Colab setup & dataset download (Roboflow → YOLOv8)

```
from IPython.display import clear_output
clear_output()
import os, json, shutil, random, glob
from pathlib import Path
import torch, os, glob, random, yaml, numpy as np
from IPython.display import display


# --- Clean install for NumPy 2.x compatible stack ---
%pip -q install -U pip setuptools wheel

# Remove conflicting builds
%pip -q uninstall -y numpy ultralytics onnx onnxruntime onnxruntime-gpu roboflow opencv-pyth

# Install NumPy 2 and libs that support it
%pip -q install "numpy==2.0.2" \
                "opencv-python-headless>=4.10.0.84" \
                "ultralytics>=8.3.184" \
                "onnx>=1.16.0" \
                "onnxruntime-gpu>=1.18.0" \
                "roboflow>=1.2.6" \
                ipywidgets>=8.1

#import IPython; IPython.get_ipython().kernel.do_shutdown(True)  # will restart kernel
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 22.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 70.8 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages t
ipython 7.34.0 requires jedi>=0.16, which is not installed.
WARNING: Skipping ultralytics as it is not installed.
WARNING: Skipping onnx as it is not installed.
WARNING: Skipping onnxruntime as it is not installed.
WARNING: Skipping onnxruntime-gpu as it is not installed.
WARNING: Skipping roboflow as it is not installed.
```

```
import numpy, cv2, onnxruntime as ort, ultralytics, roboflow
print("numpy", numpy.__version__)
print("cv2", cv2.__version__)
print("onnxruntime providers:", ort.get_available_providers())
print("ultralytics", ultralytics.__version__)
import roboflow as rf; print("roboflow", rf.__version__)
```

```
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings
```

```
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir
numpy 2.0.2
cv2 4.12.0
onnxruntime providers: ['TensorrtExecutionProvider', 'CUDAExecutionProvider', 'CPUExecut
ultralytics 8.3.186
roboflow 1.2.6
```

Gpu info

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)
```

```
Mon Aug 25 17:53:16 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. EC
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M
|                                         |                        |               MIG M
|=========================================+========================+======================
|   0  NVIDIA A100-SXM4-40GB          Off |   00000000:00:04.0 Off |
| N/A   37C    P0             53W /  400W |       0MiB /  40960MiB |      0%      Defaul
|                                         |                        |             Disable
+-----------------------------------------+------------------------+----------------------

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

Roboflow: download the dataset you referenced (YOLOv8 format)

```
from roboflow import Roboflow
rf = Roboflow(api_key="ouWxulB15Sf3lsOjdSQD")
project = rf.workspace("roboflow-universe-projects").project("license-plate-recognition-rxg4
version = project.version(11)
dataset = version.download("yolov8")
data_dir = Path(dataset.location)
data_dir
```

```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in License-Plate-Recognition-11 to yolov8:: 100%|
Extracting Dataset Version Zip to License-Plate-Recognition-11 in yolov8:: 100%|
PosixPath('/content/License-Plate-Recognition-11')
```

Inspect and show the dataset YAML path (Ultralytics uses this)

```
data_dir = Path("/content/License-Plate-Recognition-11")
yaml_path = next(data_dir.glob("*.yaml"))
print("Dataset YAML ->", yaml_path)
display(yaml.safe_load(open(yaml_path)))
```

```
Dataset YAML -> /content/License-Plate-Recognition-11/data.yaml
{'names': ['License_Plate'],
 'nc': 1,
 'roboflow': {'license': 'CC BY 4.0',
  'project': 'license-plate-recognition-rxg4e',
  'url': 'https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11',
  'version': 11,
  'workspace': 'roboflow-universe-projects'},
 'test': '../test/images',
 'train': '../train/images',
 'val': '../valid/images'}
```

Dați dublu clic (sau apăsați pe Enter) pentru a edita

```
from ultralytics import YOLO
import torch

%pip -q install -U wandb
import os, wandb
os.environ.pop("WANDB_DISABLED", None)       # ensure not disabled
os.environ.pop("WANDB_MODE", None)           # ensure not offline/dryrun
wandb.login(relogin=True)                    # will prompt or reuse your key

from ultralytics import settings as yolo_settings
yolo_settings.update({'wandb': True})


MODEL_VARIANT = "n"            # try 's' after this works (more accurate, slower)
IMGSZ = 416
EPOCHS = 50
BATCH = 16 if torch.cuda.is_available() else 8

model = YOLO(f"yolov8{MODEL_VARIANT}.pt")
results = model.train(
```

```
        data=str(yaml_path),
        imgsz=IMGSZ,
        epochs=EPOCHS,
        batch=BATCH,
        patience=20,
        device=0 if torch.cuda.is_available() else "cpu",
        optimizer="SGD",
        lr0=0.01,
        weight_decay=5e-4,
        hsv_h=0.015, hsv_s=0.7, hsv_v=0.4,
        mosaic=1.0
    )

    # Optional: print the URL of the W&B run immediately
    if wandb.run:
        print("W&B run:", wandb.run.get_url())
```

```
invalid escape sequence '\/'
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ·····
wandb: WARNING If you're specifying your api key in code, ensure this code is not shar
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wa
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: trinc-ciprian (trinc-ciprian-universitatea-de-vest-din-
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt
Ultralytics 8.3.186 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (NVIDIA A100-SXM4-40GB,
engine/trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment,
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Ari
Tracking run with wandb version 0.21.1
Run data is saved locally in /content/wandb/run-20250825_175418-qjblmo46
Syncing run train to Weights & Biases (docs)
View project at https://wandb.ai/trinc-ciprian-universitatea-de-vest-din-timisoara/Ultralytics
View run at https://wandb.ai/trinc-ciprian-universitatea-de-vest-din-timisoara/Ultralytics/runs/qjblmo46
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arg
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3,
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [12
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [12
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [25
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [25
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [No
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [38
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [No
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [19
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [19
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [12
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [38
 22        [15, 18, 21]  1    751507  ultralytics.nn.modules.head.Detect           [1,
Model summary: 129 layers, 3,011,043 parameters, 3,011,027 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11n.pt
AMP: checks passed ✅
train: Fast image access ✅ (ping: 0.0±0.0 ms, read: 753.8±135.4 MB/s, size: 24.9 KB)
train: Scanning /content/License-Plate-Recognition-11/train/labels... 7057 image
train: New cache created: /content/License-Plate-Recognition-11/train/labels.ca
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 374.0±155.0 MB/s, size: 20.8 KB)
```

```
val: Scanning /content/License-Plate-Recognition-11/valid/labels... 2048 images, 3 bac
val: New cache created: /content/License-Plate-Recognition-11/valid/labels.cache
Plotting labels to runs/detect/train/labels.jpg...
optimizer: SGD(lr=0.01, momentum=0.937) with parameter groups 57 weight(decay=0.0), 64
Image sizes 416 train, 416 val
Using 8 dataloader workers
Logging results to runs/detect/train
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50      0.93G      1.289      1.371      1.222          3       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.936      0.888      0.917      0.5

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50      1.14G      1.264     0.8446      1.183          2       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.936      0.886      0.906      0.5

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/50      1.15G      1.268     0.8345      1.188          1       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.892      0.839      0.856      0.5

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/50      1.17G      1.275     0.7852      1.203          2       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.951      0.875      0.922      0.5

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/50      1.18G       1.26     0.7278      1.204          1       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.965      0.875      0.928      0.6

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/50      1.19G      1.228     0.6857      1.179          2       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.958      0.891      0.935       0.

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/50       1.2G      1.212     0.6556      1.169          1       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.962      0.893      0.928      0.6

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/50      1.21G      1.201      0.641      1.172          1       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.959      0.879      0.918      0.6

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/50      1.22G      1.184     0.6237      1.156          1       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.964      0.903      0.941       0.6

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/50      1.23G      1.173     0.6037       1.15          2       416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
```

```
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.964      0.902      0.942      0.6
```

Validate + quick visual *checks*

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/50      1.25G      1.169     0.5923      1.148          2        416: 100% ━━
```

```
# 1) Enable W&B in Ultralytics
from ultralytics import settings as yolo_settings
yolo_settings.update({'wandb': True})

# 2) (optional) login once per session
import wandb; wandb.login(relogin=True)

# 3) Log a fresh run by validating your best checkpoint
from ultralytics import YOLO
best_pt = "runs/detect/train/weights/best.pt"  # adjust if different
model = YOLO(best_pt)
metrics = model.val(imgsz=640, name="val_wandb_run")  # creates a W&B run
print(metrics.results_dict)

from IPython.display import Image, display
import os

run_dir = "runs/detect/train2"  # adjust if needed
for f in ["results.png", "confusion_matrix.png", "PR_curve.png", "F1_curve.png"]:
    path = os.path.join(run_dir, f)
    if os.path.exists(path):
        display(Image(filename=path, width=800))


# metrics (mAP, precision, recall, etc.)
metrics = model.val(data=str(yaml_path), imgsz=IMGSZ)
print(metrics.results_dict)

# visualize predictions on a handful of val images
val_imgs = glob.glob(f"{data_dir}/valid/images/*")
sample = random.sample(val_imgs, k=min(8, len(val_imgs)))
preds = model.predict(sample, imgsz=IMGSZ, conf=0.25, save=True, project="pred_samples", nam
print("Saved examples under: pred_samples/run")
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/50      1.34G      1.117     0.5366      1.121          3        416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.975      0.921      0.955       0.

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/50      1.35G      1.113     0.5274      1.111          2        416: 100% ━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                   all       2048       2195      0.981      0.917      0.956       0.

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      21/50      1.36G      1.094     0.5189      1.109          0        416:
                 Class     Images  Instances      Box(P          R      mAP50  m
                   all       2048       2195      0.976      0.922      0.958       0.
```

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.m
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ·····
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running  wandb
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Ultralytics 8.3.186 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (NVIDIA A100-SXM4-40GB,
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 700.5±181.7 MB/s, size: 19.8 KB)
val: Scanning /content/License-Plate-Recognition-11/valid/labels.cache... 2048 images, 3
Speed: 0.1ms preprocess, 1.0ms inference, 0.0ms loss, 0.9ms postprocess per image
Results saved to runs/detect/val_wandb_run
{'metrics/precision(B)': 0.9660934496390103, 'metrics/recall(B)': 0.9346166791420323, 'm
Ultralytics 8.3.186 🚀 Python-3.12.11 torch-2.8.0+cu126 CUDA:0 (NVIDIA A100-SXM4-40GB,
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 526.3±157.6 MB/s, size: 29.2 KB)
val: Scanning /content/License-Plate-Recognition-11/valid/labels.cache... 2048 images, 3
Speed: 0.1ms preprocess, 1.0ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/val
{'metrics/precision(B)': 0.9809284331033285, 'metrics/recall(B)': 0.9384965831435079, 'me
0: 416x416 1 License_Plate, 7.9ms
1: 416x416 1 License_Plate, 7.9ms
2: 416x416 1 License_Plate, 7.9ms
3: 416x416 1 License_Plate, 7.9ms
4: 416x416 1 License_Plate, 7.9ms
5: 416x416 1 License_Plate, 7.9ms
6: 416x416 1 License_Plate, 7.9ms
7: 416x416 1 License_Plate, 7.9ms
Speed: 1.1ms preprocess, 7.9ms inference, 1.1ms postprocess per image at shape (1, 3, 41
Results saved to pred_samples/run
Saved examples under: pred_samples/run
```

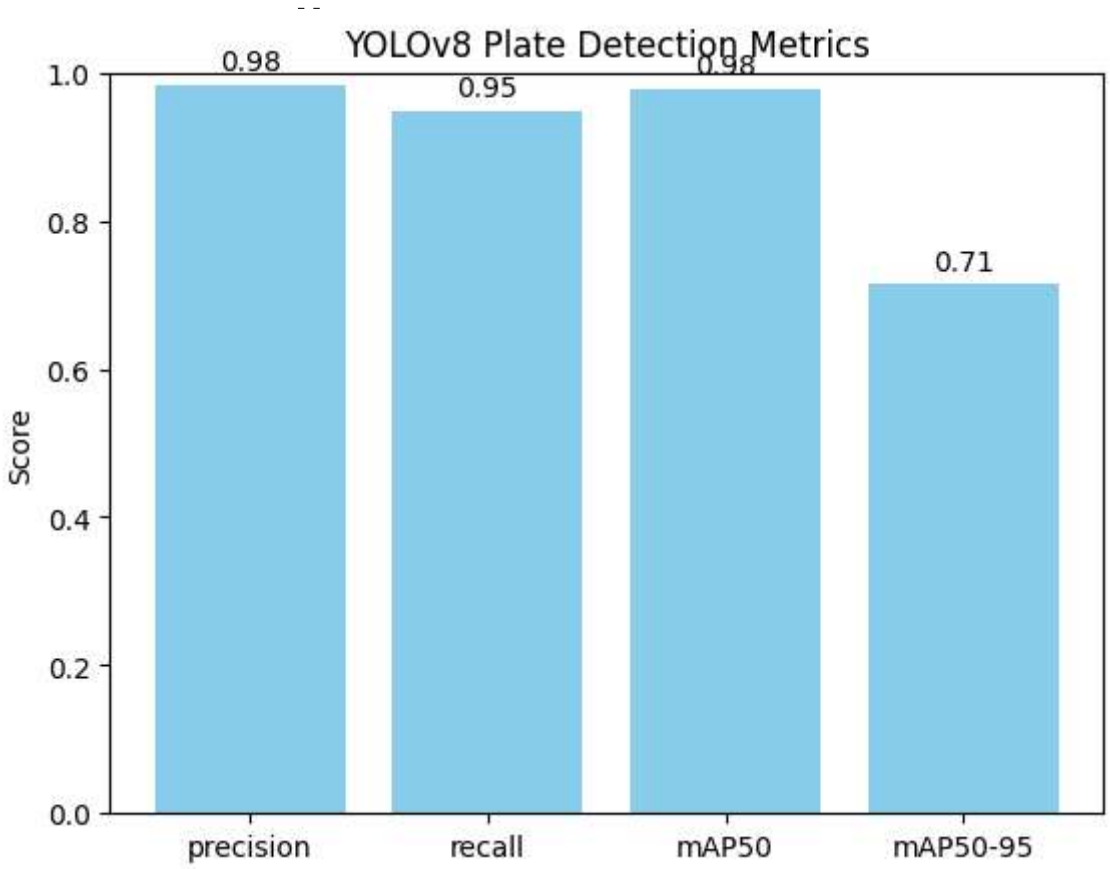
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
22/50	1.37G	1.098	0.5182	1.108	3	416: 100% ━━
23/50	1.38G	1.092	0.5095	1.103	2	416: 100% ━━
24/50	1.39G	1.088	0.5082	1.106	4	416: 100% ━━
25/50	1.41G	1.084	0.5036	1.099	4	416: 100% ━━
26/50	1.42G	1.078	0.497	1.101	4	416: 100% ━━
27/50	1.43G	1.072	0.4896	1.09	4	416: 100% ━━
28/50	1.44G	1.076	0.4931	1.092	2	416: 100% ━━
29/50	1.45G	1.067	0.4874	1.085	2	416: 100% ━━
30/50	1.46G	1.058	0.4848	1.08	2	416: 100% ━━

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	2048	2195	0.974	0.928	0.959	0.6
all	2048	2195	0.981	0.915	0.957	0.6
all	2048	2195	0.966	0.935	0.963	0.665
all	2048	2195	0.978	0.909	0.953	0.6
all	2048	2195	0.982	0.926	0.962	0.6
all	2048	2195	0.981	0.938	0.966	0.702
all	2048	2195	0.983	0.922	0.957	0.6
all	2048	2195	0.974	0.932	0.962	0.6
all	2048	2195	0.974	0.931	0.96	0.6
all	2048	2195	0.976	0.934	0.961	0.6

Plot metrics directly in Python

```
import matplotlib.pyplot as plt

metrics = {
 'precision': 0.983,
 'recall':    0.949,
 'mAP50':     0.977,
 'mAP50-95':  0.714
}

plt.bar(metrics.keys(), metrics.values(), color="skyblue")
plt.ylim(0,1)
```

```
plt.ylabel("Score")
plt.title("YOLOv8 Plate Detection Metrics")
for k,v in metrics.items():
    plt.text(k, v+0.02, f"{v:.2f}", ha="center")
plt.show()
```




```
                all       2048       2195      0.981      0.937      0.965      0.6

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     40/50      1.38G      1.011     0.4395      1.062          2        416: 100% ━━
                Class     Images  Instances      Box(P          R      mAP50  mAP50-9
                  all       2048       2195      0.973      0.939      0.964      0.6
  Closing dataloader mosaic
  Augmentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7))

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

Quick check ONNX

Export NMS-free ONNX (needed for Hailo NMS later)

```
best_pt = Path("runs/detect/train/weights/best.pt")
assert best_pt.exists(), "best.pt not found — check training run path."
onnx_path = YOLO(str(best_pt)).export(
    format="onnx",
    dynamic=False,       # static shapes
    opset=11,            # 12/13 OK
    imgsz=IMGSZ,
    half=False,          # keep FP32; Hailo will quantize
    simplify=True,
    nms=False            # <-- IMPORTANT: no builtin NMS
)
print("ONNX exported to:", onnx_path)
```

```
Ultralytics 8.3.186 🚀 Python-3.12.11 torch-2.8.0+cu126 CPU (Intel Xeon 2.20GHz)
💡 ProTip: Export to OpenVINO format for best performance on Intel hardware. Learn more
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs

PyTorch: starting from 'runs/detect/train/weights/best.pt' with input shape (1, 3, 416,
requirements: Ultralytics requirements ['onnx>=1.12.0,<1.18.0', 'onnxslim>=0.1.59'] not

requirements: AutoUpdate success ✅ 2.5s
WARNING ⚠️ requirements: Restart runtime or rerun command for updates to take effect

ONNX: starting export with onnx 1.17.0 opset 11...
ONNX: slimming with onnxslim 0.1.65...
ONNX: export success ✅ 3.7s, saved as 'runs/detect/train/weights/best.onnx' (11.6 MB)

Export complete (4.1s)
Results saved to /content/runs/detect/train/weights
Predict:         yolo predict task=detect model=runs/detect/train/weights/best.onnx imgs
Validate:        yolo val task=detect model=runs/detect/train/weights/best.onnx imgsz=41
Visualize:       https://netron.app
ONNX exported to: runs/detect/train/weights/best.onnx
```

```
import onnx, onnxruntime as ort
m = onnx.load(onnx_path); onnx.checker.check_model(m)
sess = ort.InferenceSession(str(onnx_path), providers=["CUDAExecutionProvider","CPUExecutior
print("Input:", sess.get_inputs()[0].name, sess.get_inputs()[0].shape)
for o in sess.get_outputs():
    print("Output:", o.name, o.shape)
```

```
Input: images [1, 3, 416, 416]
Output: output0 [1, 5, 3549]
```

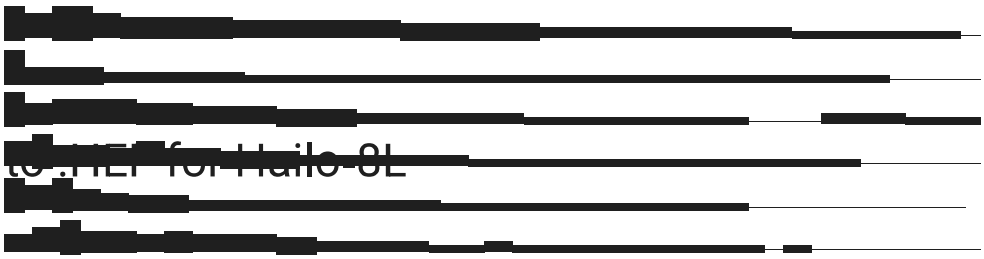
```
50 epochs completed in 0.536 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.2MB

Validating runs/detect/train/weights/best.pt...
```

Build a calibration set (for Hailo quantization)

```
import cv2, os
calib_dir = Path("calib_npys"); calib_dir.mkdir(exist_ok=True)

def preprocess(path, size=IMGSZ):
    im = cv2.imread(path)[..., ::-1]     # BGR->RGB
    im = cv2.resize(im, (size, size))
    im = im.astype("float32")/255.0
    im = np.transpose(im, (2,0,1))       # HWC->CHW
    return im[None]                      # NCHW, batch=1

train_imgs = glob.glob(f"{data_dir}/train/images/*")
val_imgs   = glob.glob(f"{data_dir}/valid/images/*")
all_imgs = train_imgs + val_imgs
random.shuffle(all_imgs)
```

```
subset = all_imgs[:300] if len(all_imgs) >= 300 else all_imgs

for i, p in enumerate(subset):
    np.save(calib_dir/f"calib_{i:04d}.npy", preprocess(p))

print("Calib samples:", len(list(calib_dir.glob('*.npy'))))
```

```
Calib samples: 300
```



(Off-Colab) Compile to .HEF for Hailo-8L

Variables

Run summary:

lr/pg0	0.0003
lr/pg1	0.0003
lr/pg2	0.0003
metrics/mAP50(B)	0.96587
metrics/mAP50-95(B)	0.70315
metrics/precision(B)	0.97924
metrics/recall(B)	0.93895
model/GFLOPs	8.194
model/parameters	3011043
model/speed_PyTorch(ms)	0.738
train/box_loss	0.94401
train/cls_loss	0.35957
train/dfl_loss	1.06887
val/box_loss	1.04907
val/cls_loss	0.39244
val/dfl_loss	1.11905

View run **train** at: https://wandb.ai/trinc-ciprian-universitatea-de-vest-din-

```
MODEL_ONNX=/path/to/yolov8_plate.onnx CALIB_DIR=/path/to/calib_npys OUTDIR=/path/to/out
mkdir -p "$OUTDIR"
```

Compile ONNX -> HEF with Hailo-NMS and batch=1 for Hailo-8L

```
hailomz compile custom_yolov8_lpr
–ckpt "
```

$$MODEL_ONNX" \ --hw\_arch hailo8l \ --batch-size 1 \ --calib-path"$$

```
CALIB_DIR"
–calib-path calib_dir
–nms
–output "$OUTDIR/custom_yolov8_lpr.hef"
–profile
```

```
# --- Setup (run once) ---
# Point to the latest training run
run_dir = sorted(Path("runs/detect").glob("train"), key=lambda p: p.stat().st_mtime)[-1]
best_pt = run_dir / "weights" / "best.pt"
print("Using:", best_pt)

model = YOLO(str(best_pt))
IMGSZ = 640
```

```
Using: runs/detect/train/weights/best.pt
```

1) Test in Colab with your trained YOLO (image)

```
from pathlib import Path
import glob, random, IPython.display as disp# Option A: use a dataset image
img_path = random.choice(glob.glob("/content/License-Plate-Recognition-11/valid/images/*"))

# Option B: upload your own image
# from google.colab import files
# up = files.upload()  # pick a local .jpg/.png
# img_path = list(up.keys())[0]

res = model.predict(img_path, imgsz=IMGSZ, conf=0.25, save=True, project="demo_out", name="i
print("Saved prediction to:", "demo_out/image")
disp.Image(filename=str(next((Path("demo_out/image")).glob("*.jpg"))))
```

```
image 1/1 /content/License-Plate-Recognition-11/valid/images/xemay351_jpg.rf.d165fe690c6
Speed: 1.8ms preprocess, 65.1ms inference, 1.6ms postprocess per image at shape (1, 3, 4
Results saved to demo_out/image
Saved prediction to: demo_out/image
```

2) Test in Colab with your trained YOLO (image & video)

```
from ultralytics import YOLO
from pathlib import Path
import glob, random, cv2, numpy as np
import IPython.display as disp

# --- paths ---
DATA_DIR = Path("/content/License-Plate-Recognition-11")
IMG_DIR  = DATA_DIR / "valid" / "images"     # you can switch to "test/images"
```

```
RUNS_DIR = Path("runs/detect")
OUT_DIR  = Path("single_test_out"); OUT_DIR.mkdir(exist_ok=True, parents=True)

# --- load best checkpoint from the latest training run ---
run_dir = sorted(RUNS_DIR.glob("train"), key=lambda p: p.stat().st_mtime)[-1]
best_pt = run_dir / "weights" / "best.pt"
print("Using model:", best_pt)
model = YOLO(str(best_pt))
IMGSZ = 640

# --- pick ONE image ---
all_imgs = sorted(glob.glob(str(IMG_DIR/"*")))
assert all_imgs, f"No images found in {IMG_DIR}"
img_path = random.choice(all_imgs)
print("Testing on:", img_path)

# --- run detection & save Ultralytics-rendered result ---
res = model.predict(
    source=img_path,
    imgsz=IMGSZ,
    conf=0.25,
    save=True,
    project=str(OUT_DIR),
    name="pred"
)[0]

# the rendered image Ultralytics saved:
rendered = next((OUT_DIR / "pred").glob("*.jpg"))
print("Rendered with boxes ->", rendered)
disp.Image(filename=str(rendered), width=900)
```

```
Using model: runs/detect/train/weights/best.pt
Testing on: /content/License-Plate-Recognition-11/valid/images/xemay2175_jpg.rf.f67a41bf

image 1/1 /content/License-Plate-Recognition-11/valid/images/xemay2175_jpg.rf.f67a41bf99
Speed: 1.9ms preprocess, 9.4ms inference, 2.1ms postprocess per image at shape (1, 3, 41
Results saved to single_test_out/pred
Rendered with boxes -> single_test_out/pred/xemay2175_jpg.rf.f67a41bf991440fab0180fac4e6
```

3) Test in Colab with your trained YOLO (video)

```
# OCR step (host-side). Install once:
%pip -q install easyocr > /dev/null

import easyocr
reader = easyocr.Reader(['en'])  # add locale codes if needed, e.g., ['en','ro','de']

# Load original image (BGR)
img_bgr = cv2.imread(img_path)
h, w = img_bgr.shape[:2]
```

```
# Copy for drawing OCR text
annot = img_bgr.copy()

# Iterate detections
texts = []
for b, conf, cls in zip(res.boxes.xyxy.cpu().numpy().astype(int),
                        res.boxes.conf.cpu().numpy(),
                        res.boxes.cls.cpu().numpy()):
    x1, y1, x2, y2 = b
    # clamp to image bounds
    x1, y1 = max(0,x1), max(0,y1)
    x2, y2 = min(w-1,x2), min(h-1,y2)
    crop = img_bgr[y1:y2, x1:x2]

    # OCR
    ocr_res = reader.readtext(crop)
    # pick the highest-score string if any
    if len(ocr_res):
        best = max(ocr_res, key=lambda t: t[2])  # (bbox, text, score)
        plate_text, score = best[1], best[2]
    else:
        plate_text, score = "", 0.0

    texts.append((b, plate_text, float(score), float(conf)))

    # Draw box + OCR text
    cv2.rectangle(annot, (x1,y1), (x2,y2), (0,255,0), 2)
    label = f"{plate_text if plate_text else 'plate'}  conf:{conf:.2f}  ocr:{score:.2f}"
    cv2.putText(annot, label, (x1, max(0,y1-8)), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2

# Save and show
annot_path = OUT_DIR / "pred_with_ocr.jpg"
cv2.imwrite(str(annot_path), annot)
print("Annotated (with OCR) ->", annot_path)
disp.Image(filename=str(annot_path), width=900)

# Also print the raw detections + OCR strings
for b, plate_text, ocr_s, det_c in texts:
    print(f"Box {b}  det_conf={det_c:.2f}  OCR='{plate_text}'  ocr_conf={ocr_s:.2f}")
```

```
WARNING:easyocr.easyocr:Downloading detection model, please wait. This may take several
Progress: |████████████████████████████████████████| 100.0% CompleteWARNING:ea
Progress: |████████████████████████████████████████| 100.0% CompleteAnnotated
Box [121  99 168 132]  det_conf=0.89  OCR='366,874'  ocr_conf=0.28
```

Dați dublu clic (sau apăsați pe Enter) pentru a edita

```
from pathlib import Path

# Look under runs/detect/
list(Path("runs/detect").glob("train*"))
```

```
[PosixPath('runs/detect/train')]
```

locate

```
run_dir = sorted(Path("runs/detect").glob("train"), key=lambda p: p.stat().st_mtime)[-1]
print("Using run:", run_dir)
best_pt = run_dir / "weights" / "best.pt"
print("Best checkpoint exists:", best_pt.exists(), best_pt)
```

```
Using run: runs/detect/train
Best checkpoint exists: True runs/detect/train/weights/best.pt
```

Export to ONNX (no NMS)

```
from ultralytics import YOLO

model = YOLO(str(best_pt))
onnx_path = model.export(
    format="onnx",
    dynamic=False,
    opset=12,
    imgsz=640,
    half=False,
    simplify=True,
    nms=False    # important for Hailo
)

print("Exported ONNX path:", onnx_path)
```

```
Ultralytics 8.3.186 🚀 Python-3.12.11 torch-2.8.0+cu126 CPU (Intel Xeon 2.20GHz)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs

PyTorch: starting from 'runs/detect/train/weights/best.pt' with input shape (1, 3, 640,

ONNX: starting export with onnx 1.17.0 opset 12...
ONNX: slimming with onnxslim 0.1.65...
ONNX: export success ✅ 1.6s, saved as 'runs/detect/train/weights/best.onnx' (1

Export complete (1.9s)
Results saved to /content/runs/detect/train/weights
```

```
Predict:         yolo predict task=detect model=runs/detect/train/weights/best.onnx imgs
Validate:        yolo val task=detect model=runs/detect/train/weights/best.onnx imgsz=64
Visualize:       https://netron.app
Exported ONNX path: runs/detect/train/weights/best.onnx
```

Dați dublu clic (sau apăsați pe Enter) pentru a edita

```
list((run_dir / "weights").glob("*.onnx"))
```

```
[PosixPath('runs/detect/train/weights/best.onnx')]
```

Test the ONNX

```
import onnx, onnxruntime as ort, numpy as np

onnx_path = str(run_dir / "weights" / "best.onnx")
m = onnx.load(onnx_path); onnx.checker.check_model(m)

sess = ort.InferenceSession(onnx_path, providers=["CPUExecutionProvider"])
inp = sess.get_inputs()[0].name
print("Input:", inp, sess.get_inputs()[0].shape)

dummy = np.random.rand(1,3,640,640).astype("float32")
_ = sess.run(None, {inp: dummy})
print("ONNX forward pass OK")
```

```
Input: images [1, 3, 640, 640]
ONNX forward pass OK
```

Dați dublu clic (sau apăsați pe Enter) pentru a edita

```
from pathlib import Path
import glob, random, cv2, numpy as np

DATA_DIR = Path("/content/License-Plate-Recognition-11")
IMGSZ = 640
CALIB_DIR = Path("calib_npys"); CALIB_DIR.mkdir(exist_ok=True, parents=True)

def preprocess(path, size=IMGSZ):
    im = cv2.imread(path)[..., ::-1]         # BGR->RGB
    im = cv2.resize(im, (size, size))
    im = (im.astype("float32")/255.0).transpose(2,0,1)  # CHW
    return im[None]                           # NCHW
```

```
train_imgs = glob.glob(str(DATA_DIR/"train/images/*"))
val_imgs   = glob.glob(str(DATA_DIR/"valid/images/*"))
all_imgs = train_imgs + val_imgs
random.shuffle(all_imgs)
subset = all_imgs[:300] if len(all_imgs) >= 300 else all_imgs

for i, p in enumerate(subset):
    np.save(CALIB_DIR/f"calib_{i:04d}.npy", preprocess(p))

len(list(CALIB_DIR.glob("*.npy")))
```

```
300
```

Dați dublu clic (sau apăsați pe Enter) pentru a edita

```
from ultralytics import YOLO
from pathlib import Path
import glob, random, IPython.display as disp

best_pt = Path("runs/detect/train/weights/best.pt")
model = YOLO(str(best_pt))
IMGSZ = 640

img_path = random.choice(glob.glob("/content/License-Plate-Recognition-11/valid/images/*"))
print("Testing on:", img_path)

res = model.predict(source=img_path, imgsz=IMGSZ, conf=0.25, save=True,
                    project="demo_out", name="image")
print("Saved:", "demo_out/image")
disp.Image(filename=str(next((Path("demo_out/image")).glob("*.jpg"))), width=900)
```

```
Testing on: /content/License-Plate-Recognition-11/valid/images/xemay38_jpg.rf.8307d0b6b8

image 1/1 /content/License-Plate-Recognition-11/valid/images/xemay38_jpg.rf.8307d0b6b881
Speed: 1.7ms preprocess, 7.2ms inference, 1.6ms postprocess per image at shape (1, 3, 41
Results saved to demo_out/image2
Saved: demo_out/image
```

OCR detection

```
%pip -q install easyocr
import easyocr, cv2
reader = easyocr.Reader(['en'])  # add your locale codes if needed

img = cv2.imread(img_path)
pred = model(img, imgsz=IMGSZ, conf=0.25)[0]
for box in pred.boxes.xyxy.cpu().numpy().astype(int):
    x1,y1,x2,y2 = box
    crop = img[y1:y2, x1:x2]
```

```
    print("OCR:", reader.readtext(crop))
```

```
0: 416x640 1 License_Plate, 7.4ms
Speed: 1.7ms preprocess, 7.4ms inference, 1.6ms postprocess per image at shape (1, 3, 41
OCR: [([[np.int32(17), np.int32(7)], [np.int32(81), np.int32(7)], [np.int32(81), np.int3
```

```
!zip -r -q onnx_and_calib.zip runs/detect/train/weights/best.onnx calib_npys
```

Dați dublu clic (sau apăsați pe Enter) pentru a edita

```
from ultralytics import YOLO
import glob, random
from pathlib import Path
from IPython.display import Image

model = YOLO("runs/detect/train/weights/best.pt")
img = random.choice(glob.glob("/content/License-Plate-Recognition-11/valid/images/*"))
pred = model.predict(img, imgsz=640, conf=0.25, save=True, project="demo_out", name="img")
Image(filename=str(next((Path("demo_out/img")).glob("*.jpg"))), width=900)
```

```
image 1/1 /content/License-Plate-Recognition-11/valid/images/00cac7ea145fc734_jpg.rf.f26
Speed: 2.5ms preprocess, 65.1ms inference, 2.0ms postprocess per image at shape (1, 3, 4
Results saved to demo_out/img
```

Example from Youtube

```
!pip -q install yt-dlp ultralytics opencv-python-headless
```

```
import yt_dlp
```

```
url = "https://www.youtube.com/watch?v=7OIoL7irIF8"  # traffic driving sample

ydl_opts = {
    "format": "best[ext=mp4]",
    "outtmpl": "input_video.%(ext)s"
}
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download([url])
```

```
[youtube] Extracting URL: https://www.youtube.com/watch?v=7OIoL7irIF8
[youtube] 7OIoL7irIF8: Downloading webpage
[youtube] 7OIoL7irIF8: Downloading tv client config
[youtube] 7OIoL7irIF8: Downloading player 6742b2b9-main
[youtube] 7OIoL7irIF8: Downloading tv player API JSON
[youtube] 7OIoL7irIF8: Downloading tv simply player API JSON
[info] 7OIoL7irIF8: Downloading 1 format(s): 18
[download] Sleeping 2.00 seconds as required by the site...
[download] Destination: input_video.mp4
[download] 100% of    3.79MiB in 00:00:00 at 11.96MiB/s
```

```
from ultralytics import YOLO

# load your trained model
model = YOLO("runs/detect/train/weights/best.pt")

# run detection on video
results = model.predict(
    source="input_video.mp4",   # downloaded video
    imgsz=640,
    conf=0.25,
    save=True,
    project="demo_out",
    name="youtube_video"
)

print("Processed video saved to:", "demo_out/youtube_video/input_video.mp4")
```

```
WARNING ⚠️
inference results will accumulate in RAM unless `stream=True` is passed, causing poten
errors for large sources or long-running streams and videos. See https://docs.ultralyt

Example:
    results = model(source=..., stream=True)  # generator of Results objects
    for r in results:
        boxes = r.boxes  # Boxes object for bbox outputs
        masks = r.masks  # Masks object for segment masks outputs
        probs = r.probs  # Class probabilities for classification outputs

video 1/1 (frame 1/1253) /content/input_video.mp4: 384x640 1 License_Plate, 62.2
video 1/1 (frame 2/1253) /content/input_video.mp4: 384x640 (no detections), 7.0ms
```

```
video 1/1 (frame 3/1253) /content/input_video.mp4: 384x640 (no detections), 7.1ms
video 1/1 (frame 4/1253) /content/input_video.mp4: 384x640 (no detections), 6.7ms
video 1/1 (frame 5/1253) /content/input_video.mp4: 384x640 (no detections), 6.7ms
video 1/1 (frame 6/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 7/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 8/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 9/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 10/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 11/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 12/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 13/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 14/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.9ms
video 1/1 (frame 15/1253) /content/input_video.mp4: 384x640 (no detections), 6.7ms
video 1/1 (frame 16/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 17/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.4ms
video 1/1 (frame 18/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.6ms
video 1/1 (frame 19/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.6ms
video 1/1 (frame 20/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 21/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.4ms
video 1/1 (frame 22/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.5ms
video 1/1 (frame 23/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 24/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 25/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 26/1253) /content/input_video.mp4: 384x640 (no detections), 6.8ms
video 1/1 (frame 27/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 28/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 29/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 30/1253) /content/input_video.mp4: 384x640 (no detections), 6.6ms
video 1/1 (frame 31/1253) /content/input_video.mp4: 384x640 (no detections), 6.9ms
video 1/1 (frame 32/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.7ms
video 1/1 (frame 33/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.7ms
video 1/1 (frame 34/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.6ms
video 1/1 (frame 35/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.5ms
video 1/1 (frame 36/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.5ms
video 1/1 (frame 37/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.5ms
video 1/1 (frame 38/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.4ms
video 1/1 (frame 39/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.4ms
video 1/1 (frame 40/1253) /content/input_video.mp4: 384x640 (no detections), 6.4ms
video 1/1 (frame 41/1253) /content/input_video.mp4: 384x640 2 License_Plates, 6.5ms
video 1/1 (frame 42/1253) /content/input_video.mp4: 384x640 1 License_Plate, 6.5ms
video 1/1 (frame 43/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
video 1/1 (frame 44/1253) /content/input_video.mp4: 384x640 (no detections), 6.7ms
video 1/1 (frame 45/1253) /content/input_video.mp4: 384x640 (no detections), 6.5ms
```

```
!ffmpeg -i /content/demo_out/youtube_video/input_video.avi \
       -vcodec libx264 -crf 23 -preset veryfast \
       -acodec aac -strict -2 \
       /content/demo_out/youtube_video/input_video.mp4 -y
```

```
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg develo
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=harde
```

```
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, avi, from '/content/demo_out/youtube_video/input_video.avi':
  Metadata:
    software        : Lavf59.27.100
  Duration: 00:00:41.77, start: 0.000000, bitrate: 15446 kb/s
  Stream #0:0: Video: mjpeg (Baseline) (MJPG / 0x47504A4D), yuvj420p(pc, bt470bg/unknowr
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x5ae680c5cd40] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 E
[libx264 @ 0x5ae680c5cd40] profile High, level 3.0, 4:2:0, 8-bit
[libx264 @ 0x5ae680c5cd40] 264 - core 163 r3060 5db6aa6 - H.264/MPEG-4 AVC codec - Copyl
Output #0, mp4, to '/content/demo_out/youtube_video/input_video.mp4':
  Metadata:
    software        : Lavf59.27.100
    encoder         : Lavf58.76.100
  Stream #0:0: Video: h264 (avc1 / 0x31637661), yuvj420p(pc, bt470bg/unknown/unknown, pr
    Metadata:
      encoder         : Lavc58.134.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: N/A
frame= 1253 fps=526 q=-1.0 Lsize=    4802kB time=00:00:41.66 bitrate= 944.0kbits/s speed
video:4786kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead
[libx264 @ 0x5ae680c5cd40] frame I:6     Avg QP:23.90  size: 49560
[libx264 @ 0x5ae680c5cd40] frame P:343   Avg QP:26.51  size: 10246
[libx264 @ 0x5ae680c5cd40] frame B:904   Avg QP:30.79  size:  1204
[libx264 @ 0x5ae680c5cd40] consecutive B-frames:  2.8%  2.4%  1.9% 92.9%
[libx264 @ 0x5ae680c5cd40] mb I  I16..4:  0.7% 20.5% 78.8%
[libx264 @ 0x5ae680c5cd40] mb P  I16..4:  0.4%  6.1%  0.8%  P16..4: 48.8% 17.0% 13.3%  0
[libx264 @ 0x5ae680c5cd40] mb B  I16..4:  0.1%  0.3%  0.0%  B16..8: 12.5%  5.4%  1.0%  0
[libx264 @ 0x5ae680c5cd40] 8x8 transform intra:71.5% inter:27.2%
[libx264 @ 0x5ae680c5cd40] coded y,uvDC,uvAC intra: 76.1% 64.5% 17.1% inter: 14.4% 9.0%
[libx264 @ 0x5ae680c5cd40] i16 v,h,dc,p: 21% 41% 34%  5%
[libx264 @ 0x5ae680c5cd40] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 23% 34% 31%  2%  2%  1%  3%  1
[libx264 @ 0x5ae680c5cd40] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 19% 28% 12%  5%  6%  6%  9%  6
[libx264 @ 0x5ae680c5cd40] i8c dc,h,v,p: 45% 36% 15%  4%
[libx264 @ 0x5ae680c5cd40] Weighted P-Frames: Y:9.6% UV:0.9%
[libx264 @ 0x5ae680c5cd40] kb/s:938.60
```

```
from IPython.display import Video
Video("/content/demo_out/youtube_video/input_video.mp4", embed=True, width=800)
```

0:00 / 0:41

Mount Drive & point to file

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# <<< path in Drive >>>
INPUT_MOV = "/content/drive/MyDrive/Video_H/IMG_0130.MOV"
```

```
Mounted at /content/drive
```

convert MOV → MP4 for smoother processing

```
!ffmpeg -y -i "$INPUT_MOV" -vcodec libx264 -crf 20 -preset veryfast -acodec aac /content/inp
```

```
ffmpeg version 4.4.2-0ubuntu0.22.04.1 Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 11 (Ubuntu 11.2.0-19ubuntu1)
  configuration: --prefix=/usr --extra-version=0ubuntu0.22.04.1 --toolchain=hard
  libavutil      56. 70.100 / 56. 70.100
  libavcodec     58.134.100 / 58.134.100
  libavformat    58. 76.100 / 58. 76.100
```

```
  libavdevice    58. 13.100 / 58. 13.100
  libavfilter     7.110.100 /  7.110.100
  libswscale      5.  9.100 /  5.  9.100
  libswresample   3.  9.100 /  3.  9.100
  libpostproc    55.  9.100 / 55.  9.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from '/content/drive/MyDrive/Video_H/IMG_0130.MOV':
  Metadata:
    major_brand     : qt
    minor_version   : 0
    compatible_brands: qt
    creation_time   : 2025-08-24T13:55:48.000000Z
    com.apple.quicktime.location.accuracy.horizontal: 7.931697
    com.apple.quicktime.location.ISO6709: +45.7326+021.2351+092.789/
    com.apple.quicktime.make: Apple
    com.apple.quicktime.model: iPhone 11
    com.apple.quicktime.software: 17.5.1
    com.apple.quicktime.creationdate: 2025-08-24T16:27:22+0300
    com.apple.photos.originating.signature: AXiC42ncYYXTiJFZFWIocCpovxdo
  Duration: 00:00:09.77, start: 0.000000, bitrate: 15319 kb/s
  Stream #0:0(und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 152 k
    Metadata:
      creation_time   : 2025-08-24T13:55:48.000000Z
      handler_name    : Core Media Audio
      vendor_id       : [0][0][0][0]
  Stream #0:1(und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(tv, bt709), 1920x1
    Metadata:
      creation_time   : 2025-08-24T13:55:48.000000Z
      handler_name    : Core Media Video
      vendor_id       : [0][0][0][0]
      encoder         : H.264
  Stream #0:2(und): Data: none (mebx / 0x7862656D), 0 kb/s (default)
    Metadata:
      creation_time   : 2025-08-24T13:55:48.000000Z
      handler_name    : Core Media Metadata
  Stream #0:3(und): Data: none (mebx / 0x7862656D), 0 kb/s (default)
    Metadata:
      creation_time   : 2025-08-24T13:55:48.000000Z
      handler_name    : Core Media Metadata
  Stream #0:4(und): Data: none (mebx / 0x7862656D), 34 kb/s (default)
    Metadata:
      creation_time   : 2025-08-24T13:55:48.000000Z
      handler_name    : Core Media Metadata
Stream mapping:
  Stream #0:1 -> #0:0 (h264 (native) -> h264 (libx264))
  Stream #0:0 -> #0:1 (aac (native) -> aac (native))
Press [q] to stop, [?] for help
[libx264 @ 0x58f7840d1d40] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3
[libx264 @ 0x58f7840d1d40] profile High, level 4.0, 4:2:0, 8-bit
[libx264 @ 0x58f7840d1d40] 264 - core 163 r3060 5db6aa6 - H.264/MPEG-4 AVC codec - Cop
Output #0, mp4, to '/content/input_videoP.mp4':
  Metadata:
```

Run YOLO plate detector on the video

```
from ultralytics import YOLO
from pathlib import Path

model = YOLO("runs/detect/train/weights/best.pt")  # your trained model

res = model.predict(
    source="/content/input_videoP.mp4",   # the mp4 we just made
    imgsz=640,
    conf=0.25,
    save=True,
    project="demo_out",
    name="drive_videoP",
    vid_stride=1
)

# Ultralytics writes output into demo_out/drive_video/
out_dir = Path("demo_out/drive_videoP")
print("Outputs in:", out_dir)
print("Files:", [p.name for p in out_dir.iterdir()])
```

```
    WARNING ⚠️
    inference results will accumulate in RAM unless `stream=True` is passed, causing poten
    errors for large sources or long-running streams and videos. See https://docs.ultralyt

    Example:
        results = model(source=..., stream=True)  # generator of Results objects
        for r in results:
            boxes = r.boxes  # Boxes object for bbox outputs
            masks = r.masks  # Masks object for segment masks outputs
            probs = r.probs  # Class probabilities for classification outputs

    video 1/1 (frame 1/293) /content/input_videoP.mp4: 384x640 4 License_Plates, 7.9ms
    video 1/1 (frame 2/293) /content/input_videoP.mp4: 384x640 4 License_Plates, 10.7ms
    video 1/1 (frame 3/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 7.2ms
    video 1/1 (frame 4/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 7.0ms
    video 1/1 (frame 5/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 7.0ms
    video 1/1 (frame 6/293) /content/input_videoP.mp4: 384x640 4 License_Plates, 7.0ms
    video 1/1 (frame 7/293) /content/input_videoP.mp4: 384x640 4 License_Plates, 7.0ms
    video 1/1 (frame 8/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 7.1ms
    video 1/1 (frame 9/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 7.1ms
    video 1/1 (frame 10/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 10.5ms
    video 1/1 (frame 11/293) /content/input_videoP.mp4: 384x640 3 License_Plates, 7.2ms
    video 1/1 (frame 12/293) /content/input_videoP.mp4: 384x640 4 License_Plates, 7.0ms
    video 1/1 (frame 13/293) /content/input_videoP.mp4: 384x640 4 License_Plates, 7.0ms
```