DS261: AI for Medical Image Analysis

Assignment 02 - Aman Pawar

MTech (1st Year), Sr. No. 22761

Department of Bioengineering, IISc

Task – I: Preparing the Dataset, Classifying the images into Mild, Severe, and Normal labels, and developing a ResNet Model for Classification.

Reading Data...

Resolution of CT images: (512, 512)

The number of CT images: 3554

Figure 1: Plotting CT Scan Data

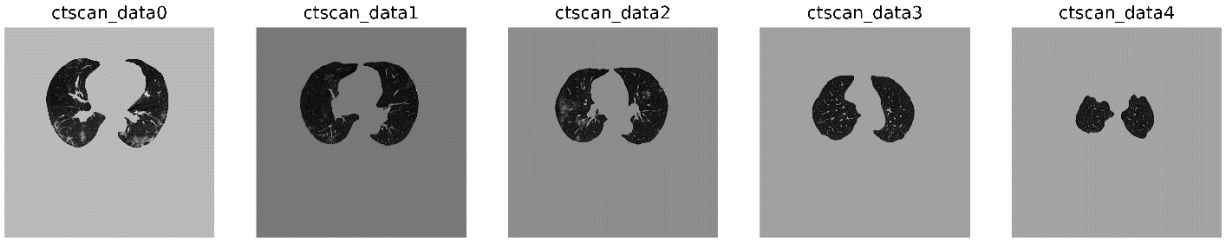


Figure 2: Corresponding Mask Data

Categorizing CT scans into three distinct groups Normal, Mild, and Severe based on the masks of the image:

Number of Normal CT: 1441

Number of Mild CT: 1954

Number of Severe CT: 159

From the above observations we can clearly see that there is class imbalance in terms of the Normal, Mils and Severe CT Scans, Necessary class imbalance techniques should be added in order to augment the data better before deep learning tasks, but since it is not mentioned in the question we are not doing it, will attempt to learn a good model with this class imbalance itself

ResNet model to classify the provided CT scans and report the performance metrics, including sensitivity, specificity, and accuracy.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 256, 256]	3,200
BatchNorm2d-2	[-1, 64, 256, 256]	128
ReLU-3	[-1, 64, 256, 256]	0
MaxPool2d-4	[-1, 64, 128, 128]	0
Conv2d-5	[-1, 64, 128, 128]	36,928
BatchNorm2d-6	[-1, 64, 128, 128]	128
ReLU-7	[-1, 64, 128, 128]	0
Conv2d-8	[-1, 64, 128, 128]	36,928
BatchNorm2d-9	[-1, 64, 128, 128]	128
ReLU-10	[-1, 64, 128, 128]	0
BasicBlock-11	[-1, 64, 128, 128]	0
Conv2d-12	[-1, 64, 128, 128]	36,928
BatchNorm2d-13	[-1, 64, 128, 128]	128
ReLU-14	[-1, 64, 128, 128]	0
Conv2d-15	[-1, 64, 128, 128]	36,928
BatchNorm2d-16	[-1, 64, 128, 128]	128
ReLU-17	[-1, 64, 128, 128]	0
BasicBlock-18	[-1, 64, 128, 128]	0
Conv2d-19	[-1, 128, 128, 128]	73,856
BatchNorm2d-20	[-1, 128, 128, 128]	256
ReLU-21	[-1, 128, 128, 128]	0
Conv2d-22	[-1, 128, 128, 128]	147,584
BatchNorm2d-23	[-1, 128, 128, 128]	256

Conv2d-24	*[-1, 128, 128, 128]*	*8,192*
BatchNorm2d-25	*[-1, 128, 128, 128]*	*256*
ReLU-26	*[-1, 128, 128, 128]*	*0*
BasicBlock-27	*[-1, 128, 128, 128]*	*0*
Conv2d-28	*[-1, 128, 128, 128]*	*147,584*
BatchNorm2d-29	*[-1, 128, 128, 128]*	*256*
ReLU-30	*[-1, 128, 128, 128]*	*0*
Conv2d-31	*[-1, 128, 128, 128]*	*147,584*
BatchNorm2d-32	*[-1, 128, 128, 128]*	*256*
ReLU-33	*[-1, 128, 128, 128]*	*0*
BasicBlock-34	*[-1, 128, 128, 128]*	*0*
Conv2d-35	*[-1, 256, 64, 64]*	*295,168*
BatchNorm2d-36	*[-1, 256, 64, 64]*	*512*
ReLU-37	*[-1, 256, 64, 64]*	*0*
Conv2d-38	*[-1, 256, 64, 64]*	*590,080*
BatchNorm2d-39	*[-1, 256, 64, 64]*	*512*
Conv2d-40	*[-1, 256, 64, 64]*	*32,768*
BatchNorm2d-41	*[-1, 256, 64, 64]*	*512*
ReLU-42	*[-1, 256, 64, 64]*	*0*
BasicBlock-43	*[-1, 256, 64, 64]*	*0*
Conv2d-44	*[-1, 256, 64, 64]*	*590,080*
BatchNorm2d-45	*[-1, 256, 64, 64]*	*512*
ReLU-46	*[-1, 256, 64, 64]*	*0*
Conv2d-47	*[-1, 256, 64, 64]*	*590,080*
BatchNorm2d-48	*[-1, 256, 64, 64]*	*512*
ReLU-49	*[-1, 256, 64, 64]*	*0*
BasicBlock-50	*[-1, 256, 64, 64]*	*0*
Conv2d-51	*[-1, 512, 32, 32]*	*1,180,160*
BatchNorm2d-52	*[-1, 512, 32, 32]*	*1,024*
ReLU-53	*[-1, 512, 32, 32]*	*0*
Conv2d-54	*[-1, 512, 32, 32]*	*2,359,808*
BatchNorm2d-55	*[-1, 512, 32, 32]*	*1,024*

Conv2d-56 [-1, 512, 32, 32] 131,072

BatchNorm2d-57 [-1, 512, 32, 32] 1,024

ReLU-58 [-1, 512, 32, 32] 0

BasicBlock-59 [-1, 512, 32, 32] 0

Conv2d-60 [-1, 512, 32, 32] 2,359,808

BatchNorm2d-61 [-1, 512, 32, 32] 1,024

ReLU-62 [-1, 512, 32, 32] 0

Conv2d-63 [-1, 512, 32, 32] 2,359,808

BatchNorm2d-64 [-1, 512, 32, 32] 1,024

ReLU-65 [-1, 512, 32, 32] 0

BasicBlock-66 [-1, 512, 32, 32] 0

AdaptiveAvgPool2d-67 [-1, 512, 1, 1] 0

Linear-68 [-1, 3] 1,539

==========

Total params: 11,175,683

Trainable params: 11,175,683

Non-trainable params: 0

Input size (MB): 1.00

Forward/backward pass size (MB): 664.00

Params size (MB): 42.63

Estimated Total Size (MB): 707.64

Figure 3: Predicted Labels of the image during Training

Epoch 48/Training: 100%| 10/10 [00:15<00:00, 1.53s/it, accuracy=81.4, f1_score=0.8142077, loss=0.816]

Epoch 48/Validation: 100%| 2/2 [00:01<00:00, 1.75it/s, accuracy=91.9, f1_score=0.9191919, loss=0.158]

Epoch 49/Training: 100%|[redacted]| 10/10 [00:15<00:00, 1.52s/it, accuracy=91.8, f1_score=0.91803277, loss=0.817]

Epoch 49/Validation: 100%|[redacted]| 2/2 [00:01<00:00, 1.91it/s, accuracy=90.9, f1_score=0.90909094, loss=0.159]

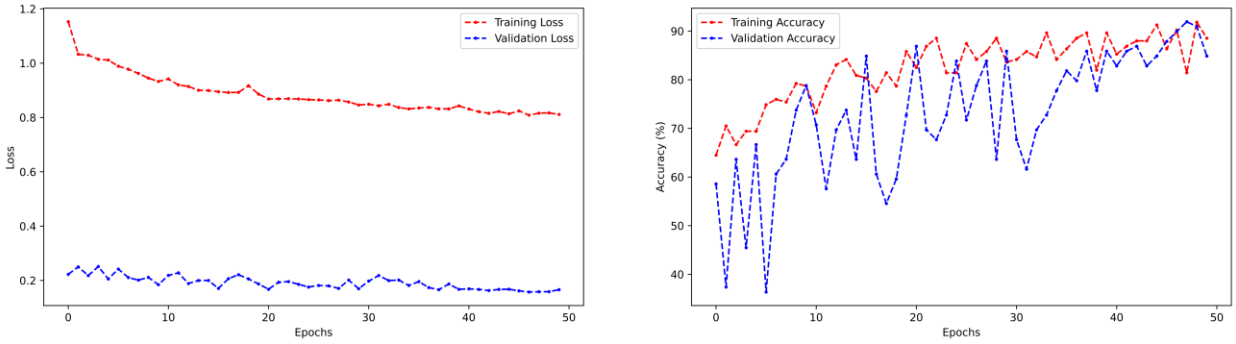
Epoch 50/Training: 100%|[redacted]| 10/10 [00:15<00:00, 1.51s/it, accuracy=88.5, f1_score=0.8852459, loss=0.811]

Epoch 50/Validation: 100%|[redacted]| 2/2 [00:01<00:00, 1.73it/s, accuracy=84.8, f1_score=0.8484849, loss=0.166]

CPU times: user 1h 20min 34s, sys: 3h 3min 56s, total: 4h 24min 31s

Wall time: 13min 56s

Figure 4: Model Loss and Accuracy for Training and Validation Data



Testing: 100%|[redacted]| 3/3 [00:01<00:00, 2.71it/s]

Figure 5: Prediction Outputs from the Test Dataset

Test Dataset:

Accuracy: 88.06%

Sensitivity: 0.92

Specificity: 0.92

CPU times: user 24 s, sys: 53 s, total: 1min 17s

Wall time: 4.6 s

Task – II: U-Net Model for Segmentation of the CT images

Model Summary

```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 512, 512]             288
       BatchNorm2d-2         [-1, 32, 512, 512]              64
              ReLU-3         [-1, 32, 512, 512]               0
            Conv2d-4         [-1, 32, 512, 512]           9,216
       BatchNorm2d-5         [-1, 32, 512, 512]              64
              ReLU-6         [-1, 32, 512, 512]               0
         MaxPool2d-7         [-1, 32, 256, 256]               0
            Conv2d-8         [-1, 64, 256, 256]          18,432
       BatchNorm2d-9         [-1, 64, 256, 256]             128
             ReLU-10         [-1, 64, 256, 256]               0
           Conv2d-11         [-1, 64, 256, 256]          36,864
      BatchNorm2d-12         [-1, 64, 256, 256]             128
             ReLU-13         [-1, 64, 256, 256]               0
        MaxPool2d-14         [-1, 64, 128, 128]               0
           Conv2d-15         [-1, 128, 128, 128]         73,728
      BatchNorm2d-16         [-1, 128, 128, 128]            256
             ReLU-17         [-1, 128, 128, 128]              0
           Conv2d-18         [-1, 128, 128, 128]        147,456
      BatchNorm2d-19         [-1, 128, 128, 128]            256
             ReLU-20         [-1, 128, 128, 128]              0
        MaxPool2d-21         [-1, 128, 64, 64]                0
           Conv2d-22         [-1, 256, 64, 64]          294,912
      BatchNorm2d-23         [-1, 256, 64, 64]              512
             ReLU-24         [-1, 256, 64, 64]                0
           Conv2d-25         [-1, 256, 64, 64]          589,824
      BatchNorm2d-26         [-1, 256, 64, 64]              512
```

ReLU-27 [-1, 256, 64, 64] 0

MaxPool2d-28 [-1, 256, 32, 32] 0

Conv2d-29 [-1, 512, 32, 32] 1,179,648

BatchNorm2d-30 [-1, 512, 32, 32] 1,024

ReLU-31 [-1, 512, 32, 32] 0

Conv2d-32 [-1, 512, 32, 32] 2,359,296

BatchNorm2d-33 [-1, 512, 32, 32] 1,024

ReLU-34 [-1, 512, 32, 32] 0

ConvTranspose2d-35 [-1, 256, 64, 64] 524,544

Conv2d-36 [-1, 256, 64, 64] 1,179,648

BatchNorm2d-37 [-1, 256, 64, 64] 512

ReLU-38 [-1, 256, 64, 64] 0

Conv2d-39 [-1, 256, 64, 64] 589,824

BatchNorm2d-40 [-1, 256, 64, 64] 512

ReLU-41 [-1, 256, 64, 64] 0

ConvTranspose2d-42 [-1, 128, 128, 128] 131,200

Conv2d-43 [-1, 128, 128, 128] 294,912

BatchNorm2d-44 [-1, 128, 128, 128] 256

ReLU-45 [-1, 128, 128, 128] 0

Conv2d-46 [-1, 128, 128, 128] 147,456

BatchNorm2d-47 [-1, 128, 128, 128] 256

ReLU-48 [-1, 128, 128, 128] 0

ConvTranspose2d-49 [-1, 64, 256, 256] 32,832

Conv2d-50 [-1, 64, 256, 256] 73,728

BatchNorm2d-51 [-1, 64, 256, 256] 128

ReLU-52 [-1, 64, 256, 256] 0

Conv2d-53 [-1, 64, 256, 256] 36,864

BatchNorm2d-54 [-1, 64, 256, 256] 128

ReLU-55 [-1, 64, 256, 256] 0

ConvTranspose2d-56 [-1, 32, 512, 512] 8,224

Conv2d-57 [-1, 32, 512, 512] 18,432

BatchNorm2d-58 [-1, 32, 512, 512] 64

ReLU-59 [-1, 32, 512, 512] 0

Conv2d-60 [-1, 32, 512, 512] 9,216

BatchNorm2d-61 [-1, 32, 512, 512] 64

ReLU-62 [-1, 32, 512, 512] 0

Conv2d-63 [-1, 3, 512, 512] 99

==========

Total params: 7,762,531

Trainable params: 7,762,531

Non-trainable params: 0

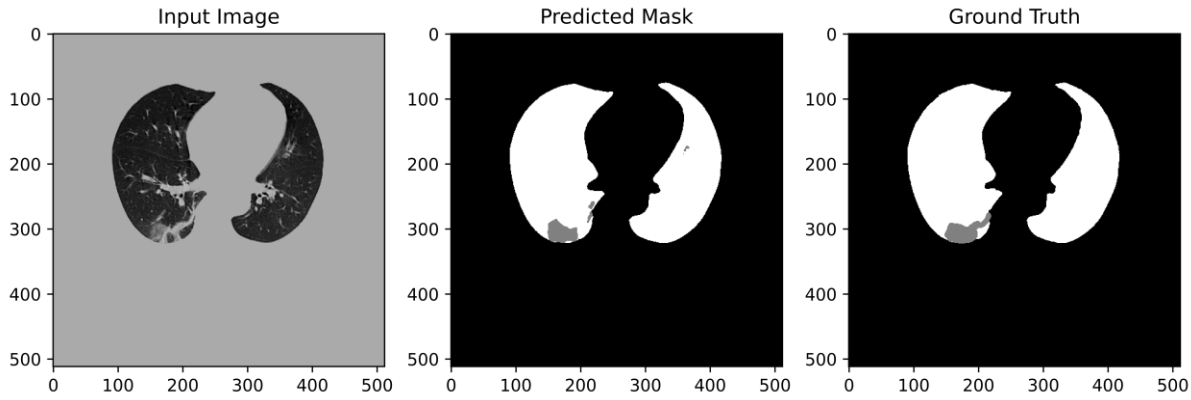
Input size (MB): 1.00

Forward/backward pass size (MB): 1620.00

Params size (MB): 29.61

Estimated Total Size (MB): 1650.61

Figure 6: Segmented Image sample during Training



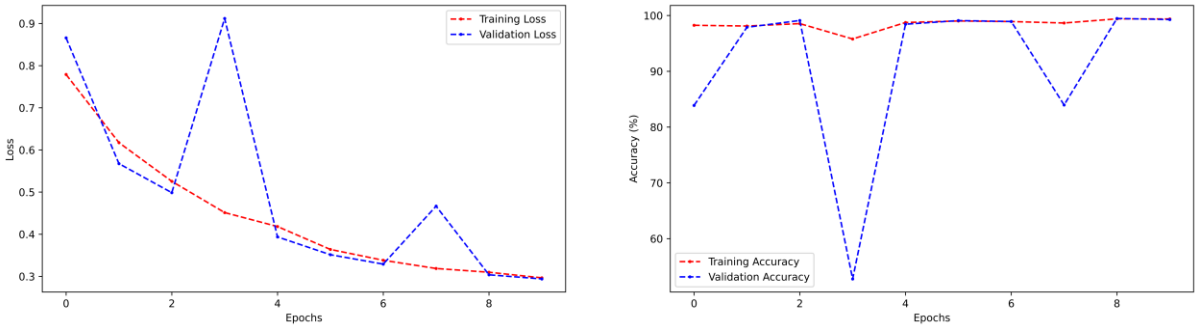
Epoch 10/Training: 100%|| 26/26 [00:44<00:00, 1.71s/it, accuracy=99.4, dice_score=0.9938103, loss=0.297]

Epoch 10/Validation: 100%|| 4/4 [00:02<00:00, 1.60it/s, accuracy=99.3, dice_score=0.9926988, loss=0.294]

CPU times: user 44min 25s, sys: 1h 42min 9s, total: 2h 26min 35s

Wall time: 8min 13s

Figure 7: Training and Validation Loss and Accuracy for the Model



Testing: 100%| 8/8 [01:11<00:00, 8.92s/it]

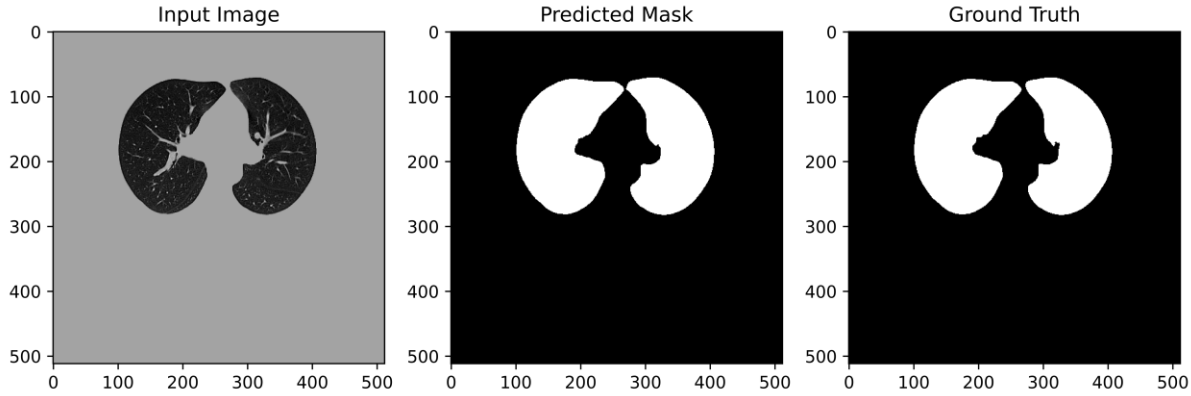
Accuracy: 99.39%

Sensitivity: 0.99

Specificity: 1.00

Avg Dice Score: 0.99

Figure 8: Segmentation of Test Set



CPU times: user 4min 4s, sys: 3.87 s, total: 4min 7s

Wall time: 1min 14s

Task – III: Utilize the predicted masks from Task 2 to classify CT scans into three distinct groups Normal, Mild, and Severe

After segmenting and running

Testing: 100%| 14/14 [00:17<00:00, 1.27s/it]

Number of Normal CT: 1364

Number of Mild CT: 2045

Number of Severe CT: 145

CPU times: user 8min 20s, sys: 16.4 s, total: 8min 36s, Wall time: 17.8s

Tasks – IV: (Part-I) Comparing the Classification model and Segmentation mode.

The Classification Model achieved an accuracy of Accuracy: ***88.06%,*** *a Sensitivity* ***of 0.92****, Specificity of* ***0.92*** *on the Test set. The Segmentation Model achieved an Accuracy of* ***99.39%*** *Sensitivity of* ***0.99****, a Specificity of* ***1.00****, and an average dice Score of* ***0.99*** *on the Test Set.*

The Training wall time of the classification model took a wall time of ***13 minutes 56 seconds****, while the training wall time of the segmentation model took a wall time of* ***8 minutes 13 seconds****.*

The classification model included ***Total params: 11,175,683****, Forward/backward pass size (MB): 664.00, Params size (MB): 42.63, and Estimated Total Size (MB): 707.64. The segmentation model included* ***Total params: 7,762,531****, Forward/backward pass size (MB): 1620.00, Params size (MB): 29.61and Estimated Total Size (MB): 1650.61.*

The classification model took a wall time of ***4.6 s for inference*** *whereas the segmentation model took a wall time of* ***17.8 s for inference*** *on the test set.*

(Part-II) Improving the Classification Model

To improve the classification model use of shuffleNet is done, as it has the following advantages over normal ResNet.

Channel Shuffle*: ShuffleNetV2 introduces the concept of channel shuffling. It divides the input channels into multiple groups and then shuffles the channels within each group. This enhances information flow between different groups and facilitates efficient feature reuse.*

Depthwise Separable Convolution*: ShuffleNetV2 employs depth wise separable convolutions, which consist of depth wise convolution (separately convolving each channel) and pointwise convolution (combining the channels). This reduces computation and parameters significantly.*

Group Convolution*: Group convolution is used in both pointwise and depth wise convolutions. Group convolution allows parallel processing and reduces computational load, making it more efficient.*

Bottleneck Blocks*: ShuffleNetV2 uses bottleneck blocks that further reduce the number of parameters while maintaining network capacity. These blocks consist of pointwise convolutions and are instrumental in preserving accuracy.*

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 24, 256, 256]	1,200
MaxPool2d-2	[-1, 24, 128, 128]	0
Conv2d-3	[-1, 24, 64, 64]	216
BatchNorm2d-4	[-1, 24, 64, 64]	48
Conv2d-5	[-1, 24, 64, 64]	576
BatchNorm2d-6	[-1, 24, 64, 64]	48
ReLU-7	[-1, 24, 64, 64]	0
Conv2d-8	[-1, 24, 128, 128]	576

BatchNorm2d-9	*[-1, 24, 128, 128]*	*48*
ReLU-10	*[-1, 24, 128, 128]*	*0*
Conv2d-11	*[-1, 24, 64, 64]*	*216*
BatchNorm2d-12	*[-1, 24, 64, 64]*	*48*
Conv2d-13	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-14	*[-1, 24, 64, 64]*	*48*
ReLU-15	*[-1, 24, 64, 64]*	*0*
InvertedResidual-16	*[-1, 48, 64, 64]*	*0*
Conv2d-17	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-18	*[-1, 24, 64, 64]*	*48*
ReLU-19	*[-1, 24, 64, 64]*	*0*
Conv2d-20	*[-1, 24, 64, 64]*	*216*
BatchNorm2d-21	*[-1, 24, 64, 64]*	*48*
Conv2d-22	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-23	*[-1, 24, 64, 64]*	*48*
ReLU-24	*[-1, 24, 64, 64]*	*0*
InvertedResidual-25	*[-1, 48, 64, 64]*	*0*
Conv2d-26	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-27	*[-1, 24, 64, 64]*	*48*
ReLU-28	*[-1, 24, 64, 64]*	*0*
Conv2d-29	*[-1, 24, 64, 64]*	*216*
BatchNorm2d-30	*[-1, 24, 64, 64]*	*48*
Conv2d-31	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-32	*[-1, 24, 64, 64]*	*48*
ReLU-33	*[-1, 24, 64, 64]*	*0*
InvertedResidual-34	*[-1, 48, 64, 64]*	*0*
Conv2d-35	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-36	*[-1, 24, 64, 64]*	*48*
ReLU-37	*[-1, 24, 64, 64]*	*0*
Conv2d-38	*[-1, 24, 64, 64]*	*216*
BatchNorm2d-39	*[-1, 24, 64, 64]*	*48*
Conv2d-40	*[-1, 24, 64, 64]*	*576*
BatchNorm2d-41	*[-1, 24, 64, 64]*	*48*

ReLU-42	*[-1, 24, 64, 64]*	*0*
InvertedResidual-43	*[-1, 48, 64, 64]*	*0*
Conv2d-44	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-45	*[-1, 48, 32, 32]*	*96*
Conv2d-46	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-47	*[-1, 48, 32, 32]*	*96*
ReLU-48	*[-1, 48, 32, 32]*	*0*
Conv2d-49	*[-1, 48, 64, 64]*	*2,304*
BatchNorm2d-50	*[-1, 48, 64, 64]*	*96*
ReLU-51	*[-1, 48, 64, 64]*	*0*
Conv2d-52	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-53	*[-1, 48, 32, 32]*	*96*
Conv2d-54	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-55	*[-1, 48, 32, 32]*	*96*
ReLU-56	*[-1, 48, 32, 32]*	*0*
InvertedResidual-57	*[-1, 96, 32, 32]*	*0*
Conv2d-58	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-59	*[-1, 48, 32, 32]*	*96*
ReLU-60	*[-1, 48, 32, 32]*	*0*
Conv2d-61	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-62	*[-1, 48, 32, 32]*	*96*
Conv2d-63	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-64	*[-1, 48, 32, 32]*	*96*
ReLU-65	*[-1, 48, 32, 32]*	*0*
InvertedResidual-66	*[-1, 96, 32, 32]*	*0*
Conv2d-67	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-68	*[-1, 48, 32, 32]*	*96*
ReLU-69	*[-1, 48, 32, 32]*	*0*
Conv2d-70	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-71	*[-1, 48, 32, 32]*	*96*
Conv2d-72	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-73	*[-1, 48, 32, 32]*	*96*
ReLU-74	*[-1, 48, 32, 32]*	*0*

InvertedResidual-75	*[-1, 96, 32, 32]*	*0*
Conv2d-76	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-77	*[-1, 48, 32, 32]*	*96*
ReLU-78	*[-1, 48, 32, 32]*	*0*
Conv2d-79	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-80	*[-1, 48, 32, 32]*	*96*
Conv2d-81	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-82	*[-1, 48, 32, 32]*	*96*
ReLU-83	*[-1, 48, 32, 32]*	*0*
InvertedResidual-84	*[-1, 96, 32, 32]*	*0*
Conv2d-85	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-86	*[-1, 48, 32, 32]*	*96*
ReLU-87	*[-1, 48, 32, 32]*	*0*
Conv2d-88	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-89	*[-1, 48, 32, 32]*	*96*
Conv2d-90	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-91	*[-1, 48, 32, 32]*	*96*
ReLU-92	*[-1, 48, 32, 32]*	*0*
InvertedResidual-93	*[-1, 96, 32, 32]*	*0*
Conv2d-94	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-95	*[-1, 48, 32, 32]*	*96*
ReLU-96	*[-1, 48, 32, 32]*	*0*
Conv2d-97	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-98	*[-1, 48, 32, 32]*	*96*
Conv2d-99	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-100	*[-1, 48, 32, 32]*	*96*
ReLU-101	*[-1, 48, 32, 32]*	*0*
InvertedResidual-102	*[-1, 96, 32, 32]*	*0*
Conv2d-103	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-104	*[-1, 48, 32, 32]*	*96*
ReLU-105	*[-1, 48, 32, 32]*	*0*
Conv2d-106	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-107	*[-1, 48, 32, 32]*	*96*

Conv2d-108	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-109	*[-1, 48, 32, 32]*	*96*
ReLU-110	*[-1, 48, 32, 32]*	*0*
InvertedResidual-111	*[-1, 96, 32, 32]*	*0*
Conv2d-112	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-113	*[-1, 48, 32, 32]*	*96*
ReLU-114	*[-1, 48, 32, 32]*	*0*
Conv2d-115	*[-1, 48, 32, 32]*	*432*
BatchNorm2d-116	*[-1, 48, 32, 32]*	*96*
Conv2d-117	*[-1, 48, 32, 32]*	*2,304*
BatchNorm2d-118	*[-1, 48, 32, 32]*	*96*
ReLU-119	*[-1, 48, 32, 32]*	*0*
InvertedResidual-120	*[-1, 96, 32, 32]*	*0*
Conv2d-121	*[-1, 96, 16, 16]*	*864*
BatchNorm2d-122	*[-1, 96, 16, 16]*	*192*
Conv2d-123	*[-1, 96, 16, 16]*	*9,216*
BatchNorm2d-124	*[-1, 96, 16, 16]*	*192*
ReLU-125	*[-1, 96, 16, 16]*	*0*
Conv2d-126	*[-1, 96, 32, 32]*	*9,216*
BatchNorm2d-127	*[-1, 96, 32, 32]*	*192*
ReLU-128	*[-1, 96, 32, 32]*	*0*
Conv2d-129	*[-1, 96, 16, 16]*	*864*
BatchNorm2d-130	*[-1, 96, 16, 16]*	*192*
Conv2d-131	*[-1, 96, 16, 16]*	*9,216*
BatchNorm2d-132	*[-1, 96, 16, 16]*	*192*
ReLU-133	*[-1, 96, 16, 16]*	*0*
InvertedResidual-134	*[-1, 192, 16, 16]*	*0*
Conv2d-135	*[-1, 96, 16, 16]*	*9,216*
BatchNorm2d-136	*[-1, 96, 16, 16]*	*192*
ReLU-137	*[-1, 96, 16, 16]*	*0*
Conv2d-138	*[-1, 96, 16, 16]*	*864*
BatchNorm2d-139	*[-1, 96, 16, 16]*	*192*
Conv2d-140	*[-1, 96, 16, 16]*	*9,216*

BatchNorm2d-141 [-1, 96, 16, 16] 192

ReLU-142 [-1, 96, 16, 16] 0

InvertedResidual-143 [-1, 192, 16, 16] 0

Conv2d-144 [-1, 96, 16, 16] 9,216

BatchNorm2d-145 [-1, 96, 16, 16] 192

ReLU-146 [-1, 96, 16, 16] 0

Conv2d-147 [-1, 96, 16, 16] 864

BatchNorm2d-148 [-1, 96, 16, 16] 192

Conv2d-149 [-1, 96, 16, 16] 9,216

BatchNorm2d-150 [-1, 96, 16, 16] 192

ReLU-151 [-1, 96, 16, 16] 0

InvertedResidual-152 [-1, 192, 16, 16] 0

Conv2d-153 [-1, 96, 16, 16] 9,216

BatchNorm2d-154 [-1, 96, 16, 16] 192

ReLU-155 [-1, 96, 16, 16] 0

Conv2d-156 [-1, 96, 16, 16] 864

BatchNorm2d-157 [-1, 96, 16, 16] 192

Conv2d-158 [-1, 96, 16, 16] 9,216

BatchNorm2d-159 [-1, 96, 16, 16] 192

ReLU-160 [-1, 96, 16, 16] 0

InvertedResidual-161 [-1, 192, 16, 16] 0

Conv2d-162 [-1, 1024, 16, 16] 196,608

BatchNorm2d-163 [-1, 1024, 16, 16] 2,048

ReLU-164 [-1, 1024, 16, 16] 0

Linear-165 [-1, 3] 3,075

ShuffleNetV2-166 [-1, 3] 0

==========

Total params: 345,371

Trainable params: 345,371

Non-trainable params: 0

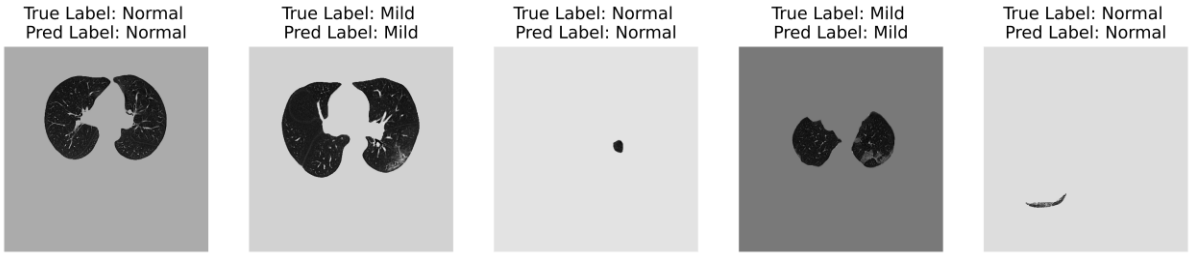
Input size (MB): 1.00

Forward/backward pass size (MB): 106.88

Params size (MB): 1.32

Estimated Total Size (MB): 109.19

Figure 9: Prediction of the Labels during Training



Epoch 46/Training: 100%|[redacted]| 5/5 [00:05<00:00, 1.14s/it, accuracy=99.8, f1_score=0.9977221, loss=0.643]

Epoch 46/Validation: 100%|[redacted]| 1/1 [00:00<00:00, 1.70it/s, accuracy=94.9, f1_score=0.94929576, loss=0.14]

Epoch 47/Training: 100%|[redacted]| 5/5 [00:05<00:00, 1.04s/it, accuracy=100, f1_score=1.0, loss=0.642]

Epoch 47/Validation: 100%|[redacted]| 1/1 [00:00<00:00, 1.95it/s, accuracy=95.2, f1_score=0.9521127, loss=0.14]

Epoch 48/Training: 100%|[redacted]| 5/5 [00:05<00:00, 1.10s/it, accuracy=99.8, f1_score=0.9977221, loss=0.642]

Epoch 48/Validation: 100%|[redacted]| 1/1 [00:00<00:00, 1.62it/s, accuracy=95.2, f1_score=0.9521127, loss=0.139]

Epoch 49/Training: 100%|[redacted]| 5/5 [00:05<00:00, 1.09s/it, accuracy=99.5, f1_score=0.9954442, loss=0.643]

Epoch 49/Validation: 100%|[redacted]| 1/1 [00:00<00:00, 3.01it/s, accuracy=95.8, f1_score=0.9577465, loss=0.139]

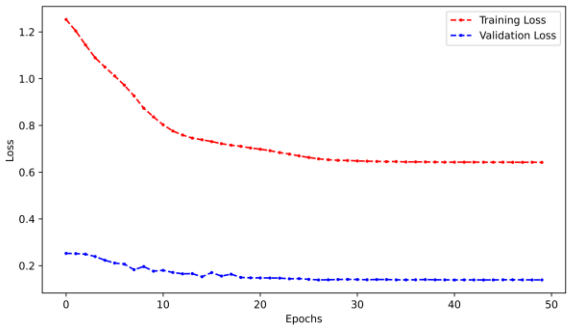
Epoch 50/Training: 100%|[redacted]| 5/5 [00:05<00:00, 1.16s/it, accuracy=100, f1_score=1.0, loss=0.642]

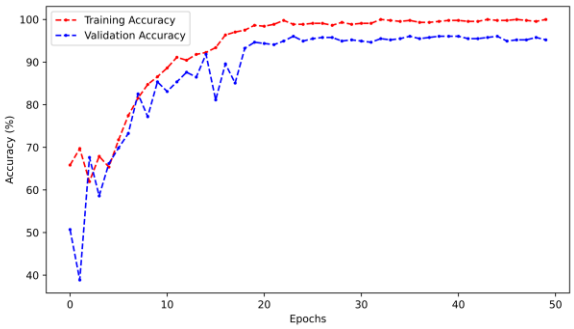
Epoch 50/Validation: 100%|[redacted]| 1/1 [00:00<00:00, 1.61it/s, accuracy=95.2, f1_score=0.9521127, loss=0.139]

CPU times: user 52min 47s, sys: 2h 8min 25s, total: 3h 1min 12s

Wall time: 5min 17s

Figure 10: Training and Validation Loss and Accuracy of the Model





Testing: 100%|[redacted]| 2/2 [00:00<00:00, 2.04it/s]

Test Dataset:

Accuracy: 95.08%

Sensitivity: 0.95

Specificity: 0.97

CPU times: user 24.7 s, sys: 57 s, total: 1min 21s

Wall time: 3.38 s

Figure 11: Images from the Test set, and the predicted labels

Task – V: The Chosen Model is the Segmentation Model

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 512, 512]	160
ReLU-2	[-1, 16, 512, 512]	0
Conv2d-3	[-1, 16, 512, 512]	2,320
ReLU-4	[-1, 16, 512, 512]	0
MaxPool2d-5	[-1, 16, 256, 256]	0
Conv2d-6	[-1, 16, 256, 256]	2,320
ReLU-7	[-1, 16, 256, 256]	0
Conv2d-8	[-1, 16, 256, 256]	2,320
ReLU-9	[-1, 16, 256, 256]	0
ConvTranspose2d-10	[-1, 16, 512, 512]	2,064
ReLU-11	[-1, 16, 512, 512]	0
Conv2d-12	[-1, 3, 512, 512]	51

(Part-I) Total Parameters of the Model

Total params: 9,235

Trainable params: 9,235

Non-trainable params: 0

(Part-II) Memory Consumed *Input size (MB): 1.00*

Forward/backward pass size (MB): 238.00

Param size (MB): 0.04

Estimated Total Size (MB): 239.04

(Part-III) Reducing the model Trainable Params

Pruning*: Pruning involves removing less important weights from a neural network. It can be done during or after training and significantly reduces the number of parameters while preserving performance.*

***Quantization**: Quantization reduces the bit-width of model weights and activations. For example, converting 32-bit floating-point values to 8-bit integers can greatly reduce the model's size and memory requirements while maintaining accuracy.*

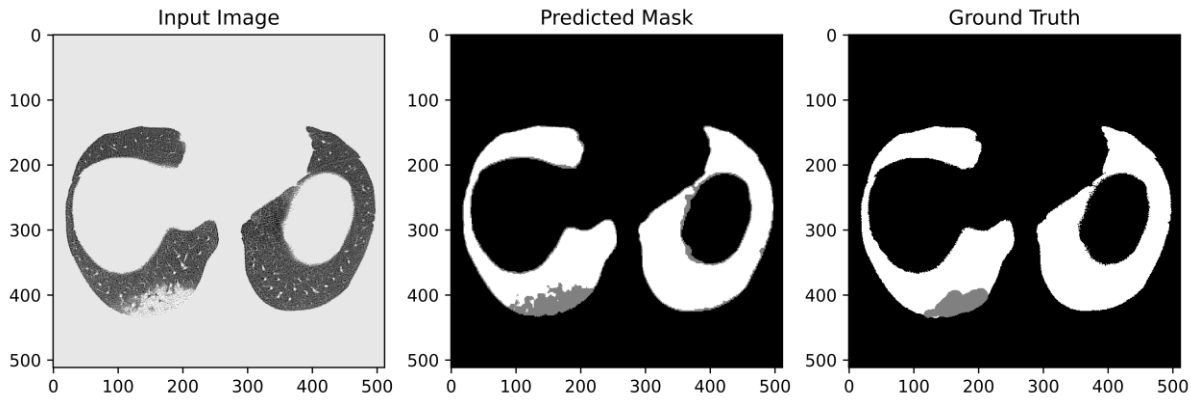
***Architectural Design**: Choose or design models with fewer layers or smaller filter sizes. Reducing the depth and width of a model can significantly decrease the number of parameters while accepting a trade-off in performance.*

***Dense-Free Convolution**: Implementing a dense-free convolutional neural network (ConvNet) architecture can lead to a substantial reduction in trainable parameters while maintaining adequate results.*

***Strided Convolutions**: Applying strided convolutions with a subsample parameter can reduce the number of parameters by downscaling the input data.*

*A modified version of the U-Net model **only one encoder and decoder**, kept **smaller kernel sizes** to reduce the number of parameters while preserving the basic U-Net structure.*

Figure 12: Output of the segmented image during the process of training.



Epoch 45/Training: 100%|| 10/10 [00:10<00:00, 1.04s/it, accuracy=97.6, dice_score=0.9755362, loss=0.353]

Epoch 45/Validation: 100%|| 2/2 [00:00<00:00, 2.31it/s, accuracy=98.3, dice_score=0.98308134, loss=0.351]

Epoch 46/Training: 100%|| 10/10 [00:10<00:00, 1.02s/it, accuracy=98.2, dice_score=0.9818435, loss=0.35]

Epoch 46/Validation: 100%|| 2/2 [00:00<00:00, 2.29it/s, accuracy=98.4, dice_score=0.98374146, loss=0.348]

Epoch 47/Training: 100%|| 10/10 [00:09<00:00, 1.01it/s, accuracy=97.7, dice_score=0.9771868, loss=0.349]

Epoch 47/Validation: 100%|| 2/2 [00:00<00:00, 2.30it/s, accuracy=98.4, dice_score=0.9837406, loss=0.346]

Epoch 48/Training: 100%|| 10/10 [00:10<00:00, 1.00s/it, accuracy=98.5, dice_score=0.9847103, loss=0.348]

Epoch 48/Validation: 100%|[redacted]| 2/2 [00:01<00:00, 1.89it/s, accuracy=98.3, dice_score=0.98307985, loss=0.352]

Epoch 49/Training: 100%|[redacted]| 10/10 [00:09<00:00, 1.02it/s, accuracy=98.7, dice_score=0.9872349, loss=0.349]

Epoch 49/Validation: 100%|[redacted]| 2/2 [00:01<00:00, 1.94it/s, accuracy=97.9, dice_score=0.9792739, loss=0.348]

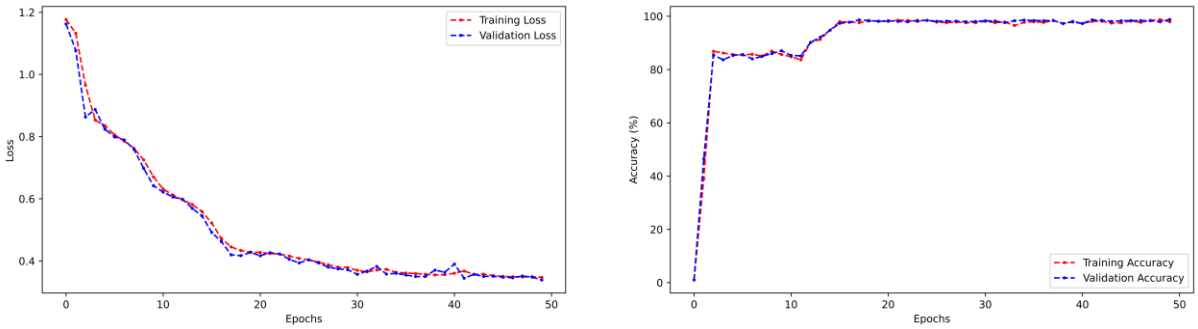
Epoch 50/Training: 100%|[redacted]| 10/10 [00:09<00:00, 1.00it/s, accuracy=97.9, dice_score=0.9792171, loss=0.347]

Epoch 50/Validation: 100%|[redacted]| 2/2 [00:00<00:00, 2.28it/s, accuracy=98.8, dice_score=0.9879852, loss=0.339]

CPU times: user 2h 1min 7s, sys: 4h 47min 58s, total: 6h 49min 5s

Wall time: 9min 27s

Figure 13: Model Loss and Accuracy for the training and validation sets



Testing: 100%|[redacted]| 3/3 [01:07<00:00, 22.34s/it]

Accuracy: 98.32%

Sensitivity: 0.96

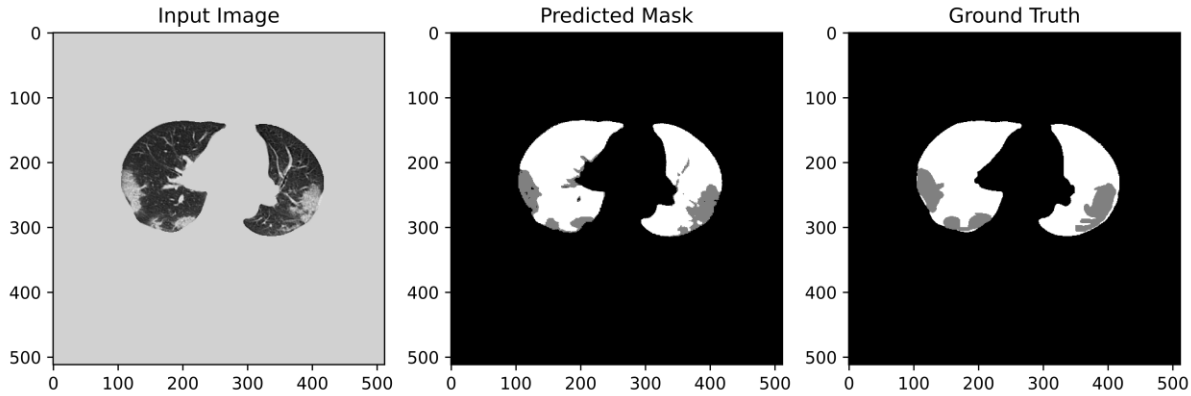
Specificity: 1.00

Avg Dice Score: 0.98

CPU times: user 1min 32s, sys: 56.2 s, total: 2min 28s

Wall time: 1min 9s

Figure 14: Outputs for the Test set of the segmented masks



(Part -IV Performance)

Yes, there is a performance drop in the model considering the scale at which we have done the parameter reduction as compared to the baseline model that we trained earlier. Also, we had to train this model for a larger number of epochs as compared to the earlier model.

Thank you 😊 ~ Aman Pawar.