Klassifizierung von Hirntumoren mit Algorithmen des Deep Learning

Freie wissenschaftliche Arbeit

zur Erlangung des akademischen Grades

„Bachelor of Science“

Studiengang: Betriebswirtschaftslehre

an der Wirtschaftswissenschaftlichen Fakultät der Universität Augsburg

Lehrstuhl für Statistik

Eingereicht bei: Prof. Dr. Yarema Okhrin

Betreuerin: MSc. Sara Garber

Vorgelegt von: Hanan Loulou

Augsburg, im Dezember 2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CCE Categorical Cross Entropy

CR Classification Report

CM Confusion Matrix

CNN Convolutional Neural Network

CT Computertomographie

DL Deep Learning

DNN Deep Neural Network

DoF degrees of freedom

DS Datensatz

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

FT Fine-Tuning

GA Gentischer Algorithmus

GD Gradient Descent

GS Grid Search

HT Hirntumoren

HP Hyperparameter

HPO Hyperparameteroptimierung

ILSVRC ImageNet Large Scale Visual Recognition Challenge

ML Machine Learning

MLP Multi-Layer-Perzeptron

MRT Magnetresonanztomographie

MSE Mean Squared Error

PSO Particle Swarm Optimization

RS Random Search

SVM Support Vector Machine

TL Transfer Learning

TN True Negative

TNR True Negative Rate

TP True Positive

TPR True Positive Rate

Disposition

Im Rahmen dieser Arbeit erfolgt eine Untersuchung des Themas **„Klassifizierung von Hirntumoren mit Algorithmen des Deep Learning“**. Das Ziel dieser Untersuchung ist die Analyse der Performance von **Convolutional Neural Network (CNN)** bei der Klassifizierung von Hirntumoren (HT) anhand von Magnetresonanztomographie (MRT)-Bildern.
Die zentrale Fragestellung lautet: Welcher Unterschied besteht in der Performance zwischen einem selbst entwickelten CNN-Modell und einem vortrainierten Modell bei der Klassifizierung von HT. Dabei wird evaluiert, ob ein durch **Fine-Tuning (FT)** an die MRT-Daten optimiertes vortrainiertes CNN-Modell eine höhere Klassifikationsgenauigkeit erzielt als ein neu entwickeltes und vollständig trainiertes CNN. Im Rahmen des FT erfolgt die Feinabstimmung eines vortrainierten Modells auf einen spezifischen Datensatz durch Anpassung tiefer Schichten.
Die Hypothese besagt, dass **Transfer Learning (TL)**, bei dem ein Modell, das zuvor auf einem umfangreichen Datensatz wie **ImageNet** trainiert wurde, in Kombination mit FT eine signifikante Verbesserung der Klassifikationsleistung bewirkt.
Zur Beantwortung der Fragestellung wird der öffentlich zugängliche Datensatz (DS) Nickparvar (2021) verwendet, der **7.023 MRT-Bilder** verschiedener HT sowie nicht tumoröser Gewebe enthält. Die Methodik beinhaltet sowohl die Implementierung eines eigenen CNN als auch die Anpassung des vortrainierten ResNet-50-Modells von He et al. (2016) an dem Datensatz. Die Modelle werden anhand von Metriken wie **Accuracy**, **Recall**, **Preciscion** und **F1-Score** evaluiert, ergänzt durch **Konfusionsmatrizen**. Die Implementierung wird in **Python** unter Verwendung von Bibliotheken wie **TensorFlow** und **Keras** durchgeführt.
Die Arbeit ist in mehrere Abschnitte gegliedert. Nach der Einführung in das Thema sowie einem Literaturüberblick erfolgt die Erläuterung des theoretischen Hintergrunds der Modelle und der Evaluierungsmethoden. Der methodische Teil beschreibt den Datensatz und die Modellierungsansätze begleitet von Architekturdiagrammen. Daraufhin werden die Resultate präsentiert und analysiert sowie im Kontext der formulierten Forschungsfragen und Hypothesen diskutiert. Abschließend werden die wesentlichen Erkenntnisse zusammengefasst und künftige Forschungsperspektiven in diesem Bereich erörtert.
Das Literaturverzeichnis enthält aktuelle, relevante und zentrale wissenschaftliche Artikel und Bücher, die als theoretische und methodische Grundlage dienen.

1 Einführung

In den vergangenen Jahren hat die rapide Entwicklung des Machine Learning (ML) auf globaler Ebene zu einem Paradigmenwechsel geführt. Als besonders bemerkenswert erweist sich die Anwendung von ML in der medizinischen Bildgebung, welche eine präzise und effiziente Diagnostik ermöglicht. Algorithmen des Deep Learning (DL), insbesondere Convolutional Neural Network (CNN), haben sich dabei als zentrale Technologie etabliert, da sie in der Lage sind, komplexe Muster in Bilddaten zu erkennen und zu klassifizieren. Diese Fortschritte eröffnen neue Perspektiven für die Diagnostik von Krankheiten, die eine schnelle und zuverlässige Analyse erfordern. (Vgl. Deprez und Robinson 2023, Vorwort).

Hirntumoren (HT) stellen ein erhebliches medizinisches Herausforderungsgebiet dar. Weltweit wird ihre Inzidenzrate mit 3,5 und die Mortalitätsrate mit 2,6 pro 100.000 Menschen angegeben (vgl. Ferlay et al. 2024). Eine präzise Differenzierung zwischen den häufigsten Arten wie Gliomen (engl. *glioma*), Meningeomen (engl. *meningioma*) und Hypophysentumoren (engl. *pituitary*), ist von grundlegender Bedeutung für die Auswahl einer geeigneten Therapie (vgl. Bilsky 2023). Die Abbildung 1.1 veranschaulicht eine Auswahl von MRT-Aufnahmen der drei HT-Arten sowie eine MRT-Aufnahme nicht tumorösen Gewebes (engl. *no tumor*):

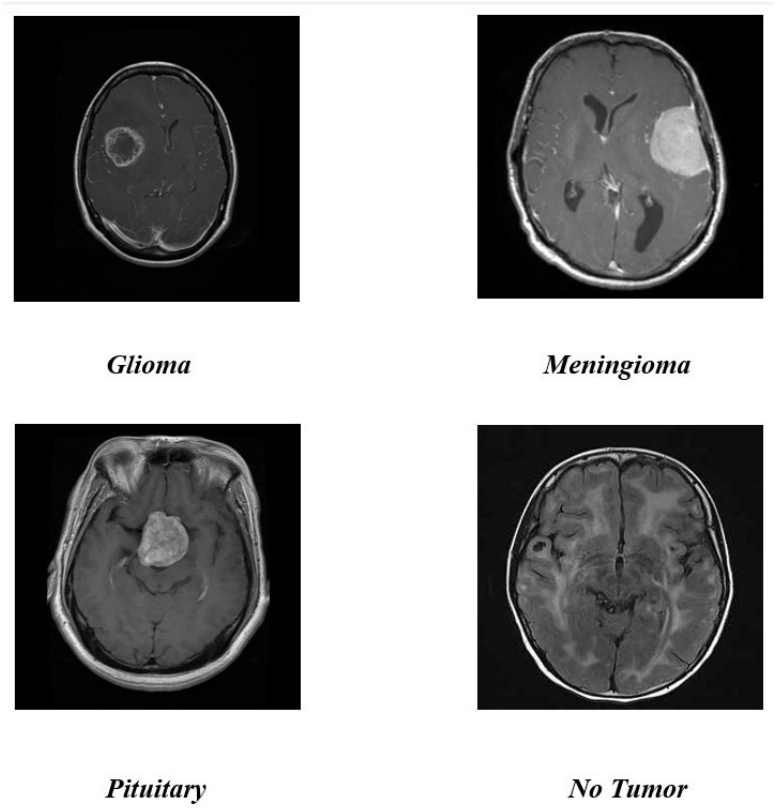


Abbildung 1.1: Auswahl von MRT-Bildern der vier Aufnahmearten.

Die Diagnose von Hirntumoren basiert traditionell auf manuellen Analysen von Aufnahmen der Magnetresonanztomographie (MRT) und Computertomographie (CT) durch Radiologen (vgl. Özkaraca et al. 2023). Der manuellen Überprüfung medizinischer Bilder ist jedoch ein hoher Zeitaufwand verbunden, zudem ist sie aufgrund des Patientenflusses potenziell fehleranfällig. Um dieses Problem zu lösen, ist die Entwicklung und Anwendung eines automatischen Bildanalyseverfahrens erforderlich, welches die Arbeitsbelastung bei der Klassifizierung und Diagnose von MRT-Bildern des Gehirns reduziert, als Hilfsmittel für Radiologen und Ärzte dient und die Mortalität durch die Früherkennung von Tumoren senken kann (vgl. Deprez und Robinson 2023, Vorwort).
In diesem Kontext hat sich DL als vielversprechende Technologie erwiesen. CNNs, die bereits bei Bildklassifikationsaufgaben wie der ImageNet Large Scale Visual Recognition Challenge (ILSVRC) Krizhevsky et al. (2012) signifikante Erfolge erzielt haben, bieten auch im medizinischen Bereich ein beträchtliches Potenzial (vgl. Lundervold und Lundervold 2019, S. 103). Die Fähigkeit zur automatischen Merkmalsextraktion stellt einen zentralen Vorteil von CNN dar. Im Gegensatz zu klassischen Methoden erfordert ihre Architektur keine manuelle Feature-Definition, sondern extrahiert komplexe Merkmale direkt aus den Daten (vgl. Lundervold und Lundervold 2019, S. 105). Dadurch sind sie für Aufgaben der medizinischen Bildverarbeitung geeignet, bei denen relevante Muster oft subtil und mehrdimensional sind.
Der Einsatz von CNNs ist jedoch mit Herausforderungen verbunden. Sie weisen einen hohen Rechenaufwand auf und sind insbesondere bei kleineren Datensätzen, die in der medizinischen Domäne häufig auftreten, anfällig für eine zu starke Anpassung an die Trainingsdaten. Dies kann zu einer eingeschränkten Generalisierungsfähigkeit auf neue Daten führen. (Vgl. Géron 2019, S. 28; 456).
Vortrainierte CNN-Modelle, die im Rahmen des Transfer Learning (TL) verwendet werden, stellen eine flexible Alternative dar. Sie ermöglichen die Übertragung bereits erlernter Merkmale aus großen Bilddatensätzen wie ImageNet Deng et al. (2009) auf spezifische medizinische Aufgaben, wodurch der Bedarf an großen Datensätzen und Trainingsressourcen reduziert wird. Obwohl dies den Trainingsaufwand reduziert, könnte die Anpassungsfähigkeit solcher Modelle auf domänenspezifische Daten eingeschränkt sein. (Vgl. Géron 2019, S. 345-347).
Die vorliegende Arbeit befasst sich mit der Untersuchung des Einsatzes von CNNs zur Klassifikation von HT auf MRT-Bildern. In diesem Kontext erfolgt eine Evaluierung eines selbst entwickelten, optimierten Modells sowie des vortrainierten Modells ResNet-50 von He et al. (2016). Ziel ist der Vergleich und die Analyse ihrer Performances anhand von verschiedenen Metriken.

2 Literaturüberblick

Der Forschungsstand zur Erkennung von HT mittels CNNs zeigt eine Vielzahl an Ansätzen, die gezielt auf spezifische Herausforderungen und Datensätze abgestimmt sind. Seetha und Raja (2018) demonstrierten die Effektivität vortrainierter Modelle mit eingefrorenen tiefen Schichten. Diese erwiesen sich im Vergleich zu traditionellen Modellen wie Deep Neural Network (DNN) und Support Vector Machine (SVM) als überlegen.
In einer weiteren Studie untersuchten Kabir et al. (2018) die Klassifikation verschiedener Gliomgrade. Dazu kombinierten sie CNNs mit Gentischer Algorithmus (GA), um die Leistung der Modellarchitektur zu optimieren. Kebir und Mekaoui (2018) evaluierten CNNs in Kombination mit einem k-Means-Algorithmus zur Klassifikation und Segmentierung eines klinischen DS.
Anaraki et al. (2019) integrierten GA in den Trainingsprozess, um die CNN-Strukturen zu optimieren und konnten präzise Klassifikationsergebnisse erzielen. Deepak und Ameer (2019) hingegen verfolgten einen Ansatz des TL, wobei GoogleNet als Feature-Extraktor für die Klassifikation mit kleinerem DS verwendet wurde.
Ismael et al. (2020) konnten die Genauigkeit durch den Einsatz von Residual Networks (ResNet50) und einer Datenaugmentation verbessern. Naser und Deen (2020) demonstrierten das Potential von U-Net und VGG16 für die Segmentierung und Klassifikation niedriggradiger Gliome (LGG).
Mehrotra et al. (2020) untersuchten verschiedene vortrainierte Modelle und optimierten die Klassifikationsleistung durch TL. Sharif et al. (2020) kombinierten DL mit Feature-Fusion-Techniken und optimierten die Ergebnisse mit Particle Swarm Optimization (PSO). Ein ensemble-basierter Ansatz wurde von Tandel et al. (2021) entwickelt, welcher sowohl DL- als auch ML-Modelle integrierte, um eine Mehrheitswahl-basierte Klassifikation zu erreichen.
In ihrer Untersuchung setzten GabAllah et al. (2021) VGG19 als Feature-Extraktor ein und nutzten Bildaugmentationstechniken wie PGGAN. Dutta und Nayak (2022) führten mit CDANet eine duale Aufmerksamkeitsstruktur ein, um fokussierte Merkmale aus MRT-Bildern mit höherer Präzision zu erfassen.
Altahhan et al. (2023) entwickelten hybride CNN-Modelle, welche DL- und ML-Klassifikatoren kombinierten. Die Optimierung von AlexNet in Verbindung mit SVM und KNN zielte auf eine Erhöhung der Genauigkeit ab. In einer späteren Studie präsentierten Ravinder et al. (2023) ein graphbasiertes CNN-Modell (GCNN), welches die Berücksichtigung nicht-euklidischer Distanzen in Bilddaten ermöglicht.
Die in Tabelle 2.1 sowie in Tabelle 2.2 zusammengefassten Studien verdeutlichen die Anpassungsfähigkeit von CNN-Modellen an spezifische Anforderungen, einschließlich GA, hybrider Architekturen und TL. Sie veranschaulichen sowohl Fortschritte als auch bestehende Herausforderungen in der medizinischen Bildverarbeitung.

Autor(Jahr)	Aufgabe	Daten	Modellarchitektur	Accuracy	Anmerkungen
Seetha und Raja (2018)	Klassifikation	BRATS (2015) & Radiopaedia	Vortrainierte CNNs	97.5%	/
Kabir et al. (2018)	Klassifikation & Einstufung	REMBRANDT Scarpace et al. (2015)	Optimiertes CNN	94.2%	GA
Kebir und Mekaoui (2018)	Klassifikation & Segmentierung	privater klinischer DS	Optimiertes CNN	95%	K-Mean Algorithmus
Anaraki et al. (2019)	Klassifikation & Einstufung	IXI (2015), TCIA Clark et al. (2013), REMBRANDT Scarpace et al. (2015), TCGA-GBM Scarpace et al. (2016) & TCGA-LGG Pedano et al. (2016)	Optimiertes CNN	90.9% 94.2%	GA
Deepak und Ameer (2019)	Klassifikation	Figshare Cheng (2017)	GoogleNet	93%	/
Ismael et al. (2020)	Klassifikation	Figshare Cheng (2017)	ResNet-50	97%	Datenaugmentation
Naser und Deen (2020)	Segmentierung & Einstufung	TCIA Clark et al. (2013)	U-net & VGG-16	92.0% 89.0%	/

Tabelle 2.1: Überblick über relevante Arbeiten zur Erkennung von Hirntumoren mittels CNNs I.

Autor(Jahr)	Aufgabe	Daten	Modellarchitektur	Accuracy	Anmerkungen
Mehrotra et al. (2020)	Klassifikation	TCIA Clark et al. (2013)	AlexNet, GoogLeNet ResNet-50, ResNet-101 & SqueezeNet	96.8% (AlexNet)	Binäre Klassifikation
Sharif et al. (2020)	Klassifikation & Segmentierung	BRATS (2013), BRATS (2014) BRATS Bakas et al. (2017) & BRATS Bakas et al. (2018)	Inception V3	98.3%, 97.8%, 96.9% & 92.5%	PSO
Tandel et al. (2021)	Einstufung	REMBRANDT Scarpace et al. (2015)	AlexNet, VGG-16, ResNet-18, ResNet-50 & GoogleNet	97.1%	Mehrheitswahl-basierter Ensemble-Algorithmus
GabAllah et al. (2021)	Klassifikation	privater klinischer DS Cheng et al. (2015)	VGG-19	98.5%	PGGAN Datenaugmentation
Dutta und Nayak (2022)	Klassifikation	Figshare Cheng (2017)	CDANet	96.7%	/
Altahhan et al. (2023)	Klassifikation	Kaggle Nickparvar (2021)	AlexNet-KNN & AlexNet-SVM	95.0% 97.0%	Hybride DL-ML-Modelle
Ravinder et al. (2023)	Klassifikation	Kaggle Bhuvaji et al. (2020)	GCNN	95.0%	Graphbasiertes Modell

Tabelle 2.2: Überblick über relevante Arbeiten zur Erkennung von Hirntumoren mittels CNNs II.

3 Theorie hinter Deep Learning

Das DL stellt einen spezialisierten Teilbereich des ML dar, welcher auf der Verwendung tiefer neuronaler Netzwerke basiert. Die theoretischen Grundlagen für DL , insbesondere im Bereich neuronaler Architekturen und Lernalgorithmen, wurden von Lang (2023, S. 22-40) umfassend behandelt. In den folgenden Unterkapiteln (3.1 bis 3.9) wird ein Großteil dieser Ausführungen aufgegriffen und systematisch erläutert. Die zugrundeliegenden Formeln stammen hauptsächlich aus Langs Werk. Eine Ausnahme bildet Unterkapitel 3.5, in dem die Formeln und Ansätze auf den Arbeiten von Tiwari (2022) sowie Grandini et al. (2020) basieren.

3.1 Tiefe neuronale Netze

Der grundlegende Ansatz hinter DL-Algorithmen basiert auf der Idee, künstliche neuronale Netzwerke zu entwickeln, die die Funktionsweise des menschlichen Gehirns nachahmen (vgl. Géron 2019, S. 279). Dieses Konzept wurde von Rosenblatt (1958) durch das Perzeptron-Modell eingeführt, welches als probabilistisches Modell für die Informationsspeicherung und -organisation im Gehirn dient. Die Bildung von Verbindungen zwischen künstlichen Neuronen erfolgt in ähnlicher Weise wie die neuronale Kommunikation im Gehirn, nämlich durch wiederholtes Training und entsprechende Anpassung der Gewichtungen (vgl. Géron 2019, S. 286-287).
Ein künstliches Neuron multipliziert einen Eingangsvektor $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ mit einem Gewichtungsvektor $\mathbf{w} = (w_1, w_2, \ldots, w_n)$, um ein skalares Signal zu bilden und addiert ein Bias-Term b, um ein Logit zu erzeugen, das durch eine Aktivierungsfunktion f in das Ausgabesignal

$$y(\mathbf{x}; \mathbf{w}, b) = f(\mathbf{x} \cdot \mathbf{w} + b) \tag{3.1}$$

umgewandelt wird.
Rosenblatts Modell eines künstlichen Neurons, das Perzeptron, verwendete die Heaviside-Stufenfunktion als Aktivierungsfunktion und einen negativen Bias-Term, um eine binäre Ausgabe zu erzeugen:

$$y = \begin{cases} 1 & \text{wenn } \sum_j x_j \cdot w_j > b \\ 0 & \text{wenn } \sum_j x_j \cdot w_j \leq b. \end{cases} \tag{3.2}$$

Die Lösung komplexer Modellierungsprobleme erfordert die Verwendung komplexer Modelle, die über die Fähigkeiten eines einzelnen Perzeptrons hinausgehen. Eine Möglichkeit der Modellierung ist die Kombination mehrerer Perzeptrons zu einem Netzwerk, welches als Multi-Layer-Perzeptron (MLP) bezeichnet wird. Layer, welche nicht direkt mit der Eingabe oder Ausgabe verbunden sind, werden als versteckte Schichten (engl. *hidden layer*) bezeichnet. Modelle, die zwei oder mehr versteckte Schichten aufweisen, werden

als tiefe neuronale Netze (engl. *DNN*) bezeichnet. Die Neuronen der ersten versteckten Schicht interagieren direkt mit den Eingabemerkmalen, während spätere Schichten fortgeschrittene Mustererkennung ermöglichen. Dies führt zu einer progressiv komplexeren und präziseren Entscheidungsfindung. (Vgl. Lang 2023, S. 25).
In einem MLP ist jedes Neuron einer Schicht mit allen Neuronen der vorhergehenden und der nachfolgenden Schicht verbunden. Diese Art von Schicht wird als dicht (engl. *dense layer*) oder vollständig verbunden (engl. *fully connected*) bezeichnet. (Vgl. Géron 2019, S. 285).
Um das Netz zu trainieren und die optimalen Gewichtungen zu finden, wird ein Algorithmus verwendet, der die Gewichtungen schrittweise anpasst. Hierzu werden differenzierbare, kontinuierliche Aktivierungsfunktionen benötigt, da die Heaviside-Funktion nicht differenzierbar ist und somit ungeeignet wäre. Ein Netzwerk, das ausschließlich lineare Aktivierungsfunktionen verwendet, würde auf ein lineares Modell reduziert, wodurch seine Fähigkeit, komplexe Daten zu modellieren, stark eingeschränkt wäre. Daher weisen versteckte Schichten nichtlineare Aktivierungen auf, die es dem Netzwerk ermöglichen, komplexe Muster zu erkennen. (Vgl. Lang 2023, S. 26 sowie Géron 2019, S. 292).
In Abbildung 3.1 werden gebräuchliche Aktivierungsfunktionen und ihre Derivate dargestellt, um ihre Auswirkungen auf die Ausgabewerte zu veranschaulichen. Diese Funktionen dienen primär dazu, Nichtlinearität in neuronale Netze einzuführen und somit komplexere Muster zu modellieren.
Eine typische Aktivierungsfunktion für versteckte Neuronen ist die ReLU (Rectified Linear Unit):

$$\phi(x) = \max(0, x). \tag{3.3}$$

Die ReLU-Funktion erweist sich aufgrund ihrer Kombination von Einfachheit und Effizienz als besonders geeignet für die Verarbeitung großer Netzwerke. Die Funktion setzt negative Werte auf null und lässt positive Werte unverändert, wodurch eine schnelle Berechnung sowie ein sparsames Speicherverhalten ermöglicht werden. Bei tiefen Netzwerken führt die Deaktivierung negativer Eingaben zu einer Steigerung der Leistungsfähigkeit. (Vgl. Géron 2019, S. 292).
Die Wahl der Aktivierungsfunktion in der Ausgabeschicht hängt vom Modellierungsproblem ab. Für die binäre Klassifikation wird typischerweise die Sigmoidfunktion verwendet:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}. \tag{3.4}$$

Diese Funktion transformiert die Ausgabewerte in den Bereich zwischen 0 und 1, was sie ideal für Wahrscheinlichkeitsvorhersagen bei binären Problemen macht (vgl. Géron 2019, S. 143).
Während bei Klassifizierungsproblemen mit K verschiedenen Klassen häufig die Softmax-

Funktion verwendet wird:

$$\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}. \tag{3.5}$$

Die Softmax-Funktion normalisiert die Ausgaben, sodass sie eine Wahrscheinlichkeitsverteilung über die K Klassen darstellen, wobei die Summe der Wahrscheinlichkeiten den Wert 1 annimmt. Diese Eigenschaft qualifiziert sie insbesondere für die Lösung von Mehrklassenklassifikationsproblemen, da jede Ausgabe als Wahrscheinlichkeit für eine Klasse interpretiert werden kann. (Vgl. Géron 2019, S. 148-149).

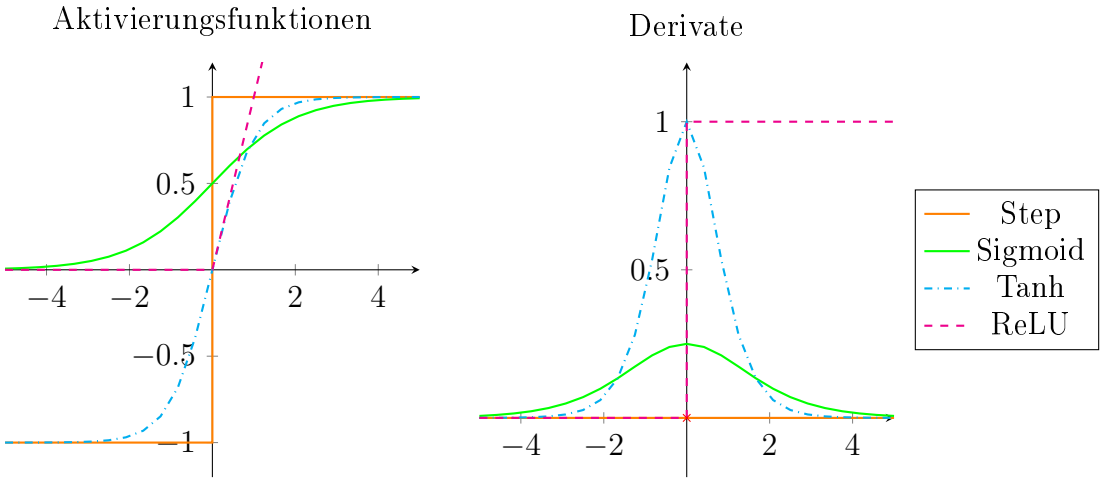


Abbildung 3.1: Aktivierungsfunktionen und deren Derivate, nach Géron (2019, S. 292).

3.2 Backpropagation & Gradientenabstieg

Die Gewichte eines DNN werden durch die Minimierung einer Verlustfunktion (engl. *loss function*) angepasst, um möglichst genaue Vorhersagen zu erzielen. Die Verlustfunktion ist entscheidend für die Optimierung der Modellparameter. (Vgl. Géron 2019, S. 113). Eine der grundlegenden Verlustfunktionen ist der mittlere quadratische Fehler (engl. *Mean Squared Error (MSE)*):

$$L_{\text{MSE}} = \frac{1}{2n} \sum_{x} \|y(x) - \tilde{y}\|^2, \tag{3.6}$$

wobei n die Anzahl der Trainingsbeispiele x und $\tilde{y}$ die zugehörigen tatsächlichen Labels darstellt, während $y(x)$ die Netzwerkausgabe repräsentiert.
Der Trainingsprozess beginnt mit der zufälligen Initialisierung der Gewichte, gefolgt von einer iterative Anpassung durch einen Optimierungsprozess zur Minimierung der Verlustfunktion (vgl. Géron 2019, S. 118). Laut Goodfellow et al. (2017, S. 83-85) ist der Gradientenabstieg (engl. *Gradient Descent (GD)*), der erstmals von Cauchy (1847) eingeführt wurde, einer der grundlegendsten Optimierungsalgorithmen. Der Algorithmus passt die Netzwerkparameter θ entlang des negativen Gradienten der Verlustfunktion

an. Die Größe der Anpassung wird durch die Lernrate ϵ bestimmt:

$$\Delta\theta \equiv -\epsilon \nabla_\theta L(\theta). \tag{3.7}$$

Im Verlauf jedes Optimierungsschritts werden die Netzwerkparameter gemäß der folgenden Regel aktualisiert:

$$\theta \leftarrow \theta - \epsilon \nabla_\theta L(\theta). \tag{3.8}$$

Bei Anwendung der MSE-Verlustfunktion (3.6) ergibt sich die Anpassung der Netzwerkparameter wie folgt:

$$\Delta\theta = -\epsilon \nabla_\theta \left(\frac{1}{2n} \sum_x \|y(x) - \tilde{y}\|^2 \right). \tag{3.9}$$

In Gleichung (3.9) wird die Summe über alle Trainingsbeispiele berechnet, weshalb jedes Eingangssignal durch das Netz geleitet werden muss, um die Ausgangssignale $y(x)$ zu berechnen. Erst im Anschluss kann eine Anpassung der Gewichte durch Gradientenabstieg erfolgen. Insbesondere bei großen Datensätzen ist dies ein rechnerisch aufwändiger Prozess. (Vgl. Lang 2023, S. 28).

Dieses Vorgehen wird durch den Backpropagation-Algorithmus unterstützt, welcher durch Rumelhart et al. (1986a) und Rumelhart et al. (1986b) entwickelt wurde. Der Prozess lässt sich in zwei Phasen unterteilen: Im ersten Schritt, dem sogenannten *„Forward Pass“*, wird jede Trainingsinstanz durch das Netzwerk geleitet, um Vorhersagen zu treffen. Im sogenannten *„Reverse Pass“* wird der durch den Vergleich der Vorhersage mit den tatsächlichen Werten entstandene Fehler durch die Schichten des Netzwerks zurückpropagiert. Auf diese Weise wird der Beitrag jeder Verbindung zum Fehler bestimmt, um die entsprechenden Gewichte anzupassen. Dieser Prozess wird als Gradientenabstiegsschritt (engl. *Gradient Descent step*) bezeichnet, bei dem die Gewichte in kleinen Schritten optimiert werden, um den Fehler zu minimieren. (Vgl. Géron 2019, S. 291).

Der hier beschriebene Algorithmus, der alle Trainingsdaten für jeden Optimierungsschritt verwendet, wird als Batch-Gradientenabstieg (engl. *Batch Gradient Descent*) bezeichnet (vgl. Géron 2019, S. 122). Obwohl dieser Ansatz präzise ist, kann er bei großen Datensätzen ineffizient sein und führt oft dazu, dass das Modell in lokalen Minima hängen bleibt (vgl. Géron 2019, S. 119). Daher wurde der Mini-Batch-Gradientenabstieg (engl. *Mini-batch Gradient Descent*) entwickelt, bei dem lediglich ein Teil der Daten in jedem Schritt Berücksichtigung findet, was den Rechenaufwand reduziert und zu einer stabileren Konvergenz führen kann (vgl. Géron 2019, S. 124-127). Die Anzahl der Beispiele, die den Algorithmen vor jedem Optimierungsschritt vorgelegt werden, die sogenannte Batchgröße (engl. *batch size*), beeinflusst die Optimierungsleistung und senkt die Berechnungsanforderungen (vgl. Géron 2019, S. 326).

Moderne Algorithmen wie der Adam-Optimierer fügen dem Gradientenabstieg einen

Impulsterm hinzu, um vergangene Gradienteninformationen in die Parameteraktualisierung einfließen zu lassen. Der Aktualisierungsschritt erfolgt folgendermaßen:

$$\begin{aligned} \mathbf{m} &\leftarrow \beta\mathbf{m} - \epsilon\nabla_\theta L(\theta) \\ \theta &\leftarrow \theta + \mathbf{m}, \end{aligned} \tag{3.10}$$

wobei $\mathbf{m}$ der Impulsterm ist und β ein Wert zwischen 0 und 1 darstellt, der die Gewichtung des Impulsterms bestimmt. Die Berücksichtigung früherer Gradienten ermöglicht eine effizientere und robustere Konvergenz als dies bei einfachen Gradientenabstiegsverfahren der Fall ist. (Vgl. Géron 2019, S. 351-352).

3.3 Modellparametrisierung & Generalisierung

Die Minimierung der Verlustfunktion auf den Trainingsdaten zielt darauf ab, das Modell zu optimieren, indem die Fehler auf den Trainingsdaten reduziert werden. Die Annahme basiert darauf, dass das Modell allgemeine Muster erlernt, welche auf das zugrunde liegende Problem übertragbar sind, sodass auch auf neuen, bisher nicht betrachteten Datensätzen adäquate Resultate erzielt werden. Diese Fähigkeit des Modells, auch auf neuen Daten eine gute Performance zu zeigen, wird als Generalisierung (engl. *generalization*) bezeichnet. Ein niedriger Trainingsfehler (engl. *training error*) indiziert eine effektive Performance des Modells auf den Trainingsdaten, während ein geringer Generalisierungsfehler (engl. *generalization error*) darauf hinweist, dass das Modell seine Leistung auch auf unabhängigen Daten beibehalten kann. Allerdings besteht häufig eine gewisse Spannung zwischen den Zielen der Optimierung und der Generalisierung, da eine Minimierung des Trainingsfehlers nicht zwangsläufig zu einer Verbesserung der Generalisierung führt. Das Ziel ist daher, einen Ausgleich zu finden, um sowohl auf den Trainings- als auch auf den Testdaten gute Ergebnisse zu erzielen. (Vgl. Chollet 2021, S. 128).

DNN verfügen über eine signifikante Anzahl an trainierbaren Parametern, die in vielen Fällen eine Größe von Zehntausenden bis zu Millionen aufweisen. Die hohe Anzahl an Parametern ermöglicht dem Modell eine hohe Flexibilität bei der Erfassung und Verarbeitung komplexer Datensätze. (Vgl. Géron 2019, S. 364). Diese Flexibilität ist auf die Freiheitsgrade (engl. *degrees of freedom (DoF)*) des Modells zurückzuführen, welche durch die Anzahl der Trainingsparameter definiert werden (vgl. Géron (2019, S. 28)). Eine hohe Anzahl an Freiheitsgraden birgt jedoch das Risiko einer Überanpassung (engl. *overfitting*), bei der das Modell spezifische Details der Trainingsdaten anstatt allgemeiner Muster erlernt. Dies kann zu einer Beeinträchtigung der Leistung des Modells auf neuen, ungesehenen Daten führen. (Vgl. Chollet 2021, S. 110-111). Demgegenüber kann ein Modell, welches über eine unzureichende Anzahl an Freiheitsgraden verfügt,

wesentliche Muster nicht adäquat erfassen, was als Unteranpassung (engl. *underfitting*) bezeichnet wird. (Vgl. Géron 2019, S. 29). Daher ist es von entscheidender Bedeutung, bei der Modellparametrisierung ein ausgewogenes Verhältnis zwischen der Optimierung der Leistung auf den Trainingsdaten und der Minimierung des Generalisierungsfehlers zu finden, um eine adäquate Leistung auf neuen Daten zu gewährleisten (vgl. Géron 2019, S. 28).
Zur Überwachung des Generalisierungsfehlers des Modells während des Trainings wird ein Teil der Trainingsdaten als Validierungsset verwendet. Das unabhängige Set, welches nicht in die Modellanpassung einfließt, findet nach jedem Optimierungsschritt bzw. jeder Trainingsepoche Anwendung, um eine unvoreingenommene Verlustfunktion und weitere Metriken zu berechnen. (Vgl. Chollet 2021, S. 102-104).
Die Generalisierungsdifferenz, definiert als die Abweichung der Modellleistung zwischen Trainings- und Validierungsset, erlaubt eine Beurteilung der Qualität der Generalisierung auf neue, bislang nicht betrachtete Daten. Eine hohe Abweichung kann auf eine Overfitting oder Underfitting hindeuten. Zur Optimierung der Modellleistung wird die Performance auf dem Validierungsset häufig zur Anpassung von Hyperparametern wie der Anzahl der Schichten, Neuronen oder der Lernrate genutzt, da diese direkt die Modellkapazität und Generalisierung beeinflussen. (Vgl. Lang 2023, S. 29).
Für die abschließende Bewertung der Modellleistung wird schließlich ein drittes, unabhängiges Testset verwendet. Das Testset dient der finalen Evaluierung und wird nicht für Training oder Validierung verwendet. Dies gewährleistet eine objektive Beurteilung der Modellleistung. Daher basiert das Training auf drei Datensätzen: dem Trainingsset zur Anpassung der Gewichte, dem Validierungsset zur Überwachung der Generalisierung und dem Testset zur abschließenden Bewertung. (Vgl. Chollet 2021, S. 151-152).

3.4 Problemformulierung

Die Auswahl einer passenden Verlustfunktion stellt einen wesentlichen Aspekt im Rahmen der Modellierung dar. Es existiert eine Vielzahl von Verlustfunktionen, die für spezifische Aufgabenbereiche adaptiert wurden. In der Praxis wird bei der Regression, bei der eine skalare Ausgabe modelliert wird, häufig die MSE-Verlustfunktion (3.6) verwendet. Im Kontext des Trainings von Klassifikationsmodellen kommt hingegen in der Regel die kategoriale Kreuzentropie (engl. *Categorical Cross Entropy (CCE)*) zum Einsatz:

$$L_{\text{CCE}} = -\sum_{i=1}^{n} y_i \log(p_i), \tag{3.11}$$

wobei y_i für das wahre Label, p_i für die Softmax-Wahrscheinlichkeit der Klasse i und n für die Gesamtanzahl der Klassen steht. (Vgl. Chollet 2021, S. 101-102). Die CCE-

Verlustfunktion bestraft falsche Vorhersagen stärker als der MSE, weshalb sie sich insbesondere für Klassifikationsaufgaben eignet (vgl. Géron 2019, S. 149).
Bei unbalancierten Datensätzen, deren Klassenverteilung eine unausgeglichene Verteilung aufweist, kann die Einführung eines Gewichtungstermins in die Verlustfunktion dazu beitragen, die Repräsentanz unterrepräsentierter Klassen zu verbessern. Dazu wird jeder Summand mit einem Gewichtungsfaktor w_i multipliziert, sodass unterrepräsentierte Klassen stärker berücksichtigt werden:

$$L_{\text{wCCE}} = -\sum_{i=1}^{n} w_i y_i \log(p_i). \tag{3.12}$$

(Vgl. Lang 2023, S. 30).

3.5 Leistungsbewertung

Neben der Verlustfunktion ist die Auswahl geeigneter Metriken für die Evaluierung der Modellleistung während der Validierung und des Tests von entscheidender Bedeutung. Diese Metriken spiegeln nicht nur die Genauigkeit des Modells wider, sondern auch dessen Fähigkeit, mit unausgewogenen Datensätzen umzugehen und auf unbekannte Daten zu generalisieren. (Vgl. Tiwari (2022, S. 34-35)).
Eine Konfusionsmatrix (engl. *Confusion Matrix (CM)*) stellt ein wesentliches Instrument zur Darstellung der tatsächlichen Labels im Verhältnis zu den prognostizierten Labels dar. Die Darstellung erfolgt in Form einer Matrix, in der die Anzahl der korrekt vorhergesagten positiven Fälle (engl. *True Positive (TP)*), der korrekt vorhergesagten negativen Fälle (engl. *True Negative (TN)*), der falsch als positiv vorhergesagten Fälle (engl. *False Positive (FP)*) sowie der falsch als negativ vorhergesagten Fälle (engl. *False Negative (FN)*) angezeigt wird. Aus der CM lassen sich mehrere Metriken ableiten, die zur Bewertung von Klassifikationsmodellen für binäre Klassen sowie für Mehrfachklassifikationen verwendet werden können. Die Abbildung 3.2 illustriert eine CM für binäre Klassifikationen. Bei einer Mehrklassenklassifikation wie in der Abbildung 3.3 wird die Matrix um die Vorhersagen für jede Klasse gegenüber den tatsächlichen Klassen erweitert, wodurch eine detaillierte Analyse der Modellleistung ermöglicht wird. (Vgl. Tiwari 2022, S. 52).

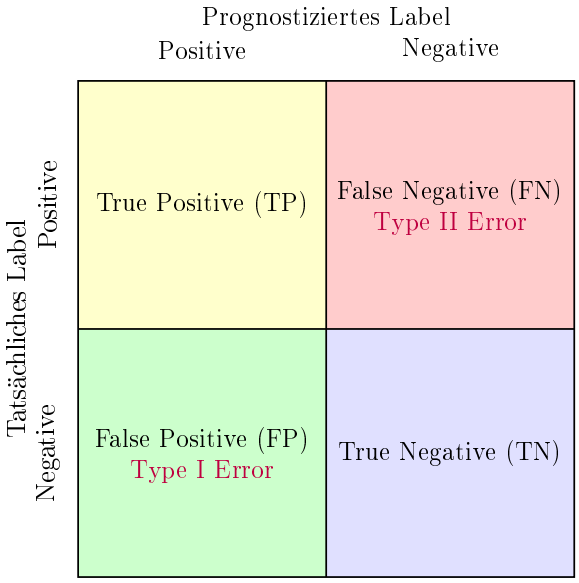


Abbildung 3.2: Binäre CM, nach Tiwari (2022, S. 52).

- ***Accuracy***:
  Die Accuracy (Genauigkeit) eines Modells gibt das Verhältnis zwischen der Gesamtzahl der vom Modell richtig prognostizierten Labels und der Gesamtzahl der Labels an.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{3.13}$$

  In Fällen einer unbalancierten Klassenverteilung kann die Accuracy jedoch irreführende Ergebnisse liefern, da sie durch überrepräsentierte Klassen dominiert werden kann (vgl. Tiwari 2022, S. 56).

- ***True Positive Rate (TPR)***:
  Die TPR beschreibt das Verhältnis der korrekt prognostizierten positiven Labels zur Gesamtzahl der tatsächlichen positiven Labels. Die TPR ist auch als ***Recall*** oder ***Sensitivity*** bekannt.

$$TPR = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.14}$$

- ***True Negative Rate (TNR)***:
  Die TNR beschreibt das Verhältnis der korrekt prognostizierten negativen Labels zur Gesamtzahl der tatsächlichen negativen Labels. Eine andere Bezeichnung für TNR ist die ***Specificity***.

$$TNR = \frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{TN}} \tag{3.15}$$

  Im Rahmen binärer Klassifikationsaufgaben, wie sie in zahlreichen klinischen Tests zum Einsatz gelangen, sind die Metriken Sensitivität und Spezifität von entscheidender Bedeutung. Sie stehen proportional zur Güte des Modells (vgl. Lang 2023, S. 30-31).

- ***False Negative Rate (FNR)***:
  Die FNR stellt das Verhältnis der falsch vorhergesagten positiven Labels zur Gesamtzahl der tatsächlichen positiven Labels dar.

$$FNR = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{3.16}$$

- ***False Positive Rate (FPR)***:
  Die FPR gibt das Verhältnis der falsch vorhergesagten negativen Labels zur Gesamtzahl der tatsächlichen negativen Labels an.

$$FPR = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3.17}$$

  Die Güte des Modells steht in inversem Verhältnis zu den Metriken TNR und FPR (vgl. Tiwari 2022, S. 57-58).

- ***Precision***:
  Die Precision gibt an, wie viele positive Labels im Verhältnis zu allen positiv prognostizierten Labels richtig prognostiziert wurden.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.18}$$

  Es besteht eine negative Korrelation zwischen Precision und Recall (TPR) (vgl. Tiwari 2022, S. 60-61).

- ***F1-Score***:
  Der F1-Score stellt ein harmonisches Mittel von Precision und Recall dar, welches eine Balance zwischen beiden Metriken bietet.

$$F_1 = \frac{2}{\text{Precision}^{-1} + \text{Recall}^{-1}} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.19}$$

  Der F1-Score gewichtet niedrige Werte deutlich stärker als hohe Werte. Folglich wird ein hoher F1-Score nur bei hohen Werten für Recall und Precision erreicht. (Vgl. Tiwari 2022, S. 61).

Im Rahmen einer Mehrklassenklassifikation erfolgt die Berechnung von Precision, Recall und F1-Score für jede Klasse. Die Ergebnisse werden in einem Klassifikationsbericht (engl. *Classification Report (CR)*) dargestellt. Die nachfolgende Tabelle 3.1 veranschaulicht dies anhand von einem CR für $n = 4$ Klassen und insgesamt 1550 Datenpunkte. Der jeweilige Support gibt die Anzahl der beobachteten Datenpunkte bei der jeweiligen Metrik an. Im Folgenden werden die verschiedenen Verfahren zur Berechnung der Metriken erläutert, nämlich Micro Average, Macro Average und Weighted Average. (Vgl.

Grandini et al. 2020, S. 6).

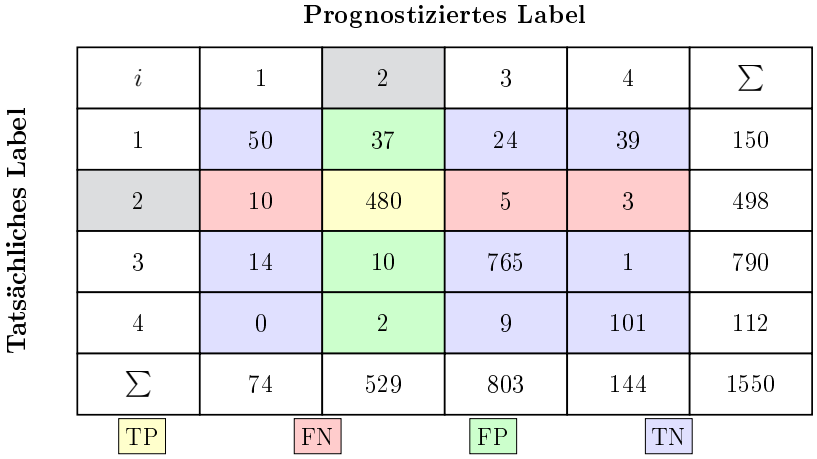


Abbildung 3.3: Mehrklassen-CM mit Klasse 2 als Bezugsklasse, nach Grandini et al. (2020, S. 6).

i	**Precision**	**Recall**	**F1-Score**	**Support**
1	0.68	0.33	0.44	150
2	0.91	0.96	0.94	498
3	0.95	0.97	0.96	790
4	0.70	0.90	0.79	112
Accuracy	0.90			1550
Macro Avg	0.81	0.79	0.78	1550
Weighted Avg	0.89	0.90	0.89	1550

Tabelle 3.1: CR basierend auf der CM in Abbildung 3.3.

- ***Micro Average***:
  Der Micro Average berechnet die Gesamtperformance eines Modells über alle Klassen hinweg, indem die TP, FP und FN aller Klassen summiert werden, bevor Precision, Recall und der F1-Score ermittelt werden. Diese Vorgehensweise gewährleistet eine gleichwertige Gewichtung aller Instanzen, unabhängig von der Klassengröße, was insbesondere bei ungleich großen Klassen von Vorteil ist.

$$\textit{Micro Average Precision} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{Total Column}_i} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\text{Grand Total}} \tag{3.20}$$

$$\textit{Micro Average Recall} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\sum_{i=1}^{n} \text{Total Row}_i} = \frac{\sum_{i=1}^{n} \text{TP}_i}{\text{Grand Total}} \tag{3.21}$$

$$\textit{Micro Average } F_1 = \frac{\sum_{i=1}^{n} \text{TP}_i}{\text{Grand Total}} \tag{3.22}$$

  Da die Werte von Precision, Recall und F1-Score auf den aggregierten TP basieren, sind sie im Micro Average identisch und spiegeln die Accuracy des Modells wider. Dem-

gemäß wird bei dem CR lediglich der Wert der Accuracy, wie in Tabelle 3.1 dargestellt, ausgewiesen. (Vgl. Grandini et al. 2020, S. 7-8).

- ***Macro Average***:
  Der Macro Average berechnet die Precision, den Recall und den F1-Score für jede Klasse i separat und bildet anschließend den Durchschnitt der einzelnen Werte.

$$Macro\ Average\ Precision = \frac{1}{n}\sum_{i=1}^{n}\text{Precision}_i \tag{3.23}$$

$$Macro\ Average\ Recall = \frac{1}{n}\sum_{i=1}^{n}\text{Recall}_i \tag{3.24}$$

$$Macro\ Average\ F_1 = \frac{1}{n}\sum_{i=1}^{n}\text{F}_1 i \tag{3.25}$$

  Der Macro Average gewichtet jede Klasse unabhängig von ihrer Größe gleich, wodurch die Leistung des Modells in allen Klassen gleichermaßen berücksichtigt wird. Dies ist insbesondere von Vorteil, wenn auch kleine Klassen mit wenigen Datenpunkten gleichwertig einbezogen werden sollen. (Vgl. Grandini et al. 2020, S. 6-7).

- ***Weighted Average***:
  Der Weighted Average berücksichtigt sowohl die Performance der einzelnen Klassen als auch den Support in jeder Klasse. In der Folge werden für jede Klasse Precision, Recall sowie F1-Score berechnet. Dabei findet eine Gewichtung nach dem Support pro Klasse.

$$Weighted\ Average\ Precision = \frac{\sum_{i=1}^{n}(\text{Precision}_i \cdot \text{Support}_i)}{\sum_{i=1}^{n}\text{Support}_i} \tag{3.26}$$

$$Weighted\ Average\ Recall = \frac{\sum_{i=1}^{n}(\text{Recall}_i \cdot \text{Support}_i)}{\sum_{i=1}^{n}\text{Support}_i} \tag{3.27}$$

$$Weighted\ Average\ F_1 = \frac{\sum_{i=1}^{n}(\text{F}_1 i \cdot \text{Support}_i)}{\sum_{i=1}^{n}\text{Support}_i} \tag{3.28}$$

  Der Weighted Average berücksichtigt die Klassengröße und ist somit bei der Anwendung des Modells auf Datensätze mit unbalancierten Klassenverteilungen vorteilhaft. Dabei tragen Klassen mit einem höheren Support stärker zum Gesamtwert bei. (Vgl. Tiwari 2022, S. 68 sowie Wang et al. 2023, S. 9).

3.6 Regularisierungs- & Augmentierungstechniken

Bei Aufgaben, bei denen große Datensätze nur schwer zu erlangen sind, ist das Risiko eines Overfittings des Netzwerks an den Trainingsdaten mit der Größe des Netzwerks signifikant erhöht. Zur Verhinderung vom Overfitting werden in der Praxis zwei gängige

Ansätze verfolgt: Regularisierung und Datenaugmentation. (Vgl. Lang 2023, S. 31).
Im Rahmen der Regularisierung wird die Anpassungsfähigkeit des Modells an die Trainingsdaten gezielt reduziert, was eine bessere Generalisierung auf unbekannte Daten zur Folge hat und zu einer verbesserten Leistung auf Validierungsdaten führen kann (vgl. Chollet 2021, S. 146). Eine übliche Methode ist die Gewichtsregularisierung, bei der ein zusätzlicher Term zur Verlustfunktion hinzugefügt wird, der große Werte für die Modellparameter bestraft (vgl. Chollet 2021, S. 148-149).Die L1-Regularisierung resultiert in der Regel in sparsamen Modellen, indem viele Parameter auf Null gesetzt werden, was dazu führt, dass das Modell nur die relevantesten Merkmale lernt (vgl. Géron 2019, S. 137-138). Im Gegensatz dazu reduziert die L2-Regularisierung die Größe aller Parameter, wodurch übermäßig große Gewichtungen vermieden werden und das Modell weniger anfällig für Overfitting wird (vgl. Géron 2019, S. 135). Dies impliziert eine Limitation der Modellwahl und eine Verhinderung der exakten Anpassung an die Trainingsdaten. Es lassen sich zwei Hauptarten unterscheiden:

L1-Regularisierung:

$$Loss \rightarrow Loss + \lambda \sum_j |\theta_i|, \tag{3.29}$$

wobei θ_i die trainierbaren Netzparameter bezeichnet, während λ die Auswirkungen des Regularisierungsterms steuert.

L2-Regularisierung:

$$Loss \rightarrow Loss + \lambda \sum_j \theta_i^2. \tag{3.30}$$

Die Entscheidung zwischen L1 und L2 hängt vom Problem ab. L1 erweist sich insbesondere bei sparsamen Modellen mit wenigen relevanten Parametern als vorteilhaft. Bei gleichmäßig verteilten Gewichten ist eine L2-Regulierung zu bevorzugen. (Vgl. Géron 2019, S. 138-139 sowie Goodfellow et al. 2017, S. 236).
Die Anwendung der Gewichtsregularisierung ist insbesondere bei kleinen DL-Modellen von Nutzen. Bei großen DL-Modellen, die häufig eine hohe Anzahl an Parametern aufweisen, zeigt sich, dass die Einführung von Beschränkungen für Gewichtswerte durch Gewichtsregularisierung einen vergleichsweise geringen Einfluss auf die Modellkapazität hat. In diesen Fällen werden daher andere Regularisierungstechniken wie Dropout eingesetzt. (Vgl. Chollet 2021, S. 150).
Im Rahmen des Dropouts werden Neuronen zufällig ausgewählt, was deren Ausschluss aus dem Optimierungsprozess zur Folge hat. Dies resultiert in einer Reduktion der Speicherkapazität einzelner Neuronen für spezifische Datenpunkte, wodurch ein Overfitting verhindert wird. Der Vorteil von Dropout besteht in der Induktion eines Ensemble-Effekts, da während des Trainings verschiedene Teilmodelle mit zufälligen neuronalen Konfigurationen generiert werden, was die Generalisierungsfähigkeit erhöht. (Vgl. Géron

2019, S. 365-367).
Die Datenaugmentation bezeichnet eine Methode der künstlichen Vergrößerung von Datensätzen, welche durch Modifikationen der Trainingsdaten erfolgt. In der Bildverarbeitung finden Techniken wie zufällige Rotationen, Spiegelungen, Zoom, Zuschneiden, Hinzufügen von Rauschen sowie Anpassungen von Helligkeit, Kontrast und Sättigung Anwendung. Die genannten Transformationen erhöhen die Robustheit des Modells, da dieses lernt, zugrundeliegende Muster in Bildern unabhängig von den genannten Variationen zu erkennen. Dies führt nicht nur zu einer Reduktion des Risikos eines Overfittings, sondern auch zu einer Verbesserung der allgemeinen Leistungsfähigkeit des Modells. (Vgl. Géron 2019, S. 465).
Insbesondere in Szenarien, die durch limitierte und unausgewogene Datensätze gekennzeichnet sind, erweist sich die Datenaugmentation als vorteilhaft. Die Anwendung von Datenaugmentation ermöglicht es Modellen, Merkmale aus unterschiedlichen Perspektiven und unter variierenden Lichtbedingungen besser zu erkennen, was zu einer Erhöhung der Klassifikationsgenauigkeit beiträgt. Des Weiteren unterstützt sie das Erlernen algorithmischer Vorannahmen über Bildinvarianten, was insbesondere bei der Analyse medizinischer Bilddaten wichtig sein könnte. (Vgl. Lang 2023, S. 31-32; 48).
In der medizinischen Bildverarbeitung sind mit der Datenaugmentation jedoch spezifische Herausforderungen verbunden. Transformationsprozesse wie Rotation oder Zuschneiden können die diagnostische Aussagekraft von Daten beeinträchtigen. Dies kann durch eine fehlerhafte Labelzuweisung oder eine Veränderung kritischer Details bedingt sein. Diese Modifikationen führen zu einer Einschränkung der klinischen Anwendbarkeit der augmentierten Daten. Des Weiteren weisen medizinische Bilder häufig eine höhere Variabilität auf als lediglich geometrische und helligkeitsbezogene Unterschiede. Dies begrenzt die Effektivität herkömmlicher Augmentationstechniken. Fortgeschrittene Ansätze wie Generative Adversarial Networks (GANs) eröffnen vielversprechende Möglichkeiten zur Generierung synthetischer Daten. Allerdings ist zu berücksichtigen, dass GANs eine hohe Anzahl an Ausgangsdaten sowie einen umfassenden Modellierungsaufwand erfordern. Zudem zeigt sich in der Praxis, dass die von GANs erzeugte Bildqualität häufig nicht den Anforderungen klinischer Anwendungen genügt. (Vgl. Shorten und Khoshgoftaar 2019, S. 9; 27).

3.7 Konvolutionale neuronale Netze

Konvoultionale neuronale Netze (engl. *Convolutional Neural Network (CNN)*) stellen eine bedeutende Architektur von DL-Modellen dar, die insbesondere für die Verarbeitung und Analyse von Bilddaten zum Einsatz kommt (vgl. Géron 2019, S. 445). Im Gegensatz zu traditionellen MLP ist die Verarbeitung von Daten durch CNN in ihrem natürlichen zweidimensionalen (2D) Format möglich, wodurch die räumlichen Beziehun-

gen zwischen den Pixeln erhalten bleiben (vgl. Géron 2019, S. 448). Im Falle von MLP erfolgt eine Umwandlung der Eingabebilder in einen eindimensionalen (1D) Vektor, was zu einer signifikanten Vergrößerung der Eingabegrößen und einer Vielzahl trainierbarer Parameter führt (vgl. Géron 2019, S. 447-448). Diese Vorgehensweise erweist sich für die Verarbeitung von Bildern jedoch als suboptimal, da MLPs keine Translationsinvarianz aufweisen, was bedeutet, dass Muster nicht erkannt werden können, wenn sie sich an verschiedenen Positionen im Bild befinden (vgl. Chollet 2021, S. 203).

CNNs beheben diese Probleme durch die Anwendung von Konvolutionsoperationen (engl. *convolutional operations*) , welche von der Struktur des visuellen Kortex im Gehirn inspiriert sind. Die Studien von Hubel (1959), Hubel und Wiesel (1959) und Hubel und Wiesel (1968) führten zu der Erkenntnis, dass der visuelle Kortex aus Schichten mit hierarchischer Struktur besteht. Eine Vielzahl von Neuronen im visuellen Kortex weist ein kleines lokales rezeptives Feld (engl. *local receptive field*) auf, welches lediglich auf visuelle Stimuli in einem begrenzten Bereich des Gesichtsfelds reagiert. In den frühen Schichten werden einfache Merkmale wie Kanten erkannt, während die tieferen Schichten diese zu komplexeren Formen und letztlich zu vollständigen Objekten kombinieren. Die genannten Erkenntnisse führten zunächst zur Anwendung von manuell erstellten Filtern zur Merkmalsextraktion aus Bilddaten und schließlich zur Entwicklung von CNNs. (Vgl. Géron 2019, S. 446-447).

Konvolutionale Schichten (engl. *convolutional layers*) stellen die zentralen Bausteine eines CNNs dar. Im Gegensatz zu vollständig verbundenen Schichten weisen Neuronen in den konvolutionalen Schichten lediglich eine geringe Verbindungsstärke zu einem kleinen Bereich der Eingabe auf, dem sogenannten rezeptiven Feld. Infolgedessen können in den initialen Schichten einfache Charakteristika wie Kanten identifiziert und in den nachfolgenden Schichten zu komplexeren Mustern kombiniert werden. Dies trägt maßgeblich zur hohen Effizienz von CNNs bei der Bearbeitung von Bilddaten bei, da die visuelle Welt prinzipiell räumlich hierarchisch ist. Um die Schichtgröße konstant zu halten, wird häufig Zero Padding angewendet, bei dem Nullen an den Eingaberändern hinzugefügt werden. Der Stride gibt an, wie stark sich das rezeptive Feld von einem Schritt zum nächsten verschiebt, und ermöglicht es, die Modellkomplexität zu verringern, indem große Eingaben zu kleineren Ausgabeschichten verbunden werden. (Vgl. Géron 2019, S. 448-449).

Ein Filter, auch Konvolutionskern genannt, besteht aus den Gewichtungen eines Neurons und kann als kleines Bild in der Größe des rezeptiven Feldes dargestellt werden. Ein Beispiel wäre ein 3×3-Filter, bei dem nur die mittlere vertikale Linie mit Einsen gefüllt ist, während der Rest Nullen enthält. Dieser Filter aktiviert nur das zentrale vertikale Merkmal in einem Bild, während andere Bereiche des Bildes ignoriert werden. Ein anderer Filter könnte in ähnlicher Weise nur horizontale Linien erkennen. Wenn alle Neuronen in einer Schicht den gleichen Filter verwenden, erzeugen sie eine Merk-

malskarte (engl. *feature map*), die die aktivierten Bildbereiche hervorhebt. Während des Trainings lernen Konvolutionale Schichten automatisch, welche Filter am besten geeignet sind, um bestimmte Merkmale in einem Bild zu erkennen, und kombinieren diese in den oberen Schichten zu komplexeren Mustern. Dieser Lernprozess ermöglicht es CNNs, immer abstraktere Darstellungen der Bilddaten zu entwickeln. (Vgl. Géron 2019, S. 450).
Pooling-Schichten (engl. *pooling layer*) in CNNs weisen eine ähnliche Funktionsweise wie konvolutionale Schichten auf. Auch hier ist jedes Neuron mit einer begrenzten Anzahl von Neuronen der vorherigen Schicht verbunden, wobei diese in einem kleinen rezeptiven Feld liegen. Im Gegensatz zu konvolutionalen Schichten verfügen Pooling-Neuronen nicht über Gewichte, sondern aggregieren die Eingaben durch eine Aggregationsfunktion wie beispielsweise den Max- oder Mittelwert. In der Praxis findet vornehmlich das Max-Pooling Anwendung, bei dem lediglich der jeweils höchste Wert aus jedem rezeptiven Feld an die nachfolgende Schicht weitergeleitet wird, während die übrigen Werte verworfen werden. (Vgl. Géron 2019, S. 457).
Die Verwendung von Max-Pooling-Schichten führt zu einer Reduktion der Rechenzeit, des Speicherbedarfs sowie der Anzahl der Parameter. Zudem wird eine gewisse Invarianz gegenüber kleinen Verschiebungen eingeführt. Dadurch wird das Modell weniger anfällig für kleine Verschiebungen in den Bilddaten, was die Generalisierungsfähigkeit erhöht. Zudem bietet das Max-Pooling eine begrenzte Rotations- und Skalierungsinvarianz, was insbesondere bei Klassifikationsaufgaben vorteilhaft ist, bei denen die Vorhersagen von diesen Details unabhängig sein sollen. (Vgl. Géron 2019, S. 457-458).
Die Architektur von CNNs umfasst in der Regel mehrere konvolutionale Schichten (+ ReLUs) und Pooling-Schichten, gefolgt von vollständig verbundenen Schichten, welche die extrahierten Merkmale verarbeiten und letztlich zur Klassifikation genutzt werden. Abbildung 3.4 zeigt die Architekur eines CNNs. (Vgl. Géron 2019, S. 460-461).

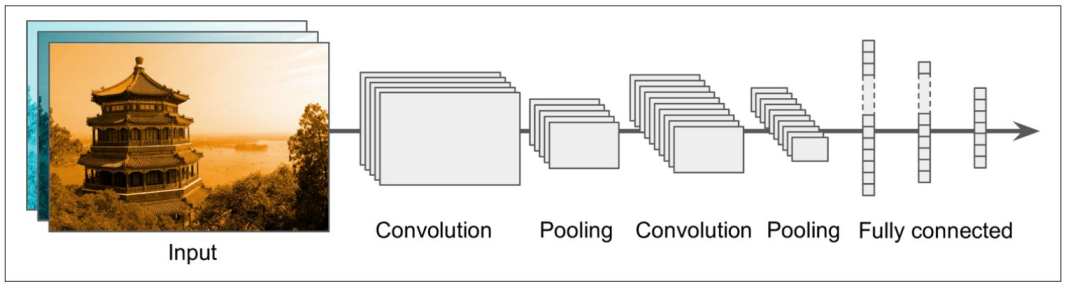


Abbildung 3.4: Architektur eines CNNs, entnommen aus Géron (2019, S. 461).

Zusammenfassend lässt sich festhalten, dass die Verarbeitung dreidimensionaler Daten durch CNNs möglich ist, insbesondere im Kontext medizinischer Bildgebungsverfahren wie MRT- oder CT-Scans. In diesen Fällen erfolgt nicht lediglich eine Stapelung der Merkmalskarten in zwei Dimensionen (Höhe und Breite), sondern ebenfalls in der dritten Dimension, wodurch die räumlichen Informationen eines Volumendatensatzes

berücksichtigt werden. Die gleichzeitige Verwendung mehrerer Merkmalskarten erlaubt die Extraktion verschiedener Aspekte eines Bildvolumens. Dies ermöglicht es CNNs, komplexe 3D-Strukturen zu erkennen und wichtige Muster in dreidimensionalen Daten an unterschiedlichen Positionen mit hoher Effizienz zu verarbeiten. Diese Netze sind speziell darauf ausgelegt, Volumendaten in verschiedenen Schichten hierarchisch zu verarbeiten und zu abstrahieren. Diese Eigenschaft erweist sich insbesondere in der medizinischen Bildgebung als vorteilhaft, da dort eine genaue Erkennung und Segmentierung von Strukturen über mehrere Schichten hinweg erforderlich ist. (Vgl. Géron 2019, S. 451 sowie Lang 2023, S. 35).

3.8 Segmentierungsnetze

DL-Modelle finden neben der Klassifikation auch bei der Bildsegmentierung Anwendung. Es existieren zwei Hauptkategorien der Segmentierung: die semantische Segmentierung (engl. *semantic segmentation*) und die Instanzsegmentierung (engl. *instance segmentation*). Im Rahmen der semantischen Segmentierung erfolgt eine pixelweise Klassifikation von Bildbereichen in vordefinierte Kategorien, wobei eine weitere Differenzierung von Objekten derselben Klasse nicht stattfindet. Im Gegensatz dazu erfolgt bei der Instanzsegmentierung eine separate Segmentierung verschiedener Objekte derselben Klasse. (Vgl. Chollet 2021, S. 231-232). Bezüglich der Segmentierung von MRT-Bildern des Gehirns bedeutet dies, dass bei der semantischen Segmentierung die Aufgabe darin besteht, Tumorgewebe von gesundem Gewebe zu unterscheiden, während bei der Instanzsegmentierung eine Differenzierung zwischen verschiedenen Instanzen desselben Tumors theoretisch möglich wäre.
Eine der Hauptschwierigkeiten bei der Segmentierung mit herkömmlichen CNNs besteht darin, dass die räumliche Auflösung der Bilder im Laufe der Verarbeitung schrittweise verloren geht. Dies hat zur Konsequenz, dass das Modell am Ende lediglich über eine unzureichende Vorstellung hinsichtlich der Lage eines Objekts verfügt, ohne genauere Informationen zu besitzen. Zur Lösung dieser Problematik wurden verschiedene Ansätze entwickelt, darunter Architekturen wie Autoencoder, die speziell für Segmentierungsaufgaben angepasst sind. (Vgl. Géron 2019, S. 492).
Die U-Net-Architektur basiert auf einem Autoencoder-Ansatz, welcher aus einem Encoder und einem Decoder besteht wie in Abbildung 3.5 dargestellt. Dabei wird das Bild durch den Encoder in eine latente Repräsentation transformiert, welche durch den Decoder wiederum in die ursprüngliche Auflösung zurückgeführt wird. (Vgl. Géron 2019, S. 569). Der Einsatz von transponierten konvolutionellen Schichten (engl. *transpose convolutions*) ermöglicht die Wiederherstellung der räumlichen Dimensionalität im Decoder (vgl. Géron 2019, S. 493). Ein wesentliches Charakteristikum des U-Net ist die Verwendung von sogenannten Skip-Verbindungen (engl. *skip connections*), welche eine unmit-

telbare Weitergabe von Informationen aus den initialen Schichten des Encoders an die korrespondierenden Schichten des Decoders ermöglichen (vgl. Géron 2019, S. 494). Dies gewährleistet die Beibehaltung der frühen, globalen Bildmerkmale über das gesamte Netzwerk, was zu einer präzisen Segmentierung führt (vgl. Deprez und Robinson 2023, S. 255).

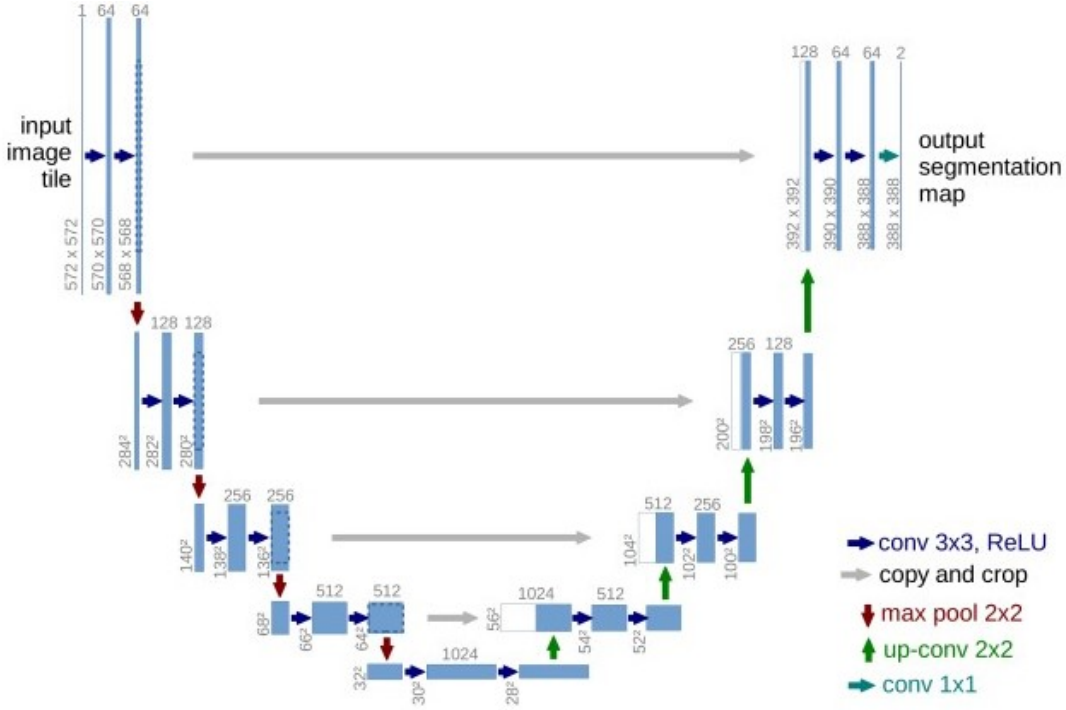


Abbildung 3.5: U-Net-Architektur, entnommen aus Ronneberger et al. (2015, S. 2). Die Bezeichnungen „*copy and crop*“ beziehen sich auf Skip-Verbindungen, während „*up-conv*“ für transponierte konvolutionelle Schichten steht.

Im Rahmen der Segmentierungsaufgabe findet in der Praxis üblicherweise eine Bewertung der Übereinstimmung zwischen den vorhergesagten und den tatsächlichen Segmentierungen unter Verwendung des Dice-Koeffizienten statt. Der Dice-Koeffizient wird gemäß folgender Definition ermittelt:

$$DSC(A, B) = \frac{2|A \cap B|}{|A| + |B|}, \tag{3.31}$$

wobei A und B die Menge der Punkte in den Segmentierungen darstellen. Ein Dice-Wert von 1 impliziert eine perfekte Übereinstimmung, während 0 keine Übereinstimmung anzeigt. (Vgl. Lang 2023, S. 37).
Der zu minimierende Dice-Verlust lässt sich wie folgt berechnen:

$$L_{DSC} = 1 - DSC(A, B). \tag{3.32}$$

Zur Messung der maximalen Abweichung zwischen den Segmentierungen wird neben dem Dice-Koeffizienten auch der Hausdorff-Abstand verwendet. Der Hausdorff-Abstand wird wie folgt definiert:

$$HD(A,B) = \max\left(\sup_{a\in A}\inf_{b\in B} d(a,b), \sup_{b\in B}\inf_{a\in A} d(a,b)\right) \tag{3.33}$$

wobei $d(a,b)$ die euklidische Distanz zwischen den Punkten a und b darstellt und sup sowie inf das Supremum und das Infimum sind. (Vgl. Lang 2023, S. 37).
Der Einsatz solcher Architekturen und Verlustfunktionen ermöglicht es CNNs, zunehmend komplexere Bilddaten zu analysieren und präzise Segmentierungen durchzuführen.

3.9 Transfer Learning

Die zentrale Idee des Transfer Learning (TL) besteht in der Nutzung von Wissen, welches in einem spezifischen Kontext [A] erlernt wurde, zu nutzen, um die Generalisierungsleistung in einem anderen Kontext [B] zu verbessern (vgl. Goodfellow et al. 2017, S. 536). Dies ist besonders relevant, wenn für die Modellierung nicht genügend umfangreiche Datensätze zur Verfügung stehen (vgl. Chollet 2021, S. 218-219). Modelle, die auf umfangreichen Datensätzen trainiert wurden, erlernen grundlegende visuelle Charakteristika, die auf andere Domänen transferierbar sind (vgl. Chollet 2021, S. 218-219). Dies impliziert, dass Merkmale, die in [A] erfasst wurden, auch in [B] nützlich sein können, da zahlreiche visuelle Informationen, wie Kanten und Formen, domänenübergreifend von Bedeutung bleiben (vgl. Chollet 2021, S. 219-220).
Innerhalb der medizinischen Bildverarbeitung, in der häufig lediglich kleine Datensätze zur Verfügung stehen, hat sich das TL als besonders wertvoll erwiesen. Ein Modell, welches beispielsweise auf dem ImageNet-Datensatz mit Millionen von Bildern aus verschiedenen Alltagsszenarien trainiert wurde, kann auch für medizinische Bilddaten, wie beispielsweise MRT-Scans, eingesetzt werden. Obwohl die Domänen – natürliche Bilder und medizinische Bilder – signifikante Differenzen aufweisen, erweisen sich grundlegende visuelle Charakteristika, die auf ImageNet erlernt wurden, auch für die medizinische Bildverarbeitung als nützlich. (Vgl. Lang 2023, S. 39).
Im Rahmen der Anwendung von TL erfolgt in der Regel eine Übertragung der Gewichtungen aus einem vortrainierten Modell auf das neue Modell. Diesbezüglich sind insbesondere die initialen Schichten des Netzwerks von Interesse, da sie für die Erkennung einfacher Merkmale verantwortlich sind. Für spezifische Aufgabenstellungen, wie beispielsweise die Erkennung von Tumoren in medizinischen Bilddaten, werden die letzten Schichten des Netzwerks modifiziert oder ersetzt. Dieser Prozess wird als Fine-Tuning (FT) bezeichnet und beinhaltet eine erneute Trainingsphase, um das Modell optimal an die jeweilige Problemstellung anzupassen. (Vgl. Géron 2019, S. 345-347).

Die Verwendung medizinischer Bilddaten, wie MRT-Scans, erfordert oft die Verarbeitung von 3D-Daten. Dies stellt eine Herausforderung dar, da vortrainierte Modelle in der Regel auf die Analyse von 2D-Daten ausgelegt sind. Um dennoch 3D-Daten nutzen zu können, werden häufig 2D-Schnitte in die Farbkanäle des Modells eingefügt, was jedoch mit einem Verlust an räumlicher Information einhergeht. Eine bessere Lösung besteht in der Entwicklung speziell für 3D-Daten optimierter TL-Modelle. Studien zur medizinischen Bildverarbeitung zeigen unterschiedliche Ergebnisse. Einige konnten keine signifikanten Vorteile durch TL nachweisen, während andere Studien klare Verbesserungen feststellten. Daher sollte die Anwendung von TL sorgfältig im Hinblick auf die konkrete Aufgabe und den verfügbaren Datensatz abgewogen werden. (Vgl. Lang 2023, S. 48).

4 Datensatz

4.1 Beschreibung der Daten

Für diese Arbeit wurde ein offen zugänglicher MRT-Datensatz aus dem Kaggle-Repository verwendet Nickparvar (2021). Der Datensatz umfasst drei öffentlich zugängliche Quellen: Figshare Cheng (2017) , SARTAJ Bhuvaji et al. (2020), und BR35H Hamada (2020). In diesem Zusammenhang werden verschiedene MRT-Bilder des menschlichen Gehirns präsentiert, welche eine Vielfalt an HT und deren charakteristische Eigenschaften repräsentieren.
Der Datensatz enthält insgesamt 7.023 RGB-Bilder mit einer Auflösung von 512 x 512 Pixeln. Diese umfassen vier Klassen:

- Gliome (engl. *glioma*): 1.621 Bilder
- Meningeome (engl. *meningioma*): 1.645 Bilder
- Hypophysentumoren (engl. *pituitary*): 1.757 Bilder
- Nicht tumoröse Gewebe (engl. *no tumor*): 2.000 Bilder

Diese Verteilung gewährleistet eine nahezu ausgewogene Repräsentation der häufigsten Tumorarten, was für das Training von Klassifikationsmodellen von essenzieller Bedeutung ist. Dennoch lässt sich eine leichte Tendenz zugunsten der nicht-tumorösen Bilder beobachten, die bei der Modellentwicklung zu berücksichtigen ist, um Verzerrungen zu minimieren.
Die Größe und Diversität des Datensatzes ermöglichen eine robuste Modellierung und ermöglichen die Erkennung generalisierbarer Muster. Abbildung 4.1 stellt die Klassenverteilung grafisch dar. Die Aufteilung des Datensatzes erfolgte in 80% Trainingsdaten und 20% Testingdaten, wobei letztere nochmals in 30% Validierung und 70% Testing unterteilt wurden. Diese Verteilung gewährleistet eine repräsentative Mischung für das Training und die Evaluierung der Modelle. Eine grafische Darstellung ist in Abbildung 4.2 zu sehen.

4.2 Datenvorverarbeitung

Um eine konsistente Eingabegröße für das Modell zu gewährleisten, wurden die RGB-Bilder von ihrer ursprünglichen Auflösung von 512 x 512 Pixeln auf 224 x 224 Pixel standardisiert. Diese Anpassung der Bildgröße ist erforderlich, um die Kompatibilität der Bildgröße zwischen dem selbst entwickelten Modell und dem vortrainierten Modell ResNet-50 sicherzustellen (vgl. He et al. 2016, S. 4 sowie Géron 2019, S. 346). Da es sich um RGB-Bilder handelt, wurden die drei Farbkanäle für jeden Pixel beibehalten

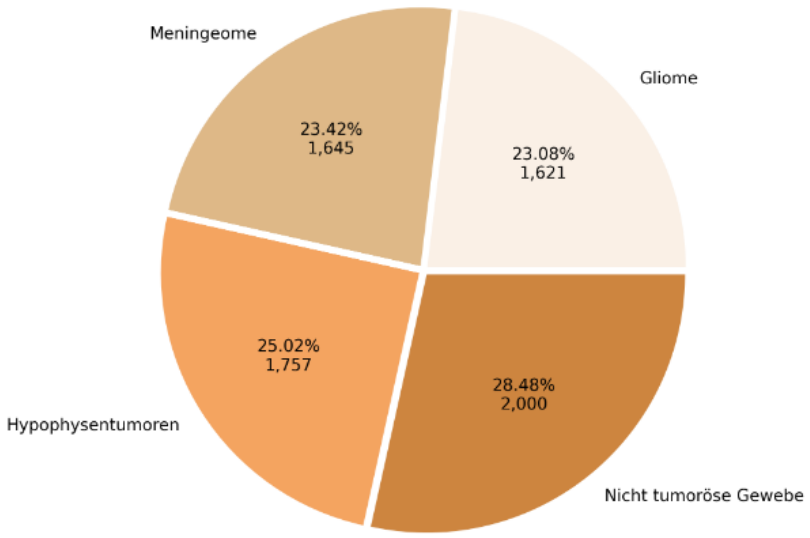


Abbildung 4.1: Verteilung der Bildklassen im Datensatz.

und die Bilddaten unter Verwendung des Parameters `scale=1/255` normalisiert. Dies führte dazu, dass die Pixelwerte für jeden Farbkanal von ihrem ursprünglichen Bereich (0–255) auf einen Bereich von 0–1 skaliert wurden. (Vgl. Goodfellow et al. 2017, S. 453). Des Weiteren wurde durch den Einsatz des Keras-Generators `ImageDataGenerator` eine effiziente Pipeline zur Aufteilung der Daten in Testing- und Validierungsset implementiert, wobei der Parameter `validation_split` verwendet wurde. Dies gewährleistet, dass die Modelle während des Trainings optimal vorbereitet werden (vgl. Géron 2019, S. 482). Die Kombination aus Standardisierung der Auflösung und Normalisierung der Werte ermöglicht es dem Modell, Eingabedaten in einem einheitlichen Format zu verarbeiten, was die Trainingsstabilität und Modellleistung fördert.
Zusätzlich wurde zur Gewährleistung einer reproduzierbaren Durchführung die Steuerung zufälliger Operationen durch das Setzen eines globalen Seeds vorgenommen. Dazu wurden die Seeds in sämtlichen relevanten Modulen, beispielsweise NumPy und TensorFlow, einheitlich definiert. Dies gewährleistet, dass die Datenaufteilung sowie weitere zufällige Operationen deterministisch erfolgt und die Ergebnisse von Experimenten reproduzierbar bleiben.
Da der Datensatz nahezu ausgewogen ist, wurde auf die Anwendung von Datenaugmentation verzichtet, um sicherzustellen, dass keine relevanten Bildmerkmale verloren gehen, die für die Tumorklassifikation von entscheidender Bedeutung sind, und um die klinische Anwendbarkeit der Ergebnisse nicht einzuschränken.
Obwohl die Klassenverteilung im Datensatz nahezu ausgewogen ist, wurde ein Gewichtungsschema implementiert, um die Modellleistung zu optimieren (vgl. Géron 2019, S. 304). Die Funktion `compute_class_weight` wurde eingesetzt, um Klassengewichte zu berechnen, wodurch eine mögliche Verzerrung durch die leicht höhere Anzahl an nicht

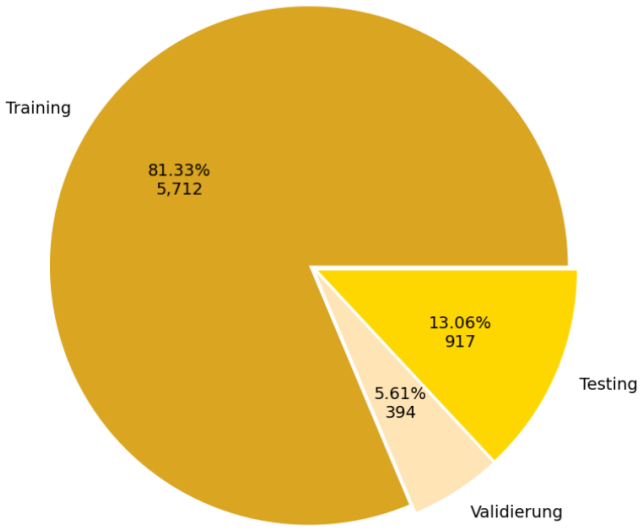


Abbildung 4.2: Verteilung der Daten auf Trainings-, Validierungs- und Testsets.

tumorösen Bildern verhindert werden sollte. Diese Maßnahme resultierte in einer Verbesserung der Modellleistung über alle Klassen hinweg.

4.3 Limitation des Datensatzes

Obschon der Datensatz eine umfassende Auswahl an HT umfasst, bestehen Einschränkungen hinsichtlich der Variabilität im Vergleich zur klinischen Realität. Die in öffentlichen Datensätzen enthaltenen Informationen decken nicht immer die gesamte Bandbreite seltener Tumorarten ab, was die Generalisierbarkeit der Modelle auf klinische Daten aus der Praxis beeinträchtigen könnte. Eine potenzielle Erweiterung könnte in der Anwendung der entwickelten Modelle auf klinische, nicht-öffentliche Datensätze liegen. Dadurch könnten die Robustheit und Übertragbarkeit der Modelle überprüft werden.

5 Implementierung der Modelle

5.1 Betriebssystem & Software-Pakete

Für die Implementierung und das Training der Modelle wurde eine speziell auf die Anforderungen moderner DL-Anwendungen abgestimmte Hardware- und Softwareumgebung verwendet. Im Folgenden wird das verwendete Betriebssystem, die Hardware sowie die eingesetzten Software-Pakete und deren Relevanz für das Projekt beschrieben.

5.1.1 Betriebssystem und Hardware

Die Implementierung erfolgte unter Verwendung des Betriebssystems Windows 11 Pro. Als Hardware stand ein System mit folgenden Spezifikationen zur Verfügung:

- Prozessor (CPU): Intel Core i9
- Arbeitsspeicher (RAM): 32 GB
- Grafikkarte (GPU): NVIDIA RTX 4070 Laptop GPU mit 8 GB Video-RAM

Die Nutzung der NVIDIA GPU ermöglichte eine signifikante Beschleunigung des Trainingsprozesses, da sowohl TensorFlow als auch Keras eine Optimierung für GPU-beschleunigtes Rechnen bieten. Die leistungsstarke CPU, der ausreichende Arbeitsspeicher und die moderne GPU erwiesen sich als essenziell für die effiziente Verarbeitung der umfangreichen Bilddaten und die komplexen Berechnungen der Modelle.

5.1.2 Software-Umgebung

Die Implementierung der Modelle erfolgte in der Programmiersprache Python 3.9.13 unter Verwendung der Entwicklungsumgebung Jupyter Notebook. Jupyter ermöglichte die nahtlose Integration von Code, Visualisierungen und begleitender Dokumentation. Im Rahmen der Implementierung wurden folgende Python-Bibliotheken und Pakete eingesetzt:

- Allgemeine Bibliotheken
  - TensorFlow von Abadi et al. (2015): Framework für DL-Anwendungen, das insbesondere für die Modellarchitektur, das Training und die Optimierung der Modelle genutzt wurde.
  - Keras von Chollet et al. (2015): High-Level-API von TensorFlow, die den Aufbau und die Anpassung neuronaler Netze erleichtert.
  - sklearn.utils (`shuffle`): Zum Durchmischen der Daten, um eine gleichmäßige Verteilung der Klassen im Training zu gewährleisten.

  - NumPy von Harris et al. (2020): Zur effizienten Verarbeitung numerischer Daten.
  - pandas von McKinney (2010): Für die Analyse und Manipulation tabellarischer Daten, insbesondere durch die Verwendung der DataFrame-Struktur.
  - os und random: Für die Verwaltung des Dateisystems und die Reproduzierbarkeit von Experimenten.
  - collections (`Counter`): Zur statistischen Analyse und Darstellung der Häufigkeit von Datenelementen.
  - warnings: Zur Unterdrückung irrelevanter Warnmeldungen.

- Visualisierungsbibliotheken
  - Matplotlib von Hunter (2007) und Seaborn von Waskom (2021): Für die Visualisierung von Datenverteilungen, Modellmetriken und Trainingsergebnissen.
  - sklearn.metrics von Pedregosa et al. (2011): Zur Berechnung von Leistungsmetriken für Modellbewertung.
  - visualkeras: Zur grafischen Darstellung der Modellarchitektur.

- Datenvorbereitung
  - sklearn.utils (`class_weight`): Zum Ausgleich unausgeglichener Klassenverteilungen.
  - ImageDataGenerator: Zur Skalierung der Daten und Unterteilung des Datensatzes in Trainings- und Validierungs-Subsets unter Verwendung eines festen Seeds für Reproduzierbarkeit.

- Modellbildung und Training mit TensorFlow/Keras
  - Modellarchitektur: Aufbau der CNNs mit Schichten wie `Conv2D`, `Dense`, `Dropout`, `MaxPooling2D`, `GlobalAveragePooling2D`, `BatchNormalization` und `Flatten`.
  - Training: Implementierung von Optimierungstechniken wie `Adam` sowie Regularisierung mit `l2` und Verwendung von Verlustsfunktionen wie CCE `categorical_crossentropy`.
  - Keras Tuner (kt): Für das Hyperparameter (HP)-Tuning.
  - Callback (`EarlyStopping`: Zur Optimierung des Trainingsprozesses und zur Vermeidung von Overfitting.

Die gewählte Softwareumgebung sowie die verwendeten Bibliotheken wurden sorgfältig dokumentiert, um die Nachvollziehbarkeit und Reproduzierbarkeit der Ergebnisse zu gewährleisten. Die spezifischen Versionen der wesentlichen Pakete sind in Tabelle 5.1 zusammengefasst.

Bibliothek	Version
TensorFlow	2.10.1
Keras	2.10.0
NumPy	1.21.5
pandas	1.4.4
Matplotlib	3.5.2
Seaborn	0.11.2
scikit-learn	1.0.2

Tabelle 5.1: Softwarebibliotheken und deren spezifische Versionen.

Die Kombination der genannten Software-Pakete ermöglichte eine effiziente und flexible Implementierung der Modelle. Die Verwendung von TensorFlow und Keras als robuste Plattform für die Modellbildung und das Training erwies sich als vorteilhaft, ebenso wie die Integration von Visualisierungsbibliotheken wie Matplotlib und Seaborn, welche eine umfassende Analyse und Präsentation der Ergebnisse erleichterten. Die Hardware- und Softwarekonfiguration bildete somit eine optimale Grundlage für die Implementierung der Modelle und die Verarbeitung des MRT-Datensatzes.

5.2 Selbst entwickeltes CNN-Modell

5.2.1 Modellarchitektur

Das entwickelte CNN wurde mit dem Ziel entworfen, die Klassifikation von MRT-Bildern in vier Klassen zu ermöglichen. Die Architektur des Modells ist in Abbildung 5.1 dargestellt und beinhaltet sowohl konvolutionale Schichten zur Merkmalsextraktion als auch dichte Schichten zur finalen Klassifikation.
Die Architektur des Modells umfasst zunächst eine Eingabeschicht und eine Abfolge von vier konvolutionalen Blöcken, welche jeweils eine Kombination aus einer konvolutionale Schicht, einer Batch-Normalisierung und einer Max-Pooling-Schicht enthalten. Im ersten Block werden 32 Filter mit einer Kernelgröße von 4×4 verwendet, wobei die ReLU-Aktivierungsfunktion für die Einbindung von Nichtlinearität sorgt. In den nachfolgenden Blöcken erfolgt eine sukzessive Erhöhung der Filteranzahl (64, 128 sowie 256), um komplexere Merkmale zu extrahieren und somit die Repräsentationsfähigkeit des Modells zu verbessern.
Die Batch-Normalisierung, welche in jedem Block nach der konvolutionalen Schicht implementiert wird, dient der Stabilisierung der Eingabeverteilung für jede Schicht. Dies resultiert in einer Beschleunigung des Trainings sowie einer Minimierung des Risikos der internen Kovariatenverschiebung (engl. *internal covariate shift*), bei dem sich die Verteilung der Eingabedaten für eine Schicht während des Trainings verändert. (Vgl. Géron 2019, S. 338-341).

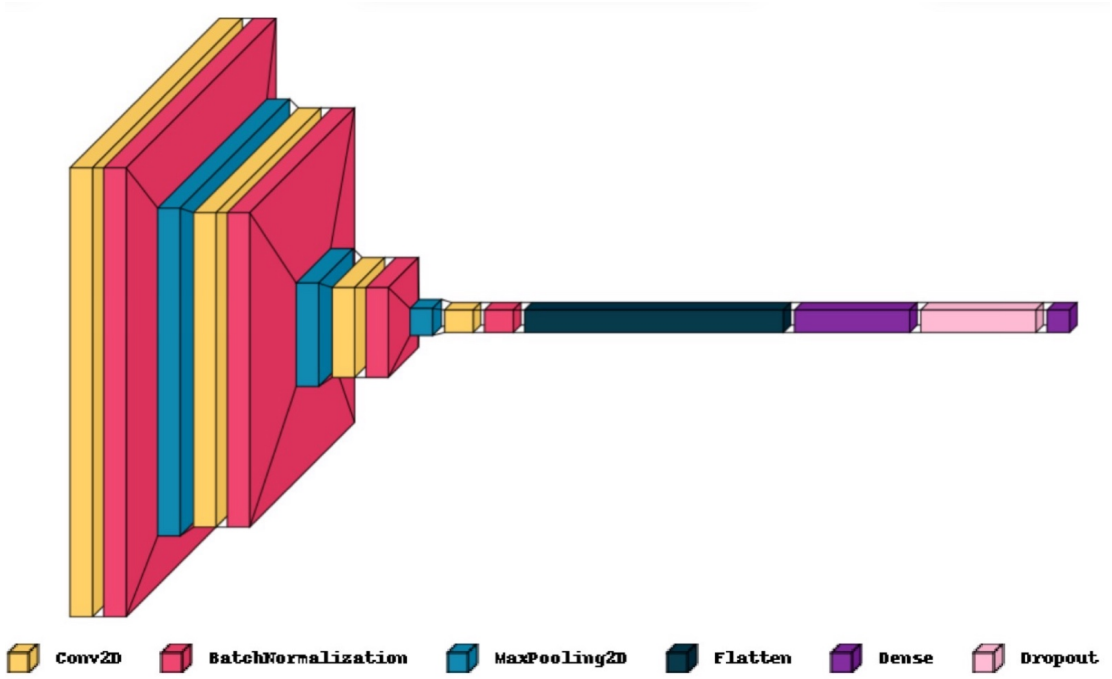


Abbildung 5.1: Architektur vom selbst entwickelten CNN-Modell.

Die Max-Pooling-Schichten mit einer Poolgröße von 3×3 reduzieren die räumliche Dimension der Merkmalskarten und tragen zur Verringerung der Rechenkomplexität sowie zur Erhöhung der translationalen Invarianz bei. In der Folge der Merkmalsextraktion erfolgt durch eine Flatten-Schicht die Überführung in einen eindimensionalen Vektor. Dieser bildet die Grundlage für die nachfolgende Klassifikation.
Im dichten Teil des Netzwerks folgt eine vollverbundene Schicht mit 1024 Neuronen. Im Rahmen dieser Konfiguration findet eine L2-Regularisierung mit $\lambda = 0,001$ Anwendung, um ein Overfitting des Modells zu vermeiden. Des Weiteren wird durch den Einsatz von Dropout-Schicht (50%) die Generalisierungsfähigkeit des Modells optimiert, indem während des Trainings zufällig Neuronen deaktiviert werden. Die Architektur endet mit einer Ausgabeschicht, die vier Neuronen enthält, was der Anzahl der zu unterscheidenden Klassen entspricht, sowie einer Softmax-Aktivierungsfunktion, welche die Wahrscheinlichkeiten der einzelnen Klassen berechnet.
Das Training des Modells erfolgte über 100 Epochen mit einer Batchgröße von 32 unter Berücksichtigung von Klassengewichten. Der Adam-Optimierer mit der Lernrate $\eta = 1e-4$ wurde für das Training eingesetzt, da er sich aufgrund seiner adaptiven Lernratenanpassung als besonders effizient für tiefere neuronale Netzwerke erwiesen hat (vgl. Géron 2019, S. 356-357). Zur Optimierung des Trainingsprozesses wurde das Callback zur frühen Beendigung (`EarlyStopping`) implementiert, sodass das Training abgebrochen wird, wenn der Validierungsverlust über 10 aufeinanderfolgende Epochen keine Verbesserung zeigte (vgl. Géron (2019, S. 315-316)). Die Integration des Callbacks führte zu einer Vermeidung vom Overfitting sowie einer Verbesserung der Modellkonvergenz.

5.3 Vortrainiertes CNN-Modell ResNet-50

5.3.1 Modellarchitektur

ResNet von He et al. (2016), auch als Residual Network bezeichnet, konnte im Jahr 2015 die ILSVRC Russakovsky et al. (2015) für sich entscheiden und wies dabei eine bemerkenswert niedrige Fehlerrate von unter 3,6% auf. ResNet ermöglicht den Aufbau extrem tiefer neuronaler Netzwerke mithilfe von Skip-Verbindungen. Diese Verbindungen bewirken, dass die Eingabe eines Layers unmittelbar dem Ausgang eines späteren Layers zugeführt wird. Infolgedessen modelliert das Netzwerk nicht die Zielfunktion direkt, sondern die Differenz zwischen der Zielfunktion und der Eingabe (Residual Learning). Diese Technik erleichtert den Trainingsprozess und reduziert die Probleme tiefer Netzwerke, wie das Verschwinden von Gradienten (engl. *vanishing gradient*). Auch wenn einzelne Schichten während des Trainings suboptimal lernen, kann das Netzwerk durch die effiziente Signalweiterleitung dennoch Fortschritte erzielen. (Vgl. Géron 2019, S. 471).
Die Architektur von ResNet umfasst sogenannte Residual Units (RUs), die aus mehreren konvolutionalen Schichten mit Skip-Verbindungen bestehen. Jede RU enthält Batch-Normalisierung und ReLU-Aktivierungen, wobei in der Regel 3×3-Kernel verwendet werden, um räumliche Dimensionen der Merkmalskarten beizubehalten. Zur Reduktion der Höhe und Breite der Merkmalskarten kommt eine Stride-2-Konvolution zum Einsatz, während 1×1-Schichten die Dimensionen der Skip-Verbindungen anpassen. (Vgl. Géron 2019, S. 471-473).
Ein wesentlicher Bestandteil von ResNet-50 sind die sogenannten Bottleneck-Residual Units. Diese Units setzen sich aus aus drei konvolutionalen Schichten: einer 1×1-Schicht zur Reduktion der Merkmalskarten, einer 3×3-Schicht zur Mustererkennung und einer weiteren 1×1-Schicht zur Wiederherstellung der ursprünglichen Merkmalskartentiefe. Die 1×1-Schichten, vielfach auch als Bottleneck-Schichten (engl. *bottleneck layer* bezeichnet, fokussieren auf Muster entlang der Tiefe, reduzieren die Dimensionalität und senken dadurch den Rechenaufwand sowie die Anzahl der Modellparameter. Dies verbessert die Effizienz und Generalisierungsfähigkeit des Modells. Die Abfolge aus 1×1-, 3×3- und 1×1-Schicht agiert wie eine besonders leistungsfähige, zusammengefasste konvolutionale Schicht. Im Vergleich zu einem einfachen linearen Klassifikator, der über ein Bild „gescannt“ wird, ermöglicht diese Schichtkombination die Erkennung komplexerer Muster, indem ein kleines neuronales Netzwerk über das Bild verschoben wird. (Vgl. Géron 2019, S. 467-468; 474).
Das ResNet-50 ist in verschiedene Blöcke unterteilt, die jeweils eine steigende Anzahl an Merkmalskarten enthalten. Dabei handelt es sich um drei Residual Units mit 256, vier mit 512, sechs mit 1024 sowie drei mit 2048 Merkmalskarten. Die Implementierung von Bottleneck-Schichten resultiert in einer gesteigerten Effizienz der Verarbeitung, ohne dass dies eine Reduktion der Modellleistung bedingt. (Vgl. He et al. 2016, S. 5).

Abbildung 5.2 illustriert die Architektur von ResNet-50, einschließlich der Bottleneck-Schichten und der Residual Units.

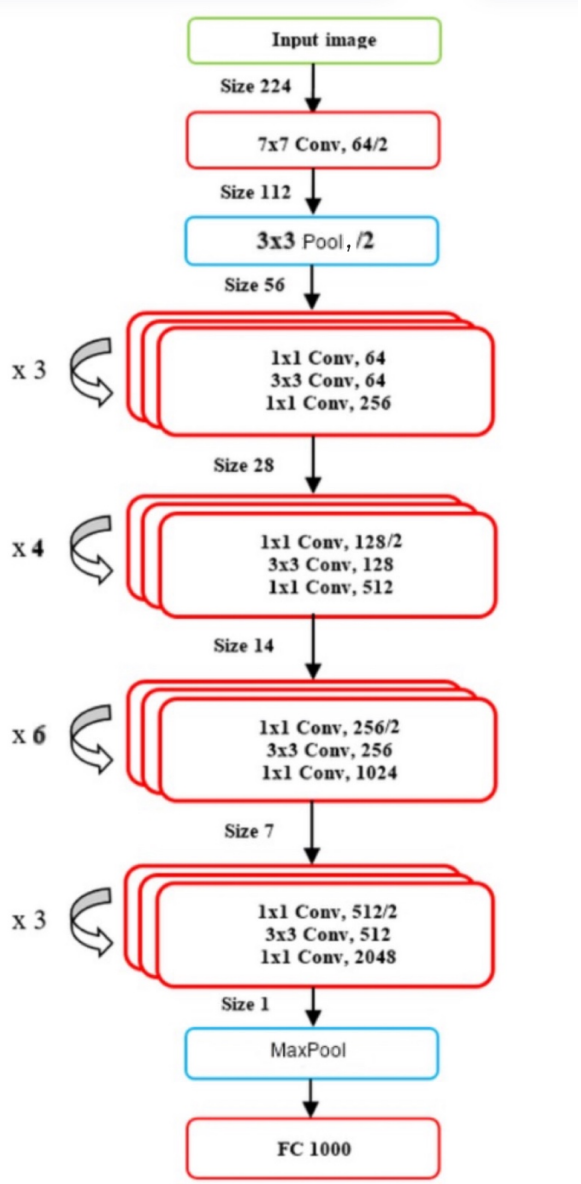


Abbildung 5.2: ResNet-50 Modellarchitektur, entnommen aus Bendjillali et al. (2020, S. 1020).

5.3.2 Praktische Anwendung des Modells

Die Wahl des ResNet-50 im Rahmen dieser Arbeit wurde aufgrund seiner tiefen, aber gleichzeitig effizienten Architektur getroffen, die durch Bottleneck-Schichten und Residual Learning auch bei komplexen Aufgaben hohe Leistungsfähigkeit zeigt. Die Verwendung eines vortrainierten Modells bringt zusätzliche Vorteile, darunter kürzere Trainingszeiten, reduzierte Parameteranzahl und die Möglichkeit, auf ImageNet extrahierte Merkmale für domänenspezifische Aufgaben zu nutzen.

Im ersten Schritt wurde das ResNet-50-Modell dahingehend angepasst, dass ausschließlich die neue Ausgabeschicht trainiert wurde. Dies hatte zur Konsequenz, dass alle anderen Schichten eingefroren blieben und ihre vortrainierten Parameter behielten. Durch diese Methode blieben die vortrainierten Gewichte erhalten, während die Anpassung auf den spezifischen Datensatz fokussiert wurde. Die Implementierung der neuen Ausgabeschicht erfolgte durch eine Kombination verschiedener Schritte. In einem ersten Schritt wurde die Ausgabe vom ResNet-50 in eine Global-Average-Pooling-Schicht überführt, wodurch eine Reduktion der Dimensionen sowie eine robuste Repräsentation der Merkmale erzielt wurde. Im Anschluss wurde eine dichte Schicht mit 1024 Neuronen und ReLU-Aktivierung implementiert, um Overfitting zu reduzieren. Zu diesem Zweck wurde eine Dropout-Schicht (50%) hinzugefügt. Die Ausgabeschicht besteht aus vier Neuronen und einer Softmax-Aktivierung.
Das Modell wurde mit dem Adam-Optimierer konfiguriert. Eine Lernrate η von $1e-4$ wurde gewählt, basierend auf stabilen und schnellen Ergebnissen in vorangegangenen Tests. In Bezug auf die Verlustfunktion wurde die Wahl auf CCE getroffen, da es sich um ein mehrklassiges Klassifizierungsproblem handelt. Die Überwachung der Modellleistung erfolgte anhand der Accuracy als Metrik, sowohl während des Trainings als auch auf den Validierungsdaten.
Das Training wurde mit 100 Epochen durchgeführt, wobei das Callback (`EarlyStopping`) integriert wurde. Des Weiteren wurden Klassengewichte eingeführt, um die Auswirkungen der ungleichen Klassenverteilung zu mildern und das Modell für seltener vertretene Klassen zu sensibilisieren.
Nach Abschluss des ersten Trainingsschritts wurde die Ausgabeschicht des Modells beibehalten und die letzten 20 Schichten zum FT freigegeben. Diese Schichten enthalten die komplexesten Merkmalsextraktionen und wurden durch das FT an die Merkmale des spezifischen Datensatzes angepasst. Um ein Overfitting zu vermeiden, wurde die Lernrate η auf $1e-5$ reduziert. Dadurch ist das Modell in der Lage, feinere Anpassungen vorzunehmen, ohne die vortrainierten Gewichte signifikant zu verändern.

5.3.3 Hyperparameteroptimierung (HPO)

Die Auswahl geeigneter HP ist ein entscheidender Schritt in der Optimierung von DNN, da sie maßgeblich die Lernfähigkeit und Generalisierung beeinflusst. Im Rahmen der HPO wurde der Algorithmus Grid Search (GS) angewendet. GS stellt eine systematische Methode zur Optimierung von HP dar, bei der für jeden HP eine endliche Menge potenzieller Werte definiert wird. Im Anschluss erfolgt die Modellierung aller Kombinationen der zuvor definierten Werte, welche einer Evaluierung unterzogen werden. Das Ziel besteht in der Identifikation derjenigen HP-Kombination, welche den geringsten Validierungsfehler bzw. die beste Validierungsgenauigkeit aufweist. Die Methode basiert

auf der Erstellung eines kartesischen Produkts aus den Werten der HP und umfasst für jede Kombination ein Training. (Vgl. Goodfellow et al. 2017, S. 432).
Der Prozess von GS kann in drei wesentliche Schritte unterteilt werden: Zunächst werden die Wertebereiche definiert. Für numerische HP wie die Lernrate wird häufig eine logarithmische Skalierung verwendet, während diskrete Parameter durch eine begrenzte Anzahl spezifischer Werte definiert werden. Anschließend werden alle möglichen Kombinationen im Rahmen einer systematischen Evaluierung getestet. Dabei steigt die Trainingszeit exponentiell mit der Anzahl der HP nach $O(n^m)$, wobei n die Anzahl der Werte pro HP und m die Anzahl der HP. Zum Beispiel würde die Optimierung von drei HP mit jeweils fünf möglichen Werten zu $5^3 = 125$ Kombinationen führen. (Vgl. Goodfellow et al. 2017, S. 434).
GS eignet sich besonders für Probleme mit wenigen HP, da das Verfahren trotz exponentiell wachsender Rechenzeit eine vollständige Exploration des Parameterraums erlaubt. Aufgrund dieser Einfachheit und Gründlichkeit ist GS trotz seiner Einschränkungen bei komplexen Parameterräumen in der Praxis weit verbreitet. (Vgl. Goodfellow et al. 2017, S. 434).
Im Rahmen dieser Arbeit wurde GS eingesetzt, um die Anzahl der trainierbaren Schichten (`trainable_layers`) im Modell ResNet-50 zu optimieren. Dieser HPs wurde auf Basis früherer Trainingsdurchläufe gewählt, bei denen das Training der letzten 20 Schichten vielversprechende Ergebnisse lieferte. Die Optimierung erfolgte durch Variation der trainierbaren Schichten im Bereich von 1 bis 25, um den Einfluss dieser Schichten auf die Modellleistung systematisch zu untersuchen.
GS erwies sich als geeignet, da lediglich ein einziger HP optimiert werden musste, was den Rechenaufwand trotz exponentieller Skalierung überschaubar hielt. Des Weiteren ermöglichte der Algorithmus im Gegensatz zum Algorithmus Random Search (RS), der die Parameterkombinationen zufällig auswählt, eine vollständige Abdeckung des definierten Bereichs, sodass eine präzise Analyse der Auswirkungen auf die Fähigkeit des Modells, domänenspezifische Merkmale zu erlernen, möglich war (vgl. Goodfellow et al. 2017, S. 433).
Das Modell ResNet-50 wurde in seiner ursprünglichen Architektur mit der neuen Ausgabeschicht geladen. Zu Beginn des Prozesses wurden sämtliche Schichten eingefroren, sodass keine Modifikationen an ihnen vorgenommen wurden. Im Anschluss wurden lediglich die letzten Schichten entsprechend der zuvor definierten Anzahl für FT freigegeben, um die Optimierung auf domänenspezifische Merkmale zu konzentrieren.
Für die Kompilierung des Modells wurde der Adam-Optimierer mit einer Lernrate von $\eta = 1e - 5$ verwendet. Das Training erfolgte auf den Trainings- und Validierungsdatensätzen, wobei die Leistungsfähigkeit des Modells kontinuierlich evaluiert wurde. Zur Durchführung der GS wurde ein Tuner-Objekt erstellt, welches die Modellarchitektur sowie die Optimierungsziele referenzierte. Die Anzahl der Versuche wurde auf maximal

25 konfiguriert, wobei jeder Versuch unterschiedliche Werte für die trainierbaren Schichten testete. Um die Stabilität der Ergebnisse zu gewährleisten, wurde jede Konfiguration dreimal wiederholt. Nach Abschluss der GS wurde die Konfiguration mit dem niedrigsten Validierungsverlust als optimal identifiziert. Die resultierenden HP wurden direkt aus dem Tuner abgerufen, um die optimale Anzahl trainierbarer Schichten zu bestimmen und die Leistung des Modells ResNet-50 weiter zu optimieren.
Neben der manuellen Optimierung und der GS stellt die Bayessche Optimierung (BO) eine weitere effiziente Alternative dar. Dieser Ansatz modelliert die Funktion der Hyperparameterleistung auf Basis einer Wahrscheinlichkeitsverteilung und optimiert iterativ, was insbesondere bei komplexen Modellen mit hohen Trainingskosten von Vorteil ist. Des Weiteren erlaubt die Methode die gezielte Exploration von Bereichen des Hyperparameterraums, wodurch sich eine Beschleunigung der Optimierung erzielen lässt. (Vgl. Bischl et al. 2023, S. 9-12).

6 Performancevergleich der Modelle

6.1 Darstellung der Ergebnisse

Das selbst entwickelte CNN-Modell erreichte auf dem Testdatensatz eine Testgenauigkeit von 98,10% und einen Testverlust von 0,1027. Die in Abbildung 6.1 dargestellten Training- und Validierungslernkurven verdeutlichen eine nahezu kontinuierliche Verbesserung sowohl der Genauigkeit als auch des Verlusts, wobei einzelne starke Schwankungen zu beobachten sind.

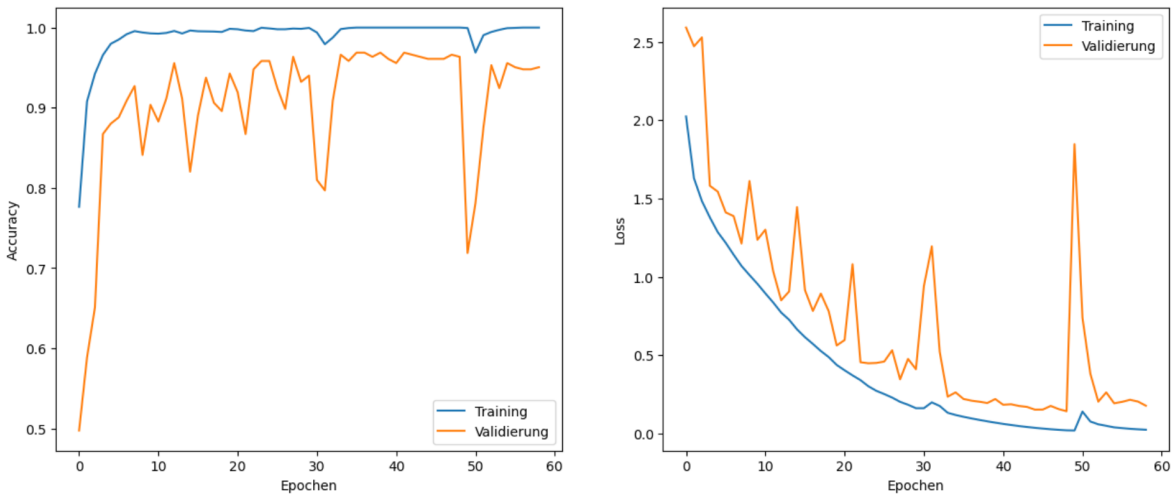


Abbildung 6.1: Lernkurven des selbst entwickelten CNNs: Links die Genauigkeit und rechts der Verlust.

Die in Abbildung 6.2 dargestellte CM zeigt eine hohe Precision sowie einen exzellenten Recall für die Klasse notumor. Demgegenüber weisen die F1-Scores der Klassen glioma und meningioma eine leichte Unterperformance gegenüber den Werten der übrigen Klassen auf, wie aus dem CR in Abbildung 6.3 ersichtlich ist.

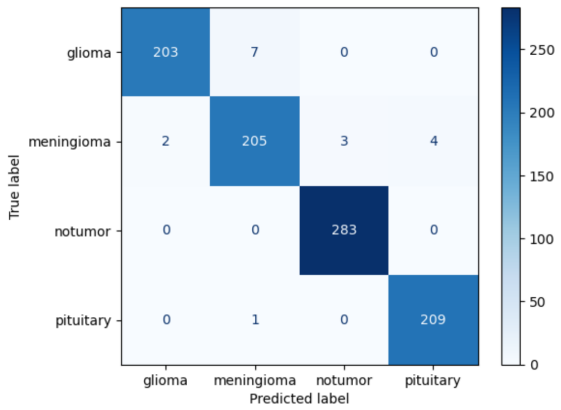


Abbildung 6.2: CM vom selbst entwickelten CNN.

	precision	recall	f1-score	support
glioma	0.99	0.97	0.98	210
meningioma	0.96	0.96	0.96	214
notumor	0.99	1.00	0.99	283
pituitary	0.98	1.00	0.99	210
accuracy			0.98	917
macro avg	0.98	0.98	0.98	917
weighted avg	0.98	0.98	0.98	917

Abbildung 6.3: CR vom selbst entwickelten CNN.

Das Modell ResNet-50, welches lediglich um eine an den Datensatz angepasste Ausgabeschicht erweitert und trainiert wurde, erzielte eine Testgenauigkeit von 76,45% bei einem Verlust von 0,5825. Die Analyse der Lernkurven, dargestellt in Abbildung 6.4, deutet auf eine limitierte Generalisierungsfähigkeit hin.

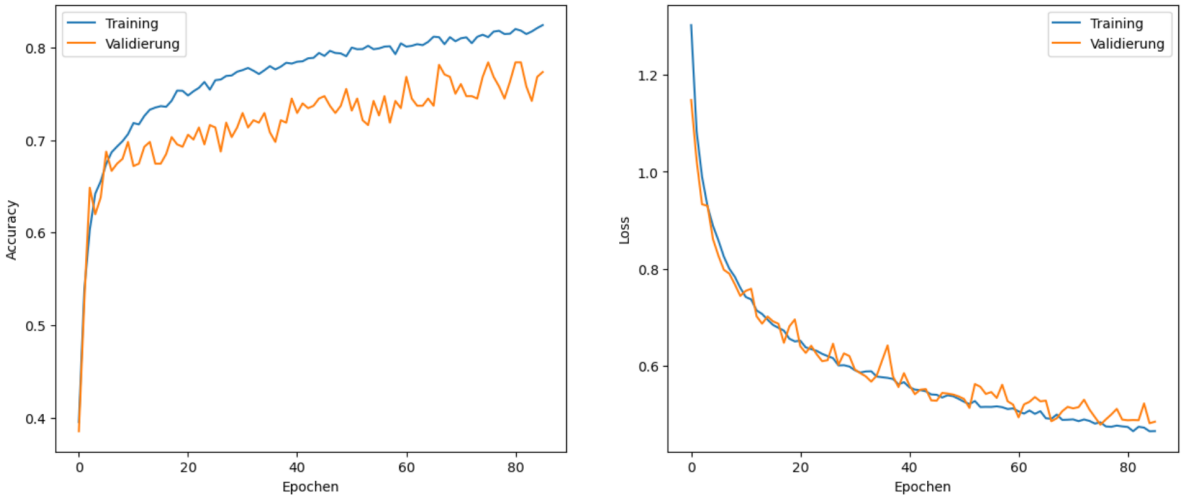


Abbildung 6.4: Lernkurven des ResNet-50 mit angepasster Ausgabeschicht: Links die Genauigkeit, rechts der Verlust.

Wie aus der CM (vgl. Abbildung 6.5) ersichtlich ist, zeigte das Modell Schwierigkeiten bei der Klassifikation der Klassen glioma und meningioma. Demgegenüber konnten die Klassen notumor und pituitary mit einer höheren Anzahl korrekter Klassifikationen zugeordnet werden. Dies wird durch den CR (vgl. Abbildung 6.6) bestätigt, der einen Macro Average F1-Score von 0,75 ausweist. Dies unterstreicht, dass eine einfache Anpassung der Ausgabeschicht ohne ein weitergehendes FT nicht ausreichend ist.

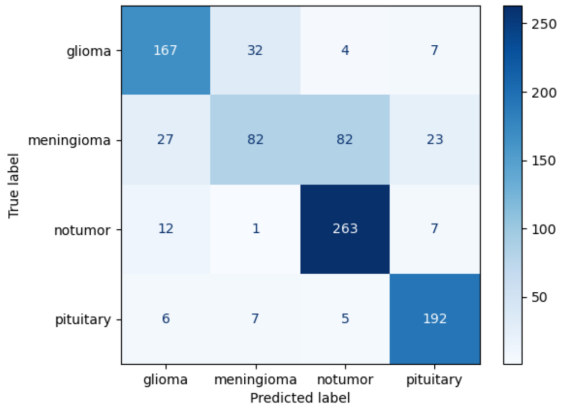


Abbildung 6.5: CM vom ResNet-50 mit angepasster Ausgabeschicht.

	precision	recall	f1-score	support
glioma	0.79	0.80	0.79	210
meningioma	0.67	0.38	0.49	214
notumor	0.74	0.93	0.83	283
pituitary	0.84	0.91	0.87	210
accuracy			0.77	917
macro avg	0.76	0.76	0.75	917
weighted avg	0.76	0.77	0.75	917

Abbildung 6.6: CR vom ResNet-50 mit angepasster Ausgabeschicht.

Das FT der letzten 20 Schichten des ResNet-50 führte zu einer signifikanten Steigerung der Ergebnisse. Die Evaluierung des Testdatensatzes resultierte in einer Genauigkeit

von 97,43% sowie einem Verlust von 0,1051. Die Lernkurven (vgl. Abbildung 6.7) demonstrieren eine signifikante Steigerung der Generalisierungsfähigkeit im Vergleich zum vorherigen Ansatz.

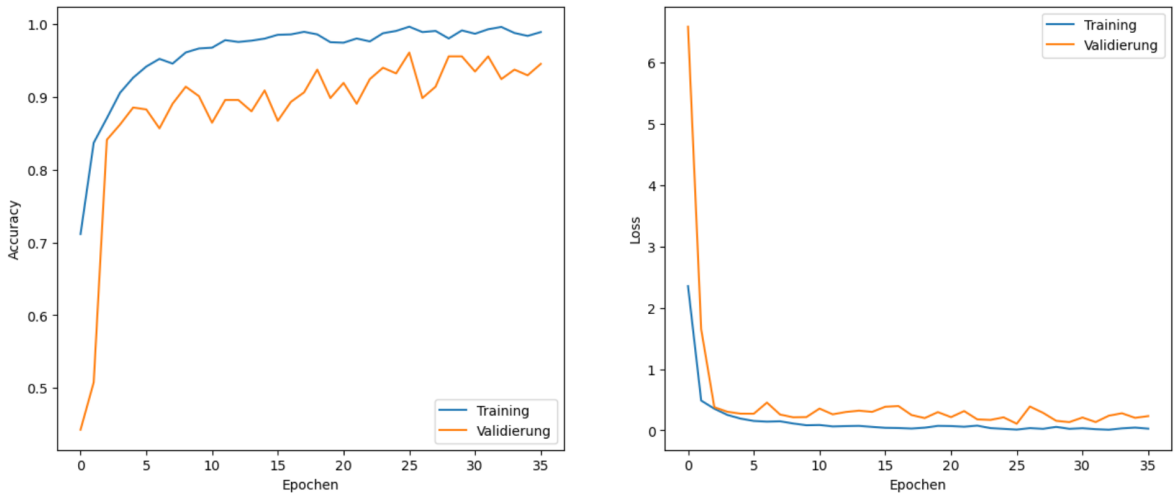
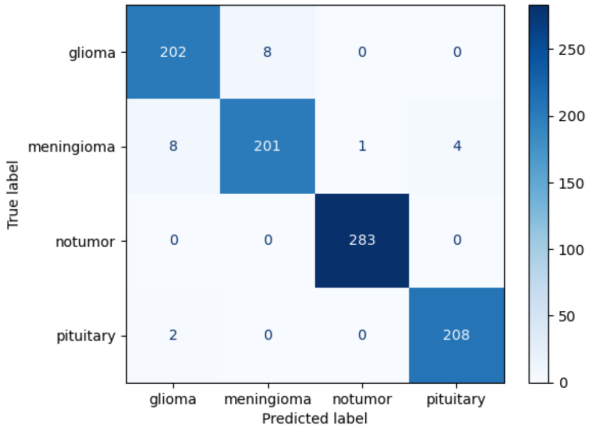


Abbildung 6.7: Lernkurven des ResNet-50 mit FT: Links die Genauigkeit, rechts der Verlust.

Die CM (Abbildung 6.8) zeigt, dass das FT eine präzisere Differenzierung der Klassen glioma und meningioma ermöglicht hat. Ergänzend verdeutlicht der CR (Abbildung 6.9) die gesteigerte Leistungsfähigkeit des Modells, was sich in einem Macro Average F1-Score von 0,97 für alle Klassen widerspiegelt.



	precision	recall	f1-score	support
glioma	0.95	0.96	0.96	210
meningioma	0.96	0.94	0.95	214
notumor	1.00	1.00	1.00	283
pituitary	0.98	0.99	0.99	210
accuracy			0.97	917
macro avg	0.97	0.97	0.97	917
weighted avg	0.97	0.97	0.97	917

Abbildung 6.8: CM vom ResNet-50 mit FT.

Abbildung 6.9: CR vom ResNet-50 mit FT.

Die HPO vom Resnet-50 mit der Anzahl der trainierbaren Schichten als HP demonstrierte, dass sowohl die Genauigkeit als auch der Verlust ab einer Anzahl von 19 Schichten eine Stabilisierung erfahren und somit ähnliche Ergebnisse im Durchschnitt erzielt werden konnten. Der optimale HP resultierte in einer Anzahl von 23 Schichten mit einer Testgenauigkeit von 97,66% bei einem Testverlust von 0,1097. Die Abbildung 6.10 veranschaulicht den Effekt des HPs auf die Genauigkeit und den Verlust. Die CM und der

CR mit dem besten HP sind in den Abbildungen 6.11 und 6.12 dargestellt.
Die Ergebnisse nach HPO (23 Schichten) belegen die Konsistenz der Performance des Modells. Die erzielte Genauigkeit, der Verlust und die F1-Scores weichen nur geringfügig von den Resultaten mit 20 Schichten ab, was auf die Robustheit der Optimierung hindeutet.

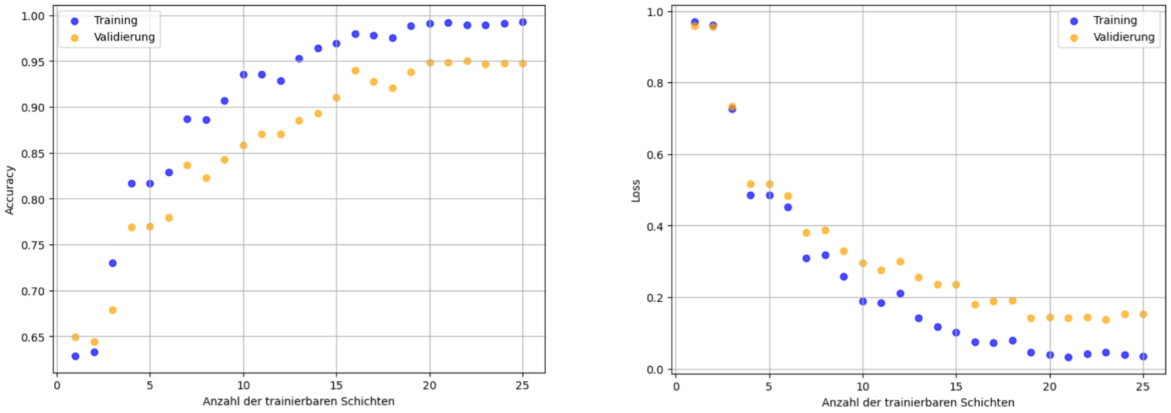


Abbildung 6.10: Durchschnittliche Genauigkeit (links) und durchschnittlicher Verlust (rechts) in Abhängigkeit von der Anzahl der trainierbaren Schichten.

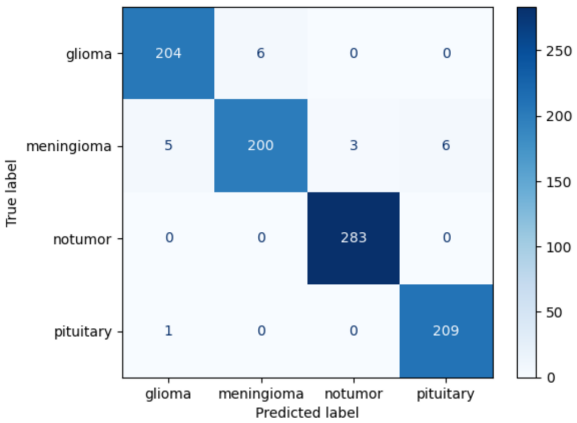


Abbildung 6.11: CM vom ResNet-50 nach HPO.

	precision	recall	f1-score	support
glioma	0.97	0.97	0.97	210
meningioma	0.97	0.93	0.95	214
notumor	0.99	1.00	0.99	283
pituitary	0.97	1.00	0.98	210
accuracy			0.98	917
macro avg	0.98	0.98	0.98	917
weighted avg	0.98	0.98	0.98	917

Abbildung 6.12: CR vom ResNet-50 nach HPO.

6.2 Diskussion der Ergebnisse

Die Analyse der Resultate beider Modelle offenbart eine beeindruckende Klassifikationsleistung von HT. Das selbst entwickelte CNN erreichte eine Genauigkeit von 98,1%, während das feinjustierte ResNet-50-Modell eine Genauigkeit von 97,43% erzielte. Die Ähnlichkeit der Resultate veranschaulicht die jeweiligen Stärken und Schwächen der Ansätze, die für die praktische und theoretische Anwendung von DL in der medizinischen Bildanalyse relevant sind.

Das CNN profitierte von spezifischen Anpassungen an den verwendeten DS und demonstrierte eine hohe Effizienz, insbesondere durch den Einsatz von Techniken wie Dropout, Batchnormalisierung und L2-Regularisierung. Im Rahmen des Modelltrainings wurden diverse Dropout-Raten (0,3, 0,4 und 0,5) evaluiert, wobei sich eine Rate von 0,5 als optimal erwies. Diese Rate gewährleistet das Gleichgewicht zwischen Regularisierung und Modellkapazität. Des Weiteren wurden diverse Konfigurationen der dichten Schicht mit 256, 512 und 1024 Neuronen evaluiert. Hierbei zeigte sich, dass eine Konfiguration mit 1024 Neuronen die beste Leistung hinsichtlich des F1-Scores und der Generalisierungsfähigkeit erzielte. Allerdings wies das Modell Anzeichen von Overfitting auf, insbesondere bei den Klassen glioma und meningioma. Dies lässt auf eine begrenzte Datenvielfalt in diesen Kategorien schließen. Dies verdeutlicht die Notwendigkeit einer besseren Datenrepräsentation, um die Modellleistung zu stabilisieren. Gleichzeitig wies das CNN eine hohe Genauigkeit auf, ohne dass hierfür ein umfassendes TL oder vortrainierte Gewichte erforderlich waren. Dies unterstreicht die Wirksamkeit der gewählten Modellarchitektur und Regularisierungstechniken.

Im Gegensatz dazu wies das ResNet-50-Modell eine höhere Robustheit gegenüber Overfitting auf, was auf die Lernkurven mit wenigen starken Schwankungen zu sehen ist. Dies lässt sich auf die Vorteile von TL und die bereits in den vortrainierten Schichten enthaltenen generalisierten Fähigkeiten zur Merkmalsextraktion zurückführen. Die Resultate demonstrieren jedoch auch, dass eine bloße Anpassung der Ausgabeschicht ohne weiterführendes FT nicht hinreichte, um optimale Resultate zu erzielen. Erst nach einem gezielten FT von Schichten konnte eine vergleichbare Performance wie beim selbst entwickelten CNN erreicht werden. Dies lässt den Schluss zu, dass die Anzahl der trainierbaren Schichten eine zentrale Rolle beim FT solcher Modelle spielt.

Der mit dem ResNet-50-Modell erzielte Genauigkeitswert von 97,66% entspricht in etwa den in der Literatur berichteten Ergebnissen. Ismael et al. (2020) nutzten ResNet-50, ergänzt durch Datenaugmentation, und erzielten einen Genauigkeitswert von 97%. Tandel et al. (2021) verwendeten einen mehrheitswahlbasierten Ensemble-Ansatz und kamen auf ähnlich hohe Werte von 96,54%. Im Gegensatz dazu erzielte das in dieser Arbeit präsentierte ResNet-50-Modell eine vergleichbare Leistung, ohne auf Datenaugmentation zurückzugreifen.

Beide Modelle wiesen eine nahezu perfekte Klassifikation der Klassen notumor und pituitary auf (F1-Scores zw. 0,99 und 1), was darauf hindeutet, dass die morphologischen Merkmale dieser Klassen im Datensatz gut repräsentiert sind und somit von den Modellen effektiv erlernt werden konnten. Diese Ergebnisse unterstreichen die Bedeutung der Datenrepräsentativität bei der Modellentwicklung und -evaluierung.

Die Gegenüberstellung beider Ansätze verdeutlicht zudem den Zielkonflikt zwischen Modellspezialisierung und Generalisierbarkeit. Das selbst entwickelte CNN erzielte eine hohe Leistung auf dem gegebenen Datensatz, wobei spezifische Anpassungen vorgenommen

wurden. Das ResNet-50-Modell hingegen zeichnet sich durch seine vortrainierten Schichten aus, wodurch eine größere Flexibilität und Robustheit gewährleistet wird. Die Resultate liefern wertvolle Erkenntnisse hinsichtlich der praktischen Anforderungen und der Einsatzmöglichkeiten von DL-Modellen in der medizinischen Diagnostik. In einem klinischen Kontext könnte die Implementierung solcher Modelle die diagnostische Effizienz steigern, indem sie Radiologen bei der Differenzierung von Tumortypen unterstützen und die Arbeitsbelastung verringern. Dennoch ist eine Validierung mit unabhängigen klinischen Datensätzen erforderlich, um die Robustheit und Generalisierbarkeit der Modelle zu gewährleisten.

Die Ergebnisse zeigen, dass die Wahl zwischen einem speziell entwickelten Modell und einem vortrainierten Ansatz von den spezifischen Anforderungen sowie den verfügbaren Ressourcen abhängig ist. Beide Ansätze besitzen ihre Relevanz und können durch gezielte Optimierungen eine weitere Verbesserung erfahren. Dies dürfte künftige Forschungsarbeiten zu Modellarchitekturen und Datenverarbeitung anregen.

7 Zukunftsaussichten

Die Weiterentwicklung der Klassifizierung von HT mithilfe von DL-Algorithmen eröffnet zahlreiche Perspektiven zur Leistungssteigerung und Bewältigung bestehender Herausforderungen. Ein wesentliches Problem stellt das Overfitting dar, welches durch eine Erweiterung des Datensatzes reduziert werden könnte. Eine Lösung könnte in der Kombination verschiedener öffentlich zugänglicher MRT-Datensätze oder der Ergänzung um CT-Bilder bestehen. Die üblicherweise begrenzte Größe medizinischer Bilddatensätze sowie deren hohe Spezialisierung könnten durch einen umfassenderen Datensatz kompensiert werden. Dies würde die Robustheit und Generalisierungsfähigkeit der Modelle fördern und zu stabileren Trainingsergebnissen sowie einer besseren Modellvalidität und -anpassungsfähigkeit gegenüber neuen, unbekannten Bildmustern führen.
Ein weiterer wesentlicher Ansatzpunkt zur Verbesserung der Modellleistung stellt die Optimierung der HP dar. Ein fortschrittliches Verfahren in diesem Bereich ist das Iterated Racing von Birattari et al. (2010), welches von Bischl et al. (2023) als eine der führenden Methoden der HPO hervorgehoben wird. Das Verfahren fokussiert die Rechenleistung auf vielversprechende Konfigurationen, indem weniger geeignete Parameterkombinationen bereits in frühen Phasen eliminiert werden. Dies erlaubt eine effizientere und präzisere Suche nach optimalen Parametereinstellungen bei gleichzeitiger Schonung von Rechenressourcen. Diese Methode kann dazu beitragen, künftige Optimierungsprozesse zu beschleunigen und die Genauigkeit der Klassifikationsmodelle weiter zu steigern.
Eine weiterer innovativer Ansatz zur Steigerung der Leistungsfähigkeit stellt der Einsatz vortrainierter Modelle dar, welche auf verschiedene Tumorarten spezialisiert sind. Eine derartige Vortrainierung ermöglicht es dem Modell, allgemeine Tumorcharakteristika besser zu erkennen, wodurch sich zugleich seine Fähigkeit erhöht, spezifische HT-Merkmale zu klassifizieren. Auch die Implementierung eines Segmentierungsansatzes birgt ein vielversprechendes Potenzial. Hierbei könnte das Modell gezielt lernen, Tumorstrukturen in Bildern zu identifizieren und anschließend zu klassifizieren. Der zusätzliche Vorverarbeitungsschritt könnte zu differenzierteren Einblicken in die Tumorbiologie führen, was insbesondere für die präzisere Erkennung seltener oder atypischer HT von Bedeutung wäre.
Im Rahmen künftiger Forschungsarbeiten wäre es vorteilhaft, die Eigenschaften unterschiedlicher Modellarchitekturen systematisch zu analysieren und deren Leistungsfähigkeit im Hinblick auf medizinische Bilddaten zu evaluieren. Eine detaillierte Bewertung der Merkmalsextraktionsfähigkeiten könnte dazu beitragen, die am besten geeignete Architektur für die Klassifikation von HT zu identifizieren und somit die Leistungsfähigkeit weiter zu steigern. Derartige Vergleiche könnten zudem Aufschluss über spezifische Architekturanpassungen geben, die sich besonders für medizinische Bildanalysen eignen.
Ein vielversprechender Forschungsbereich ist zudem die Erweiterung der Modelle auf 3D-

MRT-Daten, um volumetrische Informationen besser auszunutzen. Dies könnte durch die Integration von 3D-CNNs oder hybridisierten Modellen erfolgen, welche sowohl 2D- als auch 3D-Daten kombinieren. Darüber hinaus besteht ein Bedarf an der Entwicklung von Modellen, welche interpretierbare und klinisch relevante Ergebnisse liefern, um die Akzeptanz durch Mediziner zu fördern.
Die Umsetzung der dargestellten Optimierungsansätze konnte im Rahmen dieser Arbeit jedoch nicht realisiert werden, da sie einen signifikanten Aufwand an Zeit und Rechenkapazität erfordert. Umfassende Datenerweiterungen, fortgeschrittene Transfer- oder Segmentierungsansätze sowie eine Hyperparameteroptimierung mittels Iterated Racing würden eine erhebliche Steigerung der verfügbaren Rechenleistung sowie einen erweiterten Zeitrahmen verlangen. Zukünftige Forschungsprojekte könnten daher gezielt auf diesen Ansätzen aufbauen und damit das Potenzial moderner DL-Methoden zur HT-Klassifikation weiter ausschöpfen.

8 Schlussfolgerung

Im Rahmen dieser Arbeit wurde die Klassifikation von HT unter Zuhilfenahme von DL-Algorithmen untersucht. Zu diesem Zweck wurde ein eigener CNN-Ansatz mit 7.023 medizinischen Bildern trainiert. Die Herausforderung, ohne Datenaugmentation zu arbeiten, wurde durch die realistische Nähe zu klinischen Daten motiviert, was insbesondere im Hinblick auf das Risiko des Overfitting von Bedeutung war. Durch den Einsatz von Regularisierungstechniken wie Dropout, Batch-Normalisierung und L2-Regularisierung konnte eine Reduktion des Overfittings erzielt werden. Obwohl der Datensatz lediglich 7.023 Bilder umfasst, konnte das Modell ein Testgenauigkeit von 98,1% sowie einen Testverlust von 0,1027 erzielen, was als gut zu bewerten ist.
Zudem wurde ein Vergleich des eigenentwickelten Modells mit dem vortrainierten ResNet-50-Modell vorgenommen, um die Vorzüge von TL und von FT zu nutzen. Die ursprüngliche Architektur des ResNet-50 wurde beibehalten, wobei die Ausgabeschicht angepasst wurde. Das Modell wurde mit einer definierten Anzahl an trainierbaren Schichten an dem Datensatz feingetunt. Die Durchführung von HPO ermöglichte die Untersuchung des Einflusses der Anzahl trainierbarer Schichten auf die Modellleistung. Die Ergebnisse zeigten, dass die Konsistenz der Resultate ab einer Anzahl von 19 Schichten gegeben war. Das optimierte vortrainierte Modell erreichte eine Testgenauigkeit von 97,66% bei einem Testverlust von 0,1097.
Die ursprüngliche Hypothese, dass das vortrainierte Modell zu besseren Resultaten führen würde, konnte in dieser Untersuchung nicht bestätigt werden. Das vortrainierte Modell erzielte ähnliche Resultate zum selbst entwickelten Modell, wies jedoch kein Overfitting auf, wodurch es sich unter bestimmten Voraussetzungen für die Klassifikation von HT als potenziell besser geeignet erwies. Für Experten im Bereich des MLs und des DLs könnte das selbst entwickelte Modell aufgrund der spezifischen Anpassung an den Datensatz sowie der Möglichkeit einer erweiterten HPO von besonderem Vorteil sein. In Bezug auf die Anwendbarkeit in klinischen Umgebungen, insbesondere zur Unterstützung von Radiologen, stellt das vortrainierte Modell ResNet-50 aufgrund seiner einfachen Implementierung und der guten Ergebnisse eine attraktive Wahl dar.
In künftigen Forschungsarbeiten sollte untersucht werden, inwiefern eine Erweiterung des Datensatzes durch synthetische Daten oder zusätzliche klinische Datensätze eine weitere Verbesserung der Performance beider Modelle ermöglicht. Des Weiteren erscheint eine Vertiefung der HPO sowie eine Prüfung des Einflusses zusätzlicher Regularisierungstechniken sinnvoll, um das Modell weiter zu optimieren und eine Robustheit gegen Overfitting zu erzielen.
Zusammenfassend lässt sich sagen, dass DL ein vielversprechendes Werkzeug zur Unterstützung der medizinischen Diagnostik darstellt, dessen Potenzial durch kontinuierliche Forschung und Optimierung noch weiter ausgeschöpft werden kann.

Literatur

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C. et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Verfügbar: https://www.tensorflow.org/ (Zugriff am 19.11.2024).

Altahhan, F.E., Khouqeer, G.A., Saadi, S., Elgarayhi, A. und Sallah, S.: Refined automatic brain tumor classification using hybrid convolutional neural networks for mri scans. In: *Diagnostics*, Band 13(5):S. 864, 2023. doi:10.3390/diagnostics13050864.

Anaraki, A.K., Ayati, M. und Kazemi, F.: Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. In: *biocybernetics and biomedical engineering*, Band 39(1):S. 63–74, 2019.

Bakas, S., Akbari, H., Sotiras, A., Bilello, M., Rozycki, M., Kirby, J.S. et al.: Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features. In: *Scientific data*, Band 4(1):S. 1–13, 2017.

Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A. et al.: Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge. In: *arXiv preprint arXiv:1811.02629*, 2018.

Bendjillali, R.I., Beladgham, M., Merit, K. und Taleb-Ahmed, A.: Illumination-robust face recognition based on deep convolutional neural networks architectures. In: *Indonesian Journal of Electrical Engineering and Computer Science*, Band 18(2):S. 1015–1027, 2020.

Bhuvaji, S., Kadam, A., Bhumkar, P., Dedge, S. und Kanchan, S.: Brain Tumor Classification (MRI). Kaggle, 2020. doi:10.34740/kaggle/dsv/1183165. Verfügbar: https://www.kaggle.com/dsv/1183165 (Zugriff am 01.10.2024).

Bilsky, M.H.: Einige spezielle Hirntumoren. MSD Manual, 2023. Verfügbar: https://www.msdmanuals.com/de/heim/stÃűrungen-der-hirn-rÃijckenmarks-und-nervenfunktion/tumoren-des-nervensystems/einige-spezielle-hirntumoren (Zugriff am 01.10.2024).

Birattari, M., Yuan, Z., Balaprakash, P. und Stützle, T.: F-Race and Iterated F-Race: An Overview. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer Berlin Heidelberg. S. 311–336. 2010. ISBN 9783642025389.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S. et al.: Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Band 13(2):S. e1484, 2023.

BRATS: BRATS 2013: Brain Tumor Image Segmentation Challenge. SICAS Medical Image Repository, 2013. Verfügbar: https://www.smir.ch/BRATS/Start2013 (Zugriff am 05.10.2024).

BRATS: BRATS 2014: Brain Tumor Image Segmentation Challenge. SICAS Medical Image Repository, 2014. Verfügbar: https://www.smir.ch/BRATS/Start2014 (Zugriff am 05.10.2024).

BRATS: BRATS 2015: Brain Tumor Image Segmentation Challenge. SICAS Medical Image Repository, 2015. Verfügbar: https://www.smir.ch/BRATS/Start2015 (Zugriff am 05.10.2024).

Cauchy, A.: Méthode générale pour la résolution des systemes d'équations simultanées. In: *Comptes rendus de l'Académie des sciences de Paris*, Band 25:S. 536–538, 1847.

Cheng, J.: brain tumor dataset. figshare, 2017. doi:https://doi.org/10.6084/m9.figshare.1512427.v5. Verfügbar: https://doi.org/10.6084/m9.figshare.1512427.v5 (Zugriff am 01.10.2024).

Cheng, J., Huang, W., Cao, S., Yang, R., Yang, W., Yun, Z. et al.: Enhanced performance of brain tumor classification via tumor region augmentation and partition. In: *PloS one*, Band 10(10):S. e0140381, 2015.

Chollet, F.: *Deep learning with Python*. Simon and Schuster, 2021.

Chollet, F. et al.: Keras: The python deep learning library. 2015. Verfügbar: https://keras.io (Zugriff am 19.11.2024).

Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P. et al.: The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository. In: *Journal of digital imaging*, Band 26:S. 1045–1057, 2013.

Deepak, S. und Ameer, P.M.: Brain tumor classification using deep CNN features via transfer learning. In: *Computers in biology and medicine*, Band 111:S. 103345, 2019.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. und Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: *2009 IEEE conference on computer vision and pattern recognition*. 2009, S. 248–255.

Deprez, M. und Robinson, E.C.: *Machine Learning for Biomedical Applications: With Scikit-Learn and PyTorch*. Academic Press, 2023.

Dutta, T.K. und Nayak, D.R.: CDANet: Channel split dual attention based CNN for brain tumor classification in MR images. In: *2022 IEEE international conference on image processing (ICIP)*. IEEE, 2022, S. 4208–4212. doi:10.1109/ICIP46576.2022.9897799.

Ferlay, J., Ervik, M., Lam, F., Laversanne, M., Colombet, M., Mery, L. et al.: Global Cancer Observatory: Cancer Today. Lyon, France: International Agency for Research on Cancer, 2024. Verfügbar: https://gco.iarc.who.int/today (Zugriff am 01.10.2024).

GabAllah, A.M., Sarhan, A.M. und Elshennawy, N.M.: Classification of brain MRI tumor images based on deep learning PGGAN augmentation. In: *Diagnostics*, Band 11(12):S. 2343, 2021. doi:10.3390/diagnostics11122343.

Géron, A.: *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2019.

Goodfellow, I., Bengio, Y. und Courville, A.: *Deep learning*, Band 1. MIT press Cambridge, MA, USA, 2017.

Grandini, M., Bagli, E. und Visani, G.: Metrics for multi-class classification: an overview. In: *arXiv preprint arXiv:2008.05756*, 2020.

Hamada, A.: Br35H :: Brain Tumor Detection 2020. Kaggle, 2020. Verfügbar: https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection/data (Zugriff am 01.10.2024).

Harris, C.R., Millman, K.J., der Walt, S.J. Van, Gommers, R., Virtanen, P., Cournapeau, D. et al.: Array programming with NumPy. In: *Nature*, Band 585(7825):S. 357–362, 2020. doi:10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

He, K., Zhang, X., Ren, S. und Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 770–778.

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

Hubel, D. H. und Wiesel, T.N.: Receptive fields and functional architecture of monkey striate cortex. In: *The Journal of physiology*, Band 195(1):S. 215–243, 1968.

Hubel, D.H.: Single unit activity in striate cortex of unrestrained cats. In: *The Journal of physiology*, Band 147(2):S. 226, 1959.

Hunter, J.D.: Matplotlib: A 2D graphics environment. In: *Computing in Science & Engineering*, Band 9(3):S. 90–95, 2007.

Ismael, S.A.A., Mohammed, A. und Hefny, H.: An enhanced deep learning approach for brain cancer MRI images classification using residual networks. In: *Artificial intelligence in medicine*, Band 102:S. 101779, 2020.

IXI: IXI Dataset. brain-development.org, 2015. Verfügbar: http://brain-development.org/ixi-dataset (Zugriff am 17.11.2024).

Kabir, A., Ayati, M. und Kazemi, F.: ScienceDirect Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. In: *Integr. Med. Res*, Band 39(1):S. 63–74, 2018.

Kebir, S.T. und Mekaoui, S.: An efficient methodology of brain abnormalities detection using cnn deep learning network. In: *2018 International Conference on Applied Smart Systems (ICASS)*. IEEE, 2018, S. 1–5.

Krizhevsky, A., Sutskever, I. und Hinton, G.E: Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, Band 25, 2012.

Lang, D.M.: *Deep learning based analysis of medical imaging data in oncology with a focus on radiation therapy*. Dissertation, Technische Universität München, 2023. Verfügbar: https://mediatum.ub.tum.de/1690422 (Zugriff am 05.10.2024, Erstellung am 09.11.2022).

Lundervold, A.S. und Lundervold, A.: An overview of deep learning in medical imaging focusing on MRI. In: *Zeitschrift für Medizinische Physik*, Band 29(2):S. 102–127, 2019. ISSN 0939-3889. doi:https://doi.org/10.1016/j.zemedi.2018.11.002. URL https://www.sciencedirect.com/science/article/pii/S0939388918301181.

McKinney, W.: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt und Jarrod Millman (Hg.) *Proceedings of the 9th Python in Science Conference*. 2010, S. 56–61. doi:10.25080/Majora-92bf1922-00a.

Mehrotra, R., Ansari, M.A., Agrawal, R. und Anand, R.S: A transfer learning approach for AI-based classification of brain tumors. In: *Machine Learning with Applications*, Band 2:S. 100003, 2020.

Naser, M.A. und Deen, M.J.: Brain tumor segmentation and grading of lower-grade glioma using deep learning in mri images. In: *Computers in biology and medicine*, Band 121:S. 103758, 2020.

Nickparvar, M.: Brain Tumor MRI Dataset. Kaggle, 2021. doi:10.34740/kaggle/dsv/2645886. Verfügbar: https://www.kaggle.com/dsv/2645886 (Zugriff am 01.10.2024).

Pedano, N., Flanders, A.E., Scarpace, L., Mikkelsen, T., Eschbacher, J.M., Hermes, B. et al.: Radiology data from the cancer genome atlas low grade glioma [TCGA-LGG] collection. In: *The Cancer Imaging Archive*, Band 2, 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al.: Scikit-learn: Machine learning in Python. In: *Journal of Machine Learning Research*, Band 12:S. 2825–2830, 2011.

Radiopaedia: Radiopaedia Dataset. 2024. Verfügbar: http://radiopaedia.org (Zugriff am 05.10.2024).

Ravinder, M., Saluja, G., Allabun, S., Alqahtani, M.S., Abbas, M., Othman, M. et al.: Enhanced brain tumor classification using graph convolutional neural network architecture. In: *Scientific reports*, Band 13(1):S. 14938, 2023. ISSN 2045-2322. doi:10.1038/s41598-023-41407-8. URL https://doi.org/10.1038/s41598-023-41407-8.

Ronneberger, O., Fischer, P. und Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, S. 234–241.

Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. In: *Psychological Review*, Band 65(6):S. 386, 1958. ISSN 0033-295X. doi:10.1037/h0042519.

Rumelhart, D., Hinton, G. und Williams, R.: Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986. In: *Biometrika*, Band 71(599-607):S. 6, 1986a.

Rumelhart, D., Hinton, G. und Williams, R.: Learning representations by back-propagating errors. In: *Nature*, Band 323(6088):S. 533–536, 1986b. doi:10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S. et al.: Imagenet large scale visual recognition challenge. In: *International journal of computer vision*, Band 115:S. 211–252, 2015.

Scarpace, L., Flanders, A.E., Jain, R., Mikkelsen, T. und Andrews, D.W.: Data from REMBRANDT. In: *The Cancer Imaging Archive*, Band 10(9), 2015.

Scarpace, L., Mikkelsen, L., Cha, T., Rao, S., Tekchandani, S., Gutman, S. et al.: Radiology data from the cancer genome atlas glioblastoma multiforme [TCGA-GBM] collection. In: *The Cancer Imaging Archive*, Band 11(4):S. 1, 2016.

Seetha, J. und Raja, S.S.: Brain tumor classification using convolutional neural networks. In: *Biomedical & Pharmacology Journal*, Band 11(3):S. 1457, 2018. doi:https://dx.doi.org/10.13005/bpj/1511.

Sharif, M.I., Li, J.P., Khan, M.A. und Saleem, M.A.: Active deep neural network features selection for segmentation and recognition of brain tumors using MRI images. In: *Pattern Recognition Letters*, Band 129:S. 181–189, 2020.

Shorten, C. und Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. In: *Journal of big data*, Band 6(1):S. 1–48, 2019.

Tandel, G.S., Tiwari, A. und Kakde, O. G.: Performance optimisation of deep learning models using majority voting algorithm for brain tumour classification. In: *Computers in Biology and Medicine*, Band 135:S. 104564, 2021.

Tiwari, R. S.: Model Evaluation. *Fundamentals and Methods of Machine and Deep Learning*. John Wiley & Sons, Ltd. Kapitel 3, S. 33–100. 2022. ISBN 9781119821908. doi:https://doi-org.emedien.ub.uni-muenchen.de/10.1002/9781119821908.ch3. URL https://onlinelibrary-wiley-com.emedien.ub.uni-muenchen.de/doi/abs/10.1002/9781119821908.ch3.

Wang, J., Wang, Y., Song, J. und Cheng, H.: Iov Vulnerability Classification Algorithm Based on Knowledge Graph. In: *Electronics*, Band 12:S. 4749, 2023. ISSN 2079-9292. doi:10.3390/electronics12234749. URL https://www.mdpi.com/2079-9292/12/23/4749.

Waskom, M.L.: Seaborn: statistical data visualization. In: *Journal of Open Source Software*, Band 6(60):S. 3021, 2021. doi:10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

Özkaraca, O., Bağrıaçık, O.I., Gürüler, H., Khan, F., Hussain, J., Khan, Ja. et al.: Multiple Brain Tumor Classification with Dense CNN Architecture Using Brain MRI

Images. In: *Life (Basel, Switzerland)*, Band 13(2):S. 349, 2023. ISSN 2075-1729. doi:10.3390/life13020349. URL https://doi.org/10.3390/life13020349.