IMPROVING BRAIN TUMOR SEGMENTATION ON U-NET

DEEP LEARNING PROJECT

Aylin Topcu
Department of Computer Science
Aarhus University
202303479

Mikkel H. S. Roos
Department of Computer Science
Aarhus University
201804414

Johan K. Nielsen
Department of Computer Science
Aarhus University
201807119

December 8, 2023

ABSTRACT

In this project on brain tumor segmentation using the U-Net model on the BraTS 2021 dataset, we tried to improve the model through a set of experiments. We initially considered making architectural changes to the model. However, we shifted our focus towards optimizing other aspects such as data preprocessing, loss function, learning rate, and other hyperparameters to optimize the training process both in terms of accuracy and hardware usage. This approach led to a significant improvement in the model's performance. By using a combined loss function to address class imbalance, optimizing hyperparameters, and using a learning rate scheduler, we managed to increase the training dice score of the model from 48.44% to 77.27%, and the validation dice score from an initial 43.17% to 62.66%.

Keywords Deep Learning, Convolutional Neural Network, U-Net, Brain Tumor Segmentation

1 Introduction

Brain tumor segmentation using deep learning models has been a popular area of research for several years. Convolutional neural networks (CNNs), U-Net in particular, has been proven to be very successful in the area of medical image segmentation.

This project focuses on the utilization of the U-Net architecture for brain tumor segmentation. Even though U-Net is successful in image segmentation, the task of segmenting brain tumors presents several challenges that need to be addressed to achieve good results. One of the main challenges in this project is the inherent class imbalance present in the dataset, where some tumor types are less common than others and non-tumorous parts of the brain cover most of the MRI scans. Another key challenge is the complexity of tumor morphology and variability in tumor appearance across different scans caused by different imaging standards adopted by MRI machines.

Our goal for this project is to enhance a U-Net-based model to effectively segment brain tumors in the BraTS dataset. We would like to train a U-Net model and fine-tune it for our task through experimentation on training parameters, data preprocessing and model architecture. The main objective is to improve the model's performance in terms of accuracy and generalization by overcoming the mentioned challenges in the MRI data.

Without disregarding the importance of medical brain tumor segmentation, it should be noted that the BraTS challenge also motivates BraTS specific optimizations Isensee et al. [2020], directly aimed at improving the BraTS evaluation. We are treating BraTS purely as a dataset, and apply more general optimizations which could be applied to entirely different CNN domains.

2 Related work

Isensee et al. [2018] argued against complex architectural optimizations, and won 2nd place in Brats 2018 with the original U-net, using only minor modifications and an improved training process. This paper addresses class imbalance issue with a dice loss function and utilizes extensive data augmentation. Same point was extended in Isensee et al.

[2020] where they came 1st in BraTS20. Kugelman et al. [2022] compared architectural variants of U-Net on other data sets in the field, and found the benefits to be marginal.

Zhou et al. [2019] presented an approach of integrating separate segmentation tasks into a single model, One-pass Multi-task Network, and using a cross-task guided attention mechanism. It achieved notable results on BraTS dataset. However, despite being our initial plan, integrating OM-Net's complex architecture features into U-Net was challenging. As a result, our project concentrated on improving U-Net through other modifications.

Futrega et al. [2021] presented an optimized U-Net architecture that won the validation phase and took third place in the test phase in BraTS21 challenge. They conducted extensive experiments on different U-Net variant architectures including U-Net, nnU-Net and UNETR. Their results showed that among these architectures baseline U-Net achieved the highest score on BraTS data. They have also experimented to find the optimal depth of the U-Net encoder, number of convolutional channels, post-processing strategy, and use of drop block, residual connections, learning rate scheduler, deep supervision.

The above findings motivates non-architectural optimizations of our U-Net, which we will focus on.

3 Methods

This section describes our attempt at improving performance of image segmentation on brain MRI scans. We go through the architecture, data structure, experimentation and training of the model.

3.1 U-Net

Our project uses the original U-Net Ronneberger et al. [2015] architecture, which was originally designed for image segmentation tasks on biomedical datasets. Many of the top performing BraTS models have been utilizing U-Net or variants thereof, highlighting its performance in the field. The architecture of U-Net allows capturing of both local and global features. This makes it specifically useful for image segmentation.

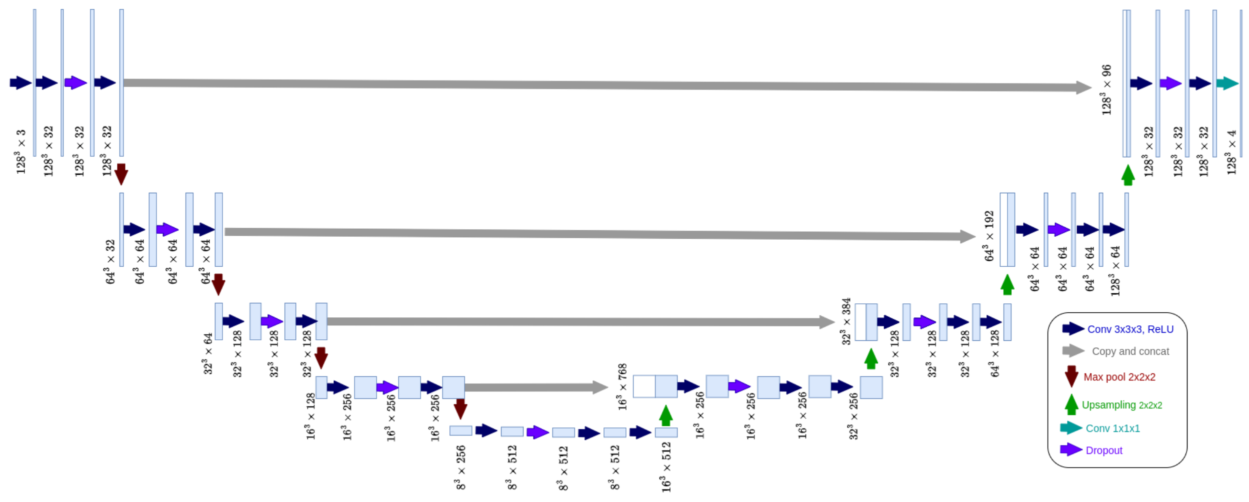


Figure 1: U-net architecture

The idea of the u-shaped architecture in U-Net is to maintain localization accuracy of the labels without losing the surrounding context of the voxel(s). This is done through the concatenate layers, also called skip connections. This connection combines the contracting (left) side and the expanding (right) side of the u-shape by copying the left, max-pooled side to match the size of the right, up-sampled side. This can be seen at the end of each *Copy and concat* connection in Figure 1.

As we focus on non-architectural optimizations, we keep the original U-Net architecture with specifications that fit our goals. Our project utilizes a 3-dimensional version of U-Net. The architecture remains the same, though the convolutional, max-pooling and up-sampling layers are 3-dimensional instead of the original 2 dimensions. Formally, this does not change the behaviour of the model, but in practice this lets us input all images from an MRI scan as a single input for training. In Limitations (5.1), we will see the consequences of this approach. Furthermore, we use a multi-class variation of U-Net. In practice, this means we have a 3-dimensional convolution layer at the end with the number of classes, that uses softmax activation. This gives us the probabilities of each class.

3.2 Dataset

The BraTS dataset consists of ~ 1250 3D MRI multimodal brain scans. Each brain MRI scan consists of 4 modalities, a) native T1-weighted (T1) and b) post-contrast T1-weighted (T1Gd/T1ce), c) T2-weighted (T2), and d) T2 Fluid Attenuated Inversion Recovery (T2-FLAIR) volumes, acquired with different clinical protocols and various scanners from multiple institutions. It is explained by organizers of BraTS Challenge that all the imaging data have been segmented manually, following the same annotation protocol, and their ground truth annotations were approved by experienced neuroradiologists. The segmentation of scans consists of 4 classes: 'NOT TUMOR', 'CORE TUMOR', 'EDEMA', and 'ENHANCING TUMOR'. The scans are available as NIfTI (.nii) files. We used the BraTS21 dataset which was the dataset used for the annual BraTS challenge in 2021. The use of the BraTS21 dataset in this project provided access to high-quality data. Two samples from the dataset can be seen in Figure 2.

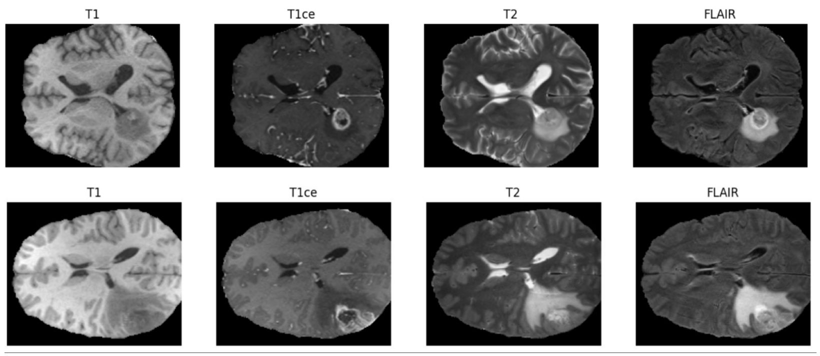


Figure 2: MRI Scan Samples from BraTS21 Dataset

The input we wanted to feed to our U-Net model was of the shape (128,128,128,3), that is: brain scan slices of size 128x128 pixels and 128 slices with 3 of the 4 modalities. The process of preprocessing is described later in the section 3.3.1.

3.3 Experimentation

In this section, we will describe our methods and argue for the parameters used to achieve our goals.

3.3.1 Data Preprocessing

In the image preprocessing stage, we prepared the MRI data to feed it into a convolutional neural network. The MRI data is loaded from NIfTI files (.nii.gz), a common format in medical imaging. The approach is inspired by a solution found on YouTube by DigitalSreeni.[1]

Normalization The initial step was normalizing the voxel intensities to the range between 0 and 1 using the MinMaxScaler from scikit-learn. Normalizing the intensities was an important step for effective training since MRI machines have varying imaging standards, and voxel intensities can vary depending on the machine and settings used.

Cropping The images were cropped to focus on relevant brain areas (specific dimensions: [56:184, 56:184, 13:141]). The dimensions were chosen after observing samples from the dataset, to remove most of the background area in the images, and to remove the start and end slices of the brain to exclude irrelevant areas. This step also reduced the input size, helping with working within the hardware limitations.

Multi-channel Stacking We stacked different MRI modalities of the same scan to create a multi-channel input. We only used the T1ce, T2, and FLAIR modalities for stacking. The decision to exclude T1 modality was reasoned by reducing data redundancy and reducing memory usage, since we have found through research that T1 and T1ce provide similar information.

[1] `https://www.youtube.com/watch?v=0Rpbhfav7tE&list=PLZsOBAyNTZwYgF8O1bTdV-lBdN55wLHDr&pp=iAQB`

Data generator Due to the image sizes vastly exceeding our available VRAM when training, we had to add a data generator, which the Keras model could use to iteratively request images for each batch. The data generator takes the paths for each specified brain scan from the corresponding folder (training, validation, test) and performs all the preprocessing steps, everytime an image/label pair is yielded. I.e. it:

1. Extracts each modality and mask from the compressed .gz files.
2. Normalizes voxel intensities.
3. Crops the image.
4. Joins channels to NumPy array.

3.3.2 Class Imbalance and Loss Function

Class imbalance is a common issue in brain tumor segmentation tasks. BraTS dataset also suffered from the class imbalance problem. This problem is illustrated in Figure 3, the distribution of pixels across classes is significantly skewed. The ’NOT TUMOR’ class dominates, with an average of 98.80% of the pixels per brain image slice, while the ’CORE TUMOR’, ’EDEMA’, and ’ENHANCING TUMOR’ classes cover only 0.16%, 0.79%, and 0.25% respectively.

To address this problem, we conducted a few experiments with different loss functions. These experiments involved testing three distinct approaches: using only categorical cross-entropy, only Dice loss, and a combination of both with varying weight distributions.

Initially, the model was trained using only categorical cross-entropy. The performance was not optimal as it cannot handle class imbalance effectively. Later, the model was trained using only Dice loss. Dice loss is suited for dealing with imbalanced datasets, as it emphasizes the correct segmentation of classes by measuring the overlap between the predicted segmentation and the ground truth. However, we had problems related to model convergence. The final experiment was using a combined loss function with varying weights between Dice loss and categorical cross-entropy. The combinations tested were (0.5, 0.5), (0.25, 0.75), and (0.75, 0.25). The reason behind this approach was to leverage the advantages of both loss functions, Dice loss for its effectiveness in dealing with class imbalance and accounting for spatial overlap, and categorical cross-entropy for voxel-wise prediction accuracy.

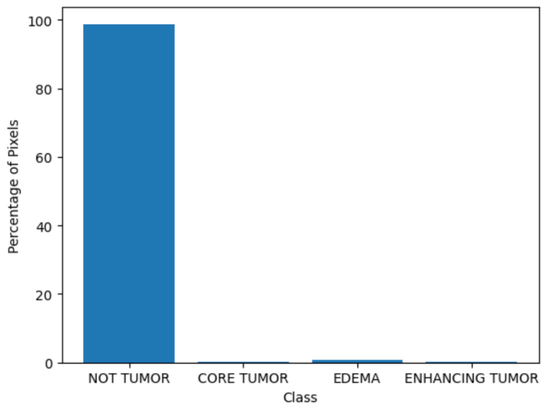


Figure 3: Average Percentage of Pixels per Class in an Image Slice

Implementation Details of Multi-label Dice Loss An updated version of Dice loss function was used to handle multi-label segmentation. For each label, the Dice coefficient is calculated using the formula:

$$\text{DiceCoef}(y_{\text{true}}, y_{\text{pred}}) = \frac{2 \times \sum(y_{\text{true}} \times y_{\text{pred}}) + \text{smooth}}{\sum y_{\text{true}} + \sum y_{\text{pred}} + \text{smooth}} \tag{1}$$

where y_{true} and y_{pred} represent the ground truth and predicted masks for a particular class, and smooth is a small constant to avoid division by zero errors. The Dice loss for each class is then calculated as $1 - \text{DiceCoef}$ for that class. These individual losses are averaged across all classes to derive the final multilabel Dice loss:

$$\text{DiceLoss}_{\text{multilabel}} = \frac{1}{N} \sum_{i=1}^{N} (1 - \text{DiceCoef}(y_{\text{true}_i}, y_{\text{pred}_i})) \tag{2}$$

where N is the number of classes. This approach gives a balanced evaluation of the model’s performance across all classes.

The combined loss was implemented using a weighted sum of multi-label dice loss and categorical cross-entropy loss.

$$\text{L}_{\text{combined}} = \alpha \times \text{L}_{\text{Dice}} + (1 - \alpha) \times \text{L}_{\text{CE}} \tag{3}$$

3.3.3 Accuracy Metric

The Dice coefficient was used as the accuracy metric. This choice was made considering that it is the primary metric used to evaluate model performance for the BraTS Challenge. It gives a good measure of the model’s accuracy since it calculates the overlap between the predicted segmentation and the ground truth.

3.3.4 Optimizer

Adam was used as the optimizer combined with a learning rate scheduler (explained in 3.3.6). We chose Adam as it was more suited for a complex task like brain tumor segmentation compared to SGD.

3.3.5 Batch Normalization

As an attempt to improve training efficiency, we tried adding batch normalization into the U-Net architecture. However, this made the model impossible to train due to memory issues caused by the large size of training data. As a result, we decided not to use batch normalization.

3.3.6 Hyperparameters

Number of Filters In the U-Net architecture, there is a progressive increase in filters (32, 64, 128, 256, 512) as the network deepens. This enables the model to learn from basic features in the initial layers to more complex patterns in the deeper layers. We experimented with different filter sizes and observed that the model performed better as the number of filters increased, so we decided to use sizes (32, 64, 128, 256, 512) which was the largest possible within the memory limit.

Dropout Rates Dropout regularization technique helps prevent overfitting by randomly disabling neurons during training. The initial U-Net model had dropout rates between 0.1 and 0.2, however we decided to experiment with higher dropout rates since there were signs of overfitting. The dropout rates were adjusted to values between 0.3 and 0.5. This increase improved the model's accuracy, and better regularization provided by higher dropout rates helped with preventing overfitting.

Kernel Initializer The He normal initializer was used as the kernel initializer for the model's convolutional layers due to its compatibility with ReLU activations. He normal initializer draws weights from a Gaussian distribution with a standard deviation based on the number of input units. This choice improved the model's performance.

Learning Rate A learning rate scheduler dynamically adjusts the learning rate during training. We decided to use a learning rate scheduler as we thought the model can learn the detailed features of the data better by using a lower learning rate in the latter stages of training. This approach balances between fast initial learning and stable convergence during the latter stages of training, optimizing the overall training efficiency. We achieved the highest improvement in the model accuracy with the addition of learning rate scheduler. The final learning rate scheduler implementation used for the model involved a warm-up phase followed by a decay phase. This approach was chosen after decay method experiments. The learning rate was initially set to increase linearly during the warm-up phase.

$$\text{LR} = \text{TargetLR} \times \frac{\text{CurrentStep}}{\text{WarmupSteps}} \tag{4}$$

This gradual increase helps in stabilizing the training process at the start. A lower starting learning rate avoids problems which can occur when training weights are randomly initialized at the start. The warm-up phase ensures that the model doesn't converge to a sub-optimal solution at the beginning. After the warm-up, the learning rate enters decay phase. Two decay methods were tested: step decay and cosine annealing. Step decay decreases the learning rate at certain intervals and cosine annealing gradually decreases the learning rate following a cosine curve. The cosine annealing method was found to perform slightly better in our experiments.

$$\text{LR} = \text{TargetLR} \times \frac{1 + \cos\left(\pi \times \frac{\text{CurrentStep}-\text{WarmupSteps}}{\text{TotalSteps}-\text{WarmupSteps}}\right)}{2} \tag{5}$$

3.3.7 Training

Training was done remotely using Google Colab. This introduced certain limitations (See 5.1), which motivated the use of a small training set. We ended up using a training set consisting of 100 brain scans (each scan with 128 brain slices of size 128x128 for each of the 3 modalities), and another 20 for validation during training. The model was trained twice over 20 epochs (i.e. 40 epochs total), using a batch size of 2 and $\frac{TrainSize}{BatchSize} = \frac{100}{2} = 50$ steps per epoch.
The training set size, as well as number of epochs, is relatively small for similar models, and has mainly been motivated by hardware limitations. Futrega et al. [2021], Cirillo et al. [2021], Isensee et al. [2020] were all trained using all scans in the dataset for 1000 epochs, 200 epochs, and 1000 epochs respectively. A main concern with such a small training set, especially in deep learning, is overfitting, which was relatively manageable with the dropout regularization. Independent experimentation was done (See 3.3.8) to gauge overfitting in an even smaller training set (50 samples), and techniques to prevent it.

3.3.8 Data Augmentation

Data augmentation is the process of increasing model generalizability by artificially expanding the training dataset. This can be in e.g. size or diversity, the latter of which has been briefly explored in this project.

The same U-Net model was trained, using the initial categorical cross-entropy as loss, but with the multi-label dice coefficient as the accuracy metric. The model was trained separately for 79 epochs on both 50 and 100 scans. This was then repeated, with the addition of some brightness noise. The brightness noise was added in the data generator by applying a randomized gamma correction, resulting in a perceived brightness variation in the interval: $0.8 - 1.2$ relative to the original. The specific operation takes the original voxel intensity v and computes a new intensity $v' = \alpha \cdot v^{\gamma}$ where α and γ are chosen randomly in the interval: $[0.8, 1.2]$. We found that the increase from 50 to 100 training samples mostly eliminated the overfitting symptoms after 79 epochs, while the addition of the data augmentation only slightly improved on this. For the purpose of experimentation, and with our hardware limitations, the training size of 100 scans with an even lower epoch number and without data augmentation seemed adequate to prevent major overfitting. The choice of brightness augmentation was motivated by Cirillo et al. [2021], who found it to be the second best performing augmentation, as well as the fact that additional augmentations does not provide further improvements.

4 Results

No comprehensive evaluation was carried out, as GPU time was prioritized for experimentation, and the validation metrics were adequate for that purpose. We are thus unable to meaningfully compare performance to official BraTS participants for two main reasons: 1. Hardware limitations. 2. The BraTS competition uses a private test dataset for official evaluation, which is a superset of the public test-set.

The base model (see Figure 4) showed a final validation dice score of 43.17% (peak: 43.99%), and validation loss of: 12.62% (peak: 10.53%). The majority of the plots shows similar performance between the training and validation metrics, indicating strong model generalization. A slight divergence can be noticed towards the end in both plots, suggesting some early signs of overfitting.

The model using combined loss (See Figure 5) showed a slight improvement with a final validation dice score of 48.49% (was also peak). We also see a higher loss because of the new combined loss function, peaking in the last epoch at 33.75%. A significant spike can be seen in the loss at epoch 17, which we don't have a concrete reason for. However, it indicates a failure to generalize the model to unseen data in that particular epoch, suggesting the start of overfitting.

The model with combined loss and a learning rate scheduler (see Figure 6) achieved the highest final validation dice score of 62.66% (peak: 63.08%), with a loss of 26.61% (peak: 25.16%). A similar spike can be seen in both the validation loss and validation dice score. This model also showed slight divergence between the training and validation dice scores, with the final training dice being 77.27%, and the validation dice showing a minor decrease in the last five epochs. This also indicates the start of overfitting.

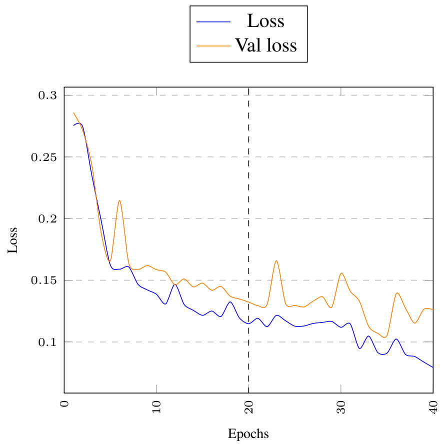
For a brief comparison to our scores, the official 2020 BraTS winner was nnU-netIsensee et al. [2020], who achieved an average dice score of 85.34 (Average over the 3 official labels scores which was: 88.95, 85.06 and 82.03 for whole tumor, tumor core and enhancing tumor respectively).

The data augmentation experiments (see Table 1) showed that a model trained on 50 images, although having a dice score (0.82), generalizes badly to unseen images (val dice: 0.47, loss: 0.29). The addition of the simple brightness data augmentation technique decreased the final generalization gap, improving the validation score slightly.

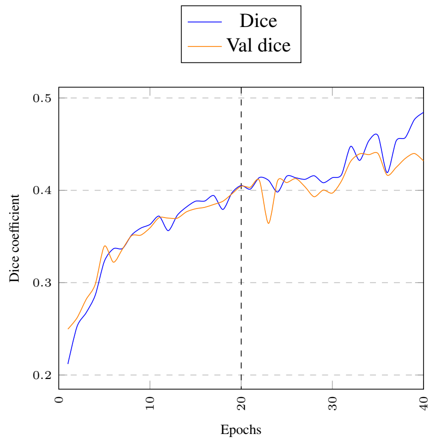
			Train		*Val*	
Training data size	*Model*	*Data augmentation*	*Loss*	*Dice*	*Loss*	*Dice*
50	1	None	0.0184	0.8208	0.2956	0.4743
	2	Brightness 0.8-1.2	0.0423	0.6898	0.2349	0.4961
100	3	None	0.1031	0.4476	0.1534	0.4164
	4	Brightness 0.8-1.2	0.0795	0.4637	0.1081	0.4453

Table 1: Data augmentation experiment

When looking at the case with 100 training samples, we see an insignificant generalization gap, even without data augmentation. The brightness technique slightly improves on this, resulting in a validation dice score of 0.44. This generally shows the importance of a larger dataset, relative to specific regularization techniques.

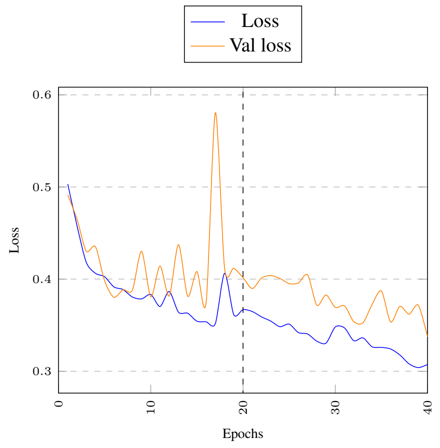


(a) Loss

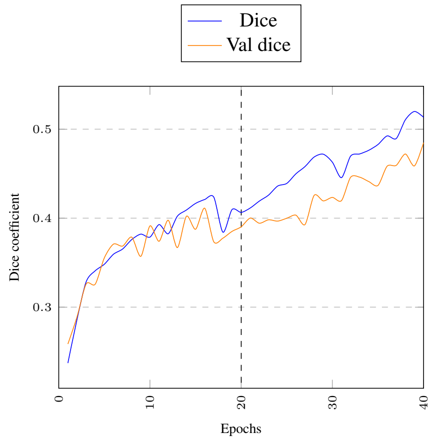


(b) Dice coefficient

Figure 4: Base Model

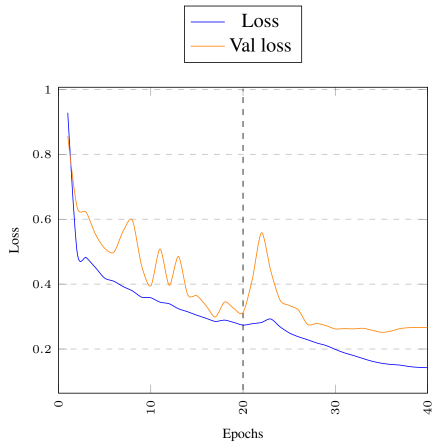


(a) Loss

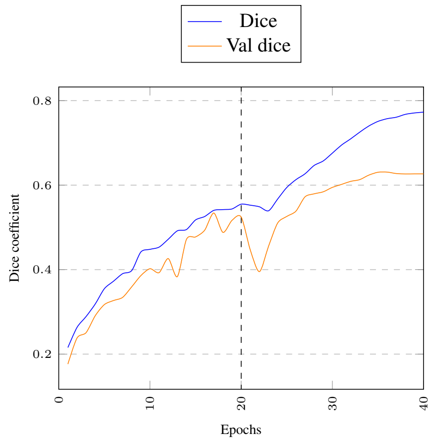


(b) Dice coefficient

Figure 5: Model with Combined loss



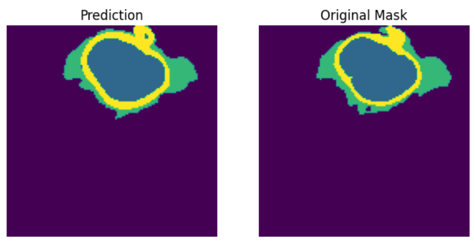
(a) Loss



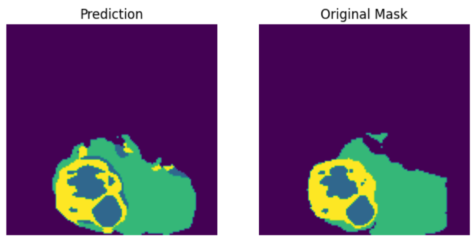
(b) Dice coefficient

Figure 6: Model with Combined Loss and Learning Rate Scheduler

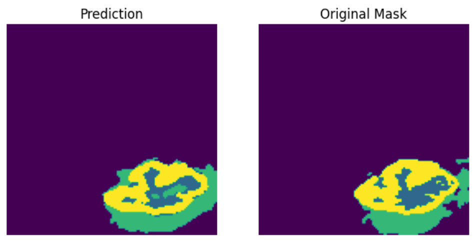
Some predictions of the final model, where scans from a separate test data folder was used, can be seen in Figure 7.



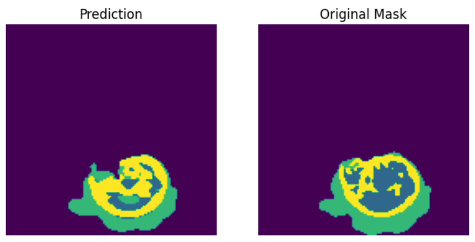
(a) Prediction 1



(b) Prediction 2



(c) Prediction 3



(d) Prediction 4

Figure 7: Predictions by the Final Model on Test Data

5 Discussion

In the end, our final model was able to segment brain tumors in 3D MRI brain scans with a dice score of 62.66% (validation score). This is still far from the top performing models, and would probably be lower if evaluated on the much larger official test data set. However, with our limitations (see 5.1), the focus of our project was more of a general exploration of techniques in the domain. The explorative approach did show how much the independent techniques can improve a model significantly. Especially the learning rate scheduler showed great improvements, showcasing the importance of general optimizations e.g. before one starts to redesign the entire architecture Regardless of the model metrics, the final predictions (See Figure 7) did seem very close to the original masks, highlighting the sensitivity of the dice coefficient.

The data augmentation experiment could have been done in a way that was more comparable to our main model. It could be interesting to look more into optimizing the relationship between training data size, number of epochs and regularization techniques. Hypothetically, the randomized gamma correction could also help increase the noise robustness, which might in fact be relevant in the BraTS field, as the data is supplied from many different MRI machines.

The training plots occasionally showed significant spikes in the validation loss and validation dice score. The root cause would have been interesting to explore, and additional logging during training might have helped. They seemed to cause setbacks in the training progress, which means that fixing them might have improved the final model performance.

5.1 Limitations

The main limitation was the available computing power in Google Colab, where we conducted our experiments. Colab has undisclosed GPU usage limits, causing some experiments to get interrupted. We solved this issue partially by periodically saving the trained model as a .keras file and reloading it in when we had GPU power again. The batch size was severely limited by the available VRAM, due to the large brain scan sizes (each channel=$\sim$18MB, mask=$\sim$9MB), which further slowed training. The data generator (See 3.3.7) made the training feasible in Colab, but only with a small batch size of 2 (which still uses $\sim$95% VRAM). It was also necessary to explicitly garbage collect and clear the Keras global state after each generated batch.

One could have used a standard 2D U-Net instead of the 3D variation. Using the 3D U-Net results in the model being forced to train on 128 depth slices. Using the 2D version would let the model use individual slices and potentially require less computing power to reach the same accuracy. This approach was never explored, but definitely could be a factor to consider with limited GPU power.

References

Fabian Isensee, Paul F. Jaeger, Peter M. Full, Philipp Vollmuth, and Klaus H. Maier-Hein. nnu-net for brain tumor segmentation, 2020.

Fabian Isensee, Philipp Kickingereder, Wolfgang Wick, Martin Bendszus, and Klaus H. Maier-Hein. No new-net. *CoRR*, abs/1809.10483, 2018. URL `http://arxiv.org/abs/1809.10483`.

Jason Kugelman, Joseph Allman, Scott A. Read, Stephen J. Vincent, Janelle Tong, Michael Kalloniatis, Fred K. Chen, Michael J. Collins, and David Alonso-Caneiro. A comparison of deep learning u-net architectures for posterior segment oct retinal layer segmentation. *Scientific Reports*, 12(1):14888, Sep 2022. ISSN 2045-2322. doi:10.1038/s41598-022-18646-2. URL `https://doi.org/10.1038/s41598-022-18646-2`.

Chenhong Zhou, Changxing Ding, Xinchao Wang, Zhentai Lu, and Dacheng Tao. One-pass multi-task networks with cross-task guided attention for brain tumor segmentation. 2019.

Michal Futrega, Alexandre Milesi, Michal Marcinkiewicz, and Pablo Ribalta. Optimized u-net for brain tumor segmentation. 2021.

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

Brats2021 dataset and challenge description. URL `http://braintumorsegmentation.org/`.

Marco Domenico Cirillo, David Abramian, and Anders Eklund. What is the best data augmentation for 3d brain tumor segmentation?, 2021.