

# [위성관측 데이터 활용 강수량 산출 AI 경진대회]

컴퓨터공학과  
12161532 김난영  
제출 날짜 : 2020년 5월 25일

## 1. 모델 훈련 조건 설명

Baseline의 코드를 기반으로 모델을 훈련시켰다.

### 1-1) Baseline 코드 상의 조건

- 밝기 온도 채널만 사용해보기 위해 0~8 채널만 사용
- 50개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.25)
- epochs = 2

점수 결과 : 4.6908

### 1-2) 조건 1

```
def trainGenerator():  
    train_path = '/Users/kimnan-young/train'  
    train_files = sorted(glob.glob(train_path + '/*'))  
  
    for file in train_files:  
        dataset = np.load(file)  
  
        target = dataset[:, :, -1].reshape(40, 40, 1)  
        cutoff_labels = np.where(target < 0, 0, target)  
        feature = dataset[:, :, :14]  
  
        if (cutoff_labels > 0).sum() < 60:  
            continue  
  
        yield (feature, cutoff_labels)  
  
train_dataset = tf.data.Dataset.from_generator(trainGenerator, (tf.float32, tf.float32), (tf.TensorShape([40, 40, 14])
```

- 채널 14개 모두 사용
- 60개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)

- epochs = 2

Baseline의 코드에서 채널의 수와 픽셀 합 조건과 Dropout 비율을 바꿔보았다.  
그러나 Baseline보다 더 성능이 떨어짐을 확인할 수 있었다.

점수 결과 : 6.7378666078점

### 1-3) 조건 2

- 0~8 채널만 사용
- 60개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 2

조건 1에서 사용한 채널 수만 바꿔보았다.

점수 결과 : 3.2355969687

### 1-4) 조건 3

- 0~8 채널만 사용
- 60개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 5

조건 2에서 epoch 수만 바꾸어보았다.  
성능이 향상되었다.

점수 결과 : 2.9763225941

### 1-5) 조건 4

- 0~8 채널만 사용
- 70개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.35)
- epochs = 5

조건 3에서 픽셀 합 조건과 Dropout 비율을 바꿔보았다.

그러나 loss 값이 조건 3과 비교했을 때 0.5, 0.35 와 같이 높게 찍혀서 중간에 실행을 중단했다.  
중단했기에 제출을 하지 않았고 점수 결과를 알 수 없었다.

## 1-6) 조건 5

- 0~8 채널만 사용
- 70개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 5

조건 3에서 픽셀 합 조건만 바꿔보았다.  
그러나 조건 3보다 점수가 더 낮게 출력되었다.

점수 결과 : 3.3780555555

## 1-7) 조건 6

- 0~8 채널만 사용
- 60개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 10

```
model_history = model.fit(train_dataset, epochs = 10, verbose=1)
```

```
Epoch 1/10
56/56 [=====] - 780s 14s/step - loss: 0.4609 - maeOverFscore_keras: 5.7412 - fscore_keras: 0.3794
Epoch 2/10
56/56 [=====] - 786s 14s/step - loss: 0.3203 - maeOverFscore_keras: 3.6157 - fscore_keras: 0.5313
Epoch 3/10
56/56 [=====] - 668s 12s/step - loss: 0.3181 - maeOverFscore_keras: 3.4821 - fscore_keras: 0.5305
Epoch 4/10
56/56 [=====] - 672s 12s/step - loss: 0.3097 - maeOverFscore_keras: 3.1541 - fscore_keras: 0.5676
Epoch 5/10
56/56 [=====] - 642s 11s/step - loss: 0.3021 - maeOverFscore_keras: 2.8564 - fscore_keras: 0.6058
Epoch 6/10
56/56 [=====] - 627s 11s/step - loss: 0.2986 - maeOverFscore_keras: 2.7792 - fscore_keras: 0.6154
Epoch 7/10
56/56 [=====] - 787s 14s/step - loss: 0.2953 - maeOverFscore_keras: 2.7255 - fscore_keras: 0.6202
Epoch 8/10
56/56 [=====] - 762s 14s/step - loss: 0.2977 - maeOverFscore_keras: 2.8387 - fscore_keras: 0.6033
Epoch 9/10
56/56 [=====] - 786s 14s/step - loss: 0.2909 - maeOverFscore_keras: 2.6354 - fscore_keras: 0.6319
Epoch 10/10
56/56 [=====] - 706s 13s/step - loss: 0.2886 - maeOverFscore_keras: 2.5616 - fscore_keras: 0.6427
```

조건 3에서 epoch수를 증가시켰다.

이제까지 조건 3이 가장 높은 점수를 받았기 때문에 조건 3의 모델로 더 많이 훈련시켜보았다.  
그 결과 성능이 향상되었다.

점수 결과 : 2.4249356135

## 1-8) 조건 7

- 0~8 채널만 사용
- 60개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 10
- kernel\_initializer = 'he normal'

조건 6에서 정규화 조건을 추가했으나 오히려 성능이 더 떨어졌다.

```
def build_model(input_layer, start_neurons):  
  
    # 40 x 40 -> 20 x 20  
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(input_layer)  
    conv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(conv1)  
    pool1 = BatchNormalization()(conv1)  
    pool1 = MaxPooling2D((2, 2))(pool1)  
    pool1 = Dropout(0.30)(pool1) ##### 0.25 -> 0.3으로 바꿈  
  
    # 20 x 20 -> 10 x 10  
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(pool1)  
    conv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(conv2)  
    pool2 = BatchNormalization()(conv2)  
    pool2 = MaxPooling2D((2, 2))(pool2)  
    pool2 = Dropout(0.30)(pool2) ##### 0.25 -> 0.3으로 바꿈  
  
    # 10 x 10  
    convm = Conv2D(start_neurons * 4, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(pool2)  
  
    # 10 x 10 -> 20 x 20  
    deconv2 = Conv2DTranspose(start_neurons * 2, (3, 3), strides=(2, 2), padding="same")(convm)  
    uconv2 = concatenate([deconv2, conv2])  
    uconv2 = Dropout(0.30)(uconv2) ##### 0.25 -> 0.3으로 바꿈  
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(uconv2)  
    uconv2 = Conv2D(start_neurons * 2, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(uconv2)  
    uconv2 = BatchNormalization()(uconv2)  
  
    # 20 x 20 -> 40 x 40  
    deconv1 = Conv2DTranspose(start_neurons * 1, (3, 3), strides=(2, 2), padding="same")(uconv2)  
    uconv1 = concatenate([deconv1, conv1])  
    uconv1 = Dropout(0.30)(uconv1) ##### 0.25 -> 0.3으로 바꿈  
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(uconv1)  
    uconv1 = Conv2D(start_neurons * 1, (3, 3), activation="relu", padding="same", kernel_initializer = 'he_normal')(uconv1)  
    uconv1 = BatchNormalization()(uconv1)  
    uconv1 = Dropout(0.30)(uconv1) ##### 0.25 -> 0.3으로 바꿈  
    output_layer = Conv2D(1, (1,1), padding="same", activation="relu", kernel_initializer = 'he_normal')(uconv1)  
  
    return output_layer
```

```
model_history = model.fit(train_dataset, epochs = 10, verbose=1)
```

```
Epoch 1/10
56/56 [=====] - 687s 12s/step - loss: 0.4914 - maeOverFscore_keras: 5.4576 - fscore_keras:
0.3872
Epoch 2/10
56/56 [=====] - 708s 13s/step - loss: 0.3434 - maeOverFscore_keras: 3.6472 - fscore_keras:
0.5103
Epoch 3/10
56/56 [=====] - 732s 13s/step - loss: 0.3226 - maeOverFscore_keras: 3.3336 - fscore_keras:
0.5490
Epoch 4/10
56/56 [=====] - 774s 14s/step - loss: 0.3119 - maeOverFscore_keras: 3.0976 - fscore_keras:
0.5753
Epoch 5/10
56/56 [=====] - 753s 13s/step - loss: 0.3049 - maeOverFscore_keras: 2.9242 - fscore_keras:
0.5979
Epoch 6/10
56/56 [=====] - 705s 13s/step - loss: 0.3016 - maeOverFscore_keras: 2.8404 - fscore_keras:
0.6075
Epoch 7/10
56/56 [=====] - 756s 13s/step - loss: 0.2970 - maeOverFscore_keras: 2.7351 - fscore_keras:
0.6223
Epoch 8/10
56/56 [=====] - 687s 12s/step - loss: 0.2920 - maeOverFscore_keras: 2.6344 - fscore_keras:
0.6344
Epoch 9/10
56/56 [=====] - 676s 12s/step - loss: 0.2918 - maeOverFscore_keras: 2.6897 - fscore_keras:
0.6221
Epoch 10/10
56/56 [=====] - 748s 13s/step - loss: 0.2895 - maeOverFscore_keras: 2.5994 - fscore_keras:
0.6356
```

점수 결과 : 2.6026103043

## 1-9) 조건 8

- 0~8 채널만 사용
- 50개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 27

epoch 를 50으로 설정했을 때 epoch=28부터 loss 값이 커지는 것을 확인하였다.  
그래서 epoch를 27로 제한하여 모델링을 해 보았다.  
가장 높은 성능이 측정되었다.

점수 결과 : 2.1903889766

## 1-10) 조건 9

- 0~8 채널만 사용
- 60개 이상의 픽셀에 강수량이 기록되어 있는 이미지만 사용
- Dropout(0.3)
- epochs = 27

```
56/56 [=====] - 682s 12s/step - loss: 0.2804 - mae0verFscore_keras: 2.3999 - fscore_keras: 0.6659
Epoch 17/27
56/56 [=====] - 679s 12s/step - loss: 0.2922 - mae0verFscore_keras: 2.6827 - fscore_keras: 0.6245
Epoch 18/27
56/56 [=====] - 665s 12s/step - loss: 0.2824 - mae0verFscore_keras: 2.4736 - fscore_keras: 0.6522
Epoch 19/27
56/56 [=====] - 649s 12s/step - loss: 0.2797 - mae0verFscore_keras: 2.4020 - fscore_keras: 0.6645
Epoch 20/27
56/56 [=====] - 655s 12s/step - loss: 0.2781 - mae0verFscore_keras: 2.3654 - fscore_keras: 0.6710
Epoch 21/27
56/56 [=====] - 662s 12s/step - loss: 0.2770 - mae0verFscore_keras: 2.3243 - fscore_keras: 0.6785
Epoch 22/27
56/56 [=====] - 654s 12s/step - loss: 0.2745 - mae0verFscore_keras: 2.2918 - fscore_keras: 0.6817
Epoch 23/27
56/56 [=====] - 641s 11s/step - loss: 0.2765 - mae0verFscore_keras: 2.3485 - fscore_keras: 0.6709
Epoch 24/27
56/56 [=====] - 619s 11s/step - loss: 0.2749 - mae0verFscore_keras: 2.3041 - fscore_keras: 0.6792
Epoch 25/27
56/56 [=====] - 628s 11s/step - loss: 0.2745 - mae0verFscore_keras: 2.3219 - fscore_keras: 0.6754
Epoch 26/27
56/56 [=====] - 682s 12s/step - loss: 0.2757 - mae0verFscore_keras: 2.3051 - fscore_keras: 0.6799
Epoch 27/27
56/56 [=====] - 674s 12s/step - loss: 0.2729 - mae0verFscore_keras: 2.2627 - fscore_keras: 0.6862
```

조건 8에서 픽셀 합 조건만 바꿔보았다.  
오히려 조건 8보다 성능이 떨어졌다.

점수 결과 : 2.909901536

## 2. 점수 결과 캡처 화면



대회   교육   코드공유   More


Search

nn   nnn\_   


위성관측 데이터 활용 강수량 산출 AI 경진대회

AiFrenz 시즌2

AI프렌즈 2 | 기상 | 한국원자력연구원 | 위성 이미지 빅데이터와 인공지능 AI 로 강수량 산출 | MAE, F1score

 상금 : 총 250만원

 2020.04.01 ~ 2020.05.25 17:59

 701팀    오늘 마감



 참여중

대회안내   데이터   코드 공유   토론   리더보드   팀   제출

가채점 순위

순위기준

● WINNER

● 1%

● 4%

● 10%

#	팀	팀 멤버	점수	제출수	등록일
113	난영		2.19038	11	5시간 전

### 3. 결과 및 고찰

이번 과제는 이전과는 다르게 다루는 데이터의 용량이 컸던 게 가장 힘들었던 점이었다.

이 전의 wine 분류 과제처럼 구글 코랩을 사용하였으나 용량 때문에 실행 도중에 지속적으로 중단 되는 문제가 발생하여 jupyter notebook 을 사용하였다.

또한 한 번 실행시키는데에도 상당한 시간이 소요되어 즉각적으로 결과를 확인하지 못하는 점도 이전과는 달라 힘들었다.

Baseline의 코드를 응용하여 Baseline에 비해 모델 성능을 향상시키긴 했으나 한계가 있음을 확인했다. 그래서 U-Net이 아닌 ResNet 기반의 모델을 사용하려고 하였으나 이 모델은 U-Net 보다 더 많은 시간이 소요되었고 시간이 부족하여 점수를 확인하지 못하였다. 충분한 시간이 있었다면 더 좋은 성능을 가진 모델을 만들 수 있었을 것 같아서 아쉬웠다.

```
k = 5
models = []

train_model(x_train, y_train, k=k, s=4)

for n in range(k):
    model = load_model('models/model'+str(n)+'.h5', custom_objects = {'score':score, 'fscore_keras':fscore_keras})
    models.append(model)
```

Train on 196144 samples, validate on 6129 samples

Epoch 1/10

10368/196144 [>.....] - ETA: 29:12:43 - loss: 0.3809 - score: 3.7169 - fscore\_keras: 0.549

9

- - - -