FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation:

Data Science

The spring semester, 2022

Open / ~~Confidential~~

Author: Sander Takvam

Course coordinator:

Supervisor(s): Morteza Esmaeili

Thesis title: Brain Tumor Segmentation and Classification Using Neural Networks

Credits (ECTS): 30

Keywords:

ANN, CNN, ML, Brain, MRI, BraTS

Pages: 74

+ Appendix: 9

Stavanger, 06/2022

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

List of Tables

Abstract

Magnetic Resonance Imaging (MRI) is widely used in the diagnostic and treatment evaluation of brain tumors. Segmentation is a critical step of the tumor assessment, which usually is a time-consuming task by conventional image analysis methods. In this thesis, I utilized deep learning methods to automate the tumor segmentation and classification tasks. Two models were used, a segmentation model and a classification model. I used U-Net for the segmentation task and a Convolutional Neural Network followed by fully connected layers for the classification task. I evaluated networks on the Multimodal Brain Tumor Segmentation Challenge 2020 (BraTS 2020) dataset. Image slices were sampled from the axial axis using three modalities, T1-Contrast-Enhanced, T2-weighted, and Fluid-attenuated inversion recovery. 2-dimensional image slices were used for training in the segmentation task, and annotated images were used for training during the classification task.

Acknowledgements

I want to thank my supervisor, Morteza Esmaeili, for his guidance and feedback throughout this thesis. His knowledge and insight has been invaluable and truly appreciated.

Chapter 1

Introduction

Gliomas are one of the most common types of malignant brain tumors in adults and even though they are rare, they have a high rate of mortality. World Health Organization (WHO) uses 4 grades of gliomas where grades I and II are classified as low-grade gliomas, and grade III/IV are classified as high-grade gliomas, depending on their level of aggressiveness where the most aggressive glioma, gliomablastoma, has a median survival of 15 months and a survival rate that is less than 5% at five years [18, 12]. Gliomas is commonly treated with chemotherapy and radiation, where the treatment is highly dependent on the grade of the tumor diagnosed.

Magnetic resonance imaging (MRI) plays a vital role in diagnosing and treatment evaluation of glioma tumors, with help of medical professionals and neuroradiologists. As gliomas differ in aggressiveness while also differing in shape, size and location, the task of segmenting MR images can be challenging due to the different types of gliomas having these heterogeneous properties. To assist the professional in the process, a variety of MRI modalities with differing amounts of contrast can be used. MR images with different amounts of contrast highlight different types of matter (water, fat, white matter, gray matter, necrotic tissue, etc.) in the brain so that each contrast image provides different types of information. Some of the more common types of contrast are T1-, T2-, and Fluid-attenuated inversion recovery (FLAIR)-weighted images. For example, T1-weighted contrast highlights fat tissue giving it a higher intensity value so that it ap-

pears white in the image, while giving a low intensity value to water making it appear dark in the image. This way MR images with different contrasts can highlight different matter depending on the localization of the tumor to achieve contrast in the areas around the tumor.

There are several challenges associated with the traditional diagnosis pipeline, such as a lack of general experts at the radiology unit or at a specific time of year or the complexity of pre-processing images. An interesting question is the ability to have an automated analyzing tool that can be used for precise diagnosis providing initial assessments of tumor aggressiveness. As MRI is a common approach for diagnosticating, the automation of diagnosing these kinds of images would be an exciting approach. Deep learning methods such as convolutional neural networks (CNN), a subclass of artificial intelligence, have proven to be quite efficient at recognizing patterns in images when there is a large amount of data available. An automated analyzing tool will benefit from identifying patterns and correctly classifying a tumor in a fraction of the time it would take a skilled professional to do the same.

The goal of this thesis is to develop a deep learning model that will be able to segment brain tumors in magnetic resonance images and use the segmented images to further classify the aggressiveness of the tumors as either lower-grade glioma (LGG) or higher-grade glioma (HGG) automating this task.

Chapter 2

Background

This chapter provides essential background information as theories, concepts, and terms used throughout this thesis. Here, the general concept of artificial neural networks will be explained, and a more specific type known as convolutional neural networks and medical imaging will be disclosed.

2.1 Artificial Neural Networks

Artificial neural networks (ANN) name and structure are based on understanding the human brain and its basic building block, the neuron. The brain comprises tens of billions of neurons where each neuron is connected to multiple other neurons through structural connections known as synapses [26]. Information flows between neurons through these links, where information moves to interconnected neurons in a hierarchical or layer-wise way.

ANNs mimic the human brains structure in the way that they have layers that comprise of nodes, or neurons, where each node has a numerical value, called a weight, associated with it. Each node is connected to other nodes where information from nodes in previous layers are handed down to nodes in deeper layers in a sequential fashion. A artificial neural network consists of at least 3 layers, an input-, hidden- and output-layer. An example can be seen in **Figure 2.1**. ANNs can be thought of as function estimators, and given a sufficient number of nodes and layers, one can compute any computational function.

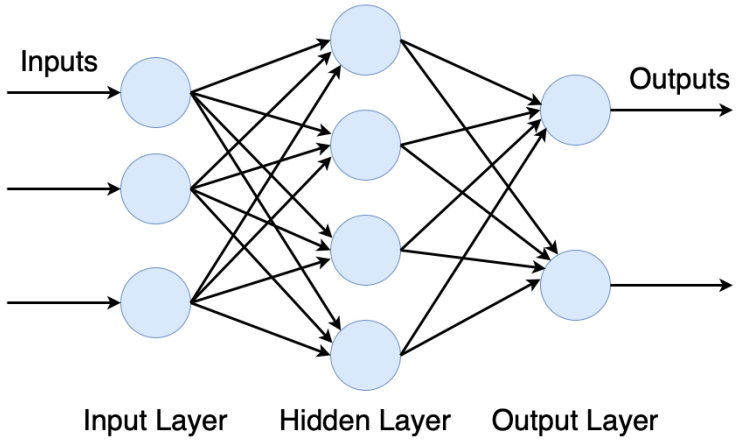


Figure 2.1: Visual representation of an artificial neural network. Number of hidden layers and neurons in each layer varies.

Input values given are assigned to nodes in the input layer. When a node passes its value to another node, the value gets multiplied with the corresponding weight in the current node before it is passed down to the ensuing node. Thus this becomes a weighted input in the receiving node. A single node can receive input values from multiple nodes. The final value is calculated as a sum of all input values(the sum also includes a bias term which can be thought of as the constant intercept in a linear equation). This weighted sum is often passed through an activation function, which

will be mentioned later. Then, the final value is multiplied by the current node's weight and passed down to the following node. This process of passing down values continues until the output layer has been reached and the final output is given.

The part of giving input to a network, processing it, and passing it to successive layers, is called forward propagation. This works well when the weights associated with each neuron are calibrated correctly according to the task at hand. To correctly configure the weights of each neuron, backpropagation is used. This is known as the process where an artificial neural network is learning.

Backpropagation is a technique used during supervised learning. Supervised learning can be a case where we want to build an ANN that can classify images containing cats and dogs. To do this, it requires a dataset of cats and dogs where each image is labeled with a category. These labels are the supervising part of the learning. An image is shown to the ANN during learning, where it calculates a score for each category. When the ANN makes an error, backpropagation is used to send information back into the network. Each neuron has its weight value updated proportionally to how much of the error in the output that they contributed. The larger the error a neuron's weight contributed, the large the change to its updated value.

2.1.1 Convolutional Neural Network

Convolutional neural networks [29], also known as ConvNets or CNN, is a specific type of ANN. CNNs are designed to use 2D arrays, usually in the form of images, as input. It can be expanded to work with 3D arrays/images in a straightforward scaling manner as well. The idea behind CNNs is to extract patterns and features using convolutions and pooling. This is achieved by having convolutional layers extract features such as corners and horizontal and vertical edges that together form a feature map. CNNs exploit the fact that many high-level features are composed of multiple lower-level ones, and by having multiple convolutional layers in succession, CNNs can capture higher-level features from lower-level feature maps. This way, edges and corners can be assembled into eyes, nose, and mouth, which again can be assembled into faces.

As mentioned, convolutional layers are used to extract features. Pooling layers are interspersed between convolutional layers, and it is used to aggregate features and downsampling. As pooling performs downsampling, the feature map becomes smaller in size, and thus, fewer parameters are required in the network.

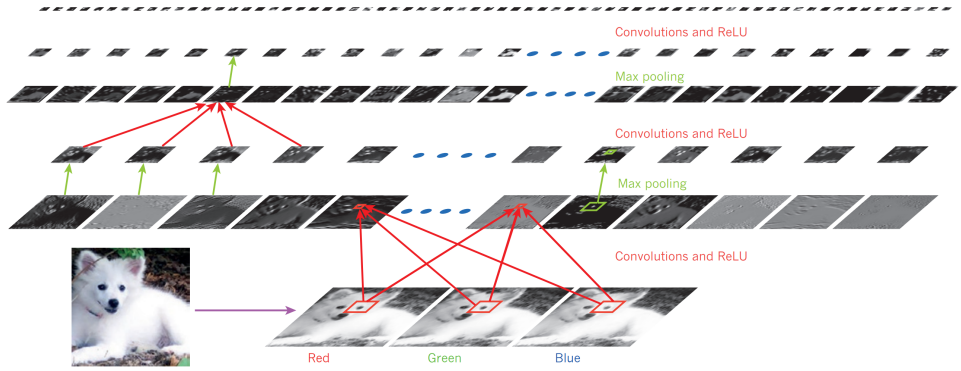


Figure 2.2: Example of a Convolutional neural network. Inside a convolutional network. The outputs (not the filters) of each layer (horizontally) of a typical convolutional network architecture applied to the image of a Samoyed dog (bottom left; and RGB (red, green, blue) inputs, bottom right). Each rectangular image is a feature map corresponding to the output for one of the learned features, detected at each image position. Information flows bottom up, with lower-level features acting as oriented edge detectors, and a score is computed for each image class in output. ReLU, rectified linear unit. The illustration is from Nature publication [29] with permission.

Depending on the task of a CNN the output varies. There are mainly two types of CNNs, semantic segmentation and classification. Segmentation does object detection and recognition where it outputs a new image where the relevant area is highlighted. This can be tasked with facial recognition or detecting cars. In a classification task, features extracted using convolutions and pooling are flattened. Flattening converts the 2D arrays in the feature map into a 1D array. Once the feature map has been flattened the CNN is followed by a traditional ANN to obtain the desired output.

The reason for using a CNN for these types of tasks instead of just a regular ANN is that ANNs are invariant to location. This is important as the localization of a feature in an image is relevant as location matters in imaging.

CNNs are trained the same way as ANN, but as in the case of ANN the weights trained are in each node. In a CNN, the trained weights are the values in the convolutional filters.

Convolution

Convolution is a mathematical operation that can be expressed as in **Equation 2.1** where w and h are the width and height of the filter, respectively. $\tilde{h}$ and $\tilde{w}$ are half the height and width of the filter, respectively. G is the filter itself where I is the image the convolution is performed on.

$$I'(x,y) = I(x,y) \circledast G(x,y) = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I(x+\tilde{w}-i, y+\tilde{h}-j)G(i,j) \quad (2.1)$$

Even though convolution is expressed as a mathematical operation, it can also be performed and explained more practically: As a 2D array consisting of $x \times y$ pixels can describe an image, the convolutional filter can be expressed as a 3×3 2D array. Other filter sizes can be used, but this is the most common size. To convolve the image with a filter, first, flip the filter about both its axis, then slide the filter along with the image from left to right. When the end has been reached, move down one row and repeat. Each time the filter moves to a new position, filter values are multiplied by the image pixel values at which it is currently residing, and the sum of each multiplication is computed. The obtained summation value is placed in a new image at the coordinate value where the filters' center value is currently at [6].

Pooling

There are two major types of pooling, max and average pooling. The usage of either of these types of pooling greatly depend on the type of data and the image analysis objectives. The average pooling smooths out the image, and hence the edges and sharp features may not be preserved. Max pooling filters out the higher intensity pixels from the image, and may perform better when the image background is darker, and only the brighter pixels of the image are of interest [5].

Pooling is also a type of convolution where it differs in how the result is computed within the filter. Using a max pool filter of size 3×3, the highest value of the 9 under scrutiny will be set as the output value. Average pooling calculates the average of the 9 values in a 3×3 matrix.

2.1.2 Activation Functions

A neuron outputs the sum of said neuron's inputs multiplied by its weight. This new value can further be used as input for another neuron. The issue is that the output will always have a linear value. Many problems have non-linear patterns, and a non-linear function is required to learn these patterns. This is achieved using an activation function. Activation function is often a non-linear function that takes the output of a neuron as input and processes it before another neuron receives it as an input.

Weights in neurons are not bounded, meaning they can take values between $-\infty$ and ∞. This can lead to weights exploding in size proportionally to weights in other neurons. Activation functions can take care of this by squeezing output values into smaller ranges, often between -1 and 1.

For backpropagation to work, a differentiable function is needed. Thereby, an activation function should be chosen with this in mind. There are many types of activation functions that have different pros and cons and different uses. Mentioned below are activation functions used in this thesis.

Rectified Linear Unit (ReLU)

Rectified linear unit [16, 21] (ReLU) is a commonly used activation function in the hidden layers. It is a non-linear function, and it is popular due to its simplicity and addresses the vanishing gradient problem. It is also an efficient activation function due to its simple derivative leading to easy calculations during backpropagation.

ReLU:

$$f(x) = \begin{cases} 0, & for\ x < 0 \\ x, & for\ x \geq 0 \end{cases} \tag{2.2}$$

ReLU derivative:

$$f'(x) = \begin{cases} 0, & for\ x < 0 \\ 1, & for\ x \geq 0 \end{cases} \tag{2.3}$$

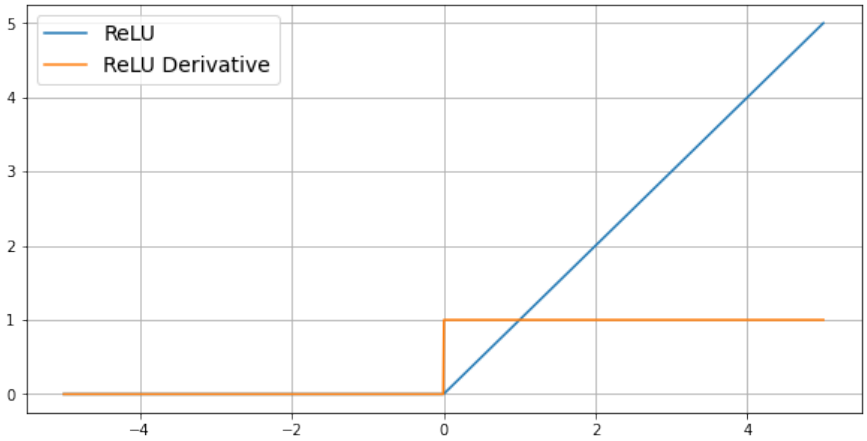


Figure 2.3: Plot of Rectified Linear Unit (ReLU) activation function.

Softmax

Softmax [21] normalizes its input value into a probability distribution consisting of N probabilities that summarizes to 1.

This function is commonly used in the final output layer of neural networks. It is used to convert output values into probabilities where each class is given a probability of how likely it is that the input belongs to this class. It ensures that the sum of the class probabilities will be equal to one, and it is often used in the case of multi-class problems.

Softmax:

$$f_i(x) = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}},\ n = 1, 2, ..., N \tag{2.4}$$

Sigmoid

Sigmoid [21], also known as the logistic function, is used to convert a linear function into probabilities between 0 and 1, as can be seen in **Figure 2.4**. This function is commonly used in the final output layer to convert the output to probabilities, similar to the softmax function. It differs from softmax in that it is used in the case of binary classification, while softmax is used in the case of more than 2 classes.

Sigmoid is one of the first activation functions that was used in every layer, but as newer functions such as ReLU came along, it has been replaced, particularly in hidden layers. Sigmoid´s strength is also its weakness in that it maps any input value into the range between 0 and 1. When the input value goes to either $-\infty$ or ∞, we can tell from **Equation 2.5** that the value will be close to 0 or 1, respectively. This leaves Sigmoid vulnerable to the vanishing gradient problem, especially if used in hidden layers.

Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Sigmoid derivative:

$$f'(x) = f(x)(1 - g(x)) \tag{2.6}$$

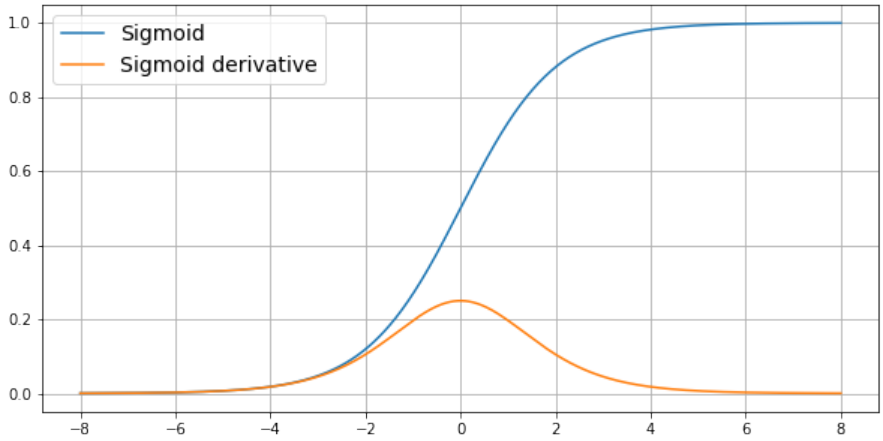


Figure 2.4: Plot of Sigmoid activation function.

Vanishing Gradient Problem

Vanishing gradient is a problem encountered when performing backpropagation in neural networks. This problem occurs when a large input is squished into a small output space, as in the case with the sigmoid function. As can be seen in **Figure 2.4** a large change in input value in the sigmoid function leads to a slight change in the output; thus, the derivative becomes small.

During backpropagation, the gradient is used to update the weights in neurons. When the derivative value is small, the change to the neuron weight will also be small. This leads to ineffective neural network training as the small changes cause slow training.

As mentioned, ReLU activation function works against the vanishing gradient problem. As we can tell from **Figure 2.3** the derivative of ReLU takes on the values of either 0 or 1, so that the derivative can't take on small values, hence ReLU becomes resistant to vanishing gradients.

2.1.3 Loss Functions

During the training of neural networks, a measure of fit, or how good it is performing, is required. The standard for doing this is known as a loss function. A loss function quantifies the deviation/error from the predicted value to the true value[26]. The goal of a loss function is to minimize the error, and different types of loss functions achieve this in different ways.

Mean squared errors (MSE) are the first loss functions used and are known from regression. It calculates the error by taking the distance from the predicted value, $\hat{Y}$, to the true value, Y, squaring it to remove negative values as it will cancel out positive values during summation. Once the distance has been calculated for every data point, the sum is calculated and divided to obtain the average value, hence the function's name.

Mean Squared Error (MSE):

$$f(x) = \frac{1}{N}\sum_{n=1}^{N}(Y - \hat{Y})^2,\ n = 1, 2, ..., N \tag{2.7}$$

Mentioned below are the loss functions used in this thesis.

Dice

Dice loss is a measure of similarity, and it originates from the Sørensen-Dice Coefficient[23]. It has later been adapted for use in neural networks in segmentation problems. In **Equation 2.8** the formula can be seen where A and B are images where one is the predicted segmented images, and the other is the true annotated image we want neural network to achieve. It is calculated to be 2 times the area of the intersection of A and B, divided

by the sum of A and B. It is multiplied by 2 so that when there is a total overlap both the intersection and sum will be equal to one.

$$Dice(A, B) = 2 * \frac{|A \cap B|}{|A| + |B|} \tag{2.8}$$

Dice loss minimizes the error based on the overlap of two images. Its value ranges from 0 to 1; 0 indicates no overlap, and 1 indicates total overlap, as A and B are the same image. Obtaining a loss function where the optimal value is 0 error, can be achieved by using Loss $= 1 - Dice$. A visual representation of Dice can be seen in **Figure 2.5**.

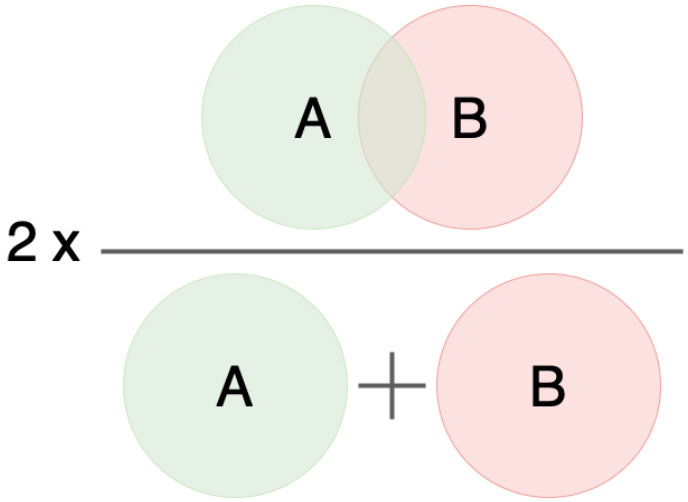


Figure 2.5: Visual representation of dice loss according to **Equation 2.8**. Dice is a measure of overlap where a value of 1 indicates full overlap and 0 indicates there is no overlap.

Binary Cross-Entropy (BCE)

Binary cross-entropy, also known as log loss, is a loss function used for classification problems where only 2 categories are present. It is based on entropy which is a measure of uncertainty, and in information theory, it is stated that information obtained in an inferential process is determined by the reduction of entropy[19]. In **Equation 2.9**, Y_i, is the class label where the labels are either 0 or 1. From the equation, we can tell that depending on which class label we are using, one of the terms will cancel out so that we only calculate the entropy for the given class.

$P(Y_i)$ is the predicted probability that the current point belongs to class Y_i. When the entropy is calculated for each datapoint, they are summarized to calculate the average. The goal of BCE is to achieve a value as close to 0 as possible. Looking at the entropy in BCE, $Y_i \ln P(Y_i)$, where $P(Y_i)$ can have a value between 0 and 1, we can tell that the entropy will be close to 0 when $P(Y_i)$ is close to either 0 or 1, and largest when $P(Y_i)$ is 0.5. In other words, we want to decrease the entropy as this will decrease the uncertainty and the BCE value.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N}\Big[Y_i \ln(P(Y_i)) + (1 - Y_i)\ln(1 - P(Y_i))\Big] \tag{2.9}$$

A multiclass function for cross-entropy exists called categorical cross-entropy. This is a generalization of cross-entropy, and if used in a 2-class-problem, the same result as binary cross-entropy is attained.

2.1.4 Optimizers

As mentioned in **Section 2.1.3**, the goal of a loss function is to minimize the error. When the function is convex, this is straightforward as the optimum can be found by calculating the derivative and solving for 0, i.e., $f'(x) = 0$. This works for convex functions as every local minimum is a global minimum. However, this is not true for non-convex functions where stationary points can be either a saddle point or a local or global mini-

mum/maximum [26]. This is often the case with neural networks; thus, optimizers are introduced to help the loss function with reduction.

There are many different optimizers with different pros and cons, but gradient descent is commonly used. Its formula can be seen in **Equation 2.10**.

$$\theta_{Next} = \theta_{Prev} - \alpha \times \nabla J(\theta) \tag{2.10}$$

Each time a new value is pushed through the neural network and backpropagation is initiated, a lower value for the loss function will be attempted to be obtained. The current point is kept until the next time backpropagation is performed, and this is what θ_{Next} represents. When backpropagation is performed, weights are updated according to the formula in **Equation 2.10**. $\nabla J(\theta)$ is the gradient of the activation function, mentioned above in **Section 2.1.2**. The gradient will tell the network which way the steepest slope is, but what we are interested in is the steepest descent, which will be the opposite direction, thus the formula subtracts the gradient from the position, θ_{Prev}, from the previous round it was updated and stores this is the new value for the next time backpropagation is performed. α is known as the learning rate, this value is control how fast the gradient moves away from the previous position towards the new position. Learning rate will be be mentioned in **Section 2.1.6**.

2.1.5 Metrics

Metrics are used to measure how well a neural network is doing during training and validation. Metrics provide similar functionality to loss functions. The main difference is whether neuron weights are updated according to the metric or not. This means that loss functions will be minimized by the optimizer while a metric is only an interface used to judge the performance of the network. Also, as mentioned earlier, for a function be used as a loss function it needs to be differentiable.

Intersection Over Union (IoU)

Intersection over union is also known as the Jaccard Index, which evaluates the overlap of two objects in the image. It is a similarity measure [28] and it ranges from 0 to 1, where no overlap between the two images equals 0, and total overlap equals 1. It is a common metric to use in semantic image segmentation.

In **Equation 2.11** A and B are two sets. A is the true image set and B is the predicted image set.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.11}$$

Accuracy

Accuracy calculates how many of the predictions are equal to their true labels. It uses total labels and a counter for how many of the predictions are correct and calculates the fraction, as seen in **Equation 2.12** to find the total accuracy. It can also be calculated in terms of statistical error, as seen in **Equation 2.13**. Accuracy is often used to evaluate classifiers ability to generalize [15].

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2.12}$$

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.13}$$

Accuracy has limitations. If we consider a 2-class problem where class 0 consist of 990 datapoints and class 1 cosnsist of 10 datapoints. If a model predicts everything to be class 0, this will result in an accuracy of $\frac{990}{1000} = 0.99$. This is misleading as the model doesn't detect any datapoints to be class 1.

Precision and Recall

Precision and recall is also known as specificity and sensitivity, respectively. Precision is the ratio of true positives of total positives predicted, as seen in **Equation 2.14**, while recall is the ratio of true positives out of all positives in ground truth, as seen in **Equation 2.15**.

$$Precision = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}} \tag{2.14}$$

$$Recall = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}} \tag{2.15}$$

Precision and recall both ranges between 0 and 1 where a precision value towards 1 signifies that the network correctly classified mostly true positives. What it doesn't show is the amount of false negatives. A recall value towards 1 signifies the network missed few true positives, a lower recall indicates missed positives predictions.

The desired outcome is to have a high precision and recall, but there is usually a trade-off between the two. As one increases, the other decreases, this is known as the precision-recall trade-off [8].

Precision and recall are both closely related to the confusion matrix talked about below in **Section 2.1.7**.

2.1.6 Hyperparameters

Hyperparameters [30] are variables that affect the performance of the model during training. While training, they remain constant and can only be changed before or after. Changing these parameters to find the optimal values for best performance is called tuning. Hyperparameters used to influence the training models in this thesis are mentioned below.

Batch Size

Batch size is the number of data points pushed through the neural network before backpropagation is performed. Larger batch sizes speed up the time it takes to train a model at the cost of a lower ability to generalize. However, changing the learning rate closes this generalization gap [13].

Larger batch sizes put higher requirements on hardware as more data points need to be available in memory simultaneously. Batch size is often chosen to be a multiple of the system's available memory.

Epochs

An epoch is the number of times a model is trained on the entire dataset. One epoch means that each datapoint will be able to affect the training of weights one time. When the number of epochs increases, each data point gets an additional chance to influence the training; increasing epoch will increase the time to train and help the model with convergence as long as the model is still learning. Increasing epoch also increases the chance of the model overfitting as the model can potentially learn to fit the training data perfectly.

Learning Rate

Learning rate is a hyperparameter within the optimizer. It is used to control the step size between each iteration during backpropagation. If the learning rate is low, the learning curves are typically smooth, but convergence can be slow. In contrast, a significant learning rate can oscillate but lead to faster convergence unless the learning rate is too large and the model will diverge [26]. The effect on the learning rate can often be seen on the loss function value, and an illustration can be seen in **Figure 2.6**.

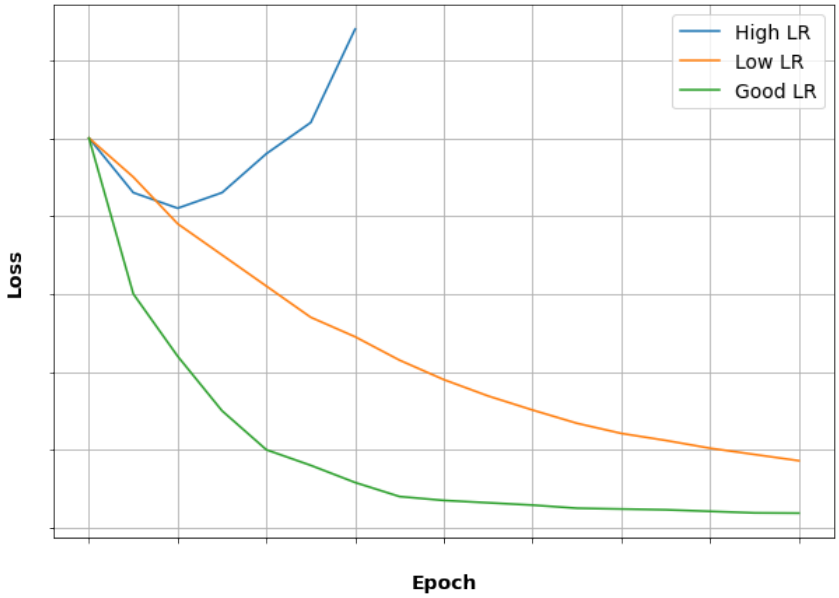


Figure 2.6: Illustrating the effect of learning rate (LR) on the loss. Blue line illustrates the effect of a high learning rate where divergence occurs. Orange shows the effect of a low learning rate which results in slow convergence. Green line represent a learning rate where we have a smooth line where convergence is achieved.

2.1.7 Confusion Matrix

Confusion matrix can be scaled to be used with any number of classes, but the binary confusion matrix is the easiest one and is the one used in this thesis; thus, the binary confusion matrix will be described here.

Confusion matrix is used to evaluate how well a model is performing. For the binary case, there are two classes, 0 and 1, also known as as false and positive classes, respectively. The confusion matrix consists of 4 values:

- **True Positive**: Number of correctly classified positive that belong to the positive class.
- **True Negative**: Number of correctly classified negative that belong to the negative class.
- **False Positive**: Number of wrongfully classified negatives that have been classified as positive.
- **False Negative**: Number of wrongfully classified positives that have been classified as negative.

We want as many positive classes to be predicted as positive cause this will increase the value of true positive. The same goes for true negatives, where we want negative classes to be predicted as negatives. A confusion matrix is often visualized using a plot where it is represented, as can be seen in **Figure 2.7**.

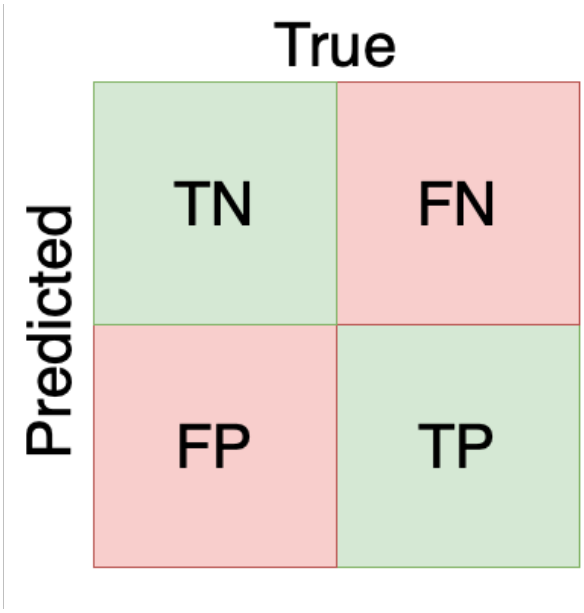


Figure 2.7: Illustration of a confusion matrix. Rows represent predicted values while columns represent true labels. Diagonal shows correctly predicted, while the anti-diagonal shows the wrongly predicted.

It is possible that other metrics mentioned as IoU and accuracy can have values close to 1, making it seem like the model has a high performance, but quite the opposite could be the case. Confusion matrix allows further evaluation so that issues such as bias in the data can be spotted, which accuracy and IoU may not tell us.

2.2 Magnetic Resonance Imaging (MRI)

Magnetic resonance imaging is a non-invasive and non-ionizing technique allowing detailed images of the brain. The technique provides the acquisition of several contracts. MRI has been established as a critical diagnostic tool for brain examinations, including initial characterization of brain tumors, post-treatment monitoring, and surgery planning. It is common to acquire complimentary MRI modalities such as T1-weighted, T1-weighted with contrast enhancement (T1ce), T2-weighted, and Fluid Attenuation Inversion Recovery (FLAIR). These protocols enable distinctive contrast from the brain tissues, providing the margin of the tumor and tumor spread throughout normal tissues. For example, the T1ce modality indicates hyperactive tumor sub-regions, which may induce necrosis.

2.2.1 MRI contrasts

MRI consists of a static magnet to produce a strong magnetic field that forces protons in the body to align with the direction of the magnetic field. A radiofrequency is then pulsed at often a 90° angle [3] of the magnetic field to stimulate the protons. The 90° pulse moves the protons perpendicular to the magnetic field. When the radiofrequency is turned off, the protons will align with the magnetic field again (equilibrium or steady-state), this is known as relaxation. As this happens, energy will be released; the MRI sensors (radiofrequency antea, MRI coils) detect that. The time it takes a proton to realign and the amount of energy released change depending on the type of tissue to which the proton belongs.

Depending on the TE (Time to Echo) and TR (Repetition Time) timings, different tissue types will release varying amounts of energy. Thus, MRI systems can generate different kinds of contrasts. The effect of different MR contrasts will be attempted explained below in a simplified manner.

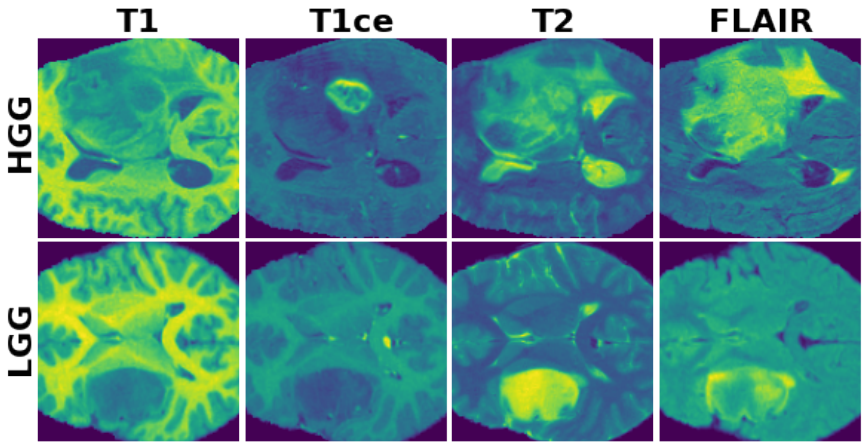


Figure 2.8: Example images of four modalities: T1-weighted (T1), T1-weighted contrast-enhanced (T1ce), T2-weighted (T2), and Fluid Attenuated Inversion Recovery (FLAIR) imaging acquired from lower-grade glioma (LGG) and higher-grade glioma (HGG) patients in the BraTS dataset.

T1-weighted

T1-weighting is achieved having short TR and TE timings. This causes tissues high in fat to have high intensity values, appearing bright in the image. Areas containing CSF (Cerebrospinal Fluid), which is high in water, has higher T1 values and thus appears darker in images.

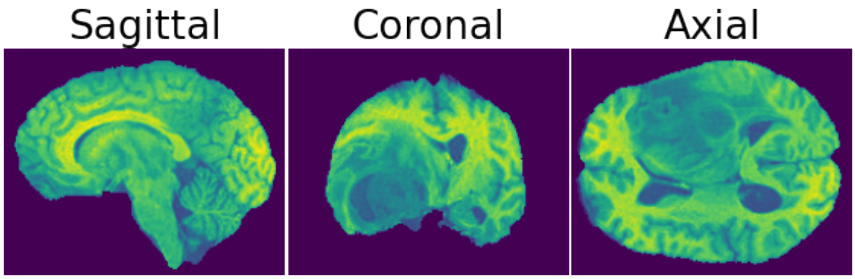


Figure 2.9: T1-weighted sagittal, coronal and axial MR images. Many areas appear bright as there is a lot of brain tissue contains fat.

T1 Contast-Enhanced

T1-contrast-enhanced-weighting is achieved having short TR and TE while also using a contrast-enhancing agent. The agent is given intravenously to the patient which increases the speed at which realignment occurs. Gadolinium is the most used agent [11] and it induces relaxation in tissue where it accumulates, increasing the intensity in the area.

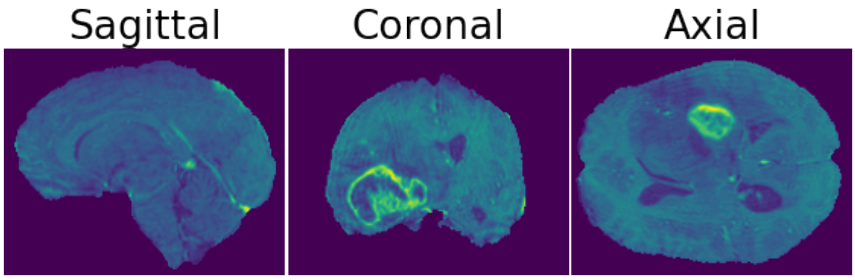


Figure 2.10: T1-Contrast-Enhanced sagittal, coronal and axial MR images. Enhancing agent speeds up realignment causing higher intensity values in regions where the agent accumulates.

T2-weighted

T2-weighting is achieved having long TR and TE timings. In simple terms, T2-weighted shows the opposite of what a T1-weighted image does. T2 causes tissues high in fat to appear dark in images, while areas high in water will appear bright. As can be seen in **Figure 2.11**, some areas appear a lot brighter, especially in the lower left corner of the coronal image. This is the tumor area where edema (swelling) causes excess fluid to be trapped resulting in a bright region.

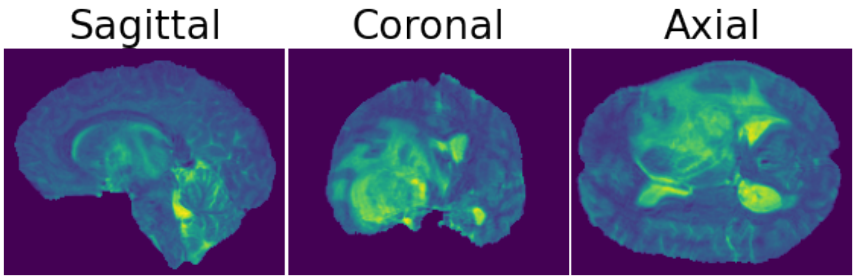


Figure 2.11: T2-weighted sagittal, coronal and axial MR images. Water has a short T2 relaxation time and thus fluids appear bright in this modality.

FLAIR

Fluid-attenuated inversion recovery (FLAIR) is an advanced MRI modality. The technique elucidates sub-regions of brain tissue T2 prolongation as bright while suppressing (grayscaled darkening the areas) CSF, resulting in a more apparent contrast of lesions in proximity to CSF, such as edema [10]. FLAIR has improved the MRI diagnosis of various brain lesions and disorders and is implemented as a complementary protocol to conventional MRI sequences.

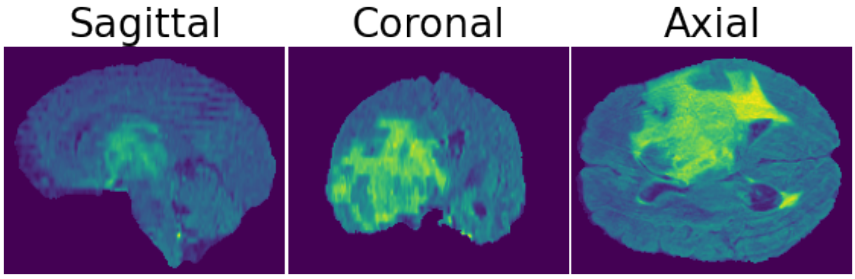


Figure 2.12: FLAIR-weighted sagittal, coronal and axial MR images. Similar to T2-weighted with the difference being in the TR and TE timings being longer causing abnormalities to remain bright while CSF is made dark [14].

Chapter 3

Methods

This section will include analysis of the dataset used in this thesis and the machine learning models used.
The model used is split into a segmentation part and a classification part. First, the segmentation part will be presented, and the output from this model will be used as input for the classification part of the model.

3.1 Environment

Environment for coding was setup using anaconda, and the work was done in Jupyter Notebook. Code was implemented using Python version 3.8.5, and the libraries used were NiBabel=3.2.1, Numpy=1.19.5, Pandas=1.4.0, Tensorflow=2.6.2, Keras=2.6.0, Matplotlib=3.5.1 and Segmentation Models=1.0.1.

3.2 BraTS 2020 Dataset

BraTS (Brain Tumor Segmentation) 2020 dataset [1] consists of 369 multi-contrast MR scan images from glioma patients, where 293 of the patients

have been diagnosed with HGG and 76 have been diagnosed with LGG. The BraTS MR data collection includes TCGA-GBM and TCGA-LGG MR imaging data, available through The Cancer Imaging Archive (TCIA) repositories [2, 9]. All MR images are multimodal and consist of the same four MRI contrasts: T1-weighted, T2-weighted, Fluid Attenuated Inversion Recovery (FLAIR), and T1-weighted Contrast Enhanced (T1-CE). Images in the training data come with a ground truth segmented image along with the 4 contrast images. Each ground truth image has been manually segmented by up to four raters to locate tumor areas where their annotations have been approved by neuroradiologists [1, 7, 25, 24].
Out of the 369 patients, 235 are deceased where all of them were diagnosed with higher-grade glioma. Metrics for these individuals can be seen in **Table 3.1**.

	Mean	Std	Median
Age (Years)	61.94	11.90	61.94
Survival Days	461.29	346.04	461.29

Table 3.1: Metrics for the 235 deceased HGG diagnosed patients from the BraTS dataset.

BraTS MR images are stored in NIfTI (.nii or .nii.gz) file format containing a sequence of images making up a 3D model of the brain. In the 3D brain model each axis can be accessed two at a time to achieve a 2D representation of the brain, this is known as a slice. Each image has been pre-processed in the sense that every image has been standardized to a size of $240 \times 240 \times 155$ pixels. Each image has been stripped of eyes and skull to leave only tissues considered as part of the brain to be left in the images as can be seen in **Figure 3.1**.

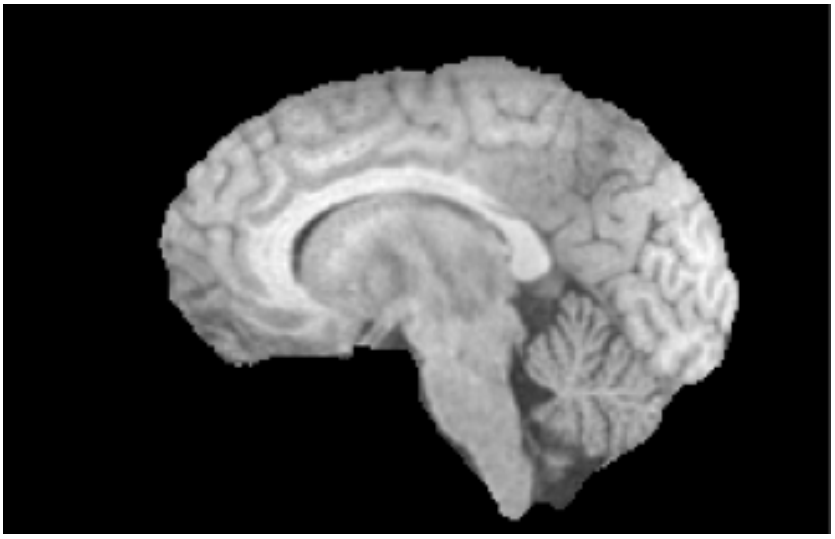


Figure 3.1: MR images showing a brain in the sagittal plane that has been stripped of all non-brain tissue.

Annotated images consist of 4 labels, 0 of which are unlabeled/background, 1 is necrotic, 2 is peritumoral edema, and label 3 is a gadolinium-enhancing tumor.

3.3 Segmentation Architecture

For the segmentation task, a modified version of the U-Net [17] was used. U-net architecture consists of a contracting path, which can be considered as a typical architecture for a CNN where the size of the input image decreases at each layer it passes. Following the contracting path comes an expansive path. This part of the network works in the inverse way where the image size increases at each progressive layer.

The modified network was built in the same way as the original U-Net, with the main difference being in the input and the output size. The original U-Net has an input size of $572 \times 572 \times 1$ pixels and an output size of $388 \times 388 \times 2$ pixels whereas the model used here has an input size of $128 \times 128 \times 3$ pixels and an output size of $128 \times 128 \times 4$ pixels. Input size consists of 3 channels

as the model uses 3 contrast images stacked on top of each other, and the output size has 4 channels as each segmented image has 4 labels used as classes for the output. A figure illustrating the modified network can be seen in **Figure 3.2**.

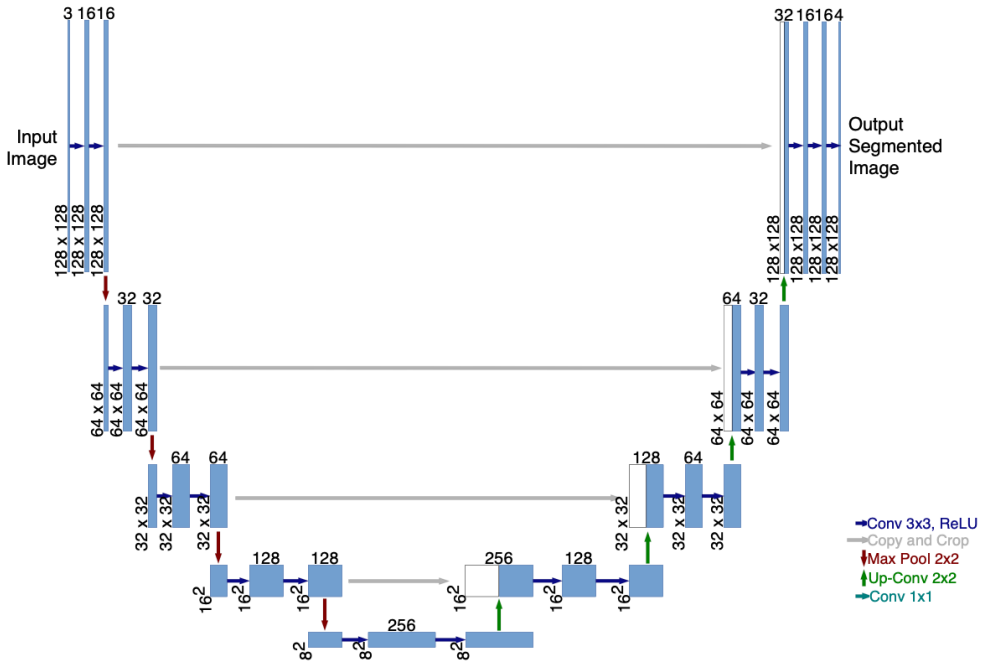


Figure 3.2: Illustration of the modified U-Net architecture for the segmentation task. Illustration is a modified version of [20] with permission.

The model used consists of 20 convolutional layers, where each convolution uses padding to ensure that the size stays the same. Dropout layers are used between each convolution, and in the contracting path, there is used a maxpool layer after every other convolution that decreases the image dimensions by half. At the same time, the number of filters is increased by the double.

In the expansive path, the dimensions are increased back up by doubling the image dimensions as well as decreasing the number of filters by half. As the

image dimensions increases in the expansive path, the image is concatenated with the image of the same dimensions from the contracting path to keep the localization of features in the image. The model architecture can been seen in **Table 3.2**.

Layer	Image Size	Filters
Input	128 × 128	3
Conv2D	128 × 128	16
Dropout	128 × 128	16
Conv2D	128 × 128	16
MaxPool	64 × 64	16
Conv2D	64 × 64	32
Dropout	64 × 64	32
Conv2D	64 × 64	32
MaxPool	32 × 32	32
Conv2D	32 × 32	64
Dropout	32 × 32	64
Conv2D	32 × 32	64
MaxPool	16 × 16	64
Conv2D	16 × 16	128
Dropout	16 × 16	128
Conv2D	16 × 16	128
MaxPool	8 × 8	128
Conv2D	8 × 8	256
Dropout	8 × 8	256
Conv2D	8 × 8	256
Conv2DTrans	16 × 16	128
Concat	16 × 16	256
Conv2D	16 × 16	128
Dropout	16 × 16	128
Conv2D	16 × 16	128
Conv2DTrans	32 × 32	64
Concat	32 × 32	128
Conv2D	32 × 32	64
Dropout	32 × 32	64
Conv2D	32 × 32	64
Conv2DTrans	64 × 64	32
Concat	64 × 64	64
Conv2D	64 × 64	32
Dropout	64 × 64	32
Conv2D	64 × 64	32
Conv2DTrans	128 × 128	16
Concat	128 × 128	32
Conv2D	128 × 128	16
Dropout	128 × 128	16
Conv2D	128 × 128	16
Output	128 × 128	4

Table 3.2: Modified U-Net archtitecture in sequential order.

3.3.1 Data Pre-Processing

As the images in the BraTS dataset consists of 3D images, each image needs to be pre-processed to fit the 2D U-Net model used. Each image can be split into the 3 axes sagittal, coronal, and axial, as can be seen in **Figure 3.3**.

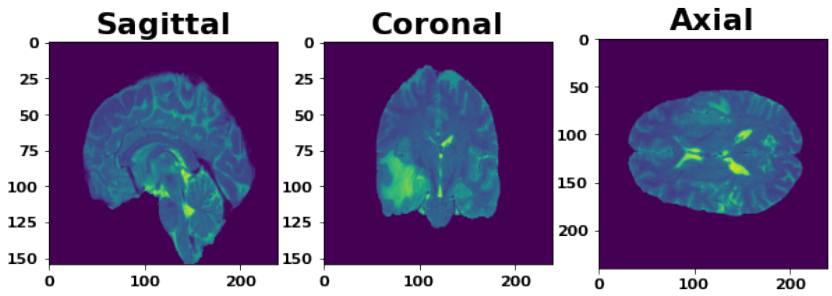


Figure 3.3: T2 contrast image showing the same unprocessed brain image from the different axes sagittal, coronal, and axial. Sagittal and coronal has a dimension of 155×240, while axial has a dimension of 240×240. Aspect ratio has been increased for sagittal and coronal to fit figure.

the following pre-processing methods has been used on each image:

Slicing

Each image has been sliced to go from a 3D image to a 2D image. Here the axial axis has been used as this axis contains much information and more commonly are investigated by expert. Each slice has been collected from the center of the 3D image as the area is more prominent.

Location of the tumor can affect how visible the tumor is in each slice and the tumor can easier to distinguish in different MRI contrasts. Therefore, the MRI contrasts T1 contrast-enhanced, T2-weighted and FLAIR has all been used for the same slice where they have been stacked on top of each other to create an image with 3 channels to be used as input for the network. This way the model will have 3 contrasts that can be used to identify the tumor area.

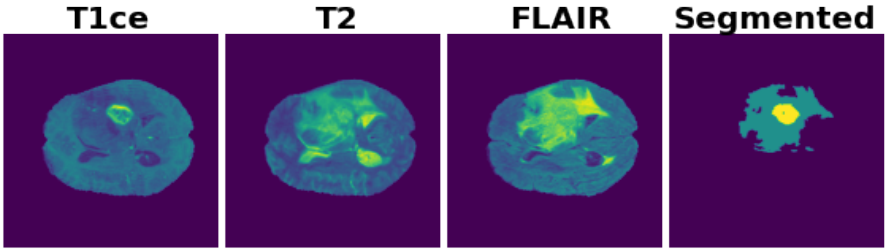


Figure 3.4: Illustration shows the tumor's appearance in the three contrast types T1 contrast-enhanced, T2 weighted, and FLAIR. The image to the far right shows the ground truth of the tumor.

Image Cropping

As can be seen in **Figure 3.4** above, each image consist of a considerable amount of background. To make the model's training faster and easier, and to avoid confusing the model, each image has been cropped in size to remove as much of the background region as possible while keeping the region of interest.

Segmented image is also cropped to keep the localization equal to that of the images.

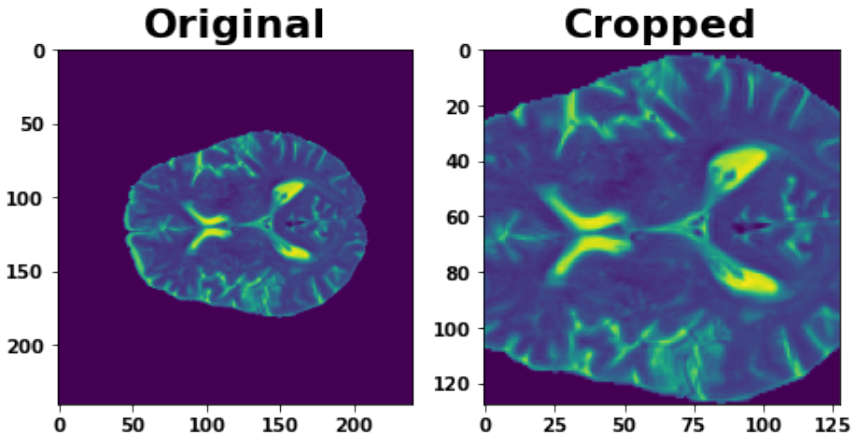


Figure 3.5: Comparison of the original image and after the cropping.

Scaling

Data has been scaled to be in a range between 0 and 1 as a difference in pixel range between images can affect the training of the model.

Label Normalization

Segmented images consist of 4 labels with 0, 1, 2, and 4. None of the segmented images has any pixel with a value of 3. Thus, 4 is changed to be 3 instead to get the 4 labels of values 0, 1, 2, and 3. Further, each segmented is turned into categorical images. This turns each segmented image into 4 binary-valued images where each image only has a pixel value of 0 or 1 depending on if the label is present or not. Each segmented image has 4 channels giving them a shape of $128 \times 128 \times 4$.

Data Augmentation Augmentation has been used to increase the number of data points and help with generalization and avoid overfitting. Augmentation has been done by flipping images both horizontally and/or vertically. Images are augmented at random using a uniform distribution and they can be either flipped horizontally, vertically, or both or not flipped at all. This way, each slice can potentially create 4 different data points.

Thresholding

The tumor's position affects which slices contain the most information about the tumor. This can cause some of the slices to be blank as the tumor is not present in a given slice which renders the image unusable as it will not contribute to the improvement of the model. These images has been avoided by using a threshold on every ground truth image, checking that at least 1% of the pixels in the image are of either label 1, 2 and/or 3. If this threshold is not met, the slice is discarded.

Data Split

From the 369 3D images, 11 slices were sampled per image and after each image was pre-processed, 3 421 slices were left in total for use. These slices were split into validation, training, and testing splits with a distribution of 10%, 70%, and 20%, respectively.

3.3.2 Validation and Training

The validation set consisted of 363 slices, approximately 10% of the data, and was used to tune hyperparameters before completing the training set. Toward an optimized training, some hyperparameters were tuned including epoch, batch size, learning rate, and class weights in the dice loss function. Validation was done multiple times on different subsets of the data to achieve K-fold cross-validation to eliminate noise from affecting the validation results.

Initial values for hyperparameters were chosen based on similar segmentation work using U-Net [27, 22]. Each hyperparameter was tuned to some extent through trial and error. The epoch was tuned the most by trying an increasing amount of epochs until the model started to decrease in performance during validation. A total of 150 epochs was attempted at max during validation, and it resulted in a decrease in performance where 100 epochs yielded the best metric values.

Model was trained on 2 380 slices, approximately 70%, and it was trained for 100 epochs using a batch size of 2, a learning rate of 0.0001 for the Adam optimizer, and class weights of 0.005 for the background label where the remaining 0.995 was split evenly between the other classes, these weights were chosen to accommodate the imbalance in the class distribution of the class labels in the segmented images in the dataset.

3.4 Classification Architecture

For the classification task, the model used was trial and error to find a model that could fit. Papers using convolutional neural networks to solve classification based on MR imaging were considered, which led to the model in this paper [4]. This paper attempts various models and compares them to find the optimal model. The task they are solving is the same classification of glioma tumor grade as in this thesis, only with a different dataset and a different amount of output parameters.

The model has an input size of $128 \times 128 \times 1$ pixels, as this is the output image size from the segmentation model, and a binary output of a single value as the model is to predict either 0 or 1 where 0 represents HGG and 1 represents LGG. The input to the model is an image containing a tumor extracted from an MRI image of a brain which can be obtained by running a brain MRI image through the segmentation model from the first part of this thesis in section 3.3. The output is 0 or 1 because the model's objective is to classify the tumor grade based on the image features. As the dataset has only two types of grades, this becomes a binary task. An illustration of the model architecture can be seen in **Figure 3.6**.

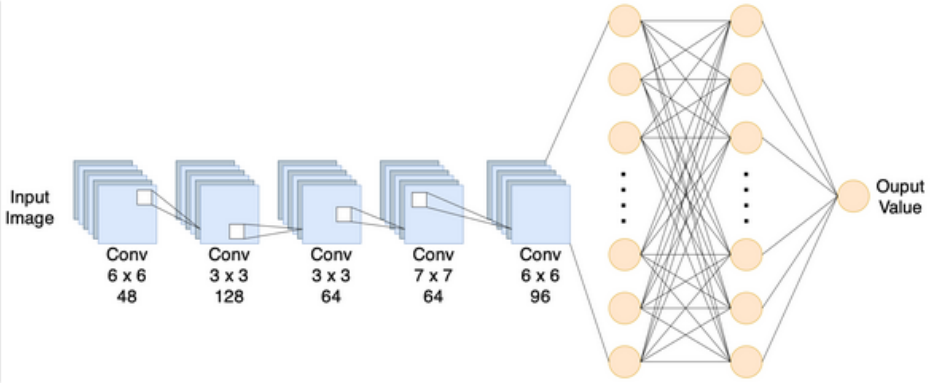


Figure 3.6: Illustration of the classification model architecture.

The model consists of 5 convolutional layers and 2 fully connected layers. Each convolution uses padding to ensure the image size is kept the same; each convolution is followed by a max-pooling layer that extracts features

from the previous layer. Weights in each of the convolutional layers is also initialized using He_normal initialization; this has been observed in the paper [4] where the model was introduced to help with speeding up convergence. Following the two fully connected dense layers, there is a dropout layer to help with overfitting and ensure a better generalization of the model. The model architecture can be seen in **Table 3.3**

Layer	Kernel Size	Filters/Units
Conv2D	5×5	48
MaxPool	2×2	-
Conv2D	2×2	128
MaxPool	2×2	-
Conv2D	3×3	64
MaxPool	2×2	-
Conv2D	7×7	128
MaxPool	2×2	-
Conv2D	6×6	96
MaxPool	2×2	-
Flatten	-	-
Dense	-	512
Dense	-	512
Dropout(0.2)	-	-
Output	-	1

Table 3.3: Classification model architecture in sequential order.

Even though the output from the segmentation model in section 3.3 is intended to be used as input for the classification model talked about in this section, a modification has been done to the data before its being used as input. The modification is mentioned in subsection 3.4.1 under intersection.

3.4.1 Data Pre-Processing

Data used as input for the model is based on the segmented output images from the segmentation model. As the segmentation model is trained on image slices from the axial axis, the same axis is used to sample data for

this model. Even though the output from the segmentation model is used as input for the classification model, the data has been pre-processed to some degree, and the pre-processing methods used are as follows:

Oversampling

Due to the imbalance of class distributions in the BraTS dataset 76 of the patients are classified as LGG, and 293 are classified as HGG. This is an approximate ratio of 1:5, and to avoid the model being biased towards the majority class, oversampling has been used on the LGG images to obtain a more even ratio closer to 1:1.

Oversampling has been done by sampling more slices from each LGG patient. There were 11 slices sampled from each HGG patient and 42 slices from each LGG patient. This was approximately four times as many from LGG as from HGG, which ensured a more even split. There are around four times as many HGG patients as LGG patients. This has led to a total dataset of 17428 image slices with a split of 7808 LGG slices and 9620 HGG slices.

Note that even though it seems to be a 50/50 split from the sampling method, the same thresholding technique used in sampling data for the segmentation model was used here. Thus, more of the LGG patient slices were discarded than the HGG slices, and the data split ended at roughly 55% HGG and 45% LGG.

Intersection

To obtain a higher amount of information for the model to learn from the intersection between the input image and output mask from the segmentation model was done. First, the image was converted into a binary image where any value greater than 0 was set to 1, and everything else was kept as 0. This way, a tumor silhouette was obtained. An example of this can be seen in **Figure 3.7**.

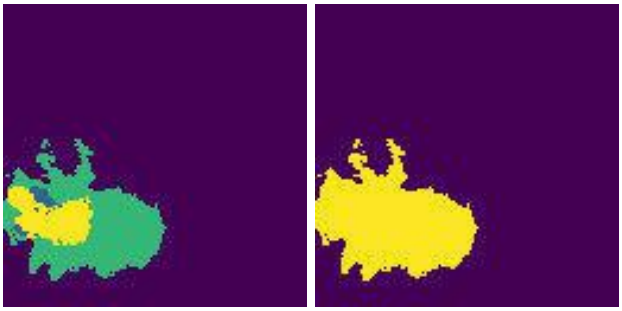


Figure 3.7: Left tumor segmentation is converted to binary silhouette in the right image using thresholding.

Further, this tumor silhouette mask was together with the input image used to find the intersection between the two. The result is an image containing only the values of the tumor where every value is present instead of having the mask with only four different values representing the labels. This way, more information will be available to the model. A figure showing the intersection result can be seen in **Figure 3.8**.

Figure 3.8: Intersection between left image and the mask in the middle is calculated to obtain the tumor image to the right.

Data Augmentation

Each image slice occurs in the four contrast types, T1-weighted, T1 contrast-enhanced, T2-weighted, and FLAIR. Each of these has been used to create intersection images, as mentioned above, to increase the amount of data available during validation, training, and testing.

Data was augmented on the fly during training to increase the training data's size and achieve better generalization. Augmentation methods used were the normalization of pixel values to a scale between 0 and 1, images were flipped, and shearing and zooming were used.

Data Split

From the 17428 image slices created during oversampling, the data was split into validation, training, and testing, where the distribution used was 20%, 70%, and 10%, respectively.

3.4.2 Validation and Training

Validation was done on 3485 image slices that were sampled at random from the entire dataset of 17428, and it was used to tune hyperparameters of the model prior to model training. Hyperparameters tuned were epochs, batch size, and learning rate.

Initial values were 100 epochs, batch size of 8, and learning rate of 0.01. The number of epochs was chosen as it was the selected number in the paper where the model was introduced. At the same time, the learning rate is the default value used with the Adam optimizer, and the batch size was chosen to be a small size to start as it is less intense on memory, and smaller batch sizes have shown to provide improved ability to generalize [13].

Learning rate had the most significant impact on the model result. During the validation, epoch values between 50 to 100 epochs, depending on other hyperparameter values, were used, particularly with changing learning rates. Learning rates used were chosen from [0.1, 0.01, 0.001, 0.0001], where any other value than 0.001 led to early convergence and an underfitted model where larger epoch values were attempted. However, the model would always converge after less than 10 epochs and get stuck. The batch size seemed to have little to no effect on the models' ability to learn and was increased from 8 to 64 to decrease the time to train.

Chapter 4

Results

4.1 Segmentation Model Validation and Training Results

The evaluation employed Dice loss, accuracy, and intersection over union to assess the performance of the segmentation model. These metrics were calculated for training, validation, and testing separately.

4.1.1 Validation

The segmentation model achieved a dice loss of 0.8421, an accuracy of 0.9496, and a mean IoU of 0.9356. As mentioned earlier, validation was performed on 10% of the dataset, which accounts for 363 slices. Validation was repeated multiple times on different samples to ensure an unbiased result due to the current validation sample and similar results found for different samples.

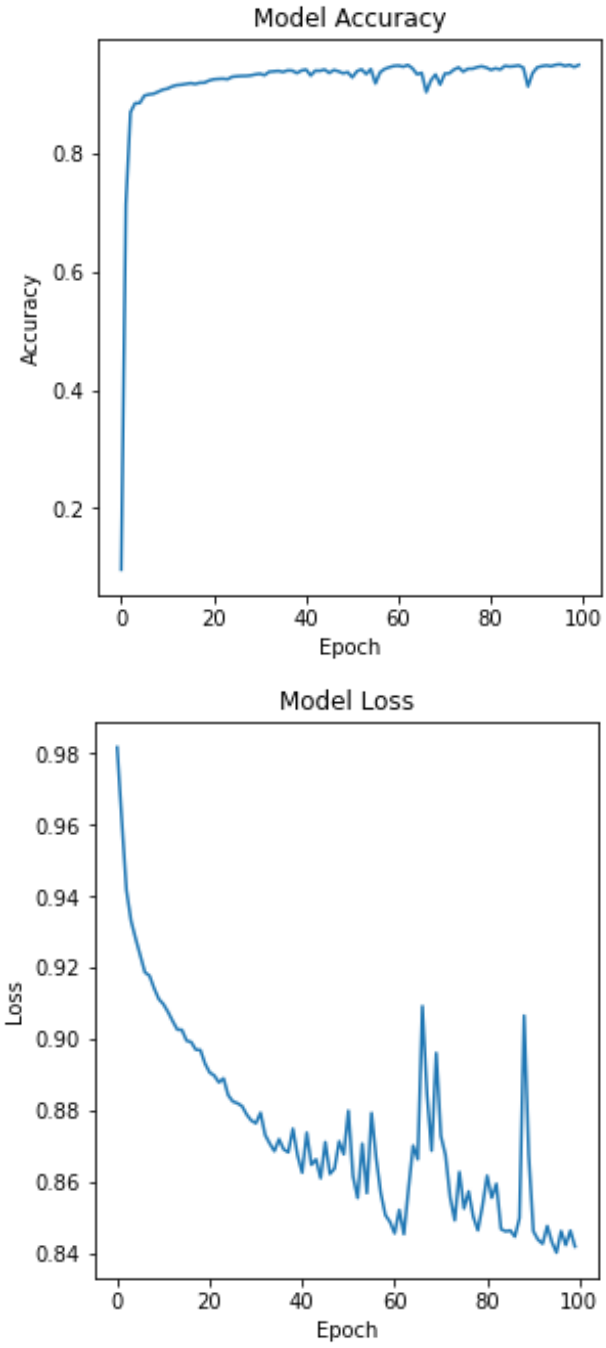


Figure 4.1: Accuracy and loss achieved during validation of segmentation model.

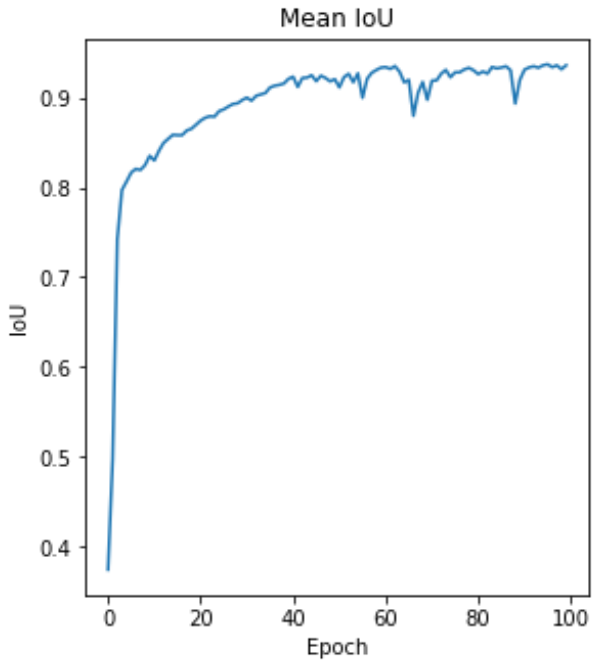


Figure 4.2: Mean IoU (Intersection Over Union) achieved during validation of segmentation model.

4.1.2 Training

Segmentation model was trained for 100 epochs using a batch size of 2 and a learning rate of 0.0001 using the Adam optimizer. The model obtained a dice loss of 0.8383, an accuracy of 0.9540, and a mean IoU of 0.9383 during training. The training was done on 2380 slices, approximately 70% of the data.

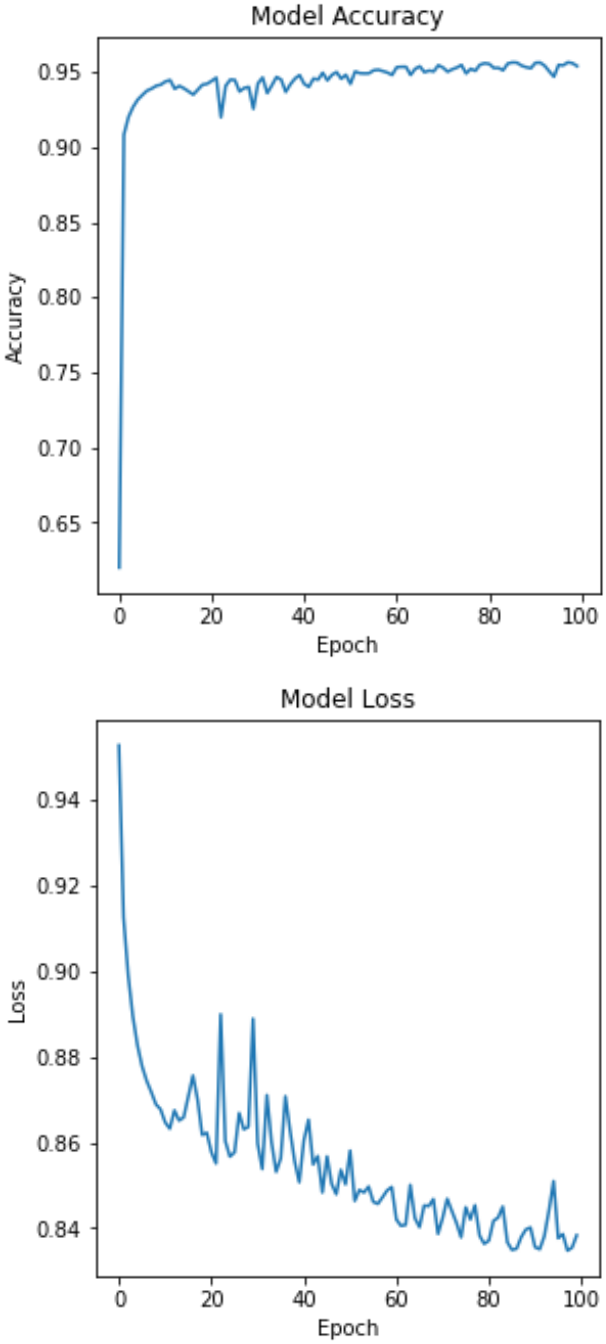


Figure 4.3: Accuracy and loss achieved during training of segmentation model.

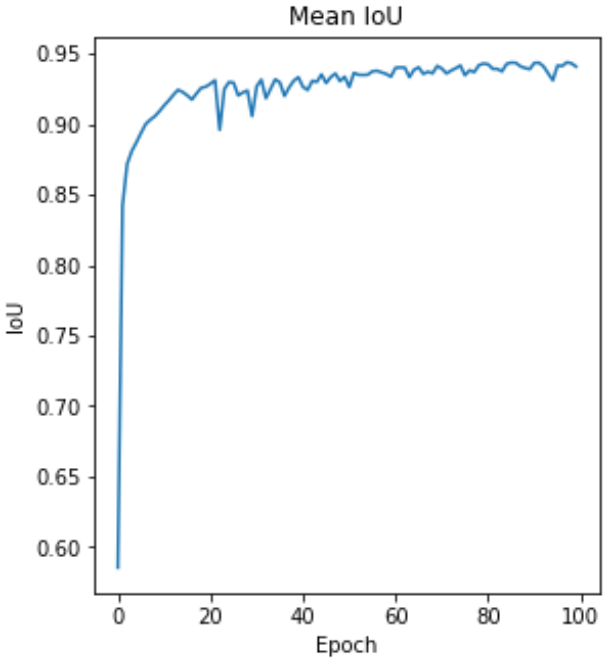


Figure 4.4: Mean IoU (Intersection Over Union) achieved during training of segmentation model.

4.1.3 Testing

Segmentation model achieved a dice loss of 0.8498, an accuracy of 0.9517, and a mean IoU of 0.9383 during testing. In **Figure 4.5** a sample from testing data can be seen giving a visual representation of the model performance comparing the corresponding input image and ground truth from the BraTS 2020 dataset to the segmented output image from the model.

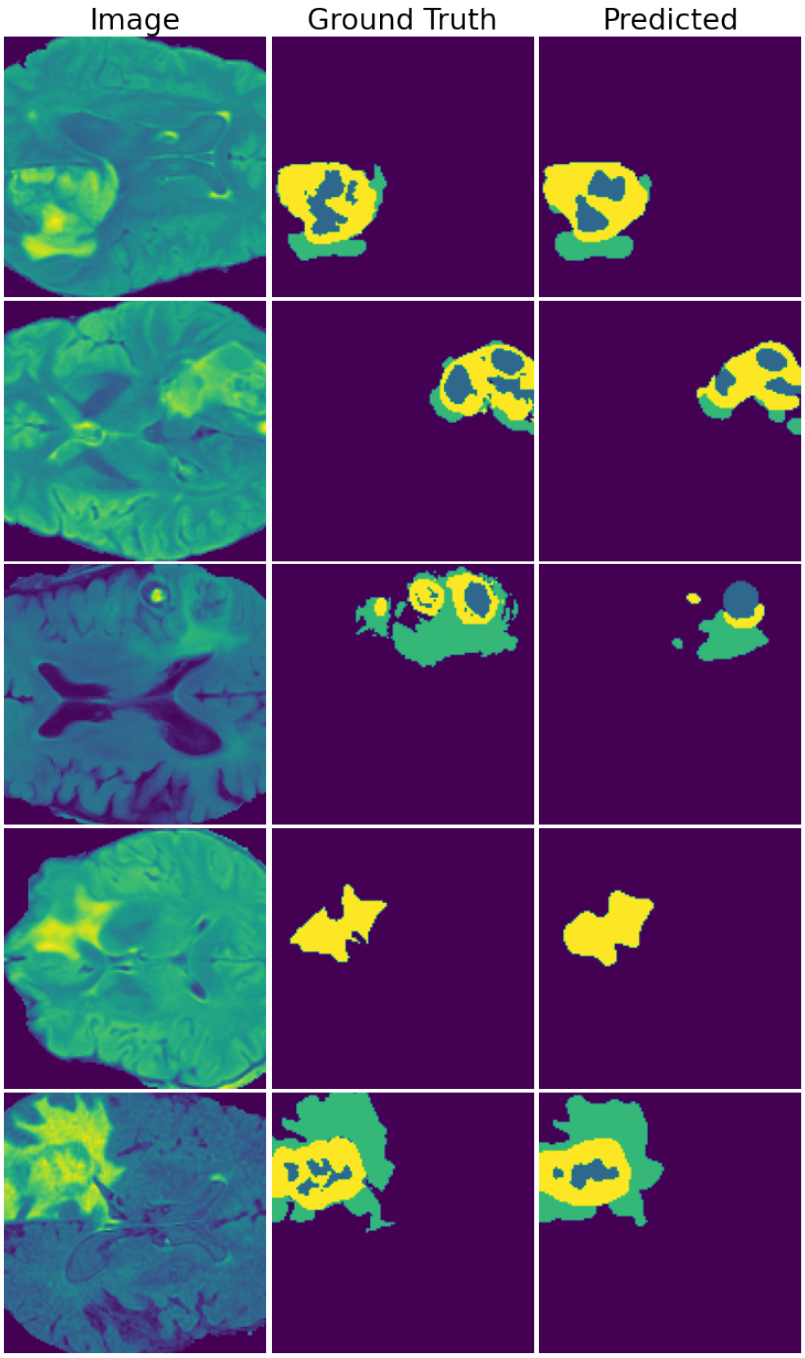


Figure 4.5: Visual comparison of input image, ground truth segmentation and output segmentation of segmentation model.

As can be seen from the visual testing result in **Figure 4.5** the model is able to localize the tumor area quite well. Model is doing a great job finding the tumor location, size, and shape, but somewhat worse on the correct tumor label. An example of this can be seen in **Figure 4.6**. Here, the model predicts the location, size, and shape but identifies the wrong class label for the tumor.

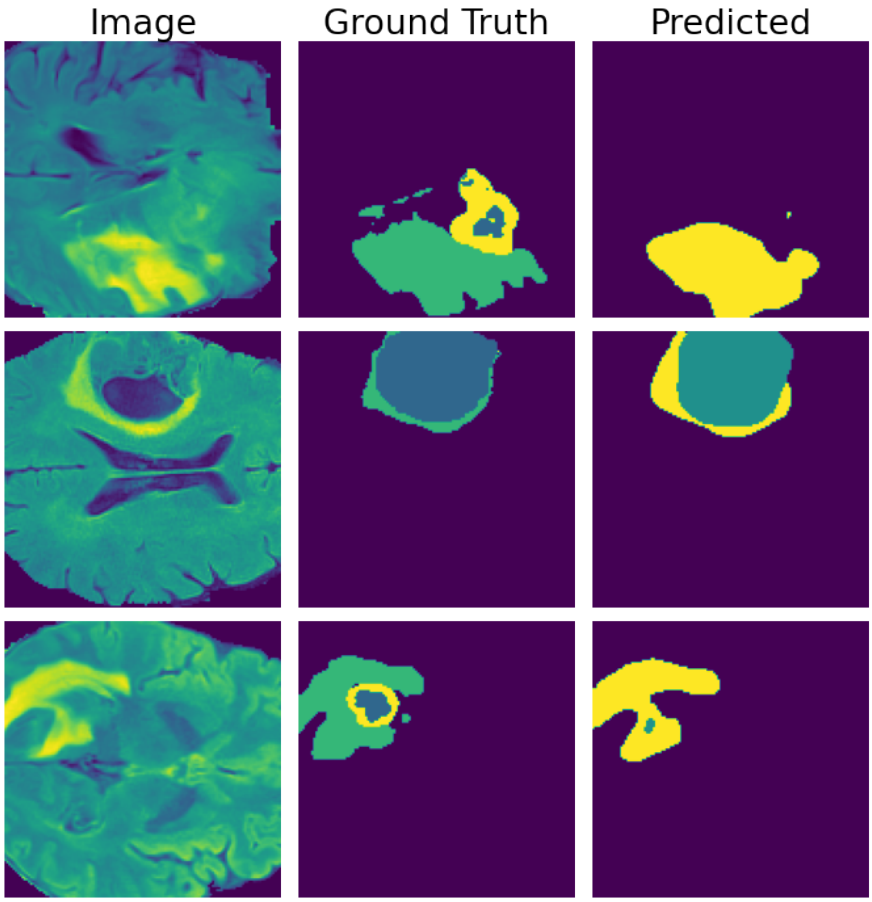


Figure 4.6: Example showing how the model correctly predicts the tumor location, shape and size while wrongfully classifying the tumor class labels.

4.2 Classification Model Validation and Training Results

To evaluate the classification model, binary cross-entropy has been used as a loss function, while accuracy, the area under the curve (AUC), precision, recall, and confusion matrix have been used as performance metrics. When calculating true positive, true negative, false positive, and false negative for usage in precision, recall, and confusion matrix, LGG has been used as the positive class and HGG as the negative class.

A total of 3 models were attempted during validation, where 2 out of 3 returned promising results. One of them had issues with overfitting, and this was attempted to be fixed using regularization techniques such as dropout and L2-regularization, which improved the model a bit. However, it was still outperformed by the 3rd model, which results will be presented in this section.

4.2.1 Validation

The classification model obtained a loss value of 0.043, accuracy of 0.98, AUC of 0.998, precision and recall of 0.984, and 0.982, respectively. Validation was performed on 20% of the dataset, accounting for 3485 image slices, where 1924 comes from HGG and 1561 come from LGG.

During validation, the model was also tested up against the test split of the dataset. This was done as there were issues with overfitting. Thus, it was hard to tell if the result from the validation was good or not without having data the model had not seen yet.

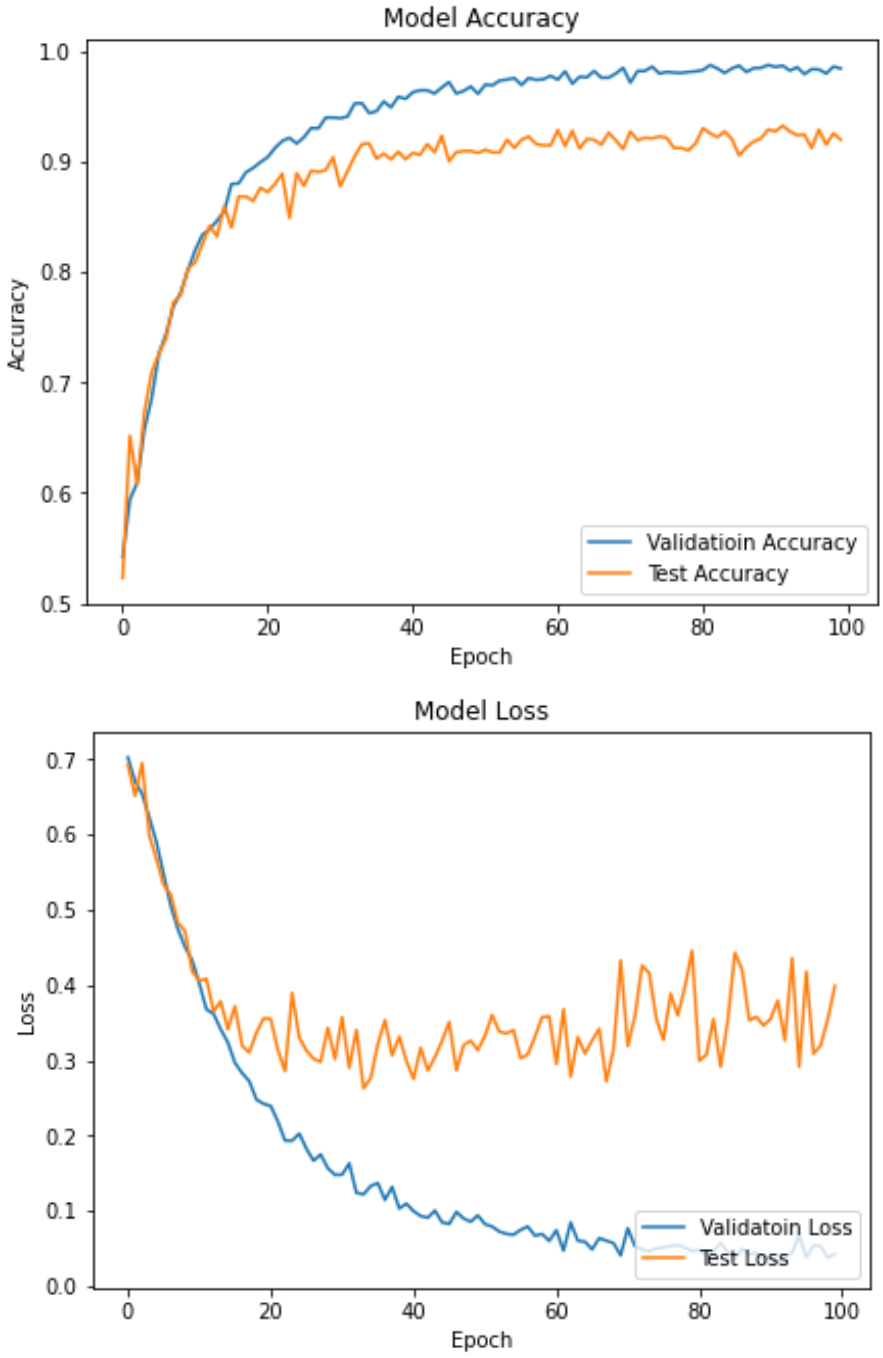


Figure 4.7: Top plot shows classification accuracy and bottom plot shows classification loss.

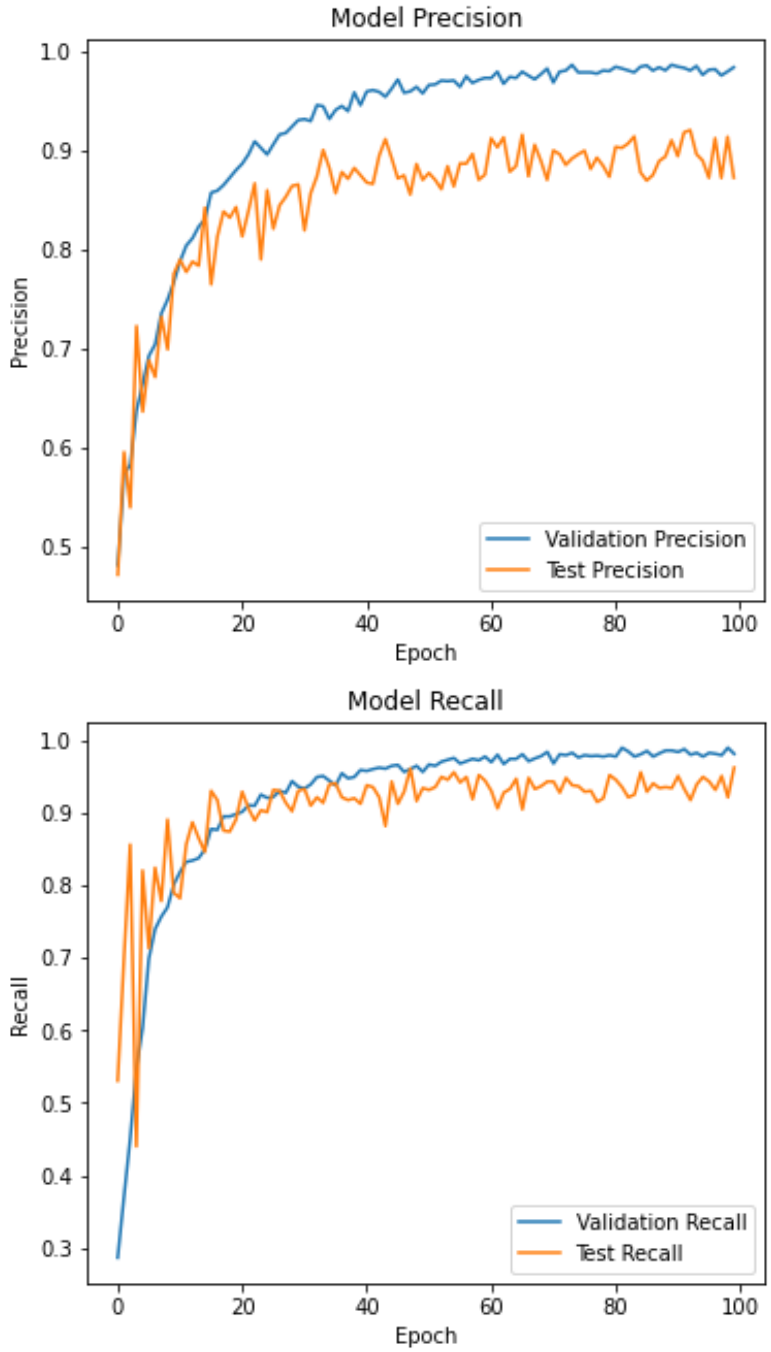


Figure 4.8: Recall and precision during validation of classification model.

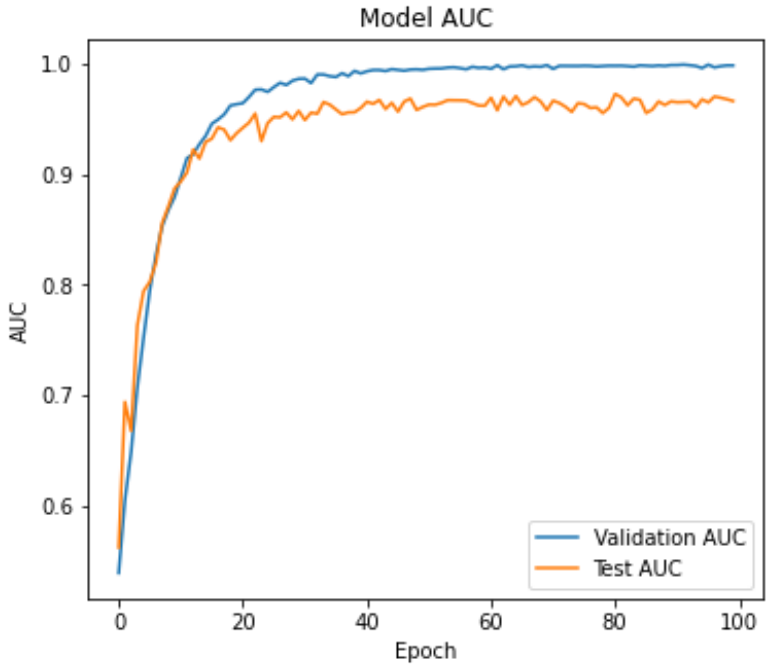


Figure 4.9: Area Under the Curve (AUC) for classification model.

As can be seen in **Figures 4.7, 4.8** and **4.9**, the classification model starts to converge and achieve high values of accuracy, AUC, precision, and recall as early as 20-40 epochs in, depending on the metric. After this, it is still learning and improving its results. However, it is with diminishing returns, as can be seen in **Table 4.1**. This has been taken into consideration during the model's training and will be discussed in the training subsection of the classification model.

Epochs	Loss	Accuracy	AUC	Precision	Recall
20	0.2426	0.899	0.963	0.880	0.899
40	0.1100	0.957	0.991	0.946	0.959
60	0.060	0.978	0.997	0.973	0.978
80	0.047	0.982	0.998	0.980	0.980
100	0.043	0.985	0.998	0.984	0.982

Table 4.1: Table illustrating the diminishing returns from increasing numbers of epochs during validation.

Confusion matrix in **Figure 4.10** summarizes how many images has been correctly and incorrectly predicted. On the diagonal shows the correctly predicted images and we can tell from the count that the model rarely predicts incorrectly during validation.

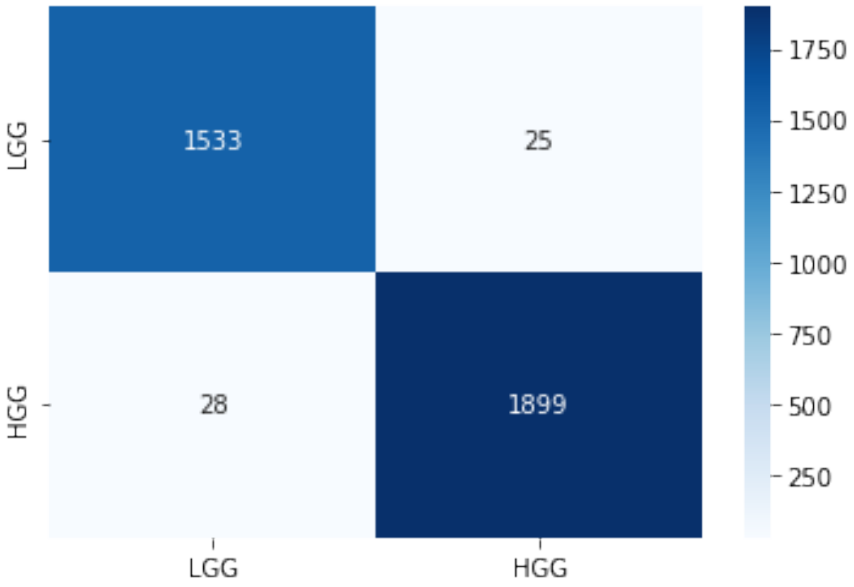


Figure 4.10: Confusion matrix results from validation of the classification model. Rows show the predicted labels while columns show the true labels.

4.2.2 Training and Testing

Model obtains a training loss of 0.031, accuracy of 0.989, AUC of 0.998, precision of 0.988 and a recall of 0.988. During testing the model obtained a loss of 0.025, accuracy of 0.992, AUC of 0.999, precision of 0.994 and a recall of 0.988. A more readable representation can be seen in **Table 4.2**.

Classification model was trained on 12199 image slices and tested on 1744 image slices responding to 70% and 10% of the data, respectively. Training was done using binary-crossentropy as loss function, Adam as optimizer using a learning rate of 0.001, a batch size of 64 and the model was trained for 60 epochs. During validation a total number of 100 epochs was used, but as mentioned in chapter 4.2.1, the model reaches high metric values during the earlier epochs and improves with diminishing returns during the later epochs as can be seen in **Table 4.1**. To reduce the time to train the model, a lower number of epochs was used for this reason and as can be seen in **Figures 4.11, 4.12** and **4.13** the model achieves good metric values during training and testing even with lower number of epochs.

	Loss	**Accuracy**	**AUC**	**Precision**	**Recall**
Training	0.031	0.989	0.999	0.988	0.988
Testing	0.025	0.992	0.999	0.994	0.988

Table 4.2: Results for the classification model on training and testing dataset.

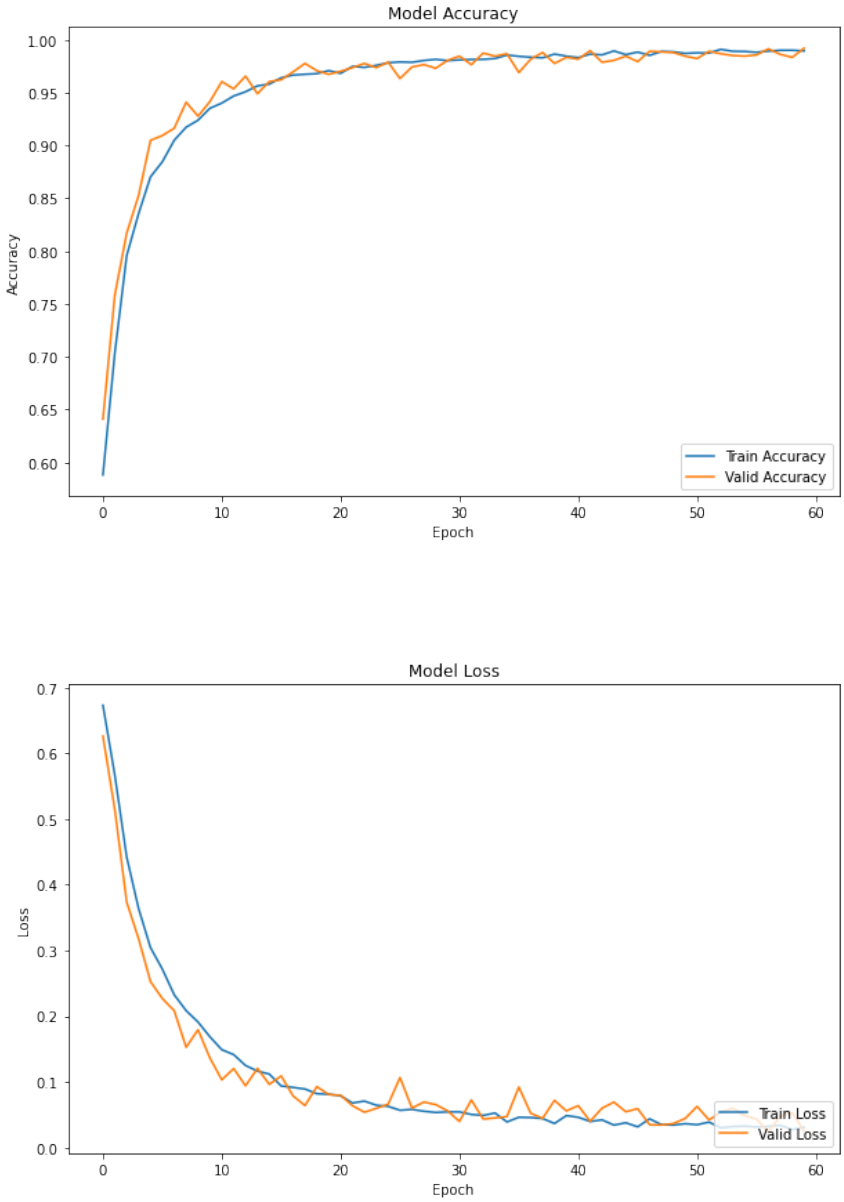


Figure 4.11: Accuracy and loss per epoch during training and testing for classification model. Accuracy in the top plot, loss in the bottom plot.

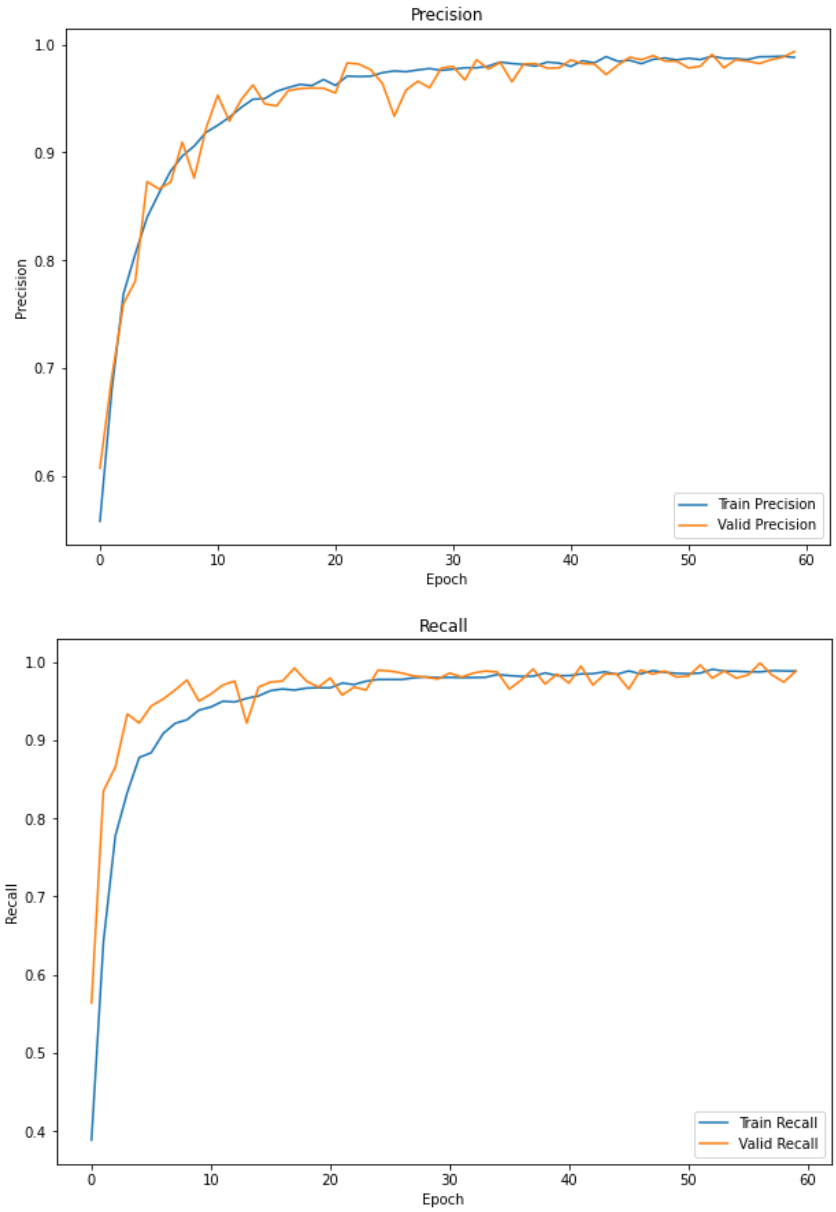


Figure 4.12: Precision and recall per epoch during training and testing for classification model.

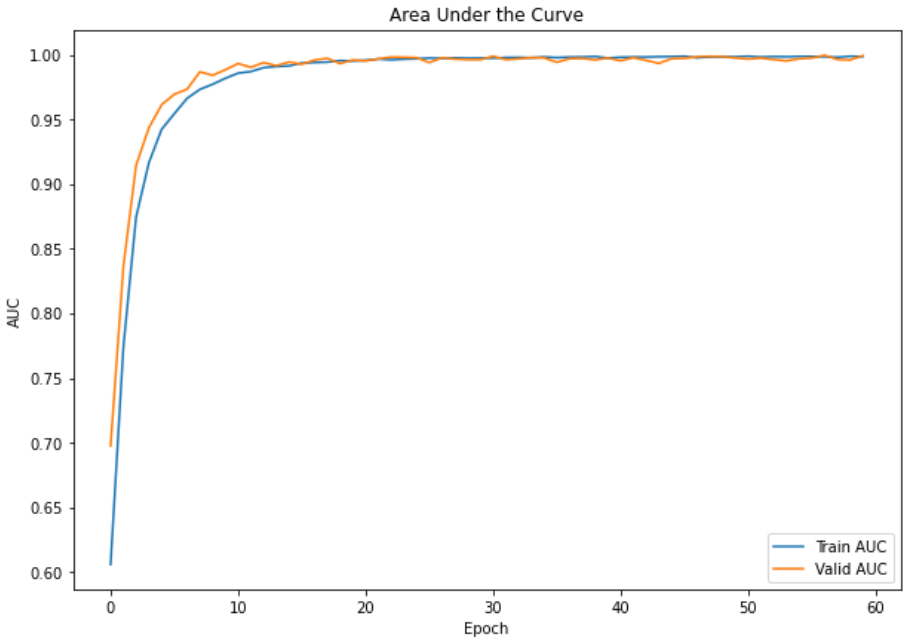


Figure 4.13: AUC (Area Under the Curve) per epoch during training and testing for classification model.

Confusion matrices showing the predicted results of training and testing data can be seen in **Figure 4.14**. On the diagonal we see the number of correctly predicted images for each class and the number of wrongly predicted on the anti-diagonal.

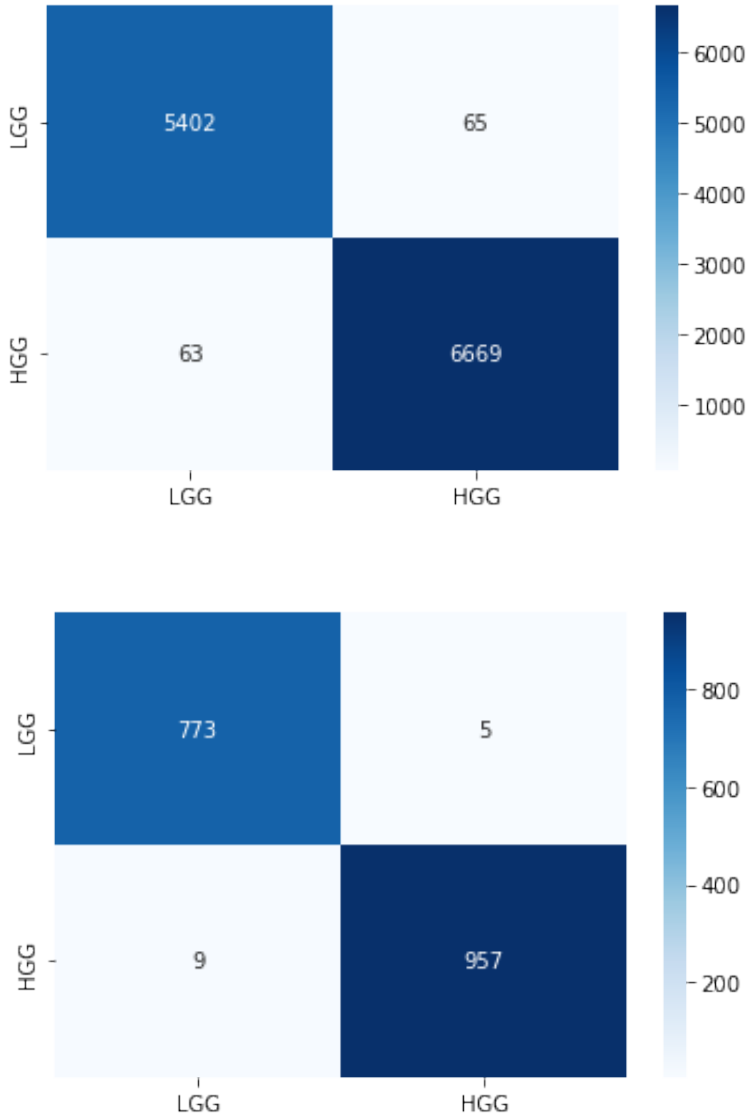


Figure 4.14: Confusion matrices for training (Top) and testing (Bottom). Rows show the predicted labels while columns show the true labels.

Chapter 5

Conclusion

This thesis aimed to classify tumor types based on data from multi-parametric magnetic resonance imaging (MRI) using deep learning. I split the project question into two parts, segmentation and classification, to allow for more in-depth image analysis than what would have been achieved from a pure classification task.

Modified version of U-Net, a state-of-the-art convolutional neural network for biomedical image segmentation, was used to perform image analysis. The segmentation model has good performance where it manages to predict tumor location, size and shape with high accuracy with slightly lower performance on internal tumor label prediction. In general, the inter tumor label performance was not an issue as the information was redundant and not required for further use in classification. It was rather additional data that allowed for greater analysis to base the model evaluation.

For classification, a convolutional neural network was performed on segmented tumor images. A total of three different models were examined, and two of them provided promising results. One of the models provided overfitting during training and thus was incapable of generalizing the trained outcomes, leading to undesired classification results on the test dataset. Regularization was attempted on the model to no avail. Finally, I chose the model with the best performance, providing the desired training outcome with justifiable high classification accuracy. The model showed that it could

learn features with ease as high values for accuracy were obtained in the early stages of training, allowing for shorter training times.

Both segmentation and classification networks have great performance considering their relatively shallow network architectures and short training times. Both networks were trained on a laptop without a dedicated graphics card using a 2.3 GHz Quad-Core Intel Core i5 CPU. It took approximately 5 and a half hours, and 4 hours to train segmentation and classification networks, respectively.

5.1 Future Work

The BraTS repository used in this study consists of 369 multi-institutional 3D images acquired from glioma pateints, where 2D images were generated from this cohort. This is a relatively small dataset in the context of deep learning, which may hamper the generalizability of the model's predictions. Extension of the dataset by including data from other sources could help obtain a model that is not as prone to being biased and rather inclined to generalize.

A future study may analyze internal tumor label prediction to determine if any class labels are over- or under-represented to adjust class weights further. This can help with the classification of internal tumor labels and thus improve the accuracy of the model.

Bibliography

[1] Brats 2020 challenge. https://www.med.upenn.edu/cbica/brats2020/data.html.

[2] Cancer genome atlas research network. comprehensive, integrative genomic analysis of diffuse lower-grade gliomas. *N. Engl. J. Med.*, 372:2481–2498, 2015.

[3] *Magnetic Resonance Imaging*, chapter 4, pages 211–252. John Wiley Sons, Ltd, 2019.

[4] Foad Kazemi Amin Kabir, Moosa Ayati. Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. https://www.researchgate.net/publication/328373113_Magnetic_resonance_imaging-based_brain_tumor_grades_classification_and_grading_via_convolutional_neural_networks_and_genetic_algorithms, 2018.

[5] Florentin Bieder, Robin Sandkühler, and Philippe C. Cattin. Comparison of methods generalizing max- and average-pooling. *CoRR*, abs/2103.01746, 2021.

[6] S. Birchfeld. *Image Processing Analysis*. Cengage Learning, 2016.

[7] Stefan Bauer Bjoern H Menze, Andras Jakab. The multimodal brain tumor image segmentation benchmark (brats). https://pubmed.ncbi.nlm.nih.gov/25494501/, 2015.

[8] Kevin Chu. An introduction to sensitivity, specificity, predictive values and likelihood ratios. *Emergency Medicine*, 11(3):175–181, 1999.

[9] Smith K. et al. Clark K., Vendt B. The cancer imaging archive (TCIA): Maintaining and operating a public information repository. *J. Digit. Imaging*, 26:1045–1057, 2013.

[10] et al. De Coene BHajnal JV, Gatehouse P. Mr of the brain using fluid-attenuated inversion recovery (flair) pulse sequences. *AJNR Am J Neuroradiol*, 13:1555–1564, 1992.

[11] Pinho MC Rofsky NM Sherry AD De León-Rodríguez LM, Martins AF. Basic mr relaxation mechanisms and contrast agent design, 2015.

[12] Corrales-García Delgado-López, P.D. Survival in glioblastoma: a review on the impact of treatment modalities. `https://doi.org/10.1007/s12094-016-1497-x`, 2016.

[13] Daniel Soudry Elad Hoffer, Itay Hubara. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. `https://arxiv.org/pdf/1705.08741.pdf`, 2018.

[14] Kasuboski L Pattany PM De Coene B Lewis PD Pennock JM Oatridge A Young IR Bydder GM Hajnal JV, Bryant DJ. Use of fluid attenuated inversion recovery (flair) pulse sequences in mri of the brain, 1992.

[15] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[16] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[17] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. `https://arxiv.org/abs/1505.04597`, 2015.

[18] Faith G. Davis Isabelle Deltour James L. Fisher Chelsea Eastman Langer Melike Pekmezci Judith A. Schwartzbaum Michelle C. Turner Kyle M. Walsh Margaret R. Wrensch Jill S. Barnholtz-Sloan Quinn T. Ostrom, Luc Bauchet. The epidemiology of glioma in adults: a "state of the science" review. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4057143/`, 2014.

[19] Daniel Ramos, Javier Franco-Pedroso, Alicia Lozano-Diez, and Joaquin Gonzalez-Rodriguez. Deconstructing cross-entropy for probabilistic binary classifiers. *Entropy*, 20(3):208, Mar 2018.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[21] Simone Sharma Siddharth Sharma. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 4:310–316, 2020.

[22] David Vazquez Adriana Romero Yoshua Bengio Simon Jégou, Michal Drozdzal. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. `https://arxiv.org/abs/1611.09326`, 2016.

[23] Thorvald Julius Sørensen. *A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons.* I kommission hos E. Munksgaard, 1948.

[24] Andras Jakab Spyridon Bakas, Mauricio Reyes. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. `https://arxiv.org/abs/1811.02629`, 2018.

[25] Aristeidis Sotiras Michel Bilello Martin Rozycki Justin S Kirby John B Freymann keyvan Farahani Christos Davatzikos Spyridon Bakas, Hamed Akbari. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. `https://pubmed.ncbi.nlm.nih.gov/28872634/`, 2017.

[26] Sergios Theodoridis. *Machine Learning: A Bayesian and Optimization Perspective.* 2nd edition, 2020.

[27] Mark Jenkinson Vaanathi Sundaresan, Ludovica Griffanti. Brain tumour segmentation using a triplanar ensemble of u-nets on mr images. `https://arxiv.org/pdf/2105.11356.pdf`, 2021.

[28] Dietrich Van der Weken, Mike Nachtegael, and Etienne E. Kerre. Using similarity measures and homogeneity for the comparison of images. *Image and Vision Computing*, 22(9):695–702, 2004.

[29] Geoffrey Hinton Yan LeCun, Yoshua Bengio. Deep learning. pages 349–440, 2015.

[30] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

Appendix A

Poster

Brain Tumor Segmentation and Cla

Sander

Introduction

Gliomas are on the most common types of malignant brain tumors in adults and even though they are rare, they have a high rate of mortality [4]. Magnetic resonance imaging (MRI) plays a vital role in diagnosing treatment evaluation of glioma tumors. As gliomas differ in aggresiveness as well as in shape size and location, the task of segmenting MR images can be challenging due to different types of gliomas having heterogeneous properties.

Due to the process of manually diagnosing being a difficult and time consuming task, an interesting question is the ability to have an automated analyzing tool that can be used for diagnosis and assessment of how aggresive a tumor is. As MRI is a common approach for diagnostization, the automation of diagnosing these kinds of images would be an exciting approach. Deep learning methods such as convolutional neural networks (CNN), a subclass of artificial intelligence, have proven to be quite efficient at recognising patterns in images when there is a large amount of data available. An automated analyzing tool will have the

Res

Segmentation

To evaluate performance of t
loss, accuracy and intersectio

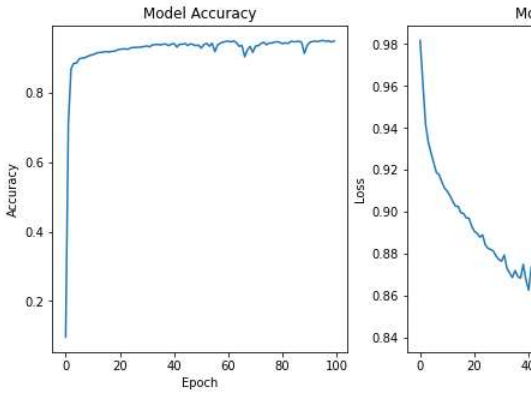
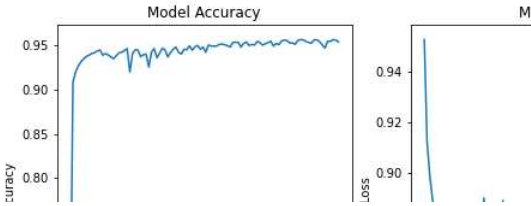


Figure 4. Validation results

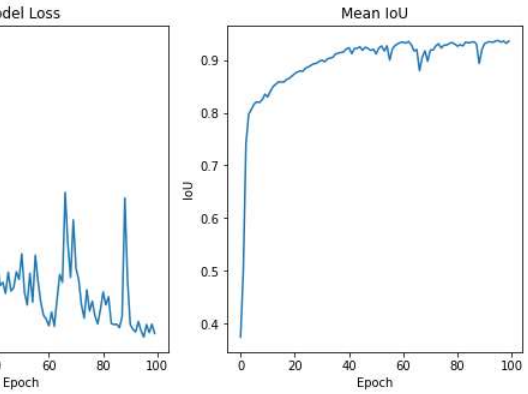


assification Using Neural Networks

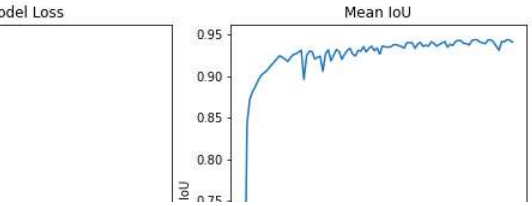
Takvam

ults

the segmentation model dice
on over union was used.



s for segmentation model.



Conclusion

The aim for this thesis was to classify tumor type based on data from multi-parametric magnetic resonance imaging (MRI) using deep learning. Problem was split into two parts, segmentation and classification, to allow for more in depth image analysis than what would be able to achieve from a pure classification task.

Modified version of U-Net, a state-of-the-art convolutional neural network for biomedical image segmentation, was used to perform image analysis. The segmentation model has good performance where it manages to predict tumor location, size and shape with high accuracy with a slightly lower performance on internal tumor label prediction. In general, the inter tumor label performance was not an issue as the information was redundant and not required for further use in classification. It was rather additional data that allowed for greater analysis to base the model evaluation on.

benefit of being able to recognise patterns and correctly classify a tumor in a fraction of the time that it would take a skilled professional to do the same.

The goal of this thesis is to develop a deep learning model that will be able to segment brain tumors in MRI and use the segmented images to further classify the aggressiveness of the tumors.

Methods

An open source multimodal brain tumor MRI dataset known as BraTS [2, 6, 5], was used for this thesis. It consists of 369 3D images from glioma patients where 293 has been diagnosed with high-grade glioma and 76 have been diagnosed with low-grade glioma.

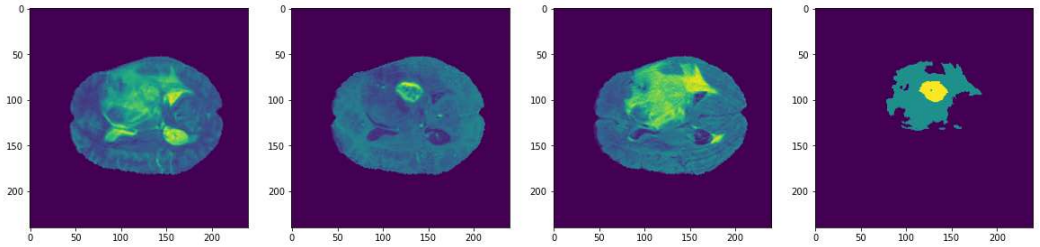


Figure 1. BraTS consists of 4 types of modalities. Here 3 of them can be seen including the tumor segmented image. These are all images from the same patient. From left to right: T1ce, T2, FLAIR and segmented.

A modified version of the convolutional neural network, U-Net [3], was used for the segmentation task. This is a neu-

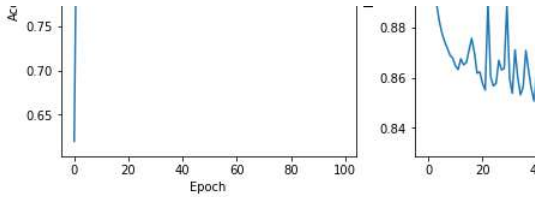


Figure 5. Training results

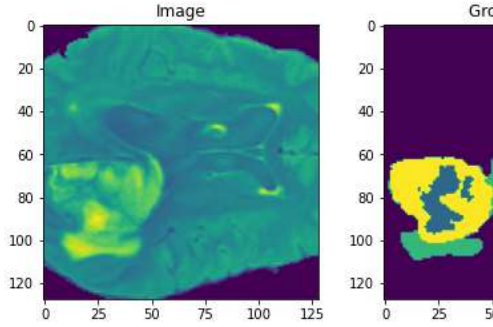
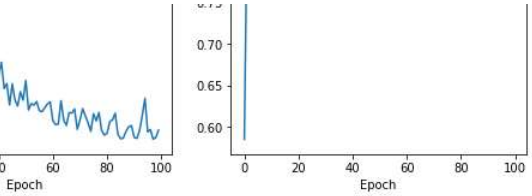


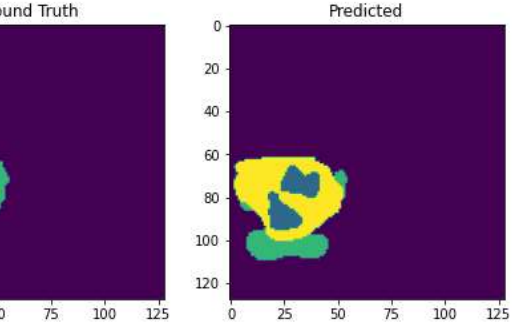
Figure 6. Illustration of model inpu
versus ground truth.

Segmentation model achieve
racy of 0.9496 and a mean Io
performed on 10% of the da
slices. Validation was repeate
samples to ensure an unbias
idation sample and similar re
samples.

Classification



for segmentation model.



t, and comparison of output

d a dice loss of 0.8421, accu-
oU of 0.9356. Validation was
ataset which accounts to 363
ed multiple times on different
ed result due to current val-
sult were found for different

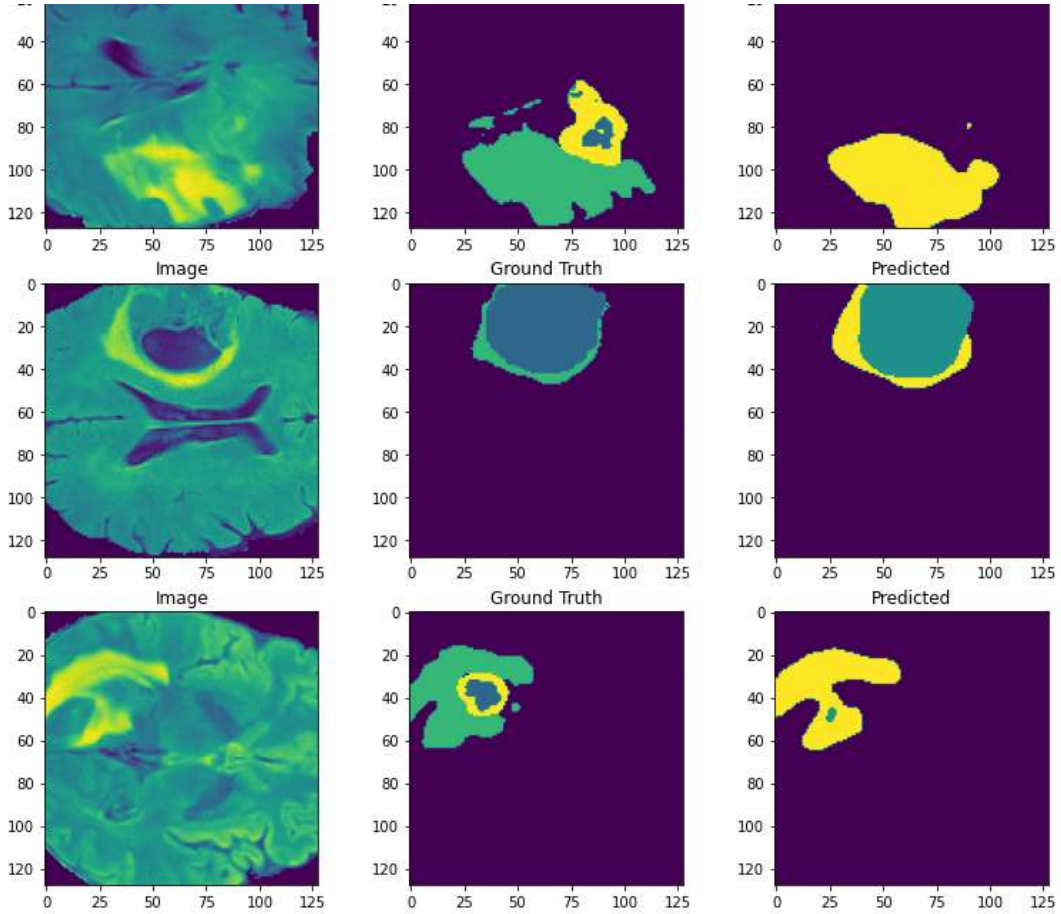


Figure 9. Example showing how the model correctly predicts the tumor location, shape and size while wrongfully classifying the tumor class labels.

For classification, a convolutional neural network using segmented tumor images was used. A total of 3 different models were attempted where the 2 of them had promising results. One of them had issues with overfitting and thus an inability to generalize leading to terrible results on test dataset. Regularization was attempted on the model

ral network commonly used for biomedical image segmentation. The one used differs from the original essentially in the input and output size of each image, where the original has an input of $572 \times 572 \times 1$ and output of $388 \times 388 \times 2$ whereas the one used has an input of $128 \times 128 \times 3$ and output of $128 \times 128 \times 4$.

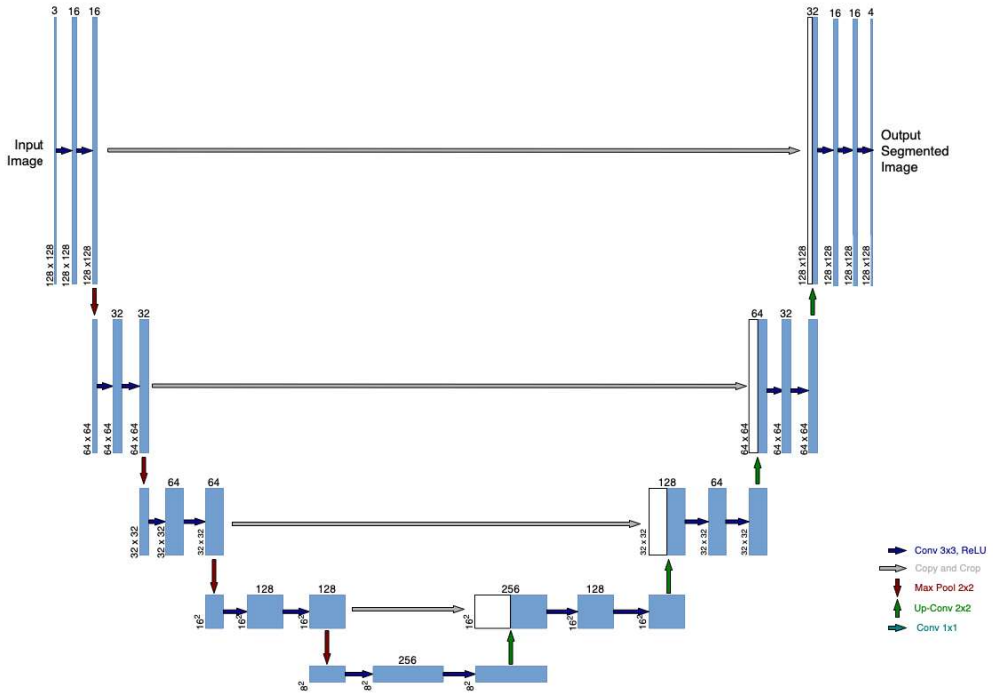


Figure 2. Illustration of a modified version of the U-Net architecture used for the segmentation task.

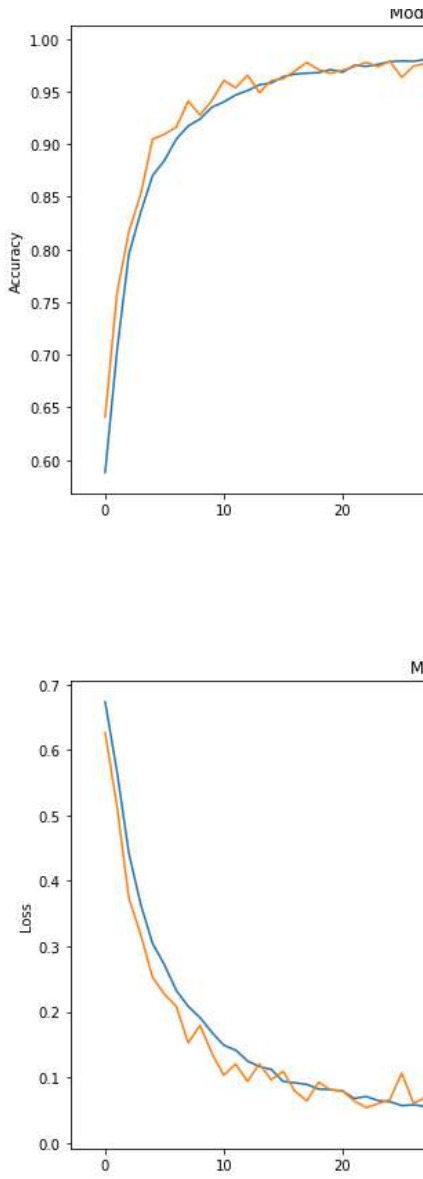
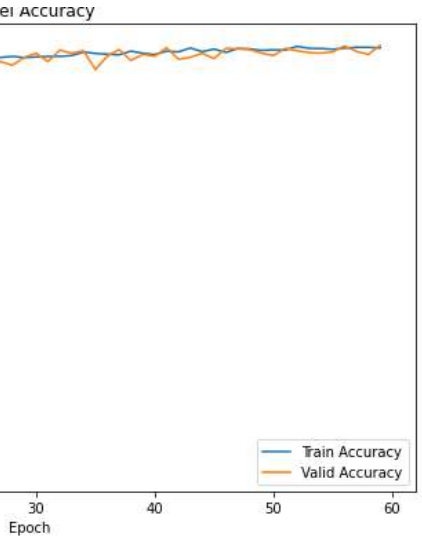
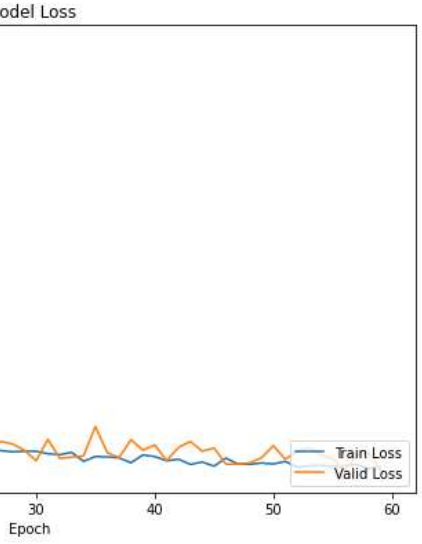


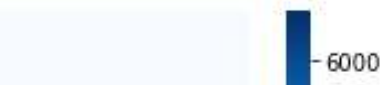
Figure 7. Illustration of model inpu versus ground truth.

Model used for classification is a convolutional neural network with 2 fully connected layers attached to the end of





t, and comparison of output



to no avail. Final model achieved good results with high accuracy. The model showed that it had the ability to learn features with ease as high values for accuracy was obtained in early stages of training allowing for shorter training times.

Both segmentation and classification networks have great performance considering their relatively shallow network architectures and short training times. Both networks were trained on a laptop without a dedicated graphics card using a 2.3 GHz Quad-Core Intel Core i5 CPU. It took approximately 5 and a half hours, and 4 hours to train segmentation and classification networks, respectively.

Future Work

BraTS consists of only 369 3D images where every 2D image was sampled from these patients. This is quite a small dataset in the context of deep learning which can cause issues with generalization. Extension of the dataset by including data from other sources could help with obtaining a model that is not as prone to being biased and rather be inclined to generalize.

Analysis of internal tumor label prediction to find if any of the class labels are over- or under-represented to further adjust class weights. This can help with the classification of internal tumor labels and thus improve the accuracy of the model.

it [1]. Final layers flatten features collected during convolutions into 2D arrays that is further used to compute the output value for classification.

This model uses a modified version of the output from the segmentation model as input. Essentially what it does is based on the segmented tumor it classifies the tumor as either low-grade or high-grade glioma where the output value of the network is the classification of either of the two gliomas.

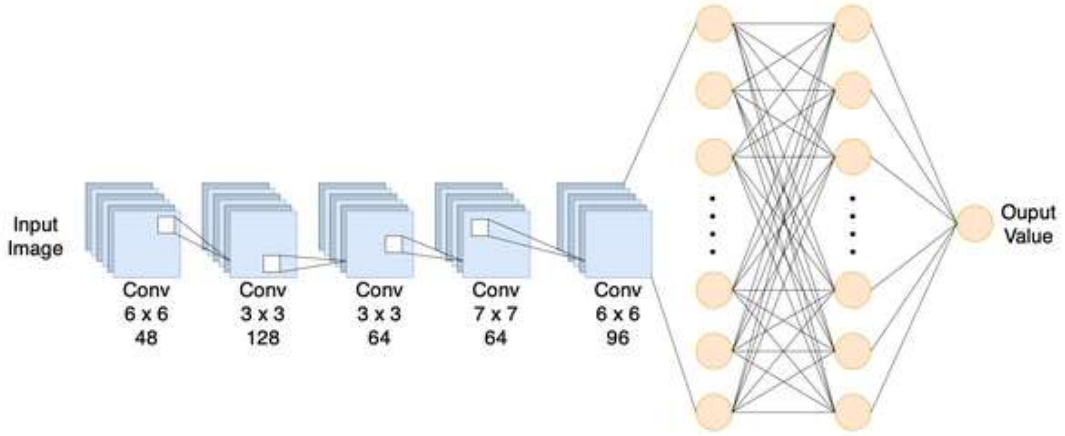


Figure 3. Illustration of the architecture used of for the classification task.

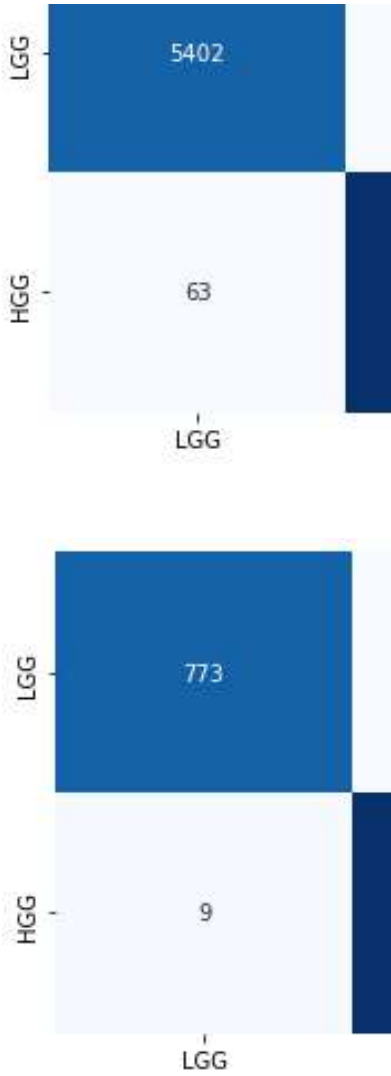
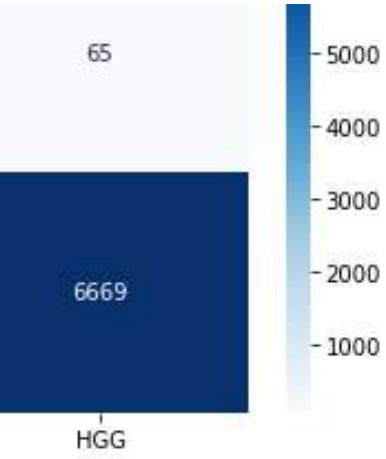
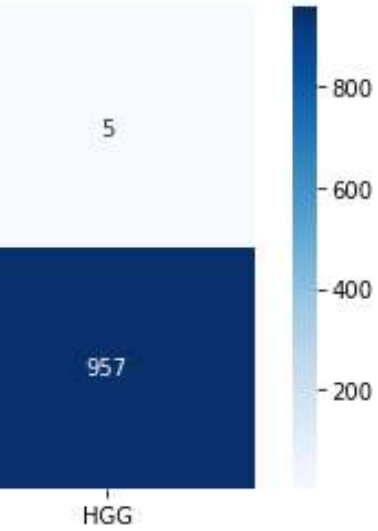


Figure 8. Confusion matrices for tr
Rows show the predicted labels wh





aining (Top) and testing (Bottom).
hile columns show the true labels.

References

[1] Foad Kazemi Amin Kabir, Moosa Ayati. Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. `https://www.researchgate.net/publication/328373113_Magnetic_resonance_imaging-based_brain_tumor_grades_classification_and_grading_via_convolutional_neural_networks_and_genetic_algorithms`, 2018.

[2] Stefan Bauer Bjoern H Menze, Andras Jakab. The multimodal brain tumor image segmentation benchmark (brats). `https://pubmed.ncbi.nlm.nih.gov/25494501/`, 2015.

[3] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. `https://arxiv.org/abs/1505.04597`, 2015.

[4] Faith G. Davis Isabelle Deltour James L. Fisher Chelsea Eastman Langer Melike Pekmezci Judith A. Schwartzbaum Michelle C. Turner Kyle M. Walsh Margaret R. Wrensch Jill S. Barnholtz-Sloan Quinn T. Ostrom, Luc Bauchet. The epidemiology of glioma in adults: a “state of the science” review. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4057143/`, 2014.

[5] Andras Jakab Spyridon Bakas, Mauricio Reyes. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge. `https://arxiv.org/abs/1811.02629`, 2018.

[6] Aristeidis Sotiras Michel Bilello Martin Rozycki Justin S Kirby John B Freymann keyvan Farahani Christos Davatzikos Spyridon Bakas, Hamed Akbari. Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features. `https://pubmed.ncbi.nlm.nih.gov/28872634/`, 2017.