YOLO (You Only Look Once) 모델

구성

---

1) YOLO 이론

1.1 객체 탐지 개념과 YOLO 시리즈

1.1.1 객체 탐지(OD)

- 이미지를 입력받아, **물체 위치(바운딩 박스)** + **클래스**를 예측.
- **2-Stage**(R-CNN 계열) vs **1-Stage**(YOLO, SSD): YOLO는 실시간성에 강점.

1.1.2 YOLO 시리즈 (v1 ~ v8)

버전	연도	Backbone	주요 특징
v1	2016	Darknet-19	빠름, but 작은 물체 취약
v2	2017	Darknet-19	Anchor Box, BN, Multi-Scale, 속도/정확도 모두 개선
v3	2018	Darknet-53	FPN 3-스케일, 잔차 구조, 작은 물체 대응 강화
v4	2020	CSPDarknet53	Mosaic, SPP, Mish, Bag-of-Freebies, 성능↑
v5	2020+	PyTorch impl	AutoAnchor, 다양한 size(n,s,m,l,x), Colab 친화, 하이퍼파라미터 자동화
v8	2023+	Anchor-free	Detection+Classification+Segmentation, Pythonic API, TTA 등

1.2 Anchor 기반 vs Anchor-free

- **Anchor 기반**: v2~v5 → 사전에 정의된 (w,h)로부터 박스 편차 회귀.
- **Anchor-free**: v8 등, grid cell이 직접 (cx,cy,w,h) 예측.

1.3 YOLOv8 구조

- **Backbone**: C2f/C3(CSP 변형), SPPF.
- **Neck**: FPN + PANet.
- **Head**: anchor-free, IoU/L1.

1.4 손실 함수

- BBox: IoU 기반(CIoU, DIoU, GIoU). 예) CIoU:

$$\mathrm{CIoU}(\hat{B}, B) = IoU(\hat{B}, B) - \frac{\rho^2(\hat{c}, c)}{d^2} - \alpha\nu.$$

- Confidence: BCE or Focal.
- Class: CE / BCE / Focal.

1.5 Bag-of-Freebies, Mosaic, Copy-Paste

- **Mosaic**: 4장 이미지 합성.
- **Copy-Paste**: 객체를 잘라 다른 배경에 붙이는 증강.
- **Focal Loss**: class imbalance 개선.
- **AutoAnchor**: anchor-based 모델에서 anchor 자동 설정.

---

2) 실습 환경 세팅 & GPU 확인

Colab에서 GPU(T4) 사용 가정.

```
!pip install ultralytics opencv-python matplotlib --upgrade
import torch
```

```
print("Torch version:", torch.__version__)
!nvidia-smi
print("[Setup Complete]")
```

```
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.58.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.4->ultralytics) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.4->ultralytics) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.23.0->ultralytics) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.23.0->ultralytics) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.23.0->ultralytics) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.23.0->ultralytics) (2025.4.26)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (4.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (2025.3.2)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (12.
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (1
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (12.
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (9.1.0.70
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (12.4.5.
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (11.2.1.3
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (10.3.
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (11.6.
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (12.
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (12.4.127)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (12.4
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=1.8.0->ultralytics) (1.
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.8.0->ultralytics) (3.0.2)
Torch version: 2.6.0+cu124
Tue May 27 22:38:34 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. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |                    0 |
| N/A   77C    P0             35W /   70W |     286MiB /  15360MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
+-----------------------------------------------------------------------------------------+
[Setup Complete]
```

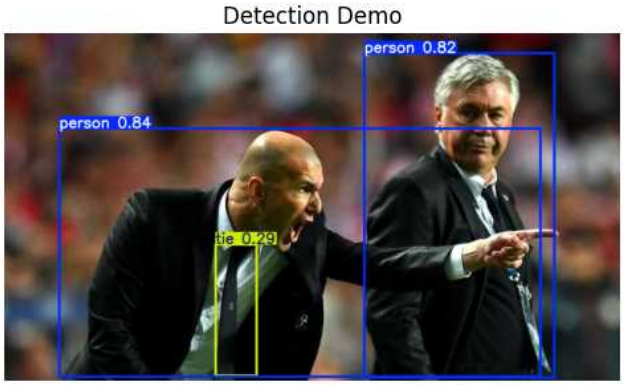
3) Detection / Classification / Segmentation 예시 (상세 코드)

3.1 Detection

```
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Download an example image
!wget -q -O example_det.jpg https://ultralytics.com/images/zidane.jpg

# 1) Load detection model
model_det = YOLO('yolov8n.pt')

# 2) Inference
results_det = model_det.predict(
    source='example_det.jpg',
    conf=0.25,
    device=0
)
```

```
# 3) Visualization
res_det0 = results_det[0]
res_img = res_det0.plot()  # draw boxes

plt.figure(figsize=(6,4))
plt.imshow(res_img[..., ::-1])
plt.axis('off')
plt.title("Detection Demo")
plt.show()
print("[Detection Results]")
print(res_det0.boxes)
```

```
image 1/1 /content/example_det.jpg: 384x640 2 persons, 1 tie, 72.8ms
Speed: 3.6ms preprocess, 72.8ms inference, 2.7ms postprocess per image at shape (1, 3, 384, 640)
```




```
[Detection Results]
ultralytics.engine.results.Boxes object with attributes:

cls: tensor([ 0.,  0., 27.], device='cuda:0')
conf: tensor([0.8360, 0.8190, 0.2910], device='cuda:0')
data: tensor([[1.1487e+02, 1.9741e+02, 1.1145e+03, 7.1189e+02, 8.3597e-01, 0.0000e+00],
        [7.4846e+02, 4.1855e+01, 1.1431e+03, 7.1302e+02, 8.1896e-01, 0.0000e+00],
        [4.3947e+02, 4.3707e+02, 5.2435e+02, 7.0916e+02, 2.9097e-01, 2.7000e+01]], device='cuda:0')
id: None
is_track: False
orig_shape: (720, 1280)
shape: torch.Size([3, 6])
xywh: tensor([[614.6694, 454.6507, 999.5989, 514.4774],
        [945.7686, 377.4395, 394.6143, 671.1688],
        [481.9103, 573.1166,  84.8712, 272.0862]], device='cuda:0')
xywhn: tensor([[0.4802, 0.6315, 0.7809, 0.7146],
        [0.7389, 0.5242, 0.3083, 0.9322],
        [0.3765, 0.7960, 0.0663, 0.3779]], device='cuda:0')
xyxy: tensor([[ 114.8700,  197.4120, 1114.4689,  711.8894],
        [ 748.4614,   41.8552, 1143.0757,  713.0239],
        [ 439.4747,  437.0735,  524.3459,  709.1597]], device='cuda:0')
xyxyn: tensor([[0.0897, 0.2742, 0.8707, 0.9887],
        [0.5847, 0.0581, 0.8930, 0.9903],
        [0.3433, 0.6070, 0.4096, 0.9849]], device='cuda:0')
```

3.2 Classification

```
# Classification example
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os

model_cls = YOLO('yolov8n-cls.pt')

!wget -q -O example_cls.jpg https://ultralytics.com/images/bus.jpg

# Add a check to see if the file exists and is readable
if not os.path.exists('example_cls.jpg'):
    print("[ERROR] example_cls.jpg was not downloaded successfully.")
elif os.path.getsize('example_cls.jpg') == 0:
    print("[ERROR] example_cls.jpg was downloaded but is empty.")
else:
    print("[INFO] example_cls.jpg downloaded successfully.")
    results_cls = model_cls.predict(
        source='example_cls.jpg',
        device=0
    )

    cls0 = results_cls[0]
```

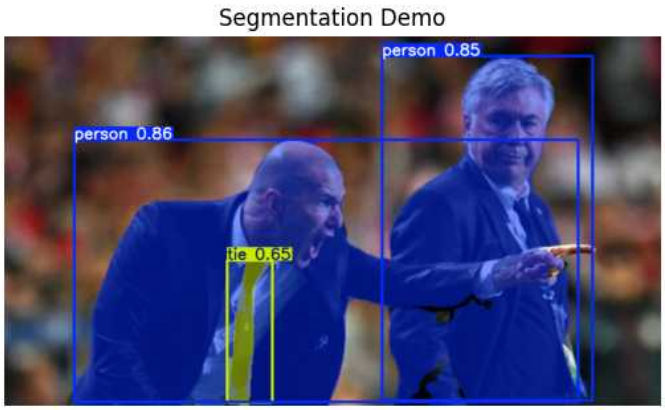
```
# Access top 5 indices and confidence directly from the Probs object
top5_indices = cls0.probs.top5
top5_confs = cls0.probs.top5conf

print("[Classification Top-5]")
# Iterate through the top 5 results
for cls_id, score in zip(top5_indices, top5_confs):
    # cls_id is already an int, no need for .item()
    print(f"  {model_cls.names[cls_id]}: {score.item():.4f}")

# show input image
img_c = cv2.imread('example_cls.jpg')
img_c_rgb = cv2.cvtColor(img_c, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(img_c_rgb)
plt.axis('off')
plt.title("Classification Input")
plt.show()
```

```
[INFO] example_cls.jpg downloaded successfully.

image 1/1 /content/example_cls.jpg: 224x224 minibus 0.50, police_van 0.29, trolleybus 0.05, golfcart 0.02, jinrikisha 0.02, 4.6ms
Speed: 13.4ms preprocess, 4.6ms inference, 0.1ms postprocess per image at shape (1, 3, 224, 224)
[Classification Top-5]
  minibus: 0.5049
  police_van: 0.2934
  trolleybus: 0.0497
  golfcart: 0.0185
  jinrikisha: 0.0177
```

3.3 Segmentation

```
# Segmentation example
model_seg = YOLO('yolov8n-seg.pt')
!wget -q -O example_seg.jpg https://ultralytics.com/images/zidane.jpg

results_seg = model_seg.predict(
    source='example_seg.jpg',
    conf=0.25,
    device=0
)

seg0 = results_seg[0]
seg_img = seg0.plot()
plt.figure()
plt.imshow(seg_img[..., ::-1])
plt.axis('off')
plt.title("Segmentation Demo")
plt.show()
print("[Segmentation Results]")
print("Masks:", seg0.masks.data.shape if seg0.masks is not None else None)
```

```
image 1/1 /content/example_seg.jpg: 384x640 2 persons, 1 tie, 81.7ms
Speed: 2.4ms preprocess, 81.7ms inference, 12.5ms postprocess per image at shape (1, 3, 384, 640)
```




```
[Segmentation Results]
Masks: torch.Size([3, 384, 640])
```

4) COCO Demo: Training, Validation, Inference, Export (고급 코드)

4.1 COCO val 다운로드

```
import torch
torch.hub.download_url_to_file(
    'https://ultralytics.com/assets/coco2017val.zip',
    'tmp.zip'
)
!unzip -q tmp.zip -d datasets && rm tmp.zip
print("[INFO] COCO val dataset unzipped at ./datasets/coco2017val.")
```

```
100%|██████████| 780M/780M [00:20<00:00, 40.8MB/s]
replace datasets/coco/.DS_Store? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
[INFO] COCO val dataset unzipped at ./datasets/coco2017val.
```

4.2 data.yaml

```
import os
import yaml

data_yaml = {
    'train': '/content/datasets/coco/images/val2017',
    'val': '/content/datasets/coco/images/val2017',
    'nc': 80,
    'names': [
        'person','bicycle','car','motorcycle','airplane','bus','train','truck','boat','traffic light',
        'fire hydrant','stop sign','parking meter','bench','bird','cat','dog','horse','sheep','cow',
        'elephant','bear','zebra','giraffe','backpack','umbrella','handbag','tie','suitcase','frisbee',
        'skis','snowboard','sports ball','kite','baseball bat','baseball glove','skateboard','surfboard',
        'tennis racket','bottle','wine glass','cup','fork','knife','spoon','bowl','banana','apple',
        'sandwich','orange','broccoli','carrot','hot dog','pizza','donut','cake','chair','couch','potted plant',
        'bed','dining table','toilet','tv','laptop','mouse','remote','keyboard','cell phone','microwave',
        'oven','toaster','sink','refrigerator','book','clock','vase','scissors','teddy bear','hair drier','toothbrush'
    ]
}

os.makedirs('coco_demo', exist_ok=True)
with open('./coco_demo/data.yaml', 'w') as f:
    yaml.dump(data_yaml, f)
print("[INFO] data.yaml created under ./coco_demo/data.yaml")
```

```
[INFO] data.yaml created under ./coco_demo/data.yaml
```

4.3 Training (epochs=1, 고급 옵션)

```
from ultralytics import YOLO

model_coco = YOLO('yolov8n.pt')

results_train = model_coco.train(
```

```
    data='./coco_demo/data.yaml',
    epochs=1,
    imgsz=640,
    batch=8,
    name='yolov8n_coco_demo',
    amp=True,          # AMP
    lr0=1e-2,          # init LR
    lrf=0.01,          # final LR factor
    mosaic=1.0,        # mosaic
    copy_paste=0.5,    # copy-paste prob
    device=0
)
print("[INFO] Training done. Check runs/detect/yolov8n_coco_demo/")
```

```
Ultralytics 8.3.145 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment, batch=8, bgr=0.0, box=7.5, cache=False, cfg=None, clas
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100%|██████████| 755k/755k [00:00<00:00, 143MB/s]
                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    897664  ultralytics.nn.modules.head.Detect           [80, [64, 128, 256]]

Model summary: 129 layers, 3,157,200 parameters, 3,157,184 gradients, 8.9 GFLOPs

Transferred 355/355 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 to 'yolo11n.pt'...
100%|██████████| 5.35M/5.35M [00:00<00:00, 399MB/s]
AMP: checks passed ✅
train: Fast image access ✅ (ping: 0.0±0.0 ms, read: 2491.2±604.7 MB/s, size: 139.7 KB)
train: Scanning /content/datasets/coco/labels/val2017... 4952 images, 48 backgrounds, 0 corrupt: 100%|██████████| 5000/5000 [00:02<00:00, 2381.
train: New cache created: /content/datasets/coco/labels/val2017.cache
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, method='weighted_average', num_output_c
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 1111.6±1067.1 MB/s, size: 143.4 KB)
val: Scanning /content/datasets/coco/labels/val2017.cache... 4952 images, 48 backgrounds, 0 corrupt: 100%|██████████| 5000/5000 [00:00<?, ?it/s
Plotting labels to runs/detect/yolov8n_coco_demo27/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatical
optimizer: AdamW(lr=0.000119, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/yolov8n_coco_demo27
Starting training for 1 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        1/1      1.47G      1.235      1.573      1.272         90        640: 100%|██████████| 625/625 [01:49<00:00,  5.71it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 313/313 [00:57<00:00,  5.47it/s]
                   all       5000      36335      0.612       0.48      0.513      0.364

1 epochs completed in 0.048 hours.
Optimizer stripped from runs/detect/yolov8n_coco_demo27/weights/last.pt, 6.5MB
```

4.4 Validation

```
metrics = model_coco.val(
    data='coco_demo/data.yaml',
)
print("[INFO] Validation metrics:")
print("   mAP@0.5:", metrics.box.map50)
print("   mAP@0.5:0.95:", metrics.box.map)
```

```
Ultralytics 8.3.145 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 2498.9±982.8 MB/s, size: 123.3 KB)
val: Scanning /content/datasets/coco/labels/val2017.cache... 4952 images, 48 backgrounds, 0 corrupt: 100%|██████████| 5000/5000 [00:00<?, ?it/s
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 625/625 [00:50<00:00, 12.32it/s]
```

all	5000	36335	0.612	0.48	0.513	0.364
person	2693	10777	0.802	0.627	0.737	0.507
bicycle	149	314	0.695	0.334	0.425	0.25
car	535	1918	0.683	0.492	0.559	0.358
motorcycle	159	367	0.578	0.616	0.651	0.404
airplane	97	143	0.734	0.755	0.824	0.645
bus	189	283	0.735	0.643	0.714	0.587
train	157	190	0.874	0.768	0.826	0.625
truck	250	414	0.384	0.49	0.414	0.276
boat	121	424	0.6	0.316	0.397	0.218
traffic light	191	634	0.549	0.374	0.398	0.2
fire hydrant	86	101	0.854	0.703	0.76	0.61
stop sign	69	75	0.657	0.653	0.7	0.631
parking meter	37	60	0.578	0.583	0.594	0.462
bench	235	411	0.531	0.265	0.284	0.188
bird	125	427	0.705	0.351	0.426	0.279
cat	184	202	0.761	0.827	0.836	0.647
dog	177	218	0.739	0.665	0.716	0.578
horse	128	272	0.725	0.614	0.697	0.521
sheep	65	354	0.489	0.726	0.663	0.46
cow	87	372	0.722	0.605	0.655	0.456
elephant	89	252	0.715	0.827	0.805	0.612
bear	49	71	0.75	0.831	0.845	0.688
zebra	85	266	0.619	0.85	0.854	0.639
giraffe	101	232	0.811	0.832	0.881	0.686
backpack	228	371	0.569	0.113	0.204	0.106
umbrella	174	407	0.63	0.506	0.549	0.362
handbag	292	540	0.399	0.113	0.144	0.0744
tie	145	252	0.688	0.349	0.423	0.262
suitcase	105	299	0.497	0.435	0.449	0.311
frisbee	84	115	0.72	0.765	0.764	0.578
skis	120	241	0.667	0.307	0.37	0.189
snowboard	49	69	0.706	0.304	0.358	0.248
sports ball	169	260	0.679	0.432	0.468	0.325
kite	91	327	0.641	0.526	0.579	0.391
baseball bat	97	145	0.519	0.364	0.364	0.22
baseball glove	100	148	0.587	0.486	0.499	0.296
skateboard	127	179	0.747	0.592	0.653	0.436
surfboard	149	267	0.564	0.479	0.485	0.3
tennis racket	167	225	0.638	0.64	0.649	0.395
bottle	379	1013	0.579	0.398	0.447	0.294
wine glass	110	341	0.563	0.397	0.408	0.258
cup	390	895	0.579	0.428	0.482	0.337
fork	155	215	0.571	0.279	0.361	0.243
knife	181	325	0.465	0.151	0.171	0.1
spoon	153	253	0.377	0.127	0.159	0.0943
bowl	314	623	0.619	0.456	0.501	0.377
banana	103	370	0.543	0.297	0.361	0.224
apple	76	236	0.297	0.314	0.251	0.182
sandwich	98	177	0.535	0.417	0.451	0.33
orange	85	285	0.373	0.498	0.345	0.259
broccoli	71	312	0.429	0.407	0.341	0.183
carrot	81	365	0.407	0.272	0.221	0.21

4.5 Inference

```
!wget -q -O demo_infer.jpg https://ultralytics.com/images/zidane.jpg
res_infer = model_coco.predict(
    source='demo_infer.jpg',
    conf=0.25,
    device=0,
    save=True
)
print("[INFO] Inference done.")
print(res_infer)
print("Check runs/detect/predictN folder.")
```

```
image 1/1 /content/demo_infer.jpg: 384x640 2 persons, 2 ties, 10.2ms
Speed: 2.2ms preprocess, 10.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
Results saved to runs/detect/yolov8n_coco_demo276
[INFO] Inference done.
[ultralytics.engine.results.Results object with attributes:

boxes: ultralytics.engine.results.Boxes object
keypoints: None
masks: None
names: {0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10
obb: None
orig_img: array([[[44, 51, 76],
        [43, 50, 75],
        [41, 48, 73],
        ...,
        [20, 18, 54],
        [18, 16, 52],
        [17, 15, 51]],
```

```
          [[44, 51, 76],
           [43, 50, 75],
           [41, 48, 73],
           ...,
           [20, 18, 54],
           [18, 16, 52],
           [18, 16, 52]],

          [[44, 51, 76],
           [43, 50, 75],
           [41, 48, 73],
           ...,
           [21, 18, 57],
           [19, 16, 55],
           [18, 15, 54]],

          ...,

          [[53, 44, 40],
           [52, 43, 39],
           [51, 42, 38],
           ...,
           [50, 50, 38],
           [51, 51, 39],
           [52, 52, 40]],

          [[53, 44, 40],
           [52, 43, 39],
           [51, 42, 38],
           ...,
           [50, 50, 38],
           [51, 51, 39],
           [52, 52, 40]],

          [[53, 44, 40],
           [52, 43, 39],
           [51, 42, 38]
```

4.6 Export

```
export_formats = ['onnx']
for fmt in export_formats:
    print(f"Exporting to {fmt}...")
    model_coco.export(format=fmt)
print("[INFO] Export complete.")
```

```
Exporting to onnx...
Ultralytics 8.3.145 🚀 Python-3.11.12 torch-2.6.0+cu124 CPU (Intel Xeon 2.20GHz)

PyTorch: starting from 'runs/detect/yolov8n_coco_demo27/weights/best.pt' with input shape (1, 3, 640, 640) BCHW and output shape(s) (1, 84, 8400)

ONNX: starting export with onnx 1.17.0 opset 19...
ONNX: slimming with onnxslim 0.1.53...
ONNX: export success ✅ 1.5s, saved as 'runs/detect/yolov8n_coco_demo27/weights/best.onnx' (12.2 MB)

Export complete (1.9s)
Results saved to /content/runs/detect/yolov8n_coco_demo27/weights
Predict:         yolo predict task=detect model=runs/detect/yolov8n_coco_demo27/weights/best.onnx imgsz=640
Validate:        yolo val task=detect model=runs/detect/yolov8n_coco_demo27/weights/best.onnx imgsz=640 data=./coco_demo/data.yaml
Visualize:       https://netron.app
[INFO] Export complete.
```

5) 하이퍼파라미터 & Data Augmentation

5.1 Mosaic

- 4장 이미지를 합침.

```
model_coco.train(mosaic=1.0)
```

5.2 Copy-Paste

```
model_coco.train(copy_paste=0.5)
```

5.3 Focal Loss

- (Experimental) Class imbalance 클 때.

5.4 AutoAnchor

- anchor-based(YOLOv5)에서 anchor 자동 계산.

---

6) 고급 후처리 (Soft-NMS, TTA 등)

6.1 Soft-NMS

- Ultralytics에는 기본 미포함. 별도 구현.

6.2 TTA(Test-Time Augmentation)

```
res = model_coco.predict('demo_infer.jpg', tta=True)
```

---

7) Q&A & 마무리

Q1) 학습이 너무 느려

- imgsz, batch 줄이거나 AMP/multi-GPU 사용.

Q2) mAP가 NaN

- LR가 너무 큼, Annotation 이슈.

Q3) Anchor-free vs Anchor-based?

- Anchor-free(YOLOv8) 구조 단순, anchor tuning 필요 X.
- Anchor-based(YOLOv5)도 안정성, 전통성.

Q4) Export 후 성능 차이?

- ONNX/TensorRT 등에서 INT8 최적화 가능, 정확도 ↔ 속도 절충.

결론

- **YOLO**(특히 YOLOv8) 이론 + **COCO** 데모 + **고급**(Mosaic, Copy-Paste, TTA, Focal) 모두 시연.
- 실제론 epochs↑ + COCO train(118k) + 하이퍼파라미터 튜닝.
- Export → 배포.

감사합니다!