WHZ Westsächsische Hochschule Zwickau
Hochschule für Mobilität

Masterarbeit

Planung und Entwicklung einer Pipeline für die Segmentierung von zerebralen Blutgefäßen in MRT-Bildsequenzen

Dullinger Julian

geboren am 20.06.1999 in Zwickau

Studiengang Informatik

Westsächsische Hochschule Zwickau
Fakultät Physikalische Technik/Informatik

Betreuer(in):	Prof. Dr. Silke Kolbig M. Sc. Christian Fiedler
Abgabetermin:	15.11.2023

Autorenreferat

Für die Behandlung der Parkinson-Krankheit wird die Methode der Tiefen Hirnstimulation immer häufiger verwendet. Dafür muss der Zugangsweg der Elektroden für den Eingriff unter Berücksichtigung von Risikobereichen festgelegt werden. Die Segmentierung und Darstellung dieser Risikobereiche ist ein Hauptbestandteil der präoperativen Planung.

Für die Westsächsische Hochschule Zwickau wird ein Demonstrator für die Darstellung dieser Risikobereiche in virtueller Realität entwickelt. Dieser benötigt Segmentierungen von Blutgefäßen aus MRT-Bildsequenzen als Eingabedaten.

In dieser Arbeit wird eine Pipeline geplant und entwickelt, welche Blutgefäße aus MRT-Bildsequenzen segmentieren kann und die Resultate dem Demonstrator zur Verfügung stellt.

Hierfür wurde eine umfangreiche Literaturrecherche durchgeführt, auf deren Basis die Anforderungen der Pipeline festgelegt wurden. Entsprechend der Anforderungen wurden die Verfahren UNet, BRAVE-Net, VMTK und Vesselness-Filter getestet und mit Metriken verglichen. Aus den daraus gewonnen Erkenntnissen wurde eine funktionsfähige Pipeline mit einem Convolutional Neural Network auf Basis der UNet-Architektur entwickelt und eine alternative Pipeline für die Verwendung von Vesselness-Filtern demonstriert.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AHD	Average Hausdorff Distance
CNN	Convolutional Neural Network
DSC	Dice Similarity Coefficient
FC	Fully connected
ReLU	Rectified linear unit

1 Einleitung

Die medizinische Bildgebung ist eine essentielle Grundlage für die Untersuchung und Behandlung von Patienten. Dabei ist ein zentraler Bestandteil der Forschung die Erleichterung der Verwendung der ausgegebenen Bilddaten.

Die Segmentierung ist dabei einer der wichtigsten Verarbeitungsschritte in der medizinischen Bildverarbeitung und ein Gebiet intensiver Forschung besonders im Hinblick auf die Automatisierung der Segmentierung.

1.1 Motivation

Bériault et al. [7] berichtet von der Behandlungsmethode Tiefe Hirnstimulation, die seit Jahren ein wichtiger Bestandteil der Behandlung der Parkinson-Krankheit geworden ist. Dabei werden Elektroden durch einen minimalinvasiven Eingriff in bestimmte Bereiche des Gehirns eingesetzt, wo sie durch die eingestellte Stimulation die Symptome der Parkinson-Krankheit vermindern sollen.

Es ist von entscheidender Bedeutung, in der präoperativen Planung eine geeignete, sichere Trajektorie für die Einführung der Elektroden zu finden. Dabei müssen verschiedene Bereiche des Gehirns gemieden werden, um das Risiko von Verletzungen zu minimieren. Eine wichtige Aufgabe ist die Markierung der Risikobereiche. Es müssen insbesondere Blutgefäße erkannt werden. Wegen ihrer komplexen Form ist diese Aufgabe nicht trivial und seit langer Zeit ein Gebiet intensiver Forschung. [7]

1.2 Problemstellung und Ziel der Arbeit

Für den Fachbereich Mathematik der Westsächsischen Hochschule Zwickau wird ein Demonstrator entwickelt, der die Darstellung von MRT-Bildsequenzen und segmentierten Risikobereichen in virtueller Realität ermöglichen soll. Dieser benötigt zur Eingabe die segmentierten Risikobereiche, wie die Blutgefäße des Gehirns.

Ziel dieser Arbeit ist es, eine Pipeline zu entwickeln und zu implementieren, die Blutgefäße des Gehirns in MRT-Bildsequenzen segmentiert und dem Demonstrator zur Verfügung stellt.

Dabei werden Bedingungen durch eine Recherche und eine Besprechung mit den Betreuern des Demonstrator-Projektes festgelegt, die die Pipeline erfüllen muss und dementsprechend bei der Auswahl von Verfahren und der Implementierung beachtet werden sollten.

Es werden Verfahren für die Segmentierung durch die Anwendung von Metriken miteinander verglichen, um die bestmögliche Lösung für diesen Anwendungsfall zu finden und auszuwählen.

Hierbei soll ein Überblick über mögliche Lösungsansätze, Anforderungen und Herausforderungen gegeben und eine finale Pipeline sowie ihr Aufbau dargestellt und erläutert werden.

1.3 Aufbau der Arbeit

Kapitel 1 dient der Einführung in das Thema und gibt einen Überblick über die Problemstellung, Ziele und den Aufbau der Arbeit.

Kapitel 2 befasst sich mit einer Recherche des Stands der Forschung und der Darstellung aller nötigen Grundlagen für die weiteren Kapitel.

Kapitel 3 definiert die genauen Bedingungen, die aus der Recherche des Stands der Forschung hervorgehen und legt fest, welche Teile benötigt und wie diese ausgewählt werden.

Kapitel 4 enthält die gesamte Darstellung der Ergebnisse, der finalen Pipeline und derer Entwicklung anhand der definierten Bedingungen.

In Kapitel 5 werden die Ergebnisse aus dem vorherigen Kapitel diskutiert. Getroffene Entscheidungen werden begründet, sowie Einblicke in aufgetretene Probleme und deren Lösungsansätze gegeben.

Zuletzt wird in Kapitel 6 eine Zusammenfassung der gesamten Arbeit präsentiert. Außerdem wird ein Ausblick bezüglich möglicher Erweiterungen und Verbesserungen der dargelegten Lösungen gegeben.

2 Grundlagen und Stand der Forschung

In diesem Kapitel wird der durch eine Literaturrecherche erfasste Stand der Forschung aufgezeigt, um so einen Überblick über verschiedene für die Arbeit relevante Teilbereiche zu erhalten. Dabei werden zuerst Segmentierungsverfahren und deren Kategorien vorgestellt, wonach eine Übersicht über Datensätze und Metriken folgt.

2.1 Segmentierungsverfahren

Segmentierung in der medizinischen Bildverarbeitung ist ein vielfältiges, ausführlich erforschtes Gebiet, welches weiterhin signifikante Forschungsfortschritte verzeichnet. Im Bereich der Blutgefäßsegmentierung haben sich einige Gruppen gebildet, in die sich die meisten Lösungen des Segmentierungsproblems eingliedern lassen [1].

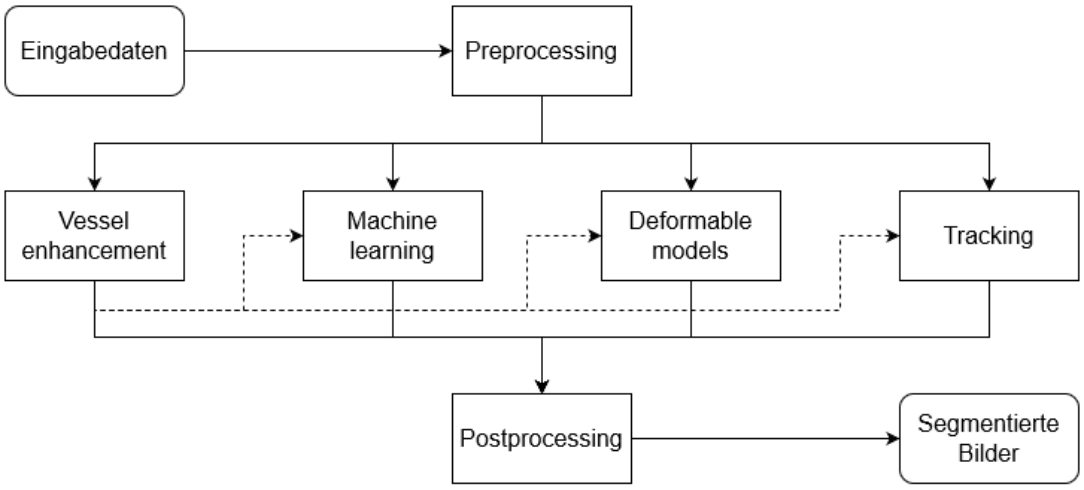


Abbildung 2.1: Vereinfachte Übersicht des Segmentierungs-Workflows aus Moccia et al. [1]

Die Segmentierung lässt sich im Allgemeinen in mehrere Schritte einteilen. Ein Eingabe-Bild wird optional zuerst durch unterschiedliche vorverarbeitende Schritte für das gewählte Segmentierungsverfahren angepasst, um bessere Ergebnisse in den nächsten Schritten

zu ermöglichen. Anschließend wird das eigentliche Segmentierungsverfahren durchgeführt. Hier sind vier Kategorien hervorzuheben: Vessel enhancement, Machine learning, Deformable models und Tracking. [1]

2.1.1 Preprocessing

Vor der Segmentierung sind vorbereitende Schritte verwendbar, um die Daten aufzubereiten und ein besseres Ergebnis der Segmentierung zu ermöglichen. Im Bereich der Segmentierung von medizinischen Bildgebungsverfahren sind besonders Schritte im Einsatz, die zu einer Verringerung von Artefakten oder Rauschen in den Ursprungsdaten führen.

Normalisierung

In verschiedenen Scans eines Datensatzes kann es dazu kommen, dass die einzelnen Scans einen unterschiedlichen Bereich an Intensitätswerten der Pixel aufweisen. Diese Unterschiede treten durch viele verschiedene Faktoren der eingesetzten Scanner auf und betreffen auch Aufnahmen eines Patienten die zu unterschiedlichen Zeiten aufgenommen wurden. Dadurch ist unklar, welche Bedeutung die genauen Intensitätswerte haben, wodurch bei der Segmentierung, je nach Intensitätsbereich, andere Ergebnisse erzeugt werden können. Die Normalisierung wird eingesetzt, um die Werte in einen festgelegten Bereich zu transformieren. Somit kann in allen Daten von einem ähnlichen Wertebereich ausgegangen werden. [1, 8]

Die grundlegendste Form der Normalisierung ist die lineare Transformation der Intensitäten in einen neuen Wertebereich. Hierbei werden die neuen Maximum- und Minimum-Werte manuell festgelegt, wonach alle Werte in diesen Bereich skaliert werden. Dabei können einzelne Ausreißer in den Werten dazu führen, dass die meisten Werte in einen kleinen Bereich verschoben werden, wodurch eine Unterscheidung erschwert wird. [9]

Dieses Problem verbessert die Z-Score Normalisierung (Standardisierung). Hier wird der Intensitätswert mit dem Mittelwert der Daten subtrahiert und durch die Standardabweichung dividiert. Die berechneten Werte geben hier die Entfernung zum Mittelwert

an, wobei dieser null betragen soll. Die Entfernung in positiver sowie negativer Richtung ergibt sich aus der Standardabweichung. [9]

Region of Interest

Es gibt viele Bereiche eines Bildes, die für die Segmentierung einzelner relevanter Bereiche zu Problemen führen können. Bei einem Segmentierungsproblem, wie es in dieser Arbeit vorliegt, sind Bereiche außerhalb des eigentlichen Gehirns irrelevant und können zu falschen Klassifizierungen führen. Hierbei ist besonders auf das Rauschen in den äußeren Bereichen, der Schädelknochen und weiteres Gewebe zu achten. Um eine Eingrenzung vorzunehmen, wird meist eine vollständige Entfernung des Schädelknochen (Skull-Stripping) vorgenommen. Hierfür existieren eine Vielzahl an verschiedenen Verfahren. Für die Umsetzung dieser Methoden kann es von Nöten sein, weitere Preprocessing Verfahren einzusetzen, da beispielsweise ein starkes Rauschen auch hier die Definition des relevanten Bereichs erschweren kann.

Bias field correction

Ein sogenanntes „Bias field signal“ ist ein Signal, welches ein Problem der MRT-Bildgebung ist. Dieses Signal stört die Aufnahme von MRT-Sequenzen und führt zu ungleichmäßigen Intensitäten in Bildsequenzen [10]. Zur Lösung von diesem Problem existieren verschiedene Algorithmen. Ein solcher Algorithmus ist der „nonparametric nonuniform intensity normalization“-Algorithmus (N3). Dieser Algorithmus ist weitverbreitet und wurde beispielsweise in ITK und als Vorbereitung für Skull-Stripping-Verfahren implementiert, da dieser Algorithmus keine Masken benötigt und so früh in einer Pipeline eingesetzt werden kann. [11]

Resampling und Patches

Das gesamte Volumen eines Scans direkt zu Segmentieren, kann eine rechenintensive Aufgabe sein. Deshalb sind einige Segmentierungsverfahren darauf spezialisiert, einzelne Bereiche der Bilder (Patches) zu segmentieren. Es gibt unterschiedliche Ansätze zur Auswahl solcher Patches. Es können Patches komplett zufällig oder nach bestimmten Kriterien ausgewählt werden. Beispielsweise kann eine Auswahl daran erfolgen, dass das Zentrum des Patches Teil einer festgelegten Maske sein muss. Dadurch kann sichergestellt

werden, dass bestimmte Bereiche definitiv in den ausgewählten Patches vorkommen. Des Weiteren können die Bilder komplett in Patches in Form einer Rasterung aufgeteilt werden.

Durch diese Aufteilung in Bereiche wird der Kontext der angrenzenden Bereiche verloren, was eine Verschlechterung der Segmentierung zur Folge haben kann. Dieser Kontextverlust kann zum einen dadurch behoben werden, dass ein größerer Bereich um die Patches herum als Kontext weitergegeben wird. Zum anderen kann durch ein Resampling der Voxelgröße ein größerer Bereich abgedeckt werden. Durch diese Vergrößerung wird allerdings die Auflösung der Patches verringert, wodurch Details, wie zum Beispiel kleine Arterien, verloren gehen können.

2.1.2 Vessel Enhancement

Verfahren der Kategorie „Vessel Enhancement“ versuchen die Blutgefäße deutlich von anderen Strukturen unterscheidbar zu machen, indem der Kontrast zu diesen erhöht wird. Damit kann durch eine Binarisierung des Bildes direkt eine Segmentierung vorgenommen werden. Ebenso wird „Vessel Enhancement“ als ein weiterer vorbereitender Schritt für die folgenden Segmentierungsverfahren genutzt. Im aktuellen Stand der Forschung werden die „Vessel Enhancement“-Verfahren meist als ein solcher vorausgehender Schritt verwendet. [1, 12]

Vesselness-Filter

Die bekanntesten Vertreter der „Vessel Enhancement “-Verfahren sind Vesselness-Filter. Darunter sind die Verfahren von Frangi et al. [12] und Sato et al. [13] prominente Ansätze.

Diese Ansätze berechnen die Wahrscheinlichkeit, dass ein Voxel zu einer röhrenförmigen Struktur gehört. Dazu nutzen sie die Hesse-Matrix, die zweite Ableitung der gegebenen Bilder, die durch die Evaluierung der Eigenwerte an allen Punkten geometrisch interpretiert werden kann. [2]

Die Eigenwerte repräsentieren hier die Veränderung der Intensität in die Richtung des dazugehörigen Eigenvektors. Damit kann eine Struktur anhand der Ausprägungen dieser

Eigenwerte definiert und erkannt werden. Krissian et al. [14] hat hierfür eine Definition einer möglichst perfekten Röhre beschrieben. Wie in Abbildung 2.2 zu sehen ist, entspricht eine perfekte Röhre, wie in diesem Beispiel ein Objekt, welches entlang der Z-Achse verläuft und die selbe Ausdehnung in X- und Y-Richtung hat.

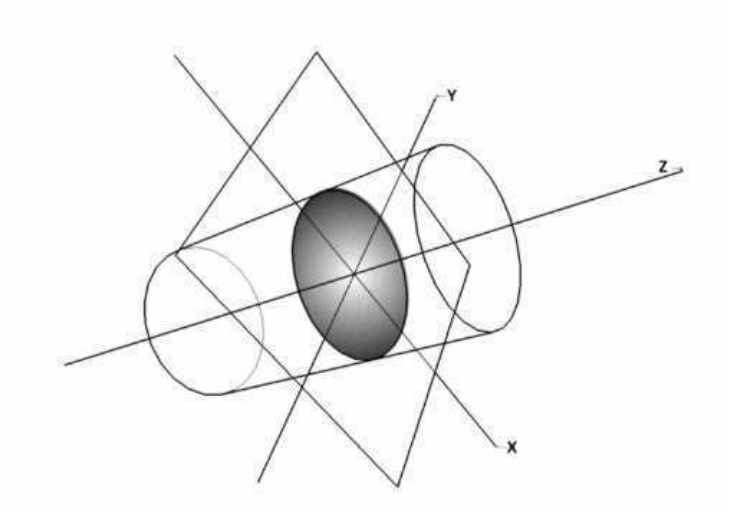


Abbildung 2.2: Darstellung einer Röhre mit Gauss-Profil aus Drechsler and Oyarzun Laura [2]

Die Eigenwerte sollten optimal dementsprechend wie folgt ausfallen:

$$\lambda_1 = 0, \lambda_2 < 0, \lambda_3 = \lambda_2 \quad [14] \tag{2.1}$$

λ_1 gibt hier an, dass der Eigenvektor entlang der Röhre verläuft und keine Veränderung in der Intensität auftritt. λ_2 und λ_3 sind für den Fall, dass eine helle Struktur auf dunklem Hintergrund erkannt werden soll, negativ (aufgrund des negativen Bereichs der zweiten Ableitung der Gauß-Funktion) und bei einer perfekten Röhre gleich.

Durch diese Definition kann nun mit Hilfe eines Filters die Wahrscheinlichkeit errechnet

werden. Sato et al. [13] hat die folgende Funktion aufgestellt:

$$f(\lambda_1, \lambda_c) = \begin{cases} e^{\frac{-\lambda_1^2}{2(\alpha_1 \lambda_c)^2}} \cdot \lambda_c & \lambda_1 \leq 0, \lambda_c \neq 0 \\ e^{\frac{-\lambda_1^2}{2(\alpha_2 \lambda_c)^2}} \cdot \lambda_c & \lambda_1 > 0, \lambda_c \neq 0 \\ 0 & \lambda_c = 0 \end{cases} \tag{2.2}$$

Hierbei ist $\alpha_1 < \alpha_2$ und $\lambda_c = \min(-\lambda_2, -\lambda_3)$.

Diese Funktion wird in den Modulen „Vesselness“ von MeVisLab und „Hessian3DToVesselnessMeasureImageFilter“ von ITK implementiert.

Frangi et al. [12] hat die folgende Funktion aufgestellt:

$$V_o(s) = \begin{cases} 0 & \text{wenn } \lambda_2 > 0 \text{ oder } \lambda_3 > 0 \\ (1 - \exp(-\frac{R_a^2}{2\alpha^2})) \exp(\frac{R_b^2}{2\beta^2})(1 - \exp(-\frac{s^2}{2c^2})) & \end{cases} \tag{2.3}$$

Hier sind α, β und c Schwellenwerte, die die Sensitivität des Filters bei den jeweiligen Teilberechnungen festlegen.

R_a und R_b berechnen die Abweichung von anderen Strukturen (Flächen respektive Kugeln).

$$R_a = \frac{|\lambda_2|}{|\lambda_3|} \tag{2.4}$$

$$R_b = \frac{|\lambda_1|}{\sqrt{|\lambda_2 \lambda_3|}} \tag{2.5}$$

2.1.3 Machine Learning

Machine-Learning-Ansätze haben in den letzten Jahren signifikante Popularität erlangt, da damit komplexe Muster und Merkmale erkannt werden können, die mit älteren Verfahren deutlich schwerer zu erfassen sind. Sie werden in die Kategorien „Unsupervised“, „Supervised“ und „Reinforcemnet Learning“ unterteilt, wobei die Kategorie der Reinforcement-Learning-Ansätze in diesem Bereich kaum angewandt und im Folgenden nicht weiter erwähnt wird. [1]

Unsupervised Machine Learning

Wie in Abbildung 2.3 zu sehen ist, werden Merkmale, wie Intensität oder Gradienten, aus dem zu segmentierenden Datensatz extrahiert. Das Modell wird danach anhand einer Funktion evaluiert und angepasst (Model tuning), um eine bestmögliche Unterscheidung zwischen klassifizierten Blutgefäßen und anderen Klassen zu finden. Das Besondere an diesen Ansätzen ist, dass für die Segmentierung grundsätzlich keine vergleichbaren Ground-Truth-Daten benötigt werden, die eine feste Definition der Klassifizierung vorgeben, sondern nur anhand der Merkmale der Daten und der ausgewählten Modelle eine Klassifizierung vorgenommen wird. Trotzdem wird ein Ground-Truth benötigt, um die Performanz zu testen und mit anderen Ansätzen zu vergleichen. [1, 15]

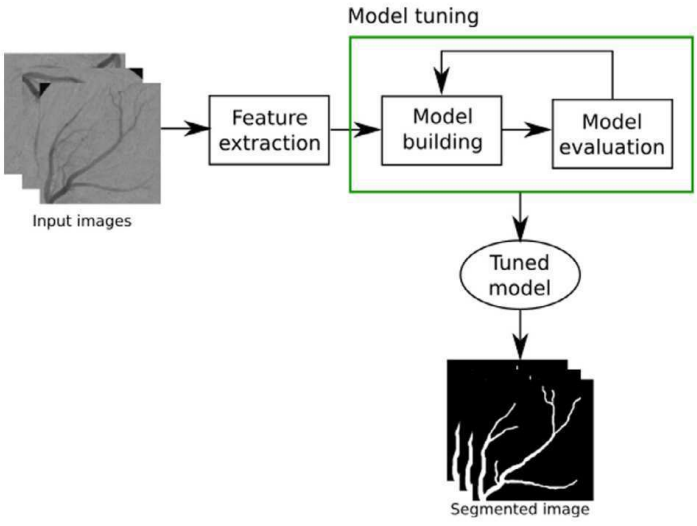


Abbildung 2.3: Ablauf eines Unsupervised Machine Learning Ansatzes aus [1]

Supervised Machine Learning

Diese Ansätze leiten die Regeln zur Klassifizierung von Trainingsdaten ab, in denen beschrieben ist, welche Eingabemerkmale einer Klassifizierung als Blutgefäß entsprechen. In Abbildung 2.4 ist der Ablauf dargestellt und wird im Folgenden erläutert. Es wird ein Goldstandard für den Trainingsdatensatz benötigt, der klar definiert, welche Pixel ein Blutgefäß darstellen und welche nicht. Des Weiteren werden im Schritt „Feature extraction“ Merkmale, wie zum Beispiel die Intensität oder Farbe, aus dem Trainingsdatensatz entnommen, mit deren Hilfe das Modell die Unterscheidung erlernen soll. Danach wird das ausgewählte Modell anhand dieser Merkmale mit den gelabelten Goldstandard-Daten trainiert. Nach dem Training und einer Evaluierung auf einem Teil des Datensatzes, kann das trainierte Modell auf einem neuen unbekannten Datensatz angewendet werden. Dabei werden die gleichen Merkmale wie während dem Training extrahiert. Danach klassifiziert das Modell anhand der während des Trainings erlernten Regeln den neuen Datensatz selbstständig. [1, 8]

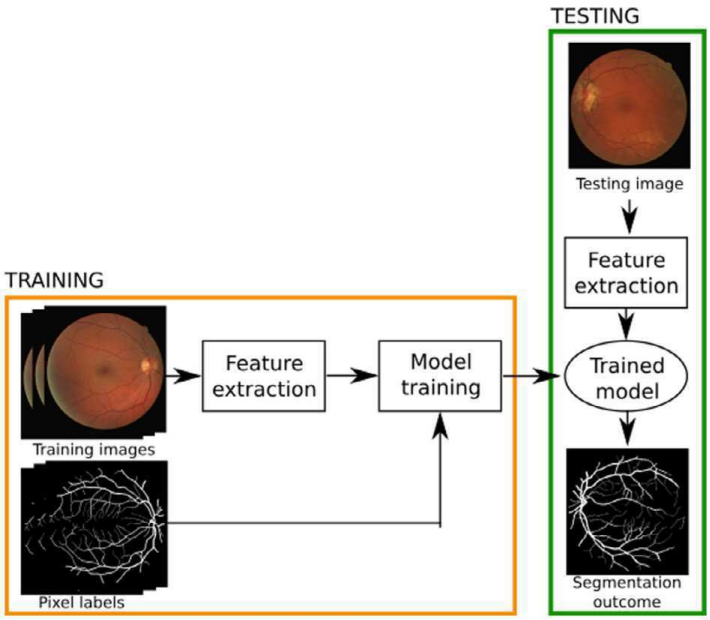


Abbildung 2.4: Ablauf eines supervised machine learning Ansatzes aus [1]

Convolutional Neural Network

Convolutional Neural Networks sind Neuronale Netze, die auf Bilder als Eingabedaten spezialisiert sind und häufig in der Bildsegmentierung eingesetzt werden. CNN[1]s bestehen typischer weise aus drei Arten von Schichten: Convolutional-Schicht, Pooling-Schicht und FC (Fully connected)-Schicht. In einer Convolutional-Schicht wird eine Faltungsoperation durchgeführt. Dabei wird ein Filterkern (Kernel) definiert, der über die Eingabedaten gelegt und verschoben wird. Ein Kernel besteht aus einer Matrix $\boldsymbol{M}(m \times k)$. In einer Filterschicht werden meist mehrere Filterkerne parallel angewendet. Wie in Abbildung 2.5 zu sehen ist, werden in dieser Matrix Koeffizienten definiert, mit denen die Eingabewerte an den jeweiligen Positionen multipliziert und miteinander addiert werden. Der daraus berechnete Wert ist der Ausgabewert für diesen Bereich, welcher in der Output-Feature-Map eingetragen wird. Je nach Größe und Schrittweite der Filterkerne, wird die Ausgabe einen geringen Verlust der Größe verzeichnen, da die Randbereiche selbst keine Position in der Output-Feature-Map erhalten und nur zur Berechnung im angrenzenden Bereich genutzt werden. Die Koeffizienten der Filterkerne werden nicht vorgegeben, sondern durch das Netz trainiert und immer wieder angepasst. [1, 16]

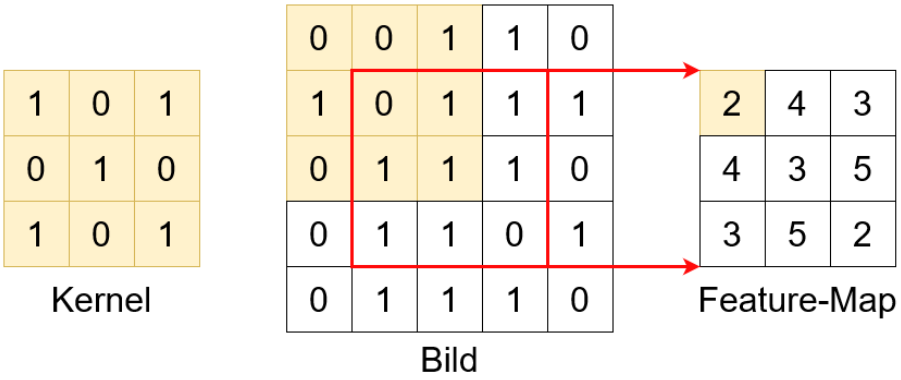


Abbildung 2.5: Beispiel einer Convolutional-Schicht

Die Pooling-Schicht reduziert die räumliche Größe der Eingabedaten, wodurch die benötigte Rechenleistung und Anzahl an Parametern gesenkt werden. Es wird ebenfalls ein Kernel definiert, der über die Eingabedaten geschoben wird. In dieser Schicht wird entweder das Maximum (Max Pooling) oder der Durchschnitt (Average Pooling) des durch den Kernel abgedeckten Bereichs ausgegeben. Max Pooling wird häufig eingesetzt, da nur

[1] Convolutional Neural Network

die höchste Aktivierung beibehalten und dadurch der Einfluss durch Rauschen reduziert wird. Wie in Abbildung 2.6 dargestellt ist, verringert sich die Menge an Parametern durch das Pooling je nach Größe des Kernel und der gewählten Schrittlänge. Meist wird eine Größe von 2×2 eingesetzt. [16]

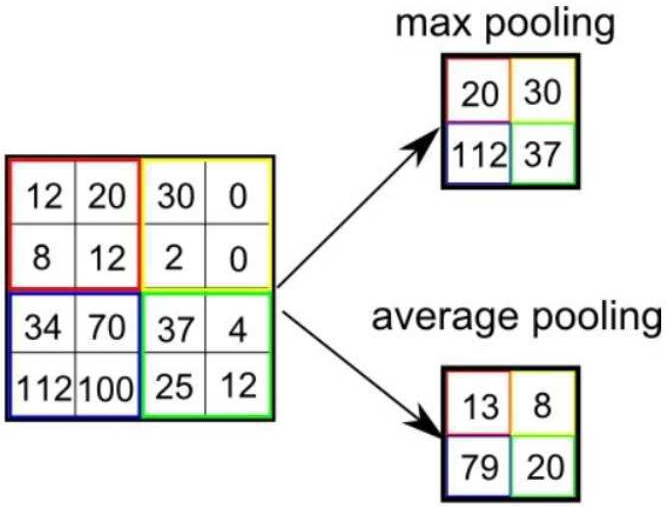


Abbildung 2.6: Beispiel einer Pooling-Schicht mit den unterschiedlichen Arten von Pooling aus [3]

Die FC[2]-Schicht versucht anhand der Aktivierungen in der vorherigen Schicht wie in klassischen neuralen Netzen eine Klassifizierung vorzunehmen. Wie am Namen erkennbar, sind hier alle Neuronen einer Schicht mit allen Neuronen der vorherigen Schicht verknüpft. [16]

U-Net

U-Net ist ein Convolutional Neural Network, welches an der Universität Freiburg für Segmentierungsprobleme der Biomedizin entwickelt wurde. Das Netzwerk wurde so konzipiert, dass es mit wenigen Daten trainiert werden kann, da in der Biomedizin häufig keine großen Datensätze mit Ground-Truth-Label verfügbar sind. In Abbildung 2.7 ist die U-Net-Architektur abgebildet. [4]

Diese Architektur wird als Encoder-Decoder-Architektur bezeichnet. Der linke Teil der Architektur (Encoder) fasst die Daten über mehrere Ebenen weiter zusammen. In jeder

[2]Fully connected

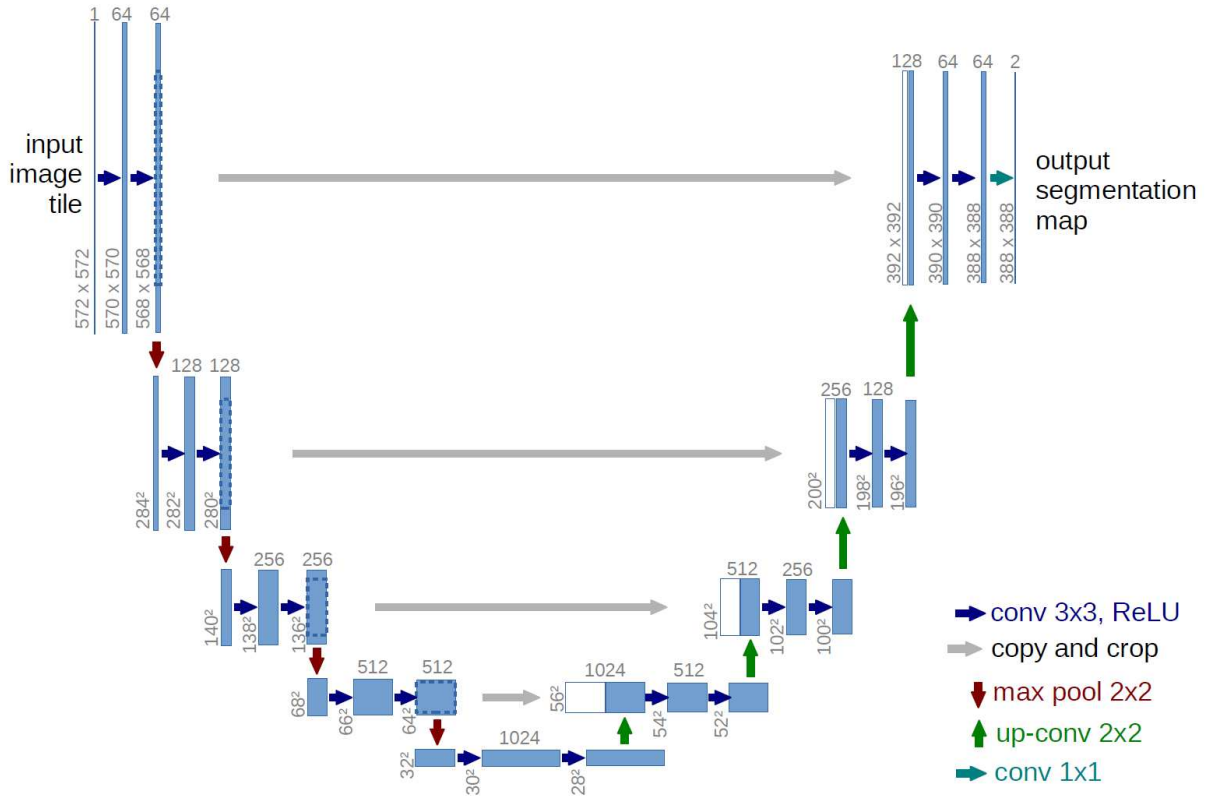


Abbildung 2.7: Die U-Net-Architektur aus [4]

Ebene der Architektur werden zwei Convolutional-Schichten verwendet, jeweils mit einer ReLU[3] Aktivierungsfunktion. Diese Funktion $f(z)$ verhindert negative Ausgabewerte und ist wie folgt definiert:

$$f(z) = \max(0, z) \tag{2.6}$$

Darauf folgt jeweils eine Max-Pool-Schicht mit einem 2×2 Kernel und einer Schrittlänge von 2, wodurch eine Reduzierung der Größe um 50% erreicht wird. Zu beachten ist, dass bei den Convolutional-Schichten ein 3×3 Kernel verwendet und kein Padding genutzt wird, was ein Verlust der Randpixel zur Folge hat. Der rechte Teil der Architektur (Decoder) besteht pro Ebene aus jeweils einem Up-Sampling-Schritt mit Hilfe einer „Transpose Convolution “ bei der, wie in Abbildung 2.8 zu sehen ist, eine eingegebene Feature-Map

[3]Rectified linear unit

mit Hilfe eines Kernel vergrößert wird. Dabei wird jeder Wert der Eingabe mit jedem Wert des Kernel multipliziert und an die jeweilige Position des verwendeten Kernel-Wert in der neuen Feature-Map geschrieben. Danach verschiebt sich der Kernel um die definierte Schrittlänge und wiederholt die Berechnung mit dem nächsten Eingabewert. Bei einer Überschneidung werden die Werte eines Feldes addiert. [4, 16]

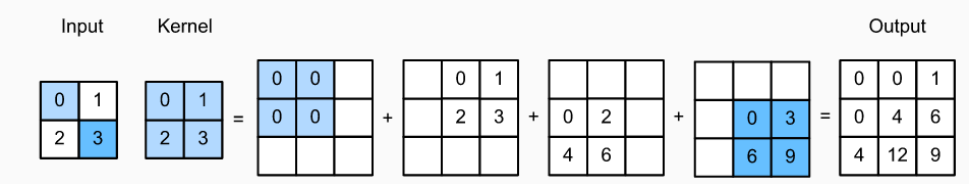


Abbildung 2.8: Beispiel einer Transpose-Convolution-Schicht aus [5]

Nach jedem Up-Sampling-Schritt folgen weitere Convolutional-Schichten mit ReLU wie im linken Teil der Architektur. Zu beachten ist, dass die Feature-Map aus dem Encoder-Teil der Architektur kopiert und in einem Concatenate-Schritt verwendet wird. Die Feature-Map muss dafür verkleinert werden, da in jeder Convolutional-Schicht die Randpixel entfallen und so die Größe im Decoder-Teil der Architektur kleiner ist. Am Ende wird eine Convolutional-Schicht mit einem 1×1 Kernel verwendet. Damit wird jeder einzelne Feature-Vektor der letzten Feature-Map auf die gewünschte Anzahl an Klassen gemappt. [16]

BRAVE-NET

Eine häufig zu findende Weiterentwicklung der grundlegenden U-Net-Struktur ist die BRAVE-NET-Architektur. Bei dieser Architektur wird nicht auf zweidimensionale komplette Schnittbilder als Eingabe gesetzt, sondern jeweils kleine dreidimensionale Ausschnitte des Volumen. Dies begründen die Autoren damit, dass die Segmentierung eines ganzen Volumen viele Ressourcen verbraucht. Des Weiteren soll der Anteil der zu segmentierenden erkennbaren Arterien nur 1,5% des gesamten Volumen des Gehirns betragen. [6]

Die Aufteilung der Eingabedaten in kleine Ausschnitte ist der Grund für die erste Erweiterung der U-Net-Architektur. Durch diese Zerstückelung fehlt dem Modell der räumliche Kontext eines gesamten Volumen. Ebenfalls soll die U-Net-Architektur Schwierigkeiten

haben, kleine Blutgefäße zu erkennen, die sich meist nahe der Schädeldecke befinden. [6]

Hierfür wird der Architektur, wie in Abbildung 2.9 zu sehen ist, eine Kontextaggregation auf der Encoder-Seite hinzugefügt. Hierbei wird ein größerer Kontextbereich um den eigentlichen Kern des Ausschnitts als Eingabe genutzt. Dieser Ausschnitt wird mit einem Pooling des Durchschnitt auf die räumliche Größe des eigentlichen Bereichs reduziert, aber besitzt dadurch eine geringere Auflösung als dieser, um Details zu vernachlässigen und nur den größeren Kontext zu betrachten. Dieser Kontext wird in jeder Schicht wie der normale Pfad durch die gleichen Schichten angepasst. In jeder Ebene werden die Ergebnisse beider Pfade durch einen Concatenate-Schritt zusammengeführt und der jeweiligen Eingabe der Decoder-Seite angehangen, sodass der Kontext erhalten bleibt. [6]

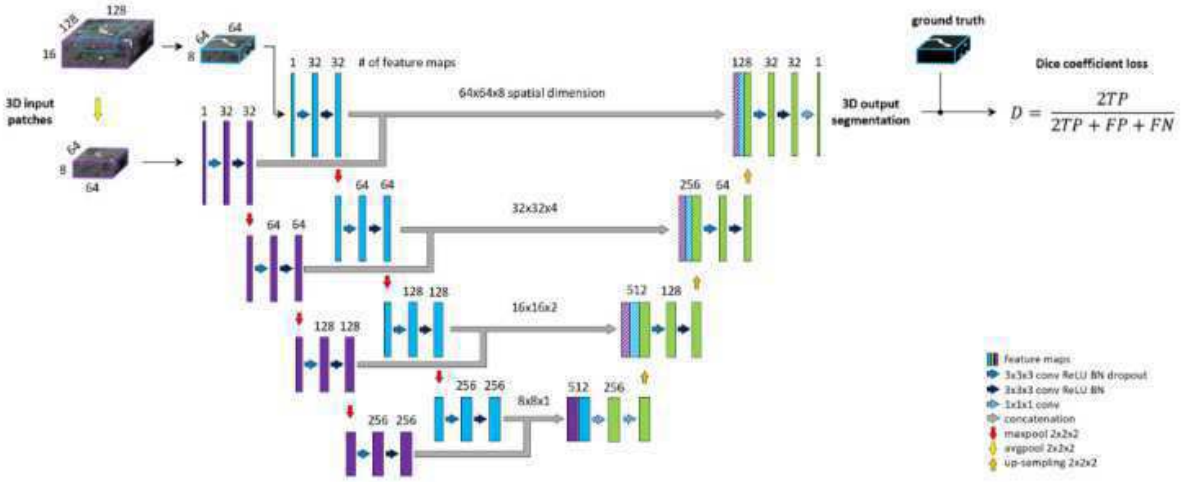


Abbildung 2.9: U-Net Architektur erweitert um eine Kontextaggregation aus [6]

Als zweites wird bei dieser Architektur die Methode der Deep Supervision eingesetzt. Diese wird bei Neuralen Netzen eingesetzt, um die Ergebnisse der Netzwerke zu verbessern, da es beispielsweise bei dem "Vanishing gradient problem"hilft. Dieses Problem entsteht bei dem Training von Neuralen Netzen. Dabei werden die verwendeten Gradienten als Merkmale in den Netzen so klein, dass sie so gut wie verschwinden. Das kann dazu führen, dass das Netz sich nur noch schlecht anpassen kann. Dadurch können die Netze in ihrem Training stark eingeschränkt und die Ergebnisse verschlechtert werden. [6, 17]

Hierbei wird eine Ausgabe von jeder Ebene einzeln in der Verlustfunktion berücksichtigt. Dabei gibt es viele verschiedene Möglichkeiten eine solche Supervision zu implementieren. In Abbildung 2.10 ist die gesamte BRAVE-NET-Architektur mit der Erweiterung um die Deep Supervision zu sehen.

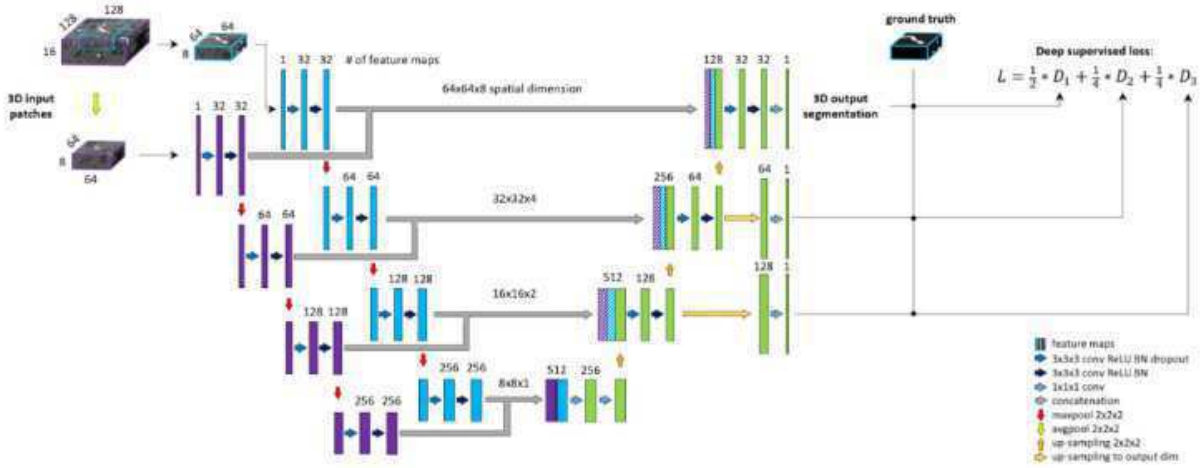


Abbildung 2.10: BRAVE-NET Architektur mit Kontextaggregation und Deep Supervision aus [6]

2.1.4 Deformable Models

Ansätze der Kategorie „Deformable Models“ segmentieren Bilder basierend auf Formen wie Kurven oder Oberflächen, die durch Deformation an Bildinhalte angepasst werden. Die Kräfte werden genutzt, um die gewählte Form zu den Grenzen der Blutgefäße zu bewegen und die Form während dieser Annäherung zu glätten, bis ein Kräftegleichgewicht entsteht. Deformierbare Modelle können in verschiedene Kategorien unterteilt werden. Die Modelle können in kantenbasierte Modelle und regionsbasierte Modelle unterteilt werden. Kantenbasierte Modelle werden wiederum in parametrische und geometrische Modelle unterteilt. [1, 18]

Kantenbasierte parametrische Modelle

Parametrische Modelle werden durch eine Menge an Parametern definiert, die die internen und externen Kräfte beeinflussen. Die Parameter der internen Kraft beeinflussen die Elastizität und die Resistenz gegen eine Biegung der Form während sich die Parameter der

externen Kraft je nach Ansatz unterscheiden. Laut Moccia et al., ist eine schlechte Anpassung an die sich verändernde Topologie der Blutgefäße die größte Limitierung dieser Modelle, aber bietet durch Nutzung von Parametern eine einfache Formulierung und schnelle Konvergenz, wodurch Berechnungskosten minimiert werden. [1, 18]

Kantenbasierte geometrische Modelle

Geometrische Modelle werden durch Punkte, Kurven und Oberflächen repräsentiert. Sie basieren auf der „curve-evolution theory“ und implementieren die Evolution durch die Level-Set-Methode. Diese Methode definiert eine Funktion, die kontrolliert durch eine Geschwindigkeitsfunktion, sich entwickelt und dadurch an die Kanten der ROI annähert. [1, 18]

Regionsbasierte Modelle

Regionsbasierte Modelle teilen Bilder in homogene Regionen auf, die ähnliche Eigenschaften aufweisen. Sie haben einige Vorteile gegenüber anderen deformierbaren Modellen, da sie durch den Einsatz von Constraints, basierend auf den Eingabedaten, robust gegenüber Schwankungen in Intensität und gegenüber Rauschen sind. In anderen Ansätzen wird durch diese Schwankungen und Rauschen das „boundary leakage“-Problem ausgelöst, bei dem die Segmentierungsgrenzen nicht mit dem tatsächlichen Objekt übereinstimmen und damit zu falscher Klassifizierung führen. [1, 18]

2.1.5 Tracking

Die Algorithmen der Kategorie „Tracking“ identifizieren und verfolgen Blutgefäße von einem Startpunkt über eine gewisse Strecke. Sie bestehen meistens aus der Definition von Seed-Punkten und einem Wachstumsprozess, der durch Constraints geleitet wird, die aus den Bildern abgeleitet werden. Diese Algorithmen können durch die unterschiedlichen Definitionen von Constraints in die Kategorien „Model-based“ und „minimum cost path“ unterteilt werden. [1, 14]

Model-based

Modell-basierte Tracking-Algorithmen versuchen ein Objekt zu verfolgen, indem ein mathematisches Modell an dieses angepasst wird. Zu Beginn muss ein Modell ausgewählt werden. Meistens werden einfache geometrische Formen gewählt. Für die Segmentierung von Blutgefäßen eignen sich dazu runde oder elliptische Zylinder. Es können aber auch komplexere Modelle genutzt werden, wie zum Beispiel ein deformierbares Modell. Das Modell wird initialisiert, indem es entweder automatisch oder manuell an das zu verfolgende Objekt positioniert und orientiert wird. In der folgenden Phase wird das Objekt über die einzelnen Slides der Aufnahme verfolgt, indem lokal die Position und Orientierung des Modells berechnet wird. Es werden verschiedene Eigenschaften aus den Bildern genutzt, um die beste Übereinstimmung des Modells mit dem Bild zu finden. Die Parameter des Modells werden danach angepasst, um auf die neue Erscheinung des Objektes zu berücksichtigen. Eine Schwäche dieser Algorithmen ist die verfrühte Terminierung der Verfolgung aufgrund von Inhomogenität der Intensität, Rauschen oder das Scheitern der Erkennung der vollständigen Gefäßstruktur durch eine vorhandene Pathologie. [1, 14]

Minimum cost path

Algorithmen dieser Kategorie basieren auf den Graphenalgorithmen und versuchen einen Pfad zwischen zwei Knoten zu finden der eine Energiefunktion minimiert beziehungsweise das kleinstmögliche Gewicht aufweist. Die Kosten eines Pfades werden meist durch bestimmte Eigenschaften der Bilder festgelegt, wie zum Beispiel die Größe des Gradienten zwischen zwei Pixeln. [1, 19]

2.2 Datensätze und Metriken

In Moccia et al. wurde gezeigt, dass in vielen verschiedenen Bereichen der Blutgefäßsegmentierung eine große Menge an Datensätzen und Metriken eingesetzt werden. Es gibt besonders in der Segmentierung von Blutgefäßen der Retina eine signifikante Menge an öffentlichen Datensätzen, während es in anderen Bereichen nur vereinzelte Datensätze gibt. Besonders im Bereich der zerebrovaskulären Segmentierung ist die Verfügbarkeit von Daten stark begrenzt [1]. Die großen Datensätze „1000Plus “, „PEGASUS“ und „7UP“ unterliegen strengen Datenschutzrichtlinien und können nicht öffentlich geteilt werden [6]. Der ebenfalls große Datensatz „IXI Dataset"“ [20] bietet knapp 600 MRA-Datensätze, die öffentlich zugänglich sind.

Es gibt viele verschiedene Metriken, die zu der Evaluierung der Performanz von Verfahren zur Lösung von semantischen Segmentierungsproblemen genutzt werden können. Die Performanz bezieht sich dabei auf die Similarität und korrekten Zuordnung der Pixel basierend auf einem Vergleichswert eines Ground-Truth. Diese korrekte Zuordnung kann durch die Einteilung der Pixel in die vier Bereiche einer Confusion Matrix (True Positive, False Positive, True Negative, False Negative) definiert werden. Im Bereich der zerebralen Blutgefäßsegmentierung gibt es keinen festen Standard für die Evaluierung von Ergebnissen, wodurch jede Veröffentlichung unterschiedliche Metriken einsetzt. Im Vergleich von mehreren Veröffentlichungen hat Goni et al. [8] dargestellt, dass eine Vielzahl der Verfahren mit den Metriken DSC[4] und Sensitivität evaluiert werden. [8]

[4]Dice Similarity Coefficient

Metrics	Expression
Sensitivity	$\frac{TP}{TP + FN}$
Specificity	$\frac{TN}{TN + FP}$
Precision	$\frac{TP}{TP + FP}$
Dice Similarity Coefficient	$\frac{2 \times TP}{FP + FN + (2 \times TP)}$
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
False Positive Rate	$\frac{FP}{TN + FP}$
Average Hausdorff Distance	$\left(\frac{1}{X}\sum_{x \in X, y \in Y} \min d(x,y) + \frac{1}{Y}\sum_{x \in X, y \in Y} \min d(x,y)\right)$

Abbildung 2.11: Liste häufig verwendeter Metriken aus [1]

Des Weiteren hat Aydin et al. [21] in einer Untersuchung die Performanz verschiedener Metriken bei zerebraler Blutgefäßsegmentierungen bewertet. Sie haben aus den Ergebnissen geschlussfolgert, dass Metriken basierend auf der AHD[5] standardmäßig genutzt werden sollten. Während der DSC ebenfalls eine gute Performanz aufgezeigt hat, wurde die Sensitivität in allen Ranglisten auf dem letzten Platz geführt. In der folgenden Abbildung 2.11 sind die genannten Metriken und einige Weitere mit den jeweiligen Formeln aufgeführt.

[5] Average Hausdorff Distance

3 Methodik

Das Kapitel Methodik klärt über die wissenschaftliche Vorgehensweise bei der Bearbeitung und die festzulegenden Voraussetzungen auf, bevor die Auswahl aller benötigten Bestandteile getroffen und mit den festgelegten Voraussetzungen begründet wird.

3.1 Erläuterung der Vorgehensweise

Um die Problemstellung der vorliegenden Abschlussarbeit zu lösen, wird zu Beginn der Stand der Forschung erhoben. Dazu wird eine Literaturrecherche durchgeführt. Durch diese Recherche werden die einzelnen benötigten Bestandteile erfasst, die im Anschluss weiter analysiert werden. Aus diesem Stand der Forschung ergeben sich Anforderungen, die nachfolgend für jeden Bestandteil definiert und in den weiteren Recherchen berücksichtigt werden.

Für die Umsetzung wird mindestens ein Datensatz eines MRA-Scan benötigt. Dieses bildgebende Verfahren wird verwendet, da es Blutgefäße visuell gut unterscheidbar von anderen Bereichen des Gehirns darstellt. Die untersuchten Lösungsansätze setzen einen solchen Datensatz voraus, um eine Segmentierung zu erstellen. Dabei ist ein vollständiger Scan des Gehirns eine Voraussetzung, da nur so eine möglichst vollständige Segmentierung der Blutgefäße möglich ist. Des Weiteren muss ein Ground-Truth-Datensatz verfügbar sein der eine schichtweise Segmentierung des MRA-Scan beinhaltet. Diese Segmentierung wird als Vergleich für alle Ergebnisse der Experimente genutzt und ist ebenfalls eine Voraussetzung als zusätzliche Eingabedaten für bestimmte Lösungsansätze. Hierfür wird eine weitere Recherche durchgeführt, um Datensätze zu erfassen, die aufgrund der genannten Anforderungen in diesem Projekt verwendet werden können. Mit Hilfe des Programms „MeVisLab“ werden alle Datensätze analysiert und anhand der genannten Kriterien eine Auswahl getroffen.

Zur Lösung des Segmentierungsproblems werden verschiedene Ansätze betrachtet und basierend auf der Verfügbarkeit von Quellcode und Beispielprojekten ausgewählt. Jedes

Projekt mit diesen Kriterien wird auf Funktionsfähigkeit geprüft. Funktionsfähige Projekte werden in Experimenten mit den zur Verfügung stehenden Datensätzen verwendet. Dabei entstehen neue Segmentierungen der Datensätze. Diese Daten müssen analysiert und mit dem Ground-Truth verglichen werden, um die Segmentierleistungen zu bewerten.

Um die Ergebnisse der einzelnen Experimente möglichst objektiv miteinander zu vergleichen, werden Metriken benötigt. Die Auswahl der Metriken basiert auf der Popularität im aktuellen Stand der Forschung und der durch die Forschung untersuchten Performanz der verschiedenen Metriken bei der zerebralen Blutgefäßsegmentierung. Mit Hilfe der gewählten Metriken werden alle Ergebnisse der Experimente analysiert und verglichen, um zu entscheiden, welche möglichen Lösungsansätze für diese Problemstellung verwendet werden sollen.

3.2 Auswahl der Datensätze, Verfahren und Metriken

Datensätze Die großen Datensätze „1000Plus “, „PEGASUS“ und „7UP“ stehen nicht öffentlich zur Verfügung und können deshalb nicht verwendet werden. Der Datensatz „IXI Dataset“ kann in dieser Arbeit nicht berücksichtigt werden, da wie in Abbildung 3.1 kein vollständiger Scan des Gehirns vorliegt und dementsprechend keine vollständige Segmentierung der Blutgefäße möglich ist. Ebenfalls fehlt bei allen der Zugang zu einem Ground-Truth-Datensatz, der für den Vergleich der Verfahren benötigt wird.

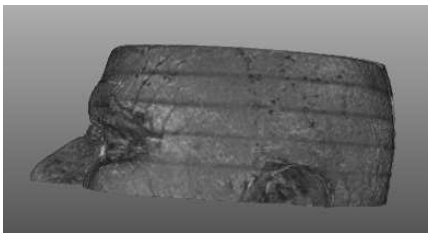


Abbildung 3.1: Unvollständiger Scan des IXI-Datensatzes

Der Datensatz „Designed Database of MR Brain Images of Healthy Volunteers “ [22] der durch das CASILab der University of North Carolina zur Verfügung gestellt wurde beinhaltet vollständige MRA-Scans der Gehirne von etwa 100 gesunden Testpersonen, aber ebenfalls keine Ground-Truth Segmentierungen. Jedoch hat Hilbert et al. [23] im Jahr 2020 für die Entwicklung des „BRAVE-NET“-Segmentierungsverfahren einen Ground-Truth-Datensatz für 20 Testpersonen aus den verschiedenen Altersgruppen erstellt und veröffentlicht. Nach intensiver Recherche wurde sich für die Nutzung in dieser Arbeit, den Anforderungen an einen vollständigen MRA-Scan des Gehirns und an einen vorhandenen Ground-Truth-Datensatz entsprechend, für den Datensatz „Designed Database of MR Brain Images of Healthy Volunteers “ entschieden.

Verfahren Die Verfahren werden anhand der festgelegten Voraussetzungen ausgewählt und in diesem Abschnitt kurz beschrieben. Eine genaue Beschreibung der Durchführung wird in Kapitel 4 vorgenommen.

U-Net Das U-Net wird als ein Vertreter der Machine Learning-Verfahren ausgewählt, da es ein etabliertes Verfahren im Bereich der Segmentierung ist und durch Beispielprojekte und Quellcode in relativ kurzer Zeit ein erstes Modell erstellbar ist. Hierzu wird die in Kapitel 2 behandelte Architektur mit Hilfe der Keras API von Tensorflow aufgebaut und Anhand der vorhandenen Daten von Grund auf trainiert und getestet.

VMTK Das Vascular Modeling Toolkit ist eine Sammlung von Bibliotheken und Werkzeugen für verschiedene Bildverarbeitungsaufgaben für die Verwendung bei Blutgefäßen. Es enthält eine Level-Set-Segmentierung, ein Verfahren der Kategorie der Deformable Models. Dazu bietet VMTK eine Reihe von Tutorials und eine Dokumentation zur Verwendung der einzelnen Werkzeuge an.

BRAVENET Das BRAVE-NET wird als ein weiterer Vertreter der Machine Learning-Verfahren ausgewählt, da es eine Erweiterung der bereits verwendeten U-Net-Architektur ist, aber auch einen anderen Ansatz bezüglich der Eingabedaten verfolgt und so eine deutlich andere Vorgehensweise erfordert.

Metriken Aus Kapitel 2 ergibt sich die Auswahl der Metriken für dieses Projekt. Aydin et al. empfehlen Metriken, die auf AHD basieren. In einer weiteren Veröffentlichung wird von der Nutzung der AHD selbst abgeraten und die Balanced Average Hausdorff Distance empfohlen, da hier ein Fehler der AHD verringert wird [18]. Aus diesen Empfehlungen und der Häufigkeit der Verwendung und Bewertung der Metriken im Stand der Forschung ergibt sich, dass die Metriken Balanced Average Hausdorff Distance und der DSC in dieser Arbeit verwendet werden sollten. Aufgrund der Möglichkeiten in der Implementation der Metriken wird jedoch der AHD verwendet.

4 Ergebnisse

In diesem Kapitel werden die Tests der ausgewählten Verfahren beschrieben und die daraus resultierten Ergebnisse vorgestellt. Die auf Grundlage dieser Ergebnisse erstellten Pipeline wird im Anschluss vorgestellt.

4.1 Anwendung der Verfahren

Die in Kapitel 3 ausgewählten Verfahren werden durch eine Reihe von Tests untersucht und deren Ergebnisse dargestellt. Dabei wird auch der Zusammenhang mit vorverarbeitenden Schritten aufgezeigt.

4.1.1 Preprocessing

Da in dem verwendeten Datensatz ein starkes Rauschen vorhanden ist, muss vor einem Skull-Stripping dieses Rauschen verringert werden. Dazu wird eine Normalisierung von ITK verwendet. Hierbei wird die Standardisierung (Z-Score) gefolgt von einer Skalierung der Werte vorgenommen. Das daraus resultierende Ergebnis setzt die meisten Werte außerhalb des Schädel auf null, wodurch das Rauschen stark vermindert wird, wie in den Abbildungen 4.1 zu sehen ist.

Anschließend wird der Schädelknochen samt umliegendem Gewebe durch das ROBEX-Verfahren entfernt. Dieses Verfahren hat eine einfach zu verwendende Implementierung unter Windows und enthält zusätzlich weitere Preprocessing-Schritte. Durch ein Resampling werden die Voxel hier vergrößert, wodurch leichte Verlusste von Details entstehen. Danach findet eine grobe „Bias field correction“ statt durch eine Implementation des bereits erwähnten 3N-Algorithmus. Die Ergebnisse dieses Skull-Stripping sind je nach verwendeter Bildsequenz unterschiedlich gut. In Abbildung 4.2 ist eine vollständige Entfernung des Schädelknochen zu sehen. Hier sind vereinzelt Ränder des Gehirns entfernt, wodurch einige wenige Blutgefäße nicht vorhanden sind. In anderen Datensätzen, wie in

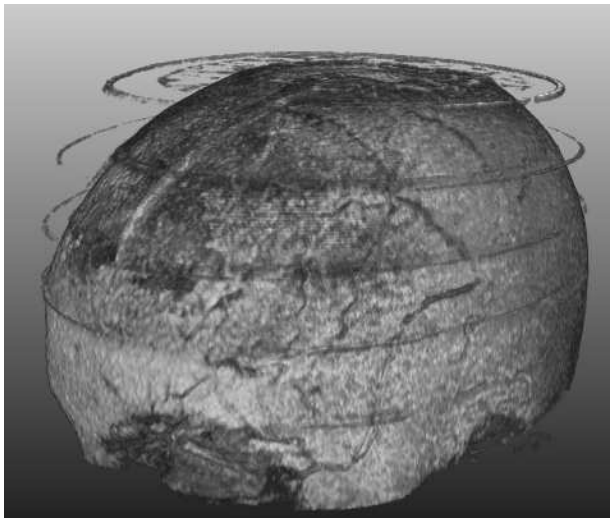


Abbildung 4.1: Ergebnis der Normalisierung durch Standardisierung und Skalierung

Abbildung 4.3, ist ein Teil des Schädelknochens weiterhin vorhanden. Eine solche Reduzierung des Schädelknochen wird trotzdem als gutes Ergebnis betrachtet, da eine Vielzahl an möglichen falschen Zuordnungen weiterhin verhindert werden.

Abbildung 4.2: Skull stripping durch ROBEX mit einer vollständigen Entfernung des Schädelknochen

Abbildung 4.3: Skull-Stripping durch ROBEX mit einer teilweisen Entfernung des Schädelknochen

4.1.2 U-Net

Für die Erstellung des U-Net wird die Bibliothek Keras verwendet. Der Prozess, um ein lauffähiges Modell zu erstellen, wird in drei Teile eingeteilt: Vorbereitung der Eingabedaten, Erstellung und Training des U-Net-Modells sowie Test und Einsatz bei neuen Datensätzen.

Zunächst müssen die Datensätze für den Einsatz im Training des Modells in einzelne Schichtbilder konvertiert werden. Dafür wurde ein Python-Skript erstellt, welches einen MRA-Datensatz und dessen Ground-Truth in einzelne Schichtbilder aufteilt, exportiert und diese ebenfalls normalisieren kann. Beim Export der Bilder kommt es jedoch zu Veränderungen der Intensität der Pixel, wobei niedrige Intensitäten zu stark und hohe Intensitäten zu schwach dargestellt werden. Als Alternative wird eine semi-automatische Variante unter Nutzung der Software MeVisLab für die ersten Tests eingesetzt. Die beobachtete Veränderung kann durch das Ändern der Minimum- und Maximum-Werte der Normalisierung verändert werden. Hierdurch wird das Ergebnis verbessert, aber die Maske hat in Tests eine geringere Übereinstimmung gegenüber der Variante mit MeVisLab. Für diesen ersten Test besteht der Trainingsdatensatz aus 152 Schichten aus Scans von zwei verschiedenen Patienten. Der Testdatensatz besteht aus den 128 Schichten aus einem Scan eines dritten Patienten. Der Trainingsdatensatz besteht hier aus einem gesamten Scan eines Patienten und einigen zusätzlichen Schichten des zweiten Scans, um ein Overfitting wäh-

rend des Trainings zu vermeiden. Alle Daten stammen aus den ausgewählten öffentlichen Datensätzen, deren Auswahl in Kapitel 3 genauer beschrieben ist.

Nach der Erstellung und Kompilierung des U-Net-Modells, wird das Netz mit den genannten Daten trainiert. Diese werden mit Hilfe der Generator-Funktion von Keras in den Trainingsprozess eingebunden. Hier wird eine Anzahl von 100 Iterationen über die Daten festgelegt. Die Anzahl der Iterationen ist ein Hyperparameter, der festlegt, wie häufig die Trainingsdaten durchlaufen werden. Er wird für die ersten Tests aus untersuchten Beispielprojekten übernommen und später durch Tests mit verschiedenen Werten optimiert. Das Training auf einer leistungsfähigen CPU kann damit in weniger als zwei Stunden durchgeführt werden. Ein aus diesem Training generiertes Schichtbild ist in Abbildung 4.4 mit dem eingegebenen Bild und der dazugehörigen Ground-Truth-Maske abgebildet.

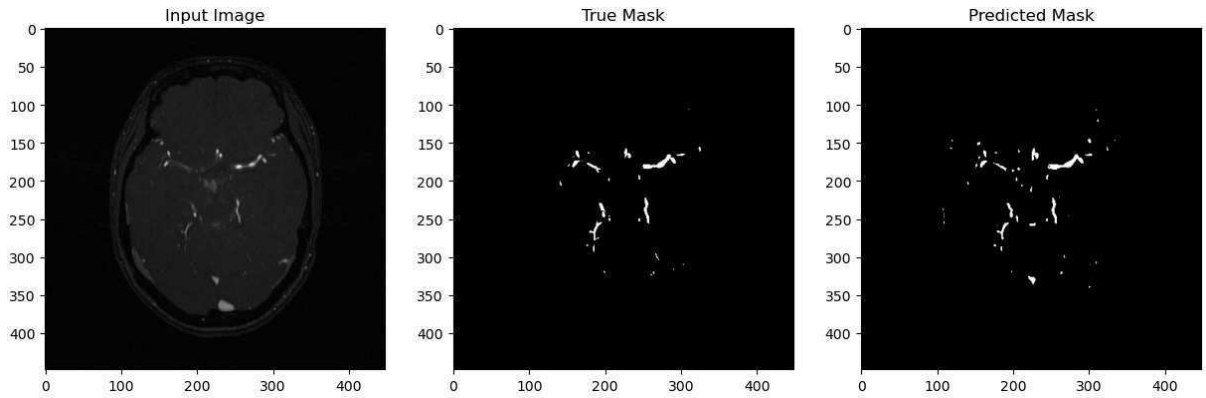


Abbildung 4.4: Darstellung eines eingegebenen Bildes mit der dazugehörigen Maske und der durch das Modell generierten Maske

Abschließend wird das gespeicherte Modell getrennt von dem Trainingsprozess neu geladen und mit einem weiteren Datensatz mit völlig neuen Schichtbildern aus einem Scan eines anderen Patienten überprüft. Hier wurde versucht, den kompletten Datensatz eines Scans aus der Ursprungsdatei einzulesen und zu einzelnen Schichtbildern zu verarbeiten. In dieser Variante hat das Modell unbrauchbare Ergebnisse produziert, wie in Abbildung 4.5 zu sehen ist. Deshalb wird ein zweiter Test durchgeführt, in dem die Eingabedaten wie beim Training des Modells in einzelne Schichtbilder aufgeteilt werden. Abbildung 4.6 zeigt, dass in diesem Test das Modell vergleichbare Ergebnisse zu den Testdaten des

Trainingsprozesses liefert. Nach einer weiteren Vorverarbeitung der Schichtbilder des ersten Versuchs, durch eine Anpassung der Achsenrotation und der Normalisierung, sind vergleichbare Ergebnisse wie bei der Vorverarbeitung mit dem erwähnten Skript erreicht worden.

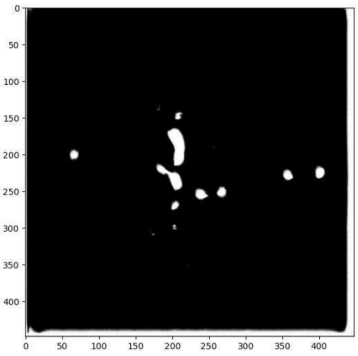


Abbildung 4.5: Generierte Maske mit der Ursprungsdatei als Eingabe

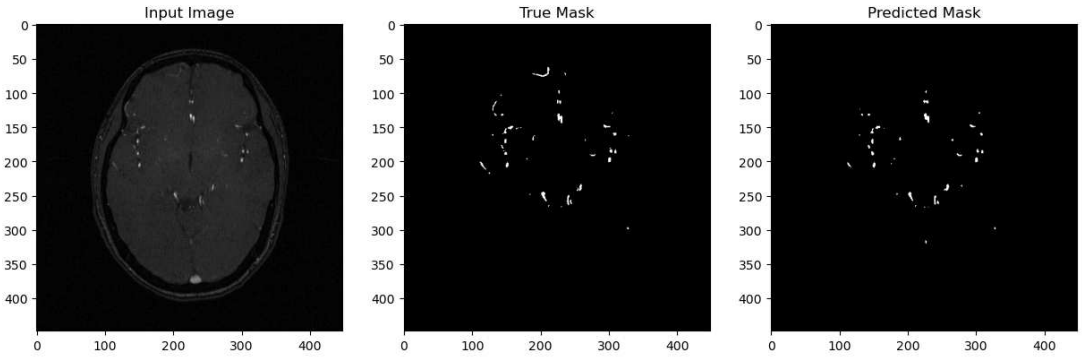


Abbildung 4.6: Ergebnis mit Schichtbildern, wie im Training

Um die Genauigkeit der getesteten Implementation des U-Net-Modells weiter zu untersuchen, wird als nächstes eine Reihe von Verbesserungen an der Software vorgenommen. Der Preprocessing-Vorgang wird verändert, indem die Datenstruktur umgestellt wird. Die Umstellung und eine Anpassung des Skripts ermöglicht eine automatische Aufteilung der Ursprungsdaten des Scans und der dazugehörigen Masken in Schnittbilder und eine Einteilung der generierten Daten in einen Trainingsdatensatz und einen Testdatensatz. Durch eine kleine Erweiterung kann ebenfalls ein dritter Datensatz für die Validierung der Ergebnisse erstellt werden.

Im folgenden Test wird der gesamte zur Verfügung stehende Datensatz verwendet. Hierbei wird eine Aufteilung der Daten gewählt, bei der 75 Prozent als Trainingsdaten und die übrigen 25 Prozent als Testdaten genutzt werden.

Da in diesem Test eine signifikant höhere Datenmenge vorliegt, müssen zuerst plausible Parameter festgelegt werden, damit ein Ergebnis in einer angemessenen Zeit und Qualität vorliegt. Besonders der Zeitfaktor ist hier zu beachten. Bisherige Tests wurden mit 100 Iterationen über den gesamten Trainingsdatensatz vorgenommen. Nach einem ersten Versuch einer Durchführung, kann die ungefähre Dauer einer Iteration ermittelt werden. Diese Dauer liegt bei einer Datenmenge von ca. 2000 Schnittbildern und der dazugehörigen Validierung anhand der Ground-Truth-Daten bei durchschnittlich 40 Minuten auf dem verwendeten System. Vergleichsweise hat eine Iteration mit 152 Schnittbildern ca. 2 Minuten auf dem selben System gedauert. Aufgrund dieser signifikant höheren Dauer, wird in diesem Test nur 10 Mal über die Trainingsdaten iteriert. Der Test wird nur mit dieser Änderung durchgeführt, um den Einfluss der einzelnen Parameter abschätzen zu können.

Die Ergebnisse sind, subjektiv betrachtet, von ähnlicher Qualität wie bei den vorherigen Tests. Um eine genauere Analyse zu ermöglichen, wird der Dice-Score als erste in diesem Projekt eingesetzte Metrik angewendet. Die Betrachtung unterschiedlicher Ergebnisse anhand der Metrik ergibt, dass es je nach Schnittbild starke Unterschiede in der Bewertung gibt. Bei einer großen Stichprobe des Testdatensatzes von 200 Schnittbildern, ergibt der Durchschnitt einen Wert von ca. 0,73.

In den folgenden Abbildungen sind zufällig generierte Ergebnisse mit den dazugehörigen Werten des DSC abgebildet. Diese Bilder zeigen die genannten Unterschiede auf und geben

einen Überblick über die Abweichungen zu den Ground-Truth-Masken. Es ist zu beachten, dass für die Berechnung und Anzeige der Ergebnisse nur Werte über einer Intensität von 0,5 genutzt werden.

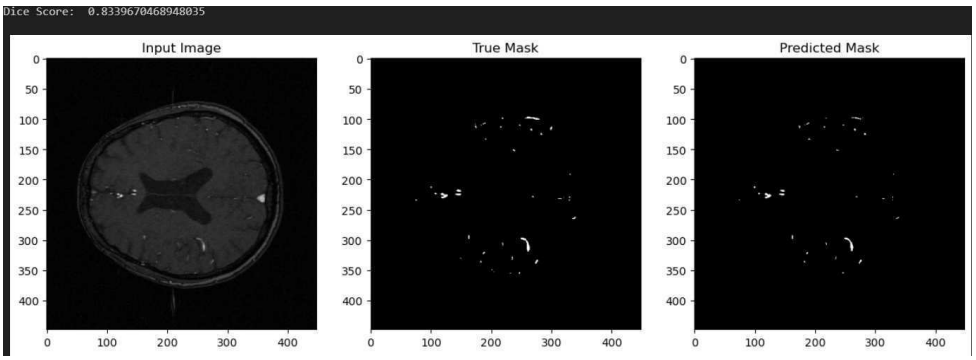


Abbildung 4.7: Ergebnis mit überdurchschnittlich hohem DSC

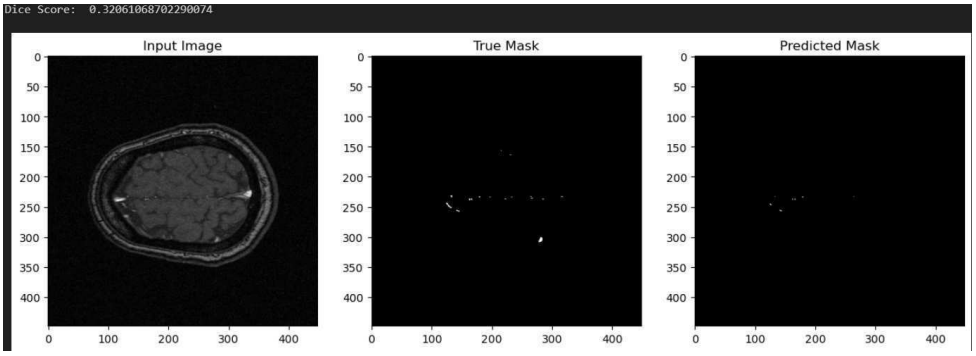


Abbildung 4.8: Ergebnis mit niedrigem DSC

Die berechneten Ergebnisse sind in diesem Test nur in der gleichen Umgebung erstellbar, in der auch das Training durchgeführt wird. Eine weitere Durchführung in einer separaten Umgebung mit den gleichen Testdatensätzen, führt zu unbrauchbaren Ergebnissen. Hier ist zu erwähnen, dass bei einzelnen Tests der ersten trainierten Modelle ein ähnliches Verhalten feststellbar ist und eine deutlich unterschiedliche Leistung des Modells zu beobachten ist.

Als abschließender Test wird ein Training anhand der Schichtbilder von sieben Patienten durchgeführt, um im Anschluss einen Test der erstellten Modelle an den Schichtbildern von drei weiteren Patienten durchzuführen. Dabei werden zwei Modelle mit unterschiedlichen Eingabedaten trainiert. Das erste Modell erhält die Rohdaten ohne jegliche

Preprocessing-Schritte. Das zweite Modell erhält die Ergebnisse einer Kombination der Preprocessing-Verfahren Normalisierung und Skull-Stripping. Nach dem Training erfolgt der Vergleich der Modelle anhand der Rohdaten von drei Patienten, die nicht im Traininsprozess involviert waren. Die Ergebnisse werden anhand der ausgewählten Metriken (Dice Similarity Coefficient, Average Hausdorff Distance) verglichen. Die Metriken für die Segmentierungsqualität der Resultate wurden durch das Tool „SegementationEvalutation“ ermittelt. Das Tool ermöglicht Volumen oder einzelne Schichtbilder mit einer Maske zu Vergleichen und die Metriken daraus zu berechnen. Da die Resultate als einzelne Schichtbilder vorliegen, wird hierfür ein Skript erstellt, welches die Berechnungen automatisiert. Durch das Tool werden die Metriken der einzelnen Schichtbilder in XML-Dateien abgespeichert. Aus diesen Dateien extrahiert das Skript die Werte der einzelnen Metriken und bildet den Durchschnitt über die jeweiligen Datensätze.

In diesem Test werden insgesamt zehn Datensätze verwendet, was der Hälfte der verfügbaren Daten entspricht. Die Datensätze wurden so ausgewählt, weil eine Nutzung aller Datensätze zu erheblich längeren Rechenzeiten führt, ohne deutlich bessere Ergebnisse zu erzielen. Die Daten wurden pro Patient eingeteilt, damit bei der Validierung völlig neue Bilder von anderen Patienten verwendet werden. Insbesondere sollen dabei äußerst ähnliche Bilder aus einer nahe gelegenen, im Training verwendeten Schicht, vermieden werden. Sieben von diesen Datensätzen sind für das Training des Netzes eingeteilt. Die weiteren drei Datensätze werden für den Test der Ergebnisse eingeteilt. Die Ergebnisse von diesem Test sind in der folgenden Tabelle 4.1 aufgeführt.

Datensatz	DICE	AHD
Pat_3	0.63	5.25
Pat_3_skull	0.25	23.42
Pat_37	0.62	3.42
Pat_37_skull	0.16	24.09
Pat_65	0.66	3.33
Pat_65_skull	0.29	20.82

Tabelle 4.1: Ergebnisse des Tests der U-Net-Methode anhand drei verschiedener Patienten mit und ohne Skull-Stripping

4.1.3 BRAVE-NET

Die Erstellung dieses Netzes wird mit der gleichen Software vorgenommen, wie die U-Net-Architektur. Da das U-Net die Basis von BRAVE-NET ist, wird die bereits erstellte Architektur auch für diese Erweiterung genutzt. Als erstes wird hier der Pfad der Kontextaggregation hinzugefügt. Dieser Pfad ist für die Bereitstellung des Kontextbereiches des eigentlichen Patches zuständig. Dafür wird ein Patch mit der zweifachen Größe des ursprünglichen verwendet. Da dieser Patch am Ende der Encoder-Struktur durch eine Concatenation an den Patch angehangen werden soll, muss dieser die gleichen Dimensionen haben. Der Kontextbereich wird deshalb auf die Größe des Patches skaliert, wodurch die Auflösung verringert wird, aber der relevante Kontext erhalten bleibt.

Da der Pfad dem der eigentlichen Encoder-Struktur identisch ist, kann dieser als Vorlage genutzt werden. In den jeweiligen Ebenen werden am Ende Concatenation-Schritte eingefügt, bevor die Ausgabe an die Decoder-Seite der jeweiligen Schicht übergeben wird. Diese Seite wird um die Schritte der Deep Supervision erweitert. Wobei mehr Features durch Zwischenschichten erzeugt werden sollen. Alle Schichten außer die oberste und unterste Schicht werden hier durch Up-Sampling in die Dimension des Output gebracht und danach als einen letzten Schritt durch eine Convolutional-Schicht ergänzt.

Der Aufbau des BRAVE-NET basiert im Gegensatz zu dem bereits verwendeten U-Net-Ansatz auf dessen 3D-Variante. Die bei U-Net verwendete 2D-Version benötigt als Eingabedaten 2D-Schichtbilder mit denen das Modell trainiert wird. Dementsprechend müssen die Schichtbilder am Ende in einem weiteren Schritt zu einem Volumen zusammengesetzt werden. Die hier verwendete 3D-Variante benutzt die vorhandenen Volumen der Bildsequenzen als Eingabedaten. Da Berechnungen mit solch einem Volumen deutlich komplexer sind, werden nur jeweils kleine Bereiche betrachtet. Dazu müssen die Scans in einzelne Teilbereiche (Patches) aufgeteilt werden. Der durch die Autoren zur Verfügung gestellte Quellcode wird als Basis verwendet. Der Code wird zuerst auf Lauffähigkeit überprüft. Da keine Dokumentation bezüglich der benötigten Package-Versionen vorhanden ist, muss zuerst eine Umgebung erstellt werden, die alle Anforderungen erfüllt. Hier muss darauf geachtet werden, dass alle aus Keras importierten Funktionen und Klassen aus der selben Quelle kommen. Es kann zu schwer identifizierbaren Fehlern kommen, wenn Keras aus unterschiedlichen Subpackages von Tensorflow importiert wird. Der Quellcode muss eben-

falls an Neuerungen der aktuellen Versionen der verwendeten Packages angepasst werden, um veraltete Schnittstellen zu entfernen. Nach dieser aufwendigen Anpassung kann der Code bis zur Erstellung des Modells durchgeführt werden.

Für die benötigten Eingabedateien muss ein neues Skript erstellt werden, da die Autoren nur eine unvollständige Version der Extrahierung und Einbindung der Patches zur Verfügung stellen. Dabei werden die Scans in Teilbereiche der Größe 64x64x8 und 128x128x16 eingeteilt. Die Bereiche sollen eine Zuordnung erhalten, ob sie ein Blutgefäß im Zentrum haben oder nicht. Hierfür wird der simple Ansatz verwendet, auf einen Intensitätswert über einem bestimmten Schwellenwert zu überprüfen. Dabei ist zu erwähnen, dass falsche Zuordnungen durch einen schlechten Schwellenwert häufig entstehen können. Ein durch dieses Skript erstellter Bereich ist in Abbildung 4.9 zu sehen. Da durch einen Schwellenwert schlecht zu definieren ist, ob wirklich ein Blutgefäß im Zentrum des Patches ist, können selbst hohe Schwellenwerte auch falsche Zuordnungen erstellen. Bei zu vielen falschen Zuordnungen wird die Qualität der Ergebnisse der Segmentierung verschlechtert, da das Modell davon ausgeht, in vielen Patches Blutgefäße vorzufinden. Solang sich in den Patches Blutgefäße befinden, die nur nicht im Zentrum sind, kann der erzeugte Fehler trotzdem minimal sein. Dabei kann nicht sichergestellt werden, ob genügend Blutgefäße vorhanden sind. Eine Erweiterung des Ansatzes um eine Wahrscheinlichkeitsbasierte Auswahl kann einen solchen Fehler weiter verringern. Dabei wird über eine Maske die Wahrscheinlichkeit definiert, mit der ein Pixel das Zentrum eines Patches sein soll.

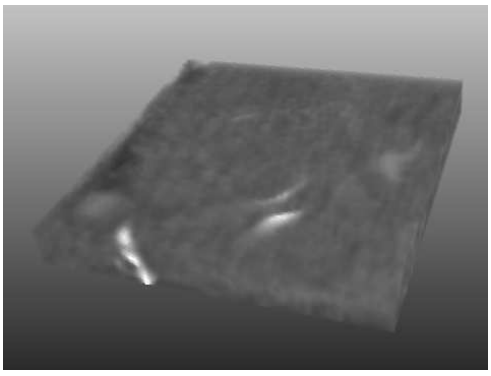


Abbildung 4.9: Erstellter Patch für das BRAVENET-Modell

Um die Funktion der einzelnen Neuerungen zu überprüfen wird ein Testdurchlauf mit den Eingabedaten des ursprünglichen U-Net durchgeführt. Dieser Test dient hauptsächlich der

Entdeckung von Fehlern und der Überprüfung, ob ein Durchlauf erfolgreich abschließt. Bei diesem Test wurden Unstimmigkeiten bei den Auflösungen der beiden Pfade entdeckt. Ein zweiter Durchlauf ohne die Schicht, die den Fehler ausgelöst hat, kann erfolgreich abschließen, aber keine brauchbaren Ergebnisse erzeugen.

Der finale Test der BRAVE-NET-Pipeline findet unter Nutzung der selben Datensätze wie bei U-Net statt. Diese werden durch das erstellte Skript in Patches aufgeteilt und anschließend BRAVE-NET für das Training als Eingabedaten übergeben. Trotz Fehler im Durchlauf der Pipeline, wird ein Modell durch die verwendeten Patches trainiert und im Anschluss auf ein vollständiges Volumen eines Gehirns des Trainingsdatensatzes angewendet wurde.

Die dabei entstandene Segmentierung ist in Abbildung 4.10 zu sehen. Dort ist ersichtlich, dass das Ergebnis unbrauchbar ist. Viele Voxel besonders außerhalb des Gehirns und an den Rändern der einzelnen Patches werden segmentiert. Bei genauerer Analyse der visuellen Darstellung ist ersichtlich, dass innerhalb des Gehirns deutlich weniger segmentierte Voxel zu finden sind, aber immer noch sehr viele Ränder der Patches. Ein Blick in die inneren Bereiche einzelner Patches innerhalb des Gehirns lässt vereinzelte Bereiche ersichtlich werden, die zumindest annähernd wie die mögliche Segmentierung eines Blutgefäßes aussehen. Ein Beispiel ist in Abbildung 4.11 dargestellt, wo vereinzelt plausible Segmente in den zentralen Patches zu sehen sind.

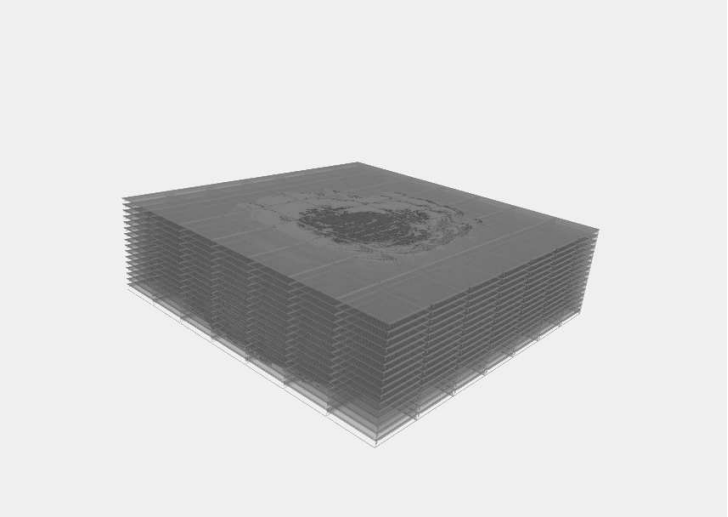


Abbildung 4.10: Segmentierung durch BRAVE-NET

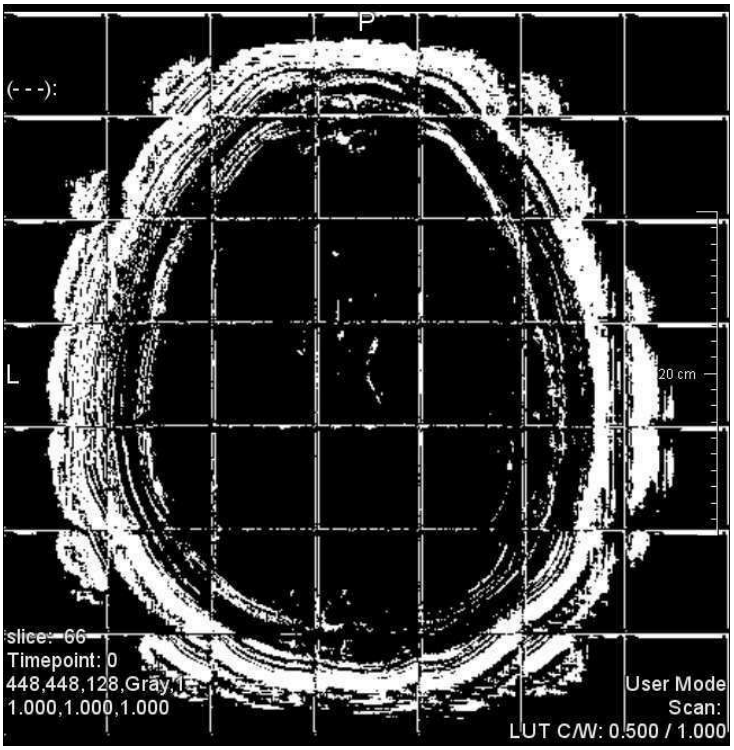


Abbildung 4.11: Schichtbild der BRAVE-NET-Segmentierung

Durch die visuell eindeutig nicht brauchbaren Ergebnisse, wird von weiteren Tests und einer Überprüfung durch Metriken abgesehen und BRAVE-NET als nicht brauchbar bewertet.

4.1.4 VMTK

Da VMTK ein eigenständiges Softwareprogramm ist, werden keine eigenen Implementationen benötigt. Dadurch kann die Testphase deutlich schneller beginnen gegenüber den CNN-Ansätzen. Es wurde sich dafür entschieden, VMTK über die GUI „Pypepad “ zu benutzen. In Pypepad können die verschiedenen Skripte des VMTK ausgeführt und die damit erstellbaren Pipelines abgespeichert werden. Es können Tests schnell angepasst und eine schnelle Wiederholung von Testdurchläufen sichergestellt werden. Für den ersten Test wird eine Pipeline erstellt, die das Laden eines Scans, die Segmentierung und das Abspeichern in einem brauchbaren Format ermöglicht. Während es für das Laden und Speichern der Dateien standardmäßige Lese- und Schreib-Skripte gibt, wird für die Segmentierung das Skript „vmtklevelsetsegmentation“ eingesetzt. Sobald ein Scan durch den Reader geladen wurde, öffnet das Skript eine grafische Benutzeroberfläche. In dieser wird der Scan in einer 3D-Umgebung und alle vorhandenen Optionen für die Segmentierung angezeigt.

Die Segmentierung läuft in hauptsächlich zwei Schritten ab. Ein Initialisierungsschritt und die Level-Set-Segmentierung. In der Initialisierung werden Pixel festgelegt, die den Startzustand des Deformable-Modells darstellen. Es sind fünf verschiedene Varianten für diesen Schritt implementiert.

1. „Colliding Fronts“ ermöglicht diesen Schritt durch die manuelle Auswahl von zwei Startpunkten. Von diesen Punkten aus wird eine Welle ausgesendet, die je nach Intensität der Pixel unterschiedlich schnell ist. Durch die Richtungen der Wellen kann der Bereich zwischen den Punkten ermittelt werden, welcher dann das Level-Set bildet. Hierbei können Thresholds festgelegt werden, um zu niedrige oder hohe Intensitäten auszuschließen.

   Bei dieser Variante kann zwar ein visuell akzeptables Ergebnis erzielt werden, jedoch werden nur einzelne verbundene Blutgefäße dargestellt, die sich zwischen den jeweiligen Punkten befinden. Es wird zwar ermöglicht verschiedene Segmentierungen zusammenzufügen, aber es würde ein zu hoher manueller Aufwand entstehen. Deshalb wird diese Variante nicht weiter verfolgt.

2. Die „Fast Marching“-Methode funktioniert ähnlich wie die „Colliding Fronts“-Variante. Hier werden eine Menge von Startpunkten und eine Menge von Zielpunkten festgelegt. Es werden ebenfalls Wellen von den Startpunkten ausgesendet, die fortlaufen, bis der erste Zielpunkt erreicht wurde. Der Bereich zwischen Start und Ziel ist dann der festgelegte Bereich des Modells.

   Es entsteht wie bei der vorherigen Methode ein zu hoher Aufwand, wodurch auch diese Methode nicht weiter berücksichtigt wird.

3. „Threshold “ ist die bekannte Methode der Festlegung eines hohen und niedrigen Schwellenwertes. Alle Pixel innerhalb dieses Bereichs werden dann dem Modell zugeordnet. Diese Methode liefert mit sinnvoll ausgewählten Werten ein akzeptables Ergebnis, welches durch Preprocessing der Rohdaten mit Hilfe von Anwendung der Methoden der Kategorie „Vessel Enhancement“ noch weiter verbessert werden kann.

   Diese Variante wird im weiteren Verlauf weiter genutzt, da mit einem geringen Aufwand visuell akzeptable Ergebnisse erzielt werden.

4. Die verbleibenden Varianten „Isosurface “ und „Seeds “ wurden getestet, aber aufgrund von mangelhafter Dokumentation und unbrauchbarer Ergebnisse nicht weiter betrachtet.

Nach der Initialisierung wird die Segmentierung durchgeführt, wobei das Modell deformiert wird. Hierfür werden zuerst Parameter gesetzt, die diese Deformation steuern. Für die Tests werden die in der Dokumentation vorgeschlagenen Werte für die Parameter genutzt. Nach der Ausführung wird nach wenigen Minuten ein Ergebnis generiert. Wie in Abbildung 4.12 zu sehen ist, wurde schon bei diesem Test ein Ergebnis erzielt, bei dem eine Vielzahl an Blutgefäßen visuell erkennbar segmentiert werden. Viele Pixel werden aber falsch zugeordnet. Sichtbar ist eine größere Zahl an falschen Zuordnungen im Bereich des Schädelknochens. Um das zu bestätigen, wird das Ergebnis mit Hilfe des Dice Similarity Coefficient mit dem Ground Truth verglichen. Der Wert aus diesem Test beläuft sich auf 0,12 und bestätigt damit eine Vielzahl an falschen Zuordnungen.

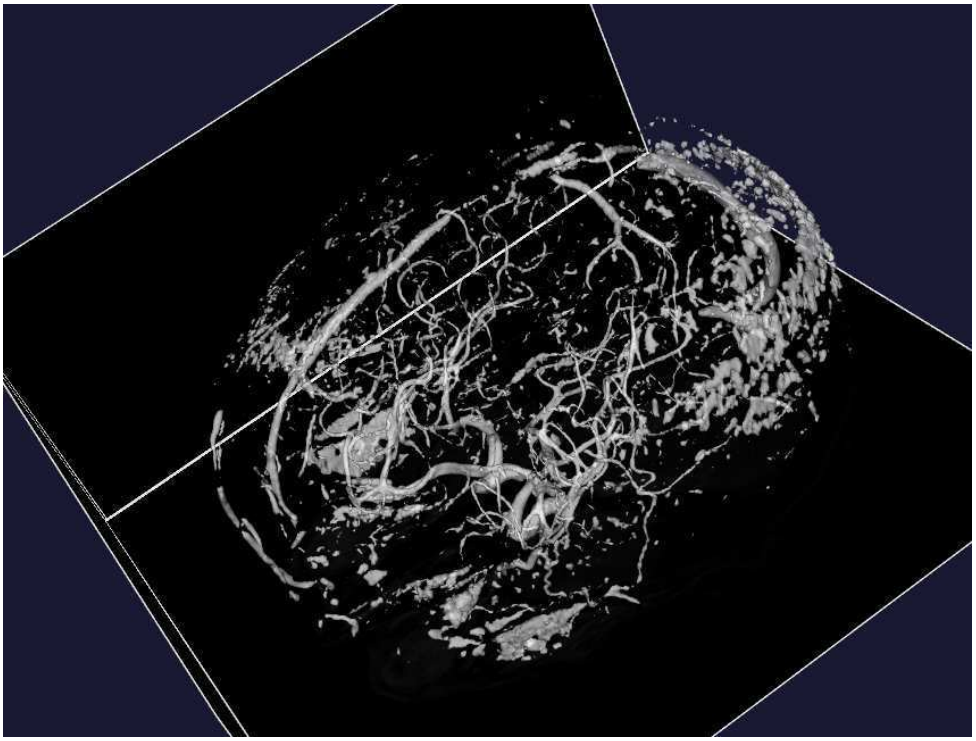


Abbildung 4.12: Ergebnis des VMTK-Tests

Um die Ergebnisse zu verbessern, werden im folgenden vorbereitende Schritte vor der Segmentierung ausgeführt. Dabei werden Methoden der bereits erwähnten Vessel-Enhancement-Kategorie verwendet. Zu Beginn wird die häufig für diesen Schritt eingesetzte Methode eines Vesselness-Filters eingesetzt. Ein solcher Filter auf Basis des Hessian-Filters ist in ITK vorhanden und liefert die in Abbildung 4.13 zu sehenden Ergebnisse in einem Testdurchlauf. Hierbei sind klare gefäßähnliche Strukturen zu erkennen. Hier ist zu beachten, dass trotzdem Pixel, die wahrscheinlich nicht zu einem Blutgefäß gehören, eine Intensität erhalten, die größer als null ist. Aus diesem Grund wird versucht, mit Hilfe eines Threshold diese geringen Bereiche zu entfernen. Das Resultat wird anschließend im gleichen Verfahren mit der Level-Set-Methode von VMTK segmentiert. Das daraus resultierende Volumen ist in Abbildung 4.14 zu sehen. Dabei ist zu erkennen, dass deutlich weniger Pixel segmentiert werden im Vergleich mit den Ergebnissen ohne Preprocessing. Zu beachten ist, dass viele kleine Bereiche segmentiert werden, die keine zusammenhängende Strukturen ergeben.

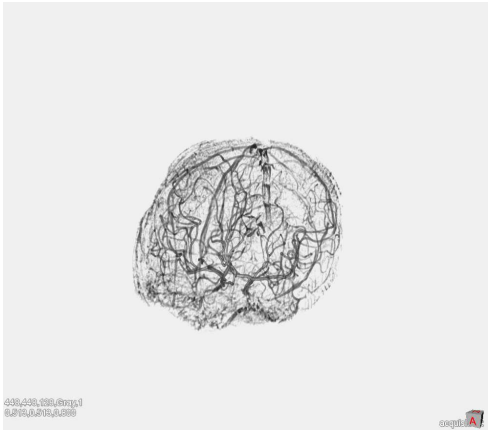


Abbildung 4.13: Ergebnis des Vesselness-Filter

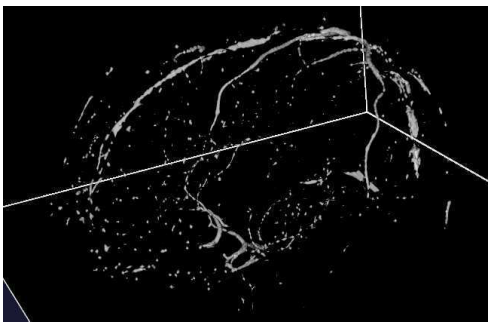


Abbildung 4.14: Ergebnis des VMTK-Tests mit einem Vesselness-Filter

Da die Threshold-Funktion des VMTK selbst eine simple Implementation mit festzulegenden oberen und unteren Grenzwerten ist, wird versucht, diese Variante durch die Methode von Otsu zu ersetzen. Hierbei wird anhand des Histogramm der Intensitäten ein Grenzwert der beiden Klassen ermittelt. Bei dem verwendeten Datensatz ergibt diese Funktion mit den vorgegebenen Parametern nur ungenügende Ergebnisse, da zu viele Pixel fälschlich nicht als Hintergrund erkannt werden. Nach einer Anpassung des Parameter „Background Threshold“ auf einen Wert von 100 wird das in Abbildung 4.15 zu sehende Ergebnis erzielt.

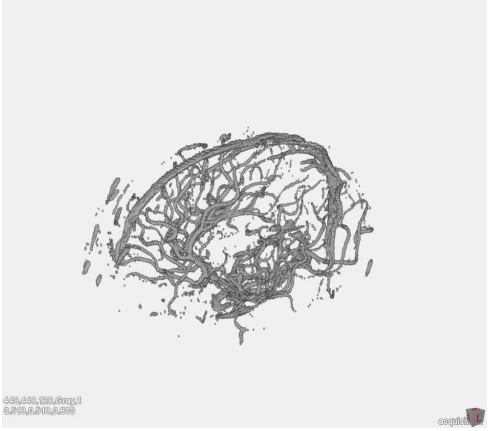


Abbildung 4.15: Ergebnis des Otsu-Thresholding

In VMTK muss das dortige Thresholding den Grenzwert 1 erhalten, da sonst die durch die Otsu-Methode segmentierten Bereiche nicht erkannt werden. Das in Abbildung 4.16 zu sehende Resultat nach der Level-Set-Segmentierung zeigt deutlich weniger erkennbare, nicht zusammenhängende Stukturen und besser erkennbare gefäßähnliche Strukturen im Vergleich zu der Vesselness-Filter-Methode.

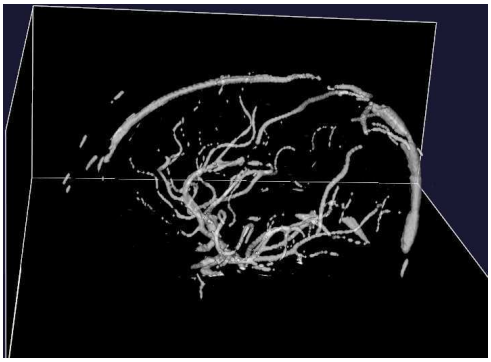


Abbildung 4.16: Ergebnis des VMTK mit Otsu-Thresholding

4.1.5 Vesselness-Filter

Die in Absatz 4.1.1 für das Preprocessing verwendeten Techniken lassen sich auch als eigenständige Segmentierungsverfahren verwenden. In diesem Kapitel wird eine Pipeline ausschließlich mit den bereits verwendeten Methoden des Preprocessing und den für VMTK bereits angewendeten Vesselness-Filter und dem Otsu-Thresholding erstellt.

Zunächst wird mit den vorverarbeiteten Eingabedaten getestet, indem eine Pipeline mit Hilfe des Tools MeVisLab erstellt wird. Dazu müssen die entsprechend der ITK-Methode benannten Module, sowie Module für das Lesen und Schreiben der Ein- und Ausgabedaten miteinander verbunden werden. Die resultierende Pipeline ist in Abbildung 4.17 zu sehen. Hier ist zu beachten, dass Module in MeVisLab am unteren Rand Eingabe-Slots und am oberen Rand Ausgabe-Slots enthalten und dementsprechend die Pipeline von unten nach oben zu lesen ist.

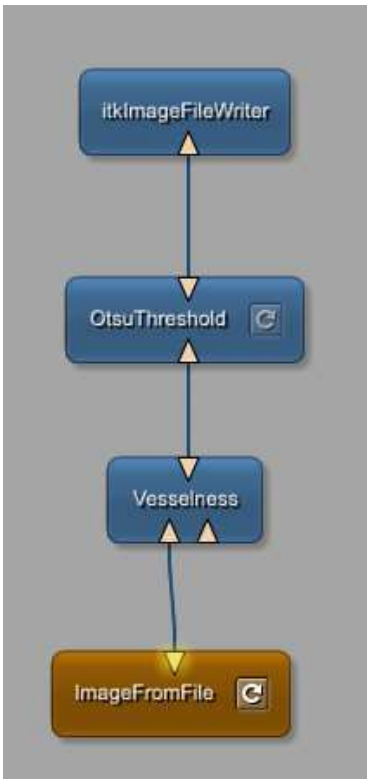


Abbildung 4.17: Darstellung der MeVisLab Pipeline

Durch eine Anpassung der Parameter des Vesselness-Filters kann eine feinere Segmen-

tierung erstellt werden. Durch das Erhöhen der Anzahl an Scales und der Einstellung des oberen Sigma-Wertes, können einige fehlerhafte Bereiche entfernt werden. Durch diese Anpassung sind die durch die Filter als Wahrscheinlichkeiten generierten Intensitäten deutlich kleiner. Es wird ein Rescale-Filter für die Transformation der Intensitätswerte hinzugefügt, damit der Otsu-Threshold angewendet werden kann. Da dieses Modul nur mit ganzzahligen Werten verwendet werden kann, wird die Intensität zwischen 0 und 100 skaliert. Die folgenden Abbildungen zeigen das Resultat der Pipeline mit unterschiedlichen Parametern des Vesselness-Filters. Dabei ist zu erkennen, dass eine Erhöhung der Anzahl der Scales bei gleichbleibendem Minimum einzelne fehlerhafte Bereiche der Segmentierung entfernt.

Abbildung 4.18: Ergebnis mit einem Scale mit Sigma 1.0

Abbildung 4.19: Ergebnis mit 4 Scales mit Sigma Minimum 1.0 und Maximum 4.0

Die segmentierten Volumen werden in Schichtbilder aufgeteilt, und mit Hilfe der Metriken evaluiert. Die Ergebnisse der Metriken sind in der folgenden Tabelle aufgelistet 4.2. Dabei sind die Datensätze je nach verwendeten Parametern gekennzeichnet („A “= 1 Scale, „B “= 4 Scales).

Datensatz	DICE	AHD
Pat_3 _A	0.25	12.08
Pat_3_B	0.23	14.39
Pat_37 _A	0.31	9.45
Pat_37_B	0.28	10.58
Pat_65 _A	0.28	9.02
Pat_65_B	0.26	9.6

Tabelle 4.2: Ergebnisse des Test der Vesselness-Filter-Methode anhand drei verschiedener Patienten

4.2 Erstellung der Pipeline

Die Entwicklung der Pipeline orientiert sich an allen vorher durchgeführten Tests, wobei besonders die Automatisierung der Ausführung als wichtiges Kriterium berücksichtigt wird. In Abbildung 4.20 ist eine Darstellung des Aufbaus der Pipeline zu sehen, welcher im Folgenden genauer beschrieben wird.

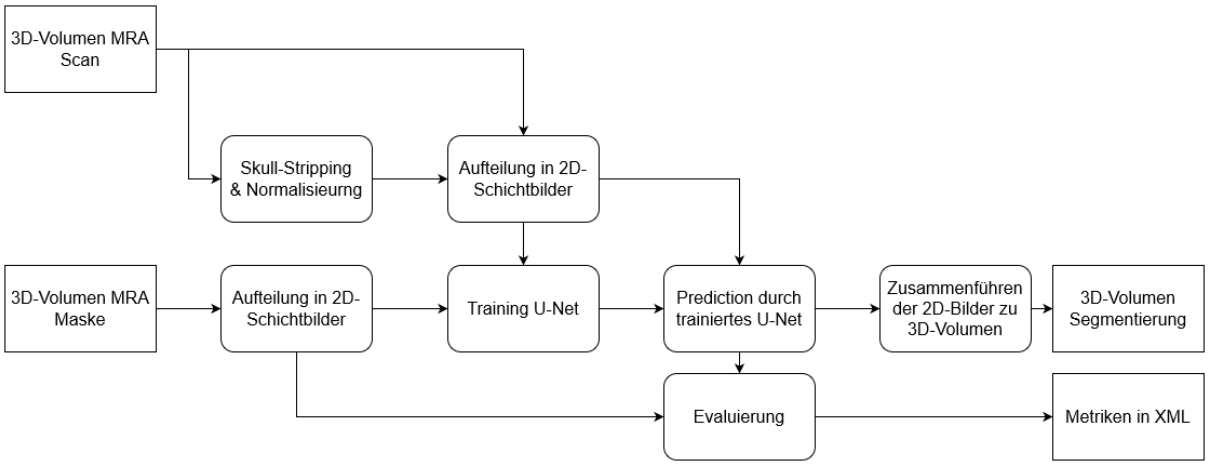


Abbildung 4.20: Darstellung der erstellten Pipeline

Als Eingabedaten für die gesamte Pipeline dienen die bereits verwendeten Datensätze der MIDAS-Datenbank und die dazugehörigen Ground-Truth-Masken für das Training. Um das vorhandene Rauschen in den Bildern zu minimieren und weniger möglicherweise fehlerhafte Segmentierungen beim Training zu erhalten, wird zu erst durch das Tool ROBEX ein Skull-Stripping und die Normalisierung der Daten vorgenommen. Hierfür wird das Tool durch ein Skript ausgeführt und erhält als Parameter den Pfad zum zu verarbeitenden Datensatz und den Pfad für die Ausgabe. Da für das Training mehrere Datensätze genutzt werden, wird der angegebene Ordner nach allen enthaltenen MHA-Daten durchsucht, welche anschließend nacheinander durch das Tool bearbeitet werden.

Die Tests haben eindeutig gezeigt, dass ein Ansatz mit einem Convolutional Neural Network basierend auf der U-Net Architektur die besten, brauchbaren Ergebnisse erzeugt. Aus diesem Grund werden die in den Tests verwendeten Skripte in die Pipeline integriert. Dazu werden die vorbereiteten 3D-Volumen in 2D-Schichtbilder aufgeteilt, um anschließend im Training eingesetzt zu werden. Dabei werden die Schichtbilder automatisch in

einer Ordnerstruktur gespeichert, die für das Training vorgesehen ist. Dabei werden die Datensätze je nach Einstellung zu einem bestimmten Verhältnis in Trainings- und Testdatensätze aufgeteilt.

Das Training des Modells wird danach automatisch gestartet. Hier gilt es zu beachten, dass die Dauer des Trainings je nach Auswahl der Epochen und Menge der verwendeten Daten viel Zeit in Anspruch nehmen kann. Nach Abschluss des Trainings wird das neue Modell im Projektordner abgespeichert. Unter Angabe des Namen kann dadurch ohne erneutes Training die Segmentierung von neuen Datensätzen berechnet werden.

Als zusätzlicher Postprocessing-Schritt wird auf die Ergebnisse ein Threshold angewendet, damit nur Bereiche genutzt werden, die eine höhere Wahrscheinlichkeit als diesen vordefinierten variabel bestimmbaren Threshold haben.

Nach der Segmentierung ist eine optionale Evaluierung der Ergebnisse anhand von vorhandenen Masken durchführbar, die in den dafür vorgesehenen Ordner gelegt werden müssen. Dabei werden alle Schichtbilder einzeln evaluiert, dementsprechend muss die Maske ebenfalls in Schichtbilder mit den selben Maßen geteilt werden. Die Ergebnisse der Bewertung werden einzeln in XML-Daten gespeichert. Ein Durchschnitt der Evaluierung wird ebenso abgespeichert und in der Konsole angezeigt.

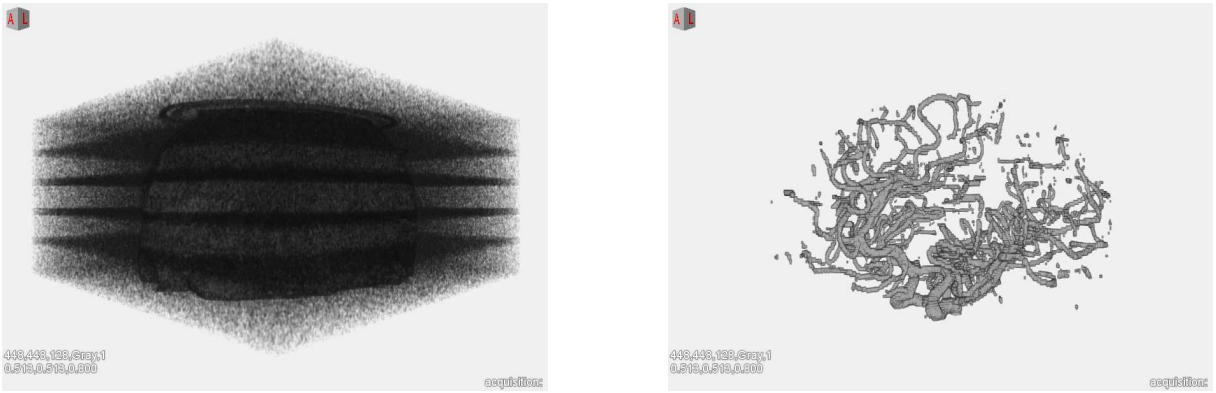
Der letzte Schritt der Pipeline führt die Schichtbilder wieder zu einem 3D-Volumen zusammen. Dafür werden die Schichtbilder, die als 2D-Arrays vorliegen, einem 3D-Array angehängt, wobei darauf geachtet wird, dass die Z-Achse zum anhängen der Daten verwendet wird. Dieser 3D-Array benötigt außerdem Daten zu den Abständen der einzelnen Bilder, um ein funktionsfähiges Volumen zu erstellen. Dafür wird eine affine Transformation aus einem der Eingabedatensätze kopiert und zusammen mit dem 3D-Array wird daraus eine Nifti-Datei erstellt, die die finale Ausgabe der Pipeline darstellt.

Bei dieser Pipeline muss ein Nutzer nur Pfadvariablen angeben und hat zudem die Möglichkeit, einige Parameter zu individualisieren. Dabei sind die Preprocessing-, Trainings- und Evaluations-Abschnitte optional und müssen nicht durchgeführt werden. Dadurch kann eine schnelle Segmentierung nur durch ein trainiertes U-Net-Modell erfolgen, die innerhalb weniger Sekunden Ergebnisse erzeugt. In den Projektdaten dieser Arbeit befinden sich zwei trainierte Modelle, die mit jeweils sieben Trainings- und drei Testdatensätzen trainiert wurden. Sie unterscheiden sich nur in den verwendeten Daten, wobei

ein Modell vorverarbeitete Daten erhalten hat, während das zweite Modell nur Rohdaten benutzt. Beide Modelle wurden über eine Dauer von acht Stunden mit 40 Epochen trainiert.

Die Pipeline läuft, nachdem die nötigen Einstellungen getroffen wurden, völlig automatisch bis zu der Erstellung von Ergebnissen. Im Folgenden wird die Nutzung der Pipeline anhand der Erstellung einer der enthaltenen Modelle demonstriert.

Nach der Installation der benötigten Packages, die in der beigefügten ReadMe-Datei genauer erläutert wird, müssen die Daten der Scans und dazugehörigen Masken in den jeweils dafür vorgesehenen Unterordnern abgespeichert werden. In den folgenden Abbildungen 4.21a und 4.21b sind die Daten visuell dargestellt.

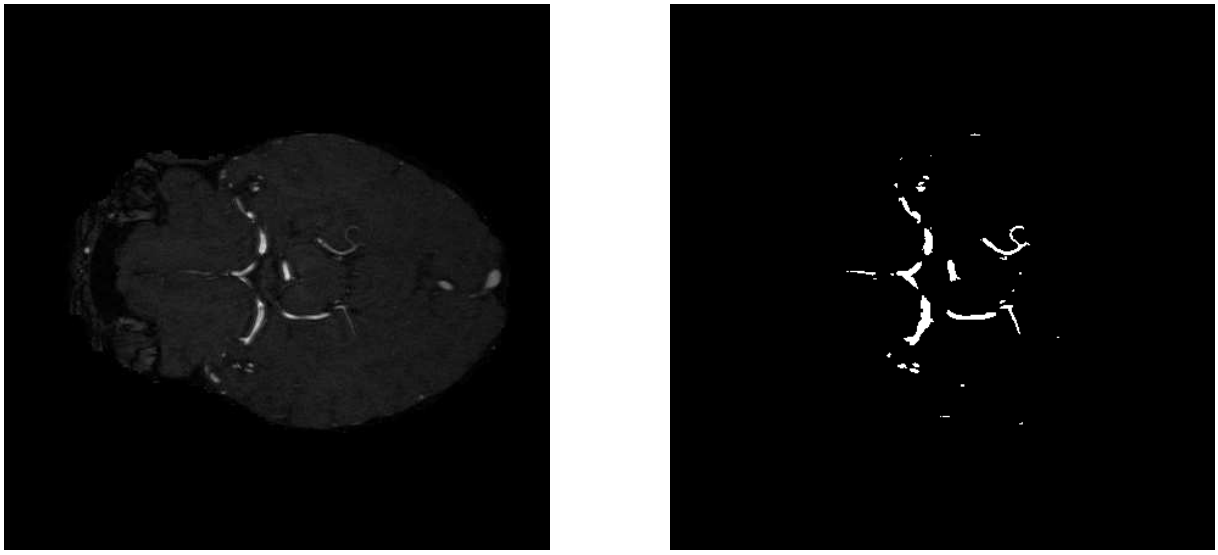


(a) Scan mit sichtbarem Rauschen **(b)** Maske

Abbildung 4.21: 3D-Darstellungen der Eingabedaten

Die Pipeline kann danach über die Kommandozeile gestartet werden, wobei die gewünschten Parameter geändert werden können. Änderungen bezüglich der Pfade und die Auswahl eines zu verwendenden Modell können im Skript selbst im Bereich „Nutzervariablen “geändert werden.

Nach Ausführung sind verschiedene Konsolenausgaben zu sehen, die über den aktuellen Status der Pipeline informieren. Am Ende eines jeden Zwischenschrittes werden die Daten in den vorgesehenen Unterordnern zwischengespeichert. Nach der Vorverarbeitung liegen die Daten als 2D-Schichtbilder vor, wie in 4.22a und 4.22b beispielhaft zu sehen ist.



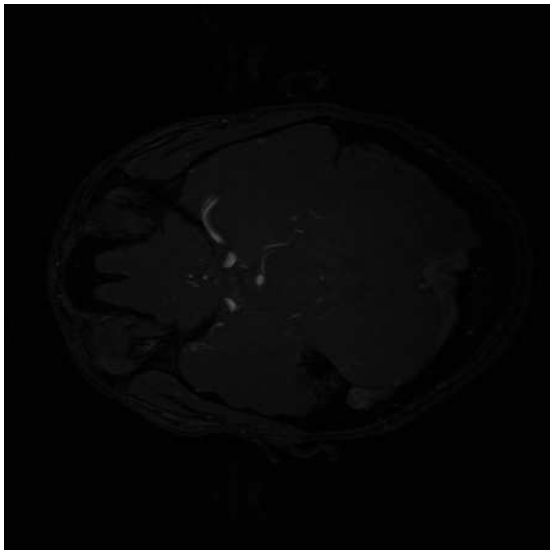
(a) Scan **(b)** Maske

Abbildung 4.22: Extrahierte Schichtbilder

Mit diesen Schichtbildern startet die Pipeline das Training. Durch die Angabe von 40 Epochen trainiert das Programm nun wie bereits erwähnt das neue Modell, wonach es automatisch mit einem als Parameter definierbaren Namen gespeichert wird.

Am Ende der Durchführung werden 3D-Volumen aller Testdatensätze erstellt. Zum weiteren Test der Pipeline wird ein erneuter Durchlauf ohne Training, mit einem völlig neuen Datensatz, der nicht für das Training genutzt wurde, gestartet.

Daraus resultieren jeweils 2D und 3D Ergebnisse der Segmentierung. In den folgenden Abbildungen 4.23a, 4.23b und 4.23c werden ein Schnittbild mit der dazugehörigen Ground-Truth-Maske und der errechneten Maske angezeigt.



(a) Scan

(b) Maske

(c) Segmentierung

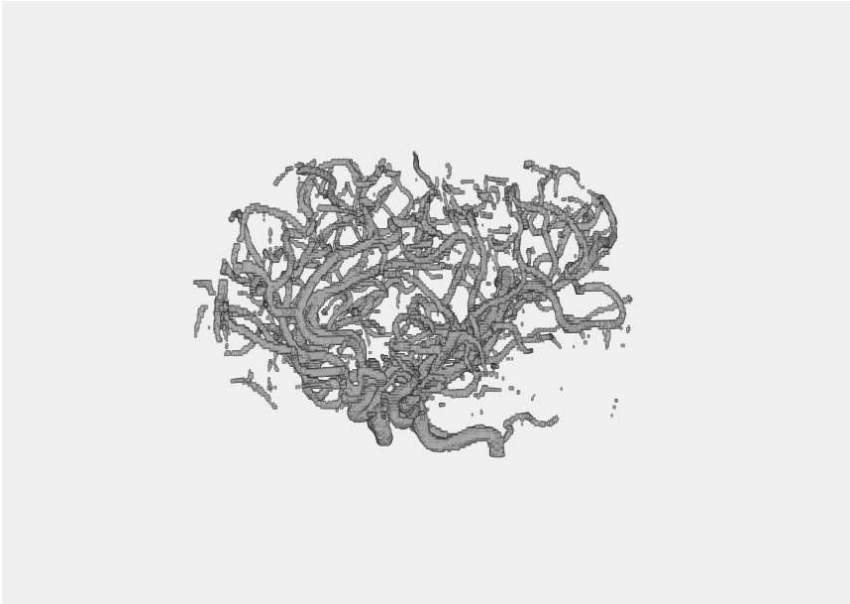
Abbildung 4.23: Vergleich eines Schichtbildes

Hier wird ein Threshold von 0,8 verwendet, um zu verdeutlichen, wie sicher sich das Modell mit den Berechnungen ist.

Wie bereits erwähnt, wird das Resultat noch zu einem 3D-Volumen zusammengesetzt. Da dieses Volumen direkt als Nifti-Datei exportiert wird, kann es ohne weitere Schritte sofort in anderen Anwendungen wie MeVisLab oder dem VR-Demonstrator eingebunden werden.

In der Abbildung 4.24a ist das 3D-Volumen in MeVisLab dargestellt. Zum Vergleich wird in Abbildung 4.24b das Ground-Truth-Volumen gezeigt.

(a) Ergebnis



(b) Ground-Truth

Abbildung 4.24: Vergleich Ergebnis mit Ground-Truth

5 Diskussion

Dieses Kapitel dient zur Diskussion aller Bestandteile der Arbeit. Hier werden Begründungen für getroffene Entscheidungen gegeben. Besonders wird die Verwendung der Verfahren, Datensätze und Metriken bewertet.

5.1 Recherche, Auswahl und Nutzung der Verfahren

Die anfängliche Literaturrecherche hat einen Überblick über vorhandene Verfahren und deren Einteilung in Gruppen gegeben. Dabei konnte bereits das Problem erkannt werden, dass viele Verfahren nicht auf zerebrale Blutgefäße ausgelegt und teilweise nicht auf MRA-Datensätze anwendbar sind. Das Problem der automatisierten Segmentierung ist zum Zeitpunkt der Erstellung dieser Arbeit immer noch ein Gebiet, welches aktiv erforscht wird. Es gibt keine vollständigen Lösungen des Problems und starke Unterschiede zwischen den Ansätzen, die aus verschiedenen Disziplinen kommen. Dieser große Umfang lässt keine Erschließung von allen Bereichen und Verfahren in dieser Arbeit zu.

Viele Publikationen sind nicht öffentlich einsehbar oder beschreiben die Umsetzung des vorgestellten Verfahrens nur teilweise. Die Recherche nach Quellcode und Beispielprojekten wurde durch seltene Nennungen in den Veröffentlichungen erschwert. Im Laufe der Recherche ist auch deutlich geworden, dass die wenigsten Publikationen einen eigenen Quellcode veröffentlicht haben oder diesen nur geringfügig instand halten.

Die Auswahl wurde nur auf solche Verfahren beschränkt, für die Quellcode oder Beispielprojekte vorhanden sind, da diese die Einarbeitung in das Thema deutlich unterstützen. Der bereits erwähnte Umfang des Themas und die damit verbundene Fehleranfälligkeit machen eine eigene Implementierung der mathematischen Grundlagen von Veröffentlichungen impraktikabel und nicht im Rahmen einer Masterarbeit umsetzbar.

5.1.1 Nutzung der Verfahren

5.1.1.1 UNet

Der Einsatz von UNet war, verglichen mit den weiteren Verfahren, der Funktionalste von allen betrachteten Verfahren. Durch die hohe Bekanntheit des Verfahrens, sind viele Implementationen verfügbar, wodurch die Einarbeitung und Umsetzung signifikant erleichtert wurden. Neben der ursprünglichen Veröffentlichung sind viele Materialien zu der Theorie des UNet vorhanden. Da dieser Ansatz auf einem Convolutional Neural Network basiert, können hierfür häufig verwendete Frameworks, wie Tensorflow, genutzt werden, wodurch die Umsetzung weiter erleichtert wird.

Der Bereich der Neuralen Netze ist sehr komplex. Der gesamte Aufbau hat viele Fehlerquellen, d.h. ein Fehler im Aufbau des Netzwerkes oder die fehlerhafte Einstellung eines Parameters kann die Ergebnisse stark beeinflussen. Durch unklare Fehlercodes und Ausgaben kann eine Fehlersuche sehr zeitaufwendig sein.

Durch die Abhängigkeit von unterschiedlichen Versionen der Packages entstehen ebenfalls weitere Fehlerquellen. Einige Funktionen der genutzten Implementation mussten dementsprechend selbst angepasst werden.

Beispielsweise ist es dazu gekommen, dass einige der trainierten Modelle nicht funktionell waren. Um die Quelle des Fehlers zu finden, wurde eine intensive Fehlersuche durchgeführt. Mehrere Ausführungen der Modelle mit unterschiedlichen Eingabedaten zeigten das gleiche Verhalten. Es muss sich daher um einen Fehler im Training der Modelle handeln. Bei der weiteren Untersuchung sind einige mögliche Faktoren für dieses Verhalten festgestellt worden. Die Dauer des Trainings und die verwendeten Eingabedaten haben deutlich Einfluss auf das Resultat des Trainings. Weitere mögliche Fehlerquellen bestehen in den verwendeten Packages, deren Funktionen und möglicher interner Fehler während des Trainings durch Tensorflow. Deutliche Verbesserungen konnten erzielt werden, indem andere Eingangsdaten verwendet und die Dauer des Trainings erhöht wurde.

Hier ist zu erwähnen, dass die Metrik „Accuracy“ von Keras am Ende des Trainings des ersten Neuralen Netzes eine sehr hohe Genauigkeit von 0.9978 anzeigt. Die Ergebnisse

lassen sich auf den großen Anteil des schwarzen Hintergrund und eine relativ gut übereinstimmende Zuordnung der Blutgefäße zurückführen. Dabei ist zu beachten, dass selbst bei dieser Genauigkeit noch durchschnittlich ca. 440 Pixel in den 448x448 Pixel großen Aufnahmen falsch zugeordnet werden. Es wird dabei visuell ersichtlich, dass die falsche Zuordnung in beide Richtungen geschieht, d.h. sowohl Falsch-Positiv als auch Falsch-Negativ. Die wahrscheinlichsten Fehlerquellen sind hierbei die restlichen Bestandteile des Gehirns, welche keine Blutgefäße sind, der Schädelknochen und in der Aufnahme vorhandenes Bildrauschen. Trotzdem zeigt der Test gute Ergebnisse, die in vielen Bereichen mit dem vorgegebenen GroundTruth eine hohe Übereinstimmung haben, wie in der Abbildung 4.4 anhand einer Schicht zu sehen ist.

Das UNet-Modell hat hochwertige Ergebnisse erzeugt, trotz der sehr geringen Verfügbarkeit von einsetzbaren Daten. Hier kann es schnell zu Overfitting kommen. Eine klare Trennung der Daten in Trainings- und Testdaten und eine größere Menge an Daten für das Training verhindert dieses Verhalten. Des Weiteren wird durch die Nutzung eines Generators von Tensorflow eine Datenaugmentation während des Trainings durchgeführt, um dieses Problem weiter zu verringern.

Während der Testphase des Verfahrens wurde eine Conda-Umgebung genutzt und die Implementation des Verfahrens mit Hilfe eines Jupyter Notebook umgesetzt. Zur einfachen Ausführung und dem Management der Umgebungen wurde Visual Studio Code verwendet. Hier ist zu erwähnen, dass es dabei ein signifikantes Problem mit Visual Studio Code gab. Während der Ausführung des Trainings ist der Kernel des Programms häufig abgestürzt. Dieser Fehler wird durch ein Package verursacht, welches für die Nutzung des Jupyter Notebook in Visual Studio Code benötigt wird. Das Problem kann durch die Nutzung von Jupyter Notebook außerhalb der VSC Umgebung verhindert werden. Eine Verwendung eines einfachen Python-Skript per Konsole ist ebenfalls möglich.

5.1.1.2 VMTK

Die Nutzung von VMTK ist durch die Verfügbarkeit als fertig gebautes Programm gegeben und damit die einfachste Lösung. Der Einsatz des Tools ist durch eine Dokumentation weiter unterstützt. Trotzdem gestaltete sich die Nutzung als schwierig, da das Tool sehr stark von den Eingabedaten abhängig ist. Es wurde eine Vielzahl von vorbereitenden Schritten getestet, um die Ergebnisse zu verbessern. Insbesondere ist eine Entfernung

des Rauschens der Originaldaten notwendig. Bei dieser Entfernung dürfen allerdings die Verbindungen der Blutgefäße nicht beeinträchtigt werden, da sonst Fehler in der Segmentierung entstehen. Es ist vorteilhaft, dass VMTK im Gegensatz zu den neuronalen Netzen völlig ohne Groundtruth-Daten auskommt. Die sequenzielle Eingabe von verschiedenen Parametern verhindert allerdings eine vollständig automatisierte Umsetzung, da diese manuell in einem Viewer eingegeben werden müssen.

Die Level-Set-Segmentierung hat signifikante Schwierigkeiten, kleine Blutgefäße korrekt zu segmentieren. Während einige größere Blutgefäße visuell ersichtlich sind, werden häufig nur einzelne Voxel von vielen Blutgefäßen segmentiert.

Alle Tests mit VMTK zeigten ein schlechteres Ergebnis im Vergleich zu der groben Vorsegmentierung durch die Preprocessing-Schritte, die als Eingabe für das VMTK-Tool verwendet wurden. Besonders die verwendeten Eingabedaten und die verwendeten Parameter sind hierfür mögliche Ursachen.

5.1.1.3 BRAVE-NET

Da BRAVE-NET auf UNet basiert, sind die Grundlagen bereits durch dessen Umsetzung bekannt. Trotzdem gibt es einige signifikante Unterschiede, die die Umsetzung erschweren.

Die Daten werden in Form von Patches verarbeitet, diese müssen in einem weiteren Schritt zunächst generiert werden. Der Quellcode der Publikation enthält hierfür keine Implementation. Die damit verbundene Erstellung einer eigenen Lösung stellt eine potentielle Fehlerquelle dar. Der Code ist zudem kaum dokumentiert. Durch eine intensive Einarbeitung konnte eine lauffähige Version erzeugt werden, indem der Quellcode angepasst und erweitert wurde.

In den folgenden Tests sind durchgängig Fehler in der Pipeline des Netzwerkes aufgetreten. Diese sind auf den Einsatz eines Generators zurückzuführen. Dieser, durch die Ersteller des BRAVE-NET entwickelte Generator, wird benötigt, da Tensorflow keine direkte Implementation eines 3D-Generators hat. Dieses Problem könnte durch die Verwendung eines anderen Generators umgangen werden. Ebenfalls könnte auf den Einsatz eines Generators verzichtet werden, wenn die Daten klein genug sind, um direkt geladen zu werden. Ein weiteres Problem bestand in einem Fehler der verhindert, dass mehr als ein Batch genutzt

werden kann. Hierbei ist jeder Durchlauf fehlgeschlagen, in dem eine größere Anzahl an Volumen genutzt wurde.

Eine weitere Möglichkeit für das Auftreten dieser Probleme ist die unterschiedliche Verwendung von CPU und GPU. Die Fehlermeldungen könnten durch die Nutzung der CPU in den Tests entstehen. Die neusten Versionen von Tensorflow haben keine GPU-Unterstützung auf nativen Windows-Systemen. Um dies zu verifizieren, musste ein Test mit der Tensorflow Version 1.10 durchgeführt werden, die die erwähnte GPU-Unterstützung noch enthält. Ebenfalls müssen die Version 11.2 des CUDA Toolkit und die Version 8.1 der cuDNN Bibliothek von NVIDIA als Voraussetzung installiert werden. Zusätzlich werden die dazugehörigen Packages per Conda installiert. Dieser Test hat zu keiner Verbesserung der Probleme mit BRAVE-NET geführt. Es ist möglich, dass weiterhin die verwendeten Versionen ein Problem darstellen, aber ohne genaue Dokumentation der Entwickler von BRAVE-NET ist diese Möglichkeit nicht vollständig auflösbar.

Die Ergebnisse des BRAVE-NET sind nicht brauchbar, da eine Vielzahl an falschen Zuordnungen geschieht. Hier ist zu beobachten, dass ein großer Anteil von eigentlichen Falschpositiven Klassifizierungen von BRAVE-NET in einer Fehlermaske als True Positives erkannt werden, obwohl diese in der dazugehörigen Maske nicht als solche gekennzeichnet sind. Dabei ist die Aufteilung in Patches eine mögliche Fehlerquelle, da die Ränder der jeweiligen Patches durchgängig falsch zugeordnet werden, wie in 5.1 durch die Darstellung der als True Positive markierten Voxel dargestellt ist.

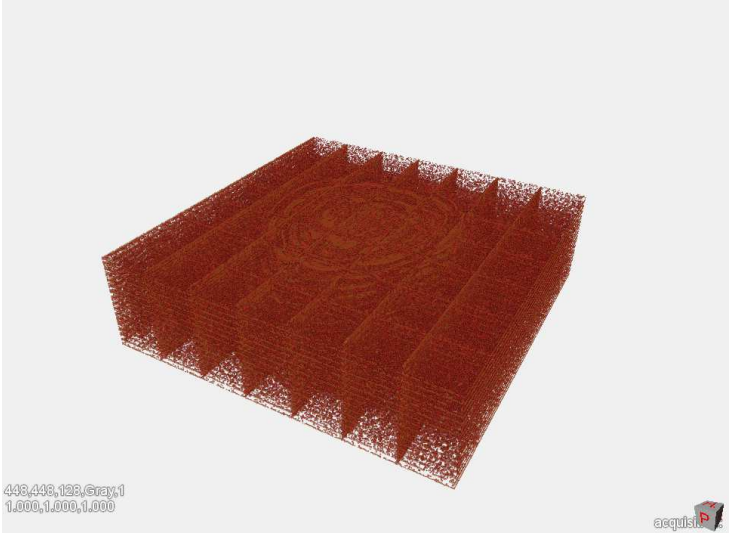


Abbildung 5.1: Durch BRAVE-NET erstellte Maske aller als richtig-positiv klassifizierten Voxel

5.1.1.4 Vesselness-Filter

Als ein Vertreter der traditionellen Ansätze für die Segmentierung von Blutgefäßen, ist der Vesselness-Filter durch die weite Verbreitung einfach einsetzbar und zu implementieren, besonders bei Nutzung von ITK. Die Ergebnisse sind, optisch und auch durch die Metriken bestätigt, deutlich besser, als jene von VMTK und BRAVE-NET. Die Ergebnisse erscheinen durchaus brauchbar, ordnen sich jedoch im Hinblick auf die Metriken qualitativ unterhalb der U-Net Segmentierungen ein. Hierfür sind sowohl segmentierte Bereiche von Venen, als auch andere Blutgefäße außerhalb des Gehirns, die nicht korrekt entfernt wurden, verantwortlich. Auch die genutzten Parameter des Filters und des Thresholdings können falsch gewählt sein. Das Thresholding durch die Otsu-Methode berechnet den Schwellenwert für die Binarisierung selbständig, während verschiedene Parameter-Werte des Vesselness-Filter getestet wurden. Bei diesen Tests fällt auf, dass - obwohl die Ergebnisse visuell schlechter erscheinen - die Verwendung von nur einem Scale mit Sigma 1.0 eine bessere Bewertung durch die Metriken erfährt. Das ist darauf zurückführbar, dass trotz der Entfernung von falsch segmentierten Bereichen, auch Voxel der korrekt zugeordneten Blutgefäße entfernt werden. Beispielsweise könnten Voxel an Rändern von Blutgefäßen durch den geringeren Wahrscheinlichkeitswert als Hintergrund deklariert werden.

Obwohl die Segmentierungsqualität nicht an die von U-net heranreicht, kann sie als eine sehr einfach zu verwendende Alternative eingesetzt werden. Besonders die Anbindung an den VR-Demonstrator über die Bibliothek SimpleITK ist nahtlos möglich. Dazu müssen lediglich die korrespondierenden Filter von ITK in SimpleITK verwendet werden. Hier ist zu beachten, dass der Vesselness-Filter als solcher nicht direkt in SimpleITK existiert, sondern im Objectness-Filter integriert ist. Damit kann, im Gegensatz zu U-Net, die gesamte Pipeline dieser Methode direkt in Unity integriert werden.

5.2 Datensatzauswahl und Probleme

Die Auswahl eines geeigneten Datensatzes ist einer der wichtigsten Bestandteile im Gebiet der medizinischen Bildverarbeitung. Besonders im Bereich der Blutgefäßsegmentierung ist die Auswahl begrenzt.

Da in diesem Fall MRA-Datensätze des Gehirns benötigt werden, können die meisten öffentlichen Datensätze nicht verwendet werden, da diese häufig nur MRT-Aufnahmen enthalten.

Die Voraussetzung eines vollständigen Gehirns in den Aufnahmen hat sich als nötiges Kriterium für dieses Projekt erwiesen, da besonders Ansätze, wie die Level-Set-Methode, auf die Strukturen der Blutgefäße angewiesen sind, um ein möglichst vollständiges Bild zu erzeugen.

Die Anforderung, dass Groundtruth-Daten für die Segmentierung vorliegen, hat die Auswahl der möglichen Datensätze signifikant begrenzt. In vielen Veröffentlichungen sind Groundtruth-Segmentierungen nicht enthalten, wodurch sie nicht für das Training der Neuralen Netze in Frage kommen. Es besteht die Möglichkeit, ein eigenes Groundtruth durch eine andere vorbereitende Segmentierung zu erzeugen. Diese Varianten der Segmentierung sind meist ungenau und können viele fehlerhafte Klassifizierungen enthalten, welche die Qualität der trainierten Netze stark vermindern können. Es ist zu erwähnen, dass der Groundtruth des ausgewählten Datensatzes ausschließlich segmentierte Arterien enthält. Folglich werden Segmentierungen ohne Training auf Basis von diesem Groundtruth eine deutlich schlechtere Bewertung durch Metriken erhalten, die den Groundtruth als Grundlage für Berechnungen verwenden.

Aufgrund von strikten Datenschutzrichtlinien werden die in diesem Gebiet häufig verwendeten Datensätze nicht veröffentlicht und sind nur durch Anträge bei Ethikkommissionen erhältlich.

Zusätzlich bieten die verfügbaren Datensätze weitere Herausforderungen durch das vorhandene Rauschen in den Bildern und die hohe Intensität des Schädelknochen. Die Lösung der genannten Herausforderungen wurde in dieser Arbeit nur teilweise behandelt, da diese Probleme als eigenständige Forschungsgebiete betrachtet werden und eine umfassende Erörterung den Umfang dieser Arbeit übersteigen würde.

5.3 Metriken

Die Auswahl von geeigneten Metriken ist ein wichtiger Bestandteil, um eine objektive Einschätzung der erzeugten Ergebnisse zu erstellen und diese miteinander vergleichen zu können.

Die Wahl der Metriken in dieser Arbeit basiert auf den Ergebnissen von früheren wissenschaftlichen Veröffentlichungen im Gebiet der zerebralen Blutgefäßsegmentierung. Die fehlende Standardisierung von verwendeten Metriken lässt keine Schlussfolgerung zu, ob die ausgewählten Metriken eine optimale Auswahl für diesen Anwendungsfall sind.

Wie bereits im vorherigen Abschnitt erwähnt, hat sich die Auswahl des Datensatzes massiv auf die Verwendung der Metriken ausgewirkt. Der auf Arterien ausgelegte Groundtruth lässt keinen fundierten Vergleich zwischen den Ergebnissen der Neuronalen Netze und anderer Lösungsansätze zu, die ohne Groundtruth auskommen.

Es wäre prinzipiell möglich, die groben Segmentierungsergebnisse aus Vorverarbeitungsschritten als Groundtruth zu verwenden. Allerdings wäre dabei das Training der Neuronalen Netze durch die vorhandenen fehlerhaften Klassifizierungen solcher ungenauer Segmentierungen stark beeinträchtigt. Zudem würde VMTK schlechtere Ergebnisse liefern, da hierfür ebenfalls eine solche Segmentierung aus Preprocessing-Schritten als Eingabe benötigt wird.

6 Zusammenfassung und Ausblick

In diesem Abschnitt wird eine Zusammenfassung insbesondere der erreichten Ergebnisse, dem Erreichen der Ziele und ein Ausblick für Weiterentwicklungsmöglichkeiten gegeben.

6.1 Zusammenfassung

Die in dieser Arbeit behandelte Problemstellung der Segmentierung von Blutgefäßen in MRT-Aufnahmen ist eine vielschichtige und komplexe Problematik der medizinischen Bildverarbeitung. Die Ziele dieser Arbeit wurden, durch die Erstellung einer funktionierenden Pipeline für die Segmentierung von Blutgefäßen, vollständig erreicht. Dafür wurde der Stand der Forschung in diesem Gebiet recherchiert und anhand dessen eine Auswahl an Datensätzen, Metriken und Methoden akquiriert. Diese wurden durch das Festlegen verschiedener Bedingungen eingegrenzt, um möglichst einfach zu implementierende Methoden mit Datensätzen und Metriken zu verwenden, die für diese Problemstellung nutzbar sind. Es wurden Pipelines für vier Methoden entwickelt, die visuell und anhand ausgewählter Metriken auf Brauchbarkeit und Qualität überprüft wurden. Qualitativ hochwertige Segmentierungsergebnisse konnten bei Verwendung eines Convolutional Neural Networks mit der U-net Architektur erreicht werden. Durch geeignete Vorverarbeitungsschritte der Eingangsdaten konnte die Qualität der Ergebnisse weiter gesteigert werden. Die daraus resultierende Pipeline wurde implementiert und ihre Verwendung im Zusammenhang mit einem 3D-Demonstrator für Virtual Reality-Brillen eingesetzt.

6.2 Ausblick

Durch die Erstellung dieser voll funktionsfähigen Pipeline wurde eine Grundlage für die weitere Betrachtung dieser Problemstellung geliefert. Die Verwendung von U-Net ist nicht optimiert. Dies ist an den Unterschieden in den Ergebnissen der Metriken im Vergleich zu den ursprünglichen Veröffentlichungen zu sehen. Aufbauend kann versucht werden, eine bessere Implementation der U-Net Architektur zu erstellen und zu verwenden. Ebenfalls ist eine Optimierung der Preprocessing-Schritte und eine Untersuchung von Postprocessing-Schritten denkbar, um die Pipeline weiter zu verbessern.

Die Methoden VMTK und BRAVE-NET müssten genauer untersucht werden, um eine bessere Funktionalität und Nutzbarkeit dieser Ansätze zu erreichen. Dabei wäre besonders BRAVE-NET durch eine bessere Anpassung des Quellcodes und einer Entwicklung eines eigenen Datengenerators weiter optimierbar.

Die Verwendung von mehreren verschiedenen Datensätzen könnte die Ergebnisse dadurch verbessern, dass Eigenschaften des verwendeten Datensatzes weniger schwer im Training des CNN gewichtet sind. Dadurch könnte besser sichergestellt werden, dass die Ergebnisse dieses Ansatzes auch auf andere Datensätze zutreffen und so weniger Overfittig vorhanden ist.

Zukünftig könnten noch mehr Metriken in die Überprüfung der Ergebnisse einbezogen werden. Besonders die Verwendung der Balanced Average Hausdorff Distance würde hier für eine Verbesserung der Aussagekraft der Metriken führen.

Im Bezug auf den VR-Demonstrator könnte eine noch bessere Integration der entwickelten Pipeline erstellt werden. Hierzu sollte entweder eine Verwendung von Integration als externer Prozess in Unity sichergestellt werden oder die Ausführung des trainierten U-Net Modell in Unity selbst durch das Package Barracuda erfolgen.

Literaturverzeichnis

[1] S. Moccia, E. De Momi, S. El Hadji, and L. S. Mattos, "Blood vessel segmentation algorithms — Review of methods, datasets and evaluation metrics," *Computer Methods and Programs in Biomedicine*, vol. 158, pp. 71–91, May 2018.

[2] K. Drechsler and C. Oyarzun Laura, "Comparison of vesselness functions for multiscale analysis of the liver vasculature," in *Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine*, Nov. 2010, pp. 1–5.

[3] S. Saha, "Transposed convolution demystified," Dez. 2018, https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/, Abgerufen am 10.04.2023.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 2015.

[5] D. Mishra, "Transposed convolution demystified," März 2020, https://www.codecademy.com/article/normalization, Abgerufen am 10.04.2023.

[6] A. Hilbert, V. I. Madai, E. M. Akay, O. U. Aydin, J. Behland, J. Sobesky, I. Galinovic, A. A. Khalil, A. A. Taha, J. Wuerfel, P. Dusek, T. Niendorf, J. B. Fiebach, D. Frey, and M. Livne, "BRAVE-NET: Fully Automated Arterial Brain Vessel Segmentation in Patients With Cerebrovascular Disease," *Frontiers in Artificial Intelligence*, vol. 3, p. 552258, Sep. 2020.

[7] S. Bériault, F. Al Subaie, K. Mok, A. F. Sadikot, and G. B. Pike, "Automatic trajectory planning of DBS neurosurgery from multi-modal MRI datasets," *Medical image computing and computer-assisted intervention: MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, vol. 14, no. Pt 1, pp. 259–266, 2011.

[8] M. R. Goni, N. I. R. Ruhaiyem, M. Mustapha, A. Achuthan, and C. M. N. Che Mohd Nassir, "Brain Vessel Segmentation Using Deep Learning—A Review," *IEEE Access*, vol. 10, pp. 111 322–111 336, 2022.

[9] Codeacademy, "Normalization," o.D., https://www.codecademy.com/article/normalization, Abgerufen am 10.11.2023.

[10] A. Dovrou, K. Nikiforaki, D. Zaridis, G. C. Manikis, E. Mylona, N. Tachos, M. Tsiknakis, D. I. Fotiadis, and K. Marias, "A segmentation-based method improving the performance of N4 bias field correction on T2weighted MR imaging data of the prostate," *Magnetic Resonance Imaging*, vol. 101, pp. 1–12, Sep. 2023.

[11] N. Tustison and J. Gee, "N4ITK: Nick's N3 ITK implementation for MRI bias field correction," *Insight J*, pp. 1–8, Jan. 2009.

[12] A. F. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Multiscale vessel enhancement filtering," in *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*, ser. Lecture Notes in Computer Science, W. M. Wells, A. Colchester, and S. Delp, Eds. Berlin, Heidelberg: Springer, 1998, pp. 130–137.

[13] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis, "Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images," *Medical Image Analysis*, vol. 2, no. 2, pp. 143–168, Jun. 1998.

[14] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset, "Model-Based Detection of Tubular Structures in 3D Images," *Computer Vision and Image Understanding*, vol. 80, pp. 130–171, Nov. 2000.

[15] M. S. Hassouna, A. A. Farag, S. Hushek, and T. Moriarty, "Cerebrovascular segmentation from TOF using stochastic models," *Medical Image Analysis*, vol. 10, no. 1, pp. 2–18, Feb. 2006.

[16] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Dec. 2015.

[17] L. Wang, C.-Y. Lee, Z. Tu, and S. Lazebnik, "Training Deeper Convolutional Networks with Deep Supervision," May 2015.

[18] A. Kumar and S. K. Jain, "Deformable models for image segmentation: A critical review of achievements and future challengesImage 1," *Computers & Mathematics with Applications*, vol. 119, pp. 288–311, Aug. 2022.

[19] S. Olabarriaga, M. Breeuwer, and W. Niessen, "Minimum Cost Path Algorithm for Coronary Artery Central Axis Tracking in CT Images," in *Lecture Notes in Computer Science*, vol. 2879, Nov. 2003, pp. 687–694.

[20] Imperial College London, "IXI Dataset – Brain Development," o.D., https://brain-development.org/ixi-dataset, Abgerufen am 10.04.2023.

[21] O. U. Aydin, A. A. Taha, A. Hilbert, A. A. Khalil, I. Galinovic, J. B. Fiebach, D. Frey, and V. I. Madai, "An evaluation of performance measures for arterial brain vessel segmentation," *BMC Medical Imaging*, vol. 21, no. 1, p. 113, Jul. 2021.

[22] E. Bullitt, D. Zeng, G. Gerig, S. Aylward, S. Joshi, J. K. Smith, W. Lin, and M. G. Ewend, "Vessel tortuosity and brain tumor malignancy: A blinded study," *Academic Radiology*, vol. 12, no. 10, pp. 1232–1240, Oct. 2005.

[23] A. Hilbert, V. I. Madai, E. M. Akay, O. U. Aydin, J. Behland, J. Sobesky, I. Galinovic, A. A. Khalil, A. A. Taha, J. Wuerfel, P. Dusek, T. Niendorf, J. B. Fiebach, D. Frey, and M. Livne, "Ground truth labels for "BRAVE-NET: Fully Automated Arterial Brain Vessel Segmentation In Patients with Cerebrovascular Disease"," Jul. 2020.

Selbständigkeitserklärung gem. § 13 Absatz 5 MPO

Hiermit versichere ich, Julian Dullinger, dass ich die vorliegende Masterarbeit mit dem Titel

Planung und Entwicklung einer Pipeline für die Segmentierung von zerebralen Blutgefäßen in MRT-Bildsequenzen

selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Zwickau, 14.11.2023

Ort, Datum

J. Dullinger

Unterschrift