Lab 30

Developing a full-fledged project using different DL algorithms and python packages from scratch​

For an End-to-end model deployment, we will use Visual Studio Code. As it is efficient to maintain modular development. Here, we will use pretrained YOLO model to make predictions.

YOLO or You Only Look Once, is a popular real-time object detection algorithm. YOLO combines what was once a multi-step process, using a single neural network to perform both classification and prediction of bounding boxes for detected objects. As such, it is heavily optimized for detection performance and can run much faster than running two separate neural networks to detect and classify objects separately. It does this by repurposing traditional image classifiers to be used for the regression task of identifying bounding boxes for objects. It looks at the entire image at once, and only once — hence the name You Only Look Once — which allows it to capture the context of detected objects. This halves the number of false-positive detections it makes over R-CNNs which look at different parts of the image separately. Additionally, YOLO can generalize the representations of various objects, making it more applicable to a variety of new environments. Now that we have a general overview of YOLO, let’s use YOLO to make a full-fledged model.

Instructions of implementation

1. Create Virtual Environment

Open Terminal and enter the following commands.

python -m venv yologui

2. Activate Virtual Environment

yologui\Scripts\activate

3. Install YOLOv8’s dependencies.

pip install -r https://raw.githubusercontent.com/ultralytics/ultralytics/main/requirements.txt

4. Install YOLOv8.

pip install ultralytics

5. Now, you need to create a new file named as postprocessing.py. Open that file and add the following code to make main script cleaner.

import cv2  
  
def resize\_image(img, scale\_percent) :  
 # Calculate new size  
 width = int(img.shape[1] \* scale\_percent / 100)  
 height = int(img.shape[0] \* scale\_percent / 100)  
 dim = (width, height)  
 # Resize image  
 resized = cv2.resize(img, dim, interpolation = cv2.INTER\_AREA)  
 return resized  
  
def draw\_box(img, result, class\_list) :  
 # Get information from result  
 xyxy= result.boxes.xyxy.numpy()  
 confidence= result.boxes.conf.numpy()  
 class\_id= result.boxes.cls.numpy().astype(int)  
 # Get Class name  
 class\_name = [class\_list[x] for x in class\_id]  
 # Pack together for easy use  
 sum\_output = list(zip(class\_name, confidence,xyxy))  
 # Copy image, in case that we need original image for something  
 out\_image = img.copy()  
 for run\_output in sum\_output :  
 # Unpack  
 label, con, box = run\_output  
 # Choose color  
 box\_color = (0, 0, 255)  
 text\_color = (255,255,255)  
 # Draw object box  
 first\_half\_box = (int(box[0]),int(box[1]))  
 second\_half\_box = (int(box[2]),int(box[3]))  
 cv2.rectangle(out\_image, first\_half\_box, second\_half\_box, box\_color, 2)  
 # Create text  
 text\_print = '{label} {con:.2f}'.format(label = label, con = con)  
 # Locate text position  
 text\_location = (int(box[0]), int(box[1] - 10 ))  
 # Get size and baseline  
 labelSize, baseLine = cv2.getTextSize(text\_print, cv2.FONT\_HERSHEY\_SIMPLEX, 1, 2)   
 # Draw text's background  
 cv2.rectangle(out\_image   
 , (int(box[0]), int(box[1] - labelSize[1] - 10 ))  
 , (int(box[0])+labelSize[0], int(box[1] + baseLine-10))  
 , box\_color , cv2.FILLED)   
 # Put text  
 cv2.putText(out\_image, text\_print ,text\_location  
 , cv2.FONT\_HERSHEY\_SIMPLEX , 1  
 , text\_color, 2 ,cv2.LINE\_AA)  
 return out\_image

When you reach 5th step, your folder should be something like this.

|-project folder  
 |-yologui  
 |-postprocessing.py

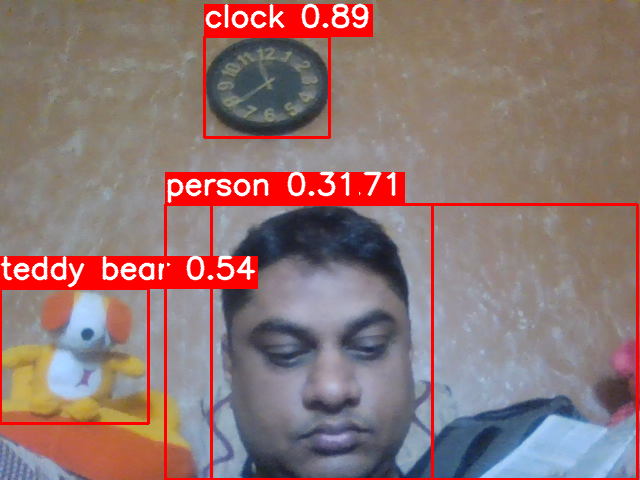
6. Create a yolo script by creating a new file called as yolo.py.

# https://docs.ultralytics.com/python/  
from ultralytics import YOLO  
import cv2  
from postprocessing import \*  
  
# Set up model and parameter  
model = YOLO("yolov8s.pt")  
class\_list = model.model.names  
scale\_show = 100  
# Read Video  
video = cv2.VideoCapture(0)  
# Run Loop  
while True :  
 ret, frame = video.read()  
 if ret :  
 results = model.predict(frame)  
 labeled\_img = draw\_box(frame, results[0], class\_list)  
 display\_img = resize\_image(labeled\_img, scale\_show)  
 # Show Image  
 cv2.imshow('Frame', display\_img)  
 # Press Q on keyboard to exit  
 if cv2.waitKey(25) & 0xFF == ord('q'): break  
# Break the loop if not read  
 else: break  
  
# When everything done, release  
video.release()  
# Closes all the frames  
cv2.destroyAllWindows()

After that, try to execute the script and you should get the object detection by YOLOv8.

python yolo.py

And this is the result, press “Q” to exit when satisfy.



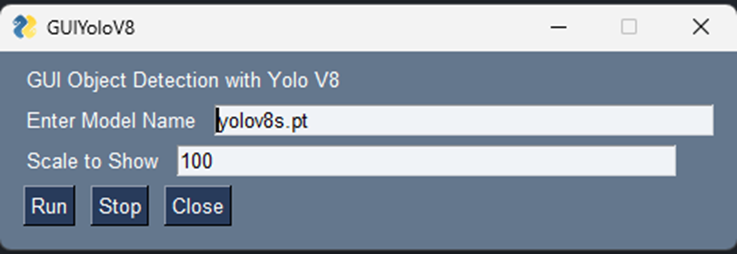
Adding GUI frame into a new script called main.py as below.

import PySimpleGUI as sg  
from ultralytics import YOLO  
import cv2  
from postprocessing import \*  
  
# Create Layouy of the GUI  
layout = [   
 [sg.Text('GUI Object Detection with Yolo V8')],  
 [sg.Text('Enter Model Name'), sg.InputText(default\_text="yolov8s.pt",key='model\_name')],  
 [sg.Text('Scale to Show'), sg.InputText(default\_text="100", key= 'scale\_percent')],  
 [sg.Button('Run'), sg.Button('Stop'), sg.Button('Close')],  
 [sg.Image(filename='', key='image')]  
 ]  
  
# Create the Window  
window = sg.Window('GUIYoloV8', layout, finalize=True)  
run\_model = False  
# Event Loop to process "events"  
while True:  
 event, values = window.read(timeout=1)  
 # When press Run  
 if event == 'Run' :   
 # Set up model and parameter  
 model = YOLO(values['model\_name'])  
 class\_list = model.model.names  
 scale\_show = int(values['scale\_percent'])  
 # Read Video  
 video = cv2.VideoCapture(0)  
 # Run Signal  
 run\_model = True  
 # When press Stop or close window or press Close  
 elif event in ('Stop', sg.WIN\_CLOSED, 'Close'):  
 if run\_model :   
 run\_model = False # Stop running  
 video.release() # Release video  
 window['image'].update(filename='') # Destroy picture  
 # When close window or press Close  
 if event in (sg.WIN\_CLOSED, 'Close'): break  
 # Run Model  
 if run\_model :   
 ret, frame = video.read()  
 if ret :  
 results = model.predict(frame)  
 labeled\_img = draw\_box(frame, results[0], class\_list)  
 display\_img = resize\_image(labeled\_img, scale\_show)  
 # Show Image  
 imgbytes = cv2.imencode('.png', display\_img)[1].tobytes()  
 window['image'].update(data=imgbytes)  
 else:   
 # Break the loop if not read  
 video.release()  
 run\_model = False  
  
# Close window  
window.close()

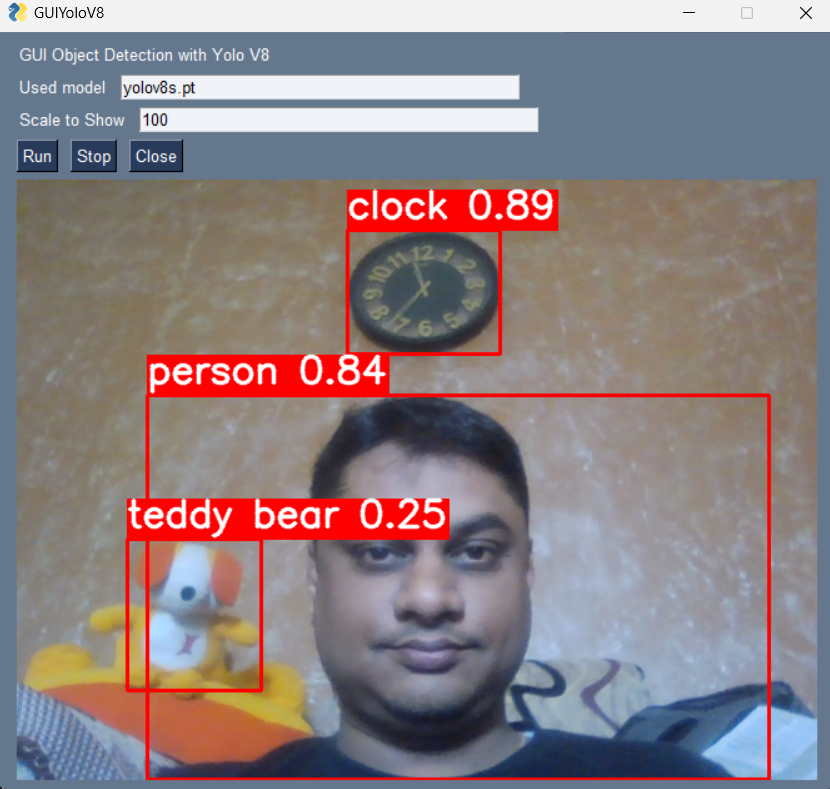
After that, we can run the main script.

python main.py

And we can have our Object Detection GUI as pictures below.



Initial Stage



After clicking Run.