yolo8 모델 성능 개선

In [ ]:
```
!pip install ultralytics
```

```
Collecting ultralytics
  Downloading ultralytics-8.3.17-py3-none-any.whl.metadata (34 kB)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.10/dist-pa
ckages (from ultralytics) (1.26.4)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dis
t-packages (from ultralytics) (3.7.1)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/
dist-packages (from ultralytics) (4.10.0.84)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-pa
ckages (from ultralytics) (10.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-pa
ckages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist
-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-pac
kages (from ultralytics) (1.13.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-pac
kages (from ultralytics) (2.4.1+cu121)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/di
st-packages (from ultralytics) (0.19.1+cu121)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-pac
kages (from ultralytics) (4.66.5)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages
(from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packa
ges (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-pa
ckages (from ultralytics) (2.2.2)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-
packages (from ultralytics) (0.13.2)
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.9-py3-none-any.whl.metadata (9.3 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist
-packages (from matplotlib>=3.3.0->ultralytics) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-pac
kages (from matplotlib>=3.3.0->ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib>=3.3.0->ultralytics) (4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib>=3.3.0->ultralytics) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib>=3.3.0->ultralytics) (24.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist
-packages (from matplotlib>=3.3.0->ultralytics) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/
dist-packages (from matplotlib>=3.3.0->ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pac
kages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-p
ackages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pac
kages (from requests>=2.23.0->ultralytics) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/di
st-packages (from requests>=2.23.0->ultralytics) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/di
st-packages (from requests>=2.23.0->ultralytics) (2024.8.30)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-package
s (from torch>=1.8.0->ultralytics) (3.16.1)
```

```
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python
3.10/dist-packages (from torch>=1.8.0->ultralytics) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch>=1.8.0->ultralytics) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-package
s (from torch>=1.8.0->ultralytics) (3.4.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.8.0->ultralytics) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch>=1.8.0->ultralytics) (2024.6.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-package
s (from python-dateutil>=2.7->matplotlib>=3.3.0->ultralytics) (1.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-
packages (from jinja2->torch>=1.8.0->ultralytics) (3.0.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/di
st-packages (from sympy->torch>=1.8.0->ultralytics) (1.3.0)
Downloading ultralytics-8.3.17-py3-none-any.whl (876 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 876.6/876.6 kB 40.7 MB/s eta 0:00:00
Downloading ultralytics_thop-2.0.9-py3-none-any.whl (26 kB)
Installing collected packages: ultralytics-thop, ultralytics
Successfully installed ultralytics-8.3.17 ultralytics-thop-2.0.9
```

In [ ]:
```python
from ultralytics import YOLO
import cv2                                          # OpenCV 라이브러리를 불러오기
from google.colab.patches import cv2_imshow      # from google.colab.patches impo

# YOLOv8 모델 로드
model = YOLO('yolov8n.pt')  # YOLOv8의 경량 모델

# 이미지 로드
img = cv2.imread('Dog_rawPixel01.jpg')

# 객체 탐지
# 로드된 YOLOv8 모델을 사용하여 이미지에서 객체를 탐지합니다. results 변수에는 탐
results = model(img)

# 결과 시각화
# plot() 함수는 탐지된 객체 주변에 경계 상자를 그려 이미지를 반환
img_with_detections = results[0].plot()  # 첫 번째 결과를 시각화

# 결과 이미지 표시
cv2_imshow(img_with_detections)
```

Output hidden; open in https://colab.research.google.com to view.

다른 이미지 확인

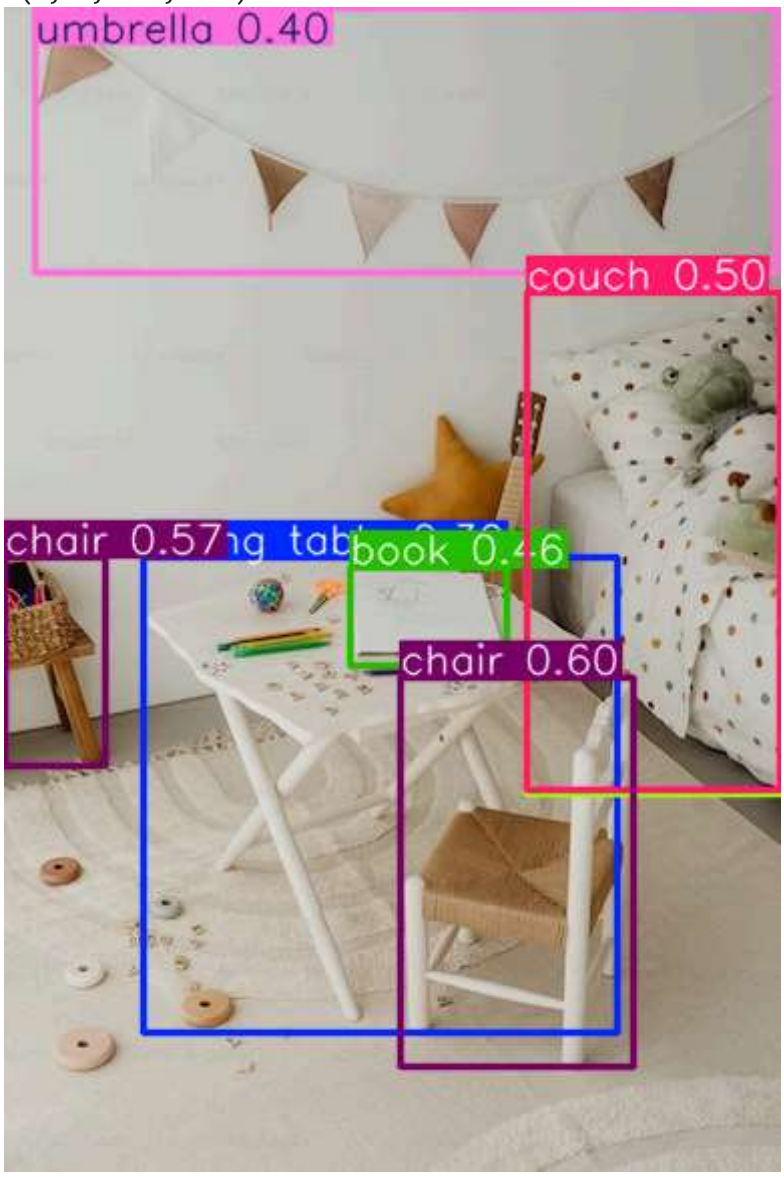
In [ ]:
```python
# 이미지 로드
img = cv2.imread('life_unsplash.jpeg')

# 객체 탐지
# 로드된 YOLOv8 모델을 사용하여 이미지에서 객체를 탐지합니다. results 변수에는 탐
results = model(img)

# 결과 시각화
# plot() 함수는 탐지된 객체 주변에 경계 상자를 그려 이미지를 반환
img_with_detections = results[0].plot()  # 첫 번째 결과를 시각화
```

```
# 결과 이미지 표시
cv2_imshow(img_with_detections)
```

```
0: 640x448 1 umbrella, 2 chairs, 1 couch, 1 bed, 1 dining table, 1 book, 39.6ms
Speed: 2.9ms preprocess, 39.6ms inference, 1.4ms postprocess per image at shape
(1, 3, 640, 448)
```



성능 개선 - 더 좋은 모델 사용

In [ ]:
```
# YOLO8의 주요 모델 크기는 다음과 같습니다 (작은 것부터 큰 순서로):

# YOLOv8n (nano)
# YOLOv8s (small)
# YOLOv8m (medium)
# YOLOv8l (large)
# YOLOv8x (extra large)
```

In [ ]:
```
from ultralytics import YOLO
import cv2                                      # OpenCV 라이브러리를 불러오기
from google.colab.patches import cv2_imshow     # from google.colab.patches impo

# YOLOv8 모델 로드
model = YOLO('yolov8l.pt')  # YOLOv8의 경량 모델

# 이미지 로드
```

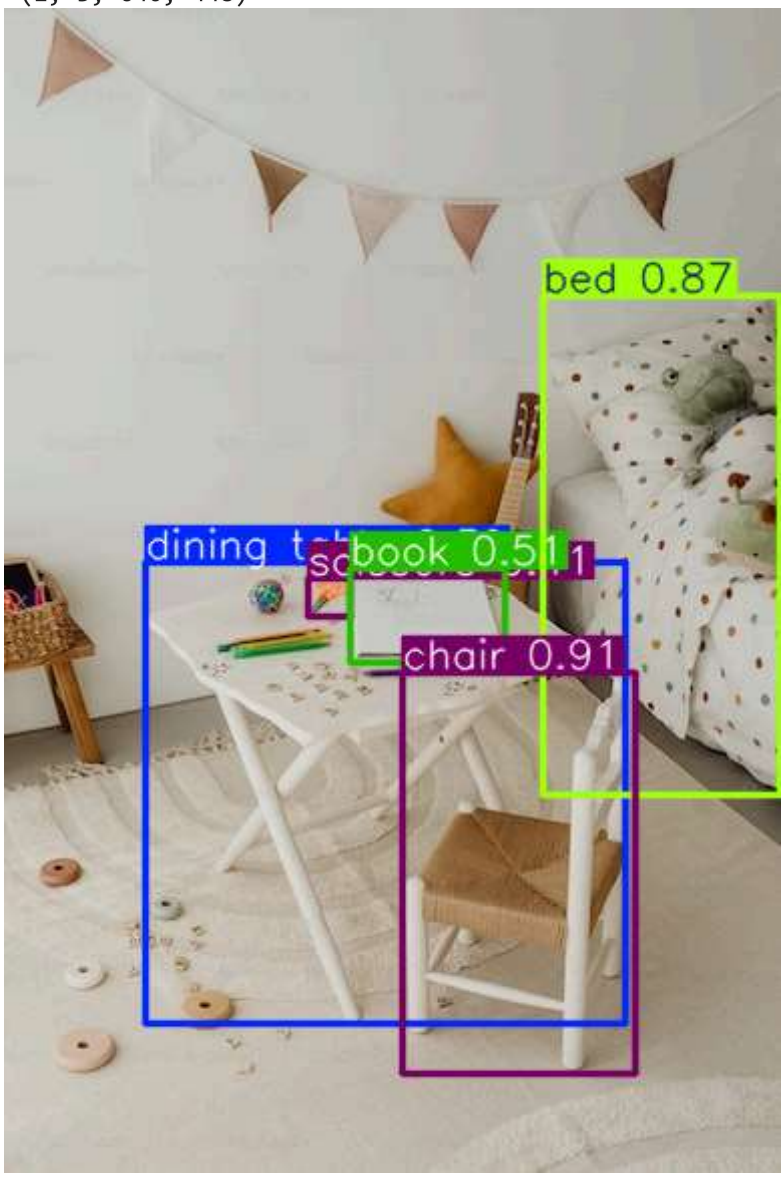
```
img = cv2.imread('life_unsplash.jpeg')

# 객체 탐지
# 로드된 YOLOv8 모델을 사용하여 이미지에서 객체를 탐지합니다. results 변수에는 탐
results = model(img)

# 결과 시각화
# plot() 함수는 탐지된 객체 주변에 경계 상자를 그려 이미지를 반환
img_with_detections = results[0].plot()  # 첫 번째 결과를 시각화

# 결과 이미지 표시
cv2_imshow(img_with_detections)
```

```
0: 640x448 1 chair, 1 bed, 1 dining table, 1 book, 1 scissors, 48.9ms
Speed: 2.2ms preprocess, 48.9ms inference, 1.9ms postprocess per image at shape
(1, 3, 640, 448)
```



COCO128 데이터 셋을 이용한 모델 성능 평가

데이터 준비

In [ ]:
```
!rm -rf /content/datasets
!rm -rf /content/coco128
!rm -rf /content/runs
```

데이터 다운로드 및 압축 풀기

In [ ]:
```
# 필요한 모듈 임포트
from ultralytics import YOLO
from ultralytics.utils.downloads import download
import os
import zipfile

# COCO128 데이터셋 다운로드
download(url='https://ultralytics.com/assets/coco128.zip')

# 다운로드된 파일 확인
print("Downloaded files:", os.listdir())

# ZIP 파일 압축 해제
with zipfile.ZipFile('coco128.zip', 'r') as zip_ref:
    zip_ref.extractall('.')

# 압축 해제 후 파일 확인
print("Files after extraction:", os.listdir())
print("Contents of coco128 folder:", os.listdir('coco128'))
```

```
Unzipping /content/coco128.zip to /content/coco128...: 100%|██████████| 263/263
[00:00<00:00, 3458.06file/s]
Downloaded files: ['.config', 'coco128.zip', 'yolov8_coco128_trained_custom_impro
ved.pt', 'yolov8s.pt', 'coco128_custom.yaml', 'yolov8_transfer_learning_50epoch.p
t', 'Dog_rawPixel01.jpg', 'yolov8x.pt', 'life_unsplash.jpeg', 'yolov8l.pt', 'coco
128', 'yolov8_coco128_trained_custom.pt', 'yolo11n.pt', 'yolov8n.pt', 'ferrari.jp
g', 'sample_data']
Files after extraction: ['.config', 'coco128.zip', 'yolov8_coco128_trained_custom
_improved.pt', 'yolov8s.pt', 'coco128_custom.yaml', 'yolov8_transfer_learning_50e
poch.pt', 'Dog_rawPixel01.jpg', 'yolov8x.pt', 'life_unsplash.jpeg', 'yolov8l.pt',
'coco128', 'yolov8_coco128_trained_custom.pt', 'yolo11n.pt', 'yolov8n.pt', 'ferra
ri.jpg', 'sample_data']
Contents of coco128 folder: ['LICENSE', 'labels', 'README.txt', 'images']
```

In [ ]:
```
import cv2                                          # OpenCV 라이브러리를 불러오기
from google.colab.patches import cv2_imshow         # from google.colab.patches impo

# YOLOv8 모델 로드
model = YOLO('yolov8l.pt')  # YOLOv8의 경량 모델

# 이미지 로드
img = cv2.imread('coco128/images/train2017/000000000081.jpg')

# 객체 탐지
# 로드된 YOLOv8 모델을 사용하여 이미지에서 객체를 탐지합니다. results 변수에는 탐
results = model(img)

# 결과 시각화
# plot() 함수는 탐지된 객체 주변에 경계 상자를 그려 이미지를 반환
img_with_detections = results[0].plot()  # 첫 번째 결과를 시각화
```

```
# 결과 이미지 표시
cv2_imshow(img_with_detections)
```

```
0: 448x640 1 airplane, 65.1ms
Speed: 1.7ms preprocess, 65.1ms inference, 2.7ms postprocess per image at shape
(1, 3, 448, 640)
```

yaml 파일 생성

- YAML 파일은 YAML(YAML Ain't Markup Language)의 줄임말로, 데이터를 구조화해서 표현할 때 사용하는 사람이 읽기 쉬운 데이터 형식입니다. 주로 설정 파일로 많이 사용됩니다.
- YOLOv8에서 YAML 파일은 주로 모델 설정, 데이터셋 정보, 학습 파라미터 등을 정의하는 데 사용됩니다. 예를 들어, 데이터셋 경로나 클래스 정보, 학습 시 사용될 매개변수 등을 YAML 파일에서 설정합니다.

```
# 데이터셋의 루트 디렉토리 경로
path: /path/to/dataset

# 훈련 이미지의 상대 경로
train: images/train

# 검증 이미지의 상대 경로
val: images/val

# 테스트 이미지의 상대 경로 (선택사항)
test: images/test

# 클래스 이름
names:
  0: class1
  1: class2
```

```
    2: class3
    # ... 추가 클래스들

# 또는 클래스 이름을 리스트로 표현할 수 있습니다
# names: ['class1', 'class2', 'class3', ...]

# 추가 설정 (선택사항)
nc: 3  # 클래스의 수


path: ./coco128  # 데이터셋의 경로
train: images/train  # 학습 이미지 경로
val: images/val  # 검증 이미지 경로
test: images/test  # 테스트 이미지 경로
nc: 80  # 클래스 개수
names: ['person', 'bicycle', 'car', ...]  # 클래스 이름 리스트
```

yaml 파일 작성해 보기

In [ ]:

```
import yaml
from ultralytics import YOLO
import os


# COCO128 데이터셋 경로 확인
dataset_path = 'coco128'
if not os.path.exists(dataset_path):
    raise FileNotFoundError(f"Dataset directory '{dataset_path}' not found.")


# YAML 파일 생성
yaml_content = {
    'path': os.path.abspath(dataset_path),  # 절대 경로 사용
    'train': 'images/train2017',
    'val': 'images/train2017',  # COCO128은 검증 세트가 없으므로 훈련 세트를 사용
    'names': {0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane
              11: 'stop sign', 12: 'parking meter', 13: 'bench', 14: 'bird', 15:
              21: 'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbre
              31: 'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat'
              41: 'cup', 42: 'fork', 43: 'knife', 44: 'spoon', 45: 'bowl', 46: '
              51: 'carrot', 52: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake',
              61: 'toilet', 62: 'tv', 63: 'laptop', 64: 'mouse', 65: 'remote', 6
              71: 'sink', 72: 'refrigerator', 73: 'book', 74: 'clock', 75: 'vase
}


yaml_path = 'coco128_custom.yaml'
with open(yaml_path, 'w') as file:
    yaml.dump(yaml_content, file, default_flow_style=False)

print(f"Created custom YAML file: {yaml_path}")
```

```
Created custom YAML file: coco128_custom.yaml
```

모델 원본 성능

In [ ]:
```python
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# 원본 YOLOv8 모델 로드
model = YOLO('yolov8n.pt')

# 테스트 데이터셋 경로 (COCO 형식의 데이터셋 사용)
# test_data = 'coco128_custom.yaml'

# 모델 평가
results = model.val(data=test_data, conf=0.25, iou=0.5)

# 결과 출력
print("원본 YOLOv8 모델 성능:")
print(f"mAP50: {results.box.map50:.4f}")
print(f"mAP50-95: {results.box.map:.4f}")

# 샘플 이미지에 대한 예측
sample_image = cv2.imread('Dog_rawPixel01.jpg')
sample_results = model(sample_image)

# 결과 시각화
img_with_boxes = sample_results[0].plot()
plt.figure(figsize=(12, 8))
plt.imshow(cv2.cvtColor(img_with_boxes, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.title("원본 YOLOv8 모델 예측 결과")
plt.show()

# 예측 결과 분석
for r in sample_results:
    for box in r.boxes:
        class_id = int(box.cls[0])
        confidence = float(box.conf[0])
        x1, y1, x2, y2 = box.xyxy[0].tolist()
        print(f"클래스: {model.names[class_id]}, 신뢰도: {confidence:.2f}, 바운딩

# 클래스 수 출력
print(f"클래스 수: {len(model.names)}")
print("클래스 목록:")
for i, name in model.names.items():
    print(f"{i}: {name}")
```

```
Ultralytics 8.3.16 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102Mi
B)
YOLOv8n summary (fused): 168 layers, 3,151,904 parameters, 0 gradients, 8.7 GFLOP
s
val: Scanning /content/coco128/labels/train2017.cache... 126 images, 2 background
s, 0 corrupt: 100%|██████████| 128/128 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mA
P50-95): 100%|██████████| 8/8 [00:03<00:00,  2.61it/s]
```

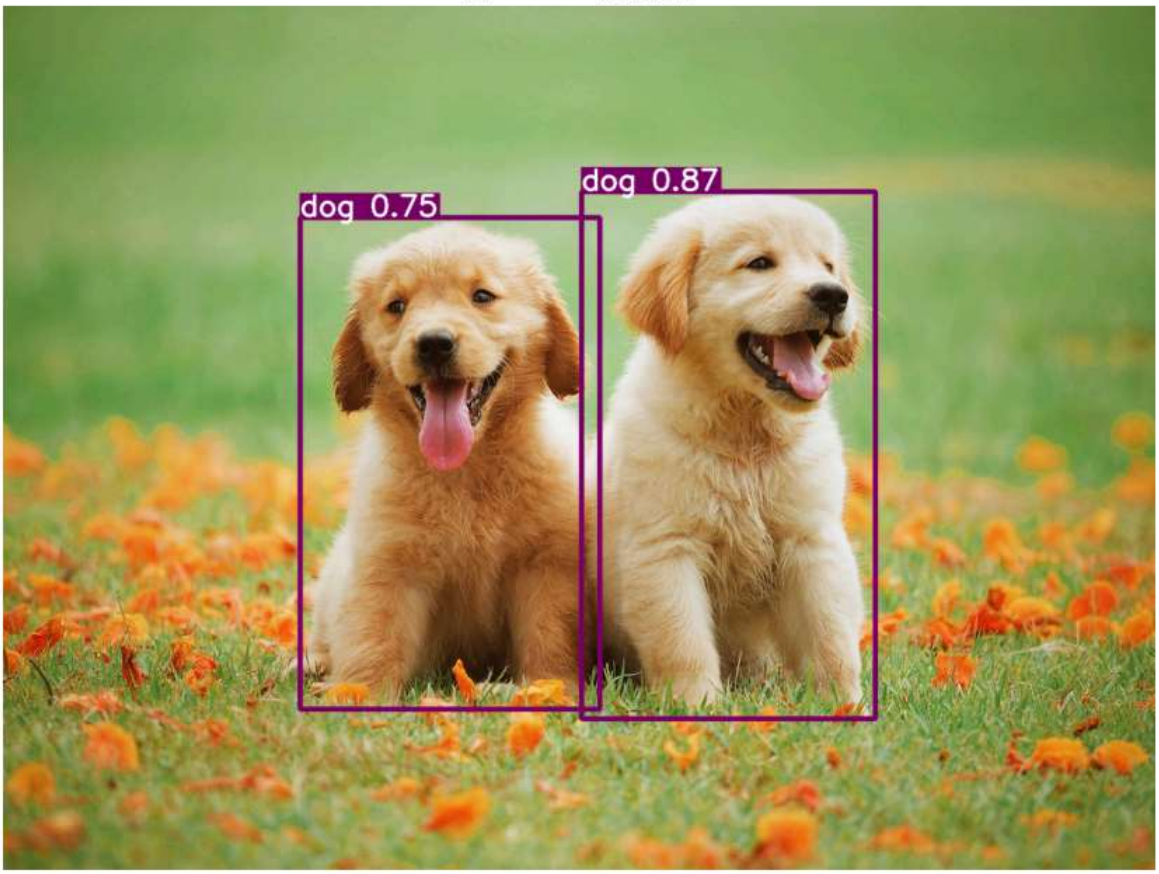
all	128	929	0.699	0.493	0.626
0.494					
person	61	254	0.84	0.661	0.78
0.609					
bicycle	3	6	0.667	0.333	0.499
0.43					
car	12	46	0.909	0.217	0.571
0.406					
motorcycle	4	5	0.667	0.8	0.825
0.671					
airplane	5	6	0.8	0.667	0.8
0.612					
bus	5	7	0.556	0.714	0.794
0.742					
train	3	3	1	0.667	0.833
0.733					
truck	5	12	1	0.25	0.625
0.444					
boat	2	6	0.5	0.167	0.374
0.0748					
traffic light	4	14	0.667	0.143	0.429
0.386					
stop sign	2	2	1	0.5	0.75
0.6					
bench	5	9	1	0.222	0.611
0.412					
bird	2	16	0.917	0.688	0.825
0.493					
cat	4	4	1	1	0.995
0.821					
dog	9	9	0.615	0.889	0.766
0.627					
horse	1	2	0.667	1	0.995
0.514					
elephant	4	17	0.929	0.765	0.874
0.706					
bear	1	1	1	1	0.995
0.995					
zebra	2	4	1	1	0.995
0.973					
giraffe	4	9	0.889	0.889	0.926
0.712					
backpack	4	6	0.667	0.333	0.556
0.364					
umbrella	4	18	1	0.444	0.722
0.527					
handbag	9	19	0	0	0
0					
tie	6	7	0.833	0.714	0.785
0.583					
suitcase	2	4	0.5	0.5	0.539
0.479					
frisbee	5	5	0.8	0.8	0.839
0.775					
skis	1	1	1	1	0.995
0.497					
snowboard	2	7	0.8	0.571	0.744
0.555					
sports ball	6	6	0.667	0.333	0.556
0.369					

kite	2	10	0.8	0.4	0.619	0.271
baseball bat	4	4	1	0.25	0.625	0.375
baseball glove	4	7	0.75	0.429	0.642	0.465
skateboard	3	5	1	0.6	0.8	0.563
tennis racket	5	7	0.667	0.286	0.524	0.424
bottle	6	18	0.545	0.333	0.382	0.27
wine glass	5	16	0.714	0.312	0.559	0.45
cup	10	36	0.667	0.278	0.503	0.419
fork	6	6	1	0.167	0.583	0.525
knife	7	16	0.778	0.438	0.656	0.481
spoon	5	22	0.667	0.182	0.438	0.275
bowl	9	28	0.692	0.643	0.689	0.55
banana	1	1	0	0	0	0
sandwich	2	2	0.333	0.5	0.25	0.25
orange	1	4	1	0.25	0.625	0.312
broccoli	4	11	0.5	0.182	0.388	0.365
carrot	3	24	0.786	0.458	0.63	0.434
hot dog	1	2	0.5	0.5	0.622	0.622
pizza	5	5	0.833	1	0.995	0.865
donut	2	14	0.667	1	0.94	0.873
cake	4	4	0.8	1	0.995	0.904
chair	9	35	0.6	0.514	0.503	0.291
couch	5	6	0.6	0.5	0.648	0.561
potted plant	9	14	0.692	0.643	0.746	0.541
bed	3	3	1	0.667	0.833	0.733
dining table	10	13	0.4	0.462	0.514	0.422
toilet	2	2	1	0.5	0.75	0.75
tv	2	2	0.5	0.5	0.622	0.622
laptop	2	3	0	0	0	0
mouse	2	2	0	0	0	0

```
                remote          5          8          1        0.5       0.75
0.692
            cell phone          5          8          0          0          0
0
             microwave          3          3      0.667      0.667      0.777
0.744
                  oven          5          5        0.5        0.4      0.481
0.389
                  sink          4          6        0.5      0.167      0.292
0.233
          refrigerator          5          5      0.667        0.4      0.598
0.489
                  book          6         29       0.75      0.103      0.423
0.311
                 clock          8          9      0.778      0.778       0.85
0.743
                  vase          2          2        0.4          1      0.828
0.8
              scissors          1          1          0          0          0
0
            teddy bear          6         21          1      0.333      0.667
0.487
            toothbrush          2          5          1        0.4        0.7
0.469
Speed: 0.3ms preprocess, 5.2ms inference, 0.0ms loss, 4.8ms postprocess per image
Results saved to runs/detect/val2
원본 YOLOv8 모델 성능:
mAP50: 0.6260
mAP50-95: 0.4942

0: 480x640 2 dogs, 67.3ms
Speed: 4.3ms preprocess, 67.3ms inference, 2.1ms postprocess per image at shape
(1, 3, 480, 640)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 50896 (\N{HANGUL SYLLABLE WEON}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 48376 (\N{HANGUL SYLLABLE BON}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 47784 (\N{HANGUL SYLLABLE MO}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 45944 (\N{HANGUL SYLLABLE DEL}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 50696 (\N{HANGUL SYLLABLE YE}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 52769 (\N{HANGUL SYLLABLE CEUG}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 44208 (\N{HANGUL SYLLABLE GYEOL}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 44284 (\N{HANGUL SYLLABLE GWA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```

□□ YOLOv8 □□ □□ □□



```
클래스: dog, 신뢰도: 0.87, 바운딩 박스: [602.20, 193.65, 908.10, 742.98]
클래스: dog, 신뢰도: 0.75, 바운딩 박스: [309.01, 220.69, 621.53, 732.43]
클래스 수: 80
클래스 목록:
0: person
1: bicycle
2: car
3: motorcycle
4: airplane
5: bus
6: train
7: truck
8: boat
9: traffic light
10: fire hydrant
11: stop sign
12: parking meter
13: bench
14: bird
15: cat
16: dog
17: horse
18: sheep
19: cow
20: elephant
21: bear
22: zebra
23: giraffe
24: backpack
25: umbrella
26: handbag
27: tie
28: suitcase
29: frisbee
30: skis
31: snowboard
32: sports ball
33: kite
34: baseball bat
35: baseball glove
36: skateboard
37: surfboard
38: tennis racket
39: bottle
40: wine glass
41: cup
42: fork
43: knife
44: spoon
45: bowl
46: banana
47: apple
48: sandwich
49: orange
50: broccoli
51: carrot
52: hot dog
53: pizza
54: donut
55: cake
```

```
56: chair
57: couch
58: potted plant
59: bed
60: dining table
61: toilet
62: tv
63: laptop
64: mouse
65: remote
66: keyboard
67: cell phone
68: microwave
69: oven
70: toaster
71: sink
72: refrigerator
73: book
74: clock
75: vase
76: scissors
77: teddy bear
78: hair drier
79: toothbrush
```

YOLOv8 모델 정보

- 버전: Ultralytics 8.3.16
- Python 버전: Python-3.10.12
- PyTorch 버전: torch-2.4.1+cu121
- CUDA: 사용되는 GPU (Tesla T4, 15102MiB)
- 모델 요약:
  - 총 168 layers
  - 3,151,904 개의 파라미터
  - 0 gradients
  - 8.7 GFLOPs (FLOPs는 부동소수점 연산 횟수를 나타냄)

```
val: Scanning /content/coco128/labels/train2017.cache... 126
images, 2 backgrounds, 0 corrupt: 100%|██████████| 128/128
[00:00<?, ?it/s]
```

검증 단계

- 검증 데이터셋: 총 126장의 이미지(배경 포함)에서 데이터를 스캔했습니다.
- 검증 결과 요약:
- 각 클래스에 대한 이미지 수, 인스턴스 수, 박스 정확도(P), 재현율(R), 평균 정밀도(mAP50), mAP50-95의 값을 요약합니다.

```
Images  Instances      Box(P          R      mAP50  mAP50-95):
100%|██████████| 8/8 [00:03<00:00,  2.61it/s]
```

```
                   all        128        929      0.699
0.493      0.626      0.494
                person         61        254       0.84
0.661       0.78      0.609
               bicycle          3          6      0.667
0.333      0.499       0.43
                   car         12         46      0.909
0.217      0.571      0.406
            motorcycle          4          5      0.667
0.8      0.825      0.671
              airplane          5          6        0.8
0.667        0.8      0.612
                   bus          5          7      0.556
0.714      0.794      0.742
                 train          3          3          1
0.667      0.833      0.733
                 truck          5         12          1
0.25      0.625      0.444
                   ...
```

클래스별 성능

- "all" 클래스(모든 클래스 통합):
  - 이미지 수: 128
  - 인스턴스 수: 929
  - Box P: 0.699 (정확도)
  - R: 0.493 (재현율)
  - mAP50: 0.626 (평균 정밀도에 대한 값)
  - mAP50-95: 0.494
- 특정 클래스 예시:
  - 사람(person): P: 0.84, R: 0.661, mAP50: 0.78
  - 자동차(car): P: 0.909, R: 0.217, mAP50: 0.571
  - 개(dog): P: 0.615, R: 0.889, mAP50: 0.766
  - 과일(banana): P: 0.0, R: 0.0, mAP50: 0.0 (모델이 과일 인식에 실패)

```
Speed: 0.3ms preprocess, 5.2ms inference, 0.0ms loss, 4.8ms
postprocess per image
Results saved to runs/detect/val2
원본 YOLOv8 모델 성능:
mAP50: 0.6260
mAP50-95: 0.4942
```

성능 데이터

- 속도:
  - 이미지 전처리: 0.3ms
  - 추론(inference): 5.2ms
  - 손실(loss): 0.0ms

  - 후처리(postprocess): 4.8ms
- 최종 성능 결과
  - YOLOv8 모델의 성능 지표:
  - mAP50: 0.6260 (0.626)
  - mAP50-95: 0.4942 (0.4942)

주요 지표인 mAP50과 mAP50-95를 분석

mAP50 (Mean Average Precision at IoU=0.50): 0.6260

이 값은 IoU(Intersection over Union) 임계값이 0.5일 때의 평균 정밀도를 나타냅니다.
0.6260은 62.60%의 정확도를 의미합니다.
이는 모델이 객체의 위치를 대략적으로 잘 찾아내고 있음을 나타냅니다.

mAP50-95 (Mean Average Precision at IoU=0.50:0.95): 0.4942
이 값은 IoU 임계값을 0.5에서 0.95까지 변화시키면서 계산한 평균 정밀도의 평균입니다.

0.4942는 49.42%의 정확도를 의미합니다.

이 값이 mAP50보다 낮은 것은 정상적이며, 더 엄격한 기준으로 평가했기 때문입니다.

성능 평가:
일반적인 기준:

mAP50이 0.6 이상이면 괜찮은 성능으로 간주됩니다.
mAP50-95가 0.4 이상이면 준수한 성능으로 볼 수 있습니다.

이 모델의 경우:
mAP50이 0.6260으로, 기본적인 객체 탐지 능력이 양호합니다.
mAP50-95가 0.4942로, 더 엄격한 기준에서도 준수한 성능을 보여줍니다.

해석:
이 모델은 객체의 대략적인 위치를 잘 찾아내며, 정확한 경계 상자 예측에서도 준수한 성능을 보입니다.
일반적인 객체 탐지 작업에 적합한 수준의 성능을 가지고 있습니다.
개선 가능성:

더 높은 성능이 필요하다면, 모델 아키텍처 개선, 데이터 증강, 더 긴 훈련 시간 등을 통해 성능을 향상시킬 수 있습니다.
용도에 따른 평가:

일반적인 객체 탐지 작업에는 충분한 성능입니다.
매우 정밀한 경계 상자가 필요한 특수한 응용 분야에서는 추가적인 개선이 필요할 수 있습니다.
결론적으로, 이 YOLOv8 모델은 전반적으로 양호한 성능을 보여주고 있으며, 대부분의 일반적인 객체 탐지 작업에 적합할 것으로 판단됩니다. 특정 응용 분야나 더 높은 정확도가 요구되는 경우에는 추가적인 최적화나 훈련이 필요할 수 있습니다.

In [ ]:
```python
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# 원본 YOLOv8 모델 로드
model = YOLO('yolov8n.pt')

# 테스트 데이터셋 경로 (COCO 형식의 데이터셋 사용)
test_data = 'coco128_custom.yaml'

# 모델 평가
results = model.val(data=test_data, conf=0.25, iou=0.5)

# 결과 출력
print("원본 YOLOv8 모델 성능:")
print(f"mAP50: {results.box.map50:.4f}")
print(f"mAP50-95: {results.box.map:.4f}")

# 샘플 이미지에 대한 예측
sample_image = cv2.imread('life_unsplash.jpeg')
sample_results = model(sample_image)

# 결과 시각화
img_with_boxes = sample_results[0].plot()
plt.figure(figsize=(12, 8))
plt.imshow(cv2.cvtColor(img_with_boxes, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.title("원본 YOLOv8 모델 예측 결과")
plt.show()

# 예측 결과 분석
for r in sample_results:
    for box in r.boxes:
        class_id = int(box.cls[0])
        confidence = float(box.conf[0])
        x1, y1, x2, y2 = box.xyxy[0].tolist()
        print(f"클래스: {model.names[class_id]}, 신뢰도: {confidence:.2f}, 바운딩

# 클래스 수 출력
print(f"클래스 수: {len(model.names)}")
print("클래스 목록:")
for i, name in model.names.items():
    print(f"{i}: {name}")
```

```
Ultralytics 8.3.16 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102Mi
B)
YOLOv8n summary (fused): 168 layers, 3,151,904 parameters, 0 gradients, 8.7 GFLOP
s
val: Scanning /content/coco128/labels/train2017.cache... 126 images, 2 background
s, 0 corrupt: 100%|██████████| 128/128 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mA
P50-95): 100%|██████████| 8/8 [00:04<00:00,  1.90it/s]
```

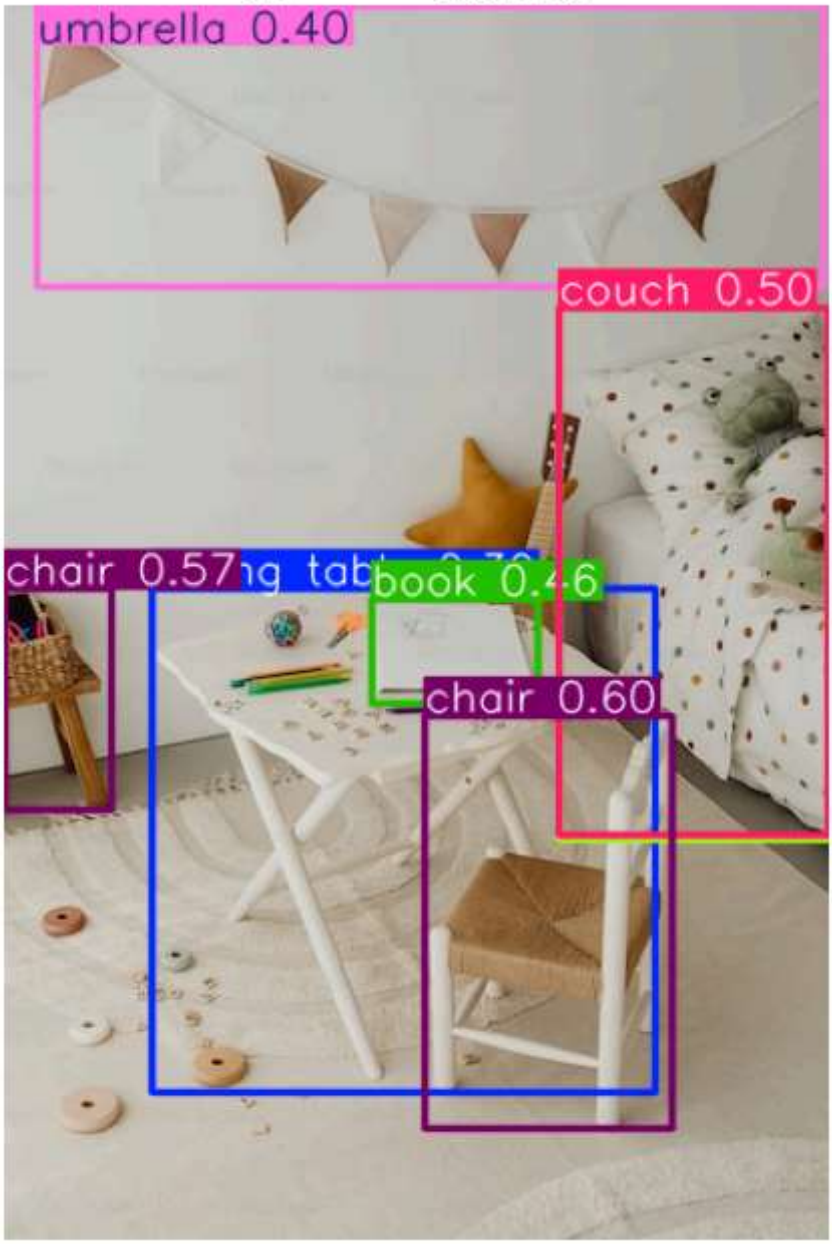
```
                   all        128        929      0.699      0.493      0.626
0.494
                person         61        254       0.84      0.661       0.78
0.609
               bicycle          3          6      0.667      0.333      0.499
0.43
                   car         12         46      0.909      0.217      0.571
0.406
            motorcycle          4          5      0.667        0.8      0.825
0.671
              airplane          5          6        0.8      0.667        0.8
0.612
                   bus          5          7      0.556      0.714      0.794
0.742
                 train          3          3          1      0.667      0.833
0.733
                 truck          5         12          1       0.25      0.625
0.444
                  boat          2          6        0.5      0.167      0.374
0.0748
         traffic light          4         14      0.667      0.143      0.429
0.386
             stop sign          2          2          1        0.5       0.75
0.6
                 bench          5          9          1      0.222      0.611
0.412
                  bird          2         16      0.917      0.688      0.825
0.493
                   cat          4          4          1          1      0.995
0.821
                   dog          9          9      0.615      0.889      0.766
0.627
                 horse          1          2      0.667          1      0.995
0.514
              elephant          4         17      0.929      0.765      0.874
0.706
                  bear          1          1          1          1      0.995
0.995
                 zebra          2          4          1          1      0.995
0.973
               giraffe          4          9      0.889      0.889      0.926
0.712
              backpack          4          6      0.667      0.333      0.556
0.364
              umbrella          4         18          1      0.444      0.722
0.527
               handbag          9         19          0          0          0
0
                   tie          6          7      0.833      0.714      0.785
0.583
              suitcase          2          4        0.5        0.5      0.539
0.479
               frisbee          5          5        0.8        0.8      0.839
0.775
                  skis          1          1          1          1      0.995
0.497
             snowboard          2          7        0.8      0.571      0.744
0.555
           sports ball          6          6      0.667      0.333      0.556
0.369
```

kite	2	10	0.8	0.4	0.619	0.271
baseball bat	4	4	1	0.25	0.625	0.375
baseball glove	4	7	0.75	0.429	0.642	0.465
skateboard	3	5	1	0.6	0.8	0.563
tennis racket	5	7	0.667	0.286	0.524	0.424
bottle	6	18	0.545	0.333	0.382	0.27
wine glass	5	16	0.714	0.312	0.559	0.45
cup	10	36	0.667	0.278	0.503	0.419
fork	6	6	1	0.167	0.583	0.525
knife	7	16	0.778	0.438	0.656	0.481
spoon	5	22	0.667	0.182	0.438	0.275
bowl	9	28	0.692	0.643	0.689	0.55
banana	1	1	0	0	0	0
sandwich	2	2	0.333	0.5	0.25	0.25
orange	1	4	1	0.25	0.625	0.312
broccoli	4	11	0.5	0.182	0.388	0.365
carrot	3	24	0.786	0.458	0.63	0.434
hot dog	1	2	0.5	0.5	0.622	0.622
pizza	5	5	0.833	1	0.995	0.865
donut	2	14	0.667	1	0.94	0.873
cake	4	4	0.8	1	0.995	0.904
chair	9	35	0.6	0.514	0.503	0.291
couch	5	6	0.6	0.5	0.648	0.561
potted plant	9	14	0.692	0.643	0.746	0.541
bed	3	3	1	0.667	0.833	0.733
dining table	10	13	0.4	0.462	0.514	0.422
toilet	2	2	1	0.5	0.75	0.75
tv	2	2	0.5	0.5	0.622	0.622
laptop	2	3	0	0	0	0
mouse	2	2	0	0	0	0

```
                remote          5          8          1        0.5       0.75
0.692
            cell phone          5          8          0          0          0
0
             microwave          3          3      0.667      0.667      0.777
0.744
                  oven          5          5        0.5        0.4      0.481
0.389
                  sink          4          6        0.5      0.167      0.292
0.233
          refrigerator          5          5      0.667        0.4      0.598
0.489
                  book          6         29       0.75      0.103      0.423
0.311
                 clock          8          9      0.778      0.778       0.85
0.743
                  vase          2          2        0.4          1      0.828
0.8
              scissors          1          1          0          0          0
0
            teddy bear          6         21          1      0.333      0.667
0.487
            toothbrush          2          5          1        0.4        0.7
0.469
Speed: 0.5ms preprocess, 5.1ms inference, 0.0ms loss, 9.1ms postprocess per image
Results saved to runs/detect/val3
원본 YOLOv8 모델 성능:
mAP50: 0.6260
mAP50-95: 0.4942

0: 640x448 1 umbrella, 2 chairs, 1 couch, 1 bed, 1 dining table, 1 book, 48.3ms
Speed: 2.3ms preprocess, 48.3ms inference, 1.7ms postprocess per image at shape
(1, 3, 640, 448)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 50896 (\N{HANGUL SYLLABLE WEON}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 48376 (\N{HANGUL SYLLABLE BON}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 47784 (\N{HANGUL SYLLABLE MO}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 45944 (\N{HANGUL SYLLABLE DEL}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 50696 (\N{HANGUL SYLLABLE YE}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 52769 (\N{HANGUL SYLLABLE CEUG}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 44208 (\N{HANGUL SYLLABLE GYEOL}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 44284 (\N{HANGUL SYLLABLE GWA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```

□□ YOLOv8 □□ □□ □□



```
클래스: chair, 신뢰도: 0.60, 바운딩 박스: [197.60, 334.23, 313.21, 528.42]
클래스: chair, 신뢰도: 0.57, 바운딩 박스: [0.00, 274.12, 50.74, 378.62]
클래스: couch, 신뢰도: 0.50, 바운딩 박스: [260.07, 142.78, 386.80, 390.63]
클래스: book, 신뢰도: 0.46, 바운딩 박스: [172.28, 279.28, 250.58, 328.04]
클래스: bed, 신뢰도: 0.44, 바운딩 박스: [260.65, 142.66, 386.50, 392.55]
클래스: umbrella, 신뢰도: 0.40, 바운딩 박스: [15.75, 0.23, 384.39, 132.08]
클래스: dining table, 신뢰도: 0.30, 바운딩 박스: [69.72, 274.91, 305.50, 511.90]
클래스 수: 80
클래스 목록:
0: person
1: bicycle
2: car
3: motorcycle
4: airplane
5: bus
6: train
7: truck
8: boat
9: traffic light
10: fire hydrant
11: stop sign
12: parking meter
13: bench
14: bird
15: cat
16: dog
17: horse
18: sheep
19: cow
20: elephant
21: bear
22: zebra
23: giraffe
24: backpack
25: umbrella
26: handbag
27: tie
28: suitcase
29: frisbee
30: skis
31: snowboard
32: sports ball
33: kite
34: baseball bat
35: baseball glove
36: skateboard
37: surfboard
38: tennis racket
39: bottle
40: wine glass
41: cup
42: fork
43: knife
44: spoon
45: bowl
46: banana
47: apple
48: sandwich
49: orange
50: broccoli
```

```
51: carrot
52: hot dog
53: pizza
54: donut
55: cake
56: chair
57: couch
58: potted plant
59: bed
60: dining table
61: toilet
62: tv
63: laptop
64: mouse
65: remote
66: keyboard
67: cell phone
68: microwave
69: oven
70: toaster
71: sink
72: refrigerator
73: book
74: clock
75: vase
76: scissors
77: teddy bear
78: hair drier
79: toothbrush
```

```
클래스: chair, 신뢰도: 0.60, 바운딩 박스: [197.60, 334.23, 313.21, 528.42]
클래스: chair, 신뢰도: 0.57, 바운딩 박스: [0.00, 274.12, 50.74, 378.62]
클래스: couch, 신뢰도: 0.50, 바운딩 박스: [260.07, 142.78, 386.80, 390.63]
클래스: book, 신뢰도: 0.46, 바운딩 박스: [172.28, 279.28, 250.58, 328.04]
클래스: bed, 신뢰도: 0.44, 바운딩 박스: [260.65, 142.66, 386.50, 392.55]
클래스: umbrella, 신뢰도: 0.40, 바운딩 박스: [15.75, 0.23, 384.39, 132.08]
클래스: dining table, 신뢰도: 0.30, 바운딩 박스: [69.72, 274.91, 305.50, 511.90]
```

이 결과에서 주목할 점:

신뢰도: 대부분의 탐지가 중간 정도의 신뢰도(0.4~0.6)를 가지고 있어, 모델이 완전히 확신하지는 않지만 어느 정도 신뢰할 만한 탐지를 했음을 나타냅니다.
중복 탐지: 의자가 두 번 탐지되었는데, 이는 실제로 두 개의 의자가 있거나 같은 의자를 다르게 인식했을 가능성이 있습니다.
혼동: 소파와 침대가 거의 같은 위치에 탐지되었는데, 이는 모델이 이 두 객체를 구분하는 데 어려움을 겪고 있음을 나타냅니다.
낮은 신뢰도 탐지: 식탁의 신뢰도가 30%로 가장 낮아, 이 탐지는 다소 불확실할 수 있습니다.

다양한 객체: 모델이 가구(의자, 소파, 침대, 식탁)부터 작은 물건(책)과 우산까지 다양한 종류의 객체를 탐지할 수 있음을 보여줍니다.

In [ ]:
```python
from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

# 원본 YOLOv8 모델 로드
model = YOLO('yolov8l.pt')

# 테스트 데이터셋 경로 (COCO 형식의 데이터셋 사용)
test_data = 'coco128_custom.yaml'

# 모델 평가
results = model.val(data=test_data, conf=0.25, iou=0.5)

# 결과 출력
print("원본 YOLOv8 모델 성능:")
print(f"mAP50: {results.box.map50:.4f}")
print(f"mAP50-95: {results.box.map:.4f}")

# 샘플 이미지에 대한 예측
sample_image = cv2.imread('life_unsplash.jpeg')
sample_results = model(sample_image)

# 결과 시각화
img_with_boxes = sample_results[0].plot()
plt.figure(figsize=(12, 8))
plt.imshow(cv2.cvtColor(img_with_boxes, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.title("원본 YOLOv8 모델 예측 결과")
plt.show()

# 예측 결과 분석
for r in sample_results:
    for box in r.boxes:
        class_id = int(box.cls[0])
        confidence = float(box.conf[0])
        x1, y1, x2, y2 = box.xyxy[0].tolist()
        print(f"클래스: {model.names[class_id]}, 신뢰도: {confidence:.2f}, 바운딩

# 클래스 수 출력
print(f"클래스 수: {len(model.names)}")
print("클래스 목록:")
for i, name in model.names.items():
    print(f"{i}: {name}")
```

```
Ultralytics 8.3.16 🚀 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102Mi
B)
YOLOv8l summary (fused): 268 layers, 43,668,288 parameters, 0 gradients, 165.2 GF
LOPs
val: Scanning /content/coco128/labels/train2017.cache... 126 images, 2 background
s, 0 corrupt: 100%|██████████| 128/128 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mA
P50-95): 100%|██████████| 8/8 [00:05<00:00,  1.34it/s]
```

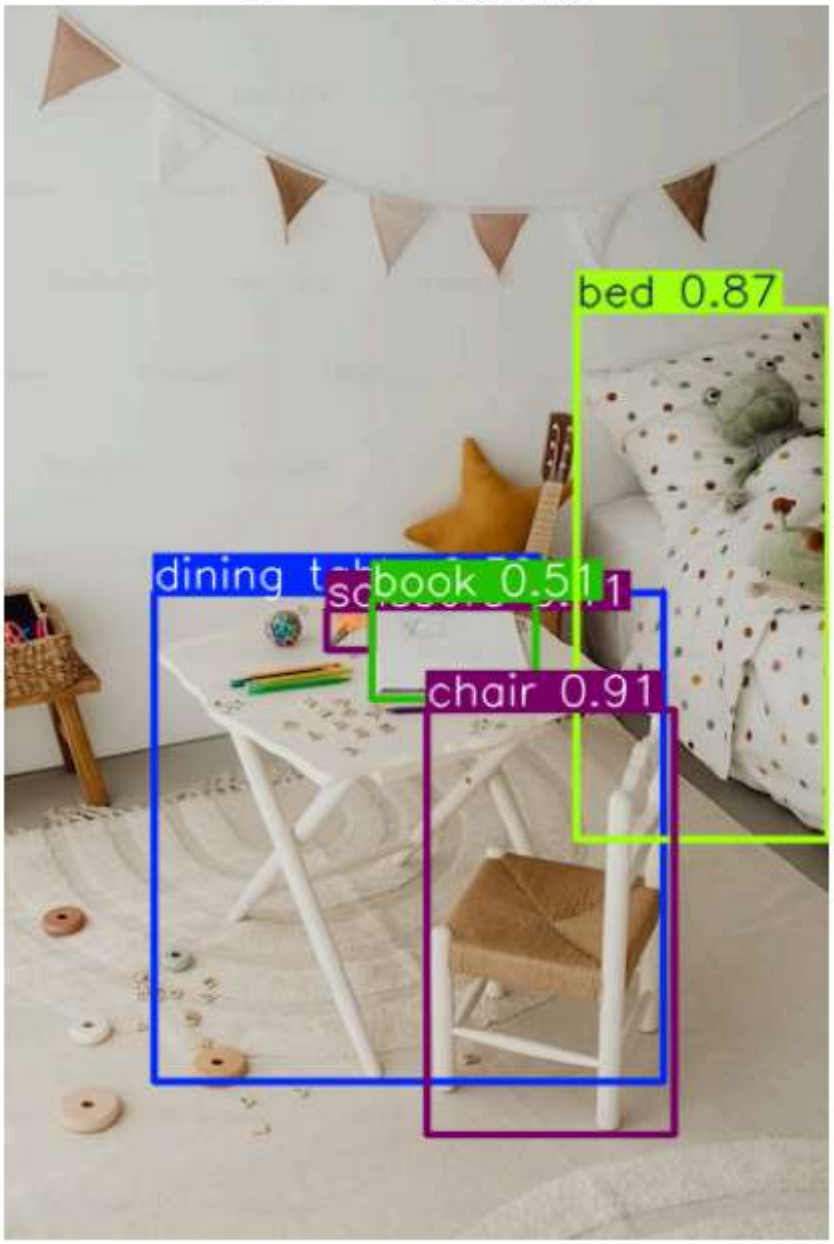
all	128	929	0.788	0.723	0.793
0.667					
person	61	254	0.88	0.748	0.846
0.726					
bicycle	3	6	1	0.333	0.667
0.55					
car	12	46	0.955	0.457	0.711
0.473					
motorcycle	4	5	0.833	1	0.962
0.853					
airplane	5	6	1	1	0.995
0.986					
bus	5	7	0.833	0.714	0.833
0.762					
train	3	3	1	1	0.995
0.973					
truck	5	12	0.6	0.5	0.566
0.393					
boat	2	6	1	0.667	0.833
0.595					
traffic light	4	14	1	0.214	0.607
0.423					
stop sign	2	2	1	1	0.995
0.895					
bench	5	9	0.857	0.667	0.809
0.715					
bird	2	16	1	1	0.995
0.734					
cat	4	4	1	1	0.995
0.926					
dog	9	9	0.818	1	0.995
0.901					
horse	1	2	1	1	0.995
0.846					
elephant	4	17	1	0.941	0.971
0.887					
bear	1	1	1	1	0.995
0.895					
zebra	2	4	1	1	0.995
0.98					
giraffe	4	9	0.889	0.889	0.938
0.793					
backpack	4	6	1	0.667	0.833
0.627					
umbrella	4	18	0.889	0.889	0.928
0.7					
handbag	9	19	0.667	0.316	0.514
0.406					
tie	6	7	0.833	0.714	0.833
0.705					
suitcase	2	4	1	0.75	0.875
0.732					
frisbee	5	5	0.8	0.8	0.879
0.793					
skis	1	1	1	1	0.995
0.895					
snowboard	2	7	1	0.857	0.928
0.834					
sports ball	6	6	0.6	0.5	0.648
0.423					

kite	2	10	0.6	0.3	0.508	0.146
baseball bat	4	4	0.8	1	0.995	0.513
baseball glove	4	7	0.667	0.571	0.685	0.465
skateboard	3	5	0.6	0.6	0.668	0.513
tennis racket	5	7	0.667	0.571	0.715	0.272
bottle	6	18	0.733	0.611	0.643	0.521
wine glass	5	16	0.917	0.688	0.815	0.602
cup	10	36	0.903	0.778	0.851	0.679
fork	6	6	0.75	0.5	0.604	0.457
knife	7	16	0.8	0.75	0.816	0.643
spoon	5	22	0.75	0.545	0.671	0.567
bowl	9	28	0.846	0.786	0.852	0.771
banana	1	1	1	1	0.995	0.995
sandwich	2	2	0.667	1	0.995	0.995
orange	1	4	0	0	0	0
broccoli	4	11	0.5	0.182	0.388	0.365
carrot	3	24	0.867	0.542	0.708	0.564
hot dog	1	2	0.667	1	0.995	0.995
pizza	5	5	0.833	1	0.995	0.861
donut	2	14	0.737	1	0.967	0.904
cake	4	4	0.8	1	0.995	0.911
chair	9	35	0.7	0.8	0.814	0.616
couch	5	6	0.714	0.833	0.811	0.692
potted plant	9	14	0.769	0.714	0.801	0.68
bed	3	3	0.75	1	0.995	0.723
dining table	10	13	0.588	0.769	0.786	0.652
toilet	2	2	1	1	0.995	0.958
tv	2	2	0.667	1	0.995	0.954
laptop	2	3	1	0.333	0.667	0.667
mouse	2	2	0	0	0	0

```
                remote          5          8          1      0.625      0.812
0.757
            cell phone          5          8      0.833      0.625       0.74
0.563
             microwave          3          3          1          1      0.995
0.897
                  oven          5          5      0.333        0.4      0.365
0.285
                  sink          4          6      0.429        0.5      0.406
0.293
          refrigerator          5          5        0.8        0.8      0.879
0.761
                  book          6         29      0.727      0.276      0.523
0.396
                 clock          8          9        0.8      0.889      0.911
0.772
                  vase          2          2      0.333          1      0.995
0.995
              scissors          1          1          0          0          0
0
            teddy bear          6         21      0.938      0.714      0.845
0.681
            toothbrush          2          5          1          1      0.995
0.844
Speed: 0.2ms preprocess, 30.2ms inference, 0.0ms loss, 1.5ms postprocess per imag
e
Results saved to runs/detect/val4
원본 YOLOv8 모델 성능:
mAP50: 0.7933
mAP50-95: 0.6669

0: 640x448 1 chair, 1 bed, 1 dining table, 1 book, 1 scissors, 59.4ms
Speed: 3.0ms preprocess, 59.4ms inference, 2.1ms postprocess per image at shape
(1, 3, 640, 448)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 50896 (\N{HANGUL SYLLABLE WEON}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 48376 (\N{HANGUL SYLLABLE BON}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 47784 (\N{HANGUL SYLLABLE MO}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 45944 (\N{HANGUL SYLLABLE DEL}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 50696 (\N{HANGUL SYLLABLE YE}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 52769 (\N{HANGUL SYLLABLE CEUG}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 44208 (\N{HANGUL SYLLABLE GYEOL}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.10/dist-packages/IPython/core/pylabtools.py:151: UserWarni
ng: Glyph 44284 (\N{HANGUL SYLLABLE GWA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```

□□ YOLOv8 □□ □□ □□



```
클래스: chair, 신뢰도: 0.91, 바운딩 박스: [198.58, 331.67, 314.12, 531.07]
클래스: bed, 신뢰도: 0.87, 바운딩 박스: [268.38, 143.27, 386.54, 392.72]
클래스: book, 신뢰도: 0.51, 바운딩 박스: [172.49, 279.84, 249.48, 326.53]
클래스: scissors, 신뢰도: 0.41, 바운딩 박스: [151.41, 284.57, 172.81, 303.06]
클래스: dining table, 신뢰도: 0.39, 바운딩 박스: [70.52, 276.22, 309.80, 506.33]
클래스 수: 80
클래스 목록:
0: person
1: bicycle
2: car
3: motorcycle
4: airplane
5: bus
6: train
7: truck
8: boat
9: traffic light
10: fire hydrant
11: stop sign
12: parking meter
13: bench
14: bird
15: cat
16: dog
17: horse
18: sheep
19: cow
20: elephant
21: bear
22: zebra
23: giraffe
24: backpack
25: umbrella
26: handbag
27: tie
28: suitcase
29: frisbee
30: skis
31: snowboard
32: sports ball
33: kite
34: baseball bat
35: baseball glove
36: skateboard
37: surfboard
38: tennis racket
39: bottle
40: wine glass
41: cup
42: fork
43: knife
44: spoon
45: bowl
46: banana
47: apple
48: sandwich
49: orange
50: broccoli
51: carrot
52: hot dog
```

```
53: pizza
54: donut
55: cake
56: chair
57: couch
58: potted plant
59: bed
60: dining table
61: toilet
62: tv
63: laptop
64: mouse
65: remote
66: keyboard
67: cell phone
68: microwave
69: oven
70: toaster
71: sink
72: refrigerator
73: book
74: clock
75: vase
76: scissors
77: teddy bear
78: hair drier
79: toothbrush
```

전체적인 해석:

신뢰도 분포:
의자와 침대는 매우 높은 신뢰도(90% 이상)로 탐지되었습니다.
책은 중간 정도의 신뢰도로 탐지되었습니다.
가위와 식탁은 비교적 낮은 신뢰도로 탐지되었습니다.

객체 위치:
탐지된 객체들은 이미지의 다양한 부분에 분포되어 있습니다.
의자와 식탁이 이미지의 하단 부분에 큰 영역을 차지하고 있습니다.

가능한 시나리오:
이 이미지는 아마도 침실이나 거실의 일부를 보여주고 있을 것 같습니다.
큰 의자(또는 소파)와 침대가 주요 가구로 보입니다.
책과 가위 같은 작은 물건들이 테이블이나 다른 표면 위에 있을 가능성이 있습니다.

주의점:
가위와 식탁의 탐지는 신뢰도가 낮아 실제로 존재하지 않을 수 있습니다.
식탁으로 탐지된 것은 실제로는 다른 큰 가구(예: 커피 테이블)일 수 있습니다.

모델 성능:
모델은 큰 가구들(의자, 침대)을 매우 확실하게 인식하고 있습니다.
작은 물체(책, 가위)에 대해서는 덜 확신하고 있습니다.
일부 객체(식탁)에 대해서는 오탐지의 가능성이 있습니다.

모델 요약

In [ ]:
```python
from ultralytics import YOLO
import os

# 모델 로드
all_model_list = ['yolov8n', 'yolov8s', 'yolov8m', 'yolov8l', 'yolov8x']

for one_model in all_model_list:
    model = YOLO(f"{one_model}.pt")

    # 모델 요약 정보 출력
    print(f"\n{one_model} 모델 정보:")
    # print(model)

    # 모델 레이어 수 확인
    num_layers = len(list(model.model.parameters()))
    print(f"총 레이어 수: {num_layers}")

    # 모델 파라미터 수 확인
    num_params = sum(p.numel() for p in model.model.parameters())
    print(f"총 파라미터 수: {num_params:,}")

    # 모델 파일 용량 확인
    model_path = f"{one_model}.pt"
    file_size = os.path.getsize(model_path) / (1024 ** 2)  # 파일 크기를 MB 단위
    print(f"모델 파일 크기: {file_size:.2f} MB")
```

```
yolov8n 모델 정보:
총 레이어 수: 184
총 파라미터 수: 3,157,200
모델 파일 크기: 6.25 MB

yolov8s 모델 정보:
총 레이어 수: 184
총 파라미터 수: 11,166,560
모델 파일 크기: 21.54 MB

yolov8m 모델 정보:
총 레이어 수: 244
총 파라미터 수: 25,902,640
모델 파일 크기: 49.72 MB

yolov8l 모델 정보:
총 레이어 수: 304
총 파라미터 수: 43,691,520
모델 파일 크기: 83.73 MB

yolov8x 모델 정보:
총 레이어 수: 304
총 파라미터 수: 68,229,648
모델 파일 크기: 130.55 MB
```

In [ ]:
```python
from ultralytics import YOLO

# 모델 로드
model = YOLO("yolov8n.pt")  # 'yolov8n', 'yolov8s', 'yolov8m', 'yolov8l', 'yolov

# 모델 레이어 수 확인
```

```
num_layers = len(list(model.model.parameters()))
print(f"YOLOv8n 모델의 총 레이어 수: {num_layers}")

# 모델 요약
print(model.model)
```

```
YOLOv8n 모델의 총 레이어 수: 184
DetectionModel(
  (model): Sequential(
    (0): Conv(
      (conv): Conv2d(3, 16, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bi
as=False)
      (bn): BatchNorm2d(16, eps=0.001, momentum=0.03, affine=True, track_running_
stats=True)
      (act): SiLU(inplace=True)
    )
    (1): Conv(
      (conv): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), b
ias=False)
      (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_running_
stats=True)
      (act): SiLU(inplace=True)
    )
    (2): C2f(
      (cv1): Conv(
        (conv): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_runnin
g_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(48, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_runnin
g_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0): Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(16, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(16, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (3): Conv(
      (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), b
ias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_
stats=True)
      (act): SiLU(inplace=True)
    )
    (4): C2f(
      (cv1): Conv(
        (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_runnin
```

```
g_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_runnin
g_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0-1): 2 x Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (5): Conv(
      (conv): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_running
_stats=True)
      (act): SiLU(inplace=True)
    )
    (6): C2f(
      (cv1): Conv(
        (conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0-1): 2 x Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (7): Conv(
      (conv): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track_running
_stats=True)
      (act): SiLU(inplace=True)
    )
    (8): C2f(
      (cv1): Conv(
        (conv): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(384, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0): Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_r
unning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_r
unning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (9): SPPF(
      (cv1): Conv(
        (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): MaxPool2d(kernel_size=5, stride=1, padding=2, dilation=1, ceil_mode=Fa
```

```
lse)
    )
    (10): Upsample(scale_factor=2.0, mode='nearest')
    (11): Concat()
    (12): C2f(
      (cv1): Conv(
        (conv): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0): Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (13): Upsample(scale_factor=2.0, mode='nearest')
    (14): Concat()
    (15): C2f(
      (cv1): Conv(
        (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_runnin
g_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(96, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_runnin
g_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0): Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
```

```
          (cv2): Conv(
            (conv): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(32, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (16): Conv(
      (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), b
ias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_running_
stats=True)
      (act): SiLU(inplace=True)
    )
    (17): Concat()
    (18): C2f(
      (cv1): Conv(
        (conv): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(192, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0): Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (19): Conv(
      (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1),
bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_running
_stats=True)
      (act): SiLU(inplace=True)
    )
    (20): Concat()
    (21): C2f(
      (cv1): Conv(
```

```
        (conv): Conv2d(384, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (cv2): Conv(
        (conv): Conv2d(384, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(256, eps=0.001, momentum=0.03, affine=True, track_runni
ng_stats=True)
        (act): SiLU(inplace=True)
      )
      (m): ModuleList(
        (0): Bottleneck(
          (cv1): Conv(
            (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_r
unning_stats=True)
            (act): SiLU(inplace=True)
          )
          (cv2): Conv(
            (conv): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(128, eps=0.001, momentum=0.03, affine=True, track_r
unning_stats=True)
            (act): SiLU(inplace=True)
          )
        )
      )
    )
    (22): Detect(
      (cv2): ModuleList(
        (0): Sequential(
          (0): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (1): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (2): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
        )
        (1): Sequential(
          (0): Conv(
            (conv): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (1): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (2): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
        )
        (2): Sequential(
          (0): Conv(
            (conv): Conv2d(256, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (1): Conv(
            (conv): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(64, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (2): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
        )
      )
      (cv3): ModuleList(
        (0): Sequential(
          (0): Conv(
            (conv): Conv2d(64, 80, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(80, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (1): Conv(
            (conv): Conv2d(80, 80, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(80, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (2): Conv2d(80, 80, kernel_size=(1, 1), stride=(1, 1))
        )
        (1): Sequential(
          (0): Conv(
            (conv): Conv2d(128, 80, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(80, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (1): Conv(
            (conv): Conv2d(80, 80, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(80, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (2): Conv2d(80, 80, kernel_size=(1, 1), stride=(1, 1))
        )
        (2): Sequential(
```

```
          (0): Conv(
            (conv): Conv2d(256, 80, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1), bias=False)
            (bn): BatchNorm2d(80, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (1): Conv(
            (conv): Conv2d(80, 80, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
            (bn): BatchNorm2d(80, eps=0.001, momentum=0.03, affine=True, track_ru
nning_stats=True)
            (act): SiLU(inplace=True)
          )
          (2): Conv2d(80, 80, kernel_size=(1, 1), stride=(1, 1))
        )
      )
      (dfl): DFL(
        (conv): Conv2d(16, 1, kernel_size=(1, 1), stride=(1, 1), bias=False)
      )
    )
  )
)
```

In [ ]: