h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

Fachbereiche Mathematik und
Naturwissenschaften & Informatik

Automatische Erkennung von Biomarkern in OCT-Aufnahmen und Berechnung eines Riskscores für altersabhängige Makuladegeneration

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.) im Studiengang Data Science

vorgelegt von

Artur Moser

Matrikelnummer: 736594

Referent	:	Prof. Dr. Arnim Malcherek
Korreferent	:	Prof. Dr. Horst Zisgen

Ausgabedatum	:	01.10.2020
Abgabedatum	:	18.03.2021

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 18.03.2021

Artur Moser

ABSTRACT

The age-related macular degeneration (AMD) is one of the most common reasons why people in high age lose visual acuity. For this disease it is critical that it gets noticed at an early stage, since treatments in the later stages aren't promising. There is no treatment at all for the late stage of dry AMD. The assessment of the AMD progress of a patient is laborious and requires expertise. That's why an automatic support is of interest.

The aim of this thesis is to detect biomarkers in Optical Coherence Tomography (OCT) images and use these in combination with metadata of the patients to calculate a riskscore.

For the detection of biomarkers we're using a neural network called Mask R-CNN that can detect and classifiy objects in an image. It will be explained how the trainingdata was prepared and which difficulties arose. The neural network will detect the biomarkers in all OCT images for different examination dates of a patient to determine number and size of all biomarkers. This data will be used to train a DeepSurv model that can calculate the probability if the AMD stage of an patient will progress to the late stage in a given time.

The Mask R-CNN model was able to detect most biomarkers, but couldn't always classify them correctly. The DeepSurv model wasn't able to calculate a risk score, that represents the correct progress of the patients, with the given data.

ZUSAMMENFASSUNG

Die altersbedingte Makuladegeneration (AMD) ist einer der am häufigsten auftretenden Gründe dafür, dass Menschen im fortgeschrittenen Alter Sehschärfe verlieren. Bei dieser Erkrankung ist es kritisch, dass sie im frühen Stadium erkannt wird, da Behandlungen im fortgeschrittenen Stadium nicht erfolgversprechend sind. Für die trockene Form der AMD gibt es in der Spätform überhaupt keine Behandlungsmöglichkeiten. Die Einschätzung des AMD-Fortschritts eines Patienten ist aufwändig und erfordert Fachkenntnisse. Aus diesem Grund ist man an einer automatisierten Unterstützung interessiert.

Ziel dieser Arbeit ist es, Biomarker in Optical Coherence Tomography (OCT) Aufnahmen zu erkennen und diese, in Kombination mit weiteren Metadaten der Patienten, für die Berechnung eines Riskscores zu verwenden.

Die Erkennung der Biomarker erfolgt mittels eines Neuronalen Netzes, das verschiedene Objekte in einem Bild erkennen kann und diese einer Klasse zuordnet, dem Mask R-CNN. Es wird beschrieben, wie die Trainingsdaten aufbereitet werden, und welche Schwierigkeiten sich dabei herausstellen. Das neuronale Netz soll für alle Aufnahmen eines Patienten zu verschiedenen Untersuchungsterminen die Biomarker erkennen, um für diese Biomarker die Größe und Anzahl zu bestimmen. Anschließend werden diese Daten für das Trainieren eines DeepSurv Modells genutzt, um die Wahrscheinlichkeit zu berechnen, ob in nächster Zeit der Übergang in ein fortgeschrittenes AMD-Stadium erfolgt.

Das Mask R-CNN Modell konnte die meisten Biomarker erkennen, jedoch nicht immer richtig klassifizieren. Das DeepSurv Modell konnte mit den gegeben Daten kein Riskscore erstellen, der den tatsächlichen Verlauf der Patienten korrekt widerspiegelt.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

AMD Altersbedingte Makuladegeneration

OCT Optical Coherence Tomography

RPE Retinales Pigmentepithel

CNN Convolutional Neural Network

R-CNN Region-Based Convolutional Neural Network

RPN Regional Proposal Network

FCN Fully Convolutional Network

ROI Region Of Interest

SVM Support-Vector Machine

FPN Feature Pyramid Network

RESNET Residual Neural Network

COX PH Cox Proportional Hazard

C-INDEX Concordance-Index

BS Brier-Score

IBS Integrated Brier-Score

Teil I

THESIS

1 EINLEITUNG

Die altersbedingte Makuladegeneration (AMD) ist eine Augenkrankheit und der häufigste Grund, warum Personen älter als 50 Jahre erblinden.[1] Bei der AMD unterscheidet man zwischen der trockenen und feuchten AMD. Die trockene AMD entwickelt sich über mehrere Jahre. Erkrankte können in einen blinden Punkt im Sichtfeld nichts mehr erkennen. Dieser Punkt vergrößert sich langsam und ist besonders auffällig beim Lesen. [2] Die trockene AMD taucht häufiger auf als die feuchte Variante und es gibt keine Behandlungsmethode.[3] Eine Umstellung des Ess-, Sport- und Rauchverhalten kann jedoch einen positiven Effekt haben, wenn die AMD früh genug festgestellt wurde.[4] Der Sehverlust bei der feuchten AMD tritt über mehrere Wochen ein.[2] Es kann durch Injektionen im Auge der Verlust der Sehfähigkeit angehalten werden.[5] Bei beiden Varianten ist ein frühes Erkennen der Krankheit von Vorteil.

Das Stadium der AMD kann durch Optical Coherence Tomography (OCT)-Aufnahmen eingeschätzt werden. In den OCT-Aufnahmen sind bestimmte Biomarker sichtbar, die das Kategorisieren des AMD-Stadiums erlauben. Das Erkennen dieser Biomarker kann jedoch aufwändig sein und erfordert Spezialisten. Aus diesem Grund wird nach automatisierten Methoden geforscht, um diese Biomarker zu bestimmen.[6]

Das Universitätsklinikum Münster hat bereits in mehreren Projekten zum Thema AMD mit dem Data Science Studiengang der Hochschule Darmstadt zusammen gearbeitet.[7][8] Auch diese Arbeit wurde durch das Interesse des Universitätsklinikum ermöglicht. Das Interesse liegt dabei an einer Methode, die eine Vorhersage über den Verlauf eines AMD-Patienten erlaubt. Es wurden dafür anonymisierte Daten von AMD-Patienten bereitgestellt.

Ein Ziel dieser Arbeit ist es, die Biomarker mit Hilfe eines neuronalen Netzes zu segmentieren. Dabei wird auf dem vorherigen Projekt [8] aufgebaut. In dem Projekt wird das Mask Region-Based Convolutional Neural Network (R-CNN) verwendet, um anhand von 200 OCT-Aufnahmen drei Arten von Biomarker zu segmentieren. In dieser Arbeit werden für das Trainieren des Netzes 600 Aufnahmen verwendet, um zehn Biomarker zu segmentieren. Zudem sollen Anzahl und Größe dieser Biomarker bestimmt werden.

Das zweite Ziel dieser Arbeit ist die Verwendung der Daten, um ein Modell zu erstellen, das das Patientenrisiko, in das nächste AMD-Stadium zu wechseln, anhand eines Riskscores bestimmen kann. Hierfür wird das DeepSurv Modell ([54]) aus der Survival-Analyse verwendet. Dieses Modell er-

laubt eine zeitliche Vorhersage, wann ein Patient in die nächste AMD-Stufe wechselt. Zusätzlich kann damit der Riskscore berechnet werden.

Der Rest der Arbeit ist in vier Kapitel unterteilt. In Kapitel 2 werden die medizinischen und technischen Grundlagen geklärt. In Kapitel 3 wird das erste Ziel der Arbeit besprochen und die Ergebnisse präsentiert. Kapitel 4 behandelt das zweite Ziel und stellt die Modelle und deren Ergebnisse vor. Zum Schluss wird in Kapitel 5 die Arbeit rekapituliert und mögliche Vorschläge für Folgearbeiten erwähnt.

2 GRUNDLAGEN

In diesem Kapitel werden die Grundlagen erläutert, die für die Folgekapitel von Bedeutung sind. Anfangs werden die medizinischen Grundlagen behandelt. Dies beinhaltet zum einen den üblichen Krankheitsverlauf eines AMD-Patienten und zum anderen die verschiedenen Arten von Biomarkern, die das neuronale Netz am Ende erkennen können soll. Zu diesen gehören verschiedene Arten von Drusen und Flüssigkeiten. Anschließend wird im Abschnitt der technischen Grundlagen erläutert, welche Daten durch die Kooperation mit der Uniklinik Münster bereitgestellt wurden und es wird die Funktionsweise von Neuronalen Netzen, dem Mask R-CNN und die Survival-Analyse beschrieben.

2.1 MEDIZINISCHE GRUNDLAGEN

Bei den medizinischen Grundlagen werden die Kriterien für die verschiedenen Krankheitsphasen beschrieben. Zudem werden die relevanten Biomarker genannt und in OCT-Aufnahmen gezeigt. OCT-Bilder sind vergleichbar mit Ultraschallbildern, da beide Methoden die Erstellung von Bildern erlauben, die das Innere des Körpers zeigen. Der Hauptunterschied zwischen beiden liegt darin, dass bei der OCT Licht verwendet wird.[9]

In Abbildung 2.1 kann man den Aufbau des menschlichen Auges erkennen. Mithilfe der OCT kann man die Schichten der Netz- und Aderhaut im Mikrometerbereich betrachten.[9] Abbildung 2.2 zeigt die Schichten, die in OCT-Aufnahmen sichtbar sind. Ein Großteil der Biomarker, die in 2.1.2 vorkommen werden, sind zwischen dem retinalen Pigmentepithel (RPE) und der Bruch-Membran auffindbar. Die Biomarker, die am häufigsten auftauchen, sind verschiedene Formen von Drusen. Bei Drusen handelt es sich um extrazelluläre Ablagerungen, die sich unter dem RPE auf der Bruch-Membran ansammeln und aus zahlreichen Proteinen bestehen.[10] In Abbildung 2.3 ist eine OCT-Aufnahme mit zwei Drusen zu sehen. Die Bruch-Membran ist im Bild rot nachgezeichnet und das RPE orange. Unterhalb der Bruch-Membran ist die Aderhaut sichtbar. Die Wölbungen sind die Aderhautgefäße. Oberhalb des RPE ist die neurosensorische Netzhaut, und der schwarze Hintergrund über dieser ist der Glaskörper.

2.1.1 *Krankheitsverlauf*

Allgemein kann man den Verlauf der AMD in vier Stufen aufteilen. Die folgende Klassifizierung basiert auf der Studie "Age-Related Eye Disease Studies"[13]. Hierbei wird die AMD in nicht existent, früh, intermediär oder

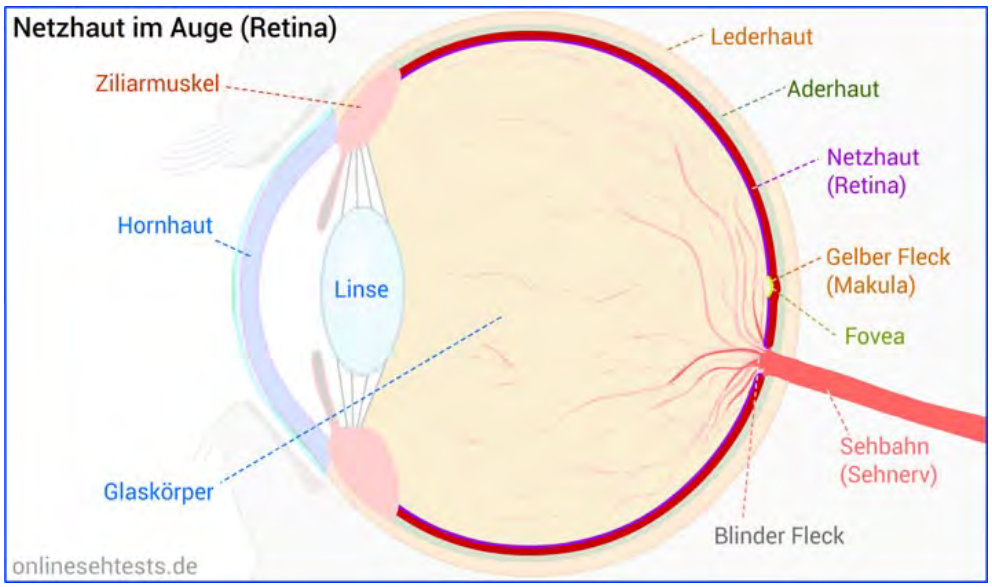

Abbildung 2.1: Aufbau des menschlichen Auges aus [11]

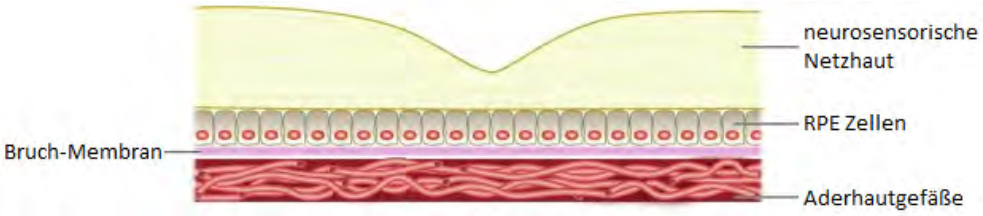

Abbildung 2.2: Schichten an der Netzhaut (übersetzt aus [12])

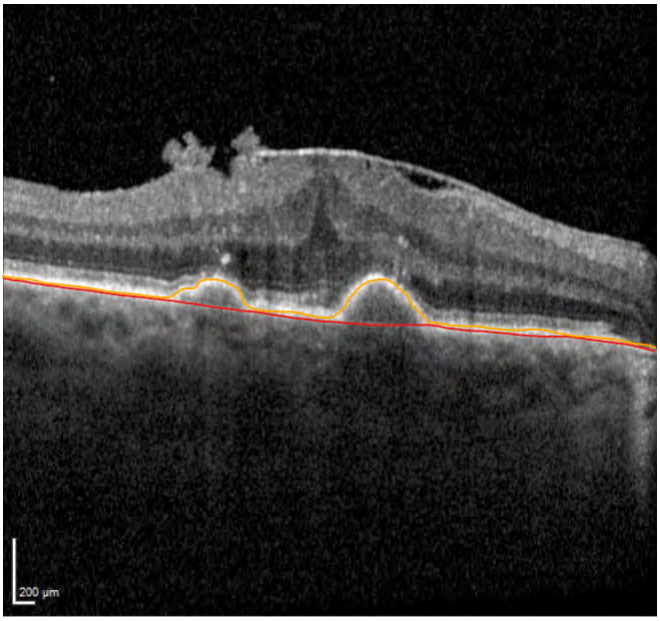

Abbildung 2.3: OCT mit eingezeichneter RPE (orange) und Bruch-Membran (rot)

fortgeschritten eingeteilt. Die Tabelle 2.1 beinhaltet die benötigten Eigenschaften für jede Stufe und entstammt [14]. Einige genannte Eigenschaften in der Tabelle sind von den angewandten neuronalen Netzen nicht erkennbar und werden deswegen nicht beachtet. Mit dem Durchmesser einer Druse ist zudem der kleinstmögliche Durchmesser gemeint. In der Tabelle unter "Fortgeschrittene AMD" werden Eigenschaften für die trockene und feuchte AMD genannt. Die Unterteilung zwischen trockener und feuchter AMD findet in der intermediären und fortgeschrittenen AMD-Stadien statt, da die Anzeichen dafür erst in diesen Stadien erkennbar sind. Trockene AMD wird als chronische Krankheit beschrieben, die mit der Zeit zu starkem Sehverlust führen kann. Zu erkennen ist diese anhand von geographischen Atrophien (2.1.2.10). Die feuchte AMD ist synonym zur neovaskulären AMD. Diese tritt plötzlicher auf und kann innerhalb von Wochen zu starkem Sehverlust führen.[15] Das Wort neovaskulär beschreibt den Prozess, bei denen sich neue Blutgefäße bilden. Weil diese Blutgefäße auslaufen wird die AMD auch "feucht" genannt.[67] Dadurch dass sich die bereitgestellten Daten auf AMD-Patienten konzentrieren, sind auch nur Patienten enthalten, die entweder zur intermediären oder fortgeschrittenen AMD eingestuft werden können. Aus diesem Grund fokussieren wir uns auf diese beiden AMD-Stadien.

2.1.2 *Biomarker*

In diesem Abschnitt werden die einzelnen Biomarker vorgestellt, die vom neuronalen Netz erkannt werden können. In dieser Arbeit sind mit Biomarkern die vorgestellten Arten von Drusen, Abhebungen, Flüssigkeiten und Atrophien gemeint. Insgesamt wird das Netz zehn Biomarker erkennen können. Für alle Biomarker werden verschiedene Metriken wie Anzahl, Fläche, Länge und Breite berechnet. In den kommenden Unterabschnitten wird für jeden Biomarker auch ein Beispielbild vorhanden sein. In nicht offensichtlichen Beispielen, sind diese auch markiert. Es gibt durchaus mehr Einteilungen für Drusen, Epithelabhebungen und Flüssigkeiten als die in diesem Kapitel beschriebenen. Theoretisch ist es auch möglich diese weiteren Biomarker in einem neuronalen Netz zu trainieren. Um dadurch jedoch ein zuverlässiges Netzmodell zu erhalten, muss man für eine Datenmenge sorgen, in der jeder Biomarker ausreichend repräsentiert ist.

2.1.2.1 *Harte Drusen*

Harte Drusen können in allen Altersgruppen auftreten.[16] Sie sind kleiner im Vergleich zu weichen Drusen. Harte Drusen haben keinen bemerkbaren Einfluss auf das Sehvermögen. Mit der Zeit können sie jedoch zu weichen Drusen heranwachsen. Laut [17] haben Patienten, die viele harte oder kutikuläre Drusen (2.1.2.4) haben, aber sonst keine sichtbaren Drusen, eine erhöhte Chance auf eine Verschlechterung der Sehfähigkeit in den nächsten zehn bis zwanzig Jahren durch retinale Epithelabhebung (2.1.2.8/2.1.2.9) oder geographische Athrophie (2.1.2.10). In der Regel sind harte Drusen wegen ihrer Größe in OCT-Aufnahmen schwerer zu erkennen. Meist sind diese

Keine AMD	kleine Drusen mit Durchmesser $< 63\mu$m Drusengesamtfläche $< 125\mu\text{m}^2$ keine Pigmentveränderungen
Frühe AMD	Mindestens eine der folgenden Eigenschaften: • Kleine Druse mit Durchmesser $< 63\mu$m oder intermediäre Druse mit Durchmesser zwischen 63 und 125μm • Drusengesamtfläche $\geq 125\mu\text{m}^2$ • Pigmentepithelveränderung durch Auftreten einer der folgenden Veränderungen: - Depigmentierung - Hyperpigmentierung $\geq 125\mu\text{m}^2$ - Hyperpigmentierung vorhanden und Depigmentierung vermutet
Intermediäre AMD	Mindestens eine der folgenden Eigenschaften: • große Drusen mit Durchmesser $> 125\mu$m • weiche, scharf begrenzte Druse mit Durchmesser $\geq 63\mu$m und gesamter Drusenfläche $\geq 656\mu\text{m}^2$ • weiche, unscharf begrenzte Druse mit Durchmesser $\geq 63\mu$m und gesamter Drusenfläche $\geq 360\mu\text{m}^2$ • nicht zentrale geographische Atrophie
Fortgeschrittene AMD	Mindestens eine der folgenden Eigenschaften: • Zentrale geographische Atrophie • Nachweis einer neovaskulären AMD durch: - fibrovaskuläre / seröse Pigmentepithelabhebung - Seröse Ablösung der neurosensorischen Netzhaut - subretinale Blutung - subretinales fibröses Gewebe - Photokoagulation im Rahmen der AMD • Sehschärfe $< 20/32$ bzw. $0,67$ aufgrund von AMD

Tabelle 2.1: Tabelle mit verschiedenen AMD-Stufen

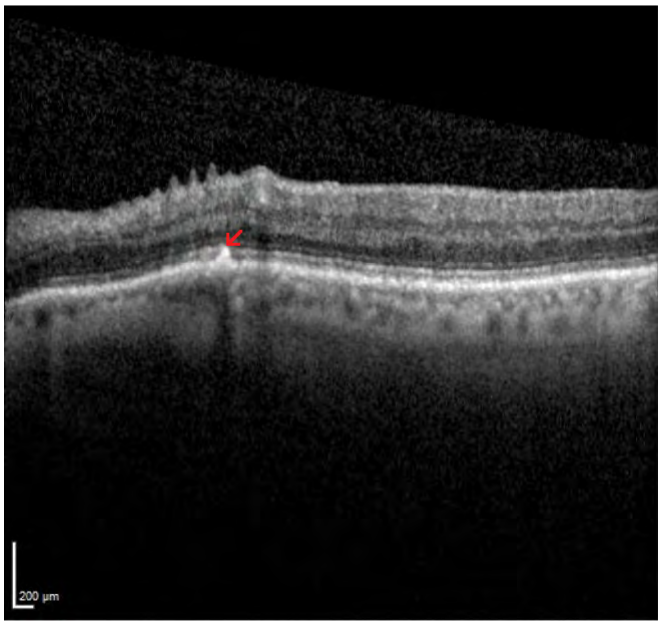

Abbildung 2.4: OCT mit harter Druse

Drusen heller im Vergleich zu anderen, was bei der Unterscheidung von anderen Drusen, vor allem von retikulären Pseudodrusen (2.1.2.3), hilft. Im Beispielbild 2.4 ist eine etwas größere harte Druse zu sehen. Es wurde ein roter Pfeil ins Bild eingefügt, welcher auf die harte Druse zeigt.

2.1.2.2 *Weiche Drusen*

Weiche Drusen sind dunkler und oft größer als alle anderen genannten Drusen. In OCT-Aufnahmen ist der Farbton hellgrau bis schwarz und zudem sind diese oft kuppelförmig.[18] Weiche Drusen sind ein erstes Anzeichen dafür, dass die AMD im Auge bereits zur intermediären Stufe vorangeschritten ist (Tabelle 2.1). An Stellen, wo sich weiche Drusen bilden, können geographische Atrophien (2.1.2.10) entstehen, die das Sehvermögen sehr stark reduzieren.[19] In Tabelle 2.1 wird zusätzlich zwischen scharf und unscharf begrenzter weicher Druse unterschieden. Das neuronale Netz erkennt zwar weiche Drusen, jedoch wurden diese nicht zusätzlich aufgrund ihrer scharfen oder unscharfen Umrandung unterschieden. In der folgenden Abbildung 2.5 ist eine kleinere weiche Druse zu sehen, die mit einem roten Pfeil markiert wurde.

2.1.2.3 *Retikuläre Pseudodrusen*

Retikuläre Pseudodrusen unterscheiden sich von Drusen anhand ihrer Zusammensetzung, da sie aus größeren Anteilen an Cholesterin und Vitronectin bestehen. Zudem sammeln sie sich auf der RPE an, statt der Bruch-Membran.[20] In frühen Stadien sind retikuläre Pseudodrusen kaum zu erkennen, da sie meist sehr klein sind, und der Farbton sich nur leicht vom Hintergrund unterscheidet. Wenn diese zum zweiten Stadium heranwachsen,

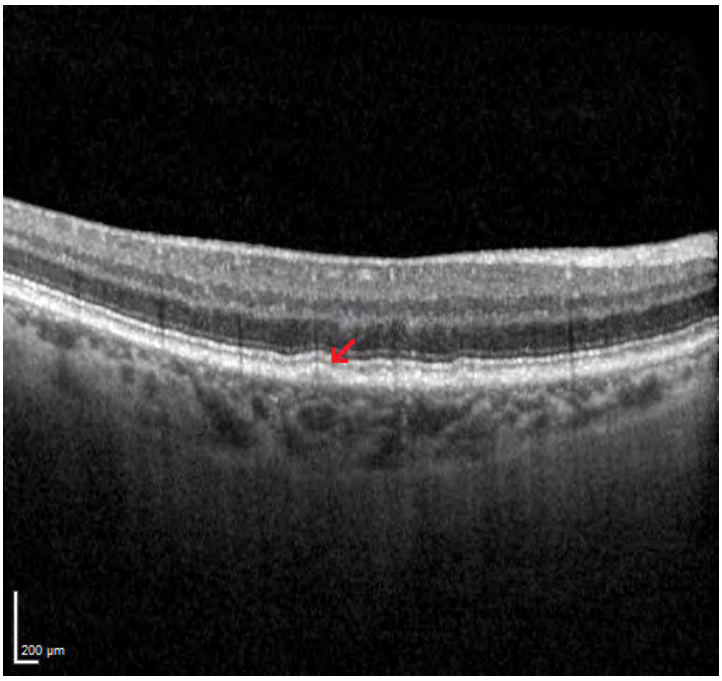

Abbildung 2.5: OCT mit weicher Druse

erkennt man, dass sie im Grauton meist zwischen weichen und harten Drusen liegen. In den letzten Stadien kann man sie gut anhand der fehlenden klaren Umrandung erkennen, die bei anderen Drusenarten durch das angehobene RPE zustande kommt. In [21] wird genannt, dass das Rauchverhalten eine Rolle spielt und weibliche Patienten eine erhöhte Chance auf retikuläre Pseudodrusen haben. In Abbildung 2.6 sieht man im linken Bild mehrere retikuläre Pseudodrusen, wobei die zwei Markierten am besten erkennbar sind. In der rechten Abbildung sieht man hingegen eine späte Pseudodruse.

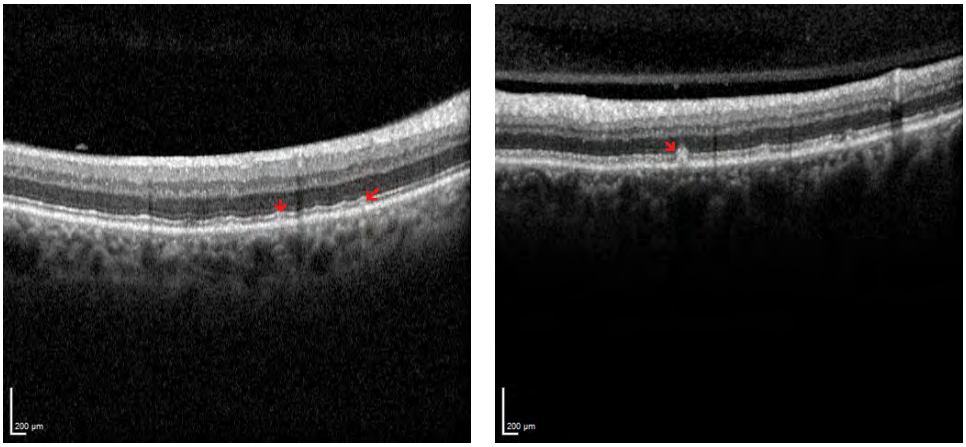

Abbildung 2.6: OCT mit retikulären Pseudodrusen

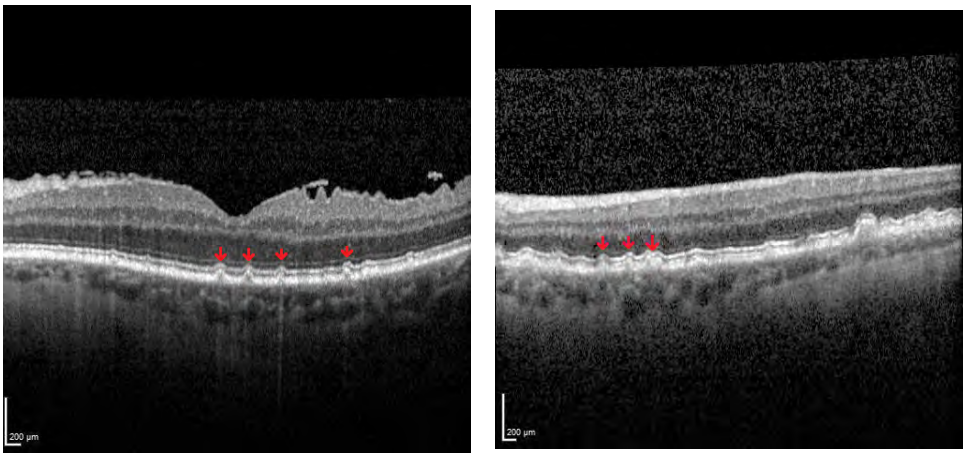

Abbildung 2.7: OCT mit kutikulären Drusen

2.1.2.4 *Kutikuläre Drusen*

Kutikuläre Drusen ähneln weichen Drusen sehr stark. Die wesentlichen Unterschiede sind eine für gewöhnlich sägezahnartige Form und ein Durchmesser von 50 bis 75μm.[22] Zudem taucht diese Art von Drusen oft gruppiert auf. Laut [17] haben Patienten mit kutikulären Drusen eine durchschnittlich bessere Sehschärfe im Vergleich zu anderen Patienten. In Abbildung 2.7 sind zwei Bilder mit mehreren kutikulären Drusen zu sehen. Die offensichtlichsten wurden mit einem Pfeil markiert. Im linken Bild kann man die Sägezahnform gut erkennen, und in dem Bild rechts davon sieht man, wie eng sich diese Drusen gruppieren können.

2.1.2.5 *Hyperreflektiver Punkt*

In OCT-Aufnahmen sind hyperreflektive Punkte weiße Stellen, die oberhalb des RPE auftauchen können. Meist sind diese kreisförmig und klein. Bei Patienten im späten AMD-Stadium kann es vorkommen, dass viele Punkte nebeneinander zu sehen sind. Hyperreflektive Punkte tauchen meist auf, wenn aufgrund von Krankheiten das RPE in die Netzhaut übergeht oder Photorezeptoren in der neurosensorischen Netzhaut degenerieren.[23] Wenn man Abbildung 2.3 betrachtet, kann man einen einzelnen hyperreflektiven Punkt direkt über der linken Druse erkennen. In Abbildung 2.8 ist neben mehreren Punkten, wovon einige mit einem Pfeil markiert wurden, auch eine große Pigmentepithelabhebung (2.1.2.8/2.1.2.9) zu sehen.

2.1.2.6 *Subretinale Flüssigkeit*

Flüssigkeiten können unter anderem durch Entzündungen von der Aderhaut in die Netzhaut (Retina) gelangen.[24] Man spricht von subretinaler Flüssigkeit, wenn sich diese Flüssigkeiten zwischen der Netzhaut und dem RPE ansammeln. Da in OCT-Aufnahmen Flüssigkeiten schwarz dargestellt werden, sind diese meistens auch gut zu erkennen. Sowohl subretinale als auch intraretinale Flüssigkeiten sind starke Indizien für eine fortgeschrittene

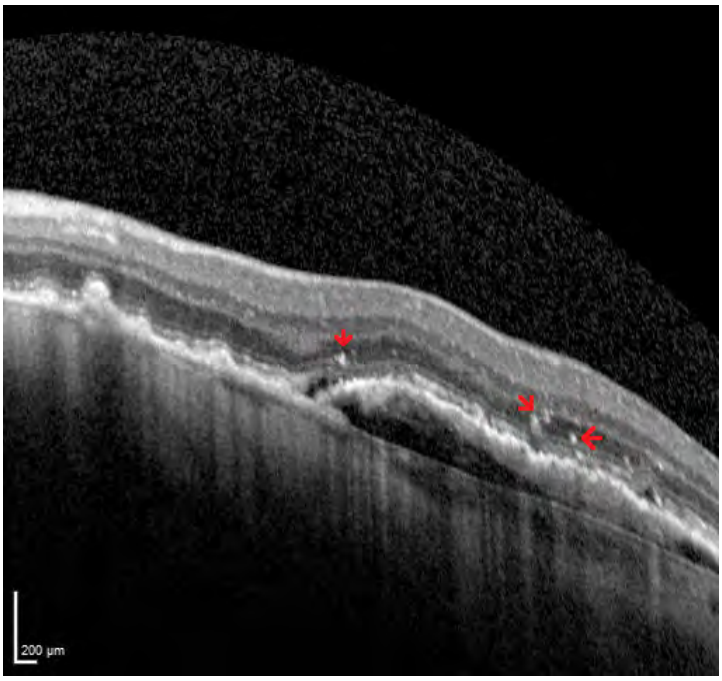

Abbildung 2.8: OCT mit hyperreflektiven Punkt

AMD, da diese meist einer Pigmentepithelabhebung folgen. Die subretinale Blutung, die in Tabelle 2.1 unter Fortgeschrittene AMD zu finden ist, zählt auch als subretinale Flüssigkeit. Laut [25] können subretinale Flüssigkeiten auf Langzeitsicht zur Erhaltung der Sehfähigkeit beisteuern. Zudem besagt dieselbe Quelle auch, dass einige Anzeichen zur Vermutung führen, dass subretinale Flüssigkeiten vor zukünftiger geographischer Atrophie schützen sollen. In Abbildung 2.8 kann man unterhalb des linken markierten hyperreflektiven Punktes und etwas weiter rechts vom rechten markierten Punkt Stellen mit subretinaler Flüssigkeit sehen. In Abbildung 2.9 erkennt man eine große Stelle mit subretinaler Flüssigkeit links neben einer ausgesprochen großen Pigmentepithelabhebung (2.1.2.8/2.1.2.9).

2.1.2.7 *Intraretinale Flüssigkeit*

Intraretinale Flüssigkeiten liegen bei OCT-Bildern in den Schichten der neurosensorischen Netzhaut. Es handelt sich dabei oft um zystenartige Blasen, die mit Flüssigkeit gefüllt sind, oder Verdichtungen der Netzhaut, durch die die Reflektivität im OCT gesenkt werden.[26] Bei beiden Möglichkeiten können Entzündungen oder Verletzungen eine Ursache sein. Laut [25] haben intraretinale Flüssigkeiten einen starken negativen Einfluss auf die Sehfähigkeit. In beiden Bildern in Abbildung 2.10 kann man mehrere intraretinale Flüssigkeiten erkennen. Im linken Bild sind nur kleine Stellen zu sehen, welche mit einem roten Pfeil markiert wurden, und im rechten Bild sieht man ein Extremfall, bei dem sich die Netzhaut an vielen Stellen verdichtet hat.

Abbildung 2.9: OCT mit subretinaler Flüssigkeit

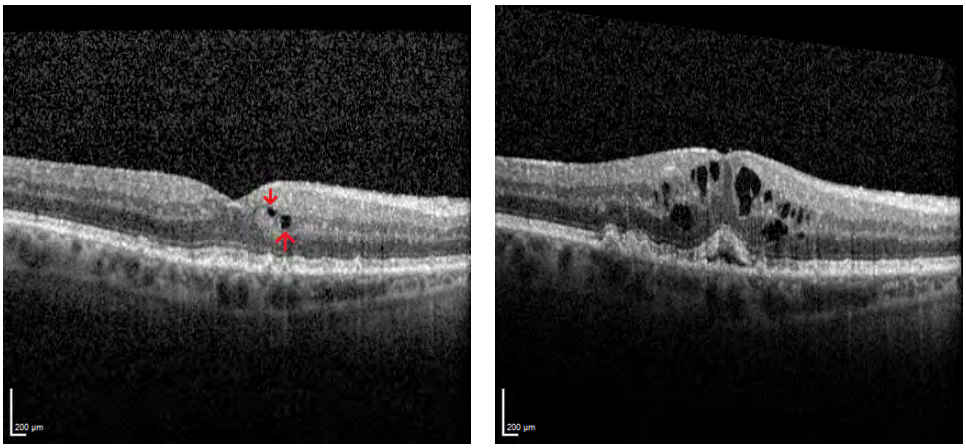

Abbildung 2.10: OCT mit intraretinaler Flüssigkeit

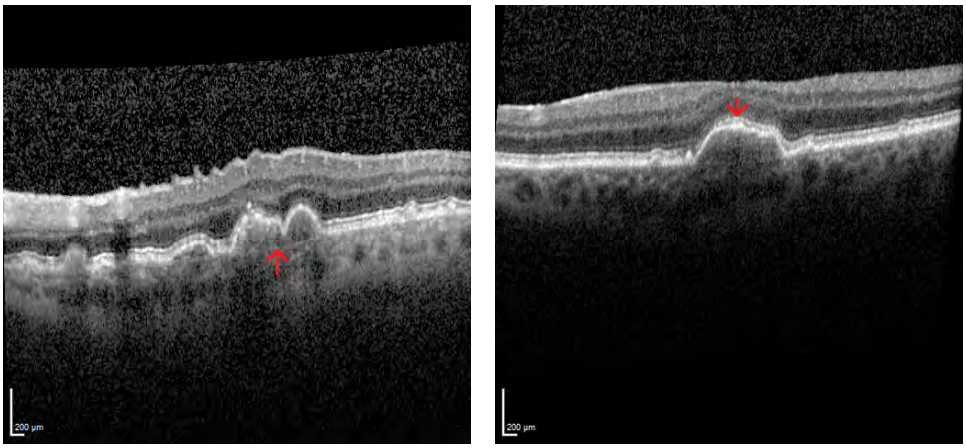

Abbildung 2.11: OCT mit drusenartiger Pigmentepithelabhebung

2.1.2.8 *Drusenartige Pigmentepithelabhebung*

Eine drusenartige Pigmentepithelabhebung kann entstehen, wenn entweder mehrere weiche Drusen sich zusammenschließen oder eine Druse zu einer bestimmten Größe heranwächst. Laut [29] muss eine Mindestbreite von der Hälfte des Durchmessers des Sehnervkopfs erreicht werden. Im Durchschnitt ist der Kopf des Sehnervs etwa 1,5mm groß, worauf die Drusen eine Gesamtbreite von ungefähr 750μm haben muss, um als drusenartige Pigmentepithelabhebung eingestuft zu werden. Drusenartige Pigmentepithelabhebungen haben eine hohe Chance, sich in den nächsten fünf Jahren zu einer geographischen Atrophie zu entwickeln und haben einen Einfluss auf den Verlust der Sehstärke.[30] In Abbildung 2.11 sieht man die beiden genannten Arten von drusenartigen Pigmentepithelabhebungen. Im linken Bild ist eine Drusengruppe zu sehen, die etwa 1200μm breit ist und rechts erkennt man eine große Druse mit einer Breite von 1000μm. Beide werden somit als drusenartige Pigmentepithelabhebung kategorisiert.

2.1.2.9 *Seröse Pigmentepithelabhebung*

Bei der serösen Pigmentepithelabhebung gelangen Flüssigkeiten durch Entzündungen oder andere Verletzungen zwischen RPE und Bruch-Membran, wodurch sich beide voneinander lösen.[27] Im Gegensatz zu drusenartigen Abhebungen sind diese durch die Flüssigkeiten in OCT-Aufnahmen meist schwarz bzw. sehr dunkel mit einer klaren Umrandung. Meist ist ihre Form oval- oder domartig, jedoch ist diese nicht darauf beschränkt. Wie bereits der Tabelle 2.1 zu entnehmen ist, sind seröse Pigmentepithelabhebungen ein starker Hinweis darauf, dass man sich bereits in der fortgeschrittenen feuchten AMD befindet. Gegen seröse Pigmentepithelabhebungen gibt es noch kein effektives Heilmittel. In [31] hat man mit Injektionen versucht, gegen seröse Pigmentepithelabhebungen anzukämpfen und somit die Sehschärfe wiederherzustellen. Obwohl durch diese Behandlung die Abhebungen zum Teil verschwunden sind, und man eine Verbesserung der Sehkraft erkennen konnte, waren die Effekte nur temporär, und die Sehschärfe der Patienten

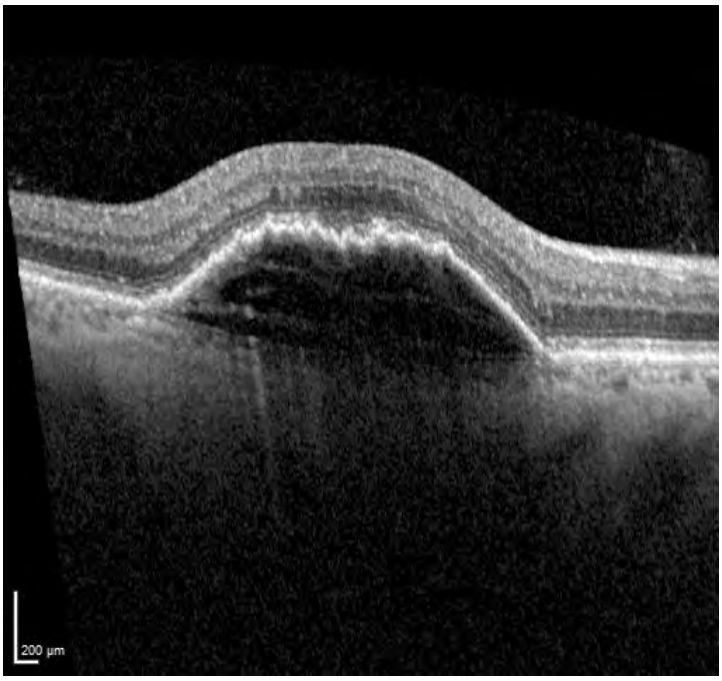

Abbildung 2.12: OCT mit seröser Pigmentepithelabhebung

nahm wieder ab. In Abbildung 2.12 ist eine etwa 3000μm breite und 300μm hohe seröse Pigmentepithelabhebung zu sehen.

2.1.2.10 *Geographische Atrophie*

Im Gegensatz zu allen anderen genannten Arten von Biomarkern erkennt man Geographische Atrophien unterhalb der Bruch-Membran in der Aderhaut. Sie sind als trockene Form der fortgeschrittenen AMD bekannt und werden als Verkümmerung des RPE und der Aderhaut beschrieben.[72] Oft entstehen geographische Atrophien an Orten, in denen Drusen präsent waren, und in manchen Fällen können auch drusenartige Epithelabhebungen Grund für das Auftauchen sein.[28] In OCT-Aufnahmen sind Atrophien als große weiße Stellen in der Aderhaut zu erkennen. In Abbildung 2.13 sind zwei OCT-Bilder zu sehen. Beide Bilder zeigen denselben Schnitt eines Patienten zu zwei verschiedenen Zeitpunkten. Zwischen den Aufnahmen liegt etwa ein Jahr. Im linken Bild sieht man eine große Epitelabhebung mit etwas Flüssigkeit. Im rechten Bild sieht man, dass die Abhebung in die Höhe gewachsen ist und die komplette Retina an der Stelle verbogen hat. Mittig kann man gut die geographische Atrophie erkennen. An dieser Stelle ist das RPE auch besonders dünn und fast nicht mehr wahrzunehmen. Geographische Atrophien haben einen großen Einfluss auf den Verlust der Sehfähigkeit.[32]

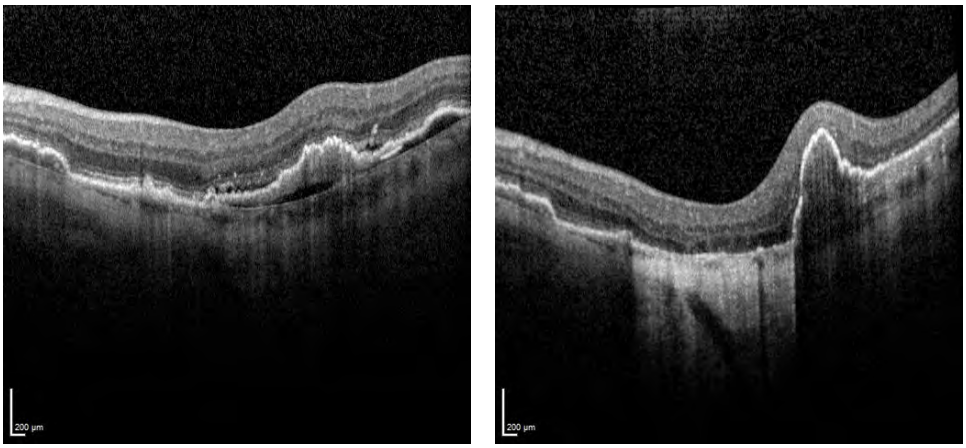

Abbildung 2.13: OCT mit geographischer Atrophie

2.2 TECHNISCHE GRUNDLAGEN

In diesem Abschnitt werden die Grundlagen auf der technischen Seite erläutert. Dies beinhaltet die ungefähre Struktur und Transformation der erhaltenen Daten und die Funktionsweise von neuronalen Netzen. Zudem wird geklärt, wodurch sich das Mask R-CNN von einem gewöhnlichen Netz unterscheidet. Zum Schluss wird die Survival-Analyse beschrieben.

2.2.1 *Datengrundlage*

Bei der Datengrundlage wird erklärt, welche Daten die Universitätsklinik Münster überreicht hat, und wie diese verarbeitet wurden. Insgesamt wurden drei verschiedene Arten von Daten von ausgesuchten Patienten geschickt. Jeder Patient wurde vor der Übergabe der Daten mit einer Patienten ID anonymisiert. Zu jedem Patienten existieren Bild- und Metadaten. Für viele wurden zusätzlich noch Visusdaten übergeben. Dank der Patienten ID ist das Zusammensetzen der verschiedenen Datensätze möglich. Alle Daten sind nach der Struktur Person, Auge, Termin aufgebaut. Das heißt, dass die Kombination aus Patienten ID, rechtes oder linkes Auge und Datum der Untersuchung in tabellarischen Datensätzen einzigartig ist.

2.2.1.1 *Bilddaten*

Die Bilddaten wurden als eine Videodatei im avi-Format versendet. Die Videos bestehen aus 25 Bildern. Diese Bilder sind OCT-Aufnahmen von verschiedenen Bereichen der Netzhaut. In Abbildung 2.14 ist ein Beispielbild einer solchen Aufnahme zu sehen. Rechts ist das bereits bekannte OCT-Bild zu sehen. Bei dem linken Bild handelt es sich um ein Schwarz-Weiß-Bild der Retina und ermöglicht so gesehen eine Draufsicht. Der dunkle Halbkreis links im Bild ist Teil des Sehnervs. Die dunkelgrünen Linien zeigen, an welcher Stelle die OCT-Aufnahmen entstanden sind. Der hellgrüne Pfeil zeigt die Position des rechten Bildes. Da für das Erkennen der Drusen nur

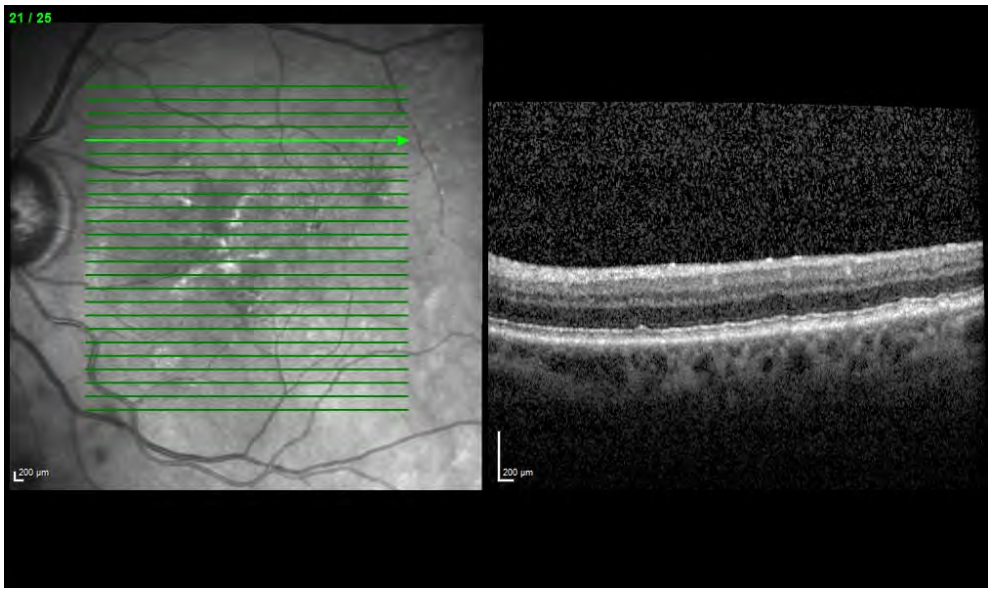

Abbildung 2.14: Ausschnitt aus überreichter Videodatei

das rechte Bild benötigt wird, wird mit einem Python-Skript jede dieser Videodateien in 25 Bilder transformiert und anschließend so geschnitten, dass nur noch der rechte Teil zu sehen ist.

2.2.1.2 *Metadaten*

Bei den Metadaten handelt es sich um eine Tabelle im xlsx-Format. Hier wurden zu jedem Termin eines Patienten einige Eigenschaften und Messungen erfasst. Die wichtigste Einschätzung ist, ob der Patient sich in der intermediären oder fortgeschrittenen AMD befindet. Dies nutzen wir in Kapitel 4, um die Survival-Modelle (2.2.7) zu trainieren. Die genauen Spalten zusammen mit den ersten drei Einträgen der Tabelle kann man in Appendix A finden.

2.2.1.3 *Visusdaten*

Die Visusdaten enthalten die Ergebnisse des Sehtests zu einem Termin und wurden in einer separaten Tabelle versendet. Die Informationen über die Sehstärke der Patienten zu den verschiedenen Terminen ist nicht für alle Termine vorhanden. Die Skala der genutzten dezimalen Sehstärke geht von 0 bis 2. Dabei ist eine Sehstärke von 1 ein durchschnittlicher Wert und ein Wert von 2 der Bestmögliche. Bei einer Sehstärke kleiner $0,1$ wird man als blind eingestuft. Die Patienten, in den überreichten Visusdaten, haben einen Höchstwert von $1,25$.

Da die Survival-Modelle in Abschnitt 4 keine fehlenden Werte akzeptieren, müssen Ersatzwerte verwendet werden. Hierfür nutzen wir die durchschnittliche Sehstärke der vorhandenen Patienten. Beim Mitteln der Sehstärke ist darauf zu achten, dass die Abstufungen der Sehstärken logarithmisch sind, weswegen die Visuswerte vor dem Mitteln logarithmiert werden müssen.

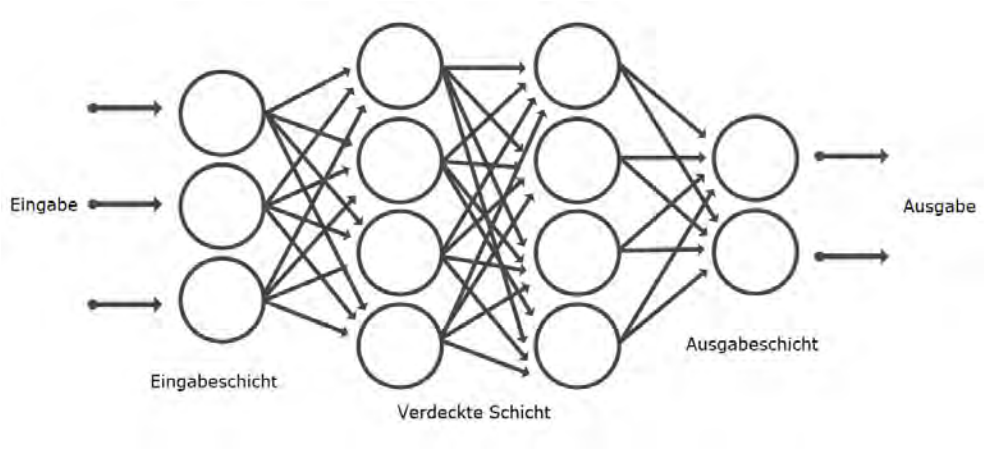

Abbildung 2.15: Aufbau eines neuronalen Netzes (übersetzt aus [35])

Der anschließende Mittelwert wird dann entlogarithmiert. Wenn nun die Visusdaten für einen Patienten komplett fehlen, wird der errechnete Mittelwert für alle Patienten von 0,36 verwendet. Falls bei einem Patienten Visusdaten vorhanden sind, aber an einem Termin fehlen, wird der Durchschnitt für den einzelnen Patienten berechnet, statt dass der Durchschnittswert für alle Patienten verwendet wird.

2.2.2 *Neuronale Netze*

Dieser Abschnitt erklärt vereinfacht den Aufbau und die Arbeitsweise von neuronalen Netzen. Eine genauere Erklärung kann man in Kapitel 5 von [33] finden. Neuronale Netze erhielten ihren Namen von neuronalen Netzen im Gehirn von Lebewesen. Grund dafür ist die Anlehnung des Modells an die natürlichen neuronalen Netze, bei denen verschiedene Neuronen miteinander verbunden sind und ein Netzwerk bilden. Die theoretischen Grundlagen dafür führen bis hin zu 1873 in der Quelle [34] zurück. In diesem Netzwerk werden verschiedene Signale zwischen den verbundenen Neuronen gesendet. Ähnlich sind auch die Knotenpunkte von künstlichen neuronalen Netzen aufgebaut. Bei dem hier beschriebenen Netz handelt es sich um ein vorwärtsgerichtetes Netz. Den Namen verdankt es der Tatsache, dass es die Knoten in eine Richtung abarbeitet. Dies kann man auch anhand der Struktur in Abbildung 2.15 sehen. Bei anderen Varianten eines neuronalen Netzes können auch andere Bedingungen und Eigenschaften für Knoten, Kanten und Schichten gelten.

Knotenpunkte sind mit Kanten verbunden und diese haben eine variable Gewichtung. Eine Zahl wird über die Kanten zwischen Knotenpunkten gesendet. In den Knotenpunkten wird eine Aktivierungsfunktion auf die Zahl, die vorher mit der Gewichtung der Kante multipliziert wurde, angewandt und anschließend als Ausgabe an den nächsten Knotenpunkt gesendet.

Ein Knoten kann auch mehrere Ein- und Ausgabeverbindungen haben. Alle Eingabeparameter müssen dafür mit ihren Gewichten durch eine Propagierungsfunktion zu einer einzigen Zahl zusammengefasst werden, bevor das Ergebnis der Propagierungsfunktion anschließend an die Aktivierungsfunktion weitergegeben wird. Bei der Propagierungsfunktion handelt es sich in der Regel um die gewichtete Summe von allen Eingabeparametern und Gewichtungen der Kanten. Das Ergebnis des Knotenpunktes wird an alle Ausgabeverbindungen gesendet.

Theoretisch ist die einzige Bedingung für eine Aktivierungsfunktion, dass diese differenzierbar ist. In der Praxis sorgen die meisten verwendeten Aktivierungsfunktionen jedoch auch dafür, dass der Ergebnisbereich limitiert ist. Damit wird dafür gesorgt, dass das Ergebnis in einem bestimmten Wertebereich liegt. Oft geht der Wertebereich von 0 bis 1 oder von -1 bis 1.

Neuronale Netze haben, neben den gewöhnlichen Knotenpunkten, auch eine beliebige Anzahl von Eingabe- und Ausgabeknotenpunkten. In den Eingabeknotenpunkten werden die Parameter an das Netzwerk weitergegeben, und in den Ausgabeknoten werden die Ergebnisse dann ausgegeben. Die Knoten eines neuronalen Netzes werden in Schichten unterteilt. Alle Eingabeknoten liegen in der Eingabeschicht und alle Ausgabeknoten in der Ausgabeschicht (siehe Abbildung 2.15). Die anderen Knotenpunkte, die dazwischen sind, liegen in verdeckten Schichten. Neuronale Netze können auch miteinander gekoppelt werden, wodurch die Ein- und Ausgabeknoten mit anderen Netzen verbunden sein können. Den Aufbau eines neuronalen Netzes kann man in Abbildung 2.15 sehen. Dabei repräsentieren die Kreise Knotenpunkte und die Pfeile zwischen den Knoten Kanten.

Wenn man bei neuronalen Netzen von Training spricht, redet man davon, die verschiedenen Gewichtungen zwischen den Knoten im Netz anzupassen, um ein besseres beziehungsweise passenderes Ergebnis zu erhalten. Dieses Verfahren soll einen Lernprozess imitieren und besteht aus zwei Schritten, dem Berechnen des Netzfehlers und der Backpropagation.

Um zu wissen, wie die Gewichtungen angepasst werden sollen, benötigt man Trainingsdaten, die Eingabeparameter enthalten und die zu erwarteten Ergebnisse. Die Anfangsgewichte der Kanten werden zufällig gewählt. Die Parameter der Trainingsdaten werden zunächst dem neuronalen Netz als Eingabe gegeben, um ein Ergebnis zu bekommen, welches man mit einer Fehlerfunktion mit dem Ergebnis aus dem Trainingsdatensatz vergleichen kann. Die Fehlerfunktion liefert den Unterschied zwischen dem erwarteten und tatsächlichen Ergebnis als Zahl. Dieses Ergebnis der Fehlerfunktion wird Fehler genannt und Ziel der Anpassung der Gewichte ist es, den Fehler zu minimieren.

Um herauszufinden, mit welchen Wert die Gewichtungen jeweils angepasst werden müssen, wird der Backpropagationsalgorithmus angewendet. Hierfür wird durch partielle Differenzierung der Fehler- und Aktivierungsfunktionen der Gradient für jeden Knotenpunkt berechnet, um ein Optimum zu finden. Durch die Eigenschaft des Gradienten, in Richtung der steilsten Steigung an einem Punkt zu zeigen, ist es mit diesem möglich, ein zumindest lokales Minimum zu finden. In diesem Schritt ist auch die Lernrate relevant. Diese bestimmt den Faktor, wie stark die Gewichtung in Richtung des Gradienten geändert werden soll. Bei einer hohen Lernrate ist es wahrscheinlicher, dass die Änderung zu groß ist und das Minimum überschritten wird. Bei einem zu kleinen Wert werden mehr Durchführungen des Backpropagationalgorithmus benötigt, um ein gewünschtes Ziel zu erreichen. Der Algorithmus wird wiederholt, bis ein Abbruchkriterium erreicht wurde. Dieses kann eine vorgelegte Anzahl an Wiederholungen sein oder ein bestimmter Wert, den die Fehlerfunktion unterschreiten soll.

Wenn man nun ein Modell optimiert hat, ist es möglich, dass das neuronale Netz zu sehr auf die Trainingsdaten angepasst wurde und beim Verwenden ähnlicher Daten, die nicht Teil der Trainingsdaten waren, unerwartete Ergebnisse zeigt. Dieses Phänomen nennt man Overfitting. Um dagegen vorzugehen, teilt man den Datensatz vor dem Trainieren in Trainings- und Validierungsdaten auf. Die Gewichte werden dann mit den Trainingsdaten angepasst und mit den Validierungsdaten kann nach dem Anpassen der Gewichte der Fehler des Netzes bestimmt werden, um zu Überprüfen, ob das Abbruchkriterium erreicht wurde.

2.2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) werden auch Faltungsnetze genannt und sind neuronale Netze, die häufig bei der Bilderkennung verwendet werden, um Objekte in Bildern zu erkennen und zu klassifizieren. Die Informationen und Abbildungen, die in diesem Abschnitt vorkommen, entstammen Kapitel 3.5 von [36]. Der Ursprung des CNN entsprang dem Neocognitron, welcher 1980 von Kunihiko Fukushima erwähnt wurde. Dieses Netz konnte erstmalig Strukturen unabhängig von ihrer Position erkennen.[37] Im Jahr 1989 hat Yann LeCun das erste moderne CNN erstellt, dass mit Backpropagation gelernt hat.[38] Bei Faltungsnetzen werden die Gewichtungen als Matrizen zusammengefasst. Diese Matrizen nennt man Faltungskern oder Filter. Beim Trainieren des Netzes werden die Werte dieser Faltungskerne angepasst. Auch die Knoten der Eingabeschicht werden zu Matrizen umgewandelt.

Da bei CNN meist Bilder als Eingabe verwendet werden, ist die Transformation zu Matrizen recht simpel. Schwarz-Weiß- und Graustufen-Bilder lassen sich einfach umwandeln, da man für jeden Pixel einen Wert finden kann, 0 oder 1 für Schwarz-Weiß-Bilder und 0 bis 255 für Graustufen-Bilder. Ein Bild

mit Breite n und Höhe m, also $n \times m$ Pixel, wird in eine $n \times m$ Matrix umgewandelt. Bei Farbbildern werden die RGB-Werte jedes Pixels verwendet. Ein RGB-Wert besteht aus einem Tripel mit drei Zahlen. Jede Zahl geht für gewöhnlich von 0 bis 255 und repräsentiert den Farbwert für die Farben Rot, Grün und Blau. Aus diesem Grund wird aus einem Farbbild mit $n \times m$ Pixel drei $n \times m$ Matrizen. Jede dieser Matrizen erhält den Farbwert einer der drei genanten Farben für jeden Pixel im Eingabebild.

Neben den bereits genannten Schichten, die in Abbildung 2.15 zu sehen sind, verwenden Faltungsnetze zwei weitere Arten von Schichten: die Faltungs- und die Poolingschicht. Bei der Faltungsschicht wird der Faltungskern verwendet und über die Eingabematrix geschoben. Die überlappenden Zahlen des Faltungskerns und der Eingabematrix werden multipliziert und anschließend summiert. Diese Summe ist dann ein Wert der Ausgabematrix. Die Ausgabe der Pooling- und Faltungsschicht wird auch Feature-Map genannt. Mit jedem Schieben des Faltungskerns, ändern sich die überlappenden Zahlen und somit auch der Wert der Ausgabematrix für die nächste Position. In Abbildung 2.16 ist ein Beispiel zu sehen, bei dem eine 4×4-Matrix mit einem 3×3-Faltungskern gefaltet wird. Die Ausgabe ist eine 2×2-Matrix. Der gelb markierte Teil der Eingabematrix zeigt an, an welcher Stelle der Faltungskern überlappt, um die erste Position der Ausgabematrix zu bestimmen. Bei der Berechnung handelt es um die Multiplikation und Addition der Eingabematrix und des Faltungkern, um den Wert für die Ausgabematrix zu berechnen. Für den zweiten Wert der Ausgabematrix wird der gelbe Bereich in der Eingabematrix nach rechts geschoben und die Berechnung wiederholt. Also wird $2 \cdot 9 + 3 \cdot 8 + 4 \cdot 7 + 6 \cdot 6 + 5 \cdot 7 + 8 \cdot 4 + 10 \cdot 3 + 11 \cdot 2 + 12 \cdot 1 = 237$ für die zweite Position der Ausgabematrix berechnet. Für den Fall, dass der Faltungskern den rechten Rand erreicht hat, wird diese für die nächste Position eine Zeile nach unten geschoben und wieder an den linken Rand gesetzt. Dies wird wiederholt, bis der Faltungskern das Ende der Eingabematrix erreicht hat.

Anders als bei der Faltungsschicht wird bei der Poolingschicht kein Faltungskern verwendet, sondern ein Pooling-Operator. Dies hat zur Folge, dass solche Schichten nicht durch den Trainingsalgorithmus angepasst werden, da veränderbare Parameter fehlen. Durch das Pooling soll die Dimension der Matrizen reduziert werden. Dafür werden Bereiche der Matrix zusammengefasst. Max-Pooling ist einer der am häufigst verwendeten Pooling-Operatoren. Hierbei wird das Maximum des Bereichs wiedergegeben. Zusätzlich wird gemerkt, an welcher Position sich das Maximum befand. Dies ist für den Backpropagationsalgorithmus wichtig, um die Eingabematrix zum Teil wiederherzustellen. Anders als bei der Faltung schneiden sich die Bereiche nicht, die durch den Pooling-Operator zusammengefasst werden. Beim Pooling wird ein Pooling-Faktor p verwendet, der die Größe des Poolingbereichs bestimmt. Ein Pooling-Faktor von p führt zu einem $p \times p$ Bereich. In Abbildung 2.17 ist ein Max-Pooling-Beispiel zu sehen. Bei der

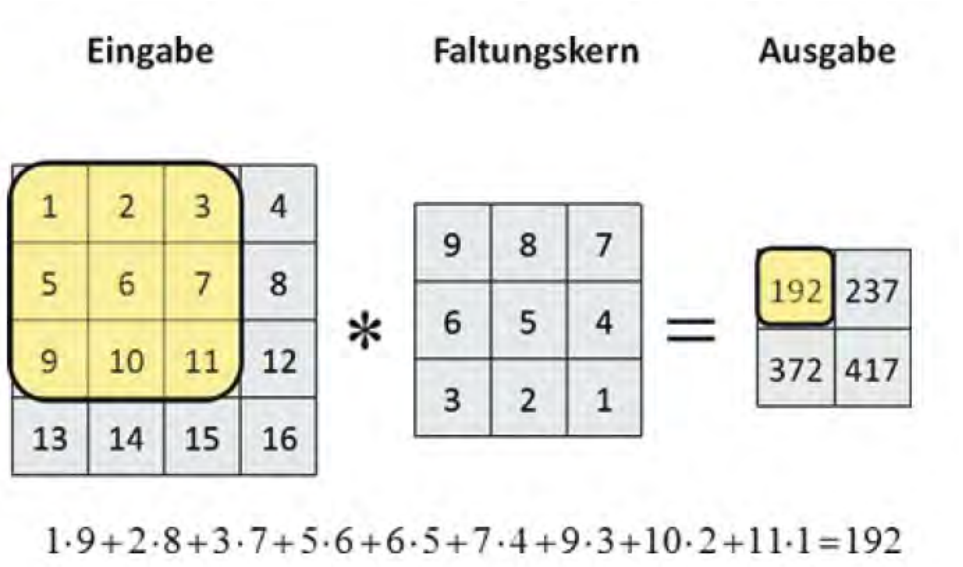

Abbildung 2.16: Beispiel einer Faltung aus [36]

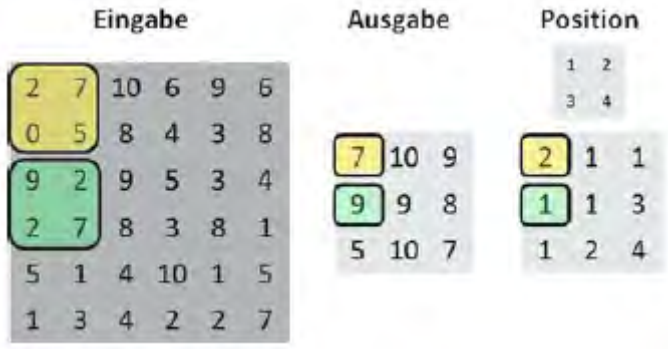

Abbildung 2.17: Beispiel eines Max-Pooling-Operators aus [36]

Eingabe handelt es sich um eine 6×6 Matrix, die durch das Max-Pooling mit dem Pooling-Faktor 2 zu einer 3×3 Matrix zusammengefasst wurde. Die Positionsmatrix ist im selben Format wie die Ausgabematrix und zeigt an, an welcher Stelle die größte Zahl im Poolingbereich der Eingabematrix liegt. Die Positionsmatrix wird jedoch nicht mit ausgegeben. Durch die Eigenschaften der Faltungs- und Poolingschicht, sich in Bereichen zu bewegen und diese zusammenzufassen, fällt es Faltungsnetzen leichter lokale Merkmale zu erkennen.

In Abbildung 2.18 ist der Aufbau eines Faltungsnetzes zu sehen. Als Eingabe wird ein 28×28 Schwarz-Weiß-Bild verwendet. In der 1. Faltungsschicht werden drei verschiedene Faltungskerne genutzt, um drei 24×24 Matrizen zu erhalten. Jede der Verbindungen vor der Faltungsschicht repräsentiert ein Faltungskern. In der zweiten Faltungsschicht werden 18 Faltungskerne verwendet, um drei Eingabematrizen zu sechs Ausgabematrizen zu transformieren. In diesem Fall werden auf jede Eingabematrix mit drei Kernen

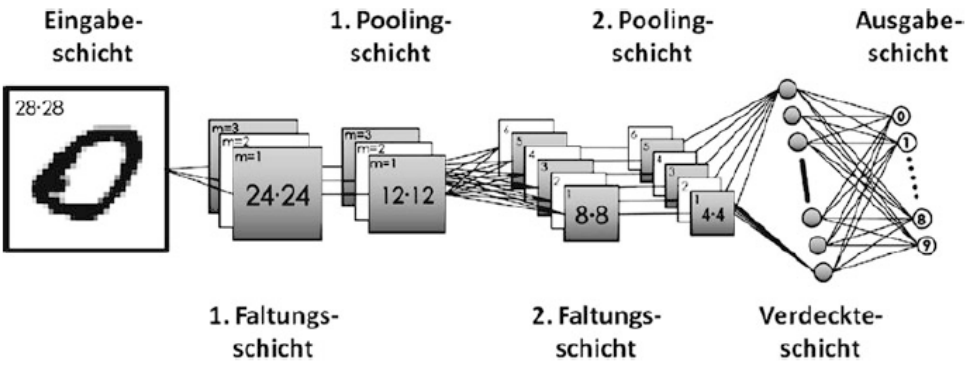

Abbildung 2.18: Beispielstruktur eines CNN aus [36]

wie in Abbildung 2.16 gefaltet und das Ergebnis nochmals addiert. Alle besagten Faltungskerne haben die Größe 5×5. Das Verwenden mehrerer Faltungskerne für eine Schicht erhöht die Anzahl an änderbarer Parameter und damit die Lernfähigkeit des Netzes. Da nicht alle Kerne voneinander abhängig sind, können damit auch unterschiedliche Eigenschaften gelernt werden.

In beiden Poolingschichten wird durch den Poolingfaktor 2 die Matrixgröße von 24×24 auf 12×12 in der Ersten und von 8×8 auf 4×4 in der zweiten Schicht reduziert. Die verdeckte Schicht funktioniert genau wie die verdeckten Schichten aus Abschnitt 2.2.2. Jeder Knoten der verdeckten Schicht ist mit jeder Zahl aus der zweiten Poolingsschicht verbunden. Damit hat jeder Knotenpunkt in der verdeckten Schicht für sechs 4×4 Matrizen $4 \cdot 4 \cdot 6 = 96$ Eingabeparameter.

Die zu sehende Ausgabeschicht wird für Klassifizierungsaufgaben verwendet. In diesem Fall ist jeder Klasse ein Ausgabeknotenpunkt gewidmet, der die Wahrscheinlichkeit berechnen soll, dass diese Klasse im Bild zu sehen ist. Die Klasse mit dem höchsten Wahrscheinlichkeitswert wird anschließend ausgesucht.

Zum Trainieren des Netzes wird der Backpropagationalgorithmus mit ein paar Veränderungen verwendet. Die ersten Schritte sind noch identisch mit dem in Abschnitt 2.2.2 beschriebenen Algorithmus. Zuerst werden zufällige oder vorbestimmte Werte für die Gewichte bestimmt und anschließend für die Testdaten die Ergebnisse des Netzes berechnet. Nachdem die Ergebnisse mit den tatsächlichen Daten in der Fehlerfunktion verglichen wurden, startet die Backpropagation. Für die Schichten zwischen Ausgabeschicht und verdeckter Schicht ist dies identisch mit Abschnitt 2.2.2. Die Gewichte zwischen verdeckter Schicht und der Poolingschicht werden genauso berechnet, bilden jedoch Fehlerwertmatrizen. Im oben genannten Beispiel wären das für die zweite Poolingschicht sechs Fehlermatrizen im 4×4 Format. Um nun das Pooling rückgängig zu machen und die vorherige Dimension zu erhalten, wird auf die Fehlermatrix das Uppooling verwendet. Dafür ist die be-

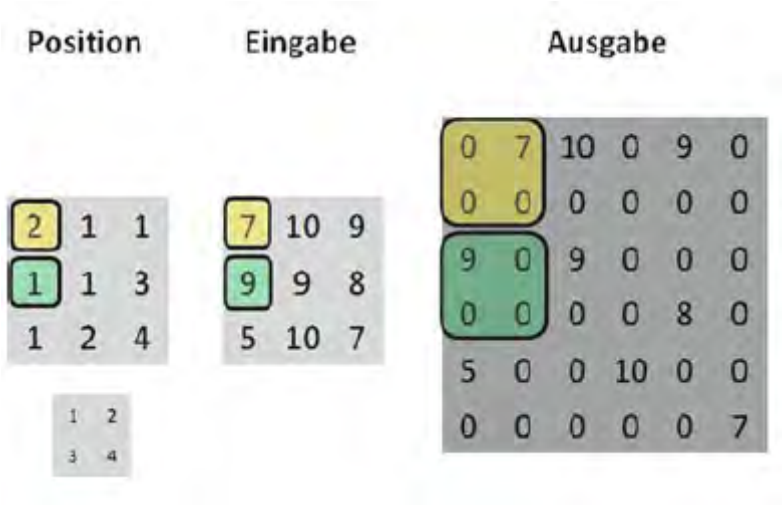

Abbildung 2.19: Uppoolingsbeispiel aus [36]

reits genannte Positionsmatrix von Relevanz, die die Position des Maximum in dem Bereich verrät. Die nicht-maximalen Werte werden mit 0 gefüllt. Ein Beispiel ist in Abbildung 2.19 zu sehen.

Um die Faltung rückgängig zu machen, werden die Fehlermatrizen mit dem geflippten Faltungskern gefaltet. Bei einem geflippten Faltungskern werden die Werte um die horizontale und vertikale Achse getauscht. Abstrakt kann man sich das als eine Punktspiegelung mit der Mitte der Matrix vorstellen. Um nun die Fehlermatrizen der vorherigen Schicht zu erhalten, müssen die berechneten Fehlermatrizen entsprechend ihrer Verbindungen summiert werden. Diese summierten Fehlermatrizen werden aber nicht das gewünschte Format haben, weswegen die Randbereiche mit Nullen aufgefüllt werden müssen. Wenn nun die Fehlermatrizen für alle Schichten berechnet wurden, können die Gewichtungen innerhalb der Faltungskerne mit den Fehlermatrizen und der Lernrate angepasst werden. Beim Trainieren von CNNs ist zusätzlich drauf zu achten, dass alle Bilder die selbe Größe haben oder zur selben Größe transformiert werden, da die Eingabeschicht und alle Pooling- und Faltungsschichten auf bestimmte Dimensionen angepasst sind. Dies führt dazu, dass das Klassifizieren nur mit einem gleich großen Eingabebild möglich ist. Zudem muss aufgrund des Faltungskerns das Eingabebild auch quadratisch sein.

2.2.4 *Mask R-CNN*

Dieser Abschnitt erklärt, wodurch sich das Mask R-CNN vom einfachen CNN unterscheidet. Das Mask R-CNN ist keine direkte Weiterentwicklung eines CNN. Dazwischen liegen einige Architekturen, die das CNN erweitern und durch weitere Funktionen ergänzen. Diese Architekturen werden in Unterabschnitten kurz zusammengefasst. Anschließend wird auf das Mask R-CNN eingegangen. Die meisten der Informationen in diesem Abschnitt stammen

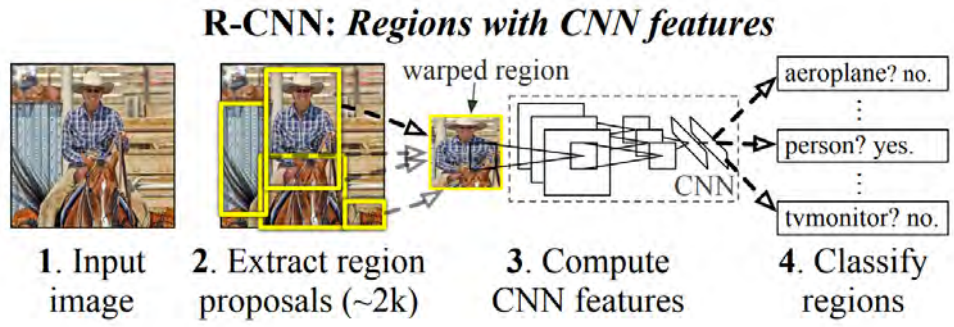

Abbildung 2.20: R-CNN aus [40]

aus [39]. Eventuelle Ergänzungen werden im entsprechenden Unterabschnitt angemerkt.

2.2.4.1 *R-CNN*

Die Informationen aus diesem Abschnitt kommen aus [40]. Beim gewöhnlichen CNN ist es nur möglich zu sagen, was für eine Klassifizierung im Bild gefunden wurde. R-CNN erweitern CNN durch eine Positionsangabe des zu klassifizierenden Objekts. Diese Positionsangabe erfolgt durch eine Bounding Box, einen Rahmen um das Objekt.

Dem R-CNN setzt voraus, dass durch einen Algorithmus Regionen aus dem Eingabebild vorgeschlagen werden. In [40] werden einige dieser Algorithmen genannt, die kompatibel mit dem R-CNN sind. Einer dieser Algorithmen ist der Selective-Search-Algorithmus ([41]), welcher in [40] verwendet wird. Dieser sucht sich bis zu 2000 Bereiche von Interesse, auf Englisch Region of Interest (ROI), aus einem Bild und schlägt diese anschließend dem R-CNN vor. Bei den ROI handelt es sich um Ausschnitte des Bildes, die ein zu klassifizierendes Objekt beinhalten könnten.

Diese ROI werden zu einer festen quadratischen Größe transformiert und anschließend einem CNN als Eingabe übergeben. Das CNN klassifiziert jedoch die Region nicht, sondern fasst seine Ausgabeknoten zu einen Vektor zusammen, welcher an diverse lineare Support-Vector Maschinen (SVM) gesendet wird, die das klassifizieren für die ROI übernehmen. Lineare SVM sind binäre Klassifikator, die Daten in Klassen unterteilen können. Mehr dazu ist in Kapitel 4 von [42] finden. Jeder dieser SVM ist für das Klassifizieren einer einzigen Klasse zuständig. In Abbildung 2.20 ist der Prozess visuell dargestellt. Nach dem Klassifizieren wird der Rand des ROI regressiert und angepasst, damit die gezeigten Bounding Boxes das komplette Objekt umschließen und nicht nur einen Teil davon. Dies wird für jede ROI wiederholt. Dadurch wird der Prozess sehr rechenintensiv, da pro Bild bis zu 2000 ROI klassifiziert werden.

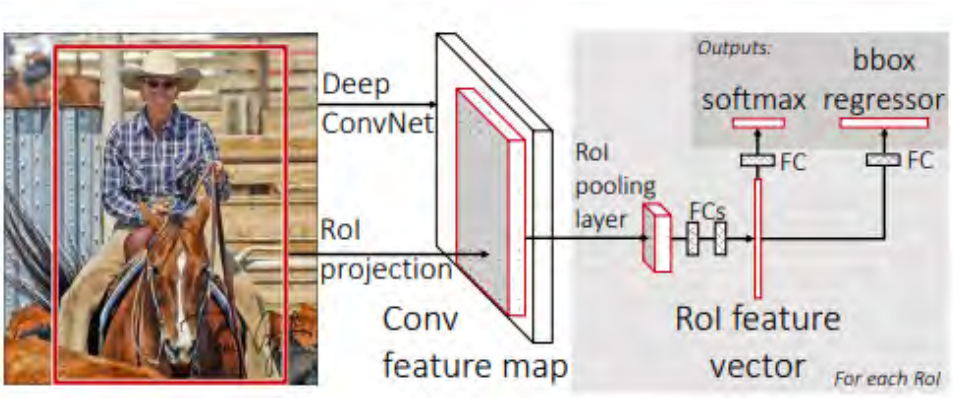

Abbildung 2.21: Fast R-CNN aus [43]

2.2.4.2 *Fast R-CNN*

In [43] sind die Informationen aus diesem Abschnitt zu finden. Die größten Nachteile des R-CNN sind die langsame Objekterkennung (47 Sekunden pro Bild) und das Trainieren der SVM, die auf den Ausgabe-Vektor des CNN aufbauen. Das Fast R-CNN benötigt weiterhin vorgeschlagene ROI. In Abbildung 2.21 kann man den Ablauf des Fast R-CNN sehen. Das Fast R-CNN bekommt als Eingabe das Eingabebild und die vorgeschlagenen ROI für das Bild. Das Bild läuft durch mehrere Faltungs- und Poolingschichten und bildet eine Feature-Map. Dieser Prozess wird Feature Extraction genannt. Alle vorgeschlagenen ROI laufen durch eine ROI-Poolingschicht. In dieser Schicht wird Max-Pooling auf den ROI verwendet um eine Feature-Map mit fester Größe zu erhalten. In den darauf folgenden "fully connected layers" (FC-Schicht) werden diese Feature-Maps zu Feature-Vektoren umgewandelt. Fully connected layers sind Schichten von Knotenpunkten, bei denen jeder Knoten der einen Schicht mit jedem Knoten der Anderen verbunden ist. Die Feature-Vektoren werden an zwei FC-Schichten mit verschiedenen Aufgaben gesendet. Die eine Schicht soll die ROI klassifizieren. In Abbildung 2.21 erfolgt diese Klassifizierung mit der Softmax-Funktion. Diese ist differenzierbar und hat als Ausgabe den Maximalwert aller Eingabeparameter. Dadurch wird dem ROI die Klasse mit der höchsten Wahrscheinlichkeit zugeteilt. Der Hintergrund wird auch als eine Klassifizierung gezählt und ist am höchsten, wenn kein Objekt im ROI zu sehen ist. Die zweite FC-Schicht gibt Änderungsparameter für die Bounding Box des ROI an, damit das komplette zu klassifizierende Objekt in der Bounding Box liegt.

2.2.4.3 *Faster R-CNN*

Informationen aus diesem Abschnitt entstammen [44]. Bei dem Faster R-CNN werden die ROI nicht mehr von einem Algorithmus vorgeschlagen, sondern vom Netz selbst berechnet. Den Aufbau des Faster R-CNN kann man in Abbildung 2.22 sehen. Zu Beginn werden die Features des Eingabebildes durch mehrere Faltungs- und Poolingschichten extrahiert, um daraus Feature-Maps zu bilden. Diese Feature-Maps dienen einem Regional Proposal Network

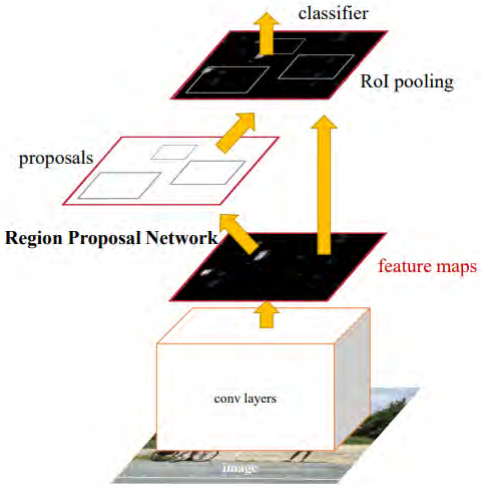

Abbildung 2.22: Faster R-CNN aus [44]

(RPN) als Eingabe, welches ROI vorschlägt. Die ROI werden dann zusammen mit den Feature-Maps analog zum Fast R-CNN klassifiziert.

Das RPN gibt die ROI als Rechteck an und erteilt jedem ROI einen Score, der aussagt, wie wahrscheinlich es sich bei dem ROI um ein zu klassifizierendes Objekt handelt. Damit das RPN Vorschläge erzeugt, wird ein kleines neuronales Netz mit fester Größe über die Feature-Maps gezogen, ähnlich wie die Faltungskerne. Dieses Netz wird Sliding Window genannt. Als Ausgabe gibt dieses Sliding Window einen Feature-Vektor, welcher an eine Box-Klassifizierungsschicht und eine Box-Regressionsschicht weitergeleitet wird. Das Sliding Window kann an jeder Stelle mehrere ROI vorschlagen. Die Box-Regressionschicht berechnet die Koordinaten von möglichen ROI im Sliding Window und die Box-Klassifizierungsschicht kümmert sich um den Wahrscheinlichkeitsscore für diese ROI, ob ein zu klassifizierendes Objekt präsent ist.

2.2.4.4 *Mask R-CNN*

In der bereits genannten Quelle [39] wird das Mask R-CNN zum ersten Mal erwähnt. Das Mask R-CNN erweitert das Faster R-CNN mit einer weiteren Abzweigung, die für jede ROI zusätzlich Segmentierungsmasken erstellt. Die Masken werden durch ein kleinen Fully Convolutional Network (FCN) erstellt. FCN sind Netze, die eine pixel-basierte Segmentierung von Objekten ermöglicht.[45] Jeder Pixel wird dabei klassifiziert. In Abbildung 2.23 ist die Ergänzung der Segmentierungsabzweigung sichtbar. Mit dem Backbone in dieser Abbildung ist ein CNN gemeint, das die Features aus einem Bild extrahiert. Dies ist äquivalent mit den "conv layers" aus Abbildung 2.22. Durch das Verwenden einer ROI-Poolingschicht ist es dem Fast R-CNN und Faster R-CNN nicht möglich eine pixelgenaue Maske zu erstellen, da beim Pooling exakte räumliche Informationen verloren gehen. Das Pooling hat ei-

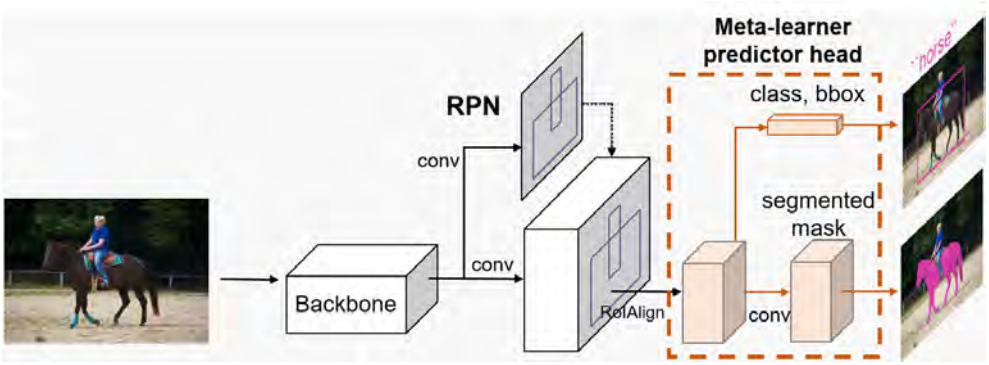

Abbildung 2.23: Mask R-CNN Aufbau (Ausschnitt aus [46])

ne große Auswirkung auf das pixelgenaue Segmentieren aber eine Kleine auf die Klassifizierung. Deswegen wird die ROI-Poolingschicht mit einer ROI-Alignschicht ersetzt. Diese erlaubt ein genaueres Abbilden der ROI auf der Feature-Map durch bilineare Interpolation, während die Feature-Map der ROI zu einer festen Größe transformiert wird. Die Segmentierung verläuft parallel und unabhängig von der Klassifizierung. In einem ROI wird stets für jede Klasse eine Maske erstellt. Durch die Klassifizierungsschicht wird lediglich entschieden, welche Maske für das ROI verwendet wird.

2.2.5 *Verwendete Struktur*

In diesem Abschnitt werden die Schichten des Mask R-CNN präsentiert, die tatsächlich verwendet wurden. Bei dem verwendeten Netz handelt es sich um eine Python-Implementierung von Matterport, die basierend auf [39] veröffentlicht wurde. Matterport ist ein Unternehmen, das sich auf Daten- und 3D-Technologien spezialisiert hat. In Appendix B sind alle Schichten genannt. Zu jeder Schicht wird die Art der Schicht genannt und welche Schicht als Eingabe benutzt wird. In Appendix B ist auch der ungefähre Startpunkt der jeweiligen Architektur gekennzeichnet. Das Mask R-CNN kann grob in drei Bestandteile kategorisiert werden. Diese Bestandteile sind Backbone, RPN und das eigentliche Mask R-CNN. Das Backbone extrahiert die Features aus einem Bild und erzeugt daraus eine Feature-Map, die das Bild repräsentiert. In [39] wird von einer Kombination aus Residual Neural Network (ResNet) und Feature Pyramid Network (FPN) als Backbone gesprochen. Diese Variante wurde auch von Matterport implementiert. Aus der erstellten Feature-Map wird im RPN nach ROI gesucht, die Objekte beinhalten könnten. Diese ROI werden anschließend dem eigentlichen Mask R-CNN weitergegeben. Mit ROIAlign werden die ROI auf der Feature-Map abgebildet. Zum Schluss wird für jede ROI die Klasse, Bounding Box und Maske erstellt.

2.2.5.1 *ResNet*

ResNet kann als Residuen-Netzwerk übersetzt werden und wurde erstmals in [47] veröffentlicht. Ein Problem bei standardmäßigen Netzwerk-Architektur-

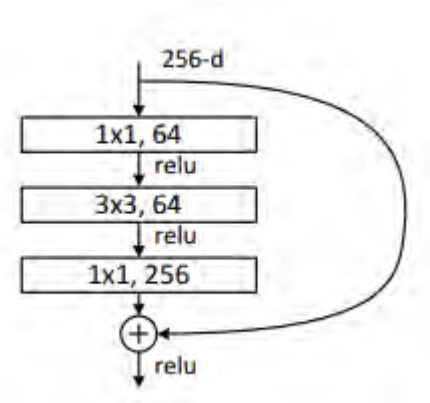

Abbildung 2.24: Residuenblock des ResNet aus [47]

en ist, dass erhöhte Trainingsfehler auftauchen, je tiefer das Netz geht. Unter anderem kann sich der Fehler durch das "Vanishing Gradient"-Problem erhöhen, bei dem die Gradienten der Verlustfunktion gegen Null konvergieren, wenn das Netz zu tief ist. ResNet versucht das Problem der höheren Trainingsfehler mit "Skip Connections" zu lösen. Wenn bei der Backpropagation eine Schicht für In- und Output dieselben Dimensionen hat, wird die Identitätsfunktion benutzt um die Schicht zu überspringen. Die Gewichte werden dadurch dem Output des übersprungenen Layers einfach hinzugefügt. In Abbildung 2.24 ist ein Residuenblock der ResNet Architektur zu sehen. Die Skip-Connection ist in der Abbildung das Plus-Symbol. Die komplette ResNet-50 Architektur ist auf [1] sichtbar. Die Zahl beim Namen ResNet-50 stellt die Anzahl der verwendeten Schichten dar. Bei ResNet-50 wird der Residuenblock, der in Abbildung 2.24 zu sehen ist, 16 mal wiederholt. Pro Residuenblock wird die Faltung drei mal verwendet. Durch die Schichten vor und nach der wiederholenden Residuenblöcke wird der Anzahl der Schichten jeweils eine Schicht hinzugefügt. Dadurch erhält man $1 + 3 \times 16 + 1 = 50$ Schichten. Die verwendete Aktivierungsfunktion relu, gibt die Eingabe aus, solange diese positiv ist. Bei einer negativen Eingabe gibt die Funktion 0 aus. Das 1×1 bzw. 3×3 steht für die Größe des verwendeten Faltungskerns und die Zahl hinter der Größe ist die Anzahl der verwendeten Faltungskerne.

2.2.5.2 *Feature Pyramid Network*

Direkt nach dem ResNet-50 folgt in der verwendeten Struktur aus Appendix B das FPN. Dieses wurde in [48] vorgestellt. Eine Feature-Pyramide kann man sich als Stapelung von mehreren Feature-Maps eines Eingabebilds vorstellen. Die Feature-Maps haben dabei unterschiedliche Dimensionen, da die Auflösung (Höhe und Breite) der Feature-Map durch Faltung reduziert wurde. Dafür wird die Feature-Map jedoch tiefer. Diese Darstellung erlaubt eine skaleninvariante Objekterkennung. Dadurch kann dieses Netz zum Beispiel ein kleines Objekt im Hintergrund eines Bildes erkennen, obwohl das Objekt in den Trainingsdaten nur im Vordergrund auftaucht. In Abbildung 2.25

1 https://transcranial.github.io/keras-js/#/resnet50

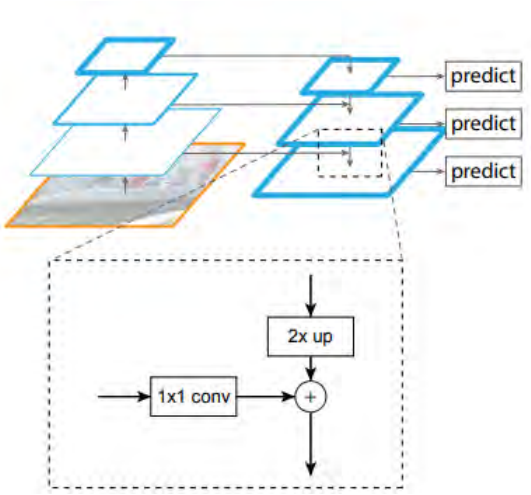

Abbildung 2.25: Feature Pyramid Network aus [48]

kann man den Ablauf des FPN sehen. In dem Bild kann man auch die pyramidenförmige Darstellung der Feature-Maps sehen. In der Abbildung ist das Bild mit orangenen Rahmen das Eingabebild und die blauen Quadrate die Feature-Maps. Der Prozess kann grob in zwei Pfade geteilt werden. Der erste Pfad ist der "bottom-up pathway", der auf der linken Seite der Abbildung dargestellt ist. Hierbei werden die Feature-Maps des Eingabebildes durch mehrfache Faltung erstellt. In der verwendeten Struktur in Appendix B übernimmt das ResNet-50 diesen Teil. Zwischen den Residuenblöcken wird nach einer Anzahl von Blöcken die Auflösung der Feature-Map reduziert. Aus Platzgründen ist die Ausgabedimension in Appendix B nicht sichtbar. Die Faltungen zur Auflösungssreduktion finden in Appendix B direkt nach den Skip-Connections "add_19", "add_23", "add_29" und "add_32" statt. An diesen vier Punkten gibt das ResNet-50 ebenfalls die derzeitige Feature-Map an das FPN weiter.

Der zweite Pfad ist der "top-down pathway", der auf der rechten Seite der Abbildung zu sehen ist. Jede der Feature-Maps wird zunächst mit einem 1×1-Faltungskern gefaltet. Dadurch verringert sich die Tiefe der Feature-Map. Nach der Faltung hat jede der Feature-Maps die selbe Tiefe. Daraufhin wird mit "UpSampling" und Addition mit der jeweils gefalteten Feature-Map von oben nach unten die nächste Feature-Map erzeugt. Beim Upsampling wird die Auflösung der Feature-Map durch "Nearest-neighbor" erhöht. Dabei werden die blanken Stellen der Feature-Map einfach durch Kopien des nächsten Nachbarn gefüllt. Zum Schluss werden diese neuen Feature-Maps nochmals gefaltet und dem RPN überreicht. Das verwendete RPN funktioniert wie das in Abschnitt 2.2.4.3 beschriebene, mit dem Unterschied, dass es mehrere Feature-Maps als Eingabe bekommt und in allen Feature-Maps nach ROI sucht.

2.2.5.3 *Mask R-CNN*

Das implementierte Mask R-CNN hat verschiedene Strukturen, je nachdem, ob das Netz lernt (Training) oder Objekte in einem Eingabebild erkennen soll (Inference). Wie bereits in 2.2.4.4 beschrieben wurde, läuft die Segmentierung der Maske und Klassifizierung parallel. Dies ist nur beim Training der Fall. Die gezeigte Struktur in Appendix B zeigt die Inference-Struktur. Bei dieser wird für jedes ROI zuerst die Klasse und Bounding Box bestimmt und danach die Maske erstellt. In Appendix C ist der Mask R-CNN Abschnitt für die Trainingsstruktur zu sehen. Im Rest diesen Abschnitts wird nur noch von der Inference-Struktur gesprochen.

Zu Beginn werden die ROI mit ROIAlign auf jeder Feature-Map der Feature-Pyramide abgebildet und zu einer festen Größe transformiert. Anschließend werden diese durch mehrere Faltungsschichten bearbeitet. In Appendix B sind diese durch einer Kombination aus zwei TimeDistributed- und einer Activationschicht dargestellt. Mit TimeDistributed ist die TimeDistributed-Klasse aus der Keras Bibliothek gemeint. Diese erlaubt das gleichzeitige Anwenden der Schicht auf einen kompletten Batch. Ein Batch ist eine Ansammlung von Stichproben. Nach der Berechnung der Klasse und Bounding Box aller ROI werden diese dem DetectionLayer weitergegeben. Hier werden die klassifizierten ROI angepasst und gefiltert, um Überlappungen der ROI zu umgehen und um ROI mit einem zu geringen Score zu entfernen. Diese neuen gefilterten ROI werden nochmals mit ROIAlign transformiert und anschließend mehrfach gefaltet, um die Maske zu erstellen.

2.2.6 *Confusion Matrix*

Information aus diesem Abschnitt entstammen [49] und [50]. Eine Confusion Matrix fasst die Performance eines Klassifikationsmodells als Matrix zusammen. In 2.26 ist die Darstellung der Confusion Matrix zu sehen. Es wird anhand von Daten gezählt, wie oft das Modell eine Klasse erkannt oder nicht erkannt hat. Zusätzlich wird unterschieden, ob die Klasse tatsächlich zu erkennen war oder nicht. Dadurch erhalten wir vier Werte, die in der Confusion Matrix dargestellt werden. Bei den Werten handelt es sich um True Positives (TP), False Positives (FP), True Negatives (TN) und False Negatives (FN). Positives und Negatives spiegelt in dem Kontext wieder, ob eine Klasse zu sehen ist oder nicht. Anhand dieser Werte können wir die Sensitivität und Spezifität berechnen, die in Kapitel 3 zur Modellauswahl beitragen. Die Sensitivität ist als $TP/(TP+FN)$ definiert und beschreibt den Anteil der Positives, der richtig klassifiziert wurde. Die Spezifität hingegen wird als $TN/(TN+FP)$ definiert und beschreibt den Anteil der Negatives, der richtig klassifiziert wurde.

		Assigned Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Abbildung 2.26: Confusion Matrix aus [49]

2.2.7 *Survival-Analyse*

Die Information in diesem Abschnitt entstammt [51] und Kapitel 14 von [52]. Bei [51] handelt es sich um die Dokumentation des Pysurival-Pakets für Python. Erstellen der Modelle, Graphiken und Riskscores erfolgten mit diesem Paket. Die Survival-Analyse beschreibt eine Art von statistischen Modellen, die das Auftreten und die Zeit eines Events analysiert. Daten für Survival-Analysen bestehen somit aus den erklärenden Variablen X, dem Zeitpunkt des Events T und dem Event E. In der Praxis kann das Event eine Vielzahl von Anwendungsmöglichkeiten finden, da das Auftreten des Event als dichotome Variable dargestellt wird und gut eine Zustandsveränderung wiedergeben kann. Einige Beispiele für ein Event wären das Abbezahlen eines Kredits, das Auftreten einer Krankheit oder der Todesfall.

Anders als übliche Regressionsanalysen profitiert die Survival-Analyse von zensierten Daten. Mit zensierten Daten sind Fälle gemeint, in denen das Event nicht eingetreten ist. Solche Fälle verraten uns nicht, wann genau das Event passiert ist. Jedoch erhalten wir Informationen darüber, dass das Event in der beobachteten Zeit nicht aufgetreten ist. Da die Survival-Analyse auch den Zeitpunkt betrachtet, wann das Event aufgetreten ist, erlaubt dies die Prognose, wie wahrscheinlich das Event zu einem Zeitpunkt T auftritt.

Die Survival-Analyse wird durch fünf Funktionen beschrieben, die miteinander zusammenhängen. Sei $f(t)$ eine Wahrscheinlichkeitsdichtefunktion von der Lebenszeit T und T eine positive stetige Zahl, dann ist die kumulierte Verteilungsfunktion $F(t)$ definiert als

$$F(t) = P(T \leq t) = \int_0^t f(x)\mathrm{d}x.$$

Mit $F(t)$ lässt sich die Wahrscheinlichkeit berechnen, ob ein Event bis zum Zeitpunkt t eingetreten ist. Die kumulierte Verteilungsfunktion $F(t)$ zweier Patienten ist in Abbildung 2.27 als Beispiel eingezeichnet. In der Abbildung stellt die Zeitachse (t-Achse) die Anzahl der Monate seit dem ersten Termin der Patienten dar und die y-Achse die Wahrscheinlichkeit, dass ein Wechsel in die späte AMD stattgefunden hat. Wenn man bei $t = 50$ die beiden Kurven betrachtet, erkennt man, dass Patient A eine 80%-ige Wahrschein-

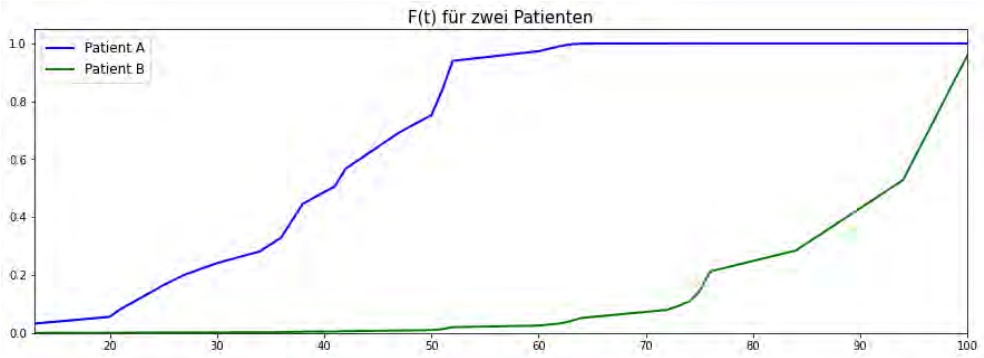

Abbildung 2.27: Beispiel einer kumulierten Verteilungsfunktion zweier Patienten

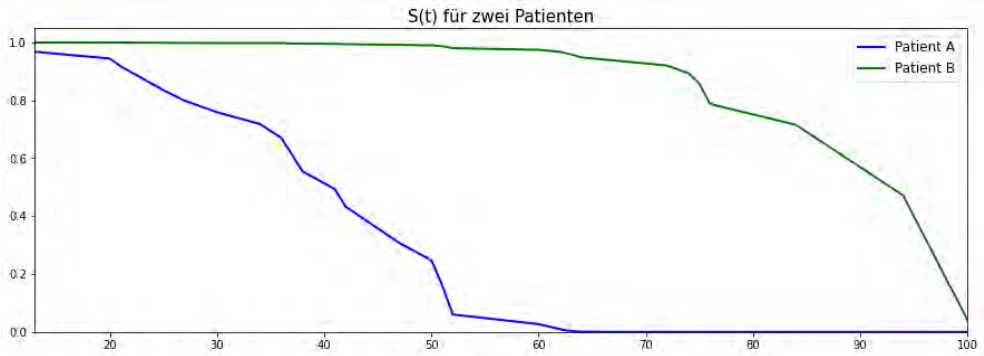

Abbildung 2.28: Beispiel einer Survival-Funktion zweier Patienten

lichkeit hat innerhalb dieser 50 Monate in die späte AMD zu wechseln, während die Wahrscheinlichkeit von Patient B nahe 0 ist. Man erkennt in dieser Abbildung, dass Patient A über die komplette Zeitspanne eine höhere Wahrscheinlichkeit hat, in die späte AMD zu wechseln.

Die Survival-Funktion $S(t)$ kann als Gegenstück zur kumulierten Verteilungsfunktion betrachtet werden. Die Funktion beschreibt die Wahrscheinlichkeit, dass ein Event bis zum Zeitpunkt t nicht eingetreten ist und ist definiert als

$$S(t) = 1 - F(t) = P(T > t).$$

In Abbildung 2.28 ist die Survival-Funktionen für dieselben Patienten wie in Abbildung 2.27 dargestellt. In der Abbildung erkennt man, dass Patient A bei $t = 50$ eine 20%-ige Wahrscheinlichkeit besteht, dass für den Patienten innerhalb der nächsten 50 Monate kein Wechsel in die späte AMD stattfinden wird. Im Grunde kann man aus beiden Grafiken dieselben Schlüsse ziehen.

Die Hazard-Funktion $h(t)$ beschreibt das Risiko, dass ein Event naheliegend zum Zeitpunkt t stattfindet. Das Risiko zum Zeitpunkt t ist groß, wenn die Überlebenswahrscheinlichkeit bis zum Zeitpunkt t, beschrieben durch $S(t)$, klein ist, oder die Wahrscheinlichkeit, dass ein Event stattfindet, be-

schrieben durch $f(t)$, zum Zeitpunkt t groß ist. Die Hazard-Funktion ist definiert als

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T \leq t + \Delta t | T \geq t)}{\Delta t} = \frac{f(t)}{S(t)}.$$

Damit lässt sich die kumulierte Hazard-Funktion $H(t) = \int_0^t h(x)\mathrm{d}x$ berechnen. In dieser Arbeit nutzen wir $F(t)$, um die Wahrscheinlichkeit, dass ein Patient in die späte AMD wechselt, grafisch darzustellen. In den Modellen, die in Abschnitten 2.2.7.1 und 2.2.7.2 beschrieben werden, wird $h(t)$ geschätzt. Die kumulierte Hazard-Funktion wird in Abschnitt 2.2.7.3 genutzt, um den Riskscore zu berechnen.

2.2.7.1 *Cox Proportional Hazard*

Das Cox Proportional Hazard (Cox PH) Modell wurde erstmals 1972 in [53] vorgestellt. Informationen aus diesem Abschnitt entstammen [53], [52] und [54]. Modelle der Survival-Analyse versuchen die Hazard-Funktion $h(t)$ vorherzusagen. Das Cox PH Modell beschreibt diese mit folgenden Modell. Sei $X = (X_1, X_2, \ldots, X_p)$ der Vektor der erklärenden Variablen und X^T deren Transponierte, dann ist das Modell definiert als

$$\log \frac{h(t, X)}{h_0(t)} = X^T \beta = \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p.$$

Das Modell kann auch als $h(t, X) = h_0(t) e^{(X^T \beta)}$ beschrieben werden. Dabei ist $h_0(t)$ eine unbekannte Funktion, die Grundgefahr (baseline hazard) genannt wird und den Fall beschreibt, dass alle erklärenden Variablen null sind. [53] Das Cox PH Modell hat die Eigenschaft, dass die erklärenden Variablen zu jeder Zeit den selben Effekt auf das Modell haben, da die Zeit t nur für die Grundgefahr eine Rolle spielt. Für zwei Stichproben i und j mit den Vektoren der erklärenden Variablen x_i und x_j kann das Verhältnis der Hazard-Funktion beider Stichproben zum Zeitpunkt T wie folgt dargestellt werden.

$$\frac{h(T, x_i)}{h(T, x_j)} = e^{(x_i - x_j)\beta}$$

Die rechte Seite der Gleichung ist nicht von der Zeit abhängig. Dadurch hat die Hazard-Funktion zweier Stichproben immer dasselbe Verhältnis unabhängig von der Zeit. Aus diesem Grund hat das Modell auch proportional im Namen. Die β-Parameter werden durch die partielle Maximum-Likelihood-Methode geschätzt. Die Methode wird partiell genannt, da $h_0(t)$ bei der Methode nicht beachtet wird. Sei k die Anzahl aller Stichproben, bei denen das Event stattgefunden hat, x_i die erklärenden Variablen der i-ten Stichprobe, T_i der Zeitpunkt des Events der i-ten Stichprobe und $\mathfrak{R}(T_i)$ die Menge aller Stichproben, bei denen das Event noch nicht zum Zeitpunkt T_i stattgefunden hat. [54] In $\mathfrak{R}$ sind auch die Stichproben vorhanden, bei denen kein Event

stattgefunden hat. Die partielle Log-Likelihood-Funktion ist dann definiert als

$$l(\beta) = \sum_{i=1}^{k} x_i \beta - \sum_{i=1}^{k} \log \left(\sum_{j \in \Re(T_i)} e^{x_j \beta} \right).$$

Diese Funktion wird maximiert, um die β-Parameter zu schätzen. In [53] wird keine Annahme über $h_0(t)$ gemacht. In der implementierten Version von [51] wird $h_0(t)$ durch folgende Formel geschätzt. Sei T ein Zeitpunkt, an dem ein Event stattgefunden hat und d_T die Anzahl der Events in den Daten zum Zeitpunkt T, dann ist $\hat{h}_0(T)$ definiert als

$$\hat{h}_0(T) = \frac{d_T}{\sum_{j \in \Re(T)} e^{x_j \beta}}.$$

In der implementierten Funktion wird auf die Quellen [2], [3] und [4] verwiesen.

2.2.7.2 *DeepSurv*

Das DeepSurv Modell ist eine nichtlineare Erweiterung des Cox PH Modells und wurde 2017 in [54] veröffentlicht. Bei dem Modell handelt es sich um ein Neuronales Netz, dass die geschätzte Funktion $\hat{h}_\theta(x)$ ausgibt. Die Bezeichnung θ sagt lediglich aus, dass die Ausgabe aus dem Netz θ kommt und wurde so in [54] gewählt. Dabei ist $h(t, X) = h_0(t) e^{\hat{h}_\theta(X)}$ und $\hat{h}_\theta(X)$ eine nicht-lineare Funktion. Die Linearkombination $X^T \beta$ wird somit mit der Ausgabe des Netzes $\hat{h}_\theta(X)$ ersetzt. Als Eingabe erhält das Netz die Vektoren der erklärenden Variablen X.[55] Die Parameter der Funktion werden durch die Gewichte des Neuronalen Netzes bestimmt. Das Netz hat mehrere versteckte Schichten und verwendet als Verlustfunktion die negative partielle Likelihood-Funktion. Die Funktion $\hat{h}_\theta(X)$ wird aus einer Linearkombination der letzten versteckten Schicht gebildet. Die Aktivierungsfunktion und Anzahl der Knotenpunkte pro versteckter Schicht kann man beliebig wählen. Dadurch kann man das Netz gut auf einen Datensatz anpassen.

Die implementierte Version von [51] weicht etwas von dem Modell in [54] ab, da die Efron Approximation ([68]) verwendet wird. Dabei wird über alle vorhandenen einzigartigen Zeitpunkte im Datensatz iteriert. Sei n die Anzahl der einzigartigen Zeitpunkte, t_j der j-te solche Zeitpunkt, H_j die Menge aller Indizes der Stichproben, bei denen das Event stattgefunden hat und deren Eventzeit gleich t_j ist, m_j die Anzahl der Indizes in H_j, $i : T_i \geq t_j$ die Menge aller Indizes der Stichproben, deren Eventzeit größer oder gleich t_j ist, x_i der Vektor der erklärenden Variablen der i-ten Stichprobe und $\hat{h}_\theta(x)$

2 https://github.com/cran/survival/blob/master/R/basehaz.R
3 https://personal.utdallas.edu/~pkc022000/6390/SP06/NOTES/survival_week_5.pdf
4 https://stats.stackexchange.com/questions/46532/cox-baseline-hazard

die Funktion, die vom neuronalen Netz ausgegeben wurde, dann ist die partielle Log-Likelihood-Funktion definiert als

$$l(\theta) = \sum_{j=1}^{n} \left(\sum_{i \in H_j} \hat{h}_\theta(x_i) - \sum_{s=0}^{m_j - 1} \log \left(\sum_{i:T_i \geq t_j} e^{\hat{h}_\theta(x_i)} - \frac{s}{m_j} \sum_{i \in H_j} e^{\hat{h}_\theta(x_i)} \right) \right).$$

In der implementierten Version von [51] wird anschließend der Gradient über das PyTorch-Paket[5] berechnet und ein Optimierungs-Algorithmus verwendet. Der Algorithmus bestimmt, wie die Gewichte des Netzes angepasst werden. Zur Auswahl für den Optimierungsalgorithmus stehen AdaDelta ([69]), AdaGrad ([70]), Adam ([65]), AdaMax ([65]), rmsprop ([71]) und SparseAdam[6].

2.2.7.3 *Riskscore*

Der Riskscore, der für eine Stichprobe berechnet wird und von [51] implementiert wurde, ist das Ergebnis der kumulierten Hazard-Funktion über die komplette beobachtete Zeit. Der Riskscore ist kein absoluter Wert, sondern kann sich stark zwischen Modellen unterscheiden. Er dient als Vergleichswert, um das Risiko mehrerer Stichproben miteinander zu vergleichen. Ein höherer Riskscore bedeutet, dass das geschätzte Auftreten des Events früher passiert. Bei der Berechnung wird die Zeitachse in J Stücke geteilt und für jedes Stück die kumulierte Hazard-Funktion berechnet. Dabei ist J die Anzahl der Zeitpunkte, an denen ein Event stattgefunden hat und t_j der j-te solche Zeitpunkt. Das Risiko $r(x)$ für die Stichprobe x ist dann die Summe dieser kumulierten Hazard-Funktionen ([51]) und ist definiert als

$$r(x) = \sum_{j=1}^{J} H(t_j, x).$$

2.2.7.4 *Concordance-Index*

Informationen aus diesem Abschnitt wurden den Quellen [51], [56] und [57] entnommen. Der Concordance-Index (C-Index) ist eine Metrik, die aussagt, wie gut ein Algorithmus die Reihenfolge von vorausgesagten Werten bestimmt. Der C-Index kann Werte von 0 bis 1 annehmen. Ein C-Index gleich 1 weißt auf eine perfekte Voraussage der Reihenfolge und ein Wert von 0,5 auf eine Zufällige. Wenn der Wert 0 ist, heißt das, dass der Algorithmus die umgekehrte Reihenfolge vorausgesagt hat. Für die Berechnung des C-Index des DeepSurv Modells wird der Riskscore für jede Stichprobe eines Datensatzes

5 https://pytorch.org
6 https://pytorch.org/docs/stable/_modules/torch/optim/sparse_adam.html#SparseAdam

berechnet. Anschließend werden die Zeitpunkte T und Riskscores r in der folgenden Formel miteinander verglichen, um den C-Index zu berechnen.

$$\text{C-Index} = \frac{\sum_{i,j} 1_{T_j<T_i} \cdot 1_{r_j>r_i} \cdot \delta_j}{\sum_{i,j} 1_{T_j<T_i} \cdot \delta_j}$$

Dabei ist $1_{T_j<T_i}$ und $1_{r_j>r_i}$ die jeweilige Indikatorfunktion, die den Wert 1 annimmt, wenn die Bedingung $T_j < T_i$ bzw. $r_j > r_i$ erfüllt ist. Ansonsten ist das Ergebnis der Indikatorfunktion 0. δ_j wird 1, wenn ein Event bei der Stichprobe j beobachtet wurde und somit nicht zensiert ist, ansonsten ist der Wert 0. Der C-Index bewertet somit die korrekte Reihenfolge der Zeitpunkte T und der Riskscores r.

2.2.7.5 *Brier-Score*

Informationsquellen für diesen Abschnitt sind [51] und [58]. Der Brier-Score (BS) ist der durchschnittliche quadratische Abstand zwischen einem vorausgesagten und tatsächlichen, binären Wert. Der vorausgesagte Wert ist dabei eine Wahrscheinlichkeit mit einem Wert von 0 bis 1. Ebenso hat der BS eine Wertemenge von 0 bis 1. Dabei ist 0 das bestmögliche Ergebnis und 1 das Schlechteste. Bei der Survival-Analyse kann der Brier-Score $BS(t)$ für einen bestimmten Zeitpunkt t durch folgende Formel bestimmt werden.

$$BS(t) = \frac{1}{N} \sum_{i=1}^{N} (1_{T_i>t} - \hat{S}(t, x_i))^2$$

Dabei ist N die Anzahl aller Stichproben, $\hat{S}(t, x_i)$ die Wahrscheinlichkeit der geschätzten Survival-Funktion $\hat{S}$ für die Stichprobe i zum Zeitpunkt t und $1_{T_i>t}$ die Indikatorfunktion, die den Wert 1 annimmt, wenn das Event für die Stichprobe i zum Zeitpunkt t noch nicht stattgefunden hat, und ansonsten 0 ist. Es ist zu beachten, dass die Formel nur korrekt ist, wenn keine zensierte Daten vorhanden sind. Durch Hinzufügen von zensierten Daten muss die Gleichung wie folgt angepasst werden.

$$BS(t) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\left(0 - \hat{S}(t, x_i)\right)^2 \cdot 1_{T_i \leq t, \delta_i = 1}}{\hat{G}(T_i)} + \frac{\left(1 - \hat{S}(t, x_i)\right)^2 \cdot 1_{T_i > t}}{\hat{G}(t)} \right)$$

Dabei ist $\hat{G}(t)$ die geschätzte Survival-Funktion der zensierten Verteilung, die über die Kaplan-Meier-Methode ([59], [60, Kap. 3.5.5])) berechnet wurde. Im Quellcode des verwendeten PySurvival-Pakets wird zum Thema Brier-Score mit zensierten Daten auf [61] und [62] hingewiesen. Sei T_i der i-te Zeitpunkt, an dem ein Event stattgefunden hat und alle T sortiert, sodass $T_1 < T_2 < \ldots$ gilt. Zudem sei d_i die Anzahl der Stichproben, bei denen zum Zeitpunkt T_i ein Event stattgefunden hat und n_i die Anzahl der Stichproben, bei denen bis zum Zeitpunkt T_i noch kein Event stattgefunden hat, dann

ist der Schätzer der Survival-Funktion über die Kaplan-Meier-Methode definiert als

$$\hat{S}_{KM}(t) = \prod_{T_i \leq t} \frac{n_i - d_i}{n_i}.$$

Um $\hat{G}(t)$ zu erhalten, wird $\hat{S}_{KM}$ für die zensierte Verteilung geschätzt. Dabei wird die Zensur der Daten als Event verwendet.[58] Damit der BS als eine Metrik für Survival-Modelle benutzt werden kann, wird der BS über die komplette beobachtete Zeit integriert. Sei $t_{\max}$ der letzte beobachte Zeitpunkt aller Stichproben, dann ist der Integrated Brier-Score (IBS) definiert als

$$IBS(t_{\max}) = \frac{1}{t_{\max}} \int_0^{t_{\max}} BS(t)dt.$$

Dieser Score sagt aus, wie groß der Unterschied zwischen der geschätzten Wahrscheinlichkeit für ein Event und dem tatsächlichen Event ist.

3

ERKENNUNG VON BIOMARKERN

In diesem Kapitel wird erläutert, mit welchen Mitteln das neuronale Netz trainiert wird, und wie der Output des Netzes verarbeitet wird um die Daten in Survival-Modellen verwenden zu können. Zuerst werden die Einzelheiten des Trainierens des Netzes in Abschnitt 3.1 geklärt. Dies beinhaltet, wie die Bilddaten für das Training bearbeitet wurden und der Prozess, in denen mit den Einstellungen experimentiert wurde, um ein besseres Ergebnis zu bekommen. In Abschnitt 3.2 werden die Ergebnisse des Netzes präsentiert, welches die vielversprechendsten Aussagen macht. Anschließend wird in Abschnitt 3.3 erläutert, wie die Ausgabe des Netzes verarbeitet wird, um diese mit den Metadaten zu kombinieren und für die Analysen in Kapitel 4 vorzubereiten.

3.1 TRAINIEREN DES NEURONALEN NETZES

Für das Mask R-CNN verwenden wir ein Modell mit bereits trainierten Gewichten als Startpunkt, welches auf der GitHub-Seite von Matterport[1] zur Verfügung gestellt wurde, und trainieren dieses mit eigenen Daten weiter. Das Modell kann man ebenso mit dem Befehl `download_trained_weights()` in einer Python-Umgebung herunterladen, wenn man das Mask R-CNN Paket von Matterhorn importiert hat. Das bereitgestellte Modell wurde mit dem MS-COCO Datensatz vortrainiert, um Alltagsgegenständen und Tiere zu erkennen. Durch die hohe Anzahl der Epochen beim Training, spielt dies aber keine große Rolle. Eine Epoche beschreibt einen Durchlauf des Backpropagationalgorithmus, in dem alle Trainingsdaten einmal durchgerechnet und rückpropagiert wurden.

Durch Markieren der Bilddaten, beschrieben in 3.1.1, erhalten wir Daten, mit denen wir das neuronale Netz trainieren können. Diese Daten werden nach dem Zufallsprinzip gemischt und anschließend werden 80% der Daten als Trainingsdaten deklariert und die restlichen 20% den Validierungsdaten. Dadurch dass der Datensatz Markierungen von 600 Bildern beinhaltet, werden den Trainingsdaten 480 Bilddaten zugeteilt und als Validierungsdaten 120. Zusätzlich wurden weitere 50 Bilder ausgesucht, die nicht Teil des Datensatzes sind, und als Testdaten festgesetzt. Bei diesen Testdaten wurde pro Bild festgestellt, welche Biomarker zu sehen sind. Dies wird nach dem Training genutzt, um die Spezifität und die Sensitivität eines Modells zu bestimmen. Diese beiden Werte werden als Metrik unabhängig vom Trainings- und Validierungsdatensatz genutzt. Die Trainingsdaten werden zusätzlich noch augmentiert. Das heißt, dass die Trainingsdaten mit leicht veränderten

1 https://github.com/matterport/Mask_RCNN

Kopien erweitert werden. Bei den Veränderungen handelt es sich um horizontale Spiegelungen, Abschneiden der Ränder und leichte Drehungen des Bildes.

Dadurch dass das Mask R-CNN Paket auf den Paketen Tensorflow und Keras aufbaut, ist es möglich die Berechnungen über eine Grafikkarte laufen zu lassen, statt die CPU zu benutzen. Dies reduziert die Rechenzeit enorm. Es wurde die Grafikkarte NIVIDIA GTX 970 mit 4GB Grafikkarten-Arbeitsspeicher verwendet, welche für eine Trainingsepoche zwischen 150 und 160 Sekunden benötigt. Pro Trainingssession wurden 280 Epochen berechnet und für jede Epoche ein Modell abgespeichert. Zu jeder Epoche wird noch der Fehler des Trainings- und Validierungsdatensatzes bestimmt. Nach dem Training werden die Modelle ausgesucht, die einen niedrigen Validierungsfehler haben. Von diesen ausgesuchten Modellen wird Spezifität und Sensitivität für jede Klassifizierung bestimmt und Durchschnittswerte über alle Klassifizierungen gebildet. Diese Metriken sagen aber nur aus, ob eine Klasse entdeckt wurde und nicht wo. Deswegen wird bei den Modellen mit den höchsten durchschnittlichen Sensitivitätswerten die Ausgabebilder manuell betrachtet, um zu überprüfen, welches Modell geeigneter ist, die Biomarker zu markieren.

3.1.1 *Markieren der Bilddaten*

Das Markieren der Bilder erfolgte mit dem VGG Image Annotator[2]. Dieses Tool erlaubt das Hochladen und Markieren von Bildern. Beim Markieren werden die Umrandungen der Biomarker angegeben. Diese Markierungen können einem Attribut zugeordnet werden, welches verwendet wird, um die Biomarker zu klassifizieren. Anschließend werden alle Markierungen mit der Referenz zum entsprechenden Bild in eine json-Datei exportiert. Das Netz benötigt diese json-Datei zusammen mit den Bilddaten, um ein Modell zu trainieren. In Abbildung 3.1 sind zwei Ausschnitte von Bildern zu sehen, die in dem Tool markiert wurden.

3.1.2 *Mask R-CNN Einstellungen*

In diesem Unterkapitel werden verschiedene Einstellungen des Mask R-CNN vorgestellt, mit denen man diverse Parameter beim Trainieren umstellen kann. Diese haben eine große Auswirkung auf das Modell. Es wird versucht, ein möglichst optimales Netz zu trainieren, das keine falschen Klassifizierungen macht und möglichst viele Drusen findet. Das Experimentieren mit diesen Einstellungen ist sehr zeitintensiv, da das Mask R-CNN Paket fast 50 verschiedene Einstellparameter hat, von denen die meisten Parameter Zahlen sind. Dadurch kann es allein für einen Parameter unendlich viele Möglichkeiten geben, auch wenn diese nur bis zu einem gewissen Wert sinnvolle Ergebnisse produzieren. Aufgrund dieser Tatsachen wird beim Experimen-

2 http://www.robots.ox.ac.uk/~vgg/software/via/via.html

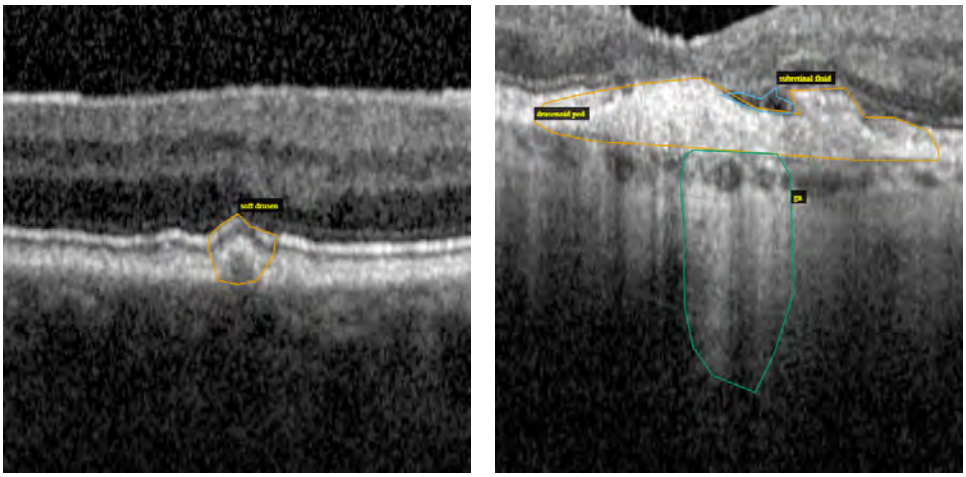

Abbildung 3.1: Ausschnitte von markierten Bildern im VGG Image Annotator

tieren nur ein Wert geändert und der Rest wird auf den Standardeinstellungen gelassen.

In Abschnitt 3.1.2.1 wird mit den zwei verfügbaren Backbone-Architekturen experimentiert. In Abschnitt 3.1.2.2 wird mit zwei Datensätzen trainiert. Der Unterschied zwischen den Datensätzen liegt darin, dass in dem einen Datensatz die Bilder entfernt wurden, in dem keine Biomarker zu sehen sind. Das Training mit den Einstellungen für Abschnitt 3.1.2.2 und 3.1.2.1 lief unabhängig voneinander. Damit ist gemeint, dass für beide Backbone-Architekturen ein Modell mit dem kompletten Datensatz trainiert wird und ein Modell mit dem gefilterten Datensatz. Dadurch ergeben sich vier Modelle, die mit einander verglichen werden. Anschließend wird das passendste Modell ausgesucht und in 3.1.2.3 wird mit dessen Einstellungen die Lernrate erhöht, um eventuell ein besseres Modell zu erhalten. Es wurde noch mit anderen Einstellparameter experimentiert, jedoch führten diese zu keinen Verbesserungen und wurden deshalb weggelassen. In Appendix D sind alle Parameter mit Beispieleinstellungen zu sehen.

3.1.2.1 *Backbone*

Eine mögliche Einstellung des Mask-RCNNs ist die Auswahl der Netzwerkarchitektur, die Features aus den Bildern extrahiert. Die Netzwerkarchitektur die für die Feature Extraktion verwendet wird, wird auch Backbone genannt. Das Mask-RCNN bietet als Backbone standardmäßig zwei Varianten des ResNet an: ResNet-50 und ResNet-101.

Die Zahl bei ResNet steht für die Anzahl der verwendeten Schichten. Aus einer höheren Zahl folgt auch ein tieferes Netz und damit eine bessere Lernfähigkeit des Netzes. Wenn wir das ResNet-101 als Backbone verwenden, sollten also besser trainierte Netze entstehen, da diese tiefer sind und somit eigentlich kleinere Trainingsfehler haben sollten. Wenn man jedoch die Tabellen 3.1 und 3.2 betrachtet, sieht man, dass die durchschnittliche Sensitivität des

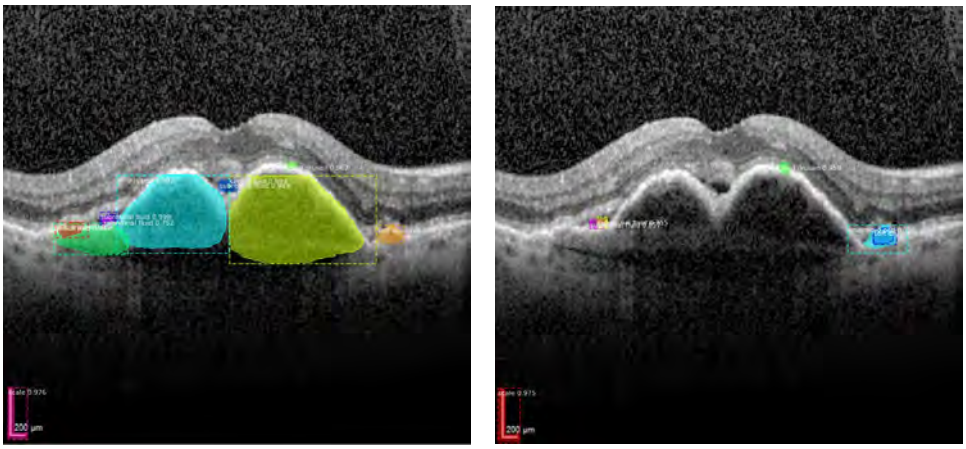

Abbildung 3.2: ResNet-50 (links) und ResNet-101 (rechts) Ausgabe

ResNet-101 um einiges schlechter ist als die Werte vom ResNet-50-Gegenstück. Der Begriff Negativprobe dient zur Differenzierung zwischen den Modellen, die in 3.1.2.2 betrachtet werden und wird dort weiter erläutert. Die Werte, die in den Tabellen zu sehen sind, entstammen jeweils dem ausgesuchten Modell, das am passendsten markiert. In manchen Fällen hatten Modelle zwar etwas bessere Sensitivitätswerte, jedoch waren Positionen oder Größen von den Markierungen falsch.

Besonders fatal bei den Werten ist die Tatsache, dass seröse Pigmentepithelabhebungen gar nicht erkannt wurden. Da diese zu den größeren Klassifizierungen zählen und ein starkes Indiz für fortgeschrittene AMD ist, ist es wichtig diese zu erkennen. In Abbildung 3.2 kann man die Ausgabebilder von Modellen mit beiden Backbone-Architekturen sehen. Beide Modelle erhielten das selbe Eingabebild. Während ResNet-50 alle Vorkommnisse im Bild erkennen konnte, hat das ResNet-101 die Epithelabhebung gar nicht erkannt.

Einen konkreten Grund, warum das ResNet-101 schlechter abschneidet, wurde nicht gefunden. Ein möglicher Grund wäre, dass die Anzahl der Schichten vom ResNet-50 bereits reichen, um die Klassifizierungen zu lernen und die zusätzlichen Schichten vom ResNet-101 zu Overfitting führen. Ein ähnlicher Fall wurde in Kapitel 3.2.2 von [63] beobachtet.

3.1.2.2 *Bilddaten*

Die nächste Einstellung wird nicht direkt als Parameter eingestellt, sondern muss manuell in den Trainingsdaten geändert werden. Es handelt sich dabei um Bilder ohne Biomarker. Wenn man keine Biomarker in einem Bild findet, heißt dies, dass zumindest dieser Abschnitt wahrscheinlich gesund ist. Da solche Bilder häufig vorkommen werden, ist es vermutlich vom Vorteil, dass das Netz auch darauf trainiert wird, keine Biomarker in solch einem Bild zu markieren. Von den 600 Bildern im oben genannten Datensatz enthalten ca. 400 Bilder Biomarker und die anderen 200 keine. Die Bilder ohne Biomarker

	ohne Negativprobe		mit Negativprobe	
	Sensitivität	Spezifität	Sensitivität	Spezifität
Harte Drusen	0,62	0,67	0,62	0,79
Weiche Drusen	0,2	0,96	0,7	0,86
Retikuläre Pseudodrusen	0,12	0,92	0,08	1
Kutikuläre Drusen	0,87	0,8	0,5	0,92
Hyperreflektiver Punkt	0,42	0,93	0,28	0,9
Subretinale Flüssigkeit	0,25	1	0,25	1
Intraretinale Flüssigkeit	0,75	1	0,75	0,93
Drusenartige Epithelabhebung	0,33	0,81	0,33	1
Seröse Epithelabhebung	0	1	0	1
Geographische Atrophie	0,75	0,91	1	0,82
Durchschnitt	0,43	0,89	0,45	0,92

Tabelle 3.1: Sensitivität und Spezifität mit ResNet-101-Architektur

	ohne Negativprobe		mit Negativprobe	
	Sensitivität	Spezifität	Sensitivität	Spezifität
Harte Drusen	0,62	0,97	0,5	0,67
Weiche Drusen	0,7	1	0,65	0,93
Retikuläre Pseudodrusen	0,92	0,84	0,6	0,84
Kutikuläre Drusen	1	0,92	1	0,5
Hyperreflektiver Punkt	0,14	1	0,28	0,95
Subretinale Flüssigkeit	0,5	1	1	1
Intraretinale Flüssigkeit	0,75	0,97	1	0,97
Drusenartige Epithelabhebung	0,66	0,92	0,33	0,89
Seröse Epithelabhebung	1	1	1	1
Geographische Atrophie	0,5	1	1	0,95
Durchschnitt	0,68	0,96	0,73	0,87

Tabelle 3.2: Sensitivität und Spezifität mit ResNet-50-Architektur

nennen wir Negativprobe.

Das Mask R-CNN Paket von Matterhorn filtert Bilder ohne Klassifizierungen aus den Trainingsdaten, da diese sonst zu Fehlern führen. Deswegen fügen wir die Skala, die unten links in jedem Bild zu sehen ist, als eine weitere Klassifizierung hinzu. Diese Skala wird in jeder Negativprobe markiert. Dadurch haben wir nun zwei Datensätze. Einen Datensatz, der nur aus den 400 Bildern mit Biomarkern besteht und einen weiteren Datensatz, der zusätzlich noch die 200 Negativproben enthält. Durch das Hinzufügen der Negativproben sollte der zweite Datensatz bessere Ergebnisse erzielen.

Wenn man sich Tabelle 3.2 betrachtet, sieht man, dass der Unterschied in der durchschnittlichen Sensitivität nicht so groß ist. Bei dem Modell mit Negativproben ist jedoch die durchschnittliche Sensitivität geringer. Wenn man die Werte für die kutikulären und harten Drusen betrachtet, kann man erkennen, dass diese am meisten für den Verlust der Spezifität beigetragen haben. Beim Betrachten der Ausgabebilder der beiden Modelle fällt auf, dass das Modell ohne Negativproben weniger Biomarker markiert. Gleichzeitig erkennt das Modell mit Negativproben zwar fast alle Biomarker, klassifiziert kleinere Biomarker aber oft als kutikuläre Druse, wodurch im Endeffekt die Spezifität dieser Klassifikation reduziert wurde. Am Ende ist es eine Frage, welche Eigenschaften der Modelle für die weitere Aufgaben wünschenswerter sind. Beide Modelle zeigen relativ gute Ergebnisse. Da das Endziel ist, vorherzusagen, ob der Patient sich in der intermediären oder fortgeschrittenen Phase der AMD befindet, fällt die Entscheidung auf das Modell, das mit Negativproben trainiert wurde. Dieses erkennt geographische Atrophien, sub- und intraretinale Flüssigkeiten besser, welche größere Faktoren bei der Einschätzung des AMD-Stadiums sind.

In Abbildung 3.3 kann man die Ausgabebilder betrachten, die gut die Eigenschaften beider Modelle zeigen. In der oberen Abbildung erkennt das Modell mit Negativproben die intraretinale Flüssigkeit, dafür jedoch auch eine kutikuläre Druse in der serösen Epithelabhebung. Im mittleren Ausschnitt erkennt das Modell mit Negativproben alle gezeigten Drusen, auch wenn die zweite Druse von links öfter markiert wurde. Überlappende Flächen einer Klassifizierung werden in Abschnitt 3.3 der Flächenberechnung abgezogen. Im untersten Bild sind leichte Atrophien zu sehen, die vom Modell mit Negativproben erkannt wurden, aber nicht vom anderen Modell.

3.1.2.3 *Lernrate*

Zum Schluss experimentieren wir mit der Lernrate. In [39] wird eine Lernrate von $0,02$ verwendet. In den Anmerkungen des Mask R-CNN Pakets steht jedoch, dass diese Lernrate zu hoch für die implementierte Version ist und die Gewichte dadurch "explodieren". Damit ist gemeint, dass die Werte gegen unendlich steigen. Aus diesem Grund ist die Standartlernrate auf $0,001$

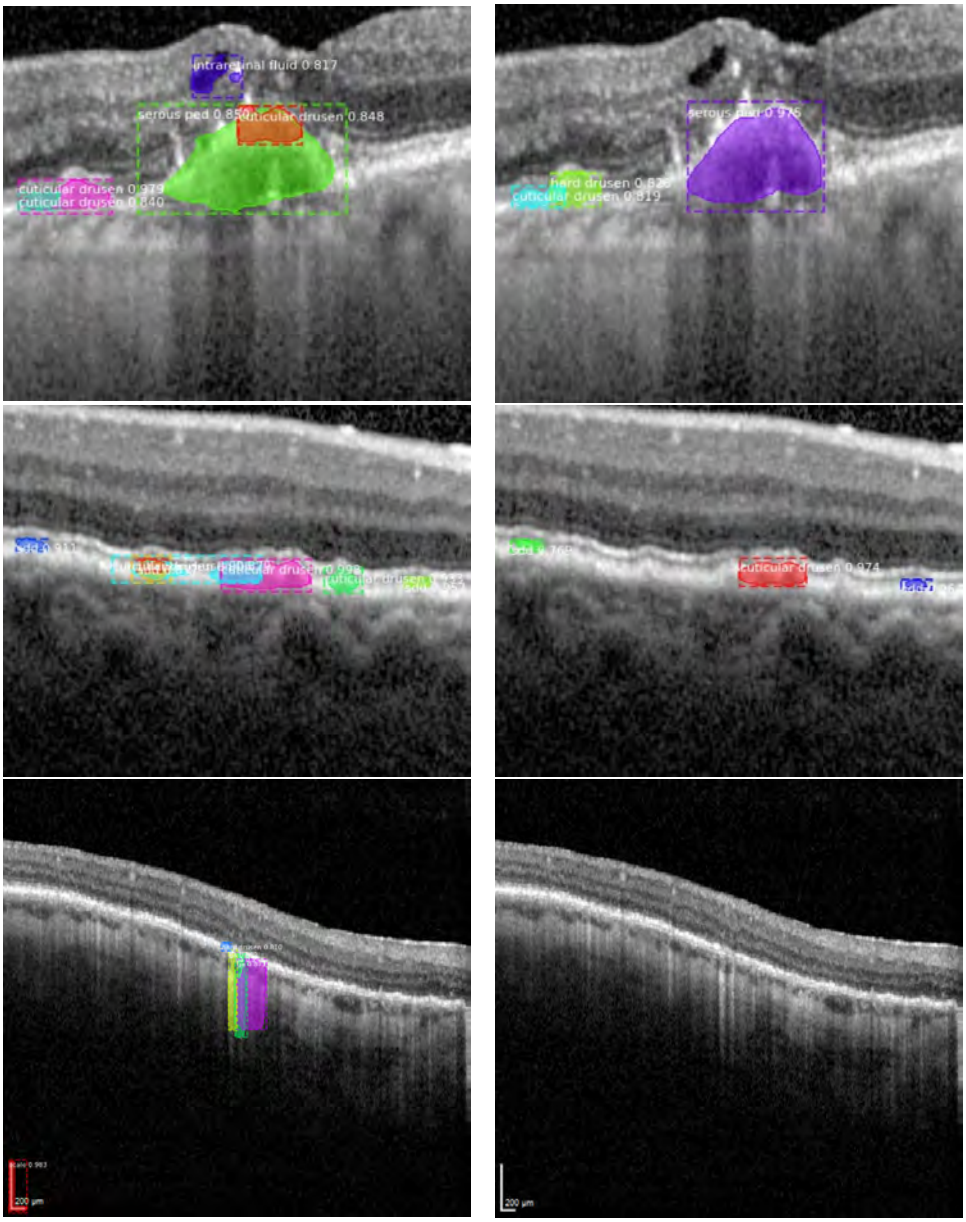

Abbildung 3.3: Ausschnitt aus Ausgaben vom Modell mit Negativprobe (links) und ohne Negativprobe (rechts)

	*LR*2 (Lernrate=0,002)		*LR*3 (Lernrate=0,003)	
	Sensitivität	Spezifität	Sensitivität	Spezifität
Harte Drusen	0,56	0,88	0,75	0,85
Weiche Drusen	0,9	0,8	0,8	0,8
Retikuläre Pseudodrusen	0,6	0,84	0,84	0,8
Kutikuläre Drusen	0,87	0,73	1	0,66
Hyperreflektiver Punkt	0,42	0,97	0,42	0,97
Subretinale Flüssigkeit	0,75	1	0,25	1
Intraretinale Flüssigkeit	1	0,93	1	0,95
Drusenartige Epithelabhebung	0,41	0,92	0,66	0,89
Seröse Epithelabhebung	1	1	0,57	1
Geographische Atrophie	1	1	1	0,97
Durchschnitt	0,75	0,90	0,73	0,89

Tabelle 3.3: Sensitivität und Spezifität mit veränderter Lernrate

gestellt. In diesem Abschnitt werden die Modelle vorgestellt, bei denen die Lernrate auf 0,002 und 0,003 erhöht wurde. Das Modell mit der Lernrate von 0,002 nennen wir *LR*2 und das mit 0,003 *LR*3. Das ausgesuchte Modell mit Negativproben aus dem vorherigen Abschnitt 3.1.2.2 nennen wir *LR*1. Es wurde auch eine Lernrate von 0,0005 und 0,005 ausprobiert. Jedoch führte dies zu einer wesentlichen Verschlechterung der Modelle.

Die durchschnittliche Spezifität und Sensitivität hat sich nicht stark von dem Modell mit einer Lernrate von 0,001 geändert. Die Erkennung von harten und weichen Drusen, hyperreflektive Punkten und drusenartiger Pigmentepithelabhebungen hat sich bei *LR*2 und *LR*3 verbessert. Dafür ist die Performance bei subretinaler Flüssigkeit schlechter. Beim direkten Vergleich der Ausgabebilder der drei Modelle fällt auf, dass das Modell *LR*3 ab und zu Biomarker übersieht, die die anderen beiden Modelle erkannt hat. Das Modell *LR*2 ähnelt von der abgedeckten Fläche dem Modell *LR*1. Der Hauptunterschied liegt darin, dass *LR*2 Drusen nicht so oft als kutikuläre Druse deklariert. Dies ist auch an der besseren Spezifität bei den kutikulären Drusen zu sehen. Auch wenn beide Modelle noch manche Drusen überlappend deklarieren, hat sich dies in Modell *LR*2 etwas verbessert. Aus diesem Grund entscheiden wir uns für das Modell *LR*2.

3.2 ERGEBNISSE

In diesem Abschnitt werden sechs Ausschnitte von Ausgabebildern des Modells *LR*2 präsentiert, um zu zeigen, wie das Modell markiert. Die Ausschnitte sind in Abbildung 3.4 zu sehen. Im ersten Ausschnitt wurden fast alle intraretinalen Flüssigkeiten markiert. Die große nicht-markierte Stelle lässt sich dadurch erklären, dass intraretinale Flüssigkeiten für gewöhnlich nicht

so groß werden und wahrscheinlich auch nicht in den Trainingsdaten in dieser Größe vertreten waren. Die unteren Drusen wurden als weiche und kutikuläre Druse erkannt. Im zweiten Ausschnitt wurde die untere Atrophie gut abgedeckt. Auch die seröse Epithelabhebung wurde erkannt. Die Flüssigkeit rechts davon jedoch nicht. Auch hier ist eine solch große Flüssigkeitsstelle eher unüblich. Im dritten Abschnitt wurden die beiden kutikulären Drusen erkannt und mehrfach markiert. In der vierten Abbildung wurden die zwei kleineren Drusen zwischen den Markierungen nicht erkannt. Im fünften Abschnitt wurde zwar die intraretinale Flüssigkeit erkannt, aber nicht die kleine Druse mittig im Bild. In der letzten Abbildung wurde die Flüssigkeit und die Epithelabhebung erkannt. Die Abhebung wurde jedoch sowohl als drusenartig als auch serös deklariert. Dieses Modell erkennt nicht alle gezeigten Biomarker, jedoch wurden auch keine Stellen markiert, an denen kein Biomarker zu sehen ist. Eine Schwachstelle ist das Markieren mehrerer Arten von Biomarkern für ein erkanntes Objekt.

3.3 WEITERVERARBEITUNG DES OUTPUTS

In diesem Abschnitt wird erklärt, wie die Ausgaben des Mask R-CNN transformiert werden, um Tabellen zu erstellen, die mit der Survival-Analyse in Kapitel 4 kompatibel sind. Beim Erstellen der Masken werden auch die Koordinaten der Eckpunkte der segmentierten Objekte erstellt. Diese Koordinaten können wir nutzen, um daraus Polygone mit dem shapely-Paket zu erstellen. Das shapely-Paket erlaubt das Erzeugen von diversen geometrischen Objekten und stellt Funktionen für diese zur Verfügung. Mit Hilfe dieser Polygone können wir für jedes Objekt die Fläche, Länge und Breite berechnen. Durch Verwenden der Skala, die in jedem Bild zu sehen ist, lässt sich die Fläche auch in μm^2 berechnen. Die Polygon-Objekte erlauben auch das Berechnen der Schnittflächen mit anderen Polygonen. Dies erlaubt das Erstellen einer Flächenangabe für die komplette markierte Fläche eines Bildes. Dies wird auch pro Klassifizierung verwendet, um für jede Klassifizierung eine Flächenangabe pro Bild zu erhalten. Gleichzeitig wird auch gezählt wie viele Objekte einer Klasse gefunden wurden.

Zur Erstellung der notwendigen Tabelle für Kapitel 4 müssen wir zunächst eine Zwischentabelle erstellen. Dafür schicken wir jedes Bild von jedem Patienten durch das Netz und berechnen Anzahl, Fläche, maximale Länge und Breite für alle Biomarker. Diese Werte speichern wir in einer Tabelle wie in Tabelle 3.4 ab. Mit date ist dabei der Untersuchungstermin gemeint und mit slice der verwendete Bildausschnitt des OCT. An einem Termin werden 25 Bilder erstellt. Dadurch dass wir zehn Biomarker haben, werden auch zehn Reihen für jedes Bild erstellt. Insgesamt erstellen wir für alle Patienten 723340 Reihen in dieser Zwischentabelle.

Diese Zwischentabelle können wir aggregieren und mit den Metadaten aus Abschnitt 2.2.1.2 kombinieren, um die Tabelle für die Survival-Analyse

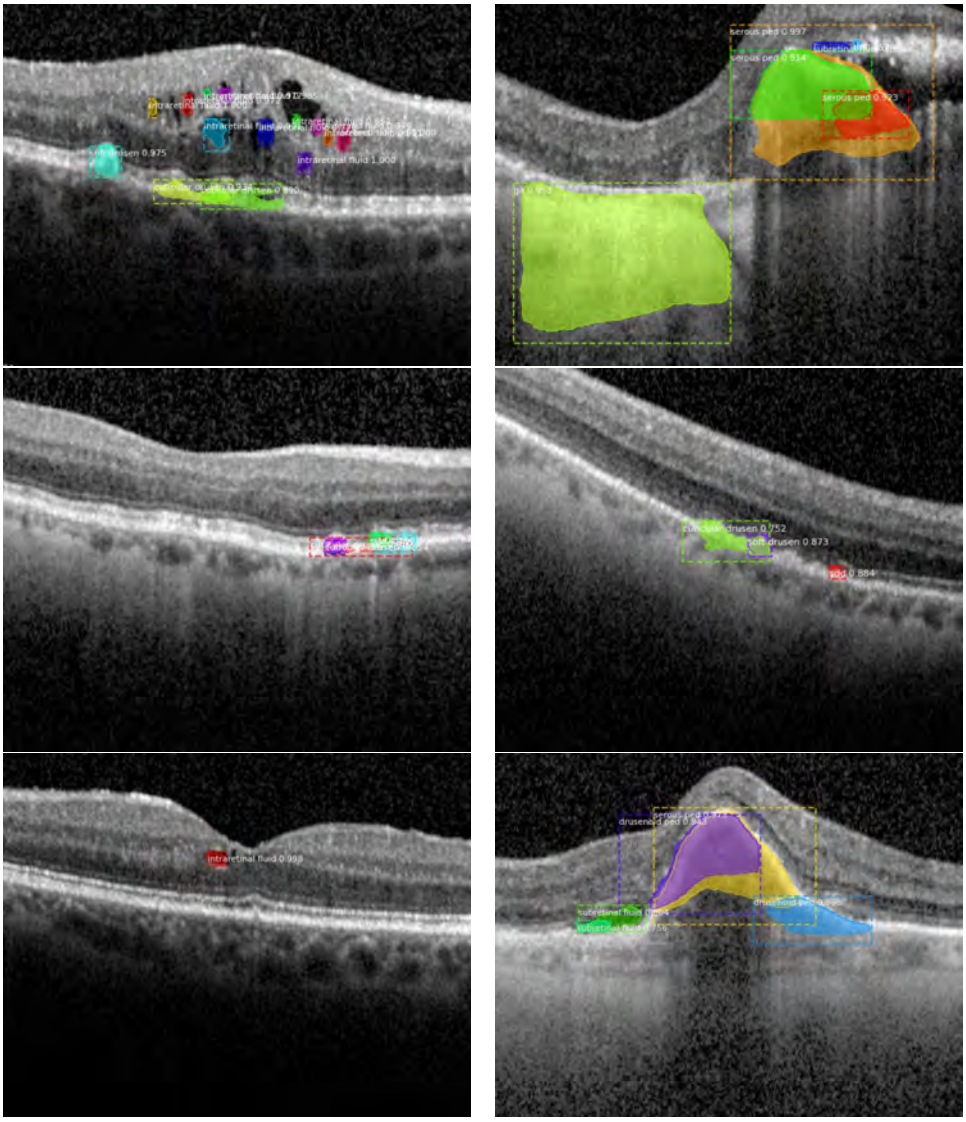

Abbildung 3.4: Ausschnitte aus Ausgabebildern von Modell *LR*2

id	eye	date	slice	biomarker	amount	area	max length	max width
5	OD	30112017	24	soft drusen	0	0	0	0
5	OD	30112017	24	hard drusen	1	8303	30	247

Tabelle 3.4: Ausschnitt aus Zwischentabelle

zu erstellen. Die endgültige Tabelle benötigt drei wichtige Eigenschaften. Zum einen muss jede Reihe einen eindeutigen Klassifikator haben, nach dem die Survival-Modelle trainiert werden sollen. Dafür nutzen wir die Angabe aus den Metadaten, ob der Patient im intermediären oder fortgeschrittenen AMD-Stadium ist. Zum Anderen benötigen wir eine Zeitangabe. Für diese gibt es zwei mögliche Kandidaten. Wir können entweder das Alter des Patienten zum Zeitpunkt der Untersuchung verwenden oder die Anzahl der Monate seit seinem ersten Untersuchungstermin. Die letzte Bedingung ist, dass alle Tabellenwerte numerisch sind. Das heißt, dass alle kategorischen Daten auf mehrere Spalten aufgeteilt und binär mit 1 und 0 dargestellt werden müssen. Dieser Schritt bleibt uns erspart, da wir in der Endtabelle keine kategorischen Daten haben werden. Da wir nur anhand der Daten eines Termins eine Aussage über die AMD-Stufe treffen wollen, müssen wir auch alle Daten eines Termins zusammenfassen. Da zu jedem Termin nicht immer das OCT von beiden Augen aufgenommen wurde, behalten wir in der Endtabelle die Trennung für beide Augen bei. Damit haben wir von einem Patienten zu einem Termin maximal zwei Reihen.

Die einfachste Zusammenfassung der Daten wäre die Daten aus der Zwischentabelle zu nehmen und mit diesen Daten die Spalten zu erweitern. Da wir jedoch pro Auge 25 Bilder und 10 Klassifizierungen haben, erweitern wir die Tabelle pro berechneten Wert mit 250 Spalten. Mit den vier berechneten Werten (Anzahl, Fläche, Breite, Länge) aus der Zwischentabelle wären das insgesamt über 1000 Spalten, in denen viele Werte einfach 0 sind, da nicht in jedem Bildausschnitt immer jede Klassifizierung gefunden wird. Da wir Daten zu 285 Terminen haben, wäre die Spaltenanzahl auch weitaus größer als die Reihenanzahl. Dies ist aufgrund des Fluches der Dimensionalität (erstmals in [64] erwähnt) keine gute Grundlage, um ein statistisches Modell zu trainieren. Aus diesem Grund müssen wir die Daten weiterhin zusammenfassen und in Kauf nehmen, dass Informationen verloren gehen.

Eine mögliche Zusammenfassung wäre das Aggregieren über die Klassifizierungen. Da wir dabei jedoch Informationen über die Art des Biomarkers verlieren, ist dies keine gute Idee. Eine bessere Alternative bietet das Aggregieren über die Bildausschnitte. Dadurch verlieren wir die Information, in welchem Abschnitt des Auges die Biomarker erkannt wurden. Das einzige lokale Kriterium, das in Tabelle 2.1 aus Abschnitt 2.1.1 genannt wird, ist, ob eine geographische Atrophie zentral beobachtet wurde oder nicht. Mit zentral ist gemeint, dass sich die Atrophie in einem 500μm Radius um das foveale Zentrum befindet. Da wir dies jedoch nicht automatisiert feststellen können, wissen wir nicht, ob das Kriterium erfüllt ist oder nicht. Aus diesem Grund ist das Verlieren der lokalen Information nicht gravierend. Durch das Aggregieren über die Bildausschnitte fügen wir der Tabelle 40 Spalten hinzu. Als Aggregation wird für die Anzahl und Fläche die Summe verwendet. Bei der Länge und Breite wird das Maximum benutzt.

4

RISKSCORE

In diesem Kapitel werden die verwendeten Daten in 4.1 untersucht, die Modelle der Survival-Analyse in 4.2 vorgestellt und anschließend die Ergebnisse in 4.3 betrachtet.

4.1 UNTERSUCHUNG DER DATEN

Zuerst werden die Korrelationen zwischen den Spalten über die Korrelationsmatrix betrachtet. Dabei fällt auf, dass die Korrelationen zwischen den Messungen desselben Biomarkers sehr hoch sind. In Abbildung 4.1 ist die Korrelationsmatrix der Messungen von weichen Drusen zu sehen. Diese hohe Korrelation zwischen der Anzahl und Fläche und der Breite und Länge eines Biomarkers sind bei allen Biomarkern zu erkennen. Aus diesem Grund wird für jeden Biomarker jeweils zwischen Anzahl und Fläche entschieden und eine der beiden Spalten entfernt. Dasselbe gilt für die Länge und Breite. Es werden die Spalten entfernt, die stärker mit allen anderen Spalten korrelieren.

In Abbildung 4.1 wird zum Beispiel die Fläche und Breite entfernt, da sie neben der hohen Korrelation mit der Anzahl bzw. Länge auch stark miteinander korrelieren. Es wurde auch mit anderen Aufstellungen der Spalten experimentiert, wie zum Beispiel Flächen und Längen zu entfernen oder alle Spalten beizubehalten. Diese Methode führte jedoch zu den Modellen mit den durchschnittlich besten Metriken.

Beim Aufteilen des Datensatzes in Trainings- und Testdaten wird auch darauf geachtet, dass die geteilten Datensätze dasselbe Verhältnis zwischen fortgeschrittener und intermediärer AMD haben, verglichen mit dem ursprünglichen Datensatz. In der verwendeten Funktion `train_test_split()` des sklearn-Pakets wird dies mit der stratify-Option eingestellt. Dem Testdatensatz werden 30% des Datensatzes hinzugefügt. Bei 285 Reihen sind dies 86, wovon 16 als späte AMD eingestuft wurden. Die restlichen 199 Reihen im Trainingsdatensatz haben 36 Reihen mit später AMD.

Vor der Survival-Analyse wird der Datensatz mit einem einfachen Klassifikator analysiert, um zu ermitteln, ob es möglich ist, mit den gegebenen Daten den AMD-Status zu bestimmen. Als Klassifikator verwenden wir die logistische Regression. Zusätzlich führen wir mit der LASSO-Methode einen Strafterm für die Regression ein. Durch den Strafterm werden Modelle bestraft, die eine höhere Anzahl an Spalten verwenden. Somit werden bei der

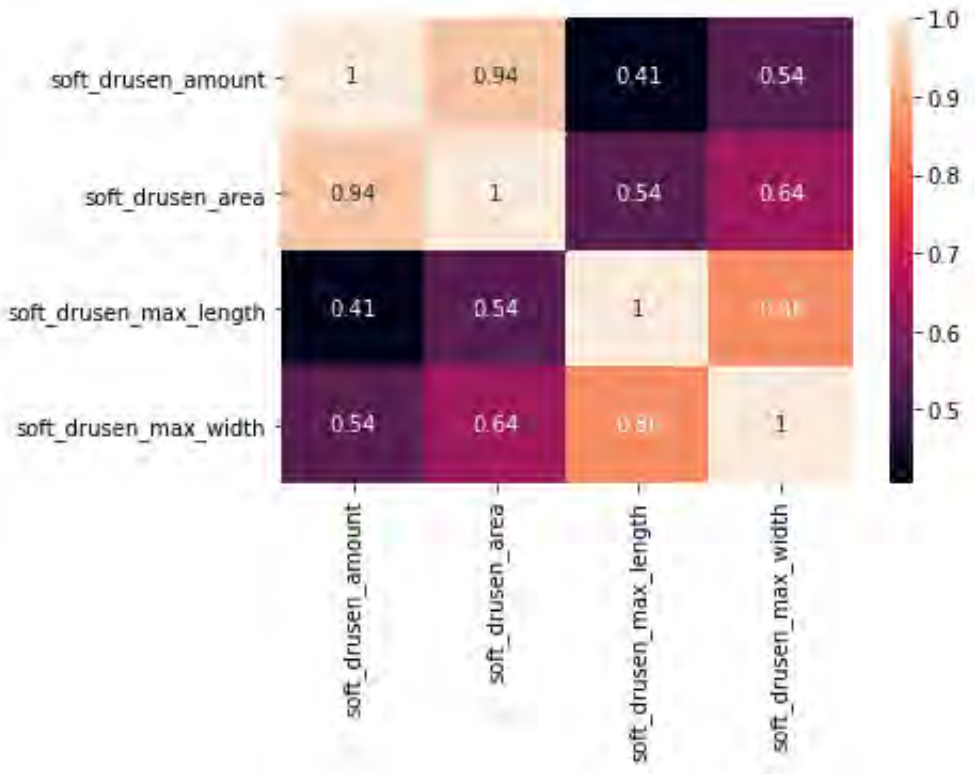

Abbildung 4.1: Ausschnitt aus der Korrelationsmatrix

	Minimum	Durchschnitt	Median	Maximum
Sensitivität	0,43	0,71	0,75	0,87
Spezifität	0,82	0,93	0,93	0,97

Tabelle 4.1: Metriken der logistischen Regression

Regression Spalten entfernt, die nur eine sehr kleine Auswirkung auf das Modell haben. Es wurden zehn Modelle trainiert und Metriken für diese berechnet. Jedes Modell benutzte eine andere zufällige Zusammensetzung von Trainings- und Testdaten. Einzig das Verhältnis zwischen späten und intermediären AMD-Stufen war identisch. In Tabelle 4.1 sind Minimum, Durchschnitt, Median und Maximum von den Metriken dieser Modelle zu sehen.

Während die Spezifität bei jedem Modell sehr hoch ist, schwankt die Sensitivität sehr stark. Die Modelle können somit mit einer hohen Wahrscheinlichkeit korrekt klassifizieren, dass eine Stichprobe im intermediären Stadium ist. Jedoch zeigt die schwankende Sensitivität, dass eine verlässliche Einschätzung, ob eine Stichprobe im späten AMD-Stadium ist, mit diesem Datensatz vermutlich schwierig wird.

Eine Überlegung wäre auch gewesen, eine Scorecard zu erstellen. Eine solche Scorecard wird zum Beispiel bei Kreditanträgen benutzt, um das Risiko anhand einer Punktzahl zu bestimmen. Die Scorecard würde aber lediglich

bewerten, ob ein Patient derzeit in der späten AMD ist. Da das Ziel jedoch ein Score ist, der das Risiko misst, dass ein Patient in die späte AMD-Stufe wechselt, widmen wir uns der Survival-Analyse.

4.2 SURVIVAL-ANALYSE

Als Modell für die Survival-Analyse wurde das DeepSurv-Modell gewählt. Bei diesem Modell kann man neben den Parametern des Modells auch die versteckten Schichten bestimmen. Für jede Schicht wählt man die Aktivierungsfunktion und die Anzahl der Knoten. Um ein passendes Modell zu finden, wurde zuerst mit der Struktur des Netzwerks experimentiert. Die Parameter des Modells blieben dabei auf den Default-Einstellungen.

Es wurden alle vorhandenen Aktivierungsfunktionen durchprobiert und für jede Aktivierungsfunktion zehn Modelle erstellt und der durchschnittliche C-Index berechnet. Für jedes Modell wurde der Datensatz neu gemischt und in Trainings- und Testdatensatz wie bei der logistischen Regression aufgeteilt. Die Schichten mit den höchsten durchschnittlichen C-Index wurden ausgesucht und an diesen wurde eine zweite Schicht hinzugefügt und alle Aktivierungsfunktionen erneut durchprobiert. Dieser Prozess wurde wiederholt bis das Netzwerk fünf Schichten hatte. Die Anzahl der Knoten pro Schicht wurde während des Experimentieren auf 50 Knoten gesetzt. Nach Bestimmung des besten Modells wurde die Anzahl anschließend nach Durchprobieren auf 150 Knoten pro Schicht festgesetzt.

Nach der fünften Schicht gab es keine sichtbare Verbesserung des durchschnittlichen C-Index. Am Ende hatte das Modell Schichten mit den folgenden Aktivierungsfunktionen in dieser Reihenfolge: LogLog, Softmax, Sigmoid, Tanh, Softplus. Alle vorhandenen Aktivierungsfunktionen sind in [1] aufgelistet.

Die Parameter des Modells wurden dann anschließend mit der GridSearchCV-Klasse des sklearn-Pakets optimiert. Diese Klasse durchsucht alle möglichen Kombinationen der Parameter, die dieser Klasse übergeben werden. Dieser Klasse wurden die Parameter für den Initialisierungs- und Optimierungsalgorithmus und die Lernrate übergeben. Die Initialisierungsmethode entscheidet, wie die Gewichte zu Beginn bestimmt werden. Der Optimierungsalgorithmus bestimmt, wie die Gewichte beim Trainieren angepasst werden sollen. Alle möglichen Parameter werden bei der fit-Funktion auf [2] genannt. Eine Lernrate von $0,0001$ zusammen mit dem Adam-Algorithmus ([65]) für die Optimierung und dem Glorot-Normal-Algorithmus ([66]) zur Initialisierung haben sich als optimale Parameter herausgestellt.

1 https://square.github.io/pysurvival/miscellaneous/activation_functions.html
2 https://square.github.io/pysurvival/models/nonlinear_coxph.html

Mit den genannten Einstellungen erstellen wir nun zwei Modelle. Der Unterschied zwischen den Modellen liegt in der verwendeten Zeitachse. Bei einem Modell wird die Anzahl der Monate seit dem ersten Termin verwendet. Jeder Zeitangabe wird ein Monat hinzugefügt, damit die ersten Termine nicht auf 0 liegen. Bei dem anderen Modell wird das jeweilige Alter an dem Termin in Jahren verwendet. Um die Zeitachse zu verkleinern, wird dem Alter 53 Jahre abgezogen, da 54 das kleinste Alter ist, an dem ein Patient einen Termin hatte. Das erste Modell nennen wir M_{Termin} und das Zweite M_{Alter}.

Modell M_{Termin} hat einen durchschnittlichen C-Index von $0,87$ und einen IBS von $0,09$. Bei M_{Alter} liegt der C-Index bei $0,83$ und der IBS bei $0,05$. Dies sind beides sehr gute Metriken für ein Survival-Modell.

Das Problem bei diesen Modellen ist, dass mehrere Termine von Patienten im Datensatz enthalten sind. In der Survival-Analyse ist es jedoch üblich, pro Patient nur eine Stichprobe zu haben, damit eine Unabhängigkeit zwischen den Stichproben gewährt ist. Da jedoch nicht an jedem Termin beide Augen begutachtet wurden, können wir nicht für jeden Patienten die Informationen von beiden Augen zusammenfassen. Aus diesem Grund behandeln wir jedes Auge unabhängig vom anderen Auge. Somit behalten wir nur noch einen Termin pro Auge bei und entfernen den Rest. Bei Augen ohne Wechsel wird der letzte Termin behalten und bei Augen mit Wechsel zur späten AMD wird der Termin beibehalten, an dem der Wechsel wahrgenommen wurde.

Durch die Einhaltung dieser Unabhängigkeit verringern wir die Reihen des Datensatzes von 285 auf 100 Reihen. Davon sind 17 als späte AMD klassifiziert. Im jeweiligen Trainingsdatensatz sind dadurch 12 von 70 Reihen in der späten AMD und im Testdatensatz 5 von 30. Dadurch ändert sich bei Modell M_{Termin} der C-Index von $0,87$ auf $0,81$ und der IBS von $0,09$ auf $0,14$. Bei M_{Alter} ändert der C-Index sich von $0,83$ auf $0,86$ und der IBS von $0,05$ auf $0,06$. Die Modelle mit den unabhängigen Datensatz werden MU genannt.

4.3 ERGEBNISSE

Obwohl die Verringerung der Daten keine drastischen Änderungen bei den Metriken zeigt, sorgt die geringere Datenmenge dafür, dass die Aussagekraft des Modells darunter leidet. Dies erkennt man, wenn man die kumulierte Verteilungsfunktion eines Patienten betrachtet. In Abbildung 4.2 ist die kumulierte Verteilungsfunktion für Modell M_{Termin} und MU_{Termin} von fünf Terminen desselben Patienten zu sehen. Die x-Achse in dieser Abbildung zeigt die Anzahl der Monate seit dem ersten Termin und die y-Achse die geschätzte Wahrscheinlichkeit, dass der Wechsel stattgefunden hat. Wenn die Kurve also bei $x = 50$ den y-Wert $0,6$ hat, dann besteht eine 60%-ige Wahrscheinlichkeit, dass der Wechsel zur späten AMD innerhalb der nächsten 50 Monate stattfinden wird. In der Legende ist die Anzahl der Monate seit dem

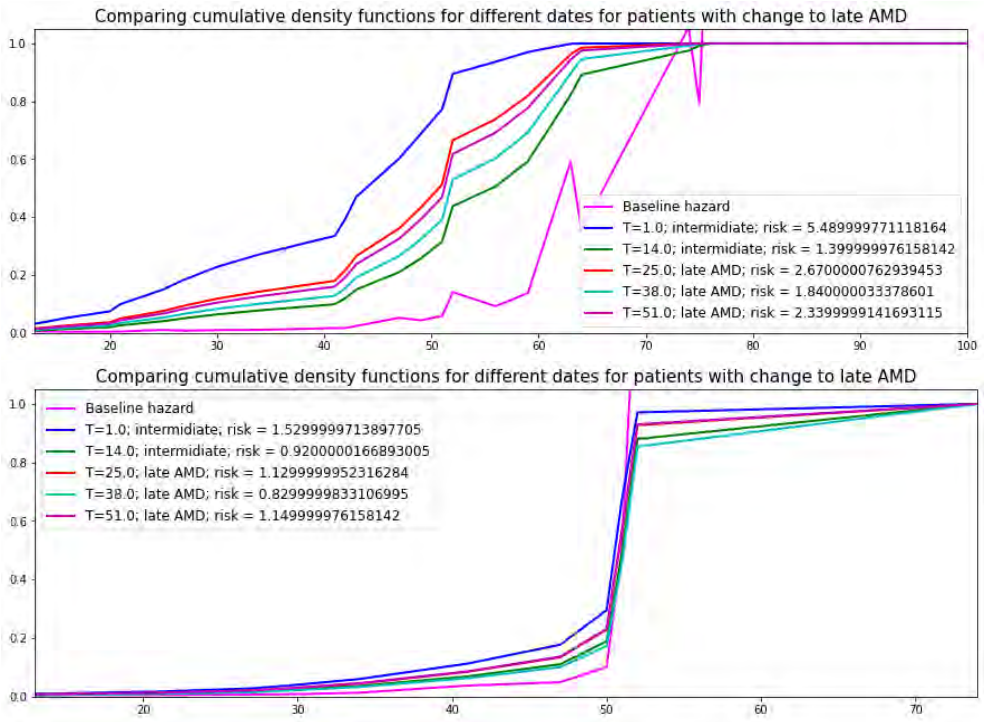

Abbildung 4.2: kumulierte Verteilungsfunktion eines Patienten für Modell M_{Termin} (oben) und MU_{Termin} (unten)

ersten Termin plus 1 als T, die tatsächliche Klassifizierung des Auges zum Termin und der berechnete Riskscore für den Termin aufgelistet. Zusätzlich ist noch die geschätzte Grundgefahr als pinke Kurve eingeblendet. Wenn man bei beiden Grafiken die Grundgefahr mit den jeweils anderen Kurven vergleicht, sieht man, dass die Kurven für Modell MU_{Termin} einen ähnlichen Verlauf zur Grundgefahr haben. Es ist also zu sehen, dass die geringere Menge an Trainingsdaten dafür sorgt, dass das Modell nicht weit von der Grundgefahr abweicht. Das Problem bei Modell MU_{Termin} ist, dass die Riskscores von neuen Patienten auch sehr nah an der Grundgefahr sein werden und damit nicht unbedingt ein guter Vergleichswert sind.

Wir betrachten nun die kumulierten Verteilungsfunktionen mehrerer Patienten für Modell M_{Termin} in Abbildung 4.3. Jede Grafik enthält alle vorhandenen Termine eines Patienten genau wie in Abbildung 4.2, jedoch ohne Grundgefahr. Bei den oberen beiden Grafiken in Abbildung 4.3 wurden die Daten von Patienten mit Wechsel zur späten AMD und in der unteren Grafik die Daten eines Patienten ohne Wechsel zur späten AMD verwendet. Unter der Annahme, dass die Termine unabhängig voneinander wären, wäre die gewünschte Beobachtung, dass spätere Termine beim selben Patienten auch einen höheren Riskscore haben. Eine alternative gewünschte Beobachtung wäre, wenn der Riskscore kurz vor dem Wechsel am Höchsten ist und die Termine nach dem Wechsel ein geringeres Risiko haben als die vor dem Wechsel. In diesem Abschnitt wurden nur die Ergebnisse für die Modelle M_{Termin} und MU_{Termin} betrachtet, da die Erkenntnisse identisch sind. Die

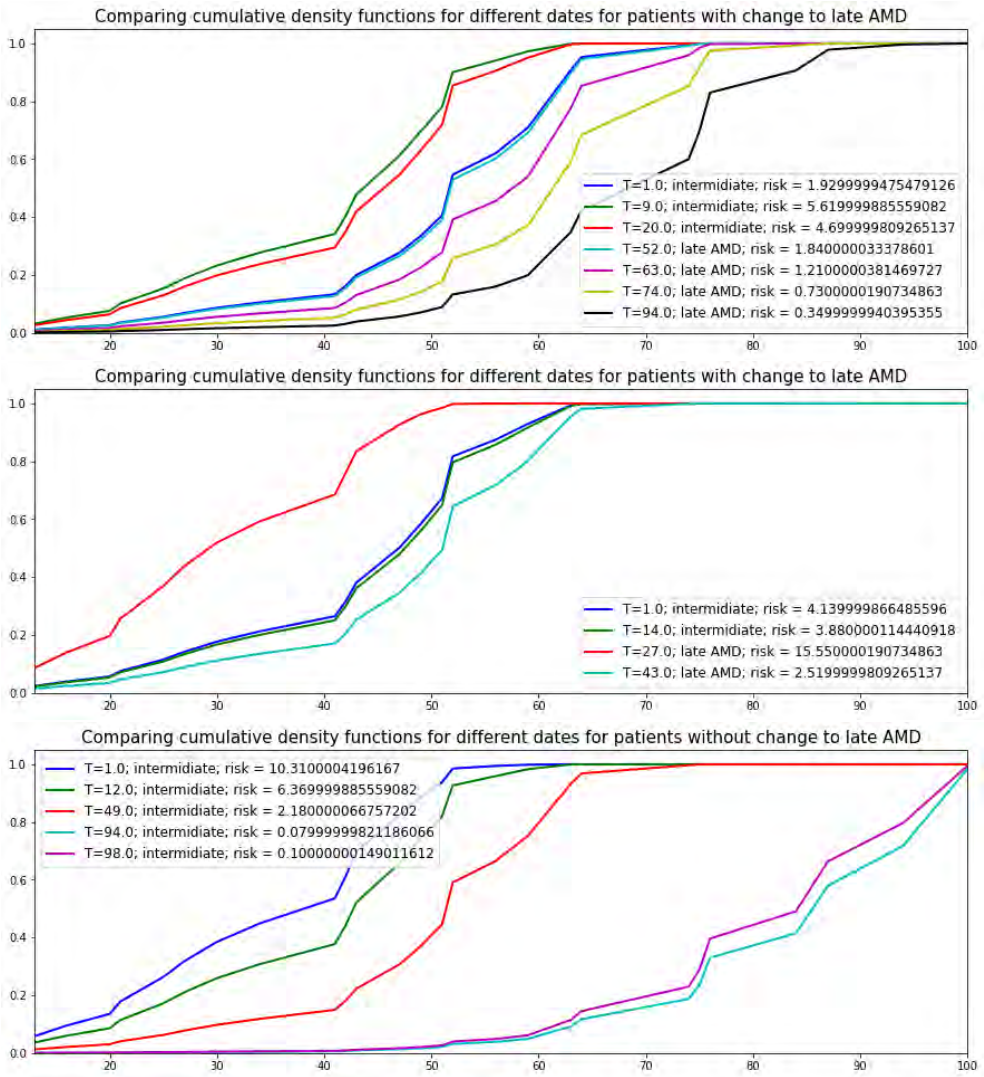

Abbildung 4.3: kumulierte Verteilungsfunktion mehrerer Patienten für Modell M_{Termin}

Grafiken der Modelle M_{Alter} und MU_{Alter} für dieselben Patienten sind in Appendix E zu finden.

In Abbildung 4.4 sieht man für dieselben Patienten wie in Abbildung 4.3 die kumulierte Verteilungsfunktion von Modell MU_{Termin}. Auch bei diesem Modell kann man nicht die Beobachtung machen, dass Leute im späten AMD-Stadium einen höheren Riskscore besitzen. Bei diesem Modell wird für viele der Termine vorausgesagt, dass der Wechsel zur späten AMD nach 50 Monaten stattfinden wird. Diese Aussage ist nur auf den ersten Termin der obersten Grafik passend, da bei diesem Patienten der Wechsel zur späten AMD bei $T = 52$ festgestellt wurde. Es werden keine der gewünschten Beobachtungen wahrgenommen. Mit diesen Ergebnissen kann man davon ausgehen, dass diese Modelle keine aussagekräftigen Schätzungen berechnen.

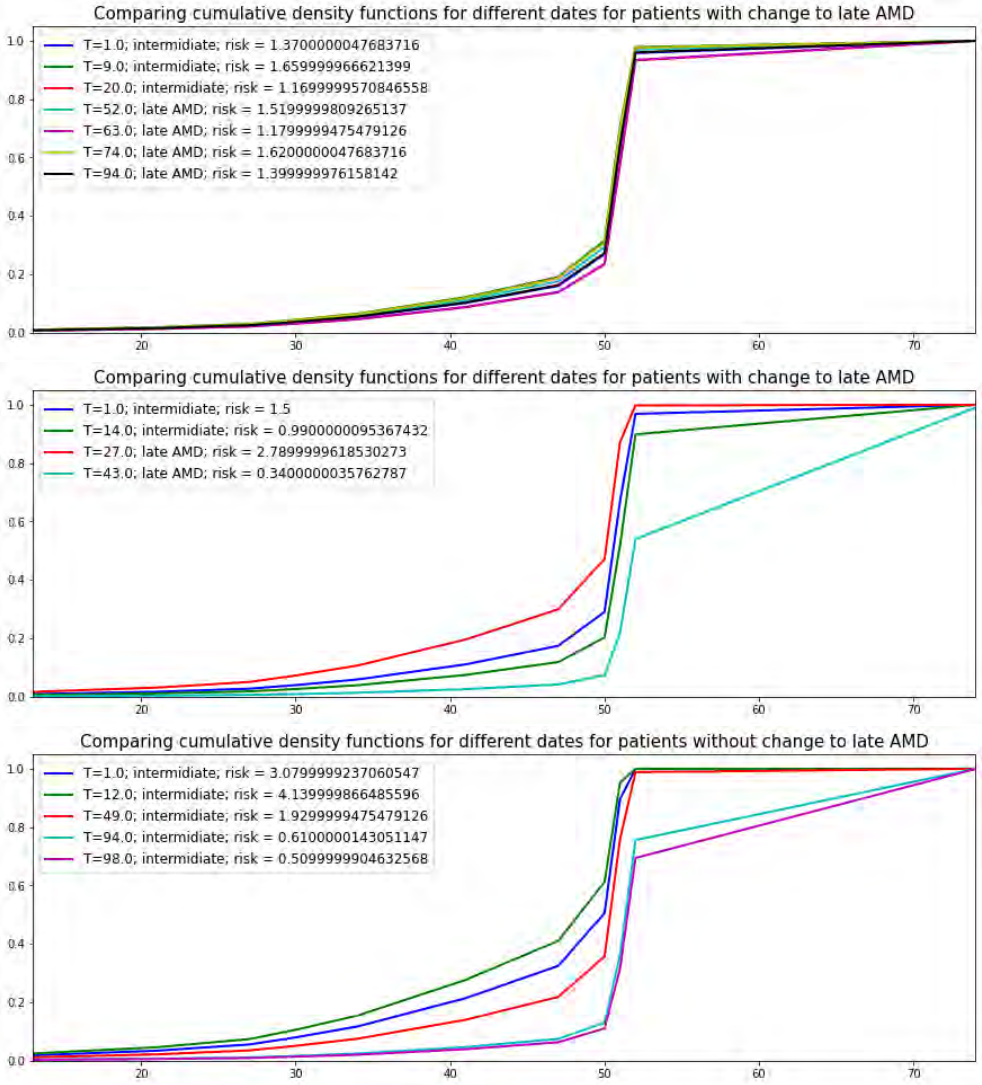

Abbildung 4.4: kumulierte Verteilungsfunktion mehrerer Patienten für Modell MU_{Termin}

5 FAZIT UND AUSBLICK

In diesem Kapitel werden im Fazit die Ergebnisse und Erkenntnisse zusammengefasst. Im Ausblick werden anschließend Vorschläge genannt, die in Folgearbeiten verwendet werden könnten.

5.1 FAZIT

Diese Arbeit hatte zwei Ziele: Das automatisierte Erkennen von Biomarkern, die in OCT-Aufnahmen zu sehen sind und die Erstellung eines Riskscores, der das Risiko bestimmt, wie schnell die AMD fortschreitet.

Das Erkennen der Biomarker erfolgte durch das Mask R-CNN. Aus den OCT-Aufnahmen von AMD-Patienten wurden 600 ausgesucht. Anschließend wurden in diesen Bildern die Biomarker selbständig in dem VGG Image Annotator Tool markiert. Es wurde ein Mask R-CNN Modell erstellt, das mit den Bilddaten trainiert und anhand diverser Parameter optimiert wurde. Die vorgestellten Ergebnisse zeigten, dass das Modell viele der Biomarker erkennen konnte, jedoch nicht immer die richtige Klassifikation gefunden wurde. Mit diesem Modell wurden bei den Bilddaten aller Patienten die Biomarker segmentiert und gemessen. Die Messungen wurden anschließend in einer Tabelle zusammengefasst.

Die Erstellung des Riskscores wurde durch die Survival-Analyse ermöglicht. Es wurden zuerst die Daten untersucht, um festzustellen, ob man mit den Daten das derzeitige AMD-Stadium feststellen kann. Die schwankende Sensitivität der Modelle hatte uns verraten, dass das Einschätzen, ob ein Patient sich in der späten AMD befindet, mit einem einfachen logistischen Regressionsmodell nicht präzise ist. Anschließend wurde beschrieben, wie die verschiedenen DeepSurv-Modelle trainiert und optimiert wurden. Die daraus folgenden Modelle hatten gute Metriken. Jedoch konnte man in keinen der Modelle die gewünschten Beobachtungen machen, dass entweder alle Termine eines Patienten vor dem Wechsel des AMD-Stadiums einen hohen Riskscore haben oder dass der Riskscore mit der Zeit bei Patienten mit einem Wechsel des AMD-Stadiums steigt.

5.2 AUSBLICK

In diesem Abschnitt werden einige Vorschläge genannt, die bei Folgearbeiten nützlich sein könnten. Bei dem Mask R-CNN wäre die Verbesserung der Qualität der Trainingsdaten ein Anfang. Die Biomarker wurden anhand von Beispielbildern markiert. In einigen Fällen weichen die Formen der Biomar-

ker jedoch von diesen Beispielen ab. Eine mögliche Verbesserung der Qualität wäre, wenn ein Experte die Daten betrachtet und korrigiert. Dies würde wahrscheinlich das Problem mit dem multiplen Klassifizieren eines Objektes beheben. Man kann das Modell durchaus erweitern, um noch mehr Arten von Biomarkern zu erkennen. Jedoch spielt die Quantität eine große Rolle beim Trainieren des Mask R-CNN. Es sollte vor allem dafür gesorgt werden, dass jede Art von Biomarker auch oft genug in den Trainings- und Testdaten vorhanden ist. Es ist schwierig ein Minimum an Bildern zu nennen, die notwendig sind, damit ein Biomarker richtig erkannt wird. 50 Bilder pro Biomarker, in denen diese gut zu erkennen sind, sollten jedoch ein guter Startpunkt sein. Folgearbeiten könnten zudem die lokale Information des Schnittes mit einbeziehen. Diese Information ist theoretisch durch die linken Bilder der OCT-Videos, beschrieben in Abschnitt 2.2.1.1, vorhanden. Dadurch könnte man zum Beispiel bestimmen, ob eine geographische Atrophie zentral liegt (siehe "Fortgeschrittene AMD" in Tabelle 2.1).

Bei der Survival-Analyse ist es schwer zu ermitteln, woran diese genau gescheitert ist. Potentielle Begründungen wäre eine zu geringe Anzahl an Patienten oder fehlende Metadaten. Vor allem die Metadaten können bei der Survival-Analyse helfen, da diverse Faktoren, wie Rauchverhalten, Gewicht und Herzkrankheiten nicht betrachtet wurden. Die Messungen der Biomarker könnten aber auch ein Grund sein, da man auf Grund der Sensitivitäten der logistischen Regression darauf schließen könnte, dass die Messungen nicht präzise genug für eine Klassifizierung eines Patienten sind. Man kann jedoch davon ausgehen, dass die Messungen der Biomarker zusammen mit Alter und Visuswerten nicht ausreichen, um ein aussagekräftiges Survival-Modell zu erstellen. Folgearbeiten sollten zudem erwägen, die Informationen beider Augen zusammenfassen, damit die Reihen unabhängig voneinander sind. Dies war in unserem Fall nicht möglich, da nicht von jedem Patienten immer beide Augen untersucht wurden. Zudem sollten die Daten von mindestens 300 verschiedenen Patienten verwendet werden, um für eine größere Varianz zwischen den kumulierten Verteilungsfunktionen zu sorgen. Dies konnte man gut erkennen, als wir den Datensatz von 285 auf 100 Reihen reduzierten. Auch das Verwenden einer anderen Zeitachse könnte man ausprobieren. In dieser Arbeit wurde das Alter und die Termine der Patienten für die Zeitachse verwendet. Eine andere mögliche Zeitachse wäre das Verwenden eines festen Zeitraums. Damit ist gemeint, dass ein festes Datum als Startpunkt genommen wird, der für alle Patienten gleich ist. Ein möglicher Startpunkt wäre der früheste Termin im Datensatz.

Teil II

APPENDIX

A

METADATEN

Patient ID	date of birth	side of eyes	date of visit	OCT avaiable	FAF avaiable	CFF avaiable	FLA avaiable	ICG avaiable
5	16.09.1945	right	30.11.2017	yes	no	no	no	no
5	16.09.1945	left	30.11.2017	yes	no	no	no	no
7	01.01.1940	right	04.12.2017	yes	yes	yes	no	no

AMD stage (Beckman Classification) in OCT	hart Drusen in OCT	soft Drusen in OCT	cuticular Drusen in OCT
intermediate	yes	yes	no
intermediate	yes	yes	no
intermediate	yes	yes	no

reticular Pseudodrusen in OCT	Pachydrusen in OCT	Pachydrusen in OCT	drusen area in disc areas in infrared image
no	no	no	5
no	no	no	3
no	no	no	1

maximal drusen diameter (µm) in OCT	Drusen in zentral 3000µm in OCT	elipsoid zone intact in OCT
309	yes	yes
277	yes	yes
468	yes	yes

hyperreflective spots in OCT	atrophy signs in OCT	outer retinale tubulations in OCT	intraretinal fluid in OCT
no	no	no	no
no	no	no	no
yes	no	no	no

subretinal fluid in OCT	PED in FLA	GA in OCT	neovascular AMD in OCT	if GA: pattern in FAF (Bindewald)
no	no	no	no	/
no	no	no	no	/
no	no	no	no	/

if GA: GA area (region finder mm2)	if GA: maximal lesion size (µm) in FAF	if GA: fovea affected? In FAF
/	/	/
/	/	/
/	/	/

if exudative AMD? Classic, predominant classic, minimal classic, occult, RAP, PCV in FLA
/
/
/

B

MASK R-CNN SCHICHTEN

```
__________________________________________________________________________
Layer (type)                                    Connected to
==========================================================================
input_image (InputLayer)
__________________________________________________________________________
----------------------------ResNet50-------------------------------


zero_padding2d_2 (ZeroPadding2D)                input_image[0][0]
__________________________________________________________________________
conv1 (Conv2D)                                  zero_padding2d_2[0][0]
__________________________________________________________________________
bn_conv1 (BatchNorm)                            conv1[0][0]
__________________________________________________________________________
activation_41 (Activation)                      bn_conv1[0][0]
__________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)                  activation_41[0][0]
__________________________________________________________________________
res2a_branch2a (Conv2D)                         max_pooling2d_2[0][0]
__________________________________________________________________________
bn2a_branch2a (BatchNorm)                       res2a_branch2a[0][0]
__________________________________________________________________________
activation_42 (Activation)                      bn2a_branch2a[0][0]
__________________________________________________________________________
res2a_branch2b (Conv2D)                         activation_42[0][0]
__________________________________________________________________________
bn2a_branch2b (BatchNorm)                       res2a_branch2b[0][0]
__________________________________________________________________________
activation_43 (Activation)                      bn2a_branch2b[0][0]
__________________________________________________________________________
res2a_branch2c (Conv2D)                         activation_43[0][0]
__________________________________________________________________________
res2a_branch1 (Conv2D)                          max_pooling2d_2[0][0]
__________________________________________________________________________
bn2a_branch2c (BatchNorm)                       res2a_branch2c[0][0]
__________________________________________________________________________
bn2a_branch1 (BatchNorm)                        res2a_branch1[0][0]
__________________________________________________________________________
add_17 (Add)                                    bn2a_branch2c[0][0]
                                                bn2a_branch1[0][0]
__________________________________________________________________________
```

```
res2a_out (Activation)                  add_17[0][0]
__________________________________________________________________________
res2b_branch2a (Conv2D)                 res2a_out[0][0]
__________________________________________________________________________
bn2b_branch2a (BatchNorm)               res2b_branch2a[0][0]
__________________________________________________________________________
activation_44 (Activation)              bn2b_branch2a[0][0]
__________________________________________________________________________
res2b_branch2b (Conv2D)                 activation_44[0][0]
__________________________________________________________________________
bn2b_branch2b (BatchNorm)               res2b_branch2b[0][0]
__________________________________________________________________________
activation_45 (Activation)              bn2b_branch2b[0][0]
__________________________________________________________________________
res2b_branch2c (Conv2D)                 activation_45[0][0]
__________________________________________________________________________
bn2b_branch2c (BatchNorm)               res2b_branch2c[0][0]
__________________________________________________________________________
add_18 (Add)                            bn2b_branch2c[0][0]
                                        res2a_out[0][0]
__________________________________________________________________________
res2b_out (Activation)                  add_18[0][0]
__________________________________________________________________________
res2c_branch2a (Conv2D)                 res2b_out[0][0]
__________________________________________________________________________
bn2c_branch2a (BatchNorm)               res2c_branch2a[0][0]
__________________________________________________________________________
activation_46 (Activation)              bn2c_branch2a[0][0]
__________________________________________________________________________
res2c_branch2b (Conv2D)                 activation_46[0][0]
__________________________________________________________________________
bn2c_branch2b (BatchNorm)               res2c_branch2b[0][0]
__________________________________________________________________________
activation_47 (Activation)              bn2c_branch2b[0][0]
__________________________________________________________________________
res2c_branch2c (Conv2D)                 activation_47[0][0]
__________________________________________________________________________
bn2c_branch2c (BatchNorm)               res2c_branch2c[0][0]
__________________________________________________________________________
add_19 (Add)                            bn2c_branch2c[0][0]
                                        res2b_out[0][0]
__________________________________________________________________________
res2c_out (Activation)                  add_19[0][0]
__________________________________________________________________________
res3a_branch2a (Conv2D)                 res2c_out[0][0]
__________________________________________________________________________
```

```
bn3a_branch2a (BatchNorm)                res3a_branch2a[0][0]
__________________________________________________________________________
activation_48 (Activation)               bn3a_branch2a[0][0]
__________________________________________________________________________
res3a_branch2b (Conv2D)                  activation_48[0][0]
__________________________________________________________________________
bn3a_branch2b (BatchNorm)                res3a_branch2b[0][0]
__________________________________________________________________________
activation_49 (Activation)               bn3a_branch2b[0][0]
__________________________________________________________________________
res3a_branch2c (Conv2D)                  activation_49[0][0]
__________________________________________________________________________
res3a_branch1 (Conv2D)                   res2c_out[0][0]
__________________________________________________________________________
bn3a_branch2c (BatchNorm)                res3a_branch2c[0][0]
__________________________________________________________________________
bn3a_branch1 (BatchNorm)                 res3a_branch1[0][0]
__________________________________________________________________________
add_20 (Add)                             bn3a_branch2c[0][0]
                                         bn3a_branch1[0][0]
__________________________________________________________________________
res3a_out (Activation)                   add_20[0][0]
__________________________________________________________________________
res3b_branch2a (Conv2D)                  res3a_out[0][0]
__________________________________________________________________________
bn3b_branch2a (BatchNorm)                res3b_branch2a[0][0]
__________________________________________________________________________
activation_50 (Activation)               bn3b_branch2a[0][0]
__________________________________________________________________________
res3b_branch2b (Conv2D)                  activation_50[0][0]
__________________________________________________________________________
bn3b_branch2b (BatchNorm)                res3b_branch2b[0][0]
__________________________________________________________________________
activation_51 (Activation)               bn3b_branch2b[0][0]
__________________________________________________________________________
res3b_branch2c (Conv2D)                  activation_51[0][0]
__________________________________________________________________________
bn3b_branch2c (BatchNorm)                res3b_branch2c[0][0]
__________________________________________________________________________
add_21 (Add)                             bn3b_branch2c[0][0]
                                         res3a_out[0][0]
__________________________________________________________________________
res3b_out (Activation)                   add_21[0][0]
__________________________________________________________________________
res3c_branch2a (Conv2D)                  res3b_out[0][0]
__________________________________________________________________________
```

```
bn3c_branch2a (BatchNorm)                res3c_branch2a[0][0]
__________________________________________________________________________
activation_52 (Activation)               bn3c_branch2a[0][0]
__________________________________________________________________________
res3c_branch2b (Conv2D)                  activation_52[0][0]
__________________________________________________________________________
bn3c_branch2b (BatchNorm)                res3c_branch2b[0][0]
__________________________________________________________________________
activation_53 (Activation)               bn3c_branch2b[0][0]
__________________________________________________________________________
res3c_branch2c (Conv2D)                  activation_53[0][0]
__________________________________________________________________________
bn3c_branch2c (BatchNorm)                res3c_branch2c[0][0]
__________________________________________________________________________
add_22 (Add)                             bn3c_branch2c[0][0]
                                         res3b_out[0][0]
__________________________________________________________________________
res3c_out (Activation)                   add_22[0][0]
__________________________________________________________________________
res3d_branch2a (Conv2D)                  res3c_out[0][0]
__________________________________________________________________________
bn3d_branch2a (BatchNorm)                res3d_branch2a[0][0]
__________________________________________________________________________
activation_54 (Activation)               bn3d_branch2a[0][0]
__________________________________________________________________________
res3d_branch2b (Conv2D)                  activation_54[0][0]
__________________________________________________________________________
bn3d_branch2b (BatchNorm)                res3d_branch2b[0][0]
__________________________________________________________________________
activation_55 (Activation)               bn3d_branch2b[0][0]
__________________________________________________________________________
res3d_branch2c (Conv2D)                  activation_55[0][0]
__________________________________________________________________________
bn3d_branch2c (BatchNorm)                res3d_branch2c[0][0]
__________________________________________________________________________
add_23 (Add)                             bn3d_branch2c[0][0]
                                         res3c_out[0][0]
__________________________________________________________________________
res3d_out (Activation)                   add_23[0][0]
__________________________________________________________________________
res4a_branch2a (Conv2D)                  res3d_out[0][0]
__________________________________________________________________________
bn4a_branch2a (BatchNorm)                res4a_branch2a[0][0]
__________________________________________________________________________
activation_56 (Activation)               bn4a_branch2a[0][0]
__________________________________________________________________________
```

```
res4a_branch2b (Conv2D)                  activation_56[0][0]
_____________________________________________________________________
bn4a_branch2b (BatchNorm)                res4a_branch2b[0][0]
_____________________________________________________________________
activation_57 (Activation)               bn4a_branch2b[0][0]
_____________________________________________________________________
res4a_branch2c (Conv2D)                  activation_57[0][0]
_____________________________________________________________________
res4a_branch1 (Conv2D)                   res3d_out[0][0]
_____________________________________________________________________
bn4a_branch2c (BatchNorm)                res4a_branch2c[0][0]
_____________________________________________________________________
bn4a_branch1 (BatchNorm)                 res4a_branch1[0][0]
_____________________________________________________________________
add_24 (Add)                             bn4a_branch2c[0][0]
                                         bn4a_branch1[0][0]
_____________________________________________________________________
res4a_out (Activation)                   add_24[0][0]
_____________________________________________________________________
res4b_branch2a (Conv2D)                  res4a_out[0][0]
_____________________________________________________________________
bn4b_branch2a (BatchNorm)                res4b_branch2a[0][0]
_____________________________________________________________________
activation_58 (Activation)               bn4b_branch2a[0][0]
_____________________________________________________________________
res4b_branch2b (Conv2D)                  activation_58[0][0]
_____________________________________________________________________
bn4b_branch2b (BatchNorm)                res4b_branch2b[0][0]
_____________________________________________________________________
activation_59 (Activation)               bn4b_branch2b[0][0]
_____________________________________________________________________
res4b_branch2c (Conv2D)                  activation_59[0][0]
_____________________________________________________________________
bn4b_branch2c (BatchNorm)                res4b_branch2c[0][0]
_____________________________________________________________________
add_25 (Add)                             bn4b_branch2c[0][0]
                                         res4a_out[0][0]
_____________________________________________________________________
res4b_out (Activation)                   add_25[0][0]
_____________________________________________________________________
res4c_branch2a (Conv2D)                  res4b_out[0][0]
_____________________________________________________________________
bn4c_branch2a (BatchNorm)                res4c_branch2a[0][0]
_____________________________________________________________________
activation_60 (Activation)               bn4c_branch2a[0][0]
_____________________________________________________________________
```

```
res4c_branch2b (Conv2D)                  activation_60[0][0]
_________________________________________________________________________
bn4c_branch2b (BatchNorm)                res4c_branch2b[0][0]
_________________________________________________________________________
activation_61 (Activation)               bn4c_branch2b[0][0]
_________________________________________________________________________
res4c_branch2c (Conv2D)                  activation_61[0][0]
_________________________________________________________________________
bn4c_branch2c (BatchNorm)                res4c_branch2c[0][0]
_________________________________________________________________________
add_26 (Add)                             bn4c_branch2c[0][0]
                                         res4b_out[0][0]
_________________________________________________________________________
res4c_out (Activation)                   add_26[0][0]
_________________________________________________________________________
res4d_branch2a (Conv2D)                  res4c_out[0][0]
_________________________________________________________________________
bn4d_branch2a (BatchNorm)                res4d_branch2a[0][0]
_________________________________________________________________________
activation_62 (Activation)               bn4d_branch2a[0][0]
_________________________________________________________________________
res4d_branch2b (Conv2D)                  activation_62[0][0]
_________________________________________________________________________
bn4d_branch2b (BatchNorm)                res4d_branch2b[0][0]
_________________________________________________________________________
activation_63 (Activation)               bn4d_branch2b[0][0]
_________________________________________________________________________
res4d_branch2c (Conv2D)                  activation_63[0][0]
_________________________________________________________________________
bn4d_branch2c (BatchNorm)                res4d_branch2c[0][0]
_________________________________________________________________________
add_27 (Add)                             bn4d_branch2c[0][0]
                                         res4c_out[0][0]
_________________________________________________________________________
res4d_out (Activation)                   add_27[0][0]
_________________________________________________________________________
res4e_branch2a (Conv2D)                  res4d_out[0][0]
_________________________________________________________________________
bn4e_branch2a (BatchNorm)                res4e_branch2a[0][0]
_________________________________________________________________________
activation_64 (Activation)               bn4e_branch2a[0][0]
_________________________________________________________________________
res4e_branch2b (Conv2D)                  activation_64[0][0]
_________________________________________________________________________
bn4e_branch2b (BatchNorm)                res4e_branch2b[0][0]
_________________________________________________________________________
```

```
activation_65 (Activation)                bn4e_branch2b[0][0]
__________________________________________________________________________
res4e_branch2c (Conv2D)                   activation_65[0][0]
__________________________________________________________________________
bn4e_branch2c (BatchNorm)                 res4e_branch2c[0][0]
__________________________________________________________________________
add_28 (Add)                              bn4e_branch2c[0][0]
                                          res4d_out[0][0]
__________________________________________________________________________
res4e_out (Activation)                    add_28[0][0]
__________________________________________________________________________
res4f_branch2a (Conv2D)                   res4e_out[0][0]
__________________________________________________________________________
bn4f_branch2a (BatchNorm)                 res4f_branch2a[0][0]
__________________________________________________________________________
activation_66 (Activation)                bn4f_branch2a[0][0]
__________________________________________________________________________
res4f_branch2b (Conv2D)                   activation_66[0][0]
__________________________________________________________________________
bn4f_branch2b (BatchNorm)                 res4f_branch2b[0][0]
__________________________________________________________________________
activation_67 (Activation)                bn4f_branch2b[0][0]
__________________________________________________________________________
res4f_branch2c (Conv2D)                   activation_67[0][0]
__________________________________________________________________________
bn4f_branch2c (BatchNorm)                 res4f_branch2c[0][0]
__________________________________________________________________________
add_29 (Add)                              bn4f_branch2c[0][0]
                                          res4e_out[0][0]
__________________________________________________________________________
res4f_out (Activation)                    add_29[0][0]
__________________________________________________________________________
res5a_branch2a (Conv2D)                   res4f_out[0][0]
__________________________________________________________________________
bn5a_branch2a (BatchNorm)                 res5a_branch2a[0][0]
__________________________________________________________________________
activation_68 (Activation)                bn5a_branch2a[0][0]
__________________________________________________________________________
res5a_branch2b (Conv2D)                   activation_68[0][0]
__________________________________________________________________________
bn5a_branch2b (BatchNorm)                 res5a_branch2b[0][0]
__________________________________________________________________________
activation_69 (Activation)                bn5a_branch2b[0][0]
__________________________________________________________________________
res5a_branch2c (Conv2D)                   activation_69[0][0]
__________________________________________________________________________
```

```
res5a_branch1 (Conv2D)                  res4f_out[0][0]
_________________________________________________________________
bn5a_branch2c (BatchNorm)               res5a_branch2c[0][0]
_________________________________________________________________
bn5a_branch1 (BatchNorm)                res5a_branch1[0][0]
_________________________________________________________________
add_30 (Add)                            bn5a_branch2c[0][0]
                                        bn5a_branch1[0][0]
_________________________________________________________________
res5a_out (Activation)                  add_30[0][0]
_________________________________________________________________
res5b_branch2a (Conv2D)                 res5a_out[0][0]
_________________________________________________________________
bn5b_branch2a (BatchNorm)               res5b_branch2a[0][0]
_________________________________________________________________
activation_70 (Activation)              bn5b_branch2a[0][0]
_________________________________________________________________
res5b_branch2b (Conv2D)                 activation_70[0][0]
_________________________________________________________________
bn5b_branch2b (BatchNorm)               res5b_branch2b[0][0]
_________________________________________________________________
activation_71 (Activation)              bn5b_branch2b[0][0]
_________________________________________________________________
res5b_branch2c (Conv2D)                 activation_71[0][0]
_________________________________________________________________
bn5b_branch2c (BatchNorm)               res5b_branch2c[0][0]
_________________________________________________________________
add_31 (Add)                            bn5b_branch2c[0][0]
                                        res5a_out[0][0]
_________________________________________________________________
res5b_out (Activation)                  add_31[0][0]
_________________________________________________________________
res5c_branch2a (Conv2D)                 res5b_out[0][0]
_________________________________________________________________
bn5c_branch2a (BatchNorm)               res5c_branch2a[0][0]
_________________________________________________________________
activation_72 (Activation)              bn5c_branch2a[0][0]
_________________________________________________________________
res5c_branch2b (Conv2D)                 activation_72[0][0]
_________________________________________________________________
bn5c_branch2b (BatchNorm)               res5c_branch2b[0][0]
_________________________________________________________________
activation_73 (Activation)              bn5c_branch2b[0][0]
_________________________________________________________________
res5c_branch2c (Conv2D)                 activation_73[0][0]
_________________________________________________________________
```

```
bn5c_branch2c (BatchNorm)                  res5c_branch2c[0][0]
__________________________________________________________________________________
add_32 (Add)                               bn5c_branch2c[0][0]
                                           res5b_out[0][0]
__________________________________________________________________________________
res5c_out (Activation)                     add_32[0][0]
__________________________________________________________________________________
------------------------------FPN-----------------------------------


fpn_c5p5 (Conv2D)                          res5c_out[0][0]
__________________________________________________________________________________
fpn_p5upsampled (UpSampling2D)             fpn_c5p5[0][0]
__________________________________________________________________________________
fpn_c4p4 (Conv2D)                          res4f_out[0][0]
__________________________________________________________________________________
fpn_p4add (Add)                            fpn_p5upsampled[0][0]
                                           fpn_c4p4[0][0]
__________________________________________________________________________________
fpn_p4upsampled (UpSampling2D)             fpn_p4add[0][0]
__________________________________________________________________________________
fpn_c3p3 (Conv2D)                          res3d_out[0][0]
__________________________________________________________________________________
fpn_p3add (Add)                            fpn_p4upsampled[0][0]
                                           fpn_c3p3[0][0]
__________________________________________________________________________________
fpn_p3upsampled (UpSampling2D)             fpn_p3add[0][0]
__________________________________________________________________________________
fpn_c2p2 (Conv2D)                          res2c_out[0][0]
__________________________________________________________________________________
fpn_p2add (Add)                            fpn_p3upsampled[0][0]
                                           fpn_c2p2[0][0]
__________________________________________________________________________________
fpn_p5 (Conv2D)                            fpn_c5p5[0][0]
__________________________________________________________________________________
fpn_p2 (Conv2D)                            fpn_p2add[0][0]
__________________________________________________________________________________
fpn_p3 (Conv2D)                            fpn_p3add[0][0]
__________________________________________________________________________________
fpn_p4 (Conv2D)                            fpn_p4add[0][0]
__________________________________________________________________________________
fpn_p6 (MaxPooling2D)                      fpn_p5[0][0]
__________________________________________________________________________________
------------------------------RPN-----------------------------------
```

```
rpn_model (Model)                       [0]
                                        fpn_p3[0][0]
                                        fpn_p4[0][0]
                                        fpn_p5[0][0]
                                        fpn_p6[0][0]
__________________________________________________________________________
rpn_class (Concatenate)                 rpn_model[1][1]
                                        rpn_model[2][1]
                                        rpn_model[3][1]
                                        rpn_model[4][1]
                                        rpn_model[5][1]
__________________________________________________________________________
rpn_bbox (Concatenate)                  rpn_model[1][2]
                                        rpn_model[2][2]
                                        rpn_model[3][2]
                                        rpn_model[4][2]
                                        rpn_model[5][2]
__________________________________________________________________________
input_anchors (InputLayer)
__________________________________________________________________________
ROI (ProposalLayer)                     rpn_class[0][0]
                                        rpn_bbox[0][0]
                                        input_anchors[0][0]
__________________________________________________________________________
------------------------------Mask R-CNN------------------------------


input_image_meta (InputLayer)
__________________________________________________________________________
roi_align_classifier (PyramidROIAlign)  ROI[0][0]
                                        input_image_meta[0][0]
                                        fpn_p2[0][0]
                                        fpn_p3[0][0]
                                        fpn_p4[0][0]
                                        fpn_p5[0][0]
__________________________________________________________________________
mrcnn_class_conv1 (TimeDistributed)     roi_align_classifier[0][0]
__________________________________________________________________________
mrcnn_class_bn1 (TimeDistributed)       mrcnn_class_conv1[0][0]
__________________________________________________________________________
activation_74 (Activation)              mrcnn_class_bn1[0][0]
__________________________________________________________________________
mrcnn_class_conv2 (TimeDistributed)     activation_74[0][0]
__________________________________________________________________________
mrcnn_class_bn2 (TimeDistributed)       mrcnn_class_conv2[0][0]
__________________________________________________________________________
```

```
activation_75 (Activation)                mrcnn_class_bn2[0][0]
__________________________________________________________________________
pool_squeeze (Lambda)                     activation_75[0][0]
__________________________________________________________________________
mrcnn_class_logits (TimeDistributed)      pool_squeeze[0][0]
__________________________________________________________________________
mrcnn_bbox_fc (TimeDistributed)           pool_squeeze[0][0]
__________________________________________________________________________
mrcnn_class (TimeDistributed)             mrcnn_class_logits[0][0]
__________________________________________________________________________
mrcnn_bbox (Reshape)                      mrcnn_bbox_fc[0][0]
__________________________________________________________________________
mrcnn_detection (DetectionLayer)          ROI[0][0]
                                          mrcnn_class[0][0]
                                          mrcnn_bbox[0][0]
                                          input_image_meta[0][0]
__________________________________________________________________________
lambda_7 (Lambda)                         mrcnn_detection[0][0]
__________________________________________________________________________
roi_align_mask (PyramidROIAlign)          lambda_7[0][0]
                                          input_image_meta[0][0]
                                          fpn_p2[0][0]
                                          fpn_p3[0][0]
                                          fpn_p4[0][0]
                                          fpn_p5[0][0]
__________________________________________________________________________
mrcnn_mask_conv1 (TimeDistributed)        roi_align_mask[0][0]
__________________________________________________________________________
mrcnn_mask_bn1 (TimeDistributed)          mrcnn_mask_conv1[0][0]
__________________________________________________________________________
activation_77 (Activation)                mrcnn_mask_bn1[0][0]
__________________________________________________________________________
mrcnn_mask_conv2 (TimeDistributed)        activation_77[0][0]
__________________________________________________________________________
mrcnn_mask_bn2 (TimeDistributed)          mrcnn_mask_conv2[0][0]
__________________________________________________________________________
activation_78 (Activation)                mrcnn_mask_bn2[0][0]
__________________________________________________________________________
mrcnn_mask_conv3 (TimeDistributed)        activation_78[0][0]
__________________________________________________________________________
mrcnn_mask_bn3 (TimeDistributed)          mrcnn_mask_conv3[0][0]
__________________________________________________________________________
activation_79 (Activation)                mrcnn_mask_bn3[0][0]
__________________________________________________________________________
mrcnn_mask_conv4 (TimeDistributed)        activation_79[0][0]
__________________________________________________________________________
```

```
mrcnn_mask_bn4 (TimeDistributed)          mrcnn_mask_conv4[0][0]
__________________________________________________________________________
activation_80 (Activation)                mrcnn_mask_bn4[0][0]
__________________________________________________________________________
mrcnn_mask_deconv (TimeDistributed)       activation_80[0][0]
__________________________________________________________________________
mrcnn_mask (TimeDistributed)              mrcnn_mask_deconv[0][0]
==========================================================================
```

C

MASK R-CNN TRAININGSSCHICHTEN

Training-Struktur

Layer (type)	Connected to
proposal_targets (DetectionTarget)	ROI[0][0] input_gt_class_ids[0][0] lambda_1[0][0] input_gt_masks[0][0]
input_image_meta (InputLayer)	
roi_align_mask (PyramidROIAlign)	proposal_targets[0][0] input_image_meta[0][0] fpn_p2[0][0] fpn_p3[0][0] fpn_p4[0][0] fpn_p5[0][0]
mrcnn_mask_conv1 (TimeDistributed)	roi_align_mask[0][0]
mrcnn_mask_bn1 (TimeDistributed)	mrcnn_mask_conv1[0][0]
activation_37 (Activation)	mrcnn_mask_bn1[0][0]
mrcnn_mask_conv2 (TimeDistributed)	activation_37[0][0]
roi_align_classifier (PyramidROIAlign)	proposal_targets[0][0] input_image_meta[0][0] fpn_p2[0][0] fpn_p3[0][0] fpn_p4[0][0] fpn_p5[0][0]
mrcnn_mask_bn2 (TimeDistributed)	mrcnn_mask_conv2[0][0]
mrcnn_class_conv1 (TimeDistributed)	roi_align_classifier[0][0]
activation_38 (Activation)	mrcnn_mask_bn2[0][0]

```
mrcnn_class_bn1 (TimeDistributed)      mrcnn_class_conv1[0][0]
__________________________________________________________________
mrcnn_mask_conv3 (TimeDistributed)     activation_38[0][0]
__________________________________________________________________
activation_34 (Activation)             mrcnn_class_bn1[0][0]
__________________________________________________________________
mrcnn_mask_bn3 (TimeDistributed        mrcnn_mask_conv3[0][0]
__________________________________________________________________
mrcnn_class_conv2 (TimeDistributed)    activation_34[0][0]
__________________________________________________________________
activation_39 (Activation)             mrcnn_mask_bn3[0][0]
__________________________________________________________________
mrcnn_class_bn2 (TimeDistributed)      mrcnn_class_conv2[0][0]
__________________________________________________________________
mrcnn_mask_conv4 (TimeDistributed)     activation_39[0][0]
__________________________________________________________________
activation_35 (Activation)             mrcnn_class_bn2[0][0]
__________________________________________________________________
mrcnn_mask_bn4 (TimeDistributed)       mrcnn_mask_conv4[0][0]
__________________________________________________________________
pool_squeeze (Lambda)                  activation_35[0][0]
__________________________________________________________________
activation_40 (Activation)             mrcnn_mask_bn4[0][0]
__________________________________________________________________
mrcnn_bbox_fc (TimeDistributed)        pool_squeeze[0][0]
__________________________________________________________________
mrcnn_mask_deconv (TimeDistributed)    activation_40[0][0]
__________________________________________________________________
rpn_class_logits (Concatenate)         rpn_model[1][0]
                                       rpn_model[2][0]
                                       rpn_model[3][0]
                                       rpn_model[4][0]
                                       rpn_model[5][0]
__________________________________________________________________
mrcnn_class_logits (TimeDistributed)   pool_squeeze[0][0]
__________________________________________________________________
mrcnn_bbox (Reshape)                   mrcnn_bbox_fc[0][0]
__________________________________________________________________
mrcnn_mask (TimeDistributed)           mrcnn_mask_deconv[0][0]
__________________________________________________________________
input_rpn_match (InputLayer)
__________________________________________________________________
input_rpn_bbox (InputLayer)
__________________________________________________________________
lambda_4 (Lambda)                      input_image_meta[0][0]
__________________________________________________________________
```

```
mrcnn_class (TimeDistributed)          mrcnn_class_logits[0][0]
__________________________________________________________________
output_rois (Lambda)                   proposal_targets[0][0]
__________________________________________________________________
rpn_class_loss (Lambda)                input_rpn_match[0][0]
                                       rpn_class_logits[0][0]
__________________________________________________________________
rpn_bbox_loss (Lambda)                 input_rpn_bbox[0][0]
                                       input_rpn_match[0][0]
                                       rpn_bbox[0][0]
__________________________________________________________________
mrcnn_class_loss (Lambda)              proposal_targets[0][1]
                                       mrcnn_class_logits[0][0]
                                       lambda_4[0][0]
__________________________________________________________________
mrcnn_bbox_loss (Lambda)               proposal_targets[0][2]
                                       proposal_targets[0][1]
                                       mrcnn_bbox[0][0]
__________________________________________________________________
mrcnn_mask_loss (Lambda)               proposal_targets[0][3]
                                       proposal_targets[0][1]
                                       mrcnn_mask[0][0]
==================================================================
```

D

MASK R-CNN PARAMETER

```
Configurations:
BACKBONE                        resnet50
BACKBONE_STRIDES                [4, 8, 16, 32, 64]
BATCH_SIZE                      1
BBOX_STD_DEV                    [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE          None
DETECTION_MAX_INSTANCES         100
DETECTION_MIN_CONFIDENCE        0.75
DETECTION_NMS_THRESHOLD         0.3
DEVICE                          /gpu:0
FPN_CLASSIF_FC_LAYERS_SIZE      1024
GPU_COUNT                       1
GRADIENT_CLIP_NORM              5.0
IMAGES_PER_GPU                  1
IMAGE_CHANNEL_COUNT             3
IMAGE_MAX_DIM                   1024
IMAGE_META_SIZE                 24
IMAGE_MIN_DIM                   800
IMAGE_MIN_SCALE                 0
IMAGE_RESIZE_MODE               square
IMAGE_SHAPE                     [1024 1024    3]
LEARNING_MOMENTUM               0.9
LEARNING_RATE                   0.002
LOSS_WEIGHTS                    {'rpn_class_loss': 1.0,
                                 'rpn_bbox_loss': 1.0,
                                 'mrcnn_class_loss': 1.0,
                                 'mrcnn_bbox_loss': 1.0,
                                 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                  14
MASK_SHAPE                      [28, 28]
MAX_GT_INSTANCES                100
MEAN_PIXEL                      [123.7 116.8 103.9]
MINI_MASK_SHAPE                 (56, 56)
NAME                            OCT
NUM_CLASSES                     12
POOL_SIZE                       7
POST_NMS_ROIS_INFERENCE         1000
POST_NMS_ROIS_TRAINING          2000
PRE_NMS_LIMIT                   6000
ROI_POSITIVE_RATIO              0.33
RPN_ANCHOR_RATIOS               [0.5, 1, 2]
```

RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE	1
RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
STEPS_PER_EPOCH	150
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_BN	False
TRAIN_ROIS_PER_IMAGE	100
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001

E

ERGEBNISSE FÜR ALTERNATIVE DEEPSURV-MODELLE

In diesem Appendix sind die Ergebnisse der DeepSurv-Modelle M_{Alter} und MU_{Alter} zu sehen. Es ist bei der Zeitachse zu beachten, dass das Alter des Patienten minus 53 Jahren als x-Achse verwendet wird. In Abbildung E.1 ist die kumulierte Verteilungsfunktion desselben Patienten in beiden Modellen zu sehen. Zusätzlich wurde noch die Grundgefahr eingezeichnet. Die Abbildung E.2 zeigt die kumulierte Verteilungsfunktionen mehrerer Patienten für Modell M_{Alter}. Das Ergebnis von Modell MU_{Alter} für dieselben Patienten ist in Abbildung E.3 zu sehen.

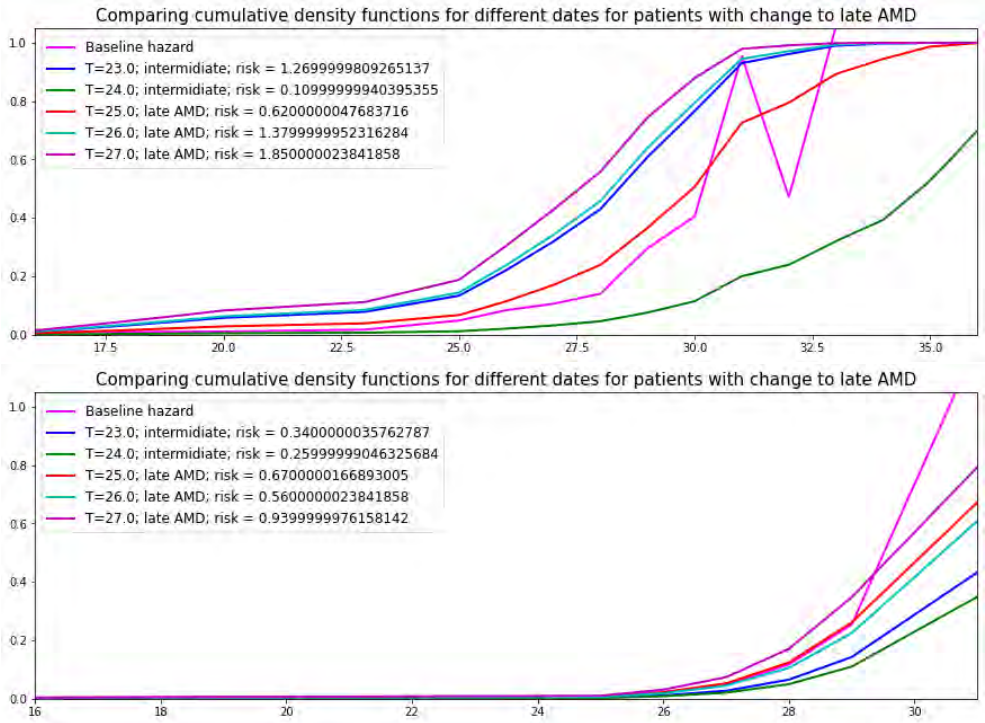

Abbildung E.1: kumulierte Verteilungsfunktion eines Patienten für Modell M_{Alter} (oben) und MU_{Alter} (unten)

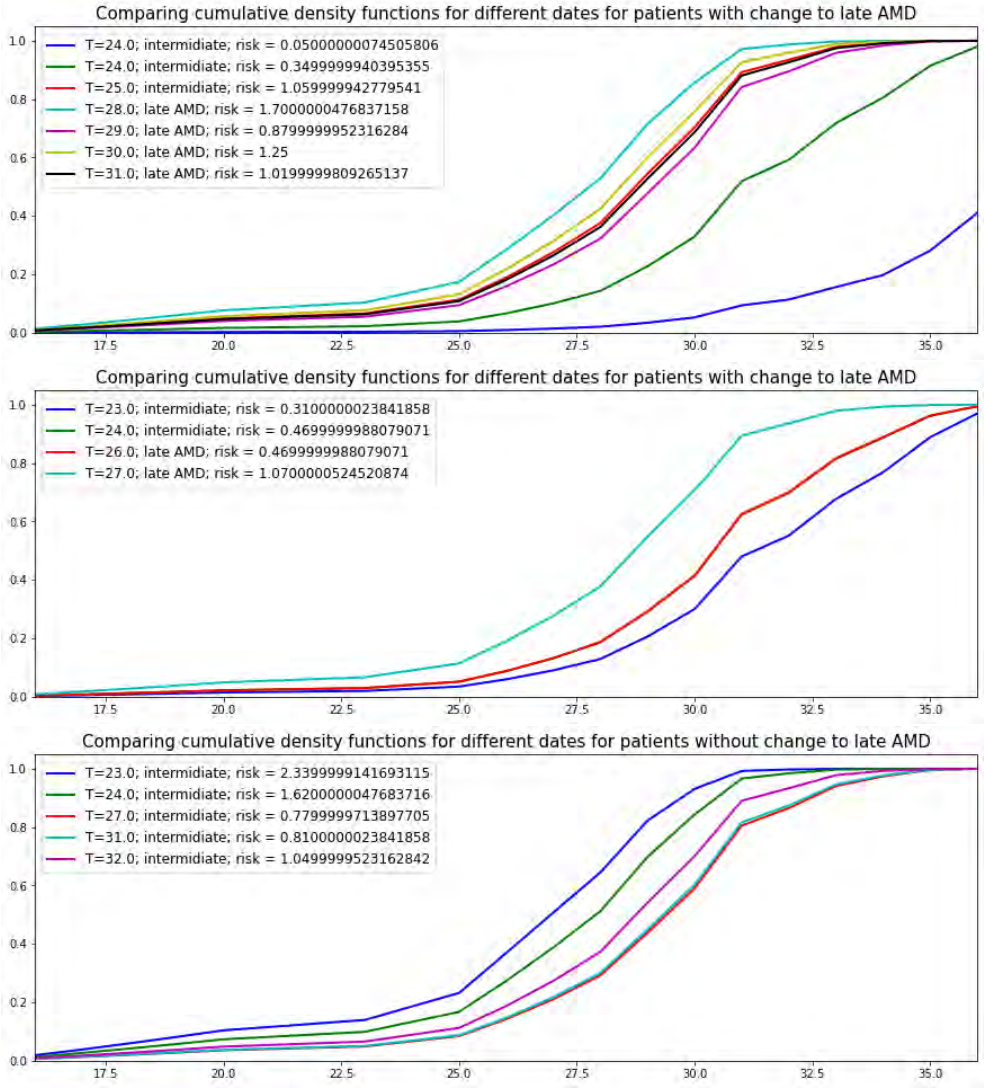

Abbildung E.2: kumulierte Verteilungsfunktion mehrerer Patienten für Modell M_{Alter}

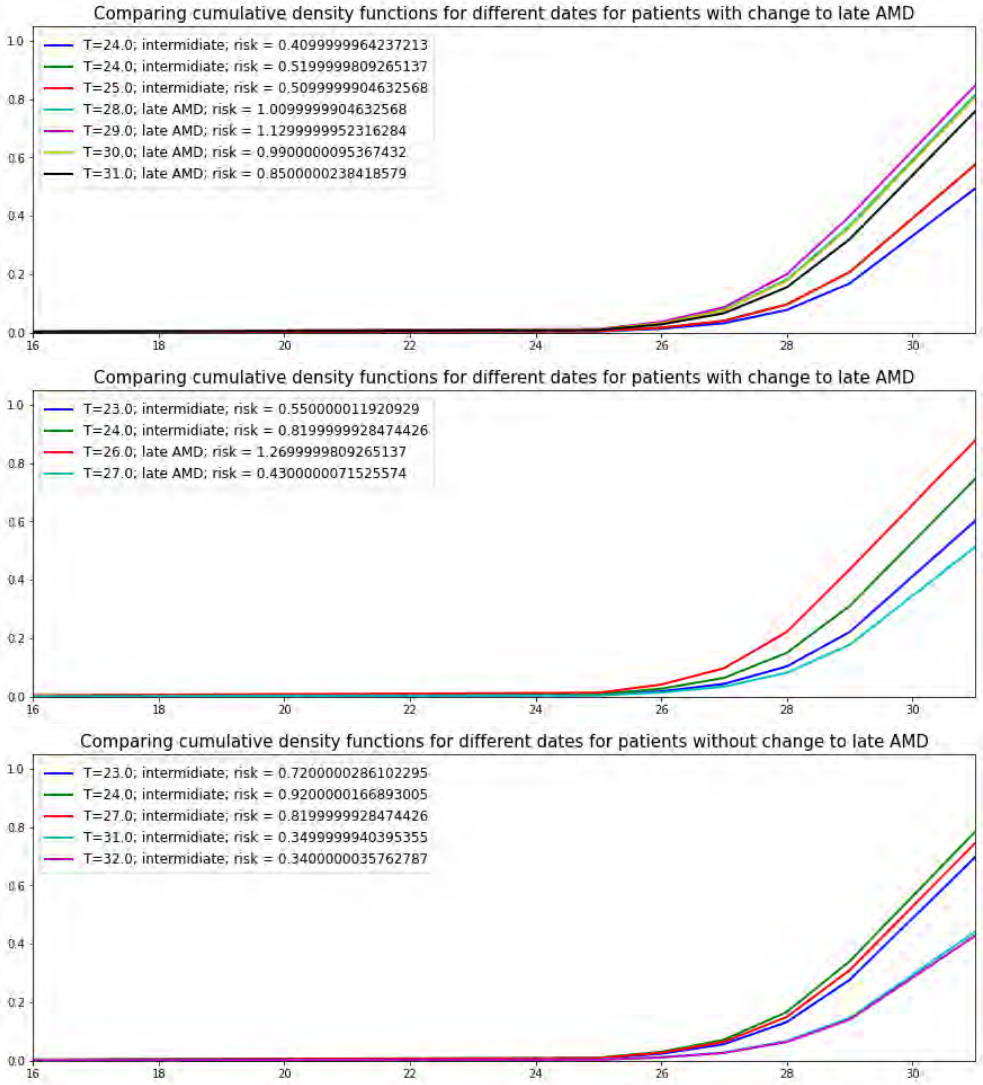

Abbildung E.3: kumulierte Verteilungsfunktion mehrerer Patienten für Modell MU_{Alter}

LITERATURVERZEICHNIS

[1] Rama D. Jager, William F. Mieler, Joan W. Miller. "Age-Related Macular Degeneration". In: *N Engl J Med* 2008;358:2606-17. DOI: 10.1056/NEJMra0801537

[2] Wanjiku Mathenge. "Age-related macular degeneration". In: *Community Eye Health* 2014;27(87):49-50. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4322741/

[3] Jean-François Girmens, José-Alain Sahel, and Katia Marazova1. "Dry age-related macular degeneration: A currently unmet clinical need". In: *Intractable Rare Dis Res* 2012;1(3):103-114. DOI: 10.5582/irdr.2012.v1.3.103

[4] "AMD: a preventable form of vision loss". In: *Havard Men's Health Watch* 2013. URL: https://www.health.harvard.edu/diseases-and-conditions/amd-a-preventable-form-of-vision-loss, letzter Aufruf am 10.03.2021

[5] Waseem M Al-Zamil, Sanaa A Yassin. "Recent developments in age-related macular degeneration: a review". In: *Clin Interv Aging* 2017;12:1313-1330. DOI: 10.2147/CIA.S143508

[6] Thomas Kurmann, Siqing Yu, Pablo Márquez-Neila, Andreas Ebneter, Martin Zinkernagel, Marion R. Munk, Sebastian Wolf, Raphael Sznitman. "Expert-level Automated Biomarker Identification in Optical Coherence Tomography Scans". In: *Sci Rep* 2019;9:13605. DOI: 10.1038/s41598-019-49740-7

[7] Daniela Di Schiena, Alexander Kniesz, Nina Krüger, Artur Moser, Boro Nedic, Thi Mai Tran, Christoph Wies. Projekt: "Berechnung des Drusenvolumens bei AMD / Vorhersage der Brechkraftänderung bei DMEK". *Hochschule Darmstadt* 2019

[8] Dominik Ludwig, David Sommer, Ivar van den Boogert, Mario Kevin Rinke, Maximilian Neudert, Oskar Rudolf, Sandra Sonnabend. Projekt: "Altersbedingte Makuladegeneration-Diagnose". *Hochschule Darmstadt* 2020

[9] James G Fujimoto, Costas Pitris, Stephen A Boppart, Mark E Brezinski. "Optical Coherence Tomography: An Emerging Technology for Biomedical Imaging and Optical Biopsy". In: *Neoplasia* 2000;2(1-2):9-25. DOI: 10.1038/sj.neo.7900071

[10] John W. Crabb, Masaru Miyagi, Xiaorong Gu, Karen Shadrach, Karen A. West, Hirokazu Sakaguchi, Motohiro Kamei, Azeem Hasan, Lin Yan, Mary E. Rayborn, Robert G. Salomon, Joe G. Hollyfield. "Drusen proteome

analysis: An approach to the etiology of age-related macular degeneration“. In: *PNAS* 2002;99(23):14682-14687. DOI: 10.1073/pnas.222551899

[11] Martin Mißfeldt. “Netzhaut (Retina) im Auge“. In: *OnlineSehtests*. URL: https://www.onlinesehtests.de/auge/netzhaut-retina.php, letzter Aufruf am 10.03.2021

[12] Yao Jin, Chen Xi, Jiang Qin, Ji Yong. “9 - Vitamin D and Age-Related Macular Degeneration“. In: *Handbook of Nutrition, Diet, and the Eye (Second Edition)* 2019;-:147-163. ISBN: 9780128152454. DOI: 10.1016/B978-0-12-815245-4.00009-0

[13] The Age-Related Eye Disease Study Research Group. “The Age-Related Eye Disease Study (AREDS): Design Implications AREDS Report No. 1“. In: *Control Clin Trials* 1999;20(6):573-600. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1473211/

[14] Arno P. Göbel, Monika Fleckenstein, Frank G. Holz, Bernd Bertram, Daniel Pauleikhoff, Horst Helbig. Äktuelle Stellungnahme der Deutschen Ophthalmologischen Gesellschaft, der Retinologischen Gesellschaft und des Berufsverbandes der Augenärzte Deutschlands zu Nahrungsergänzungsmitteln bei altersabhängiger Makuladegeneration (AMD)". In: *Deutsche Ophthalmologische Gesellschaft* 2014. URL: https://www.dog.org/wp-content/uploads/2013/03/zu-Nahrungsergänzungsmitteln-bei-AMD-Oktober-2014.pdf

[15] Jayakrishna Ambati, Benjamin J. Fowler. “Mechanisms of age-related macular degeneration“. In: *Neuron* 2012;75(1):26-39. DOI: 10.1016/j.neuron.2012.06.018

[16] Inger Christine Munch, Birgit Sander, Line Kessel, Jesper Leth Hougaard, Nina Charlotte Bille Brahe Taarnhøj, Thorkild I. A. Sørensen, Kirsten Ohm Kyvik, Michael Larsen. “Heredity of Small Hard Drusen in Twins Aged 20–46 Years“. In: *Invest. Ophthalmol. Vis. Sci.* 2007;48(2):833-838. DOI: 10.1167/iovs.06-0529

[17] Chandrakumar Balaratnasingam, Svetlana Cherepanoff, Rosa Dolz-Marco, Murray Killingsworth, Fred K. Chen, Randev Mendis, Sarah Mrejen, Lay Khoon Too, Orly Gal-Or, Christine A. Curcio, K. Bailey Freund, Lawrence A. Yannuzzi. “Cuticular Drusen - Clinical Phenotypes and Natural History Defined Using Multimodal Imaging“. In: *Ophthalmology* 2018;125:100-118. DOI: 10.1016/j.ophtha.2017.08.033

[18] Richard F. Spaide, Christine A. Curcio. “Drusen characterization with multimodal imaging.“ In: *Retina* 2010;30(9):1441-1454. DOI: 10.1097/IAE.0b013e3181ee5ce8

[19] Christine A. Curcio. “Soft Drusen in Age-Related Macular Degeneration: Biology and Targeting Via the Oil Spill Strategies“. In: *Invest. Ophthalmol. Vis. Sci.* 2018;59(4):AMD160-AMD181. DOI: 10.1167/iovs.18-24882

[20] Alessandro Rabiolo, Riccardo Sacconi, Maria Vittoria Cicinelli, Lea Querques, Francesco Bandello, Giuseppe Querques. "Spotlight on reticular pseudodrusen". In: *Clin Ophthalmol* 2017;11:1707–1718. DOI: 10.2147/OPTH.S130165

[21] Monika Fleckenstein, Steffen Schmitz-Valckenberg, Moritz Lindner, Athanasios Bezatis, Eva Becker, Rolf Fimmers, Frank G. Holz. "The 'Diffuse-Trickling' Fundus Autofluorescence Phenotype in Geographic Atrophy". In: *Investigative Ophthalmology & Visual Science* 2014;55:2911–2920. DOI:10.1167/iovs.13-13409

[22] Florian Alten. "Differenzialdiagnose makulärer Drusen". In: *Augenblickmal* 2016;1:2-3. URL: https://www.ukm.de/fileadmin/ukminternet/daten/kliniken/augenklinik/Newsletter/UKM_Augenklinik_Newsletter_2016_DRUCK.pdf

[23] Kaveh Abri Aghdam, Amelie Pielen, Carsten Framme, Bernd Junker. "Correlation Between Hyperreflective Foci and Clinical Outcomes in Neovascular Age-Related Macular Degeneration After Switching to Aflibercept". In: *Invest. Ophthalmol. Vis. Sci.* 2015;56(11):6448-6455. DOI: 10.1167/iovs.15-17338

[24] John A. Fieldling. "CHAPTER 47 - The eye and orbit". In: *Clinical Ultrasound (Third Edition)* 2011;-:938-964. ISBN: 9780702031311. DOI: 10.1016/B978-0-7020-3131-1.00047-X

[25] Joel Yap, Efrem D. Mandelcorn. "The Good and Bad of Retinal Fluid". In: *Retina Specialist* 30 Mar 2018. URL: https://www.retina-specialist.com/article/the-good-and-bad-of-retinal-fluid, letzter Aufruf am 10.03.2021

[26] Alfredo Garcia-Layana, Gianfranco Ciuffo, Javier Zarranz-Ventura, Aurora Alvarez-Vidal. "Optical Coherence Tomography in Age-related Macular Degeneration". In: *AMD Book*. URL: https://amdbook.org/content/optical-coherence-tomography-age-related-macular-degeneration, letzter Aufruf am 10.03.2021

[27] Ugo Introini, Giuseppe Casalino. "Serous PED". In: *AMD Book*. URL: https://amdbook.org/content/serous-ped-0, letzter Aufruf am 10.03.2021

[28] Fernanda Vaz, Maria Picoto. "Geographic Atrophy". In: *AMD Book*. URL: https://amdbook.org/content/geographic-atrophy-0, letzter Aufruf am 10.03.2021

[29] W. Roquet, F. Roudot-Thoraval, G. Coscas, G. Soubrane. "Clinical features of drusenoid pigment epithelial detachment in age related macular degeneration". In: *Br J Ophthalmol* 2004;88(5):638-642. DOI: 10.1136/bjo.2003.017632

[30] Min Seok Kim, Na-Kyung Ryoo, Kyu Hyung Park. "Laser and anti-vascular endothelial growth factor treatment for drusenoid pigment epithelial detachment in age-related macular degeneration". In: *Sci Rep* 2020;10:14370. DOI: 10.1038/s41598-020-71401-3

[31] Nataliya V. Pasyechnikova, Volodymyr A. Naumenko, Andrii R. Korol, Oleg S. Zadorozhnyy, Taras B. Kustrin, Illya O. Nasinnyk. "Serous Pigment Epithelium Detachment Associated with Age-Related Macular Degeneration: A Possible Treatment Approach". In: *Med Hypothesis Discov Innov Ophthalmol* 2012;1(4):72-75. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3939730/

[32] Janet S. Sunness, Joel Gonzalez-Baron, Carol A. Applegate, Neil M. Bressler, Yan Tian, Barbara Hawkins, Yolanda Barron, Akiva Bergman. "Enlargement of atrophy and visual acuity loss in the geographic atrophy form of age-related macular degeneration". In: *Ophthalmology* 1999;106(9):1768-1779. DOI: 10.1016/S0161-6420(99)90340-8

[33] Klaus Backhaus, Bernd Erichson, Rolf Weiber. "Fortgeschrittene Multivariate Analysemethoden". *3. Auflage*. Springer Gabler, 2015. ISBN: 978-3-662-46086-3. DOI: 10.1007/978-3-662-46087-0 2015 Fortgeschrittene Multivariate Analysemethoden Klaus Backhaus

[34] Alexander Bain, Henry S. King. "Mind and Body. The Theories of their Relation". In: *Journal of Mental Science* 1874;19(88):582-587. DOI: 10.1192/bjp.19.88.582

[35] "Was ist ein neuronales Netz?". In: *MathWorks*. URL: https://de.mathworks.com/discovery/neural-network.html, letzter Aufruf am 10.03.2021

[36] Sebastian Dörn. "Programmieren für Ingenieure und Naturwissenschaftler". Springer Vieweg, 2018. ISBN: 978-3-662-54303-0. DOI: 10.1007/978-3-662-54304-7

[37] Kunihiko Fukushima. "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position". In: *Biol. Cybernetics* 1980;36:193-202. DOI: 10.1007/BF00344251O

[38] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 1998;86(11):2278-2324. DOI: 10.1109/5.726791

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. "Mask R-CNN". In: *arXiv e-prints* 2017. arXiv:1703.06870v3[cs.CV]

[40] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *arXiv e-prints*. 2013. arXiv:1311.2524[cs.CV]

[41] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders. "Selective Search for Object Recognition". In: *Int J Comput Vis* 2013;104:154-171. DOI: 10.1007/s11263-013-0620-5

[42] M. N. Murty, Rashmi Raghava. "Support Vector Machines and Perceptrons". Springer. 2016. ISBN: 978-3-319-41062-3. DOI: 10.1007/978-3-319-41063-0

[43] Ross Girshick. "Fast R-CNN". In: *arXiv e-prints* 2015. arXiv:1504.08083[cs.CV]

[44] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *arXiv e-prints* 2015. arXiv:1506.01497[cs.CV]

[45] Jonathan Long, Evan Shelhamer, Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *arXiv e-prints* 2014. arXiv:1411.4038[cs.CV]

[46] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, Liang Lin. "Meta R-CNN : Towards General Solver for Instance-level Few-shot Learning". In: *arXiv e-prints* 2019. arXiv:1909.13032[cs.CV]

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. "Deep Residual Learning for Image Recognition". In: *arXiv e-prints* 2015. arXiv:1512.03385[cs.CV]

[48] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. "Feature Pyramid Networks for Object Detection". In: *arXiv e-prints* 2016. arXiv:1612.03144[cs.CV]

[49] Kai Ming Ting. "Confusion Matrix". In: *Encyclopedia of Machine Learning*. Springer 2010. ISBN: 978-0-387-34558-1. DOI: 10.1007/978-0-387-30164-8_157

[50] Kai Ming Ting. "Sensitivity and Specificity". In: *Encyclopedia of Machine Learning*. Springer 2010. ISBN: 978-0-387-34558-1. DOI: 10.1007/978-0-387-30164-8_752

[51] Stephane Fotso. "PySurvival: Open source package for Survival Analysis modeling". 2019. URL: https://www.pysurvival.io/

[52] Thriyambakam Krishnan. "Chapter 14: Survival Analysis". In: *Essentials of Business Analytics*. Springer 2019. ISBN: 978-3-319-68836-7. DOI: 10.1007/978-3-319-68837-4

[53] D. R. Cox. "Regression Models and Life-Tables". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 1972;34(2):187-220. JSTOR: www.jstor.org/stable/2985181

[54] Jared Katzman, Uri Shaham, Jonathan Bates, Alexander Cloninger, Tingting Jiang, Yuval Kluger, "DeepSurv: Personalized Treatment Recommender System Using A Cox Proportional Hazards Deep Neural Network". In: *arXiv e-prints* 2016. arXiv:1606.00931[stat.ML]

[55] Jared Katzman, Uri Shaham, Jonathan Bates, Alexander Cloninger, Tingting Jiang, Yuval Kluger. "DeepSurv: Personalized Treatment Recommender System Using A Cox Proportional Hazards Deep Neural Network". In: *BMC Med Res Methodol* 2018;18:24. DOI: 10.1186/s12874-018-0482-1

[56] Hajime Uno, Tianxi Cai, Michael J. Pencina, Ralph B. D'Agostino, L. J. Weib. "On the C-statistics for Evaluating Overall Adequacy of Risk Prediction Procedures with Censored Survival Data". In: *Stat Med* 2011;30(10):1105-1117. DOI: 10.1002/sim.4154

[57] Matthias Schmid, Marvin Wright, Andreas Ziegler. "On the use of Harrell's C for clinical risk prediction via random survival forests". In: *arXiv e-prints* 2015. arXiv:1507.03092[stat.ML]

[58] Jesse Islam. "Cox model: right-censored Brier score". 7 Feb 2021. URL: `https://www.jesseislam.com/post/brier-score/`, letzter Aufruf am 11.03.2021

[59] E. L. Kaplan, Paul Meier. "Nonparametric Estimation from Incomplete Observations". In: *Journal of the American Statistical Association* 1958;53(282),457-481. DOI: 10.1080/01621459.1958.10501452

[60] Karl Michael Ortmann. "Praktische Lebensversicherungsmathematik 2. Auflage". Springer Spektrum 2016. ISBN: 978-3-658-10199-2. DOI: 10.1007/978-3-658-10200-5

[61] E Graf, C Schmoor, W Sauerbrei, M Schumacher. "Assessment and comparison of prognostic classification schemes for survival data". In: *Stat Med* 1999;18(17-18):2529-45. DOI: 10.1002/(sici)1097-0258(19990915/30)18:17/18<2529::aid-sim274>3.0.co;2-5

[62] Thomas A Gerds, Martin Schumacher. "Consistent estimation of the expected Brier score in general survival models with right-censored event times". In: *Biom J* 2006;48(6):1029-40. DOI: 10.1002/bimj.200610301

[63] Belal Ahmed, T. Aaron Gulliver, Saif alZahir. "Image splicing detection using mask-RCNN". In: *Signal, Image and Video Processing* 2020;14:1035-1042. DOI: 10.1007/s11760-020-01636-0

[64] Richard Ernest Bellman. "Dynamic Programming". Courier Dover Publications 1957. ISBN: 0486428095

[65] Diederik P. Kingma, Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv e-prints* 2014. arXiv:1412.6980[cs.LG]

[66] Xavier Glorot, Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* 2010;9:249-256. URL: http://proceedings.mlr.press/v9/glorot10a.html

[67] "Wet Age-related Macular Degeneration". In: *AMDF*. URL: https://www.macular.org/wet-amd, letzter Aufruf am 14.03.2021

[68] Bradley Efron. "The Efficiency of Cox's Likelihood Function for Censored Data". In: *Journal of the American Statistical Association* 1977;72(359):557-565. DOI: 10.1080/01621459.1977.10480613

[69] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv e-prints* 2012. arXiv:1212.5701[cs.LG]

[70] John Duchi, Elad Hazan, Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *JMLR* 2011;12(61):2121-2159. URL: https://jmlr.org/papers/v12/duchi11a.html

[71] Geoffrey Hinton, Nitish Srivastava, Kevin Swersky. "Neural Networks for Machine Learning". 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[72] Marco Nassisi, Yue Shi, Wenying Fan1, Enrico Borrelli, Akihito Uji, Michael S. Ip, Srinivas R. Sadda. "Choriocapillaris impairment around the atrophic lesions in patients with geographic atrophy: a swept-source optical coherence tomography angiography study". In: *British Journal of Ophthalmology* 2019;103:911-917. DOI: 10.1136/bjophthalmol-2018-312643