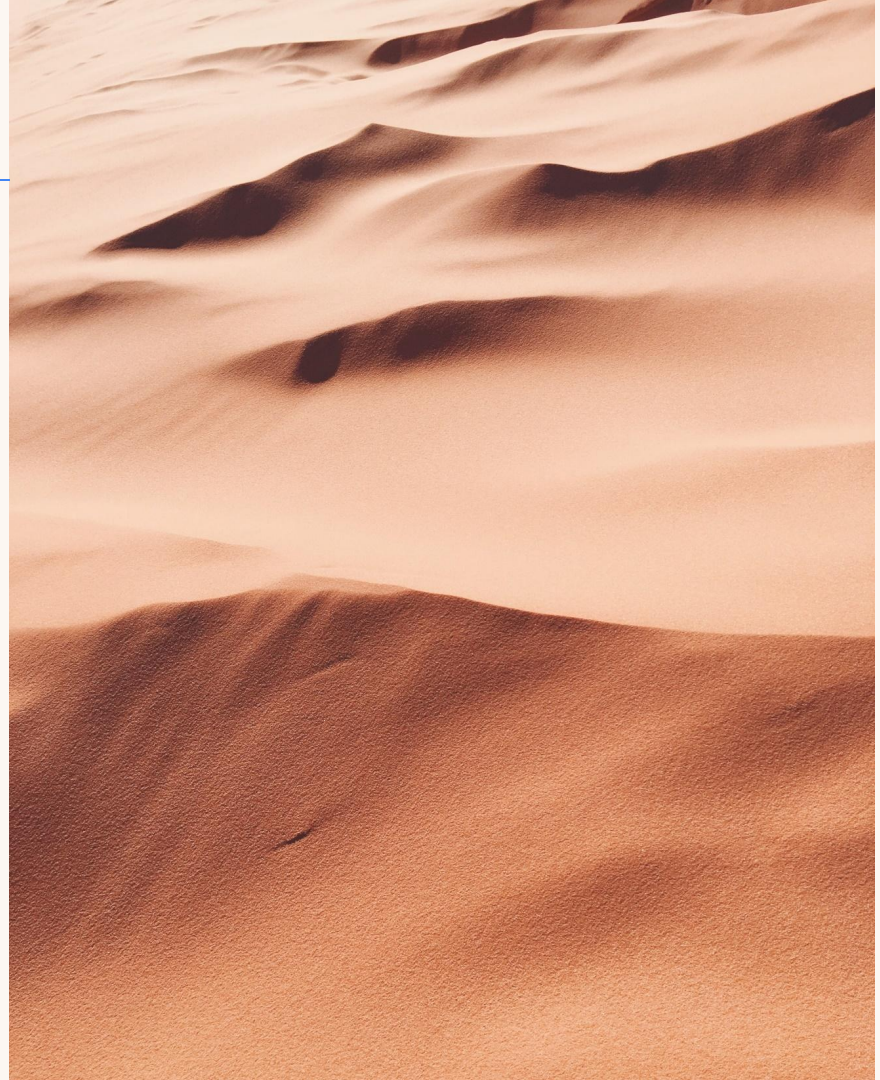
딥러닝 기반 CSQ 브래그 영역 탐지 : U-Net vs Detectron2

인턴 최종 발표
2021.12.03
김동화

목차 A table of contents.



지난 발표 리뷰

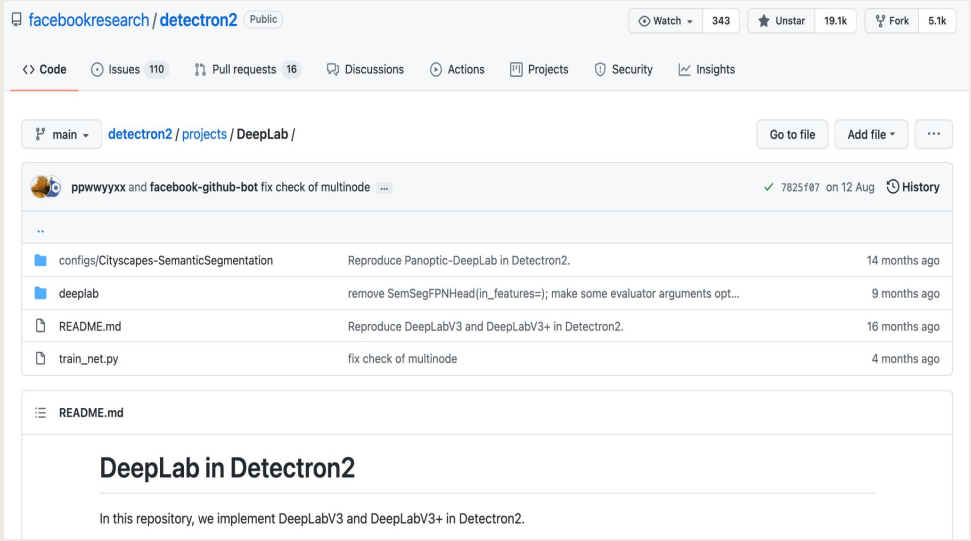
Detectron2(Mask R-CNN) vs Deeplabv3+ vs Unet

- Mask R-CNN : 객체가 있다고 예상되는 박스 영역 추출 후 그 안에서 픽셀 단위 분류 → 인스턴스 분할(Instance Segmentation) task 수행
- Deeplabv3+와 Unet : Encoder-Decoder 구조를 사용한다는 점에서 동일 → 의미론적 분할(Semantic Segmentation) task 수행
- Deeplabv3+ 이미지 전체의 문맥을 파악하는 장점, U-Net 모델 구조가 간단 • 직관적이며 객체와 배경을 분리하도록 학습

구현 사례 조사 : DeepLabv3+

Detectron2

〈 Detectron2 공식 깃허브에 있는 Deeplab 폴더〉



- Detectron2에서 구현 가능할 것으로 예상했지만 상세한 설명 및 코드가 없어서 실패
- Detectron2의 메인 수행 모델이 아니고 서브 프로젝트에서 개발하고 있는 모델이라서 사용자화시켜서 구현하기엔 정보의 한계가 있었음
- 기존에 블로그 글 등을 통해서 본 구현 사례들도 Deeplabv3+가 아니라 메인 모델인 Mask R-CNN 모델이라서 참고하기 어려움
- Deeplabv3+이 외에도 Panoptic-Deeplab, TensorMask 같은 모델도 정보의 부족으로 실제 구현은 어려울 것으로 보임

구현 사례 조사 : DeepLabv3+

Tensorflow

- tensorflow 공식 github에 나온 deeplabv3+ 소스코드 : tensorflow 버전 1.X를 기반으로 하기 때문에 현재의 버전 2.X과 충돌
- 현재 설치된 tensorflow 삭제 후 하위 버전 설치를 시도했으나 실패
- tensorflow 1.X 에서 2.X로 업그레이드하면서 바뀐 함수가 너무 많아서 직접 코드를 수정하기에는 시간상의 제약으로 불가능
- tensorflow 2.X에서 deeplabv3+ 수행하는 개인 github를 찾았지만 설명 부족으로 구현 실패

참고한 깃허브, 블로그

- Tensorflow 공식 깃허브 [링크] – 버전 문제
- 블로그 “구글 DeepLab v3+” [링크] – 버전 문제, 참고한 블로그 게시글 삭제
- Tensorflow2.2에서 수행하는 DeepLabv3+ 깃허브 [링크] – 설명 부족
- Keras로 수행하는 DeepLabv3+ 깃허브 [링크] – 설명 부족

U-Net 구현

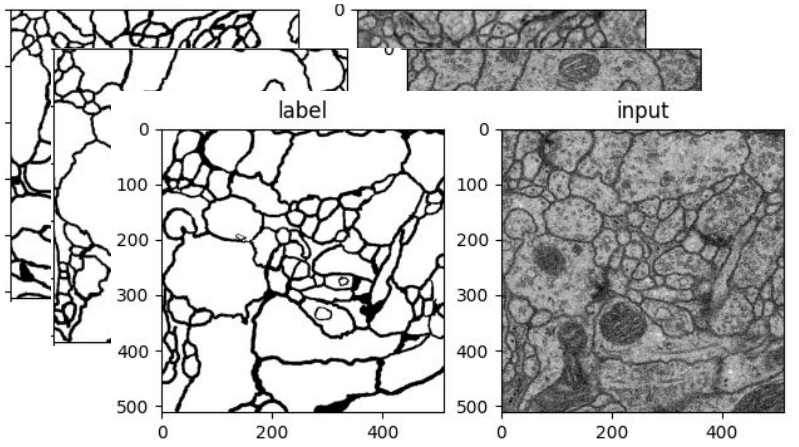
깃허브(링크)에 샘플 데이터로 구현된 코드

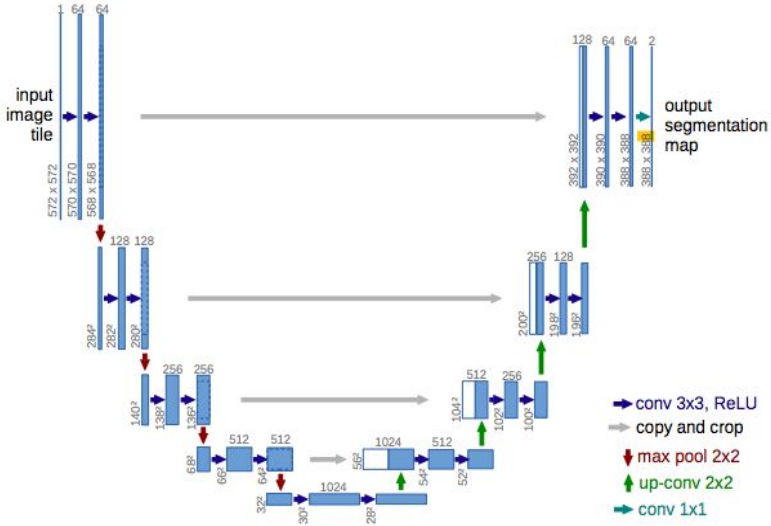
기본 U- Net 구조 : 4단계의 Encoder + Decoder

Deeplabv3+에 비해서 네트워크의 깊이가 깊지 않다는 단점이 있지만 사용자 데이터를 간편하게 훈련시키기 좋음

샘플 데이터

- 512x512 크기의 흑백(channel=1) 이미지 30개 → train 3개 / val 3개 / test 3개로 분할하여 훈련/검증/테스트 수행
- 훈련 및 검증 결과 꽤 잘 분할
- 레이블 0: 배경 255: 세포으로 이진 분류 → 커스텀시 전처리 필요 0: 배경 255:브래그영역으로

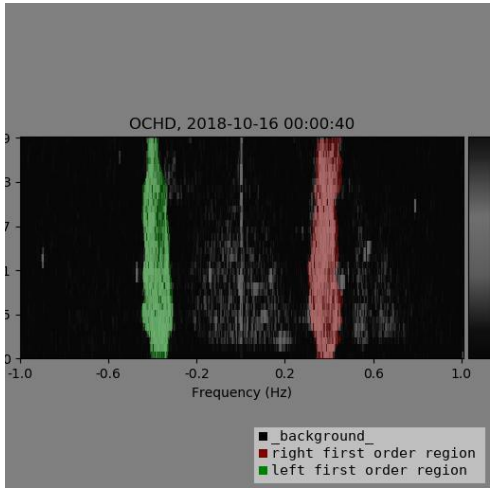




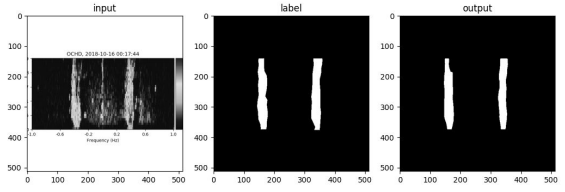
데이터 전처리 및 훈련, 테스트

실행 내용	파일/폴더 위치
이미지(png)와 레이블(json) 파일 U-Net에 입력하기 위한 npy 형식으로 input과 label 파일 생성 input은 흑백 이미지로, label은 0:배경/255:브래그영역으로 레이블링 *0:배경/1:오른쪽 브래그 영역/2:왼쪽 브래그 영역으로 레이블링 후 훈련 시, 테스트셋 예측 실패 → 클래스끼리의 레이블링 값 차이가 커야 잘 학습되는 것으로 보임	labelme/ • data/ : 입력 데이터 폴더 (png, json) • json2npy.py : 실행 파일 • results/ : npy 저장 폴더
총 108개의 이미지 중 train 100개 / test 3개 / val 5개로 나누어 데이터셋 준비 *detectron2와 동일한 test 데이터셋 사용	datasets/train,test,val/ : 입력 데이터 폴더
훈련 및 모델 저장	train.py : 실행 파일 checkpoint/ : 학습 모델 저장 폴더
훈련 모델 로드 및 테스트셋 적용	eval.py : 실행 파일 results/numpy, png/ : 테스트셋 예측 결과 저장 폴더
결과 시각화/mIoU 계산	display_results.py/miou.py : 실행 파일

〈전처리 결과 예시〉



〈테스트셋 예측 결과 예시〉

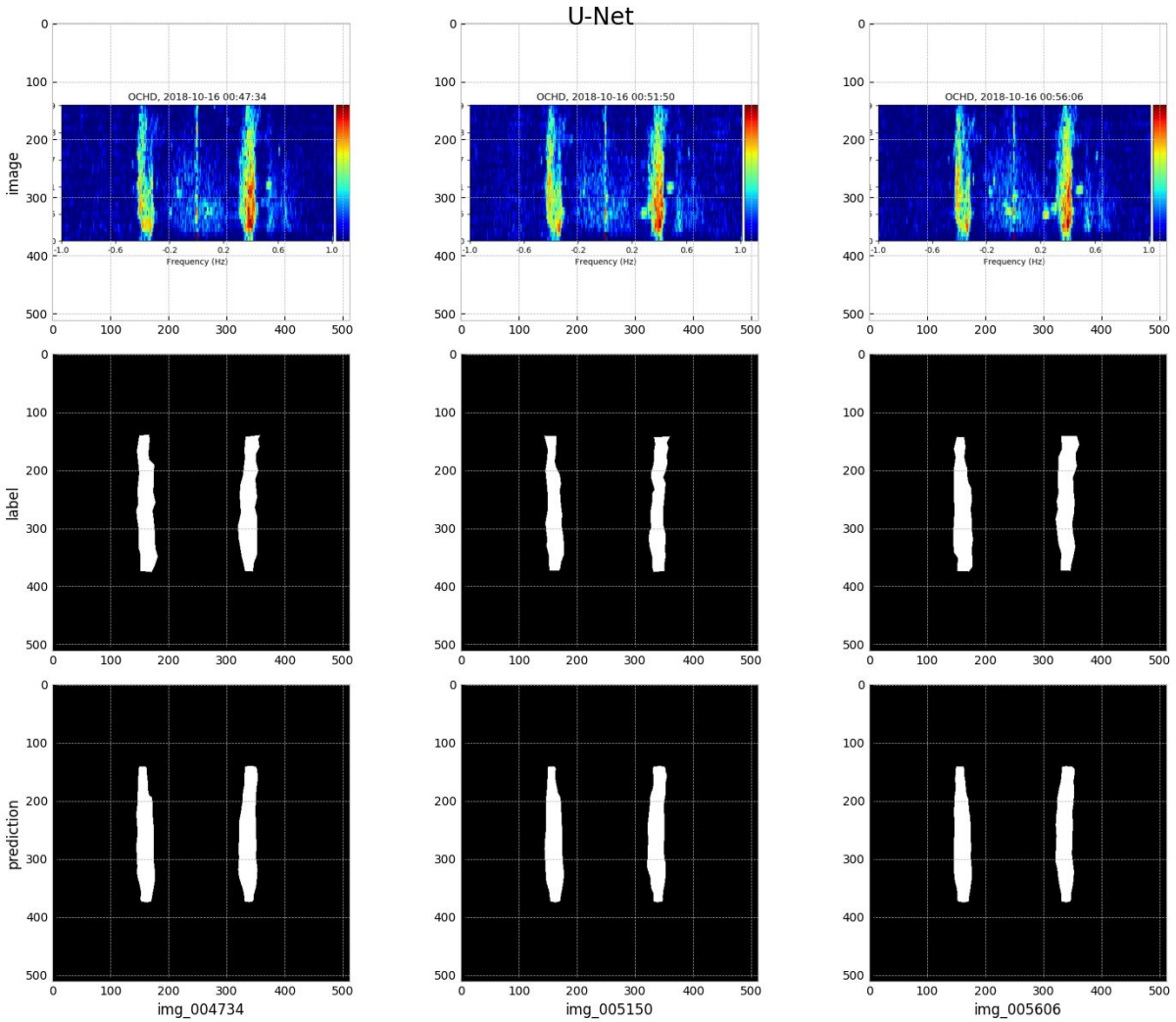


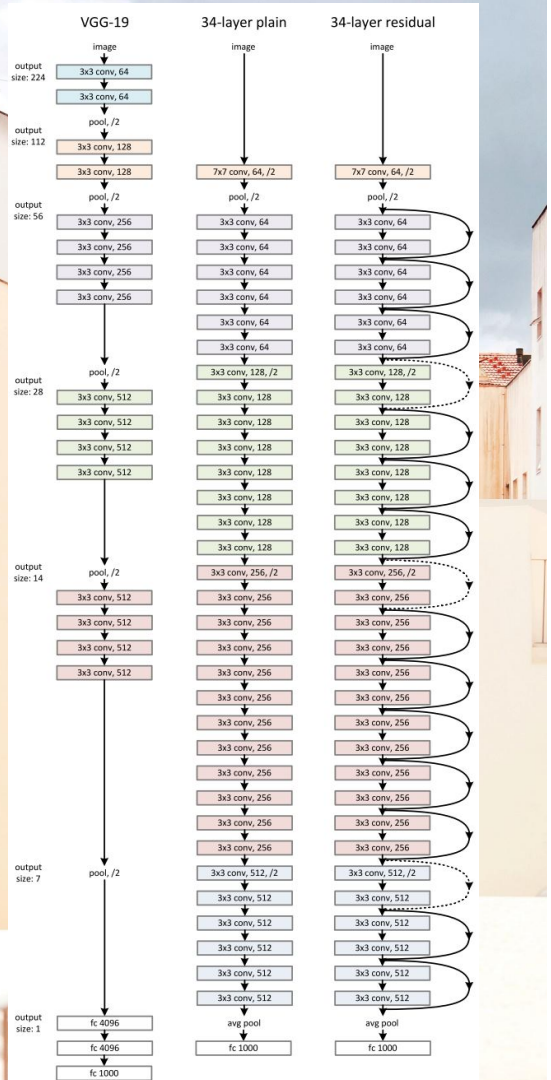
U-Net 결과

1) 대체적으로 레이블보다 더 매끄러운 경계를 예측
2) 레이블(정답지)보다 더 CSQ 이미지에 일반화가 잘 된 것처럼 보임. (예. img_005150)
3) 오히려 레이블보다 실제 이미지의 브래그 영역을 잘 탐지하는 것으로 보임.
4) 만약 그렇다면 실제 평가 지표보다 성능이 더 좋을 가능성이 있음

* 평가 지표 mIoU

- 왼쪽 : 87.59%
- 오른쪽 : 84.13%
- 전체 : 85.86%





Mask R-CNN 구현

기존 Detectron2 연구 코드 사용

COCO*-Instance Segmentation 데이터로 학습된 Mask R-CNN 모델을 베이스로 CSQ 데이터 추가 학습

Detectron2에서 제공하는 모델 중 ResNet50+FPN(Feature Pyramid Network)을 backbone으로 사용

- ResNet은 기존에 레이어를 쌓을 수록 최적화 성능이 떨어지는 문제를 해결한 CNN구조, 더 사이즈가 큰 /높은 해상도의 이미지를 네트워크가 입력받아서 학습할 수 있게됨 = “더 깊은 네트워크”
- FPN은 이미지 내에 다양한 크기의 피처/객체들을 잘 인식할 수 있도록 만든 네트워크

old_sample의 train 105개, test 3개 분할하여 학습 및 테스트

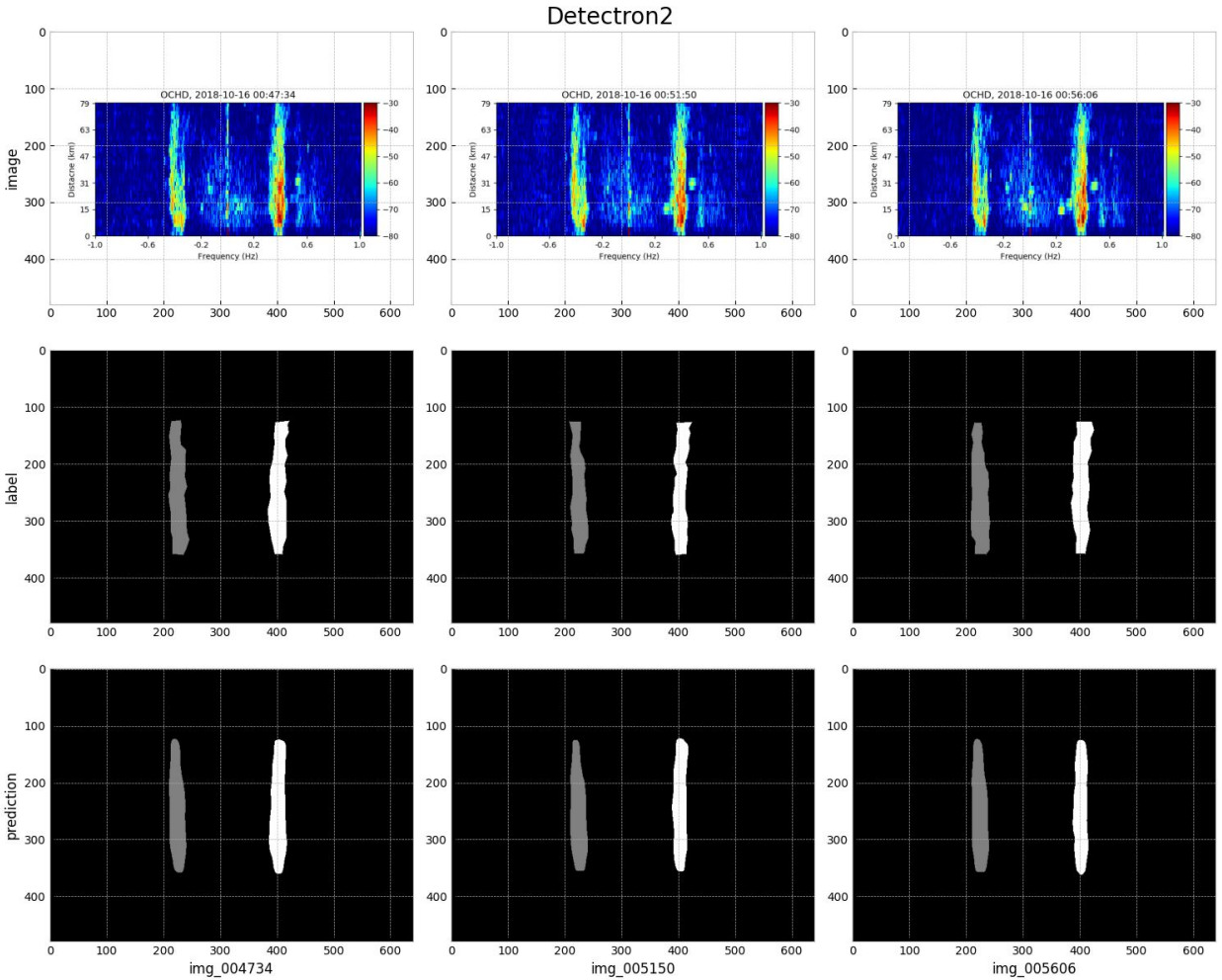
U-Net과 동일한 평가 기준으로 비교하기 위해서 label(정답지)와 예측 결과를 npy 파일로 저장하여 mIoU 계산

Mask R-CNN 결과

1) U-Net과 동일하게 레이블보다 매끄러운 경계로 예측
2) U-Net 결과보다 더 일반화가 되어있고 경계가 더 매끄러움
3) 대체적으로 얇은 브래그 신호를 예측해냄

*평가지표 mIoU

- 왼쪽 : 88.63%
- 오른쪽 : 81.25%
- 전체 : 84.94%



Part 3

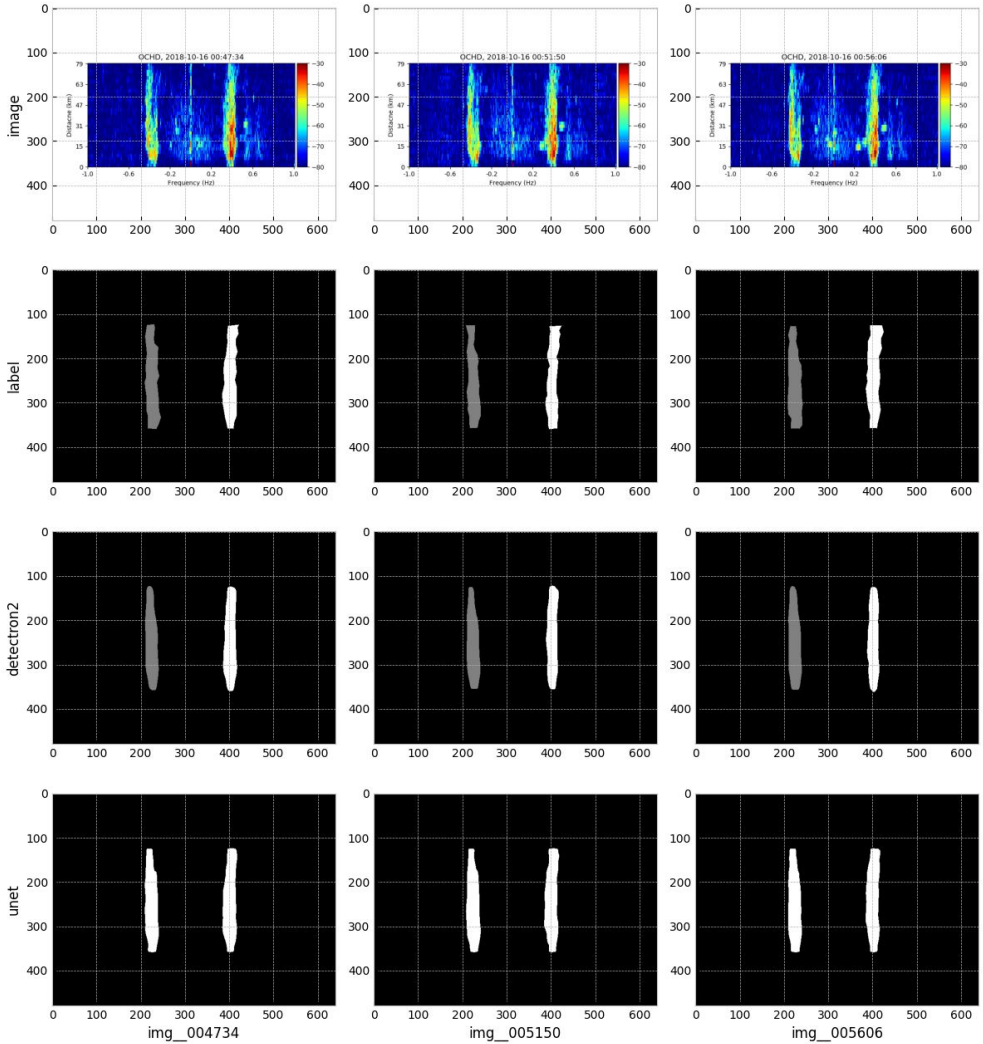
U-Net vs Mask R-CNN 결과 비교

브래그 영역의 형태나 신호가 약해지는 부분에서는 배경으로 예측하는 경향성은 U-Net과 Mask R-CNN 모두 보인다.

왼쪽 브래그 영역에서의 mIoU보다 오른쪽이 더 낮은 성능을 보이는데, 분산을 비교해본 결과 오른쪽의 mIoU의 분산이 4배 정도 더 컸다. 이미지마다, 브래그 영역의 레이블에 의존하는 경향이 크고 test 이미지가 세 개 밖에 되지 않기 때문에 이러한 편향이 생긴 것으로 이해된다. 훈련 : 검증 데이터셋 비율 조정

전체 수치 상으로는 U-Net이 Mask R-CNN보다 약 1% 정도 성능이 좋다.

브래그 영역 구분	U-Net	Mask R-CNN
왼쪽	87.59%	88.63%
오른쪽	84.13%	81.25%
전체	85.86%	84.94%



결론 & 한계점

U-Net이 네트워크가 더 얕지만 브래그 신호 영역 탐지에서는 Mask R-CNN만큼의 성능을 보인다.

Mask R-CNN은 일상 사진인 COCO 데이터셋을 학습시킨 베이스라인 모델을 사용했는데 브래그 신호 영역 탐지에서는 그렇지 않은 U-Net과의 차이가 크지 않은 것으로 보인다. U-Net 예측 결과보다는 경계가 더 매끄러운 것은 확인할 수 있었다.

Mask R-CNN의 예측 영역의 폭이 좁은 이유도 객체의 경계면을 더 정확하게 확보하기 위해서 부정확한 경계는 날려버리기 때문인 것 같다. 이를 선박 탐지에 적용한다면 보다 확실한 경계면을 확보한다는 장점이 될 수 있지만 약하고 부정확한 신호는 무시하는 단점을 가질 수도 있겠다.

브래그 영역 탐지 결과만 가지고 선박 탐지에서 어떤 모델이 더 성능이 좋을지 판단하기는 어렵다. 하지만 본 프로젝트에서 진행한 U-Net 코드로는 다중 분류가 불가능하기 때문에 이를 위해서는 U-Net의 multi-class 활용 사례를 더 찾아봐야할 것 같다.

레이블이 흑백 이미지에만 있어서 신호가 강한 부분(빨강색)~약한 부분(하늘색)에 대한 세부 학습 및 예측은 불가능했다. 향후에 r/g/b 채널 1개씩 분해해서 각 채널에서 브래그 신호 영역을 레이블한 후 딥러닝을 수행하면 조금 더 디테일한 예측이 가능하겠다.

감사합니다

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