Brain Tissue segmentation project (IBSR18 dataset)

Raneim Mohamed, Saud Khan

January 16, 2022

1 Introduction

Brain tissue segmentation is an important step in many clinical applications. It has a crucial rule in morphometric analysis of brain structures and could be used for preoperative evaluation, surgical planning, and longitudinal monitoring for disease progression. The objective of this challenge is to successfully segment the Cerebrospinal Fluid (CSF), White Matter(WM), and Gray Matter (GM) region in the brain MRI in the T1 modality. This Task could be very challenging depending on the nature of the dataset provided. The dataset given to us is IBSR18 dataset for MR brain image segmentation. It consists of 10 volumes for training, 5 for validation and three for testing. This dataset is challenging because there is heterogeneity in intensities, different spatial resolution, and the ground truth is not accurately segmenting the brain tissues. To mitigate these problems, three different models(2D Unet, 3DUnet, and transfer learning) have been tested and the best model was chosen to give final predictions on the test set.

2 Challenges in the Dataset

There is some inconsistency in the data due to the source of all images not being the same scanner. This means that the data can have different levels of noise and intensity differences that we have to take into account and also to make sure we have a segmentation model which is not biased towards data of any specific modality. Another issue we noticed after reading the dataset through ITK snap and overlaying the provided segmentation masks was that there were many unlabeled regions which had intensities different from background but were assigned the background label. These were mostly the outer cerebral regions and regions of the brain connected to the spinal brain. This type of mislabeling issue can cause a model to associate data of this nature to the wrong labels like CSF, WM and GM which are our labels of interest. This could also lead to associating intensities with information to the background label which is what we would want to avoid. The third issue is that the data is not enough for a strong deep learning model as large datasets are essential to successful deep learning approaches. Also, processing this type of data is computationally expensive and we need to have an approach which can make the whole process easy for the computer and the GPU memory limitations. In the next paragraphs, we will discuss some of the techniques used to remedy these challenges.

2.1 Label Generation

To remedy the issue to unlabeled regions mentioned in data challenges, we decided to assign them a unique label which will prevent the model from mixing these regions with our actual labels of interest. To this end we used the fact that the intensities that were labeled as background had a much higher value compared with the null value of the background. This helped us select just those regions in the masked image that have higher intensities in the actual image and we assigned them a different label as shown in figure(1).

3 2D Unet

preprocessing
After analyzing the data visually and through histogram distributions, it was noticed that IBSR_07 has a good contrast compared to the rest of the training set as it has a wide histogram, so it was used as a reference volume for histogram matching. First, z-score normalization was applied to all datasets. Then, adaptive histogram equalization was applied on reference volume to further improve its contrast. After that, the histogram of all volumes was matched to the histogram of the reference image. After preprocessing the data, useful patches are extracted(meaning discarding the large number of background batches) to be used for training the model. The training dataset was splitted into 7 volumes training and 3 validation and the validation set was used for testing.
Architecture
Our implementation mainly benefits from basic framework provided by DR. Jose Bernal in the tutorial sessions. The 2D U-Net is formed by 2 main parts. An encoder, whose main function is to extract features from the data while reducing the spatial size of the images using double convolutional blocks; and a decoder, that progressively recovers the original matrix size of the input image by a series of concatenations and double convolutional blocks. The Unet started with 32 intial filters and every double convolutional block is formed by a convolutional layer using a 3x3 kernel, followed by a ReLu activation function. In the encoder, every double convolutional block is followed by a dropout layer to avoid overfitting and a Max Pooling layer to reduce the matrix size by half. On the other hand, number of filters in the next double convolutional block is doubled to compensate for the loss in spatial information and to encode this spatial information into high-level feature information. The last double convolutional block of the model is called the bottleneck layer, after which the decoder is defined. To recover the original matrix size, a transposed deconvolution is applied to the bottleneck to increase the matrix size by 2. At the same time, the number of filters in the upsampled layer, is reduced by half. After the transposed convolution layer, a concatenation layer is applied with the corresponding filters in the encoder. After that, a double convolutional block is applied before the next transposed convolution until the original matrix size of the input is recovered. At the last layer, a softmax activation function is applied.
Experimentation
There are different parameters to be optimized in this model and the ones that were optimized in this project at first were the number of initial filters, the number of double convolutional blocks, and the dropout rate. The optimal parameters were chosen based on the DCS and the best model architecture was explained in the previous section. After choosing the model, other experiments were performed to choose the optimal patch size, stride, and patch threshold. Patch threshold determines the least percentage of background to be included in each patch. The best combination of these parameters are shown in table 1.

4 Transfer Learning

Architecture and Preprocessing
The aim of this trial is to use a pretrained model on a similar task and use it for inference or finetune it with our data. The chosen model is Neuronet which is a deep convolutinal network trained on 5,000 T1-weighted brain MRI scans from the UK Biobank Imaging Study to segment brian into brain tissue and cortical and sub-cortical structures using the standard neuroimaging pipelines. The architecture consists of a Resnet encoder as a feature extractor and multiple FCN decoders for the multiple outputs. In our project, we are only interested in one FCN for brian tissue segmentation, so the tissue model configuration file was chosen to produce only one output from one FCN decoder. The code for this model is available on Github for DLTK models. Preprocessing was the same as the one used in the 2D Unet.
Experimentation
For transfer learning, we performed two experiments. In the first one, all the layers were frozen and the model was tested on the validation dataset while in the second experiment, the model was retrained on the provided training dataset after splitting it into 7 volumes training and 3 volumes validation and tested on the 5 volumes validation set.

5 Final Model:3D Unet

5.1 Preprocessing

The first thing we notice in the data is that the brain MRI scans are generated from different scanners. It is therefore very important to normalize the intensities of the given images to bring them to an equivalent level. To implement the standard deviation and mean normalization, we first need to decide what mean and standard deviation to use to normalize by. For this we choose the mean and the standard deviation of what we consider the region of interest of the images.

$$I_{z-scored} = \frac{I - \overline{I_{ROI}}}{\sigma(I_{ROI})}$$

$$where\ \overline{I_{ROI}}\ \&\ \sigma(I_{ROI})\ are\ the\ mean\ \&\ std\ of\ the\ Region\ of\ Interest\ respectively$$

Before we proceed, we also want to convert this data into a form that can be processed by the GPU without any memory problems. To achieve this, we perform a 3D voxeling technique where the data is converted into blocks of overlapped 3D voxel region. This is also called the patching approach. Another benefit of this preprocessing is that it assists against the limited training data issue. Now this data is fit for being fed into the model.

5.2 Modifications to the Standard 3D UNet

In this section we will briefly explain the model used for segmentation. We started with the basic 3D UNet but we make several modifications in the architecture to suit it better for our task. The main modifications we make to our UNet model are the addition of the Batch Normalization and the Dropout layers. The benefits of these are multifold as it prevents the model from overfitting on the training data which helps in making the model more robust. This also makes the training process very fast. The idea of batch normalization is that just in the way we normalize the data before we feed it to the model, we also want to replicate this process for the inner layers of our model as well. Therefor the input of each layer is normalized in the same z-scoring fashion as we discussed in the preprocessing section. This step is done at every output of the activation function. Since we have both encoder and decoder layers in a UNet, we apply batch normalization after every ReLU layer. In case of the dropout, it is applied before each ReLU layer in the encoder part and just after the maxpooling layer during the decoder part. Since dropout involves randomly dropping out nodes in training, this produces a regularization effect and also helps against overfitting.

Loss Function: With regard to this, several loss functions were considered. Cross Entropy Loss Function is commonly used for multi-class problems but since the data we are dealing with has an imbalance class distribution not just in the background vs. brain tissue but also intra brain tissue in terms of CSF being a more challenging area. To remedy this, a more robust loss function was used which is the Multi-Dice Loss function. This loss function works on the overlap between each class and the rescaling of corresponding labels from 0 to 1. This Multi-Class Dice Loss function is given by the following equation:

$$\boldsymbol{MultiDice\ Loss} = \mathbf{1} - \frac{\sum_{N}^{i=1} \mathbf{2}|\boldsymbol{A_i} \cup \boldsymbol{B_i}|}{\frac{|\boldsymbol{A_i}| + |\boldsymbol{B_i}|}{\boldsymbol{N}}}$$

$$where\ i = 1,2,3 \ldots N$$

5.3 Experimentation

Our implementation mainly benefits from basic framework provided by Dr.Sergi Valverde, one of the instructors who had a deep learning workshop with some previous batches of MAIA. Our modifications to the UNet are also based on Rehman et al's "BU-Net: Brain Tumor Segmentation Using Modified U-Net Architecture". This architecture called the BU-Net was designed for the BraTS 2017 challenge which focuses more on tumors and lesions in the brain. Following their approach and as explained in the previous paragraphs, we added the recommended dropout and batch normalization layers and also adjusted the depth of our modified model. For the data used in training in this experiment, we mainly used three different distributions. First, did a 70-30 split on the data in the order the images are name. Here we ended up with IBSR_09, IBSR_16 and IBSR_18 as our training-validation data. The second time we did a different split taking into account the modality and the type of scanner for each provided image in the dataset. For this we chose for 70-30 split into training and training-validation data, IBSR_05, IBSR_07 and IBSR_18 as each comes from a different scanner. In the third case, there was no split on the training data. All training data was used with the validation data being used as the validation data.

6 Results

6.1 Results of the 2D Unet and Transfer Learnign

The first table shows the optimization of patch size, stride, and patch threshold of the 2D Unet. According to the results, it was noticed that changing the patch size to 64 gave the best results with stride 32 to have overlapping patches which means more data for training and agrees with our expectations. Also, changing the patch threshold to 0.5 dropped the average DSC of WM by 0.01 which means that it is better to have less background in the patches, so the final model has 0.2 threshold value. The second table, shows our experimentation on the transfer learning technique. According to the average DCS, retraining the model with our data greatly improved the results especially for CSF which makes sense since our data has different nature than the ones used originally to train the model. Also, this model outperformed the 2D Unet for WM and GM, but it was the same for CSF.

patch size	stride	Threshold	BG	CSF	GM	WM
32	32	0.2	1	0.82	0.91	0.87
32	16	0.2	1	0.88	0.92	0.91
64	32	0.2	1	0.88	0.93	0.92
64	32	0.5	1	0.88	0.93	0.91

Table 1: Average Dice Scores for the given validation set using 2D Unet

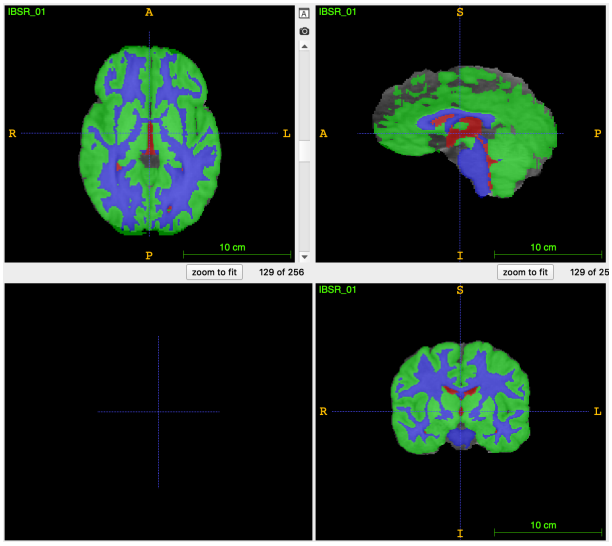
Tissue type	Freezing all	Retraining
BG	1	1
CSF	54	88
WM	83	94
GM	83	93

Table 2: Average Dice Scores for the given validation set using transfer learning

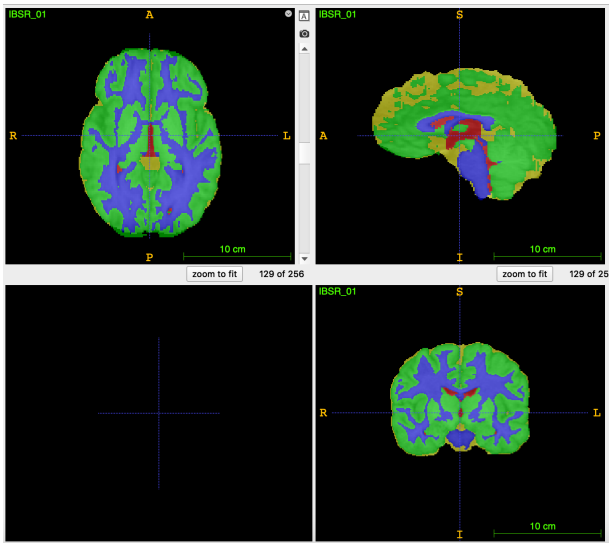
6.2 Final Model Results

At first, some experiments were done to choose the parameters that produce the best results. The optimal parameters for this model were 0.003 learning rate, 128 batch size,and 16 block step size. In each experiment of the three main experiments regarding data splitting, the model took one hour with 100 epochs.

After performing three different experiments, the results showed that the three models performed much better than the previous two models especially for CSF and had more consistent DCS for the 5 cases. The final model that was chosen was based on the best multi-dice score on the data that was chosen as the validation data. As shown in table 3, 4 & 5, the DCS are very close, the best model that was chosen was the one which produced more robust results overall on all the three important classes, mainly the CSF, WM and GM. We also took into account the prediction masks for the test data and found that the 70-30 split with IBSR_05, IBSR_07 and IBSR_18 being used for training validation produced the most accurate looking masks with no holes and a good match to the testing volumes as shown in figure(9). Hence, this was chosen as our best model.

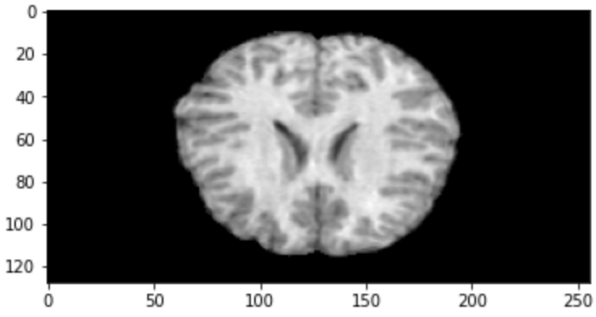


4 labels segmentation.

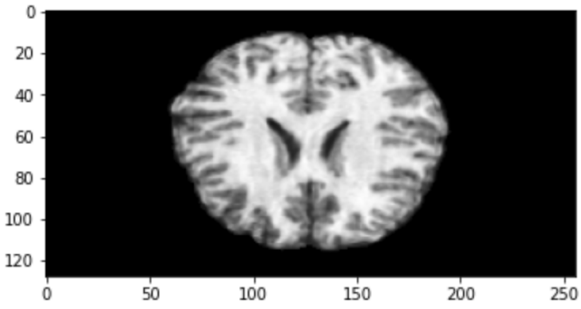


5 labels segmentation.

Figure 1: Before and after adding an extra label for IBSR_01 case



before preprocessing.



after preprocessing.

Figure 2: Before and after preprocessing for IBSR_03 case

Training Data Split into 70-30 with Pseudovalidation Data = IBSR_09,16,18			
Metric	**CSF:**	**GM:**	**WM:**
Dice	0.901628	0.942165	0.957648
	0.906363	0.930011	0.933989
	0.916708	0.953318	0.946267
	0.894813	0.940190	0.920917
	0.904252	0.939581	0.918459

Table 3: Dice Scores on the 5 volumes using the model following the first split

Training Data Split into 70-30 with Pseudovalidation Data = IBSR_05,07,18			
Metric	CSF:	GM:	WM:
Dice	0.912285	0.947861	0.933611
	0.919733	0.954250	0.946506
	0.901403	0.940333	0.921156
	0.908523	0.930708	0.937067
	0.906997	0.939458	0.956474

Table 4: Dice Scores on the 5 volumes using the model following the second split

No Split on the Training Data. All Validation Data is used for Validation			
Metric	CSF:	GM:	WM:
Dice	0.906206	0.936981	0.944343
	0.905648	0.946058	0.930449
	0.919492	0.951813	0.940063
	0.908735	0.950858	0.961366
	0.911188	0.950694	0.934234

Table 4: Mean, Max and Min Dice Scores on the 5 volumes using the model trained on all given data

		ValidationTraining Data for Model Evaluation		
		IBSR_09,16,18	**IBSR_05,07,18**	**Actual Validation Data**
Mean	CSF	0.904753	0.909788	**0.910254**
	GM	0.941053	0.942522	**0.947281**
	WM	0.935456	0.938963	**0.942091**
Max	CSF	0.916708	**0.919733**	0.919492
	GM	0.953318	**0.954250**	0.951813
	WM	0.957648	0.956474	**0.961366**
Min	CSF	0.894813	0.901403	**0.905648**
	GM	0.930011	0.930708	**0.936981**
	WM	0.918459	0.921156	**0.930449**

Table 4: Average Dice Scores on the 5 volumes using the 3 model

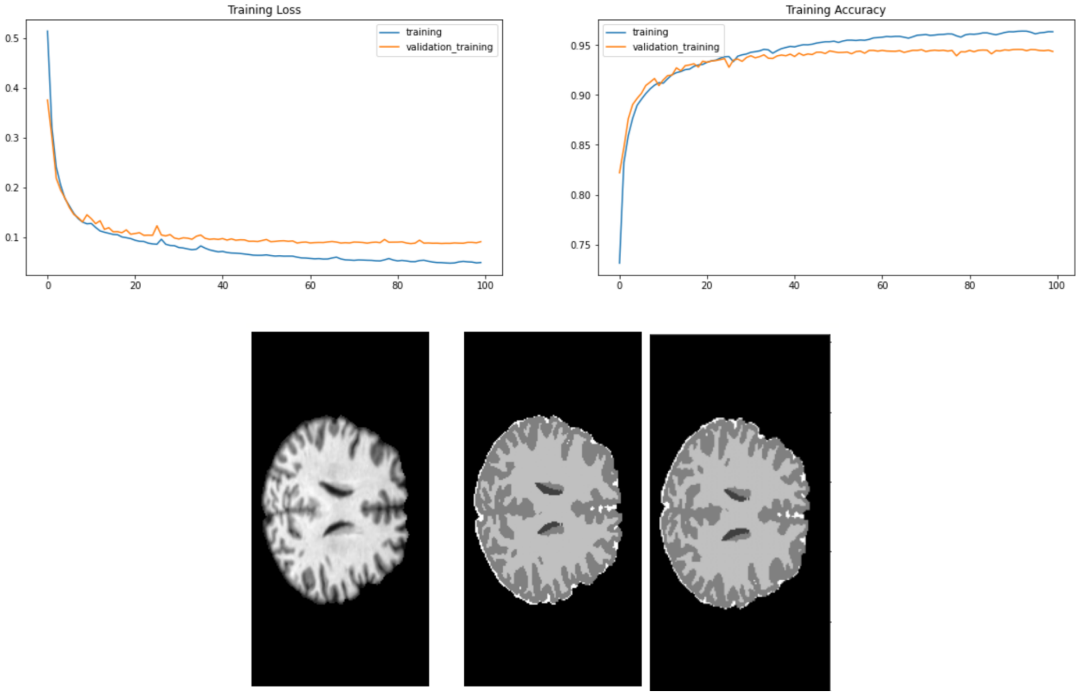


Figure 3: Final Model results for IBSR_11

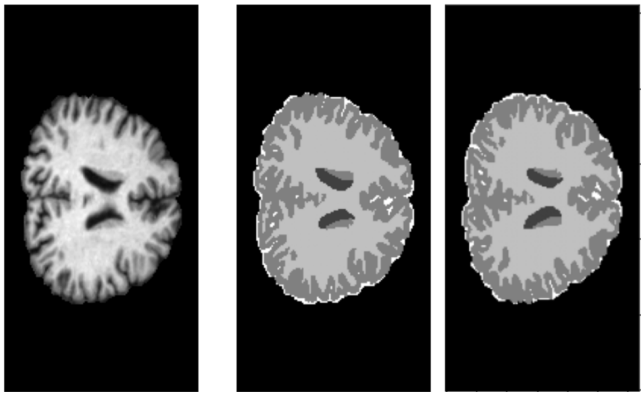


Figure 4: Final Model results for IBSR_12

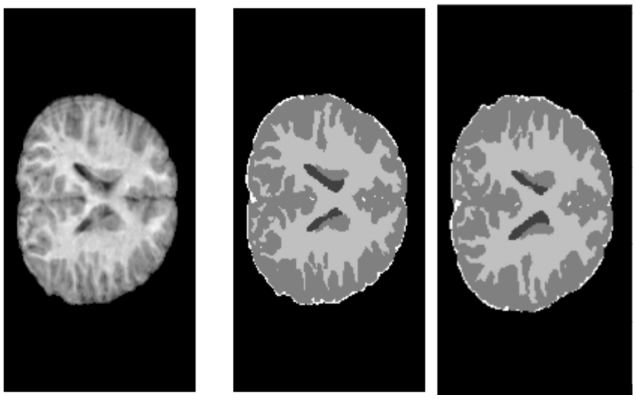


Figure 5: Final Model results for IBSR_13

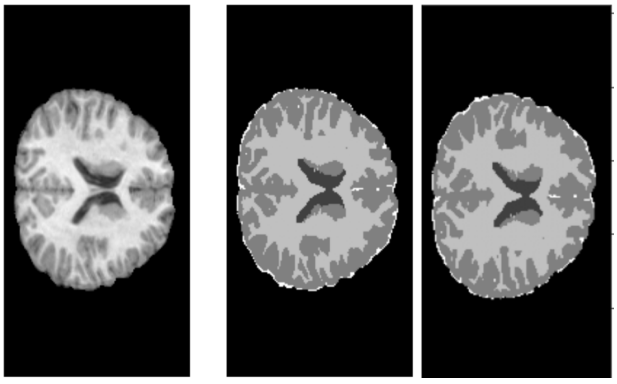


Figure 6: Final Model results for IBSR_14

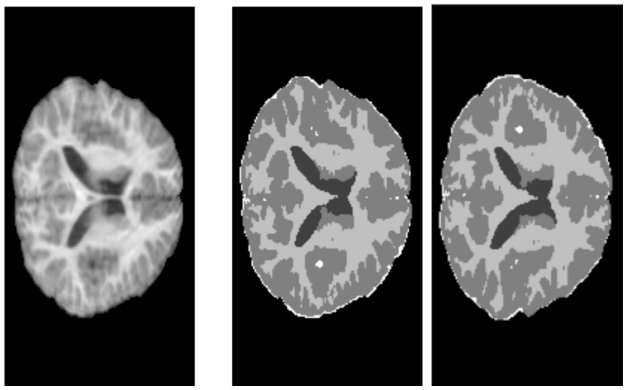


Figure 7: Final Model results for IBSR_17

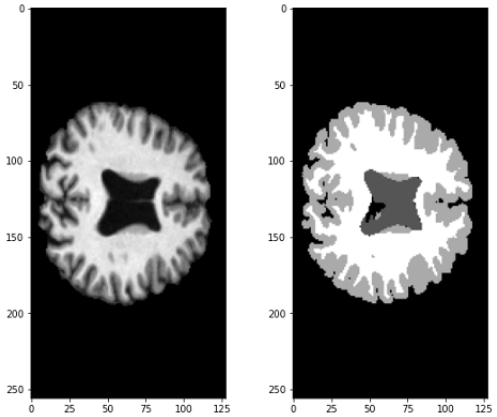


Figure 8: Resultd on IBSR_10 for Model Trained on TrainingValidation Data of IBSR 09, 16, 18

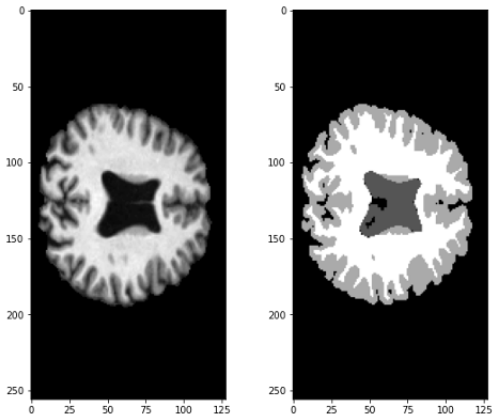


Figure 9: Resultd on IBSR_10 for Model Trained on TrainingValidation Data of IBSR 05,07,18

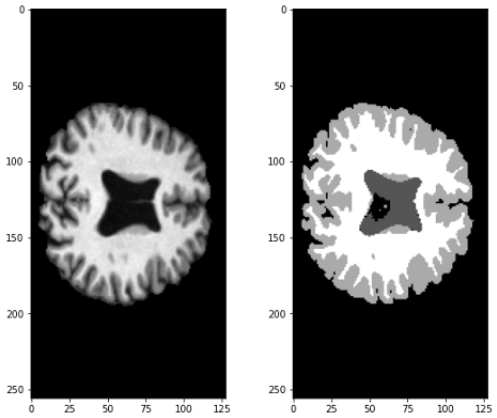


Figure 10: Resultd on IBSR_10 for Model Trained on all Validation data instead of splitting training data

7 Time Management

The first week, I and Saud started looking up literature reviews on different approaches to solve this challenge. Then, we met and decided on three different approaches(2D Unet, 3D Unet, and transfer learning) that was best fit to the problem and the limited time. The following two weeks, we implemented the three approaches with the preprocessing. The fourth week, we were optimizing the models and comparing their performance. The fifth week, we decided on one model and used it for predictions on the test and started writing the report.

8 Conclusions

The main objective of the project was to successfully segment the three tissues of the brain i.e CSF, White Matter and Gray Matter. There were many challenges involved in the project regarding the dataset which was to be dealt with. The data comes from different scanners and thus have some different modalities. There is a limited data for training. Also, there are some regions in the data which are not labeled. These are mostly around the Cerebral and the regions connected to the spinal cord. In the original data, these are marked as background which can be problematic as they have different intensities. The other issue with the data is the computational complexity and the GPU memory limitations that are required for volume segmentation. To this end, the concept of converting volumes into 3D vowels and patches was sought out. The main architecture used for this segmentation task is the UNet Architecture. In our project we made three experiments based on different interpretations of the UNet. In the first approach we developed the version of UNet the basic framework of which was provided during our Deep Learning workshop by Dr. Jose Bernal. Another approach was based on transfer learning from segmentation models and 3D NeuroNets which are already trained on brain image data. The last experiment employed the use of modified 3D UNets in the form of batch normalization and dropout. For all these three experiments we used two different approaches. Since we tackled the problem of unlabeled regions by using label generation. We ran our experiments using the data with both the original labels as well as with the generated labels. Our best model was from the third experiment which achieved the best dice scores on the validation data in terms of 0.91 0.95 0.94 for mean CSF, Gray Matter and White Matter respectively. We noticed that the extra label generation technique helped in the model segmenting all the regions better. We also noticed the cerebral regions were also very well identified by the model and did not misclassify them as white or gray matter as it would have otherwise.

9 Refrences

1. MAIA Seminar by Sergi Valverde[https://github.com/sergivalverde/MAIA seminar.git]

2. “BU-Net: Brain Tumor Segmentation Using Modified U-Net Architecture” https://www.mdpi.com/2079-9292/9/12/2203/pdf

3. Dltk models. https://github.com/DLTK/models/tree/master/ukbb_neuronet_brain_ segmentation.htm. Accessed: 2019-01-05.

4. Martin Rajchl, Nick Pawlowski, Daniel Rueckert, Paul M. Matthews, and Ben Glocker. NeuroNet: Fast and Robust Reproduction of Multiple Brain Image Segmentation Pipelines. arXiv e-prints, page arXiv:1806.04224, June 2018.