



Published in Towards Data Science



Thomas A Dorfer

Follow

Mar 23 · 7 min read · ✨ · 🎧 Listen

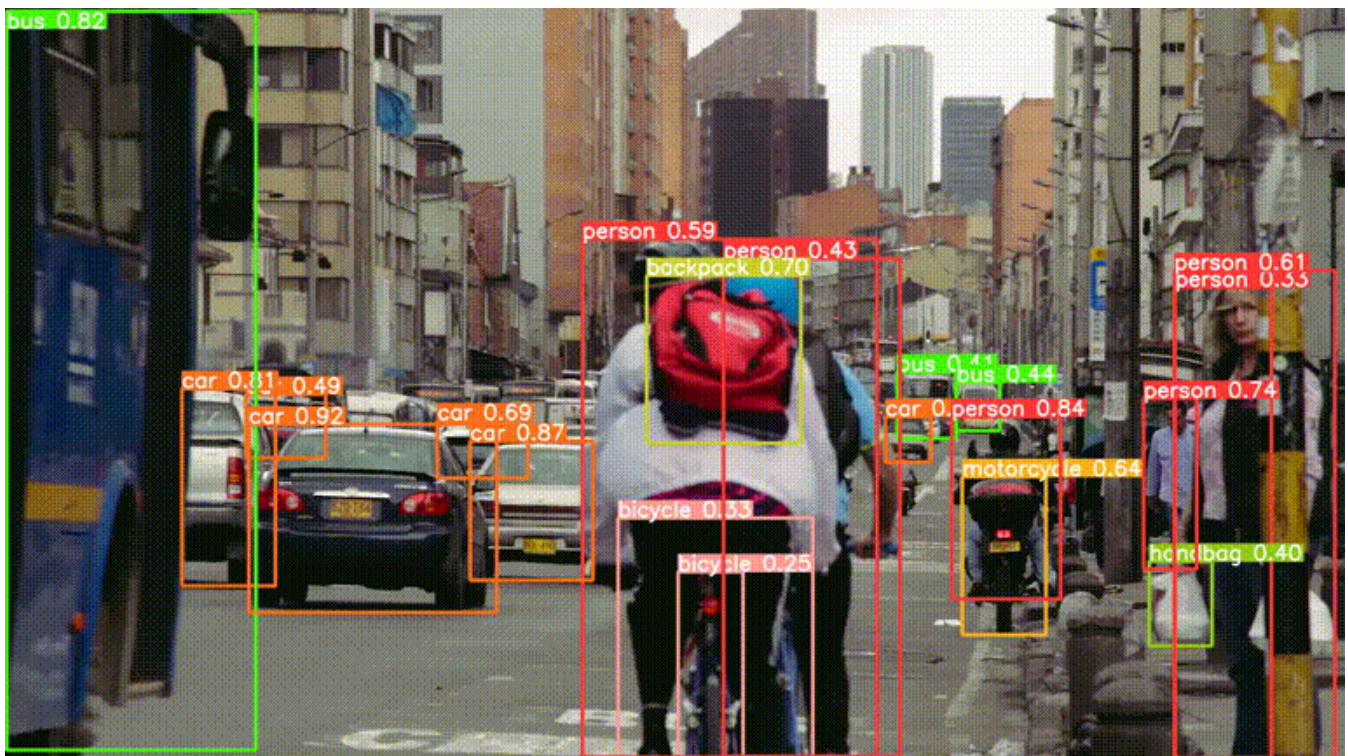


Save



# Enhanced Object Detection: How To Effectively Implement YOLOv8

A practical guide to object detection in images, videos, and real-time webcam feed using both CLI and Python



Video by [Camilo Calderón](#) on [Pexels](#). Converted to GIF format by the Author.

## Introduction

Object detection, a subfield of computer vision, is primarily concerned with the identification and localization of objects in images or videos with a certain degree

of confidence. An identified object is generally annotated with a bounding box, which provides information to the viewer about the object's nature and location in the scene.

In 2015, the debut of YOLO, or **You Only Look Once**, shook the world of computer vision as its system was capable of real-time object detection with astounding accuracy and speed. Since then, YOLO has undergone several iterations of improvements in prediction accuracy and efficiency, eventually culminating in the release of its latest family member: **YOLOv8** by Ultralytics.

YOLOv8 comes in five versions: nano (n), small (s), medium (m), large (l), and extra large (x). Their respective improvements can be demonstrated by their mean average precisions (mAP) and latencies, evaluated by the the COCO val2017 dataset.

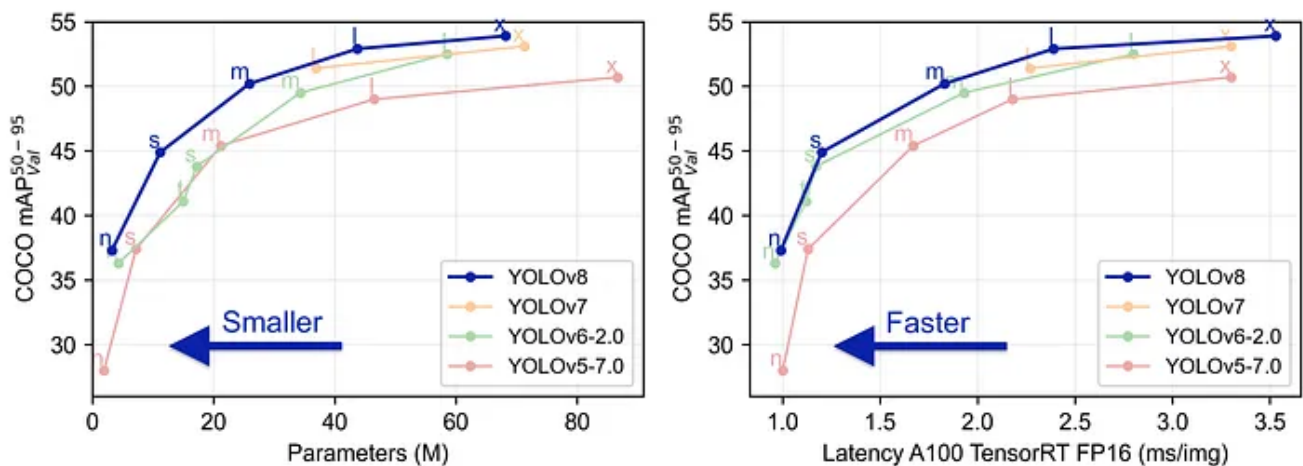


Image by Ultralytics. License: GNU General Public License.

Compared to previous versions, YOLOv8 is not only faster and more accurate, but it also requires fewer parameters to achieve its performance and, as if that wasn't enough, comes with an intuitive and easy-to-use command-line interface (CLI) as well as a Python package, providing a more seamless experience for users and developers.

In this article, I will demonstrate how YOLOv8 can be applied to detect objects in static images, videos, and a live webcam using both CLI and Python.

Without further ado, let's get into it!



## Installation

All you need to do to get started with YOLOv8 is to run the following command in your terminal:

```
pip install ultralytics
```

This will install YOLOv8 via the `ultralytics` `pip` package.

## Image Detection

Object detection in static images has proven useful in a variety of domains, such as surveillance, medical imaging, or retail analytics. Whatever domain you choose to apply your detection system, YOLOv8 has made it incredibly simple for you to do so.

Below is the raw image that we're going to perform object detection on.



Photo by [James Coleman](#) on [Unsplash](#).

In order to run YOLOv8, we will look into both CLI and Python implementations. While in this particular case we'll be using a `jpg` image, YOLOv8 supports a variety

of different image formats.

## CLI

Assuming we'd like to run the extra large YOLOv8x on our image (let's call it `img.jpg`), the following command can be put into the CLI:

```
yolo detect predict model=yolov8x.pt source="img.jpg" save=True
```

Here, we specify the following arguments: `detect` to use object detection, `predict` to perform a prediction task, `model` to select the model version, `source` to provide the file path of our image, and `save` to save the processed image with the object's bounding boxes and their predicted classes and class probabilities.

## Python

In Python, the exact same task can be achieved with the following intuitive and low-code solution:

```
from ultralytics import YOLO

model = YOLO('yolov8x.pt')
results = model('img.jpg', save=True)
```

Whether you use the CLI or Python; in either case, the saved, processed image looks as follows:

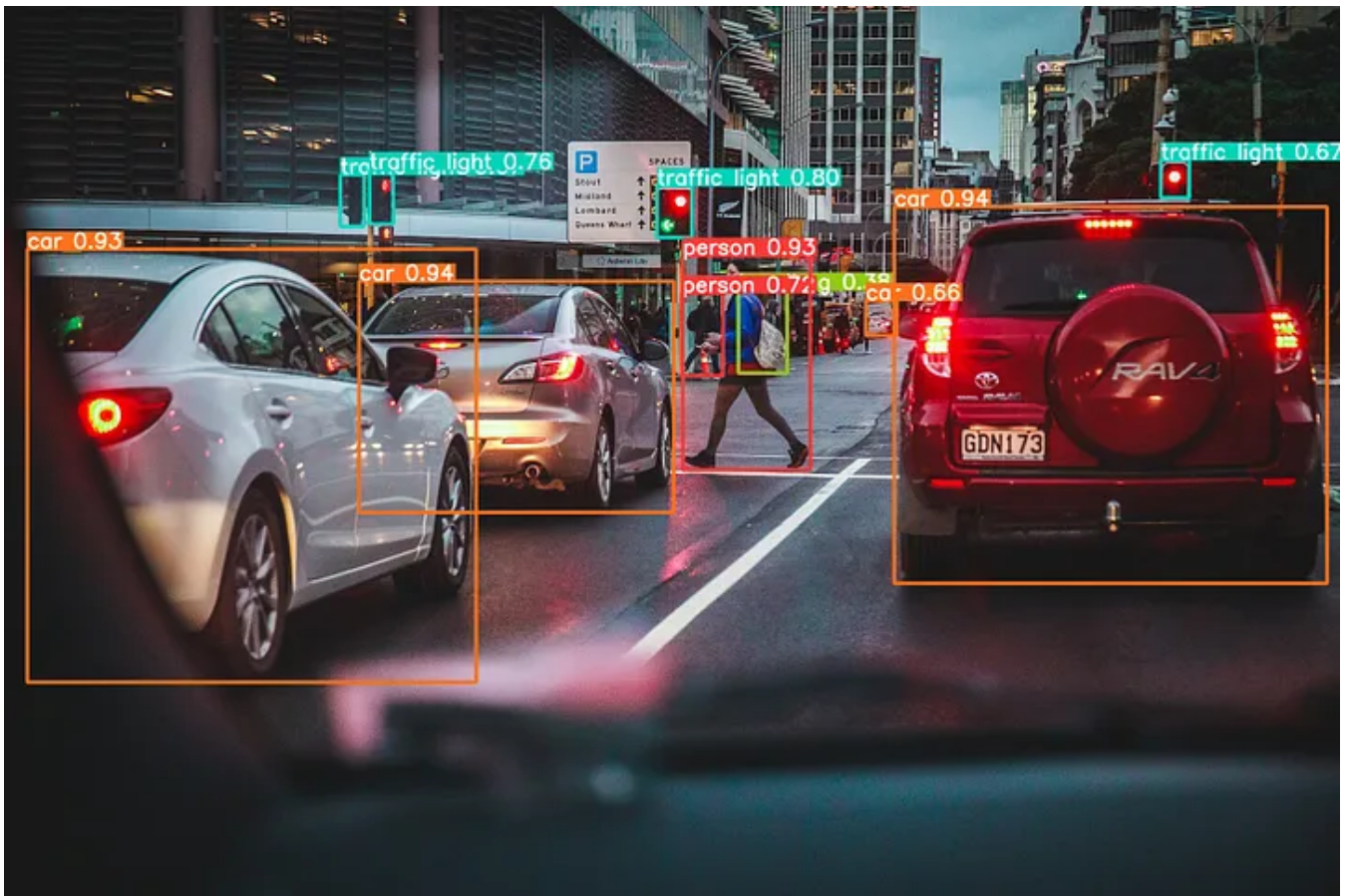


Photo by [James Coleman](#) on [Unsplash](#). Processed with YOLOv8 by the Author.

We can clearly see the bounding boxes around every object it detected, as well as their corresponding class labels and probabilities.

## Video Detection

Performing object detection on video files is almost identical to image files, with the only difference being the source file format. Just like with images, YOLOv8 supports a variety of different video formats that can be fed as an input to the model. In our case, we'll be using an `mp4` file.

Let's again look at both CLI and Python implementations. For faster computation, let's now use the YOLOv8m model instead of the extra large version.

### CLI

```
yolo detect predict model=yolov8m.pt source="vid.mp4" save=True
```

### Python



```
from ultralytics import YOLO

model = YOLO('yolov8m.pt')
results = model('vid.mp4', save=True)
```

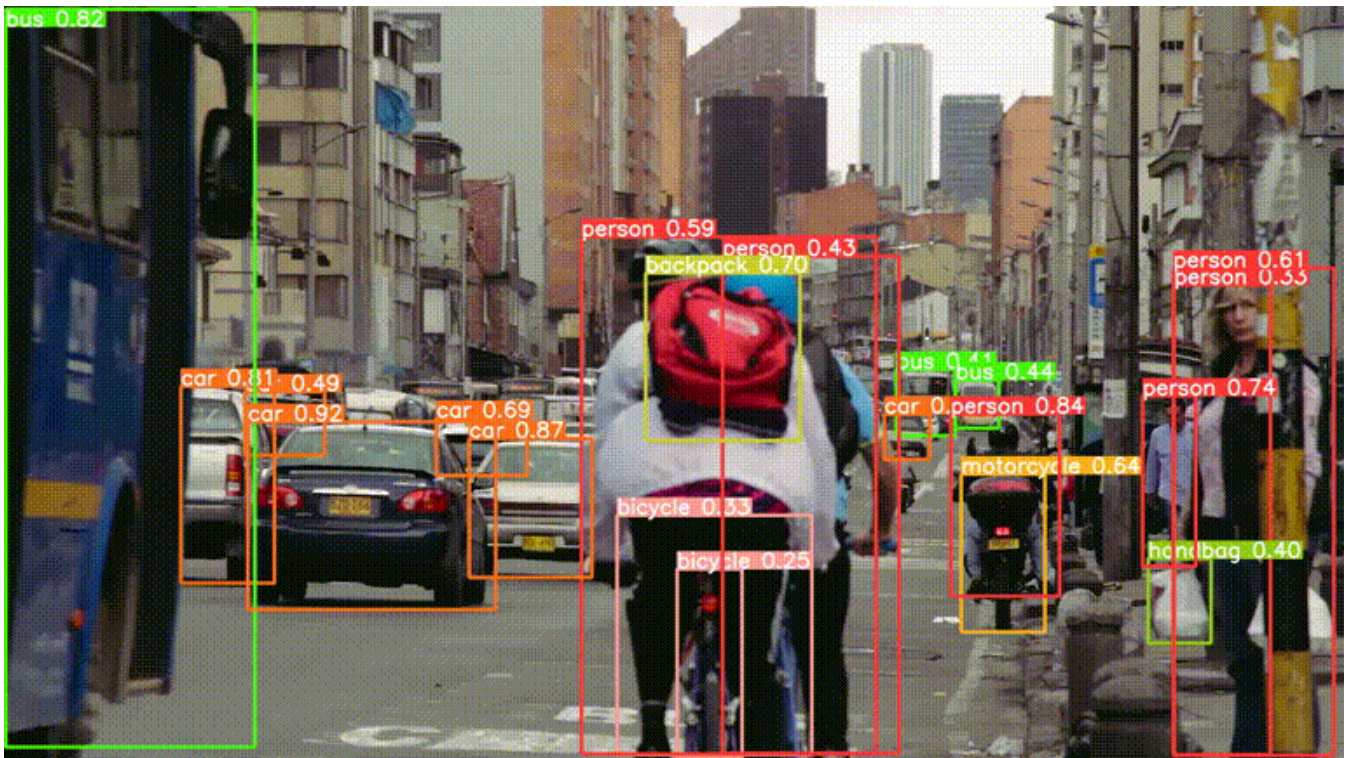
First, let's inspect our raw, `vid.mp4` file before we perform object detection on it:



Video by [Camilo Calderón](#) on [Pexels](#). Converted to GIF format by the Author.

The video shows a scene of a busy city with lots of traffic, including cars, busses, trucks, and cyclists, as well as some people on the right side apparently waiting for a bus.

After processing this file using YOLOv8's medium version, we get the following result:



Video by [Camilo Calderón](#) on [Pexels](#). Processed with YOLOv8m and converted to GIF format by the Author.

Again, we can see that YOLOv8m does a really good job at accurately capturing the objects in the scene. It even detects smaller objects that are part of a larger whole, such as a person on a bicycle wearing a backpack.

## Live Detection

Finally, let's take a look at what is required to detect objects in a live webcam feed. To do so, I'll use my personal webcam and, just like before, both CLI and Python approaches.

To reduce the latency and subsequently the lag in the video, I'll be using the light-weight nano version of YOLOv8.

### CLI

```
yolo detect predict model=yolov8n.pt source=0 show=True
```

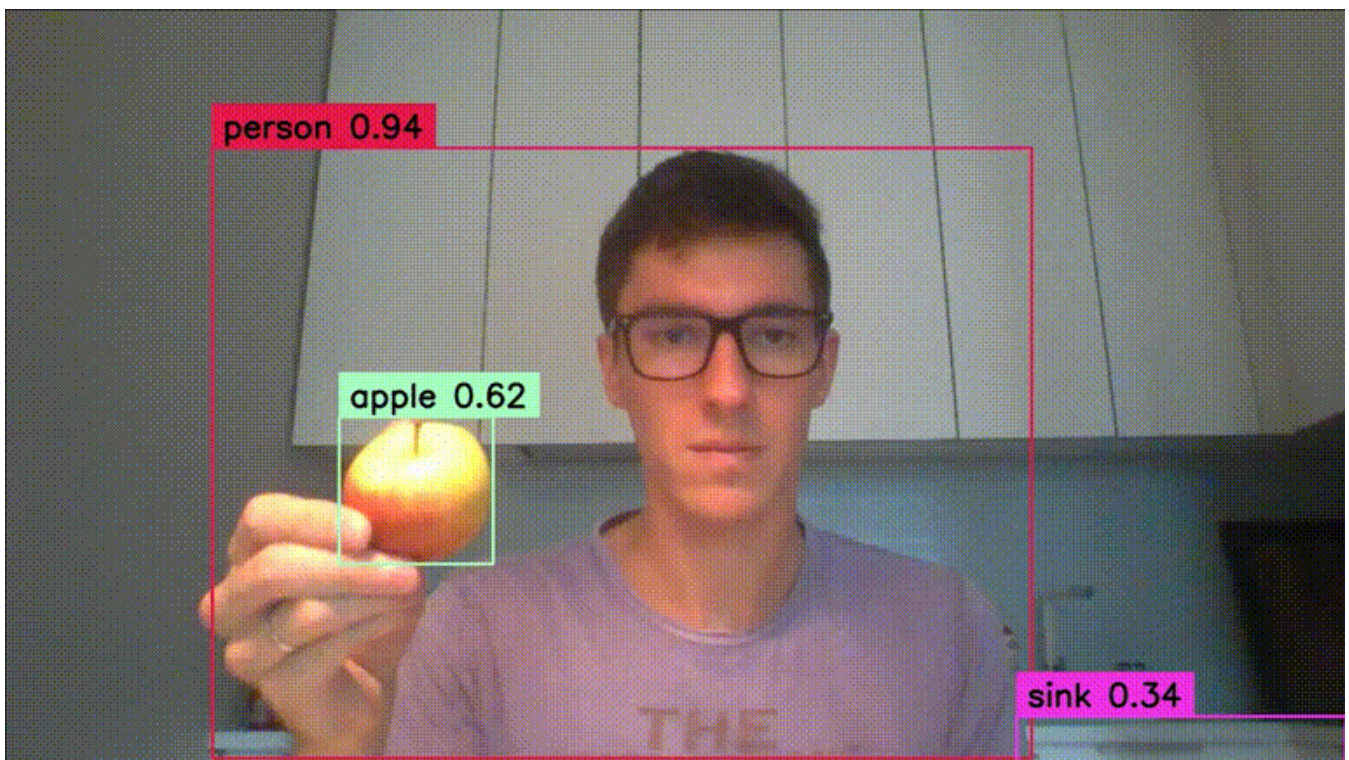
Most of these arguments are identical to what we have seen above for image and video files, with the exception of `source`, which allows us to specify which video source to use. In my case, it's the inbuilt webcam (0).



## Python

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
model.predict(source="0", show=True)
```

[Open in app](#)




GIF by the Author. Recorded from webcam using YOLOv8n.

Impressive! Despite the rather low video quality and poor lighting conditions, it still captures the objects pretty well and even detects some objects in the background, such as the olive oil and vinegar bottles on the left and the sink on the right.

It is worth noting that while these intuitive CLI commands and low-code Python solutions are great ways to quickly get started on an object detection task, they do have limitations when it comes to custom configurations. For instance, if we'd like



to configure the aesthetics of the bounding boxes or perform a simple task such as counting and displaying the number of objects that are being detected at any given time, we would have to code up our own custom implementation using packages such as cv2 or supervision.

In fact, the webcam footage above has been recorded using the following Python code in order to adjust the webcam's resolution and custom-define the bounding boxes and their annotations. (Note: This was mainly done to make the GIF above more presentable. The CLI and  77 |  1 |  ons shown above would suffice to produce similar outcomes.)

```
import cv2
import supervision as sv
from ultralytics import YOLO

def main():

    # to save the video
    writer= cv2.VideoWriter('webcam_yolo.mp4',
                           cv2.VideoWriter_fourcc(*'DIVX'),
                           7,
                           (1280, 720))

    # define resolution
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

    # specify the model
    model = YOLO("yolov8n.pt")

    # customize the bounding box
    box_annotator = sv.BoxAnnotator(
        thickness=2,
        text_thickness=2,
        text_scale=1
    )

    while True:
        ret, frame = cap.read()
        result = model(frame, agnostic_nms=True)[0]
        detections = sv.Detections.from_yolov8(result)
        labels = [
            f"{model.model.names[class_id]} {confidence:0.2f}"
            for _, confidence, class_id, _
```

```
        in detections
    ]
    frame = box_annotator.annotate(
        scene=frame,
        detections=detections,
        labels=labels
    )

    writer.write(frame)

    cv2.imshow("yolov8", frame)

    if (cv2.waitKey(30) == 27): # break with escape key
        break

    cap.release()
    writer.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```

While the details in this code are beyond the scope of this article, here's a great reference that uses a similar approach in case you are interested in upping your object detection game:

## Conclusion



YOLOv8 does not only outperform its predecessors in accuracy and speed, but it also considerably improves user experience through an extremely easy-to-use CLI and low-code Python solutions. It also comes in five different model versions, providing the user with the opportunity to choose depending on their individual needs and tolerance limits for latency and accuracy.

Whether your goal is to perform object detection on static images, videos, or a live webcam, YOLOv8 enables you to do this in a seamless manner. However, should your application require custom configurations, you may have to resort to additional computer vision packages such as `cv2` and `supervision`.

### More Resources

- [ultralytics/ultralytics: NEW — YOLOv8 🚀 in PyTorch > ONNX > CoreML > TFLite \(github.com\)](#)
- [YOLOv8 Docs \(ultralytics.com\)](#)
- [YOLOv8 Object Counting in Real-time with Webcam, OpenCV and Supervision — YouTube](#)

### Liked this article?

Let's connect! You can find me on [Twitter](#) and [LinkedIn](#).

If you like to support my writing, you can do so through a [Medium Membership](#), which provides you access to all my stories as well as those of thousands of other writers on Medium.

#### Join Medium with my referral link - Thomas A Dorfer

Read every story from Thomas A Dorfer (and thousands of other writers on Medium). Your membership fee directly supports...

[medium.com](https://medium.com)

[Artificial Intelligence](#)[Machine Learning](#)[Data Science](#)[Computer Vision](#)[Python](#)

---

## Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Emails will be sent to dnolivieri@gmail.com. [Not you?](#)



Get this newsletter