2019년 2학기 인공지능 최종 과제

under standing Clouds from images

# **주제 설명** 기후변화는 다양한 의사 결정 과정에서 중요한 역할을 하고 있다. 구름은 기후변화를 예측하는 데에 있어서 중요한 역할을 하는데, 형태에 대한 이해가 어렵고 구름 종류 간의 경계가 모호해 구름을 활용한 기후 모델을 만드는 것은 어려운 일이 되고 있다. 이에 따라 과학자들은 위성영상을 통해 구름을 4가지 타입(Flower, Fish, Gravel, Sugar)으로 분류했고, 기계 학습을 통해 정확성이 높은 구름 분류 모델을 만들 수 있다면 앞으로 기후 예측의 정확성을 높일 수 있을 것으로 예상하고 있다. 따라서 이 과제에서는 구름의 위성영상을 입력으로 받았을 때, 최대한 정확하게 해당 영상의 구름을 알맞게 분류하는 모델을 만드는 것을 목표로 한다

# **조team 설명**

조이름: KMUAI\_CAT

곽홍재(20172813)

권민수(20142697)

수행한 업무 분장 (기여도 %) : 50 : 50

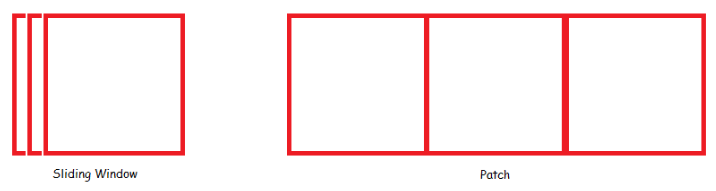
# **적용한 방법 : u-net + post-processing**

1. 세부 설명

U-Net은 네트워크 형태가 알파벳 U와 비슷하게 생긴 신경망이다

기존 Segmentation Network들의 문제점을 크게 두 가지로 나눴을 때 느린 속도의 문제, Localization – Context 간의 Trade-off 문제가 있었다.

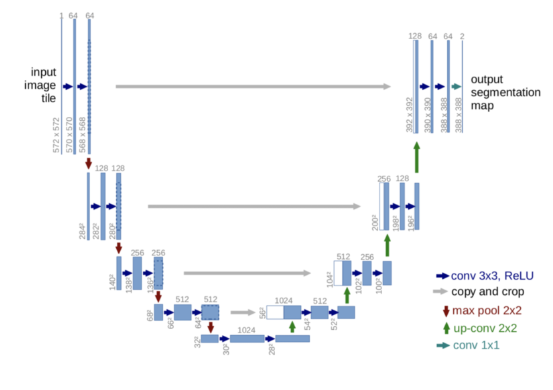
먼저 속도의 경우는 Sliding Window를 사용할 시 이미 사용한 구역을 다음 Window에서 다시 검증하는 과정을 거쳤기 때문에 똑 같은 일을 반복하는 비효율을 보였다. 이를 해결하기 위해 U-Net은 이미 검증이 끝난 부분은 아예 건너뛰고 다음 Patch 부분부터 검증을 진행해 속도가 빠르다.



다음으로 Trade-off 문제는 Patch-Size의 문제라고 볼 수 있는데, 이 Patch Size를 너무 크게 하면 Context 인식이 유리하지만 Localization에 불리해지고, 너무 작게 하면 Localization은 유리하지만 Context 인식이 불리해지는 Trade-Off 문제가 있었다. 이를 해결하기 위해 U-Net은 여러 Layer의 Output을 동시에 검증해 Localization과 Context 인식 두 마리 토끼를 모두 다 잡았다.

추가적으로 U-Net은 Image Mirroring 기법을 사용한다. Contracting Path에서이미지의 외곽부분이 잘리는 것이 누적되어 output의 사이즈가 작아지는 문제가 있었는데, 없어지는 부분을 Zero-padding 하는 대신 안쪽 이미지를 거울에 반사시킨 형태인 Mirror Padding 기법을 사용해 사이즈를 유지한다.

U-Net의 Architecture는 다음과 같다



* 컨볼루션 층 : 3x3 Convolution 사용, 활성함수 = ReLU
* 풀링 층 : 2x2 MAX Pooling, 매 층마다 1/2 다운샘플링, 채널은 2배
* Expanding Path(이미지 크기를 키우는 부분) : 2x2 UP Convolution
* Mirror Padding에 대한 보상 :   
  Contracting Path의 데이터를 적당한 크기로 자른 후, Concat 하는 방식으로 보강

추가적으로 Output에 대한 Post-processing을 진행했다. Post Processing은 Precision – Recall Curve를 이용해 라벨 각각의 최적의 Recall Threshold를 구해서, 해당 Threshold 값을 통과한 Output만 값이 유효한 것으로 판단하여 EncodedPixels 값을 채워서 제출하도록 했다.

여기서 Precision은 정확도(실제 물체가 얼마나 포함되어 있는지)를 의미하고, Recall은 검출율(대상 물체를 빠트리지 않고 얼마나 잘 잡아내는지)를 의미하며, 일반적으로 둘은 반비례 관계이기 때문에 Graph를 통해 최적의 값을 찾아내는 과정이 필요하다.

1. 관련 연구 동향 및 배경 지식

**Segmentation**

이미지를 찾아서 박스를 치는 Detection과는 달리 이미지를 픽셀단위로 분류하는 기법이다. 즉, 각 픽셀이 어느 클래스에 포함될 확률이 높은 지 판별해내는 기법이라고 할 수 있다.

Segmentation은 Semantic Segmentation과 instance Segmentation두가지로 구분할 수 있다.

Semantic Segmentation은 클래스만 구분하는 기법이고 instance Segmentation은 같은 클래스라도 다른 것을 분류한다.

우리 과제에서 사용하는 것은 Semantic Segmentation이다.

Semantic Segmentation은 인코더와 디코더 형식과 비슷하다.

인코더에서는 CNN을 통해 Feature map을 출력하고 이미지의 사이즈를 줄인다.

사이즈가 줄어들고 차원이 달라진 상황에서는 Segmentation을 할 수 없기 때문에 Up-sampling 기법을 통해서 Feature map을 다시 원 공간의 크기로 바꾼다.

대표적인 알고리즘으로는 FCN, deep lab, U-Net 등이 있다.

1. 상세한 구현 환경, 구현 방법

* 캐글 커널, Google CoLabatory, Local에서 진행
* Tensorflow(keras 2.2.4-tf), keras 2.3.1기반으로 모델링, segmentation 모델은 텐서플로우와 케라스 기반으로 만들어진 Segmentation models 패키지 사용
* post-processing에는 EfficientNet을 사용(텐서플로우와 케라스 기반)
* 기타 openCV2, numpy, pandas, sklearn, seaborn, matplotlib 패키지 사용

다양한 Discussion과 notebook에서 성능이 높은 사람들이 처음에는 Segmentation models패키지를 통해 Semantic Segmentation을 하고 이미 검출된 클래스를 보정하는 post-processing을 적용한다는 사실을 알아냈다.

캐글을 참조해서 Segmentation models패키지를 통해 기본 모델을 만들고 post-processing을 적용하는 방향으로 구현함

1. Segmentation models의 Unet 모듈에서 resnet34 encoder 구조를 가져와서 학습시킨다

(class의 개수와 마지막 층의 activation, data generator를 통해 들어오는 이미지의 shape를 추가로 지정해준다.)

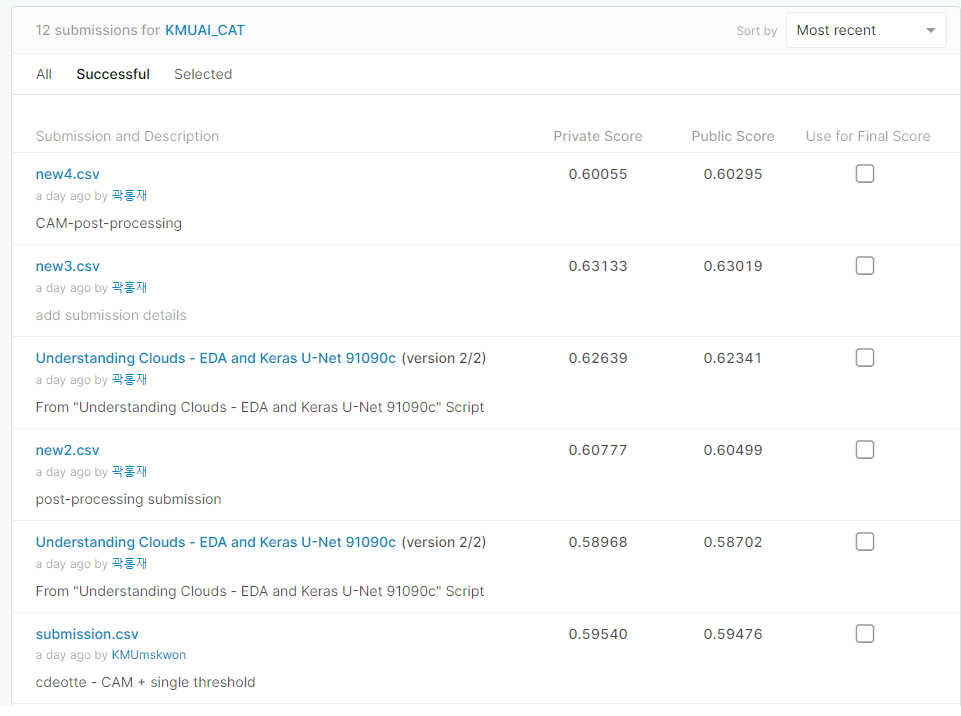
1. 예측 값에 적용하고 1차 submission을 만든다.
2. EfficientNet에서 pre-train된 Imagenet가중치를 사용해서 모델을 학습시킨 뒤 클래스별로 threshold값을 구한다.
3. 구한 threshold값을 통해 1차 submission을 수정하여 최종 submission을 만들어 낸다.
4. 코드 상세 설명

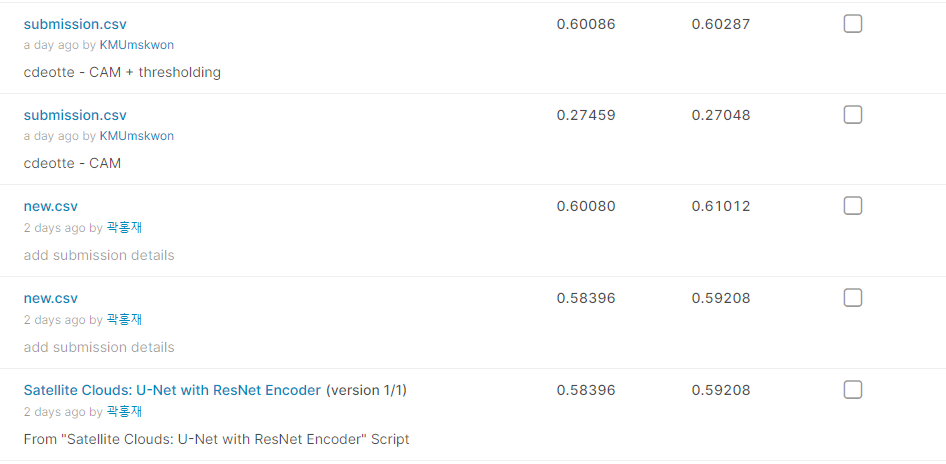
1차 서브미션 : U-Net+ResNet.pdf

2차 서브미션 : EfficientNet.pdf

# **결과**

제출한 모든 결과들을 출력해서 화면





# **결론**

조사를 하는 과정에서 U-Net이 Localization과 Context Tradeoff를 해결한 방식이 놀라웠다. Zero base에서 신경망을 짜는 것은 힘들 것 같아서 다른 모델들과 코드를 많이 참고했는데, 시간이 된다면 아예 zero base에서 시작해보고 싶다.