IMAGE SEGMENTATION USING CNN

Riddhi Bhalerao(212CS005)
Shelley Tated(212CS028)
Shreyasi Shirbhate(212CS030)
Siddhi Bhalerao(212IS032)

First Year M.Tech

Department Of Computer Science and Engineering

National Institute of Technology Karnataka,

Surathkal, Mangalore- 575025

January 2022

About Dataset

We are given medical images. Data was divided into three parts namely :

- Training
- Validation
- Testing

Again each of them contains:

- Images
- Masks

Size of the Data Set:

Train :

Image: 1160 (Each of size (320, 320, 3))

Mask: 1160 (Each of size (320, 320, 3))

Test :

Image: 542 (Each of size (320, 320, 3))

Mask : 542 (Each of size (320, 320, 3))

Valid :

Image: 569 (Each of size (320, 320, 3))

Mask : 569 (Each of size (320, 320, 3))

Dataset:

Figure: Sample Data set Images

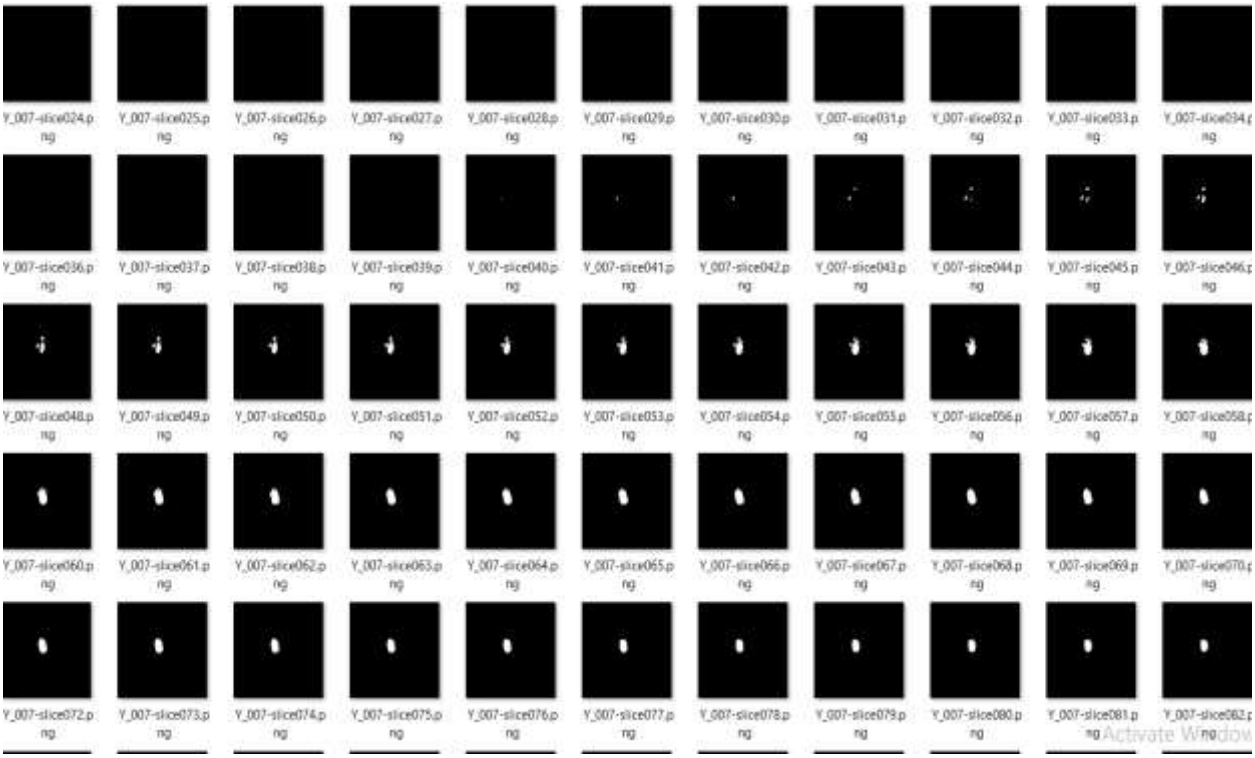


Figure: Sample Data set Masks

Flow of Model

The flow of our model is as follows:

- Uploading the dataset
- Defining the train, test and model directories
- Data Preparation
- Resizing Images
- Normalization the Data
- Defining Model
- Training and developing our model
- Testing the model
- Summarizing the result

Data Preparation

- Load Dataset into My Drive.
- Load Images.
- Resizing Images to (256,256)
- Convert images into the NPY array.for each training, testing and valid data.
- Normalize it by dividing it by 255.
- Applied Batch Normalization

Architecture

We have developed our model using Scratch.

We got inspiration and referred model architecture U-Net and using that we have made our modelusing scratch.

Activation Function:

We used **ReLU** activation function in hidden layers to speed up the convergence and **Sigmoid** at the outermost layer as it is multi classification model Last Two layers are Fully Connected.

Down-Sampling:

Initial input image size: 256,256

Filter sizes used: 3,3.

Max_Pooling: 2,2

Kernel initialization used : he_normal

We have used Dropout before the last fully connected layer.

Up-Sampling:

Convolutional Transpose method is used for retrieving the dimension of original image.

```
lr=0.001
x='he_normal'
y=0.0
a='relu'

model = create_model(lr,x,y,a)
model.summary()
```

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
=================================================================================================
 input_1 (InputLayer)           [(None, 256, 256, 1  0           []
                                )]

 conv2d (Conv2D)                (None, 256, 256, 32  320         ['input_1[0][0]']
                                )

 conv2d_1 (Conv2D)              (None, 256, 256, 32  9248        ['conv2d[0][0]']
                                )

 batch_normalization (BatchNorm  (None, 256, 256, 32  128        ['conv2d_1[0][0]']
 alization)                     )

 max_pooling2d (MaxPooling2D)   (None, 128, 128, 32  0           ['batch_normalization[0][0]']
                                )

 conv2d_2 (Conv2D)              (None, 128, 128, 64  18496       ['max_pooling2d[0][0]']
                                )

 conv2d_3 (Conv2D)              (None, 128, 128, 64  36928       ['conv2d_2[0][0]']
                                )

 batch_normalization_1 (BatchNo  (None, 128, 128, 64  256        ['conv2d_3[0][0]']
 rmalization)                   )

 max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 64)  0           ['batch_normalization_1[0][0]']

 conv2d_4 (Conv2D)              (None, 64, 64, 128)  73856       ['max_pooling2d_1[0][0]']

 conv2d_5 (Conv2D)              (None, 64, 64, 128)  147584      ['conv2d_4[0][0]']

 batch_normalization_2 (BatchNo  (None, 64, 64, 128)  512        ['conv2d_5[0][0]']
 rmalization)

 max_pooling2d_2 (MaxPooling2D)  (None, 32, 32, 128)  0          ['batch_normalization_2[0][0]']

 conv2d_6 (Conv2D)              (None, 32, 32, 256)  295168      ['max_pooling2d_2[0][0]']

 conv2d_7 (Conv2D)              (None, 32, 32, 256)  590080      ['conv2d_6[0][0]']

 batch_normalization_3 (BatchNo  (None, 32, 32, 256)  1024       ['conv2d_7[0][0]']
 rmalization)

 conv2d_transpose (Conv2DTransp  (None, 64, 64, 128)  131200     ['batch_normalization_3[0][0]']
 ose)

 concatenate (Concatenate)      (None, 64, 64, 256)  0           ['conv2d_transpose[0][0]',
                                                                  'batch_normalization_2[0][0]']
```

```
conv2d_8 (Conv2D)              (None, 64, 64, 128)  295040      ['concatenate[0][0]']

dropout (Dropout)              (None, 64, 64, 128)  0           ['conv2d_8[0][0]']

conv2d_9 (Conv2D)              (None, 64, 64, 128)  147584      ['dropout[0][0]']

batch_normalization_4 (BatchNo  (None, 64, 64, 128)  512        ['conv2d_9[0][0]']
rmalization)

conv2d_transpose_1 (Conv2DTran  (None, 128, 128, 64  32832      ['batch_normalization_4[0][0]']
spose)                         )

concatenate_1 (Concatenate)    (None, 128, 128, 12  0           ['conv2d_transpose_1[0][0]',
                               8)                                'batch_normalization_1[0][0]']

conv2d_10 (Conv2D)             (None, 128, 128, 64  73792       ['concatenate_1[0][0]']
                               )

conv2d_11 (Conv2D)             (None, 128, 128, 64  36928       ['conv2d_10[0][0]']
                               )

batch_normalization_5 (BatchNo  (None, 128, 128, 64  256        ['conv2d_11[0][0]']
rmalization)                   )

conv2d_transpose_2 (Conv2DTran  (None, 256, 256, 32  8224       ['batch_normalization_5[0][0]']
spose)                         )

concatenate_2 (Concatenate)    (None, 256, 256, 64  0           ['conv2d_transpose_2[0][0]',
                               )                                 'batch_normalization[0][0]']

conv2d_12 (Conv2D)             (None, 256, 256, 32  18464       ['concatenate_2[0][0]']
                               )

conv2d_13 (Conv2D)             (None, 256, 256, 32  9248        ['conv2d_12[0][0]']
                               )

batch_normalization_6 (BatchNo  (None, 256, 256, 32  128        ['conv2d_13[0][0]']
rmalization)                   )

conv2d_14 (Conv2D)             (None, 256, 256, 1)  33          ['batch_normalization_6[0][0]']
```

Training

Number of Epochs: 20

Batch size: 32

Shuffle : True

Callbacks: ModelCheckpoint, CSVLogger, Early Stopping

Loss Function: Binary Cross Entropy

```
history = model.fit(train_X,train_y,batch_size=32, epochs=20,validation_data = (valid_X,valid_y),verbose=1,callbacks=[mc,cv,es],shuffle=True)

Epoch 1/20
37/37 [==============================] - ETA: 0s - loss: 0.6416 - dice_coef: 0.0107 - accuracy: 0.7981
Epoch 00001: val_loss improved from inf to 1.69475, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 86s 2s/step - loss: 0.6416 - dice_coef: 0.0107 - accuracy: 0.7981 - val_loss: 1.6947 - val_dice_coef: 0.0100 - val_a
Epoch 2/20
37/37 [==============================] - ETA: 0s - loss: 0.4757 - dice_coef: 0.0181 - accuracy: 0.9900
Epoch 00002: val_loss improved from 1.69475 to 0.51861, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.4757 - dice_coef: 0.0181 - accuracy: 0.9900 - val_loss: 0.5186 - val_dice_coef: 0.0157 - val_a
Epoch 3/20
37/37 [==============================] - ETA: 0s - loss: 0.3078 - dice_coef: 0.0277 - accuracy: 0.9972
Epoch 00003: val_loss improved from 0.51861 to 0.38477, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.3078 - dice_coef: 0.0277 - accuracy: 0.9972 - val_loss: 0.3848 - val_dice_coef: 0.0265 - val_a
Epoch 4/20
37/37 [==============================] - ETA: 0s - loss: 0.1958 - dice_coef: 0.0400 - accuracy: 0.9978
Epoch 00004: val_loss improved from 0.38477 to 0.14076, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.1958 - dice_coef: 0.0400 - accuracy: 0.9978 - val_loss: 0.1408 - val_dice_coef: 0.0390 - val_a
Epoch 5/20
37/37 [==============================] - ETA: 0s - loss: 0.1291 - dice_coef: 0.0565 - accuracy: 0.9980
Epoch 00005: val_loss improved from 0.14076 to 0.12054, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.1291 - dice_coef: 0.0565 - accuracy: 0.9980 - val_loss: 0.1205 - val_dice_coef: 0.0584 - val_a
Epoch 6/20
37/37 [==============================] - ETA: 0s - loss: 0.0831 - dice_coef: 0.0857 - accuracy: 0.9984
Epoch 00006: val_loss improved from 0.12054 to 0.05841, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0831 - dice_coef: 0.0857 - accuracy: 0.9984 - val_loss: 0.0584 - val_dice_coef: 0.1182 - val_a
```

```
history = model.fit(train_X,train_y,batch_size=32, epochs=20,validation_data = (valid_X,valid_y),verbose=1,callbacks=[mc,cv,es],shuffle=True)

Epoch 6/20
37/37 [==============================] - ETA: 0s - loss: 0.0831 - dice_coef: 0.0857 - accuracy: 0.9984
Epoch 00006: val_loss improved from 0.12054 to 0.05841, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0831 - dice_coef: 0.0857 - accuracy: 0.9984 - val_loss: 0.0584 - val_dice_coef: 0.1182 - val_accu
Epoch 7/20
37/37 [==============================] - ETA: 0s - loss: 0.0575 - dice_coef: 0.1227 - accuracy: 0.9986
Epoch 00007: val_loss improved from 0.05841 to 0.03190, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0575 - dice_coef: 0.1227 - accuracy: 0.9986 - val_loss: 0.0319 - val_dice_coef: 0.1412 - val_accu
Epoch 8/20
37/37 [==============================] - ETA: 0s - loss: 0.0429 - dice_coef: 0.1563 - accuracy: 0.9986
Epoch 00008: val_loss improved from 0.03190 to 0.02197, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0429 - dice_coef: 0.1563 - accuracy: 0.9986 - val_loss: 0.0220 - val_dice_coef: 0.1797 - val_accu
Epoch 9/20
37/37 [==============================] - ETA: 0s - loss: 0.0327 - dice_coef: 0.1996 - accuracy: 0.9987
Epoch 00009: val_loss improved from 0.02197 to 0.01763, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0327 - dice_coef: 0.1996 - accuracy: 0.9987 - val_loss: 0.0176 - val_dice_coef: 0.2214 - val_accu
Epoch 10/20
37/37 [==============================] - ETA: 0s - loss: 0.0258 - dice_coef: 0.2388 - accuracy: 0.9988
Epoch 00010: val_loss improved from 0.01763 to 0.01700, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0258 - dice_coef: 0.2388 - accuracy: 0.9988 - val_loss: 0.0170 - val_dice_coef: 0.2593 - val_accu
Epoch 11/20
37/37 [==============================] - ETA: 0s - loss: 0.0209 - dice_coef: 0.2820 - accuracy: 0.9989
Epoch 00011: val_loss improved from 0.01700 to 0.01467, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0209 - dice_coef: 0.2820 - accuracy: 0.9989 - val_loss: 0.0147 - val_dice_coef: 0.2795 - val_accu
```

```
Epoch 14/20
37/37 [==============================] - ETA: 0s - loss: 0.0129 - dice_coef: 0.3930 - accuracy: 0.9989
Epoch 00014: val_loss did not improve from 0.01189
37/37 [==============================] - 53s 1s/step - loss: 0.0129 - dice_coef: 0.3930 - accuracy: 0.9989 - val_loss: 0.0127 - val_dice_coef: 0.3311 - val_acc
Epoch 15/20
37/37 [==============================] - ETA: 0s - loss: 0.0111 - dice_coef: 0.4321 - accuracy: 0.9990
Epoch 00015: val_loss did not improve from 0.01189
37/37 [==============================] - 53s 1s/step - loss: 0.0111 - dice_coef: 0.4321 - accuracy: 0.9990 - val_loss: 0.0126 - val_dice_coef: 0.3390 - val_acc
Epoch 16/20
37/37 [==============================] - ETA: 0s - loss: 0.0099 - dice_coef: 0.4593 - accuracy: 0.9990
Epoch 00016: val_loss improved from 0.01189 to 0.01147, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0099 - dice_coef: 0.4593 - accuracy: 0.9990 - val_loss: 0.0115 - val_dice_coef: 0.3478 - val_acc
Epoch 17/20
37/37 [==============================] - ETA: 0s - loss: 0.0088 - dice_coef: 0.4826 - accuracy: 0.9990
Epoch 00017: val_loss did not improve from 0.01147
37/37 [==============================] - 53s 1s/step - loss: 0.0088 - dice_coef: 0.4826 - accuracy: 0.9990 - val_loss: 0.0126 - val_dice_coef: 0.3683 - val_acc
Epoch 18/20
37/37 [==============================] - ETA: 0s - loss: 0.0079 - dice_coef: 0.5176 - accuracy: 0.9990
Epoch 00018: val_loss improved from 0.01147 to 0.01097, saving model to spleen_Exp1_Aug_10.h5
37/37 [==============================] - 53s 1s/step - loss: 0.0079 - dice_coef: 0.5176 - accuracy: 0.9990 - val_loss: 0.0110 - val_dice_coef: 0.3834 - val_acc
Epoch 19/20
37/37 [==============================] - ETA: 0s - loss: 0.0072 - dice_coef: 0.5442 - accuracy: 0.9990
Epoch 00019: val_loss did not improve from 0.01097
37/37 [==============================] - 53s 1s/step - loss: 0.0072 - dice_coef: 0.5442 - accuracy: 0.9990 - val_loss: 0.0126 - val_dice_coef: 0.4004 - val_acc
Epoch 20/20
37/37 [==============================] - ETA: 0s - loss: 0.0067 - dice_coef: 0.5546 - accuracy: 0.9990
Epoch 00020: val_loss did not improve from 0.01097
37/37 [==============================] - 53s 1s/step - loss: 0.0067 - dice_coef: 0.5546 - accuracy: 0.9990 - val_loss: 0.0128 - val_dice_coef: 0.3872 - val_acc
```

HyperParameters

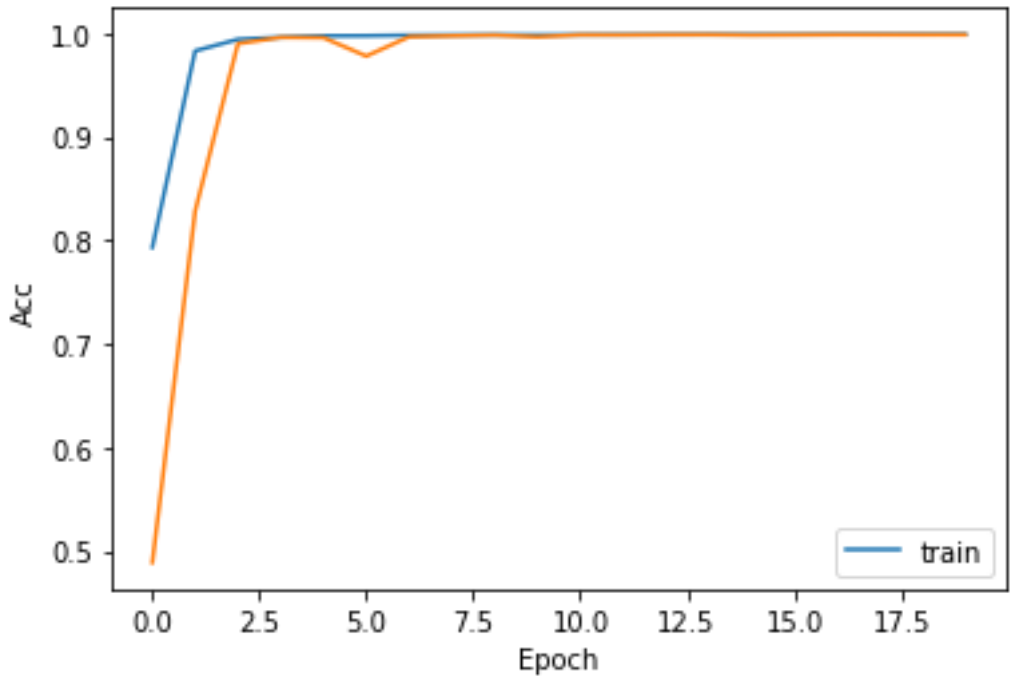
Loss Function: **Binary crossentropy loss function**, because more than two classes are there.

Activation Function: **Sigmoid** at the output layer and **ReLU** for the hidden layer.

Optimizer: **Adam optimizer** with learning rate 0.001

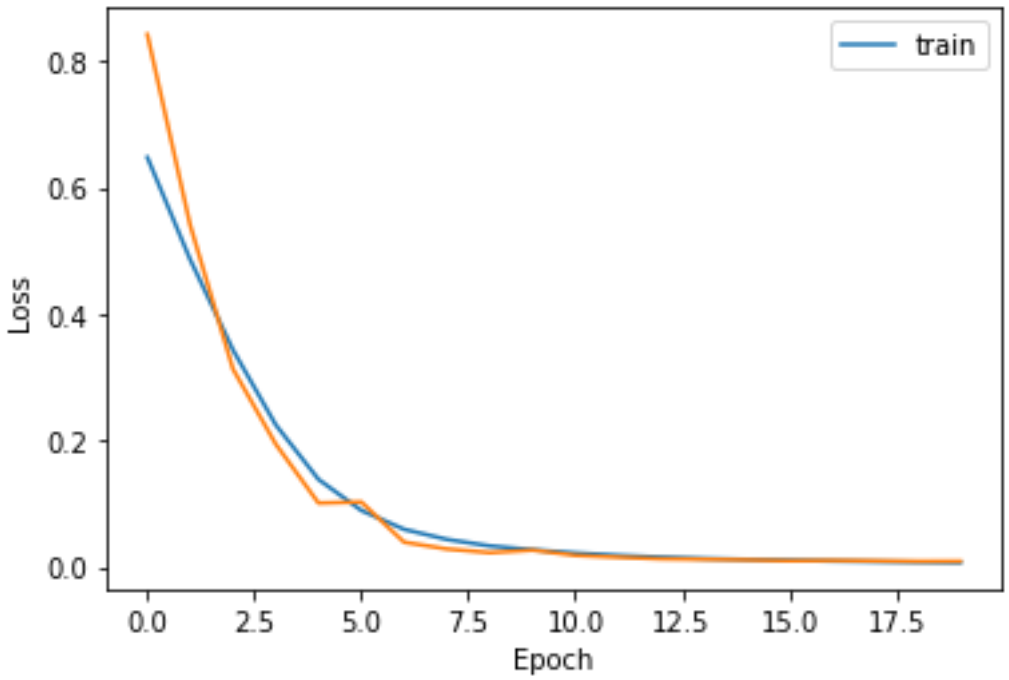
Accuracy Curve

X-axis: Epoch , Y-axis: Accuracy



Loss Curve

X-axis: Epoch , Y-axis: Loss



Result

Loss: 0.0026

```
Epoch 17/20
36/36 [==============================] - ETA: 0s - loss: 0.0028 - dice_coef: 0.7747 - accuracy:
Epoch 00017: val_loss did not improve from 0.00553
36/36 [==============================] - 41s 1s/step - loss: 0.0028 - dice_coef: 0.7747 - accuracy: 0.9990 - val_loss: 0
Epoch 18/20
36/36 [==============================] - ETA: 0s - loss: 0.0028 - dice_coef: 0.7699 - accuracy: 0.9991
Epoch 00018: val_loss did not improve from 0.00553
36/36 [==============================] - 41s 1s/step - loss: 0.0028 - dice_coef: 0.7699 - accuracy: 0.9991 - val_loss: 0
Epoch 19/20
36/36 [==============================] - ETA: 0s - loss: 0.0027 - dice_coef: 0.7734 - accuracy: 0.9990
Epoch 00019: val_loss did not improve from 0.00553
36/36 [==============================] - 46s 1s/step - loss: 0.0027 - dice_coef: 0.7734 - accuracy: 0.9990 - val_loss: 0
Epoch 20/20
36/36 [==============================] - ETA: 0s - loss: 0.0026 - dice_coef: 0.7847 - accuracy: 0.9991
Epoch 00020: val_loss did not improve from 0.00553
```

We got Training Accuracy: 0.9991

```
[ ] Testx = np.load('/content/drive/My Drive/Testx.npy')
    Testy = np.load('/content/drive/My Drive/Testy.npy')
    test_accuracy=model.evaluate(Testx,Testy)
    print(test_accuracy[2]*100)

    17/17 [==============================] - 6s 317ms/step - loss: -10.3862 - dice_coef: 1.0544 - accuracy: 0.7091
    70.91382145881653
```

We got Testing Accuracy: 70.913

```
[ ]
    test_pred = model.predict(test_X, batch_size=32) #saved-model-50.h5
    test_result = np.zeros(test_pred.shape)
    test_result[test_pred>0.5] = 1
    test_result[test_pred<=0.5] = 0
    dice = dc(test_result, test_y)
    pre  = precision(test_result,test_y)
    re   = recall(test_result,test_y)
    print('Test dc: '  + str(dice))
    print('Test pre: ' + str(pre))
    print('Test re: '  + str(re))

    Test dc: 0.7514473376775098
    Test pre: 0.9643297389141227
    Test re: 0.615558564370122
```

Observation and Conclusion

In this Assignment we have tried to segment the infectious part using the ground truth given to us. Our model is built from Scratch:

1. Testing Accuracy : 70.913 %
2. Test Dice: 0.75144
3. Test Precision : 0.9432
4. Test Recall: 0.61555