```
  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1411 in
fbeta_score
    _, _, f, _ = precision_recall_fscore_support(

  File ~\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py:184 in wrapper
    return func(*args, **kwargs)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1721 in
precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1499 in
_check_set_wise_labels
    y_type, y_true, y_pred = _check_targets(y_true, y_pred)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:93 in
_check_targets
    raise ValueError(

ValueError: Classification metrics can't handle a mix of continuous and binary targets


In [99]:
    ...:       =     .
    ...:                    =        > 0.5
    ...:
    ...:              =   .
    ...:       =   .
    ...:           =   .                  /   .
    ...: print "IoU socre is: "
3/3 [==============================] - 0s 76ms/step
Traceback (most recent call last):

  Cell In[99], line 4
    intersection = np.logical_and(y_valid, y_pred_thresholded)

MemoryError: Unable to allocate 1.27 TiB for an array with shape (72, 128, 128, 1179648)
and data type bool


In [100]:

Removing all variables...
```

```
In [100]:
     ...: """
     ...: Created on Sat Jan  6 17:25:02 2024
     ...:
     ...: @author: Sabbir Ahmed Sibli
     ...: source: https://github.com/hlamba28/UNET-TGS/blob/master/TGS%20UNET.ipynb
     ...: """
     ...:
     ...: from             import
     ...: import
     ...: import       as
     ...: import                  as
     ...: import
     ...: from         import
     ...:
     ...: from                    import
     ...: from                          import
     ...:
     ...: from               import
     ...: from                  import
     ...: from                   import
     ...:
     ...: # Set some params
     ...:          = 128
     ...:           = 128
     ...:        = 5
     ...:
     ...: # Loading the dataset and the masks
     ...: # Loading the dataset (Original and Masked)
     ...:            = 'D:/Course Materials [Erasmus MSc]/University of Kragujevac/
Biomedical Image Processing/Assignments/Datasets/Spine_DICOM'
     ...:             = 'D:/Course Materials [Erasmus MSc]/University of Kragujevac/
Biomedical Image Processing/Assignments/Datasets/masked_spines'
     ...:
     ...: # Reading original dicom slices
     ...:                  =        .                     format='dcm'
     ...:
     ...:
     ...: # Reading masked images
     ...:       =
     ...: for      in                   .
     ...:     if not     .
     ...:         continue
     ...:          .              .
     ...:
     ...:            =   .                  # converting list into numpy array
     ...:            =              -1
     ...:
     ...: # Resize Image to 128x128
     ...:           = 128
     ...:          = 128
Reading DICOM (examining files): 717/717 files (100.0%)
  Found 1 correct series.
Reading DICOM (loading data): 717/717  (100.0%)
C:\Users\Sabbir Ahmed Sibli\AppData\Local\Temp\ipykernel_18272\312730056.py:41:
DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch
to that of iio.v3.imread. To keep the current behavior (and make this warning disappear)
use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  masks.append(imageio.imread(file))

In [101]:         'D:/Course Materials [Erasmus MSc]/University of Kragujevac/Biomedical
Image Processing/Assignments/Seminar Paper/Codes/UNET_Segmentation/UNET_Example_02/
Unet_Portion.py'      ='D:/Course Materials [Erasmus MSc]/University of Kragujevac/
Biomedical Image Processing/Assignments/Seminar Paper/Codes/UNET_Segmentation/
UNET_Example_02'
Reloaded modules: Unet_Portion

In [102]:
     ...:   =   .       len                                           1
     =  .             # Create array of zeros for data
     ...:   =   .       len                                     1        =  .
#Create array of zeros for masks
     ...: for   in range 0 len
     ...:     # Load original images
     ...:         =
     ...:           =                                   1        = 'constant'
               = True
     ...:     # Load masks
     ...:          =
     ...:          =                                    1        = 'constant'
               = True
     ...:     # Creating Normalized image (converting all pixel values between 0 and 1)
     ...:          =      /255.0
     ...:          =     /255.0
     ...:
     ...: # Split train and valid
     ...:                                    =                                  =0.1
            =42

In [103]:
     ...:
     ...:              =       .        0  len
     ...:    .              = 12   6
     ...:    .        121
     ...:    .      'Dicom Slice'
     ...:    .         .                        128  128        ='gray'
     ...:    .        122
     ...:    .      'Corresponding Mask'
     ...:    .         .                        128  128        ='gray'
     ...:    .

In [104]:
     ...:
     ...:           =                             1       ='img'
     ...:       =                          =16         =0.05           =True
     ...:      .                 =            ="binary_crossentropy"
       = "accuracy"

In [105]:
     ...:
     ...:      .
     ...:
     ...:           =                 'best_model.h5'         ='val_loss'
              =True      ='min'         =1
     ...:
     ...:         =      .                               =32        =20
         =                          =
Model: "model_4"
__________________________________________________________________________________________
__________
 Layer (type)                Output Shape                 Param #    Connected to
==========================================================================================
==========
 img (InputLayer)            [(None, 128, 128, 1)]        0          []

 conv2d_115 (Conv2D)         (None, 128, 128, 16)         160        ['img[0][0]']
 batch_normalization_109 (B  (None, 128, 128, 16)         64        ['conv2d_115[0][0]']
 atchNormalization)

 activation_109 (Activation  (None, 128, 128, 16)         0
['batch_normalization_109[0][0
 )                                                                  ]']

 max_pooling2d_24 (MaxPooli  (None, 64, 64, 16)           0         ['activation_109[0]
[0]']
 ng2D)

 dropout_32 (Dropout)        (None, 64, 64, 16)           0
['max_pooling2d_24[0][0]']

 conv2d_117 (Conv2D)         (None, 64, 64, 32)           4640      ['dropout_32[0][0]']

 batch_normalization_111 (B  (None, 64, 64, 32)           128       ['conv2d_117[0][0]']
 atchNormalization)

 activation_111 (Activation  (None, 64, 64, 32)           0
['batch_normalization_111[0][0
 )                                                                  ]']

 max_pooling2d_25 (MaxPooli  (None, 32, 32, 32)           0         ['activation_111[0]
[0]']
 ng2D)

 dropout_33 (Dropout)        (None, 32, 32, 32)           0
['max_pooling2d_25[0][0]']

 conv2d_119 (Conv2D)         (None, 32, 32, 64)           18496     ['dropout_33[0][0]']

 batch_normalization_113 (B  (None, 32, 32, 64)           256       ['conv2d_119[0][0]']
 atchNormalization)

 activation_113 (Activation  (None, 32, 32, 64)           0
['batch_normalization_113[0][0
 )                                                                  ]']

 max_pooling2d_26 (MaxPooli  (None, 16, 16, 64)           0         ['activation_113[0]
[0]']
 ng2D)

 dropout_34 (Dropout)        (None, 16, 16, 64)           0
['max_pooling2d_26[0][0]']

 conv2d_121 (Conv2D)         (None, 16, 16, 128)          73856     ['dropout_34[0][0]']

 batch_normalization_115 (B  (None, 16, 16, 128)          512       ['conv2d_121[0][0]']
 atchNormalization)

 activation_115 (Activation  (None, 16, 16, 128)          0
['batch_normalization_115[0][0
 )                                                                  ]']

 max_pooling2d_27 (MaxPooli  (None, 8, 8, 128)            0         ['activation_115[0]
[0]']
 ng2D)
 dropout_35 (Dropout)        (None, 8, 8, 128)            0
['max_pooling2d_27[0][0]']

 conv2d_123 (Conv2D)         (None, 8, 8, 256)            295168    ['dropout_35[0][0]']

 batch_normalization_117 (B  (None, 8, 8, 256)            1024      ['conv2d_123[0][0]']
 atchNormalization)

 activation_117 (Activation  (None, 8, 8, 256)            0
['batch_normalization_117[0][0
 )                                                                  ]']

 conv2d_transpose_24 (Conv2  (None, 16, 16, 128)          295040    ['activation_117[0]
[0]']
 DTranspose)

 concatenate_24 (Concatenat  (None, 16, 16, 256)          0
['conv2d_transpose_24[0][0]',
 e)                                                                  'activation_115[0]
[0]']

 dropout_36 (Dropout)        (None, 16, 16, 256)          0         ['concatenate_24[0]
[0]']

 conv2d_125 (Conv2D)         (None, 16, 16, 128)          295040    ['dropout_36[0][0]']

 batch_normalization_119 (B  (None, 16, 16, 128)          512       ['conv2d_125[0][0]']
 atchNormalization)

 activation_119 (Activation  (None, 16, 16, 128)          0
['batch_normalization_119[0][0
 )                                                                  ]']

 conv2d_transpose_25 (Conv2  (None, 32, 32, 64)           73792     ['activation_119[0]
[0]']
 DTranspose)

 concatenate_25 (Concatenat  (None, 32, 32, 128)          0
['conv2d_transpose_25[0][0]',
 e)                                                                  'activation_113[0]
[0]']

 dropout_37 (Dropout)        (None, 32, 32, 128)          0         ['concatenate_25[0]
[0]']

 conv2d_127 (Conv2D)         (None, 32, 32, 64)           73792     ['dropout_37[0][0]']

 batch_normalization_121 (B  (None, 32, 32, 64)           256       ['conv2d_127[0][0]']
 atchNormalization)

 activation_121 (Activation  (None, 32, 32, 64)           0
['batch_normalization_121[0][0
 )                                                                  ]']

 conv2d_transpose_26 (Conv2  (None, 64, 64, 32)           18464     ['activation_121[0]
[0]']
 DTranspose)
 concatenate_26 (Concatenat  (None, 64, 64, 64)           0
['conv2d_transpose_26[0][0]',
 e)                                                                 'activation_111[0]
[0]']

 dropout_38 (Dropout)        (None, 64, 64, 64)           0         ['concatenate_26[0]
[0]']

 conv2d_129 (Conv2D)         (None, 64, 64, 32)           18464     ['dropout_38[0][0]']

 batch_normalization_123 (B  (None, 64, 64, 32)           128       ['conv2d_129[0][0]']
 atchNormalization)

 activation_123 (Activation  (None, 64, 64, 32)           0
['batch_normalization_123[0][0
 )                                                                  ]']

 conv2d_transpose_27 (Conv2  (None, 128, 128, 16)         4624      ['activation_123[0]
[0]']
 DTranspose)

 concatenate_27 (Concatenat  (None, 128, 128, 32)         0
['conv2d_transpose_27[0][0]',
 e)                                                                 'activation_109[0]
[0]']

 dropout_39 (Dropout)        (None, 128, 128, 32)         0         ['concatenate_27[0]
[0]']

 conv2d_131 (Conv2D)         (None, 128, 128, 16)         4624      ['dropout_39[0][0]']

 batch_normalization_125 (B  (None, 128, 128, 16)         64        ['conv2d_131[0][0]']
 atchNormalization)

 activation_125 (Activation  (None, 128, 128, 16)         0
['batch_normalization_125[0][0
 )                                                                  ]']

 conv2d_132 (Conv2D)         (None, 128, 128, 1)          17        ['activation_125[0]
[0]']

========================================================================================
==========
Total params: 1179121 (4.50 MB)
Trainable params: 1177649 (4.49 MB)
Non-trainable params: 1472 (5.75 KB)
________________________________________________________________________________________
__________
Epoch 1/20
21/21 [==============================] - ETA: 0s - loss: 0.4104 - accuracy: 0.8572
Epoch 1: val_loss improved from inf to 0.23607, saving model to best_model.h5
C:\Users\Sabbir Ahmed Sibli\anaconda3\Lib\site-packages\keras\src\engine\training.py:
3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras format,
e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
21/21 [==============================] - 15s 567ms/step - loss: 0.4104 - accuracy:
```

```
0.8572 - val_loss: 0.2361 - val_accuracy: 0.8944
Epoch 2/20
21/21 [==============================] - ETA: 0s - loss: 0.2596 - accuracy: 0.9216
Epoch 2: val_loss improved from 0.23607 to 0.21248, saving model to best_model.h5
21/21 [==============================] - 24s 1s/step - loss: 0.2596 - accuracy: 0.9216 -
val_loss: 0.2125 - val_accuracy: 0.9039
Epoch 3/20
21/21 [==============================] - ETA: 0s - loss: 0.1964 - accuracy: 0.9246
Epoch 3: val_loss did not improve from 0.21248
21/21 [==============================] - 25s 1s/step - loss: 0.1964 - accuracy: 0.9246 -
val_loss: 0.5947 - val_accuracy: 0.7334
Epoch 4/20
21/21 [==============================] - ETA: 0s - loss: 0.1609 - accuracy: 0.9248
Epoch 4: val_loss did not improve from 0.21248
21/21 [==============================] - 24s 1s/step - loss: 0.1609 - accuracy: 0.9248 -
val_loss: 0.4217 - val_accuracy: 0.7762
Epoch 5/20
21/21 [==============================] - ETA: 0s - loss: 0.1358 - accuracy: 0.9249
Epoch 5: val_loss did not improve from 0.21248
21/21 [==============================] - 24s 1s/step - loss: 0.1358 - accuracy: 0.9249 -
val_loss: 0.3346 - val_accuracy: 0.8301
Epoch 6/20
21/21 [==============================] - ETA: 0s - loss: 0.1164 - accuracy: 0.9249
Epoch 6: val_loss improved from 0.21248 to 0.14507, saving model to best_model.h5
21/21 [==============================] - 18s 861ms/step - loss: 0.1164 - accuracy:
0.9249 - val_loss: 0.1451 - val_accuracy: 0.9178
Epoch 7/20
21/21 [==============================] - ETA: 0s - loss: 0.1005 - accuracy: 0.9249
Epoch 7: val_loss improved from 0.14507 to 0.10951, saving model to best_model.h5
21/21 [==============================] - 15s 736ms/step - loss: 0.1005 - accuracy:
0.9249 - val_loss: 0.1095 - val_accuracy: 0.9236
Epoch 8/20
21/21 [==============================] - ETA: 0s - loss: 0.0874 - accuracy: 0.9249
Epoch 8: val_loss improved from 0.10951 to 0.08731, saving model to best_model.h5
21/21 [==============================] - 15s 716ms/step - loss: 0.0874 - accuracy:
0.9249 - val_loss: 0.0873 - val_accuracy: 0.9248
Epoch 9/20
21/21 [==============================] - ETA: 0s - loss: 0.0765 - accuracy: 0.9249
Epoch 9: val_loss improved from 0.08731 to 0.07442, saving model to best_model.h5
21/21 [==============================] - 20s 971ms/step - loss: 0.0765 - accuracy:
0.9249 - val_loss: 0.0744 - val_accuracy: 0.9249
Epoch 10/20
21/21 [==============================] - ETA: 0s - loss: 0.0676 - accuracy: 0.9249
Epoch 10: val_loss improved from 0.07442 to 0.06252, saving model to best_model.h5
21/21 [==============================] - 25s 1s/step - loss: 0.0676 - accuracy: 0.9249 -
val_loss: 0.0625 - val_accuracy: 0.9249
Epoch 11/20
21/21 [==============================] - ETA: 0s - loss: 0.0601 - accuracy: 0.9249
Epoch 11: val_loss improved from 0.06252 to 0.05419, saving model to best_model.h5
21/21 [==============================] - 25s 1s/step - loss: 0.0601 - accuracy: 0.9249 -
val_loss: 0.0542 - val_accuracy: 0.9249
Epoch 12/20
21/21 [==============================] - ETA: 0s - loss: 0.0540 - accuracy: 0.9249
Epoch 12: val_loss improved from 0.05419 to 0.04805, saving model to best_model.h5
21/21 [==============================] - 24s 1s/step - loss: 0.0540 - accuracy: 0.9249 -
val_loss: 0.0481 - val_accuracy: 0.9249
Epoch 13/20
21/21 [==============================] - ETA: 0s - loss: 0.0490 - accuracy: 0.9249
```

```
Epoch 13: val_loss improved from 0.04805 to 0.04250, saving model to best_model.h5
21/21 [==============================] - 24s 1s/step - loss: 0.0490 - accuracy: 0.9249 -
val_loss: 0.0425 - val_accuracy: 0.9249
Epoch 14/20
21/21 [==============================] - ETA: 0s - loss: 0.0444 - accuracy: 0.9249
Epoch 14: val_loss improved from 0.04250 to 0.03971, saving model to best_model.h5
21/21 [==============================] - 16s 752ms/step - loss: 0.0444 - accuracy:
0.9249 - val_loss: 0.0397 - val_accuracy: 0.9249
Epoch 15/20
21/21 [==============================] - ETA: 0s - loss: 0.0408 - accuracy: 0.9249
Epoch 15: val_loss improved from 0.03971 to 0.03616, saving model to best_model.h5
21/21 [==============================] - 24s 1s/step - loss: 0.0408 - accuracy: 0.9249 -
val_loss: 0.0362 - val_accuracy: 0.9249
Epoch 16/20
21/21 [==============================] - ETA: 0s - loss: 0.0375 - accuracy: 0.9249
Epoch 16: val_loss improved from 0.03616 to 0.03385, saving model to best_model.h5
21/21 [==============================] - 25s 1s/step - loss: 0.0375 - accuracy: 0.9249 -
val_loss: 0.0338 - val_accuracy: 0.9249
Epoch 17/20
21/21 [==============================] - ETA: 0s - loss: 0.0347 - accuracy: 0.9249
Epoch 17: val_loss improved from 0.03385 to 0.03341, saving model to best_model.h5
21/21 [==============================] - 16s 768ms/step - loss: 0.0347 - accuracy:
0.9249 - val_loss: 0.0334 - val_accuracy: 0.9249
Epoch 18/20
21/21 [==============================] - ETA: 0s - loss: 0.0325 - accuracy: 0.9249
Epoch 18: val_loss improved from 0.03341 to 0.03024, saving model to best_model.h5
21/21 [==============================] - 19s 894ms/step - loss: 0.0325 - accuracy:
0.9249 - val_loss: 0.0302 - val_accuracy: 0.9249
Epoch 19/20
21/21 [==============================] - ETA: 0s - loss: 0.0304 - accuracy: 0.9249
Epoch 19: val_loss improved from 0.03024 to 0.02858, saving model to best_model.h5
21/21 [==============================] - 16s 779ms/step - loss: 0.0304 - accuracy:
0.9249 - val_loss: 0.0286 - val_accuracy: 0.9249
Epoch 20/20
21/21 [==============================] - ETA: 0s - loss: 0.0285 - accuracy: 0.9249
Epoch 20: val_loss improved from 0.02858 to 0.02718, saving model to best_model.h5
21/21 [==============================] - 16s 757ms/step - loss: 0.0285 - accuracy:
0.9249 - val_loss: 0.0272 - val_accuracy: 0.9249
```

```
In [106]:
     ...:
     ...:
     ...:      =        .        'loss'
     ...:          =        .        'val_loss'
     ...:        = range 1  len       + 1
     ...:    .                   'y'       ='Training loss'
     ...:    .                       'r'       ='Validation loss'
     ...:    .      'Training and validation loss'
     ...:    .       'Epochs'
     ...:    .       'Loss'
     ...:    .
     ...:    .
```

```
In [107]:
     ...:
     ...:     =        .        'accuracy'
     ...:         =        .        'val_accuracy'
     ...:        = range 1  len      + 1
     ...:    .                  'y'       ='Training acc'
     ...:    .                      'r'       ='Validation acc'
     ...:    .      'Training and validation accuracy'
     ...:    .       'Epochs'
     ...:    .       'Accuracy'
     ...:    .
     ...:    .

In [108]:
     ...:
     ...:                =       .        0  len
     ...:         =
     ...:            =
     ...:             =             0      None
     ...:              =  .                           0
     ...:           =       .                        0     0  > 0.2 .         .
1/1 [==============================] - 0s 301ms/step

In [109]:
     ...:
     ...:    .              = 16  8
     ...:    .        231
     ...:    .      'Testing Image'
     ...:    .                    0       ='gray'
     ...:    .        232
     ...:    .      'Testing Label'
     ...:    .                        0       ='gray'
     ...:    .        233
     ...:    .      'Prediction on test image'
     ...:    .                       ='gray'
     ...:
     ...:    .

In [110]:
     ...:
     ...:      =     .
     ...:                   =        > 0.5
     ...:
     ...:             =   .
     ...:      =   .
     ...:          =   .                  /   .
     ...: print "IoU socre is: "
3/3 [==============================] - 0s 74ms/step
IoU socre is:  0.5673644703919933

In [111]:
     ...:
     ...: from                 import
     ...:
     ...: # Reshape y_valid and y_pred_thresholded if needed (e.g., if they are 3D)
     ...:                =        .        -1
     ...:                           =                   .        -1
     ...:
     ...: # Calculate Dice Coefficient using the f1_score function
     ...:                 =
     ...:
     ...: print "Dice Coefficient Score:"
Traceback (most recent call last):
  Cell In[111], line 8
    dice_coefficient = f1_score(y_valid_reshape, y_pred_thresholded_reshape)

  File ~\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py:211 in wrapper
    return func(*args, **kwargs)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1238 in f1_score
    return fbeta_score(

  File ~\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py:184 in wrapper
    return func(*args, **kwargs)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1411 in
fbeta_score
    _, _, f, _ = precision_recall_fscore_support(

  File ~\anaconda3\Lib\site-packages\sklearn\utils\_param_validation.py:184 in wrapper
    return func(*args, **kwargs)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1721 in
precision_recall_fscore_support
    labels = _check_set_wise_labels(y_true, y_pred, average, labels, pos_label)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1499 in
_check_set_wise_labels
    y_type, y_true, y_pred = _check_targets(y_true, y_pred)

  File ~\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:93 in
_check_targets
    raise ValueError(

ValueError: Classification metrics can't handle a mix of continuous and binary targets


In [112]:
     ...:
     ...:       =     .
     ...:                    =        > 0.5
     ...:
     ...:              =   .
     ...:       =   .
     ...:           =   .                  /   .
     ...: print "IoU socre is: "
3/3 [==============================] - 0s 78ms/step
IoU socre is:  0.5673644703919933

In [113]:
     ...:                  =  2.0 *   .                   /    .            +
  .                        + 1e-8
     ...: print "Dice Coefficient:"
Dice Coefficient: 0.9992650215136403

In [114]:
     ...:       =     .
     ...:                    =        > 0.5
     ...:
     ...:              =   .
     ...:       =   .
     ...:           =   .                  /   .
     ...: print "IoU socre is: "
3/3 [==============================] - 0s 76ms/step
IoU socre is:  0.5673644703919933

In [115]:                  =  2.0 *   .                   /    .             +
  .                        + 1e-8
     ...: print "Dice Coefficient:"
Dice Coefficient: 0.9992650215136403

In [116]:
```