

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. (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 τ_s und τ_t , zielen Domänenadaptations-basierte Verfahren darauf ab, ein Modell mit Parametern θ zu

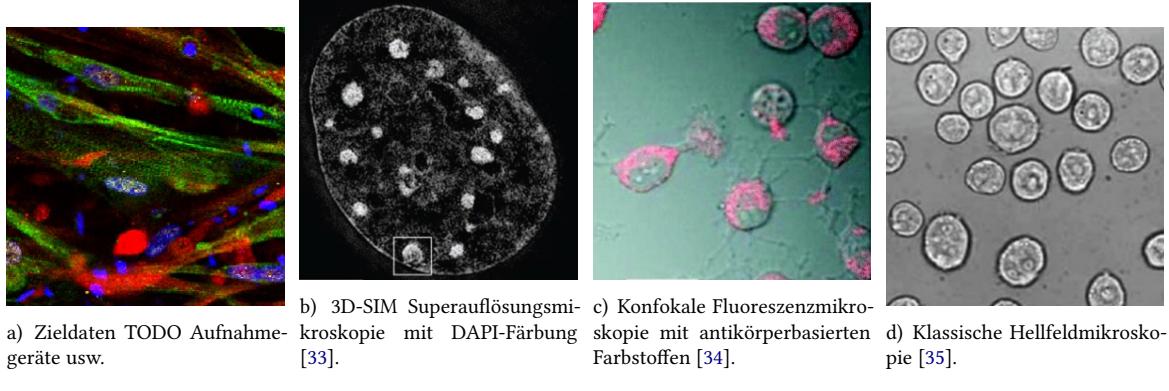


Abb. 2.1 | Vier Aufnahmen von Zellen. Die Bilddomänen sind durch die verschiedenen Marker und Aufnahmetechniken klar zu unterscheiden [33, 34, 35].

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-

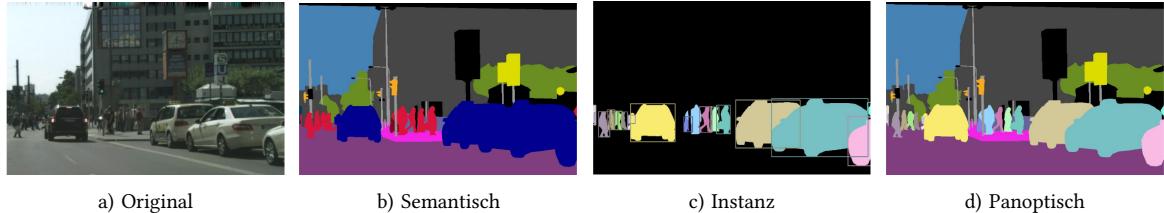


Abb. 2.2 | Die verschiedenen Arten der Segmentierung. Links ist das Originalbild zu sehen, rechts folgend sind die zugeordneten pixelweisen Annotationen farblich eingezeichnet. Gleiche Farben bedeuten gleiche Annotationen. In der semantischen Maske sind gleiche Annotationen mehrerer Objekte von semantisch 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, γ 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 θ 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 θ 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.

Um das Verhalten von Klassifikatoren zu visualisieren gibt es verschiedene Methoden. Eine dieser Methoden ist Grad-CAM [96]. Grad-CAM steht für Gradient-weighted Class Activation Mapping und ist eine Methode um räumlich aufgelöste Relevanzkarten aus tiefen neuronalen Netzen zu erzeugen. Dabei werden die Gradienten einer bestimmten Klasse in Bezug auf die Aktivierungen einer Schicht des Modells verwendet, um Regionen des Eingabebildes zu finden, die besonders stark zur Vorhersage beitragen. Die resultierende "WichtigkeitsKarte wird auf die Eingabe zurückprojiziert und typischerweise als farbige Heatmap dargestellt.

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 [97], 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 [98], 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 [99, 100], 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 [101], eine Sammlung aus 13 Datensätzen verschiedener Mikroskopiemodalitäten, die sich zum Messen der Segmentierungs- und Verfolgungsfähigkeiten für verschiedene Zelltypen eignen.
- MoNuSeg [102], eine Zusammenstellung manuell annotierter Gewebeschnitte von sieben verschiedenen Organen. Über 21.000 Zellen sind pro Bild in den 30 Bildern mit verschiedenen Färbungen und Aufnahmetechniken verteilt.
- TissueNet [103] ein umfassender Datensatz mit über 1.000.000 Zellen mit diversen Gewebearten und unterschiedlichen Aufnahmetechniken.
- S-BIAD1518 [104, 105], ein Datensatz der neben manuell annotierten Bildern von acht verschiedenen Zellarten sind synthetisch erzeugte Daten enthält. Mit Hilfe von SpCycleGAN [106] 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 [107, 108, 109, 110, 111]. Beispielsweise können 3D-Trainingsdaten mit realistischer

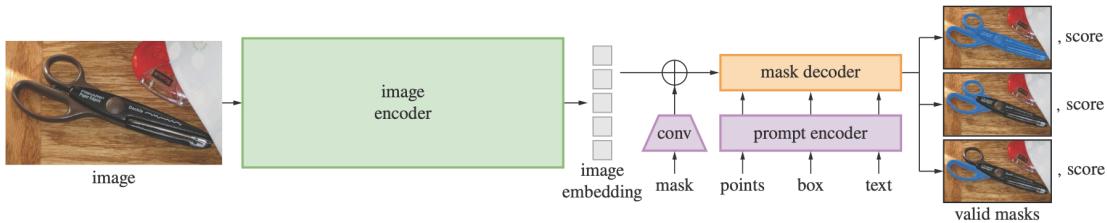


Abb. 2.3 | [120]. Architektur des **SAM**. Eingabebilder werden durch einen Bildencoder zu Repräsentationen umgeformt. Zusätzliche optionale Hinweise auf das zu segmentierende Objekt werden durch Bildfaltungen oder einen Prompt Encoder repräsentiert. Anschließend prädiziert der Decoder mehrere mögliche Masken und zugehörige Zuversichtlichkeiten.

Zellform und -ausrichtung und umgebenden Markern synthetisch erzeugt und durch ein Generative Adversarial Network an eine gewünschte Bilddomäne angepasst werden [112].

2.3.2 Segmentierungsmodelle

Foundation-models sind für viele moderne Künstliche Intelligenz (KI)-Anwendungen unerlässlich [113]. Sie werden zunächst für allgemeine Aufgaben vortrainiert und anschließend auf spezifische Anwendungen angepasst (fine-tuning), meist unter Einfrieren von Teilen der Gewichte [114]. Auch Segmentierungsmodelle profitieren stark von umfangreichem Vortraining [115]. In der aktuellen Forschung werden verschiedene foundation models für Segmentierung angewandt [116, 117, 118, 119]. Ein prominentes Exemplar ist das Segment Anything Model (**SAM**) von Meta AI [120] (siehe Abb. 2.3). Es besteht aus einem Bild-Encoder, einem Prompt-Encoder und einem Masken-Decoder. Als Bild-Encoder dient ein Vision Transformer [121], mit Vortraining als Masked Auto Encoder [122] 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 [123] 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 [124, 125, 126]. Auch für bestehende biologische Segmentierungsanwendungen, wie etwa Cellpose[127], wurde **SAM** auf Zelldaten angepasst [128]. 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.

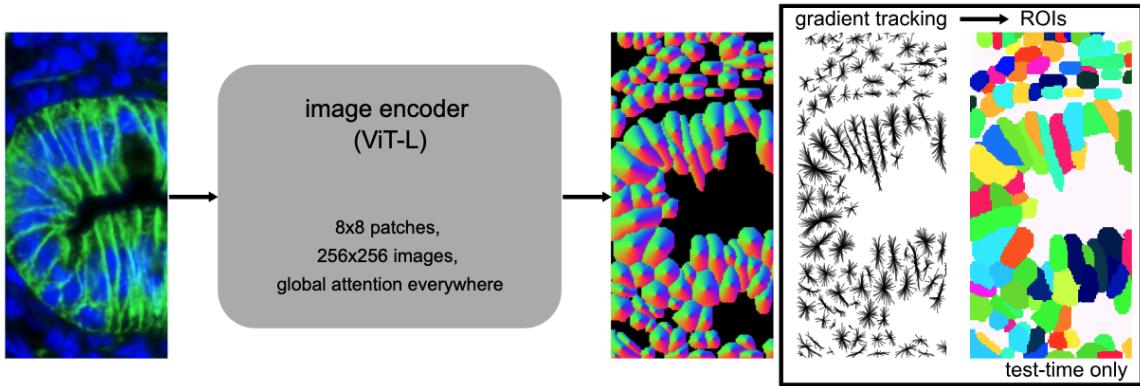


Abb. 2.4 | Ablauf des CellposeSAM-Modells. Eingabebilder werden durch einen Bild-Encoder (ViT-L) direkt zu sogenannten *Flows* umgeformt, einer Repräsentation von vorhergesagten Objektmerkmalen, deren Wert von der relativen Position innerhalb des detektierten Objekts abhängt. Die Gradienten der Flows werden verfolgt (gradient tracking) und aus dem daraus entstehenden Gradientenfeld werden Segmentierungsinstanzen (ROIs) vorhergesagt [128].

2.4 zeigt diesen Ablauf als Diagramm.

Deepcell [129, 130, 131] bietet weitere Zellsegmentierungsmodelle. Das Deepcell-Caliban-Modell [132] nutzt als Encoder eine EfficientNetV2L-Architektur [133], 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.

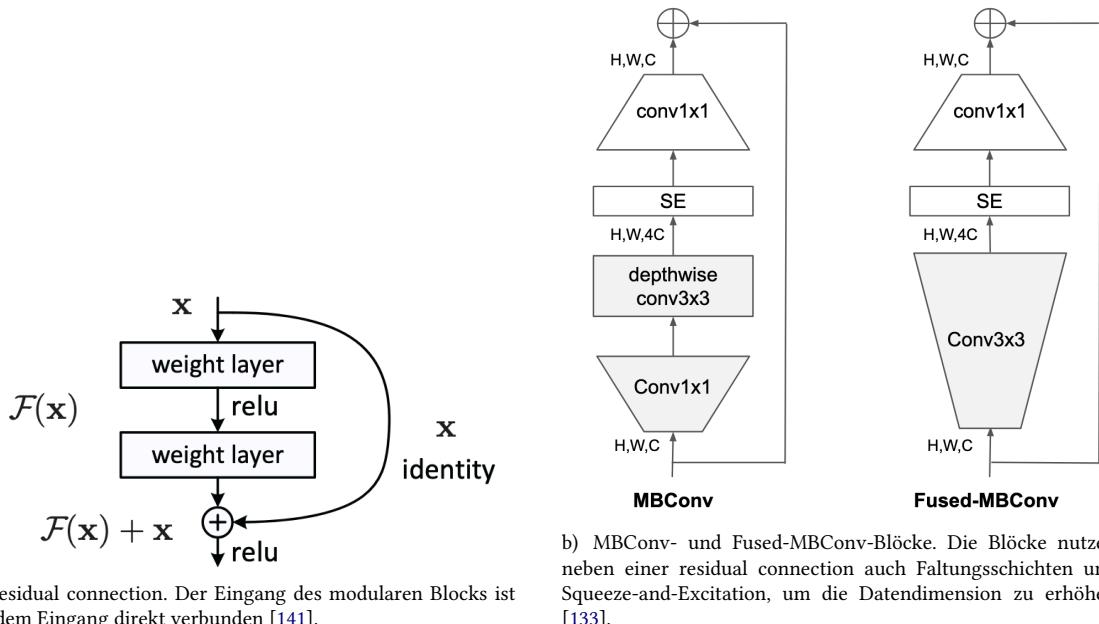
In der Literatur sehr verbreitet ist außerdem das nnU-Net [134], ein Segmentierungsframework, das sich automatisch an neue biomedizinische Aufgaben anpasst. Es konfiguriert Vorverarbeitung, Netzwerkarchitektur, Training und Nachbearbeitung dynamisch auf Basis der Eigenschaften des jeweiligen Datensatzes. 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, 114, 135]. State-of-the-Art für Bild-Encoder sind CNNs oder ViTs, die auf dem ImageNet [136] Datensatz vorgenommen werden [137, 138]. Klassifikatoren profitieren stark von ImageNet-Vortraining [139, 140]. **ResNet** ist ein Residual Neural Network, ein CNN mit sogenannten residual connections [141]. Diese residual connections verbinden den Ein- und Ausgang modularer Faltungsschichten und verbessern die Leistungsfähigkeit tiefer neuronaler Netze [142, 141, 143] (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 [144] und der ReLU [145] Aktivierungsfunktion.

EfficientNetV2 ist ein Nachfolger der EfficientNet-Modellfamilie [146, 133]. Die Architektur basiert auf modularen Blöcken von Bildfaltungsoperatoren mit besonders kleinen Faltungskernen und Squeeze-and-Excitations, genannt MBConv [147, 146] und Fused-MBConv [148] (siehe Abb. 2.5b).

ConvNeXt [149] ist eine CNN-Modellfamilie mit besonders großen Faltungskernen. Die



a) Residual connection. Der Eingang des modularen Blocks ist mit dem Eingang direkt verbunden [141].

b) MBConv- und Fused-MBConv-Blöcke. Die Blöcke nutzen neben einer residual connection auch Faltungsschichten und Squeeze-and-Excitation, um die Datendimension zu erhöhen [133].

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

ConvNeXt-Architektur umfasst fünf modulare Blöcke mit Faltungen und residual connections, wie die ResNet-Architektur [141]. Allerdings verändert dabei **ConvNeXt** einige Details der ResNet Architektur, wie beispielsweise die GeLU Aktivierungsfunktion [150] und Layer normalization [93].

Swin Transformer [23] ist eine beliebte Modellfamilie von ViTs. Ihr Nachfolger, die **Swin Transformer V2** [151], vergrößert die Modelle noch. Die Architektur kombiniert Bildausschnitte mit einem Positions-Bias. Hierzu werden 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 [152] 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. A.1 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 vortrainierter 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 Hyperparameteroptimierung durchgeführt werden. Des Weiteren ist der Umgang mit dreidimensionalen Daten, besonders in Umgebungen ohne große Rechenleistung ein offenes Problem. Diverse 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 Voraussagen 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.

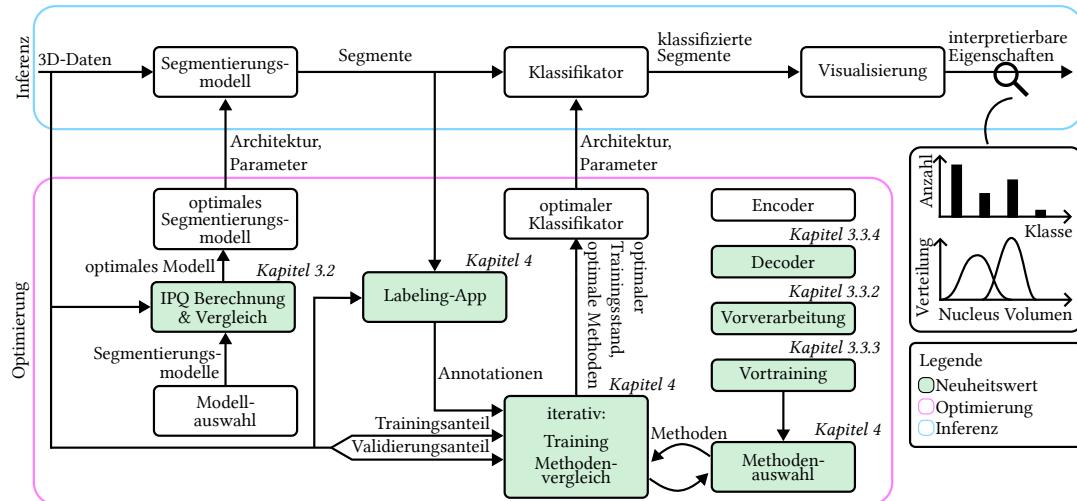


Abb. 3.1 | Aufbau der Anwendung der vorliegenden Arbeit. Die Anwendung kann regulär angewandt (Inferenz) oder optimiert werden (Optimierung). Die Optimierung ist zweiteilig. Zur Optimierung der Segmentierungsmodelle wird das neu entwickelte Injektive Panoptische Qualität (IPQ) Bewertungskriterium eingesetzt. 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 [104, 105] 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

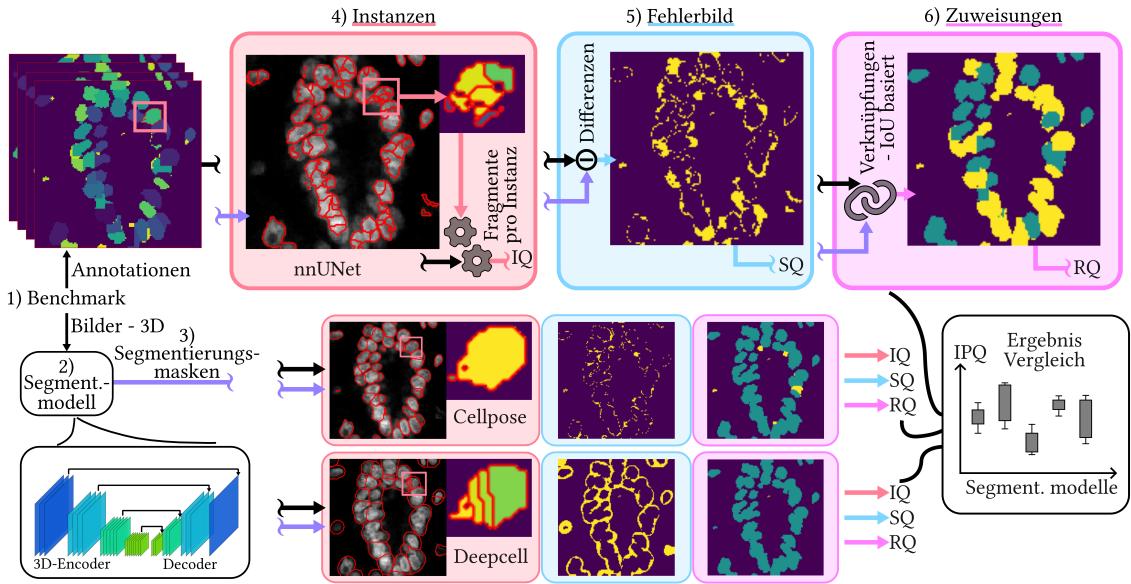


Abb. 3.2 | IPQ Visualisierung - Das obere Ablaufdiagramm stellt den Prozess dar, durch den das Segmentierungsnetz gewählt wird, das für die Anwendung der vorliegenden Arbeit eingesetzt wird. Ein peer-reviewed Benchmark-Datensatz aus dreidimensionalen Bildern (1) von diversen Zellkulturen mit dazugehörigen 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_{opt}

Für id_{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_{opt} , IoU_{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

3.3.1 Übersicht

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

3.3.2 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 Klassifikatorapplikationen, die auf dem ImageNet-Datensatz [136] 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 [136] 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 |

3.3.3 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, hier Masken-Methode genannt, 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 hier Distanz-Methode genannt. Mit der Distanz-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 Segmen-

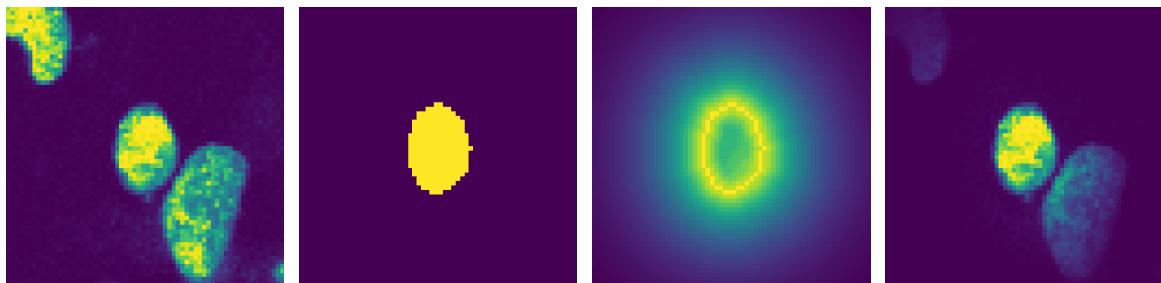
tierungsmaske wie folgt multipliziert:

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

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 Distanz-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 Masken-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 Distanz-Methode.
d) Darstellung der Distanz Methode, 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). Um einen Einfluss auf die Ergebnisse durch ungleiche Verteilung der Annotationen

auf die vorhandenen Klassen zu vermeiden, extrahiert der Retreiver außerdem vor dem Training, bei der Initialisierung, die Anzahl der Stichproben pro Klasse. Mithilfe dieser Verteilung der Anzahlen werden dann die Gradienten stärker gewichtet, die zu unterrepräsentierten Klassen gehören. 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 'Retreiver'. Dieser Retreiver versieht die aktuell gewünschten Bildausschnitte mit der Vorverarbeitungsmethode und verschiebt dann ausschließlich diese Daten auf die GPU.

3.3.4 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.

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.

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

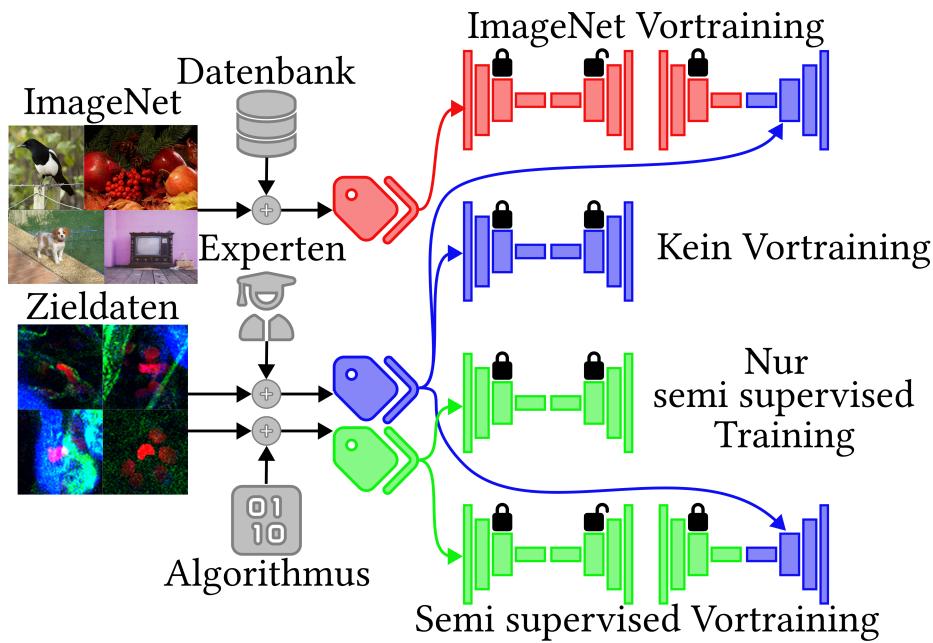


Abb. 3.4 | Übersicht über die Vortrainingsmethoden. Links zu sehen sind die beiden verfügbaren Bildermengen, ImageNet und die Zieldaten. Rechts von diesen Bildermengen werden diesen Bildern Annotationen hinzugefügt, entweder durch die ImageNet-Datenbank, Experten oder einen Algorithmus. Jeder Encoder (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 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 fine-tuned.

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 vorgenutzt 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 Vortaining 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.

3.3.5 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 *Schichten-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 [134].
- 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: $maske_{refined}$

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

$marker \leftarrow$ LokaleMaxima($distanz$)

$gradient \leftarrow -distance$

$maske_{refined} \leftarrow$ 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 $maske_{refined}$

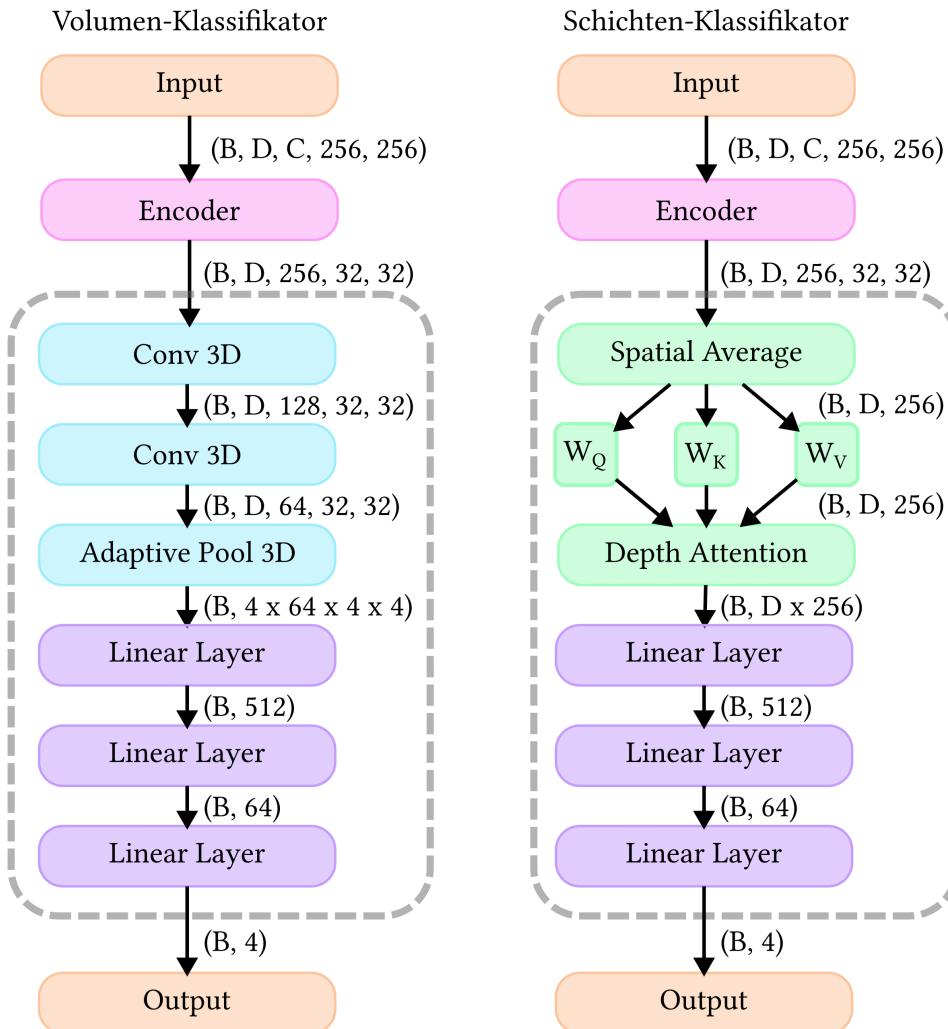


Abb. 3.5 | Architektur der beiden Klassifikatoren. Der Encoder wird bei beiden modular ausgetauscht. Links zu sehen ist die Architektur des *Volumen-Klassifikators*, der die Merkmale, die der Encoder generiert, als Volumen interpretiert und mithilfe von 3D-Faltungen und Pooling daraus eine Repräsentation 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 [132]. 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-

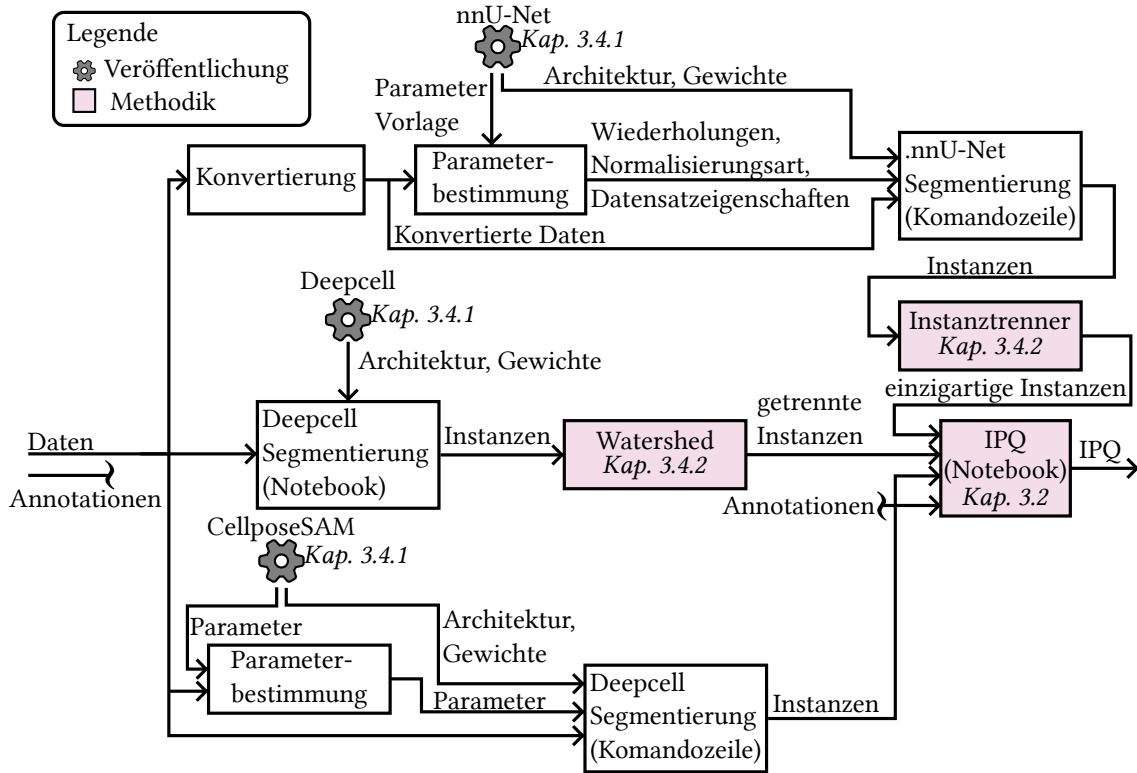


Abb. 4.1 | Signalflussdiagramm der praktischen Umsetzung der Segmentierungsmodelle.

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. 3.3.5).

In das zweite Modul wird auch eine Methodenauswahl eingegeben und abhängig von der Vortrainings- und Vorverarbeitungsmethode wird ein entsprechender Datensatz zurückgegeben (siehe Kap. 3.3.4 und Kap. 3.3.3). 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. 3.3.3) 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

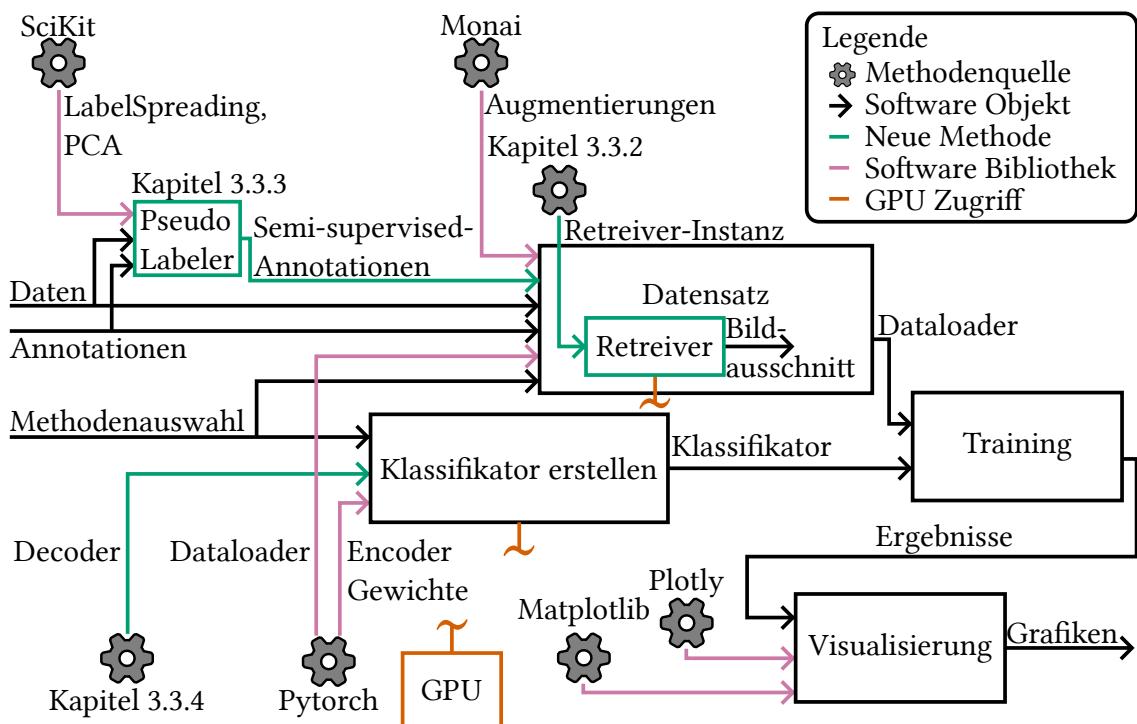


Abb. 4.2 | Signalflossdiagramm der praktischen Umsetzung der Klassifikatoren. Zu sehen sind die Module, aus denen sich die Anwendung der Klassifikatoren zusammensetzt. Links sind als Eingabe Daten mit zugehörigen Annotationen und eine Kombination von Methoden und Rechts Grafiken, die die Ergebnisse visualisieren 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. 3.3.4).

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. 3.3.3). 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 Segmentationsmasken eingelesen. In einer Schleife werden mit Hilfe eines Klassifikator-Helfer 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. 3.3.3). Ein Pseudo-Labler-Modul erstellt Semi-supervised-Annotationsen für den Datensatz, die je nach Vortrainingsmethode in der Schleife genutzt werden können (siehe Kap. 3.3.4). 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

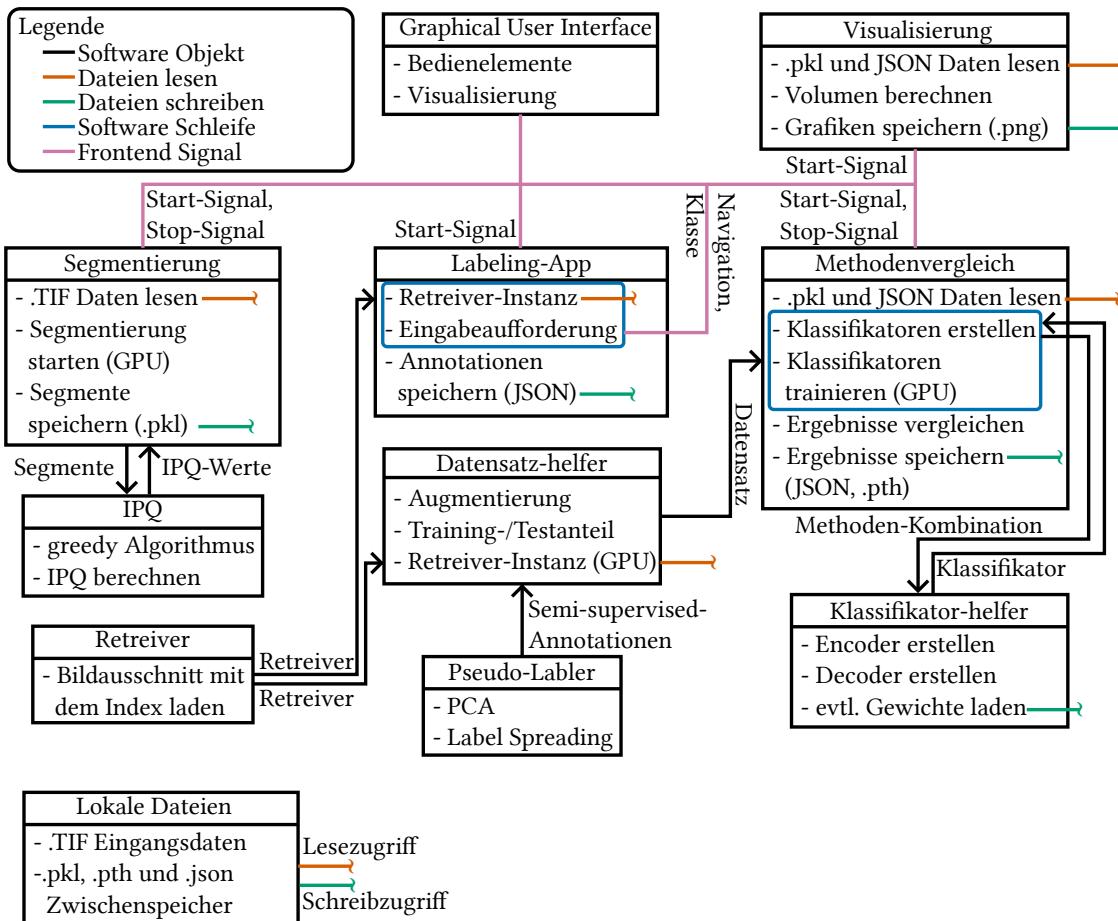
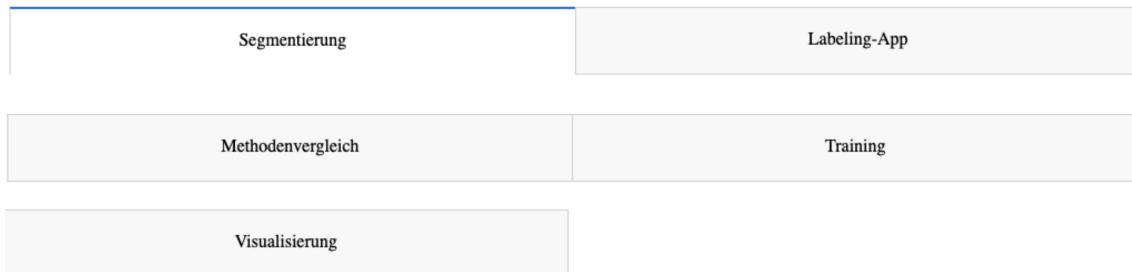


Abb. 4.3 | Übersicht der Architektur der neu entwickelten 3D-Zelldaten-Pipeline. Als Kästen sehen sind die Module die die 3D-Zelldaten-Pipeline bilden. Schwarze Pfeile stellen die Weitergabe von Software Objekten zwischen den Modulen dar. In Orange und Grün sind Zugriffe auf lokale Dateien dargestellt, 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.

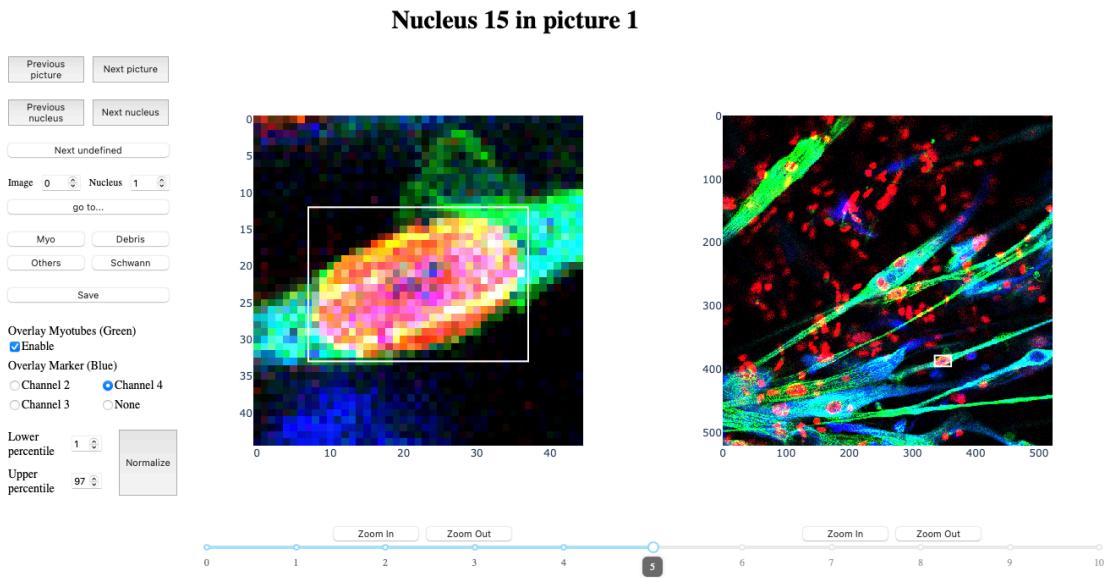


Abb. 4.5 | Die Abbildung zeigt die GUI der neu entwickelten Labeling-App. In der Mitte wird die Zelle angezeigt, die gelabelt werden soll, und links sind Bedienelemente zu sehen. Unter der dargestellten Zelle befindet sich ein Schieberegler, der die Navigation entlang der Z-Achse ermöglicht. Über die 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. 3.3.4). 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) | |
| | |
| Encoder | |
| <input type="checkbox"/> CellposeSAM | |
| <input type="checkbox"/> ResNet18 | |
| <input type="checkbox"/> ResNet101 | |
| <input type="checkbox"/> SwinV2 | |
| <input type="checkbox"/> ConvNeXt | |
| <input type="checkbox"/> EfficientNetV2 | |
| Decoder | |
| <input type="checkbox"/> Schichten-Klassifikator | |
| | |
| <input type="checkbox"/> Volumen-Klassifikator | |
| Vorverarbeitung (Nucleus Kanal) | |
| <input type="checkbox"/> Distanztransformation | |
| <input type="checkbox"/> Segmentierungsmaske | |
| Vortraining | |
| <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 |
| <input type="button" value="Starte Training"/> | |

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 werden in mehreren Durchläufen der 3D-Zelldaten-Pipeline durchgeführt. Auf Grundlage der Messergebnisse der 3D-Zelldaten-Pipeline wird eine statistische Auswertung vorgenommen. Zuerst werden die Bewertungen der Instanzsegmentierungsmasken anhand der IPQ-Metrik und anschließend die der Klassifikatoren anhand ihrer Genauigkeit betrachtet. Die IPQ-Ergebnisse werden hier auch in die drei Faktoren: Recognition-Qualität (RQ), Segmentierungs-Qualität (SQ) und die neu eingeführte Injektive-Qualität gegliedert (siehe Kap. 3.2). Daraus wird ersichtlich, dass jedes der Segmentierungsmodelle in einem der Faktoren 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 wird 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_BIAD1518-Datensatz vorgestellt. Die IPQ wird auf dem Datensatz mit den Masken der drei vortrainierten Segmentierungsmodelle und zur Validierung mit den Annotationen durchgeführt. Die Masken unterscheiden sich optisch deutlich (siehe Abb. 5.1). Masken des nnUNet-Modells sehen kantiger aus als die Masken anderer Modelle und weisen oft eine raue Kontur mit hervorstehenden Extremitäten oder Einkerbungen auf. Deepcell-Masken sehen glatt und ausgebreitet aus. Durch die Watershed-Nachverarbeitung sind teilweise sichtbare Artefakte entstanden, an Stellen, an denen das Trennen der Instanzen für Watershed nicht möglich ist. Die CellposeSAM-Masken sehen intuitiv am besten aus, sie haben oft nahezu elliptische Konturen und trennen Instanzen so wie es für das menschliche Auge sinnvoll erscheint. Das spiegelt sich auch in deutlichen Unterschieden in der IPQ wider.

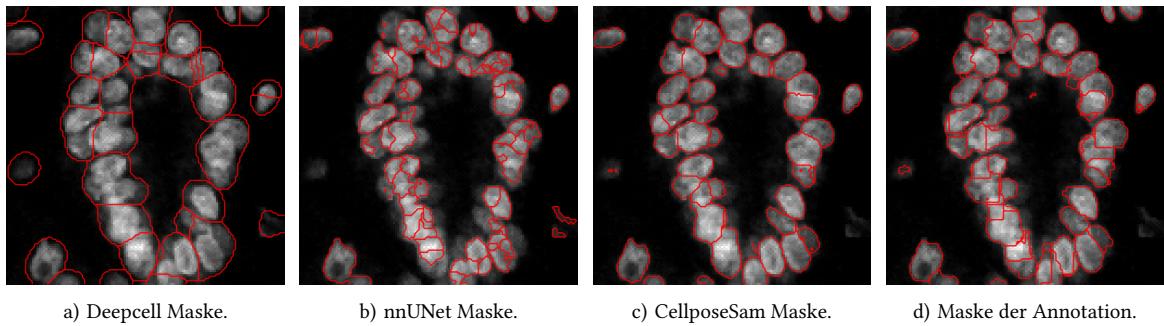


Abb. 5.1 | Darstellung der Segmentierungsmasken der verschiedenen Segmentierungsmodelle sowie der Annotation als Konturen auf einem zweidimensionalen Durchschnitt einer dreidimensionalen Stichprobe des S_BIAD1518-Datensatzes.

Abb. 5.1 zeigt exemplarisch einen 2D-Schnitt eines 3D-Bildes mit roten Linien als Konturen der Annotation oder der vorhergesagten Segmentierungsmaske je Modell. Alle Annotations erreichen einen **IPQ**-Wert von genau Eins. Die Ergebnisse jedes Segmentierungsnetzes sind einzeln und für jedes Bild im Appendix A.2 angehängt. Eine Zusammenfassung der Ergebnisse ist in den Boxplots in Abb. 5.2 gegeben. Das zentrale Ergebnis ist, dass CellposeSAM die besten **IPQ**-Werte liefert. Mit einem Mittelwert von 0,64 ist die **IPQ** von CellposeSAM höchst signifikant höher als der Mittelwert bei nnUNet (0,04) und Deepcell (0,02), bestätigt durch einseitige T-Tests. Dennoch zeigt sich, dass CellposeSAM lediglich in der Kategorie Segmentierungs-Qualität (SQ) den höchsten Mittelwert aufweist. Die Recognition-Qualität (RQ) der nnUNet-Masken ist höchst signifikant höher als die der CellposeSAM-Masken. Ebenso ist die Injektive Qualität (IQ) der Deepcell-Masken höchst signifikant höher als die IQ der CellposeSAM-Maske. Obwohl nnUNet und Deepcell jeweils eine Metrik dominieren, wird ihr **IPQ**-Wert durch die beiden niedrigsten 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 zwischen den Bildkategorien, die der Datensatz enthält, geschuldet sein kann. In Abb. 5.3 sind Beispiele für die einzelnen Fehlerarten zu sehen.

Das IQ-Beispiel zeigt die Verletzung injektiver Abbildung der Nuclei auf die Annotation anhand einer nnUNet-Maske. Zentral ist hier ein nicht-eliptischer Nucleus zu sehen. Dieser Nucleus wird vom nnUNet-Modell in mehrere Segmente unterteilt, wodurch die räumliche Konzentration der Nuclei überschätzt wird. Nicht-eliptische Nuclei führen häufig zu dieser Art von Fehlern.

Mit dem Fehler gehen RQ-Fehler einher, weil zwei der Segmente im Vergleich zur Annotation zu klein sind, um als **TP** erkannt zu werden. In der zweiten Zeile ist anhand einer CellposeSAM-Maske ein RQ-Beispiel gegeben. Das Segmentierungsmodell hat hier einen Nucleus halluziniert. Der Nucleus unten links, direkt unter dem Linken der beiden großen Nuclei, ist in der Annotation nicht zu finden. Dieser Fehler geht nicht mit einem schlechteren SQ- oder IQ-Wert einher. Durch den Fehler wird die Anzahl der Nuclei überschätzt. Zuletzt ist ein SQ-Beispiel dargestellt. Neben und unter dem sichtbaren Nucleus oben sind vermutlich Schatten von Nuclei aus anderen Z-Ebenen zu sehen, die vom Deepcell-Modell als Nuclei erfasst wurden. Dadurch ist das Segment, das den sichtbaren Nucleus abbil-

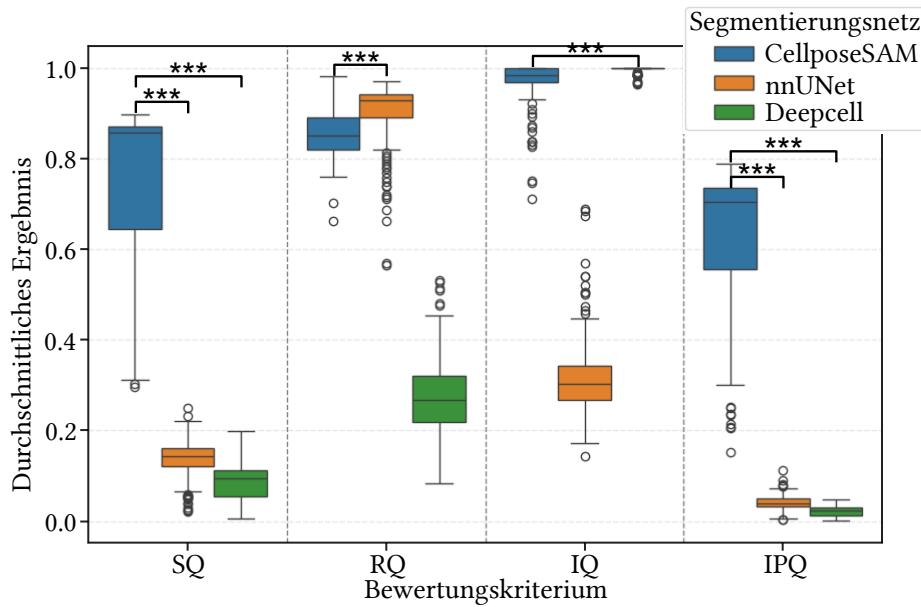


Abb. 5.2 | Boxplots der Ergebnisse der IPQ-Berechnungen mit den Faktoren $k_i = 1$ (siehe Formel (3.2)). Die X-Achse unterteilt die Daten in die Kriterien Segmentierungs-Qualität (SQ), Recognition-Qualität (RQ), Injektivitäts-Qualität (IQ) und Injektive Panoptische Qualität (IPQ), wie in der Formel (3.2) beschrieben. Für jede Metrik sind drei farbige Boxplots zu sehen, jeweils einer pro Segmentierungsmodell. Die Sterne zeigen die Signifikanz relevanter t-Tests.

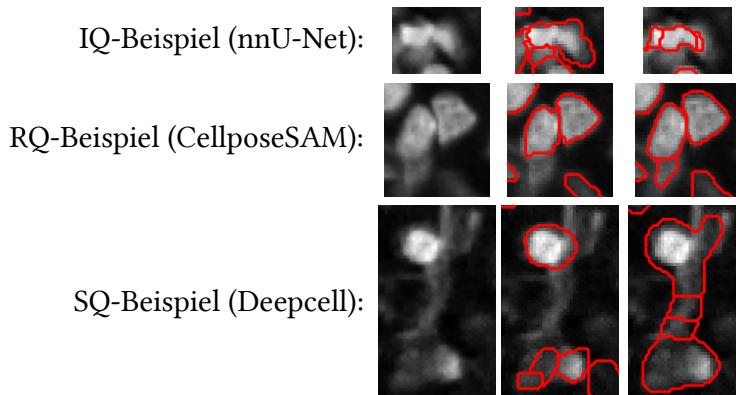


Abb. 5.3 | Exemplische Darstellung einzelner Ausprägungen verschiedener Fehlerarten. Hierzu sind jeweils, in dieser Reihenfolge von links nach rechts, ein Ausschnitt eines 2D-Durchschnitts einer Stichprobe, die Annotation als Kontur und die vorhergesagte Maske eines Modells zu sehen. Die erste Zeile zeigt ein Beispiel für einen schlechten IQ-Wert anhand einer nnUNet-Maske. In der zweiten Zeile ist ein RQ-Fehler anhand einer CellposeSAM-Maske zu sehen und unten ist ein Beispiel für einen schlechten SQ-Wert anhand einer Deepcell-Maske zu sehen.

det, deutlich zu groß, was die IoU-Werte beeinträchtigt. Mit diesem Fehler geht auch ein schlechterer RQ-Wert einher, da der Schatten zu mehreren Halluzinationen geführt hat. Da diese Halluzinationen nicht mit den Annotationssegmenten überschneiden, ist der IQ-Wert hier nicht betroffen. Der Fehler führt zu einer Fehleinschätzung des Nucleivolumens.

5.4 Klassifikation

5.4.1 Überblick

Die in Kapitel 3 eingeführten Methoden zur Klassifikation werden anhand eines separaten Anteils des manuell annotierten Datensatzes durch die neu entwickelte 3D-Zelldaten-Pipeline getestet. Der manuell annotierte Datensatz enthält: 125 Stichproben von Myotubenzellkernen, 95 Stichproben von der Klasse Debris, 148 Stichproben der Klasse SSonstiges und nur 16 Stichproben der Klasse Schwannzellen-Nucleus. Um die Genauigkeit der Klassifikatoren anhand des Test-Anteil möglichst repräsentativ zu erhalten werden manuell 50% statt 20% der Schwannzellen-Nuclei in den Test-Anteil der Daten eingesetzt. 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 aufweisen und es zu Overfitting kommen kann, wird Early-Stopping implementiert, mit der mittleren Genauigkeit des Klassifikators auf dem Test-Anteil der manuell annotierten Daten über fünf Epochen. Abb. 5.4 zeigt exemplarisch zwei Trainingsverläufe, die dieses Phänomen belegen. Im oberen Verlauf ist zu sehen, dass die Genauigkeit des Klassifikators tendenziell steigt.

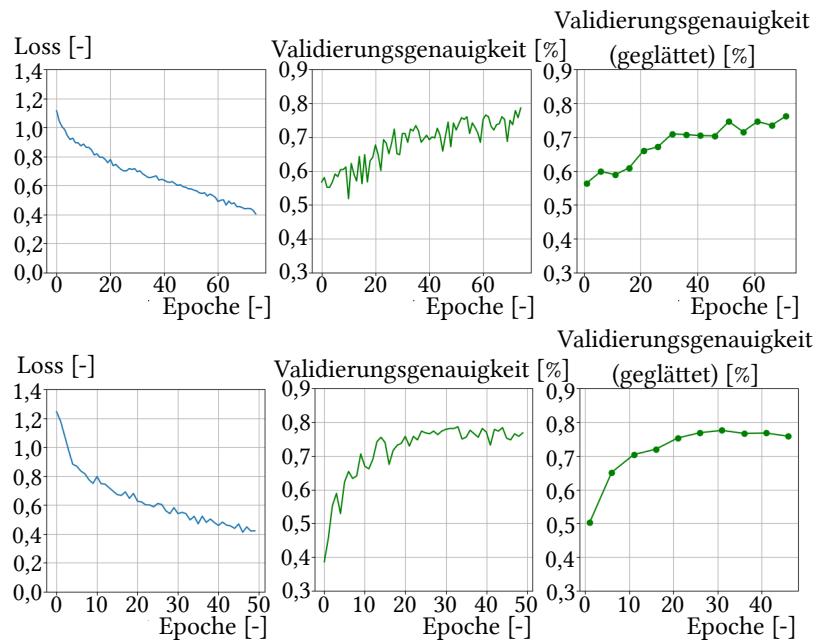


Abb. 5.4 | Die Diagramme zeigen den Trainingsverlauf eines Klassifikators. Links ist der Verlauf des Loss zu sehen. In der Mitte und rechts sind die Verläufe der Validierungsgenauigkeit zu sehen. Rechts ist die Kurve zur Anschaulichkeit durch Mittelung von je fünf Werten geglättet. Der obere Verlauf zeigt einen Klassifikator ohne Overfitting, im unteren Verlauf ist Overfitting zu erkennen.

Der Verlauf ist zwar nicht monoton steigend, aber es liegt nahe, dass bei Fortsetzung des Trainings weitere Verbesserungen zu erwarten sind. Im unteren Verlauf ist Overfitting zu erkennen. Die Genauigkeit auf dem Test-Anteil des Datensatzes sinkt etwa ab Epoche

30 und erreicht ihr Maximum in Epoche 32. Dementsprechend wird die Genauigkeit des Klassifikators in Epoche 32 für den Vergleich herangezogen.

5.4.2 Encoder

In Abb. 5.5 ist die Genauigkeit der Klassifikatoren pro Encoder gegeben. Zu sehen ist sowohl das Maximum, als auch der Durchschnitt der Genauigkeitswerte der verschiedenen Methodenkombinationen jedes Klassifikators. Die Klassifikatoren sind nach dem aufsteigenden Maximalwert sortiert. Zu sehen ist, dass der beste Wert vom ResNet18-Encoder stammt. Mit 85,9% Genauigkeit auf dem Test-Anteil des Datensatzes ist diese Kombination das gefundene Optimum. Auch im Mittelwert haben die Klassifikatoren mit dem ResNet18-Encoder mit 79,1% die höchste Genauigkeit. CellposeSAM und ResNet101 haben als Encoder die zweit- und dritthöchste durchschnittliche Genauigkeit mit 77,4% und 77,3%, jeweils. Während CellposeSAM allerdings die zweithöchste maximale Genauigkeit aufweist, hat ResNet101 die geringste. Besonders auffällig ist Swin V2. Mit 64,6% hat es die niedrigste durchschnittliche Genauigkeit, aber der Maximalwert von 83,9% ist der dritthöchste.

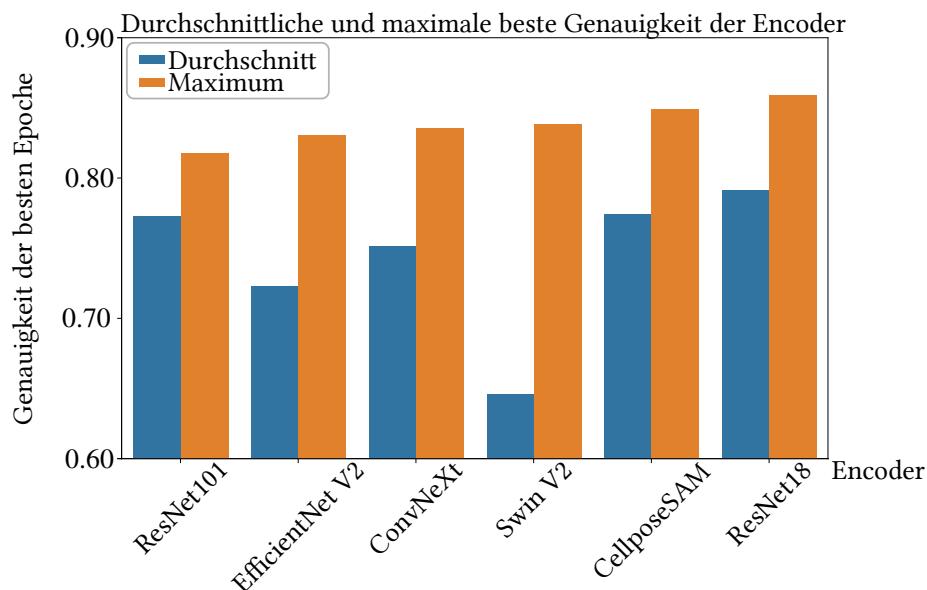


Abb. 5.5 | Das Balkendiagramm zeigt an der Y-Achse die Genauigkeit der Klassifikatoren bei Verwendung der verschiedenen Encoder. Auf der X-Achse sind die Encoder in Gruppen aufgetragen. Jede Gruppe enthält einen Maximalwert (Orange) und einen Durchschnittswert (Blau), da jeder Encoder mit verschiedenen Kombinationen von Methoden getestet wird. Die Encoder sind nach aufsteigendem Maximalwert von links nach rechts sortiert.

5.4.3 Decoder

Abb. 5.6 zeigt eine Gegenüberstellung der Genauigkeit beider Decoder in einem Balkendiagramm. Dabei werden nur die Klassifikatoren berücksichtigt, bei denen sich die Methodenkombination ausschließlich im verwendeten Decoder unterscheidet. Das ist notwendig,

da nicht alle möglichen Kombinationen trainiert wurden und ein Vergleich mit unvollständigen Kombinationen keine aussagekräftigen Ergebnisse liefert. Die blauen Balken des Balkendiagramms zeigen jeweils die Ergebnisse der Architekturen, die den Schichten-Klassifikator nutzen, die orangenen Balken die Ergebnisse des Volumen-Klassifikators. Auf der X-Achse sind als Gruppen zuerst die Encoder und zuletzt der Durchschnitt aufgetragen, und auf der Y-Achse die durchschnittliche Genauigkeit der Modelle mit dem jeweiligen Decoder. An den Daten lässt sich erkennen, 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 umgekehrt ist. Für den CellposeSAM Encoder ist der Genauigkeitsunterschied 12,6 Prozentpunkte groß, für EfficientNetV2 einen halben Prozentpunkt. Die Daten zeigen also, dass die Intensität des Unterschieds in der Genauigkeit der beiden Decoder stark vom Encoder abhängt. Die Balken rechts, die je den Durchschnitt aller Modelle mit einem Decoder darstellen, zeigen mit einem signifikanten t-Test, dass der Volumen-Klassifikator insgesamt bessere Ergebnisse liefert.

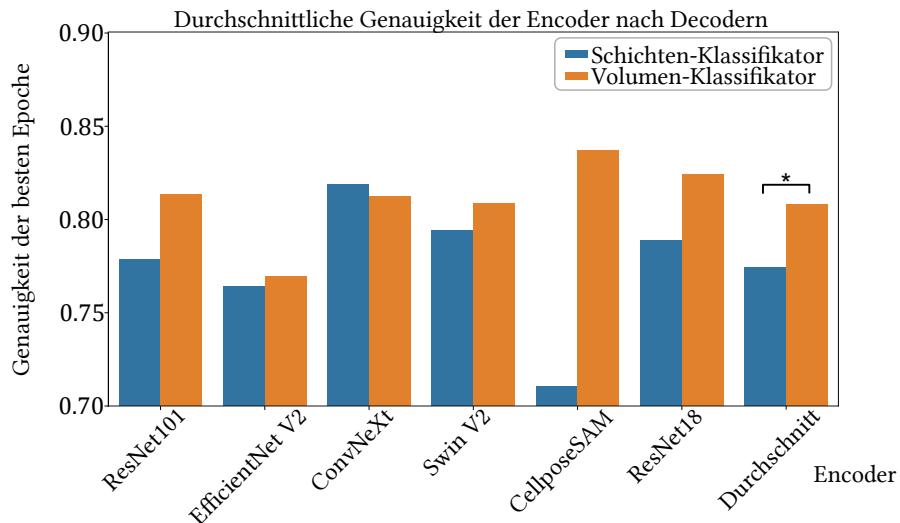


Abb. 5.6 | Das Balkendiagramm zeigt den Genauigkeitswert der Klassifikatoren unter Verwendung eines bestimmten Decoder. Auf der Y-Achse ist der Durchschnitt der Genauigkeiten der besten Trainingsepochen aller Klassifikatoren, die diesen Decoder nutzen, aufgetragen. Die X-Achse zeigt Gruppen von Encodern, 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.

5.4.4 Vorverarbeitung

In Abb. 5.7 sind die durchschnittlichen Genauigkeiten der Modelle gezeigt, die jeweils eine der Vorverarbeitungsmethoden nutzen. Wie für den Decodervergleich sind auch hier die Klassifikatoren ausgeschlossen, deren Methodenkombination mit nur einer der beiden Vorverarbeitungsmethoden trainiert wurde. Auf der Y-Achse ist die Genauigkeit aufgetragen, auf der X-Achse sind die Encoder und der Durchschnitt aller Encoder als Grup-

pen. Die Ergebnisse zeigen deutlich, dass die Masken-Methode, also das Ersetzen des Nucleus-Kanals mit der Segmentierungsmaske, zu einer höheren Genauigkeit führt. Mit der Distanz-Methode, also dem Anwenden einer Distanztransformation von der Segmentierungsmaske auf den Nucleus-Kanal, beträgt die Genauigkeit durchschnittlich 77,2%, mit der Masken-Methode sind es 81,5%. Außerdem bestätigt ein hoch signifikanter t-Test diese Aussage. 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 der Masken-Methode um 0,8 Prozentpunkte steigt gegenüber der Verwendung von der Distanz-Methode, macht die Vorverarbeitung beim EfficientNetV2 Encoders 11,5 Prozentpunkte aus.

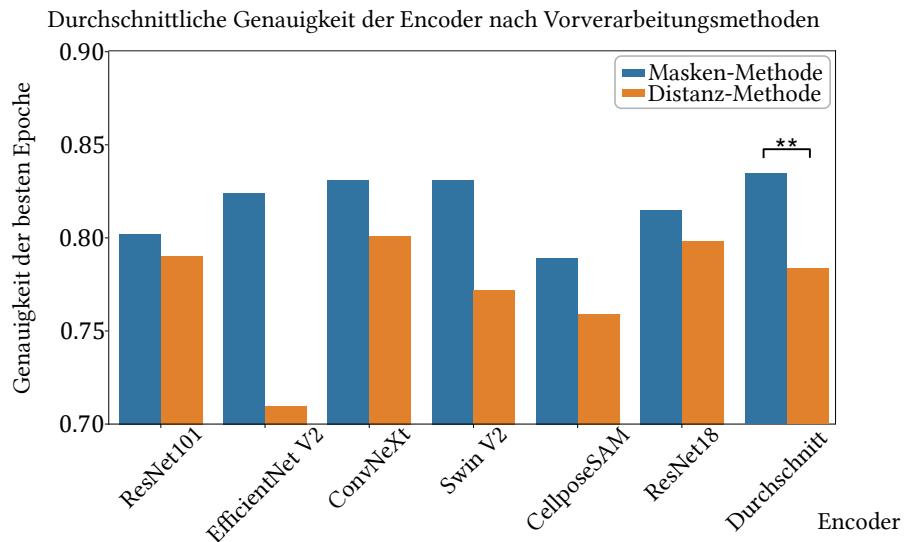


Abb. 5.7 | Das Balkendiagramm zeigt den Genauigkeitswert der Klassifikatoren bei Verwendung einer bestimmten Vorverarbeitungsart. Auf der Y-Achse ist der Durchschnitt der Genauigkeiten der besten Trainingsepochen aller Klassifikatoren, die diese Vorverarbeitungsart nutzen, aufgetragen. Die X-Achse zeigt Gruppen von Encodern, jeweils besetzt mit einem Wert für die Masken-Methode, die den Nucleus-Kanal des Bildes durch die Segmentierungsmaske ersetzt, und die Distanz-Methode, die eine Distanztransformation auf den Nucleus-Kanal anwendet. Rechts zu sehen sind Balken, die den Durchschnitt der Werte aller Encoder, abhängig von der Vorverarbeitungsmethode, zeigen.

Mithilfe von GradCAM [96] wird außerdem das Verhalten der Klassifikatoren exemplarisch visualisiert. Abb. 5.8 zeigt den Nucleus-Kanal und einen Marker-Kanal von zwei 2D-Schichten. Die beiden 2D-Schichten sind aus unterschiedlichen 3D-Stapeln, links einer der mit der Masken-Methode, und rechts einer der mit der Distanz-Methode vorverarbeitet wurde. Zu sehen ist jeweils eine Heatmap, die, für einen bestimmten Klassifikator, gradientenbasiert die Aktivierungen in einer Schicht des Merkmalsraums zurück auf die Eingangsdaten projiziert. Die Heatmaps sind jeweils zur besseren Interpretierbarkeit einer 2D-Schicht der Segmentierungsmaske des Nucleus und einem Marker-Kanal überlagert. Dadurch wird räumlich aufgezeigt, welche Regionen der Eingangsdaten zur Entscheidung für eine bestimmte Klasse besonders beigetragen haben. Die beiden Nucleus-Kanal-Ansichten zeigen, dass Regionen direkt um den Nucleus herum für die Klassifikationsent-

scheidung wichtiger sind als die Oberfläche des Nucleus. Selbst unter Verwendung der Distanz-Methode, die zum Ziel hat, Oberflächen-Merkmale zu erhalten und dafür klare Kanten und eine eindeutige Geometrie der Segmentierungsmaske augibt, wird die Oberfläche kaum betrachtet. Des Weiteren sind in den Heatmaps der Marker-Kanäle zu sehen, dass auch abseits des Nucleus-Kanals eine intuitiv sinnvolle Lokalisierung der Aufmerksamkeit des Klassifikators erfolgt. Die Maxima der Heatmaps sind auf sichtbare Strukturen im Marker-Kanal platziert.

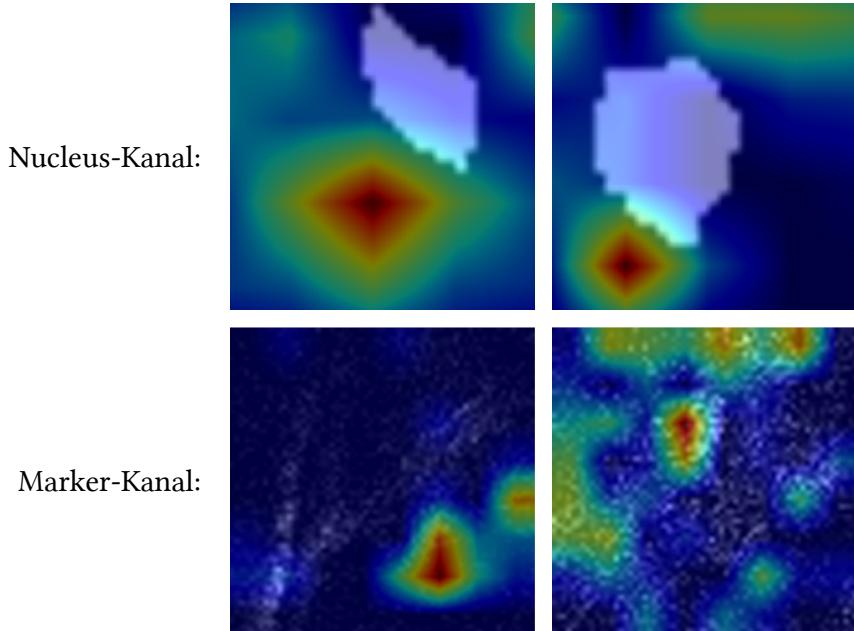


Abb. 5.8 | GradCAM-Visualisierung für zwei exemplarische 2D-Schichten, jeweils mit einem Nucleus-Kanal und einem Marker-Kanal.

5.4.5 Vortraining

Abb. 5.9 zeigt eine Übersicht über die Effektivität der verwendeten Vortrainingsmethoden. Um die Ergebnisse möglichst unabhängig von den anderen Klassifikatormethoden zu betrachten, werden hier nur die Klassifikatoren mit der besten Vorverarbeitungsmethode für den Encoder und dem Volumen-Klassifikator als Decoder betrachtet. Auf der vertikalen Achse sind die verschiedenen Encoder sowie der Durchschnitt aller Encoder zu sehen. Auf der horizontalen Achse sind die vier Vortrainingsmethoden aufgeführt. Links ist nur semi-supervised-Training zu sehen. Daneben steht das semi-supervised-Vortraining, also das Vortraining mit den Pseudo-Labler-Annotationen, Einfrieren der Encoder-Gewichte und Fortsetzen des Trainings mit vollständig annotierten Daten. Rechts ist das fully-supervised ImageNet-Vortraining mit eingefrorenen Encoder-Gewichten und kein Vortraining aufgegragen (siehe Kap. 3.3.4). Zu sehen ist, dass das Vortraining mit semi-supervised-Daten durchschnittlich deutlich schlechtere Ergebnisse liefert als das fully-supervised ImageNet-Vortraining und etwas schlechtere Ergebnisse als kein Vortraining. Im Durchschnitt führt das Fortführen des Trainings mit den annotierten Daten nach dem semi-supervised-Training

zu einer Steigerung der Genauigkeit um 13 Prozentpunkte. Für den ConvNeXt-Encoder ist der Vorteil des fortgesetzten Trainings besonders hoch mit 39 Prozentpunkten. Der ConvNeXt-Encoder ist außerdem mit dem semi-supervised-Vortraining um sieben Prozentpunkte genauer als ohne Vortraining. Auch die beiden ResNet-Modelle erzielen mit dem semi-supervised-Vortraining ähnlich gute Ergebnisse wie mit anderen Vortrainingsmethoden. Gegenüber dem ImageNet-Vortraining, das deutlich mehr Rechenaufwand erfordert, sind die Klassifikatoren mit ResNet18 und ResNet101 nur um zwei bzw. drei Prozentpunkte schlechter, wenn sie die Annotationen des Pseudo-Lablers als Vortraining nutzen. Beide haben außerdem eine höhere Genauigkeit als der Durchschnitt ohne Vortraining. Sie sind beide auch ohne Fortsetzung des Trainings nach dem semi-supervised-Vortraining bereits 64% und 62% genau. Besonders auffällig ist das Swin V2-Modell, das ohne Fully-supervised-Vortraining nur eine durchschnittliche Genauigkeit von 44,0% erreicht. Vortraining auf dem ImageNet-Datensatz führt im Vergleich dazu bei diesem Encoder zu einer Steigerung der Genauigkeit um 40 Prozentpunkte. 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 von 86%.

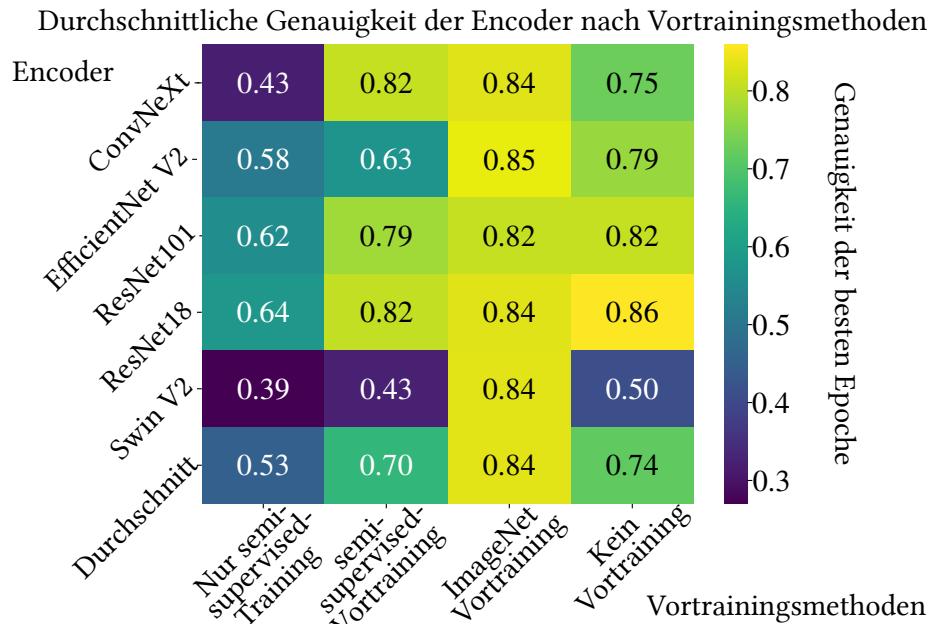


Abb. 5.9 | Die Heatmap zeigt die Genauigkeiten der Klassifikatoren bei Verwendung der verschiedenen Vortrainingsmethoden. Auf der Y-Achse sind die verschiedenen Encoder aufgeführt. Die X-Achse zeigt die getesteten Methoden des Vortrainings an. Die Farbe der Felder und der Wert darin zeigen die Durchschnittsgenauigkeiten aller Klassifikatoren, die die entsprechende Vortrainingsmethode nutzen. Die letzte Zeile zeigt den Durchschnitt der Werte aller Encoder, abhängig von der Vortrainingsmethode.

5.4.6 Allgemeines

Eine weitere angestellte Untersuchung ist die Quantifizierung der Wichtigkeiten der verschiedenen Marker-Kanäle. Auch für Expert*innen ist es schwer, eindeutige Beziehungen zwischen den Strukturen, die durch bestimmte Marker gefärbt werden, und den verschiedenen Klassen der Nuclei herzustellen. Zur Abschätzung der globalen Markerrelevanz werden die Gradienten der Modellvorhersage bis auf die Eingangsebene zurückprojiziert und die betragsmäßigen Gradienten über die räumlichen Dimensionen gemittelt. Die Wichtigkeitswerte, die sich ergeben, sind nahezu gleich verteilt.

Abb. 5.10 zeigt exemplarisch Gradientenfelder einer 2D-Schicht aus einer Stichprobe der dreidimensionalen Zieldaten. In den rohen Gradienten sind die sensitivsten Bereiche des Modells direkt aus den Ableitungen der Vorhersage bezüglich der Eingabe ablesbar. Diese Sensitivitäten sehen rauschig aus, weil die Effekte von punktweisen Variationen für jeden Pixel der Eingabedaten dargestellt sind. In der rechten Darstellung sind die integrierten Gradienten dargestellt. Durch die Integration sind Rauscheffekte deutlich reduziert, und eine stabilere, interpretierbarere Verteilung der relevanten Regionen im Gradientenfeld ist sichtbar. Zur Beurteilung der Konsistenz des Klassifikators werden die Ähnlichkeiten zwischen den Gradienten-Feldern quantitativ verglichen. Die Felder haben eine Kosinus-Ähnlichkeit von 0,913 und damit eine hohe Übereinstimmung in der Richtung der Wichtigkeitsverteilungen. Der Klassifikator reagiert also auf ähnliche Muster in beiden Feldern. Der hohe Wert spricht für eine stabile und konsistente Aufmerksamkeitslokalisierung des Klassifikators, was wiederum auf eine robuste interne Repräsentation der relevanten Merkmale hindeutet.

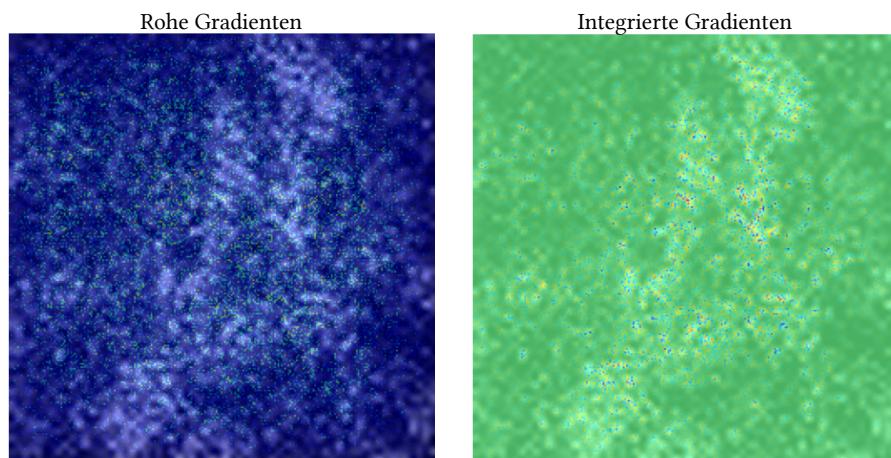


Abb. 5.10 | Die Abbildung zeigt zwei Gradienten-Felder einer 2D-Schicht einer Stichprobe der dreidimensionalen Zieldaten. Das rechte Gradientenfeld ist integriert, um hochfrequente lokale Unterschiede zu entfernen.

Des Weiteren wird die durchschnittliche Genauigkeit pro Klasse betrachtet. Abb. 5.11 zeigt die durchschnittliche Genauigkeit pro Klasse und Sterne als Hinweise auf signifikante Ergebnisse von paarweise-t-Tests. Die Klassen Eins bis Vier sind hier der Reihe nach Myotuben-Kerne, Debris, Andere und Schwannzellen-Kerne. In den Trainingsdaten

der vorliegenden Experimente sind zu sehen ist, dass insbesondere die Schwannzellen-Kerne schlecht erkannt werden. Wie eingangs beschrieben, sind nur 16 Beispiele der Klasse Schwannzellen-Nucleus im Datensatz enthalten, acht hiervon im Trainings- und acht im Test-Anteil. Wie in Kap. 3.3.3 beschrieben, wird dieses Problem der ungleichen Verteilung der Klassen durch den Retreiver durch eine Gewichtung der Gradienten unterrepräsentierter Klassen angegangen. In der Abbildung ist jedoch zu sehen, dass dieser Ansatz das Problem nicht vollständig gelöst hat. Die unterrepräsentierte Klasse der Schwannzellen-Nuclei wird höchst signifikant schlechter erkannt als alle anderen Klassen. Beinahe alle Methoden ergeben isoliert betrachtet eine ähnliche Verteilung der Genauigkeit pro Klassen, nur das Vortraining hat einen Einfluss. Vortraining mit semi-supervised-Annotationen und anschließendes Fortsetzen des Trainings mit manuell annotierten Daten führen zu Klassifikatoren, die teilweise wesentlich ausgeglichener die Klassen erkennen. Mit dem Volumen-Klassifikator und der Masken-Methode zur Vorverarbeitung erreichen der ResNet101-Encoder 83,2%, der ResNet18-Encoder 73,5% und der ConvNeXt-Encoder 56,3% Genauigkeit für die Klasse der Schwannzellen-Nuclei. Das ist im Vergleich zu den 21,7% durchschnittlicher Genauigkeit für Klasse vier ein starker Gewinn.

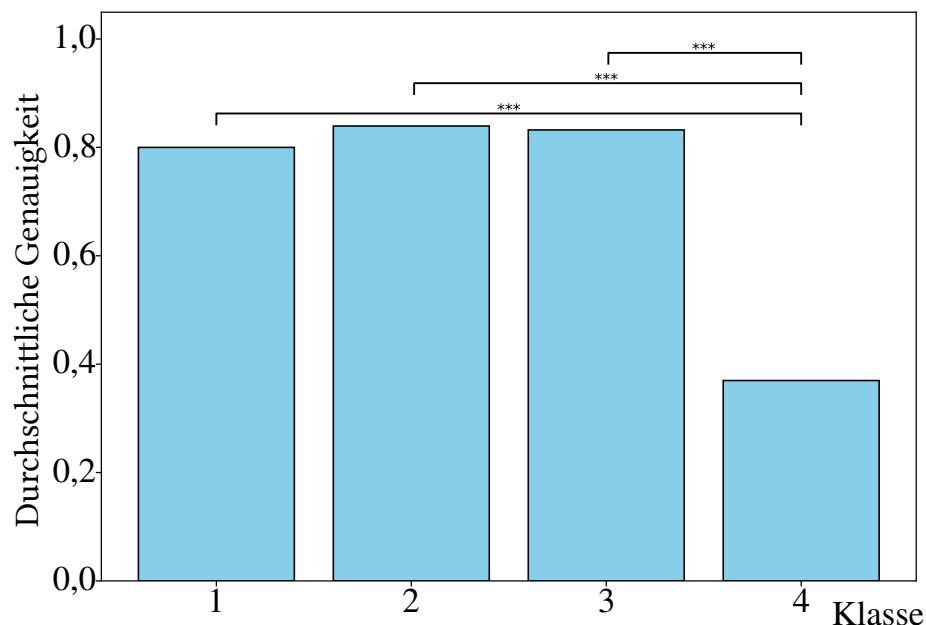


Abb. 5.11 | Das Balkendiagramm zeigt die durchschnittliche Genauigkeit pro Klasse für alle Klassifikatoren. Die Klassen Eins bis Vier sind hier der Reihe nach Myotuben-Kerne, Debris, Andere und Schwannzellen-Kerne.

Diskussion 6

6.1 Überblick

Die nachfolgenden Kapitel diskutieren die Ergebnisse der vorliegenden Arbeit. Dabei wird sowohl auf die quantitative Auswertung der durchgeführten Experimente eingegangen, als auch auf die Bewertung der angewandten Methoden. Es wird gezeigt, dass die neu entwickelte IPQ-Metrik geeignet ist um Instanzsegmentierungsmodelle, im Bezug auf ihre Eignung interpretierbare Merkmale zu extrahieren, zu bewerten. Außerdem wird gezeigt, dass die neu entwickelte Methode des Vortraining einen positiven Einfluss auf die Klassifikationsgenauigkeit haben kann und, dass die 3D-Zelldaten-Pipeline effizient optimale Deep-Learning-Methoden für neue Bilddomänen identifiziert.

6.2 Segmentierung

Durch einen Vergleich der Masken mit der Annotationen 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. Mit dem nnUNet-Modell sind außerdem aufgrund der kleinen Segmente, die das Modell vorhersagt, Überschneidungen zwischen Annotationsmaske und Segmentierungsmaske gering. Ein Weg die Leistung des Modells im Bezug auf die IPQ-Metrik zu verbessern ist deshalb vermutlich das Modell, beispielsweise durch fine-tuning, auf größere Segmente anzupassen. Eingangsdaten, beispielsweise durch einen Mittelwert-Filter, kleiner zu dimensionieren und mehrere Bilder aneinander gereiht einzugeben kann auch zu Verbesserungen führen, dabei besteht allerdings das Risiko, dass der Informationsverlust durch den Filter das Ergebnis negativ beeinflusst.

Deepcell (siehe Abb. 5.1a) übersegmentiert die Nuclei, wodurch die Segmentierungsqualität stark abnimmt. Das Ergebnis sind Masken, die intuitiv 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 Annotation durch die Deepcell-Masken geteilt werden, was zu einer guten Injektiven Qualität führt. Außerdem ist durch die großen Segmente die Segmentierungs-

Qualität besonders schlecht, weil die Größe der Segmente den Nenner der IoU erhöht. Die Deepcell-Masken können im Bezug auf die IPQ von weiterer Nachverarbeitung profitieren. Mithilfe eines Erosion-Filter können die Masken kleiner gemacht werden, was auch zu einer besseren Leistung der Watershed-Nachverarbeitung führen kann. Umgekehrt kann nach der Erosion die Maske aber auch Kerben an der Kontur einzelner Instanzen aufweisen, die zur Spaltung der Instanz durch die Watershed-Nachverarbeitung führen. Auch für das Deepcell-Modell ist fine-tuning zu Anpassung an die Größe der Nuclei vermutlich ein Weg die Qualität zu verbessern.

Sowohl für das nnUNet-, als auch für das Deepcell-Modell gibt es zwar Vorschläge für Methoden zur Verbesserung der IPQ auf dem Benchmark-Datensatz, aber der Übertrag der Effektivität dieser Methoden auf die Zieldaten ist fraglich. Die Bilddomäne und auch die Eigenschaften wie Größe, Exzentrizität und Rundheit der Zieldaten unterscheiden sich von denen des Benchmark. Da keine Annotationen für die Instanzsegmentierung der Zieldaten verfügbar sind wird von der Anwenden der Methoden im Zuge dieser Arbeit abgesehen, um kein Overfitting auf den Benchmark-Datensatz zu riskieren.

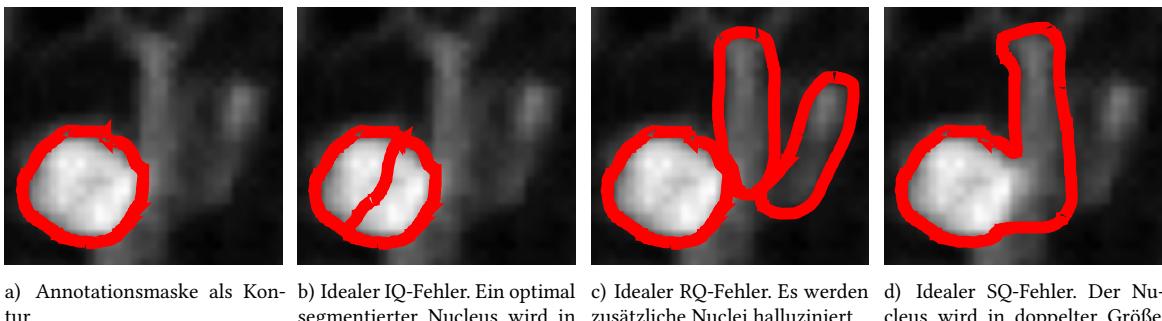


Abb. 6.1 | Darstellung der Segmentierungsmasken verschiedener idealer Fehler und der Annotation als Kontur.

Anhand der Interpretation der Werte und der exemplarischen Fehler in Abb. 5.3 wird die Effektivität der neu entwickelten IPQ-Metrik ersichtlich. Verschiedene Fehlerarten, die zu unterschiedlichen Verfälschungen der interpretierbaren Merkmale der Daten führen werden gezielt von der Metrik erfasst. In der Praxis gehen die Fehler oft mit Fehlern einer anderen Art einher, was daran liegt, dass die Modelle nicht dem selben Denkverhalten folgen wie ein menschlicher Betrachter. Abb. 6.1 zeigt idealisiert die Fehlerarten, die mit der IPQ-Metrik erkannt werden. Alle Fehler führen für den betrachteten Nucleus zu einem Faktor von 0,5 in ihrer Kategorie und einem optimalen Wert von 1 in den anderen Kategorien. Hier wird deutlich, dass die Metrik geeignet ist um:

- Fehleinschätzungen der Konzentration von Nuclei durch den neu eingeführten IQ-Wert zu bestrafen,
- Fehleinschätzungen des Volumens der Nuclei durch einen geringen SQ-Wert zu bestrafen und

- Fehleinschätzungen der Anzahl der Nuclei durch einen geringen RQ-Wert zu bestrafen.

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. Vor allem aber zeigen die Ergebnisse, dass die Anwendung der optimalen Klassifikator-Methoden einen sehr hoch signifikanten Einfluss auf die Genauigkeit des Klassifikators hat. Die 3D-Zelldaten-Pipeline findet diese optimalen Methoden effizient. Wie die Ergebnisse zeigen, ermöglicht die Anwendung der 3D-Zelldaten-Pipeline das Training eines Klassifikators, der deutlich leistungsfähiger ist als ein Klassifikator mit zufällig gewählten Methoden – und das ganz ohne Programmierkenntnisse oder Vorerfahrung mit Deep-Learning-Methoden seitens der Anwender*innen. Zuerst werden die Ergebnisse abhängig vom Encoder diskutiert. Die Ergebnisse haben gezeigt, dass die beiden ResNet-Encoder besonders hohe Genauigkeiten im Durchschnitt haben. Da die ResNet-Encoder auch die Encoder mit der geringsten Parameteranzahl sind liegt die Vermutung nahe, dass die gewählten Encoder zu groß sind und schlechte Ergebnisse einem Architektur-bedingtem Overfitting geschuldet sind. Abb. 6.2 zeigt die durchschnittliche Genauigkeit aller Klassifikatoren abhängig von der Parameteranzahl des genutzten Encoders.

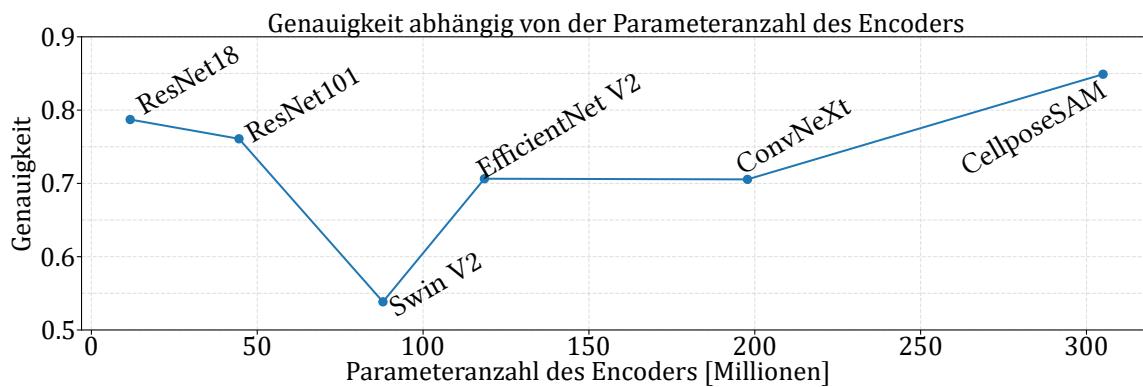


Abb. 6.2 | Die Abbildung zeigt die durchschnittliche Genauigkeit aller Klassifikatoren abhängig von der Parameteranzahl ihres Encoders. Die X-Achse zeigt die Parameteranzahl, skaliert mit einer Millionen und die Y-Achse die durchschnittliche Genauigkeit.

Intuitiv ist kein Zusammenhang zwischen der Parameteranzahl und der durchschnittlichen Genauigkeit zu erkennen und auch die Spearman-Korrelationsanalyse deutet nicht auch einen Zusammenhang hin. Das bedeutet, die Encoder sind nicht einfach nur zu groß. Die Architektur und die Gewichte der Encoder bestimmen den Merkmalsraum, den die Encoder mit der internen Repräsentation der Eingangsdaten aufspannen. Dieser Merkmalsraum muss für eine genaue Klassifikation relevante Merkmale des klassifizierten Objekts

abbilden. Der Swin V2-Encoder hat eine Transformer-Architektur und schneidet besonders schlecht im Durchschnitt. Vermutlich sind die Ansprüche an den Trainingsumfang für Transformer-Architekturen besonders hoch und in den relativ kurzen Trainingsabläufen wird kein ausreichender Merkmalsraum gelernt. Für die CNN-Architekturen ist das relativ kurze Training weniger problematisch. Bei Betrachtung ausschließlich der CNN zeigt sich ein negativer Zusammenhang zwischen Parameteranzahl des Encoders und der durchschnittlichen Genauigkeit, allerdings ist die Stichprobenanzahl für eine Korrelationsanalyse sehr gering. Zukünftige Forschung kann weitere, kleinere CNN-Encoder in Betracht ziehen, um die Hypothese zu prüfen, dass die Encoder zu groß sind um eine sinnvolle Repäsentation der räumlich relativ kleinen Nuclei zu finden.

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 kann daran liegen, dass die dreidimensionale Faltung räumliche Mittlungen 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 führt. Stattdessen können auch die Kombination räumlicher Merkmale und der Merkmale entlang der Z-Achse wichtig sein, was die gute Leistung des Volumen-Klassifikators bedingen kann. 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. Besonders da die ImageNet-Encoder nicht auf dreidimensionale Eingabedaten vtrainiert werden ist die Abwägung zwischen einem Fokus auf die Merkmale entlang der Z-Achse und räumlichen Merkmalen essenziell. Auch für zukünftige Encoder besteht die Möglichkeit, dass sie Beziehungen von Merkmalen innerhalb einer 2D-Schicht der Eingabedaten so effizient und einheitlich zusammenfassen, dass der Attention-Mechanismus des Schichten-Klassifikators die Informationen besser als der Volumen-Klassifikator erhalten kann.

Auch für Vorverarbeitung ist eine signifikant bessere Methode zu erkennen. Die Masken-Methode ist signifikant besser als die Distanz-Methode. Kapitel 3.3 beschreibt den Vorteil und das Risiko der Methoden. Insbesondere der Vorteil der Masken-Methode wird an den Ergebnissen sichtbar. Anhand der GradCAM-Heatmap in Abb. 5.8 wird abgeleitet, dass die Oberflächenmerkmale der Nuclei für die Klassifikation keine Rolle spielen. Ein Risiko der Masken-Methode ist ein Qualitäts-Verlust aufgrund der Eliminierung der Oberflächenmerkmale, aber selbst mit der Distanz-Methode werden diese Merkmale nicht betrachtet. Die Masken-Methode hebt die Geometrie der Nuclei besonders hervor und sowohl die statistische Betrachtung der Ergebnisse als auch die GradCAM-Heatmaps legen nahe, dass diese Geometrie besonders wichtig für die Klassifikation der Nuclei ist. Mit der Distanz-Methode wird nicht nur die Geometrie nicht besonders hervorgehoben, unter Umstän-

den wird sie sogar durch direkt umliegende Nuclei beeinträchtigt. Da die Segmentierungs-Maske außer in der Distanztransformation nicht in den Daten der Distanz-Methode vorhanden ist, geht die Information über die Kanten vollständig verloren, überall umliegende Nuclei den betrachteten Nucleus direkt berühren. Der berührende Nucleus wird in diesem Fall als Teil des betrachteten Objekts interpretiert. Für kommende Anwendungen ist dennoch die Überlegung der Auswahl einer Vorverarbeitungsmethode interessant, da die Nuclei von anderen Zieldaten Informationen auf der Oberfläche tragen können, die für die Klassifikation wichtig sind.

Der Einfluss der Vortrainingsmethoden ist besonders bedeutsam. Mit dem fully-supervised Vortraining ist die durchschnittliche Genauigkeit höher und auch konsistenter als mit anderen Vortrainingsmethoden. Das legt nahe, dass das rechenaufwändige Vortraining auf dem ImageNet-Datensatz mit den diversen Klassen und zahlreichen Stichproben für jede Klasse einen stark generalisierten Merkmalsraum erzeugt. Obwohl die Bilddomäne biologischer Zelldaten stark von den Bildern des ImageNet-Datensatz abweicht werden bedeutsame Bildmerkmale extrahiert. Auch bei der Adaption von 2D-Encodern zu 3D-Encodern bleibt der Merkmalsraum der fully-supervised-vortrainierten Encoder sinnvoll. Da die Zieldaten niemals vom Encoder gesehen werden ist Overfitting weitgehend ausgeschlossen, was die geringe Varianz der Genauigkeiten innerhalb der fully-supervised-vortrainierten Klassifikatoren erklärt.

Auch komplett ohne Vortraining wird ein sinnvoller Merkmalsraum gelernt und die Klassifikatoren erreichen teilweise hohe Genauigkeitswerte. Der Swin V2-Encoder hat ohne Vortraining nur 50% Genauigkeit erreicht, was vermutlich daran liegt, dass die aufwändige Transformer-Architektur nicht ausreichen mit dem geringen Trainingsumfang der Zieldaten lernen kann. Es wird keine sinnvolle Merkmalsextraktion gelernt, bevor der Encoder mit dem Overfitting beginnt. Das legt auch nahe, dass Encoder mit besonders hoher Parameteranzahl nicht ohne ImageNet-Vortraining auskommen. Die Beobachtung, dass der kleinste Encoder, der ResNet18-Encoder, ohne Vortraining das beste Ergebnis liefert hat unterstützt diese Aussage noch. Abb. 6.3 zeigt die Genauigkeiten der verschiedenen Vortrainingsmethoden abhängig von der Parameteranzahl des genutzten Encoders.

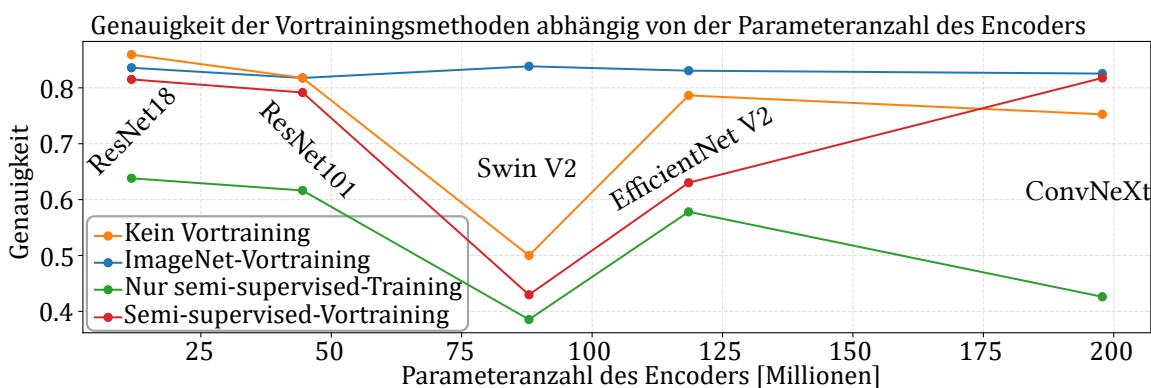


Abb. 6.3 | Die Abbildung zeigt die Genauigkeiten der verschiedenen Vortrainingsmethoden abhängig von der Parameteranzahl des genutzten Encoders.

Die Abbildung zeigt, dass der Zusammenhang zwischen der Effizienz der Vortrainingsmethoden und der Parameteranzahl nicht so einfach darzustellen ist. Mit den ResNet-Encodern ist es möglich das ImageNet-Vortaining auszulassen und die Merkmalsextraktion direkt durch die Zieldaten zu lernen. Sowohl ohne Vortraining, als auch mit dem semi-supervised-Vortraining ist die Genauigkeit der Klassifikatoren vergleichbar zum ImageNet-Vortraining. In Anbetracht des hohen Rechenaufwand, den das ImageNet-Vortraining beansprucht, ist dieses Ergebnis besonders interessant. Beide Methoden erhöhen den Rechenaufwand für das fine-tuning eines Klassifikators auf eine neue Domäne, machen dafür aber das exzessive Vortraining überflüssig. Insgesamt ergibt sich hierdurch vermutlich eine geringere Generalisierbarkeit, aber für den angewandten Datensatz wird dafür die Genauigkeit besser. Auch die anderen beiden CNN-Encoder können ohne ImageNet-Vortraining genutzt werden. Der EfficientNet V2-Encoder ist ohne Vortraining in der Lage ein vergleichbare Genauigkeit zu erzielen wie mit dem fully-supervised-Vortraining. Dass das semi-supervised-Vortraining ein schlechtes Ergebnis erzielt ist demnach vermutlich dem kurzen Training geschuldet. Da der EfficientNet V2-Encoder ohne Vortraining einen sinnvollen Merkmalsraum direkt auf den Zieldaten lernen kann liegt nahe, dass entweder zu wenige Epochen mit den semi-supervised-Annotationen gelernt wurde um diesen Merkmalsraum zu erfassen oder, dass anschließend der Klassifikator-Kopf zu kurz auf dem neu geschaffenen Merkmalsraum trainiert wurde, um die Klassifikationsentscheidung darin optimal zu lernen. Der ConvNeXt-Encoder hingegen hat in den gegebenen Epochen eine sinnvolle Repräsentation gelernt und ist durch das semi-supervised-Vortraining besser, als ohne Vortraining. Vermutlich liegt das daran, dass der Encoder im umfangreichen semi-supervised-Vortraining eine besser generalisierte Repräsentation gelernt hat. Außerdem liegt es nahe, dass der ConvNeXt-Encoder ohne Vortraining den Merkmalsraum auf die Trainingdaten über-anpasst. Der Swin V2-Encoder ist viel weniger genau, wenn das ImageNet-Vortraining ausgelassen wird. Das liegt vermutlich daran, dass der relativ große Encoder mit der komplexen ViT-Architektur aus den wenigen Stützvektoren der Repräsentation der Eingangsdaten keinen ausreichenden Merkmalsraum aufspannen kann. Außerdem ist es möglich, dass der Klassifikator-Kopf nur schwach generalisiert die Beziehung zwischen Merkmalen und Entscheidung lernen kann, weil der finale Merkmalsvektor der ViT-Architektur eine viel höhere Dimension hat.

Die Genauigkeit der meisten Encoder ist auch ohne ImageNet-Vortaining annehmbar, vor allem da mithilfe der 3D-Zelldaten-Pipeline auch ein optimaler Encoder passend zur Vortrainingsmethode gewählt wird. Die durchschnittlich geringere Genauigkeit ist deshalb nicht wichtig, wenn das Optimum besser ist.

Trotz der hohen durchschnittlichen Genauigkeit des fully-supervised-Vortraining und ohne Vortraining ist der Merkmalsraum nicht geeignet um besonders Myotuben-Kerne und Schwannzellen-Kerne zu unterscheiden. Die Unterscheidung dieser Klassen ist für die vorliegenden Daten wichtiger, als die Unterscheidung der Ändereklasse und der Debris-Klasse. Die Myotuben- und Schwannzellen-Klassen sind optisch sehr ähnlich und nur durch Expert*Innen unter Berücksichtigung umliegender Strukturen trennbar. Abb. 6.4 zeigt jeweils zwei Beispiele der beiden Klassen. Zu sehen sind die Nuclei in Rot und Myotuben in Grün, sowie der S100 β Marker. Die beiden Beispiele aus derselben Reihe

sind visuell kaum voneinander zu unterscheiden. Durch die Projektion gehen Distanzen zwischen den Stichproben entlang der vielen Dimensionen des Merkmalvektors verloren, aber mithilfe der TSNE ist die Abbildung als Übersicht dennoch geeignet. Der linke Merkmalsraum ist aus einem ImageNet-Vortraining entstanden, der rechte durch ein semi-supervised-Vortraining. Zu sehen ist, dass im linken Scatterplot die Klassen Eins und Vier, also Myotuben- und Schwannzellen-Nuclei eine besonders starke Überschneidung haben.

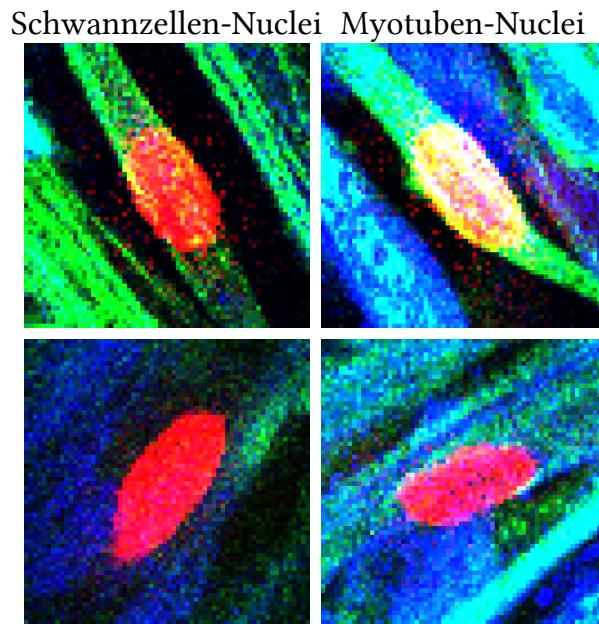


Abb. 6.4 | Die Abbildung zeigt zwei Beispiele von Nuclei der Schwannzellen-Klasse (links) und zwei Beispiele von Nuclei der Myotuben-Klasse (rechts). Die beiden Reihen zeigen Nuclei die visuell besonders ähnlich sind.

Die Schwannzellen-Klasse wird mit einer höchst signifikant schlechteren Genauigkeit erkannt (siehe Kap. 5.4.6). Auch für die Klassifikatoren sind die beiden Klassen also schwer trennbar. Das zeigt insbesondere auch die Projektion der Klassenentscheidungen der Klassifikatoren auf einen niederdimensionalen, visualisierbaren Merkmalsraum. In Abb. 6.5 sind zwei exemplarische Merkmalsräume als 2D-Projektion zu sehen. Für die Klassen Zwei und Drei sind dagegen separate Cluster erkennbar. Im rechten Scatterplot ist das Schwannzellen-Klassen-Cluster zwar immernoch nicht weit entfernt von anderen Clustern, aber wesentlich kompakter, als in der linken Abbildung.

Für die Unterscheidung der Myotuben- und Schwannzellen-Nuclei ist also das Vortraining mit semi-supervised-Annotationen besonders hilfreich. Vermutlich wird durch das Exzessive Vortraining auf den Zieldaten dem Encoder ein so gut generalisierter Merkmalsraum antrainiert, dass der Decoder die beiden Klassen trotz der Überschneidungen ihrer Merkmale trennen kann. In den Zieldaten hingegen sind diese Merkmale häufig vertreten und werden deshalb so hochauflösend extrahiert, dass es zur Trennung der Klassen reicht. Die Auflösung genau der Merkmale die für das Trennen der beiden Klassen nötig sind, ist nach dem ImageNet-Datensatz-Vortraining vermutlich nicht sehr hoch.

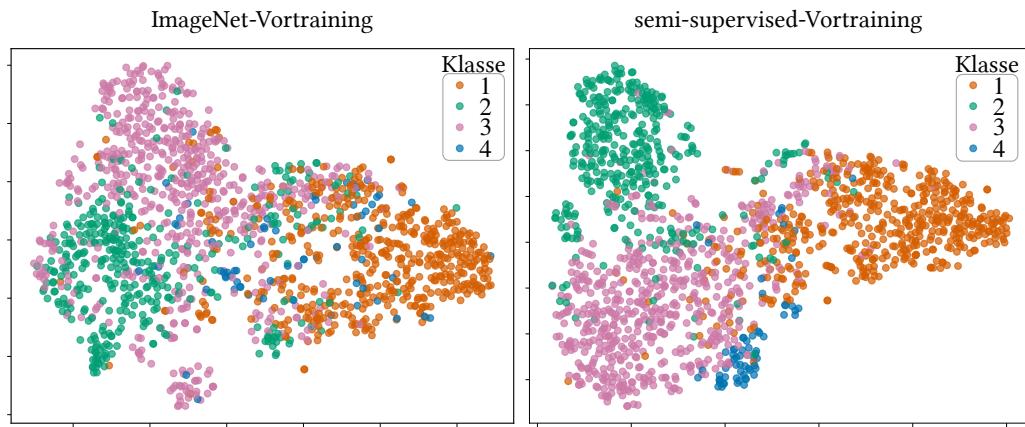


Abb. 6.5 | Die Scatterplots zeigen 2D-Projektionen des Merkmalraums von zwei Klassifikatoren durch TS-NE. Der linke Scatterplot zeigt den Merkmalsraum eines ImageNet-vortrainierten Klassifikators, der rechte eines semi-supervised-vortrainierten Klassifikators. Die Werte der X- und Y-Achse sind arbiträre Aktivierungsintensitäten ohne physische Interpretation.

Die durchschnittliche Genauigkeit der semi-supervised-vortrainierten Klassifikatoren ist zwar geringer, als die der anderen Vortrainingsmethoden, dafür ist der entstandene Klassifikator aber besser generalisiert. Da im Test-Anteil der annotierten Daten nur acht der 81 Stichproben die Klasse Schwannzellen-Nucleus besitzen ist die Genauigkeit der semi-supervised-vortrainierten Klassifikatoren nicht zwingend repräsentativ für die tatsächliche Leistung. Für die Wahl einer Vortrainingsmethode müssen demnach die genauen Ziele der Anwendung feststehen. Wenn einzelne Klassen besonders wichtig für die Interpretation der Ergebnisse sind, aber nicht häufig vorkommen lohnt es sich den semi-supervised-Ansatz zu testen, um eventuell durch die hohe Generalisierungsfähigkeit einen besseren Klassifikator zu erhalten.

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

Appendix A

A.1 SWINV2 Architektur

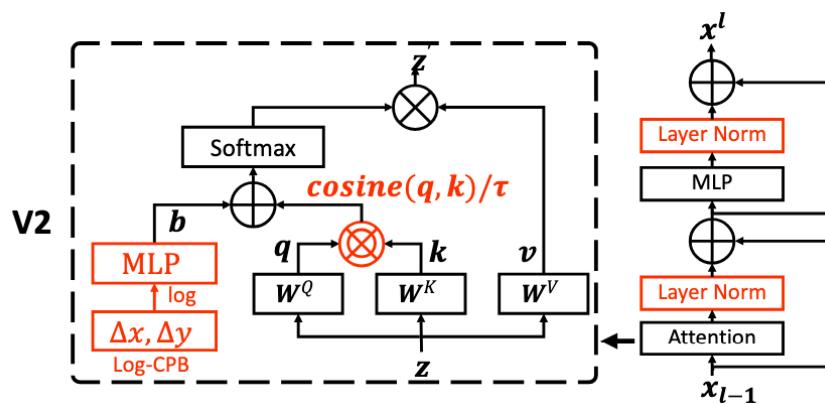


Abb. A.1 | Swin Transformer V2 Architektur [151]. Ein Bildfenster z und dessen relative Koordinaten im Bild Δx und Δy werden in einem Attention-Mechanismus zusammengeführt. Mithilfe einer Kosinus-Ähnlichkeitsfunktion, der Softmax-Funktion [152] und elementweiser Multiplikation sowie Addition werden diese 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**.

A.2 IPQ-Ergebnisse

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 1 | 0.315 | 0.082 | 0.010 | 2 | 0.349 | 0.066 | 0.015 |
| 3 | 0.326 | 0.139 | 0.008 | 4 | 0.405 | 0.138 | 0.017 |
| 5 | 0.313 | 0.082 | 0.006 | 6 | 0.323 | 0.139 | 0.008 |
| 7 | 0.349 | 0.066 | 0.015 | 8 | 0.406 | 0.139 | 0.018 |
| 9 | 0.656 | 0.249 | 0.013 | 10 | 0.449 | 0.052 | 0.026 |
| 11 | 0.500 | 0.056 | 0.022 | 12 | 0.500 | 0.056 | 0.025 |
| 13 | 0.452 | 0.052 | 0.026 | 14 | 0.645 | 0.039 | 0.063 |
| 15 | 0.529 | 0.056 | 0.112 | 16 | 0.645 | 0.039 | 0.063 |
| 17 | 0.529 | 0.056 | 0.112 | 18 | 0.670 | 0.148 | 0.058 |
| 19 | 0.304 | 0.048 | 0.015 | 20 | 0.297 | 0.197 | 0.013 |
| 21 | 0.464 | 0.058 | 0.077 | 22 | 0.604 | 0.086 | 0.021 |
| 23 | 0.516 | 0.114 | 0.029 | 24 | 0.585 | 0.151 | 0.057 |
| 25 | 0.544 | 0.115 | 0.029 | 26 | 0.507 | 0.097 | 0.017 |
| 27 | 0.576 | 0.172 | 0.030 | 28 | 0.502 | 0.118 | 0.030 |
| 29 | 0.697 | 0.022 | 0.040 | 30 | 0.627 | 0.025 | 0.014 |
| 31 | 0.646 | 0.023 | 0.031 | 32 | 0.613 | 0.029 | 0.030 |
| 33 | 0.586 | 0.143 | 0.022 | 34 | 0.874 | 0.149 | 0.141 |
| 35 | 0.862 | 0.159 | 0.054 | 36 | 0.868 | 0.180 | 0.080 |
| 37 | 0.879 | 0.148 | 0.079 | 38 | 0.854 | 0.131 | 0.154 |
| 39 | 0.852 | 0.167 | 0.121 | 40 | 0.874 | 0.210 | 0.044 |
| 41 | 0.846 | 0.222 | 0.095 | 42 | 0.881 | 0.143 | 0.096 |
| 43 | 0.876 | 0.160 | 0.109 | 44 | 0.880 | 0.139 | 0.067 |
| 45 | 0.860 | 0.149 | 0.104 | 46 | 0.838 | 0.121 | 0.099 |
| 47 | 0.872 | 0.146 | 0.113 | 48 | 0.852 | 0.138 | 0.080 |
| 49 | 0.863 | 0.174 | 0.098 | 50 | 0.867 | 0.162 | 0.094 |
| 51 | 0.854 | 0.142 | 0.091 | 52 | 0.872 | 0.199 | 0.118 |
| 53 | 0.871 | 0.134 | 0.121 | 54 | 0.873 | 0.154 | 0.076 |
| 55 | 0.877 | 0.157 | 0.114 | 56 | 0.880 | 0.135 | 0.126 |
| 57 | 0.857 | 0.148 | 0.075 | 58 | 0.861 | 0.183 | 0.099 |
| 59 | 0.865 | 0.151 | 0.085 | 60 | 0.850 | 0.184 | 0.085 |
| 61 | 0.857 | 0.170 | 0.154 | 62 | 0.881 | 0.160 | 0.097 |
| 63 | 0.851 | 0.113 | 0.102 | 64 | 0.861 | 0.173 | 0.170 |
| 65 | 0.885 | 0.119 | 0.105 | 66 | 0.875 | 0.185 | 0.144 |
| 67 | 0.863 | 0.169 | 0.084 | 68 | 0.853 | 0.171 | 0.129 |
| 69 | 0.850 | 0.154 | 0.158 | 70 | 0.867 | 0.146 | 0.110 |
| 71 | 0.859 | 0.139 | 0.132 | 72 | 0.856 | 0.133 | 0.153 |
| 73 | 0.898 | 0.114 | 0.094 | 74 | 0.879 | 0.123 | 0.098 |
| 75 | 0.860 | 0.233 | 0.117 | 76 | 0.879 | 0.162 | 0.138 |
| 77 | 0.855 | 0.133 | 0.099 | 78 | 0.842 | 0.185 | 0.137 |

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 79 | 0.876 | 0.167 | 0.137 | 80 | 0.872 | 0.154 | 0.117 |
| 81 | 0.880 | 0.161 | 0.088 | 82 | 0.879 | 0.160 | 0.142 |
| 83 | 0.879 | 0.124 | 0.106 | 84 | 0.869 | 0.141 | 0.096 |
| 85 | 0.856 | 0.158 | 0.100 | 86 | 0.837 | 0.163 | 0.062 |
| 87 | 0.863 | 0.149 | 0.108 | 88 | 0.867 | 0.187 | 0.119 |
| 89 | 0.874 | 0.135 | 0.086 | 90 | 0.845 | 0.141 | 0.099 |
| 91 | 0.860 | 0.181 | 0.199 | 92 | 0.868 | 0.168 | 0.111 |
| 93 | 0.828 | 0.139 | 0.131 | 94 | 0.858 | 0.146 | 0.084 |
| 95 | 0.886 | 0.144 | 0.093 | 96 | 0.863 | 0.115 | 0.102 |
| 97 | 0.862 | 0.139 | 0.112 | 98 | 0.863 | 0.123 | 0.133 |
| 99 | 0.856 | 0.168 | 0.130 | 100 | 0.875 | 0.129 | 0.111 |
| 101 | 0.870 | 0.150 | 0.077 | 102 | 0.863 | 0.142 | 0.092 |
| 103 | 0.843 | 0.145 | 0.125 | 104 | 0.856 | 0.161 | 0.197 |
| 105 | 0.879 | 0.143 | 0.091 | 106 | 0.873 | 0.135 | 0.142 |
| 107 | 0.870 | 0.148 | 0.105 | 108 | 0.845 | 0.143 | 0.055 |
| 109 | 0.869 | 0.177 | 0.097 | 110 | 0.855 | 0.171 | 0.112 |
| 111 | 0.894 | 0.143 | 0.118 | 112 | 0.865 | 0.130 | 0.051 |
| 113 | 0.863 | 0.175 | 0.112 | 114 | 0.884 | 0.154 | 0.116 |
| 115 | 0.848 | 0.137 | 0.099 | 116 | 0.874 | 0.154 | 0.104 |
| 117 | 0.857 | 0.171 | 0.061 | 118 | 0.876 | 0.137 | 0.091 |
| 119 | 0.871 | 0.143 | 0.050 | 120 | 0.872 | 0.151 | 0.111 |
| 121 | 0.876 | 0.128 | 0.138 | 122 | 0.873 | 0.180 | 0.104 |
| 123 | 0.857 | 0.170 | 0.108 | 124 | 0.870 | 0.185 | 0.131 |
| 125 | 0.861 | 0.183 | 0.085 | 126 | 0.577 | 0.050 | 0.078 |
| 127 | 0.560 | 0.107 | 0.037 | 128 | 0.572 | 0.090 | 0.037 |
| 129 | 0.585 | 0.113 | 0.059 | 130 | 0.591 | 0.088 | 0.013 |

Tab. A.1 | Einzelne SQ-Ergebnisse von jedem Segmentierungsnetz

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 1 | 0.794 | 0.741 | 0.271 | 2 | 0.958 | 0.872 | 0.176 |
| 3 | 0.808 | 0.722 | 0.212 | 4 | 0.946 | 0.845 | 0.225 |
| 5 | 0.797 | 0.739 | 0.275 | 6 | 0.814 | 0.716 | 0.222 |
| 7 | 0.961 | 0.868 | 0.179 | 8 | 0.950 | 0.839 | 0.225 |
| 9 | 0.844 | 0.664 | 0.347 | 10 | 0.942 | 0.565 | 0.509 |
| 11 | 0.862 | 0.760 | 0.430 | 12 | 0.862 | 0.769 | 0.382 |
| 13 | 0.937 | 0.570 | 0.514 | 14 | 0.902 | 0.892 | 0.306 |
| 15 | 0.953 | 0.921 | 0.347 | 16 | 0.902 | 0.892 | 0.306 |
| 17 | 0.953 | 0.921 | 0.347 | 18 | 0.663 | 0.688 | 0.453 |
| 19 | 0.702 | 0.751 | 0.308 | 20 | 0.773 | 0.712 | 0.184 |

A Appendix

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 21 | 0.773 | 0.825 | 0.144 | 22 | 0.956 | 0.930 | 0.167 |
| 23 | 0.947 | 0.813 | 0.341 | 24 | 0.922 | 0.789 | 0.452 |
| 25 | 0.945 | 0.879 | 0.346 | 26 | 0.935 | 0.821 | 0.348 |
| 27 | 0.964 | 0.824 | 0.329 | 28 | 0.904 | 0.783 | 0.418 |
| 29 | 0.937 | 0.948 | 0.323 | 30 | 0.947 | 0.870 | 0.346 |
| 31 | 0.930 | 0.918 | 0.381 | 32 | 0.938 | 0.903 | 0.321 |
| 33 | 0.980 | 0.804 | 0.532 | 34 | 0.847 | 0.937 | 0.261 |
| 35 | 0.872 | 0.909 | 0.300 | 36 | 0.833 | 0.927 | 0.218 |
| 37 | 0.813 | 0.943 | 0.295 | 38 | 0.862 | 0.957 | 0.289 |
| 39 | 0.816 | 0.951 | 0.229 | 40 | 0.820 | 0.940 | 0.196 |
| 41 | 0.819 | 0.914 | 0.259 | 42 | 0.826 | 0.951 | 0.308 |
| 43 | 0.820 | 0.953 | 0.330 | 44 | 0.794 | 0.881 | 0.168 |
| 45 | 0.823 | 0.944 | 0.206 | 46 | 0.867 | 0.890 | 0.158 |
| 47 | 0.862 | 0.930 | 0.283 | 48 | 0.843 | 0.940 | 0.364 |
| 49 | 0.870 | 0.909 | 0.149 | 50 | 0.813 | 0.906 | 0.263 |
| 51 | 0.761 | 0.945 | 0.294 | 52 | 0.813 | 0.901 | 0.323 |
| 53 | 0.826 | 0.947 | 0.227 | 54 | 0.877 | 0.935 | 0.178 |
| 55 | 0.877 | 0.940 | 0.275 | 56 | 0.847 | 0.961 | 0.168 |
| 57 | 0.864 | 0.935 | 0.265 | 58 | 0.833 | 0.950 | 0.224 |
| 59 | 0.813 | 0.943 | 0.250 | 60 | 0.829 | 0.908 | 0.267 |
| 61 | 0.872 | 0.941 | 0.283 | 62 | 0.826 | 0.960 | 0.226 |
| 63 | 0.816 | 0.943 | 0.082 | 64 | 0.857 | 0.950 | 0.289 |
| 65 | 0.800 | 0.910 | 0.263 | 66 | 0.840 | 0.937 | 0.330 |
| 67 | 0.855 | 0.952 | 0.317 | 68 | 0.816 | 0.942 | 0.176 |
| 69 | 0.872 | 0.945 | 0.200 | 70 | 0.794 | 0.933 | 0.235 |
| 71 | 0.864 | 0.943 | 0.294 | 72 | 0.836 | 0.933 | 0.217 |
| 73 | 0.840 | 0.973 | 0.272 | 74 | 0.800 | 0.947 | 0.286 |
| 75 | 0.862 | 0.931 | 0.182 | 76 | 0.893 | 0.943 | 0.220 |
| 77 | 0.833 | 0.928 | 0.238 | 78 | 0.911 | 0.935 | 0.222 |
| 79 | 0.862 | 0.919 | 0.174 | 80 | 0.847 | 0.908 | 0.229 |
| 81 | 0.794 | 0.945 | 0.224 | 82 | 0.893 | 0.963 | 0.323 |
| 83 | 0.870 | 0.952 | 0.206 | 84 | 0.791 | 0.908 | 0.240 |
| 85 | 0.887 | 0.935 | 0.272 | 86 | 0.847 | 0.936 | 0.218 |
| 87 | 0.909 | 0.937 | 0.162 | 88 | 0.909 | 0.927 | 0.348 |
| 89 | 0.806 | 0.933 | 0.289 | 90 | 0.773 | 0.920 | 0.168 |
| 91 | 0.806 | 0.924 | 0.152 | 92 | 0.840 | 0.933 | 0.289 |
| 93 | 0.852 | 0.927 | 0.271 | 94 | 0.813 | 0.917 | 0.118 |
| 95 | 0.820 | 0.962 | 0.296 | 96 | 0.833 | 0.966 | 0.268 |
| 97 | 0.823 | 0.950 | 0.351 | 98 | 0.895 | 0.945 | 0.326 |
| 99 | 0.872 | 0.939 | 0.217 | 100 | 0.870 | 0.938 | 0.377 |
| 101 | 0.813 | 0.937 | 0.240 | 102 | 0.829 | 0.934 | 0.245 |
| 103 | 0.926 | 0.940 | 0.240 | 104 | 0.850 | 0.927 | 0.220 |

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 105 | 0.877 | 0.942 | 0.204 | 106 | 0.877 | 0.935 | 0.213 |
| 107 | 0.820 | 0.943 | 0.215 | 108 | 0.812 | 0.952 | 0.320 |
| 109 | 0.781 | 0.929 | 0.255 | 110 | 0.872 | 0.948 | 0.275 |
| 111 | 0.847 | 0.949 | 0.261 | 112 | 0.826 | 0.939 | 0.278 |
| 113 | 0.862 | 0.951 | 0.214 | 114 | 0.820 | 0.941 | 0.365 |
| 115 | 0.864 | 0.933 | 0.220 | 116 | 0.840 | 0.922 | 0.317 |
| 117 | 0.885 | 0.941 | 0.216 | 118 | 0.847 | 0.908 | 0.305 |
| 119 | 0.862 | 0.927 | 0.226 | 120 | 0.847 | 0.913 | 0.237 |
| 121 | 0.800 | 0.909 | 0.235 | 122 | 0.820 | 0.935 | 0.274 |
| 123 | 0.864 | 0.937 | 0.267 | 124 | 0.862 | 0.938 | 0.364 |
| 125 | 0.840 | 0.893 | 0.280 | 126 | 0.984 | 0.876 | 0.267 |
| 127 | 0.922 | 0.797 | 0.305 | 128 | 0.958 | 0.808 | 0.476 |
| 129 | 0.934 | 0.780 | 0.528 | 130 | 0.922 | 0.832 | 0.481 |

Tab. A.2 | Einzelne RQ-Ergebnisse von jedem Segmentierungsnetz

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 1 | 0.826 | 0.429 | 1.000 | 2 | 0.748 | 0.389 | 1.000 |
| 3 | 0.894 | 0.506 | 1.000 | 4 | 0.861 | 0.458 | 1.000 |
| 5 | 0.832 | 0.439 | 1.000 | 6 | 0.900 | 0.520 | 1.000 |
| 7 | 0.753 | 0.395 | 1.000 | 8 | 0.869 | 0.474 | 1.000 |
| 9 | 0.992 | 0.673 | 1.000 | 10 | 0.991 | 0.689 | 1.000 |
| 11 | 0.957 | 0.401 | 1.000 | 12 | 0.954 | 0.385 | 1.000 |
| 13 | 0.991 | 0.685 | 1.000 | 14 | 0.966 | 0.320 | 0.991 |
| 15 | 0.840 | 0.242 | 0.986 | 16 | 0.966 | 0.320 | 0.991 |
| 17 | 0.840 | 0.242 | 0.986 | 18 | 0.986 | 0.405 | 1.000 |
| 19 | 0.712 | 0.334 | 0.987 | 20 | 0.932 | 0.570 | 1.000 |
| 21 | 0.838 | 0.171 | 0.985 | 22 | 0.875 | 0.233 | 1.000 |
| 23 | 0.950 | 0.505 | 1.000 | 24 | 0.944 | 0.400 | 0.987 |
| 25 | 0.933 | 0.447 | 1.000 | 26 | 0.909 | 0.427 | 1.000 |
| 27 | 0.959 | 0.541 | 1.000 | 28 | 0.940 | 0.501 | 1.000 |
| 29 | 0.957 | 0.143 | 1.000 | 30 | 0.954 | 0.307 | 1.000 |
| 31 | 0.954 | 0.201 | 1.000 | 32 | 0.935 | 0.277 | 1.000 |
| 33 | 0.935 | 0.416 | 1.000 | 34 | 1.000 | 0.261 | 1.000 |
| 35 | 0.985 | 0.324 | 1.000 | 36 | 1.000 | 0.326 | 0.968 |
| 37 | 1.000 | 0.301 | 1.000 | 38 | 1.000 | 0.261 | 0.985 |
| 39 | 0.986 | 0.321 | 1.000 | 40 | 1.000 | 0.365 | 1.000 |
| 41 | 0.972 | 0.405 | 0.986 | 42 | 1.000 | 0.280 | 1.000 |
| 43 | 1.000 | 0.308 | 1.000 | 44 | 1.000 | 0.283 | 1.000 |
| 45 | 0.986 | 0.279 | 1.000 | 46 | 0.968 | 0.238 | 1.000 |

A Appendix

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 47 | 1.000 | 0.281 | 1.000 | 48 | 0.985 | 0.261 | 1.000 |
| 49 | 1.000 | 0.307 | 1.000 | 50 | 1.000 | 0.308 | 1.000 |
| 51 | 0.985 | 0.275 | 1.000 | 52 | 1.000 | 0.368 | 0.984 |
| 53 | 1.000 | 0.251 | 1.000 | 54 | 1.000 | 0.281 | 1.000 |
| 55 | 1.000 | 0.302 | 0.986 | 56 | 1.000 | 0.253 | 1.000 |
| 57 | 0.984 | 0.272 | 1.000 | 58 | 1.000 | 0.339 | 1.000 |
| 59 | 1.000 | 0.316 | 1.000 | 60 | 0.986 | 0.363 | 1.000 |
| 61 | 0.985 | 0.309 | 1.000 | 62 | 1.000 | 0.330 | 1.000 |
| 63 | 0.983 | 0.211 | 1.000 | 64 | 0.986 | 0.312 | 0.971 |
| 65 | 1.000 | 0.235 | 1.000 | 66 | 1.000 | 0.343 | 1.000 |
| 67 | 1.000 | 0.324 | 1.000 | 68 | 0.984 | 0.311 | 1.000 |
| 69 | 0.984 | 0.286 | 1.000 | 70 | 1.000 | 0.267 | 1.000 |
| 71 | 0.985 | 0.265 | 0.985 | 72 | 0.983 | 0.244 | 1.000 |
| 73 | 1.000 | 0.213 | 0.985 | 74 | 1.000 | 0.240 | 1.000 |
| 75 | 1.000 | 0.420 | 1.000 | 76 | 1.000 | 0.291 | 0.984 |
| 77 | 0.986 | 0.264 | 1.000 | 78 | 0.985 | 0.340 | 1.000 |
| 79 | 1.000 | 0.304 | 1.000 | 80 | 1.000 | 0.305 | 1.000 |
| 81 | 1.000 | 0.311 | 1.000 | 82 | 1.000 | 0.302 | 0.986 |
| 83 | 1.000 | 0.234 | 1.000 | 84 | 0.984 | 0.256 | 1.000 |
| 85 | 0.986 | 0.323 | 0.986 | 86 | 0.970 | 0.322 | 1.000 |
| 87 | 1.000 | 0.278 | 0.984 | 88 | 1.000 | 0.335 | 1.000 |
| 89 | 1.000 | 0.271 | 1.000 | 90 | 0.983 | 0.267 | 1.000 |
| 91 | 0.984 | 0.331 | 1.000 | 92 | 1.000 | 0.278 | 1.000 |
| 93 | 0.970 | 0.276 | 0.985 | 94 | 1.000 | 0.275 | 1.000 |
| 95 | 1.000 | 0.269 | 0.986 | 96 | 1.000 | 0.228 | 1.000 |
| 97 | 0.985 | 0.262 | 1.000 | 98 | 0.983 | 0.239 | 1.000 |
| 99 | 0.984 | 0.293 | 1.000 | 100 | 1.000 | 0.258 | 0.985 |
| 101 | 1.000 | 0.284 | 1.000 | 102 | 0.984 | 0.258 | 1.000 |
| 103 | 0.985 | 0.281 | 1.000 | 104 | 0.983 | 0.278 | 1.000 |
| 105 | 1.000 | 0.276 | 1.000 | 106 | 1.000 | 0.245 | 0.966 |
| 107 | 1.000 | 0.275 | 1.000 | 108 | 0.971 | 0.282 | 1.000 |
| 109 | 1.000 | 0.325 | 0.985 | 110 | 0.985 | 0.328 | 0.985 |
| 111 | 1.000 | 0.249 | 1.000 | 112 | 1.000 | 0.263 | 1.000 |
| 113 | 1.000 | 0.320 | 1.000 | 114 | 1.000 | 0.300 | 0.985 |
| 115 | 0.985 | 0.256 | 0.970 | 116 | 1.000 | 0.303 | 0.986 |
| 117 | 1.000 | 0.306 | 1.000 | 118 | 1.000 | 0.288 | 0.986 |
| 119 | 1.000 | 0.278 | 1.000 | 120 | 1.000 | 0.296 | 1.000 |
| 121 | 1.000 | 0.256 | 1.000 | 122 | 1.000 | 0.316 | 1.000 |
| 123 | 0.986 | 0.342 | 1.000 | 124 | 1.000 | 0.325 | 1.000 |
| 125 | 1.000 | 0.349 | 1.000 | 126 | 0.979 | 0.287 | 1.000 |
| 127 | 0.923 | 0.361 | 1.000 | 128 | 0.975 | 0.465 | 1.000 |

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 129 | 0.969 | 0.540 | 1.000 | 130 | 0.938 | 0.442 | 1.000 |

Tab. A.3 | Einzelne IQ-Ergebnisse von jedem Segmentierungsnetz

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 1 | 0.207 | 0.026 | 0.003 | 2 | 0.250 | 0.022 | 0.003 |
| 3 | 0.235 | 0.051 | 0.002 | 4 | 0.330 | 0.053 | 0.004 |
| 5 | 0.207 | 0.027 | 0.002 | 6 | 0.237 | 0.052 | 0.002 |
| 7 | 0.253 | 0.023 | 0.003 | 8 | 0.335 | 0.055 | 0.004 |
| 9 | 0.549 | 0.112 | 0.005 | 10 | 0.419 | 0.020 | 0.013 |
| 11 | 0.412 | 0.017 | 0.009 | 12 | 0.411 | 0.016 | 0.010 |
| 13 | 0.419 | 0.020 | 0.013 | 14 | 0.563 | 0.011 | 0.019 |
| 15 | 0.424 | 0.013 | 0.038 | 16 | 0.563 | 0.011 | 0.019 |
| 17 | 0.424 | 0.013 | 0.038 | 18 | 0.438 | 0.041 | 0.026 |
| 19 | 0.152 | 0.012 | 0.005 | 20 | 0.214 | 0.080 | 0.002 |
| 21 | 0.300 | 0.008 | 0.011 | 22 | 0.505 | 0.019 | 0.003 |
| 23 | 0.464 | 0.047 | 0.010 | 24 | 0.509 | 0.048 | 0.025 |
| 25 | 0.480 | 0.045 | 0.010 | 26 | 0.431 | 0.034 | 0.006 |
| 27 | 0.532 | 0.077 | 0.010 | 28 | 0.426 | 0.046 | 0.012 |
| 29 | 0.625 | 0.003 | 0.013 | 30 | 0.566 | 0.007 | 0.005 |
| 31 | 0.573 | 0.004 | 0.012 | 32 | 0.537 | 0.007 | 0.010 |
| 33 | 0.537 | 0.048 | 0.012 | 34 | 0.741 | 0.036 | 0.037 |
| 35 | 0.741 | 0.047 | 0.016 | 36 | 0.724 | 0.054 | 0.017 |
| 37 | 0.714 | 0.042 | 0.023 | 38 | 0.736 | 0.033 | 0.044 |
| 39 | 0.685 | 0.051 | 0.028 | 40 | 0.717 | 0.072 | 0.009 |
| 41 | 0.673 | 0.082 | 0.024 | 42 | 0.728 | 0.038 | 0.030 |
| 43 | 0.718 | 0.047 | 0.036 | 44 | 0.699 | 0.035 | 0.011 |
| 45 | 0.697 | 0.039 | 0.021 | 46 | 0.703 | 0.026 | 0.016 |
| 47 | 0.752 | 0.038 | 0.032 | 48 | 0.708 | 0.034 | 0.029 |
| 49 | 0.751 | 0.049 | 0.015 | 50 | 0.705 | 0.045 | 0.025 |
| 51 | 0.640 | 0.037 | 0.027 | 52 | 0.709 | 0.066 | 0.038 |
| 53 | 0.720 | 0.032 | 0.027 | 54 | 0.765 | 0.040 | 0.014 |
| 55 | 0.769 | 0.045 | 0.031 | 56 | 0.746 | 0.033 | 0.021 |
| 57 | 0.729 | 0.038 | 0.020 | 58 | 0.717 | 0.059 | 0.022 |
| 59 | 0.703 | 0.045 | 0.021 | 60 | 0.695 | 0.061 | 0.023 |
| 61 | 0.736 | 0.049 | 0.043 | 62 | 0.728 | 0.051 | 0.022 |
| 63 | 0.683 | 0.022 | 0.008 | 64 | 0.728 | 0.051 | 0.048 |
| 65 | 0.708 | 0.026 | 0.027 | 66 | 0.736 | 0.059 | 0.048 |
| 67 | 0.737 | 0.052 | 0.027 | 68 | 0.684 | 0.050 | 0.023 |
| 69 | 0.729 | 0.042 | 0.032 | 70 | 0.688 | 0.036 | 0.026 |

| Bild | CellposeSAM | nnUNet | Deepcell | Bild | CellposeSAM | nnUNet | Deepcell |
|------|-------------|--------|----------|------|-------------|--------|----------|
| 71 | 0.731 | 0.035 | 0.038 | 72 | 0.704 | 0.030 | 0.033 |
| 73 | 0.754 | 0.024 | 0.025 | 74 | 0.703 | 0.028 | 0.028 |
| 75 | 0.741 | 0.091 | 0.021 | 76 | 0.785 | 0.044 | 0.030 |
| 77 | 0.703 | 0.033 | 0.023 | 78 | 0.755 | 0.059 | 0.030 |
| 79 | 0.755 | 0.047 | 0.024 | 80 | 0.739 | 0.043 | 0.027 |
| 81 | 0.698 | 0.047 | 0.020 | 82 | 0.785 | 0.047 | 0.045 |
| 83 | 0.764 | 0.028 | 0.022 | 84 | 0.676 | 0.033 | 0.023 |
| 85 | 0.749 | 0.048 | 0.027 | 86 | 0.688 | 0.049 | 0.014 |
| 87 | 0.785 | 0.039 | 0.017 | 88 | 0.789 | 0.058 | 0.041 |
| 89 | 0.705 | 0.034 | 0.025 | 90 | 0.642 | 0.035 | 0.017 |
| 91 | 0.682 | 0.055 | 0.030 | 92 | 0.730 | 0.044 | 0.032 |
| 93 | 0.684 | 0.036 | 0.035 | 94 | 0.698 | 0.037 | 0.010 |
| 95 | 0.726 | 0.037 | 0.027 | 96 | 0.719 | 0.025 | 0.027 |
| 97 | 0.699 | 0.035 | 0.039 | 98 | 0.759 | 0.028 | 0.043 |
| 99 | 0.734 | 0.046 | 0.028 | 100 | 0.761 | 0.031 | 0.041 |
| 101 | 0.707 | 0.040 | 0.019 | 102 | 0.704 | 0.034 | 0.022 |
| 103 | 0.769 | 0.038 | 0.030 | 104 | 0.715 | 0.042 | 0.043 |
| 105 | 0.771 | 0.037 | 0.019 | 106 | 0.765 | 0.031 | 0.029 |
| 107 | 0.714 | 0.038 | 0.023 | 108 | 0.666 | 0.038 | 0.018 |
| 109 | 0.679 | 0.053 | 0.024 | 110 | 0.735 | 0.053 | 0.030 |
| 111 | 0.758 | 0.034 | 0.031 | 112 | 0.715 | 0.032 | 0.014 |
| 113 | 0.744 | 0.053 | 0.024 | 114 | 0.724 | 0.044 | 0.042 |
| 115 | 0.722 | 0.033 | 0.021 | 116 | 0.734 | 0.043 | 0.032 |
| 117 | 0.758 | 0.049 | 0.013 | 118 | 0.742 | 0.036 | 0.027 |
| 119 | 0.751 | 0.037 | 0.011 | 120 | 0.739 | 0.041 | 0.026 |
| 121 | 0.701 | 0.030 | 0.032 | 122 | 0.716 | 0.053 | 0.029 |
| 123 | 0.730 | 0.054 | 0.029 | 124 | 0.750 | 0.056 | 0.048 |
| 125 | 0.724 | 0.057 | 0.024 | 126 | 0.555 | 0.012 | 0.021 |
| 127 | 0.476 | 0.031 | 0.011 | 128 | 0.534 | 0.034 | 0.018 |
| 129 | 0.530 | 0.048 | 0.031 | 130 | 0.511 | 0.032 | 0.006 |

Tab. A.4 | Einzelne ipq-Ergebnisse von jedem Segmentierungsnetz

A.3 Einzelne Ergebnisse aller Klassifikator Kombinationen

| Encoder | Decoder | Vortraining | Mask Channel | Best accuracy |
|-----------------|-------------------------|------------------------------|-----------------|---------------|
| CellposeSAM | Schichten-Klassifikator | Fully-supervised | Masken-Methode | 72,9% |
| CellposeSAM | Volumen-Klassifikator | Fully-supervised | Masken-Methode | 84,9% |
| ResNet18 | Schichten-Klassifikator | Fully-supervised | Masken-Methode | 79,4% |
| ResNet18 | Volumen-Klassifikator | Fully-supervised | Masken-Methode | 83,6% |
| ResNet101 | Schichten-Klassifikator | Fully-supervised | Masken-Methode | 78,6% |
| ResNet101 | Volumen-Klassifikator | Fully-supervised | Masken-Methode | 81,8% |
| Efficientnet V2 | Schichten-Klassifikator | Fully-supervised | Masken-Methode | 81,8% |
| Efficientnet V2 | Volumen-Klassifikator | Fully-supervised | Masken-Methode | 83,1% |
| ConvNeXt | Schichten-Klassifikator | Fully-supervised | Masken-Methode | 83,6% |
| ConvNeXt | Volumen-Klassifikator | Fully-supervised | Masken-Methode | 82,6% |
| Swin V2 | Schichten-Klassifikator | Fully-supervised | Masken-Methode | 82,3% |
| Swin V2 | Volumen-Klassifikator | Fully-supervised | Masken-Methode | 83,9% |
| CellposeSAM | Schichten-Klassifikator | Fully-supervised | Distanz-Methode | 69,3% |
| CellposeSAM | Volumen-Klassifikator | Fully-supervised | Distanz-Methode | 82,6% |
| ResNet18 | Schichten-Klassifikator | Fully-supervised | Distanz-Methode | 78,4% |
| ResNet18 | Volumen-Klassifikator | Fully-supervised | Distanz-Methode | 81,2% |
| ResNet101 | Schichten-Klassifikator | Fully-supervised | Distanz-Methode | 77,1% |
| ResNet101 | Volumen-Klassifikator | Fully-supervised | Distanz-Methode | 81,0% |
| Efficientnet V2 | Schichten-Klassifikator | Fully-supervised | Distanz-Methode | 71,1% |
| Efficientnet V2 | Volumen-Klassifikator | Fully-supervised | Distanz-Methode | 70,8% |
| ConvNeXt | Schichten-Klassifikator | Fully-supervised | Distanz-Methode | 80,2% |
| ConvNeXt | Volumen-Klassifikator | Fully-supervised | Distanz-Methode | 79,9% |
| Swin V2 | Schichten-Klassifikator | Fully-supervised | Distanz-Methode | 76,6% |
| Swin V2 | Volumen-Klassifikator | Fully-supervised | Distanz-Methode | 77,9% |
| ResNet18 | Volumen-Klassifikator | Kein Vortraining | Masken-Methode | 85,9% |
| ResNet101 | Volumen-Klassifikator | Kein Vortraining | Masken-Methode | 81,8% |
| Efficientnet V2 | Volumen-Klassifikator | Kein Vortraining | Masken-Methode | 78,6% |
| ConvNeXt | Volumen-Klassifikator | Kein Vortraining | Masken-Methode | 75,3% |
| Swin V2 | Volumen-Klassifikator | Kein Vortraining | Masken-Methode | 50,0% |
| ResNet18 | Volumen-Klassifikator | Nur semi-supervised | Masken-Methode | 63,8% |
| ResNet101 | Volumen-Klassifikator | Nur semi-supervised | Masken-Methode | 61,6% |
| Efficientnet V2 | Volumen-Klassifikator | Nur semi-supervised | Masken-Methode | 57,8% |
| ConvNeXt | Volumen-Klassifikator | Nur semi-supervised | Masken-Methode | 42,6% |
| Swin V2 | Volumen-Klassifikator | Nur semi-supervised | Masken-Methode | 38,5% |
| ResNet18 | Volumen-Klassifikator | Semi-supervised und Transfer | Masken-Methode | 81,5% |
| ResNet101 | Volumen-Klassifikator | Semi-supervised und Transfer | Masken-Methode | 79,2% |
| Efficientnet V2 | Volumen-Klassifikator | Semi-supervised und Transfer | Masken-Methode | 63,0% |
| ConvNeXt | Volumen-Klassifikator | Semi-supervised und Transfer | Masken-Methode | 81,8% |
| Swin V2 | Volumen-Klassifikator | Semi-supervised und Transfer | Masken-Methode | 43,0% |

Tab. A.5 | Ergebnisse der einzelnen Methodenkombinationen der trainierten Klassifikatoren

References

- [1] Warren H Lewis und Margaret R Lewis. „Behavior of cross striated muscle in tissue cultures“. In: *American Journal of Anatomy* 22.2 (1917), S. 169–194.
- [2] Emeka Enwere et al. „Role of the TWEAK-Fn14-cIAP1-NF-κB Signaling Axis in the Regulation of Myogenesis and Muscle Homeostasis“. In: *Frontiers in Immunology* 5 (Feb. 2014), S. 34. doi: [10.3389/fimmu.2014.00034](https://doi.org/10.3389/fimmu.2014.00034).
- [3] Scott F Gilbert. *Developmental biology*. Englisch. 10. Aufl. Sunderland, Massachusetts: Sinauer Associates, 2014.
- [4] Irene A. Pogogeff und Margaret R. Murray. „Form and behavior of adult mammalian skeletal muscle in vitro“. en. In: *The Anatomical Record* 95.3 (1946), S. 321–335. ISSN: 1097-0185. doi: [10.1002/ar.1090950308](https://doi.org/10.1002/ar.1090950308). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ar.1090950308> (besucht am 16.07.2025).
- [5] Antoine Weisrock et al. „MyoFInDer: An AI-Based Tool for Myotube Fusion Index Determination“. In: *Tissue Engineering Part A* 30.19-20 (Okt. 2024), S. 652–661. ISSN: 1937-3341. doi: [10.1089/ten.tea.2024.0049](https://doi.org/10.1089/ten.tea.2024.0049). URL: <https://www.liebertpub.com/doi/10.1089/ten.tea.2024.0049> (besucht am 16.07.2025).
- [6] Benjamin Lair et al. *MyoFuse: A fully AI-based workflow for automated quantification of skeletal muscle cell fusion in vitro*. en. Techn. Ber. Type: article. bioRxiv, Feb. 2025. Kap. New Results, S. 2025.02.17.638596. doi: [10.1101/2025.02.17.638596](https://doi.org/10.1101/2025.02.17.638596). URL: <https://www.biorxiv.org/content/10.1101/2025.02.17.638596v1> (besucht am 16.07.2025).
- [7] Kyungchang Jeong et al. „SEPO-FI: Deep-learning based software to calculate fusion index of muscle cells“. In: *Computers in Biology and Medicine* 186 (März 2025), S. 109706. ISSN: 0010-4825. doi: [10.1016/j.combiomed.2025.109706](https://doi.org/10.1016/j.combiomed.2025.109706). URL: <https://www.sciencedirect.com/science/article/pii/S0010482525000563> (besucht am 16.07.2025).
- [8] Juergen Scharner und Peter S Zammit. „The muscle satellite cell at 50: the formative years“. In: *Skeletal Muscle* 1 (Aug. 2011), S. 28. ISSN: 2044-5040. doi: [10.1186/2044-5040-1-28](https://doi.org/10.1186/2044-5040-1-28). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3177780/> (besucht am 16.07.2025).

- [9] Pedro Veliça und Chris M. Bunce. „A quick, simple and unbiased method to quantify C2C12 myogenic differentiation“. eng. In: *Muscle & Nerve* 44.3 (Sep. 2011), S. 366–370. ISSN: 1097-4598. doi: [10.1002/mus.22056](https://doi.org/10.1002/mus.22056).
- [10] Andy Nolan et al. „Fluorescent characterization of differentiated myotubes using flow cytometry“. eng. In: *Cytometry. Part A: The Journal of the International Society for Analytical Cytology* 105.5 (Mai 2024), S. 332–344. ISSN: 1552-4930. doi: [10.1002/cyto.a.24822](https://doi.org/10.1002/cyto.a.24822).
- [11] Chibeza C. Agley et al. „An Image Analysis Method for the Precise Selection and Quantitation of Fluorescently Labeled Cellular Constituents“. In: *Journal of Histochemistry and Cytochemistry* 60.6 (Juni 2012), S. 428–438. ISSN: 0022-1554. doi: [10.1369/0022155412442897](https://doi.org/10.1369/0022155412442897). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3393072/> (besucht am 16.07.2025).
- [12] Min-Wen Jason Chua et al. „Assessment of different strategies for scalable production and proliferation of human myoblasts“. In: *Cell Proliferation* 52.3 (März 2019), e12602. ISSN: 0960-7722. doi: [10.1111/cpr.12602](https://doi.org/10.1111/cpr.12602). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6536385/> (besucht am 16.07.2025).
- [13] Aref Shahini et al. „Efficient and high yield isolation of myoblasts from skeletal muscle“. In: *Stem cell research* 30 (Juli 2018), S. 122–129. ISSN: 1873-5061. doi: [10.1016/j.scr.2018.05.017](https://doi.org/10.1016/j.scr.2018.05.017). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6090567/> (besucht am 16.07.2025).
- [14] Jessica Brunetti et al. „Nanopattern surface improves cultured human myotube maturation“. In: *Skeletal Muscle* 11.1 (Mai 2021), S. 12. ISSN: 2044-5040. doi: [10.1186/s13395-021-00268-3](https://doi.org/10.1186/s13395-021-00268-3). URL: <https://doi.org/10.1186/s13395-021-00268-3> (besucht am 16.07.2025).
- [15] Gisela Nogales-Gadea et al. „Expression of Glycogen Phosphorylase Isoforms in Cultured Muscle from Patients with McArdle’s Disease Carrying the p.R771PfsX33 PYGM Mutation“. en. In: *PLOS ONE* 5.10 (Okt. 2010), e13164. ISSN: 1932-6203. doi: [10.1371/journal.pone.0013164](https://doi.org/10.1371/journal.pone.0013164). URL: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0013164> (besucht am 16.07.2025).
- [16] Simon Noë et al. „The Myotube Analyzer: how to assess myogenic features in muscle stem cells“. en. In: *Skeletal Muscle* 12.1 (Juni 2022), S. 12. ISSN: 2044-5040. doi: [10.1186/s13395-022-00297-6](https://doi.org/10.1186/s13395-022-00297-6). URL: <https://doi.org/10.1186/s13395-022-00297-6> (besucht am 16.07.2025).
- [17] Ahmed M. Abdelmoez et al. „Comparative profiling of skeletal muscle models reveals heterogeneity of transcriptome and metabolism“. In: *American Journal of Physiology - Cell Physiology* 318.3 (März 2020), S. C615–C626. ISSN: 0363-6143. doi: [10.1152/ajpcell.00540.2019](https://doi.org/10.1152/ajpcell.00540.2019). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7099524/> (besucht am 16.07.2025).

- [18] Haruki Inoue et al. „Automatic Quantitative Segmentation of Myotubes Reveals Single-cell Dynamics of S6 Kinase Activation“. eng. In: *Cell Structure and Function* 43.2 (Aug. 2018), S. 153–169. ISSN: 1347-3700. doi: [10.1247/csf.18012](https://doi.org/10.1247/csf.18012).
- [19] Josh Moore et al. „OME-NGFF: a next-generation file format for expanding bioimaging data-access strategies“. en. In: *Nature Methods* 18.12 (Dez. 2021), S. 1496–1498. ISSN: 1548-7105. doi: [10.1038/s41592-021-01326-w](https://doi.org/10.1038/s41592-021-01326-w). URL: <https://www.nature.com/articles/s41592-021-01326-w> (besucht am 03.09.2025).
- [20] Mingjie Pan et al. „DiffuseIR: diffusion models for isotropic reconstruction of 3D microscopic images“. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2023, S. 323–332.
- [21] Xuesong Li et al. „Three-dimensional structured illumination microscopy with enhanced axial resolution“. In: *Nature Biotechnology* 41.9 (2023), S. 1307–1319. ISSN: 1087-0156. doi: [10.1038/s41587-022-01651-1](https://doi.org/10.1038/s41587-022-01651-1). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10497409/> (besucht am 03.09.2025).
- [22] Neda Bagheri et al. „The new era of quantitative cell imaging—challenges and opportunities“. In: *Molecular cell* 82.2 (Jan. 2022), S. 241–247. ISSN: 1097-2765. doi: [10.1016/j.molcel.2021.12.024](https://doi.org/10.1016/j.molcel.2021.12.024). URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10339817/> (besucht am 03.09.2025).
- [23] Ze Liu et al. „Swin transformer: Hierarchical vision transformer using shifted windows“. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, S. 10012–10022.
- [24] Jim James et al. „Segmentation of tomography datasets using 3D convolutional neural networks“. In: *Computational Materials Science* 216 (Jan. 2023), S. 111847. ISSN: 0927-0256. doi: [10.1016/j.commatsci.2022.111847](https://doi.org/10.1016/j.commatsci.2022.111847). URL: <https://www.sciencedirect.com/science/article/pii/S0927025622005584> (besucht am 03.09.2025).
- [25] Henry Chan et al. „Machine learning enabled autonomous microstructural characterization in 3D samples“. en. In: *npj Computational Materials* 6.1 (Jan. 2020), S. 1. ISSN: 2057-3960. doi: [10.1038/s41524-019-0267-z](https://doi.org/10.1038/s41524-019-0267-z). URL: <https://www.nature.com/articles/s41524-019-0267-z> (besucht am 03.09.2025).
- [26] Yu Hirabayashi et al. „Deep learning for three-dimensional segmentation of electron microscopy images of complex ceramic materials“. en. In: *npj Computational Materials* 10.1 (März 2024), S. 46. ISSN: 2057-3960. doi: [10.1038/s41524-024-01226-5](https://doi.org/10.1038/s41524-024-01226-5). URL: <https://www.nature.com/articles/s41524-024-01226-5> (besucht am 03.09.2025).

- [27] P. A. Midgley und M. Weyland. „3D electron microscopy in the physical sciences: the development of Z-contrast and EFTEM tomography“. In: *Ultramicroscopy*. Proceedings of the International Workshop on Strategies and Advances in Atomic Level Spectroscopy and Analysis 96.3 (Sep. 2003), S. 413–431. ISSN: 0304-3991. doi: [10.1016/S0304-3991\(03\)00105-0](https://doi.org/10.1016/S0304-3991(03)00105-0). URL: <https://www.sciencedirect.com/science/article/pii/S0304399103001050> (besucht am 03.09.2025).
- [28] Yaroslav Ganin und Victor Lempitsky. „Unsupervised domain adaptation by back-propagation“. In: *International conference on machine learning*. PMLR. 2015, S. 1180–1189.
- [29] Han Zhu et al. „Domain adaptation using class similarity for robust speech recognition“. In: *arXiv preprint arXiv:2011.02782* (2020).
- [30] Tahereh Koohi-Var und Morteza Zahedi. „Cross-domain graph based similarity measurement of workflows“. In: *Journal of Big Data* 5 (2018), S. 1–16.
- [31] Yuanzhe Cai et al. „Efficient algorithm for computing link-based similarity in real world networks“. In: *2009 Ninth IEEE International Conference on Data Mining*. IEEE. 2009, S. 734–739.
- [32] Wei Yuan, Jianfeng Gao und Hisami Suzuki. „An empirical study on language model adaptation using a metric of domain similarity“. In: *International Conference on Natural Language Processing*. Springer. 2005, S. 957–968.
- [33] Lothar Schermelleh et al. „Subdiffraction multicolor imaging of the nuclear periphery with 3D structured illumination microscopy“. eng. In: *Science (New York, N.Y.)* 320.5881 (Juni 2008), S. 1332–1336. ISSN: 1095-9203. doi: [10.1126/science.1156947](https://doi.org/10.1126/science.1156947).
- [34] Hong-Shang Peng und Daniel T. Chiu. „Soft fluorescent nanomaterials for biological and biomedical imaging“. en. In: *Chemical Society Reviews* 44.14 (2015), S. 4699–4722. doi: [10.1039/C4CS00294F](https://doi.org/10.1039/C4CS00294F). URL: <https://pubs.rsc.org/en/content/articlelanding/2015/cs/c4cs00294f> (besucht am 03.09.2025).
- [35] Rehan Ali et al. „Automatic segmentation of adherent biological cell boundaries and nuclei from brightfield microscopy images“. en. In: *Machine Vision and Applications* 23.4 (Juli 2012), S. 607–621. ISSN: 1432-1769. doi: [10.1007/s00138-011-0337-9](https://doi.org/10.1007/s00138-011-0337-9). URL: <https://doi.org/10.1007/s00138-011-0337-9> (besucht am 03.09.2025).
- [36] Mei Wang und Weihong Deng. „Deep visual domain adaptation: A survey“. In: *Neurocomputing* 312 (2018), S. 135–153.
- [37] Xingchao Peng et al. „Visda: The visual domain adaptation challenge“. In: *arXiv preprint arXiv:1710.06924* (2017).

- [38] Tianyu Han, Lifeng Zhang und Shixiang Jia. „Bin similarity-based domain adaptation for fine-grained image classification“. In: *International Journal of Intelligent Systems* 37.3 (2022), S. 2319–2334.
- [39] Pedro O Pinheiro. „Unsupervised domain adaptation with similarity learning“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, S. 8004–8013.
- [40] Yaroslav Ganin et al. „Domain-adversarial training of neural networks“. In: *Journal of machine learning research* 17.59 (2016), S. 1–35.
- [41] Soumyadeep Ghosh et al. „Domain Adaptation for Visual Understanding“. en. In: Hrsg. von Richa Singh et al. Cham: Springer International Publishing, 2020, S. 1–15. ISBN: 9783030306717. doi: 10.1007/978-3-030-30671-7_1. URL: https://doi.org/10.1007/978-3-030-30671-7_1 (besucht am 05.09.2025).
- [42] Bharath Hariharan et al. „Simultaneous Detection and Segmentation“. en. In: Hrsg. von David Fleet et al. Bd. 8695. Cham: Springer International Publishing, 2014, S. 297–312. ISBN: 9783319105833 9783319105840. doi: 10.1007/978-3-319-10584-0_20. URL: http://link.springer.com/10.1007/978-3-319-10584-0_20 (besucht am 05.08.2025).
- [43] John Winn und Jamie Shotton. „The layout consistent random field for recognizing and segmenting partially occluded objects“. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Bd. 1. IEEE, 2006, S. 37–44. URL: <https://ieeexplore.ieee.org/abstract/document/1640739/> (besucht am 05.08.2025).
- [44] Alexander Kirillov et al. „Panoptic segmentation“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, S. 9404–9413. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Kirillov_Panoptic_Segmentation_CVPR_2019_paper.html (besucht am 05.08.2025).
- [45] Shervin Minaee et al. „Image segmentation using deep learning: A survey“. In: *IEEE transactions on pattern analysis and machine intelligence* 44.7 (2021), S. 3523–3542. URL: <https://ieeexplore.ieee.org/abstract/document/9356353/> (besucht am 05.08.2025).
- [46] Anurag Arnab und Philip HS Torr. „Pixelwise instance segmentation with a dynamically instantiated network“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, S. 441–450. URL: http://openaccess.thecvf.com/content_cvpr_2017/html/Arnab_Pixelwise_Instance_Segmentation_CVPR_2017_paper.html (besucht am 05.08.2025).

- [47] Liang-Chieh Chen et al. „Masklab: Instance segmentation by refining object detection with semantic and direction features“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, S. 4013–4022. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Chen_MaskLab_Instance_Segmentation_CVPR_2018_paper.html (besucht am 05.08.2025).
- [48] Olaf Ronneberger, Philipp Fischer und Thomas Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation“. en. In: Hrsg. von Nassir Navab et al. Bd. 9351. Cham: Springer International Publishing, 2015, S. 234–241. ISBN: 9783319245737 9783319245744. doi: [10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28). URL: http://link.springer.com/10.1007/978-3-319-24574-4_28 (besucht am 05.08.2025).
- [49] Ross Girshick et al. „Rich feature hierarchies for accurate object detection and semantic segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, S. 580–587. URL: http://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html (besucht am 05.08.2025).
- [50] Jonathan Long, Evan Shelhamer und Trevor Darrell. „Fully convolutional networks for semantic segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 3431–3440. URL: http://openaccess.thecvf.com/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html (besucht am 05.08.2025).
- [51] Matthew D. Zeiler et al. „Deconvolutional networks“. In: ISSN: 1063-6919. Juni 2010, S. 2528–2535. doi: [10.1109/CVPR.2010.5539957](https://doi.org/10.1109/CVPR.2010.5539957). URL: <https://ieeexplore.ieee.org/document/5539957> (besucht am 27.08.2025).
- [52] Matthew D. Zeiler und Rob Fergus. „Visualizing and Understanding Convolutional Networks“. en. In: *Computer Vision – ECCV 2014*. Hrsg. von David Fleet et al. Cham: Springer International Publishing, 2014, S. 818–833. ISBN: 9783319105901. doi: [10.1007/978-3-319-10590-1_53](https://doi.org/10.1007/978-3-319-10590-1_53).
- [53] Hyeonwoo Noh, Seunghoon Hong und Bohyung Han. „Learning Deconvolution Network for Semantic Segmentation“. In: ISSN: 2380-7504. Dez. 2015, S. 1520–1528. doi: [10.1109/ICCV.2015.178](https://doi.org/10.1109/ICCV.2015.178). URL: <https://ieeexplore.ieee.org/document/7410535> (besucht am 27.08.2025).
- [54] Mohammadreza Mostajabi, Payman Yadollahpour und Gregory Shakhnarovich. „Feed-forward semantic segmentation with zoom-out features“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 3376–3385. URL: http://openaccess.thecvf.com/content_cvpr_2015/html/Mostajabi_Feedforward_Semantic_Segmentation_2015_CVPR_paper.html (besucht am 27.08.2025).

- [55] Jiuxiang Gu et al. „Recent advances in convolutional neural networks“. In: *Pattern recognition* 77 (2018), S. 354–377. URL: <https://www.sciencedirect.com/science/article/pii/S0031320317304120> (besucht am 27.08.2025).
- [56] Keinosuke Fukunaga. „Statistical pattern recognition“. In: WORLD SCIENTIFIC, Aug. 1993, S. 33–60. ISBN: 9789810211363. doi: [10.1142/9789814343138_0002](https://doi.org/10.1142/9789814343138_0002). URL: https://www.worldscientific.com/doi/abs/10.1142/9789814343138_0002 (besucht am 31.07.2025).
- [57] S.R. Kulkarni, G. Lugosi und S.S. Venkatesh. „Learning pattern classification-a survey“. In: *IEEE Transactions on Information Theory* 44.6 (Okt. 1998), S. 2178–2206. ISSN: 1557-9654. doi: [10.1109/18.720536](https://doi.org/10.1109/18.720536). URL: <https://ieeexplore.ieee.org/abstract/document/720536> (besucht am 31.07.2025).
- [58] A. K. Jain, M. N. Murty und P. J. Flynn. „Data clustering: a review“. In: *ACM Comput. Surv.* 31.3 (Sep. 1999), S. 264–323. ISSN: 0360-0300. doi: [10.1145/331499.331504](https://doi.org/10.1145/331499.331504). URL: <https://doi.acm.org/doi/10.1145/331499.331504> (besucht am 28.08.2025).
- [59] Shai Shalev-Shwartz und Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. en. Google-Books-ID: Hf6QAwAAQBAJ. Cambridge University Press, Mai 2014. ISBN: 9781139952743.
- [60] Marco Loog. „Supervised classification: Quite a brief overview“. In: *Machine Learning Techniques for Space Weather* (2018), S. 113–145. URL: <https://www.sciencedirect.com/science/article/pii/B9780128117880000056> (besucht am 31.07.2025).
- [61] Isabelle Guyon und André Elisseeff. „An Introduction to Feature Extraction“. en. In: Hrsg. von Isabelle Guyon et al. Berlin, Heidelberg: Springer, 2006, S. 1–25. ISBN: 9783540354888. doi: [10.1007/978-3-540-35488-8_1](https://doi.org/10.1007/978-3-540-35488-8_1). URL: https://doi.org/10.1007/978-3-540-35488-8_1 (besucht am 31.07.2025).
- [62] M. Kunaver und J.F. Tasic. „Image feature extraction - an overview“. In: Bd. 1. Nov. 2005, S. 183–186. doi: [10.1109/EURCON.2005.1629889](https://doi.org/10.1109/EURCON.2005.1629889). URL: <https://ieeexplore.ieee.org/abstract/document/1629889> (besucht am 04.08.2025).
- [63] Wamidh K. Mutlag et al. „Feature Extraction Methods: A Review“. en. In: *Journal of Physics: Conference Series* 1591.1 (Juli 2020), S. 012028. ISSN: 1742-6596. doi: [10.1088/1742-6596/1591/1/012028](https://doi.org/10.1088/1742-6596/1591/1/012028). URL: <https://dx.doi.org/10.1088/1742-6596/1591/1/012028> (besucht am 31.07.2025).
- [64] Charles T. Zahn und Ralph Z. Roskies. „Fourier Descriptors for Plane Closed Curves“. In: *IEEE Transactions on Computers* C-21.3 (März 1972), S. 269–281. ISSN: 1557-9956. doi: [10.1109/TC.1972.5008949](https://doi.org/10.1109/TC.1972.5008949). URL: <https://ieeexplore.ieee.org/abstract/document/5008949> (besucht am 28.08.2025).

- [65] Frank P Kuhl und Charles R Giardina. „Elliptic Fourier features of a closed contour“. In: *Computer Graphics and Image Processing* 18.3 (März 1982), S. 236–258. ISSN: 0146-664X. DOI: [10.1016/0146-664X\(82\)90034-X](https://doi.org/10.1016/0146-664X(82)90034-X). URL: <https://www.sciencedirect.com/science/article/pii/0146664X8290034X> (besucht am 28.08.2025).
- [66] Karl Pearson. „LIII. On lines and planes of closest fit to systems of points in space“. In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), S. 559–572.
- [67] H. Hotelling. „Analysis of a complex of statistical variables into principal components“. In: *Journal of Educational Psychology* 24.6 (1933), S. 417–441. ISSN: 1939-2176. DOI: [10.1037/h0071325](https://doi.org/10.1037/h0071325).
- [68] Rui Xu und D. Wunsch. „Survey of clustering algorithms“. In: *IEEE Transactions on Neural Networks* 16.3 (Mai 2005), S. 645–678. ISSN: 1941-0093. DOI: [10.1109/TNN.2005.845141](https://doi.org/10.1109/TNN.2005.845141). URL: <https://ieeexplore.ieee.org/abstract/document/1427769> (besucht am 28.08.2025).
- [69] Bernhard E. Boser, Isabelle M. Guyon und Vladimir N. Vapnik. „A training algorithm for optimal margin classifiers“. In: *Proceedings of the fifth annual workshop on Computational learning theory*. COLT '92. New York, NY, USA: Association for Computing Machinery, Juli 1992, S. 144–152. ISBN: 9780897914970. DOI: [10.1145/130385.130401](https://doi.org/10.1145/130385.130401). URL: <https://dl.acm.org/doi/10.1145/130385.130401> (besucht am 28.08.2025).
- [70] David Yarowsky. „Unsupervised word sense disambiguation rivaling supervised methods“. In: *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. ACL '95. USA: Association for Computational Linguistics, Juni 1995, S. 189–196. DOI: [10.3115/981658.981684](https://doi.org/10.3115/981658.981684). URL: <https://dl.acm.org/doi/10.3115/981658.981684> (besucht am 28.08.2025).
- [71] Bernhard Schölkopf. „Support vector learning“. PhD Thesis. Oldenbourg München, Germany, 1997. URL: https://pure.mpg.de/rest/items/item_1794215/component/file_3214422/content (besucht am 28.08.2025).
- [72] Alexander J. Smola und Bernhard Schölkopf. *Learning with kernels*. Bd. 4. Citeseer, 1998. URL: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1655f14b601fe6ff80ca935091e3cd78c9733d86> (besucht am 28.08.2025).
- [73] Dengyong Zhou et al. „Learning with Local and Global Consistency“. In: *Advances in Neural Information Processing Systems*. Bd. 16. MIT Press, 2003. URL: <https://proceedings.neurips.cc/paper/2003/hash/87682805257e619d49b8e0dAbstract.html> (besucht am 28.08.2025).
- [74] David Lowe und D. Broomhead. „Multivariable functional interpolation and adaptive networks“. In: *Complex systems* 2.3 (1988), S. 321–355. URL: <http://wpmedia.wolfram.com/uploads/sites/13/2018/02/02-3-5.pdf> (besucht am 28.08.2025).

- [75] Olivier Delalleau, Yoshua Bengio und Nicolas Le Roux. „Efficient non-parametric function induction in semi-supervised learning“. In: *International Workshop on Artificial Intelligence and Statistics*. PMLR, 2005, S. 96–103. URL: <http://proceedings.mlr.press/r5/delalleau05a/delalleau05a.pdf> (besucht am 28.08.2025).
- [76] Sotiris B. Kotsiantis, Ioannis Zaharakis und P. Pintelas. „Supervised machine learning: A review of classification techniques“. In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), S. 3–24. URL: [https://books.google.com/books?hl=de&lr=&id=vLiTXDHR_sYC&oi=fnd&pg=PA3&dq=Kotsiantis+\(2007\)+Supervised+machine+learning:+A+review+of+classification+techniques&ots=C_pvsyXHon&sig=cptgUhNhe2RQAWGK255Kwyti0R4](https://books.google.com/books?hl=de&lr=&id=vLiTXDHR_sYC&oi=fnd&pg=PA3&dq=Kotsiantis+(2007)+Supervised+machine+learning:+A+review+of+classification+techniques&ots=C_pvsyXHon&sig=cptgUhNhe2RQAWGK255Kwyti0R4) (besucht am 31.07.2025).
- [77] G. Hughes. „On the mean accuracy of statistical pattern recognizers“. In: *IEEE Transactions on Information Theory* 14.1 (Jan. 1968), S. 55–63. ISSN: 1557-9654. doi: [10.1109/TIT.1968.1054102](https://doi.org/10.1109/TIT.1968.1054102). URL: <https://ieeexplore.ieee.org/abstract/document/1054102> (besucht am 29.08.2025).
- [78] Asifullah Khan et al. „A survey of the vision transformers and their CNN-transformer based variants“. en. In: *Artificial Intelligence Review* 56.3 (Dez. 2023), S. 2917–2970. ISSN: 1573-7462. doi: [10.1007/s10462-023-10595-0](https://doi.org/10.1007/s10462-023-10595-0). URL: <https://doi.org/10.1007/s10462-023-10595-0> (besucht am 04.08.2025).
- [79] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems*. Bd. 25. Curran Associates, Inc., 2012. URL: <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e92Abstract.html> (besucht am 04.08.2025).
- [80] Max Bain et al. „Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval“. en. In: 2021, S. 1728–1738. URL: https://openaccess.thecvf.com/content/ICCV2021/html/Bain_Frozen_in_Time_A_Joint_Video_and_Image_Encoder_for_ICCV_2021_paper.html (besucht am 04.08.2025).
- [81] Jo Plested und Tom Gedeon. „Deep transfer learning for image classification: a survey“. In: *arXiv preprint arXiv:2205.09904* (Mai 2022). arXiv:2205.09904 [cs]. doi: [10.48550/arXiv.2205.09904](https://doi.org/10.48550/arXiv.2205.09904). URL: [http://arxiv.org/abs/2205.09904](https://arxiv.org/abs/2205.09904) (besucht am 04.08.2025).
- [82] Alireza Ghods und Diane J Cook. „A Survey of Techniques All Classifiers Can Learn from Deep Networks: Models, Optimizations, and Regularization“. In: *arXiv preprint arXiv:1909.04791* (Sep. 2019). arXiv:1909.04791 [cs]. doi: [10.48550/arXiv.1909.04791](https://doi.org/10.48550/arXiv.1909.04791). URL: [http://arxiv.org/abs/1909.04791](https://arxiv.org/abs/1909.04791) (besucht am 04.08.2025).

- [83] Jürgen Schmidhuber. „Deep learning in neural networks: An overview“. In: *Neural Networks* 61 (Jan. 2015), S. 85–117. ISSN: 0893-6080. doi: [10.1016/j.neunet.2014.09.003](https://doi.org/10.1016/j.neunet.2014.09.003). URL: <https://www.sciencedirect.com/science/article/pii/S0893608014002135> (besucht am 04.08.2025).
- [84] Sainbayar Sukhbaatar et al. „Training convolutional networks with noisy labels“. In: *arXiv preprint arXiv:1406.2080* (2014).
- [85] Elliott Gordon-Rodriguez et al. „Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning“. en. In: PMLR, Feb. 2020, S. 1–10. URL: <https://proceedings.mlr.press/v137/gordon-rodriguez20a.html> (besucht am 17.09.2025).
- [86] Anqi Mao, Mehryar Mohri und Yutao Zhong. „Cross-Entropy Loss Functions: Theoretical Analysis and Applications“. en. In: PMLR, Juli 2023, S. 23803–23828. URL: <https://proceedings.mlr.press/v202/mao23b.html> (besucht am 17.09.2025).
- [87] Jacob Goldberger und Ehud Ben-Reuven. „Training deep neural-networks using a noise adaptation layer“. In: *International conference on learning representations*. 2017. URL: <https://openreview.net/forum?id=H12GRgcxg> (besucht am 17.09.2025).
- [88] Bo Han et al. „Masking: A New Perspective of Noisy Supervision“. In: *Advances in Neural Information Processing Systems*. Bd. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/hash/aee92f16efd522b9326c25cc3237ac15-Abstract.html (besucht am 17.09.2025).
- [89] Dan Hendrycks et al. „Using Trusted Data to Train Deep Networks on Labels Corrupted by Severe Noise“. In: *Advances in Neural Information Processing Systems*. Bd. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/hash/ad554d8c3b06d6b97ee76a2448bd791Abstract.html (besucht am 17.09.2025).
- [90] Zhilu Zhang und Mert Sabuncu. „Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels“. In: *Advances in Neural Information Processing Systems*. Bd. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/f2925f97bc13ad2852a7a551802feea0-Abstract.html> (besucht am 17.09.2025).
- [91] P. R. Smith. „Bilinear interpolation of digital images“. In: *Ultramicroscopy* 6.2 (Jan. 1981), S. 201–204. ISSN: 0304-3991. doi: [10.1016/0304-3991\(81\)90061-9](https://doi.org/10.1016/0304-3991(81)90061-9). URL: <https://www.sciencedirect.com/science/article/pii/0304399181900619> (besucht am 05.09.2025).
- [92] Sergey Ioffe und Christian Szegedy. „Batch normalization: Accelerating deep network training by reducing internal covariate shift“. In: *International conference on machine learning*. pmlr, 2015, S. 448–456. URL: <http://proceedings.mlr.press/v37/ioffe15.html> (besucht am 05.09.2025).

- [93] Jimmy Lei Ba, Jamie Ryan Kiros und Geoffrey E Hinton. „Layer normalization“. In: *arXiv preprint arXiv:1607.06450* (2016).
- [94] Shibani Santurkar et al. „How Does Batch Normalization Help Optimization?“ In: *Advances in Neural Information Processing Systems*. Bd. 31. Curran Associates, Inc., 2018. URL: <https://proceedings.neurips.cc/paper/2018/hash/905056c1ac1dad141560467e0a99e1cf-Abstract.html> (besucht am 05.09.2025).
- [95] Jingjing Xu et al. „Understanding and Improving Layer Normalization“. In: *Advances in Neural Information Processing Systems*. Bd. 32. Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/2f4fe03d77724a7217006e5d16728874-Abstract.html> (besucht am 05.09.2025).
- [96] Ramprasaath R. Selvaraju et al. „Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Okt. 2017.
- [97] Christoffer Edlund et al. „LIVECell—A large-scale dataset for label-free live cell segmentation“. In: *Nature methods* 18.9 (2021), S. 1038–1045.
- [98] Nicola Dietler et al. „A convolutional neural network segments yeast microscopy images with high accuracy“. In: *Nature communications* 11.1 (2020), S. 5723.
- [99] S. Holden und M. Conduit. *DeepBacs – Bacillus subtilis fluorescence segmentation dataset*. Data set. 2021. doi: [10.5281/zenodo.5550968](https://doi.org/10.5281/zenodo.5550968). URL: <https://doi.org/10.5281/zenodo.5550968>.
- [100] Christoph Spahn et al. „DeepBacs: Bacterial image analysis using open-source deep learning approaches“. In: *bioRxiv* (2021). doi: [10.1101/2021.11.03.467152](https://doi.org/10.1101/2021.11.03.467152). URL: <https://www.biorxiv.org/content/early/2021/11/03/2021.11.03.467152>.
- [101] Vladimír Ulman et al. „An objective comparison of cell-tracking algorithms“. In: *Nature methods* 14.12 (2017), S. 1141–1152.
- [102] Neeraj Kumar et al. „A dataset and a technique for generalized nuclear segmentation for computational pathology“. In: *IEEE transactions on medical imaging* 36.7 (2017), S. 1550–1560.
- [103] Noah F Greenwald et al. „Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning“. In: *Nature biotechnology* 40.4 (2022), S. 555–565.
- [104] Florian Kromp et al. „An annotated fluorescence image dataset for training nuclear segmentation methods“. In: *Nature Scientific Data* 7.262 (2020), S. 1–8. doi: [10.1038/s41597-020-00608-w](https://doi.org/10.1038/s41597-020-00608-w).
- [105] Alain Chen et al. „3d ground truth annotations of nuclei in 3d microscopy volumes“. In: *bioRxiv* (2022), S. 2022–09.

- [106] Chichen Fu et al. „Three dimensional fluorescence microscopy image synthesis and segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, S. 2221–2229.
- [107] Nazmiye Ceren Abay et al. „Privacy Preserving Synthetic Data Release Using Deep Learning“. en. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Hrsg. von Michele Berlingario et al. Bd. 11051. Cham: Springer International Publishing, 2019, S. 510–526. ISBN: 9783030109240 9783030109257. doi: [10.1007/978-3-030-10925-7_31](https://doi.org/10.1007/978-3-030-10925-7_31). url: https://link.springer.com/10.1007/978-3-030-10925-7_31 (besucht am 27.08.2025).
- [108] Trivellore E. Raghunathan. „Synthetic Data“. en. In: *Annual Review of Statistics and Its Application* 8.1 (März 2021), S. 129–140. ISSN: 2326-8298, 2326-831X. doi: [10.1146/annurev-statistics-040720-031848](https://doi.org/10.1146/annurev-statistics-040720-031848). url: <https://www.annualreviews.org/doi/10.1146/annurev-statistics-040720-031848> (besucht am 27.08.2025).
- [109] Sergey I Nikolenko et al. *Synthetic data for deep learning*. Bd. 174. Springer, 2021.
- [110] Edward Choi et al. „Generating multi-label discrete patient records using generative adversarial networks“. In: *Machine learning for healthcare conference*. PMLR, 2017, S. 286–305. url: <http://proceedings.mlr.press/v68/choi17a> (besucht am 27.08.2025).
- [111] Yingzhou Lu et al. „Machine learning for synthetic data generation: a review“. In: *arXiv preprint arXiv:2302.04062* (2023).
- [112] Roman Bruch et al. „Improving 3D deep learning segmentation with biophysically motivated cell synthesis“. In: *Communications Biology* 8.1 (2025), S. 43.
- [113] Rishi Bommasani et al. „On the opportunities and risks of foundation models“. In: *arXiv preprint arXiv:2108.07258* (2021).
- [114] Jason Yosinski et al. „How transferable are features in deep neural networks?“ In: *Advances in neural information processing systems* 27 (2014). url: [https://proceedings.neurips.cc/paper_files/paper/2014/file/375c71349b295fbe2dcda9206f20a06 - Paper . pdf](https://proceedings.neurips.cc/paper_files/paper/2014/file/375c71349b295fbe2dcda9206f20a06-Paper.pdf) (besucht am 05.08.2025).
- [115] Jonas Dippel et al. „Transfer Learning for Segmentation Problems: Choose the Right Encoder and Skip the Decoder“. In: *arXiv preprint arXiv:2207.14508* (2022).
- [116] Huiyu Wang et al. „Max-deeplab: End-to-end panoptic segmentation with mask transformers“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, S. 5463–5474.
- [117] Xueyan Zou et al. „Segment everything everywhere all at once“. In: *Advances in neural information processing systems* 36 (2023), S. 19769–19782.

- [118] Jitesh Jain et al. „Oneformer: One transformer to rule universal image segmentation“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, S. 2989–2998.
- [119] Feng Li et al. „Mask dino: Towards a unified transformer-based framework for object detection and segmentation“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, S. 3041–3050.
- [120] Alexander Kirillov et al. „Segment anything“. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, S. 4015–4026.
- [121] Alexey Dosovitskiy et al. „An image is worth 16x16 words: Transformers for image recognition at scale“. In: *arXiv preprint arXiv:2010.11929* (2020).
- [122] Kaiming He et al. „Masked autoencoders are scalable vision learners“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, S. 16000–16009.
- [123] Alec Radford et al. „Learning Transferable Visual Models From Natural Language Supervision“. en. In: PMLR, Juli 2021, S. 8748–8763. URL: <https://proceedings.mlr.press/v139/radford21a.html> (besucht am 10.07.2025).
- [124] Anwai Archit et al. „Segment anything for microscopy“. In: *Nature Methods* (2025), S. 1–13.
- [125] Uriah Israel et al. „A foundation model for cell segmentation“. In: *arXiv preprint arXiv:2311.11004* (2023).
- [126] Alexandra D VandeLoo et al. „SAMCell: Generalized Label-Free Biological Cell Segmentation with Segment Anything“. In: *bioRxiv* (2025).
- [127] Carsen Stringer et al. „Cellpose: a generalist algorithm for cellular segmentation“. In: *Nature methods* 18.1 (2021), S. 100–106.
- [128] Marius Pachitariu, Michael Rariden und Carsen Stringer. „Cellpose-SAM: superhuman generalization for cellular segmentation“. In: *bioRxiv* (2025), S. 2025–04.
- [129] David A Van Valen et al. „Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments“. In: *PLoS computational biology* 12.11 (2016), e1005177.
- [130] Dylan Bannon et al. „DeepCell Kiosk: scaling deep learning–enabled cellular image analysis with Kubernetes“. In: *Nature methods* 18.1 (2021), S. 43–45.
- [131] Noah F Greenwald et al. „Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning“. In: *Nature biotechnology* 40.4 (2022), S. 555–565.
- [132] Erick Moen et al. „Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning“. In: *Biorxiv* (2019), S. 803205.

- [133] Mingxing Tan und Quoc Le. „Efficientnetv2: Smaller models and faster training“. In: *International conference on machine learning*. PMLR, 2021, S. 10096–10106. URL: <http://proceedings.mlr.press/v139/tan21a.html> (besucht am 05.08.2025).
- [134] Fabian Isensee et al. „nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation“. In: *Nature methods* 18.2 (2021), S. 203–211.
- [135] Jonas Dippel et al. „Transfer Learning for Segmentation Problems: Choose the Right Encoder and Skip the Decoder“. In: *arXiv preprint arXiv:2207.14508* (2022).
- [136] Olga Russakovsky et al. „ImageNet Large Scale Visual Recognition Challenge“. en. In: *International Journal of Computer Vision* 115.3 (Dez. 2015), S. 211–252. ISSN: 0920-5691, 1573-1405. doi: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y). URL: <http://link.springer.com/10.1007/s11263-015-0816-y> (besucht am 18.08.2025).
- [137] Yang You et al. „ImageNet Training in Minutes“. en. In: *Proceedings of the 47th International Conference on Parallel Processing*. Eugene OR USA: ACM, Aug. 2018, S. 1–10. ISBN: 9781450365109. doi: [10.1145/3225058.3225069](https://doi.org/10.1145/3225058.3225069). URL: <https://d1.acm.org/doi/10.1145/3225058.3225069> (besucht am 18.08.2025).
- [138] Simon Kornblith, Jonathon Shlens und Quoc V. Le. „Do Better ImageNet Models Transfer Better?“ In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Juni 2019, S. 2661–2671. URL: https://openaccess.thecvf.com/content_CVPR_2019/html/Kornblith_Do_Better_ImageNet_Models_Transfer_Better_CVPR_2019_paper.html (besucht am 18.08.2025).
- [139] Lucas Beyer et al. „Are we done with ImageNet?“ In: *arXiv preprint arXiv:2006.07159* (Juni 2020). arXiv:2006.07159 [cs]. doi: [10.48550/arXiv.2006.07159](https://doi.org/10.48550/arXiv.2006.07159). URL: <http://arxiv.org/abs/2006.07159> (besucht am 18.08.2025).
- [140] Benjamin Recht et al. „Do imagenet classifiers generalize to imagenet?“ In: *International conference on machine learning*. PMLR, 2019, S. 5389–5400. URL: <http://proceedings.mlr.press/v97/recht19a.html> (besucht am 18.08.2025).
- [141] Kaiming He et al. „Deep residual learning for image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 770–778. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (besucht am 11.08.2025).
- [142] Kaiming He und Jian Sun. „Convolutional neural networks at constrained time cost“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, S. 5353–5360. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/He_Convolutional_Neural_Networks_2015_CVPR_paper.html (besucht am 18.08.2025).

- [143] Rupesh Kumar Srivastava, Klaus Greff und Jürgen Schmidhuber. „Highway networks“. In: *arXiv preprint arXiv:1505.00387* (2015).
- [144] Sergey Ioffe. „Batch renormalization: Towards reducing minibatch dependence in batch-normalized models“. In: *Advances in neural information processing systems* 30 (2017). URL: <https://proceedings.neurips.cc/paper/2017/hash/c54e7837e0cd0ced286cb5995327d1ab-Abstract.html> (besucht am 18.08.2025).
- [145] Vinod Nair und Geoffrey E Hinton. „Rectified linear units improve restricted boltzmann machines“. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, S. 807–814.
- [146] Mingxing Tan und Quoc Le. „Efficientnet: Rethinking model scaling for convolutional neural networks“. In: *International conference on machine learning*. PMLR, 2019, S. 6105–6114. URL: <https://proceedings.mlr.press/v97/tan19a.html?ref=ji> (besucht am 12.08.2025).
- [147] Mark Sandler et al. „Mobilenetv2: Inverted residuals and linear bottlenecks“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, S. 4510–4520. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html (besucht am 18.08.2025).
- [148] Suyog Gupta und Mingxing Tan. „EfficientNet-EdgeTPU: Creating accelerator-optimized neural networks with AutoML“. In: *Google AI Blog* 2.1 (2019).
- [149] Zhuang Liu et al. „A convnet for the 2020s“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, S. 11976–11986. URL: http://openaccess.thecvf.com/content/CVPR2022/html/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.html (besucht am 11.08.2025).
- [150] Dan Hendrycks und Kevin Gimpel. „Gaussian error linear units (gelus)“. In: *arXiv preprint arXiv:1606.08415* (2016).
- [151] Ze Liu et al. „Swin transformer v2: Scaling up capacity and resolution“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, S. 12009–12019. URL: http://openaccess.thecvf.com/content/CVPR2022/html/Liu_Swin_Transformer_V2_Scaling_Up_Capacity_and_Resolution_CVPR_2022_paper.html (besucht am 11.08.2025).
- [152] John Bridle. „Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters“. In: *Advances in neural information processing systems* 2 (1989). URL: <https://proceedings.neurips.cc/paper/1989/hash/0336dcbab05b9d5ad24f4333c7658a0e-Abstract.html> (besucht am 19.08.2025).

References

- [153] Diederik Kinga, Jimmy Ba Adam et al. „A method for stochastic optimization“. In: *International conference on learning representations (ICLR)*. Bd. 5. 6. California; 2015.

Erklärung

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

Karlsruhe, den 30. Oktober 2025

.....