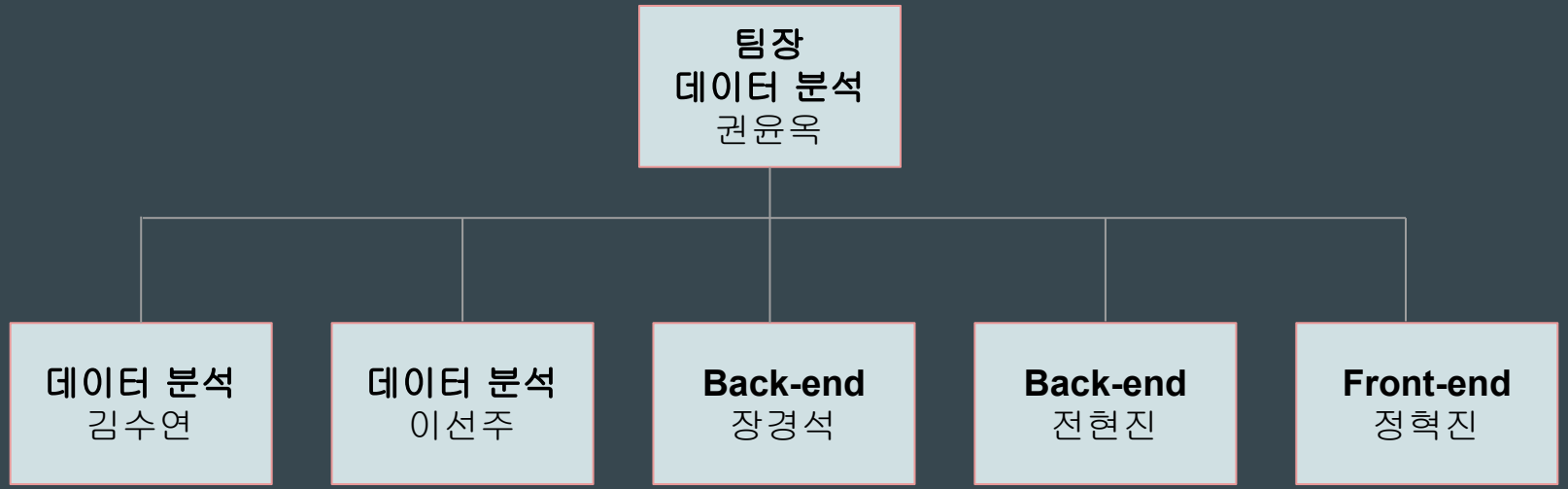
Steel Defect Detection

STEEL TO SUPREME

Index

about Team


팀장
데이터 분석
권윤옥
데이터 분석
김수연
데이터 분석
이선주
Back-end
장경석
Back-end
전현진
Front-end
정혁진

about Project

- 철강은 현대에서 가장 중요한 건축 자재 중 하나입니다.
- 철강 결함 식별을 자동화 하는 것은 생산성에 큰 도움이 됩니다.
- 철강 회사들은 자동화를 개선하고 효율성을 높이며 생산에서 고품질을 유지하기 위해 기계 학습을 찾고 있습니다.
- 이를 해결해보고자 프로젝트를 진행하였습니다.

about Schedule

구분	기간	활동
사전 기획	3/17	프로젝트 기획 및 팀 구성
	3/17	프로젝트 주제 선정
프로젝트 수행 및 완료	3/18	프로젝트 수행 - 데이터 분석 및 딥러닝 모델 선정 - 웹 어플리케이션 구상
	3/18 ~ 3/21	프로젝트 설계 - 모델 분석 및 시각화 - UI 구상 및 기능 설계 - 웹 어플리케이션 프레임워크 설계
	3/22	구현 및 테스트 - 모델 성능 테스트 - UI 구현 및 수정 - 웹 어플리케이션 구현 및 테스트
	3/23	최종 발표 - 구축 완료 보고

Result

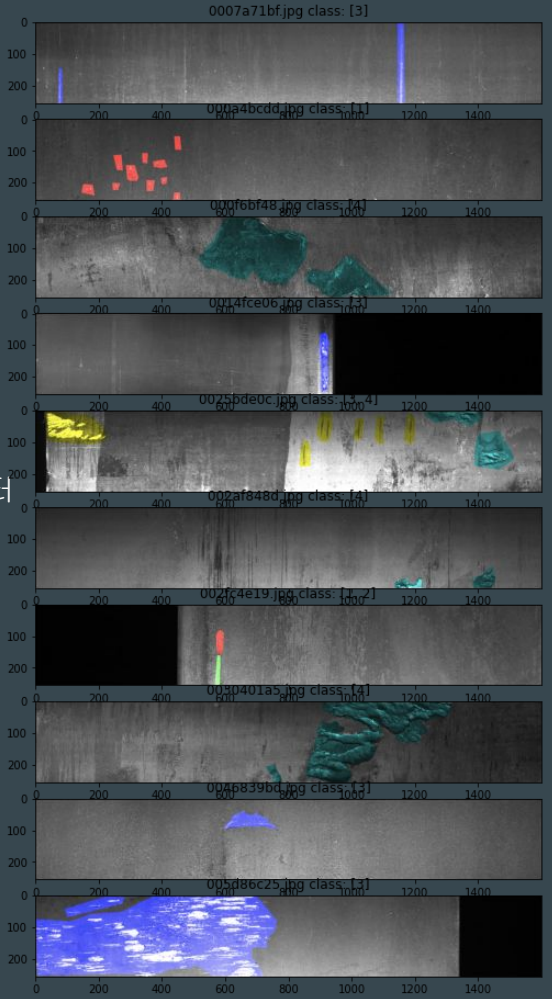
1. 데이터 수집

1-1. 데이터 출처

Kaggle 사이트 “Severstal: Steel Defect Detection”

1-2. 데이터 구성

- train_images : 학습에 필요한 철강 이미지 데이터셋
- test_imag3s : 학습후 결함 예측에 사용될 이미지 데이터셋
- train.csv : 학습에 필요한 철강 이미지 ID, Class ID, Encoded Pixels 데이터

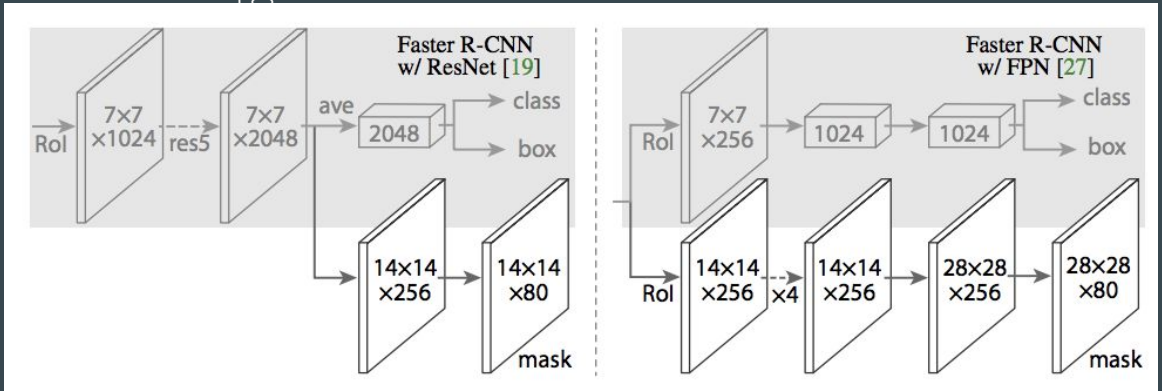


Result

2. 데이터 분석

2-1. 데이터 분석 모델 선정

- Mask R-CNN : Faster R-CNN(Object Detection)을 확장해 Instance Segmentation에 적용

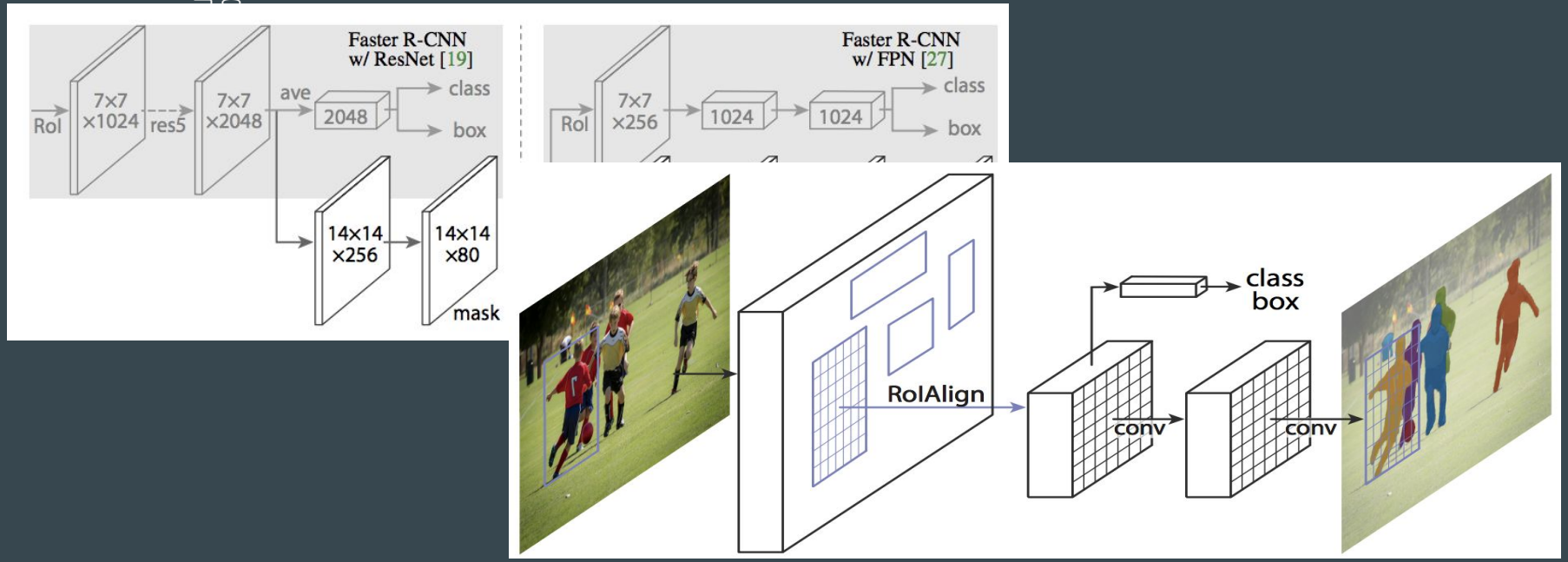


Result

2. 데이터 분석

2-1. 데이터 분석 모델 선정

- Mask R-CNN : Faster R-CNN(Object Detection)을 확장해 Instance Segmentation에 적용



Result

2-2. 데이터 전처리

- Image ID를 기준으로 데이터를 그룹화 -> 하나의 철강 이미지에 여러 가지 결함을 한번에 확인
- ‘Distinct Defect Types’ 컬럼을 추가하고 결함의 갯수를 표현

```
squashed = data[['ImageId','EncodedPixels','ClassId']].groupby('ImageId',as_index = False).agg(list)
squashed['Distinct Defect Types'] = squashed.ClassId.apply(lambda x: len(x))
squashed.head(10)
```

	ImageId	EncodedPixels	ClassId	Distinct Defect Types
0	0002cc93b.jpg	[29102 12 29346 24 29602 24 29858 24 30114 24 ...	[1]	1
1	0007a71bf.jpg	[18661 28 18863 82 19091 110 19347 110 19603 1...	[3]	1
2	000a4bcdd.jpg	[37607 3 37858 8 38108 14 38359 20 38610 25 38...	[1]	1
3	000f6bf48.jpg	[131973 1 132228 4 132483 6 132738 8 132993 11...	[4]	1
4	0014fce06.jpg	[229501 11 229741 33 229981 55 230221 77 23046...	[3]	1
5	0025bde0c.jpg	[8458 14 8707 35 8963 48 9219 71 9475 88 9731 ...	[3, 4]	2

Result

- 결함 클래스 별 이미지 수 시각화

Result

- EncodedPixels 컬럼 : 결함 위치 정보를 Run-Length Encoding(RLE, 비손실 압축 방법)한 정보

```
The mask reconstructed from the run-length encoding ("1 1 9 1") for our example would b
e:
[[1 0 0]
 [0 0 0]
 [0 0 1]]
```

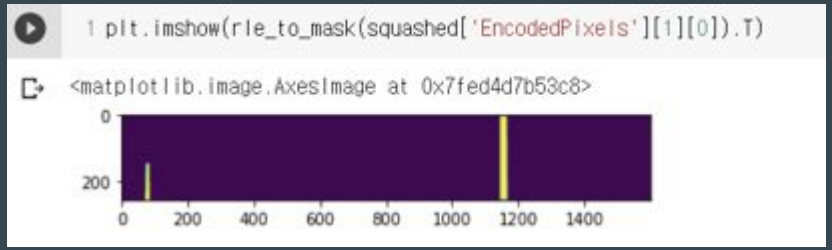
- RLE된 정보를 Decoding

```
[21]  1 rle_to_mask(squashed['EncodedPixels'][1][0])

array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

Result

- 디코딩한 정보는 철강 이미지에서 결함의 위치 정보를 나타냄



Result

2-2. 데이터 전처리

- 모델 설계 : Mask R-CNN 모델에 철강 이미지 데이터를 학습 시키기 위한 환경 설정

```
class SeverstalConfig(Config):

    NAME = "severstal"
    IMAGES_PER_GPU = 1

    # 분류할 클래스 개수
    NUM_CLASSES = 1 + 4  # background + steel defects

    # epoch당 training 단계 수
    STEPS_PER_EPOCH = 100

    # 60% 이하의 신뢰도를 가진 detection은 제외
    DETECTION_MIN_CONFIDENCE = 0.6

    # 성능이 낮은 가중치는 제외
    SAVE_BEST_ONLY = True

# SeverstalConfig 클래스 인스턴스 생성
severstal_config = SeverstalConfig()
```

Result

- 학습시킬 데이터셋 불러오기

```
RANDOM_SEED = 42

from sklearn.model_selection import train_test_split

# stratified split to maintain the same class balance in both sets
train, validate = train_test_split(squashed, test_size=0.2, random_state=RANDOM_SEED)
```

```
# training 데이터 셋 생성
dataset_train = SeverstalDataset(dataframe=train)
dataset_train.load_dataset()
dataset_train.prepare()

# validation 데이터 셋 생성
dataset_validate = SeverstalDataset(dataframe=validate)
dataset_validate.load_dataset()
dataset_validate.prepare()
```

Result

- 모델 생성 및 학습

```
# train을 위한 모델 생성
model = MaskRCNN(mode='training', config=severstal_config, model_dir='modeldir')

# 학습시키기 전 coco 가중치를 사용
model.load_weights('mask_rcnn_coco.h5',
                   by_name=True,
                   exclude=['mrcnn_bbox_fc',
                            'mrcnn_class_logits',
                            'mrcnn_mask',
                            'mrcnn_bbox'])
```

```
# training at last
model.train(dataset_train,
            dataset_validate,
            epochs=10,
            layers='heads',
            learning_rate=severstal_config.LEARNING_RATE)
```

Result

- 테스트 - 결함 감지

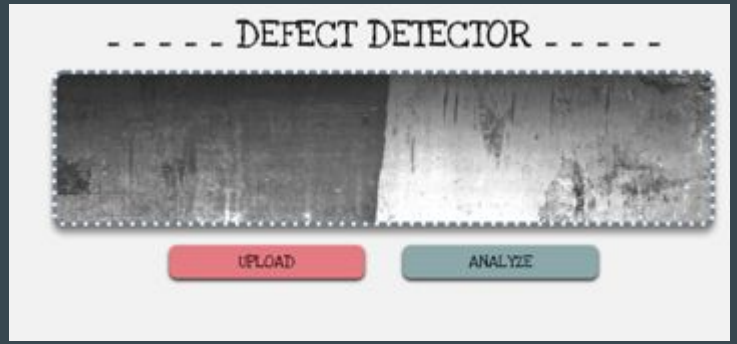
```
# Test on a random image
image_id = random.choice(dataset_validate.image_ids)
original_image, image_meta, gt_class_id, gt_bbox, gt_mask =#
    modellib.load_image_gt(dataset_validate, inference_config,
                           image_id, use_mini_mask=False)

log("original_image", original_image)
log("image_meta", image_meta)
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)

visualize.display_instances(original_image, gt_bbox, gt_mask, gt_class_id,
                            dataset_train.class_names, figsize=(8, 8))
```

Result

2-3. 데이터 분석 결과
- 웹에서 데이터 분석 결과 확인



Suggestion

1. Epoch 횟수를 늘인다.
2. Data Generator를 통해 학습 데이터를 늘인다.
3. Mask R-CNN 외의 모델을 적용해 기존의 모델과 성능을 비교해본다.
    - U-Net(의료 영상 분석에 많이 사용되는 모델)
4. 위에서 제안한 개선 사항을 통해 모델의 정확도 및 테스트 소요 시간을 줄여 나간다.

Expected Effect

고주파 카메라 이미지 ▶ 알고리즘 강화

▼

철강 제조 상향 표준화 ◀ 철강 결함 식별 성능 향상