IE643: Deep Learning - Theory and Practice **July-Nov 2022**

End-term Project Report : CNN-Based Medical Image Segmentation

Team Name: MetaNet *Team Members: Shivprasad Kathane 180110076, Raj Vhora 180110058*

Abstract

This project report contains the details on our project which aims to detect and segment critical regions in medical images with a specific demonstration for brain tumours. We describe the deep learning tools involved in our project. We have employed 2 different types of models and experimented with different aspects such as learning rates, loss functions, image sizes, modalities, batch sizes, and noted the performance achieved using metrics like accuracy and F1 score. We identified that the model trained using focal loss on 'Flair' MRI modality had the highest validation performance. We extend our concept to multi-tumour classification as well.

1 Introduction

Convolutional Neural Networks (CNNs) are used in a wide variety of computer vision tasks. Recent advances in semantic segmentation have enabled their application in medical image processing. Medical images may come from a variety of imaging technologies such as computed tomography (CT), ultrasound, X-ray and magnetic resonance imaging (MRI). They are used in the depiction of different anatomical structures throughout the human body, including bones, blood vessels, vertebrae and major organs. As medical images depict healthy human anatomy along with different types of unhealthy structures (tumours, injuries, lesions, etc.), segmentation commonly has two goals: delineating different anatomical structures (such as different types of bones) and detecting unhealthy tissue (such as brain lesions). Deep Learning based approaches work well for image segmentation because of the good feature-extraction capability of image convolutions in CNN and the great learning capabilities of encoder-decoder architectures. U-Net was a breakthrough neural network developed for biomedical image segmentation which consisted of contracting and expanding blocks. The paper given [2], discusses a U-Net inspired medical image segmentation method on hand and brain MRI, with the tasks of bone and tumour segmentation respectively. We closely investigate and experiment with building models on the BRATS 3D brain images data for tumour segmentation extending to classification. Certain aspects of medical images call for modifications on existing designs that are created for semantic segmentation, most notably the increased memory demands of 3D images and the imbalance between labels found in 3D ground truth data [2]. Minor modifications were done to the 3D U-Net and a high-performing model was presented in [1] for solving the BRATS 2020 challenge. We perform experiments using this model and investigate effects of different hyperparameters like image size, batch size, learning rate and aspects like loss functions, image modalities and performance metrics. Section 2 specifically highlights our contributions. Section 3 discusses the various approaches in image segmentation and has a look at the latest research in BRATS. The models, losses and other aspects of our approach are explained in detail in Section 4. Details about data and its processing is described in Section 5 while our experiments and results are shown in Section 6. Finally, we convey possible future work 7 and conclusions 8 from our project.

2 Contributions

- We investigated and employed a simple U-Net architecture
- We replicated the architecture in paper for solving BRATS 2015 challenge and experimented with it
- Because of issue with that model, investigated newer approaches and recent models
- Replicated architecture in the paper for solving BRATS 2020 challenge and experimented with it
- Varied learning rate, batch size, image size and experimented with different loss functions and image modalities, noting down performance using different metrics
- Implemented and experimented for tumour segmentation with extension to classification

3 Background and Related Works

Summary of research in image segmentation [2]: Typical classifiers like AlexNet, VGGNet, etc read in image and output a set of class probabilities. To get a semantic segmentation of the image, classification needs to be done on every pixel. This segmentation goal can be achieved by applying classifiers to patches of input image in a sliding window fashion. However, this method suffers from redundant computation. Moreover, it learns features that are local in the patches and can't learn global features as the patch size goes down. Replacing fully connected layers with 1x1 filters allows for multiclass prediction of multiple pixels in a single forward pass. This technique is efficient when the size of input is larger. Since the convolutional layers and pooling layers decrease the size of input layers, the resolution of output is often lower. This can be tackled by padding of input images. Proposals for creating a CNN that generates a segmentation map for an entire image include up-sampling the down-sampled feature maps back to original size using deconvolution operations. Alternative way of up-sampling feature maps is by storing the indices of activations that were preserved by max-pooling layers and using them later for up-sampling. Coupling a CAN with a Conditional Random Field (CRF) to enhance segmentation results has also been suggested. However, pixel-wise training using an encoder-decoder architecture has been popular in recent years for segmentation tasks.

We need to use a network that generates labels of the same size as data images. For such image segmentations, a UNet [3] network approach is best suited. The basic structure of a UNet architecture has 3 contracting blocks, a bottleneck, and 3 expanding blocks. Each contracting block has a Convolutional Layer, followed by Batch Normalisation and ReLU activation. In each successive contracting block, the number of filters in the convolutional layer increases by a fixed multiple (usually 2) while the dimensions of the activation continuously decrease. An expanding block also has a similar structure to a convolutional block, with the upsampling layer replacing the maxpool or strided convolution. However, the input to an expanding block is the concatenation (merging) of activations from its corresponding contracting block (via feature-forwarding long skip connections) and the previous expanding block. In an expanding block, the number of filters in a convolutional block decreases, and the dimensions of activations increase to generate the output with the same dimensions as the input. The bottleneck has a series of dimension-preserving sequences of convolutional layers, batch normalisation followed by ReLU. Utilising U-Net like designs is advantageous because they simultaneously capture global features and local context and also yield fast predictions (segmentation masks) after training on few samples.

We describe in Section 4, the exact details of our two networks, which are based on those found in literature and inspired by the 3D U-Net. The following figures illustrate the architectures used by [2] and [1] for solving BRATS 2015 and BRATS 2020 challenges respectively.

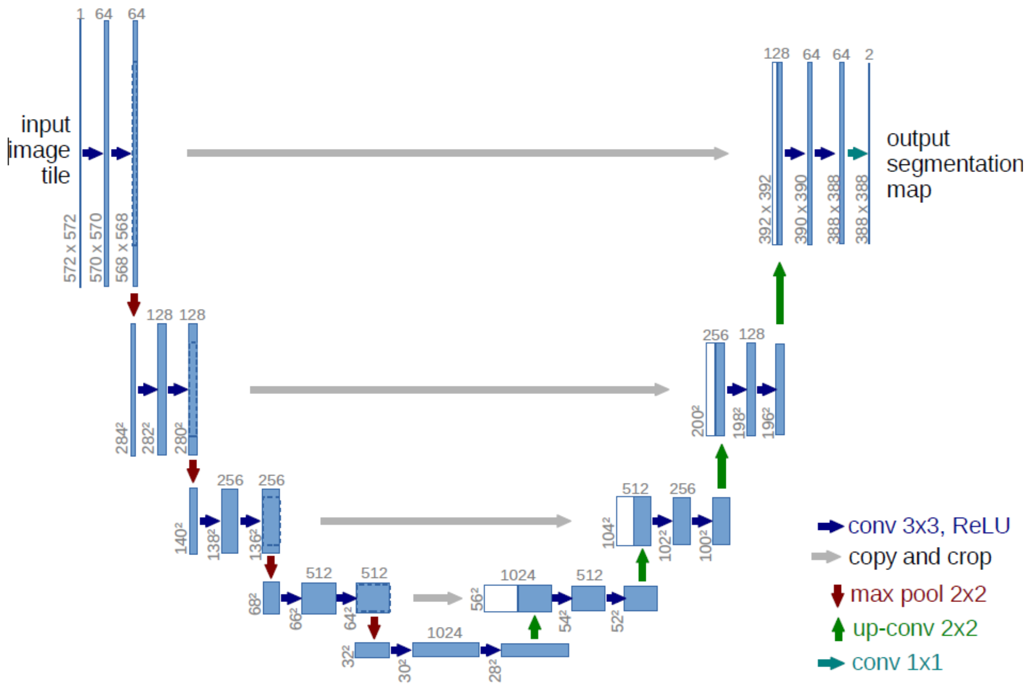


Figure 1: U-Net architecture obtained from [3]

4 Methods and Approaches

4.1 Work done before mid-term project review

- Data Acquisition: The BRATS 2015 dataset was unavailable on the website as open-source. We found the open-sourced version of the BRATS2015 dataset on Kaggle, which is used for experimentation throughout the project.
- Data Structuring: The dataset was not easily parsable. Apart from MRI scans, it had many text and licence files. The file structures were also not easily understood. As a result, a decision to parse the directory and restructure the entire dataset was taken.
- GPU Compute: After restructuring the dataset, the dataset was uploaded on Google Drive to leverage the higher RAM, and GPU compute capability of Google Colab.
- Code: The code corresponding to the paper was developed in 2017 using python 2.7 and the Theano framework. Since the project requirements were to use PyTorch or Keras frameworks, a decision was taken to rewrite the code entirely using Keras and Python 3.7 instead of cloning the old repository and replicating the work done.
- Experiment: We began with a very small image size and trained a simple UNet. Details in Section 6.

4.2 Work done after mid-term project review

Data Generator: Designed a data generator that accepts batch size, image size, and modality as inputs and gives us preprocessed data-label pairs of size equal to batch size.

Network:

We tried 2 networks which are variants of the 3D U-Net. The first network (BRATS 2015 [2]) in addition to being a U-Net consists of residual blocks (2 convolution blocks with short skip (residual) connection from input summing into output) instead of simple convolution blocks. Also, it utilises outputs from each expanding block to generate the final classification. This is done so as to combine segmentation maps across stages of the network so as to capture both global and local image features via element-wise summation and speed-up convergence. Convolution filters in blocks are of size 3 while stride or maxpool and upsampling is of factor 2.

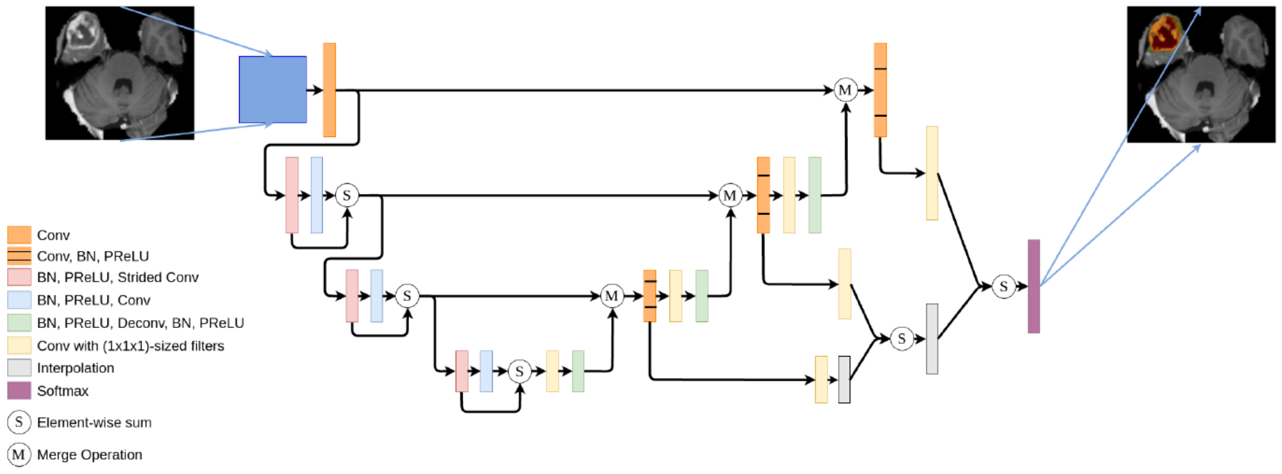


Figure 2: Architecture used to solve BRATS 2015 challenge as presented in [2]. We have replicated it.

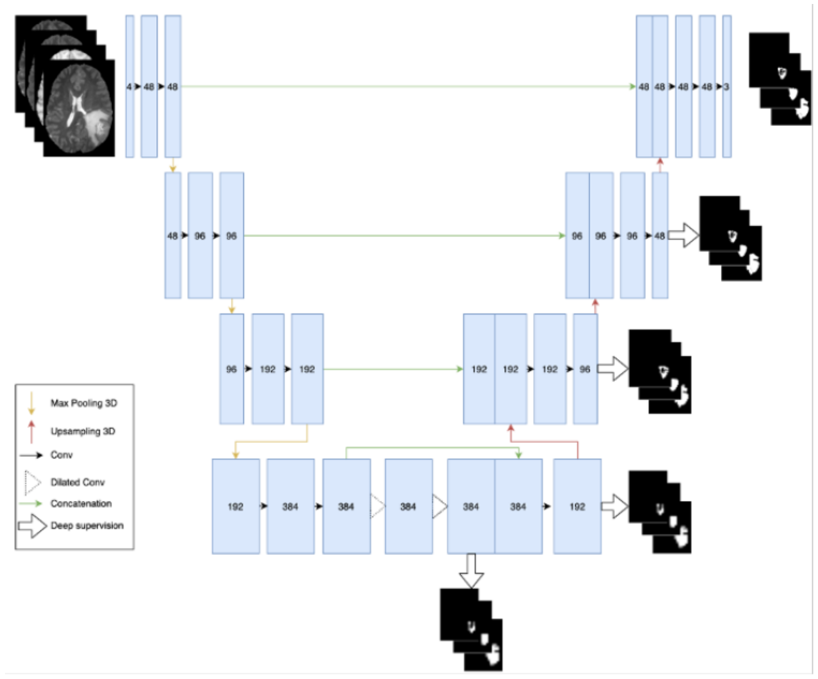


Figure 3: Architecture used to solve BRATS 2020 challenge as presented in [1]. Our design is similar.

The second network (BRATS 2020 [1]) uses dilated convolutional layers (rate = 2) for the bottleneck within the 4th stage of U-Net itself as shown in 3. This helps to expand the area of feature space covered without pooling. It also ensures there is no loss of spatial information and avoids unnecessary computational cost which otherwise comes with an additional stage in the network. It uses the output of only the final layer to generate the class. Thus, in contrast to the previous model, here there is no combination of output segmentation maps nor use of residual connections.

The networks were trained on T1c modality with a range of hyperparameters to qualitatively determine the choice of hyperparameters to be used to perform training using various loss functions. The best performing model settings were then used to train the model individually on 4 modalities. The results obtained were tabulated and the predicted masks were visualised.

Losses: We used 4 losses = BCE, BFCE, IoU Loss and Dice Loss as shown in 4.

1. Binary Cross Entropy = $L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$
2. Binary Focal Cross Entropy = $FocalLoss = -\sum_{i=1}^{i=n} \alpha_i (i - p_i)^{\gamma} log_b(p_i)$
3. Jaccard/IoU Loss = 1 - IoU where IoU = TP/(TP+FP+FN)
4. Dice Loss = 1 - Dice Score where Dice = 2TP/(2TP+FP+FN)

Figure 4: Losses used for training models in this work

The last 2 losses directly try to minimise pixel-wise false predictions. Binary Cross Entropy is the standard loss used in binary classification models. The focal loss is a variation of BCE that takes in parameters alpha (different weights for each class-term in loss) and gamma (focussing parameter) and combinedly addresses class-imbalance and hard-to-classify samples in data.

Metrics: We record the following 4 metrics in our experiments:

- Accuracy = (TP+TN)/(TP+FP+TN+FN)
- Recall R = TP/(TP+FN)
- Precision P = TP/(TP+FP)
- F1 Score = 2*P*R/(P+R)

Where TP = True Positive, FP = False Positive, TN = True Negative, FN = False Negative

In case of problems with high class imbalance such as the case here, accuracy isn't a good metric as it just focuses on correct predictions pixel-wise. With the focus of the problem being tumour detection which is the rare class among the pixel, it is important that a large proportion of them are detected (high recall) and healthy ones aren't falsely categorised as tumours (high precision). Hence, F1 which is a balanced score of the two is evaluated and desired to be high.

5 Data set Details

The 3D Brain MRI image data was downloaded from the Kaggle BRATS 2015 dataset. It was available as two folders: Training and Testing. Each of them had sub-folders corresponding to images recorded in 4 different MRI modalities: T1c, T1, T2, Flair and the ground truth (segmentation mask - label image). Each of the training sub-folders had 220 .mha files each encoding a 3D image of a brain. Each of the testing sub-folders had similar 54 files. We split the testing folder equally into the validation and test set (each having sets of 27 images). We downloaded this data and uploaded it on the drive so that we can run our experiments in Google Colab.

In order to experiment without posing the out-of-memory issue on the free GPU (12.68 GB RAM) available on Colab, we built a custom data generator which can sample images of a particular modality and shape in batches as defined by the user and allows training to take place without having all data available for once and throughout. The original dataset has all images of shape = (155, 240, 240). The authors had cropped to (128, 144, 160) for their experiments. We have experimented with different image shapes and sizes for our model. We visualised the recorded images, ground truths and predicted masks to decide which 2D image slices need to be selected and what portion needs to be cropped or resized so that input to the model is most informative for effective learning. Brains largely occupied the central 150x150 portion of the 2D image in the last 2 dimensions with almost all tumorous regions lying within it and all such 2D slices of the 3D brain could be of relevance in the 3rd dimension. Along with this, the batch size and the number of channels corresponding to the modalities make up a 5D tensor for each image which goes beyond open-source GPU capabilities. Hence, we restricted training on individual modalities and centrally cropped each image to shape = (144, 144) in the last 2 dimensions and resized the thickness to 96. We kept batch size as 1 based on training observations as well so that ultimately the output shape of our generator or model input = (96,144,144,1) is a reasonably sized 4D tensor which is well within the GPU memory limits.

We chose against data augmentation to not make training time-consuming. As the input image pixels have values between 0 to 255, we normalised them to lie between 0 and 1 by dividing it by 255. The output mask had pixel values 0, 1, 2, 3, 4 corresponding to the background and 4 tumour classes: necrosis (1), edema (2), non-enhancing (3), enhancing (4). For detecting and segmenting tumour regions, we converted all non-zero labels to 1. On the other hand, for classifying tumours, we created a separate output channel in the model corresponding to each class with 0 and 1 as labels (new ground truths) indicating the absence or presence of that tumour type at a pixel location. This input-output processing was performed in the generator.

6 Experiments and Results

As an initial preliminary experiment we trained a simple U-Net with the input layer, followed by convolution with elu activation, maxpooling, upsampling and concatenation layers (20 in total) and finally output layer with sigmoid activation for binary classification of each pixel as healthy or tumorous. The settings for this training were: learning rate = 10-6, image size = (96,96,96), loss = binary cross entropy and metric = accuracy. We trained for 100 epochs with batch size = 8 and validation split = 20% ultimately yielding loss = 0.0743 and accuracy = 0.9926 on training data and loss = 0.0727 and accuracy = 0.9934 on validation data. However, despite such a high accuracy, the predictions failed to detect tumours. This is because of high class imbalance as most of the pixels represent healthy brain sections. Therefore, metrics which quantify performance taking all classes into account and loss functions which enable learning targeting this issue in mind need to be used. In different sessions of Colab, the following were varied and we obtained similar results on different modalities using various losses. We emphasised on qualitative inference in these experiments.

Model Architecture: We replicated the model present in [2]. It consisted of strided convolutions. This had 130M+ parameters. We replaced the strided convolutions with max pooling and shifted a few layers across the blocks. This reduced the number of parameters to 90M+. We tried different image sizes, model parameters, and varied the number of blocks while training for a varied number of epochs but soon realised that the same results were obtained and training wasn't actually taking place as the loss and metrics didn't change with epochs leading to the conclusion that there were some issues with these models. So we decided to search for models in recent literature solving the latest BRATS challenges. This is where we found a model in [1] which was a solution submission to BRATS 2020. We replicated this model which consisted of 23M+ parameters. This model was able to train well and we decided to employ it for future experiments but each epoch took 10-12 minutes as training time. As there is a limit on maximum duration of continuous free-GPU access on Colab, we decided to train the model in most of our experiments for 2 or 5 epochs to check whether training happens or not and to check how it proceeds and record changes in loss values and metrics.

Image Size: Majority of the experimentation in image sizes was done with two aims. One was to reduce the issue of class imbalance apriori during data-processing itself by clever use of cropping and resizing. The other was to observe the tradeoff between increased memory-demand of large-sized images versus their high-resolution information capture. We decided to stick with image shape = (96, 144, 144) so as to capture maximum information within the memory limit.

Batch Size: We tried out different batch sizes = [1, 2, 3, 4] for different image sizes. Higher batch sizes led to faster training but posed memory issues requiring reduced image sizes. We decided not to reduce image sizes for further experiments while keeping the batch size as 1.

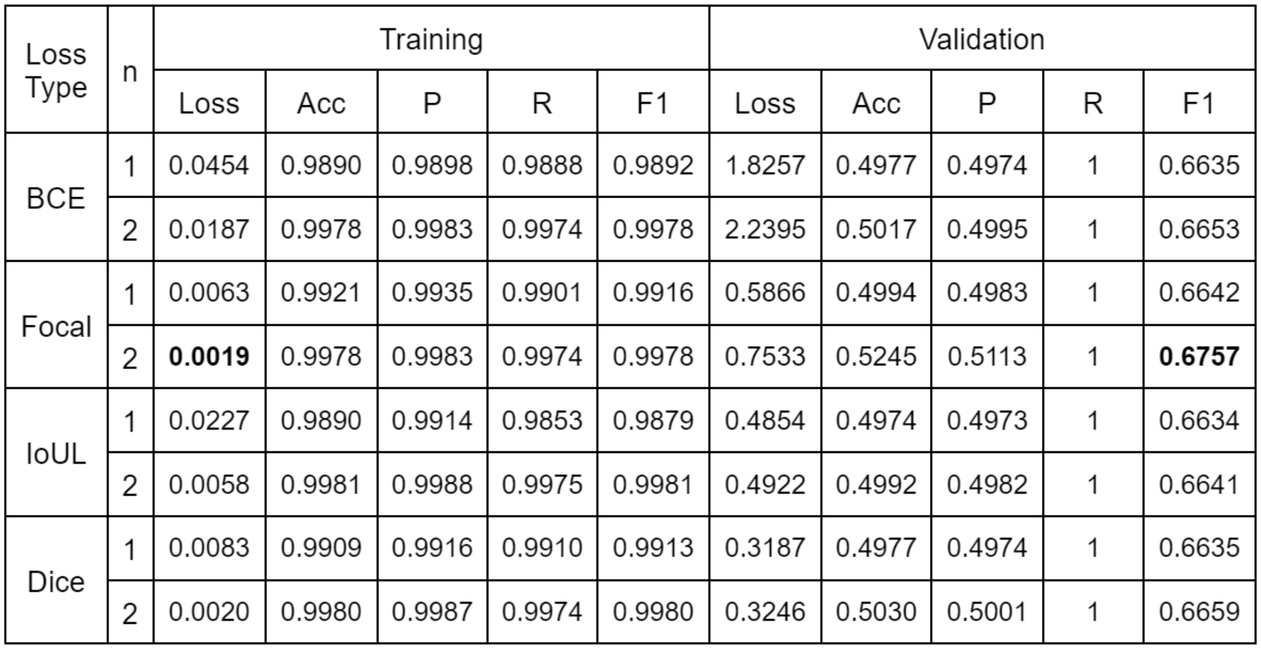
Learning Rates: We tried out different learning rates = [0.01, 0.001, 0.0001]. Smoother training with regular improvement in metrics was observed for 0.0001 in contrast to 0.01 where the metrics fluctuated with epochs. Hence, we decided to fix 0.0001 as the LR for future training.

We then trained our model with 4 different loss functions on T1c modality each for 2 epochs. P = Precision, R = Recall, Acc = Accuracy, F1 = F1 Score, n = Epoch BCE = Binary Cross Entropy Loss Function available in Keras Focal = Binary Focal Cross Entropy Loss available in Keras IoUL = Loss based on Intersection-Over-Union coefficient = 1 - IoU Dice = Loss based on the Dice Coefficient = 1 - Dice Score IoUL and Dice were functions defined by us in a CustomLoss class

Although the loss functions aren't comparable yet we note that the model trained using the focal loss had yielded the highest validation F1 score and lowest training loss. We then use focal loss to train our model on 4 different modalities: 'Flair', 'T1c', 'T1', 'T2'. Note: B - for Best F1, F - Final

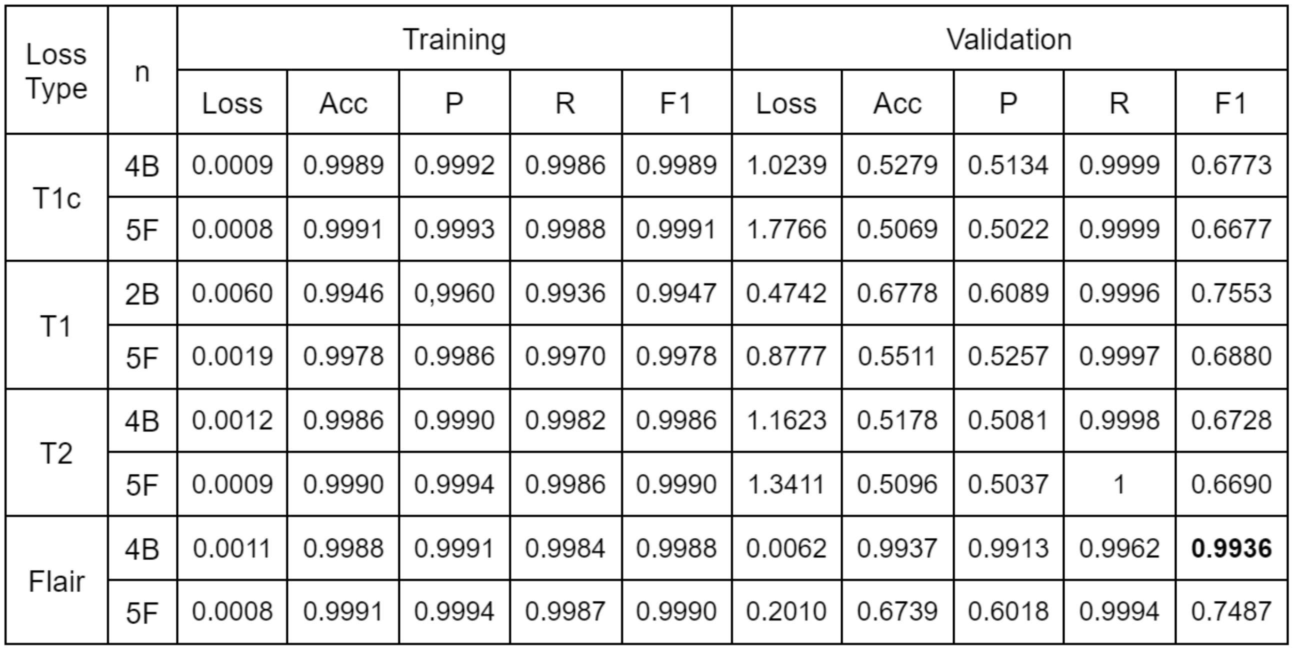
Highest validation F1 score is observed for the model trained on 'Flair' modality. Although trained for 5 epochs only, it was observed that F1 rises and then falls after a few epochs. Going for an even lower learning rate and reducing image size to observe learning behaviour in the long-run over more epochs could be performed next for conclusive results.

Note: The results seemed to be good so when we went for visualising our predictions vs labels, it is there when we realised that while adding functionality to the data generator for sampling images of any modality we had mistakenly loaded copy of input as labels instead of the ground truth because of which essentially our network acted like an autoencoder. Thus, all these results essentially are inconclusive but represent our approach. However, before we had added this last complexity, our code was error-free and we were able to segment some tumorous portions.



Loss Type	n	Training					Validation				
		Loss	Acc	P	R	F1	Loss	Acc	P	R	F1
BCE	1	0.0454	0.9890	0.9898	0.9888	0.9892	1.8257	0.4977	0.4974	1	0.6635
	2	0.0187	0.9978	0.9983	0.9974	0.9978	2.2395	0.5017	0.4995	1	0.6653
Focal	1	0.0063	0.9921	0.9935	0.9901	0.9916	0.5866	0.4994	0.4983	1	0.6642
	2	**0.0019**	0.9978	0.9983	0.9974	0.9978	0.7533	0.5245	0.5113	1	**0.6757**
IoUL	1	0.0227	0.9890	0.9914	0.9853	0.9879	0.4854	0.4974	0.4973	1	0.6634
	2	0.0058	0.9981	0.9988	0.9975	0.9981	0.4922	0.4992	0.4982	1	0.6641
Dice	1	0.0083	0.9909	0.9916	0.9910	0.9913	0.3187	0.4977	0.4974	1	0.6635
	2	0.0020	0.9980	0.9987	0.9974	0.9980	0.3246	0.5030	0.5001	1	0.6659

Figure 5: Loss and metrics recorded on train and validation sets for training using various loss functions



Loss Type	n	Training					Validation				
		Loss	Acc	P	R	F1	Loss	Acc	P	R	F1
T1c	4B	0.0009	0.9989	0.9992	0.9986	0.9989	1.0239	0.5279	0.5134	0.9999	0.6773
	5F	0.0008	0.9991	0.9993	0.9988	0.9991	1.7766	0.5069	0.5022	0.9999	0.6677
T1	2B	0.0060	0.9946	0,9960	0.9936	0.9947	0.4742	0.6778	0.6089	0.9996	0.7553
	5F	0.0019	0.9978	0.9986	0.9970	0.9978	0.8777	0.5511	0.5257	0.9997	0.6880
T2	4B	0.0012	0.9986	0.9990	0.9982	0.9986	1.1623	0.5178	0.5081	0.9998	0.6728
	5F	0.0009	0.9990	0.9994	0.9986	0.9990	1.3411	0.5096	0.5037	1	0.6690
Flair	4B	0.0011	0.9988	0.9991	0.9984	0.9988	0.0062	0.9937	0.9913	0.9962	**0.9936**
	5F	0.0008	0.9991	0.9994	0.9987	0.9990	0.2010	0.6739	0.6018	0.9994	0.7487

Figure 6: Loss and metrics recorded on train and validation sets for model trained on various modalities

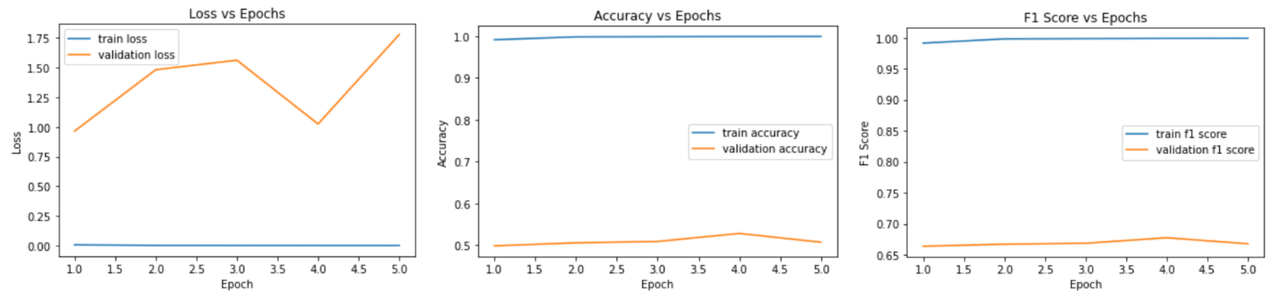


Figure 7: Comparing train vs validation loss, accuracy and F1 score with epochs for 'T1c' modality

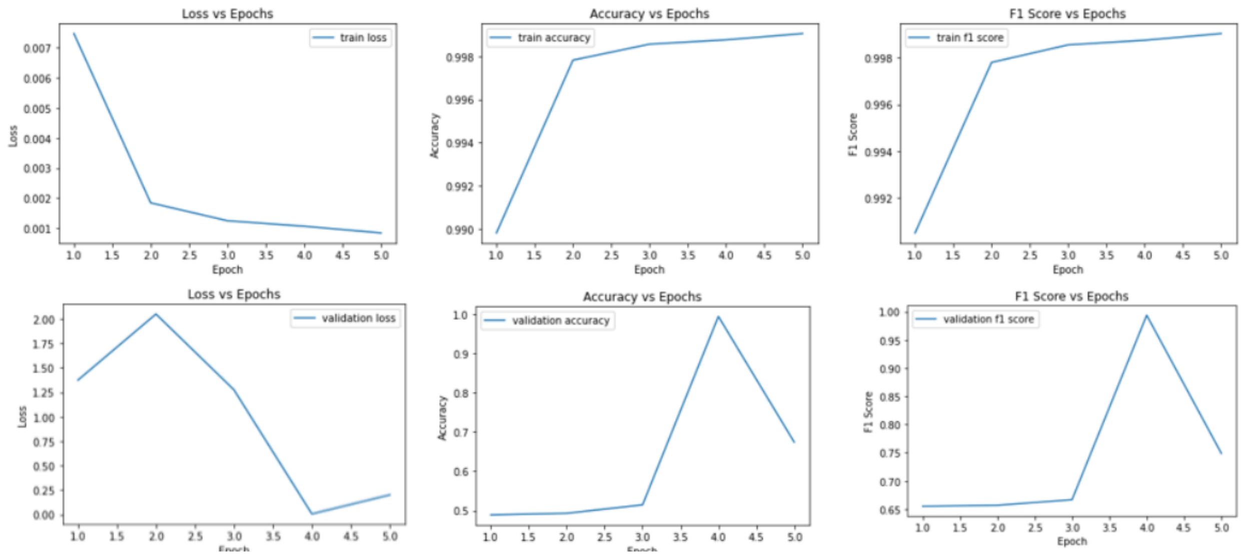


Figure 8: A representative visualisation of variation in loss, accuracy and F1 score with epochs (for Flair modality) in order to closely probe training and validation individually

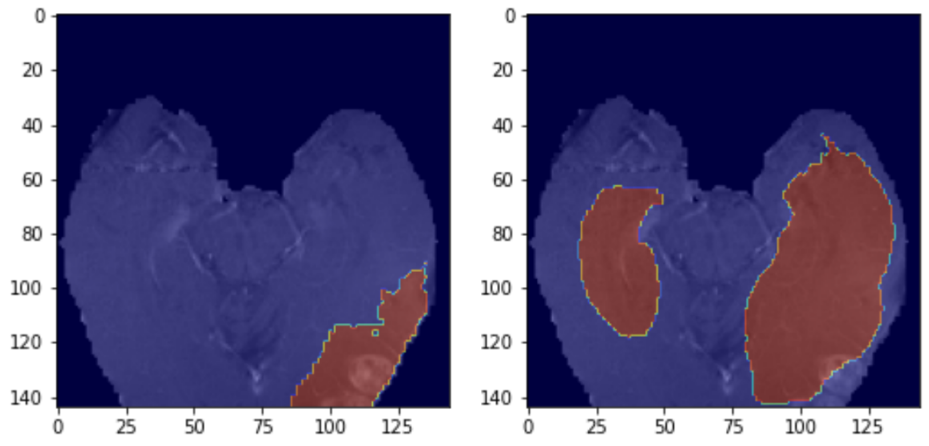


Figure 9: Example of a label (left) vs prediction (right) on test data for such a model trained earlier

7 Future Work

The following experiments could be tried as an extension to the work done. We experimented with individual modalities. Experiments can be tried using combinations of modalities taken together. Taking a larger image size increases memory usage. A smaller image size and increased batch size can be tried to speed up the training of models. The experiments in the report were largely based on binary classification between tumour and background image. A more sophisticated multilabel classification can be tried out to detect the type of tumour. A huge class imbalance is observed in the dataset. The model can be trained using a weight adjusted loss function that depends upon the weights given to particular classes.

8 Conclusion

We describe CNN-based approaches to detect and segment tumorous regions from 3D images of the brain. We have employed 2 different types of models and experimented with different aspects such as learning rates, loss functions, image sizes, modalities, batch sizes, and noted the performance achieved using metrics like accuracy and F1 score. We identified that the model trained using focal loss on 'Flair' MRI modality had the highest validation performance. We extend our concept to multi-tumour classification as well. Through our investigations on this problem of medical image segmentation we realise data processing is intricate but necessary due to high memory demand of 3D images. A properly designed datagenerator can effectively be used to address RAM limitations and memory problems. Medical images have a very high class imbalance and can be addressed using grouping of labels and better loss function design. Neural Network Architectures need to be robust and overfitting should be avoided by limiting the training epochs. Neural Networks should be appropriately deep. Networks too deep take a long time to train and sometimes don't train at all while shallow networks don't learn all the details.

References

[1] Theophraste Henry, Alexandre Carre, Marvin Lerousseau, Theo Estienne, Charlotte Robert, Nikos Paragios, and Eric Deutsch. Brain tumor segmentation with self-ensembled, deeply-supervised 3d u-net neural networks: a brats 2020 challenge solution, 2020.

[2] Baris Kayalibay, Grady Jensen, and Patrick van der Smagt. Cnn-based segmentation of medical imaging data, 2017.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.