Object Detection

이재형

국민대학교 전자공학부

harry8121@kookmin.ac.kr

요 약

TorchVision에서 제공하는 사전 훈련된 객체 탐지 모델을 통해 특정 이미지를 로드하여 객체를 탐지하고, Bounding Box와 레이블, 신뢰도를 이미지 위에 표시하여 출력하였습니다.

I. 서론

이번 Object Detection과제에 사용되는 pytorch의 torchvision 라이브러리에는 pre-trained된 객체 탐지 모델들이 포함되어 있습니다.

객체 탐지 모델은 이미지 내에서 특정 객체들의 위치를 식별하고, 해당 객체들에 대한 Bounding box와 클래스 레이블, 신뢰도 등을 출력합니다. 이러한 모델들은 대부분 딥러닝 기반이며, 대량의 이미지 데이터로 사전 훈련된 가중치를 지니고 있습니다.

torchvision에서 제공하는 객체 탐지 모델 중 Faster R-CNN모델을 사용하여 진행했습니다. 이는 높은 정확도와 빠른 속도를 가지고 있어 가장 대중적으로 사용되는 모델 중 하나입니다.

'이미지 로드 -> 사전 훈련 모델 로드 -> 객체 탐지 실행 -> 결과 표시 -> 탐지 신뢰도에 대한 임계값 적용' 위의 순서로 작성된 코드에 대한 전체적인 설명과 결과 이미지를 첨부하여 설명하겠습니다.

II. 과제 수행 내용

라이브러리

```
port torch
port torchvision.transforms as T
om torchvision.models.detection import fasterrcnn_resnet50_fp
om PIL import Image, ImageDraw
```

torch: pytorch의 핵심 라이브러리
torchvision.transforms : 이미지 변환을 위한 함수 제공 라이브러리
fasterrcnn_resnet50_fpn: torchvision으로부터 Faster R-CNN 모델 import
PIL: 이미지 처리와 조작을 위한 파이썬 라이브러리

이미지 로드

```
Step 1: Load the image using PIL
mage_path = "img_example.jpg"
mage = Image.open(image_path).convert("RG
```

image_path' 변수에 이미지 파일 경로를 지정한 후, mage_open()' 함수를 통해 이미지를 PIL객체로 열었니다. 'convert("RGB")'를 호출하여 이미지를 RGB형으로 변환했습니다.

3. Pre-trained 모델 로드

```
# Step 2: Load the pre-trained object detection
model = fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()
```

- torchvision에서 제공하는 사전 훈련된 Faster R-CNN모델입니다. 모델을 호출하고, 'model.eval()'을 통해 모델을 평가 모드로 설정합니다. 이는 모델의 학습에 사용되는 기능들을 비활성하고 결과를 도출하고, 추론을 위한 최적의 동작을 수행하기 위함입니다.

4. 로드된 이미지를 통한 객체 탐지

```
# Step 3: Run object detection on the loaded image
transform = T.Compose([T.ToTensor()])
image_tensor = transform(image)
predictions = model([image_tensor])
```

- **transform = T.Compose([T.ToTensor()])**: 이미지 변환을 위한 변환 함수를 정의했습니다. 이는 PIL이미지를 pytorch 텐서 형태로 변환하여 모델이 예상하는 입력 형식에 이미지를 맞춘 것입니다.

- **image_tensor**: transform을 통해 텐서로 변환한 결과를 저장합니다. 이는 모델의 입력으로 사용 가능한 이미지 텐서 형태입니다.

- **predictions**: 변환된 이미지 텐서를 모델에 전달하여 객체 탐지를 실행하여 객체 탐지 결과가 변수에 저장됩니다.

5. 객체 탐지 결과 표시 & 신뢰도 임계값 적용

```
# Step 4: Display object detection results
draw = ImageDraw.Draw(image)

# Step 5: Apply thresholding on detection confidence
threshold = 0.5  # Adjust the threshold value as needed

for box, label, score in zip(predictions[0]["boxes"], predictions[0]["labels"], predictions[0]["score
    if score >= threshold:
        box = box.tolist()  # Convert box tensor to a list
        draw.rectangle(xy=box, outline="yellow")
        draw.text((box[0], box[1]), str(label.item()), fill="red")
        draw.text((box[0], box[1] - 10), f"{score:.2f}", fill="red")

image.show()
```

- **ImageDraw.Draw()** : 이 함수를 통해 draw 객체를 생성합니다. 이 객체를 이용하여 이미지 위에 박스, 텍스트 등을 그릴 수 있게 됩니다.
for루프를 사용하여 predictions의 각 객체 탐지 결과에 대해 반복합니다.

- for문의 변수를 보면, bow, label, score 각각 바운딩 박스, 레이블, 신뢰도를 의미하는 변수입니다.
'predictions[0][]' 과 같이 인덱싱을 통해서 객체 탐지 결과 중 3가지 정보를 담고 있는 값으로 접근하여 반복합니다.

- **threshold** 변수를 설정하여 일정 신뢰도 이상을 가진 객체에 대해서만 결과를 시각화 할 수 있도록 합니다. 이는 for문 안의 if 조건문에 의해 실현됩니다.

- 신뢰도가 임계값 이상이라면 조건문안의 문장들이 실행됩니다.

- **box = box.tolist()**: 바운딩 박스를 텐서에서 리스트로 변환합니다. 이는 ImageDraw 모듈을 통해 바운딩 박스를 그릴 때 좌표 값을 용이하게 사용할 수 있도록 하기 위함입니다.

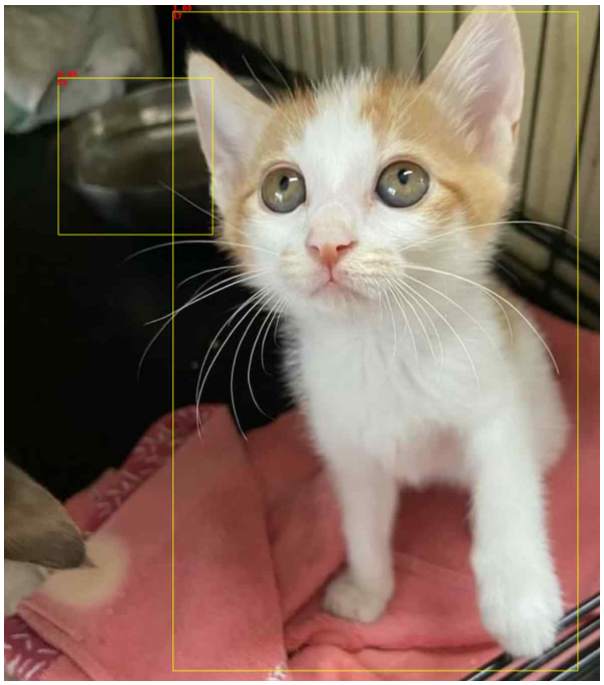
- **draw.rectangle(xy=box, outline="yellow")**: ImageDraw 객체를 이용하여 이미지 위에 바운딩 박스를 그립니다. 바운딩 박스의 좌표를 xy 매개변수로 받고, outline 매개변수로 선의 색상을 지정할 수 있습니다

- **draw.text((box[0], box[1]), str(label.item()), fill="red") :** 앞의 좌표에 레이블을 이미지 위에 표시합니다. label.item()을 문자열로 변환하고, fill 매개변수로 텍스트의 색상을 지정합니다.

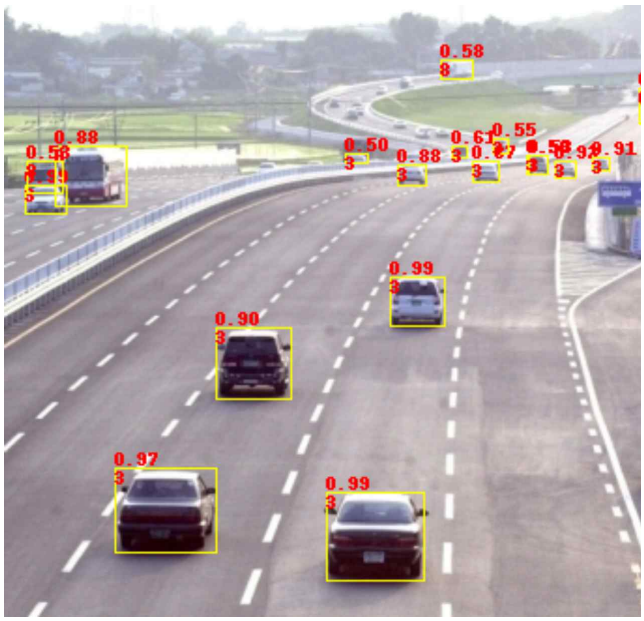
- **draw.text((box[0], box[1] - 10), f"{score:.2f}", fill="red") :** 이미지 위에 객체의 탐지 신뢰도를 표시합니다. 앞의 좌표에 신뢰도 값을 표시하고, f-string형식으로 신뢰도 값을 소수점 2자리까지 표시합니다.

3. 실험 결과 및 분석

threshold 값을 변경해가면서 몇가지 종류의 사진을 통해서 객체 탐지된 결과에 대해 서술하겠습니다.

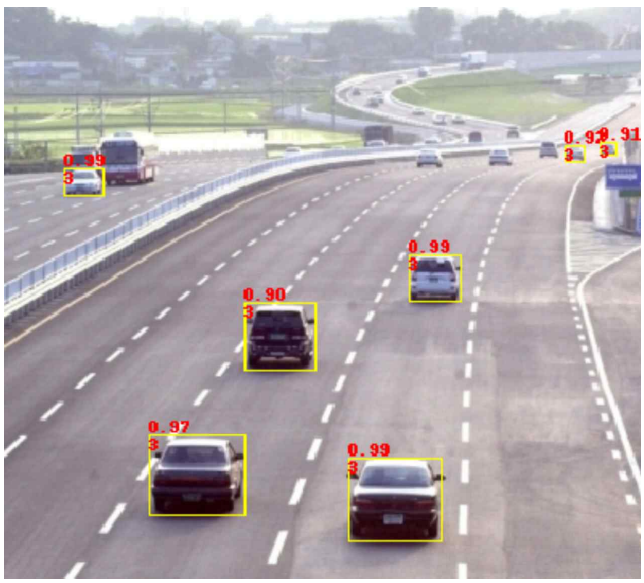


threshold = 0.5

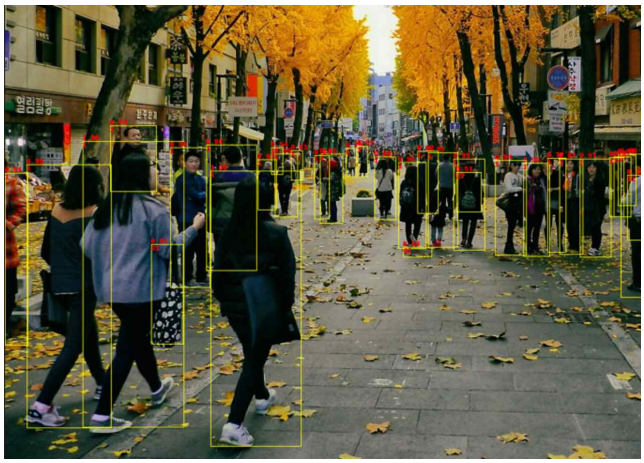


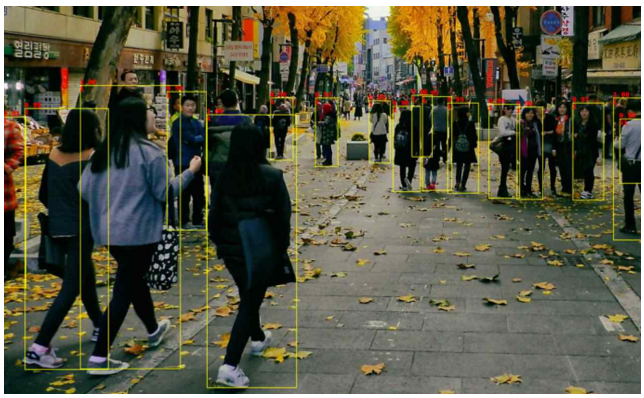
threshold = 0.5

threshold =0.9



threshold = 0.9


threshold = 0.5


threshold = 0.9

임계값이 낮을 수록 같은 이미지에서도 더 많은 객체가 탐지되는 것을 쉽게 확인할 수 있었습니다. 또한 Faster R-CNN 이름답게 객체 탐지의 속도가 상당히 우수하다는 것을 느낄 수 있었습니다.

4. 결론

pytorch에서 이렇게 수많은 데이터를 통해 사전 훈련된 모델 자체를 제공한다는 점과 이를 이용하여 손쉽게 객체를 탐지할 수 있다는 점이 좋았습니다.

하지만, 실제 모델을 학습하는 과정을 직접 경험해 본다면 데이터 전처리 과정에 상당히 오랜 시간이 걸린다는 것을 알고 있습니다.

이렇게 사전 훈련된 모델을 가져와서 사용하는 것이 편리하긴 하지만, 특정 상황에 따른 객체 탐지 성능을 비교하고 싶은 경우에는 직접 다양한 형태의 데이터를 학습시키는 것도 좋은 방법이라고 생각합니다.

참고문헌

[1] chatGPT

- 전체 코드 작성에 도움을 받았습니다.