roboflow /notebooks

How to Train YOLOv8 Object Detection on a Custom Dataset

Roboflow Blog | Youtube | GitHub

Ultralytics YOLOv8 is the latest version of the YOLO (You Only Look Once) object detection and image segmentation model developed by Ultralytics. The YOLOv8 model is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and image segmentation tasks. It can be trained on large datasets and is capable of running on a variety of hardware platforms, from CPUs to GPUs.

⚠️Disclaimer

YOLOv8 is still under heavy development. Breaking changes are being introduced almost weekly. We strive to make our YOLOv8 notebooks work with the latest version of the library. Last tests took place on **03.01.2024** with version **YOLOv8.0.196**.

If you notice that our notebook behaves incorrectly - especially if you experience errors that prevent you from going through the tutorial - don't hesitate! Let us know and open an issue on the Roboflow Notebooks repository.

Accompanying Blog Post

We recommend that you follow along in this notebook while reading the blog post on how to train YOLOv8 Object Detection, concurrently.

Pro Tip: Use GPU Acceleration

If you are running this notebook in Google Colab, navigate to `Edit` -> `Notebook settings` -> `Hardware accelerator`, set it to `GPU`, and then click `Save`. This will ensure your notebook uses a GPU, which will significantly speed up model training times.

Steps in this Tutorial

Steps in this tutorial

In this tutorial, we are going to cover:

- Before you start
- Install YOLOv8
- CLI Basics
- Inference with Pre-trained COCO Model
- Roboflow Universe
- Preparing a custom dataset
- Custom Training
- Validate Custom Model
- Inference with Custom Model

Let's begin!

Before you start

Let's make sure that we have access to GPU. We can use `nvidia-smi` command to do that. In case of any problems navigate to `Edit` -> `Notebook settings` -> `Hardware accelerator`, set it to `GPU`, and then click `Save`.

```
!nvidia-smi
```

```
Wed May 15 21:08:31 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version
|-----------------------------------------+----------------------+----------
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile U
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util
|                                         |                      |
|=========================================+======================+==========
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |
| N/A   75C    P8              12W /  70W |      0MiB / 15360MiB |      0%
|                                         |                      |
+-----------------------------------------+----------------------+----------

+---------------------------------------------------------------------------------------+
| Processes:
|  GPU   GI   CI        PID   Type   Process name
|        ID   ID
|=======================================================================================
|  No running processes found
+---------------------------------------------------------------------------------------+
```

```
import os
HOME = os.getcwd()
```

```
HOME = os.getcwd()
print(HOME)
```

```
/content
```

Install YOLOv8

⚠️YOLOv8 is still under heavy development. Breaking changes are being introduced almost weekly. We strive to make our YOLOv8 notebooks work with the latest version of the library. Last tests took place on **03.01.2024** with version **YOLOv8.0.196**.

If you notice that our notebook behaves incorrectly - especially if you experience errors that prevent you from going through the tutorial - don't hesitate! Let us know and open an issue on the Roboflow Notebooks repository.

YOLOv8 can be installed in two ways-from the source and via pip. This is because it is the first iteration of YOLO to have an official package.

```
# Pip install method (recommended)

!pip install ultralytics==8.0.196

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 30.8/78.2 GB disk)
```

```
# Git clone method (for development)

# %cd {HOME}
# !git clone github.com/ultralytics/ultralytics
# %cd {HOME}/ultralytics
# !pip install -e .

# from IPython import display
# display.clear_output()

# import ultralytics
# ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image
```

CLI Basics

If you want to train, validate or run inference on models and don't need to make any modifications to the code, using YOLO command line interface is the easiest way to get started. Read more about CLI in Ultralytics YOLO Docs.

```
yolo task=detect    mode=train    model=yolov8n.yaml      args...
          classify       predict        yolov8n-cls.yaml  args...
          segment        val            yolov8n-seg.yaml  args...
                         export         yolov8n.pt        format=onnx  args...
```

Inference with Pre-trained COCO Model

💻CLI

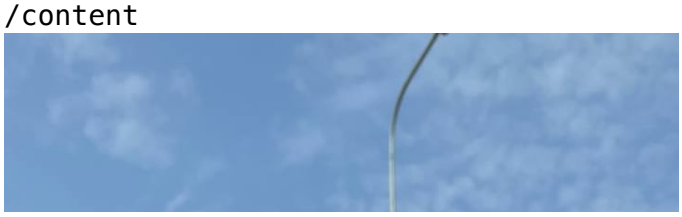
`yolo mode=predict` runs YOLOv8 inference on a variety of sources, downloading models automatically from the latest YOLOv8 release, and saving results to `runs/predict`.

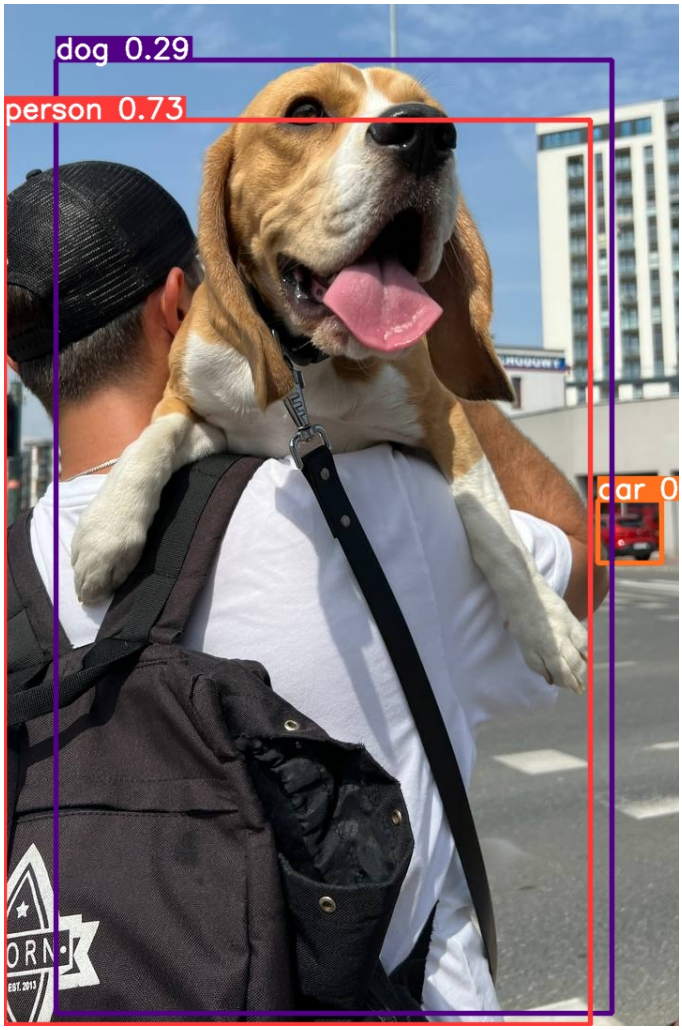
```
%cd {HOME}
!yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source='https://media.r
```

```
/content
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4
YOLOv8n summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFL

Found https://media.roboflow.com/notebooks/examples/dog.jpeg locally at dog.j
WARNING ⚠️ NMS time limit 0.550s exceeded
image 1/1 /content/dog.jpeg: 640x384 1 person, 1 car, 1 dog, 194.7ms
Speed: 8.3ms preprocess, 194.7ms inference, 1008.7ms postprocess per image at
Results saved to runs/detect/predict5
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

```
%cd {HOME}
Image(filename='runs/detect/predict/dog.jpeg', height=600)
```

```
/content
```





🐍Python SDK

The simplest way of simply using YOLOv8 directly in a Python environment.

```
model = YOLO(f'{HOME}/yolov8n.pt')
results = model.predict(source='https://media.roboflow.com/notebooks/examples/dog
```

```
Found https://media.roboflow.com/notebooks/examples/dog.jpeg locally at dog.j
WARNING ⚠️ NMS time limit 0.550s exceeded
image 1/1 /content/dog.jpeg: 640x384 1 person, 1 car, 1 dog, 206.8ms
Speed: 3.3ms preprocess, 206.8ms inference, 1099.7ms postprocess per image at
```

```
results[0].boxes.xyxy
```

```
tensor([[   0.0000,  314.4717,  625.0754, 1278.1946],
        [  55.1731,  250.0220,  648.1080, 1266.2720],
        [ 633.2291,  719.5391,  701.0538,  786.0336]], device='cuda:0')
```

```
results[0].boxes.conf
```

```
    tensor([0.7271, 0.2907, 0.2846], device='cuda:0')

results[0].boxes.cls
    tensor([ 0., 16.,  2.], device='cuda:0')
```

Roboflow Universe

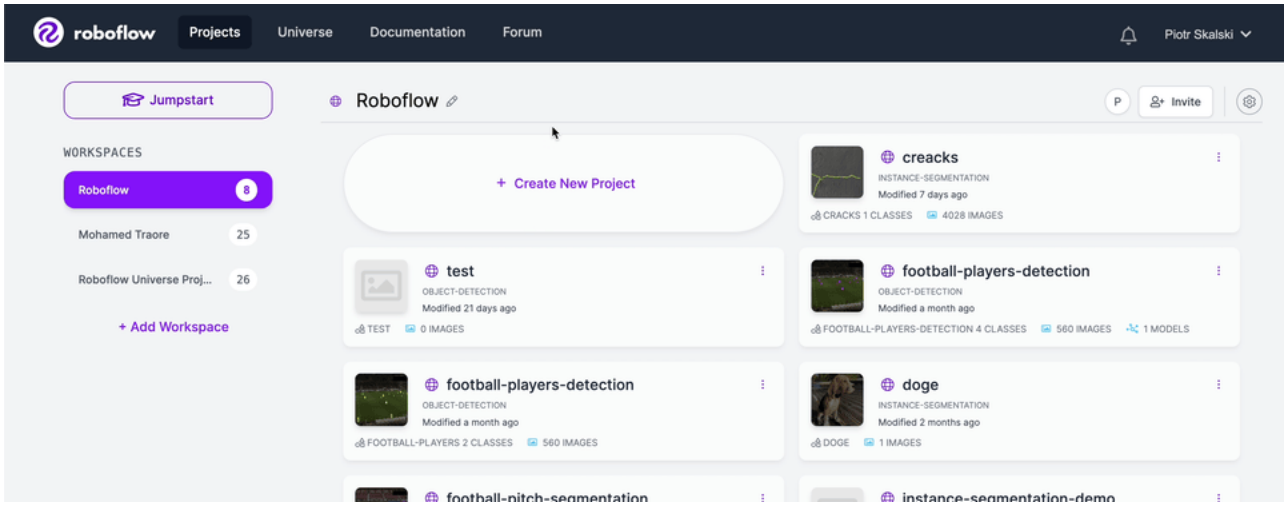
Need data for your project? Before spending time on annotating, check out Roboflow Universe, a repository of more than 110,000 open-source datasets that you can use in your projects. You'll find datasets containing everything from annotated cracks in concrete to plant images with disease annotations.

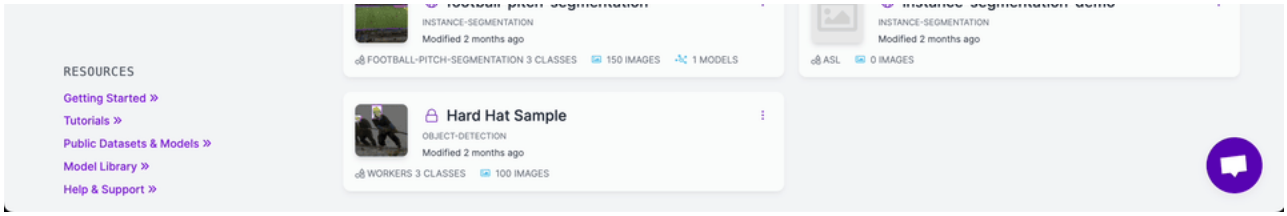
Preparing a custom dataset

Building a custom dataset can be a painful process. It might take dozens or even hundreds of hours to collect images, label them, and export them in the proper format. Fortunately, Roboflow makes this process as straightforward and fast as possible. Let me show you how!

Step 1: Creating project

Before you start, you need to create a Roboflow account. Once you do that, you can create a new project in the Roboflow dashboard. Keep in mind to choose the right project type. In our case, Object Detection.

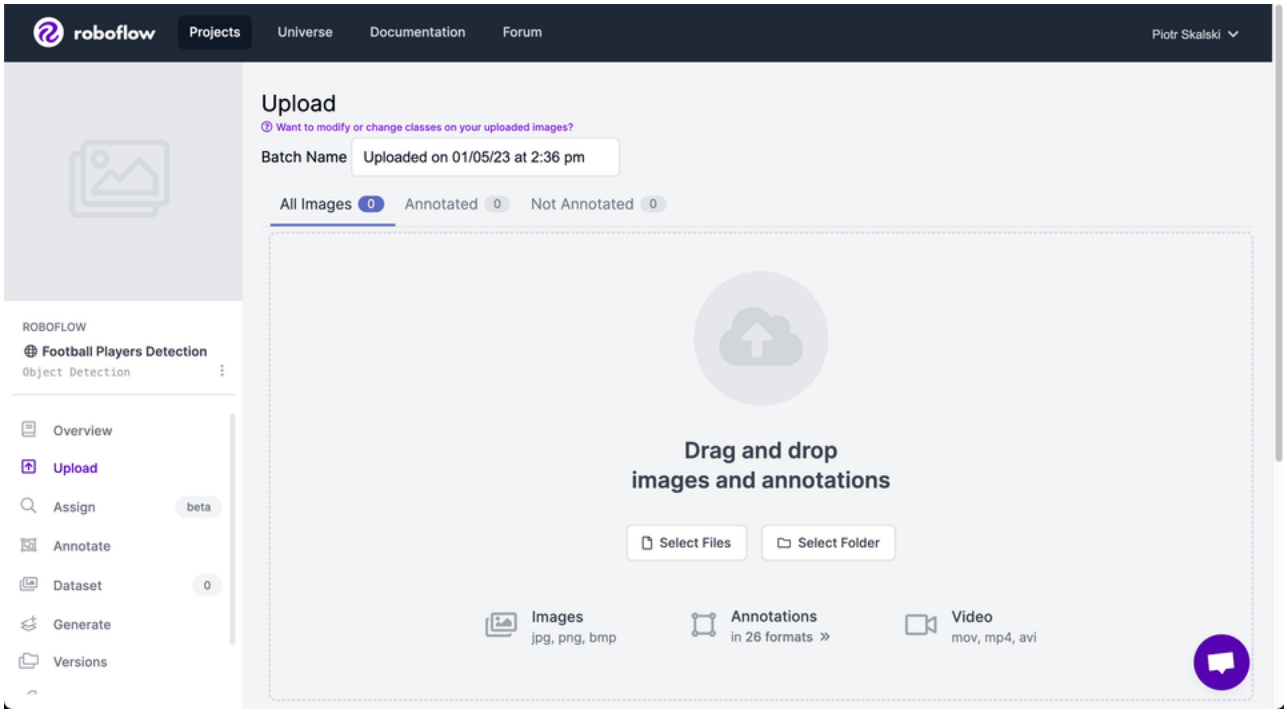




Step 2: Uploading images

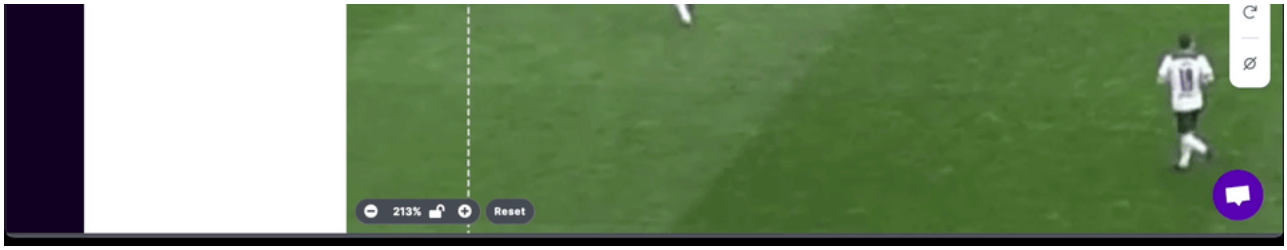
Next, add the data to your newly created project. You can do it via API or through our web interface.

If you drag and drop a directory with a dataset in a supported format, the Roboflow dashboard will automatically read the images and annotations together.



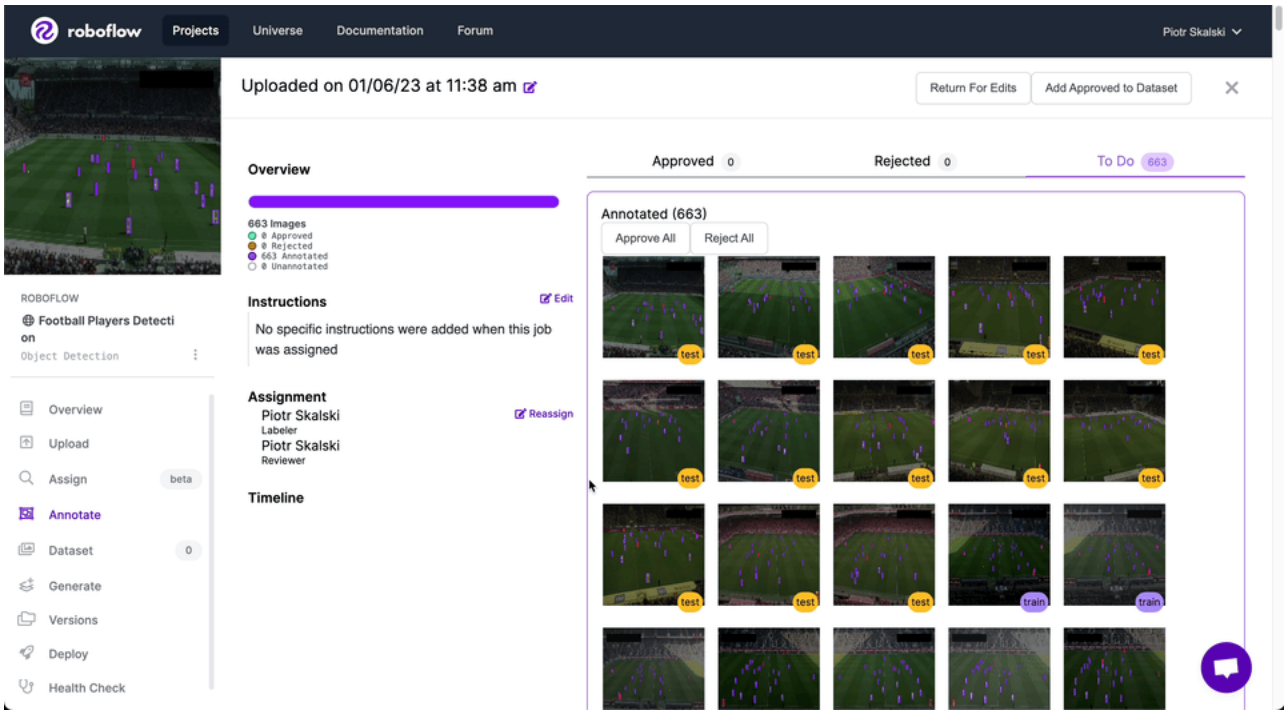
Step 3: Labeling

If you only have images, you can label them in Roboflow Annotate.



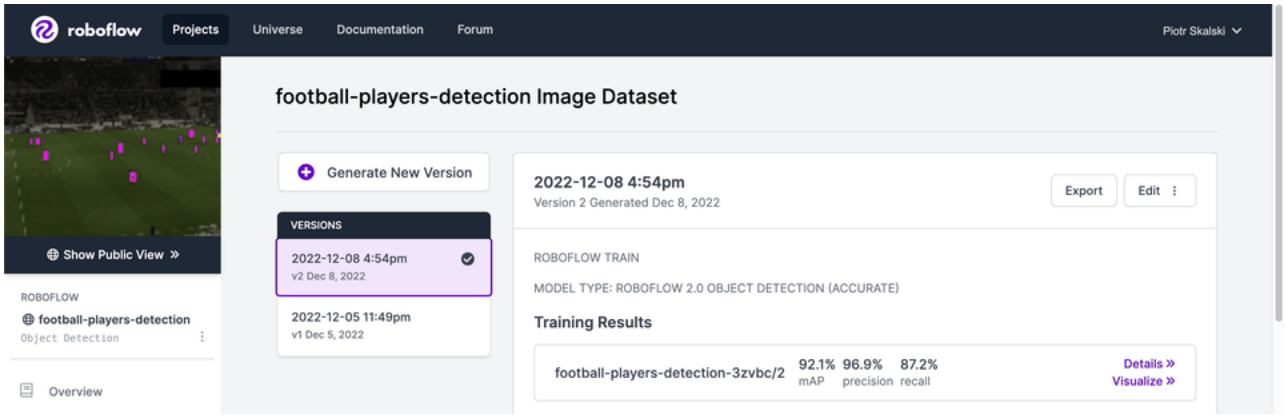
Step 4: Generate new dataset version

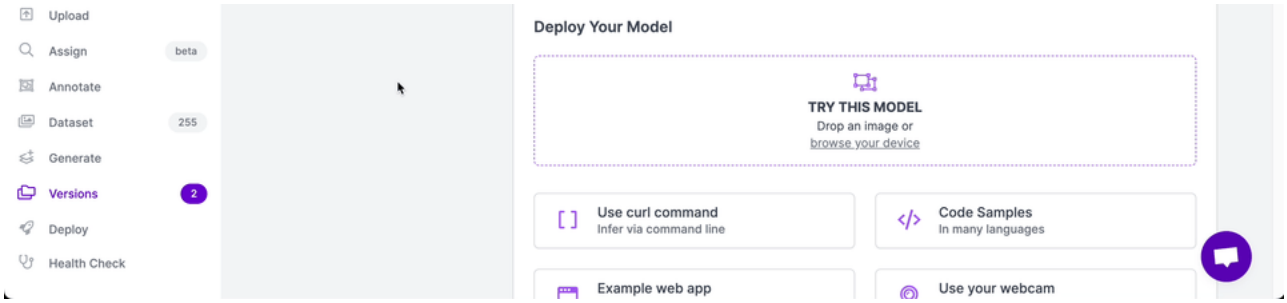
Now that we have our images and annotations added, we can Generate a Dataset Version. When Generating a Version, you may elect to add preprocessing and augmentations. This step is completely optional, however, it can allow you to significantly improve the robustness of your model.



Step 5: Exporting dataset

Once the dataset version is generated, we have a hosted dataset we can load directly into our notebook for easy training. Click `Export` and select the `YOLO v5 PyTorch` dataset format.






```
!mkdir {HOME}/datasets
%cd {HOME}/datasets

!pip install roboflow --quiet

from roboflow import Roboflow

rf = Roboflow(api_key="6n4vryXJRgiaPTgLHstB")
project = rf.workspace("personal-ubphn").project("klt_detection")
version = project.version(2)
dataset = version.download("yolov8")
```

```
mkdir: cannot create directory ‘/content/datasets’: File exists
/content/datasets
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in klt_detection-2 to yolov8:: 100%|
Extracting Dataset Version Zip to klt_detection-2 in yolov8:: 100%|
```

Custom Training

```
%cd {HOME}

!yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml e
```

```
/content
New https://pypi.org/project/ultralytics/8.2.16 available 😃 Update with 'pip
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=/content/data
2024-05-15 21:09:16.840500: E external/local_xla/xla/stream_executor/cuda/cud
2024-05-15 21:09:16.840567: E external/local_xla/xla/stream_executor/cuda/cud
2024-05-15 21:09:16.842583: E external/local_xla/xla/stream_executor/cuda/cud
Overriding model.yaml nc=80 with nc=4

                   from  n    params  module
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv
```

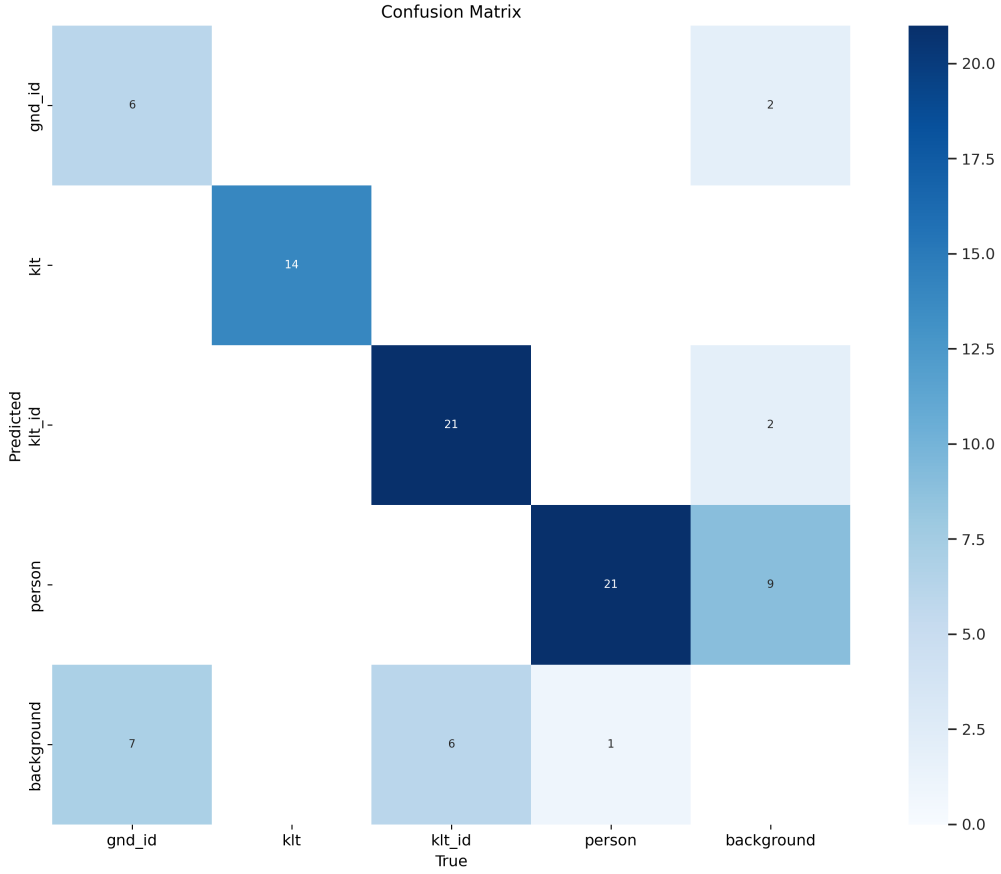
```
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f
 22        [15, 18, 21]  1    752092  ultralytics.nn.modules.head.Detect
Model summary: 225 layers, 3011628 parameters, 3011612 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train5', view at ht
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks with YOLOv8n...
WARNING ⚠️ NMS time limit 0.550s exceeded
AMP: checks passed ✅
WARNING ⚠️ updating to 'imgsz=640'. 'train' and 'val' imgsz must be an intege
train: Scanning /content/datasets/klt_detection-2/train/labels... 200 images,
train: New cache created: /content/datasets/klt_detection-2/train/labels.cach
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limi
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork
  self.pid = os.fork()
val: Scanning /content/datasets/klt_detection-2/valid/labels... 19 images, 2
val: New cache created: /content/datasets/klt_detection-2/valid/labels.cache
Plotting labels to runs/detect/train5/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' a
optimizer: AdamW(lr=0.00125, momentum=0.9) with parameter groups 57 weight(de
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train5
Starting training for 100 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
!ls {HOME}/runs/detect/train5/
```

```
args.yaml                                          P_curve.png          tra
confusion_matrix_normalized.png                    PR_curve.png         tra
confusion_matrix.png                               R_curve.png          tra
events.out.tfevents.1715807361.4109b3c2dbac.25026.0  results.csv        tra
F1_curve.png                                       results.png          val
labels_correlogram.jpg                             train_batch0.jpg     val
labels.jpg                                         train_batch1170.jpg  wei
```

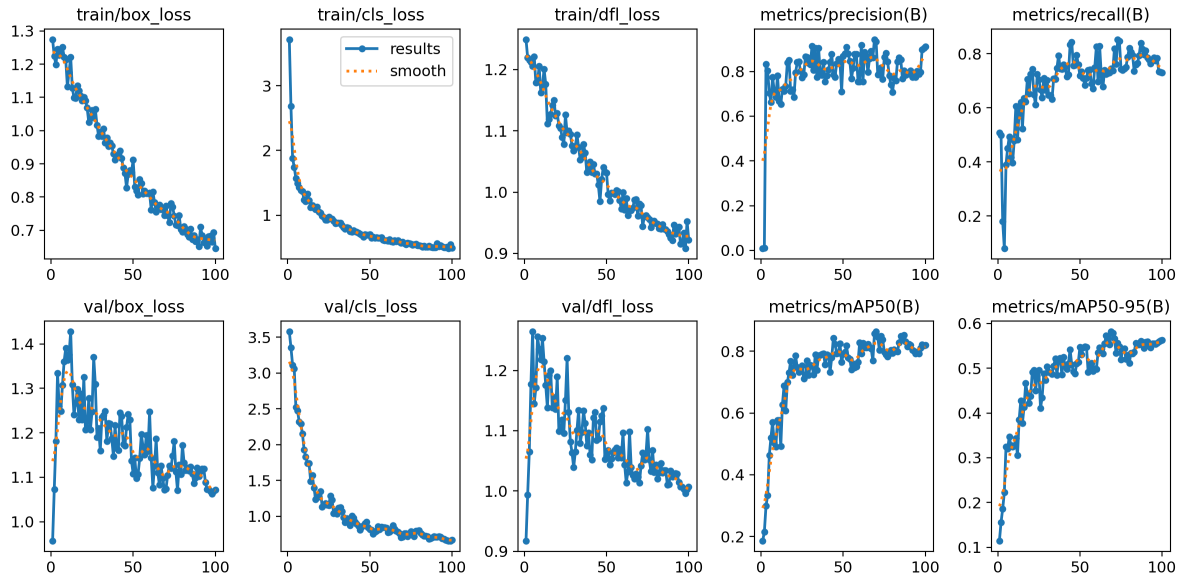
```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train5/confusion_matrix.png', width=600)
```

```
/content
```



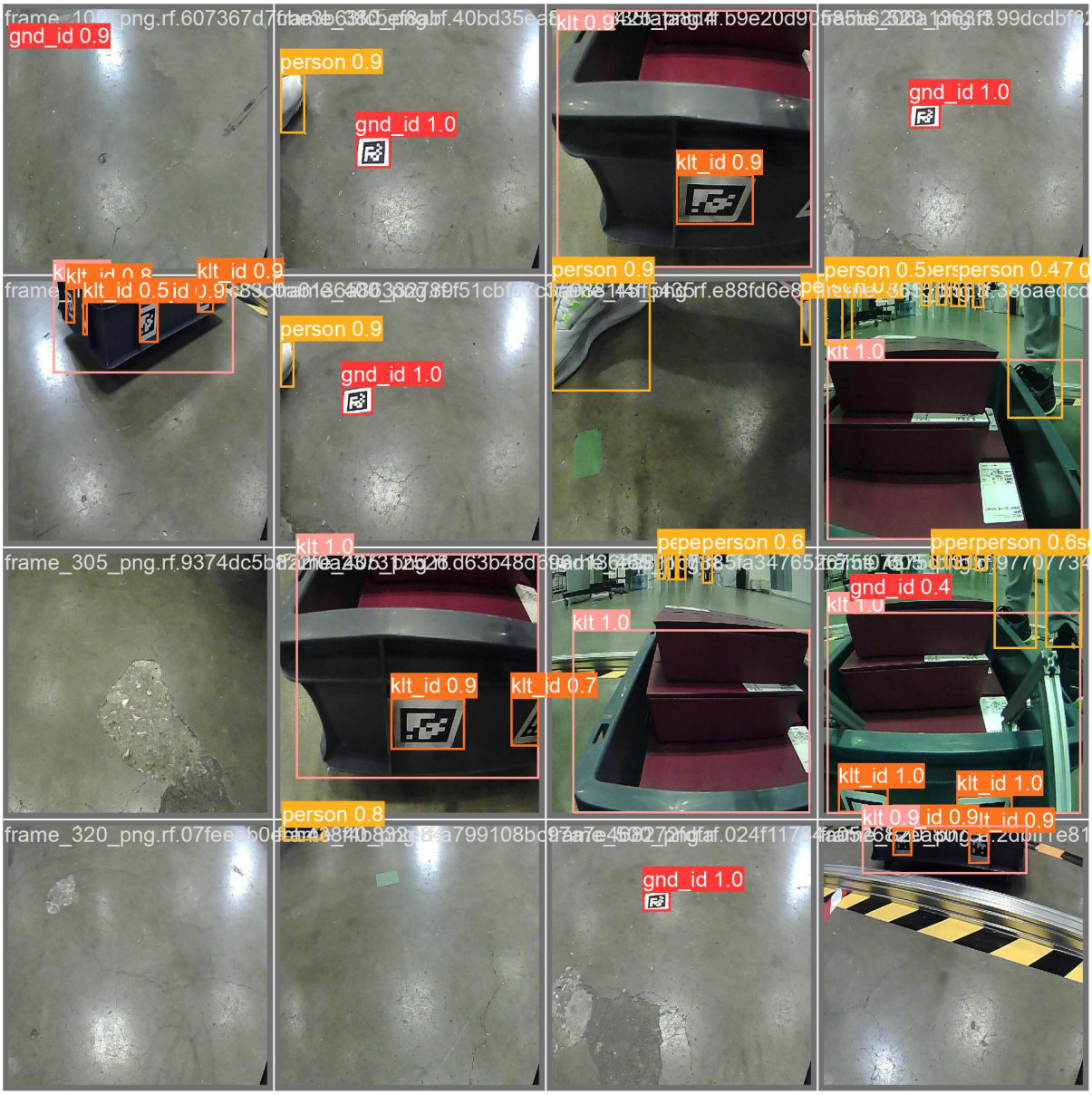

```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train5/results.png', width=600)
```

```
/content
```



```
%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train5/val_batch0_pred.jpg', width=600)
```

```
/content
```



Validate Custom Model

```
%cd {HOME}

!yolo task=detect mode=val model={HOME}/runs/detect/train5/weights/best.pt data={
```

```
/content
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4
Model summary (fused): 168 layers, 3006428 parameters, 0 gradients, 8.1 GFLOP
val: Scanning /content/datasets/klt_detection-2/valid/labels.cache... 19 imag
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork
  self.pid = os.fork()
                 Class     Images  Instances      Box(P          R      mAP50
                   all         19         76      0.947      0.746      0.846
```

```
                 all         19         76      0.947      0.748      0.846
             gnd_id         19         13          1      0.457      0.725
                klt         19         14          1          1      0.995
             klt_id         19         27      0.998      0.667      0.794
             person         19         22      0.791       0.86      0.872
Speed: 0.3ms preprocess, 22.3ms inference, 0.0ms loss, 44.2ms postprocess per
Results saved to runs/detect/val3
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Inference with Custom Model

```
%cd {HOME}
!yolo task=detect mode=predict model={HOME}/runs/detect/train4/weights/best.pt cc
```

```
/content
Ultralytics YOLOv8.0.196 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4
Model summary (fused): 168 layers, 3006038 parameters, 0 gradients, 8.1 GFLOP

image 1/10 /content/datasets/klt_detection-2/test/images/frame_1000_png.rf.5c
image 2/10 /content/datasets/klt_detection-2/test/images/frame_145_png.rf.876
image 3/10 /content/datasets/klt_detection-2/test/images/frame_185_png.rf.e95
image 4/10 /content/datasets/klt_detection-2/test/images/frame_285_png.rf.2c8
image 5/10 /content/datasets/klt_detection-2/test/images/frame_325_png.rf.aa7
image 6/10 /content/datasets/klt_detection-2/test/images/frame_440_png.rf.225
image 7/10 /content/datasets/klt_detection-2/test/images/frame_720_png.rf.9d5
image 8/10 /content/datasets/klt_detection-2/test/images/frame_740_png.rf.c41
image 9/10 /content/datasets/klt_detection-2/test/images/frame_780_png.rf.6fd
image 10/10 /content/datasets/klt_detection-2/test/images/frame_980_png.rf.c8
Speed: 1.6ms preprocess, 7.9ms inference, 59.9ms postprocess per image at sha
Results saved to runs/detect/predict6
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

NOTE: Let's take a look at few results.

```
import glob
from IPython.display import Image, display

for image_path in glob.glob(f'{HOME}/runs/detect/predict3/*.jpg')[:3]:
      display(Image(filename=image_path, width=600))
      print("\n")
```

```
!zip /content/runs runs
```

```
updating: runs/ (stored 0%)
```