COSE474 Deep Learning: Project #3: Encoder-Decoder Implementation

Project report REUTELSTERZ ELIAS

Description of the code:

First of all I filled in the gaps in main_skeleton.py and modules_skeleton.py. In the main file I had to add the loss function and the optimizer. Then I had to load the trained model with the extra-parameter 'map_location=device' and save it later at the already generated 'savepath2'. In the file modules_skeleton.py I basically added the same model configuration as in project 2. This time we should think about the differences in get_loss_train and val_model to the first loop in train_model. In those two loops we were in the with torch.no_grad(): environment because we don't need to backpropagate. Additionally we are first in .eval mode and then in val_model the model is not in .eval mode and not in .train mode. In the last gap I had to implement the calculation of the right colours for the pixel recognition for the output files.

While implementing UNet_skeleton.py the most important thing was to consider which channels the Up-Path has. Because of the concatenation in every step upwards the dimension got higher with the factor 1.5.

In the implementation of resnet_encoder_unet_skeleton.py one could use most of the parts of project 2. The only difference was the concatenation in the Up-Path similar to UNet.

To change between the two models one has to change 3 lines of code: the import, the creation of the model and the scripting of the trained model.

Results:

I could only take 20 pictures for 1 epoch because otherwise my laptop was too weak. Those were my results:

U-Net :

epoch 1 train loss : 0.5974014103412628 train acc : 0.918581485748291

epoch 1 val loss : 1.3074302673339844 val acc : 0.7852725982666016

ResNet:

epoch 1 train loss : 0.6174842417240143 train acc : 0.8225803375244141

epoch 1 val loss : 1.1609834134578705 val acc : 0.7150864601135254

I also saved the files in history and the model parameters in output in my files.

Discussion:

One could see that a validation accuracy of around 70 percent is really impressive for the difficulty level of this task. A point of discussion that concerns the accuracy may be that most of the accuracy follows from just not detecting a class, which is for most pixels in the background the right answer. That's why it could be interesting to just look at the class-pixel which are in the foreground.

In addition to that it's exciting to look at the difference of U-Net and ResNet. We can see that U-Net has a far better accuracy and that could be related to the depth of the model and the higher number of concatenation steps in between the Down- and the Up-Path.