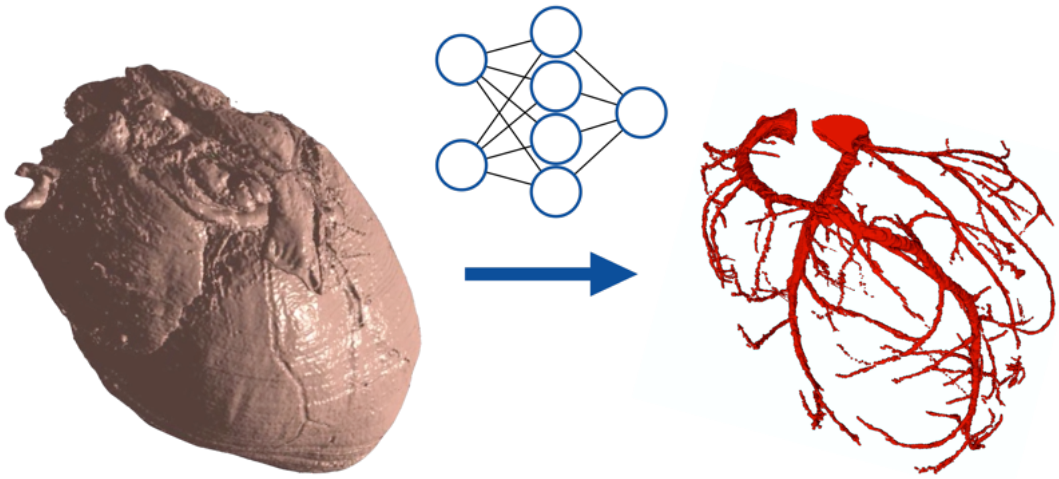
Institut für
Mechatronische Systeme
Leibniz Universität Hannover

Leibniz
Universität
Hannover

Medizinische Hochschule
Hannover

3D-Gefäßbaumsegmentierung mittels neuronaler Netze



Masterarbeit M-01/20-919

János Somogyi
Matrikelnummer 10002896

Hannover, 15. Januar 2020

Erstprüfer	Prof. Dr.-Ing. Tobias Ortmaier
Zweitprüfer	Prof. Dr.-Ing. Bodo Rosenhahn
Betreuer	Max-Heinrich Laves, M.Sc.
	Petrissa Zell, M.Sc.

Ich, János Somogyi, versichere hiermit, dass die vorliegende Arbeit selbstständig verfasst wurde, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden, alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen hat.

Hannover, 15. Januar 2020

(János Somogyi)

Institut für Mechatronische Systeme
imes Leibniz Universität Hannover

Leibniz Universität Hannover

Masterarbeit

János Somogyi, Matr.-Nr. 1002896

3D-Gefäßbaumsegmentierung mittels neuronaler Netze

3D Vessel Tree Segmentation with means of Neural Networks

Allgemeines Für die komplette, biologische Ersetzung krankhafter Organe steht aktuell nur eine Transplantation zur Verfügung, was jedoch einen Organspender voraussetzt. Da die meisten Organe im Menschen jedoch nicht entbehrlich sind, sind Transplantationen oft mit dem vorhergehenden Exitus des Organspenders verbunden. Aus diesem Grund ist es in der Medizin angestrebt, implantierbare Organe auf künstlichem Weg herzustellen. Damit entsprechende Verfahren dazu entwickelt werden können, muss zunächst der genaue anatomische Aufbau des jeweiligen Organs bekannt sein. Dazu muss im Vorfeld eine Modellierungsphase stattfinden. Das in dieser Arbeit betroffene Modell soll anhand von Mikroskopaufnahmen eines ganzen Schweineherzens mit Hilfe entsprechender Segmentierungsmethoden erstellt werden.

Aufgabe Ziel dieser Arbeit ist es, in der Modellierungsphase des Gefäßbaumes einen Beitrag zu leisten. In dieser Arbeit sollen Techniken aus dem Bereich des Machine Learning (ML) herangezogen werden. Insbesondere sollen im Rahmen der Arbeit folgende Punkte bearbeitet werden:

- Literaturrecherche über synthetische Gefäßbaumgenerierung sowie über ML-Segmentierungsverfahren
- Erstellung synthetischer, 3-dimensionaler Daten, die im Trainingsprozess benutzt werden können.
- Implementierung eines ML-Segmentierungsverfahrens, das Voxel lokal nach Gefäß/Hintergrund unterscheiden sowie Gefäßmittellinien und Bifurkationen identifizieren kann
- Validieren an den echten, aus den Mikroskopaufnahmen stammenden Daten
- Dokumentation

Die Bearbeitungszeit beträgt 6 Monate.
Ausgabe der Aufgabenstellung: 15.07.2019, Abgabe der Arbeit: 15.01.2020
Betreuer: Max-Heinrich Laves, M. Sc.

Prof. Dr.-Ing. T. Ortmaier (Erstprüfer)

Student

Prof. Dr.-Ing. B. Rosenhahn (Zweitprüfer)

Kurzfassung

Versagt ein menschliches Organ, steht nach heutigem Standard der Medizin nur eine begrenzte Auswahl von Möglichkeiten zur Verfügung. Die regenerative Medizin versucht unter anderem diese Auswahl mit synthetisch hergestellten Geweben sowie Organen zu erweitern. Für eine synthtetische Herstellung ist die möglichst genaue anatomische Kenntnis des jeweiligen Organs unentbehrlich. Ein wichtiger Aspekt der Anatomie eines jeden Organs ist der Aufbau seines Gefäßsystems. In dieser Arbeit wird ein Beitrag zur Gefäßsegmentierung aus mikroskopischen Bilddaten geleistet basierend auf neuronalen Netzen.
Die vorhandenen mikroskopischen Bilddaten sind histologische Schnitte eines Schweineherzens. Da das Labeln dieser Bilder einen enormen Zeitaufwand nach sich ziehen würde, werden die Daten in dieser Arbeit für das Training der neuronalen Netze synthetisch hergestellt. Dadurch können zudem zu jedem Zeitpunkt beliebig viele Daten generiert werden. Während der synthetischen Datenherstellung wächst ein Gefäßbaum iterativ aus einem Startgefäßsegment aus und füllt ein vorgegebenes Volumen. Das Verfahren basiert auf den Algorithmen von [Sch+12]. Es wird jedoch als Ziel gesetzt, hinsichtlich der Radien ein möglichst großes Spektrum der im Herzen vorkommenden Gefäße synthetisch nachzubilden. Im Gegensatz zu [Sch+12] würde für den Fall, alle diese Radien auflösen zu können, ein rechentechnisch nicht tragbares Gesamtvolumen zustande kommen. Aus diesem Grund werden neue Algorithmen entwickelt, die von der Baumgenerierung bis zum Sampling den Schritt umgehen, den synthetischen Gefäßbaum als Volumen zu bearbeiten. Unter anderem arbeiten diese Algorithmen mit der Häufigkeitsverteilung der Knotenpunkte, die den Gefäßbaum als Graphen repräsentieren. Des Weiteren wird das Wachstumsverhalten von Gefäßen in der Nähe des zu füllenden Volumens weiter entwickelt, sodass sich realistischere Gefäßbäume ergeben.
Im nächsten Schritt werden in Form von Ausschnitten aus dem generierten Gefäßbaum Samples definiert. Ein hierzu entworfenes Algorithmus findet und wandelt den relevanten Teilgraphen des Gefäßbaumes in ein Volumen um. Das Volumen wird anschließend mit drei verschiedenen Arten von Rauschen versetzt. Diese beinhalten Deformationen, Unterbrechungen der Gefäßstrukturen und das Hinzufügen von Störobjekten. Mit den so erhaltenen Samples wird ein fully convolutional Netz traininert. Die Netzarchitektur basiert auf der Arbeit in[Tet+18]. Das Netz wird anschließend an den synthetischen Daten und zusätzlich auf einem echten Datenset, das aus manueller Bildsegmentierung gewonnen wird. Für die Gefäßerkennung in verrauschten Daten beträgt der Dice-Koeffizient am synthetischen Evaluierungsset $0{,}5046 \pm 0{,}2268$ und am echten Evaluierungsset $0{,}7213 \pm 0{,}1404$.

Inhaltsverzeichnis

Bildverzeichnis

Tabellenverzeichnis

1 Einleitung

Im ersten Kapitel wird zunächst der Leser in die Thematik eingeleitet und die Gründe sowie die Motivation dargelegt, die zur Entstehung dieser Masterarbeit geführt haben. Im Anschluss wird der Aufbau der hier zu lesenden Ausarbeitung vorgestellt.

1.1 Motivation

Der menschliche Körper ist ein hochkomplexes System. Zur Funktion leisten verschiedene Organe ihren Beitrag. Dabei definiert die Funktion eines jeden Organs der Gewebetyp, aus dem es aufgebaut ist. [Lom03] Erkrankungen oder Traumata können zur Beeinträchtigung oder zum Erlöschen der Funktionsfähigkeit von Gewebe bzw. somit in bestimmten Fällen auch ganzer Organe führen. Ist ein Organ zur Funktion nicht mehr fähig, steht nach heutigem Standard der Medizin nur eine begrenzte Auswahl an Möglichkeiten zur Verfügung. Eine Möglichkeit ist die Transplantation von Organen entweder aus einem anderen Menschen oder aus einem Tier. Im ersteren Fall spricht man von einer *Allotransplantation*, im anderen Fall von einer *Xenotransplantation*. Dadurch dass, bei der Xenotransplantation das zu implantierende Organ nicht menschlichen Ursprungs ist, treten viele Effekte auf, die zu einem Misserfolg führen können, sodass diese Möglichkeit lediglich als Forschungsgegenstand zählt. [Coc00] Allotransplantation bildet heute eine erprobte Behandlungsart bei Organversagen in der Endphase. [LW10] Das Spenden menschlicher Organe ist in den meisten Fällen mit dem Exitus des Spenders verbunden. [Sec09]
Da Spender nur in einer sehr begrenzten Anzahl zur Verfügung stehen, werden in der Medizin deutlich mehr Organe gebraucht als zur Verfügung stehen. Dies führt zu Engpässen bei Behandlungen mittels Organtransplantation. [AJM16] Diese Engpässe werden in Zukunft möglicherweise wegen der alternden Bevölkerung weiter zunehmen, sodass der Organmangel eine Herausforderung der medizinischen Forschung darstellt. [BS19]
Eine Antwort auf diese Herausforderung versucht die regenerative Medizin bzw. das Forschungsfeld *Tissue Engineering* zu geben. [Lev+09] Die Methoden bestehen u.a. aus Stammzellentransplantation, Transplantation künstlich, unter Laborbedingungen hergestellter Gewebe und das Anregen körpereigenen Zellen zur Regeneration. Forschungen in der regenerativen Medizin finden auch an der *Herz-, Thorax-, Transplantations- und Gefäßchirurgie* (*HTTG*) der *Medizinischen Hochschule Hannover* (*MHH*) statt, vor allem im Rahmen des Exzellenzclusters

REBIRTH.
Für die Funktion eines jeden Organs ist die Versorgung der Zellen mit Nährstoffen unentbehrlich, sodass für die künstliche Herstellung von Gewebe im Tissue Engineering ein möglichst genaues Bild von der Anatomie des Gefäßsystems des jeweiligen Organs erwünscht ist. In Zusammenarbeit mit dem *Institut für Informationsverarbeitung* (*TNT*) der *Leibniz Universität Hannover* werden hierzu Mikroskopaufnahmen eines Schweineherzens verarbeitet und analysiert.
Ziel dieser Masterarbeit ist es, mittels Bildsegmentierung der Gewinnung anatomischer Daten über das Gefäßsystems beizutragen. In einer Vorarbeit wurden die histologischen Schnitte des Schweineherzens mit einem Mikroskop hochauflösend eingescannt. Es wurden zudem die Grundsteine für einen komplexen Segmentierungsalgorithmus gelegt, der so entworfen wurde, dass er mit der extrem großen Datenmenge ($\approx$1,7 TB) umgehen kann.
Ein Ansatz für die Identifizierung von Gefäßen in Bilddaten ist das Heranziehen von Machine Learning Algorithmen wie beispielsweise von neuronalen Netzen. Damit neuronale Netze eine gute Vorhersage treffen können, werden für den Lernprozess Daten möglichst guter Qualität gebraucht, in denen die später zu erkennenden Muster ausgeprägt vertreten sind. Sind solche Daten – wie im Fall dieser Arbeit – nicht oder lediglich in einem für das Training nicht ausreichenden Maße vorhanden, besteht die Möglichkeit Daten synthetisch herzustellen, was den ersten Aufgabenteil ausmacht.
Im zweiten Teil werden verschiedene Netzarchitekturen untersucht und mit den zuvor synthetisch hergestellten Daten trainiert. Die implementierten neuronalen Netze werden anschließend u.a. auch an echten Daten validiert.
Bei der Segmentierung bestehen mehrere Teilaufgaben, die zur Gefäßidentifikation gehören: Es soll nicht nur erkannt werden, ob ein Voxel im 3D-Volumen ein Gefäß darstellt, sondern auch die Mittellinien der Gefäße sowie Verzweigungspunkte (Bifurkationen) sollen registriert werden. Anschließend sollen die Ergebnisse dieser Arbeit mit dem bestehenden, unter Entwicklung stehendem Segmentierungslgorithmus zusammengeführt werden.

2 Grundlagen neuronaler Netze

Das zweite Kapitel erläutert die Grundlagen, die für das Verständnis der darauffolgenden Kapiteln notwendig sind. Zunächst werden die allgemeinen Grundsätze neuronaler Netze dargestellt. Das Konzept eines künstlichen Neurons wird vorgestellt und es wird geschildert wie ein Netzwerk von Neuronen Muster erkennen und Zusammenhänge lernen kann. Neuronale Netze finden eine bedeutende Anwendung in der Segmentierung von Bildern, was im Anschluss thematisiert wird. Zum Schluss werden die Parameter dargestellt, die den Trainingserfolg eines neuronalen Netzes beeinflussen. Diese werden Hyperparameter genannt.

2.1 Allgemeines

Das Zentrum im Menschlichen Körper, das auf die unterschiedlichsten Inputs aus der Umwelt eine Antwort, einen Output befiehlt, ist das Gehirn. Wie auf die Inputs jeweils zu reagieren ist, entscheidet es basierend auf Ereignissen aus der Vergangenheit, es kann aus diesen lernen. Die Grundbausteine des Gehirns sind die Neuronen; ihre hochkomplexe Vernetzung ermöglicht dem Gehirn die kompliziertesten Zusammenhänge zu erlernen [TS97].
Neuronalen Netzen, wie dem menschlichen Gehirn, sind auch die künstlichen neuronalen Netze nachempfunden. Ihre Grundbausteine sind die künstlichen Neuronen, die in ihrem Aufbau den biologischen Neuronen ähneln [DS13] (vgl. Bild 2.1).

2.1.1 Das künstliche Neuron

Ein Neuron besteht aus biologischer Sicht aus dem Zellkern, den Dendriten und dem Axon. Mit letzteren beiden kann es sich mit den anderen Neuronen verbinden. Über die Dendrite werden Signale empfangen, sie werden im Zellkern verarbeitet und über das Axon weitergeleitet. Die Weiterleitung zwischen dem Axon und den Dendriten des nächsten Neurons erfolgt über die sogenannte Synapse. Die Verbindungen zwischen den Neuronen sind nicht statisch, sie können abgebrochen, wieder aufgebaut oder es können Neue geschaffen werden. [TS97]
Nach [MP43] kann dieses biologische Konzept vereinfacht wie folgt modelliert werden:
Grundsätzlich besteht ein solches künstliches Neuron, das auch als McCulloh-Pitts-Neuron bezeichnet wird, aus drei Komponenten:

a) gewichtete Inputs x_i, w_i

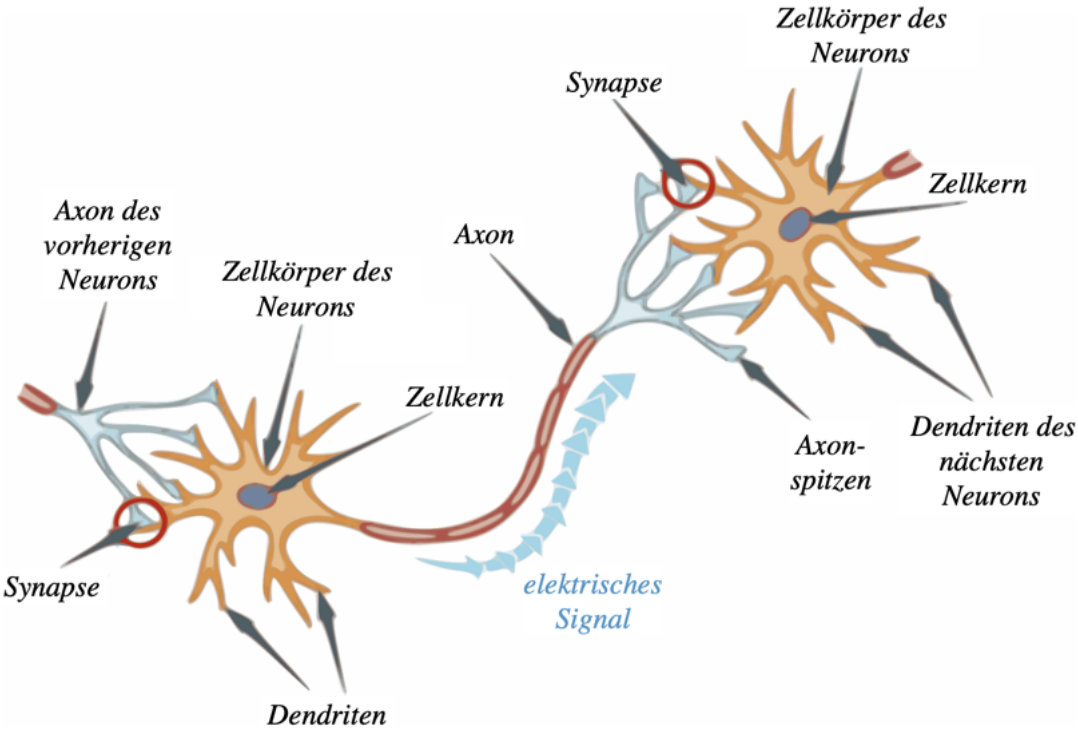


Bild 2.1: Aufbau und Vernetzung eines Neurons [Mah17]

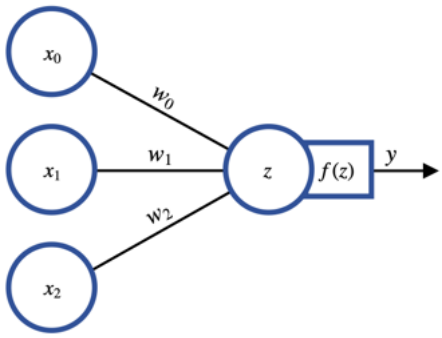


Bild 2.2: Das künstliche Neuron von McCulloch & Pitts [Vin18]

b) Addierer z

c) Aktivierungsfunktion $f(z)$

Ein Input stellt ein von einem anderen Neuron über die Dendriten einkommendes Eingangssignal dar. Die Gewichte w_i, mit denen die Eingänge multipliziert werden, dienen einer Art Priorisierung der Eingänge. Inputs mit einem größeren Gewicht beeinflussen den Output in

einem größeren Maß und Eingänge, die mit einem kleineren Gewicht versehen werden, tragen nur wenig zum Output bei. Der Addierer z steht für den Zellkörper des Neurons und fasst die gewichteten Eingangssignale zu einem Signal zusammen. Mathematisch lässt sich dies mit

$$\boldsymbol{x} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} \quad \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{n-1} \end{pmatrix} \tag{2.1}$$

als ein Skalarprodukt der Eingänge und Gewichte formulieren.

$$z = \boldsymbol{x} \cdot \boldsymbol{w} = \sum_{i=0}^{n-1} x_i \cdot w_i \tag{2.2}$$

Die Aktivierungsfunktion gibt abhängig vom Input z einen Output zurück. Ursprünglich schlugen McCulloh und Pitts folgende Schwellwertfunktion vor.

$$f(z) = \begin{cases} 1, & \text{falls } z > \theta \\ 0, & \text{falls } z \leqslant \theta \end{cases} \tag{2.3}$$

In diesem Fall gibt es erst dann einen Output $\neq 0$, wenn die Summe der gewichteten Eingangssignale z den Schwellwert θ überschreitet. Im Abschnitt 2.2.1 wird noch näher auf verschiedene Aktivierungsfunktionen eingegangen.
Zur Einstellung der Gewichte w_i werden dem Neuron beispielhaft Eingang-Ausgang-Paare vorgelegt. Dabei werden die Gewichte so eingestellt, dass sie das in den Eingang-Ausgang-Paaren vorhandene Muster widerspiegeln. Mit anderen Worten wird das Neuron mit dem *Trainingsdatenset* $X_{\text{train}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n_{\text{train}}-1}\}$, $Y_{\text{train}} = \{y_1, y_2, \cdots, y_{n_{\text{train}}-1}\}$ trainiert. Sind die Gewichte entsprechend eingestellt, kann bei einem Test mit einem zufälligen Eingangsvektor $\boldsymbol{x}_{\text{test}}$ der Ausgang y_{pred} möglichst gemäß dem Muster, das während des Trainings gelernt wurde, vorhergesagt werden. Die Vorhersage y_{pred} wird dann mit dem tatsächlichen Ausgang y_{test} verglichen ums so eine Aussage über die Genauigkeit der Vorhersage zu treffen. Das Datenset, mit dem das Neuron auf dieser Weise ausgewertet wird, wird *Testdatenset* genannt. Analog zum Trainingsdatenset gilt: $X_{\text{test}} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{n_{\text{test}}-1}\}$, $Y_{\text{test}} = \{y_1, y_2, \cdots, y_{n_{\text{test}}-1}\}$ bzw. für die Vorhersagen $Y_{\text{pred}} = \{y_1, y_2, \cdots, y_{n_{\text{pred}}-1}\}$ wobei $n_{\text{pred}} = n_{\text{test}}$. [Vin18; DS13]

2.1.2 Das neuronale Netz [Vin18; DS13]

Vernetzt man mehrere künstliche Neuronen, entsteht ein künstliches neuronales Netz (im Folgenden neuronales Netz bezeichnet), das auch komplexe Zusammenhänge erlernen kann. Ein

neuronales Netz kann dazu benutzt werden, kontinuierliche oder diskrete Größen anhand der erlernten Zusammenhänge vorherzusagen. Im ersten Fall spricht man von einer *Regression*, im zweiten von einer *Klassifikation*. Durch Regression wird bspw. der Preis eines Artikels prognostiziert, bei einer Klassifikation wird dahingegen ein Input einer Klasse zugeteilt. Beispielhaft ist in Bild 2.3 ein neuronales Netz dargestellt.

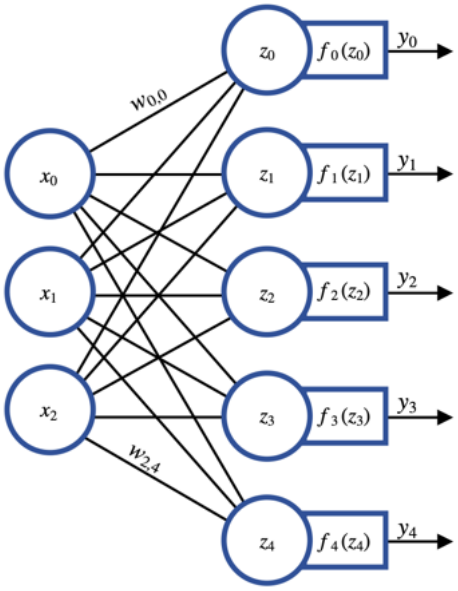


Bild 2.3: Einfaches Netz von Neuronen mit $m = 3$ Ein- und $n = 5$ Ausgängen [Vin19]

Im Beispiel sind alle Ausgänge jeweils mit allen Eingängen verbunden, in der Fachliteratur wird eine solche Anordnung der Neuronen als *fully connected layer* oder *dense layer* bezeichnet. Die Bezeichnung $w_{i,j}$ steht für das Gewicht des i-ten Eingangssignals, das zum j-ten Ausgang beiträgt. Bei m Eingängen und n Ausgängen bei einem Layer gilt für die Indizes $0 \leqslant i \leqslant m$ und $0 \leqslant j \leqslant n$. Die Gewichte $w_{i,j}$ können in einer Gewichtsmatrix $\boldsymbol{W}^{m \times n}$ zusammengefasst werden:

$$\boldsymbol{W} = \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n-1} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m-1,0} & w_{m-1,1} & \cdots & w_{m-1,n-1} \end{pmatrix}. \tag{2.4}$$

Für die Berechnung des Ausgangsvektors $\boldsymbol{y}$ gilt damit:

$$\boldsymbol{z} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{x} \Leftrightarrow \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ z_{n-1} \end{pmatrix} = \begin{pmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,m-1} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n-1,0} & w_{n-1,1} & \cdots & w_{n-1,m-1} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{m-1} \end{pmatrix} \tag{2.5}$$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{z}) \Leftrightarrow \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} f_0(z_0) \\ f_1(z_1) \\ \vdots \\ f_{n-1}(z_{n-1}) \end{pmatrix} \tag{2.6}$$

Verknüpft man wie in Bild 2.4 mehr als zwei Layers, so erhält man zwischen dem Inputlayer und dem Outputlayer sog. *hidden Layers*. Sind mehrere hidden Layers in einem neuronalen Netz verbaut, spricht man von einem *deep neural network* bzw. von *deep learning*.

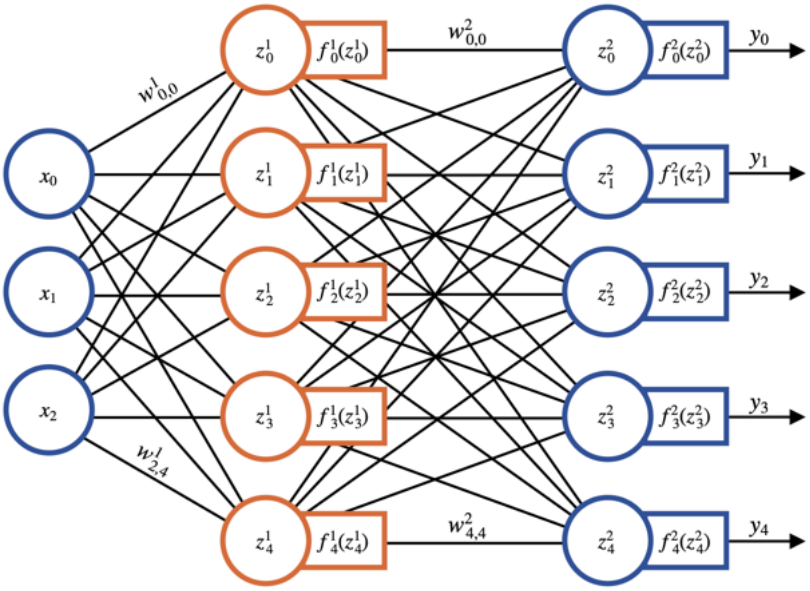


Bild 2.4: Neuronales Netz mit einem hidden Layer ($L = 2$)

Je mehr hidden Layers in einem neuronalen Netz zu finden sind, desto komplexere Zusammenhänge können gelernt werden. Für nur einen Eingangs- sowie Ausgangslayer gilt nach Gleichung (2.5) und Gleichung (2.6) insgesamt: $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{x})$. Analog lässt sich der Ausgang $\boldsymbol{y}$ bei der Verkettung von L Layers insgesamt wie folgt bestimmen.

$$\boldsymbol{y} = \boldsymbol{f}^L(\boldsymbol{W}^{L^{\mathrm{T}}} \boldsymbol{f}^{L-1}(\boldsymbol{W}^{L-i^{\mathrm{T}}} \cdots \boldsymbol{f}^1(\boldsymbol{W}^{1^{\mathrm{T}}} \boldsymbol{x}) \cdots)) \tag{2.7}$$

2.1.3 Das Training [McG+; DS13]

Den Ausgang $\boldsymbol{y}$ für einen Eingang $\boldsymbol{x} = \boldsymbol{x}_{\mathrm{train}}$ und eine Gewichtsmatrix $\boldsymbol{W}^l$ soll $\boldsymbol{t}$ bezeichnen. Beim Training eines neuronalen Netzes ist das Ziel in jedem Layer l jene Konfiguration von $\boldsymbol{W}^l$ zu erhalten, die den Zusammenhang zwischen den Eingängen $\boldsymbol{x}$ und den Ausgängen $\boldsymbol{y}$ möglichst genau beschreibt. Um die entsprechende $\boldsymbol{W}^l$ zu finden, werden die Gewichte $w_{i,j}$ entsprechend der Abweichung zwischen $\boldsymbol{t}$ und $\boldsymbol{y}_{\mathrm{train}}$ angepasst. Ist ein Gewicht $w_{i,j}$ für einen großen Teil der Abweichung verantwortlich, so soll es entsprechend stark angepasst werden. In welchen Maßen welche Gewichte zur Abweichung beitragen, lässt sich durch die Berechnung von $\boldsymbol{t}$ (*forward propagation*) und anschließender Rückführung der Abweichungen (*backward propagation* oder *backpropagation*) ermitteln.
Folgend der bisherigen Nomenklatur und mit der zusätzlichen Bezeichnung o_j^l für den Output des j-ten Neurons im l-ten Layer setzt sich der Algorithmus wie folgt zusammen:

1. **Forward Propagation:** Für jedes Neuron werden die Größen z_j^l und o_j^l berechnet und gespeichert. Dabei gilt für das Neuron $j \in \{1, 2, \ldots, n^l\}$ im fully connected hidden Layer $l \in \{2, 3, \ldots, L-1\}$:

$$z_j^l = \sum_{i=0}^{n^{l-1}} o_i^{l-1} \cdot w_{i,j}^l \tag{2.8}$$

$$o_j^l = f_j^l(z_j^l) \tag{2.9}$$

   Für das erste Layer $l = 1$ nach dem Inputlayer gilt:

$$z_j^1 = \sum_{i=0}^{n^0} x_i \cdot w_{i,j}^1 \tag{2.10}$$

$$o_j^1 = f_j^1(z_j^1) \tag{2.11}$$

   Für das letzte, das Ausgangslayer $l = L$ gilt:

$$z_j^L = \sum_{i=0}^{n^{L-1}} o_i^{L-1} \cdot w_{i,j}^L \tag{2.12}$$

$$t_j = o_j^L = f_j^L(z_j^L) \tag{2.13}$$

2. **Backward Propagation:** $E(t_j, y_{\mathrm{train},j})$ definiert eine Fehlerfunktion, sie misst die Ab-

weichung zwischen dem errechneten Ausgang t_j und dem tatsächlichen Ausgang $y_{\text{train},j}$. Für das Updaten der Gewichte wird die Ableitung $\frac{\partial E}{\partial w_{i,j}^l}$ nach jedem Gewicht $w_{i,j}^l$ gebraucht. Dabei bezeichnet $w_{i,j}^l$ jenes Gewicht, das das Neuron i im Layer $l-1$ mit dem Neuron j im Layer l verbindet, vgl. dazu auch Bild 2.5.

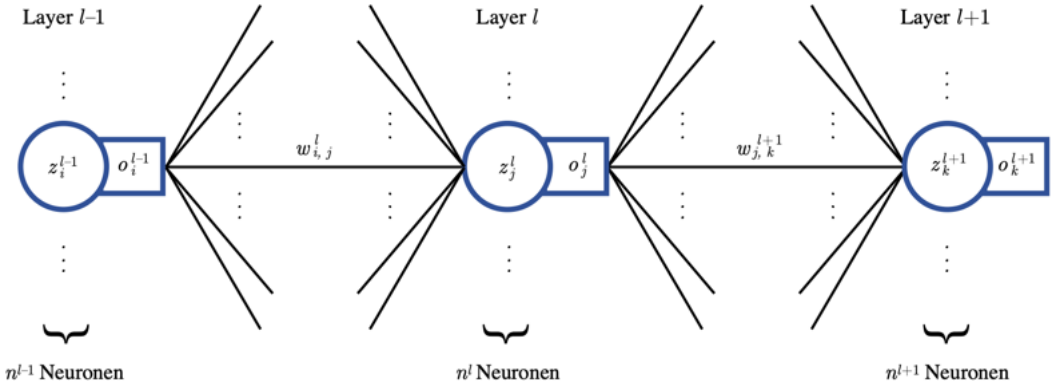


Bild 2.5: Beliebiges Layer l sowie die Layer unmittelbar davor und danach

Die gesuchten Ableitungen können für $l \in \{1, 2, \ldots, L-1\}$ mit Hilfe der Kettenregel ermittelt werden:

$$\frac{\partial E}{\partial w_{i,j}^l} = \frac{\partial E}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{i,j}^l} \tag{2.14}$$

Für die zweite Komponente gilt:

$$\frac{\partial z_j^l}{\partial w_{i,j}^l} = \frac{\partial}{\partial w_{i,j}^l} z_j^l = \frac{\partial}{\partial w_{i,j}^l} \sum_{i=0}^{n^{l-1}} o_i^{l-1} \cdot w_{i,j}^l = o_i^{l-1} \tag{2.15}$$

Die erste Komponente kann wiederum mit der Kettenregel umgeformt werden:

$$\frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial \boldsymbol{z}^{l+1}} \frac{\partial \boldsymbol{z}^{l+1}}{\partial z_j^l} \tag{2.16}$$

Mit $\boldsymbol{z}^{l+1} = \left(z_1^{l+1}, \quad z_2^{l+1}, \quad \ldots, \quad z_k^{l+1}, \quad \ldots, \quad z_{n^{l+1}}^{l+1}\right)$ kann das Matrizenprodukt der beiden Ableitungsterme in Gleichung (2.16) in eine Summenform umgeschrieben werden:

$$\frac{\partial E}{\partial z_j^l} = \sum_{k=1}^{n^{l+1}} \frac{\partial E}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \tag{2.17}$$

Der erste Term in der Summe kann durch rekursive Anwendung von Gleichung (2.17)

berechnet werden. Der zweite Term lässt sich mit erneuter Anwendung der Kettenregel weiterhin bestimmen zu:

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = \frac{\partial z_k^{l+1}}{\partial o_k^{l+1}} \frac{\partial o_k^{l+1}}{\partial z_j^l} \tag{2.18}$$

Die erste Ableitung in Gleichung (2.18) ergibt sich ähnlich zu Gleichung (2.15) zu:

$$\frac{\partial z_k^{l+1}}{\partial o_j^l} = \frac{\partial}{\partial o_j^l} z_k^{l+1} = \frac{\partial}{\partial o_j^l} \sum_{j=0}^{n^l} o_j^l \cdot w_{j,k}^{l+1} = w_{j,k}^{l+1} \tag{2.19}$$

Die zweite lässt sich unter Verwendung von Gleichung (2.9) bestimmen:

$$\frac{\partial o_k^l}{\partial z_k^l} = \frac{\partial}{\partial z_k^l} o_k^l = \frac{\partial}{\partial z_k^l} f_k^l(z_k^l) = f_k^{l\prime}(z_k^l) \tag{2.20}$$

Damit sind alle Terme bestimmt und der Einfluss eines jeden Gewichts $w_{i,j}^l, l \in \{1, 2, \ldots, L-1\}$ auf den Fehler E kann bestimmt werden. Die Gleichungen (2.14) bis (2.20) lassen sich folgender Weise zusammenfassen:

$$\frac{\partial E}{\partial w_{i,j}^l} = \left[\sum_{k=1}^{n^{l+1}} \frac{\partial E}{\partial z_k^{l+1}} w_{j,k}^{l+1} f_j^{l\prime}(z_j^l) \right] o_i^{l-1} \tag{2.21}$$

$$\frac{\partial E}{\partial z_j^l} = \sum_{k=1}^{n^{l+1}} \frac{\partial E}{\partial z_k^{l+1}} w_{j,k}^{l+1} f_j^{l\prime}(z_j^l) =: \delta_j^l \tag{2.22}$$

Mit der Einführung von δ_j^l lässt sich Gleichung (2.21) umschreiben zu:

$$\frac{\partial E}{\partial w_{i,j}^l} = \delta_j^l o_i^{l-1} = \left[\sum_{k=1}^{n^{l+1}} \delta_k^{l+1} w_{j,k}^{l+1} f_j^{l\prime}(z_j^l) \right] o_i^{l-1} \tag{2.23}$$

Durch die Rekursion von δ_j^l ist es noch notwendig den Term $\delta_j^L = \frac{\partial E}{\partial z_j^L}$ zu bestimmen. Durch Anwendung der Kettenregel gilt:

$$\frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial o_j^L} \frac{\partial o_j^L}{\partial z_j^L} \tag{2.24}$$

Da nach Gleichung (2.13) $o_j^L = t_j$, ist der erste Term die partielle Ableitung der Fehler-

funktion $E(t_j, y_{\text{train},j})$:

$$\frac{\partial E}{\partial o_j^L} = \frac{\partial E(t_j, y_{\text{train},j})}{\partial t_j} \tag{2.25}$$

Der zweite Term ergibt sich gemäß Gleichung (2.20) zu:

$$\frac{\partial o_j^L}{\partial z_j^L} = \frac{\partial}{\partial z_k^L} o_k^L = \frac{\partial}{\partial z_k^L} f_k^L(z_k^L) = f_k^{L\prime}(z_k^L) \tag{2.26}$$

Somit gilt für δ_j^L

$$\delta_j^L = \frac{\partial E(t_j, y_{\text{train},j})}{\partial t_j} f_k^{L\prime}(z_k^L) \tag{2.27}$$

Das Vorgehen unter dem Schritt Backward Propagation lässt sich wie folgt zusammenfassen:

a) $\frac{\partial E}{\partial w_{i,j}^L}$ bestimmen mit

$$\delta_j^L = \frac{\partial E(t_j, y_{\text{train},j})}{\partial t_j} f_k^{L\prime}(z_k^L) \tag{2.28}$$

$$\frac{\partial E}{\partial w_{i,j}^L} = \delta_j^L o_i^{L-1} \tag{2.29}$$

b) $\frac{\partial E}{\partial w_{i,j}^l}$ sukzessiv von $l = L - 1$ bis $l = 1$ bestimmen mit

$$\delta_j^l = \sum_{k=1}^{n^{l+1}} \delta_k^{l+1} w_{j,k}^{l+1} f_j^{l\prime}(z_j^l) \tag{2.30}$$

$$\frac{\partial E}{\partial w_{i,j}^l} = \delta_j^l o_i^{l-1} \tag{2.31}$$

3. **Gewichte Updaten:** Die Gewichte werden nach jedem Iterationsschritt angepasst. Pro Schritt kann entweder ein Samplepaar aus dem Trainingsdatenset ausgewertet werden oder aber mehrere und die errechneten $\frac{\partial E}{\partial w_{i,j}^l}$ gemittelt werden. Das Updaten erfolgt proportional zu $\frac{\partial E}{\partial w_{i,j}^l}$. Die Proportionalitätskonstante ist die Lernrate α, sie bestimmt in welchem Maße die Gewichte zum Iterationsschritt $s + 1$ des Trainings angepasst werden sollen:

$$w_{i,j}^l\big|_{s+1} = w_{i,j}^l\big|_s - \Delta w_{i,j}^l\big|_s \tag{2.32}$$

$$\Delta w_{i,j}^l\big|_s = \alpha \frac{\partial E}{\partial w_{i,j}^l}\bigg|_s, \quad l \in \{1, 2, \ldots, L\} \tag{2.33}$$

Die Variabe, die die Anzahl von Samplepaaren $(\boldsymbol{x}_i, \boldsymbol{y}_i)$, die in einem Iterationsschritt ausgewer-

tet werden, bestimmt, wird als *batch size* bezeichnet. Bei insgesamt $n_{\text{train}} = |X_{\text{train}}| = |Y_{\text{train}}|$ Samplepaaren und einer Batchsize von b ergibt sich die Anzahl von Iteartionsschritten n_{it} zu:

$$n_{it} = \frac{n_{\text{train}}}{b} \tag{2.34}$$

Diese Anzahl wird benötigt, um einmal das komplette Trainingsdatenset auszuwerten. Das Trainingsdatenset kann auch mehrmals durchlaufen werden, jeden solchen Durchlauf bezeichnet man als eine Epoche.

2.2 Hyperparameter

Der Lernerfolg und die Genauigkeit von neuronalen Netzen hängt von vielen Faktoren ab. Mit dem Finden von passenden Werten für die Gewichte $w_{i,j}^{l}$ können durch das Training komplexe Zusammenhänge gelernt werden. Einflussgrößen wie die Netzarchitektur und die Lernrate gehören zu den sogenannten *Hyperparametern*, die vor dem Training festgelegt werden müssen. Während die Gewichtswerte $w_{i,j}^{l}$ gelernt werden können, müssen die Hyperparameter vorher explizit eingestellt werden. Die Einstellung der Hyperparameter kann empirisch über *trial and error*, auf Erfahrungswerten basierend oder aus systematischen, mit einem Validierungsdatensatz ausgewerteten Versuchen abgeleitet werden. Hyperparameter bestimmen entweder die Netzarchitektur oder den Trainingsprozess. Die wichtigsten Hyperparameter, eingeteilt in diese zwei Gruppen, sind in Tabelle 2.1 zusammengefasst. [Rad17; Alt19; Bok19; MMJ19]

Tabelle 2.1: Hyperparameter

Architektursparameter	**Trainingsparameter**
Anzahl von hidden Layers	Lernrate
Anzahl von Neuronen	Momentum
Dropout	Anzahl von Epochen
Gewichtinitialisierung	Batchsize
Aktivierungsfunktion	

Bei der Wahl der Hyperparameter ist vor allem darauf zu achten, dass weder *Underfitting*, noch *Overfitting* auftritt. Mit der Anzahl von hidden Layers und Neuronen steigt die Fähigkeit eines neuronalen Netzes komplexe Zusammenhänge modellieren zu können. Ist die Netzarchitektur jedoch zu einfach, bedeutet es, dass das Netz die zu lernenden Zusammenhänge und Muster nicht erfassen, somit nicht lernen kann. In einem solchen Fall spricht man von *Underfitting*. Beim Overfitting hingegen ist die Netzarchitektur im Vergleich zur Komplexität des Datensets unbegründet komplex, sodass auch rauschenartige Muster gelernt werden, die aus Sicht der

zu lernenden Zusammenhänge irrelevant sind. Charakteristisch für Overfitting ist, dass die Genauigkeit während des Trainings hoch ist, nicht aber, wenn das Modell auf das Validierungsdatenset angewandt wird. Das Netz hat also das Trainingsdatenset gelernt, ist aber dabei gescheitert, verallgemeinernde Muster zu entdecken. Eine Veranschaulichung von Overfitting und Underfitting ist in Bild 2.6 zu sehen. [Rad17; Alt19; Bok19; MMJ19]

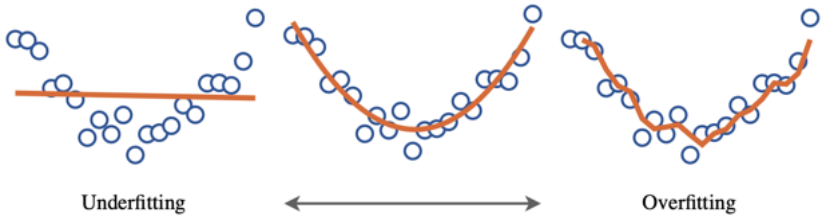


Bild 2.6: Underfitting und Overfitting

Im Folgenden wird auf die einzelnen Hyperparameter eingegangen, unter anderem auch in Bezug auf die Problematik von Over- bzw. Underfitting.

2.2.1 Architektursparameter [Rad17; Alt19; Bok19; MMJ19]

Anzahl von hidden Layers und Neuronen

Bei der Anzahl von hidden Layers und Neuronen gilt es das Gleichgewicht zwischen einer zu einfachen und zu komplexen Architektur zu finden. Hinsichtlich der Trainingszeit, die mit der Komplexität des Netzes steigt, ist es von Vorteil die Architektur möglichst einfach auszulegen, d.h. so wenige Layers und Neuronen zu verwenden wie möglich. Auch die Gefahr von Overfitting ist geringer, wenn die Netzarchitektur einfach gehalten wird. Auf der anderen Seite sollte das Netz komplex genug sein, um Underfitting vermeiden zu können. Insgesamt kann also die Strategie verfolgt werden eine einfache Architektur als Ausgang zu nehmen und diese solange zu erweitern, bis der gewünschte Trainingserfolg erreicht ist. Aus der Anzahl von hidden Layers bzw. Neuronen ergibt sich eine Anzahl von Gewichtsparametern, die während des Trainings optimiert werden. Die Netzarchitektur sollte so ausgelegt werden, dass die Gesamtanzahl dieser Parameter stets kleiner ist als die Anzahl der zur Verfügung stehenden Samples bzw. Datenpunkte im Trainingsdatenset.

Dropout

Dropout ist die Technik während des Trainings vorübergehend, zufällige Neuronen aus dem Netz zu entfernen und somit die Komplexität des Netzes zu reduzieren. Die Reduktion der

Komplexität verringert die Trainingszeiten und die Gefahr für Overfitting. Wie viele Neuronen entfernt werden, bestimmt die Dropout-Rate. welche üblicherweise bei 20-50% liegt. Sie sollte nicht zu niedrig sein, sodass der Dropout eine Wirkung zeigt, sie sollte aber auch nicht zu hoch sein, sonst besteht die Gefahr, dass sich das Netz zu sehr vereinfacht und Underfitting auftritt. Insgesamt hat die Benutzung von Dropout einen rauschbehafteten Trainingsprozess zufolge, der schließlich in einer geringeren gegenseitigen Abhängigkeit der Layers und somit in einer höheren Generalisierungsfähigkeit des Netzes resultiert.

Gewichtinitialisierung

Die Gewichtinitialisierung spielt eine Rolle, da eine passende Wahl für die Startwerte der Gewichte $w_{i,j}^{l}$ das Konvergieren der Fehlerfunktion gegen das globale Minimum deutlich beschleunigen kann. Dadurch kann die Trainingszeit verringert werden. Zusätzlich kann eine gute Initialisierung der Gewichte die Wahrscheinlichkeit erhöhen bei der Optimierung der Fehlerfunktion statt eines lokalen ein globales Minimum zu finden. Die Gewichte werden in der Regel anhand einer Normal- oder Gleichverteilung zufällig initialisiert.

Aktivierungsfunktionen

Die im Abschnitt 2.1.1 bereits thematisierte Aktivierungsfunktion wird hauptächlich aus zwei Gründen benutzt: Ohne sie können sich die Outputs zwischen $-\infty$ und $+\infty$ strecken. Mit einer Aktivierungsfunktion ist es möglich, diesen Wertebereich – wie bspw. bei der Sigmoidfunktion – auf das Intervall von $[0, 1]$ zu beschränken. Zudem können neuronale Netze ohne eine nichtlineare Aktivierungsfunktion nur lineare Zusammenhänge erlernen. Reale Zusammenhänge sind jedoch nur selten linear, sodass es von Vorteil ist, über die Aktivierungsfunktionen dem Netz auch nichtlineare Eigenschaften zu verleihen. Die gängigsten Aktivierungsfunktionen sind in Bild 2.7 dargestellt. Üblicherweise wird bei Klassifikationsaufgaben – das heißt bei Aufgaben, die eine Wahrscheinlichkeit aus $[0, 1]$ als Output haben – in den hidden Layers die rechentechnisch günstigere ReLU (Rectified Linear Unit), während im letzten Layer die glattere Sigmoidfunktion verwendet wird. [Jai19]

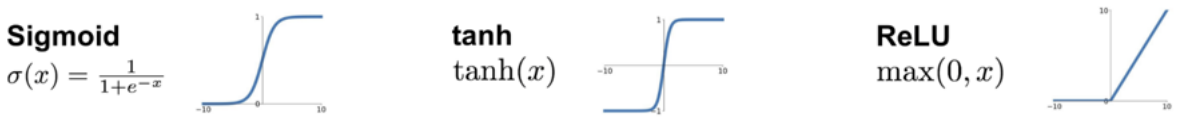


Bild 2.7: Die meistbenutzten Aktivierungsfunktionen [Jai19]

2.2.2 Trainingsparameter [Rad17; Alt19; Bok19; MMJ19]

Lernrate

Die Lernrate bestimmt nach Gleichung (2.32) in welchen Schritten die Gewichte $w_{i,j}^{l}$ nach einer Epoche geändert werden. Bei der Lernrate gilt es ebenfalls ein passendes Gleichgewicht zu finden zwischen zu niedriegen und zu hohen Werten. Eine zu niedrige Lernrate bewirkt, dass die Minimierung der Fehlerfunktion nur sehr langsam erfolgt. Im Fall einer zu hohen Lernrate kann es dazu kommen, dass gar kein Minimum gefunden wird und die Fehlerfunktion divergiert. Wird ein Minimum trotzdem gefunden, passiert dies möglicherweise zwar schneller, jedoch ist es wahrscheilnich, dass statt des globalen Minimums nur ein lokales Minimum gefunden wurde, wie dies in Bild 2.8 zu erkennen ist. Es besteht auch die Möglichkeit keine konstante

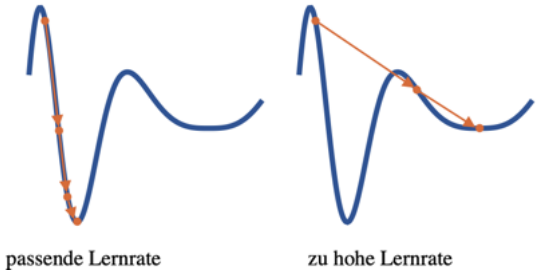


Bild 2.8: Konvergenz bei zu hoher Lernrate

Lernrate zu verwenden, sondern die Lernrate gemäß eines *schedules* sukzessive zu verkleinern. Dies ermöglicht, dass am Anfang des Trainings die Minimierung schneller erfolgt, danach aber keine Divergenz eintritt bzw. das globale Minimum verfehlt wird. Eine mögliche Strategie bei der Lernrate ist es, bei etwa $\alpha = 0.1 \ldots 0.01$ anzufangen und sie iterationsschrittbasiert immer weiter zu verringern. Allerdings können sich die initialen Werte für α je Aufgabe stark variieren.

Momentum

Im Allgemeinen kann bedingt durch Rauschen eine große Varianz in den Gradienten $\frac{\partial E}{\partial w_{i,j}^{l}}\Big|_{s}$ auftreten. Insbesondere bei der Minimierung kann demzufolge eine Oszillation auftreten, das heißt die Iterationswerte nähern zwar das Minimum immer weiter an, schwingen aber darum aus. Das heißt das Finden des Minimums wird zwar nicht verhindert, die Trainingszeit aber erheblich verlängert. Mit Hilfe der Benutzung eines Momentums kann der Varianz in den Gradienten entgegengewirkt werden. Wird ein Momentum bei der Minimierung der Fehlerfunktion verwendet, fließt beim Updaten der Gewichte zum Iterationsschritt s nicht nur die

aktuell errechnete Ableitung $\frac{\partial E}{\partial w_{i,j}^l}\Big|_s$ ein, sondern die vom vorherigen Iterationsschritt ebenfalls:

$$w_{i,j}^l\big|_{s+1} = w_{i,j}^l\big|_s - \Delta w_{i,j}^l\big|_s \tag{2.35}$$

$$\Delta w_{i,j}^l\big|_s = \alpha \frac{\partial E}{\partial w_{i,j}^l}\Big|_s + \gamma \Delta w_{i,j}^l\big|_{s-1} \tag{2.36}$$

$$\text{mit } \Delta w_{i,j}^l\big|_0 = 0 \tag{2.37}$$

Der Faktor γ regelt den Einfluss des Gradienten aus dem letzten Iterationsschritt. Auf diese Weise wird die Varianz gedämpft. Besonders bei der Minimierung der Fehlerfunktion resultiert daraus eine schnellere Konvergenz gegen das Minimum. Der Wert für γ sollte am Anfang des Trainings niedrig gehalten werden, da bei zu hohen Werten die Gradientenrichtung des vorherigen Iterationsschrittes einen zu großen Einfluss hat. Dadurch besteht bei der Minimierung die Gefahr, dass das globale Minimum verfehlt wird, weil während der Iteration die aktuelle Gradientenrichtung zu wenig Einfluss hat. Im Laufe des Trainings kann γ dann kontinuierlich in kleinen Schritten erhöht werden. Je weiter nämlich das Training vorangeschritten ist, umso näher müsste der Wert der Fehlerfunktion dem gesuchten Minimum sein, sodass die Gefahr immer weiter abnimmt, dass durch einen zu großen Wert für γ das Minimum nicht gefunden wird.

Anzahl von Epochen

In welchem Bereich sich die ideale Anzahl für die Epochen befindet, hängt von der Art der Aufgabe ab. Zu wenig Epochen können dazu führen, dass das Netz das Trainingsdatenset nicht oft genug durchlaufen hat und deswegen noch nicht genug Zusammenhänge erlernen konnte. Zu viele Epochen haben jedoch auf der anderen Seite eine längere Trainingszeit zu Folge, sodass auch in diesem Fall ein Gleichgewicht gefunden werden muss. In der Regel rät sich vorzugehen, indem ein kleiner Startwert genommen wird und die Epochenanzahl solange weiter erhöht wird, bis sich die Genauigkeit am Validierungsdatenset nicht mehr weiter erhöht.

Batchsize

Bei der Wahl der Batchsize muss beachtet werden, dass ein zu kleiner Wert zu langen Trainingszeiten führt, da in einem solchen Fall die Backpropagation für jedes einzelne Sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ durchgeführt werden muss. Ist die Batchsize größer als 1, muss die Bakcpropagation nur einmal pro Batch durchgeführt werden und es können allgemeinere Zusammenhänge erlernt werden. Wird die Batchsize jedoch zu groß gewählt, besteht die Gefahr von Underfitting. Die Gewichte werden dann lediglich anhand von batchweise berechneten $\frac{\partial E}{\partial w_{i,j}^l}$ angepasst, wodurch das Netz

nur zu verallgemeinernde Zusammenhänge lernt.

2.3 Loss-Functions & Metrics

Die zu minimierende Fehlerfunktion wird auch *loss function* genannt. Funktionen, die ebenfalls eine Aussage über die Abweichung zwischen t_j und $y_{\text{train},j}$ treffen, jedoch für die Minimierung nicht verwendet werden bzw. eventuell aus mathematischen Gründen nicht oder nicht gut geeignet sind, werden als *metrics* bezeichnet. Die beiden Begriffe werden im Folgenden näher beschrieben. [MMJ19]

Loss-Functions [MMJ19]

Mithilfe eines neuronalen Netzes können Vorhersagen für entweder Regressions- oder Klassifikationsaufgaben getroffen werden. Der Unterschied liegt darin, ob die zu vorhersagende Größe kontinuierlich oder diskret ist. Bei Regressionsaufgaben soll eine kontinuierliche Größe, wie zum Beispiel der Preis eines Artikels, prognostiziert werden. Bei Klassifikationsaufgaben soll ein Input einer Klasse zugeteilt werden, der Output des Netzes ist in diesem Fall eine diskrete Größe, die eine bestimmte Klasse repräsentiert. Eine Klassifikationsaufgabe stellt bspw. die Bildsegmentierung dar: Jedes Pixel steht für einen Input und soll einer Klasse von zu identifizierenden Objekten zugeordnet werden. Allerdings arbeitet das Netz selbst in der Regel nicht mit binären Werten, weswegen der Output genauer genommen zunächst nur eine Wahrscheinlichkeit ist, mit der der Input zu einer bestimmten Klasse gehört. Das Maximum dieser Wahrscheinlichkeiten bestimmt dann, welche Klasse schließlich vorhergesagt wird. Bezeichnet zum Beispiel $t_j = P_n(c)$ für ein Pixel n aus insgesamt N Pixeln die vorhergesagte Wahrscheinlichkeit dafür, dass es zu Klasse c von insgesamt C Klassen gehört, so bezeichnet

$$y_{\text{pred},n} = \text{argmax}(P_n(c)),\ c \in \{1, 2, \ldots, C\} \tag{2.38}$$

die vorhergesagte Klasse. Die Loss-Function $E(t_j, y_{\text{train},j})$ kann dann auch als $E(y_{\text{pred},n}, y_{\text{true},n})$ geschrieben werden. Dabei bezeichnet $y_{\text{true},n}$ die tatsächliche Klasse von Pixel n. Der Unterschied zwischen $y_{\text{true},n}$ und $y_{\text{train},j}$ liegt darin, dass in $y_{\text{train},j}$ die tatsächliche Klasse nicht als Angabe von c sondern als Wahrscheinlichkeit angegeben wird. Diese beträgt für die tatsächliche Klasse 1, für alle anderen 0. Jedes Ausgangsneuron im Netz repräsentiert in diesem Zusammenhang die Wahrscheinlichkeit für die Zuordnung des Pixels n zu einer Klasse c. Eine Übersicht über die Nomenklatur für die Beispielanwendung der Bildsegmentierung gibt zusätzlich Tabelle 2.3.

Je nach Aufgabenstellung variieren auch die für die Optimierung genutzten Loss-Functions. An dieser Stelle werden nur jene genannt, die für Klassifikationsaufgaben geeignet sind, da andere

Tabelle 2.3: Nomenklatur für die tatsächlichen und vorhergesagten Outputs

tatsächlich	**vorhergesagt**	**Art**
$y_{\text{train},j}$	$t_j = P_n(c)$	kontinuierliche Wahrscheinlichkeit
$y_{\text{true},n}$	$y_{\text{pred},n} = \text{argmax}(P_n(c))$	diskrete Klasse

aus Sicht dieser Arbeit nicht relevant sind. Handelt es sich um eine binäre Klassifikation, das heißt es gibt nur zwei Klassen, so bietet sich bspw. die *binary cross entropy* als Loss-Function an:

$$E = -\frac{1}{N}\sum_{n=1}^{N} y_{\text{true},n} \cdot \log(P_n(1)) + (1 - y_{\text{true},n}) \cdot \log(1 - P_n(0)) \tag{2.39}$$

Die binäre Cross-Entropy wird desto größer, je niedriger die Wahrscheinlichkeiten für die Zuordnung eines Samples zu seiner tatsächlichen Klasse ausfallen. Damit aus niedrigen Wahrscheinlichkeiten ein großer Loss-Wert resultiert, wird auf die Wahrscheinlichkeiten der Logarithmus angewandt. Da für die Berechnung die Wahrscheinlichkeiten für eine falsche Zuordnung nicht von Interesse sind, ist die binäre Cross-Entropy so definiert, dass in der Summierung je nach Klasse entweder der erste oder der zweite Term zu Null wird. Es bleibt stets nur der Term übrig, der die Wahrscheinlichkeit für die richtige Zuordnung enthält, der Term mit der Wahrscheinlichkeit für die falsche Zuordnung entfällt. In der Definition werden dazu die Logarithmen mit dem Klassenlabel multipliziert. Zum Schluss wird über alle Samples der Mittelwert gebildet.
Im Falle, dass es mehr als zwei Klassen gibt, kann Gleichung (2.39) zur *categorical cross entropy* verallgemeinert werden:

$$E = -\frac{1}{N}\sum_{n=1}^{N} \log(P_n(y_{\text{true},n})) \tag{2.40}$$

Metrics [Min19]

Die Wahl der Metrics hängt – genauso wie im Falle der Loss-Function – von der Art der Aufgabenstellung ab. Im Folgenden werden einige Metrics vorgestellt, die es erlauben, die Leistung von neuronalen Netzen zur Klassifikation zu evaluieren.
Eine der grundlegendsten Metrics ist die Genauigkeit. Sie ist definiert durch die Anzahl von richtigen Vorhersagen, geteilt durch die Gesamtanzahl von Vorhersagen. Besteht jedoch ein Ungleichgewicht bezüglich der Anzahl von Samples, die einer Klasse gehören, kann diese Metric irreführend sein. Gibt es bspw. 100 Samples im Datenset, von denen 99 der Klasse 1 angehören und ein Sample, das zu Klasse 2 gehört, ergibt sich bereits eine Genauigkeit von 99%, auch wenn alle Samples Klasse 1 zugeordnet werden, das heißt von den Samples von

Klasse 2, keines richtig zugeordnet wird.
Aus diesem Grund wurden Metrics entwickelt, die auch bei einem Klassenungleichgewicht eine präzise Aussage über die Leistung des Netzes treffen können. Für das Verständnis dieser müssen zunächst vier Größen definiert werden, die in der binären Klassfikation gängig sind. Bezeichnet man die beiden Klassen mit positiv und negativ, so werden richtig klassifizierte Samples als *true positives* (TP) bzw. *true negatives* (TN) bezeichnet. Gehört ein Sample in der Wirklichkeit zu den Positiven, wird aber als negativ klassifiziert, spricht man von einem *false negative* (FP). Im umgekehrten Fall, wenn ein Sample in der Wirklichkeit zu den Negativen gehört, aber als positiv klassifiziert wird, handelt es sich um ein *false positive* (FP). Darauf basierend können zunächst vier Raten – die TP-, TN-, FP- und FN-Rate (TPR, TNR, FPR, FNR) – definiert werden, die üblicherweise für weitere Definitionen von Metrics benutzt werden.

$$TPR = \frac{TP}{TP + FN} = 1 - FNR \tag{2.41}$$

$$TNR = \frac{TN}{TN + FP} = 1 - FPR \tag{2.42}$$

$$FPR = \frac{FP}{TN + FP} = 1 - TNR \tag{2.43}$$

$$FNR = \frac{FN}{TP + FN} = 1 - TPR \tag{2.44}$$

Im Folgenden werden Metrics für die binäre Klassifikation vorgestellt, diese können größtenteils jedoch für multiclass Klassifikation angepasst werden. Die Metrics *precision* und *recall* – oder auch *sensitivity* genannt – können folgender Weise definiert werden:

$$precision = \frac{TP}{TP + FP} \tag{2.45}$$

$$recall = TPR = \frac{TP}{TP + FN} \tag{2.46}$$

Die $precision$ setzt demnach richtig klassifizierte Samples einer Klasse in Verhältnis zur Gesamtanzahl von Samples, die tatsächlich dieser Klasse gehören. Der $recall$ hingegen setzt sie in Verhältnis zur Gesamtanzahl von Samples, die dieser Klasse – richtig oder auch falsch – zugeordnet wurden.
Das Pendant zu $recall$ bzw. $sensitvity$ wird als *specificity* bezeichnet und ist wie folgt definiert:

$$specificity = TNR = \frac{TN}{TN + FP} \tag{2.47}$$

Bei binärer Klassifikation kann alternativ zu Gleichung (2.38) die vorhergesagte Klasse $y_{\mathrm{pred},n}$

mittels einer Wahrscheinlichkeitsschwelle θ definiert werden:

$$y_{\text{pred},n} = \begin{cases} 1, & P_n(1) \geqslant \theta \\ 0, & \text{sonst} \end{cases} \tag{2.48}$$

Abhängig davon, wie θ gewählt wird, verändern sich die TPR bzw. die FPR. Wird $\theta = 0$ gewählt, wird alles Klasse 1 zugeordnet und es gilt $TPR = 1$ und $FPR = 1$. Wählt man $\theta = 1$, wird hingegen alles Klasse 0 zugeordnet[1] und es gilt $TPR = 0$ und $FPR = 0$. Für $0 < \theta < 1$ verlaufen die Werte für TPR und FPR ebenfalls zwischen 0 und 1. Plottet man diesen Verlauf, erhält man die sog. *ROC-Kurve* (receiver operating characteristic). Bild 2.9 stellt beispielhaft eine ROC-Kurve dar. Mit Hilfe der Größen *area under the curve*, AUC und

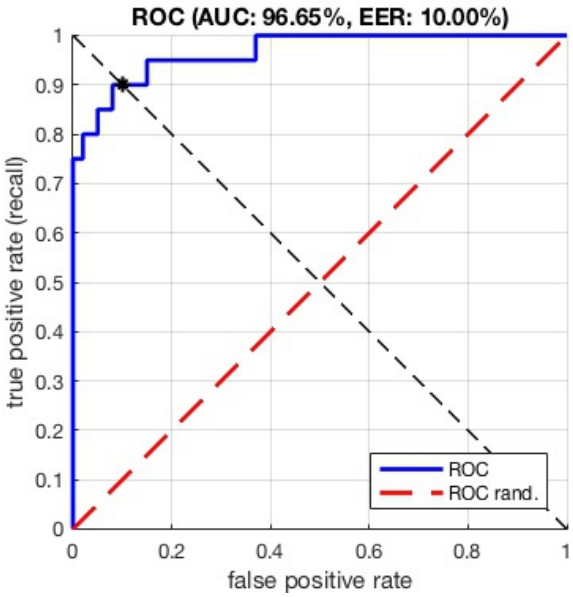


Bild 2.9: Beispielhafte ROC-Kurve [VLF]

equal error rate, EER kann der Verlauf der ROC-Kurve charakterisiert werden. Die AUC ist ein Maß für die Abweichung zwischen ROC-Verlauf und dem idealen Verlauf sowie dem Verlauf der ROC-Kurve, der bei einer zufälligen Klassifikation auftreten würde. Im Idealfall ist $AUC = 1$, da unabhängig von θ in jedem Fall $TPR = 1$ und $FPR = 0$ gilt. Werden die Klassen zufällig zugeordnet, beträgt die AUC 0,5, da in diesem Fall stets $TPR = FPR$ gilt, vgl. „ROC rand." in Bild 2.9.

[1] Dies gilt nur, wenn $P_n(1) < 1$ für alle Samples. Theoretisch ist $P_n = 1$ möglich, jedoch kommt dies in der Praxis nur selten vor, sodass $P_n(1) < 1$ fast immer zutrifft.

Die EER kennzeichnet genau das TPR- FPR Wertepaar, das sich für einen optimalen Schwellwert θ ergibt, in dem TPR möglichst groß ist, FPR jedoch möglichst niedrig bleibt. Die EER ist genau genommen die FPR für diesen optimalen Fall. Die EER lässt sich durch den Schnittpunkt der ROC-Kurve und der Kurve $1 - FPR$ ermitteln, sie ist die zugehörige FPR-Stelle im Graphen.
Zwei weitere Metrics, die in der Bildsegmentierung Anwendung finden, sind die Intersection-over-Union IoU und der Dice-Koeffizient Dc. Sie können ebenfalls mit Hilfe der obigen Definitionen ausgedrückt werden.

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.49}$$

$$Dc = \frac{2TP}{2TP + FP + FN} \tag{2.50}$$

Diese beiden Maße können als eine Komposition der Metrics $precision$ und $recall$ aufgefasst werden. Bei der $precision$ wird die Anzahl der richtig klassifizierten Samples auf die Gesamtanzahl der Samples dieser Klasse bezogen, beim $recall$ auf die Gesamtanzahl der dieser Klasse zugeordneten Samples. Beim Dice-Koeffizient Dc wird die Anzahl der richtig klassifizierten Samples auf die Summe beider Gesamtanzahlen bezogen. Um das so entstandene Maß auf den Wertebereich $[0, 1]$ zu mappen, wird der Zähler mit dem Faktor 2 multipliziert. Alternativ dazu kann auch aus dem Nenner TP subtrahiert werden, auf diese Weise kommt die Definition des Dice-Koeffizienten Dc zustande.

2.4 Neuronale Netze in der Bildsegmentierung

Unter Segmentierung eines Bildes, sei es zwei- oder dreidimensional, versteht man die Zuordnung von Pixeln bzw. Voxeln entweder dem Vordergrund oder dem Hintergrund. [Jäh12] Der Vordergrund verkörpert dabei das zu identifizierende Objekt oder die zu identifizierenden Objekte im Bild. Mit anderen Worten werden die Pixel bzw. Voxel *klassifiziert*. Gibt es nur eine Klasse von Objekten, die zu erkennen ist, spricht man von *binärer Klassifikation*, gibt es mehrere Klassen, spricht man von *multiclass Klassifikation*.
Neuronale Netze, die mit der mathematischen Operation *Faltung* arbeiten, die sogenannten *Convolutional Neural Networks* (*CNN*) – sind ein leistungsfähiges Tool zur Bildverarbeitung. Im Folgenden werden zunächst CNNs im Allgemeinen sowie zwei aufeinander aufbauende Konzepte vorgestellt, die in der Bildsegmentierung gängig sind.

2.4.1 Convolutional Neural Networks [DS13; Wan+20]

Grundlage für CNNs ist die Faltungsoperation. Wie in Bild 2.10 dargestellt, wird bei der Faltungsoperation der Faltungsoperator $f(x, y)$ – der sog. Kernel – über jedes Pixel gelegt und die Pixel, die darunter liegen mit dem Operator verrechnet. Für ein Bild $g(x, y)$ und

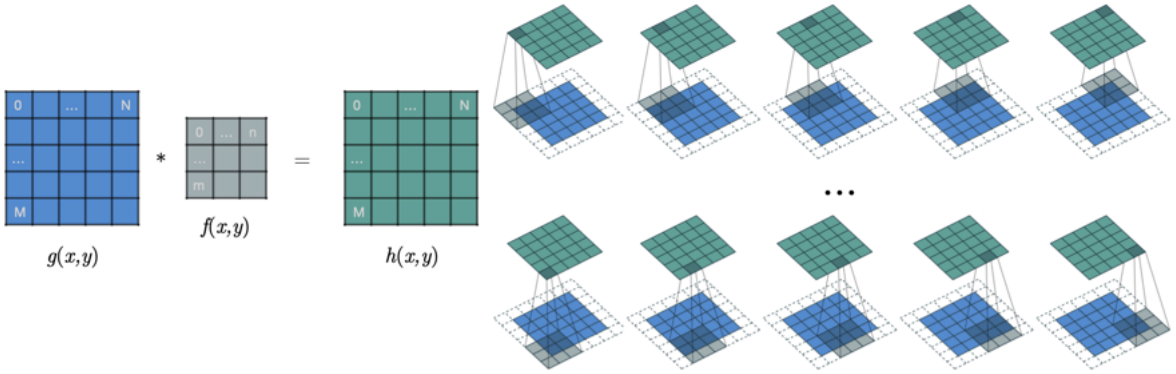


Bild 2.10: 2D-Faltung [Aru18]

einen Faltungsoperator $f(x, y)$ kann das Ergebnisbild $h(x, y)$ mathematisch wie folgt bestimmt werden:

$$h(i,j) = (g * f)(i,j) = \sum_{\substack{x=0 \\ i-n \leqslant x \\ x \leqslant i}}^{N-1} \sum_{\substack{y=0 \\ j-m \leqslant x \\ x \leqslant j}}^{M-1} g(x,y) f(i-x, j-y) \tag{2.51}$$

Soll $h(x, y)$ die genau gleichen Dimensionen wie $g(x, y)$ haben, so muss $g(x, y)$ mit zusätzlichen Pixeln ergänzt werden, diese sind in Bild 2.10 gestrichelt dargestellt und werden *padding* genannt. Das Padding kann bspw. aus Nullpixeln bestehen, das Bild spiegelverkehrt fortsetzen oder periodisch den gegenüberliegenden Rand fortsetzen. Die Ausgangsbildgröße hängt zusätzlich vom sogenannten *stride* ab. Diese Größe beschreibt die Iteartionsschrittgröße der Laufvariablen x und y in Gleichung (2.51). In Bild 2.10 beträgt der Stride 1. Insgesamt gilt mit P Zeilen Padding, einem Stride von S und einer symmetrischen Kernelgröße von $K = m = n$ sowie mit der Eingangsbildhöhe bzw. -breite W für die Ausgangsbildhöhe bzw. -breite O folgender Zusammenhang:

$$O = \frac{W - K + 2P}{S} + 1 \tag{2.52}$$

Fasst man die Pixel sowohl im Bild, als auch im Kernel als Vektoren $\boldsymbol{g} = (g_{0,0}, g_{1,0}, \ldots, g_{N,M})$ bzw. $\boldsymbol{f} = (f_{0,0}, f_{1,0}, \ldots, f_{n,m})$ auf, kann die Faltungsoperation in der in Abschnitt 2.1 vorgestellten Betrachtung für neuronale Netze dargestellt werden. Bild 2.11 veranschaulicht am Eintrag $h_{0,0}$ eines Faltungsbeispiels das resultierende Faltungslayer. Solche Faltungslayer sind

die Grundbausteine von CNNs. Besteht ein neuronales Netz nur aus Faltungslayern, wird dieses auch *fully convolutional network* , kurz *FCN* genannt.

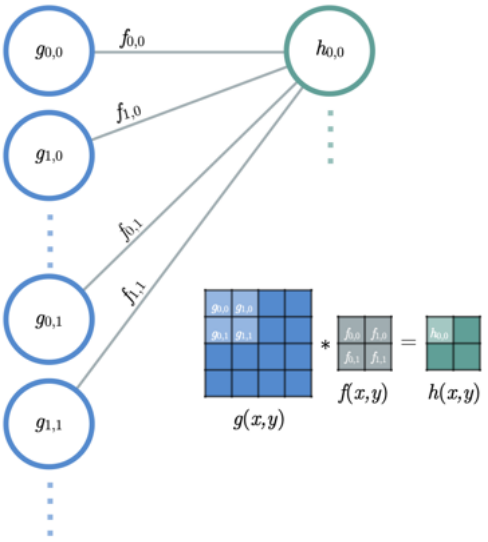


Bild 2.11: Faltungslayer ($S = 2$, $P = 0$)

2.4.2 Fully Convolutional Networks

Long et al. stellen in [LSD15] das Konzept für FCNs vor, vgl. Bild 2.12. FCNs unterscheiden

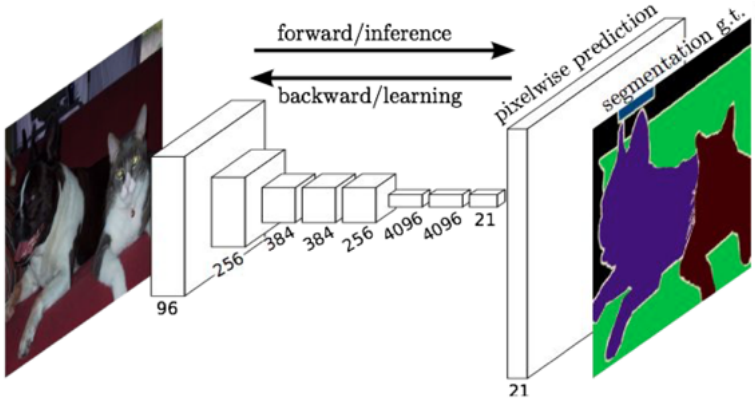


Bild 2.12: Fully Convolutional Network von [LSD15]

sich von anderen CNNs darin, dass im neuronalen Netz ausschließlich Faltungslayer verwendet

werden. Die Verwendung von fully connected Layers, die eine feste Bildgröße verlangen, entfällt also. Das ermöglicht eine von der Bildgröße unabhängige Funktion von FCNs. Die Bildgröße kann somit frei gewählt werden.
Grundsätzlich bestehen FCNs aus einem Convolution-Teil und einem Deconvolution-Teil – auch Downsampling bzw. Upsampling genannt. Im ersteren Teil nimmt die Bildgröße immer weiter ab, und die Tiefe, d.h. die Anzahl von Channels nimmt kontinuierlich zu. Der Aspekt der Tiefe wird noch weiter unten erläutert. Der erste Teil führt dazu, dass die Merkmale im Bild am Ende des Schrittes in einer sehr komprimierten Form vorliegen. Zum Downsampling trägt nicht nur die Faltungsoperation bei: im FCN finden üblicherweise auch Poolinglayer Anwendung. Diese dienen ebenfalls zur Reduktion bzw. zur Verdichtung der Merkmalsinformationen. Bild 2.13 stellt das Konzept von *MaxPooling* dar. Beim MaxPooling wird der

Bild 2.13: MaxPooling ($S = 2$, $P = 0$)

MaxPool-Operator – ähnlich zur Faltung – entsprechend dem festgelegten Stride über das Bild bewegt. Der MaxPool-Operator gibt das Maximum der betrachteten Region zurück.
Der Deconvolution-Teil ist dafür zuständig, die komprimierten Merkmalsinformationen durch Upsampling auf die originale Bildgröße zu projizieren. Dadurch wird eine pixelweise Bildsegmentierung möglich. Das Upsampling erfolgt mit Hilfe der Umkehrung der Faltungsoperation, die Operation wird Dekonvolution, transponierte Faltung (*transposed convolution*) oder *upconvolution* genannt. Bild 2.14 verdeutlicht das Prinzip der umgekehrten Faltungsoperation. Genauso wie bei der gewöhnlichen Faltung, ergibt die Wahl der Parameter Stride und Padding die Ausgangsbildgröße. Der Zusammenhang ist Gleichung (**??**) zu entnehmen, welche die Umkehrung von Gleichung (2.53) ist. Die Parameter Stride und Padding haben dadurch auch eine umgekehrte Wirkung. Analog zu Bild 2.11 ist das Konzept der umgekehrten Faltung auch konsistent mit dem Framework für neuronale Netze.

$$O = S(W - 1) + K - 2P \tag{2.53}$$

Mit der Gesamtzahl von Faltungen, die in einem convolutional Layer an einem Bild ausgeführt werden, kann die Anzahl ausgegebener Channels beeinflusst werden. Darüber wie die Anzahl der Channels mit anderen Faltungsparametern zusammenhängt, gibt Bild 2.15 Auskunft. Bei einem Eingansgbild mit nur einem Channel – wie etwa in Bild 2.10 sowie in Bild 2.14 – wird

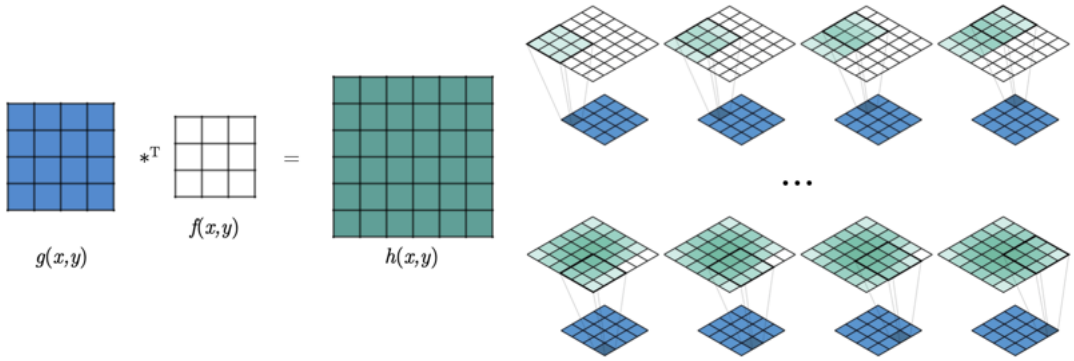


Bild 2.14: Umgekehrte 2D-Faltung [Lan18]

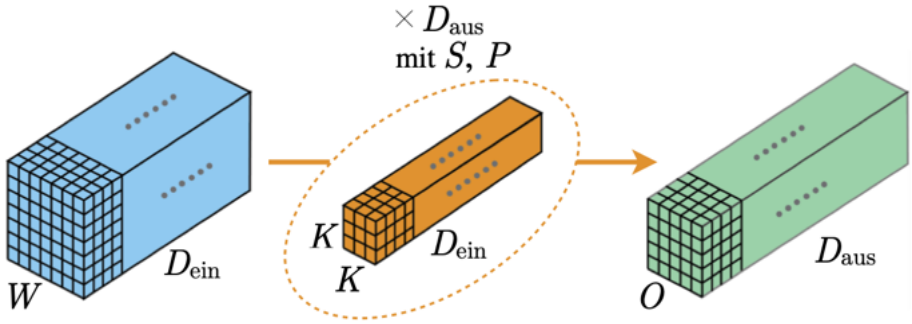


Bild 2.15: Multichannel Convolution [Bai19]

die Faltung mit nur einem Kernel ausgeführt. Liegen im Gegensatz dazu mehrere Channels beim Eingangsbild vor, so werden bei D_{ein} Channels D_{ein} Kernels verwendet. Die Kernels können in einem 3D-Kernel – auch genannt Filter, in Bild 2.15 orange dargestellt – zusammengefasst werden. Pro Filter resultiert ein Ausgangschannel, sodass bei D_{aus} gewünschten Ausgangschannels D_{aus} Filter verwendet werden müssen. Die Verhältnisse zwischen Ein- und Ausgangsbildhöhe bzw. -breite lassen sich durch die Wahl von Kernelgröße, Stride und Padding gemäß Gleichung (**??**) bei der Faltung und gemäß Gleichung (2.53) bei der umgekehrten Faltung bestimmen. Insgesamt definieren also die Größen D_{aus}, K, S und P bei einer Faltung oder einer umgekehrten Faltung die Dimensionen des Ausgangs vollständig.

2.4.3 U-Net

Das sog. U-Net wurde spezifisch für die medizinische Bildverarbeitung entwickelt [RFB15] und basiert ebenfalls auf den Schritten Down- und Upsampling. Zunächst nimmt beim Downsamp-

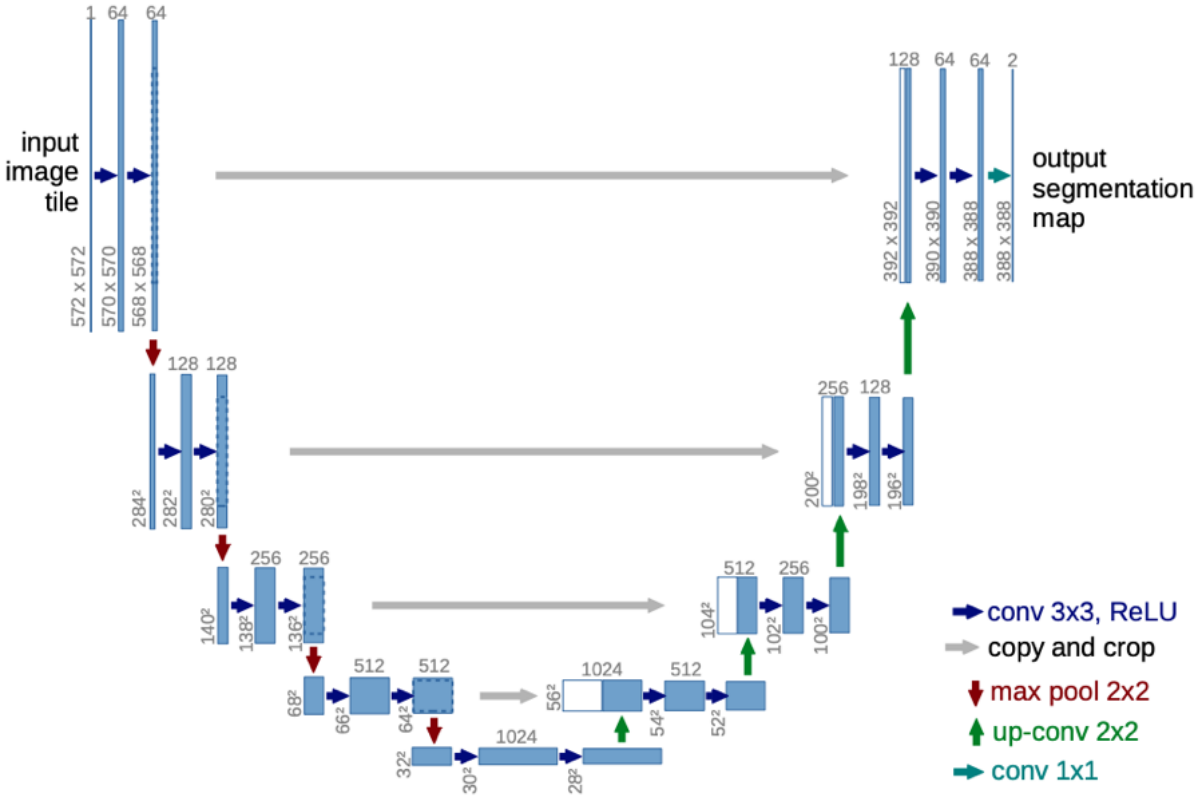


Bild 2.16: Architektur des U-Nets [RFB15]

ling die Bildgröße ab und die Tiefe zu, danach nimmt beim Upsampling die Tiefe wieder ab und die Bildgröße zu bis die gewünschten Ausgangsdimensionen erreicht sind. Die Ausgangstiefe am Ende des Upsamplings korrespondiert mit der Anzahl von Objektklassen, die im Bild zu segmentieren bzw. zu erkennen sind. Mit Hinblick auf den Verlauf der Informationsverdichtung können die Down- und Upsampling-Teile auch *Encoder* und *Decoder* genannt werden. Denn abstrakt gesehen wird im ersten Schritt erkannt, *was* für Objekte sich im Bild befinden und im zweiten wird konkretisiert *wo* sie sich befinden.
Das U-Net besteht ausschließlich aus Faltungslayern, es ist also ein fully convolutional Network. Wie es Bild 2.16 zu entnehmen ist, zeichnet sich das U-Net durch die sog. *skip connections* aus (*copy and crop*), welche durch die grauen Pfeile dargestellt sind. Diese Kurzschlüsse sollen beim Upsampling helfen die im Downsampling gefundenen Objekte genauer zu lokalisieren. Dazu werden die Outputs nach jeder Faltung im Downsamplingzweig als zusätzliche Channels zu den Outputs nach jeder umgekehrten Faltung im Upsamplingzweig dazugenommen.

3 Stand der Technik

Im Folgenden wird zunächst die Vorarbeit vorgestellt, im Rahmen deren die Entwicklung des vollständigen Segmentierungsalgorithmus stattfindet. Das zu entwickelnde neuronale Netz soll später in diesen Algorithmus integriert werden. Im Anschluss werden die Arbeiten vorgestellt, an denen sich die Ansätze dieser Arbeit orientiert haben. Behandelt wird dabei die synthetische Datenherstellung sowie die Netzarchitektur.

3.1 Wurm – Deformierbare Organismen

Um den vollständigen Segmentierungsalgorithmus zu verstehen, wird in diesem Abschnitt zu Anfang der Datensatz beschrieben, aus dem die Gefäße zu segmentieren sind. Es wird dabei zuerst auf den rohen Datensatz und darauffolgend auf die Vorverarbeitung dessen eingegangen. Danach wird das Konzept des Algorithmus vorgestellt und es wird erläutert, wie diese Arbeit an die Entwicklung des Gesamtalgorithmus anknüpft.

3.1.1 Daten

Zur Datenerfassung wurden an der HTTG der MHH histologische Schnittbilder von einem Schweineherzen mit einem Lichtmikroskop bei 100-facher Vergrößerung eingescannt. Einer der Schnitte aus dem mittleren Teil des Herzens ist in Bild 3.1 dargestellt. Das Herz hat etwa die Maße 39 mm$\times$33 mm$\times$33 mm, der größte Querschnitt in der z-Ebene beträgt ca. 11 cm^2. Die kleinsten Gefäße, die Kapillare, haben etwa einen Durchmesser von 2 µm bis 3 µm. Der Durchmesser der größten zu erfassenden Gefäße liegt zwischen 1 mm und 1,5 mm. Das bedeutet, zwischen den Maßen des kleinsten und des größten Gefäßes liegt ein Faktor von etwa 500. In den Schnittbildern sind pro mm^2 etwa 1000 Gefäße zu finden.
Insgesamt liegen 2362 Schnitte vor, sodass die Auflösung entlang der z-Achse etwa 15 µm beträgt. Die Auflösung des Mikroskops in der x-y-Ebene liegt bei der verwendeten Vergrößerung bei 0,645 µm. Eine derartig hohe Auflösung hat eine erhebliche Datenmenge zufolge, was in Bild 3.2 veranschaulicht wird. Die rohen Daten, welche Aufnahmen in der x-y-Ebene sind, wurden zunächst binarisiert. Wie in Bild 3.3 zu erkennen ist, erfolgte die Binarisierung basierend auf den Intensitätsunterschieden zwischen dem Volumen des Herzens und dem Hintergrund bzw. dem Lumen der Gefäße, sowie der Herzkammer und Vorhöfe mittels Schwellwertverfahren. Die Binärbilder wurden anschließend paarweise rigide registriert, sodass sie zu einem 3D-Volumen

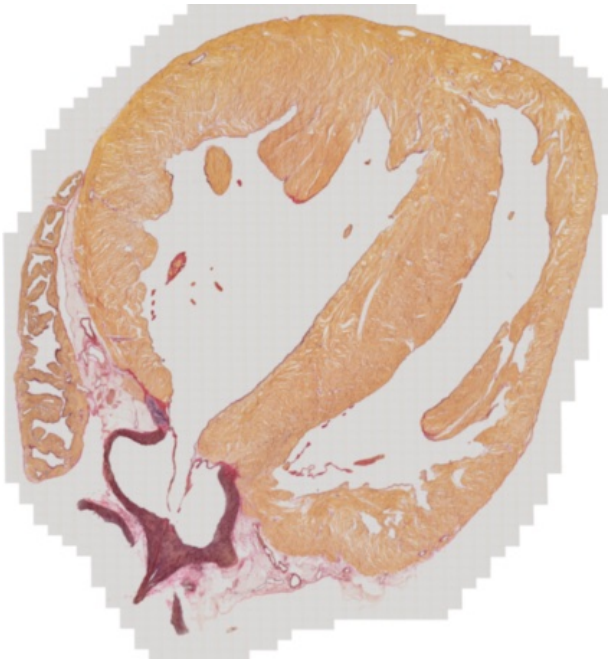


Bild 3.1: Beispiel für die rohen Daten aus dem Mikroskop

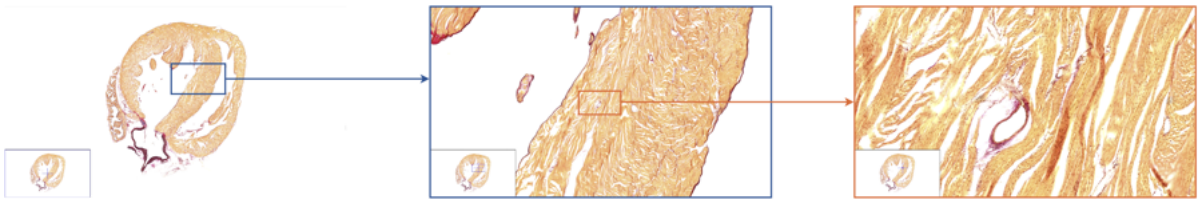


Bild 3.2: Beispielschnitt mit Zoom (mit Histogrammanpassung)

zusammengesetzt werden konnten. Die Registrierung stellt Bild 3.4 an einem Beispielpaar dar. Auf der linken Seite sind die betroffenen Schichten übereinander gelegt abgebildet. An den schwarzen und weißen Stellen stimmen die beiden Bilder überein, an den farbigen nicht. Die beiden Farben stehen jeweils für eine Schicht.

Das zusammengesetzte 3D-Volumen enthält erhebliches Rauschen. Die ursprünglichen Gründe dafür sind in der Herstellung der histologischen Schnitte zu suchen: Manche Schnitte weisen Beschädigungen und Deformitäten auf, die im Schneideprozess entstanden sind. Des Weiteren sind die Kontraste zwischen den Färbungstönen nicht über den gesamten Datensatz konstant. Außerdem ist das Schwellwertverfahren nicht in der Lage zwischen den Lumina der Kammer, der Vorhöfe sowie dem Hintergrund selbst und den gesuchten Lumina der Gefäße zu unter-

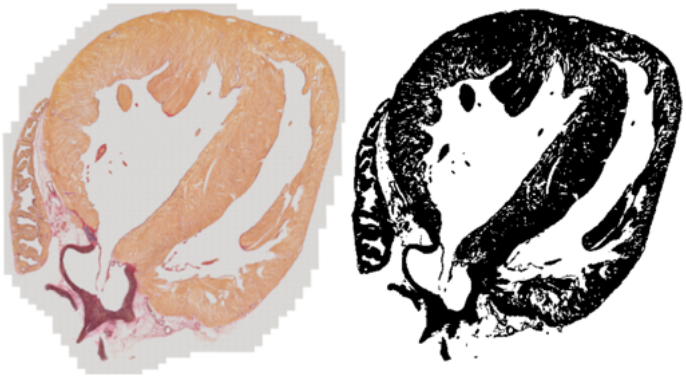


Bild 3.3: Binarisierung mittels Schwellwertverfahren

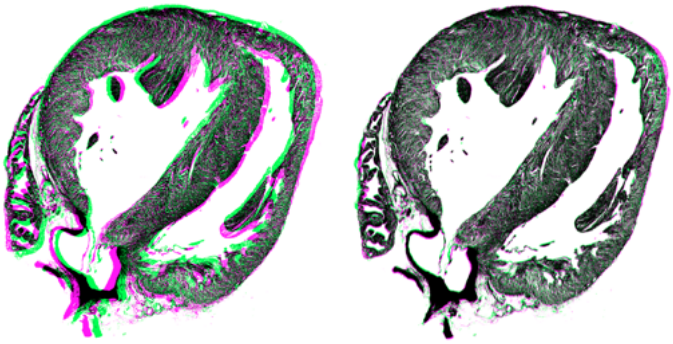


Bild 3.4: Paarweise rigide Registrierung

scheiden. Das heißt, die Binarisierung durch das Schwellwertverfahren kann lediglich als eine Vorstufe zur eigentlichen Segmentierung betrachtet werden. Die *semantische* Segmentierung ist dadurch noch nicht erfolgt.
Nach der Vorsegmentierung durch das Schwellwertverfahren sind die Lumina im Herzen mit dem Hintergrundvolumen verbunden. Um ersteres vom letzteren trennen zu können, wurde mittels der umhüllenden Form des Herzens eine Maske definiert. Der resultierende Datensatz wird in dieser Arbeit zur Segmentierung der Gefäße verwendet. Bild 3.5 stellt diesen Datensatz dar: an den beiden Seiten sind die Gefäße der beiden Koronararterien zu erkennen. Die Herausforderung bei der Segmentierung steckt somit darin, zwischen Voxel der Gefäßlumina und Voxel anderer Lumina sowie Rauschen unterscheiden zu können.

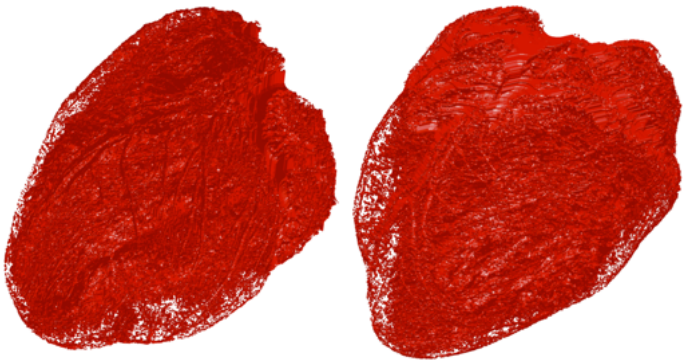


Bild 3.5: Datensatz nach Vorverarbeitung

3.1.2 Konzeptbeschreibung

Das Herz wird über die Koronararterien mit frischem Blut versorgt. Es gibt eine linke (*Arteria coronaria sinistra*) und eine rechte Koronararterie (*Arteria coronaria dextra*), die ihren Ursprung nach der Aortenklappe in der Aorta haben. Der Grundansatz ist es, ausgehend vom jeweiligen Anfang der Korornararterien dem Verlauf der Arterien über die Kapillare und Venen hin bis zurück ins Herz zu folgen. Sollte die Bildauflösung an bestimmten Stellen die Segmentierung der Kapillare nicht mehr erlauben, so kann zusätzlich von den Venen ausgegangen werden bis die gleiche Grenze von der anderen Seite wieder erreicht ist. Der Saatpunkt von der rechten bzw. linken Koranararterie ist jeweils in Bild 3.6 dargestellt.

Den Verlauf soll ein, einem Wurm ähnlicher Algorithmus verfolgen, sodass die Segmentierung nur lokal, in einem in der Größenordnung des aktuellen Gefäßdurchmessers definierten Sichtfeld am Kopf des Wurmes stattfindet. Auf diese Weise lässt sich der Rechenaufwand reduzieren, da nicht das Gesamtvolumen verarbeitet wird, sondern nur lokale Subvolumina betrachtet werden, die auch Gefäße enthalten. Dieses Konzept für die Gefäßsegmentierung wird in [MH06] – basierend auf dem wurmähnlichen Verhalten des Algorithmus – *Deformierbarer Organismus* genannt. Das Vorgehen von McIntosh und Hamarneh muss jedoch für die Anwendung auf den vorliegenden Datensatz wegen der Datengröße vereinfacht werden. Der deformierbare Organismus in [MH06] wird – wie in Bild 3.7 zu sehen – mit einer großen Anzahl von Massenpunkten (schwarz), und tangentialen sowie radialen Federn (blau) sehr detailliert modelliert. Im vorliegenden Fall wird der Organismus in zylindrische Segmente aufgeteilt und pro Segment nur folgende Informationen mitgeführt bzw. abgespeichert: die Koordinaten der Anfangs- bzw. Endknotenpunkte und der Radius. Auf eine detaillierte geometrische und physikalische Modellierung wird verzichtet, dies würde schätzungsweise zu einer nicht mehr behandelbaren Menge von Massenpunkten führen. Ebenfalls wird das Sichtfeld des Wurmes

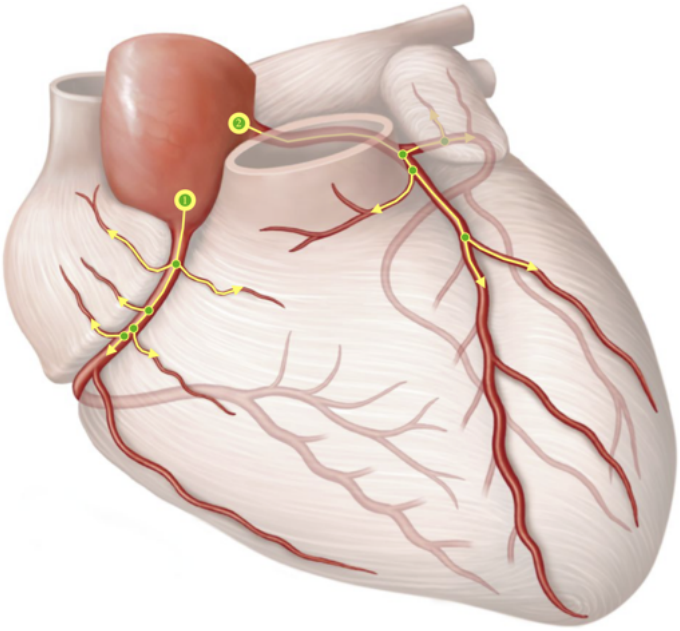


Bild 3.6: Saatpunkte in der rechten (1) und in der linken (2) Koranararterie [SSS18]

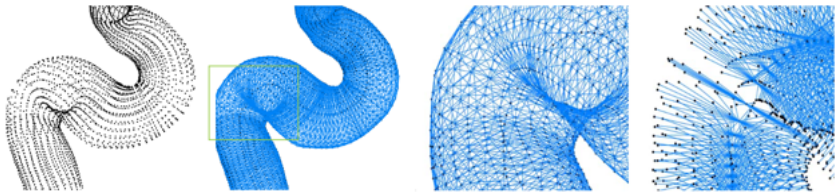


Bild 3.7: Deformierbarer Organismus von McIntosh und Hamarneh [MH06]

statt einer Halbkugel – wie in [MH06] definiert – durch einen rechentechnisch günstigeren Würfel ersetzt. Ein Würfel ermöglicht effiziente Berechnungen basierend auf Matrizen, da so die Voxel des Sichtfelds ohne Transformation in einer 3D- Matrix abgebildet werden können. Das Vorgehen fasst Algorithmus 1 zusammen. Als Input nimmt der Algorithmus neben den Saatpunkten in den Koronararterien eine jeweils zu den Saatpunkten gehörende Startrichtung. Außerdem werden dem Algorithmus die Bilddaten in Form einer Multiskalenbibliothek übergeben. Die Multiskalenbibliothek wird mittels Downsampling, durch siebenmaliges Halbieren der ursprünglichen Kantenlänge der Schnittbilder erstellt. Somit haben die größten Bilder auf der niedrigsten Auflösungsstufe die Dimensionen von etwa $500\,\text{px} \times 500\,\text{px}$. Die niedrigste Auflösungsstufe wird als Stufe H, die größte als A bezeichnet. Zu jeder Auflösungsstufe wird die jeweilige Auflösung in einer Look-Up-Table hinterlegt, diese wird dem Algorithmus ebenfalls

Algorithmus 1: Wurm

Input: Saatpunktkoordinaten $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$
Startrichtungen $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$
Multiskalenbibliothek MSL
Auflösungs-LUT $resolutionLUT$

Output: Gefäßbaumgraph G

```
1  G ←initializeGraph(S, D);
2  growingPoints ← S;
3  foreach resolutionLevel ∈ {'H','G','F','E','D','C','B','A'} do
4      currentVolume ← getVolume(MSL, resolutionLevel);
5      currentResolution ← getResolution(resolutionLUT, resolutionLevel);
6      while growingPoints ≠ ∅ do
7          bifurcationPoints ← ∅;
8          foreach currentGrowingPoint ∈ growingPoints do
9              minimalRadius ← getRadius(currentGrowingPoint);
10             currentTip ← currentGrowingPoint;
11             while minimalRadius ⩾ currentResolution do
12                 currentRadius ← getRadius(currentTip);
13                 fieldOfView ← getFieldOfView(currentTip, currentRadius);
14                 newTip ← getBestGrowthCandidate(fieldOfView);
15                 newRadius ← getRadius(newTip);
16                 minimalRadius ← min(minimalRadius, newRadius);
17                 G ←updateGraph(currentTip, newTip, newRadius);
18                 if isBifurcation(currentTip) then
19                     bifurcationPoints ← {bifurcationPoints, currentTip};
20                 end
21                 currentTip ← newTip;
22             end
23         end
24         growingPoints ← bifurcationPoints;
25     end
26     growingPoints ← getLeafPoints(G);
27     if resolutionLevel ≠'A' then
28         MSL ← registrateFollowingVolumes(MSL, resolutionLevel, G);
29     end
30 end
```

bereitgestellt.
In Bild 3.8 ist der Durchlauf der For-Loop in Zeile 3, die nach dem Input angefangen wird, für die Auflösungsstufe H anhand eines einfachen Beispiels veranschaulicht. Der Wurm fängt in

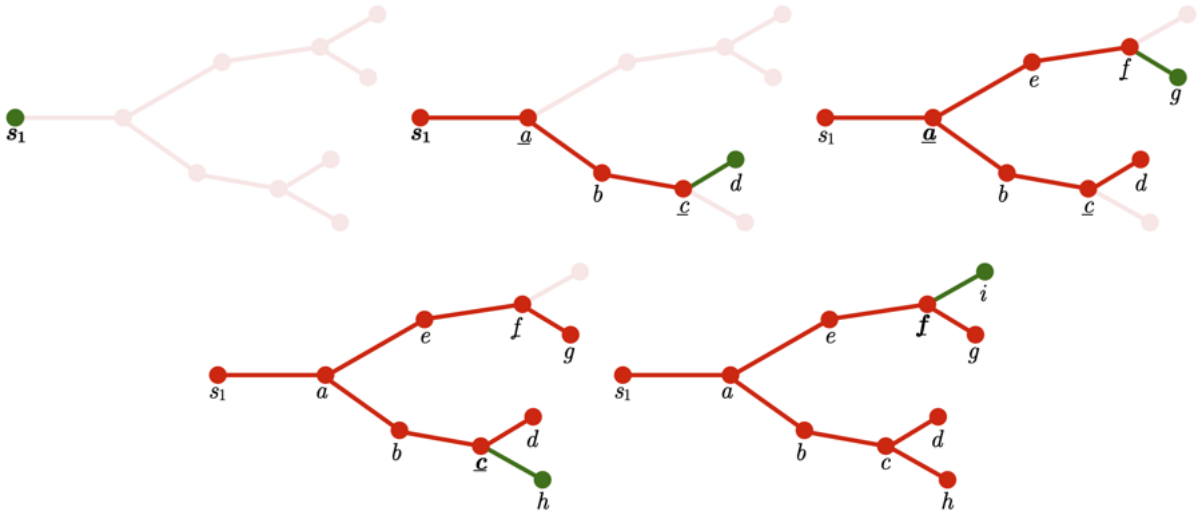


Bild 3.8: Der Segmentierungsalgorithmus an einem einfachen Beispiel

der Auflösungsstufe H an und durchgeht alle Saatpunkte nacheinander. Im Saatpunkt wird das würfelförmige Sichtfeld definiert und mittels verschiedener Kriterien – unter anderem auf der Basis eines Frangi-Filters [Fra+98] – jedem Voxel eine Wahrscheinlichkeit zugeteilt, mit der sie zum aktuell verfolgten Gefäß gehören. Das Voxel mit der höchsten Wahrscheinlichkeit wird als Endknotenpunkt für das nächste Segment ausgewählt und der Graph, der den Gefäßbaum repräsentiert, wird um das Segment erweitert. Dazu wird auch der Radius an der neuen Stelle berechnet. Als nächstes wird geprüft, ob der neue Punkt eine Bifurkation ist. Ist dies der Fall, wird der Punkt abgespeichert. In Bild 3.8 sind die Namen solcher Knotenpunkte unterstrichen. Der Wurm wächst auf diese Weise solange, bis die Auflösung nicht mehr ausreicht, um noch kleinere Gefäße zu identifizieren. Ist diese Grenze erreicht (in Bild 3.8 grün dargestellt), bricht er ab und nimmt die zuvor abgespeicherten Bifurkationen und wächst an den Stellen (Knotenpunktnamen in Bild 3.8 fett gekennzeichnet) genauso weiter. Sind keine Punkte mehr für Wachstum abgespeichert, oder mit anderen Worten, wurden alle Gefäße erkannt, die die aktuelle Auflösungsstufe zulässt, wird zur nächstgrößeren Auflösungsstufe gewechselt und der Prozess setzt sich an den Endknotenpunkten des aktuellen Graphs fort.
Bild 3.4 zufolge scheint die rigide Registrierung die Schichten zufriedenstellend zueinander auszurichten. Allerdings gibt es neben den Dislokationen auf makroskopischer Ebene, die von dem Scannprozess herrühren, auch Dislokationen mikroskopischer Art. Diese sind im Laufe des

Schneideprozesses wegen mechanischer Beanspruchung entstanden: an manchen Stellen sind die Schichten gestaucht, an anderen gedehnt. Das bedeutet, auch wenn die groben Verschiebungen durch den Scannprozess vollständig eliminiert wären, gäbe es Dislokationen zwischen den Schichten. Aus diesem Grund wird die Registrierung von Auflösungsstufe zu Auflösungsstufe verfeinert. In Zeile 28 des Algorithmus findet deshalb eine elastische Registrierung statt, bei der die bereits identifizierten Gefäße als Ankerpunkte genommen werden. Auf diese Weise werden mit steigender Auflösung immer feinere Verbesserungen an der Registrierung vorgenommen und die Transformationsvorschrift der Registrierung konvergiert gegen eine optimale.
Der Vorteil, der sich aus dem im Algorithmus verwendeten Multiskalenansatz ergibt, ist, dass auf dieser Weise die Maße des Sichtfelds minimal gehalten werden. Würde die Segmentierung unmittelbar in der höchsten Auflösungsstufe starten, wäre das Sichtfeld der Größten Gefäße bei einer Kantenlänge des zweifachen aktuellen Durchmessers etwa $3900\,\mathrm{px} \times 3900\,\mathrm{px} \times 170\,\mathrm{px}$ groß. Der Multiskalenansatz ermöglicht dies auf Sichtfelder mit einer Kantenlänge von etwa $30\,\mathrm{px}$ zu verkleinern. Dies entspricht einer Reduzierung der Speicherkapazität für das Sichtfeld, bzw. der Anzahl der Datenpunkte, die durch die Funktion $\mathrm{getBestGrowthCandidate}$ zu verarbeiten sind, um den Faktor von ca. 100000.
Der Gesamtalgorithmus befindet sich noch in der Entwicklung. Für die beiden herausforderndsten Punkte des Algorithmus, die sich hinter den Funktionen $\mathrm{getBestGrowthCandidate}$ und $\mathrm{isBifurcation}$ verbergen, sollen Ansätze mittels neuronaler Netze in Betracht gezogen werden. Das heißt, das zu entwickelnde neuronale Netz hat die Aufgabe, im Sichtfeld des Wurmes Voxel zu klassifizieren. Relevant ist es dabei, ob ein Voxel einem Gefäß bzw. einer Bifurkation angehört.

3.2 Synthetische Datenherstellung

In [Sch+12] stellen Schneider et al. einen komplexen Algorithmus für die Erstellung von 3D-Gefäßbäumen vor. Der Ansatz basiert sowohl auf physiologischen Aspekten, als auch auf geometrischen, morphologischen Randbedingungen. Der Gefäßbaum wächst von Saatsegmenten aus. Diese verzweigen sich und die Gefäße werden immer feiner. In physiologischer Hinsicht wird für das Wachstum die Entstehung von Gefäßen – in der Fachliteratur *Angiogenese* genannt – modelliert. Minderdurchblutete – das heißt *ischämische* – Zellen sezernieren Stoffe, die die Angiogenese anregen. Der Algorithmus lässt die neuen Gefäßsegmente in chemotaktischer[1] Antwort auf diese Sekrete wachsen. In morphometrischer Hinsicht richtet sich das Gefäßwachstum nach Statistiken, die die Bildung von Bifurkationen erfassen. Das Ergebnis des Algorithmus ist somit ein solcher, physiologisch plausibler Gefäßbaum, der zusätzlich den

[1] Chemotaxis bezeichnet ursprünglich die Anlockung von Immunzellen als Antwort auf die Ausschüttung von Botenstoffen.

vorgegebenen morphologischen Forderungen entspricht. Die Plausibilität des resultierenden Gefäßbaumes belegen Schneider et al. durch den Vergleich der Gefäßbaumstruktur mit echten Daten.
Eine Übersicht des gesamten Ablaufs der Gefäßbaumgenerierung ist in Bild 3.9 zu sehen. Das Gefäßbaummodell wird zunächst initialisiert. Bei gegebenem Sauerstoffverbrauch wird

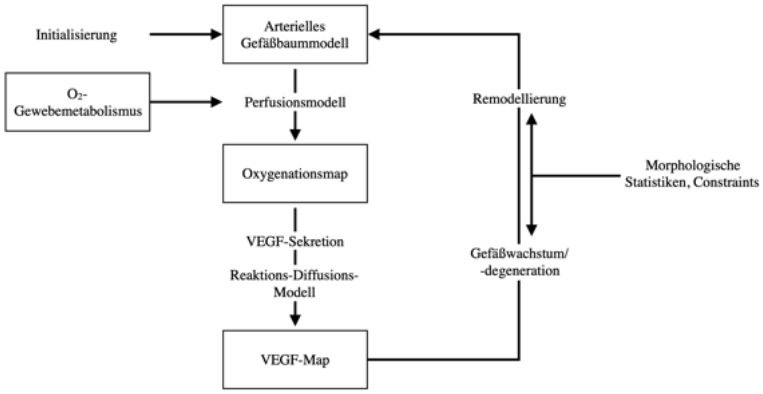


Bild 3.9: Gesamtablauf der Gefäßbaumgenerierung

für das initialisierte Gefäßbaummodell mittels eines Perfusionsmodells[2] die Oxygenation in der Simulationsdomain berechnet. Die Simulationsdomain ist das zu Anfang definierte, zu durchblutende Volumen. Anhand der Oxygenation wird die Ausschüttung des VEGF, des *vascular endothelial growth factors*, modelliert und in einer Map hinterlegt. Anschließend folgt die Bestimmung der Verteilung des VEGF in der Simulationsdomain, was über ein Diffusionsmodell erfolgt. Ist diese bekannt, können die Stelle und die Richtung des Baumwachstums bestimmt werden. Da die Angiogenese ein äußerst dynamischer Prozess ist, entstehen im Laufe deren nicht nur Gefäße, sondern Gefäße werden auch zurückgebaut, wenn sie sich wegen der Entstehung günstigerer Durchblutungswege als uneffektiv oder sogar überflüssig erweisen. Aus diesem Grund werden am Gefäßbaum Segmente im Rahmen eines Degenerationsschrittes auch entfernt. Durch die neu wachsenden bzw. entfernten Gefäßsegmente ändert sich der Status quo hinsichtlich der morphometrischer Statistiken, sodass eine Remodellierung erwünscht ist. Nach dem Wachstum, Degeneration und Remodellierung wird das Gefäßbaummodell aktualisiert und mit der Berechnung der Oxygenation setzt sich der Wachstumsprozess fort bis ein gewünschter Perfusionsgrad der Simulationsdomain erreicht ist.
Aus Bild 3.9 können die Komponenten O_2-Gewebemetabolismus, Perfusionsmodell, Oxygenationsmap, VEGF-Sekretion, Reaktions-Diffusions-Modell und VEGF-Map als Angiongenesemodell zusammengefasst werden. Dieses Modell nimmt als Input das Gefäßbaummodell

[2]Perfusion bezeichnet die Durchblutung eines Gewebes oder eines Organs.

und gibt als Output die VEGF-Map für das Wachstum bzw. für die Degeneration zurück. Die VEGF-Map beschreibt die VEGF-Konzentration in jedem Voxel der Simulationsdomain und drückt damit aus, in welchem Maß ein Voxel ischämisch ist. Damit wird im nächsten Schritt beeinflusst, an welcher Stelle der Baum das Wachstum fortsetzt. Auf eine detailliertere Ausführung des Angiogenesemodells wird an dieser Stelle verzichtet, da dies hinsichtlich dieser Arbeit nicht relevant ist. Im Folgenden werden jedoch die Grundzüge der restlichen Komponenten in Bild 3.9 näher vorgestellt.

3.2.1 Das Gefäßbaummodell

Schneider et al. beschränken ihre Arbeit lediglich auf den arteriellen Teil des Gefäßsystems. Nach [Zam76] treten arterielle Furkationen, die in mehr als zwei Zweigen enden, nur äußerst selten auf, sodass arterielle Gefäßbäume in der Regel als binäre Bäume aufgefasst werden können. Schneider et al. merken jedoch an, dass im Gegensatz zu den Arterien für die verwobene Struktur des Kapillarnetzes die Annahme von binären Bäumen eine starke Vereinfachung darstellt.
Binäre Bäume können effektiv als gerichtete Graphen, bestehend aus Knotenpunkten und Kanten, abgespeichert werden. Eine Kante dieses Graphs stellt dabei ein zylinderartiges Gefäßstück mit Länge l und Radius r dar. Der Radius kann als Kantengewicht aufgefasst werden. Die Länge ergibt sich aus den Koordinaten, mit denen die Knotenpunkte zusätzlich versehen werden. Aus der Graphrepräsentation ergeben sich die in Bild 3.10 veranschaulichten Begriffe. Die Anfangs-

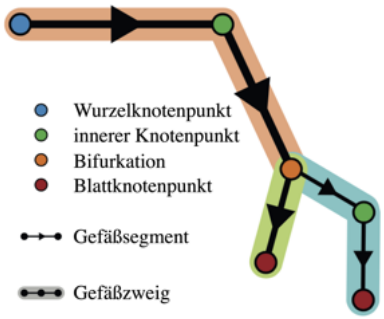


Bild 3.10: Terminologie für das Gefäßbaummodell

knotenpunkte des initialen Segments heißen Wurzelknotenpunkte, die Knotenpunkte mit einer Verzweigung Bifurkationen und die Knotenpunkte ohne Ausgangskanten Blattknotenpunkte. Alle anderen Knotenpunkte sind innere Knotenpunkte. Eine Kante wird als Gefäßsegment bezeichnet und eine Folge von Kanten, die keine Bifurkationsknotenpunkte enthält, wird Ge-

fäßzweig genannt. Das *proximale* Ende eines Zweiges oder Segmentes bezeichnet das Ende, das sich in Richtung Wurzelknotenpunkt befindet, das *distale* hingegen jenes, das in Richtung der Blattknotenpunkte liegt.
Der Radius eines Zweiges ist konstant, er ändert sich lediglich im Falle einer Bifurkation. Die Gesetzmäßigkeit, nach der sich die Abnahme der Radien richtet, wird von *Murrays Gesetz* [Mur26] beschrieben. Es definiert das Verhältnis zwischen dem Radius des proximalen Segments r_p und den Radien der linken und rechten distalen Segmente r_l bzw. r_r mit Hilfe des Bifurkationsexponenten $\gamma \in [2{,}0; 3{,}0]$.

$$r_{\mathrm{p}}^{\gamma} = r_{\mathrm{l}}^{\gamma} + r_{\mathrm{r}}^{\gamma} \tag{3.1}$$

Die Universalität von Murrays Gesetz ist jedoch nicht bewiesen: Dass es neben der makroskopischer Ebene auch für die mikroskopische Ebene gilt, ist fraglich. Den Zusammenhang zwischen den Bifurkationswinkeln ϕ_l und ϕ_r und den Radien beschreibt Gleichung (3.2) bzw. Gleichung (3.3).

$$\cos(\phi_{\mathrm{l}}) = \frac{r_{\mathrm{p}}^4 + r_{\mathrm{l}}^4 - r_{\mathrm{r}}^4}{2r_{\mathrm{p}}^2 r_{\mathrm{l}}^2} \tag{3.2}$$

$$\cos(\phi_{\mathrm{r}}) = \frac{r_{\mathrm{p}}^4 + r_{\mathrm{r}}^4 - r_{\mathrm{l}}^4}{2r_{\mathrm{p}}^2 r_{\mathrm{r}}^2} \tag{3.3}$$

Nach [Mur26] entspricht das Gesetz – folgend den Regeln der Thermodynamik – dem Prinzip der minimalen Energie.

3.2.2 Gefäßwachstum

Die Stelle des aktuellen Gefäßwachstums wird aus der VEGF-Map bestimmt. Dabei werden zunächst N_{p} Punkte aus der Simulationsdomain gleichverteilt gesamplet, von denen der Punkt mit der höchsten VEGF-Konzentration ausgewählt wird. Mit anderen Worten wird das Gefäßwachstum im nächsten Wachstumsschritt der Perfusion dieses einen Punktes untergeordnet.
Als nächstes wird bestimmt, an welchem Knotenpunkt genau aus diesem Zweck das Wachstum einsetzen soll. Dazu werden die N_{n} Kontenpunkte ausgewählt, die zum ausgewählten Punkt in der Simulationsdomain am nächsten sind. Für das Wachstum wird von diesen schließlich der Knotenpunkt mit der höchsten VEGF-Konzentration bestimmt.
Da das Wachstum an einem Wurzelknotenpunkt die Initialisierung unerwünscht abändern würde und deswegen nicht erlaubt ist, findet Wachstum lediglich entweder an einem inneren Knotenpunkt oder an einem Blattknotenpunkt statt. Wird das Wachstum an einem inneren Knotenpunkt fortgesetzt, so wird es Sprossung genannt. Beim Wachstum an einem Blattknotenpunkt besteht die Möglichkeit entweder in Form einer Elongation den Zweig fortzuführen

oder eine Bifurkation zu bilden. Bei einer Bifurkation wird der Zweig beendet und zwei neue angefangen, bei Sprossung hingegen wird ein Zweig in zwei geteilt und zusätzlich ein neuer angefangen.
Tabelle 3.1 gibt eine Übersicht, wie bei den einzelnen Wachstumsarten jeweils verfahren wird. Für jede Art von Wachstum gilt zunächst: Das Wachstum an einem inneren oder an einem

Tabelle 3.1: Die drei Gefäßwachstumsarten

	Bifurkation	**Elongation**	**Sprossung**
Ort	Blattknotenpunkte		innere Knotenpunkte
Bedingung	$u \sim \mathcal{U}(0,1)$		
	$p_\mathrm{b}(X) = \Phi(\frac{\ln X - \mu_\mathrm{b}}{\sigma_\mathrm{b}})$		$p_\mathrm{s} = \sqrt{p_\mathrm{b}(X_\mathrm{p}) p_\mathrm{b}(X_\mathrm{d})}$
	$p_\mathrm{b} > u$	$p_\mathrm{b} \leqslant u$	$p_\mathrm{s} > u$
Radius r_neu	$r_\mathrm{c} = 2^{\frac{-1}{\gamma}} r$ $r_1 \sim \mathcal{N}(r_\mathrm{c}, \frac{r_\mathrm{c}}{32})$ $r_2 = (r^\gamma - r_1^\gamma)^{\frac{1}{\gamma}}$ $r_\mathrm{neu} \in \{r_1, r_2\}$	$r_\mathrm{neu} = r$	$r'_\mathrm{neu} = (\psi^\gamma - 1)^{\frac{1}{\gamma}} r$ $r_\mathrm{neu} = \begin{cases} \sim \mathcal{U}(r_\mathrm{min}, r'_\mathrm{neu}), & r_\mathrm{min} < r'_\mathrm{neu} \\ r_\mathrm{min}, & \text{sonst} \end{cases}$
Proportion X_neu	$X_\mathrm{neu} \sim \mathcal{N}_\mathrm{ln}(\mu_\mathrm{b}, \sigma_\mathrm{b})$		
Länge l_neu	$l_\mathrm{neu} = X_\mathrm{neu} r_\mathrm{neu}$		
Wachstumsrichtung $\boldsymbol{g}$	VEGF-Gradient		
zusätzliche Ausrichtung(en)	ϕ_l (3.2), ϕ_r (3.3); Rotation um den Normalen in der Bifurkationsebene; Bifurkationsrelokation	–	Bifurkationsrelokation

Blattknotenpunkt tritt nur ein, wenn die in Tabelle 3.1 dargestellte Bedingung erfüllt wird. Dazu wird an Blattknotenpunkten die Bifurkationswahrscheinlichkeit p_b, an inneren Knotenpunkten die Sprossungswahrscheinlichkeit p_s bestimmt und mit einer gleichverteilten Zufallszahl u verglichen. Beide Wahrscheinlichkeiten basieren auf der kumulierten Verteilungsfunktion Φ der logarithmisch normal verteilten Zufallsgröße X mit einem Mittelwert von μ_b und einer Standardabweichung von σ_b. X bezeichnet dabei das Verhältnis von Länge und Radius l/r eines Gefäßzweiges. An den Blattknotenpunkten wird X auf den Zweig bezogen, der im Blattknotenpunkt endet und im nächsten, proximalen Bifurkationsknotenpunkt seinen Anfang

hat. An einem inneren Knotenpunkt, werden der proximale und der distale Teil des gesplitteten Zweiges betrachtet und die Proportionen X_p bzw. X_d auf diese bezogen. Die Zufallsgröße X führt umso wahrscheinlicher zu einer Bifurkation oder Sprossung, je größer sie ist. Dies ist damit zu erklären, dass X umso größer ist, je länger die Strecke an einem Zweig ist, an der keine Abzweigungen vorkommen. Kommt an einer langen Strecke keine Abzweigung vor, so steigt die Wahrscheinlichkeit für eine Bifurkation oder Sprossung. Sprossungen kommen in der Mitte eines Zweiges wahrscheinlicher vor, da dann das Produkt $p_\mathrm{b}(X_\mathrm{p})p_\mathrm{b}(X_\mathrm{d})$ am größten ist. Ist die Bedingung zum Wachstum erfüllt, wird zunächst der neue Radius berechnet. Im Falle einer Bifurkation wird initial angenommen, dass die Radien der beiden Zweige gleich sind. Setzt man $r_\mathrm{l} = r_\mathrm{r}$, ergibt Gleichung (3.1) den Radius r_c. Dieser wird mit einem normalverteilten Rauschen versetzt und wird dann für den einen Zweig der Bifurkation verwendet. Der Radius des anderen wird nach Murrays Gesetz aus dem proximalen und dem bereits errechneten Radius des einen Zweiges aus Gleichung (3.1) bestimmt. Bei einer Elongation ist der Radius des neuen Segments per Definition dem des proximalen Segments gleich. Findet eine Sprossung statt, wird zunächst der sich aus Murrays Gesetz ergebende Radius r'_neu bestimmt. Dabei wird der ursprüngliche Radius des Zweiges r mit einem Faktor $\psi > 1$ skaliert, dadurch wird das Intervall $[r_\mathrm{min}, r'_\mathrm{neu}]$ erweitert und damit dem neuen Sprossradius eine größere Variabilität verliehen. Der Radius r_min bezeichnet dabei den kleinsten Radius, der sich durch Gefäßwachstum ergeben kann. Der tatsächliche Radius des neuen Zweiges r_neu wird dann nach einer Gleichverteilung zufällig aus dem Intervall $[r_\mathrm{min}, r'_\mathrm{neu}]$ gewählt. Ist jedoch $r'_\mathrm{neu} \leqslant r_\mathrm{min}$, wird r_neu zu r_min.
Nachdem der Radius des neuen Zweiges oder bei Bifurkationen der neuen Zweige bestimmt ist, wird die Länge des neuen Segments l_neu über eine zufällig, nach der logarithmischen Normalverteilung ausgewählten Verhältnis X_neu berechnet. μ_b und σ_b ergeben sich aus der beobachteten logarithmisch normalen Verteilung von X.
Sind der Radius und die Länge des neuen Zweiges bzw. der neuen Zweige bekannt, muss lediglich die Wachstumsrichtung bestimmt werden, um das neue Segment vollständig zu definieren. Die Wachstumsrichtung entspricht bei einer Elongation oder Sprossung der Richtung des Gradienten des VEGF-Feldes in dem zum Wachstum ausgewählten Knotenpunkt. Zum Verständnis der Wachstumsrichtung bei Bifurkationen wird Bild 3.11 betrachtet. Die Vektoren $\boldsymbol{p}_\mathrm{p}$, $\boldsymbol{p}_\mathrm{b}$, $\boldsymbol{p}_\mathrm{r}$ und $\boldsymbol{p}_\mathrm{l}$ sind die Ortsvektoren des proximalen Knotenpunktes, des zum Wachstum ausgewählten Blattknotenpunktes und der neuen Blattknotenpunkte rechts sowie links. Zunächst wird die Bifurkationsebene definiert. Diese wird durch den Vektor $\boldsymbol{p}_\mathbf{b} - \boldsymbol{p}_\mathbf{p}$ und dem VEGF-Gradienten aufgespannt. Die Mittellinien der beiden neuen Segmente sollen in dieser Ebene liegen. Als nächstes werden die Bifurkationswinkel ϕ_l bzw. ϕ_r berechnet und es wird bestimmt, welches Segment das linke und welches das rechte wird. Dazu wird gefordert, dass das der Winkel zwischen dem Richtungsvektor der größeren Abzweigung und dem VEGF-Gradienten minimal ist. Somit wären die neuen Segmente vollständig bestimmt, jedoch kann es vorkommen, dass

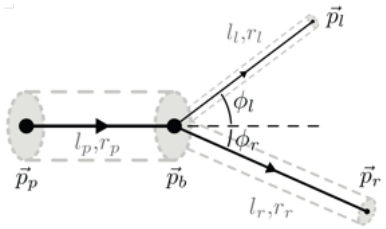


Bild 3.11: Aufbau einer Bifurkation nach [Sch+12]

die durch die Bifurkationswinkel bestimmte Wachstumsrichtungen stark von der Richtung des VEGF-Gradienten abweichen. Um diesem Problem entgegenzuwirken, werden die neuen Segmente zusätzlich so um den Normalen der Bifurkationsebene rotiert, dass die Wachstumsrichtung des größeren Segmentes mit der Richtung des VEGF-Gradienten koinzidiert.
Im Falle einer Bifurkation oder Sprossung werden außerdem die zum Wachstum ausgewählten Knotenpunkte relokalisiert, sodass die neu entstandene Verzweigung Gleichung (3.2) und Gleichung (3.3) entspricht.

3.2.3 Remodellierung

Durch Sprossen kann lokal das Murray Gesetz in Gleichung (3.1) verletzt werden. Aus diesem Grund werden nach dem Wachstumsschritt vom ausgewählten Knotenpunkt bis hin zurück zum entsprechenden Wurzelknotenpunkt die Radien gemäß Gleichung (3.1) aktualisiert. Der Prozess wird als Radius-Backpropagation bezeichnet. Bei Änderung der Radien in Bifurkationspunkten oder infolge einer vorherigen, distalen Bifurkationsrelokation kann es vorkommen, dass sich der Bifurkationsknotenpunkt nicht mehr an der nach Gleichung (3.2) und Gleichung (3.3) optimalen Stelle befindet. Dies wird entsprechend korrigiert.
Die Änderung der Radien führt zu einer Zunahme in proximaler Richtung. Diese Zunahme hat zur Folge, dass die bei der Initialisierung vorgegebenen Anfangsradien nicht mehr gelten. Deshalb werden nach der Radius-Backpropagation die Radien reskaliert. Bezeichnet r_0 den initialen Radius und r' den aus den Änderungen resultierenden Radius an einem Wurzelknotenpunkt, so werden alle Radien mit dem Faktor r_0/r' skaliert. Diese Skalierung verletzt das Murray Gesetz nicht, da es alle Äste einer Bifurkation betrifft.
Nach der Remodellierung können bei manchen Zweigen sehr kleine Radien auftreten. Deswegen wird nach der Remodellierung nach Zweigen mit einem zu kleinen Radius gesucht und diese aus dem Gefäßbaum entfernt.

3.2.4 Degeneration

Der Degenerationsschritt setzt sich aus drei Teilen zusammen. Zum einen werden Blattsegmente, deren Radius über einem bestimmten Wert liegt, geschrumpft. Zum anderen werden überflüssige Blattsegmente entfernt und als letztes werden Blattsegmente mit einem zu hohen Verhältnis X aus dem Gefäßbaum gelöscht.
Das Schrumpfen von Blattsegmenten ist notwendig, da am Ende der Modellierung alle Blattsegmente im kapillaren Bereich sein sollen. Fände dieser Schritt nicht statt, gäbe es Blattsegmente, die sich nicht oft genug verästelt haben, um in den kapillaren Bereich zu kommen. Solche müssen folglich während der Modellierung von Schritt zu Schritt explizit geschrumpft werden. Der Radius nach dem Schrumpfen wird mittels Interpolation erhalten:

$$r' = (1 - \lambda_s)r + \lambda_s r_{\text{prune}} \tag{3.4}$$

Der Radius r_{prune} ist der absolut kleinste erlaubte Radius im Gefäßbaum. Da Radien geändert werden, kann Murrays Gesetz verletzt werden, aus diesem Grund findet nach dieser Degeneration eine Remodellierung an den betroffenen Knotenpunkten statt.
Fällt die VEGF-Konzentration an einem Blattknotenpunkt unter einen bestimmten Schwellwert, wird das Blattsegment als überflüssig gezählt. Es sollen jedoch nicht alle derartige Blattsegmente entfernt werden, damit eine zu exzessive Degeneration verhindert wird. Aus diesem Grund wird das Blattsegment nur entfernt, wenn zusätzlich auch noch eine gleichverteilte Zufallszahl einen Schwellwert überschreitet.
Für zu hohe Proportionen wird – anstatt für X einen Schwellwert zu definieren – die Bifurkationswahrscheinlichkeit betrachtet. Ist diese zu hoch, indiziert das eine abnormale Proportion von l und r. Blattsegmente werden somit entfernt, wenn die Bifurkationswahrscheinlichkeit im Blattknotenpunkt einen bestimmten Schwellwert übersteigt.
Im Allgemeinen muss bei der Entfernung von Segmenten beachtet werden, dass es dazu kommen kann, dass aus Bifurkationspunkten innere Knotenpunkte werden. In einem solchen Fall findet eine Remodellierung statt und der betroffene Knotenpunkt wird relokalisiert, um den Gefäßverlauf glatter zu gestalten.

3.2.5 Multiskalenansatz

Das in Bild 3.9 dargestellte Vorgehen ist zudem in einen Multiskalenansatz eingebettet. Dadurch kann sich das Wachstum stufenweise auf immer feinere Strukturen konzentrieren. Der Wachstumsprozess startet mit einer downgesampleten Instanz der Simulationsdomain. Die gewünschte Perfusion ist erreicht, wenn eine bestimmte Mindestanzahl von Voxeln einen zuvor festgelegten Schwellwert der VEGF-Konzentration erreicht hat. Ist die geforderte Perfusion in

der Simulationsdomain erreicht, findet ein Upsampling der Simulationsdomain statt und der Wachstumsprozess setzt sich fort. Bei insgesamt N_s Skalen findet die Skalierung $T_\mathrm{s} : \Omega_0 \to \Omega_\mathrm{s}$ für die aktuelle Skala s in Form einer Transformation der Koordinaten $\boldsymbol{x}$ der ursprünglichen Simulationsdomain Ω_0 zu denen von Ω_s statt:

$$\boldsymbol{x} \mapsto s_0^{-\frac{N_\mathrm{s}-s}{N_\mathrm{s}-1}} \boldsymbol{x} \tag{3.5}$$

An die Transformation zwischen den Skalen ist die im Abschnitt 3.2.4 beschriebene Degeneration gebunden. Sie findet immer unmittelbar vor der Transformation T_s statt. Bei $s = N_\mathrm{s}$ wird der Skalierungsfaktor $s_0^{-\frac{N_\mathrm{s}-s}{N_\mathrm{s}-1}}$ zu 1 und der Gesamtprozess ist beendet, wenn die gewünschte Perfusion in dieser Skala erreicht wird.

3.3 DeepVesselNet

Tetteh et al. stellen in [Tet+18] zwei CNN-Architekturen für die Segmentierung von Gefäßen sowie die Erkennung von Mittellinien und Bifurkationen in angiographischen[3] Aufnahmen vor. Diese drei Aufgaben interpretieren sie als drei binäre Klassifikationsaufgaben. Ihre Arbeit fokussiert insbesondere auf die Fragestellungen des hohen Rechenaufwands, des Klassenungleichgewichts und des Mangels von echten Daten. Die Ansätze von Tetteh et al. zu diesen Fragestellungen werden im Folgenden erläutert. Anschließend werden die von Tetteh et al. vorgeschlagenen Netzarchitekturen und die verwendeten Hyperparameter vorgestellt.

3.3.1 Reduzierung des Rechenaufwands

Die Verarbeitung von 3D-Bilddaten mittels CNNs ist deutlich rechenintensiver, als von 2D-Bilddaten. Es besteht die Möglichkeit 3D-Bilddaten als 2D-Schichtbilder zu betrachten, doch auf diese Weise gehen in der Regel relevante Informationen verloren, sodass die Verfolgung des 3D-kurvenartigen Verlaufs von Gefäßen problematisch wird. Tetteh et al. versuchen einen Kompromiss zwischen Reduzierung des Rechenaufwands und Informationsverlust zu finden und schlagen vor, die 3D-Faltung durch einen Crosshair-Filter zu ersetzen, vgl. Bild 3.12. Bei der herkömmlichen 3D-Faltung wird analog zur Faltungsoperation bei 2D-Bildern ein 3D-Kernel über jedes Voxel gelegt und die Voxel, die darunter liegen mit dem Kernel verrechnet. Beim Corosshair-Filter sollen anstelle des 3D-Kernels M die drei 2D-Kernel M_i, M_j und M_k benutzt werden, die jeweils senkrecht auf die drei Raumachsen stehen. Bei der bereits in Gleichung (2.51) verwendeter Nomenklatur, bei der g das Eingangs- und h das Ausgangsbild, f das Kernel und K seine Kantenlänge bezeichnet, wird der Crosshair-Filter durch die Gleichungen

[3] Die Angiographie ist das Darstellen von Gefäßen mittels Kontrasmittel in der Radiologie.

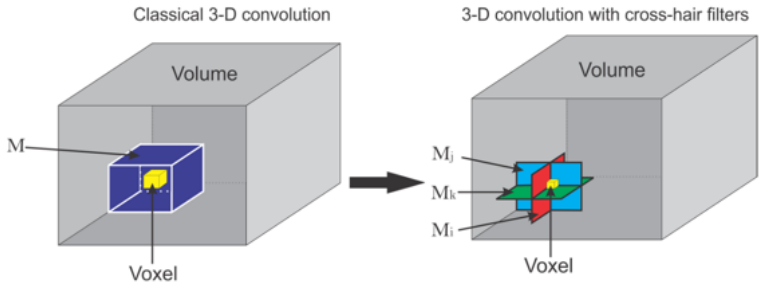


Bild 3.12: Der Crosshair-Filter nach [Tet+18]

(3.6) bis (3.9) definiert.

$$\begin{aligned} h(x,y,z) = &\sum_{s=1}^{K}\sum_{t=1}^{K} g(x,S,T) f_{\mathrm{x}}(s,t) \\ &+ \sum_{r=1}^{K}\sum_{t=1}^{K} g(R,y,T) f_{\mathrm{y}}(r,t) \\ &+ \sum_{r=1}^{K}\sum_{s=1}^{K} g(R,S,z) f_{\mathrm{z}}(r,s) \end{aligned} \tag{3.6}$$

$$R = x + r - (1 + [K/2]) \tag{3.7}$$

$$S = y + s - (1 + [K/2]) \tag{3.8}$$

$$T = z + t - (1 + [K/2]) \tag{3.9}$$

Verglichen mit der gewöhnlichen 3D-Faltung, bei der pro Voxel K^3 Multiplikationen und $K^3 - 1$ Additionen erfolgen, müssen beim vorgeschlagenen Crosshair-Filter lediglich $3K^2$ Multiplikationen und $3K^2 - 1$ Additionen ausgeführt werden. Daraus folgt, dass die Reduzierung des Rechenaufwandes durch die Anwendung des Crosshair-Filters ab $3 \leqslant K$ einsetzt.

3.3.2 Kompensierung des Klassenungleichgewichts

Ein Ungleichgewicht der Klassen besteht, wenn im Datensatz das Vorkommen der Klassen nicht (ungefähr) gleichverteilt ist. Dies ist insbesondere der Fall bei Gefäßen. Vom Gesamtvolumen machen Gefäßvoxel lediglich einen kleinen Teil aus, von denen die Mittellinienvoxel wiederum und von denen die Bifurkationsvoxel ebenfalls nur einen Bruchteil ausmachen. In den Datensätzen von Tetteh et al. stellen bspw. weniger als 2,5 % aller Voxel ein Gefäß dar.
Wie bereits im Abschnitt 2.3 ausgeführt, muss bei einem Klassenungleichgewicht die Loss-

Function entsprechend angepasst werden. Tetteh et al. nehmen dazu die für binäre Klassifikationsaufgaben übliche Cross-Entropy als Grundlage, die folgenderweise definiert ist:

$$L = -\frac{1}{|Y_+| + |Y_-|} \left(\sum_{n \in Y_+} \log P_n(1) + \sum_{n \in Y_-} \log P_n(0) \right) \tag{3.10}$$

Es werden jeweils die Logarithmen der Wahrscheinlichkeiten $P_n(c)$ addiert, dass ein Sample der Klasse c zugeordnet wird, zu der sie auch wirklich gehört. In der ersten Summe werden dazu die Samples n aus der positiven Klasse Y_+, in der zweiten die negative Klasse Y_- betrachtet. Anschließend wird über alle Samples $N = |Y_+| + |Y_-|$ gemittelt. Die Terme $|Y_+|$ bzw. $|Y_-|$ beziehen sich auf die Anzahl der Samples die in Y_+ bzw. Y_- enthalten sind.
Die Formulierung in Gleichung (3.10) wird um einen Gewichtungsparameter β erweitert, der das Klassenungleichgewicht ausgleichen soll. Er ist definiert als $\beta = \frac{|Y_-|}{|Y_+|+|Y_-|}$ bzw. $1 - \beta = \frac{|Y_+|}{|Y_+|+|Y_-|}$ und wird entsprechend Gleichung (3.11) verwendet.

$$L = -\beta \sum_{n \in Y_+} \log P_n(1) - (1 - \beta) \sum_{n \in Y_-} \log P_n(0) \tag{3.11}$$

Anders jedoch als in Gleichung (3.10) durch den Faktor $\frac{1}{|Y_+|+|Y_-|}$, sind die Summenterme in Gleichung (3.11) durch β bzw. $(1 - \beta)$ nicht unabhängig von der Anzahl von Samples $N = |Y_+| + |Y_-|$. Die Summen werden in Gleichung (3.11) umso größer, über je mehr Samples summiert wird. Dies führt zu einer Instabilität in der Gradientenberechnung für große N. Außerdem führt die Formulierung nach Gleichung (3.11) dazu, dass bei bspw. $\beta = 0{,}95$ die false Positives mit einem sehr kleinen Gewicht in die Loss-Function einfließen, was zu einer hohen FPR führen kann. Um diesem und der Instabilität entgegenzuwirken, entwerfen Tetteh et al. einen neuen Gewichtungsfaktor statt β und erweitern die Loss-Function zusätzlich um einen zweiten Term:

$$L = L_1 + L_2 \tag{3.12}$$

$$L_1 = -\frac{1}{|Y_+|} \sum_{n \in Y_+} \log P_n(1) - \frac{1}{|Y_-|} \sum_{n \in Y_-} \log P_n(0) \tag{3.13}$$

$$L_2 = -\frac{\gamma_1}{|Y_+|} \sum_{n \in Y_{\mathrm{f}+}^{\mathrm{pred}}} \log P_n(0) - \frac{\gamma_2}{|Y_-|} \sum_{n \in Y_{\mathrm{f}-}^{\mathrm{pred}}} \log P_n(1) \tag{3.14}$$

$$\gamma_1 = 0{,}5 + \frac{1}{|Y_{\mathrm{f}+}^{\mathrm{pred}}|} \sum_{n \in Y_{\mathrm{f}+}^{\mathrm{pred}}} |P_n(0) - 0{,}5| \tag{3.15}$$

$$\gamma_2 = 0{,}5 + \frac{1}{|Y_{\mathrm{f}-}^{\mathrm{pred}}|} \sum_{n \in Y_{\mathrm{f}-}^{\mathrm{pred}}} |P_n(1) - 0{,}5| \tag{3.16}$$

Der erste Term L_1 ist eine numerisch stabilere Version von Gleichung (3.11). Der zweite Term L_2 bestraft die false positives bzw. die false negatives. Dies soll jedoch nicht mit der gleichen Gewichtung in die Loss-Function einfließen wie die Bestrafungen aus dem ersten Term. Aus diesem Grund wird zusätzlich mit den Faktoren γ_1 und γ_2 gewichtet. Diese sind ein Maß dafür, wie groß die Abweichung zwischen den Wahrscheinlichkeiten der false positives bzw. negatives, dass sie ihrer tatsächlichen Klasse angehören und einer zufälligen Vorhersage von 0,5 ist. Das heißt der zweite Term konzentriert sich auf die false positives bzw. negatives, deren Vorhersagen von einer zufälligen möglichst signifikant abweichen.

3.3.3 Datensätze

Insgesamt benutzen Tetteh et al. drei verschiedene Datensätze für ihre Arbeit. Von denen sind zwei echte, gelabelte Datensätze, bestehend aus Aufnahmen mittels Magnetoresonanzangiographie sowie mittels Synchrotron-Röntgen-Mikrotomographie. In diesen beiden Fällen entstehen die Bilder durch die Verwendung von Kontrastmitteln, die die Gefäße in den Aufnahmen hervorheben.
Da echte, gelabelte Datensätze, jedoch äußerst begrenzt zur Verfügung stehen, entscheiden sich Tetteh et al. zusätzlich Daten synthetisch zu generieren. Hierfür benutzen sie die Algorithmen von Schneider et al., die bereits in Abschnitt 3.2 vorgestellt wurden. Den daraus resultierenden Graphen wandeln sie in ein 3D-Volumen um. Entsprechend angiographischen Aufnahmen werden die Intensitätswerte für Gefäße im Intervall $[128, 255]$, für den Hintergrund im Intervall $[0, 100]$ zufällig definiert. Die Intensitätswerte werden zusätzlich mit Gaußschem Rauschen überlagert, bei dem der Mittelwert dazu zufällig aus dem Intervall $[-5, 5]$ und die Standardabweichung aus $[-15, 30]$ gewählt wird. Durch diesen Schritt kommen die synthetischen Daten den echten näher.

3.3.4 Netzwerkarchitekturen und Hyperparameter

Tetteh et al. stellen zwei Netzarchitekturen vor. Die beiden bezeichnen sie als *DeepVesselNet-VNet* (*DVN-VNet*) sowie *DeepVesselNet-FCN* (*DVN-FCN*). Die Netzarchitekturen sind jeweils in Bild 3.13 und Bild 3.14 dargestellt. Die Netzarchitektur für das DeepVesselNet-VNet entnehmen Tetteh et al. aus einer Arbeit von Milletari et al. [MNA16]. Milletari et al. stellen in ihrer Arbeit das *V-Net* vor, das über eine dem – in Abschnitt 2.4.3 bereits erläuterten – U-Net ähnliche Architektur verfügt. Das V-Net arbeitet jedoch im Gegensatz zum U-Net nicht mit 2D-, sondern mit 3D-Bildern. Demzufolge findet im V-Net anstelle der 2D-Faltung eine 3D-Faltung statt. Tetteh et al. ersetzen jedoch die 3D-Faltung des Rechenaufwands halber durch den in Abschnitt

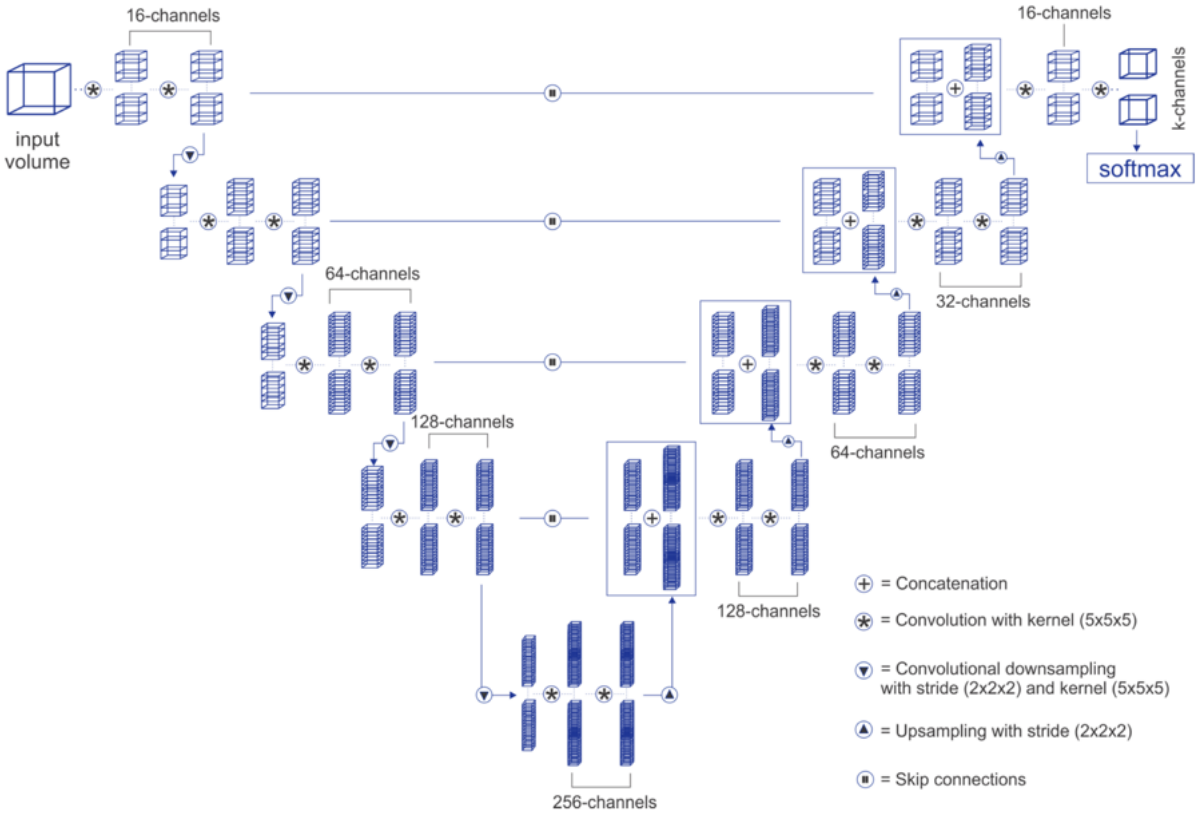


Bild 3.13: DeepVesselNet-VNet nach [Tet+18]

3.3.1 vorgestellten Crosshair-Filter. Diese Architektur benutzen sie, um den Gewinn bezüglich Speicherbedarf und Rechenzeit, der aus der Verwendung des Crosshair-Filters resultiert, zu evaluieren. Zudem untersuchen sie, ob Speicherbedarf und Rechenzeit einen Einfluss auf die Genauigkeit haben.

Das DeepVesselNet-FCN besteht aus vier Faltungslayern und einem fully connected Layer mit einer Sigmoid-Aktivierungsfunktion für die Klassifikation. Tetteh et al. ändern die FCN-Architektur aus Abschnitt 2.4.2 insofern, dass die Kernelgrößen so gewählt werden, dass kein Downsampling stattfindet. Das bedeutet, dass die Bildgrößen über alle Layer gleich bleiben, sodass auch kein Upsampling notwendig ist. Diese Änderung hat den Grund, dass die Gefäße, Mittellinien und Bifurkationen, die für die Segmentierung von Interesse sind, im Gesamtvolumen feine Strukturen sind. Diese feinen Strukturen würden beim Downsampling verloren gehen, infolge dessen bei der Faltung über die Voxelnachbarschaften eine Mittelung stattfindet. Die Segmentierungsaufgabe wird in drei binäre Klassifikationsaufgaben zerlegt, sodass hierfür drei DVN-FCNs verwendet werden. Die drei Klassifikationsaufgaben betreffen die Erkennung von Gefäßen im Allgemeinen, von Mittellinien sowie von Bifurkationen. Die Gewichte

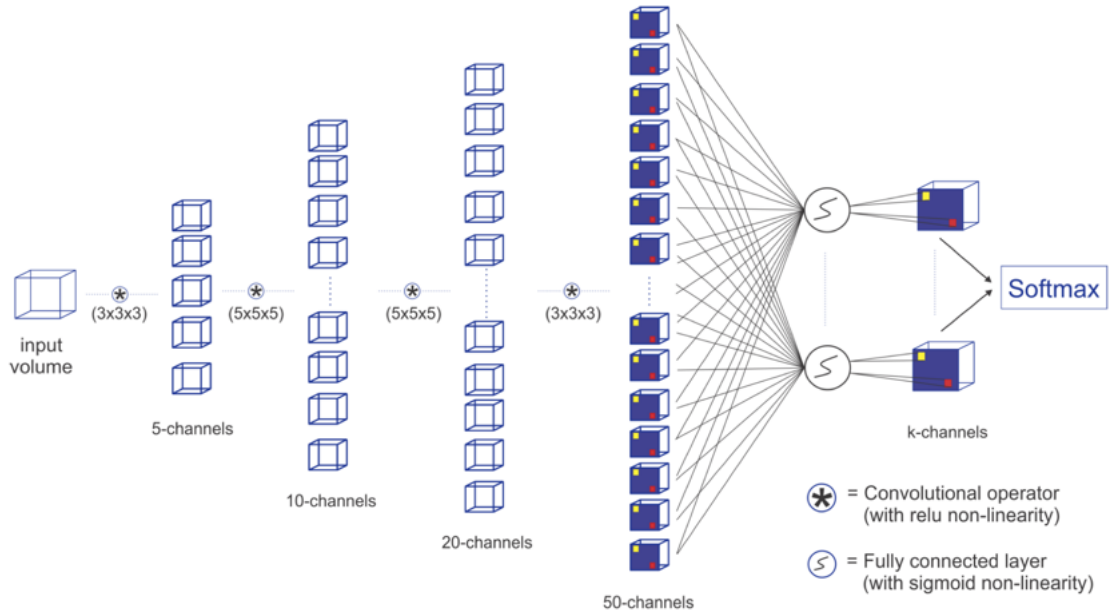


Bild 3.14: DeepVesselNet-FCN nach [Tet+18]

werden zufällig initialisiert, indem die Anfangswerte aus einer Gleichverteilung im Intervall $\left[-1/\sqrt{k_x k_y k_z}, 1/\sqrt{k_x k_y k_z}\right]$ gesamplet werden. Dabei steht $(k_x \times k_y \times k_z)$ für die Größe des Kernels im jeweiligen Layer. Aus den Gesamtvolumina der in Abschnitt **??** vorgsestellten Datensätzen werden $(64 \times 64 \times 64)$ große Volumina extrahiert. Mit den aus den synthetischen Daten stammenden Volumina werden die Netze vortrainiert und mit denen aus echten Daten stammend zu Ende trainiert. Im ersten Schritt wird eine Lernrate von 0,01, im zweiten eine von 0,001 verwendet.
Für das Netz für die Mittellinienerkennung wird als zusätzlicher Input der Output – genauer gesagt die vorhergesagten Wahrscheinlichkeiten – des Netzes für die Gefäßerkennung benutzt. Für das Netz für die Bifurkationserkennung werden als Input die Gefäß- sowie die Mittellinienvorhersagen verwendet.

4 Daten

Zur Verfügung dieser Arbeit steht der in Abschnitt 3.1.1 beschriebene Datensatz. Dieser liegt sowohl in der rohen RGB-Form, als auch in der binären, vorverarbeiteten Form vor. Das zu erarbeitende neuronale Netz soll dabei aufgrund der bereits komprimierten Informationsdichte durch die Vorverarbeitung letzteren benutzen. Für das Training werden gelabelte Daten gebraucht, die vorliegenden sind jedoch nicht solche. In diesem Fall können entweder die vorliegenden Daten gelabelt oder synthetische Daten generiert werden. Da das Labeln der vorliegenden Daten einen enormen und nicht genau abschätzbaren Zeitaufwand nach sich zieht, werden synthetische Daten hergestellt, die dem vorverarbeiteten Datensatz möglichst ähnlich sein sollen. Im Folgenden wird der verwendete Prozess der synthetischen Gefäßbaumgenerierung beschrieben. Im Anschluss wird erläutert, wie aus diesem die für das Training zu benutzende Samples erstellt werden.

4.1 Synthetische Datenherstellung

Die synthetische Datenherstellung basiert auf dem Verfahren von Schneider et al. [Sch+12], das in Abschnitt 3.2 vorgestellt wurde. Schneider et al. stellen jedoch nicht Daten für einen kompletten Gefäßbaum eines ganzen Organs her, wie dies im Rahmen dieser Arbeit beabsichtigt ist. In ihrer Arbeit bewegen sich die Gefäßradien lediglich in Nähe des kapillaren Bereichs zwischen 2 µm und 17 µm. Das heißt der Faktor zwischen dem kleinsten und größten Radius beträgt nur 8,5 im Gegensatz zu dem in Abschnitt 3.1.1 berechneten, für diese Arbeit beabsichtigten Faktor 500. Dieser Unterschied erlaubt es nicht, das Verfahren von Schneider et al. ohne Änderungen zu übernehmen. Das Verfahren muss geändert werden, sodass die deutlich größere Skala von Gefäßen aufgelöst werden kann, ohne dass es in untragbaren Rechenzeiten resultiert. Insgesamt muss also das Verfahren möglichst vereinfacht werden, die Realistik der synthetischen Daten sollte dadurch jedoch möglichst erhalten bleiben. Das genaue Problem und die verwendeten Ansätze werden im Folgenden beschrieben. Schneider et al. bilden den Sauerstoffverbrauch sowie die Oxygenation und die VEGF-Konzentration in der Simulationsdomain jeweils in einer Map ab. Sie arbeiten mit einem Gesamtvolumen von $992\,\mu\mathrm{m} \times 864\,\mu\mathrm{m} \times 1760\,\mu\mathrm{m}$, das sie mit einem isotropen Voxel-Spacing von 32 µm in ein Volumen von $31\,\mathrm{px} \times 27\,\mathrm{px} \times 55\,\mathrm{px}$ diskretisieren. Würde man beim vorliegenden Volumen des Schweineherzens von etwa $39\,\mathrm{mm} \times 33\,\mathrm{mm} \times 33\,\mathrm{mm}$ analog vorgehen, würde man ein

diskretes Volumen der Größe $1219\,\mathrm{px} \times 1031\,\mathrm{px} \times 1031\,\mathrm{px}$ erhalten. Dies entspricht insgesamt $1{,}3 \cdot 10^9$ Voxeln. Werden diese in MATLAB als *Single-Precision Floating Point* abgespeichert, ist ein Speicherplatz von etwa 130 GB für eine solche Map notwendig. Eine solche Datengröße würde den Rechenaufwand für die Datenherstellung steigern, sodass es vermieden werden soll, Volumina explizit zu beschreiben und alternative Ansätze erwünscht sind.
Kann eine Oxygenationsmap bzw. VEGF-Map nicht in der von Schneider et al. beschriebenen Form benutzt werden, sind drei Stellen des Generierungsverfahrens davon betroffen. Welcher Knotenpunkt gesamplet wird, wird anhand der VEGF-Konzentration ermittelt und die Wachstumsrichtung des neuen Segments an dieser Stelle hängt vom VEGF-Gradienten ab. Die Verfahren zum Knotenpunktsampling und zur Ermittlung der Wachstumsrichtung müssen daher umgestellt werden. Zusätzlich muss auch der Multiskalenansatz angepasst werden, da sich dieser beim Wechsel zwischen den Auflösungsstufen auch auf die VEGF-Map bezieht.

4.1.1 Histogrammbasiertes Knotenpunktsampling

Zunächst wird die Vereinfachung getroffen, dass der Sauerstoffverbrauch gleich verteilt ist. In diesem Fall wird keine Map für den Sauerstoffverbrauch gebraucht. Des Weiteren findet die Angiogenese immer da statt, wo die Perfusion noch nicht ausreichend ist, an solchen Stellen ist die VEGF-Konzentration hoch. Vereinfacht gesehen sind die Stellen mit hoher VEGF-Konzentration jene, an denen die Gefäßdichte verhältnismäßig niedrig ist. Das Wachstum muss demnach das Ziel haben, diese Stellen zu erreichen. Ein Maß für die Verteilung der Gefäße im Volumen ist die Häufigkeitsverteilung der Koordinaten der Knotenpunkte. Das Sampling des Knotenpunktes, an dem sich das Wachstum fortsetzen soll, soll daher auf der Häufigkeitsverteilung bzw. graphisch gesehen auf dem Histogramm der Knotenpunktkoordinaten basieren. In [Sch+12] samplen Schneider et al. N_p Punkte aus der Simulationsdomain, von denen der Punkt mit der höchsten VEGF-Konzentration ausgewählt wird. Die N_n nächsten Knotenpunkte zu diesem Punkt werden dann ausgewählt. Von denen wird einer für die Fortsetzung des Wachstums bestimmt. Dieser Prozess wird durch Algorithmus 2 ersetzt bzw. angepasst.
Als Input nimmt der Algorithmus den aktuellen Gefäßbaumgraphen G, die Simulationsdomain $domain$ – die das zu perfundierende Volumen repräsentiert – sowie die aus der aktuellen Auflösungsstufe s folgende Auflösung r. Zurückgegeben wird der Knotenpunkt n, an dem sich das Wachstum fortsetzen soll.
Ausgangspunkt sind die Histogramme $hist_i(b)$ und $hist_{\mathrm{Dm},i}(b)$ für die Koordinaten der Knotenpunkte sowie die der Domain in Raumrichtung $i \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$. Die Variable b steht dabei für Nummer eines Bins. Bei der Erstellung werden die Koordinaten jeweils auf insgesamt r Bins aufgeteilt, sodass $b \in [0, r]$ gilt. Die unterste Bingrenze beträgt dabei 0, die oberste ist gleich der Breite der Domain in der entsprechenden Raumrichtung. Die Histogramme werden außerdem so genormt, dass statt der absoluten die relativen Häufigkeiten im Histogramm enthalten sind

Algorithmus 2: Histogrammbasiertes Punktsampling

Input: Gefäßbaumgraph G
diskerete, binäre Simulationsdomain $domain$
Auflösung $r = 2^s$ mit $s \in \{1, 2, \ldots, S\}$
Anzahl naheliegender Knotenpunkte N_n
Output: Knotenpunkt n

foreach $i \in \{x, y, z\}$ **do**
  $hist_i^0(b) \leftarrow$ getNormalizedHistogram(G, r Bins);
  $hist_{\mathrm{Dm},i}^0(b) \leftarrow$ getNormalizedHistogram($domain$, r Bins);
  $P_i(b) = \max(0, hist_{\mathrm{Dm},i}^0(b) - hist_i^0(b))$;
  if $\sum_{b=1}^{r} P_i(b) = 0$ **then**
    foreach $b \in [1; r]$ **do**
      $P_i(b) \leftarrow 1$;
    end
  end
  $\tilde{b}_i \leftarrow$ sampleFromPmf($P_i(b)$);
  $p_i \leftarrow$ sampleBetweenBinEdges($\tilde{b}_i, hist_i^0$);
end
$\boldsymbol{p} = (p_\mathrm{x}, p_\mathrm{y}, p_\mathrm{z})^T$;
while $domain(\boldsymbol{p}) = 0$ **do**
  foreach $i \in \{x, y, z\}$ **do**
    $\tilde{b}_i \leftarrow$ sampleFromPmf(pmf_i);
    $p_i \leftarrow$ sampleBetweenBinEdges($\tilde{b}_i, hist_i^0$);
  end
  $\boldsymbol{p} = (p_\mathrm{x}, p_\mathrm{y}, p_\mathrm{z})^T$;
end
$N = \{\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_{N_\mathrm{n}}, \} \leftarrow$ findClosestNodes($\boldsymbol{p}$, N_n);
$S = \{s_1, s_2, \ldots, s_{N_\mathrm{n}}, \} \leftarrow$ getScores(N);
$n \leftarrow N(\mathrm{argmin}(S))$;

und die maximale relative Häufigkeit 1 beträgt. Die normierten Histogrammen werden mit $hist_i^0(b)$ und $hist_{\mathrm{Dm},i}^0(b)$ bezeichnet.
Der Grundansatz ist es, zunächst einen Bin aus dem Histogramm auszuwählen und einen Punkt $\boldsymbol{p} = (p_\mathrm{x}, p_\mathrm{y}, p_\mathrm{z})$ zwischen dessen unteren und oberen Grenze zufällig festzulegen. Der Bin soll jeweils aus den Bereichen der Histogramme $hist_i^0(b)$ gewählt werden, in denen die Diskrepanz zwischen den Histogrammen $hist_i^0(b)$ und $hist_{\mathrm{Dm},i}^0(b)$ am größten ist. Der Knotenpunkt n wird dann aus der Nähe von $\boldsymbol{p}$ bestimmt.
Die Histogramme $hist_i^0(b)$ und $hist_{\mathrm{Dm},i}^0(b)$ können auch als Ist- bzw. Soll-Verteilungen aufgefasst werden. Die Bins sollen immer so ausgewählt werden, dass durch das Wachstum am Knotenpunkt n die Ist-Verteilung der Soll-Verteilung näher kommt. Das Konzept veranschaulicht qualitativ Bild 4.1.

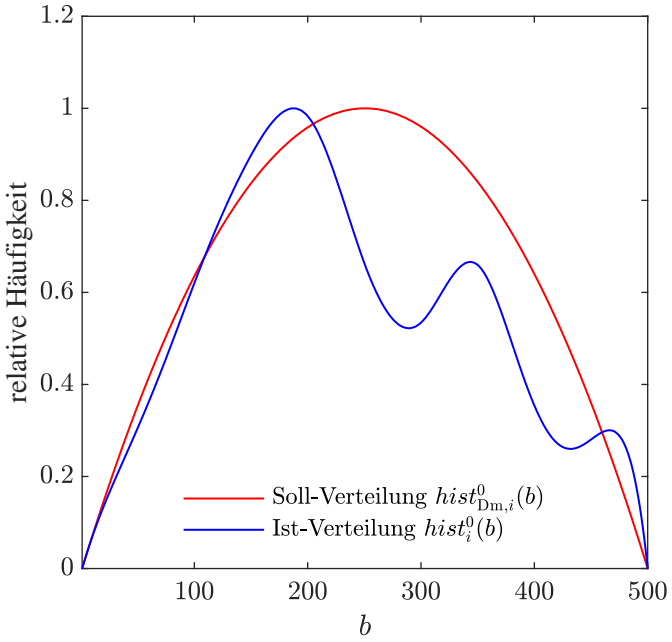


Bild 4.1: Histogramme $hist_{\mathrm{Dm},i}^0(b)$ und $hist_i^0(b)$ als Soll- und Ist-Verteilungen

Im nächsten Schritt wird jedem Histogrammbin eine Auswahlwahrscheinlichkeit $P_i(b) \in [0; 1]$ zugeteilt. Diese ist gleich der Differenz aus dem Domain-Histogramm und dem Knotenpunkt-Histogramm für die Raumrichtung i. Ist die Differenz groß, soll der Bin an der entsprechenden Stelle mit einer größeren Wahrscheinlichkeit ausgewählt werden. Negative Differenzen werden durch Null ersetzt, da diese Bins gar nicht ausgewählt werden sollen. Dies wäre hinsichtlich der Anpassung der Ist-Verteilung an die Soll-Verteilung kontraproduktiv, da es in diesem Schritt um die Bestimmung von Knotenpunkten für das weitere Wachstum geht und nicht um das

Entfernen von Gefäßen. Für den Fall, dass $P_i(b) = 0$ für alle b, werden alle Wahrscheinlichkeiten durch 1 ersetzt, andernfalls würde kein Bin ausgewählt. Dieser ideale Fall tritt ein, wenn $hist^0_{\mathrm{Dm},i}(b) - hist^0_i(b))$ gilt, das heißt wenn die Soll-Verteilung erreicht wurde. Aus dem Erreichen der Soll-Verteilung soll jedoch nicht folgen, dass kein Bin ausgewählt wird, sondern lediglich dass alle mit der gleichen Wahrscheinlichkeit ausgewählt werden sollen.
Anhand der Wahrscheinlichkeitsfunktion $P_i(b)$ wird schließlich ein Bin $\tilde{b}_i$ zufällig bestimmt. Zwischen dem linken und dem rechten Rand dieses Bins wird gleich verteilt die Koordinate i des Punktes $\boldsymbol{p}$ gesamplet. Anschließend wird überprüft, ob $\boldsymbol{p}$ in der Domain liegt. Es kann vorkommen, dass $\boldsymbol{p}$ außerhalb der Domain liegt, wenn die Domain nicht überall konvex ist. Liegt der Punkt $\boldsymbol{p}$ nicht in der Domain, werden solange neue $\tilde{b}_i$ gesamplet, bis er darin liegt.
Ist mittels Algorithmus 2 ein Punkt $\boldsymbol{p}$ gesamplet, werden entsprechend dem Vorgehen von Schneider et al. in [Sch+12] die N_n zu diesem Punkt nächsten Knotenpunkte ausgewählt. Für den in dieser Arbeit benutzten histogrammbasierten Ansatz wird von den N_n Knotenpunkten jener ausgewählt, bei dem in den umliegenden Bins die Häufigkeit der Knotenpunktkoordinaten den niedrigsten Wert aufweist. In einem Optimum, in dem die Häufigkeitswerte für alle drei Raumrichtungen minimal sind, ist auch ihr Produkt minimal. Somit wird ihr Produkt betrachtet und daraus der Score σ in einem beliebigen Punkt $\boldsymbol{x} = (x, y, z)^T$ wie folgt definiert:

$$\sigma(\boldsymbol{x}) = hist_\mathrm{x}(bin_\mathrm{x}(x)) \cdot hist_\mathrm{y}(bin_\mathrm{y}(y)) \cdot hist_\mathrm{z}(bin_\mathrm{z}(z)) \tag{4.1}$$

Dabei steht der Ausdruck bin_i für eine Funktion, die die x-, y- und z-Koordinaten jeweils auf die Bins der entsprechenden Raumrichtung mappt. Die Histogramme $hist_i(b)$ enthalten die nicht normierten, absoluten Häufigkeitswerte der r Bins.
Die Bins von $hist_i(b)$ können als Schichten der Domain mit einer Dicke r und die Häufigkeitswerte als die Anzahl der darin befindlichen Knotenpunkte aufgefasst werden. Bei dieser Auffassung stellt sich die Frage, ob statt Bins – bzw. statt drei Bins – dieser zweidimensionaler Art die Knotenpunkte nicht auch in einem kleinen Volumen um $\boldsymbol{p}$, das heißt in einem Bin dreidimensionaler Art, gezählt werden könnten. Da jedoch später beim Wechsel zwischen den Auflösungsstufen der Score in der ganzen Domain gefragt ist, ist es von Vorteil ihn möglichst rechengünstig zu definieren. Um für die Größenordnung der Kapillare sinnvolle Werte für die Häufigkeitsverteilung der Knotenpunktkoordinaten zu erhalten, sollten die Bins entsprechend klein definiert werden. Für eine Binkantenlänge in etwa 10-facher Länge der kleinsten Kapillardurchmesser sollte die Domain in mindestens $2^{10} = 1024$ Bins in jede Achsenrichtung aufgeteilt werden. Dies würde bei der dreidimensionalen Definition der Bins für jeden Iterationsschritt zu einer $1024 \times 1024 \times 1024$ großen 3D-Matrix führen. Da eine solch große Matrix eine große Rechenkapazität zufolge hat, wird entschieden, die obige, raumrichtungsweise Definition von σ zu bevorzugen. Dadurch kann die Anzahl der benötigten Bins deutlich, um

den Faktor von $\frac{(2^{10})^3}{3 \cdot 2^{10}} = \frac{1}{3} \cdot 2^{2 \cdot 10} \approx 349\,525$ reduziert werden.
Die Scores der N_n ausgewählten Knotenpunkten sollen nun verglichen werden und derjenige mit dem niedrigsten Score zum Wachstum ausgewählt werden. Scores verschiedener Punkte der Domain können allerdings in der in Gleichung (4.1) definierten Form nicht sinnvoll verglichen werden. Eine Scoredifferenz an zwei beliebigen Domainpunkten kann sowohl aus einer Differenz in der Häufigkeit der Knotenpunktkoordinaten folgen, oder aber dadurch bedingt sein, dass die beiden Punkte eine unterschiedliche Distanz zum Domainrand aufweisen. Den Grund dahinter veranschaulicht das Beispiel einer kugelförmigen Domain.
Angenommen, dass in der Domain die Verteilung der Knotenpunktkoordinaten einer perfekten Gleichverteilung entspricht, sollte der Score an einem Punkt aus einem Bin im Inneren der Domain den gleichen Wert haben, wie aus einem Bin am Rand. Durch die Verwendung Bins zweidimensionaler Art und der Multiplikation der Häufigkeitswerte der verschiedenen Raumrichtungen wird dies jedoch nicht gewährleistet. Das Problem stellt Bild 4.2 grafisch dar.
Im Allgemeinen gilt bei jeder Domain, die nicht quaderförmig ist, eine nicht überall konvexe

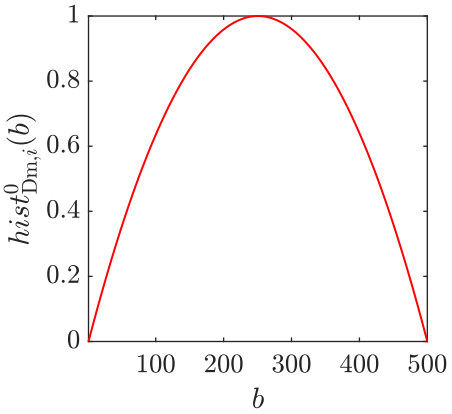


Bild 4.2: Relative Häufigkeitsverteilung $hist^0_{\mathrm{Dm},i}(b)$ am Beispiel einer kugelförmigen Domain

Oberfläche hat oder Hohlräume aufweist: Eine gleichmäßige Verteilung der Knotenpunktkoordinaten in der Simulationsdomain ist nicht gleichbedeutend mit einer Gleichverteilung im jeweiligen Histogramm. Denn je näher die Knotenpunkte zum Rand einer nicht quaderförmigen Simulationsdomain liegen, umso mehr nimmt ihre Häufigkeit ab, da umso weniger Platz für sie zur Verfügung steht. Dieser Effekt ist auch in Bild 4.2 zu sehen: Die Histogramme erwecken durch die Begrenzung auf jeweils eine Koordinatendimension den Eindruck, dass die Verteilung der Knotenpunktkoordinaten in der Simulationsdomain nicht gleichmäßig ist.
Der histogrammbasierte Score σ kann unter Punkten der Domain nur dann sinnvoll verglichen werden, wenn aus einer Gleichverteilung der Knotenpunktkoordinaten in der Simulationsdomain eine Gleichverteilung in den Histogrammen der jeweiligen Koordinatendimensionen

folgt. Um dies zu gewährleisten sollen die Histogramme $hist_i(b)$ mit einer Funktion $c_i(b)$ skaliert werden. Die Skalierung soll die Histogramme der jeweiligen Koordinatendimensionen so verzerren, dass sich im Falle einer Gleichverteilung der Knotenpunktkoordinaten in der Simulationsdomain genau eine Gleichverteilung ergibt.
Die Skalierungsfunktion $c_i(b)$ wird aus der Morphologie der Simulationsdomain hergeleitet. Die Morphologie der Domain wird dabei ebenfalls mittels Histogramme erfasst. Die Simulationsmaske hat entlang der jeweiligen Raumachsen eine durch die Diskretisierung festgelegte Größe von $s_{\mathrm{discr,x}} \times s_{\mathrm{discr,y}} \times s_{\mathrm{discr,z}}$ – wobei $[s_{\mathrm{discr},i}] = \mathrm{px}$. Diese entspricht in der Wirklichkeit den Maßen $s_\mathrm{x} \times s_\mathrm{y} \times s_\mathrm{z}$ mit $[s_i] = \mathrm{m}$. Die Histogramme $hist_{\mathrm{Dm},i}$ zur Erfassung der Morphologie der Simulationsdomain geben die Häufigkeiten der Matrixkoordinaten der Domainvoxel wieder. Für die Dimension i gilt dabei, dass die Koordinaten auf $s_{\mathrm{discr},i}$ Bins aufgeteilt werden. Die Histogramme werden anschließend so genormt, dass statt der absoluten die relativen Häufigkeiten im Histogramm $hist^0_{\mathrm{Dm},i}$ enthalten sind, sodass die maximale relative Häufigkeit 1 beträgt.
Die Koordinaten der Domain sind innerhalb der Domain selbst gleich verteilt, sodass nach der obigen Forderung, $c_i(b)$ das Histogramm $hist^0_{\mathrm{Dm},i}$ so verzerren sollte, dass sich eine Gleichverteilung ergibt. Es wird soll daher folgende Gleichung für jeden Bin b erfüllt werden:

$$1 = c_i(b) \cdot hist^0_{\mathrm{Dm},i}(b) \tag{4.2}$$

Nach einer Umstellung von Gleichung (4.2) würde sich für c_i der Term $1/hist^0_{\mathrm{Dm},i}$ ergeben. Da es $0 \leqslant hist^0_{\mathrm{Dm},i} \leqslant 1$ gilt, kann es bei Bins mit sehr niedrigen relativen Häufigkeitswerten zu sehr großen Werten kommen. In diesem Fall würde c_i eventuelles Rauschen im zu skalierenden Histogramm in einem zu großen Maß verstärken. Um dies vorzubeugen, wird die alternative Definition in Gleichung (4.3) verwendet.

$$c_i(b) := \widetilde{hist}_{\mathrm{Dm},i}(b) = \begin{cases} 1 + \log\left(\frac{1}{hist^0_{\mathrm{Dm},i}(b)}\right), & \text{sonst} \\ \max(\widetilde{hist}_{\mathrm{Dm},i}(b)), & \text{falls } hist^0_{\mathrm{Dm},i}(b) = 0 \end{cases} \tag{4.3}$$

Auf $1/hist^0_{\mathrm{Dm},i}$ wird die Logarithmusfunktion angewandt, um die Gefahr, dass zu große Werte entstehen, zu verringern. Um für $hist^0_{\mathrm{Dm},i}(b) = 1$ Gleichung (4.2) zu erfüllen wird zusätzlich 1 dazu addiert. Für den Fall, dass $hist^0_{\mathrm{Dm},i}(b) = 0$, nimmt $c_i(b)$ den maximalen Wert aller sonstiger Fälle an.
Gleichung (4.3) ergibt somit eine Skalierungsfunktion $c_i(b)$, die elementweise angewandt auf $hist_{\mathrm{Dm},i}$ im Histogrammverlauf eine Gleichverteilung annähert. Die Histogramme $hist_i(b)$ in Gleichung (4.1) werden schließlich mit der Skalierungsfunktion $c_i(b)$ elementweise skaliert:

$$hist^\mathrm{s}_i(b) = c_i(b) \cdot hist_i(b) \tag{4.4}$$

Bild 4.3 zeigt beispielhaft den Effekt von $c_i(b)$ bei einer kugelförmigen Domain. In der Mitte findet keine Skalierung statt, die Werte an den Rändern werden hochskaliert. Entsprechend

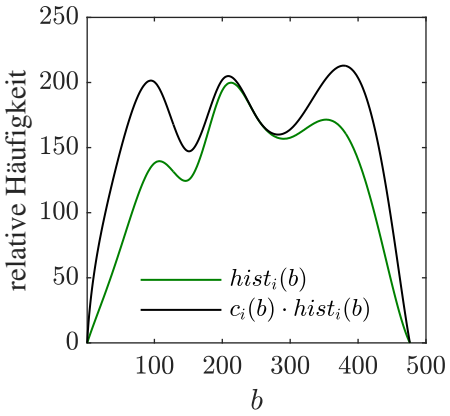


Bild 4.3: Skalierung von $hist_i(b)$ durch $c_i(b)$ bei einer kugelförmigen Domain

wird die Definition von $\sigma(\boldsymbol{x})$ angepasst auf:

$$\sigma(\boldsymbol{x}) = hist^{\mathrm{s}}_{\mathrm{x}}(bin_{\mathrm{x}}(x)) \cdot hist^{\mathrm{s}}_{\mathrm{y}}(bin_{\mathrm{y}}(y)) \cdot hist^{\mathrm{s}}_{\mathrm{z}}(bin_{\mathrm{z}}(z)) \tag{4.5}$$

Nun können Scores aus verschiedenen Punkten der Domain sinnvoll miteinander verglichen werden und der Knotenpunkt n aus N mit dem niedrigsten Score zum Wachstum kann ausgewählt werden. Damit endet Algorithmus 2.

4.1.2 Vektorbasiertes Baumwachstum

Schneider et al. benutzen für die Richtung des Wachstums am ausgewählten Knotenpunkt n den VEGF-Gradienten. Dies soll in dieser Arbeit ebenfalls ersetzt und die Wachstumsrichtung kraftbasiert ermittelt werden: Die Wachstumsrichtung soll der resultierende Richtungsvektor aus einer Addition von abstoßenden Kräften sein, die ihren Ursprung in nahen Knotenpunkten haben.
Um eine Liste nahe liegender Knotenpunkte zu erhalten, werden zunächst die Distanzen zwischen Knotenpunkt n und allen anderen Knotenpunkten berechnet. Als nahe bzw. relevante Knotenpunkte werden diejenigen Knotenpunkte angesehen, deren Distanz zu n kleiner ist, als die Länge der Eingangskante. Dadurch entfallen hauptsächlich kleine Gefäßsegmente, die weit genug liegen. Große Gefäße werden dafür auch dann berücksichtigt, wenn sie etwas weiter entfernt liegen. Bild 4.4 stellt die Auswahl relevanter Knotenpunkte dar, diese sind blau markiert. Der ausgewählte Knotenpunkt n ist in orange zu sehen und die restlichen Knotenpunkte ohne

Farbe werden für die späteren Berechnungen als nicht relevant eingestuft.
Der Betrag der Kräfte soll so gestaltet werden, dass er am größten für Kräfte ist, die aus den

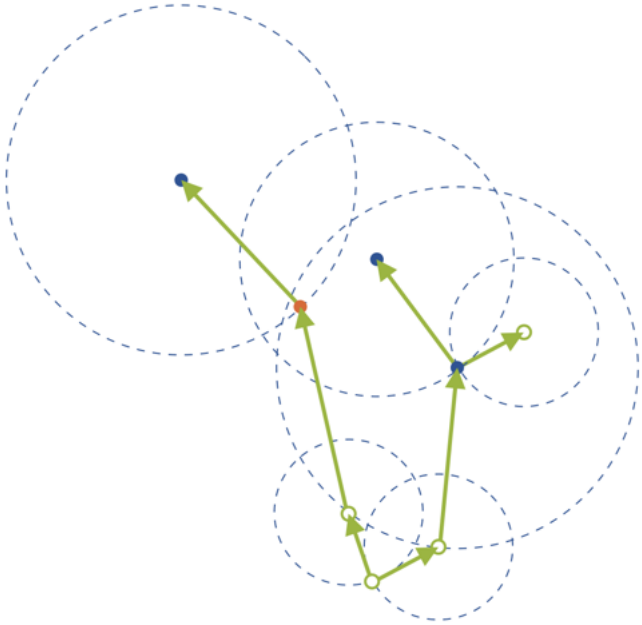


Bild 4.4: Auswahl relevanter Knotenpunkte

nächsten Knotenpunkten mit dem größten Radius und dem höchsten Score herrühren. Die Kraft $\boldsymbol{G}_i$ für einen vorhin als relevant eingestuften Knotenpunkt i setzt sich demnach wie folgt aus dem Abstand d_i, dem Radius r_i, dem Score σ_i und dem genormten Richtungsvektor

$$\boldsymbol{r}_i = \frac{\boldsymbol{p}_n - \boldsymbol{p}_i}{\|\boldsymbol{p}_n - \boldsymbol{p}_i\|} \tag{4.6}$$

zwischen Knotenpunkt n und i zusammen:

$$\boldsymbol{G}_i = (\frac{1}{d_i^0} + r_i^0 + \sigma_i^0)\boldsymbol{r}_i \tag{4.7}$$

Die Größen $1/d_i^0$, r_i^0 und σ_i^0 sind relative Größen, ihr Maximum ist unter der relevanten Knotenpunkte $i \in N_{\text{rel}}$ jeweils auf 1 normiert, sodass $\max_{i \in N_{\text{rel}}}(1/d_i^0) = 1$ usw. gilt. Die resultierende Kraft $\boldsymbol{G}_{\text{res}}$ ergibt sich aus der Summe aller Kräfte $\boldsymbol{G}_i$ für die relevanten Knotenpunkte $i \in N_{\text{rel}}$:

$$\boldsymbol{G}_{\text{res}} = \sum_{i \in N_{\text{rel}}} \boldsymbol{G}_i \tag{4.8}$$

Bild 4.5 zeigt anhand des Beispiels aus Bild 4.4 die Ermittlung von G_{res}.
Aus der normierten $\boldsymbol{G}_{\text{res}}$ ergibt sich schließlich die Wachstumrichtung im Knotenpunkt n:

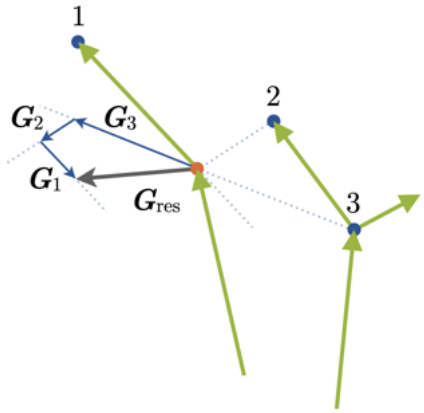


Bild 4.5: Beispiel für die Ermittlung von G_{res}

$$\boldsymbol{g} = \frac{\boldsymbol{G}_{\text{res}}}{\|\boldsymbol{G}_{\text{res}}\|} \tag{4.9}$$

4.1.3 Verhalten am Domainrand

Das Wachstumsverhalten am Domainrand muss ebenfalls angepasst werden. In der Arbeit von Schneider et al. Werden Gefäße, die an den Domainrand herangewachsen sind und darin enden, immer weiter geschrumpft, sodass sie am Ende des Wachstumsprozesses zu den Kapillaren werden, die den Rand des zu pefundierenden Domainvolumens versorgen. Dieses Verhalten führt zu Gefäßbaumstrukturen, die Bäumen aus der Natur ähnlich sind. Auf lokaler Ebene ist eine solche Struktur der tatsächlichen ähnlich, bei Berücksichtigung der kompletten Gefäßradienskala – wie in dieser Arbeit – ist jedoch eine globale Sichtweise erwünscht. Global gesehen sind Gefäßbaume bspw. kompletter Organe nicht einem Baum aus der Natur ähnlich. Insbesondere im Falle des Herzens verlaufen die großen Gefäße zunächst am Rand und verzweigen ins Innere.
Um dies in der Simulationsdomain nachzuahmen, sollen die Gefäße am Domainrand nicht gestoppt, sondern abgelenkt werden, sodass sie transversal zum Domainrand wachsen bevor sie sich ins Domaininnere verzweigen. Ähnlich zur Ermittlung der bevorzugten Wachstumsrichtung, soll hierfür ein kraftbasierter Ansatz verwendet werden. Die Abstoßungskräfte sollen am Domainrand möglichst groß, im Domaininneren möglichst klein sein. Aus der resultierenden Kraft ergibt sich schließlich ein Richtungsvektor für die Ablenkung am Rand, der mit dem Richtungsvektor der bevorzugten Wachstumsrichtung verrechnet werden kann.

Die Kräfte sollen ihren Ursprung in Punkten des Domainrandes haben. Hierzu wird eine bestimmte Anzahl an Punkten aus dem Domainrand gesamplet. Der Domainrand wird dabei mittels Subtraktion einer erodierten Domain von der ursprünglichen Domain erhalten. Das Erodieren findet dazu mit einem kugelförmigen Kernel mit einem Radius von einem Voxel statt. Der Domainrand hat somit eine durchschnittliche Dicke von einem Voxel. Gesamplet werden 20 % der Randvoxel, dieser Anteil reicht aus, um die Domain so zu repräsentieren, dass die Morphologie nicht zu sehr vereinfacht wird. Die Punktewolke der Randpunkte der verwendeten Simulationsdomain ist in Bild 4.6 zu sehen, die herzförmige Domain ist klar zu erkennen. Weist

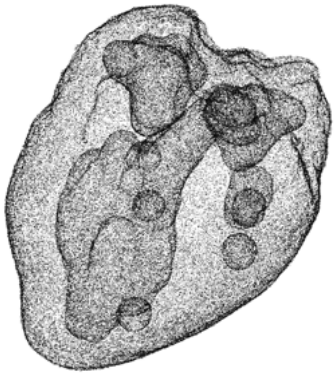


Bild 4.6: Punktewolke der Simulationsdomainrandpunkte

die Domain auch feinere Strukturen auf, kann der Prozentsatz erhöht werden, dies ist jedoch auch mit einer Erhöhung des Rechenaufwands verbunden. Im Weiteren wird die Menge der gesampleten Randpunkte mit E bezeichnet.
Für die Berechnung der resultierenden Ablenkungskraft $\boldsymbol{D}$ im Knotenpunkt n wird zunächst aus E die relevante Teilmenge E_{rel} bestimmt. Diese enthält die Punkte aus E, die sich zum Knotenpunkt n in einem kleineren eukledischen Abstand als $\theta_{\mathrm{D,dist}}$ befinden. Der Schwellwert $\theta_{\mathrm{D,dist}}$ hängt von den Abständen $d_{\min}$ und $d_{\max}$ des nächst- bzw. weitestgelegenen Punktes aus E ab:

$$\theta_{\mathrm{D,dist}} = d_{\min} + 0{,}2 \cdot (d_{\max} - d_{\min}) \tag{4.10}$$

Somit wird die Ablenkungskraft $\boldsymbol{D}_i$ für einen relevanten Randpunkt i aus E_{rel} unter Gewichtung des genormten Richtungsvektors

$$\boldsymbol{r}_i = \frac{\boldsymbol{p}_i - \boldsymbol{p}_n}{\|\boldsymbol{p}_i - \boldsymbol{p}_n\|} \tag{4.11}$$

mit den errechneten Abständen d_i wie folgt berechnet:

$$\boldsymbol{D}_i = \frac{1}{d_i}\boldsymbol{r}_i \tag{4.12}$$

Nun werden die Teilkräfte $\boldsymbol{D}_i$ analog zu Gleichung (4.8) bzw. Bild 4.5 vektoriell addiert und die resultierende Ablenkungskraft $\boldsymbol{D}$ ergibt sich zu:

$$\boldsymbol{D}_{\text{res}} = \sum_{i \in E_{\text{rel}}} \boldsymbol{D}_i \tag{4.13}$$

Wird die resultierende Ablenkungskraft $\boldsymbol{D}_{\text{res}}$ normiert, ist sie gleich[1] einer normierten Normalen, die senkrecht auf den Domainrand steht und ins Domaininnere zeigt.

$$\boldsymbol{n}_\Delta = \frac{\boldsymbol{D}_{\text{res}}}{\|\boldsymbol{D}_{\text{res}}\|} \tag{4.14}$$

Der Richtungsvektor $\boldsymbol{d}$, der später mit $\boldsymbol{g}$ verrechnet werden soll, sollte jedoch mit $\boldsymbol{n}_\Delta$ nicht koinzidieren. Wären die beiden parallel, würde das zu zu großen Ablenkungswinkeln beim Wachstum neuer Segmente in Domainrandnähe führen. Besser eignet sich ein $\boldsymbol{d}$, der senkrecht auf $\boldsymbol{n}_\Delta$ steht. Allerdings gibt es von solchen Vektoren unendlich viele. Es soll derjenige ausgesucht werden, der in der vom Blattsegment und $\boldsymbol{g}$ aufgespannten Ebene Γ liegt. Diese Einschränkung bedeutet, dass $\boldsymbol{d}$ in der Schnittgeraden der Ebene Γ und der durch den Normalenvektor $\boldsymbol{n}_\Delta$ aufgespannten Ebene Δ liegt. In Bild 4.7 sind die beiden Ebenen Γ und Δ sowie die zur Berechnung von $\boldsymbol{d}$ verwendeten Vektoren. Die Richtung der Schnittgeraden

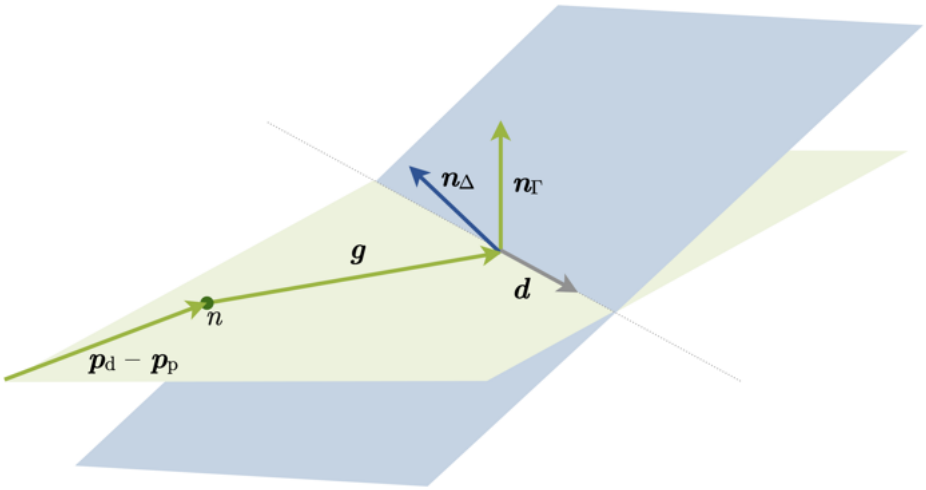


Bild 4.7: Lage von $\boldsymbol{d}$

kann aus den Normalvektoren der beiden Ebenen berechnet werden. Dazu wird zunächst der

[1] Gilt nicht exakt da die Menge E lediglich eine Teilmenge des Domainrands darstellt. Mit $E_{\text{rel}} \subset E$ trifft dies auch auf E_{rel} selbst zu.

Normalenvektor $\boldsymbol{n}_\Gamma$ bestimmt:

$$\boldsymbol{n}_\Gamma = \frac{(\boldsymbol{p}_d - \boldsymbol{p}_p) \times \boldsymbol{g}}{\|(\boldsymbol{p}_b - \boldsymbol{p}_p) \times \boldsymbol{g}\|} \tag{4.15}$$

Dabei stehen die Vektoren $\boldsymbol{p}_d$ und $\boldsymbol{p}_p$ für die Ortsvektoren der distalen bzw. proximalen Knotenpunkte des Blattsegments. Der Richtungsvektor der Schnittgeraden $\boldsymbol{r}_{\Gamma\Delta}$ ergibt sich aus den Normalenvektoren dann zu:

$$\boldsymbol{r}_{\Gamma\Delta} = \frac{\boldsymbol{n}_\Gamma \times \boldsymbol{n}_\Delta}{\|\boldsymbol{n}_\Gamma \times \boldsymbol{n}_\Delta\|} \tag{4.16}$$

Das heißt, dass somit die Anzahl der möglichen $\boldsymbol{d}$ auf die beiden Vektoren $\boldsymbol{r}_{\Gamma\Delta}$ und $-\boldsymbol{r}_{\Gamma\Delta}$ beschränkt wurde. Von den beiden wird derjenige als $\boldsymbol{d}$ definiert, der mit $\boldsymbol{g}$ den kleineren Winkel einschließt.
In [Sch+12] verrechnen Schneider et al. den VEGF-Gradienten mit dem normierten Richtungsvektor $(\boldsymbol{p}_d - \boldsymbol{p}_p)/\|\boldsymbol{p}_d - \boldsymbol{p}_p\|$ des Blattsegments, um die letztendliche Wachstumsrichtung $\tilde{\boldsymbol{g}}$ zu erhalten. Ähnlich dazu soll $\tilde{\boldsymbol{g}}$ auch in dieser Arbeit berechnet werden, jedoch wird statt des VEGF-Gradienten der Vektor $\boldsymbol{g}$ benutzt und zusätzlich wird auch die Ablenkungsrichtung $\boldsymbol{d}$ dazugerechnet. Insgesamt wird $\tilde{\boldsymbol{g}}$ folgenderweise definiert:

$$\tilde{\boldsymbol{g}}' = (1-\delta)\left(\boldsymbol{g} + \lambda\frac{\boldsymbol{p}_d - \boldsymbol{p}_p}{\|\boldsymbol{p}_d - \boldsymbol{p}_p\|}\right) + \delta\boldsymbol{d} \tag{4.17}$$

$$\tilde{\boldsymbol{g}} = \frac{\tilde{\boldsymbol{g}}'}{\|\tilde{\boldsymbol{g}}'\|} \tag{4.18}$$

Der Faktor λ wird von Schneider et al. gleich Eins gesetzt. Der Faktor δ soll mit kleiner werdendem Abstand des Knotenpunkts zum Domainrand exponentiell zunehmen. Der exponentielle Zusammenhang begründet sich darin, dass die Ablenkung durch den Domainrand erst möglichst nahe dem Rand einsetzen soll. Der Abstand zum Domainrand berechnet sich nicht trivial, da es sich um einen Abstand zwischen einem Knotenpunkt und einer Punktewolke bestehend aus E handelt. Zudem soll der Abstand ein relatives und kein absolutes Maß sein. Ausgangspunkt ist der Abstandsvektor $\boldsymbol{d}_{\min}$, der den Knotenpunkt n und den nächstgelegenen Domainrandpunkt verbindet und dessen Norm $d_{\min}$ beträgt. Der Abstandsvektor gibt einen diskreten Abstand wieder. Die Beträge der einzelnen Komponenten $j \in \{x, y, z\}$ werden daher jeweils auf die diskrete Kantenlänge $s_{\mathrm{discr},j}$ der Domain bezogen. Aus den relativen Komponenten wird durch gewichtetes Mitteln mittels der Beträge der Komponenten von $\boldsymbol{d}_{\min}$ das relative Abstandsmaß

$0 \leqslant \tilde{d} \leqslant 1$ errechnet. Zusammengefasst wird $\tilde{d}$ mit $\boldsymbol{d}_{\min} = (d_{\min,\mathrm{x}}, d_{\min,\mathrm{y}}, d_{\min,\mathrm{z}})^T$ definiert als:

$$\tilde{d} = \begin{pmatrix} \frac{|d_{\min,\mathrm{x}}|}{s_{\mathrm{discr,x}}} \\ \frac{|d_{\min,\mathrm{y}}|}{s_{\mathrm{discr,y}}} \\ \frac{|d_{\min,\mathrm{z}}|}{s_{\mathrm{discr,z}}} \end{pmatrix} \cdot \frac{\boldsymbol{d}_{\min}}{\sum_j |d_{\min,\mathrm{j}}|} \tag{4.19}$$

Die exponentielle Abhängigkeit zwischen dem Faktor δ und dem relativen Abstandsmaß $\tilde{d}$ ist in Bild 4.8 veranschaulicht und wird schließlich wie folgt definiert:

$$\delta(\tilde{d}) = e^{-10 \cdot \tilde{d}} \tag{4.20}$$

Somit ist die Wachstumsrichtung $\tilde{\boldsymbol{g}}$ mit dem Ablenkungsvektor $\boldsymbol{d}$ identisch für den Fall, dass

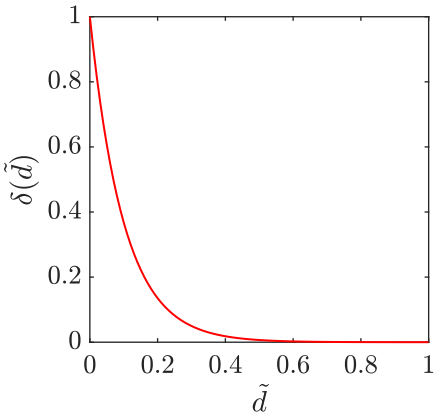


Bild 4.8: Exponentielle Abhängigkeit vom Faktor δ und dem relativen Abstandsmaß $\tilde{d}$

sich der Knotenpunkt n im Domainrand befindet. Ab etwa $\tilde{d} = 0{,}5$ beträgt δ nur noch $\leqslant 0{,}0067$, sodass der Einfluss von $\boldsymbol{d}$ in Gleichung (4.17) marginal wird.
Das Verhalten am Domainrand beinhaltet zudem – neben der Ablenkung – auch das Kürzen von Gefäßen, sollte ihr neuer distaler Endknotenpunkt außerhalb von der Domain liegen. Insbesondere tritt das bei Bifurkationen in Domainrandnähe auf, da hier nur die Hauptwachstumsrichtung, das heißt die Wachstumsrichtung der größeren Abzweigung abgelenkt wird. Die Gefäße werden in diesen Fällen so gekürzt, dass der neue distale Endpunkt im Domainrand liegt. In Verbindung mit dem Verhalten am Domainrand müssen auch bei der Remodellierung Veränderungen am Vorgehen von Schneider et al. getroffen werden. Bei jedem Remodellierungsschritt wird der Baum auf zwei zusätzliche Aspekte überprüft. Der eine Aspekt betrifft

die Fälle, bei denen ein Gefäß nach Kürzen im Domainrand endet. Wird ein solches Gefäß zum weiteren Wachstum ausgewählt, haben neue Segmente in jedem Fall die Länge Null. Solche Gefäße werden während der Remodellierung aus dem Gefäßbaum entfernt.
Durch den Einfluss der Ablenkungskraft kann es zudem in Einzelfällen zu zu großen Winkeln zwischen benachbarten Segmenten kommen. Werden solche Segmentpaare gefunden, wird das distale Segment während der Remodellierung ebenfalls aus dem Gefäßbaum gelöscht.

4.1.4 Baumgenerierung und Degenerierung

In der Arbeit von Schneider et al. findet das Wachstum auf einer Auflösungsstufe s so lange statt, bis in einem bestimmten Mindestanteil der Voxeln eine Mindestkonzentration des VEGFs erreicht wird. In dieser Arbeit wird diese Bedingung durch eine andere, auf dem in Gleichung (4.5) definierten Score basierende Bedingung ersetzt. Die Bedingung von Schneider et al. betrifft zwei Aspekte des Wachstums: Es sollte erst in die nächste Auflösungsstufe gewechselt werden, wenn die Domain möglichst gleichmäßig und in einem genügenden Maß perfundiert ist.
Der Ansatz in dieser Arbeit ist somit, diese zwei Aspekte histogrammbasiert zu betrachten. Dazu werden die Scores und ihre Verteilung in der Domain betrachtet. Es wird in die nächste Auflösungsstufe gewechselt, wenn die Scores in einem bestimmten Mindestanteil der Domain eine bestimmte Schwelle erreicht haben. Da der Score sich aus dem Produkt der Häufigkeiten der drei Raumrichtungen zusammensetzt, wird auch die Schwelle θ_σ in der entsprechenden Größenordnung festgelegt. Eine analytische Herleitung eines passenden Wertes für θ_σ gestaltet sich sehr komplex, denn es steht lediglich die Angabe aus Abschnitt 3.1.1 zur Verfügung, dass die Dichte der Gefäße im Querschnitt etwa 1000 Gefäße$/\mathrm{mm}^2$ beträgt. Diese Angabe bezieht sich auf die im Querschnitt geschnittenen Gefäße und ist hinsichtlich der Tatsache, dass ein Gefäß in einem Querschnitt auch öfter vorkommen kann sehr ungenau. Außerdem ist es nicht trivial, einen Zusammenhang zwischen der Anzahl von Knotenpunkte in einem Querschnitt und der Gefäßdichte analytisch herzustellen. Aus diesem Grund wird auf eine analytische Herleitung von θ_σ verzichtet und der Wert empirisch auf $2 \cdot 10^6$ festgelegt.
Für ein 2D-Bin bedeutet der Schwellwert $\theta_\sigma = 2 \cdot 10^6$, dass dieser im Durchschnitt mindestens $\sqrt[3]{2 \cdot 10^6}$ Knotenpunkte enthalten soll. Die Scorebeiträge im Produkt aus Gleichung (4.5) werden für jede Raumrichtung, für jedes 2D-Bin betrachtet. Die erreichten Scorebeiträge $hist_i^s(b)$ in den 2D-Bins werden auf den Soll-Wert θ_σ bezogen, sodass sich für jedes Bin ein Maß aus $[0, 1]$ ergibt. Der Ausdruck $hist_i^s(b)$ steht hierbei für die skalierte Histogrammfunktion aus Abschnitt 4.1.1 nach Gleichung (4.4).
Für jede Raumrichtung wird der Verlauf dieses Maßes über den Bins aufgetragen. Schließlich wird die Fläche unter der Kurve berechnet und mit der Fläche verglichen, die sich ergeben würde, wenn jedes Bin den Schwellwert erreichen würde. Wird der Schwellwert in jedem Bin

genau erreicht, beträgt die Fläche unter der Kurve genau $1 \cdot r$, wobei r die aktuelle Anzahl der 2D-Bins in einer Raumrichtung bezeichnet. Sie ergibt sich aus der aktuellen Auflösungsstufe s zu 2^s.
Es wird in die nächste Auflösungsstufe gewechselt, wenn das Mittel der Flächen der drei Raumrichtungen, bezogen auf die Soll-Fläche, den Schwellwert $\theta_\mathrm{A}(s)$ erreicht. Gleichung (4.21) und Gleichung (4.22) fassen das Vorgehen zusammen.

$$\bar{A} = \frac{A^0_\mathrm{x} + A^0_\mathrm{y} + A^0_\mathrm{z}}{3} \tag{4.21}$$

$$A^0_i = \frac{1}{r} \int_1^r \min\left(1, \frac{hist^s_i(b)}{\theta_\sigma}\right) \, \mathrm{d}b \tag{4.22}$$

Die Flächen werden mittels Integrieren berechnet und die Teilflächen oberhalb 1 werden nicht miteingerechnet. Würde man diese auch berücksichtigen, könnten sie – wie in Bild 4.9 zu erkennen – fehlende Flächenstücke unter dem bei 1 liegenden Waagerechten kompensieren.
Der Schwellwert θ_A wird von s abhängig definiert. Er soll vom initialen Wert $0{,}5$ über die

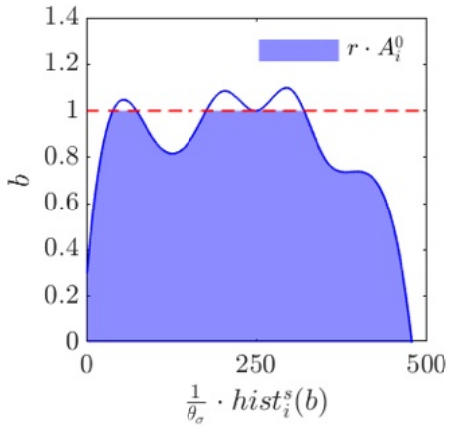


Bild 4.9: Berechnung der Flächen A^0_i

Auflüsungsstufen exponentiell auf $0{,}95$ wachsen, wie es in Bild 4.10 zu sehen ist. Für die niedrigen Auflösungsstufen sollte θ_A nicht zu hoch gewählt werden, da sonst bereits am Anfang die Gefäße dazu gezwungen sind das Volumen zu füllen, obwohl ihre Radien unter Umständen noch viel zu groß sind für die dafür verlangte Flexibilität. θ_A wird schließlich wie folgt definiert:

$$\theta_\mathrm{A}(s) = 0{,}5 + \frac{0{,}95 - 0{,}5}{e^{\frac{1}{3}S} - e^{\frac{1}{3}}} \left(e^{\frac{1}{3}s} - e^{\frac{1}{3}}\right) \tag{4.23}$$

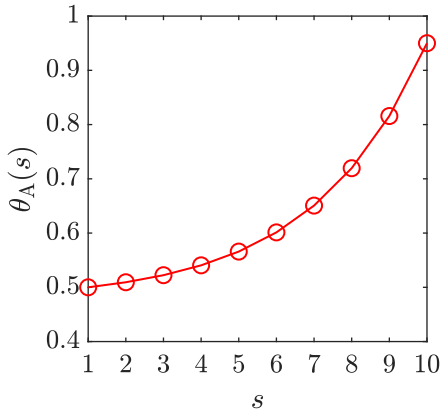


Bild 4.10: Verlauf von θ_A

Die Faktoren $\frac{1}{3}$ für die Wachstumsgeschwindigkeit wurden empirisch festgelegt, S bezeichnet die Anzahl der Auflösungsstufen s. Im Degenerationsschritt werden unter anderem Segmente aus dem Gefäßbaum gelöscht, die der Perfusion nicht bedeutend beitragen oder zu einer lokal größeren Perfusion führen, als gefordert. Schneider et al. lassen solche Segmente durch eine zu hohe VEGF-Konzentration kennzeichnen. In dieser Arbeit wird für jeden Knotenpunkt der Score σ berechnet. Diejenigen Knotenpunkte, deren Score den Schwellwert θ_σ übersteigt, werden aus dem Gefäßbaum entfernt.

4.1.5 Simulationsdomain

Aus der in Abschnitt vorgestellten Vorarbeit wird ein 3D-Volumen zur Verfügung gestellt, das etwa dem Volumen des Herzens ohne die Kammer und Vorhöfe entspricht. Es ist lediglich eine Annäherung. Für die synthetische Datenherstellung bzw. hinsichtlich der Segmentierungsaufgabe ist die Genauigkeit jedoch ausreichend. In Bild 4.6 wurde bereits in Form einer Punktewolke die Simulationsdomain gezeigt, in Bild 4.11 ist sie auch als Volumen zu sehen. Ein Teil ist entfernt um die inneren Hohlräume, die grob die Kammer und Vorhöfe darstellen, erkennen zu können.

4.1.6 Datenrepräsentation und -darstellung

Die synthetische Datenherstellung findet in MATLAB statt. Der Gefäßbaum wird als Graph repräsentiert, sodass kein Volumen sondern lediglich die Koordinaten der Knotenpunkte, die Anfangs- und Endknotenpunkte jeder Kante, der dazugehörige Radius und die originalen sowie diskreten Größen s_i bzw. $s_{\mathrm{discr},i}$ der Domain abgespeichert werden.
Für die Verfolgung der synthetischen Datenherstellung ist es unentbehrlich den generierten

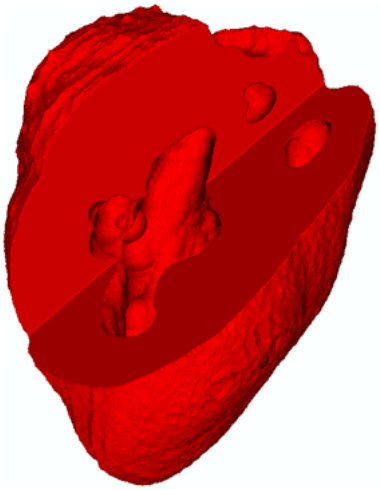


Bild 4.11: Die Simulationsdomain

Gefäßbaum auch grafisch darstellen zu können. Eine Darstellung, bei der die komplette Radienskala des Gefäßbaumes aufgelöst wird, ist nach Abschnitt 4.1 nicht trivial. Die Darstellung muss erfolgen, ohne dass es einen Zwischenschritt gibt, bei dem ein Volumen in Form einer 3D-Matrix erstellt wird.
Die Umgebung *Processing* bietet sich zu grafischen Darstellungszwecken an. Es wird die Library *PeasyCam* von Jonathan Feinberg benutzt. Die Segmente werden als Zylinder bzw. genauer genommen als n-eckige Prismen dargestellt. Der Parameter n kann dabei soweit reduziert werden, dass die Darstellung schnell genug auf Benutzereingaben reagiert. Zwischen zwei Segmenten befindet sich eine Kugel, die die Lücken in der Darstellung verhindert, die entstehen würden, wenn zwei Zylinder nicht koaxial zueinander liegen würden. *Processing* erlaubt bei der Veranschaulichung die Vergrößerung, Rotation und Translation des dargestellten 3D-Körpers. Bild 4.12 und Bild 4.13 zeigen eine Übersicht und einen näheren Ausschnitt des generierten Gefäßbaumes.

4.2 Sampling

Beim Sampling sollen würfelförmige Volumina erstellt werden, die einen Ausschnitt des Gefäßbaumes darstellen und sich für das spätere neuronale Netz als Input eignen. Aus vorher bereits erläuterten Gründen soll auch bei diesem Schritt vermieden werden, dass eine 3D-Matrix erstellt wird, die den ganzen Gefäßbaum auflöst. Stattdessen soll für ein beliebiges Sample überprüft werden, welche Gefäßstücke es durchqueren, sodass nur diese als Volumen gerendert werden müssen. Die nächsten drei Abschnitte beschäftigen sich damit, wie die Grenzen und

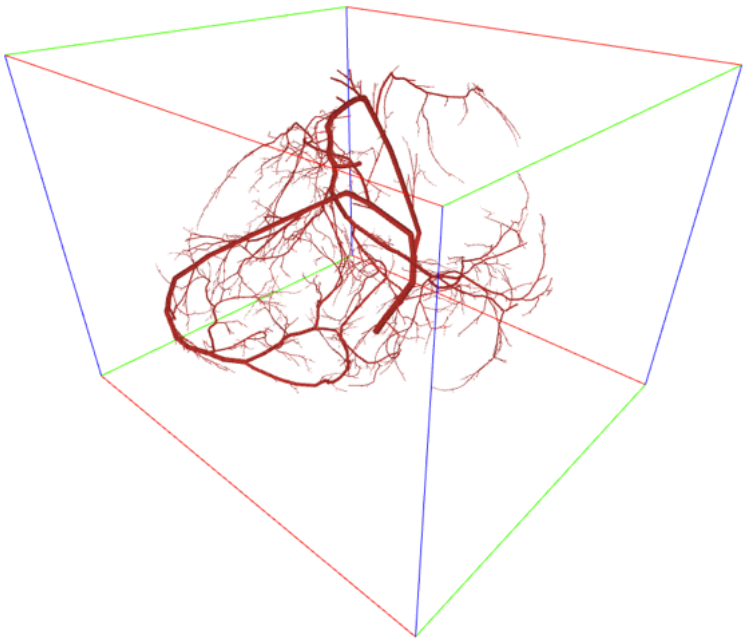


Bild 4.12: Synthetischer Gefäßbaum

Maße für die Subvolumina ausgewählt werden und wie aus dem Graph ein 3D-Volumen hergestellt wird.
Zum gerenderten 3D-Volumen wird Rauschen verschiedener Arten hinzugefügt, sodass die Samples dem in Abschnitt 3.1.1 beschriebenen Datensatz möglichst ähnlich werden. Dieser Prozess wird im Anschluss beschrieben. Zum Schluss werden die verschiedenen Datensets erläutert, die durch das vorgestellte Samplingsverfahren erstellt werden.

4.2.1 Definition der Subvolumina

Für das Sampling muss zuvor lediglich die diskerete Kantenlänge $\varepsilon_{\text{discr}}$ festgelegt werden. Im folgenden ersten Schritt des Samplingverfahrens werden für das Sample S der Mittelpunkt $\boldsymbol{c}$ des Ausschnittes, die Kantenlänge ε und die Auflösung ρ festgelegt. Der Mittelpunkt des Samplingsubvolumens wird zufällig ausgewählt. Die Samples sollen jedoch grundsätzlich aus der Nähe von Gefäßen stammen, sodass es keine Samples geben soll, die ausschließlich Hintergrund enthalten. Der Grund dafür ist, dass sich der Mittelpunkt des Wurmsichtfeldes ebenfalls in Gefäßnähe bzw. im Idealfall auf der Gefäßmittellinie befindet. Ein anderer Grund ist zudem, dass das neuronale Netz aus einem Sample mit nur Hintergrund keine Zusammenhänge

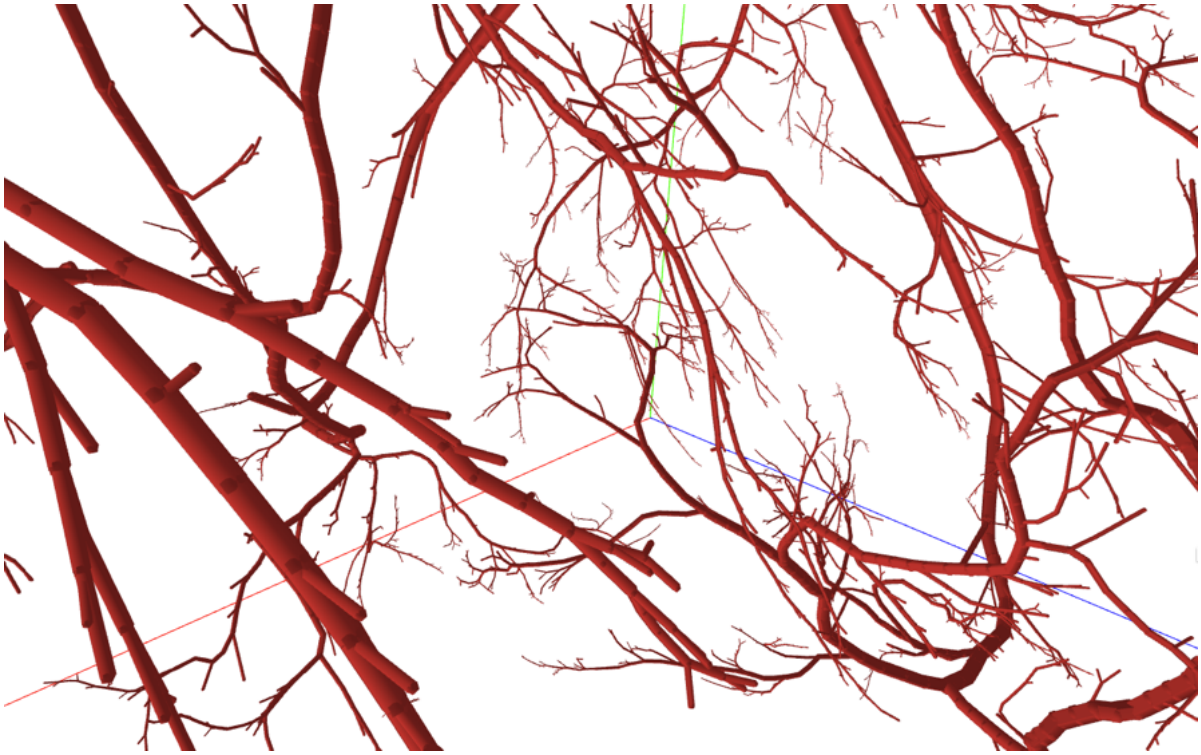


Bild 4.13: Synthetischer Gefäßbaum (Ausschnitt)

ableiten bzw. erlernen kann. Es soll daher für jedes Sample ein Segment aus dem Gefäßbaum ausgewählt werden und der Ausschnitt in einer bestimmten Nähe und Position bezüglich dieses Segments festgelegt werden.
Zunächst wird dazu ein Knotenpunkt, der kein Wurzelknotenpunkt ist, zufällig ausgewählt. Für den weiteren Verlauf wird das Segment genommen, das in dem ausgewählten Knotenpunkt endet. Als nächstes wird die Kantenlänge ε des Samples in Meter festgelegt. Diese soll in der Größenordnung des Radius r des ausgewählten Segments liegen, sodass das ausgewählte Segment zum einen aufgelöst werden kann, zum anderen nicht komplett das Subvolumen ausfüllt. Damit auch die Umgebung vom Segment im Sample enthalten wird, wird die Beschränkung $4r \leqslant \varepsilon$ getroffen. Um die Auflösung des ausgewählten Segments nicht zu weit zu reduzieren, wird zudem $\varepsilon \leqslant 10r$ gefordert. Die Kantenlänge ε wird schließlich durch

$$\varepsilon = \alpha r \tag{4.24}$$

festgelegt, wobei α zufällig aus dem Intervall $[4; 10]$ gewählt wird. Dadurch, dass nun sowohl die Kantenlänge ε in Weltkoordinaten, als auch die diskrete Kantenlänge $\varepsilon_{\text{discr}}$ bekannt sind,

kann die Auflösung ρ berechnet werden. Diese ergibt sich zu:

$$\rho = \frac{\varepsilon}{\varepsilon_{\text{discr}}} \tag{4.25}$$

Bevor das Sample gerendert werden kann muss nur noch der Mittelpunkt $\boldsymbol{c}$ bestimmt werden. Dieser wird wie folgt definiert:

$$\boldsymbol{c} = \boldsymbol{p}_{\text{p}} + \tilde{l}(\boldsymbol{p}_{\text{d}} - \boldsymbol{p}_{\text{p}}) + \tilde{r}\tilde{\boldsymbol{n}} \tag{4.26}$$

Die Vektoren $\boldsymbol{p}_{\text{d}}$ und $\boldsymbol{p}_{\text{p}}$ bezeichnen den Ortsvektor des distalen bzw. des proximalen Ende des ausgewählten Segments. Die Größen $\tilde{l}$ und $\tilde{r}$ werden zufällig aus dem Intervall $[0; 1]$ bzw. $\left[0; \frac{3}{8}\varepsilon\right]$ gesamplet. Der Vektor $\tilde{\boldsymbol{n}}$ ist ein zufällig aus der zum Vektor $\boldsymbol{p}_{\text{d}} - \boldsymbol{p}_{\text{p}}$ orthogonal liegenden Ebene gewählt. Nach Gleichung (4.26) liegt der Mittelpunkt, bestimmt durch $\boldsymbol{c}$, zufällig auf einer Kreisbahn um das ausgewählte Segment mit einem Radius von $\tilde{r}$. Die durch die Kreisbahn aufgespannte Ebene teilt das ausgewählte Segment im Verhältnis $\tilde{l} : 1 - \tilde{l}$ in ein proximales und distales Stück auf, wie Bild 4.14 dies auch verdeutlicht.

Die Intervalle für $\tilde{l}$ und $\tilde{r}$ resultieren aus folgender Überlegung: Dadurch, dass ein Segment

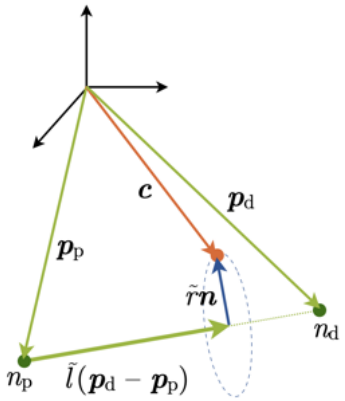


Bild 4.14: Berechnung von c

zufällig aus dem Gefäßbaum ausgewählt wird und $\boldsymbol{c}$ sich durch $\tilde{l} \in [0; 1]$ zwischen seinen Endpunkten befindet, wird der komplette Gefäßbaum abgedeckt, der Mittelpunkt kann sich überall entlang der Gefäße befinden. Für $\tilde{r}$ wurde die obere Grenze so ausgelegt, dass der Abstand der Segmentmittellinie vom Rand des Samples in $\boldsymbol{c}$, das heißt $\frac{1}{2}\varepsilon - \frac{3}{8}\varepsilon = \frac{1}{8}\varepsilon$ minimal $\frac{r}{2}$ beträgt, denn es gelten mit $4r \leqslant \varepsilon \leqslant 10r$ die Relationen $\frac{1}{8}4r \leqslant \frac{1}{8}\varepsilon \leqslant \frac{1}{8}10r \Leftrightarrow \frac{1}{2}r \leqslant \frac{1}{8}\varepsilon \leqslant \frac{5}{4}r$.

4.2.2 Ermittlung des Subgraphen

Im nächsten Schritt sollen die Knotenpunkte bzw. Kanten des Gefäßbaumgraphen gefunden werden, die im – durch den Mittelpunkt $\boldsymbol{c}$ und die Kantenlänge ε definierten – Ausschnitt enthalten sind. Das Vorgehen fasst Algorithmus 3 zusammen.
Für einen späteren Schritt, der erfolgt, nachdem das Volumen mit Rauschen versetzt wurde, ist es notwendig, mit einem mit der halben Kantenlänge $\frac{\varepsilon_{\text{discr}}}{2}$ gepaddeten Volumen zu rechnen. Die Kantenlänge ε wird somit zunächst auf $\varepsilon_{\text{p}} = 2\varepsilon$ erweitert. Als nächstes werden die unteren sowie oberen Volumengrenzen $\boldsymbol{b}_{\text{l}}$ bzw. $\boldsymbol{b}_{\text{u}}$ bestimmt. Das zu erstellende Volumen ist würfelförmig. Jedoch ist es rechentechnisch günstiger nicht alle Kanten auf einen Schnittpunkt mit einem Würfel, sondern stattdessen zunächst mit einer Kugel zu untersuchen. Aus diesem Grund wird zunächst um den Volumenwürfel die Umkugel gebildet und die Segmente auf einen Schnittpunkt mit dieser untersucht. Die Umkugel hat dabei den Radius $r_{\text{shell}} = \frac{\sqrt{\varepsilon_{\text{p}}}}{2}$.
Die Segmente werden zum Finden der Schnittpunkte als Geraden betrachtet. In Form einer Geradengleichung kann das Segment s folgender Weise aufgeschrieben werden:

$$s : \boldsymbol{x} = \boldsymbol{p}_{\text{p}} + \lambda \cdot (\boldsymbol{p}_{\text{d}} - \boldsymbol{p}_{\text{p}}) \tag{4.27}$$

In Zeile 7 des Algorithmus wird die Formel für die Berechnung des Abstandes einer Geraden von einem beliebigen Punkt benutzt. In diesem Fall wird der Abstand d zwischen s und $\boldsymbol{c}$ berechnet. Hat die Gerade einen Schnittpunkt mit der Umkugel, muss für den Abstand d, den Radius $r(e)$ der Kante e und den Umkugelradius r_{shell} der Zusammenhang $d - r(e) \leqslant r_{\text{shell}}$ gelten. Die Bedingung verdeutlicht Bild 4.15. Segmente, für die diese Bedingung zutrifft,

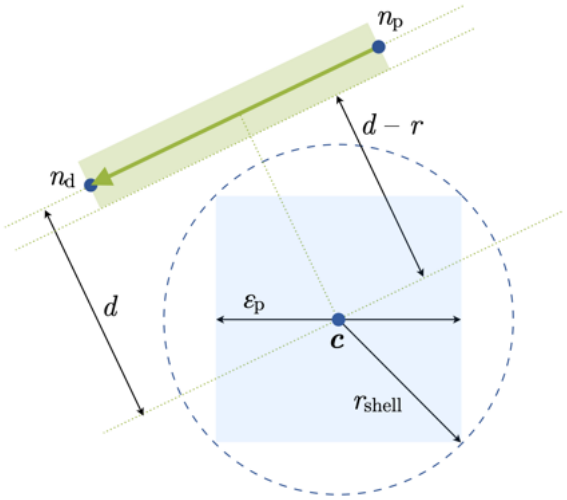


Bild 4.15: Lage von Gefäßsegmenten und der Umkugel

Algorithmus 3: Ermittlung des Subgraphen

Input: Gefäßbaumgraph G
Mittelpunkt $\boldsymbol{c}$
Kantenlänge ε
Auflösung ρ
Output: Subgraph G_{sub}

$\varepsilon_{\mathrm{p}} \leftarrow 2\varepsilon$;
$b_{\mathrm{l}} \leftarrow \boldsymbol{c} - \frac{\varepsilon_{\mathrm{p}}}{2} \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^{\mathrm{T}}$;
$b_{\mathrm{u}} \leftarrow \boldsymbol{c} + \frac{\varepsilon_{\mathrm{p}}}{2} \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^{\mathrm{T}}$;
$r_{\mathrm{shell}} \leftarrow \frac{\sqrt{\varepsilon_{\mathrm{p}}}}{2}$;
$E_1 \leftarrow \emptyset$;
foreach *edge* e *in* G **do**
  $d(e) \leftarrow \frac{\|(\boldsymbol{c}-\boldsymbol{p}_{\mathrm{p}}(e))\times(\boldsymbol{p}_{\mathrm{d}}(e)-\boldsymbol{p}_{\mathrm{p}}(e))\|}{\|\boldsymbol{p}_{\mathrm{d}}(e)-\boldsymbol{p}_{\mathrm{p}}(e)\|}$;
  if $(d - r(e) \leqslant r_{\mathrm{shell}})$ & $(2r(e) > \rho)$ **then**
    E_1.add(e);
  end
end
$G_{\mathrm{sub}} \leftarrow$ extractSubGraph(G,E_1);
$E_2 \leftarrow \emptyset$;
foreach *edge* e *in* G_{sub} **do**
  $\boldsymbol{r} = \boldsymbol{p}_{\mathrm{d}}(e) - \boldsymbol{p}_{\mathrm{p}}(e)$;
  $f_{\mathrm{p}}, \lambda_{\mathrm{p}} \leftarrow$ rayBoxIntersection($\boldsymbol{p}_{\mathrm{p}}(e), \boldsymbol{r}, \boldsymbol{b}_{\mathrm{l}}, \boldsymbol{b}_{\mathrm{u}}$);
  if $(f_{\mathrm{p}} = 1)$ & $(0 \leqslant \lambda_{\mathrm{p}} \leqslant 1)$ **then**
    $\boldsymbol{p}_{\mathrm{p}} \leftarrow \boldsymbol{p}_{\mathrm{p}} + \lambda_{\mathrm{p}}\boldsymbol{r}$;
    if $t(n_{\mathrm{p}}(e)) = 2$ **then**
      $t(n_{\mathrm{p}}(e)) \leftarrow 1$;
    end
  end
  $f_{\mathrm{d}}, \lambda_{\mathrm{d}} \leftarrow$ rayBoxIntersection($\boldsymbol{p}_{\mathrm{d}}(e), -\boldsymbol{r}, \boldsymbol{b}_{\mathrm{l}}, \boldsymbol{b}_{\mathrm{u}}$);
  if $(f_{\mathrm{d}} = 1)$ & $(0 \leqslant \lambda_{\mathrm{d}} \leqslant 1)$ **then**
    $\boldsymbol{p}_{\mathrm{d}} \leftarrow \boldsymbol{p}_{\mathrm{d}} - \lambda_{\mathrm{d}}\boldsymbol{r}$;
    if $t(n_{\mathrm{d}}(e)) = 2$ **then**
      $t(n_{\mathrm{d}}(e)) \leftarrow 1$;
    end
  end
  $c_{\boldsymbol{p}_{\mathrm{p}}} = (\boldsymbol{p}_{\mathrm{p}} < \boldsymbol{b}_{\mathrm{l}} \mid \boldsymbol{p}_{\mathrm{p}} > \boldsymbol{b}_{\mathrm{u}})$;
  $c_{\boldsymbol{p}_{\mathrm{d}}} = (\boldsymbol{p}_{\mathrm{d}} < \boldsymbol{b}_{\mathrm{l}} \mid \boldsymbol{p}_{\mathrm{d}} > \boldsymbol{b}_{\mathrm{u}})$;
  $c_{\lambda_{\mathrm{p}}} = (\lambda_{\mathrm{p}} < 0 \mid \lambda_{\mathrm{p}} > 1)$;
  $c_{\lambda_{\mathrm{d}}} = (\lambda_{\mathrm{d}} < 0 \mid \lambda_{\mathrm{d}} > 1)$;
  if $c_{\boldsymbol{p}_{\mathrm{p}}}$ & $c_{\boldsymbol{p}_{\mathrm{d}}}$ & $c_{\lambda_{\mathrm{p}}}$ & $c_{\lambda_{\mathrm{d}}}$ **then**
    E_2.add(e);
  end
end
$G_{\mathrm{sub}} \leftarrow$ deleteEdges(G,E_2);

werden in E_1 vermerkt. Allerdings werden hier auch Segmente berücksichtigt, die selbst zwar die Umkugel nicht schneiden, aber ihre Gerade es tut. Mit zunehmender Entfernung von $\boldsymbol{c}$ nimmt jedoch die Wahrscheinlichkeit für die Existenz solcher Segmente rasant ab. Zusätzlich wird durch die zweite Bedingung $2r(e) > \rho$ auch sicher gestellt, dass nur Segmente vermerkt werden, die später auch aufgelöst werden können. Abschließend wird aus G anhand der Liste E_1 der vorläufige Subgraph G_{sub} extrahiert. Dieser enthält bereits alle relevanten Segmente, jedoch nicht ausschließlich diese. Zusätzlich zu den vorhin erwähnten nicht relevanten Segmenten sind in dieser Menge auch die enthalten, die sich im Raum zwischen Umkugel und Volumenwürfel befinden.
Im nächsten Schritt werden die Segmente auf einen Schnittpunkt mit dem Volumenwürfel untersucht. Dazu wird Gleichung (4.27) in zwei Unterformen aufgesplittet, in denen $\lambda \geqslant 0$ gilt. Es ergeben sich dadurch für jedes Segment s zwei gerichtete Geraden s_{p} bzw. s_{d}, die jeweils in die proximale bzw. distale Richtung zeigen.

$$s_{\mathrm{p}} : \boldsymbol{x} = \boldsymbol{p}_{\mathrm{p}} + \lambda \cdot (\boldsymbol{p}_{\mathrm{d}} - \boldsymbol{p}_{\mathrm{p}}) \tag{4.28}$$

$$s_{\mathrm{d}} : \boldsymbol{x} = \boldsymbol{p}_{\mathrm{d}} - \lambda \cdot (\boldsymbol{p}_{\mathrm{d}} - \boldsymbol{p}_{\mathrm{p}}) \tag{4.29}$$

Für die Untersuchung, ob s_{p} und s_{d} sich mit dem Volumenwürfel schneiden, wird eine MATLAB-Implementierung von Jesús Mena, basierend auf den Algorithmen beschrieben in [Smi98; Wil+05] verwendet. Die Funktion $\mathrm{rayBoxIntersection}$ gibt zwei Outputs zurück. Der Output f nimmt den Wert 1 an, wenn die gerichtete Gerade den Würfel schneidet und 0, falls nicht. Der zweite Output gibt den Wert für die Geradenvariable λ an, der in die Gleichung der gerichteten Geraden eingesetzt werden muss, um den Schnittpunkt zu erhalten. Wichtig ist anzumerken, dass es hierbei um den Schnittpunkt der Geraden geht, dieser entspricht nicht in jedem Fall dem des Segments, das nur einen Teil der Geraden ausmacht.
Eine gerichtete Gerade kann einen Würfel entweder nicht, in einem oder in zwei Punkten schneiden. Für das Sampling sind jedoch die Schnittpunkte der Segmente von Interesse. Die Anordnung eines Segments s des Subgraphen und des Würfels können fünf verschiedene Konfigurationen annehmen, welche in Bild 4.16 zu sehen sind. Außer dem Fall *e* schneidet die Segmentgerade immer den Würfel. Im Fall *a* liegt das Segment komplett im Würfel, beim Fall *b* liegt ein Knotenpunkt des Segments im und der andere außerhalb des Würfels. Beide Knotenpunkte liegen im Fall *c* außerhalb des Würfels, ein Stück des Segments verläuft jedoch durch den Würfel. Ebenfalls liegen beide Knotenpunkte außerhalb, jedoch kein Teilstück des Segments im Würfel in den Fällen *d* und *e*.
In den Fällen *b* und *c* müssen die außen liegenden Knotenpunkte neu definiert werden. Dies erfolgt in den Zeilen 18 und 25 für das proximale und das distale Ende jeweils. Den Knotenpunkten wird der Schnittpunkt zugeordnet, der durch λ_{p} bzw. λ_{d} ermittelt werden kann, diese

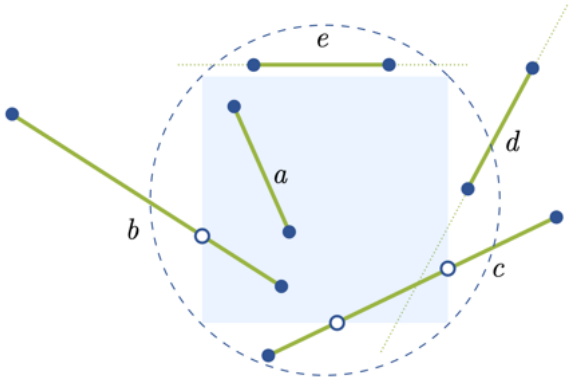


Bild 4.16: Lage von Segmenten und dem Würfel

sind in Bild 4.16 mit den weißen Knotenpunkten gekennzeichnet.
Knotenpunkte im Gefäßbaumgraphen sind entweder Wurzel-, Bifurkations, Blatt- oder innere Knotenpunkte. Dadurch, dass sich ein Segment mit dem Würfel schneidet, kann es vorkommen, dass sich der Knotenpunkttyp ändert. Dies tritt auf, wenn der außenliegende Knotenpunkt ein Bifurkationsknotenpunkt ist. Durch den Schnittpunkt wird das entsprechende Segment aufgeteilt, sodass der Schnittpunkt einem inneren Knotenpunkt entspricht. In diesen Fällen muss deshalb der Typ t von Bifurkation $t = 2$ auf inneren Knotenpunkt $t = 1$ geändert werden. Anschließend werden Segmente, die unter die Fälle *d* und *e* fallen, in E_2 vermerkt. Nachdem über alle Segmente iteriert wurde, werden die in E_2 vermerkten Segmente aus G_{sub} entfernt.

4.2.3 Rendern

Für das Rendern wird der Subgraph G_{sub} sowie die diskrete Kantenlänge $\varepsilon_{\text{discr}}$ benötigt. Zunächst werden die Knotenpunktkoordinaten sowie die Segmentradien diskretisiert. Für jede Kante e und jeden Knotenpunkt n in G_{sub} werden $r(e)$ und $\boldsymbol{p}(n)$ wie folgt neu festgelegt:

$$r'(e) \leftarrow \frac{r(e)}{\rho} \tag{4.30}$$

$$\boldsymbol{p}'(n) \leftarrow \frac{1}{\rho} \cdot (\boldsymbol{p}(n) - \boldsymbol{b}_{\text{l}}) \tag{4.31}$$

Für die Knotenpunktkoordinaten wird außerdem ein neuer Ursprung in $\boldsymbol{b}_{\text{l}}$ definiert. Im Folgenden soll ein Volumen $\boldsymbol{V}_{\text{smpl}} \in \mathbb{R}^{\varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}}}$ erstellt werden. Die in Gleichung (4.31) neu bestimmten Knotenpunktkoordinaten entsprechen bereits den Matrixkoordinaten des zu erstellenden Volumens. Die Segmente sollen als Zylinder abgebildet werden. Damit die Übergänge zwischen den Segmenten glatt sind, werden die Zylinder an den beiden Enden jeweils mit einer

Halbkugel mit dem gleichen Radius wie der des Zylinders erweitert.
Für jedes Segment wird zunächst eine Matrix und darin der Zylinder erstellt, sodass die Achse des Zylinders mit der z-Koordinate der Matrix koinzidiert. Danach wird diese Matrix mithilfe des MATLAB-Befehls imwarp transformiert und schließlich im Volumen an der entsprechenden Stelle platziert. Das Ergebnis stellt Bild 4.17 dar.
Die Voxel sollen jeweils mit einem Label markiert werden je nachdem, ob sie zum Hintergrund,

Bild 4.17: Gerendertes Volumen

zum Gefäß oder zur Mittellinie gehören, oder aber einen Bifurkationspunkt darstellen. Die Mittellinien- und Bifurkationsvoxel bestehen lediglich aus einzelnen Voxeln. Durch Verzerrungen bei der Transformation mittels imwarp kann es infolge von Rundungsfehlern dazu kommen, dass Mittellinien- und Bifurkationsvoxel durch Gefäß- oder Hintergrundvoxel ersetzt werden. Aus diesem Grund werden für die Markierung der Mittellinien- und Bifurkationsvoxel alternative Ansätze verwendet.
Für die Transformation der Mittellinien wird ein Volumen $\boldsymbol{V}_{\mathrm{C}}$ auf der gleichen Weise erstellt, wie dies für die Gefäßvoxel mit Zylinder durchgeführt wurde. Allerdings werden die Mittellinienvoxel markiert, indem gekennzeichnet wird, von welchem Segment aus G_{sub} sie herrühren, vgl. Bild 4.18. Auf diese Weise können die Teilmittellinien verschiedener Segmente auseinander gehalten werden. Anstelle der Zylinder befinden sich in $\boldsymbol{V}_{\mathrm{C}}$ transformierte Geraden. Die Geraden haben infolge der Anwendung von imwarp Unterbrechungen, weil sie während der Transformation verzerrt werden. Im nächsten Schritt werden die Mittellinien für jedes Segment so vervollständigt, dass sie eine zusammenhängende Gerade in $\boldsymbol{V}_{\mathrm{C}}$ bilden. Nach diesem Schritt können die Ergebnisse aus $\boldsymbol{V}_{\mathrm{C}}$ in $\boldsymbol{V}_{\mathrm{smpl}}$ übertragen werden.
Die Bifurkationsvoxel sind alleinstehende Voxel und kommen in einem Sample in der Regel nur vereinzelt vor. Dadurch, dass sie alleinstehend sind, können sie in sich nicht verzerrt werden. Daher wird die Stelle jedes Bifurkationsvoxels einzeln mit Anwendung der gleichen

Bild 4.18: Mittellinen aus verschiedenen Segmenten

Transformationsvorschrift wie für die Zylinder berechnet. Die Bifurkationsvoxel werden dann nachträglich ins Volumen $\boldsymbol{V}_{\text{smpl}}$ eingetragen.
Die Reihenfolge, in der die Voxel gelabelt werden, ist entscheidend. Dies hat den Grund, dass für die Mengen der Bifurkationspunkte B, der Mittellinien- sowie der Gefäßvoxeln C bzw. V der Zusammenhang $B \subset C \subset V$ gilt. Die verwendeten Labelwerte stellt Tabelle 4.1 dar.

Tabelle 4.1: Labelwerte

Label	**Wert**
Hintergrund	$l_{\text{bg}} = 0$
Gefäß	$l_{\text{V}} = 1$
Mitellinie	$l_{\text{C}} = 2$
Bifurkation	$l_{\text{B}} = 3$

Dem Hintergrund wird der Labelwert $l_{\text{bg}} = 0$ zugeordnet. Zu Anfang wird das Volumen mit l_{bg} initialisiert. Das Ergebnis des Renderns ist somit das Volumen $\boldsymbol{V}_{\text{smpl}}$, in dem nacheinander die Gefäß-, die Mittellinien- und die Bifurkationsvoxel markiert wurden.

4.2.4 Rauschen

Wie in Abschnitt 3.1.1 beschrieben, sind die Daten, an denen das neuronale Netz die Segmentierung durchführen soll, verrauscht. Das Rauschen soll aus drei Teilen zusammengesetzt nachgeahmt werden. Zunächst sollen die Gefäße in $\boldsymbol{V}_{\text{smpl}}$ an zufälligen Stellen unterbrochen werden. Im Samplingvolumen $\boldsymbol{V}_{\text{smpl}}$ wurden zudem die Gefäße bisher als perfekte Zylinder dargestellt. Am Volumen sollen Deformationen angewandt werden, die sowohl ein Rauschen im Verlauf des Gefäßes, als auch auf dessen Oberfläche verursachen. In einem letzten Schritt sollen zufällig Rauschobjekte in $\boldsymbol{V}_{\text{smpl}}$ platziert werden. Die Generierung der drei Rauscharten wird im Folgenden beschrieben.

Diskontinuitäten

Der Grundansatz bei der Herstellung der Diskontinuitäten im Gefäßverlauf ist es, $\boldsymbol{V}_{\mathrm{smpl}}$ durch eine bestimmte Anzahl von zufällig orientierter und positionierter Ebenen einer zufälligen Dicke schneiden zu lassen. Damit $\boldsymbol{V}_{\mathrm{smpl}}$ allerdings nicht zu sehr degeneriert wird sollen der zufälligen Orientierung bzw. Positionierung der Ebenen Grenzen gesetzt werden. Die Ebenen sollten sich in $\boldsymbol{V}_{\mathrm{smpl}}$ so verteilen, dass keine zu große Diskontinuitäten zustande kommen, andernfalls könnten komplette Gefäße wegfallen. Insbesondere fallen die Diskontinuitäten groß aus, wenn sich die Ebenen nicht weit genug voneinander entfernt befinden oder wenn sie einen zu großen Winkel mit einem Gefäß einschließen. Um eine optimale Verteilung zu erhalten, werden verschiedene Maße während der zufälligen Generierung der Ebenen betrachtet.
Zum einen wird gefordert, dass die Anzahl der Voxel, in denen sich die Ebenen gegenseitig schneiden, möglichst niedrig ist. Zum anderen wird gefordert, dass die Anzahl der Voxel, in denen die Ebenen $\boldsymbol{V}_{\mathrm{smpl}}$ schneiden, nicht zu hoch wird. Die Voxelanzahl wird in beiden Fällen auf die Anzahl der Voxel in $\boldsymbol{V}_{\mathrm{smpl}}$ bezogen. Bei diesen beiden Forderungen wird die Gesamtheit der bereits generierten Ebenen betrachtet. Somit wird für Ebenen, die zuerst generiert werden, mehr Flexibilität gelassen, aber insgesamt wird verlangt, dass die Forderungen erfüllt werden.
Als ein weiteres Maß wird die Anzahl der Voxel geprüft, in denen $\boldsymbol{V}_{\mathrm{smpl}}$ von der aktuell generierten Ebene, bezogen auf ihre Dicke, geschnitten wird. Diese Anzahl sollte auch nicht zu hoch ausfallen. Das Beziehen auf die Dicke hat den Grund, dass die Generierung von dickeren Ebenen nicht grundsätzlich benachteiligt werden soll. Schließlich wird noch gefordert, dass die Diskontinuitäten sich möglichst gut auf alle in $\boldsymbol{V}_{\mathrm{smpl}}$ befindlichen Gefäßsegmente verteilen. Damit wird verhindert, dass – im Verhältnis zu den Maßen der Ebenen – kleine Segmente durch ungünstig generierte Ebenen wegfallen.
Insgesamt wird eine Ebene so lange zufällig neu generiert, bis die empirisch festgelegten Grenzwerte erreicht werden. Es werden insgesamt drei Ebenen auf diese Weise definiert, zur Generierung wird ein zufälliger Normalvektor und ein zufälliger Punkt aus $\boldsymbol{V}_{\mathrm{smpl}}$ gesamplet. Die Ebenen werden außerdem mit Perlin-Rauschen versetzt, sodass die Dicke der Ebene lokal variiert. Allerdings erzeugen die Ebenen scharfe Schnittkanten an den Gefäßen, welche wegen ihres unnatürlichen Aussehens unerwünscht sind. Aus diesem Grund wird das durch die Ebenen geschnittene $\boldsymbol{V}_{\mathrm{smpl}}$ gefiltert. Der Filter erodiert dabei lediglich die Schnittkanten und lässt die restlichen Strukturen unverändert. Das in Abschnitt 4.2.2 erwähnte Padding wird für das Filtern benutzt. Nach dem Filtern hat $\boldsymbol{V}_{\mathrm{smpl}}$ die Kantenlänge ε. Das von den Ebenen geschnittene Volumen vor und nach dem Filtern ist in Bild 4.19 zu sehen.
Für die Mittelinien- und Bifurkationsvoxel, die in eine Diskontinuität fallen, werden die Labelwerte $l_{\mathrm{C}\times} = 4$ und $l_{\mathrm{B}\times} = 6$ eingeführt.

Bild 4.19: Gefäßvoxel mit Diskontinuitäten vor und nach Filtern der Schnittkanten

Deformationen

Um Deformationen sowohl in Größenordnung der Gefäße, als auch in Form von Oberflächentextur zu erzeugen, wird das komplette Volumen $\boldsymbol{V}_{\text{smpl}}$ mit Perlin-Rauschen in drei verschiedenen Größenordnungen versetzt. Diese korrespondieren mit der 1,00-, 0,25- und 0,05-fachen Kantenlänge $\varepsilon_{\text{discr}}$. Darunter ist zu verstehen, dass zunächst ein $n \cdot \varepsilon_{\text{discr}} \times n \cdot \varepsilon_{\text{discr}} \times n \cdot \varepsilon_{\text{discr}}$ großes 3D-Skalarfeld mit Perlin-Rauschen erstellt wird, das dann auf die Größe von $\boldsymbol{V}_{\text{smpl}}$ hochskaliert wird. Die verschiedenen Rauschen werden superponiert und in Form eines Verschiebungsfeldes auf die Voxel in $\boldsymbol{V}_{\text{smpl}}$ angewandt. Dabei kann durch Verzerrungen ähnlich wie beim Rendern die Kontinuität der Mittellinienvoxel verletzt werden oder Bifurkationsvoxel überschrieben werden. Aus diesem Grund wird das Verschiebungsfeld separat auf die Gefäß-, Mitellinien- und Bifurkationsvoxel angewandt. Die Kontinuität der Mittellinien wird analog zum Verfahren nach dem Rendern wieder hergestellt. Danach werden die Ergebnisse nacheinander, der oben beschriebenen Reihenfolge nach in $\boldsymbol{V}_{\text{smpl}}$ geschrieben. Zuerst werden die verschobenen Gefäßvoxel festgehalten, diese werden von den verschobenen Mittellinien- und zum Schluß von den verschobenen Bifurkationsvoxeln überschrieben. In Bild 4.20 ist $\boldsymbol{V}_{\text{smpl}}$ nach der Deformation dargestellt. Von links nach rechts sind nur die Mittellinien- und Bifurkationsvoxel dargestellt, im nächsten nur die Gefäßvoxel und im letzteren sind alle Label vertreten.

Random Objects

In diesem letzten Schritt werden $\boldsymbol{V}_{\text{smpl}}$ zufällige Objekte (Random Objects, RO) hinzugefügt. Die Herstellung dieser basiert ebenfalls auf dem Perlin-Rauschen. Es werden verrauschte Volumina drei verschiedener Stufen generiert. Die Stufen korrespondieren mit der 0,20-, 0,10- und 0,05-fachen Kantenlänge $\varepsilon_{\text{discr}}$. Die Objekte entstehen durch Binarisierung der 3D-Perlin-Skalarfelder. Die Objekte der drei verschiedenen Stufen werden superponiert und anschließend

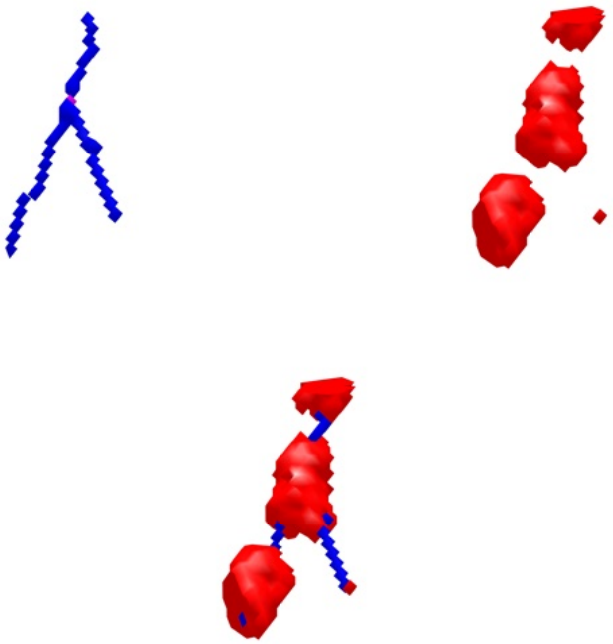


Bild 4.20: Auswirkungen der Deformation

wird darauf das gleiche Verschiebungsfeld wie auf $\boldsymbol{V}_{\text{smpl}}$ für die Erzeugung von Deformationen angewandt. Schließlich werden die Objekte in $\boldsymbol{V}_{\text{smpl}}$ platziert. RO-Voxel erhalten den Labelwert $l_{\text{ro}} = 5$. Die Random Objects sind in Bild 4.21 grün dargestellt.

4.2.5 Erstellung der Datensets

Nach den Schritten der letzten beiden Abschnitte liegt nun ein $\boldsymbol{V}_{\text{smpl}}$ vor, in dem die Hintergrund-, Gefäß-, Mitellinien- Bifurkations- und RO-Voxel einen Wert von 0 bis 6 annehmen. Tabelle 4.3 fasst alle möglichen Labelwerte zusammen.

Aus $\boldsymbol{V}_{\text{smpl}}^{\varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}}}$ sollen Input- sowie Outputmatrizen $\boldsymbol{X}^{\varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}}}$ bzw. $\boldsymbol{Y}^{\varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}} \times \varepsilon_{\text{discr}}}$ erstellt werden. Je nach Ziel der Segmentierung bestehen mehrere Möglichkeiten. Für die binäre Segmentierung hinsichtlich Gefäß-, Mittellinien- oder Bifurkationsvoxel werden die In- und Outputs wie folgt definiert:

$$\boldsymbol{X}_{\text{V}}(i) = \begin{cases} 0, & \boldsymbol{V}_{\text{smpl}}(i) \in \{0, 4, 6\} \\ 1, & \boldsymbol{V}_{\text{smpl}}(i) \in \{1, 2, 3, 5\} \end{cases} \tag{4.32}$$

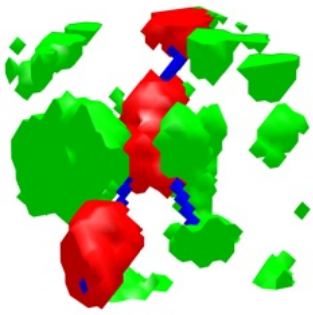


Bild 4.21: Random Objects (grün) im Volumen

Tabelle 4.3: Labelwerte inkl. Rauschen

Label	**Wert**
Hintergrund	$l_{\mathrm{bg}} = 0$
Gefäß	$l_{\mathrm{V}} = 1$
Mittellinie	$l_{\mathrm{C}} = 2$
Mittellinie in Diskontinuität	$l_{\mathrm{C}^{\times}} = 4$
Bifurkation	$l_{\mathrm{B}} = 3$
Bifurkation in Diskontinuität	$l_{\mathrm{B}^{\times}} = 6$
Random Objects	$l_{\mathrm{ro}} = 5$

$$\boldsymbol{X}_{\mathrm{C}}(i) = \begin{cases} 0, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{0,4,5,6\} \\ 1, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{1,2,3\} \end{cases} \tag{4.33}$$

$$\boldsymbol{X}_{\mathrm{B}}(i) = \begin{cases} 0, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{0,1,4,5,6\} \\ 1, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{2,3\} \end{cases} \tag{4.34}$$

$$\boldsymbol{Y}_{\mathrm{V}}(i) = \begin{cases} 0, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{0,4,5,6\} \\ 1, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{1,2,3\} \end{cases} \tag{4.35}$$

$$\boldsymbol{Y}_{\mathrm{C}}(i) = \begin{cases} 0, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{0,1,5\} \\ 1, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{2,3,4,6\} \end{cases} \tag{4.36}$$

$$\boldsymbol{Y}_{\mathrm{B}}(i) = \begin{cases} 0, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{0,1,2,4,5\} \\ 1, & \boldsymbol{V}_{\mathrm{smpl}}(i) \in \{3,6\} \end{cases} \tag{4.37}$$

Die Unterschiede in der Definition der Inputs begründet sich darin, jeweils der Output von der Gefäßsegmentierung V als Input für die Mittelliniensegmentierung C genommen wird, und der Output davon als Input für die Bifurkationserkennung B.
Soll eine Multiklassensegmentierung stattfinden, werden Input- und Outputmatrizen der Form $\boldsymbol{X}^{\varepsilon_{\mathrm{discr}} \times \varepsilon_{\mathrm{discr}} \times \varepsilon_{\mathrm{discr}}}$ bzw. $\boldsymbol{Y}^{\varepsilon_{\mathrm{discr}} \times \varepsilon_{\mathrm{discr}} \times \varepsilon_{\mathrm{discr}} \times 3}$ erstellt. Die vierte Dimension beim Output steht für die drei zu identifizierenden Klassen Hintergrund/Rauschen, Gefäß, Mitellinie sowie Bifurkation. Für die Erstellung des Inputs X_{VCB} gilt in diesem Fall Gleichung (4.32), sodass $X_{\mathrm{VCB}} = X_{\mathrm{V}}$. Beispielhaft ist in Bild 4.22 der Input veranschaulicht, der aus dem Volumen in Bild 4.21 erzeugt wird. Für den Output $\boldsymbol{Y}_{\mathrm{VCB}}$ werden vier Matrizen – entsprechend den

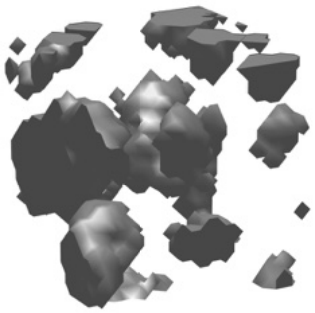


Bild 4.22: Beispiel für Input $\boldsymbol{X}$

Klassen Hintergrund, Gefäß, Mittellinie und Bifurkation – entlang einer vierten Dimension

verkettet. Diese sind:

$$n = 1 : \boldsymbol{Y}_1(i) = \begin{cases} 0, & \boldsymbol{V}_{\text{smpl}}(i) \in \{1, 2, 3, 4, 6\} \\ 1, & \boldsymbol{V}_{\text{smpl}}(i) \in \{0, 1\} \end{cases} \tag{4.38}$$

$$n = 2 : \boldsymbol{Y}_2(i) = \begin{cases} 0, & \boldsymbol{V}_{\text{smpl}}(i) \in \{0, 2, 3, 4, 5, 6\} \\ 1, & \boldsymbol{V}_{\text{smpl}}(i) \in \{1\} \end{cases} \tag{4.39}$$

$$n = 3 : \boldsymbol{Y}_3(i) = \begin{cases} 0, & \boldsymbol{V}_{\text{smpl}}(i) \in \{0, 1, 3, 5, 6\} \\ 1, & \boldsymbol{V}_{\text{smpl}}(i) \in \{2, 4\} \end{cases} \tag{4.40}$$

$$n = 4 : \boldsymbol{Y}_4(i) = \begin{cases} 0, & \boldsymbol{V}_{\text{smpl}}(i) \in \{0, 1, 2, 4, 5\} \\ 1, & \boldsymbol{V}_{\text{smpl}}(i) \in \{3, 6\} \end{cases} \tag{4.41}$$

Die auf diese Weise erstellten Datensets $\{\boldsymbol{X}_{\text{V}}, \boldsymbol{Y}_{\text{V}}\}, \{\boldsymbol{X}_{\text{C}}, \boldsymbol{Y}_{\text{C}}\}, \{\boldsymbol{X}_{\text{B}}, \boldsymbol{Y}_{\text{B}}\}$ sind direkt anwendbar für neuronale Netze zwecks binärer Segmentierung sowie das Datenset $\{\boldsymbol{X}_{\text{VCB}}, \boldsymbol{Y}_{\text{VCB}}\}$ für Multiklassensegmentierung.

5 Das neuronale Netz

Die Netzarchitektur von [Tet+18] soll auf synthetischen Daten angewandt werden, die nach dem im vorigen Kapitel beschriebenen Verfahren hergestellt wurden. Zunächst wird die Klassenverteilung dieser Daten untersucht. Im Anschluss werden die Veränderungen an der Arbeit von Tetteh et al. insbesondere hinsichtlich verwendeter Loss-Function und Metrics werden im Folgenden dargelegt. Darauf folgend werden die Versuche vorgestellt und das Training beschrieben. Im zweiten Teil des Kapitels findet die Evaluierung statt. Diese basiert auf einem synthetischen und einem echten Datensatz.

5.1 Traininig

5.1.1 Untersuchung der Klassenverteilung

Zur Untersuchung der Klassenverteilung wird die relative Häufigkeitsverteilung betrachtet. Es wird untersucht, welchen Anteil Voxel einer Klasse aus dem Samplevolumen ausmachen. Eine Darstellung des daraus resultierenden Histogramms ist in Bild 5.1 zu sehen. Bei einer Darstellung der tatsächlichen relativen Häufigkeitswerte wären die Anteile der Mittellinienklasse kaum und die der Bifurkationsklasse gar nicht erkennbar, sodass eine logarithmische Darstellung notwendig ist. In Bild 5.1 sind die Unterschiede mehrerer Größenordnungen erkennbar. Die Klassen sind desto seltener vertreten, je größer die Werte im Graphen ausfallen. Ein Wert von 0 für eine Klasse würde bedeuten, dass das Sample lediglich aus dieser einen Klasse besteht.
Die Untersuchung ergibt des Weiteren, dass etwa 10,32 % aller Voxel ein Gefäß, die Mittellinie oder eine Bifurkation darstellen. Von diesen Voxeln beträgt der Anteil von Mittellinien und Bifurkationen nur noch circa 1,64 %. Die Bifurkationensvoxel bilden von diesen Voxeln wiederum einen Anteil von ungefähr 3,07 %. Der Anteil von 10,32 % fällt, verglichen mit dem Wert von 2,5 % aus [Tet+18], deutlich höher aus trotz der Tatsache, dass im Datensatz im Gegensatz zu Tetteh et al. auch Diskontinuitäten enthalten sind. Die Erklärung für diesen hohen Wert ist es, dass er sich nicht auf das ursprüngliche Volumen bezieht, sondern auf die Samples, die in dieser Arbeit absichtlich in der Nähe von Gefäßen erzeugt wurden. Somit sind Bereiche des Gesamtvolumens, in denen sich keine Gefäße befinden, in dieser Statistik nicht enthalten.
Da Bifurkationsvoxel im Durchschnitt nur lediglich etwa 0,0052 % aller Voxel ausmachen, sollte überprüft werden, ob die Anzahl der Samples mit Bifurkationsvoxeln ausreichend hoch ist. Das Verhältnis von Samples mit und ganz ohne Bifurkationen stellt Bild 5.2 dar. Nur etwa

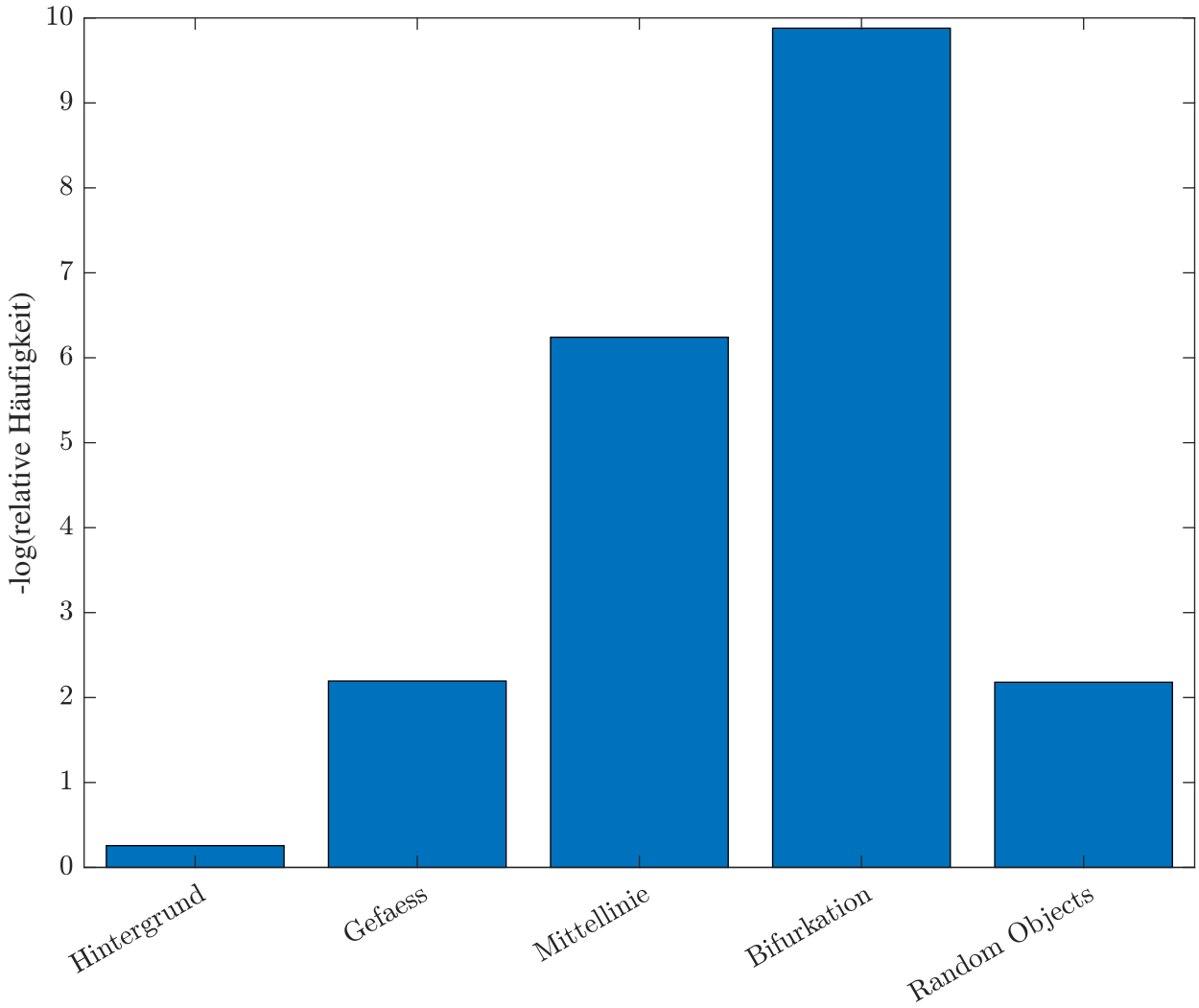


Bild 5.1: Klassenverteilung

40 % aller Samples enthalten eine Bifurkation. Nichtsdestotrotz sind damit Bifurkationen in einem bedeutenden Teil der Samples präsent. Insgesamt weist jedoch die Verteilung der Klassen entsprechend der vorherigen Erwartungen eine extreme Unausgeglichenheit auf. Das soll mit Hilfe einer entsprechenden gewichteten Loss-Function kompensiert werden.

5.1.2 Netzarchitektur

Tetteh et al. teilen die Segmentierung in drei binäre Klassifikationsaufgaben auf. In dieser Arbeit wird zusätzlich auch ein Multiklassenklassifikationsansatz getestet. Für beide Ansätze wird die in Abschnitt 3.3 vorgestellte Netzarchitektur, die in Bild 5.3 nochmal dargestellt ist,

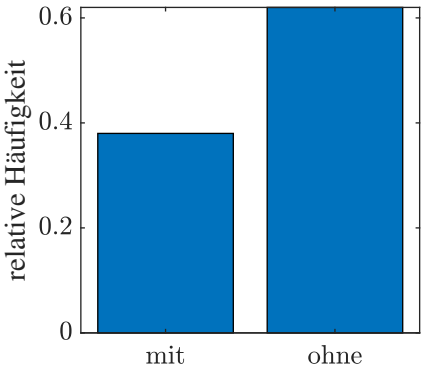


Bild 5.2: Verteilung der Samples mit und ohne Bifurkationspunkt

erweitert. Durch hinzufügen von einem bis zu zwei Layers soll die Modellkomplexität erhöht werden. Dies begründet sich darin, dass sich die Segmentierungsaufgabe in dieser Arbeit durch die verschiedenen Rauscharten deutlich herausfordernder gestaltet als in der Arbeit von Tetteh et al.

Im Folgenden werden zwei Netzarchitekturen getestet: Bei der einen wird der Architektur von

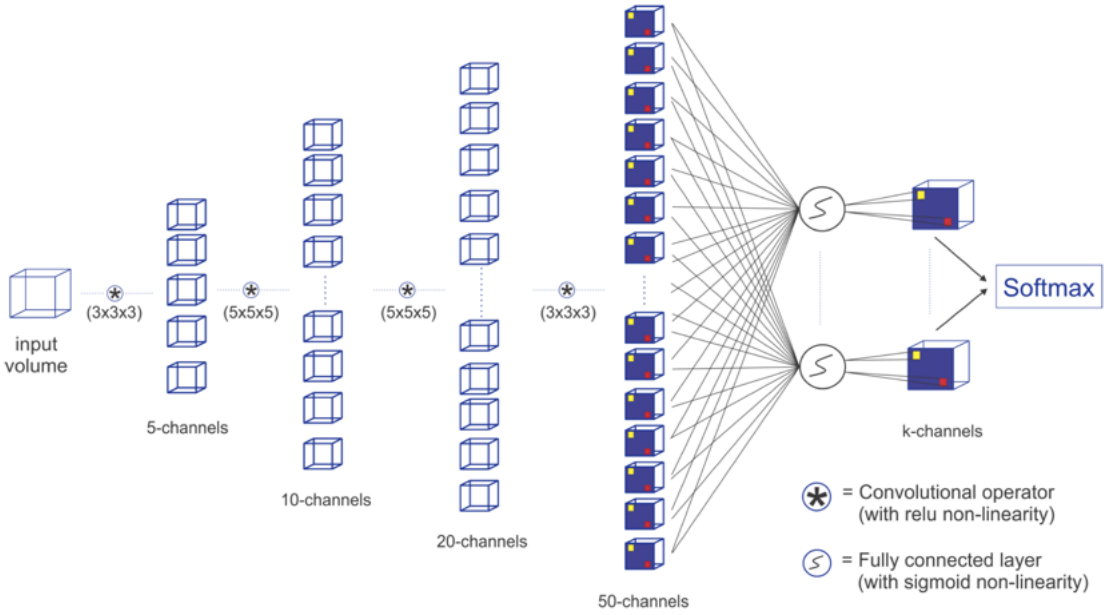


Bild 5.3: DeepVesselNet-FCN nach [Tet+18]

Tetteh et al. ein Layer aus 100, bei der anderen außerdem ein Layer aus 200 Channels vor dem Outputlayer hinzugefügt.

Hinsichtlich der beiden Ansätze ändert sich die Anzahl der Channels des letzten Layers, diese ist in Bild 5.3 als k gekennzeichnet. Die Anzahl der Outputchannels ist gleich der Anzahl der zu unterscheidenden Klassen. Im Fall der binären Klassifikationen gilt somit jeweils $k = 2$, während bei der Multiklassenklassifikation $k = 4$ gilt.
In ihrer Arbeit ersetzen Tetteh et al. die Operation 3D-Faltung durch den in Abschnitt 3.3.1 beschriebenen Cross-Hair-Filter. Dadurch kann zwar Rechenzeit eingespart werden, jedoch geht auch ein Teil der dreidimensionalen Informationen verloren. In dieser Arbeit ergibt sich, dass der Ersatz der 3D-Faltung nicht notwendig ist, da genug Rechenkapazität zur Verfügung steht. Stattdessen wird das Vermeiden von Informationsverlust bevorzugt und die gewöhnliche 3D-Faltung benutzt.
Das neuronale Netz wird in *python* implementiert und benutzt hauptsächlich die Library *keras*.

5.1.3 Loss Functions

Für den binären Klassifikationsansatz wird die in Abschnitt 3.3.2 vorgestellte Loss-Function übernommen, diese ist wie folgt definiert:

$$L = L_1 + L_2 \tag{5.1}$$

$$L_1 = -\frac{1}{|Y_+|} \sum_{n \in Y_+} \log P_n(1) - \frac{1}{|Y_-|} \sum_{n \in Y_-} \log P_n(0) \tag{5.2}$$

$$L_2 = -\frac{\gamma_1}{|Y_+|} \sum_{n \in Y_{\mathrm{f}+}^{\mathrm{pred}}} \log P_n(0) - \frac{\gamma_2}{|Y_-|} \sum_{n \in Y_{\mathrm{f}-}^{\mathrm{pred}}} \log P_n(1) \tag{5.3}$$

$$\gamma_1 = 0{,}5 + \frac{1}{|Y_{\mathrm{f}+}^{\mathrm{pred}}|} \sum_{n \in Y_{\mathrm{f}+}^{\mathrm{pred}}} |P_n(0) - 0{,}5| \tag{5.4}$$

$$\gamma_2 = 0{,}5 + \frac{1}{|Y_{\mathrm{f}-}^{\mathrm{pred}}|} \sum_{n \in Y_{\mathrm{f}-}^{\mathrm{pred}}} |P_n(1) - 0{,}5| \tag{5.5}$$

Diese Formulierung gilt für eine eine negative und eine positive Klasse wie etwa Hintergrund – Gefäß, Hintergrund – Mittellinie oder Hintergrund – Bifurkationspunkte. Ein Sample wird der positiven Klasse zugeordnet, wenn die vorhergesagt Wahrscheinlichkeit den Schwellwert 0,5 erreicht hat. Für den Multiklassenansatz wird die obige Formulierung auf die folgende verallgemeinert:

$$L = L_1 + L_2 \tag{5.6}$$

$$L_1 = \sum_{c=1}^{C} \left[-\frac{1}{|Y_c|} \sum_{n \in Y_c} \log P_n(c) \right] \tag{5.7}$$

$$L_2 = \sum_{c=1}^{C} \left[-\frac{\gamma_c}{|Y_{\bar{c}}|} \sum_{n \in Y_{\mathrm{f}\bar{c}}} \log P_n(c) \right] \tag{5.8}$$

$$\gamma_c = \frac{1}{4} + \frac{1}{|Y_{\mathrm{f}\bar{c}}|} \sum_{n \in Y_{\mathrm{f}\bar{c}}} \left| P_n(c) - \frac{1}{4} \right| \tag{5.9}$$

In den Termen L_1 und L_2 wird über alle Klassen $c \in [1, C]$ iteriert. Bei der binären Klassifikation ist die Gegenwahrscheinlichkeit $\bar{P}_n(c)$ für die Klasse c, das heißt die Wahrscheinlichkeit, dass ein Sample *nicht* Klasse c gehört mit $\bar{P}_n(0) = 1 - P_n(0) = P_n(1)$ bzw. $\bar{P}_n(1) = 1 - P_n(1) = P_n(0)$ eindeutig definiert. Bei einer Multiklassenklassifikation kann die Gegenwahrscheinlichkeit als $\bar{P}_n(c) = 1 - P_n(c) = P_n(\bar{c})$ verallgemeinert werden. Dabei steht $P_n(c)$ für die Wahrscheinlichkeit, dass ein Sample n der Klasse c zugeordnet bzw. $\bar{P}_n(c)$ für die Wahrscheinlichkeit, dass ein Sample n der Klasse c *nicht*, das heißt $\bar{c}$ zugeordnet wird. Somit bezeichnen die Mengen Y_c und $Y_{\bar{c}}$ die Menge aller Samples, die zu Klasse c bzw. nicht zu Klasse c, das heißt zu $\bar{c}$ gehören. Der Ausdruck $Y_{\mathrm{f}\bar{c}}^{\mathrm{pred}}$ bezeichnet die Menge aller Samples, die Klasse c gehören, dieser aber nicht zugeordnet wurden, das heißt $\bar{c}$ zugeordnet wurden. Des Weiteren wird die Wahrscheinlichkeit 0,5 in Gleichung (3.15) bzw. Gleichung (3.16) auf $\frac{1}{4}$ geändert, da $\frac{1}{4}$ gleich der Wahrscheinlichkeit ist, mit der ein Sample zufällig einer der Klassen Hintergrund, Gefäß, Mittellinie oder Bifurkation zugeordnet wird. Bei dieser Formulierung gilt zudem, dass ein Sample zur Klasse mit der größten vorhergesagten Wahrscheinlichkeit zugeordnet wird.

5.1.4 Metrics

In der Arbeit von Tetteh et al. werden die Metrics $precision$, $recall$ und der Dice-Koeffizient Dc benutzt, welche auch für diese Arbeit übernommen werden. Diese Metrics sind jedoch ursprünglich für binäre Klassifikationsaufgaben definiert worden. Aus diesem Grund müssen für den Multiklassenansatz auch die Metrics angepasst werden. Die in Abschnitt 2.3 vorgestellten Definitionen waren die folgenden:

$$precision = \frac{TP}{TP + FP} \tag{5.10}$$

$$recall = TPR = \frac{TP}{TP + FN} \tag{5.11}$$

$$Dc = \frac{2TP}{2TP + FP + FN} \tag{5.12}$$

Die Definitionen beziehen sich auf die Anzahl der true/false Positives/Negatives (TP, FP, TN, FN). Analog zum vorigen Abschnitt werden die Definitionen verallgemeinert indem die

Größen TP, FP, TN, FN jeweils für eine Klasse c betrachtet werden und die Klasse *positive* als Klasse c und *negative* als Klasse nicht c, das heißt $\bar{c}$ interpretiert wird. Diese Interpretation basiert auf der Tatsache, dass die positive Klasse die Klasse der Interesse bzw. des Vordergrunds ist bei einer binären Segmentierung. Die folgenden Zuordnungen (5.16) bis (5.16) verdeutlichen die Verallgemeinerung sowie die Notation.

$$TP \rightarrow TC \tag{5.13}$$

$$FN \rightarrow F\bar{C} \tag{5.14}$$

$$TN \rightarrow T\bar{C} \tag{5.15}$$

$$FP \rightarrow FC \tag{5.16}$$

Für eine bestimmte Klasse – wie beispielsweise die Klasse V der Gefäße – wird die im Folgenden die Notation TC_V usw. verwendet. Die Metrics $precision$, $recall$ und Dc können nun wie folgt definiert werden.

$$precision_c = \frac{TC}{TC + FC} \tag{5.17}$$

$$recall_c = \frac{TC}{TC + F\bar{C}} \tag{5.18}$$

$$Dc_c = \frac{2TC}{2TC + FC + F\bar{C}} \tag{5.19}$$

Der Ausdruck $precision_V$ soll die Genauigkeit für die Klasse V usw. bezeichnen.

5.1.5 Datensätze

Es war die Absicht dieser Arbeit synthetische Daten herzustellen, die sich über die komplette Radienskala strecken. Es soll hier jedoch angemerkt werden, dass aufgrund der mit jeder Auflösungsstufe exponentiell zunehmenden Rechenzeit die Baumgenerierung lediglich bis zur vorletzten Auflösungsstufe durchgeführt wurde.
Die Samples wurden zwecks Training aus diesem Gefäßbaum generiert. Wie in Abschnitt 4.2.1 erläutert, muss für die Erstellung eines Datensatzes zuvor lediglich die diskrete Kantenlänge $\varepsilon_{\text{discr}}$ festgelegt werden. Diese wird so gewählt, dass sie der Kantenlänge des Wurmsichtfeldes ähnlich ist. Die Größenordnung der Sichtfeldkantenlänge liegt aufgrund des verwendeten Multiskalenansatzes in jeder Auflösungsstufe bei den in Abschnitt 3.1.2 angegebenen $30\,\text{px}$. Aus rechenzeittechnischer Überlegung wird $\varepsilon_{\text{discr}}$ für die Generierung der im Folgenden benutzten Datensätze auf $25\,\text{px}$ gesetzt.

Die restlichen Parameter werden von den Samplingalgorithmen so gewählt, dass die Samplingvolumina optisch dem vorverarbeiteten Datensatz möglichst ähnlich sind. Bei der Erstellung der Samples wird in Abschnitt 4.2.1 eine gewisse Ungenauigkeit des Wurmes angenommen. Somit befindet sich der Mittelpunkt der Samplingsvolumina nicht stets in der Mitte eines Gefäßes, sondern liegt im zufälligen Abstand von $\tilde{r}$. Es sollen sowohl Datensätze erstellt werden, die diese eventuelle Ungenauigkeit des Wurmes berücksichtigen, andere sollen dagegen den idealen Fall annehmen, dass sich der Mittelpunkt des Wurmsichtfeldes, somit des Samplingsvolumens auf der Gefäßmittellinie befindet. Diese Datensätze werden mit dem Index c markiert.
Für das Training, die Validierung und die Evaluierung bzw. Testen werden unabhängige Datensätze erstellt. Mit dem Trainingsdatenset D_{train} werden die Gewichte im neuronalen Netz optimiert. Das Validierungsdatenset D_{val} wird zur Einstellung der Hyperparameter verwendet. Das Testdatenset D_{test} wird lediglich zur finalen Auswertung benutzt. Für die Größe der Sets gilt $|D_{\mathrm{train}}| = 600$ sowie $|D_{\mathrm{val}}| = |D_{\mathrm{test}}| = 400$. Des Weiteren wird auch ein Testdatenset auf echten Daten basierend benutzt. Dieses wird im Folgenden als $D_{\mathrm{test,echt}}$ bezeichnet. Dieses Datenset enthält 52 Samples. Der zuvor eingeführten Notation nach bezeichnen $D^{\mathrm{c}}_{\mathrm{train}}$, $D^{\mathrm{c}}_{\mathrm{val}}$ und $D^{\mathrm{c}}_{\mathrm{test}}$ die Datensets, bei denen eine Zentrierung auf die Gefäßmittellinie stattgefunden hat. Mit den definierten Setgrößen stehen beim Training insgesamt $600 \cdot 25 \cdot 25 \cdot 25 = 9{,}375 \cdot 10^6$ Datenpunkten zur Verfügung.

5.1.6 Versuche

Die Gestaltung der Versuche orientiert sich – neben den vorhin vorgestellten Datensätzen – hauptsächlich daran, zwei Konzepte zu untersuchen. Das eine teilt die Segmentierungsaufgabe in drei binäre Klassifikationen auf, das andere fasst diese in eine Multiklassenklassifikation zusammen. Die Versuche werden insgesamt so durchgeführt, dass die verschiedenen Datensätze, Ansätze und Netzarchitekturen getestet werden.
Die folgende Tabelle 5.1 fasst relevante Versuche zusammen. Die Versuchskonfigurationen wurden mit Hilfe der Validierungsdatensätze und der oben definierten Metrics empirisch bestimmt. Es werden acht Versuche zur binären Klassifikation und zwei zur Multiklassensegmentierung durchgeführt. Die Spalte Architektur gibt an, ob das Netz lediglich mit einem 100-Channel oder zusätzlich auch noch mit einem 200-Channel Layer erweitert wurde. Außerdem ist bezüglich der Versuche noch hervorzuheben, dass die Bifurkationssegmentierung mit zwei verschiedenen Inputs $\boldsymbol{X}_{\mathrm{B}}$ und $\boldsymbol{X}_{\mathrm{C}}$ getestet wird. Im ersteren sind lediglich die Gefäßmittellinien enthalten, im letzteren das ganze Gefäß, beide sind jedoch gemäß Abschnitt 4.2.5 ohne Rauschen.
Auf die Versuche wird im Folgenden entsprechend den Zeilen in Tabelle 5.1 mit $\mathrm{V}A_{200}D$ usw. Bezug genommen. Zusätzlich wird zwischen den beiden Versuchen zur Bifurkationserkennung mit $\mathrm{B}A_{100}DB$ und $\mathrm{B}A_{100}DC$ usw. unterschieden.
Um die Entscheidung treffen zu können, welche Versuche für eine weitere Evaluierung relevant

Tabelle 5.1: Übersicht der Versuche

Nr.	Segmentierung	Architektur	Datensatz	Input	Output	Lernrate	Epochs
1	V	A_{200}	D	$\boldsymbol{X}_{\mathrm{V}}$	$\boldsymbol{Y}_{\mathrm{V}}$	$2 \cdot 10^{-4}$	300
2	V	A_{200}	D^{c}	$\boldsymbol{X}_{\mathrm{V}}$	$\boldsymbol{Y}_{\mathrm{V}}$	$2 \cdot 10^{-4}$	300
3	C	A_{100}	D	$\boldsymbol{X}_{\mathrm{C}}$	$\boldsymbol{Y}_{\mathrm{C}}$	$2 \cdot 10^{-5}$	500
4	C	A_{100}	D^{c}	$\boldsymbol{X}_{\mathrm{C}}$	$\boldsymbol{Y}_{\mathrm{C}}$	$2 \cdot 10^{-5}$	500
5	B	A_{100}	D	$\boldsymbol{X}_{\mathrm{B}}$	$\boldsymbol{Y}_{\mathrm{B}}$	$1 \cdot 10^{-4}$	300
6	B	A_{100}	D	$\boldsymbol{X}_{\mathrm{C}}$	$\boldsymbol{Y}_{\mathrm{B}}$	$2 \cdot 10^{-4}$	300
7	B	A_{100}	D^{c}	$\boldsymbol{X}_{\mathrm{B}}$	$\boldsymbol{Y}_{\mathrm{B}}$	$1 \cdot 10^{-4}$	300
8	B	A_{100}	D^{c}	$\boldsymbol{X}_{\mathrm{C}}$	$\boldsymbol{Y}_{\mathrm{B}}$	$2 \cdot 10^{-4}$	300
9	VCB	A_{100}	D	$\boldsymbol{X}_{\mathrm{VCB}}$	$\boldsymbol{Y}_{\mathrm{VCB}}$	$2 \cdot 10^{-4}$	300
10	VCB	A_{100}	D^{c}	$\boldsymbol{X}_{\mathrm{VCB}}$	$\boldsymbol{Y}_{\mathrm{VCB}}$	$2 \cdot 10^{-4}$	300

sind, werden die Verläufe der Metrics für die einzelnen Versuche betrachtet. Die Metrics sind dabei auf den Validierungsdatensatz bezogen. Die Verläufe in Bezug auf das Trainingsdatenset werden nicht dargestellt, die beiden Verläufe verhalten sich sehr ähnlich. Dies hat den Grund, dass die Samples untereinander durch den Samplingsprozess sehr verschieden sind, sodass das Netz gut verallgemeinern kann.
Zunächst werden die Versuche $\mathrm{V}A_{200}D$ und $\mathrm{V}A_{200}D^{\mathrm{c}}$ betrachtet. In Bild 5.4 sind die Verläufe der Metrics während des Trainings aufgetragen. Alle Metrics bewegen sich am Ende des Trainings im Bereich von $[0{,}6, 0{,}8]$. Das Netz, das auf dem Datensatz D^{c} trainiert wurde schneidet etwas besser ab. Von den beiden wird somit dieses zur späteren Evaluation ausgewählt. Als nächstes werden die Metricsverläufe analog für die binäre Mittelliniensegmentierung, das heißt für die Versuche $\mathrm{C}A_{100}D$ und $\mathrm{C}A_{100}D^{\mathrm{c}}$ betrachtet. Bild 5.5 stellt die Metrics während des Trainings dar. Die Metrics fallen alle sehr niedrig aus. Dies ist unerwartet, da die Inputs für die Mittelliniensegmentierung kein Rauschen enthalten. Intuitiv würde man also die Gefäßsegmentierung als die herausforderndere Aufgabe einschätzen. Offensichtlich konnten keine Hyperparameter gefunden, die zur zufriedenstellenden Konvergenz des Netzes geführt hätten können. Im nächsten Bild 5.6 folgen die Metricsverläufe zur binären Bifurkatonssegmentierung. Es werden die vier Versuche $\mathrm{B}A_{100}DB$, $\mathrm{B}A_{100}DC$, $\mathrm{B}A_{100}D^{\mathrm{c}}B$ und $\mathrm{B}A_{100}D^{\mathrm{c}}C$ betrachtet. Es

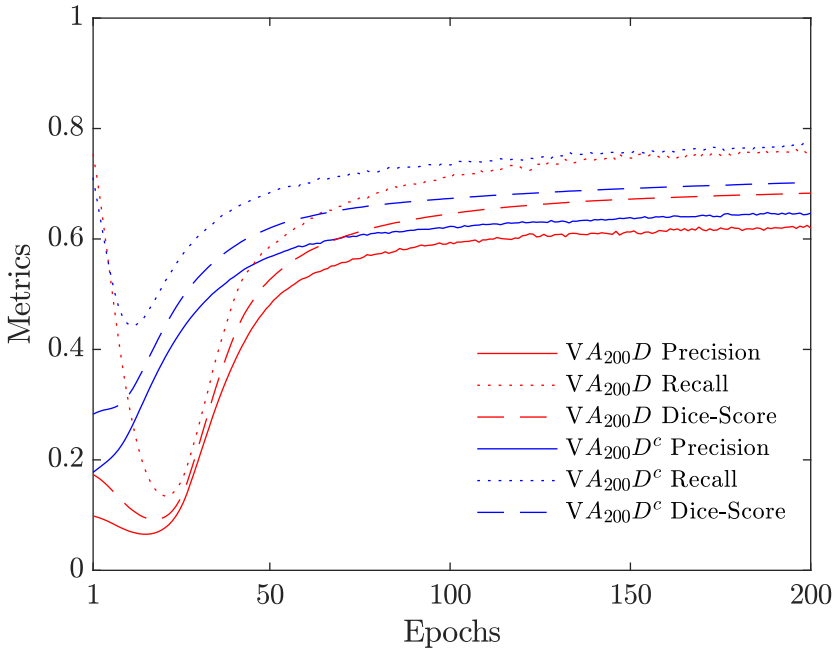


Bild 5.4: Verläufe der Metrics für das Validierungsdatenset bei der binären Gefäßsegmentierung

kann festgestellt werden, dass die Netze, die mit den Daten aus D^{c} traininert wurden, bei der Validierung deutlich besser abschneiden. Des Weiteren ist es von Vorteil für den Input statt $\boldsymbol{X}_{\mathrm{B}}$ den Input $\boldsymbol{X}_{\mathrm{C}}$ zu nehmen. Der Input $\boldsymbol{X}_{\mathrm{B}}$ beinhaltet lediglich die Mittellinien ohne Rauschen. Ein möglich Erklärung für das beobachtete Verhalten ist es, dass die Mittellinien aufgrund ihrer filigranen Struktur bei der Faltung verschwinden. Stellt man $\boldsymbol{X}_{\mathrm{C}}$ als Input zur Verfügung passiert dies nicht und das Netz kann lernen. Insgesamt fallen die Genauigkeiten jedoch ähnlich zur binären Mittelliniensegmentierung sehr gering aus. Aus diesem Grund werden diese Netze nicht zur Evaluierung ausgewählt. Als nächstes werden die Validierungsergebnisse der Multiklassensegmentierung betrachtet. Für die entsprechenden Versuche $\mathrm{VCB}A_{100}D$ und $\mathrm{VCB}A_{100}D^{\mathrm{c}}$ sind die Metricsverläufe in Bild 5.7 zusammengefasst. Ähnlich wie bei der binären Segmentierung können Mittellinien und Bifurkationen am Validierungsset nicht gut klassifiziert werden. Auch bei diesen Versuchen schneidet das Netz, das mit dem Datensatz D_{c} trainiert wurden, besser ab. Dieses wird zur Evaluierung ausgewählt.
Insgesamt kann festgehalten werden, dass für den Fall $\tilde{r} = 0$, das heißt für den Datensatz D_{c} die Metrics besser ausfallen. Die Validierungsergebnisse für die Gefäßsegmentierung fallen sowohl für den binären, als auch für den Multiklassenansatz zufriedenstellend aus. Im Gegensatz dazu liegt für die Mittellinien- und Bifurkationssegmentierung Verbesserungsbedarf vor.

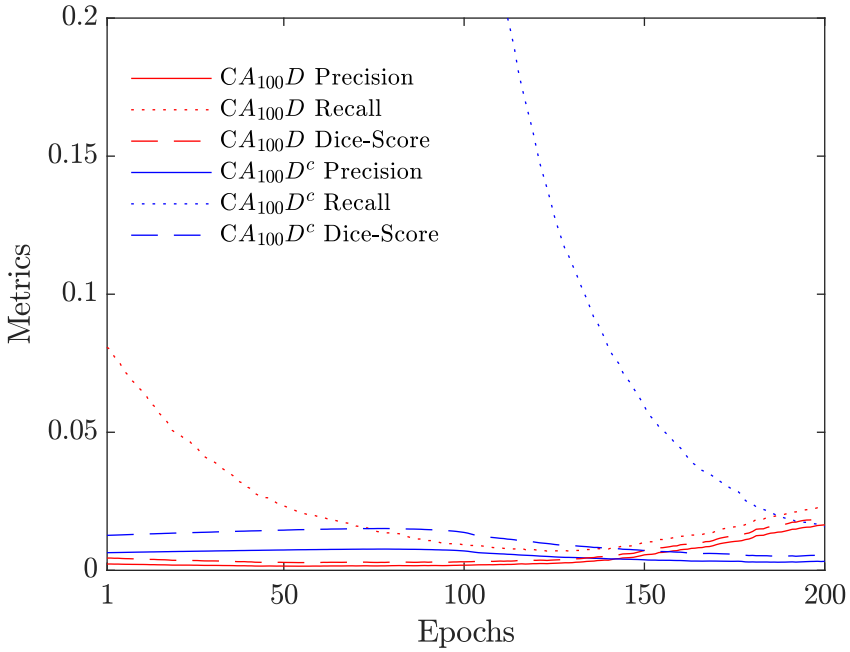


Bild 5.5: Verläufe der Metrics für das Validierungsdatenset bei der binären Mittelliniensegmentierung

5.2 Evaluierung und Diskussion

Basierend auf den Validierungsergebnissen werden nun Evaluierungsergebnisse für die Netze aus den Versuchen $\mathrm{V}A_{200}D^{\mathrm{c}}$ und $\mathrm{VCB}A_{100}D^{\mathrm{c}}$ präsentiert. Letzteres wird lediglich für die Klasse der Gefäße evaluiert, da die anderen Klassen laut Validierung nur ein unzufriedenstellendes Ergebnis aufzeigten. Für das Netz aus dem Versuch $\mathrm{V}A_{200}D^{\mathrm{c}}$ erfolgt die Evaluierung auf zwei Datensätzen, einem synthetischen und einem echten. Das Netz aus $\mathrm{VCB}A_{100}D^{\mathrm{c}}$ wird lediglich an synthetischen Daten evaluiert, da die echten Evaluierungsdaten lediglich binär gelabelt wurden in Gefäß- und Hintergundklassen. Der synthetische Evaluierungssdatensatz D_{test} ist ein Datensatz der Art D^{c}, sodass auch $D^{\mathrm{c}}_{\mathrm{test}}$ geschrieben werden kann.
Der echte Evaluierungsdatensatz $D_{\mathrm{test,echt}}$ entsteht, indem Bilddaten aus Abschnitt 3.1.1 zu einem Teil gelabelt werden. Dabei wird in den Bildern das Gefäßlumen markiert. Das Labeln erfolgt manuell und wird lediglich auf der Auflösungsstufe H durchgeführt. Die Detailliertheit des segmentierten Gefäßbaumes wird somit deutlich durch die niedrige Auflösungsstufe begrenzt. Eine größere Auflösungsstufe kommt jedoch aufgrund des Zeitaufwands nicht in Frage. Die Evaluierung am echten Datensatz soll nicht nur sampleweise erfolgen, sondern die Vorhersagen sollen auch in Form einer Gefäßbaumsegmentierung veranschaulicht werden. Dies soll zur Beurteilung des Netzes beitragen. Im Folgenden wird zunächst die Evaluierung am

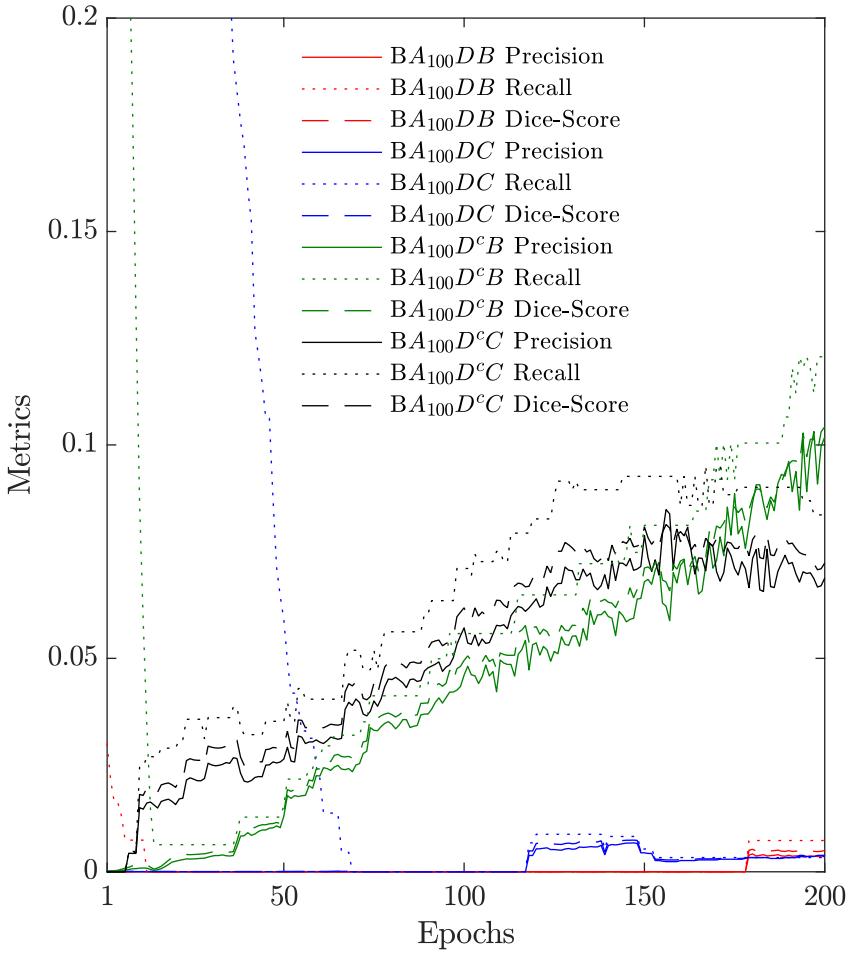


Bild 5.6: Verläufe der Metrics für das Validierungsdatenset bei der binären Bifurkatonssegmentierung

synthetischen, darauffolgend am echten Datensatz vorgestellt.

5.2.1 Synthetischer Evaluierungsdatensatz

Die in Abschnitt 5.1.4 vorgestellten und beim Training verwendeten Metrics werden im Folgenden auf den Vorhersagen ausgewertet. Bild 5.8 zeigt die Metrics für das neuronale Netz aus Versuch $\mathrm{V}A_{200}D^{\mathrm{c}}$, Bild **??** für die Multiklassensegmentierung der Gefäßklasse.
Die Performance der Segmentierung von Tetteh et al. hinsichtlich der betrachteten Metrics wird im Allgemeinen nicht erreicht. Die Ergebnisse dieser Arbeit sind jedoch nur bedingt mit denen von Tetteh et al. vergleichbar, da sich hier die Segmentierungsaufgabe bedingt durch das ausgeprägte Rauschen und die große Radienskala im generierten Gefäßbaum deutlich komplexer

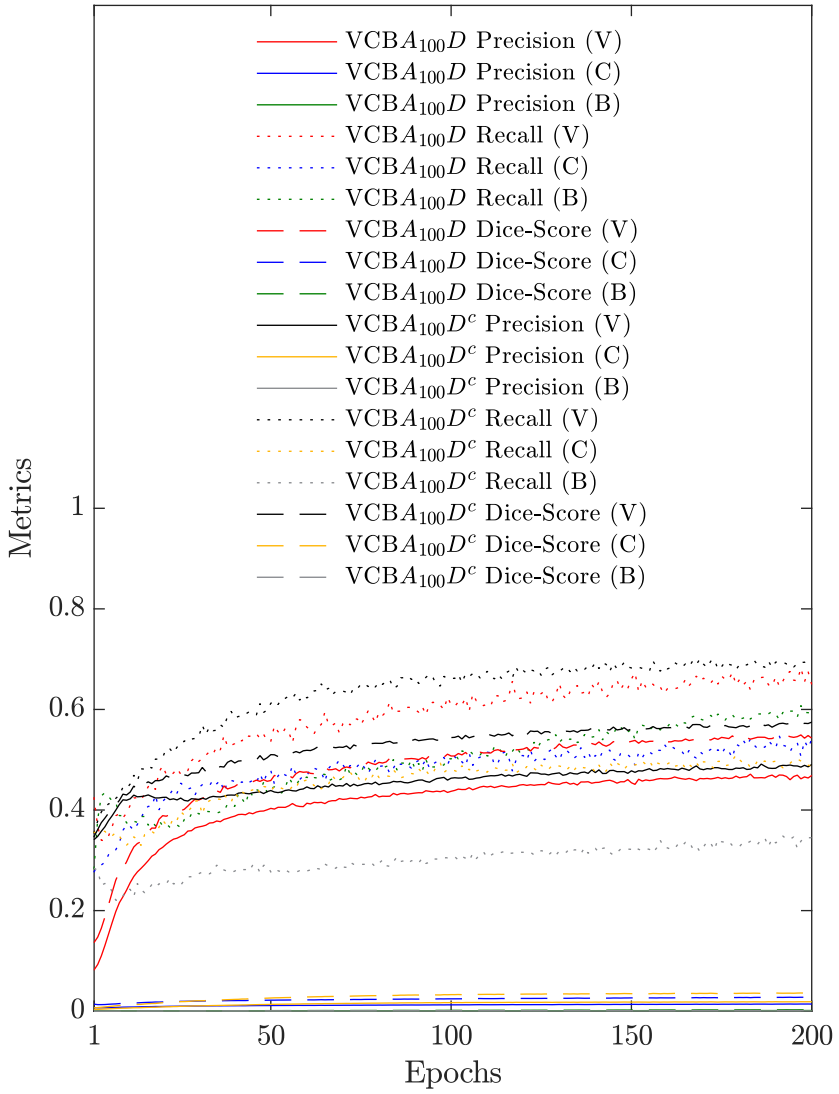


Bild 5.7: Verläufe der Metrics für das Validierungsdatenset bei der Multiklassensegmentierung

gestaltet. An den Ergebnissen fällt auf, dass der Lerneffekt bei der Mittellinien- und Bifurkationssegmentierung nicht einsetzt, weder bei der binären, noch bei der Multiklassenklassifikation. In letzterem Fall könnte ein Grund in der entworfenen Loss-Function liegen. Verglichen mit dem binären Ansatz, muss beim Multiklassenansatz nicht nur ein Ungleichgewicht zwischen zwei Klassen balanciert werden, sondern zwischen mehreren. Im Fall dieser Arbeit fallen die Unterschiede unterhalb der Klassen selbst auch extrem aus. Das heißt, es bestehen mehrfach Unterschiede in der Größenordnung hinsichtlich der Klassengrößen. Es sollte untersucht werden, ob die entworfene Loss-Function in der Lage ist mit solchen mehrfachen Ungleichgewichten

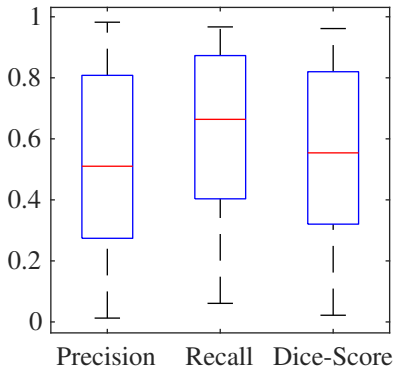


Bild 5.8: Evaluierungsergebnisse für binäre Gefäßsegmentierung $\mathrm{V}A_{200}D^{\mathrm{c}}$

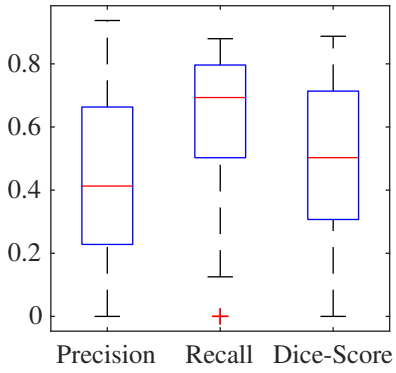


Bild 5.9: Evaluierungsergebnisse Multiklassensegmentierung der Gefäße $\mathrm{VCB}A_{100}D^{\mathrm{c}}$

umzugehen.
Bei der Mittellinien- und Bifurkationssegmentierung besteht eine hohe FC-Rate, sodass der Recall zwar höher ausfällt, die Precision und der Dice-Score aber dagegen nicht. Das heißt die gesuchten Voxel werden zwar erkannt, aber viele andere werden ebenfalls als Bifurkation oder Mittellinie zugeordnet.
Im Folgenden sind einige Sample-Prediction-Paare dreidimensional dargestellt für das Netz aus Versuch $\mathrm{V}A_{200}D^{\mathrm{c}}$. Die Bilder zeigen, dass das trainierte neuronale Netz einem Filter ähnlich arbeitet. Es versucht störende Größen zu eliminieren, aber die eigentlichen Gefäße unberührt zu lassen.

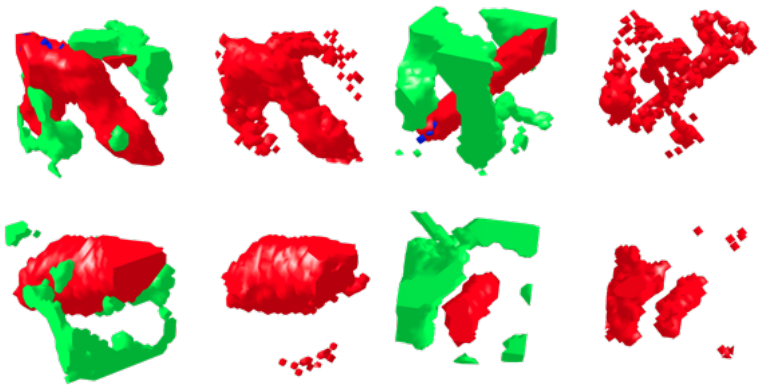


Bild 5.10: Synthetische Sample-Prediction-Paare

5.2.2 Echter Evaluierungsdatensatz

Der aus den Bilddaten manuell segmentierte Gefäßbaum ist in Bild 5.11 zu sehen und das Volumen, in das er eingebettet ist, hat die diskreten Maße von $804\,\mathrm{px} \times 702\,\mathrm{px} \times 429\,\mathrm{px}$. Um die Predictions bestimmen zu können, muss das Volumen in Würfel der Kantenlänge ε

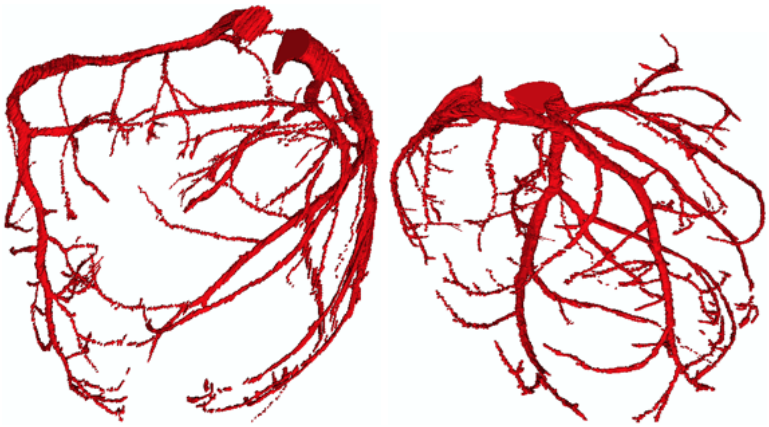


Bild 5.11: Manuell segmentierter Gefäßbaum

aufgeteilt werden. Eine gitterartige Aufteilung würde zu – aus Sicht des Wurmalgorithmus – unrealistischen Schnittkanten führen, sodass bei der Aufteilung ein anderer Ansatz verfolgt wird. Der segmentierte Gefäßbaum wird zunächst skeletonisiert, um die Mittellinie zu erhalten. Aus der Menge S, die alle Voxel des Skeletts enthält, werden Voxel zufällig ausgewählt und als Mittelpunkt des Würfels gesetzt. Dadurch wird gewährleistet, dass die Würfel ihren Mittelpunkt immer in einem Gefäß haben. Bild 5.12 zeigt auf der linken Seite in schwarz die Position eines

Würfelsets und auf der rechten Seite in rot die erkannten Gefäße. In rosa und hellblau sind die

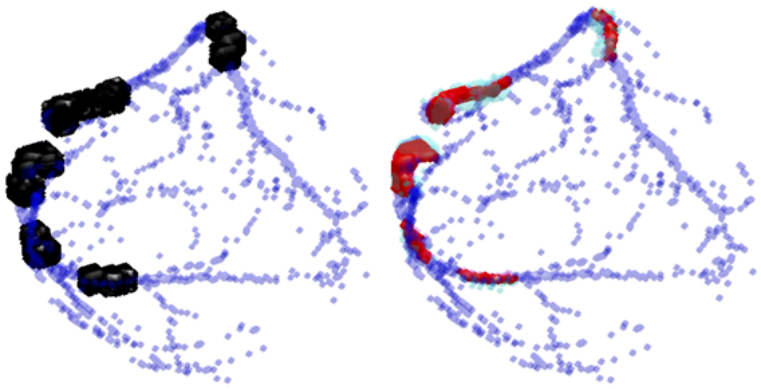


Bild 5.12: Lage des Würfelsets im Gefäßbaum und erkannte Gefäße

false Positives und Negatives veranschaulicht. Die Fehlklassifikation dieser Voxel rührt jedoch nicht in allen Fällen von den Fehlern des Netzes her, da auch Ungenauigkeiten im gelabelten Datensatz vorzufinden sind. Wird bspw. ein nicht richtig gelabeltes Voxel, basierend auf dem Wissen des neuronalen Netzes richtig zugeordnet, wird dies als false Positive eingeordnet.
Analog zur Evaluierung mittels synthetischer Daten, sollen die Metrics auch für den echten Evaluierungsdatensatz bestimmt werden. Die Metrics für ein zufälliges Set aus 52 Samples aus dem segmentierten Gefäßbaum sind in Bild 5.13 dargestellt. Diese weichen von den Evaluie-

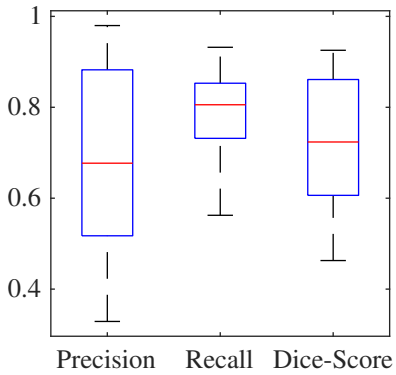


Bild 5.13: Metrics für Evaluierungsset am echten Datensatz

rungsergebnissen des Netzes aus Versuch $\mathrm{V}A_{200}D^{\mathrm{c}}$ an synthetischen Daten nicht bedeutend ab. Dies kann so gedeutet werden, dass in den synthetischen Daten das Rauschen ausgeprägter ist, als in den echten.

6 Fazit und Ausblick

Im Folgenden wird eine Zusammenfassung über die Erkenntnisse dieser Masterarbeit gegeben. Die Ergebnisse werden hinsichtlich der zu Anfang beschriebenen Motivation und des dargelegten Ziels der Arbeit betrachtet. Im Anschluss werden in einem Ausblick Verbesserungsvorschläge vorgestellt und zukunftsrelevante, aus den Ergebnissen abgeleitete, sinnvolle Forschungsaspekte thematisiert.

6.1 Fazit

Das Ziel dieser Masterarbeit war es, in Bezug auf die Gefäßbaumsegmentierung aus mikroskopischen Bilddaten eines Schweineherzens einen neuen Ansatz in Form neuronaler Netze zu erproben. Insbesondere war dies mit der Herstellung synthetischer, möglichst realistischer Daten verbunden. Diese Daten wurden dann für das Training eines implementierten neuronalen Netzes herangezogen, das auch schon an einem ersten, echten Datensatz untersucht werden konnte. Das so entstehende neue Vorgehen soll später in den bereits unter Entwicklung stehenden, umfassenden Segmentierungsalgorithmus integriert werden, sodass die daraus resultierenden Randbedingungen bei der Datenherstellung und beim Training des neuronalen Netzes stets zu beachten waren.
In Zukunft soll die Gefäßbaumsegmentierung Gefäße aller vorkommenden Radien beinhalten. Aus diesem Grund wurde die synthetische Datenherstellung so ausgelegt, dass unter den synthetischen Daten auch Radien aller Größen vorkommen. Im Fall des behandelten Schweineherzens bedeutete dies Gefäße mit einem Durchmesser im Bereich von etwa 2 µm-3 µm bis 1 mm-1,5 mm. Für die synthetische Datenherstellung diente die Arbeit von Schneider et al. in [Sch+12] als Ausgangspunkt. Sie beschreiben ein komplexes, iteratives Vorgehen, das aus einem Saatgefäßsegment einen Gefäßbaum wachsen lässt. Das Vorgehen ist in einen Multiskalenansatz eingebettet, sodass mit dem Durchlaufen verschiedener Skalen Gefäße mit Durchmesser verschiedener Größenordnungen wachsen.
In der Arbeit von Schneider et al. werden Algorithmen verwendet, die den Gefäßbaum nicht als Volumen – etwa in Form einer 3D-Matrix – sondern als Graphen abbilden. Für die Bestimmung des Knotenpunktes, an dem sich das Wachstum im nächsten Iterationsschritt fortsetzt, für die Berechnung der Wachstumsrichtung in diesem Knotenpunkt sowie für die Anwendung des Multiskalenansatzes werden jedoch Skalarfelder verwendet. Diese haben eine Auflösung,

die in der Größenordnung der kleinsten Gefäße liegt. Für die vorliegende Arbeit hätte die Verwendung solcher Skalarfelder bei Auflösung der geforderten Radienskala zu einem enormen Rechenaufwand geführt. Daher wurden neue Algorithmen entwickelt. Diese basieren auf der Verteilung der Knotenpunktoordinaten des Gefäßbaumgraphen und ermöglichen eine möglichst effiziente Alternative zu den Algorithmen von Schneider et al.
Das Ergebnis der synthetischen Gefäßbaumgenerierung ist ein Graph, in dem die Knotenpunkte und Kanten abgespeichert sind, die die Segmente des Gefäßbaumes repräsentieren. Ebenfalls abgespeichert sind die Koordinaten der Knotenpunkte, die sie im repräsentierten Gesamtvolumen einnehmen, sowie die Radien der Kanten. Der nächste Schritt bestand darin, aus dieser Repräsentation für das neuronale Netz nutzbare Samples zu erstellen. Zunächst wird dafür zufällig ein Knotenpunkt aus dem Gefäßbaumgraphen ausgewählt. Das gesamplete Volumen befindet sich in der Nähe dieses Knotenpunkts. Nach Festlegung der Position und Größe des Samplingsvolumens im imaginären Gesamtvolumen, wird mit Hilfe eines dafür entworfenen Algorithmus jener Teilgraph aus dem Gefäßbaumgraphen extrahiert, der die für das Sample relevanten Knotenpunkte und Kanten beinhaltet. Basierend darauf wird ein Volumen in Form einer 3D-Matrix gerendert. Dabei werden die Voxel gelabelt, indem sie entweder dem Hintergund, einem Bifurkationspunkt, der restlichen Gefäßmittellinie oder dem restlichen Gefäß zugeordnet werden.
Um die aus den Mikroskopaufnahmen stammenden, bereits vorverarbeiteten Daten möglichst nachzuahmen, wird das Samplevolumen weiter verarbeitet. Die Gefäße werden in mehreren Größenordnungen deformiert, es werden Diskontinuitäten und zusätzliches Rauschen in Form von zufälligen Störobjekten dem Sample hinzugefügt. Die so erhaltenen Samples sind die synthetischen Äquivalente von den Ausschnitten aus dem vorverarbeiteten Datensatz, die der vollständige Segmentierungsalgorithmus in einem Iterationsschritt verarbeitet.
Die implementierte Netzarchitektur basiert auf der Arbeit von Tetteh et al., die in [Tet+18] ein neuronales Netz zwecks Gefäßsegmentierung benutzen. Die Segmentierung umfasst die Klassifikation von Voxeln in Hintergund-, Gefäß-, Mittellinien- und Bifurkationsvoxel. Hinsichtlich der Verteilung der zu diesen Klassen gehörenden Voxeln besteht im echten sowie im synthetischen Datensatz ein extremes Ungleichgewicht. Dieses Ungleichgewicht wurde mit einer entsprechenden Loss-Function kompensiert, welche aus [Tet+18] entnommen wurde. Während Tetteh et al. die Segmentierungsaufgabe in drei binäre Klassifikationen in Gefäß-, Mittellinien- und Bifurkationsvoxel aufteilen wurde in dieser Arbeit zusätzlich ein Multiklassenansatz ausprobiert. Bei diesem wird nur insgesamt ein Netz für alle Klassen trainiert. Die verwendete Netzarchitektur basiert auf der von Tetteh et al. und wurde mit einem bis zwei zusätzlichen hidden Layers erweitert, um die höhere Komplexität durch das ausgeprägte Rauschen vor allem durch Störobjekte im Sample erfassen zu können. Die Netze wurden auf zwei synthetischen Datensets von je 1000 Samples trainiert, wovon je 600 zum Training und 400

zur Validierung benutzt wurden. Der Unterschied zwischen den beiden Datensets bestand darin, dass bei dem einen die Idealisierung getroffen wurde, dass sich der Samplemittelpunkt genau auf der Gefäßmittellinie befindet. Diese Idealisierung ergibt sich bei einer idealen Funktion des vollständigen Segmentierungsalgorithmus. Basierend auf den Validierungsergebnissen fand die Evaluierung schließlich auf dem idealisierten Datenset statt für die Gefäßsegmentierung mittels binärer Klassifikation sowie für die Multiklassenklassifikation. Für die anderen Klassifikationsaufgaben konnten im Training im Rahmen dieser Arbeit keine zufriedenstellenden Validierungsergebnisse erarbeitet werden.
Die Evaluierung erfolgte mittels eines Datensets von 400 synthetischen – ebenfalls hinsichtlich der Mittelpunkte idealisierten – Samples sowie für den binären Ansatz zusätzlich mittels eines Datensets von 52 echten Samples. Als Metrics wurden dafür entsprechend Tetteh et al. die Größen Precision, Recall und Dice-Score verwendet. Für die Klassifikation von Gefäßen ergaben sich die besten Ergebnisse: Der Dice-Koeffizient beträgt nach der Evaluierung auf dem synthetischen Datensatz $0{,}4472 \pm 0{,}2468$ für den binären und $0{,}5046 \pm 0{,}2268$ für den Multiklassenansatz. Die Evaluierung auf dem echten Datensatz ergab für den binären Ansatz einen Dice-Koeffizienten von $0{,}7213 \pm 0{,}1404$.
Die Ergebnisse dieser Arbeit zeigen, dass ein Bildsegmentierungsansatz in Form eines neuronalen Netzes – zumindest für die Detektion von Gefäßen in verrauschten Daten – ein viel versprechender Ansatz ist. Die trainierten Netze stellen jedoch lediglich einen Anfang eines Findungsprozesses von Hyperparametern dar. Die bedeutendste Errungenschaft dieser Arbeit ist es somit, dass das Training von Netzen durch synthetische Daten ermöglicht wurde. Dadurch konnte ein enormer Zeitaufwand erspart werden, den andernfalls das Labeln der Daten in Anspruch genommen hätte. Es wurden insgesamt etliche Konzepte und Algorithmen präsentiert, die ein flexibles Framework mit vielen frei bestimmbaren Parametern bilden. Mittels dieses Frameworks besteht in Zukunft die Möglichkeit beliebig viele Daten zu generieren und die Leistung der neuronalen Netze durch Finetuning der Hyperparameter zu erhöhen. Somit wird eine wichtige Basis für zukünftige Arbeiten geschaffen.

6.2 Ausblick

Aufbauend auf dieser Basis sollte in Zukunft eine umfassende Versuchsreihe stattfinden, die eine höhere Genauigkeiten für die Gefäß-, und insbesondere für die Mittellinien und Bifurkationserkennung anzielt. Die Gründe sollten untersucht werden, warum die Validierungsergebnisse für die Gefäßmittellinie und die Bifrukationspunkten im Gegensatz zur Gefäßerkennung unzufriedenstellend ausfallen. Es sollte die Verallgemeinerung der von Tetteh et al. benutzten Loss-Function für Multiklassenklassifikationszwecke überprüft und ggf. erweitert werden. Das neuronale Netz könnte zudem mit den synthetischen Daten vortrainiert und den echten

zu Ende trainiert werden. Des Weiteren könnten auch alternative Netzarchitekturen für die Weiterentwicklung miteinbezogen werden. Da der Durchmesser des Gefäßes am aktuellen Punkt der Wurmspitze stets bekannt ist, könnte diese Information beispielsweise als ein zweiter Input für das neuronale Netz dienen. Auch die aktuelle Auflösungsstufe, die im Algorithmus 1 durchlaufen wird, könnte als zusätzlicher Input für das neuronale Netz verwendet werden. Darüber hinaus wurde in dieser Arbeit kein Ähnlichkeitsmaß für einen quantitativen Vergleich zwischen echten und synthetischen Daten definiert. Es sollte versucht werden, ein solches Maß herzuleiten. Dadurch könnte die Ähnlichkeit verbessert und damit die Genauigkeit des neuronalen Netzes erhöht werden.
Des Weiteren sollte die Form der genauen Integration der Vorhersagen des neuronalen Netzes in den vollständigen Segmentierungsalgorithmus festgelegt werden. Insbesondere sollte dabei zwischen den Vorhersagen für die jeweiligen Klassen unterschieden werden. Hinsichtlich des vollständigen Segmentierungsalgorithmus sollte zudem die Option untersucht werden, die originalen Intensitätswerte statt der in dieser Arbeit benutzten binären Daten zu verwenden.
Der aktuelle, vollständige Segmentierungsalgorithmus umfasst lediglich die Erkennung des arteriellen Gefäßbaumes, genauso wie die synthetische Baumgenerierung nur den arteriellen Teil abdeckt. Als ein langfristiges Ziel sollte daher gesetzt werden, das Kapillarnetz im Detail sowie den venösen Gefäßbaumteil zu erarbeiten. Nicht nur der vollständige Segmentierungsalgorithmus sollte diesbezüglich angepasst werden, sondern über die Baumgenerierung sollte auch das neuronale Netz entsprechend neu trainiert werden. Eine zusätzliche Klassifikationsaufgabe neben der Erkennung von Gefäßen im Allgemeinen könnte in diesem Zusammenhang auch die Unterscheidung zwischen arteriellen, kapillären und venösen Gefäßen sein.

Literatur

[AJM16] Galia Assadi, Ralf J Jox und Georg Marckmann. „Organ Transplantation in Times of Donor Shortage. An Introduction“. In: *Organ Transplantation in Times of Donor Shortage*. Springer, 2016, S. 1–3.

[Alt19] Valentina Alto. *Neural Networks: parameters, hyperparameters and optimization strategies*. Juli 2019. URL: https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5.

[Aru18] Arunava. *Convolutional Neural Network*. Dez. 2018. URL: https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05.

[Bai19] Kunlun Bai. „A Comprehensive Introduction to Different Types of Convolutions in Deep Learning“. In: (Feb. 2019). URL: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.

[Bok19] Kaushik Bokka. *Guide to choosing Hyperparameters for your Neural Networks*. Feb. 2019. URL: https://towardsdatascience.com/guide-to-choosing-hyperparameters-for-your-neural-networks-38244e87dafe.

[BS19] Lauren Brasile und Bart Stubenitsky. „Will cell therapies provide the solution for the shortage of transplantable organs?“ In: *Current opinion in organ transplantation* 24.5 (2019), S. 568–573.

[Coc00] Pierre Cochat. *Transplantation and Changing Management of Organ Failure: Proceedings of the 32nd International Conference on Transplantation and Changing Management of Organ Failure [sic], Lyon, 25-26 May 2000*. Bd. 32. Springer Science & Business Media, 2000. ISBN: 0792364201.

[DS13] Ke-Lin Du und Madisetti NS Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013. ISBN: 1447155718.

[Fra+98] Alejandro F Frangi, Wiro J Niessen, Koen L Vincken und Max A Viergever. „Multiscale vessel enhancement filtering“. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 1998, S. 130–137.

[Jäh12] Bernd Jähne. *Digitale Bildverarbeitung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-04951-4. DOI: 10.1007/978-3-642-04952-1. URL: http://link.springer.com/10.1007/978-3-642-04952-1.

[Jai19] Pawan Jain. *Complete Guide of Activation Functions*. Juni 2019. URL: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044.

[Lan18] Thom Lane. *Transposed Convolutions explained with... MS Excel!* Nov. 2018.

[Lev+09] Nataša Levičar, Ioannis Dimarakis, Catherine Flores, Evangelia I Prodromidi, Myrtle Y Gordon und Nagy A Habib. „Stem Cells and Organ Replacement“. In: *Artificial Organs*. Hrsg. von Nadey S Hakim. London: Springer London, 2009, S. 137–163. ISBN: 978-1-84882-283-2. URL: https://doi.org/10.1007/978-1-84882-283-2_9.

[Lom03] A T L van Lommel. *From Cells to Organs: A Histology Textbook and Atlas*. Springer Nature Book Archives Millennium. Kluwer Academic Pub., 2003. ISBN: 9781402072574. URL: https://books.google.de/books?id=EvYjLNKLu9sC.

[LSD15] Jonathan Long, Evan Shelhamer und Trevor Darrell. „Fully convolutional networks for semantic segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 3431–3440.

[LW10] Helen Liapis und Hanlin L Wang. *Pathology of solid organ transplantation*. Springer Science & Business Media, 2010. ISBN: 3540793437.

[Mah17] Jahnavi Mahanta. *Introduction to Neural Networks, Advantages and Applications*. Juli 2017. URL: https://towardsdatascience.comintroduction-to-neural-networks-advantages-and-applications-bda.

[McG+] John McGonagle, George Shaikouski, Christopher Williams, Andrew Hsu und Jimin Khim. *Backpropagation*. URL: https://brilliant.org/wiki/backpropagation/.

[MH06] Chris McIntosh und Ghassan Hamarneh. „Vessel crawlers: 3D physically-based deformable organisms for vasculature segmentation and analysis“. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. IEEE, 2006, S. 1084–1091. ISBN: 0769525970. URL: http://www.worldcat.org/title/ieee-computer-society-conference-on-computer-vision-and-pattern-recognition-2006-cvpr-2006-june-17-22-2006-new-york-ny-proceedings/oclc/836609931.

[Min19] Shervin Minaee. *20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics*. Okt. 2019. URL: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce.

[MMJ19] Jojo Moolayil, Moolayil und Suresh John. *Learn Keras for Deep Neural Networks*. Springer, 2019. ISBN: 1484242394.

[MNA16] Fausto Milletari, Nassir Navab und Seyed-Ahmad Ahmadi. „V-net: Fully convolutional neural networks for volumetric medical image segmentation“. In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, S. 565–571. ISBN: 1509054073.

[MP43] Warren S McCulloch und Walter Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *The bulletin of mathematical biophysics* 5.4 (1943), S. 115–133.

[Mur26] Cecil D Murray. „The physiological principle of minimum work: I. The vascular system and the cost of blood volume“. In: *Proceedings of the National Academy of Sciences of the United States of America* 12.3 (1926), S. 207.

[Rad17] Pranoy Radhakrishnan. *What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?* Aug. 2017.

[RFB15] Olaf Ronneberger, Philipp Fischer und Thomas Brox. „U-net: Convolutional networks for biomedical image segmentation“. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, S. 234–241.

[Sch+12] Matthias Schneider, Johannes Reichold, Bruno Weber, Gábor Székely und Sven Hirsch. „Tissue metabolism driven arterial tree generation“. In: *Medical image analysis* 16.7 (2012), S. 1397–1414.

[Sec09] Secretariat of the World Health Organization. *Human organ and tissue transplantation*. Techn. Ber. März 2009.

[Smi98] Brian Smits. „Efficiency issues for ray tracing“. In: *Journal of graphics tools* 3.2 (1998), S. 1–14.

[SSS18] Michael Schünke, Erik Schulte und Udo Schumacher. *PROMETHEUS Innere Organe: LernAtlas Anatomie*. Georg Thieme Verlag, 2018. ISBN: 3132420883.

[Tet+18] Giles Tetteh, Velizar Efremov, Nils D Forkert, Matthias Schneider, Jan Kirschke, Bruno Weber, Claus Zimmer, Marie Piraud und Bjoern H Menze. „Deepvesselnet: Vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes“. In: *arXiv preprint arXiv:1803.09340* (2018).

[TS97] Gerhard Thews und Robert F Schmidt. *Physiologie des Menschen*. Bd. 27. Springer-Verlag, 1997.

[Vin18] Vinhas, Adriano. „A gentle introduction to Neural Networks“. In: (Nov. 2018). URL: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-14e5c02bafe.

[Vin19] Adriano Vinhas. *The magic behind the perceptron network*. Feb. 2019. URL: https://towardsdatascience.com/the-magic-behind-the-perceptron-network-eaa461088367.

[VLF] VLFeat. *Plotting AP and ROC curves*. URL: http://www.vlfeat.org/overview/plots-rank.html.

[Wan+20] M Arif Wani, Farooq Ahmad Bhat, Saduf Afzal und Asif Iqbal Khan. *Advances in Deep Learning*. Bd. 57. Springer, 2020. ISBN: 9811367949.

[Wil+05] Amy Williams, Steve Barrus, R Keith Morley und Peter Shirley. „An efficient and robust ray-box intersection algorithm“. In: *Journal of graphics tools* 10.1 (2005), S. 49–54.

[Zam76] M Zamir. „Optimality principles in arterial branching“. In: *Journal of Theoretical Biology* 62.1 (1976), S. 227–251.