[RT704] Advanced Medical Image Processing

< Assignment 3 >

(2021-11-27)

Robotics Engineering

202123008 Jinmin Kim

Phone: 010-6266-6099

Mail: rlawlsals@dgist.ac.kr

Problem #1

- 파일 => “pro1.py” 참조

1) 코드 설명

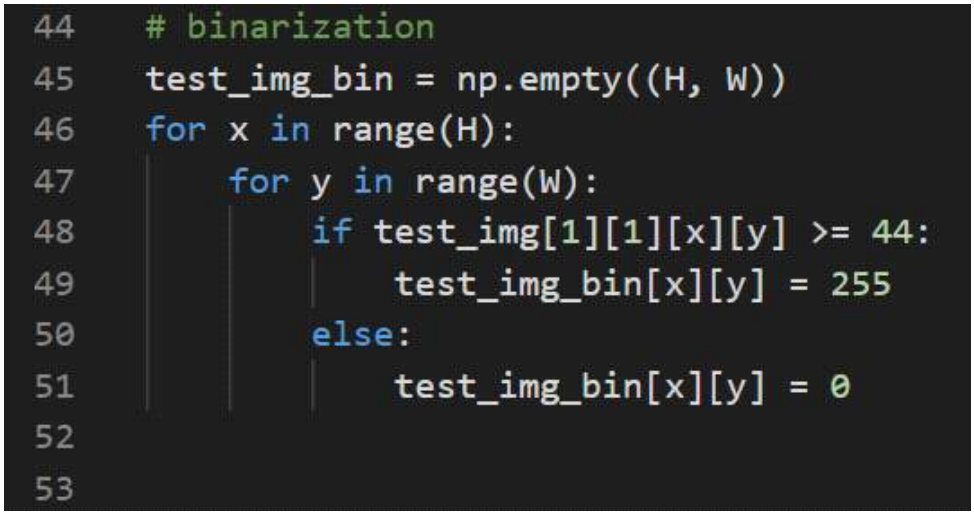
```python
import numpy as np
import os
from matplotlib import pyplot as plt

```

1~4 : 필요 라이브러리를 불러온다.

```python
# ##############################################################################
# 데이터 셋업

# Global values 및 ndarray 생성
n_test = 84
ch = 4
H = 240
W = 240
test_img = np.empty((n_test, ch, H, W))
test_sol = np.empty((n_test, 1))
test_seg = np.empty((n_test, H, W))

Data_path = os.getcwd()

# test 데이터 리스트 생성
walking_test = os.path.join(Data_path, 'clf_w_mask', 'test')
i = 0
for path, dirs, files in os.walk(walking_test):
    if 'img.npy' in files:
        path_img = os.path.join(path, 'img.npy')
        path_label = os.path.join(path, 'label.npy')
        path_seg = os.path.join(path, 'seg.npy')
        img = np.load(path_img)
        label = np.load(path_label)
        seg = np.load(path_seg)
        test_img[i] = img
        test_sol[i] = label
        test_seg[i] = seg
        i += 1

# img의 intensity를 0~255사이 범위로 Normalization
print('Before test(min, max) :', np.min(test_img), np.max(test_img))
print('Before testseg(min, max) :', np.min(test_seg), np.max(test_seg))
test_img = ( test_img - np.min(test_img) ) / (np.max(test_img) - np.min(test_img)) * 255
test_seg = ( test_seg - np.min(test_seg) ) / (np.max(test_seg) - np.min(test_seg)) * 255
print('After test(min, max) :', np.min(test_img), np.max(test_img))
print('After testseg(min, max) :', np.min(test_seg), np.max(test_seg))


```

5~43 : Global values 및 ndarray를 생성하고, 데이터를 정규화(0~255)

- test_img[0][1][x][y] : 1번째 test image set, 2채널에서 x, y 좌표의 밝기를 나타냄.
- seg[0][x][y] : 1번째 seg image에서 x, y좌표의 밝기를 나타냄.



```
44    # binarization
45    test_img_bin = np.empty((H, W))
46    for x in range(H):
47        for y in range(W):
48            if test_img[1][1][x][y] >= 44:
49                test_img_bin[x][y] = 255
50            else:
51                test_img_bin[x][y] = 0
52
53
```

44~53 : Thresholding 방법을 사용하여 데이터를 이진화. Threshold는 목표로 하는 부분의 intensity값으로 정하였다.

```
54  # Region Growing 함수 구현
55  class Point(object):
56      def __init__(self,x,y):
57          self.x = x
58          self.y = y
59
60      def getX(self):
61          return self.x
62      def getY(self):
63          return self.y
64
65      def getGrayDiff(img,currentPoint,tmpPoint):
66          return abs(int(img[currentPoint.x,currentPoint.y]) - int(img[tmpPoint.x,tmpPoint.y]))
67
68      def selectConnects(p):
69          if p != 0:
70              connects = [Point(-1, -1), Point(0, -1), Point(1, -1), Point(1, 0), Point(1, 1), \
71                  Point(0, 1), Point(-1, 1), Point(-1, 0)]
72          else:
73              connects = [ Point(0, -1), Point(1, 0),Point(0, 1), Point(-1, 0)]
74          return connects
75
76      def regionGrow(img,seeds,thresh,p = 1):
77          height, weight = img.shape
78          seedMark = np.zeros(img.shape)
79          seedList = []
80          for seed in seeds:
81              seedList.append(seed)
82          label = 255
83          connects = Point.selectConnects(p)
84          while(len(seedList)>0):
85              currentPoint = seedList.pop(0)
86
87              seedMark[currentPoint.x,currentPoint.y] = label
88              for i in range(8):
89                  tmpX = currentPoint.x + connects[i].x
90                  tmpY = currentPoint.y + connects[i].y
91                  if tmpX < 0 or tmpY < 0 or tmpX >= height or tmpY >= weight:
92                      continue
93                  grayDiff = Point.getGrayDiff(img,currentPoint,Point(tmpX,tmpY))
94
95                  if grayDiff < thresh and seedMark[tmpX,tmpY] == 0:
96                      seedMark[tmpX,tmpY] = label
97                      seedList.append(Point(tmpX,tmpY))
98          return seedMark
99
```

54~99 : Region growing 클래스 함수 구현

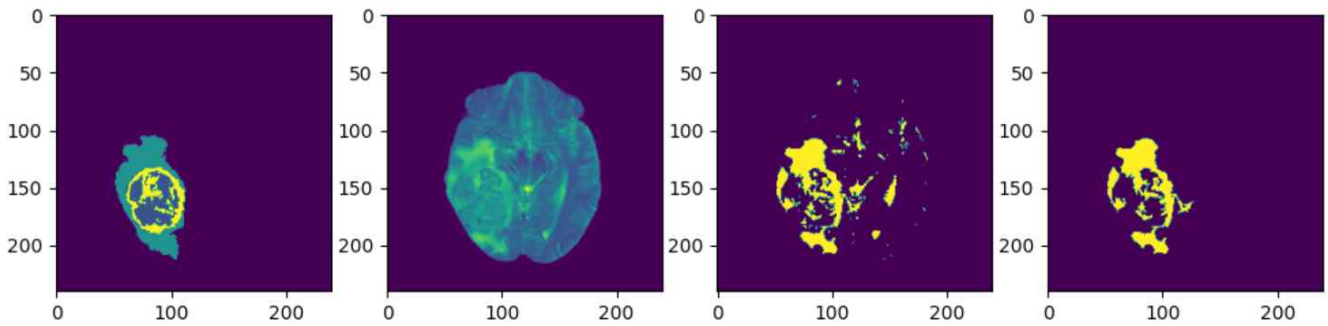
– 함수 설명 : seed point를 정의하고 그 point에서 각각 상하좌우로 스캔하도록 한다.

```
100  # region growing 구현
101  img1 = test_img_bin
102  seeds = [Point(124,80), Point(141,57), Point(169, 83)]
103  binaryImg = Point.regionGrow(img1,seeds,44)
104
105  # Region Growing할 이미지와 segmentation 정답을 열어 확인
106  plt.figure(figsize = (12,6))
107  plt.subplot(141)
108  plt.imshow(test_seg[1])
109  plt.subplot(142)
110  plt.imshow(test_img[1][1])
111  plt.subplot(143)
112  plt.imshow(test_img_bin)
113  plt.subplot(144)
114  plt.imshow(binaryImg)
115  plt.show()
```

100~115 : seed 포인트를 3개 정의하고 Threshold를 이진화 할때와 동일하게 44로 설정하였다. 그리고 이미지를 플롯팅하였다.
plot1 : 정답 이미지, plot2 : 원본 이미지, plot3 : Thresholding만 진행한 이미지
plot4 : Region growing을 진행한 이미지

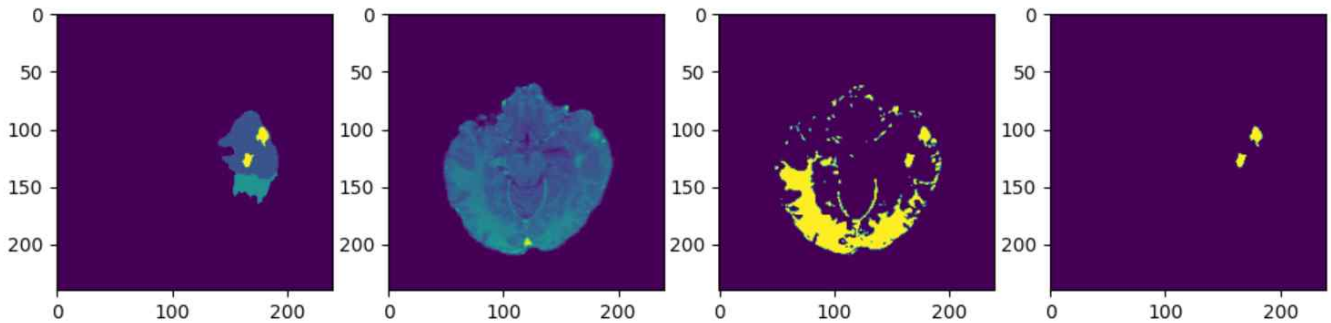
2) 결과

- Good case



- 설정 : test img 2번째 모델, 2번째 채널, thres = 44
- 결과 : 분리하고자 하는 부분이 적당한 밝기로 분포되어 있기때문에 좋은 결과를 보임.

- Bad case



- 설정 : test img 6번째 모델, 3번째 채널, thres = 35
- 결과 : 분리하고자 하는 부분이 균일하지 못한 밝기를 가지고 있어서 밝기값이 큰 부분만 도출되었음.

Problem #2

- 파일 => “pro2.py” 참조

1) 코드 설명

```
1   import numpy as np
2   import os
3   import cv2
4   from matplotlib import pyplot as plt
5   from collections import defaultdict
6
7   # ##############################################################################
8   # 데이터 셋업
9
10  # Global values 및 ndarray 생성
11  n_test = 84
12  ch = 4
13  H = 240
14  W = 240
15  test_img = np.empty((n_test, ch, H, W))
16  test_sol = np.empty((n_test, 1))
17  test_seg = np.empty((n_test, H, W))
18
19  Data_path = os.getcwd()
20
21  # test 데이터 리스트 생성
22  walking_test = os.path.join(Data_path, 'clf_w_mask', 'test')
23  i = 0
24  for path, dirs, files in os.walk(walking_test):
25      if 'img.npy' in files:
26          path_img = os.path.join(path, 'img.npy')
27          path_label = os.path.join(path, 'label.npy')
28          path_seg = os.path.join(path, 'seg.npy')
29          img = np.load(path_img)
30          label = np.load(path_label)
31          seg = np.load(path_seg)
32          test_img[i] = img
33          test_sol[i] = label
34          test_seg[i] = seg
35          i += 1
36
37  # img의 intensity를 0~255사이 범위로 mapping
38  print('Before test(min, max) :', np.min(test_img), np.max(test_img))
39  print('Before testseg(min, max) :', np.min(test_seg), np.max(test_seg))
40  test_img = ( test_img - np.min(test_img) ) / (np.max(test_img) - np.min(test_img)) * 255
41  test_seg = ( test_seg - np.min(test_seg) ) / (np.max(test_seg) - np.min(test_seg)) * 255
42  print('After test(min, max) :', np.min(test_img), np.max(test_img))
43  print('After testseg(min, max) :', np.min(test_seg), np.max(test_seg))
44
```

1~44 : 라이브러리를 임포트하고 데이터를 가져옴.

```
45  plt.imsave('assign3_img.jpg', test_img[1][3])
46
47  img = cv2.imread('assign3_img.jpg')
48
49  img = cv2.resize(img,dsize=(500,500), interpolation=cv2.INTER_AREA)
50
51  img_draw = img.copy()
52  mask = np.zeros(img.shape[:2], dtype=np.uint8)  # 마스크 생성
53  rect = [0,0,0,0]    # 사각형 영역 좌표 초기화
54  mode = cv2.GC_EVAL  # 그랩컷 초기 모드
55  # 배경 및 전경 모델 버퍼
56  bgdmodel = np.zeros((1,65),np.float64)
57  fgdmodel = np.zeros((1,65),np.float64)
58
59  # 마우스 이벤트 처리 함수
60  def onMouse(event, x, y, flags, param):
61      global mouse_mode, rect, mask, mode
62      if event == cv2.EVENT_LBUTTONDOWN : # 왼쪽 마우스 누름
63          if flags <= 1: # 아무 키도 안 눌렀으면
64              mode = cv2.GC_INIT_WITH_RECT # 드래그 시작, 사각형 모드 ---①
65              rect[:2] = x, y # 시작 좌표 저장
66      # 마우스가 움직이고 왼쪽 버튼이 눌러진 상태
67      elif event == cv2.EVENT_MOUSEMOVE and flags & cv2.EVENT_FLAG_LBUTTON :
68          if mode == cv2.GC_INIT_WITH_RECT: # 드래그 진행 중 ---②
69              img_temp = img.copy()
70              # 드래그 사각형 화면에 표시
71              cv2.rectangle(img_temp, (rect[0], rect[1]), (x, y), (0,255,0), 2)
72              cv2.imshow('img', img_temp)
73          elif flags > 1: # 키가 눌러진 상태
74              mode = cv2.GC_INIT_WITH_MASK    # 마스크 모드 ---③
75              if flags & cv2.EVENT_FLAG_CTRLKEY : # ############## 컨트롤 키, 분명한 전경
76                  # 흰색 점 화면에 표시
77                  cv2.circle(img_draw,(x,y),3, (255,255,255),-1)
78                  # 마스크에 GC_FGD로 채우기      ---④
79                  cv2.circle(mask,(x,y),3, cv2.GC_FGD,-1)
80              if flags & cv2.EVENT_FLAG_SHIFTKEY : # ############## 쉬프트키, 분명한 배경
81                  # 검정색 점 화면에 표시
82                  cv2.circle(img_draw,(x,y),3, (0,0,0),-1)
83                  # 마스크에 GC_BGD로 채우기      ---⑤
84                  cv2.circle(mask,(x,y),3, cv2.GC_BGD,-1)
85              cv2.imshow('img', img_draw) # 그려진 모습 화면에 출력
86      elif event == cv2.EVENT_LBUTTONUP: # 마우스 왼쪽 버튼 뗀 상태 ---⑥
87          if mode == cv2.GC_INIT_WITH_RECT : # 사각형 그리기 종료
88              rect[2:] =x, y # 사각형 마지막 좌표 수집
89              # 사각형 그려서 화면에 출력 ---⑦
90              cv2.rectangle(img_draw, (rect[0], rect[1]), (x, y), (255,0,0), 2)
91              cv2.imshow('img', img_draw)
92          # 그랩컷 적용 ---⑧
93          cv2.grabCut(img, mask, tuple(rect), bgdmodel, fgdmodel, 1, mode)
94          img2 = img.copy()
95          # 마스크에 확실한 배경, 아마도 배경으로 표시된 영역을 0으로 채우기
96          img2[(mask==cv2.GC_BGD) | (mask==cv2.GC_PR_BGD)] = 0
97          cv2.imshow('grabcut', img2) # 최종 결과 출력
98          mode = cv2.GC_EVAL # 그랩컷 모드 리셋
99  # 초기 화면 출력 및 마우스 이벤트 등록
100
101
102 cv2.imshow('img', img)
103 cv2.setMouseCallback('img', onMouse)
104 while True:
105     if cv2.waitKey(0) & 0xFF == 27 : # esc
106         break
107 cv2.destroyAllWindows()
```

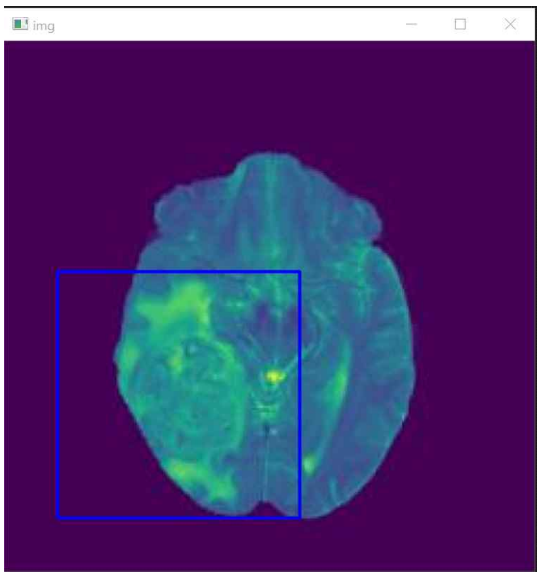
45~107 : Graph cut method 함수를 가져옴

- 마우스 좌클릭 : 분리할 부분의 대략적인 사각형 범위를 지정
- Ctrl키+마우스 좌클릭 : 분명한 전경
- Shift키+마우스 좌클릭 : 분명한 배경

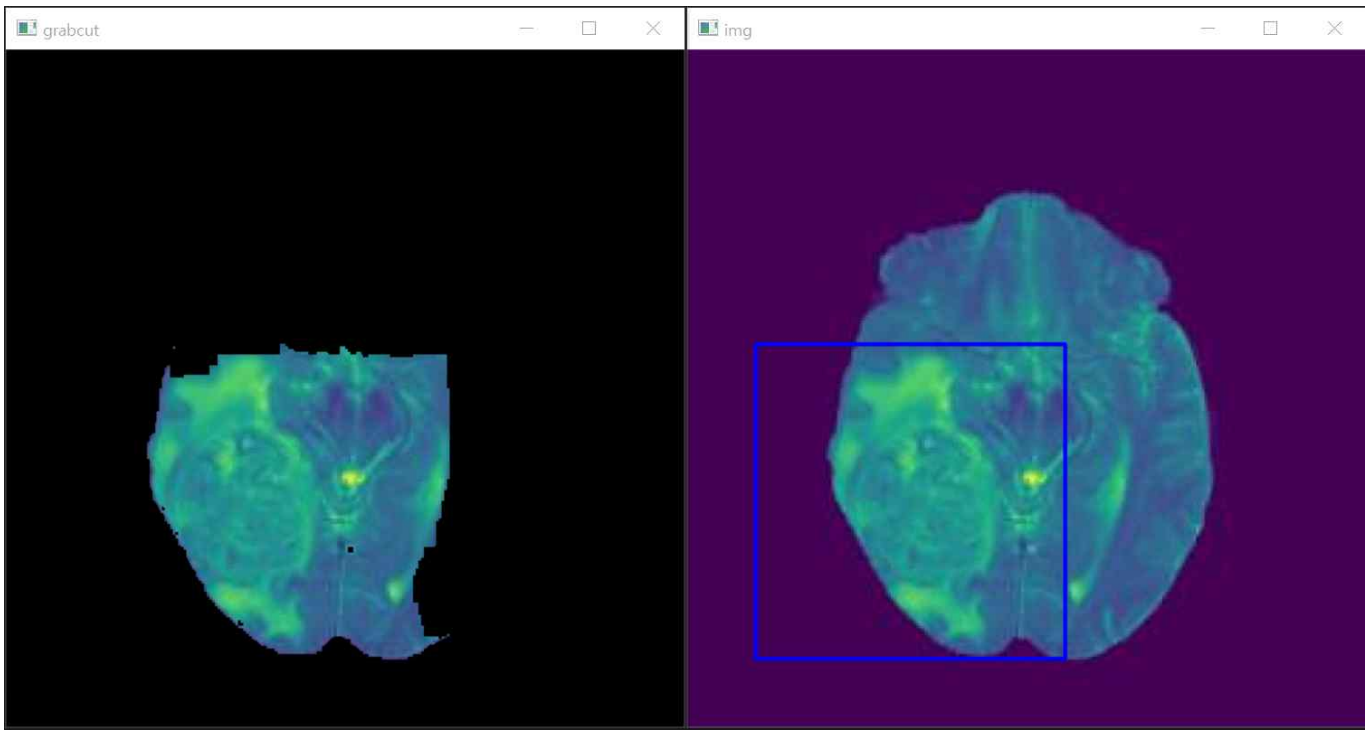
2) 결과

※ Problem1에서 사용했던 Test 이미지의 2번째 모델, 2번째 채널 이미지를 사용함.

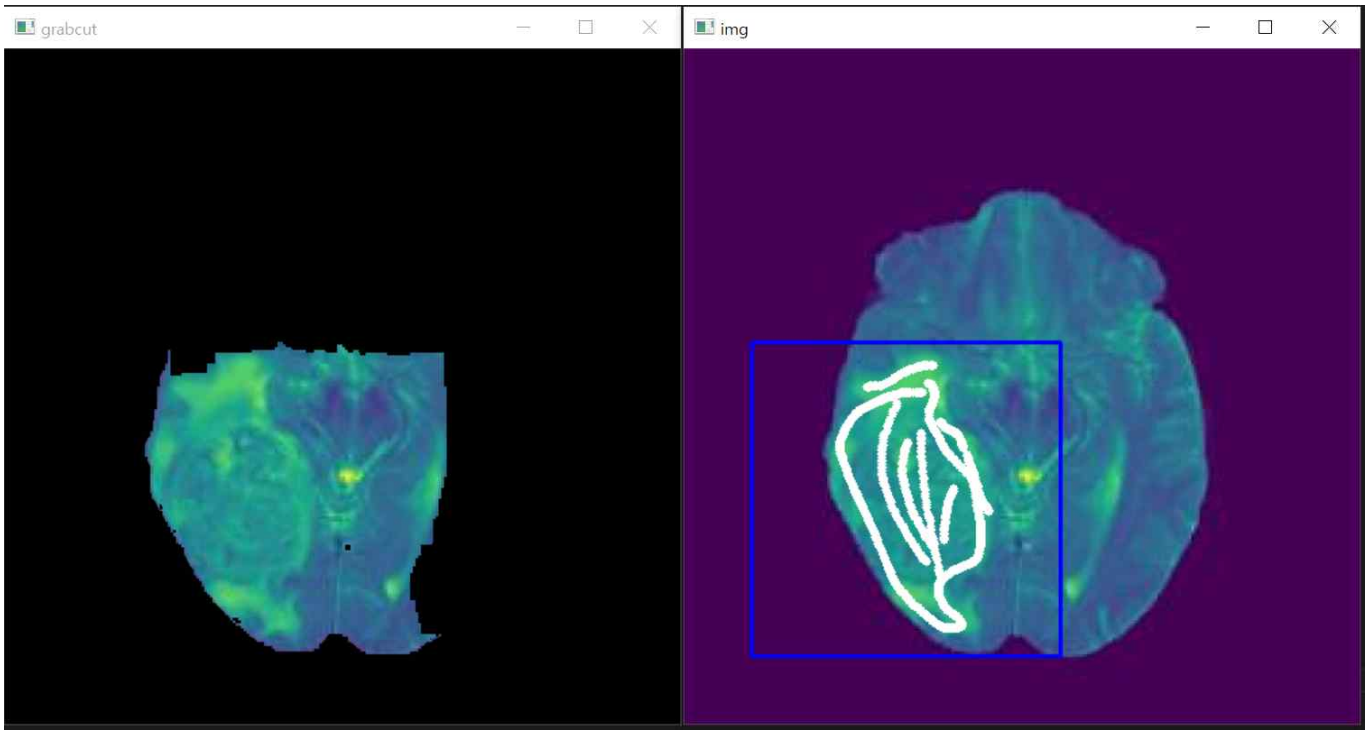
1. 오른쪽 이미지에서 마우스 좌클릭 드래그로 파란색 사각형 영역을 만든다.



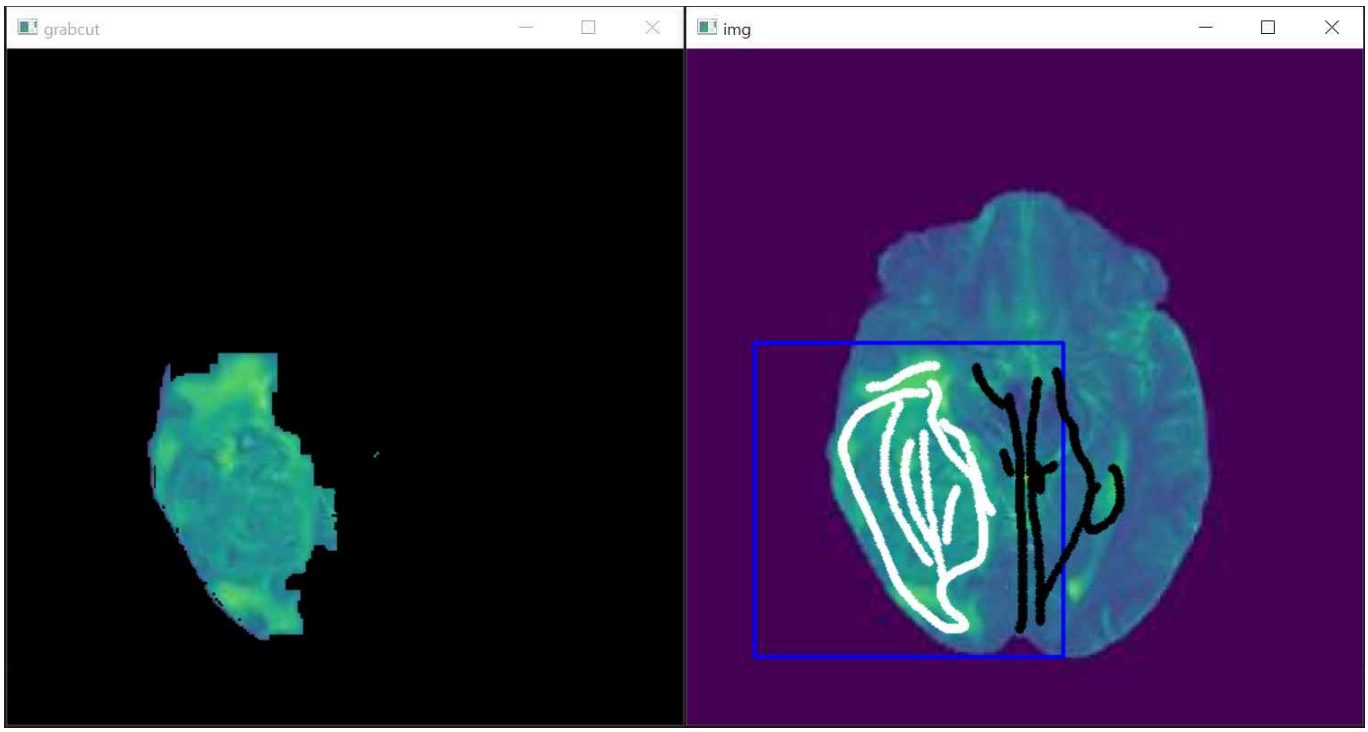
2. 좌측에 대략적으로 분리된 이미지가 띄워진다.



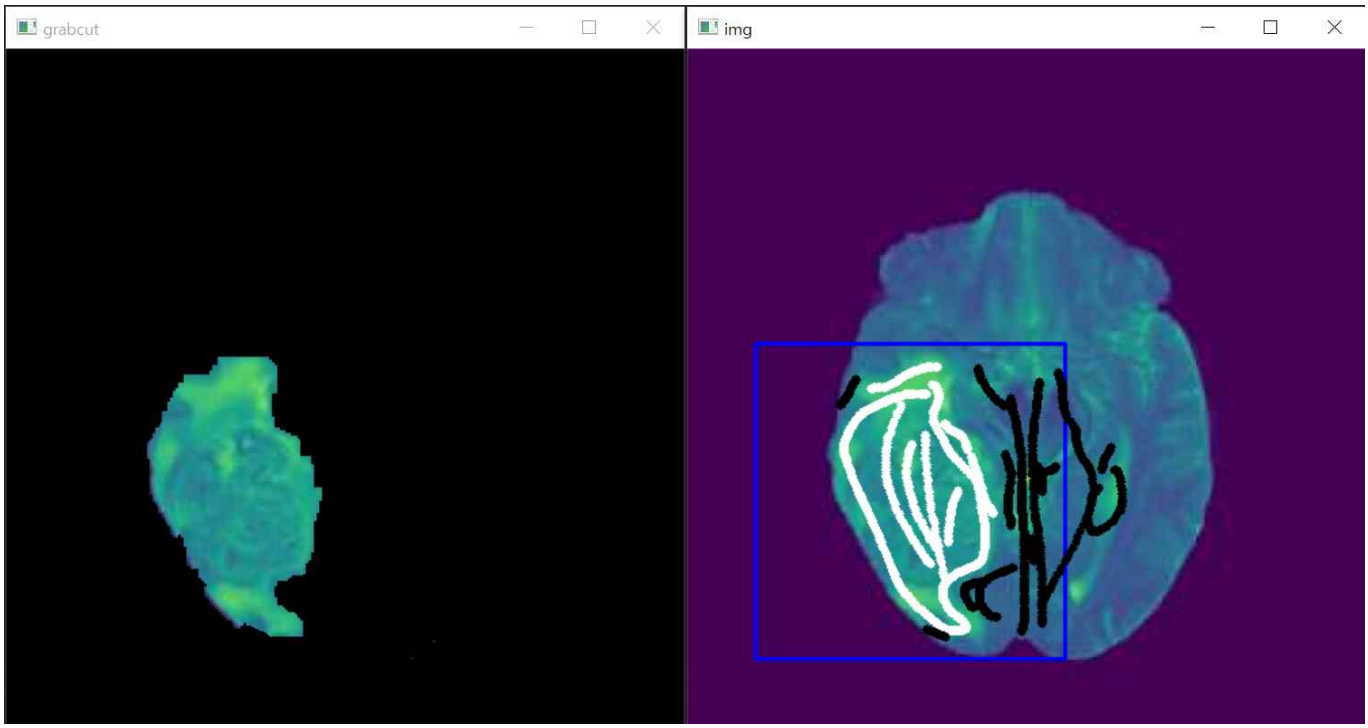
3. Ctrl키를 누르고 분명한 전경을 지정한다.(흰색 선)



4. Shift키를 누르고 분명한 배경을 지정한다.(검정 선)



5. 미흡한 부분이 있으면 Ctrl키, Shift키+마우스 좌클릭으로 다듬어 완성한다.



Problem #3

- 파일 => “pro3.py” 참조

1) 코드 설명

```
1   import numpy as np
2   import os
3   from matplotlib import pyplot as plt
4   from PIL import Image
5   import seaborn as sns
6   import pandas as pd
7   from sklearn.model_selection import train_test_split,StratifiedKFold
8   import tensorflow as tf
9   from tensorflow.keras.layers import Dense, Input, Activation, Flatten
10  from tensorflow.keras.layers import BatchNormalization,Add,Dropout
11  from tensorflow.keras.optimizers import Adam
12  from tensorflow.keras.models import Model, load_model
13  from tensorflow.keras.layers import LeakyReLU, ReLU, Conv2D, MaxPooling2D, BatchNormalization, Conv2DTranspose, UpSampling2D, concatenate
14  from tensorflow.keras import callbacks
15  from tensorflow.keras import backend as K
16
```

1~16 : 라이브러리 임포트 (Pytorch 대신 Tensorflow Keras를 사용함)

```
17  # ################################################################################
18  # 데이터 셋업
19
20  # Global values 및 ndarray 생성
21  n_train = 222
22  n_val = 29
23  n_test = 84
24  ch = 4
25  H = 240
26  W = 240
27  train_img = np.empty((n_train, ch, H, W))
28  train_sol = np.empty((n_train, H, W))
29  val_img = np.empty((n_val, ch, H, W))
30  val_sol = np.empty((n_val, H, W))
31  test_img = np.empty((n_test, ch, H, W))
32  test_sol = np.empty((n_test, H, W))
33
34  Data_path = os.getcwd()
35
36  # train 데이터 리스트 생성
37  walking_train = os.path.join(Data_path, 'clf_w_mask', 'train')
38
39  i = 0
40  for path, dirs, files in os.walk(walking_train):
41      if 'img.npy' in files:
42          path_img = os.path.join(path, 'img.npy')
43          path_seg = os.path.join(path, 'seg.npy')
44          img = np.load(path_img)
45          seg = np.load(path_seg)
46          train_img[i] = img
47          train_sol[i] = seg
48          i += 1
49
50  # validation 데이터 리스트 생성
51  walking_val = os.path.join(Data_path,'clf_w_mask', 'valid')
52  i = 0
53  for path, dirs, files in os.walk(walking_val):
54      if 'img.npy' in files:
55          path_img = os.path.join(path, 'img.npy')
56          path_seg = os.path.join(path, 'seg.npy')
57          img = np.load(path_img)
58          seg = np.load(path_seg)
59          val_img[i] = img
60          val_sol[i] = seg
61          i += 1
62
63  # test 데이터 리스트 생성
64  walking_test = os.path.join(Data_path, 'clf_w_mask', 'test')
65  i = 0
66  for path, dirs, files in os.walk(walking_test):
67      if 'img.npy' in files:
68          path_img = os.path.join(path, 'img.npy')
69
70          path_seg = os.path.join(path, 'seg.npy')
71          img = np.load(path_img)
72          seg = np.load(path_seg)
73          test_img[i] = img
74          test_sol[i] = seg
75          i += 1
76
77
```

17~77 : Training, Validation, Test 데이터 셋을 가져옴

```
78  # img의 intensity를 0~1사이 범위로 Normarlize
79  for i in range(n_train):
80      train_img[i] = ( train_img[i] - np.min(train_img[i]) ) / (np.max(train_img[i]) - np.min(train_img[i]))
81      train_sol[i] = ( train_sol[i] - np.min(train_sol[i]) ) / (np.max(train_sol[i]) - np.min(train_sol[i]))
82
83  for i in range(n_val):
84      val_img[i] = ( val_img[i] - np.min(val_img[i]) ) / (np.max(val_img[i]) - np.min(val_img[i]))
85      val_sol[i] = ( val_sol[i] - np.min(val_sol[i]) ) / (np.max(val_sol[i]) - np.min(val_sol[i]))
86
87  for i in range(n_test):
88      test_img[i] = ( test_img[i] - np.min(test_img[i]) ) / (np.max(test_img[i]) - np.min(test_img[i]))
89      test_sol[i] = ( test_sol[i] - np.min(test_sol[i]) ) / (np.max(test_sol[i]) - np.min(test_sol[i]))
90
91
92  # img의 Shape를 조정 (n, 4, 240, 240) -> (n, 240, 240, 4)
93  train_img = np.transpose(train_img, (0, 2, 3, 1))
94  val_img = np.transpose(val_img, (0, 2, 3, 1))
95  test_img = np.transpose(test_img, (0, 2, 3, 1))
96
97  train_x = train_img
98  valid_x = val_img
99  test_x = test_img
100
101 train_y = train_sol.reshape(n_train, 240, 240, 1)
102 valid_y = val_sol.reshape(n_val, 240, 240, 1)
103 test_y = test_sol.reshape(n_test, 240, 240, 1)
104
105 print(train_x.shape, train_y.shape, valid_x.shape, valid_y.shape)
106
```

78~106 : 학습을 위한 세팅으로 이미지의 밝기를 0~1범위로 정규화하고, 이미지의 차원을 조정함

```
107  # 모델 생성
108  def contract_path(input_shape):
109      input= tf.keras.layers.Input(shape = input_shape)
110      x =  Conv2D(64, (3,3), padding = "same", activation = "relu")(input)
111      x =  Conv2D(64, (3,3), padding = "same", activation = "relu", name = "copy_crop1")(x)
112      x = MaxPooling2D((2, 2))(x)
113      x =  Conv2D(128, (3,3), padding = "same", activation = "relu")(x)
114      x =  Conv2D(128, (3,3), padding = "same", activation = "relu", name = "copy_crop2")(x)
115      x = MaxPooling2D((2, 2))(x)
116      x =  Conv2D(256, (3,3), padding = "same", activation = "relu")(x)
117      x =  Conv2D(256, (3,3), padding = "same", activation = "relu", name = "copy_crop3")(x)
118      x = MaxPooling2D((2, 2))(x)
119      x =  Conv2D(512, (3,3), padding = "same", activation = "relu")(x)
120      x =  Conv2D(512, (3,3), padding = "same", activation = "relu", name = "copy_crop4")(x)
121      x = MaxPooling2D((2, 2))(x)
122      x =  Conv2D(1024, (3,3), padding = "same", activation = "relu")(x)
123      x =  Conv2D(1024, (3,3), padding = "same", activation = "relu", name = "last_layer")(x)
124      contract_path =  tf.keras.Model(inputs = input, outputs = x)
125      return contract_path
126
127  def unet(input_shape, n_classes):
128      contract_model = contract_path(input_shape=input_shape)
129      layer_names  = ["copy_crop1", "copy_crop2",  "copy_crop3" ,"copy_crop4", "last_layer"]
130      layers = [contract_model.get_layer(name).output for name in layer_names]
131
132      extract_model = tf.keras.Model(inputs=contract_model.input, outputs=layers)
133      input= tf.keras.layers.Input(shape =input_shape)
134      output_layers = extract_model(inputs = input)
135      last_layer = output_layers[-1]
136
137      x = Conv2DTranspose(512, 4, (2,2), padding = "same", activation = "relu")(last_layer)
138      x = BatchNormalization()(x)
139      x = tf.keras.layers.Concatenate()([x, output_layers[3]])
140
141      x =  Conv2D(256, (3,3), padding = "same", activation = "relu")(x)
142      x = BatchNormalization()(x)
143
144      x =  Conv2D(256, (3,3), padding = "same", activation = "relu")(x)
145      x = BatchNormalization()(x)
146
147      x = Conv2DTranspose(256, 4, (2,2), padding = "same", activation = "relu")(x)
148      x = BatchNormalization()(x)
149      x = tf.keras.layers.Concatenate()([x, output_layers[2]])
150
151      x =  Conv2D(128, (3,3), padding = "same", activation = "relu")(x)
152      x = BatchNormalization()(x)
153      x =  Conv2D(128, (3,3), padding = "same", activation = "relu")(x)
154      x = BatchNormalization()(x)
155
156      x = Conv2DTranspose(128, 4, (2,2), padding = "same", activation = "relu")(x)
157      x = BatchNormalization()(x)
158      x = tf.keras.layers.Concatenate()([x, output_layers[1]])
159
160
161      x =  Conv2D(64, (3,3), padding = "same", activation = "relu")(x)
162      x = BatchNormalization()(x)
163      x =  Conv2D(64, (3,3), padding = "same", activation = "relu")(x)
164      x = BatchNormalization()(x)
165
166      x = Conv2DTranspose(64, 4, (2,2), padding = "same", activation = "relu")(x)
167      x = BatchNormalization()(x)
168      x = tf.keras.layers.Concatenate()([x, output_layers[0]])
169
170      x =  Conv2D(64, (3,3), padding = "same", activation = "relu")(x)
171      x = BatchNormalization()(x)
172      x =  Conv2D(64, (3,3), padding = "same", activation = "relu")(x)
173      x = BatchNormalization()(x)
174      x =  Conv2D(n_classes, (1,1), activation = "relu")(x)
175
176      model = tf.keras.Model(inputs = input , outputs = x)
177
178      return model
179
```

107~179 : U-net 모델을 정의함.

```
180  model = unet((240, 240, 4), 4)
181  model.summary()
182  model.compile(optimizer = 'adam',
183                loss = 'binary_crossentropy',
184                metrics = ['acc'])
185
186  hist = model.fit(train_x, train_y,
187                   shuffle=True,
188                   validation_data=(valid_x, valid_y),
189                   epochs = 24,
190                   verbose = 1 )
191
192
193  # 모델 저장
194  learning_model_path = os.path.join(Data_path, 'U-NET_Segmentation_trained_model.h5')
195  model.save(learning_model_path)
196  print('Saved trained model at %s ' % learning_model_path)
197
```

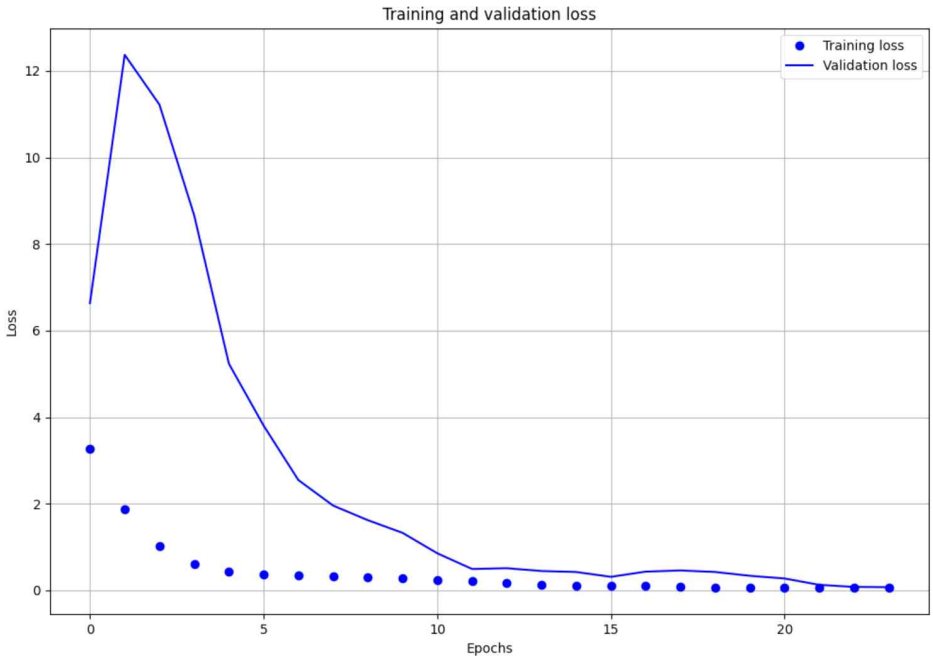
180~197 : 모델을 생성하고 학습시킴. 그리고 데이터 경로에 학습된 모델을 저장함.

```
198  # train val의 loss 출력
199  train_loss =  hist.history['loss'][-1]
200  val_loss =  hist.history['val_loss'][-1]
201  print('Training loss: ', train_loss)
202  print('validation loss: ', val_loss)
203
204  # train, val의 loss를 그래프로 표현
205  plt.figure(figsize = (12,8))
206  plt.plot(hist.history['loss'], 'bo', label='Training loss')
207  plt.plot(hist.history['val_loss'], 'b', label='Validation loss')
208  plt.title('Training and validation loss')
209  plt.xlabel('Epochs')
210  plt.ylabel('Loss')
211  plt.legend()
212  plt.grid()
213  plt.show()
214
```

198~214 : Training과 Validation loss를 출력함.

```
Training loss:  0.060032121837139 13
validation loss:  0.07289446145296097
```

Epoch를 24로 설정하였고, 학습이 끝나자 Training, validation loss가 모두 0에 가까워짐.




```
215  # ##############################################################################
216  # Test 모델 predict
217  my_model = load_model(learning_model_path)
218  test_img_number = 7
219  test_img_ch = 0
220  test_image = test_x[test_img_number]
221  ground_truth= test_y[test_img_number]
222  test_image_input=np.expand_dims(test_image, 0)
223  prediction = (my_model.predict(test_image_input)[0,:,:,0] > 0.5).astype(np.uint8)
224
225  plt.figure(figsize=(16, 8))
226  plt.subplot(241)
227  plt.title('Testing Image')
228  plt.imshow(test_image[:,:,test_img_ch], cmap='gray')
229  plt.subplot(242)
230  plt.title('Testing Solution')
231  plt.imshow(ground_truth[:,:,0], cmap='gray')
232  plt.subplot(243)
233  plt.title('Prediction')
234  plt.imshow(prediction, cmap='gray')
235  plt.subplot(244)
236  plt.title('Prediction on test image')
237  plt.imshow(test_image[:,:,test_img_ch]+prediction, cmap='gray')
238  plt.show()
```

215~238 : 학습된 모델을 이용하여 test 이미지로 테스트

- test 이미지는 총 84개의 세트 중에서 원하는 세트와 채널을 지정할 수 있음.
- 그리고 이미지를 플롯팅함.

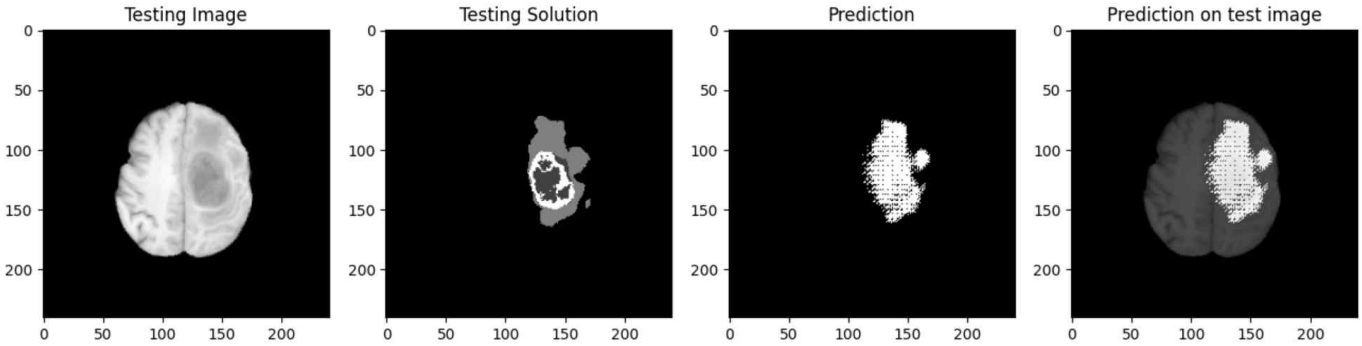
plot1 : 원본 이미지

plot2 : 정답 이미지

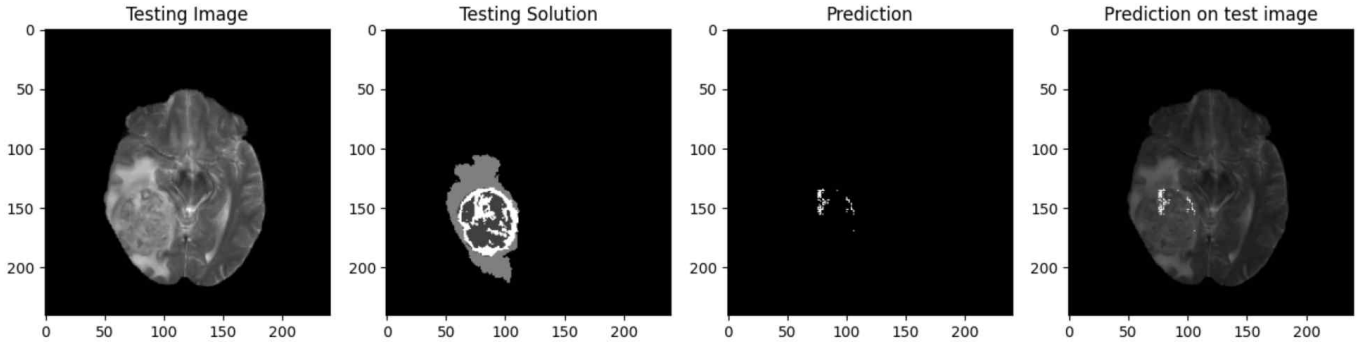
plot3 : U-net을 이용하여 학습한 모델로 Segmentation한 결과

plot4 : 원본 이미지에 학습된 모델을 겹친 이미지

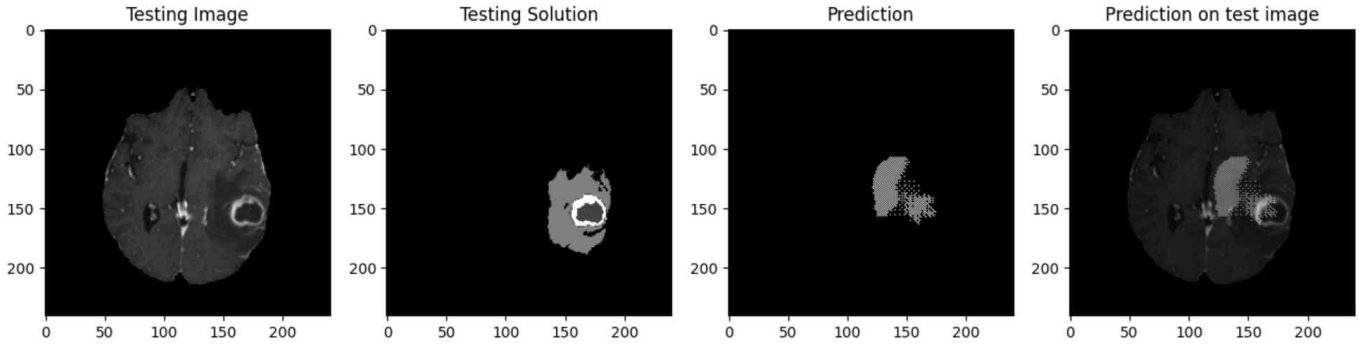
2) 결과



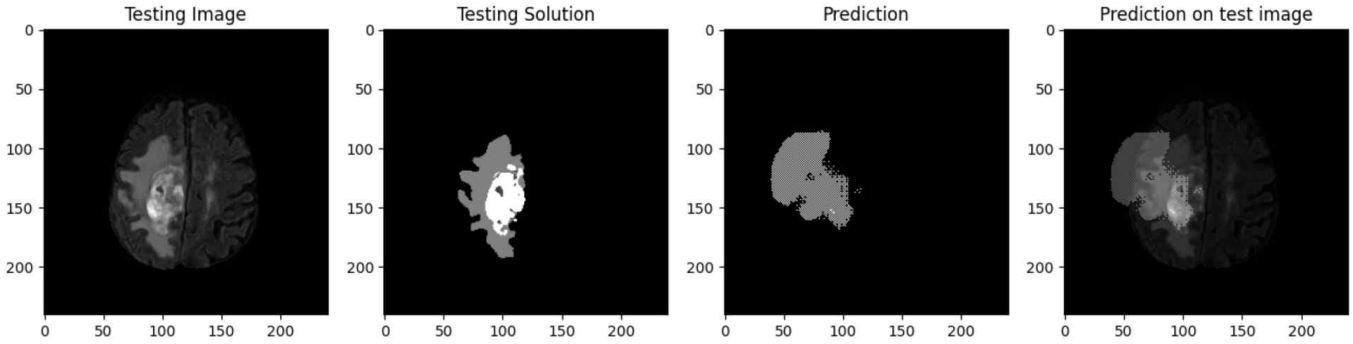
Test 이미지의 8번째 모델, 1번째 채널 이미지



Test 이미지의 2번째 모델, 2번째 채널 이미지



Test 이미지의 22번째 모델, 3번째 채널 이미지



Test 이미지의 38번째 모델, 4번째 채널 이미지

※ Dice Similarity Score는 진행하지 못하였습니다.