[RT704] Advanced Medical Image Processing

< Assignment 2 >

(2021-11-20)

Robotics Engineering

202123008 Jinmin Kim

Phone: 010-6266-6099

Mail: rlawlsals@dgist.ac.kr

Problem #1

- 파일 => “pro1.py” 참조

1) 라이브러리 임포트

```
1   import numpy as np
2   import os
3   import SimpleITK as sitk
4   import pandas as pd
5   from tensorflow import keras
6   from tensorflow.keras.models import Sequential
7   from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
8   from tensorflow.keras.layers import Conv2D, MaxPooling2D
9   from  tensorflow.keras.models import load_model
10  from matplotlib import pyplot as plt
11
```

1~11 : 필요 라이브러리를 불러온다.

2) 데이터 셋업

```
12  # ##############################################################################################
13  # 데이터 셋업
14
15  # Global values 및 ndarray 생성
16  n_train = 222
17  n_val = 29
18  n_test = 84
19  ch = 4
20  H = 240
21  W = 240
22  train_img = np.empty((n_train, ch, H, W))
23  train_sol = np.empty((n_train, 1))
24  val_img = np.empty((n_val, ch, H, W))
25  val_sol = np.empty((n_val, 1))
26  test_img = np.empty((n_test, ch, H, W))
27  test_sol = np.empty((n_test, 1))
28
29  Data_path = os.getcwd()
30
```

12~30 : Global values 및 ndarray를 생성한다.

```
31  # train 데이터 리스트 생성
32  walking_train = os.path.join(Data_path, 'clf_w_mask', 'train')
33  i = 0
34  for path, dirs, files in os.walk(walking_train):
35      if 'img.npy' in files:
36          path_img = os.path.join(path, 'img.npy')
37          path_label = os.path.join(path, 'label.npy')
38          img = np.load(path_img)
39          label = np.load(path_label)
40          train_img[i] = img
41          train_sol[i] = label
42          i += 1
43
44  # validation 데이터 리스트 생성
45  walking_val = os.path.join(Data_path,'clf_w_mask', 'valid')
46  i = 0
47  for path, dirs, files in os.walk(walking_val):
48      if 'img.npy' in files:
49          path_img = os.path.join(path, 'img.npy')
50          path_label = os.path.join(path, 'label.npy')
51          img = np.load(path_img)
52          label = np.load(path_label)
53          val_img[i] = img
54          val_sol[i] = label
55          i += 1
56
57  # test 데이터 리스트 생성
58  walking_test = os.path.join(Data_path, 'clf_w_mask', 'test')
59  i = 0
60  for path, dirs, files in os.walk(walking_test):
61      if 'img.npy' in files:
62          path_img = os.path.join(path, 'img.npy')
63          path_label = os.path.join(path, 'label.npy')
64          img = np.load(path_img)
65          label = np.load(path_label)
66          test_img[i] = img
67          test_sol[i] = label
68          i += 1
69
```

31~69 : train, validation, test 데이터를 ndarray에 저장한다. 파일 폴더를 os.walk를 통해 스캔하여 저장하면, ndarray에 폴더 개수만큼 데이터가 저장된다.
각각의 ndarray는 ((데이터 수, 채널 수, 이미지 높이, 이미지 폭))로 구성된다.

```
70  # img의 Shape를 조정 (n, 4, 240, 240) -> (n, 240, 240, 4)
71  train_img = np.transpose(train_img, (0, 2, 3, 1))
72  val_img = np.transpose(val_img, (0, 2, 3, 1))
73  test_img = np.transpose(test_img, (0, 2, 3, 1))
74
75  # img의 intensity를 0~1사이 범위로 mapping
76  print('Before train(min, max) :', np.min(train_img), np.max(train_img))
77  print('Before val(min, max) :', np.min(val_img), np.max(val_img))
78  print('Before test(min, max) :', np.min(test_img), np.max(test_img))
79  train_img = ( train_img - np.min(train_img) ) / (np.max(train_img) - np.min(train_img))
80  val_img = ( val_img - np.min(val_img) ) / (np.max(val_img) - np.min(val_img))
81  test_img = ( test_img - np.min(test_img) ) / (np.max(test_img) - np.min(test_img))
82  print('After train(min, max) :', np.min(train_img), np.max(train_img))
83  print('After val(min, max) :', np.min(val_img), np.max(val_img))
84  print('After test(min, max) :', np.min(test_img), np.max(test_img))
85
```

70~85 : keras에서 학습시킬 수 있도록 ndarray의 shape를 조정한다.
그리고 각 이미지의 intensity를 0~1사이로 mapping한다.

```
Before train(min, max) : -0.5122154030022901 16.408137753418224
Before val(min, max) : -0.4555891359986302 14.91161505002145
Before test(min, max) : -0.4796270691538732 20.10646886577503
After train(min, max) : 0.0 1.0
After val(min, max) : 0.0 1.0
After test(min, max) : 0.0 1.0
```

출력해서 확인해보면 정상적으로 mapping되었음을 알 수 있다.

3) CNN 모델 생성

```
86  # ##############################################################################
87  # CNN 모델 생성
88
89  # train model
90  model = Sequential()
91
92  # Convolution Layer 16
93  model.add(Conv2D(16, (3, 3), padding='same', input_shape=train_img.shape[1:]))
94  model.add(Activation('relu'))
95  # Pooling Layer
96  model.add(MaxPooling2D(pool_size=(2, 2)))
97  model.add(Dropout(0.25))
98
99  # Convolution Layer 32
100 model.add(Conv2D(32, (3, 3), padding='same'))
101 model.add(Activation('relu'))
102 # Pooling Layer
103 model.add(MaxPooling2D(pool_size=(2, 2)))
104 model.add(Dropout(0.25))
105
106 # Fully Connected Layer 512
107 model.add(Flatten())
108 model.add(Dense(512))
109 model.add(Activation('relu'))
110 model.add(Dropout(0.25))
111
112 # OUTPUT
113 model.add(Dense(1))
114 model.add(Activation('sigmoid'))
115
116 # model compile
117 model.compile(loss='binary_crossentropy',
118               optimizer='adam',
119               metrics=['acc'])
120
121 # 학습 모델 출력
122 model.summary()
123
```

86~123 : CNN 모델을 생성한다.

input shape는 (240, 240, 4)로 지정해주었다.

Convolution Layer를 2번 사용하였고, 각각 필터 개수를 16개, 32개를 사용하였다.

Pooling Layer는 CL이 끝날 때마다 (2, 2)의 Max pooling으로 넣어주었다.

Fully Connected Layer는 ReLU function을 이용하여 구현하였다.

마지막으로 OUTPUT의 경우는 sigmoid function을 사용하여 출력하였다.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 240, 240, 16)      592

activation (Activation)      (None, 240, 240, 16)      0

max_pooling2d (MaxPooling2D  (None, 120, 120, 16)      0
)

dropout (Dropout)            (None, 120, 120, 16)      0

conv2d_1 (Conv2D)            (None, 120, 120, 32)      4640

activation_1 (Activation)    (None, 120, 120, 32)      0

max_pooling2d_1 (MaxPooling  (None, 60, 60, 32)        0
2D)

dropout_1 (Dropout)          (None, 60, 60, 32)        0

flatten (Flatten)            (None, 115200)            0

dense (Dense)                (None, 512)               58982912

activation_2 (Activation)    (None, 512)               0

dropout_2 (Dropout)          (None, 512)               0

dense_1 (Dense)              (None, 1)                 513

activation_3 (Activation)    (None, 1)                 0

=================================================================
Total params: 58,988,657
Trainable params: 58,988,657
Non-trainable params: 0
_________________________________________________________________
```

<학습 모델 출력 결과>

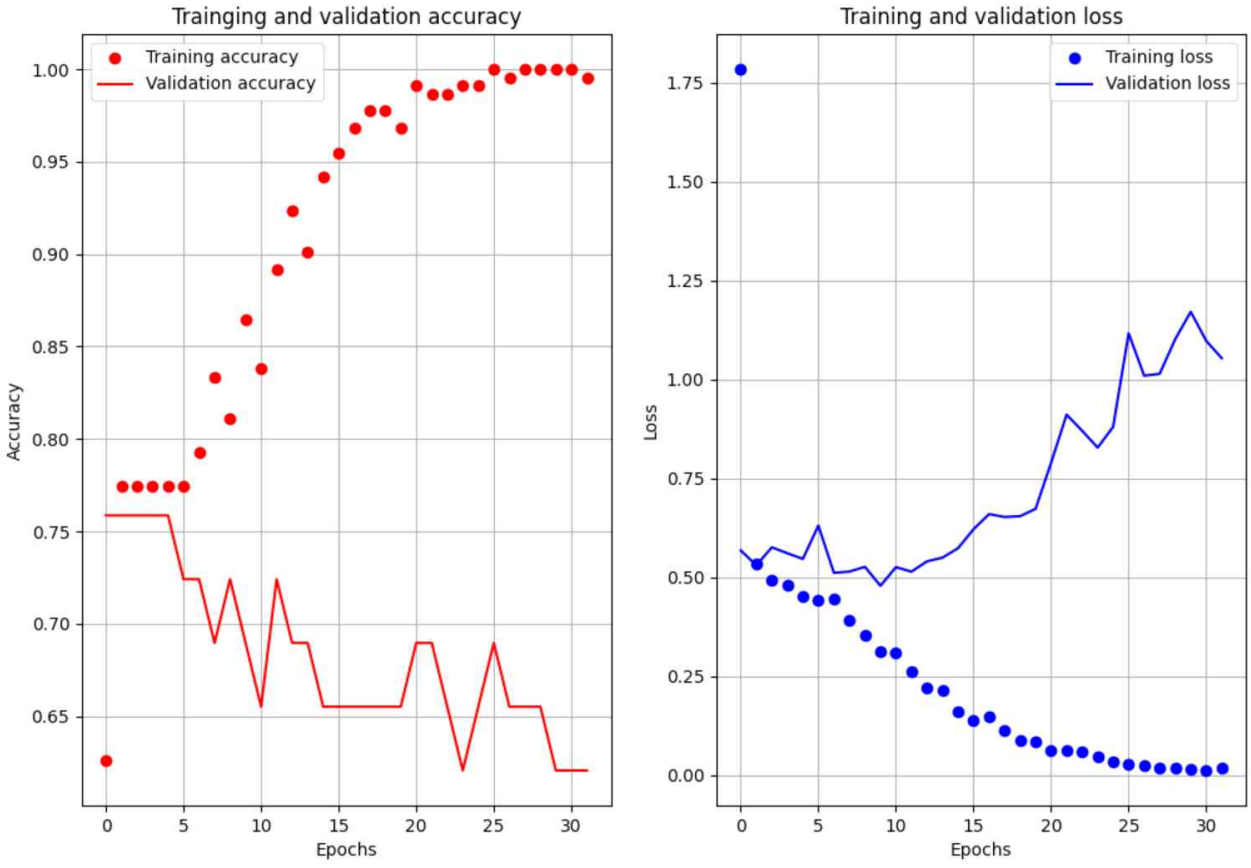
4) train, val 데이터셋을 모델에 fitting

```
124  # ##############################################################################
125  # train, val을 모델에 fitting
126
127  epochs = 32
128  hist = model.fit(train_img, train_sol,
129                   epochs=epochs,
130                   shuffle=True,
131                   validation_data=(val_img, val_sol),
132                   verbose=1)
133
134  # 모델 저장
135  learning_model_path = os.path.join(Data_path, 'HGGLGG_trained_model.h5')
136  model.save(learning_model_path)
137  print('Saved trained model at %s ' % learning_model_path)
138
139  # train accuray, loss 출력
140  train_acc = hist.history['acc'][-1]
141  train_loss =  hist.history['loss'][-1]
142  print('train accuracy: ', train_acc)
143  print('train loss: ', train_loss)
144
145
146  # train, val의 accuracy, loss를 그래프로 표현
147  plot_target = ['loss', 'val_loss', 'acc', 'val_acc']
148  plt.figure(figsize = (12,8))
149
150  plt.subplot(121)
151  plt.plot(hist.history['acc'], 'ro', label='Training accuracy')
152  plt.plot(hist.history['val_acc'], 'r', label='Validation accuracy')
153  plt.title('Trainging and validation accuracy')
154  plt.xlabel('Epochs')
155  plt.ylabel('Accuracy')
156  plt.legend()
157  plt.grid()
158
159  plt.subplot(122)
160  plt.plot(hist.history['loss'], 'bo', label='Training loss')
161  plt.plot(hist.history['val_loss'], 'b', label='Validation loss')
162  plt.title('Training and validation loss')
163  plt.xlabel('Epochs')
164  plt.ylabel('Loss')
165  plt.legend()
166  plt.grid()
167
168  plt.show()
169
```

124~169 : train, val 데이터셋을 이전에 생성한 CNN 모델에 fitting 시킨다.
학습 모델은 데이터 경로에 'HGGLGG_trained_model.h5'로 저장한다.

```
Epoch 1/32
2021-11-15 13:54:17.729508: I tensorflow/stream_executor/cuda/cuda_dnn.cc:366] Loaded cuDNN version 8
101
7/7 [==============================] - 3s 110ms/step - loss: 1.7833 - acc: 0.6261 - val_loss: 0.5673
- val_acc: 0.7586
Epoch 2/32
7/7 [==============================] - 0s 45ms/step - loss: 0.5320 - acc: 0.7748 - val_loss: 0.5296 -
 val_acc: 0.7586
Epoch 3/32
7/7 [==============================] - 0s 44ms/step - loss: 0.4915 - acc: 0.7748 - val_loss: 0.5751 -
 val_acc: 0.7586
Epoch 4/32
7/7 [==============================] - 0s 45ms/step - loss: 0.4810 - acc: 0.7748 - val_loss: 0.5602 -
 val_acc: 0.7586
Epoch 5/32
7/7 [==============================] - 0s 44ms/step - loss: 0.4515 - acc: 0.7748 - val_loss: 0.5458 -
 val_acc: 0.7586
Epoch 6/32
7/7 [==============================] - 0s 44ms/step - loss: 0.4422 - acc: 0.7748 - val_loss: 0.6301 -
 val_acc: 0.7241
Epoch 7/32
7/7 [==============================] - 0s 45ms/step - loss: 0.4444 - acc: 0.7928 - val_loss: 0.5107 -
 val_acc: 0.7241
Epoch 8/32
7/7 [==============================] - 0s 45ms/step - loss: 0.3909 - acc: 0.8333 - val_loss: 0.5138 -
 val_acc: 0.6897
Epoch 9/32
7/7 [==============================] - 0s 45ms/step - loss: 0.3539 - acc: 0.8108 - val_loss: 0.5258 -
 val_acc: 0.7241
```

Epoch = 32로 학습을 진행 중인 모습



Accuracy와 Loss 그래프

=> Training accuracy는 점점 증가하고, loss는 점점 감소하는 모습을 볼 수 있다.

```
train accuracy:  0.9954954981803894
train loss:  0.018662244081497192
```

=> 학습 완료 후 training accuracy는 거의 1에 가깝고, loss는 0에 가까운 결과를 얻음.

5) Test 모델로 성능 테스트

```
170  # ##############################################################################
171  # Test 모델로 성능 테스트
172
173  # predict 함수 정의
174  def predict_HGGLGG(x, model):
175      x_data =(np.expand_dims(x, 0))
176      predict = model.predict(x_data)
177      #print(predict)
178      if predict < 0.5:
179          return 'LGG'
180      else:
181          return 'HGG'
182
183  # test 모델 84개에 대한 학습된 CNN의 예상 평가 결과
184  model = load_model(learning_model_path)
185  for p in range(n_test):
186      test_data = test_img[p]
187      result = predict_HGGLGG(test_data, model)
188      print(p+1, '번째 이미지는', result, '입니다.')
189
```

170~189 : test 모델을 사용하여 학습 모델의 성능을 테스트.

```
1 번째 이미지는 LGG 입니다.
2 번째 이미지는 LGG 입니다.
3 번째 이미지는 HGG 입니다.
4 번째 이미지는 LGG 입니다.
5 번째 이미지는 LGG 입니다.
6 번째 이미지는 HGG 입니다.
7 번째 이미지는 LGG 입니다.
8 번째 이미지는 LGG 입니다.
9 번째 이미지는 LGG 입니다.
10 번째 이미지는 HGG 입니다.
11 번째 이미지는 HGG 입니다.
12 번째 이미지는 HGG 입니다.
13 번째 이미지는 HGG 입니다.
14 번째 이미지는 HGG 입니다.
15 번째 이미지는 HGG 입니다.
16 번째 이미지는 HGG 입니다.
17 번째 이미지는 HGG 입니다.
18 번째 이미지는 HGG 입니다.
19 번째 이미지는 HGG 입니다.
20 번째 이미지는 HGG 입니다.
21 번째 이미지는 HGG 입니다.
22 번째 이미지는 HGG 입니다.
```

=> test 데이터 84개에 대한 예측 결과를 출력하였음.

Problem #2

- 파일 => “pro2.py” 참조

=> pyradiomics를 이용하여 nrrd를 읽어왔는데, feature extraction하는 부분을 이해하지 못해 구현에 실패하였습니다.