MISA
Final Report on :
Brain Tissue Segmentation using UNet

Anwai Archit and Sheikh Adilina
University of Girona

1 Abstract

Brain Volume Segmentation is one of the most important jobs in terms of precise diagnosis of malignancies in Human Brain Magnetic Resonance Images. The tissue segmentations of Cerebrospinal Fluid (CSF), White Matter (WM) and Gray Matter (GM) help the specialists identify the lesions and atrophy changes in the Human Brain. The IBSR18 (Internet Brain Segmentation Repository) dataset has been provided for the project which consists of 18 Skull-Stripped T1-weighted MRIs availed from the Center for Morphometric Analysis at the Massachusetts General Hospital. For our best model's performance, we achieved **0.8220**, **0.8814** and **0.9163** Dice Similarity Coefficient (DSC) for CSF, WM and GM respectively using the 2D UNet. In addition to that, we obtained **0.8133**, **0.9234**, and **0.9488** DSC for the CSF, WM and GM with the introduction of Residual Block in the UNet, and **0.8015**, **0.9187**, and **0.9412** DSC for the CSF, WM and GM with the introduction of Dense Block in the UNet. In addition to the 2D Architectures, we obtained **0.7245**, **0.9563**, and **0.9771** DSC for the CSF, WM and GM using the 3D UNet. On top of all the implementations, we experimented with the recent challenge-winning algorithm, nnUNet - The "no-new-UNet". All the aforementioned models have been trained from scratch.

2 Introduction

The IBSR18 Dataset provides brain magnetic resonance imaging data with expert-handled segmentation results. Our automatic algorithm on the Tissue Segmentation task on Brain MRIs can support diagnostic decisive systems. Hence, we are in requirement of a good automatic segmentation methodology of Brain Tissues to improve the efficiency of decision making in diagnostic imaging. The tissue segmentation project focuses on diving deep into the cutting-edge trends of deep learning applications in medical imaging. Beyond our submissions, this project marks a key milestone of our journey beginning towards the world of Medical Image Segmentation and Applications.

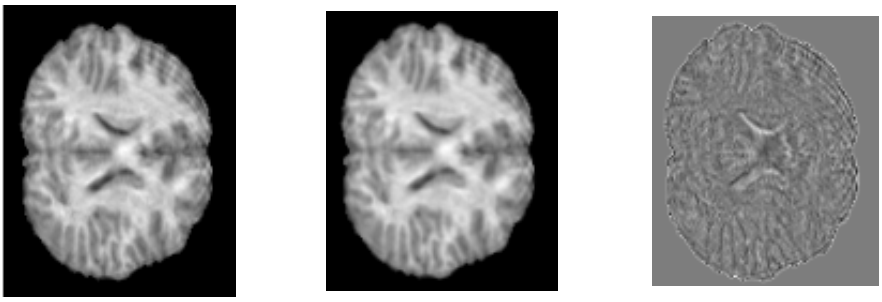
3 Preprocessing

3.1 Normalization of the Dataset

The CSF, White Matter and Gray Matter Tissues have a varying intensity distribution in the MRI Scans of Brain in the provided IBSR18 Dataset. To normalize the intensity distributions, we performed Intensity Normalization using the z-score Standardization. We take the mean and the standard deviation of the intensity distribution and tried to have a comparative understanding. The intensity overlap is minimized but it did not encourage a boost in our results.

3.2 Denoising

For the removal of noise, we implemented Anisotropic Diffusion, also known as Perona-Mallik Diffusion. It is a technique aiming at reducing image noise without removing significant parts of the image content, typically edges, lines or other details that are important for the interpretation of the image. The dataset benefited a lot from the denoising, but the results were extremely worse, so we disintegrated it from the pipeline. The 150th slice of denoised IBSR 04 volume is shown in Fig: 1.



(a) Original Image (b) Denoised Image (c) Noise Field

Figure 1: Denoising of IBSR 04

3.3 Bias Field Correction

The **N4 Bias Field Correction** was used in our implementations. This algorithm is a popular method for correcting intensity non-uniformities present in MRI image data known as a Bias or Gain Field. This method assumes a simple parametric model. This variation of N3 Algorithm couples a robust B-Spline approximation algorithm with a modified strategy which includes a multi-resolution option to capture a range of bias modulations. In our case, we used the algorithm consistently throughout all the tryouts as it provided us an improvement in our results. The 150th slice of bias corrected IBSR 04 volume is shown in Fig: 2.

(a) Original Image (b) Bias Corrected

Figure 2: Bias Correction of IBSR 04

3.4 Data Augmentation

To train deep neural networks, we are always in need of ample amount of data. To satisfy the need of our UNet to have more data and have a better generalization capability in the segmentations, we performed image transformations on the brain patches, namely flipping the patch across vertical and horizontal axis, rotation along the central axis with a particular degree of rotation and some affine transforms with predetermined range of changes. The result was an improvement in our overall performance and ensured a certain degree of robustness towards overfitting. We further continued with the pipeline throughout our experiments.

3.5 Sampling - Patch-Based Methodology

One of a major hindrances for getting higher performances with the deep neural networks is the increase in the number of parameters. Hence, it clearly identifies the requirement of more samples during the training phase. In our case, we tried it with data augmentation on the training data in the beginning. In addition to that, we thought of obtaining the segmentation patches from the brain volumes and providing them to the 2D and 3D pipelines as 2D Patches and 3D Patches respectively. We defined the patch sizes as well in the respective pipelines. For our understanding of this step, more the overlapping, more the required amount of patches for the training dataset. This means more computational cost, and more time to train the neural networks. Some of the illustrative examples of 2D Patches used in the 2D UNet pipeline are shown below in Fig: 3.

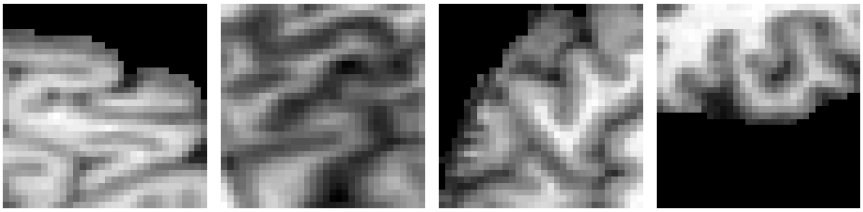


Figure 3: Several 2D Patches Extracted from the Training Dataset

4 UNet

Relentless alterations and experiments have been tested to make the best out of the fine-tuned parameters in the UNet Architectures [7]. The significant ones playing an important role in uplifting the performance are:

4.1 Normalization Layer

It plays a key role in the training of our neural network enabling faster convergence and hence effective trained outputs. There are various types of normalization methods in the market, namely : Batch Normalisation, Group Normalisation, Layer Normalisation, Weight Normalisation and so on. In our case, we implemented the normalization which was the most effective :

4.1.1 Batch Normalization

This technique has been used in the encoder section of the UNet Architecture to improve the stability and performance of the neural network. Batch Normalization evaluates the mean and standard deviation in a mini-batch across the features to subtract it with the mean and scale it with the standard deviation.

4.2 Dropout

Dropout is one of the most common practices in the training of Deep Learning Architectures. To avoid the possibilities of overfitting, the Dropout layer is added to the UNet Model. The Dropout literally drops out randomly some predefined proportion of connections from the training model, helping to synthesise the process of training in the same architecture and taking utmost advantage of a better fitting of dataset. For instance, a dropout of **0.4** has been used for our UNet Architectures. We experimented with 0.25, 0.3, and 0.5 values as well, and chose the above highlighted best one. After the addition of the dropout layer, the difference between the loss for training and validation instances reduced, symbolising generalization in the model training. The severity of overfitting sharply reduces with the introduction of Dropout in the neural network.

4.3 Depth of the Architecture

In our experiments, the depth of the UNet Architectures has played a crucial role in the stage of hyperparameter tuning and upgrading the configuration. The increment in the depth of the neural

network leads to an increase in the performance, but it clearly has a higher chance to overfit because of the increased number of parameters. In the 2D-UNet, we increased the encoder-decoder respective depths from four layers to five layers. In addition to that, we introduced the **Residual** and **Dense Blocks** to the 2D-UNet network to exploit their advantages (which will be elaborated soon). For both of our our 2D and 3D-UNet networks, we initially kept the depth for four layers. After our realisation of it's dependence with the patch size, we preferred to use the uniform batch size (also the highest configuration) as 32 (although we experimented with batch size configurations of 8, 16, and 24 as well). Our intuition for the best depth came from here, which is five layers deep neural network. We further experimented and decided upon this for all our architectures.

4.4 Optimizer

The below mentioned optimizers have been tried and have provided optimal performances for our UNet Architectures :

4.4.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is one of the trivial optimizers to draw intuition from. Theoretically, it is responsible for optimizing our objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It is regarded to be a stochastic approximation of gradient descent optimization since it replaces the actual gradient by an estimate thereof. Our dataset wasn't benefited from the trend of the descent. Although, we observed decent results with aggressive hyper parameter tuning, we decided to continue with our experiments.

4.4.2 Adadelta

Adadelta Optimization [10] is a stochastic gradient descent method that is based on adaptive learning rate per dimension to address the drawbacks of the continual decay of learning rates throughout training and the need for a manually selected global learning rate. This particular optimizer performed fairly well in our 2D UNet Architectures.

4.4.3 Adam

Adam [4] is a replacement optimization algorithm for stochastic gradient descent for training deep learning models like ours, for instance. It combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems nevertheless. In our case, Adam seemed relatively easier to configure as the default configuration works well on our problem. It gave us the best performance. It took lesser time to converge than AdaDelta. A learning rate of 1e-4 was initially configured. Later, the learning rate was reduced to 1e-5 to find our optimal solution. We kept the optimization step open ended after that using schedulers which will be explained below. Adam is hence used for all of our final UNet Architectures.

4.5 Scheduler

Schedulers are used to adjust the learning rate on the number of epochs. There are several methods such as decays the learning rate when the learning plateaus, or decays the learning rate of each parameter group by a small constant factor until the number of epochs reach a predefined milestone. Our preference in our UNet Architecture was made for using *ReduceLROnPlateau* as it allows dynamic learning rate based on some validation measurements. Scheduling has been applied after optimizer's update and seems to work fairly well.

4.6 Loss Function

The choice of loss function played a critical part in our UNet Architecture, as it plays the key role in backpropagation to achieve better trained models and optimal performances (Note : We have shown the performance trends of the Loss Functions in Table : 4, Table : 5 and Table : 6 in the Quantitative Result Assessment Section):

4.6.1 Cross Entropy Loss

The Cross Entropy Loss [12] is responsible for looking after the individual pixels and compare them with the target labels. For instance, pixel wise log-loss is calculated and summed over the classes. This is computed across all the pixels and then averaged. The below mentioned equation shows the expression for the log-loss:

$$CrossEntropyLoss = -log(p_t) \tag{1}$$

We provide balanced learning per-pixel in the volume because each pixel's loss is calculated individually. In the case of unbalanced classes, this leads to be a problem. In our case, since the Cerebrospinal Fluid is the inferior class in the Brain, we tried adding more weights to the CSF Class while training with the aforementioned loss function. Whilst adding a weight coefficient to the CSF benefited the dice coefficient, but at the cost of penalising the dice coefficient of White Matter and Gray Matter. Cross Entropy Loss provides us astonishing results with our architecture when trained with the patch-based and aggressively fine-tuned model.

4.6.2 Dice Loss

Dice Loss [9] in its simplest ways is basically an overlap between the predicted segmentation and the available/predefined ground truth. This overlap is henceforth scaled by the total sum of the total pixels in the predicted and actual masks. There are some types of the dice loss to play with in our case as we clearly observe a data imbalance, namely Individual Dice Loss and Weighted Dice Loss. The below mentioned equation shows the expression for the dice loss (for ϵ tends to 0 to avoid the denominator becoming zero):

$$DiceLoss = 1 - \frac{2 * TP + \epsilon}{(TP + FP) + (TP + FN) + \epsilon} \tag{2}$$

4.6.3 Focal Loss

To focus learning on hard misclassified samples, focal loss [6] adds a modifying term to the cross entropy loss. It is a dynamically scaled cross entropy loss, with the scaling factor vanishing as confidence in the proper class grows. In other words, it puts more focus on hard and misclassified examples. The value of p is estimated by the model for the rare class at different time points.

$$FocalLoss = -(1 - p_t)^{\gamma} log(p_t) \tag{3}$$

4.6.4 Tversky Loss

Tversky Loss [8] was created to improve segmentation on unbalanced medical datasets by adjusting the severity with which different types of errors are penalized in the loss function using constants. This loss function is weighted by the variables 'alpha' and 'beta,' which penalize false positives and false negatives to a greater extent as their value increases in the loss function. The beta constant, in particular, has implications in situations where very conservative prediction can lead to misleadingly favorable results.

$$TverskyLoss = 1 - \frac{TP}{TP + \alpha FN + \beta FP} \tag{4}$$

4.6.5 Focal-Tversky Loss

The Focal Tversky Loss (FTL) [1] is a generalisation of the tversky loss. The non-linear nature of the loss gives you control over how the loss behaves at different values of the tversky index obtained.

γ is the parameter that controls the non-linearity of the loss. In the case of class imbalance, the Focal-Tversky Loss becomes useful when $\gamma > 1$. This results in a higher loss gradient for examples where $TverskyLoss < 0.5$. This forces the model to focus on harder examples.

$$FocalTverskyLoss = (1 - TverskyLoss)^{\gamma} \tag{5}$$

4.7 Metrics

The metrics suggested to us for quantitative evaluation purposes of the devised segmentation algorithms are mentioned below :

4.7.1 Dice Similarity Coefficient

Dice Similarity Coefficient (DSC) [13] is a spatial overlap index and a reproducible validation metrics. The value of a DSC ranges from 0, indicating no spatial overlap between two sets of binary segmentation results, to 1, indicating complete overlap. The DSC equation is :

$$DiceSimilarityCoefficient = \frac{2 * TP}{(TP + FP) + (TP + FN)} \tag{6}$$

4.7.2 Hausdorff Distance

Theoretically, Hausdorff Distances measures how far two subsets of a metric space A and B are from each other. It turns the set of non-empty compact subsets of a metric space into a metric space in its own right. The Hausdorff Distance equation is :

$$HausdorffDistance = max(h(A, B), h(B, A)) \tag{7}$$

where,

$$h = maxmin_{x \in A, y \in B}||x - y|| \tag{8}$$

4.7.3 Average Volumetric Distance

The Average Volume Difference equation is defined as :

$$AverageVolumetricDistance = \frac{|GroundTruth - PredictedVolume|}{\sum GroundTruth} \tag{9}$$

4.8 Early Stopping

To counter the overfitting in the process of training and in the same time to take the optimal advantage of the neural network that still might have a chance of learning further, we have implemented the early stopping to track the validation loss in such a way that allows our algorithm to be patient for a certain number of epochs to watch for improvement using the parameter called *Patience*. We also observe the training loss and avoid using the combination of models and hyperparameters where the difference between the training loss and the validation loss is excessively high. We also attempted to add more than one schedulers, but it was not much effective on our performance, rather was increasing the complexity in the loss function space.

5 Architecture

5.1 2D UNet

The 2D UNet [7] Architecture is useful for segmentation solutions in the current era as the deep neural networks can be utilised for better performances. The Figure 4 shows the baseline architecture. We have used the Axial Planar Cross Section to implement our Patch-Based 2D UNet strategy.

The code has been adapted from the baseline architecture given by Jose Bernal's session on Deep Learning. In the beginning, the traditional SegNet and the feature maps transforms were used to gain better understanding of the encoding and decoding steps, after that the feature maps were transferred to build the UNet with requisite tunings.

The major hurdle with deeper networks for 3D UNet Architectures is that they require a large amount of memory and computational resources because of the parameters required for training. With our 2D UNet Architecture, initially the whole slices were fitted into the network for segmentation. After that, we utilised the power of patch-based method in the pipeline. Using our architecture, we obtained the training results of 0.8220, 0.8814 and 0.9163 for CSF, WM and GM respectively.

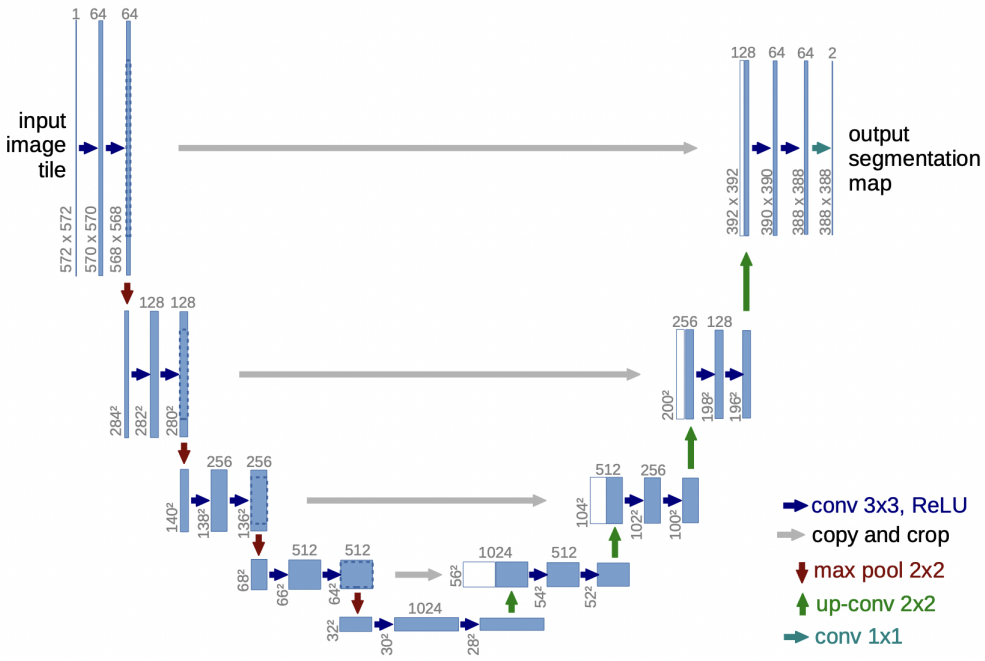


Figure 4: Baseline Architecture of the 2D UNet

5.2 2D ResUNet

The 2D ResUNet Architecture [11] refers to as the Deep Residual UNet. It is a similar encoder-decoder architecture as 2D UNet. It is designed to obtain higher performance with fewer parameters. Theoretically, it works as an improvement on the UNet Architecture and takes advantage of the Deep Residual Learning.

The residual block constructions have been built on the 2D UNet Baseline using the structuring from the simple scripts [5] of the UNet modifications.

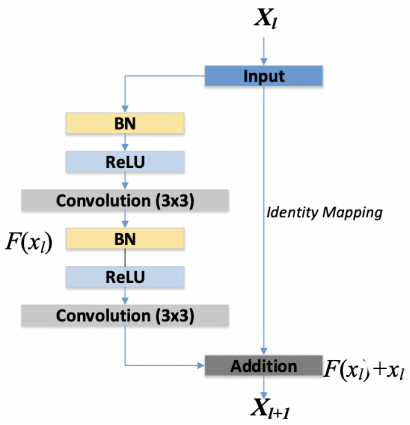
As per our understanding, the use of residual blocks in the aforementioned idea helps building a deeper network without worrying about the problem of vanishing/exploding gradients, and supports the ease of training the network. Also, the rich skip connections in the ResUNet helps in the better flow of information between the different layers which helps in the better flow of gradients while training (backpropagation).

These addition blocks were easy to play with in the 2D UNet baseline architecture because it was computationally easier to experiment and handle with a suitable depth of the neural network on top of the requisite best performing hyperparameters.

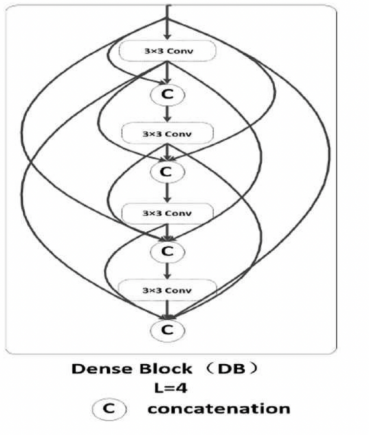
5.3 2D DenseUNet

The 2D DenseUNet [2] is introduced to address similar problems handled by the previous residual blocks. To handle the millions of parameters to handle the training and issue of gradients, we reflect the improvement using DenseUNet. The Dense Block solves the vanishing gradient problem a little better than the residual blocks as each feature map is connected by dense connectivity to drive our model to reuse the information from the previous feature maps.

The dense block constructions have been built on the 2D UNet Baseline using the structuring from the simple scripts [5] of the UNet modifications.



(a) Residual Block



(b) Dense Block

5.4 3D UNet Architecture

The 3D UNet [14] is an end-to-end learning method that automatically segments a 3D volume from a sparse annotation. It offers an almost accurate segmentation for the highly variable structure. In most of the cases, the results of the 2D and 3D Architectures in Medical Image Segmentation are comparable. In our case, we have used the patch-based methodology here as well.

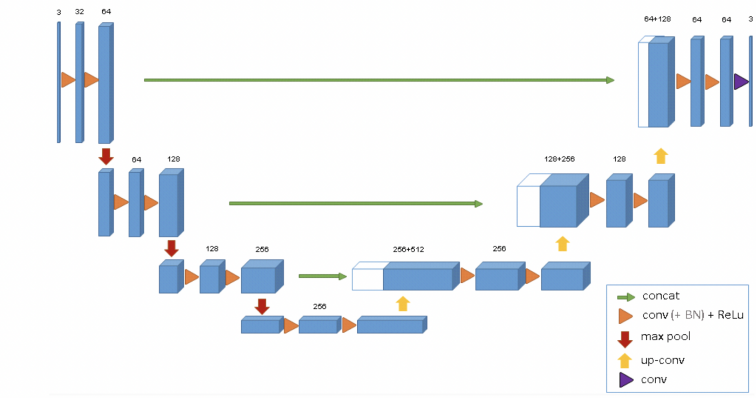


Figure 6: Baseline Architecture of the 3D UNet

5.5 nnUNet

In the 3D Medical Image Segmentation, dataset properties like imaging modalities, image sizes, voxel spacings, class ratios, etc. vary drastically per patient.

In the current research practice, segmentation pipelines are designed manually and with one specific dataset in mind. Hereby, many heuristics depend directly or indirectly on the properties of the dataset and display a complex co-dependence: Image Size, for example, affects the patch size, which in turn affects the required receptive field of the network, a factor that itself influences several other hyperparameters in the pipeline. As a result, pipelines that were developed on one (type of) dataset are inherently incompatible with other datasets in the domain.

nnUNet [3] is the first segmentation method that is designed to deal with the dataset diversity found in the domain. It condenses and automates the key decisions for designing a successful segmentation pipeline for any given dataset.

5.5.1 Preprocessing

As the first step, any custom dataset needs to be organized in their exact format for the code to be able to process the data. The folder naming too has to be exact. So we arranged the dataset in their given format and ran the code to preprocess the data. The algorithm first verifies if the data is kept in the correct format and then extracts a dataset fingerprint (a set of dataset-specific properties such as image sizes, voxel spacings, intensity information etc).

5.5.2 Model Training and Testing

After the data has been preprocessed, we moved to the training of the model. We experimented with two different configurations: 2D U-Net and 3D full resolution U-Net. For each of the configurations, the training needs to be done for Fold 0, 1, 2, 3 and 4 and for each fold the code runs for 1000 epochs by default. For the 2D configuration, each fold took around 25 hours to complete, which means it took more than 5 days to train the model. For the 3D configuration, it takes more than 33 hours to complete just 1 fold, which means it would have taken 7 days to complete training. Due to time constraints, we had to stop training the 3D configuration mid way and continue with the 2D nnUNet training.

After all the folds have been trained, nnUNet provided a command that determines which U-Net configuration to utilize for test set prediction. Then we have another command to make the predictions on the test set and output the predictions into a folder. Unfortunately, our model did the segmentation in binary because we were not aware of the extra command we needed to add for multi class segmentation during the training of the model. In the end, we did not have enough time to retrain the whole model for another 5 days. The nnUNet also generates a figure to show the progress of training which is shown in Fig: 7.

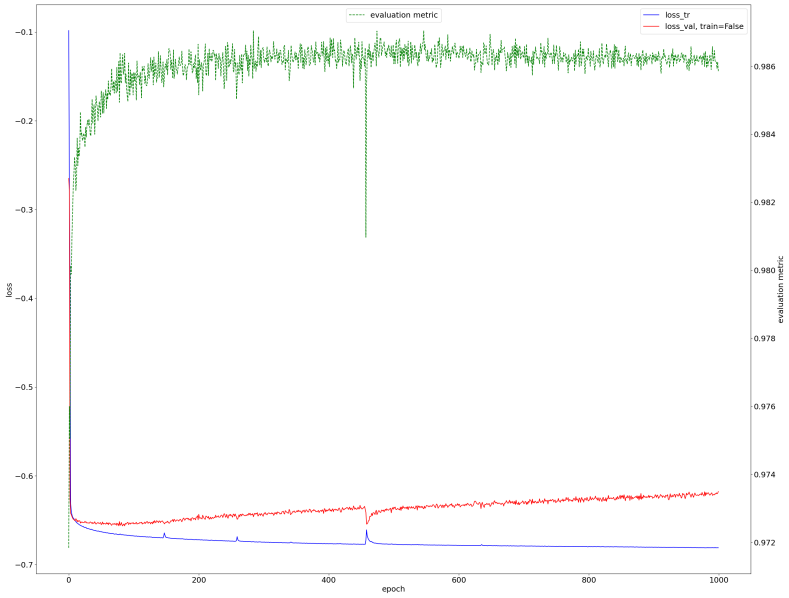


Figure 7: Auto-generated Progress of nnUNet during training of Fold 1 of 2D configuration

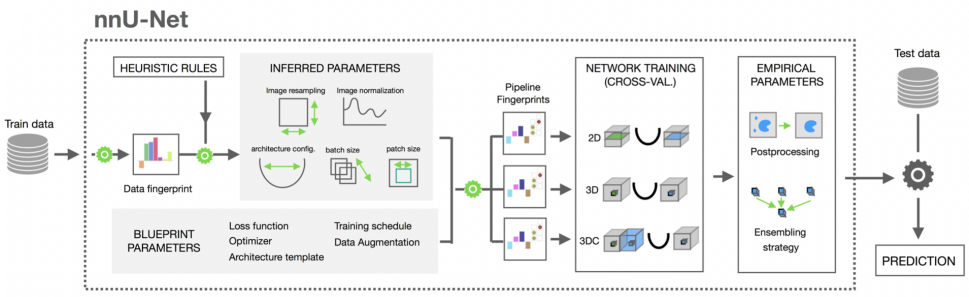


Figure 8: nnUNet

6 Project Management

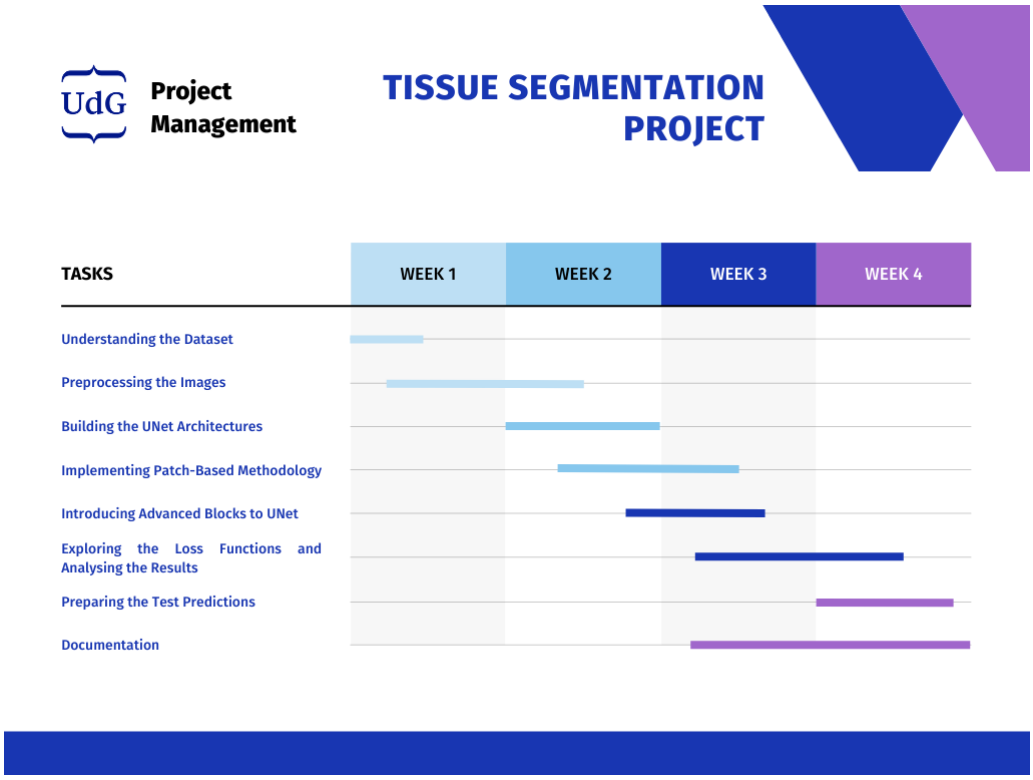


Figure 9: Gantt Chart

7 Experimental Setup

7.1 Deep Learning Frameworks

We have used Keras and PyTorch as the Deep Learning Framework here in our classical UNet and nnUNet respectively for the final project.

7.2 Computational Power

To implement the 2D UNet(s) and 3D UNet, we needed to obtain segmentations at a high computation cost. Hence, we used Google Colab which has GPU of Tesla K80-12GB and RAM of 12GB and Macbook M1-Pro Chip 16GB GPU.

To implement nnUNet we used the documentation provided in their official Github Repository. It is written in Pytorch and it cannot train without a GPU. Hence, we used Google Colab Pro which has GPU of Tesla P100-PCIE-16GB and RAM that goes upto 26.30GB.

8 Results and Discussion

8.1 Quantitative Assessment

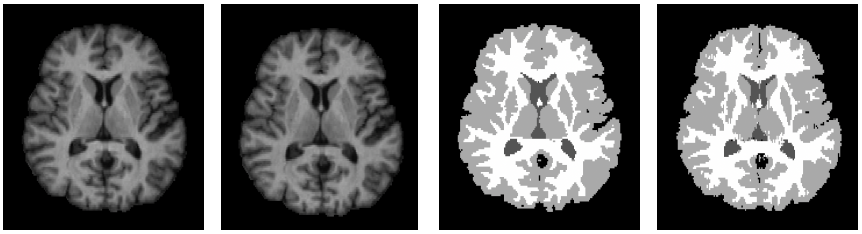
Due to the dearth of data and particularly one tissue group, our choice went forward to generalise our pipeline to use one particular simplistic model for the training data, validation data and finally the testing predictions. There have been a plethora of experiments and combinations as mentioned beforehand. In short, we have streamlined our experiments on the basis of loss functions, baseline architectures and their significant other halves with requisite betterment to increase the robustness and performance of the models. We tried to broaden our horizons by taking the necessary advancements into consideration in the paradigm of segmentation. Right from the introduction to the residual and dense blocks up until using a computationally expensive nnUNet, we diversified our choice of parameters and experimented a line of suitable set of tunings. It could have been a situation with the help of some computational support and time span to introduce new algorithms and train the dataset from scratch to bring in better results.

We also provide the relative quantitative results of the algorithms in our case of all the aforementioned implementations. In addition to that, we have included the computation time for the different trained model with their respective specifications. For the purpose of better understanding how good the overfitting has been handled, we have provided the training and validation loss trends with their provided average dice scores as well.

8.2 Qualitative Assessment

We obtained the predicted segmentation outputs of all the different models and paid utmost attention whilst performing the qualitative analysis of the tissue segmentations. This enabled us to understand that there were some appreciable parts and some parts to put our intuition into use. At the instance of nnUNet (as already mentioned above), the qualitative results needed a parameter tuning which is in-progress at the moment. Besides that, some of the brain volumes in the 3D UNet models had atrophies and missing tissue portions and/or atrophies, which put us into concerns to look upon then. For the 2D UNet models, they were surprisingly the best looking results as far as our understanding and matching through the previously provided segmentations for the validation dataset.

The Figure : 10 proves the point in our case. For our case, the patch-based method in the 3D UNet does not have the entire spatial information. Hence, it is likely to have errors unlike a full-volume segmentation strategy. For instance, in the trials of our 2D UNet with the full images, they would never mistake a CSF outside the central focused region (eg. around the edge or corner voxels of the image). Hence, it has a minor hindrance to recover which can be supported with the integration of some information from multi-atlas or ensemble of probabilistic predictions.



(a) Original Image (b) Preprocessed (c) GroundTruth (d) Prediction

Figure 10: Qualitative Results of 2D UNet on IBSR 12

Table 1: Dice Similarity Coefficient, Hausdorff Distance and Average Volumetric Distance for the CSF, WM and GM for our trained 2D and 3D UNet Architectures

		2D UNet	**2D ResUNet**	**2D DenseUNet**	**3D UNet**
Dice Coefficient	CSF	0.8220	0.8133	0.8015	0.7245
	WM	0.8814	0.9234	0.9187	0.9563
	GM	0.9163	0.9488	0.9412	0.9771
Hausdorff Distance	CSF	136.11	139.20	147.98	165.54
	WM	121.26	120.44	101.12	96.87
	GM	87.70	84.61	79.92	62.35
Average Volumetric Distance	CSF	0.1780	0.1909	0.2112	0.2998
	WM	0.1186	0.1132	0.1194	0.1084
	GM	0.0837	0.08216	0.07969	0.0761

Table 2: Epoch Times for our trained 2D and 3D UNet Architectures (with Early Stopping)

Architecture	**Per Epoch Time (sec)**	**No. of Epochs**	**Total Time**
2D UNet	17	200	28mins
2D ResUNet	23	200	40mins
2D DenseUNet	34	200	49mins
3D UNet	72	500	379mins
nnUNet 2D	90	5000	5.21 days
nnUNet 3D	120	5000	6.94 days

Table 3: Training and Validation Loss Scores for our Trained 2D and 3D UNet Architectures

Architecture	**Train Loss**	**Val Loss**	**Average Dice**
2D UNet	0.1344	0.2495	0.9184
2D ResUNet	0.1781	0.2763	0.9012
2D DenseUNet	0.1460	0.3021	0.8994
3D UNet	0.0979	0.1128	0.9025

Table 4: Dice Coefficient for 2D UNet Fine Tuning

Loss Function	**Background**	**CSF**	**GM**	**WM**
Categorical Crossentropy	0.9959	0.8729	0.9132	0.8506
Focal	0.9956	0.8450	0.9117	0.8603
Tversky	0.9962	0.7954	0.9114	0.8378
Focal-Tversky	0.9962	0.8121	0.9103	0.8343

Table 5: Hausdorff Distance for 2D UNet Fine Tuning

Loss Function	**Background**	**CSF**	**GM**	**WM**
Categorical Crossentropy	24.4949	38.1314	93.7763	109.0229
Focal	22.3159	139.0899	88.5946	111.6333
Tversky	25.4951	51.0784	100.0500	11.4455
Focal-Tversky	25.5734	58.5320	97.9081	11.9164

Table 6: Volumetric Difference for 2D UNet Fine Tuning

Loss Function	**Background**	**CSF**	**GM**	**WM**
Categorical Crossentropy	0.0041	0.1271	0.0868	0.1494
Focal	0.0044	0.1550	0.0883	0.1397
Tversky	0.0038	0.2046	0.0886	0.1622
Focal-Tversky	0.0038	0.1879	0.0897	0.1657

9 Conclusion

For our experiments, working on different types of volumes and different introduction of blocks to handle gradients and encourage skip connections in the training phase of the baseline neural network architectures with different patch sizes, depths of the UNet, introduction of dropout or normalization, additional blocks to the convolutional layers, we finally manage to obtain the Dice Coefficients of **0.8220**, **0.8814** and **0.9163** for the CSF, White Matter and Gray Matter respectively. The average DSC for the validation dataset was 0.9184.

According to our research, in IBSR18 Dataset, almost all of the models exhibit lower DSC when segmenting CSF. This outcome might be a consequence of the reduced number of samples available for this class (only the ventricular region).

Finally, we observed fairly good aforementioned outcomes from the provided dataset whilst predicting on the validation dataset. For the future works, we could introduce some strategic preprocessing/postprocessing to enhance the input images, or to improve the segmentations. We also feel the need of ensemble learning as experienced in the other projects which could aid in the robust outcomes (for instance, multi-atlas information, or probabilistic predictions from other neural networks), or maybe include segmentation using the 2.5D pipeline for the Brain MRI Images. Also, transfer learning could aid the performance as it has been trained on huge datasets with a blend of high and low-level features possibly beneficial for the segmentation, if not better. In addition to all of these, we plan to send the nnUNet predictions regardless of the results in the near future and definitely expect a better segmentation results, as we are currently working on it.

References

[1] Nabila Abraham and Naimul Khan. A novel focal tversky loss function with improved attention u-net for lesion segmentation. pages 683–687, 04 2019.

[2] Yue Cao, Shigang Liu, Yali Peng, and Jun Li. Denseunet: Densely connected unet for electron microscopy image segmentation. *IET Image Processing*, 14, 04 2020.

[3] Fabian Isensee, Paul F Jaeger, Simon A A Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203—211, February 2021.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[5] Martin Kolař'ık, Radim Burget, V'aclav Uher, Kamil Ř'ıha, and Malay Kishore Dutta. Optimized high resolution 3d dense-u-net network for brain and spine segmentation. *Applied Sciences*, 9(3):404, 2019.

[6] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. cite arxiv:1505.04597Comment: conditionally accepted at MICCAI 2015.

[8] Seyed Sadegh Salehi, Deniz Erdogmus, and Ali Gholipour. Tversky loss function for image segmentation using 3d fully convolutional deep networks. pages 379–387, 09 2017.

[9] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Lecture Notes in Computer Science*, page 240–248, 2017.

[10] Matthew D. Zeiler. Adadelta: An adaptive learning rate method, 2012.

[11] Zhengxin Zhang and Qingjie Liu. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, PP, 11 2017.

[12] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8792–8802, Red Hook, NY, USA, 2018. Curran Associates Inc.

[13] Kelly H. Zou, S. Warfield, Aditya Bharatha, Clare M. Tempany, Michael R. Kaus, Steven Haker, William M. Wells, Ferenc A. Jolesz, and Ron Kikinis. Statistical validation of image segmentation quality based on a spatial overlap index. *Academic radiology*, 11 2:178–89, 2004.

[14] Özgün Çiçek, Ahmed Abdulkadir, Soeren Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. pages 424–432, 10 2016.