```
# Cell 1: Install YOLOv8 and check GPU

!pip install -q ultralytics

import torch

print("PyTorch version:", torch.__version__)
print("CUDA available:", torch.cuda.is_available())
if torch.cuda.is_available():
    print("GPU name:", torch.cuda.get_device_name(0))
else:
    print("⚠️ No GPU detected (training will be slow)")
```

```
PyTorch version: 2.9.0+cu126
CUDA available: True
GPU name: Tesla T4
```

```
# Cell 2: Mount Drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
```

```
# Cell 3: Extract dataset ZIP from Drive

import zipfile, os

zip_path = "/content/drive/MyDrive/Colab Notebooks/archive.zip"  # change if needed
extract_to = "/content/data"

os.makedirs(extract_to, exist_ok=True)

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_to)

print("✅ Dataset extracted to:", extract_to)
```

```
✅ Dataset extracted to: /content/data
```

```
# Cell 4: Find data.yaml automatically and inspect

import os, yaml

yaml_path = None
for root, dirs, files in os.walk("/content/data"):
    if "data.yaml" in files:
        yaml_path = os.path.join(root, "data.yaml")
        break

if yaml_path is None:
```

```
    raise FileNotFoundError("❌ data.yaml not found inside /content/data")

print("✅ Found data.yaml at:", yaml_path)

with open(yaml_path, "r") as f:
    data_yaml = yaml.safe_load(f)

print("Number of classes:", len(data_yaml.get("names", [])))
print("First 10 classes:", data_yaml.get("names", [])[:10])
```

```
✅ Found data.yaml at: /content/data/data/data.yaml
Number of classes: 74
First 10 classes: ['Accès interdit aux motocycles', 'Sens giratoire obligatoire', 'Parc de st
```

```
# Cell 5: Fix data.yaml paths to absolute ones

import yaml

with open(yaml_path, "r") as f:
    d = yaml.safe_load(f)

d["train"] = "/content/data/data/train"
d["val"]   = "/content/data/data/val"
d["test"]  = "/content/data/data/test"

with open(yaml_path, "w") as f:
    yaml.dump(d, f)

print("✅ data.yaml updated:")
print("train:", d["train"])
print("val:  ", d["val"])
print("test: ", d["test"])
```

```
✅ data.yaml updated:
train: /content/data/data/train
val:   /content/data/data/val
test:  /content/data/data/test
```

```
# Cell 6: Import YOLO and set random seed (for more stable runs)

from ultralytics import YOLO
import random, numpy as np
import torch

seed = 42
random.seed(seed)
np.random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed_all(seed)

print("✅ Seed set to", seed)
```

```
✅ Seed set to 42
```

```
# Cell 7: Train a stronger model (yolov8l)

model = YOLO("yolov8l.pt")  # larger than m → usually better accuracy, slower but OK on T4

results = model.train(
    data=yaml_path,
    imgsz=640,                    # good size for traffic signs
    epochs=150,                   # more than before
    batch=-1,                     # auto batch size
    device=0,                     # first GPU
    project="runs_morocco_v2",
    name="yolov8l_moroccan_signs",
    pretrained=True,
    patience=30,                  # early stop if no improvement
    optimizer="SGD",              # default, stable
    lr0=0.01,                     # you can try 0.005 if it overfits
    weight_decay=0.0005,
    cos_lr=True,
    workers=2,
    plots=True,
)

print("✅ Training finished. Run dir:", results.save_dir)
```

```
Stationnement interdit         14         17      0.522      0.941      0.824      0.498
 Limitation de Vitesse         21         21          1      0.943      0.965      0.679
                  Stop          7          7      0.822          1       0.96      0.644
Passage d'animaux domestiques          1          1      0.422          1      0.995      0
Succession de virages dont le premier est à gauche          3          3          1      0.8
         Sens interdit          8          8      0.782      0.897      0.874      0.514
   Cassis ou dos d'âne          1          1          1          0      0.995      0.796
Arrêt et stationnement interdit          3          3      0.492          1      0.746
Obligation de tourner à gauche avant le panneau          5          5          1      0.395
Chaussée rétrécie par la gauche          1          1      0.434          1      0.995
Obligation de tourner à drote avant le panneau          4          4       0.33       0.75
        Autres dangers          2          3      0.876          1      0.995      0.574
Piste obligatoire pour cyclistes          1          1          1          0      0.995
Fin de toutes les interdictions précédemment signalées, imposées aux véhicules en mouvement
Fin d'interdiction de dépasser          1          1      0.709          1      0.995      0
Arrêt au barrage de gendarmerie          1          1          1          0          0
Chaussée rétrécie par la droite          2          2          0          0          0
Speed: 0.3ms preprocess, 16.4ms inference, 0.0ms loss, 4.8ms postprocess per image
Results saved to /content/runs_morocco_v2/yolov8l_moroccan_signs5
✅ Training finished. Run dir: /content/runs_morocco_v2/yolov8l_moroccan_signs5
```

```
# Cell 8: Locate best.pt and last.pt

from pathlib import Path

run_dir = Path(results.save_dir)
best_weights = run_dir / "weights" / "best.pt"
last_weights = run_dir / "weights" / "last.pt"

print("Run dir:", run_dir)
print("Best weights:", best_weights)
print("Last weights:", last_weights)
```

```
Run dir: /content/runs_morocco_v2/yolov8l_moroccan_signs5
Best weights: /content/runs_morocco_v2/yolov8l_moroccan_signs5/weights/best.pt
Last weights: /content/runs_morocco_v2/yolov8l_moroccan_signs5/weights/last.pt
```

```
# Cell 9: Save the trained model

best_weights = run_dir / "weights" / "best.pt"
last_weights = run_dir / "weights" / "last.pt"

print("Best weights:", best_weights.exists(), best_weights)
print("Last weights:", last_weights.exists(), last_weights)

# Optional: Export to ONNX or other formats if needed
# model.export(format="onnx", imgsz=512)
```

```
Best weights: True /content/runs_morocco_v2/yolov8l_moroccan_signs5/weights/best.pt
Last weights: True /content/runs_morocco_v2/yolov8l_moroccan_signs5/weights/last.pt
```

```
# Cell 10: Inference on one test image

# change this path to any image in your dataset or a new photo
test_image = "/content/data/data/test/images/bbc1cdfe4b3-bd59-47c1-8b24-35d65c030e45.jpg"

pred_results = best_model.predict(
```

```
    source=test_image,
    imgsz=640,
    conf=0.4,
    save=True,
)

print("✅ Prediction done. Saved to:", pred_results[0].save_dir)
```

```
image 1/1 /content/data/data/test/images/bbc1cdfe4b3-bd59-47c1-8b24-35d65c030e45.jpg: 640x640
Speed: 2.4ms preprocess, 63.1ms inference, 2.0ms postprocess per image at shape (1, 3, 640, 6
Results saved to /content/runs/detect/predict
✅ Prediction done. Saved to: /content/runs/detect/predict
```

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c
```

```
# Cell 11: Copy best.pt to Google Drive

import shutil
from pathlib import Path

src = best_weights
dst = Path("/content/drive/MyDrive/Colab Notebooks/moroccan_yolov8l_best.pt")

shutil.copy(src, dst)
print("✅ best.pt copied to:", dst)
```

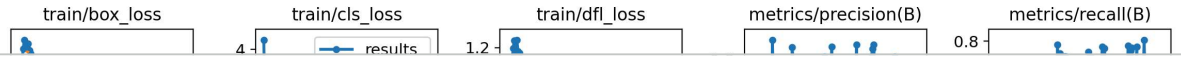
```
✅ best.pt copied to: /content/drive/MyDrive/Colab Notebooks/moroccan_yolov8l_best.pt
```

```
# Cell 12 (optional): Download model to your laptop

from google.colab import files

files.download(str(best_weights))
```

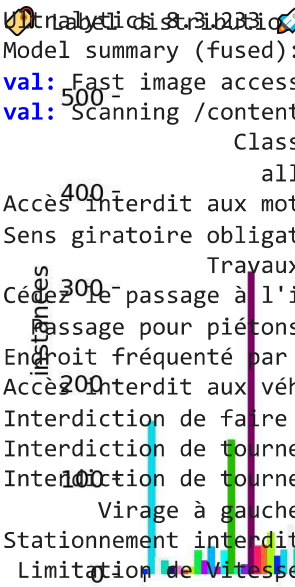
```
# Visualize training curves and label distribution

from pathlib import Path
from IPython.display import Image, display

run_dir = Path(results.save_dir)  # uses the results object from training

print("Run dir:", run_dir)

print("📈 Training curves (loss, mAP, P/R):")
display(Image(filename=str(run_dir / "results.png")))

print("🏷️ Label distribution:")
```

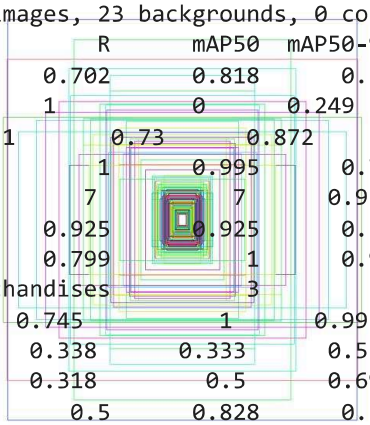
```
    display(Image(filename=str(run_dir / "labels.jpg")))
```

```
Run dir: /content/runs_morocco_v2/yolov8l_moroccan_signs5
📈 Training curves (loss, mAP, P/R):
```

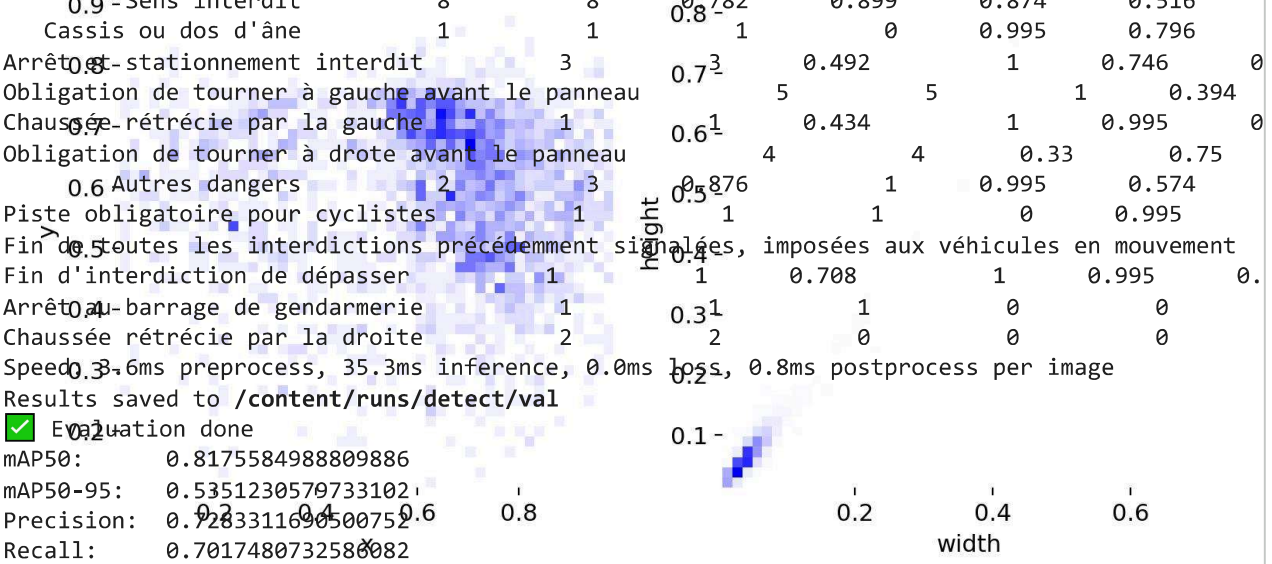



```
 # Evaluate best model on the validation set

from ultralytics import YOLO

best_weights = run_dir / "weights" / "best.pt"
best_model = YOLO(str(best_weights))

metrics = best_model.val()  # this will also save some plots in the run folder

print("✅ Evaluation done")
print("mAP50:     ", metrics.box.map50)
print("mAP50-95:  ", metrics.box.map)
print("Precision: ", metrics.box.mp)
print("Recall:    ", metrics.box.mr)
```





```
Ultralytics 8.3.233 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 112 layers, 43,663,662 parameters, 0 gradients, 165.1 GFLOPs
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 1576.9±603.8 MB/s, size: 103.6 KB)
val: Scanning /content/data/data/val/labels.cache... 106 images, 23 backgrounds, 0 corrupt: 1
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100
                   all        120        133      0.728      0.702      0.818      0.535
Accès interdit aux motocycles          1          1          1          0      0.249      0.1
Sens giratoire obligatoire             7          8          1      0.73      0.872      0.531
               Travaux                 1          1       0.43          1      0.995      0.796
Cédez le passage à l'intersection - signal de position         7          7      0.955
  Passage pour piétons                21         23       0.81      0.925      0.925      0.519
Endroit fréquenté par les enfants      2          2      0.799          1      0.995
Accès interdit aux véhicules affectés au transport de marchandises          3          3
Interdiction de faire demi-tour        1          1      0.745          1      0.995          0
Interdiction de tourner à droite       3          3      0.338      0.333      0.555
Interdiction de tourner à gauche       2          2      0.318        0.5      0.695
       Virage à gauche                 2          2       0.94        0.5      0.828      0.309
Stationnement interdit                14         17      0.522      0.941      0.824      0.513
 Limitation de vitesse                21         21          1      0.943      0.965      0.671
                 Stop                  7          7      0.822          1       0.96      0.644
Passage d'animaux domestiques          1          1      0.422          1      0.995      0.8
Succession de virages dont le premier est à gauche          3          3          1      0.87
          Sens interdit                8          8      0.782      0.899      0.874      0.516
   Cassis ou dos d'âne                 1          1          1          0      0.995      0.796
Arrêt et stationnement interdit        3          3      0.492          1      0.746          0
Obligation de tourner à gauche avant le panneau          5          5          1      0.394
Chaussée rétrécie par la gauche        1          1      0.434          1      0.995          0
Obligation de tourner à drote avant le panneau          4          4       0.33       0.75
       Autres dangers                  2          3      0.876          1      0.995      0.574
Piste obligatoire pour cyclistes       1          1          1          1          0      0.995
Fin de toutes les interdictions précédemment signalées, imposées aux véhicules en mouvement
Fin d'interdiction de dépasser         1          1      0.708          1      0.995      0.
Arrêt au barrage de gendarmerie        1          1          1          1          0          0
Chaussée rétrécie par la droite        2          2          1          0          0          0
Speed: 3.6ms preprocess, 35.3ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to /content/runs/detect/val
✅ Evaluation done
mAP50:      0.8175584988809886
mAP50-95:   0.5351230570733102
Precision:  0.7283311690500752
Recall:     0.7017480732586082
```



```
# Per-class AP (mAP50-95) for each class

import yaml
import pandas as pd

# yaml_path is already defined earlier in your notebook
with open(yaml_path, "r") as f:
    data_yaml = yaml.safe_load(f)

class_names = data_yaml.get("names", [])
maps = metrics.box.maps  # AP per class

rows = []
for i, name in enumerate(class_names):
    ap = float(maps[i]) if i < len(maps) else float("nan")
    rows.append({"id": i, "class": name, "AP_50_95": ap})

df_ap = pd.DataFrame(rows).sort_values("AP_50_95", ascending=False)

print("🔝 Best classes:")
display(df_ap.head(10))

print("🔻 Worst classes:")
display(df_ap.tail(10))
```

🔝 Best classes:

	id	class	AP_50_95
53	53	Piste obligatoire pour cyclistes	0.895500
29	29	Passage d'animaux domestiques	0.895500
40	40	Cassis ou dos d'âne	0.796000
64	64	Fin d'interdiction de dépasser	0.796000
8	8	Travaux	0.796000
27	27	Limitation de Vitesse	0.671298
17	17	Endroit fréquenté par les enfants	0.652375
28	28	Stop	0.644034
13	13	Cédez le passage à l'intersection - signal de ...	0.625868
18	18	Accès interdit aux véhicules affectés au trans...	0.616589

🔻 Worst classes:

	id	class	AP_50_95
26	26	Stationnement interdit	0.513062
21	21	Interdiction de tourner à droite	0.499500
50	50	Obligation de tourner à droite avant le panneau	0.479614