TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Detecting and Understanding Glioblastoma Multiforme Using Convolutional Neural Networks

zur Erlangung des akademischen Grades

DIPLOM-INGENIEURIN

im Rahmen des Studiums

BIOMEDICAL ENGINEERING

eingereicht von

DIPL.-ING. MARIELLA GLORIA GREGORICH
Matrikelnummer: 1127193

ausgeführt am Institut für Wirtschaftsmathematik und Stochastik
der Fakultät für Mathematik und Geoinformation
der Technischen Universität Wien

Betreuung:
Univ.Prof. Dipl.-Ing. Dr.techn. Peter FILZMOSER
Assoc.Prof. Dipl.-Ing. Dr. Georg LANGS

Wien, am 30.04.2019

(Unterschrift Verfasserin) (Unterschrift Betreuer)

Abstract

Glioblastoma multiforme (GBM) is among the most lethal neoplasms associated with extremely poor overall survival (OS) of patients. The median survival time of patients diagnosed with GBM is around 16 months. Understanding the nature of these aggressive tumors through the analysis of imaging data capturing the malignant tissue can provide beneficial information for the prognosis and treatment of the patient.

In this master thesis, the state-of-the-art machine learning (ML) methods based on convolutional neural networks for brain tumor image analysis of MRI scans will be thoroughly assessed. In particular, the work will focus on i) the detection and segmentation of glioblastoma multiforme in pre-operative MRI scans, ii) the prediction of overall survival from pre-operative MRI scans and additional clinical features from patients with glioblastoma, and iii) assessing the potential of improved survival prediction by reducing the task to overall survival group classification between short-term (< 24 months) and long-term (≥ 24 months) survival at the time of the pre-operative MRI scan.

Promising results were obtained for the task of tumor segmentation as well as survival prediction even though the data set was small which highlights the capability and limitations of convolutional neural networks in radiomics. However, the thesis also empathizes that overall survival in GBM patients is heterogeneous and cannot be solely determined by pre-operative MRIs with current computational approaches. Therefore, the scientific future lies in computational approaches combining the power of deep learning algorithms with additional clinical features to achieve accurate survival predictions and risk assessment that may help clinical decision making.

Zusammenfassung

Glioblastoma multiforme (GBM) gehört zu den tödlichsten Neoplasmen, die mit einem extrem schlechten Gesamtüberleben der Patienten einhergehen. Die mittlere Überlebenszeit von Patienten mit GBM liegt bei etwa 16 Monaten. Das Verständnis der Natur dieser aggressiven Tumore durch die Analyse von Bilddaten, die das maligne Gewebe erfassen, kann nützliche Informationen für die Prognose und Behandlung des Patienten liefern.

In dieser Masterarbeit werden moderne Machine-Learning Verfahren basierend auf neuronalen Faltungsnetzen für die Bildanalyse von Hirntumoren von MRI-Scans gründlich untersucht. Die Arbeit konzentriert sich insbesondere auf i) die Erkennung und Segmentierung von Glioblastoma multiforme in präoperativen MRI-Scans, ii) die Vorhersage des Gesamtüberlebens basierend auf präoperativen MRI-Scans und zusätzlichen klinische Merkmalen von Patienten mit Glioblastoma und iii) die Beurteilung des Potenzials einer verbesserten Überlebensvorhersage, indem die Aufgabe des Machine Learning Algorithmus auf die Klassifizierung der Gesamtüberlebensgruppe zwischen kurzzeitiger (< 24 Monate) und langfristiger (≥ 24 Monate) Überlebenszeit zum Zeitpunkt der präoperativen MRI-Untersuchung reduziert wird.

Vielversprechende Ergebnisse wurden für die Aufgabe der Tumorsegmentierung sowie für die Überlebensvorhersage erhalten, obwohl der verfügbare Datensatz klein war, was die Möglichkeiten und Grenzen der neuralen Faltungsnetze in der Radiomik hervorhebt. Die Masterarbeit verdeutlicht jedoch auch, dass das Gesamtüberleben bei GBM-Patienten heterogen ist und nicht allein durch präoperative MRI unter Verwendung von aktuellen rechnerischen Ansätzen bestimmt werden kann. Daher liegt die wissenschaftliche Zukunft in rechnerischen Ansätzen, die die Leistungsfähigkeit von Deep-Learning-Algorithmen mit klinischen Merkmalen kombinieren, um genaue Überlebensvorhersagen und Risikobewertungen zu erzielen, die bei der klinischen Entscheidungsfindung hilfreich sein können.

Acknowledgements

This master thesis would not have been possible without the advice and support of several individuals who in one way or another contributed their valuable time and knowledge in the preparation and completion of this work.

First and foremost I want to thank Professor Georg Langs from the Computational Imaging Research Lab (CIR) at the Medical University of Vienna for his supervision, support and assistance throughout the process of this work. He has contributed significantly to the creation of this work through his constructive suggestions and feedback. Thank you for the opportunity to be part of your research team during the course of this project.

Secondly, I want to thank Professor Peter Filzmoser from the Institute of Statistics and Mathematical Methods in Economics of the University of Technology Vienna. Thank you for your willingness to take on the co-supervision of my second master thesis and providing thoughtful and helpful feedback to give this work its final touch.

Many thanks go in particular to Karl-Heinz Nenning and Thomas Roetzer for their valuable advice, guidance and constant support from the early stages of the project until the very end. It was a pleasure having you two as my junior supervisors.

My sincerest thanks also go to Philipp Seeböck, Adelheid Wöhrer, Barbara Kiesel and Julia Furtner which offered their support in the most crucial times and always found time to provide valuable thoughts and expertise.

Last but not the least I would like to thank my parents in particular my mother for their unending support, encouragement and love. Without their support this work would not have been possible.

Contents

Chapter 1

Introduction

1.1 Motivation

High-grade glioblastomas constitute one of the most severe and aggressive forms of gliomas in adults and account for around 16% of all brain tumors. Treatment options of diagnosed individuals include surgical resection, radiation and therapy. But even with new advances in treatment and aggressive tumor resection, the median overall survival (OS) of treated patients remains at approximately fifteen months (Thakkar et al., 2014). Patients surviving more than three years after the initial diagnosis are considered long-term survivors (Adeberg et al., 2014). The extremely poor prognosis of the high-grade tumor is mainly based on its considerable extent of temporal and spatial heterogeneity making treatment more difficult and increasing the risk of local malignant recurrences (Klughammer et al., 2018). Biopsy-related analysis of the tumorous tissue disregards the heterogeneity of the tumor by extracting solely a small sample and requires an invasive surgical procedure. Computer-aided diagnostic tools and medical image processing can provide the preoperative, quantitative and non-invasive assessment of brain tumor characteristics and thereby contribute valuable information to the planning of the patient's treatment. In particular, analysis of preoperative magnetic resonance image (MRI) can be an important pre-surgical instrument for extracting descriptive features of the cancer and choosing a suitable therapeutic strategy for patients with glioblastoma. As a result, the study of radiomics has gained an increasing interest in recent years. Radiomics describes novel processes for the high-throughput detection and extraction of quantitative features from radiological imaging data that are difficult for the human eye to perceive and the

subsequent development of predictive models relating these features to clinical outcomes (Gillies et al., 2015).

In recent years, machine learning approaches have contributed significantly to the world of radiomics. Machine learning (ML) is a branch of artificial intelligence, based on the idea that systems can learn from data, recognize predictive patterns and make decisions by using the resulting models with minimal human intervention. Despite its recent success, the principle of machine learning is not a new concept. The general idea has existed since the 1950s, but more powerful graphics processing units (GPUs) and large amounts of available data have enabled ML to tackle increasingly complex learning tasks. Recently, deep learning(DL) a subfield of machine learning exploiting artificial neural network architectures with increasing numbers of layers have gained relevance in this context. Novel deep learning techniques can be used for four main tasks of radiomics: imaging, segmentation, feature extraction and analysis (Vial et al., 2018). Deep learning algorithms are able to learn specific image characteristic (patterns) not easily visible to the human eye by identifying and weighing relevant features from labelled medical imaging data and using the acquired knowledge to solve the desired task. Even though, machine learning originated as an imitation of human thinking, machines are nowadays exceeding human-level performance and are able to perform more complex visual classification problems than humans are able to (Nguyen et al., 2015). Recently, a study by Nguyen et al. (2015) focused on determining if there still are remaining differences in visual recognition between humans and deep neural networks by producing images that are completely unrecognizable to humans. Despite the encoding of the images, the deep neural networks were able to identify the correct class with almost 100% accuracy.

Currently, the most popular deep learning approach used in the context of medical image processing are Convolutional Neural Networks (CNNs) (Tajbakhsh et al., 2016). In radiology, the algorithms can be trained to segment anatomical structures such as gray or white brain mass, pathological abnormalities or bones from the three-dimensional MRI data sets. These anatomical structures are described by various features, including texture, volume, diameter, centroids, aspect ratios, symmetry factors. Such neural networks have become known primarily through the classification of everyday objects, faces or animals. The CNN's ability of visual perception can profoundly enhance a doctor's ability to analyse medical images more accurately and objectively. Therefore, clinicians would be able to

adapt a patient's medical treatment according to the newly gained insights. The classical approach to pattern recognition in medical imaging is still to design and extract hand-made features dependent on the proficiency of the medical practitioner. Although the clinical relevancy of the conventional approach to image interpretation is indisputable, it is prone to inter-reader variability and in addition, time-consuming due to its complexity. According to the report by the Institute of Medicine at the National Academies of Science (Ball et al., 2015), diagnostic errors contribute to around 10% of patient deaths and account for 6 to 17% of hospital complications. Although the number does not solely comprise man-made mistakes, automated systems are expected to improve reliability in tasks that are particularly error prone. It is for these promising reasons, recent years have seen a surge of interest in the application of deep learning approaches to radiomics (Tajbakhsh et al., 2016).

1.2 Aim of the Thesis

In this master thesis, machine learning approaches based on convolutional neural network will be assessed for the analysis of pre-operative magnetic resonance imaging scans of patients diagnosed with Glioblastoma Multiforme. The work will focus on three main aims for the better understanding of the aggressive tumor associated with extremely poor survival:

i The detection and segmentation of the tumorous tissue in MRI scans taken a few days before resection surgery,

ii Predicting the overall survival from pre-operative MRI scans and additional clinical features from patients with glioblastoma

iii Assessing the potential of improving prediction accuracy by simplifying the problem to a classification task between short-term (< 24 months) and long-term (≥ 24 months) overall survival at the time of the pre-operative MRI scanning.

1.3 Structure of the Thesis

The structure of this work is described in the following. Chapter 2 describes the background of machine learning to give the reader an overview of the relevant

concepts necessary for the understanding of the subsequent chapters. The underlying idea of machine learning will be thoroughly discussed, and an essential instrument of machine learning applications will be presented namely artificial neural networks. In Chapter 3, we will introduce the basic components of a convolutional neural network and underline the contrast to conventional neural nets. After the fundamental concepts of the methodology used in this work have been presented, we will focus on the medical motivation behind this work in Chapter 4 to highlight the importance of machine learning for radiomics and oncology. Chapter 5 will then present our contribution to solve the brain tumor segmentation problem as well as our approach to the overall survival prediction of patients with glioblastoma multiforme using convolutional neural networks. The cohort for the experiment was selected through the population-based Austrian Brain Tumor Registry and reflects routine clinical practice for glioblastoma at eight contributing medical centers across Austria. At last, we will discuss the results obtained in this thesis and the potential for future research in Chapter 6.

Chapter 2

Relevant State-of-the-Art in Machine Learning

Artificial Intelligence (AI) is not a new concept of computer science but has gained a lot of attention in the past decade. The area of AI concerns itself with the creation of intelligent machines able to learn in a human-like manner. Machine learning is a subfield of AI that focuses on learning from large quantities of observations to solve a given task such as the prediction of a label associated with an instance of the data. In this chapter, an introduction to machine learning and the fundamental concepts behind this uprising intersection of statistics and computer science will be presented in Section 2.1. One approach to machine learning are artificial neural networks, which are a simple approach to approximate human-like learning and will be subject of Section 2.2.

2.1 The Fundamentals of Machine Learning

Although machine learning is a known concept since the 1950s with Alan Turing famously proposing the first learning machine and posing the question "Can machines think?", it has progressively gained more importance in the past decades due to profound technical advances such as more powerful GPUs and the increasing availability of large volumes of data from the internet. In its basic form, machine learning is a statistical data analysis technique in which computers acquire the ability to automatically learn and improve from experience without human intervention or assistance. It generally falls in the intersection of computer science and statistics. The fundamental theoretical concepts enabling machine learning

are derived from probability theory and statistics such as the Bayes Theorem (**?**) and the Least Squares method (Legendre, 1805). Nonetheless, the primary objective is to develop machines that are able to learn, thus machine learning is usually considered an area of computer science.

In contrast to traditional programming, in which hard-coded rules are executed step-by-step, machine learning can carry out complex processes by directly learning from examples and improving without human supervision or assistance. The system independently determines a set of rules given a specific task and well-defined examples in the form of data rather than following predefined hard-coded guidelines. To put it briefly, machine learning algorithms search for general predictive patterns within given input data to accomplish a desired task. A considerable benefit of machine learning is the fact that it can be applied in situations, where humans cannot encode the data due to its complexity or quantity making it a desirable data analysis tool in areas like health care and medical research, marketing and sales and data security. Machine learning contains the attractive ability to capture hidden relationships, detect patterns and correlations within the data. However, this raises the crucial question of how a machine learning model actually teaches itself. Mitchell (1997) provide a thorough definition of what is meant by learning:

Definition 1. *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

It is the most widely used definition of machine learning and essentially means that when a computer program improves in performance with given experience, we can speak of machine learning. Further, the formulation by Mitchell (1997) can be used to simplify a complex problem to its key parts by defining the available and recorded data at hand (E), the task that the ML-algorithm should be able to solve (T) and how the results should be evaluated (P). Thus, the concept of machine learning is easily understandable.

2.1.1 Types of Machine Learning Algorithms

In general, machine learning algorithms experience a fixed data set consisting of a collection of examples, which in turn are composed of a collection of fea-

tures (Goodfellow et al., 2016). Most implementations of machine learning can be broadly divided into four groups according to their type of experience: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. In this context the concept of supervision does not refer to the algorithm being supervised or assisted by humans, but whether or not the algorithm receives labelled or unlabelled input.

Supervised learning

Supervised systems target well-defined problems, where a full set of data labelled according to its ground truth is available. These labels categorize each sample of the data set in one or more classes and are the desired outcome for the model to predict through the process of learning. In order to determine features indicating a certain outcome class, supervised models adapt themselves to reproduce outputs known from a training set and predict the classes of unseen test data. Thus, supervised learning is mainly applied when the desired task for the machine learning algorithm is regression or classification.

Unsupervised learning

In unsupervised learning, no desired outcome or ground truth is handed to the machine learning algorithm. The system is trained with unlabelled data and no instructions. The unsupervised model attempts to determine a data structure by extracting useful features and analysing patterns. Unsupervised learning is not widely used but proves helpful in uncovering hidden data structures and can be used as a preliminary step before supervised learning. The main tasks for unsupervised learning are clustering and dimensionality reduction.

Semi-supervised learning

While the previous mentioned approaches of machine learning are either dealing with labelled or unlabelled data, semi-supervised learning algorithms are trained on a combination of labelled and unlabelled sets of data. This approach proves beneficial in data settings, where the process of data labelling by experts is often expensive, complex and time-consuming and hence, small amounts of labelled but large collections of unlabelled data are available as often seen in medical applications. Semi-supervised learning uses the unsupervised learning approach in order to improve the supervised learning process.

Reinforcement learning

In a typical reinforcement learning setting, an agent or computer program learns to interact with its environment by rewarding actions towards the goal and thus, iteratively optimizing the behaviour to increase the payoff. An agent in reinforcement learning is trained through past experiences and exploration of new strategies, which offer an improved prospect of success. The main objective is for the agent to learn the implications of its actions. In contrast to the aforementioned approaches, the learning algorithm does not experience a fixed dataset, but rather trains through a feedback loop between the learning system and its environment.

2.1.2 Evaluating Machine Learning Models

The final crucial component of a machine learning algorithm is a quantitative measure of its performance in carrying out the desired task. The performance is usually evaluated on the unseen test data mirroring real world data to obtain an unbiased and true estimate. Depending on the requested task, the performed measures of interest can vary. In the case of a classification task, the accuracy of the model is measured as a reliable estimate of the model's performance. Accuracy is defined as the proportion of instances for which the model produces the correct output class (Goodfellow et al., 2016).

However, the main requirement for a machine learning algorithm is to perform well on unseen data. The contrast in performance of the algorithm between the training data and when applied to unseen test data is referred to as the generalization error. Generalization refers to the ability of a trained algorithm to apply the learned concepts to new examples and perform well. The two main reasons for poor performance in machine learning is generally either overfitting or underfitting the data. Overfitting can be observed when the algorithm performs well on the training data, but the test error is high. In this case, the algorithm has learned concepts of the training data which do not apply to the test data. For example, the model can pick up noise and random fluctuations in the training set, which are not helpful in the test data. The model is said to not generalize to new data. Methods to prevent overfitting will be discussed in Section 2.2.5.

Underfitting occurs when the algorithm neither learns predictive patterns in the training set nor in the test data. This issue can be caused by either training

with a too shallow network or when the desired task at hand is not solvable with the given data. Underfitting is generally easy to spot with proper model metrics and can be resolved by trying different machine learning algorithms or network architectures.

2.2 Artificial Neural Networks

Artificial neural networks (ANNs) are brain-inspired mathematical systems that have an extensive range of application in industry and academia. Their design composed of multiple nodes is intended to simulate human learning through the linkage of biological neurons. ANNs constitute nowadays the fundamental foundation of artificial intelligence (AI), which comprises a large variety of nonlinear statistical models and learning methods. A special extension of neural networks for visual pattern recognition called convolutional neural networks (CNN) is the primary focus of this thesis and will be discussed in the subsequent section.

2.2.1 The Principles of Neural Networks

Biological neural nets are complex interconnected systems consisting of nerve cells (neurons), which communicate concurrently through electrical triggers passing through the neuron's axon. Since a neuron can possess up to several hundred synaptic junctions, multiple nerve cells can be interconnected. However, these links are not equally weighted but rather according to their priority (Graupe, 2013).

Artificial neural networks (ANNs) are computational systems reminiscent of neurons and their beneficial characteristics of node parallelism and weighted node connections. The development of ANNs, or simply neural networks (NNs), dates back to the 1940s with McCulloch and Pitts (1943) being one of the first to use a model topology inspired by biological neural networks. However, early NNs did not have the ability to learn. The historically earliest single-layer ANN known as 'The Perceptron' was proposed fifteen years later by the psychologist Rosenblatt (1958) and poses even nowadays the fundamental building block of nearly all subsequent ANNs.

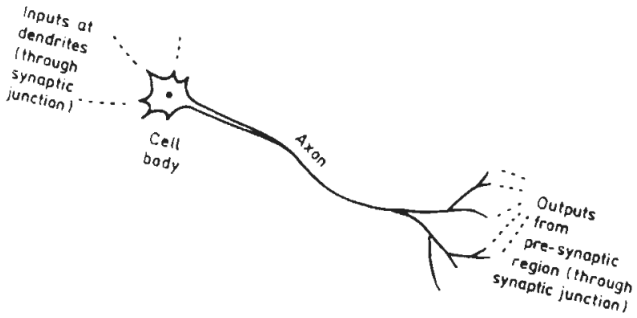

Figure 2.1: Illustration of a biological neuron. Image with permission from Graupe (2013)

The architecture of the artificial neuron follows a simple input-output principle:

$$y = \phi\left(\sum_{i=0}^{n} w_i x_i + b\right) \tag{2.1}$$

where w_i ($i \in \{0, 1, ..., n\}$) denotes the weight of the respective input x_i ($i \in \{1, ..., n\}$), x_0 the bias input of the artificial neuron with a value equal to 1 and z the resulting node output. The weights are learned during the training process. After the linear combination of inputs, weights and bias has been computed, the resulting output is processed by the non-linear activation function ϕ as illustrated in Figure 2.2.

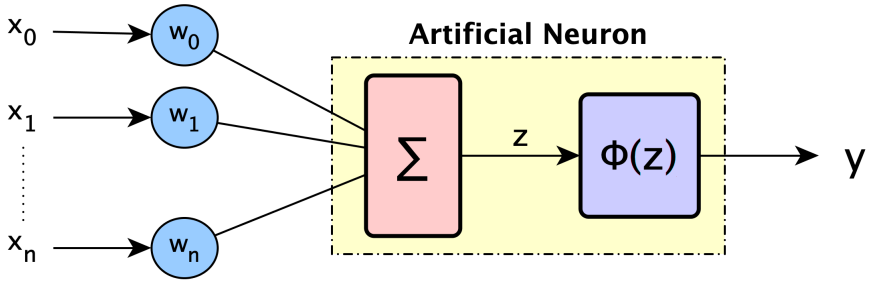

Figure 2.2: A perceptron or artificial neuron with n inputs and one output

The activation function ϕ is also known as the squashing or transfer function, since it maps the node's output to a certain value range. There are various activation functions with different properties, but one of the simplest functions is given by the Heavyside function, which is a threshold-based function resulting

in a binary output:

$$\phi = \begin{cases} 1, & \text{if} z_i \geq 0 \\ 0, & \text{if} z_i < 0 \end{cases} \tag{2.2}$$

This activation function activates the output, if the threshold is reached. Since the output $y = \phi(z)$ is either equal to 0 or 1, the step function is mainly suitable for binary classification problems.

Despite the pioneering approach, the restricted applicability of single-layer neural nets was soon pointed out (Minsky and Papert, 1969). The single-layer architecture of the early NN allows only for a small fraction of classes of patterns to be recognized. For instance, if we consider a neuron with a binary input and n variables, the neuron can have 2^n different input patterns. If we know consider the 2 different binary outputs of the neuron for each input pattern, then there can be 2^{2^n} different functions of n variables. However, the number of linearly solvable problems of these 2^{2^n} possibilities is only a small fraction (Graupe, 2013). Single-layer neural nets can only solve classification problems of data points that lie in a convex open or convex closed region as shown by Minsky and Papert (1969). Continuous further development of NNs to overcome these limitations led to the multilayer extension of neural networks.

2.2.2 Multi-Layer Neural Networks

In principle, neural networks can generally be viewed as a black box model mapping a set of inputs to a desired output without any knowledge of its internal workings. In contrast to traditional statistician approaches for prediction and classification such as regression or decision tree models, which usually can be easily interpreted, the deeper inner workings of neural networks are largely shielded from human understanding. Although, one cannot directly determine how a neural network derived at its final decision due to its high-dimensionality and complex connections, one can follow the reasoning of the machine through the understanding of its building blocks.

Multi-layer neural nets consist of a multitude of artificial neurons arranged in multiple layers, which are fully connected i.e. nodes from adjacent layers are connected. The layers of a multilayer network can generally be categorized into

three types: the input layer, the hidden layer(s) and the output layer. The first layer or input layer receives the input and usually passes the unaltered input to the next layer. The hidden layers lie in between the input and output layer and mainly take in a set of weighted inputs to perform the computations through activation functions. They have no explicit connection to the input or the output and are hence referred to as hidden. The last layer is called the output layer and transfers the information of the hidden layers to the outside world. The depth of a neural network describes the total number of layers and the width of a single layer describes its total number of units.

A neural network is called a feed-forward network, if it does not contain feedback loops within its network topology. This means information is always fed-forward in the model architecture in a sense that the output of one layer serves as the input of the next layer. Neural networks including feedback loops are referred to as recurrent neural networks, but do not lie within the scope of this work and will therefore not considered in the following section.

In general, a feed-forward neural network can be described via graph theory as a directed acyclic graph $G = (V, E)$ with a weight function $w : E \to \mathbb{R}$, where V denotes the set of nodes or neurons and E refers to the set of edges connecting the nodes as stated in Shalev-Shwartz and Ben-David (2014). The neurons are arranged in multiple layers, hence we can write $v_{j,t}$ for the j^{th} neuron of the t^{th} layer. We refer to T as depth of the neural network, which simply is the number of layers in the network. The total set of nodes can be decomposed into a union of non-empty and disjoint subsets, $V = \bigcup_{t=0}^{T} V_t$, in a way that every edge $e_{i,t-1\to t}(\forall i \in I)$ in the set E connects some node in V_{t-1} to some node in V_t, for some $t \in T$. In short, only nodes of adjacent layers are connected. The layer $V_{t=0}$ denotes the input layer, also called bottom layer, whereas $V_{t=T}$ refers to the output layer. All layers in between these two, $V_1, ..., V_{d-1}$, are hidden layers. The width of the neural net is given by $max_{t\in T}|V_t|$. Each edge in E links two neurons in V from separate layers. However, one neuron can be linked to multiple nodes. Hence, the output of one neuron serves as the input of one or more neurons in the next layer. The input of a node is determined as the weighted sum of all outputs of connected nodes in the previous layer.

The bottom layer contains $n + 1$ neurons, since the dimensionality of the input is n. Similar to a single-layer neural network, the first layer of the multilayer neural

net does not alter the input. The last neuron of V_0 denotes the input bias and its output is always equal to 1. Therefore, if we denote the output of the i^{th} neuron of the t^{th} layer by $o_{t,i}(x)$, where x refers to the input vector, we can write the output bias as $o_{0,i}(x) = 1$ with $i = 0$. In Figure 2.3, the index of the biased term is marked with a b. For $i = 1, ..., n$, we obtain $o_{t,i}(x) = x_i$ as the output of the i^{th} neuron of the t^{th} layer.

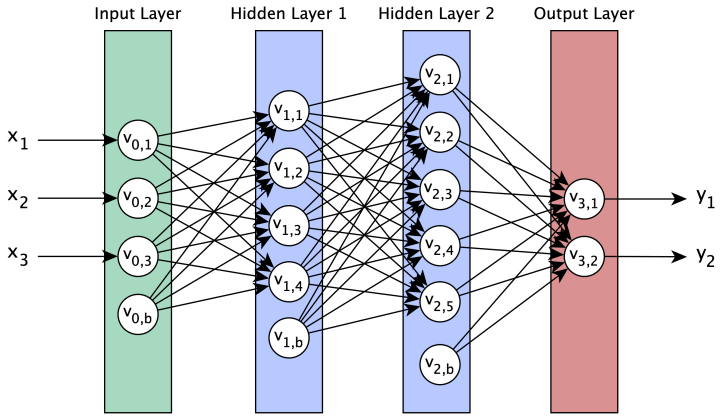

Figure 2.3: A fully-connected multilayer feed-forward neural network with input of dimensionality 3, two hidden layers and a binary output

With the help of these notations, we can now consider the methodological procedure within a neural network. The computations of the layers are conducted in a sequential manner, whereas the neurons within one layer run in parallel. Now suppose the computations of layer t have terminated and consider a fixed $v_{t+1,j} \in V_{t+1}$. Then, we can calculate the input $a_{t+1,j}(x)$ of $v_{t+1,j}$ through the weighted sum of the outputs of the nodes in preceding layer V_t that are connected to neuron $v_{t+1,j}$ in layer V_{t+1}:

$$a_{t+1,j}(x) = \sum_{r:(v_{t,r},v_{t+1,j})\in E} w((v_{t,r}, v_{t+1,j}))o_{t,r}(x) \tag{2.3}$$

where w denotes the respective weighting, which is adjusted during training of the net. The output of node $v_{t+1,j}$ is simply the activation function ϕ applied to the input $a_{t+1,j}(x)$:

$$o_{t+1,j}(x) = \phi(a_{t+1,j}(x)) \tag{2.4}$$

2.2.3 The Activation Function

The activation function $\phi(z)$ is an essential element of a neural network, which enables the network to solve non-linear problems. As previously stated, a neuron is a simple scalar activation function $\phi : \mathbb{R} \to \mathbb{R}$ such as the Heavyside step function in Section 2.2.1.

$$\phi(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{2.5}$$

In the beginnings of neural networks, the most commonly used activation function was the continuously differentiable sigmoid or logistic function, which can be expressed by:

$$\phi(x) = \frac{1}{1 + e^{(-x)}} \tag{2.6}$$

The sigmoid function squashes the range from $(-\infty, \infty)$ to $(0, 1)$, which is why it is especially useful when predicting the probability as an output.

Another approach to the sigmoid activation function is the tanh function.

$$\phi(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.7}$$

The tanh activation function is the scaled alternative of the sigmoid function with a range between -1 to 1, which simplifies optimization.

A recent alternative activation function to the sigmoid function in neural networks has become the Rectified Linear Units function or in abbreviated form ReLU and is nowadays the default activation function recommended for most feed-forward neural networks (Goodfellow et al., 2016). The ReLU activation function is defined as:

$$\phi(x) = max(0, x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{2.8}$$

The activation function offers numerous benefits compared to the previously stated functions. It poses the beneficial feature of faster computation since no exponential computation is needed. In addition, they preserve a proper number

of desirable traits from linear functions due to their almost linearity. Another major advantage compared to the previously stated approaches is that ReLU diminishes the vanishing gradient effect.

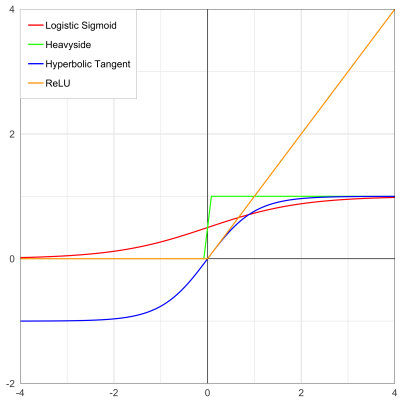

Figure 2.4: Commonly used activation functions in neural networks

Since the main purpose of the activation function is to introduce non-linearities to the neural network, the right choice of the activation function can have a significant effect on the training procedure and is in general tied to the desired task. The right choice of activation function will approximate the function faster and hence reduce the training time. In case of a classification task, the sigmoid function is a common pick for the activation function of the output layer, because it aims at predicting the probability of a binary output (0 or 1). For multi-class classification tasks, the activation function softmax is used since it calculates the probabilities of each output class over all possible output classes. The probability range is then squeezed so that the sum of the probabilities equals one. The predicted class will have the highest probability. If the desired task is regression, a linear activation function will be used in the output layer.

In the early stages of neural networks, the most used activation function for the hidden layers was the sigmoid function. However, this led to the vanishing gradient problem: the gradient of the sigmoid functions can be very small and create computational problems. This behaviour caused a shift toward the Rectified Linear Unit function which is 0 if the input is negative and otherwise follows a linear trend. It has been shown that the ReLU function does not suffer from the vanishing gradient problem. On the other hand, the downside of ReLU is the 'zero dying' problem caused by setting the negative input to zero. The neuron stuck on

the negative side will be considered dead and is not likely to recover. A solution to this problem is posed by the Exponential Linear Unit (ELU) (Clevert et al., 2015), which only differs from ReLU in the way ELU deals with negative inputs. In contrast to ReLU, ELU is not exactly zero for negative values, but it has a small slope for them. ELU is defined as

$$\phi(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{if } x < 0 \end{cases} \tag{2.9}$$

where α is a hyperparameter that controls the value to which the function saturates for negative values.

All in all, the choice of activation function mainly depends on the available data and the task the machine learning algorithm should be able to perform. The universal approximation theorem states that a feedforward network with a single hidden layer consisting of a finite number of neurons is sufficient to approximate any continuous function (Hornik, 1991). So as Hornik (1991) showed, it is not the specific choice of the activation function, but rather the multilayer feedforward architecture itself, which enables the neural network to be a universal approximator. In practice, however, the general standard for the hidden layers is ReLU and in case of dead neurons, other options such as ELU are available.

2.2.4 Training a neural network

Now that we have mathematically defined the architecture of a feed-forward neural network by $G = (V, E, \phi, w)$, we will focus on the most important component of a model, the training process. The training process of a neural network generally consists of four main steps, which are iteratively repeated until sufficient network performance is achieved:

1. Weight and bias initialization
2. Forward propagation
3. Computation of the loss function
4. Backward propagation

A neural network is trained by iteratively adjusting the weights and biases of all neurons such that the network learns to accurately predict the target outputs from known inputs. Let again $V_1, ..., V_T$ be the layers of the feed-forward neural

network G with $V_t = \{v_{t,1}, ..., v_{t,k_1}\}$ and a network depth of T. The initial network parameters $w^{(0)} \in \mathbb{R}^{|E|}$ are commonly set at random but from a distribution such that the weights are close to 0.

After initialization, each optimization cycle mainly consists of three subsequent steps: first forward-propagation, computation of the loss function and then, backward-propagation. These steps are iteratively conducted until acceptable performance is reached. When the entire data set has passed forward and backward through the neural network only once, we speak of one epoch. For training purposes, up to a thousand epochs can be required for the network to achieve sufficient performance depending on the complexity of the task. Since one epoch generally requires too much memory to be fed to the computer at once, the data is divided into small subsets called batches. So, in order to finish one epoch, multiple iterations over the different batches are needed. For example, if one has an input data set of 1000 examples and divides the data into batches of 100 examples, then it would take 10 iterations to finish one epoch.

The first step of training, forward-propagation, occurs when the initial information sequentially passes through each layer so that each neuron can apply its transformation function to the input it receives from the previous layer. In the last layer, the respective predicted labels of the target output for each sample of the training set are computed.

Next, we compute the loss function, which is an important part in artificial neural networks. The loss function, also known as cost or error, of the network indicates the predictive accuracy of the network by measuring the inconsistency between the predicted outcome $\hat{y}$ and the true value y. The loss function $\mathcal{L}(\cdot)$ is a core component of the structural risk function of the neural network, which can be written down as

$$\theta^* = arg\min_{\theta} \mathcal{L}(\theta) + \lambda\omega(\theta) \tag{2.10}$$

$$= arg\min_{\theta} \frac{1}{n}\sum_{i=1}^{n} L(y^{(i)}, \phi(x^{(i)}, \theta)) + \lambda\omega(\theta) \tag{2.11}$$

where L denotes the mathematical realization of how the difference between the observed and predicted outcome is computed, θ the parameters of the model that have to be learned (e.g. the weights vectors w and bias terms) and $\omega(\cdot)$

represents the regularization term with the respective penalty parameter λ. The regularization term adds a slight modification to the neural network in order for it to generalize better. Different techniques commonly used will be discussed in Section 2.2.5. The activation function is represented by $\phi(\cdot)$, which is applied on the training sample $(x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)}) \in \mathbb{R}^m$.

The best fitting set of parameters θ for a certain task can be obtained through the minimization of an appropriate loss function for that task. Therefore, the training of a neural network can be viewed as a problem of global optimization, which is commonly solved through Stochastic Gradient Descent (SGD) (Gu et al., 2018). Ideally, the loss would equal zero. From Equation (2.11), we know the empirical risk term is given by

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(y^{(i)}, \phi(x^{(i)}, \theta)) \ . \tag{2.12}$$

The mathematical expression of the loss function mainly depends on the desired task the algorithm should be able to perform. The main performance measure for regression tasks is the mean-squared error (MSE), which is defined as the average squared difference between the estimated values $\hat{y}$ and true values y.

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \tag{2.13}$$

The target of the neural networks is to carefully select the weighted connections in a way that minimizes the quadratic sum of the residuals. A drawback of the MSE is the great influence of larger errors, which makes the loss function sensitive to outliers. A solution is presented by the mean absolute error (MAE). The MAE is more robust to outliers, because it considers the absolute difference between the estimated value $\hat{y}$ and the true value y.

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}| \tag{2.14}$$

For classification problems, the cost function commonly used is the categorical cross-entropy defined as

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} y_j^{(i)} \ log \ \hat{y}_j^{(i)} \ . \tag{2.15}$$

It represents the distance between the probability distributions of $\hat{y}^{(i)}$ and $y^{(i)}$ summed over all of the class predictions in the y vector. For binary classifications, the formula reduces to

$$\mathcal{L}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y^{(i)}\ log\ \hat{y}^{(i)} + (1 - y^{(i)})\ log\ (1 - \hat{y}^{(i)})\ . \tag{2.16}$$

Back-propagation is the last step of the iterative learning process. The back-propagation algorithm is the standard method for training neural networks. After the evaluation of the loss function, the computed information is propagated backwards into the model to compute the gradient of the cost function and then, uses the first derivative of the loss function to update the parameters of the network. Therefore, the parameters θ of the loss function $\mathcal{L}(\theta)$ can be obtained by

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta E[\mathcal{L}(\theta_t)] \tag{2.17}$$

where η represents the step size or learning rate for each iteration.

In contrast to estimating the expectation $E[\mathcal{L}(\theta_t)]$ of $\mathcal{L}(\theta_t)$ over the entire available training sample, Stochastic Gradient Descent (SGD) computes the gradients of the loss function on the basis of a single randomly picked sample $(x^{(t)}, y^{(t)})$ of the training set to update the parameters θ. Thus, the SGD algorithm can be expressed by

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta \mathcal{L}(\theta_t; (x^{(t)}, y^{(t)}))\ . \tag{2.18}$$

However, in practice, each parameter will be performed on the basis of a mini-batch rather than a single example from the training data in order reduce variance in the update and obtain more stable results (Gu et al., 2018). The intuitive idea behind gradient descent is the following: if the change of a weight leads to an increase or decrease of the error function, then we want to decrease or increase the respective weight in order to minimize the error. This can be mathematically described by the change in loss given a unit of change in the weight.

We obtain the following minimization problem: Find the network weights to minimize the training error between true and estimated labels of training examples and update the weights by gradient descent.

2.2.5 Regularization

2.2.5.1 Weight Initialization

The random initiation of the neural network weights can potentially cause two commonly experienced issues namely the vanishing or the exploding gradient problem. Therefore, the weight parameters should be chosen carefully. In order to avoid the vanishing or exploding gradient problem, a proper network initialization is one of the most important prerequisites (Zhang et al., 2019). In practice, the weights are commonly initialized to random small numbers that have zero mean.

2.2.5.2 L1- and L2- regularization

In order to control for overfitting and improve the generalization error of a neural network, regularization practices are frequently used in application. The most common approach involves the inclusion of a penalty factor $\lambda > 0$ in the loss function affecting the behaviour of the weight matrix. The hyperparameter λ represents the strength of regularization with larger values of λ affecting the steps of the gradient descent optimization algorithm greater than smaller penalties. Two well-known versions of this approach are L1- and L2-regularization.

L1-regularization is also known as method of least absolute deviation. The L1-regularized loss function is given by

$$\hat{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \lambda|\theta| \ . \tag{2.19}$$

L1-regularization will shrink values within the weight matrix belonging to less important features towards zero. Therefore, regularization by the L1-norm will lead to sparse weight vectors during optimization. As a result, it is preferable in situations in which the data inputs are expected to be very noisy. In addition, a useful property of L1-regularization is the built-in feature selection due to the sparsity of the parameter vector (Ng, 2004). $L1$-regularization tends to be more robust against outliers, whereas $L2$-regularization will treat noisy input more thoroughly and thus, result in larger error terms for these data points making it more sensitive to outliers.

$L2$-regularization, also known as least squares method, is a regularization technique, which penalizes larger weights stricter than $L1$-regularization due to λ controlling the squared values of the weights. The loss function $\mathcal{L}(\theta)$ is modified

through an additional penalty term $\omega(\theta) = \lambda\frac{1}{2}\|\theta\|^2$. Thus, we obtain for our new loss function $\hat{\mathcal{L}}(\theta)$

$$\hat{\mathcal{L}}(\theta) = \mathcal{L}(\theta) + \lambda\frac{1}{2}\|\theta\|^2 \ . \tag{2.20}$$

The least squares method squares the error in contrast to $L1$-regularization and thus is more sensitive to outliers. However, it is also possible to combine both $L1$- and $L2$-regularization, which is commonly known as elastic net regularization (Zou and Hastie, 2005).

2.2.5.3 Dropout

Another common approach to regularization is presented by introducing a dropout layer to the neural network, which was first described by Hinton et al. (2012). The idea behind dropout is to assign each neuron in a layer the probability p of being deactivated during computation. The probability p is commonly referred to as the drop rate. In each iteration of the neural network computation, neurons are randomly selected and turned off according to the drop rate. As a result, the output y of a dropout layer is given by

$$y = r\phi(W^T x) \tag{2.21}$$

where $x = (x_1, x_2, ..., x_n)^T$ represents the input to the fully connected layer, $W \in \mathbb{R}^{nxd}$ a weight matrix and $r = (r_1, r_2, ..., r_d)$ denotes a binary vector with $r_i \sim Bernoulli(p)$ (Gu et al., 2018). Although this approach seems counterintuitive at first due to the deactivation of trained units, dropout mainly reinforces each individual neuron to independently learn relevant features without relying on other units to correct for its mistakes or be responsible for the correct classification or prediction. Therefore, the random deactivation ensures a more evenly distributed weight matrix and leads the model to better generalization (Hinton et al., 2012). Network dependence on a small number of units is therefore counteracted.

2.2.5.4 Early Stopping

A further possibility to avoid overtraining a deep neural network is presented by Early Stopping, which was introduced by Srivastava et al. (2014) together with the concept of Dropout (see Section 2.2.5.3). As the name suggests, the approach

tries to combat overfitting of the training data by stopping the training procedure of the network early. Too much training as well as too little training can lead to poor performance when the neural network is applied to the test data. In the case of too little training, we speak of underfitting the data. Early Stopping will intervene in the training process at the time point, when the performance on the validation data is deteriorating and thus, prevent the overtraining of the deep neural network model.

Chapter 3

Convolutional Neural Networks

In this chapter, we will present a unique type of multi-layer neural networks used for computer vision and image processing tasks namely convolutional neural networks (CNNs). An in-depth analysis of the underlying concepts and building units of the standard convolutional network architecture will be given in this chapter. The distinctive layers associated with CNNs will be thoroughly explained to illustrate the contrast to traditional neural nets. In addition, an overview of optimization techniques for CNNs will be given and lastly, CNN architectures will be described that are widely known in the deep learning community.

3.1 Introduction

The class of convolutional neural networks (CNNs) forms an extension to that of artificial neural networks and has been used in visual pattern recognition since the late 1980s. The first successful convolutional neural network was introduced by LeCun et al. (1989), who applied the backpropagation algorithm to CNNs in order to solve the problem of recognizing handwritten numbers. The work of LeCun et al. (1998) was one of the first milestones in the history of deep learning. In the following years, LeCun et al. extended the architecture to perform face detection (Vaillant et al., 1994) and document recognition (LeCun et al., 1998). However, the application and extension of deep learning algorithms to large scale and high-resolution images was severely limited in the 90s by the not yet developed computational power.

Convolutional neural networks are multi-layer supervised networks inspired by how the brain processes visual information, which can learn features automatically from input images and take the spatial dependence of the imaging data into account. CNNs are powerful deep learning tools and have been successfully used for both visual object recognition and image classification. Despite the numerous variants of CNN architectures found in the literature the main building blocks of these networks remain the same namely convolutional, pooling, and fully-connected layers. A schematic representation of a convolutional neural network can be seen in Figure 3.1. CNNs and ordinary neural networks differ in various aspects with the most obvious being the addition of a convolutional layer to the network layer set-up, which will be explained in detail in Section 3.2.1.

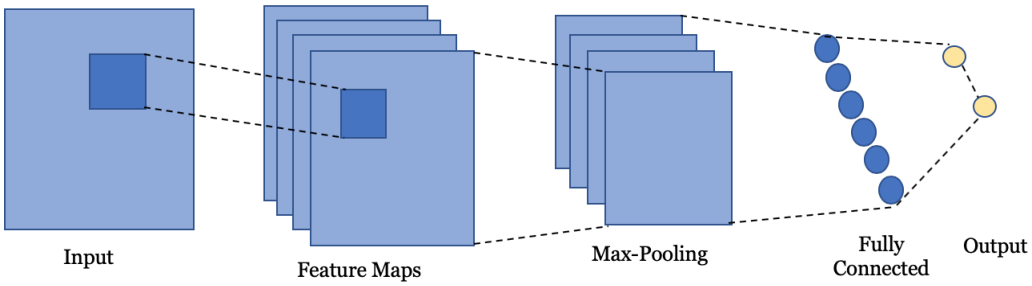

Figure 3.1: A simple convolutional neural network architecture consisting of one convolutional layer, max-pooling and a fully-connected layer at the end for binary classification

Even though CNNs are an extension of artificial neural networks, there exist numerous differences between the two network architectures. One of the essential differences lies in the input it requires namely imaging data. Essentially, images can be represented as a matrix of pixel values. Standard coloured images are in RGB format, which denotes the red, green and blue channel of the image. Thus, RGB images consist of three channels that can be represented by three pixel matrices stacked together, whereas greyscale images only have one channel. In general, the value of each pixel in the matrix of a greyscale image will range from 0 to 255 where zero indicates the colour black and 255 indicates white.

The three main benefits of CNNS in comparison with ANNs are the 3D-arrangement of the units, weight sharing and local connectivity. In contrast to ANNs, neurons

in CNNs are three dimensionally arranged since the image input is represented by a 3D matrix with a dimension for width, height and depth. Depth corresponds to the colour channels of the image in RGB format. The parameters of each convolutional layer consist of a set of learnable filters. Each of these filters spatially covers a predefined subregion of the input in regard to its height and width but extends through the full depth of the input volume.

The feature of local connectivity is inspired by the receptive fields in the visual cortex of cats, which contain neurons that solely receive input from a small field of the visual field (Hubel and Wiesel, 1959). Similarly, each neuron in a convolutional neural network is only connected to a small subset of input neurons. In contrast, each input value in a traditional neural network is connected to every neuron within a layer. However, when dealing with high-dimensional image data, connecting all neurons of two consecutive layers would require a high computational effort. Therefore, convolutional networks only receive information from a small group of neighbouring neurons from the previous layer. The property of local connectivity (also referred to as sparse connectivity, sparse weights or sparse interaction) allows CNNs to consider spatial dependencies and thus, spatially exploit the local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. This concept helps to reduce the total number of parameters in the network and hence, makes it more statistical efficient (Goodfellow et al., 2016).

Parameter sharing significantly reduces the number of parameters the system has to learn during training by imposing the useful assumption that neurons in each depth can be constrained to the same weights and bias. In an ordinary neural net, each element of the weight matrix will be used exactly once for the computation of the output layer. In convolutional neural networks, the set of parameters learned for one location will be used at another location (Goodfellow et al., 2016). Filter weights are shared across all receptive fields. This important property of CNNs leads to a neural architecture that relies on far fewer parameters than traditional neural nets. In addition, CNNs possess a property called equivariance to translation due to the parameter-sharing characteristic. A function $f(\cdot)$ is said to be equivariant to function g, if $f(g(x)) = g(f(x))$. In the context of CNNs and translation, this means that a shift of input features will result in an equivalent shift of outputs.

In the following sections, the underlying concepts and conventional building blocks of convolutional neural networks will be further introduced. However, it is important to note that this chapter does not contain a comprehensive review of the entire methodology and innovative advances made in this research field but rather current approaches relevant to the application in Chapter 5. For more detailed expositions on new advances of convolutional neural networks see Zhang et al. (2019).

3.2 Network Architecture

CNN architectures can vary widely in the number and type of layers used in the convolutional network set-up largely dependent on the specific task the network should be able to perform. If continuous responses are the outcome of interest, the CNN should include a regression layer after the fully-connected layers at the end of the network, whereas for categorical responses the regression layer has to be exchanged for a classification function.

3.2.1 Convolution Layer

Mathematically speaking, a convolution is an operation, which takes the integral of the product of two functions after shifting and reversing one of them. It is usually denoted with an asterisk as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{3.1}$$

However, the operation used in a convolutional neural network does not correspond precisely to the definition of convolution as used in other fields, such as engineering or pure mathematics (Goodfellow et al., 2016). The term convolution stems from the convolution of two signals.

The convolution in CNNs is performed on the input using a filter (also commonly called kernel) in order to produce a feature map. The operation is implemented by sliding the filter window over the input data. The step size of the filter movement is called the stride of the filter. A stride of one will represent a shift of one unit (e.g. pixel) at a time. Hence, the stride controls how the filter convolves on the image data. For every filter position, a matrix multiplication is performed to sum the result onto the feature map. As mentioned previously, the subregion

of the filter is also called the receptive field. The convolutional layer extracts features over all spatial locations from an input image with the use of multiple filters and generates feature maps. In principle, the depth of the linear convolutional filter will equal the depth of the input and the depth of the output (the activation map) is identical to the number of filters applied to the input image. In practice, padding is commonly applied to the input image to preserve the spatial dimension of the input image. A common padding technique is zero padding due to its simplicity and effectiveness. As the name states, zeros are added to the margin of the input image. The three hyperparameters depth, stride and setting zero-padding alter the spatial dimensionality of the convolutional layers output.

The width or height of the output for any given convolutional layer can be obtained through the following formula:

$$h_o = \frac{h_i - h_f + 2p}{s} + 1 \tag{3.2}$$

where h_o, h_i and h_f denotes the height of the output, input and filter, respectively. The parameter s represents the stride of the filter and p the amount of padding added to the input image. The calculation of the width is carried out in the same manner. The depth of the output solely depends on the number of filters applied to the input.

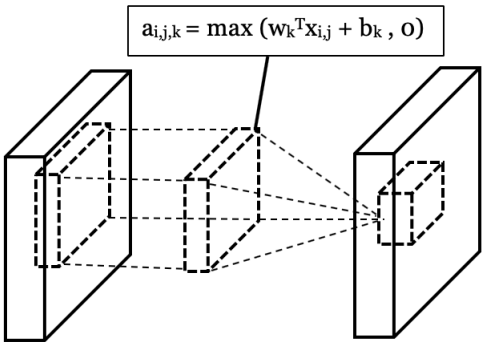

Figure 3.2: The operating principle of a convolutional layer

In mathematical terms, the feature map of a convolutional layer as illustrated in Figure 3.2 can be postulated as

$$a_{i,j,k} = max(w_k^T x_{i,j} + b_k, 0) \tag{3.3}$$

where $a_{i,j,k}$ is the value of k^{th} feature map positioned at (i, j) and $x_{i,j}$ is the input patch centered at (i, j). The weight and bias vector of the k^{th} filter are denoted by w_k and b_k, respectively (Zhang et al., 2019).

Suppose we have a 32x32x3 image input (a coloured RGB image with dimensionality 32x32) and want to apply a 5x5x3 filter. The depth of the filter equals the colour channels of the image due to the 3D-arrangement of the units, hence the last dimension of the filter and the input match. The stride will be set to 1 i.e. the filter will be shifted by one-pixel steps over the entire image. No padding will be applied to the input. The filter will slide over all spatial location of the image and perform element-wise matrix multiplications and summations. According to Equation (3.2), this operation will result in an activation map of size 28x28x1. If now multiple filters are applied, for instance, 4, we will obtain an activation map with the dimension 28x28x4.

A nonlinear layer always follows immediately after a linear convolutional layer to introduce non-linearity to the neural net. A standard activation function is ReLU, which was discussed in Section 2.2.3 and shifts the negative activations to zero. Overall, the standard convolutional layer uses multiple linear filters of a certain height and width followed by a nonlinear activation function to convolve over the input and extract predictive patterns.

3.2.2 Pooling Layer

In addition to the convolutional layer, the pooling layer (also known as downsampling layer) forms another core component of a convolutional neural network and is usually applied after one or more convolutional layers. The pooling layer receives the activation maps of the previous convolutional layer as its input and summarizes each subregion of each individual map into one value by, for example, taking the maximum or the average. The summary functions of the subregions are referred to as maximum-pooling and average-pooling, respectively. Consequently, the pooling operation is a form of dimensionality reduction of the feature maps and hence, often referred to as downsampling. The pooling units are spaced a few pixels apart, so that in total fewer pooling units are obtained than there were convolutional unit outputs in the previous layer (Hinton et al., 2012). As a result, the next layer gets a smaller version of the feature maps. The predefined spacing between each pooling unit within a pooling layer is called the stride. The work-

ing principle behind pooling can be imagined similar to sliding a window over an input image and passing the contents within the window to the pooling function, which summarizes the obtained information. Pooling is fundamental to lower the computational burden of the CNN by reducing the number of connections between the convolutional layers (Gu et al., 2018).

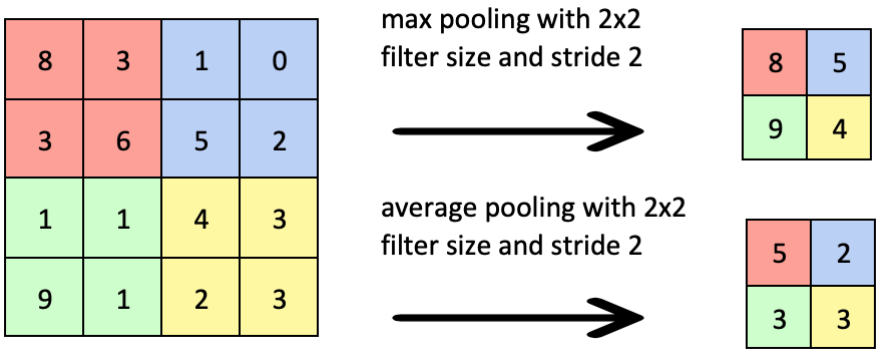

Figure 3.3: Example of a max pooling operation (above) and an average pooling operation (below) of size 2x2 of an 4x4 input image with a stride of 2

The pooling layer neither has weights to learn nor uses an activation function. Solely hyperparameters such as the filter size of the pooling unit and the stride have to be defined beforehand. If the feature map size is not divisible without a remainder, the sliding window of the pooling operation will be cut off at the last row or column. In case important information is believed to be situated at the margins of the feature map, padding should be specified.

3.2.3 Fully-Connected Layer

After the sequence of convolutional and pooling layers have extracted high-level features of the input image, fully connected layers receive these features for classifying the input image into various classes. This is where the operating principles of artificial neural networks and convolutional neural networks collide. While we spoke about hidden layers in the context of ANNs, we will now refer to them in the context of CNNs as fully-connected layers to differentiate between the layers acting as feature detectors and the layers learning non-linear combinations of these features for the classification task. In the fully connected layer, each neuron is connected to all neurons of the previous layer, whereas in the convo-

lutional layer, each neuron is only connected to the neurons of a small region of the previous layer.

For classifying the input image into various classes, the learned feature maps of the last convolutional block are vectorized also referred to flattening and fed into fully connected layers followed by an output layer. Therefore, this structure forms a bridge between the structure of artificial neural networks and its extension, the convolutional network. The convolutional layers can thus be seen as feature extractors, whereas the fully-connected layers take the resulting set of features and performs a given task such as classification (Lin et al., 2013).

3.3 Optimization

This section provides a brief review of different techniques for network optimization. The two most common methods for optimizing a convolutional neural network in case of overfitting are presented by data augmentation and batch normalization. In essence, data augmentation increments the available image data to increase the number of samples, whereas batch normalization tries to correct for the values of the input being shifted while flowing through the network.

3.3.1 Data Augmentation

The predicament most machine learning practitioners' face is the problem of data availability. In principle, one can say the more data, the better the model will perform. However, large data collections of images are hard to obtain and associated with high costs as well as an enormous expenditure of time due to manual annotation in the medical field. Image data augmentation is a machine learning technique that can be used to artificially generate additional images and hence, expand the size of a training dataset by modifying and altering existing images in the training set. The applied geometric transformations should not alter the nature of the data, since you want your sample to remain representative for the vision task. Popular image data augmentation techniques include shift, rotate, flip, contrast, scale, brightness, noise, colour and zooming (Zhao, 2017). These methods reduce the network's capacity to overfit the training data and helps generalization. One drawback of data augmentation is that it requires an understanding of the underlying task in order to perform well and in addition, introduces additional hyperparameters such as the appropriate amount of shifting

or rotation without changing the representability of the training data.

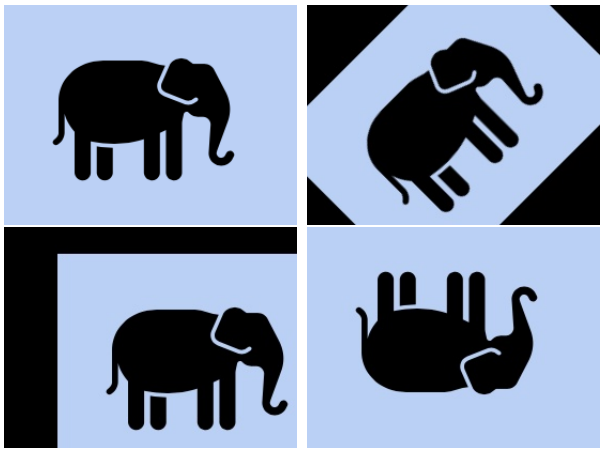

Figure 3.4: Example of image data augmentation obtained through rotation, translation and horizontal flipping

3.3.2 Batch Normalization

In practice, global normalization is usually the first step of data pre-processing in order to shift the data to follow a zero-centered, unit variance distribution. However, as the data sample flows through the convolutional neural network during training, the weights and parameters of the network adjust the data values and hence, the distribution of the inputs of later layers can widely vary, which slows down training since the weights of the later layers have to readjust to the varying inputs. This phenomenon is referred to as internal covariate shift (Ioffe and Szegedy, 2015). Batch normalization can combat this issue by transforming layer outputs into a unit Gaussian distribution. Mean and variance are estimated after each mini-batch rather than the entire training data, which leads to significantly less variation in between batches Gu et al. (2018). During the training process, the d-dimensional input $x = (x_1, x_2, ..., x_d)^T$ of the activation function is transformed to have zero-mean and unit variance. For instance, the k^{th} dimension of the input is transformed by

$$\hat{x_k} = \frac{x_k - \mu_B}{\sigma_B^2 + \epsilon} \tag{3.4}$$

where μ_B and σ_B^2 are the batch mean and variance, respectively, and ϵ denotes a small constant to prevent division by zero. Next, $\hat{x_k}$ is linearly transformed by a pair of learned parameters γ_k and β_k to restore the representation ability of the

network:

$$y_k = \gamma_k \hat{x_k} + \beta_k \ . \tag{3.5}$$

The parameter pair γ_k and β_k scale and shift the normalized values, because solely normalizing the input of a layer might affect what the layer actually represents.

Batch normalization is supposed to have a variety of benefits according to the Google researchers Ioffe and Szegedy (2015) who introduced the technique a few years ago. The inclusion of batch normalization into the neural network will allow higher learning rates to be used, which usually, without batch normalization, would lead to exploding or vanishing gradients during training as well as the algorithm being stuck in poor local minima. In addition, the authors claim that batch normalization can act as a regularization technique such that dropout in networks using batch normalization can either be completely removed or at least its strength reduced according to Ioffe and Szegedy (2015).

3.3.3 Transfer Learning

The training of a CNN involves the search for the right weights for each filter of each layer in order for the network to associated certain input images with its respective label. However, this process requires large amounts of available data depending on the complexity of the task. Training a model from scratch when only limited data is available can be fatal due to overfitting. Thus, one solution to this problem is transfer learning. One can take the pre-trained weights of an already trained model and either use the already learned features to predict new instances or modify the network for a different task. The advantage of pre-training is that it doesn't require as much data and is also associated with much less computational power, since the weights merely have to be adapted and not learned from scratch. There are several models openly available on the internet, which have been pre-trained on the ImageNet data set. The database of ImageNet consists of more than one million images, which were used by multiple models for training classification. Thus, the networks based on the ImageNet database can classify a variety of images into thousand different object categories such as cat, pencil, shirt and hat.

The reason behind why the weights of a model performing a different task can

be used for another classification problem lies in the operating principles of the convolutional layers. In general, the filters of earlier layers of a convolutional neural network tend to get activated by trivial features such as colour, horizontal lines and vertical lines. The deeper we go into the network, the more complex the features the model is learning will be. In deeper layers, the model will be able to recognize textures and shapes (e.g. eyes in a portrait) and finally, the filters in the last layer will be activated by the desired object on the image itself (conditional on sufficient data availability). Transfer learning used the fact that simple features are trained earlier on. Therefore, pre-trained networks can be used by either just adding a few fully-connected layers at the end or even unfreeze the last convolutional blocks if the weights have to be altered substantially.

3.3.4 Global Pooling Layer

In recent years, experts have shifted towards global average pooling (GAP) in order to eliminate overfitting by reducing the parameters a model is required to learn during training. The conventional fully connected layers generally got very large with increasing network size and thus needed a very large number of weights to be learned during the training process. GAP is a method that aims at fully replacing the traditional fully connected layers in CNNs. There are two forms of GAP present in application. The basic idea of the first GAP approach is to transform each feature map of the last convolutional layer into a single number computed as the average of all the values in the feature map. Hence, instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is then fed directly into the output layer. The second approach does not directly pass the flattened feature maps to the output layer, but rather feeds its outputs to one or more fully connected layers beforehand. In simple words, GAP means that if we have a 3D tensor of shape 5x5x10 as the output of the last convolutional layer, we compute the average value over each of the 10 5x5 slices and hence, end up with a 3D tensor of shape 1x1x10 which is then transformed to a 1D vector of shape 10. In the traditional approach of GAP, this vector would then be directly fed to the output layer.

According to the creator of GAP Lin et al. (2013), the advantage of global average pooling over the sequence of fully connected layers on top of the network is presented by its more native approach to the convolution structure by enforcing correspondences between feature maps and categories in addition to the ability of

the global pooling scheme to significantly reduce the overall number of parameters the model has to learn.

3.4 Popular Architectures

3.4.1 U-Net

Before the development of the U-Net architecture, previous convolutional neural networks had shown decent results merely in easy and straightforward segmentation tasks. U-Net was the first convolutional network yielding good performance in complex biomedical image segmentation application (Ronneberger et al., 2015). Segmentation requires a network to partition the visual input into different segments to capture the exact contours of the area of interest in CT and MRI recordings and can largely be simplified to a pixel-wise classification task. However, one challenge is the fact that for a segmentation task, simply classifying a pixel as associated to the desired partition represents merely the first step in this process, since the outcome of interest is the segmented image itself. For the purpose of semantic image segmentation, it is therefore not sufficient only to convert the feature maps into an outcome vector. In addition, an image has to be recreated from the vector. The intuitive concept behind U-Net is the use of the feature maps learned during the classification step to grow a vector back to an image. The pooling operators used in the first step for classification are replaced in the second step by upsampling operators combined with the high-resolution feature maps to expand the output vector to an image.

The schematic shape formed like a 'U', which is depicted in Figure 3.5, illustrates why the original model was named U-Net. The architecture can be separated into two sections: the contracting path (left side) and the expansive path (right side) (Ronneberger et al., 2015). The number of contracting blocks is equal to the number of expansive blocks. The left-hand side of the 'U'-structure represents the architecture of a classical convolutional neural network consisting of multiple blocks made up of convolutional layers followed by a maxpooling operator for downsampling. In each block, two convolutional operators with a filter size of 3x3, no padding and a rectified linear unit (ReLU)-activation function are applied followed by a 2x2 maxpooling layer. This leads to the doubling of the feature maps after each block as denoted by the horizontal numbers above each block in Figure 3.5. The vertical numbers in between the convolutional layers de-

note the dimension reduction of the input after the two convolutional operators, respectively.

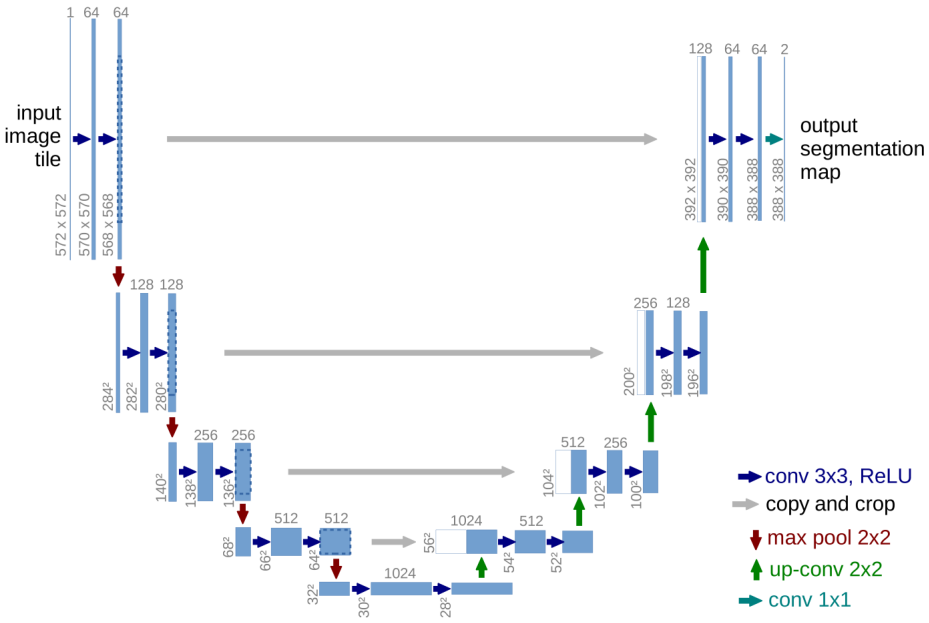

Figure 3.5: Example of the Vanilla U-Net architecture. Image with permission from Ronneberger et al. (2015)

The righthand side of the U-Net architecture constitutes the core of the network. It performs upsampling as well as concatenation to expand the output vector, obtained by the first part of the network, to the segmented image. Each expansion block consists of a 2x2 convolutional upsampling layer followed by two successive 3x3 convolutional layers, which halve the feature channels. After upsampling is performed, the input to the next block in the expansive path is extended by the cropped feature maps of the corresponding contracting block to assist in the reconstruction of the image. The cropping of the feature map is required, because border pixels are lost during convolution. The final layer consists of a 1x1 convolution operator such that each feature vector is mapped to the desired number of classes.

3.4.2 ResNet

Deep convolutional neural networks have led to a series of breakthroughs for visual pattern recognition and image classification in the past decade. Therefore, the urge for increasing network depth for solving more complex tasks and en-

hancing the accuracy of current state-of-the-art networks have led to issues with deep networks composed of multiple layer. As the gradient is back-propagated to earlier layers in the neural network, the gradient can get infinitely small due to repeated multiplications. As a consequence, the deeper we go back in the network, the more saturated the performance can get. This common problem is referred to as the vanishing gradient problem.

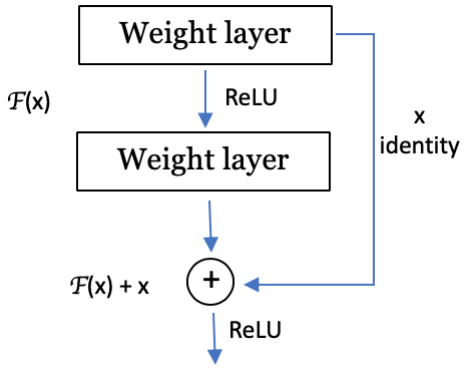

Figure 3.6: Building block of a residual network

The deep residual network called ResNet is a ground-breaking type of convolutional neural network in which the gradients can flow unhindered through shortcut connections to earlier layer (He et al., 2016). Because of this, ResNet architecture can handle deeper and more complex tasks than a regular convolutional neural network without suffering from the vanishing gradient problem. The innovative concept behind residual learning is that no matter how deep a network is, it should not be any worse than the shallower network as the author (He et al., 2016) of ResNet stated. According to the universal approximation theorem, we know that a feedforward network with a single hidden layer is sufficient to approximate any function (Hornik, 1991) as stated in Section 2.2.3. Therefore, a layer should also be able to approximate the identity function. This led to the development of residual blocks, which allow to skip one or more layers, which represents the identity mapping (i.e. input = output) and thus preserve the unaltered inputs of another layer as illustrated in Figure 5.9. ResNet enables the training of CNNs composed of up to hundreds or even thousands of layers and still achieves compelling performance, a crucial property for a network, since the tasks have been getting more complex over time. Although the universal approximation theorem states that one layer should be enough to approximate any continuous function,

implementing a massive layer will usually lead to overfitting, which is why there exists the specialized research area of machine learning namely deep learning, that focuses on how to go even deeper.

Chapter 4

Glioblastoma Multiforme: Imaging and Analysis

This chapter will give an overview of the medical motivation behind this master thesis. The severity of being diagnosed with glioblastoma multiforme due to the poor prognosis is underlined and prognostic factors associated with short- or long-term survival are stated. The importance of magnetic resonance imaging is highlighted, and a broad overview is given to understand the unique characteristics of the data used in the subsequent chapter.

4.1 Introduction

Malignant gliomas present some of the biggest challenges in the field of oncology. In general, the lethal neoplasm can be clinically divided into four grades (I-IV). The most aggressive glioma is grade IV also known as glioblastoma multiforme (GBM) and is interestingly also the most common in humans (Thakkar et al., 2014). Clinical research in glioblastomas has a long history, with hundreds of papers being published about various types of well-designed randomized clinical trials (RCT) to determine the best treatment approach. However, the treatment and diagnosis of patients with high-grade glioblastomas still remains a challenge for modern medical care because of the resistance of the tumor to therapeutic interventions. The heterogeneity of the tumorous tissues makes the neoplasm hard to treat and resect. The complex nature of glioblastomas of grade IV led to the choice of its name: multiforme. Glioblastoma of grade IV are multiforme grossly, multiforme microscopically and multiforme genetically. In essence, this means

that the tumor shows varying types of tumorous tissue, regions of microvascular proliferation and various deletions, amplifications, and point mutations, respectively(Holland, 2000). Despite the understanding of its genetically heterogeneous nature, treatment options effectively targeting all patients are non-existent. Therefore, the focus is shifting towards personalized medicine with personalized treatments that are individually tailored to patients diagnosed with GBM. At present, prognosis of overall survival (OS) in GBM patients still remains poor.

The overall survival of patients diagnosed with glioblastoma multiforme (GBM) is around 15 months (Thakkar et al., 2014) even in case of aggressive therapy, consisting of surgery followed by chemoradiation and post-surgical adjuvant chemotherapy. Unfortunately, the aggressive growth of GBMs usually disables early diagnosis in initial growing stages. The most common noticeable symptoms and signs for patients with GBM are progressive focal neurologic deficits, headaches and seizures (Adamson et al., 2009). Only 2% of all GBM patients survive longer than 38 months (Smoll et al., 2013). The need to identify prognostic parameters for shortened or prolonged survival is therefore a vital part of medical research in oncology. Research in radiomics aims at predicting disease prognosis by extracting relevant imaging features from multiple sequences of magnetic resonance images (MRIs). Multi-modal therapy approaches have improved over the past decade, but the process of manual feature extraction for the prognosis has remained the same.

4.2 Brain Imaging

The gold standard imaging technique for observing disease progression and surgical planning are Magnetic Resonance Images (MRIs). The basic principle behind magnetic resonance imaging is the use of a magnetic field to produce an in-depth depiction of organs, soft tissue, bone and other parts of the internal body structure. MRI is a non-invasive, radiation-free method for the visualization of the anatomical structure of the brain that provides finer details of the brain due to tissue discrimination, multiple views (axial, sagittal and coronal) and multiple sequences depending on the acquisition settings i.e. T1-weighted, T2-weighted, and the fluid attenuated inversion recovery sequence (FLAIR) (Vial et al., 2018; Tandel et al., 2019). The differences between the views are illustrated in Figure 4.1. The tissues of the brain and present abnormalities appear differently on

different MRI sequences, which gives additional pathological information. The T1-weighted scan demonstrates differences in grey and white matter contrast, whereas the T2-weighted highlights water content. The last sequence is the fluid attenuated inversion recovery (FLAIR) which is more sensitive to the cerebrospinal fluid (CSF) making the CSF darker and present tissue abnormalities appear brighter.

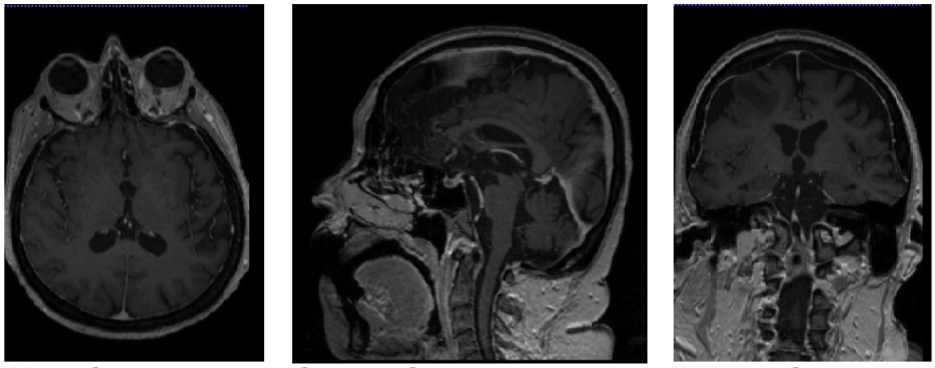

Figure 4.1: Illustration of the three views (axial (a), sagittal (b) and coronal (c) of MRIs

T1-weighted scans are commonly reported for various intracranial abnormalities, such as tumors, white-matter diseases or infectious lesions. However, a paramagnetic contrast agent such as gadolinium can be administered to obtain contrast-enhanced T1-weighted MR images that can detect cancerous cells and may reveal additional magnetic resonance features for lesions that are generally not detectable on conventional T1-weighted images of some patients. T1c can improve the contrast of the areas boarders and hence makes the boundaries of the tumor more visible (Higano et al., 2000).

One issue that arises when working with medical images is the non-uniformity of the intensities caused by field inhomogeneities. In general, the intensity of the same tissue should always be similar in one image and not vary at different locations within the same tissue. This facilitates the learning procedure of image processing algorithms such as convolutional neural networks. However, in practice, the piecewise constant property in medical images is often violated by artefacts introduced by the image acquisition technology. The undesired artefacts or random fluctuations are called the bias field or intensity non-uniformity and are not always visible to the human eye (Song et al., 2017). Bias field is not

only inherent in MRIs but can also occur in computer tomography (CT), X-rays and other imaging modalities. Several algorithms exist that take the MR image corrupted by a bias field as the input and outputs the corrected image. One of these approaches is the N4 algorithm (Tustison et al., 2010). The algorithm works by log-transforming the intensities of the input image and initializing a log bias field only consisting of zeros. The corrected image is then obtained by iteratively alternating between the estimation of the unbiased log image and the estimation of the log of the bias field. In simple words, the correction is performed by sharpening the input in the log intensity space. Uncorrected images can yield poor performance of the algorithm due to the possibility for the network to focus on negligible intensity fluctuations. Therefore, bias field correction constitutes an import image pre-processing step.

4.3 Survival Prognosis of Glioblastoma

In the past, numerous patient characteristics have been identified as potential prognostic factors of overall survival (OS) in GBM patients. Clinical features such as age, extent of resection, the Karnofsky Performance Scale (KPS) and features of pre-operative MRIs have been found to be associated with the OS of a patient (Lacroix et al., 2001).

In general, survival does not differ between male and female patients, but most authors agree that young age at diagnosis is a strong prognostic factor of long-term survival in patients with GBM. For instance, five-year survival rates are usually about 13% for patients aged between 15 to 45 years and are only around 1% for individuals aged above 75 years (SEER, 2007). Studies investigating prognostic factors of patients surviving past 3 years after the initial diagnosis have estimated the average age to be around 40 years, whereas patients older than 60 years are more likely to be associated with shorter survival (Naydenov et al., 2011).

Another important predictor of long-term survival in patients with GBM is given by the Karnofsky Performance Scale (KPS) at the time of admission. The KPS score is an attempt in oncology to quantify the status, activity, quality of life and well-being of a patient diagnosed with cancer. The score range from 0 to 100, where 0 represents dead and 100 indicates a perfectly healthy state without

any evidence of the disease. To name examples, a patient associated with a KPS of 20 would be considered very sick and require hospital admission, whereas patients with a score of around 70 would be able to care for themselves but would be unable to work. Many studies have shown that a KPS score larger than 80 before undergoing resection surgery is associated with improved survival times (Naydenov et al., 2011).

The Eastern Cooperative Oncology Group (ECOG) score (Oken et al., 1982) is a simplified scoring system to quantify the effect of the illness on the state of the patient. It runs from 0 to 5 with 5 indicated death and 0 perfect health. Two states of the KPS are generally represented by one state of the ECOG. For example, grade 0 of ECOG would include a KPS score of 100 and 90. A full description of the ECOG score is presented in Table 4.1.

Table 4.1: Performance status of cancer patients

Grade	ECOG description
0	Fully active, no restrictions
1	Restricted physical activity, but able to carry out light work
2	Symptomatic and $> 50\%$ waking hours in the day
3	Symptomatic and $< 50\%$ waking hours in the day, but not bedbound
4	Completely disabled and bedbound
5	Death

Further, it is proven that the extent of resection is a significant independent predictor of overall survival in GBM patients. In general, a resection of more than 98% of the malignant tissue can lead to a doubling of survival. However, GBM is known for infiltrating adjacent brain regions, which is why a complete resection is often times very difficult or impossible (Adamson et al., 2009). The topographically diffuse nature of the glioblastoma and the variable location of the tumor cells makes accurately defining the delimitation of the malignant tissue and thus the surgical resection a challenging and demanding task. The extent of tumor resection that has been undertaken can be classified into the categories of gross total resection, subtotal resection, partial resection, and biopsy sampling (Lacroix et al., 2001). General guidelines recommend removing as much of the tumorous tissue as possible. However, the exact location and borders of the glioblastoma have to be considered to avoid healthy brain regions.

Although, glioblastoma multiforme is the most common form of gliomas in adults, the high-grade neoplasm has a global incidence of less than 10 per 100,000 people Hanif et al. (2017). Therefore, data acquisition to obtain a large enough training sample for survival prediction models is difficult. Because of the small data sets when it comes to GBM, a common method is survival discretization. Since, the outcome variable (survival) is continuous, obvious predictable grouping patterns among patient survival times may not arise. Therefore, patients surviving longer than a predefined threshold are grouped together and patients surviving shorter are placed in another group. Results based on survival discretization are not only easy to interpret, but also intuitive to the clinician.

Thus, the major reason for choosing an OS group classification scheme is to ensure sufficient training samples to create a predictive model with good accuracy and overall performance. However, consensus regarding a proper cut-off point for the grouping of survival times in the scientific literature is non-existent. Patients are usually either stratified into two or three groups with arbitrary thresholds to reflect short-, mid- and long term survivors of GBM. A few most common options for the cut-off point for OS grouping are illustrated in Table 4.2.

Table 4.2: Summary of used cut-offs points (in months) for OS group survival classification in patients with GBM in the literature

short-term	mid-term	long-term	Literature
< 6	6-18	> 18	Macyszyn et al. (2015)
< 12	12-36	> 36	Adeberg et al. (2014)
< 18	18 - 36	> 36	Burton et al. (2002)
< 22	-	> 22	Nie et al. (2016)
< 24	-	> 24	Piccolo and Frey (2013), Colman et al. (2010), Nigro et al. (2005)

4.4 Related Work: Deep Learning in Glioma Imaging

In the past decades, deep learning has gained a considerable amount of attention, because it opens up new dimensions for radiology. Learning algorithms can classify medical images in some applications as surely as doctors. From this, radiologists could benefit very well in their daily work, for example, by automatically differentiating normal findings from pathological findings. Numerous clinical studies have focused on two main applications of machine learning to medical image data of brain tumors in recent years namely tumor segmentation and MRI radiomics. Although the two areas will be treated separately in this section, they can be linked together since segmentation forms a good basis for further image processing analysis.

Segmentation aims at capturing the contours of an object of interest through the assignment of each individual pixel to the corresponding object category. In order to derive accurate deep learning algorithms for segmentation, an important requirement is having high-quality ground truth data that is representative of the reality. However, often times data acquisition is tedious and a time-consuming task. In addition, the data is often times heterogeneous. As a solution, the Medical Image Computing and Computer Assisted Intervention (MICCAI) Community introduced the annual Brain Tumor Segmentation (BraTS) challenge in 2012 (Menze et al., 2015). Participating international teams of the challenge receive data consisting of co-registered and skull-stripped unenhanced T2-weighted, contrast-enhanced T1-weighted, T2-weighted FLAIR images, 3D segmentation masks and survival times of patients with high-grade (n=210) or low-grade (n=75) gliomas in order to derive an algorithm for tumor segmentation and survival prediction. Another publicly available data set is presented The Cancer Genome Atlas of The Cancer Imaging Archive (Clark et al., 2013), which consists of preoperative MRI recordings and a set of additional radiomics features of patients diagnosed with glioblastoma (n=135) and low-grade glioma (n=108).

Various implementation approaches of the segmentation of gliomas can be found in the literature but can mainly be categorized into two approaches, classic machine learning and deep learning (Lotan et al., 2019). Classic machine learning methods such as support vector machines (SVM), k-means clustering and ran-

dom forests require hand-crafted features, which is why deep learning using convolutional neural networks has become the standard procedure for segmentation tasks. One downside of deep learning approaches in comparison with conventional classifier methods is the additional computational cost due to the complex architecture of the networks. Hence, CNNs for segmentation can be divided according to three different implementation approaches: patch wise, semantic-wise or cascaded. Initially, the most common network architecture was the patch wise approach (Pereira et al., 2016; Zhang et al., 2015) in which small patches of the image data are extracted and high-level features are captured.

Pereira et al. (2016) proposed a patch wise 2D network with patch wise intensity normalization consisting of 3x3 convolutional filters which have a positive effect against overfitting and allow designing of a deeper architecture. Hence, the approach obtained the overall first position by the online evaluation platform of the BraTS challenge in 2013. In 2017, Havaei et al. (2017) presented a patch wise 2D deep learning approach consisting of a 2-phase training procedure to eliminate problems associated with the imbalance of tumor labels. This is referred to as a cascaded approach where the output probability vector of one CNN is fed as an additional input to the layers of another CNN. Multiple 3D CNNs were proposed by Casamitjana et al. (2016) which showed good accuracy while incorporating the 3D nature of the gliomas. Later, the implementation shifted towards semantic segmentation with one of the most known deep learning algorithms being the U-Net proposed by Ronneberger et al. (2015). Semantic segmentation works by dividing the images into subregions and stitching the outputs of these regions back together by using additional convolutional layers. This procedure is performed in the U-Net through down- and upsampling layers as described in Section 3.4.1. U-Net outperformed all other approaches in the 2017 BraTS competition with a dice score (measure of similarity between ground truth and predicted segmentation) of 0.90, sensitivity of 91% and a specificity greater than 99% (Lotan et al., 2019).

Comparably new is the field of study in radiomics that detects and extracts quantitative features from medical imaging recordings to investigate if these deep features can predict overall survival or the recurrence time of patients with gliomas. In 2017, Lao et al. (2017) combined deep learning with traditional survival modelling. First, around 98 304 features using CNNs and 1403 hand-crafted features were extracted from pre-operative multi-modality MR images (n=112). After

feature extraction, a four step- feature selection method was applied reducing the 99 707 features to only 150 robust and predictive features which were used in the subsequent analysis in a least absolute shrinkage and selection operator (LASSO) Cox regression model to obtain a radiomics signature for prediction of survival in GBM patients. The obtained radiomics signature achieved a C-index of 0.710 (95% CI: 0.588,0.932) for the independent validation cohort. Nie et al. (2016) proposed a 3D CNNs for multi-modal pre-operative brain images (T1 MRI, fMRI and DTI) in combination with a support vector machine to predict short or long overall survival time of patients (n=69) diagnosed with high-grade glioma. The approach achieved an accuracy around 90% and in particular, the fMRI and the DTI proved to play a significant role in the successful prediction of survival, not the T1 MRI scan. In 2018, Mobadersany et al. (2018) presented a survival convolutional neural network (SCNN) to combine the power of adaptive machine learning with conventional survival models using high-resolution histology images of 769 patients diagnosed with glioma. The SCNN model showed substantial predictive power with a C index of 0.754.

In 2016, Kickingereder et al. (2016) stated that machine learning-based radiomics models are insufficient for clinical use. However, recent years have provided clinically relevant breakthroughs and advances in the analysis of glioma imaging as the mentioned publications show. The field will continue to evolve and move towards more complex and deeper network architectures for assessing brain tumors as early as possible. Difficulties still exist in the shortage of well-annotated medical imaging records following a general standard to facilitate learning of the deep learning algorithms. Another often mentioned point of criticism of neural networks is the fact that it is a black box approach. However, research in the visualization and deeper comprehension of the inner operating principles is still in a very early phase but can soon solve this mystery as well (**?**)

Chapter 5

Methodology and Results

In the following chapter, our contribution to machine learning will be demonstrated on a data set consisting of the pre-surgical MRI volumes of patients diagnosed with glioblastoma multiforme (GBM). Before the chapter presents the computer algorithms developed to solve the segmentation and survival prediction problem, the software and programming environment used for the task will be described. Then, the chapter will be structured into two main parts. First, the focus will be placed on the implementation and the results of the glioblastoma segmentation algorithm. Second, we will present two convolutional neural networks able to perform overall survival (OS) prediction and OS group classification, when survival is discretized into two classes. At the end of the chapter, the results will be analysed and discussed.

5.1 The Implementation Environment

The implementation of the segmentation and prediction algorithm was carried out in the programming language Python, which offers a wide variety of Deep Learning (DL) libraries. The algorithms proposed in the following sections are mainly based on the Keras Application-Programming-Interface (API), which is running on top of the TensorFlow library. Therefore, both libraries will be introduced shortly in the following. For image pre-processing the Python library Nipype was used which provides a uniform interface to currently available neuroimaging software and facilitates the exploration of existing packages through a single workflow.

Keras is a high-level neural networks API written in the programming language

Python, which was introduced in 2017 and has gained traction since then. The deep learning library provides easy and flexible implementation of CNNs due to four core guiding principles: modularity, minimalism, easy extensibility and work with Python. Modularity describes the ability to implement the building blocks (e.g. convolutional layer, pooling) of a convolutional neural network easily as a system of connected modules. To keep the system of modules manageable and comprehensible, minimalism is an important feature. The function and use of a module should be clearly understandable. Next, the system should be capable to be easily extensible to a more complex architecture if needed (Chollet, 2015). Lastly, the use of Python allows for all of the above mentioned characteristics to be easily implemented and debugged.

As mentioned, Keras can be run on top of TensorFlow which solely means that Keras is a simplified interface to TensorFlow. Although TensorFlow is the most famous open source library used in the development and training of deep learning models, it is not easy to use. Thus, it is commonly recommended for beginners of deep learning to start with Keras and then, in case further functionality or flexibility is needed to drop down to TensorFlow. The presented implementations in this work will be solely based on Keras. A simplified example of a small CNN implemented in the Keras API is illustrated in the Appendix to give the reader a concept of its operating principle based on modules.

5.2 The Data Set

The provided GBM data set consists of 136 patients who had their MRIs taken a few days before undergoing surgical resection of the glioblastoma within the time period of 2003 to 2015. The following institutions provided MRIs and clinical information for each patient: Medical University of Vienna (including Hospital Rudolfstiftung Vienna), Kepler University Hospital Linz, Paracelsus Medical University Salzburg, Medical University of Innsbruck, Karl-Landsteiner University Hospital St. Pölten, State Hospital Klagenfurt, State Hospital Wiener Neustadt, and the Medical University of Graz (Klughammer et al., 2018). All MRI volumes contained in the supplied MRI dataset were in the NIfTI-1 data format created by the Neuroimaging Informatics Technology Initiative.

The NIFTI imaging data of an individual include MRI volumes from two co-

registered brain tumor MR sequences namely T1- contrast enhanced (T1c) and fluid-attenuated inversion-recovery MRI (FLAIR). Since the MRI scan acquisition was carried out between 2003 to 2015 varying image resolution of the NIFTI files were observed across the data set as can be seen in the scatterplot in Figure 5.1. Each dot represents the 3D-data point consisting of (width, height, depth) of one MRI volume. Significant differences were not only present in height and width of the MRI volumes, but also in the depths which ranged from 19 to 512. The height and width of all axial images of each MRIs sequence were rescaled to a dimension of 512x512.

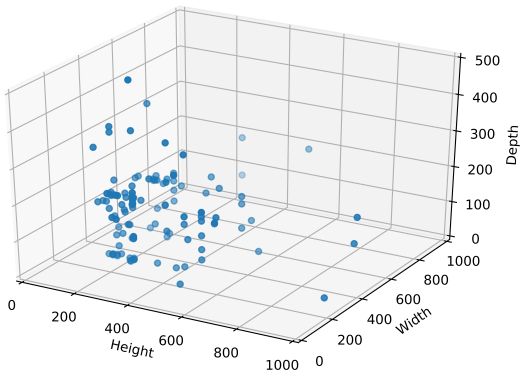

Figure 5.1: Scatterplot of the width and height of each MRI volume in the GBM data set

The sequences from one sample subject of the GBM cohort can be observed in Figure 5.2.

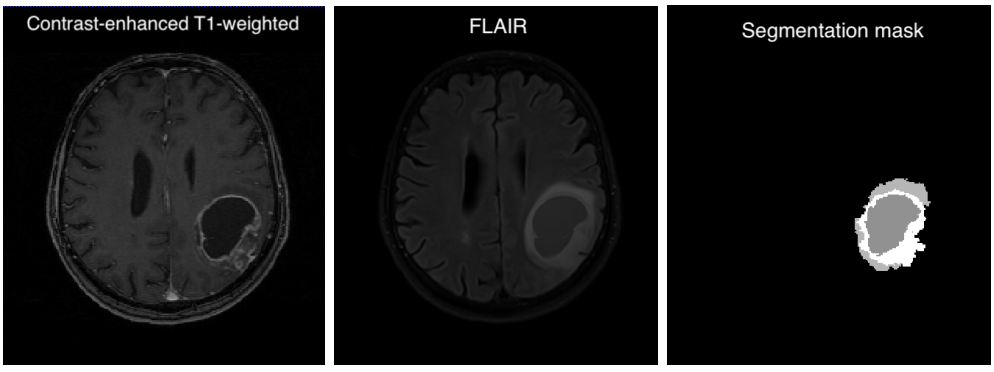

Figure 5.2: The same axial image under the two MRI sequences (T1c (left), FLAIR (middle)) and its segmentation mask for a sample subject diagnosed with glioblastoma multiforme within the data set

The differences of the magnetic resonance features in T1c and the FLAIR MRI are easily apparent. As we can see in Figure 5.2, the FLAIR sequence is very sensitive to pathological tissue and therefore makes the differentiation between the cerebrospinal fluid (CSF) and an abnormality much easier. However, the T1c sequence had in most cases a much higher spatial resolution in comparison with FLAIR. The segmentation files used as the ground truth were not manually annotated by experts but automatically segmented by the Brain Image Analysis software BraTumIA and had to be manually corrected slice by slice due to the presence of faulty segmentations.

Additional clinical information included the age, gender and gene-expression subtype of the patient. Further information, which was only partly present, consisted of the extent of resection and the recurrence time, which were extracted from the electronic medical records of the corresponding hospital. None of the patients had undergone prior GB resection. The overall survival (OS) of a patient diagnosed with glioblastoma multiforme was defined as the duration from the date of the pre-operative MRI scan of the patient to the date of tumor-related death. All causes of death were tumor related. Patients of the GBM data set underwent resection surgery typically 2-3 days after the baseline MRI and received a post-operative MRI scan as well. However, only the pre-operative MRI scans will be used in this work. Overall survival of the patients was classified into two categories according to literature (Table 4.2): short-term (< 24 months) and long-term (≥ 24 months). Loss to follow up was only present in 12 patients. Out of these 12 patients, 6 patients survived longer than 24 months.

Table 5.1: Baseline characteristic distributions of the GBM study participants stratified by the survival class

Variables	Long-Term Group	Short-Term Group	P	Available (%)
Total	41	95		100
Age	56.7 (11.4)	59.3 (12.7)	0.258	100
Male (%)	27 (65.9)	61 (64.2)	1.00	100
Survival Time (months)	39.1 (17.2)	12.25 (6.41)	<0.001	91.5
Recurrence Time (months)	20.2 (11.3)	9.2 (4.1)	<0.001	78.5
Extent of resection (%)			0.669	65
Full	11 (26.8)	31 (32.6)		
Partial	14 (34.1)	34 (35.8)		
Unknown	16 (39.0)	30 (31.6)		

Note: Continuous variables are denoted as mean ± standard deviation, dichotomous variables as total numbers with the condition (%) and categorical variables are stated in terms of total numbers with the condition within each subgroup (%).

In Figure 5.3, we can clearly see the differences in overall survival and progression-free survival between the two survival groups. Short-term and long-term survival was classified according to overall survival.

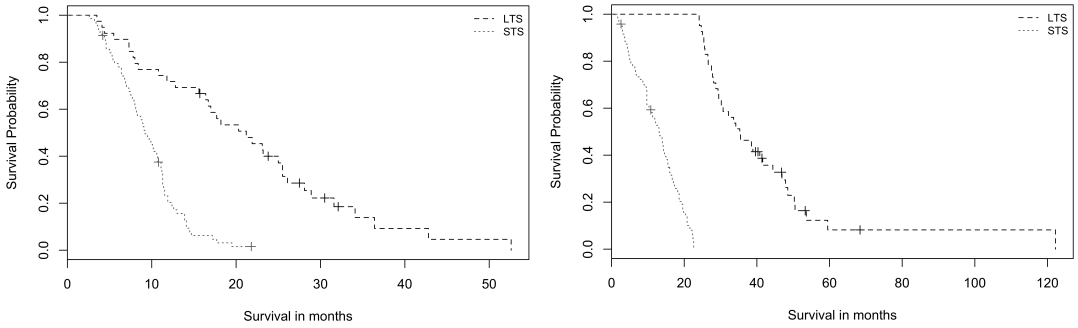

Figure 5.3: Progression-free survival (left) and overall survival (right) of short-term survivors (STS) and long-term survivors (LTS) of glioblastoma patients.

5.3 Segmentation of Glioblastoma

In this section, we present a fully automatic brain tumor segmentation method based on Deep Convolutional Neural Networks (DCNNs). The neural network architecture is inspired by the U-Net structure as described in Section 3.4.1. Semantic segmentation can be viewed as a fine-grained pixel classification task where each point in an image is, in the case of brain cancer segmentation, either classified as tumorous tissue (tumor and edema) or healthy tissue. In general, only three different types of tissue can be observed in the brain of a healthy individual: the white matter, the grey matter, and the cerebrospinal fluid. The main objective of brain tumor segmentation models are to detect the location and extension of malignant abnormalities such as tumorous and necrotic tissue.

5.3.1 Image Preprocessing

In order to obtain an automatic segmentation algorithm, which does not rely on prior image pre-processing steps, extensive image pre-processing was not applied for the model performing the glioblastoma detection and segmentation. The MRI data set was restricted to the contrast-enhanced T1-weighted scans $\mathcal{I}_C$ and the 3D segmentation masks $\mathcal{I}_M$. The FLAIR volumes $\mathcal{I}_F$ and the clinical information of a patient were not required as an additional input. The 3D segmentation

masks form the ground truth for the segmentation algorithm. Therefore, the data used for the supervised segmentation task can be denoted by the set $(\mathcal{I}_C, \mathcal{I}_M)$. Each of the $\mathcal{I}_{C;i}$ with $i \in \{1, .., 136\}$ was separated into its axial images $I^j_{T;i}$ with $j \in 1, ..., d_i$, where d_i denotes the depth of $\mathcal{I}_{C;i}$ and $\mathcal{I}_{M;i}$ of patient i. It follows $\cup_{j=1}^{d_i} I^j_{C;i} = \mathcal{I}_{C;i}$. Slices of the sagittal MRI that did not contain tumorous tissue were removed. Further, axial images of the peripheral region of the tumor were removed as well by cropping one fourth of the axial tumor images in the upper plane and one forth in the lower plane, since the focus was placed on the detection and segmentation of the core area of the glioblastoma. Thus, we obtain as the input for the segmentation model the subset $(\tilde{\mathcal{I}}_C, \tilde{\mathcal{I}}_M)$ of $(\mathcal{I}_C, \mathcal{I}_M)$, where $\tilde{\mathcal{I}}_{C;i} = \cup_{j=1+\frac{d_i}{4}}^{d_i - \frac{d}{4}} I^j_{C;i}$ and $\tilde{\mathcal{I}}_{M;i} = \cup_{j=1+\frac{d_i}{4}}^{d_i - \frac{d}{4}} I^j_{M;i}$.

An illustration of the image pre-processing procedure can be observed in Figure 5.4. The area confined by the red dashed lines indicates the axial region of the brain containing the malignant neoplasm. Nine example slices are illustrated on the right side of Figure 5.4. In total, 37 axial images of size 512x512 containing the core tumor region were extracted from a MRI volume of size 512x512x192.

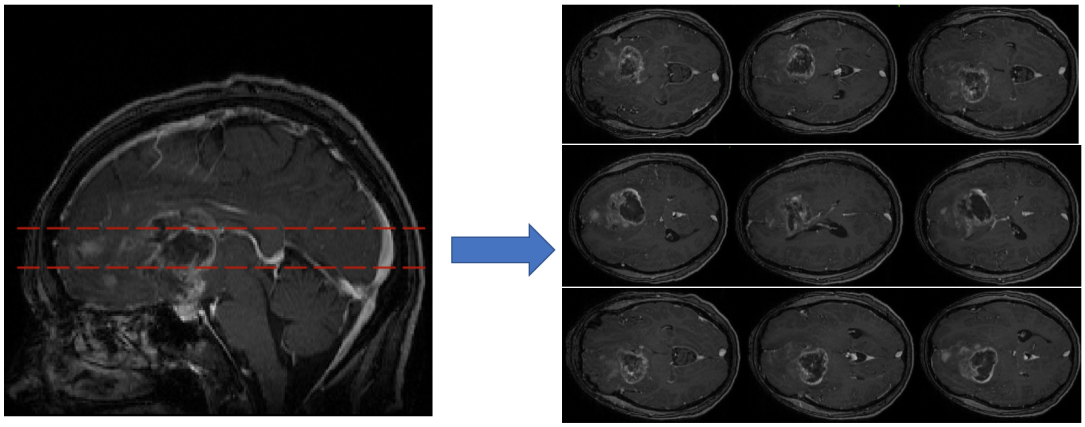

Figure 5.4: Illustration of sagittal MRI view (left) and 9 axial example images (right) extracted from the tumorous region confined by the two dashed red lines

5.3.2 Network Architecture

The Network architecture for the segmentation task is inspired by the popular segmentation architecture U-Net of Ronneberger et al. (2015), which we described in Section 3.4.1. The architecture contains nine convolutional blocks, 4 belong-

ing to the downsampling part of the U-Net structure and 4 for the upsampling part of the network. Each convolution block in the contracting path consists of two consecutive convolutional layers with a receptive field of size 3 followed by a Rectified Linear Units layer and a max pooling layer. The number of filters is 8 in the initial block and then doubles for the following blocks. Hence, when the j^{th} axial image of patient i denoted by $I^{j}_{C;i}$ of size 512x512x1 is fed into the first convolutional block of the network, the input image is reshaped to a 3D tensor of size 256x256x8 after flowing through two convolutional layers with 8 filters of size 3 followed by a ReLU layer and one maxpooling layer which is responsible for the downsampling of the image input. The 3D tensor is then put through 3 additional convolutional blocks and changes from 256x256x8 to 32x32x64. The convolutional layers in the blocks increase the number of feature maps from 8 to 64, whereas the max pooling layers reduce the height and width of the input. The ReLU layer is a supplementary step to the convolution operation to increase the non-linearity in our images.

Two convolutional layers connect the down- and upsampling parts of the U-Net structure, hence the number of feature maps increases from 64 to 128 before the input enters the upsampling pathway.

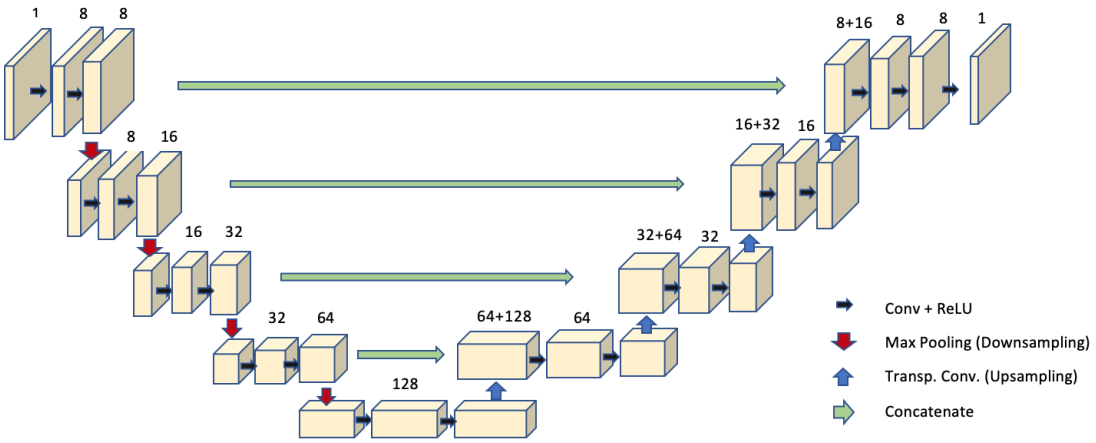

Figure 5.5: Overview of the network architecture inspired by U-Net

Each convolutional block in the expansive path of the U-Net inspired architecture consists of a transposed convolutional layer followed by two convolutional layers with a receptive field of size 3 and a max pooling layer. The transposed convolutional layer is responsible for the upsampling of the output vector of size

32x32x128 obtained by the contracting path. After the upsampling of the 3D tensor has been performed by the transposed convolutional layer, the 3D tensor is of size 64x64x64. Next, the input to the next block in the expansive path is concatenated by the feature maps of the corresponding contracting block. Thus, the two inputs which both have the dimensions 64x64x64 are stacked together forming a 3D tensor of size 64x64x128. Then, two convolutional layers are applied followed by the ReLU function. The number of filters of the two convolutional layers in the initial block equals 64 and halves for every following block. The output is then fed to the next convolutional block where it is concatenated with the output feature maps of the corresponding contracting block.

Lastly, the output layer consists of a convolutional layer with one filter and a receptive field of one to associate the feature mappings with the desired classes (tumor or no tumor). The 3D tensor is thereby transformed from 512x512x 8 to the segmented image output of size 512x512x1. For all layers except the output layer, the Rectified Linear Unit activation function was used to introduce non-linearities to the output of the convolution layer. For the output layer, a sigmoid activation function was used mapping the feature maps in the last layer to the binary segmentation classes. The sigmoid function predicts if a pixel belongs to a specific class in our case the tumorous tissue by restricting the input values from a large scale to be within the probability range 0 to 1 (see Section 2.2.3). The resulting number essentially indicates the confidence the algorithm has in classifying the pixel in the image as belonging to the tumor tissue. The obtained output is then compared to the ground truth by means of a loss function yielding an error measure. The loss function for the segmentation task was set to the binary crossentropy loss given by

$$\mathcal{L}(\theta) = -\sum_{x} \big(y_x \; log(\hat{y_x}) + (1 - y_x) \; log(1 - \hat{y_x})\big) \tag{5.1}$$

where θ denotes the model parameters, y refers to the true label mask, y_x to a single element of the true segmentation mask , $\hat{y}$ to the prediction of the output image and $\hat{y}_x$ to a single predicted element of the segmentation prediction.

In addition, as a measure of performance, the overall Pixel Accuracy (oPA) was

computed in each epoch by

$$oPA = \frac{\sum_i n_{i,i}}{\sum_i \sum_j n_{i,j}} \tag{5.2}$$

where $n_{i,j}$ denotes the number of pixels of class i predicted to belong to class j. The oPA measures the proportion of correctly labelled pixels of the image. However, one significant downside of the oPA is its inherent bias in the case of imbalanced classes meaning that only a small portion of the image belongs to a certain class. Hence, the standard metric for the evaluation of performance is generally the Intersection over Union (IoU) given by

$$IoU = \frac{\sum_i n_{i,i}}{\sum_j n_{i,j} + \sum_j n_{i,j} - n_{i,i}} \tag{5.3}$$

The IoU as the name states measures the intersection over the union of the labelled segments for each class and takes the average. Hence, the false alarms as well as missed pixels are taken into account (**?**). A limitation of this approach is that it only evaluates the total number of correctly labelled pixels, but not the accuracy of the segmentation boundaries. Therefore, the oPA as well as the IoU were computed for this experiment. In general, an IoU score above 0.5 is normally considered a "good" prediction, whereas a score around 0.6 is an excellent result.

5.3.3 Training

Training and testing was conducted using the full data set consisting of 136 GBM patients, 80% of the patients (108 patients) were randomly assigned to the training data set, while the remaining 20% of the patients (28 patients) were used as an independent validation set. Since each MRI was separated into its axial images and the upper and lower images not containing tumorous tissue cropped, a total of 3517 images was used for the training of the segmentation algorithm and 985 axial images were used for validation purposes. The validation data set was not touched until training was complete. Every patient was part of either the training or validation set, but never of both.

For training purposes, a batch size of 32 was chosen. Thus, 110 iterations were required in order for one epoch to finish. Initially, the hyperparameters of the optimization function were set to the original parameters of the U-Net used in

Ronneberger et al. (2015), which involves Stochastic Gradient Descent (SGD) as an optimizer with a learning rate of 0.0001 and momentum of 0.99. SGD is a simplified version of the standard gradient descent approach where the stochasticity of the optimizer comes from employing mini-batches of the input to compute the gradient at each descent. This property makes the SGD particularly interesting for the optimisation of highly non-convex loss functions. However, the original parameters did not yield sufficient results, which is why the hyperparameters and the optimizer were tuned until acceptable performance was achieved. Multiple hyperparameters of the network were varied in order to make the network reach a convergence point. For instance, we tried various combinations of varying learning rate, different optimizers and filter sizes of the convolutional blocks to make the training converge faster. We first trained the model using learning rate of 0.01 and observed fluctuations in the cost function during training, so we tested a smaller value of 0.0001 which caused the network to converge after a reasonable number of epochs. In the end, the adaptive learning rate optimization algorithm called Adam (Adaptive Moment Estimation) (Kingma and Ba, 2014) was used with a learning rate of 0.0001. The Adam optimizer is currently the state-of-the-art optimization algorithm and consists of the RMSProp with momentum. RMSProp stands for root mean squared propagation and divides the gradient by a running average of its recent magnitude. Momentum is an update rule of the learning rate similar to friction or damping in physics. For the loss function, the binary crossentropy loss function as stated in Equation (5.1) was chosen. The final training of the automatic segmentation algorithm proceeded for 8 epochs until convergence was reached.

Another important aspect of training for small data sets is data augmentation. Various methods of data augmentation can be available such as flipping the image horizontally and vertically, rotation, zooming, changing the brightness of the image, translation and rotation (see Section 3.3.1). The choice of applied transformations highly depends on the task, the model should be able to perform after the images have been altered. It is important to note that the image should not be transformed in a way that might interfere with the ability of the algorithm to learn from the images. Horizontally flipping an arbitrary axial image of the T1c recording still yields a valid training instance since the flipped image still contains the same information but due to the change presents a "new" training sample. By horizontally flipping each sample of the data set, the sample size is essentially doubled. In this experiment, data augmentation solely consisted of horizontally

and vertically flipping the image to increment the sample size of the axial images obtained from the contrast-enhanced T1-weighted records.

5.3.4 Results

A deep learning algorithm using a convolutional neural network inspired by the U-Net architecture was developed in order to detect and segment the tumorous tissue defined as the core tumor region and the associated edema. We used the axial images of the contrast-enhanced T1-weighted MRI recordings and the 3D segmentation mask of all patients but excluded images that did not contain the core tumor region. The model was trained until the binary crossentropy function reached its minimum. The overall pixel accuracy (oPA) and the Intersection over Union (IoU) score were recorded as a measure of similarity between the ground truth and the prediction.

The final segmentation model reached an overall pixel accuracy (oPA) of about 95.0%, an intersection over union (IoU) of 50% and a loss of 0.046 % in the training set. In the validation set, the model achieved similar performance with an oPA of 95.1%, an IoU of 50% and an only slightly higher loss of 0.065%. The oPA and the IoU indicate that the model performed well. The graphical results of the CNN algorithm for high-grade glioblastoma segmentation are presented in Figure 5.6.

Patient A and B demonstrate the capability of the automatic segmentation algorithm at its best performance. The lighter borders of the predicted segmentation represent uncertainty, whereas bright yellow pigmentation imply highly confident prediction of the algorithm. Only a small border of uncertainty can be observed around the two predicted segmentation. Overall the algorithm was able to capture the shape and size of these two subjects accurately.

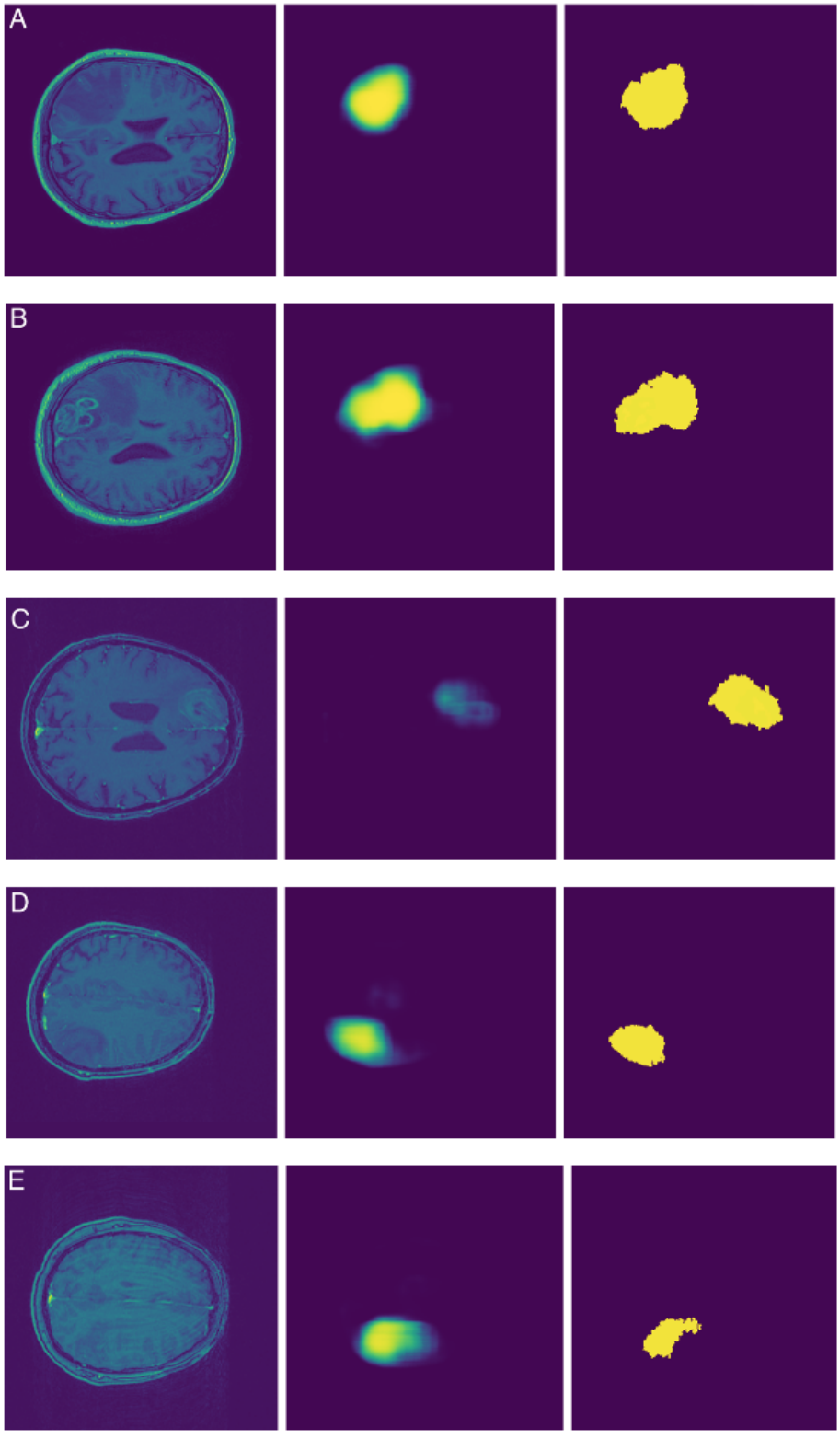

Figure 5.6: Five sample segmentation results (A-E) of the validation cohort from the segmentation CNN model. Results are presented by the T1c image (left), the predicted segmentation (middle) and the ground truth (right)

In contrast, it can be observed for patient C that although the tumor is even clearly visible to the human eye the algorithm can detect it only with high uncertainty. This may suggest either that the algorithm was not able to generalize very well to this instance or that the algorithm might have learned features for the detection and segmentation task which are not visible to the eyes. The latter hypothesis finds support in the predicted segmentation of patient E. Although, the tumorous tissue is clearly more visible in the T1c image of patient C, the algorithm predicts the tumor of patient E with a much higher certainty. The shape of the prediction and the ground truth slightly vary but overall it is able to detect and segment the tumorous tissue from the presented image of the independent validation cohort after extracting prognostic features from the training data set. Patient D also shows a good result. However, more uncertainty of the segmentation can be observed than for patient A and B.

5.4 Survival Prediction of Glioblastoma

The ability of convolutional neural networks can not only be used to detect and segment glioblastoma multiforme (GBM) but also to extract knowledge from the image environment that is prognostic for overall survival (OS) of patients with GBM. Hence, the aim of this experiment was to develop a machine learning model for predicting the survival time of patients with GBM based on clinical features, the pre-operative MRI recordings and the 3D segmentation masks.

5.4.1 Image Preprocessing

Since prediction of survival is a rather complex task, all available MRI sequences $(\mathcal{I}_C, \mathcal{I}_F, \mathcal{I}_M)$ were used for the model. Further, nearly all image-based segmentation algorithms crucially depend on the pre-processing procedures applied to the input image. Data pre-processing constitutes of multiple important steps for subsequent image analysis tasks. In pre-processing of MRI scans, this usually includes skull-stripping, bias field correction and intensity normalization. Segmenting brain tissue from non-brain tissue is a crucial step in many MR brain image pre-processing algorithms, because the structural analysis of brain regions can be more easily and accurately performed after the brain is extracted. The process is called skull-stripping and was applied to all MRI sequences of the GBM data set by using the Brain Extraction (BET) tool provided by FSL, a library of analysis tools for FMRI, MRI and DTI brain imaging data. An example before

and after skull-stripping can be observed in A1 and A2 in Figure 5.7, respectively.

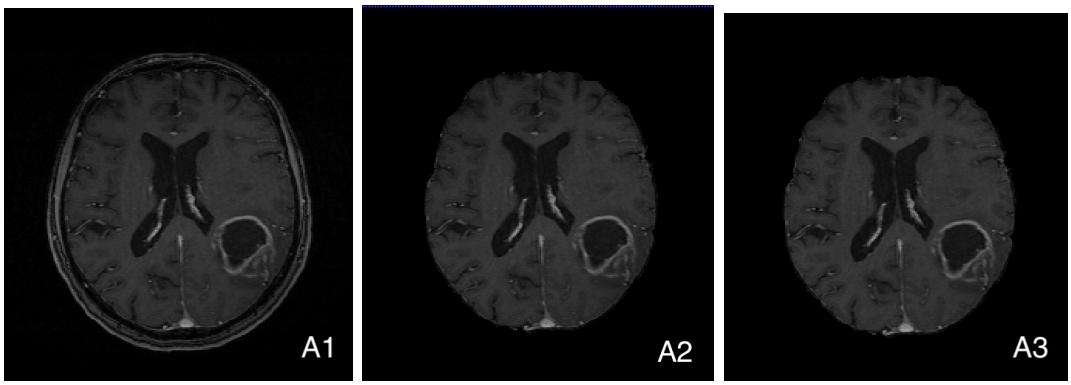

Figure 5.7: Example MRI slice before skull stripping and N4ITK bias field correction (A1), after skull stripping (A2) and after performing skull stripping and N4ITK bias field correction (A3)

After brain extraction, N4ITK bias field correction was performed using the Python library Nipype to eliminate image intensity non-uniformities caused by magnetic field variations. The artefacts are small spatial variations of the same tissue making it appear brighter in some areas. This can confound a tissue classifier which is why bias field correction using the N4 algorithm was applied to the skull-stripped MRI volumes. These intensity variations are hardly visible to the human eye, but an example of before and after bias field correction is illustrated in the middle (A2) and left (A3) graph of Figure 5.7.

In order to reduce the potential effects of varying contrasts across the image data due to deviations in their pixel intensity values, intensity normalization was applied to all NIFTI data files. The pixels of each MRI volume were normalized to a range of [0,1] by subtracting the minimum pixel value and dividing it by the intensity range of the MR image's pixels. Pixel normalization is an important step in order for the same neural network to use MRI data from different institutes with varying MRI scanners. After normalization, images with dissimilar width and height were padded so that its width equals its height to avoid image distortion during scaling. As a last step, all axial images of the MRI scans were scaled to the same width and height of 512x512. Although, the dimensions between the individual MRIs widely ranged and more MRIs were found with a lower dimension than 512x512, scaling the high-resolution images down in order to comply with the dimensions of the majority of the scan would lead to a

greater information loss in the images than scaling the low-resolution images up. Up- and downscaling of the images was performed by bicubic interpolation over a 4x4 pixel neighbourhood using the Python library OpenCV.

Image pre-processing then followed the path taken for the segmentation model by splitting all MRI sequences into the individual axial slices containing the tumorous tissue. Since prediction models generally require more data then segmentation tasks, only one fifth of the upper and lower axial images were cropped. Hence, we obtain $\tilde{\mathcal{I}}_{C;i} = \cup_{j=1+\frac{d_i}{5}}^{d_i-\frac{d}{5}} I^j_{C;i}$, $\tilde{\mathcal{I}}_{F;i} = \cup_{j=1+\frac{d_i}{5}}^{d_i-\frac{d}{5}} I^j_{F;i}$ and $\tilde{\mathcal{I}}_{M;i} = \cup_{j=1+\frac{d_i}{5}}^{d_i-\frac{d}{5}} I^j_{M;i}$ where $I^j_{C;i}, I^j_{F;i}$ and $I^j_{M;i}$ denotes the axial images of patient i, which were transformed by the above mentioned image pre-processing steps. To analyse all three axial images of the three 3D volumes at once, the corresponding axial images of the sequences were saved in the RGB image format with the colour channel (red,green,blue) corresponding to $(\tilde{I}^j_{C;i}, \tilde{I}^j_{F;i}, \tilde{I}^j_{M;i})$.

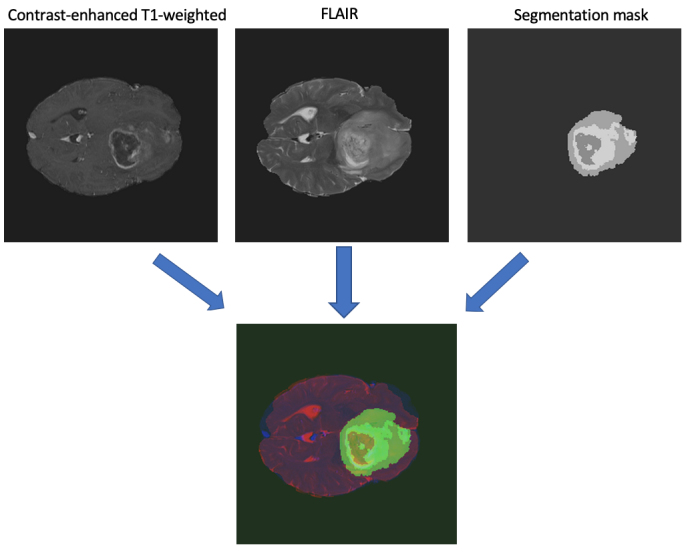

Figure 5.8: Illustration of the axial RGB image input for the survival prediction model

5.4.2 Network Architecture

For the architecture of the survival prediction task, ResNet-18 was used, which is a pretrained deep residual convolutional neural network trained on more than one million images from the ImageNet database. The neural network is 18 layers deep and can classify a variety of images into around thousand different object categories. Consequently, the network has learned rich feature representations

for a wide range of images and can be used as a starting ground for different image classification tasks. The ImageNet pre-trained weights of ResNet-18 were imported and three layers were added to the network. This approach is referred to as transfer learning and was thourougly discussed in Section 3.3.3. As stated in the previous section, the input of the model consisted of an RGB image of size 512x512x3 with the channels corresponding to $(\tilde{I}^j_{C;i}, \tilde{I}^j_{F;i}, \tilde{I}^j_{M;i})$. The output of the last convolutional layer of the pretrained ResNet-18 model was a 3D tensor of size 16x16x512. The output was then fed to a global max pooling (GAP) layer (see Section 3.3.4) which computed the maximum of all 512 feature maps and yielded a tensor of size 1x1x512. The feature vector was concatenated to a vector of size 513 with the addition of a data input vector namely the age of the patients and passed to a fully-connected layer with 16 units followed by a dropout layer with a probability of neuron deactivation of 60%. Lastly, the vector of size 32 was fed to the output layer using stochastic gradient descent with a learning rate of 0.001 as an optimizer. For the loss function, the mean squared error was used (see Section 2.2.4), since it is the most common loss function used in the machine learning literature for regression tasks. However, the data set was small, thus outliers might affect the weight configuration more heavily as they should, if the MSE loss function is used instead of the mean absoluter error (MAE) loss function. The scheme of the full model design can be seen in Figure 5.9.

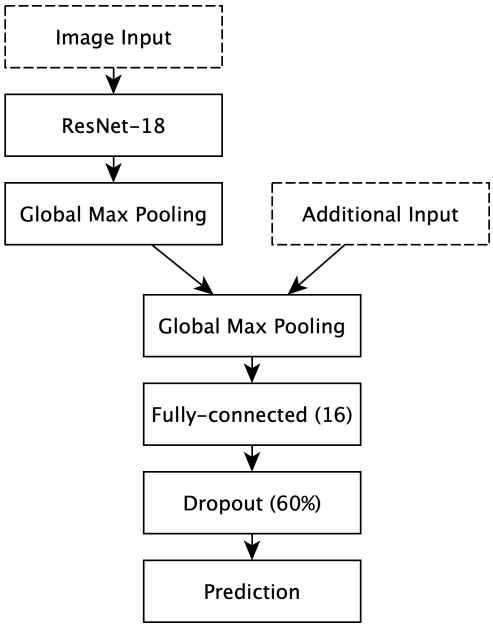

Figure 5.9: Model design for the OS time prediction based on the pre-operative MRI scans and the additional input age

5.4.3 Training

For the training of the CNN, patients that were lost to follow-up were excluded from the experiment, because CNNs can't handle censored data and thus right-censored patients would bias the results. Thus, a total of 125 patients were included in the training and validation procedure with 75% (93 patients) of GBM patients being assigned to the training set and 25% (32 patients) to the validation set. Again, the algorithm was trained on the axial images (1693 images) of the pre-processed MRI volumes. The model was validated on 584 axial images of the MRI sequences. One fifth of the axial images of each MRI volume and sequence was cropped to not bias the results by images containing only small peripheral areas of the tumor that are not representative of the true extent of the tumor.

During the tuning process to find suitable parameters, the CNN showed symptoms of overfitting very fast on the training data and showed highly variable performance between epochs. Model selection during training was therefore a challenging task, since both the mean absolute error and the mean squared error - when used as the cost function - behaved very unstable. The following approaches for model optimization were considered: the reduction of the model parameters by simplifying the CNN architecture, intense data augmentation and the inclusion of the optimization and regularization techniques described in the Sections 3.3 and 2.2.5 with various values for the present hyperparameters. However, each approach did not lead to an improvement of the generalization ability of the model or yielded a model that performed well on the training set but was not able to generalize at all in the validation data. In the end, the batch size was set to 16, thus 106 iterations were required to complete one epoch. To approximate the ground truth labels for all training inputs, we defined the loss function as the Stochastic Gradient Descent (SGD) optimizing function with a learning rate of 0.001. Minimal loss in the validation cohort was achieved at epoch 49. The CNN design required the training of 11 203 370 weights.

Data augmentation consisted of horizontal flipping, vertical flipping, shifting the height and width by a factor of 0.1, zooming by a factor of 0.1, altering the brightness by a minimum factor of 0 and a maximum factor of 0.5 and rotating the image by a maximum angle of 20 degrees. Points outside the boundaries of the input were filled with zeros. The examples of the axial images used for the training of the deep learning algorithm after data augmentation can be observed

in Figure 5.10.

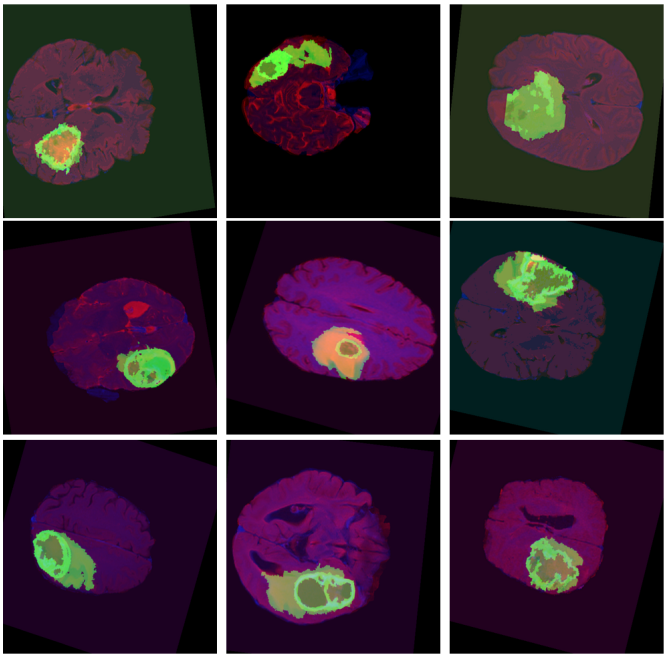

Figure 5.10: Examples of the axial images after data augmentation has been performed

5.4.4 Results

In this experiment, the aim was to develop a deep learning algorithm using a convolutional neural network for the overall survival prediction of patients diagnosed with glioblastoma multiforme. All MRI sequences and the 3D segmentation mask of 125 patients were used for the training process and the age of the patient was fed as an additional input to the model. The performance measures for the CNN model included train loss and validation loss represented by the mean squared error.

The minimal achieved value of test loss and train loss was 9.8 and 8.6, respectively. Although various regularization techniques and intense data augmentation were applied, the deep neural net did not improve. Best prediction for overall survival of patients with glioblastoma solely based on the image input was thus achieved through the use of the pretrained ResNet18 architecture in combination with the clinical feature age of the patient as an additional input. CNN model designs without the inclusion of the variable age showed a significant deterioration of the model performance.

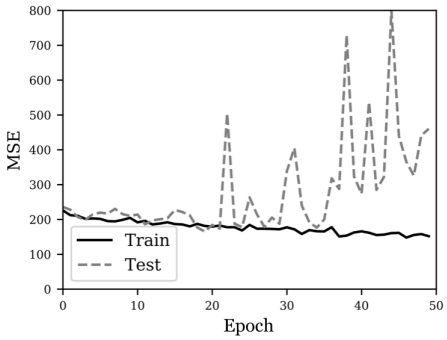

Figure 5.11: Mean squared error of the training and test cohort

The calibration plot of the prediction results can be observed in Figure 5.12. The left plot illustrates the prediction performed on each axial image of the independent validation set, whereas the righthand plot shows the average prediction of all images belonging to one patient. Although the model was not able to accurately predict each patient's overall survival, the predictions also did not show a lot of variation within the predictions of images associated with an individual despite the small amount of data available. Three extreme outliers can be observed in the right bottom corner of the calibration plot.

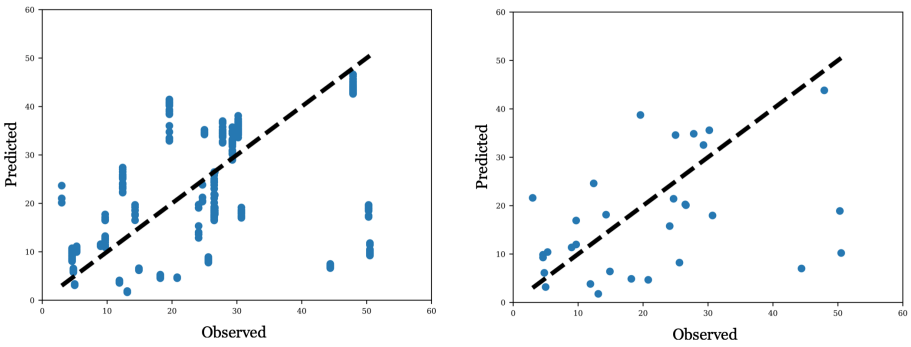

Figure 5.12: Calibration curve for the OS prediction according to the final CNN algorithm; prediction results for all axial images on the right; prediction results averaged for each patient on the left

The prediction model had difficulties accurately predicting the survival times of patients that survived more than 40 months. This might be due to the fact that the data set did not contain many individuals who survived that long in general. Thus, there were less instances for the CNN to detect features in the axial images of the MRI sequences associated with survival of more than 40 months. However, one patient who lived beyond 40 months was predicted rather precise. We can

take a closer look at the MRI sequences of the three instances that were mistaken for short-term survivors in Figure 5.13.

In the MRI sequences and the segmentation of patient A and B in Figure 5.13, we can see that during image pre-processing not all components of the skull could successfully be removed. These leftovers might have affected the subsequent bias field correction or the model training itself. In addition, the segmentation of patient B appears to have cut off large part of the edema of the tumor which is presented in a light grey colour in Figure 5.13. In contrast to the first two instances, the MRI sequences of patient C seem to represent a suitable sample of the GBM data set but still was wrongly predicted. One explanation apart from the underrepresentation of extreme long-term survivors in the data might also be found in the image pre-processing steps conducted prior to the analysis of the MRI scans. The original dimensions of the MRI volume were 208x256x192 (=192 axial images of size 208x256). Since CNNs require a standard dimension of all received inputs, the axial images were upsampled to a dimension of 512x512 which might have resulted in a loss of information.

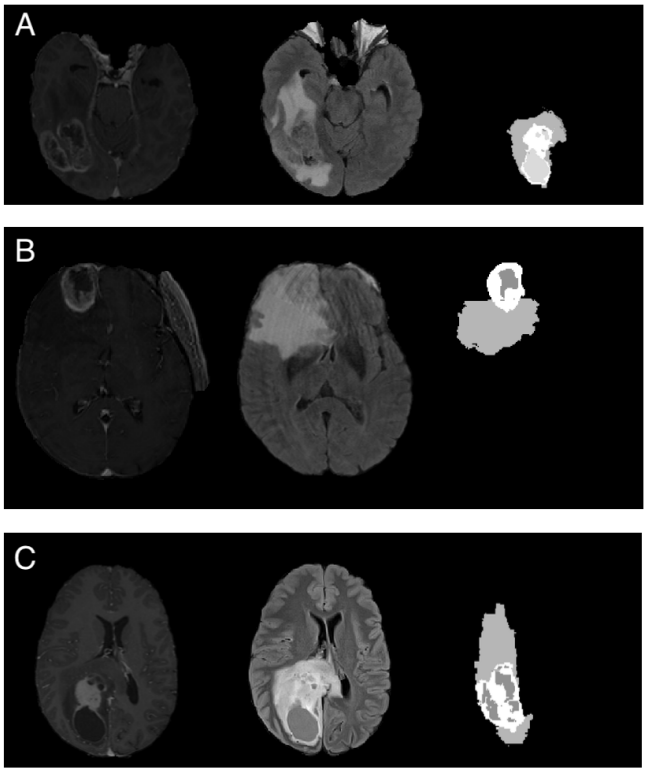

Figure 5.13: Three examples of the GBM data set which OS times were inaccurately predicted by the CNN

5.5 Survival Group Classification of Glioblastoma

The variety and complexity of interactions between clinical, biological and treatment related factors make it difficult to identify accurate overall survival predictions for each individual. Another approach to survival prediction is to be able to reliably differentiate between the groups Longer-Term Survivors (LTS) and Shorter-Term Survivors (STS). As discussed in Section 4.3, a common threshold between LTS and STS is generally set to 24 months. Patients surviving longer than that threshold are often referred to as long-term survivors of glioblastoma multiforme. Hence, an attempt was made in the following sections to simplify the desired task of the CNN that was presented in the last section to a classification problem between long-term and short-term survival in an attempt to improve the prediction model for overall survival of patients with GBM from pre-operative MRI scans.

5.5.1 Image Preprocessing

The two sequences of the pre-operative MRI scans (T1c and FLAIR) were preprocessed following the conventional image pre-processing pipelines presented in Section 4.3. The MRI volumes were skull-stripped, N4 bias field corrected, up- and downsampled to a standardized height and width of 512x512 with a bicubic interpolator and intensity normalized by subtracting the minimal pixel intensity value and dividing by the range of the pixel values. The image information for the CNN was saved in RGB image format consisting of the associated axial T1c, FLAIR and segmentation image in place of the colour channels.

5.5.2 Network Architecture

In the last experiment transfer learning was used to leverage previous learnings and avoid learning the model weights from scratch. This approach is commonly used in state-of-the-art network architectures especially when sample size is low. However, in this experiment transfer learning led to two cases: extreme overfitting or the model only predicting one OS group to achieve the maximal accuracy and the minimal loss. As a result, a different approach was taken. The model was built from scratch. The network architecture consisted of three convolutional blocks consisting of one convolutional layer of kernel size 3 followed by batch normalization, dropout with a probability of 25% for neuron deactivation and lastly, a max pooling layer.

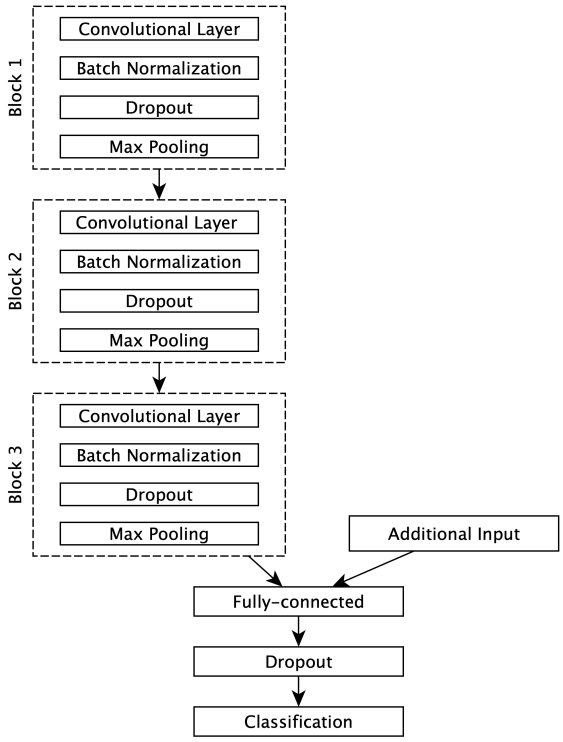

Figure 5.14: Model architecture implemented for the OS group classification using the pre-operative MRI scan and the variable age of patients diagnosed with glioblastoma multiforme

The input of the model consisted of the two MRI sequences and the segmentation mask compiled in an RGB image format and hence, the input was of size 512x512x3. The number of filters used by each convolutional layer increased in each layer by a factor of 2. Hence, the first layer used 8 filters, the second 16 and the last 32 with a kernel size of 3. After the first convolutional layer followed by a Rectified Linear Units layer, the input 512x512x3 had size 510x510x8. Batch normalization was applied to eliminate the internal covariate shift by transforming the data to follow a zero-centered, unit variance distribution as described in Section 3.3.2 and hence facilitate the learning process. The process of batch normalization does not affect the dimension of the tensor. Next, dropout as presented in Section 2.2.5.3 was performed to deactivate a quarter of the trained neurons to tackle the potential of overfitting. This scheme subsequently repeated three times with increasing kernel size of the convolutional layers. At the end of the last max pooling layer, we obtained a 3D tensor of size 62x62x32, which was fed to a global max pooling layer. Global max pooling follows the same principle as global average pooling (see Section 3.3.4 but takes the maximum over each

62x62 feature map. Hence, we get a 1D tensor of size 1x1x32. Just like the CNN used for the linear OS prediction in the last experiment, the architecture for the OS group classification comprised an additional input pathway for the variable age, which was attached to the main image network path after the global max pooling layer. The concatenation of the two data processing paths is followed by a fully-connected layer consisting of 48 units and the ReLU activation function followed by dropout with a deactivation probability of 50%. The final vector fed to the output layer was of size 48. Since the aim of this network structure is the classification of two groups, the output layer used the sigmoid activation function as described in Section 2.2.3. The adaptive learning rate optimization algorithm Adam was chosen as the optimizer with a learning rate of 0.001 and binary crossentropy as the loss function to minimize during the training process.

5.5.3 Training

For this experiment, right-censored patients who survived longer than 24 months were included in the analysis. As a result, we obtained a total of 131 patients in the training and validation procedure with 70% (100 patients) of GBM patients being assigned to the training set and 30% (31 patients) to the validation set. The algorithm was trained on the axial images (1944 images) of the pre-processed MRI volumes. The model was validated on 570 axial images of the MRI sequences. One quarter of the axial images of each MRI volume and sequence was cropped to not bias the results by images containing only small peripheral areas of the tumor.

In order to obtain the best performing model with the given data, the space of possible hyperparameters was explored in a similar fashion to the last experiment. The search for parameters suitable for the task is usually called grid search and involves the manual adaptation of parameters such as the learning rate, weight decay, learning rate decay, amount of dropout, kernel size, stride, number of filters, etc, in order to obtain the best performance in terms of both accuracy and training time.

Again, data augmentation consisted of horizontal flipping, vertical flipping, shifting the height and width by a factor of 0.1, zooming by a factor of 0.1, altering the brightness by a minimum factor of 0 and a maximum factor of 0.5 and rotating the image by a maximum angle of 20 degrees. Points outside the boundaries of the input were filled with zeros (see Figure 5.10).

5.5.4 Results

In the final experiment, the main aim was to facilitate the learning process of the deep learning approach through the discretization of the continuous outcome variable survival. The overall survival of the GBM patients was classified into short-term (< 24 months) and long-term (> 24 months). Again, both MRI sequences (T1c, FLAIR), the 3D segmentation mask and the age of the patients were used to train the prediction model. In total, the medical records and clinical data of 131 patients was used in this experiment. Since the task was reduced to a classification problem of two classes, the binary crossentropy function was used as the cost function. To assess the performance of the deep learning model, the accuracy and the F1-score (defined as the harmonic mean between recall and precision) were computed in each epoch as well.

Convergence of the model was reached after training for 19 epochs. The loss of the training and the test set was 0.85 and 0.56, respectively. The accuracy of the training data achieved a value of 0.52, whereas the test accuracy was higher with a value of 0.7. In addition, the F1-score was computed which is the harmonic mean between the recall and the precision in a binary classification task. The F1-score of the training set was 0.55 and 0.59 for the test set. Possible explanations for the lower loss and higher accuracy in the test set might be due to easier instances in the test cohort or the fact that the small size of the test data affects the range of observed performance. Either way a larger size of the test cohort would yield more precise estimates.

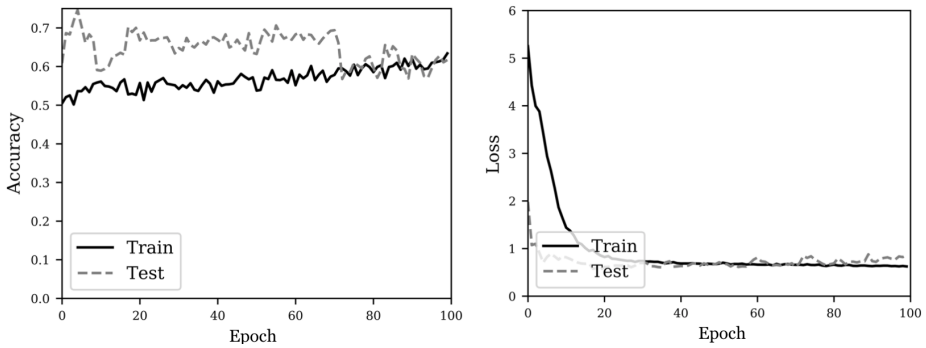

Figure 5.15: Accuracy and loss of the CNN for OS group classification with the additional input age

A common way of illustrating the classification error of the model is the confusion matrix. The columns represent the predicted classes of the images, whereas the

rows refer to the actual classes.

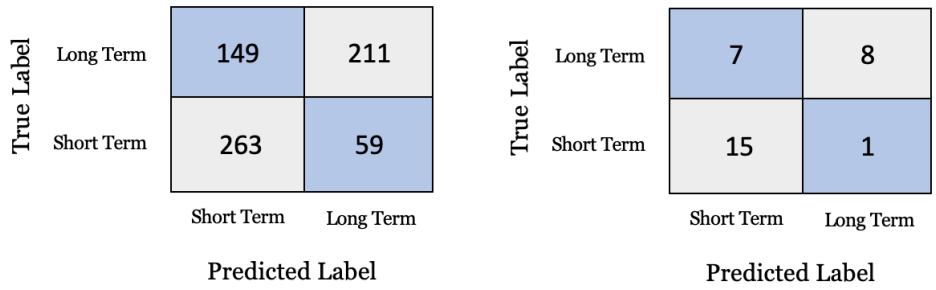

Figure 5.16: Confusion matrix of the OS group classification model using a CNN with the additional input age; results for all axial images in the test set on the right and results averaged for each patient on the left

The confusion matrix of the final CNN design after the last epoch for the validation cohort can be seen in Figure 5.16. On the left side of Figure 5.16, the predicted and actual classifications of each axial image are presented, whereas on the right-hand side we can see the results when the output probability vector is averaged over all axial images belonging to one patient.

Similar to the experiment discussed in Section 5.4, the model had difficulties in correctly classifying long-term survivors. This further supports the assumption that more data is required for the algorithm to extract feature predictive for long-term survivors. From the patients pictured in Figure 5.13, only patient A was assigned to the validation cohort for this experiment. The OS group classification model was also not able to accurately classify patient A into the group of long-term survivors. What's noticeable too is the significantly higher number of correct predictions of the axial images which vanishes when the average for each patient is taken. Since the depth of the MRI scan varied between patients, more axial images can be found for one patient than the other which is why we can see a significant difference from the confusion matrix of the axial image prediction and the averaged prediction. The MRI sequences and the segmentation of the only patient who actual belonged to the group of short-term survivors but was misclassified into the group of long-term survivors can be seen in Figure 5.17.

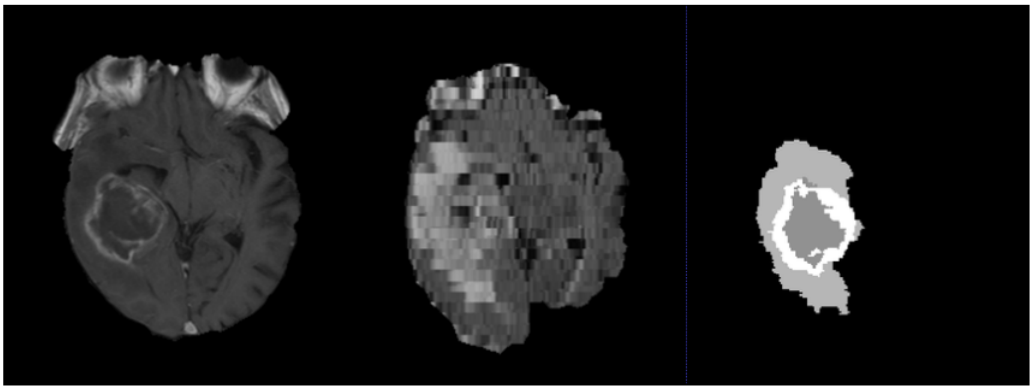

Figure 5.17: The MRI-sequences T1c(left), FLAIR (middle) and the segmentation (right) of the patient who was misclassified as a long-term survivor by the OS group classification CNN

As we can see in Figure 5.17, the pre-processing steps also did not remove the skull on the MRIs of this patient. The non-brain area could not be fully removed without the skull-stripping process also removing brain area which is why segments of the skull and eye region can be seen in Figure 5.17. However, in addition to the leftovers of non-brain fractions, the resolution of the FLAIR sequences was very poor. These two reasons might have led to difficulties in accurately classifying this instance in its respective class.

5.6 Conclusion

We constructed a deep convolutional neural network approach to perform pixel-wise classification to segment T1c MRI images based on pre-operative MRI scans. A series of model configurations and tuning were carried out during the development of the CNN architecture to determine the best performing network architecture for the task at hand. The final convolutional neural network provided a maximal accuracy of 0.95 and an IoU of 0.5 when tested on newly seen data and was even able to detect and segment glioblastoma not visible to the trained eye. One issue that still needs to be solved is the misclassification and uncertainty of some pixels at the edges of the tumor. The segmentation files used as the ground truth were not manually annotated by experts but automatically segmented by the Brain Image Analysis software BraTumIA and were manually corrected slice by slice due to the presence of faulty segmentations. However, faulty segmentation still remained. Hence, the ground truth was affected by errors. Despite this

shortcoming, the algorithm proved its ability to learn and improve.

Next, a deep learning approach for the OS prediction of patients with glioblastoma using different sequences of MRI scans, the segmentation and the additional prognostic variable age was presented. Although accurate predictions of patient survival were not possible given the limited available clinical data of each patient, we showed the potential of deep learning and the inclusion of imaging data can offer. The prediction CNN (predCNN) only achieved a validation loss of 9.8 and experienced difficulties in the prediction of long-term survivors. To facilitate the prediction problem, an attempt was made to simplify the task for the CNN to a binary classification problem by grouping the GBM patients into two classes: short-term (< 24 months) and long-term (≥ 24 months) survivors. Interestingly, the approach of transfer learning which helped in preventing extreme overfitting in the predCNN model, did not yield stable models for the classification model. The final CNN for overall survival group classification (classCNN) was build and trained from scratch, since it did not benefit from the pretrained weights of ResNet-18. The classCNN model yielded a loss of 0.56 and an accuracy of 0.7. Similar to the predCNN model, the classCNN faced problems in the accurate classification of long-term survivors. The GBM data set consists of more than twice as much short-term survivors (95 patients) than long-term survivors (41) which reinforces the assumption that both the predCNN as well as the classCNN model would benefit from more data and more balanced data regarding short- and long-term survival.

In the end, both prediction networks are were prone to overfitting visible in Figure 5.11 and 5.15. Both of the prediction models, predCNN and classCNN, are not of immediate clinical use, but clearly demonstrate the capability of these deep networks even when trained on a small data set consisting of noisy MRI scans with barely any additional clinical information. Pre-operative MRI scans cannot capture important prognostic factors for the prognosis of the patient such as the Karnofsky performance score and the extent of the subsequent resection but contain on themselves meaningful information that in some cases be sufficient to predict accurate overall survival of the patient.

Chapter 6

Discussion

In this master thesis, algorithms and approaches for the understanding, detection and survival prediction of glioblastoma were applied to explore the potential, radiomics can contribute to future research in the diagnosis and prognosis of patients with glioblastoma multiforme. In particular, the aim of this work was to establish a reliable algorithm able to detect and segment glioblastoma multiforme within a pre-operative MRI scan of a patient. Further, we proposed two deep learning algorithms using convolutional neural networks for learning high-level features from MRIs to predict the overall survival (OS) time for glioblastoma patients and a second deep learning approach that is able to distinguish between long-term survivors and short-term survivors at the time of undergoing magnetic resonance imaging. Both approaches were trained in a supervised manner. We examined various approaches to find a suitable model for survival prediction and employed a rigorous literature search to achieve the best possible results within the time available.

The first part of the work focused on the development of an automatic segmentation algorithm able to detect and segment glioblastoma from pre-operative MRI data. Manual segmentation of MRI tumor images is a tedious and challenging task due to the irregular nature of tumors but of tremendous importance for clinical decision-making. In addition to the time-consuming task for one individual expert, the segmentation might be shaped and sized according to subjective standards. This underlines the importance of automatic segmentation procedures. Therefore, we present a fully-automatic deep learning based approach using convolutional neural networks able to perform brain tumor segmentation based on axial images of MRIs. The segmentation model achieved an accuracy of 95.1%

and an intersection over union score of 0.5 when assessed on the validation cohort and showed good performance when the results were examined graphically. Small uncertainties at the edges of the tumorous tissue and lack of confidence in some instances might be improved by the acquisition of more data or the presence of a ground truth that was manually segmented by experts and not itself segmented by an automatic software.

In the second part of this thesis, we aimed at exploring the potential of predicting patient overall survival at the time of the pre-operative MRI scan solely based on the MRI images in combination with the only fully available predictive variable of glioblastoma survival namely age as an additional input. Our findings highlight a common opinion in the scientific literature that overall survival in GBM patients is heterogeneous, influenced by multiple clinical, demographic and biological factors and cannot be solely based on imaging data alone. While neither of the two proposed deep learning algorithms were able to accurately predict overall survival based on the images alone, the algorithm was successful at identifying subsets of patients who experienced significantly longer or shorter survival. A significant improvement for the linear prediction model (predCNN) was observed through transfer learning. The pretrained ResNet-18 model seemed somewhat promising in dealing with the low amounts of data but did not yield similar results for the OS group classification task. Furthermore, it must be noted that the model mentioned in this work were not cross validated. Instead we trained on one part of the data, and tested on different part of the GBM data set, analogous to a single fold of cross-validation. This was done due to time and computational constraints, since cross-validation of several deep learning models is computationally expensive and requires a lot of time.

Nevertheless, we saw that image-based survival analysis has the potential to contribute additional information to enhance the accuracy of survival prediction in patients with glioblastoma in the future. More experiments have to be conducted to further improve the performance of the model. For example, the network could be extended to include further clinical features known to be predictive for survival such as the Karnofsky performance score and the extent of resection. However, these features were only partially available at the time of this work. Pre-operative images might not comprise sufficient information to exactly pinpoint the time a diagnosed patient has left to live, but the results of the model show that a trend can be derived solely based on the age and the pre-operative MRI of a patient.

Despite the extensive amount of work dedicated to brain tumor segmentation and prediction, the problem remains complex and results are often not satisfactory. Lack of medical data is one of the main problems that is commonly encountered in radiomics. Finding the right network configuration to work with such a small amount of data was a major challenge during the course of this work. We explored different networks and configurations to find the most suitable network. Designing a CNN includes tuning a large set of hyperparameters such as, the proper selection and order of the different layers, the number of filters, the size of its receptive field, the appropriate optimizer and its learning rate. Thus, sufficient large data sets are a necessary requirement. In addition, many methods require the use of different image modalities which are often not systematically available in a clinical multi-center setting. Another challenge in the given data apart from the small cohort size was the varying resolution and size of the MRIs between the MRI scanning centres. High resolution images can facilitate the learning of predictive patterns associated with short and long survival times. In addition, noise might be present in the images by patients moving slightly while taking the MRI which has to be accounted for in image pre-processing. Otherwise, the model might be affected by random intensity fluctuations within the MRI images.

Deep learning algorithms are said to eliminate the need for feature engineering, whereas manual feature engineering by experts can introduce man-made errors. However, when the resolution of the available MRI images is poor, neither an expert nor the algorithm will yield sufficient results. The robustness of the learned features relies heavily on the image quality which varies across MR imaging centers. Neural networks based on non-imaging data have shown the ability to outperform classical survival models in the past (Luck et al., 2017; Lee et al., 2018; Katzman et al., 2018). Nonetheless, machine learning approaches for imaging data still require additional input for more complex problems in the medical field. Therefore, the power of deep learning models can hopefully be exploited to its full potential in the future by combining medical high-resolution images with additional clinical data or features motivated by biology to enhance the overall performance of survival prediction models.

Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

CNN Convolutional Neural Network

CT Computer Tomography

GBM Glioblastoma Multiforme

IoU Intersection Over Union

KPS Karnofsky Performance Scale

LASSO Least Absolute Shrinkage and Selection Operator

MAE Mean Absolute Error

MRI Magnetic Resonance Imaging

MSE Mean Squared Error

NN Neural Network

oPA Overall Pixel Accuracy

OS Overall Survival

PH Proportional Hazards

RCT Randomized Clinical Trial

SGD Stochastic Gradient Descent

List of Figures

List of Tables

Bibliography

Adamson, C., Kanu, O. O., Mehta, A. I., Di, C., Lin, N., Mattox, A. K., and Bigner, D. D. (2009). Glioblastoma multiforme: a review of where we have been and where we are going. *Expert Opinion on Investigational Drugs*, 18(8):1061–1083.

Adeberg, S., Bostel, T., König, L., Welzel, T., Debus, J., and Combs, S. E. (2014). A comparison of long-term survivors and short-term survivors with glioblastoma, subventricular zone involvement: a predictive factor for survival? *Radiation Oncology*, 9(1):95.

Ball, J., Balogh, E., and Miller, B. T. (2015). *Improving diagnosis in health care*. National Academies of Sciences, Engineering, and Medicine, Washington, DC.

Burton, E. C., Lamborn, K. R., Feuerstein, B. G., Prados, M., Scott, J., Forsyth, P., Passe, S., Jenkins, R. B., and Aldape, K. D. (2002). Genetic aberrations defined by comparative genomic hybridization distinguish long-term from typical survivors of glioblastoma. *Cancer Research*, 62(21):6205–6210.

Casamitjana, A., Puch, S., Aduriz, A., and Vilaplana, V. (2016). 3D convolutional neural networks for brain tumor segmentation: a comparison of multi-resolution architectures. *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries Lecture Notes in Computer Science*, page 150–161.

Chollet, F. (2015). Keras documentation. `https://keras.io/`. [Online; accessed April 28, 2019].

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., Phillips, S., Maffitt, D., Pringle, M., et al. (2013). The cancer imaging archive (tcia): maintaining and operating a public information repository. *Journal of Digital Imaging*, 26(6):1045–1057.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.

Colman, H., Zhang, L., Sulman, E. P., McDonald, J. M., Shooshtari, N. L., Rivera, A., Popoff, S., Nutt, C. L., Louis, D. N., Cairncross, J. G., et al. (2010). A multigene predictor of outcome in glioblastoma. *Neuro-Oncology*, 12(1):49–57.

Gillies, R. J., Kinahan, P. E., and Hricak, H. (2015). Radiomics: images are more than pictures, they are data. *Radiology*, 278(2):563–577.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT Press Cambridge.

Graupe, D. (2013). *Principles of artificial neural networks*, volume 7. Singapore: World Scientific Press.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377.

Hanif, F., Muzaffar, K., Perveen, K., Malhi, S. M., and Simjee, S. U. (2017). Glioblastoma multiforme: A review of its epidemiology and pathogenesis through clinical presentation and treatment. *Asian Pacific Journal of Cancer Prevention: APJCP*, 18(1):3.

Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H. (2017). Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV.

Higano, S., Shrier, D. A., Numaguchi, Y., Shibata, D. K., and Kwok, E. (2000). Characteristics and pitfalls of contrast-enhanced, t1-weighted magnetization transfer images of the brain. *Academic Radiology*, 7(3):156–164.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Holland, E. C. (2000). Glioblastoma multiforme: the terminator. *Proceedings of the National Academy of Sciences*, 97(12):6242–6244.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):24.

Kickingereder, P., Bonekamp, D., Nowosielski, M., Kratz, A., Sill, M., Burth, S., Wick, A., Eidel, O., Schlemmer, H.-P., Radbruch, A., Debus, J., Herold-Mende, C., Unterberg, A., Jones, D., Pfister, S., Wick, W., von Deimling, A., Bendszus, M., and Capper, D. (2016). Radiogenomics of glioblastoma: Machine learning-based classification of molecular characteristics by using multiparametric and multiregional mr imaging features. *Radiology*, 281(3):907–918.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klughammer, J., Kiesel, B., Roetzer, T., Fortelny, N., Nemc, A., Nenning, K.-H., Furtner, J., Sheffield, N. C., Datlinger, P., Peter, N., et al. (2018). The dna methylation landscape of glioblastoma disease progression shows extensive heterogeneity in time and space. *Nature Medicine*, 24(10):1611.

Lacroix, M., Abi-Said, D., Fourney, D. R., Gokaslan, Z. L., Shi, W., DeMonte, F., Lang, F. F., McCutcheon, I. E., Hassenbusch, S. J., Holland, E., et al. (2001). A multivariate analysis of 416 patients with glioblastoma multiforme: prognosis, extent of resection, and survival. *Journal of Neurosurgery*, 95(2):190–198.

Lao, J., Chen, Y., Li, Z.-C., Li, Q., Zhang, J., Liu, J., and Zhai, G. (2017). A deep learning-based radiomics model for prediction of survival in glioblastoma multiforme. *Scientific Reports*, 7(1):Article ID 10353.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Taipei, Taiwan.

Lee, C., Zame, W. R., Yoon, J., and van der Schaar, M. (2018). Deephit: A deep learning approach to survival analysis with competing risks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Legendre, A. M. (1805). *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot. Courcier, Paris.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Lotan, E., Jain, R., Razavian, N., Fatterpekar, G. M., and Lui, Y. W. (2019). State of the art: Machine learning applications in glioma imaging. *American Journal of Roentgenology*, 212(1):26–37.

Luck, M., Sylvain, T., Cardinal, H., Lodi, A., and Bengio, Y. (2017). Deep learning for patient-specific kidney graft survival analysis. *arXiv preprint arXiv:1705.10245*.

Macyszyn, L., Akbari, H., Pisapia, J. M., Da, X., Attiah, M., Pigrish, V., Bi, Y., Pal, S., Davuluri, R. V., Roccograndi, L., et al. (2015). Imaging patterns predict patient survival and molecular subtype in glioblastoma via machine learning techniques. *Neuro-Oncology*, 18(3):417–425.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., Burren, Y., Porz, N., Slotboom, J., Wiest, R., et al. (2015). The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024.

Minsky, M. and Papert, S. (1969). Perceptrons: An essay in computational geometry. Cambridge, MA: MIT Press.

Mitchell, T. M. (1997). Does machine learning really work? *AI magazine*, 18(3):11–11.

Mobadersany, P., Yousefi, S., Amgad, M., Gutman, D. A., Barnholtz-Sloan, J. S., Vega, J. E. V., Brat, D. J., and Cooper, L. A. (2018). Predicting cancer outcomes from histology and genomics using convolutional networks. *Proceedings of the National Academy of Sciences*, 115(13):E2970–E2979.

Naydenov, E., Tzekov, C., Minkin, K., Nachev, S., Romansky, K., and Bussarsky, V. (2011). Long-term survival with primary glioblastoma multiforme: a clinical study in bulgarian patients. *Case Reports in Oncology*, 4(1):1–11.

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning*, page 78. Association for Computing Machinery (ACM). Banff, Alberta, Canada.

Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition*, pages 427–436. Boston, MA, USA.

Nie, D., Zhang, H., Adeli, E., Liu, L., and Shen, D. (2016). 3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 212–220. Springer. Athens, Greece.

Nigro, J. M., Misra, A., Zhang, L., Smirnov, I., Colman, H., Griffin, C., Ozburn, N., Chen, M., Pan, E., Koul, D., et al. (2005). Integrated array-comparative genomic hybridization and expression array profiles identify clinically relevant molecular subtypes of glioblastoma. *Cancer Research*, 65(5):1678–1686.

Oken, M. M., Creech, R. H., Tormey, D. C., Horton, J., Davis, T. E., Mcfadden, E. T., and Carbone, P. P. (1982). Toxicity and response criteria of the eastern cooperative oncology group. *American Journal of Clinical Oncology*, 5(6):649–656.

Pereira, S., Pinto, A., Alves, V., and Silva, C. A. (2016). Brain tumor segmentation using convolutional neural networks in MRI images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251.

Piccolo, S. R. and Frey, L. J. (2013). Clinical and molecular models of glioblastoma multiforme survival. *International Journal of Data Mining and Bioinformatics*, 7(3):245.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer. Munich, Germany.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386.

SEER (2007). Seer*stat release 6.3.5. `http://seer.cancer.gov/seerstat/software`. [Online; accessed April 28, 2019].

Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press. New York, USA.

Smoll, N. R., Schaller, K., and Gautschi, O. P. (2013). Long-term survival of patients with glioblastoma multiforme (GBM). *Journal of Clinical Neuroscience*, 20(5):670–675.

Song, S., Zheng, Y., and He, Y. (2017). A review of methods for bias correction in medical images. *Biomedical Engineering Review*, 1(1).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312.

Tandel, G. S., Biswas, M., G Kakde, O., Tiwari, A., S Suri, H., Turk, M., Laird, J. R., Asare, C. K., A Ankrah, A., N Khanna, N., et al. (2019). A review on a deep learning perspective in brain cancer classification. *Cancers*, 11(1):111.

Thakkar, J. P., Dolecek, T. A., Horbinski, C., Ostrom, Q. T., Lightner, D. D., Barnholtz-Sloan, J. S., and Villano, J. L. (2014). Epidemiologic and molecular prognostic review of glioblastoma. *Cancer Epidemiology and Prevention Biomarkers*, 23(10):1985–1996.

Tustison, N. J., Avants, B. B., Cook, P. A., Zheng, Y., Egan, A., Yushkevich, P. A., and Gee, J. C. (2010). N4itk: improved n3 bias correction. *IEEE Transactions on Medical Imaging*, 29(6):1310.

Vaillant, R., Monrocq, C., and Le Cun, Y. (1994). Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250.

Vial, A., Stirling, D., Field, M., Ros, M., Ritz, C., Carolan, M., Holloway, L., and Miller, A. A. (2018). The role of deep learning and radiomic feature extraction in cancer-specific predictive modelling: a review. *Translational Cancer Research*, 7(3):803–816.

Zhang, Q., Zhang, M., Chen, T., Sun, Z., Ma, Y., and Yu, B. (2019). Recent advances in convolutional neural network acceleration. *Neurocomputing*, 323:37–51.

Zhang, W., Li, R., Deng, H., Wang, L., Lin, W., Ji, S., and Shen, D. (2015). Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage*, 108:214–224.

Zhao, W. (2017). Research on the deep learning of the small sample data based on transfer learning. In *AIP Conference Proceedings*, volume 1864, pages 020018–1 – 020018–8. AIP Publishing.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: series B (Statistical Methodology)*, 67(2):301–320.

Appendix

A code example to illustrate the simple implementation of a small CNN in Python using the Keras API is presented.

```
# Load libraries
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, MaxPooling2D

# Define CNN model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
        input_shape=(100, 100, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(units=32, activation='relu', input_dim=100))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(optimizer='Adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Generate 1000 dummy RGB images of size 100x100
img_data = np.random.random((1000, 100, 100, 3))

# Generate binary dummy labels for images
labels = np.random.randint(2, size=(1000, 1))

# Train the model, iterating on the data in batches of 32 samples
model.fit(img_data, labels, epochs=10, batch_size=32)
```