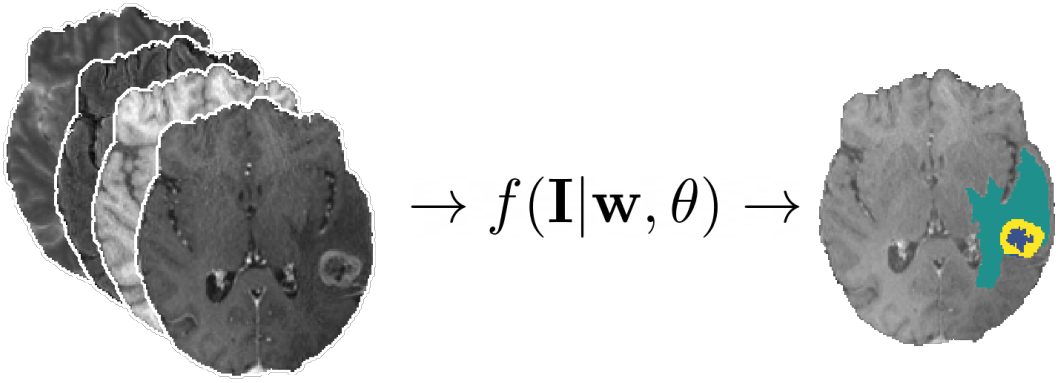
CHALMERS
UNIVERSITY OF TECHNOLOGY



Brain Tumor Segmentation Using Deep Learning

Master's thesis in Biomedical Engineering

Linus Lagergren and Carl Rosengren

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Brain Tumor Segmentation Using Deep Learning

Using deep neural networks to perform semantic segmentation of brain tumors in multi modal Magnetic Resonance Images

Linus Lagergren & Carl Rosengren

Supervisor and Examiner: Prof. Irene Yu-Hua Gu,
Dept. of Electrical Engineering, Chalmers Univ. of Technology

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Brain Tumor Segmentation Using Deep Learning
Using deep neural networks to perform semantic segmentation of brain tumors in multi modal Magnetic Resonance Images
Linus Lagergren Carl Rosengren

Supervisor Examiner: Prof. Irene Yu-Hua Gu, Dept. of Electrical Engineering, Chalmers Univ. of Technology

Master's Thesis 2020
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: MRI images $\mathbf{I}$ are given as input to the artificial neural network denoted f with weights $\mathbf{w}$ and biases θ which outputs a brain tumor segmentation.

Typeset in LaTeX
Printed by Tecknolog Tryck
Gothenburg, Sweden 2020

Brain Tumor Segmentation Using Deep Learning
Using deep neural networks to perform semantic segmentation of brain tumors in multi-modal Magnetic Resonance Images
Linus Lagergren Carl Rosengren
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Brain cancer, in particular Glioma, is a disease with a very low survival rate. The five-year relative survival rate for patients diagnosed with high grade Glioma is 3.6%. An important procedure for the medical clinician when diagnosing and planning treatment for patients is the segmentation of brain tumors into different classes. Magnetic resonance imaging is the imaging modality mainly used during clinical work with brain tumors. Deep learning methods have shown great results for brain tumor segmentation and other biomedical applications. This thesis compares two different deep learning architectures for semantic segmentation of brain tumors using fully convolutional networks. The architectures take inspiration from the well renowned U-Net structure, first used for semantic segmentation of cells. The first architecture takes a multimodal MRI-image as input in a Single-stream approach and the other architecture processes each modality separately in a Multi-stream approach. A number of experiments are conducted for testing and optimizing different regularization methods and hyperparameters. In the Multi-stream approach training is implemented in two ways. First in an end-to-end manner and secondly in a sequential manner. From the segmentation results the volume is estimated and visualized in 3D. The results of the experiments in this thesis suggests that the Multi-stream approach trained in an end-to-end manner outperforms the other approaches.

Keywords: deep learning, brain tumors, MRI, U-Net, convolutional neural networks

Acknowledgements

First of all we would like to thank our examiner Irene Gu for all the support during the project. It has been a treat to work with this project on brain tumors and we hope see this work continued in the future so that it in a not so distant future can be implemented into the clinical work.

Secondly we would like to thank MedTech West for providing us with an office space and being very inclusive and inviting us to different seminars and lectures.

Thirdly we would like to thank Asgeir Jakkola and Tomáz Gómez Vecchio for providing us with clinical insights and data from Sahlgrenska. Even though the project did not go as far as using this data for training any models.

Lastly thanks to family and friends for providing support during hard times in this project and also in general through out the not so short time here at Chalmers. Without your support we would probably not have made it this far.

Linus Lagergren, Carl Rosengren, Gothenburg, January 2020

Contents

Abbreviations

Artificial Intelligence - **AI**

Artificial Neural Network - **ANN**

Rectified Linear Unit - **ReLU**

Convolutional Neural Networks - **CNN**

Fully Convolutional Network - **FCN**

Magnetic Resonance Imaging - **MRI**

Multi Layer Perceptron - **MLP**

Computed Tomography - **CT**

Gray Matter - **GM**

White Matter - **WM**

Cerebrospinal Fluid - **CSF**

Dice Score - **DSC**

List of Figures

List of Tables

1
Introduction

1.1 Background

Cancer is a disease that each year accounts for 9 million deaths globally which is 22% of all the chronic diseases and is second largest after cardiovascular which accounts for 17.9 million deaths each year according the World Health Organization [1]. The uncontrollable division of cells called cancer forms a tumor which is a mass of cells in the body that is abnormal, a tumor can be either benign or malignant. Benign tumors grow slowly and rarely spread to surrounding tissue, they can however grow to push on tissue or nerves hence causing problem. Malignant tumors vary in growth rate and are capable of invading healthy tissue.

In the United States 1.4% of all new cancer diagnosis each year are primary malignant brain tumors. Brain tumors do however account for 2.7% of all cancer deaths in the United States [2]. Malignant primary brain tumors rarely spread beyond the brain and spinal cord [3]. A Primary brain tumor is a tumor that originates in the brain, the most common primary malignant brain tumor is named glioma. There are also secondary brain tumors, a secondary brain tumor starts somewhere else in the body and travels to the brain. Secondary brain tumors are more common and often spread from other tissues such as lung, skin, breast, kidney and colon.

In the diagnosis of cancer there are different imaging modalities that are used, the one most commonly used in brain tumor diagnosis is Magnetic resonance imaging(MRI) which provides detailed images of the brain. In MRI different radio frequency sequences are used to flip the hydrogen nuclei which emits a radio frequency when misaligned with the magnetic field. The emitted radio frequency are measured and transformed into an image. The rate at which the hydrogen nuclei returns to equilibrium is different in different tissues and this gives rise to contrast differences in the produced image. There are often multiple radio frequency sequences used for diagnosis to produce different contrast weighted images to get a better basis for diagnosis, this however generates data that the doctor needs to examine for a diagnose. Unlike x-ray and computed tomography(CT) the voxel values of the MR image is not standardized meaning that different acquisition protocols leads to the same cells and tissues having different gray scale intensities when pictured.

A healthy brain commonly contain three types of tissues namely gray matter(GM), white matter(WM) and cerebrospinal fluid(CSF). A brain tumor is tissue that is

abnormal to that of the healthy brain, the location, size and texture has a large variance which makes the localisation of a brain tumor a complex task. In the examination of MR images doctors segment out the tumor to compare it to the results of a segmentation done at another time instance or another doctor, the segmentation can contain multiple tumor classes. The manual segmentation of brain tumors is a time consuming task and a variance in segmentation's of the same tumor can be seen between doctors. Research is being conducted to automate the process of brain tumor segmentation to reduce the time for segmentation and lessen the bias and variance introduced by different doctors performing manual segmentations [4], [5]. An automated process would yield faster segmentation results and results would not differ as much between hospitals with different resources and would hence lead to a more consistent diagnosis of brain tumors.

Artificial neural networks(ANN) and in particular convolutional neural networks has been proven to outperform humans in the task of image classification and segmentation, an example is in the classification of melanoma [6]. In 2015 Ronneberger et.al published a network for segmentation of biomedical images named U-Net[7]. This method showed outstanding results in the ISBI challenge out performing all other competing network structures. The U-Net structure has since then been implemented in many different fields for segmentation one being brain tumor segmentation where it has shown promising results increasing the performance each year in the BraTS challenge [8], [9], [10].

1.2 Aim and scope

The aim of this thesis will be to develop and implement an Artificial Neural Network which performs automatic segmentation of brain tumors. Experiments will be conducted to observe the performance of different network architectures. To achieve this the fundamental building blocks of the U-Net structure will be unraveled and implemented.

1.3 Limitations

No attempts will made to complete a fully functional application with a graphical user interface. The project will rather focus on writing code for training an artificial neural network to perform the task of brain tumor segmentation. There is a wide variety of segmentation networks to narrow the scope we will look at a U-Net structure using 2D convolutions. The computational resources are limited for this project and focus will hence be to implement a working solution for the computational resources available to us. The datasets on brain tumors used is in this project are limited to the semi public MICCAI BraTS dataset.

1.4 Outline of thesis

Chapter 2 Introduces the theory needed to understand the methods used in the project. The chapter looks at labeling and segmentation of brain tumors and the fundamental building blocks of a convolutional neural network. It then moves on to study the U-Net structure and how these building blocks are combined for segmentation purposes.

Chapter 3 describes the methods used. Firstly it looks the dataset that was used and the image preprocessing steps on MRI brain scans for deep learning. Different software and python libraries that are used in the project to implement the segmentation network are then looked upon. It then moves on to describe the the network architecture and experimental design.

Chapter 4 presents the results from the experiments presented in the methods chapter. A comparison between the Single-stream U-Net and the Multi-stream U-Net trained in two different ways is done one the test dataset. Volume estimation and 3D vizualisation will also be shown for the network architectures

In **Chapter 5** the methods and results are discussed together with with future work that can be done to gain further knowledge and improve results. Issues that were stumbled upon during the thesis are also discussed.

Chapter 6 summarizes the findings of the work accomplished in a conclusion.

2
Theory

This chapter aims to give the necessary theoretical knowledge to understand the content of the report. The task of segmentation is introduced together with the fundamental building blocks of a segmentation network. This chapter also presents metrics for evaluating the networks and previous work in the field.

2.1 Semantic segmentation

Semantic segmentation is the task of assigning a class label to each pixel in an image. Formally the task of semantic segmentation could be stated as finding the mapping between a real valued tensor with integer valued dimensions H, W and C to a non-negative integer valued tensor with integer valued dimensions H and W where the integer corresponds to a finite space of classes of the segmentation problem at hand. This can be mathematically formulated as in equation 2.1

$$\mathbb{R}^{H\times W\times C} \rightarrow {\mathbb{Z}_+}^{H\times W} \tag{2.1}$$

2.1.1 Segmentation of brain tumors

In segmentation of brain tumors labels are assigned to tissue that share same characteristics. It can coarsely be divided into tumor and non-tumor parts of the brain. Tissue within the tumor that share the same characteristics can then be further divided in to subcategories where each subcategory is given its own label. The labels are assigned to each MRI slice and are used for training and evaluation of the network. It is therefore of importance that these labels are correctly assigned to yield a good segmentation network. Often multiple experts are used in the labeling of data to ensure the quality of the labeling.

2.1.2 Brain tumor classes

The most frequent primary brain tumor is glioma which account for 30 percent of all brain tumors and 80 percent of all malignant brain tumors [11]. Glioma is a tumor that arises from the glial cells, two types of glioma are Astrocytoma and Oligondendroglioma. A primary brain tumor is a tumor that has started in the brain, whereas a secondary brain tumor is a tumor that has originated somewhere else and spread to the brain. Tumors are graded on a scale I to IV where grade I tumors are slow-growing tumors and grade IV are fast-growing tumors. A grade I tumor

is considered benign and a grade IV tumor as malignant. The tumor classification system was developed by the World Health Organization in 2007 and updated in 2016 [12]. In a clinical setting a more coarse scale is sometimes used for gliomas where grade I to II is refered to as Low Grade Glioma(LGG) and gliomas graded III to IV is refered to as High Grade Gliomas(HGG). The survival rate for people with glioma is about 33% according to the National Cancer Institute [13]. This is one of the lowest survival rates for cancer. But this type of tumor is classified into three different categories in the BraTS-dataset [8], which is the dataset used in this thesis and is described further in Section 3.1.1. Some tumor tissue types that are are often segmented are necrosis, edema and enhancing tumor, which are further described next.

Enhancing tumor

The **enhancing tumor** class is assigned to the part of the tumor where cell division is active. This part of the tumor is where the growth of the tumor takes place, the pace of growth determines whether the tumor is benign or malignant. A faster rate of cell division yields a faster rate of growth and is considered more malignant.

Necrosis

The **necrosis** class is assigned to the part of the tumor where the cells are dead. This part of the tumor has undergone uncontrolled cell death. This part of the tumor is most commonly at the center of the tumor.

Edema

The **edema** class is assigned to the part of the tumor where fluid has been built up. Edema is also known as fluid retention or swelling and is most commonly observed in the vicinity of the enhancing tumor and necrosis.

2.2 Deep learning

Deep learning is a sub field of machine learning which itself is a sub field of Artificial Intelligence(AI), a graphical representation is presented in Figure 2.1. Deep learning takes inspiration from how biological processing of information works. Biological brains can perform very complex tasks with ease and therefore building systems that try to mimic their behaviour is reasonable. More concretely deep learning is performing many layers of non-linear information processing in order to learn multiple levels of representation and abstraction. The goal of deep learning is to make sense of data such as images, sound, and text. Deep learning is also sometimes referred to as Artificial Neural Networks.

2.2.1 Supervised learning

The most common learning scheme in the field of machine learning is called supervised learning where one wants to learn the mapping from x to y in equation 2.2 by

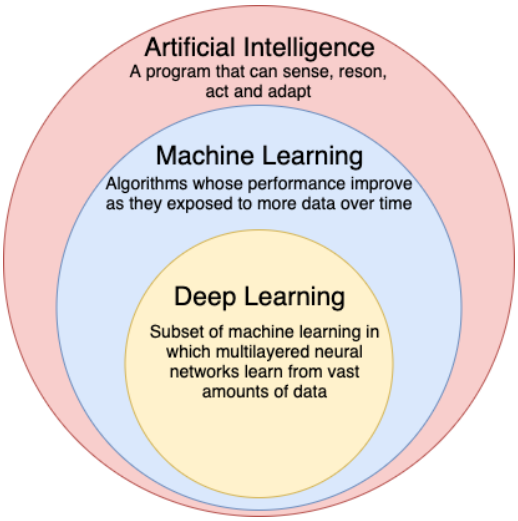


Figure 2.1: Picture of Deep Learning in the context of AI

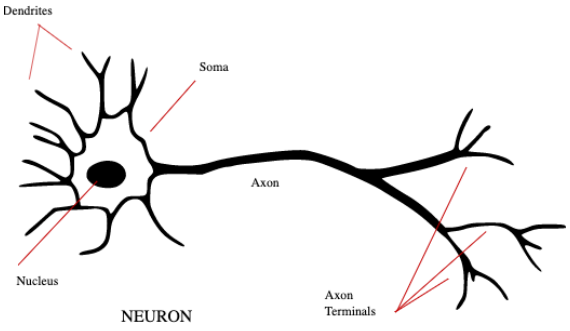
tuning the parameters $\mathbf{w}$.

$$f(x; \mathbf{w}) = y \tag{2.2}$$

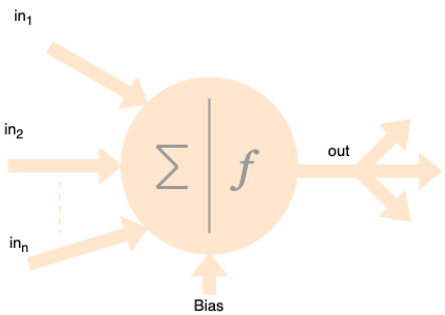
Where x in this thesis is a picture and y is the segmented picture where each pixel has been classified as belonging to a class and f is the neural network with weights and biases $\mathbf{w}$.

2.2.2 Artificial neurons

Artificial Neural Networks(ANN) basic building block is the **Artificial Neuron** which takes loose inspiration from how the biological neuron works. The biological neuron takes multiple inputs from the dendrites and outputs a single electrical impulse to the axon. This is however not a linear process. There is no output from the neuron until the input signals reaches the **threshold** and when the threshold is reached there is a relatively strong electrical impulse. This is known as an **activation** of the neuron.



(a) Illustration of a biological neuron licensed under CC BY-SA 4.0 [14]



(b) Illustration of mathematical neuron

Figure 2.2: Figure depicting the biological and mathematical neuron

This behaviour of the neuron can be modelled mathematically as

$$O = \sigma(\sum_{i=1}^{n} w_i x_i - \theta) \tag{2.3}$$

Where O denotes the output of the artificial neuron, w_i is how much input x_i should contribute to the output and θ is the threshold, which is also often referred to as the **bias**, and σ is a non-linear **activation function**. Visualizations of the biological and mathematical neurons are depicted in Figure 2.2.

2.2.3 Multi layer perceptron

The natural extension of modelling a single neuron is to model a neural network. The simplest kind of ANN is called a **Multi Layer Perceptron**(MLP). Which consists of many connected artificial neurons. This can be formulated mathematically as in equation 2.4

$$V_i^{(l)} = \sigma(b_i^{(l)}), \quad b_i^{(l)} = \sum_{j=1}^{N} w_{ij}^{(l)} V_j^{(l-1)} - \theta_i^{(l)} \tag{2.4}$$

Where $V_i^{(l)}$ denotes the value of neuron i in layer l, σ is the non-linear activation function, $b_i^{(l)}$ is the **local field** of neuron i, $w_{ij}^{(l)}$ is the weight that connects neuron i to neuron $V_j^{(l)}$ in the previous layer, $l-1$ and θ_i is the bias or threshold.

In this simple case the neurons are **fully connected** and **feed forward** meaning that a given neuron in layer l have connections to all neurons in layer $l-1$ and symmetrically each neuron in layer l is connected to all neurons in layer $l+1$, as an example in Figure 2.4 "Input layer" is $l = 1$, "Hidden layer" is $l = 2$ and "Output layer" is $l = 3$. An illustration of this can be seen in Figure 2.3. All layers that are not input or output layers are called hidden layers. That is because they are hidden from the interface of the network, i.e. one can only actively influence the data or the loss function.

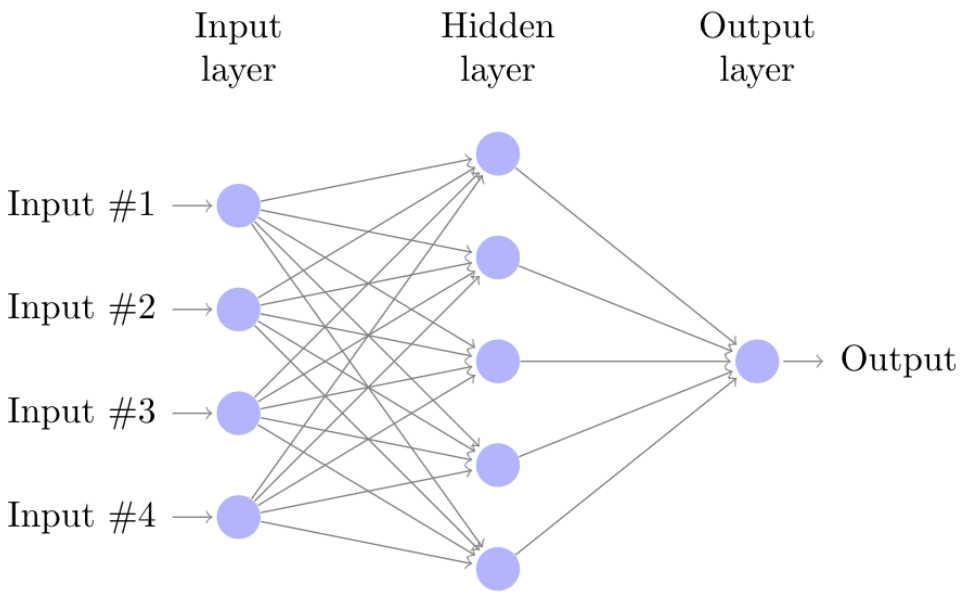


Figure 2.3: Picture of a fully connected feed forward MLP

2.2.4 Activation functions

An activation function is what adds non-linearity to the ANN. Without the non-linearity's an ANN could be reduced to a single affine transformation. So the activation function is where the modelling power of a ANN comes from.

Rectified linear unit

A commonly used activation function is the **Rectified Linear Unit**(ReLU). This activation function was proposed by Glorot et al [15] with motivation that it is qualitatively similar to modelling the dynamics of current through the cell membrane of the neuron cell. The formula for ReLU can be seen in equation 2.5. Where b_i denotes the local field seen in equation 2.4.

$$\sigma(b_i) = \max(0, b_i) \tag{2.5}$$

The choice of ReLU is due to the substantial empirical evidence of its learning enhancing effect [16]. The intuition for it being more effective is that it does not suffer saturation when feeding forward which the classical sigmoid and tanh activation functions does. ReLU also helps with the vanishing gradient problem when performing backpropagation which can be seen if one differentiates the expression in equation 2.5 which can be seen in equation 2.6. This is assuming that we can neglect the discontinuity in zero during numerical calculations. Since the gradient will be unity or zero everywhere it will be more stable during updates, this will be explained more in depth in Section 2.2.7 on vanishing gradients.

$$\sigma' = \begin{cases} 1 \text{ if } b_i > 0 \\ 0 \text{ if } b_i < 0 \end{cases} \tag{2.6}$$

Softmax layer

Softmax is a special type of activation function which often is used as the last layer of ANNs for classification tasks. As the name implies softmax is a soft version of the maximum function, i.e. the largest element in the real valued input vector of the function should tend to unity and all other elements to zero. But this should happen smoothly with non-discrete values. The mathematical formulation for this activation function can be seen in equation 2.7.

$$O_i = \frac{e^{\alpha b_i}}{\sum_j e^{\alpha b_j}} \tag{2.7}$$

Where b_i denotes the local field seen in equation 2.4. The parameter α determines how strong the maximisation effect of the function and is often set equal to unity. If $\alpha \to \infty$ one can see that $\text{softmax}(b_i) \to \max(b_i)$.
Softmax has three preferable properties:

1. $0 < \text{softmax}(b_i) < 1$, meaning that each element can be interpreted as a probability.
2. $\sum_i \text{softmax}(b_i) = 1$ which means that the output vector can be interpreted as a probability distribution.
3. $\text{softmax}(b_i)$ grows monotonously with b_i.

Property 1 and 2 is what makes softmax so popular for classification tasks since the output of a certain neuron can be interpreted as a probability of the class belonging to that neuron.

2.2.5 Loss function

A loss function is how one measures the error of an ANN and subsequently what the gradient will be when updating the parameters of the ANN. A loss function should be non-negative. Different problems require different loss functions, depending on whether the task is regression or classification etc. A mathematical definition of a loss function can be seen in equation 2.8, where y denotes ground truth and $\hat{y}$ denotes predictions of the model.

$$L(y, \hat{y}) \in \mathbb{R}_0^+ \tag{2.8}$$

Categorical Crossentropy Loss

Categorical cross-entropy is a loss function used for classification tasks. The loss function is presented in equation 2.9.

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{\text{C}} y_{ij} \log(\hat{y}_{ij}) \tag{2.9}$$

y denotes ground truth and $\hat{y}$ denotes the prediction. N represents the number of voxels and C the number of classes. Indices i and j denotes pixels and classes respectively. Since $\hat{y}_{ij} \in (0, 1)$, equation 2.9 is a well defined loss function but it also demands that it is used in conjunction with a softmax layer so that the output values can be interpreted as probabilities.

2.2.6 Backpropagation

Backpropagation is an algorithm which is used to tune the parameters of ANNs, it computes the gradient of the loss function with respect to the parameters, $\mathbf{w}$ of the network. The method is based on the classical optimization method gradient descent where one takes incremental steps in the opposite direction of the largest gradient of the loss function for a given number of training examples.
A schematic version of the update of parameters w can be seen in equations 2.10, 2.11

$$\mathbf{w}_{t+1} \coloneqq \mathbf{w}_t + \Delta \mathbf{w}_t \tag{2.10}$$

$$\Delta \mathbf{w}_t = -\eta \frac{\partial L}{\partial \mathbf{w}_t} \tag{2.11}$$

t denotes the current iteration of updates, η denotes the *learning rate* and L is the *loss function*. An example of a loss function can be seen in Section 2.2.5. The value of the loss function can give an indication of how good or bad the predictions of the model are.
Often one computes the error δ seen in equation 2.12 which then is used to compute the parameter updates in equation 2.13, where i and j denotes indices in a matrix and l denotes the current layer of the L layers in the network.

$$l \in \{L, ..., 2\}, \quad b_j^{(l)} = \sum_k w_{jk}^{(l)} V_k^{(l-1)} - \theta_j^{(l)}$$

$$\begin{aligned} \delta_i^L &\leftarrow \sigma'(b_i^{(L)}) L(y, \hat{y}) \\ \delta_j^{l-1} &\leftarrow \sum_i \delta_i^l w_{ij}^{(l)} \sigma'(b_j^{(l-1)}) \end{aligned} \tag{2.12}$$

$$\begin{aligned} w_{ij}^{(l)} &\leftarrow w_{ij}^{(l)} + \eta \delta_i^{(l)} V_j^{(l)}, \\ \theta_i^{(l)} &\leftarrow \theta_i^{(l)} - \eta \delta_i^{(l)} \end{aligned} \tag{2.13}$$

A more rigorous derivation of backpropagation can be found in Goodfellow et al [17] or in the original paper by Rumelhart et al [18].

2.2.7 Vanishing gradients

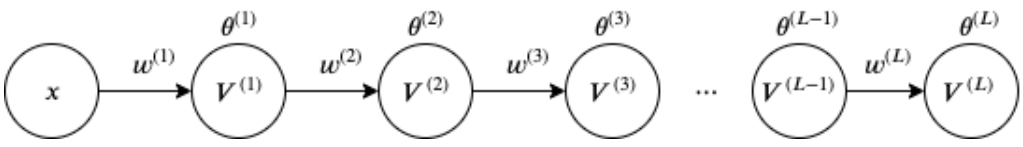


Figure 2.4: A schematic picture of a simple ANN used to illustrate how the problem of vanishing gradients appear

One major problem when training deep artificial neural networks is that long series of numbers are multiplied together when feeding data to a network and when propagating the error backwards. If these numbers aren't equal to unity the values will either explode or vanish. This is schematically illustrated by Mehlig [19] with a simple neural network, an equivalent network can be seen in Figure 2.4.
The feedforward procedure of this simple network is defined in equation 2.14

$$\begin{aligned} V^{(L)} &= \sigma(w^{(L)}V^{(L-1)} - \theta^{(L)}), \\ V^{(L-1)} &= \sigma(w^{(L-1)}V^{(L-2)} - \theta^{(L-1)}), \\ &\vdots \\ V^{(1)} &= \sigma(w^{(1)}x - \theta^{(1)}) \end{aligned} \tag{2.14}$$

When one starts to differentiate these expressions with regards to the neurons the last expression in equation 2.15 appears. This is closely related to the update rule of the weights.

$$\begin{aligned} \frac{\partial V^{(L)}}{\partial V^{(L-1)}} &= \sigma'(w^{(L)}V^{(L-1)} - \theta^{(L)})w^{(L)}, \\ \frac{\partial V^{(L)}}{\partial V^{(L-2)}} &= \sigma'(w^{(L-1)}V^{(L-2)} - \theta^{(L-1)})\sigma'(w^{(L)}V^{(L-1)} - \theta^{(L)})w^{(L)}, \\ &\vdots \\ \frac{\partial V^{(L)}}{\partial V^{(l)}} &= \prod_{i=L}^{l+1}[\sigma'(w^{(i)}V^{i} - \theta^{(i)})w^{(i)}] \end{aligned} \tag{2.15}$$

Noticing that there is some similarity between the product in equation 2.15 and in the weight updates in 2.13 one can end up with the expression.

$$\delta^l = L(y, \hat{y})\sigma'(b^{(L)}) \prod_{i=L}^{l+1}[w^i\sigma'(b^{i-1})] \tag{2.16}$$

And from equation 2.16 we can clearly see that if the terms in the product are smaller than unity as the number of layers, L, gets large the updates will tend to zero and symmetrically if the weights are large than unity the updates will tend to ∞.

2.2.8 Weight initialisation

One way to minimize the problem of vanishing gradients is to initialize weights in a structured manner. When using ReLU as an activation function He et al [20] has shown that it is theoretically justifiable to initialize weights according to equation 2.17. This method will here on be refereed to as he-normal.

$$w_l \sim \mathcal{N}(0, \sigma_{w_l}) \tag{2.17}$$

σ_l denotes the variance of the initial weights w_l in layer l of the ANN and n_l denotes the number of weights in layer l. σ_l is defined in equation 2.18

$$\sigma_{w_l} = \sqrt{\frac{2}{n_l}} \tag{2.18}$$

The intuition behind doing this weight initialization is that if the weights are initialized with constant variance the expected value in each neuron will depend on how many neurons there were in the previous layer. So if there were a lot of weights in the previous layer the variance will be small and the expected value of a local field in a given neuron will be closer to zero.

2.2.9 Convolutional neural network

A Convolutional Neural Network(CNN) introduced in 1999 by LeCun et al [21], is an ANN which is designed to make use of spatial information in input data. The conceptual idea is that the earlier layers recognize simple shapes and features and later layers can put together these simpler features and construct more complex representations. This idea is used to complete complex tasks in fields of computer vision and natural language processing for the task of image classification or speech-to-text.

The main idea behind a CNN is that one incrementally moves a **filter** over the input data. The filter is usually a square matrix. Each element in the filter contains a real number. During the convolution each element in the filter is multiplied with the corresponding element in a subset of the input. The output from a convolution is known as a **feature map**. A visualization of this procedure can be seen in Figure 2.5. The step size of the filter is called **stride** which controls the size of the output from the convolution. This can be expressed as the mathematical expression in equation 2.19. If one considers the case of images then V_{ijk} is the feature map where i and j can be thought of as enumerating the height and width of the feature map and k enumerates the number of filters in the convolutional layer. p and q denotes the rows and columns of the filter and r is the number of channels in the input. This expression assumes a stride of 1 in both spatial dimensions.

$$V_{ijk} = \sigma(\sum_{pqr} w_{pqkr} x_{p+i-1,q+j-1,r} - \theta_k) \tag{2.19}$$

Depending on what output size is needed from the convolution a **padding** can be used to shape the output by adding empty entries on the edges of the input to the convolution.
To further simplify the feature maps one can use **pooling** to filter the values in the feature map so that a down sampled feature map containing only the maximum values from a $n \times n$ neighbourhood in each pixel in the pooled feature map.

2.2.10 Z-score normalization

A common practice when working with ANNs is to center the values of the input around zero and make them have unit variance. This is done by using equation 2.20

$$Z = \frac{X - \mu}{\sigma} \tag{2.20}$$

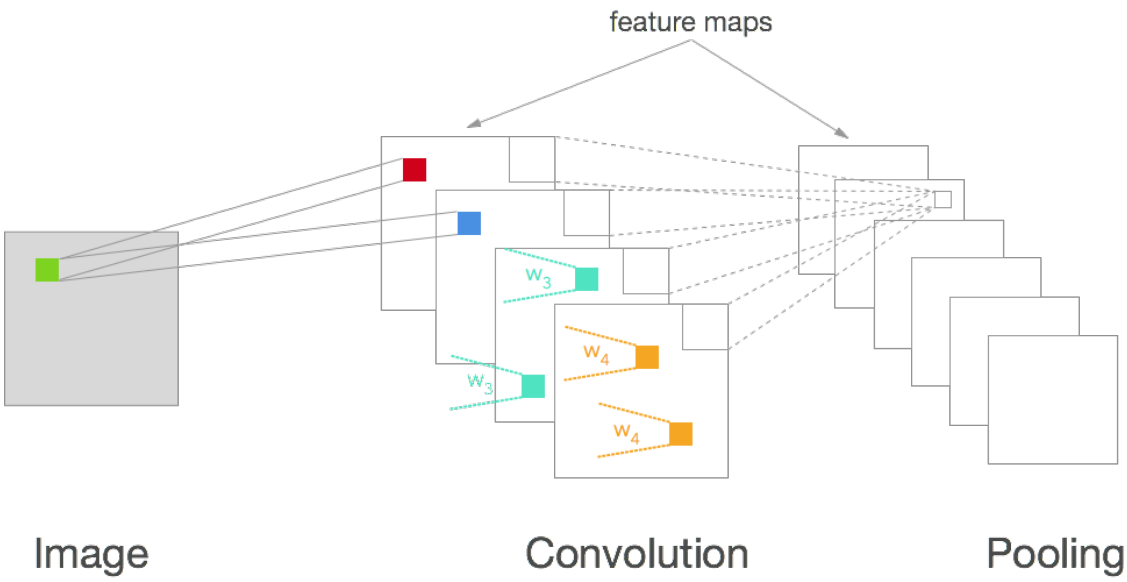


Figure 2.5: Conceptual picture of a CNN by Will Whitney licensed under CC BY-SA 4.0 [22]

X denotes a random variable, μ is the mean value of the random variable and σ is the standard deviation of X. Z is the standardized random variable.

2.3 Regularization

One big problem when training deep learning models is overfitting. Overfitting is when a model starts memorizing the training data instead of generalising. Memorizing the training data will make the model perform worse when making inference on out-of-sample data. A simple example of what overfitting is can be seen in Figure 2.7. The data generated by adding a noise to a second degree polynomial, then three different polynomials were fit to the data. From left to right there is a first degree polynomial, then a second degree polynomial and finally a ninth degree polynomial. One can see that the first and last polynomial underfits and overfits the data respectively. This poses a problem when evaluating models since the right model will have a better score in terms of for example *Mean Squared Error*, but it will probably perform worse on an out of sample data point.
Regularization is the general term for methods that aim to make models overfit less. Some of these methods will be briefly described in the next Section.

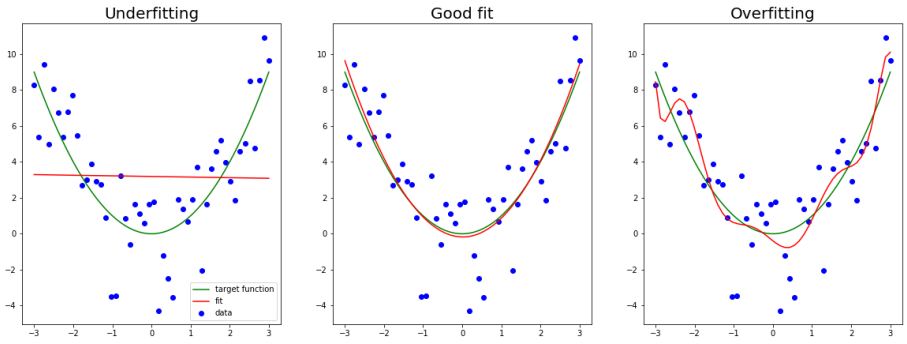


Figure 2.6: An illustration of how the number of parameters can influence the robustness of fit. From left to right there is a first degree polynomial fitted to the data then a second degree polynomial and finally a ninth degree polynomial

2.3.1 Early stopping

Early stopping is a method where training is stopped due to an increase in the loss function on the validation data set over a predefined number of *epochs*, this predefined number of epochs is referred to as *patience* in the software library Keras [23]. This method works because an increase in the loss function evaluated on the validation data hints at that the model has stopped generalizing and started memorizing the training data. At this point it is recommended to stop training the model. A visualisation of the typical behavior when training a model can be seen in fig 2.3

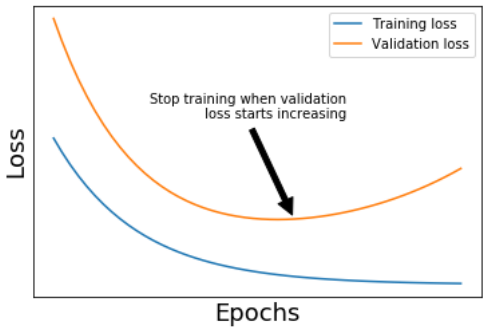


Figure 2.7: Figure depicting the archetypical behavior of training and validation loss. When validation loss starts increasing this usually means that the model has started to overfit

2.3.2 L2 regularization

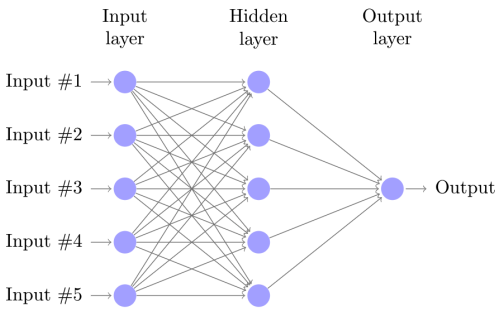
L2 regularization, also known as weight decay in deep learning, is a general regularisation method where the model is penalized by the squared norm of the parameters. This is realized adding a penalization term to the loss function L yielding the new loss function $\tilde{L}$ in equation 2.21 where w denotes the the weights and biases.

$$\tilde{L} = L + \lambda ||w||^2 \tag{2.21}$$

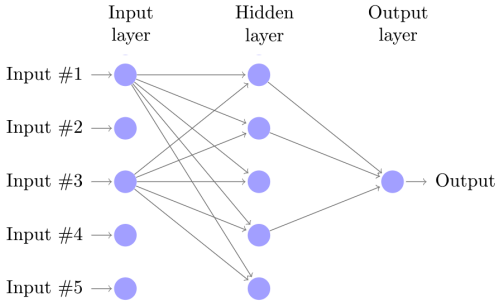
The intuition for this is that one wants to minimize the cost function and the square of the norm of the weights will favor smaller weights. A more rigorous derivation can be found in the book by Goodfellow et al [17]

2.3.3 Dropout

Dropout is a method where each neuron is dropped randomly with a certain probability p at each forward pass through the network during training. This can be viewed as training an ensemble of network architectures, see Figure 2.8. This forces the network to learn different pathways through the network in which information can flow. Dropout often makes the network less prone to overfit because neurons cannot co-adapt as much since there is no guarantee that a given neuron will be in the network at a given forward pass.



(a) Illustration of network with no dropout implemented



(b) Illustration of network where dropout has been implemented

Figure 2.8: In this figure one can see how using dropout makes the network take on a different topological structure

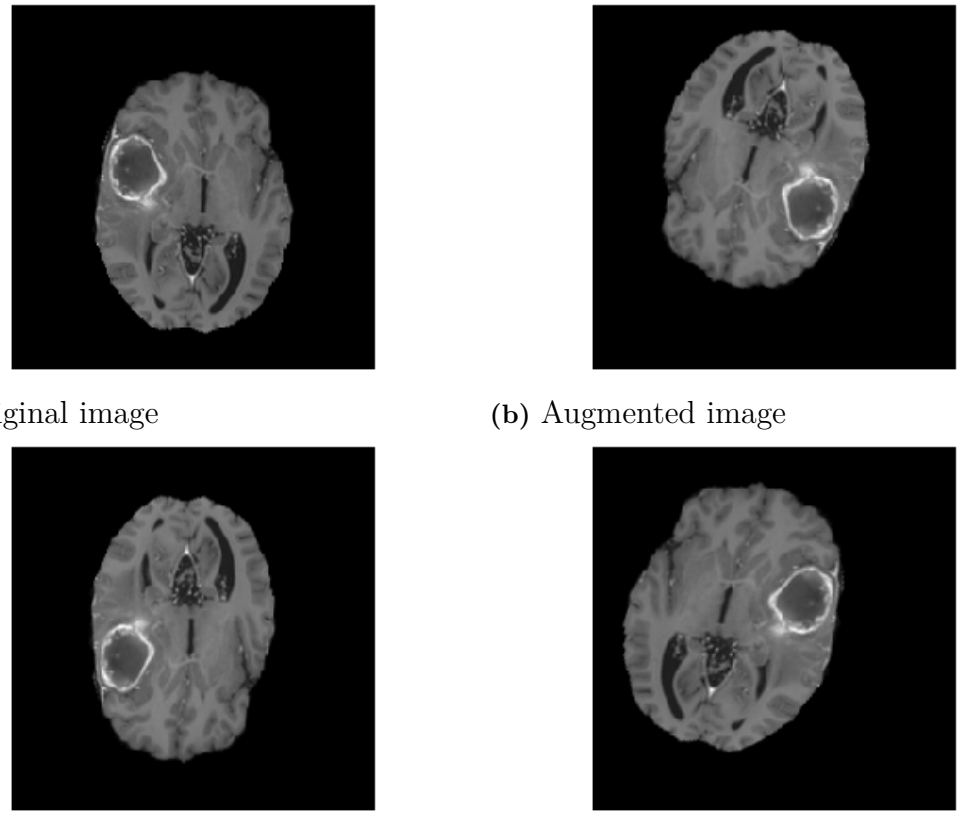
2.3.4 Data augmentation

Data augmentation is a regularisation method where the input data of the network is transformed in different ways. This is done to enlarge the number of training example and reduce overfitting. An example of data augmentation can be seen in Figure 2.9 where a skull stripped MR image of the brain has been translated, flipped and rotated in various different combinations.

A short list of examples of data augmentation

- Adding noise to data
- Shearing transformations
- Rotational transformations
- Horizontal and vertical flips
- Zooming

The main idea is that one in a synthetic manner generates more data. So that the effective size of the dataset gets larger thereby reducing the risk of overfitting.



(a) Original image **(b)** Augmented image

(c) Augmented image **(d)** Augmented image

Figure 2.9: An example of data augmentation on a picture of a brain from the MICCAI BraTS dataset [8]

2.4 Optimizer

A popular optimizer which is widely used is **ADAM** which is short for adaptive learning rate optimization algorithm which was proposed in a paper by Kingma and Ba in 2014 [24]. This algorithm is a flavor of stochastic gradient descent. But it uses moving averages of the first 2 moments of the gradients in the parameter updates to make the updates more stable. The equations for these updates and their approximations can be seen in equation 2.22 through 2.26.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.22}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.23}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.24}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.25}$$

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.26}$$

m_t and v_t denotes the moving averages of the first and second moment of the gradients respectively. β_1 and β_2 are hyper parameters which decide how much the gradients in the previous time step should influence the gradient updates in the current time step. To estimate these statistics one uses the formula for the unbiased estimators $\hat{m}_t$ and $\hat{v}_t$ which then can be used to update the weights of the ANN. η denotes the learning rate and ϵ is a small constant used to ensure that one never divides by zero.

2.5 Evaluation metrics

Evaluation of the segmentation network is an essential part in finding a network that performs the task of segmentation well. The performance can be measured with different metrics but is essentially a measure of how well the output of the network corresponds ground truth. Evaluation metrics that are commonly used in the task of segmentation are the accuracy and the dice coefficient. These are introduced below together with the adaptions of the dice coefficient to fit the task of brain tumor segmentation.

2.5.1 Accuracy

One of the simplest evaluation metrics for classification tasks is **accuracy**, which is simply the ratio of correctly classified samples and total number of samples. This is formulated mathematically in equation 2.27.

$$\text{Accuracy} = \frac{\text{Correctly classified samples}}{\text{Total number of samples}} \tag{2.27}$$

But this metric can however be hard to interpret how well the model is performing in cases where there is class imbalance. Which is the case in this thesis, which will be covered in Section 3.1.3.

2.5.2 Dice coefficient

The Dice Coefficient is a measure of spatial overlap that has a value between 0 and 1 were a result of 1 means total overlap between the prediction and ground truth and a value of 0 meaning no overlap. The metric is calculated by equation 2.28 where $\hat{y}_i$ and y_i is the class prediction and ground truth of pixel i respectively.

$$D = \frac{2 \times \sum_i^N \hat{y}_i y_i}{\sum_i^N \hat{y}_i^2 + \sum_i^N y_i^2} \tag{2.28}$$

In the evaluation of the MICCAI BraTS dataset a few problem specific dice coefficients are introduced to evaluate how well the network performs. This is done by computing dice scores between a few combined classes. The combined classes are enhancing tumor, tumor core and whole tumor, which are described further below

Enhancing tumor dice coefficient The dice coefficient of the predictions of the enhancing tumor class. An example of this can be seen in the upper right plot in Figure 2.10

Tumor core dice coefficient The tumor core is the combination of enhancing tumor and necrosis. An example of this can be seen in the bottom left plot in Figure 2.10

Whole tumor dice coefficient Whole tumor is specified as everything that is not labeled as background, i.e. enhancing tumor, necrosis and edema. An example of this can be seen in the bottom right Figure 2.10

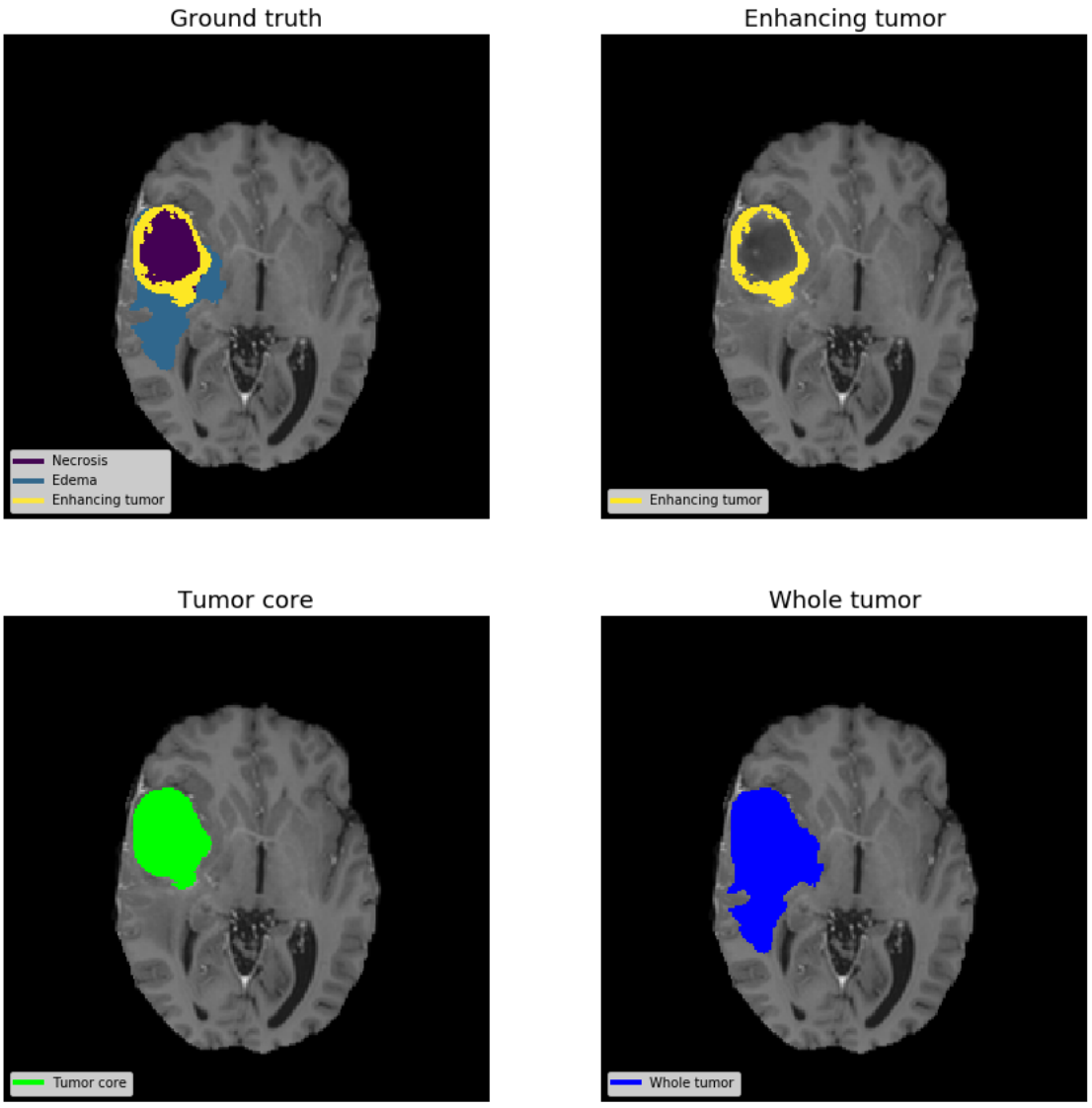


Figure 2.10: This figure visualises the different classes that are used for evaluating the performance of the network. Picture of the brain is from the BraTS dataset [8]

2.5.3 Confusion Matrix

A confusion matrix is a way to present the predictions of a model on a data set in a way that shows the performance of the model with regards to false positives and false negatives. An example of a confusion matrix can be seen in Table 2.1. Presenting the predictions in this way is useful to analyse the performance of the model. It also provides quantities to calculate other metrics such as F-score, sensitivity and specificity etc.

		True	
		Positive	**Negative**
Predicted	**Positive**	True positive	False positive
	Negative	False negative	True negative

Table 2.1: Example of a confusion matrix

2.6 Previous works

2.6.1 Sliding window approach

The earlist ANN approaches to segmentation tasks was performed using regular CNNs where each pixel was its own classification task. So one would have a *sliding window* where a patch of the full input image was given as input and the central pixel of this patch was predicted. Then in the next step the sliding window was moved one pixel and then a new patch was fed to the network. An example of this type of architecture was suggested by Havaei et al [5]. The architecture can be seen in Figure 2.11. The input to the network is a $4 \times 33 \times 33$ tensor where a number of convolutions are performed and then the feature maps are flattened and a softmax layer is used to perform classification for the central pixel of the given input patch.

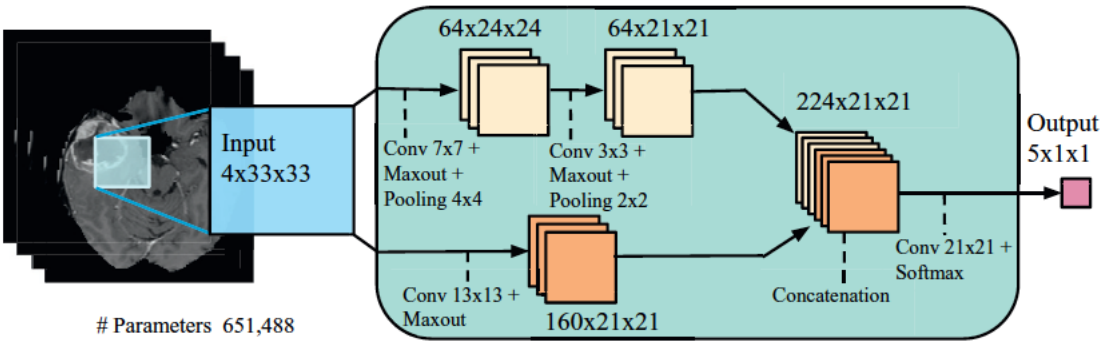


Figure 2.11: Picture of an architecture using a sliding window approached from the paper by Havaei et al [5]

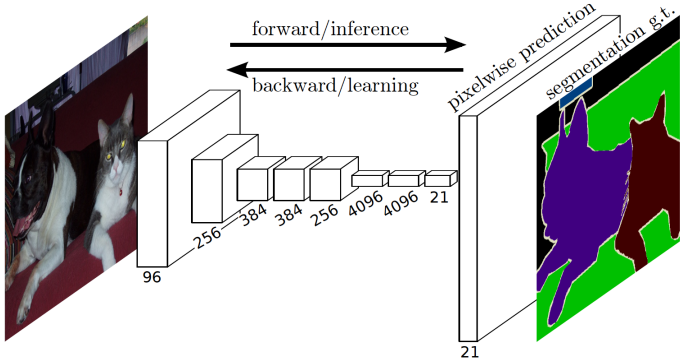


Figure 2.12: Picture of architecture introduced by Long et al [25]

2.6.2 Fully convolutional network

In contrast to a MLP or a classical CNN for classification tasks a fully convolutional neural network (FCN) contains only convolutional layers. The upsides of a CNN compared to a Multi Layer Perceptron is that it is less rigid when it comes to the size of the input, since the number of weights in the first layer of a multi layer perceptron is a direct function of the input size, where as the filters of a CNN has the same size and is independent to the size of the input.
The advantage of a FCN compared to a classical CNN is that the feature maps never have to be flattened and that fully connected layers are present which means that a FCN generally have less weights then a CNN. FCNs also makes it possible to output full images instead of just outputting a single pixel at a time. In the example of images this type of network can take any image size as input. The first FCN was introduced by Long et al[25]. The task was sematic segmentation where inputs of the network was a full picture and the target was the mask of the given picture.

2.6.3 U-Net

U-Net is a fully convolutional neural network (FCN) introduced by Ronneberger et al[7]. The network acts as a encoder-decoder. Which means that it takes an input and outputs an object of approximately the same shape as the input. In terms of semantic segmentation U-Net takes an $H \times W \times C_{in}$ input and outputs an $H \times W \times C_{out}$. If the input is an image then H is height, W is width, C_{in} is the number of channels of the input and C_{out} is the number of channels of the output. In the problem of semantic segmentation the C_{out} is equal to the number of unique classes among the labels.
The segmentation starts by 2 layers of convolutions followed by down sampling via max-pooling. This procedure is done 4 times and then followed by another set of convolutions. Then up sampling via transposed convolutions is performed on the last feature maps. These up-sampled feature maps are then concatenated with the feature maps of the convolutions before the last down sampling, these concatenations are represented by the gray lines in Figure 2.13. This concatenation is what made the U-Net-architecture novel compared to earlier works. This re-use of

earlier outputs helps with restoring the fine detail when up-sampling since without the skip connections "over" the down-sampling blocks this information might be lost in the smaller feature maps.

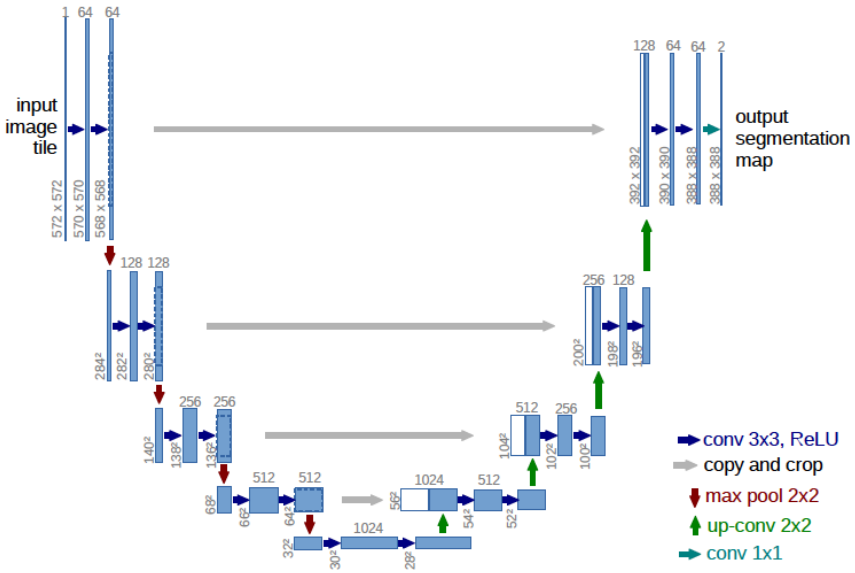


Figure 2.13: Picture of the architecture from introduced by Ronneberger et al[7]

3
Methods and data

This Section describes the methods used for brain tumor segmentation using deep learning. The chapter will look at preprocessing of the MR images, the dataset and software used, the different segmentation networks and their architecture. Lastly the chapter will look at the how the experiments for the different networks and regularization techniques is to be conducted.

3.1 Dataset

This Sections looks at the dataset that was used during the training, validation and testing of the brain tumor segmentation CNN.

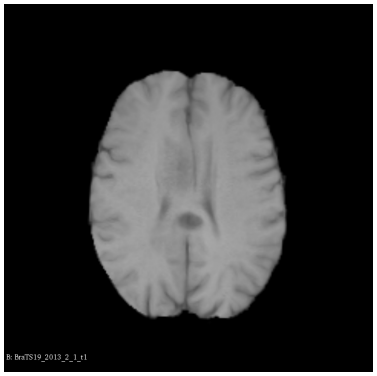
3.1.1 MICCAI BraTS data

The MICCAI BraTS training dataset from 2019 contains 335 patients with 4 different **modalities** which are T1, T2, T1CE and Flair. In table **??** it can be seen how different tissues are depicted in different modalities. All scans are in a NIfTI format [26] and have been acquired with different clinical protocols and with various scanners from 19 institutions. All images have been segmented manually by one to four raters all following the same annotation protocol and the annotations have been approved by experienced neuro-radiologists. The notations are of the enhancing tumor(label 4), the peritumoral edema(label 2), the necrotic and non-enhancing tumor core(label 1) and the 0 label is for everything else in the image. The data has been pre-processed with co-registration to the same anatomical template, interpolated to the same resolution($1mm^3$) and skull-stripped [8], [9], [10]. Each patient consists of 155 pictures here called **slices** of each **modality**.

The data set consists of both low grade glioma(LGG) and high grade glioma(HGG) patients, the distinction of the different tumor classifiactions have been made by experts in the field. In the data set there are 76 LGG and 259 HGG patients which together make up the whole data set of 335 patients.

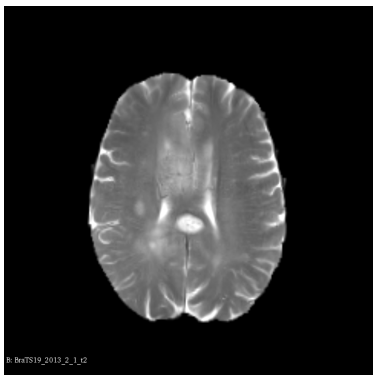
Tissue	T1	T2	Flair
CSF	Dark	Bright	Dark
White Matter	Light	Dark Gray	Dark Gray
Cortex	Gray	Light Gray	Light Gray
Fat(within bone marrow)	Bright	Light	Light
Inflammation(infection, demyelination)	Dark	Bright	Bright

Table 3.1: Table showing intensity for different brain tissue in respective modality [27]

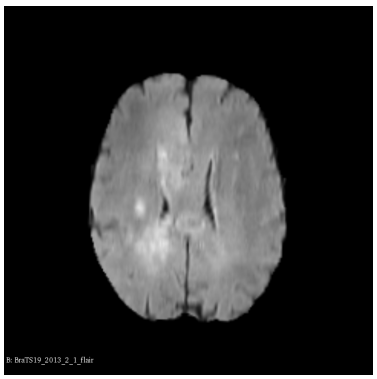


(a) T1

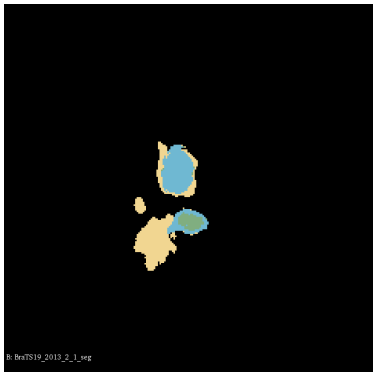
(b) T1CE



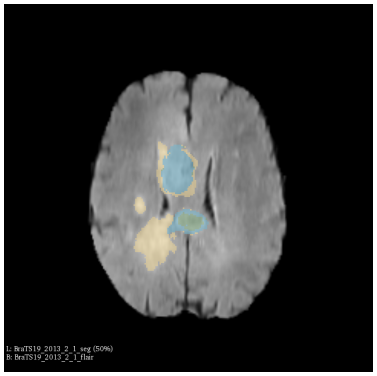
(c) T2



(d) Flair



(e) Segmentation



(f) Segmentation overlayed on Flair

Figure 3.1: Depicts the different MRI modalities, segmentation and overlay of the segmentation on the flair modality. In Figure e) and f) the colors yellow, blue and green corresponds to the edema, enhancing tumor and necrosis respectively

3.1.2 Training, testing, validation

The 335 patients are shuffled to mix the LGG and HGG patients. The shuffled data is then split into three groups according to Table 3.2, where the training, validation and test sets consists of 235, 50 and 50 patients respectively.

Data portion	Explanation	Chosen split	# of patients
Training data	The portion of the data which the model is trained with	70%	235
Validation data	A portion of the data which is used to peak at general performance during testing and can also be used for hyper-parameter optimization	15%	50
Test data	A portion of the data which is used to evaluate the model after training and hyper-parameter optimization	15%	50

Table 3.2: Table with explanation of data split and the chosen split sizes

3.1.3 Imbalanced data

The training data is quite imbalanced as can be seen by inspecting equation 3.1 and 3.2

$$\#\text{ of slices in training set} = 155 \times 235 = 36425, \tag{3.1}$$

$$\#\text{ of slices in training set containing tumor tissue} = 15539 \tag{3.2}$$

About 43% of the MR images in the training set contain pixels with tumor tissue. These MR images will be used for training. This reduces class imbalance and will help the network converge. In this subset of images containing tumor tissue, all pictures can be cropped from the original size of 240×240 to 176×176 without removing any brain pixels, an example of this cropping can be seen in Figure 3.3. This cropping is motivated by a brute force algorithm checking where the first and last pixel containing brain tissue is positioned for every sample in both horizontal and vertical direction of the brain image. The reduction in number of voxels is 46% and does not lead to any loss in information since all relevant information lies within the brain pixels. This filtering and cropping is beneficial for both class imbalance and computational time during training due to reduced image size.
To visualize the class imbalance further one can see the boxplots in Figure 3.2. From these boxplots it can be seen that the number of voxels with no tumor is in order 10^6

where as the pixels containing tumor related classes are of order 10^4. Even though all slices that contain no tumor class are removed and the images are cropped there still remains a quite large class imbalance.

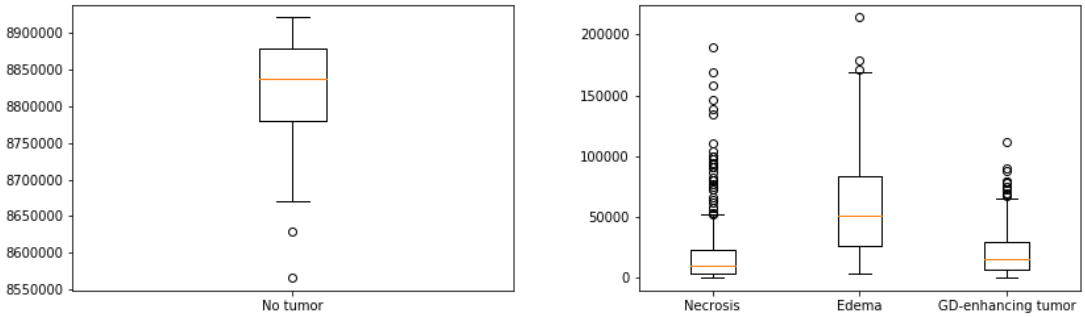
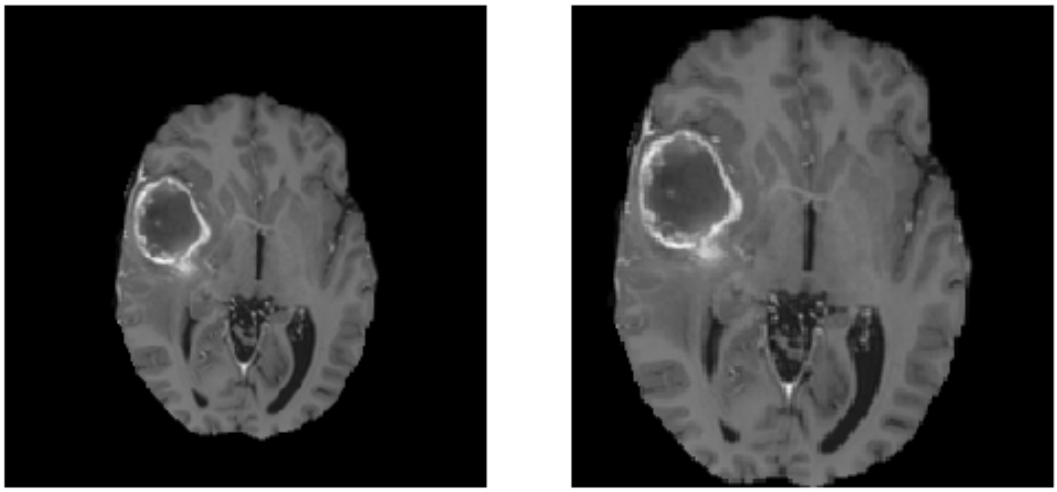


Figure 3.2: Boxplots to showcase the imbalance of the data. The class label is denoted on the x-axis and the number of voxels are represented by the y-axis.



(a) Image before cropping **(b)** Image after cropping

Figure 3.3: Images depicting the difference before and after cropping of the MR images

3.2 Software and libraries

3.2.1 TensorFlow

TensorFlow [28] was first made public by the google brain team in 2015 and is an open-source software library created for expressing machine learning algorithms and for execution of the algorithms. It was built to be able to scale, enabling running the computations on multiple CPUs and GPUs for faster calculations. TensorFlow is fast at computing matrix operations due to it being implemented in C++, it can

however be accessed and controlled by other languages such as python and Java. It has become one of the most used machine learning libraries due to its accessibility, ease of use and speed. There are other libraries that are capable of running on top of TensorFlow, one such program is Keras which will be introduced next.

3.2.2 Keras

Keras [23] is a high-level neural networks API which is written in python and capable of running on top of TensorFlow. It allows for easy and fast prototyping through user friendliness, modularity and extensibility. Keras runs on top of TensorFlow and is able to also run on multiple CPUs and GPUs allowing for scalability and speed.

3.2.3 3D Slicer

3D Slicer is an open source software that can be used for visualization and manipulation of medical images. The 3D slicer platform provides a large number of applications that are used for image pre- and postprocessing with different medical fields. The applications provided within 3D Slicer are maintained and developed by the user community. In this work we used 3D slicer for image preprocessing and for 2D and 3D visualizations of the tumor and brain structure.

3.3 Data pre-processing

There are several preprocessing steps that are required before a brain image can be further explored. The studies of MR brain images often require processing to isolate the brain from the extra-cranial or non-brain tissue which is often referred to as skull stripping. With skull stripping information is removed that does not add value when inspecting MR brain images for tumors and is irrelevant from an machine learning perspective. The brain regions should be skull-stripped before the application of other image processing algorithms such as co-registration to the same anatomicaltemplate and interpolation to the same resolution [29].

3.3.1 Skull stripping

When investigating skull stripping methods it is of importance to find methods with good accuracy to get the best data possible for further analysis. Moreover it is also important to find methods with speed since there is much data acquired from MR scans and processing should be made as efficient as possible to save both money and time which can lead to faster diagnosis and thereby also higher survival rates. There are broadly five different methods which are presented in literature, each skull striping method has its limitations and merits which together will be presented next.

Morphology-Based Methods, are methods that use the morphological erosion and dilation operations to separate the brain region from the skull. Such methods use the combination of edge detection and thresholding to find the region of interest(ROI). The limitations of these methods are that they depend on parameters such as size

and structural element shape and the values of these parameters directly influences the output of the method [29].

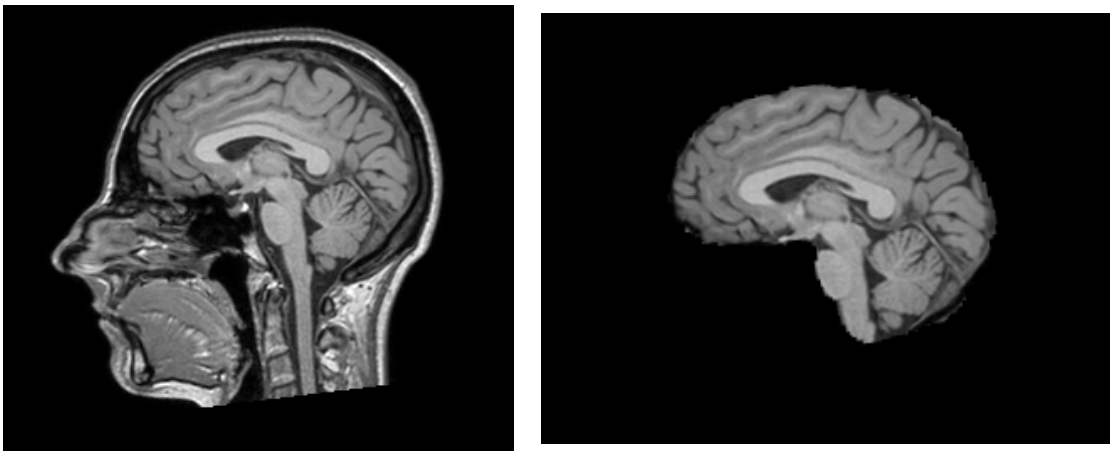
Intensity-Based Methods, use the pixel value intensities in the image to separate the non-brain and brain region. These methods rely on modeling the intensity distribution function to find the ROI. The drawbacks of these methods are that they are sensitive to intensity biases which are introduced by the MRI by nature such as noise, contrast variations and imaging artifacts [29].

Deformable Surface-Based Methods, the basic idea behind these methods are an active contour that moves under the influence of energy functional towards the desired object boundaries. An active contour model starts with an initial closed curve which is iteratively shrunk or expanded with respect to the boundary of the object which is in this case the brain. This expansion or shrinking of the boundary is referred to as the curve evolution. The deformable surface-based methods have in general a higher potential to produce robust and accurate results then the previous discussed methods [29].

Atlas or Template-Based Methods tries to fit an atlas or template on the MR brain image to extract the brain. The merits of this method is that it is able to extract the brain when the relation between pixel intensities and regions are not well defined in the brain image. These models differ from each other in how many templates or atlases are used to distinguish the brain regions and also in how they are applied to the MR brain image.

Hybrid Methods, are methods which combines earlier mentioned methods to increase accuracy by using merits from one or more methods to account for flaws in other methods.

As mentioned there are many different skull stripping methods, one method to extract the brain from an MR image is the ROBEX skull stripping which is a module that can be used in 3DSlicer to easily remove the skull in MRI images. The method is based on the non-parametric algorithm proposed by Iglesias et. al. [30]. The algorithm is automatic, runs fast and yields especially good results in T1 weighted images which can be seen in Figure 3.4. The automatic skull stripping is a practical alternative to manual skull stripping since it is more time efficient, it must how ever be noted that the results needs to be reviewed to ensure an accurate extraction of the brain. In subjects with lesions close to the non-brain tissue skull stripping methods are known to be less efficient due to smaller gradient differences between non-brain tissue and brain tissue therefore extra care should be taken when extracting the brain in these cases.



(a) Clinical MRI scan

(b) Skull stripped MRI scan

Figure 3.4: The dataset IXI was used together with the ROBEX skull stripping module in 3DSlicer. `http://brain-development.org/ixi-dataset/` under CC-BY-SA 3.0 license

3.3.2 Registration and co-registration

Registration is the process in which two or more images taken at different times, different viewpoints and by different sensors can be overlaid. In the diagnosis of brain related diseases the scan time is usually somewhere between 30 and 60 minutes it is common that the patient moves during the examination and this yields shifts in the acquired MRI images. With registration in the preprocessing step of the data this shift can be corrected for with manual or automatic registration methods. The automatic registration algorithms usually do a better job than manual registration but the result should be inspected upon execution.

Image registration can in general be divided into four main groups where the most common ones used in medical imaging is multitemporal analysis and multimodal analysis. Multitemporal analysis is where images of the same scene are acquired at different times. The aim of the registration in this case is to align images to evaluate changes in the scene which appear between acquisitions of the images, this can be used to observe changes in a disease over time. Multimodal analysis is where images are taken of the same scene by different sensors. The aim of this registration is to align information from multiple sources to gain a more detailed view of the scene. In the case of MRI different modalities are often used on the same subject and alignment of these imaging modalities yield a more detailed view of the subject.

There is not one universal registration method that is applicable on all registration tasks due to diversity and degradation of the images. There are however four steps broadly that the majority of registration methods follow. The first step is feature detection wherein which the algorithm finds features that are present in the images to be overlaid. The second step is feature matching where the previous detected features are matched. Third step is transform model estimation which is the process

where the registration algorithm finds a mapping function that can be used to align one image to the other. The last step is image resampling and transformation, this is where the mapping function is applied to one of the images to align it with the other image.

Two main approaches has been formed for the automation of image registration namely area-based and feature-based methods. The feature-based method extracts structures in the in the images such as lines or points. For this method to work well the structures should be distinct and spread over the whole image. The method does not work directly with the intensity variations in the image, in such cases the area-based method is a better approach. In medical images there is sometimes a lack in distinctive objects therefore an area-based method would then be the choice of registration method [31].

3.3.3 Spatial resolution

In the analysis of images in machine learning it is necessary to have the same image sizes of the images being used in the network for training, validation and testing. In case of MR images different machines and clinical protocols give rise to different sized images and it is hence necessary to resample the images in to the same size. The resizing of an image is a scaling operation which belongs under image registration and is done by interpolation methods. There are many different interpolation methods but they all aim at adding information by observing surrounding values to compute an informed guess. In a paper on medical image interpolation [32] that studies different interpolation methods show that when comparing the gray-value errors of the 84 filters evaluated the linear interpolation works best for 2 grid points and cubic convolution filter for 4 grid points. The cubic convolution shows a 28% - 75% reduction in grey-value errors. For larger filters using Welch, Cosine, Lanczos and Kaiser windowed sinc filters even better results where obtain with a reduction of 44% - 95% these methods do however yield heavier computations there by increasing computational time.

3.3.4 Normalizing pixel-values

In the specific case of the MICCAI BraTS dataset all pixels which has value 0 is ignored due to it not being brain tissue and using equation 2.20 all other pixel values are standardized.

3.4 Volume estimation and 3D visualization

By estimating the volume of the brain tumor at two different points in time a conclusion regarding the growth of the tumor can be made and a treatment plan can be set in place. A pixel in one of the images from the MICCAI BraTS dataset has the volume $1mm^3$ and is known as a voxel. All the images in the MICCAI BraTS dataset has been interpolated to have this resolution so by adding all the segmented

pixels together the volume can be calculated. Using 3D slicer the tumor volume can be depicted in 3D giving a good overview of the whole segmentation. Two or more tumor scans at different time instances could in practise be overlayed and in 3D slicer and the volumes could be compared to observe how the tumor has changed between the acquired scans. This will however not be observed in this thesis since the datset consist of data from one point in time.

3.5 Research question

The main question of this thesis will be comparing a conventional U-Net architecture, hereafter referred to as Single-stream U-Net, to a Multi-stream U-Net trained in two different ways. Both of which will be described in the next Section. The U-Net architecture by Ronneberger et. al [7], described in Section 2.6.3, will be used as a stepping stone for the architecture design. The Single-stream U-Net will firstly be evaluated then different techniques known to improve a generalization of the network will be implemented and tested. Thereafter the Multi-stream U-Net will be trained in two different ways. The performance between the two ways of training the Multi-sream U-Net and the Single-stream U-Net will be evaluated.

Design and hyper parameters

Activation function

ReLU described in equation 2.5 is used as activation function.

Convolutional block

A convolutional block in Figure 3.6 represents two convolutional layers following each other. The number of filters and transposed convolutions for each convolutional block is presented in table 3.3 and 3.4.

Padding

A small difference from the original U-Net is that padding is used to make the outputs from the convolutional blocks have the same height and width as the input.

Dropout

When dropout is used it is used on the two deepest convolutional blocks in the downsampling part of the network. A small dropout rate of 0.1 has been chosen.

Number of filters

Each convolutional layer in the convolutional blocks has a certain number of filters associated with them. These are specified in Table 3.3. The upsampling procedure

called transposed convolutions [33] also has a number of filters associated with them which are specified in Table 3.4.

Convolutional block	# of filters Single-stream U-Net	# of filters Multi-stream U-Net
1	64	32
2	128	64
3	256	128
4	512	256
5	1024	512
6	512	256
7	256	128
8	128	64
9	64	32

Table 3.3: Table enumerating the number of filters in each convolutional block of the U-Nets

Transposed convolution	# of filters Single-stream U-Net	# of filters Multi-stream U-Net
1	256	256
2	128	128
3	64	64
4	32	32

Table 3.4: Table enumerating the number of filters in each transposed convolution of the U-Nets

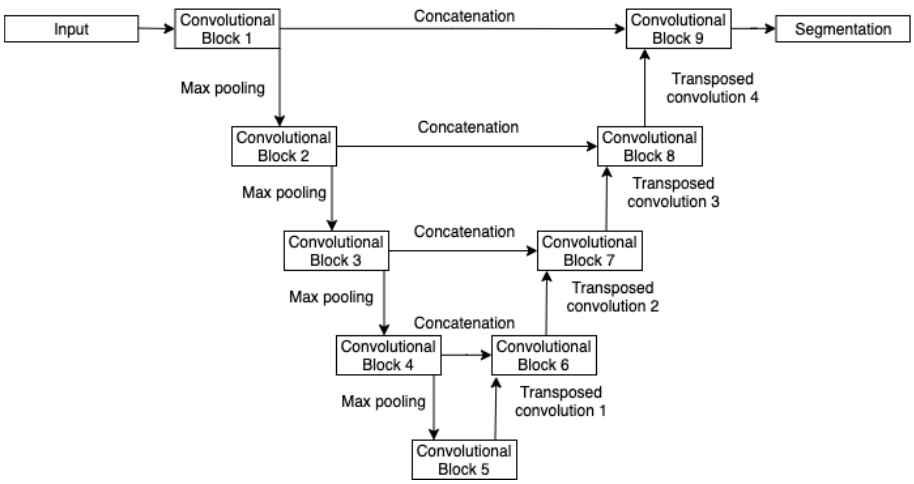


Figure 3.5: A schematic chart of the basic U-Net used for this project. A convolutional block in this image represents two convolutional layers following each other

3.6 Single stream U-Net

The Single-stream U-Net has been implemented as described in Figure 3.5 and 3.6. The network takes all modalities combined in 4 channels as inputs and outputs 4 channels each containing the probability of a pixel belonging to a class. The number of filters and transposed convolutions used in each convolutional block can be seen in Table 3.3 and 3.4 respectively. The part described as U-Net in Figure 3.6 is the U-Net depicted in Figure 3.5.

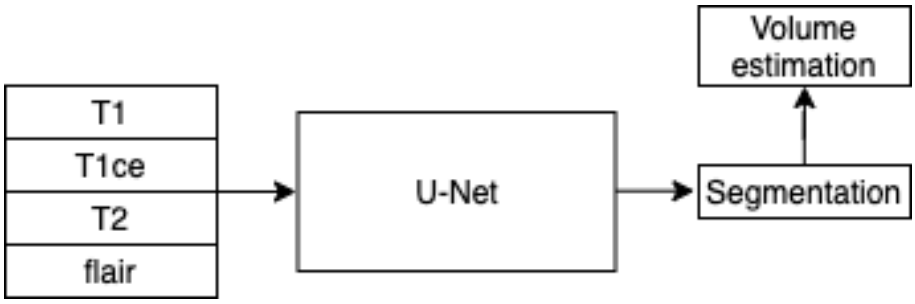


Figure 3.6: A schematic chart of the Single-stream U-Net used for this project.

3.7 Multi-stream U-Net

The novel architecture presented in this thesis is a Multi-stream U-Net. Which processes each modality separately before fusing the results of each modality. This is done using the U-Net architecture presented in Figure 3.5 with the number of filters and transposed convolutions presented in Table 3.3 and 3.4 respectively. This architecture differs to that of the Single-stream U-Net in the last 1×1 convolution and the softmax layer is omitted and the four $176 \times 176 \times 32$ outputs of the U-Nets are concatenated and then fed through a final convolutional block. An illustration of this can be seen in figure 3.7. The hypothesis is that as much independent information processing is done before merging the layers. All other hyperparameters are the same as in the Single-stream U-Net.

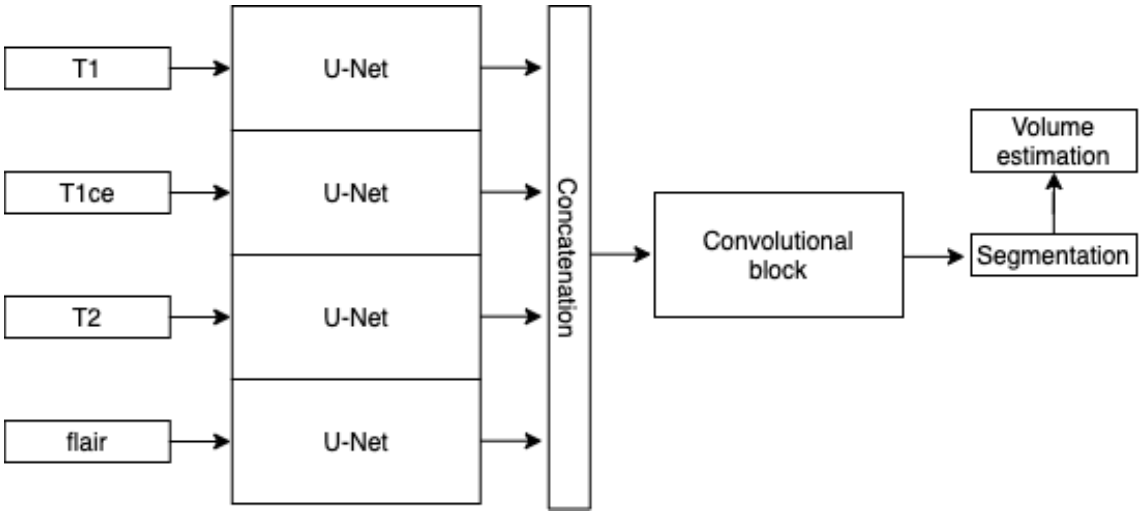


Figure 3.7: A flowchart of the Multi-stream U-Net. Where each of the four modalities are feed through a reduced U-Net before being merged via concatenation and fed through a final convolutional block.

3.8 Experimental design

Training setup

- The network architectures that will be experimented with are as described in Section 3.5
- The dataset used is the BraTS training dataset with cropping and tumor slice selection as described in Section 3.1.1 and 3.1.3
- Early stopping is used as described in Section 2.3.1 with a patience of ten
- The maximum number of epochs is set to 100
- The batch size is set to 16
- The loss function used is the categorical-crossentropy loss which is described in Section 2.2.5
- Accuracy is used as a measure of performance and implemented as described in Section 2.5.1

3.8.1 Single-stream U-Net experiments

The Single-stream U-Net as described in Section 3.6 is implemented and trained with a number of experiments observing regularization methods, learning rates and weight initialization which are described in this section. The training and validation loss and accuracy as a function of epochs will be presented for each of the Single-stream U-Net experiments. An evaluation on the test dataset will be presented in terms of a confusion matrix and the problem specific dice scores for the best performing experiment.

Learning rate

The first experiment will look at the effects of using different learning rates. The weight initialization used is described by the normal distribution in equation 3.3 where $\mu = 0$ and $\sigma = 0.01$ The learning rates that are investigated are presented in the Table 3.5.

$$w_{ij} \in \mathcal{N}(\mu, \sigma) \tag{3.3}$$

Learning rates (η)	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}

Table 3.5: Learning rates in experiment one

Weight initialization

In this experiment the weight initialization scheme using he-normal, described in Section 2.2.8, is used to look at the effects it has on the training of the network. The learning rate used is determined by the first experiment.

L2 regularization

L2 regularization is added to the previous experiment to look at how the training process is effected. It is implemented as described in Section 2.3.2 and initialized with a λ of 0.001

Dropout

This experiment looks at the effects off adding dropout to experiment on L2 regularization. The dropout is set to 10% at the end of the contracting path as described in the original U-Net paper by Ronneberger et. al [7].

Data augmentation

Data augmentation is implemented in this experiment to look at the effects it has on adding it on top of dropout. The data augmentation is used to vary existing data in order increase the effective dataset since there is a limited amount of data. The augmentations made to the dataset is as follows:

- Rotation with a variation of ±20 degrees from the original image
- Horizontal and vertical flip
- Shear transformation with a variation of shear angle of 0.2 degrees
- Zoom with a range variation of 10%

3.8.2 Multi-stream U-Net experiment

End-to-end training

The Multi-stream U-Net described in Section 3.7 will be implemented and trained with the regularization methods, learning rate and weight initialization which shows best performance in terms of training and validation loss and accuracy in the Single-stream U-Net case. The training will be conducted by training the whole network at once hence naming it end-to-end training. The training and validation loss and accuracy will be presented. An evaluation on the test dataset will also be made showing the confusion matrix and problem specific dice scores.

Sequential training experiment

The Multi-stream U-Net described in Section 3.7 will be implemented and trained with the same regularization methods, learning rate and weight initialization as used in the Multi-stream U-Net experiment trained end-to-end.

Four networks with the architecture described in Figure 3.5 are trained with one modality each until early stopping stops the training process. The weights of all the trained networks are then set as constant and a second training cycle with the addition of concatenation and convolutions as seen in the right hand side of Figure 3.7 is performed. Where the weights of the final convolutional block is further trained until early stopping.

The training and validation loss error and accuracy error from training each network on a modality will be presented. The modality that shows the smallest loss and accuracy error will be depicted together with the results from concatenating and training the last convolutional blocks, these graphs will show the training and validation loss and accuracy. This is done so that a fair comparison can be made between training one path and training the sensor fusion. An evaluation on the test dataset will also be made showing the confusion matrix and problem specific dice scores.

3.8.3 Performance evaluation

The test dataset which is described in Section 3.1.1 is used for a final evaluation of the three networks. The performance will be looked upon by comparing the confusion matrices, dice scores of whole tumor, tumor core and enhancing tumor. The final part of the performance evaluation will look at qualitative results of the networks segmentation's. In this part the segmentation results of the three networks will be depicted together with the ground truth.

3.8.4 Volume estimation and 3D visualization

The volume that has been chosen to investigate is the volume of the tumor core. The volume of the ground truth and the volume estimation made by each network

will be compared. A comparison will be made by observing the volume error which is the difference between the ground truth and and network estimated volumes. The volume error will be presented in histograms to observe if the networks over or under predicts the tumor volumes. To show the whole segmentation performance a 3D visualization will be made of one tumor for the best performing network. The 3D visualization will be shown from multiple different angles.

4
Results

This chapter presents the results obtained from the experiments described in methods chapter. The results are presented with graphs of the training and validation loss and accuracy for learning rate, weight initialization and regularization techniques in the Single-stream U-Net cased. For the Multi-stream U-Nets the learning rates regularization techniques used in the last experiment of Single-stream U-Net are implemented and the training and validation loss and accuracy is presented. A comparison of the performance of the networks is also presented as described in Section 3.8.3 performance comparison. The graphs depicted for the training of the networks are a function of epochs.

4.1 Single-stream U-Net

Learning rate

In Figure 4.1 and 4.2 the results from training the network with different learning rates are presented. Figure 4.1 present the difference between training and validation loss called loss error. In Figure 4.2 the accuracy error is presented which is the difference between training and validation accuracy.

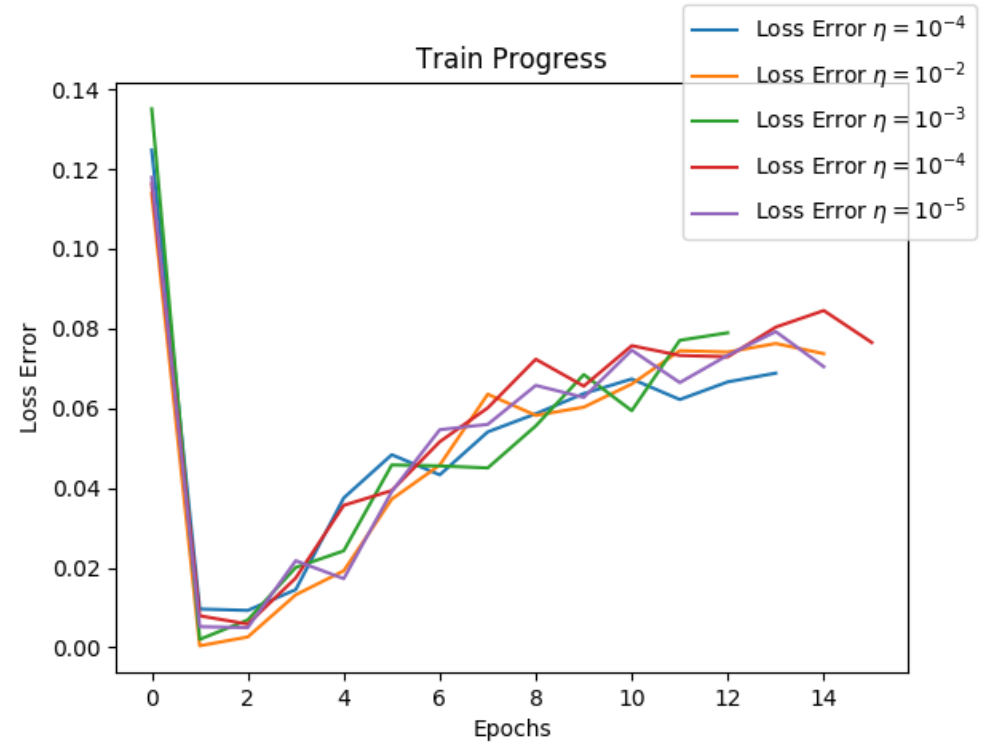


Figure 4.1: Loss Error for the learning rate search

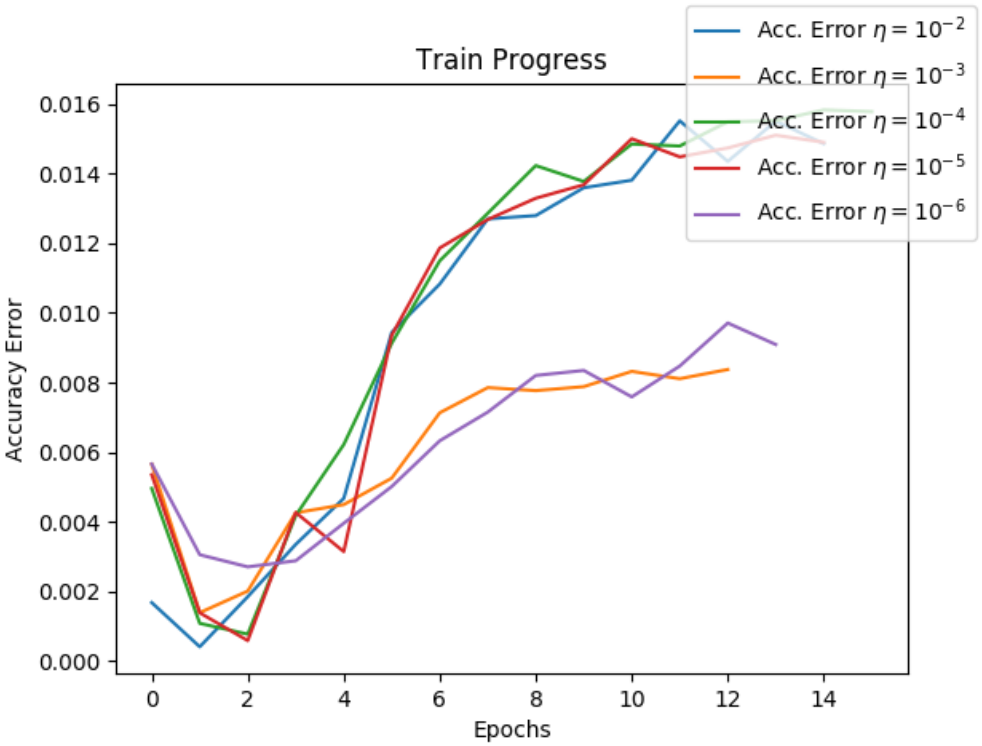


Figure 4.2: Accuracy Error for the learning rate search

From Figure 4.1 it can be seen that all learning rates achieve similar loss error.

The primary objective is to have a loss curve that is decaying with the number of epochs for both training and validation. This is of primary objective because the loss function is a direct function of the networks learning process and should hence be valued above all other metrics during training. Since the loss curves show marginal difference a choice was made to consult a paper by Dong et al. [4] that had a similar architecture and showed competitive results on the MICCAI BraTS dataset. This paper had used a learning rate of 10^{-4}. Since only one training session was done for each of the learning rate and the loss error curves showed similar results it was decided to continue with the same learning rate of 10^{-4} as Dong et al. [4]. The loss and accuracy for the training and validation for the learning rate of 10^{-4} can be seen in Figure 4.3 and 4.4.

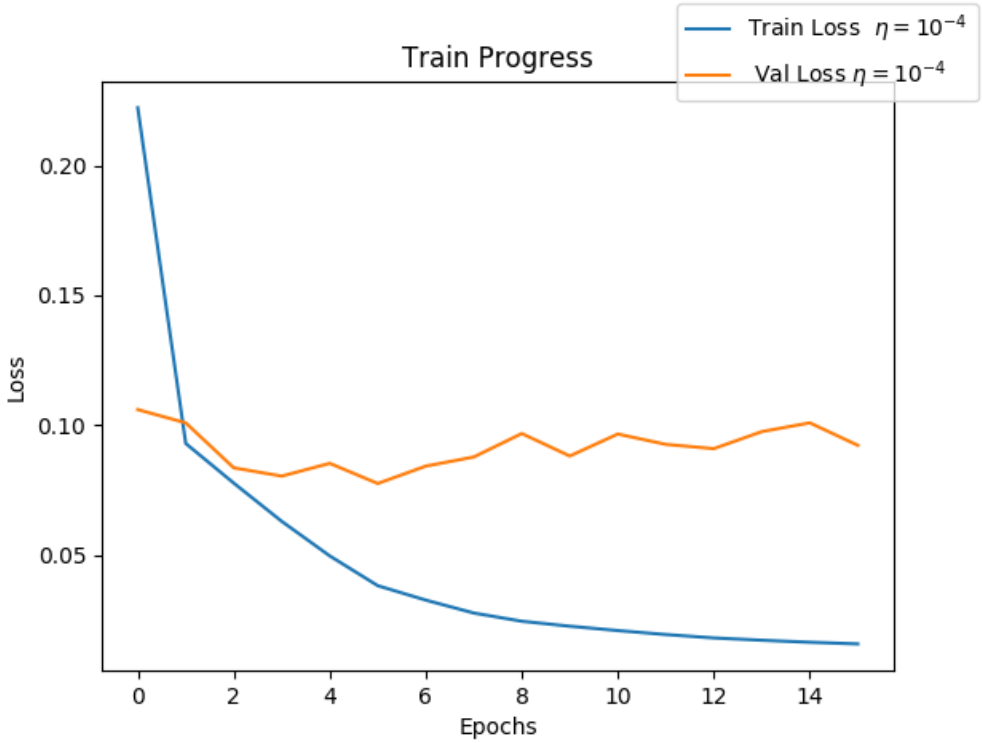


Figure 4.3: Training and Validation Loss for learning rate 10^{-4}

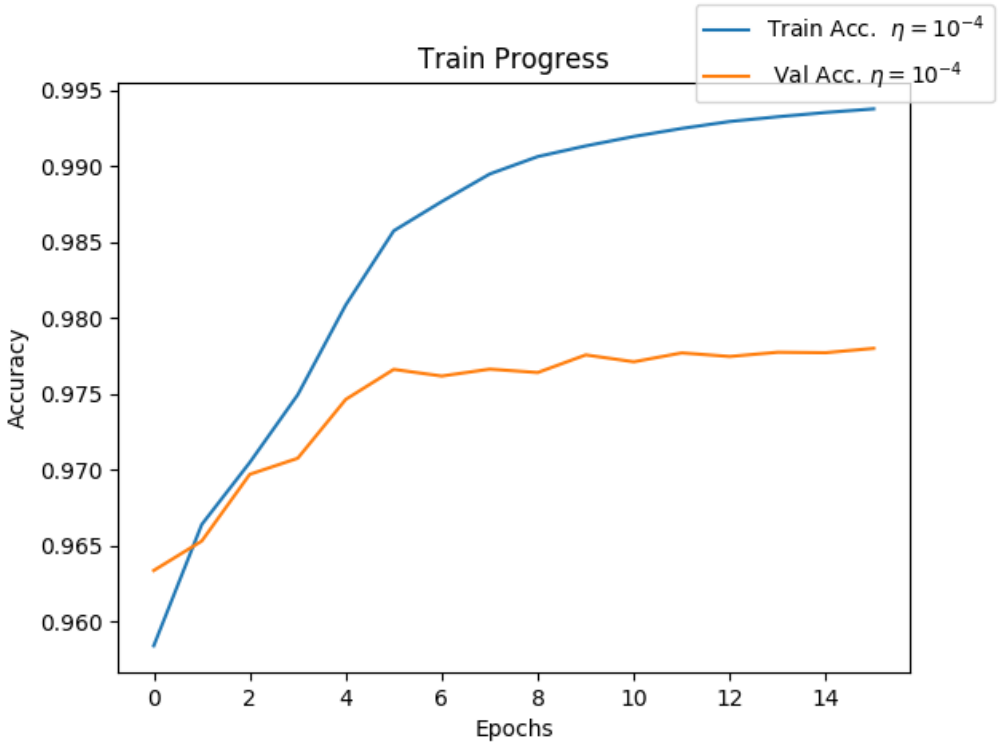


Figure 4.4: Training and Validation Accuracy for learning rate 10^{-4}

Weight initialization

A learning rate of 10^{-4} was chosen for training the network with he-normal as weight initialization method instead of using a normal distribution as described in equation 3.3. In Figure 4.5 and 4.6 are the results from training this network with the he-normal weight initialization as described in Section 2.2.8. In Figure 4.5 the training and validation loss are presented and in Figure 4.6 the training and validation accuracy can be seen. Comparing the loss in Figure 4.5 with the loss in Figure 4.3 it can be seen that the validation loss has dropped, this indicates a better performance of the network. The difference between the training and validation loss is still however substantial.

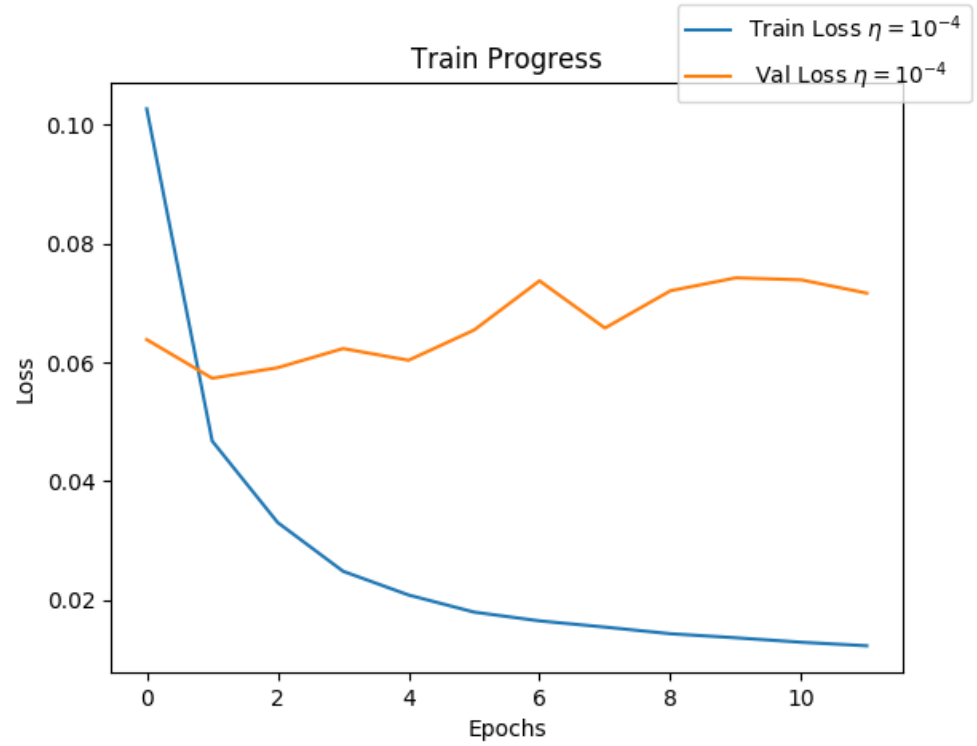


Figure 4.5: Training and Validation Loss for he-norm weight initialization

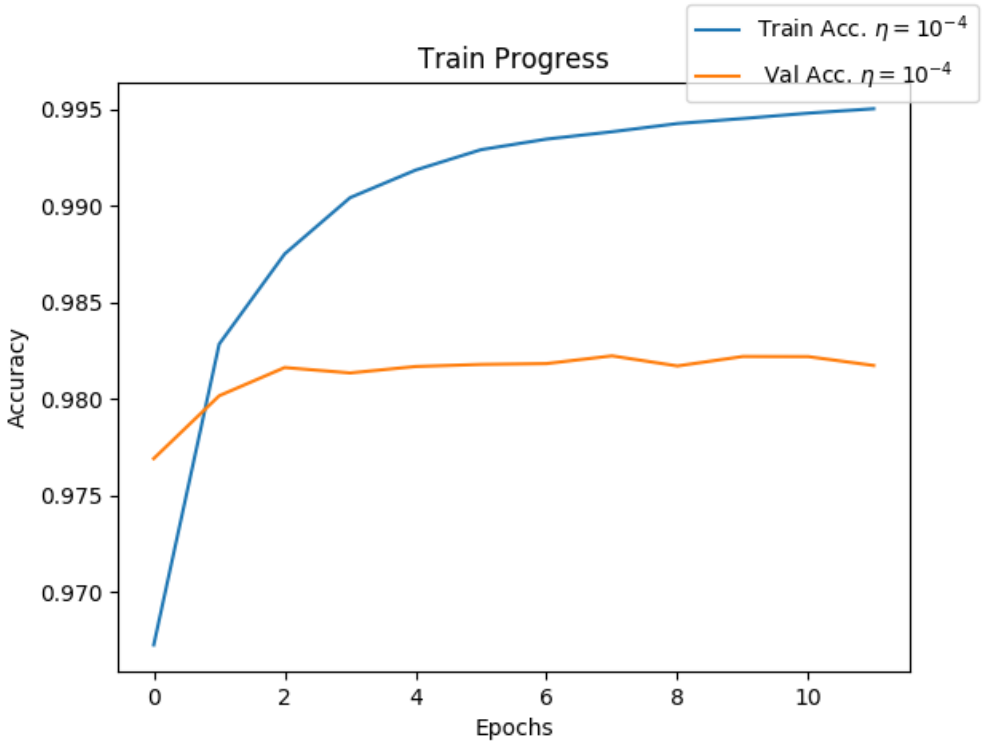


Figure 4.6: Training and Validation Accuracy for he-norm weight initialization

L2 regularization

In this experiment L2 regularization is added to the experiment on weight initialization, the results of training this network can be seen in Figure 4.7 and 4.8. The training and validation loss and accuracy is presented in Figure 4.7 and 4.8 respectively. Comparing Figure 4.7 with 4.5 it can be seen that the difference in training and validation loss has been improved which should make the network complete the task of segmentation more similar on the test dataset than the previous network.

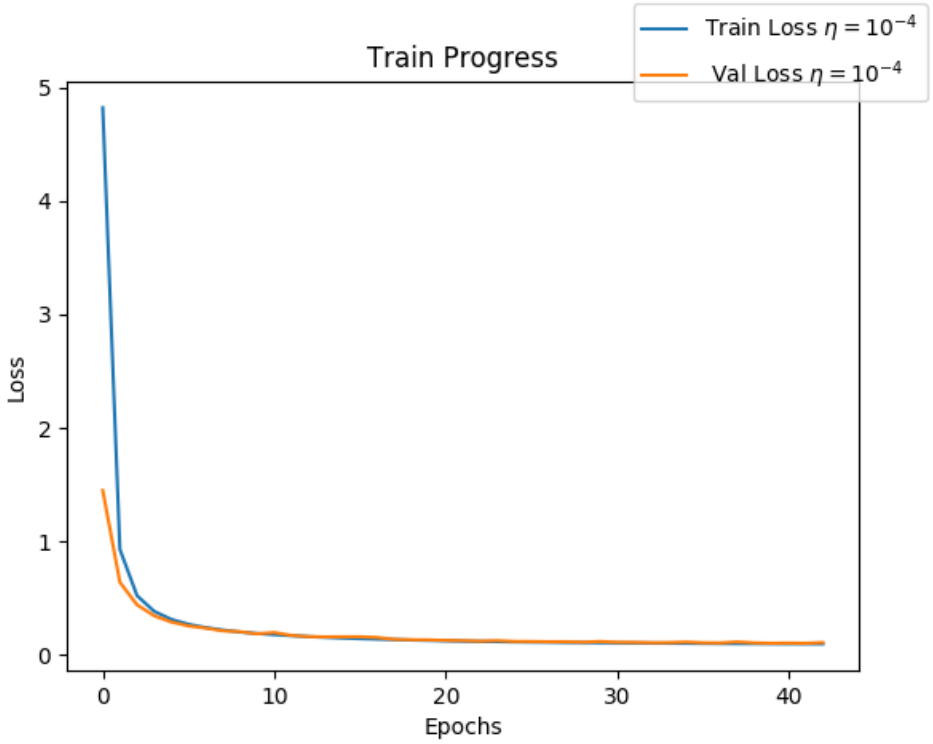


Figure 4.7: Training and Validation Loss when L2 regularization is added

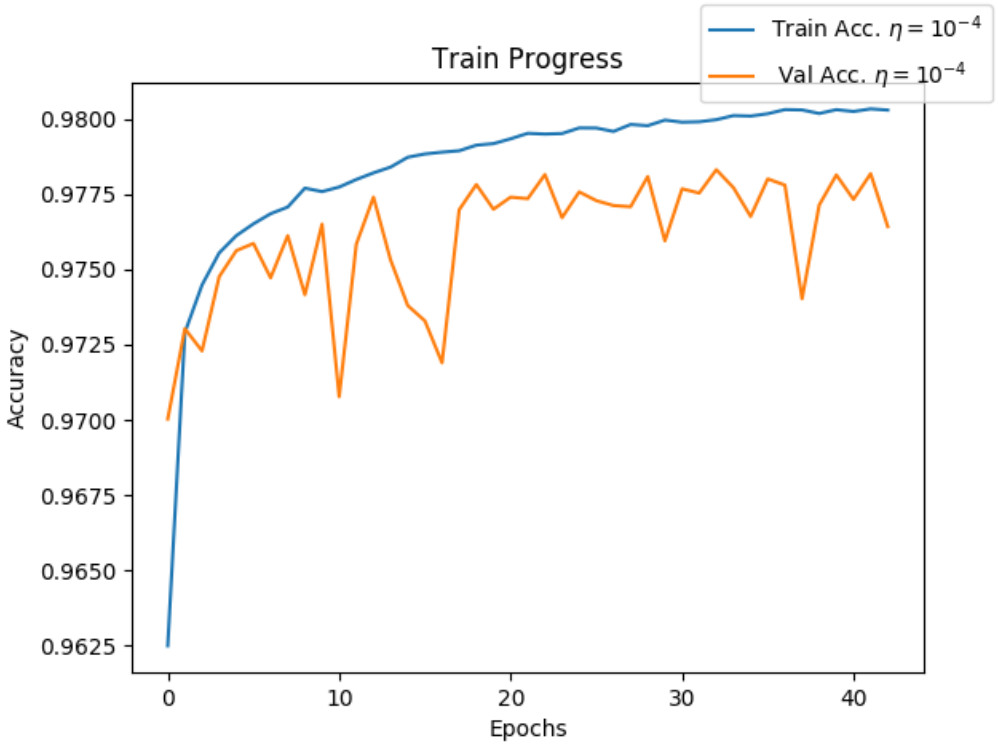


Figure 4.8: Training and Validation Accuracy when L2 regularization is added

Dropout

Dropout is added to the experiment with L2 regularization and the results are presented in Figure 4.9 and 4.10 . The training and validation loss of this experiment is presented in Figure 4.9. In Figure 4.10 the training and validation accuracy can be observed. In the comparison of the loss graphs in Figure 4.7 and 4.9 no major difference can be seen. Observing Figure 4.8 and 4.10 it can be seen that the difference in training and validation accuracy is smaller in 4.10 therefore this network should perform more similar on training and validation data, this indicates that performance on test dataset would be similar.

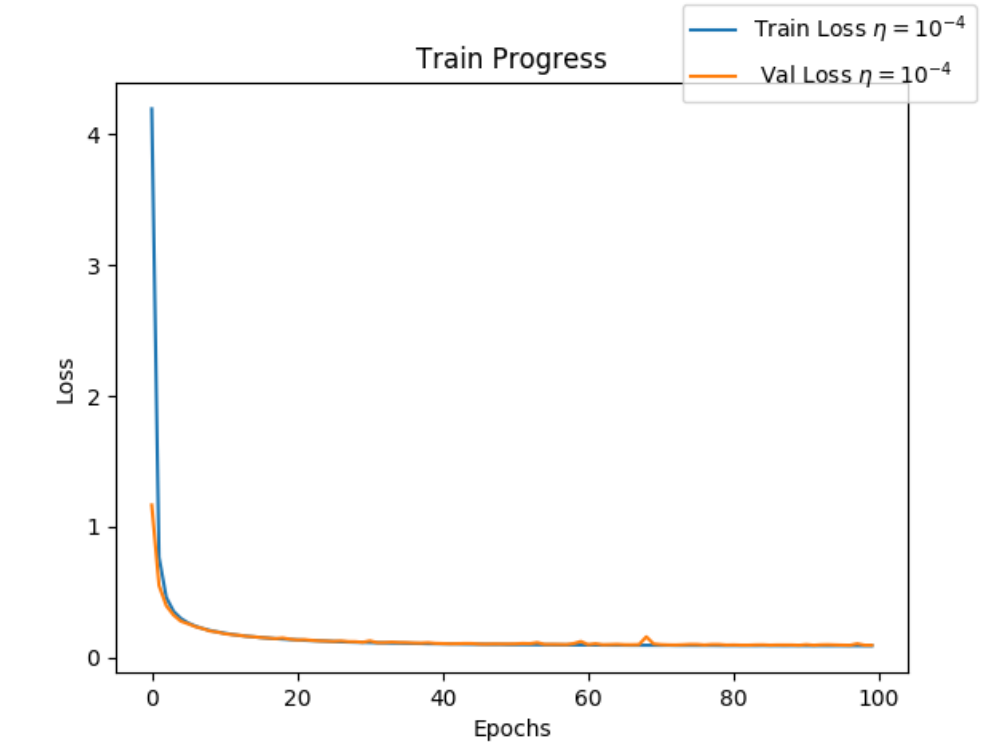


Figure 4.9: Training and Validation Loss when Dropout is added

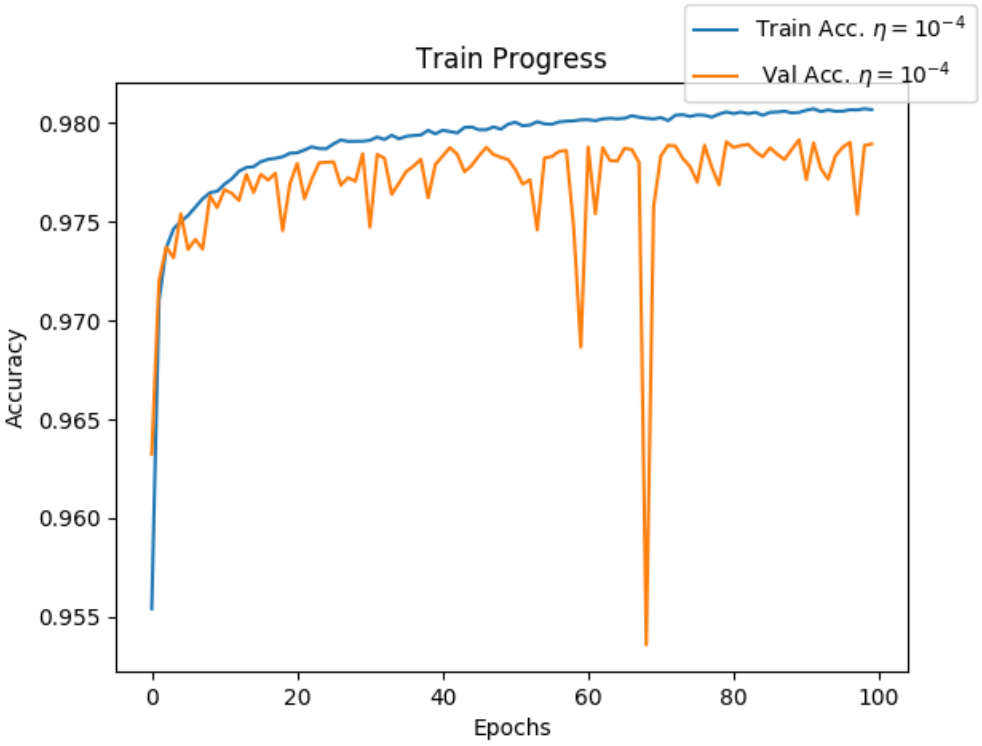


Figure 4.10: Training and Validation Accuracy when Dropout is added

Data augmentation

In this experiment data augmentation is added to the previous experiment when dropout was used, the results can be seen in Figure 4.11 and 4.12. The training and validation loss is presented in Figure 4.11 and in Figure 4.12 the training and validation accuracy can be observed. Also in this case the loss function looks fairly similar between Figure 4.9 and 4.11. In Figure 4.12 a smaller deviation between the training and validation accuracy is achieved than in Figure 4.10. The smaller deviation in Figure 4.12 indicate that this network would perform even more similarly on the test dataset than previous networks.

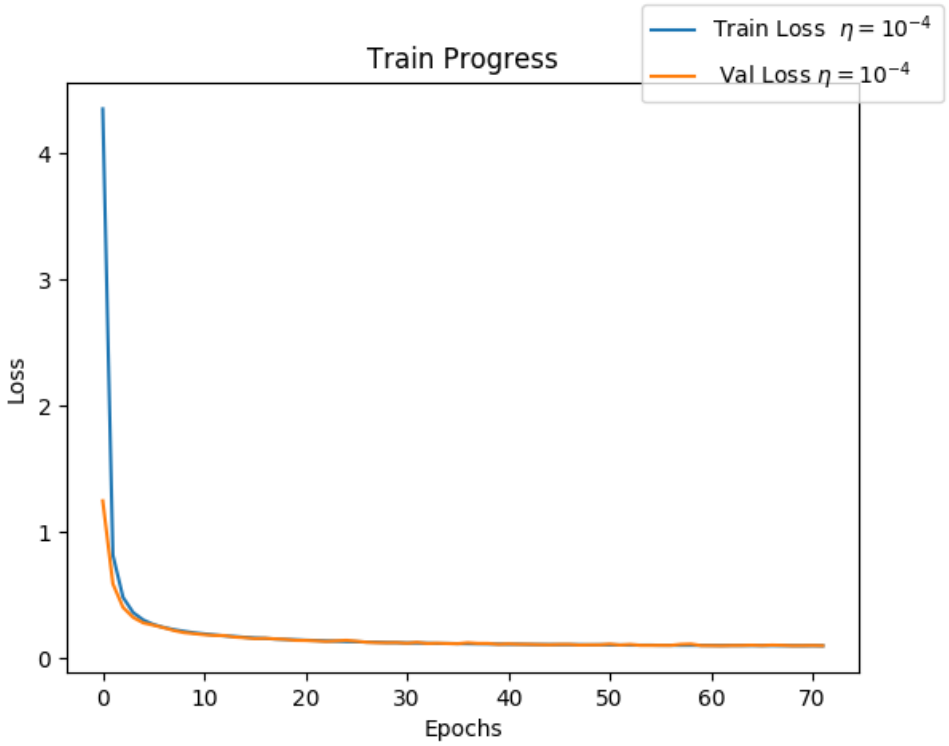


Figure 4.11: Training and Validation Loss when Data augmentation is added

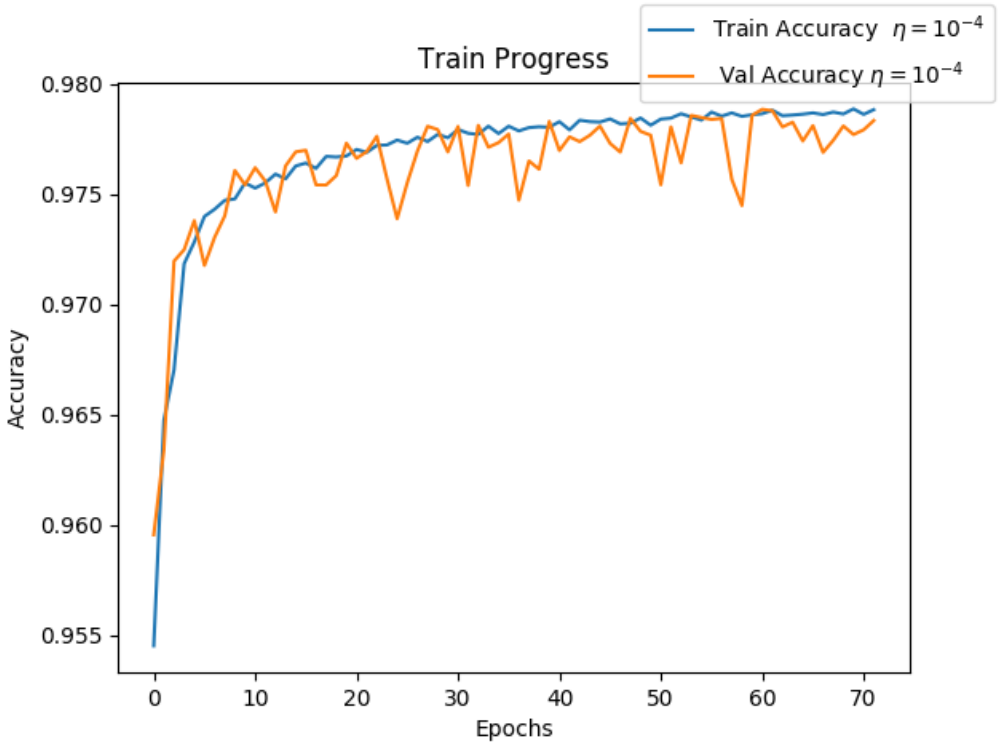


Figure 4.12: Training and Validation Accuracy when Data augmentation is added

Evaluation on test dataset

In this section the confusion matrix and dice scores are presented for the Single-stream U-Net trained in accordance with the data augmentation experiment. The problem specific dice scores are presented in three ways; as an average of all patient, in a boxplot and in a histogram.

		True			
		Background	**Necrosis**	**Edema**	**Enhancing**
Predicted	**Background**	23490	3.604	50.21	6.801
	Necrosis	7.387	52.56	49.16	6.768
	Edema	54.01	8.511	192.1	4.376
	Enhancing	5.895	9.062	8.993	59.75

Table 4.1: Confusion matrix from predictions on the test dataset for Single-stream U-Net, all entries in the table have been divided by 10^4 to give a cleaner view

Mean dice Scores for Single-stream U-Net

Enhancing	Tumor core	Whole tumor
0.586	0.710	0.817

Table 4.2: Mean dice scores for Single-stream U-Net on enhancing tumor, tumor core and whole tumor

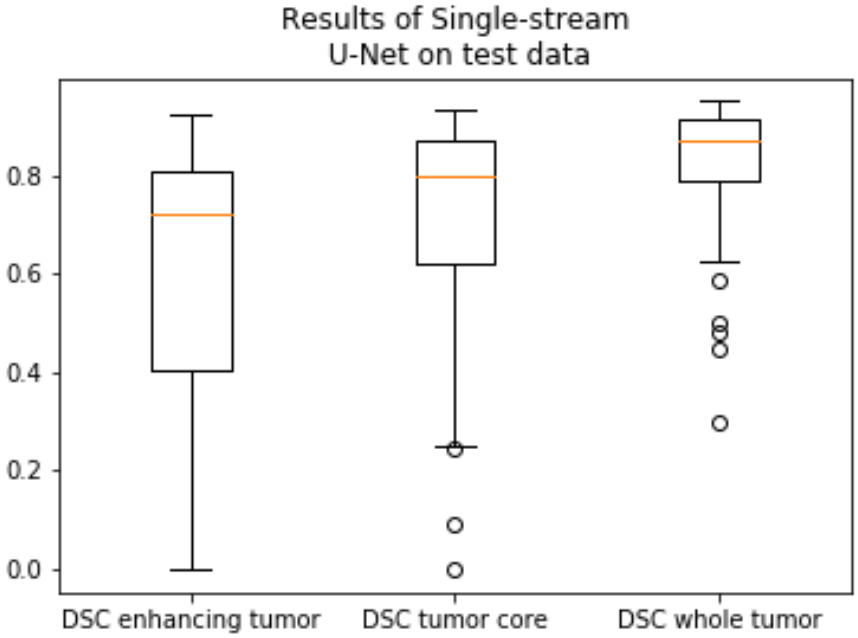


Figure 4.13: Boxplot of patient wise dice scores for Single-stream U-Net

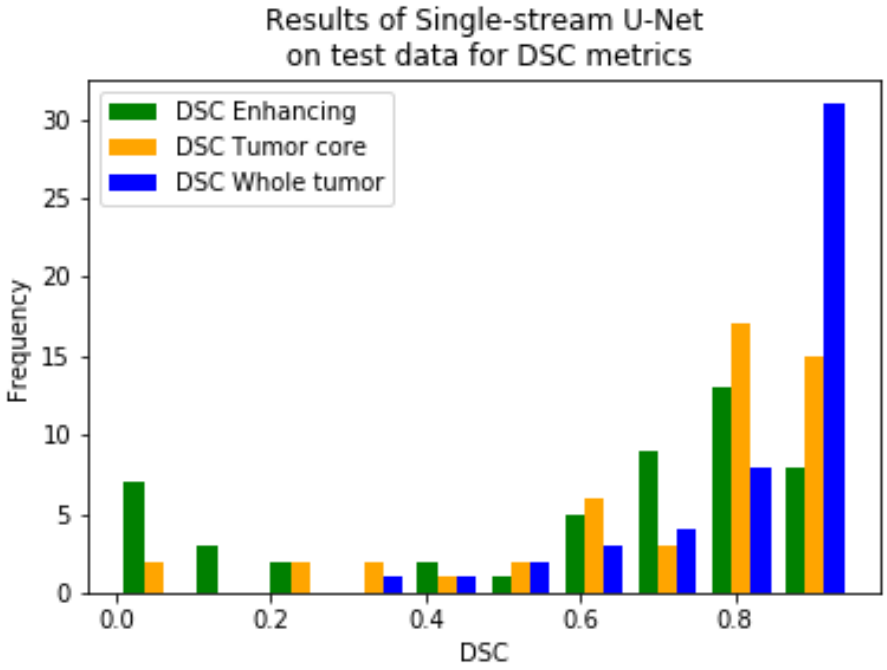


Figure 4.14: Histogram of patient wise dice scores for Single-stream U-Net

4.2 Multi-stream U-Net

4.2.1 End-to-end training

The learning rate and regularization methods used in Single-stream experiment "data augmentation" was chosen to implement on the Multi-stream U-Net trained end-to-end described in Section 3.7. The training and validation loss and accuracy results from training the Multi-stream U-Net end-to-end can be seen in Figure 4.23 and 4.24 respectively.

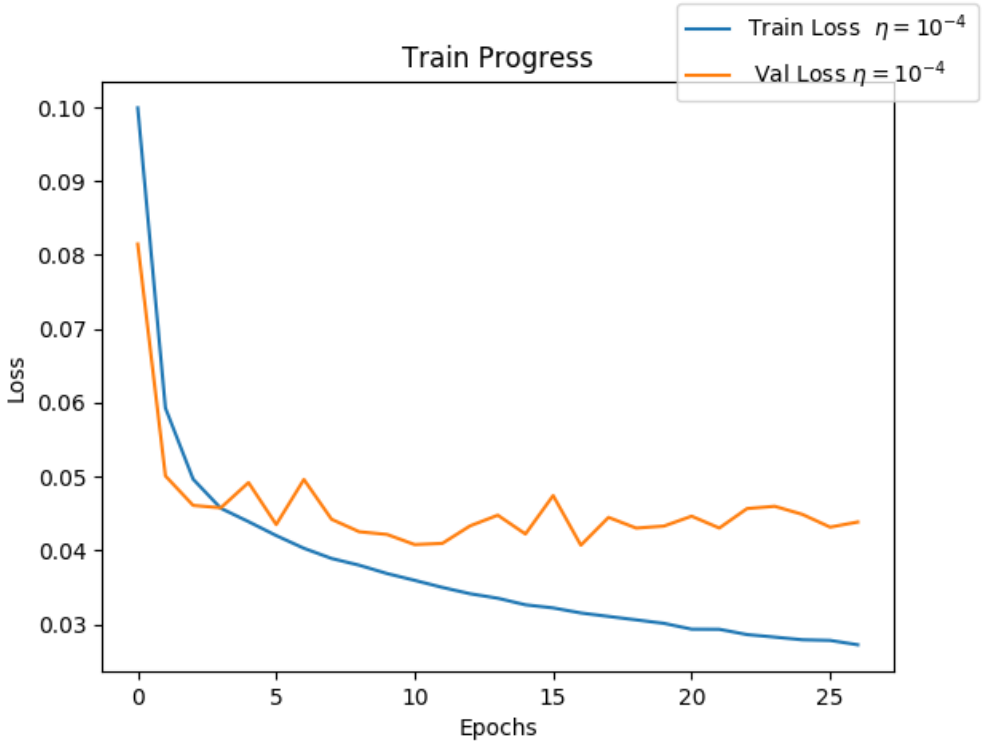


Figure 4.15: Training and Validation Loss for Multi-stream U-Net trained end-to-end

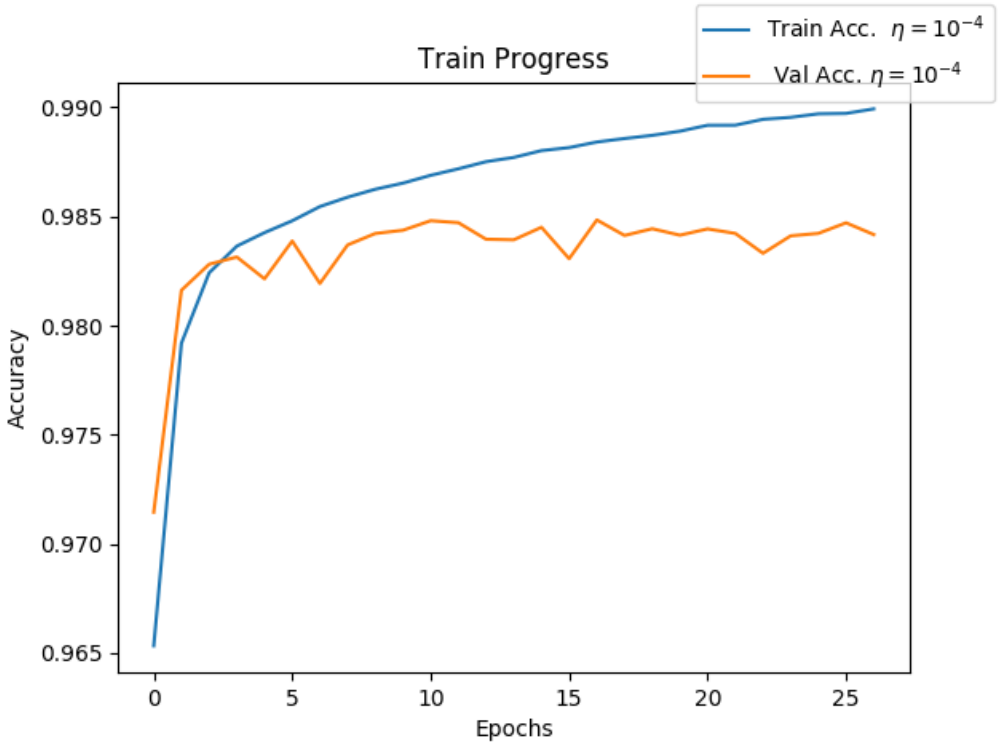


Figure 4.16: Training and Validation Accuracy for Multi-stream U-Net trained end-to-end

Evaluation on test dataset

In this section the confusion matrix and dice scores are presented for the Multi-stream U-Net trained end-to-end. The problem specific dice scores are presented in three ways; as an average of all patient, in a boxplot and in a histogram.

		True			
		Background	**Necrosis**	**Edema**	**Enhancing**
Predicted	**Background**	23510	5.458	29.01	2.698
	Necrosis	7.472	83.96	19.21	5.236
	Edema	48.45	12.82	194.5	3.260
	Enhancing	4.730	6.531	5.026	67.42

Table 4.3: Confusion matrix from predictions on the test dataset for Multi-stream U-Net trained end-to-end, all entries in the table have been divided by 10^4 to give a cleaner view

Mean dice Scores for Multi-stream U-Net trained end-to-end

Enhancing	Tumor core	Whole tumor
0.691	0.822	0.856

Table 4.4: Mean dice scores for Multi-stream U-Net trained end-to-end on enhancing tumor, tumor core and whole tumor

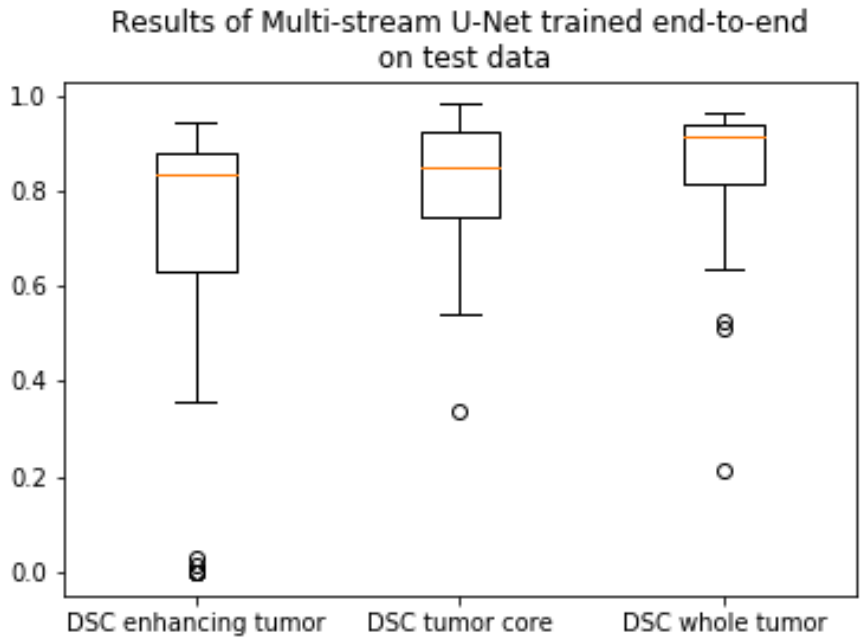


Figure 4.17: Boxplot of patient wise dice scores for Multi-stream U-Net trained end-to-end

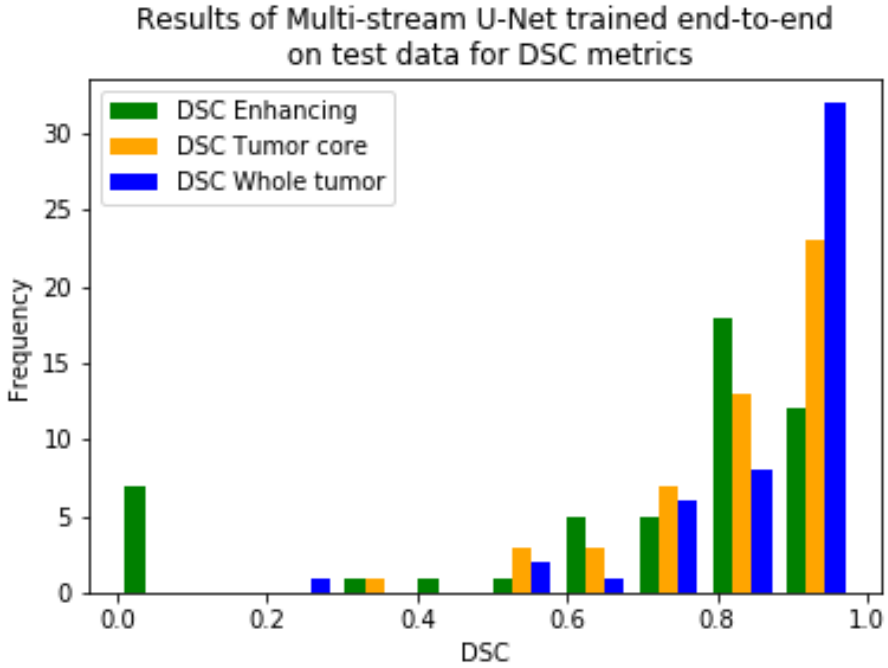


Figure 4.18: Histogram of patient wise dice scores for Multi-stream U-Net trained end-to-end

4.2.2 Sequential training

The same learning rate and regularization methods were used as in the training of the Multi-stream U-Net trained end-to-end. Training was conducted in two steps, first each path was trained with one modality each, the training and validation loss error and accuracy error of each path can be seen in Figure 4.19 and 4.20. The Flair modality yields smallest error when inspecting training and validation loss errors and accuracy errors in Figures 4.19 and 4.20. The Flair training and validation loss and accuracy is depicted in Figure 4.19 and 4.20 respectively to easier be able to see the difference between this result and the Multi-stream U-Net trained sequentially result. The weights of each path were then locked and training was proceeded after adding the concatenation and convolutional blocks. The results from the second cycle of training can be viewed by inspecting the training and validation loss and accuracy seen in Figure 4.19 and 4.20. Comparing the training results of training one path with Flair and the Multi-stream U-Net trained sequentially it can be seen that the loss curves look very similar. From inspecting the accuracy it can however be seen that the training is less volatile in the pretraiend Multi-stream U-Net trained end-to-end and that the accuracy is increased to roughly 97%.

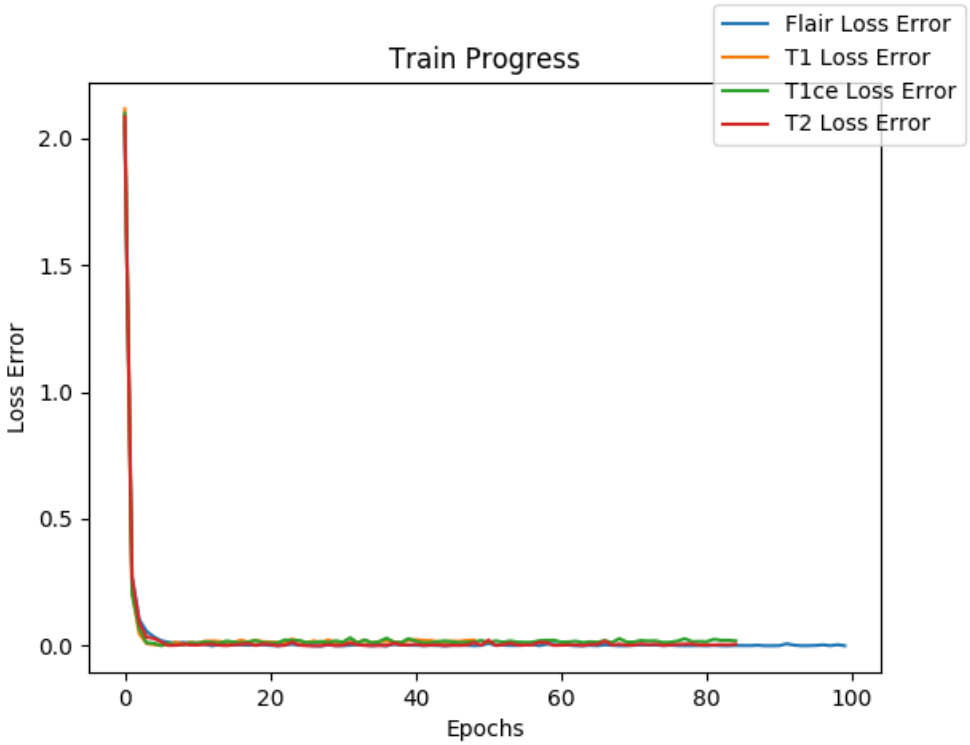


Figure 4.19: Training and Validation Loss Error for the paths in the Multi-stream U-Net trained sequentially

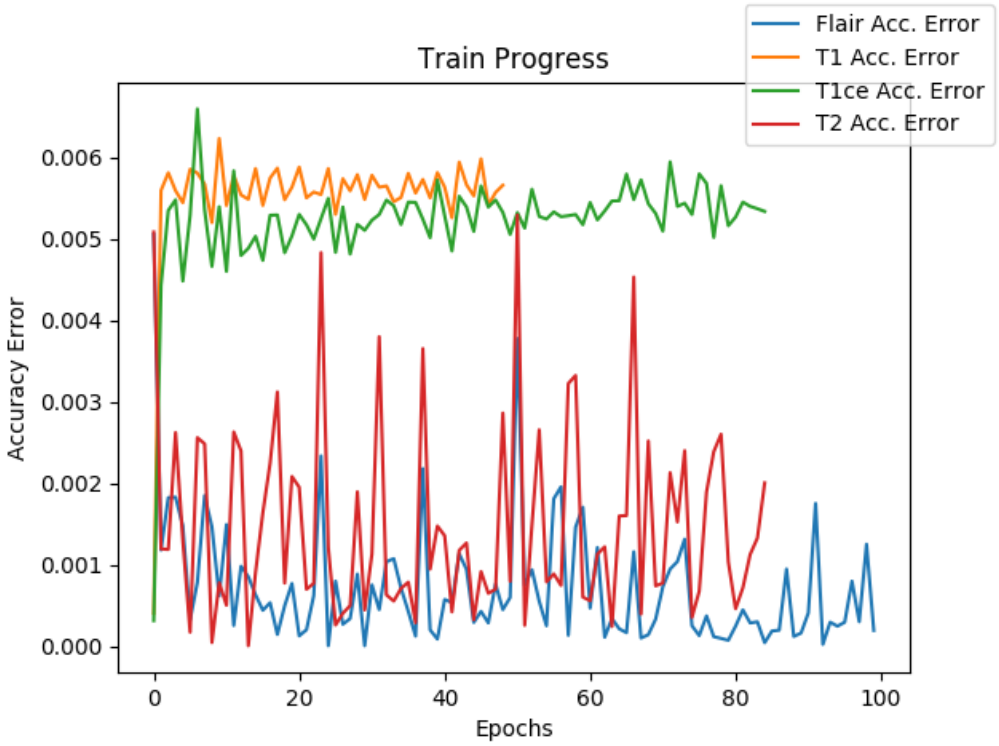


Figure 4.20: Training and Validation Accuracy Error for the paths in the Multi-stream U-Net trained sequentially

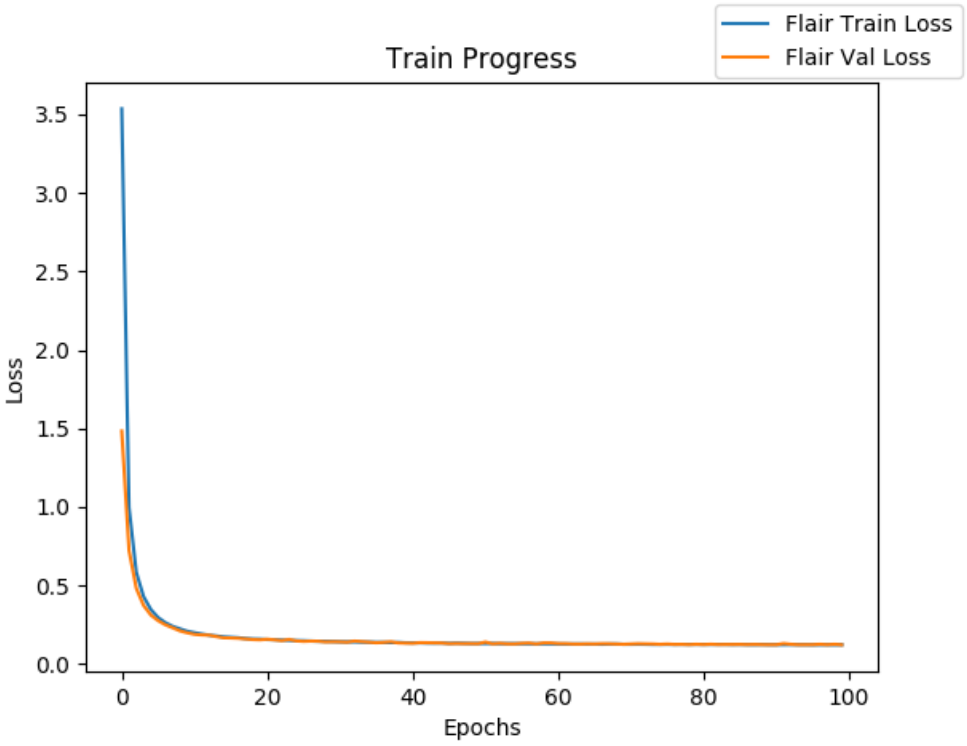


Figure 4.21: Training and Validation Loss for the pretrained Flair path

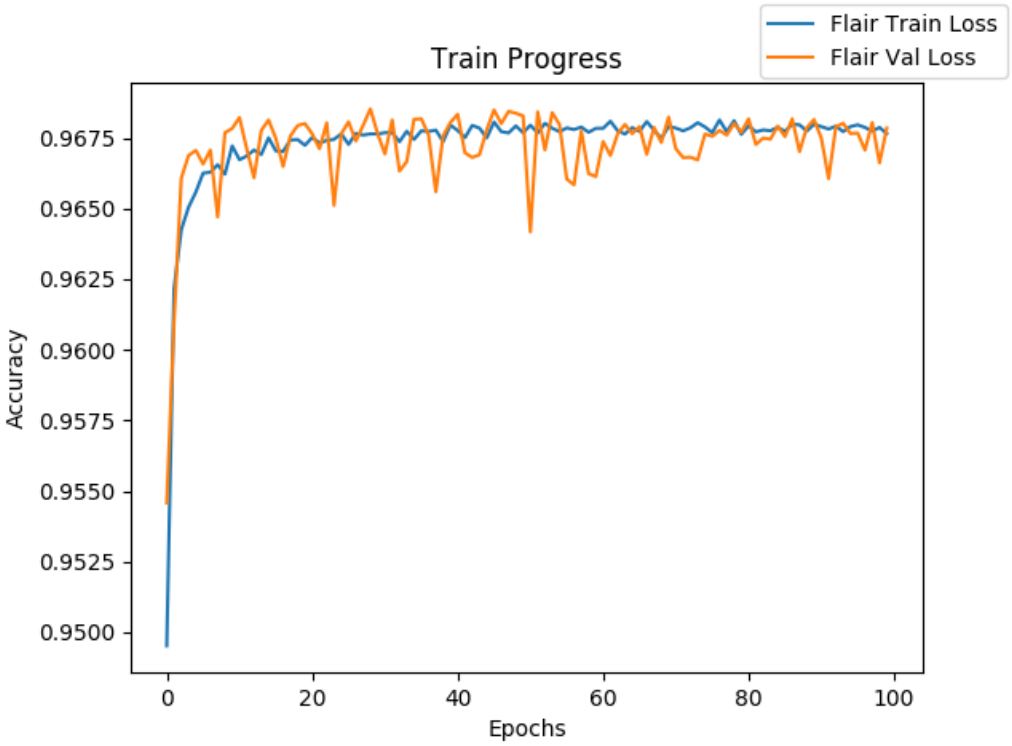


Figure 4.22: Training and Validation Accuracy the pretrained Flair path

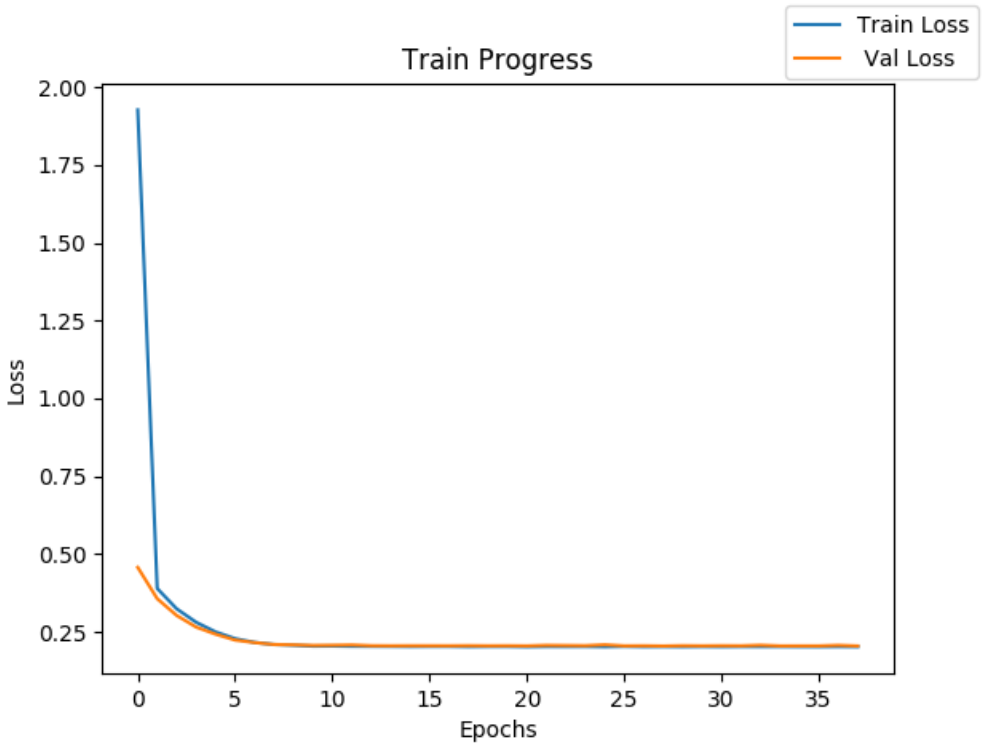


Figure 4.23: Training and Validation Loss when training with final convolutional block in the Multi-stream U-Net trained sequentially

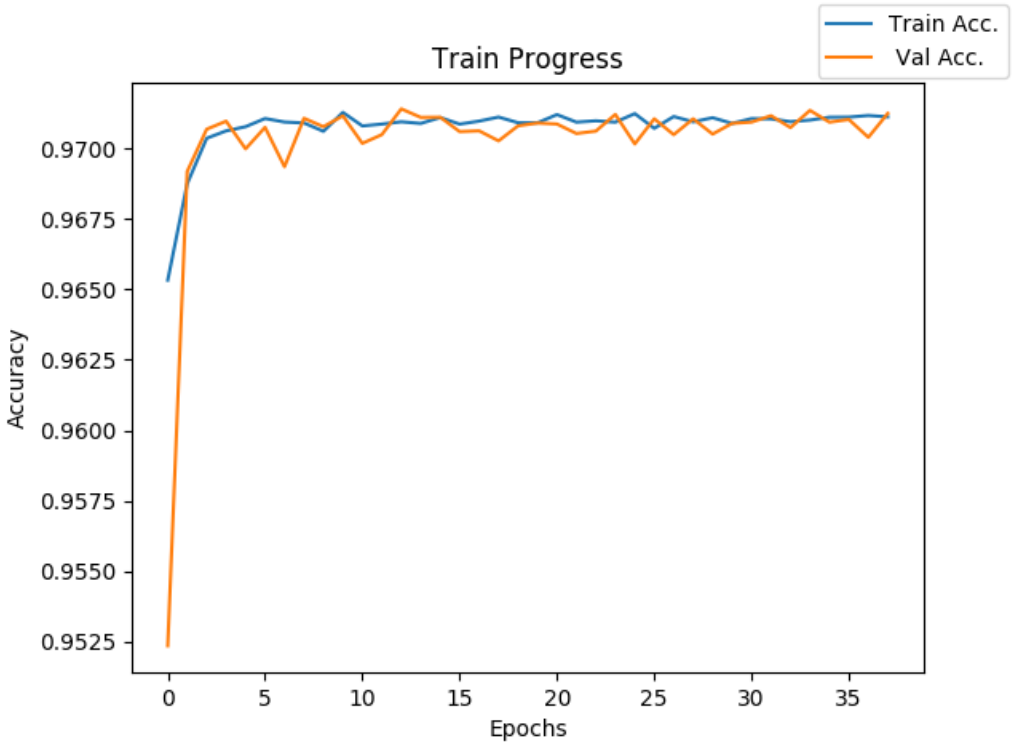


Figure 4.24: Training and Validation Accuracy when training with final convolutional block in the Multi-stream U-Net trained sequentially

Evaluation on test dataset

In this section the confusion matrix and dice scores are presented for the Multi-stream U-Net trained sequentially. The problem specific dice scores are presented in three ways; as an average of all patient, in a boxplot and in a histogram.

		True			
		Background	**Necrosis**	**Edema**	**Enhancing**
Predicted	**Background**	23500	0.7107	50.35	0.04760
	Necrosis	11.66	28.70	75.43	0.08970
	Edema	60.75	5.598	192.6	0.09520
	Enhancing	11.15	5.671	66.72	0.1580

Table 4.5: Confusion matrix from predictions on the test dataset for Multi-stream U-Net trained sequentially, all entries in the table have been divided by 10^4 to give a cleaner view

Mean dice Scores for Multi-stream U-Net trained sequentially

Enhancing	Tumor core	Whole tumor
0.042	0.199	0.796

Table 4.6: Mean dice scores for Multi-stream U-Net trained sequentially on enhancing tumor, tumor core and whole tumor

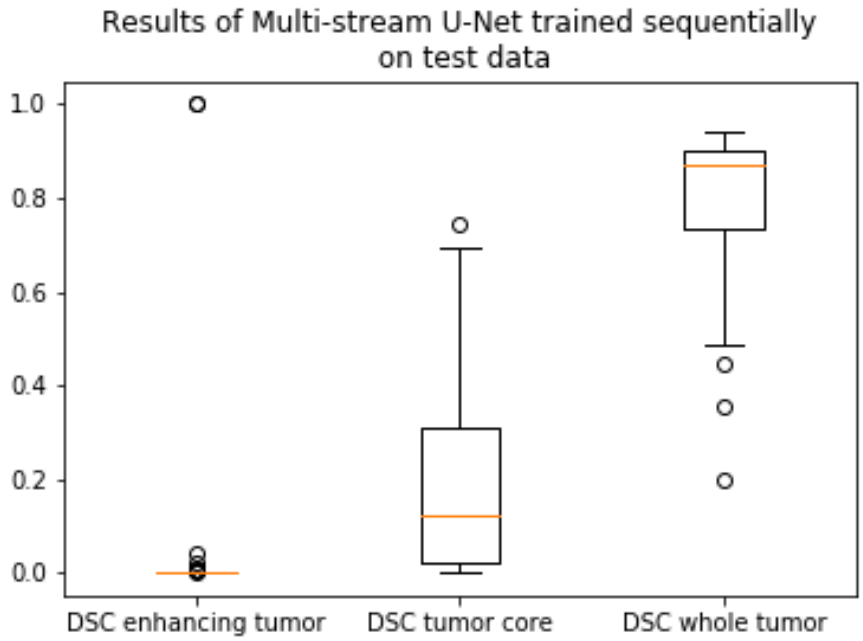


Figure 4.25: Boxplot of patient wise dice scores for Multi-stream U-Net trained sequentially

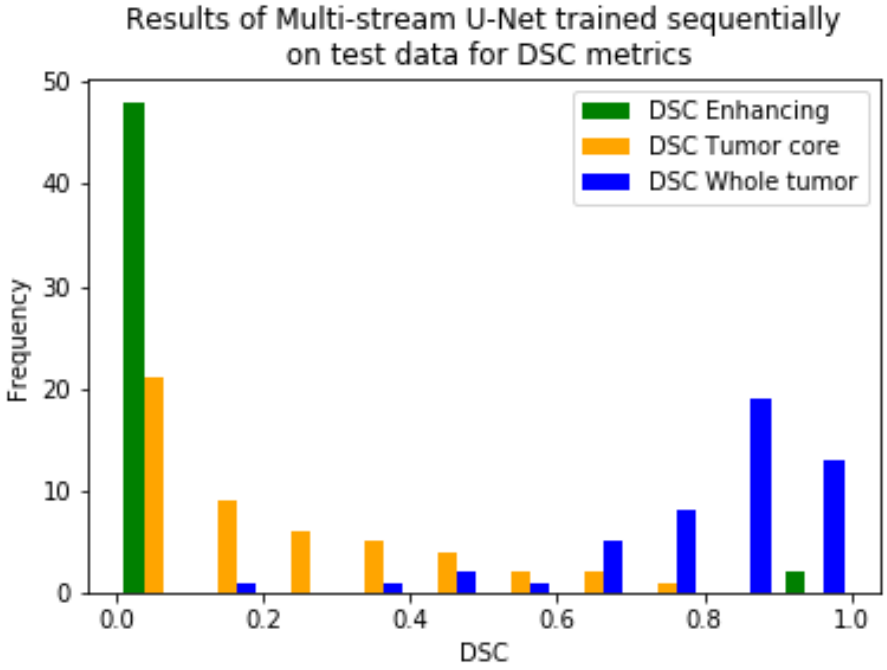


Figure 4.26: Histogram of patient wise dice scores for Multi-stream U-Net trained sequentially

4.3 Performance evaluation

This section compares the performance of the three networks on the test dataset. The confusion matrices, and dice scores are compared to observe the quantitative results.

4.3.1 Confusion matrices

The confusion matrices are presented for the three networks in Table 4.1, 4.3 and 4.5. Observing the diagonals of the confusion matrices it can be seen the the same order of magnitude of correctly classified pixels are achieved for all networks when looking at Background, Necorsis and Edema. The Multi-stream U-Net trained sequentially does however not perform as well on classifying enhancing tumor where it has two orders of magnitude lower correctly classified pixels. The Multi-stream U-Net trained end-to-end is predicting more necrosis as edema than the Single-stream U-Net, on the other hand more enhancing tumor is predicted as background by the Single-stream U-Net. By inspecting the confusion matrix of the Multi-stream U-Net trained sequentially it can be seen that the performance is worse than for both the Multi-stream U-Net trained end-to-end and the Single-stream U-Net. Observation of the Multi-stream U-Net trained sequentially reveals that it incorrectly predicts more background pixels as edema, necrosis and enhancing tumor than the other networks. The opposite can be seen for pixels which are enhancing tumor, in these cases it predicts less pixels to belong to other classes than enhancing tumor than the other two networks. From observing the confusion matrix of the Multi-stream U-Net trained end-to-end it can be seen that it is out performing both the other networks.

4.3.2 Dice scores

The mean dice scores on the test dataset for the three networks of the enhancing tumor, tumor core and whole tumor are presented in Figure 4.2, 4.4 and 4.6. An observation of the mean dice scores clearly show that the Multi-stream U-Net trained end-to-end achieves better results than both the other architectures. A more detailed view of the dice scores for the network architectures can be obtained by observing the boxplots and histograms for Single-stream and Multi-stream trained U-Nets in Figure 4.13, 4.17, 4.25 and 4.14, 4.18, 4.26 respectively. From observing the boxplots it can be seen that for enhancing tumor and tumor core the interval for the first through fourth quartile is smaller in the Multi-stream U-Net trained end-to-end than for the other networks. The median is higher in the Multi-stream U-Net trained end-to-end for both the enhancing and tumor core metrics. In the case of whole tumor the box plots look quite similar for the Single-stream U-Net and Multi-stream U-Net trained end-to-end architectures but a slightly higher median can be seen for the dice scores of the Multi-stream U-Net trained end-to-end. The Multi-stream U-Net trained sequentially performs worse than the other two networks on all tumor classes, the dice score for whole tumor can however compete with the

Single-stream U-Net and Multi-stream U-Net trained end-to-end architectures. By observing the histogram it can be seen that the Multi-stream U-Net trained end-to-end has higher concentration of dice scores closer to 1.0 and a shorter tail than the Single-stream U-Net. Inspecting the histogram of the Multi-stream U-Net trained sequentially, previous observations of the performance of this architecture can be further strengthened.

4.3.3 Predictions of models

In Figures 4.27 to 4.32, the ground truth and predictions of the three networks are compared for six patients. From inspecting the segmentation's of the networks it can be seen that the Multi-stream U-Net trained end-to-end has most similar results to that of the ground truth. These results are consistent with those of the quantitative results. It should however be noted that in some cases the Multi-stream U-Net trained sequentially seems to out perform the other networks for example when mainly necrosis is present. From inspecting segmentation results in patient 273 and 270 such a behavior can be seen. It can be observed that the Multi-stream U-Net trained sequentially and the Single-stream U-Net architectures are missing some of the finer details from the ground truth.

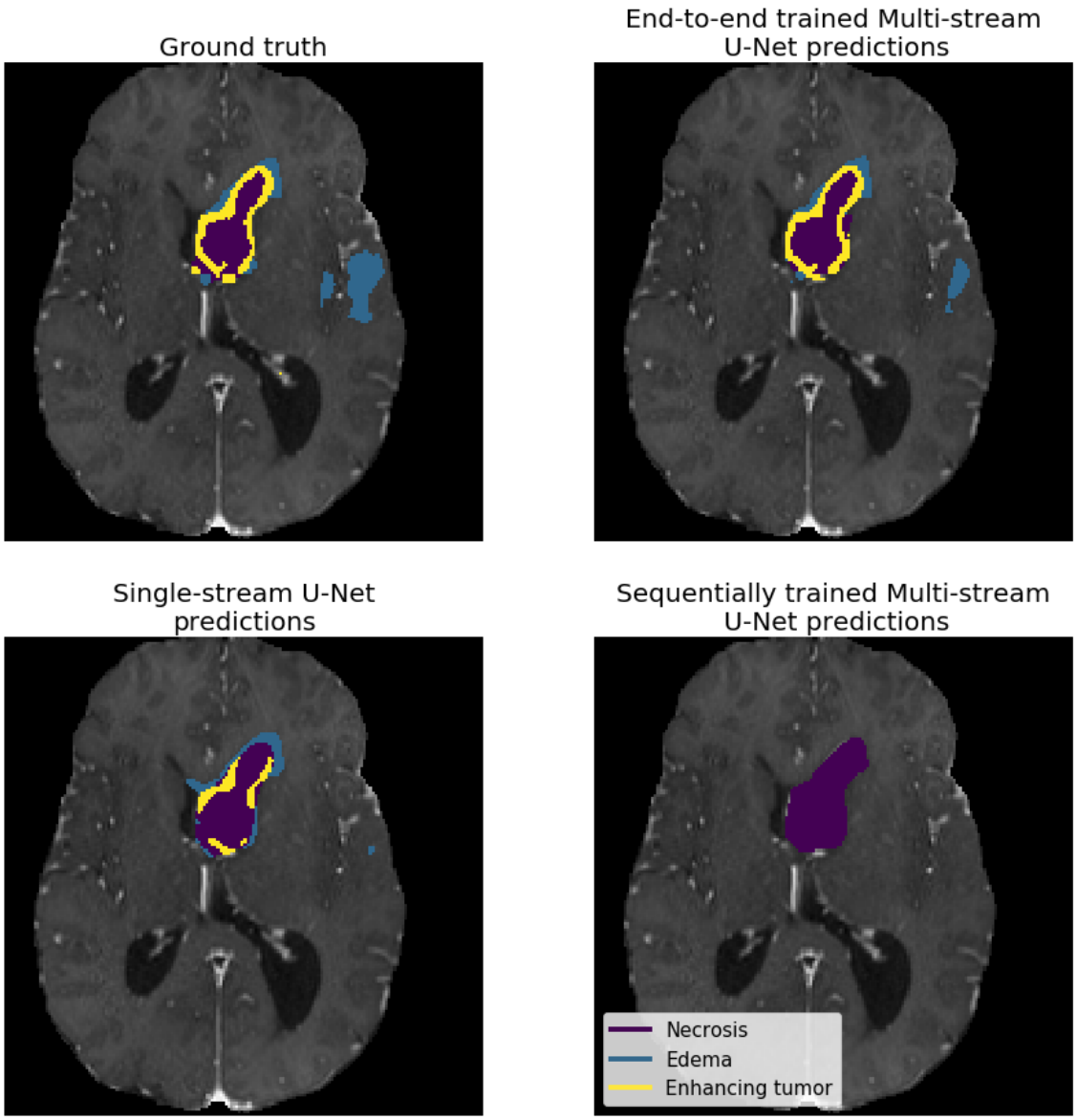


Figure 4.27: Ground truth and segmentation results for all networks for Patient 50

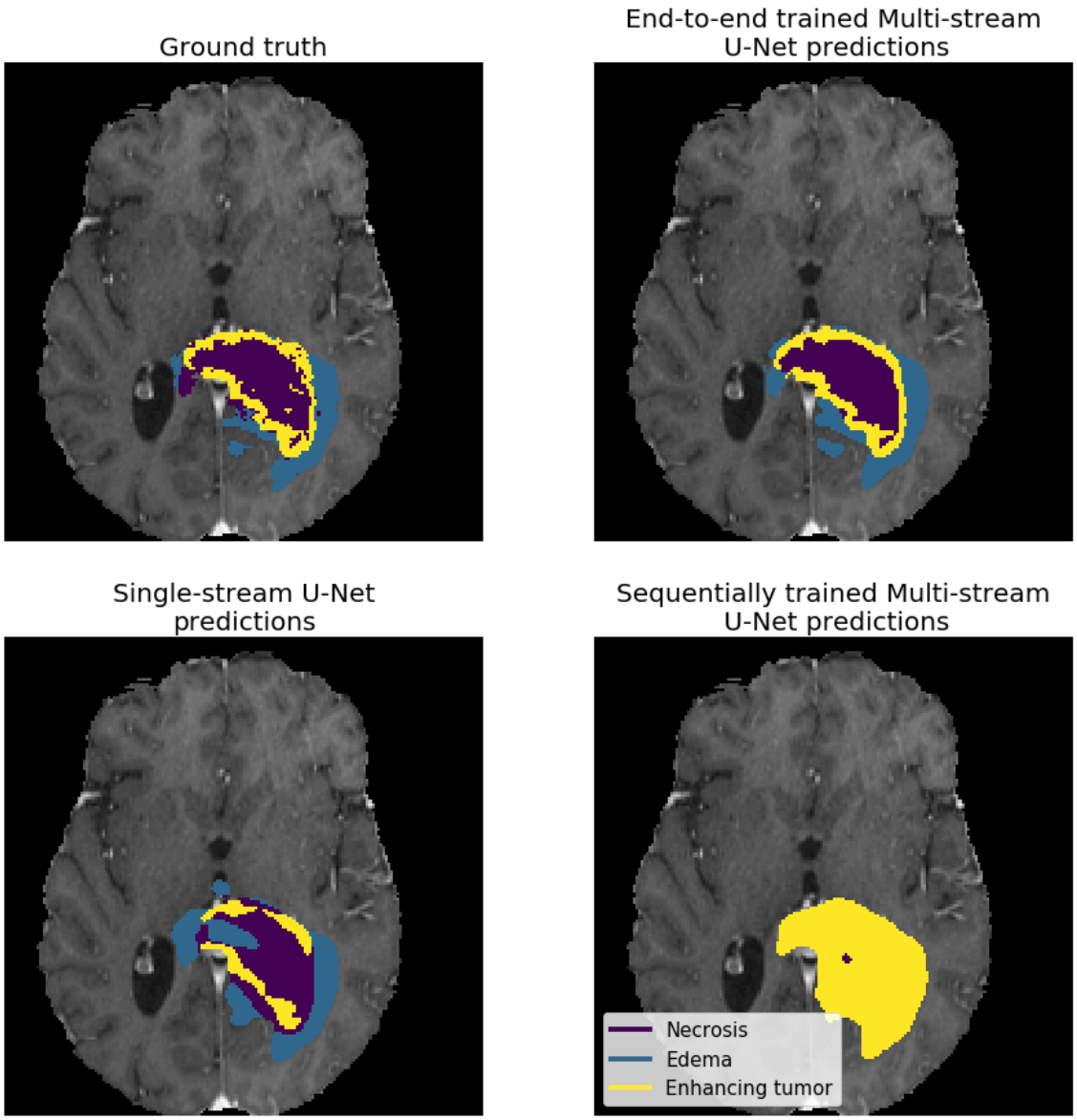


Figure 4.28: Ground truth and segmentation results for all networks for Patient 52

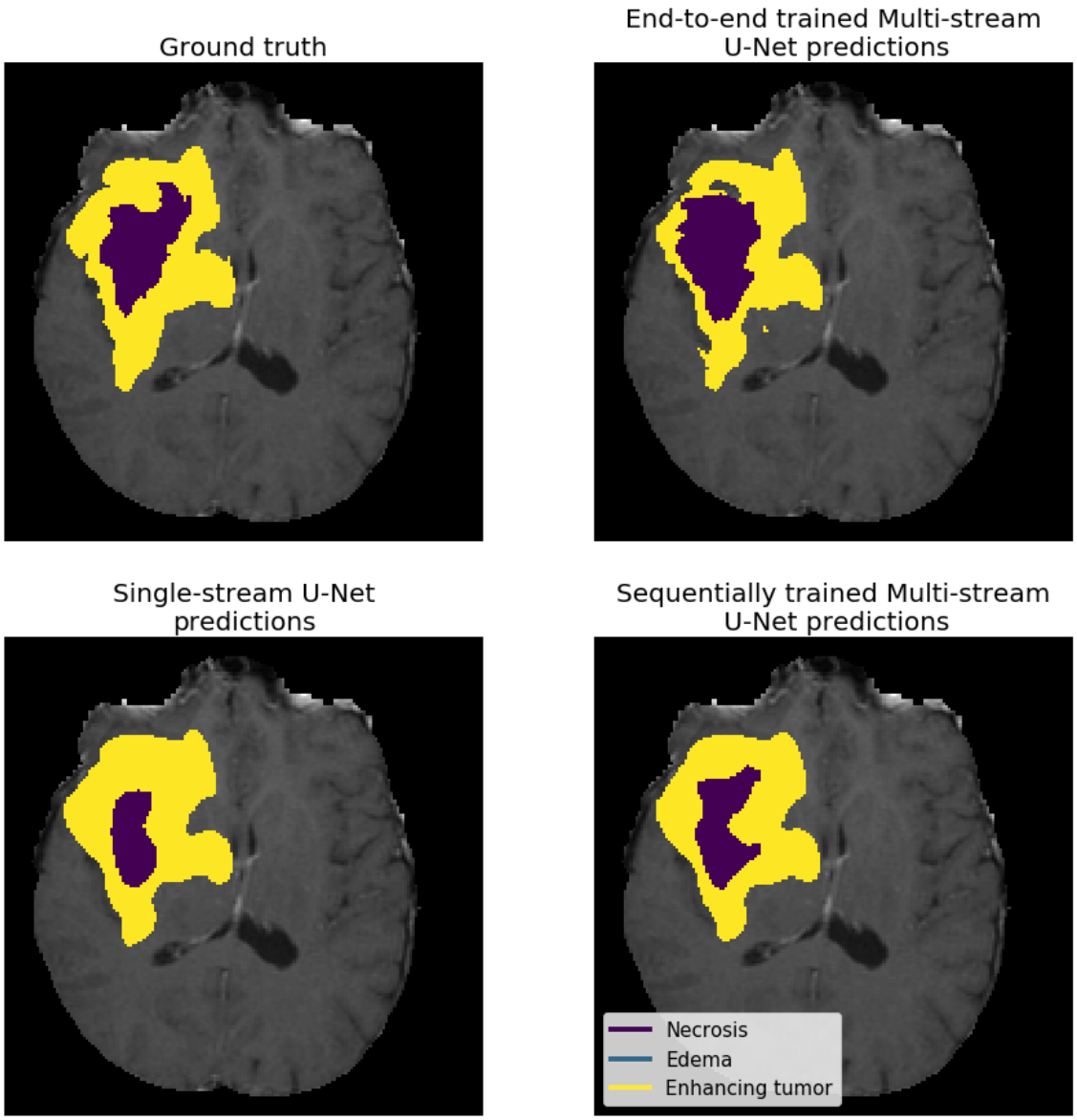


Figure 4.29: Ground truth and segmentation results for all networks for Patient 329

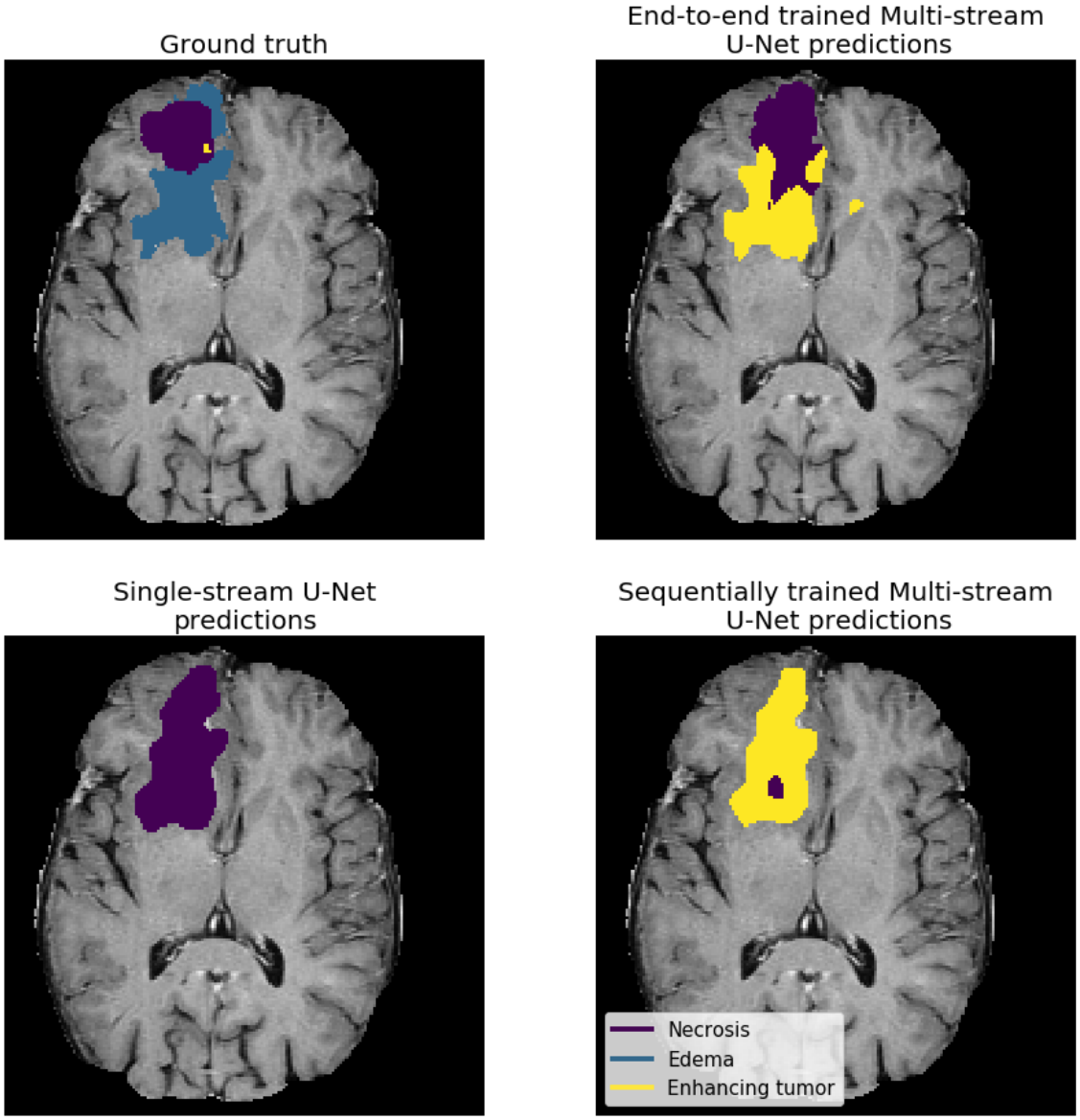


Figure 4.30: Ground truth and segmentation results for all networks for Patient 322

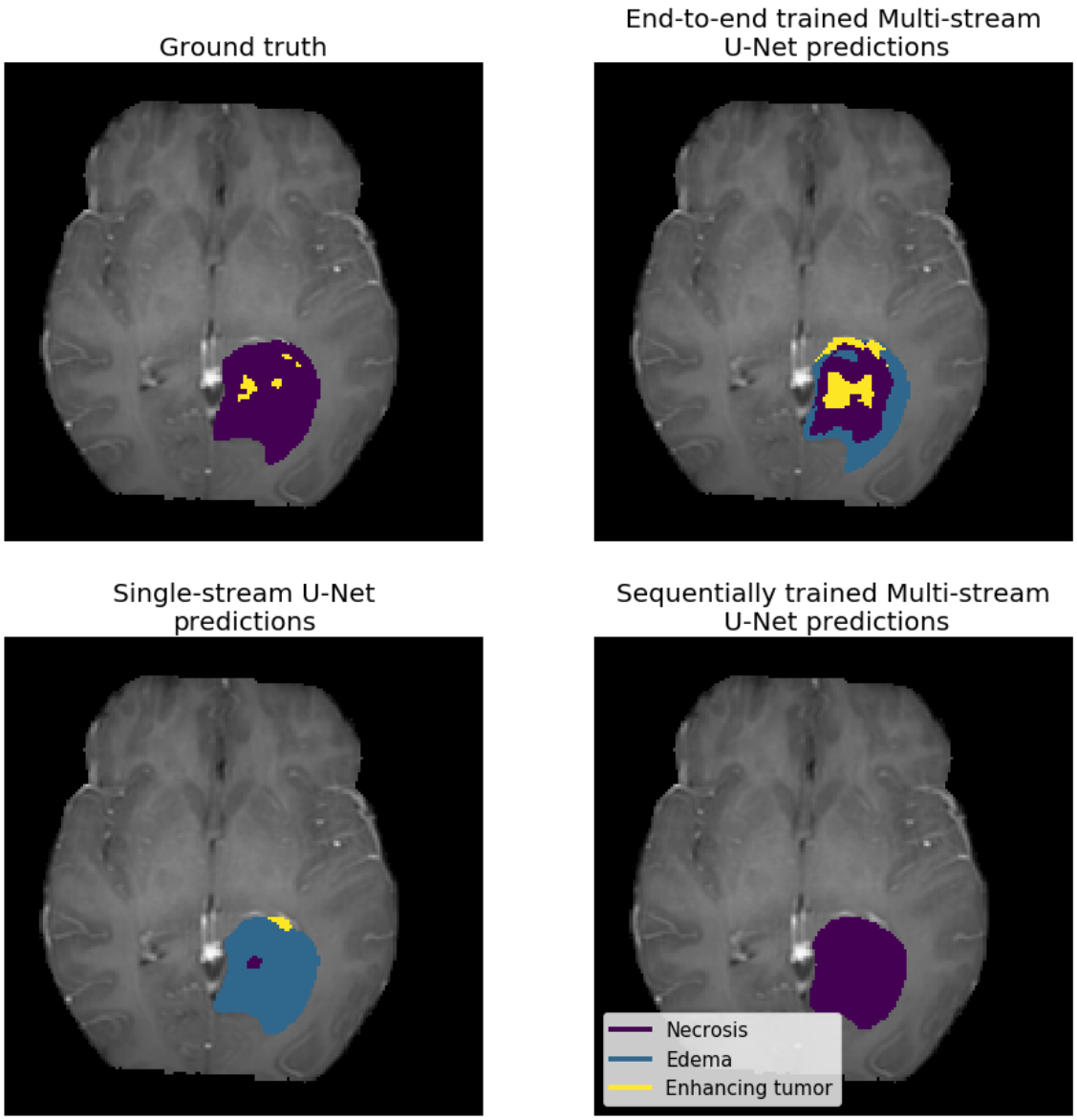


Figure 4.31: Ground truth and segmentation results for all networks for Patient 273

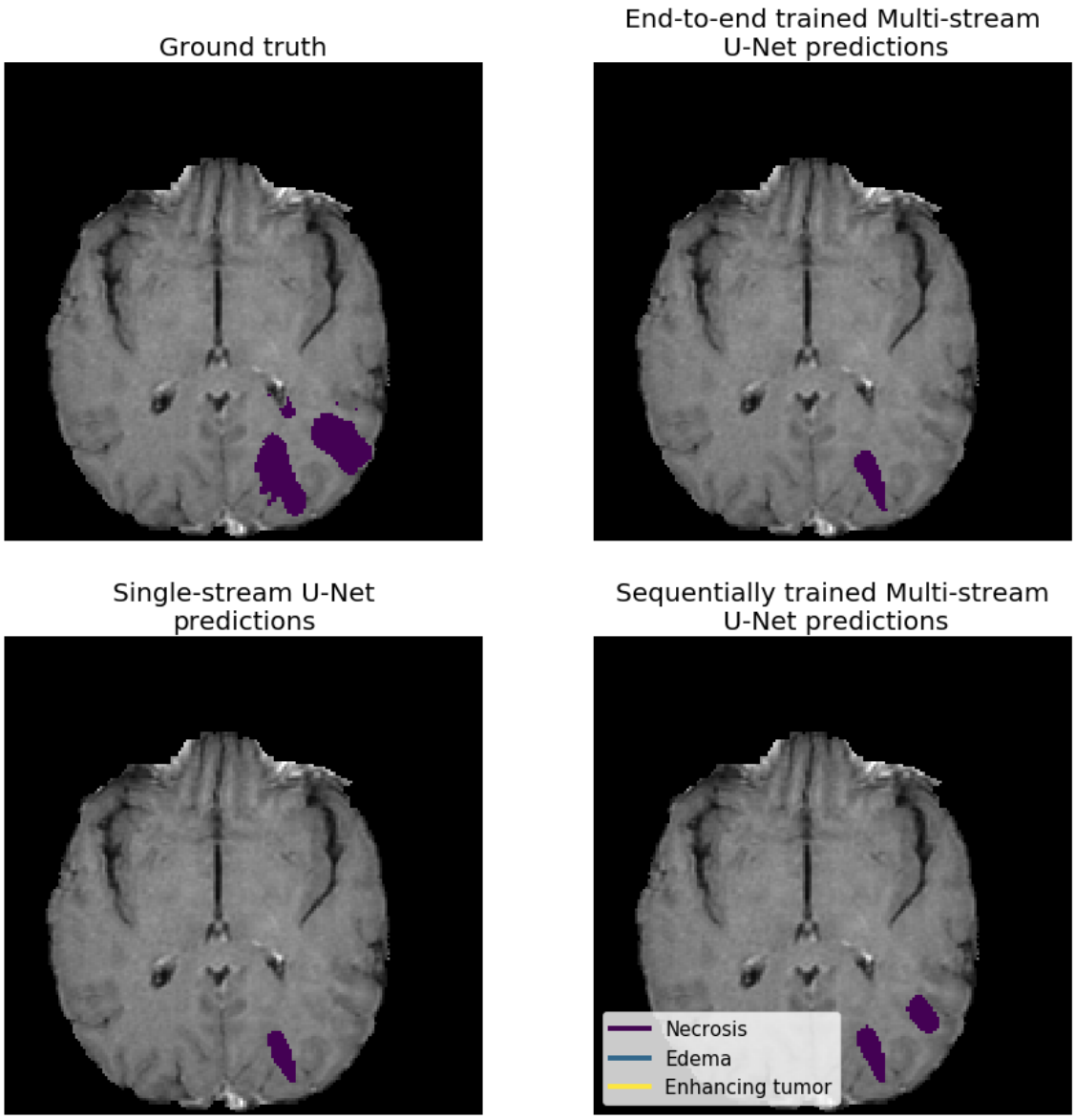


Figure 4.32: Ground truth and segmentation results for all networks for Patient 270

4.3.4 Volume estimation and visualization

Volume estimation

In Figure 4.33, 4.34 and 4.35 the volume differences between the ground truth and predicted tumor segmentation's can be seen for the tumor core. An inspection of these plots show that the Multi-stream U-Net trained end-to-end has a higher concentration around zero than the Single-stream U-Net. There are about as many over as under predictions for the Multi-stream U-Net trained end-to-end whereas the Single-stream U-Net seams more biased towards under estimating the tumor volume. Inspecting the volume difference of the Multi-stream U-Net trained sequentially it can be seen that the network heavily under estimates the tumor volume.

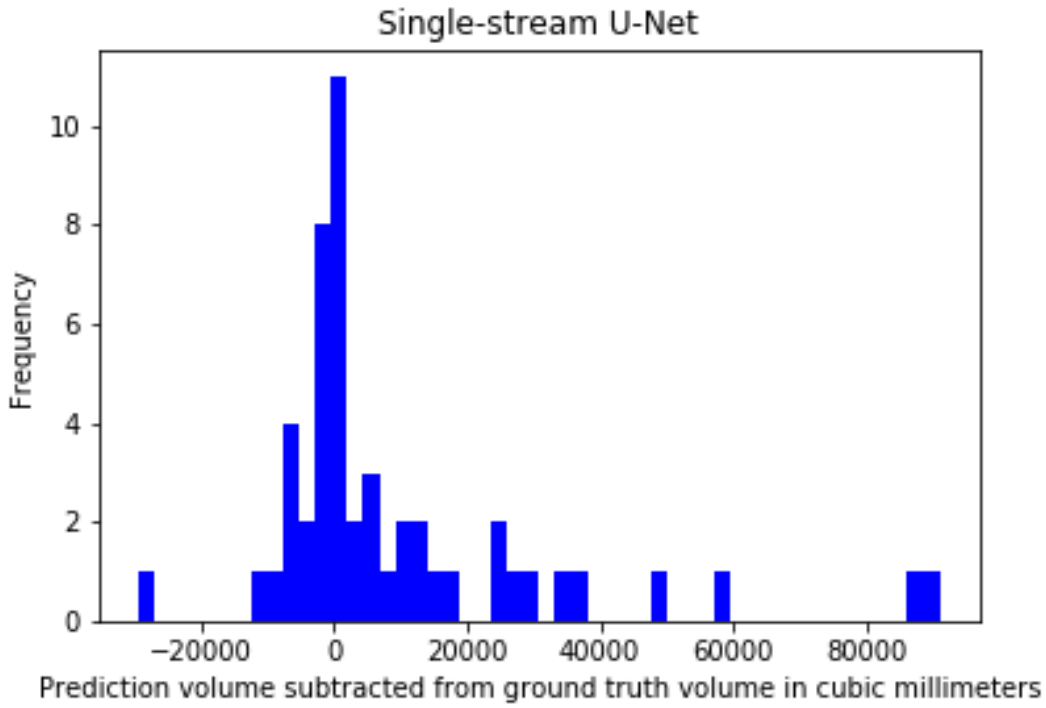


Figure 4.33: Volume difference between prediction and ground truth for Single-stream U-Net

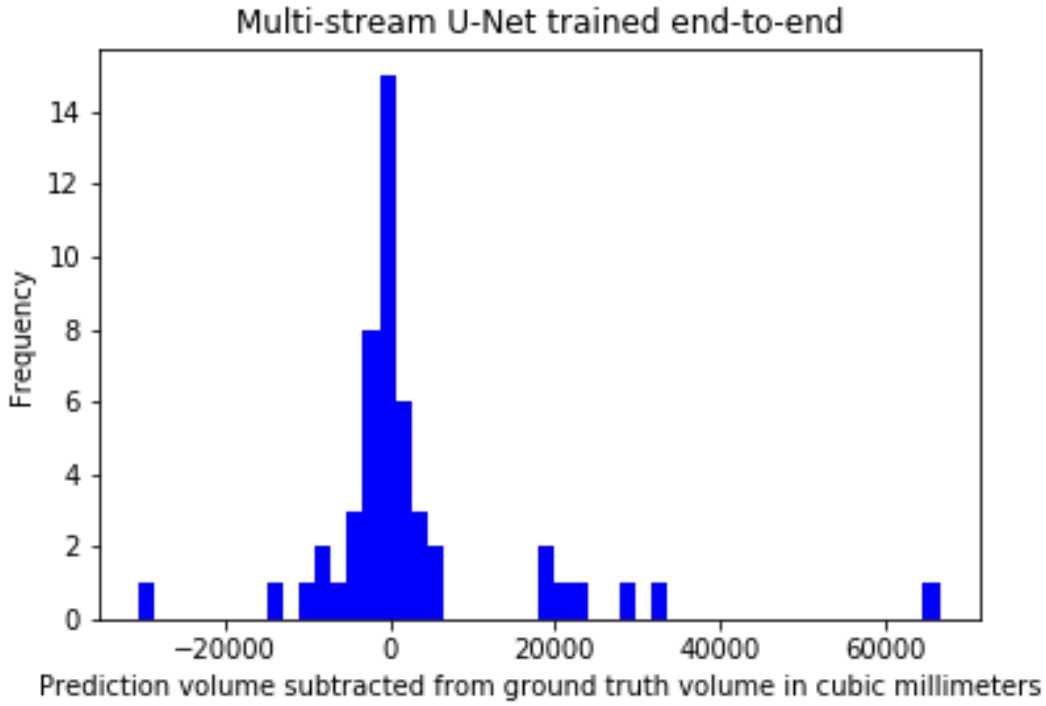


Figure 4.34: Volume difference between prediction and ground truth for Multi-stream U-Net trained end-to-end

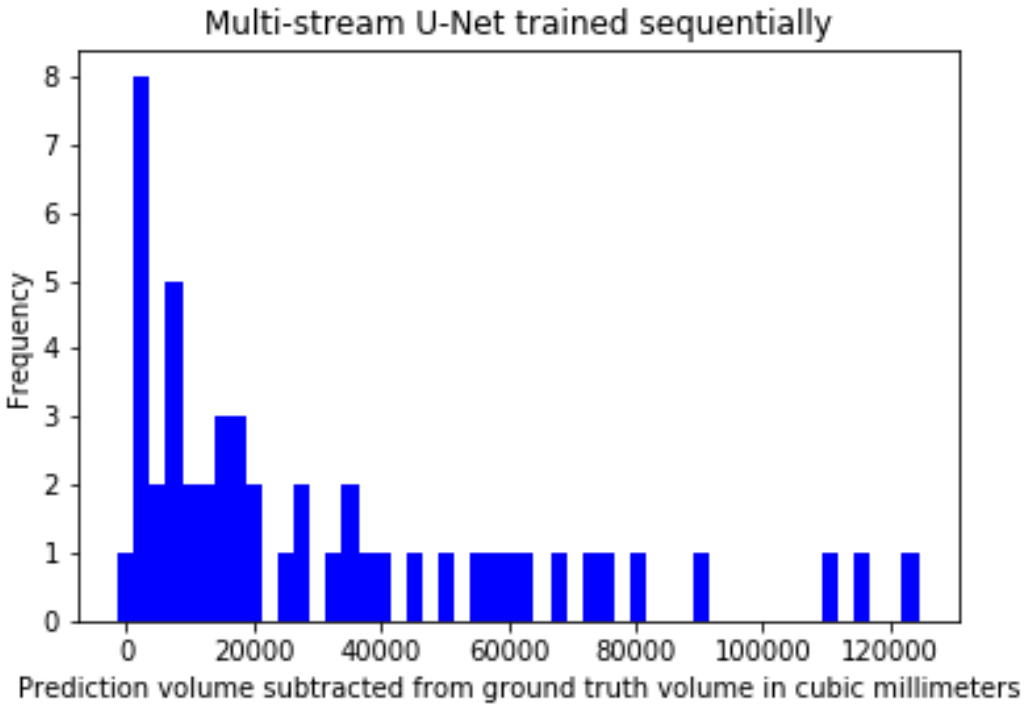


Figure 4.35: Volume difference between prediction and ground truth for Multi-stream U-Net trained sequentially

Volume visualization

The tumor of patient 52 is shown from different angles in 3D to visualize the whole performance of the Multi-stream U-Net trained end-to-end. The volume of the tumor is depicted in Figure 4.36 and the colors, navy blue, purple, yellow corresponds to edema, necrosis and enhancing tumor respectively.

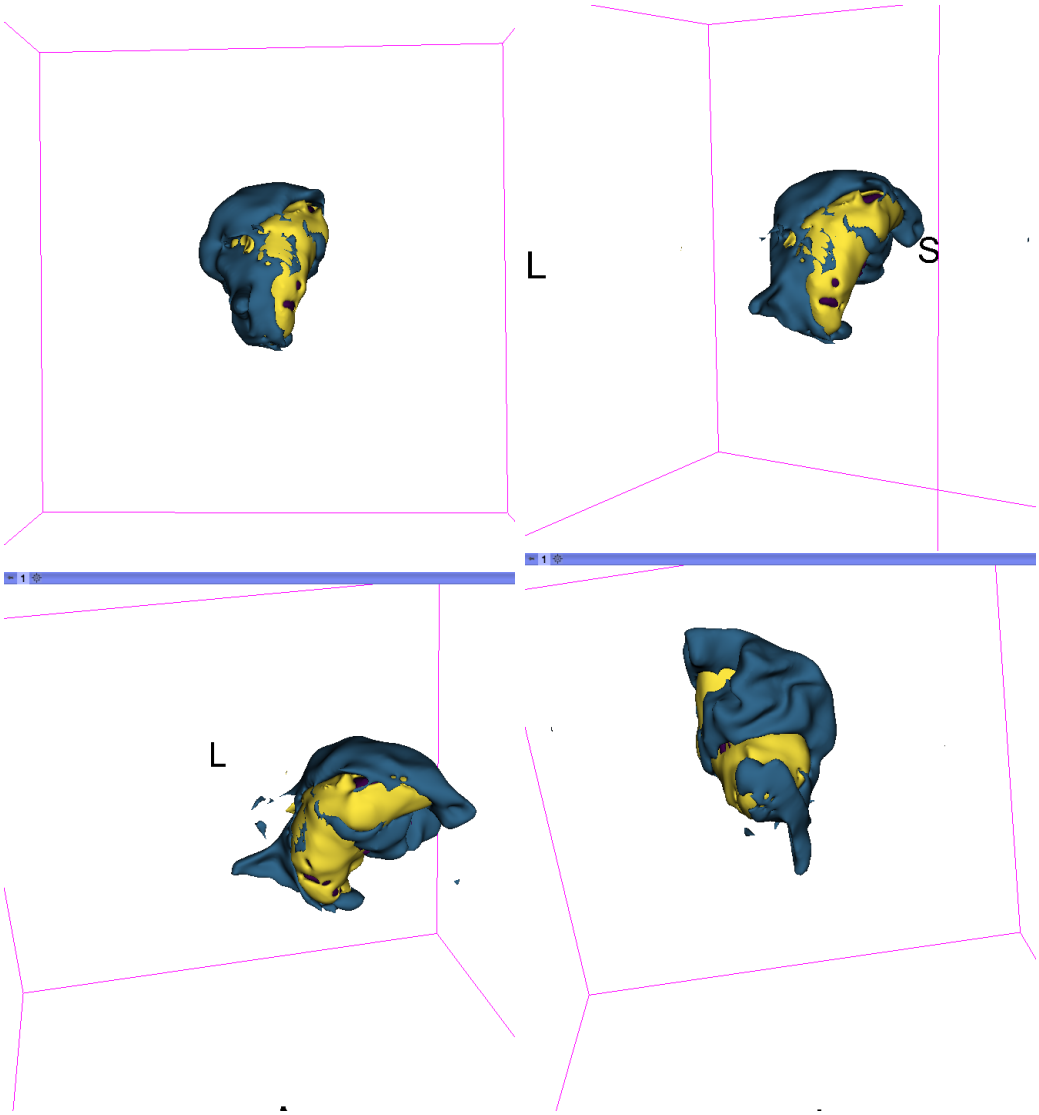


Figure 4.36: Segmentation results of the Multi-stream U-Net on patient 52 depicted in 3D from different angles

5
Discussion

5.1 Experimental design

This thesis compares two different types of network architectures and how different regularization techniques influences the training process of the Single-stream U-Net. It also looks at how training the Multi-stream U-Net in two different ways affects performance. The Multi-stream U-Net trained end-to-end has the best performance.

A challenging and time consuming part of the thesis was to acquire the computational resources needed for training of the networks. Due to little prior experience in training ANNs on large datasets a flexible solution was needed to be able to test run the experiments at any time hence time bookings on clusters was not a viable option. If less time had been spent on computational solutions for training a deeper analysis of the regularization techniques and hyperparameter tuning for the networks could have been performed.

The experimental setup for the Single-stream U-Net was to add each regularization technique on top of the other. It is not known how the networks would perform if the regularization techniques would have been implemented independently. Another way of setting up the experiment could have been to try each regularization technique separately and then combine the best performing techniques in the training of a network and observe the results. When working with ANNs there are many factors that affect the performance of the network which is why the trial and error approach is most commonly used when it comes to regularization techniques and hyperparamter tuning.

When training these networks there are many regularization techniques and hyperparamters that can be tested and tuned. For each new regularization technique and hyperparameter a new network needs to be trained. A more rigorous approach would be to try all combinations of regularization techniques and hyperparamters, this however quickly becomes unfeasible due to the time consuming task of training a network and the combinatorial explosion of combinations of hyperparameters and regularisation techniques.

Due to patient integrity the datasets are relatively small when it comes to medical images in general and specifically in the brain tumor case. A larger dataset means more examples to learn from which increases the performance of the network. The

MICCAI BraTS dataset is one of the bigger brain tumor dataset but is still a quite small dataset when it comes to deep learning. The dataset is a bench mark dataset for the task of brain tumor segmentation. By including other brain tumor dataset a larger dataset could have been obtained for training the network which would most probably increase the performance of the network.

The batch size used in this project was 16, this was set by trial and error hence when using a larger batch size we ran into computational problems. Different results might have been obtained by having a larger batch size. If more computational resources could be obtained then experiments could be done on how the batch size affect the training and performance of the networks.

5.2 Results

Adding the regularization techniques on top of one another did not lead to an increased performance in all of the Single-stream U-Net experiments. The performance difference from one regularization technique to the other is in some cases quite small. Looking at the difference in performance after adding L2 regularization the accuracy is decreased by 0.5%. This decrease could have been from the stochastic properties of the ANN. If the L2 regularization technique would have been removed it is possible that the network would have performed better. Running the experiment multiple times would have increased the reliability and reproducibility of our results.

When acquiring the T1ce MR images a contrast fluid containing Gadolinium is injected into the patient which gives increased contrast to the tumor and among others also blood vessels. From the predicted segmentations it can be seen that the networks segments these high contrast blood vessels as tumors in some cases. Hence it seems like the networks learns that high contrast areas are associated with tumors. To deal with the issue of segmenting high contrast non tumor tissue as tumor tissue, one might be able to do image preprocessing in the areas of the vessels. These areas are located in the same location in the head which makes this preprocessing theoretically feasible. By lowering the image contrast in these areas in the images better segmentation results might be achieved.

Further the networks struggles with finer details in tumor borders, this could be overcome by adding extra weight to the border pixels of the different classes in the ground truth. Adding extra weight to the boarder pixels would penalize the network more for wrongly classifying border pixels which could hopefully drive it towards completing this task more successfully.

There is a big variety in brain tumor's location and shape. Having a bigger dataset means that the network would see more cases hence becoming better at predicting similar cases which it has not seen before. To deal with problem of data scarcity data augmentation was used, this increased the variety in the data and seems to have a good impact on the training process. Even better results would probably be obtained if the dataset was even bigger and data augmentation was used.

Inspecting the predicted segmentation results it can be seen that there are one or just a few pixels that are classified as tumor in some cases when there is no tumor in the ground truth. These false positives do in some cases also appear scattered. A few pixels scattered over the brain is very rare in the case of brain tumors. By using image postprocessing these false positives could be removed. This could be achieved by setting a threshold for the number of pixels that need to be present and have boundaries with each other. A simple postprocessing step like this would increase the segmentation performance even further.

The volume estimation indicated that the Single-stream U-Net was biased towards under estimating the tumor whereas the Multi-stream U-Net trained end-to-end had about the same number of over and under estimations. The sequentially trained Multi-stream U-Net was heavily under estimating the tumor volumes and performed a lot worse than both the other networks. In a clinical setting the volume of the tumor is an important parameter for treatment planning. By using deep learning for segmentation the human bias can be minimized. With an increased consistency in segmentation's a better estimation of volume difference between two time instances can be obtained. In today's healthcare many decisions are based on a combination of fact and experience. With the help of AI and deep learning decisions can be more data driven which lessens the room for human error.

If one compares the results between the two different ways of training the Multi-stream U-Net it is clear that the end-to-end approach is advantageous. A qualitative argument for why this might be the case is that some classes are really hard to see in certain modalities, for example edema in T1. This makes the task of identifying all four classes too complex to be performed using only a single modality. A more nuanced approach could be to only try to segment the classes which are clearly visible in a given modality, something that the authors does not have enough domain knowledge about and is therefore not done.

5.3 Future work

In this thesis we used the categorical crossentropy loss function which does not take the class imbalance in to consideration. An investigation on how different loss functions such as weighted categorical cross entropy and dice loss affects the performance of the segmentation network would be interesting.

When working with MR images the intensity scales differs between scanners and intensity does not map directly to a tissue. There are many different ways that the MR images could be normalized and it is not know how the normalization of the MR images effect the training and performance of the network. An investigation looking at how different normalization techniques influences training and segmentation performance would be interesting to gain further understanding.

In the encoder path of the network the data is downsampled with the pooling operation at each layer. In this thesis we used four downsamplings. It is not known how the performance of the network would change if the number of downsamplings was reduced or increased. An investigation of how the number of downsamplings affect the training and performance of the segmentation network would be interesting to conduct.

The application created in this thesis requires an understanding of python programming and has not been designed to be used in a clinical setting. The segmentation results obtained are promising but not at the level of an experienced radiologist. From a meeting with an experienced neurosurgeon at Sahlgrenska University hospital an application that could be used for pre-segmentation of brain tumors by the doctors would decrease the time for segmentation. A next step for our program could be to implement a user interface where doctors can process MR images, get segmentation predictions by the network and the edit the segmentations. A program used today by the doctors at Sahlgrenska is 3D slicer, this program is open source and an automatic segmentation module could hence be created to include the features described.

A further complication for clinical use of the proposed architectures, is that when patients are screened only one or two of the four modalities used in this thesis are captured. Which means that the architectures presented in this thesis cannot be used without some sort of data synthesis or retraining of the network on a certain modality.

In healthcare today much of the decisions taken are based on a combination of fact and experience. With the healthcare becoming more digitized a shift can be made to have even more data driven decisions rather then decisions made on experience. For the healthcare to become more data driven in their decisions it is believed that nation wide guidelines should be setup with in the healthcare sector on how medical information should be saved so that it can be used for machine learning tasks. Doing this would make it easier to use the data for machine learning tasks hence excelling this field within healthcare. There are multiple things that need to be discussed, some examples are; how the data should be anonymized, what file format should be used and what resolution it should be interpolated to. By having a more data driven decision process the human error can be minimized, increasing the chance of survival.

6
Conclusion

In brain tumor segmentation it is of great importance to segment out the brain tumor with high precision if it is to be used in a clinical setting. If a high precision can be ensured deep learning segmentation could at this time be used as a second rater in the task of segmentation. A long term goal of deep learning segmentation of medical images is to relieve the doctors fully from this time consuming task. From the research conducted in this thesis the following conclusions can be made.

1. The cropping of the MR images decrease the amount of pixels thereby increasing the speed of training while maintaining competitive segmentation results.

2. The novel Multi-stream U-Net architecture trained end-to-end outperformed the conventional approach.

3. Using deep learning decreases the variance in segmentations. A more consistent estimation of the tumor volume can be made, yielding a better understanding of tumor progression.

From this thesis there are a number of ideas for future work.

1. Development of a user interface that is adapted for clinical use where healthcare personnel can process MR images to get segmentation's from the network and edit the segmentation results.

2. Further investigation on neural network architectures for example how the number of downsamplings in the contracting path of the U-Net affects performance. Looking at different loss functions such as weighted categorical cross-entropy and dice loss. Pre-processing of MR images with for example intensity normalization. Investigating algorithms for giving extra weight to pixels at the border between different tumor classes to better predict the finer details of segmentations.

Using deep learning for segmentation of medical images has been pushing forward fast thanks to a growing body of publicly available resources. It is believed that segmentation of medical images with deep learning will continue to increase in performance and in a not so distant future become a natural part of the clinical work.

References

[1] World Health Organization. *Global Health Observatory. Geneva: World Health Organization*. URL: https://apps.who.int/iris/bitstream/handle/10665/272596/9789241565585-eng.pdf?ua=1. (accessed: 03.09.2019).

[2] Rebecca Leece et al. "Global incidence of malignant brain and other central nervous system tumors by histology, 2003–2007". In: *Neuro-Oncology* 19.11 (May 2017), pp. 1553–1564. ISSN: 1522-8517. DOI: 10.1093/neuonc/nox091. eprint: http://oup.prod.sis.lan/neuro-oncology/article-pdf/19/11/1553/22934243/nox091.pdf. URL: https://doi.org/10.1093/neuonc/nox091.

[3] Laura J. Martin MD. *Types of Brain Cancer*. URL: https://www.webmd.com/cancer/brain-cancer/brain-tumor-types#1. (accessed: 04.09.2019).

[4] Hao Dong et al. "Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks". In: *annual conference on medical image understanding and analysis*. Springer. 2017, pp. 506–517.

[5] Mohammad Havaei et al. "Brain tumor segmentation with deep neural networks". In: *Medical image analysis* 35 (2017), pp. 18–31.

[6] H A Haenssle et al. "Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists". In: *Annals of Oncology* 29.8 (May 2018), pp. 1836–1842. ISSN: 0923-7534. DOI: 10.1093/annonc/mdy166. eprint: http://oup.prod.sis.lan/annonc/article-pdf/29/8/1836/25682281/mdy166.pdf. URL: https://doi.org/10.1093/annonc/mdy166.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[8] B. H. Menze A. Jakab S. Bauer J. Kalpathy-Cramer K. Farahani J. Kirby et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)". In: *IEEE Transactions on Medical Imaging* 34.10 (2015), pp. 1993–2024. DOI: 10.1109/TMI.2014.2377694.

[9] S. Bakas H. Akbari A. Sotiras M. Bilello M. Rozycki J.S. Kirby et al. "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features". In: *Nature Scientific Data* 4.170117 (2017). DOI: 10.1038/sdata.2017.117.

[10] S. Bakas M. Reyes A. Jakab S. Bauer M. Rempfler A. Crimi et al. "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation,

Progression Assessment, and Overall Survival Prediction in the BRATS Challenge". In: *arXiv preprint* (2018).

[11] McKinsey L.GoodenbergerRobert B.Jenkins. *Genetics of adult glioma*. URL: https://doi.org/10.1016/j.cancergen.2012.10.009. (accesed: 04.09.2019).

[12] David N. Louis et al. "The 2016 World Health Organization Classification of Tumors of the Central Nervous System: a summary". In: *Springer* (2016). DOI: 10.1007/s00401-016-1545-1.

[13] *Cancer of the Brain and Other Nervous System - Cancer Stat Facts*. URL: https://seer.cancer.gov/statfacts/html/brain.html.

[14] Wikimedia Commons. *Annotated Neuron*. 2008. URL: https://commons.wikimedia.org/wiki/File:Neuron_-_annotated.svg.

[15] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 315–323.

[16] Bing Xu et al. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016. ISBN: 0262035618, 9780262035613.

[18] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. "Learning representations by back-propagating errors". In: *Cognitive modeling* 5.3 (1988), p. 1.

[19] B. Mehlig. *Artificial Neural Networks*. 2019. arXiv: 1901.05639 [cs.LG].

[20] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[21] Yann LeCun et al. "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.

[22] Wikimedia Commons. *Convolutional Neural Network*. 2014. URL: https://commons.wikimedia.org/wiki/File:Convolutional_Network.png.

[23] François Chollet et al. *Keras*. https://keras.io. 2015.

[24] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[26] Murino L. Larobina M. "Medical Image File Formats". In: *J Digit Imaging* 27 (2014), pp. 200–206. DOI: 10.1007/s10278-013-9657-9.

[27] Case Western Reserve University. *MRI Basics*. 2006.

[28] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. URL: https://arxiv.org/pdf/1603.04467.pdf. (accesed: 13.12.2019).

[29] Kalavathi Palanisamy and Surya Prasath. "Methods on Skull Stripping of MRI Head Scan Images—a Review". In: *Journal of Digital Imaging* 29 (Dec. 2015). DOI: 10.1007/s10278-015-9847-8.

[30] Iglesias JE. Liu CY. Thompson P. Tu Z. “Robust Brain Extraction Across Datasets and Comparison with Publicly Available Methods”. In: *IEEE Transactions on MedicalImaging* 30(9) (2011), pp. 1617–1634.

[31] Barbara Zitová and Jan Flusser. “Image registration methods: a survey”. In: *IMAGE AND VISION COMPUTING* 21 (2003), pp. 977–1000.

[32] Erik H. W. Meijering et al. “Quantitative Comparison of Sinc-Approximating Kernels for Medical Image Interpolation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI’99*. Ed. by Chris Taylor and Alain Colchester. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 210–217. ISBN: 978-3-540-48232-1.

[33] Vincent Dumoulin and Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).

A

Appendix

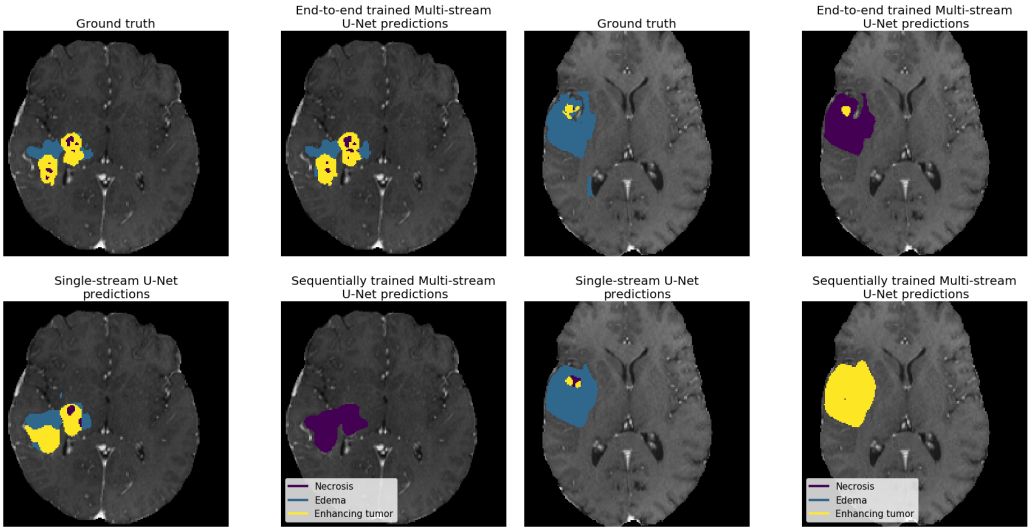
A.1 Code and models

Link to github repository:
https://github.com/Laaggan/Master_thesis

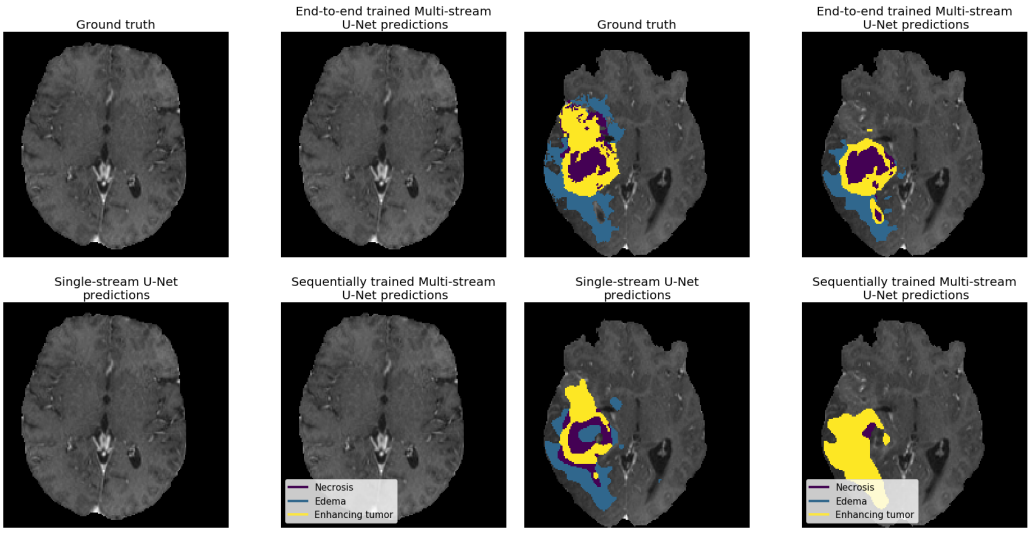
Link to trained Keras models:
https://drive.google.com/drive/folders/1_bwPKMN2R7OXwfuVn8ThqN8VnUDTffEG

A.2 Predictions of the different network architectures

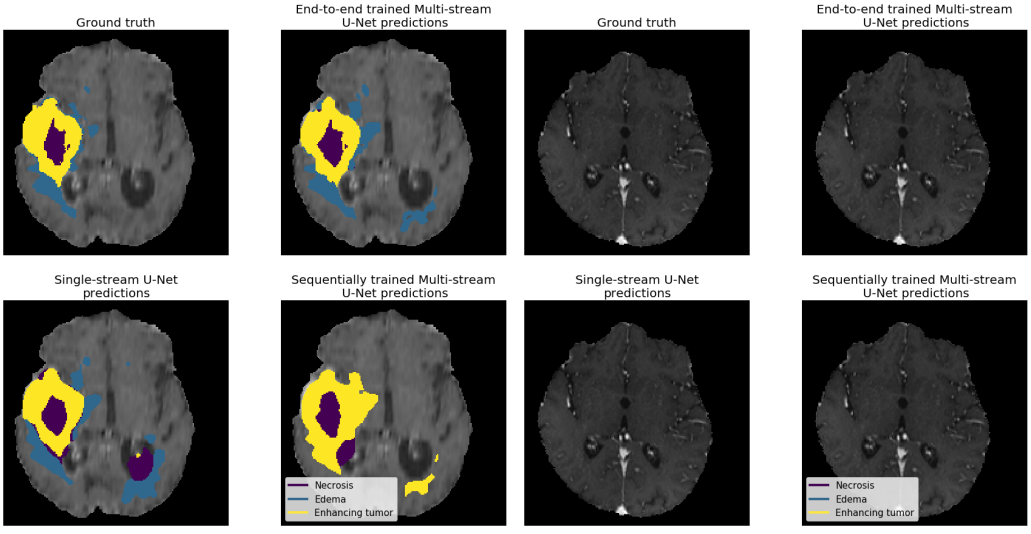
The segmentation results from the network architectures experimented with and the ground truth are depicted. The slices shown are from the test dataset and consists of slice 70 from all the patients.



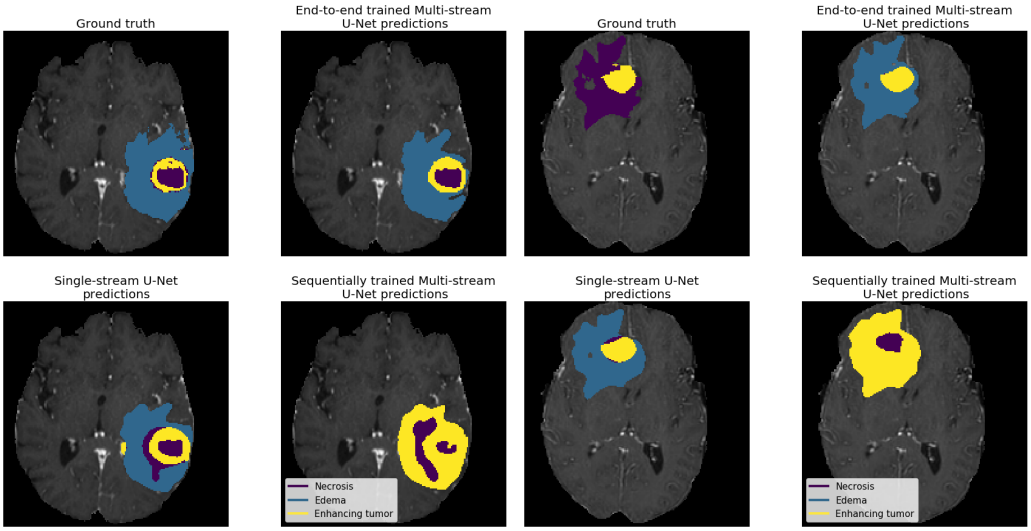
(a) Patient 102 **(b)** Patient 106



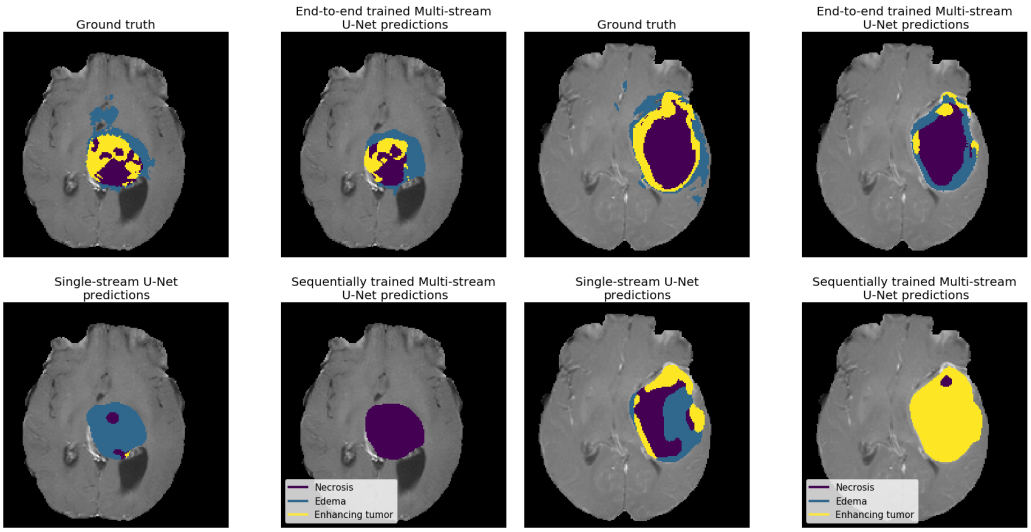
(c) Patient 121 **(d)** Patient 13



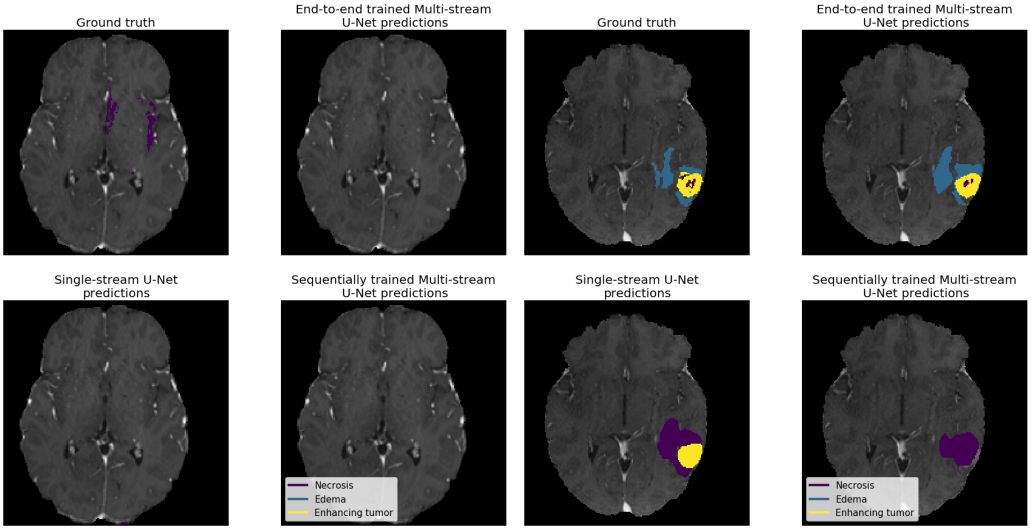
(e) Patient 130 **(f)** Patient 134



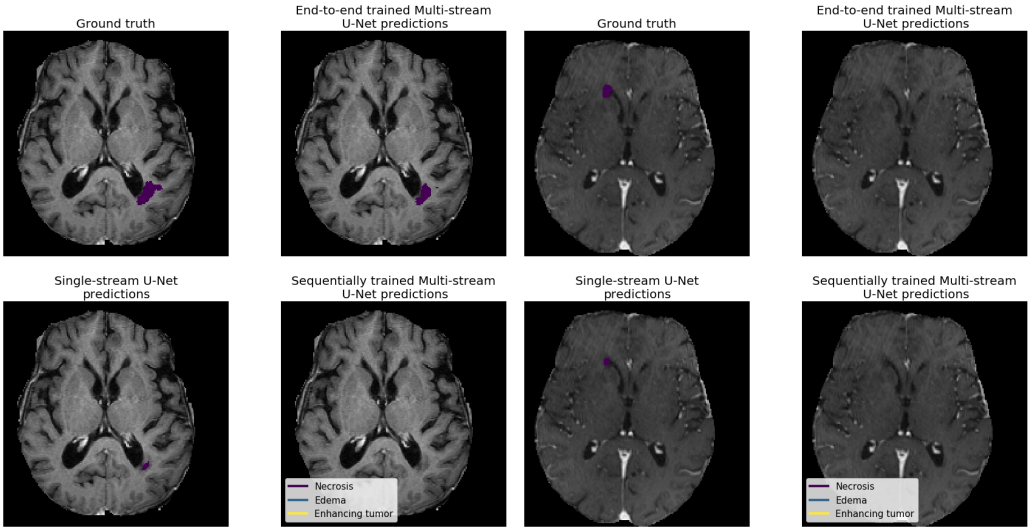
(a) Patient 149 **(b)** Patient 151



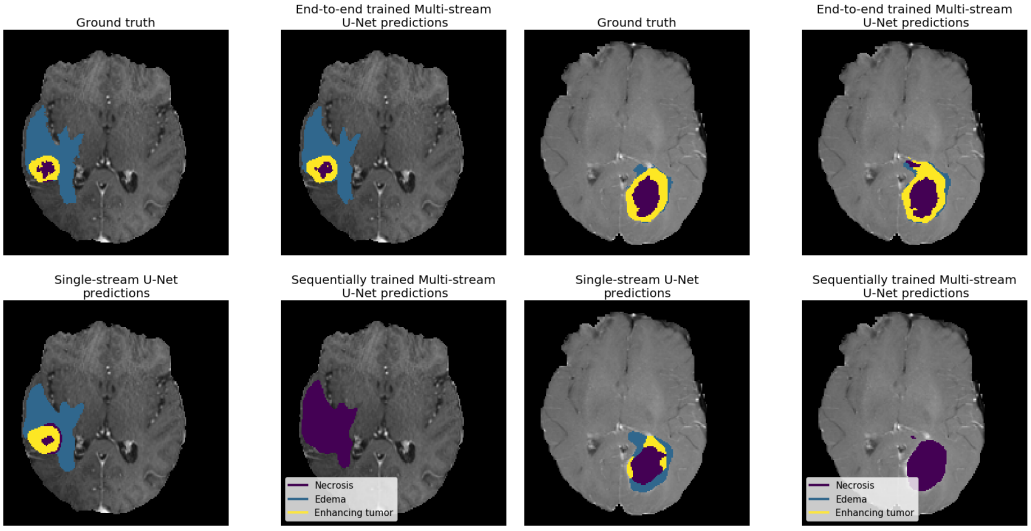
(c) Patient 160 **(d)** Patient 166



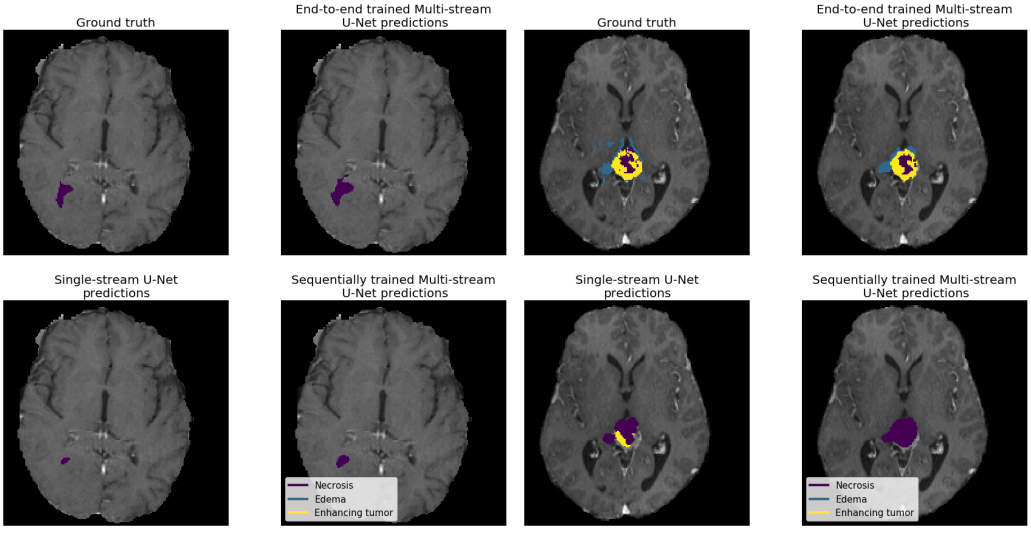
(e) Patient 169 **(f)** Patient 174



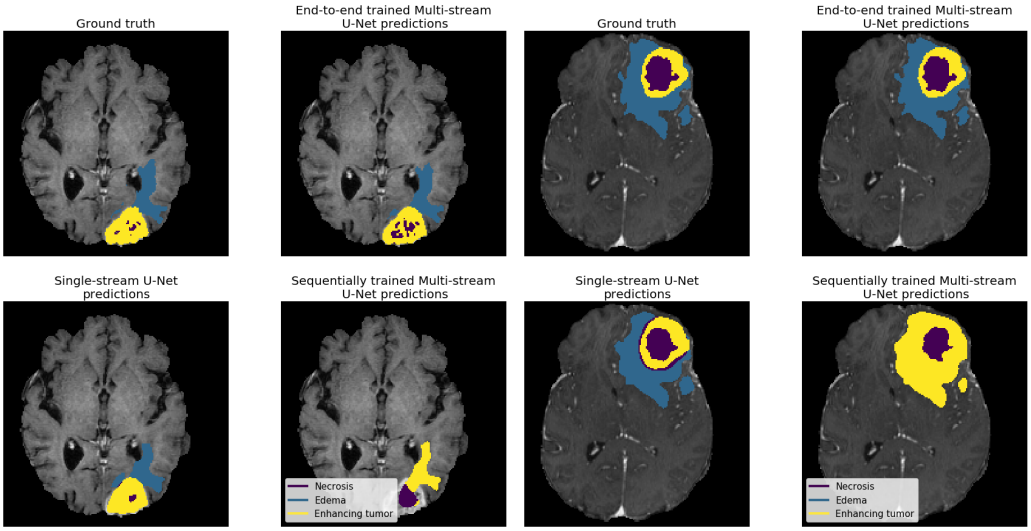
(a) Patient 187 **(b)** Patient 188



(c) Patient 189 **(d)** Patient 191

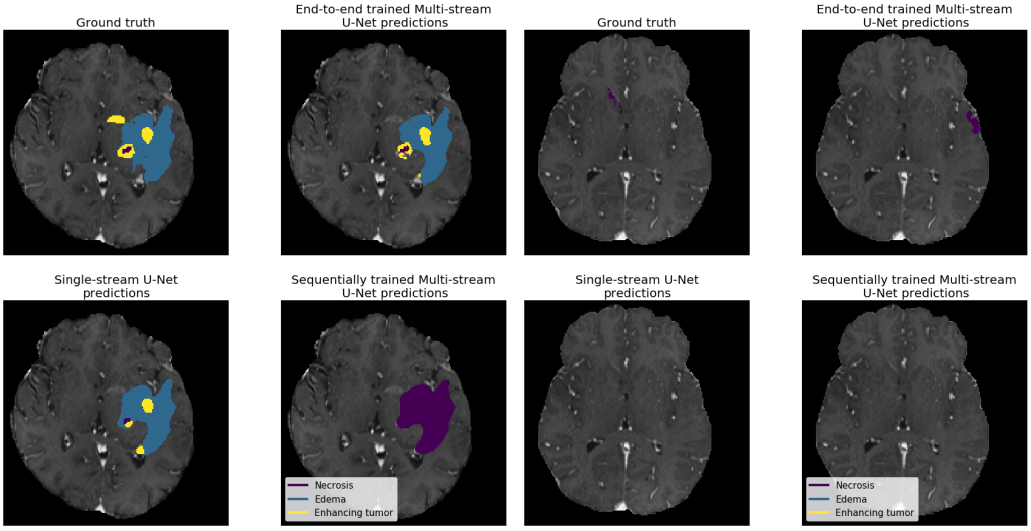


(e) Patient 20 **(f)** Patient 205



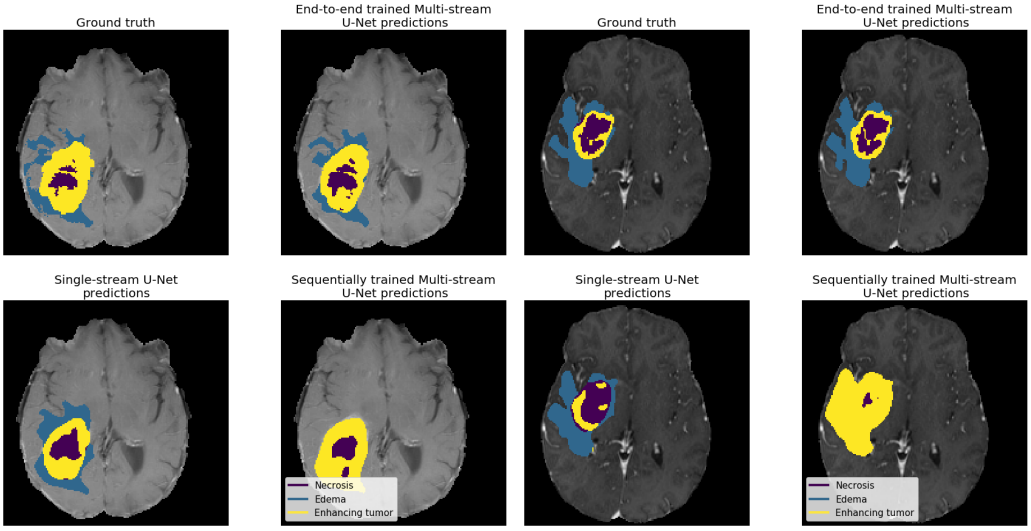
(a) Patient 21

(b) Patient 214



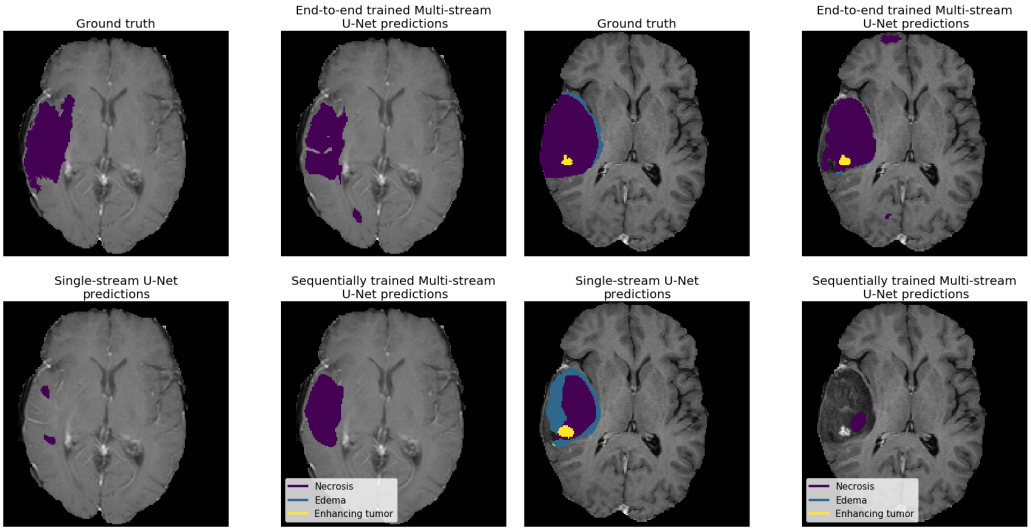
(c) Patient 235

(d) Patient 241



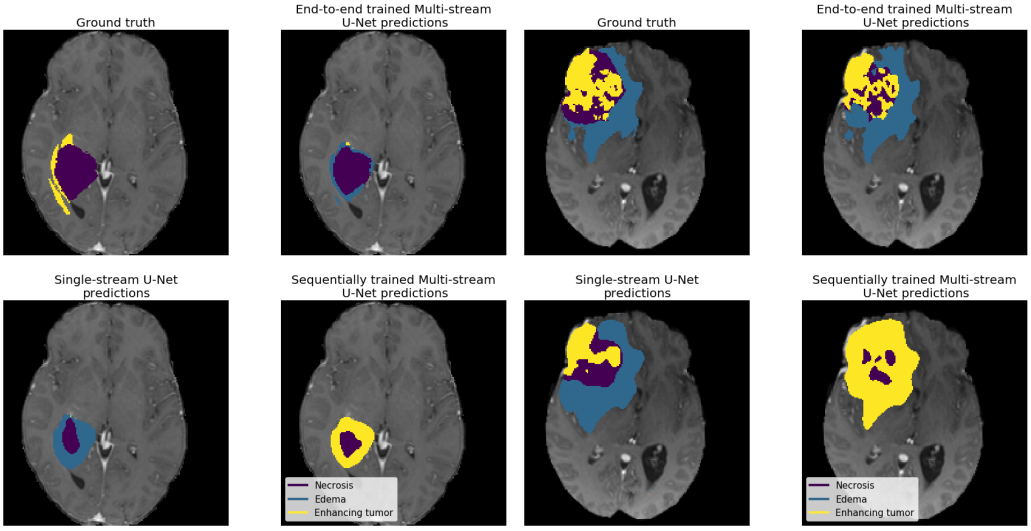
(e) Patient 243

(f) Patient 252



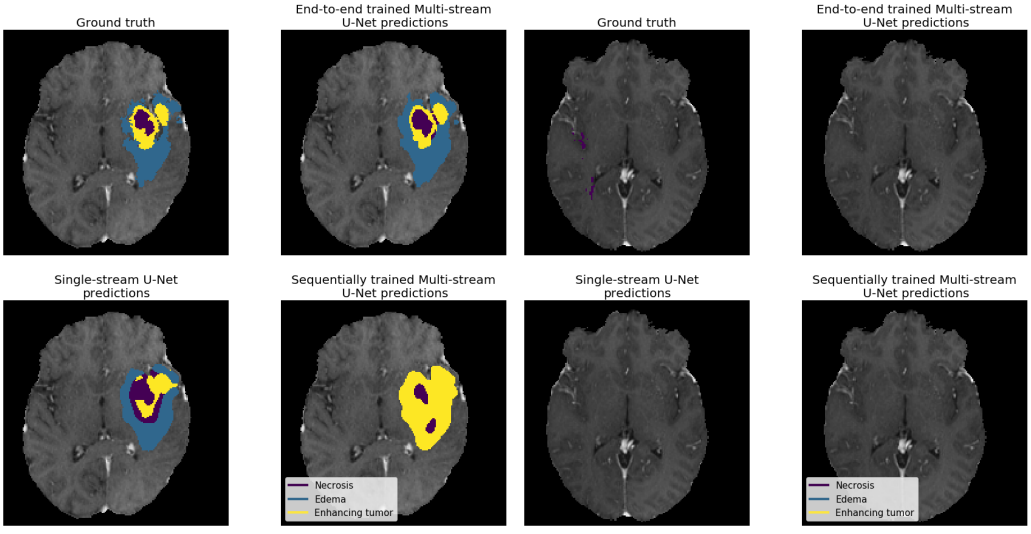
(a) Patient 257

(b) Patient 263



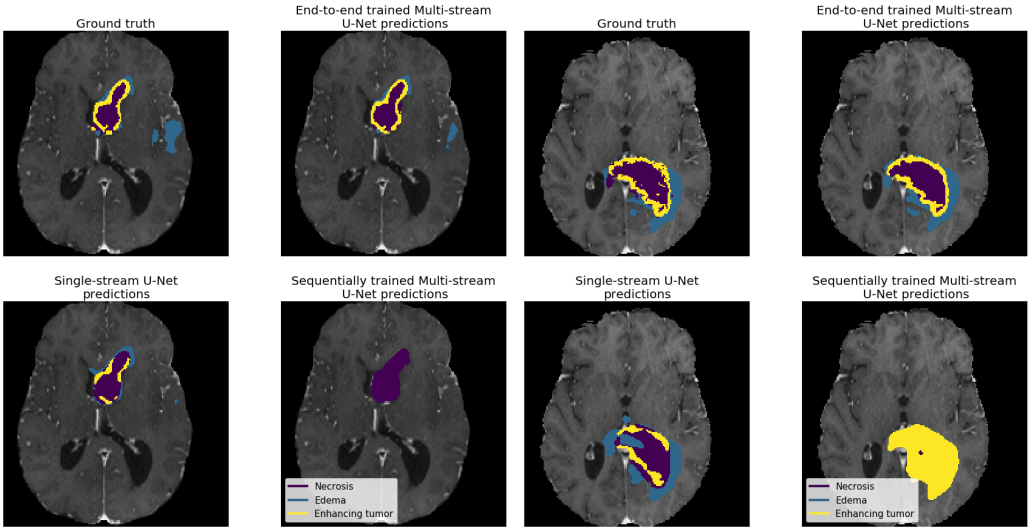
(c) Patient 264

(d) Patient 34



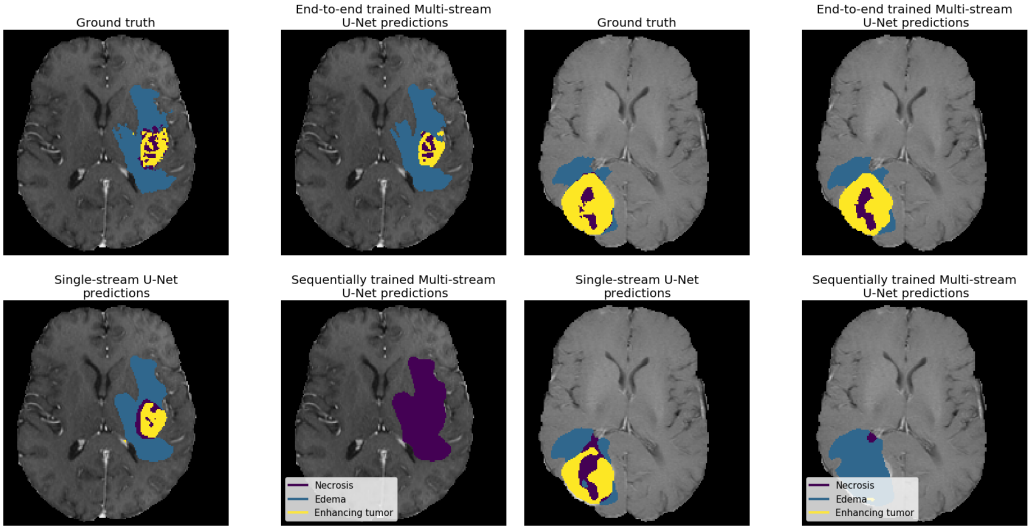
(e) Patient 48

(f) Patient 49



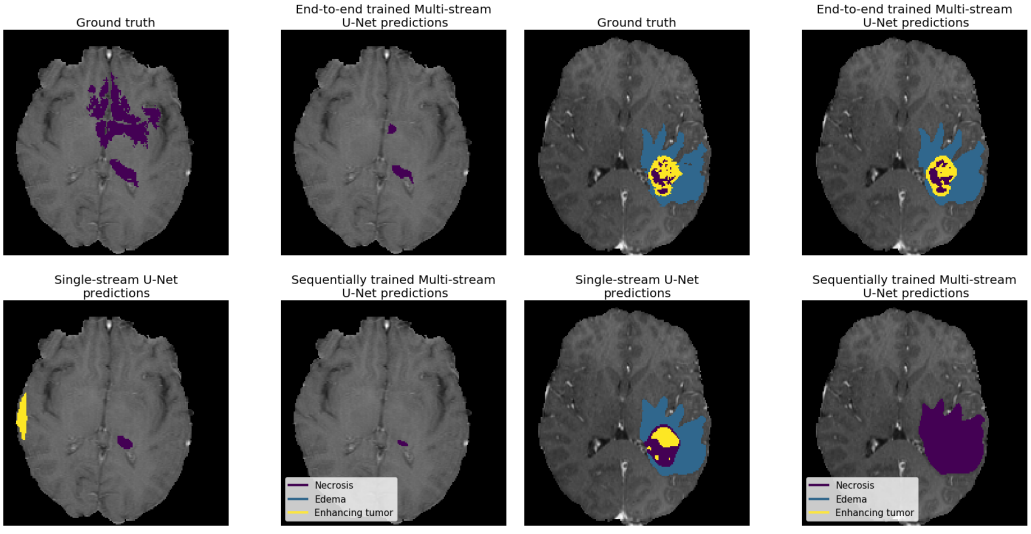
(a) Patient 50

(b) Patient 52



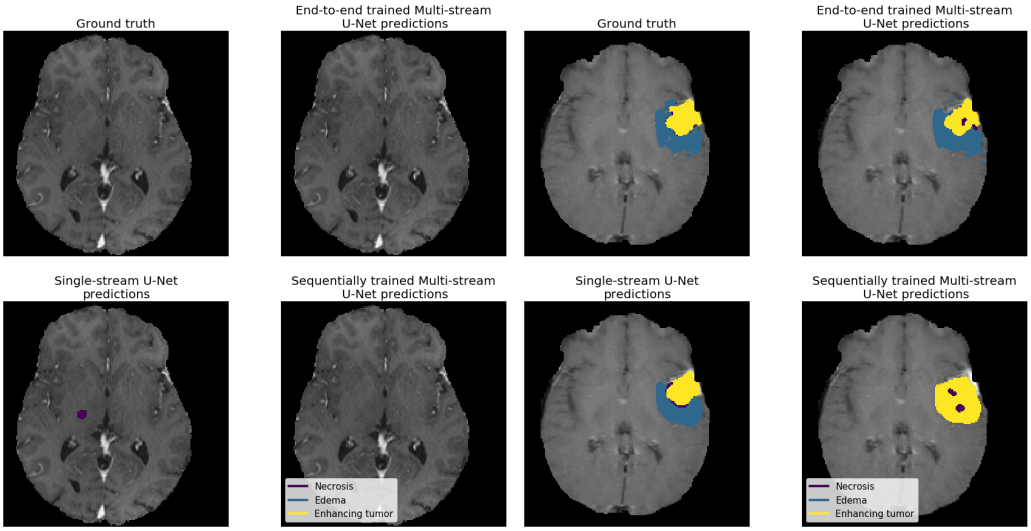
(c) Patient 54

(d) Patient 58

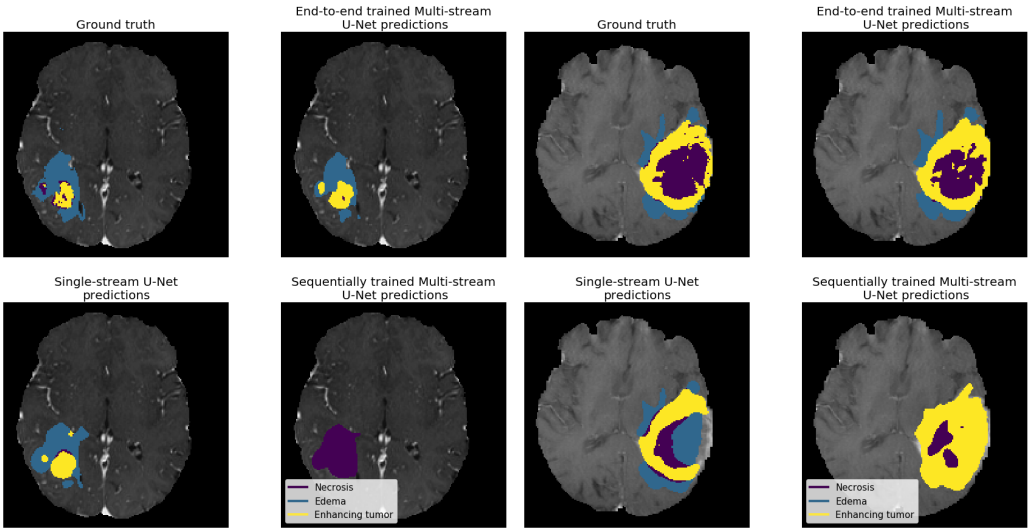


(e) Patient 71

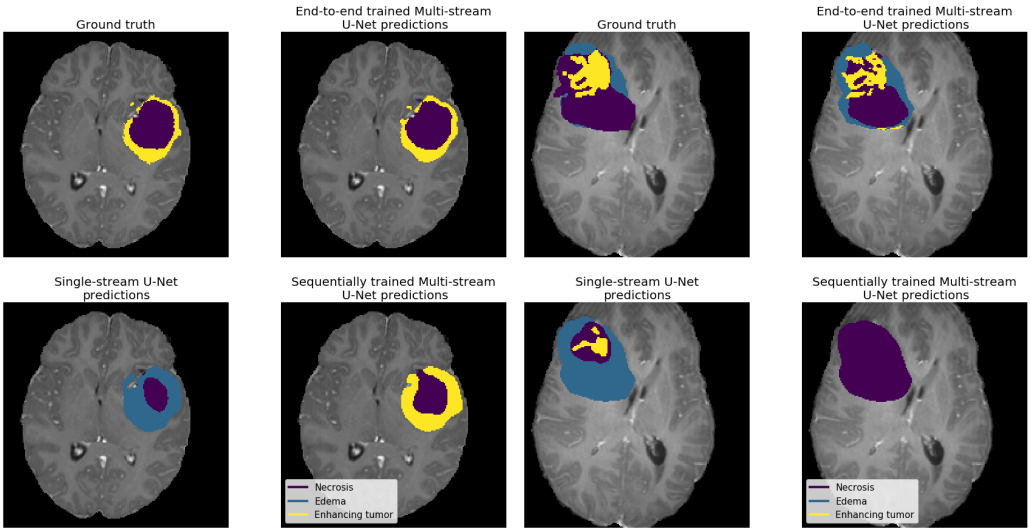
(f) Patient 80



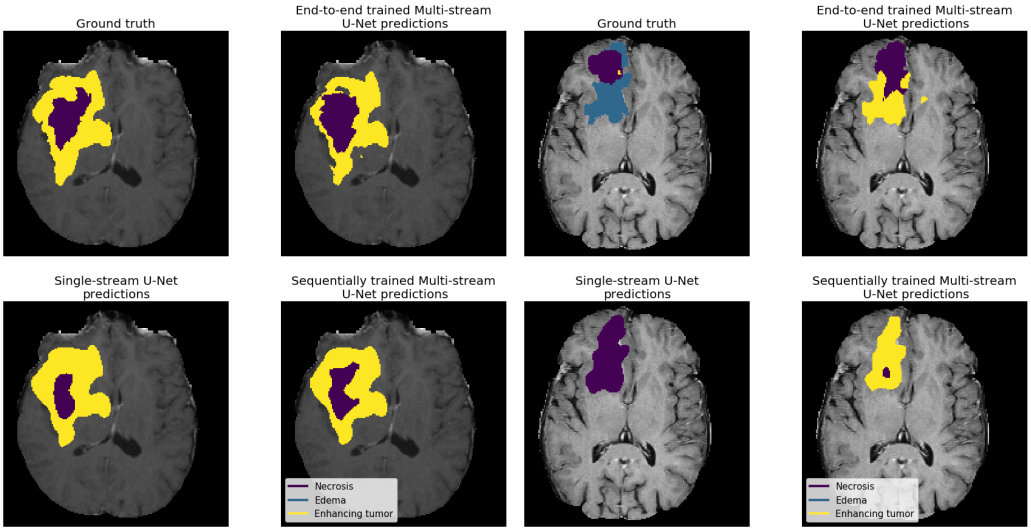
(a) Patient 87 **(b)** Patient 88



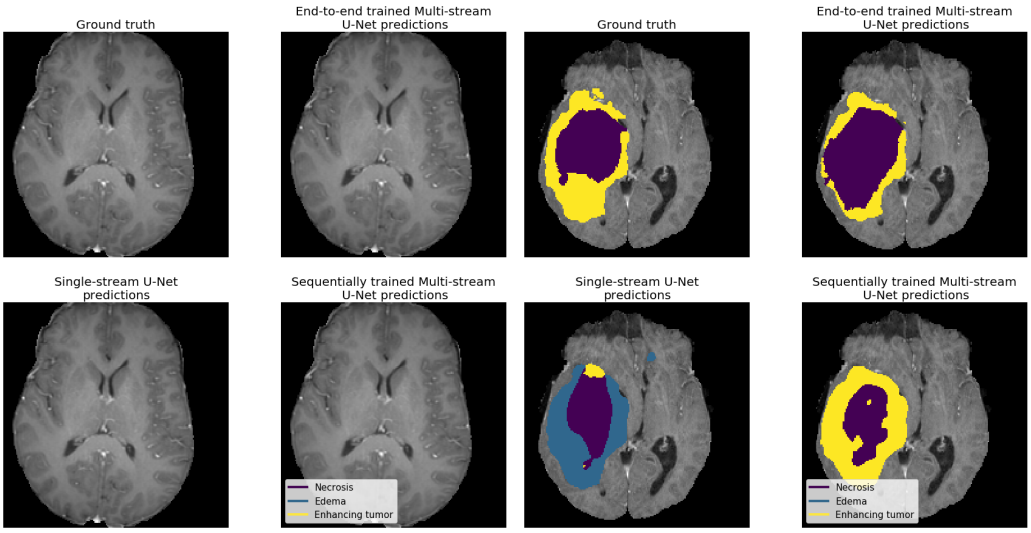
(c) Patient 91 **(d)** Patient 99



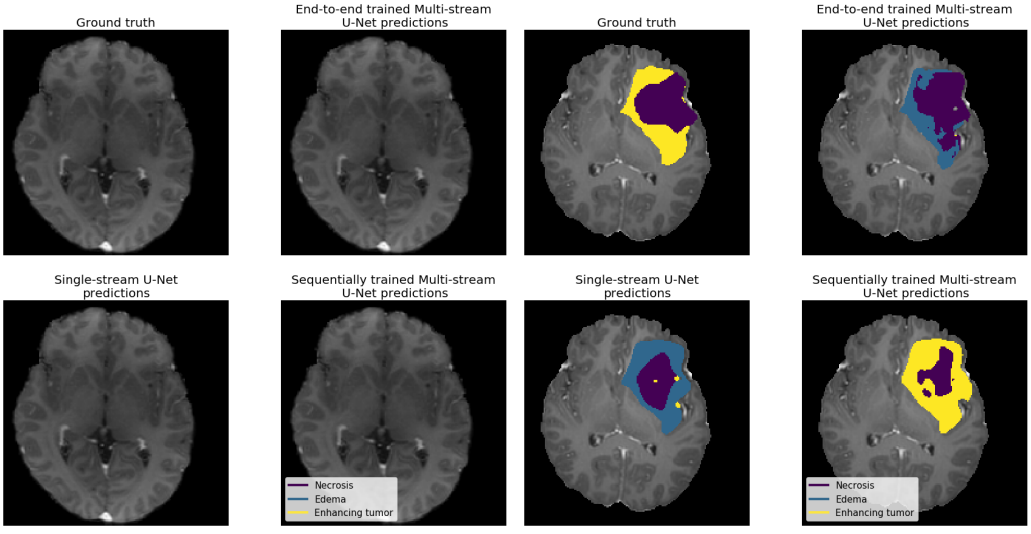
(e) Patient 334 **(f)** Patient 333



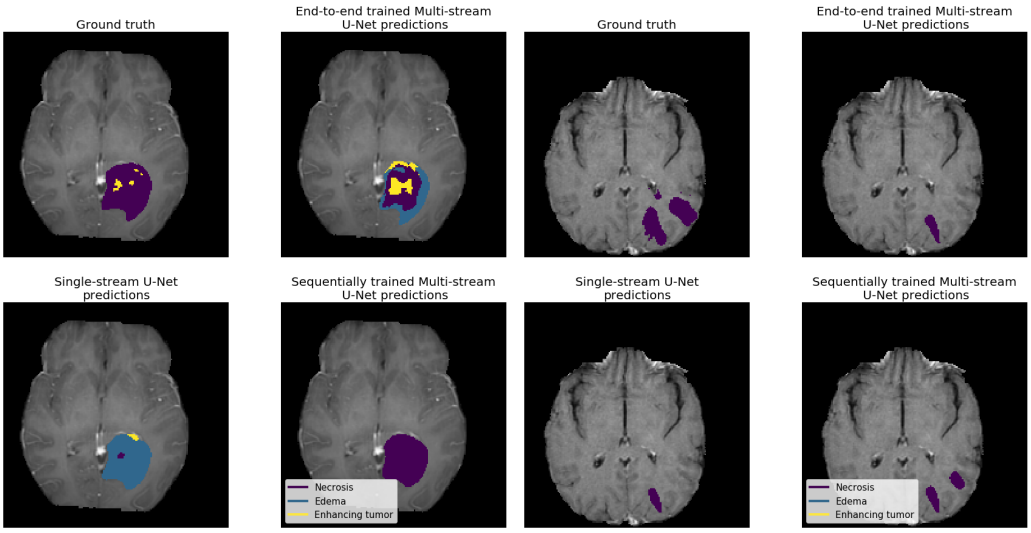
(a) Patient 329 (b) Patient 322



(c) Patient 313 (d) Patient 308



(e) Patient 293 (f) Patient 276



(a) Patient 273 **(b)** Patient 270