```
!python -V
!pip install -U pip setuptools wheel
```

```
Python 3.12.12
Requirement already satisfied: pip in /usr/local/lib/python3.12/dist-packages (24.1.2)
Collecting pip
  Downloading pip-25.3-py3-none-any.whl.metadata (4.7 kB)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (75.2.0)
Collecting setuptools
  Downloading setuptools-80.9.0-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: wheel in /usr/local/lib/python3.12/dist-packages (0.45.1)
Downloading pip-25.3-py3-none-any.whl (1.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.8/1.8 MB 35.2 MB/s eta 0:00:00
Downloading setuptools-80.9.0-py3-none-any.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 82.2 MB/s eta 0:00:00
Installing collected packages: setuptools, pip
  Attempting uninstall: setuptools
    Found existing installation: setuptools 75.2.0
    Uninstalling setuptools-75.2.0:
      Successfully uninstalled setuptools-75.2.0
  Attempting uninstall: pip
    Found existing installation: pip 24.1.2
    Uninstalling pip-24.1.2:
      Successfully uninstalled pip-24.1.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This b
ipython 7.34.0 requires jedi>=0.16, which is not installed.
Successfully installed pip-25.3 setuptools-80.9.0
WARNING: The following packages were previously imported in this runtime:
  [_distutils_hack]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
!pip install -q "pillow==10.4.0" "sse-starlette==0.10.3" "ultralytics==8.3.227" matplotlib pandas tqdm
```

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This b
mcp 1.20.0 requires sse-starlette>=1.6.1, but you have sse-starlette 0.10.3 which is incompatible.
```

```
import sys, importlib
import PIL, torch
from ultralytics import YOLO
import matplotlib, pandas as pd

print("python", sys.version.split()[0])
print("pillow", PIL.__version__)
print("torch", torch.__version__)
print("ultralytics", importlib.import_module("ultralytics").__version__)
print("matplotlib", matplotlib.__version__)
print("pandas", pd.__version__)
```

```
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://d
python 3.12.12
pillow 11.3.0
torch 2.8.0+cu126
ultralytics 8.3.227
matplotlib 3.10.0
pandas 2.2.2
```

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

import zipfile, os, shutil
from pathlib import Path

ZIP_PATH = Path("/content/drive/MyDrive/ODA.zip")
WORKDIR = Path("/content/awc_workspace")
PROJECT = Path("/content/awc_project")

if WORKDIR.exists():
    shutil.rmtree(WORKDIR)
WORKDIR.mkdir(parents=True, exist_ok=True)
PROJECT.mkdir(parents=True, exist_ok=True)

if not ZIP_PATH.exists():
    raise SystemExit(f"Dataset not found: {ZIP_PATH}")
```

```
print(f"Extracting {ZIP_PATH} ...")
with zipfile.ZipFile(ZIP_PATH, "r") as z:
    z.extractall(WORKDIR)

print("Unzipped. Top-level content:")
for p in WORKDIR.iterdir():
    print(" ", p.name)
```

```
Mounted at /content/drive
Extracting /content/drive/MyDrive/ODA.zip ...
Unzipped. Top-level content:
  __MACOSX
  ODA
```

```
from pathlib import Path

ROOT = WORKDIR / "ODA"
if not ROOT.exists():
    ROOT = WORKDIR

print("DATASET ROOT:", ROOT)
print("Train images:", len(list((ROOT / "train" / "images").glob("*"))))
print("Train labels:", len(list((ROOT / "train" / "labels").glob("*.txt"))))
print("Val images:", len(list((ROOT / "valid" / "images").glob("*"))))
print("Val labels:", len(list((ROOT / "valid" / "labels").glob("*.txt"))))
print("Test images:", len(list((ROOT / "test" / "images").glob("*"))))
print("Test labels:", len(list((ROOT / "test" / "labels").glob("*.txt"))))

yaml_path = ROOT / "data.yaml"
if yaml_path.exists():
    print("\nPreview of data.yaml:\n")
    print(yaml_path.read_text()[:400])
else:
    raise SystemExit("data.yaml not found — verify dataset structure.")
```

```
DATASET ROOT: /content/awc_workspace/ODA
Train images: 4764
Train labels: 4764
Val images: 197
Val labels: 197
Test images: 204
Test labels: 204

Preview of data.yaml:

train: ../train/images
val: ../valid/images
test: ../test/images

nc: 7
names: ['BUS', 'Bus', 'Car', 'People', 'Traffic lights', 'Truck', 'car']

roboflow:
  workspace: ml-hdvo4
  project: object-detection-65ur8
  version: 5
  license: CC BY 4.0
  url: https://universe.roboflow.com/ml-hdvo4/object-detection-65ur8/dataset/5
```

```
from ultralytics import YOLO
from pathlib import Path

ROOT = Path("/content/awc_workspace/ODA")
if not (ROOT / "data.yaml").exists():
    ROOT = Path("/content/awc_workspace")

data_yaml = str((ROOT / "data.yaml").resolve())

model = YOLO("yolo11n.pt")
model.train(
    data=data_yaml,
    epochs=30,
    batch=8,
    imgsz=640,
    name="ODA_yolo11n_train",
    seed=42,
    workers=2
)
```

```
0.76076,     0.76176,     0.76276,     0.76376,     0.76476,     0.76577,     0.76677,     0.76777,
           0.76877,     0.76977,     0.77077,     0.77177,     0.77277,     0.77377,     0.77477,     0.77578,
0.77678,     0.77778,     0.77878,     0.77978,     0.78078,     0.78178,     0.78278,     0.78378,
0.78478,     0.78579,     0.78679,     0.78779,     0.78879,     0.78979,     0.79079,     0.79179,
           0.79279,     0.79379,     0.79479,      0.7958,      0.7968,      0.7978,      0.7988,      0.7998,
0.8008,      0.8018,      0.8028,      0.8038,      0.8048,     0.80581,     0.80681,     0.80781,     0.80881,
0.80981,     0.81081,     0.81181,     0.81281,     0.81381,     0.81481,     0.81582,
           0.81682,     0.81782,     0.81882,     0.81982,     0.82082,     0.82182,     0.82282,     0.82382,
0.82482,     0.82583,     0.82683,     0.82783,     0.82883,     0.82983,     0.83083,     0.83183,
0.83283,     0.83383,     0.83483,     0.83584,     0.83684,     0.83784,     0.83884,     0.83984,
           0.84084,     0.84184,     0.84284,     0.84384,     0.84484,     0.84585,     0.84685,     0.84785,
0.84885,     0.84985,     0.85085,     0.85185,     0.85285,     0.85385,     0.85485,     0.85586,
0.85686,     0.85786,     0.85886,     0.85986,     0.86086,     0.86186,     0.86286,     0.86386,
           0.86486,     0.86587,     0.86687,     0.86787,     0.86887,     0.86987,     0.87087,     0.87187,
0.87287,     0.87387,     0.87487,     0.87588,     0.87688,     0.87788,     0.87888,     0.87988,
0.88088,     0.88188,     0.88288,     0.88388,     0.88488,     0.88589,     0.88689,     0.88789,
           0.88889,     0.88989,     0.89089,     0.89189,     0.89289,     0.89389,     0.89489,      0.8959,
0.8969,      0.8979,      0.8989,      0.8999,      0.9009,      0.9019,      0.9029,      0.9039,      0.9049,
0.90591,     0.90691,     0.90791,     0.90891,     0.90991,     0.91091,     0.91191,
           0.91291,     0.91391,     0.91491,     0.91592,     0.91692,     0.91792,     0.91892,     0.91992,
0.92092,     0.92192,     0.92292,     0.92392,     0.92492,     0.92593,     0.92693,     0.92793,
0.92893,     0.92993,     0.93093,     0.93193,     0.93293,     0.93393,     0.93493,     0.93594,
           0.93694,     0.93794,     0.93894,     0.93994,     0.94094,     0.94194,     0.94294,     0.94394,
0.94494,     0.94595,     0.94695,     0.94795,     0.94895,     0.94995,     0.95095,     0.95195,
0.95295,     0.95395,     0.95495,     0.95596,     0.95696,     0.95796,     0.95896,     0.95996,
           0.96096,     0.96196,     0.96296,     0.96396,     0.96496,     0.96597,     0.96697,     0.96797,
0.96897,     0.96997,     0.97097,     0.97197,     0.97297,     0.97397,     0.97497,     0.97598,
0.97698,     0.97798,     0.97898,     0.97998,     0.98098,     0.98198,     0.98298,     0.98398,
           0.98498,     0.98599,     0.98699,     0.98799,     0.98899,     0.98999,     0.99099,     0.99199,
0.99299,     0.99399,     0.99499,       0.996,       0.997,       0.998,       0.999,           1]), array([[
0.96154,     0.96154,     0.96154, ...,           0,           0,           0],
       [    0.97727,     0.97727,     0.97727, ...,           0,           0,           0],
       [    0.97867,     0.97867,     0.97156, ...,           0,           0,           0],
       ...,
       [    0.69231,     0.69231,     0.69231, ...,           0,           0,           0],
       [       0.92,        0.92,        0.88, ...,           0,           0,           0],
       [    0.93606,     0.93606,     0.92327, ...,           0,           0,           0]]), 'Confidence',
'Recall']]
fitness: np.float64(0.2849265602792566)
keys: ['metrics/precision(B)', 'metrics/recall(B)', 'metrics/mAP50(B)', 'metrics/mAP50-95(B)']
maps: array([    0.19679,      0.2913,     0.46772,     0.47881,    0.048769,     0.20102,     0.31007])
names: {0: 'BUS', 1: 'Bus', 2: 'Car', 3: 'People', 4: 'Traffic lights', 5: 'Truck', 6: 'car'}
nt_per_class: array([ 26,  44, 422,  98,  13,  50, 391])
nt_per_image: array([22, 39, 89, 64, 11, 41, 91])
results_dict: {'metrics/precision(B)': 0.35251382046585683, 'metrics/recall(B)': 0.5085456986477997,
'metrics/mAP50(B)': 0.363365902800351, 'metrics/mAP50-95(B)': 0.2849265602792566, 'fitness':
0.2849265602792566}
save_dir: PosixPath('/content/runs/detect/ODA_yolo11n_train')
speed: {'preprocess': 0.5004130253759018, 'inference': 4.288859857865069, 'loss': 0.0008937715789714478,
'postprocess': 7.046624116741311}
stats: {'tp': [], 'conf': [], 'pred_cls': [], 'target_cls': [], 'target_img': []}
task: 'detect'
```

```
from ultralytics import YOLO
from pathlib import Path
import pandas as pd

ROOT = Path("/content/awc_workspace/ODA")
weight_candidates = list(Path("/content/runs/detect").rglob("ODA_yolo11n_train/weights/best.pt"))
if not weight_candidates:
    weight_candidates = list(Path("/content").rglob("best.pt"))
if not weight_candidates:
    raise SystemExit("No trained weights found.")
w = str(weight_candidates[0])

model = YOLO(w)
try:
    res
except NameError:
    res = model.val(data=str(ROOT / "data.yaml"), save_json=True, plots=False)

mean_prec, mean_rec, mean_map50, mean_map5095 = res.mean_results()
fitness_val = float(res.fitness()) if callable(getattr(res, "fitness", None)) else float(getattr(res, "fitness",
overall = {"precision": mean_prec, "recall": mean_rec, "mAP50": mean_map50, "mAP50-95": mean_map5095, "fitness":
overall_df = pd.DataFrame([overall])
per_class = pd.DataFrame(res.summary())
display(overall_df.T.rename(columns={0:"value"}))
display(per_class)
```

	value
precision	0.347086
recall	0.514309
mAP50	0.363333
mAP50-95	0.285492
fitness	0.285492

	Class	Images	Instances	Box-P	Box-R	Box-F1	mAP50	mAP50-95
0	BUS	22	26	0.26916	0.46154	0.34002	0.24766	0.19646
1	Bus	39	44	0.30365	0.50000	0.37784	0.34678	0.29421
2	Car	89	422	0.48208	0.79384	0.59987	0.57932	0.46967
3	People	64	98	0.61015	0.65306	0.63088	0.68644	0.48102

```
from pathlib import Path
import yaml
ROOT = Path("/content/awc_workspace/ODA")
DATA_YAML = ROOT / "data.yaml"
train_dirs = [ROOT/"train"/"labels", ROOT/"train_dataset"/"train_labels", ROOT/"train"/"labels"]
valid_dirs = [ROOT/"valid"/"labels", ROOT/"val"/"val_labels", ROOT/"valid_dataset"/"val_labels"]
test_dirs  = [ROOT/"test"/"labels", ROOT/"test_dataset"/"test_labels"]

with open(DATA_YAML, "r") as f:
    data = yaml.safe_load(f)
old_names = data.get("names", [])
old_names = [str(n).strip() for n in old_names]

canonical = []
for n in old_names:
    c = n.strip().lower().title()
    canonical.append(c)
seen = []
for n in canonical:
    if n not in seen:
        seen.append(n)
new_names = seen

name_to_new_idx = {n:i for i,n in enumerate(new_names)}
old_idx_to_name = {}
for i,n in enumerate(old_names):
    old_idx_to_name[i] = n.strip().lower().title()

old_to_new = {}
for old_idx, name in old_idx_to_name.items():
    new_idx = name_to_new_idx.get(name)
    if new_idx is None:
        name_clean = name.strip().lower().title()
        new_idx = name_to_new_idx.setdefault(name_clean, len(name_to_new_idx))
    old_to_new[old_idx] = new_idx

label_dirs = []
for cand in train_dirs + valid_dirs + test_dirs:
    if cand.exists():
        label_dirs.append(cand)

total_files = 0
mapped_changes = 0
for lbl_dir in label_dirs:
    for txt in lbl_dir.glob("*.txt"):
        if txt.name == "classes.txt":
            continue
        total_files += 1
        lines = txt.read_text(encoding="utf-8").splitlines()
        new_lines = []
        changed = False
        for L in lines:
            parts = L.strip().split()
            if not parts:
                continue
            try:
                cls = int(parts[0])
                rest = parts[1:]
            except:
                continue
            mapped = old_to_new.get(cls, cls)
            if mapped != cls:
                changed = True
```

```
            new_lines.append(" ".join([str(mapped)] + rest))
        txt.write_text("\n".join(new_lines), encoding="utf-8")
        if changed:
            mapped_changes += 1

data["names"] = new_names
with open(DATA_YAML, "w") as f:
    yaml.safe_dump(data, f, sort_keys=False)

print("data.yaml updated at", DATA_YAML)
print("old names:", old_names)
print("new names:", new_names)
print("total label files processed:", total_files)
print("files with class id changes:", mapped_changes)
```

```
data.yaml updated at /content/awc_workspace/ODA/data.yaml
old names: ['BUS', 'Bus', 'Car', 'People', 'Traffic lights', 'Truck', 'car']
new names: ['Bus', 'Car', 'People', 'Traffic Lights', 'Truck']
total label files processed: 9929
files with class id changes: 9549
```

```
from pathlib import Path
import yaml
ROOT = Path("/content/awc_workspace/ODA")
DATA_YAML = ROOT / "data.yaml"
data = yaml.safe_load(DATA_YAML.read_text())
print("before nc:", data.get("nc"), "len(names):", len(data.get("names",[])))
names = [str(n).strip() for n in data.get("names",[])]
seen = []
for n in names:
    key = n.strip().lower()
    if key not in [s.lower() for s in seen]:
        seen.append(n.strip().title())
data["names"] = seen
data["nc"] = len(seen)
DATA_YAML.write_text(yaml.safe_dump(data, sort_keys=False))
print("after nc:", data["nc"], "len(names):", len(data["names"]))
print("names:", data["names"])
print("wrote:", DATA_YAML.resolve())
```

```
before nc: 7 len(names): 5
after nc: 5 len(names): 5
names: ['Bus', 'Car', 'People', 'Traffic Lights', 'Truck']
wrote: /content/awc_workspace/ODA/data.yaml
```

```
from pathlib import Path
import json, pickle
import pandas as pd
import matplotlib.pyplot as plt
from ultralytics import YOLO

ROOT = Path("/content/awc_workspace/ODA")
OUT = Path("/content/awc_project/metrics")
OUT.mkdir(parents=True, exist_ok=True)

def get_val_res():
    if "res" in globals():
        return globals()["res"]
    candidates = list(Path("/content/runs/detect").rglob("*/weights/best.pt")) + list(Path("/content").rglob("be
    if not candidates:
        raise SystemExit("No trained weights found and no existing 'res' object.")
    w = str(candidates[0])
    model = YOLO(w)
    return model.val(data=str((ROOT / "data.yaml")), save_json=True, plots=False)

res = get_val_res()

mean_prec, mean_rec, mean_map50, mean_map5095 = res.mean_results()
fitness = res.fitness() if callable(getattr(res, "fitness", None)) else res.fitness
fitness = float(fitness)

overall = {
    "precision": float(mean_prec),
    "recall": float(mean_rec),
    "mAP50": float(mean_map50),
    "mAP50-95": float(mean_map5095),
    "fitness": fitness
}

per_class_df = pd.DataFrame(res.summary())
per_class_df = per_class_df.where(pd.notnull(per_class_df), None)
per_class_records = per_class_df.replace({pd.NA: None}).to_dict(orient="records")
```

```
(OUT / "overall_metrics.json").write_text(json.dumps(overall, indent=2))
(OUT / "per_class_metrics.json").write_text(json.dumps(per_class_records, indent=2))

pd.DataFrame([overall]).T.rename(columns={0: "value"}).to_csv(OUT / "overall_metrics.csv")
per_class_df.to_csv(OUT / "per_class_metrics.csv", index=False)

with open(OUT / "metrics.pickle", "wb") as f:
    pickle.dump({"overall": overall, "per_class": per_class_records}, f)

plt.figure(figsize=(6,4))
keys = ["precision","recall","mAP50","mAP50-95","fitness"]
vals = [overall[k] for k in keys]
plt.bar(keys, vals)
plt.ylim(0,1)
plt.title("Overall detection metrics")
plt.tight_layout()
plt.savefig(str(OUT / "plot_overall_metrics.png"))
plt.close()

if "mAP50" in per_class_df.columns:
    plt.figure(figsize=(10,4))
    plt.bar(per_class_df["Class"].astype(str), per_class_df["mAP50"].astype(float))
    plt.xticks(rotation=45, ha="right")
    plt.ylim(0,1)
    plt.title("Per-class mAP50")
    plt.tight_layout()
    plt.savefig(str(OUT / "plot_perclass_map50.png"))
    plt.close()

print("Saved files to:", OUT)
for p in sorted(OUT.iterdir()):
    print(" ", p.name)
```

```
Saved files to: /content/awc_project/metrics
  metrics.pickle
  overall_metrics.csv
  overall_metrics.json
  per_class_metrics.csv
  per_class_metrics.json
  plot_overall_metrics.png
  plot_perclass_map50.png
```

Training Loss Curves

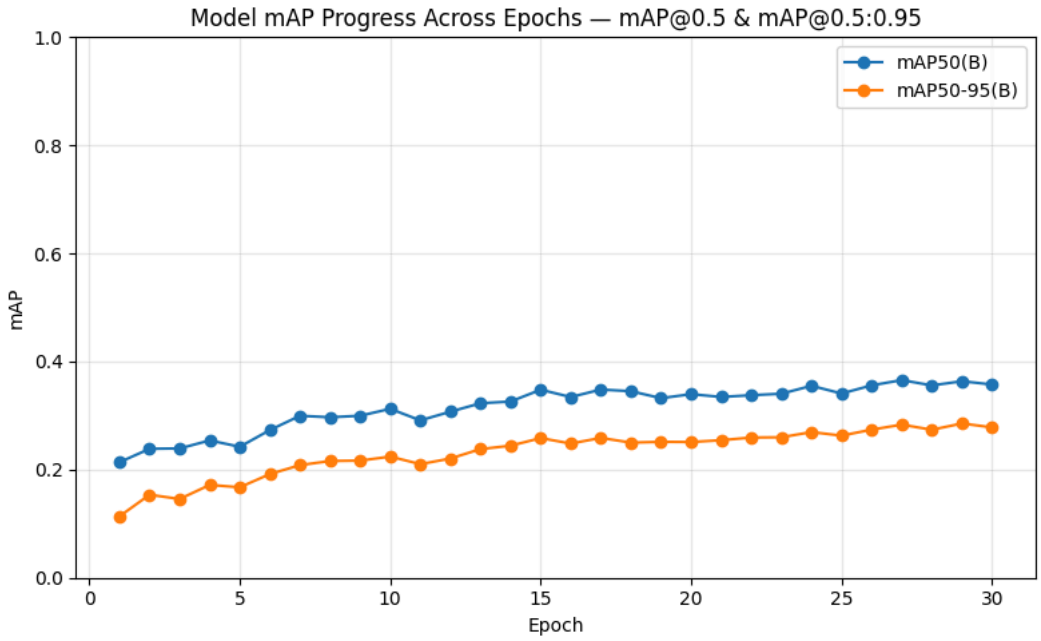
```
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
import numpy as np

def find_results_csv():
    files = list(Path("/content/runs/detect").rglob("results.csv"))
    if not files:
        raise SystemExit("No results.csv found under /content/runs/detect/")
    files.sort(key=lambda f: f.stat().st_mtime, reverse=True)
    return files[0]

csv_path = find_results_csv()
df = pd.read_csv(csv_path)
print("Using:", csv_path)

plt.figure(figsize=(8,5))
for col in ["train/box_loss", "train/cls_loss", "train/dfl_loss"]:
    if col in df.columns:
        plt.plot(df["epoch"], df[col], marker="o", label=col.split("/")[-1])
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training Loss Curves — Box, Class, DFL")
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

```
Using: /content/runs/detect/ODA_yolo11n_train/results.csv
```

mAP Progress

```
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path

csv_path = sorted(Path("/content/runs/detect").rglob("results.csv"), key=lambda f: f.stat().st_mtime, reverse=Tru
df = pd.read_csv(csv_path)
print("Using:", csv_path)

plt.figure(figsize=(8,5))
for col in ["metrics/mAP50(B)", "metrics/mAP50-95(B)"]:
    if col in df.columns:
        plt.plot(df["epoch"], df[col], marker="o", label=col.split("/")[-1])
plt.xlabel("Epoch")
plt.ylabel("mAP")
plt.title("Model mAP Progress Across Epochs — mAP@0.5 & mAP@0.5:0.95")
plt.legend()
plt.grid(True, alpha=0.3)
plt.ylim(0,1)
plt.tight_layout()
plt.show()
```

```
Using: /content/runs/detect/ODA_yolo11n_train/results.csv
```

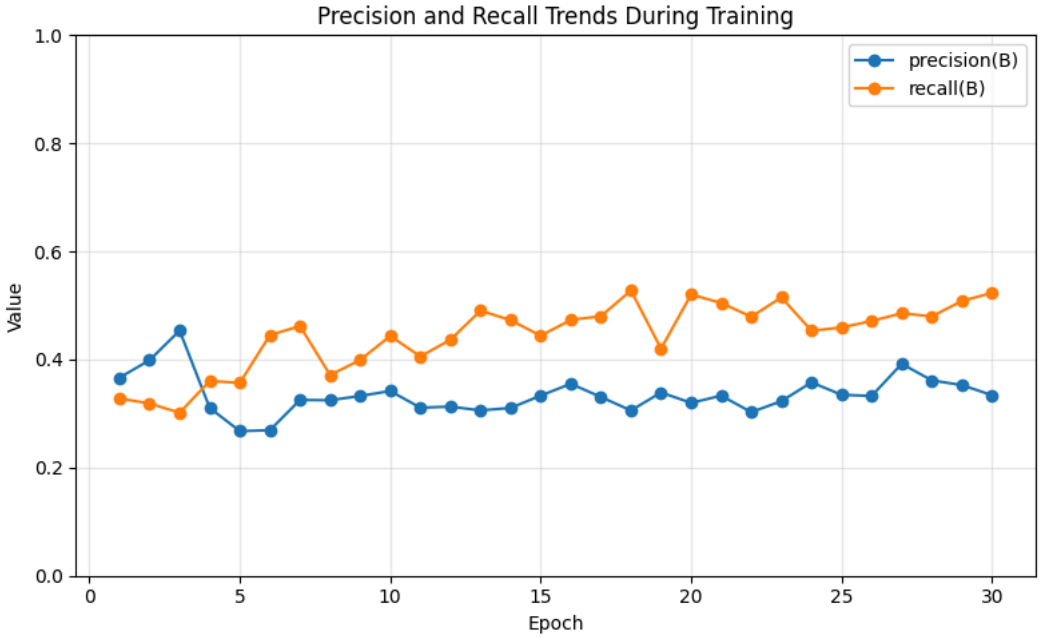


Precision and Recall over Epochs

```
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path

csv_path = sorted(Path("/content/runs/detect").rglob("results.csv"), key=lambda f: f.stat().st_mtime, reverse=Tru
df = pd.read_csv(csv_path)
print("Using:", csv_path)

plt.figure(figsize=(8,5))
```

```
plt.figure(figsize=(8,5))
for col in ["metrics/precision(B)", "metrics/recall(B)"]:
    if col in df.columns:
        plt.plot(df["epoch"], df[col], marker="o", label=col.split("/")[-1])
plt.xlabel("Epoch")
plt.ylabel("Value")
plt.title("Precision and Recall Trends During Training")
plt.legend()
plt.grid(True, alpha=0.3)
plt.ylim(0,1)
plt.tight_layout()
plt.show()
```

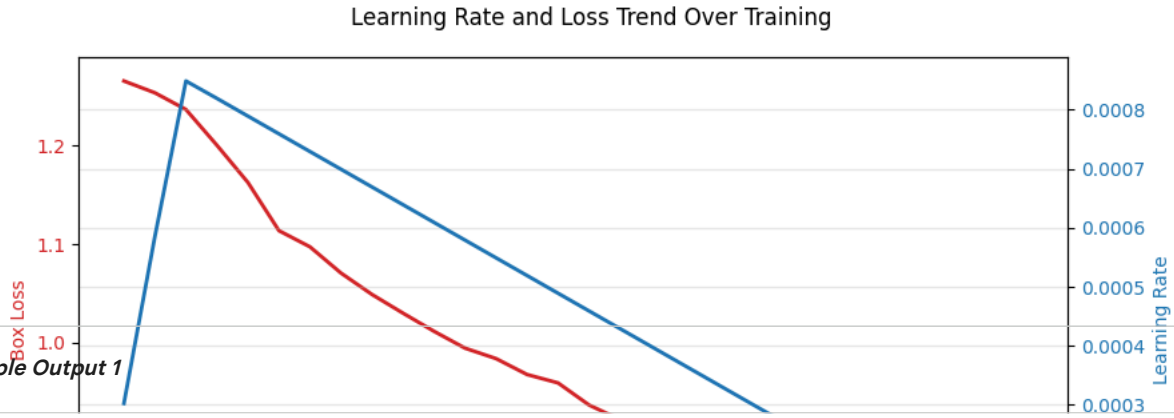
```
Using: /content/runs/detect/ODA_yolo11n_train/results.csv
```



Learning Rate and Fitness Trend

```
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path

csv_path = sorted(Path("/content/runs/detect").rglob("results.csv"), key=lambda f: f.stat().st_mtime, reverse=Tru
df = pd.read_csv(csv_path)
print("Using:", csv_path)

fig, ax1 = plt.subplots(figsize=(9,5))
if "train/box_loss" in df.columns:
    ax1.plot(df["epoch"], df["train/box_loss"], color="tab:red", label="Box Loss", linewidth=2)
ax1.set_xlabel("Epoch")
ax1.set_ylabel("Box Loss", color="tab:red")
ax1.tick_params(axis='y', labelcolor="tab:red")

if "lr/pg0" in df.columns:
    ax2 = ax1.twinx()
    ax2.plot(df["epoch"], df["lr/pg0"], color="tab:blue", label="Learning Rate", linewidth=2)
    ax2.set_ylabel("Learning Rate", color="tab:blue")
    ax2.tick_params(axis='y', labelcolor="tab:blue")

fig.suptitle("Learning Rate and Loss Trend Over Training")
fig.tight_layout()
plt.grid(True, alpha=0.3)
plt.show()
```

```
Using: /content/runs/detect/ODA_yolo11n_train/results.csv
```



Sample Output 1

```
from google.colab import files
from pathlib import Path
from ultralytics import YOLO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display

uploaded = files.upload()
if not uploaded:
    raise SystemExit("No file uploaded.")
fn = list(uploaded.keys())[0]
img_path = Path("/content")/fn

weight_candidates = list(Path("/content/runs/detect").rglob("*/weights/best.pt")) + list(Path("/content").rglob("
if not weight_candidates:
    raise SystemExit("No trained weights found.")
weight_candidates.sort(key=lambda p: p.stat().st_mtime, reverse=True)
w = str(weight_candidates[0])
print("Using weights:", w)

model = YOLO(w)
img = Image.open(img_path).convert("RGB")
res = model.predict(str(img_path), device="cpu", conf=0.25, iou=0.45, verbose=False)
pred_img = img.copy()
draw = ImageDraw.Draw(pred_img)
try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 14)
except:
    font = None

if res and len(res) and res[0].boxes is not None and len(res[0].boxes):
    xyxy = res[0].boxes.xyxy.cpu().numpy()
    cls = res[0].boxes.cls.cpu().numpy().astype(int)
    conf = res[0].boxes.conf.cpu().numpy()
    for (x1,y1,x2,y2), c, cf in zip(xyxy, cls, conf):
        draw.rectangle([x1,y1,x2,y2], outline="lime", width=max(2, int(min(img.size)/200)))
        label = f"{int(c)} {cf:.2f}"
        tx, ty = max(0, x1), max(0, y1-18)
        draw.rectangle([tx, ty, tx+len(label)*8+6, ty+18], fill="lime")
        draw.text((tx+3, ty+1), label, fill="black", font=font)
else:
    print("No predictions (on CPU).")

w, h = img.size
combined = Image.new("RGB", (w*2, h))
combined.paste(img, (0,0))
combined.paste(pred_img, (w,0))
display(combined.resize((900, int(900*h/(2*w)))))
```

Choose Files 00436.jpg

00436.jpg(image/jpeg) - 100847 bytes, last modified: 12/19/2021 - 100% done

Saving 00436.jpg to 00436.jpg

Using weights: /content/runs/detect/ODA_yolo11n_train/weights/best.pt

Sample Output 2

```
from google.colab import files
from pathlib import Path
from ultralytics import YOLO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display

uploaded = files.upload()
if not uploaded:
    raise SystemExit("No file uploaded.")
fn = list(uploaded.keys())[0]
img_path = Path("/content")/fn

weight_candidates = list(Path("/content/runs/detect").rglob("*/weights/best.pt")) + list(Path("/content").rglob(
if not weight_candidates:
    raise SystemExit("No trained weights found.")
weight_candidates.sort(key=lambda p: p.stat().st_mtime, reverse=True)
w = str(weight_candidates[0])
print("Using weights:", w)

model = YOLO(w)
img = Image.open(img_path).convert("RGB")
res = model.predict(str(img_path), device="cpu", conf=0.25, iou=0.45, verbose=False)
pred_img = img.copy()
draw = ImageDraw.Draw(pred_img)
try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 14)
except:
    font = None

if res and len(res) and res[0].boxes is not None and len(res[0].boxes):
    xyxy = res[0].boxes.xyxy.cpu().numpy()
    cls = res[0].boxes.cls.cpu().numpy().astype(int)
    conf = res[0].boxes.conf.cpu().numpy()
    for (x1,y1,x2,y2), c, cf in zip(xyxy, cls, conf):
        draw.rectangle([x1,y1,x2,y2], outline="lime", width=max(2, int(min(img.size)/200)))
        label = f"{int(c)} {cf:.2f}"
        tx, ty = max(0, x1), max(0, y1-18)
        draw.rectangle([tx, ty, tx+len(label)*8+6, ty+18], fill="lime")
        draw.text((tx+3, ty+1), label, fill="black", font=font)
else:
    print("No predictions (on CPU).")

w, h = img.size
combined = Image.new("RGB", (w*2, h))
combined.paste(img, (0,0))
combined.paste(pred_img, (w,0))
display(combined.resize((900, int(900*h/(2*w)))))
```

Choose Files 00074.jpg

00074.jpg(image/jpeg) - 72466 bytes, last modified: 12/19/2021 - 100% done

Saving 00074.jpg to 00074 (1).jpg

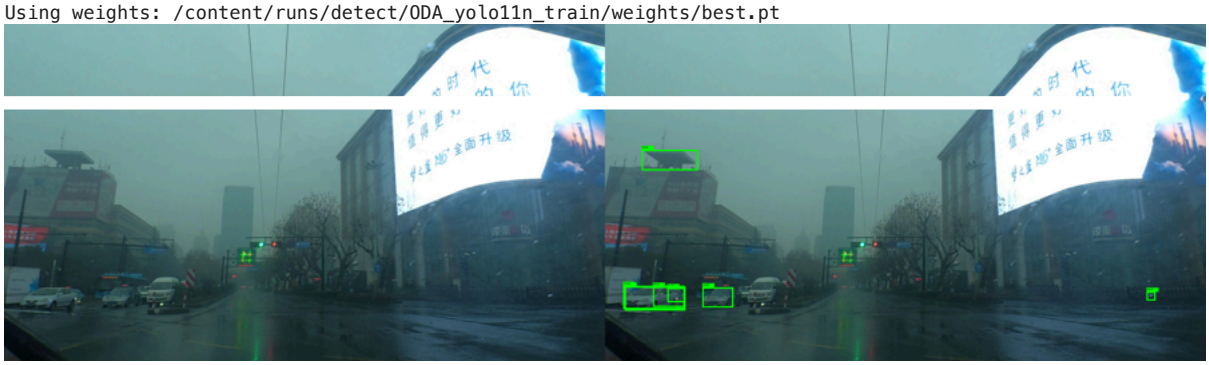
Using weights: /content/runs/detect/ODA_yolo11n_train/weights/best.pt

Sample Output 3

```
from google.colab import files
from pathlib import Path
from ultralytics import YOLO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display

uploaded = files.upload()
```

```
if not uploaded:
    raise SystemExit("No file uploaded.")
fn = list(uploaded.keys())[0]
img_path = Path("/content")/fn

weight_candidates = list(Path("/content/runs/detect").rglob("*/weights/best.pt")) + list(Path("/content").rglob(
if not weight_candidates:
    raise SystemExit("No trained weights found.")
weight_candidates.sort(key=lambda p: p.stat().st_mtime, reverse=True)
w = str(weight_candidates[0])
print("Using weights:", w)

model = YOLO(w)
img = Image.open(img_path).convert("RGB")
res = model.predict(str(img_path), device="cpu", conf=0.25, iou=0.45, verbose=False)
pred_img = img.copy()
draw = ImageDraw.Draw(pred_img)
try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 14)
except:
    font = None

if res and len(res) and res[0].boxes is not None and len(res[0].boxes):
    xyxy = res[0].boxes.xyxy.cpu().numpy()
    cls = res[0].boxes.cls.cpu().numpy().astype(int)
    conf = res[0].boxes.conf.cpu().numpy()
    for (x1,y1,x2,y2), c, cf in zip(xyxy, cls, conf):
        draw.rectangle([x1,y1,x2,y2], outline="lime", width=max(2, int(min(img.size)/200)))
        label = f"{int(c)} {cf:.2f}"
        tx, ty = max(0, x1), max(0, y1-18)
        draw.rectangle([tx, ty, tx+len(label)*8+6, ty+18], fill="lime")
        draw.text((tx+3, ty+1), label, fill="black", font=font)
else:
    print("No predictions (on CPU).")

w, h = img.size
combined = Image.new("RGB", (w*2, h))
combined.paste(img, (0,0))
combined.paste(pred_img, (w,0))
display(combined.resize((900, int(900*h/(2*w)))))
```

```
Choose Files 01656.jpg
01656.jpg(image/jpeg) - 231989 bytes, last modified: 12/19/2021 - 100% done
Saving 01656.jpg to 01656.jpg
Using weights: /content/runs/detect/ODA_yolo11n_train/weights/best.pt
```



Sample Output 4

```
from google.colab import files
from pathlib import Path
from ultralytics import YOLO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display

uploaded = files.upload()
if not uploaded:
    raise SystemExit("No file uploaded.")
fn = list(uploaded.keys())[0]
img_path = Path("/content")/fn

weight_candidates = list(Path("/content/runs/detect").rglob("*/weights/best.pt")) + list(Path("/content").rglob("
if not weight_candidates:
    raise SystemExit("No trained weights found.")
weight_candidates.sort(key=lambda p: p.stat().st_mtime, reverse=True)
w = str(weight_candidates[0])
print("Using weights:", w)

model = YOLO(w)
```

```
img = Image.open(img_path).convert("RGB")
res = model.predict(str(img_path), device="cpu", conf=0.25, iou=0.45, verbose=False)
pred_img = img.copy()
draw = ImageDraw.Draw(pred_img)
try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf", 14)
except:
    font = None

if res and len(res) and res[0].boxes is not None and len(res[0].boxes):
    xyxy = res[0].boxes.xyxy.cpu().numpy()
    cls = res[0].boxes.cls.cpu().numpy().astype(int)
    conf = res[0].boxes.conf.cpu().numpy()
    for (x1,y1,x2,y2), c, cf in zip(xyxy, cls, conf):
        draw.rectangle([x1,y1,x2,y2], outline="lime", width=max(2, int(min(img.size)/200)))
        label = f"{int(c)} {cf:.2f}"
        tx, ty = max(0, x1), max(0, y1-18)
        draw.rectangle([tx, ty, tx+len(label)*8+6, ty+18], fill="lime")
        draw.text((tx+3, ty+1), label, fill="black", font=font)
else:
    print("No predictions (on CPU).")

w, h = img.size
combined = Image.new("RGB", (w*2, h))
combined.paste(img, (0,0))
combined.paste(pred_img, (w,0))
display(combined.resize((900, int(900*h/(2*w)))))
```

```
Choose Files 01850.jpg
01850.jpg(image/jpeg) - 211519 bytes, last modified: 12/19/2021 - 100% done
Saving 01850.jpg to 01850.jpg
Using weights: /content/runs/detect/ODA_yolo11n_train/weights/best.pt
```

```
from ultralytics import YOLO
from pathlib import Path

weights_candidates = list(Path("/content/runs/detect").rglob("*/weights/best.pt")) + list(Path("/content").rglob("
if not weights_candidates:
    raise SystemExit("No trained weights found.")
weights_candidates.sort(key=lambda p: p.stat().st_mtime, reverse=True)
wpath = str(weights_candidates[0])
print("Using weights:", wpath)

model = YOLO(wpath)
export_path = model.export(format="onnx", opset=12, simplify=True, dynamic=False, imgsz=640)

print("\nExport complete!")
print("ONNX file saved at:", export_path)
```

```
Using weights: /content/runs/detect/ODA_yolo11n_train/weights/best.pt
Ultralytics 8.3.227 🚀 Python-3.12.12 torch-2.8.0+cu126 CPU (Intel Xeon CPU @ 2.20GHz)
YOLO11n summary (fused): 100 layers, 2,583,517 parameters, 0 gradients, 6.3 GFLOPs

PyTorch: starting from '/content/runs/detect/ODA_yolo11n_train/weights/best.pt' with input shape (1, 3, 640, 640)

ONNX: starting export with onnx 1.19.1 opset 12...
ONNX: slimming with onnxslim 0.1.74...
ONNX: export success ✅ 1.2s, saved as '/content/runs/detect/ODA_yolo11n_train/weights/best.onnx' (10.1 MB)

Export complete (1.7s)
Results saved to /content/runs/detect/ODA_yolo11n_train/weights
Predict:         yolo predict task=detect model=/content/runs/detect/ODA_yolo11n_train/weights/best.onnx imgsz=64
Validate:        yolo val task=detect model=/content/runs/detect/ODA_yolo11n_train/weights/best.onnx imgsz=640 da
Visualize:       https://netron.app

Export complete!
ONNX file saved at: /content/runs/detect/ODA_yolo11n_train/weights/best.onnx
```