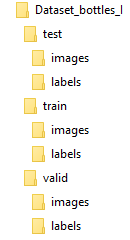
**Instructions**

1. Creating the Dataset

* **Image Gathering:** Large quantities of images with different resolutions must be procured. The image quality must be curated to ensure optimal training results. For every class, there should be a proportional number of images to ensure representation is not overrepresented. The images taken must feature real-life conditions to ensure a good performance model when deployed for real-time object Detection tasks. The images can be in various formats (e.g., PNG, JPEG, etc.).
* **Structuring:** The dataset must have a 'train' and 'validation' folder (train and val) in order to execute the training process. A separate test folder is optional.



1. Labelling

* Each image must have a .txt label file in the YOLO format. The standard YOLO format is compatible with any version (e.g., YOLOv5, YOLOv8, YOLOv11). The most comprehensive technique for labeling is the bounding box, but other methods exist, such as segmentation masks or key points.
* The .txt files must be in the same folder as the images.



1. Training

* Install YOLOv8 with the following steps:

**Windows**

* 1. Download Python from https://www.python.org/downloads/
  2. During installation, check "Add Python to PATH".
  3. Verify installation
     1. Python –version
  4. Create a virtual environment (optional)
     1. python -m venv yolov8env
     2. yolov8env\Scripts\activate
  5. Upgrade pip
     1. Python -m pip install-- upgrade pip
  6. Install YOLOv8
     1. pip install ultralytics
  7. Verify Installation
     1. yolo help
  8. Run example detection
     1. yolo task=detect mode=predict model=yolov8n.pt source='https://ultralytics.com/images/bus.jpg'

**Linux**

3.1 Install Python & pip

3.1.1 python3 –version

If not installed:

* + 1. sudo apt update
    2. sudo apt install python3 python3-pip

3.2 Create virtual environment (optional)

3.2.1 python3 -m venv yolov8env

3.2.2 source yolov8env/bin/activate

3.3 Upgrade pip

* + 1. python3 -m pip install --upgrade pip
  1. Install YOLOv8

3.4.1 pip install ultralytics

* 1. Verify Installation
     1. yolo help
  2. Run example detection
     1. yolo task=detect mode=predict model=yolov8n.pt source='https://ultralytics.com/images/bus.jpg'

1. Training execution

* Where YOLOv8 is saved, the training can be executed with the following command:

Yolo detect train model=yolov8s.pt data=data.yaml epochs=300 imgsz=640 batch=16 name= name\_file

* The training command has the following parameters

|  |  |
| --- | --- |
| *Parameter* | *Description* |
| *model* | *The base* |
| *data* | *Path to data.yaml file* |
| *epochs* | *Number of training iterations (epochs)* |
| *imgsz* | *Input image resolution (e.g., 640 for 640x640)* |
| *batch* | *Batch size per training step* |
| *name* | *Name for the training run* |

* The parameters can be adjusted depending on the testing situation

1. Training validation

* While the training results can be interpreted from the metrics found in the previously created folder, it is possible to validate them using the following command and use a detection test to estimate the detection rate and accuracy:

yolo detect val model=C:/yolov5/Dataset/weights/best.pt data= C:/yolov5/Dataset/data.yaml conf=0.80

yolo detect predict model= C:/yolov5/Dataset/weights/best.pt source= C:/yolov5/Dataset/test/images conf=0.80 save\_txt save\_conf

YOLOv8 Validation Command Parameters

|  |  |
| --- | --- |
| *Parameter* | *Description* |
| *model* | *Path to the trained YOLOv8 model* |
| *data* | *Path to data.yaml containing test/val dataset and class info* |
| *conf* | *Confidence threshold for predictions* |
| *save\_txt* | *Saves predictions in YOLO text format* |
| *save\_conf* | *Saves the confidence score with each prediction* |
| *save\_hybrid* | *Combines predictions and ground truth for easier analysis* |

YOLOv8 Detection Command Parameters

|  |  |
| --- | --- |
| *Parameter* | *Description* |
| *model* | *Path to the trained YOLOv8 model* |
| *source* | *Path to input images or video for detection* |
| *conf* | *Confidence threshold to filter out low-confidence detections* |
| *save\_txt* | *Saves detection outputs in YOLO text format* |
| *save\_conf* | *Appends confidence scores to detection outputs* |
| *save\_crop* | *Saves cropped images of detected objects* |
| *project* | *Custom output directory for results* |
| *name* | *Name inside the project.* |

True Positives (TP):

* TP = Recall × Number of Instances

False Positives (FP):

* FP = (TP / Precision) – TP

False Negatives (FN):

* FN = (TP / Recall) – TP

Where:

- TP: True Positives (correct detections)

- FP: False Positives (incorrect detections)

- FN: False Negatives (missed detections)

- Precision and Recall are derived from the YOLOv8 validation output.

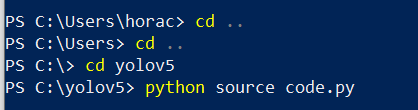
- Instances refer to the number of labeled objects for that class in the validation set.

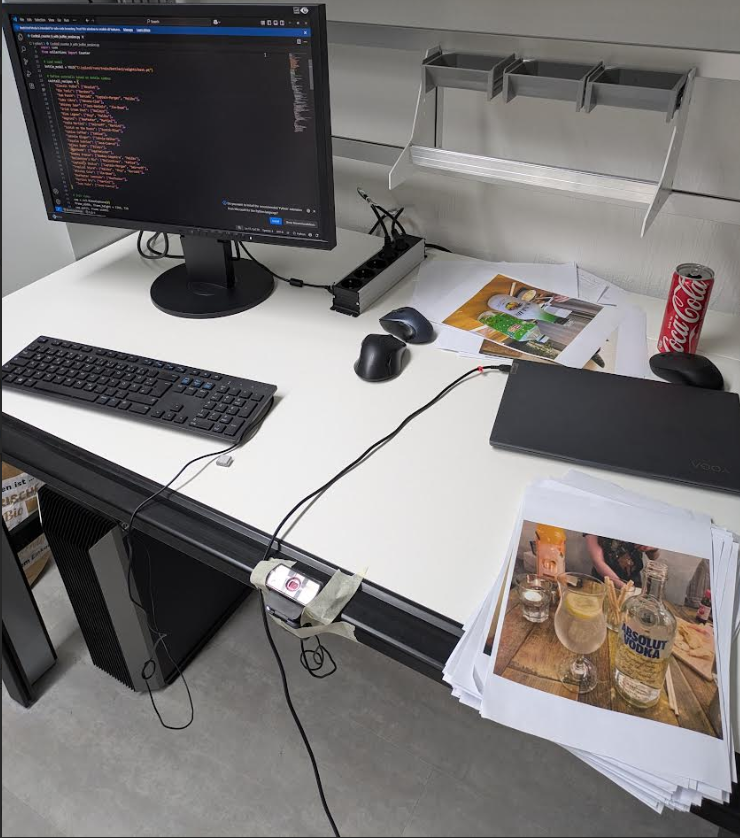
6) Testing

6.1 The weights best.pt results are loaded on the source code, around line 11. The full path can be inserted.

bottle\_model = YOLO("C:/yolov5/runs/train/Bottles2/weights/best.pt")

6.2 Execute the script on Windows PowerShell in the saved folder. The script will only run with a connected USB camera. The source code is available for download from https://github.com/hbo-bomb/Cocktail-Log-BI\_II





7) Exporting the results

To format the weights best.pt results in the following command being executed:

## yolo export model=C:/yolov5/Dataset/weights/best.pt format=onnx dynamic=True simplify=True opset=12 half=True

|  |  |
| --- | --- |
| *Parameter* | *Description* |
| *model* | *Path to the trained YOLOv8 model file* |
| *format* | *Export format for deployment* |
| *dynamic* | *Exports a dynamic ONNX model for variable input sizes (optional)* |
| *simplify* | *Simplifies the ONNX model to optimize the graph structure (optional)* |
| *opset* | *Specifies the ONNX opset version (optional, default is 12)* |
| *half* | *Converts model weights to FP16 (half precision) where supported (optional)* |

8) Working on Linux

The execution of the source code is the same as with the Windows environment, but using the best.onnx results are from the conversion in step 7. A USB camera is also required to run this version. The source code is the same as the one in step 6