**COSE474 Deep Learning Project #3:**

**Encoder-Decoder Implementation**

**1. Introduction**

In this assignment, we are going to use what we learned in the Encoder-Decoder lecture, and find out how the actual model works and how to adjust parameters in detail.

It is intended to flexibly change the model by changing the Encoder part of Unet to Resnet50. Finally, for the segmentation work of Pascal Voc 2012 Dataset, it starts with library import and proceeds to output and evaluation. I want to learn a lot while working on this practical project.

**2. Model Settings**

To proceed this project, first of all, we set up the execution environment. Because it handles a lot of data and requires a lot of computation in one epoch, it was performed in a GPU environment.

There are two main types of models. It is a Unet model used by many people because it shows good performance in biomedical image segmentation in MICCAI. Then, next model uses the Resnet50 Encoder Unet which used the structure in Project 2. This makes it easier to understand the encoder structure.

Unlike the CNN model, the Upsampling process and Skip Architecture (Concat before layer) had to be performed. It was very helpful in understanding the model structure such as input and output channel tuning during the process. Also, while setting up the GPU, I was able to learn a lot about how to use the GPU. However, since it takes a lot of time to learn, the number of epochs was smaller than the desired number.

Based on PyTorch, Image Processing (Resize Normalize) was performed. And the designated hyper parameters are as follows.

• Loss\_Func = Cross Entropy Loss

• Optimizer, Learning Rate = Adam, 0.001

• Batch\_size, Epoch = 4, 5

**3. Dataset**

The data is Pascal VOC Dataset introduced at Visual Object Classes Challenge 2012 which

has 20 classes. The train/validation data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations.

**4. Experiments**

First, we had to adjust the input and output channel values ​​of convdown and convup in Orignial Unet. In the decoder process, it is concated afterward, so it has a different channel value from the previously thought layer. Also, in the skip architecture, concat was performed to match the unet architecture.Through the def conv(nxn) codes, It can be seen that the function conv(nxn) is sequential of conv2d, batchnorm, and relu.

Next, we proceeded with the Resnet\_encoder\_ unet structure. Since the feature was the architecture overlapping with project2, the sequential layer was structured based on the structure and the table in the pdf. In particular, I had to adjust the stride for the input output size, because an error occurred if the dimension was not adjusted in the skip architecture concat part later. Finally, after performing the initialize model, hyperparameter, and load pretrained model in ‘main\_skeleton’ and in ‘modu les\_skeleton’, training and validation processes were conducted. After proceeding from initial to verification until epoch 5, the result without using the pretrained model is as follows.

• Original\_Unet Train : 0.6981687965632984

• Original\_Unet Test : 0.6921466670624198

•Resnet\_encode\_Unet Train: 0.6945011363566569

•Resnet\_encode\_Unet Valid: 0.6881795909306775

**5. Conclusion & Discussion**

When looking at these results, it can be confirmed that the structure is well constructed. But there were several discussions during the experiment.

1. In the last ‘module\_skeleton’ part, in the process of working with temp\_rgb and temp\_label, it was not completed because it was not possible to accurately distinguish between them. I divided it by label and checked the result, but it didn't go well. I want to think about it in detail while looking at the solution.

It was the most worthwhile project because I was able to think deeply and come up with a model structure while working on the project.