mbrTop[no])
177
178         ,
179         cv2.FONT_HERSHEY_SIMPLEX, 0.4, colorConf)
180         no+= 1
181
182         # Print class name (using pillow for Korean character output)
183         pillowFrame = Image.fromarray(colorImage)
184         draw = ImageDraw.Draw(pillowFrame)
185         for ndx in range(no):
186             posX = int(round((mbrLeft[ndx]+mbrRight[ndx])/2))
187             posY = mbrBottom[ndx] -10
188             if foundClassNdxList[ndx] == 0 :
189                 posY = mbrTop[ndx]-20
190             # Write Class Name
191             draw.text(xy=(posX,posY), text = foundClassNamesList[ndx], font=font,
192             fill=cYellow)
193             colorImage = np.array(pillowFrame)
194
195             resultFileName = ''
196             if saveMode == 'save':
197                 resultFileName = imgFileName.replace(originalExt,"_result"+newExt,1)
198                 cv2.imwrite(resultFileName,colorImage)
199
200             if show_image(colorImage,resultFileName,0,0,showMode,'destroy') == ord('q'):
201                 cv2.destroyAllWindows()
202                 break
203
204             print("..... [%d] of %d predicted."%(imgSeq, len(imgDataList)), end='\n')
205
206             if saveMode == 'save':
207                 print(f'\nRecognized-images are saved at {dataFolder} folder')
208             if showMode == 'auto': cv2.waitKey(0)
209             print("\n", "*** Prediction Ended(Stopped) !!! ***")
210
211 #=====
212 ==
213 def show_program_usage():
214     print(""*80)
215     print("USAGE[train]:  python yolov8_detect.py  train  data-car.yaml  conf-car.yaml
216     100 ")
217     print("USAGE[predict]: python yolov8_detect.py  predict
218     trainResult/car/weights/best.pt  datasets/car/valid/images  manual  nosave")
219     print(""*80)
220
221 #=====
222 ==
223 # main
224 #=====
225 ==
226 if __name__ == "__main__":
227     os.environ['KMP_DUPLICATE_LIB_OK']='True' # Library Collision issue
228     font =ImageFont.truetype('./MapoBackpacking.ttf',20)
229
230     cBlue=(255,0,0); cRed=(0,0,255); cOrange=(0,128,255); cDarkGray=(80,80,80); cMagenta=
231     (255,0,255)

```

```

221     cGreen=(0,255,0); cYellow=(0,255,255); cLightGray=(160,160,160); cCyan=(255,255,0)
222
223     # read class label names from the yaml file
224     with open('./ultralalytics/cfg/datasets/data-car.yaml', 'r',encoding='utf-8') as
dataFile:
225         yamlData = yaml.safe_load(dataFile)
226         labelStr = yamlData['names']
227
228     # Check Parameters
229     if len(sys.argv) > 3 :
230         screenWidth, screenHeight = get_screen_resolution()
231         # Check Training/Prediction Mode
232         if sys.argv[1] == 'train' :
233             dataYaml = sys.argv[2]; configYaml=sys.argv[3];
234             epochNo = int(sys.argv[4]);
235             train_custom_data(dataYaml, configYaml, epochNo)
236         elif sys.argv[1] == 'predict' :
237             bestWeight = sys.argv[2]; dataFolder = sys.argv[3];
238             showMode = sys.argv[4]; saveMode = sys.argv[5]
239             predict_data(bestWeight, dataFolder, showMode, saveMode)
240         else :
241             show_program_usage()
242     else :
243         show_program_usage()
244
245     #-----end of the source -----

```