Report for the COSE474 project3 (Encoder-Decoder Implementation)

2016320187 컴퓨터학과 홍성윤

1. Code description

(1) main_skeleton.py, modules_skeleton.py

- main code loads, saves and initialize model and do training by using functions in modules.
- module code: train_model func trains model by loss gradient. get_loss_train and val_model compute loss and accuracy of training data and validation data respectively. To visualize the result, we convert segmentation mask into r,g,b image using cls_invert[].

(2) Unet class

- According to the network architecture, we can easily determine the in_channels, out_channels of each conv layer(convDown, convUp). The number of in_channels of convUp layer should consider the channels from skip connection.
- In forward func, applying each layer is already written. We should only consider the skip connection. Concatenate corresponding tensor appropriately in dimension 1.
- If downsample is False, we should maintain activation map size. So, all strides of 3 conv layer should be 1 and padding of

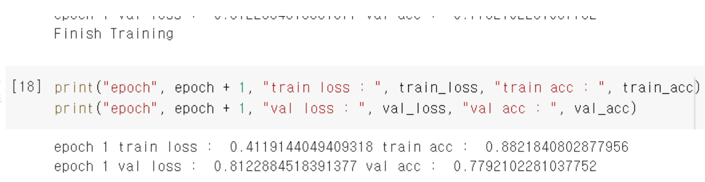
(3) ResidualBlock class

- Code overlaps with previous assignments. There are two types of ResidualBlock. One reduces activation map and the other doesn't. We can represent this by downsample parameter.

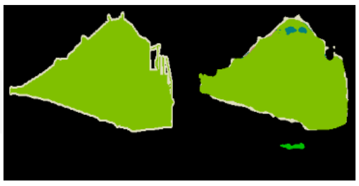
(4) UnetWithResnet50Encoder class

- Layer initialization is similar to previous project. But downsample of the last ResBlock of layer3 is not True anymore because layer3 should remain the spatial resolution in this architecture.
- forward func is similar to that of Unet class. It consists of contracting path and expanding path. Bridge which consists of Conv2d, Relu, and BatchNorm, is used in the process of converting from contracting path to expanding path
- Similar to Unet class, we should consider skip connection in expanding path. We should concatenate out1, out2, out3 tensors with corresponding tensor respectively.

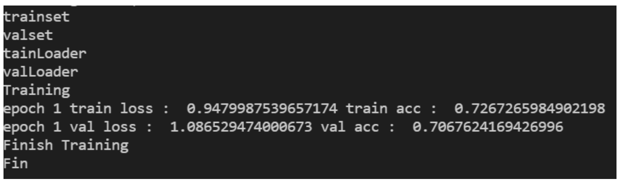
2. Result and Discussions



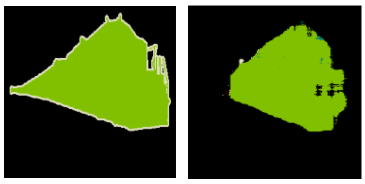
```
Finish Training

[18] print("epoch", epoch + 1, "train loss : ", train_loss, "train acc : ", train_acc)
     print("epoch", epoch + 1, "val loss : ", val_loss, "val acc : ", val_acc)

epoch 1 train loss :  0.4119144049409318 train acc :  0.8821840802877956
epoch 1 val loss :  0.8122884518391377 val acc :  0.7792102281037752
```



- 1epoch Training result of Unet(left). Example RGB images of ground truth label(middle) and result(right)
- I used colab environment to use cuda. (training Unet took so long using CPU)



```
trainset
valset
tainLoader
valLoader
Training
epoch 1 train loss :  0.9479987539657174 train acc :  0.7267265984902198
epoch 1 val loss :  1.086529474000673 val acc :  0.7067624169426996
Finish Training
Fin
```



- 1epoch Training result of UnetWithResnet(left). Example RGB images of ground truth label(middle) and result(right)
- The accuracy came out as expected. Predicted label was quite accurate as you can see above.
- Unet had higher val accuracy. In fact, we can see that the results of Unet were better for the same label than UnetWithResnet.
- By implementing two types of Unet architecture, I realized that a variety of encoder-decoder architectures with better performance could be created.