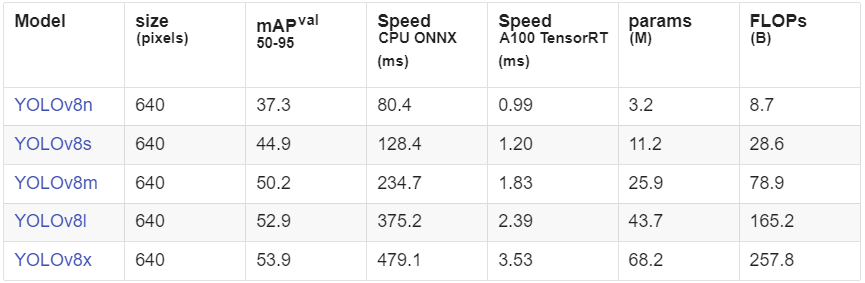
* Firstly, install PyTorch on our machine

<https://pytorch.org/get-started/locally/>

* Install Ultralytics: pip install ultralytics
* Go to <https://docs.ultralytics.com/quickstart/> to refer the documentation of Ultralytics

**Inference YOLOv8 on images**

Let’s try a simple inference of YOLOv8 using images, videos, and webcam. There are multiple pretrained yolov8 weights that we can use for inference. The weights can be referred to at <https://docs.ultralytics.com/tasks/detect/#models>.



There are 2 ways we can do this:

1. Use Ultralytics with command line interface CLI (<https://docs.ultralytics.com/usage/cli/>)

Ultralytics yolo commands use the following syntax:

A close-up of a text

Description automatically generated

For example, we can use a pre-trained Yolov8n model to run predictions (detection/segmentation) on images (<https://docs.ultralytics.com/usage/cli/#predict>) :

yolo detect predict model=yolov8n.pt source='https://ultralytics.com/images/bus.jpg'

yolo segment predict model=yolov8n-seg.pt source='https://ultralytics.com/images/bus.jpg'

1. Use Ultralytics with Python

<https://docs.ultralytics.com/quickstart/#use-ultralytics-with-python>

<https://docs.ultralytics.com/usage/python/>

A black screen with text

Description automatically generated

If we run the above code in vscode, the code will show our image and it will close very quickly. Therefore, we need to add a delay using OpenCV cv2.waitKey function.

A bus with people standing on the side of the road

Description automatically generatedA screenshot of a computer

Description automatically generated

After running the input through the model, it returns an array of results for each input image. As we provided only a single image, it returns an array with a single item that you can extract like this:

result = results[0]

The result contains detected objects and convenient properties to work with them. The most important one is the boxes array with information about detected bounding boxes on the image. You can determine how many objects it detected by running the len function:

len(result.boxes)

When we ran this, let say we got "2", which means that there are two boxes detected.

Then we can analyze each box either in a loop or manually. Let's get the first one:

box = result.boxes[0]

The box object contains the properties of the bounding box, including:

xyxy – the coordinates of the box as an array [x1,y1,x2,y2]

cls – the ID of object type

conf – the confidence level of the model about this object. If it's very low, like < 0.5, then you can just ignore the box.

Let's print information about the detected box:

print("Object type:", box.cls)

print("Coordinates:", box.xyxy)

print("Probability:", box.conf)

For the first box, you will receive the following information:

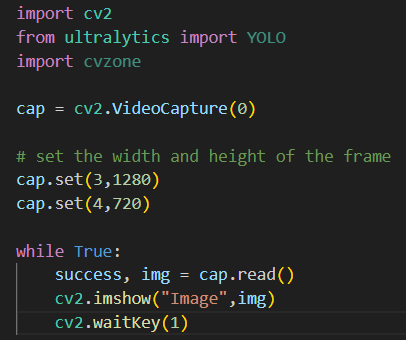
Object type: tensor([16.])

Coordinates: tensor([[261.1901, 94.3429, 460.5649, 312.9910]])

Probability: tensor([0.9528])

**Inference YOLOv8 on webcam using OpenCV and cvzone**

As we will be utilizing another python package called cvzone, we will need to start by installing the package using pip i.e. pip install cvzone. We then test the webcam using OpenCV.



Then, we can run the inference of the model on the webcam frames. This can be achieved using the previous predict method, unpack the bounding boxes and show them on the screen. Note that we also convert the x-y values into integers so that they can be used in other OpenCV functions.

A screen shot of a computer program

Description automatically generated

If we want a bit more fancier bounding box style, we can also draw the bounding box using cvzone. We also add the confidence score and class name here. Note that we also round the values to 2 decimal places using math package (therefore need to add import math at the beginning).

A computer screen shot of many colorful text

Description automatically generated

Inference YOLOv8 on webcam using Ultralytics Annotator

Another method to run the inference of yolov8 on webcam is by using Ultralytics Annotator. Firstly, we will need to import the package by typing:

from ultralytics.yolo.utils.plotting import Annotator

Then, the annotator class needs to be initialized by taking the frame of the webcam as input annotator = Annotator(frame). We can then use the attribute annotator.box\_label to draw the bounding box by taking the bounding box info and class name as input annotator.box\_label(b, model.names[int(c)]). Finally, we can get the annotated frame by using the result attribute frame = annotator.result().

A computer screen shot of text

Description automatically generated

Training YOLOv8 using custom dataset

<https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov8-object-detection-on-custom-dataset.ipynb#scrollTo=jbVjEtPAkz3j>