




Airiss 인턴십

진민주

안녕하세요 에이리스에서 2달간 인턴십을 진행한 진민주입니다.
지금까지 진행한 것을 발표하도록 하겠습니다.



목차

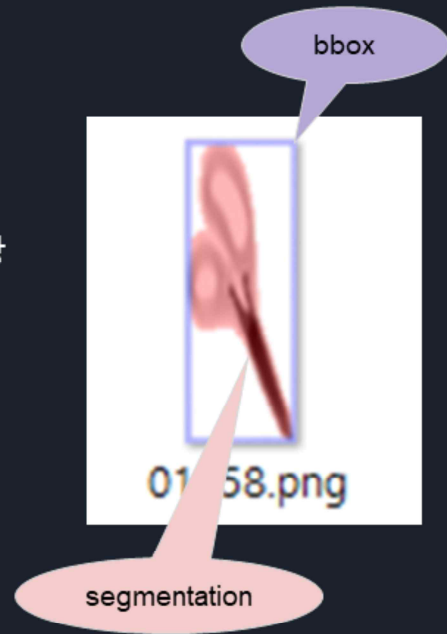
json 설명

1. augmentation
2. groundtruths
3. categorical
4. manipulation
5. jitter
6. Mask R-CNN

목차는 다음과 같습니다. 진행했던 순서대로 간단하게 발표하도록 하겠습니다.

json

1. segmentation: 겉 테두리 x, y 값이 순서대로 저장
2. bbox: x0, y0, w, h



프로젝트를 하면서 주어진 이미지의 정보들이 저장된 json 파일을 잘 변경하는 것이 중요했습니다.

json 정보 중에서도 제일 중요했던 부분이 segmentation과 bbox여서 해당 부분만 소개하도록 하겠습니다.

segmentation은 해당 이미지의 겉 테두리 x, y 좌표 값이 차례대로 저장되어 있고 bbox는 해당 이미지의 x, y, w, h 값이 저장되어 있습니다.

1. augmentation

! 주어진 이미지를 랜덤하게 resize해서 증강

```
fx = round(random.uniform(0.5, 1.5), 1)
```

```
fy = round(random.uniform(0.5, 1.5), 1)
```

```
update_img = cv2.resize(img, dsize=(0, 0), fx=fx, fy=fy,  
interpolation=cv2.INTER_LINEAR)
```

※ json update

1. segmentation: $\text{seg}(x) * fx, \text{seg}(y) * fy$
2. bbox: update_img의 shape 값으로 저장

먼저 주어진 이미지를 랜덤하게 resize해서 증강하는 것을 했습니다.

증강하는 방식은 (클릭)

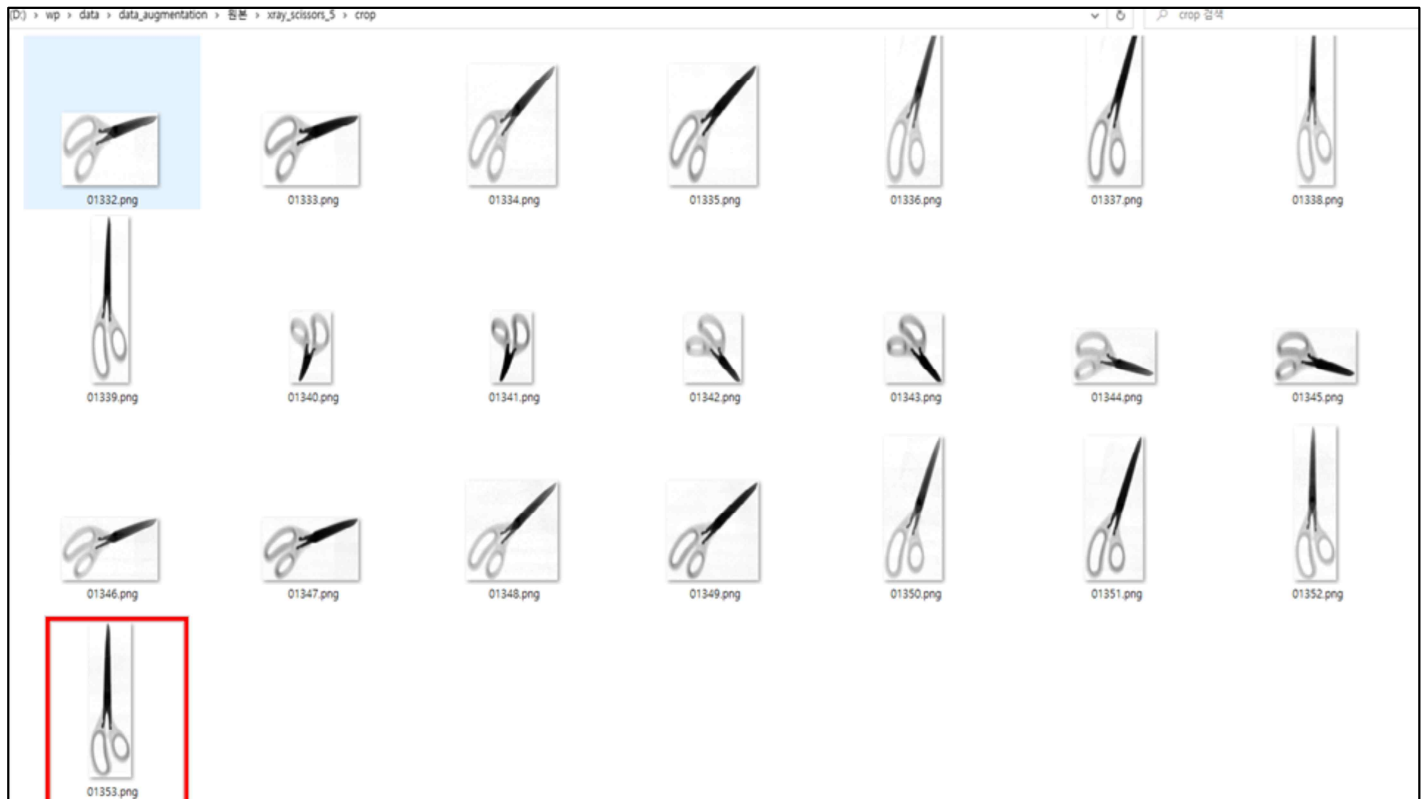
1. fx, fy를 랜덤하게 구하고,

(클릭)

1. 원본 이미지에서 fx, fy를 이용해서 상대크기로 변경했습니다.

(클릭)

json에 저장될 segmentation은 fx, fy을 이용한 해당 공식으로 저장하고, bbox는 resize한 이미지의 shape값으로 저장했습니다.

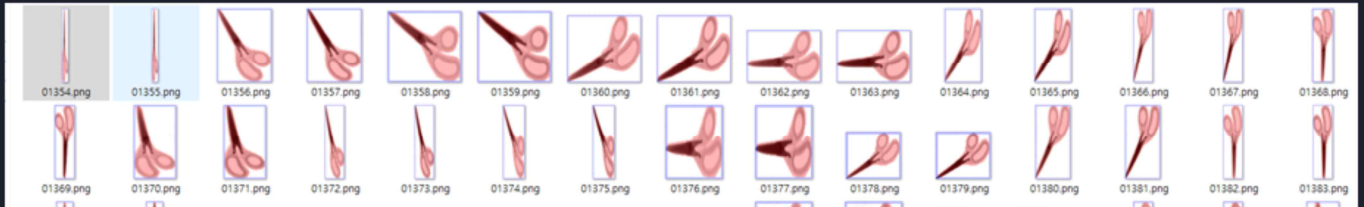


증강 시키기 전의 예제 폴더입니다. 현재 01353.png가 마지막에 있는 것을 볼 수 있습니다.



증강 시키면 원본 첫번째 이미지부터 **resize**한 이미지가 순서대로 저장됩니다.

2. groundtruths



```
# seg 칠하기
cv2.fillPoly(ground_truths_img, [seg], category_color[category_id-1])

# bbox 그리기
cv2.rectangle(ground_truths_img, (bbox[0], bbox[1]),
              (bbox[2]+bbox[0], bbox[3]+bbox[1]),
              category_color[category_id-1], 3)
```

증강한 이미지들의 json 파일이 잘 저장되었는지 segmentation과 bbox를 그려서 확인했습니다.

opencv의 fillpoly로 segmenation을 칠하고, rectangle로 bbox를 그렸습니다.



3. categorical

1. 카테고리 단일화

1: knife

- atknife, chefknife, fruitknife, jackknife, officeutilityknife, steakknife, swissarmyknife

2: gun

- gasgun, toygun

3: battery

4: laserpointer

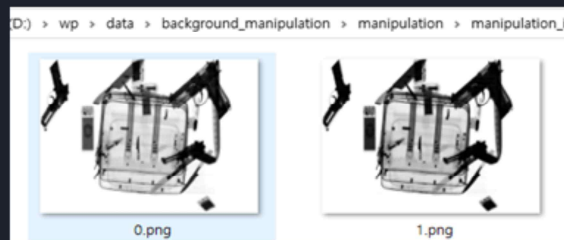
2. 카테고리당 이미지 10,000개를 resize해서 생성

다양한 이미지 중에서 knife, gun, battery, laserpointer만 사용하기 위해서 해당 카테고리들을 단일화했습니다.

그리고 카테고리 당 10,000개의 이미지를 앞서 한 것처럼 resize해서 생성했고 해당 이미지를 계속해서 사용했습니다.

4. manipulation

- 카테고리를 단일화 시킨 이미지와 background image들을 랜덤한 위치, 랜덤한 이미지로 합성
- 한 개의 background image에 카테고리 당 3개~5개, 총 12~20개의 카테고리 이미지를 합성



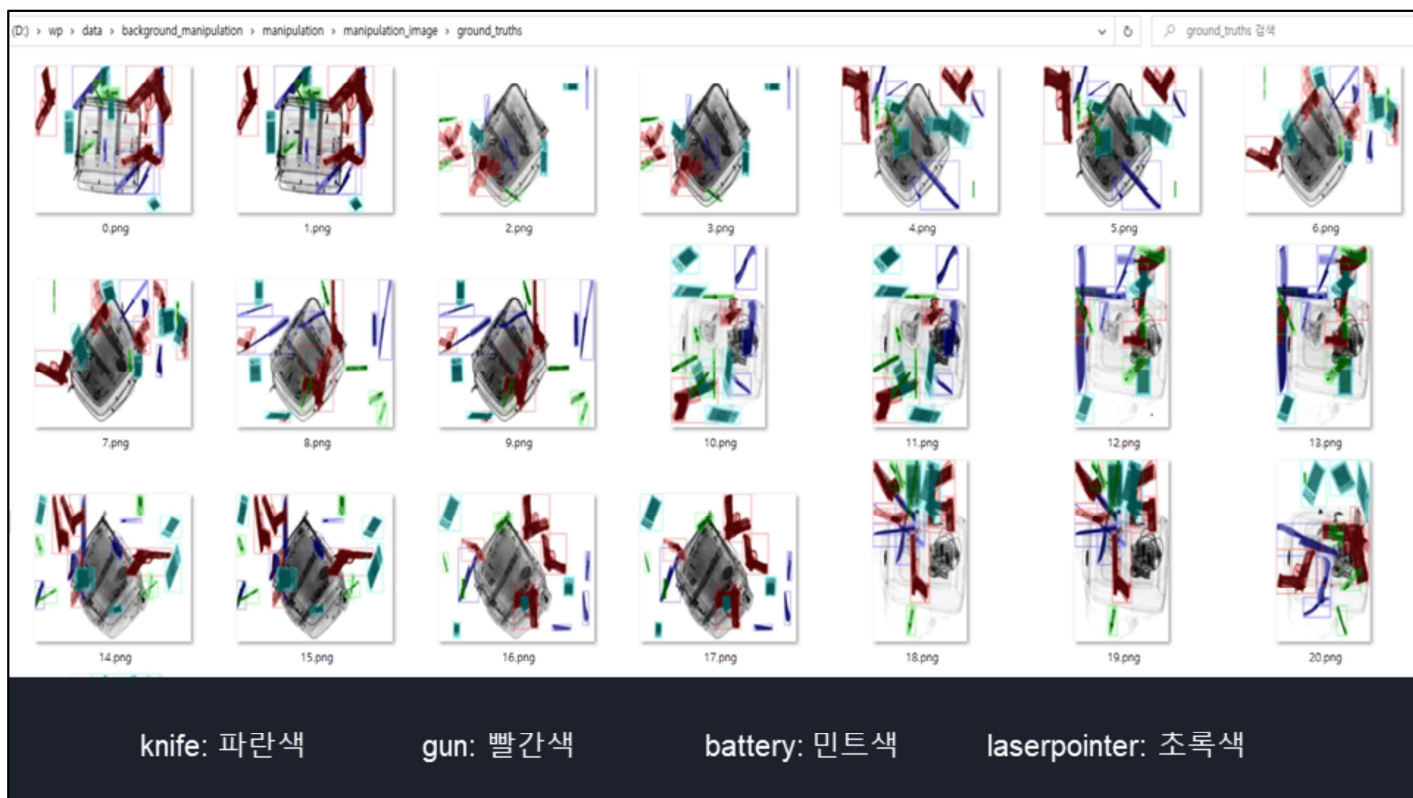
-high

- low

총 20,000장

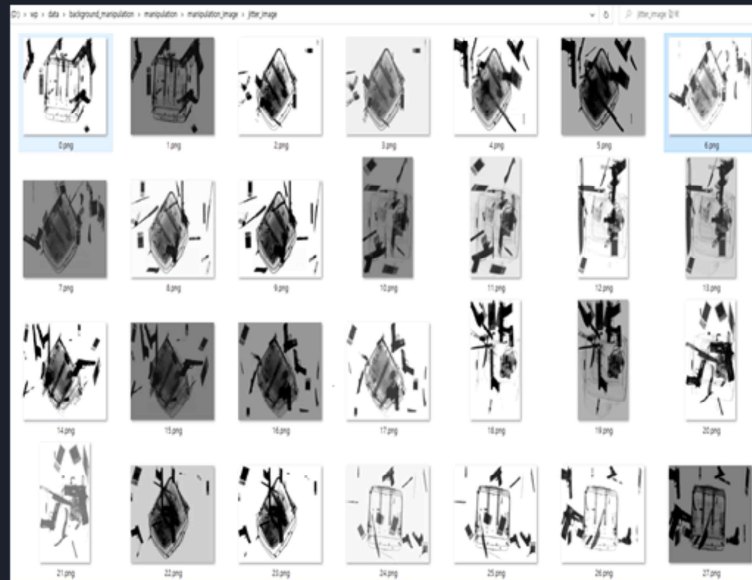
앞서 만든 이미지와 background image들을 랜덤한 위치와 랜덤한 이미지로 합성시켰습니다.
한 개의 background image에 카테고리 당 3개~5개의 카테고리 이미지를 합성시켰습니다.

그리고 사진처럼 high와 low 이미지는 동일한 위치에 합성해서 총 20,000장의 이미지를 만들었습니다.



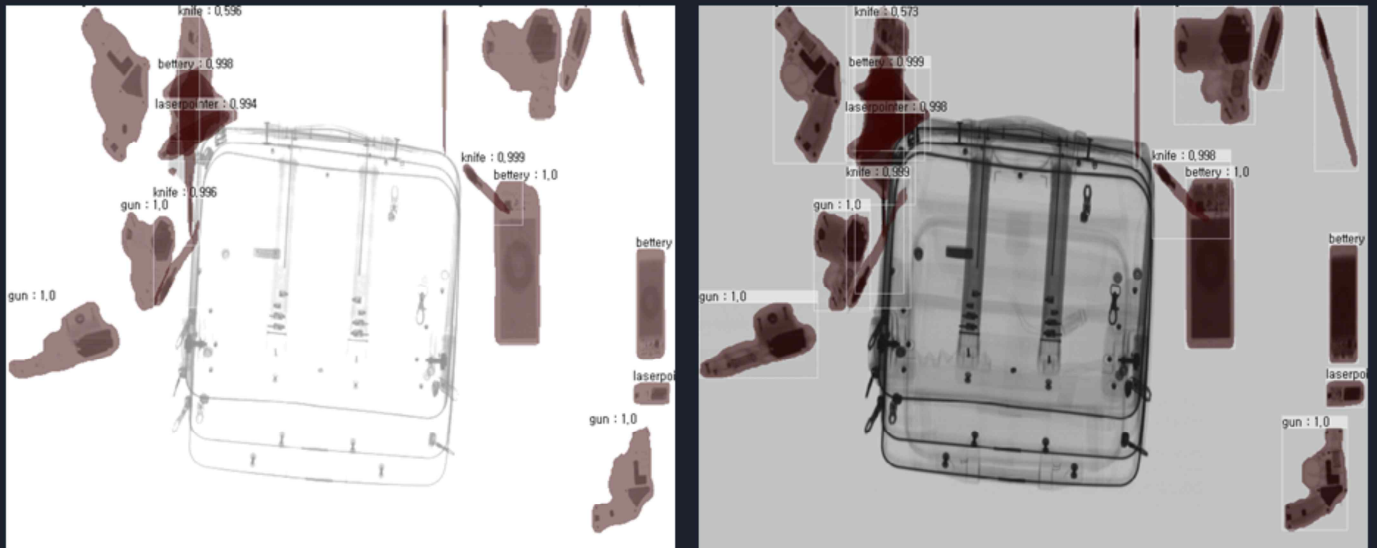
합성한 이미지도 seg, bbox를 그려서 확인했습니다.
파란색은 칼, 빨간색은 총, 배터리는 민트, 레이저포인트는 초록색으로 색칠했습니다.

5. jitter



그리고 해당하는 이미지들을 jitter를 이용해서 증강시켰습니다.

6. Mask R-CNN



이때까지 만든 이미지를 mask rcnn 모델을 사용해서 학습시켰습니다.
진행할 때 high low 이미지를 동시에 학습시키기도 하고 high 따로 low 따로 학습시키기도 했습니다.


해당 사진은 highlow 이미지를 같이 돌리고, train, validation, test을 8:1:1 비율로 나뉘었을 때의 결과입니다. 잘 찾아내는 것을 볼 수 있습니다.



그리고 만든 이미지가 아닌 다른 test 이미지로 탐지하기도 했습니다.
사진을 보시면 train, validation 이미지와 달라서 오 탐지되는 것이 많았습니다.

해당 하는 부분을 해결하고 싶었지만 아쉽게도 인턴십이 끝나서 여기까지 진행했습니다.

그리고 사진은 없지만 average precision과 recall을 구하기 위해서 P-R 곡선까지 그리는 것까지 진행했었습니다.



감사합니다

발표는 여기까지입니다. 들어주셔서 감사합니다.