Automatic Segmentation of COVID-19 CT Images using MultiRes-U-Net

By

Kumari Pooja, Yunke Li, Qi Yang

Under the guidance of

Dr. Abdul QAYYUM & Dr. Fabrice MERIAUDEAU

Universite De Bourgogne

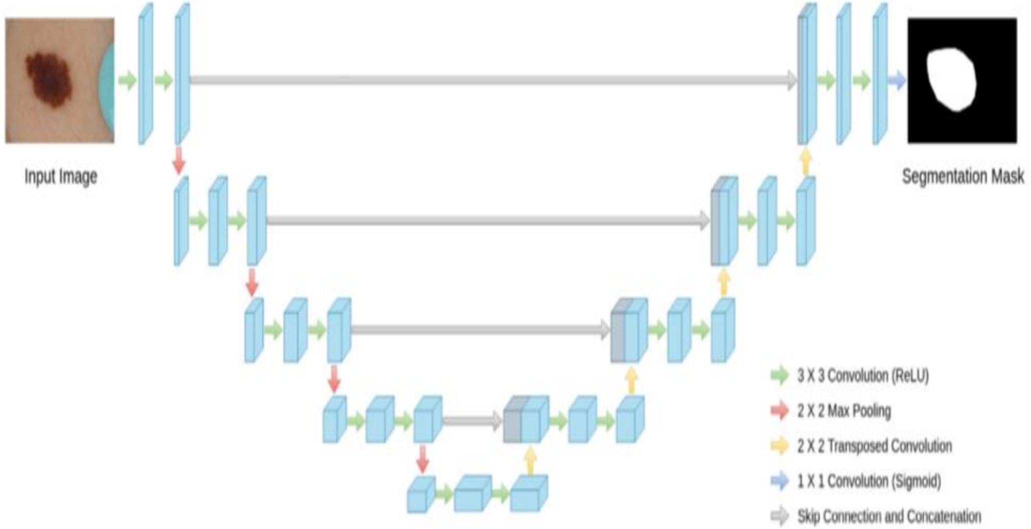
June 11, 2020

CONTENTS

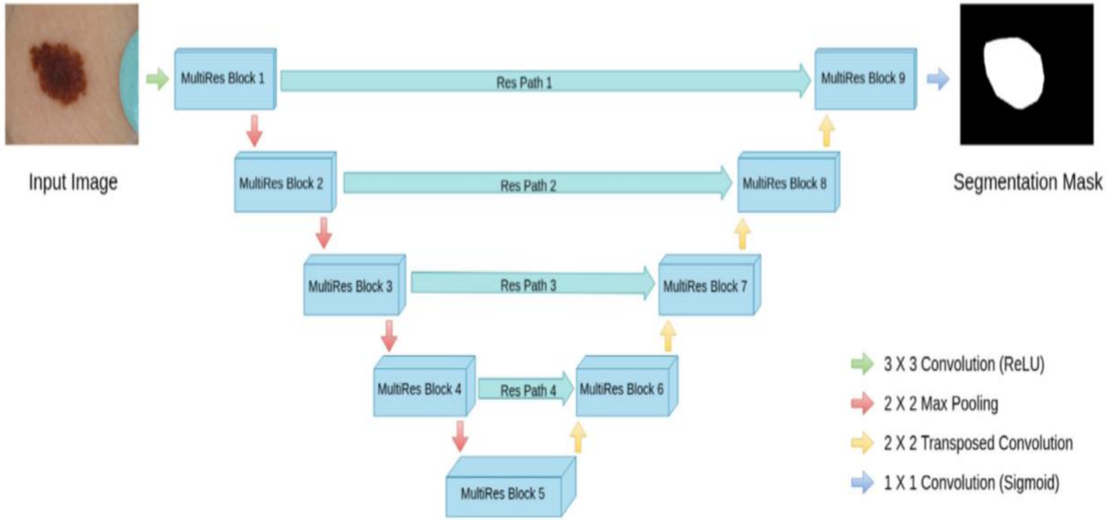
Introduction

- Accurate segmentation of lung and infection in COVID-19 CT scans plays an important role in the quantitative management of patients.
- In this project, we focused on $2D$ segmentation of right & left lung and the disease area using COVID-19 CT scans through Deep Learning Models. (supervised multi-label segmentation)
- U-Net is the most used model for medical image segmentation and in our case, we have used this base model along with concept of residual block.

Basic Model – MultiRes U-Net



U-Net



MultiRes U-Net

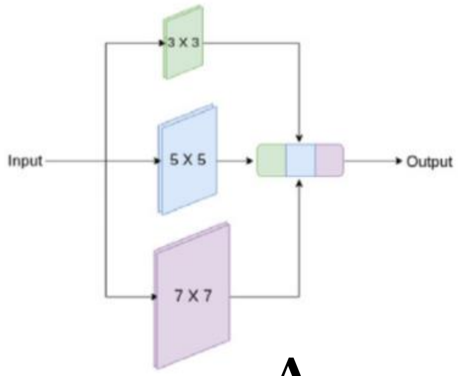
Motivation:

- ✓ Variation of scale in medical images.
- ✓ Probable semantic gap between the corresponding levels of encoder–decoder

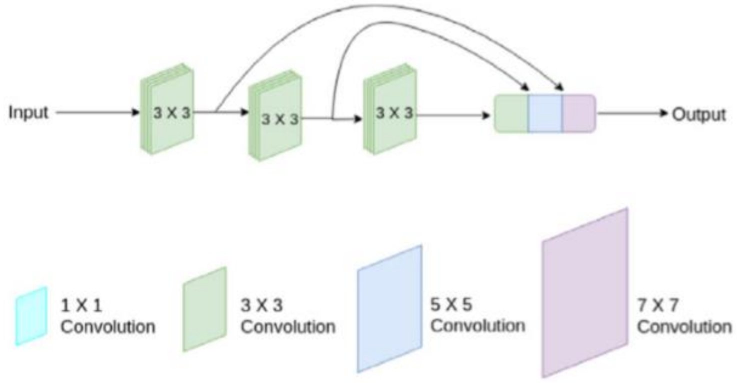
Basic Model – MultiRes Block

Motivation 1: Variation of scale in medical images.

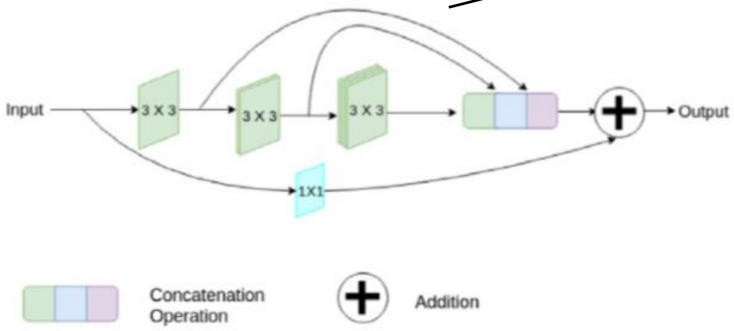
MultiRes block



A



B



C

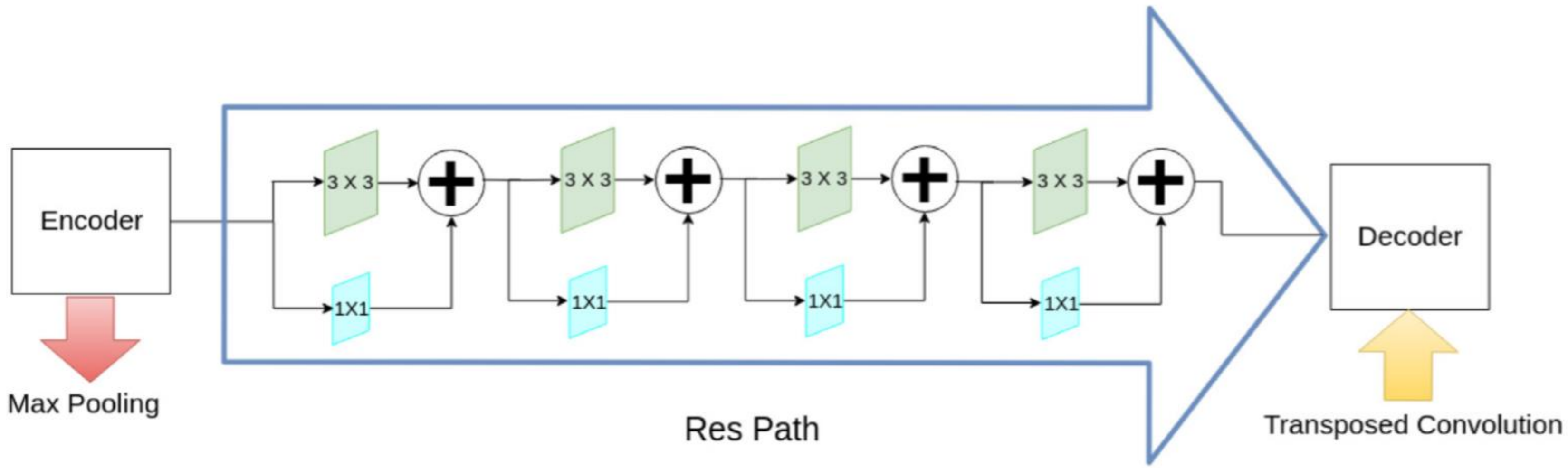
improvements:
Inception architecture
weakness:
Large memory requirement

improvements:
serial structure
weakness:
Large memory requirement

improvements:
gradually increase the filters in the three consecutive convolutional layers

Basic Model – Res Path

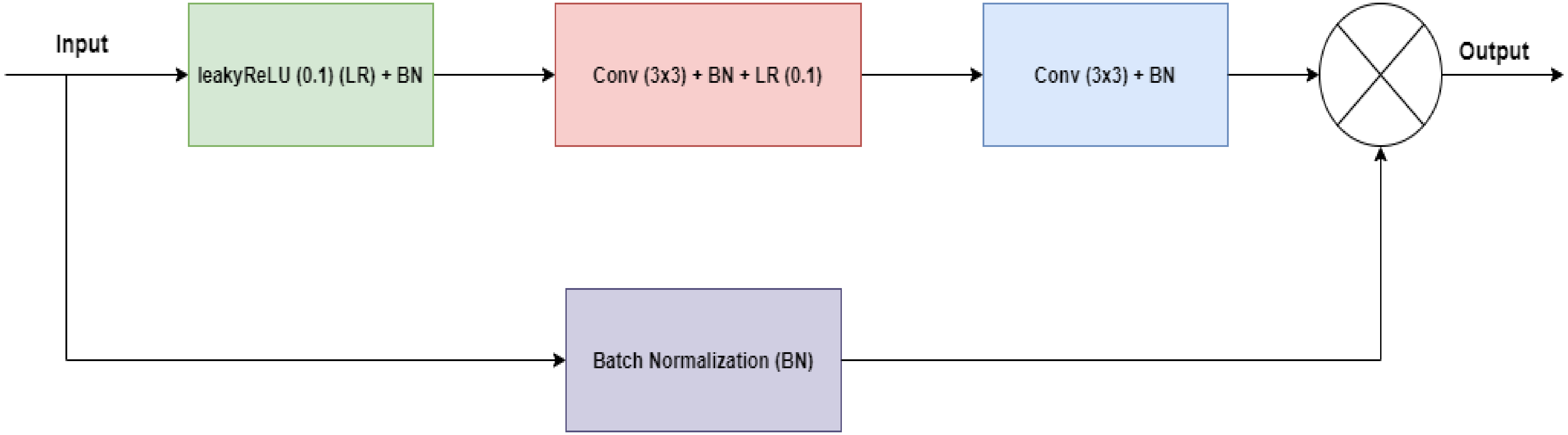
Motivation 2: Probable semantic gap between the corresponding levels of encoder–decoder



- Pass them through a chain of convolutional layers with residual connections
- Concatenate them with the decoder features

Modified Model – Residual Block

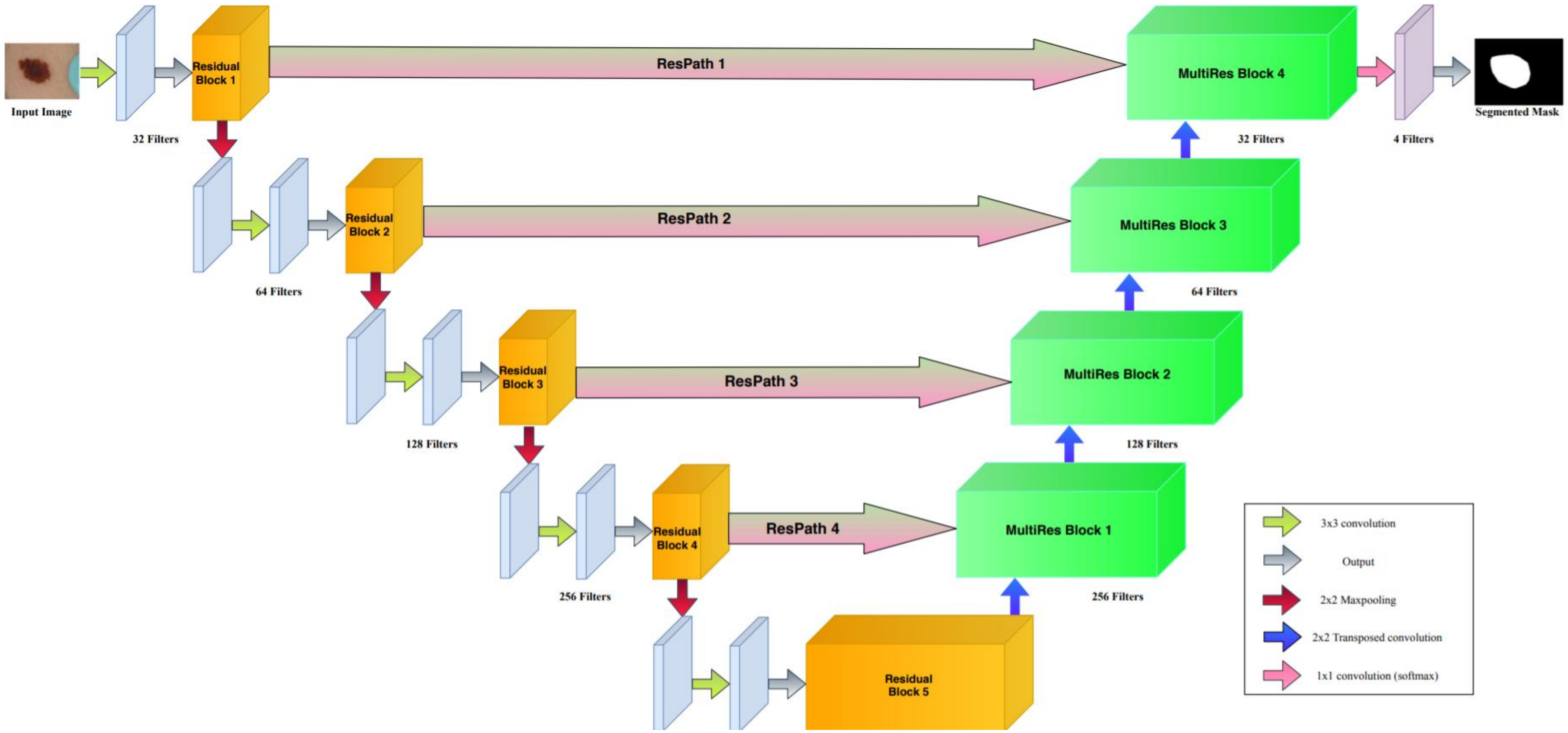
Motivation 3: Used the concept of skip connection along with Leaky ReLU and Batch Normalization in the encoder side.



Residual Block

- Add input of previous layer with the convoluted output.

Final Modified Architecture



Encoder has residual block and decoder has multi res block

Modifications - Regularization

```python
def convolution_block(x, filters, size, strides=(1,1), padding='same', activation=True):
    x = Conv2D(filters, size, strides=strides, padding=padding, kernel_regularizer=regularizers.l1_l2(0.00001, 0.00001))(x)
```

L_1 and L_2 regularization in convolution blocks

$$\ell_1 \; penalty = \ell_1 \sum_{i=0}^{n} |x_i|$$

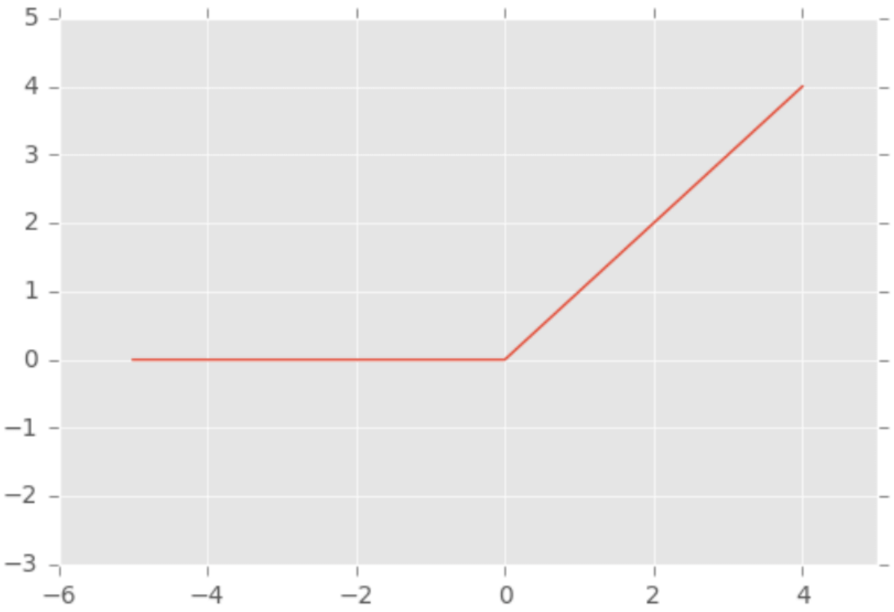
where $l_1 = 10^{-5}$;
$l_2 = 10^{-5}$;

$$\ell_2 \; penalty = \ell_2 \sum_{i=0}^{n} x_i^2$$

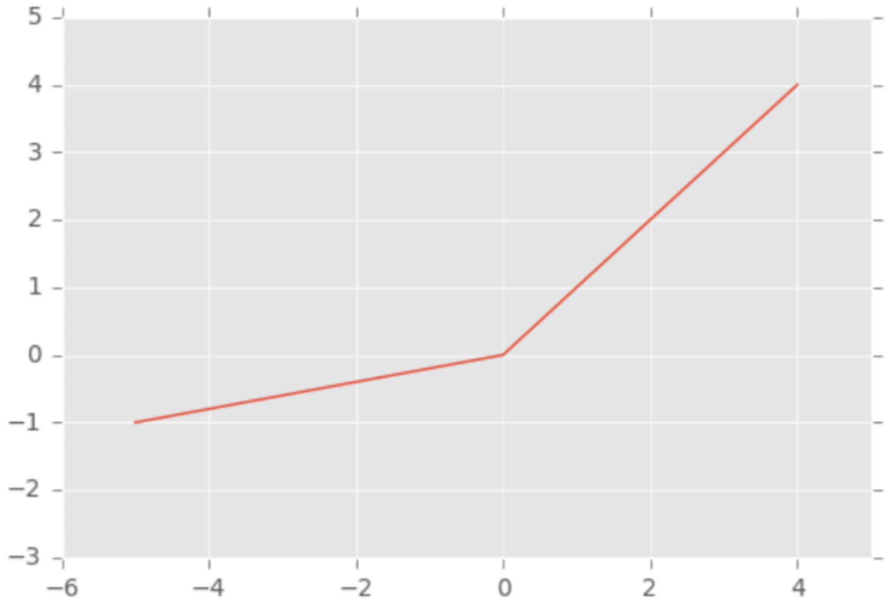
Modifications - Activation Function

```
    if activation == True:
        x = LeakyReLU(alpha=0.1)(x)
```

ReLU: $f(x) = \begin{cases} x, \text{ if } x > 0 \\ 0, \text{ if } x \le 0 \end{cases}$



LeakyReLU: $f(x) = \begin{cases} x, \text{ if } x > 0 \\ \lambda x, \text{ if } x \le 0 \end{cases}$



Modifications - Dropout

```
749        pool2 = Dropout(0.5)(pool2)
764        pool4 = Dropout(0.5)(pool4)
```

Max-pooling dropout:

"Experimental evidence confirms the benefits of using max-pooling dropout, and validates the superiority of probabilistic weighted pooling over max-pooling and scaled max-pooling."

Wu H, Gu X. Max-pooling dropout for regularization of convolutional neural networks[C]//International Conference on Neural Information Processing. Springer, Cham, 2015: 46-54.

Modifications – Loss function

```
792        model.compile(optimizer=Adam(lr=1e-4), loss=Combo_Loss, metrics=[Dice_coef])
```

Combo loss：

0.4 * Weighted Binary Cross-Entropy + 0.2 * Focal Loss + 0.4 * Tversky Loss

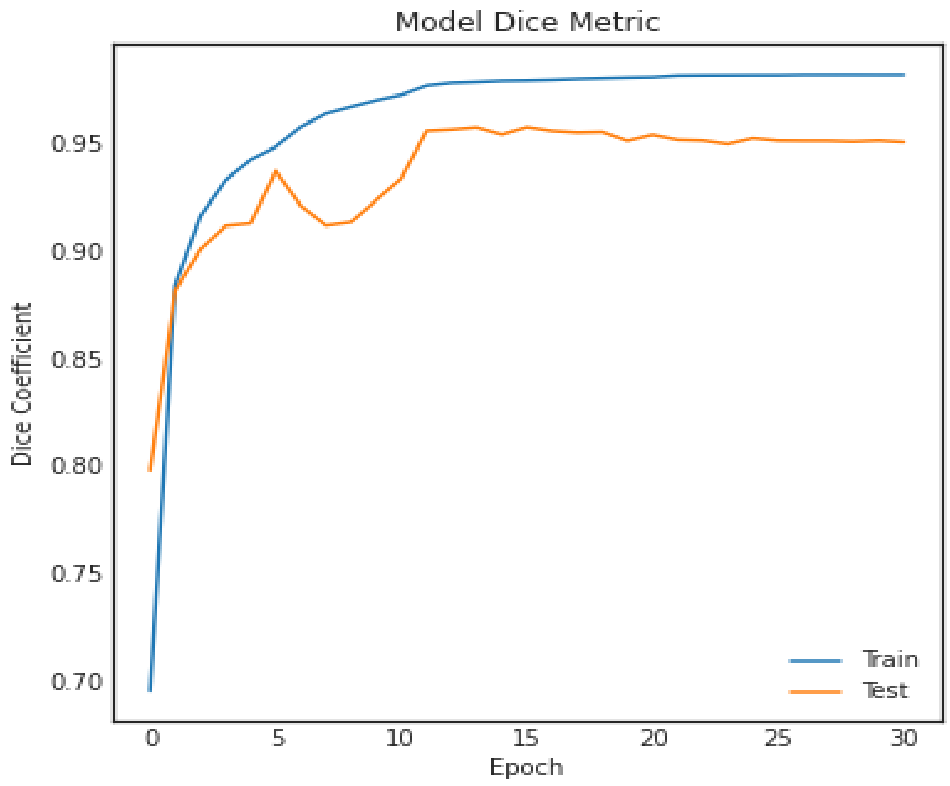
Dice similarity coefficient as metrics and Adam optimizer

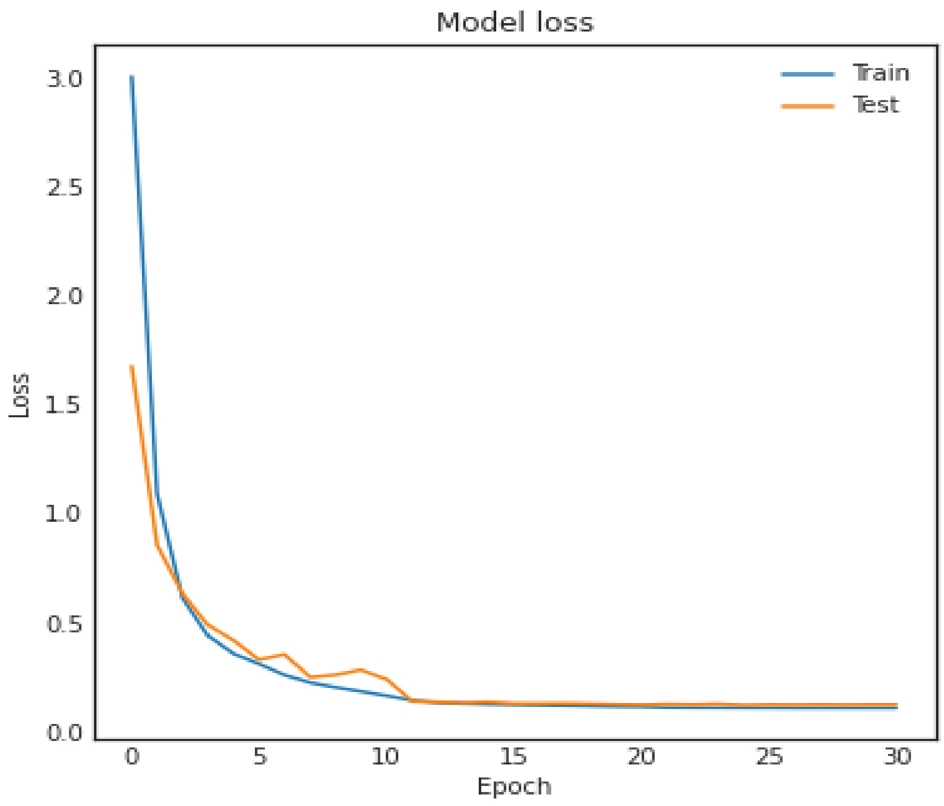
Wu H, Gu X. Max-pooling dropout for regularization of convolutional neural networks[C]//International Conference on Neural Information Processing. Springer, Cham, 2015: 46-54.

Demo

Results & Analysis

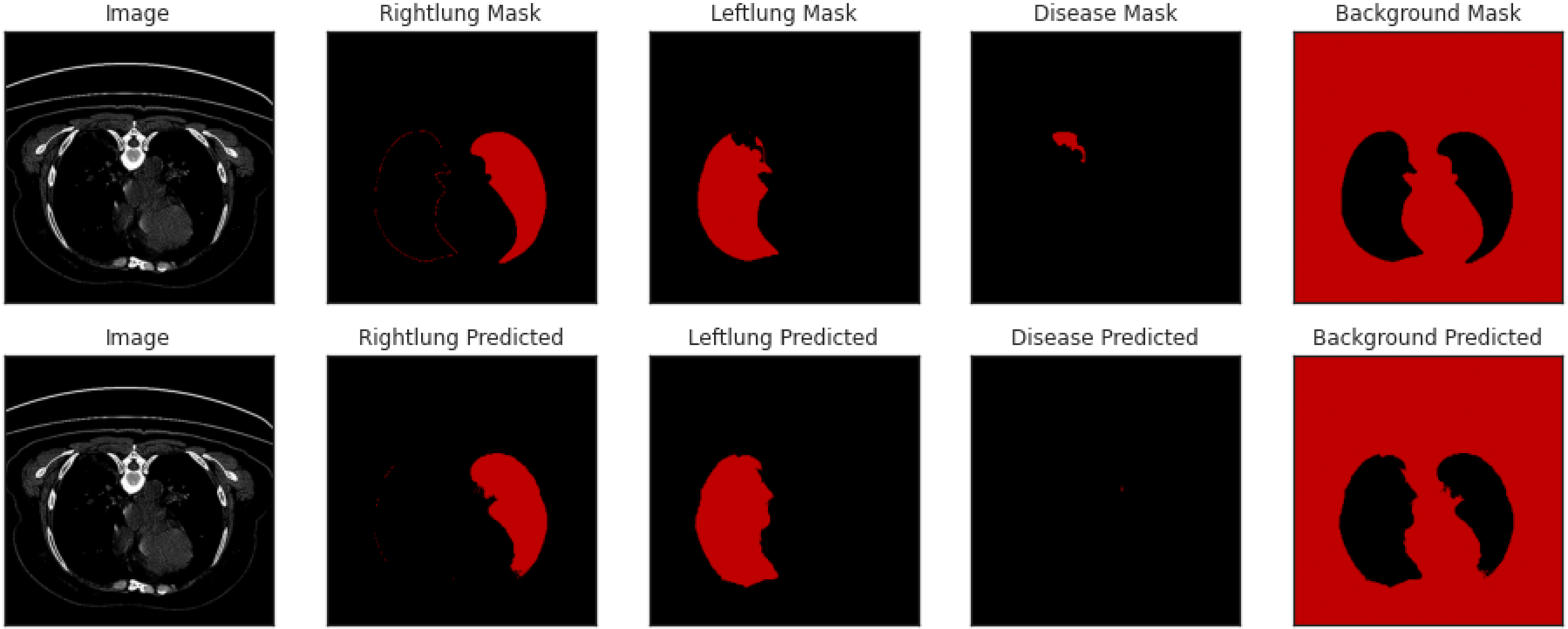
- Training and validation Dice similarity coefficient and Combo loss after training the modified model





Results & Analysis

- Randomly taken image segmentation into different classes and corresponding metrics using trained network



Classes	Dice Similarity Coefficient	Average Precision Score	Hausdorff Distance
Right Lung	91.71	84.66	3.85
Left Lung	93.08	87.04	4.43
Disease	0.3	0.53	4.4
Background	98.96	99.23	4.35

Overall Dice Similarity Coefficient: 97.6

Results & Analysis

- ROC and AUC curve on the set of test image after prediction (170 images)

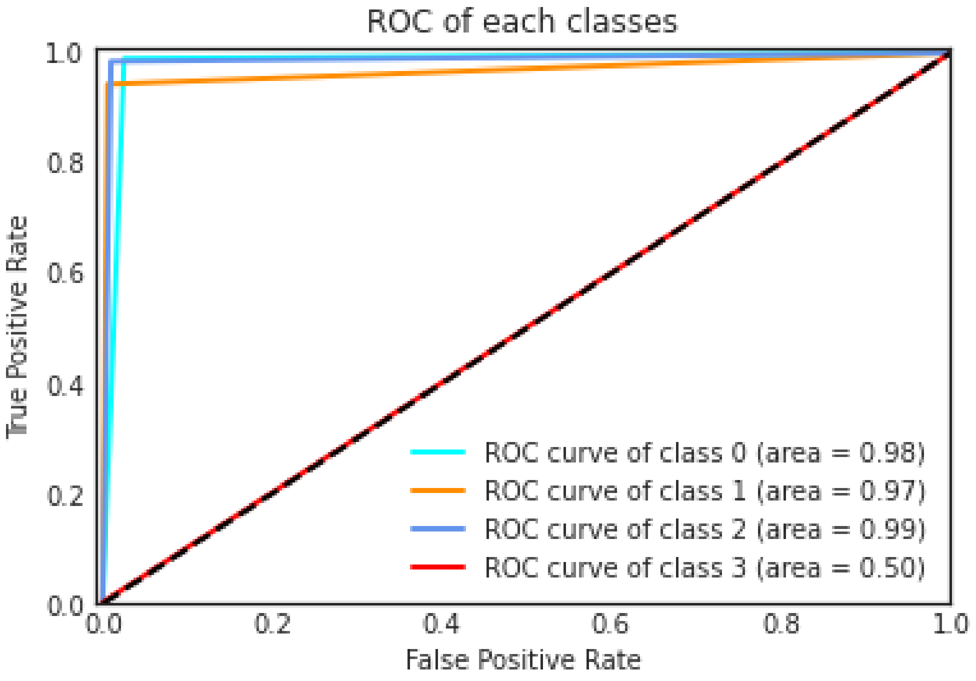


Fig.: ROC and AUC after selecting max probabilities

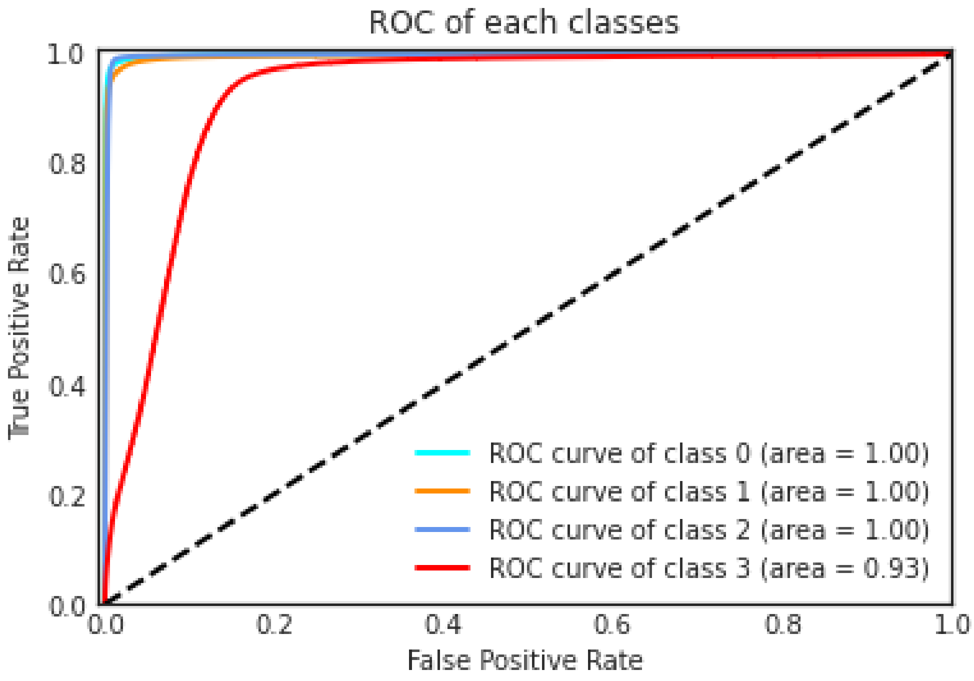


Fig.: ROC and AUC after model prediction

0: Background, 1: Right lung, 2: Left lung, 3: Disease

Results & Analysis

- Pearson correlation coefficients & Dice similarity coefficients of set of test data

Classes	Dice Similarity Coefficient	Pearson correlation coefficients
Right Lung	90.85	90.50
Left Lung	88.91	88.86
Disease	0.51	1.66
Background	99.37	93.05

Drawbacks & Future Works

- This dataset contain very less images having disease masks and deep learning networks need large amount of data to learn results bad prediction of disease mask.
- Background overpowers in our case even using dice similarity coefficient.
- Due to resource constraint could not do augmentation in the training dataset.
- Need more data related to disease mask.
- Need to do augmentations to add more versatility.

References

[1] Course Lectures and Lab Materials

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." arXiv preprint arXiv:1505.04597 (2015).

[3] Ibtehaz, Nabil, and M. Sohel Rahman. "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation." Neural Networks 121 (2020): 74-87.

[4] Google Colaboratory

THANK YOU