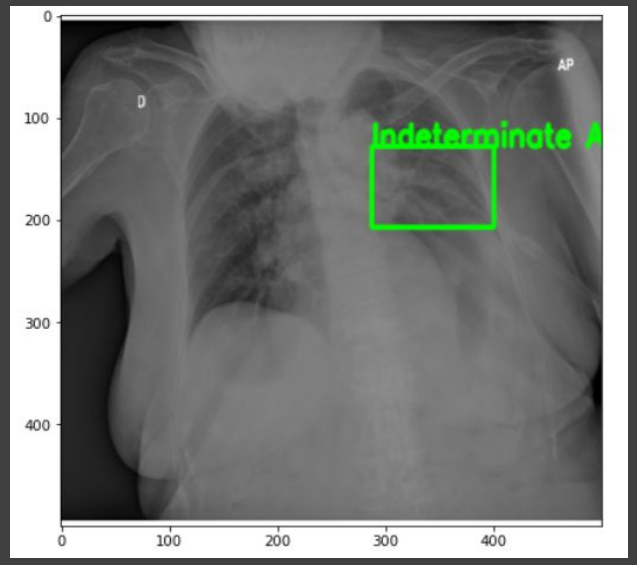
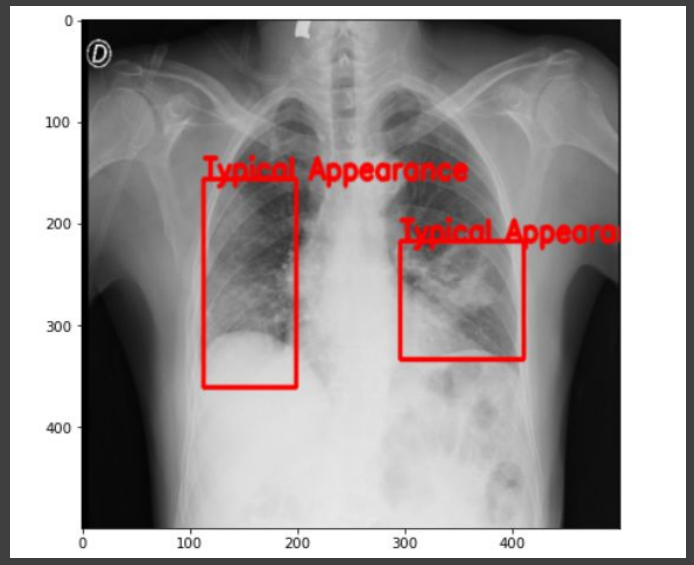
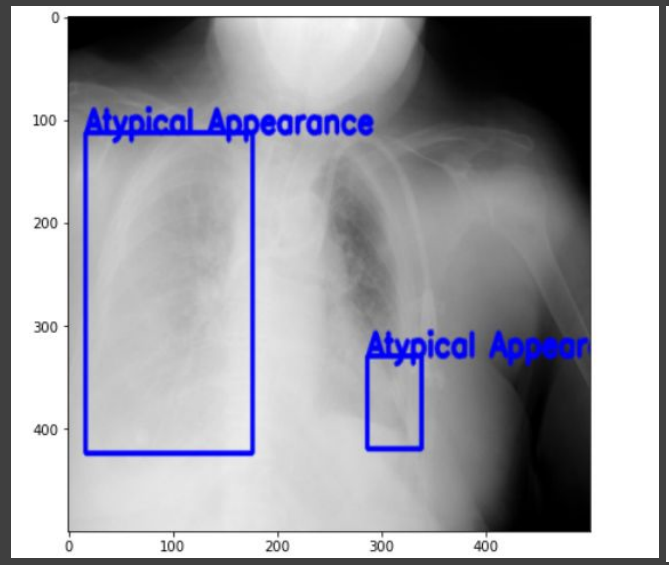
Faster IR-CNN을 이용한 폐 x-ray detection

CV 분반 : 구은아, 김혜림, 박상준, 이연정

프로젝트 아이디어


Indeterminate A


Typical Appearance
Typical Appeara


Atypical Appearance
Atypical Appear

프로젝트 아이디어

Mission 1

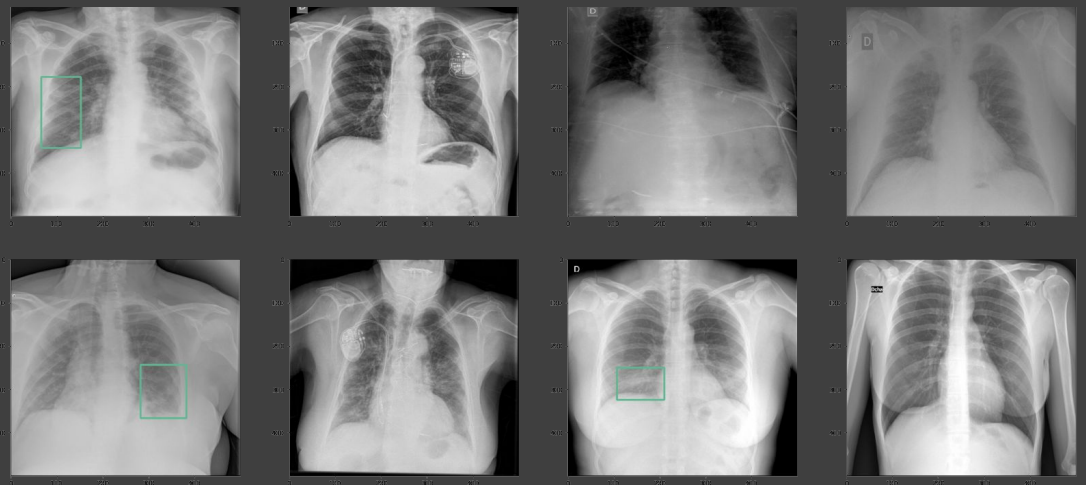
- x-ray 이미지를 학습해 이상이 있는 이미지(Opacity)와 없는 이미지(None) 구분
- 이상이 있는 경우 이상이 있는 곳의 위치 detection → *One box & Multi box*

Mission 2

- x-ray 이미지를 학습해 Negative for Pneumonia, Typical Appearance, Indeterminate Appearance, Atypical Appearance 중 어느 class에 속하는지 구분
- Negative for Pneumonia (이상 없음) 이 아닌 모든 경우 각각의 모습을 나타내는 부분 detection

  → *One box*

COVID-19 Dataset



6,334 chest scans in DICOM format

	id	boxes	label	StudyInstanceUID
2	000a312787f2_image	[{'x': 789.28836, 'y': 582.43035, 'width': 1026.65662, 'height': 1917.30292}, {'x': 2245.91208, 'y': 591.20528, 'width': 1094.66162, 'height': 1761.54944}]	opacity 1 789.28836 582.43035 1815.94498 2499.73327 opacity 1 2245.91208 591.20528 3340.5737 2352.75472	5776db0
3	000c3a3f293f_image		none 1 0 0 1 1	ff087
4	0012ff7358bc_image	[{'x': 677.42216, 'y': 197.97662, 'width': 867.79767, 'height': 999.78214}, {'x': 1792.69064, 'y': 402.5525, 'width': 617.02734, 'height': 1204.358}]	opacity 1 677.42216 197.97662 1545.21983 1197.75876 opacity 1 1792.69064 402.5525 2409.71798 1606.9105	9d514c
5	001398f4ff4f_image	[{'x': 2729, 'y': 2181.33331, 'width': 948.00012, 'height': 604}]	opacity 1 2729 2181.33331 3677.00012 2785.33331	28dddc8559b2
6	001bd15d1891_image	[{'x': 623.23328, 'y': 1050, 'width': 714, 'height': 1106}, {'x': 2578.56661, 'y': 998.66667, 'width': 662.66667, 'height': 1120}]	opacity 1 623.23328 1050 1337.23328 2156 opacity 1 2578.56661 998.66667 3241.23328 2118.66667	dfd9fdd85a3e
7	0022227f5adf_image	[{'x': 1857.2065, 'y': 508.30565, 'width': 376.02734, 'height': 399.52911}]	opacity 1 1857.2065 508.30565 2233.23384 907.83476	84543edc24c2
8	0023f02ae886_image		none 1 0 0 1 1	2fa400b873f5

Image Bounding box label
(box의 네 꼭짓점 좌표)

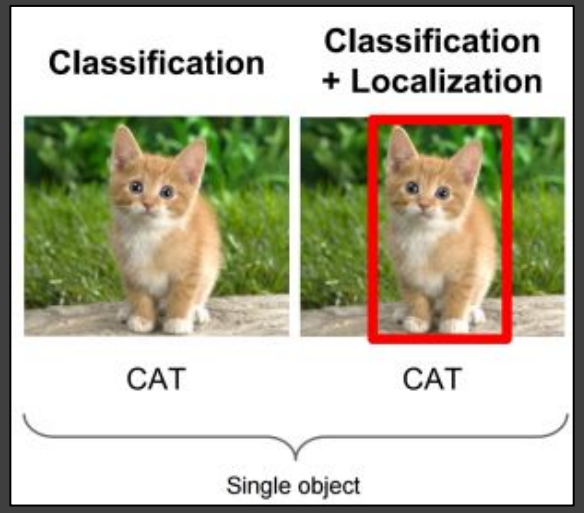
	id	Negative for Pneumonia	Typical Appearance	Indeterminate Appearance	Atypical Appearance
2	00086460a852_study	0	1	0	0
3	000c9c05fd14_study	0	0	0	1
4	00292f8c37bd_study	1	0	0	0
5	005057b3f880_study	1	0	0	0
6	0051d9b12e72_study	0	0	0	1

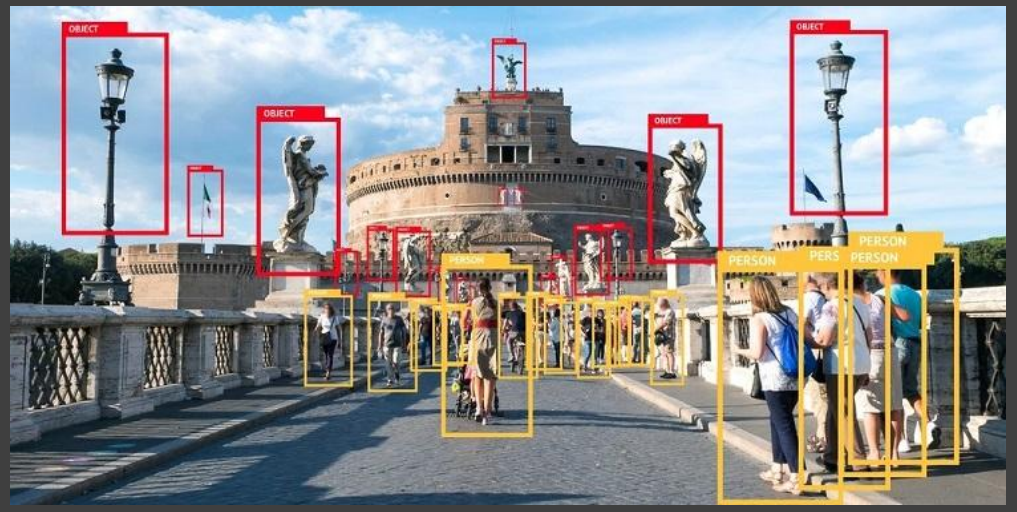
Image class label
(num_classes = 4)

Object detection 이란?

Object Detection

= Multi-Labeled Classification + Bounding Box Regression (Localization)





여러 물체에 대해 어떤 물체인지 분류하는 **Classification** 문제와 그 물체가 어디 있는지 박스를 통해

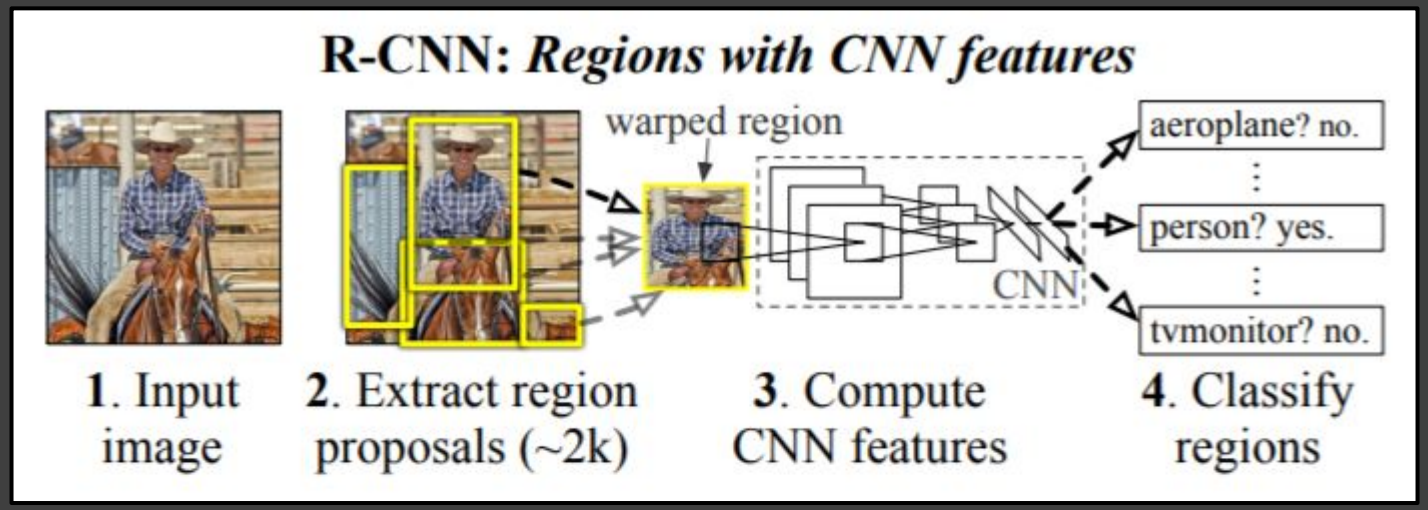
(Bounding box) 위치 정보를 나타내는 **Localization** 문제

사용한 모델 : Faster R-CNN

Faster R-CNN

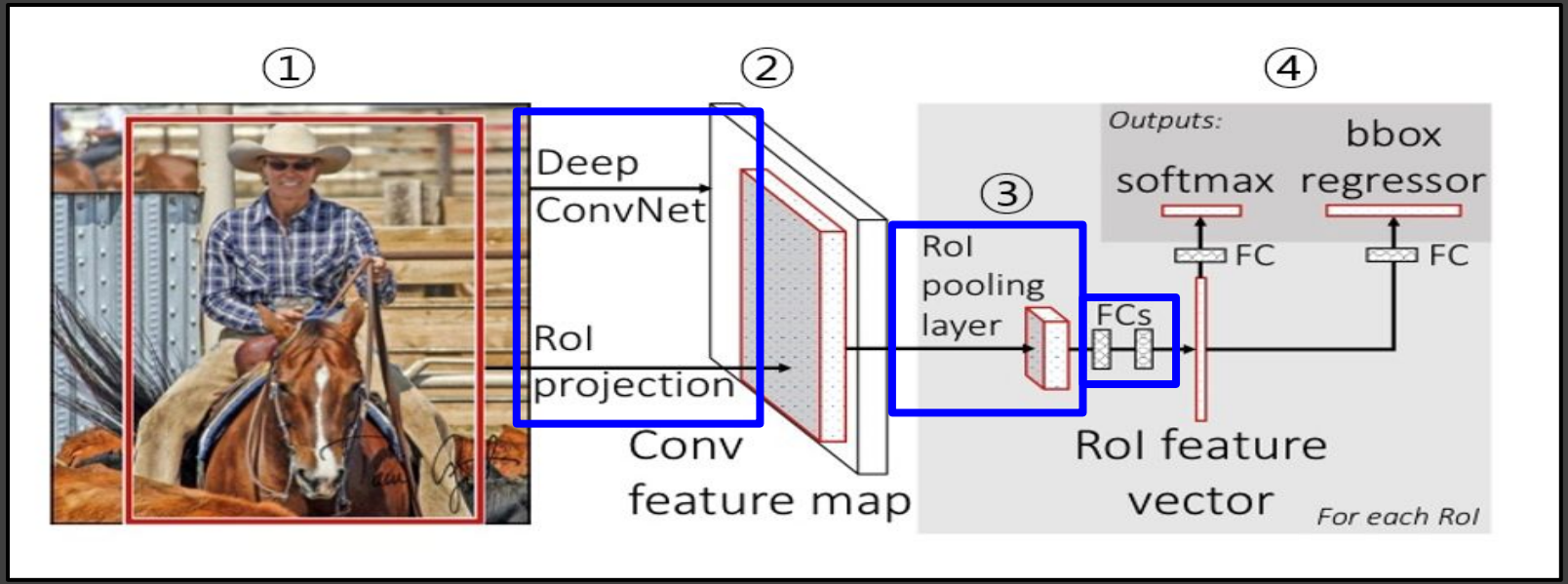
- Object detection에 사용되는 모델 중 하나
- 발전 과정

R-CNN : region-based convolutional network



1. 입력 : 이미지
2. bottom-up 기법을 이용해 region proposals (노란색 박스들) 추출 후 CNN의 입력 값에 맞춰 크기를 warping
3. CNN을 통해 region proposal 마다 feature를 계산한 후 고정된 길이의 feature vector 생성
4. 클래스마다 linear SVMs(Support Vector Machine)을 적용하여 각 region proposal의 클래스 분류
5. 출력 : 각 bounding box들과 이에 해당하는 클래스

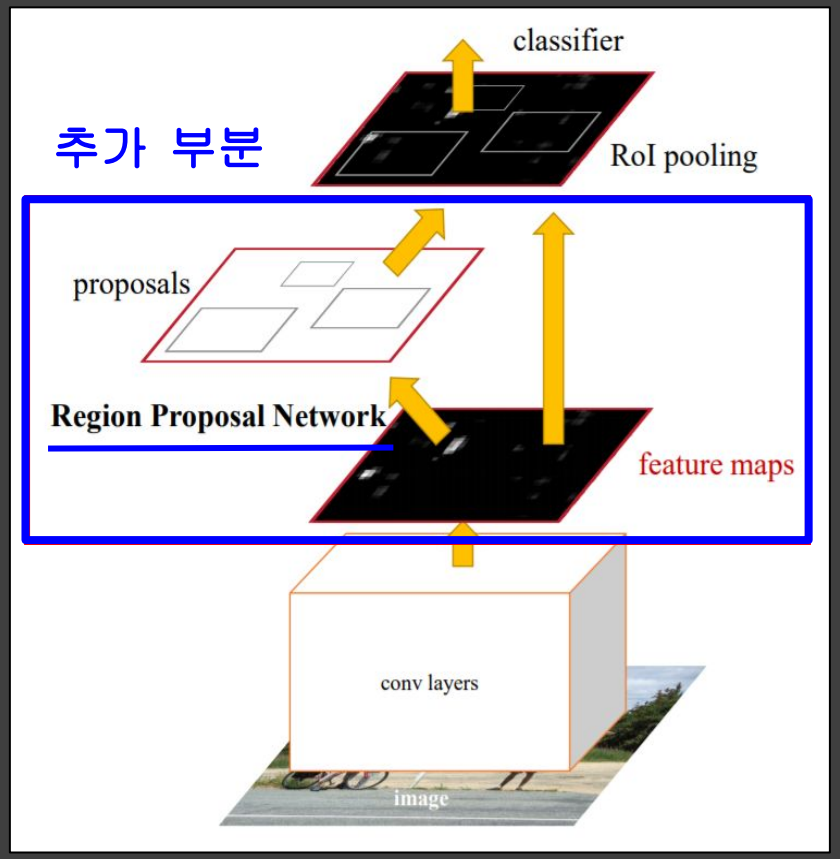
Fast R-CNN



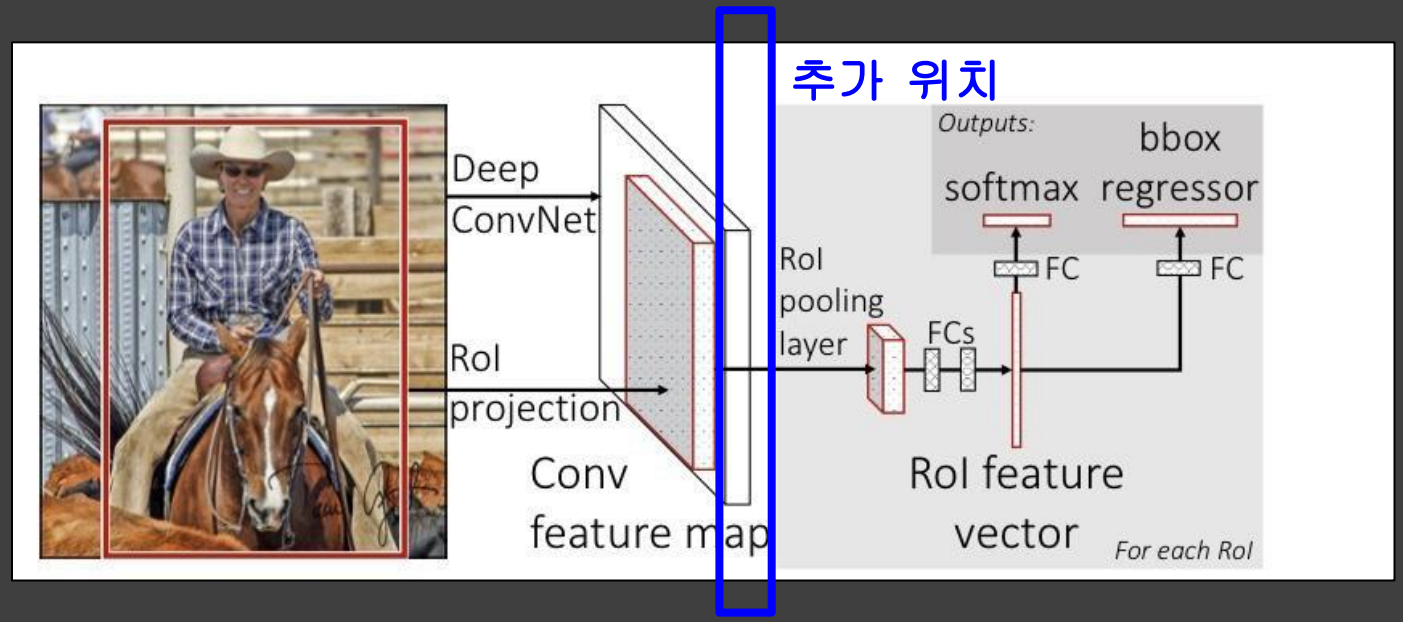
[ ] & 노란색 글자
: 기존의 R-CNN과의 차이점

1. 입력 : 이미지 전체 + object region proposal set
2. CNN을 통과시켜 feature map을 형성
3. feature map을 이용해 Region of interest(ROI) pooling layer에서 region proposal 마다 feature를 계산한 후 고정된 길이의 feature vector 생성
4. fully-connected(FC) layers에 feature vector를 통과시켜 각 region proposal의 클래스 분류
5. 출력 : (K개의 class + 1 background에 대한 softmax) & K개의 class 각각에 대한 bounding box 위치

Faster R-CNN



[ ] & 노란색 글자
: Faster R-CNN과의 차이점



- Faster R-CNN = RPN(region proposal network) + Fast R-CNN
- Fast R-CNN구조에서 CNN과 RoI Pooling사이에 RoI를 생성하는 Region Proposal Network 추가

입력

	id	boxes	label
3169	79ab8bbf9c7c_image	[{'x': 1806.1277, 'y': 1107.73378, 'width': 703.60449, 'height': 587.38416}, {'x': 694.18126, 'y': 796.7657, 'width': 524.56226, 'height': 1039.70135}]	opacity 1 1806.1277 1107.73378 2509.73219 1695.11794 opacity 1 694.18126 796.7657 1218.74352 1836.4670500000002

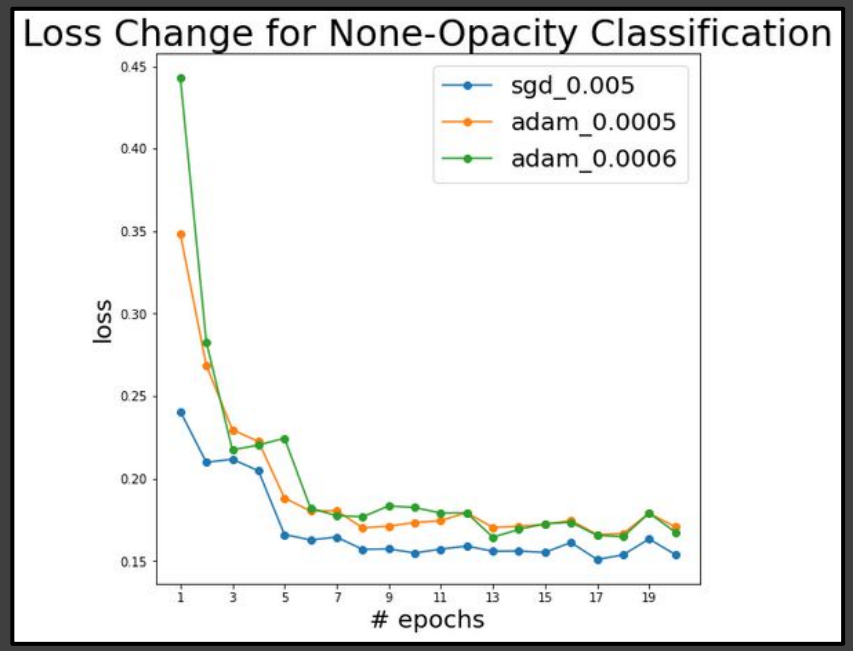
	id	boxes	label	StudyInstanceUID	class	x_min	y_min	x_max	y_max	bbox_area
3169	79ab8bbf9c7c_image	[{'x': 1806.1277, 'y': 1107.73378, 'width': 703.60449, 'height': 587.38416}, {'x': 694.18126, 'y': 796.7657, 'width': 524.56226, 'height': 1039.70135}]	opacity 1 1806.1277 1107.73378 2509.73219 1695.11794	f154086bd7b9	opacity	1806.12770	1107.73378	2509.73219	1695.11794	413286.132331
8062	79ab8bbf9c7c_image	[{'x': 1806.1277, 'y': 1107.73378, 'width': 703.60449, 'height': 587.38416}, {'x': 694.18126, 'y': 796.7657, 'width': 524.56226, 'height': 1039.70135}]	opacity 1 694.18126 796.7657 1218.74352 1836.4670500000002	f154086bd7b9	opacity	694.18126	796.76570	1218.74352	1836.46705	545388.089881

mission 1의 경우

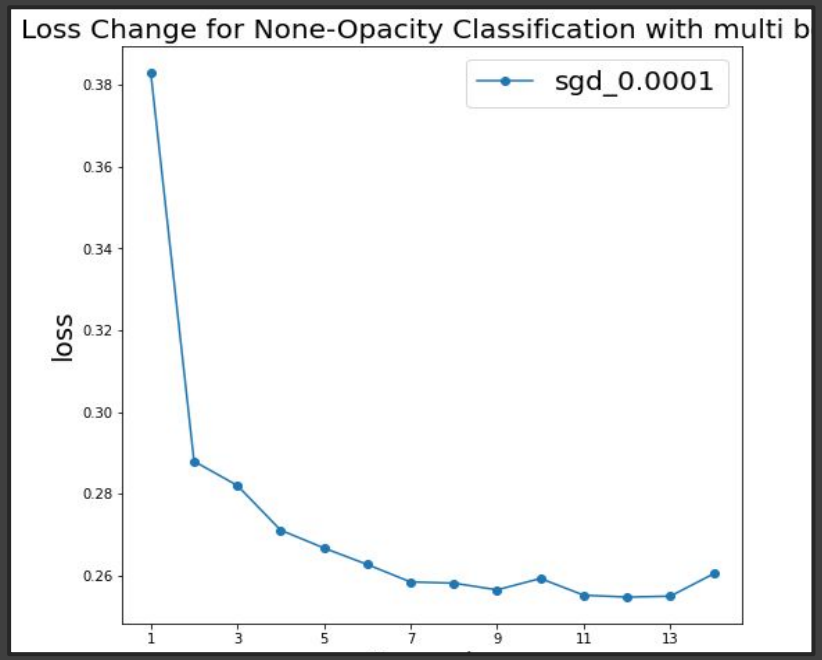
한 이미지에 여러 바운딩 박스가 있을 때,

모든 바운딩 박스를 이용 (Multi Box)

결과 : training dataset loss graph

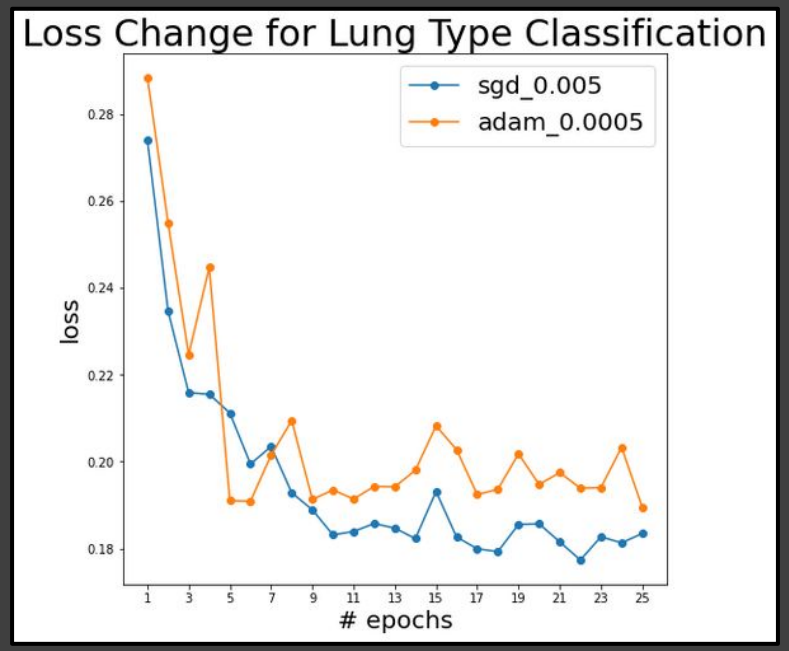


- 폐 x-ray의 이상 없음(none) / 이상 있음(opacity)을 구분하는 모델에 대한 loss 그래프 → One box
- Optimizer
  1. SGD(stochastic gradient descent)
     - learning rate = 0.005
  2. Adam
     - learning rate = 0.0005, 0.0006



- 폐 x-ray의 이상 없음(none) / 이상 있음(opacity)을 구분하는 모델에 대한 loss 그래프 → Multi box
- Optimizer
  1. SGD(stochastic gradient descent)
     - learning rate = 0.001

결과 : training dataset loss graph



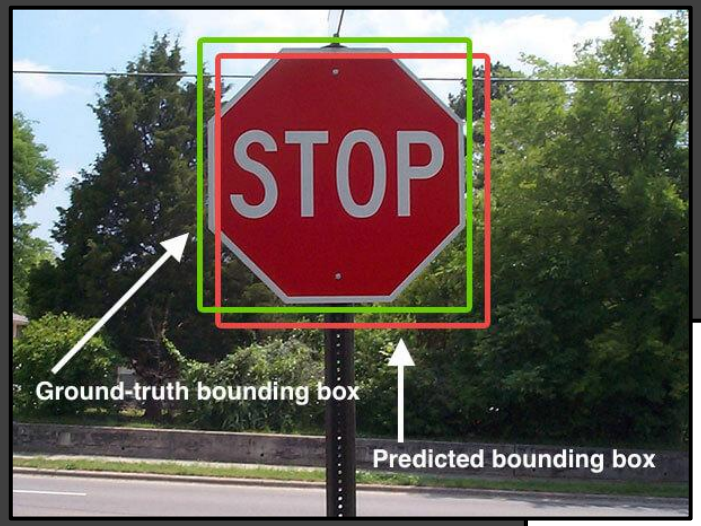
- 폐 x-ray의 4 classes 를 구분하는 모델에 대한 loss 그래프
- Optimizer
  1. SGD(stochastic gradient descent)
     - learning rate = 0.005
  2. Adam
     - learning rate = 0.0005

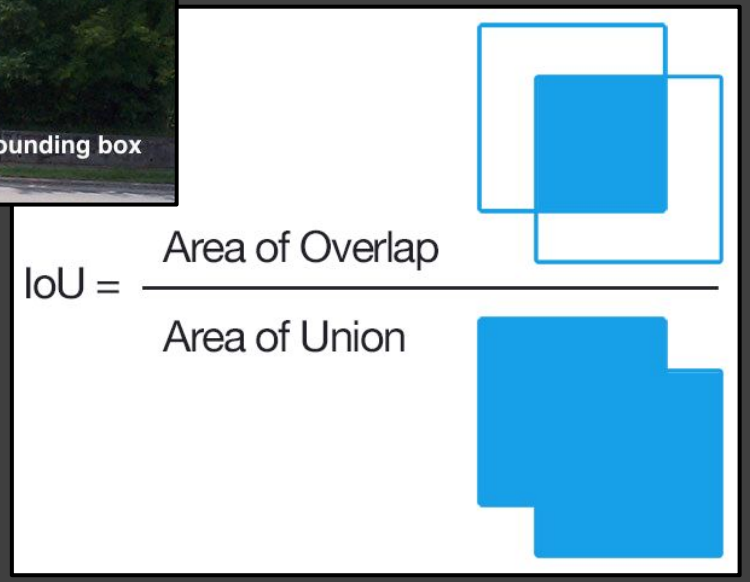
세 그래프 모두 **loss** 값 감소 확인

모델 성능 측정 : IoU

IoU

예측한 **bbox (bounding box)** 와 실제 **bbox**의 교집합을 합집합으로 나누어 계산된 스코어






```
## iou 계산
def iou(bb1, bb2):
    ## 오른쪽 좌표가 왼쪽 좌표보다 커야 하고, 위 좌표가 아래 좌표보다 커야 함 그렇지 않을 경우 ass
erterror
    assert bb1['x1'] < bb1['x2']
    assert bb1['y1'] < bb1['y2']
    assert bb2['x1'] < bb2['x2']
    assert bb2['y1'] < bb2['y2']

    ## 두개의 bounding box가 겹치는 영역의 좌표
    x_left = max(bb1['x1'], bb2['x1'])
    x_right = min(bb1['x2'], bb2['x2'])
    y_bottom = max(bb1['y1'], bb2['y1'])
    y_top = min(bb1['y2'], bb2['y2'])

    if x_right < x_left or y_top < y_bottom: return 0

    intersection_area = (x_right - x_left) * (y_top - y_bottom)

    bb1_area = (bb1['x2'] - bb1['x1']) * (bb1['y2'] - bb1['y1'])
    bb2_area = (bb2['x2'] - bb2['x1']) * (bb2['y2'] - bb2['y1'])

    iou = intersection_area / (bb1_area + bb2_area - intersection_area)

    assert iou <= 1
    assert iou >= 0
    return iou
```

모델 성능 측정 : Recall / precision

Recall / precision

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

❓ 그렇다면 object detection에서 TP, FP, FN은 어떻게 정의할 것인가 ? ❓

모델 성능 측정 : Recall / precision

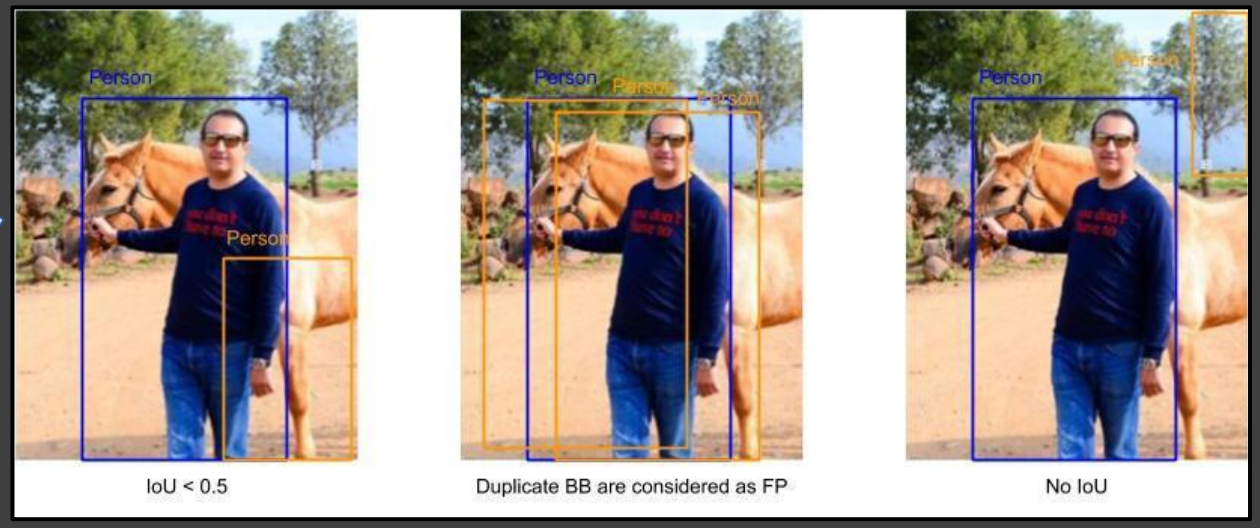
1. **TP (True Positive)**

   iou 가 0.5 이상인 경우, 정확한 class

2. **FP (False Positive)**

   iou가 threshold 값을 못 넘거나, iou는 threshold를 넘지만 높은 정확도를 가진 다른 박스가 있을 경우

FP



3. **FN (False Negative)**

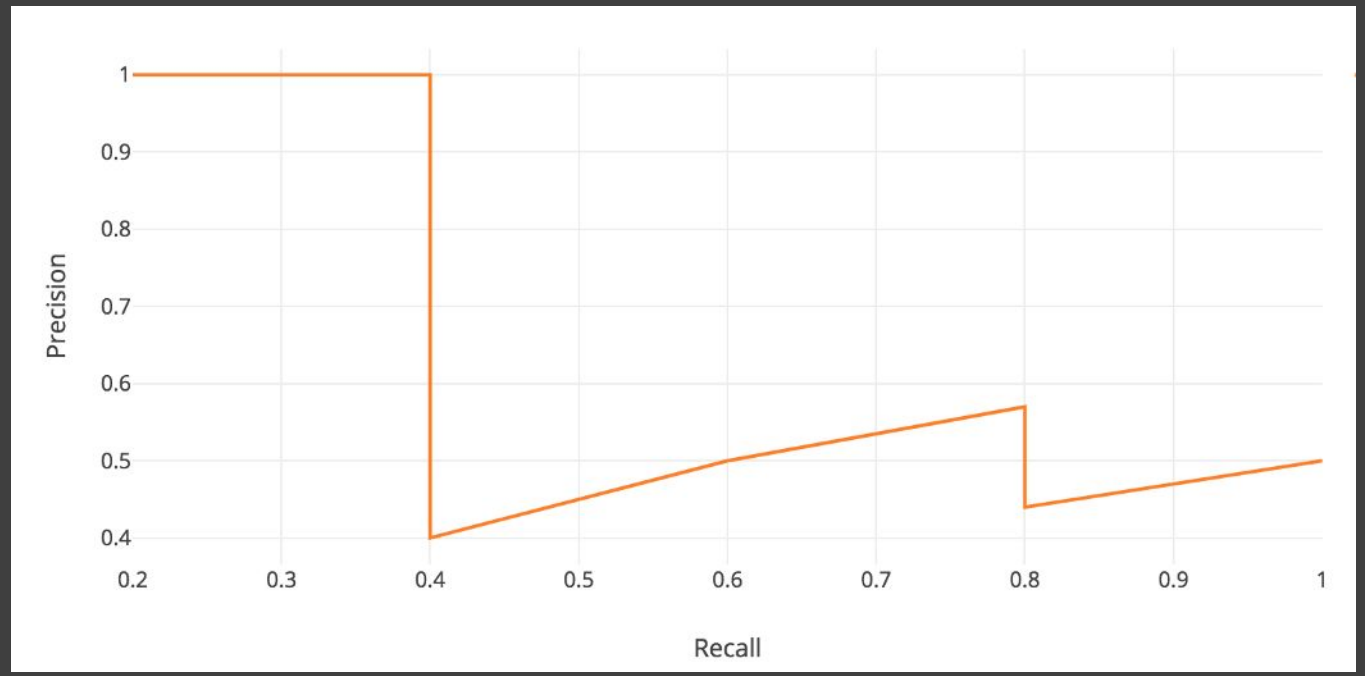
   정답이 있는 사진을 배경으로 예측하는 경우, 혹은 클래스를 정확하게 맞추지 못한 경우

FN

모델 성능 측정 : AP 계산

AP (Average Precision) 계산 과정

1. 앞에서 정의한 **TP,FP,FN**을 통해 **precision-recall curve**를 그림
2. **curve** 아래 면적을 적분



$$\mathrm{AP} = \int_0^1 p(r)dr$$

모델 성능 측정

TP/FP/NP 판단

```
for detection_idx, detection in enumerate(detections):
    ground_truth_img = [bbox for bbox in ground_truths if bbox[0] == detection[0]]
    num_gts = len(ground_truth_img)
    best_iou = 0

    for idx, gt in enumerate(ground_truth_img):
        iou = iou_calc(torch.tensor(detection[3:]),
                                    torch.tensor(gt[3:]),)

        if iou > best_iou:
            best_iou = iou
            best_gt_idx = idx

    if best_iou > iou_threshold:
        if amount_bboxes[detection[0]][best_gt_idx] == 0:
            TP[detection_idx] = 1
            amount_bboxes[detection[0]][best_gt_idx] = 1
        else:
            FP[detection_idx] = 1

    else:
        FP[detection_idx] = 1
```

AP 계산

```
# cumsum은 누적합을 의미합니다.
# [1, 1, 0, 1, 0] -> [1, 2, 2, 3, 3]
TP_cumsum = torch.cumsum(TP, dim=0)
FP_cumsum = torch.cumsum(FP, dim=0)
recalls = TP_cumsum / (total_true_bboxes + epsilon)
precisions = torch.divide(TP_cumsum, (TP_cumsum + FP_cumsum + epsilon))
precisions = torch.cat((torch.tensor([1]), precisions))
recalls = torch.cat((torch.tensor([0]),recalls))

# torch.trapz(y,x) : x-y 그래프를 적분합니다.
average_precisions.append(torch.trapz(precisions, recalls))
```

모델 성능 측정 : mAP (mean AP) 결과

mAP : none / opacity (one box)

1. threshold = 0.5
   - mAP : 0.0741

mAP : none / opacity (multi box)

1. threshold = 0.5
   - mAP : 0.0433

2. threshold = 0.3
   - mAP : 0.0718

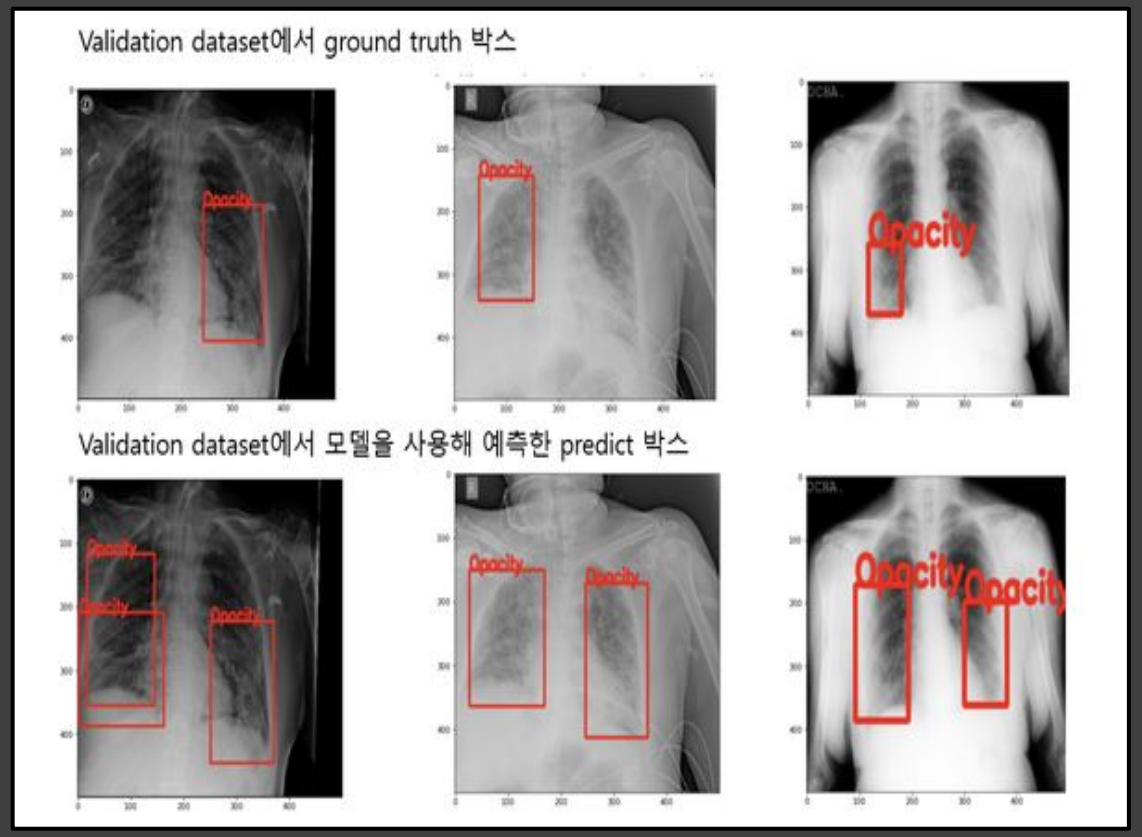
mAP : 4 classes

1. threshold = 0.5
   - mAP : 0.0589

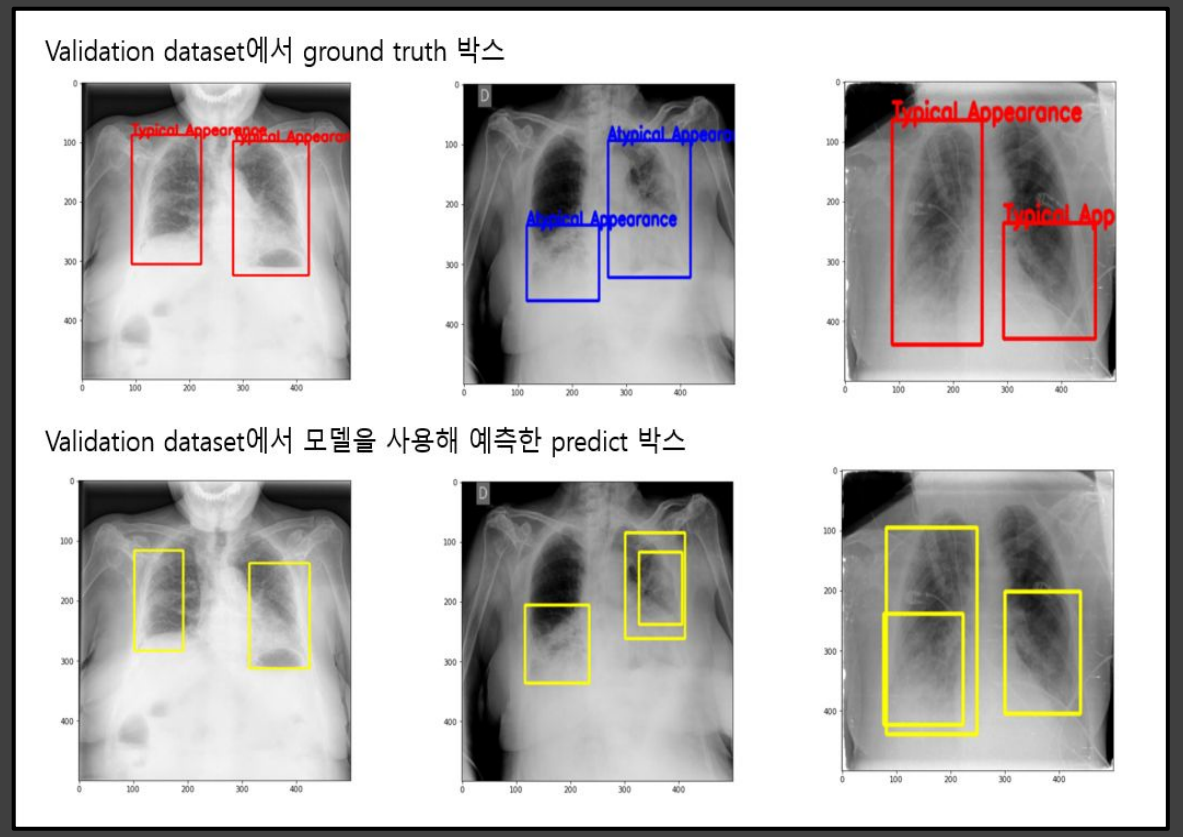
2. threshold = 0.3
   - mAP : 0.07

시각화 (Visualization)

none / opacity (one box)



none / opacity (multi box)



어려웠던 점 & 소감

- backprop 부분의 "RuntimeError: Found dtype Double but expected Float"
  - Pytorch가 같은 자료형 타입끼리만 연산을 해서 문제 발생
  - .backward()부분부터 디버깅을 해보고 각 결과값을 print()문으로 확인해 본 결과 bounding box를 만드는 영역에서만 float64 자료형이 나오는 것을 확인
  - 많은 시도 끝에 데이터로더를 바꾸어보는 시도를 하였고, bounding box를 출력하는 부분의 dtype을 모두 float32로 변환해 문제 해결

상준

이 문제를 해결하려고 며칠을 고생했는데, 디버깅을 할 때 세세한 부분을 고치려 하기보다는 전체적인 흐름 속에서 틀린 부분을 찾아내는 것이 더 중요하다는 사실을 깨달았다.

전체적인 소감

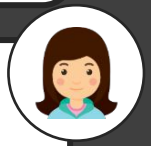
혜림

그동안 배웠던 것을 직접 구현할 수 있었던 점에서 흥미로웠던 프로젝트였다. 코드를 구현하면서 파이토치에서는 변수의 타입과 cpu를 통한 학습으로 어디서 에러가 나는지 확인하는 것이 중요함을 느낄 수 있었다. 비록 시간을 투자한 것에 비해서 기대하는 것만큼의 성능은 안 나왔으나, 팀과 많은 시도를 하면서 열심히 프로젝트를 마무리할 수 있어 기억에 남는 프로젝트가 될 것 같다.

연정

논문으로 공부했던 Faster R-CNN의 코드 구조를 알 수 있었던 유익한 시간이었다. 우여곡절이 많았지만 다함께 노력해서 데이터로더부터 시각화까지 도전해 볼 수 있었던 좋은 기회였던 것 같다.

은아



새로운 것을 시도할 때마다 에러가 나서 한단계, 한단계 나아가는 게 너무 힘들었던 것 같다. 그래도 어떤 종류의 에러가 어디서 발생했는지 확인하고 해결해나가는 과정을 배울 수 있어서 보람 있었다. 그리고 팀원들과 끊임없이 소통하면서 문제를 해결하는 과정을 처음 경험할 수 있어서 의미 있었다.

참고

- https://nuggy875.tistory.com/20
- https://www.kaggle.com/c/siim-covid19-detection/data
- https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
- https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

감사합니다