Report Practical Exercise 02

Task 1:

1.a)

Code explanation:

For all three methods we started by iterating over the four indices (BG, GM, WM, CSF). For each of those we then calculated the intersection = True positives by summing up all pixels where the prediction and the ground truth both evaluated to the current index. The False Negatives and False Positives were calculated analogously. We then applied the formulas from the slides to calculate the dice score, precision and recall respectively.

The Dice coefficient measures how big the overlap between the predicted segmentation and the ground truth is. The formula is “2 \* intersection of A and B / A + B”. This can turn into “2\*TP / TP+FP + FN+TP” for Boolean data like in our case, which equivalent to the F1 score.

The recall compares the amount of correctly labelled area to the area that should be labelled as positive (TP /TP+FN).

The precision compares the amount of correctly positively segmented area to the total positively segmented area (TP/TP+FP).

The F1-score and therefore also the Dice score is the harmonic mean of precision and recall.

TODO: Pros/ Cons

1.b) Code explanation: get best and worst samples.  
We iterated over all predictions to compute the patients mean and standard deviation of the dice score. We then used the np.argmax and np.argmin methods to find the patient (index) with the highest/ lowest mean dice score. We return both the patients with the best and worst score as well as the corresponding scores for these patients.

1.c) Code explanation:

Get Results dictionary:

We iterate over the first TODO samples and evaluate the Dice score, the precision, and the recall for each and add them to the results dictionary that is returned.

Plot results summary:

We use matplotlib subplots to generate a plot over the given results.

1.d) Bonus optional

TODO

Task 2:

2.a)

We chose KMeans because it is a very simple and fast clustering algorithm. It may perform poorly when clusters are of drastically different sizes or densities, which is the case for the background label and somewhat the white matter label, nonetheless it serves as a good baseline to compare other methods to and should yield acceptable results.

2.b)

We chose Gaussian Matrix Modelling as a more sophisticated version to tackle the issues regarding the different sizes of the clusters mentioned in 2.a). The algorithm is a bit less vulnerable to the aforementioned scenarios, but the basic implementation of sklearn is not laid out for minibatches and the large dataset could not fit into the RAM on google colab. We therefore split the data into minibatches manually and fit the model multiple times, always taking the weights of the last as initialization for the next.

Another issue with the GMM algorithm, is that it is highly dependent on the initialization values. It necessitates KMeans to find good base values otherwise it could easily get stuck in suboptimal local minima.

Task 3:

We decided to use the enhanced 3D UNet implementation of MONAI. MONAI is based on PyTorch and provides a framework for deep learning in healthcare imaging. MONAI’s UNet implementation is an enhanced UNet which makes use of residual units (in our case two per convolutional layer). These units provide a kind of shortcut connection to allow a direct flow of information from the input to the output. By using them, the model’s ability to capture complex patterns in our brain MRI 3D images is improved.

Our net consists of 5 convolutional layers each for the encoder and decoder. The number of filters at each conv. layer are: 16, 32, 64, 128 and 256. We use a 2x2 stride at each level. Furthermore, batch normalization is applied.

We had to preprocess our tensors using padding, resulting in 1x128x128x128 Tensors which we fed into the net. After the training we removed our padding accordingly.

Due to memory limitations, we used a batch\_size of 2. Another hyperparameter we used was a learning\_rate of 0.001. We trained our model for 10 epochs.

We achieved very good results with a best case dice of 0.95 and a worst case dice of 0.92 on the test set.

See: <https://medium.com/codex/architectures-for-medical-image-segmentation-part-3-residual-unet-ac5a4ca4212d>

<https://docs.monai.io/en/stable/networks.html#unet>

Task 4:

TODO: ergibt das mit dem intensity plot so Sinn?

4.a)

The most intuitive, non-learning-based approach would be to take thresholds and segment based on those and the Intensity Density Plot. Thresholds could simply be put around the peaks (local maxima) of the graph with the borders between thresholds being in the local minima. Rough estimates of the thresholds for the graph in 2 could then be [-100; 90], (90, 225], (225, 400], (400, 550] for background, CSF, white matter and grey matter respectively. These local minima would be calculated by the code automatically.

4.b) Evaluation of 2.a (KMeans)

Ein Bild, das Screenshot, Text, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

Qualitative: (looking at images)

Especially for the worst prediction on the test set, even an untrained eye can see, that parts of the brain that should be labelled remain unlabelled as background. Even in the best performing prediction on the test set there are black zones, that should be labelled differently especially in the axial and sagittal view. Nonetheless the overall image, especially for the best prediction largely resembles the ground truth.

Quantitative: (scores & numbers)

While the maximum dice score of 0.74 would be good if it were an average of the whole validation set, the lowest achieved dice score of 0.46 is too bad to be useful in a real-world scenario.

It might still be useful as a basis for further human segmentation.