DHBW
Duale Hochschule
Baden-Württemberg

Deep Learning auf medizinischen Bilddaten

Studienarbeit

des Studiengangs Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg **Mosbach Campus Bad Mergentheim**

von

Patrick Arndt

Mai 2023

Matrikelnummer, Kurs	7414700, INF20
Gutachter	Malte Tölle

Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema *Deep Learning auf medizinischen Bilddaten* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Eckartshausen, Mai 2023

Patrick Arndt

Abstract

Inhaltsverzeichnis

Glossar

Algorithmus

Ein Algorithmus ist eine genau definierte Folge von Aktionen, für die Lösung eines Problems[vgl. 1].

Batch

Definiert die Anzahl an Trainingsdaten, die durch das Neuronale Netz gereicht werden, bevor die interenen Parameter in diesem angepasst werden.

Ionisierende Strahlung

Strahlung die so stark ist, das sie Elektronen aus Atomen oder Molekühlen entfernt kann, sodass diese positiv geladen sind (ionisiert). Diese Art von Strahlung kann Mensch und Umwelt schädigen.[vgl. 2].

Modell

Ein Modell beschreibt eine komplexe mathematische Funktion, das anhand von Daten trainiert wurde, um bestimmte Vorhersagen oder Entscheidungen zu treffen. [vgl. 3].

Proton

Ein stabiles, elektrisch positiv geladenes Teilchen.

Abkürzungsverzeichnis

BraTS	Brain Tumor Segmentation Challenge
CNN	Convolutional Neural Network
KI	Künstliche Intelligenz
KNN	Künstliches Neuronales Netz
MRT	Magnetresonanztomographie
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Einführung und Hintergrund

In der Medizin gibt es schon seit mehr als 50 Jahren bildgebende Verfahren, mit denen Aufnahmen vom Inneren des menschlichen Körpers möglich sind. Diese Verfahren wurden kontinuierlich weiterentwickelt und sind inzwischen in der Lage detaillierte Bilder zu erzeugen. Technologien wie eine Computertomographie (CT) oder eine MRT sind in der Lage präzise Informationen über innere Organe, Strukturen und Gewebe im Körper zu liefern. [vgl. 4] Mit den modernen bildgebenden Verfahren, steigt die Datenmenge und Komplexität, wie auch die Herausforderung der Analyse und Verarbeitung der Daten bzw. Bilder.

Aufgrund der immer weiter fortschreitenden Technologien, setzen immer mehr medizinische Institutionen auf bildgebende Verfahren, um so Krankheiten zu diagnostizieren und behandeln. Allerdings ist die Analyse und Verarbeitung dieser Daten mit der Zeit komplexer geworden, sodass Ärzte und Mediziner vor eine Herausforderung gestellt werden. Die manuelle Analyse und Verarbeitung der Daten bringt oft Schwierigkeiten und Fehlerquellen mit sich. Der Prozess ist meist sehr zeitintensiv, weshalb es häufig zu langen Wartezeiten bei Auswertungen für die Patienten kommen kann. Nicht nur die Wartezeit ist ein Problem, sondern auch menschliche Fehler bei Interpretation der Daten, welche zu Fehldiagnosen oder -behandlungen führen können. [vgl. 5]

Diese Herausforderungen können mithilfe des vielversprechenden Ansatzes der Automatisierung gelöst werden. So kann bspw. die Effizienz gesteigert und die Fehlerrate gesenkt werden. Ein wichtiges Stichwort ist hierbei Künstliche Intelligenz (KI), die ebenfalls in den vergangenen Jahren enorme

Fortschritte gemacht hat. Insbesondere im Bereich der Bildanalyse haben KI-Systeme durch die Entwicklung von Deep Learning-Modellen enorm an Genauigkeit gewonnen. Die Anwendung von Deep Learning auf medizinische Bilddaten bietet vielversprechende Möglichkeiten für eine verbesserte Diagnose, Therapieplanung und -überwachung sowie für die Entwicklung von personalisierten Medizinlösungen.

Besonders im medizinischen Bereich hat die Nutzung von KI-Systemen auf medizinischen Bilddaten in den letzten Jahren an Bedeutung gewonnen. Mithilfe von Deep Learning-Modellen können Ärzte und Forscher komplexe Bilder von medizinischen Bildgebungsverfahren wie Röntgenaufnahmen, CT-Scans und MRT-Scans automatisiert analysieren. Diese Analyse kann zur Erkennung von Krankheiten, Identifizierung von Tumoren, sowie zur Bewertung der Therapieeffektivität eingesetzt werden.

1.2. Problemstellung und Zielsetzung

Die zunehmende Komplexität und Menge an medizinischen Daten und Bildern stellt Mediziner vor große Herausforderungen. Die manuelle Analyse und Verarbeitung der Daten ist zeitaufwendig, fehleranfällig und es besteht die Gefahr wichtige Sachen zu übersehen, welche für die Diagnose nützlich sind. Gerade im Bereich der Tumore, also der Krebszellen, ist die frühzeitige Erkennung aber auch die Behandlung sehr wichtig für das Wohl des Patienten.

Vor allem im Bereich des Gehirns ist mit Tumoren sehr behutsam umzugehen, um so keine bleibenden Schäden beim Patienten zu hinterlassen. Bei Gehirntumoren ist es wichtig die genaue Position und Lage zu kennen, damit eine geeignete Behandlung gefunden werden kann. Die Operation bei einem Gehirntumor bedarf großer Vorbereitung und ist komplex, da Fachärzte zunächst den Tumor genau segmentieren müssen, bevor sie weitere Schritte planen. Diese Vorbereitung ist unter anderem zeitintensiv und kann nur von ausgebildeten Fachärzten übernommen werden.

Befindet sich ein Tumor im Gehirn, so bedarf es einer noch genaueren Vorbereitung und Arbeit, um diesen zu entfernen. Krebs ist ein vielseitiges Thema, mit dem sich die Medizin seit Jahren beschäftigt. Heutzutage setzen Ärzte vor allem bei der frühzeitigen Erkennung von Tumoren auf die Hilfe von KI. Doch nicht nur bei der Früherkennung, sondern auch bei der Behandlung solcher Tumore. Sind solche im Bereich des Gehirns, ist es wichtig behutsam vorzugehen, um so keine bleibenden Schäden zu verursachen.

Die Problemstellung dieser Studienarbeit ist die Identifizierung und Klassifizierung von Tumoren im Gehirn, genauer gesagt eine Multi Klassen Segmentierung, bei welcher jeder Bildpunkt eine Tumor Klasse zugewiesen bekommt. Die verwendeten Daten sind aus dem öffentlichen Datensatz BraTS (Brain Tumor Segmentation Challenge), welcher aus rund 1250 MRT-Bildern von Gehirnen besteht, welche von Fachärzten auf Tumore untersucht und wenn vorhanden markiert wurden. Eine Herausforderung dabei ist es, ein geeignetes Deep Learning Modell zu entwickeln, dass für die Segmentierung von Tumoren im Gehirn geeignet ist und möglichst genaue Ergebnisse liefert.

Die Zielsetzung dieser Arbeit ist die Erstellung eines Modells für die Segmentierung von Gehirntumoren, anhand des BraTS Datensatzes. Das Modell soll in der Lage sein zwischen drei verschiedenen Tumor-Klassen zu unterscheiden und für diese eine Maske zu erstellen. Für die Entwicklung des Modells wird das Open Source Machine-Learning Framework PyTorch verwendet, welches von Facebook entwickelt wurde. Das Framework bietet zahlreiche Tools und Funktionen für das Entwickeln von Deep Learning-Modelle. [vgl. 6]

2. Theoretische Grundlagen

2.1. Künstliche Intelligenz

KI wird zunehmend in verschiedenen Bereichen eingesetzt und erleichtert die Arbeit der Menschen enorm. Viele Arbeiten können mithilfe von KI schneller und besser erledigt werden. Im nachfolgenden Abschnitt werden die Begriffe "Intelligenz" und "Künstliche Intelligenz" erklärt, um ein einfaches Verständnis dafür zu schaffen. Es werden außerdem die verwandten Begriffe Machine Learning und Deep Learning in den Kontext eingeordnet.

In der Psychologie bezieht sich Intelligenz auf die Fähigkeit logische, sprachliche, mathematische oder sinnliche Probleme zu lösen, es besteht jedoch keine allgemeine Definition darüber wie Intelligenz definiert ist. Der Mensch kann sich durch seine kognitiven Fähigkeiten, Beschreibungen oder Erklärungen von Sachen merken, welcher er zu einem späteren Zeitpunkt abrufen und verwenden kann. Das menschliche Gehirn, welches aus Milliarden von Neuronen besteht, lernt dabei gewisse Strukturen, Konzepte und Fähigkeiten kennen und speichert diese ab. Beim Lernen der Informationen werden Verbindungen zwischen den Neuronen im Gehirn gestärkt. Je öfter eine Sache gelernt wird, desto mehr stärken sich gewisse Verbindungen von Neuronen im Gehirn. [vgl. 7, 8]

Die Künstliche Intelligenz hingegen, ist ein wissenschaftlicher Bereich, der sich damit befasst, den Computern das menschliche Verhalten anzueignen. Gemeint ist hierbei, dass ein Computer darauf trainiert wird in der Lage zu sein, ähnlich wie ein Menschen zu denken und darauf folgende Aktionen auszuführen. [vgl. 9] Anders als bei biologischen Neuronen, lösen Computer Probleme wie die Klassifizierung eines Bildes durch die Anwendung von mathematischen Funktionen und Algorithmen. Man versucht die Fähigkeit

des Denkens vom Menschen zu imitieren, um so einen Mehrwert bei Lösung von Problemen mit Computern zu bewirken. Mit Hilfe der künstlichen Intelligenz sind Computer heutzutage in der Lage Probleme zu lösen, die vorher ein höheres intellektuelles Verständnis erforderten. Computer können so bspw. Bilder klassifizieren oder auch Vorhersagen treffen. Ähnlich wie beim Menschen, lernen Computer durch verfügbare Daten und stärken dadurch bestimmte mathematische Verbindungen. [vgl. 10]

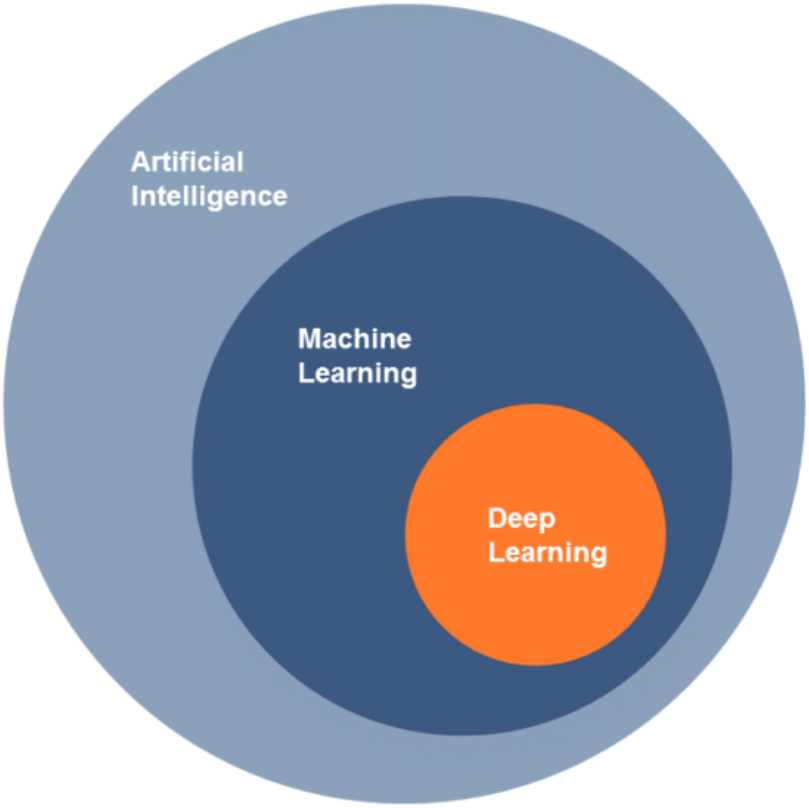


Abbildung 2.1.: Übersicht der Bereiche von künstlicher Intelligenz (Quelle: https://www.alexanderthamm.com/de/blog/ki_artificial-intelligence-ai-kuenstliche-neuronale-netze-machine-learning-deep-learning/)

Im Kontext der KI ist oft auch die Rede von Machine Learning, Deep Learning und Neuronalen Netzen. KI ist der Überbegriff, welcher sowohl Machine Learning als auch Deep Learning beinhaltet. Deep Learning ist eine Teilbereich des Machine Learnings, in welchem die künstlichen Neuronalen Netze zum Einsatz kommen, mit denen Computer in der Lage sind Dinge zu lernen und Probleme zu lösen. Nachfolgend liegt der Fokus auf dem Bereich des Deep Learnings.

2.2. Deep Learning

2.2.1. Einführung

Viele Probleme oder Aufgaben können heutzutage mit einem Computer effizient gelöst werden. Dazu wird in der Regel ein Algorithmus verwendet, der die Aufgabe systematisch löst. Es gibt jedoch komplexe Aufgaben, bei denen ein Algorithmus nicht im Stande ist diese zu lösen.

Ein Beispiel dafür ist, vorherzusagen was ein Kunde in nächster Zeit kaufen wird, um ihm passende Vorschläge zu unterbreiten. Probleme dieser Art können nicht mit einem Algorithmus gelöst werden, sondern die Lösung muss anhand von vorhandenen Daten bestimmt werden. In solchen Fällen soll der Computer selbst einen Algorithmus erstellen, welcher dann auf neue Daten angewendet werden kann. Der Computer lernt mithilfe bereits vorhandener Daten oder Erfahrungen einen Algorithmus zu extrahieren, hierbei werden unter anderem Muster und Strukturen erkannt. Mit dem extrahierten Algorithmus kann anschließend das Kaufverhalten des Kunden approximiert werden. Möglich wird dies durch die enormen Datenmengen, die sich in den letzten Jahren angesammelt haben. Bei jedem Online Einkauf, Webseitenaufruf oder beim Öffnen einer App werden Daten produziert und gesammelt. Diese Daten können letztendlich dazu verwendet werden, um Probleme wie oben beschrieben zu lösen.[vgl. 11]

Aufgaben wie das Klassifizieren von Bildern, komponieren von Musik[1] oder auch das erzeugen von Daten anhand eines Textes sind heutzutage mit Deep Learning möglich. Es gibt drei verschiedene Hauptarten wie ein Computer lernen kann, diese sind in den Abschnitten 2.2.2, 2.2.3 und 2.2.4 beschrieben.

[1] siehe [12]

2.2.2. Überwachtes Lernen

Überwachtes Lernen ist ein wichtiger Bereich des maschinellen Lernens, bei dem ein Modell anhand von beschrifteten (oder "gelabelt") Daten lernt. Beschriftet bedeutet, dass den Daten bereits das gewünschten Ergebnis oder der erwartete Wert zugeordnet ist. Ziel des überwachten Lernens ist es, einen Algorithmus oder ein Modell zu finden, welches für neue und ungesehene Daten die richtigen Beschriftungen liefert.[vgl. 13, 14]
Das Überwachte Lernen beinhaltet zwei verschiedene Problemstellungen, die Klassifikation und die Regression. Bei einer Klassifikation bekommen Bilder eine oder auch mehrere Klassen zugeordnet, es können aber auch Texte, Videos oder andere Dinge klassifiziert werden. Ein Beispiel dafür wäre die Klassifizierung von Hunden und Katzen auf Bildern. Dabei wird eine große Anzahl an Bildern von Hunden und Katzen in das Modell gereicht und dieses fügt den Bildern jeweils die richtigen Klassen hinzu. Sieht das Modell z.B. ein Bild mit einem Hund, so kann es nach dem Training die Klasse "Hund" zuordnen. In den meisten Email Programmen wird ein solches Klassifikationsmodell verwendet, um von seriösen und Spam Emails zu unterscheiden. Das Ergebnis einer Klassifikation ist immer eine Kategorie, welche die zu klassifizierenden Daten am besten beschreibt.
Die zweite Problemstellung, die es beim Überwachten Lernen gibt ist die Regression, diese wird oft für Vorhersagen und Prognosen verwendet. Sie ist ähnlich zur Klassifizierung, jedoch ist hier die Zielmenge eine andere. Es wird versucht Werte für einen meist kontinuierlichen Bereich vorherzusagen,dazu wird der Datensatz mittels einer Funktion approximiert. Das Ergebnis einer Regression sind Werte bzw. Punkte, welche am besten die Funktion beschreiben.[vgl. 13]
Die Studienarbeit befasst sich nachfolgend hauptsächlich mit dem Überwachten Lernen, genauer gesagt mit der Klassifikation. Hierbei wird jedoch kein komplettes Bild klassifiziert, wie im oben genannten Beispiel, sondern es werden Klassen auf die einzelnen Pixel im Bild zugewiesen.

2.2.3. Unüberwachtes Lernen

Beim Unüberwachten Lernen besitzen Daten keine Beschriftungen oder Kategorien, wie es beim Überwachten Lernen der Fall ist. Diese Methode des maschinellen Lernens kommt zum Einsatz, wenn das Beschriften der Daten nicht ohne weiteres möglich ist. Das Modell sucht innerhalb der Daten, ohne vorgegebene Kategorien, nach Mustern und Strukturen. Anhand von statistischen Methoden wird die beste Kategorisierung der Daten gebildet. Die Ergebnisse des Unüberwachten Lernens sind verschiedene Gruppen von Daten, die allerdings keine Bezeichnung besitzen, sondern durchnummeriert werden. Die verschiedenen Daten in den Gruppen haben ähnliche oder die gleichen Eigenschaften.
Ein gutes Beispiel ist der Teil "Kunden kauften auch" auf Amazon. Hier werden dem Kunden ähnliche Produkte vorgeschlagen oder Artikel die oft in Kombination gekauft werden. Es wird dabei keine bestimmte Kategorie von Produkten genannt, sondern nur ähnliche Produkte vorgeschlagen. Es ist schwierig die Ergebnisse des Unüberwachten Lernens zu beurteilen, da keine vergleichbaren Daten mit entsprechenden Beschriftungen vorliegen und so auch kein Fehler berechnet werden kann.[vgl. 9, 13]

2.2.4. Bestärkendes Lernen

Es gibt oft Probleme, bei denen nicht genau bestimmt werden kann, ob etwas richtig oder falsch ist. Die Daten bei derartigen Problemen, insofern welche vorliegen, besitzen deshalb keine Beschriftungen. Es ist allerdings möglich zu sagen, ob Ergebnisse geeignet oder ungeeignet für die Lösung des Problems sind, in diesen Fällen kommt das Bestärkte Lernen zum Einsatz.Im Bereich der Robotik ist Bestärktes Lernen ein klassischer Anwendungsfall. Für Roboter sind in der Regel keine Trainingsdaten verfügbar, daher muss dieser eigenständig lernen, was gut und schlecht ist. [vgl. 7]
Das Ziel des Roboters ist es eine geeignete Lösung bzw. Strategie für das Problem zu finden. Beim Bestärkten Lernen ist oft die Rede von sogenannten

Agenten, die für eine Lösung eines Problems trainiert werden. Bei der Suche nach einer geeigneten Strategie, werden diese kontinuierlich belohnt oder bestraft für gute und schlechte Aktionen. Ein einfaches Beispiel wäre ein Saugroboter,dieser saugt in einem Hotel die Zimmer. In den Zimmern, in welchen der Roboter z.B. nichts umgestoßen hat, wird er belohnt. Stößt der Roboter Dinge um, oder saugt etwas nicht gewolltes ein, so wird er bestraft. [vgl. 13] Das Belohnungssystem kann auf Punkten basieren, so würde der Saugroboter im positiven Fall Pluspunkte und im negativen Fall Minuspunkte bekommen und anhand dessen seine Aktionen anpassen.

2.2.5. Künstliches Neuronales Netz (KNN)

Die Struktur und Funktion des menschlichen Gehirns ist das Vorbild der künstlichen Neuronalen Netze. Sie besitzen eine Vielzahl künstlicher Neuronen, die miteinander verbunden sind und Informationen verarbeiten. Ein Künstliches Neuronales Netz (KNN) wird für verschiedene Anwendungen eingesetzt, wie z.B. Mustererkennung, Datenanalyse oder Vorhersagen.

2.2.5.1. Biologisches Neuron

Das biologische Neuron (siehe Abb. 2.2) ist das Vorbild der künstlichen Neuronen. Es besteht aus Dendriten, welche die elektrischen Signale der anderen Neuronen entgegen nehmen und diese an den Zellkörper weiterleiten. Am Axonhügel, der zwischen dem Zellkörper und Axon liegt, werden die elektrischen Signale gesammelt und summiert. Erst beim Überschreiten eines gewissen Schwellenwertes, wird das Signal weiter an das Axon geleitet. Durch den Schwellenwert wird verhindert, das sehr kleine Signale weitergeleitet werden, die keine Relevanz haben. Ohne das Filtern der Signale, wäre eine Verarbeitung der relevanten Signale nicht möglich. Das Signal welches letztendlich über das Axon an die Synapsen transportiert wird, heißt Aktionspotential. An die Synapsen schließen andere Dendriten von Neuronen an und empfangen die weitergeleiteten Signale. Ein biologisches

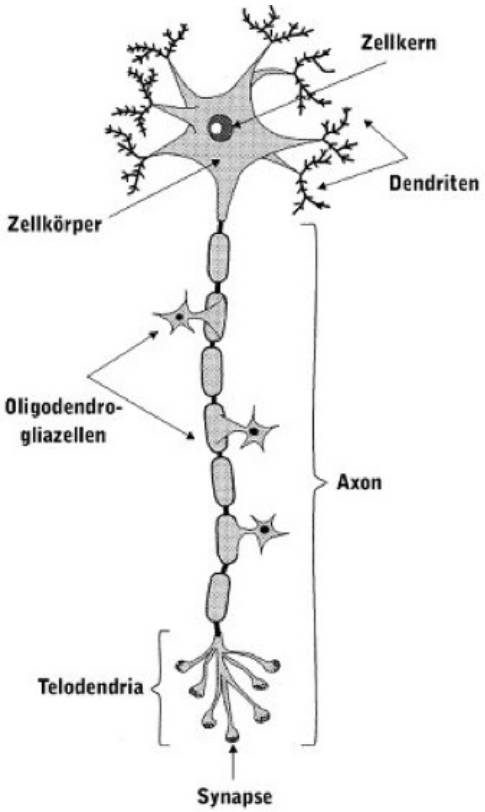


Abbildung 2.2.: Aufbau biologisches eines biologischen Neurons (Quelle: https://www.spektrum.de/lexikon/psychologie/neuron/10516)

Neuron kann dabei bis zu mehreren tausend Verbindungen zu anderen Neuronen besitzen, diese können wieder dementsprechend viele Verbindungen haben.[vgl. 8, 15]

2.2.5.2. Künstliches Neuron

Ein künstliches Neuron ist ein mathematisches Modell, das die Funktion eines biologischen Neurons nachahmt. Das künstliche Neuron in Abb. 2.3 besitzt Eingaben x_n, die mit den Dendriten des biologischen Neurons vergleichbar sind. Die Eingabeinformationen können dabei von anderen Neuronen oder aus der Umgebung stammen.
Die eingehenden Informationen werden, bevor sie in das Neuron gelangen, mit einer Gewichtung w_{nj} multipliziert. Anschließend werden die Eingaben mithilfe einer Übertragungsfunktion (Propagierungsfunktion) verknüpft, was bedeutet die Werte werden aufsummiert. Das aufsummierte Ergebnis ist die Netzeingabe net_j, diese fasst die Informationen zusammen welche in

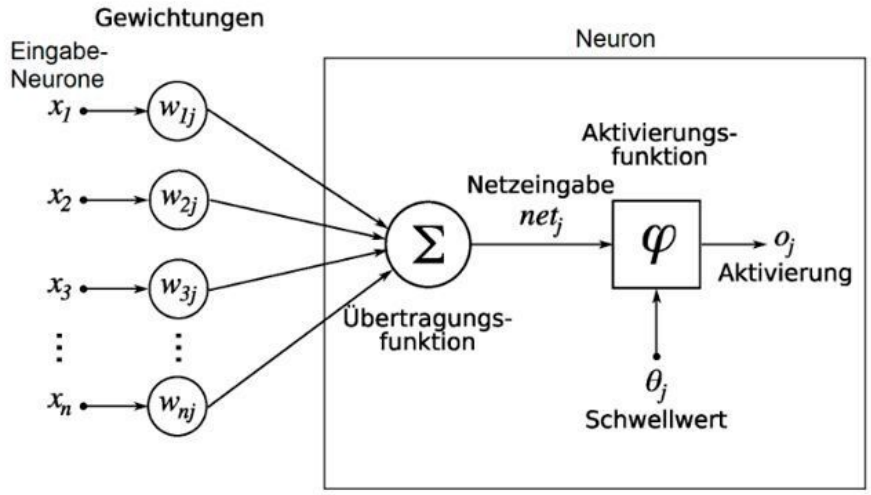


Abbildung 2.3.: Schema eines künstlichen $Neurons_j$ (Quelle: https://www.dev-insider.de/grundbegriffe-und-einfache-beispiele-a-864507/)

das Neuron gehen. Die Aktivierungsfunktion berechnet die Aktivierung o_j, welche je nach Schwellwert θ_j an alle verbundenen Neuronen weitergeleitet wird. [vgl. 15]

2.2.5.3. Das Perzeptron

Ein Perzeptron ist ein einfaches künstliches Neuronales Netz, welches von Frank Rosenblatt als Modell vorgestellt wurde. Als Beispiel wird ein Photo-Perzeptron beschrieben, dass in der Lage ist einfache visuelle Muster zu erkennen. Es besteht aus der Retina (Netzhaut), Assoziationsschicht und der Ausgabeschicht. Die Netzhaut liefert binäre Eingabewerte, welche über gewichtete Verbindungen an die Assoziationsschicht weitergegeben werden. Die Retina Neuronen sind fest mit den Neuronen der Abbildungsschicht verbunden, jedoch sind nicht alle Neuronen miteinander verknüpft. Die Verbindungen zwischen der Abbildungs- und Ausgabeschicht sind hingegen variabel. Während des Trainings, also des Lernens, werden die Gewichtungen zwischen diesen angepasst.
Beim Training werden dem Perzeptron verschiedene Beispiele gezeigt, anhand denen es lernen soll. Ein Beispiel besteht dabei immer aus einem Eingabevektor x und einem Ausgabevektor y. Die Aufgabe des Perzeptrons

ist es, durch Anpassung der internen Werte zu jedem Eingabevektor x, einen passenden Ausgabevektor y zu erzeugen. Das Ziel des Trainings ist es, auf bisher ungesehene, aber ähnliche Eingabevektoren x' ebenfalls einen geeigneten Ausgabevektor y zu assoziieren, in diesem Fall spricht man von einer Generalisierung. [vgl. 16]
Das Perzeptron ist mit seinem Aufbau allerdings beschränkt bei der Klassifikation von Daten. Es ist nur in der Lage linear separierbare Datenmengen zu trennen. Das bedeutet, die Datenmenge kann anhand einer bestimmten Gerade, in zwei separate Datenmengen getrennt werden. Die Ausgaben des Perzeptrons sind also beschränkt auf 0 oder 1, allerdings sind nicht alle Datenmengen trennbar, sondern nur solche deren Trenngerade durch den Ursprung geht.[7]

2.2.5.4. Aufbau

Der klassische Aufbau eines Neuronalen Netzes besteht aus einer Eingabeschicht (engl.: Input-Layer), einer oder mehreren versteckten Schichten (engl.: Hidden-Layer), und der Ausgabeschicht (engl.: Output-Layer). Die Eingabeschicht nimmt Werte aus der Umgebung an, und leitet sie weiter an die versteckte Schicht, von wo aus sie an weitere versteckte Schichten oder an die Ausgabeschicht geschickt werden, die das fertige Ergebnis ausgibt. Die Verbindungen der Neuronen zwischen den verschiedenen Schichten besitzen Gewichtungen, mit denen die eingehenden Werte multipliziert werden. Die Komplexität des Aufbaus hängt von der Anzahl der Neuronen und Schichten ab. [vgl. 13]

2.2.5.5. Training eines Neuronalen Netzes

Das Training eines Neuronalen Netzes ist ein wesentlicher Aspekt des Deep Learnings, bei welchem die Anpassungen der Gewichte und Schwellenwerte innerhalb des Netzwerks stattfinden. Das Training kann oft mehrere Stunden oder sogar Tage dauern, je nach Art und Menge der Daten. Ein Neuronales

Netz gilt dann als trainiert, wenn es zu einem Eingabevektor x, einen geeigneten Ausgabevekotr y zuverlässig bestimmen kann. Das Netz ist danach in der Lage für bisher ungesehene Eingabevektoren x' ebenfalls passende Ausgabevektoren y' zu zuordnen. [vgl. 16]

Ein entscheidendes Konzept im Training ist die Fehler- oder Verlustfunktion (engl.: Loss), genauer beschrieben in Abschnitt 2.2.6, welche den Unterschied der vorhergesagten Ausgaben mit den tatsächlichen Ergebnissen vergleicht und einen Fehler errechnet. Das Ziel des Trainings ist es, die Verlustfunktion auf ein Minimum zu reduzieren, indem die internen Parameter des Neuronalen Netzes angepasst werden. Die Parameter des Netzwerks werden zu Beginn des Trainings zufällig initialisiert und schrittweise im Training angepasst. [vgl. 17]
Die Rückwärtspropagierung (engl.: Backpropagation) ist ein weit verbreitetes Lernverfahren für Neuronale Netze, dass auf eben genannter Minimierung des Fehlers basiert. Für die Rückwärtspropagierung wird im Allgemeinen die Deltalernregel verwendet, die besagt, dass die Fehlerminimierung durch Änderung eines Gewichtes w_{ij} einen Gradientenabstieg bezweckt. Der Ausgangspunkt für diese Lernregel ist wie folgt definiert:

$$\Delta_p w_{ij} \equiv -\frac{dE_p}{dw_{ij}} \tag{2.1}$$

$\Delta_p w_{ij}$ entspricht dabei der Änderung eines Gewichts zwischen i und j. dE_p ist der gemessene Fehler innerhalb der Lernmenge und dw_{ij} ist die Gewichtung der Neuronen von der Schicht i zur Schicht j. [vgl. 18]

Der Backpropagation Algorithmus besteht im wesentlichen aus zwei Schritten, zuerst die Vorwärtspropagierung (engl.: Forwardpropagation) und anschließend die Rückwärtspropagierung. Der erste Schritt ist die Vorwärtspropagierung, bei dem die Eingabevektoren durch das Netzwerk verarbeitet werden. Die Eingaben werden von Schicht zu Schicht geschickt und durch die Gewichtungen und Aktivierungsfunktionen transformiert. Das Ende dieses Schrittes stellt eine Ausgabe bereit, die dann für den zweiten Schritt weiter verwendet wird.

Im nächsten und letzten Schritt des Trainings, der Rückwärtspropagierung, wird die Ausgabe mit dem tatsächlich erwarteten Wert verglichen und anschließend aus der Differenz von Soll- und Ist-Wert ein Fehler errechnet. Als Beispiels Funktion für eine Berechnung eines Fehlers, kann man den mittleren quadratischen Fehler(MSE) verwenden der in 2.2 definiert ist.

$$E = \frac{1}{2}\sum(Y_i - \hat{Y}_i)^2 \tag{2.2}$$

Nach der Fehlerberechnung, werden die Fehler für die Neuronen der einzelnen Schichten ermittelt. Dabei wird der Beitrag eines jeden Neurons zum Gesamtfehler berechnet. Anschließend wird anhand des Fehlers eines Neurons die Gewichtung von diesem angepasst. Die Gewichte werden dabei mit einem konstanten Faktor ν und entsprechend des Fehlers mit einem weiteren Term multipliziert. Dieser Prozess wird iterativ wiederholt, bis der Fehler auf ein geeignetes Minimum reduziert wurde. Ist der Fehler auf einen minimalen Wert gesunken und verändert sich nicht mehr großartig, so spricht man davon, dass das Modell konvergiert, und das Training beendet werden kann.[vgl. 16]

2.2.6. Aktivierungsfunktionen, Verlust-Funktionen und Optimierer

Für das Training und die allgemeine Funktionalität eines Neuronalen Netzes sind noch einige Komponenten notwendig.
Eine Komponente ist die Aktivierungsfunktion, welche bereits in Abschnitt 2.2.5.2 erwähnt wurde. Sie berechnet die Aktivierung o_j, welche in Abhängigkeit eines Schwellwertes θ_j an die nachgelagerten Neuronen weitergegeben wird. Sollte der Schwellwert θ_j nicht überschritten werden, wartet das Neuron solange auf weitere Eingabesignale und wiederholt die Berechnung der Aktivierung o_j, bis der Schwellwert θ_j überschritten wird und das Neuron seine Ausgabe weitergibt. Es gibt unterschiedliche Aktivierungsfunktionen für verschiedene Zwecke.

Die Identitätsfunktion ist eine einfache Aktivierungsfunktion und sorgt für eine lineare Abbildung der Eingabe auf die Ausgabe. Diese ist für einfache Aufgaben geeignet, wird die Problemstellung komplexer, so eignen sich mehr nicht-lineare Aktivierungsfunktionen wie der Tangens hyperbolicus, Rectified Linear Unit (ReLU) oder Sigmoid. Je nach Anwendung und Bedarf wird eine geeignete Funktion ausgewählt. Die Sigmoid-Funktion normalisiert die Ausgaben eines Netzes auf den Bereich zwischen 0 und 1, während der Tangens hyperbolicus einen die Werte in einen Bereich zwischen -1 und 1 bringt. Die ReLU-Funktion verhält sich etwas anders zu den vorangegangen Funktionen, da diese für alle positiven Werte die jeweiligen Ausgaben und für alle negativen Werte den Wert 0 zurückgibt. Eine weitere nennenswerte Aktivierungsfunktion ist die Softmax-Funktion, welche ebenfalls Werte in einen Bereich zwischen 0 und 1 überführt, jedoch aber eine Wahrscheinlichkeitsverteilung darstellt. Die Besonderheit der Softmax-Funktion ist die Gegebenheit, dass sich alle Ausgaben zu 1 addieren und somit die Ausgabewerte als Wahrscheinlichkeiten interpretiert werden können. [vgl. 19]

Eine weitere Komponente ist die Verlust-Funktion, welche essenziell für das Trainieren eines Neuronalen Netzes ist. Diese hat zum Ziel, den Fehler, auch Verlust (engl.: Loss) genannt, den ein Netz bei einer Vorhersage produziert, zu minimieren. Ein Fehler entsteht, wenn das Netz eine falsche oder abweichende Ausgabe als erwartet erzeugt. Um den Gesamtfehler eines Netzes zu berechnen, werden die Ausgaben des Neuronalen Netz mit den erwarteten Ergebnissen verglichen. Die Verlust-Funktion hilft dabei den Unterschied zu quantifizieren und den Gesamtfehler des Netzes zu berechnen. [19]

Wie auch bei den Aktivierungsfunktionen gibt es je nach Art der Aufgabe verschiedene Verlust-Funktionen, wie Binary CrossEntropy (BCE), Categorical Cross Entropy, oder Mean Squared Error. Binary Cross Entropy wird verwendet für binäre Klassifikationsprobleme, wie z.B. das Unterscheiden von seriösen und Spam Emails, währenddessen Categorical Cross Entropy für Probleme mit mehr als zwei Klassen verwendet wird. Die Wahl der Verlust-Funktion hängt dementsprechend auch von der Ausgabe des Netzes ab.

Die letzte wesentliche Komponente ist der Optimierer (engl.: Optimizer), welcher die Gewichte und Biases anpasst, um den Fehler zu minimieren und die Leistung des Netzes zu verbessern. Biases sind zusätzliche Konstanten, die ein Neuron zugeführt bekommt und bei der Berechnung der Ausgabe eine Rolle spielen. Der Bias erhöht die Fähigkeit des Netzes komplexe Beziehungen zwischen Eingaben und Ausgaben zu modellieren und kann so auch nicht-lineare Zusammenhänge erkennen.
Es gibt unterschiedliche Arten von Optimierungsalgorithmen, der am häufigsten verbreitete Algorithmus ist der Stochastic Gradient Descent (SGD), welcher auch Grundlage für weitere Optimierungsverfahren ist. Der SGD aktualisiert Gewichte und Biases basierend auf der Ableitung des Verlustes. Dabei wird eine zufällige Stichprobe aus dem Trainingsdatensatz verwendet, um die Berechnung zu beschleunigen. [vgl. 20]
Ein anderer Optimierungsalgorithmus ist Adam, der eine Erweiterung des SGD ist und eine adaptive Lernrate verwendet, um die internen Parameter des Netze anzupassen. Der Algorithmus erzielt durch die adaptiven Lernraten eine schnellere Konvergenz und eine höhere Genauigkeit als der SGD. [vgl. 21]

2.2.7. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) sind eine spezielle Form von Neuronalen Netzen, die in der Lage sind Bilddaten zu verarbeiten. Sie erkennen mit verschiedenen Filtern Muster, Strukturen und Geometrien in Bildern und können diese anhand dessen klassifizieren. Im nachfolgenden Abschnitt wird der Aufbau und die Funktionsweise eines CNNs erklärt, dabei wird auf die verschiedenen Schichten innerhalb des Netzes, sowie auf die mathematische Operation Faltung (engl.: Convolution) eingegangen.

Der Aufbau eines CNN besteht aus mehreren Faltungs-Schichten, Pooling-Schichten und voll vernetzten Schichten, wie in Abbildung 2.4 zu sehen.

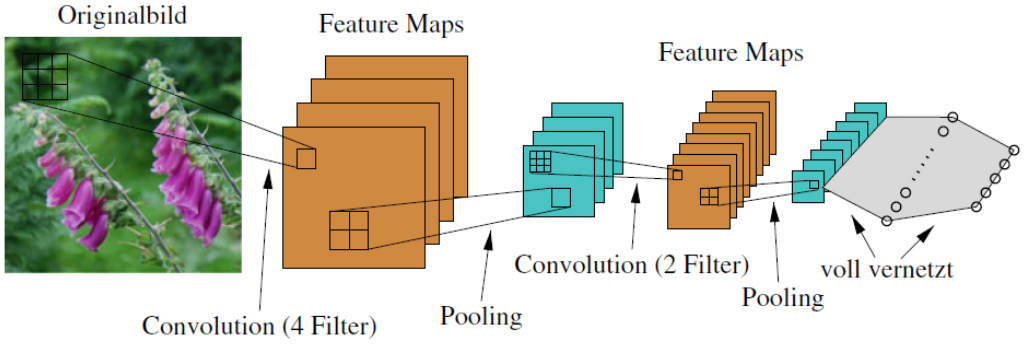


Abbildung 2.4.: Ein CNN mit zwei Faltungs-Schichten gefolgt von je einer Pooling-Schicht und am Ende zwei voll vernetzten Schichten (Quelle: [7])

Convolutional-Layer ist die Faltungs-Schicht, welche in der Lage ist Merkmale, wie Linien, Kanten und geometrische Formen in Bildern zu erkennen und diese zu extrahieren. Bei einer Faltung wandert ein Filter (engl.: Kernel) über das Eingabebild und erzeugt so ein neues Bild, oder auch sogenannte Feature-Maps. Im mathematischen Sinne gesehen, wandert der Filter über eine Funktion, anstatt über ein Bild und erzeugt somit eine neue. Bei einer Faltung eines Bildes wird meist ein Filter in Form einer 2D-Matrix verwendet, wie er in Gleichung 2.3 als Beispiel zu sehen ist. Die Werte des Filters werden zu Beginn zufällig bestimmt und im Laufe des Trainings angepasst.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{2.3}$$

Der Filter wird schrittweise über das Eingabebild geschoben, wobei jedes Element des Filters mit dem darunterliegendem Bildpunkt des Eingabebildes multipliziert und anschließend aufsummiert wird. Die Summe ist dann das Ergebnis des neuen Bildpunktes. Die Schrittweite des Filters wird als "Stride" bezeichnet, und sagt aus, wie viele Bildpunkte, bzw. Pixel, der Filter verschoben wird. Bei einer 2D-Convolution, können zwei Werte für den Stride angegeben werden, die Verschiebung in Richtung der X-Achse und Y-Achse. Die Gleichung 2.5 zeigt das Ergebnis einer Faltung mit dem Ausdruck aus 2.4.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} * \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \tag{2.4}$$

$$\begin{pmatrix} (a_{11}w_{11} + a_{12}w_{12} + a_{21}w_{21} + a_{22}w_{22}) & (a_{12}w_{11} + a_{13}w_{12} + a_{22}w_{21} + a_{23}w_{22}) \\ (a_{21}w_{11} + a_{22}w_{12} + a_{31}w_{21} + a_{32}w_{22}) & (a_{22}w_{11} + a_{23}w_{12} + a_{32}w_{21} + a_{33}w_{22}) \end{pmatrix} \tag{2.5}$$

Es gilt zu beachten, dass das Ergebnis in 2.5 kleiner ist, als das Ursprungsbild. Dies liegt daran, dass keine Fülldaten (engl.: Padding) verwendet wurde, um die Ränder des Eingabebildes aufzufüllen. Die Größe der Ergebnismatrix lässt sich durch Gleichung 2.6 berechnen, wobei W die Eingabegröße des Bildes, F die Filtergröße, S der Stride und P das Padding ist. [vgl. 22]

$$W_{out} = \frac{W + 2P - F}{S} + 1 \tag{2.6}$$

Pooling-Layer dient zur Extraktion von Merkmalen (engl.: Features) des Bildes, welche eine hohe semantische Bedeutung haben. Die Eingaben dieser Schicht, sind die Ausgaben der vorangegangenen Faltungs-Schicht. Diese Schicht verwendet entweder das Maximal- oder Mittelwert-Pooling (engl.: Max- oder Average-Pooling). Dabei werden überflüssige und redundante Informationen aus der Datenmenge entfernt. Durch das Entfernen der unwesentlicher Bestandteile reduziert sich Berechnungszeit und die Merkmale werden verdichtet.

Beim Pooling wandert ebenfalls ein Filter, meist mit einer Größe von $2x2$, und einer Schrittweite von zwei über das Bild. In Abbildung 2.5 werden die zwei Arten des Poolings durchgeführt. Das Max-Pooling extrahiert jeweils den höchsten Wert welcher innerhalb der Filter-Matrix liegt. Das

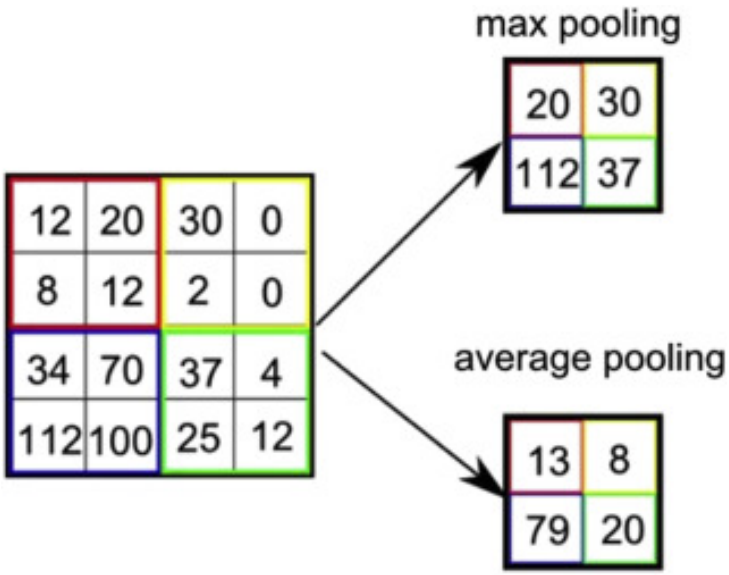


Abbildung 2.5.: Maximal und Mittelwert-Pooling, sowie das Ergebnis der beiden Methoden. (Quelle: https://www.sciencedirect.com/topics/mathematics/pooling-layer)

Average-Pooling nimmt jeweils den Mittelwert aller Elemente innerhalb der Filter-Matrix. Durch das Pooling verkleinert sich das Bild je nach Filtergröße und Schrittweite.[vgl. 22] Im Beispiel von Abbildung 2.5 verkleinert sich das Ausgangsbild um die Hälfte, die Größe nach dem Pooling berechnet sich ebenfalls mit der Formel aus 2.6.

Fully-Connected-Layer oder auch voll verknüpfte Schicht, bildet den Abschluss eines CNNs. Die Merkmale der vorangegangenen Schicht, also des Pooling-Layers, werden hier mit allen Ausgabemerkmalen verknüpft, es handelt sich dabei um ein normales Neuronales Netz.

Zunächst muss aus der Ausgabe der vorangegangenen Schicht eine 1D-Matrix erstellt werden, diesen Prozess nennt man auch “flatten”. Zuletzt wird schließlich jedes Element aus der 1D-Matrix, mit jedem Element der Ausgabe-Schicht verbunden. Die Ausgabe des Netzes zeigt dann z.B. die Klassifikation eines Bildes, in dem berechnet wird, welches Ausgabeelement die höchste Wahrscheinlichkeit besitzt. [vgl. 23]

2.2.8. Over- und Underfitting

Over- und Underfitting sind zwei Probleme, die beim Training eines Neuronalen Netzes auftreten können. Diese Probleme beeinflussen die Generalisierungsfähigkeit des Modells, also die Fähigkeit auf neuen und bisher ungesehene Daten gute Ergebnisse zu erzielen.
Overfitting tritt auf, wenn sich ein Modell zu stark auf die Trainingsdaten abgestimmt hat und somit keine generelle Strukturen mehr erkennt. Das Modell hat sich in diesem Fall die Trainingsdaten gemerkt und kann daher nur schlecht gute Ergebnisse auf neuen Daten erzielen. Ist die Kapazität des Modells zu groß, so neigt das Neuronale Netz dazu kleine Schwankungen in den Daten zu lernen und verliert somit die Generalisierung der Daten.
Beim Underfitting hingegen, ist das Modell nicht in der Lage eine zugrundeliegende Struktur innerhalb der Daten zu erkennen, was zu einer schlechten Leistung bei den Trainings- und auch bei den Testdaten führt. Im Falle des Underfittings hat das Modell meist eine zu geringe Kapazität, um sich alle relevanten Informationen zu speichern.[vgl. 20]

2.3. Magnetresonanztomographie (MRT)

Die MRT, auch Kernspintomographie genannt, ist ein bildgebendes Verfahren zur Darstellung von Struktur und Funktion der Weichteile und Organe im Körper. Mit einer MRT können Schnittbilder des Körpers oder einzelner Körperteile erstellt werden. Für die Bildgebung werden starke Magnetfelder verwendet, diese sind jedoch unbedenklich für den menschlichen Körper, da bei diesem Verfahren keine Ionisierende Strahlung verwendet wird. [vgl. 24]

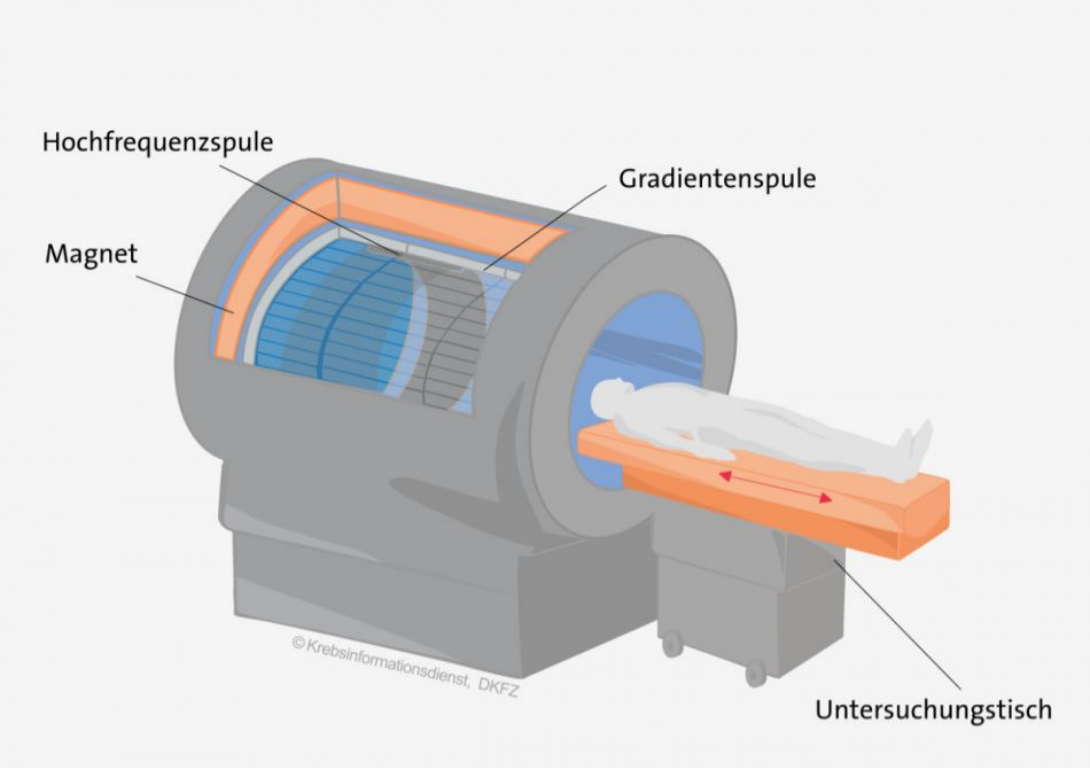


Abbildung 2.6.: Schematischer Aufbau eines Magnetresonanztomographen (Quelle: https://www.krebsinformationsdienst.de/untersuchung/bildgebung/kernspintomographie.php)

Der Magnetresonanztomograph besteht aus mehreren Komponenten:

- einem supraleitenden Hauptmagneten in einer Röhre, der ein starkes und konstanten Magnetfeld erzeugt. Ein supraleitender Magnet wird durch flüssigen Stickstoff gekühlt, sodass der Spulenwiderstand nahezu null wird. Dies ermöglicht die Erzeugung starker Magnetfelder.

- einem Gradientensystem, das zusätzliche Magnetfelder generiert, die je nach Position im Körper unterschiedlich sind und so eine räumliche Auflösung ermöglichen.
- einem Hochfrequenzsystem, welches aus einem Sende- und Empfangsspulensystem besteht. Dieses sendet Radiowellen zur Anregung der Wasserstoffatome aus, welche dadurch ihrer Ausrichtung verlieren.
- einem Computersystem, das geeignet ist für die Verarbeitung der Daten eines MRTs
- und einem Untersuchungstisch, auf welchem der Patient während der Untersuchung liegt

Das Verfahren basiert auf einem physikalischen Prinzip, dem Eigendrehimpuls auch Spin genannt. Dabei dreht sich ein Proton um seinen eignen Schwerpunkt (Kernspin) und erzeugt durch die Rotation ein Magnetfeld. Die Wasserstoffkerne in unserem Körper besitzen ebenfalls einen Spin und verhalten sich so wie kleine Magnete, die ein messbares Magnetfeld erzeugen.
Die Kernspins sind im Normalzustand ungeordnet, legt man jedoch ein starkes Magnetfeld an, so richten sich die Kernspin-Achsen entlang den magnetischen Feldlinien aus. Sie führen dabei eine Kreiselbewegung aus, bei der sie sich immer näher an die Feldlinien bewegen, aber sich niemals ganz daran ausrichten, dies wird auch als Präzessionsbewegung bezeichnet. Die Frequenz dieser Bewegung nennt sich die Larmorfrequenz.
Allein durch die Ausrichtung der Kernspins erfolgt noch keine Bildgebung, es muss zunächst ein hochfrequenter Impuls (HF-Impuls) senkrecht zum Hauptmagnetfeld gegeben werden. Der Impuls muss dabei eine bestimmte Frequenz besitzen, nämlich die der Kernspins, also die Larmorfrequenz. Durch diesen Impuls synchronisieren sich die Protonen und manche der Kernspin-Achsen kippen um 90°, wobei sie bei diesem Vorgang Energie aufnehmen.[vgl. 25]
Nach dem Impuls, kippen die Kernspins wieder zurück in ihre ursprüngliche Lage entlang des Hauptmagnetfeldes. Beim Ausrichten in die ursprüngliche

Position, wird die aufgenommene Energie in Form von wärme wieder an die Umgebung abgegeben. Dieser Prozess der Wiederausrichtung wird als longitudinale Relaxation oder "T1-Relaxation" bezeichnet. Hierbei kommt es auf die Wärmeleitfähigkeit des Gewebes an. Ein weiterer Prozess der beim Ausschalten des HF-Impulses in Gang gesetzt wird, ist der Verlust der synchronen Kreiselbewegung, wodurch die transversal Magnetisierung verloren geht. Unterschiedliche Gewebe können die transversal Magnetisierung unterschiedlich lang aufrecht erhalten und erzeugen so Kontraste auf den Bildern, dies wird auch als "T2-Relaxation" bezeichnet. Man misst im wesentlichen die Dauer der verschiedenen Prozesse und kann anhand dessen auf die Gewebearten schließen und daraus die Kontraste berechnen.[vgl. 26]

Es können zusätzlich Kontrastmittel verwendet werden, um bestimmte Teile heller erscheinen zulassen. Je nach Art des Kontrastmittels werden andere Bereicher dunkler, bzw. heller dargestellt. Die Kontrastmittel selbst sind nicht auf dem Bild zusehen, jedoch ihre Auswirkungen auf das umliegende Gewebe, so erscheint bei einer kontrastverstärkten T1-Gewichtung die Umgebung wie z.B. Blut oder Tumore heller. [vgl. 25]

Es gibt weitere Methoden für eine MRT, um bestimmte Strukturen, Gewebearten oder Flüssigkeiten heller bzw. dunkler erscheinen zu lassen. Eine davon ist die Flair-Gewichtung (fluid-attenuated inversion recovery), eine spezielle Methode zur Unterdrückung von Flüssigkeiten auf den entstehenden Bildern. Es unterdrückt z.B. das Liquor im Gehirn, wodurch bestimme Tumore besser sichtbar werden. Besitzt der Tumor jedoch einen hohen Wasseranteil, so wird dieser ebenfalls auf dem Bild unterdrückt. [vgl. 27]

3. Material und Methoden

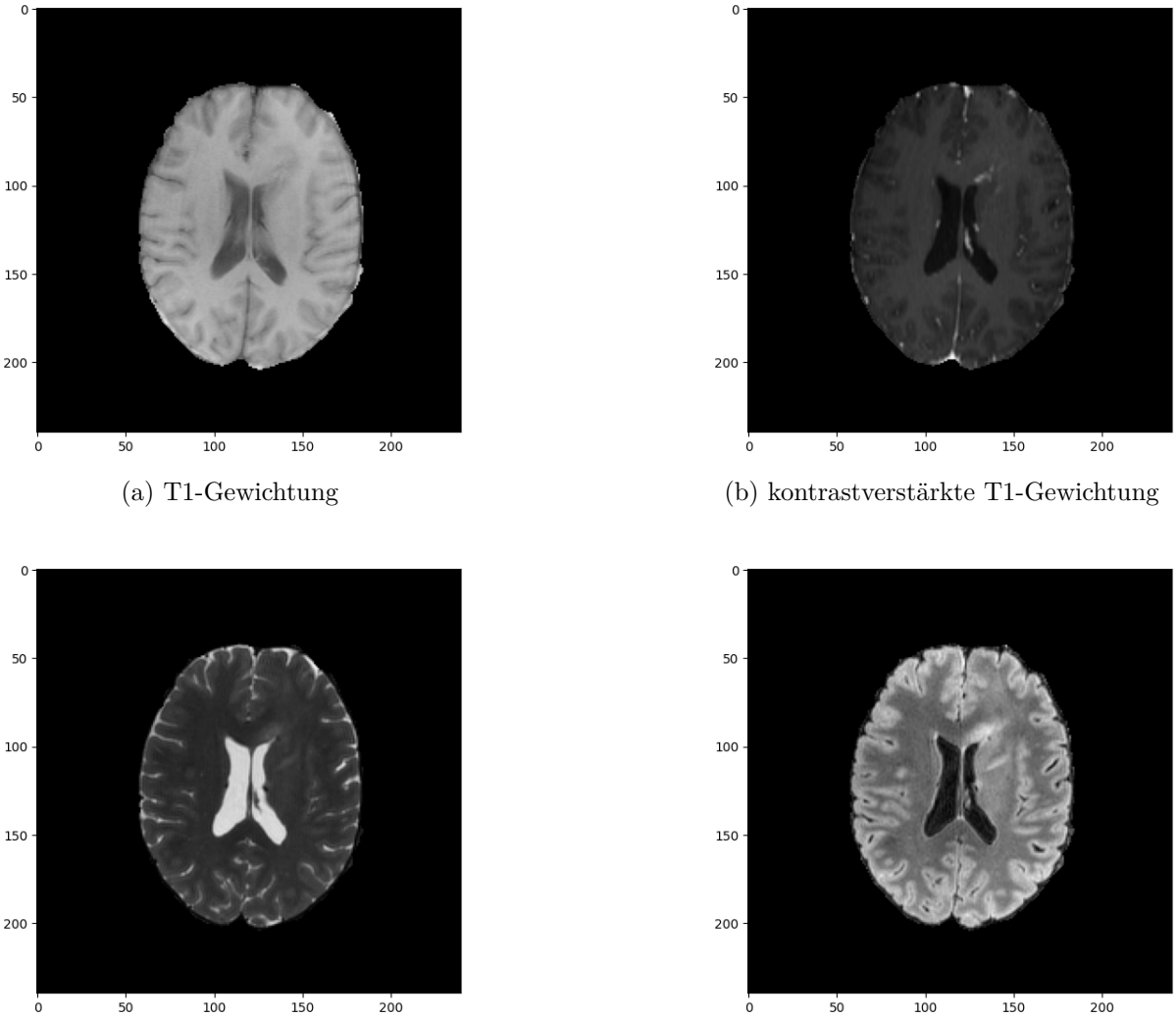
3.1. Datensatz: BraTS

Der BraTS Datensatz ist ein öffentlich verfügbarer Datensatz, der für die Entwicklung und Evaluierung von Modelle für die Segmentierung von Gehirntumoren verwendet wird. Er wurde von der Radiological Society of North America in Partnerschaft mit der American Society of Neuroradiology(ASNR) und Medical Image Computing and Computer Assisted Interventions(MICCAI) erstellt und besteht aus über tausend MRT Bildern von verschiedenen Patienten mit Hirntumoren. [vgl. 28]

3.1.1. Beschreibung

Der Datensatz enthält multi-institutionellen und multi-parametrischen MRT Scans (mpMRT), die unter klinischen Standardbedingungen mit unterschiedlichen Geräten und Protokollen aufgenommen wurden. "Multi-institutionell", bezieht sich darauf, dass die Daten für den Datensatz von mehreren medizinischen Einrichtungen bereitgestellt und zusammengeführt wurden. "Multi-parametrisch" bedeutet, dass der Datensatz mehrere Parameter bzw. Merkmale enthält, welche für die Segmentierung der Hirntumore verwendet werden können.
Für die Aufgabe der Segmentierung wurden alle Gehirntumore mit den führenden BraTS-Algorithmen segmentiert und anschließend von freiwilligen Neuroradiologie Experten mit unterschiedlicher Erfahrung nochmals verfeinert. Für die manuellen Beschriftung der Daten wurde ein strenges Protokoll eingehalten, um einheitliche Daten zu erhalten. Abschließend wurden die manuell verfeinerten Bilder von zertifizierten und erfahrenen

Neuro-Radiologen mit mehr als 15 Jahren Erfahrung in dieser Tätigkeit bestätigt. Die beschrifteten Tumor-Regionen basieren auf den VASARI-Merkmalen[1], welche für einen geschulten Radiologen sichtbar sind. Diese Merkmale umfassen den mit Gadolinium anreichernden Tumor (Kennzeichnung 4), das peritumorale ödematöse/invasive Gewebe (Kennzeichnung 2) und den nekrotischen Tumorkern (Kennzeichnung 1). [vgl. 30]



(a) T1-Gewichtung

(b) kontrastverstärkte T1-Gewichtung

(c) T2-Gewichtung

(d) Flair-Gewichtung

Abbildung 3.1.: Die vier Modalitäten des MRT Scan im Datensatz (Quelle: Eigene Darstellung)

Der Datensatz enthält 1251 Einträgen, jeder Eintrag besteht aus einer T1-, kontrastverstärkte T1-, T2- und Flair-Gewichtung (siehe Abb. 3.1) sowie

[1] VASARI-Merkmale(Visually Accesable Rembrandt Images) sind ein System für die konsistente Beschreibung von Gliomen (Tumoren) mithilfe definierter visueller Merkmale [vgl. 29]

einer fertigen Segmentierung des Hirntumors. Die Dateien selbst sind im NIfTI-Dateiformat, was für "Neuroimaging Informatics Technology Initiative" steht und ein gängiges Format für medizinische Bilddaten ist. Das Dateiformat enthält die 3D-Volumendaten, welche aus mehreren Schichten von Bildern bestehen. Jedes Voxel, ein 3D-Bildpunkt, enthält einen numerischen Wert, der das Signal bzw. den Kontrast widerspiegelt. Zusätzlich enthält das NIfTI-Dateiformat Metadaten, wie die räumliche Orientierung, Bildgröße,Raumkoordinaten und weitere Metadaten. [31] Die Bilder liegen in einer Größe von 150x240x240 Pixeln vor, dies entspricht wie bereits erwähnt den 3D-Volumendaten, welche in 150 Schichten aufgeteilt wurden, wobei jede Schicht ein Bild in der Größe von 240x240 Pixeln entspricht.

3.1.2. Vorverarbeitung

Die Vorverarbeitung der Daten ist ein wichtiger Schritt, damit sich die Qualität und Effektivität des Trainings von einem Neuronalen Netz verbessern. Der bereitgestellte Datensatz wurde bereits einer Standard Vorverarbeitung (engl.: Pre-Processing) unterzogen. Zunächst wurden die Daten vom DICOM-Dateiformat in das NIfTI-Dateiformat konvertiert, was zur einfacheren Handhabung der Daten dient. Ebenfalls wurde eine Anpassung auf gleiche anatomische Vorlage und isotrope Auflösung getätigt. Abschließend wurde das Skull-Stripping[1] durchgeführt um irrelevante Strukturen zu entfernen. [vgl. 30]

Normalisierung Nach diesen Schritten sind die Daten noch nicht für das Training eines Neuronalen Netzes geeignet. Die Daten müssen zunächst normalisiert werden, dies kann auf zwei Wegen geschehen. Entweder die Werte werden auf einen Bereich von 0 bis 1 skaliert oder man verwendet die

[1] Skull-Stripping ist der Prozess des Entfernens von Schädel und anderen nicht auf das Hirn bezogenen Strukturen vom Bild. [vgl. 32]

Z-Standardisierung. Bei einer einfachen Skalierung der Werte wird folgende Gleichung verwendet

$$f(x) = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

Dies führt zwar zu einer Vereinheitlichung der Werte, jedoch ist die Verteilung der Merkmale im Bild so nicht normiert. Für dieses Vorhaben verwendet man die Z-Standardisierung, welche den Mittelwert der Daten auf 0 und die Standardabweichung auf 1 bringt. Es wird verhindert, dass bestimmte Merkmale dominant werden, im Vergleich zu anderen. Zudem wird die Genauigkeit des Modelle erhöht.[vgl. 20] Die gleich für die Z-Standardisierung lautet

$$f(x) = \frac{x - \mu_x}{\sigma_x} \tag{3.2}$$

wobei μ_x der Mittelwert und σ_x die Standardabweichung des Datensatzes ist. Diese Werte müssen vorab berechnet werden, in dem der Mittelwert und die Standardabweichung von jedem Bild berechnet wird, anschließend aufsummiert und durch die Anzahl der Bilder geteilt wird

$$\begin{aligned} \mu &= \frac{1}{n} \sum_{i=1}^{n} x_i \\ \sigma &= \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2} \end{aligned} \tag{3.3}$$

wobei n die Anzahl der Bilder und x_i ein Bild des Datensatzes ist.

Data Augmentation Im medizinischen Bereich sind die Bilddaten oft begrenzt, weshalb sich das Training eines Neuronalen Netzes dadurch erschwert. Eine Methode dem entgegen zu wirken ist die Data Augmentation, bei welcher die Diversität des Trainingsdatensatzes erhöht wird. Verschiedene Augmentierungen sind Bildbearbeitungsmethoden wie das Rotieren, Skalieren, Ausschneiden oder auch Spiegeln eines Bildes. Es ist wichtig darauf zu achten, dass alle Augmentierungen sowohl auf die Eingangsbilder,

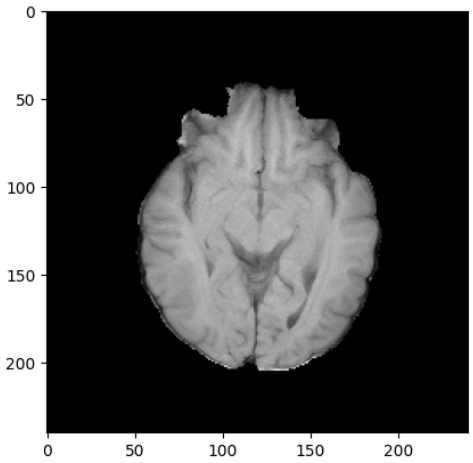
als auch auf die dazugehörigen Segmentierungen angewendet werden, da das Neuronale Netz sonst falsche Masken für die Eingaben erlernt. [vgl. 33]

3D-Bild zu 2D-Bilder Die Daten liegen als 3D-Bilddaten vor, jedoch ist die Anzahl der Daten mit 1251 Bildern eher gering. Um mehr Bilder aus den vorhandenen Daten zu bekommen, werden die 3D-Bilder in einzelne 2D-Bilder geschnitten. Wie bereits in Abschnitt 3.1.1 erwähnt, besteht ein 3D-Bild aus 150 geschichteten Bildern mit der Größe von 240x240 Pixeln. Jede einzelne Schicht wird als Separates Bild abgespeichert, um so die Anzahl der Daten zu erhöhen, so entstehen aus einem 3D-Bild 150 einzelne 2D-Bilder. Es wird jedoch nicht nur die Anzahl der Daten erhöht, sondern auch die Geschwindigkeit der Berechnungen, da dies im zweidimensionale weniger Rechenleistung und Speicher benötigt. Die Konvertierung von 3D- in 2D-Bilder bringt aber auch Nachteile mit sich, wie der Verlust des räumlichen Zusammenhangs zwischen den einzelnen Schichten. [vgl. 34]

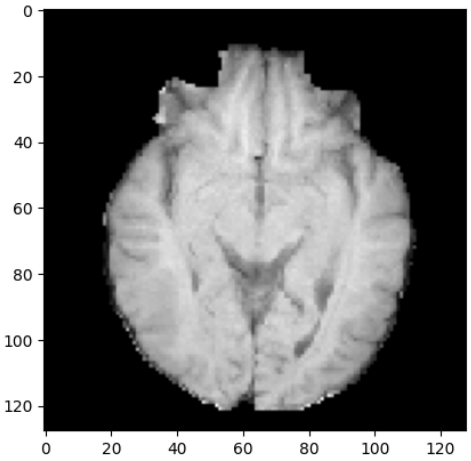
Größen Skalierung Bei manchen Neuronalen Netzen, kann die Größe des Bildes eine Rolle spielen. Es gibt Architekturen, bei denen die Bildgröße beliebig sein kann, oder das Eingabebild eine bestimmte Größe vorweisen muss, damit es verarbeitet werden kann.
Die Architektur welche für die Segmentierung der Gehirntumore verwendet wird, ist ein U-Net und wird in Kapitel 3.2.1 genauer beschrieben. Aufgrund der besonderen Architektur des UNets, müssen die Bilder eine bestimmte Größe haben. Der Grund für die spezifische Eingabegröße liegt in der Architektur des UNets, genauer gesagt durch die Down- und Upsampling Pfade. Die Größe des Bildes, wird schrittweise durch Konvolution und Max-Pooling Schichten auf der einen Seite reduziert und auf der anderen Seite wieder vergrößert.
Aufgrund des Designs des U-Net muss die Größe der Bilder ein Vielfaches von der Anzahl der Downsampling-Schichten sein. Die Größe der Bilder wird durch die Gegebenheiten definiert durch 2^N, wobei N die Anzahl der

Downsampling-Schichten ist. Sei $N = 4$, so muss die Eingabegröße der Bilder ein vielfaches von $2^4 = 16$ sein. Ist nicht die richtige Größe gegeben, so kann es zur Inkonsistenz der Größen kommen, was wiederum die Leistung des Modells beeinflusst. [vgl. 35]

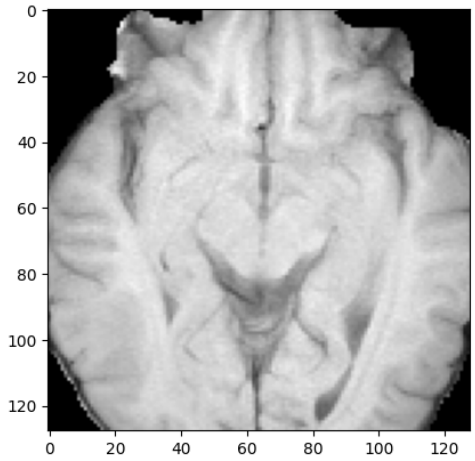
Um die Bilder in eine geeignete Größe zu bringen werden sie von der Mitte ausgehend nach außen hin ausgeschnitten. Die Größe des Bildausschnittes beträgt im Grunde 128x128 Pixel. Es kommt jedoch vor, dass Teile des Gehirns abgeschnitten werden (siehe Abb. 3.2c), weshalb noch ein Puffer von 60 Pixeln je Rand hinzugefügt wurde. Nach Hinzufügen des Puffers ist das Bild nicht mehr in einer geeigneten Größe und wird abschließend auf eine Größe von 128x128 Pixeln skaliert (siehe Abb. 3.2b).



(a) Original Bild



(b) Ausschnitt mit Puffer



(c) Ausschnitt ohne Puffer

Abbildung 3.2.: Orginal Bild (T1-Gewichtung) im Vergleich zum Ausschnitt mit und ohne Puffer. (Quelle: Eigene Darstellung)

3.1.3. Aufteilung

Es ist wichtig den Datensatz in verschiedene Teile aufzuteilen, damit die Entwicklung eines Modells gelingt. Der Datensatz wird dafür in einen Trainings-, Validerungs- und Testdatensatz aufgeteilt. Diese verschiedenen Datensätze ermöglichen es die Leistung und die Generalisierungsfähigkeit des Modells zu beurteilen.
Der Trainingsdatensatz wird verwendet, um das Modell zu trainieren, indem es die internen Modellparameter anhand der Eingabedaten und den dazugehörigen Beschriftungen anpasst. Je größer der Trainingsdatensatz ist, desto besser kann das Modell generalisieren, da es viele Daten mit unterschiedlichen Merkmalen lernt.
Um die Leistung des Modells während des Trainings zu beurteilen wird der Validierungsdatensatz verwendet. Anhand der Ergebnisse auf dem Validierungsdatensatz, werden die Hyperparameter des Modells angepasst. Es ist wichtig darauf zu achten, dass dieser Datensatz keine Überschneidungen mit dem Trainingsdatensatz hat, da es sonst zu falschen Ergebnissen kommt. [vgl. 23] Möchte man das Modell abschließend nochmals evaluieren, so kommt oft ein Testdatensatz zum Einsatz, den das Modell noch nie vorher gesehen hat. Nachdem das Training vollständig abgeschlossen wurde, kann der Testdatensatz herangezogen werden, um die finale Leistung des Modelle zu beurteilen.

3.2. Modellentwicklung

Die Entwicklung des Modells ist ein anspruchsvolles Unterfangen, welches im nachfolgenden Kapitel genauer erläutert wird. Zunächst wird die Architektur des Modells vorgestellt und wie dieses aufgebaut ist. Der nächste Punkt erklärt was Hyperparameter sind und was diese bewirken, gefolgt von den Problemen bei der Entwicklung, sowie die Implementierung mit PyTorch. Zum Schluss wird noch der Prozess des Trainings beschrieben.

3.2.1. Architektur des Neuronalen Netzes

Die gewählte Architektur ist ein U-Net, das seinen Namen aufgrund seines U-förmigen Aufbaus (siehe Abb. 3.3) bekommen hat. Das U-Net wurde speziell für die Segmentierung biomedizinischer Bilder entwickelt und ist eine spezielle Form von CNNs. Das U-Net ist speziell darauf ausgelegt auch mit kleinen Datensätzen gute Ergebnisse zu erzielen. In der Medizin stehen oft nur begrenzt beschriftete Daten zur Verfügung, da der Aufwand für die Beschriftung der Daten hoch ist. Aufgrund dieser Tatsache ist das U-Net für diese Zwecke besonders geeignet. Die folgende Beschreibung der Architektur basiert auf dem originalen U-Net und wird in dieser Arbeit als Basis für das Modell verwendet, welches genauer in Abschnitt 3.2.3 beschrieben wird.

Die Architektur des Netzes besteht aus zwei Teilen, dem Downsampling Pfad, auch Encoder genannt, und dem Upsampling Pfad, der Decoder. Der Encoder ähnelt einem klassischen CNN und besteht aus einer reihe von Convolutional und Max-Pooling Operationen, wobei es jedoch keine vollständig vernetzen Schichten gibt.

Der Encoder wird verwenden, um die Feature-Maps aus dem Bild zu extrahieren. Bei jedem Schritt, den die Eingabe durch den Encoder macht, werden mehr Feature-Maps extrahiert und die räumliche Dimension des Bildes verringert. Die Ausgabe wird an die nächsthöhere Schicht weitergegeben, dies dient dazu möglichst viele Details auf verschiedenen Ebenen zu

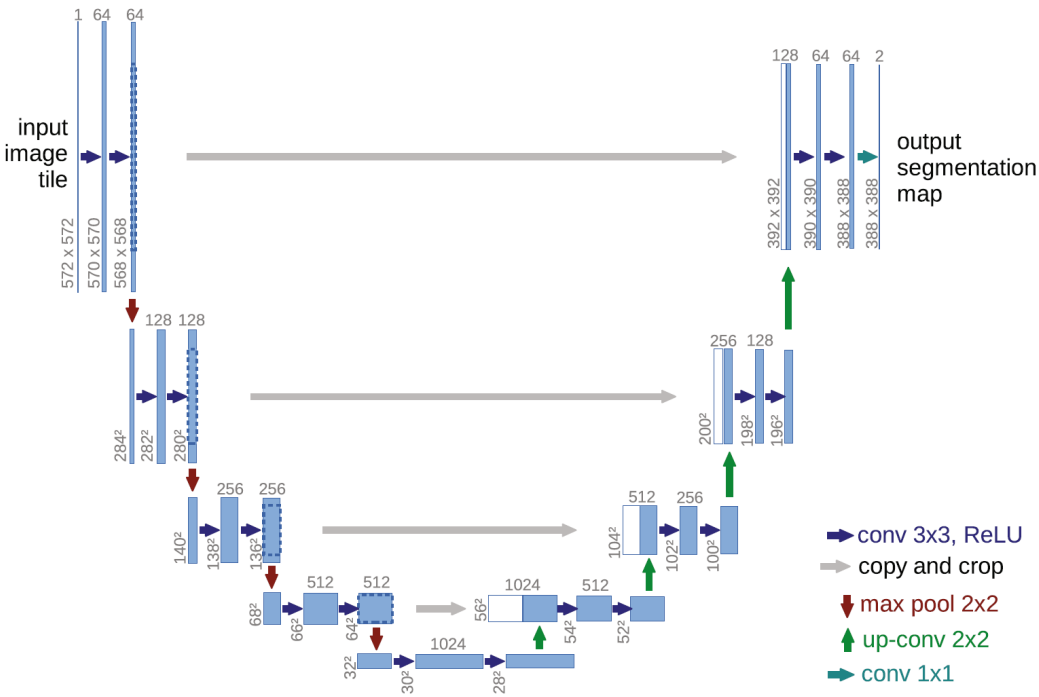


Abbildung 3.3.: Aufbau eines U-Nets mit einem Encoder und Decoder Abschnitt für die Extrahierung von Merkmalen und die Wiederherstellung des Original Bildes mit der fertigen Segmentierung (Quelle: [35])

erhalten. Der Aufbau besteht dabei aus einer Reihe von zweifach Faltungen mit einem 3x3 Filter, auf die jeweils immer eine Max-Pooling Schicht mit einer Filtergröße von 2x2 und einer Schrittweite von 2 folgt. Bei jedem Encoder Schritt wird die Anzahl der Filter, bzw. Feature-Maps in den Faltungsschichten verdoppelt.

Im Decoder wird die komprimierte Eingabe wieder auf die ursprüngliche Bildgröße skaliert, auch ”Upsampling“ genannt. Dies geschieht durch eine Aneinanderreihung von Upsampling Schichten jeweils gefolgt von zweifach Faltungen und einer Aktivierungsfunktion. Nach jedem Upsampling der Feature-Maps folgt eine Faltung mit einem 2x2 Filter, welche die Anzahl der Feature-Maps jeweils halbiert. Hierbei findet zusätzlich eine Verkettung der Feature-Maps aus dem entsprechenden Encoder Abschnitt statt. Nach der 2x2-Faltung folgen jeweils immer zwei 3x3-Faltungen mit einer abschließenden ReLU Aktivierungsfunktion. Abschließend an den Decoder Abschnitt folgt noch eine Faltung mit einem 1x1 Filter, der die Features auf

die gewünschte Anzahl von Klassen der Segmentierungsaufgabe abbildet. [vgl. 35]

3.2.2. Hyperparameter

Hyperparameter sind Parameter, deren Werte vor dem Training eines Modells festgelegt und während des Trainings, bis auf gewisse Ausnahmen nicht angepasst werden. Sie sind entscheidend für die Leistung des Modells und müssen sorgfältig ausgewählt werden. Die Auswahl der richtigen Hyperparameter ist oft eine Kunst und erfordert sowohl Erfahrung als auch experimentelles Ausprobieren.

In der Praxis wird häufig eine Technik namens Hyperparametersuche oder -optimierung verwendet, bei der verschiedene Kombinationen von Hyperparametern systematisch ausprobiert werden, um diejenige zu finden, die die beste Leistung liefert. In diesem Abschnitt werden die wichtigsten Hyperparameter, die bei der Entwicklung des eines Deep Learning Modelle relevant sind genauer beschrieben.

Lernrate Die Lernrate ist ein entscheidender Hyperparameter in fast allen Optimierungsalgorithmen für Neuronale Netze. Sie bestimmt, wie stark die Gewichte im Netzwerk in jedem Schritt des Trainings angepasst werden. Eine zu hohe Lernrate kann dazu führen, dass das globale Minimum der Verlustfunktion übersprungen wird und das Modell möglicherweise nicht konvergiert. Bei einer zu niedrigen Lernrate, wird der Fehler immer nur um sehr kleine Werte verringert, was zu einer langsamen Konvergenz führt. Es kann auch passieren, das der Fehler um einen Wert oszilliert und damit bei einer suboptimalen Lösung steckenbleibt. [vgl. 36]

Batch Größe Die Batch Größe bezieht sich auf die Anzahl der Trainingsbeispiele, die das Netzwerk gleichzeitig sieht, bevor es seine Gewichte aktualisiert. Eine größere Batch Größe kann zu stabileren Aktualisierungen

der Gewichte führen, aber benötigt mehr Speicherplatz und verlangsamt das Training. Eine kleinere Batch Größe hingegen kann zu einem schnelleren, aber weniger stabilen Training führen. Übliche Größen für eine Batch reichen von 2, 8, 16 bis hin zu hunderten von Bildern in einer Batch. [vgl. 37]

Anzahl der Epochen Eine Epoche bezeichnet einen Durchlauf des gesamten Datensatzes während des Trainings. Die Anzahl der Epochen, für die das Modell trainiert wird, beeinflusst, wie gut es Muster aus den Daten erlernen kann. Es ist wichtig eine geeignete Anzahl von Epochen auszuwählen, um einen optimalen Trainingsprozess zu erhalten. Bei einer zu niedrigen Anzahl an Epochen, hat das Netz nicht genügend Zeit, wichtige Muster zu lernen, wobei es zum Underfitting kommen kann. Ist die Anzahl der Epochen hingegen zu hoch, besteht die Gefahr, dass das Netz sich zu sehr an die Trainingsdaten anpasst und es zum Overfitting kommt. [vgl. 20]

Architektur spezifische Parameter Im Kontext der U-Net Architektur gibt es auch spezifische Hyperparameter, die bei der Entwicklung des Modelle berücksichtigt werden müssen, wie die Anzahl und Größe der Filter in den Faltungsschichten, sowie die Tiefe des Netzes und die Art der Aktivierungsfunktionen.

3.2.3. Implementierung

Die Implementierung des Modells für die Segmentierung der Gehirntumore erfolgt in Python mit PyTorch und basiert auf der U-Net Architektur, welche in 3.2.1 erläutert wurde. Die Implementierung weicht in geringen Teilen von der originalen Architektur des U-Net ab. Die grobe Struktur des U-Net wurde weitestgehend übernommen, mit jeweils vier Down- und Upsampling Schichten.

3.2.3.1. Padding

Aufgrund der Faltungen innerhalb des Netzes reduziert sich die Bildgröße, da keine Fülldaten verwendet werden. Um dem entgegenzuwirken, aus Gründen der besseren Implementierung, wird an jeden Rand des Bildes eine Bestimmte Anzahl von Pixeln hinzugefügt. Da die Bildgröße bekannt ist aufgrund der gegebenen Architektur (siehe Abschnitt 3.1.2), kann für die Berechnung der Füllmenge P die Formel 2.6 verwendet werden. Die Bildgröße W ist definiert durch ein vielfaches von 2^N, wobei N die Anzahl der Downsampling Schichten beschreibt. Im konkreten Fall ist $N = 4$ und die Eingabegröße der Bilder ist 128x128 Pixel, was ein vielfaches von $2^N = 2^4 = 16$ ist. Die Schrittweite S und Filtergröße F sind bereits durch den Aufbau des U-Net mit $S = 1$ und $F = 3$ gegeben. Daraus ergibt sich die Füllmenge mit Formel 2.6:

$$P = \frac{W \cdot S - 1 - W + F}{2} \tag{3.4}$$

$$P = \frac{2^N \cdot 1 - 1 - 2^N + 3}{2} = 1 \tag{3.5}$$

Durch das Beibehalten der Eingabegröße, ist die Implementierung aufgrund der einfachen Berechnung der Bildgröße für jede Schicht sehr von Vorteil.

3.2.3.2. Batch Normalisierung

Die Batch Normalisierung ist eine Technik, die häufig beim Deep Learning verwendet wird, um das Training zu beschleunigen und die Stabilität des Netzwerks zu erhöhen. Das Vorgehen bei der Batch Normalisierung ist es, die Eingaben zu jeder Schicht zu normalisieren, indem die mittlere Aktivierung auf 0 und die Standardabweichung der Aktivierung auf 1 gesetzt wird.
Hierfür wird in der Regel der Mittelwert und die Standardabweichung für jede Batch berechnet, daher auch der Name Batch Normalisierung. Es zeigte sich, dass durch Batch Normalisierung höhere Lernraten verwendet werden

können und das Training im Allgemeinen schneller und effektiver verläuft. Zusätzlich wird die Stabilität des Netzwerks erhöht und die Notwendigkeit für sorgfältige Initialisierung der Modellparameter entfällt. Die Schicht wirkt außerdem teilweise als eine Art Dropout, bei welchem das Modell bestimmte Neuronen "vergisst" und neu initialisiert, um so ein Overfitting zu vermeiden.[vgl. 38] PyTorch stellt bereits eine Batch Normalisierungs Schicht zur Verfügung, die für die Normalisierung der Batches verwendet werden kann. Diese Schicht wird nach jeder Faltung und noch vor der Aktivierungsfunktion angewendet.

3.2.4. Verlustfunktion

Die Auswahl der richtigen Verlustfunktion ist entscheidend für den Erfolg eines Neuronalen Netz. Bei der Klassifizierung von medizinischen Bilddaten, gibt es einige Herausforderungen, die eine spezielle Auswahl der Verlustfunktion erfordern.
Die Kreuzentropie (engl.: Cross Entropy) ist eine gängige Wahl für die Verlustfunktion bei Klassifizierungsproblemen. Sie misst, wie gut die geschätzte Wahrscheinlichkeitsverteilung des Modells mit der tatsächlichen Verteilung der Daten übereinstimmt. Wenn die Vorhersage des Modells genau mit den tatsächlichen Klassen übereinstimmt, ist die Kreuzentropie Null. Wenn die Vorhersage jedoch weit von der tatsächlichen Klasse entfernt ist, steigt der Verlust exponentiell an. [vgl. 39]
Obwohl die Kreuzentropie in den meisten Fällen recht effektiv ist, kommt diese insbesondere bei der Segmentierung von medizinischen Bildern an ihre Grenzen. Eines der Hauptprobleme ist, dass die Kreuzentropie Pixelweise berechnet wird und somit kein globales Verständnis für die räumliche Struktur des Bildes hat. Bei der Segmentierung von Gehirntumoren ist es jedoch wichtig auch benachbarte Pixel zu betrachten, um so die räumliche Struktur im Blick zu behalten.
Ein anderes Problem ist das Ungleichgewicht der verschiedenen Klassen. Die zu segmentierenden Tumore sind meist wesentlich kleiner als das gesamte

Bild, was zur ungleichen Verteilung von Tumor- und Hintergrundklassen führt. Aufgrund dessen ist die Kreuzentropie stark beeinflusst durch die Hintergrundelement und fällt damit schnell gegen Null. [40]
Aus diesen Gründen wird häufig eine weitere Verlustfunktion hinzugezogen. Eine der verbreitetsten Verlustfunktionen für die Segmentierung ist der Dice Loss. Dieser ist eine regionsabhängige Verlustfunktion, welcher auch die räumliche Struktur des Bildes für die Minimierung des Fehlers berücksichtigt. Der Dice Loss basiert auf dem Dice-Koeffizienten oder auch Sørensen-Dice-Koeffizient, einer Metrik zur quantitativen Bewertung der Ähnlichkeit zwischen zwei Mengen, welcher in Abschnitt 4.1.2 genauer beschrieben wird. Im Kontext der Bildsegmentierung wird der Dice-Koeffizient verwendet, um die Ähnlichkeit zwischen der vorhergesagten Segmentierung und der tatsächlichen Ground-Truth-Segmentierung zu messen. Der Dice-Koeffizient kann als Verlustfunktion dargestellt werden durch

$$1 - DSC(A, B) \tag{3.6}$$

oder alternativ auch durch den negative Wert des Dice-Koeffizient. [vgl. 41]

Weiter besteht die Möglichkeit eine Kombination der beiden Verlustfunktionen zu verwenden. Bei dieser Methode bekommt jede der Funktionen eine Gewichtung zugeordnet, wie stark sie Einfluss auf den Verlust hat. Da Kreuzentropie allein nicht geeignet ist für die Segmentierung medizinischer Bilder, jedoch aber für die allgemeine Segmentierung, wird diese oft in Kombination mit dem Dice Loss verwendet. Durch die Verwendung des Dice Loss als zweite Verlustfunktion wird ebenfalls Wert auf die unausgeglichen Klassen gelegt und führt so zu besseren Ergebnissen. [vgl. 42]

3.2.5. Probleme bei der Entwicklung

Bei der Entwicklung von Deep Learning Modellen für medizinische Bildsegmentierung können unterschiedliche Probleme auftreten, die durch ver-

schiedene Faktoren verursacht werden. Eines der häufigsten Probleme bei der Entwicklung ist der Mangel an Grafikkartenspeicher. Aufgrund der Bildgröße der MRT Scans passen relativ wenige Bilder in den Speicher der Grafikkarte. Behält man die Dimension der ursprünglichen Bilder bei, so würde nur eine sehr geringe Größe für eine Batch funktionieren, ohne den Speicher zu überfüllen.
Die Bilder wurden daher wie in Abschnitt 3.1.2 beschrieben, in 2D Bilder konvertiert und auf eine kleinere Bildgröße skaliert. Aufgrund der Konvertierung von 3D zu 2D kann nun eine deutlich Größe Anzahl an Bildern pro Batch verwendet werden, um das Modell zu trainieren.
Ein weiteres Problem, ist das Fehlen von leistungsstarker Hardware für schnellere Berechnungen. Die Hardware welche für das Training von Deep Learning Modellen verwendet wird ist häufig sehr kostenintensiv und benötigt eine Menge Strom. Die verwendete Grafikkarte bei der Entwicklung ist daher lediglich eine Consumer Grafikkarte, welche für das Training von vergleichsweise kleinen Modellen geeignet ist.

3.2.6. Training

Bevor ein Neuronales Netz trainiert werden kann, muss gewährleistet sein, dass die Daten ausreichend vorverarbeitet wurden und somit geeignet sind für das Training. Sind die Daten in einer geeigneten Form, kann das Training beginnen. Das Training eines Netzes verläuft über mehrere Epochen und kann eine Menge Zeit in Anspruch nehmen. Bei jeder Epoche werden alle Bilder des Trainingsdatensatz einmal durch das Netzwerk gegeben und anschließend die Leistung mittels eines Validierungsdatensatzes gemessen. Dieser Prozess wird für eine vorher ausgewählte Anzahl an Epochen durchgeführt.

Der Prozess beginnt mit der Initialisierung des Modells, bei welchem alle internen Parameter auf einen zufälligen Wert gesetzt und im Verlauf des Trainings angepasst werden. Anschließend werden die Trainingsdaten in Form von Batches durch das Netzwerk gereicht, auch Feedforward genannt.

Bei diesem Teil, werden verschiedenen Berechnungen innerhalb des Netzes ausgeführt, um eine Vorhersage bzw. Segmentierung für die Daten zu erstellen.
Nachdem eine Batch durch das Netzwerk gereicht wurde und die Segmentierungen erstellt wurden, wird der Fehler berechnet. Dieser sagt aus, wie weit die Ausgaben des Netzes mit dem tatsächlichen Ergebnissen auseinander liegen. Anhand dieses Fehlers werden dann im nächsten Schritt die Parameter innerhalb des Modells angepasst. Mithilfe der Rückwärtspropagierung werden die Fehler eines jeden Neurons zum Gesamtfehler berechnet. Anhand dessen können die Gewichte und Bias eines jeden Neuron optimiert werden. [vgl. 20]

4. Experimente

In diesem Kapitel werden verschiedene Experimente durchgeführt, um die Auswirkungen von verschiedenen Parametern und Verarbeitungsschritten zu untersuchen. Insbesondere werden die Lernrate, Batch Größe und die Vorverarbeitung genauer untersucht. Jeder dieser Komponenten spielt eine wesentliche Rolle bei dem Training des Modells. Bei der Untersuchung von Lernrate und Batch Größe werden verschiedene Werte getestet, um zu sehen wie der Trainingsprozess verläuft. Bei der Komponente Vorverarbeitung wird das Thema der Größenskalierung bzw. des Bildausschnitts eine Rolle spielen.
Zunächst werden verschiedene Metriken vorgestellt, anhand derer man die Auswirkungen der verschiedenen Parameter beurteilen kann. Anschließend werden die einzelnen Experimente genauer erläutert und deren Ergebnisse präsentiert.

4.1. Metriken

Die Bewertung eines Modells ist ein wichtiger Schritt in der Entwicklung. Mit Metriken wird ein Modell beurteilt und es wird die Leistung evaluiert. Es gibt verschieden Methoden bzw. Berechnungen, um die Leistungsfähigkeit zu untersuchen. Es werden nachfolgend verschiedene Metriken aufgezeigt, sowie deren Ergebnisse auf die jeweiligen Modelle vorgestellt. Metriken sind bestimmte Funktionen, anhand derer die Leistung eines Modelle bewertet werden kann. Sie berechnen die Übereinstimmung zwischen der Vorhersage und dem tatsächlichen Ergebnis. Abhängig von der Problemstellung sind eine oder mehrere geeignete Metriken auszuwählen. Eine einzelne Metrik reicht oft nicht aus für eine allumfassende Bewertung, deshalb werden meist mehrere Metriken betrachtet.

4.1.1. Jaccard-Index

Der Jaccard-Index, auch Intersection over Union genannt, ist eine Metrik für die Bewertung der Ähnlichkeit oder Überlappung zweier Mengen A und B. Der Jaccard-Index misst das Verhältnis der Schnittmenge beider Mengen zur Vereinigung dieser. In Bezug auf Bildsegmentierung bedeutet dies, dass der Jaccard-Index das Verhältnis des Bereichs des gemeinsamen Segments zum Bereich der vereinigten Segmente misst. Der Jaccard-Index kann einen Wert zwischen 0 und 1 haben, wobei 0 für keine Überlappung und 1 für perfekte Übereinstimmung steht. Mathematisch wird der Jaccard-Index wie folgt berechnet: [vgl. 43]

$$J(A, B) = \frac{|A \cup B|}{|A \cap B|} \tag{4.1}$$

4.1.2. Dice-Koeffizient

Der Dice-Koeffizient, auch als Sørensen-Dice-Koeffizient oder F1-Score bekannt, misst ebenfalls die Ähnlichkeit zwischen zwei Mengen oder Segmentierungen A und B. Der Dice-Koeffizient berechnet das Verhältnis des doppelten der Schnittmenge zur Summe der Größe beider Mengen. In Bezug auf Bildsegmentierung misst der Dice-Koeffizient das Verhältnis der doppelten Fläche des gemeinsamen Segments zur Summe der Flächen beider Segmente. Wie der Jaccard-Index kann auch der Dice-Koeffizient Werte zwischen 0 und 1 annehmen, wobei 0 für keine Überlappung und 1 für perfekte Übereinstimmung steht. Mathematisch wird der Dice-Koeffizient wie folgt berechnet:

$$DSC(A, B) = \frac{2|A \cup B|}{|A| + |B|} \tag{4.2}$$

Alternativ kann der Dice-Koeffizient auch mittels boolschen Daten dargestellt werden:

$$DSC = \frac{2TP}{2TP + FP + FN} \tag{4.3}$$

”True Positive“ (TP) enthält die Anzahl der korrekt als positiv segmentierten Pixel, die tatsächlich zur gewünschten Klasse gehören. ”False Positive“(FP) gibt die Anzahl der fälschlicherweise als positiv segmentierten Pixel, die tatsächlich nicht zur gewünschten Klasse gehören. “False Negative” (FP) hingegen beinhaltet die Anzahl der Pixel die tatsächlich zur gewünschten Klasse gehören, aber fälschlicherweise als negativ segmentiert wurden. [vgl. 44]

4.1.3. Pixel Accuracy

Pixelgenauigkeit, auch als Pixel Accuracy bezeichnet, ist eine Bewertungsmetrik, die häufig im Bereich der Computer Vision und der Bildsegmentierung verwendet wird. Sie misst den Prozentsatz der korrekt klassifizierten Pixel in einem Bild oder einer Gruppe von Bildern. Die Pixelgenauigkeit wird berechnet, indem die vorhergesagten Labels jedes Pixels im Bild mit den Ground-Truth-Labels verglichen werden. Wenn das vorhergesagte Label mit dem Ground-Truth-Label für einen Pixel übereinstimmt, wird dies als korrekte Klassifikation betrachtet. Die Pixelgenauigkeit wird dann bestimmt, indem die Gesamtzahl der korrekt klassifizierten Pixel durch die Gesamtzahl der Pixel im Bild geteilt wird:

$$PA = \frac{\sum_{i=0}^{k} p_{ii}}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}} \tag{4.4}$$

Hierbei sind alle p_{ii} als True Positive anzusehen und alle p_{ij} als False Positive bzw. False Negative. [vgl. 45]

4.2. Beschreibung Experimente

Nachfolgend werden die verschiedenen Experimente bzw. Modelle beschrieben, sowie die Rahmenbedingungen festgelegt. Der verwendete Datensatz

enthält 25000 2D Bilder mit einer Größe von je 128x128 Pixeln. Die Vorverarbeitung beinhaltet das zusammenführen der vier Modalitäten, sowie ein Zuschneiden und Skalieren der einzelnen Bilder. Als Standardwerte für die jeweils nicht zu untersuchenden Parameter werden folgende Werte festgelegt: Die Auswertung der Modelle erfolgt über die drei genannten Evaluierungs-

Komponente	**Wert**
Batch Größe	32
Lernrate	0.01
Anzahl der Filter	64, 128, 256, 512
Vorverarbeitung	Auschnitt und Skalierung des Bildes
Optimierer	Adam
Verlustfunktion	Dice Loss

Tabelle 4.2.: Standardwerte für das Training des Neuronalen Netzes

funktionen. Die Ausgaben des Neuronalen Netzes werden zunächst mit Hilfe der Softmax-Funktion in den Wertebereich zwischen 0 und 1 überführt. Die neuen Werte stellen dabei eine Wahrscheinlichkeitsverteilung dar, wie bereits in Abschnitt 2.2.6 erwähnt. Anschließend werden diese Werte mit der Argmax-Funktion[1] in die eigentlichen Tumorklassen überführt.

4.2.1. 1. Experiment - Lernrate

Der erste zu untersuchende Parameter ist die Lernrate. Sie bestimmt wie stark die Gewichte innerhalb des Modells angepasst werden. Eine sehr hohe Lernrate kann dazu führen, dass wichtige Minima der Verlustfunktion übersprungen werden, während eine zu kleine Lernrate nur sehr langsam konvergiert. Im Folgenden wurde das Neuronale Netz mit verschiedenen Lernraten trainiert, wobei sowohl das Training als auch die abschließende Evaluation betrachtet. Die Lernrate wurde in diesem Versuch, ausgehend

[1] Identifiziert die Kategorie mit dem höchsten Wert und gibt somit die Klassifizierung an

vom Standardwert 0.01, jeweils einmal um den Faktor $1 \cdot 10^{-1}$ verkleinert und vergrößert, so dass für das Experiment die Lernraten 0.1 und 0.001 ergeben.

	Dice	Jaccard Index	Pixel Accuracy
LR0.1	81,49	79,45	98,79
Standard	88,44	85,85	99,22
LR0.001	**90,32**	**87,81**	**99,33**

Tabelle 4.3.: Ergebnisse des 1. Experiments mit der Lernrate in Prozent (Quelle: Eigene Darstellung)

4.2.2. 2. Experiment - Batch Größe

Der zweite Parameter ist die Batch Größe, welche angibt nach wie vielen Trainingsbeispielen die internen Parameter angepasst werden. Es wird untersucht, wie sich das Trainingsverhalten und die endgültige Leistung des Modells in Abhängigkeit der Batch Größe unterscheiden. Die zu untersuchenden Werte ergeben sich aus dem Standardwert, indem dieser einmal halbiert und einmal verdoppelt wird. Daraus ergeben sich somit die Batch Größen von 8 und 64.

	Dice	Jaccard Index	Pixel Accuracy
B8	83,03	81,08	98,88
Standard	**88,44**	**85,85**	**99,22**
B64	85,28	82,66	99,04

Tabelle 4.4.: Ergebnisse des 2. Experiments mit der Batch Größe in Prozent (Quelle: Eigene Darstellung)

4.2.3. 3. Experiment - Vorverarbeitung

Das letzte Experiment untersucht keinen Hyperparameter, sondern die Auswirkungen der Vorverarbeitungen auf die Leistung des Modells. Wie bereits in Abschnitt 3.1.2 erwähnt, müssen die Eingaben für das U-Net eine bestimmte Größe aufweisen, damit diese verarbeitet werden können. Die Größe der Bilder kann auf unterschiedliche Weise in die gewünschte Eingabegröße gebracht werden. Eine der Methoden ist eine einfache Skalierung von der Originalgröße auf die Zielgröße, dabei kommt es zu Verlusten der Bildqualität, da umliegende Pixel bzw. dessen Farbwerte miteinander verrechnet werden. Die zweite Methode, welche genauer in Abschnitt 3.1.2 beschrieben wurde, besteht darin zunächst die nicht relevanten Teile des Bildes zu entfernen und anschließend auf die gewünschte Zielgröße zu skalieren. Durch diese Methode enthält das Bild weniger irrelevante Informationen und erfährt einen geringeren Qualitätsverlust.

	Dice	Jaccard Index	Pixel Accuracy
Standard	**88,44**	**85,85**	**99,22**
Scaled	81,43	79,22	98,80

Tabelle 4.5.: Ergebnisse des 2. Experiments mit der Batch Größe in Prozent (Quelle: Eigene Darstellung)

4.3. Diskussion der Ergebnisse

Im folgenden Abschnitt werden die Ergebnisse der Experiment genauer betrachtet. Sowohl wie Trainingsverläufe, als auch die Resultate der Evaluation des Modells, über den Testdatensatz, werden genauer beschrieben. Bei den Trainingsverläufen stellt immer die blaue Kurve das Referenzmodell mit den Standardwerten dar. Es wird dabei außerdem auf die Auswirkungen der verschiedenen Parameter, so wie die Aussagekraft der Metriken eingegangen.

4.3.1. 1. Experiment

Das erste Experiment untersuchte die Lernrate im Hinblick auf das Training und die Leistung des Modells. Die Lernrate ist einer der wichtigsten Hyperparameter den es beim Training zu konfigurieren gilt. Sie bestimmt, wie sehr die internen Gewichte angepasst werden. Zunächst wird der Verlauf des Trainings betrachtet, welcher in Anhang A zu sehen ist. Zuerst wird dabei auf den Verlust im Trainingsdatensatz eingegangen, welcher über 20 Epochen protokolliert wurde. Die grüne Kurve, welche das Modell "LR0.1" mit der Lernrate 0.1 beschreibt, startet direkt von Anfang an mit weniger Verlust als das Referenzmodell, aber verringert diesen über die Zeit nur noch minimal. Möglicherweise werden hier wichtige Minima der Verlustfunktion übersprungen, weshalb das Modell "LR0.1" hier einen relativ hohen Verlust hat.

Im Vergleich dazu hat das Modell "LR0.001", die rote Kurve, mit einer Lernrate von 0.001, zu Beginn einen höheren Verlust, welcher über ein einige Epochen so bleibt und dann schlagartig fällt. Der Verlust von Modell "LR0.001" unterschreitet den vom Referenzmodell und hat am Ende der 20 Epochen einen geringeren Verlust als "Standard". Durch die geringe Lernrate von Modell "LR0.001" sinkt der Verlust zu Beginn langsam, da die internen Parameter nur minimal angepasst werden. Auf Dauer führen diese kleinen Schritte jedoch zu einem besseren Ergebnis, da wichtige Minima der Verlustfunktion weniger, bis gar nicht übersprungen werden.

Die Ergebnisse der Metriken unterstreichen den Verlauf des Trainings, so hat Modell "LR0.001" die beste Leistung erbracht. Ein Blick auf die Metriken ergibt einen Dice Koeeffizenten von über 90% und einen Jaccard Index von 87.81%, was recht gute Ergebnisse sind. Die Pixel Accuracy hingegen ist eher unaussagekräftig, da sich die Werte nur minimal mit unterscheiden mit einer Differenz von 0.73%.

4.3.2. 2. Experiment

4.3.3. 3. Experiment

4.4. Fehleranalyse und Verbesserungen

5. Resultate

6. Zusammenfassung und Ausblick

Literatur

1. O.V. *Was ist ein Algorithmus?* [online]. 2022-01-16. [besucht am 22. Feb. 2023]. Abger. unter: https://www.itwissen.info/Algorithmus-algorithm.html.
2. O.V. *Ionisierende Strahlung* [online]. 2020-06-05. [besucht am 13. März 2023]. Abger. unter: https://www.bmuv.de/themen/atomenergie-strahlenschutz/strahlenschutz/ionisierende-strahlung.
3. O.V. *Was ist ein Machine Learning Modell?* [online]. [besucht am 26. Feb. 2023]. Abger. unter: https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model.
4. HEINRICHS, Jan-Hendrik; CASPERS, Svenja; SCHNITZLER, Alfons; SEITZ, Frederike. *Bildgebung in den Neurowissenschaften*. Verlag Karl Alber, 2022. Auch verfügbar unter: https://www.ebook.de/de/product/45689204/jan_hendrik_heinrichs_svenja_caspers_alfons_schnitzler_frederike_seitz_bildgebung_in_den_neurowissenschaften.html.
5. RAMSUNDAR, Bharath; EASTMAN, Peter; WALTERS, Patrick; PANDE, Vijay. *Deep Learning für die Biowissenschaften*. Dpunkt.Verlag GmbH, 2020. Auch verfügbar unter: https://www.ebook.de/de/product/38103997/bharath_ramsundar_peter_eastman_patrick_walters_vijay_pande_deep_learning_fuer_die_biowissenschaften.html.
6. O.V. *PyTorch* [online]. [besucht am 2. Apr. 2023]. Abger. unter: https://pytorch.org/.

7. ERTEL, Wolfgang. *Grundkurs Künstliche Intelligenz*. Springer Fachmedien Wiesbaden, 2021. Auch verfügbar unter: https://www.ebook.de/de/product/41631013/wolfgang_ertel_grundkurs_kuenstliche_intelligenz.html.

8. POSTHOFF, Christian. *Computer und Künstliche Intelligenz: Vergangenheit - Gegenwart - Zukunft*. Springer Fachmedien Wiesbaden GmbH, 2022.

9. LANG, Volker. *Digitale Kompetenz*. Springer Berlin Heidelberg, 2023. Auch verfügbar unter: https://www.ebook.de/de/product/45515417/volker_lang_digitale_kompetenz.html.

10. ONLINE, Brockhaus Enzyklopädie. *Künstliche Intelligenz* [online]. [besucht am 20. Feb. 2023]. Abger. unter: https://brockhaus.de/ecs/enzy/article/kunstliche-intelligenz.

11. ALPAYDIN, Ethem. *Introduction to Machine Learning*. The MIT Press, 2014.

12. AGOSTINELLI, Andrea; DENK, Timo I.; BORSOS, Zalán; ENGEL, Jesse; VERZETTI, Mauro; CAILLON, Antoine; HUANG, Qingqing; JANSEN, Aren; ROBERTS, Adam; TAGLIASACCHI, Marco; SHARIFI, Matt; ZEGHIDOUR, Neil; FRANK, Christian. *MusicLM: Generating Music From Text*. arXiv, 2023. Auch verfügbar unter: https://arxiv.org/abs/2301.11325.

13. FROCHTE, Jörg. *Maschinelles Lernen*. Hanser, Carl GmbH + Co., 2020. Auch verfügbar unter: https://www.ebook.de/de/product/39878236/joerg_frochte_maschinelles_lernen.html.

14. O.V. *Supervised Learning* [online]. [besucht am 26. Feb. 2023]. Abger. unter: https://www.ibm.com/de-de/topics/supervised-learning.

15. JÜRGEN CLEVE, Uwe Lämmel. *Künstliche Intelligenz*. Hanser, Carl GmbH + Co., 2020. Auch verfügbar unter: https://www.ebook.de/de/product/38679436/juergen_cleve_uwe_laemmel_kuenstliche_intelligenz.html.

16. SCHERER, Andreas. *Neuronale Netze Grundlagen und Anwendungen: Grundlagen und Anwendungen*. Vieweg+Teubner Verlag, 1997.

17. CHOO, Kenny; GREPLOVA, Eliska; FISCHER, Mark H.; NEUPERT, Titus. *Machine Learning kompakt*. Springer Fachmedien Wiesbaden, 2020.

18. RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. *Nature*. 1986, **323**(6088), 533–536.

19. FRICK, Detlev. *Data Science Konzepte, Erfahrungen, Fallstudien und Praxis: Konzepte, Erfahrungen, Fallstudien und Praxis*. Springer Fachmedien Wiesbaden GmbH, 2021.

20. IAN GOODFELLOW Yoshua Bengio, Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

21. KINGMA, Diederik P.; BA, Jimmy. *Adam: A Method for Stochastic Optimization*. arXiv, 2014.

22. TEOH, Teik Toe. *Convolutional Neural Networks for Medical Applications*. Springer, 2023.

23. WEIDMAN, Seth. *Deep Learning - Grundlagen und Implementierung : Neuronale Netze mit Python und PyTorch programmieren: Neuronale Netze mit Python und PyTorch programmieren*. Dpunkt.Verlag GmbH, 2020.

24. KRAMME, Rüdiger. *Medizintechnik Verfahren - Systeme - Informationsverarbeitung: Verfahren - Systeme - Informationsverarbeitung*. Springer Berlin / Heidelberg, 2016.

25. CHRISTOPH PABST, Dr. med. *Magnetresonanz-Tomographie: Lernskript für Mediziner* [Grundlagen der Magnetresonanz-Tomographie] [online]. 2013. [besucht am 13. März 2023]. Abger. unter: https://www.ukgm.de/ugm_2/deu/umr_rdi/Teaser/Grundlagen_der_Magnetresonanztomographie_MRT_2013.pdf.

26. ANTWERPES, Dr. Frank. *Kernspintomographie* [online]. 2022-11-05. [besucht am 13. März 2023]. Abger. unter: https://flexikon.doccheck.com/de/Kernspintomographie.

27. GIZEWSKI, E. RM. R. Epidermoid oder Arachnoidalzyste: CISS, FLAIR und Diffusionsbilder als Ausweg aus dem diagnostischen Dilemma. *RöFo - Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebenden Verfahren*. 2001, **173**(1), 77–78.

28. O.V. *Brain Tumor AI Challenge (2021)* [online]. 2021. [besucht am 24. Apr. 2023]. Abger. unter: https://www.rsna.org/education/ai-resources-and-training/ai-image-challenge/brain-tumor-ai-challenge-2021.

29. NAM, Yeo Kyung; PARK, Ji Eun; PARK, Seo Young; LEE, Minkyoung; KIM, Minjae; NAM, Soo Jung; KIM, Ho Sung. Reproducible imaging-based prediction of molecular subtype and risk stratification of gliomas across different experience levels using a structured reporting system. *European Radiology*. 2021, **31**(10), 7374–7385.

30. BAID, Ujjwal u. a. The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification. 2021. Abger. unter arXiv: 2107.02314 [cs.CV].

31. GROUP, Data Format Working. *NIfTI (Neuroimaging Informatics Technology Initiative)* [online]. 2013-12. [besucht am 27. Apr. 2023]. Abger. unter: https://nifti.nimh.nih.gov/.

32. SWIEBOCKA-WIEK, Joanna. Skull Stripping for MRI Images Using Morphological Operators. In: SAEED Khalidand Homenda, Władysław (Hrsg.). *Computer Information Systems and Industrial Management*. Cham: Springer International Publishing, 2016.

33. SHORTEN, Connor; KHOSHGOFTAAR, Taghi M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*. 2019, **6**(1).

34. STEVENS, Eli Eli; ANTIGA, Luca Luca; VIEHMANN, Thomas Thomas. *Deep Learning with Pytorch*. Manning Publications Company, 2020.

35. RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. Abger. unter arXiv: 1505.04597 [cs.CV].

36. PFANNSTIEL, Mario A. (Hrsg.). *Künstliche Intelligenz im Gesundheitswesen*. Springer Fachmedien Wiesbaden, 2022.

37. YU, Tong; ZHU, Hong. Hyper-Parameter Optimization: A Review of Algorithms and Applications. 2020. Abger. unter arXiv: 2003.05689 [cs.LG].

38. IOFFE, Sergey; SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015. Abger. unter arXiv: 1502.03167 [cs.LG].

39. MURPHY, Kevin P. *Machine learning a probabilistic perspective: a probabilistic perspective*. MIT Press, 2012.

40. YEUNG, Michael; SALA, Evis; SCHÖNLIEB, Carola-Bibiane; RUNDO, Leonardo. *Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation*. arXiv, 2021.

41. SUDRE, Carole H; LI, Wenqi; VERCAUTEREN, Tom; OURSELIN, Sébastien; CARDOSO, M. Jorge. Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. 2017, 240–248. Abger. unter arXiv: 1707.03237 [cs.CV].

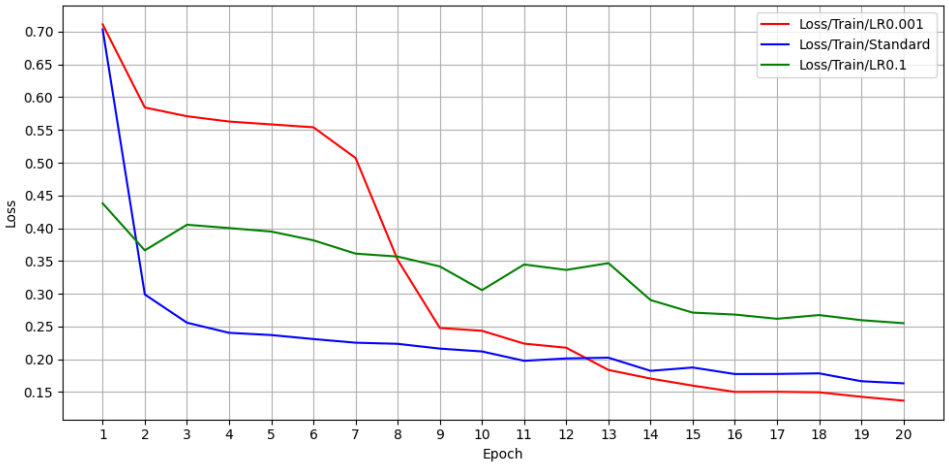
42. JADON, Shruti. A survey of loss functions for semantic segmentation. *2020 IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology*. 2020. Abger. unter arXiv: 2006.14822 [eess.IV].

43. GARCIA-GARCIA, Alberto; ORTS-ESCOLANO, Sergio; OPREA, Sergiu; VILLENA-MARTINEZ, Victor; GARCIA-RODRIGUEZ, Jose. A Review on Deep Learning Techniques Applied to Semantic Segmentation. 2017. Abger. unter arXiv: 1704.06857 [cs.CV].

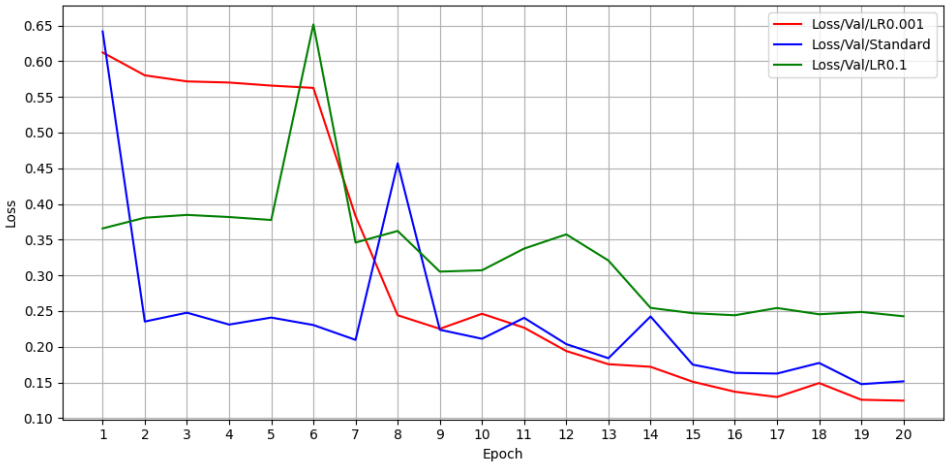
44. DICE, Lee R. Measures of the Amount of Ecologic Association Between Species. *Ecology*. 1945, **26**(3), 297–302.

45. HURTADO, Juana Valeria; VALADA, Abhinav. Semantic scene segmentation for robotics. In: *Deep Learning for Robot Perception and Cognition*. Elsevier, 2022, S. 279–311.

A. Experiment 1 - Verlustkurven



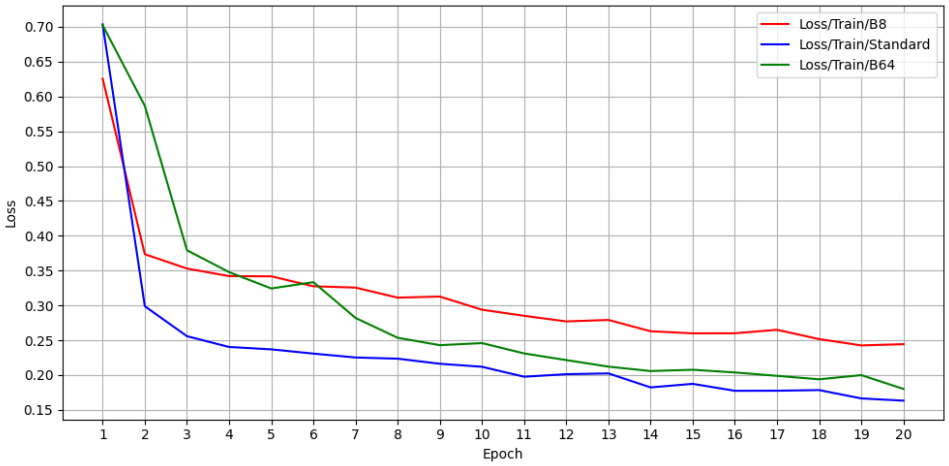
(a) Trainingsverlust



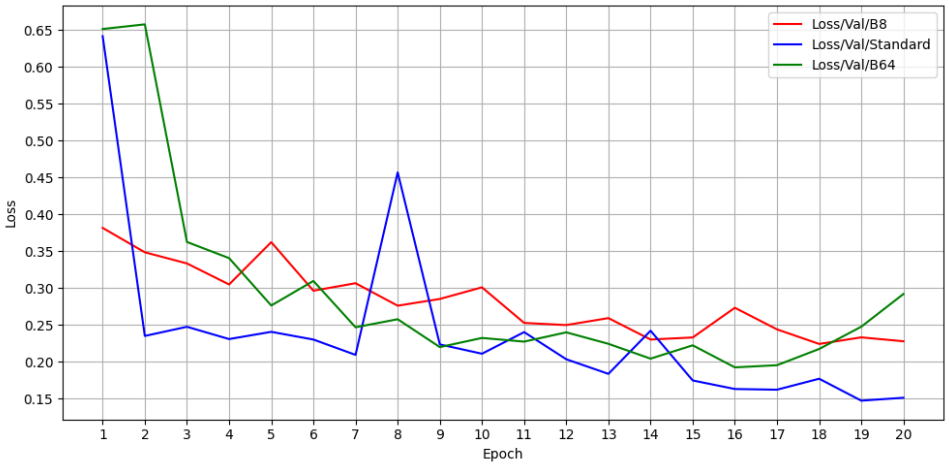
(b) Validierungsverlust

Abbildung A.1.: Experiment 1 - Verlustkurven von Training und Validierung über 20 Epochen, mit den Lernraten 0.1, 0.01(Standard) und 0.001 (Quelle: Eigene Darstellung)

B. Experiment 2 - Verlustkurven



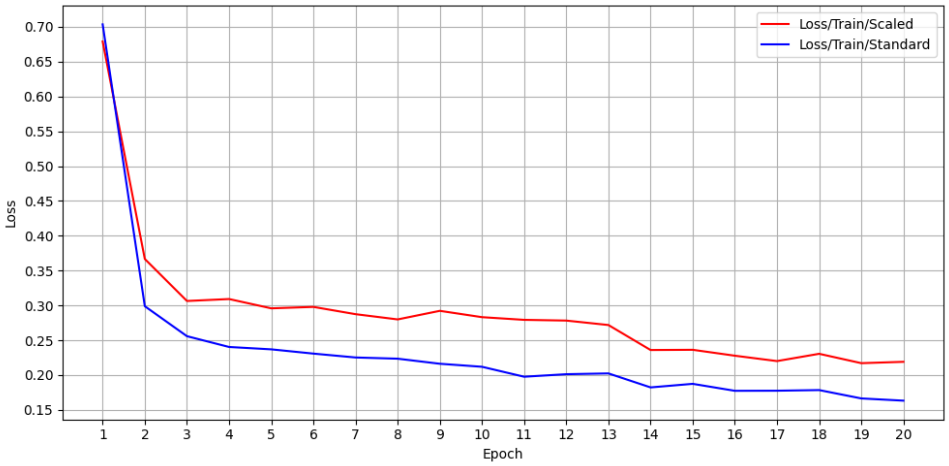
(a) Trainingsverlust



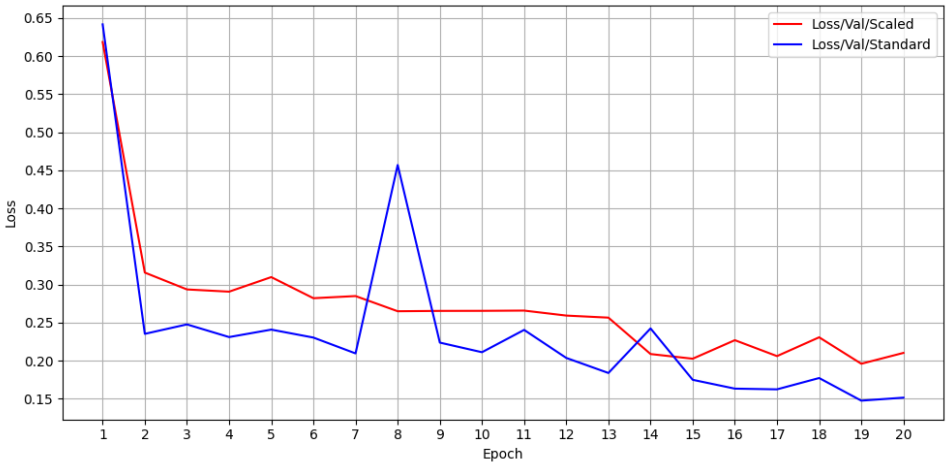
(b) Validierungsverlust

Abbildung B.1.: Experiment 2 - Verlustkurven von Training und Validierung über 20 Epochen, mit den Batch Größen 8, 32(Standard) und 64 (Quelle: Eigene Darstellung)

C. Experiment 3 - Verlustkurven



(a) Trainingsverlust



(b) Validierungsverlust

Abbildung C.1.: Experiment 3 - Verlustkurven von Training und Validierung über 20 Epochen, mit je Zuschneiden (Standard) und Größenskalierung (Scaled) (Quelle: Eigene Darstellung)