Model Training with Ultralytics YOLO

Training a deep learning model involves feeding it data and adjusting its parameters so that it can make accurate predictions. Train mode in Ultralytics YOLOv8 is engineered for effective and efficient training of object detection models, fully utilizing modern hardware capabilities. This guide aims to cover all the details you need to get started with training your own models using YOLOv8's robust set of features.

What is YOLO exactly?

YOLO (You Only Look Once) is a method / way to do object detection. It is the algorithm /strategy behind how the code is going to detect objects in the image.

Earlier detection frameworks, looked at different parts of the image multiple times at different scales and repurposed image classification technique to detect objects. This approach is slow and inefficient.

YOLO takes entirely different approach. It looks at the entire image only once and goes through the network once and detects objects. Hence the name. It is very fast. That's the reason it has got so popular.

How does YOLO work? YOLO Architecture The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.

YOLO architecture The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is non-maximum suppression (NMS). NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all

represent the same object. NMS is used to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

Now, let us look into the improvements that the later versions of YOLO have brought to the parent model.

Common classes that YOLO models, including YOLOv4, are trained to detect in datasets like COCO include:

- Person
- Car
- Bicycle
- Truck
- Bus
- Traffic light
- Stop sign
- Animal (various categories)
- Chair
- Table

```
# from ultralytics import YOLO

# # Create a new YOLO model from scratch
# model = YOLO('yolov8n.yaml')

# # Load a pretrained YOLO model (recommended for training)
# model = YOLO('yolov8n.pt')

# # Train the model using the 'coco128.yaml' dataset for 3 epochs
# results = model.train(data='coco128.yaml', epochs=3)

# # Evaluate the model's performance on the validation set
# results = model.val()
```

YOLO v8

At the time of writing this article, the release of YOLO v8 has been confirmed by Ultralytics that promises new features and improved performance over its predecessors. YOLO v8 boasts of a new API that will make training and inference much easier on both CPU and GPU devices and the framework will support previous YOLO versions. The developers are still working on releasing a scientific paper that will include a detailed description of the model architecture and performance.

```
!pip install ultralytics
```

```
Requirement already satisfied: ultralytics in
/usr/local/lib/python3.10/dist-packages (8.0.222)
Requirement already satisfied: matplotlib>=3.3.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
Requirement already satisfied: numpy>=1.22.2 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (1.23.5)
Requirement already satisfied: opencv-python>=4.6.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
Requirement already satisfied: pillow>=7.1.2 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (1.11.4)
Requirement already satisfied: torch>=1.8.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics)
(2.1.0+cu118)
Requirement already satisfied: torchvision>=0.9.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics)
(0.16.0+cu118)
Requirement already satisfied: tqdm>=4.64.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.1)
Requirement already satisfied: pandas>=1.1.4 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (1.5.3)
Requirement already satisfied: seaborn>=0.11.0 in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (0.12.2)
Requirement already satisfied: psutil in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in
/usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: thop>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from ultralytics)
(0.1.1.post2209072238)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (4.45.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
```

```
>ultralytics) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4-
>ultralytics) (2023.3.post1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.23.0-
>ultralytics) (2023.11.17)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (3.13.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (4.5.0)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (1.12)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (3.2.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (3.1.2)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torch>=1.8.0-
>ultralytics) (2.1.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
>matplotlib>=3.3.0->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0-
>ultralytics) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.8.0-
>ultralytics) (1.3.0)
```

```
import numpy as np
import pandas as pd
import cv2

from sklearn.utils import shuffle
from matplotlib.patches import Rectangle
import matplotlib.pyplot as plt

import warnings

warnings.simplefilter('ignore')

import ultralytics
from ultralytics import YOLO

from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
%cd /content/drive/My Drive/Master/Autonmous Cars/Car Dataset/
```

```
/content/drive/My Drive/Master/Autonmous Cars/Car Dataset
```

```
!pwd
```

```
/content/drive/My Drive/Master/Autonmous Cars/Car Dataset
```

```
df = pd.read_csv('labels_train.csv')
df = shuffle(df)
df.head()
```

	frame	xmin	xmax	ymin	ymax	class_id
50821	1478897949260017032.jpg	82	113	130	161	2
79753	1479500555115367926.jpg	240	270	111	158	2
33675	1478896545030322435.jpg	230	245	148	164	1
39916	1478897104895202219.jpg	224	250	148	168	1
90490	1479502370215844219.jpg	182	233	133	178	1

```
classes = df.class_id.unique()
print(classes)
```

```
[2 1 3 5 4]
```

```
labels = { 1:'car', 2:'truck', 3:'person', 4:'bicycle', 5:'traffic
light'}
```

```
from ultralytics import YOLO
import PIL
from PIL import Image
from IPython.display import display
```

```
import os
import pathlib

model = YOLO("yolov8m.pt")

%cd /content/drive/My Drive/Master/Autonmous Cars/Car Dataset/images/
```

```
/content/drive/My Drive/Master/Autonmous Cars/Car Dataset/images
```

```
!pwd
```

```
/content/drive/MyDrive/Master/Autonmous Cars/Car Dataset/images
```

```
results=model.predict(source="14780199526863110006.jpg",
              save=True, conf=0.2,iou=0.5)
```

```
image 1/1 /content/drive/MyDrive/Master/Autonmous Cars/Car
Dataset/images/14780199526863110006.jpg: 416x640 1 person, 2 cars,
1385.2ms
Speed: 26.4ms preprocess, 1385.2ms inference, 35.3ms postprocess per
image at shape (1, 3, 416, 640)
Results saved to runs/detect/predict
```

```
result = results[0]
box = result.boxes[0]

for result in results:
    boxes = result.boxes  # Boxes object for bbox outputs
    masks = result.masks  # Masks object for segmentation masks
outputs
    probs = result.probs  # Class probabilities for classification
outputs

cords = box.xyxy[0].tolist()
class_id = box.cls[0].item()
conf = box.conf[0].item()
print("Object type:", class_id)
print("Coordinates:", cords)
print("Probability:", conf)
```

```
Object type: 0.0
Coordinates: [433.603271484375, 121.6790542602539, 454.5157470703125,
181.96151733398438]
Probability: 0.7077028155326843
```

```
for box in result.boxes:
    class_id = result.names[box.cls[0].item()]
    cords = box.xyxy[0].tolist()
    cords = [round(x) for x in cords]
    conf = round(box.conf[0].item(), 2)
    print("Object type:", class_id)
```

```
    print("Coordinates:", cords)
    print("Probability:", conf)
    print("---")
```

```
Object type: person
Coordinates: [434, 122, 455, 182]
Probability: 0.71
---
Object type: car
Coordinates: [240, 145, 251, 154]
Probability: 0.38
---
Object type: car
Coordinates: [215, 145, 223, 153]
Probability: 0.21
---
```

```
results1 = model.predict(source="1478020211690815798.jpg",
              save=True, conf=0.2,iou=0.5)

Results = results1[0]
```

```
image 1/1 /content/drive/MyDrive/Master/Autonmous Cars/Car
Dataset/images/1478020211690815798.jpg: 416x640 4 cars, 3 traffic
lights, 1401.6ms
Speed: 3.6ms preprocess, 1401.6ms inference, 1.4ms postprocess per
image at shape (1, 3, 416, 640)
Results saved to runs/detect/predict
```

```
# Plotting results
plot = results1[0].plot()
plot = cv2.cvtColor(plot, cv2.COLOR_BGR2RGB)
display(Image.fromarray(plot))
```


traffic light 0.70
traffic light 0.39
car 0.90
car 0.81
car 0.84