

# Einleitung 1

---

Myotuben sind mehrkernige Muskelzellfäden [1, 2]. Sie repräsentieren ein intermediäres Stadium der Muskelentwicklung, in dem sich die grundlegende Organisation der Muskelfaser bildet [3]. Menschliche Myotube-Kulturen [4] können für diverse Forschungszwecke als Modellsystem eingesetzt werden. Sie werden beispielsweise verwendet, um Muskelkrankheiten zu modellieren [5], Antworten auf neue Medikamente vorherzusagen [6] oder synthetische Muskeln [7] sowie Muskelregeneration [8] zu erforschen. In den meisten Fällen werden die Myotuben, ihre Zellkerne und zugehörige, umliegende Strukturen eingefärbt und unter dem Mikroskop analysiert [9, 10, 11]. Die Bilddaten entstehen dabei mit unterschiedlichen Herstellungsbedingungen, Färbungen und Aufnahmegeräten, was eine generalisierte Automatisierung erschwert [12, 13, 14, 15].

Im Zuge der vorliegenden Arbeit werden *TODO Beschreibung der Daten und Aufnahmebedingungen* Daten analysiert und die Ergebnisse automatisiert ausgewertet. Aus den Daten werden interpretierbare Merkmale wie die Zellkernanzahl, die Verteilung der Zellkernklassen und das Myotubenvolumen extrahiert. Diese Merkmale ermöglichen es, die Entwicklung der Myotuben ohne manuellen Aufwand zu vermessen und zu überwachen. Alle hierzu angewandten Methoden sind auf Basis quantitativer Vergleiche aktueller Forschung gewählt und bestmöglich auf die Anforderungen angepasst.

Um die Analyse biologischer Daten effizient und in großem Umfang durchführen zu können, sind Bildverarbeitungsprogramme notwendig [16], da die manuelle Auswertung sowohl anspruchsvoll als auch zeitintensiv ist [5]. Myotuben in Forschungsumgebungen ordnen sich in chaotischen Netzen mit Verschränkungen und Überkreuzungen an [17]. Aktuell sind Bildverarbeitungsmethoden nicht in der Lage, zuverlässig einzelne Myotuben in einem dreidimensionalen Bild zu erkennen und von ihrem Anfang bis zum Ende zu verfolgen [18]. Besonders die Bündel, die in der Entwicklung der Myotuben häufig entstehen, verhindern oft die getrennte Segmentierung der einzelnen Myotuben [7]. Außerdem stellen die dreidimensionalen Daten eine große Herausforderung für Hard- und Software dar [19, 20, 21, 22]. Besonders herausfordernd sind dabei die stark erhöhten Speicher- und GPU-Anforderungen und dass weniger Methoden sowie Datensätze etabliert sind [23, 24, 25]. Der Mehrwert einer dritten räumlichen Dimension kann allerdings die Ergebnisse wesentlich verbessern, was die Verarbeitung von 3D-Daten daher zu einem zentralen Forschungsaspekt macht [26, 27].

Das übergeordnete Ziel der vorliegenden Arbeit ist es, die Segmentierung einzelner Myotuben in dreidimensionalen Mikroskopiedaten zu ermöglichen. Aus den Segmentierungsmasken werden daraufhin morphologische Eigenschaften der einzelnen Myotuben gewonnen. Außerdem können die Ergebnisse genutzt werden, um den Status der Entwicklung

einzelner Myotuben, unter anderem, anhand der darin enthaltenen Zellkerne zu ermitteln. Deshalb ist ein weiteres Ziel, die Zellkerne zu segmentieren und zu klassifizieren. Für die Klassifikation soll im Rahmen dieser Arbeit Expertenwissen zeiteffizient erfasst und genutzt werden. In einer Applikation die keine Programmierkenntnisse benötigt sollen sowohl die Segmentierung, als auch die Annotation durch Experten, der Klassifikator-Methodenvergleich zur Anpassung an neue Daten und das Klassifikatortraining durchführbar sein. Aus diesen Zielen ergeben sich die folgenden Mehrwerte der vorliegenden Arbeit:

- Ein Vergleich etablierter Segmentierungsmodelle für Nuclei auf vorher ungesiehenen, dreidimensionalen Daten.
- Eine Labeling App, die zeiteffizient Expertenwissen über die Klassen von Nuclei in dreidimensionalen Daten erfasst.
- Ein neues Kriterium zum Vergleich von Segmentierungsmodellen im Hinblick auf die Eignung der entstehenden Segmentierungsmasken zur Extraktion interpretierbarer Merkmale.
- Ein Vergleich von Methoden des Vortraining, der Vorverarbeitung sowie etablierter Encoder und neuer Decoder für die Klassifikation von dreidimensionalen Zellkernen.
- Ein optimaler Klassifikator für die Nuclei der vorliegenden Arbeit.
- Eine Applikation mit automatischem Ablauf, die Nutzer\*Innen durch die Annotation und das Training von Klassifikatoren leitet und einen Methodenvergleich für den Klassifikator ermöglicht.
- Eine Methode, um das Klassifikationsergebnis zu nutzen, um die Instanzsegmentierung von Myotuben zu verbessern. (*NOTIZ: Das klappt sehr schlecht, soll das dann raus?*)

Die nachfolgende Ausarbeitung ist wie folgt strukturiert. In Kapitel 2 sind Grundlagen, relevante Literatur und der Stand der Technik beschrieben. Außerdem sind dort offene Probleme, sowie die Ansätze, die die vorliegende Arbeit verfolgt um sie zu lösen, dargestellt. Die Methodik (Kapitel 3) beschreibt das neue, praktische Konzept, das die Arbeit einführt, und das Kapitel 4 (Implementierung), wie diese Methoden umgesetzt werden. Im Kapitel 5 (Ergebnisse) werden ungewertet die Ergebnisse der durchgeführten Experimente dargestellt, im Kapitel 6 (Diskussion) werden dann diese Ergebnisse ausgewertet. Mit dem Kapitel 7 (Zusammenfassung) schließt die Ausarbeitung ab, legt kurz die gesamte Arbeit dar und liefert einen Ausblick für zukünftige Ziele. Der Code dieser Arbeit ist verfügbar unter: [github.com/DavidExler/Masterarbeit](https://github.com/DavidExler/Masterarbeit). (TODO Öffentlich stellen)

# Theorie 2

---

## 2.1 Überblick

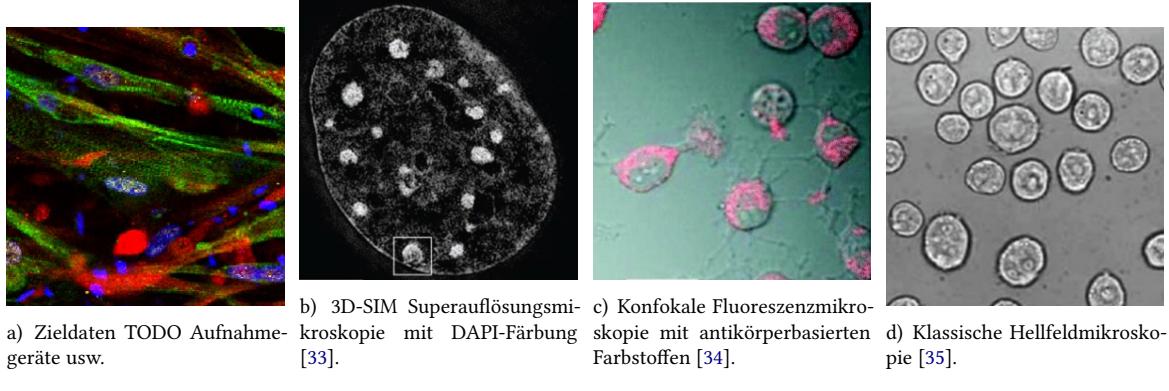
Im nachfolgenden Kapitel wird der theoretische Hintergrund der vorliegenden Thesis behandelt. Hierzu werden sowohl die Methoden verwandter Projekte und Studien als auch die Literatur zum aktuellen Stand der Technik beleuchtet. Dem theoretischen Hintergrund wird der Neuheitswert der Arbeit gegenübergestellt, um den Beitrag des vorliegenden Projekts zur Forschung zu verdeutlichen.

## 2.2 Methoden

### 2.2.1 Benchmark

Um die Leistungsfähigkeit der Applikation, die im Rahmen der vorliegenden Thesis erstellt wird, zu prüfen, sind umfangreiche Datensätze notwendig. Für jede isolierbare Aufgabe muss ein Datensatz gewählt werden, der Annotationen entsprechend dieser Aufgabe mitbringt. Die Herausforderung bei der Auswahl eines Datensatzes ist, dass die darin enthaltenen Daten (Quelldaten) den Daten, für die die Anwendung entworfen wird (Zieldaten), *ähnlich* sein müssen. So wird sichergestellt, dass sich die auf den Quelldaten gemessene Qualität der Anwendung sinnvoll auf die Zieldaten übertragen lässt [28]. Der Begriff *Ähnlichkeit* ist mehrdeutig und das Definieren von Merkmalen in Daten, die das Messen von *Ähnlichkeit* ermöglichen, ist anspruchsvoll. Metriken, die als Maß für *Ähnlichkeit* genutzt werden müssen, müssen maßgeschneidert zur Anwendung passen und sind bereits breit erforscht [29, 30, 31]. Daten können mithilfe passender Metriken *Ähnlichkeitsgruppen*, sogenannten *Domänen*, zugeordnet werden [32]. Beispiele für Domänenunterschiede in biomedizinischen Bildaufnahmen sind verschiedene Farb-Marker, Aufnahmegeräte oder Zoom-Stufen (siehe Abb. 2.1). Sind verschiedene Darstellungen gleicher Objekte in Bildern wie in Abb. 2.1 dargestellt.

Intuitiv gehören die sichtbaren Objekte zur Klasse *Zellkern*, aber der Stil unterscheidet sich stark. Sie unterscheiden sich also in ihrer Domäne. In der Bildverarbeitung ist es essenziell, die Domäne der Quelldaten im Hinblick auf die Aufgabe der Applikation zu beachten [36, 37]. Hierzu kann ein passender Datensatz gewählt werden oder eine Domänenanpassung durchgeführt werden [38, 39, 40]. Ghosh et al. definieren Domänenanpassung: "Gegeben Quell- und Ziel-Domänen  $D_s$  und  $D_t$  sowie die Aufgaben  $\tau_s$  und  $\tau_t$ , zielen Domänenadaptations-basierte Verfahren darauf ab, ein Modell mit Parametern  $\theta$  zu



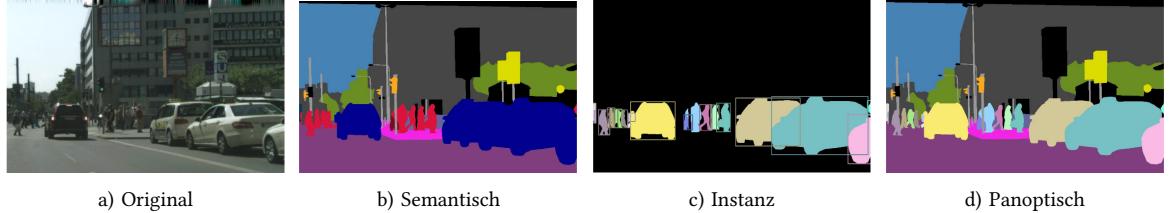
**Abb. 2.1** | Vier Aufnahmen von Zellen. Die Bilddomänen sind durch die verschiedenen Marker und Aufnahmetechniken klar zu unterscheiden [33, 34, 35].

erlernen, das für die Zielaufgabe geeignet ist, wenn  $D_s \neq D_t$  und  $\tau_s = \tau_t$ .” [41].

### 2.2.2 Segmentierung

Segmentierung ist die Aufgabe, Pixel mit semantischen Annotationen zu klassifizieren (semantische Segmentierung [42]), einzelne Objekte voneinander abzugrenzen (Instanzsegmentierung [43]) oder beides zu kombinieren (panoptische Segmentierung [44]) [45]. In Abb. 2.2 sind beispielhaft Annotationen der verschiedenen Arten von Segmentierung zu sehen [44].

Für die verschiedenen Segmentierungsarten werden Architekturen an die Aufgabe ange-



**Abb. 2.2** | Die verschiedenen Arten der Segmentierung. Links ist das Originalbild zu sehen, rechts folgend sind die zugeordneten pixelweisen Annotationen farblich eingezeichnet. Gleiche Farben bedeuten gleiche Annotationen. In der semantischen Maske sind gleiche Annotationen mehrerer Objekte von semantisch gleichen Klassen zu finden. In der Instanz-Maske ist jedem Objekt eine individuelle Annotation zugeordnet, ungeachtet der Klasse des Objekts. In der panoptischen Maske sind auch einzelne Objekte getrennt, den Annotationen verschiedener Klassen werden allerdings noch semantische Klassen zugeordnet [44].

passt [46, 47]. Modelle sind dabei in der Regel nach dem Vorbild des *U-Net* [48] aus einem Merkmalsextraktor (Encoder) und einem Vorhersagenetz (Decoder) aufgebaut [49]. Der Encoder nutzt beispielsweise Bildfaltungen mit Kernen, deren optimale Gewichte anhand von annotierten Daten gelernt werden, zur Merkmalsextraktion [50]. Dabei verringert der Encoder iterativ die Größe der Eingabe jeder Schicht des Netzes in X, Y und im dreidimensionalen Fall in Z Richtung, erhöht dabei aber die Informationstiefe pro verbleibendem Pixel, bis ein hochdimensionaler Merkmalsvektor übrig bleibt. Der Decoder hebt, meist

durch transponierte Bildfaltungen [51, 52], die räumliche Auflösung schrittweise wieder an, indem er die Bilddimensionen vergrößert und die Merkmalskanäle gleichzeitig reduziert [53]. Über sogenannte Skip-Connections [48] werden dabei Merkmale aus den entsprechenden Encoder-Schichten mit den Decoder-Stufen verknüpft, sodass sowohl globale Kontextinformationen als auch feine Strukturen für die Segmentierung erhalten bleiben [54, 55].

TODO: 3D-bzw 2.5D - Welche Methoden werden speziell für 3D-genutzt?

### 2.2.3 Klassifikation

Klassifikation beschreibt das Zuordnen einer Kategorie oder Klasse zu der eine gegebene Stichprobe gehört [56]. Hierzu werden die Merkmale des Objekts, das in der Stichprobe präsentiert wird, durch Beobachtung oder Messung gewonnen [57]. Nach wiederholter Extraktion der Merkmale von Objekten verschiedener Klassen werden Muster in den Merkmalen gesucht, um Regeln für die Zuweisung von Objekten zu Klassen auf Basis der Muster festzulegen [58, 59]. Sowohl die Algorithmen zur Merkmalsextraktion als auch zum Ableiten der Muster und Regeln können mit unterschiedlich hohem Rechenaufwand, Abstraktionsgrad und Maß an Generalisierbarkeit implementiert werden [60]. Zur Merkmalsextraktion werden klassisch beschreibende Eigenschaften des Objekts berechnet und kombiniert [61]. Als Eigenschaften eignen sich beispielsweise die Verteilung der Farbkanäle, eine Charakterisierung der Textur, die Fläche des Objekts [62, 63]. Eine weitere verbreitete Eigenschaft ist eine Kombination von Parametern der Fourier-Entwicklung einer Kontur, die aus der diskreten komplexen Zahlenfolge

$$c[n] = x[n] + i \cdot y[n], n = 0, \dots, N - 1 \quad (2.1)$$

mithilfe der diskreten Fourier-Transformation

$$F[k] = \sum_{n=0}^{N-1} c[n] \cdot e^{-2\pi i \frac{kn}{N}}, k=0, \dots, N-1 \quad (2.2)$$

gebildet werden [64, 65]. Hierbei sind  $x[n]$  und  $y[n]$  die Koordinaten des  $n$ -ten von  $N$  equidistanten Stützpunkten entlang der Kontur des Objekts,  $c[n]$  ihre komplexe Darstellung und  $F[k]$  die Fourier-Transformation der komplexen Darstellungen ist. Die Ergebnisse der Fourier-Transformationen mehrerer Stützpunkte werden dann als Eigenschaft verwendet. Merkmalsvektoren werden häufig abstrahiert und in ihrer Dimension reduziert, beispielsweise durch eine Principal Component Analysis [66, 67]. Sind keine Annotationen verfügbar, werden diese Metriken zum Clustering verwendet [58, 68]. Wenn nur wenige Annotationen vorhanden sind, können semi-supervised-Verfahren angewandt werden, die besonders die Ähnlichkeit der Stichproben ohne Annotationen herausarbeiten [69, 70]. Eine prominente Methode des semi-supervised-Lernens ist das label spreading, das mithilfe einer Kernfunktion [71, 72] die Dimension von Merkmalsvektoren ändert und in einen alternativen Merkmalsraum transformiert [73]. Verschiedene Kernfunktionen wie die

### Radiale-Basis-Funktion [74]

$$\phi(x, y) = \exp(-\gamma \|x - y\|^2), \quad \gamma > 0 \quad (2.3)$$

werden für das label spreading eingesetzt [75]. Hierbei sind  $x, y \in \mathbb{R}^d$  die Koordinaten der Stichprobe im Merkmalsraum,  $\gamma$  ein Parameter, der die Breite der Radialbasisfunktion steuert und  $\phi(x, y)$  der Wert der Radialen-Basis-Funktion. Die meist genutzten Methoden der Klassifikation sind logik-basierte Ansätze wie Entscheidungsbäume, Perzepron-basierte Ansätze wie neuronale Netze, statistische Ansätze wie Bayes'sche Netzwerke oder Nächster-Nachbar-Verfahren und Support Vector Maschinen [76]. Diese Methoden basieren direkt auf Ähnlichkeiten zwischen den Merkmalen von unbekannten Stichproben und Stichproben mit bekannter Klasse [77]. Moderne Anwendungen nutzen zur Merkmalsextraktion verschiedene Deep-Learning-basierte Methoden [78]. Vor allem Convolutional Neural Networks (CNNs) [79] und Vision Transformers (ViTs) [80] sind in der Lage, aus Bildern aussagekräftige, abstrakte Merkmale zu generieren [81]. Ein Netz, das zur Merkmalsextraktion eingesetzt wird, wird als **Encoder** bezeichnet. Als **Decoder** wird der zusammenfassende Teil des Klassifikators bezeichnet, er gibt einen Zuverlässigkeitswert für jede Klasse aus. Der State-of-the-Art für den Decoder ist ein neuronales Netz, das auf Basis der abstrakten Merkmale des Encoders eine Zuverlässlichkeit für jede Klasse ausgibt [82]. Hierzu lernt in der Regel ein Multi-Layer-Perzepron auf Basis von Trainingsdaten mit zugehöriger Annotation den Zusammenhang zwischen den Merkmalen und der assoziierten Klasse [83].

Für das Training von Klassifikatoren sind ein Loss-Funktion und häufig Vorverarbeitungsmethoden notwendig. Der Cross-Entropy Loss [84]

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^N -\log p(\tilde{y} = \tilde{y}_n | x_n, \theta) \quad (2.4)$$

ist die etablierte Loss Funktion für das Training von Klassifikatoren [85, 86]. Hierbei ist Loss  $L$  abhängig von der Annotation  $\tilde{y}$  und der Vorhersage  $\tilde{y}_n$ , die von den Eingangsdaten  $x_n$  und den Parametern des Modells  $\theta$  bestimmt wird. Ein Problem der Cross-Entropy Loss Minimierung ist ihre Anfälligkeit für Rauschen der Annotationen. Viele verschiedene Ansätze in der Forschung gehen dieses Problem an [87, 88, 89]. Eine häufig genutzte Methode ist die Minimierung des Generalized Cross Entropy Loss [90]

$$\arg \min_{\theta, w \in [0,1]^n} \sum_{i=1}^n w_i \mathcal{L}_q(f(x_i; \theta), y_i) - \mathcal{L}_q(k) \sum_{i=1}^n w_i, \quad (2.5)$$

wobei  $\mathcal{L}_q$  die generalisierte Form des Cross-Entropy Loss beschreibt, die durch den Parameter  $q$  reguliert wird. Dieser kontrolliert den Einfluss fehlerhafter oder unsicherer Trainingsbeispiele. Die Gewichte  $w_i \in [0, 1]$  dienen der Gewichtung einzelner, besonders

unsicherer Trainingsinstanzen. Das Modell  $f(x_i; \theta)$  gibt die Vorhersage für die Eingabe  $x_i$  basierend auf den Modellparametern  $\theta$  aus, während  $y_i$  die entsprechende Zielannotation ist.

Bilineare Interpolation ist ein Verfahren zur Bildvorverarbeitung, das die Dimension eines Bilds erhöht, indem Werte für neue Pixel zwischen bestehenden geschätzt werden [91]. Zur Schätzung des neuen Wert wird dabei ein gewichtetes Mittel aus den Vier benachbarten Pixeln genommen:

$$\hat{f}(x, y) = (1 - p)(1 - q)f_{i,j} + p(1 - q)f_{i+1,j} + (1 - p)qf_{i,j+1} + pqf_{i+1,j+1}, \quad (2.6)$$

wobei  $\hat{f}(x, y)$  der neue Wert,  $p, q \in [0, 1]$  die relativen Abstände zu den Nachbarpixeln,  $i, j$  die Indizes der Nachbarpixel und  $f$  die Intensität der Nachbarpixel sind.

Normierungsmethoden werden während dem Training eingesetzt, um die Daten zu regulieren und Signale nicht unverhältnismäßig groß oder verschwindend klein werden zu lassen. Batch Normalization normalisiert die Eingaben einer Schicht über ein Mini-Batch, indem für jedes abstrakte Merkmal der Schicht, das bei der Dimensionsreduktion des Bilds entsteht, eine Transformation durchgeführt wird [92]. Für die Transformation wird der Mittelwert des Mini-Batches vom Wert abgezogen und durch die Standardabweichung geteilt. Layer Normalization verfolgt einen ähnlichen Ansatz, berechnet Mittelwert und Varianz aber pro Schicht von allen Neuronen [93]. Beide stabilisieren das Training tiefer neuronaler Netze und lassen die Normalisierung als Bestandteil der Modellarchitektur lernen, statt sie nur als Vorverarbeitungsschritt durchzuführen [94, 95].

Als Vergleichsmetrik der Klassifikation wird für gewöhnlich die Genauigkeit des getesteten Netz auf den annotierten Daten verwendet:

$$\text{Genauigkeit} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}\{\hat{y}_i = y_i\}, \quad (2.7)$$

wobei  $N$  die Anzahl der Vorhersagen,  $\hat{y}_i$  die Vorhersage des Klassifikators und  $y_i$  die Annotation ist.

*TODO: 3D-bzw 2.5D - Welche Methoden werden speziell für 3D-genutzt, gibt es Methoden zur Anpassung?*

## 2.3 Literaturrecherche

### 2.3.1 Benchmark

Da das manuelle Erstellen der Annotationen für die Segmentierung von Zelldaten mit erheblichem manuellem Aufwand verbunden ist und zusätzlich Expertenwissen voraussetzt, sind Datensätze hierfür selten. Einige prominente Datensätze mit Annotationen für eine Instanzsegmentierung, deren Domänen zu den Zieldaten der Anwendung dieser Arbeit *ähnlich* sind, sind:

- LiveCell [96], ein manuell annotierter und Experten-validierter Datensatz aus 5.239

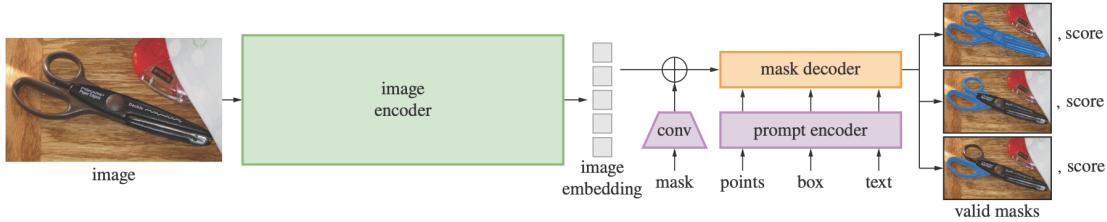
2D-Bildern. Die Daten sind mit Phasenkontrastmikroskopie gesammelt und enthalten 1.686.352 individuelle Zellen von acht verschiedenen Zelltypen.

- YeaZ [97], ein zweiteiliger Datensatz von Hefe-Zellen aus 87 Phasenkontrast-Bildern mit insgesamt 10.422 Zellen und 614 Hellfeld-Bildern mit insgesamt 3.841 Zellen in 6 Beleuchtungsstufen aufgenommen. Die Annotationen sind semi-maniuell erstellt, da die Phasenkontrast-Bilder manuell, und die Lichtfeld-Bilder aus den Phasenkontrast-Segmentierungsmasken annotiert wurden.
- DeepBas [98, 99], ein Datensatz von *B. subtilis strain SH130* Bakterien. Er besteht aus Weitfeldaufnahmen (Fluoreszenz), aufgenommen mit einem inversen Mikroskop, bestehend aus sieben manuell annotierten Bildern mit je zwischen 46 und 335 Zellen.
- die Cell Tracking Challenge [100], eine Sammlung aus 13 Datensätzen verschiedener Mikroskopimodalitäten, die sich zum Messen der Segmentierungs- und Verfolgungsfähigkeiten für verschiedene Zelltypen eignen.
- MoNuSeg [101], eine Zusammenstellung manuell annotierter Gewebeabschnitte von sieben verschiedenen Organen. Über 21.000 Zellen sind pro Bild in den 30 Bildern mit verschiedenen Färbungen und Aufnahmetechniken verteilt.
- TissueNet [102] ein umfassender Datensatz mit über 1.000.000 Zellen mit diversen Gewebearten und unterschiedlichen Aufnahmetechniken.
- S-BIAD1518 [103, 104], ein Datensatz der neben manuell annotierten Bildern von acht verschiedenen Zellarten sind synthetisch erzeugte Daten enthält. Mit Hilfe von SpCycleGAN [105] wurden dazu auf Basis von simulierten Annotationen Bildern generiert, die anstreben die Merkmale der realen Bilder zu reproduzieren. Es handelt sich hierbei um 3D-multispektrale Daten, aufgenommen mittels Fluoreszenzbildgebung.

Aufgrund des geringen Volumens frei zugänglicher Daten sind diese Sammlungen auch für das Training von Segmentierungsnetzen begehrte. Neben annotierten Datensätzen bietet die Literatur auch Methoden zum eigenständigen Erzeugen domänenspezifischer Datensätze [106, 107, 108, 109, 110]. Beispielsweise können 3D-Trainingsdaten mit realistischer Zellform und -ausrichtung und umgebenden Markern synthetisch erzeugt und durch ein Generative Adversarial Network an eine gewünschte Bilddomäne angepasst werden [111].

### 2.3.2 Segmentierungsmodelle

Foundation-models sind für viele moderne Künstliche Intelligenz (KI)-Anwendungen unerlässlich [112]. Sie werden zunächst für allgemeine Aufgaben vorgenutzt und anschließend auf spezifische Anwendungen angepasst (fine-tuning), meist unter Einfrieren von



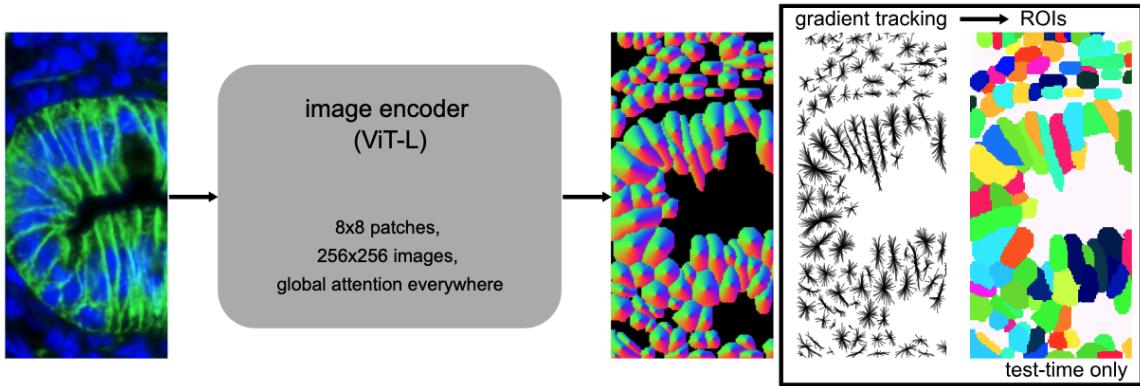
**Abb. 2.3 | [119].** Architektur des **SAM**. Eingabebilder werden durch einen Bildencoder zu Repräsentationen umgeformt. Zusätzliche optionale Hinweise auf das zu segmentierende Objekt werden durch Bildfaltungen oder einen Prompt Encoder repräsentiert. Anschließend prädiziert der Decoder mehrere mögliche Masken und zugehörige Zuversichtlichkeiten.

Teilen der Gewichte [113]. Auch Segmentierungsmodelle profitieren stark von umfangreichem Vortraining [114]. In der aktuellen Forschung werden verschiedene foundation models für Segmentierung angewandt [115, 116, 117, 118]. Ein prominentes Exemplar ist das Segment Anything Model (**SAM**) von Meta AI [119] (siehe Abb. 2.3). Es besteht aus einem Bild-Encoder, einem Prompt-Encoder und einem Masken-Decoder. Als Bild-Encoder dient ein Vision Transformer [120], mit Vortraining als Masked Auto Encoder [121] und zusätzlichem Training für höhere Bildauflösung. Der Prompt Encoder ist mehrstufig. Ein angelernter Positional Encoder generiert Repräsentationen aus Positions-Nutzereingaben wie Punkten und Boxen. Für textuelle Prompts wird der Encoder des CLIP [122] Models genutzt. Außerdem werden Bildfaltungen als Encoder auf Masken-Nutzereingaben angewandt. Mithilfe dieser Encoder wird dem Modell eine Repräsentation von dem zu segmentierenden Bild und optionalen, manuellen Hinweisen auf das erwünschte Ergebnis gegeben, die bereits semantische Informationen und abstrakte Bildmerkmale beinhalten. Aus diesen Repräsentationen generiert dann der Decoder mehrere mögliche Masken mit zugehörigen Zuversichtlichkeiten, aus denen ein finales Segmentierungsergebnis ausgewählt wird.

**SAM** wurde bereits für viele explizite Mikroskopie-Zelldaten Anwendungen angepasst [123, 124, 125]. Auch für bestehende biologische Segmentierungsanwendungen, wie etwa Cellpose[126], wurde **SAM** auf Zelldaten angepasst [127]. Dieser *Fine-tune* nennt sich CellposeSAM. Er kombiniert den Bild-Encoder von **SAM** mit dem *Flow*-Segmentierungsansatz von Cellpose. Dabei generiert der Bild-Encoder direkt Vekotoren, die Zwischenrepräsentationen, die sogenannten *Flows*, darstellen. Diese *Flows* werden pixelweise zu einem Gradientenfeld überführt. Mithilfe der Gradienten werden Objektinstanzen vorhergesagt. Abb. 2.4 zeigt diesen Ablauf als Diagramm.

Deepcell [128, 129, 130] bietet weitere Zellsegmentierungsmodelle. Das Deepcell-Caliban-Modell [131] nutzt als Encoder eine EfficientNetV2L-Architektur [132], an deren Ausgabeschichten (C1C5) eine Pyramiden-Struktur zur Merkmalsfusion (P1P7) angeschlossen ist. Eine Besonderheit des Netzes ist, dass Eingabebildern zusätzlich Koordinatenkarten hinzugefügt werden. Als Decoder dienen drei Segmentierungsköpfe, die verschiedene Transformationen der gelabelten Trainingsmasken vorhersagen.

In der Literatur sehr verbreitet ist außerdem das nnU-Net [133], ein Segmentierungsframe-



**Abb. 2.4** | Ablauf des CellposeSAM-Modells. Eingabebilder werden durch einen Bild-Encoder (ViT-L) direkt zu sogenannten *Flows* umgeformt, einer Repräsentation von vorhergesagten Objektmerkmalen, deren Wert von der relativen Position innerhalb des detektierten Objekts abhängt. Die Gradienten der Flows werden verfolgt (gradient tracking) und aus dem daraus entstehenden Gradientenfeld werden Segmentierungsinstanzen (ROIs) vorhergesagt [127].

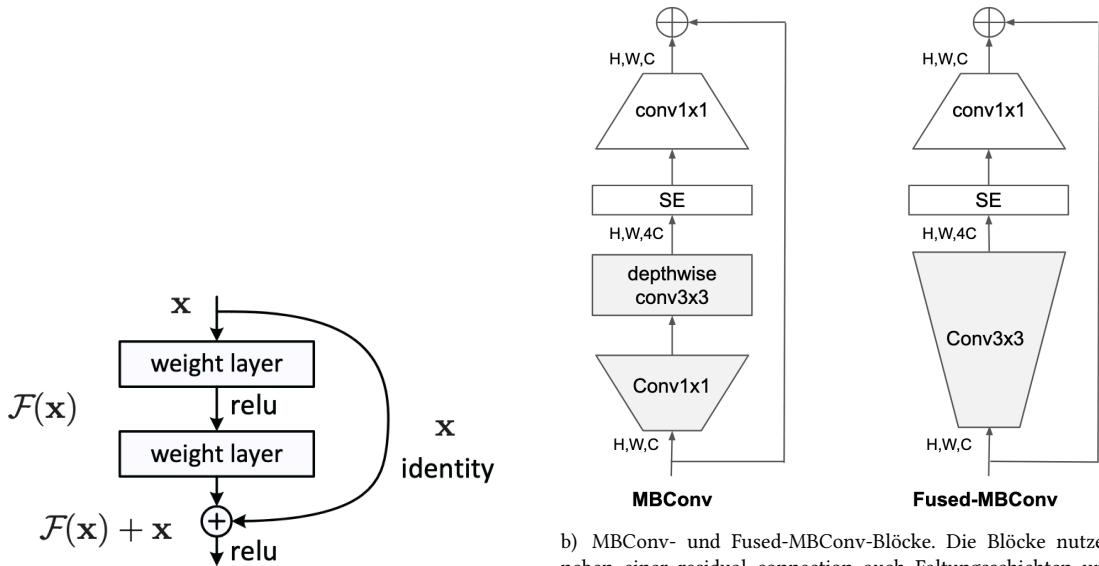
work, das sich automatisch an neue biomedizinische Aufgaben anpasst. Es konfiguriert Vorverarbeitung, Netzwerkarchitektur, Training und Nachbearbeitung dynamisch auf Basis der Eigenschaften des jeweiligen Datensatzes. Die Leistungsfähigkeit des Ansatzes ergibt sich nicht aus einer neuen Architektur oder Lernmethode, sondern aus der konsequenten Automatisierung und Systematisierung von Entwurfsentscheidungen.

### 2.3.3 Klassifikator

Für Klassifikatoren werden in der Regel nur Encoder vorgenommen, der Decoder muss an die Klassen des vorliegenden Problems angepasst werden [81, 113, 134]. State-of-the-Art für Bild-Encoder sind **CNNs** oder **ViTs**, die auf dem ImageNet [135] Datensatz vorgenommen werden [136, 137]. Klassifikatoren profitieren stark von ImageNet-Vortraining [138, 139]. **ResNet** ist ein Residual Neural Network, ein **CNN** mit sogenannten residual connections [140]. Diese residual connections verbinden den Ein- und Ausgang modularer Faltungsschichten und verbessern die Leistungsfähigkeit tiefer neuronaler Netze [141, 140, 142] (siehe Abb. 2.5a). In ihrem Paper stellen die Autoren fünf unterschiedlich tiefe Varianten der **ResNet**-Architektur vor. Jede Variante enthält fünf Blöcke mit residual connections. Die Blöcke bestehen aus Faltungen mit verschiedenen Kernelgrößen und Strides, Batch normalization [143] und der ReLU [144] Aktivierungsfunktion.

**EfficientNetV2** ist ein Nachfolger der EfficientNet-Modellfamilie [145, 132]. Die Architektur basiert auf modularen Blöcken von Bildfaltungsoperatoren mit besonders kleinen Faltungskernen und Squeeze-and-Excitations, genannt MBConv [146, 145] und Fused-MBConv [147] (siehe Abb. 2.5b).

**ConvNeXt** [148] ist eine **CNN**-Modellfamilie mit besonders großen Faltungskernen. Die **ConvNeXt**-Architektur umfasst fünf modulare Blöcke mit Faltungen und residual connections, wie die ResNet-Architektur [140]. Allerdings verändert dabei **ConvNeXt** einige Details der ResNet Architektur, wie beispielsweise die GeLU Aktivierungsfunktion [149]



a) Residual connection. Der Eingang des modularen Blocks ist mit dem Eingang direkt verbunden [140].

b) MBConv- und Fused-MBConv-Blöcke. Die Blöcke nutzen neben einer residual connection auch Faltungsschichten und Squeeze-and-Excitation, um die Datendimension zu erhöhen [132].

Abb. 2.5 | Diagramme a) der Residual Connections und b) MBConv-Blöcke und Fused-MBConv-Blöcke.

und Layer normalization [93].

Swin Transformer [23] ist eine beliebte Modellfamilie von ViTs. Ihr Nachfolger, die **Swin Transformer V2** [150], vergrößert die Modelle noch. Die Architektur kombiniert Bildausschnitte mit einem Positions-Bias. Hierzu werden ein Bildfenster  $z$  und dessen relativen Koordinaten im Bild  $\Delta x$  und  $\Delta y$  in einem Attention-Mechanismus zusammengeführt. Die Positionen werden in einem MLP-Netz verarbeitet, während das Bildfenster mit drei verschiedenen Gewichtsmatrizen multipliziert wird. Mithilfe einer Kosinus-Ähnlichkeitsfunktion, der Softmax-Funktion [151] und elementweiser Multiplikation sowie Addition werden diese Ergebnisse in einen Merkmalsraum überführt. Zwei Layer normalization [93] Schichten, ein weiteres MLP-Netz und residual connections vervollständigen anschließend den modularen **Swin Transformer V2** Block. Dieser Aufbau ist in Abb. ?? dargestellt.

## 2.4 Offene Probleme

Einzelne Myotuben können durch kein Segmentierungsmodell der Literatur instanzsegmentiert werden. Selbst für Experten sind in dichten Strukturen Instanzen von Myotuben nicht immer eindeutig trennbar. Nicht viele Segmentierungsmodelle für Nuclei sind erhältlich, besonders für dreidimensionale Daten. Die verfügbaren Modelle verhalten sich unterschiedlich bei verschiedenen Datensätzen und ihre Eignung für bestimmte Aufgaben muss für jede Anwendung individuell geprüft werden. Außerdem gibt es für die spezifischen Aufnahmebedingungen und Marker der **TODO Daten kurz Beschreiben** Daten keinen angepassten Klassifikator. Der Erfolg von einem Übertrag verschiedener vorgeführter Encoder und etablierter Methoden auf die vorliegenden Daten ist unvorhersehbar,

da sich die gelernten Merkmalsräume eventuell nicht für die Klassifikation der neuartigen dreidimensionalen Daten eignen. Eine weitere Fragestellung ist deshalb, wie Methoden der Klassifikation mit der Kombination der Marker umgehen. Für jeden Datensatz mit neuen Zellkernklassen und Aufnahmebedingungen muss für optimale Klassifikatorleistung nicht nur ein neues Modell trainiert werden, sondern auch Methoden- und Hyperparameter-optimierung durchgeführt werden. Des Weiteren ist der Umgang mit dreidimensionalen Daten, besonders in Umgebungen ohne große Rechenleistung ein offenes Problem. Verschiedene Lösungen existieren, um dreidimensionale Daten mit Expertenwissen zu versehen. Diesen Lösungen fehlt bislang, ein Arbeitsablauf der Daten unmittelbar segmentiert und vorbereitet, um relevante Bildausschnitte direkt aus dem Datensatz zu extrahieren, sodass Experten ausschließlich annotieren müssen.

## 2.5 Zielsetzung

Im Zuge der vorliegenden Arbeit soll die automatische Extraktion von interpretierbaren Eigenschaften der Myotubenkulturen ermöglicht werden. Zu diesem Ziel bearbeitet die vorliegende Arbeit Zwischenziele und liefert Folgendes:

- Es soll ein Segmentierungsmodell gefunden werden, das die Eigenschaften der Nuclei in den vorliegenden, dreidimensionalen Daten besonders wenig durch Segmentierungsfehler verfälscht. Dazu wird ein neues Bewertungskriterium für Instanzsegmentierung eingeführt und auf einige etablierte Modelle angewandt.
- Das Segmentierungsmodell soll dann genutzt werden, um außerdem einen Ablauf zu schaffen, in dem Experten die Klassen der Nuclei besonders zeiteffizient annotieren können, um einen Klassifikator zu trainieren. Durch das Anreichern der dreidimensionalen Daten durch die Segmentierungsmasken und durch automatisches Fokussieren einzelner Nuclei soll sowohl die Rechenzeit optimiert werden als auch der Aufwand einzelne Nuclei entlang drei Dimensionen zu suchen, eliminiert werden. Hierzu wird eine neue Anwendung entwickelt, die 3D-Zelldaten liest und dann eine Oberfläche zum Annotieren bereitstellt.
- Die entstehenden Annotationen sollen direkt in einen Trainingsablauf für Klassifikatoren eingebunden sein. Hierbei müssen Klassifikatoren mit dreidimensionalen Daten von variierender Tiefe umgehen können. Ein ausführlicher Methodenvergleich verschiedener Encoder, Decoder, Vorverarbeitungsmethoden und auch Vortrainingsmethoden soll für Nutzer\*Innen ohne Programmierkenntnisse ermöglicht werden. Dazu wird eine neue Anwendung entwickelt, die die zuvor erstellten Annotationen und Segmentierungsmasken nutzt um einen Klassifikator zu trainieren. Nutzer\*Innen können in einer grafischen Oberfläche verschiedene Methoden zum Vergleich auswählen und in einem automatischen Prozess werden Klassifikatoren aller Kombinationen trainiert und verglichen.

- Außerdem sollen die ausgelesenen Eigenschaften der eingegebenen Zelldaten leicht zugänglich sein. In einer weiteren neuen Anwendung werden automatisch die Vorfahrtswerte des Klassifikators mit dem besten Ergebnis im Methodenvergleich genutzt, um Nutzer\*Innen Graphen mit den Eigenschaften der Zellkultur darzubieten.
- Zuletzt sollen alle neu entwickelten Module zu einer Gesamtanwendung zusammengefasst und getestet werden. In einer Parameterbestimmung wird das Optimum explizit für die vorliegenden Aufnahmen mithilfe der neuen Anwendung bestimmt.



# Neues Konzept 3

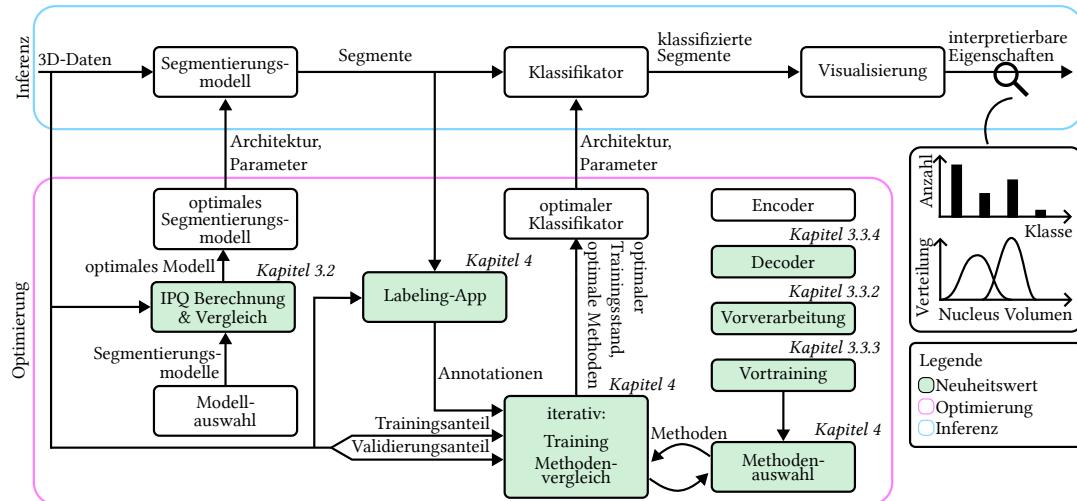
---

## 3.1 Überblick

Das nachfolgende Kapitel beschreibt und diskutiert das angewandte Konzept der vorliegenden Thesis im Detail. Es behandelt die selbstentwickelten Beiträge zu den Methoden. Die Methodik wird in einer modularen Anwendung umgesetzt die 3D-Daten als Eingabe annimmt und interpretierbare Eigenschaften wie die Verteilung der Klassen und Volumen der anwesenden Nuclei ausgibt. In Kapitel 4.4 ist die praktische Umsetzung dieser modularen Anwendung beschrieben. Diese Anwendung kann regulär angewandt (Inferenz) oder optimiert werden (Optimierung). Zur Optimierung werden die 3D-Daten einem Ablauf für den Vergleich der Segmentierungsmodelle oder der Klassifikatormethoden zur Verfügung gestellt. Zuerst wird dazu das neu eingeführte Bewertungskriterium für Segmentierungsmodelle Injektive Panoptische Qualität (IPQ) für verschiedene Modelle berechnet (siehe Kapitel 3.2). Dieses Bewertungskriterium quantifiziert die Eignung der Segmentierungsmodelle zur Extraktion der gewünschten Eigenschaften. Mithilfe der IPQ-Werte wird das beste Segmentierungsmodell für die Anwendung gewählt. Die Architektur und die Parameter des optimalen Model werden in den Inferenzablauf eingesetzt. Um die Klassifikatormethoden zu optimieren werden die Daten und die Segmente in die neu entwickelte Labeling-App eingegeben. Die Labeling-App ermöglicht das zeiteffiziente Annotieren von Nuclei, indem die relevanten Bildausschnitte automatisch anhand der Segmente extrahiert werden. In Kapitel 4.4.2 ist die Umsetzung der Labeling-App beschrieben. Mit den erstellten Annotationen und den 3D-Daten werden verschiedene Klassifikatoren trainiert. Diese Klassifikatoren ergeben sich aus Kombinationen der verfügbaren Klassifikatormethoden. Die Klassifikatormethoden sind beschrieben in Kapitel 3.3 und umfassen 1. Encoderarchitekturen, 2. Decoderarchitekturen, 3. Vorverarbeitungsmethoden und 4. Vortrainingsmethoden. Unter den Methoden sind sowohl etablierte, als auch neu entwickelte Ansätze. Anhand der Genauigkeit der trainierten Klassifikatoren auf einem separaten Validierungsanteil des Datensatz werden die Methoden verglichen und eine optimale Konfiguration ausgegeben. Die Architektur und die Parameter dieser Konfiguration werden dann in den Inferenzablauf der Applikation eingesetzt. Am Ende des Ablaufs werden die klassifizierten Segmente genutzt, um verschiedene Grafiken zu erzeugen, die die interpretierbaren Eigenschaften visualisieren.

Dieses Kriterium wird eingeführt, um die Eignung von Segmentierungsmodellen zum Extrahieren interpretierbarer Merkmale, wie Zellkernanzahlen oder -volumina zu bewerten. Darauf folgen Beschreibungen von Klassifikatormethoden. Die Methoden umfassen Architekturen von Encodern und Decodern, sowie Methoden der Vorverarbeitung und

des Vortrainings. Es werden sowohl Anwendungen bestehender Methoden, als auch neu entwickelte Ansätze eingeführt. Ziel dieser Methoden ist es, die Optimierung eines Klassifikators für individuelle, dreidimensionale Zelldaten in einem standardisierten Ablauf zu ermöglichen. Verschiedene Kombinationen der eingeführten Methoden werden beispielhaft auf die Daten der vorliegenden Arbeit angewandt und die dabei entstehenden Klassifikatoren werden bezüglich ihrer Genauigkeit auf einem separaten Validierungsanteil des Datensatzes verglichen.

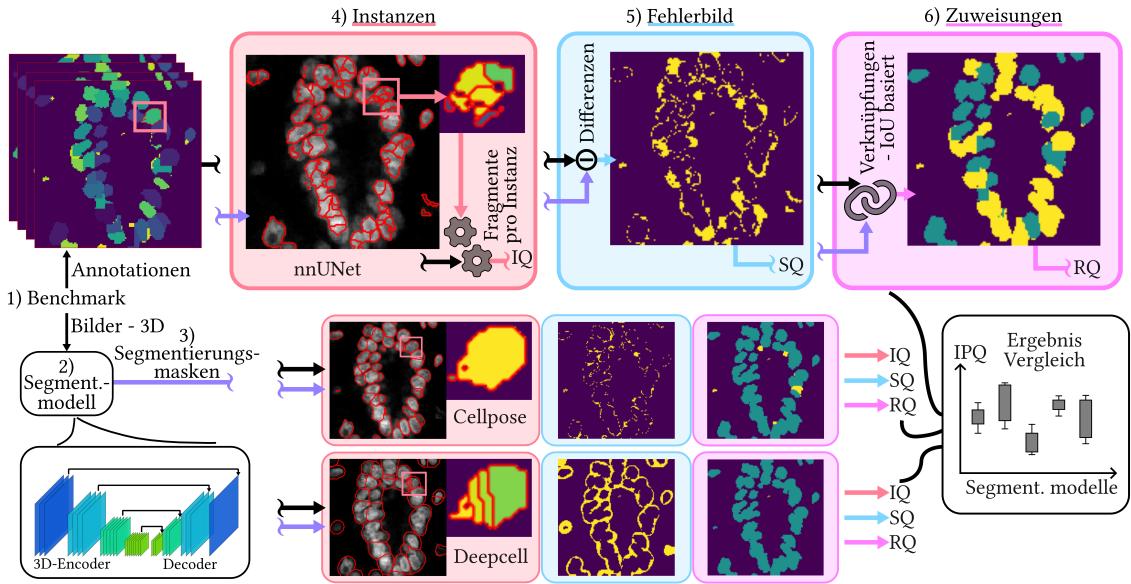


**Abb. 3.1 |** Aufbau der Anwendung der vorliegenden Arbeit. Die Anwendung kann regulär angewandt (Inferenz) oder optimiert werden (Optimierung). Die Optimierung ist zweiteilig. Zur Optimierung der Segmentierungsmodelle wird das neu entwickelte Injektive Panoptische Qualität (IPQ) Bewertungskriterium eingesetzt. Für den Klassifikator werden Kombinationen verschiedener Encoder, Decoder, Vorverarbeitungsmethoden und Vortrainingsmethoden iterativ trainiert und verglichen.

## 3.2 Injektive Panoptische Qualität

Zur Wahl des Segmentierungsmodells wird ein neues Bewertungskriterium eingeführt und auf einem annotierten Datensatz getestet. Als Datensatz wird aus den in Kapitel 2.3.1 vorgestellten Benchmarks der S-BIAD1518 [103, 104] genutzt, da dieser nicht in den Trainingsdaten eines zu testenden Segmentierungsnetz vorkommt. Im Gegensatz zu selbstentwickelten synthetischen Daten weicht die Bilddomäne dieses Benchmarks zwar stärker von der Domäne der Zieldaten ab, aber dafür sind die Daten an eine Veröffentlichung mit standardisiertem Peer-Review-Prozess gebunden.

Die Aufgabe des Bewertungskriterium ist es, zu quantifizieren, wie gut sich eine vorliegende Instanzsegmentierung eignet, um reale Eigenschaften einer Aufnahme, wie die Zellkernanzahl, Größe der Zellkerne und Verteilung der Zellkernarten auszuwerten. Das neu entwickelte Kriterium ist eine Abwandlung der Panoptic Quality (PQ) [44], die hier **IPQ** genannt wird. Durch standard **PQ** wird die Intersection over Union (IoU) für individuelle Instanzen bewertet und es werden **FP** sowie **FN** Detektionen bestraft. Zusätzlich sollen



**Abb. 3.2 | IPQ Visualisierung** - Das obere Ablaufdiagramm stellt den Prozess dar, durch den das Segmentierungsnetz gewählt wird, das für die Anwendung der vorliegenden Arbeit eingesetzt wird. Ein peer-reviewed Benchmark-Datensatz aus dreidimensionalen Bildern (1) von diversen Zellkulturen mit dazugehörigen Ground Truths wird links eingegeben. Das zu bewertende Segmentierungsmodell (2) führt eine Inferenz für die Bilder des Benchmark aus, um Segmentierungsmasken (3) bereitzustellen. Die entstehenden Masken werden dann zur Berechnung der neu eingeführten IPQ (siehe Kapitel 3.2) eingesetzt. Der Ablauf der Bewertung ist dreigeteilt. Aus jeder Maske werden zuerst die einzelnen Fragmente extrahiert, die sich mit einer einzelnen Instanz der Annotation überlagern (4). Außerdem wird ein Fehlerbild als die logische XOR-Fläche der Maske und der Annotation dargestellt (5), als Platzhalter für die Berechnung der Intersection over Union. Zuletzt wird ein Zuweisungsbild erstellt, das die True Positives (TPs), False Positives (FPs) und False Negatives (FNs) festhält (6). Durch Vergleiche der Ergebnisse verschiedener Modelle kann dann das optimale Segmentierungsnetz für die Anwendung gewählt werden.

durch einen neuen Faktor Verletzungen der injektiven Abbildung von segmentierten Nuclei auf die Instanzen der Annotation negativ bewertet werden, da die genaue Anzahl der Nuclei eine bedeutungsvolle Metrik für Nutzer\*Innen ist. Für die Berechnung wird im ersten Schritt der nachfolgende Brute Force Algorithmus angewandt, der die Zuordnung von Segmentierungsinstanzen zu Annotationsinstanzen durchführt.

Die nachfolgende Formel zeigt das IPQ Bewertungskriterium unterteilt in die 3 Aufgaben:

$$\text{IPQ} = \underbrace{\frac{k_1 \times \sum_{(p,g) \in TP} \text{IoU}\left(\bigcup_{p_i \in p} p_i, g\right)}{|TP|}}_{\text{Segmentierungs-Qualität (SQ)}} \times \underbrace{\frac{k_2 \times |TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition Qualität (RQ)}} \times \underbrace{\frac{k_3 \times |GT|}{\sum_{p \in P} (\max(1, n_p - 1))}}_{\text{Injektivitäts-Qualität (IQ) (neu)}}, \quad (3.1)$$

---

**Algorithm 1** Beste Annotation-Zuordnung für jede Segmentierungsinstanz

---

**Eingabe:**  $maske_{\text{Vorhersage}}$ ,  $maske_{\text{Annotation}}$

**Ausgabe:**  $annotation_{\text{opt}}$ ,  $IoU_{\text{opt}}$

**Für**  $id_{\text{Instanz}}$  in  $|maske_{\text{Vorhersage}}|$  **tue:**

$Instanz \leftarrow maske_{\text{prediction}}[id_{\text{Instanz}}]$

**Für**  $annotation$  in  $maske_{\text{Annotation}}$  **tue:**

$IoU \leftarrow IoU(annotation, Instanz)$

**Wenn**  $IoU > IoU_{\text{opt}}[id_{\text{Instanz}}]$  **dann:**

$IoU_{\text{opt}}[id_{\text{Instanz}}] \leftarrow iou$

$annotation_{\text{opt}}[id_{\text{Instanz}}] \leftarrow annotation\_id$

**EndeWenn**

**EndeFür**

**EndeFür**

**Rückgabe**  $gt_{\text{opt}}$ ,  $IoU_{\text{opt}}$

---

wobei:

- $k_1, k_2, k_3$  Optionale Vorfaktoren zur Gewichtung der drei Teile der Metrik sind,
- $TP$  die Menge aller **TP**-Tupel  $(p, g)$  ist, wobei  $g$  eine Annotationeninstanz und  $p$  der Vektor aller zugehörigen Segmentierungsinstanzen ist,
- $|TP| \in \mathbb{Z}$  die Anzahl an korrekt erkannten Instanzen bezeichnet, also Annotationsinstanzen mit  $\text{IoU} > 0,5$ ,
- $\text{IoU}(\bigcup_{p_i \in p} p_i, g) \in [0, 1]$  die **IoU** zwischen allen Segmentierungsinstanzen  $p_i$  in der **TP**-Instanz  $p$  und der zugehörigen Annotationsinstanz  $g$  beschreibt,
- $|FP| \in \mathbb{Z}$  die Anzahl an falsch-positiven Segmentierungen ist, d. h. vorhergesagte Instanzen ohne Annotationsentsprechung,
- $|FN| \in \mathbb{Z}$  die Anzahl an nicht erkannten Annotationsinstanzen ist, also Annotationsinstanzen ohne zugehörige Vorhersage,
- $|GT| \in \mathbb{Z}$  die Anzahl der Annotationsinstanzen ist,
- $P$  die Menge aller Segmentierungsinstanzen ist, ungeachtet der Annotationszuordnung,
- $p \subseteq P$  ein Vektor aller Segmentierungsinstanzen, die der gleichen Annotationsinstanz zugeordnet sind, ist,
- $n_p$  die Dimension des Vektors  $p$  ist,
- $SQ \in [0, 1]$  ein Faktor ist, der die Qualität der Segmentierung anhand der **IoU** von den segmentierten und den erwarteten Instanz vergleicht,

- $RQ \in [0, 1]$  ein Faktor ist, der bewertet, wie vollständig und das Segmentierungsnetz die vorhandenen Nuclei gefunden hat und, ob es dabei zu Halluzinationen kam,
- $IQ \in [0, 1]$  ein Faktor ist, der das Unterteilen von Nuclei durch das Segmentierungsnetz zu bestrafen. Wird ein Nucleus durch mehrere Instanzen der Segmentierungsmaske dargestellt, wird  $n_p$  größer als eins und der Faktor sinkt,
- $IPQ \in [0, 1]$  ein Maß für die panoptische Segmentierungsqualität mit der Voraussetzung von injektiver Abbildung der Segmentierungsmasken-Instanzen auf die Annotationsinstanzen darstellt, wobei höhere Werte bessere Übereinstimmung bedeuten,

### 3.3 Klassifikatormethoden

Jedem instanzsegmentierten Nucleus muss eine Klasse zugewiesen werden, um die Ausgabe zur panoptischen Segmentierungsmaske zu erweitern. Erst die panoptische Segmentierungsmaske ermöglicht das automatische Extrahieren interpretierbarer Eigenschaften aus den Daten. Nuter\*Innen der vorgestellten Methoden wird mit dieser panoptischen Maske und den extrahierten Eigenschaften der Kultur ein klarer Überblick über den Status der Zellkultur geboten.

Für die Klassenzuweisung ist ein Klassifikator notwendig, der einen Bildausschnitt mit einer Nucleus-Instanz als Eingabe annimmt und ihr eine der vier Klassen als Ausgabe zuweist. Um diesen Klassifikator optimal zu entwerfen, wird ein umfangreicher Benchmark aus den Zieldaten erstellt, mit dem die vorgestellten Methoden verglichen werden. Benchmarks aus der Literatur umfassen weder dieselben Klassen noch dieselben Objektmerkmale, deshalb wird ein eigener, kein etablierter Benchmark verwendet. Für das Training wird einheitlich der Adam-Algorithmus [152] mit einer Lernrate von 0.0001 eingesetzt. Außerdem wird der Cross-Entropy-Loss [84] verwendet. Aus den annotierten Bilddaten werden für jede Anwendung ein Test- und ein Trainingsanteil im Verhältnis eins zu neun extrahiert. Alle betrachteten Variationen des Klassifikators werden ausschließlich mit den Trainingsdaten trainiert und ihre Leistung ausschließlich mithilfe der Testdaten getestet. Beide Anteile des Datensatzes werden durch Augmentierung erweitert und in Batches zusammengefasst. Zur Datenaugmentierung werden die folgenden Methoden eingesetzt:

- **Rotation:** Mit einer Wahrscheinlichkeit von 50% werden die Eingabedaten um 90° in der XY-Ebene rotiert.
- **Spiegelung:** Ebenfalls mit einer Wahrscheinlichkeit von 50% erfolgt eine Spiegelung entlang der Z-Achse.
- **Gaußsches Rauschen:** Mit einer Wahrscheinlichkeit von 20% wird Rauschen mit einem Mittelwert von 0 und einer Standardabweichung von 0,01 hinzugefügt.

Prominente Encoder aus der Literatur werden vergleichend eingesetzt. Darüber hinaus werden hier verschiedene Methoden der Vorverarbeitung, des Vortraining und der Decoderarchitektur eingeführt und verglichen. Im Folgenden sind diese Methoden einzeln be-

schrieben. Da jede mögliche Kombination mit jedem Netz zu trainieren einen unausführbar hohen Rechenaufwand bedeutet, wird eine Vorauswahl von Kombinationen getroffen.

### 3.3.1 Encoder

Tab. 3.1 zeigt die verschiedenen Encoder, die hier eingesetzt werden. Bis auf das Segmentierungsmodell CellposeSAM handelt es sich dabei um Encodern, die aus Klassifikatoranwendungen, die auf dem ImageNet-Datensatz [135] vorgenommen wurden, stammen. In der Tabelle sind die Namen, Anzahl der Parameter und, falls vorhanden, die Top-1-Genauigkeit (Acc@1) und die Top-5-Genauigkeit (Acc@5) auf dem ImageNet Datensatz angegeben.

**Tab. 3.1** | Vergleich der sechs vorgenommenen Netze hinsichtlich Genauigkeit auf dem ImageNet-Datensatz [135] und der Anzahl an Parametern. Angegeben sind sowohl die Top-1-Genauigkeit (Acc@1) als auch die Top-5-Genauigkeit (Acc@5), also ob die korrekte Klasse unter den besten 1 bzw. 5 Vorhersagen enthalten ist.

Name	Acc@1 (ImageNet)	Acc@5 (ImageNet)	Params (M)
ResNet18	69.76%	89.08%	11.7
ResNet101	77.37%	93.55%	44.5
Swin V2	84.11%	96.87%	87.9
ConvNeXt	84.41%	96.98%	197.8
EfficientNet V2	85.81%	97.79%	118.5
CellposeSAM	-	-	305

### 3.3.2 Vorverarbeitung

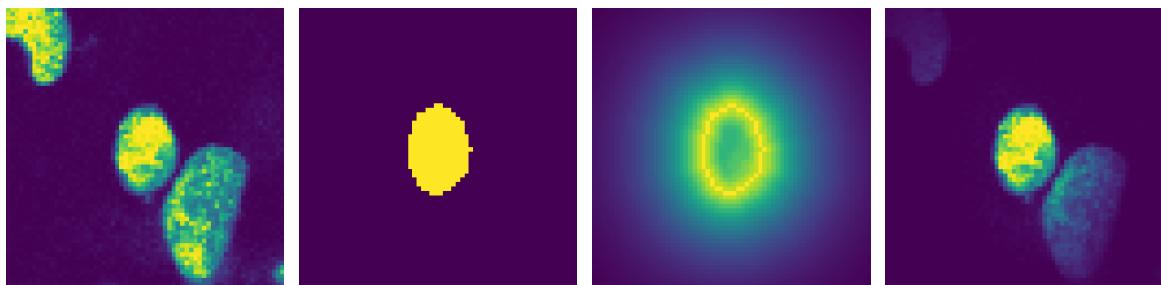
Da in vielen Bildausschnitten Nuclei sehr nah aneinander liegen werden zwei Vorverarbeitungsmethoden eingeführt, die dem Klassifikator signalisieren, welcher der sichtbaren Nuclei klassifiziert werden soll. Diese Methoden unterscheiden sich darin, wie die Segmentierungsmaske des Nucleus dem Klassifikator zugänglich gemacht wird. Die erste Methode ersetzt den Nucleus-Kanal mit der Segmentierungsmaske des gesuchten Nucleus. Das Ziel ist dabei, das Risiko zu minimieren, dass umliegende Nuclei das Klassifikationsergebnis verfälschen. Mit dieser Risikominimierung geht allerdings der Verlust der Oberflächenmerkmale einher. Außerdem ist das Klassifikationsergebnis bei dieser Vorverarbeitungsart von der Qualität der Segmentierung abhängig. Für die zweite Methode wird der Nucleus-Kanal mit einer Entfernungsmaske skaliert. Hierzu wird pixelweise der originale Nucleus-Kanal mit einer Transformation des Abstand aller Pixel außerhalb der Segmentierungsmaske wie folgt multipliziert:

$$I'(x) = I(x) \cdot \exp\left(-\frac{1}{\sigma} \min_{y \in \neg M} \|x - y\|_2\right), \quad (3.2)$$

wobei:

- $I(x) \in [0, 1]$  der Intensitätswert des Nucleus Kanal an der Position  $x$  ist,
- $I'(x) \in [0, 1]$  der Intensitätswert des neuen, transformierten Nucleus Kanal an der Position  $x$  ist,
- $x \in \Omega \subset \mathbb{N}^3$  die Position eines Voxels im diskreten Bildraum ist,
- $M \subseteq \Omega$  die Segmentierungsmaske und  $\neg M = \Omega \setminus M$  deren Komplement im Bildraum sind,
- und  $\sigma \in \mathbb{R}^+$  ein Parameter zur Steuerung des exponentiellen Abfalls ist.

Die Verwendung der zweiten Methode hat zum Ziel, dass die Oberflächenmerkmale des Nucleus erhalten bleiben. Außerdem wird mit der Vorverarbeitungsmethode der Einfluss der eventuell fehlerhaften Segmentierungsmasken durch die kontinuierliche Abstands- transformation minimiert. Allerdings ist hierdurch auch das Risiko von Einflussnahme auf das Klassifikationsergebnis durch umliegende Nuclei nicht vollständig eliminiert, sondern nur vermindert. In Abb. 3.3 sind die Nuclei-Kanäle der verschiedenen Methoden dargestellt.



a) Ausgeschnittener Bereich des originalen Nucleus-Kanals mit mehreren, intuitiv trennbaren Nuclei.  
b) Segmentierungsmaske des Nucleus. Die erste Methode ersetzt den originalen Nucleus-Kanal mit dieser Maske. Die binäre Maske zeigt keine Oberflächenmerkmale des Nucleus, lediglich die geometrischen Merkmale.  
c) Entfernungsmaske des Nucleus. Diese wird pixelweise mit dem Nucleus-Kanal multipliziert für die zweite Vorverarbeitungsmethode.  
d) Darstellung der Vorverarbeitungsmethode zwei, die durch die Multiplikation des Nucleus-Kanal mit der Entfernungsmaske entsteht. Zu sehen ist, dass der nahegelegene ungewünschte Nucleus noch stellenweise mit hoher Intensität vertreten ist.

**Abb. 3.3** | Darstellungen der verschiedenen Vorverarbeitungsmethoden.

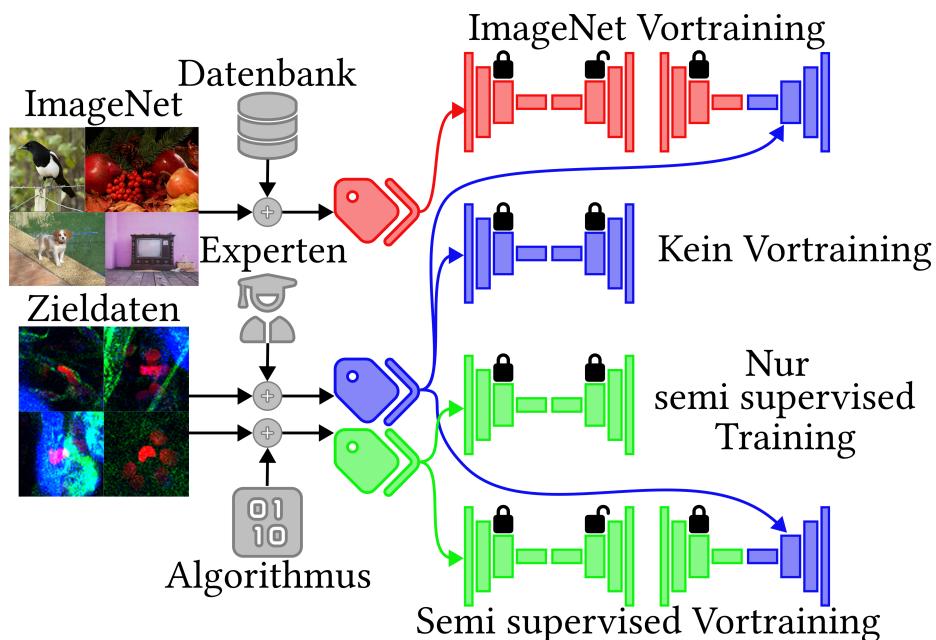
Je nach Modellarchitektur müssen die Bilddaten noch skaliert werden, bevor sie den vortrainierten Modellen übergeben werden können, da die Klassifikatoren Eingaben konstanter Größe benötigen. Dazu wird einfache bilineare Interpolation verwendet (siehe Kap. 2.2.3). Weil hier mit Rechenzeit-intensiven dreidimensionalen Daten umgegangen werden muss, ist auch ein dynamisches Speichermanagement Teil der Vorverarbeitungsmethoden. Die Anwendung der beschriebenen Methoden wird dazu zusammengefasst in einem neu entwickelten 'Retriever'. Dieser Retriever versieht die aktuell gewünschten Bildausschnitte mit der Vorverarbeitungsmethode und verschiebt dann ausschließlich diese Daten auf die GPU.

### 3.3.3 Vortraining

Aus der Literatur sind verschiedene Methoden des Vortraining bekannt. Hier werden:

- Kein Vortraining,
- semi-supervised, und
- fully-supervised Vortraining

betrachtet. Die Abb. 3.4 zeigt die hier umgesetzten Methoden.



**Abb. 3.4** | Übersicht über die Vortrainingsmethoden. Links zu sehen sind die beiden verfügbaren Bildermengen, ImageNet und die Zieldaten. Rechts von diesen Bildermengen werden diesen Bildern Annotationen hinzugefügt, entweder durch die ImageNet-Datenbank, Experten oder einen Algorithmus. Jeder Encoder (linke Seite eines Netzwerks) und jeder Decoder (rechte Seite eines Netzwerks) wird mit einer dieser drei Label-Mengen trainiert. Die Farbe der Label und des Netzwerks zeigt dabei die Zuordnung. Mit offenen oder geschlossenen Schlössern über den En- und Decodern ist dargestellt, ob die Gewichte eingefroren werden. Vier verschiedene Versionen jedes Klassifikators werden hier trainiert. Das erste Netzwerk wird auf den ImageNet-Daten vorgenommen. Anschließend wird mit Experten-Labels der Decoder neu trainiert. Das Zweite erhält kein Vortraining, es ist komplett mit den Zieldaten trainiert. Für das dritte Netzwerk werden lediglich die Label des semi supervised Algorithmus eingesetzt. Die letzte Variante wird semi supervised vorgenommen und anschließend mit den Zieldaten fine-tuned.

**Kein Vortraining** Ohne Vortraining startet der Encoder, der den Großteil der Gewichte umfasst, mit zufällig initialisierten Werten. Da diese zufälligen Werte keine sinnvollen Merkmale extrahieren, wird ein besonders langes Training mit den Zieldaten durchgeführt. Jedes Modell wird jeweils für 75 Epochen trainiert.

**Semi supervised** Die semi supervised Annotationen werden mithilfe eines few-shot gestützen Cluster Algorithmus erstellt, der hier 'Pseudo-Labler' genannt wird. Ein Experte erstellt hierzu Annotationen von wenigen Nuclei. Danach werden aus den restlichen Segmentierungsmasken einige Merkmale generiert und zu einem Vektor zusammengefasst. Zuerst werden das Volumen, die Oberfläche und die Achsenlängen jedes Nucleus direkt bestimmt. Außerdem wird die Exzentrizität aus dem Verhältnis der längsten und der kürzesten Achse berechnet. Für die Kompaktheit wird das Volumen der Maske durch die kleinste mögliche Begrenzungsbox geteilt. Darüber hinaus wird aus der Z-Schicht, in der die Segmentierungsmaske am größten ist, die 2D-Kontur erfasst. Aus dieser Kontur wird eine komplexe Zahlenfolge berechnet und der Absolutwert der ersten zehn Koeffizienten als einzelne, weitere Merkmale dem Merkmalsvektor hinzugefügt.

Der Pseudo-Labler normalisiert die Werte des Merkmalsvektoren zu einem Mittelwert von Null und einer Varianz von Eins und wendet eine Principal Component Analysis [66] an, um redundante Informationen zu entfernen. Das Ziel dabei ist es, einen Mittelweg zwischen Informationserhalt und Overfitting-Gefahr sowie Rechenaufwand zu erzielen. Mithilfe des Label-Spreading-Algorithmus [73], mit einer Radial Basis Funktion [74] als Kernelfunktion, werden die Experten-Annotationen über die Struktur der Daten auf alle Stichproben ausgebreitet. Durch den Pseudo-Labler entstehen Annotationen für die Daten ohne Experten-Annotationen. Diese neuen Annotationen werden dann eingesetzt, um in 25 Epochen sowohl die Encoder, als auch die Decoder zu trainieren, mit dem Ziel unter geringem Aufwand für die Experten umfangreiche Klassifikatoren zu trainieren. Optional werden die Gewichte des Encoder hiernach, bis auf die letzten beiden Schichten eingefroren und nur der Encoder wird in weiteren 35 Epochen mit dem Trainingsdatensatz der Zieldaten trainiert. Das Ziel dieses Vorgehens ist es, eine stärkere Generalisierung zu erreichen, indem Overfitting bei der Merkmalsextraktion vermieden wird. Da der Encoder mit anderen Daten vorgenommen wird, ist zu erwarten, dass er eine sinnvolle Merkmalsextraktion lernt, ohne auf die expliziten Merkmale der individuellen Stichproben im Trainingsdatensatz angewiesen zu sein. Dadurch sind die Beziehungen zwischen Merkmalen und Klassen, die der Decoder lernt, nicht nur auf die Merkmale des Trainingsdatensatzes beschränkt.

**Fully-supervised** Das Fully-supervised Vortraining bezieht sich hier auf das initialisierten eines Encoders mit den Gewichten einer entsprechenden Veröffentlichung. Diese Gewichte sind durch Vortraining auf dem ImageNet-Datensatz entstanden oder stammen aus dem SAM-Encoder. Bis auf die letzten 20 bis 30 Prozent der Schichten werden alle Gewichte des Encoders während dem Training eingefroren. **Kommentar: Ist '20 bis 30' okay?** Ich habe alles 3 mal trainiert, mit 20, 25 und 30 und dann das beste genommen. In 50 Epochen werden dann die verbleibenden Encoder-Schichten und der Decoder trainiert. Die Merkmalsextraktion ist aus einem Datensatz einer anderen Domäne gelernt, was Overfitting verhindert. Nur der Decoder wird auf Zieldaten trainiert, wobei aufgrund der diversifizierten Merkmale eine hohe Generalisierbarkeit angestrebt wird.

### 3.3.4 Decoder

An die Encoder werden verschiedene Decoder angehängt. Hier werden dazu zwei neue Decoderarchitekturen eingeführt. (**NOTIZ: "nach dem Vorbild vergleichbarer Decoder in der Literatur oder ähnliches?"**) Abb. 3.5 zeigt die beiden Architekturen systematisch. Der erste Klassifikator wird hier *Volumen-Klassifikator* genannt. Er interpretiert die Merkmale, die der Encoder generiert, als Volumen und generiert mithilfe von 3D-Faltungen und Pooling eine Repräsentation daraus. Diese Repräsentation wird dann durch Linear Layers zu vier Ausgabe-Klassen umgeformt. Die Idee des *Volumen-Klassifikators* ist es, die Merkmale, die der Encoder generiert, möglichst vollständig zu erfassen und alle räumlichen Beziehungen, auch in Z-Richtung, festzustellen. Hierbei ist das Ziel, dass durch die 3D-Faltungen eine domänenspezifische Interpretation der Merkmale gelernt wird, sodass die neuen Merkmale nach dem anschließenden Pooling aussagekräftig und niederdimensional sind. Daneben wird hier der *Schichten-Klassifikator* eingeführt. Der *Schichten-Klassifikator* betrachtet die einzelnen Schichten des Bilds anhand der individuellen Schichten, die der Encoder ausgibt. Dazu werden die räumlichen X- und Y-Dimensionen durch einen spatial-average zusammengefasst. Durch eine multihead Attention mit vier Attention-Köpfen und Embedding-Dimension 256 wird dann eine Repräsentation aus den individuellen Schichten der Merkmale erstellt. Lineare Layers formen anschließend die Repräsentation zu den vier Klassen um. Für den textitSchichten-Klassifikator ist das Ziel, dass durch die Vereinfachung der Daten aussagekräftige, schichtenweise Merkmale entstehen und, dass diese räumlich invariant sind, da der betrachtete Nucleus in den Bildfenstern zentriert sind. Auf einem geringfügigen Datensatz werden alle angeführten Methoden in den möglichen Kombinationen umgesetzt, um Vergleiche zu ermöglichen. Dieser geringfügige Datensatz besteht aus Bildern der Zieldomäne und halb automatisch generierten Annotationen. Anschließend werden die besten Methoden ausgewählt und auf den finalen Datensatz trainiert.

## 3.4 Segmentierung

### 3.4.1 Modelle

Für die Instanzsegmentierung der Nuclei werden die folgenden drei Modelle eingesetzt:

- Ein Modell des nnU-Net Framework, das selbstkonfigurierte Modelle auf der U-Net-Architektur basiert erstellt [133].
- Das DeepCell-Caliban-Modell, das Bildfaltungen und speziell entwickelten Nachverarbeitungsstrategien vereint.
- Cellpose-SAM, das die Architekturen der Cellpose-Modelle mit der Architektur und den Gewichten vom Foundation-Model **SAM** vereint.

### 3.4.2 Nachverarbeitungsmethoden

Zur Nachbearbeitung der Instanzsegmentierungsmasken wird der Instanz-Trenner eingeführt. Diese Methode ist dazu da, separate Instanzen mit gleichem Label zu trennen und mit einzigartigen Labels zu versehen. Pro Instanz werden hierzu ein zufälliges Pixel betrachtet und davon ausgehend alle Pixel mit direkter Verbindung über die Segmentierungsmaske gesucht. Diese verbundenen Pixel werden als neue einzigartige Instanz abgelegt, bis keine Pixel ohne Verbindung übrig sind. Eine weitere Methode der Nachbearbeitung die auf die Segmentierungsmasken eingesetzt werden kann ist der Watershed-Algorithmus (siehe 2). Der Algorithmus trennt überlagerte Instanzen, die das Modell als einzelne Instanz segmentiert hat.

---

#### Algorithm 2 Watershed-Nachbearbeitung zur Trennung überlappender Instanzen

---

**Eingabe:**  $maske_{instanz}$

**Ausgabe:**  $maske_{refined}$

$distanz \leftarrow \text{DistanzTransformation}(maske_{instanz})$

$marker \leftarrow \text{LokaleMaxima}(distanz)$

$gradient \leftarrow -distance$

$maske_{refined} \leftarrow \text{Watershed}(gradient, marker)$

**Für** Region  $r_i$  in  $maske_{refined}$  **tue:**

**Wenn**  $Flche(r_i) < \tau$  **dann:**

**Wenn**  $r_i$  grenzt an größere Nachberregion  $r_n$  **dann:**

            Vereinige  $r_i$  mit  $r_n$

**Sonst:**

            Entferne  $r_i$

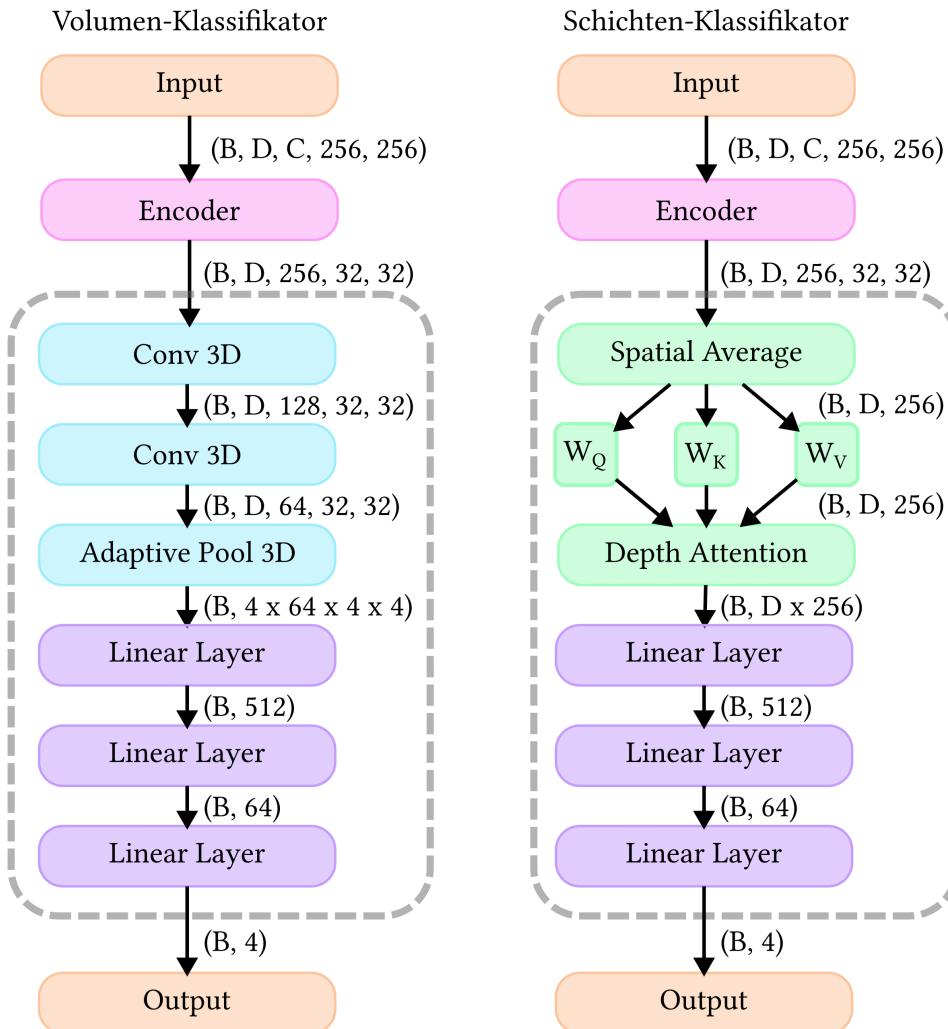
**EndeWenn**

**EndeWenn**

**EndeFür**

**Rückgabe**  $maske_{refined}$

---



**Abb. 3.5 |** Architektur der beiden Klassifikatoren. Der Encoder wird bei beiden modular ausgetauscht. Links zu sehen ist die Architektur des *Volumen-Klassifikators*, der die Merkmale, die der Encoder generiert, als Volumen interpretiert und mithilfe von 3D-Faltungen und Pooling daraus eine Repräsentation generiert. Diese Repräsentation wird dann durch Linear Layers zu vier Ausgabe-Klassen umgeformt. Rechts zu sehen ist der *Schichten-Klassifikator*. Die räumlichen X- und Y-Dimensionen werden im *Schichten-Klassifikator* durch einen spatial-average zusammengefasst. Durch eine multihead Attention mit vier Attention-Köpfen und Embedding-Dimension 256 wird dann eine Repräsentation aus den individuellen Schichten der Merkmale erstellt. Lineare Layers formen anschließend die Repräsentation zu den vier Klassen um.

# Implementierung 4

---

## 4.1 Überblick

Im nachfolgenden Kapitel wird die Implementierung aller relevanten Methoden erklärt mit Fokus auf die praktische Anwendung dieser Methoden in Form von Nutzerschnittstellen oder als Entwicklerskript auf ausgewiesener Hardware. Eine neu entwickelte Anwendung, die 3D-Zelldaten-Pipeline, vereint verschiedene Software-Module, die jeweils ein Zwischenziel der vorliegenden Arbeit umsetzen. Die Anwendung wird mit einer grafischen Nutzeroberfläche bedient, die auch ohne Programmierkenntnisse genutzt werden kann. Die beschriebenen Funktionen sind im bereitgestellten Repository zu finden.

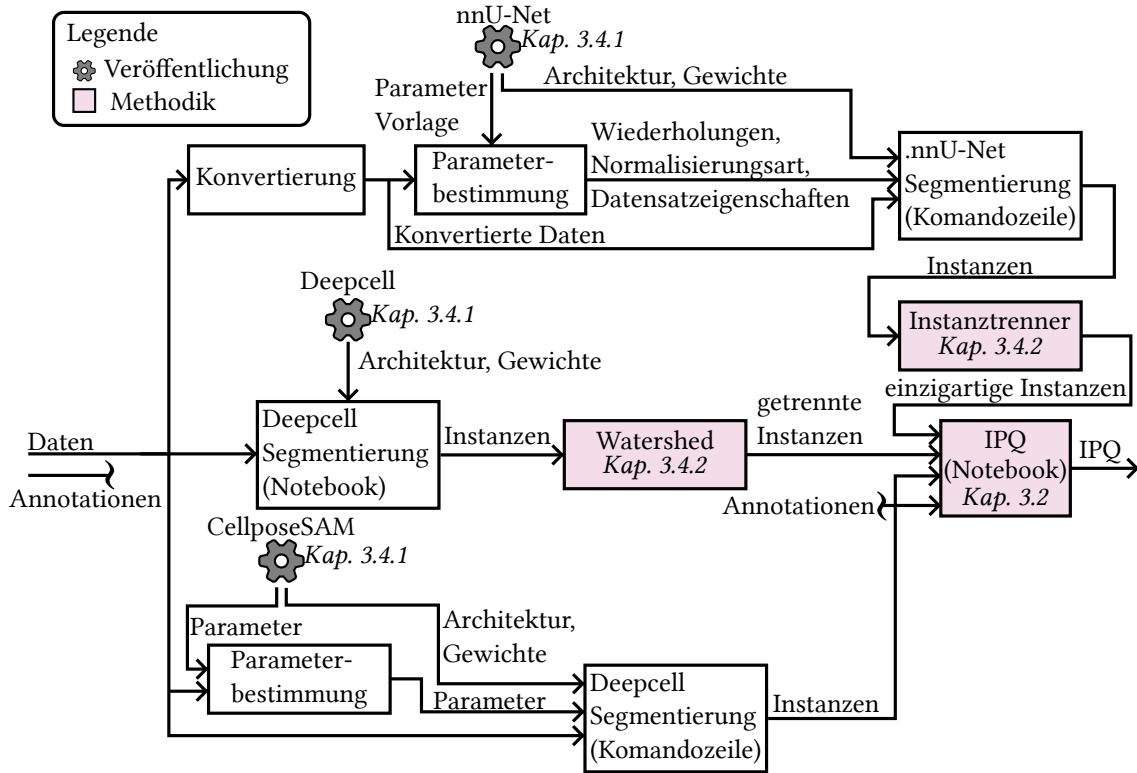
## 4.2 Segmentierungsmodelle

Abb. 4.1 zeigt den Signalfluss der Anwendung der Segmentierungsmodelle und ihrer Bewertung. Die Segmentierungsmodelle werden zur Evaluation in einfachen Entwicklerskripten über die Kommandozeile oder als Jupyter Notebook ausgeführt. Für das nnU-Net-Modell (siehe Kap. 3.4.1) wird auf die Eingabedaten zuerst eine Konvertierung angewandt, die sie durch Padding in die geforderte Größe gebracht und zu .nii.gz-Dateien umgeformt werden. Eine Parameterbestimmung für die Anzahl der Wiederholungen, Normalisierungsart und Datensatz-Eigenschaften wird anhand der Parameter Vorlage der Autoren erstellt. Das Modell wird direkt mit der Architektur und den Gewichten der Veröffentlichung initialisiert und über die Kommandozeile auf alle Daten angewandt. Die entstehenden Instanzen werden in einen Instanztrenner gegeben (siehe Kap. 3.4.2), um den Instanzen einzigartige Labels zu vergeben.

Für das Deepcell-Modell (siehe Kap. 3.4.1) wird direkt das bereitgestellte Jupyter-Notebook der Deepcell-Veröffentlichung auf die Benchmarkdaten angewandt [131]. Die Ergebnisse werden daraufhin mit dem Watershed-Algorithmus der OpenCV-Bibliothek nachbearbeitet (siehe Kap. 3.4.2).

Das CellposeSAM-Modell (siehe Kap. 3.4.1) wird als Entwicklerskript weitgehend mit der Architektur, den Gewichten und den vorgeschlagenen Parametern der Autoren angewandt. In einer Parameterbestimmung wird der durchschnittliche Durchmesser der Nuclei an die eingegebenen Daten angepasst. Auf die Ergebnisse wird keine Nachbearbeitung angewandt.

Die Instanzen von jedem der drei Modelle und die Annotationen werden in einem Jupyter Notebook zur Berechnung der IPQ eingegeben (siehe Kap. 3.2). Mithilfe einer neu entwi-



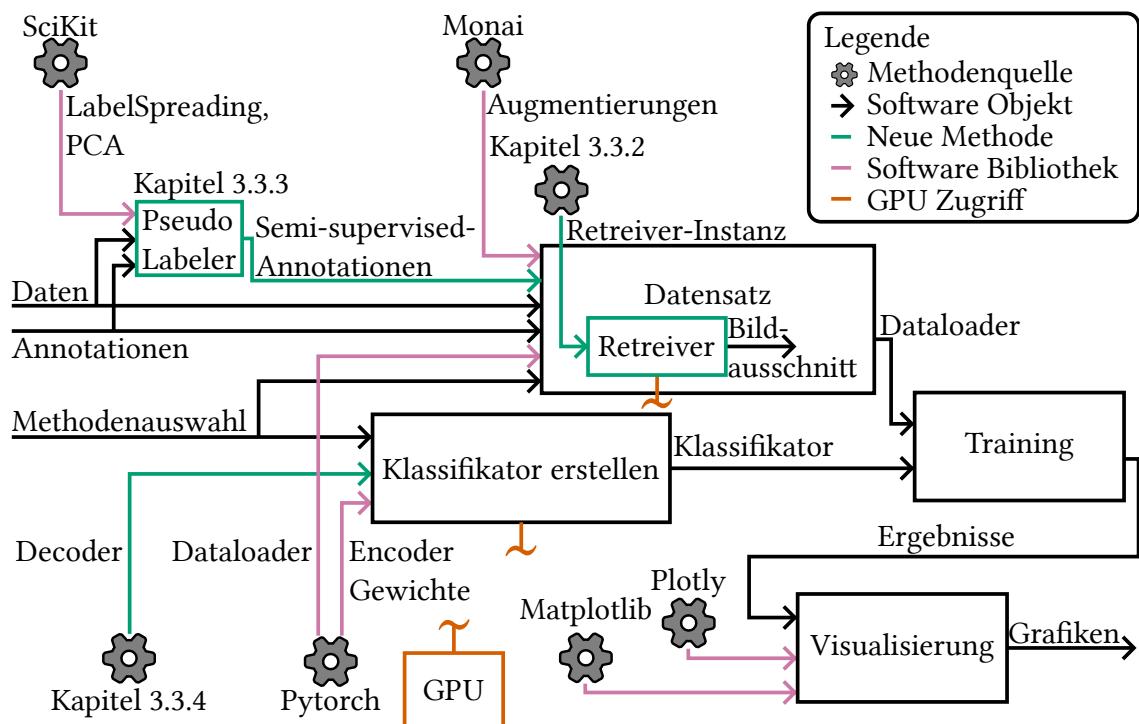
**Abb. 4.1 |** Signalflussdiagramm der praktischen Umsetzung der Segmentierungsmodelle.

ckelten Funktion werden hier die Ergebnisse mit der [IPQ](#)-Metrik bewertet.

### 4.3 Klassifikatoren

Für die Klassifikatoren werden vier Software-Module entwickelt, wie in Abb. 4.2 dargestellt. Das erste Modul nimmt als Eingabe eine Methodenauswahl (siehe Kap. 3.3), die verwendet werden sollen, und gibt einen Klassifikator zurück. Mithilfe der Pytorch-Bibliothek werden entsprechende Encoder und, abhängig von der Vortrainingsmethode, entsprechende Gewichte geladen. Außerdem werden neu entwickelte Klassen von PyTorch-Modellen für das Erstellen der Decoder aufgerufen (siehe Kap. 3.3.4).

In das zweite Modul wird auch eine Methodenauswahl eingegeben und abhängig von der Vortrainings- und Vorverarbeitungsmethode wird ein entsprechender Datensatz zurückgegeben (siehe Kap. 3.3.3 und Kap. 3.3.2). Dieser Datensatz wird als Pytorch Dataloader erstellt und mit einer Batch-Größe und mit Augmentierungen der Monai-Bibliothek versehen. In diesen Dataloader wird eine neu entwickelte Retriever-Instanz (siehe Kap. 3.3.2) eingebettet. Der Retriever ermöglicht es, die großen, dreidimensionalen Daten nicht alle auf einmal als Variablen im Datensatz unterzubringen, sondern nur die Indizes von Bildern und Instanzen. Diese Indizes werden dann genutzt, um dynamisch nur die gewünschten



**Abb. 4.2** | Signalflossdiagramm der praktischen Umsetzung der Klassifikatoren. Zu sehen sind die Module, aus denen sich die Anwendung der Klassifikatoren zusammensetzt. Links sind als Eingabe Daten mit zugehörigen Annotationen und eine Kombination von Methoden und Rechts Grafiken, die die Ergebnisse visualisieren als Ausgabe zu sehen. Verschiedene Python-Bibliotheken werden genutzt (Rosa), aber auch neue Methoden werden eingesetzt (Grün). Der Klassifikator und der Retriever, der dynamisch Bildausschnitte lädt, greifen auf die GPU zu (Orange).

Bildausschnitte auf die GPU zu laden. In einem Pseudo-Labler-Modul werden zusätzlich Semi-supervised-Annotationen mithilfe der PCA und der Label-Spreading-Funktion der SciKit-Bibliothek erstellt (siehe Kap. 3.3.3).

Das dritte Modul erhält einen Klassifikator und einen Datensatz und führt einen Trainingsdurchlauf durch. Es speichert die Ergebnisse und die Gewichte der neu trainierten Klassifikatoren in automatisch benannten Dateien.

Im vierten Modul werden mithilfe der plotly-Bibliothek Grafiken aus den eingegebenen Ergebnissen erstellt und, mithilfe der Matplotlib-Bibliothek, separat als .png gespeichert.

## 4.4 3D-Zelldaten-Pipeline

In Abb. 4.3 ist die Software-Architektur der 3D-Zelldaten-Pipeline als Signalflossdiagramm dargestellt. Die Anwendung vereint die eingeführten Methoden der vorliegenden Arbeit (siehe Kap. 3). Oben zentral zu sehen ist das Graphical User Interface (GUI), die grafische Schnittstelle mit Bedienelementen und Visualisierungen für Interaktionen mit Nutzer\*Innen. Von der GUI gehen Start- und Stop-Signale an die wichtigsten Module aus (Segmentierung,

Labeling-App, Visualisierung und Methodenvergleich). Diese vier Module werden direkt von Nutzer\*Innen bedient. Das Segmentierung-Modul greift auf lokale Dateien zu, um Bilder einzulesen und startet dann die Segmentierung für diese Daten mit verschiedenen Modellen (siehe Kap. 4.2). Die gefundenen Segmente werden dann an das IPQ-Modul gegeben, das die IPQ-Metrik für die gefundenen Segmente zurückgibt, insofern Annotations zu den Daten existieren (siehe Kapitel 3.2). Das Labeling-App-Modul beinhaltet eine Schleife aus einer Retriever-Instanz und einer Eingabeaufforderung (siehe Kap. 3.3.2). Diese Retriever-Instanz greift auf lokale Dateien zu, um abhängig von der Nutzereingabe einen Bildausschnitt zu laden. Dem Bildausschnitt weisen Nutzer\*Innen dann Annotations zu, die anschließend lokal gespeichert werden. Im Methodenvergleich werden die Annotationsen und Semgentierungsmasken eingelesen. In einer Schleife werden mit Hilfe eines Klassifikator-Helper Klassifikatoren aus verschiedenen Methoden-Kombinationen erstellt und trainiert. Der Datensatz hierzu wird von einem Datensatz-Helper-Modul erstellt und beinhaltet eine Retriever Instanz (siehe Kap. 3.3.2). Ein Pseudo-Labler-Modul erstellt Semi-supervised-Annotationsen für den Datensatz, die je nach Vortrainingsmethode in der Schleife genutzt werden können (siehe Kap. 3.3.3). Die Ergebnisse der verschiedenen Klassifikatoren werden anschließend verglichen und gespeichert. Das Visualisierung-Modul liest die Ergebnisse des Klassifikators und stellt diese in Grafiken dar.

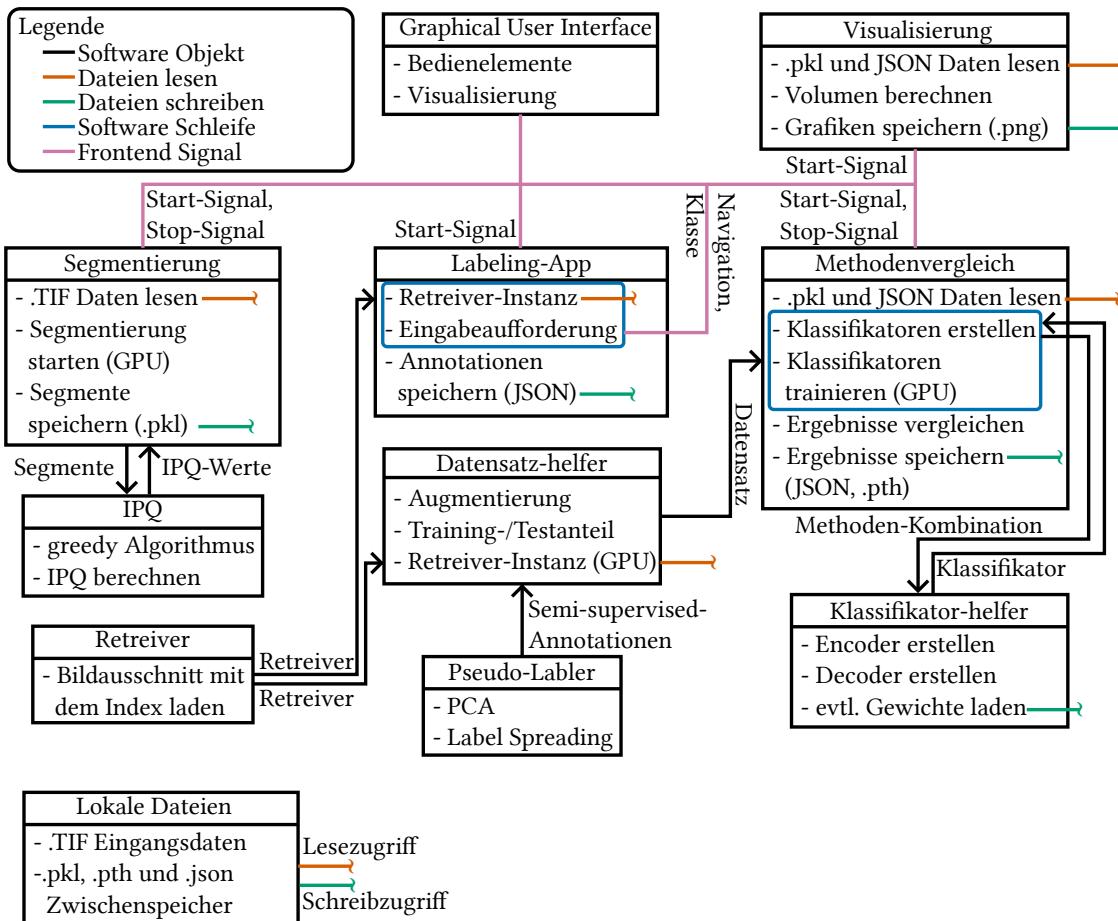
Abb. 4.4 zeigt die GUI der 3D-Zelldaten-Pipeline. In den fünf Tabs (Segmentierung, Labeling-App, Methodenvergleich, Training und Visualisierung) werden die Aufgaben der Anwendung ausgeführt. Als Frontend der App dient eine Dash-Anwendung, die eine einfache HTML-Seite bereitstellt. Das Backend ist mit Python erstellt, und die gesammelten Daten werden als JSON gespeichert. Zwischenergebnisse wie trainierte Modelle und verarbeitete Daten werden als nicht interpretierbare Python-spezifische Datentypen abgelegt. Hierzu ist ein Docker mit Zugriff auf das lokale Dateiensystem versehen und über Kubernetes betrieben.

### 4.4.1 Segmentierung

Abb. 4.4 zeigt den Tab der 3D-Zelldaten-Pipeline, in dem die Segmentierung der Zellkerne vorgenommen wird. Im Eingabefeld 'Eingabe Ordner' wird ein Ordner, relativ zu dem Speicherort der Anwendung, angegeben, aus dem TIF-Bilder gelesen werden. Unter 'Ausgabe Ordner' wird angegeben, wohin die Instanzsegmentierungsmasken gespeichert werden. Mit dem Knopf 'Starte Segmentierung' wird die Segmentierung gestartet und mit dem 'Abbrechen'-Knopf wieder abgebrochen. Das Textfeld 'Segmentation running...' blinkt im Takt von einer Sekunde, solange die Segmentierung durchgeführt wird.

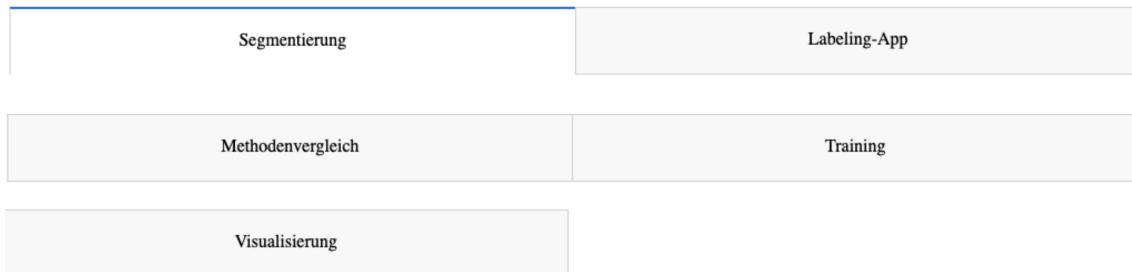
### 4.4.2 Labeling-App

Um den Klassifikator zu trainieren, werden einigen Zieldaten mithilfe der neu entwickelten Labeling-App-Annotationen hinzugefügt. Die Anforderungen an die Labeling-App sind ein nutzerfreundlicher, zeiteffizienter Ablauf und Zugänglichkeit für Nutzer\*Innen ohne Programmiererfahrung. Funktional muss die Labeling-App imstande sein, die zu



**Abb. 4.3 |** Übersicht der Architektur der neu entwickelten 3D-Zelldaten-Pipeline. Als Kästen sehen sind die Module die die 3D-Zelldaten-Pipeline bilden. Schwarze Pfeile stellen die Weitergabe von Software Objekten zwischen den Modulen dar. In Orange und Grün sind Zugriffe auf lokale Dateien dargestellt, Orange ist Dateien lesen und Grün Dateien schreiben. Software Schleifen sind als Blaue Kästen dargestellt. Rosa Verbindungen stellen die Signale von und an die GUI dar.

### 3D-Zelldaten-Pipeline



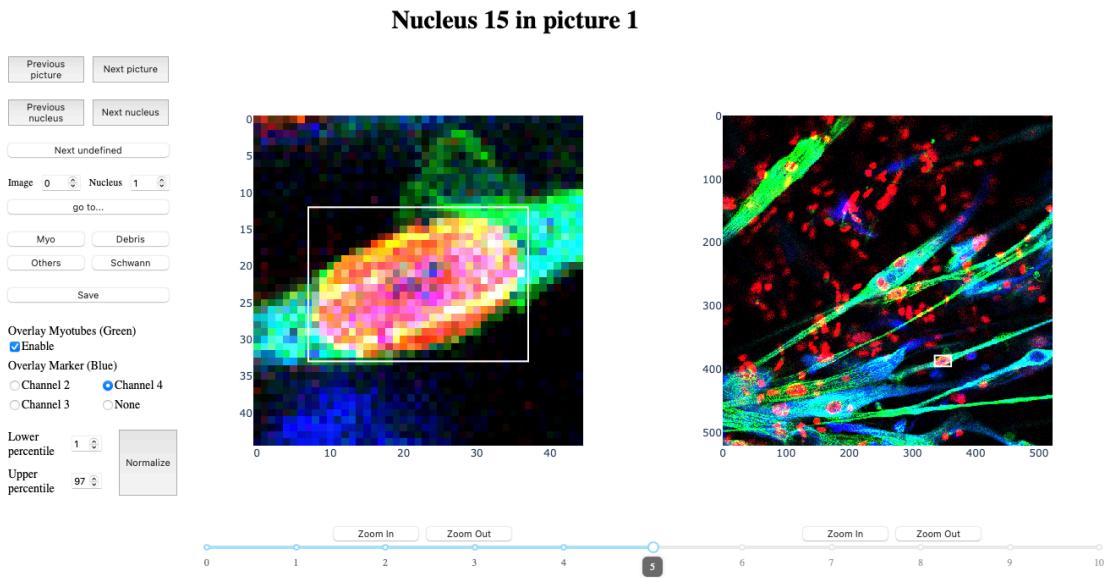
#### 1) Segmentierung

Eingabe Ordner   
Auszabe Ordner

*Eingabe Ordner: Demo/Segmentation*  
 *Output folder: Demo/Methodenvergleich*

Segmentation running...

**Abb. 4.4** | Die Abbildung zeigt einen Ausschnitt der 3D-Zelldaten-Pipeline. Oben sind fünf Tabs zu sehen, mit denen die auszuführende Aufgabe gewählt werden kann. Diese Tabs sind zur besseren Darstellung untereinander statt nebeneinander platziert. Darunter ist ein Ausschnitt des ausgewählten 'Segmentierung' Tab zu sehen.



**Abb. 4.5** | Die Abbildung zeigt die GUI der neu entwickelten Labeling-App. In der Mitte wird die Zelle angezeigt, die gelabelt werden soll, und links sind Bedienelemente zu sehen. Unter der dargestellten Zelle befindet sich ein Schieberegler, der die Navigation entlang der Z-Achse ermöglicht. Über die Bedienelemente kann der/die Nutzer\*In zwischen Bildern und Zellen umschalten, die Klasse der Zelle bestimmen, den gezeigten Ausschnitt mit prozentualen Schwellenwerten normalisieren und die Fenstergröße ändern.

klassifizierenden Nuclei zu visualisieren und Eingabemöglichkeiten zu bieten, mit denen die Experten die Klasse des Nucleus eintragen können. Diese eingetragenen Annotationen müssen sinnvoll gespeichert werden. Abb. 4.5 zeigt die neu entwickelte GUI der Labeling-App. Mit den Bedienelementen werden die Anforderungen an die Funktionalität umgesetzt. Zentral zu sehen sind zwei Fenster, die je einen 2D-Schnitt des ausgewählten Nucleus anzeigen. Diese Schnitte werden von einem neu entwickelten Retreiver dynamisch geladen (siehe Kap. 3.3.3). Das linke Bild zeigt den Nucleus stark vergrößert, das rechte Bild zeigt die Umgebung des Nucleus. Auf den Bildern ist jeweils ein Kasten gezeichnet, der den ausgewählten Nucleus umrandet, um Nuclei, die dicht aneinander liegen, zu unterscheiden. Mit dem Schieberegler unter den Fenstern wird ausgewählt, welche der Schichten, in denen der Nucleus anwesend ist, gezeigt werden soll. Über dem Fenster ist der Index des aktuell dargestellten Nucleus und des Bilds angegeben. Links neben dem Fenster befinden sich Bedienelemente mit den folgenden Funktionen:

- *Previous picture*: Vorheriges Bild auswählen,
- *Next picture*: Nächstes Bild auswählen,
- *Previous nucleus*: Vorherigen Nucleus auswählen,
- *Next nucleus*: Nächsten Nucleus auswählen,
- *Next undefined*: Nächsten Nucleus ohne eingetragene Annotation auswählen,

- *Image*: Eingabefeld für das auszuwählende Bild,
- *Nucleus*: Eingabefeld für den auszuwählenden Nucleus,
- *go to...*: Bild und Nucleus auswählen, wie in den Eingabefeldern,
- *Myo*: 'Myotuben-Zellkern'-Klasse als Annotation des ausgewählten Nucleus definieren,
- *Debris*: 'Überreste'-Klasse als Annotation des ausgewählten Nucleus definieren,
- *Other*: 'Andere'-Klasse als Annotation des ausgewählten Nucleus definieren,
- *Schwann*: 'Schwannzellen-Zellkern'-Klasse als Annotation des ausgewählten Nucleus definieren,
- *Save*: Manuell die festgelegten Annotations speichern. Die Labeling-App speichert außerdem eigenständig periodisch,
- *Overlay Myotubes (Green)*: Mit einem Haken bei 'Enable' wird der Marker, der die Myotuben einfärbt, in Grün eingeblendet,
- *Overlay Marker (Blue)*: Einer oder keiner der restlichen vorhandenen Marker wird in Blau eingeblendet,
- *Lower Percentile*: Unteren prozentualen Schwellwert wählen, der bei der nächsten Normalisierung angewandt werden soll,
- *Upper Percentile*: Oberen prozentualen Schwellwert wählen, der bei der nächsten Normalisierung angewandt werden soll,
- *Normalize*: Normalisierung lokal auf den Ausschnitt des aktuell ausgewählten Nucleus anwenden. Intensitätswerte, die im beziehungsweise über dem Perzentil der eingetragenen Schwellwerte, werden hierbei zusammengefasst,
- *Zoom In*: Verkleinert den anzuzeigenden Ausschnitt und
- *Zoom Out*: Vergrößert den anzuzeigenden Ausschnitt.

Aufgrund der Anforderung einer nutzerfreundlichen, zeiteffizienten Bedienung sind alle Berechnungen, die während der Nutzung der Labeling-App ausgeführt werden, darauf ausgelegt, die Rechenzeit zu minimieren. Hierzu werden alle Bilder und Masken bei der Initialisierung der App in den Cache geladen. Des Weiteren ist eine Python-Klasse angelegt, die separat noch das aktuell ausgewählte Bild und die ausgewählte Zelle speichert. Erst wenn eine Änderung der Auswahl ausgeführt wird, wird eine neue Zelle oder ein neues gesamtes Bild geladen und selbst dann lediglich aus dem Cache.

### 3) Methodenvergleich

Eingabe Ordner	<input type="text" value="Demo/Methodenvergleich"/>
Ausgabe Ordner	<input type="text" value="Demo/Training"/>
<input checked="" type="checkbox"/> Eingabe Ordner: Demo/Methodenvergleich	
<input checked="" type="checkbox"/> Ausgabe Ordner: Demo/Training	
Methoden Optimierung (Mindestens eine Methode je Gruppe)	
<b>Encoder</b>	
<input type="checkbox"/> CellposeSAM	
<input type="checkbox"/> ResNet18	
<input type="checkbox"/> ResNet101	
<input type="checkbox"/> SwinV2	
<input type="checkbox"/> ConvNeXt	
<input type="checkbox"/> EfficientNetV2	
<b>Decoder</b>	
<input type="checkbox"/> Schichten-Klassifikator	
<input type="checkbox"/> Volumen-Klassifikator	
<b>Vorverarbeitung (Nucleus Kanal)</b>	
<input type="checkbox"/> Distanztransformation	
<input type="checkbox"/> Segmentierungsmaske	
<b>Vortraining</b>	
<input type="checkbox"/> Kein Vortraining	
<input type="checkbox"/> Semi-supervised	
<input type="checkbox"/> Fully-supervised	
<input type="button" value="Starte Methodenvergleich"/>	<input type="button" value="Abbrechen"/>

**Abb. 4.6** | Die Abbildung zeigt einen Ausschnitt des Tabs der 3D-Zelldaten-Pipeline der den Vergleich der Klassifikatormethoden ermöglicht. Methoden der Kategorien Encoder, Decoder, Vorverarbeitung und Vortraining können ausgewählt werden, um einen Vergleich aller Kombinationen der Methoden zu starten.

#### 4.4.3 Methodenvergleich

Der Tab 'Methodenvergleich' ist in Abb. 4.6 dargestellt. Die Eingabefelder 'Eingabe Ordner' und 'Ausgabe Ordner' bestimmen, aus welchem und in welchen Ordner, relativ zum Speicherort der Anwendung, die Daten gelesen oder geschrieben werden. Im Eingabe Ordner müssen dreidimensionale Bilder, Segmentierungsmasken und Annotationen vorhanden sein. Die vorangegangenen Tabs speichern die Daten direkt im erwarteten Format ab. Unter der Anzeige für die gewählten Ordner sind einige Checkbox-Felder gegeben. Diese sind in die vier Gruppen Encoder, Decoder, Vorverarbeitung (Nucleus-Kanal) und Vortraining unterteilt. Alle hier gewählten Methoden werden nachfolgend genutzt und in jeder Kombination trainiert. Die verfügbaren Methoden sind in Kapitel 3.3 beschrieben. Mit dem 'Start'-Knopf wird der Methodenvergleich gestartet und mit dem 'Abbrechen'-Knopf wieder abgebrochen. Nach Abschluss des Vergleichs werden die Voraussagen des Modells mit der höchsten Genauigkeit auf dem Validierungsanteil der Eingangsdaten für alle Nuclei im Datensatz abgelegt, auch für die Daten ohne Annotationen. Außerdem wird die Kombination von Methoden gespeichert, mit der die höchste Genauigkeit erzielt wurde.

#### 4.4.4 Training

Im 'Training'-Tab (Abb. 4.7) wird der Klassifikator mit der zuvor ermittelten optimalen Kombination von Methoden erneut trainiert in mehr Epochen. Dabei können auch mehr Annotationen eingesetzt werden, die während des Methodenvergleichs nachgeliefert wurden. Dieser Tab ist optional, das Training während dem Methodenvergleich kann das intensivere Training in diesem Tab ersetzen, wenn die Genauigkeit bereits zufriedenstellend war. In den Eingabefeldern 'Eingabe Ordner' und 'Ausgabe Ordner' wird angegeben, welche Ordner genutzt werden sollen. Im Eingabe Ordner muss eine Datei vorhanden sein, die

#### 4) Finales Training

Eingabe Ordner	Demo/Training
Ausgabe Ordner	Demo/Vis

**Abb. 4.7** | Die Abbildung zeigt einen Ausschnitt des Tabs der 3D-Zelldaten-Pipeline, der das finale Training des Klassifikators durchführt. Hier wird ein Ordner gewählt, aus dem die beste Kombination von Methoden gelesen werden soll, und anschließend wird ein längeres Training gestartet.

die beste Kombination der Methoden auszeichnet. Nachdem das Training mit dem 'Starte Training'-Knopf gestartet und vollendet wurde, werden die Vorhersagen für alle Nuclei der Eingabedaten im Ausgabe Ordner abgelegt. Sind hier keine Annotationen verfügbar, wird eine einfache Inferenz mit dem trainierten Modell im Eingabe Ordner durchgeführt und die Vorhersagen gespeichert.

#### 4.4.5 Visualisierung

In dem letzten Tab werden die Ergebnisse der Anwendung dargestellt (siehe Abb. 4.8). Mithilfe der beiden Eingabefelder wird festgelegt, aus welchen Ordnern die Ergebnisse stammen sollen und wohin die erstellten Grafiken gespeichert werden. Mit dem 'Starte Visualisierung'-Knopf wird der Algorithmus gestartet, der die interpretierbaren Eigenschaften aus den Ergebnissen ausliest. Daraufhin werden unten auf der Seite drei Grafiken dargestellt. Links wird beispielhaft eine Instanzsegmentierungsmaske von einer Schicht eines der 3D-Bilder gezeigt. In der Mitte wird ein Balkendiagramm dargestellt, das die Anzahl der Nuclei jeder Klasse angibt. Das rechte Balkendiagramm zeigt die Verteilung der Nucleus-Volumen in Pixeln hoch drei. Um dieses Volumen zu interpretieren, muss die Umrechnung von Pixeln in Micrometer abhängig von der Auflösung des Aufnahmegeräts und der Zoom-Stufe manuell durchgeführt werden.

### 5) Visualisierung

Eingabe Ordner   
 Ausgabe Ordner



**Abb. 4.8 |** Die Abbildung zeigt den 'Visualisierung'-Tab der 3D-Zelldaten-Pipeline. Hier werden die Ergebnisse der Anwendung in interpretierbaren Grafiken dargeboten.



# Ergebnisse 5

---

## 5.1 Überblick

Das nachfolgende Kapitel beschreibt die Ergebnisse der durchgeführten Experimente. Die Experimente sind die quantitative Bewertungen, die die 3D-Zelldaten-Pipeline durchführt. Zuerst werden die Bewertungen der Instanzsegmentierungsmasken anhand der [IPQ](#)-Metrik und dann der Klassifikatoren anhand ihrer Genauigkeit betrachtet. Die [IPQ](#)-Ergebnisse werden hier auch gegliedert in die drei Faktoren der Metrik betrachtet (siehe Kap. 3.2). Daraus wird ersichtlich, dass jedes der Segmentierungsmodelle in einer der Kategorien dominiert, insgesamt aber CellposeSAM den anderen Modellen signifikant überlegen ist. Auch die Ergebnisse der Klassifikation werden unterteilt, um den Einfluss einzelner Methoden auf die Genauigkeit des Klassifikators zu identifizieren.

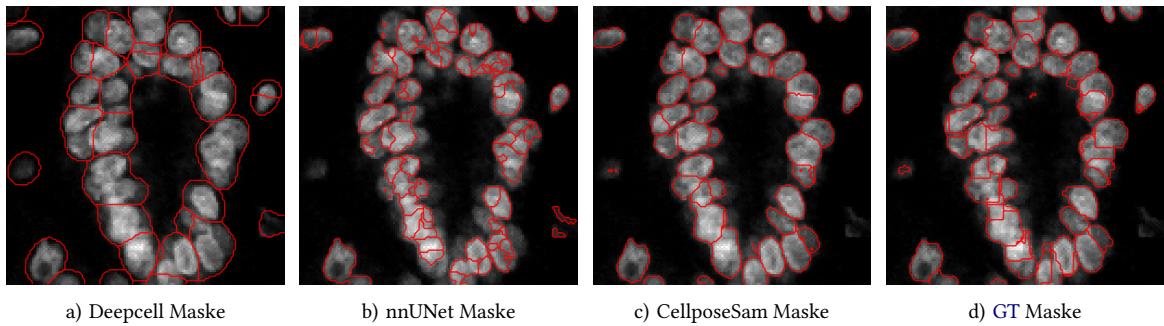
## 5.2 Hardware

Für die Anwendung wurde eine NVIDIA GeForce RTX 3090 Ti mit 24 GB VRAM verwendet. Der verwendete Server verfügt über eine 12th Gen Intel(R) Core(TM) i9-12900KF CPU mit 16 Kernen und 64 GB RAM.

## 5.3 Segmentierung

Für die Wahl eines Segmentierungsnetzes wird in Kapitel 3 das Bewertungskriterium [IPQ](#) eingeführt. Außerdem wird in Kapitel 3 der annotierte S\_BIAD1518Datensatz vorgestellt. Die [IPQ](#) wird auf dem Datensatz mit den Masken von drei vortrainierten Segmentierungsnetzen und, zur Validierung, mit der Ground Truth ([GT](#)) ausgeführt. Die Masken unterscheiden sich optisch stark (siehe Abb. 5.1), was sich auch in starken Unterschieden in der [IPQ](#) äußert.

Die Ergebnisse jedes Segmentierungsnetzes sind einzeln und für jedes Bild im Appendix ?? angehängt. Eine Zusammenfassung der Ergebnisse ist in den Boxplots in Abb. 5.2 gegeben. Das zentrale Ergebnis ist, dass CellposeSAM die besten [IPQ](#)-Werte erzielt. Mit einem Mittelwert von 0,64 ist die [IPQ](#) von CellposeSAM signifikant höher als der Mittelwert bei nnUNet (0,04) und Deepcell (0,02), mit entsprechenden p-Werten von  $1,80 \cdot 10^{-80}$  bzw.  $1,59 \cdot 10^{-81}$  bei einseitigen T-Tests. Dennoch zeigt sich, dass CellposeSAM lediglich in der Kategorie Segmentierungs-Qualität (SQ) sowohl den höchsten Median als auch den



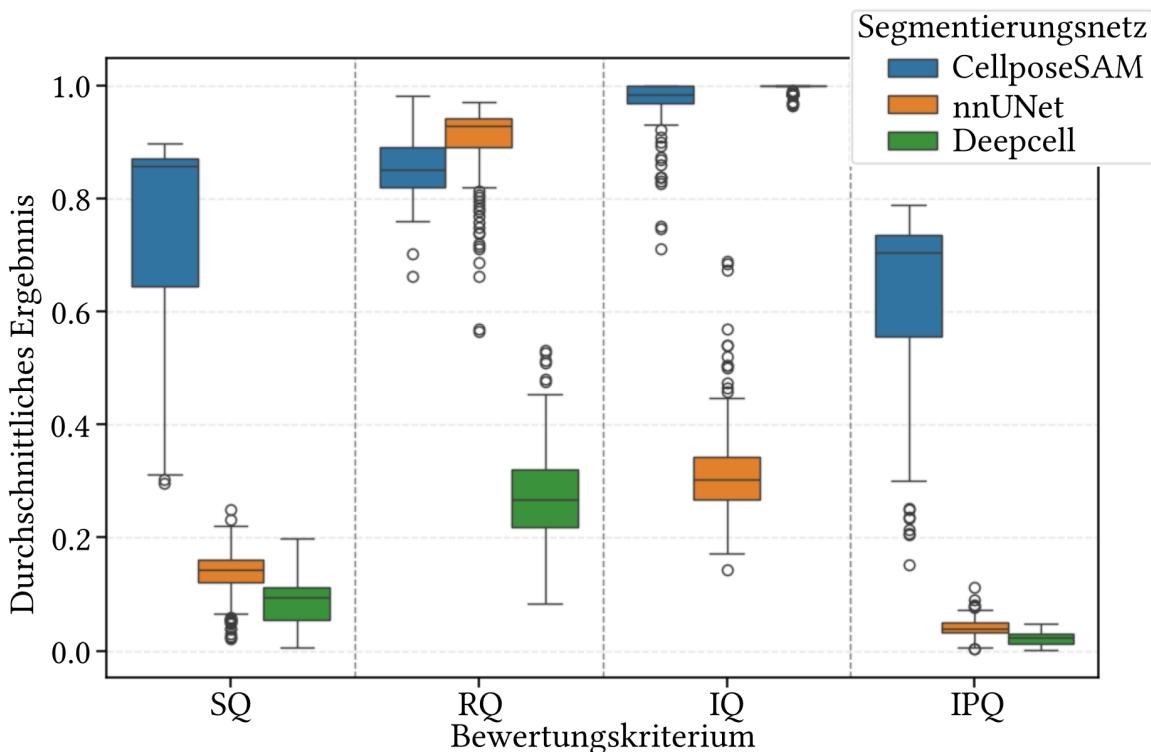
**Abb. 5.1** | Darstellung der Segmentierungsmasken der verschiedenen Segmentierungsnetze als Konturen auf einem zweidimensionalen Durchschnitt einer Stichprobe des S\_BIAD1518 Datensatzes.

höchsten Mittelwert erreicht. Die Recognition-Qualität (RQ) der nnUNet-Masken ist signifikant höher als die der CellposeSAM-Masken ( $p$ -Wert:  $3,48 \cdot 10^{-6}$ ), und ebenso die Injektive Qualität (IQ) der Deepcell-Masken ( $p$ -Wert:  $6,78 \cdot 10^{-8}$ ). Obwohl nnUNet und Deepcell jeweils eine Metrik dominieren, wird ihr IPQ-Wert durch die beiden schlechten Faktoren stark heruntergezogen, während CellposeSAM in jeder Metrik gut, wenn auch nicht am besten, abschneidet. Außerdem zeigen die Boxplots viele Ausreißer in den Daten, was den Unterschieden der Bildkategorien, die der Datensatz enthält, geschuldet sein könnte.

## 5.4 Klassifikation

Die in Kapitel 3 vorgestellten Methoden zur Klassifikation werden anhand eines separaten Anteils des manuell gelabelten Datensatzes getestet. Die Genauigkeit, also der prozentuale Anteil richtiger Vorhersagen, auf dem Test-Anteil des Zieldatensatzes wird als Kriterium verwendet. Da die Modelle während des Trainings rauschbehaftete Verläufe der Genauigkeit und es zu Overfitting kommen kann, wird nicht das Modell am Ende des Trainings zur Evaluation eingesetzt, sondern das Modell der Trainingsepoke, in der die Genauigkeit am höchsten ist. *TODO: Trainings-Verlauf Kurve einfügen TODO: Text zu den Abbildungen, t-Tests ausführen* In Abb. 5.3 ist die Genauigkeit der Klassifikatoren pro Encoder gegeben.

Abb. 5.4 zeigt eine Gegenüberstellung der Genauigkeit, die die beiden Decoder erreichen anhand eines Balkendiagramms. Die blauen Balken zeigen dabei jeweils die Ergebnisse der Architekturen, die den Schichten-Klassifikator einbeziehen, die orangenen Balken die Ergebnisse des Volumen-Klassifikator. Auf der X-Achse sind als Gruppen die Encoder oder der Durchschnitt aufgetragen und auf der Y-Achse die durchschnittliche Genauigkeit der Modelle mit entsprechendem Decoder. An den Daten ist zu sehen, dass der Einfluss des Decoders nicht homogen ist. Mit dem ConvNeXt Encoder ist der Schichten-Klassifikator performanter als der Volumen-Klassifikator, während es bei allen anderen Encodern geringfügig ist. Für den CellposeSAM Encoder ist der Genauigkeitsunterschied 12,6 Prozentpunkte groß, für EfficientNetV2 lediglich einen halben Prozentpunkt. Die Daten zeigen also außerdem, dass die Intensität des Unterschieds zwischen den Genauigkeiten der beiden Decoder stark vom Encoder abhängt. Die Balken rechts, die je den Durchschnitt aller Mo-

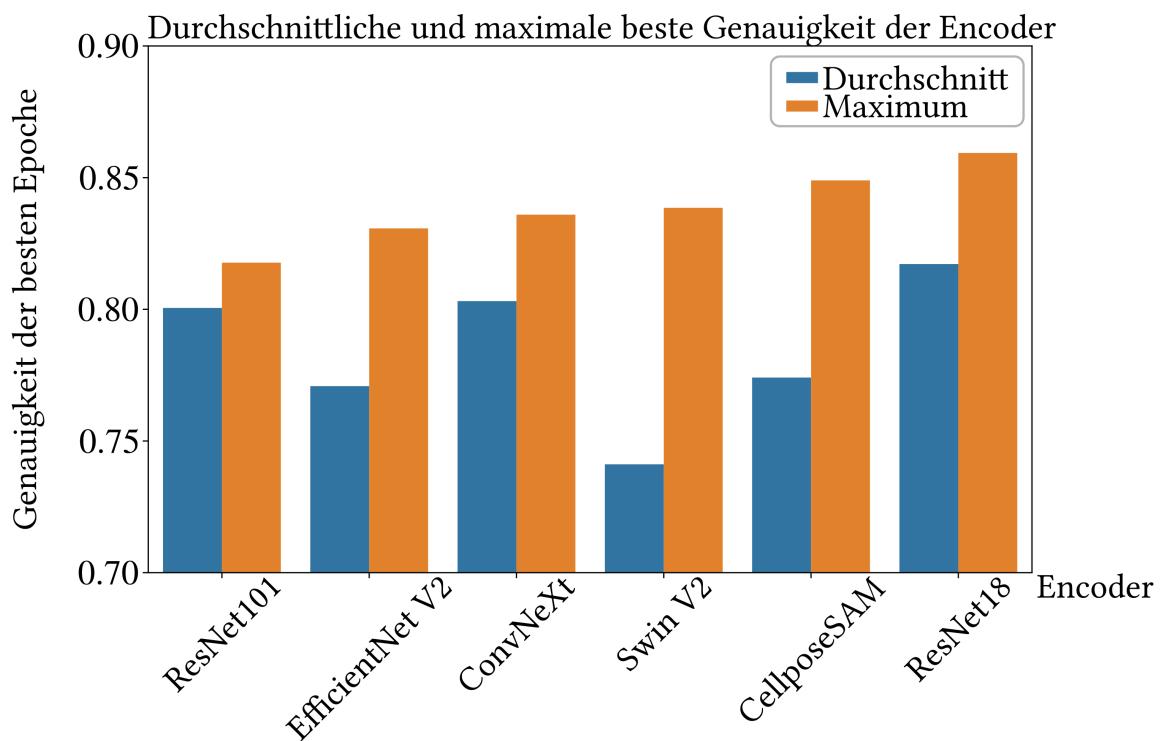


**Abb. 5.2 |** Boxplots der Ergebnisse der IPQ Berechnungen mit Faktoren  $k_1$ ,  $k_2$  und  $k_3$  jeweils gleich eins. Die X-Achse unterteilt die Daten in die Kriterien Segmentatierungs-Qualität (SQ), Recognition Qualität (RQ), Injektivitäts-Qualität (IQ) und Injektive Panoptische Qualität (IPQ), wie in der Formel 3.2 beschrieben. Für jede Metrik sind drei farbige Boxplots zu sehen, einer für jedes Segmentierungsnetz. Die Boxplots visualisieren hierbei die Verteilung der Metriken. Die Box repräsentiert das Interquartilsintervall (25.–75. Perzentil), wobei der Median als Linie innerhalb der Box dargestellt ist. Die sogenannten Whisker reichen bis zum 1,5-fachen des Interquartilsabstands über die Box hinaus. Darüber hinausgehende Punkte gelten als Ausreißer und werden einzeln dargestellt.

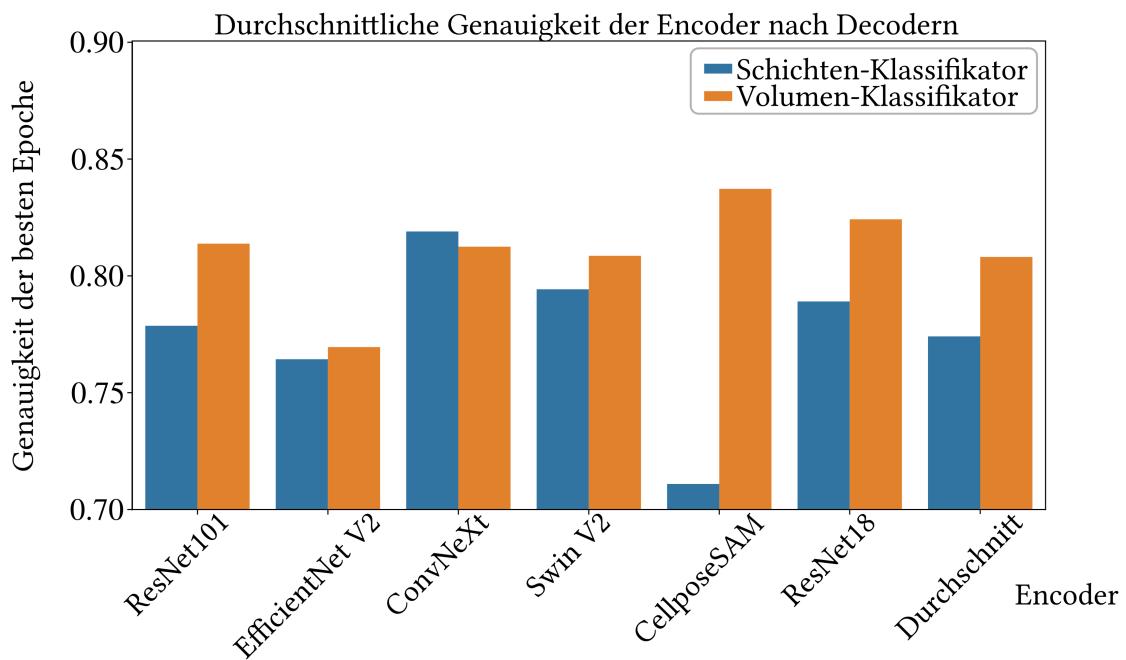
delle mit einem Decoder darstellen, deuten an, dass der Volumen-Klassifikator insgesamt bessere Ergebnisse liefert. Mit einem t-Test wird diese Vermutung bestätigt ( $p = 0,023$ ).

In Abb. 5.5 sind die durchschnittlichen Genauigkeiten der Modelle gegeben, die jeweils eine der Vorverarbeitungsmethoden nutzen. Auf der Y-Achse ist die Genauigkeit aufgetragen, auf der X-Achse die Encoder und der Durchschnitt als Gruppen. Die Durchschnittsbalken geben den Durchschnitt aller Modelle mit einer bestimmten Vorverarbeitung an, unabhängig vom Encoder. Die Ergebnisse zeigen deutlich, dass der Vorverarbeitungstyp Eins, also das Ersetzen des Nucleus Kanals mit der Segmentierungsmaske dem Anwenden einer Distanztransformation auf den Nucleus Kanal überlegen ist. Durch einen stark signifikanten t-Test wird diese Aussage bestätigt ( $p = 0,0018$ ). Zu sehen ist aber, dass der Einfluss der Vorverarbeitung unterschiedlich stark ist, je nach Encoder. Während die Genauigkeit für Modelle mit dem ResNet101 Encoder mit dem Vorverarbeitungstyp Eins um 0,8 Prozentpunkte steigt gegenüber der Verwendung von Typ Zwei, macht die Vorverarbeitung bei Nutzen des EfficientNetV2 Encoders 11,5 Prozentpunkte aus.

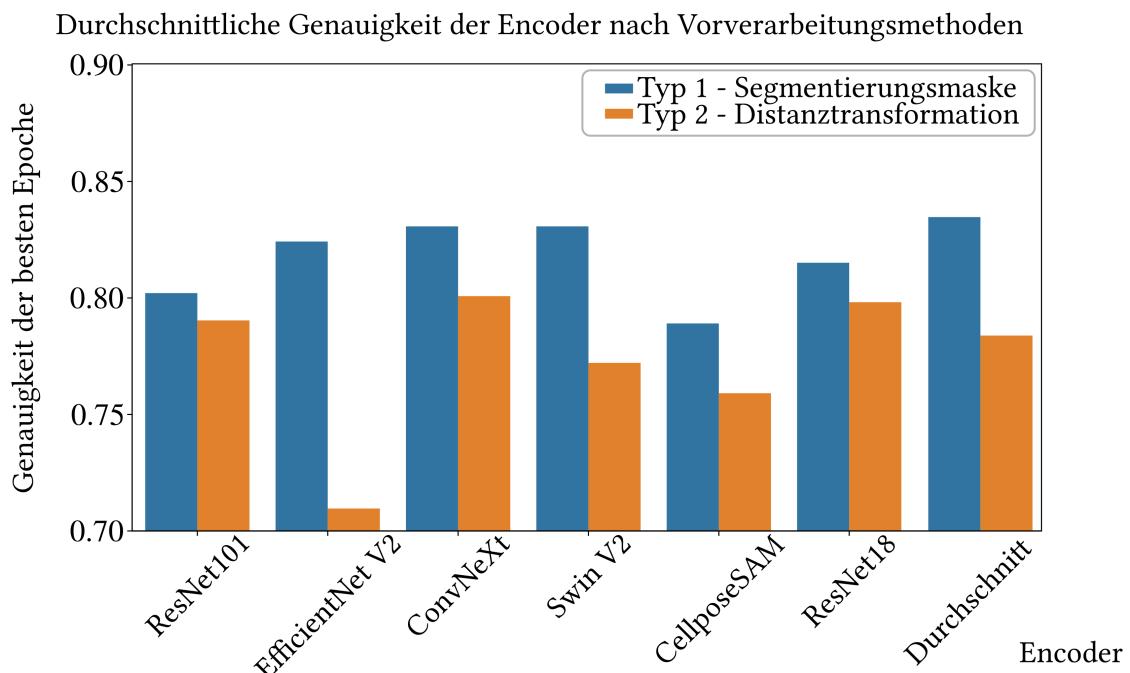
Abb. 5.6 zeigt eine Übersicht der Effektivität der verwendeten Vortrainingsmethoden.



**Abb. 5.3 |** Das Balkendiagramm zeigt an der Y-Achse die Genauigkeit der Klassifikatoren unter Verwendung der verschiedenen Encoder. Auf der X-Achse sind die Encoder als Gruppen aufgetragen. Jede Gruppe enthält einen Maximalwert (Orange) und einen Durchschnittswert (Blau), da jeder Encoder mit verschiedenen Kombinationen von Methoden getestet wird. Die Encoder sind nach aufsteigendem Maximalwert von links nach rechts sortiert.

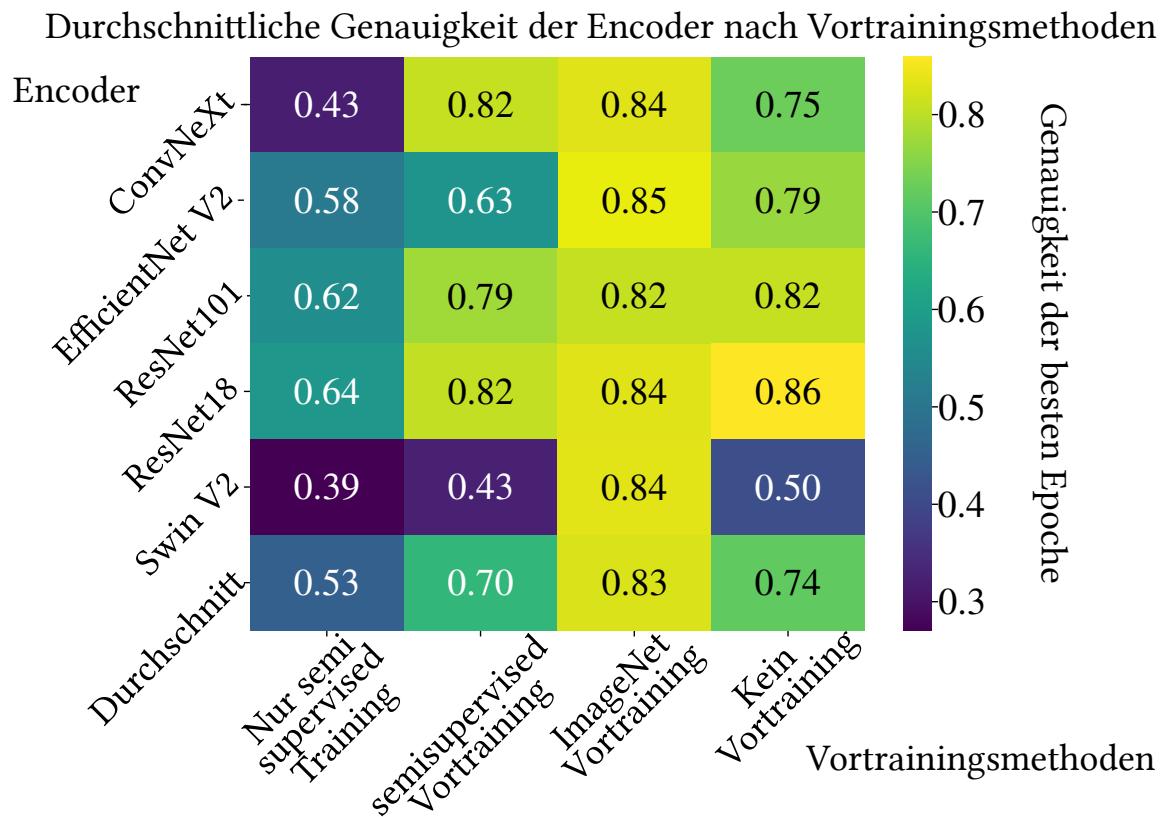


**Abb. 5.4 |** Das Balkendiagramm zeigt den Genauigkeitswert der Klassifikatoren unter Verwendung eines bestimmten Decoder. Auf der Y-Achse ist der Durchschnitt der Genauigkeiten der besten Trainingsepochen aller Klassifikatoren, die diesen Decoder nutzen, aufgetragen. Die X-Achse zeigt Gruppen von Encodern, jeweils besetzt mit einem Wert für den Schichten- und den Volumen-Klassifikator. Rechts zu sehen sind Balken, die den Durchschnitt der Werte aller Encoder, abhängig von dem Decoder, zeigen.



**Abb. 5.5** | Das Balkendiagramm zeigt den Genauigkeitswert der Klassifikatoren unter Verwendung einer bestimmten Vorverarbeitungsart. Auf der Y-Achse ist der Durchschnitt der Genauigkeiten der besten Trainings-epochen aller Klassifikatoren, die diese Vorverarbeitungsart nutzen, aufgetragen. Die X-Achse zeigt Gruppen von Encodern, jeweils besetzt mit einem Wert für den Typ Eins, der den Nucleus-Kanal des Bilds mit der Segmentierungsmaske ersetzt, und Typ 2, der eine Distanztransformation auf den Nucleus-Kanal anwendet. Rechts zu sehen sind Balken, die den Durchschnitt der Werte aller Encoder, abhängig von der Vorverarbeitungsmethode, zeigen.

Auf der vertikalen Achse sind die verschiedenen Encoder, sowie der Durchschnitt zu sehen. Auf der horizontalen Achse sind die vier Vortrainingsmethoden aufgetragen. Die Farbe der Felder gibt die Genauigkeit einer Kombination von Encoder und Vortrainingsmethode, beziehungsweise den Durchschnitt aller Modelle die mit der entsprechenden Methode trainiert wurden, an. Hierbei reicht die Farbskala von hellem Gelb (bestes Ergebnis) bis zu dunklem Blau (schlechtestes Ergebnis). In den Feldern ist außerdem der entsprechende Genauigkeitswert eingetragen. Zu sehen ist, dass das Vortraining mit semi-supervised-Daten schlechtere Ergebnisse liefert, als ImageNet- oder gar kein Vortraining. Im Durchschnitt liefert das Fortführen des Trainings mit den annotierten Daten nach dem semi-supervised-Training zu einer Steigerung der Genauigkeit um zehn Prozentpunkte. Außerdem ist zu sehen, dass die beiden ResNet Encoder gut mit dem semi-supervised-Vortraining umgehen, während alle anderen Encoder schlechte Ergebnisse zeigen. Das kleinere ResNet18 Modell erreicht 64% und das ResNet101 Modell 62% Genauigkeit nach ausschließlich semi-supervised-Lernen. Beide Modelle werden noch um mehr als zehn Prozentpunkte besser nach fortgesetztem Training mit limitierten Annotationen. ImageNet-Vortraining liefert im Durchschnitt die besten Ergebnisse mit einer Genauigkeit von 84%. Außerdem ist die Varianz entlang der Encoder mit dem ImageNet-Vortraining ( $5,6 \cdot 10^{-5}$ ) deutlich geringer als ohne Vortraining (0,016). Dennoch erreicht ResNet18 ohne Vortraining die beste Genauigkeit mit 86%.



**Abb. 5.6** | Die Heatmap zeigt die Genauigkeiten der Klassifikatoren unter Verwendung der verschiedenen Vortrainingsmethoden. Auf der Y-Achse sind die verschiedenen Encoder aufgetragen. Die X-Achse stellt die getesteten Methoden des Vortrainings dar. Die Farbe der Felder und der Wert darin zeigen die Durchschnitte der Genauigkeiten aller Klassifikatoren, die die entsprechende Vortrainingsmethode nutzen. Die letzte Zeile zeigt den Durchschnitt der Werte aller Encoder, abhängig von der Vortrainingsmethode.

# Diskussion 6

---

## 6.1 Überblick

## 6.2 Segmentierung

Durch einen Vergleich der Masken mit der **GT** in Abb. 5.2 und ihren zugehörigen Ergebnissen der einzelnen Bewertungskriterien sind die Schwächen und Stärken der individuellen Netze ersichtlich. Die nnUNet-Masken sind sichtbar kleiner als die Nucleus-Instanzen, was eine schlechte Segmentierungsqualität bedingt. Oft zerteilen mehrere nnuNet-Masken eine Nucleus-Instanz, was von der Injektiven Qualität bestraft wird. Wie auch die gute Recognition Qualität zeigt, findet dafür nnUNet sehr zuverlässig die anwesenden Nuclei mit mindestens einer Maske. Deepcell (siehe Abb. 5.1a) übersegmentiert die Nuclei, wodurch die Segmentierungsqualität stark abnimmt. Das Ergebnis sind Masken, die zu groß sind und oft mehr als einen Nucleus enthalten. Das bedeutet auch, dass einige Nuclei nicht von einer eigenen Maske gefunden werden, was sie als **FN**-Instanzen kategorisiert und eine schlechte Recognition Qualität bedingt. Durch diese Übersegmentierung wird vermieden, dass Instanzen der **GT** durch die Deepcell-Masken geteilt werden, was zu einer guten Injektiven Qualität führt.

## 6.3 Klassifikation

Die Interpretation der Ergebnisse der Genauigkeit von Klassifikatoren mit verschiedenen Encodern, Decodern, Vorverarbeitungsmethoden und Vortrainingsmethoden legt einige Aussagen über Effektivität der Methoden nahe.

Der Volumen-Klassifikator ist signifikant besser als der Schichten-Klassifikator (siehe Kapitel 5.4). Dieses Erkenntnis legt nahe, dass die dreidimensionalen Faltungsschichten besser den Zusammenhang der Merkmale erfassen können, als der Attention-Mechanismus des Schichten-Klassifikators. Das könnte daran liegen, dass die dreidimensionale Faltung räumliche Durchschnittswerte nur mit gelernten Gewichten durchführt, während die Spatial Average Operation großflächig die räumliche Beziehung der Merkmale vernachlässigt. Außerdem ist es möglich, dass die für die Klassifikation ausschlaggebenden Informationen nicht ausreichend entlang der Z-Dimension erhalten sind, was zu einer schlechten Leistung des tiefen fokussierten Schichten-Klassifikators führen würde. Stattdessen könnte auch die Kombination räumlicher Merkmale und der Merkmale entlang der Z-Achse wichtig sein, was die gute Leistung des Volumen-Klassifikators bedingen würde. Da der

Schichten-Klassifikator mit dem ConvNeXt Encoder im Durchschnitt höhere Genauigkeiten erzielt, als mit dem Volumen-Klassifikator, ist die Abwägung zwischen den beiden Methoden dennoch für zukünftige Anwendungen sinnvoll. Bestimmte Encoder erfassen die relevanten Merkmale auf unterschiedliche Weise und beide Decoder haben das Potential mit bestimmten Merkmalsräumen besonders gut zu synergieren.

Auch für Vorverarbeitung ist eine signifikant bessere Methode zu erkennen. Den Nucleus Kanal mit der Segmentierungsmaske zu ersetzen ist signifikant besser als eine Distanztransformation auf den Kanal anzuwenden. Kapitel 3.3 beschreibt das Risiko der Distanztransformation ist der Fortbestand von Einfluss umliegender Nuclei auf das Klassifikationsergebnis.

...

...

# Zusammenfassung

7

---

## 7.1 Überblick

## 7.2 Zusammenfassung

**KI** Künstliche Intelligenz

**GAN** Generative Adversarial Network

**GUI** Graphical User Interface

**SAM** Segment Anything Model

**GT** Ground Truth

**IoU** Intersection over Union

**PQ** Panoptic Quality

**IPQ** Injektive Panoptische Qualität

**TP** True Positive

**FP** False Positive

**FN** False Negative

**CNN** Convolutional Neural Network

**ViT** Vision Transformer

## **Erklärung**

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Karlsruhe, den 25. Oktober 2025

.....