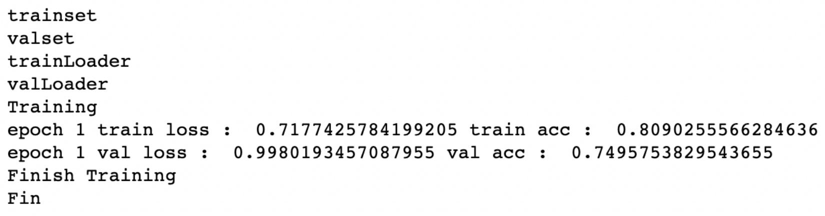
Project 3 - Encoder-Decoder Architecture Implementation

2021320117 Bae Minseong

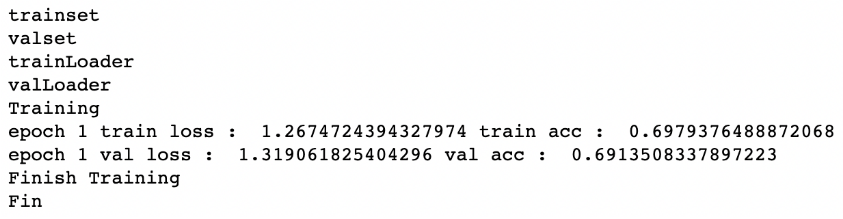
In this project, we need to implement basic U-Net architecture and a variant of U-Net architecture replaced the encoder with ResNet-50, and code for train/test process with Pascal VOC 2012 dataset.

In ‘modules_skeleton.py’, we could easily implement the code for train and validation referring to previous projects. The key point in the implementation is for input images and predicted results, we need to convert the segmentation mask into RGB value, by using the dictionary ‘cls_invert’. In ‘main_skeleton.py’, we could also easily implement the code for model initializing, loss function and optimizer, loading pre-trained model parameters, and saving trained model parameters referring to the previous project. In ‘UNet_skeleton.py’, we need to implement basic U-Net architecture. We have to fill in the blanks of channel sizes in the convolutional layers for downsampling and upsampling referring to the U-Net architecture learned in the class. At each downsampling step, we double the number of feature channels. And it is important to designate the size of input channels in upsampling layer with the added value of the previous size of channels and the size of channels of the concatenating cropped feature map. In the implementation of forward propagation, we can use torch.cat function for implementing the concatenation of upsampled feature map and correspondingly cropped feature map. Finally, in ‘resnet_encoder_unet_skeleton.py’, we have to implement the ResNet-50 architecture for the encoder in U-Net, and it overlaps with the code of previous project. By just slightly modifying the code for previous project, we can easily implement it with same channel sizes for its layers. And we can use torch.cat function for implementing forward function in ResNet-based encoder.

By using pre-trained parameters for U-Net and ResNet-encoder-U-Net, by training an epoch, we can get these results. (I used Jupyter Notebook for the execution of main_skeleton.py. ‘main_skeleton.ipynb’ is just exactly same code with main_skeleton.py.)



```
trainset
valset
trainLoader
valLoader
Training
epoch 1 train loss :  0.7177425784199205 train acc :  0.8090255566284636
epoch 1 val loss :  0.9980193457087955 val acc :  0.7495753829543655
Finish Training
Fin
```



```
trainset
valset
trainLoader
valLoader
Training
epoch 1 train loss :  1.2674724394327974 train acc :  0.6979376488872068
epoch 1 val loss :  1.319061825404296 val acc :  0.6913508337897223
Finish Training
Fin
```

In the first figure, we can check out the result for U-Net architecture. In the second figure, we can check out the result of U-Net architecture with ResNet-50 encoder.