Computer Vision

Lecture 03: 객체 감지 모델 Tutorial

Jaesung Lee

curseor@cau.ac.kr

2025. 03. 19.

Contents

1. 환경설정 및 Pascal VOC 데이터셋 다운로드

- 주피터 노트북(Jupyter Notebook)에서 객체 감지 모델을 학습하기 위한 라이브러리 설치 및 임포트
- YOLOv5 학습을 위한 PyTorch 설치 및 Pascal VOC 데이터셋 다운로드를 위한 Torchvision 임포트

1.라이브러리 설치 & 임포트

```
[1]: import torch
     import torchvision
     from torchvision import datasets
     import os

     print("PyTorch 버전 :", torch.__version__)
     print("Torchvision 버전 :", torchvision.__version__)
     device = "cuda" if torch.cuda.is_available() else "cpu"
     print("사용 중인 디바이스 :", device)
```

```
PyTorch 버전 : 2.6.0+cu118
Torchvision 버전 : 0.21.0+cu118
사용 중인 디바이스 : cuda
```

1. 환경설정 및 Pascal VOC 데이터셋 다운로드

- torchvision.datasets.VOCDetection 또는 VOCSegmentation을 사용하여 Pascal VOC 데이터셋을 자동으로 다운로드하여 로컬에서 사용 가능함
- download=True로 설정 시, 'root' 경로에 VOCdevkit 폴더가 생성되어 그 안에 VOC2007 (또는 VOC2012)가 풀리게 됨

2.Pascal VOC 데이터셋 다운로드 (torchvision)

```
•[2]: from torchvision.datasets import VOCDetection

# 로컬(혹은 Colab)에서 사용할 루트 경로 지정
# 예: 현재 디렉토리('.') 아래에 VOCdevkit 폴더가 생성됨
DATA_ROOT = "./"

# VOCDetection으로 train 데이터셋을 다운로드
voc_train = VOCDetection(
    root=DATA_ROOT,
    year="2007",
    image_set="train",  # trainval, test 등 가능
    download=True,      # 자동 다운로드
    transform=None,
    target_transform=None
)

print("VOC 2007 trainset 다운로드 완료!")
print("데이터 수:", len(voc_train))
print("저장 경로 구조 확인:", os.listdir(os.path.join(DATA_ROOT, "VOCdevkit")))
```

```
Using downloaded and verified file: ./VOCtrainval_06-Nov-2007.tar
Extracting ./VOCtrainval_06-Nov-2007.tar to ./
VOC 2007 trainset 다운로드 완료!
데이터 수: 2501
저장 경로 구조 확인: ['VOC2007']
```

2. 데이터셋 구조 탐색 및 포맷 변환

- YOLOv5를 학습하려면 이미지와 라벨(.txt)이 필요한데, 각 이미지마다 같은 이름의 .txt 파일이 필요함 (Ex: 000001.jpg → 000001.txt)
- 이를 위해 다운로드한 데이터셋에서 XML 어노테이션을 파싱하여 YOLO txt로 변환

3. VOC XML을 YOLO 텍스트로 변환하는 함수

```python
import os
import xml.etree.ElementTree as ET
import glob
import random
import shutil
import numpy as np

# 경로 설정
voc_root = "/jupyter_project/Spring_cv/VOCdevkit/VOC2007"
voc_annotations = os.path.join(voc_root, "Annotations")
voc_images = os.path.join(voc_root, "JPEGImages")

# YOLO 데이터셋 저장 위치
yolo_dataset = "/jupyter_project/Spring_cv/yolo_voc_dataset"
os.makedirs(yolo_dataset, exist_ok=True)

# 클래스 목록
voc_classes = [
    "aeroplane", "bicycle", "bird", "boat", "bottle",
    "bus", "car", "cat", "chair", "cow",
    "diningtable", "dog", "horse", "motorbike", "person",
    "pottedplant", "sheep", "sofa", "train", "tvmonitor"
]

# VOC XML을 YOLO 텍스트로 변환하는 함수
def convert_voc_to_yolo(xml_file, output_dir):
    tree = ET.parse(xml_file)
    root = tree.getroot()

    size = root.find("size")
    width = int(size.find("width").text)
    height = int(size.find("height").text)

    filename = root.find("filename").text
    base_name = os.path.splitext(filename)[0]

    yolo_lines = []

    for obj in root.findall("object"):
        # 클래스 이름 및 ID 가져오기
        class_name = obj.find("name").text
        if class_name not in voc_classes:
            continue

        class_id = voc_classes.index(class_name)

        # difficult 플래그 확인 (옵션)
        difficult = obj.find("difficult")
        if difficult is not None and int(difficult.text) == 1:
            continue

        # 바운딩 박스 좌표 가져오기
        bbox = obj.find("bndbox")
        xmin = float(bbox.find("xmin").text)
        ymin = float(bbox.find("ymin").text)
        xmax = float(bbox.find("xmax").text)
        ymax = float(bbox.find("ymax").text)

        # YOLO 형식으로 변환 (중심점 x, y, 너비, 높이) - 모두 0~1 사이 값
        x_center = ((xmin + xmax) / 2) / width
        y_center = ((ymin + ymax) / 2) / height
        box_width = (xmax - xmin) / width
        box_height = (ymax - ymin) / height

        # YOLO 형식으로 저장
        yolo_lines.append(f"{class_id} {x_center:.6f} {y_center:.6f} {box_width:.6f} {box_height:.6f}")

    # 변환된 라벨 저장
    if yolo_lines:  # 유효한 객체가 있는 경우에만 저장
        with open(os.path.join(output_dir, f"{base_name}.txt"), "w") as f:
            f.write("\n".join(yolo_lines))
        return True
    return False
```

3. Train/Test 데이터셋 분할 및 data.yaml 구성

- Pascal VOC는 ImageSets/Main 폴더 내 txt 파일로 이미지 분할(train, val, test)을 관리
- 해당 실습에서는 VOC2007 trainval.txt, test.txt를 이용해서 Train Set : Validation Set = 0.8 : 0.2 비율로 분할하여 사용

```python
# 모든 XML 파일에 대한 라벨 변환
xml_files = glob.glob(os.path.join(voc_annotations, "*.xml"))
valid_images = []

for xml_file in xml_files:
    base_name = os.path.splitext(os.path.basename(xml_file))[0]
    img_file = os.path.join(voc_images, f"{base_name}.jpg")

    # 이미지 파일이 존재하고 라벨 변환에 성공한 경우만 유효
    if os.path.exists(img_file) and convert_voc_to_yolo(xml_file, temp_labels_dir):
        valid_images.append(base_name)

print(f"총 {len(valid_images)}개의 유효한 이미지와 라벨을 찾았습니다.")

# train/val 분할 (80/20)
random.shuffle(valid_images)
split_idx = int(len(valid_images) * 0.8)
train_images = valid_images[:split_idx]
val_images = valid_images[split_idx:]

print(f"학습용: {len(train_images)}개, 검증용: {len(val_images)}개")

# 이미지와 라벨 파일 복사
for img_set, subset in [(train_images, "train"), (val_images, "val")]:
    for img_name in img_set:
        # 이미지 복사
        src_img = os.path.join(voc_images, f"{img_name}.jpg")
        dst_img = os.path.join(yolo_dataset, "images", subset, f"{img_name}.jpg")
        shutil.copy(src_img, dst_img)

        # 라벨 복사
        src_label = os.path.join(temp_labels_dir, f"{img_name}.txt")
        dst_label = os.path.join(yolo_dataset, "labels", subset, f"{img_name}.txt")
        if os.path.exists(src_label):
            shutil.copy(src_label, dst_label)
```

3. Train/Test 데이터셋 분할 및 data.yaml 구성

- YOLOv5가 참고할 data.yaml을 작성해야 함
- Ultralytics에서 권장하는 방식으로, 'train' 키에 train.txt 경로, 'val' 키에 val.txt 경로를 기입하여 진행
- 클래스 이름은 Pascal VOC 20개로 설정

5. data.yaml 파일 생성

```
def create_data_yaml():
    # 경로 내 백슬래시 처리 (백슬래시를 슬래시로 변경)
    dataset_path = yolo_dataset.replace('\\', '/')

    yaml_content = .format(dataset_path, len(voc_classes), voc_classes)

    yaml_path = os.path.join(yolo_dataset, "data.yaml")
    with open(yaml_path, "w") as f:
        f.write(yaml_content)

    return yaml_path

# 전체 과정 실행
print("VOC 데이터셋을 YOLO 형식으로 변환 중...")
train_images, val_images = setup_yolo_dataset()
yaml_path = create_data_yaml()
print(f"변환 완료! data.yaml 파일: {yaml_path}")
```

```
# 클래스 명
names:
  0: aeroplane
  1: bicycle
  2: bird
  3: boat
  4: bottle
  5: bus
  6: car
  7: cat
  8: chair
  9: cow
  10: diningtable
  11: dog
  12: horse
  13: motorbike
  14: person
  15: pottedplant
  16: sheep
  17: sofa
  18: train
  19: tvmonitor
"""
```

4. YOLOv5 모델 학습

- YOLOv5 모델의 가중치를 로드하고 위에서 만든 data.yaml 파일을 통해 학습 준비
- Epochs 수, batch 사이즈, 입력 이미지 크기 등을 설정하고 학습 진행
- 사전에 분류된 항목별로 분류된 결과와 성능측정 결과를 확인

6. YOLOv5 학습 (Ultralytics)

```python
from ultralytics import YOLO
import os
import torch

# 모델 로드
model = YOLO("yolov5s.pt")
print("YOLOv5 모델 로드 완료!")

# 학습 - GPU 사용
model.train(
    data=os.path.join(yolo_dataset, "data.yaml"),
    epochs=5,
    batch=4,
    imgsz=416,
    name="yolov5_voc_demo",
    device=0,
    workers=0
)
```

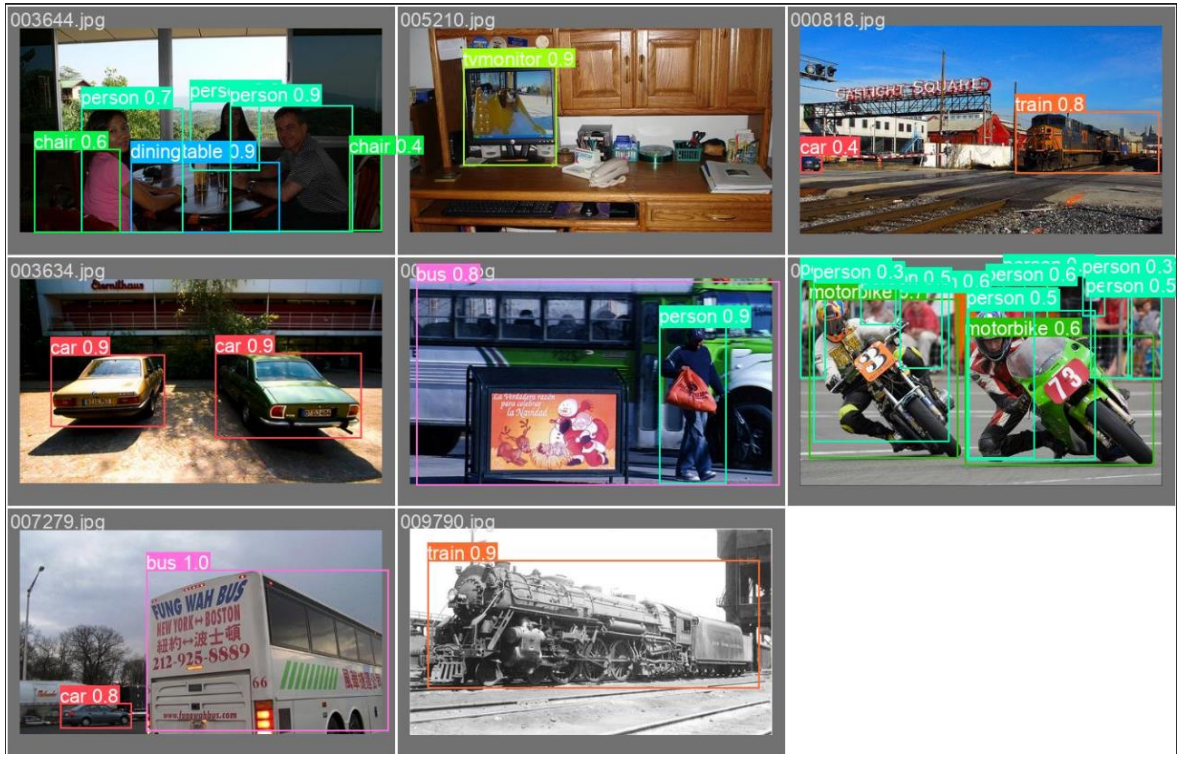
```
Validating runs/detect/yolov5_voc_demo2/weights/best.pt...
Ultralytics 8.3.91 🚀 Python-3.9.21 torch-2.6.0+cu118 CUDA:0 (NVIDIA GeForce RTX 3090, 24253MiB)
YOLOv5s summary (fused): 84 layers, 9,119,276 parameters, 0 gradients, 23.9 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████|
                   all       1003       2621      0.753       0.71      0.759      0.557
             aeroplane         44         57      0.892      0.874      0.943      0.754
               bicycle         46         60      0.765      0.883      0.884      0.642
                  bird         81        112      0.801      0.661      0.709      0.523
                  boat         32         57      0.557      0.579       0.53      0.323
                bottle         53         87      0.778       0.46      0.615      0.441
                   bus         41         53       0.71       0.74      0.823       0.64
                   car        147        259      0.885      0.784      0.896      0.676
                   cat         57         62      0.952      0.645      0.793      0.594
                 chair         89        155      0.731      0.555      0.657      0.449
                   cow         30         45       0.72      0.689      0.751      0.574
           diningtable         40         44      0.686      0.793      0.763      0.573
                   dog         86        103      0.687      0.524      0.654      0.515
                 horse         59         82      0.789      0.878      0.888      0.688
             motorbike         47         71       0.74      0.803      0.811      0.531
                person        414       1028      0.834      0.777      0.865      0.578
           pottedplant         50        120      0.472      0.525      0.412      0.238
                 sheep         23         62      0.767      0.742      0.785      0.564
                  sofa         42         45      0.615      0.644      0.632      0.475
                 train         52         58      0.951      0.828      0.934        0.7
             tvmonitor         52         61       0.73       0.82      0.836      0.664
Speed: 0.3ms preprocess, 1.0ms inference, 0.0ms loss, 1.2ms postprocess per image
Results saved to runs/detect/yolov5_voc_demo2
```

5. 모델 검증 및 추론

- 학습 결과로 나온 .pt 파일을 불러와서 검증 과정 진행
- 이미지 디렉토리에서 이미지 파일 목록을 가져오고 첫 번째 이미지 파일을 사용하여 모델로 예측 수행
- 시각화를 통해 이미지 데이터에서 처리된 객체 감지 결과를 확인

7. 검증(Validation) 및 추론 테스트

```python
# 학습 결과로 나온 best.pt 모델을 불러와 val 진행
results = model.val()
print(results)

# 추론 테스트
import cv2
import matplotlib.pyplot as plt

# 테스트 이미지 경로 설정
# 직접 이미지 ID 지정
image_dir = "/userHome/userhome1/chaewoon/VOCdevkit/VOC2007/JPEGImages"

# 이미지 디렉토리에서 이미지 파일 목록 가져오기
import glob
image_files = glob.glob(os.path.join(image_dir, "*.jpg"))

if not image_files:
    print(f"경로에 이미지 파일이 없습니다: {image_dir}")
else:
    # 첫 번째 이미지 파일 사용
    sample_img_path = image_files[0]
    sample_img_id = os.path.splitext(os.path.basename(sample_img_path))[0]

    print(f"테스트할 이미지: {sample_img_path}")

    # 모델로 예측 수행
    preds = model.predict(source=sample_img_path, conf=0.25, save=True)
    output_img_path = preds[0].path

    # 시각화
    img_result = cv2.imread(output_img_path)
    img_result = cv2.cvtColor(img_result, cv2.COLOR_BGR2RGB)

    plt.figure(figsize=(8,8))
    plt.imshow(img_result)
    plt.axis("off")
    plt.title("YOLOv5 Inference on Pascal VOC")
    plt.show()
```

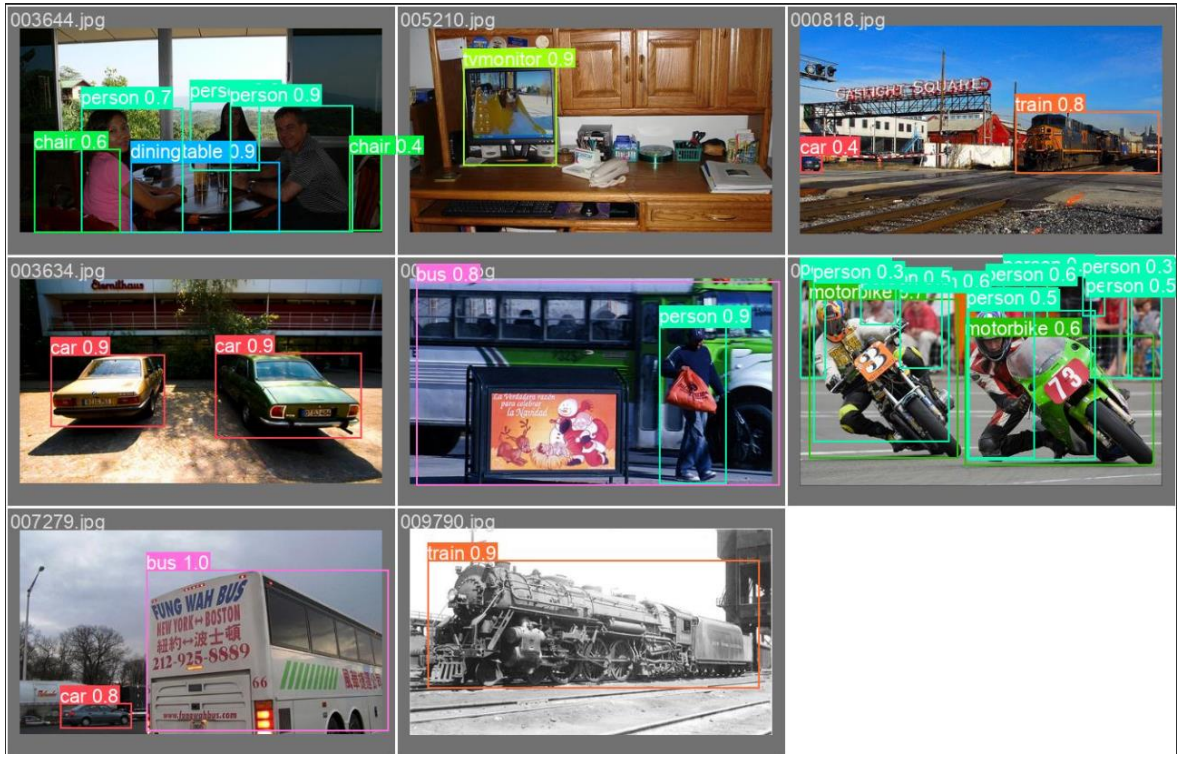


5. 모델 검증 및 추론

- 학습 결과로 나온 .pt 파일을 불러와서 검증 과정 진행
- 이미지 디렉토리에서 이미지 파일 목록을 가져오고 첫 번째 이미지 파일을 사용하여 모델로 예측 수행
- 시각화를 통해 이미지 데이터에서 처리된 객체 감지 결과를 확인

7. 검증(Validation) 및 추론 테스트

```python
# 학습 결과로 나온 best.pt 모델을 불러와 val 진행
results = model.val()
print(results)

# 추론 테스트
import cv2
import matplotlib.pyplot as plt

# 테스트 이미지 경로 설정
# 직접 이미지 ID 지정
image_dir = "/userHome/userhome1/chaewoon/VOCdevkit/VOC2007/JPEGImages"

# 이미지 디렉토리에서 이미지 파일 목록 가져오기
import glob
image_files = glob.glob(os.path.join(image_dir, "*.jpg"))

if not image_files:
    print(f"경로에 이미지 파일이 없습니다: {image_dir}")
else:
    # 첫 번째 이미지 파일 사용
    sample_img_path = image_files[0]
    sample_img_id = os.path.splitext(os.path.basename(sample_img_path))[0]

    print(f"테스트할 이미지: {sample_img_path}")

    # 모델로 예측 수행
    preds = model.predict(source=sample_img_path, conf=0.25, save=True)
    output_img_path = preds[0].path

    # 시각화
    img_result = cv2.imread(output_img_path)
    img_result = cv2.cvtColor(img_result, cv2.COLOR_BGR2RGB)

    plt.figure(figsize=(8,8))
    plt.imshow(img_result)
    plt.axis("off")
    plt.title("YOLOv5 Inference on Pascal VOC")
    plt.show()
```



Thank you