G-팔로미1 프로젝트

최종 발표

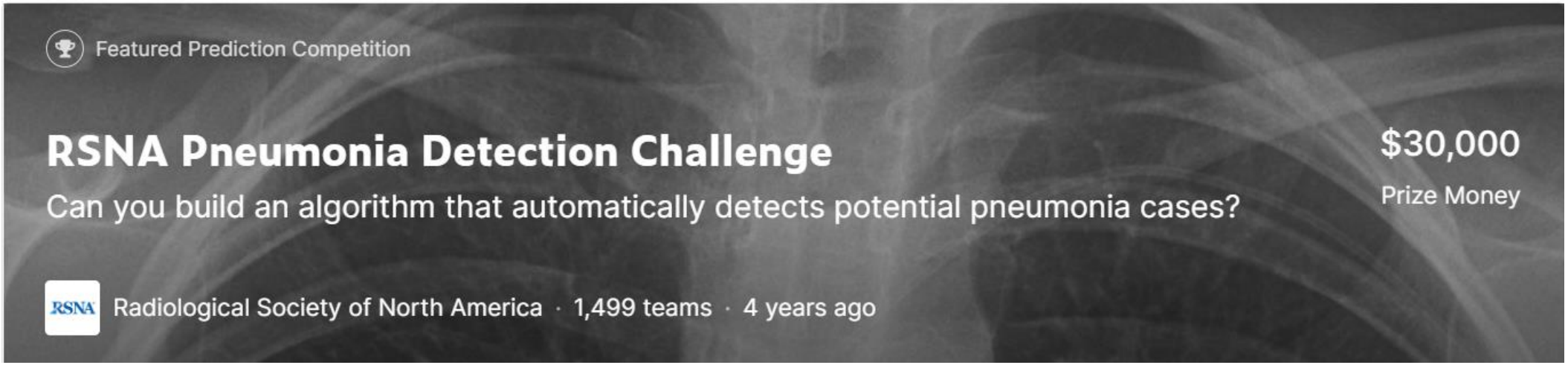
의용생체공학과
(바이오-인공지능 융합전공)
201838205
김나연

가천대학교
Gachon University

CONTENTS

I. 프로젝트 소개 및 필요성

I-1. 프로젝트 소개

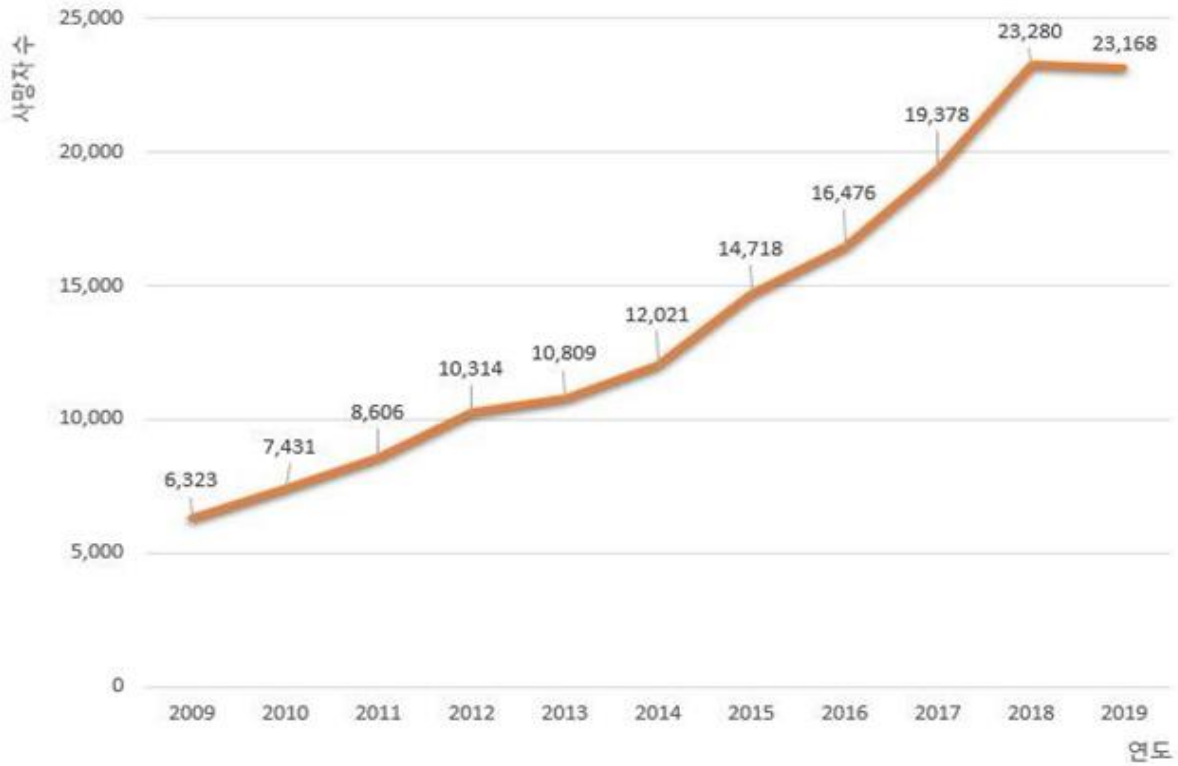


- Kaggle의 RSNA Pneumonia Detection Challenge
- Chest X-ray 영상을 입력받아 Pneumonia 여부에 대한 클래스를 분류하고 Pneumonia를 Localization하는 챌린지

I-2. 프로젝트의 필요성

- 기존의 항생제로 치료하기 어려운 내성균 등장
- 2015년에는 세계적으로 92만 여명의 5세 미만 어린이가 폐렴으로 사망 (5세 미만 사망률의 15% 이상 차지)
- 우리나라에서도 매년 꾸준히 폐렴 사망자 증가 (2019년 기준 23,168명)
- 조기 발견 시 사망률 현저히 감소

< 대한민국의 연도별 폐렴 사망자 수 >



출처: 통계청

I-2. 프로젝트의 필요성

		영상의학과 의사 단독 판독	인공지능 시스템 보조 판독
초응급 질환 (기흉, 기복증, 대동맥 박리)	진단 정확도	29.2%(7/24)	70.8%(17/24)
	판독 대기 소요 시간	3371 초	640 초
응급 질환 (폐렴, 폐부종, 활동성 결핵, 간질성 폐질환, 폐결절, 흉수, 종격동 종양, 늑골 골절)	진단 정확도	78.2%(244/312)	82.7%(258/312)
	판독 대기 소요 시간	2127 초	1840 초
비응급 질환/ 정상	진단 정확도	91.4%(801/876)	93.8%(822/876)
	판독 대기 소요 시간	2815 초	3267 초

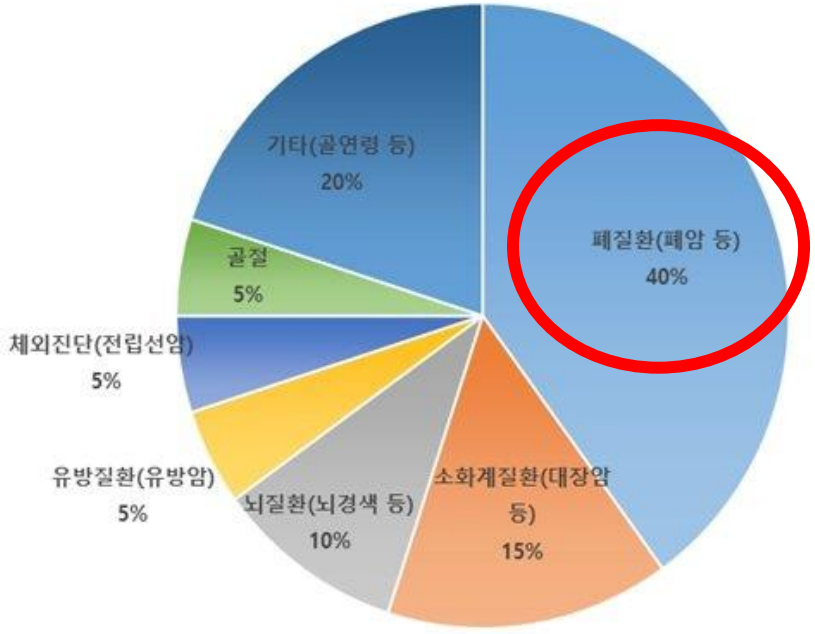
출처: 서울대병원

- 서울대-루닛 공동개발한 폐질환 진단 보조 SW ‘루닛 인사이트CXR’로 진행한 응급실 모의 판독 실험 결과
- 영상의 단독 판독보다 인공지능 시스템과 같이 판독했을 때의 정확도가 더 높고, 소요시간이 짧다.
- 특히 인력 대비 검사량이 많은 응급 질환일 때 더욱 그 격차가 크다.

II. 현재 개발 수준

II-1. 현재 개발 수준 - 국내 의료기기 허가

□ 인공지능 의료기기 허가 질환별 제품현황



업체명	제품명(모델명)	사용목적	비고
(주)뷰노	VUNO Med-Chest X-ray	흉부 이상부위 검출 보조	폐질환
(주)JLK 인스펙션	JLD-01A	폐 영상 분석	폐질환
	JLD-01B	폐 영상 분석	폐질환
	JLD-02A	폐 영상 분석	폐질환
(주)루닛	Lunit Insight CXR Nodule	폐결절 검출 보조	폐질환
	Lunit Insight CXR MCA	흉부 이상부위 검출 보조	폐질환
(주)코어라인소프트	AVIEW	폐 영상 분석	폐질환

출처: 식약처

- 2020년 기준 국내 인공지능 의료기기 허가 질환별 제품현황 중 40%가 폐질환 영상 분석 또는 검출 보조
- 타 질환에 비해 폐질환을 위한 인공지능 개발 활발

II-2. 현재 개발 수준 예시 - JLK

JLD-02K (인공지능 기반 흉부 X-Ray 폐 영상 분석 솔루션)

03 인공지능 분석 결과 UI

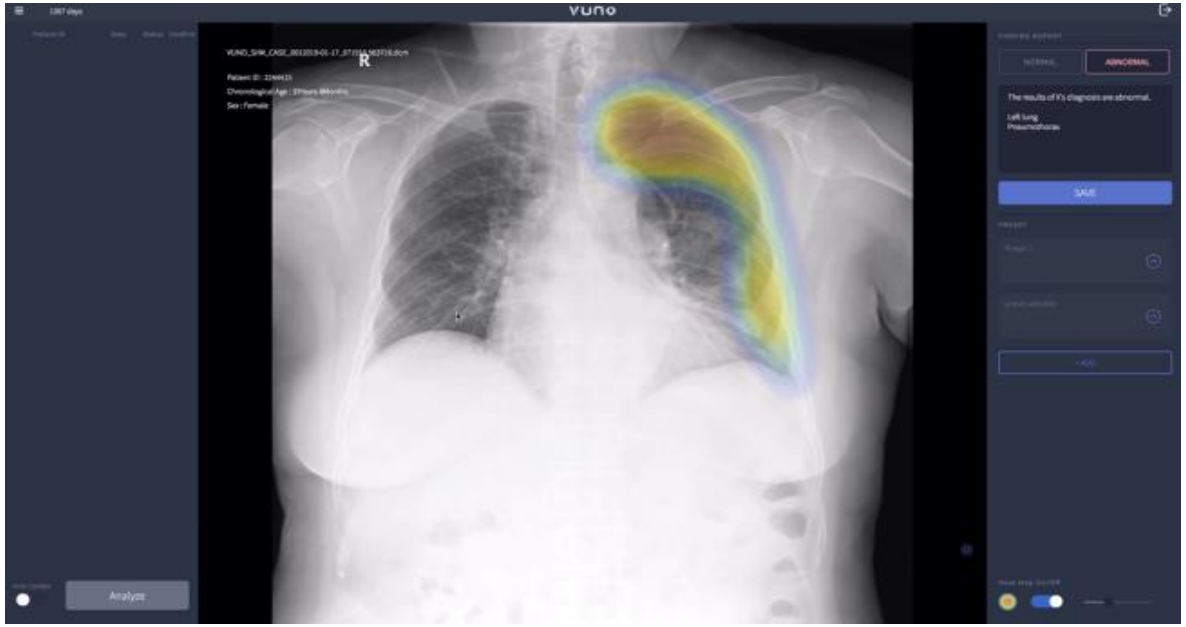
04 분석 결과 보고서 UI

출처: JLK

- 폐렴 등 폐 병변 16가지에 대한 전문의 판독 보조
- AUC : 99% (단일 기관 데이터), 분석시간: 20s 이내
- 영상 분석 결과에 대한 Heat-map과 정량 점수 제공

II-2. 현재 개발 수준 예시 - VUNO

VUNO Med Chest X-Ray



출처: VUNO

- 영상의 단독 판독 시보다 병변탐지 및 영상분류의 정확도가 높고, 판독시간도 50% 감소
- 주요 5대 소견(간질성 음영, 경화, 결절, 기흉, 흉수)에 대해 정상/비정상 여부를 3초내 분석해 폐렴, 폐암 등 폐질환 진단 보조
- 19년 8월 식약처 허가, 20년 6월 유럽 CE인증 → 국내 및 유럽국가에 판매

II-2. 현재 개발 수준 예시 - VUNO

VUNO Med Chest X-Ray

출처: VUNO

- 삼성전자 이동형 디지털 X-ray 촬영 장비 ‘GM85’와 천장 고정형 ‘GC85A’에 기본 탑재
- 촬영과 동시에 인공지능으로 결과를 분석해 신속 확인 가능

III. 프로젝트 진행

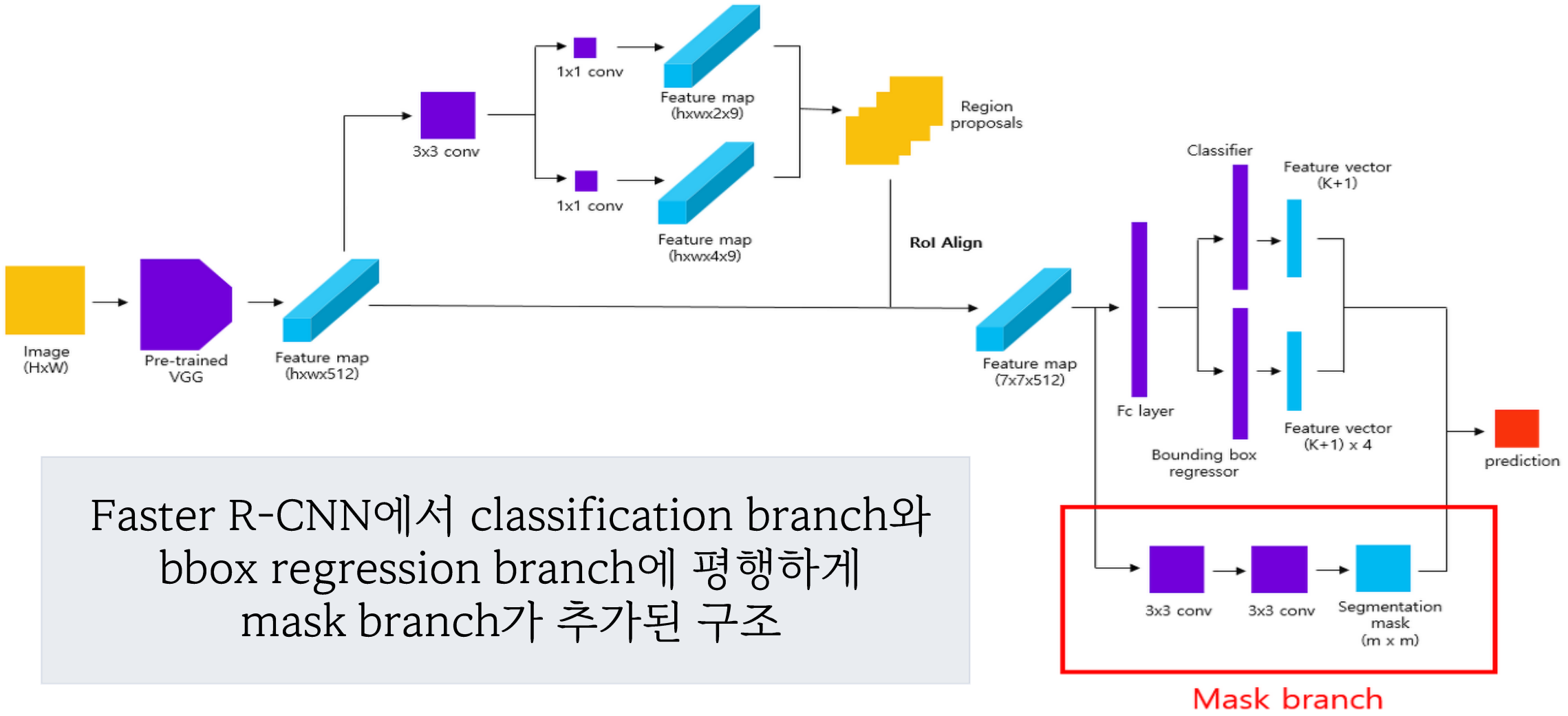
III-1. 사용한 알고리즘
- Mask R-CNN -

III-1. 사용한 알고리즘

Mask R-CNN

- Regional Proposal과 Classification이 순차적으로 이뤄지는 2-stage Detector 중 대표 알고리즘인 Mask R-CNN 사용
- 정확도가 높고 R-CNN 계열 중 빠른 속도를 보인다.
- bounding box 뿐만 아니라 mask까지 표현 가능
- Detection뿐만 아니라 Instance segmentation에서도 많이 사용
- 한 이미지 내에서 감지해야 할 폐렴 부분이 두 개 이상인 이미지도 있으므로 이미지 내에서 같은 클래스더라도 각각의 객체로 탐지할 수 있는 Mask R-CNN 사용

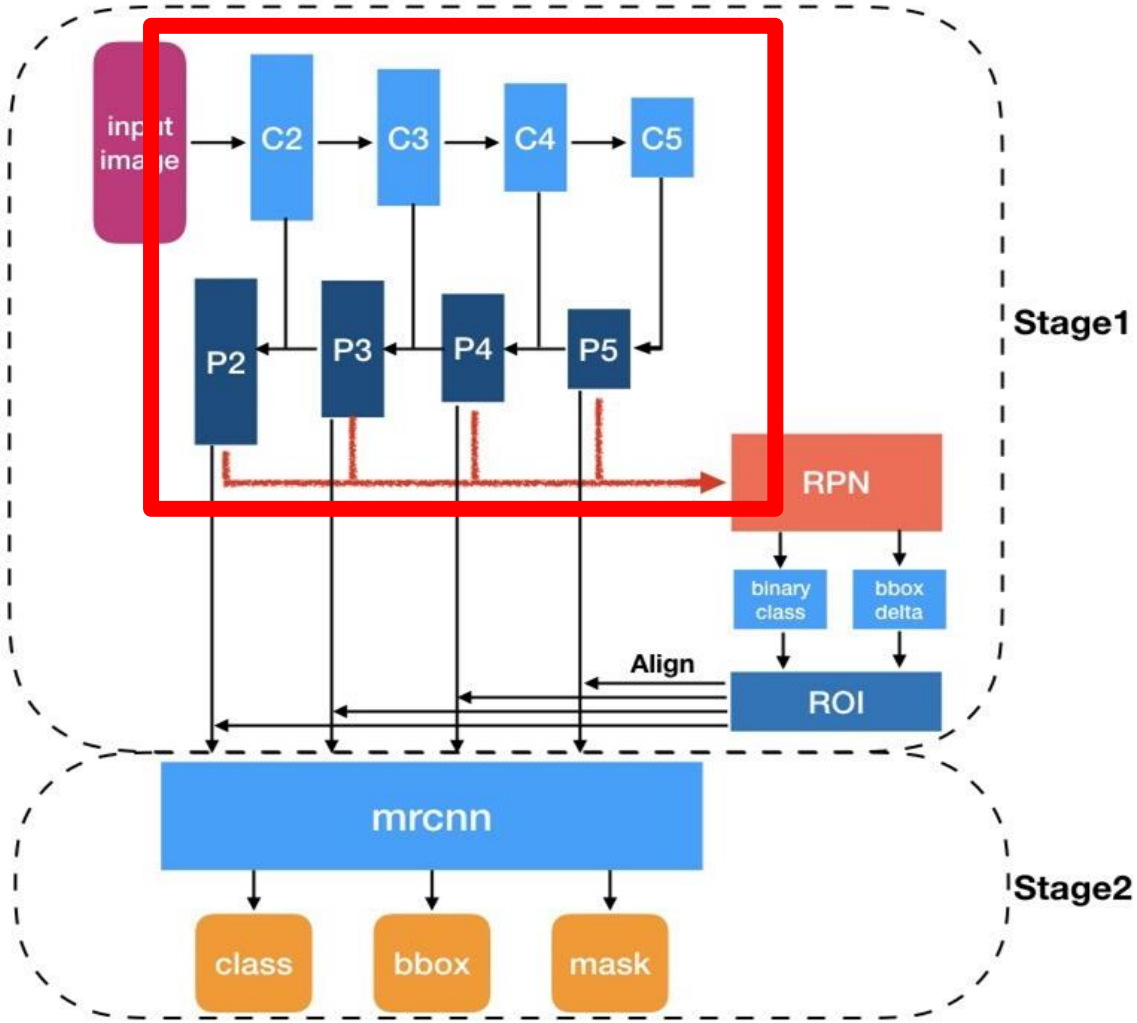
III-1. Mask R-CNN



Faster R-CNN에서 classification branch와 bbox regression branch에 평행하게 mask branch가 추가된 구조

III-1. Mask R-CNN

FPN (Feature Pyramid Network)



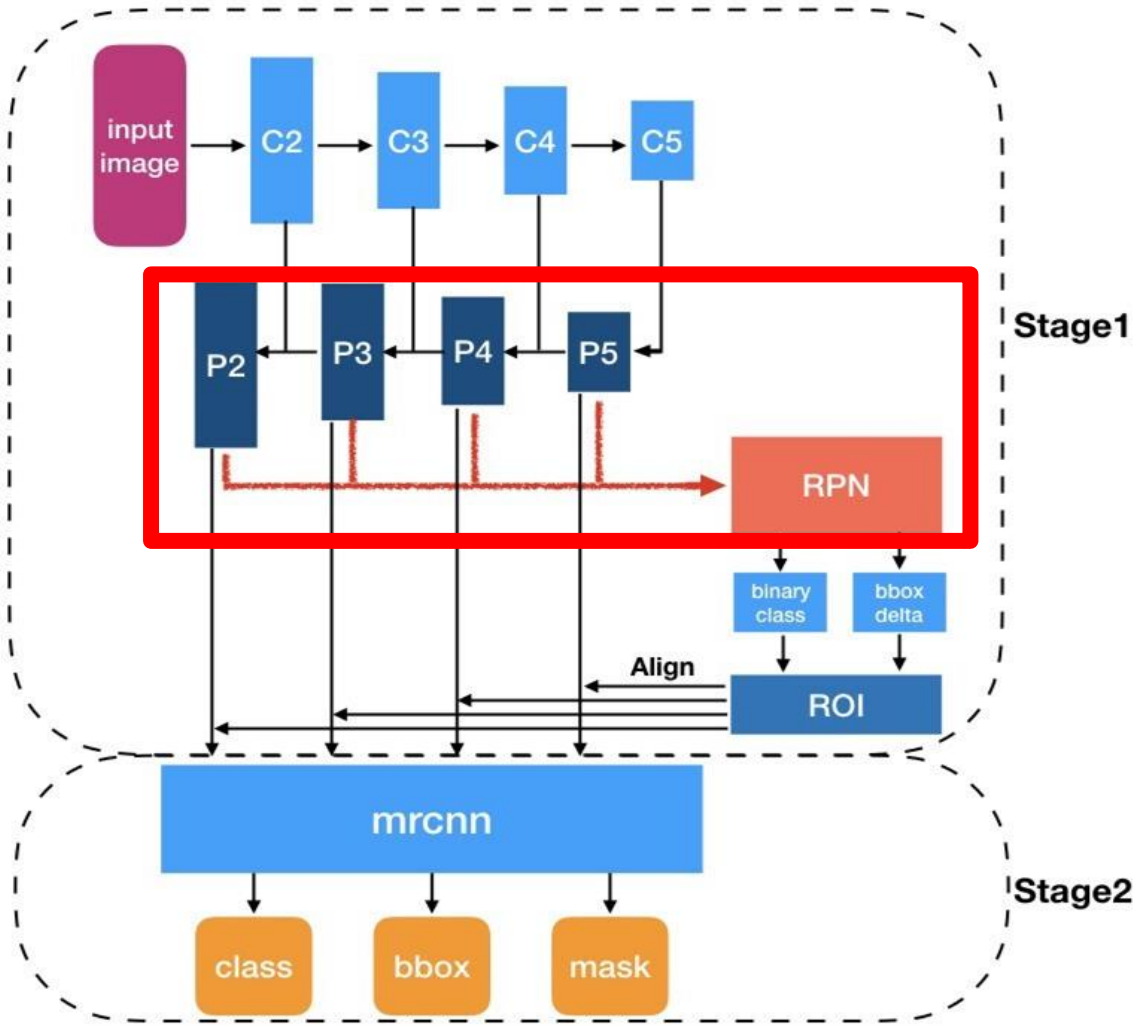
1. 신경망을 통과하면서 단계별로 feature map을 생성

2. 상위 레이어에서부터 거꾸로 내려오면서 feature map을 합쳐준다

3. 상위 레이어의 추상화된 정보와 하위 레이어의 작은 물체에 대한 정보를 동시에 가져갈 수 있음.

III-1. Mask R-CNN

RPN (Region Proposal Network)

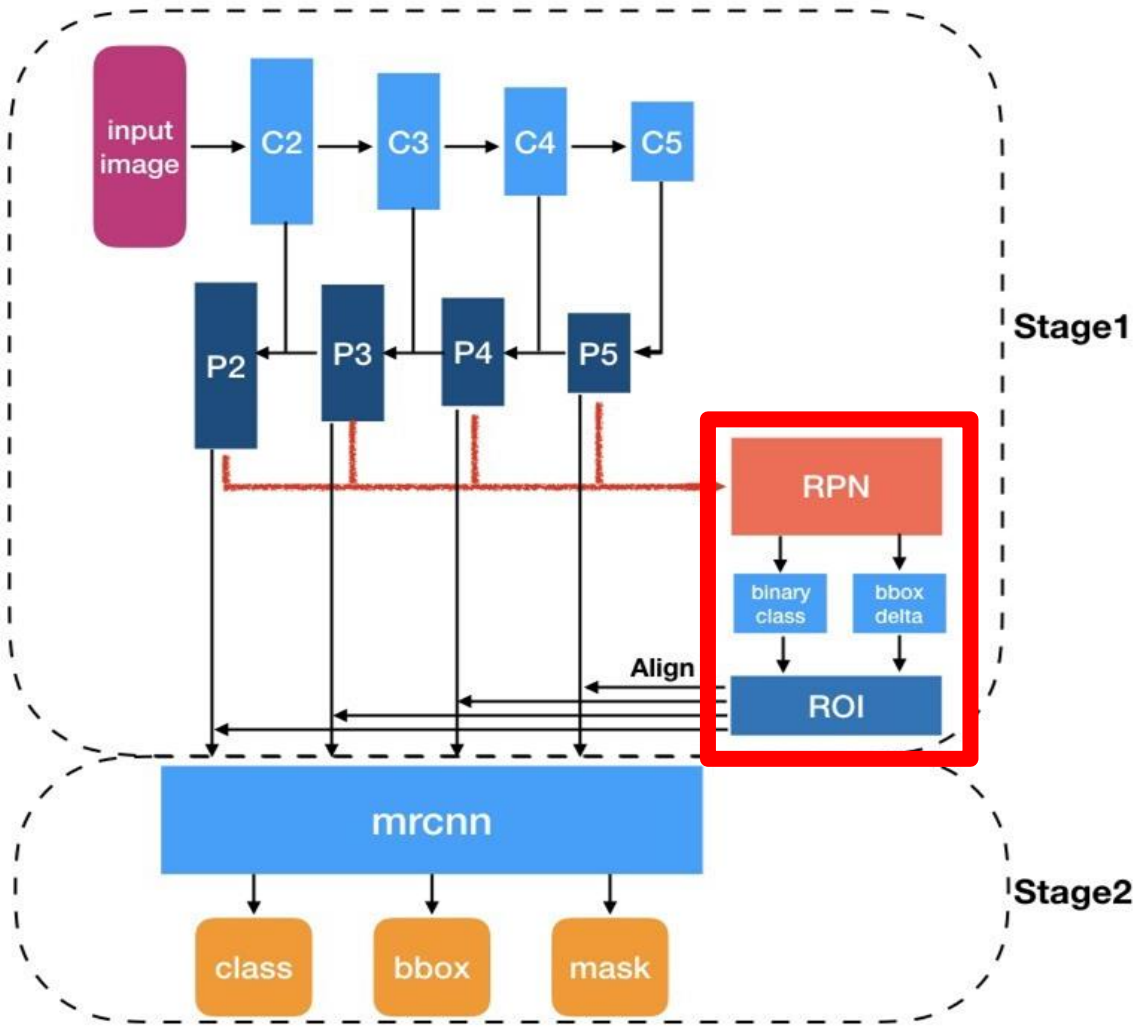


1. 원본 이미지에서 anchor box를 통해 여러 개의 region proposals 생성

2. RPN으로 confidence score와 bbox regressor를 가진 region proposal을 출력

III-1. Mask R-CNN

Proposal Layer RPN을 통해 얻은 Region Proposal 중 최적의 ROI를 선정



1. confidence score가 높은 순으로 K개의 anchor선정

2. bbox regressor에 따라 anchor box 크기 조정

3. 이미지 밖으로 나가는 anchor box 제거

4. NMS 수행

5. FPN 결과와 anchor box에 대한 정보 결합 후 objectness score가 높은 순으로 N개의 anchor box선정

III-1. Mask R-CNN

NMS (Non Maximum Suppression)

anchor box 중 정확한 bbox를 선택하도록 하는 기법

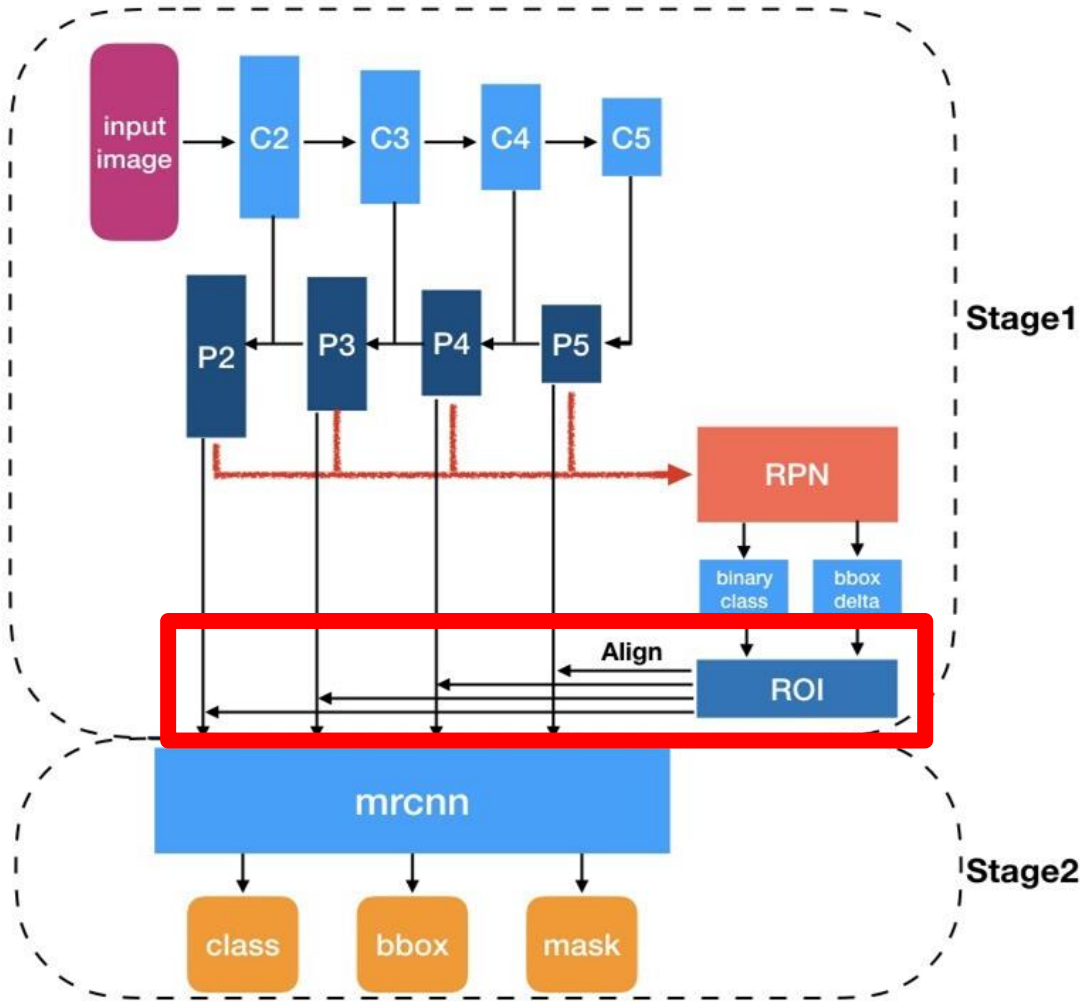
- 1차 필터링
  - confidence score threshold 이하의 score를 가지는 anchor box는 제거
  - Mask R-CNN의 config에서 DETECTION_MIN_CONFIDENCE로 조절

- 2차 필터링
  - 남은 anchor box들을 confidence score를 내림차순으로 정렬 후 첫번째의 anchor box를 기준으로 설정
  - 기준 box와 IoU가 threshold 이상인 anchor box는 같은 물체를 감지하는 것(detection)으로 간주하고 제거
  - Mask R-CNN의 config에서 DETECTION_NMS_THRESHOLD로 조절

- 즉, 1차 threshold가 높을수록, 2차 threshold가 낮을수록 더 많은 box 제거

III-1. Mask R-CNN

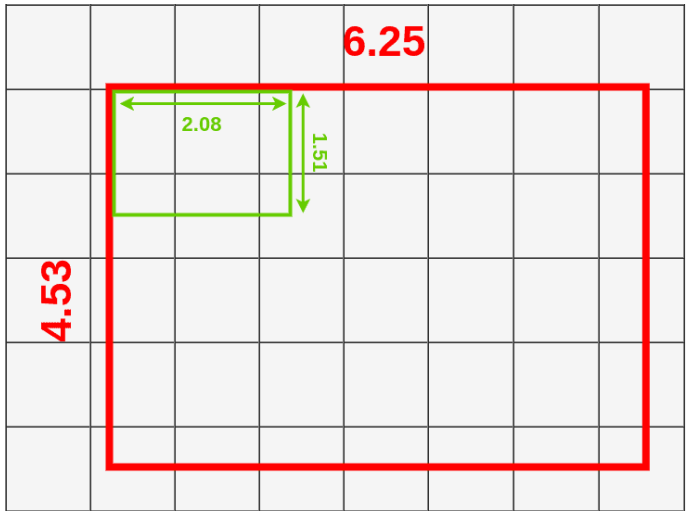
ROI Align Layer



- ROI를 어떤 scale의 feature map과 연결시킬지 결정
- feature와 ROI 사이가 어긋나 mask예측 성능이 떨어지는 ROI pooling의 문제를 해결

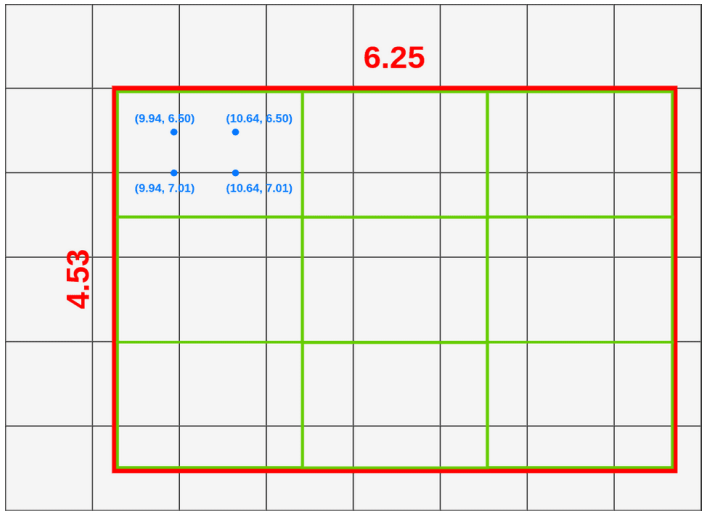
III-1. Mask R-CNN

ROI Align Layer



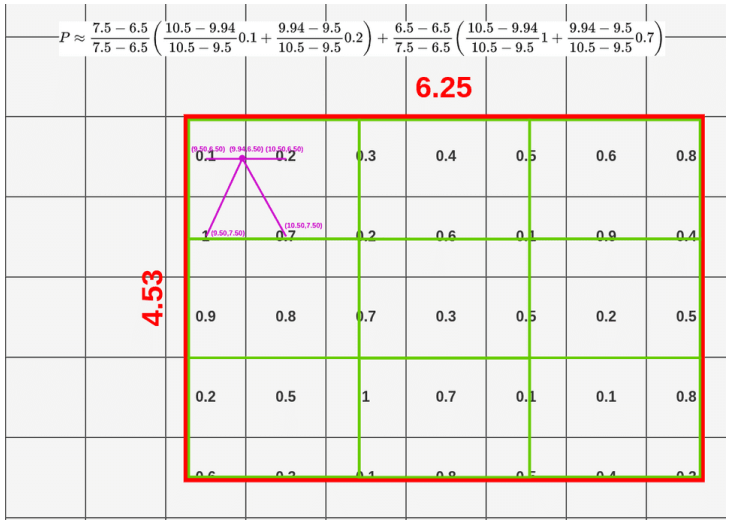
① 분할

ROI projection을 통해 얻은 ROI를 분할
위의 예시에서는 3x3 feature map으로 출력할 것이므로 3x3 형태로 분할
(; 초록색 선)



② Sampling point

분할된 하나의 칸(cell)에서 가로 및 세로를 삼등분 하는 지점을 찾는다.
(; 파란색 점)

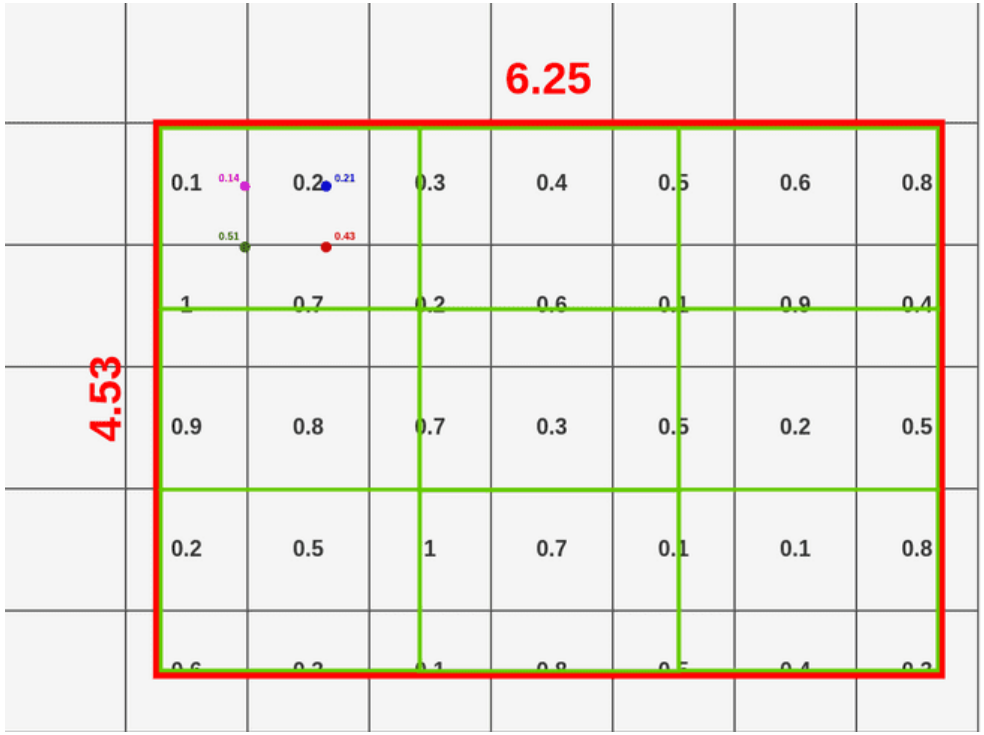


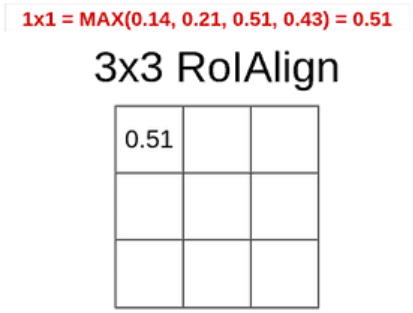
③ Bilinear interpolation

Sampling point와 인접한 4개의 feature map pixel의 값을 통해 sampling point의 값을 구한다.

III-1. Mask R-CNN

ROI Align Layer





④ Max pooling

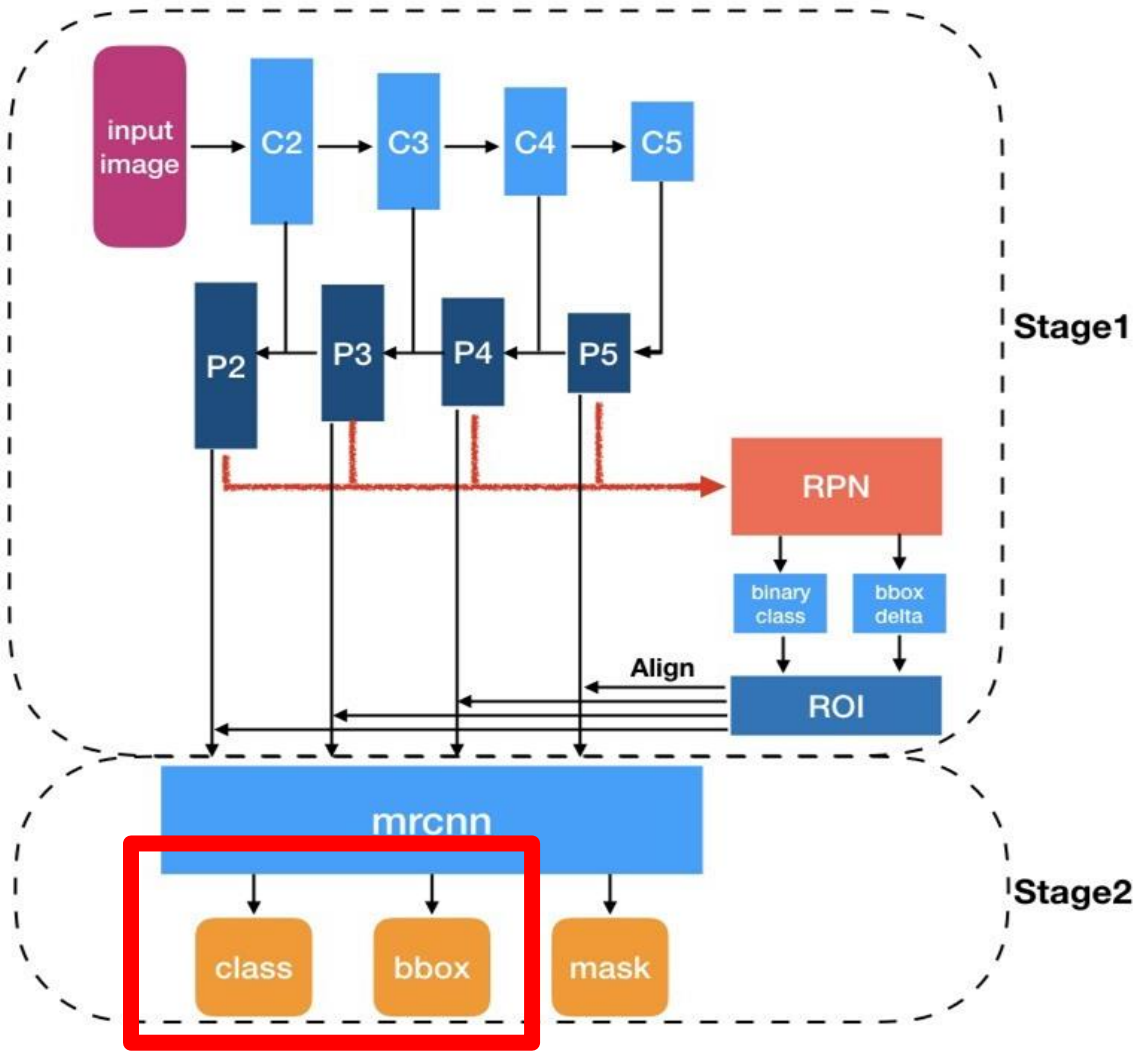
각 cell의 sampling 값을 연산 후, 각 cell마다 가장 큰 값을 선택한다.

⑤ 장점

ROI의 정확한 spatial location 보존 가능
Mask accuracy 향상

III-1. Mask R-CNN

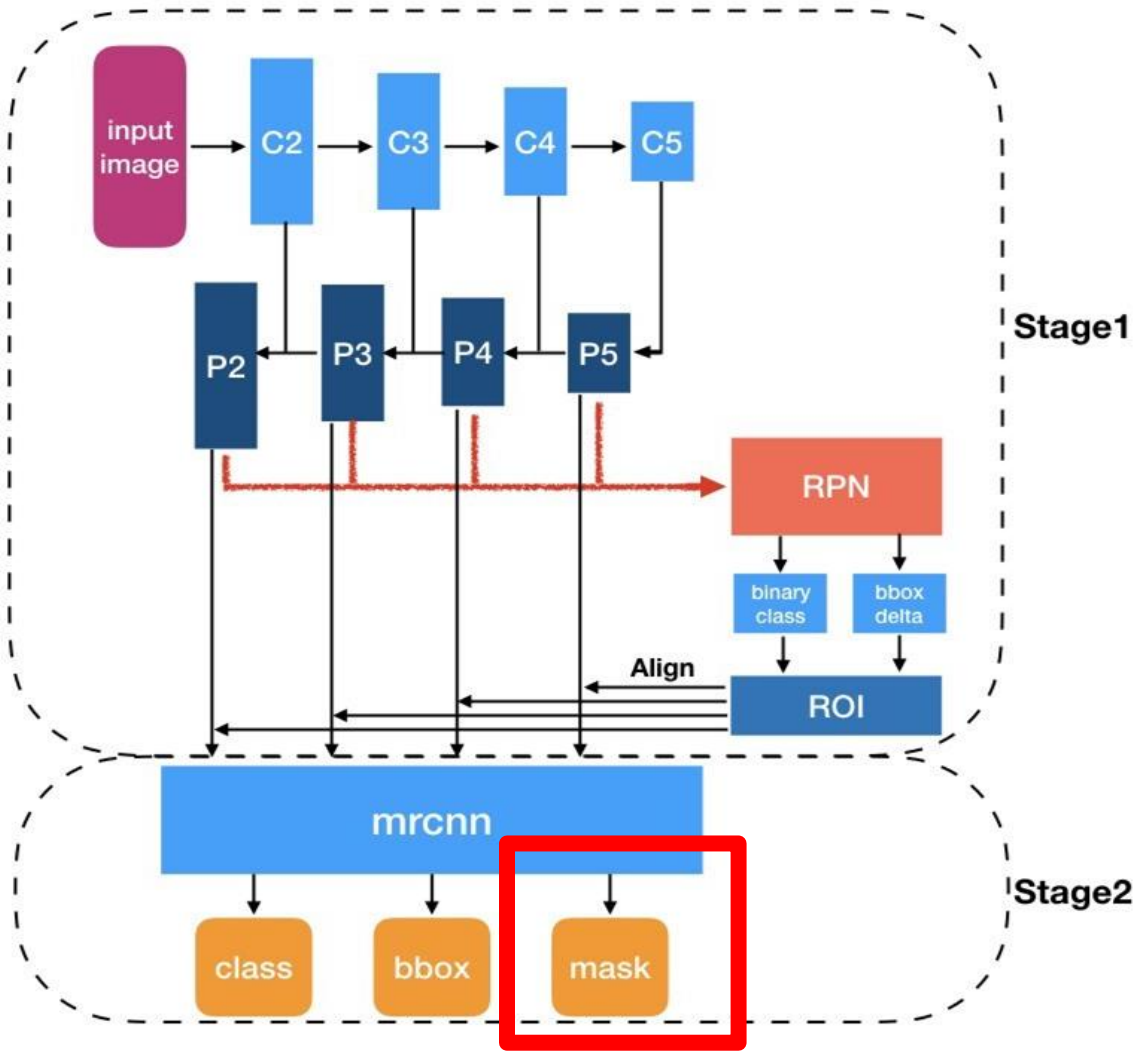
Class & BBox Branch



- 최종 class score와 bbox regressor 획득

III-1. Mask R-CNN

Mask Branch



1. class branch에서 얻은 가장 높은 score를 갖는 feature map을 선정

2. feature map의 각 pixel별로 sigmoid 함수를 적용해 0~1 사이의 값으로 조정

3. pixel 값이 mask threshold 값 이상이면 1을, 미만이면 0을 할당해서 mask 생성

III-2. Training 준비

1. 데이터 정보 및 분할

- **Class**
  - ○ Normal
  - ○ No Lung Opacity / Not Normal] Target 0
  - ○ Lung Opacity ← Pneumonia - Target 1

- **Target**
  - ○ 0 (→ Background) : 20672개
  - ○ 1 (→ class 1 (pneumonia)) : 6012개

] 비율을 맞춰 데이터 분할 진행

	Total	Train	Valid	Test
Target 0	20672 (약 3.438배)	15996 (약 3.412배)	2338 (약 3.532배)	2338 (약 3.532배)
Target 1	6012	4688	662	662
Total	26684	20684	3000	3000

2. Tensorflow 버전 설정 및 GPU 환경 조성

```
1 print(tf.__version__)
2 print(keras.__version__)
```

```
1.15.0
2.2.5
```

- Mask R-CNN 라이브러리 사용을 위해 keras와 tensorflow의 버전을 다운그레이드

2. Tensorflow 버전 설정 및 GPU 환경 조성

```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 7166802008727748107
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 23164095693
locality {
  bus_id: 1
  links {
  }
}
incarnation: 11894051080499729273
physical_device_desc: "device: 0, name: TITAN RTX, pci bus id: 0000:42:00.0, compute capability: 7.5"
]
```

- GPU는 NVIDIA TITAN RTX 사용 (학교 제공)
- GPU 사용 가능 환경인지 확인

```
tf.test.is_gpu_available()
```

```
True
```

- Tensorflow 버전(v1.15.0)에 호환되는 Cuda 10.0.13, cndnn 7.6.5
- 처음에 mask rcnn의 tensorflow 버전을 확인하지 않고 환경 조성을 해 GPU 사용 X
- training 이 되지 않아 기존에 설치된 것을 모두 삭제하고 호환 버전으로 재설치

```
!nvcc --version
```

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Sat_Aug_25_21:08:04_Central_Daylight_Time_2018
Cuda compilation tools, release 10.0, V10.0.130
```

- 재설치를 했으나 여전히 이전 버전으로 인식

```
!conda list cudnn
```

```
# packages in environment at C:\Users\Owner2\Anaconda3\envs\GPU_env:
#
# Name                    Version                   Build  Channel
cudnn                     7.6.5                cuda10.0_0
```

- 아나콘다 가상환경에서 conda install tensorflow-gpu==1.15.0을 통해 이와 맞는 라이브러리 자동 설치되는 방법으로 해결

3. Hyperparameter 조정

```
class DetectorConfig(Config):
    NAME = 'pneumonia'
    GPU_COUNT = 1
    IMAGES_PER_GPU = 8

    DEVICE = "/GPU:0"

    BACKBONE = 'resnet101'

    NUM_CLASSES = 2

    IMAGE_MIN_DIM = 256
    IMAGE_MAX_DIM = 256
    RPN_ANCHOR_SCALES = (32, 64, 128, 256)

    TRAIN_ROIS_PER_IMAGES = 32
    MAX_GT_INSTANCES = 4
    DETECTION_MAX_INSTANCES = 3

    DETECTION_MIN_CONFIDENCE = 0.7
    DETECTION_NMS_THRESHOLD = 0.1

    STEPS_PER_EPOCH = 200
    VALIDATION_STEPS = 50
```

- Backbone Model : ResNet 101
- Class 개수 : background(=pneumonia 아님) & pneumonia , 총 2개
- FPN 후 이미지 사이즈 (256X256)
- 한 이미지 당 최대 Ground Truth 개수

box	1	2	3	4
image	23286	3266	119	13

- 한 이미지 당 detection 가능한 최대 개수 4개가 최대였지만 확률이 희박하므로 3개로 제한
- Confidence score가 0.7 이하면 제거
- 남은 bbox 중 가장 score가 높은 것과 IoU가 0.1 이하면 제거

4. Augmentation

```python
from imgaug import augmenters as iaa
```

```python
aug = iaa.Sequential([
    iaa.OneOf([
        iaa.Affine(
            scale={"x":(0.98, 1.04), "y":(0.98, 1.04)},
            translate_percent={"x":(-0.03,0.03), "y":(-0.05, 0.05)},
            rotate=(-2,2),
            shear=(-1,1),
        ),
        iaa.PiecewiseAffine(scale=(0.002, 0.03)),
    ]),
    iaa.OneOf([
        iaa.Multiply((0.85, 1.15)),
        iaa.ContrastNormalization((0.85, 1.15)),
    ]),
    iaa.OneOf([
        iaa.GaussianBlur(sigma=(0.0, 0.12)),
        iaa.Sharpen(alpha=(0.0, 0.12)),
    ]),
])
```

- Augmentation으로 인한 원본 이미지 변화에 Ground truth도 함께 적절한 형태로 변경해주는 imgaug 라이브러리
- iaa.Oneof 메소드를 이용해 이 중 하나의 기법이 선택돼서 적용
- iaa. Affine : scale 변화, 평행이동, 회전
- 밝기 조절
- Blur 또는 Sharpness

4. Augmentation

Original Image

Augmentation Image

III-3. Training

1. Model

```
model = modellib.MaskRCNN(mode='training',
                          config=config,
                          model_dir=model_save_dir)
```

```
==================================
Total params: 63,733,406
Trainable params: 63,621,918
Non-trainable params: 111,488
----------------------------------
```

Mrcnn.model 라이브러리 중 MaskRCNN 클래스를 사용해 모델링

```
LEARNING_RATE = 0.006
```

```
%%time

model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE*2,
            epochs=2,
            layers='all',
            augmentation=aug)
```

```
Checkpoint Path: ./model_weights₩pneumonia20220611T0605₩mask_rcnn_pneumonia_{epoch:04d}.h5
```

1. Epoch 2까지는 빠른 weight 감소를 위해 기본 learning rate의 2배로 training 진행
2. 각 epoch마다 지정한 경로(model_save_dir)에 weight 저장

1. Model

```
%%time

model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE,
            epochs=13,
            layers='all',
            augmentation=aug)
```

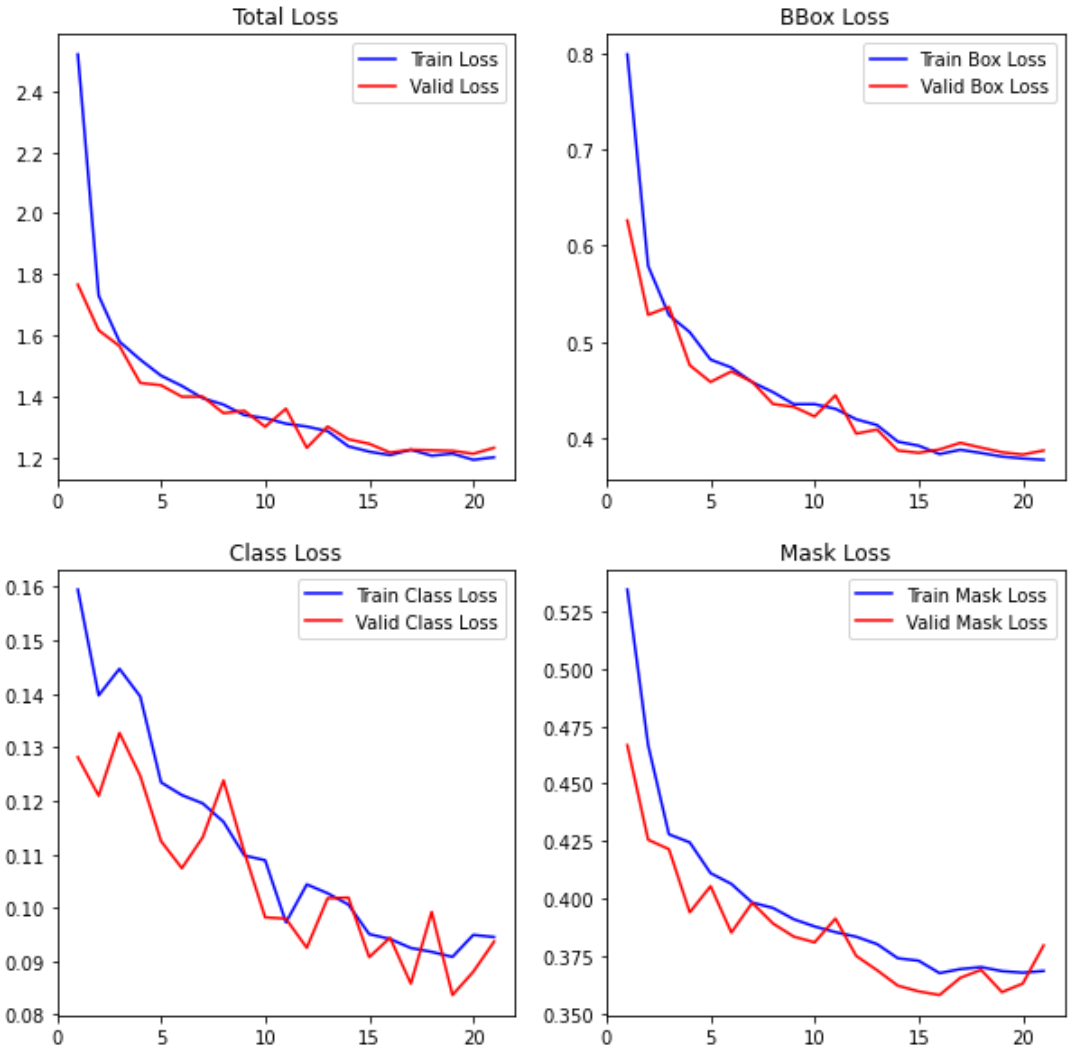
Epoch 3~13까지 Learning rate 0.006으로 Training

```
%%time

model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE/2,
            epochs=21,
            layers='all',
            augmentation=aug)
```

Epoch 14~21까지 Learning rate 0.003으로 Training

III-4. Loss & Inference

1. Loss Graph



Best epoch = 20

	Loss	Mrcnn_bbox
Train	1.190968	0.378605
Valid	1.210959	0.382718
	Mrcnn_class	Mrcnn_mask
Train	0.094921	0.368511
Valid	0.088031	0.363020

2. Inference

```
model_inference = modellib.MaskRCNN(mode='inference',
                                    config=inference_config,
                                    model_dir=model_save_dir)
```

```
wpath = os.path.join(model_save_dir, 'mask_rcnn_pneumonia_0020.h5')
```

```
model_inference.load_weights(wpath, by_name = True)
```

- Inference 모델 구축
- Best epoch인 epoch 20의 weight를 로드

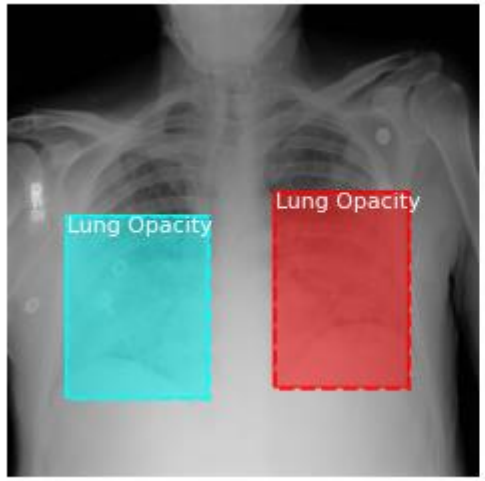
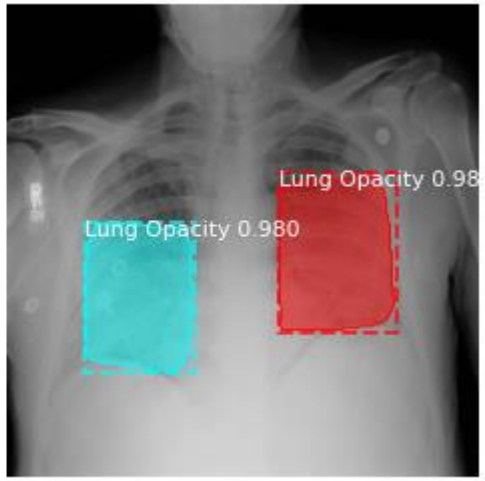
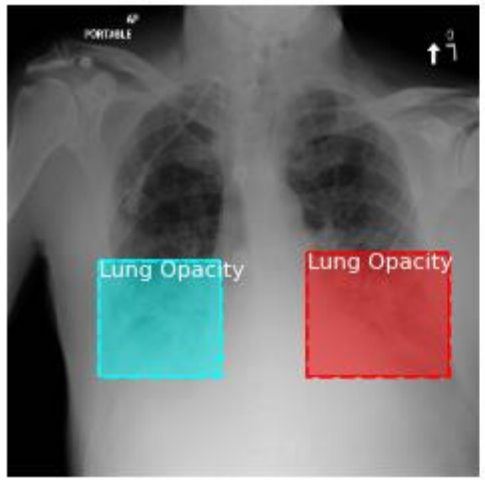
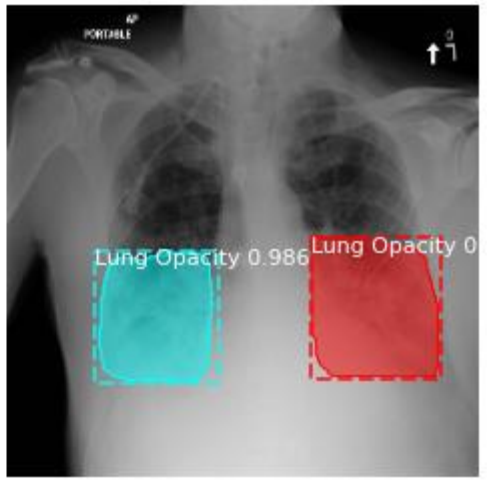
```
ytrue=[]
ypred=[]
for i in dataset_test.image_ids:
    image_id = i
    mask, class_ids = dataset_test.load_mask(image_id)
    if class_ids[0]==0:
        continue
    image_fp = dataset_test.image_reference(image_id)
    image = dataset_test.load_image(image_id)

    org_img,image_meta,gt_class_id,gt_bbox, gt_mask = modellib.load_image_gt(dataset_test,
                                                                             inference_config,
                                                                             image_id, u
                                                                             se_mini_mask=False)
    test_data.append(org_img)
    ytrue.append(gt_bbox)
    res = model_inference.detect([org_img])
    r = res[0]
    ypred.append(r['rois'])

    fig = plt.figure(figsize=(10,20))
    plt.subplot(1,2,1)
    visualize.display_instances(org_img, gt_bbox,
                                gt_mask, gt_class_id,
                                dataset_test.class_names,ax=fig.axes[-1])

    plt.subplot(1,2,2)
    visualize.display_instances(org_img, r['rois'], r['masks'], r['class_ids'],
                                dataset_test.class_names, r['scores'],ax=fig.axes[-1])
```

- Class 별 Inference 진행
- Class0 확인 → if(class_ids[0]==1):continue
- Class1 확인 → if(class_ids[0]==0):continue

- Prediction 진행

2. Inference

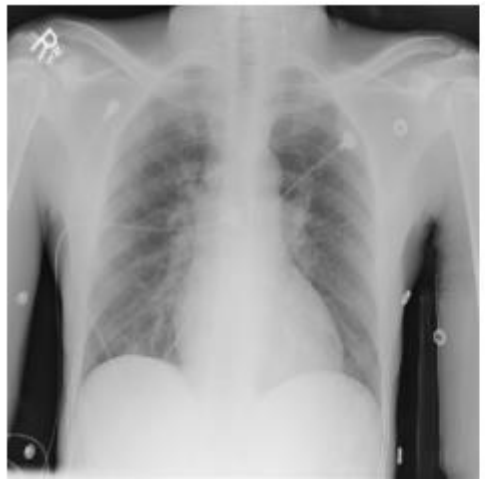
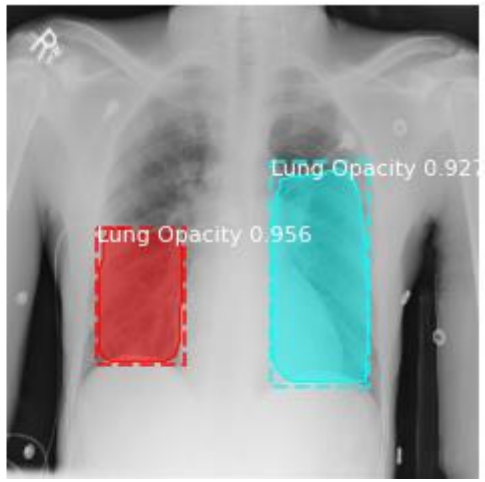
Class 1 (Pneumonia)

Ground Truth	Prediction
	
	

Test data 중 class 1 이미지를
prediction 했을 때,
거의 모든 이미지를
detection 성공
(; True Positive 높음)

2. Inference

Class 0 (Background)

Ground Truth	Prediction
	
	

반면, Test data 중 class 0 이미지를 prediction했을 때
거의 모든 데이터에서
Ground Truth 없는데 있다고 판단
(; False Positive 높음)

→ Recall은 높은데 Precision은 낮다고 판단되어
Hyperparameter 수정 후
새로 훈련

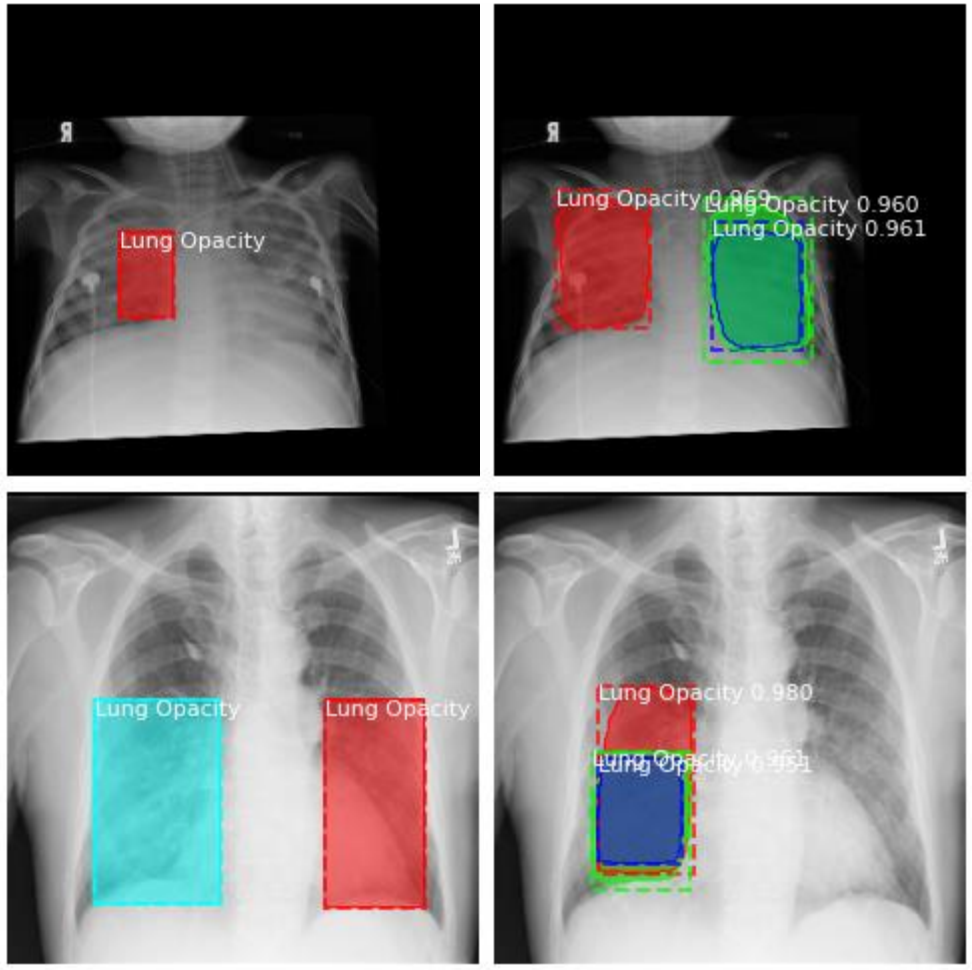
III-4. Modification

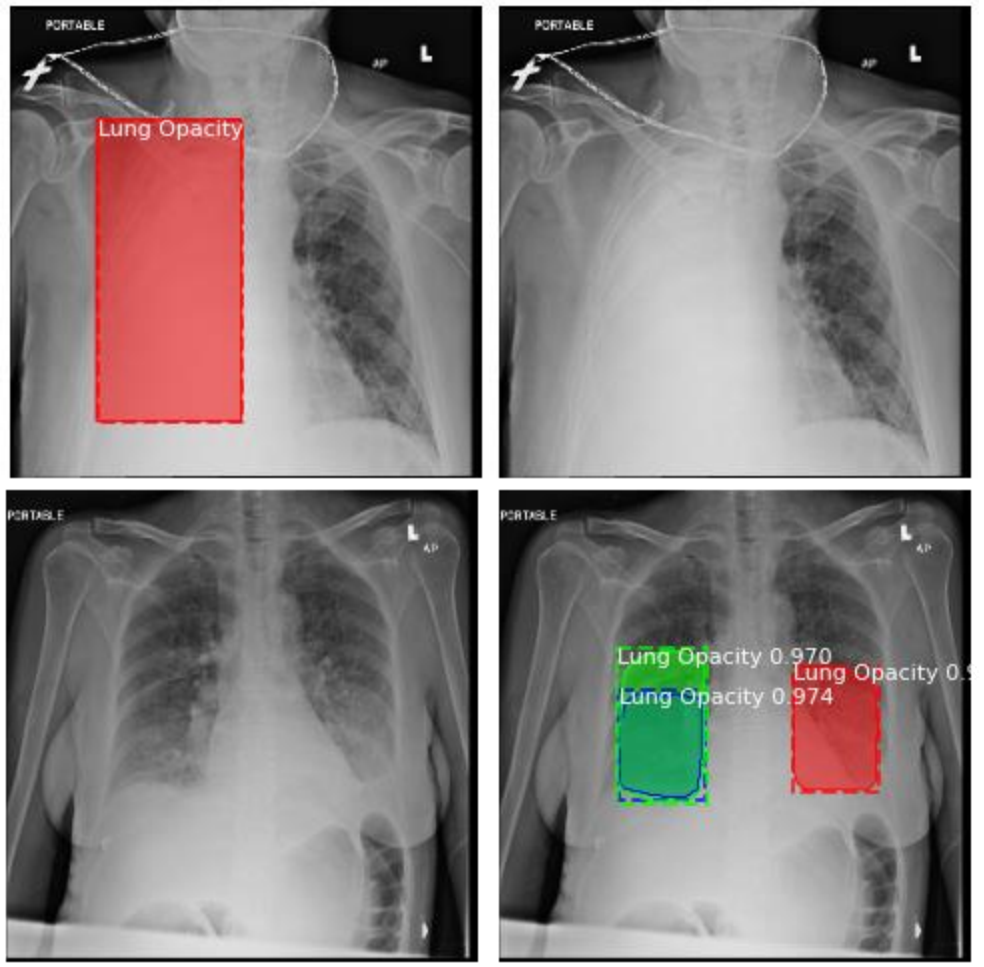
1. Hyperparameter 수정 - 1

```
DETECTION_MIN_CONFIDENCE = 0.9
DETECTION_NMS_THRESHOLD = 0.7
```

→ FP를 제거하기 위해 confidence threshold 증가

→ 1차 시도와 반대로 threshold 증가 → 겹치는 bbox를 제거 하지 못하는 결과 발생



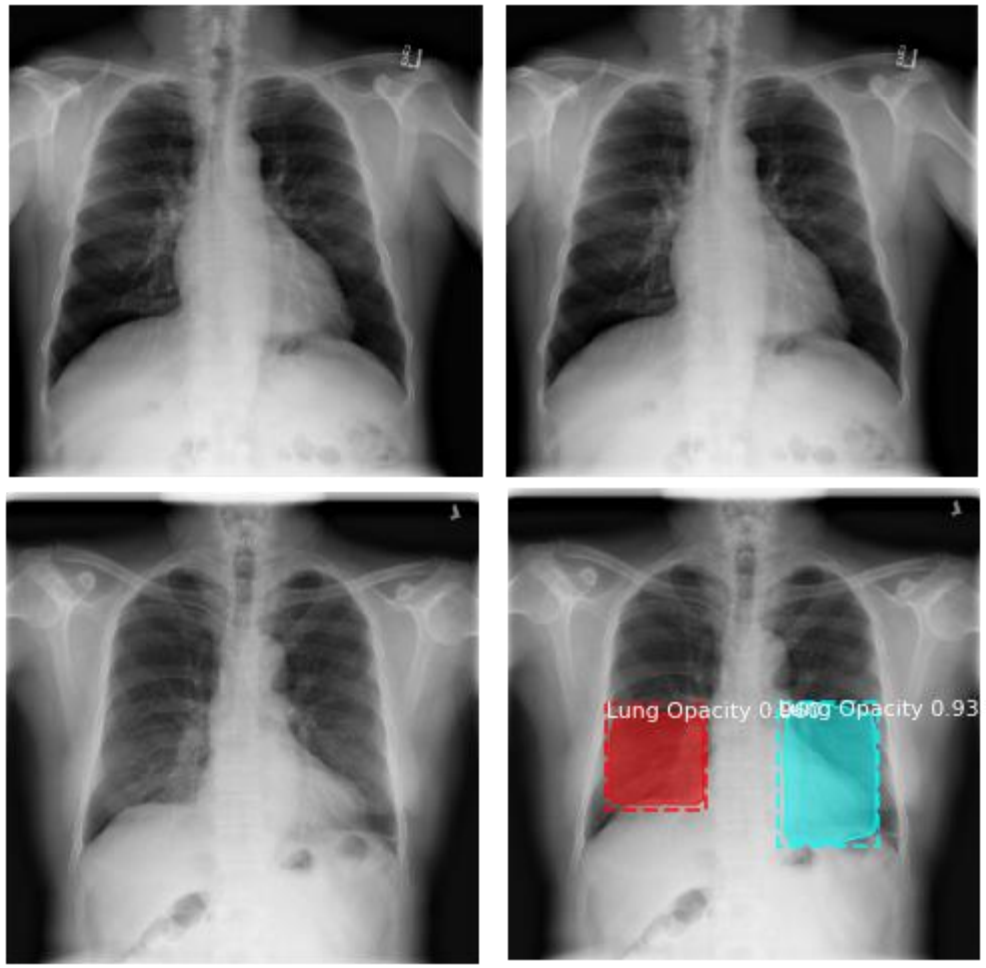


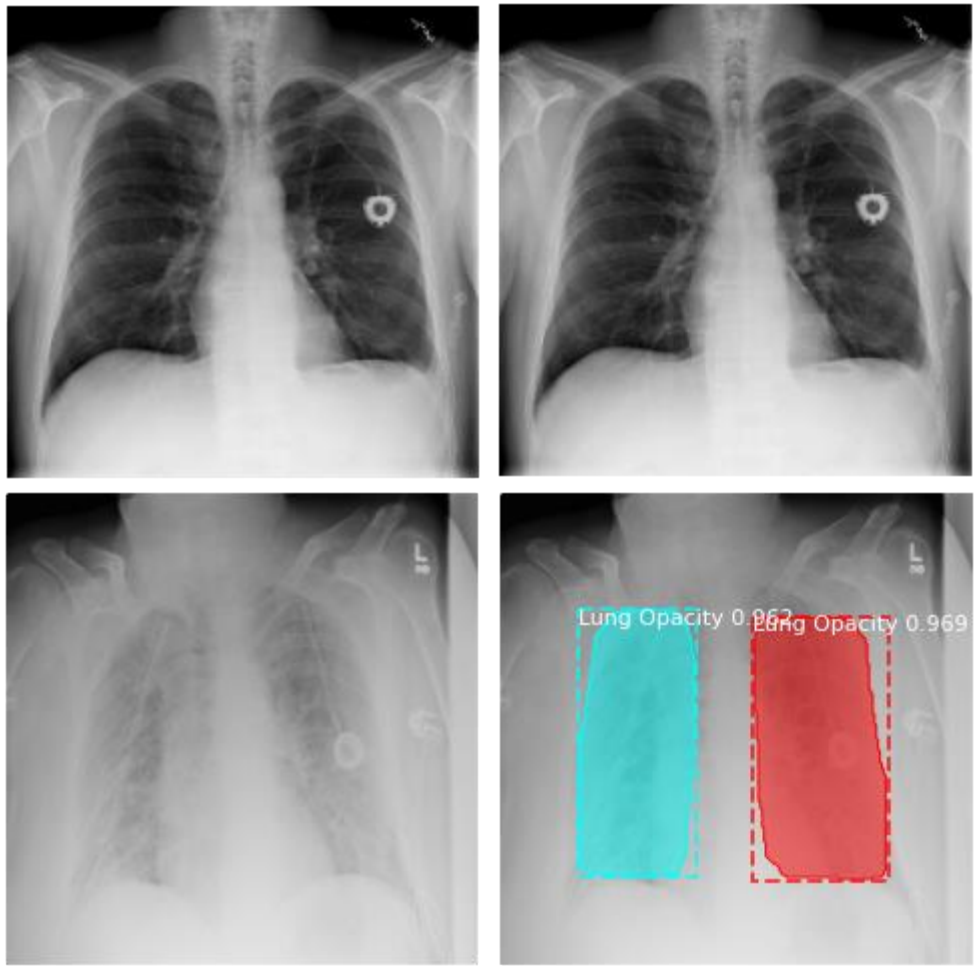
1. Hyperparameter 수정 - 2

```
RPN_NMS_THREDSHOLD = 0.3

DETECTION_MIN_CONFIDENCE = 0.92
DETECTION_NMS_THRESHOLD = 0.1
```

1차 수정에서 FP가 잘 잡히지 않아 값을 더 증가

1차 수정에서 값이 높으면 같은 곳을 가리키는 bbox가 제거되지 않는 것을 확인해 값을 감소





IV. 프로젝트를 마치며

IV. 프로젝트를 마치며

개선할 점	깨달은 점
• False Positive 제거 • confidence score가 0.95이상인 FP가 발생하는 이유 분석 • Detection 성능평가 실행 • Detection 속도 감소 • 다양한 시각화 구현	• GPU 세팅 및 알고리즘에 따른 라이브러리 버전관리 중요성 • Detection 알고리즘 개념 이해 • 2-stage detector 계보 논문 이해 • NMS에서의 threshold 역할 이해 • Github의 라이브러리를 클론하여 커스터마이징하는 방법

감사합니다