



Karlsruher Institut für Technologie

Fakultät für Maschinenbau

Institut für Automation and angewandte Informatik

Segmentierung und Merkmalsextraktion in 3D-Mikroskopiedaten

Masterarbeit in Elektrotechnik und Informationstechnik

eingereicht von

David Exler

Erstprüfer: Prof. Dr. Gerardo Hernandez-Sosa

Zweitprüfer: apl. Prof. Dr. Markus Reischl

2025

Abstract Abstract.

Zusammenfassung Deutscher Abstract.

Inhaltsverzeichnis

| | |
|--|-----------|
| 1 Einleitung | 1 |
| 1.1 Überblick | 1 |
| 1.2 Allgemeine Beschreibung | 1 |
| 1.3 Offene Probleme | 1 |
| 1.4 Zielsetzung | 2 |
| 1.5 Strukturierung der Arbeit | 2 |
| 2 Theory | 3 |
| 2.1 Überblick | 3 |
| 2.2 Methoden | 3 |
| 2.2.1 Unfertiger Zwischendurch-Text! | 3 |
| 2.2.2 Benchmark | 4 |
| 2.2.3 Segmentierung | 4 |
| 2.2.4 Klassifikation | 6 |
| 2.3 Literaturrecherche | 7 |
| 2.3.1 Benchmark | 7 |
| 2.3.2 Segmentierungsnetz | 8 |
| 2.3.3 Klassifikator | 10 |
| 2.4 Offene Probleme | 12 |
| 2.5 Zielsetzung | 12 |
| 3 Neues Konzept | 13 |
| 3.1 Überblick | 13 |
| 3.2 Bewertungskriterien | 14 |
| 3.3 Klassifikator | 16 |
| 3.3.1 Encoder | 16 |
| 3.3.2 Vorverarbeitung | 17 |
| 3.3.3 Vortraining | 18 |
| 3.3.4 Decoder | 20 |
| 4 Implementation | 23 |
| 4.1 Überblick | 23 |
| 4.2 Segmentierungsnetze | 23 |
| 4.3 Labeling App | 23 |
| 4.4 Training | 25 |

Inhaltsverzeichnis

| | |
|-------------------------------|-----------|
| 5 Results | 27 |
| 5.1 Überblick | 27 |
| 5.2 Segmentierung | 27 |
| 5.3 Klassifikation | 28 |
| 6 Diskussion | 29 |
| 6.1 Überblick | 29 |
| 6.2 Segmentierung | 29 |
| 7 Zusammenfassung | 31 |
| 7.1 Überblick | 31 |
| 7.2 Zusammenfassung | 31 |

Einleitung 1

1.1 Überblick

Das nachfolgende Kapitel leitet in das Thema der vorliegenden Thesis ein. Es beschreibt zu diesem Zweck das Projekt allgemein und beleuchtet die offenen Probleme, an denen das Projekt ansetzt. Davon ausgehend erläutert das Kapitel die Zielsetzung der Arbeit. Zuletzt wird die Struktur der schriftlichen Ausarbeitung erklärt.

1.2 Allgemeine Beschreibung

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 umliegenden Strukturen eingefärbt und unter dem Mikroskop analysiert [9, 10, 11]. Die Bilddaten entstehen dabei mit unterschiedlichen Herstellungsbedingungen, Färbungen und Aufnahmegeräten [12, 13, 14, 15], was eine generalisierte Automatisierung erschwert.

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 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.

1.3 Offene Probleme

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].

1.4 Zielsetzung

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. Um die Segmentierung der Myotuben zu ermöglichen, werden verschiedene Zwischenziele definiert, die zusätzliche, unterstützende Informationen sammeln, welche den Segmentierungsalgorithmen zur Verfügung gestellt werden. Die Zwischenziele umfassen:

- Die Segmentierung von individuellen Zellkernen in den dreidimensionalen Daten.
- Das Klassifizieren von Zellkernen anhand ihrer Morphologie sowie der antikörperbasierten Marker in ihrer unmittelbaren Umgebung.
- Das Kodieren der gewonnenen Informationen auf eine Weise, die sie für bestehende Segmentierungsalgorithmen nutzbar macht.

1.5 Strukturierung der Arbeit

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 durchgeföhrten 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. (TODO Öffentlich stellen)

Theory 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 Unfertiger Zwischendurch-Text!

Bilineare Interpolation ist ein Verfahren zur Bildnachbearbeitung, das die Dimension eines Bilds erhöht, indem Werte für neue Pixel zwischen bestehenden geschätzt werden [28]. Zur Schätzung des neuen Wert wird dabei ein gewichtetes Mittel aus den Vier benachbarten Pixeln genommen:

$$\hat{f}(x, y) = (1 - p)(1 - q)f_{i,j} + p(1 - q)f_{i+1,j} + (1 - p)qf_{i,j+1} + pqf_{i+1,j+1}, \quad (2.1)$$

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

Normierungsmethoden:

Batch Normalization normalisiert die Eingaben einer Schicht über eine Mini-Batch, indem für jeden abstrakte Merkmal der Schicht, das bei der Dimensionsreduktion des Bilds entsteht, eine Transformation durchgeführt wird [29]. Für die Transformation werden der Mittelwert des Mini-Batch vom Wert abgezogen und durch die Standardabweichung geteilt. Layer Normalization verfolgt einen ähnlichen Ansatz, berechnet Mittelwert und Varianz aber pro Schicht von allen Neuronen [30]. Beide stabilisieren das Training tiefer neuronaler Netze und lassen die Normalisierung als Bestandteil der Modellarchitektur lernen, statt sie nur als Preprocessing-Schritt durchzuführen [31, 32].

Cross-Entropy-Loss:

Der Cross-Entropy Loss [33]

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^N -\log p(\tilde{y} = \tilde{y}_n | x_n, \theta), \quad (2.2)$$

ist die etablierte Loss Funktion für das Training von Klassifikatoren [34, 35]. Ein Problem der Cross-Entropy Loss Minimierung ist ihre Anfälligkeit für Rauschen der Ground Truths (GTs). Viele verschiedene Ansätze in der Forschung gehen diesen Problem an [36, 37, 38]. Eine häufig genutzte Methode ist die Minimierung des Generalized Cross Entropy Loss:

$$\arg \min_{\theta, w \in [0,1]^n} \sum_{i=1}^n w_i \mathcal{L}_q(f(x_i; \theta), y_i) - \mathcal{L}_q(k) \sum_{i=1}^n w_i, \quad (2.3)$$

für das Klassifikator-Training [39].

2.2.2 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 eine GT 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 [40]. 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 maßgeschneidert zur Anwendung passen und sind bereits breit erforscht [41, 42, 43]. Daten können mithilfe passender Metriken *Ähnlichkeits*-gruppen, sogenannten *Domänen*, zugeordnet werden [44]. Beispiele für Domänenunterschiede in biomedizinischen Bildaufnahmen sind verschiedene Farb-Marker, Aufnahmegeräte oder Zoom-Stufen (siehe Abb. 2.1). 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 [48, 49]. Hierzu kann ein passender Datensatz gewählt werden oder eine Domänenanpassung durchgeführt werden [50, 51, 52]. Ghosh et al. definieren Domänenanpassung: "Gegeben Quell- und Ziel-Domänen D_s und D_t , sowie die Aufgaben τ_s und τ_t , zielen Domänenadaptations-basierte Verfahren darauf ab, ein Modell mit Parametern θ zu erlernen, das für die Zielaufgabe geeignet ist, wenn $D_s \neq D_t$ und $\tau_s = \tau_t$." [53].

2.2.3 Segmentierung

Segmentierung ist die Aufgabe, Pixel mit semantischen Labels zu klassifizieren (semantische Segmentierung [54]), einzelne Objekte voneinander abzugrenzen (Instanzsegmentie-

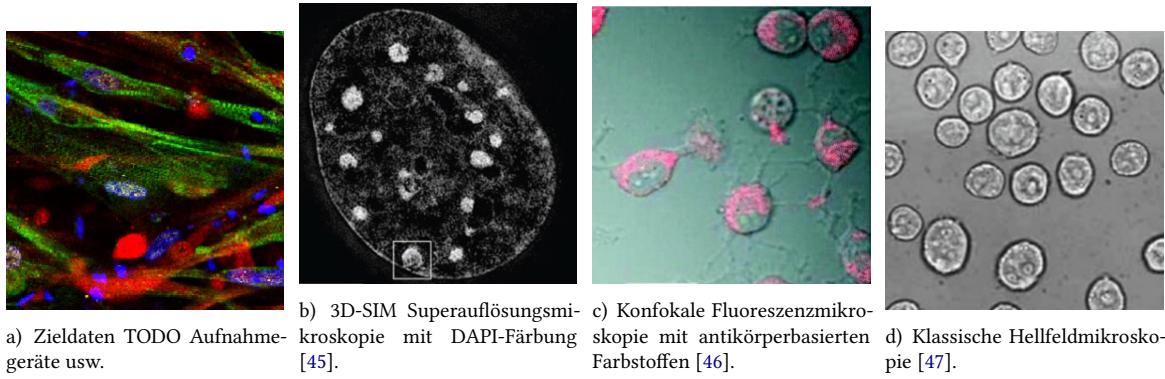


Abb. 2.1 | Vier Aufnahmen von Zellen. Die Bilddomänen sind durch die verschiedenen Marker und Aufnahmetechniken klar zu unterscheiden [45, 46, 47].

rung [55]) oder beides zu kombinieren (panoptische Segmentierung [56]) [57]. In Abb. 2.2 [56] sind beispielhaft GTs der verschiedenen Arten von Segmentierung zu sehen. Für die

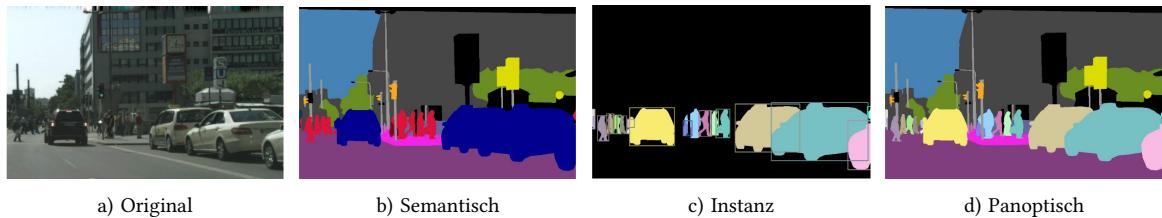


Abb. 2.2 | Die verschiedene Arten der Segmentierung. Links ist das Originalbild zu sehen, rechts folgend sind die zugeordneten pixelweisen Labels farblich eingezeichnet. Gleiche Farben bedeuten gleiche Labels. In der semantischen Maske sind gleiche Labels mehrerer Objekte von semantisch gleichen Klassen zu finden. In der Instanz Maske ist jedem Objekt ein individuelles Label zugeordnet, ungeachtet der Klasse des Objekts. In der panoptischen Maske sind auch einzelne Objekte getrennt, den Labels verschiedener Klassen werden allerdings noch semantische Klassen zugeordnet [56].

verschiedenen Segmentierungsarten werden Architekturen an die Aufgabe angepasst [58, 59]. Modelle sind dabei in der Regel nach dem Vorbild des *U-Net* [60] aus einem Merkmalsextraktor (Encoder) und einem Vorhersagenetz (Decoder) aufgebaut [61]. Der Encoder nutzt beispielsweise Bildfaltungen mit Kernen, deren optimale Gewichte anhand von annotierten Daten gelernt werden, zur Merkmalsextraktion [62]. 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 [63, 64], die räumliche Auflösung schrittweise wieder an, indem er die Bilddimensionen vergrößert und die Merkmalskanäle gleichzeitig reduziert [65]. Über sogenannte Skip-Connections [60] 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 [66, 67].

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

2.2.4 Klassifikation

- Was ist Klassifikation generell?
- Was ist Klassifikation durch panoptische Segmentierung?
- Welche klassischen Ansätze gibt es?
- Welche KI-Lösungen gibt es?

Klassifikation beschreibt das Zuordnen einer Kategorie oder Klasse zu der eine gegebene Stichprobe gehört [68]. Hierzu werden die Merkmale des Objekts, das in der Stichprobe präsentiert wird, durch Beobachtung oder Messung gewonnen [69]. 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 [70, 71]. 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 [72]. Zur Merkmalsextraktion werden klassisch Metriken zu einem Vektor kombiniert [73]. Diese Metriken können beispielsweise die Verteilung der Farbkanäle, eine Charakterisierung der Textur, die Fläche des Objekts und vieles mehr sein [74, 75]. Eine weitere verbreitete Metrik ist eine Kombination von Parametern der Fourier-Entwicklung einer Kontur, die aus der diskreten komplexen Zahlenfolge

$$c[n] = x[n] + i \cdot y[n] \quad (2.4)$$

mithilfe der diskreten Fourier-Transformation

$$F[k] = \sum_{n=0}^{N-1} c[n] \cdot e^{-2\pi i \frac{kn}{N}} \quad (2.5)$$

gebildet werden [76, 77]. Merkmalsvektoren werden häufig abstrahiert und in ihrer Dimension reduziert, beispielsweise durch eine Principal Component Analysis [78, 79]. Sind keine Beispiele von Stichproben der verfügbaren Klassen verfügbar, werden diese Metriken zum Clustering verwendet [70, 80]. Sind nur wenige GTs vorhanden können semi supervised Verfahren angewandt werden, die besonders die Ähnlichkeit der Stichproben ohne Labels herausarbeiten [81, 82]. Eine prominente Methode des semi supervised Lernens ist das label spreading, die mithilfe einer Kernfunktion [83, 84] die Dimension von Merkmalsvektoren ändert und in einen alternativen Merkmalsraum transformiert [85]. Verschiedene Kernfunktionen wie Radiale Basis Funktionen [86]

$$\phi(x, y) = \exp(-\gamma \|x - y\|^2), \quad \gamma > 0 \quad (2.6)$$

werden für das label spreading eingesetzt [87]. Die meist genutzten Methoden der Klassifikation sind Logik-basierte Ansätze wie Entscheidungsbäume, Perzepton-basierte Ansätze wie Neuronale Netze, statistische Ansätze wie Bayes'sche Netzwerke oder Nächster-

Nachbar Verfahren und Support Vector Maschinen [88]. Diese Methoden basieren direkt auf Ähnlichkeiten zwischen den Metriken von unbekannten Stichproben und Stichproben mit bekannter Klasse [89]. Moderne Anwendungen nutzen zur Merkmalsextraktion verschiedene Deep Learning basierte Methoden [90]. Vor allem Convolutional Neural Networks (CNNs) [91] und Vision Transformers (ViTs) [92] sind in der Lage aus Bildern aussagekräftige, abstrakte Merkmale zu generieren [93]. Ein Netz, das zur Merkmalsextraktion eingesetzt wird, wird als **Encoder** bezeichnet. Als **Decoder** wird der klassifizierende Teil des Klassifikators bezeichnet. Der State-of-the-Art für den Decoder ist ein Neuronales Netz, das auf Basis der abstrakten Merkmale des Encoder eine Zuversichtlichkeit für jede Klasse ausgibt [94]. Hierzu lernt in der Regel ein Multi-Layer-Perzepton auf Basis von Trainingsdaten mit zugehöriger **GT** den Zusammenhang zwischen den Merkmalen und der assoziierten Klasse [95].

- . 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 **GT** für die Segmentierung von Zelldaten mit erheblichem manuellem Aufwand verbunden ist und zusätzlich Expertenwissen voraussetzt, sind Datensätze hierfür selten. Einige prominente Datensätze, 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 incucyter HD 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 Mikroskopiemodalitäten, 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 enthalten. Mit Hilfe von SpCycleGAN [105] wurden dazu auf Basis von simulierten GTs Bilder generiert, die anstreben die Merkmale der realen Bilder zu reproduzieren. Es handelt sich hierbei um 3D-multispektrale Daten, aufgenommen mittels Fluoreszenzbildgebung.

Aufgrund des geringen Volumen frei zugänglicher Daten sind diese Sammlungen auch für das Training von Segmentierungsnetzen begehrt. Jeder Datensatz, der bereits im Trainingssatz eines gewählten Segmentierungsnetzes enthalten war eignet sich nicht mehr zur unabhängigen Bewertung der Netze. Neben annotierten Datensätzen bietet die Literatur auch Methoden zum eigenständigen Erzeugen domänenspezifischer Datensätze [106, 107, 108, 109, 110]. Das cell synthesis Projekt [111] ermöglicht das synthetische Generieren von 3D-Trainingsdaten mit realistischer Zellform und -ausrichtung und umgebenden Markern. Zudem umfasst das Framework eine Trainingsroutine, um ein Generative Adversarial Network (GAN) zu trainieren, das Bilddaten und passende Annotationen automatisch erzeugt und an eine gewünschte Bilddomäne anpasst.

2.3.2 Segmentierungsnetz

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 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) [119] von Meta AI. Es besteht aus einem Bild Encoder, einem Prompt Encoder und einem Masken Decoder (siehe Abb. 2.3). 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] Model 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

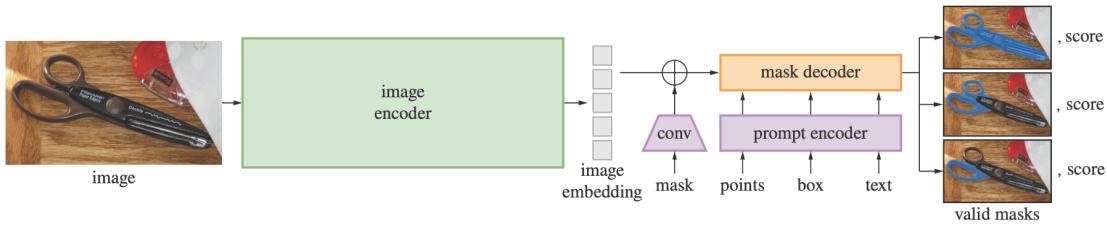


Abb. 2.3 | [119]. Architektur des **SAM**. Eingabebilder werden durch einen Bild Encoder zu Repräsentationen umgeformt. Zusätzliche optionale Hinweise auf das zu segmentierende Objekt werden durch Bildfaltungen oder einen Prompt Encoder repräsentiert. Anschließend prädiziert der Decoder mehrere mögliche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

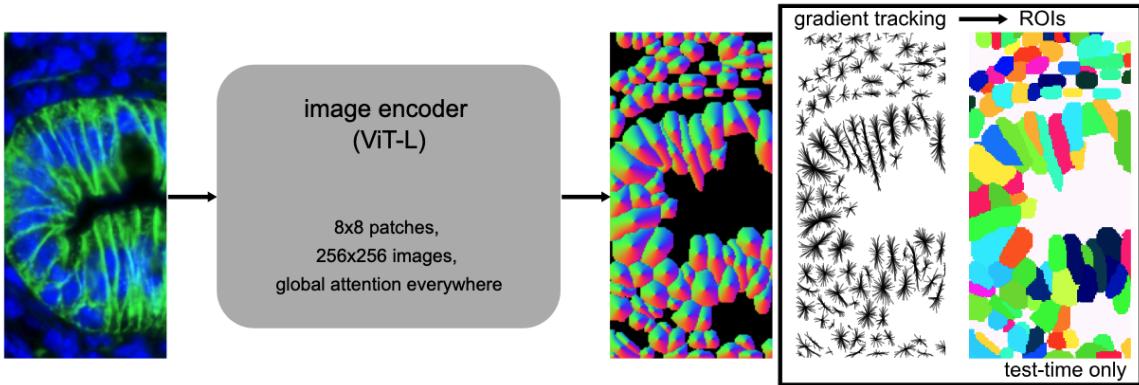


Abb. 2.4 | [127]. Ablauf des CellposeSAM Modells. Eingabebilder werden durch einen Bild Encoder direkt zu *Flows* umgeformt. Die Gradienten der *Flows* werden verfolgt und aus dem entstehenden Gradientenfeld werden Segmentierungsinstanzen vorhergesagt.

wird.

SAM wurde bereits für viele Mikroskopie-Zelldaten downstream Anwendungen *fine-tuned* [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 Zellsegmentierungsnetze. Das Deepcell-Caliban [131]-Netz 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-

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.

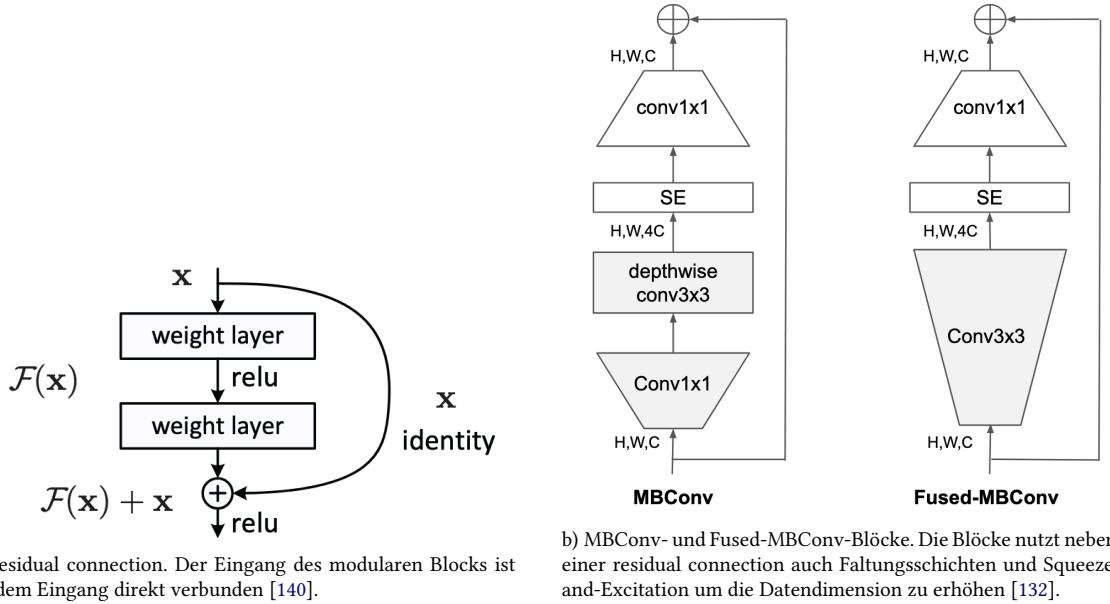
2.3.3 Klassifikator

Für Klassifikatoren werden in der Regel nur Encoder vorgenommen, der Decoder muss den Klassen der downstream Anwendung angepasst werden [113, 134, 93]. 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 (siehe Abb. 2.5a). Es wurde gezeigt, dass residual connections dem Degradierungsproblem [141, 142] von Netzen mit steigender Tiefe entgegenwirken [140]. 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 einer Kombination aus MBConv- [146, 145] und Fused-MBConv-Blöcken [147]. Beide MBConv-Arten nutzen dabei Squeeze-and-Excitation und 1x1 Faltungsschichten. Während MBConv aber die Dimension der Daten mithilfe einer weiteren 1x1 Faltung und 3x3 Depthwise-Faltung erhöht, nutzt der Fused-MBConv-Block hierzu eine reguläre 3x3 Faltung (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] und Layer normalization [30].

Swin Transformer [23] sind eine beliebte Modellfamilie von ViTs. Ihr Nachfolger, die **Swin Transformer V2** [150] vergrößert die Modelle noch. Die Architektur ViTs kombiniert Bildausschnitte mit einem Positions-Bias. Hierzu wird ein Bildfenster z und dessen relativen Koordinaten im Bild Δx und Δy 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 [30] Schichten, ein weiteres MLP-Netz und residual connections vervollständigen anschließend den modularen **Swin Transformer V2** Block. Dieser Aufbau ist in Abb. 2.6 dargestellt.



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.

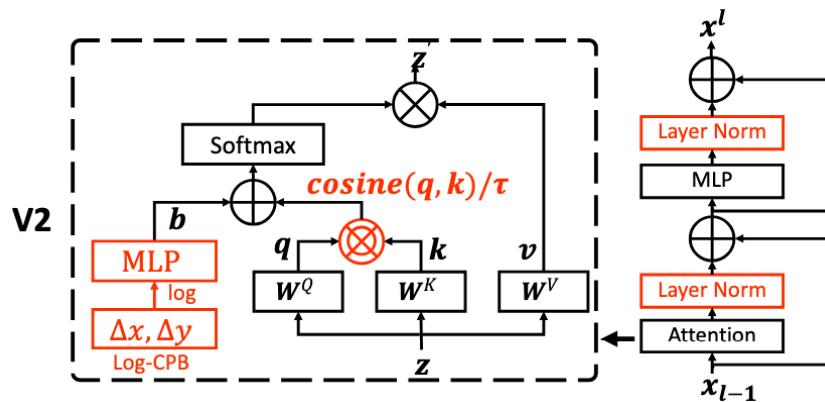


Abb. 2.6 | Swin Transformer V2 Architektur [150]. Ein Bildfenster z und dessen relative Koordinaten im Bild Δx und Δy werden in einem Attention Mechanismus zusammengeführt. Mithilfe einer Kosinus-Ähnlichkeitsfunktion, der Softmax Funktion [151] und elementweiser Multiplikation sowie Addition werden diese Ergebnisse in einen Merkmalsraum überführt. Zwei Layer normalization [30] Schichten, ein weiteres MLP-Netz und residual connections vervollständigen anschließend den modularen **Swin Transformer V2** Block.

2.4 Offene Probleme

2.5 Zielsetzung

Neues Konzept 3

3.1 Überblick

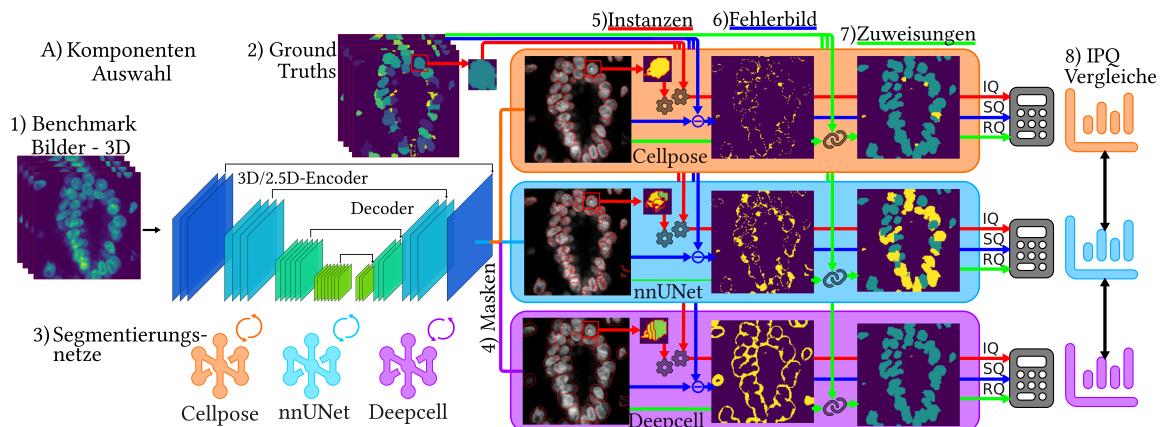


Abb. 3.1 | Graphical Abstract - Das obere Ablaufdiagramm stellt den Prozess dar, durch den das Segmentierungsnetz gewählt wird, das für die Anwendung der vorliegenden Arbeit eingesetzt wird. Ein peer-reviewed Benchmark Datensatz aus dreidimensionalen Bildern (1) von diversen Zellkulturen mit dazugehörigen Ground Truths (2) wird links eingegeben. Mehrere austauschbare Segmentierungsnetze (3) führen eine Inferenz für den Benchmark aus. Die entstehenden Masken (4) werden dann zur Berechnung der neu eingeführten IPQ (siehe Kapitel 3.2) eingesetzt. Aus jeder Maske werden hierzu die einzelnen Instanzen extrahiert, die sich mit einer einzelnen Instanz der Ground Truth überlagern (5). Außerdem wird ein Fehlerbild als die logische XOR Fläche der Maske und der Ground Truth dargestellt (6), als Platzhalter für die Berechnung der Intersection over Union. Zuletzt wird zur Berechnung der IPQ ein Zuweisungsbild erstellt, das die True Positives (TPs), False Positives (FPs) und False Negatives (FNs) festhält (7). Durch Vergleiche der Ergebnisse (8) kann dann das optimale Segmentierungsnetz für die Anwendung gewählt werden.

Das nachfolgende Kapitel beschreibt und diskutiert das angewandte Konzept der vorliegenden Thesis im Detail. Es behandelt vor allem die selbstentwickelten Beiträge zu den Methoden. Mit dem Kapitel sollen alle Verfahren und deren Bewertung klar verständlich sein.

3.2 Bewertungskriterien

Für den Vergleich der Methoden der Klassifikation wird die Genauigkeit des getesteten Netz auf dem manuell annotierten Testanteil des Zieldatensatz verwendet:

$$\text{Genauigkeit} = \frac{1}{N} \sum_{i=1}^N \mathbf{1}\{\hat{y}_i = y_i\}, \quad (3.1)$$

wobei N die Anzahl der Vorhersagen, \hat{y}_i die Vorhersage des Klassifikators und y_i die GT ist. (*Notiz: Ist das hier richtig platziert?*)

Zur Wahl des Segmentierungsnetzes wird ein neues Bewertungskriterium eingeführt und auf einem annotierten Datensatz getestet. Als Datensatz wird der S-BIAD1518 [103, 104] genutzt, da dieser nicht in den Trainingsdaten eines der Segmentierungsnetze 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. Das Kriterium ist eine Abwandlung der Panoptic Quality (PQ) [56], die hier Injektive Panoptische Qualität (IPQ) genannt wird. Ziel dabei ist es, zum einen, durch standard PQ die Intersection over Union (IoU) für individuelle Instanzen zu bewerten und FP sowie FN Detektionen zu bestrafen. Zum anderen sollen Verletzungen der injektiven Abbildung von segmentierten Nuclei auf die Instanzen der GT negativ bewertet werden, da die Anzahl der Nuclei eine bedeutungsvoll Metrik für den Nutzer ist. Für die Berechnung wird im ersten Schritt der nachfolgende Brute Force Algorithmus angewandt, der die Zuordnung von Segmentierungsinstanzen zu GT-Instanzen durchführt.

Algorithm 1 Beste GT-Zuordnung für jede Segmentierunginstanz

```

Require:  $mask_{\text{prediction}}$ ,  $mask_{\text{groundTruth}}$ 
Ensure:  $gt_{\text{opt}}$ ,  $IoU_{\text{opt}}$ 
for  $id_{\text{blob}}$  in  $|mask_{\text{prediction}}|$  do
     $blob \leftarrow mask_{\text{prediction}}[id_{\text{blob}}]$ 
    for  $gt$  in  $mask_{\text{groundTruth}}$  do
         $IoU \leftarrow IoU(gt, blob)$ 
        if  $IoU > IoU_{\text{opt}}[id_{\text{blob}}]$  then
             $IoU_{\text{opt}}[id_{\text{blob}}] \leftarrow iou$ 
             $gt_{\text{opt}}[id_{\text{blob}}] \leftarrow gt\_id$ 
        end if
    end for
end for
return  $gt_{\text{opt}}, IoU_{\text{opt}}$ 
```

Die nachfolgende Formel zeigt das IPQ Bewertungskriterium unterteilt in die 3 Aufga-

ben:

$$\text{IPQ} = \underbrace{\frac{k_1 \times \sum_{(p,g) \in TP} \text{IoU}\left(\bigcup_{p_i \in p} p_i, g\right)}{|TP|}}_{\text{Segmentierungs-Qualität (SQ)}} \times \underbrace{\frac{k_2 \times |TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{Recognition Qualität (RQ)}} \times \underbrace{\frac{k_3 \times |GT|}{\sum_{p \in P} (\max(1, n_p - 1))}}_{\text{Injektivitäts-Qualität (IQ)}}, \quad (3.2)$$

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 **GT**-Instanz und p der Vektor aller zugehörigen Segmentierungsinstanzen ist.
- $|TP| \in \mathbb{Z}$ die Anzahl an korrekt erkannten Instanzen bezeichnet, also **GT**-Instanzen mit $\text{IoU} > 0,5$.
- $\text{IoU}(\bigcup_{p_i \in p} p_i, g) \in [0, 1]$ die **IoU** zwischen der allen Segmentierungsinstanzen p_i in der **TP**-Instanz p und der zugehörigen **GT**-Instanz g beschreibt.
- $|FP| \in \mathbb{Z}$ die Anzahl an falsch-positiven Segmentierungen ist, d. h. vorhergesagte Instanzen ohne **GT**-Entsprechung.
- $|FN| \in \mathbb{Z}$ die Anzahl an nicht erkannten **GT**-Instanzen ist, also **GT**-Instanzen ohne zugehörige Vorhersage.
- $|GT| \in \mathbb{Z}$ die Anzahl der **GT**-Instanzen ist.
- P die Menge aller Segmentierungsinstanzen ist, ungeachtet der **GT**-Zuordnung.
- $p \subseteq P$ ein Vektor aller Segmentierungsinstanzen, die der gleichen **GT**-Instanz 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 der segmentierten und der 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.

- $\text{IPQ} \in [0, 1]$ ein Maß für die panoptische Segmentierungsqualität mit der Voraussetzung von injektiver Abbildung der Segmentierungsmasken-Instanzen auf die **GT**-Instanzen darstellt, wobei höhere Werte bessere Übereinstimmung bedeuten.

3.3 Klassifikator

Jedem instanzsegmentierten Nuclei muss eine Klasse zugewiesen werden, um die Ausgabe zur panoptischen Segmentierungsmaske zu erweitern. Hierzu ist ein Klassifikator notwendig, der einen Bildausschnitt mit der Instanz als Eingabe annimmt und eine der vier Klassen als Ausgabe zuweist. Um diesen Klassifikator optimal zu erstellen wird ein umfangreicher Benchmark aus den Zieldaten erstellt, da bei der Messung der Klassifikationsleistung eine vollkommene Übereinstimmung der Bilddomäne notwendig ist. Benchmarks aus der Literatur umfassen weder dieselben Klassen noch dieselben Objektmerkmale und eignen sich daher nicht. Für das Training wird einheitlich der Adam Algorithmus [152] mit einer Lernrate von 0.0001 eingesetzt. Außerdem wird der Cross-Entropy-Loss [33] 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 beschrieben. 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 Klassifikator Applikationen, 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 vortrainierten 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

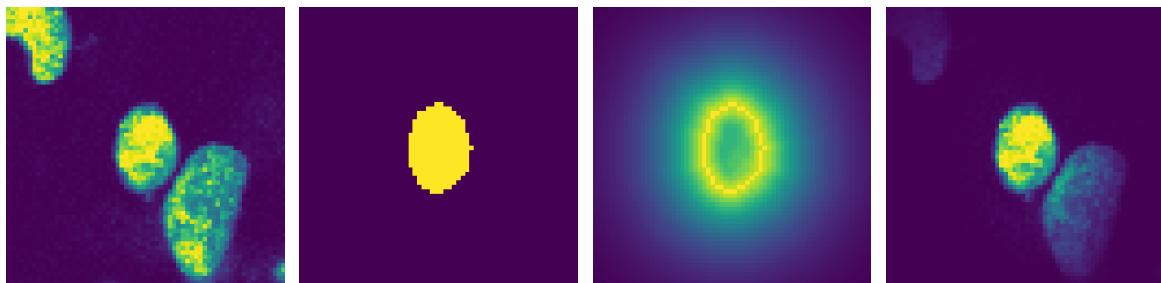
Da in vielen Bildausschnitten Nuclei sehr nah aneinander liegen und dem Klassifikator signalisiert werden muss, welcher der sichtbaren Nuclei klassifiziert werden soll, werden für jeden Klassifikator zwei verschiedene Eingangsdaten-Formate getestet. Diese Typen unterscheiden sich in der Art, wie die Segmentierungsmaske des zu klassifizierenden Nucleus dem Klassifikator zugänglich gemacht wird. Daten des ersten Typs ersetzen 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 Daten des Typ zwei 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:

$$I'(x) = I(x) \cdot \exp\left(-\frac{1}{\sigma} \min_{y \in \neg M} \|x - y\|_2\right), \quad (3.3)$$

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 des Typ zwei hat zum Ziel, dass die Oberflächenmerkmale des Nucleus erhalten bleiben. Außerdem wird mit der Vorverarbeitung des zweiten Typs der Einfluss den eventuelle fehlerhafte Segmentierungsmasken haben 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. Im Folgenden sind die Nuclei Kanäle der verschiedenen Methoden dargestellt.



a) Ausgeschnittener Bereich des originalen Nucleus-Kanals mit mehreren Nuclei.
 b) Segmentierungsmaske des zu klassifizierenden Nucleus, wie Daten des ersten Typs sie enthalten. Die binäre Maske zeigt keine Oberflächenmerkmale des Nucleus, lediglich die geometrischen Merkmale.
 c) Entfernungsmaske des zu klassifizierenden Nucleus. Diese wird pixelweise mit dem Nucleus-Kanal multipliziert für Typ zwei.
 d) Darstellung des Typs zwei, entstanden durch Multiplikation von Nuclei-Kanal und Entfernungsmaske. Zu sehen ist, dass der nahegelegene ungewünschte Nucleus noch stellenweise mit hoher Intensität vertreten ist.

Abb. 3.2 | Darstellungen der verschiedenen Typen von Eingangsdaten. Jeder Typ zielt auf eine andere Art darauf ab, dem Klassifikator zu signalisieren, welcher der sichtbaren Nuclei zu segmentieren ist.

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. * **muss ich das erklären?** *

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. 3.3 zeigt die hier umgesetzten Methoden.

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 Zeildaten durchgeführt. Jedes Modell wird jeweils für 75 Epochen trainiert.

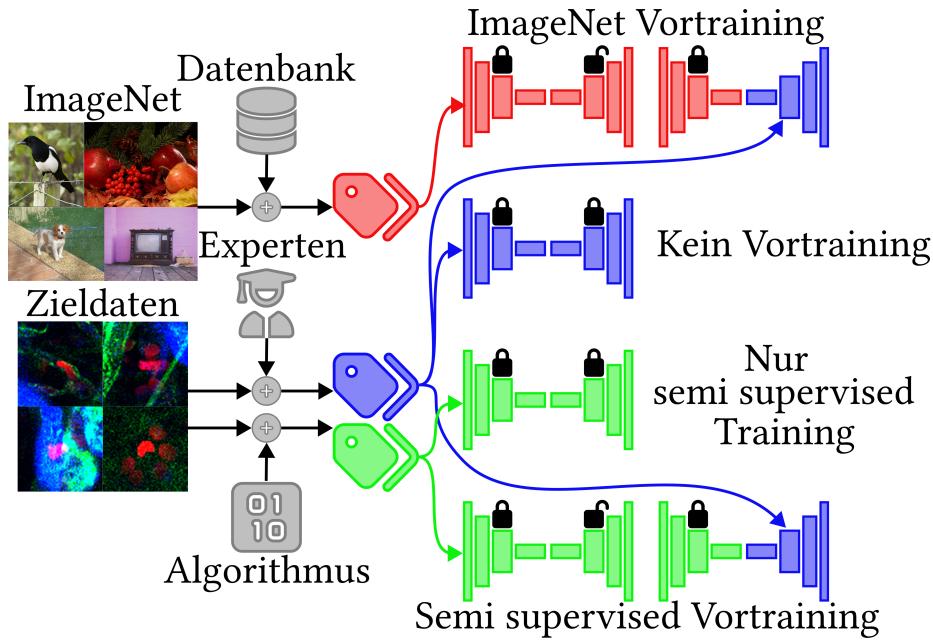


Abb. 3.3 | Übersicht über die Vortrainingsmethoden. Links zu sehen sind die beiden verfügbaren Bildermengen, ImageNet und die Zieldaten. Rechts von diesen Bildermengen werden diesen Bildern **GT** Klassen hinzugefügt, entweder durch die ImageNet Datenbank, Experten oder einen Algorithmus. Jeder Encoder (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 Schlossern 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 vorgenutzt. 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 vorgenutzt und anschließend mit den Zieldaten finetuned.

Semi supervised Die semi supervised **GTs** werden mithilfe eines few-shot gestützten Cluster Algorithmus erstellt. Ein Experte erstellt hierzu **GTs** 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.

Die entstehenden Merkmalsvektoren werden durch eine Standardisierung der Werte, zu einem Mittelwert von Null und Varianz von Eins überführt. Daraufhin wird eine Principal Component Analysis [78] angewandt, um die Dimensionalität der Merkmalsvektoren auf zehn zu reduzieren und 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 [85], mit einer Radial Basis Funktion [86] als Kernelfunktion, werden die Experten-GTs über die Struktur der Daten auf alle Stichproben ausgebreitet. Durch den Algorithmus entstehen GTs für die Daten ohne Experten-Labels. Diese neuen GTs 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 vortrainiert 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 Trainingsdatensatz beschränkt.

Fully-supervised NOTIZ: Hier soll stehen:

- Welche Gewichte werden eingefroren?
- Wie wird dann Transfertraining durchgeführt?

3.3.4 Decoder

An die Encoder werden jeweils zwei verschiedene Decoder angehängt. Abb. 3.4 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 niederdimensionale 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 multi-head 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 GTs. Anschließend werden die besten Methoden ausgewählt und auf den finalen Datensatz trainiert.

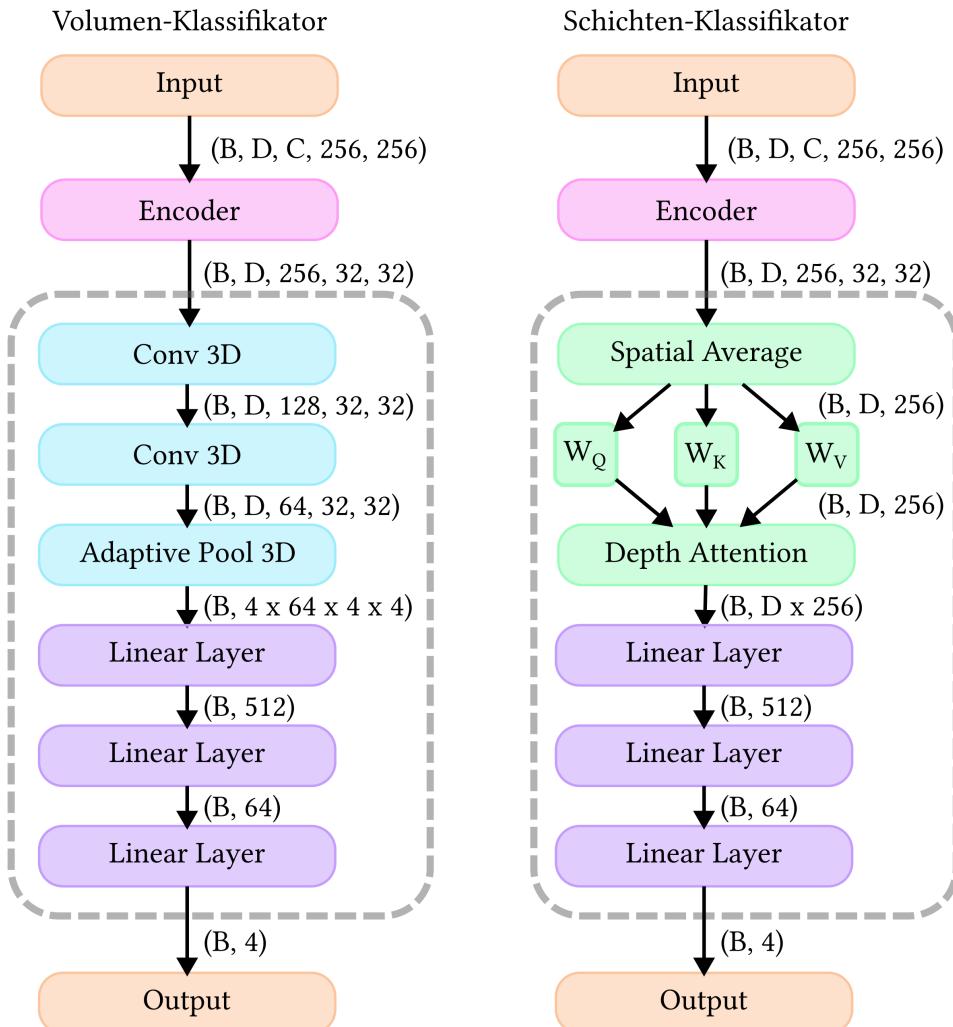


Abb. 3.4 | Architektur der beiden Klassifikatoren. Der Encoder wird bei beiden modular ausgetauscht. Links zu sehen ist die Architektur des *Volumen-Klassifikator*, der die Merkmale, die der Encoder generiert, als Volumen interpretiert und mithilfe von 3D Faltungen und Pooling daraus eine Repräsentation 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.

Implementation 4

4.1 Überblick

Im nachfolgenden Kapitel wird die Implementierung aller relevanten Methoden erklärt. Hierbei steht die praktische Anwendung dieser Methoden in Form von Nutzerschnittstellen oder als Entwicklerskript auf ausgewiesener Hardware im Vordergrund.

4.2 Segmentierungsnetze

- Wie sind die Netze je eingebunden?
- Welche Hardware?
- Welche Vor-/Nachverarbeitung?

4.3 Labeling App

Um den Klassifikator zu trainieren werden [GT](#) zu einigen Zieldaten mithilfe der Labeling App erstellt. Als Frontend dieser App dient eine dash Anwendung die eine einfache html Graphical User Interface ([GUI](#)) bereitstellt. Das Backend ist mit Python erstellt und die gesammelten Daten werden als JSON gespeichert. Hierzu ist ein Docker mit Zugriff auf das lokale Dateiensystem versehen und über Kubernetes betrieben. Abb. 4.1 zeigt die [GUI](#) der Labeling App. Zentral zu sehen sind zwei Fenster, die einen 2D Schnitt des ausgewählten Nucleus anzeigen. Darauf ist jeweils ein Kasten gezeichnet, der die 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*: Vorherige Zelle auswählen.
- *Next nucleus*: Nächste Zelle auswählen.

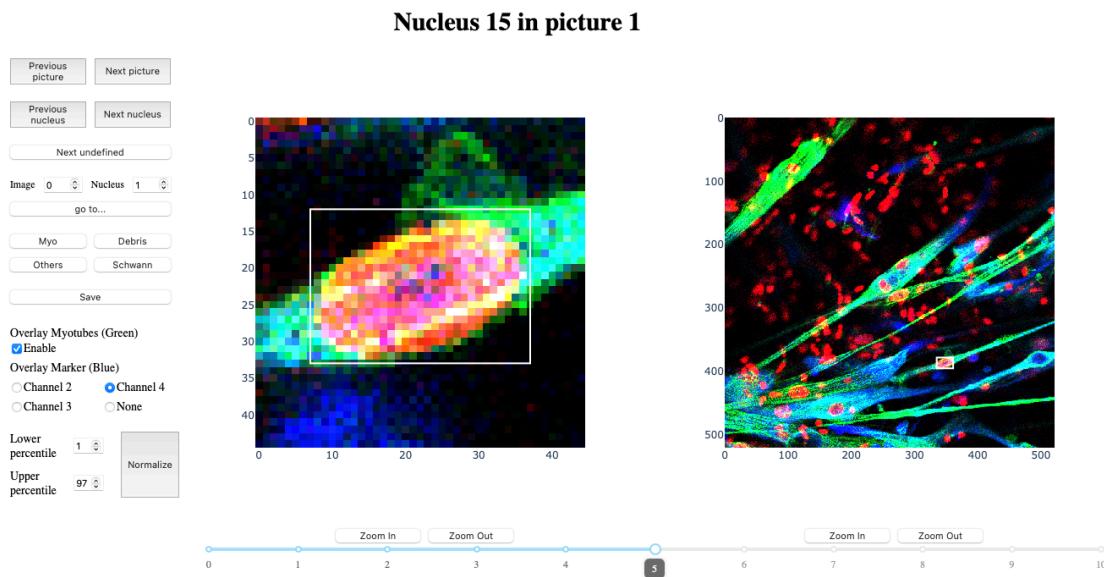


Abb. 4.1 | Die Grafik zeigt die GUI der Labeling App. In der Mitte wird die Zelle angezeigt, die gelabelt werden soll und links sind Bedienelemente zu sehen. Unter der dargestellten Zelle befindet sich ein Schieberegler, der die Navigation entlang der Z-Achse ermöglicht. Über die Bedienelemente kann 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.

- *Next undefined*: Nächste Zelle ohne eingetragene GT auswählen.
- *Image*:
- *Nucleus*:
- *go to...:*
- *Myo*: "Myotuben-Zellkern" Klasse als GT der ausgewählten Zelle definieren.
- *Debris*: "Überreste" Klasse als GT der ausgewählten Zelle definieren.
- *Other*: "Änderer" Klasse als GT der ausgewählten Zelle definieren.
- *Schwann*: "Schwannzellen-Zellkern" Klasse als GT der ausgewählten Zelle definieren.
- *Save*: Speichert manuell die festgelegten GTs ab. 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 der aktuell ausgewählten Zelle anwenden. Intensitätswerte die im beziehungsweise über dem Perzentil der einge-tragenen Schwellwerten werden hierbei zusammengefasst.
- *Zoom In*: Verkleinert den anzuzeigenden Ausschnitt.
- *Zoom Out*: Vergrößert den anzuzeigenden Ausschnitt.

Für das Backend sind einige Python Skripte direkt im Docker mounted. Da die wichtigste Anforderung der Anwendung die nutzerfreundliche Bedienung ist, 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 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.

4.4 Training

Für das Training der Klassifikatoren wurde eine NVIDIA GeForce RTX 3090 Ti mit 24GB VRAM verwendet. **CPU und RAM fehlt noch**. Die vortrainierten Modelle werden vom Open-Source Framework Pytorch bereitgestellt, genauso wie die Methoden zum Erstellen der Decoder-Köpfe, der Optimizer und die Backpropagation Funktionalitäten. Der Trainingsablauf ist modular aufgebaut, um die Kombinationen der Methoden aus Kapitel 3 nahtlos zu implementieren.

Results 5

5.1 Überblick

5.2 Segmentierung

Für die Wahl eines Segmentierungsnetzes wird in Kapitel 3 das Bewertungskriterien IPQ eingeführt. Außerdem wird in Kapitel 3 der annotierte S_BIAD1518 Datensatz vorgestellt. Die IPQ wird auf dem Datensatz mit den Masken von drei vortrainierten Segmentierungsnetzen und, zur Validierung, mit der GT ausgeführt. Die Masken unterscheiden sich optisch stark (siehe Abb. 5.1), was sich auch in starken Unterschieden in der IPQ äußert.

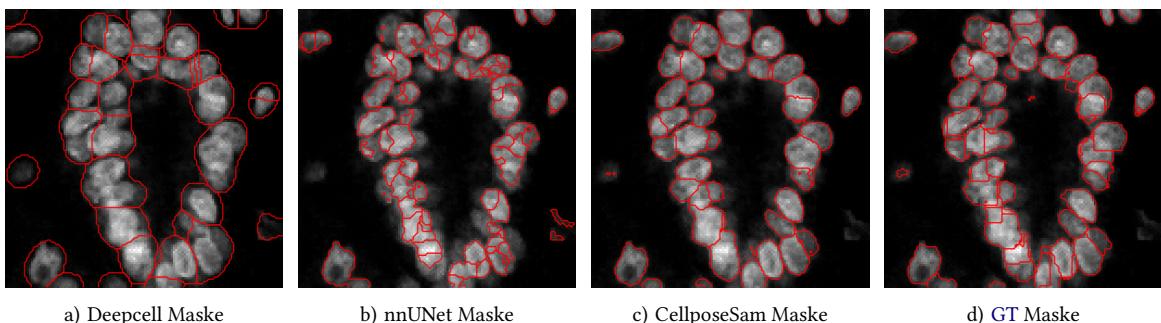


Abb. 5.1 | Darstellung der Segmentierungsmasken der verschiedenen Segmentierungsnetze als Konturen auf einem zweidimensionalen Durchschnitt einer Stichprobe des S_BIAD1518 Datensatz.

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 höchsten Mittelwert erreicht. Die Recognition Qualität (RQ) der nnUNet-Masken ist signifikant höher als die der CellposeSAM-Masken (p-Wert: $3,4764 \cdot 10^{-6}$), und ebenso die Injektive Qualität (IQ) der Deepcell-Masken (p-Wert: $6,7763 \cdot 10^{-8}$). 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 Da-

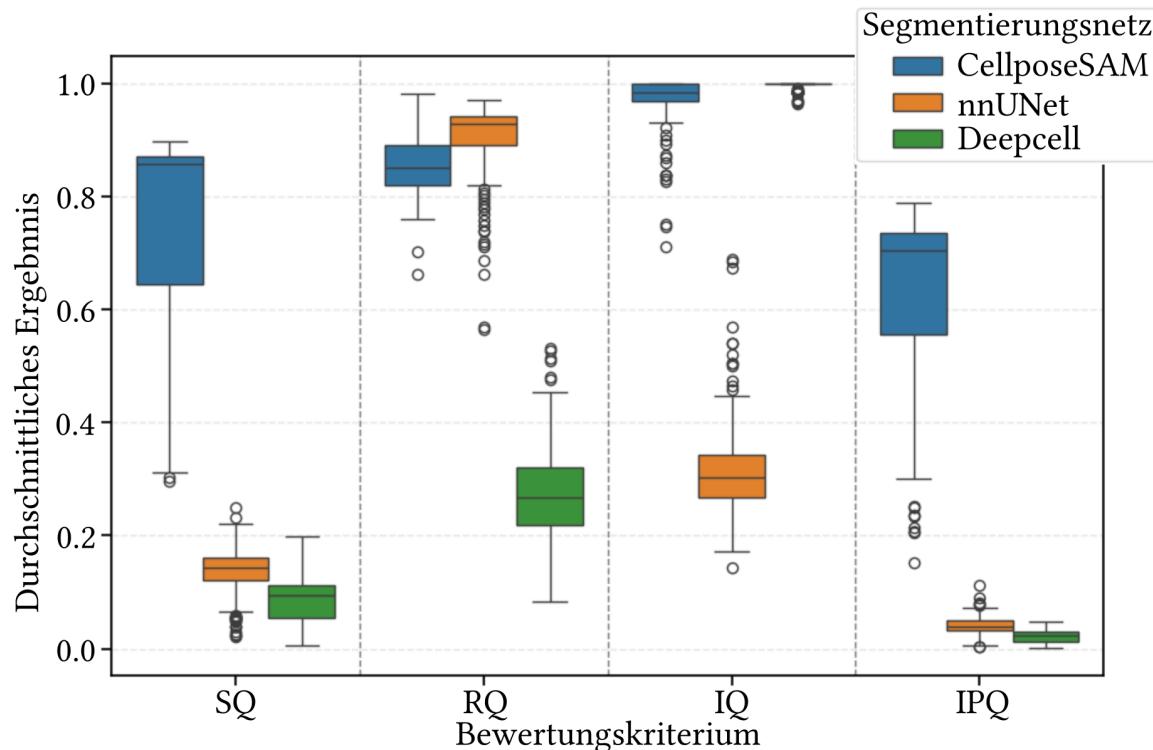


Abb. 5.2 | Boxplots der Ergebnisse der IPQ Berechnungen mit Faktoren k_1 , k_2 und k_3 jeweils gleich eins. Die X-Achse unterteilt die Daten in die Kriterien Segmentierungs-Qualität (SQ), Recognition Qualität (RQ), Injektivitäts-Qualität (IQ) und Injektive Panoptische Qualität (IPQ), wie in der Formel 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.

ten, was den Unterschieden der Bildkategorien, die der Datensatz enthält, geschuldet sein könnte.

5.3 Klassifikation

Die in Kapitel 3 vorgestellten Methoden zur Klassifikation werden anhand eines separaten Anteil des manuell gelabelten Datensatz getestet. Die Genauigkeit, also der Prozentuale Anteil richtiger Vorhersagen, wird als Kriterium verwendet.

Diskussion 6

6.1 Überblick

6.2 Segmentierung

Durch einen Vergleich der Masken mit der **GT** in Abb. 5.2 und ihren zugehörigen Ergebnisse der einzelnen Bewertungskriterien sind die Schwächen und Stärken der individuellen Netze ersichtlich. Die nnUNe-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.

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