3d-u-net

October 25, 2023

[1]:

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
# for dirname, _, filenames in os.walk('/kaggle/input'):
#     for filename in filenames:
#         print(os.path.join(dirname, filename))


# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

[2]:

```
pip install nibabel
```

```
Requirement already satisfied: nibabel in /opt/conda/lib/python3.10/site-
packages (5.1.0)
Requirement already satisfied: numpy>=1.19 in /opt/conda/lib/python3.10/site-
packages (from nibabel) (1.23.5)
Requirement already satisfied: packaging>=17 in /opt/conda/lib/python3.10/site-
packages (from nibabel) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from packaging>=17->nibabel) (3.0.9)
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
import nibabel as nib
import glob
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from tifffile import imsave
import random
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[5]:
```
path = '../input/brats20-dataset-training-validation/BraTS2020_TrainingData/
  ↪MICCAI_BraTS2020_TrainingData/'
```

0.0.1 Load Data

[6]:
```
data_t1ce = nib.load(path + 'BraTS20_Training_069/BraTS20_Training_069_t1ce.
  ↪nii').get_fdata()
data_t1ce = minmax.fit_transform(data_t1ce.reshape(-1,data_t1ce.shape[-1])).
  ↪reshape(data_t1ce.shape)
data_t1ce.shape
```

[6]:
```
(240, 240, 155)
```

[7]:
```
data_t2 = nib.load(path + 'BraTS20_Training_069/BraTS20_Training_069_t2.nii').
  ↪get_fdata()
data_t2 = minmax.fit_transform(data_t2.reshape(-1,data_t2.shape[-1])).
  ↪reshape(data_t2.shape)
```

[8]:
```
data_flair = nib.load(path + 'BraTS20_Training_069/BraTS20_Training_069_flair.
  ↪nii').get_fdata()
data_flair = minmax.fit_transform(data_flair.reshape(-1,data_flair.shape[-1])).
  ↪reshape(data_flair.shape)
```

[9]:
```
data_mask = nib.load(path + 'BraTS20_Training_069/BraTS20_Training_069_seg.
  ↪nii').get_fdata()
data_mask=data_mask.astype(np.uint8)
```

[10]:
```
import random
n=random.randint(0, data_mask.shape[2])
```

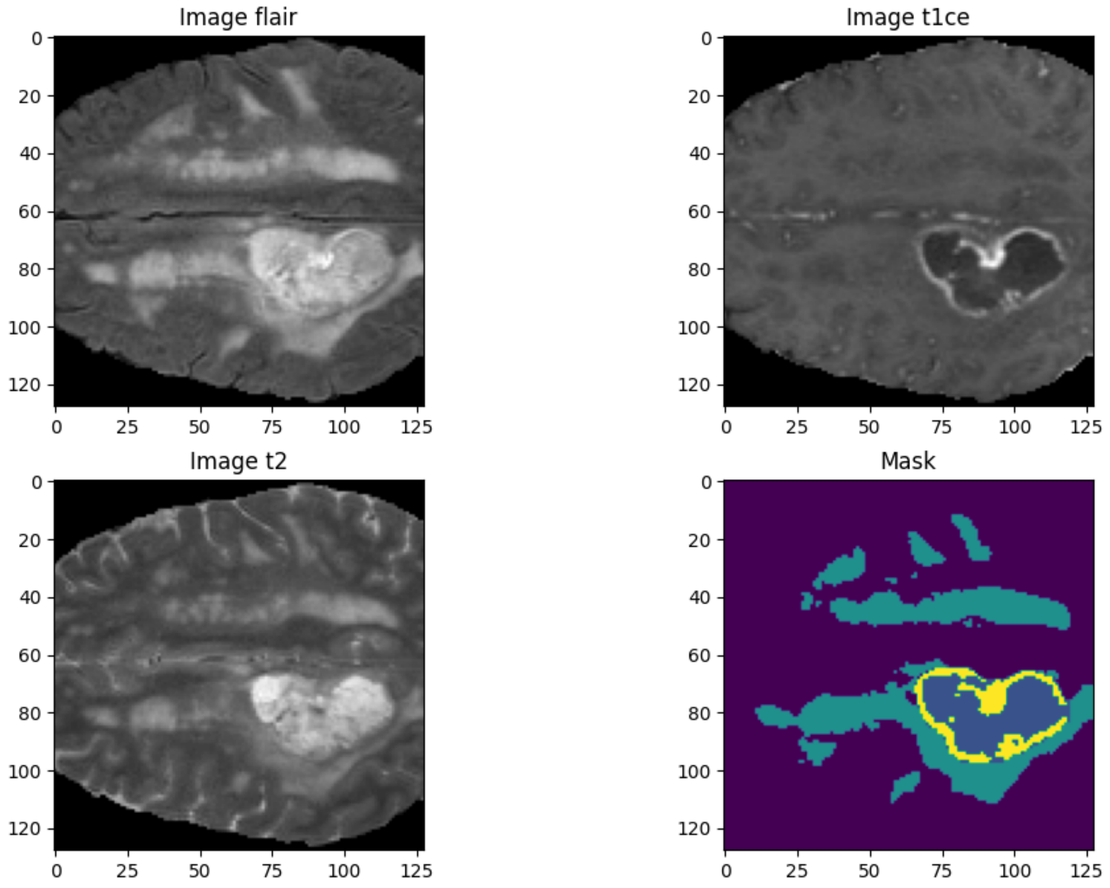
[11]:
```
# Combining 3 different images to a single channel - to explore more
comb = np.stack([data_flair, data_t1ce,data_t2], axis=3)
comb=comb[56:184, 56:184, 13:141] #Crop to 128x128x128x4
#mask_data
data_mask = data_mask[56:184, 56:184, 13:141]
```

Visualizing the data

[13]:

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

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



[14]:

```
#importing datasets and getting ready
```

[15]:
```
t2_list = sorted(glob.glob('../input/brats20-dataset-training-validation/
  ↪BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*t2.nii'))
t1ce_list = sorted(glob.glob('../input/brats20-dataset-training-validation/
  ↪BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*t1ce.nii'))
flair_list = sorted(glob.glob('../input/brats20-dataset-training-validation/
  ↪BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*flair.nii'))
mask_list = sorted(glob.glob('../input/brats20-dataset-training-validation/
  ↪BraTS2020_TrainingData/MICCAI_BraTS2020_TrainingData/*/*seg.nii'))
```

Data preprocessing

[38]:
```
import skimage.transform as skTrans
for img in range(100):
    data_flair = nib.load(flair_list[img]).get_fdata()
    data_flair = skTrans.resize(data_flair,(64,64,64),preserve_range=True)
    data_mask = nib.load(mask_list[img]).get_fdata()
    data_mask = skTrans.resize(data_mask,(64,64,64),preserve_range=True)
    if img==0:
        xtrain = data_flair.reshape((1,64,64,64))
        ytrain = data_mask.reshape((1,64,64,64))
    else:
        xtrain = np.vstack((xtrain,data_flair.reshape((1,64,64,64))))
        ytrain = np.vstack((ytrain,data_mask.reshape((1,64,64,64))))
```

[20]:
```
print(xtrain.shape,ytrain.shape)
print(len(xtrain))
```

```
(100, 64, 64, 64) (100, 64, 64, 64)
100
```

Train Test split

[39]:
```
x_train,x_test = xtrain[:70],xtrain[70:]
y_train,y_test = ytrain[:70],ytrain[70:]
```

0.0.2 Base Model - 3D UNet

[23]:
```
import tensorflow as tf
from tensorflow import keras
```

Function for convolution block

[24]:
```
def conv_block(inp,filters):
    c1 = keras.layers.
  ↪Conv3D(filters,kernel_size=3,padding="same",activation="relu")(inp)
    c2 = keras.layers.
  ↪Conv3D(filters,kernel_size=3,padding="same",activation="relu")(c1)
```

```
    return c2
```

Function for encoder block

[25]:

```
def encoder_block(inp,filters):
    # Consists of the convolution block followed by max pooling
    c = conv_block(inp,filters)
    p = keras.layers.MaxPooling3D(pool_size=(2,2,2))(c)

    return c,p
```

Function for decoder block

[26]:

```
def decoder_block(inp,skip_features,filters):
    #Upsampling to increase the dimension
    u1 = keras.layers.UpSampling3D(size=(2,2,2))(inp)
    # Concatenating the skip connection with upsampled layer
    skip = keras.layers.Concatenate()([u1,skip_features])
    # Performing convolution operation
    c = conv_block(skip,filters)

    return c
```

U-Net Architecture

[27]:

```
# Building the model
inp = keras.Input(shape=(64,64,64,1))

# Encoder block 1
c1,p1 = encoder_block(inp,64)
# Encoder block 2
c2,p2 = encoder_block(p1,128)
# Encoder block 3
c3,p3 = encoder_block(p2,256)
# Encoder block 4
c4,p4 = encoder_block(p3,512)

# Bottleneck layer - gives compressed representation of the raw image
b = conv_block(p4,1024)

# Decoder block 1
d1 = decoder_block(b,c4,512)
# Decoder block 2
d2 = decoder_block(d1,c3,256)
# Decoder block 3
d3 = decoder_block(d2,c2,128)
#Decoder block 4
d4 = decoder_block(d3,c1,64)
```

```
# Output layer
out = keras.layers.Conv3D(1,kernel_size=1,padding="same",activation="relu")(d4)
```

[28]:

```
model = keras.Model(inp,out,name="U-Net")
model.summary()
```

```
Model: "U-Net"
________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
================================================================================
==================
 input_1 (InputLayer)           [(None, 64, 64, 64,  0           []
                                 1)]

 conv3d (Conv3D)                (None, 64, 64, 64,   1792
['input_1[0][0]']
                                64)

 conv3d_1 (Conv3D)              (None, 64, 64, 64,   110656
['conv3d[0][0]']
                                64)

 max_pooling3d (MaxPooling3D)   (None, 32, 32, 32,   0
['conv3d_1[0][0]']
                                64)

 conv3d_2 (Conv3D)              (None, 32, 32, 32,   221312
['max_pooling3d[0][0]']
                                128)

 conv3d_3 (Conv3D)              (None, 32, 32, 32,   442496
['conv3d_2[0][0]']
                                128)

 max_pooling3d_1 (MaxPooling3D)  (None, 16, 16, 16,   0
['conv3d_3[0][0]']
                                128)

 conv3d_4 (Conv3D)              (None, 16, 16, 16,   884992
['max_pooling3d_1[0][0]']
                                256)

 conv3d_5 (Conv3D)              (None, 16, 16, 16,   1769728
['conv3d_4[0][0]']
                                256)
 max_pooling3d_2 (MaxPooling3D)  (None, 8, 8, 8, 256  0
['conv3d_5[0][0]']
                               )

 conv3d_6 (Conv3D)             (None, 8, 8, 8, 512  3539456
['max_pooling3d_2[0][0]']
                               )

 conv3d_7 (Conv3D)             (None, 8, 8, 8, 512  7078400
['conv3d_6[0][0]']
                               )

 max_pooling3d_3 (MaxPooling3D)  (None, 4, 4, 4, 512  0
['conv3d_7[0][0]']
                               )

 conv3d_8 (Conv3D)             (None, 4, 4, 4, 102  14156800
['max_pooling3d_3[0][0]']
                               4)

 conv3d_9 (Conv3D)             (None, 4, 4, 4, 102  28312576
['conv3d_8[0][0]']
                               4)

 up_sampling3d (UpSampling3D)  (None, 8, 8, 8, 102  0
['conv3d_9[0][0]']
                               4)

 concatenate (Concatenate)     (None, 8, 8, 8, 153  0
['up_sampling3d[0][0]',
                               6)
'conv3d_7[0][0]']

 conv3d_10 (Conv3D)            (None, 8, 8, 8, 512  21234176
['concatenate[0][0]']
                               )

 conv3d_11 (Conv3D)            (None, 8, 8, 8, 512  7078400
['conv3d_10[0][0]']
                               )

 up_sampling3d_1 (UpSampling3D)  (None, 16, 16, 16,   0
['conv3d_11[0][0]']
                               512)

 concatenate_1 (Concatenate)   (None, 16, 16, 16,   0
['up_sampling3d_1[0][0]',
                               768)
'conv3d_5[0][0]']

 conv3d_12 (Conv3D)            (None, 16, 16, 16,   5308672
['concatenate_1[0][0]']
                               256)

 conv3d_13 (Conv3D)            (None, 16, 16, 16,   1769728
['conv3d_12[0][0]']
                               256)

 up_sampling3d_2 (UpSampling3D)  (None, 32, 32, 32,   0
['conv3d_13[0][0]']
                               256)

 concatenate_2 (Concatenate)   (None, 32, 32, 32,   0
['up_sampling3d_2[0][0]',
                               384)
'conv3d_3[0][0]']

 conv3d_14 (Conv3D)            (None, 32, 32, 32,   1327232
['concatenate_2[0][0]']
                               128)

 conv3d_15 (Conv3D)            (None, 32, 32, 32,   442496
['conv3d_14[0][0]']
                               128)

 up_sampling3d_3 (UpSampling3D)  (None, 64, 64, 64,   0
['conv3d_15[0][0]']
                               128)

 concatenate_3 (Concatenate)   (None, 64, 64, 64,   0
['up_sampling3d_3[0][0]',
                               192)
'conv3d_1[0][0]']

 conv3d_16 (Conv3D)            (None, 64, 64, 64,   331840
['concatenate_3[0][0]']
                               64)

 conv3d_17 (Conv3D)            (None, 64, 64, 64,   110656
['conv3d_16[0][0]']
                               64)

 conv3d_18 (Conv3D)            (None, 64, 64, 64,   65
['conv3d_17[0][0]']
                               1)
```

```
================================================================================
==================
Total params: 94,121,473
Trainable params: 94,121,473
Non-trainable params: 0
--------------------------------------------------------------------------------
------------------
```

0.0.3 Training the Model

[29]:
```
pip install segmentation-models-3D
```

```
Collecting segmentation-models-3D
  Downloading segmentation_models_3D-1.0.4-py3-none-any.whl (33 kB)
Requirement already satisfied: tensorflow>=2.8.0 in
/opt/conda/lib/python3.10/site-packages (from segmentation-models-3D) (2.12.0)
Collecting keras-applications>=1.0.8 (from segmentation-models-3D)
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
                           50.7/50.7 kB
2.7 MB/s eta 0:00:00
Collecting classification-models-3D>=1.0.6 (from segmentation-models-3D)
  Downloading classification_models_3D-1.0.7-py3-none-any.whl (69 kB)
                           69.5/69.5 kB
9.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.9.1 in
/opt/conda/lib/python3.10/site-packages (from keras-
applications>=1.0.8->segmentation-models-3D) (1.23.5)
Requirement already satisfied: h5py in /opt/conda/lib/python3.10/site-packages
(from keras-applications>=1.0.8->segmentation-models-3D) (3.9.0)
Requirement already satisfied: absl-py>=1.0.0 in /opt/conda/lib/python3.10/site-
packages (from tensorflow>=2.8.0->segmentation-models-3D) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (23.5.26)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (1.51.1)
Requirement already satisfied: jax>=0.3.15 in /opt/conda/lib/python3.10/site-
```

```
packages (from tensorflow>=2.8.0->segmentation-models-3D) (0.4.13)
Requirement already satisfied: keras<2.13,>=2.12.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (2.12.0)
Requirement already satisfied: libclang>=13.0.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (16.0.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (3.3.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.10/site-
packages (from tensorflow>=2.8.0->segmentation-models-3D) (21.3)
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 /opt/conda/lib/python3.10/site-packages (from
tensorflow>=2.8.0->segmentation-models-3D) (3.20.3)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.10/site-
packages (from tensorflow>=2.8.0->segmentation-models-3D) (68.0.0)
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.10/site-
packages (from tensorflow>=2.8.0->segmentation-models-3D) (1.16.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (2.12.3)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (2.12.0)
Requirement already satisfied: termcolor>=1.1.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (4.6.3)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/opt/conda/lib/python3.10/site-packages (from tensorflow>=2.8.0->segmentation-
models-3D) (0.32.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/opt/conda/lib/python3.10/site-packages (from
astunparse>=1.6.0->tensorflow>=2.8.0->segmentation-models-3D) (0.40.0)
Requirement already satisfied: ml-dtypes>=0.1.0 in
/opt/conda/lib/python3.10/site-packages (from
jax>=0.3.15->tensorflow>=2.8.0->segmentation-models-3D) (0.2.0)
Requirement already satisfied: scipy>=1.7 in /opt/conda/lib/python3.10/site-
packages (from jax>=0.3.15->tensorflow>=2.8.0->segmentation-models-3D) (1.11.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-models-3D) (2.20.0)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-models-3D) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in
/opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-models-3D) (3.4.3)
Requirement already satisfied: requests<3,>=2.21.0 in
/opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-models-3D) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-models-3D) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from
tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-models-3D) (2.3.7)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/lib/python3.10/site-packages (from
packaging->tensorflow>=2.8.0->segmentation-models-3D) (3.0.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/opt/conda/lib/python3.10/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/opt/conda/lib/python3.10/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (0.2.7)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/site-
packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (4.9)
Requirement already satisfied: urllib3<2.0 in /opt/conda/lib/python3.10/site-
packages (from google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (1.26.15)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/opt/conda/lib/python3.10/site-packages (from google-auth-
oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.10/site-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
```

```
/opt/conda/lib/python3.10/site-packages (from
requests<3,>=2.21.0->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (2023.7.22)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/opt/conda/lib/python3.10/site-packages (from
werkzeug>=1.0.1->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (2.1.3)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/opt/conda/lib/python3.10/site-packages (from requests-oauthlib>=0.7.0->google-
auth-
oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow>=2.8.0->segmentation-
models-3D) (3.2.2)
Installing collected packages: classification-models-3D, keras-applications,
segmentation-models-3D
Successfully installed classification-models-3D-1.0.7 keras-applications-1.0.8
segmentation-models-3D-1.0.4
Note: you may need to restart the kernel to use updated packages.
```

Setting the optimizer & loss

```
[30]: import segmentation_models_3D as sm
      # Defining the loss function, optimizer& metrics to be used for training
      dice_loss = sm.losses.DiceLoss(class_weights=np.array([0.25,0.25,0.25,0.25]))
      lr = 0.0001
      # Also try using Adam as an optimizer
      optim = keras.optimizers.SGD(learning_rate=lr,momentum=0.35)
```

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

```
[31]: # Fitting the model
      model.compile(optimizer=optim,loss=dice_loss,metrics=['accuracy'])
```

Fitting the model

```
[40]: hist = model.
        ↪fit(x_train,y_train,batch_size=8,epochs=5,validation_data=(x_test,y_test))
```

```
Epoch 1/5
9/9 [==============================] - 45s 5s/step - loss: 0.8763 - accuracy:
0.8248 - val_loss: 0.8823 - val_accuracy: 0.8560
Epoch 2/5
9/9 [==============================] - 43s 5s/step - loss: 0.8588 - accuracy:
0.8775 - val_loss: 0.8677 - val_accuracy: 0.9070
Epoch 3/5
9/9 [==============================] - 43s 5s/step - loss: 0.8409 - accuracy:
```

```
0.9040 - val_loss: 0.8497 - val_accuracy: 0.9217
Epoch 4/5
9/9 [==============================] - 43s 5s/step - loss: 0.8188 - accuracy:
0.9195 - val_loss: 0.8291 - val_accuracy: 0.9283
Epoch 5/5
9/9 [==============================] - 43s 5s/step - loss: 0.7977 - accuracy:
0.9314 - val_loss: 0.8091 - val_accuracy: 0.9294
```

Inferences from training the base model

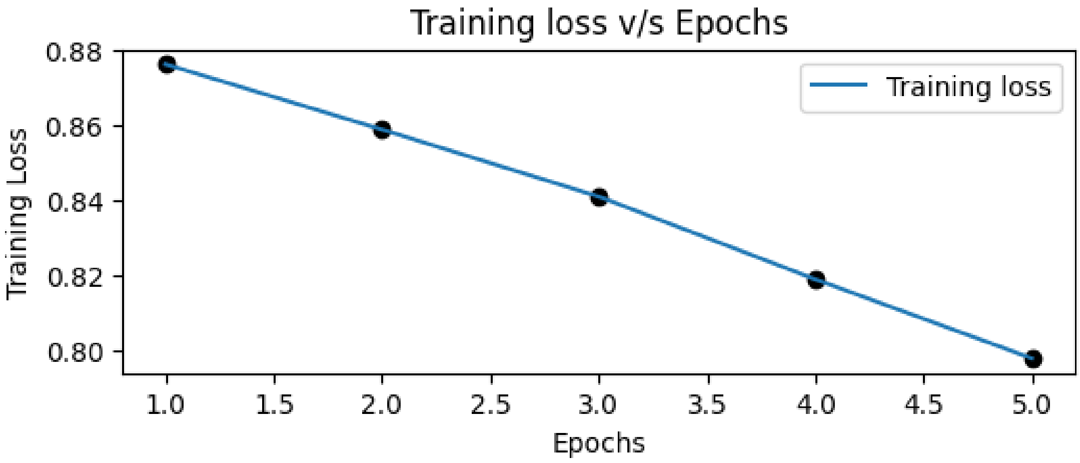
[41]:

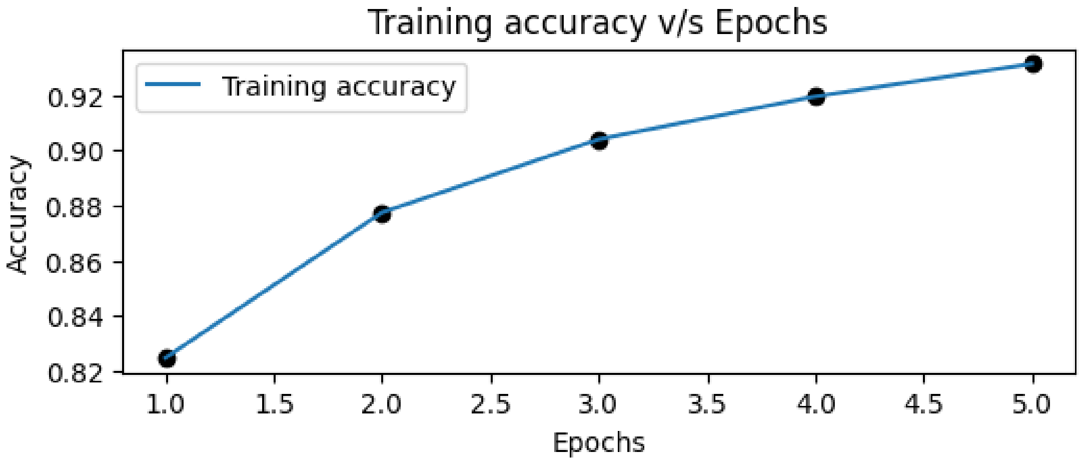
```python
import matplotlib.pyplot as plt

plt.subplot(2,1,1)
train_loss = hist.history['loss']
epochs = range(1,len(train_loss)+1)
plt.plot(epochs,train_loss,label="Training loss")
plt.scatter(epochs,train_loss,color="black")
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.title('Training loss v/s Epochs')
plt.legend()
plt.show()


plt.subplot(2,1,2)
train_acc = hist.history['accuracy']
epochs = range(1,len(train_acc)+1)
plt.plot(epochs,train_acc,label="Training accuracy")
plt.scatter(epochs,train_acc,color="black")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training accuracy v/s Epochs')
plt.legend()
plt.show()

plt.subplots_adjust(top=7,right=2.5,bottom=5)
```





```
<Figure size 640x480 with 0 Axes>
```

[43]:

```
prediction = model.evaluate(x_test,y_test,batch_size=8)
```

```
4/4 [==============================] - 5s 1s/step - loss: 0.8091 - accuracy:
0.9294
```

Test Results

[46]:

```
print('Loss obtained :',round(prediction[0],2),'\nAccuracy obtained :
  ↪',round(prediction[1],2))
```

```
Loss obtained : 0.81
Accuracy obtained : 0.93
```

0.0.4 Refinement Model - Attention 3D-UNet

Defining the attention block

[100]:

```python
def attention_gate(g, s, num_filters):
    Wg = keras.layers.Conv3D(num_filters, 1, padding="same")(g)

    Ws = keras.layers.Conv3D(num_filters, 1, padding="same")(s)
    print(Wg.shape,Ws.shape,"wwwww")
    out = keras.layers.Activation("relu")(Wg + Ws)
    out = keras.layers.Conv3D(num_filters, 1, padding="same")(out)
    out = keras.layers.Activation("sigmoid")(out)
    print(out.shape,s.shape,"ssss")

    return out * s
```

Modified decoder block

[101]:

```python
def decoder_block_attention(inp,skip_features,filters):
    #Upsampling to increase the dimension
    u1 = keras.layers.UpSampling3D(size=(2,2,2))(inp)
    print(u1.shape)
    s = attention_gate(u1,skip_features,filters)
    print(s.shape)
    skip = keras.layers.Concatenate()([u1,s])
    # Performing convolution operation
    c = conv_block(skip,filters)

    return c
```

Attention 3D-UNet Architecture

[102]:

```python
# Building the model
inp = keras.Input(shape=(64,64,64,1))

# Encoder block 1
c1,p1 = encoder_block(inp,64)
# Encoder block 2
c2,p2 = encoder_block(p1,128)
# Encoder block 3
c3,p3 = encoder_block(p2,256)
# Encoder block 4
c4,p4 = encoder_block(p3,512)

# Bottleneck layer - gives compressed representation of the raw image
b = conv_block(p4,1024)

# Decoder block 1
d1 = decoder_block_attention(b,c4,512)
```

```
# Decoder block 2
d2 = decoder_block_attention(d1,c3,256)
# Decoder block 3
d3 = decoder_block_attention(d2,c2,128)
#Decoder block 4
d4 = decoder_block_attention(d3,c1,64)

# Output layer
out = keras.layers.Conv3D(1,kernel_size=1,padding="same",activation="relu")(d4)
```

```
(None, 8, 8, 8, 1024)
(None, 8, 8, 8, 512) (None, 8, 8, 8, 512) wwwww
(None, 8, 8, 8, 512) (None, 8, 8, 8, 512) ssss
(None, 8, 8, 8, 512)
(None, 16, 16, 16, 512)
(None, 16, 16, 16, 256) (None, 16, 16, 16, 256) wwwww
(None, 16, 16, 16, 256) (None, 16, 16, 16, 256) ssss
(None, 16, 16, 16, 256)
(None, 32, 32, 32, 256)
(None, 32, 32, 32, 128) (None, 32, 32, 32, 128) wwwww
(None, 32, 32, 32, 128) (None, 32, 32, 32, 128) ssss
(None, 32, 32, 32, 128)
(None, 64, 64, 64, 128)
(None, 64, 64, 64, 64) (None, 64, 64, 64, 64) wwwww
(None, 64, 64, 64, 64) (None, 64, 64, 64, 64) ssss
(None, 64, 64, 64, 64)
```

[103]:

```
model.summary()
```

```
Model: "U-Net"
________________________________________________________________________________
__________________
 Layer (type)                Output Shape         Param #     Connected to
================================================================================
==================
 input_1 (InputLayer)         [(None, 64, 64, 64,  0           []
                               1)]

 conv3d (Conv3D)              (None, 64, 64, 64,   1792
['input_1[0][0]']

                              64)

 conv3d_1 (Conv3D)            (None, 64, 64, 64,   110656
['conv3d[0][0]']

                              64)

 max_pooling3d (MaxPooling3D)  (None, 32, 32, 32,   0
['conv3d_1[0][0]']
```

```
                                64)

 conv3d_2 (Conv3D)              (None, 32, 32, 32,   221312
['max_pooling3d[0][0]']
                                128)

 conv3d_3 (Conv3D)              (None, 32, 32, 32,   442496
['conv3d_2[0][0]']
                                128)

 max_pooling3d_1 (MaxPooling3D)  (None, 16, 16, 16,   0
['conv3d_3[0][0]']
                                128)

 conv3d_4 (Conv3D)              (None, 16, 16, 16,   884992
['max_pooling3d_1[0][0]']
                                256)

 conv3d_5 (Conv3D)              (None, 16, 16, 16,   1769728
['conv3d_4[0][0]']
                                256)

 max_pooling3d_2 (MaxPooling3D)  (None, 8, 8, 8, 256  0
['conv3d_5[0][0]']
                                )

 conv3d_6 (Conv3D)              (None, 8, 8, 8, 512  3539456
['max_pooling3d_2[0][0]']
                                )

 conv3d_7 (Conv3D)              (None, 8, 8, 8, 512  7078400
['conv3d_6[0][0]']
                                )

 max_pooling3d_3 (MaxPooling3D)  (None, 4, 4, 4, 512  0
['conv3d_7[0][0]']
                                )

 conv3d_8 (Conv3D)              (None, 4, 4, 4, 102  14156800
['max_pooling3d_3[0][0]']
                                4)

 conv3d_9 (Conv3D)              (None, 4, 4, 4, 102  28312576
['conv3d_8[0][0]']
                                4)

 up_sampling3d (UpSampling3D)   (None, 8, 8, 8, 102  0
['conv3d_9[0][0]']
```

```
                                4)

 concatenate (Concatenate)      (None, 8, 8, 8, 153  0
['up_sampling3d[0][0]',
                                6)
'conv3d_7[0][0]']

 conv3d_10 (Conv3D)             (None, 8, 8, 8, 512  21234176
['concatenate[0][0]']
                                )

 conv3d_11 (Conv3D)             (None, 8, 8, 8, 512  7078400
['conv3d_10[0][0]']
                                )

 up_sampling3d_1 (UpSampling3D)  (None, 16, 16, 16,   0
['conv3d_11[0][0]']
                                512)

 concatenate_1 (Concatenate)    (None, 16, 16, 16,   0
['up_sampling3d_1[0][0]',
                                768)
'conv3d_5[0][0]']

 conv3d_12 (Conv3D)             (None, 16, 16, 16,   5308672
['concatenate_1[0][0]']
                                256)

 conv3d_13 (Conv3D)             (None, 16, 16, 16,   1769728
['conv3d_12[0][0]']
                                256)

 up_sampling3d_2 (UpSampling3D)  (None, 32, 32, 32,   0
['conv3d_13[0][0]']
                                256)

 concatenate_2 (Concatenate)    (None, 32, 32, 32,   0
['up_sampling3d_2[0][0]',
                                384)
'conv3d_3[0][0]']

 conv3d_14 (Conv3D)             (None, 32, 32, 32,   1327232
['concatenate_2[0][0]']
                                128)

 conv3d_15 (Conv3D)             (None, 32, 32, 32,   442496
['conv3d_14[0][0]']
                                128)
 up_sampling3d_3 (UpSampling3D)  (None, 64, 64, 64,   0
['conv3d_15[0][0]']
                                128)

 concatenate_3 (Concatenate)    (None, 64, 64, 64,   0
['up_sampling3d_3[0][0]',
                                192)
'conv3d_1[0][0]']

 conv3d_16 (Conv3D)             (None, 64, 64, 64,   331840
['concatenate_3[0][0]']
                                64)

 conv3d_17 (Conv3D)             (None, 64, 64, 64,   110656
['conv3d_16[0][0]']
                                64)

 conv3d_18 (Conv3D)             (None, 64, 64, 64,   65
['conv3d_17[0][0]']
                                1)

================================================================================
==================
Total params: 94,121,473
Trainable params: 94,121,473
Non-trainable params: 0
--------------------------------------------------------------------------------
------------------
```

Training the model

```
[110]: hist_2 = model.
        ↪fit(x_train,y_train,batch_size=4,epochs=5,validation_data=(x_test,y_test))
```

```
Epoch 1/5
18/18 [==============================] - 44s 2s/step - loss: 0.6181 - accuracy:
0.9505 - val_loss: 0.9566 - val_accuracy: 0.9781
Epoch 2/5
18/18 [==============================] - 42s 2s/step - loss: 0.6956 - accuracy:
0.9278 - val_loss: 0.8106 - val_accuracy: 0.8686
Epoch 3/5
18/18 [==============================] - 48s 3s/step - loss: 0.6184 - accuracy:
0.9380 - val_loss: 0.5080 - val_accuracy: 0.9670
Epoch 4/5
18/18 [==============================] - 42s 2s/step - loss: 0.5734 - accuracy:
0.9491 - val_loss: 0.8072 - val_accuracy: 0.9781
Epoch 5/5
```

```
18/18 [==============================] - 42s 2s/step - loss: 0.5538 - accuracy:
0.9513 - val_loss: 0.6209 - val_accuracy: 0.9357
```

Inferences from the refinement model

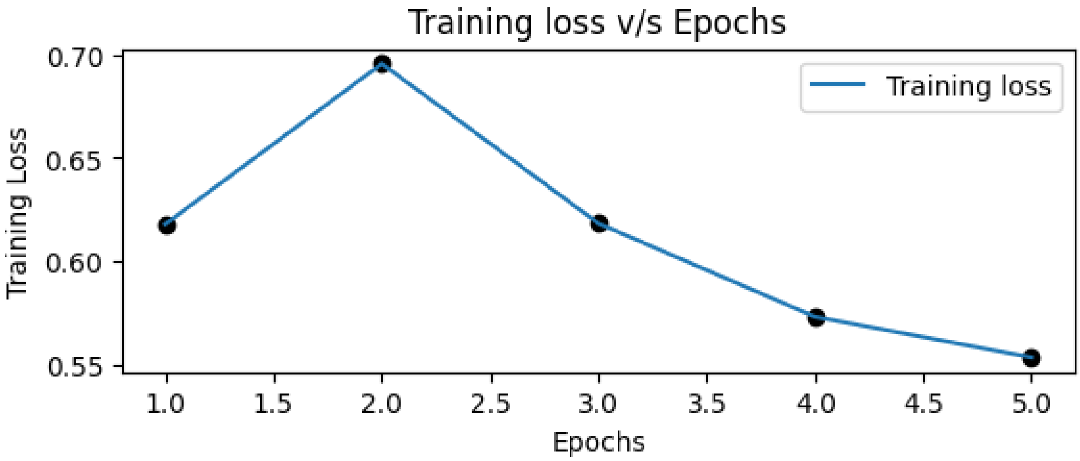
[111]:

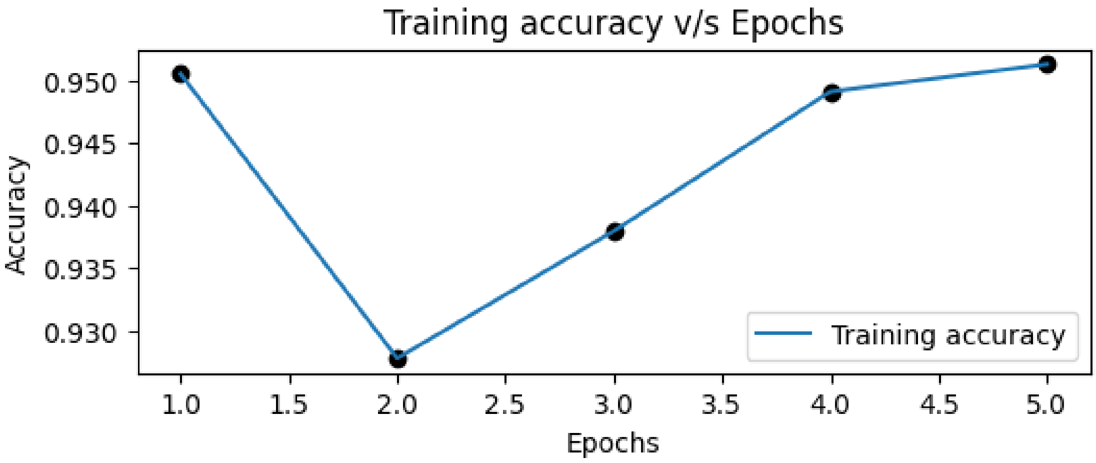
```python
import matplotlib.pyplot as plt

plt.subplot(2,1,1)
train_loss = hist_2.history['loss']
epochs = range(1,len(train_loss)+1)
plt.plot(epochs,train_loss,label="Training loss")
plt.scatter(epochs,train_loss,color="black")
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.title('Training loss v/s Epochs')
plt.legend()
plt.show()


plt.subplot(2,1,2)
train_acc = hist_2.history['accuracy']
epochs = range(1,len(train_acc)+1)
plt.plot(epochs,train_acc,label="Training accuracy")
plt.scatter(epochs,train_acc,color="black")
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training accuracy v/s Epochs')
plt.legend()
plt.show()

plt.subplots_adjust(top=7,right=2.5,bottom=5)
```





```
<Figure size 640x480 with 0 Axes>
```

[107]:
```
prediction_2 = model.evaluate(x_test,y_test,batch_size=8)
```

```
4/4 [==============================] - 5s 1s/step - loss: 0.6234 - accuracy:
0.9762
```

Test Results

[109]:
```
print('Loss obtained after refinement:',round(prediction_2[0],2),'\nAccuracy␣
↪obtained after refinement:',round(prediction_2[1],2))
```

```
Loss obtained after refinement: 0.62
Accuracy obtained after refinement: 0.98
```

[ ]:
```

```