```
!nvidia-smi
```

```
Sat Jul 22 11:34:25 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   51C    P8    10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

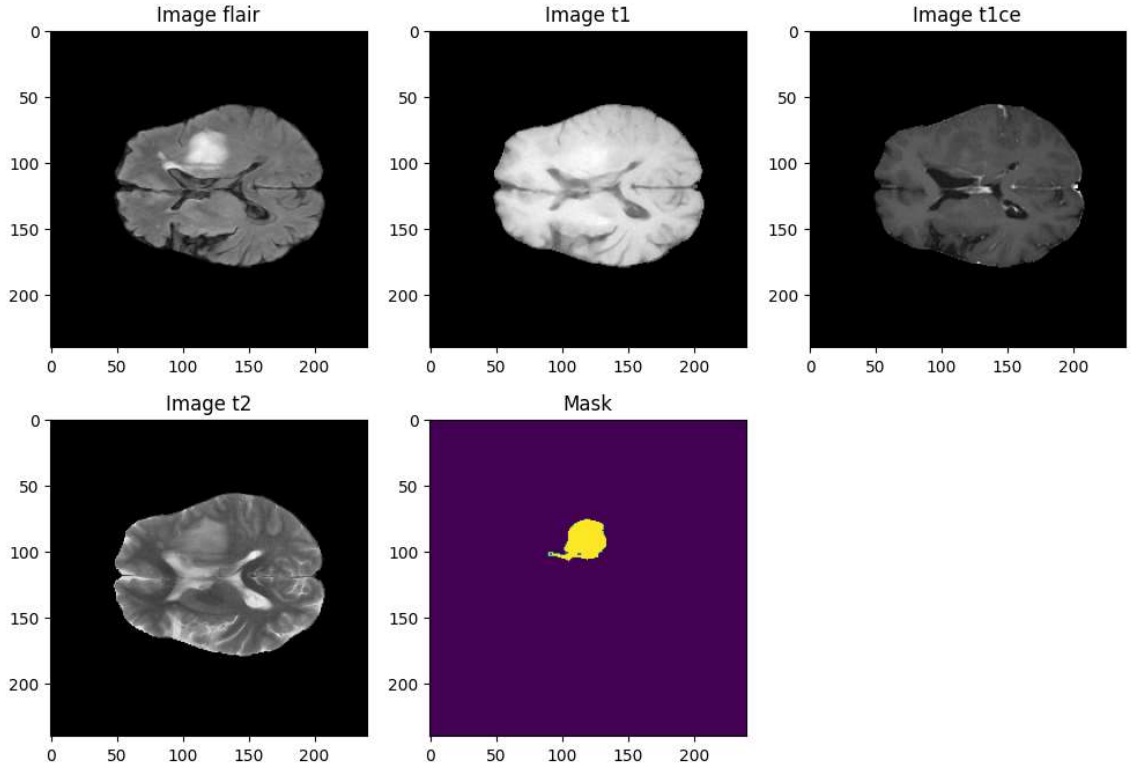
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

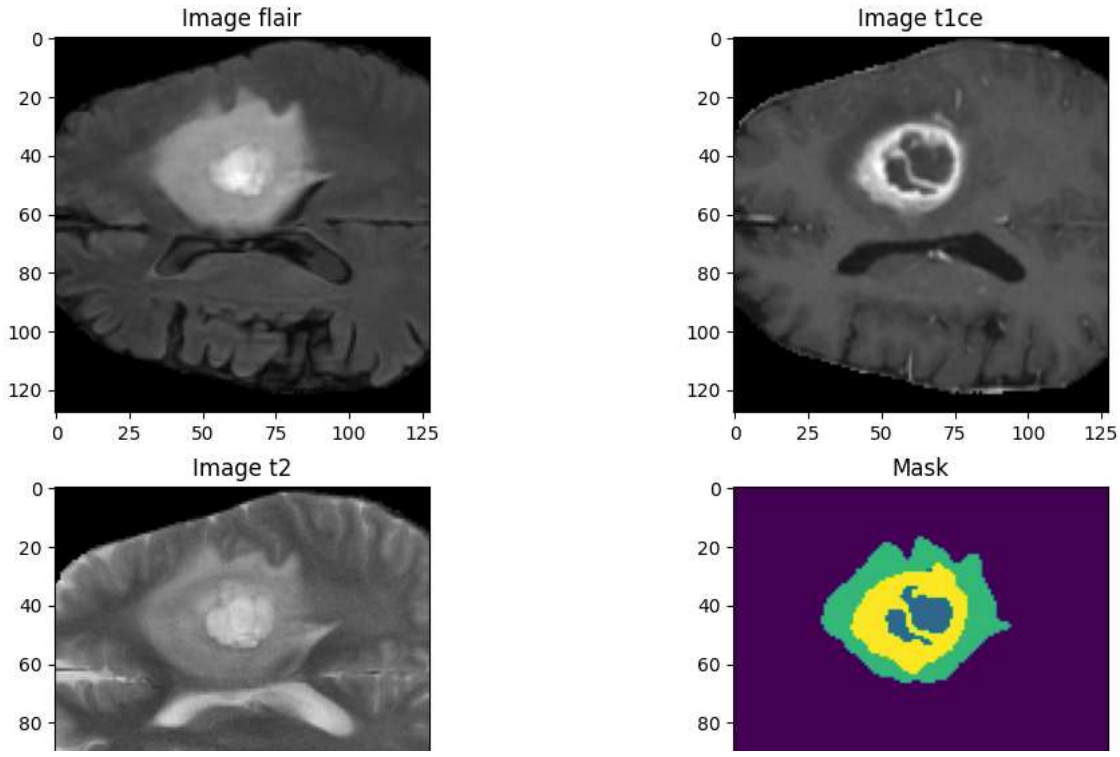
```
import numpy as np
import nibabel as nib
import glob
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from tifffile import imsave

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()


TRAIN_DATASET_PATH = '/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/'
#VALIDATION_DATASET_PATH = 'BraTS2020_ValidationData/MICCAI_BraTS2020_ValidationData'

test_image_flair=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_355/BraTS20_Training_355_flair.nii').get_fdata()
print(test_image_flair.max())
#Scalers are applied to 1D so let us reshape and then reshape back to original shape.
test_image_flair=scaler.fit_transform(test_image_flair.reshape(-1, test_image_flair.shape[-1])).reshape(test_image_flair.shape)


test_image_t1=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_355/BraTS20_Training_355_t1.nii').get_fdata()
test_image_t1=scaler.fit_transform(test_image_t1.reshape(-1, test_image_t1.shape[-1])).reshape(test_image_t1.shape)

test_image_t1ce=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_355/BraTS20_Training_355_t1ce.nii').get_fdata()
test_image_t1ce=scaler.fit_transform(test_image_t1ce.reshape(-1, test_image_t1ce.shape[-1])).reshape(test_image_t1ce.shape)

test_image_t2=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_355/BraTS20_Training_355_t2.nii').get_fdata()
test_image_t2=scaler.fit_transform(test_image_t2.reshape(-1, test_image_t2.shape[-1])).reshape(test_image_t2.shape)

test_mask=nib.load(TRAIN_DATASET_PATH + 'BraTS20_Training_355/BraTS20_Training_355_seg.nii').get_fdata()
test_mask=test_mask.astype(np.uint8)

print(np.unique(test_mask))  #0, 1, 2, 4 (Need to reencode to 0, 1, 2, 3)
test_mask[test_mask==4] = 3  #Reassign mask values 4 to 3
print(np.unique(test_mask))

import random
n_slice=random.randint(0, test_mask.shape[2])

plt.figure(figsize=(12, 8))

plt.subplot(231)
plt.imshow(test_image_flair[:,:,n_slice], cmap='gray')
plt.title('Image flair')
plt.subplot(232)
plt.imshow(test_image_t1[:,:,n_slice], cmap='gray')
plt.title('Image t1')
plt.subplot(233)
plt.imshow(test_image_t1ce[:,:,n_slice], cmap='gray')
plt.title('Image t1ce')
plt.subplot(234)
plt.imshow(test_image_t2[:,:,n_slice], cmap='gray')
plt.title('Image t2')
```

```
plt.subplot(235)
plt.imshow(test_mask[:,:,n_slice])
plt.title('Mask')
plt.show()
```

```
1854.603271484375
[0 1 2 4]
[0 1 2 3]
```




```
combined_x = np.stack([test_image_flair, test_image_t1ce, test_image_t2], axis=3)

#Crop to a size to be divisible by 64 so we can later extract 64x64x64 patches.
#cropping x, y, and z
#combined_x=combined_x[24:216, 24:216, 13:141]

combined_x=combined_x[56:184, 56:184, 13:141] #Crop to 128x128x128x4

#Do the same for mask
test_mask = test_mask[56:184, 56:184, 13:141]


n_slice=random.randint(0, test_mask.shape[2])
plt.figure(figsize=(12, 8))

plt.subplot(221)
plt.imshow(combined_x[:,:,n_slice, 0], cmap='gray')
plt.title('Image flair')
plt.subplot(222)
plt.imshow(combined_x[:,:,n_slice, 1], cmap='gray')
plt.title('Image t1ce')
plt.subplot(223)
plt.imshow(combined_x[:,:,n_slice, 2], cmap='gray')
plt.title('Image t2')
plt.subplot(224)
plt.imshow(test_mask[:,:,n_slice])
plt.title('Mask')
plt.show()
```




```
imsave('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/combined255.tif', combined_x)
np.save('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/combined255.npy', combined_x)
```

```
<ipython-input-7-cf1ae4b881e9>:1: DeprecationWarning: <tifffile.imsave> is deprecated. Use tifffile.imwrite
  imsave('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/combined255.tif', combined_x)
```

```
my_img=np.load('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/combined255.npy')

test_mask = to_categorical(test_mask, num_classes=4)


t2_list = sorted(glob.glob('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*t2.nii'))
t1ce_list = sorted(glob.glob('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*t1ce.nii'))
flair_list = sorted(glob.glob('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*flair.nii'))
mask_list = sorted(glob.glob('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*seg.nii'))


for img in range(len(t2_list)):   #Using t1_list as all lists are of same size
    print("Now preparing image and masks number: ", img)

    temp_image_t2=nib.load(t2_list[img]).get_fdata()
    temp_image_t2=scaler.fit_transform(temp_image_t2.reshape(-1, temp_image_t2.shape[-1])).reshape(temp_image_t2.shape)

    temp_image_t1ce=nib.load(t1ce_list[img]).get_fdata()
    temp_image_t1ce=scaler.fit_transform(temp_image_t1ce.reshape(-1, temp_image_t1ce.shape[-1])).reshape(temp_image_t1ce.shape)

    temp_image_flair=nib.load(flair_list[img]).get_fdata()
    temp_image_flair=scaler.fit_transform(temp_image_flair.reshape(-1, temp_image_flair.shape[-1])).reshape(temp_image_flair.shape)

    temp_mask=nib.load(mask_list[img]).get_fdata()
    temp_mask=temp_mask.astype(np.uint8)
    temp_mask[temp_mask==4] = 3  #Reassign mask values 4 to 3
    #print(np.unique(temp_mask))


    temp_combined_images = np.stack([temp_image_flair, temp_image_t1ce, temp_image_t2], axis=3)

    #Crop to a size to be divisible by 64 so we can later extract 64x64x64 patches.
    #cropping x, y, and z
    temp_combined_images=temp_combined_images[56:184, 56:184, 13:141]
    temp_mask = temp_mask[56:184, 56:184, 13:141]

    val, counts = np.unique(temp_mask, return_counts=True)

    if (1 - (counts[0]/counts.sum())) > 0.01:  #At least 1% useful volume with labels that are not 0
        print("Save Me")
        temp_mask= to_categorical(temp_mask, num_classes=4)
        np.save('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_3channels/images/image_'+str(img)+'.npy', temp_com
        np.save('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_3channels/masks/mask_'+str(img)+'.npy', temp_mask)

    else:
```

```
        print("don`t save")
```

```
Now preparing image and masks number:  0
Save Me
Now preparing image and masks number:  1
Save Me
Now preparing image and masks number:  2
Save Me
Now preparing image and masks number:  3
Save Me
Now preparing image and masks number:  4
don`t save
Now preparing image and masks number:  5
Save Me
Now preparing image and masks number:  6
Save Me
Now preparing image and masks number:  7
Save Me
Now preparing image and masks number:  8
Save Me
Now preparing image and masks number:  9
Save Me
Now preparing image and masks number:  10
Save Me
Now preparing image and masks number:  11
Save Me
Now preparing image and masks number:  12
Save Me
Now preparing image and masks number:  13
Save Me
Now preparing image and masks number:  14
Save Me
Now preparing image and masks number:  15
Save Me
Now preparing image and masks number:  16
Save Me
Now preparing image and masks number:  17
Save Me
Now preparing image and masks number:  18
Save Me
Now preparing image and masks number:  19
Save Me
Now preparing image and masks number:  20
Save Me
Now preparing image and masks number:  21
Save Me
Now preparing image and masks number:  22
Save Me
Now preparing image and masks number:  23
Save Me
Now preparing image and masks number:  24
Save Me
Now preparing image and masks number:  25
Save Me
Now preparing image and masks number:  26
Save Me
Now preparing image and masks number:  27
don`t save
Now preparing image and masks number:  28
Save Me
```

```
!pip install split-folders
```

```
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
```

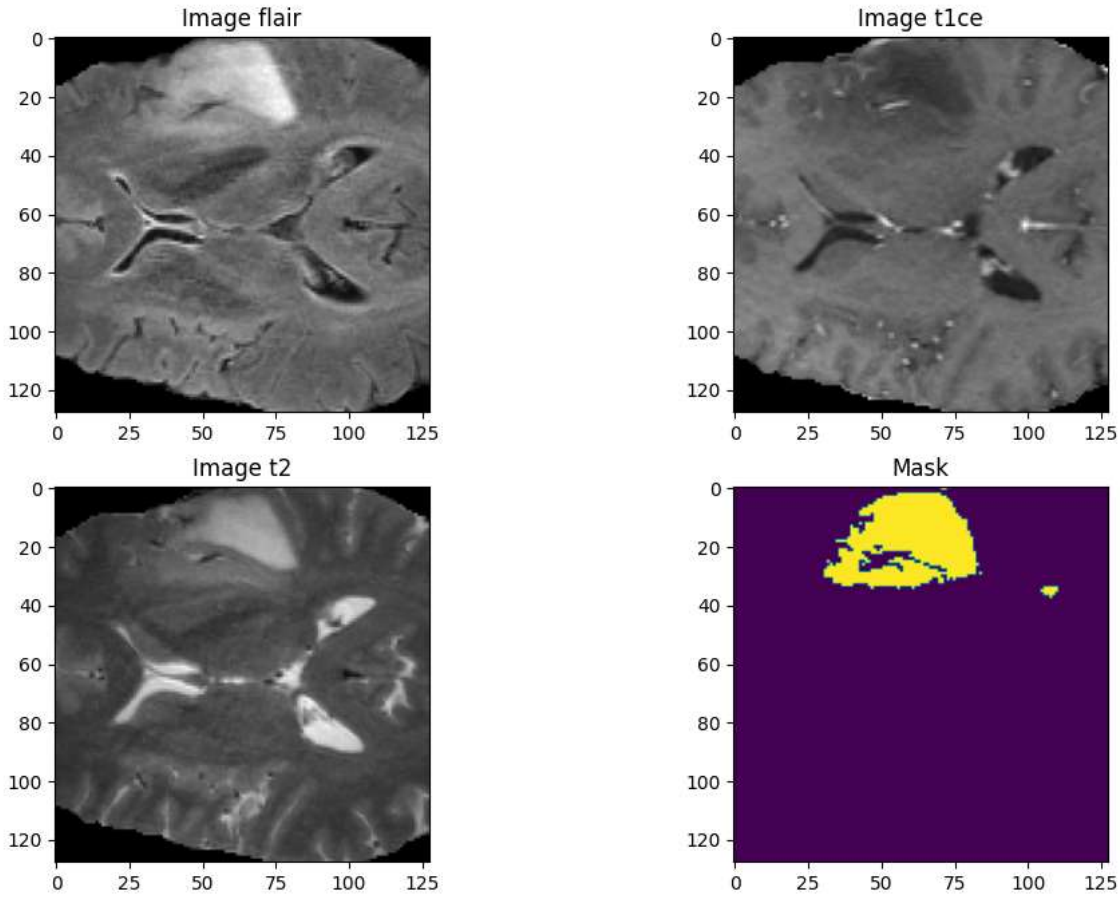
```
import splitfolders  # or import split_folders

input_folder = '/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_3channels/'
output_folder = '/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/'
# Split with a ratio.
# To only split into training and validation set, set a tuple to `ratio`, i.e, `(.8, .2)`.
splitfolders.ratio(input_folder, output=output_folder, seed=42, ratio=(.75, .25), group_prefix=None) # default values
```

```
Copying files: 0 files [00:00, ? files/s]
Copying files: 1 files [00:00,  9.79 files/s]
Copying files: 2 files [00:00,  4.27 files/s]
Copying files: 3 files [00:00,  3.71 files/s]
Copying files: 4 files [00:01,  3.33 files/s]
Copying files: 5 files [00:02,  1.80 files/s]
Copying files: 6 files [00:03,  1.36 files/s]
Copying files: 7 files [00:03,  1.53 files/s]
Copying files: 8 files [00:04,  1.77 files/s]
```

```
Copying files: 9 files [00:04,  2.03 files/s]
Copying files: 10 files [00:05,  1.47 files/s]
Copying files: 11 files [00:07,  1.05s/ files]
Copying files: 12 files [00:08,  1.02 files/s]
Copying files: 13 files [00:09,  1.01 files/s]
Copying files: 14 files [00:13,  1.93s/ files]
Copying files: 15 files [00:15,  1.87s/ files]
Copying files: 16 files [00:15,  1.47s/ files]
Copying files: 17 files [00:16,  1.40s/ files]
Copying files: 18 files [00:17,  1.08s/ files]
Copying files: 19 files [00:17,  1.14 files/s]
Copying files: 20 files [00:19,  1.34s/ files]
Copying files: 21 files [00:20,  1.03s/ files]
Copying files: 22 files [00:21,  1.04s/ files]
Copying files: 23 files [00:21,  1.18 files/s]
Copying files: 24 files [00:22,  1.13 files/s]
Copying files: 25 files [00:23,  1.10 files/s]
Copying files: 26 files [00:25,  1.23s/ files]
Copying files: 27 files [00:26,  1.03 files/s]
Copying files: 28 files [00:27,  1.06s/ files]
Copying files: 29 files [00:29,  1.37s/ files]
Copying files: 30 files [00:30,  1.31s/ files]
Copying files: 31 files [00:31,  1.14s/ files]
Copying files: 32 files [00:32,  1.16s/ files]
Copying files: 33 files [00:32,  1.06 files/s]
Copying files: 34 files [00:34,  1.01s/ files]
Copying files: 35 files [00:35,  1.02s/ files]
Copying files: 36 files [00:36,  1.01s/ files]
Copying files: 37 files [00:37,  1.04s/ files]
Copying files: 38 files [00:38,  1.04s/ files]
Copying files: 39 files [00:39,  1.09s/ files]
Copying files: 40 files [00:40,  1.09s/ files]
Copying files: 41 files [00:42,  1.21s/ files]
Copying files: 42 files [00:42,  1.03s/ files]
Copying files: 43 files [00:43,  1.08s/ files]
Copying files: 44 files [00:45,  1.10s/ files]
Copying files: 45 files [00:46,  1.20s/ files]
Copying files: 46 files [00:47,  1.24s/ files]
Copying files: 47 files [00:48,  1.21s/ files]
Copying files: 48 files [00:49,  1.01 files/s]
Copying files: 49 files [00:49,  1.22 files/s]
Copying files: 50 files [00:51,  1.07 files/s]
Copying files: 51 files [00:52,  1.00s/ files]
Copying files: 52 files [00:53,  1.00 files/s]
Copying files: 53 files [00:54,  1.01 files/s]
Copying files: 54 files [00:55,  1.23s/ files]
Copying files: 55 files [00:57,  1.23s/ files]
```

```
import os
import numpy as np


def load_img(img_dir, img_list):
    images=[]
    for i, image_name in enumerate(img_list):
        if (image_name.split('.')[1] == 'npy'):

            image = np.load(img_dir+image_name)

            images.append(image)
    images = np.array(images)

    return(images)




def imageLoader(img_dir, img_list, mask_dir, mask_list, batch_size):

    L = len(img_list)

    #keras needs the generator infinite, so we will use while true
    while True:

        batch_start = 0
        batch_end = batch_size

        while batch_start < L:
            limit = min(batch_end, L)

            X = load_img(img_dir, img_list[batch_start:limit])
            Y = load_img(mask_dir, mask_list[batch_start:limit])

            yield (X,Y) #a tuple with two numpy arrays with batch_size samples
```

```
            batch_start += batch_size
            batch_end += batch_size


from matplotlib import pyplot as plt
import random

train_img_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/images/"
train_mask_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/masks/"
train_img_list=os.listdir(train_img_dir)
train_mask_list = os.listdir(train_mask_dir)

batch_size = 2

train_img_datagen = imageLoader(train_img_dir, train_img_list,
                                train_mask_dir, train_mask_list, batch_size)



img, msk = train_img_datagen.__next__()


img_num = random.randint(0,img.shape[0]-1)
test_img=img[img_num]
test_mask=msk[img_num]
test_mask=np.argmax(test_mask, axis=3)

n_slice=random.randint(0, test_mask.shape[2])
plt.figure(figsize=(12, 8))

plt.subplot(221)
plt.imshow(test_img[:,:,n_slice, 0], cmap='gray')
plt.title('Image flair')
plt.subplot(222)
plt.imshow(test_img[:,:,n_slice, 1], cmap='gray')
plt.title('Image t1ce')
plt.subplot(223)
plt.imshow(test_img[:,:,n_slice, 2], cmap='gray')
plt.title('Image t2')
plt.subplot(224)
plt.imshow(test_mask[:,:,n_slice])
plt.title('Mask')
plt.show()
```



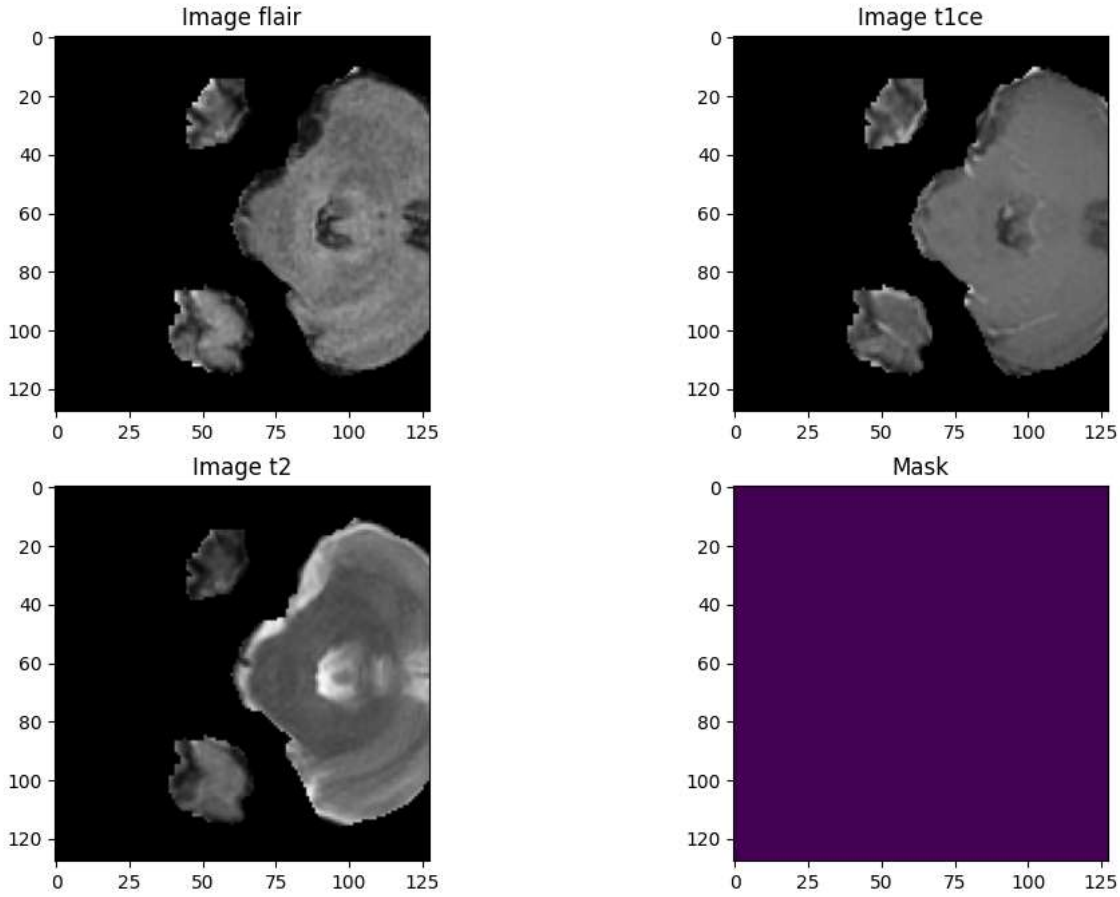
```
from keras.models import Model
from keras.layers import Input, Conv3D, MaxPooling3D, concatenate, Conv3DTranspose, BatchNormalization, Dropout, Lambda
from keras.optimizers import Adam
from keras.metrics import MeanIoU

kernel_initializer =  'he_uniform'


#################################################################
def simple_unet_model(IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH, IMG_CHANNELS, num_classes):
#Build the model
    inputs = Input((IMG_HEIGHT, IMG_WIDTH, IMG_DEPTH, IMG_CHANNELS))
    #s = Lambda(lambda x: x / 255)(inputs)   #No need for this if we normalize our inputs beforehand
    s = inputs

    #Contraction path
    c1 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(s)
    c1 = Dropout(0.1)(c1)
    c1 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c1)
    p1 = MaxPooling3D((2, 2, 2))(c1)

    c2 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p1)
    c2 = Dropout(0.1)(c2)
    c2 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c2)
    p2 = MaxPooling3D((2, 2, 2))(c2)

    c3 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p2)
    c3 = Dropout(0.2)(c3)
    c3 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c3)
    p3 = MaxPooling3D((2, 2, 2))(c3)

    c4 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p3)
    c4 = Dropout(0.2)(c4)
    c4 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c4)
    p4 = MaxPooling3D(pool_size=(2, 2, 2))(c4)

    c5 = Conv3D(256, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(p4)
    c5 = Dropout(0.3)(c5)
    c5 = Conv3D(256, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c5)

    #Expansive path
    u6 = Conv3DTranspose(128, (2, 2, 2), strides=(2, 2, 2), padding='same')(c5)
    u6 = concatenate([u6, c4])
    c6 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u6)
    c6 = Dropout(0.2)(c6)
    c6 = Conv3D(128, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c6)

    u7 = Conv3DTranspose(64, (2, 2, 2), strides=(2, 2, 2), padding='same')(c6)
    u7 = concatenate([u7, c3])
    c7 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u7)
    c7 = Dropout(0.2)(c7)
    c7 = Conv3D(64, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c7)

    u8 = Conv3DTranspose(32, (2, 2, 2), strides=(2, 2, 2), padding='same')(c7)
    u8 = concatenate([u8, c2])
    c8 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u8)
    c8 = Dropout(0.1)(c8)
    c8 = Conv3D(32, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c8)

    u9 = Conv3DTranspose(16, (2, 2, 2), strides=(2, 2, 2), padding='same')(c8)
    u9 = concatenate([u9, c1])
    c9 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(u9)
    c9 = Dropout(0.1)(c9)
    c9 = Conv3D(16, (3, 3, 3), activation='relu', kernel_initializer=kernel_initializer, padding='same')(c9)

    outputs = Conv3D(num_classes, (1, 1, 1), activation='softmax')(c9)

    model = Model(inputs=[inputs], outputs=[outputs])
    #compile model outside of this function to make it flexible.
    model.summary()

    return model

#Test if everything is working ok.
model = simple_unet_model(128, 128, 128, 3, 4)
print(model.input_shape)
print(model.output_shape)
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 128, 128, 1  0           []                            
```

```
                                    28, 3)]

 conv3d (Conv3D)                    (None, 128, 128, 12  1312        ['input_1[0][0]']
                                    8, 16)

 dropout (Dropout)                  (None, 128, 128, 12  0           ['conv3d[0][0]']
                                    8, 16)

 conv3d_1 (Conv3D)                  (None, 128, 128, 12  6928        ['dropout[0][0]']
                                    8, 16)

 max_pooling3d (MaxPooling3D)       (None, 64, 64, 64,   0           ['conv3d_1[0][0]']
                                    16)

 conv3d_2 (Conv3D)                  (None, 64, 64, 64,   13856       ['max_pooling3d[0][0]']
                                    32)

 dropout_1 (Dropout)                (None, 64, 64, 64,   0           ['conv3d_2[0][0]']
                                    32)

 conv3d_3 (Conv3D)                  (None, 64, 64, 64,   27680       ['dropout_1[0][0]']
                                    32)

 max_pooling3d_1 (MaxPooling3D)  (None, 32, 32, 32,   0           ['conv3d_3[0][0]']
                                    32)

 conv3d_4 (Conv3D)                  (None, 32, 32, 32,   55360       ['max_pooling3d_1[0][0]']
                                    64)

 dropout_2 (Dropout)                (None, 32, 32, 32,   0           ['conv3d_4[0][0]']
                                    64)

 conv3d_5 (Conv3D)                  (None, 32, 32, 32,   110656      ['dropout_2[0][0]']
                                    64)

 max_pooling3d_2 (MaxPooling3D)  (None, 16, 16, 16,   0           ['conv3d_5[0][0]']
                                    64)

 conv3d_6 (Conv3D)                  (None, 16, 16, 16,   221312      ['max_pooling3d_2[0][0]']
                                    128)

 dropout_3 (Dropout)                (None, 16, 16, 16,   0           ['conv3d_6[0][0]']
                                    128)

 conv3d_7 (Conv3D)                  (None, 16, 16, 16,   442496      ['dropout_3[0][0]']
                                    128)

 max_pooling3d_3 (MaxPooling3D)  (None, 8, 8, 8, 128  0           ['conv3d_7[0][0]']
                                    )

 conv3d_8 (Conv3D)                  (None, 8, 8, 8, 256  884992      ['max_pooling3d_3[0][0]']
                                    )
```

```
import os
import numpy as np
#import tensorflow as tf
import keras
from matplotlib import pyplot as plt
import glob
import random



train_img_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/images/"
train_mask_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/masks/"

img_list = os.listdir(train_img_dir)
msk_list = os.listdir(train_mask_dir)

num_images = len(os.listdir(train_img_dir))

img_num = random.randint(0,num_images-1)
test_img = np.load(train_img_dir+img_list[img_num])
test_mask = np.load(train_mask_dir+msk_list[img_num])
test_mask = np.argmax(test_mask, axis=3)

n_slice=random.randint(0, test_mask.shape[2])
plt.figure(figsize=(12, 8))

plt.subplot(221)
plt.imshow(test_img[:,:,n_slice, 0], cmap='gray')
plt.title('Image flair')
plt.subplot(222)
plt.imshow(test_img[:,:,n_slice, 1], cmap='gray')
plt.title('Image t1ce')
plt.subplot(223)
```

```
plt.imshow(test_img[:,:,n_slice, 2], cmap='gray')
plt.title('Image t2')
plt.subplot(224)
plt.imshow(test_mask[:,:,n_slice])
plt.title('Mask')
plt.show()
```

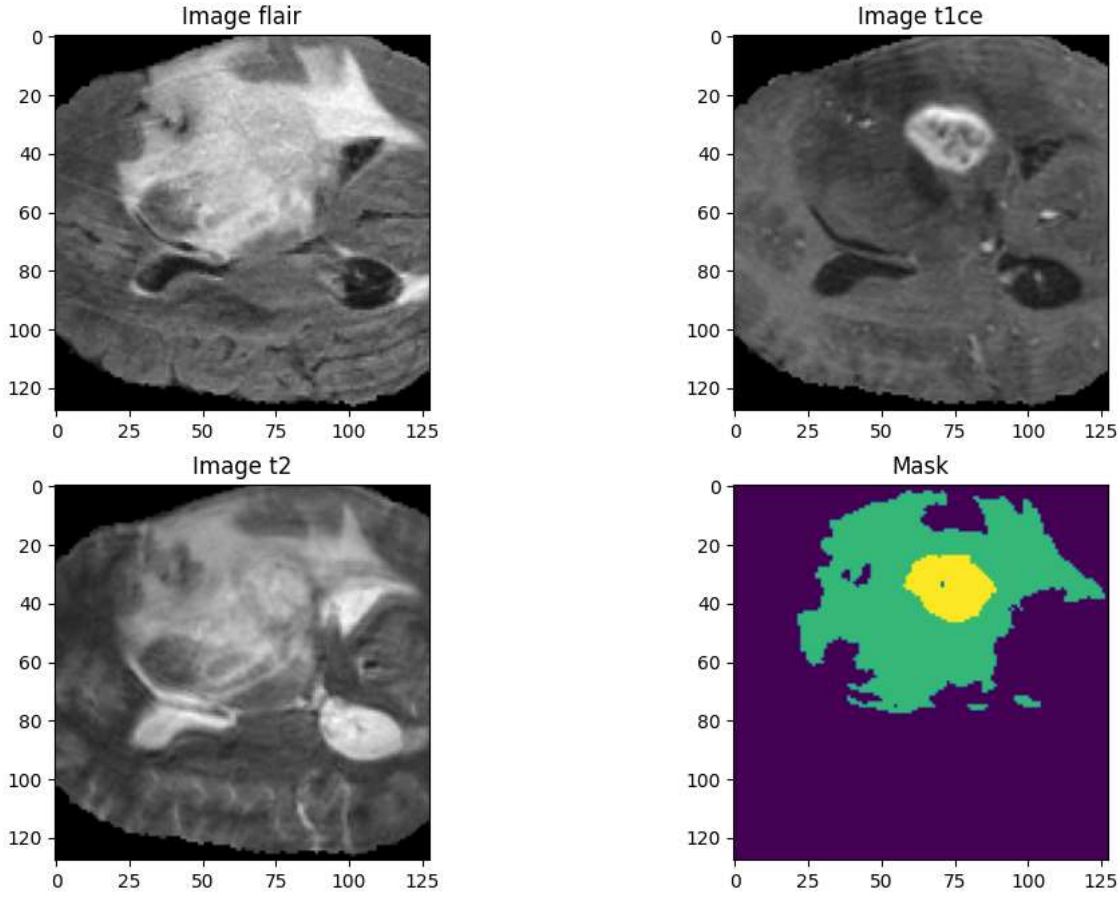



```
import pandas as pd
columns = ['0','1', '2', '3']
df = pd.DataFrame(columns=columns)
train_mask_list = sorted(glob.glob('/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/masks/*.npy'))
for img in range(len(train_mask_list)):
    print(img)
    temp_image=np.load(train_mask_list[img])
    temp_image = np.argmax(temp_image, axis=3)
    val, counts = np.unique(temp_image, return_counts=True)
    zipped = zip(columns, counts)
    conts_dict = dict(zipped)

    df = df.append(conts_dict, ignore_index=True)

label_0 = df['0'].sum()
label_1 = df['1'].sum()
label_2 = df['1'].sum()
label_3 = df['3'].sum()
total_labels = label_0 + label_1 + label_2 + label_3
n_classes = 4
#Class weights claculation: n_samples / (n_classes * n_samples_for_class)
wt0 = round((total_labels/(n_classes*label_0)), 2) #round to 2 decimals
wt1 = round((total_labels/(n_classes*label_1)), 2)
wt2 = round((total_labels/(n_classes*label_2)), 2)
wt3 = round((total_labels/(n_classes*label_3)), 2)
```

```
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
244
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
245
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
246
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
247
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
248
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
249
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
250
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
251
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
252
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
253
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
254
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
255
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
256
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
257
<ipython-input-22-0ef6bc94a4ce>:13: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fu
  df = df.append(conts_dict, ignore_index=True)
```

```
train_img_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/images/"
train_mask_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/train/masks/"

val_img_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/val/images/"
val_mask_dir = "/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/val/masks/"

train_img_list=os.listdir(train_img_dir)
train_mask_list = os.listdir(train_mask_dir)

val_img_list=os.listdir(val_img_dir)
val_mask_list = os.listdir(val_mask_dir)


batch_size = 2

train_img_datagen = imageLoader(train_img_dir, train_img_list,
                                train_mask_dir, train_mask_list, batch_size)

val_img_datagen = imageLoader(val_img_dir, val_img_list,
                                val_mask_dir, val_mask_list, batch_size)



img, msk = train_img_datagen.__next__()

img_num = random.randint(0,img.shape[0]-1)
test_img=img[img_num]
test_mask=msk[img_num]
test_mask=np.argmax(test_mask, axis=3)

n_slice=random.randint(0, test_mask.shape[2])
plt.figure(figsize=(12, 8))

plt.subplot(221)
plt.imshow(test_img[:,:,n_slice, 0], cmap='gray')
plt.title('Image flair')
plt.subplot(222)
plt.imshow(test_img[:,:,n_slice, 1], cmap='gray')
plt.title('Image t1ce')
```

```
plt.subplot(223)
plt.imshow(test_img[:,:,n_slice, 2], cmap='gray')
plt.title('Image t2')
plt.subplot(224)
plt.imshow(test_mask[:,:,n_slice])
plt.title('Mask')
plt.show()
```




```
print(wt0)
```

```
0.26
```

```
!pip install segmentation_models_3D
```

```
Requirement already satisfied: segmentation_models_3D in /usr/local/lib/python3.10/dist-packages (1.0.4)
Requirement already satisfied: tensorflow>=2.8.0 in /usr/local/lib/python3.10/dist-packages (from segmentation_models_3D) (2.12.0)
Requirement already satisfied: keras-applications>=1.0.8 in /usr/local/lib/python3.10/dist-packages (from segmentation_models_3D) (
Requirement already satisfied: classification-models-3D>=1.0.6 in /usr/local/lib/python3.10/dist-packages (from segmentation_models
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from keras-applications>=1.0.8->segmentatio
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages (from keras-applications>=1.0.8->segmentation_models
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_mode
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_m
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_mo
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_models_
Requirement already satisfied: keras<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_mo
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_m
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_models_3D
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/p
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_models_3
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_models_
Requirement already satisfied: tensorboard<2.13,>=2.12 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmenta
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation_mo
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segment
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.8.0->segmentation
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow>=2
Requirement already satisfied: ml-dtypes>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow>=2.8.0->se
Requirement already satisfied: scipy>=1.7 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow>=2.8.0->segmenta
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tens
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensor
```

```
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tenso
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensor
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorbo
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tens
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.13
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-au
```

```
wt0, wt1, wt2, wt3 = 0.25,0.25,0.25,0.25
import segmentation_models_3D as sm
dice_loss = sm.losses.DiceLoss(class_weights=np.array([wt0, wt1, wt2, wt3]))
focal_loss = sm.losses.CategoricalFocalLoss()
total_loss = dice_loss + (1 * focal_loss)

metrics = ['accuracy', sm.metrics.IOUScore(threshold=0.5)]

LR = 0.0001
optim = keras.optimizers.Adam(LR)
```

```
Segmentation Models: using `tf.keras` framework.
```

```
steps_per_epoch = len(train_img_list)//batch_size
val_steps_per_epoch = len(val_img_list)//batch_size


model = simple_unet_model(IMG_HEIGHT=128,
                          IMG_WIDTH=128,
                          IMG_DEPTH=128,
                          IMG_CHANNELS=3,
                          num_classes=4)

model.compile(optimizer = optim, loss=total_loss, metrics=metrics)
print(model.summary())

print(model.input_shape)
print(model.output_shape)
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 128, 128, 1  0           []
                                28, 3)]

 conv3d_19 (Conv3D)             (None, 128, 128, 12  1312        ['input_2[0][0]']
                                8, 16)

 dropout_9 (Dropout)            (None, 128, 128, 12  0           ['conv3d_19[0][0]']
                                8, 16)

 conv3d_20 (Conv3D)             (None, 128, 128, 12  6928        ['dropout_9[0][0]']
                                8, 16)

 max_pooling3d_4 (MaxPooling3D)  (None, 64, 64, 64,   0          ['conv3d_20[0][0]']
                                16)

 conv3d_21 (Conv3D)             (None, 64, 64, 64,   13856       ['max_pooling3d_4[0][0]']
                                32)

 dropout_10 (Dropout)           (None, 64, 64, 64,   0           ['conv3d_21[0][0]']
                                32)

 conv3d_22 (Conv3D)             (None, 64, 64, 64,   27680       ['dropout_10[0][0]']
                                32)

 max_pooling3d_5 (MaxPooling3D)  (None, 32, 32, 32,   0          ['conv3d_22[0][0]']
                                32)

 conv3d_23 (Conv3D)             (None, 32, 32, 32,   55360       ['max_pooling3d_5[0][0]']
                                64)

 dropout_11 (Dropout)           (None, 32, 32, 32,   0           ['conv3d_23[0][0]']
                                64)

 conv3d_24 (Conv3D)             (None, 32, 32, 32,   110656      ['dropout_11[0][0]']
                                64)
```

```
max_pooling3d_6 (MaxPooling3D)  (None, 16, 16, 16,   0          ['conv3d_24[0][0]']
                                64)

conv3d_25 (Conv3D)              (None, 16, 16, 16,   221312     ['max_pooling3d_6[0][0]']
                                128)

dropout_12 (Dropout)            (None, 16, 16, 16,   0          ['conv3d_25[0][0]']
                                128)

conv3d_26 (Conv3D)              (None, 16, 16, 16,   442496     ['dropout_12[0][0]']
                                128)

max_pooling3d_7 (MaxPooling3D)  (None, 8, 8, 8, 128  0          ['conv3d_26[0][0]']
                                )

conv3d_27 (Conv3D)              (None, 8, 8, 8, 256  884992     ['max_pooling3d_7[0][0]']
                                )
```

```
history=model.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=10,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )

model.save('/content/drive/MyDrive/archive (11)/newbrats_3d.hdf5')
```

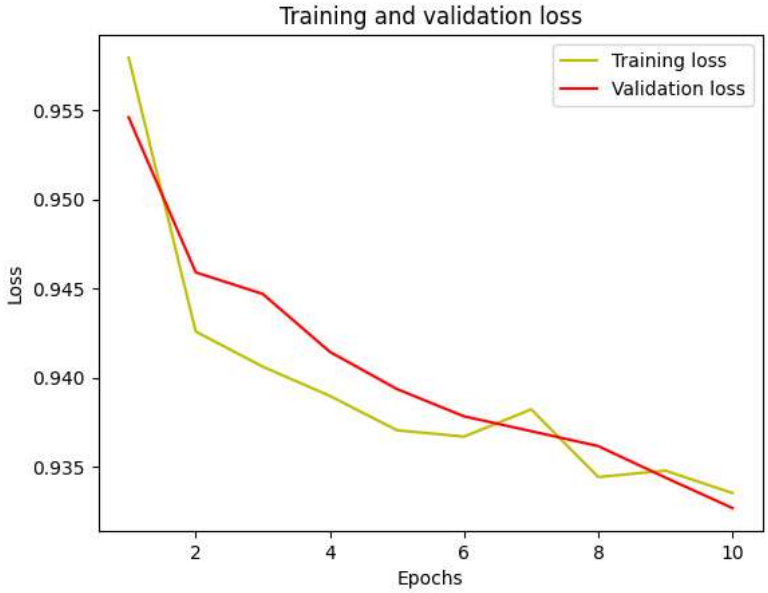
```
Epoch 1/10
129/129 [==============================] - 575s 4s/step - loss: 0.9606 - accuracy: 0.8740 - iou_score: 0.1918 - val_loss: 0.9512 -
Epoch 2/10
129/129 [==============================] - 497s 4s/step - loss: 0.9438 - accuracy: 0.9517 - iou_score: 0.2272 - val_loss: 0.9471 -
Epoch 3/10
129/129 [==============================] - 469s 4s/step - loss: 0.9421 - accuracy: 0.9517 - iou_score: 0.2274 - val_loss: 0.9419 -
Epoch 4/10
129/129 [==============================] - 468s 4s/step - loss: 0.9408 - accuracy: 0.9492 - iou_score: 0.2262 - val_loss: 0.9405 -
Epoch 5/10
129/129 [==============================] - 477s 4s/step - loss: 0.9393 - accuracy: 0.9478 - iou_score: 0.2325 - val_loss: 0.9392 -
Epoch 6/10
129/129 [==============================] - 493s 4s/step - loss: 0.9385 - accuracy: 0.9431 - iou_score: 0.2403 - val_loss: 0.9397 -
Epoch 7/10
129/129 [==============================] - 479s 4s/step - loss: 0.9383 - accuracy: 0.9470 - iou_score: 0.2502 - val_loss: 0.9365 -
Epoch 8/10
129/129 [==============================] - 501s 4s/step - loss: 0.9358 - accuracy: 0.9430 - iou_score: 0.2566 - val_loss: 0.9379 -
Epoch 9/10
129/129 [==============================] - 491s 4s/step - loss: 0.9357 - accuracy: 0.9417 - iou_score: 0.2531 - val_loss: 0.9357 -
Epoch 10/10
129/129 [==============================] - 479s 4s/step - loss: 0.9343 - accuracy: 0.9410 - iou_score: 0.2527 - val_loss: 0.9359 -
```

```
history0=model.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=10,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )

model.save('/content/drive/MyDrive/archive (11)/1_newbrats_3d.hdf5')
```

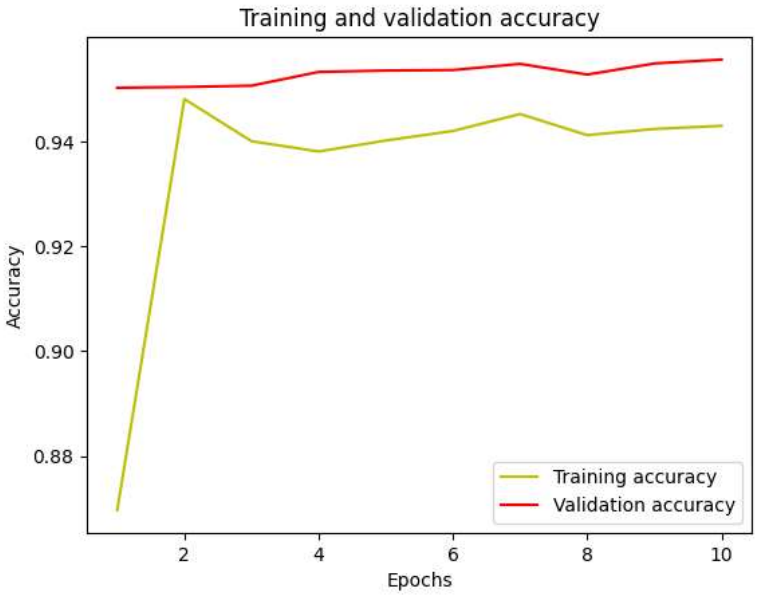
```
Epoch 1/10
129/129 [==============================] - 606s 5s/step - loss: 0.9579 - accuracy: 0.8697 - iou_score: 0.1984 - val_loss: 0.9546 -
Epoch 2/10
129/129 [==============================] - 540s 4s/step - loss: 0.9426 - accuracy: 0.9481 - iou_score: 0.2287 - val_loss: 0.9459 -
Epoch 3/10
129/129 [==============================] - 507s 4s/step - loss: 0.9406 - accuracy: 0.9400 - iou_score: 0.2301 - val_loss: 0.9447 -
Epoch 4/10
129/129 [==============================] - 509s 4s/step - loss: 0.9390 - accuracy: 0.9381 - iou_score: 0.2417 - val_loss: 0.9414 -
Epoch 5/10
129/129 [==============================] - 518s 4s/step - loss: 0.9370 - accuracy: 0.9401 - iou_score: 0.2485 - val_loss: 0.9394 -
Epoch 6/10
129/129 [==============================] - 490s 4s/step - loss: 0.9367 - accuracy: 0.9420 - iou_score: 0.2521 - val_loss: 0.9378 -
Epoch 7/10
129/129 [==============================] - 499s 4s/step - loss: 0.9382 - accuracy: 0.9452 - iou_score: 0.2586 - val_loss: 0.9370 -
Epoch 8/10
129/129 [==============================] - 486s 4s/step - loss: 0.9344 - accuracy: 0.9412 - iou_score: 0.2592 - val_loss: 0.9362 -
Epoch 9/10
129/129 [==============================] - 517s 4s/step - loss: 0.9348 - accuracy: 0.9423 - iou_score: 0.2628 - val_loss: 0.9344 -
Epoch 10/10
129/129 [==============================] - 508s 4s/step - loss: 0.9335 - accuracy: 0.9429 - iou_score: 0.2698 - val_loss: 0.9327 -
```

```
loss = history0.history['loss']
val_loss = history0.history['val_loss']
```

```
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
acc = history0.history['accuracy']
val_acc = history0.history['val_accuracy']

plt.plot(epochs, acc, 'y', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```




```
from keras.models import load_model
my_modelc = load_model('/content/drive/MyDrive/archive (11)/1_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})

#Now all set to continue the training process.
history_c=my_modelc.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=7,
          verbose=1,
          validation_data=val_img_datagen,
```

```
          validation_steps=val_steps_per_epoch,
          )
my_modelc.save('/content/drive/MyDrive/archive (11)/cnew_newbrats_3d.hdf5')
```

```
Epoch 1/7
129/129 [==============================] - 711s 5s/step - loss: 0.9317 - accuracy: 0.9435 - iou_score: 0.2762 - val_loss: 0.9318 -
Epoch 2/7
129/129 [==============================] - 522s 4s/step - loss: 0.9314 - accuracy: 0.9412 - iou_score: 0.2730 - val_loss: 0.9303 -
Epoch 3/7
129/129 [==============================] - 618s 5s/step - loss: 0.9303 - accuracy: 0.9414 - iou_score: 0.2801 - val_loss: 0.9293 -
Epoch 4/7
129/129 [==============================] - 523s 4s/step - loss: 0.9295 - accuracy: 0.9401 - iou_score: 0.2770 - val_loss: 0.9274 -
Epoch 5/7
129/129 [==============================] - 510s 4s/step - loss: 0.9292 - accuracy: 0.9399 - iou_score: 0.2795 - val_loss: 0.9269 -
Epoch 6/7
129/129 [==============================] - 513s 4s/step - loss: 0.9277 - accuracy: 0.9398 - iou_score: 0.2839 - val_loss: 0.9268 -
Epoch 7/7
129/129 [==============================] - 511s 4s/step - loss: 0.9280 - accuracy: 0.9400 - iou_score: 0.2836 - val_loss: 0.9259 -
```

```
from keras.models import load_model
my_model = load_model('/content/drive/MyDrive/archive (11)/1_newbrats_3d.hdf5',
                      compile=False)


from keras.models import load_model
my_modeld = load_model('/content/drive/MyDrive/archive (11)/cnew_newbrats_3d.hdf5',
                      compile=False)


from keras.models import load_model
my_modeld = load_model('/content/drive/MyDrive/archive (11)/cnew_newbrats_3d.hdf5',
                      custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                      'iou_score':sm.metrics.IOUScore(threshold=0.5)})

#Now all set to continue the training process.
history_d=my_modeld.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=7,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modeld.save('/content/drive/MyDrive/archive (11)/dnew_newbrats_3d.hdf5')
```

```
Epoch 1/7
129/129 [==============================] - 618s 5s/step - loss: 0.9302 - accuracy: 0.9403 - iou_score: 0.2841 - val_loss: 0.9265 -
Epoch 2/7
129/129 [==============================] - 649s 5s/step - loss: 0.9289 - accuracy: 0.9393 - iou_score: 0.2863 - val_loss: 0.9245 -
Epoch 3/7
129/129 [==============================] - 658s 5s/step - loss: 0.9274 - accuracy: 0.9374 - iou_score: 0.2876 - val_loss: 0.9227 -
Epoch 4/7
129/129 [==============================] - 699s 5s/step - loss: 0.9284 - accuracy: 0.9389 - iou_score: 0.2887 - val_loss: 0.9241 -
Epoch 5/7
129/129 [==============================] - 657s 5s/step - loss: 0.9275 - accuracy: 0.9397 - iou_score: 0.2892 - val_loss: 0.9235 -
Epoch 6/7
129/129 [==============================] - 649s 5s/step - loss: 0.9275 - accuracy: 0.9406 - iou_score: 0.2881 - val_loss: 0.9269 -
Epoch 7/7
129/129 [==============================] - 653s 5s/step - loss: 0.9264 - accuracy: 0.9360 - iou_score: 0.2878 - val_loss: 0.9213 -
```

```
from keras.models import load_model
my_modelf = load_model('/content/drive/MyDrive/archive (11)/dnew_newbrats_3d.hdf5',
                      custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                      'iou_score':sm.metrics.IOUScore(threshold=0.5)})

#Now all set to continue the training process.
history_f=my_modelf.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=15,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelf.save('/content/drive/MyDrive/archive (11)/fnew_newbrats_3d.hdf5')
```

```
Epoch 1/15
129/129 [==============================] - 534s 4s/step - loss: 0.9261 - accuracy: 0.9393 - iou_score: 0.2884 - val_loss: 0.9234 -
Epoch 2/15
129/129 [==============================] - 460s 4s/step - loss: 0.9245 - accuracy: 0.9354 - iou_score: 0.2898 - val_loss: 0.9216 -
Epoch 3/15
129/129 [==============================] - 467s 4s/step - loss: 0.9243 - accuracy: 0.9370 - iou_score: 0.2910 - val_loss: 0.9247 -
Epoch 4/15
129/129 [==============================] - 481s 4s/step - loss: 0.9249 - accuracy: 0.9368 - iou_score: 0.2936 - val_loss: 0.9265 -
```

```
Epoch 5/15
129/129 [==============================] - 462s 4s/step - loss: 0.9255 - accuracy: 0.9330 - iou_score: 0.2911 - val_loss: 0.9220 -
Epoch 6/15
129/129 [==============================] - 433s 3s/step - loss: 0.9235 - accuracy: 0.9341 - iou_score: 0.3015 - val_loss: 0.9227 -
Epoch 7/15
129/129 [==============================] - 433s 3s/step - loss: 0.9190 - accuracy: 0.9347 - iou_score: 0.3185 - val_loss: 0.9250 -
Epoch 8/15
129/129 [==============================] - 471s 4s/step - loss: 0.9185 - accuracy: 0.9319 - iou_score: 0.3220 - val_loss: 0.9213 -
Epoch 9/15
129/129 [==============================] - 440s 3s/step - loss: 0.9166 - accuracy: 0.9312 - iou_score: 0.3310 - val_loss: 0.9178 -
Epoch 10/15
129/129 [==============================] - 447s 3s/step - loss: 0.9146 - accuracy: 0.9335 - iou_score: 0.3368 - val_loss: 0.9186 -
Epoch 11/15
129/129 [==============================] - 437s 3s/step - loss: 0.9129 - accuracy: 0.9332 - iou_score: 0.3430 - val_loss: 0.9163 -
Epoch 12/15
129/129 [==============================] - 483s 4s/step - loss: 0.9129 - accuracy: 0.9302 - iou_score: 0.3432 - val_loss: 0.9107 -
Epoch 13/15
129/129 [==============================] - 442s 3s/step - loss: 0.9098 - accuracy: 0.9329 - iou_score: 0.3532 - val_loss: 0.9139 -
Epoch 14/15
129/129 [==============================] - 462s 4s/step - loss: 0.9097 - accuracy: 0.9307 - iou_score: 0.3520 - val_loss: 0.9154 -
Epoch 15/15
129/129 [==============================] - 541s 4s/step - loss: 0.9093 - accuracy: 0.9342 - iou_score: 0.3588 - val_loss: 0.9172 -
```

```
from keras.models import load_model
my_modelg = load_model('/content/drive/MyDrive/archive (11)/fnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})

#Now all set to continue the training process.
history_g=my_modelg.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=6,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelg.save('/content/drive/MyDrive/archive (11)/gnew_newbrats_3d.hdf5')
```

```
Epoch 1/6
129/129 [==============================] - 753s 6s/step - loss: 0.9078 - accuracy: 0.9309 - iou_score: 0.3591 - val_loss: 0.9144 -
Epoch 2/6
129/129 [==============================] - 819s 6s/step - loss: 0.9067 - accuracy: 0.9320 - iou_score: 0.3653 - val_loss: 0.9133 -
Epoch 3/6
129/129 [==============================] - 754s 6s/step - loss: 0.9071 - accuracy: 0.9327 - iou_score: 0.3652 - val_loss: 0.9160 -
Epoch 4/6
129/129 [==============================] - 761s 6s/step - loss: 0.9056 - accuracy: 0.9336 - iou_score: 0.3665 - val_loss: 0.9165 -
Epoch 5/6
129/129 [==============================] - 840s 7s/step - loss: 0.9041 - accuracy: 0.9330 - iou_score: 0.3729 - val_loss: 0.9124 -
Epoch 6/6
129/129 [==============================] - 808s 6s/step - loss: 0.9030 - accuracy: 0.9334 - iou_score: 0.3799 - val_loss: 0.9130 -
```

```
from keras.models import load_model
my_modelh = load_model('/content/drive/MyDrive/archive (11)/gnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})

#Now all set to continue the training process.
history_h=my_modelh.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=15,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelh.save('/content/drive/MyDrive/archive (11)/hnew_newbrats_3d.hdf5')
```

```
Epoch 1/15
129/129 [==============================] - 581s 4s/step - loss: 0.9013 - accuracy: 0.9314 - iou_score: 0.3814 - val_loss: 0.9138 -
Epoch 2/15
129/129 [==============================] - 492s 4s/step - loss: 0.9001 - accuracy: 0.9338 - iou_score: 0.3864 - val_loss: 0.9135 -
Epoch 3/15
129/129 [==============================] - 457s 4s/step - loss: 0.8988 - accuracy: 0.9355 - iou_score: 0.3887 - val_loss: 0.9113 -
Epoch 4/15
129/129 [==============================] - 461s 4s/step - loss: 0.8993 - accuracy: 0.9322 - iou_score: 0.3885 - val_loss: 0.9158 -
Epoch 5/15
129/129 [==============================] - 450s 4s/step - loss: 0.8964 - accuracy: 0.9338 - iou_score: 0.3978 - val_loss: 0.9093 -
Epoch 6/15
129/129 [==============================] - 471s 4s/step - loss: 0.8967 - accuracy: 0.9329 - iou_score: 0.3959 - val_loss: 0.9147 -
Epoch 7/15
129/129 [==============================] - 476s 4s/step - loss: 0.8958 - accuracy: 0.9325 - iou_score: 0.4002 - val_loss: 0.9116 -
Epoch 8/15
129/129 [==============================] - 481s 4s/step - loss: 0.8915 - accuracy: 0.9335 - iou_score: 0.4129 - val_loss: 0.9149 -
Epoch 9/15
```

```
129/129 [==============================] - 470s 4s/step - loss: 0.8913 - accuracy: 0.9337 - iou_score: 0.4131 - val_loss: 0.9115 -
Epoch 10/15
129/129 [==============================] - 486s 4s/step - loss: 0.8901 - accuracy: 0.9347 - iou_score: 0.4169 - val_loss: 0.9203 -
Epoch 11/15
129/129 [==============================] - 473s 4s/step - loss: 0.8856 - accuracy: 0.9372 - iou_score: 0.4306 - val_loss: 0.9229 -
Epoch 12/15
129/129 [==============================] - 456s 4s/step - loss: 0.8843 - accuracy: 0.9390 - iou_score: 0.4351 - val_loss: 0.9169 -
Epoch 13/15
129/129 [==============================] - 479s 4s/step - loss: 0.8875 - accuracy: 0.9351 - iou_score: 0.4274 - val_loss: 0.9107 -
Epoch 14/15
129/129 [==============================] - 483s 4s/step - loss: 0.8801 - accuracy: 0.9427 - iou_score: 0.4464 - val_loss: 0.9179 -
Epoch 15/15
129/129 [==============================] - 489s 4s/step - loss: 0.8818 - accuracy: 0.9409 - iou_score: 0.4412 - val_loss: 0.9211 -
```

```
from keras.models import load_model
my_modeli = load_model('/content/drive/MyDrive/archive (11)/hnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})

#Now all set to continue the training process.
history_h=my_modeli.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=5,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modeli.save('/content/drive/MyDrive/archive (11)/inew_newbrats_3d.hdf5')
```

```
Epoch 1/5
129/129 [==============================] - 603s 5s/step - loss: 0.8844 - accuracy: 0.9399 - iou_score: 0.4319 - val_loss: 0.9232 -
Epoch 2/5
129/129 [==============================] - 632s 5s/step - loss: 0.8748 - accuracy: 0.9467 - iou_score: 0.4636 - val_loss: 0.9207 -
Epoch 3/5
129/129 [==============================] - 641s 5s/step - loss: 0.8743 - accuracy: 0.9476 - iou_score: 0.4634 - val_loss: 0.9315 -
Epoch 4/5
129/129 [==============================] - 613s 5s/step - loss: 0.8744 - accuracy: 0.9468 - iou_score: 0.4645 - val_loss: 0.9336 -
Epoch 5/5
129/129 [==============================] - 611s 5s/step - loss: 0.8746 - accuracy: 0.9478 - iou_score: 0.4618 - val_loss: 0.9293 -
```

```
from keras.models import load_model
my_modeler = load_model('/content/drive/MyDrive/archive (11)/gnew_newbrats_3d.hdf5',
                       compile=False)
```

```
from keras.models import load_model
my_modelk = load_model('/content/drive/MyDrive/archive (11)/gnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_k=my_modelk.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=15,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelk.save('/content/drive/MyDrive/archive (11)/gnew_newbrats_3d.hdf5')
```

```
Epoch 1/15
129/129 [==============================] - 568s 4s/step - loss: 0.9012 - accuracy: 0.9318 - iou_score: 0.3819 - val_loss: 0.9148 -
Epoch 2/15
129/129 [==============================] - 476s 4s/step - loss: 0.9008 - accuracy: 0.9330 - iou_score: 0.3854 - val_loss: 0.9167 -
Epoch 3/15
129/129 [==============================] - 488s 4s/step - loss: 0.8993 - accuracy: 0.9335 - iou_score: 0.3885 - val_loss: 0.9140 -
Epoch 4/15
129/129 [==============================] - 507s 4s/step - loss: 0.8982 - accuracy: 0.9330 - iou_score: 0.3910 - val_loss: 0.9135 -
Epoch 5/15
129/129 [==============================] - 505s 4s/step - loss: 0.8973 - accuracy: 0.9332 - iou_score: 0.3950 - val_loss: 0.9179 -
Epoch 6/15
129/129 [==============================] - 495s 4s/step - loss: 0.8936 - accuracy: 0.9368 - iou_score: 0.4066 - val_loss: 0.9143 -
Epoch 7/15
129/129 [==============================] - 469s 4s/step - loss: 0.8937 - accuracy: 0.9347 - iou_score: 0.4043 - val_loss: 0.9154 -
Epoch 8/15
129/129 [==============================] - 493s 4s/step - loss: 0.8910 - accuracy: 0.9353 - iou_score: 0.4129 - val_loss: 0.9170 -
Epoch 9/15
129/129 [==============================] - 475s 4s/step - loss: 0.8953 - accuracy: 0.9312 - iou_score: 0.4004 - val_loss: 0.9236 -
Epoch 10/15
129/129 [==============================] - 477s 4s/step - loss: 0.8916 - accuracy: 0.9349 - iou_score: 0.4122 - val_loss: 0.9121 -
Epoch 11/15
129/129 [==============================] - 464s 4s/step - loss: 0.8882 - accuracy: 0.9334 - iou_score: 0.4250 - val_loss: 0.9155 -
Epoch 12/15
129/129 [==============================] - 503s 4s/step - loss: 0.8864 - accuracy: 0.9354 - iou_score: 0.4302 - val_loss: 0.9191 -
Epoch 13/15
```

```
129/129 [==============================] - 492s 4s/step - loss: 0.8845 - accuracy: 0.9361 - iou_score: 0.4356 - val_loss: 0.9191 -
Epoch 14/15
129/129 [==============================] - 470s 4s/step - loss: 0.8837 - accuracy: 0.9397 - iou_score: 0.4390 - val_loss: 0.9228 -
Epoch 15/15
129/129 [==============================] - 491s 4s/step - loss: 0.8837 - accuracy: 0.9404 - iou_score: 0.4372 - val_loss: 0.9166 -
```

```
from keras.models import load_model
my_modely = load_model('/content/drive/MyDrive/archive (11)/fnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_y=my_modely.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=6,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modely.save('/content/drive/MyDrive/archive (11)/yfnew_newbrats_3d.hdf5')
```

```
Epoch 1/6
129/129 [==============================] - 674s 5s/step - loss: 0.9069 - accuracy: 0.9313 - iou_score: 0.3620 - val_loss: 0.9138 -
Epoch 2/6
129/129 [==============================] - 680s 5s/step - loss: 0.9068 - accuracy: 0.9315 - iou_score: 0.3649 - val_loss: 0.9161 -
Epoch 3/6
129/129 [==============================] - 719s 6s/step - loss: 0.9075 - accuracy: 0.9342 - iou_score: 0.3645 - val_loss: 0.9153 -
Epoch 4/6
129/129 [==============================] - 715s 6s/step - loss: 0.9050 - accuracy: 0.9356 - iou_score: 0.3701 - val_loss: 0.9202 -
Epoch 5/6
129/129 [==============================] - 743s 6s/step - loss: 0.9037 - accuracy: 0.9326 - iou_score: 0.3740 - val_loss: 0.9194 -
Epoch 6/6
129/129 [==============================] - 759s 6s/step - loss: 0.9029 - accuracy: 0.9328 - iou_score: 0.3771 - val_loss: 0.9157 -
```

```
from keras.models import load_model
my_modelv = load_model('/content/drive/MyDrive/archive (11)/yfnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_v=my_modelv.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=6,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelv.save('/content/drive/MyDrive/archive (11)/vfnew_newbrats_3d.hdf5')
```

```
Epoch 1/6
129/129 [==============================] - 746s 6s/step - loss: 0.9013 - accuracy: 0.9333 - iou_score: 0.3839 - val_loss: 0.9148 -
Epoch 2/6
129/129 [==============================] - 449s 3s/step - loss: 0.9011 - accuracy: 0.9321 - iou_score: 0.3828 - val_loss: 0.9145 -
Epoch 3/6
129/129 [==============================] - 593s 5s/step - loss: 0.9011 - accuracy: 0.9306 - iou_score: 0.3859 - val_loss: 0.9202 -
Epoch 4/6
129/129 [==============================] - 544s 4s/step - loss: 0.9013 - accuracy: 0.9308 - iou_score: 0.3862 - val_loss: 0.9171 -
Epoch 5/6
129/129 [==============================] - 460s 4s/step - loss: 0.8997 - accuracy: 0.9306 - iou_score: 0.3895 - val_loss: 0.9109 -
Epoch 6/6
129/129 [==============================] - 549s 4s/step - loss: 0.8971 - accuracy: 0.9345 - iou_score: 0.3986 - val_loss: 0.9104 -
```

```
from keras.models import load_model
my_modelu = load_model('/content/drive/MyDrive/archive (11)/vfnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_u=my_modelu.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=6,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelu.save('/content/drive/MyDrive/archive (11)/ufnew_newbrats_3d.hdf5')
# this has mean iou of 51
```

```
Epoch 1/6
129/129 [==============================] - 667s 5s/step - loss: 0.8957 - accuracy: 0.9371 - iou_score: 0.4010 - val_loss: 0.9118 -
Epoch 2/6
129/129 [==============================] - 682s 5s/step - loss: 0.8967 - accuracy: 0.9360 - iou_score: 0.3971 - val_loss: 0.9037 -
Epoch 3/6
129/129 [==============================] - 655s 5s/step - loss: 0.8913 - accuracy: 0.9363 - iou_score: 0.4138 - val_loss: 0.9029 -
Epoch 4/6
```

```
129/129 [==============================] - 687s 5s/step - loss: 0.8892 - accuracy: 0.9382 - iou_score: 0.4186 - val_loss: 0.9084 -
Epoch 5/6
129/129 [==============================] - 661s 5s/step - loss: 0.8897 - accuracy: 0.9350 - iou_score: 0.4174 - val_loss: 0.9080 -
Epoch 6/6
129/129 [==============================] - 722s 6s/step - loss: 0.8901 - accuracy: 0.9362 - iou_score: 0.4168 - val_loss: 0.8996 -
```

```
from keras.models import load_model
my_modelt = load_model('/content/drive/MyDrive/archive (11)/ufnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_t=my_modelt.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=6,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelt.save('/content/drive/MyDrive/archive (11)/tfnew_newbrats_3d.hdf5')
```

```
Epoch 1/6
129/129 [==============================] - 1092s 8s/step - loss: 0.8825 - accuracy: 0.9416 - iou_score: 0.4404 - val_loss: 0.9037 -
Epoch 2/6
129/129 [==============================] - 646s 5s/step - loss: 0.8824 - accuracy: 0.9419 - iou_score: 0.4387 - val_loss: 0.9101 -
Epoch 3/6
129/129 [==============================] - 679s 5s/step - loss: 0.8790 - accuracy: 0.9436 - iou_score: 0.4508 - val_loss: 0.9134 -
Epoch 4/6
129/129 [==============================] - 628s 5s/step - loss: 0.8782 - accuracy: 0.9442 - iou_score: 0.4517 - val_loss: 0.9025 -
Epoch 5/6
129/129 [==============================] - 683s 5s/step - loss: 0.8783 - accuracy: 0.9475 - iou_score: 0.4532 - val_loss: 0.9084 -
Epoch 6/6
129/129 [==============================] - 632s 5s/step - loss: 0.8740 - accuracy: 0.9507 - iou_score: 0.4634 - val_loss: 0.9172 -
```

```
from keras.models import load_model
my_modelp = load_model('/content/drive/MyDrive/archive (11)/tfnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_p=my_modelp.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=1,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelp.save('/content/drive/MyDrive/archive (11)/pfnew_newbrats_3d.hdf5')
```

```
129/129 [==============================] - 552s 4s/step - loss: 0.8735 - accuracy: 0.9503 - iou_score: 0.4662 - val_loss: 0.9051 -
```

```
from keras.models import load_model
my_modelp1 = load_model('/content/drive/MyDrive/archive (11)/pfnew_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_p1=my_modelp1.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=2,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelp1.save('/content/drive/MyDrive/archive (11)/pf1new_newbrats_3d.hdf5')
```

```
Epoch 1/2
129/129 [==============================] - 811s 6s/step - loss: 0.8686 - accuracy: 0.9553 - iou_score: 0.4822 - val_loss: 0.9119 -
Epoch 2/2
129/129 [==============================] - 655s 5s/step - loss: 0.8678 - accuracy: 0.9548 - iou_score: 0.4843 - val_loss: 0.9139 -
```

```
from keras.models import load_model
my_modelp3 = load_model('/content/drive/MyDrive/archive (11)/pf1new_newbrats_3d.hdf5',
                       custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                       'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_p3=my_modelp3.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=10,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelp3.save('/content/drive/MyDrive/archive (11)/pf3new_newbrats_3d.hdf5')
```

```
my_modelp3.save('/content/drive/MyDrive/archive (11)/pf3new_newbrats_3d.hdf5')
# 58 mean iou
```

```
Epoch 1/10
129/129 [==============================] - 873s 7s/step - loss: 0.8718 - accuracy: 0.9527 - iou_score: 0.4727 - val_loss: 0.8876 -
Epoch 2/10
129/129 [==============================] - 841s 7s/step - loss: 0.8621 - accuracy: 0.9594 - iou_score: 0.5000 - val_loss: 0.8758 -
Epoch 3/10
129/129 [==============================] - 940s 7s/step - loss: 0.8610 - accuracy: 0.9595 - iou_score: 0.5045 - val_loss: 0.8744 -
Epoch 4/10
129/129 [==============================] - 832s 6s/step - loss: 0.8585 - accuracy: 0.9616 - iou_score: 0.5116 - val_loss: 0.8813 -
Epoch 5/10
129/129 [==============================] - 878s 7s/step - loss: 0.8575 - accuracy: 0.9627 - iou_score: 0.5134 - val_loss: 0.8780 -
Epoch 6/10
129/129 [==============================] - 725s 6s/step - loss: 0.8552 - accuracy: 0.9630 - iou_score: 0.5224 - val_loss: 0.8778 -
Epoch 7/10
129/129 [==============================] - 769s 6s/step - loss: 0.8548 - accuracy: 0.9638 - iou_score: 0.5221 - val_loss: 0.8801 -
Epoch 8/10
129/129 [==============================] - 847s 7s/step - loss: 0.8531 - accuracy: 0.9644 - iou_score: 0.5265 - val_loss: 0.8840 -
Epoch 9/10
129/129 [==============================] - 756s 6s/step - loss: 0.8519 - accuracy: 0.9657 - iou_score: 0.5306 - val_loss: 0.8726 -
Epoch 10/10
129/129 [==============================] - 830s 6s/step - loss: 0.8511 - accuracy: 0.9645 - iou_score: 0.5345 - val_loss: 0.8792 -
```

```
from keras.metrics import MeanIoU

batch_size=8 #Check IoU for a batch of images
test_img_datagen = imageLoader(val_img_dir, val_img_list,
                               val_mask_dir, val_mask_list, batch_size)

#Verify generator.... In python 3 next() is renamed as __next__()
test_image_batch, test_mask_batch = test_img_datagen.__next__()

test_mask_batch_argmax = np.argmax(test_mask_batch, axis=4)
test_pred_batch = my_modelp3.predict(test_image_batch)
test_pred_batch_argmax = np.argmax(test_pred_batch, axis=4)

n_classes = 4
IOU_keras = MeanIoU(num_classes=n_classes)
IOU_keras.update_state(test_pred_batch_argmax, test_mask_batch_argmax)
print("Mean IoU =", IOU_keras.result().numpy())
```

```
1/1 [==============================] - 1s 724ms/step
Mean IoU = 0.58479655
```

```
from keras.models import load_model
my_modelp8 = load_model('/content/drive/MyDrive/archive (11)/pf3new_newbrats_3d.hdf5',
                    custom_objects={'dice_loss_plus_1focal_loss': total_loss,
                                    'iou_score':sm.metrics.IOUScore(threshold=0.5)})
history_p8=my_modelp8.fit(train_img_datagen,
          steps_per_epoch=steps_per_epoch,
          epochs=10,
          verbose=1,
          validation_data=val_img_datagen,
          validation_steps=val_steps_per_epoch,
          )
my_modelp8.save('/content/drive/MyDrive/archive (11)/pf8new_newbrats_3d.hdf5')
```

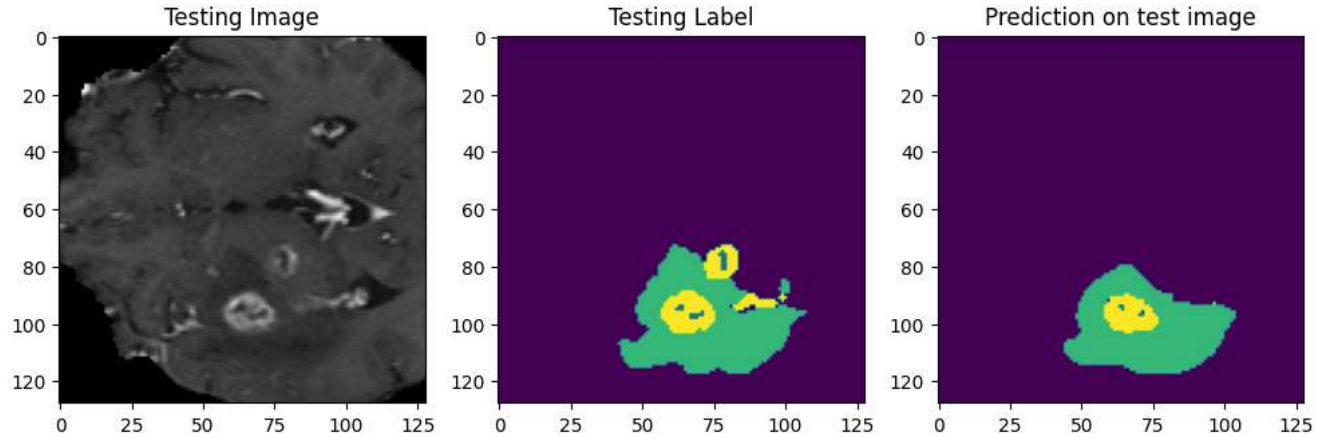
```
Epoch 1/10
129/129 [==============================] - 970s 7s/step - loss: 0.8473 - accuracy: 0.9674 - iou_score: 0.5469 - val_loss: 0.9202 -
Epoch 2/10
129/129 [==============================] - 652s 5s/step - loss: 0.8491 - accuracy: 0.9661 - iou_score: 0.5419 - val_loss: 0.9213 -
Epoch 3/10
129/129 [==============================] - 650s 5s/step - loss: 0.8458 - accuracy: 0.9681 - iou_score: 0.5510 - val_loss: 0.9195 -
Epoch 4/10
129/129 [==============================] - 650s 5s/step - loss: 0.8457 - accuracy: 0.9674 - iou_score: 0.5518 - val_loss: 0.9159 -
Epoch 5/10
129/129 [==============================] - 690s 5s/step - loss: 0.8442 - accuracy: 0.9677 - iou_score: 0.5566 - val_loss: 0.9205 -
Epoch 6/10
129/129 [==============================] - 650s 5s/step - loss: 0.8446 - accuracy: 0.9685 - iou_score: 0.5552 - val_loss: 0.9149 -
Epoch 7/10
129/129 [==============================] - 652s 5s/step - loss: 0.8426 - accuracy: 0.9694 - iou_score: 0.5611 - val_loss: 0.9174 -
Epoch 8/10
129/129 [==============================] - 654s 5s/step - loss: 0.8405 - accuracy: 0.9704 - iou_score: 0.5678 - val_loss: 0.9089 -
Epoch 9/10
129/129 [==============================] - 653s 5s/step - loss: 0.8376 - accuracy: 0.9716 - iou_score: 0.5783 - val_loss: 0.9204 -
Epoch 10/10
129/129 [==============================] - 646s 5s/step - loss: 0.8379 - accuracy: 0.9712 - iou_score: 0.5779 - val_loss: 0.9119 -
```

```
from keras.metrics import MeanIoU

batch_size=8 #Check IoU for a batch of images
test_img_datagen = imageLoader(val_img_dir, val_img_list,
                               val_mask_dir, val_mask_list, batch_size)

#Verify generator.... In python 3 next() is renamed as __next__()
test_image_batch, test_mask_batch = test_img_datagen.__next__()

test_mask_batch_argmax = np.argmax(test_mask_batch, axis=4)
test_pred_batch = my_modelp8.predict(test_image_batch)
test_pred_batch_argmax = np.argmax(test_pred_batch, axis=4)

n_classes = 4
IOU_keras = MeanIoU(num_classes=n_classes)
IOU_keras.update_state(test_pred_batch_argmax, test_mask_batch_argmax)
print("Mean IoU =", IOU_keras.result().numpy())
```

```
1/1 [==============================] - 18s 18s/step
Mean IoU = 0.5116647
```

```
img_num = 100

test_img = np.load("/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/val/images/image_"+str(img_num)+".npy")

test_mask = np.load("/content/drive/MyDrive/archive (11)/BraTS2020_TrainingData/input_data_128/val/masks/mask_"+str(img_num)+".npy")
test_mask_argmax=np.argmax(test_mask, axis=3)

test_img_input = np.expand_dims(test_img, axis=0)
test_prediction = my_modelu.predict(test_img_input)
test_prediction_argmax=np.argmax(test_prediction, axis=4)[0,:,:,:]


# print(test_prediction_argmax.shape)
# print(test_mask_argmax.shape)
# print(np.unique(test_prediction_argmax))


#Plot individual slices from test predictions for verification
from matplotlib import pyplot as plt
import random

#n_slice=random.randint(0, test_prediction_argmax.shape[2])
n_slice = 55
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img[:,:,n_slice,1], cmap='gray')
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(test_mask_argmax[:,:,n_slice])
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(test_prediction_argmax[:,:, n_slice])
plt.show()
```

```
1/1 [==============================] - 3s 3s/step
```




```
from keras.models import load_model
my_modeler = load_model('/content/drive/MyDrive/archive (11)/ufnew_newbrats_3d.hdf5',
                        compile=False)
```