Slovak University of Technology in Bratislava
Faculty of informatics and information technologies

Reg. No.: FIIT-101018-97014

Jakub Povinec

Medical Image Data Processing Using Methods of Computer Vision and Deep Neural Networks

Bachelor's thesis

Thesis supervisor: prof. Ing. Vanda Benešová, PhD.

May 2022

Slovak University of Technology in Bratislava

Faculty of informatics and information technologies

Reg. No.: FIIT-101018-97014

Jakub Povinec

Medical Image Data Processing Using Methods of Computer Vision and Deep Neural Networks

Bachelor's thesis

Study programme: Information Security

Study field: Computer Science

Training workplace: Institute of Computer Engineering and Applied Informatics, FIIT STU Bratislava

Thesis supervisor: prof. Ing. Vanda Benešová, PhD.

May 2022

Slovak University of Technology in Bratislava
Ústav počítačového inžinierstva a aplikovanej informatiky

Faculty of Informatics and Information Technologies
Academic year: 2021/2022
Reg. No.: FIIT-101018-97014

STU FIIT

BACHELOR THESIS TOPIC

Student:	**Jakub Povinec**
Student's ID:	97014
Study programme:	Information Security (conversion programme with a foundation year)
Study field:	Computer Science
Thesis supervisor:	doc. Ing. Vanda Benešová, PhD.
Head of department:	Ing. Katarína Jelemenská, PhD.

Topic: **Spracovanie obrazových medicínskych dát metódami počítačového videnia a hlbokých neurónových sietí**

Language of thesis: English

Specification of Assignment:

Pri spracovaní diagnostických dát v medicíne, ako sú napríklad dáta z počítačovej tomografie alebo magnetickej rezonancie, nachádzajú metódy počítačového videnia stále významnejšie uplatnenie. Významnými problémovými oblasťami sú hlavne detekcia a segmentácia anomálií a registrácia multimodálnych dát. Analyzujte súčasný stav problematiky využitia počítačového videnia pri spracovaní vizuálnych dát v medicíne. Zamerajte sa predovšetkým na metódy využívajúce hlboké neurónové siete. Navrhnite metódu na podporu diagnostiky s využitím moderných metód umelej inteligencie. Navrhnutú metódu realizujte softvérovým prototypom s využitím knižníc a rámcov vhodných na spracovanie medicínskych dát. Riešenie overte experimentom s reálnymi dátami z medicínskych modalít. Vyhodnoťte presnosť, robustnosť a časovú efektívnosť spracovania. Výsledky porovnajte s inými publikovanými riešeniami.

Length of thesis: 40

Deadline for submission of Bachelor thesis:	16. 05. 2022
Approval of assignment of Bachelor thesis:	23. 11. 2021
Assignment of Bachelor thesis approved by:	doc. Dr. Ing. Michal Ries – Study programme supervisor

Declaration of honor

I, Jakub Povinec, honestly declare that this thesis is my independent work under the supervision of prof. Ing. Vanda Benešová, PhD., except where I acknowledged the work of other people in cited literature.

In Bratislava, 16.5.2022 .....

Jakub Povinec

Special thanks

I would like to thank my supervisor Professor Vanda Benešová for guidance and help during the work on my Bachelor's thesis. I would also like to thank Siemens Healthineers for sponsoring the Azure Machine Learning services, which we used to train and evaluate our models.

Anotácia

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Študijný program: Informačná bezpečnosť

Autor: Jakub Povinec
Bakalárska práca: Spracovanie obrazových medicínskych dát metódami počítačového videnia a hlbokých neurónových sietí
Vedúci projektu: prof. Ing. Vanda Benešová, PhD.
May 2022

Segmentácia mozgových nádorov hrá dôležitú úlohu v diagnostike a plánovaní liečby onkologických pacientov, ale ich manuálna segmentácia predstavuje stále pomerne náročnú úlohu. V tomto ponímaní existuje potreba spoľahlivej a plne automatickej metódy určenej prave na ich segmentáciu. V práci navrhujeme dve rôzne metódy na segmentáciu gliómov. Obe tieto metódy sú založené na U-Net architektúre a boli trénované a vyhodnotené na dátach z BraTS 2021.

V prvej metóde sme segmentovali všetky podoblasti naraz a experimentovali s rôznymi premennými, ako sú rýchlosti učenia, augmentácie a použité sekvencie, aby sme dosiahli čo najlepšie výsledky. V druhom prístupe sme sa sústedili na problém nedostatočne zastúpených tried a trénovali sme model pre každú triedu samostatne. Jednotlivé segmentácie sme následne spojili a tak sme vytvorili finálnu segmentáciu.

Nakoniec obe naše metódy pomerne dobre segmentovali nádorové podoblasti. Druhá metóda dosiahla lepšie výsledky pre všetky časti gliomov a bola úspešnejšia ako prvá, ale vyžadovala si viac času na trénovanie.

Annotation

Slovak University of Technology in Bratislava
Faculty of informatics and information technologies
Degree Course: Information Security

Author: Jakub Povinec
Bachelor's thesis: Medical Image Data Processing Using Methods of Computer Vision and Deep Neural Networks
Supervisor: prof. Ing. Vanda Benešová, PhD.
May 2022

Tumour segmentation plays an integral role in the diagnosis and treatment planning of oncological patients. However, their manual segmentation is still a challenging task, and there is a need for a reliable and fully automatic method designed for this problem. In this work, we propose two different methods for segmenting gliomas and their subregions. Both of these methods are based on the U-Net architecture and were trained and evaluated on the BraTS 2021 dataset.

In the first method, we segmented all subregions at once and experimented with different variables such as learning rates, augmentations and used sequences to get the best results. In the second approach, we tried to mitigate the problem of underrepresented classes by training a model for each class independently. The individual label maps were subsequently combined to create a final segmentation.

In the end, both methods could sufficiently segment the tumour subregions. The second method achieved better results for all subregions and was more successful than the first but required more time for training.

Table of contents

List of Figures

List of Tables

List of used abbreviations

GPU	Graphics Processing Unit
CPU	Central Processing Unit
MRI	Magnetic Resonance Imaging
CT	Computer Tomography
PET	Positron Emission Tomography
BraTS	Brain Tumour Segmentation challenge
T1	T1-weighted (MRI sequence)
T1ce	post-contrast T1-weighted (MRI sequence)
T2	T2-weighted (MRI sequence)
FLAIR	T2 Fluid Attenuated Inversion Recover (MRI sequence)
NCR	Necrotic Tumour Core
ED	Peritumoral Edematous/Invaded Tissue
ET	Gd-Enhancing Tumour
WT	Whole Tumour (ED + ET + NCR)
TC	Tumour core (ET + NCR)
CNN	Convolutional Neural Networks
ANN	Artificial Neural Networks
ML	Machine Learning
ReLU	Rectified Linear Unit
DSC	Dice Score Coefficient
HD95	95th percentile Hausdorff distance
lr	Learning Rate
aug	augmentations

Chapter 1

Introduction

In recent years, machine learning and particularly deep learning have seen some dramatic progress in the field of computer vision. This can be attributed to the progress in technology, with the development of more effective and relatively inexpensive computing units, such as graphics processing units (GPUs) and central processing units (CPUs). The progress of artificial neural networks, in particular, the development of new methods such as Convolutional Neural Networks (CNN) had a huge impact on the further improvement of image processing and became an integral part of the processing of diagnostic medical data, such as Magnetic Resonance Imaging (MRI), Computer Tomography (CT) or Positron Emission Tomography (PET) [1]. Convolutional neural networks can be applied for tasks such as segmentation of malignant tissue or different tumour subregions [2]. In this work, we focus primarily on the segmentation of gliomas in multiparametric magnetic resonance images.

Magnetic resonance imaging is the modality of choice when considering neurological images due to its soft tissue contrast [3]. Brain tumour segmentation from magnetic resonance images is an indispensable part of patient diagnosis and sub-

sequent treatment planning, which is mainly used to determine the location of the tumour, the extent of the involved tissue, and the volume or mass of the tumour [4]. Multiple sequences or magnetic resonance protocols are used to precisely assess the patient's brain or the different subregions of a specific tumour. Each of these sequences represents a different method used to collect the data from magnetic resonance. These sequences have different properties such as contrast or speed of acquisition, and each highlights different tissue or part of the tumour [5]. Examples of commonly used sequences to assess brain tumours are:

- T1-weighted (T1) - highlights blood, and is also useful to distinguish white and grey matter
- post-contrast T1-weighted (T1ce) - highlights the breakdown of the blood–brain barrier, bleeding
- T2-weighted (T2) - highlights the borderline of the tumour and edema
- T2 Fluid Attenuated Inversion Recovery (FLAIR) - highlights peritumoral edema, nonenhancing tumour, white matter injury, gliosis

The differences between described sequences can be seen in the figure (Fig. 1.1).

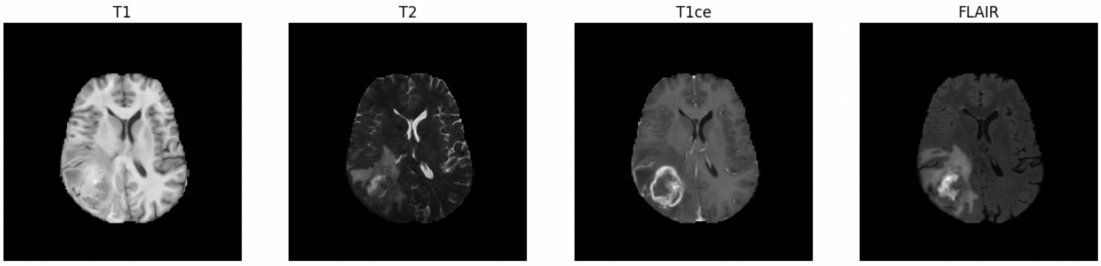


Figure 1.1: Visualisation of T1, T2, T1ce and FLAIR MRI sequences.

1.1 Motivation

Gliomas are a form of brain tumour that is considered to originate from glial or other types of cells in the central neural system [6]. They represent the most common type of malignant brain tumours among the adult population, where almost 80% of malignant brain tumours are gliomas, and they account for almost 50% of all tumours in children and adolescents aged 0-19 years [7]. Compared to benign tumours, malignant brain tumours grow fast and can spread to other areas of the brain and spine. Therefore it is even more important to find the appropriate treatment early.

Due to tumour heterogeneity, segmentation and detection of subregions are crucial in planning the correct treatment. Brain magnetic resonance imaging is the most important modality in this setting. On the other hand, while being the most profound modality, segmentation of gliomas on magnetic resonance images represents a difficult task in medical image processing due to their heterogeneity. Manual segmentation of gliomas is time consuming, tedious, and error prone, which can lead to incorrect patient treatment [8]. In this regard, there is a need for a reliable and fully automatic method designed explicitly for glioma segmentation.

Chapter 2

Analysis

The goal of machine learning is to essentially give computers the ability to solve problems of different tasks by learning from experiences (input data). This goal can be achieved by creating so-called models. These models are trained on input data to produce outputs based on features extracted from the given data [9]. The output is specific to the task, where, for example, in classification, the output can be a set of classes predicted by the model or in segmentation, the output would typically be a selected partition or set of partitions from the given input data (e.g. an image) based on the given mask.

Models gain said experiences during a training phase, which consists of optimising different parameters of the model in such way that its performance improves with every iteration [8]. For example, the model predicts different classes of input data with higher accuracy.

The main goal when creating these models is to make them generally applicable and to deliver correct predictions not only for the data used during the training phase but also for new, unseen data. This goal can be achieved by either using a

broad range of data in the training set or by using different types of augmentations. Data augmentations can be especially useful when dealing with, for example, a reasonably small or homogeneous dataset. This problem, when a model indicates high accuracy during training but shows poor results in testing when used on unseen data, is called overfitting, and it is a fairly common problem that occurs during training of machine learning models [10]. Another solution to tackle this problem can be by modifying the training process with methods such as early stopping, or dropout [11].

After the training phase, the final model should be evaluated on a different data (test dataset) to simulate how the model would perform when used on new, unseen data and test the created model's generality.

2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are a type of machine learning that draws inspiration from biological nervous systems. They consist of an extensive collection of interconnected neurons [12]. These neurons are organised in layers, where the output of a neuron from the previous layer can become an input for the neurons in subsequent layers. The adjacent layers where all neurons from these two layers are connected are called fully connected layers [13]. An example of simple network architecture with fully connected layers can be seen in the figure (Fig. 2.1).

The main layers that make up the ANN architecture are the input, output, and hidden layers. The input data from the first layer (input layer) traverses through hidden layers and ends up in the output layer. Suppose we are using a neural network for image classification; the input would be the image we want to classify (more specifically, a matrix of pixels). These pixels would then go through a series of hidden layers. The hidden layers extract increasingly abstract features from

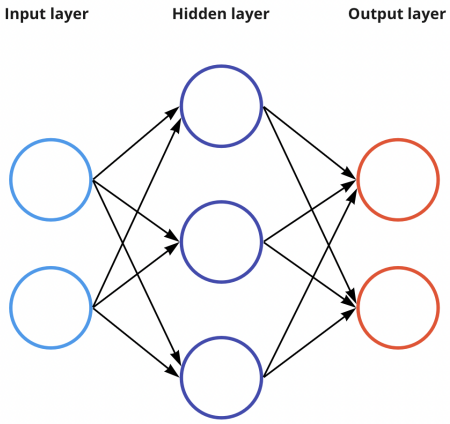


Figure 2.1: Simple artificial neural network architecture with an input layer with two neurons and two fully-connected layers (a hidden layer with three neurons and an output layer with two neurons).

the image (e.g., the first hidden layer would identify edges, and the subsequent layer would then, based on the identified edges, detect the contours of the object) [13]. Based on the output of the last hidden layer, the output of neurons in the output layer represents the probability of the object being the specified class (in classification, the number of neurons in the output layer is equal to the number of classes we are trying to predict).

ANN usually have multiple hidden layers where with the increasing number of these layers and the increasing number of individual neurons, the accuracy of the output also increases for the cost of higher complexity of the neural network.

An artificial neural network architecture with a large number of hidden layers is often described by the term deep learning [12]. The essential advantage of deep learning neural networks is their ability to be flexible and generic (e.g. if we classify multiple animals, some aspects such as the background or the animal's pose are not as relevant). The genericity of deep learning networks is achieved by their ability to learn important features by decomposing the input into a group of

abstract representations that were computed from the simpler, less abstract ones [14]. Deep learning is particularly useful for solving computer vision or speech and audio recognition tasks.

2.1.1 Neuron

The main building block of neural networks are neurons. These neurons are, simply put, computational nodes that are functionally similar to biological neurons [15]. A simple model of a singular neuron is shown in the figure (Fig. 2.2).

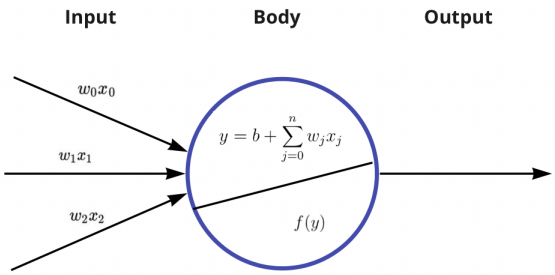


Figure 2.2: Model of a singular neuron from hidden layer.

Each neuron has several inputs that consist of the input features (x) multiplied by the weights (w). Weights represent the strength of influence between one neuron and another and are learned (changed) during training. Weights can have negative (inhibitory) or positive (excitatory) values [16]. The neuron's body calculates a sum of inputs and adds another value called bias (b), which is also a learnable parameter that changes during training. The body of the neuron can be interpreted by the following equation:

$$y = b + \sum_{j=0}^{n} w_j x_j \tag{2.1}$$

The computed sum (y) is then passed into a specified activation function (f) that

calculates the neuron's output. If the output exceeds the specified threshold value, the neuron will be activated, and the calculated output will be passed to another neuron [17].

2.1.2 Activation functions

Activation functions are fundamentally non-linear mathematical functions used to determine the output values of hidden layers. The non-linearity of these functions is an important characteristic. It allows the network to solve non-linear problems, such as image recognition, where it is necessary to distinguish between irrelevant variations of the input (e.g. different objects in the same position or on a similar background) [14].

One of the commonly used activation functions in the hidden layers of ANNs is the rectified linear unit (ReLU), described by the following equation:

$$f(y) = max(0, y) \tag{2.2}$$

One drawback of ReLU is that because of the characteristic of the function, the output values of many neurons in the network can become 0, which means that those neurons will not be activated and consequently will not contribute to the learning process [18]. There are two variations of ReLU, leaky ReLU and parametric ReLU (PReLU), that focus on fixing this problem. Instead of setting the output of the function to 0, for values smaller than 0, these functions return a small positive value [19].

Among other commonly used activation functions are tanh, maxout or sigmoid and softmax [14]. Sigmoid and softmax activation functions in particular, are typically used in the output layer of neural networks to generate probability maps for the given input. Sigmoid, described by the equation (2.3), returns real numbers in

the range between 0 and 1 and is typically present in the output layer of ANNs used in binary segmentation tasks. A problem that occurs when using the sigmoid activation function is that the function becomes insensitive to small changes when its arguments are either very positive or very negative [13].

$$f(y) = \frac{1}{1 + e^{-y}} \tag{2.3}$$

The softmax activation function is, on the other hand, commonly used in multilabel segmentation. The softmax activation function can be described by the following equation:

$$f(y)_i = \frac{e_i^y}{\sum_{j=1}^{n} e^{y_j}} \tag{2.4}$$

Similarly to the sigmoid, the softmax activation function also yields values in the range between 0 and 1.

2.1.3 Network training

During training, the network learns to separate the data and extract important features. The process of learning is represented by finding the correct values for individual weights and biases (those that minimise the value of Loss functions) used in neurons. The network uses an algorithm called backpropagation and gradient descent in order to find these values (learn) [20].

The backpropagation can be roughly described in 3 steps. First, we calculate the output for the specified input. Then we calculate the loss function (sometimes also called cost function) to find out how far away the computed values were from the desired output and compute a gradient vector for each weight. At this step, we are moving from the final output back to the original (input) values [13]. The gradient vector describes how the loss changes if we increase the weight [14]. Lastly,

we update the weights and biases accordingly. Equation (2.5) describes how the network updates its weights, where η represents the learning rate, ∇C represents the gradient vector and w represents the weight parameter [20]. These steps are repeated until we reach the minimal loss.

$$w_{new} = w - \eta \nabla C \tag{2.5}$$

The learning rate (η) is an important parameter that must have the correct value for network training to be effective. It is crucial to choose the value to be small enough, but at the same time, it should not be too small, otherwise the training would be very slow [20].

2.1.4 Loss functions

Loss functions are used to assess the output of a neuron. Among the popular loss functions used in deep learning are cross entropy (binary or categorical) or hinge loss [21]. These loss functions are mostly used for classification tasks.

In recent years, the dice loss function has become very popular for segmentation tasks, especially the segmentation of medical image data [22]. This loss function was first proposed by Milletari et al. [23] and its primary advantage is based on the fact that in medical data, the segmented region usually occupies only a small volume of the image and therefore, the network can be biased towards the background. The dice loss function is based on the dice score coefficient (DSC) commonly used as a metric to assess the performance of the trained model used for segmentation. The dice coefficient measures the overlap of the ground truth with the segmented region, and the dice loss is accordingly defined as a minimisation

of this overlap. It can be described by the following equation [24]:

$$Loss = 1 - DSC \tag{2.6}$$

Where the DSC can be described as follows:

$$DSC(A, B) = \frac{2(A \cap B)}{|A| + |B|} \tag{2.7}$$

Where A represents the computed segmentation and B ground truth.

Recently, it has become popular to combine multiple loss functions to get better results. One of the more common combinations is the combination of Dice and weighted cross entropy loss functions used for highly imbalanced segmentation tasks [25, 26].

2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep learning architecture that is more suitable for processing data in the form of a multidimensional matrix, such as images (2D data) or medical scans (volumetric data), therefore making them extremely effective in computer vision tasks such as classification or segmentation [13].

More specifically, image data are represented by a multidimensional array of shape (width, height, number of colour channels) that contains pixel values for each of these channels (e.g. for an RGB image of size 32x32, the input matrix would have shape (32, 32, 3)). Volumetric image data usually have only one channel, but they have another dimension, depth (e.g., the number of slices in a brain MRI scan).

The most significant advantage of CNNs is the fact that they need much fewer parameters for computer vision tasks compared to the regular ANN architecture [27]. A regular ANN takes input in the form of a vector, where if we had an RGB image of size 260x260, we would need to flatten this image into an input vector of size 260x260x3, which would correspond to 202 800 weights for each fully connected neuron in the first hidden layer.

CNNs improve upon regular ANNs by preserving the shape of the input, which can help identify important features while being sparse at the same time, and also reuse parameters, such as weights, in the successive layers, which saves memory and computational time [28]. CNNs being sparse means that individual neurons do not interact with (take input from) every other neuron from the previous layer but only from a small region specified by a filter, also known as a kernel. This characteristic is based on the fact that to detect important features such as edges, we do not need to look at the whole input (image), but only at its small regions [13].

2.2.1 CNNs architecture

The typical CNN architecture consists of three types of layers: convolutional layers with activation functions (typically ReLU), pooling layers, and fully connected layers (same as in regular ANN) that are used for flatting the data into a vector that is afterwards used as an input for neurons in the output layer for computing the final output (e.g. class scores in classification) [14]. A sample CNN architecture can be seen in the figure (Fig. 2.3).

The way CNNs work is that they extract meaningful features from the previous layers by using various filters in the convolutional layer while reducing the dimensions of the created feature maps with a downsampling operation in the pooling

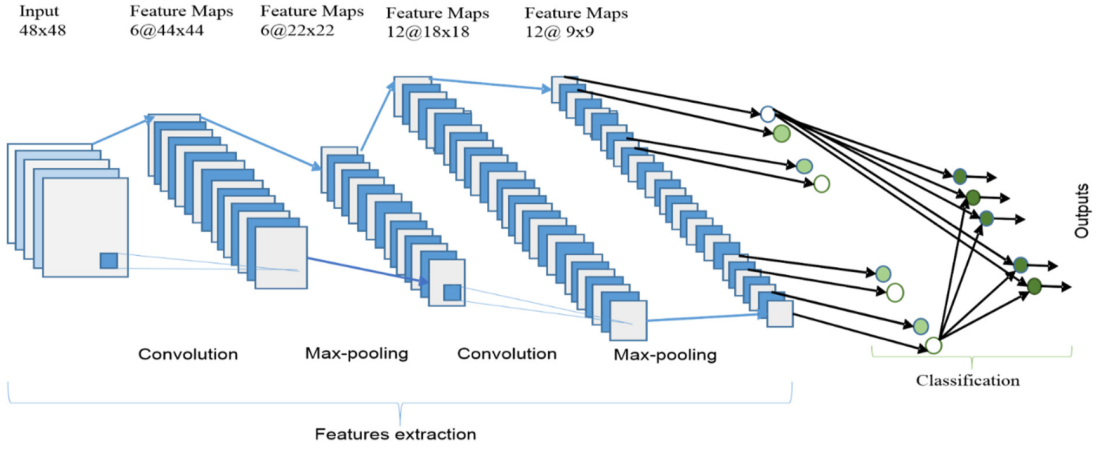


Figure 2.3: A sample CNN architecture that includes an input layer, multiple alternating convolutional and pooling layers, one fully-connected layer, and one classification layer [29].

layer. The output of the last layer in feature extraction (either convolutional or pooling) is then used as an input for the classification part of the network that is similar to regular ANNs [30].

2.2.2 Convolutional layer

A convolutional layer is the essential part of CNNs. They are used to create feature maps from the input using multiple filters, also known as kernels. These kernels are represented as multidimensional arrays of values that are learned during the training to extract useful features from the input [13].

The way the feature map is created is basically by sliding the kernel over the input and computing a sum of products of the values in the kernel with the values in the input that overlap with the kernel [28]. A simple example of convolution on a 4x4 input is described in figure (Fig. 2.4) that was recreated based on [30].

Before returning the output from the convolutional layer, the computed feature

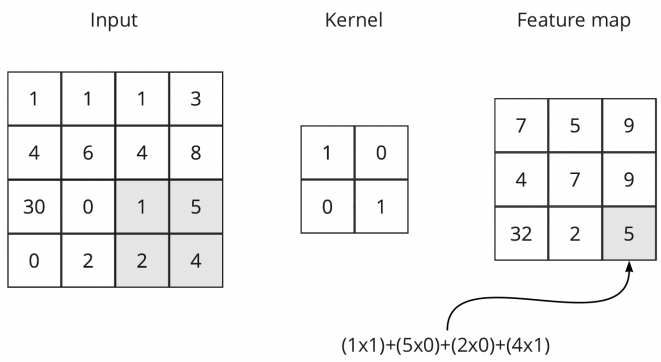


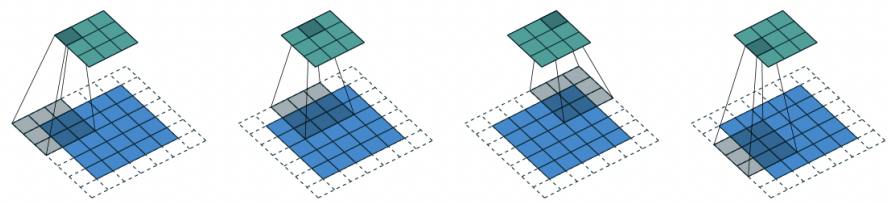
Figure 2.4: A simple example of convolution on a 4x4 input with a 2x2 kernel.

maps are sent through a non-linear activation function, such as ReLU and, after that, used as inputs for other layers [14].

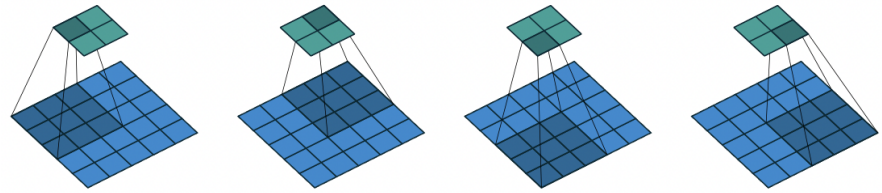
The kernels have multiple properties that can affect the size of the returned feature maps [28]:

- size (height x width), the kernels used in convolutional layers are usually much smaller than the input.
- stride, which defines the movement of the kernel (about how many pixels we move (slide) the kernel) over the input.
- zero padding, which defines how many 0 are added at the border of the input to make it bigger and essentially preserve the spatial resolution of the input after the convolution [13].
- number of filters that affect the depth (there is one feature map for every filter) of the output.

Figure (Fig. 2.5) shows the effect of padding on the size of the returned feature map. Here the blue square represents the input, a grey translucent square represents the kernel, and the green square represents the resultant feature map.



(a) Example of a 3x3 kernel used on input of size 5x5 with stride=2 and padding=1



(b) Example of a 3x3 kernel used on input of size 5x5 with stride=2 and padding=0

Figure 2.5: Effect of different zero-padding values on the size of the returned feature map [28].

2.2.3 Pooling layer

The pooling layer is another important part of the CNNs. It is used to reduce the size of the extracted feature maps by combining multiple surrounding pixel values into one, by either taking the maximum value (max pooling) or the average value (average pooling) within a rectangular surrounding [28]. Similarly to the convolutional layer, the pooling layer has several parameters: the size of the pooling window and the stride.

The pooling layer is used on every depth layer independently, thus preserving the depth of the input. A simple example of these two methods used in the pooling layer can be seen in figure (Fig. 2.6) that was recreated based on [30].

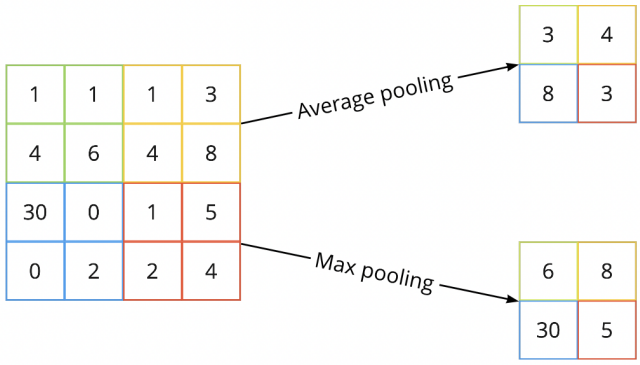


Figure 2.6: A simple example of a 2x2 max and average pooling operations used on a 4x4 input.

The pooling function reduces the size of a feature map, therefore improving the computational complexity of the network. Even though the feature map has been reduced in size, it keeps its information about the important features and also pooling makes the network "invariant to the transformations of the input" such as small rotations [13].

2.2.4 Upsampling

CNN architectures dedicated to segmentation also use in addition to downsampling layers upsampling layers. Downsampling layers are used to classify or identify the classes in the input (essentially identifying what is in the image), and upsampling layers are used to create the final segmented image (identifying the position of the classified objects in the image) that is the same size as the original image [31]. There are multiple methods of upsampling. One of those methods is max-unpooling, where we use stored indices of the elements used in max-pooling and then simply revert the process during upsampling [32].

2.3 U-Net

The U-Net architecture was proposed by Ronneberger et al. [33] in 2015 and has become much more popular since then. The architecture is primarily used for various segmentation problems, particularly the segmentation of medical images.

As discussed in the original paper, the main advantage of this architecture in comparison to the fully convolutional network proposed by Long et al. [34], which was also built for semantic segmentation, is that U-Net produces more precise segmentation, particularly of medical data. This improvement comes from a different approach in the upsampling path. The authors of U-Net designed the upsampling path to have a considerable amount of feature channels, thanks to which the network's higher-resolution layers can have more context [33]. Another advantage of the U-Net architecture is that the models require fewer images for successful training than other architectures.

2.3.1 Architecture

The architecture consists of a downsampling (contracting) path and a symmetrical upsampling (expansive) path that forms U shape-like architecture, therefore the name U-Net. The architecture of the original U-Net can be seen in the figure (Fig. 2.7).

Both the downsampling and upsampling paths consist of multiple convolutional layers with kernel size 3x3 and ReLU as activation function followed by either max-pooling operations of size 2x2 with stride 2 for the contracting path or up-convolution operation with size 2x2 for the expanding path. The max-pooling operations halve the size of the feature maps created by the convolutional layers, whereas the up-convolutions do the opposite. The final layer of the network uses a convolution with kernel size 1x1 and softmax activation function to produce the

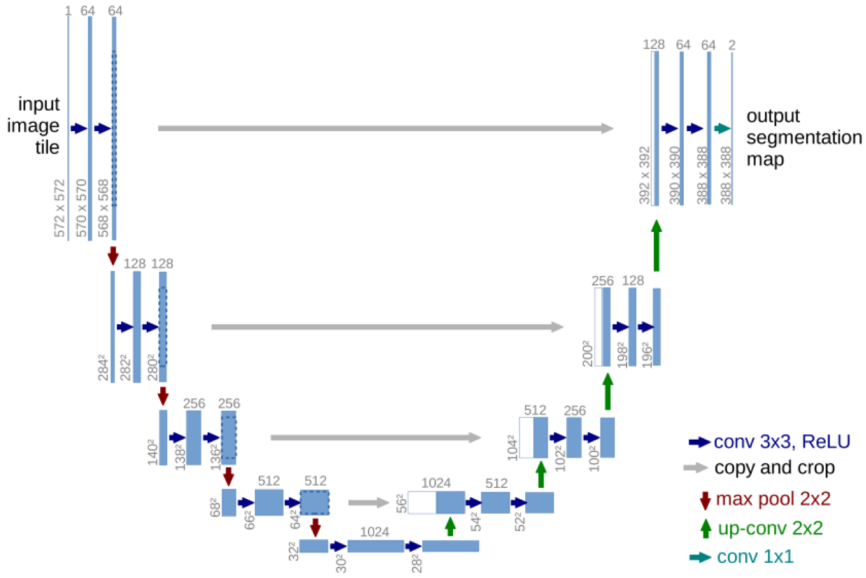


Figure 2.7: Architecture of the original U-Net [33]

desired output (probability map for each class).

An essential part of the expanding path is the concatenation of the last feature map from the opposite (contracting) path with the corresponding upsampled feature map from the expanding path. This operation creates so-called skip connections that provide more information for the network to localise the identified features more precisely [35].

The original U-Net uses 2D convolutional layers. However, there are modifications of this architecture that have used 3D convolutions for segmentation of volumetric data, such as MRI scans, and achieved impressive results [36, 37, 38, 25].

Chapter 3

Related work

In recent years, the most popular models used for medical image segmentation are based on the U-Net architecture. U-Net like architectures are consistently among the top submissions to the Brain Tumor Segmentation Challenge (BraTS)[1]. In this chapter, we will analyse multiple submissions for the recent BraTS challenges that consist of successfully segmenting individual parts of the glioma. The analysed works trained and evaluated their methods on the BraTS datasets and implemented some variations of the U-Net architecture. Each of the authors improved upon the original U-Net with numerous enhancements specific to brain tumour segmentation. Similarly, most of the works evaluated their methods with dice score coefficient (DSC) and the 95th percentile Hausdorff distance (HD95) for the individual tumour subregions (whole tumour, tumour core and enhancing tumour), typically used for medical data segmentation.

[1] `http://www.braintumorsegmentation.org`

3.1 Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks [39]

Dong et al. [39] proposed a modified 2D U-Net architecture for brain tumour detection and segmentation. The proposed method was evaluated on the BraTS 2015 dataset, consisting of 274 MRI brain scans. The scans from the dataset used for BraTS challenges come in four different MRI sequences (T1, T2, T1ce and FLAIR). The authors did not use all four sequences but only FLAIR and T1ce. The FLAIR sequence was used to segment the complete tumour, and the individual tumour subregions and the T1ce sequence were used to outline the enhancing tumour additionally.

To expand their training dataset, the authors used multiple data augmentation methods. They used simple transformations (shift, zoom, rotations) and elastic distortion.

The proposed architecture itself is very similar to the original U-Net architecture proposed by Ronneberger et al. [33]. The architecture can be seen in the figure (Fig. 3.1) The convolutional layers use ReLU as their activation function. The feature maps on the contracting path are downsampled by 2x2 max pooling. The last feature maps from the contracting path are concatenated to the corresponding upsampled feature maps in the expansive path in the same way as in the original U-Net. The main differences are in the size of the feature maps, which goes from 240x240 to 15x15, and the authors did not use zero padding. Another difference is in the deconvolutional operation. In the proposed architecture, the authors used deconvolutional operation with kernel size 3x3 as opposed to the kernels of size 2x2 in the original U-Net. In addition, to ensure the model's generality and prevent

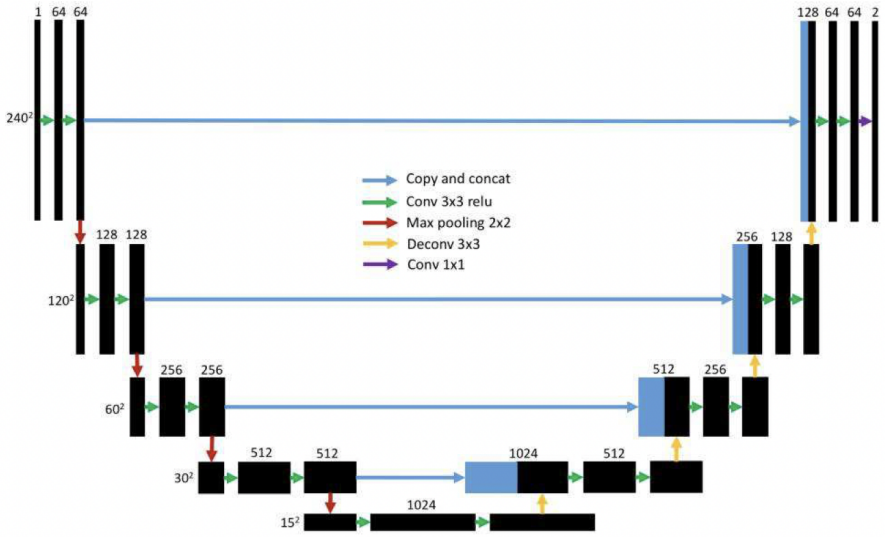


Figure 3.1: 2D U-Net architecture proposed by Dong et al. [39]

overfitting, the authors used dropout and L1 and L2 regularisation methods.

The proposed architecture was trained for 100 epochs with learning rate 0.0001 and the Adam optimiser. As a loss functions, the authors used soft dice instead of the commonly used cross-entropy.

The trained model was evaluated by computing the dice coefficient for each tumour subregion. The final dice coefficient for the complete tumour, tumour core and enhancing tumour can be seen in the figure (Fig 3.2).

			DSC		
Method	**Data**	**Grade**	**Complete**	**Core**	**Enhancing**
Proposed	Cross-Validation on BRATS 2015 Training Datasets	HGG	0.88	0.87	0.81
		LGG	0.84	0.85	0.00
		Combined	0.86	**0.86**	0.65

Figure 3.2: Evaluation by dice score coefficient of the 2D U-Net architecture proposed by Dong et al. [39] on the 2015 BraTS dataset.

Overall, the proposed solution achieved a dice score coefficient of 0.86 for the

complete tumour, 0.86 for the tumour core and 0.65 for the enhancing tumour. The model did not segment any enhancing subregions in the low-grade gliomas (LGGs), which can be expected, because they do not enhance by definition.

3.2 nnU-Net for Brain Tumor Segmentation [25]

Isensee et al. [25] proposed a U-Net like architecture designed for brain tumour segmentation that won the 2020 BraTS segmentation challenge. The proposed architecture was based on the nnU-Net, a "general-purpose deep learning architecture" which generally performs well in the segmentation of different types of medical data and which was also proposed by Isensee et al. [40]. The authors implemented numerous modifications to the nnU-Net architecture specific for brain segmentation. The main differences were in the training process and data augmentation that improved the performance and final results of the original nnU-Net. The authors also compared various combinations of the implemented enhancements to find the best overall implementation for the glioma segmentation.

The nnU-Net architecture is depicted in the figure (Fig. 3.3). The authors used

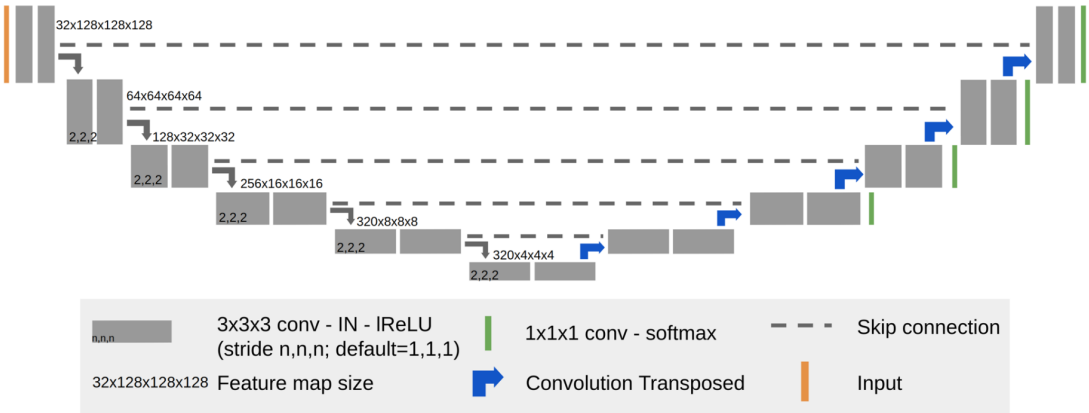


Figure 3.3: The nnU-Net architecture proposed by Isensee et al. [25]

strided convolutions for downsampling and transposed convolutions for upsampling. The architecture is intended for volumetric data, so the authors used 3D convolutional layers with kernel size 3x3x3 and Leaky ReLU as activation functions. They also rescaled the original scans to 128x128 and used 128 slices, so the initial input has a size of (128x128x128). The authors normalised the data by subtracting the mean of the voxels and dividing it by their standard deviation. The architecture also includes auxiliary segmentation outputs in the upsampling part, used for deep supervision, and consists of 1x1x1 convolution with a softmax activation function [25].

The main differences between the proposed method and nnU-Net are in the training process. While in the original nnU-Net, the authors used the sum of Dice loss and categorical cross entropy loss as loss function. In this paper, they exchanged the categorical cross entropy for binary cross entropy which is supposed to improve the network's performance by optimising for each region independently, based on what they replaced the softmax activation function with a sigmoid. The authors also modified the implementation of the dice loss, where instead of computing the dice loss for individual samples, they computed its value based on all samples in the batch. For training, they used a decaying learning rate with the initial value of 0.01, which remained unchanged in the proposed method, as well as the number of epochs which was set to 1000. The other differences were in the batch size and data augmentation. The batch size was raised from 2 to 5, which helped to improve the model's accuracy. The authors also raised the probability of individual augmentations (rotation, scaling) and implemented new methods (elastic deformation and brightness), resulting in better generality of the model.

The authors evaluated their architecture for multiple combinations of the mentioned modifications. The resulting table, in the figure (Fig. 3.4), describes the

dice coefficient scores and the 95th percentile of the Hausdorff Distance (HD95). The best overall combination (with the mean dice coefficient of 85.58) of these

Model	Dice				HD95			
	Whole	Core	Enh.	Mean	Whole	Core	Enh.	Mean
BL	90.6	84.26	77.67	84.18	4.89	5.91	35.10	15.30
BL*	90.93	83.7	76.64	83.76	4.23	6.01	41.06	17.10
BL*+R	90.96	83.76	77.65	84.13	4.41	8.80	29.82	14.34
BL*+R+DA	90.9	84.61	78.67	84.73	4.70	5.62	29.50	13.28
BL*+R+DA+BN	91.24	85.04	79.32	85.2	3.97	5.17	29.25	12.80
BL*+R+DA+BD	90.97	83.91	77.48	84.12	4.11	8.60	38.06	16.93
BL*+R+DA+BN+BD	91.15	84.19	79.99	85.11	3.72	7.97	26.28	12.66
BL*+R+DA*+BN	91.18	85.71	79.85	85.58	3.73	5.64	26.41	11.93
BL*+R+DA*+BN+BD	91.19	85.24	79.45	85.29	3.79	7.77	29.23	13.60

Figure 3.4: The results of the modified nnU-Net architecture proposed by Isensee et al. [25] on the BraTS 2020 validation dataset. The abbreviations in the model column describe individual modifications, where BL (baseline nnU-Net, * indicates the batch size of 5), R (region-based training), DA (more aggressive data augmentation, * indicates 0.3 probability for brightness augmentation), BD (model trained with batch Dice).

modifications was the baseline nnU-Net with a batch size of 5, region-based training, more aggressive data augmentation with a higher probability for the brightness augmentation and with the model trained by using their implementation of batch dice loss. The dice coefficient of this combination for the individual glioma subregions was: 91.18 for the whole tumour, 85.71 for the tumour core and 79.85 for the enhancing tumour.

3.3 Brain tumour segmentation with self-ensembled, deeply-supervised 3D U-net neural networks: a BraTS 2020 challenge solution [37].

Henry et al. [37] proposed a method for the Brats 2020 challenge, which ranked up among the top ten solutions in the challenge. Their method consists of training

multiple models with two different pipelines. They used a 3D U-Net architecture based on the work proposed by Çiçek et al. [36]. The models were trained separately, and the label maps produced by the models were later combined to ensure better generality. The architecture of their method can be seen in the figure (Fig. 3.5).

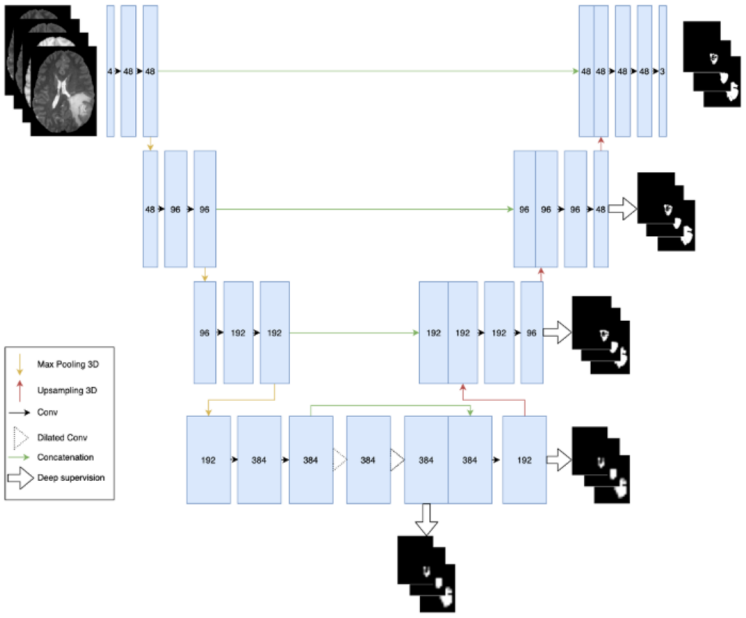


Figure 3.5: The 3D U-Net architecture proposed by Henry et al. [37]

Both the downsampling and upsampling part of the architecture consists of 4 stages. The downsampling path uses two convolutional layers with kernel size 3x3x3. The authors used a ReLU as an activation function. Each convolution was also followed by a normalisation layer, where one of the pipelines used a group normalisation and another instance normalisation. The network uses maxpooling with kernel size 2x2x2 and stride 2. The authors also included two dilated convolutions with kernel size 3x3x3 after the last convolutions that were afterwards concatenated to the feature maps produced by the last convolution [37].

The symmetrical upsampling part consists of upsampling performed by trilinear interpolation, concatenation with the outputs from the opposite stage in the downsampling path and the same convolutions used in the downsampling path. In order to produce the desired output, the final convolutional layer uses a kernel of size 1x1x1 and a sigmoid activation function. The last layer produced a three channel volume, where each channel represents the probability map for each tumour subregion [37].

The input volumes were normalised using a Min-Max scaler in the first pipeline and by z-score normalisation in the second pipeline. The input sequences were also cropped to the size 128x128x128, which removed most of the unnecessary background for both pipelines.

The pipelines also used different data augmentation to prevent overfitting and further promote the diversity of the models. The various augmentations were performed with different probabilities in the individual pipelines. The used augmentations included: rescaling (80% for the first pipeline and 20% for the second), input channel intensity shift that was only performed on the second pipeline, Gaussian noise performed on both pipelines, input channel dropping, performed only on the first pipeline and random flip along each spatial axis (80% for the first pipeline and 50% for the second pipeline) [37].

Both pipelines were trained by using Dice Loss with minor modifications. The model was trained for 380 epochs in the first pipeline with a decaying learning rate with a batch size of 1 and Ranger optimiser. The second pipeline was trained for 400 epochs with a batch size of 3 and Adam optimiser [37]. The initial learning rate for both pipelines was set to 0.0001.

The combination of the outputs provided by the models was based on their performance on the validation dataset. The models from the first pipeline performed

consistently better on the segmentation of necrotic and non-enhancing tumour and enhancing tumour, whereas the models from the second pipeline outperformed the first on the segmentation of edema.

Metric (mean)	ET	WT	TC
Dice	0.78507	0.88595	0.84273
Sensitivity	0.81308	0.91690	0.85934
Specificity	0.99967	0.99905	0.99964
Hausdorff (95%)	20.36071	6.66665	19.54915

Figure 3.6: The results of the method proposed by Henry et al. [37] on the BraTS 2020 testing dataset. Each column describes the individual scores for enhancing tumour (ET), whole tumour (WT) and tumour core (TC).

The authors evaluated their method on the Brats 2020 testing dataset. Their best model had the dice coefficient of 0.785 for the enhancing tumour, 0.886 for the whole tumour and 0.843 for the tumour core. The resulting table that describes the final can be seen in the figure (Fig. 3.6).

3.4 3D MRI brain tumour segmentation using autoencoder regularization [41]

Myromenko et al. proposed a method for segmenting gliomas from MRI images which achieved first place in the 2018 BraTS challenge. The proposed CNN architecture was based on the encoder-decoder approach with skip connections and an additional encoder branch. The additional branch was used to help regularise the network due to the small amount of images in the training set [41]. The proposed architecture is in the figure (Fig. 3.7).

The encoder path consists of three stages. At each stage, there are two blocks that are made of two convolutions with a kernel size 3x3x3, which double the number

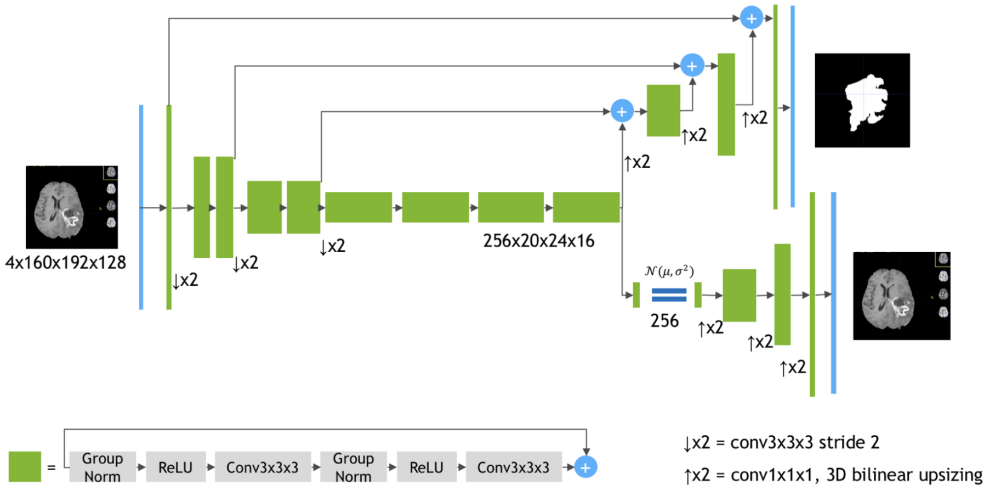


Figure 3.7: The encoder-decoder architecture proposed by Myromenko et al. [41]

of feature channels and ReLU activation function. Each block also incorporates a group normalisation. Each stage in the encoder path downsamples the input by 2. The authors used a strided convolutions as the downsampling operation [41].

The decoder path is smaller than the opposite encoder and uses only one block instead of two. The decoder blocks follow the same sequence of operations but use convolutions with a kernel size 1x1x1, which halves the number of feature channels. As opposed to the encoder, the decoder uses 3D bilinear upsampling to upsample the input by 2. Consequently, the final output of the encoder path has the same shape as the original input [41]. At the beginning of each stage in the decoder path, the output of the corresponding stage in the encoder path is added to the output of the previous upsampling operation. The last convolution has a kernel size 1x1x1 and uses a sigmoid activation function. The additional branch followed a similar structure as the decoder but without the skip connections.

The authors used all four sequences during training, a batch size of 1 and Adam optimiser with an initial learning rate of 0.0001. For the lost function, the authors

decided to use a combination of Dice and L2 losses. [41].

In the final, the authors trained ten models to improve the results further. The ensemble of ten models achieved a mean dice score of 0.8839 for the whole tumour, 0.7664 for the enhancing tumour and 0.8154 for the necrosis. The table with the final results is in the figure (Fig. 3.8).

	Dice			Hausdorff (mm)		
Testing dataset	ET	WT	TC	ET	WT	TC
Ensemble of 10 models	0.7664	0.8839	0.8154	3.7731	5.9044	4.8091

Figure 3.8: The final results of the BraTS 2018 testing dataset for the encoder-decoder architecture proposed by Myromenko et al. [41]

3.5 Conclusion

In this chapter, we analysed four methods used to segment individual parts of gliomas from MRI images. The analysed works evaluated their methods on the BraTS datasets and competed in the challenge. Most of the analysed works based their approaches on the U-Net architecture with only one method using a different encoder-decoder approach. Overall the analysed work with the highest final DSC is the U-Net based method proposed by Isensee et al. [25]. This method also won the 2020 BraTS challenge with a final DSC of 91.18 for the whole tumour, 85.71 for the tumour core and 79.85 for the enhancing tumour (Fig. 3.4). An interesting comparison is between the said method and the method proposed by Myromenko et al. [41], which won the 2018 BraTS challenge and achieved the final DSC of 0.8839 for the whole tumour, 0.7664 for the enhancing tumour and 0.8154 for the necrosis (Fig. 3.8), which is about 2 smaller across all subregions when compared to the method by Isensee et al. which was published 2 years later.

Chapter 4

Our work

The aim of our work is to design a deep learning method for segmenting the individual parts of gliomas from MRI images. For this purpose, we came up with and evaluated multiple approaches based on the 3D U-Net architecture. We opted for the BraTS 2021 challenge dataset [42, 43, 44] for training and evaluation mainly because of its quality and a large number of available scans.

4.1 Used software tools

For the implementation of our solution, we decided to use the Python[1] programming language with the machine learning framework TensorFlow[2]. We chose Python mainly because of the wide range of available libraries for data processing, mathematical operations, machine and deep learning, and its flexibility. The training and evaluation of our models were implemented by custom Python scripts that were run on the Azure Machine Learning[3] service.

[1] `https://www.python.org`
[2] `https://www.tensorflow.org`
[3] `https://azure.microsoft.com/en-us/services/machine-learning/`

4.2 Used hardware

All models were trained and evaluated on the designated GPUs available on the Azure Machine Learning service. We decided to use one NVIDIA Tesla P100[4] GPU with 122GB memory and 6 cores for training. For evaluation of our models, we used a less powerful NVIDIA Tesla K80[5] with 56GB memory.

4.3 Dataset

The 2021 BraTS challenge dataset consists of 1250 volumetric MRI brain scans with the corresponding segmentation masks. Each scan comes with four different sequences (T1, T1ce, T2, FLAIR) and an annotated mask, given as NIfTI (.nii) files. Each volume has a size of 240x240 and consists of 155 different slices. An example of different sequences as well as an annotation can be seen in the figures (Fig. 4.1 and Fig. 4.2) respectively.

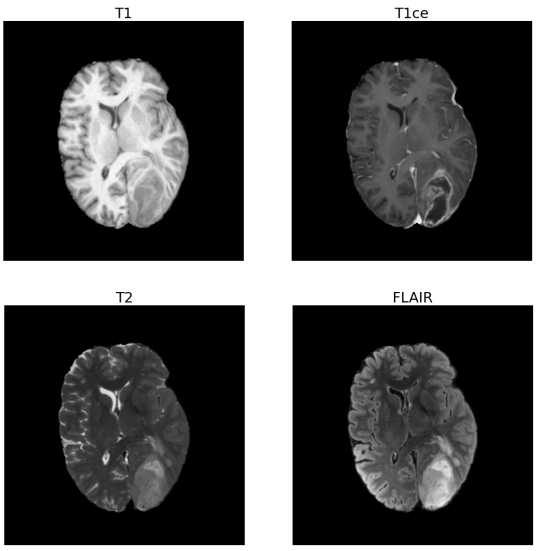


Figure 4.1: Single slice displayed in the T1, T1ce, T2 and FLAIR sequences.

[4] https://www.nvidia.com/en-gb/data-center/tesla-p100/
[5] https://www.nvidia.com/en-gb/data-center/tesla-k80/

The annotation comes with four different labels, describing different subregions of the glioma. The first label describes the area of the scan that does not include the glioma. The second label refers to Necrotic Tumour Core (NCR). NCR is the already dead (inanimate) part of the brain. It is also usually the inside part of the glioma because when gliomas reach a specific size, they can no longer supply themselves [45]. The third label is the Peritumoral Edematous/Invaded Tissue (ED). ED can either represent edema or swelling, which is the brain's reaction to the tumour. Alternatively, it can also be the invaded tissue. In this case, it is crucial to identify where the edema ends and the tumour begins, which is generally not well distinguishable on MRI. The fourth label is not present in any of the data because, since the 2017 BraTS challenge, this particular label has been merged with the second label (NCR) due to its small occurrence in the data [46]. The last label describes the Gd-Enhancing Tumour (ET). ET represents part of the tumour that is enhanced after contrast. The enhancement arises mainly because this region represents the "aggressive" part of the tumour that disrupts normal, healthy tissue, and the contrasting substance accumulates in the extracellular space [47].

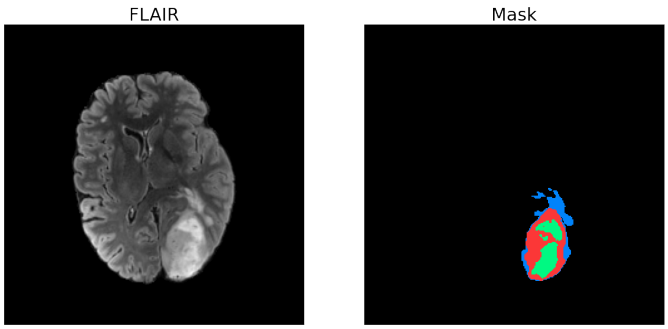


Figure 4.2: Annotated mask of a single slice. Each colour describes a different tumour subregion where black is unlabelled, blue represents ED, red represents ET and green represents NCR.

4.4 Network architecture

We decided to use a 3D U-Net architecture based on the work of Çiçek et al. [36]. The 3D U-Net has proven to be very effective and successful in segmenting volumetric medical images, as proven by the analysed works in the chapter 3.

The layout of our architecture is similar to the original U-Net [33] and can be seen in the figure (Fig. 4.3). It consists of two symmetrical paths, a downsampling path and an upsampling path, with four stages in each one and a bottleneck connecting the two paths. Each stage incorporates two convolutional layers followed by either a downsampling or an upsampling operation. The individual convolutional layer consists of 3D convolution with a kernel size 3x3x3 and zero padding. Except for the last one, all convolutional layers use ReLU as their activation function. The convolutional layers used in the downsampling path double the number of channels in the feature maps returned by the preceding stages, while the convolutional layers in the upsampling path do the exact opposite. At the beginning of each stage in the upsampling path, the feature maps from the corresponding stage in the downsampling path are concatenated to the feature maps upsampled in the preceding stage. The last convolutional layer uses a softmax activation function with a kernel size 1x1x1 and reduces the number of feature channels to match the number of classes.

The downsampling is performed by 3D max-pooling with pooling size 2x2x2 and stride 2 and consequently halves the size of the created feature maps. On the opposite, the upsampling of the feature maps is achieved by transposed 3D convolutions with kernel size 2x2x2, stride 2 and zero padding.

The first convolutional layer applies 16 filters, so the first feature map has a size of (128x128x128x16) and is subsequently reduced by pooling operations to (8x8x8x256) in the downsampling path and then enlarged to the initial resolution

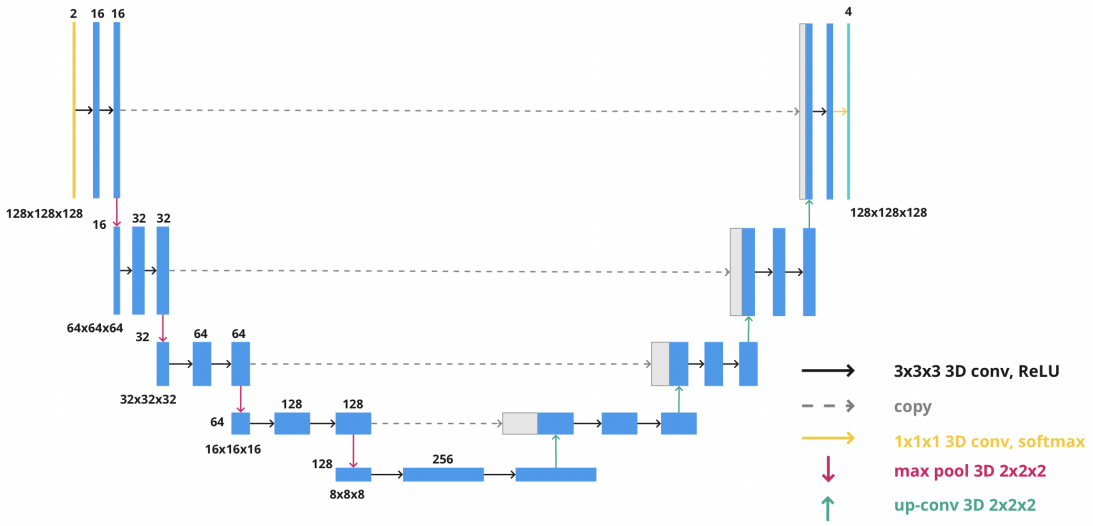


Figure 4.3: Modified 3D U-Net architecture.

in the upsampling path. The last convolutional layer has a kernel size of 1x1x1 and applies four filters (one for every label in the mask).

We used the proposed architecture across all experiments with only the shape of the input layer and output layer altering. In the input layer, the final shape is based on the number of sequences used in the given experiment. Similarly, the output layer of the network changes depending on the number of classes we are trying to segment.

4.5 Data loading

The dataset consists of 1250 folders, where each includes 5 NIfTI files (4 sequences and 1 annotated mask). We decided to split the dataset into three groups, one for training, validation and testing, with a ratio of 60 to 20 to 20. We opted for splitting the dataset into three groups instead of two in order to be able to assess the quality of the proposed method more precisely. The split resulted in the

following layout, with 750 folders for the training set and 250 folders for both the validation and testing sets.

For the data loading itself, we used the NiBabel[6] Python library, with which we were able to load the provided NIfTI files into NumPy[7] arrays, that can be further utilised in the pipeline.

Since the data generator included in TensorFlow does not support NIfTI files and loading the data beforehand would be memory intensive, we decided to implement our data generator. The data generator loads the data, runs the corresponding preprocessing, optionally augments the volumes and finally yields back batches of preprocessed volumes.

4.6 Data preprocessing

The original volumes have a lot of background around the brain. We decided to crop the original volume to include the minimum amount of background and still include the whole brain in the frame. Since the original volumes are centred, we were able to crop the images from the centre by 70%, which happened to be the correct scale.

We downsampled the volumes from the original 240x240 to 128x128. Similarly, we omitted the first 13 and last 14 slices from the original 155 because they did not include any part of the brain or the tumour. At this stage the tensors had a size of 128x128x128.

The intensities in MRI sequences can vary a lot, which can be attributed to their dependency on multiple variables, such as conditions during acquisition, settings

[6] `https://nipy.org/nibabel/`
[7] `https://numpy.org`

or manufacturer [48]. Since high variance among data can significantly affect the model's training, it is crucial to normalise the input sequences beforehand. For data normalisation, we decided to use Min-Max scaling. After the normalisation, specified sequences were stacked together similarly to image channels where in this case, one sequence resembles one image channel.

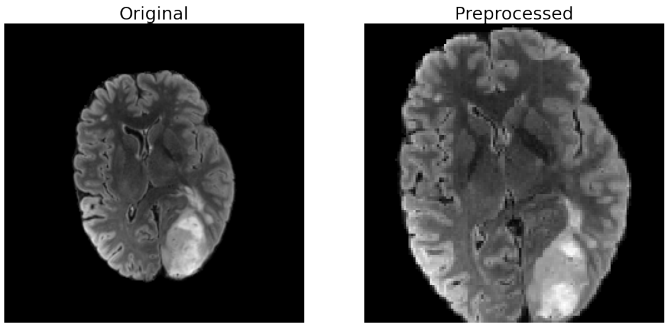


Figure 4.4: Comparison between an original and preprocessed FLAIR sequence.

The annotated masks went through the same process of cropping, downsampling and augmentation. However, since the masks consist of four distinct values (0, 1, 2, 4), one for each tumour subregion, we did not need to apply normalisation, but we did change the label 4 to label 3 for the sake of consistency. The adjusted masks were later encoded into binary tensors based on the specified number of classes we are trying to segment.

The final tensors had a shape of: `(x,128,128,128,y)`, where `x` represents the batch size, and `y` represents used sequences for input tensors or classes for output tensors.

4.7 Data augmentation

We implemented simple data augmentations to expand our training dataset. The implemented augmentations were a rotation by random degrees ranging from 10°

to -10° with 50% chance and vertical flipping with 50% chance.

4.8 Evaluation metrics

For monitoring the training process as well as for the final evaluation of the model, we decided to use the dice score coefficient (DSC), which measures the overlap between the computed segmentation and ground truth. The equation describing the DSC can be found in the analytical part of the document (Equation 2.7). Since we are segmenting multiple classes, we computed the final DSC by summing the DSC for each class and dividing it by the number of classes.

4.9 Evaluation method

We evaluated our trained models on the testing set, which was not used during training and was supposed to simulate how would the models work on unseen data. The evaluation alone consisted of computing the mean of the overall DSC and the DSCs for each class. Based on the overall DSC, the worst five segmentations were assessed manually to identify any specific problems.

4.10 Experiments

We experimented with two different approaches to segment the individual parts of glioma. Both methods used the same architecture described in 4.4 with the only differences being in the input and output shape of the network. Both methods were trained for 50 epochs with batch size 2, Adam optimiser and categorical cross entropy as their loss function.

4.10.1 Segmenting the whole tumour

In the first approach, we tried to segment all of the classes at once and compare the use of multiple sequences, augmentations and learning rates on the final segmentation. For the evaluation of this approach, we grouped the experiments by the number of used sequences and later compared the best combinations among the groups.

At first, we tried to use the FLAIR and T1ce sequences. We have chosen these two particular sequences because the overall shape of the tumour and the individual subregions are very well distinguishable on them. The tried combinations are listed in the table (Tab. 4.1). The use of augmentations improved the performance of the models significantly, so we consequently decided to train all the subsequent models with augmentations.

sequences	learning rate	augmentations
FLAIR, T1ce	0.001	no
FLAIR, T1ce	0.001	yes
FLAIR, T1ce	0.0001	yes

Table 4.1: The tested combinations with two sequences.

During the evaluation, we noticed that models trained with two sequences did not perform well on segmenting the NCR subregion, so we decided to use an additional T2 sequence (Tab. 4.2). Most state-of-the-art models also use this particular combination of three sequences [37, 25].

sequences	learning rate	augmentations
FLAIR, T1ce, T2	0.001	yes
FLAIR, T1ce, T2	0.0001	yes
FLAIR, T1ce, T2	0.00001	yes

Table 4.2: The tested combinations with three sequences.

We also tried to use all four sequences with the combination of 0.0001 learning rate

and augmentations, which performed the best in the previous experiments (Tab. 4.3).

sequences	learning rate	augmentations
FLAIR, T1ce, T2, T1	0.0001	yes

Table 4.3: The tested combination with four sequences.

4.10.2 Segmenting each class independently

In this approach, we trained an ensemble of three models, one for each class, and combined the segmented subregions into one final segmentation. A scheme describing this approach is in figure (Fig. 4.5). Similarly to the previous approach, we tried two different combinations of sequences, but this time all of the models were trained by using a 0.0001 learning rate. The data preprocessing had to be slightly modified, particularly for the annotations, where we replaced the values for the selected class with 1 and others with 0.

The combined segmentation was created by merging the individual segmentations based on their values in the selected voxel. In case of a collision, or when multiple models segmented the same voxel, we chose the one with a higher value in the probability map returned by the respective model.

At first we tried to use two different sequences, FLAIR and T1ce for training. The ensemble trained with these two sequences did not perform so well for the NCR and ED subregions. We later hypothesised that the use of an additional T2 sequence would be particularly beneficial for the segmentation of NCR and ED tumour subregions since they are more distinguishable on the T2 sequence.

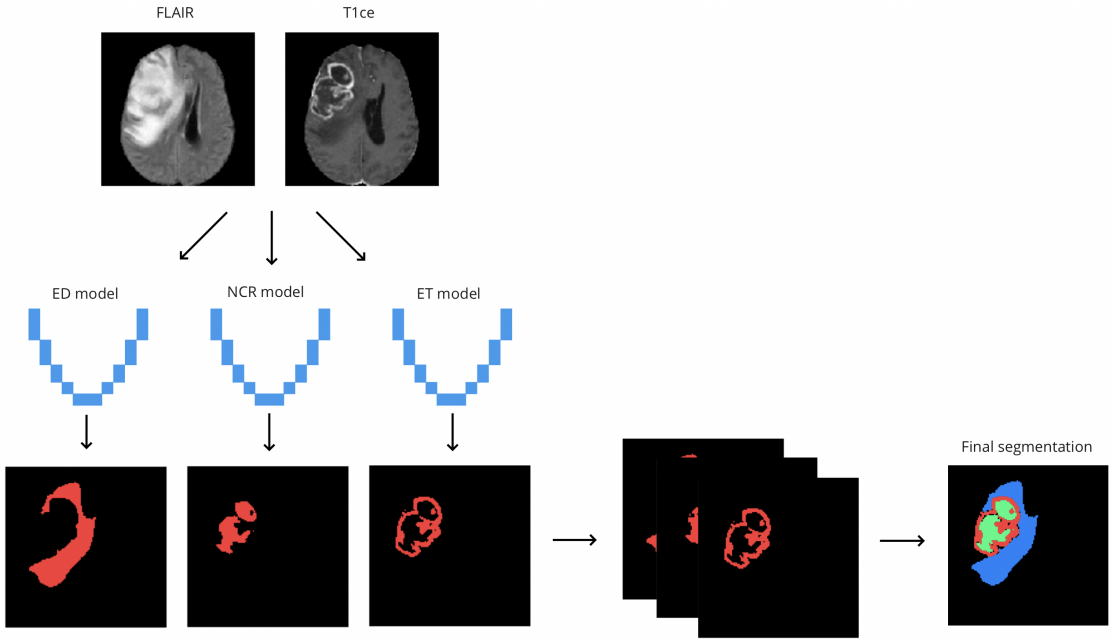


Figure 4.5: Scheme describing the separate class training, consisting of training three different models and then combining the individual segmentations into one final segmentation.

Chapter 5

Results

In this chapter we evaluated the experiments defined in the section 4.10 and compared the best results from the two approaches. For evaluation, we used the same procedure specified in 4.9 where we focused primarily on the overall DSC.

5.1 Results for the whole tumour segmentation

In this approach, we tried multiple combinations of learning rates (lr), augmentations (aug) and sequences. We divided these experiments into three groups based on the number of sequences used during training.

	Overall	NCR	ED	ET
DSC - 0.001 lr	0.6650	0.3340	0.6618	0.6225
DSC - 0.001 lr, with aug	0.7075	0.5138	0.6760	0.6440
DSC - 0.0001 lr, with aug	0.7359	0.5461	0.7196	0.6810

Table 5.1: Results for the models trained with two sequences. The best performing combination was the 0.0001 lr with augmentations.

The first group used two sequences, FLAIR and T1ce. The best results were

achieved by the combination of a 0.0001 learning rate with augmentations, which achieved an overall DSC of 0.7359, visible in the table (Tab. 5.1).

	Overall	NCR	ED	ET
DSC - 0.001 lr, with aug	0.6830	0.4551	0.6701	0.6106
DSC - 0.0001 lr, with aug	0.7511	0.5694	0.7362	0.7020
DSC - 0.00001 lr, with aug	0.5471	0.2145	0.5660	0.4125

Table 5.2: Results for the models trained with three sequences, where the best combination was a 0.0001 learning rate with augmentations.

The second group consisted of models trained with three sequences, FLAIR, T1ce and T2 and with augmentations. For this group we tried three different learning rates, from which, the 0.0001 learning rate performed the best with 0.7511 overall DSC (Tab. 5.2). In this experiment, we also tried to use a higher learning rate of 0.00001, which did not improve the final results. Comparison between the training of these three models can be seen in the Appendix A.1.

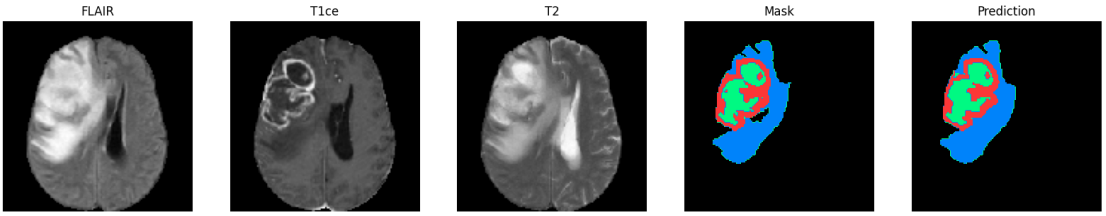


Figure 5.1: An example segmentation for the best model trained with three sequences. On this particular scan the model achieved 0.8765 overall DSC, 0.8874 DSC for NCR, 0.8693 for ED and 0.8765 for ET.

	Overall	NCR	ED	ET
DSC - 0.0001 lr, with aug	0.7354	0.5443	0.7190	0.6814

Table 5.3: Results for the model trained with all four sequences.

In the third group, we tried to use all four sequences with the combination of 0.0001 learning rate and augmentations which performed the best in previous

experiments. This model achieved an overall DSC of 0.7354 on the testing set (Tab. 5.2).

The best overall model was trained using three sequences, FLAIR, T1ce and T2 with 0.0001 learning rate and augmentations. This model achieved an overall DSC of 0.7511, 0.5694 for the NCR, 0.7362 for the ED and 0.7020 for the ET subregion. The results for this experiment are in the table (Tab. 5.4). Segmentation from this model can be seen in the figure (Fig. 5.2).

	Overall	NCR	ED	ET
DSC - 2 sequences	0.7359	0.5461	0.7196	0.6810
DSC - 3 sequences	0.7511	0.5694	0.7362	0.7020
DSC - 4 sequences	0.7354	0.5443	0.7190	0.6814

Table 5.4: The best results from each group, where the model trained with 3 sequences performed the best overall.

5.2 Results for the separate class training

In this approach, we trained three models, where each was used to segment an individual subregion. All models were trained with the same learning rate, equal to 0.0001. In this case, we compared the use of two and three sequences.

The ensemble trained with the FLAIR and T1ce sequences achieved an overall DSC of 0.7377 (Tab. 5.5).

	Overall	NCR	ED	ET
DSC	0.7377	0.5722	0.6438	0.7393

Table 5.5: The results of our ensemble trained with two sequences

We later hypothesised that the use of an additional T2 sequence would be beneficial for the performance of our ensemble, particularly for the NCR and ED tumour subregions since they are more distinguishable on the T2 sequence.

The overall DSC improved from 0.7377 to 0.8064 with the use of three sequences, which was also the case for DSCs among all of the separate subregions as can be seen in the table (Tab. 5.6).

	Overall	NCR	ED	ET
DSC	0.8064	0.6242	0.7927	0.8115

Table 5.6: The results of our ensemble trained with three sequences

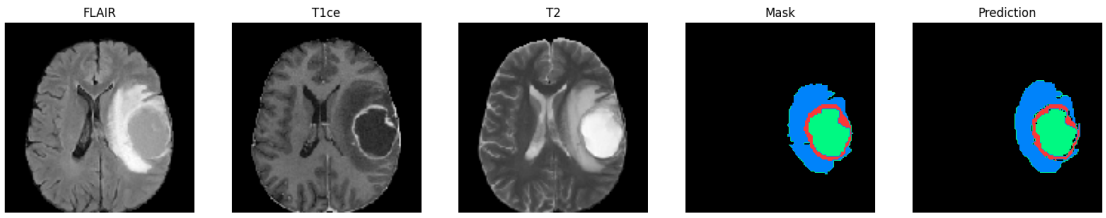


Figure 5.2: An example segmentation for the best ensemble of models. On this particular scan the model achieved 0.9510 overall DSC, 0.9396 DSC for NCR, 0.9590 for ED and 0.9074 for ET.

During training the model trained for segmenting the ED subregion achieved a final validation DSC of 0.8748 compared to the 0.8260 achieved by the two sequence model. The direct comparison between the training of these two models can be seen in the Appendix A.2. The results for the other subregions were more or less the same.

5.3 Segmenting a different type of brain tumour

We also tried to segment hypophysis from different T1ce sequences to see how would our model work on different types of brain tumours. For this experiment, we used the best model from section 5.1. Overall our model did segment the hypophysis with certain artefacts caused by the presence of skull, blood vessels or eyes in the sequences that are not present in the data we used for training, as visible in the figure (Fig. 5.3).

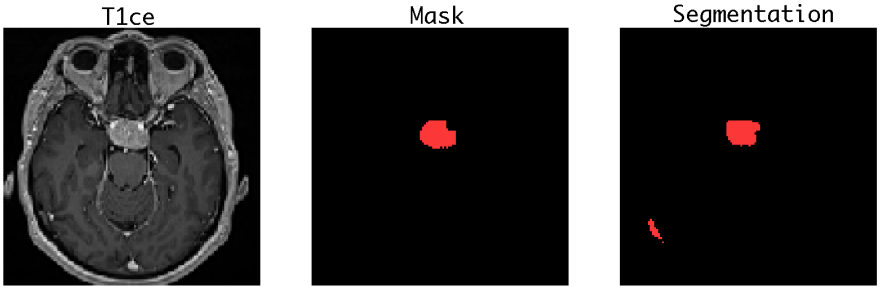


Figure 5.3: A segmentation of hypophysis from T1ce sequences, which achieved a DSC of 0.6558.

Chapter 6

Conclusion

The aim of our work was to design a deep learning method to segment individual glioma subregions from MRI sequences. We came up with and evaluated two different approaches based on the 3D U-Net architecture, which is also used by most state-of-the-art methods. For the training and evaluation, we used the BraTS 2021 dataset and the Azure Machine Learning services for the computational resources.

At first, we decided to implement a more straightforward method of optimising for all subregions at once. With this approach, we tried multiple different combinations of learning rates, augmentations and used sequences to get a broader selection of models. The best overall model was trained with three sequences (FLAIR, T1ce and T2) with augmentations and a 0.0001 learning rate. With this model, we could segment the glioma subregions relatively well. The model was notably underperforming on the segmentation of necrosis. We tried to mitigate this problem by incorporating an additional sequence, which did not prove to be particularly helpful in the end.

With the second approach, we tried to improve upon the first one regarding the network's performance on underrepresented classes. This time we trained one model for each class and later combined the individual segmentations into a final one. We also compared the use of two (FLAIR and T1ce) and three (FLAIR, T1ce and T2) sequences. Our models were trained with the same learning rate that performed the best in the previous approach, and similarly, the use of three sequences proved to be the best. With this method, we managed to significantly improve upon the previous approach for all subregions.

In the end, we successfully designed and compared two methods from which both could segment the tumour subregions sufficiently. The second method achieved higher DSCs for all subregions and was, therefore, more successful than the first, but on the other hand, the training of an ensemble of multiple models requires more time.

The performance of the models could be, for example, further improved by using a different loss function. We used a categorical cross entropy in our experiments which might not be optimal for unbalanced segmentation with underrepresented classes. In this case, it would be beneficial to use dice loss or weighted loss functions, which can mitigate the effect of the issue.

Another interesting approach would be to consider the use of another imaging modality to confirm the correctness of the segmented regions. Currently, the segmentations might not be entirely correct from the physiological standpoint, even when they achieve high evaluation scores and are therefore very similar to the ground truth masks. We noticed that the annotations suddenly changed in some masks from one class to another, mainly for ED and ET classes. After discussing with a medical doctor in the field of nuclear medicine, he confirmed to us that this was due to the fact that the annotators were not sure which label to use. This is a

common problem with the MRI images, where the parts of the tumour are not well distinguishable in some cases, and medical doctors would use another modality to help them. The Positron Emission Tomography (PET) images are commonly used in such cases because, as opposed to MRI, which mainly describes the structural changes of the tissue, the PET images also provide metabolic information that allows for better characterisation of the disease.

Resumé

6.1 Úvod

Pokrok v oblasti hlbokých neurónových sietí, najmä vývoj nových metód, akými sú napríklad konvolučné neurónové siete (CNN), mal obrovský vplyv na ďalšie zdokonaľovanie spracovania obrazu a stali sa aj neoddeliteľnou súčasťou spracovania diagnostických medicínskych dát, akými sú magnetická rezonancia (MRI) alebo počítačová tomografia (CT) [1]. Konvolučné neurónové siete môžu byť v tomto prípade aplikované na úlohy akým je aj segmentácia nádorov [2]. My sa v tejto práci zameriavame predovšetkým na segmentáciu gliómov (mozgových nádorov) z MRI snímkov.

Práve MRI je dôležitou modalitou pri práci s neurologickými dátami a segmentácia gliómov z týchto dát je nevyhnutnou súčasťou diagnostiky a následného plánovania liečby pacienta [4]. Na presné posúdenie nádoru alebo jeho rôznych podoblastí sa používajú viaceré sekvencie magnetickej rezonancie. Každá z týchto sekvencií predstavuje inú metódu použitú na získanie dát z magnetickej rezonancie. Rôzne sekvencie majú iné vlastnosti, akými sú napríklad kontrast alebo rýchlosť akvizície, vďaka ktorým zvýrazňujú iné tkanivo alebo časť nádoru [5]. Najčastejšie používanými sekvenciami sú T1 sekvencia, ktorá sa používa na rozlíšenie bielej a šedej mozgovej hmoty, T1ce, ktora je vlastne T1 sekvencia s kontrastom a používa

sa na odhalenie krvácania, T2 sekvencia, ktorá sa používa na ohraničenie edému a FLAIR sekviencia, ktorá sa používa na zvíraznenie agresívnej časti nádoru. So všetkými spomínanými sekvenciami pracujeme aj my vo svojej práci.

6.2 Analýza

Cieľom strojového učenia je v podstate poskytnúť počítačom schopnosť riešiť problémy rôznych úloh učením sa zo vstupných dát. Tento cieľ je možné dosiahnuť vytváraním tzv. modelov. Tieto modely sú trénované na vstupných dátach, aby produkovali výstupy založené na vlastnostiach extrahovaných z daných dát [9]. Výstup z týchto modelov je špecifický pre danú úlohu, kde napríklad pri klasifikácii môže byť výstupom množina tried predpovedaná modelom alebo pri segmentácii býva výstupom typicky ohraničená časť alebo viac častí z daného vstupného obrázku.

Hlavným cieľom pri vytváraní týchto modelov je urobiť ich všeobecne použiteľnými, teda aby dokázali poskytnúť správny výsledok nielen pre dáta použité počas trénovania, ale aj pre nové a doposiaľ nevidené dáta, čo môžeme dosiahnuť väčšou diverzitou vstupných dát alebo aj ich augmentáciami.

6.2.1 Umelé neurónové siete

Hlavnou časťou neurónových sietí sú neuróny. Tieto neuróny predstavujú výpočtové uzly, ktoré sú funkčne podobné biologickým neurónom [15]. Telo neurónu slúži na vypočítanie súčtu z daných vstupov. Tento súčet je následne prenesený do špecifikovanej aktivačnej funkcie, ktorá vypočíta výstup neurónu. Neuróny sú organizované vo vrstvách, kde sa výstup neurónu z predchádzajúcej vrstvy môže stať vstupom pre neuróny v nasledujúcich vrstvách [13]. Architektúra umelej neurónovej siete s veľkým počtom skrytých vrstiev sa často označuje pojmom hlboké

neurónové siete [12]. Základnou výhodou hlbokých neurónových sietí je ich flexibilnosť a všeobecnosť, vďaka ktorým sa hodia pre úlohy počítačového videnia, akými je napríklad klasifikácia alebo segmentácia.

Proces učenia umelých neurónových sietí je reprezentovaný hľadaním správnych hodnôt pre jednotlivé parametre používané v neurónoch. Tento proces je podmienený minimalizovaním výsledku stratových funkcií, ktoré slúžia na penalizovanie siete pri zlých výsledkoch [20].

6.2.2 Konvolučné neurónové siete

Konvolučné neurónové siete sú typom architektúry hlbokých neurónových sietí, ktorá je vhodná na spracovanie údajov vo forme viacrozmernej matice, akými sú napríklad obrázky, vďaka čomu sú mimoriadne efektívne pri úlohách počítačového videnia [13]. Oproti štandardným neurónovým sieťam, konvolučné neurónové siete zachovávajú tvar vstupu, čo môže pomôcť identifikovať dôležité časti obrázku akými sú napríklad hrany [27].

Typická architektúra konvolučných neurónových sietí sa skladá z troch vrstiev. Jednou z vrstiev sú konvolučné vrstvy, ktoré rozdelia vstup na viac rôznych častí alebo filtrov. V týchto filtroch dokáže sieť rozoznať dôležité prvky zo vstupu [14]. Ďalšou vrstvov sú takzvané "pooling"vrstvy, ktoré zmenšujú rozmery vstupu čím zlepšujú výpočtovú zložitosť siete. Poslednými vrstvami bývajú úplne prepojené vrstvy, ktoré sa používajú na výpočet konečného výstupu zo siete. Medzi populárne architektúry konvolučných neurónových sietí patrí architektúra U-Net, ktorá je veľmi úspešná najmä v segmentácií medicínskych dát [33].

6.3 Naša práca

Cieľom našej práce bolo navrhnúť metódu hlbokého učenia na segmentáciu jednotlivých častí gliómov z MRI snímok. Na tento účel sme vymysleli a vyhodnotili viacero prístupov založených na 3D U-Net architektúre. Na trénovanie a validáciu sme sa rozhodli použiť dáta zo súťaže BraTS 2021. BraTS 2021 dáta pozostávajú z 1250 volumetrických MRI skenov mozgu so zodpovedajúcimi segmentačnými maskami. Každý sken sa skladá so štyroch rôznymich sekvencii (T1, T1ce, T2, FLAIR) a anotácie, ktoré sú forme súborov NIfTI. Anotácia popisuje štyri rôzne časti gliomu: nekróza (NCR), edém (ED) a agresívnu časť nádoru (ET). Jednotlivé skeny sme si rozdelil do troch častí: na trénovanie (750 skenov), validáciu (250) a testovanie (250). Pri trénovaní sme ešte implementovali jednoduché augmentácie, aby sme ďalej rozšírili trénovaciu množinu.

Naše riešenie sme implementovali v jazyku Python 3 s použitím rámca pre strojového učenie TensorFlow. Trénovanie a evaluácia našich modelov boli implementované v podobe Python skriptov, ktoré boli spúšťané v službe Azure Machine Learning.

6.3.1 Použitá architektúra

V práci sme sa rozhodli použiť 3D U-Net architektúru. Štruktúra našej architektúry je podobná pôvodnej U-Net a pozostáva z dvoch symetrických častí, pričom každá časť obsahuje štyri úrovne. Jednotlivé úrovne obsahujú dve konvolučné vrstvy, po ktorých nasleduje buď prevzorkovanie nadol alebo nahor. Každá konvolučná vrstva používa ako aktivačnú funkciu ReLU, okrem poslednej, ktorá používa Softmax na výpočet konečného výstupu zo siete.

6.3.2 Predspracovanie dát

Pôvodné dáta sme orezali pretože obsahujú veľkú časť pozadia. Jednotlivé sekvencie sme taktiež zoškálovali z 240x240x155 na 128x128x128. Keďže sa intenzity v MRI sekvenciách môžu od seba veľmi líšiť a veľký rozptyl medzi hodnotami môže výrazne ovplyvniť tréning modelu, museli sme dáta normalizovať. Po normalizácii boli dané sekvencie naskladané dohromady podobne ako obrazové kanály, kde v tomto prípade jedna sekvencia pripomína jeden obrazový kanál.

6.3.3 Spôsob evaluácie

Natrénované modely sme hodnotili na testovacej množine, ktorá nebola použitá pri tréningu. Samotné hodnotenie pozostávalo z výpočtu priemeru celkového dice koeficentu (DSC) a dice koeficentu pre každú triedu. Dice koeficient meria prekrytie medzi vypočítanou segmentáciou a danou anotáciou.

6.3.4 Experimenty

Experimentovali sme s dvoma rôznymi prístupmi k segmentácii jednotlivých častí gliómu. Obe metódy používali rovnakú architektúru s jediným rozdielom vo vstupnom a výstupnom tvare siete. Obe metódy boli taktiež trénované pre 50 epoch s "batch"veľkosťou 2, Adam optimalizátorom a kategorickou krížovou entropiou ako stratovou funkciou.

V prvom prístupe sme sa pokúsili rozdeliť všetky triedy naraz a porovnať použitie viacerých sekvencií, augmentácií a rýchlosti učenia na konečnú segmentáciu. Pre vyhodnotenie tohto prístupu sme experimenty zoskupili podľa počtu použitých sekvencií a neskôr porovnali najlepšie kombinácie medzi skupinami.

V druhom prístupe sme trénovali súbor troch modelov, jeden pre každú triedu

a na koniec sme spojili jednotlivé segmentované podoblasti do jednej konečnej segmentácie. Podobne ako v predchádzajúcom prístupe sme vyskúšali dve rôzne kombinácie sekvencií, ale nechali rovnakú rýchlosť učenia a augmentácie. Kombinovaná segmentácia vznikla zlúčením jednotlivých segmentácií na základe ich hodnôt vo vybranom voxeli. V prípade kolízie, alebo keď viacero modelov segmentovalo rovnaký voxel, sme vybrali ten s vyššou hodnotou v mape pravdepodobnosti vrátenej príslušným modelom.

6.3.5 Výsledky

Najlepší celkový model pre prvý experiment bol trénovaný pomocou troch sekvencií, FLAIR, T1ce a T2 s rýchlosťou učenia 0,0001 a augmentáciami. Tento model dosiahol celkovú DSC 0,7511, 0,5694 pre NCR, 0,7362 pre ED a 0,7020 pre ET subregión. Pri tomto experimente sme si všimli, že najväčší vplyv na výsledok modelu malo použitie augmentácií.

Pri druhom experimente bola rovnako najlepšia kombinácia troch sekvencií. Tento model získal celkový DSC 0,8064, 0,6242 pre NCR, 0,7927 pre ED a 0,8115 pre ET. V tomto experimente sme všetky modely trénovali s augmentáciami a rovnakou rýchlosťou učenia, ktorá bola konzistentne najlepšia v predchádzajúcom experimente.

6.4 Záver

Cieľom našej práce bolo navrhnúť metódu hlbokého učenia na segmentáciu jednotlivých podoblastí gliómu z MRI sekvencií. Navrhli sme a vyhodnotili dva rôzne prístupy založené na U-Net architektúre, ktorú využíva aj väčšina najmodernejších metód.

Najskôr sme sa rozhodli implementovať priamočiarejšiu metódu, ktorá segmentovala všetky podoblasti naraz. Najlepší celkový model bol trénovaný s tromi sekvenciami, s augmentáciami a rýchlosťou učenia 0,0001. Pomocou tohto modelu sme už dokázali relatívne dobre segmentovať časti gliómu. Model mal, ale výrazne nedostatočnú výkonnosť pri segmentácii nekrózy. S druhým prístupom sme sa pokúsili vylepšiť prvý, s ohľadom na výkon siete pri nedostatočne zastúpených triedach. Tentokrát sme natrénovali jeden model pre každú triedu a neskôr, sme jednotlivé segmentácie spojili do finálnej.

Nakoniec sme úspešne navrhli a porovnali dve metódy, z ktorých obe dokázali dostatočne segmentovať nádorové podoblasti. Výkonnosť modelov by sa mohla ďalej zlepšiť použitím inej stratovej funkcie. V našich experimentoch sme použili kategorickú krížovú entropiu, ktorá nemusí byť optimálna pre nevyváženú segmentáciu.

References

[1] Ronald M. Summers. "Deep Learning and Computer-Aided Diagnosis for Medical Image Processing: A Personal Perspective". In: *Deep Learning and Convolutional Neural Networks for Medical Image Computing: Precision Medicine, High Performance and Large-Scale Datasets*. Cham: Springer International Publishing, 2017, pp. 3–10. ISBN: 978-3-319-42999-1. DOI: https://doi.org/10.1007/978-3-319-42999-1_1.

[2] Luyan Liu et al. "Overall survival time prediction for high-grade glioma patients based on large-scale brain functional networks". In: *Brain imaging and behavior* 13.5 (2019), pp. 1333–1351. DOI: https://doi.org/10.1007/s11682-018-9949-2.

[3] Edward F. Jackson et al. "A review of MRI pulse sequences and techniques in neuroimaging". In: *Surgical Neurology* 47.2 (1997), pp. 185–199. ISSN: 0090-3019. DOI: https://doi.org/10.1016/S0090-3019(96)00375-8.

[4] Soonmee Cha. "Update on brain tumor imaging". In: *Current neurology and neuroscience reports* 5.3 (2005), pp. 169–177. DOI: https://doi.org/10.1007/s11910-005-0044-x.

[5] Javier E. Villanueva-Meyer, Marc C. Mabray, and Soonmee Cha. "Current Clinical Brain Tumor Imaging". In: *Neurosurgery* 81.3 (May 2017), pp. 397–415. ISSN: 0148-396X. DOI: https://doi.org/10.1093/neuros/nyx103.

[6] Mary Elizabeth Davis. "Epidemiology and Overview of Gliomas". In: *Seminars in Oncology Nursing* 34.5 (2018). Adult Gliomas, pp. 420–429. ISSN: 0749-2081. DOI: https://doi.org/10.1016/j.soncn.2018.10.001.

[7] Quinn T Ostrom et al. "CBTRUS statistical report: primary brain and central nervous system tumors diagnosed in the United States in 2007–2011". In: *Neuro-oncology* 16.suppl_4 (2014), pp. iv1–iv63. ISSN: 1522-8517. DOI: https://doi.org/10.1093/neuonc/nou223.

[8] Bradley J Erickson et al. "Machine learning for medical imaging". In: *Radiographics* 37.2 (2017), pp. 505–515. DOI: https://doi.org/10.1148/rg.2017160130.

[9] Alexander Selvikvåg Lundervold and Arvid Lundervold. "An overview of deep learning in medical imaging focusing on MRI". In: *Zeitschrift für Medizinische Physik* 29.2 (2019). Special Issue: Deep Learning in Medical Physics, pp. 102–127. ISSN: 0939-3889. DOI: https://doi.org/10.1016/j.zemedi.2018.11.002.

[10] Xue Ying. "An overview of overfitting and its solutions". In: *Journal of Physics: Conference Series*. Vol. 1168. 2. IOP Publishing. 2019, p. 022022. DOI: https://doi.org/10.1088/1742-6596/1168/2/022022.

[11] Li Wan et al. "Regularization of Neural Networks using DropConnect". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 2013, pp. 1058–1066. URL: https://proceedings.mlr.press/v28/wan13.html.

[12] Richard E Neapolitan and Xia Jiang. *Artificial intelligence: With an introduction to machine learning*. CRC Press, 2018.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444. DOI: https://doi.org/10.1038/nature14539.

[15] Kevin Gurney. *An introduction to neural networks*. CRC press, 2018. DOI: https://doi.org/10.1201/b22400.

[16] MATTIAS WAHDE. "Introduction to Neural Networks". In: (2007).

[17] Eda Kavlakoglu. *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?* https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks, (22.12.2021). 2020.

[18] Raniah Zaheer and Humera Shaziya. "GPU-based empirical evaluation of activation functions in convolutional neural networks". In: *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. 2018, pp. 769–773. DOI: https://doi.org/10.1109/ICISC.2018.8398903.

[19] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks". In: *towards data science* 6.12 (2017), pp. 310–316.

[20] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. http://neuralnetworksanddeeplearning.com, (14.5.2022). Determination press San Francisco, CA, 2015.

[21] Anna Choromanska et al. *The Loss Surfaces of Multilayer Networks*. 2014. DOI: https://doi.org/10.1016/10.48550/ARXIV.1412.0233.

[22] Carole H Sudre et al. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations". In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248. DOI: https://doi.org/10.1007/978-3-319-67558-9_28.

[23] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. *V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation*. 2016. arXiv: 1606.04797 [cs.CV].

[24] Nabila Abraham and Naimul Mefraz Khan. "A Novel Focal Tversky Loss Function With Improved Attention U-Net for Lesion Segmentation". In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. 2019, pp. 683–687. DOI: https://doi.org/10.1109/ISBI.2019.8759329.

[25] Fabian Isensee et al. *nnU-Net for Brain Tumor Segmentation*. 2020. arXiv: 2011.00848 [eess.IV].

[26] Michael Yeung et al. "Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation". In: *Computerized Medical Imaging and Graphics* 95 (2022), p. 102026. ISSN: 0895-6111. DOI: https://doi.org/10.1016/j.compmedimag.2021.102026.

[27] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: https://doi.org/10.1109/ICEngTechnol.2017.8308186.

[28] Vincent Dumoulin and Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[29] Md. Zahangir Alom et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures". In: *Electronics* 8 (Mar. 2019), p. 292. DOI: https://doi.org/10.3390/electronics8030292.

[30] Phil Kim. "Convolutional Neural Network". In: *MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence*. Berkeley, CA: Apress, 2017, pp. 121–147. ISBN: 978-1-4842-2845-6. DOI: https://doi.org/10.1007/978-1-4842-2845-6_6.

[31] Panqu Wang et al. "Understanding Convolution for Semantic Segmentation". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2018, pp. 1451–1460. DOI: https://doi.org/10.1109/WACV.2018.00163.

[32] Davide Mazzini. "Guided Upsampling Network for Real-Time Semantic Segmentation". In: *CoRR* abs/1807.07466 (2018). arXiv: 1807.07466.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[34] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: https://doi.org/10.1109/TPAMI.2016.2572683.

[35] Michal Drozdzal et al. "The Importance of Skip Connections in Biomedical Image Segmentation". In: *Deep Learning and Data Labeling for Medical Applications*. Cham: Springer International Publishing, 2016, pp. 179–187. ISBN: 978-3-319-46976-8. DOI: https://doi.org/10.1007/978-3-319-46976-8_19.

[36] Özgün Çiçek et al. *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. 2016. DOI: https://doi.org/10.48550/ARXIV.1606.06650.

[37] Theophraste Henry et al. *Brain tumor segmentation with self-ensembled, deeply-supervised 3D U-net neural networks: a BraTS 2020 challenge solution*. 2020. arXiv: 2011.01045 [eess.IV].

[38] Michał Futrega et al. *Optimized U-Net for Brain Tumor Segmentation*. 2021. arXiv: 2110.03352 [eess.IV].

[39] Hao Dong et al. "Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks". In: *Medical Image Understanding and Analysis*. Cham: Springer International Publishing, 2017, pp. 506–517. ISBN: 978-3-319-60964-5. DOI: https://doi.org/10.1007/978-3-319-60964-5_44.

[40] Fabian Isensee et al. "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation". In: *Nature methods* 18.2 (2021), pp. 203–211. DOI: https://doi.org/10.1038/s41592-020-01008-z.

[41] Andriy Myronenko. *3D MRI brain tumor segmentation using autoencoder regularization*. 2018. arXiv: 1810.11654 [cs.CV].

[42] Ujjwal Baid et al. *The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification*. 2021. arXiv: 2107.02314 [cs.CV].

[43] Bjoern H Menze et al. "The multimodal brain tumor image segmentation benchmark (BRATS)". In: *IEEE transactions on medical imaging* 34.10 (2014), pp. 1993–2024. DOI: https://doi.org/10.1109/TMI.2014.2377694.

[44] Spyridon Bakas et al. "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features". In: *Scientific data* 4.1 (2017), pp. 1–13. DOI: https://doi.org/10.1038/sdata.2017.117.

[45] Stacey Watkins et al. "Disruption of astrocyte–vascular coupling and the blood–brain barrier by invading glioma cells". In: *Nature communications* 5.1 (2014), pp. 1–15. DOI: https://doi.org/10.1038/ncomms5196.

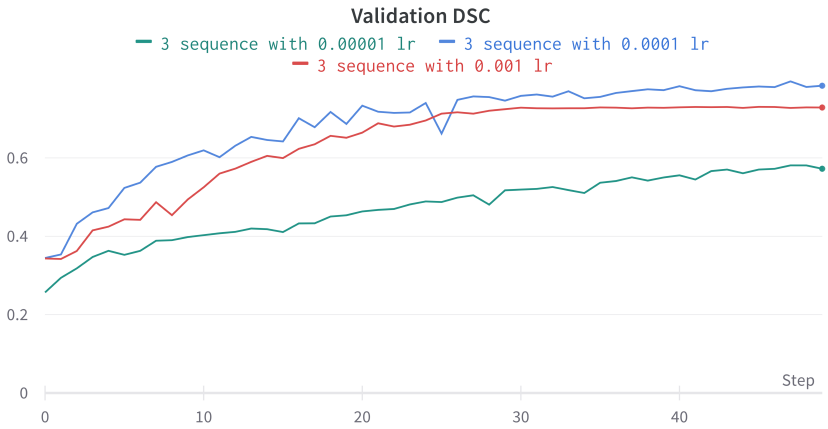
[46] Spyridon Bakas et al. *Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge*. 2019. arXiv: 1811.02629 [cs.CV].

[47] Ganesh Tameshwar et al. "Manganese-enhanced MRI of minimally gadolinium-enhancing breast tumors". In: *Journal of Magnetic Resonance Imaging* 41.3 (2015), pp. 806–813. DOI: https://doi.org/10.1002/jmri.24608.

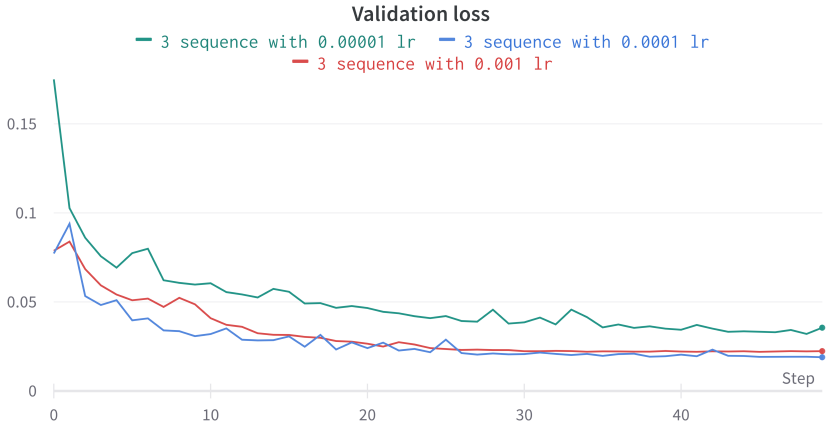
[48] G. Collewet, M. Strzelecki, and F. Mariette. "Influence of MRI acquisition protocols and image intensity normalization methods on texture classification". In: *Magnetic Resonance Imaging* 22.1 (2004), pp. 81–91. ISSN: 0730-725X. DOI: https://doi.org/10.1016/j.mri.2003.09.001.

Appendix A

Training graphs

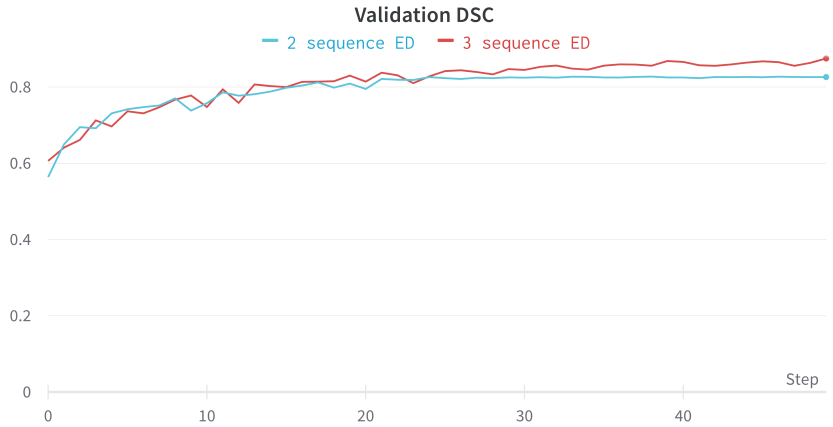


(a) Evolution of validation DSC

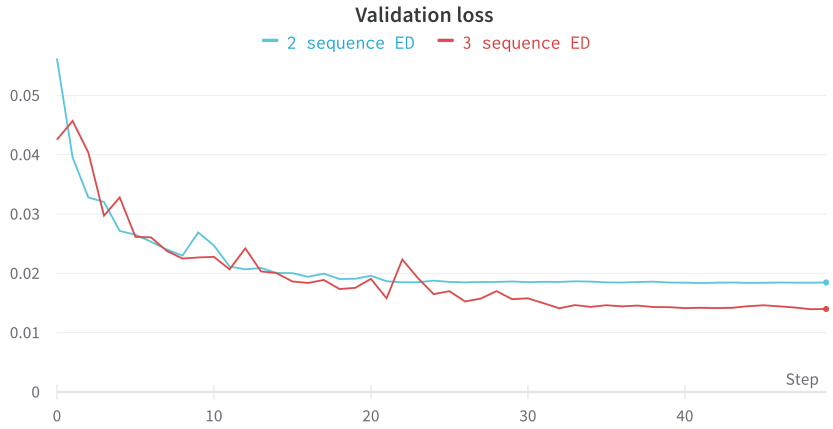


(b) Evolution of validation loss

Figure A.1: Comparison between validation DSC and loss for the models trained with three channels and three different learning rates, where the model with 0.001 LR achieved a validation DSC of 0.7286, the model with 0.0001 LR achieved 0.7842 and the model with 0.00001 LR achieved 0.5723.



(a) Evolution of validation DSC



(b) Evolution of validation loss

Figure A.2: Comparison between two sequence and three sequence training for the ED subregion, where the model trained for segmenting the ED subregion achieved a final validation DSC of 0.8748 compared to the 0.8260 achieved by the two sequence model.

Appendix B

Technical documentation

B.1 Requirements

For the implementation of our work we used the version 3.8 of the Python programming language. As for the most important libraries we used:

- tensorflow - version 2.7.0
- keras - version 2.7.0
- numpy - version 1.21.0
- nibabel - version 3.2.2
- matplotlib - version 3.5.0

All of the required libraries with their respective versions are present in the digital medium in file `requirements.txt`.

B.2 Source code

The source code is divided into multiple files, and their layout, as well as a short description for each one, can be found in the Appendix C.

One of the more important files is the `train.py`, which incorporates the training of the models. The main part of the file is a loop which defines training parameters for each subregion. The main parts of the loop can be found in the listing below:

```
if config['num_classes'] == 4: subregions = [0]
else: subregions = [1, 2, 3]

for subregion in subregions:
    # setting the training and valitation generators
    train_img_datagen = BratsGen(...)
    val_img_datagen = BratsGen(...)

    # defining and compiling the model
    model = unet(
        img_height=128, img_width=128, img_depth=128,
        img_channels=config["img_channels"],
        num_classes=config["num_classes"]
    )
    model.compile(optimizer=optim, loss="categorical_crossentropy",
        metrics=metrics)

    # training the model
    history = model.fit(
        train_img_datagen,
        steps_per_epoch=steps_per_epoch,
        epochs=config["epochs"],
        verbose=1,
        callbacks=[get_callbacks(...), WandbCallback()],
        validation_data=val_img_datagen,
        validation_steps=val_steps_per_epoch
    )
```

Listing B.1: The main loop in train.py

The idea behind the loop was to make the training of multiple models for individual subregions seamless, so when we are segmenting all subregions at once, the loop will only do one iteration. For the training of the models, we used a method `model.fit()` provided by TensorFlow.

Another important part are the scripts used for evaluation in the `evaluation/` directory. Both `evaluation.py` and `evaluation_separate.py` are very similar. The main part is the `model_eval(my_model, testing_dataset)` method that runs the evaluation of the specified model, where we make use of the `model.predict(test_image` method provided by TensorFlow, which generates a prediction, or in our case, a segmentation mask, for the given input image. Again we loop over the testing dataset and call this method for each image and calculate the metrics for the given segmentation. In the end, we print out the mean values for the metrics and save the individual slices as `.png` files for both the best five images and the worst five. The only difference in `evaluation_separate.py` is that before calling `model.predict(test_image)`, we need to merge the individual segmentations created by the models. To merge the segmentations, we use a method called `combine_models(model1, model2, model3, img)`, which generates individual segmentations and combines them into a newly merged segmentation, which is subsequently returned by the method.

Another files contains helper methods used by the mentioned files and are not supposed to be executed. One type of these files are utility files in the `utils/` directory, which includes methods for data processing and data manipulation (`data_processing.py`) and custom callbacks (`callbacks.py`). Then our implementation also uses a custom data generator defined as a class in `generator.py`. The function of the custom data generator is to provide batches of preprocessed and optionally augmented data to the training algorithm. Another helper file

is a `model.py`, which defines a TensorFlow model with U-Net architecture, later used in training. Furthermore, the last files used in our implementation are the `losses.py`, which defines our custom loss functions and `metrics.py`, which defines custom metrics.

B.3 Usage

The training and evaluation scripts were built for the Azure ML services, so the workflow outsite the Azure services might feel a little sluggish and was not properly tested. For the execution of the scripts it is required to have the mentioned requirements. The best way is to set up a virtual environment with:

```
python3 -m venv venv
```

and activating it with:

```
source venv/bin/activate
```

After activating the enviroment, you can install the requirements with:

```
pip install -r requirements.txt
```

The individual scripts can be run by the `python` command as follows:

```
python train.py
```

You can also provide a path to your dataset with the `--data_path` argument. The dataset has to include a `train/` and `val/` directories for training and `test/` for validation. The scripts will use the provided sample files if the data path is not specified. By default, the scripts use the predefined configuration that would need to be changed in the source code. For the training script, the configuration is in the `config` dictionary. The evaluation scripts would run the best respective models provided in the `models/` directory. In order to use different models, you would need to change the configuration variables on the top of the evaluation files.

Appendix C

Contents of the digital medium

Registration number of the thesis in the information system: FIIT-101018-97014

Contents of the digital medium (ZIP archive):

```
src/ - source code for the implementation
  evaluation/ - scripts used for evaluating models
    evaluation.py - for models segmenting whole tumour
    evaluation_separate.py - for models segmenting individual sub.
  utils/ - auxiliary methods used during training or evaluation
    callbacks.py - defines custom callbacks
    data_processing.py - methods used for data processing
    utils.py - miscellaneous methods
  generator.py - custom data generator
  losses.py - defines custom loss functions
  metrics.py - defines custom metrics
  model.py - defines U-Net model used in training
  train.py - script used for training
```

```
requirements.txt - files containing a list of required libraries
models/ - best models from each experiment
  separate/ - contains models segmenting individual subregions
    3ch/
      model_NCR.h5 - model segmenting NCR subregion
      model_ED.h5 - model segmenting ED subregion
      model_ET.h5 - model segmenting ET subregion
  model_3ch_aug_e4.h5 - model segmenting all subregions
BraTS2021/ - contains sample MRI scans that we used in our work
  train/ - sample training scans
    BraTS2021_00002/
    BraTS2021_00003/
  val/ - sample validation scans
    BraTS2021_00019/
    BraTS2021_00006/
  test/ - sample testing scans
    BraTS2021_01621/
    BraTS2021_01627/
docs/ - pdf versions of the final work
  main_part.pdf - main part of the work without the appendices
  appendices.pdf - only the appendices
  final.pdf - both the main part and appendices
```

We did not include the whole dataset used during our experiments because it has 13.4 GB. The original dataset can be downloaded from kaggle[1].

Name of the submitted archive: BP_JakubPovinec.zip.

[1] https://www.kaggle.com/datasets/dschettler8845/brats-2021-task1

Appendix D

Project task schedule

D.1 Winter semester

1^{st}-2^{nd} week	Specification of the domain and the dataset
3^{th}-4^{th} week	Studying basics of neural networks
5^{th}-6^{th} week	Exploring the dataset and experimenting with NNs
7^{th}-8^{th} week	Studying the state-of-the-art methods and related work
9^{th}-10^{th} week	Experimenting with U-Net and the dataset
11^{th}-12^{th} week	Working on the analytical part of the document

During the winter semester, our focus was primarily on studying convolutional neural networks and gaining experiences with neural networks in general. We managed to finish most of the analytical part of the work and successfully created our first model.

D.2 Summer semester

1st-2nd week	Defining the architecture and training first models
3th-4th week	Working on the first experiment
5th-6th week	Incorporating the Azure ML services into our workflow
7th-8th week	Working on the second experiment
9th-10th week	Evaluating the conducted experiments
11th-12th week	Finishing the document

In the summer semester, we deepened our knowledge of convolutional neural networks. We improved the model from winter and designed multiple methods for the segmentation of gliomas. Subsequently, we were able to finish and evaluate said methods, after which we focused primarily on finishing the document.