BRAIN TISSUE SEGMENTATION USING 2-D, 2.5-D AND 3D U-NET ARCHITECTURES USING DIFFERENT BACKBONES

Isaac Llorente Saguer and Zohaib Salahuddin

Universidad de Girona, Università degli studi di Cassino e del Lazio Meridionale and Université de Bourgogne

ABSTRACT

Brain Tissue Segmentation is of paramount importance for accurate diagnosis of abnormalities in Magnetic Resonance Images. Segmentation of Cerebrospinal Fluid , Grey Matter and White Matter can be used to highlight the changes which may help in identify lesions, atrophy and other changes in the brain. The Internet Brain Segmentation Repository (IBSR18) dataset was utilized which consists of 18 T1 weighted of normal subjects available from the Center for Morphometric Analysis at Massachusetts General Hospital. Dice Coefficient Values of **0.917, 0.946, 0.941** for CSF, WM and GM respectively were achieved using 2D UNet with Coronal Slices and ResNet34 backbone. Dice Coefficient Values of **0.902, 0.929, 0.921** for CSF, WM and GM respectively were obtained using 2.5D UNet with Se-ResNet50 backbones using axial, coronal and sagittal slices. The best results were obtained using patch based 3D UNet with ResNet backbone. Dice Coefficient Values of **0.927, 0.951, 0.948** were obtained for CSF, GM and WM respectively using this 3D UNet with ResNet backbone.

Index Terms— Brain Tissue Segmentation , Magnetic Resonant Imaging , IBSR18 , 2D UNet, 2.5D UNet, 3D UNet Residual Networks

1. INTRODUCTION

The Internet Brain Segmentation Repository (IBSR)[2] provides manually-guided expert segmentation results along with magnetic resonance brain image data. Its purpose is to encourage the evaluation and development of segmentation methods. Automatic volumetry on MR brain images can support diagnostic decision making [3]. The motivation and task, thus, are clearly defined: we want to have a good automatic segmentation of the brain tissues in order to improve the decision making in diagnostics.

Furthermore, this project will serve us as a stepping stone towards the Maser Thesis, diving deep into the current research trends revolving around deep learning applied to medical imaging.

2. PRE-PROCESSING

2.1. Histogram Matching

The intensities of CSF , WM and GM vary greatly for different brain MRIs in IBSR18 dataset. In order to standardize the intensity distribution, there is a need of histogram matching. The images were normalized and a reference image was chosen. The histogram of intensities of this image was used as a reference. Histograms of all images were matched to a reference image histogram. The figure 1 shows the tissue model before and after histogram matching. The intensity overlap between different tissues is minimized after histogram matching. Unfortunately, it didn't lead to better results.

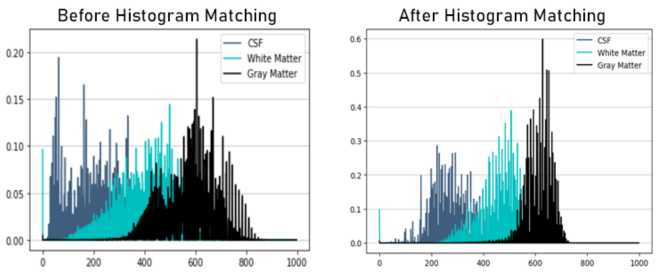


Fig. 1: This figure shows tissue models for the training images before and after histogram matching.

2.2. Data Augmentation

With the intention of having more data in the eyes of the net, and to make it more generalizing at the task of segmenting a given patch, we expanded the training set by performing whole brain intensity preserving transformations, namely flipping axis on the coronal plane. The result was a longer training time with no improvement on the performance.

2.3. Noise Removal

The experiments in this field had the effect of a shorter training time, but the results were slightly worse, so they were discontinued.

3. CONFIGURATIONS FOR UNETS

Hundreds of Experiments were done in order to tune the best parameters for 2D, 2.5D and 3D Architectures. The most important configuration parameters are discussed below.

3.1. Normalization

Normalization plays an important role in training the Neural Network effectively and enables faster convergence. Different types of normalization techniques: Batch, Group , Layer and Weight were experimented with. The following two normalization techniques were chosen in the end.

3.1.1. Batch Normalization

Batch Normalization Technique was used in the encoder part of 2D and 2.5D UNet Architectures in order to improve the performance and stability. Batch Normalization calculates the mean and standard deviation across each feature in the particular mini-batch. It moves on to subtract the mean and scale it by the standard deviation.

3.1.2. Instance Normalization

Instance Normalization Technique was utilized in the encoder part of 3D UNet Architectures. Instance Normalization is similar to Batch Normalization with the difference being that instead of normalization across the entire feature in the mini batch, normalization is applied across each channel.

3.2. Dropout

In order to avert the problem of overfitting and better generalization, dropout was added to the 3D UNet Model. Drop out helps in the synthesis of different model architectures from the same architecture by randomly drop certain proportion of nodes from the model. A dropout of **0.5** was used for the 3D UNet Architecture. After the introduction this dropout, the difference between the dice loss for training and validation was minimum which was reflective of the better generalization. Together with the Normalization techniques, the tendency of the net to overfit to the training set was less severe.

3.3. Sampling Step of 3D UNet

When dealing with 3D UNet, after defining the patch size which is dictated by the memory and GPU resources available at disposal, the overlap between the consecutive samples is defined. The greater the overlap (which means smaller sampling step), more number of patches available for training. This in turn means larger data and more training time. More data is not necessarily always good since it can hinder convergence. The Sampling Step of 8 and 16 were seen to be appropriate reflecting better performance.

In order to avoid larger training times, the model was trained with sampling step of 16 and during the inference while testing, a sampling step of 8 was used. This lead to better results and almost similar performance when using sampling step of 8 while training.

3.4. Depth of the UNet

The depth of the UNet is an important configuration parameter. The higher depth can lead to better performance but the drawback is that it can also lead to overfitting more easily, having more parameters. In case of 2D and 2.5D UNet, ResNet34 gave the best results. Increasing the depth more lead to overfitting problems while no significant increase in the accuracy. In case of 3D UNet, the maximum depth was dictated by the chosen patch size. Due to constrained resources at disposal, the maximum patch size that could be used was 32. This meant that maximum depth that could be utilized was 5. Depths from 3 to 5 were experimented with. Depth of 5 was fixed for the 3D UNet Architectures.

3.5. Optimizers

The following optimizers gave better performances for 2D , 2.5D and 3D UNet Architectures.

- **AdaDelta**: AdaDelta Optimizer decreases the amount by which the learning rate is adaptive to coordinates. This optimizer performed better for 2D and 2.5D Architectures.

- **Adam**: Adam Optimizer gave the best performance. In our configuration, it took more time to converge as compared to AdaDelta. A learning rate of 1e-3 was used at the beginning. Later, the learning rate was reduced to 1e-5 to find the optimal solution. Adam Optimizer was used for the final model for 3D UNet.

3.6. Loss Function

The Loss Funtion is a critical part of the architecture, since it drives the backpropagation in order to achieve a better performance. The following loss functions were experimented with when training 2D, 2.5D and 3D deep learning architectures.

3.6.1. Cross Entropy Loss

Cross Entropy loss examines the pixels individually and comparison is done with the target labels. Pixel wise log loss is calculated and summed over all the classes. This is computed over all the pixels and averaged. The following equation shows the expression for the log loss:

$$-\sum_{classes} y_{true} * log(y_{pred}) \qquad (1)$$

We give equal learning to each pixel in the image because of the fact that each pixel's loss is evaluated individually. In case of unbalanced classes, this could lead to a problem. Since, CSF class is under represented in the brain, more weight for CSF was also tried while training with this loss. Adding more weight to CSF improved its dice coefficient at the expense of decreasing the dice coefficient of White matter and Grey matter. Cross Entropy Loss gave comparable results to the best tuned architectures for 3D UNet architecture when training with patches. It didn't perform significantly well for 2D and 2.5 Architectures.

3.6.2. Dice Loss:

Dice Loss is essentially the overlap between the predicted segmentation and the ground truth segmentation. This overlap is then scaled by the sum of the total number of pixels in the predicted and the actual segmentation. This can be represented by the following equation:

$$DiceLoss = -\frac{Predicted \cap Actual}{\sum\limits_{pixels} Prediction + \sum\limits_{pixels} Actual} \quad (2)$$

The following versions of the dice loss were experimented with:

1. **Individual Dice Loss**: The dice loss for each of the four classes was evaluated separately. This is the reason of superior performance of dice loss as compared to cross entropy. CSF class although under represented, also gets equal weightage. Hence, Dice loss for background, white matter, grey matter and CSF were evaluated separately for each mini batch. This was then summed up and used to train the optimizer.

   This dice loss gave best performance for 2D , 2.5D and 3D Networks.

2. **Weighted Dice Loss**: The results of the above mentioned version of the dice loss were encouraging. It was noticed that the dice loss for CSF was low as compared to other classes. For this reason, a weighted version of dice loss was also introduced in order to give more importance to certain classes. CSF was given more importance and it was observed that through this weighting, maximum dice loss of 93 could be achieved for CSF class.

3.7. Metrics

The metrics used for the evaluation of the different developed Segmentation approaches are as follows:

1. **Dice Similarity Coefficient** The Dice Similarity Coefficient has already been defined in *equation 2*.

2. **Hausdorff Distance** Hausdorff distance measures how far two subsets A and B of a metric space are from each other.

   $$H = max(h(A, B), h(B, A)) \quad (3)$$

   where

   $$h = \max_{x \epsilon A} \min_{y \epsilon B} ||x - y|| \quad (4)$$

3. **Average Volumetric Distance**: the AVD is defined by:

   $$\frac{|Ground - Pred|}{\sum Ground} \quad (5)$$

   where Ground is the ground truth and Pred is the predicted volume.

3.8. Early Stopping

To further avoid overfitting, and to take advantage of a net that still has room for improvement, we implemented early stopping in a way that it tracked the validation loss, and allowed for a certain number of epochs (that we called "patience") to improve. We also checked very closely the training loss, and avoided using models where the difference between this and the validation loss was excessive (above 0.02, and watching any sudden drop).

4. EXPERIMENTS

4.1. 2D UNet Architecture

2D UNet Architecture for Segmentation is useful because of the fact that deep networks can be utilized. The problem with 3D Deep Learning Architectures is that they require a large amount of memory and resources because of the parameters needed for training. For this reason, deep networks for 3D UNet Architectures cannot be utilized. With a 2D UNet Architecture, a whole slice was fed to the network for segmentation. Mean and Standard Deviation Normalization was applied to all the images. Coronal Slices with their ground truth were used for training the 2D UNet Segmentation as it led to better results and faster convergence. This is due to the fact that the ground truth of coronal slices is not pixelated and the segmentation is smooth. Using the same ResNet Architecture, the dice loss for the axial slices based 2D models were recorded at **0.89, 0.934, 0.918** while those for model based on coronal slices were **0.917, 0.946, 0.942** respectively for CSF, Grey Matter and White Matter. Traditional UNet, ResNet 18, ResNet 34, SeResNet50, DenseNet201 were utilized for training the network. Figure 3 reflects the results which show the efficient performance of UNet with the convolutional blocks in the encoder part replaced with the Residual Blocks. The depth of this encoder was tuned at 5.

This net took about 10 seconds per epoch to train.

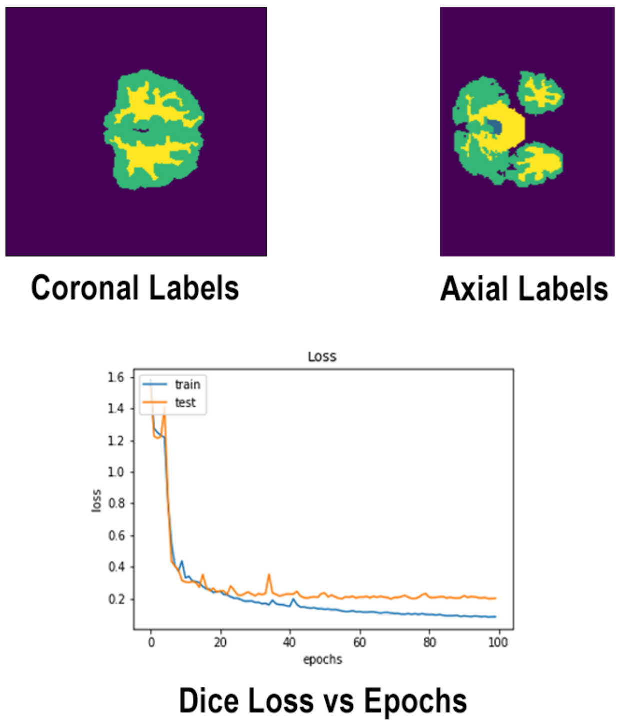


Fig. 2: This figure shows a coronal and and axial labeled slice. In the bottom row, training and validation dice loss are plotted with dropout 0.5.

4.2. 2.5 D UNet Architecture

2.5D UNet Architecture is just as fast as the 2D UNet as it is the same as segmenting an RGB image. It comes with the additional advantage that the pre-trained 2D UNet Backbones can be utilized. Sagittal, Coronal and Axial Slices are fused together after zero padding to form an RGB Image. The input image has dimensions $\mathbf{3} * \mathbf{256} * \mathbf{256}$. Mean and Standard Deviation Normalization was applied for all the images. 2.5 D UNet incorporates the information from the whole axial slice as well as information from Saggital and Coronal Views. Hence, the given three dimensional image gives more information than a 2D slice as it also incorporates information from the 3D volume. Since this information is less than all the 3D information available, it is called a 2.5D approach. UNet Architectures pre-trained on ImageNet data were used for training for 2.5D data. However, the performance of 2.5D UNet is better than the 2D UNet with axial slices but a bit worse than 2D UNet with coronal slices. This may due to the fact that the ground truth of coronal slices is smooth and easier to learn.

4.3. 3D UNet Architecture

As mentioned above, in order to be able to use a 3D analysis, due to memory constrains we have to work in patches. Being these patches 3D, it allows the net to do have some direct spatial information, but not as extensive as it would be to analyse the whole volume.

The code was adapted from the baseline given by Sergi Valverde's seminar. Figure 6 shows the architecture of this model, while Figure 5 below displays the time it took to train the model.

5. QUANTITATIVE ANALYSIS OF THE RESULTS

Although we could observe different groups of brains in the data set, in the end we went for a generalised method instead of targeting a single group for training and testing, in part due to the scarcity The amount of experiments performed are in the magnitude of hundreds. We can differentiate two main kind of experiments: the tuning ones, where we change a single parameter in order to assess the contribution of that parameter or attribute, and more disruptive experiments, in which we take a big leap from the previous known system. An example of the second could be trying a new architecture, or changing multiple attributes at the same time, hoping to start a new branch of fine tuning experiments.

On the Figure 9 we can see a bit of the evolution of our experiments: not always were we improving, and actually some of them did not finish or crashed (although they were reported anyway), but overall we could see a significant improvement of both the results and our knowledge of the different parameters. Figure 8 shows a small sample of the log file in which we kept track of the experiments we ran in order to understand the effects of different aspects of the net.

After testing different models and parameters, we designed a 7 cross-validation experiment not only to assess the generalization power of the model, but also in order to have it train with more data, and study the possibility of an ensemble. Multiple ensemble methods were considered: the addition or multiplication of each model's output for tissue probability, and the voting ensemble of the hard segmentation output of the models, with various sub models depending on how it dealt with a tie in the voting.

6. SEVEN FOLD CROSS VALIDATION

The best dice scores were obtained with 3D UNet Residual Network Model, Adam Optimizer (lr = 1e-3), Patch Size = 32, Sampling Step = 8, Instance Normalization, Drop Out = 0.5. The Validation Result for this configuration with 10 - 5 split as dictated by the project guidelines was **0.927, 0.951, 0.948**. In order to test the generalization of this model, seven fold cross validation was performed. 14 images were used for performing the seven fold cross validation and one image was used for testing.

Type of Architecture Used	Dice Coefficient				Hausdorff Distance			Average Volumetric Distance		
	CSF	GM	WM	AVG	CSF	GM	WM	CSF	GM	WM
2D ResNet	0.91459	0.94679	0.94219	0.93452	**0.0719**	**0.00098**	**0.002**	0.05502	0.02299	0.041003
2.5D ResNet34	0.9043	0.9389	0.9243	0.9225	0.27324	0.2334	0.13065	0.07849	0.02433	0.043149
3D ResNet	**0.927**	**0.951**	**0.949**	**0.9423**	0.13611	0.079088	0.07909	**0.0468**	**0.0218**	**0.034132**

Fig. 3: This figure shows the Dice Similarity Coefficient, Average Volumetric Difference and Hausdorff Distance for CSF, White Matter and Grey Matter for the best trained models of 2D, 2.5D and 3D UNet Architectures.

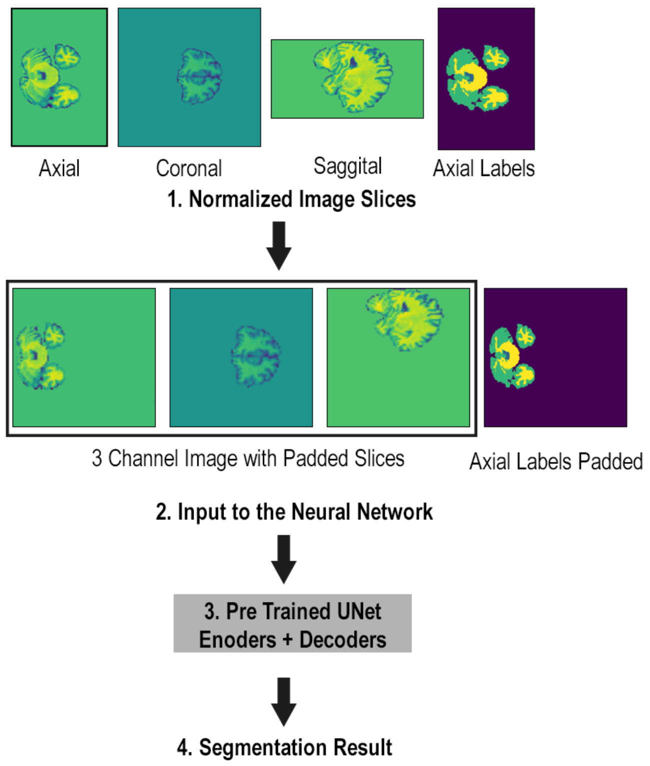

Axial
Coronal
Saggital
Axial Labels
1. Normalized Image Slices
3 Channel Image with Padded Slices
Axial Labels Padded
2. Input to the Neural Network
3. Pre Trained UNet Enoders + Decoders
4. Segmentation Result

Fig. 4: This figure shows the short pipeline for the 2.5 UNet. The 3 channel input is made after zero padding so that pre-trained image net encoders can be employed.

NET	step size	epoch [min]	num epochs	total time [min]
3D ResUNet	8	6.2	25	150
3D ResUNet	16	0.7	110	77

Fig. 5: Time measures for training the 3D model

7. QUALITATIVE ANALYSIS OF THE RESULTS

Having the segmentation output of different models, we then carefully visualized it and compared with another similar looking brain, in order to make sure that there were not any errors standing out. This helped us detect that one of the test brains, with high atrophy, wasn't properly segmented by one of our models, especially in the CSF. The 2D ResUNet based on the coronal plane proved to be the best to our knowledge,

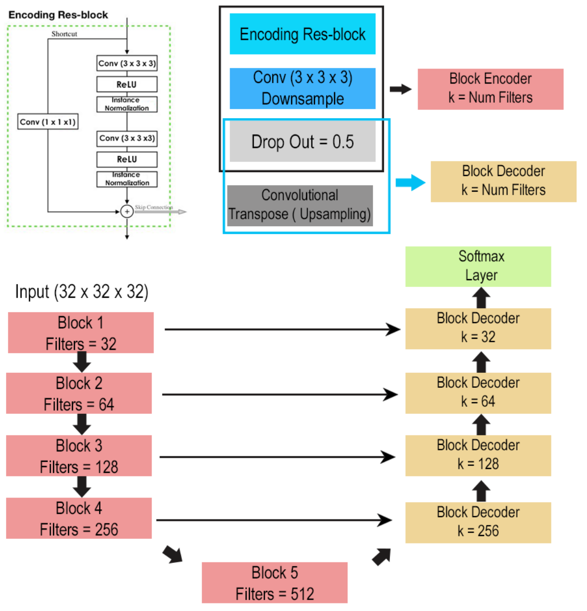


Fig. 6: The final architecture for the proposed 3D_ResUNet model

in this case, as we can observe in the following Figure 10. The patch-based 3D net doesn't have the full spatial information, so it can make mistakes that a fully "aware" method would probably not do; as an example, an atlas or our implementation of the 2D_ResUNet would probably never assign a CSF label to a voxel on a corner, having trained in this particular data set.

Another reason to load the segmentation file over the actual given test image was to make sure that the header of the nifti file had the right information (origin, orientation...).

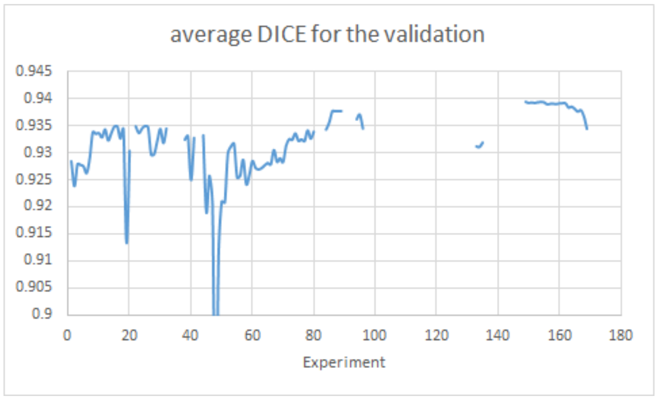
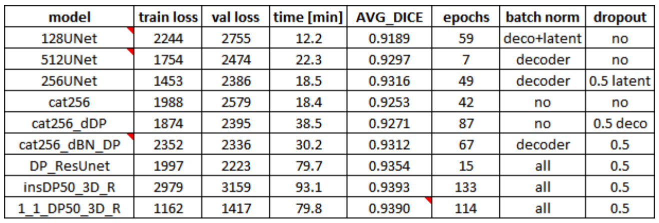
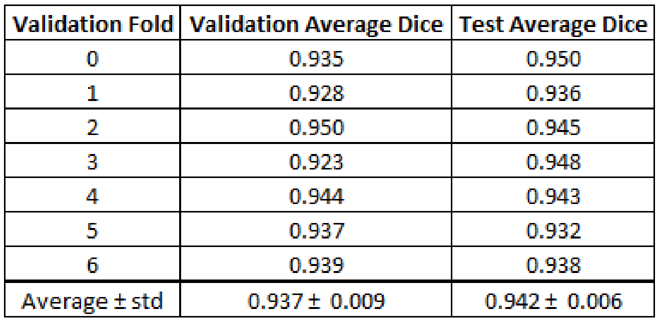


Fig. 7: Group of experiments trained on the given training set, and evaluated on the validation set



model	train loss	val loss	time [min]	AVG_DICE	epochs	batch norm	dropout
128UNet	2244	2755	12.2	0.9189	59	deco+latent	no
512UNet	1754	2474	22.3	0.9297	7	decoder	no
256UNet	1453	2386	18.5	0.9316	49	decoder	0.5 latent
cat256	1988	2579	18.4	0.9253	42	no	no
cat256_dDP	1874	2395	38.5	0.9271	87	no	0.5 deco
cat256_dBN_DP	2352	2336	30.2	0.9312	67	decoder	0.5
DP_ResUnet	1997	2223	79.7	0.9354	15	all	0.5
insDP50_3D_R	2979	3159	93.1	0.9393	133	all	0.5
1_1_DP50_3D_R	1162	1417	79.8	0.9390	114	all	0.5

Fig. 8: Sample of experiments trained on the given training set, and evaluated on the validation set. Original contains 27 columns for different evaluated parameters



Validation Fold	Validation Average Dice	Test Average Dice
0	0.935	0.950
1	0.928	0.936
2	0.950	0.945
3	0.923	0.948
4	0.944	0.943
5	0.937	0.932
6	0.939	0.938
Average ± std	0.937 ± 0.009	0.942 ± 0.006

Fig. 9: This figure shows the Average Validation Dice and Testing Image Dice for each of the seven fold. It can be seen from the result that the model generalizes well.

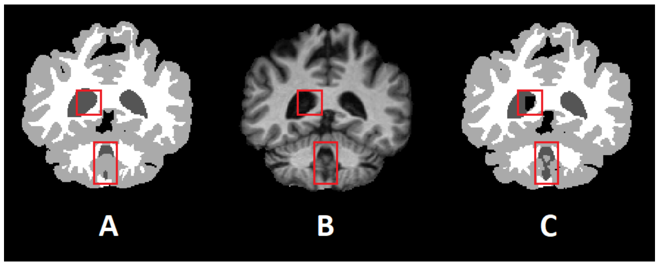


Fig. 10: Original coronal slice of the IBSR_10 (B), along with the segmentation output of the 2D_coronal ResUNet (A) and the 3D ResUNet model (C)

8. CONCLUSION

To summarize, having worked with different characteristics of different deep learning nets —such as the depth, the type of normalization, dropout, residual layers, etc., we managed to get Dice Coefficient Values of **0.927**, **0.951**, **0.948** for CSF, GM and WM respectively using a 3D UNet with ResNet backbone. The average dice score was **0.942** for the validation set.

One explanation to why the CSF score is the lowest one could be the one provided by Valverde et al. [4], where they claim that having the ground truth label Sulcal cerebrospinal fluid (SCSF) voxels as gray matter can affect the evaluation of the performance of tissue segmentation methods.

In conclusion, we saw a very good response within the available data set both when performing test-validation and cross-validation, as the results in table 10 show.

Further research could include post-processing in order to improve the segmented output, or to expand the work on the ensembles, including atlas information.

9. REFERENCES

[1] Ronneberger, O., Fischer, P. and Brox, T., 2015, October. *U-net: Convolutional networks for biomedical image segmentation*. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[2] IBSR website. Accessed 2020-01-09

[3] Heckemann, R.A., Hammers, A., Rueckert, D., Aviv, R.I., Harvey, C.J. and Hajnal, J.V., 2008. *Automatic volumetry on MR brain images can support diagnostic decision making*. BMC Medical Imaging, 8(1), p.9.

[4] Valverde, S., Oliver, A., Cabezas, M., Roura, E. and Lladó, X., 2015. *Comparison of 10 brain tissue segmentation methods using revisited IBSR annotations*. Journal of Magnetic Resonance Imaging, 41(1), pp.93-101.

[5] He, K., Zhang, X., Ren, S. and Sun, J., 2016. *Deep residual learning for image recognition*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] *Residual Networks: Implementing ResNet in Pytorch* by Francesco Zuppichini. Accessed 2020-12-12