David Dashti
Filip Söderquist

2020-09-30

Lab 5 - Report

(Note: When we are talking about loss & accuracy, we are referring to the validation values)

Task 1

Answer) No the performance is not consistent across all fold. We could observe a performance fluctuation between each fold, and this has to be accounted for when evaluating the results. In order to deal with this one could perform averaging of the results from each fold which is standard practice when doing k-fold cross validation.

Unfortunately, when increasing the number of folds to above 3, the GPU ran out of memory. This occurred regardless if deleting the model and clearing the session in-between each fold. Thus, we were not able to see if we could get better results with 10 folds which is standard.

We used data augmentation and loaded the data batch by batch with 50 epochs per fold. The number of folds was set to 3.

With these settings we got the following results:

First fold: Loss & accuracy of 0.2126 and 0.7875, precision & recall of 0.8143 and 0.7800
Second fold: Loss & accuracy of 0.1864 and 0.8137, precision & recall of 0.8433 and 0.8021
Third fold: Loss & accuracy of 0.1641 and 0.8360, precision & recall of 0.8598 and 0.8283

Mean: Loss & accuracy of 0.1877 and 0.8124, precision & recall of 0.8391 and 0.8035

First fold had stable learning, which converges after about 30 epochs. Second fold overfitted slightly. Third fold had the same behavior as the first fold.

Task 2

Answer) Yes, we could observe a discrepancy between the loss functions and the evaluation metrics. What we could observe was worse performance when adding weight maps than without weight maps. We had an 80/20 split, used data augmentation, and trained for 150 epochs. This was unexpected as we predicted that the weight map would improve the segmentation results by helping to add focus on the edges.

We got the following results: Loss & accuracy of 0.2769 and 0.5035, precision & recall of 0.8753 and 0.3556.

Task 3

Answer) Due to memory issues we could not apply data augmentation to this task and similar to task1 we could only apply 3 folds. The aim was to evaluate if auto-context could improve the dice value when adding posterior probabilities to the training. The training was done on 1800 images and each fold was around 100 epochs.

Step 1

Mean: Loss & accuracy of 0.4074 and 0.6045, precision & recall of 0.8297 and 0.4957

Step 2

Mean: Loss & accuracy of 0.4270 and 0.5730, precision & recall of 0.8045 and 0.4763

David Dashti
Filip Söderquist

2020-09-30

Final observations

Answer) According to our results, ordinary data augmentation with a simple U-net, yields the best accuracy. We expected both the weighted dice loss and auto-context to perform better but this was not the case. Because of major memory issues, we could not perform data augmentation on auto-context with all training images. Which probably decreased the performance of the auto-context network since it overfitted to the training data. We would have liked to test the auto-context network with data augmentation.

It was not possible to observe an increase in performance between auto-context steps which would be expected of the algorithm. If we would have been able to have more folds, the auto-context network could have performed better than the other networks. This is something we would have liked to test.

Despite the above arguments, we are still able to compare auto-context and weighted dice-loss due to the fact that we could not see improvement in-between steps of the auto-context. This leads us to believe that even if we had added augmentation, we still would not be able to gain any improvement. In this regard, auto-context performed better than weighted dice-loss. Thus, simple u-net with augmentation was the best, followed by auto-context without augmentation and the worst was weighted dice-loss with augmentation.