




Airiss 인턴십

진민주

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



목차

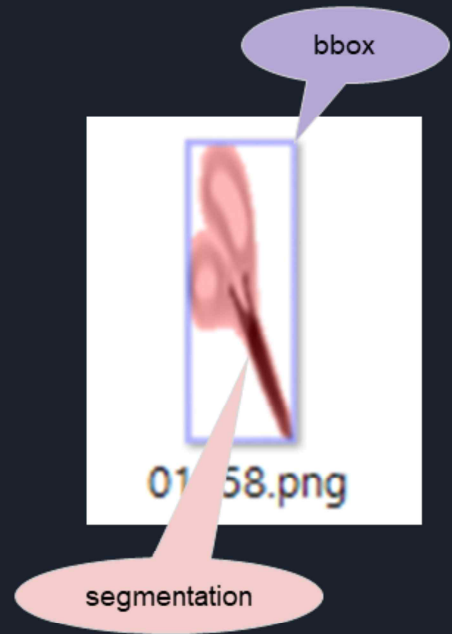
- 0. json 설명
- 1. Augmentation
- 2. groundtruths
- 3. categorical
- 4. manipulation
- 5. jitter
- 6. Mask R-CNN

목차는 다음과 같습니다. 진행했던 순서대로 간단하게 발표하도록 하겠습니다.

0. json

Annotation

1. segmentation: 겉 테두리 x, y 값이 순서대로 저장
2. bbox: x, y, width, height



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

Json에서 제일 중요했던 부분인 segmentation과 bbox를 소개하겠습니다.

segmentation은 해당 이미지의 겉 테두리 x, y 좌표 값이 차례대로 저장되어 있고
bbox는 해당 이미지의 x, y, width, height 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)

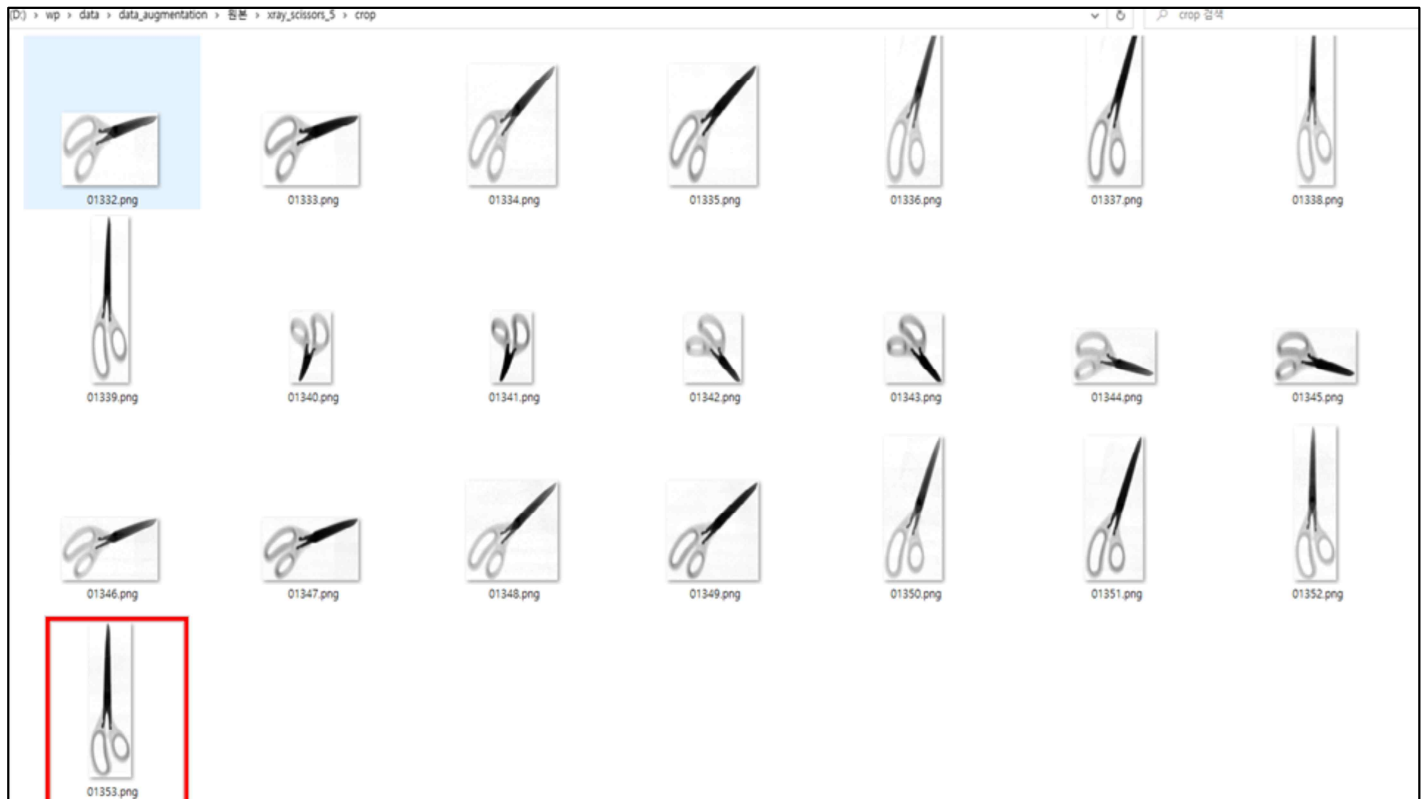
fx, fy를 랜덤하게 구하고,

(클릭)

원본 이미지에서 fx, fy를 이용해서 상대크기로 변경했습니다.

(클릭)

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

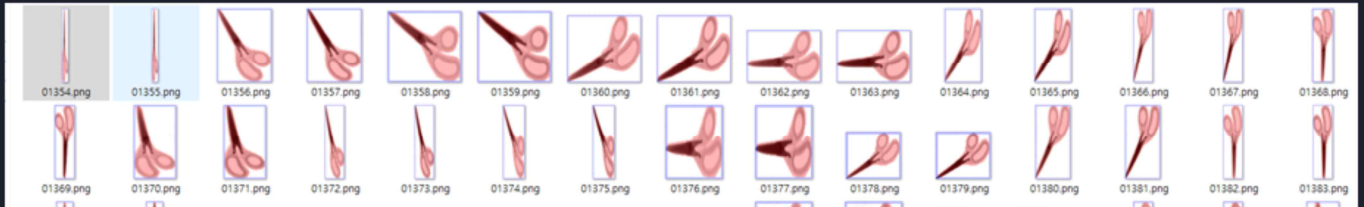


증강 시키기 전의 예제 폴더입니다. 현재 01353.png가 마지막에 있는 것을 볼 수 있습니다.



증강 시키면 원본 첫번째 이미지부터 **resize**한 이미지가
마지막 이름에서 1씩 더해 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*

- artknife, 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



앞서 만든 이미지와 background image

(클릭)

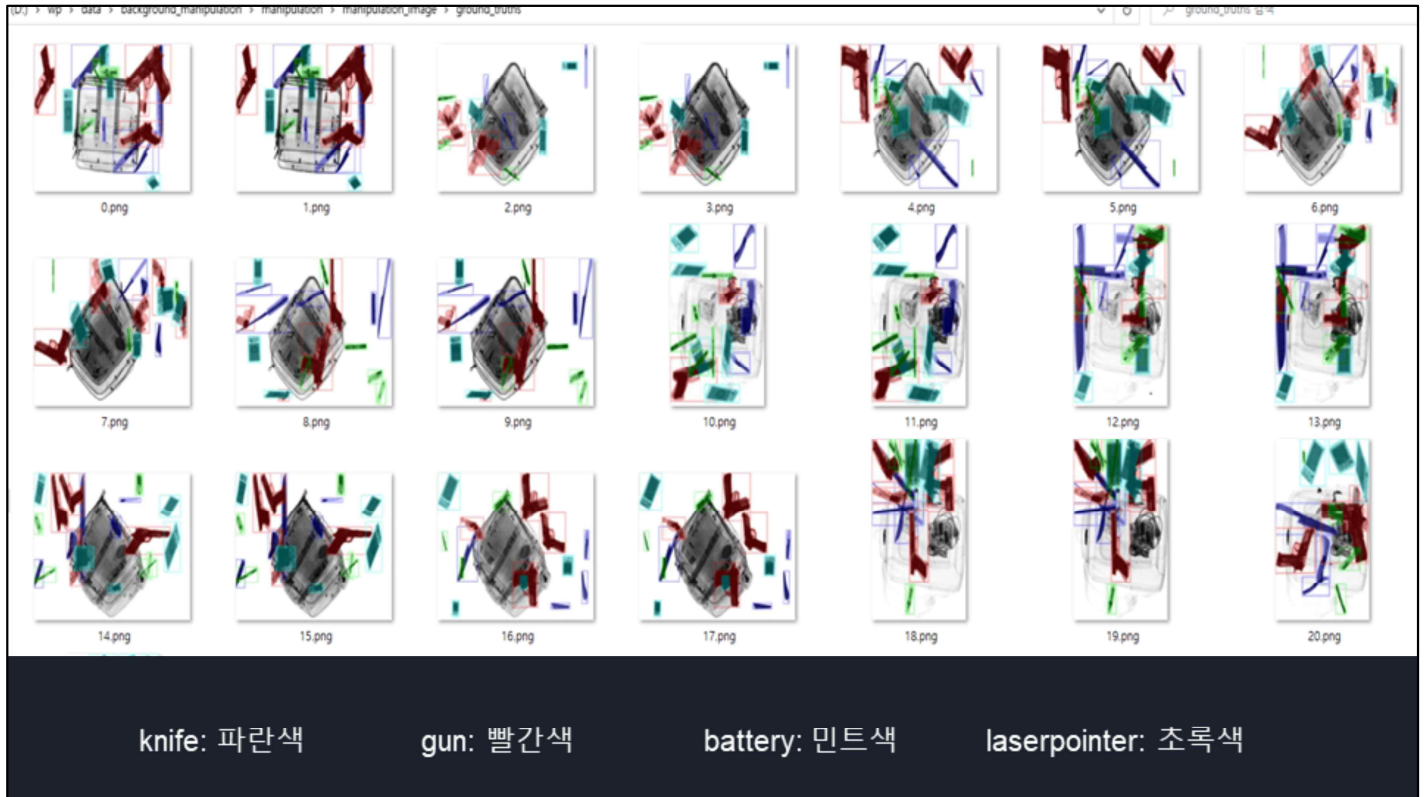
들을 랜덤한 위치와 랜덤한 이미지로 합성시켰습니다.

한 개의 background image에 카테고리 당 3개~5개의 카테고리 이미지를 합성시켰습니다.

(클릭)

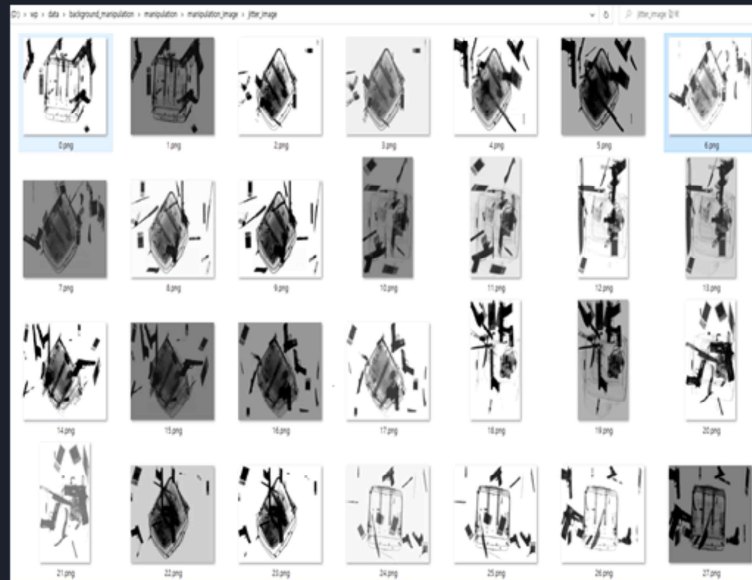
사진처럼 high와 low 이미지는 동일한 위치에 합성해서 총 20,000장의 이미지를 만들었습니다.

그리고 highlow 이미지 폴더, high만 있는 폴더, low만 있는 폴더를 만들어 각각 학습시켰습니다.



합성한 이미지도 카테고리 별로 색깔을 다르게 해서 seg, bbox의 값이 잘 저장되었는지 확인했습니다.

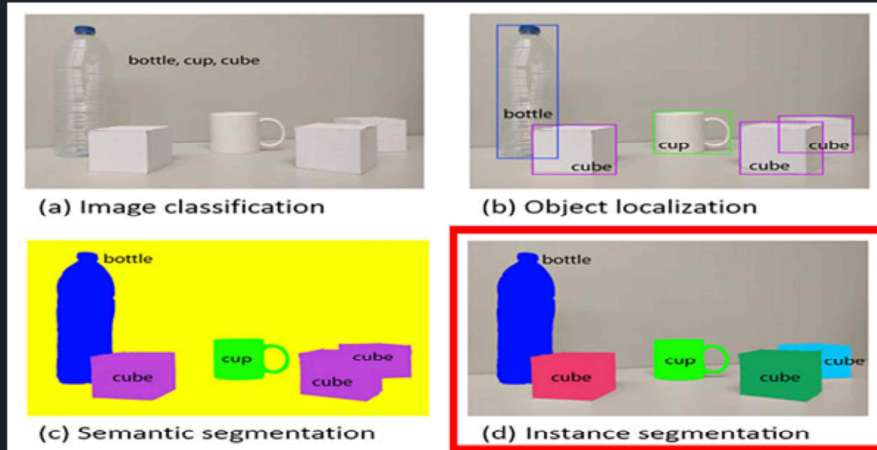
5. jitter



그리고 jitter를 이용해서 이미지를 변형시켰습니다.

6. Mask R-CNN

Instance segmentation?



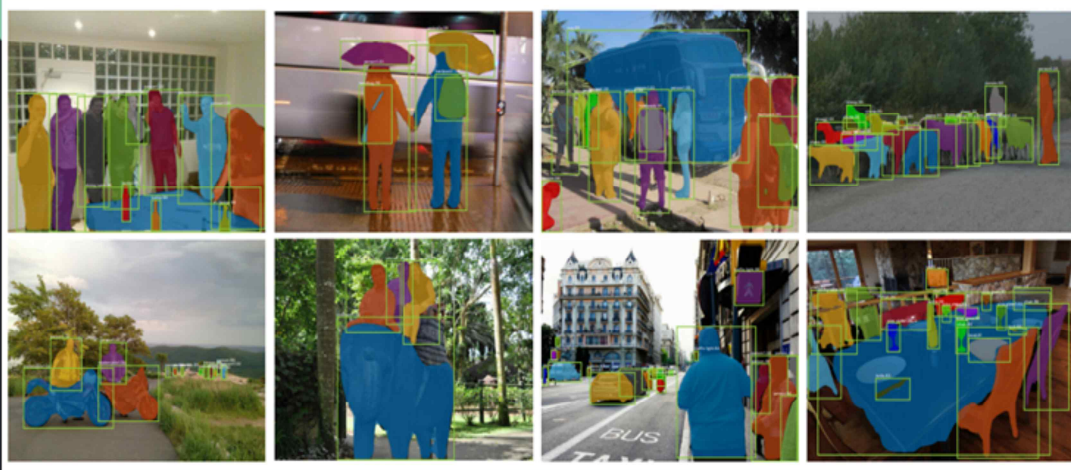
<https://ropiens.tistory.com/76>
<https://blablablab.tistory.com/139>
<https://herbwood.tistory.com/20>

이때까지 만든 이미지를 mask rcnn 모델을 사용해서 학습시켰습니다.

mask rcnn은 instance segmentation task에서 주로 사용됩니다.

instance segmentation이란 이미지 내에서 존재하는 모든 객체를 탐지하는 동시에 각각의 경우를 픽셀 단위로 분류하는 task입니다.

Mask R-CNN = Faster R-CNN + mask branch_(mask 예측)

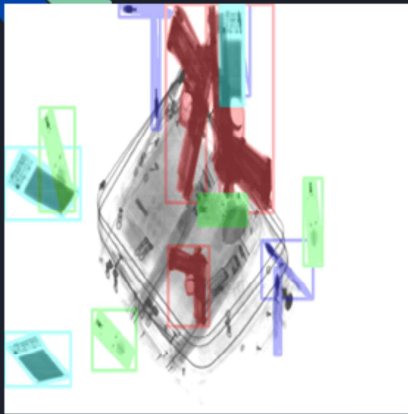


1. classification: 카테고리 별로 예측
2. Bounding box regression: 어디에 위치하는지 예측
3. Segmentation mask: 이미지에서 pixel 단위로 객체 추출

<https://ropiens.tistory.com/76>
<https://blablablab.tistory.com/139>
<https://herbwood.tistory.com/20>

이러한 Instance segmentation을 위해 faster rcnn에 mask를 예측할 수 있는 mask branch를 추가된 형태가 mask rcnn입니다.

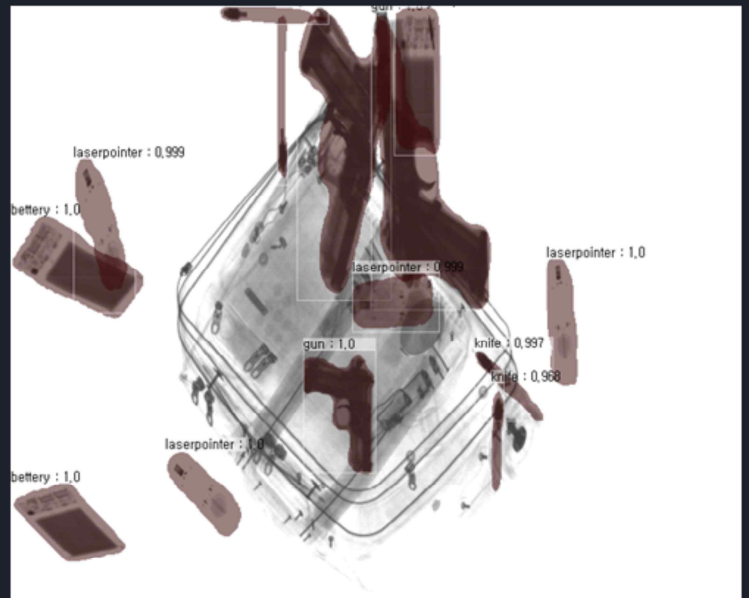
즉, mask rcnn은 Classification, box regression, segmentation mask를 동시에 처리할 수 있습니다.



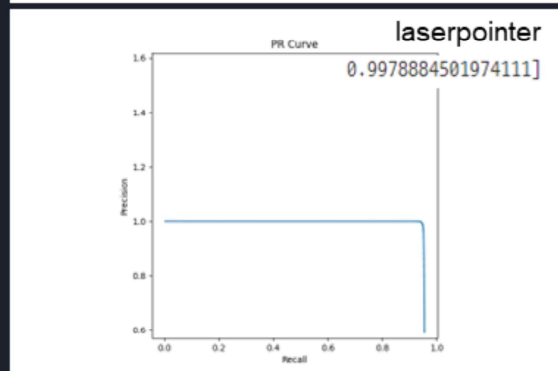
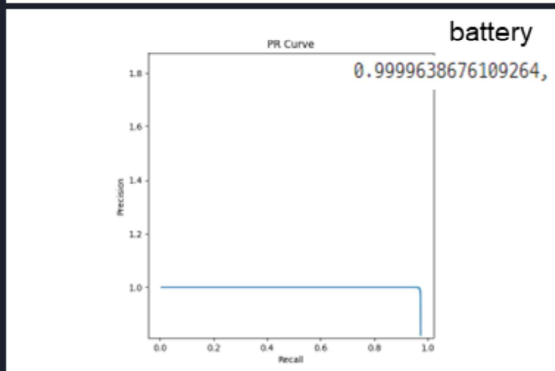
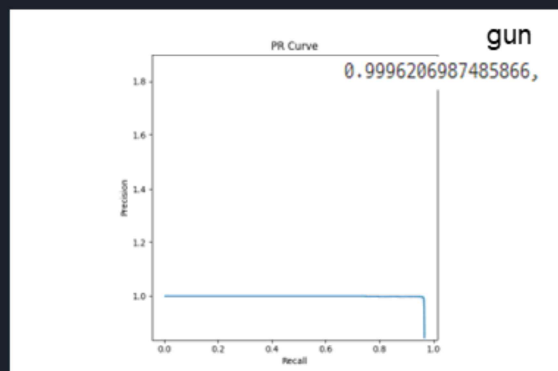
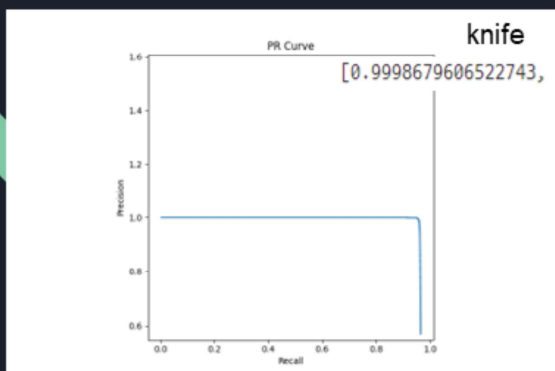
test 원본

1. HighLow & Jitter
2. Train : validation : test = 8 : 1 : 1
3. batch_size = 4 & lr=0.005

탐지



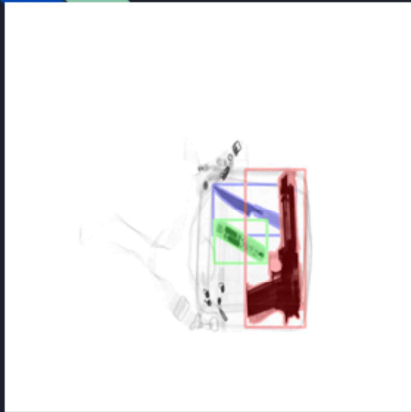
해당 사진은 Jitter 시킨 highlow 이미지를 8:1:1 비율로 나눠 학습하고 탐지했을 때 샘플 결과입니다.
해당 위치에 카테고리 별로 잘 검출해내는 것을 볼 수 있습니다.



P-R 곡선을 그렸을 때 다음과 같이 precision은 낮아지고 recall은 높아지는 것을 볼 수 있습니다.

(클릭)

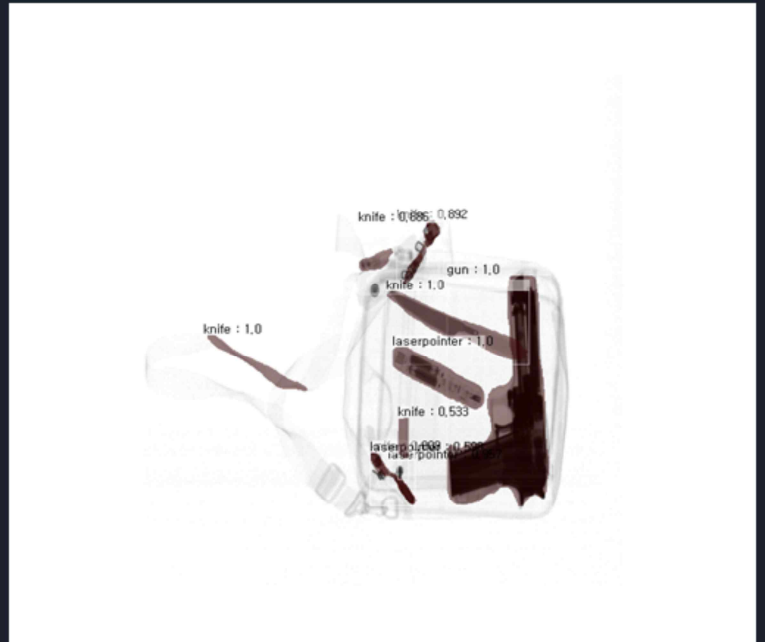
11점 보간법으로 AP를 구하면 평균적으로 0.9 값이 나옵니다.



test 원본

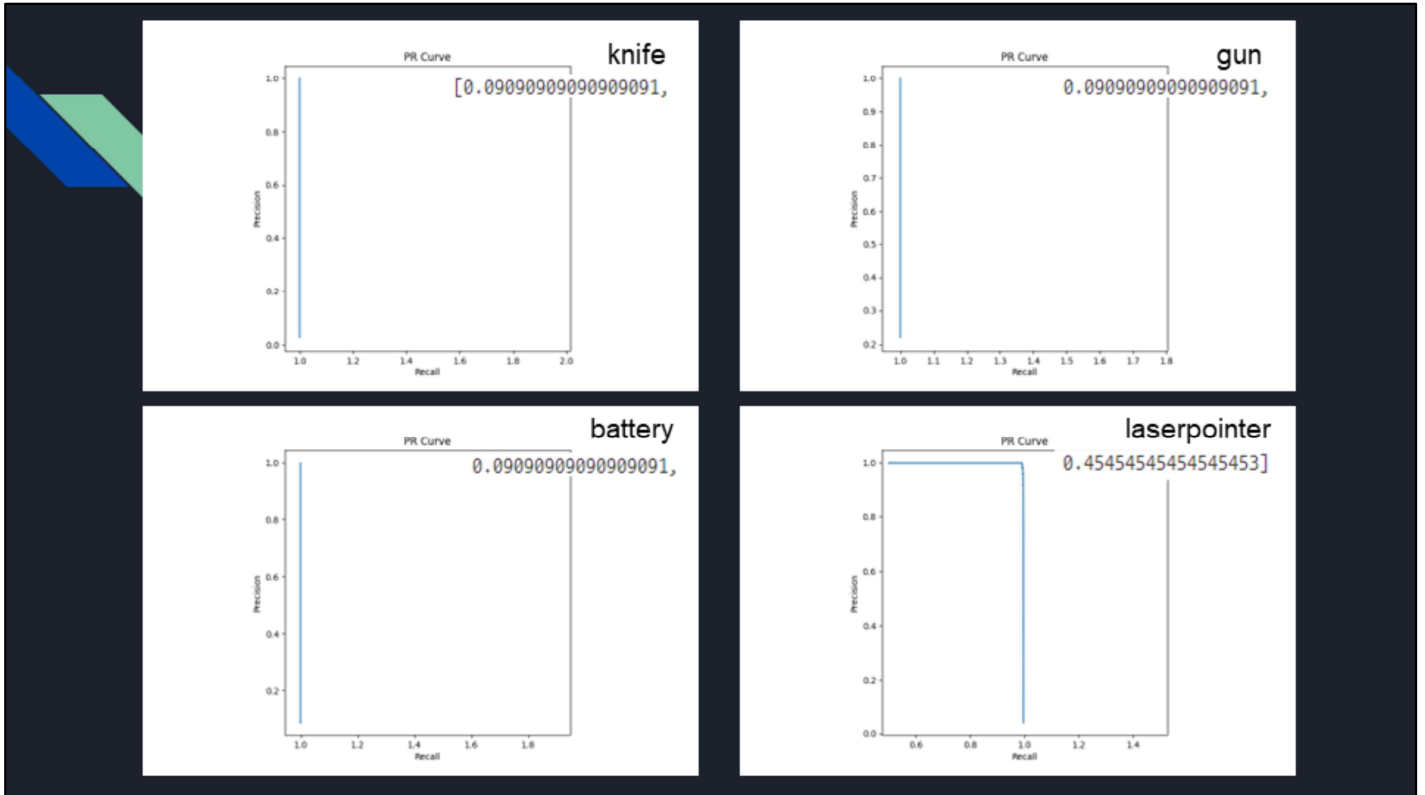
1. HighLow & Jitter
2. Train : validation = 9 : 1
3. batch_size = 4 & lr=0.005

탐지




방금과는 다르게 Train validation은 9:1 비율로 해서 모델을 만들었습니다.
그리고 만든 이미지가 아닌 다른 test 이미지로 탐지했습니다.

샘플 이미지를 보시면 원본에는 3개의 카테고리 이미지가 있는데 train, validation 이미지와 달라서 오 탐지되는 것이 많았습니다.



P-R곡선도 AP 값도 좋지 않게 나오는 것을 볼 수 있습니다.

해당 하는 부분을 해결하고 싶었지만 아쉽게도 인턴십이 끝나서 여기까지 진행했습니다.



감사합니다

발표는 여기까지입니다. 들어주셔서 감사합니다.