The purpose of this laboratory is to develop a fully convolutional network for the segmentation tasks by following one of the most efficient segmentation networks named U-Net. The architecture of this network looks like a 'U' shape. This model consists of three sections: contraction (encoder) path, bottleneck, and expansion (decoder) path. No dense layer is used in the model architecture; therefore, it is capable of maintaining the spatial information that makes it suitable for the segmentation models.

Task0) Develop a modular segmentation pipeline in a way that your pipeline contains separate functions for different steps such as image loading, segmentation mask loading, augmentation generator, similarity metrics, model architecture, and model training. Therefore, you should have a main file containing only a few lines of codes for calling the functions and setting the parameters to run the experiments.

Please remember in segmentation tasks, in general, each image has a corresponding segmentation mask. Therefore, it is quite essential to load the image and corresponding mask with the same order.

Develop a 2D UNet model with the following architecture:

First: develop a function and name it as "conv_block(…)". This function should entail two convolutional layers. Two optional batch normalization layers should also be added between the convolution layers and the activation function. This function, then, can be called several times in the model architecture.

Second: develop a function for the network architecture and name it as "get_unet(…)". The encoder part (contraction path) of the model should include four blocks, each of which contains a conv_block, max-pooling, and an optional dropout layer, respectively. As you go deeper into the encoder path, the number of feature maps within each conv_block should be increased by a factor of 2. Therefore, the fifth block of the model, the bottleneck block, will be one conv_block with 16 times more feature maps than the first conv_block. The decoder part (expansion path) of this model will be symmetric with respect to the encoder part that means this path should necessarily include four blocks as well. Each of these blocks entails a transpose convolution layer, followed by a concatenation layer (to concatenate the output of the trans. conv. layer and the related layer from the encoder path), an optional dropout layer, and a conv_block. The number of feature maps within each block in the decoder path is decreased by a factor of 0.5. Finally, there is the last block that contains only one convolution layer with the proper number of feature maps and certain activation functions, which should be consistent with the task.

Task1a) Lung segmentation in chest X-ray images:

Instantiate your pipeline with the following parameters: # of filters at first layer (base)=8; image size=256; batch size=8; learning rate = 0.0001; dropout=Fase; batch normalization=false; Metric=Dice Coefficient; no data augmentation; and loss function=binary cross-entropy. Reading

the 'X_ray' images (images and masks), randomly shuffle them and assign 80 percent of them for training and the rest of 20 percent for validation. Train the model for 150 epochs and report your final results. Please remember that you have to normalize the intensity range of the loaded images. Moreover, the segmentation masks are saved as 8-bit jpg images so that the background is labeled as 0, and the foreground is labeled as 255. Segmenting the lung regions within the X-ray images can be considered as a binary segmentation task, so that you need to normalize the loaded segmentation masks as well.

```
Path = '…./Lab3/X_ray/'
```

Task1b) keep all the parameters from the previous exercise the same, only replace the BCE loss function with "Dice loss" and repeat the exercise.

Is there any difference between model performance over validation set when you changed the loss functions? Do you expect to observe similar results when you deal with more challenging segmentation tasks? Dealing with cases in which the size of the target to be segmented would be much smaller than the image (imbalance between the background and foreground). Which loss function would you choose? Discuss it.

Task1c) Repeat the tasks 1a and 1b by including the dropout layer (rate=0.2). Does it affect model performance? What about the learning curves?

Task1d) Increase the model capacity by setting base=32. Repeat tasks 1c and evaluate the results.

Taks1e) Repeat the task 1c with setting the base=16, batch norm=True, and train the model by applying augmentation technique on both images and masks: rotation range=10; width and height shift ranges=0.1; zoom range = 0.2, and horizontal flip. Could image augmentation improve the generalization power of the model?

Task2a) Lung segmentation in CT images: Reuse the same pipeline of task1 for segmentation of lung regions within the "CT" dataset. Please note the left and right lungs are labeled with different values in the segmentation masks. So you need to binarize the loaded masks for a binary segmentation task. Use the following parameters: Base=8, batch norm=ture, image_size=256, train_percentage=0.8, LR=1e-4, loss=dice, dropout=True(0.2),batch size=8 and augmentation: false. Please note as the number of images in this data set is quite large (>8000 images), the computational time will be a bit longer so that you can train the model for only 50 epochs.

```
Path = '…./Lab3/CT/'
```

Task2b) Evaluating the segmentation performance is often done by using Dice coefficient; however, it is quite essential to assess the rate of false positives and false negatives too. Implement "precision" and "recall" metrics as well, and use these three metrics in your model (Metric = [dice_coef, precision, recall]) to monitor the model performance. Repeat the task 2a by including data augmentation technique (similar to task1e) and train the model for 50 epochs and interpret the observed values of the three metrics over the validation set. In general, what can be inferred from precision and recall metrics?

Task3) As it is already explained, the labels of the left and right lungs in the CT masks are not the same. We can consider the two lungs as two different organs and perform a multi-organ segmentation. Modify your model and adapt it for a multi-organ segmentation task and segment the left and right lungs separately in one framework. Set the image_size:256, batch norm:true, LR:1e-4, dropout=True(0.2), augmentation:true and n_epoch=80. Choose a proper loss function for multi-organ segmentation and modify the segmentation mask labels to match the problem.

Bonus Task) Brain tumor segmentation in MR images: In contrast to the lung region that entails specific shape, appearance, and intensity pattern, brain tumors can be presented in a large variety of shape, appearance, texture, and intensity patterns. More importantly, they can appear in any region within the brain; therefore, they do not have a fixed location. In this exercise, you are asked to optimize your model by tuning the hyperparameters for the task of brain tumor segmentation. Please note the original size of the brain MR images are 240 by 240. Report the employed hyperparameters and model performance. Monitor the model performance by using three metrics: [Dice, Precision, Recall].

This dataset is a part of a larger dataset related to a challenge named as Brain Tumor Segmentation challenge. The highest achieved dice coefficient over this dataset is around 0.96. Please compare your best-achieved result against this best performance and discuss the possible ways to improve the performance of your developed model.

```
Path = '…./Lab3/MRI/'
```

Good luck.