

**EBERHARD-KARLS-UNIVERSITÄT TÜBINGEN**  
Wilhelm-Schickard-Institut für Informatik  
Lehrstuhl Kognitive Systeme

**Bachelorarbeit Informatik**

**Einsatz von Deep Learning zur  
Erkennung von Messspitzen in  
Rasterelektronenmikroskopen**

David Stefan Kleindiek

**Betreuer:** Prof. Dr. rer. nat. Andreas Zell  
Wilhelm-Schickard-Institut für Informatik

Dr. Matthias Kemmler  
Kleindiek Nanotechnik GmbH

**Begonnen am:** 1. Juli 2023

**Beendet am:** 21. August 2023

## **Erklärung**

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Tübingen am 21. August 2023

---

David Stefan Kleindiek

**Kurzfassung.** Die Halbleiterindustrie, eine sich ständig weiterentwickelnde und komplexe Branche, die sich auf Strukturen unter 10 Nanometern reduziert hat, steht vor ständigen Herausforderungen bei der Fehleranalyse der entwickelten Strukturen. Insbesondere die zuverlässige Lokalisierung von Messspitzen im Nanoprobing ist entscheidend, um die Effizienz durch Automatisierung zu steigern und gleichzeitig Beschädigungen zu vermeiden. Diese Bachelorarbeit beschäftigt sich mit der Entwicklung und Implementierung von Deep Learning-Verfahren, insbesondere des Mask R-CNN Modells, zur Detektion von Messspitzen in rasterelektronenmikroskopischen Bildern.

Durch die Entwicklung und Implementierung von Schnittstellen zu Kleindiek Manipulatoren und ZEISS Mikroskopen in Python wird die Datenerfassung automatisiert und ein umfangreicher Datensatz für Nanoprobing erzeugt. Das Modell wird angepasst und trainiert, um die spezifischen Herausforderungen des Bildrauschens und der Formenvielfalt der Messspitzen zu überwinden.

Die Arbeit gibt einen detaillierten Einblick in den Nanoprobing-Prozess, und beinhaltet die umfassende Implementierung, das Training und die Evaluierung des Mask R-CNN Modells. Die Ergebnisse dieser Arbeit deuten darauf hin, dass Deep Learning eine vielversprechende Lösung für die Prozessautomatisierung in der Fehleranalyse in der Halbleiterindustrie ist und bieten eine solide Grundlage für die weitere Automatisierung und Verbesserung der Fehleranalyse, die Modelle zu optimieren und die Technologie in bestehende Prozesse zu integrieren.

# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problemstellung . . . . .	1
1.3 Aufbau der Arbeit . . . . .	2
<b>2 Grundlagen Nanoprobing</b>	<b>3</b>
2.1 Nanoprobing . . . . .	3
2.1.1 Prober Shuttle . . . . .	3
2.1.2 Messspitzen . . . . .	4
2.2 Rasterelektronenmikroskop . . . . .	5
2.3 Kleindiek Nanotechnik: Arbeitsplatz . . . . .	7
<b>3 Grundlagen neuronaler Netze</b>	<b>8</b>
3.1 Maschinelles Sehen . . . . .	8
3.1.1 Objektdetektion . . . . .	8
3.1.2 Instanz Segmentierung . . . . .	8
3.1.3 Keypoint Erkennung . . . . .	9
3.2 Künstliche neuronale Netze . . . . .	10
3.2.1 Mehrschicht-Perzeptron . . . . .	10
3.2.2 Aktivierungsfunktion . . . . .	10
3.3 Convolutional Neural Networks . . . . .	12
3.3.1 Convolution-Schichten . . . . .	12
3.3.2 Pooling-Schichten . . . . .	14
3.4 Mask R-CNN . . . . .	15
3.4.1 Backbone Netzwerk . . . . .	15
3.4.2 Region Proposal Netzwerk . . . . .	16
3.4.3 Vortraining . . . . .	17
3.4.4 Hyperparameter . . . . .	17
3.4.5 Trainingsablauf . . . . .	17
3.5 Datensatz . . . . .	18
3.5.1 COCO Data Annotation Format . . . . .	18
3.5.2 Evaluationsmetriken . . . . .	18
<b>4 Verwandte Werke</b>	<b>21</b>
4.1 Detektion und Segmentierung von Mitochondrien in Rasterelektronenmikroskop Bildern . . . . .	21
4.2 FibeR-CNN . . . . .	22
4.3 Detectron2 . . . . .	22
<b>5 Durchführung</b>	<b>23</b>
5.1 Implementierung der Nanocontrol Schnittstelle in Python . . . . .	23

5.2	Implementierung der GeminiSEM API in Python . . . . .	25
5.3	Aufnahme der Bilddaten . . . . .	25
5.3.1	Variation der aufzunehmenden Szene . . . . .	26
5.3.2	Bildaugmentierung durch REM-Parameter . . . . .	27
5.3.3	Automatisierung . . . . .	29
5.3.4	Ergebnis . . . . .	30
5.4	Daten Annotation . . . . .	31
5.4.1	Segmentierung . . . . .	31
5.4.2	Konvertierung in das COCO Datenformat . . . . .	32
5.5	Datensätze . . . . .	33
5.6	Implementierung und Training der Modelle . . . . .	35
5.6.1	Entwicklungsumgebung . . . . .	35
5.6.2	Implementierung von Mask R-CNN . . . . .	35
5.6.3	Verwendung und Anpassung der vorge trainierten Gewichte . . . . .	36
5.6.4	Wahl der Hyperparameter . . . . .	37
5.6.5	Erweiterung der Modelle: Richtungsvorhersage . . . . .	38
<b>6</b>	<b>Evaluation</b> . . . . .	<b>39</b>
6.1	Evaluationsmetriken . . . . .	39
6.2	Auswertung auf dem Validierungsdatensatz . . . . .	40
6.2.1	Ergebnisse der Segmentierung . . . . .	41
6.2.2	Ergebnisse der Keypoint-Vorhersage . . . . .	42
6.3	Fehlerhafte Vorhersagen . . . . .	43
6.4	Vorhersage mit Spitzenrichtung . . . . .	44
6.5	Optische Überprüfung . . . . .	46
<b>7</b>	<b>Diskussion der Modellleistung</b> . . . . .	<b>50</b>
7.1	Wahl des Backbones . . . . .	50
7.2	Wahrheitswert der Vorhersagen . . . . .	50
7.3	Leistungssteigerung durch Richtungsvorhersage . . . . .	51
7.4	Qualität der Segmentierung . . . . .	51
7.5	Genauigkeit und Probleme der Keypoint-Vorhersage . . . . .	52
<b>8</b>	<b>Zusammenfassung und Ausblick</b> . . . . .	<b>53</b>
<b>Abkürzungsverzeichnis</b>		<b>55</b>
<b>Abbildungsverzeichnis</b>		<b>56</b>
<b>Tabellenverzeichnis</b>		<b>58</b>
<b>Literaturverzeichnis</b>		<b>59</b>

# 1 Einführung

## 1.1 Motivation

Die Halbleiterindustrie befasste sich mit der Entwicklung und Herstellung von Halbleiterbauelementen. Diese Bauelemente sind das Herzstück vieler moderner Systeme, von Computern und Mobiltelefonen bis hin zu Fahrzeugen und medizinischen Geräten. Sie ist ein sich ständig weiterentwickelnder Bereich, der sich auf dem Weg zu immer kleineren und komplexeren Strukturen befindet. Heutige Strukturgrößen liegen unter 10 nm, was etwa einem Zehntausendstel des Haardurchmessers entspricht.

Die Fehleranalyse solch kleiner Strukturen ist eine hochspezialisierte Aufgabe, die jahrelange Erfahrung und Fachwissen erfordert, aber nach der Entwicklung und Produktion unerlässlich ist. Die Bedienung der dafür eingesetzten Geräte ist komplex, zudem kann die für die Analyse notwendige Bildgebung und Interaktion zu Beschädigungen führen, die unbedingt vermieden werden müssen. Ein wichtiger Schritt ist die zuverlässige Lokalisierung von Messspitzen. Diese Spitzen werden verwendet, um die Strukturen zu kontaktieren und Messungen an ihnen durchzuführen. Eine genaue Kenntnis der Position der Spitzen ist entscheidend, um Beschädigungen auszuschließen.

Die Technologie der Fehleranalyse entwickelt sich ständig weiter, um Prozesse zu automatisieren. Die Herausforderungen bei der Bilderkennung sind jedoch aktuell noch nicht bewältigt. Klassische Algorithmen stoßen an ihre Grenzen, wenn sich Bildbedingungen und Hintergründe stark ändern, was zu unzureichenden oder ungenauen Lokalisierungen der Messspitzen führt.

Die jüngsten Fortschritte in der Bilderkennung durch Deep Learning eröffnen jedoch neue Möglichkeiten zur Bewältigung dieser Herausforderungen. Deep Learning ist ein Teilbereich des maschinellen Lernens, der sich mit neuronalen Netzen befasst, die das Lernen und Denken des Menschen nachahmen sollen. Durch ihre Fähigkeit, komplexe Muster und Strukturen in großen Datenmengen zu erkennen, könnten Deep Learning-Algorithmen eine genauere und zuverlässigere Identifizierung von Messspitzen ermöglichen, selbst unter wechselnden Bedingungen und trotz unterschiedlicher Szenarien.

Eine verbesserte Spitzenerkennung könnte dazu beitragen, die Genauigkeit und Zuverlässigkeit von Handhabungsprozessen zu erhöhen, sie zu automatisieren und die potenzielle Schadensrate zu verringern. Vor diesem Hintergrund ist es das Hauptziel dieser Arbeit, die Anwendung und Leistungsfähigkeit von Deep Learning-Techniken für die Spitzenerkennung zu untersuchen und zu evaluieren, um einen Beitrag zur weiteren Automatisierung und Optimierung der Prozesse in der Fehleranalyse zu leisten.

## 1.2 Problemstellung

Die Genauigkeitsanforderungen zur präzisen Erkennung und Lokalisierung der Messspitzen stellen hohe Ansprüche an die Qualität der verwendeten Bildverarbeitungsalgorithmen. Darüber hinaus stellen die Eigenschaften rasterelektronenmikroskopischer Bilder zusätzliche Herausforderungen dar. Aufgrund der Funktionsweise und der Anforderungen der Fehleranalyse

enthalten die Bilder oft ein starkes Bildrauschen und große Kontrastschwankungen. Die große Variabilität der Probenstrukturen und Messspitzenformen je nach Betrachtungsart erschweren die Lokalisierung weiter.

Deep Learning-Techniken haben das Potenzial, robust gegenüber Faktoren wie Bildrauschen und anderen Bildfehlern zu sein und können verschiedene Formen von Messspitzen berücksichtigen. Die Anwendung von Deep Learning auf das Problem der Erkennung von Messspitzen bringt jedoch spezifische Herausforderungen mit sich. Eine davon ist der Bedarf an umfangreichen und qualitativ hochwertigen Trainingsdaten. Die Erstellung solcher Datensätze ist zeit- und arbeitsintensiv und erfordert Expertenwissen, um die Spitzen in den Bildern korrekt zu annotieren.

Eine weitere Herausforderung ist die Evaluation und Bewertung der Genauigkeit von Deep Learning Modellen. Dies ist im Kontext der Fehleranalyse besonders kritisch, da kleine Fehler zu großen Schäden führen können. Darüber hinaus sind Deep Learning-Modelle oft komplex und ihre Entscheidungen schwer zu interpretieren, was die Bewertung ihrer Leistung und die Verbesserung ihrer Genauigkeit zusätzlich erschwert.

Mask R-CNN ist ein spezielles Deep Learning-Modell, das für die Erkennung und Lokalisierung von Objekten in Bildern entwickelt wurde. Es kann verwendet werden, um bestimmte Merkmale in Bildern zu identifizieren sowie zu lokalisieren.

In dieser Arbeit soll daher untersucht werden, inwieweit Deep Learning-Modelle wie Mask R-CNN in der Lage sind, die Spitzen und die Konturen der einzelnen Spitzen in Bildern zu erkennen. Die genaue Definition der Problemstellung sowie die Bewertung der erzielten Ergebnisse sind entscheidend, um das Potenzial von Deep Learning zu bewerten.

### **1.3 Aufbau der Arbeit**

Diese Arbeit ist in acht Hauptkapitel unterteilt. Kapitel zwei konzentriert sich auf die technischen Aspekte des Nanoprobing-Prozesses und beschreibt die verwendeten Geräte, einschließlich des Rasterelektronenmikroskops. Das dritte Kapitel befasst sich mit den Grundlagen neuronaler Netze, einschließlich Convolutional Neural Networks (CNNs), und beschreibt detailliert das verwendete Mask R-CNN-Modell. Relevante Arbeiten auf diesem Forschungsgebiet werden in Kapitel vier vorgestellt. Kapitel fünf beschreibt die Durchführung der Arbeit, einschließlich der Implementierung, der Datenerfassung und des Trainingsprozesses. Das sechste Kapitel befasst sich mit der umfassenden Auswertung der Ergebnisse, gefolgt von einer Diskussion der Modellleistung im siebten Kapitel. Das achte Kapitel gibt eine abschließende Zusammenfassung und einen Ausblick auf mögliche zukünftige Forschungsrichtungen.

# 2 Grundlagen Nanoprobing

In diesem Kapitel werden die Grundlagen des Nanoprobing behandelt. Geräte und Software, die zum Einsatz kommen, werden erklärt.

Die Firma Kleindiek Nanotechnik stellt Geräte und Messelektronik für das Nanoprobing her und ist auf diesem Gebiet weltweit führend [Kle23]. Für eine ausführliche Erklärung von Nanoprobing sowie einen Einblick in die Firma Kleindiek Nanotechnik GmbH, wird auf das Video von Roman Hartung verwiesen. [Har21]

## 2.1 Nanoprobing

Nanoprobing ist eine Technik aus dem Bereich der Nanoelektronik, die zur Analyse von Bauteilen im Nanometerbereich eingesetzt wird. Die Technik verwendet spezielle Messspitzen, um mit nanoskaligen Strukturen zu interagieren und deren Eigenschaften zu bestimmen. Ein wichtiges Anwendungsgebiet in der Halbleiterindustrie ist die Fehleranalyse von Transistoren. Dabei werden elektrische Fehler wie Kurzschlüsse, Unterbrechungen, Widerstände und Leckpfade detektiert und analysiert.

### 2.1.1 Prober Shuttle

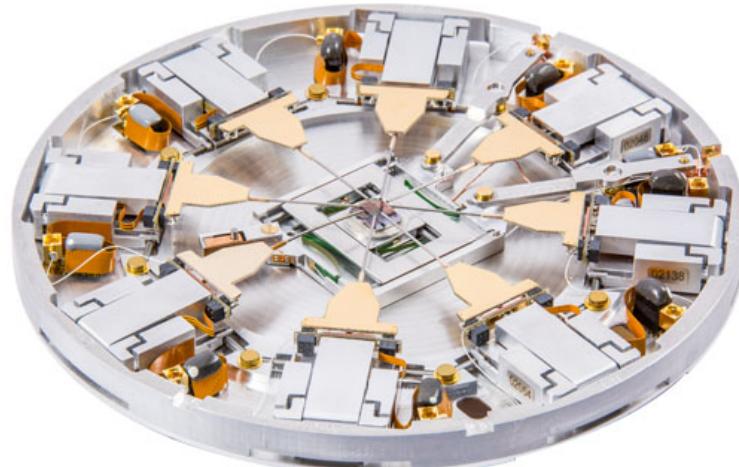


Abbildung 2.1: Ein voll bestücktes Prober Shuttle (PS8)

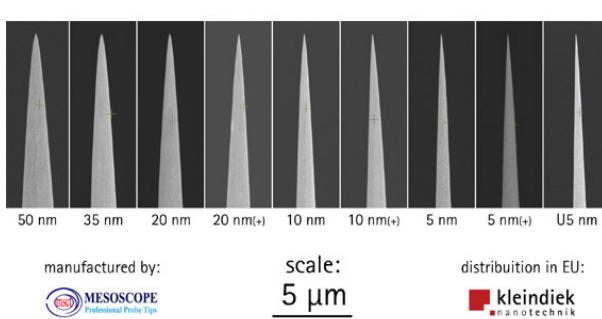
Die Firma Kleindiek Nanotechnik hat das Prober Shuttle entwickelt, ein System, das je nach Modell (PS4 oder PS8) mit vier bis acht Manipulatoren ausgestattet ist. Jeder Manipulator kann in drei Achsen bewegt werden: A für links/rechts, B für oben/unten, C für rein/raus. Durch den Einsatz eines speziell entwickelten Piezomotors arbeiten die Manipulatoren im Subnanometerbereich und sind damit in der Lage, die Kontakte der neuesten Transistorgeneration in der Chipfertigung präzise anzufahren und diese auf Fehler zu überprüfen.

Das PS8e ist eine spezielle Version des Prober Shuttles. Das „e“ steht für „encoded“ und bezieht sich auf kapazitive Felder unter den Manipulatoren, die deren Position bestimmen können. Nach einmaliger Kalibrierung kann das Prober Shuttle die Messspitzen bis auf 10 µm zusammenfahren. Aufgrund der geringen Größe der Kontakte – einige zehn Nanometer – ist jedoch eine manuelle Steuerung der Manipulatoren für die Kontaktierung erforderlich.

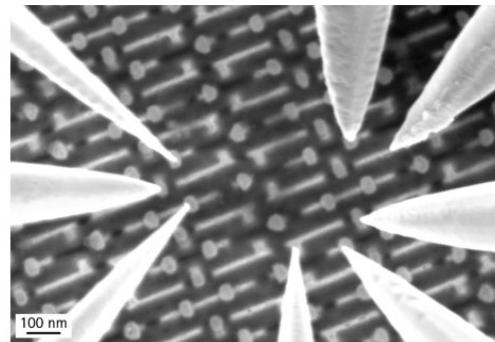
Die Ansteuerung der Manipulatoren erfolgt über das Nanocontrol (NC). Es bietet Drehknöpfe zur Ansteuerung, verarbeitet aber auch über RS232 gesendete Befehle und steuert den Manipulator entsprechend an. Jeder Manipulator hat ein eigenes NC, diese sind zusammen mit der Messelektronik in einem Rack montiert. Ein voll bestücktes Rack ist in Abbildung 2.5 zu sehen.

### 2.1.2 Messspitzen

Die Messspitzen aus massivem Wolfram sind in verschiedenen Ausführungen erhältlich. Der Schaftdurchmesser beträgt 0.25 mm, der Spitzenradius variiert zwischen 250 nm und 5 nm. Aufgrund der unterschiedlichen Spitzenradien können die Spalten in den Mikroskopiebildern zum Teil sehr unterschiedlich aussehen. Wie in Abbildung 2.2(b) zu erkennen ist.



(a) Messspitzen



(b) Auf Kontakten abgesetzte Messspitzen

Abbildung 2.2: Messspitzen der Firma Mesoscope in verschiedenen Ausführungen. Nach Spitzenradii sortiert 2.2(a) und abgesetzt auf den Kontakten mehrerer Transistoren 2.2(b)

## 2.2 Rasterelektronenmikroskop

Das Rasterelektronenmikroskop (REM) ist in der Halbleiterindustrie unverzichtbar, da es die Oberflächentopografie und -zusammensetzung von Proben mit einer Auflösung bis zu 10 nm sichtbar machen kann. Im Gegensatz zu herkömmlichen Lichtmikroskopen arbeitet das REM mit einem fokussierten Elektronenstrahl, der beim Auftreffen auf die Probenoberfläche eine Reihe von Signalen erzeugt. Diese Signale werden dann in ein sichtbares Bild umgewandelt. Zwei Arten von Elektronen-Proben-Wechselwirkungen können unterschieden werden: elastische und inelastische Wechselwirkungen. Elastische Wechselwirkungen führen zur Erzeugung von RückstreuElektronen (BSE), während inelastische Wechselwirkungen zur Erzeugung von Sekundärelektronen (SE) führen. Diese werden zusammen mit anderen Signalen wie Röntgenstrahlung und Augerelektronen zur Bildanalyse und -erzeugung verwendet.

Die Arbeitsweise eines REM besteht aus mehreren Schritten. Dargestellt sind diese in Abbildung 2.3. Zunächst erzeugt eine Elektronenkanone einen stabilen Elektronenstrahl, der auf einen kleinen Punkt gerichtet ist. Es gibt verschiedene Arten von Elektronenkanonen, zum Beispiel Wolfram-Elektronenkanonen, Lanthanhexaborid (LaB<sub>6</sub>)-Elektronenkanonen und Feldemissionskanonen. Der erzeugte Elektronenstrahl wird durch Linsen – Kondensorlinsen und Objektivlinsen – gebündelt und auf die Probe fokussiert. Die Kondensorlinsen richten den Elektronenstrahl parallel aus, während die Objektivlinsen den Strahl auf einen Punkt fokussieren. Zur Bildentstehung lenkt eine Abtastspule den Strahl entlang der x- und y-Achse ab, um die gesamte Probe abzudecken. Dies ermöglicht eine stufenlose Vergrößerung von 10x bis 2.000.000x. Der Sekundärelektronendetektor schließlich detektiert die emittierten Sekundärelektronen, die eine feine topografische Information liefern.

Die Eindringtiefe des Elektronenstrahls und die Oberflächenauflösung hängen von der Energie des Elektronenstrahls und der Zusammensetzung der Probe ab. Hier gilt es, ein Gleichgewicht zu finden und je nach Anforderung die optimale Strahlenergie zu bestimmen [Ink16][ZAWJ07].

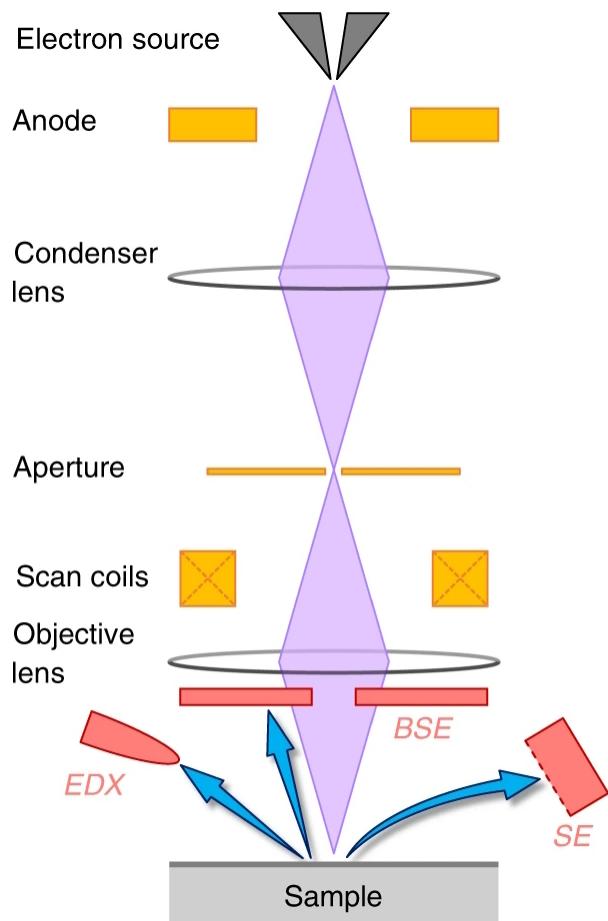


Abbildung 2.3: Grundlegender Aufbau eines Rasterelektronenmikroskops.  
Quelle: nature.com [SRP19]

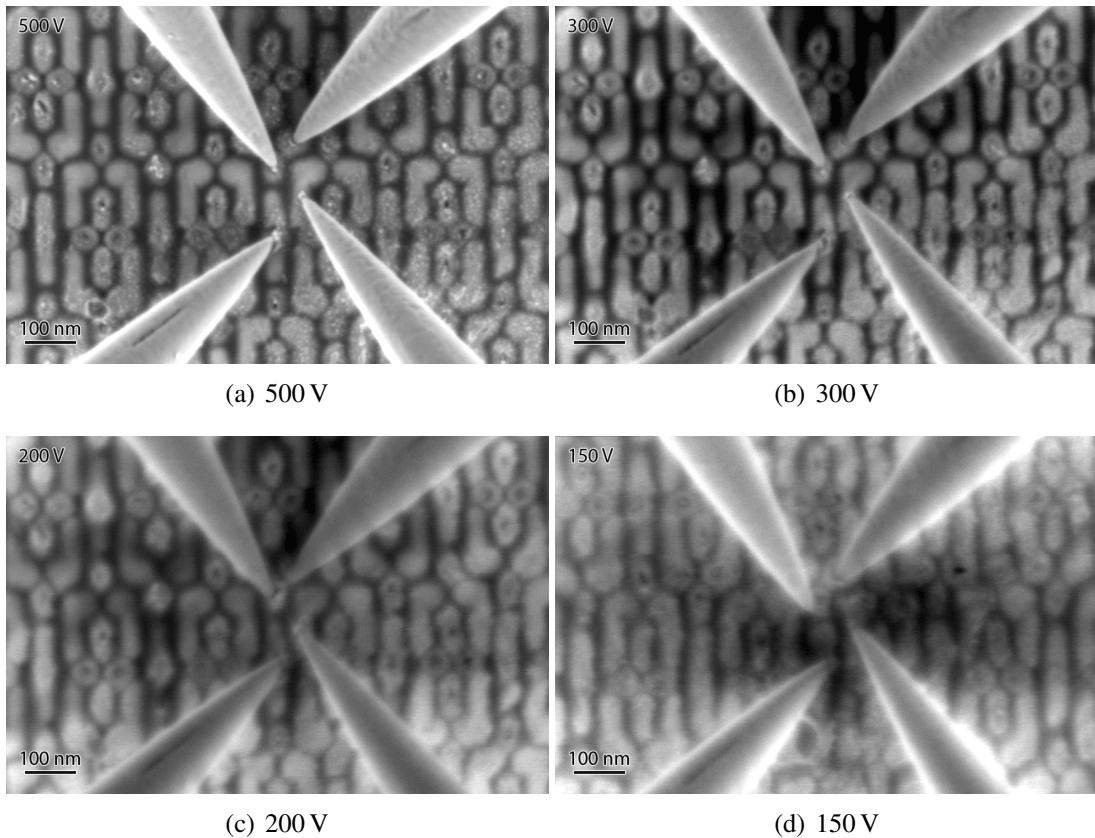


Abbildung 2.4: Visualisierung der Messspitzen bei verschiedenen Beschleunigungsspannungen. Quelle: Kleindiek Nanotechnik GmbH

Bei der Untersuchung von Transistoren mit einem REM spielt die Beschleunigungsspannung, mit der die Elektronen auf die Probe geschossen werden, eine besonders wichtige Rolle. Ist die Beschleunigungsspannung zu hoch, kann die empfindliche Struktur der Transistoren beschädigt werden. Für die Analyse neuester Chiptechnologien wird daher eine Beschleunigungsspannung von weniger als 200 Volt benötigt. Wie die Abbildung 2.4 zeigt, wirkt sich diese niedrige Beschleunigungsspannung jedoch negativ auf die Bildqualität aus, da sie eine Vielzahl von Bildstörungen verursacht. Dies äußert sich unter anderem in einer geringeren Bildschärfe und einem geringeren Kontrast.

## **ZEISS GeminiSEM Mikroskop und ZEISS SmartSEM Software**

Das in dieser Arbeit verwendete REM ist ein ZEISS GeminiSEM 300, es erfüllt die höchsten Anforderungen an Subnanometer-Imaging, Analytik und Probenflexibilität [ZEI23]. Gesteuert wird es über die „ZEISS SmartSEM Software“, die eine benutzerfreundliche Oberfläche zur effizienten Steuerung und Optimierung des REM-Betriebs bietet. Mit der „SmartSEM Remote API“ stellt ZEISS Entwicklern eine Schnittstelle zur Fernsteuerung des Mikroskops zur Verfügung. So können über speziell entwickelte Skripte analoge und digitale Parameter gesetzt, der Status des Mikroskops ausgelesen und Befehle gesendet werden.

## 2.3 Kleindiek Nanotechnik: Arbeitsplatz



Abbildung 2.5: Arbeitsplatz bei Kleindiek. Quelle: Kleindiek Nanotechnik GmbH

Der verwendete Arbeitsplatz ist in Abbildung 2.5 dargestellt. Er ist mit einem GeminiSEM 300 ausgestattet, in das ein voll ausgestattetes PS8e eingebaut ist. Das Mikroskop wird über den links zu sehenden PC gesteuert. Das Rack (rechts neben dem Mikroskop) enthält neun Nanocontroller. Acht für die Steuerung der Manipulatoren und einer für die Stage, auf der die Probe montiert ist. Die gesamte Messelektronik und ein weiterer PC sind ebenfalls im Rack untergebracht. Alle Geräte werden über die von ZEISS und Kleindiek zur Verfügung gestellte Software ferngesteuert.

Da im Rahmen dieser Arbeit eine eigens entwickelte Steuerung zum Einsatz kommt, werden die Nanocontrols und das REM über einen eigenen Laptop gesteuert und nicht über die PCs am Arbeitsplatz und im Rack.

# 3 Grundlagen neuronaler Netze

Im Rahmen dieses Kapitels werden die Grundlagen neuronaler Netze sowie die Convolutional Neural Networks (CNN) vermittelt. Es werden die Aufgaben erklärt, die mithilfe dieser Netze gelöst werden und die in dieser Arbeit verwendete Netzarchitektur erläutert.

## 3.1 Maschinelles Sehen

### 3.1.1 Objektdetektion

Ziel der Objektdetektion ist es, das Vorhandensein von Objekten zu erkennen, ihre Position zu bestimmen und ihnen geeignete Klassenbezeichnungen zuzuordnen. Diese Vorhersagen werden in der Regel durch eine Bounding Box dargestellt, die durch die Werte (x, y, Breite, Höhe) und die zugehörigen Klassenbezeichnungen definiert ist [Pap23a].

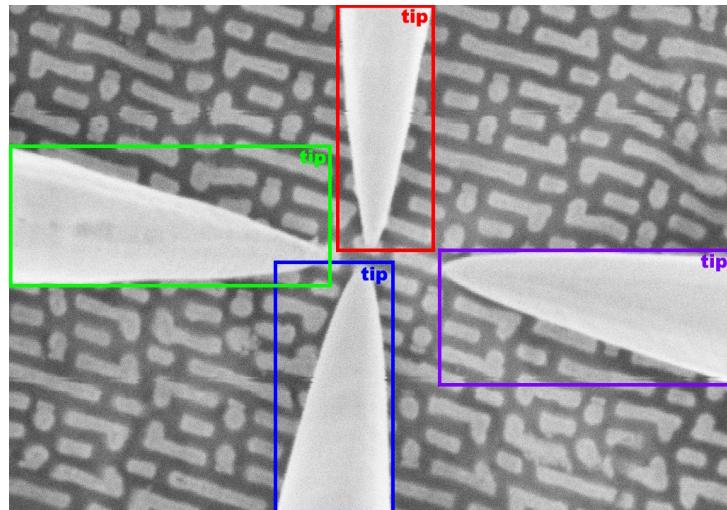


Abbildung 3.1: Messspitzen, die zur Lokalisierung und Differenzierung von Bounding Boxes umgeben sind. Quelle: Kleindiek Nanotechnik GmbH

### 3.1.2 Instanz Segmentierung

Die Instanz Segmentierung geht über die bloße Lokalisierung eines Objekts mit einer Bounding Box hinaus, indem jedem Pixel eine Klasse und Identifikationsnummer zugewiesen wird. Im Kontext der Spitzenerkennung ermöglicht dies die pixelgenaue Vorhersage, welche Bereiche des Bildes eine Spalte enthalten, sowie auch die Unterscheidung zwischen den einzelnen Spitzen. Jede Spalte erhält eine eindeutige Identifikation für genauere Analyse und Weiterverarbeitung [Pap23a].

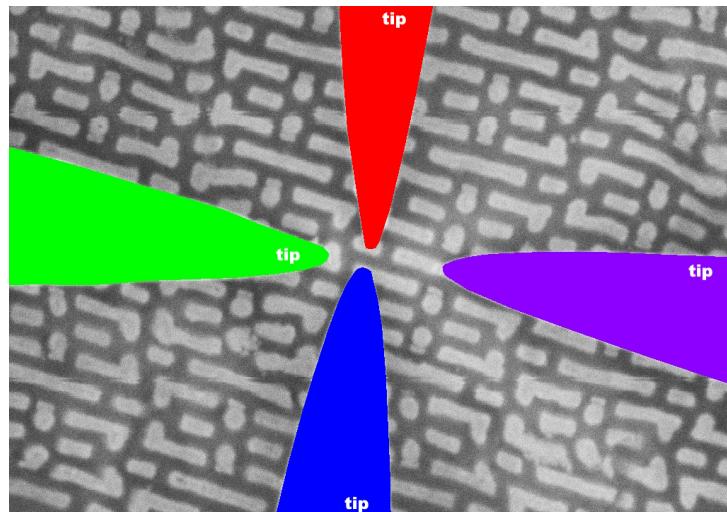


Abbildung 3.2: Messspitzen, deren spezifische Formen durch Instanz Segmentierung hervorgehoben werden. Quelle: Kleindiek Nanotechnik GmbH

### 3.1.3 Keypoint Erkennung

Die Keypoint Erkennung identifiziert spezifische, markante Punkte von Objekten in einem Bild, die durch ihre x-y-Koordinaten definiert sind. Im Zusammenhang mit der Erkennung von Messspitzen ist der vorderste Punkt von besonderem Interesse, da er den direkten Kontakt mit der Probe herstellt und seine genaue Bestimmung wichtig ist [Pap23a].

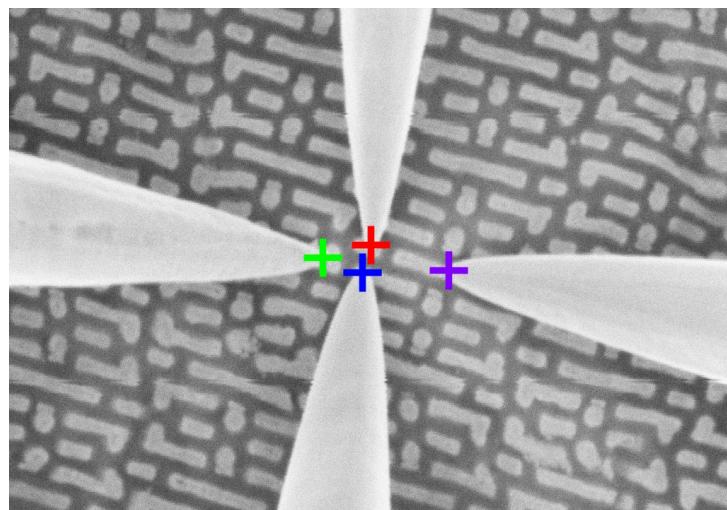


Abbildung 3.3: Messspitzen, deren vordere Punkte – die wesentlichen Kontaktpunkte bei der Probenmessung – deutlich markiert sind. Quelle: Kleindiek Nanotechnik GmbH

## 3.2 Künstliche neuronale Netze

In diesem Abschnitt werden der grundlegende Aufbau und die Funktionsweise von neuronalen Netzen sowie die verschiedenen Aktivierungsfunktionen, die bei den Berechnungen in den Netzen eine wichtige Rolle spielen, ausführlich beschrieben.

### 3.2.1 Mehrschicht-Perzeptron

Wenn mehrere künstliche Neuronen schichtweise miteinander verbunden werden, ergibt sich ein mehrschichtiges Perzeptron [Blo62][Ros58]. Es besteht aus mindestens drei Schichten, einer Eingabeschicht, einer oder mehreren versteckten Schichten und einer Ausgabeschicht [GD98]. Die Schichten werden in der Regel voll verbunden. Das bedeutet, dass das Neuron einer Schicht eine Verbindung zu jedem Neuron der vorherigen oder nachfolgenden Schicht besitzt. Die Neuronen innerhalb einer Schicht sind dabei nicht miteinander verbunden. Da die Eingabe der Daten über eine einzelne Schicht erfolgt, werden die Daten zu einem eindimensionalen Array abgeflacht. Dies birgt den Vorteil, dass das mehrschichtige Perzeptron nicht

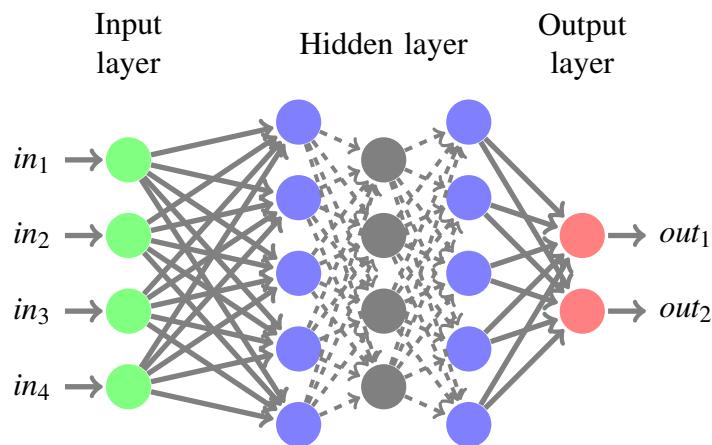


Abbildung 3.4: Ein mehrschichtiges Perzeptron mit 4 Eingängen und 2 Ausgängen. Zwischen Ein- und Ausgangsschicht liegen mehrere verborgene Schichten.

durch die Form der Daten beschränkt ist. Allerdings gehen dadurch räumliche Informationen verloren, die für die Bildverarbeitung fundamental sind. Dieses Problem wird durch das CNN gelöst, das in Abschnitt 3.3 beschrieben wird.

Die folgenden versteckten Schichten sind für die Erkennung und Modellierung komplexer Zusammenhänge in den Eingabedaten. Ihre Anzahl wirkt sich auf die Lernfähigkeit, die Komplexität und die Generalisierungsfähigkeit des Modells aus und variiert je nach Komplexität des zu lösenden Problems und der Größe der verfügbaren Daten. Werden viele versteckte Schichten verwendet, spricht man von einem tiefen neuronalen Netz.

Die Ausgabeschicht ist die letzte Schicht des Netzwerkes und hat die Aufgabe, die Ergebnisse des Netzes in einer für das Problem geeigneten Form darzustellen. Sie besteht aus einer Anzahl von Neuronen, die der Anzahl der gewünschten Ausgabewerte entspricht.

### 3.2.2 Aktivierungsfunktion

Die Aktivierungsfunktion ist ein wesentlicher Bestandteil von künstlichen neuronalen Netzen. Sie wird auf die Summe der gewichteten Eingaben eines jeden Neurons angewendet und generiert die Ausgabe.

Die Nichtlinearität der Funktion ist dabei von entscheidender Bedeutung. Sie ermöglicht dem neuronalen Netz die Modellierung komplexer Muster und Beziehungen in den Daten und damit die Approximation komplexer Funktionen. Ohne die Nutzung einer Aktivierungsfunktion, würde das Netzwerk nur lineare Transformationen durchführen, was die Generalisierungsfähigkeit stark einschränken würde. Einige Aktivierungsfunktionen haben einen Schwellenwert, bei deren Überschreitung sie aktiviert werden und einen entsprechenden Ausgabewert erzeugen. Wichtige Aktivierungsfunktionen sind die Sigmoidfunktion, die ReLU-Funktion (Rectified Linear Unit) und die Tanh-Funktion (Hyperbolischer Tangens) [ADIP21].

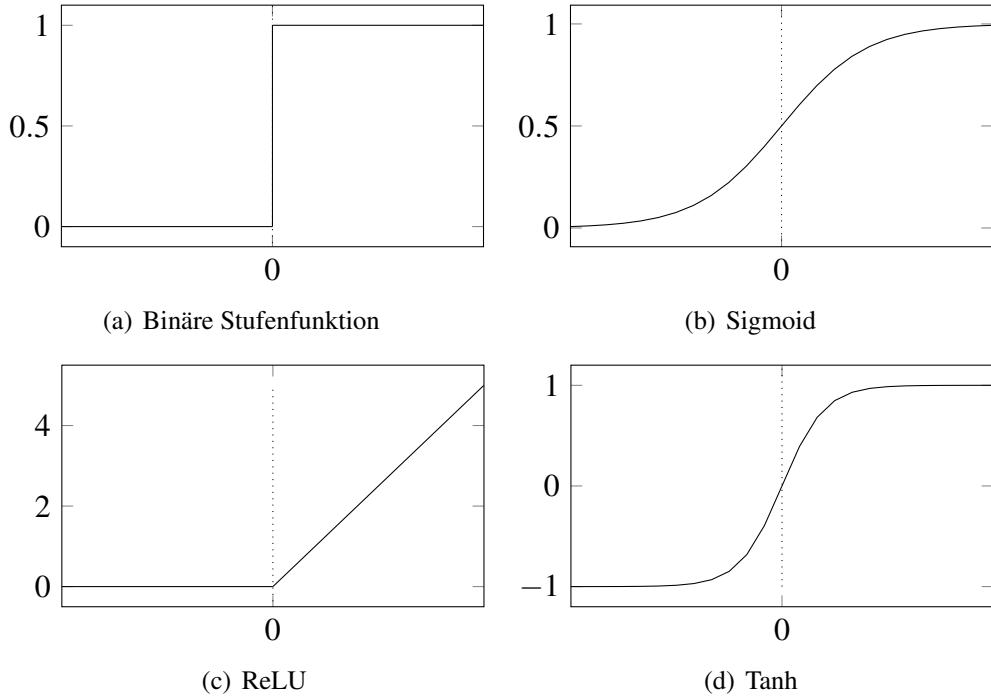


Abbildung 3.5: Graphen der Binären Stufen-, Sigmoid- und ReLU-Funktion

Die Sigmoidfunktion (siehe 3.5(b)) nimmt einen beliebigen Eingabewert und wandelt ihn in einen Ausgabewert zwischen 0 und 1 um. Sie ist besonders nützlich in der Ausgabeschicht von Klassifikationsproblemen, bei denen die Ausgabe eine Wahrscheinlichkeit ist. Die Tanh-Funktion (dargestellt in 3.5(d)) wandelt, ähnlich wie die Sigmoid-Funktion, jeden Eingabewert in einen Ausgabewert zwischen -1 und 1 um, was sie nützlicher für Ausgaben macht, die sowohl positive als auch negative Werte annehmen sollen. Im Gegensatz dazu gibt die ReLU-Funktion (vgl. 3.5(c)) den Eingabewert unverändert zurück, wenn er positiv ist, und Null, wenn er negativ ist. Ein wesentlicher Vorteil der ReLU-Funktion ist, dass sie das Problem des verschwindenden Gradienten – bei dem die tiefen Schichten des Netzes nicht lernen – vermeidet, das bei älteren Aktivierungsfunktionen wie Tanh und Sigmoid auftritt [Ato23]. Außerdem hat sie einen geringeren Rechenaufwand.

Jede dieser Aktivierungsfunktionen hat ihre eigenen Eigenschaften und Verwendungszwecke, und die Wahl der richtigen Funktion hängt von dem spezifischen Problem und den Eigenschaften der Daten ab. In der Praxis werden häufig verschiedene Aktivierungsfunktionen in verschiedenen Schichten eines neuronalen Netzes verwendet, um die Leistung zu optimieren.

## 3.3 Convolutional Neural Networks

CNNs sind eine Form künstlicher neuronaler Netze, die speziell für die Verarbeitung und Extraktion räumlicher Informationen aus Daten, insbesondere aus Bildern und anderen mehrdimensionalen Datenstrukturen, entwickelt wurden [AMAZ17]. Sie sind ein wichtiger Bestandteil in vielen modernen maschinellen Lernsystemen, insbesondere im Bereich des maschinellen Sehens und haben wesentlich dazu beigetragen, die Leistungsfähigkeit von Algorithmen in Aufgaben wie Bild- und Videoklassifikation, Objekterkennung und Segmentierung zu verbessern [ON15].

Das grundlegende Prinzip eines CNN besteht darin, kleine, lokalisierte Merkmale aus den Eingabedaten zu lernen und diese dann zu abstrahieren und zu kombinieren, um komplexe Muster und Strukturen zu erkennen. Ein CNN besteht typischerweise aus drei Arten von Schichten: Convolution-Schichten (Abbildung 3.3.1), Pooling-Schichten (dargestellt in 3.3.2) und Fully Connected-Schichten, die einem mehrschichtigen Perzeptron (Erklärung in 3.2.1) ähnlich sind.

### 3.3.1 Convolution-Schichten

Convolution-Schichten sind entscheidend für die Extraktion von Merkmalen aus den Eingabedaten. Im Folgenden wird die Arbeitsweise näher erläutert.

In einer Faltungsschicht wird ein Filter – auch Kernel genannt –, der aus einer Matrix von Gewichten besteht, über die gesamte Eingabe geschoben. Dieser Vorgang wird als Convolution (Faltung) bezeichnet und ist in Abbildung 3.6 dargestellt. Klassische Filtergrößen sind dabei  $3 \times 3$ ,  $5 \times 5$  oder  $7 \times 7$ . Die Filter sind daher in der Regel viel kleiner als die Eingabe, decken aber die gesamte Tiefe der Eingabe ab. Bei einem Farbbild beträgt die Tiefe beispielsweise drei.

Durch die Anwendung dieser Filter auf die Eingabedaten, identifizieren die Convolution-Schichten Muster, Kanten und Texturmerkmale. Die Simulation des Verschiebens der Filter über die Eingabedaten wird durch sogenannte rezeptive Felder simuliert. Dabei werden die Neuronen einer Schicht ausschließlich mit den Neuronen der vorhergehenden Schicht verbunden, die innerhalb des durch die Filtergröße definierten rezeptiven Feldes liegen. Diese lokale Verknüpfung ermöglicht es dem CNN, räumliche Informationen zu erfassen und lokal relevante Merkmale zu extrahieren, während gleichzeitig die Anzahl der zu lernenden Parameter durch die gemeinsame Nutzung von Parametern (da der Filter auf mehrere Bereiche des Bildes angewendet wird) reduziert wird [Wei23].

Um die Ausgabegröße der Convolution-Schichten zu steuern und sicherzustellen, dass sie genügend räumliche Information enthalten, können zwei wichtige Parameter verwendet werden: Padding und Stride.

Padding bezieht sich auf das Hinzufügen zusätzlicher Randpixel zur Eingabe, bevor der Filter angewendet wird. Dadurch wird sichergestellt, dass die Größe der Ausgabe erhalten bleibt, insbesondere wenn der Filter über die Ränder der Eingabe gleitet [Pan22].

Stride bezieht sich auf die Anzahl der Pixel, um die der Filter bei jedem Schritt über die Eingabe gleitet. Ein Stride von eins bedeutet, dass der Filter Pixelweise verschoben wird, während ein größerer Stride zu größeren Schritten führt. Ein größerer Stride reduziert die räumliche Dimension der Ausgabe, da weniger Positionen des Filters überprüft werden. Dadurch kann die Größe der Netzarchitektur reduziert werden [Pan22].

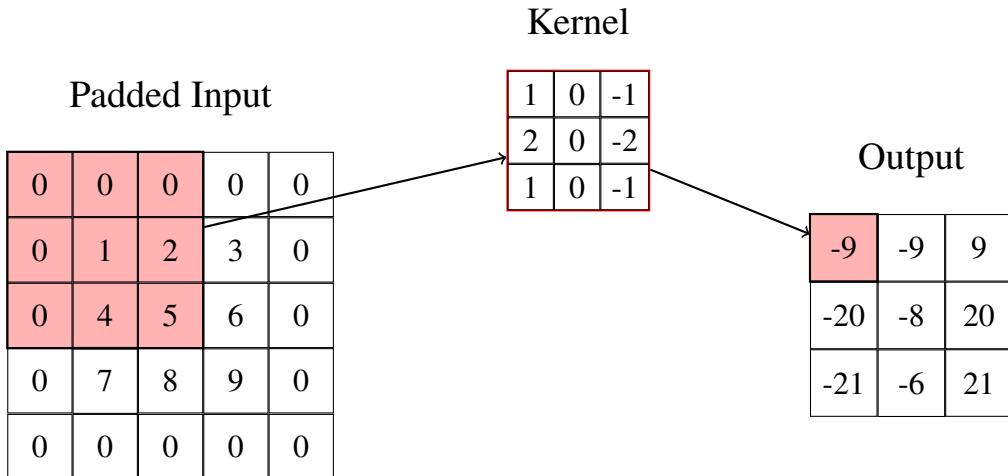


Abbildung 3.6: Zero-padded Bild. Der Stride ist eins. Somit entspricht die Ausgangsgröße der Faltung der Eingangsgröße. Der Sobel-Operator wird als Kernel verwendet [KVB88].

### Dilated-Convolution

Dilated Convolution, auch bekannt als Atrous Convolution, ist eine von Yu *et al.* entwickelte Technik zur effektiven Erweiterung des Rezeptiven Feldes in einer CNN-Architektur, ohne dabei die räumliche Auflösung zu verlieren oder die Anzahl der Parameter zu erhöhen [YK16]. Die Technik verwendet eine sogenannte Verdünnungsrate (dilation rate), die die Abstände zw-

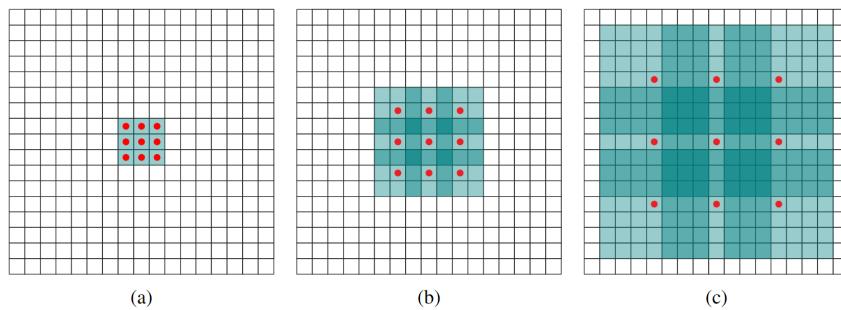


Abbildung 3.7: (a) 1-dilated convolution mit 3x3 rezeptivem Feld, (b) 2-dilated convolution mit 7x7 rezeptivem Feld, (c) 4-dilated convolution mit 15x15 rezeptivem Feld  
Quelle: Dilated Residual Networks [YKF17]

schen den Gewichten in der Filtermaske vergrößert. Ein Beispiel ist in Abbildung 3.7 dargestellt. Durch die Anwendung von Dilated-Convolutions kann das Netz somit mehr Kontextinformation aus dem Eingabebild extrahieren, ohne die Komplexität des Netzes wesentlich zu erhöhen. Yu *et al.* zeigen, dass diese Technik besonders wertvoll in Anwendungen ist, die eine detaillierte Segmentierung erfordern, da sie es ermöglicht, Informationen über größere Eingangsbereiche zu sammeln und gleichzeitig eine hohe Auflösung in der Ausgabe beizubehalten [YKF17]. Insbesondere bei Aufgaben, bei denen es auf die genaue Lokalisierung von Objekten ankommt, wie beispielsweise der Spitzenerkennung in Nanoprobing-Anwendungen, kann der Einsatz von Dilated-Convolutions zu verbesserten Ergebnissen führen.

### 3.3.2 Pooling-Schichten

Pooling-Schichten bilden einen weiteren wichtigen Bestandteil von CNNs, sie werden dazu genutzt, die Dimension der von Convolution-Schichten erzeugten Featuremaps zu reduzieren und gleichzeitig wichtige Merkmale beizubehalten [Qay22]. Dies wird erreicht, indem benachbarte Werte über eine Funktion reduziert werden. Die häufigsten Pooling-Operationen sind das Max-Pooling und das Average-Pooling.

Beim Max-Pooling wird in jedem Pooling-Bereich der maximale Wert als repräsentativer

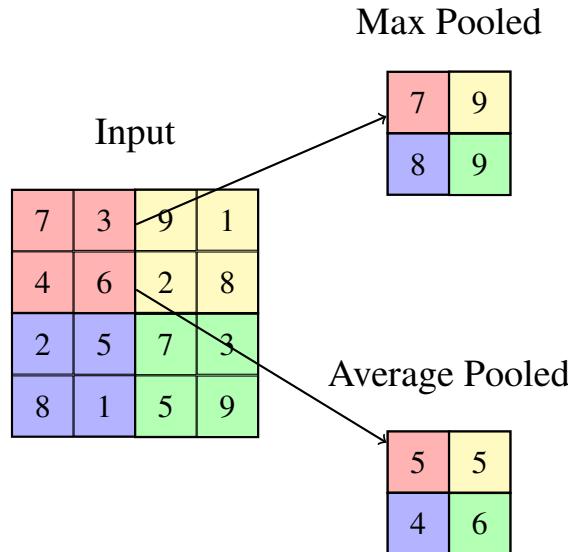


Abbildung 3.8: Pooling dient der Dimensionsreduktion unter Beibehaltung der aussagekräftigsten Merkmale, wobei Methoden wie Max-Pooling und Average-Pooling zum Einsatz kommen.

Wert verwendet. Dies hilft dabei, herausragende Merkmale oder Aktivierungen hervorzuheben und die Positionsinformationen zu erhalten. Beim Average-Pooling wird hingegen der Durchschnitt der Werte im Pooling-Bereich berechnet. Dadurch werden die Werte geglättet und das Netzwerk wird robuster gegenüber kleinen lokalen Verschiebungen. Über die Kernelgröße, den Stride und das Padding kann die Ausgabegröße der Pooling-Schicht wie bei der Convolution bestimmt werden.

## 3.4 Mask R-CNN

Mask R-CNN ist ein innovativer Ansatz für Instanzsegmentierungsaufgaben, der auf dem Konzept der Region Based Convolutional Neural Networks (R-CNNs) basiert [Gir15][GDDM16]. Mask R-CNN wurde 2017 von He *et al.* bei Facebook AI Research entwickelt und erstmals veröffentlicht [HGDG17]. Es stellt einen Meilenstein in der Entwicklung von Algorithmen zur Objekterkennung und -segmentierung dar. Mask R-CNN erweitert und verbessert das von Ren *et al.* entwickelte Modell Faster R-CNN [RHGS15] in mehreren Punkten. Dem Modell

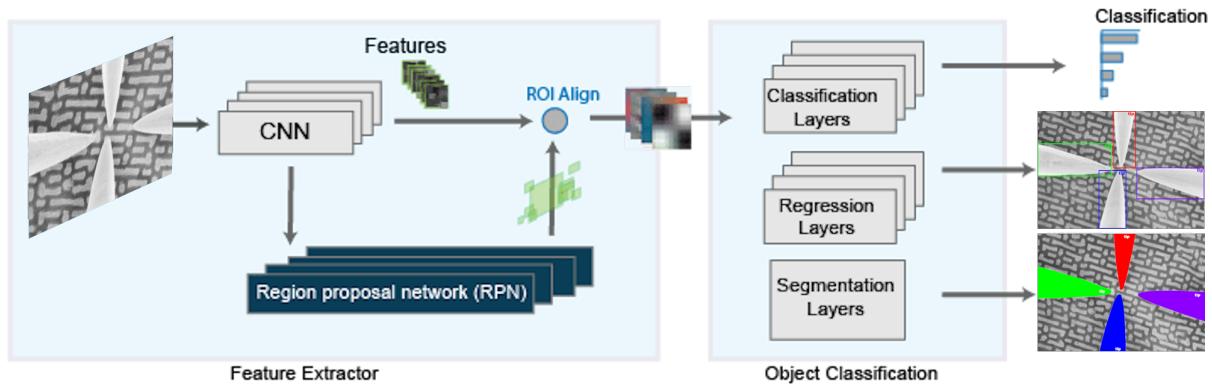


Abbildung 3.9: Mask R-CNN Architektur Quelle: Modifiziert aus der MathWorks Dokumentation [Mat23]

wird ein zusätzlicher Segmentierungszweig hinzugefügt, der es ermöglicht, für jedes erkannte Objekt eine präzise, pixelgenaue Maske zu erzeugen. Diese Eigenschaft unterscheidet Mask R-CNN von Faster R-CNN und erweitert dessen Fähigkeiten über die reine Objekterkennung und -klassifikation hinaus. Darüber hinaus implementiert Mask R-CNN eine sogenannte RoIAlign Schicht, um das Problem der Diskrepanz zwischen dem RoIPooling in Faster R-CNN und der pixelgenauen Segmentierung zu lösen, was zu einer weiteren Verbesserung der Modellleistung führt. Die Implementierung von RoIAlign wird im Rahmen dieser Arbeit nicht weiter behandelt und kann in dem von He *et al.* veröffentlichten Paper [HGDG17] entnommen werden.

### 3.4.1 Backbone Netzwerk

Das Backbone-Netz wird verwendet, um tief abstrahierte Merkmale aus den Eingabedaten zu extrahieren. Es stellt die erste Stufe von Mask R-CNN dar. In Abbildung 3.9 ist sie als „CNN“ linken dargestellt. Die extrahierten Merkmale dienen als Input für den Rest des Netzes, das spezifische Aufgaben wie Objekterkennung, Segmentierung oder Klassifikation durchführt. Die Wahl des Backbone-Netzes hat einen wesentlichen Einfluss auf die Leistungsfähigkeit des Gesamtsystems. Sie hängt von der Komplexität des Problems, den spezifischen Anforderungen der Aufgabe und der Rechenkapazität ab.

Residuale Netzwerke (ResNet) welche erstmal von He *et al.* vorgestellt wurden [HZRS16a], lösen das Problem des vanishing/exploding Gradienten [GDDM16] in tiefen Netzen durch die Nutzung von skip-connections [HZRS16b][OP17][SGS15]. Sie sind eine weit verbreitete Wahl für die Objektdetection und -segmentierung [EAAAM22]. ResNets gibt es in verschiedenen Tiefen, darunter Modelle wie ResNet-50 und ResNet-101, wobei sich jede Zahl auf die Anzahl der Convolution-Schichten im Netzwerk bezieht. Tiefere Modelle können in der Regel komplexere Merkmale erfassen, benötigen jedoch mehr Rechenleistung und Speicherplatz.

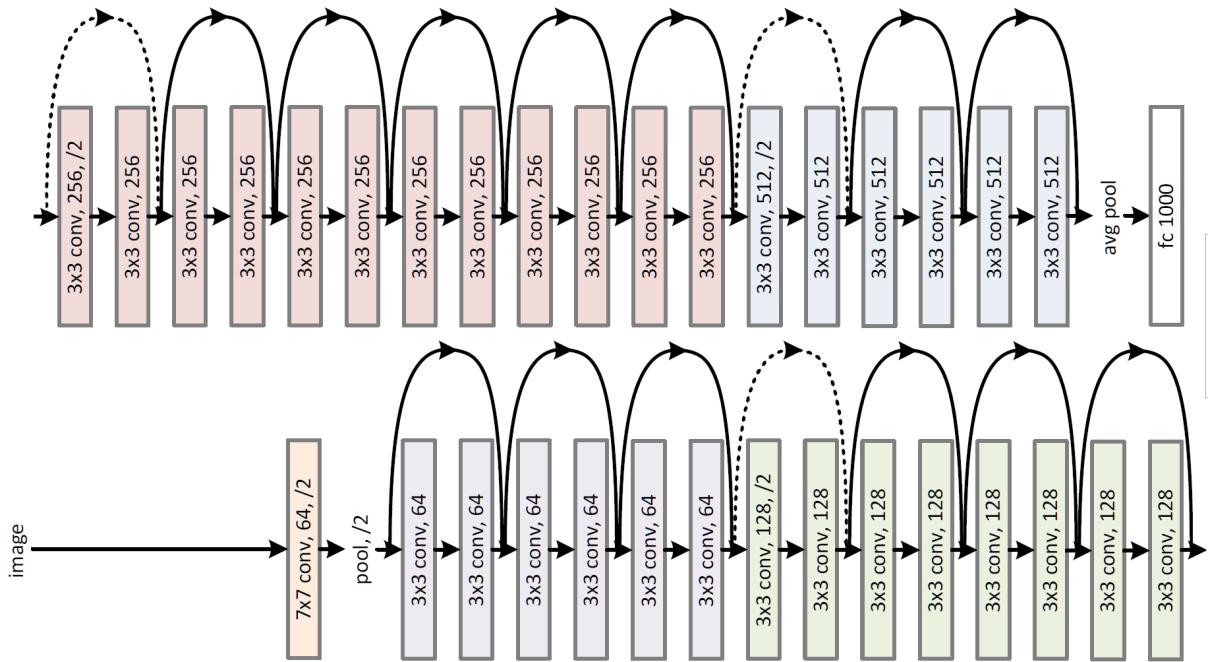


Abbildung 3.10: Architektur des ResNet-34. ResNet Netze bestehen aus mehreren Ebenen. Jede Ebene extrahiert Merkmale auf einer anderen Abstraktionsebene. Quelle: [HZRS16a]

ResNet-FPN, wobei FPN für Feature Pyramid Network steht, ist eine Variante des ResNet-Modells, die speziell für Aufgaben entwickelt wurde, die eine Objekterkennung auf unterschiedlichen Skalen erfordern. Ein FPN verbessert die Fähigkeit von ResNets, Objekte unterschiedlicher Größe zu erkennen, indem eine Pyramide von Feature-Maps erzeugt wird, die von der hochauflösenden, semantisch schwachen Ebene bis zur niedrigauflösenden, semantisch starken Ebene reicht. Für weitere Informationen zu FPN siehe [LDG<sup>+</sup>16].

Eine weitere für diese Arbeit relevante Variante der ResNet-Architektur ist ResNet-DC5. Diese Konfiguration enthält – anstelle von FPN – Dilated Convolutions 3.3.1 in der C5-Ebene von ResNet. Die Integration von Dilated Convolutions ermöglicht es dem Netzwerk, mehr Kontextinformationen aus dem Eingabebild zu extrahieren und somit feinere Details in den Daten zu erkennen. Dies ist besonders vorteilhaft für Aufgaben, bei denen präzise Informationen benötigt werden.

### 3.4.2 Region Proposal Netzwerk

Das Region Proposal Network (RPN) ist ein Schlüsselement in der Architektur von Mask R-CNN. Es ist dafür verantwortlich, Regionen von Interesse (RoI) innerhalb des Bildes zu identifizieren, die potenziell Objekte enthalten könnten. Dieses Netzwerk nimmt die extrahierten Merkmale des Backbone-Netzwerks als Eingabe und gibt eine Reihe von Rechtecken (Bounding Boxes) und Objektivitätsscores aus, die die Wahrscheinlichkeit angeben, mit der ein Objekt in dem jeweiligen Rechteck enthalten ist [HBDS16]. Die vorgeschlagenen Regionen werden durch die ROIAlign Technik mit den Merkmalen, die durch das Backbone extrahiert wurden, verbunden und werden für die anschließende Objektklassifikation und Segmentierung durch die nachfolgenden Stufen des Mask R-CNN Modells verwendet.

### 3.4.3 Vortraining

Die Verwendung vortrainierter Modelle ist eine gängige Praxis im maschinellen Lernen und insbesondere im Deep Learning, da sie erhebliche Vorteile bietet. Der Hauptvorteil ist die erhebliche Zeit- und Rechenersparnis, da das Netz nicht von Grund auf neu trainiert werden muss. Vortrainierte Modelle haben bereits eine Vielzahl von Merkmalen gelernt, die in verschiedenen Kontexten und für eine Vielzahl von Aufgaben nützlich sein können, und führen daher oft zu besseren Ergebnissen. Darüber hinaus hilft die Verwendung von vortrainierten Modellen, das Problem der Überanpassung zu reduzieren, insbesondere wenn die Menge der verfügbaren Trainingsdaten begrenzt ist.

Für das Mask R-CNN Vortraining wird häufig der COCO-Datensatz [COC23] verwendet, der eine Vielzahl von Objektkategorien aus unterschiedlichen Kontexten enthält. Dies bietet eine gute Grundlage für die Erkennung und Segmentierung verschiedener Objekttypen in den meisten gängigen Szenarien. Ein weiteres beliebtes Vortraining ist das auf dem ImageNet-Datensatz [DDS<sup>+</sup>09], der Millionen von Bildern aus Tausenden von Kategorien enthält. Diese vortrainierten Modelle können dann für die spezifischen Anforderungen einer bestimmten Aufgabe weiter trainiert werden, ein Prozess, der auch als Fine-Tuning bezeichnet wird.

### 3.4.4 Hyperparameter

Die Wahl der richtigen Hyperparameter ist entscheidend für das Training von Deep Learning Modellen und kann einen großen Einfluss auf die Leistung des Modells haben.

Ein essenzieller Parameter beim Training von neuronalen Netzen ist die Lernrate. Die Lernrate gibt an, wie stark die Gewichte des Modells in jedem Trainingsschritt angepasst werden. Ein zu hoher Wert kann dazu führen, dass das Modell die optimale Lösung „überspringt“, während ein zu niedriger Wert zu einem sehr langsamem Lernfortschritt führen kann [Rak19].

Momentum ist eine gängige Technik zur Überwindung lokaler Minima im Optimierungsprozess und beschleunigt die Konvergenz des Modells [CM20][SMDH13]. Ein weiterer Parameter für das effizientere Training ist die Gewichtsabnahme. Sie ist eine Form der Regularisierung, die dazu beiträgt, die Komplexität des Modells zu kontrollieren und Overfitting zu vermeiden, indem kleine Strafen auf die Modellgewichte angewendet werden [Yin19][Haw04].

Die Batch-Größe bestimmt die Anzahl der Bilder, die das Modell verarbeitet, bevor seine Gewichte aktualisiert werden. Sie hat einen großen Einfluss auf die Konvergenz und Genauigkeit des Modells [KC20].

### 3.4.5 Trainingsablauf

Der Trainingsprozess des Mask R-CNN Modells verwendet Verlustfunktionen, um die Erkennung und Segmentierung von Objekten zu optimieren. Dazu gehören der RPN-Klassifikations- und RPN-Regressionsverlust, der Klassifikationsverlust, der Box-Regressionsverlust und der Maskenverlust [Gir15] [WMZT22]. Üblicherweise werden Stochastic Gradient Descent (SGD) oder Varianten wie Adam zum Training verwendet [iA93] [KB17]. Dabei wird das Modell auf einem Trainingsdatensatz trainiert, die Gewichte und Bias-Werte werden durch Backpropagation angepasst, um Verluste zu minimieren [RHW86].

Das Training wird fortgesetzt, bis auf einem Validierungsdatensatz keine Verbesserung mehr erzielt wird. Die Lernrate des Modells kann angepasst werden und wird häufig schrittweise verringert, um die Optimierung zu verfeinern. Insgesamt besteht der Prozess aus einem iterativen Zyklus der Parameteranpassung, um eine kontinuierliche Verbesserung der Modellleistung bei der Erkennung und Segmentierung komplexer Objekte zu erreichen.

## 3.5 Datensatz

Für viele Aufgaben des maschinellen Sehens sind bereits große Datensätze frei verfügbar, die Tausende von Bildern mit Objekten enthalten und zum Training eigener Netze verwendet werden können [Pap23b] [AI423]. Im Bereich des Nanoprobings ist derzeit jedoch kein geeigneter Datensatz verfügbar. Die Erstellung eines Datensatzes ist daher Teil dieser Arbeit. Dazu wurde das im folgenden Abschnitt beschriebene Format verwendet.

### 3.5.1 COCO Data Annotation Format

Das Common Objects in Context (COCO) Data Annotation Format ist ein weit verbreitetes Format für die Annotation von Bilddaten, das häufig für maschinelles Sehen und Deep Learning verwendet wird. COCO bietet eine umfangreiche Sammlung von Annotationen, einschließlich Objekterkennung, Segmentierung und Schlüsselpunkterkennung [COC23]. Das im Rahmen dieser Arbeit verwendete Format, welches in Abbildung 3.11 zu sehen ist, ist abgeleitet von COCO und angepasst an die spezifischen Anforderungen.

Bei der Bounding-Box-Annotation werden die Koordinaten als x- und y-Position des oberen linken Punktes sowie als Breite und Höhe der Box angegeben. Für die Segmentierung wird eine Liste von Punktkoordinaten verwendet, die den Umriss des Objekts definieren. Ein Keypoint wird durch ein Tupel von drei Werten (x, y, v) dargestellt. Dabei sind x und y die Koordinaten des Keypoints im Bild.

Der Wert v ist ein Sichtbarkeitsflag, das den Status des Keypoints angibt. Er kann drei verschiedene Werte annehmen: 0, wenn der Keypoint im Bild nicht vorhanden ist; 1, wenn der Keypoint vorhanden, aber nicht sichtbar ist; und 2, wenn der Keypoint vorhanden und sichtbar ist.

### 3.5.2 Evaluationsmetriken

Die Bewertung von Objekterkennungs-, Segmentierungs- und Schlüsselpunktvorhersagemodellen erfordert präzise Metriken, um ihre Leistung objektiv zu bewerten. Im Rahmen des COCO-Datenformats werden verschiedene Evaluationsmetriken verwendet, um die Qualität der Ergebnisse zu quantifizieren [COC23].

#### Intersection over Union

Die Metrik Intersection over Union (IoU) – auch Jaccard-Koeffizient genannt – bewertet die räumliche Übereinstimmung zwischen den vorhergesagten (A) und den tatsächlichen (B) Bounding Boxen oder Masken der Objekte.

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \quad (3.1)$$

Der IoU wird berechnet, indem die Fläche der Überlappung durch die Fläche der Vereinigung dividiert wird. Ein hoher IoU-Wert zeigt eine hohe Übereinstimmung zwischen den vorhergesagten und den tatsächlichen Bereichen an. Im Zusammenhang mit der Objekterkennung und Segmentierung wird der IoU verwendet, um zu bestimmen, ob eine vorhergesagte Bounding Box oder Maske als korrekt angesehen wird.

## Object Keypoint Similarity

Die Metrik Object Keypoint Similarity (OKS) bewertet die Vorhersage von Schlüsselpunkten anhand einer normalisierten Distanz zwischen den vorhergesagten und den tatsächlichen Schlüsselpunkten eines Objekts. Sie spielt damit die gleiche Rolle wie die IoU. Die OKS wird mit folgender Formel berechnet:

$$\text{OKS} = \frac{\sum_i \exp\left(-\frac{d_i^2}{2s^2\kappa_i^2}\right) \delta(v_i > 0)}{\sum_i \delta(v_i > 0)} \quad (3.2)$$

Dabei ist  $d_i$  der Abstand zwischen den Schlüsselpunkten,  $s$  der Objektmaßstab,  $\kappa_i$  eine Normierungskonstante und  $\delta$  die Indikatorkfunktion für sichtbare Schlüsselpunkte ( $v_i$  ist der Sichtbarkeitsflag). Die Normierungskonstante bestimmt, wie empfindlich die OKS auf Abweichungen des vorhergesagten Punktes von dem wahren Punkt reagiert. Ein hoher OKS-Wert zeigt an, dass die vorhergesagten Schlüsselpunkte gut mit den tatsächlichen Schlüsselpunkten übereinstimmen, wobei sowohl die Genauigkeit als auch die Standardabweichung berücksichtigt werden. Für eine ausführliche Erklärung der OKS wird auf die offizielle COCO Dokumentation verwiesen [COC23].

## Average Precision

Die Metrik Average Precision (AP) ist eine zentrale Evaluationsmetrik für die Bewertung von Objekterkennungsmodellen. Sie misst die Genauigkeit der Objektlokalisierung und -klassifizierung unter Berücksichtigung von False Positives (FP) und False Negatives (FN). Die AP wird aus der Fläche unter der Precision-Recall-Kurve berechnet, wobei Precision und Recall durch die folgenden Formeln definiert sind:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ &= \frac{\text{TP}}{\#\text{Vorhersagen}} \end{aligned} \quad (3.3)$$

$$\begin{aligned} \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ &= \frac{\text{TP}}{\#\text{Ground Truth}} \end{aligned} \quad (3.4)$$

Die Precision (3.3) gibt an, wie viele der vorhergesagten Objekte tatsächlich korrekt sind, während der Recall (3.4) angibt, wie viele der tatsächlich vorhandenen Objekte korrekt vorhergesagt wurden. Berechnet man diese zwei Werte für die akkumulierten Vorhersagen des Modells und zeichnet sie in einem Diagramm auf, erhält man die Precision-Recall-Kurve.

AP wird sowohl bei der Objekterkennung als auch bei der Schlüsselpunktvorhersage verwendet, um die Gesamtleistung der Modelle zu bewerten. Es gibt sie in verschiedenen Varianten. Die für diese Arbeit relevanten AP-Metriken sind in Tabelle 3.1 aufgelistet.

Metrik	Beschreibung
AP	AP at IoU=.50:.05:.95 (primary challenge metric)
$AP^{IoU=.50}$	AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$	AP at IoU=.75 (strict metric)
AP	AP at OKS=.50:.05:.95 (primary challenge metric)
$AP^{OKS=.50}$	AP at OKS=.50 (loose metric)
$AP^{OKS=.75}$	AP at OKS=.75 (strict metric)

Tabelle 3.1: Übersicht der für diese Arbeit relevanten Average Precision (AP) Metriken Quelle: COCO Dokumentation [COC23]

```

1 {                                         19 "annotations": [
2   "images": [                           20   {
3     {                                     21     "id": int,
4       "id": int,                         22     "image_id": int,
5       "width": int,                      23     "category_id": int,
6       "height": int,                     24     "segmentation": [polygon],
7       "file_name": str                  25     "area": float,
8     }                                     26     "bbox": [x, y, width,
9   ],                                     27       height],
10  "categories": [                        28     "iscrowd": 0 or 1,
11    {                                     29     "keypoints": [x1,y1,v1,...],
12      "id": int,                         30     "num_keypoints": int,
13      "name": str,                       31   }
14      "supercategory": str,              32 }
15      "keypoints": [str],               ]
16      "skeleton": [edge]                ]
17   }                                     ]
18 ] ,                                     ]

```

Abbildung 3.11: Das in dieser Arbeit verwendete Annotation Format, abgespeichert als JSON Datei. Datensätze in COCO Format können mit den meisten Frameworks nativ zum Training Neuronaler Netze verwendet werden.

# 4 Verwandte Werke

## 4.1 Detektion und Segmentierung von Mitochondrien in Rasterelektronenmikroskop Bildern

In der Arbeit „Automatic Detection and Segmentation of Mitochondria from SEM Images using Deep Neural Network“ widmen sich Liu *et al.* der Herausforderung, Mitochondrien in hochauflösenden REM-Bildern automatisch zu erkennen und zu segmentieren [LLX<sup>+</sup>18]. Sie stellen fest, dass trotz erheblicher Fortschritte die Komplexität der Zellstrukturen, das Hintergrundrauschen und Bildartefakte diese Aufgabe nach wie vor erschweren. Um dieses Problem zu lösen, schlagen die Autoren einen verbesserten Mask R-CNN Ansatz vor. Dieses Modell ist eine Weiterentwicklung der von He *et al.* [HGDG17] vorgestellten Mask R-CNN Architektur und beinhaltet neben der Erkennung und Segmentierung von Mitochondrien auch eine morphologische Verarbeitung und die Einbeziehung von Kontextinformationen zur Korrektur lokaler Fehleinschätzungen. Ein besonderes Merkmal dieses Ansatzes ist die Nutzung des Field of View (FoV) der Maskenverzweigung zur Erzeugung mehrerer Maskenausgaben in vier Richtungen, die für die abschließende Segmentierung kombiniert werden.

Die Validierungsergebnisse, basierend auf zwei gängigen Datensätzen, zeigen, dass der vorgeschlagene Ansatz eine vergleichbare Leistung wie die bisher besten Methoden erreicht. Darüber hinaus konnten Mitochondrien verschiedener Größen und Maßstäbe erfolgreich detektiert und segmentiert werden, wobei geeignete Nachbearbeitungsverfahren eine deutliche Verbesserung der Segmentierungsleistung ermöglichten. Die vorgestellte Methode in der Arbeit unterstreicht die Effektivität von Deep Learning-Modellen, wie Mask R-CNN, bei der Bewältigung der Herausforderungen der Segmentierung in hochauflösenden REM-Bildern. Durch die gezielte Anpassung und Optimierung dieses Modells für die spezifischen Aufgaben der Mitochondrien-Erkennung konnten die Autoren bemerkenswerte Ergebnisse erzielen. Diese Ergebnisse lassen darauf schließen, dass ähnliche Anpassungen des Modells es ebenso ermöglichen könnten, die Lokalisierung und Identifizierung von Messpitzen in REM-Bildern zu automatisieren und zu verbessern. Daher könnte die hier dargelegte Vorgehensweise einen vielversprechenden Ausgangspunkt für die Anpassung von Deep Learning-Modellen an die spezifischen Anforderungen der Messpitzen-Lokalisierung bieten.

## 4.2 FibeR-CNN

Mit der Arbeit „FibeR-CNN: Expanding Mask R-CNN to Improve Image-Based Fiber Analysis“ leisten Frei und Kruis einen wichtigen Beitrag zur Herausforderung der automatisierten Bildanalyse von faserförmigen Materialien in REM-Bildern. Sie identifizieren einen Mangel an effektiven automatisierten Algorithmen zur Bildannotation von überlappenden und verdeckten Fasern, die eine genaue Bestimmung der Faserlänge und -breite ermöglichen [FK21]. Beste-hende Methoden wie CTFIRE sind nicht nur unzureichend, sondern auch sehr zeitaufwendig. Um dieses Problem zu lösen, erweitern die Autoren die Mask R-CNN Architektur innerhalb des Detectron2 Frameworks und entwickeln einen spezifischen Ansatz für die Analyse von Faserbildern, den sie FibeR-CNN nennen [WKM<sup>+</sup>19][Fre22].

FibeR-CNN wurde so modifiziert, dass es zusätzlich zur Segmentierung auch die Breite und Länge der Fasern vorhersagen kann. Durch die Verwendung von CNNs erweist sich der Ansatz als robust gegenüber Änderungen der Bildbedingungen und benötigt nach dem Training keine Parameteranpassung durch den Benutzer. Somit ist es ein effektives Werkzeug für die automatische, bildbasierte Faserformanalyse. Obwohl FibeR-CNN nicht direkt als Grundlage für diese Arbeit dient, bietet es wertvolle Einblicke, wie Mask R-CNN für spezifische Anforderungen in der REM-Bildsegmentierung modifiziert werden kann. Darüber hinaus bestätigt es, dass Mask R-CNN effektiv zur Segmentierung komplexer Strukturen in REM-Bildern eingesetzt werden kann, was Parallelen zur Aufgabe der Spitzenerkennung aufweist.

## 4.3 Detectron2

Detectron2 ist ein Open-Source-Software-Framework, das von der Facebook AI Research Group entwickelt wurde. Es bietet eine flexible, modulare und erweiterbare Architektur, die eine einfache Anpassung und Erweiterung verschiedener Modelle ermöglicht und die Entwicklung und das Experimentieren mit neuen Modellen und Algorithmen erleichtert. Detectron2 enthält Implementierungen der neuesten Objekterkennungs- und Segmentierungsalgorithmen, einschließlich Mask R-CNN und Faster R-CNN, und hat den Vorteil, dass es eine Reihe von vortrainierten Modellen bietet, die auf großen und unterschiedlichen Datensätzen wie COCO trainiert wurden. Durch die Verwendung von Detectron2 kann das Modell von diesen vortrainierten Gewichten profitieren, was den Trainingsprozess beschleunigt und die Modellleistung verbessert. Die in Detectron2 integrierten Werkzeuge für Training, Inferenz und Evaluierung sowie die umfangreichen Datenverarbeitungsfunktionen machen Detectron2 zu einem umfassenden Werkzeug, das den gesamten Prozess der Modellerstellung und -evaluierung erleichtert. Detectron2 bietet eine nahtlose Integration und Interoperabilität mit dem PyTorch Ökosystem [WKM<sup>+</sup>19].

# 5 Durchführung

Die Erstellung eines Datensatzes für das Nanoprobing stellt eine Herausforderung dar, da eine große Anzahl von Bildern aus verschiedener Szenen erforderlich ist. Die manuelle Bewegung der Manipulatoren sowie die Steuerung des REM ist zeitaufwändig und erfordert viel Geschick, was die Erstellung eines umfangreichen Datensatzes erschwert. Daher wurde eine Lösung benötigt, um die Bewegung der Manipulatoren und die Steuerung des REM zu automatisieren. Verschiedene Ansätze wurden in Betracht gezogen, einschließlich der Verwendung der offiziell bereitgestellten Programme. Letztendlich wurde jedoch die Implementierung der Schnittstellen in Python als beste Lösung identifiziert. Diese Schnittstellen ermöglichen die Fernsteuerung der Manipulatoren und des REM mittels eigens entwickelter Skripte, was die Erstellung eines umfangreichen Datensatzes erheblich erleichtert. Um diese Daten effizient zu nutzen, werden verschiedene vorgeführte Varianten des Mask R-CNN Modells mit dem Detectron2 Framework trainiert und evaluiert.

In den folgenden Abschnitten dieses Kapitels wird die Umsetzung der verschiedenen Komponenten dieser Arbeit, einschließlich der Datenerfassung und des Modelltrainingsverfahrens, detailliert beschrieben.

## 5.1 Implementierung der Nanocontrol Schnittstelle in Python

Da es sich beim Nanocontrol um ein COM-Gerät handelt, wird zur Steuerung des Manipulators, auf dem die Messspitzen montiert sind, eine spezielle Schnittstelle in Python implementiert. Hierzu wird das Paket pySerial verwendet.

Die Kommunikation mit dem Nanocontrol über die COM-Schnittstelle basiert auf einem speziellen Befehls- und Antwortformat. Die Befehle, die an das Nanocontrol gesendet werden, folgen dem Format **<command string><blank><param><CR>**, während die Antworten der Form **<status char><tab><message string><CR>** haben. Die Befehle decken eine Vielzahl von Funktionen ab, von der Abfrage von Informationen wie dem Systemstatus oder der Position des Manipulators bis hin zu spezifischen Befehlen für die Bewegung des Manipulators.

Command	Output	Description
coarse A +100	o<tab><CR>	Executes 100 positive coarse steps in channel A.
coarse A ?	o<tab>100<CR>	Returns value of coarse step counter for channel A.
fine C -10	o<tab><CR>	Sets the fine position of channel C to -10.
fine ?	o<tab>0 0 -10 0<CR>	Returns fine position A to D separated by blanks.

Abbildung 5.1: Befehle, um die Position der Manipulatoren im Grob- und Feinbereich auszulesen und zu ändern. Auszug dem Nanocontrol Software Manual. Quelle: Kleindiek Nanotechnik GmbH

Um die Bedienung des Manipulators zu vereinfachen und zu standardisieren, wird eine dedizierte Klasse in Python erstellt, die das Nanocontrol repräsentiert. Die Implementierung dieser Klasse kann Abbildung 5.2 entnommen werden. Innerhalb dieser Klasse sind alle für den Manipulator verfügbaren Befehle in Python-Funktionen eingebettet. Diese Funktionen sind mit entsprechenden Tests versehen, um die korrekte Eingabe zu gewährleisten. Dies stellt eine einfache und sichere Bedienung des Manipulators über den Python-Code sicher und trägt zur Robustheit des Gesamtsystems bei.

Die Controller-Klasse – ebenfalls in Abbildung 5.2 dargestellt ist – ist für die Erzeugung von Instanzen aller angeschlossenen Nanocontrols verantwortlich. Zur Erzeugung des Datensatzes werden acht Manipulatoren und ein Tisch verwendet. Die automatisierte Ansteuerung wird ebenfalls über den Controller realisiert.

Die Implementierung der Schnittstelle des Nanocontrols in Python bietet eine flexible und anpassbare Lösung zur Steuerung der Manipulatoren durch speziell entwickelte Skripte. Dies ist besonders nützlich, um systematisch eine Vielzahl von Szenarien und Bedingungen für die Erstellung des Datensatzes abzudecken.

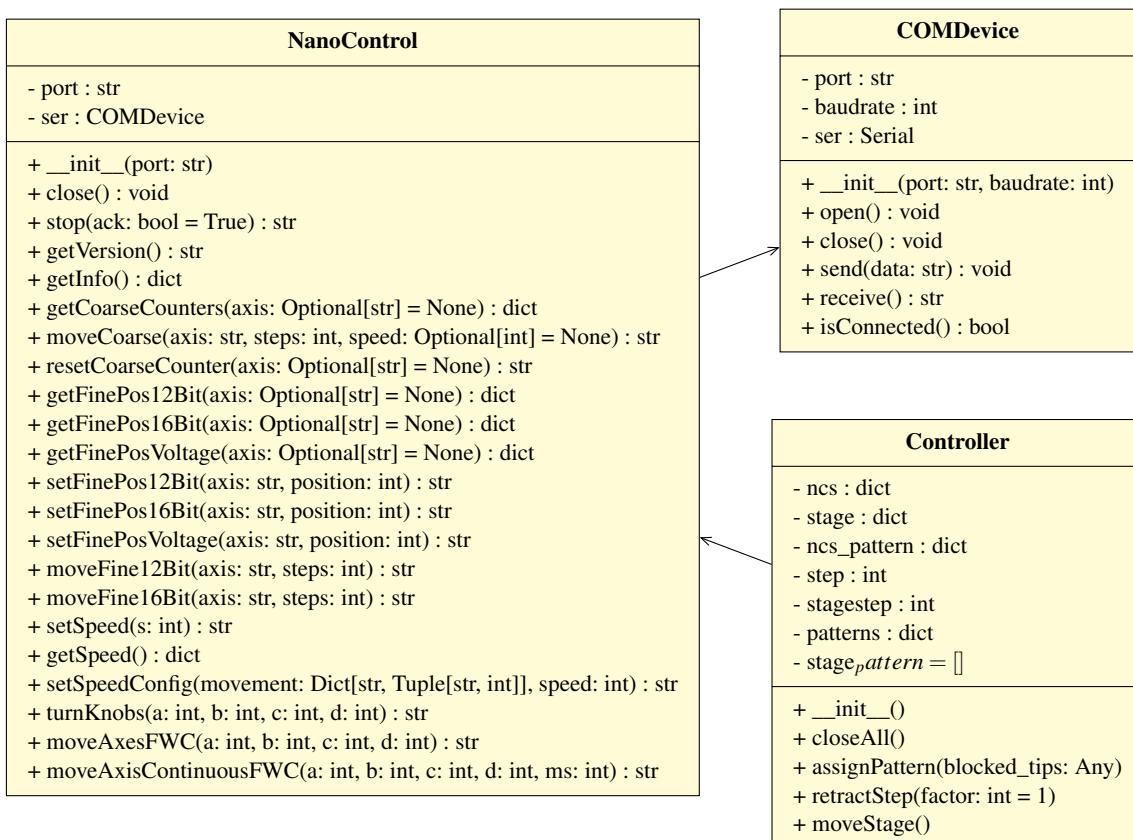


Abbildung 5.2: Die Implementierung der Schnittstelle des Nanocontrols in Python ermöglicht eine flexible Steuerung der Manipulatoren.

## 5.2 Implementierung der GeminiSEM API in Python

ZEISS stellt Entwicklern eine API zur Verfügung, um das REM in eigenentwickelten Programmen zu steuern. Zur Verwendung der bereitgestellten ocx-Datei, die unter Windows ein ActiveX-Steuerelement bereitstellt, wird die Python-Bibliothek pywin32 verwendet. Die Nutzung der API in Python wird von ZEISS beispielhaft in dem GeminiSEM API Handbuch dargestellt und wird für die Implementierung die Klasse „Sem“ genutzt. Sie bietet Funktionen zum Setzen und Auslesen der Parameter, die im Rahmen dieser Arbeit angepasst werden müssen, und enthält Funktionen wie „grabFullImage(...)“, „grabMask()“ und „grabImageWithParameters(...)“, die für eine effiziente Bildaufnahme zur Erstellung des Datensatzes mit verschiedenen REM-Parametern entwickelt wurden. In Kapitel 5.3.2 wird näher auf die in dieser Arbeit verwendeten Parameter eingegangen. Es ist stets darauf zu achten, dass das REM ein vollständiges Bild aufnimmt. Dies wird sichergestellt, indem die Aufnahmedauer mit der Funktion `getFrameTimeInSeconds()` ausgelesen und berechnet wird. Eine weitere wichtige Funktion ist das Rücksetzen des REM in den Ausgangszustand. Dies ist wichtig, da einige Parameter gleichzeitig geändert werden und ein späteres manuelles Zurücksetzen viel Zeit in Anspruch nehmen würde. Das korrekte Rücksetzen des REM wird durch die Funktionen „`getInitialParameters()`“ und „`restoreInitialParameters()`“ sichergestellt.

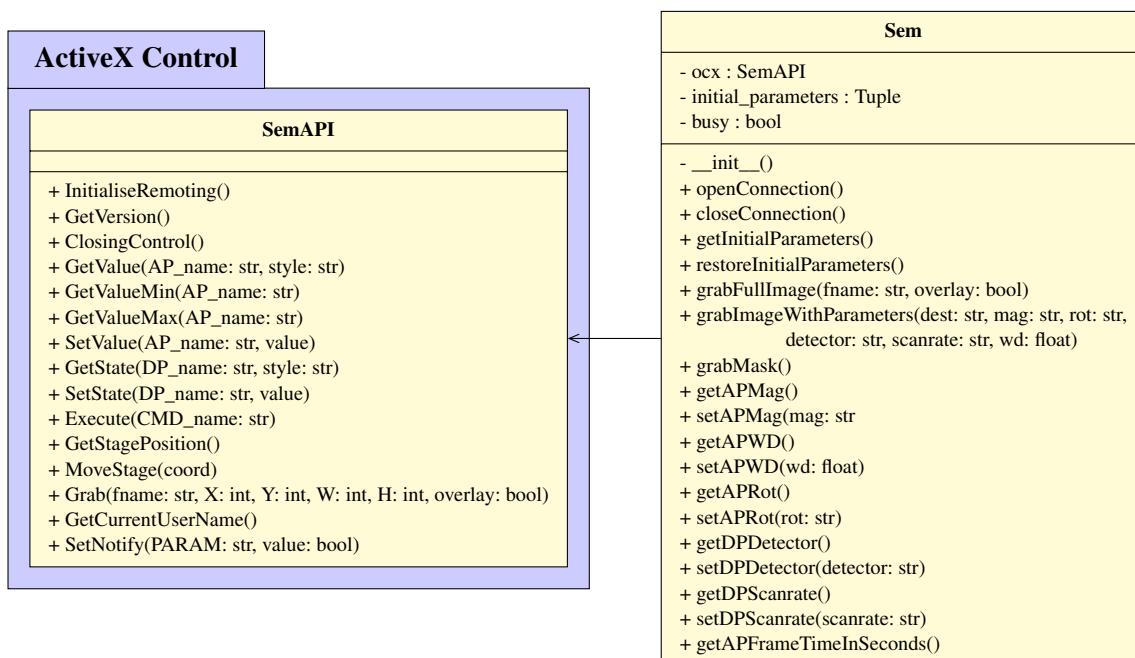


Abbildung 5.3: Die SemAPI wird von ZEISS in Form einer ocx-Datei zur Verfügung gestellt. Die Sem-Klasse bietet eine abstrahierte und angepasste Form der API, die speziell für die effiziente Erstellung von Datensätzen geeignet ist.

## 5.3 Aufnahme der Bilddaten

Die spezifischen Methoden und Techniken, die zur automatisierten Erfassung der Bilder für den Datensatz entwickelt wurden, werden im folgenden Abschnitt detailliert beschrieben.

Der für dieses Projekt entwickelte Datensatz besteht aus zwei Teilen. Der erste Teil umfasst 150 Bilder aus realen Einsätzen, die bereits vorhanden sind und lediglich annotiert werden

müssen. Diese Bilder bestehen hauptsächlich aus Aufnahmen, bei denen die Spitzen eine Probe kontaktieren. Dies ist die Zielposition der Spitzen und erfordert die höchste Genauigkeit, weshalb hier auf eine Bildaugmentation verzichtet wird und nur Aufnahmen aus realen Szenarien verwendet werden. Der zweite Teil besteht aus 600 Bildern und 100 zugehörigen Masken, die für die Erstellung des Datensatzes aufgenommen wurden. Der Begriff „Masken“ bezieht sich in diesem Zusammenhang auf Bilder, die später speziell eingefärbt werden, um die genaue Position und Form der Nadeln in den Datensätzen zu markieren. Auf diese Weise wird eine direkte Beziehung zwischen den Bildern und den Positionen der Nadeln hergestellt, was für das Training des Modells von entscheidender Bedeutung ist. Die Masken dienen während des Trainingsprozesses als „Wahrheitsquelle“ und ermöglichen es dem Modell zu lernen, wie Nadeln in den Bildern erkannt und lokalisiert werden können.

Zur Erzeugung der Bilder wird eine Art Bildaugmentation entwickelt. Dabei werden die Parameter des Mikroskops genutzt, um Rauschen und Unschärfe einzubringen und die Lichtverhältnisse zu variieren. Durch Veränderung der Probenposition wird der Hintergrund variiert. So kann ein und dieselbe Szene auf unterschiedliche Weise aufgenommen werden. Die Erstellung eines umfassenden Datensatzes wird dadurch beschleunigt. Für die Aufnahme der Masken sind die Parameter so gewählt, dass die Bilder einen starken Kontrast, scharfe Kanten und ein hohes Signal-Rausch-Verhältnis aufweisen. Dies erleichtert die spätere Annotation, da die Konturen der Spitzen leicht und akkurat identifiziert werden können.

### 5.3.1 Variation der aufzunehmenden Szene

Die Anordnung der Spitzen im Sichtfeld des REM und die Vielfalt der Szenen, die sie erzeugen, spielen eine zentrale Rolle bei der Erstellung des Datensatzes. Zunächst ist zu beachten, dass in der Regel acht Nadeln auf dem Prober Shuttle montiert sind, aber nur ein Teil davon tatsächlich im Sichtfeld des REM liegt. Die Nadeln können sich kreuzen und Teile einer Nadel verdecken, oder es kann nur ein sehr kleiner Teil in das Bild ragen, was die Erkennung erschwert.

Der Vergrößerungsfaktor hat, wie in Abbildung 5.4 zu sehen ist, ebenfalls einen großen Einfluss auf die Darstellung der Spitzen. Durch die Aufnahme von Bildern bei verschiede-

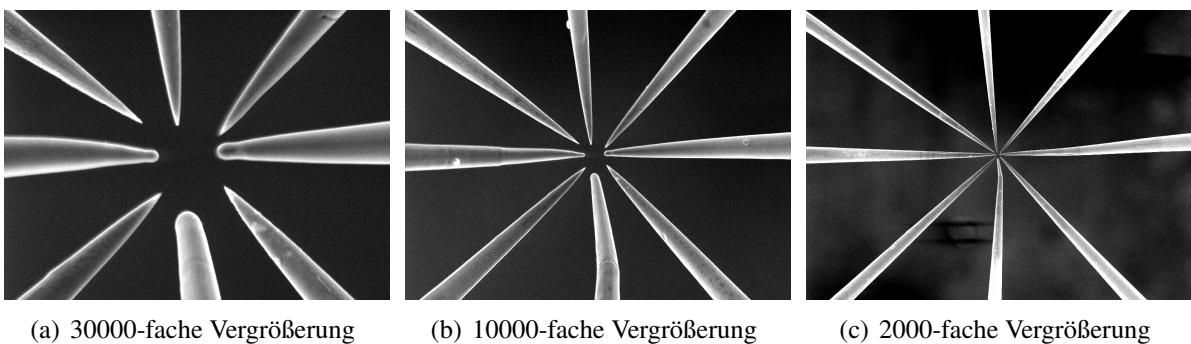


Abbildung 5.4: Messspitzen, dargestellt bei unterschiedlichen Vergrößerungen. Bei einer hohen Vergrößerung sind die Spitzen klar erkennbar. Je kleiner die Vergrößerung, desto schwerer ist es, den Apex der Spitze zu lokalisieren.

nen Vergrößerungen, wird sichergestellt, dass das trainierte Modell später in der Lage ist, die Messspitzen unabhängig vom Vergrößerungsfaktor zu erkennen und zu lokalisieren.

Ein weiterer wichtiger Aspekt ist die Untergrundvariation. Hierbei handelt es sich um die strategische Verschiebung der Probe im REM. Je nach Position und Ausrichtung der Probe können unterschiedliche Strukturen wie SRAM-Zellen, Metallisierungsebenen, Transistoren oder

andere mikro- und nanoskalige Strukturen, die typischerweise in einem Halbleiterchip vorkommen, im Hintergrund erscheinen und das Gesamtbild erheblich beeinflussen. Durch diese Variation des Hintergrunds wird das trainierte Modell widerstandsfähiger gegen Hintergrundstrukturen, die den Nadeln ähnlich sehen. Dies ist ein wichtiger Schritt, um sicherzustellen, dass das später trainierte Modell eine niedrige FP-Rate aufweist. Schließlich ist es wichtig, dass eine Vielzahl von Positionen, an denen sich die Spitzen befinden, im Datensatz vertreten sind. Dies wird durch das Anfahren verschiedener Positionen mit den Spitzen erreicht.

Durch Berücksichtigung der genannten Faktoren, wird sichergestellt, dass der Datensatz eine große Vielfalt realer Bedingungen widerspiegelt, unter denen das Modell funktionieren soll.

### 5.3.2 Bildaugmentierung durch REM-Parameter

Die Augmentierung der Trainingsdaten spielt eine wichtige Rolle beim maschinellen Lernen und wird verwendet, um die Menge der Trainingsdaten zu erhöhen und das Modell robuster gegenüber Änderungen der Eingangsdaten zu machen. Üblicherweise werden in der Bildverarbeitung Techniken wie Drehen, Beschneiden, Verzerren oder Verrauschung verwendet. In dieser Arbeit wird jedoch ein alternativer Ansatz zur Augmentierung gewählt. Die Augmentierung der Bilder durch Variation der REM-Parameter anstelle einer Nachbearbeitung der Bilder hat mehrere Vorteile.

Erstens ermöglicht diese Methode ein realistischeres Verrauschen der Bilder. Das Rauschen in REM-Bildern hat eine besondere Charakteristik, die durch nachträgliche Rauschaddition nicht einfach reproduziert werden kann. Zweitens spielt die Variation der Unschärfe eine wichtige Rolle. Da die Spitzen sich meist auf unterschiedlichen Höhen befinden, wirkt sich die Unschärfe nicht gleichmäßig auf das gesamte Bild aus. Bei einer nachträglichen Bearbeitung würde das gesamte Bild unscharf erscheinen, was nicht der Realität entspricht. Drittens hat die Variation des Kontrasts durch unterschiedliche Sekundärelektronendetektoren einen erheblichen Einfluss auf das Bild. Eine einfache Kontraständerung in der Nachbearbeitung würde diese Effekte nicht berücksichtigen.

Zusammenfassend lässt sich sagen, dass die Variation der REM-Parameter während der Bildaufnahme eine realistischere und anwendungsspezifische Augmentierung ermöglicht, die den Anforderungen besser entspricht. Im Folgenden werden die einzelnen Parameter und ihre Auswirkungen näher erläutert.

Zunächst spielt die Vergrößerung eine zentrale Rolle. Sie bestimmt den beobachteten Bereich und beeinflusst somit die Darstellung der Spitzen und der Probe. Wie in Abbildung 5.4 zu sehen ist, sind die Spitzen bei niedriger Vergrößerung fast über ihre gesamte Länge sichtbar, sie erscheinen sehr dünn und lang, und der vordere Punkt der Spitze ist schwer zu lokalisieren. Bei hoher Vergrößerung hingegen nehmen die Spitzen einen großen Teil des Bildes ein und die feinen Strukturen der Probe, wie zum Beispiel einzelne Transistoren, werden sichtbar. Aus diesem Grund werden für die Erstellung des Datensatzes verschiedene Vergrößerungsstufen verwendet, nämlich 50000-, 25000-, 10000- und 2000-fache Vergrößerung.

Ein weiterer wichtiger Parameter ist die Arbeitsdistanz. Sie beeinflusst die Bildschärfe und stellt oft eine Herausforderung bei der richtigen Einstellung dar. Um eine Variation in der Bildschärfe zu erzeugen und damit das Modell auf unterschiedliche Schärfegrade zu trainieren, werden Bilder entweder bei exakt eingestelltem Arbeitsabstand – wenn die Spitzen im Fokus sind – oder bei einem um 0,5 Prozent reduzierten Arbeitsabstand aufgenommen. Dadurch soll das mit den Daten trainierte Modell später in der Lage sein, auch unscharfe Spitzen korrekt zu lokalisieren.

Bezeichnung	Beschreibung	Einheit
AP_MAG	Vergrößerung	-
AP_WD	Arbeitsdistanz	m
AP_SCANROTATION	Rotation der Abtastrichtung	°
AP_FRAME_TIME	Zykluszeit eines Bildes	ms

Tabelle 5.1: Auszug aus den genutzten analogen Parametern des ZEISS GeminiSEM.

Um sicherzustellen, dass der Datensatz auch stark verrauschte Bilder enthält, werden verschiedene Abtastraten verwendet. Die Abtastrate des REM bestimmt, wie schnell die Szene mit dem Elektronenstrahl abgetastet wird. Dies beeinflusst das Signal-Rausch-Verhältnis des Bildes. Eine niedrige Abtastrate führt zu einem geringeren Signal-Rausch-Verhältnis, liefert aber bis zu 5 Bilder pro Sekunde, während bei höheren Abtastraten die Aufnahme eines Bildes bis zu mehreren Minuten dauern kann, dafür aber von besserer Qualität ist. Die Bilder für den Datensatz werden mit den Abtastraten 2 und 6 aufgenommen. Diese entsprechen den in der Praxis am häufigsten verwendeten Abtastraten bei Kleindiek Nanotechnik. Dadurch werden sowohl verrauschte als auch gut aufgelöste Bilder erzeugt.

Schließlich haben die verwendeten Detektoren des REM, InLens und SE2 einen großen Einfluss auf die Helligkeit und den Kontrast der Bilder. Die Verwendung unterschiedlicher Detektoren ermöglicht die Aufnahme von Bildern unter verschiedenen Beleuchtungs- und Kontrastbedingungen. Zur Erstellung der Masken wird der SE2 Detektor genutzt. Bei einer langsamen Abtastrate von 10, welche ungefähr einem Bild pro Minute entspricht, liefert dieser Bilder, auf denen die Konturen der Nadeln deutlich erkennbar sind.

Eine Rotation der Abtastrichtung wurde anfangs in Betracht gezogen, wurde aber nach einigen Tests wieder verworfen, da die Änderung der anderen Parameter bereits ausreicht, um eine Szene ausreichend abzubilden. Es wurde festgestellt, dass eine Rotation der Bilder durch das REM keinen zusätzlichen Nutzen bringt. Sie würde lediglich dazu führen, dass zu viele Bilder der gleichen Szene im Datensatz enthalten sind. Außerdem kann eine Rotation nachträglich auf die Bilder angewendet werden, wenn dies gewünscht wird.

Bezeichnung	Beschreibung	Wert	Text
DP_SCAN_ROT	Rotation der Abtastung	0	Aus
		1	An
DP_FREEZE_ON	Abtastung einfrieren	0	Ende des Bildes
		1	Ende der Zeile
		2	Nach Befehl
DP_SCANRATE	Abtastungsrate	0-21	0-21
DP_FROZEN	Abtastung	0	Live
		1	Eingefroren
DP_DETECTOR_TYPE	Aktiver Detektor	0-37	2=SE2, 9=InLens

Tabelle 5.2: Auszug aus den genutzten digitalen Parametern des ZEISS GeminiSEM.

### 5.3.3 Automatisierung

Bei der automatisierten Aufnahme wird der Abtastvorgang in umgekehrter Reihenfolge durchgeführt. Das bedeutet, dass die Nadeln von ihrer Endposition, die einem Radius von  $1\text{ }\mu\text{m}$  entspricht, bis zu einem Radius von  $50\text{ }\mu\text{m}$  zurückgezogen werden. Um Beschädigungen zu vermeiden, muss sichergestellt werden, dass sich die Spitzen während des Rückzugs nicht gegenseitig berühren. Dies wird durch die Verwendung vordefinierter Bahnen erreicht. Bei acht montierten Spitzen ergibt sich ein Bereich von 45 Grad, in dem die Spitzen ohne Kollisionsgefahr zurückgezogen werden können. Die für diesen Prozess verwendeten spezifischen Pfade sind in Abbildung 5.5 dargestellt.

Nach manueller Positionierung der Spitzen in der Startposition für die Aufnahme (Radius von  $1\text{ }\mu\text{m}$ ) und Einstellung beider Detektoren auf ein scharfes Bild startet die speziell entwickelte Routine. Dargestellt als Pseudocode wird diese Routine in Algorithmus 1.

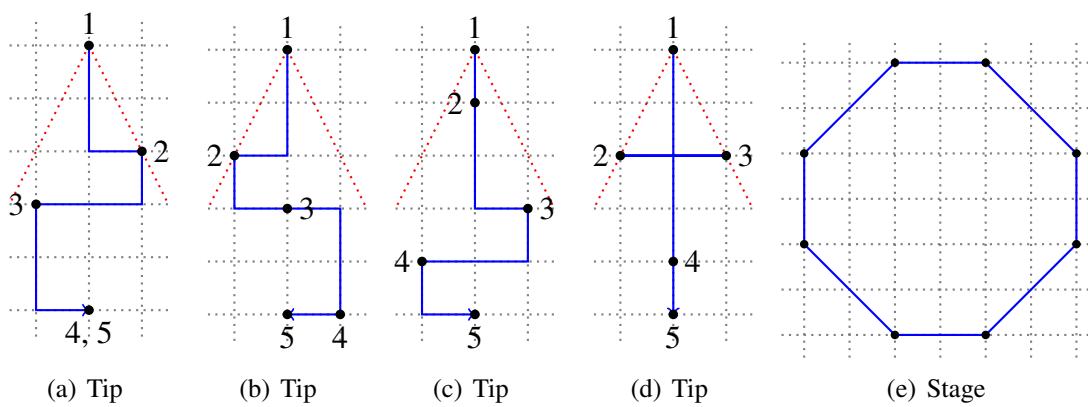


Abbildung 5.5: Vier vordefinierte Pfade für das Zurückziehen der Messspitzen. Der Verfahrensweg wird mit der Vergrößerung skaliert. Die Probe wird zyklisch verfahren.

Zunächst wird jedem Manipulator ein zufälliger Pfad zugewiesen, den er abfahren wird. Anschließend wird eine Maske eingezogen, die später für die Annotation verwendet wird. Das System nimmt nun eine vordefinierte Anzahl von Bildern der eingestellten Szene auf. Für jedes Bild wird eine zufällige Konstellation von REM-Parametern aus einem vordefinierten Satz ausgewählt. Die Probe wird zwischen jedem Bild an eine andere Position bewegt. Dazu ist eine zyklische Bahn definiert und die Probe wird so ausgerichtet, dass sie auf dem Pfad eine möglichst große Variation der Probenstruktur abdeckt. Da die Spitzen während der Aufnahme der fünf Bilder ihre Position beibehalten, kann die zuvor erstellte Maske zur Markierung der Spitzenposition für alle Bilder der Szene verwendet werden.

Dadurch reduziert sich der Aufwand für die nachfolgende Annotation auf ein Fünftel. Dieser Vorgang wird wiederholt, bis alle Spitzen die Endposition der ihnen zugewiesenen Pfade erreicht haben. Dann wird die nächst kleinere Vergrößerungsstufe (beginnend bei  $50.000x$ , dann  $25.000x$ ,  $10.000x$  und zuletzt  $2.000x$ ) verwendet und der Vorgang wiederholt.

Es ist wichtig, die zeitlichen Anforderungen dieses Prozesses zu berücksichtigen. Im Vorfeld wird viel Zeit darauf verwendet, die Nadeln in ihre Ausgangsposition zu bringen und die Parameter für ein scharfes Bild einzustellen. Beide Prozesse erfordern eine Reihe fein abgestimmter Einstellungen, die alle miteinander interagieren. Außerdem dauert ein einzelner Durchlauf der Routine etwa eine Stunde. Trotz dieser Herausforderungen stellt die automatisierte Bildaufnahme einen effizienten Ansatz dar, um eine große Anzahl von Bildern mit variablen Parametern zu erzeugen.

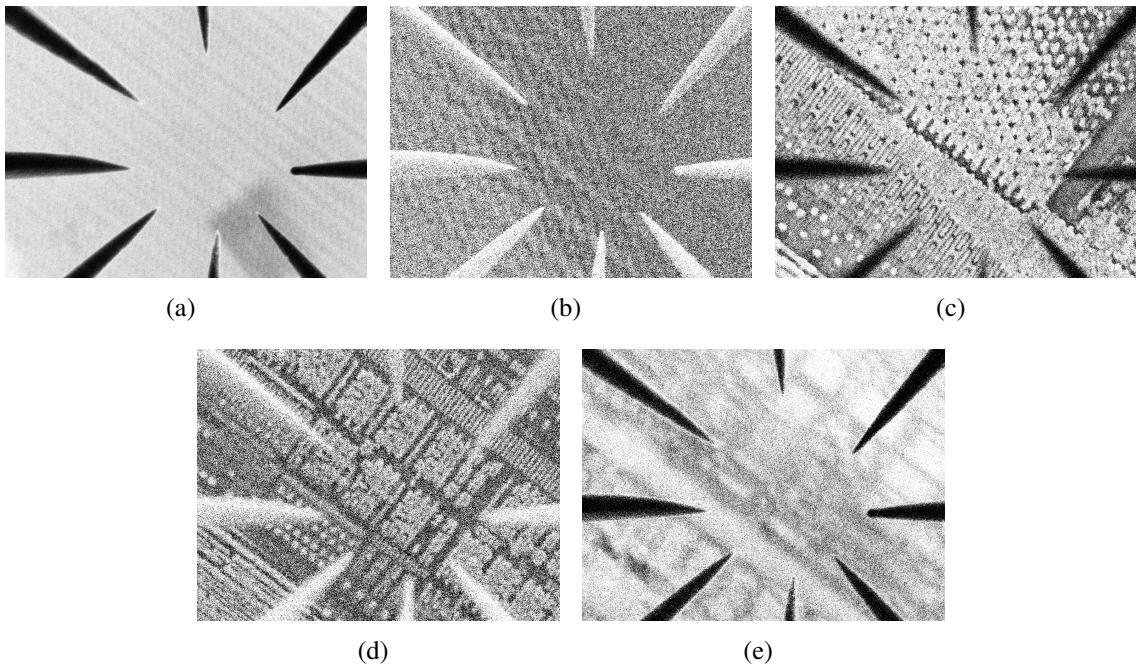


Abbildung 5.6: Eine Szene aufgenommen mit verschiedenen Parametern. Deutlich zu erkennen ist der Unterschied zwischen dem SE2 Detektor in (a), (c) und (e) und dem InLens Detektor in (b) und (d). Die Probenstruktur ist sehr unterschiedlich.

---

**Algorithmus 1** Routine für das automatische Einziehen der Bilder.

---

**Erfordert:** Liste von Vergrößerungen: mags, Anzahl an Bildern pro Szene: count, Verwendete Abtastraten: srs, Verwendete Detektoren: dets, Zwei Arbeitsdistanzen: wds

**Liefert:** Einen Datensatz an Bildern

```

1: for mag in mags do
2:   controller.assignPattern() {Weise jedem Manipulator einen zufälligen Pfad zu}
3:   for i ← 0 to controller.patternLength() do
4:     sem.grabImage(mag, dt='SE2', sr='10', wds[0]) {Ziehe ein Maskenbild ein.}
5:     for j ← 0 to count do
6:       wd, sr, dt ← random(wds, srs, dts) {Wähle zufällige Mikroskop Parameter}
7:       sem.grabImage(mag, dt, sr, wd) {Ziehe ein Bild ein.}
8:       controller.moveStage() {Bewege die Probe.}
9:     end for
10:    controller.retractStep() {Bewege die Manipulatoren}
11:  end for
12: end for
```

---

### 5.3.4 Ergebnis

Bei einem Durchlauf der Routine, mit vier Vergrößerungsstufen, fünf Spitzenpositionen und fünf Bildern pro Szene, werden insgesamt 100 Bilder und 20 Masken erzeugt. Diese Routine wird sechsmal durchgeführt, jeweils mit verschiedenen Ausgangspositionen der Spitzen und verschiedenen Positionen der Probe. Somit ergeben sich insgesamt 600 Bilder und 100 zugehörige Masken.

## 5.4 Daten Annotation

### 5.4.1 Segmentierung

Bei der Erstellung der Annotation der Masken, wird bewusst auf polygonbasierte Werkzeuge verzichtet, da diese eine pixelgenaue Segmentierung erschweren. Daher wird Adobe Photoshop [Ado] als Werkzeug der Wahl eingesetzt. Photoshop bietet eine Reihe von Werkzeugen, wie das „Magnetic Lasso“ und „Quick Select“, die eine pixelgenaue Auswahl von Objekten erleichtern.

Um die Annotation zu vereinfachen und zu standardisieren, werden acht optisch und numerisch deutlich unterscheidbare Farben zur Markierung der Spitzen verwendet. Dabei wird darauf geachtet, dass den Spitzen aus einer bestimmten Himmelsrichtung immer die gleiche Farbe zugewiesen wird. Dies ermöglicht nicht nur die Unterscheidung der Instanzen, sondern gibt unter Berücksichtigung der Bildrotation auch Auskunft darüber, in welchem Manipulator die Spalte montiert ist. Nachdem alle Spitzen markiert sind, wird der restliche Bildbereich schwarz eingefärbt.

Um die Qualität der Annotationen sicherzustellen, wird mit Experten von Kleindiek definiert, welche Bereiche zu den Spitzen gehören und wie die Konturen zu zeichnen sind.

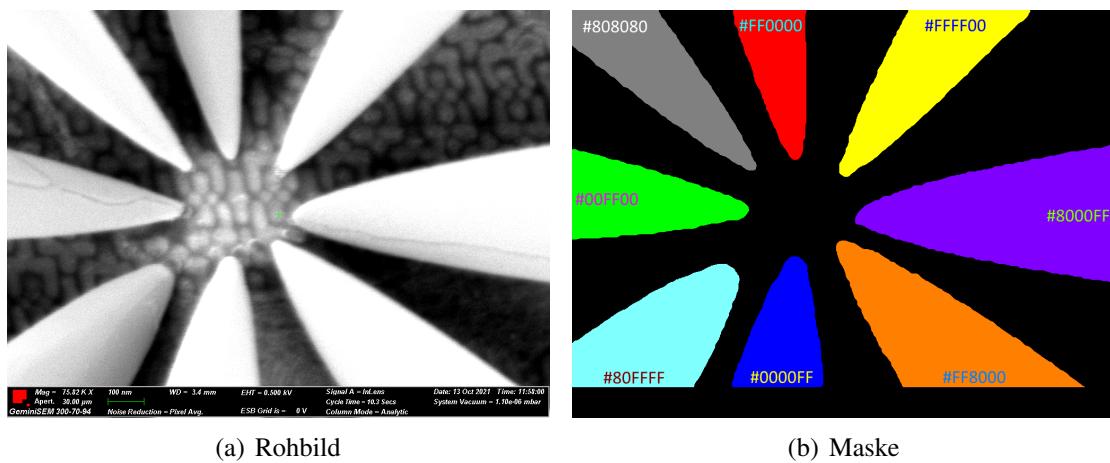


Abbildung 5.7: Bild und zugehörige annotierte Maske. Die HEX-Farbwerde sind hier in die Spitzen eingezeichnet.

Da Photoshop ursprünglich für die Bearbeitung und Verbesserung von Fotografien entwickelt wurde, treten trotz pixelgenauer Auswahl Probleme auf. Denn Photoshop erzeugt keine harten Kanten. Die Farben einiger Grenzpixel vermischen sich leicht. Dies ist problematisch, da dadurch in den Masken Farben auftreten, die nicht den vordefinierten Farben entsprechen und somit zu Fehlern in der Weiterverarbeitung führen.

Versuche, dieses Problem durch nachträgliche Manipulation von Kontrast, Helligkeit und Sättigung zu beheben, waren nicht erfolgreich. Daher wurde ein Python-Skript zur Nachbearbeitung der annotierten Masken entwickelt. Genutzt werden dazu die Python Bibliotheken von SciPy [VGO+20], Pillow [Cla15] und OpenCV [Bra00].

Da für die Annotation numerisch gut unterscheidbare Farben gewählt wurden und da ein Pixel – definiert durch das Tupel (r, g, b) – auch als Vektor im dreidimensionalen Raum interpretiert werden kann, lässt sich für jedes Pixel, das von den acht definierten Farben abweicht, die Manhattan-Distanz berechnen und der Wert des Pixels auf den nächstgelegenen Wert setzen. Auf diese Weise konnten alle Pixelfehler ohne Qualitätsverlust der Annotation beseitigt werden.

### 5.4.2 Konvertierung in das COCO Datenformat

Mehrere Schritte sind erforderlich, um die erstellten Bilder und annotierten Masken in das COCO-Datenformat zu konvertieren.

Zunächst muss der Datensatz in einen Trainings- und einen Validierungsteil aufgeteilt werden. Zu diesem Zweck werden zwei JSON-Dateien, `train.json` und `val.json`, erstellt. In diese Dateien müssen nun alle Klassen, Bilder und Annotationen eingetragen werden.

Um die erstellten Masken in das COCO Format zu übersetzen, müssen zunächst die eingefärbten Spitzen als Polygon dargestellt werden. Dazu werden die Umrisse der eingefärbten Bereiche extrahiert. Zweitens müssen die Bounding-Box-Werte aus den Grenzen des Polygons bestimmt werden. Drittens muss der vorderste Punkt der Spalte bestimmt werden, um den Keypoint zu markieren. Um diese Aufgaben zu lösen, wird ein Ansatz adaptiert, der von Chris Eijgenstein in seinem Projekt `image-to-coco-json-converter` genutzt wurde [Eij21]. Dieser Ansatz verwendet die Python-Bibliotheken `shapely` [G<sup>+</sup>], `scikit-image` [VdWSNI<sup>+</sup>14] und `Pillow` [Cla15], um die Konturen der Annotationen aus einem Maskenbild zu extrahieren. Dabei ist zu beachten, dass sich Spitzen überlappen können und somit in 2 Bereiche unterteilt werden. Die Kontur einer Spalteninstanz wird mit `skimage` aus dem Bild extrahiert. Shapely stellt die Kontur als Polygon dar und berechnet die Parameter der Bounding Box. Zur Markierung des Keypoint wird das Binärbild der Kontur mit OpenCV [Bra00] angezeigt und der Benutzer muss den vordersten Punkt der Spalte entsprechend markieren. Um ein Maximum an Informationen zu erhalten, werden die acht Spitzen während des gesamten Prozesses als getrennte Klassen behandelt.

Eine Herausforderung besteht darin, eine annotierte Maske mit allen zugehörigen Bildern zu verknüpfen, da sie nach der Konvertierung in das COCO-Format nur über eine ID referenziert wird und nicht über den Dateinamen, wie es bei der Erstellung der Bilder der Fall ist. Zu diesem Zweck werden bereits konvertierte Masken mit Verweis auf ihren Dateinamen zwischengespeichert. Bei nachfolgenden Bildern wird geprüft, ob die zugehörige Maske bereits konvertiert wurde. Falls bereits eine Annotation in COCO-Format vorhanden ist, müssen die Werte „id“ und „image\_id“ entsprechend angepasst werden. Dabei ist immer auf die korrekte Nummerierung zu achten, da es sonst zu Konflikten kommen kann. Es ist notwendig, dass alle anderen Parameter des COCO-Formats ebenfalls korrekt gesetzt sind. Diese sind jedoch trivialer und werden daher hier nicht weiter erläutert.

### Konvertierungsskript

Zur Automatisierung und Vereinfachung des Konvertierungsprozesses, wird ein spezielles Python-Skript entwickelt. Dieses Skript kann von der Kommandozeile aus aufgerufen werden und ermöglicht es, den gesamten Konvertierungsprozess mit einem einzigen Befehl auszuführen. Es nimmt den Pfad zu dem Ordner als Eingabe und führt dann die Konvertierung durch.

Jeder erzeugte Datensatz wird in einem separaten Ordner gespeichert, sodass er sofort für Trainingszwecke verwendet werden kann. Die benötigte und erstellte Ordnerstruktur ist in 5.8 dargestellt.

Das Besondere an diesem Skript ist seine Flexibilität. Es erlaubt die Spezifikation beliebiger Parameterkombinationen und erzeugt entsprechend die gewünschten Datensätze. Dies ermöglicht die Erstellung maßgeschneiderter Datensätze, die auf spezifische Anforderungen und Anwendungsszenarien zugeschnitten sind. Darüber hinaus ist das Skript einfach zu verstehen und zu warten, was zukünftige Anpassungen und Erweiterungen erleichtert. Die verfügbaren Skript-Parameter sind in Abbildung 5.9 aufgelistet.

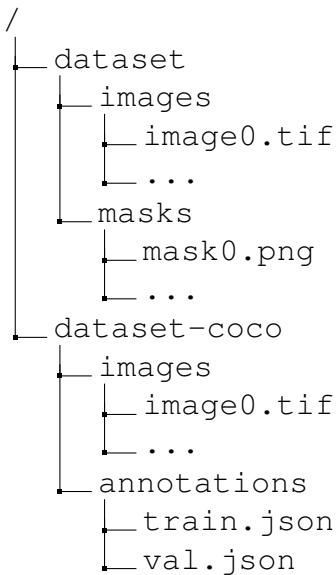


Abbildung 5.8: Darstellung der Verzeichnisstruktur des Datensatzes vor und nach der Konvertierung. Die Anzahl der erstellten Ordner hängt von der Wahl der Parameter des Konvertierungsskripts ab.

- dataset** Pfad zum Datensatz (z.B. /home/user/dataset)
- split** Aufteilungsverhältnis Training/Validierung (z.B. 0.8)
- coco** Konvertierung in das COCO-Datenformat
- yolo** Konvertierung in das YOLO-Datenformat
- oc** Verwendung einer einzigen Klasse für alle Spitzen
- ib** Einbeziehung des Hintergrunds für semantische/panoptische Segmentierung
- kp** Einbeziehung der Koordinaten der Schlüsselpunkte
- mi** Verwendung einer Maske für mehrere Bilder

Abbildung 5.9: Skript-Parameter für die Konvertierung von Datensätzen.

## 5.5 Datensätze

Um die erforderlichen Skripte zu entwickeln sowie die Entwicklungsumgebung einzurichten und zu testen, wurde eine Reihe von Datensätzen erstellt. Eine tabellarische Übersicht aller Datensätze ist in der Tabelle 5.3 zu finden. Die Benennung der Datensätze ist intuitiv und natürlich. Der erste Teil des Namens beschreibt die Anzahl der Bilder. Verschiedene Suffixe werden angehängt, um Informationen darüber zu liefern, wie die Masken annotiert sind. Beispielweise steht das Suffix „-oc“ für „one class“ (eine Klasse). Die Klassen des Datensatzes sind zusammengefasst. Die Spitzen werden also nicht nach ihrer Richtung unterschieden.

Der Datensatz „50img“ enthält 50 Bilder und Masken aus realen Einsatzszenarien. Es diente zur Überprüfung der Annotationsmethode und der Konvertierung in das COCO-Format. Er wurde auch verwendet, um die Entwicklungsumgebung so zu konfigurieren, dass das Training von Mask R-CNN später schnell auf einem großen Datensatz durchgeführt werden kann. Der Datensatz „600img\_1x5“ enthält 600 Bilder, wobei eine Maske auf fünf Bilder passt, und wurde mit der entwickelten automatischen Bildaufnahmemethode erstellt. Er wird später einen Großteil der Trainingsdaten darstellen. „617img\_batch1“ enthält 100 von insgesamt 617 sorgfältig ausgewählten Bildern aus realen Einsätzen. Sie repräsentieren eine Vielzahl von Szenarien und

sollen als Grundlage für eine genaue Darstellung dienen.

Alle Bilder und Masken der oben genannten Datensätze sind in dem Datensatz „750img\_merged“ zusammengefasst. Er besteht somit aus 600 Bildern, die durch das Skript erzeugt wurden, und 150 Bildern, die aus realen Einsätzen stammen. Nach einer genaueren Analyse des Datensatzes stellte sich heraus, dass Spitzen aus verschiedenen Richtungen unterschiedlich häufig in den Bildern vertreten sind. Dies wird auf die 150 Bilder aus realen Einsätzen zurückgeführt. Dargestellt ist die Verteilung in Abbildung 5.10. Es zeigt sich, dass insbesondere die Spitzen fünf (Süd) und sechs (Süd-West) seltener im Einsatz sind. Dennoch wird der Datensatz für das Training von Mask R-CNN verwendet und in der Evaluierung wird untersucht, inwieweit sich dieses Ungleichgewicht auf die Leistungsfähigkeit des Modells auswirkt.

Name	Bildanzahl	Augmentation	Klassen	Entwicklung	Training
50img	50	Nein	8	Ja	Nein
50img-oc	50	Nein	1	Ja	Nein
600img_1x5	600	Ja	8	Ja	Nein
617img_batch1	100	Nein	8	Nein	Nein
750img_merged	750	Teils	8	Nein	Ja
750img_merged-oc	750	Teils	1	Nein	Ja

Tabelle 5.3: Zusammenfassung der Datensätze. Alle Datensätze haben ein Train-Val-Verhältnis von 0.9

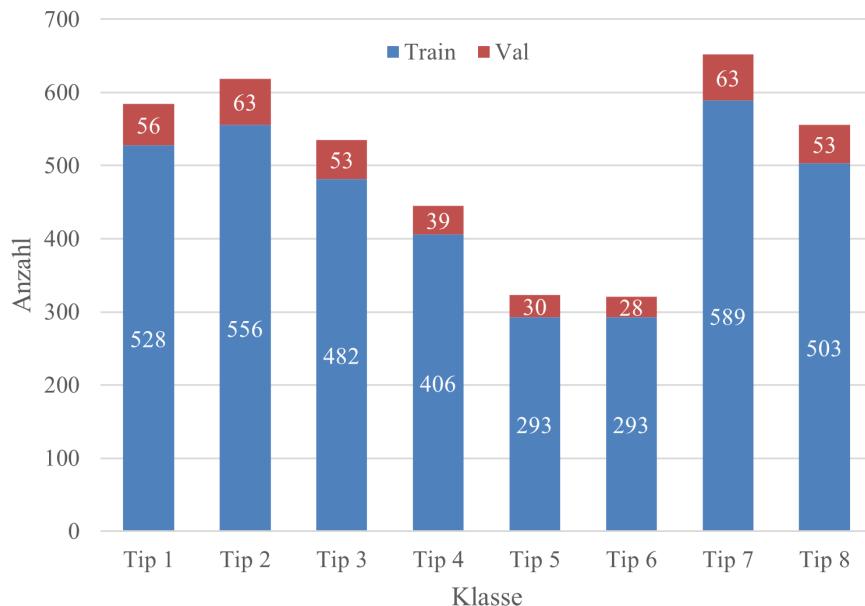


Abbildung 5.10: Anzahl der Instanzen jeder Spitzenklasse von „750img\_merged“. Dies ermöglicht einen Vergleich zwischen den Klassenverteilungen in den Trainings- und Validierungsdaten.

## 5.6 Implementierung und Training der Modelle

### 5.6.1 Entwicklungsumgebung

Für diese Entwicklung wird ein System mit Ubuntu in der Version 22.04 verwendet. Die effiziente Nutzung der zugrunde liegenden Grafikkarte vom Typ „Nvidia GTX 1080“ wird durch die Installation von CUDA 11.8 erreicht, einer von Nvidia entwickelten parallelen Berechnungsplattform und Anwendungsprogrammierschnittstelle (API). CUDA ermöglicht es Entwicklern, Software zu schreiben, die die Hardwarebeschleunigung für rechenintensive Anwendungen nutzt, insbesondere in den Bereichen maschinelles Lernen und Datenverarbeitung [NVF20].

Für die Entwicklung der Netze wird PyTorch in der Version 2.0.1 zusammen mit torchvision-0.15.1 installiert. PyTorch ist ein Open-Source-Maschinelearning-Framework, das die schnelle Entwicklung von Deep Learning-Algorithmen ermöglicht. Es bietet sowohl flexible als auch effiziente Implementierungen von verschiedenen Arten von neuronalen Netzen und anderen maschinellen Lern-Algorithmen [PGM<sup>+</sup>19]. Die entsprechenden Versionen wurden vorab für CUDA 11.8 kompiliert, um das Training auf der Grafikkarte zu ermöglichen und damit zu beschleunigen.

Für das Training der Netze wird Detectron2 verwendet. Da für diese spezielle Systemkonfiguration keine vorkompilierte Version von Detectron2 verfügbar ist, wird sie aus dem Quellcode erstellt.

Die oben beschriebene Entwicklungsumgebung wird hauptsächlich für die Vor- und Nachverarbeitung der Daten sowie für die Implementierung und das Debugging des Mask R-CNN Modells verwendet. Da diese Umgebung jedoch nicht die erforderliche Rechenleistung für ein schnelles und effizientes Training des Modells bietet, wird das Training selbst in einer Cloudbasierten Umgebung, konkret auf Google Colab, durchgeführt.

Google Colab ist eine Cloud-Service-Plattform, die GPU-Ressourcen für maschinelles Lernen und datenwissenschaftliche Forschung zur Verfügung stellt [Goo23]. Für das Training wird eine Instanz mit einer Nvidia V100 Grafikkarte ausgewählt. Die Nvidia V100 ist eine Hochleistungs-GPU, die speziell für maschinelles Lernen und rechenintensive Anwendungen entwickelt wurde. Sie bietet eine erhebliche Beschleunigung im Vergleich zu Consumer-Grafikkarten und ist mit 16 Gigabyte Grafikspeicher ideal für das Training von Deep Learning Modellen geeignet. In dieser Google Colab Umgebung ist es notwendig, Detectron2 zu installieren. Durch den Einsatz von Google Colab kann trotz der Hardwarebeschränkungen der lokalen Entwicklungsumgebung ein effizientes Training des Modells erreicht werden.

### 5.6.2 Implementierung von Mask R-CNN

Der sogenannte „Model Zoo“ von Detectron2, eine Sammlung von vorgefertigten Modellen und Konfigurationen für verschiedene Algorithmen und deren Varianten enthält auch verschiedene Implementierungen von Mask R-CNN und Keypoint R-CNN, die als Ausgangspunkt für das in dieser Arbeit verwendete Modell dienten.

Mask R-CNN wurde aufgrund seiner Modularität ausgewählt. Es kann schnell angepasst werden, um Objektdetektion, Segmentierung und Keypoint-Erkennung durchzuführen, andere Modellarchitekturen bieten diese Flexibilität nicht.

Von besonderem Interesse sind die verschiedenen Backbone-Architekturen, die im Model Zoo von Detectron2 zur Verfügung stehen. Es werden die Backbones R50-FPN, R50-DC5, R101-FPN und R101-DC5 verwendet. Eine detaillierte Erläuterung der Backbones findet sich in Kapitel 3.4.1.

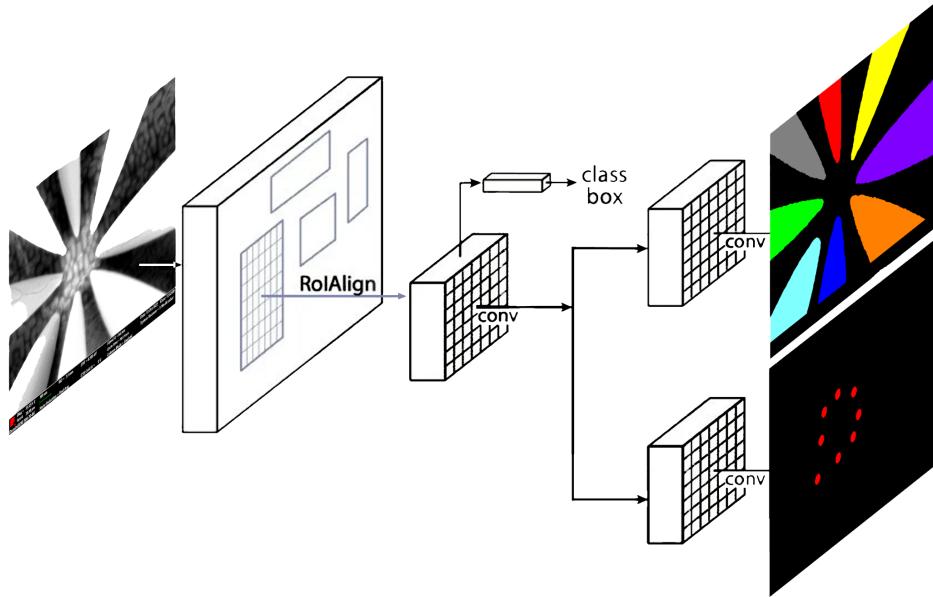


Abbildung 5.11: Die verwendete Mask R-CNN Architektur, der Keypoint-Zweig wird angefügt. Die Keypoints werden als eine One-Hot Maske ausgegeben. Quelle: Modifiziert übernommen von He *et al.* [HGDG17]

Da das Modell sowohl Schlüsselpunkte als auch Segmentierungsmasken ausgeben soll, werden die entsprechenden Konfigurationen von Mask R-CNN und Keypoint R-CNN zusammengefasst, die Architektur ist in Abbildung 5.11 dargestellt. Auf diese Weise können die Vorteile beider Modelltypen genutzt werden und das Modell erzeugt beide Ausgaben gleichzeitig.

Die Konfigurationen werden entsprechend den spezifischen Anforderungen dieser Arbeit angepasst, wobei die Details der Anpassungen im Abschnitt 5.6.4 über die Einstellung der Hyperparameter näher beschrieben werden. Mit diesen Einstellungen konnte ein maßgeschneidertes Mask R-CNN Modell entwickelt werden, das speziell für die Aufgabe der Lokalisierung von Messspitzen im REM geeignet ist.

### 5.6.3 Verwendung und Anpassung der vortrainierten Gewichte

Für diese Arbeit werden verschiedene vortrainierte Modelle aus dem Detectron2 Model Zoo verwendet, aufgelistet sind diese in Tabelle 5.4. Diese Modelle wurden ursprünglich auf dem COCO „train2017“ Datensatz trainiert, einem großen und vielfältigen Datensatz zur Objekterkennung, Segmentierung und Keypoint-Erkennung [COC23].

Da das Modell zur Spitzenerkennung beide Fähigkeiten – Masken- und Keypoint-Vorhersage – beherrschen soll, werden die vortrainierten Gewichte des Keypoint-Zweiges aus den Keypoint R-CNN-Modellen extrahiert und mit den Parametern des entsprechenden Mask R-CNN-Modells mithilfe eines Python-Skripts kombiniert. Dies ermöglicht die Erstellung eines vortrainierten Modells, das sowohl die bereits erlernten Fähigkeiten zur Maskengenerierung als auch zur Keypoint-Erkennung besitzt.

Es ist jedoch anzumerken, dass für die DC5 Varianten der Backbones keine vortrainierten Gewichte für den Keypoint-Zweig zur Verfügung stehen. In diesem Fall wird das Modell nur mit den verfügbaren Mask R-CNN Gewichten trainiert.

Um den vollen Nutzen aus den vortrainierten Gewichten zu ziehen, wird ein spezielles Trainingsschema verwendet. In einer ersten Phase wird das Backbone des Modells „eingefroren“,

Name	Backbone	LR Schedule
Mask R-CNN	R50-DC5	3x
Mask R-CNN	R50-FPN	3x
Keypoint R-CNN	R50-FPN	3x
Mask R-CNN	R101-FPN	3x
Mask R-CNN	R101-DC5	3x
Keypoint R-CNN	R101-FPN	3x

Tabelle 5.4: Genutzte vortrainierte Modelle aus dem Detectron2 Model Zoo.

das heißt, es werden keine Gradienten berechnet und die Gewichte werden nicht aktualisiert. Dadurch kann der Maskenkopf des Modells lernen, die Positionen und Konturen der Spitzen auf Grundlage der bereits gelernten Merkmale des Backbone zu rekonstruieren. In der zweiten Phase des Trainingsprozesses wird das gesamte Modell einschließlich des Backbones auf dem spezifischen Datensatz trainiert. Dieser Schritt stellt sicher, dass das Modell genau an die spezifische Aufgabe angepasst ist. Zur Unterstützung dieses zweiphasigen Trainingsprozesses wird eine speziell angepasste Lernrate verwendet, die den Fortschritt des Modells in beiden Phasen effektiv steuert. Die entwickelte Lernrate ist in Abbildung 5.12 zu sehen.

#### 5.6.4 Wahl der Hyperparameter

In Detectron2 wird für das Training auf acht GPUs eine Lernrate von 0,02 verwendet. Die lineare Skalierungsregel von Goyal *et al.* [GDG<sup>+</sup>17] besagt, dass die Lernrate proportional zur Anzahl der GPUs angepasst werden sollte. Daher wird die anfängliche Lernrate für das Training auf einer einzelnen GPU auf 0,002 reduziert.

Für den Optimierer wird ein Momentum-Wert von 0,9 festgelegt. Dies hilft, den Trainingsprozess in die richtige Richtung zu beschleunigen. Eine Gewichtsabnahme von 0,0001 wird verwendet, und reguliert die Überanpassung. 16 Bilder pro Batch sind die optimale Anzahl, um die Leistung der Nvidia V100 GPU voll auszunutzen, ohne sie zu überlasten. Dies trägt zu einer effizienten Nutzung der Hardware-Ressourcen bei, ohne die Qualität des Trainings zu beeinträchtigen.

Die Modelle werden für 3000 Epochen trainiert. Während der Entwicklung und Erprobung hat sich diese Zahl als optimal für ein ausgewogenes Verhältnis zwischen Modellleistung und Trainingszeit erwiesen. Die ersten 500 Epochen des Trainings dienen als Aufwärmphase. In dieser Phase wird die Lernrate schrittweise linear auf die eingestellte Lernrate erhöht. Liu *et al.* haben gezeigt, dass dieser Vorgang dazu beitragen kann, das Modell besser auf das Training vorzubereiten und Instabilitäten zu Beginn des Trainings zu vermeiden [LJH<sup>+</sup>19].

In den ersten 1800 Epochen werden die Masken und der Keypoint-Kopf trainiert, während das Backbone-Modell eingefroren bleibt. Die folgenden 1200 Epochen dienen der Feinabstimmung des gesamten Modells, einschließlich des Backbones. Diese zweite Trainingsphase beginnt wieder mit einer Aufwärmphase von 200 Epochen.

Die Lernrate wird während des Trainings in festgelegten Schritten abgestuft und reduziert, einerseits um die Modellkonvergenz zu fördern und andererseits ein Überspringen des globalen Minimums zu verhindern. Diese Anpassung ermöglicht eine anfänglich breitere Exploration des Lösungsraums und eine anschließende feinere Suche, wenn sich das Modell der optimalen Lösung nähert. Die entwickelte Strategie in Abbildung 5.12 dargestellt.

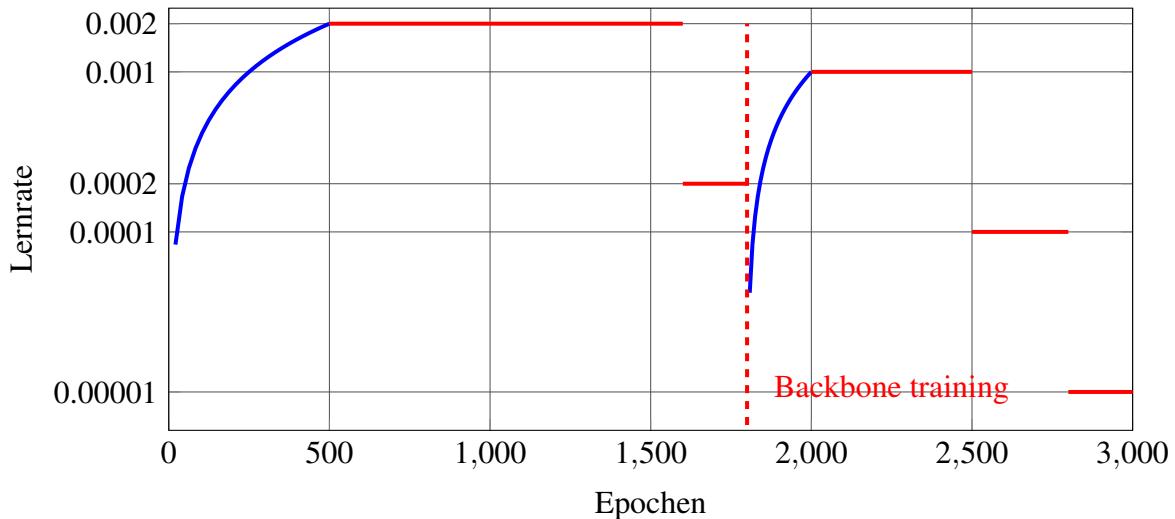


Abbildung 5.12: Der Verlauf der Lernrate über die Epochen, logarithmisch dargestellt. Lernrate der Aufwärmepochen in Blau, konstante Lernrate in Rot.

### 5.6.5 Erweiterung der Modelle: Richtungsvorhersage

In einem weiteren Schritt werden die beiden Modellvarianten, die die beste Leistung bei der Erkennung von Spitzen aus allen Richtungen zeigen, erweitert. Dazu werden sie modifiziert und auf dem Datensatz „750\_merged“ trainiert. Die Spitzen werden nun nicht nur allgemein, sondern speziell nach ihrer Richtung unterschieden. Dahinter steht die Annahme, dass ein Modell, das in der Lage ist, Spitzen aus allen Richtungen zu erkennen, auch in der Lage sein könnte, diese Spitzen anhand ihrer spezifischen Position zu unterscheiden. Dies würde eine wesentliche Verbesserung für die automatisierte Steuerung darstellen, da eine nachträgliche Zuordnung der Segmentierung und der Schlüsselpunkte zur richtigen Spalte nicht mehr erforderlich wäre.

Es ist jedoch zu berücksichtigen, dass durch die spezifische Klassifizierung der Spitzen für jede Klasse insgesamt weniger Trainingsbilder zur Verfügung stehen. In einem ausgewogenen Datensatz führt dies zu einer Reduktion der Datenmenge auf ein Achtel pro Klasse. Dies stellt eine zusätzliche Herausforderung dar, da das Modell nun feinere Unterscheidungen treffen muss, während die Anzahl der Trainingsbeispiele pro Klasse reduziert wird.

Trotz der feineren Unterscheidung der Spitzen könnte die Aufgabe aus einer anderen Perspektive einfacher sein. Da die Spitzen annähernd die gleichen Eigenschaften haben und sich hauptsächlich in ihrer Ausrichtung unterscheiden, muss jede der acht Masken Zweige des Modells nur eine der acht möglichen Darstellungen lernen. Da die auf eine Klasse trainierten Modelle bereits die Merkmale der Spitzen gelernt haben, werden die vorgeübten Backbones als Basis für die erweiterten Modelle verwendet. Die Masken-Zweige von Mask R-CNN werden von Grund auf neu trainiert.

Eine sorgfältige Bewertung der Leistung der Modelle ist entscheidend, um das Verhalten der Vorhersagen zu bestimmen, wenn das Modell auf diese Weise trainiert wird.

# 6 Evaluation

In diesem Kapitel liegt der Schwerpunkt auf der Evaluierung der im Rahmen dieser Arbeit entwickelten Modelle und Methoden. Besonderes Augenmerk wird auf die Keypoint-Erkennung gelegt. Diese Technologie hat das Potenzial, revolutionäre Veränderungen in verschiedenen Anwendungsbereichen, insbesondere in automatisierten Prozessen, zu bewirken. Dies erfordert jedoch ein hohes Maß an Genauigkeit und Zuverlässigkeit. Daher wird die Genauigkeit der Modellvorhersagen systematisch analysiert, um ein klares Bild von der Zuverlässigkeit der Modelle zu erhalten und mögliche Schwachstellen aufzudecken.

Das Hauptziel dieses Kapitels ist es, zu verstehen, wie sich die Wahl des Backbones und der Hyperparameter auf die Leistung der Modelle auswirkt. Da der Einsatz von Deep Learning für diese spezielle Aufgabenstellung ein neuer Ansatz ist und keine direkten Vergleichswerte vorliegen, ist es umso wichtiger, die vielversprechenden Möglichkeiten, aber auch die potenziellen Probleme zu diskutieren und zu bewerten.

Die Evaluierung ist in mehrere Teile gegliedert, beginnend mit der Definition der Evaluierungsmetriken, gefolgt von einer ausführlichen Evaluation auf dem Validierungsdatensatz. Diese beinhaltet die Ergebnisse der Segmentierung und der Keypoint-Vorhersage. Anschließend wird eine Analyse der fehlerhaften Vorhersagen durchgeführt, bevor die Vorhersagen in Bezug auf die Richtung der Spitze betrachtet werden. Der Abschnitt endet mit einer visuellen Überprüfung der Ergebnisse, um eine visuelle Beurteilung der Modellleistung zu ermöglichen.

## 6.1 Evaluationsmetriken

Zur Bewertung der in dieser Arbeit entwickelten Modelle werden verschiedene Metriken verwendet. Besonders betrachtet wird jedoch die  $AP^{IoU=.75}$  und  $AP^{OKS=.75}$ . Diese strengen Varianten der AP erfordern einen IoU- und OKS-Wert von mehr als 0,75. Das bedeutet, dass nur die Vorhersagen als korrekt angesehen werden, die eine hohe Übereinstimmung mit den tatsächlichen Spitzen aufweisen. Dies ist in unserer Anwendung besonders wichtig, da wir eine sehr genaue Lokalisierung und Erkennung der Spitzen anstreben. Eine niedrige  $AP^{.75}$  würde auf viele Fehlentscheidungen des Modells hindeuten und könnte bei der Anwendung in automatisierten Prozessen zu Problemen führen. Daher streben wir einen hohen  $AP^{IoU=.75}$  an, um die Zuverlässigkeit und Genauigkeit unserer Modelle zu gewährleisten.

Besonderes Augenmerk ist auch auf die Falsch-Negativ- und Falsch-Positiv-Raten zu richten. Eine Falsch-Positiv-Erkennung, bei der das Modell eine Spitze erkennt, wo keine ist, kann später zu einer Fehlsteuerung und damit zu großen Schäden führen. Dieser Fehlertyp ist daher unbedingt zu vermeiden. Ähnlich problematisch sind Falsch-Negativ-Erkennungen, bei denen das Modell eine vorhandene Spitze nicht erkennt. Dies kann insbesondere dann zu Problemen führen, wenn mehrere Spitzen dicht beieinander liegen und diese bewegt werden sollen. Um einen sicheren Betrieb zu gewährleisten, müssen dem System stets korrekte Informationen über die Position der Spitzen zur Verfügung stehen.

## 6.2 Auswertung auf dem Validierungsdatensatz

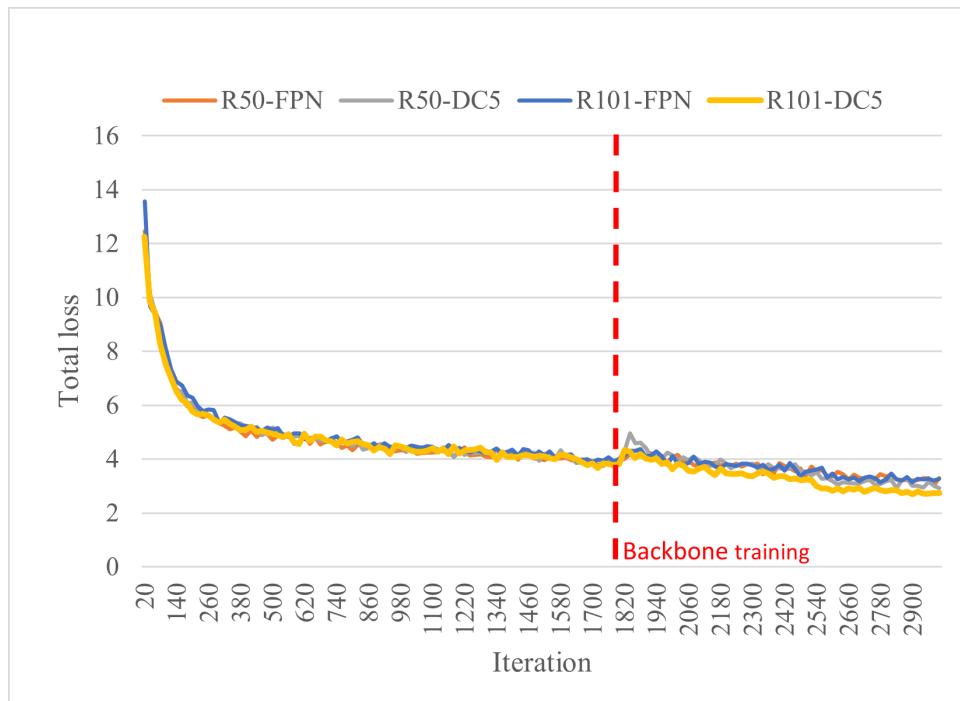


Abbildung 6.1: Gesamter Fehler der Modelle über die Trainingsepochen. Bei Iteration 1800 beginnt das training des Backbones.

Die Abbildung 6.1 zeigt den Gesamtfehler der trainierten Netze über die Epochen. Sie zeigt eine stabile Konvergenz aller Modelle ohne signifikante Sprünge, mit Ausnahme eines vorübergehenden Fehleranstiegs nach Iteration 1800, als das Training des Backbones zur Verfeinerung der Merkmalsextraktion beginnt. Die Kurven der verschiedenen Modelle überlagern sich, was auf ein ähnliches Lernverhalten hindeutet.

### 6.2.1 Ergebnisse der Segmentierung

Backbone	AP	$AP^{IoU=.5}$	$AP^{IoU=.75}$
ResNet-50-FPN	47.80	84.68	46.08
ResNet-50-DC5	53.92	<b>89.22</b>	<b>60.23</b>
ResNet-101-FPN	50.39	88.62	51.01
ResNet-101-DC5	<b>54.33</b>	89.05	59.24

Tabelle 6.1: Mittelwert der letzten 200 Epochen von AP,  $AP^{IoU=.5}$ ,  $AP^{IoU=.75}$  der Segmentierung für die verschiedenen Modelle.

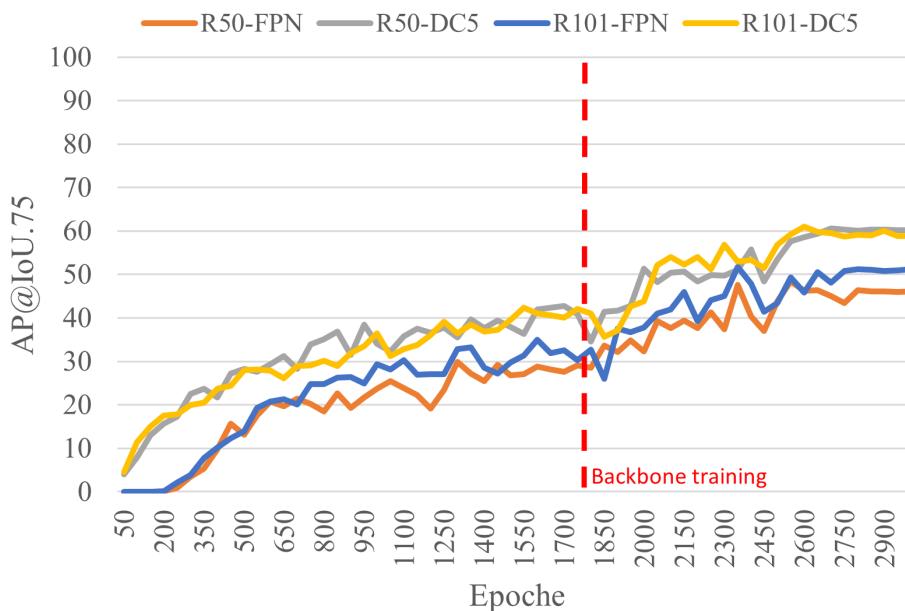


Abbildung 6.2: Entwicklung der  $AP^{IoU=.75}$  für die verschiedenen Backbones während der Trainingsepisoden.

Die AP-Werte der Segmentierung für verschiedene IoU-Werte, aufgeschlüsselt nach verschiedenen Backbone-Varianten sind in Tabelle 6.1 dargestellt. Bei den ResNet-50-Backbones ist ein Anstieg der AP um ca. fünf Prozent beim Wechsel von FPN zu DC5 zu beobachten. Dieser Trend ist bei allen AP-Varianten zu beobachten. Ähnliche Verbesserungen zeigen sich auch beim Wechsel von FPN zu DC5 in den ResNet-101-Backbones, insbesondere bei der strengen Metrik  $AP^{IoU=.75}$ , wo ein Anstieg von über acht Prozent gemessen wird. Auffällig ist, dass die Genauigkeit der Segmentierung zwischen den Netzwerken ResNet-50-DC5 und ResNet-101-DC5 nur einen Unterschied von weniger als einem Punkt aufweist.

Aus Abbildung 6.2 wird deutlich, dass die DC5-Varianten der Netze bereits nach 50 Epochen einen Trainingseffekt aufweisen. Im Gegensatz zur FPN-Variante, die erst nach 200 Epochen Vorhersagen liefert. Über den gesamten Trainingszeitraum erreichen die DC5-Varianten eine höhere  $AP^{IoU=.75}$  als die FPN-Varianten. Eine erkennbare Konvergenz aller Netze tritt nach 1800 Epochen auf. Die Aktivierung des Backbone-Trainings führt jedoch zu einem weiteren Anstieg der AP-Werte.

## 6.2.2 Ergebnisse der Keypoint-Vorhersage

Backbone	AP	$AP^{OKS=.5}$	$AP^{OKS=.75}$
ResNet-50-FPN	90.74	91.94	90.79
ResNet-50-DC5	94.88	95.08	95.08
ResNet-101-FPN	93.40	93.78	93.67
ResNet-101-DC5	<b>95.24</b>	<b>95.34</b>	<b>95.34</b>

Tabelle 6.2: AP,  $AP^{OKS=.5}$ ,  $AP^{OKS=.75}$  Werte der Keypoint-Vorhersage für die verschiedenen Modelle.

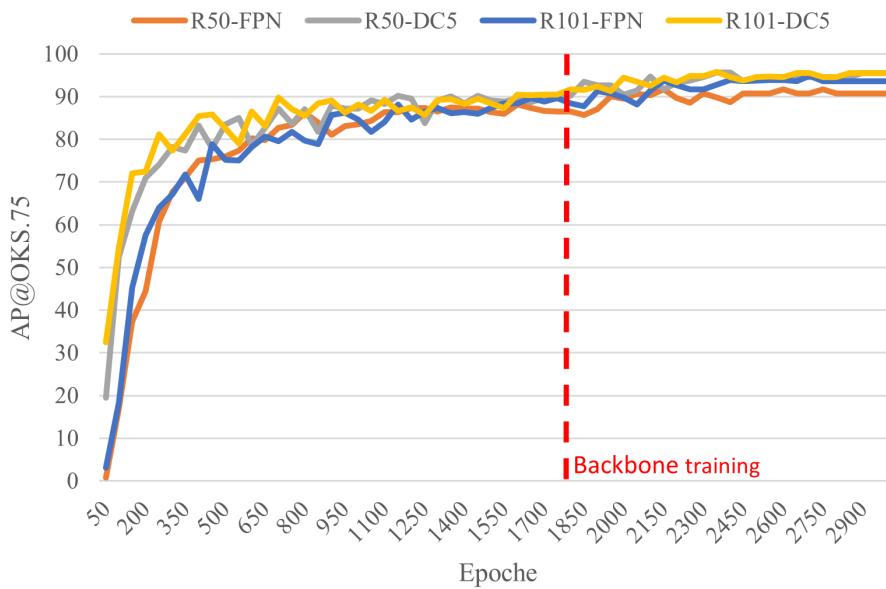


Abbildung 6.3: Darstellung der  $AP^{OKS=.75}$  der Keypoint-Vorhersage für die verschiedenen Backbones über die Trainingsepisoden.

Die in Tabelle 6.2 und Abbildung 6.3 dargestellten AP-Werte bei unterschiedlichen OKS-Werten, geben einen wertvollen Einblick in die Leistungsfähigkeit verschiedener Backbone-Modelle im Kontext der Key-Point-Vorhersage. Der steile Anstieg der  $AP^{OKS=.75}$  in den ersten 300 Epochen deutet darauf hin, dass alle Netze bereits nach wenigen Trainingsepochen die ungefähre Position des Apex der Spitze lernen. Ein stetiger Anstieg der Genauigkeit ist in den folgenden Epochen zu beobachten, bis die Netze um Epoche 1500 konvergieren. Ein weiterer kleiner, aber signifikanter Lerneffekt tritt mit dem Backbone-Training nach Epoche 1800 auf.

Die endgültigen AP-Werte, die in Tabelle 6.2 dargestellt sind, unterscheiden sich bei der Keypoint-Erkennung zwischen den Netzen nur um wenige Prozent, aber jede Verbesserung ist hier entscheidend. Insbesondere die Verwendung von DC5 anstelle von FPN führt zu einer deutlichen Verbesserung, wobei ResNet-101-DC5 in allen AP-Varianten die höchste Genauigkeit erreicht. Dies deutet auf eine überlegene Genauigkeit dieses Modells bei der Vorhersage von Schlüsselpunkten hin. Auch die Modelle ResNet-50-FPN und ResNet-101-FPN erreichen mit über 90 Prozent ebenfalls respektable Werte, insbesondere das Modell ResNet-50-DC5, dessen Leistung bis auf weniger als einen Punkt an die von ResNet-101-DC5 heranreicht. Dies legt nahe, dass diese Modelle trotz ihrer Unterlegenheit als robuste Optionen für Aufgaben mit geringeren Genauigkeitsanforderungen in Betracht gezogen werden könnten.

## 6.3 Fehlerhafte Vorhersagen

Backbone	<i>FN</i>	<i>FP</i>
ResNet-50-FPN	0.0937	0.0597
ResNet-50-DC5	0.0819	0.0429
ResNet-101-FPN	0.0941	0.0536
ResNet-101-DC5	<b>0.0780</b>	<b>0.0390</b>

Tabelle 6.3: Falsch-negativ und falsch-positiv Raten von Mask R-CNN mit verschiedenen Backbones.

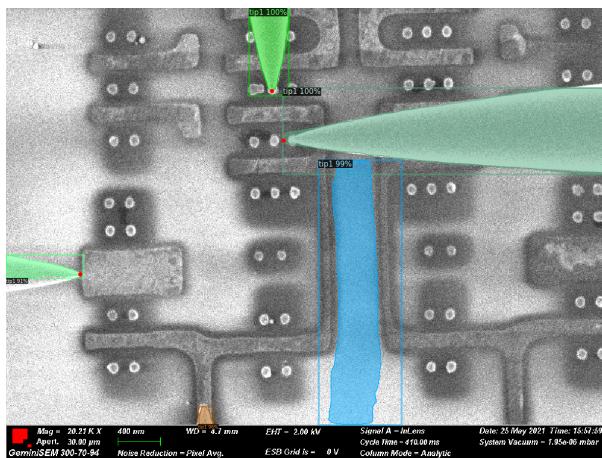


Abbildung 6.4: Visualisierung der Detektionen des ResNet-50-FPN-Netzes mit einem Konfidenzschwellenwert von 90%.

Tabelle 6.3 zeigt die FN- und FP-Raten der verschiedenen Netze. Wie eingangs erwähnt, spielen neben der Genauigkeit der Keypoint-Erkennung auch die FN- und FP-Raten eine entscheidende Rolle, da falsche Detektionen zu erheblichen Schäden führen können.

Alle Netze erreichen eine FN-Rate von unter zehn Prozent, das heißt von 100 Spitzen werden weniger als 10 Nadeln nicht erkannt. Die FP-Rate, ein Indikator dafür, wie viele spitzen ähnliche Strukturen das Netz als Spitzte identifiziert, obwohl keine vorhanden ist, liegt bei allen Netzen unter 6 Prozent. Die DC5-Varianten der Backbones zeigen erneut ihre Überlegenheit, da sie in allen Fällen um etwa eineinhalb Prozent niedrigere Werte erreichen. Das ResNet-101-DC5-Backbone schneidet von allen Kombinationen am besten ab, wenn auch nur mit geringem Abstand.

Es ist zu beachten, dass eine Verringerung der FN-Rate durch Herabsetzung des Konfidenzschwellenwertes zwangsläufig zu einer Erhöhung der FP-Rate führt und umgekehrt. Daher muss ein angemessenes Gleichgewicht zwischen diesen beiden Raten gefunden werden, das den spezifischen Anforderungen der jeweiligen Aufgabe entspricht. Die Abbildung 6.4 veranschaulicht diese Abwägung anhand eines Beispiels. Die falsch-positive Detektion, hier in hellblau dargestellt, hat eine Konfidenz von 99%. Die Detektionen von Spitze 7 (West) und 5 (Süd) haben eine Konfidenz von 91% beziehungsweise 96%. Beachtlich ist die Detektion der Spitze 5. Wenn der Schwellenwert auf 99% erhöht wird, sinkt die FP-Rate, aber die FN-Rate steigt. Diese Darstellung verdeutlicht die notwendige Abwägung zwischen der Minimierung von falsch-positiven und falsch-negativen Detektionen.

## 6.4 Vorhersage mit Spitzenrichtung

In diesem Abschnitt wird die Leistung der Modelle ResNet-50-DC5 und ResNet-101-DC5 untersucht, die so trainiert wurden, dass sie die Spitzen je nach ihrer Richtung als acht verschiedene Klassen behandeln. Ziel dieser Modifikation ist es, nicht nur das Vorhandensein von Spitzen, sondern sie auch anhand ihrer Richtung zu klassifizieren.

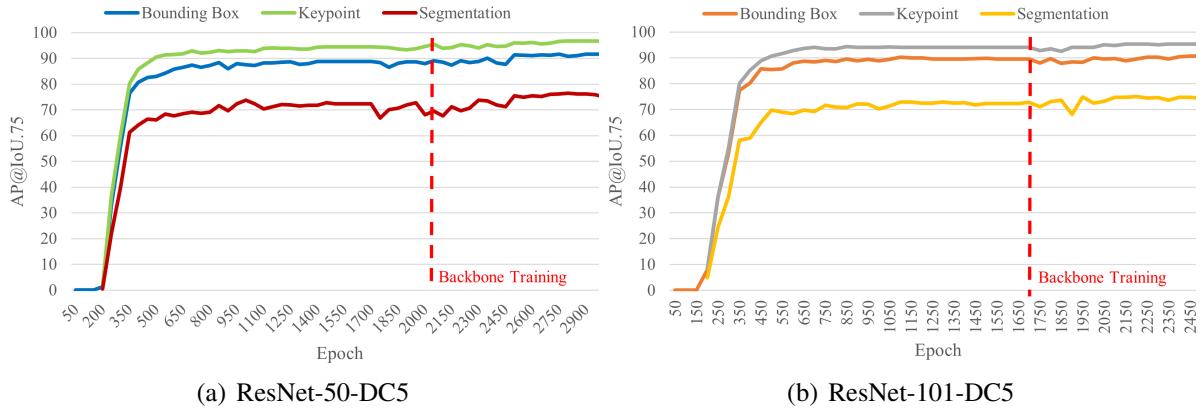


Abbildung 6.5: AP<sup>.75</sup>-Werte für die Bounding-Box-Vorhersage, die Segmentierung und die Keypoint-Erkennung. Für die Modelle, welche die Spitzen anhand ihrer Richtung unterscheiden.

Backbone	AP <sub>BB</sub> <sup>IoU=.75</sup>	AP <sub>S</sub> <sup>IoU=.75</sup>	AP <sub>K</sub> <sup>Oks=.75</sup>	FN	FP
ResNet-50-DC5	86.11	60.23	95.08	0.0819	0.0429
ResNet-50-DC5-Multiclass	89.52	<b>75.53</b>	95.18	0.0617	0.0301
ResNet-101-DC5	87.45	59.24	95.34	0.0780	0.0390
ResNet-101-DC5-Multiclass	<b>90.33</b>	74.34	<b>95.34</b>	<b>0.055</b>	<b>0.023</b>

Tabelle 6.4: Vergleich der Werte AP<sub>BB</sub><sup>IoU=.75</sup>, AP<sub>S</sub><sup>IoU=.75</sup>, AP<sub>K</sub><sup>Oks=.75</sup> des auf ein oder acht Klassen trainierten Modells. Aufgeteilt nach Backbone-Architektur.

In Abbildung 6.5 ist der Fortschritt der Modelle während des Trainingsprozesses dargestellt. Beide Modelle zeigten einen steilen Anstieg der Vorhersagegenauigkeit um die 300. Epoche und beginnen bereits nach circa 400 Epochen zu konvergieren. Das weitere Training der Backbones ab Epoche 1700 führt zunächst zu einer Instabilität der Modelle, die sich jedoch wieder ausgleicht und die Gesamtleistung nur marginal verbesserte. Dies bestätigt die Annahme, dass das auf einer Klasse vortrainierte Backbone die Eigenschaften der Spitzen bereits optimal extrahiert.

Eine deutliche Steigerung zeigt sich bei der Genauigkeit der Segmentierung beider Modelle. Tabelle 6.4 zeigt, dass die Modelle im Vergleich zu den zuvor trainierten Varianten einen Zuwachs von etwa 15 Prozent verzeichnen. Dies deutet auf eine verbesserte Fähigkeit der Modelle hin, die Struktur und Position der Nadelspitzen genauer zu erkennen. Bemerkenswert ist die drastische Reduzierung der FN- und FP-Raten um 40 Prozent. Dies zeigt, dass die Modelle nicht nur besser in der Lage sind, echte Nadelspitzen korrekt zu identifizieren, sondern auch weniger dazu neigen, andere Strukturen fälschlicherweise als solche zu interpretieren. Das Modell macht also weniger Fehler, sowohl bei übersehenen als auch bei fälschlich identifizierten Spitzen. Die Genauigkeit der Schlüsselpunkt-Vorhersage bleibt unverändert.

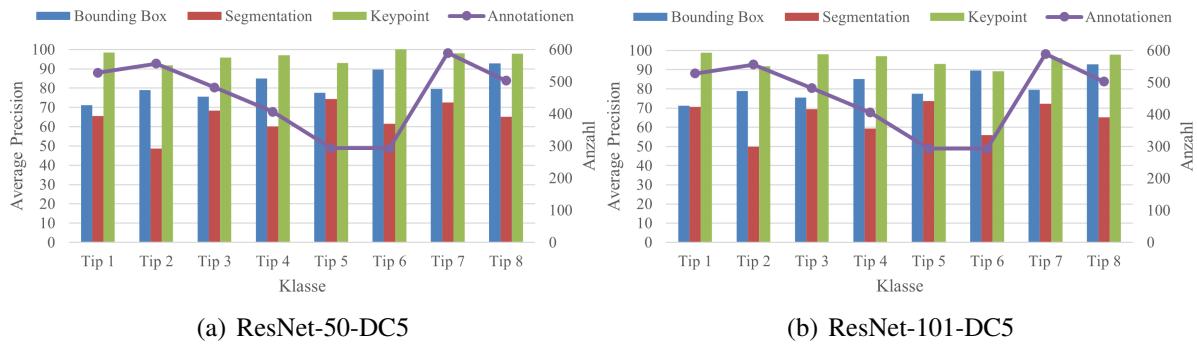


Abbildung 6.6: AP<sup>75</sup>-Werte der Vorhersagen, getrennt nach den acht Klassen und die Anzahl der Instanzen pro Klasse im Trainingsdatensatz. Es wird eine mögliche Korrelation zwischen der Menge der Trainingsdaten und der Vorhersagegenauigkeit untersucht.

Korrelation	ResNet-50-DC5			ResNet-101-DC5		
	AP <sub>BB</sub> <sup>IoU=.75</sup>	AP <sub>S</sub> <sup>IoU=.75</sup>	AP <sub>K</sub> <sup>OKS=.75</sup>	AP <sub>BB</sub> <sup>IoU=.75</sup>	AP <sub>S</sub> <sup>IoU=.75</sup>	AP <sub>K</sub> <sup>OKS=.75</sup>
Trainingsbilder	0,159	0,169	0,741	-0,089	0,047	0,526
Eckposition	-0,776	0,681	-0,220	-0,727	0,864	0,384

Tabelle 6.5: Berechnete Korrelationskoeffizienten zwischen der Anzahl der Trainingsbilder pro Spalte und der Genauigkeit des Modells sowie zwischen der Eckposition der Spitzen und der Genauigkeit.

Die Unterteilung der Spitzen in acht Klassen ermöglicht es, den Zusammenhang zwischen der Anzahl der Instanzen der Spitzen in den Trainingsbildern und der Leistungsfähigkeit der Modelle zu analysieren. Wie bereits in Abbildung 5.10 dargestellt, sind insbesondere die Spitzen 5 (Süd) und 6 (Süd-West) unterrepräsentiert.

Abbildung 6.6 veranschaulicht die Leistung der Modelle auf den verschiedenen Klassen und die Klassenverteilung des Trainingsdatensatzes. Für eine quantitative Analyse wird die Korrelation zwischen der Modellleistung und der Anzahl der Instanzen der jeweiligen Klassen berechnet. Auffällig ist dabei, dass  $AP_K^{OKS=.75}$  eine hohe positive Korrelation aufweist. Das kann bedeuten, dass je mehr Bilder von einer Spalte für das Training zur Verfügung stehen, desto genauer ist die Keypoint-Vorhersage. Die erzeugten Segmentierungen und Bounding Boxen zeigen jedoch nur für ResNet-50-DC5 eine schwache Korrelation.

Besonders interessant ist die starke Korrelation zwischen der Position einer Spalte (ob sie aus einer Ecke in das Bild hineinragt) und der Genauigkeit der Bounding Box und der Segmentierung. Die Genauigkeit der Bounding Box  $AP_{BB}^{IoU=.75}$  korreliert positiv mit der Eckposition. Dies ist darauf zurückzuführen, dass die Bounding Box bei schrägen Objekten per Definition größer ist, da sich nicht eng anliegt. Dies hat zur Folge, dass bei einer festen IoU eine größere Abweichung toleriert wird, was die berechnete Genauigkeit erhöht. Andererseits korreliert die Genauigkeit der Segmentierung negativ mit der Eckposition. Dies deutet darauf hin, dass die Netze Schwierigkeiten bei der Segmentierung von schrägen Nadeln haben, dies wird im folgenden Kapitel 6.5 durch die optische Überprüfung bestätigt.

## 6.5 Optische Überprüfung

Trotz der Bedeutung der quantitativen Metriken ist die qualitative Bewertung der Modellvorhersagen ebenso wichtig, um eine Gesamtbeurteilung der Modellleistung zu erhalten. Zu diesem Zweck werden dem Modell 100 nicht annotierte Bilder vorgelegt und die Ergebnisse manuell überprüft. Diese Methode ermöglicht eine intuitive und praktische Bewertung der Modellgenauigkeit. Beispielsweise können Muster von Fehlern oder Ungenauigkeiten identifiziert werden, die durch quantitative Metriken möglicherweise übersehen werden. Darüber hinaus bietet die visuelle Verifikation die Möglichkeit, die Qualität der Modellvorhersagen im Kontext realer Anwendungen zu beurteilen, was letztlich ein entscheidendes Kriterium für die Nützlichkeit des Modells ist.

Die Ergebnisse dieser Überprüfung und ihre Bedeutung für die Modellleistung und -auswahl werden in den folgenden Abschnitten ausführlich dargestellt und diskutiert. Alle Bilder werden von den Modellen bei einem Konfidenzschwellwert von 99% analysiert, was eine hohe Sicherheit der Modellvorhersagen erfordert.

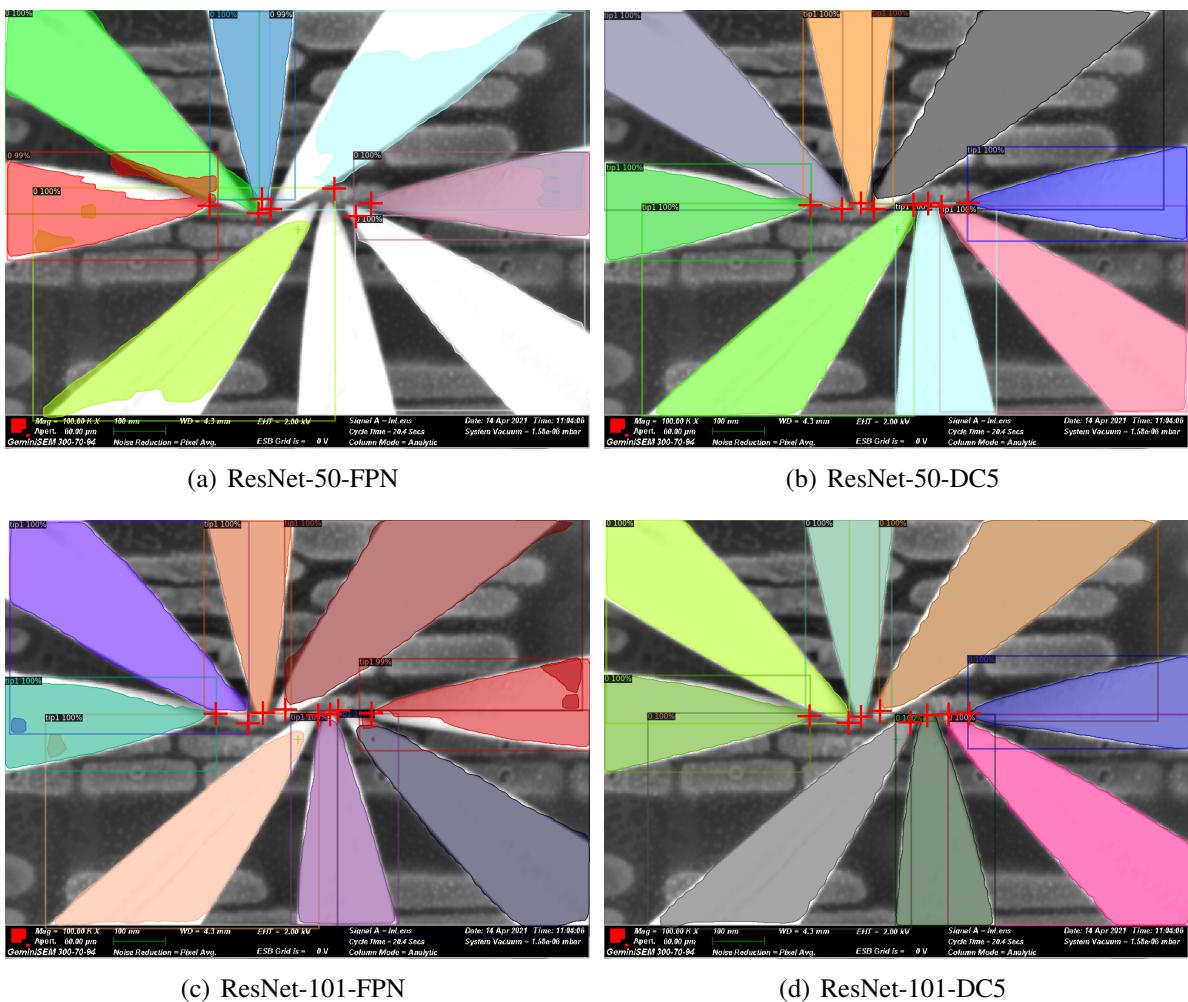


Abbildung 6.7: Acht Spitzen beim Kontaktieren einer Probe. Aufgenommen bei 100.000-facher Vergrößerung. Die Segmentierung der jeweiligen Netze ist über das Bild gelegt.

Bei der Untersuchung von Segmentierungsmasken und Keypoint-Vorhersagen für Bilder mit einer Vergrößerung von mehr als 30.000x – Bedingungen, die typischerweise während des Probings auftreten – zeigt sich, dass nur die FPN-Varianten falsch-positive Detektionen liefern, während die DC5-Varianten durchweg korrekte Detektionen enthält. Auffällig ist auch, dass die auf dem FPN-Backbone basierenden Modelle eine unsaubere Segmentierung erzeugen, die durch die Bildung von Artefakten gekennzeichnet ist. Diese Artefakte treten in der Form auf, dass Teile anderer Spitzen oder des Hintergrundes fälschlicherweise der Spitzte zugeordnet werden oder ein Teil der Spitzte überhaupt nicht erkannt wird. Zudem sind die Keypoints des ResNet-50-FPN oft nicht exakt gesetzt. Teilweise liegen sie neben der Spitzte oder sind zu weit nach hinten versetzt. Im Gegensatz dazu zeigen die DC5-Varianten der Modelle keine Artefakte. Sie segmentieren die Spitzen mit hoher Genauigkeit und auch die Keypoints scheinen zuverlässig erkannt zu werden.

In Abbildung 6.7 ist ein Szenario zu sehen, in dem die Beobachtungen zum Teil ebenfalls auftreten. Es wird auch beobachtet, dass alle Modelle Schwierigkeiten haben, wenn nur ein kleiner Teil der Spitzte sichtbar ist. Eine weitere interessante Beobachtung betrifft allerdings alle Modelle gleichermaßen: Die Konturen der Segmentierungen der in den Ecken liegenden Spitzten (Spitze 2, 4, 6, 8) erscheint wellig.

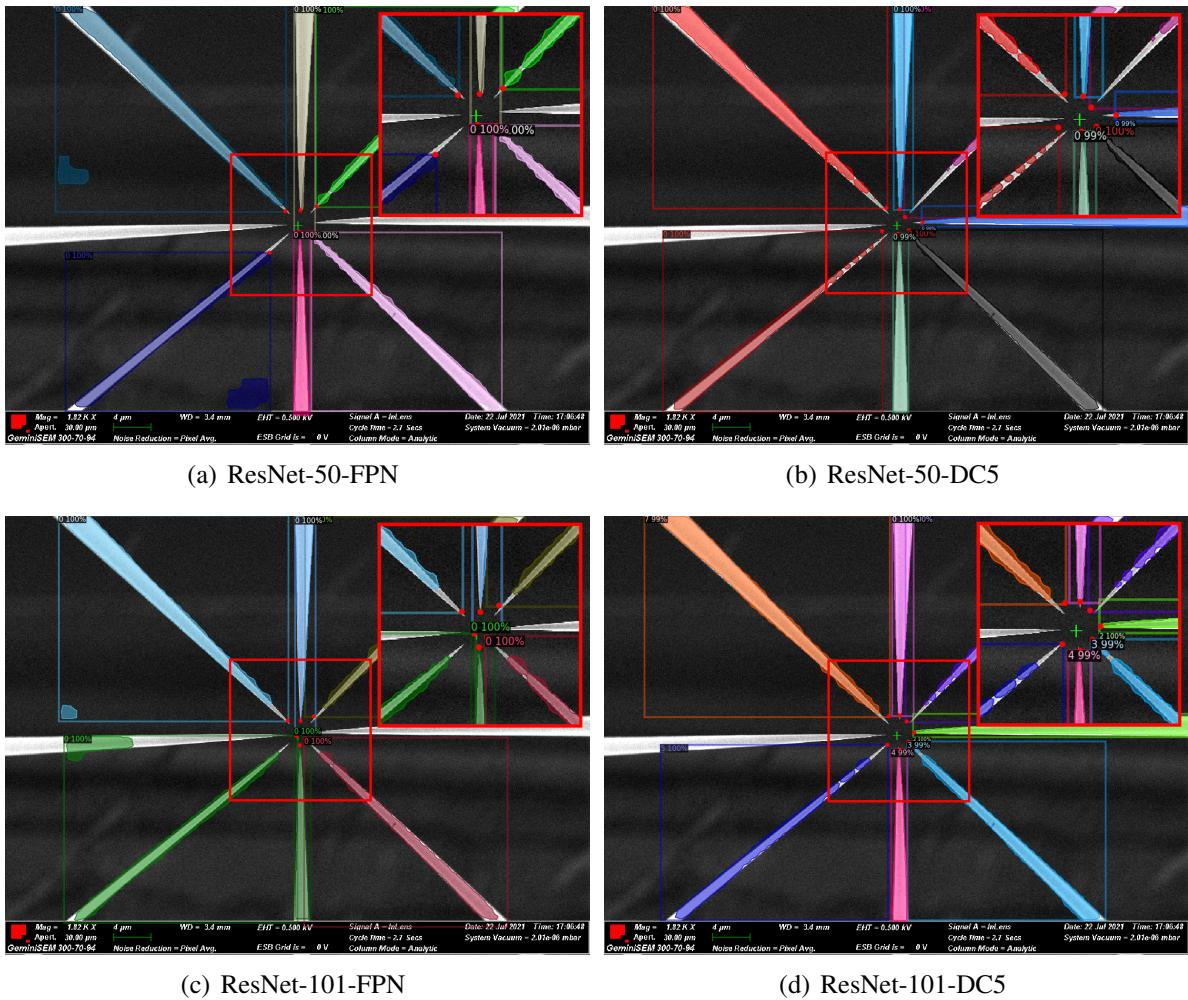


Abbildung 6.8: Segmentierung und Keypoints der verschiedenen Netzvarianten. Aufgenommen bei 2.000-facher Vergrößerung. Der zentrale Bereich ist vergrößert dargestellt.

Bei Vergrößerungen im Bereich von  $10^3$ , bei der die Nadeln lang, schlank und spitz erscheinen (siehe Abbildung 6.8), zeigen die Modelle ein ähnliches Verhalten. Vereinzelt werden Nadeln gar nicht oder nur bei einer Konfidenz im Bereich von 90% bis 98% erkannt. Die Modelle haben Schwierigkeiten, den vorderen Teil der Spitze korrekt zu segmentieren. Teilweise hört die Segmentierung vor dem vordersten Punkt auf oder die Umgebung wird der Spitze zugeordnet. Die vorher beobachtete wellenförmige Segmentierung der Eckspitzen, führt bei Vergrößerungen im Bereich von 1000 bis 5000x zu einer sogenannten „Inselbildung“ in der Segmentierung des vorderen Spitzenbereichs. Die FPN-Varianten der Modelle leiden bei der Segmentierung in diesen Szenarien unter Artefakt Bildung.

Die quantitative Messung der Genauigkeit spiegelt sich in der Keypoint-Vorhersage wider: Die DC5-Varianten der Modelle erzeugen genauere Keypoint-Vorhersagen als die FPN-Varianten. Ebenso bestätigt sich, dass die ResNet-101-Variante des Backbones zu einer genaueren Identifikation der Keypoints beiträgt.

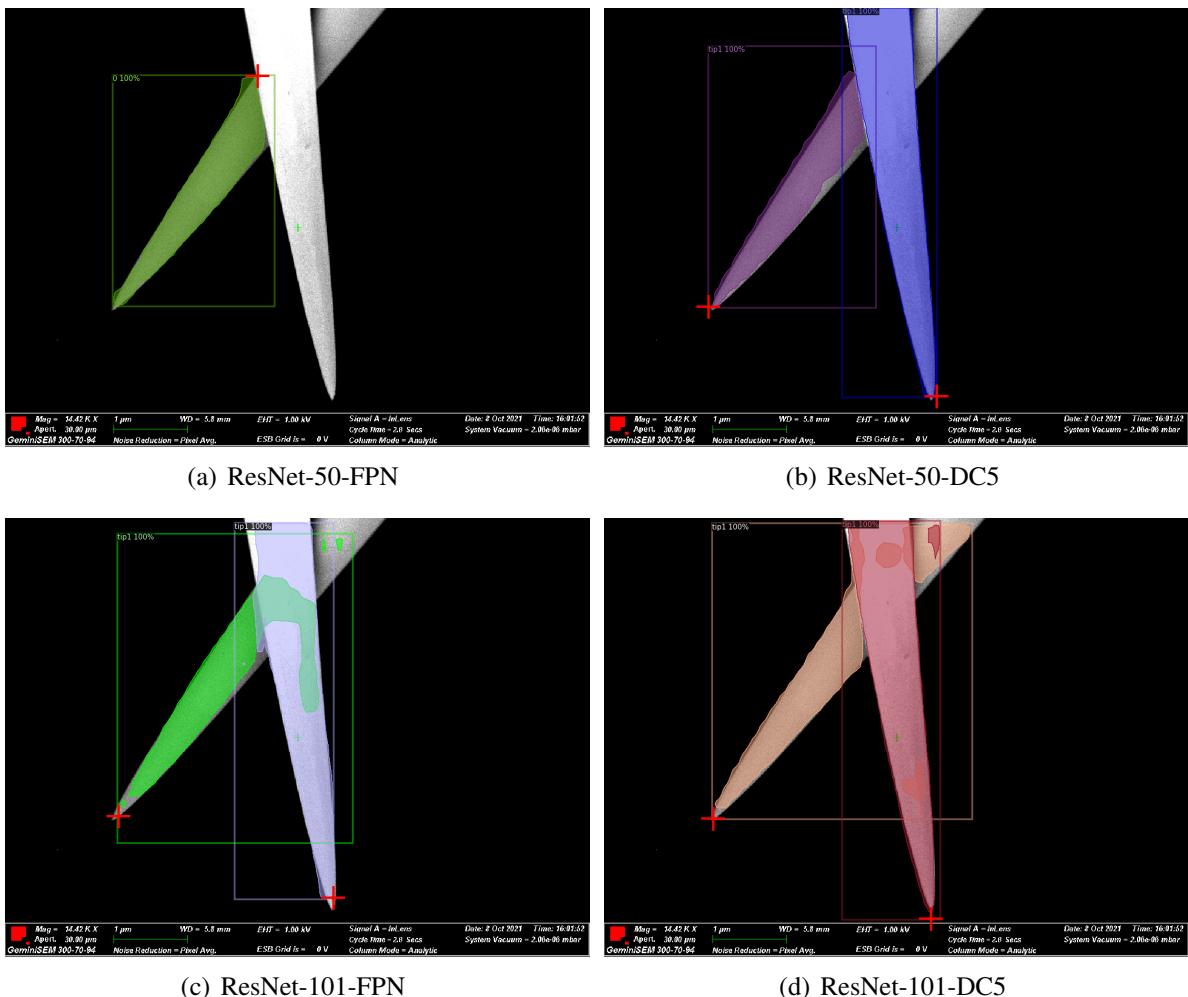


Abbildung 6.9: Zwei gekreuzte Nadeln, dieses Szenario tritt bei der Reinigung der Spitzen auf.

Abbildung 6.9 zeigt die Spitzen in einem speziellen Szenario. Dem sogenannten Tip-Cleaning. Dabei überlagern sich zwei Nadeln. Diese Situation stellt eine besondere Herausforderung dar, da eine Spalte in zwei nicht zusammenhängenden Teilen des Bildes dargestellt wird und das Modell ein gewisses Verständnis dafür haben muss, dass Spitzen kontinuierlich verlaufen und nicht abrupt enden. Keines der Modelle lieferte in dieser Situation robuste Detektionen. Sie weisen Fehler auf, die von falsch-negativen Erkennungen über Teilerkennungen bis hin zu Mehrfacherkennungen der Spalte reichen, bei denen die beiden Teile als unabhängige Nadeln angesehen werden. Das Modell ResNet-50-FPN ist in keinem der Bilder in der Lage, die Kontinuität der Nadel zu erkennen. Die Modelle ResNet-50-DC5 und ResNet-101-FPN zeigen hingegen erste Erfolge: In Einzelfällen sind sie in der Lage, zwei sich überlappende Nadeln als zusammenhängend zu segmentieren, wenn auch nicht mit hoher Genauigkeit. Das Modell ResNet-101-DC5 liefert am häufigsten eine zusammenhängende Segmentierung und kann eine gute Segmentierung liefern, wenn die Nadeln in einem Winkel von 90 Grad zueinander stehen.

# 7 Diskussion der Modellleistung

Ziel dieser Arbeit ist es zu untersuchen, inwieweit Deep Learning Modelle wie Mask R-CNN für die Spitzenerkennung in REM Bildern eingesetzt werden können. Insbesondere die schwierigen und sich ständig ändernden Bildbedingungen stellen dabei eine besondere Herausforderung dar. In den folgenden Abschnitten werden die Stärken und Schwächen der trainierten Modelle durch eine Diskussion der vorgestellten Ergebnisse aufgezeigt.

## 7.1 Wahl des Backbones

Die Ergebnisse zeigen, dass die Verwendung von Dilated Convolutions zu einer signifikanten Verbesserung der Modellleistung in allen Bereichen führt. Eine Hypothese ist, dass die Modelle aufgrund der Einfachheit und Einheitlichkeit der Konturen und Spitzen erheblich von der Vergrößerung des rezeptiven Feldes profitieren. Durch die Einbeziehung von weiter entfernten Pixeln in die Kontextinformation kann deutlicher zwischen der Probenstruktur im Hintergrund und der Spitze unterschieden werden, auch wenn der Übergang in einigen Szenarien optisch fließend ist.

Eine signifikante Verbesserung der Leistung durch die Verwendung des komplexeren Backbones ResNet-101 anstelle von ResNet-50 kann aus den Ergebnissen nicht abgeleitet werden. Es sollte eine vorsichtige Abwägung zwischen der Modellleistung und dem erhöhten Rechenaufwand sowohl für das Training als auch für die Inferenz des Modells vorgenommen werden. Weiterhin sollte analysiert werden, inwieweit der Unterschied in den FN- und FP-Raten zwischen ResNet-50 und ResNet-101 durch mehr Trainingsdaten und eine Anpassung des Konfidenzschwellenwertes ausgeglichen werden kann.

Die folgenden Abschnitte der Diskussion beziehen sich nur auf die DC5-Varianten der Modelle.

## 7.2 Wahrheitswert der Vorhersagen

Wie eingangs erwähnt, ist die Minimierung von Fehlklassifikationen von großer Bedeutung, da diese zu erheblichen Schäden führen können. Die FN-Rate der auf eine Klasse trainierten Modelle liegt bei etwa acht Prozent. Dies ist für eine vollautomatische Steuerung nicht ausreichend. Mit der durchgeföhrten Evaluierung an einem einzelnen Datensatz kann nicht näher analysiert werden, unter welchen Bedingungen diese falsch-negativen Detektionen auftreten. Aufschluss kann hier die Erstellung mehrerer kleiner Datensätze geben, die speziell für einen Szenentyp erstellt werden. Auf diese Weise können problematische Bedingungen, unter denen falsch-negative Detektionen auftreten, identifiziert werden.

Die FP-Rate der Modelle liegt bei ca. vier Prozent. Dies sind bereits gute Werte und zeigen die Resistenz der Modelle gegenüber spitzenähnlichen Strukturen. Hier können jedoch durch eine detaillierte visuelle Inspektion Hintergrundstrukturen erkannt werden, die vom Modell fälschlicherweise als Spitze erkannt werden. Durch ein nachträgliches gezieltes Training auf diese Strukturen kann die FP-Rate vermutlich noch weiter gesenkt werden.

Da bekannt ist, wie viele Spitzen auf dem Prober Shuttle montiert sind, können viele falsch-positive Detektionen herausgefiltert werden, indem nur die entsprechende Anzahl von Detektionen mit den höchsten Konfidenz werten verwendet wird. Mehrfachdetektionen einer Spalte sollten durch die Verwendung eines Schwellenwertes, der das Überlappungsverhältnis der Segmentierungen von zwei detektierten Peaks regelt, herausgefiltert werden können. Wenn dieser Wert nahe bei 1 liegt, handelt es sich mit hoher Wahrscheinlichkeit um eine Mehrfachdetektion.

## 7.3 Leistungssteigerung durch Richtungsvorhersage

Die Gesamtleistung der Modelle verbesserte sich durch die zusätzliche Klassifizierung der Spitzen nach ihrer Richtung. Dies ist ein großer Vorteil, da die nachträgliche Zuordnung der detektierten Spitzen zum jeweiligen Manipulator bereits erledigt ist. Insbesondere die Segmentierung der Spitzen profitiert davon. Es wird vermutet, dass dies daran liegt, dass jeder der acht Maskenzweige des Modells auf eine bestimmte Form spezialisiert ist und eine Rotation der Spalte bei der Rekonstruktion aus den extrahierten Merkmalen nicht berücksichtigen muss.

Es wird angenommen, dass der positive Einfluss auf die FN- und FP-Raten darauf zurückzuführen ist, dass durch die Unterscheidung der acht Spitzen weniger Hintergrundstrukturen vorhanden sind, die aus einer Mischung von Merkmalen der acht Spitzen bestehen und daher fälschlicherweise als diese erkannt werden. Die Spitzen werden weniger verallgemeinert. Daher ist eine genauere Übereinstimmung mit einer Spalte aus einer bestimmten Richtung erforderlich.

In diesem Fall können auch einige Annahmen über die Detektionen getroffen werden.

- Es können nur Spitzen detektiert werden, die montiert sind.
- Es kann maximal eine Spalte jeder Klasse detektiert existieren.
- Die acht Spitzen haben alle einen vordefinierten Randbereich, aus dem sie in das Bild ragen.

Durch diese Vorannahmen sollte es möglich sein, die FN- und FP-Raten der Modelle durch nachträgliches Filtern zu minimieren.

Die in Abbildung 6.5 dargestellten Korrelationskoeffizienten deuten auf einen komplexen Zusammenhang zwischen der Modellleistung, der Anzahl der Trainingsbilder pro Klasse und der Eckposition hin. Um hier einen tieferen Einblick zu erhalten, sollte ein Datensatz erstellt werden, in dem die Anzahl der Instanzen pro Spalte gleich verteilt ist. Dadurch kann der Zusammenhang besser analysiert werden. Eventuelle Schwächen können so auch durch ein anschließendes Feintraining des Modells gezielt angegangen werden.

## 7.4 Qualität der Segmentierung

Die durch das Modell erzeugte Segmentierung der Spitzen erweist sich bei der visuellen Überprüfung als sehr situationsabhängig. Bei Vergrößerungen über 10.000 ist sie nahezu perfekt. Bei Vergrößerungen unter 10.000 ist die Segmentierung zur Spalte hin ungenau. Hier ist eine weitere Evaluierung der Modelle anhand von Datensätzen unterschiedlicher Vergrößerungsstufen erforderlich. Es ist jedoch davon auszugehen, dass Mask R-CNN aufgrund der Kompression der Bilddaten bei der Merkmalsextraktion nicht für eine qualitativ hochwertige Segmentierung sehr feiner Strukturen geeignet ist.

Ein Ansatz zur Lösung des Problems der unsauberen Segmentierung der Eckpunkte besteht darin, das Bild zweimal durch das Modell verarbeiten zu lassen. Einmal im Original und einmal um 45 Grad gedreht. Die resultierenden Segmentierungen können zusammengeführt werden, um eine genaue Segmentierung aller Spitzen zu erhalten. Dies erfordert jedoch eine schnelle Bildverarbeitung, da die zu verarbeitende Bildrate verdoppelt wird.

Die nachträgliche Analyse des Datensatzes „750img\_merged“ ergab, dass sich unter den Bildern nur 14 Bilder mit sich kreuzenden Spitzen befinden, die meist im 90-Grad-Winkel zueinander stehen. Vor diesem Hintergrund ist es erstaunlich, dass die Modelle in einigen Szenen in der Lage sind, sich kreuzende Spitzen korrekt zu segmentieren. Um jedoch Prozesse wie das Tip-Cleaning automatisieren zu können, sollte der Datensatz erweitert werden.

Da eine pixelgenaue Segmentierung nicht so wichtig ist wie eine zuverlässige Erkennung, kann man sagen, dass die Qualität der Segmentierung in den meisten Fällen mehr als ausreichend ist.

## 7.5 Genauigkeit und Probleme der Keypoint-Vorhersage

Bei kleinen Vergrößerungen, insbesondere bei Vergrößerungen unter 10.000, weichen die vorhergesagten Positionen von der Realität ab. Die Analyse der Abweichungen zeigt, dass diese normalverteilt sein können, wodurch die Ungenauigkeit durch eine Varianz charakterisiert werden kann. Die Bestimmung der Varianz ermöglicht es, für die Vorhersage einen Konfidenzbereich anzugeben, der die Wahrscheinlichkeit umfasst, dass die wahre Position innerhalb eines bestimmten Intervalls liegt.

Die Verwendung einer Normalverteilung in dieser Analyse hat den zusätzlichen Vorteil, dass sie eine statistische Grundlage für die sichere Steuerung der Manipulatoren bietet, auch wenn die Vorhersagen ungenau sind. Durch die Quantifizierung der Unsicherheit kann die Steuerung entsprechend angepasst und optimiert werden, um mögliche Fehler zu minimieren.

Ein Ansatz zur Verbesserung könnte darin bestehen, die grobe Erkennung zu verfeinern, indem der nachträglich vergrößerte Bildbereich erneut verarbeitet und die Informationen zusammengeführt werden. Diese Methode könnte die Genauigkeit in Situationen mit geringer Vergrößerung erhöhen.

Die erreichte Genauigkeit bei der Lokalisierung des vordersten Punktes der Spitzen ist jedoch insgesamt von hoher Qualität. Die quantitativen Ergebnisse, die einen AP<sup>OKS=.75</sup> von über 95% zeigen, werden durch die optische Überprüfung bestätigt. Es ist beeindruckend, dass selbst für Spitzen, deren vordere Struktur mit der darunter liegenden Probenstruktur verschmilzt, die Vorhersagen genau sind.

# 8 Zusammenfassung und Ausblick

Im Rahmen des Projekts wurde eine umfassende Pipeline erstellt, die von der Erstellung spezifischer Datensätze bis hin zum Training von Deep Learning-Modellen zur Erkennung von Messspitzen in REM-Bildern reicht. Dabei konnte eindrucksvoll gezeigt werden, dass Deep Learning der komplexen Aufgabe der Erkennung von Messspitzen gewachsen ist.

Die Implementierung der Nanocontrol-Schnittstelle und der GeminiSEM-API in Python hat wesentlich zur Automatisierung der Datenerfassung und -verarbeitung beigetragen. Diese Implementierungen haben die Effizienz der Datenerfassung erheblich gesteigert und gleichzeitig die Genauigkeit und Zuverlässigkeit der Daten gewährleistet. Durch die Automatisierung dieser Prozesse konnte ein umfangreicher Datensatz für das Nanoprobing erstellt werden. Die Entwicklung der Python-Klassen, die die Manipulatoren und das REM repräsentieren, hat eine einfache und sichere Bedienung dieser Geräte ermöglicht. Darüber hinaus wurde die Robustheit des Gesamtsystems verbessert, indem standardisierte und getestete Methoden für die Interaktion mit den Geräten bereitgestellt wurden. Trotz dieser erfolgreichen Implementierungen gibt es immer noch Raum für Verbesserungen und zukünftige Forschung. Beispielsweise könnte die Effizienz der Datenerfassung weiter verbessert werden, indem die Encoder des Prober Shuttles verwendet werden, um komplexere Pfade mit den Spitzen abzufahren.

Die Herausforderung der zeitaufwändigen Datenannotation konnte dank der entwickelten Methode zur Mehrfachverwendung einer Maske und der Verwendung von Werkzeugen wie Photoshop, die den Annotationsprozess erheblich erleichtern, erfolgreich gemeistert werden.

Die Arbeit gibt nicht nur einen detaillierten Einblick in die Stärken und Möglichkeiten von Mask R-CNN, sondern identifiziert auch Schwierigkeiten und Probleme. Die entwickelten und trainierten Modelle bieten sehr gute Voraussetzungen für die Automatisierung. Die Modelle erreichen eine hohe Genauigkeit und können bereits im jetzigen Zustand zur visuellen Kontrolle der Steuerung eingesetzt werden. Werden gefährliche Manöver der Spitzen erkannt, kann präventiv eingegriffen werden, um Kollisionen zu vermeiden. Die Modelle bilden somit die Grundlage für einen rechnergestützten Assistenten, der den hochkomplexen Prozess der Fehleranalyse vereinfacht, bestimmte Abläufe automatisiert und damit den gesamten Prozess auch für Personen mit geringer Expertise zugänglich macht.

Darüber hinaus werden Lösungsansätze vorgestellt und erprobt, die auf den identifizierten Herausforderungen aufbauen. Durch die Analyse und Diskussion der Ergebnisse konnten viele wichtige Erkenntnisse gewonnen werden, die zur Weiterentwicklung des Forschungsgebietes beitragen können. Die entwickelten Methoden ermöglichen eine schnelle Anpassung und Weiterentwicklung.

Somit stellt diese Arbeit nicht nur einen erfolgreichen Schritt in der Erkennung von Messspitzen mittels Deep Learning dar, sondern legt auch einen Grundstein für die zukünftige Entwicklung und Forschung in diesem Bereich.

## Ausblick

Die nächsten Schritte, die auf der Grundlage dieser Arbeit unternommen werden können, bestehen darin, die entwickelten Modelle produktions- und einsatzfähig zu machen. Dazu wird vorgeschlagen, die Modelle mit Hilfe der ONNX Runtime [ONN21] und Nvidia TensorRT [NVIb] für die Nvidia Jetson Plattform [NVIa] zu optimieren. Dadurch kann eine geringe Inferenzzeit erreicht werden, die für den Einsatz in Echtzeit erforderlich ist. Für eine robuste Nadelverfolgung über mehrere Bilder sollte zusätzlich zur Erkennung durch Mask R-CNN ein Tracker nachgeschaltet werden, der die Detektionen nachfolgender Bilder den vorherigen Detektionen zuordnet und dabei ebenfalls die Ansteuerungsbefehle des Nanocontrols berücksichtigt. Hierfür können Tracking-Algorithmen wie SORT oder DeepSORT verwendet werden [BGO<sup>+</sup>16] [WBP17].

Die zuvor vorgeschlagenen Verbesserungen sollten ebenfalls berücksichtigt werden. Es sollte eine eigene Evaluierungsmethode entwickelt werden, wofür eine große Anzahl speziell angepasster Datensätze benötigt wird. Eine gezielte Erweiterung des entwickelten Datensatzes ist ebenfalls sinnvoll. Neben Mask R-CNN sollten weitere Netzarchitekturen auf dem entwickelten Datensatz trainiert und verglichen werden. Netze wie das von Wang *et al.* entwickelte HRNet könnten eine noch genauere Lokalisierung ermöglichen [SXLW19] [WSC<sup>+</sup>19]. Und YOLOv7 eine schnellere Detektion, die besser für eine Echtzeitanwendung geeignet ist [WBL23]. Auch eine Kombination verschiedener Architekturen oder eine gezielte Anpassung ist denkbar.

# Abkürzungsverzeichnis

AP	Average Precision
API	Application Programming Interface
BSE	RückstreuElektronen
COCO	Comon Objects in Context
CNN	Convolutional Neural Network
DC	Dilated Convolution
DC5	Dilated Convolution at C5 Stage
FAIR	Facebook AI Research
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
FPS	Frames Per Second
GPU	Graphical Processing Unit
IoU	Intersection over Union
NC	Nanocontrol
OKS	Object Keypoint Similarity
REM	Rasterelektronenmikroskop
ReLU	Rectified Linear Unit
ResNet	Residual Network
R-CNN	Region based Convolutional Neural Network
RoI	Region of Interest
RoIAlign	Region of Interest Align
RoIPool	Region of Interest Pooling
RPN	Region Proposal Network
SE	Sekundärelektronen
SGD	Stochastic Gradient Descent
Tanh	Hyperbolic Tangent Function
TP	True Positive

# Abbildungsverzeichnis

2.1	Ein voll bestücktes Prober Shuttle (PS8) . . . . .	3
2.2	Messspitzen der Firma Mesoscope in verschiedenen Ausführungen. Nach Spitzentradii sortiert 2.2(a) und abgesetzt auf den Kontakten mehrerer Transistoren 2.2(b) . . . . .	4
2.3	Grundlegender Aufbau eines Rasterelektronenmikroskops. Quelle: nature.com [SRP19] . . . . .	5
2.4	Visualisierung der Messspitzen bei verschiedenen Beschleunigungsspannungen.	6
2.5	Arbeitsplatz bei Kleindiek. . . . .	7
3.1	Messspitzen, die zur Lokalisierung und Differenzierung von Bounding Boxes umgeben sind. Quelle: Kleindiek Nanotechnik GmbH . . . . .	8
3.2	Messspitzen, deren spezifische Formen durch Instanz Segmentierung hervorgehoben werden. Quelle: Kleindiek Nanotechnik GmbH . . . . .	9
3.3	Messspitzen, deren vordere Punkte – die wesentlichen Kontaktpunkte bei der Probenmessung – deutlich markiert sind. Quelle: Kleindiek Nanotechnik GmbH	9
3.4	Ein mehrschichtiges Perzepron mit 4 Eingängen und 2 Ausgängen. Zwischen Ein- und Ausgangsschacht liegen mehrere verborgene Schichten. . . . .	10
3.5	Graphen der Binären Stufen-, Sigmoid- und ReLU-Funktion . . . . .	11
3.6	Zero-padded Bild. Der Stride ist eins. Somit entspricht die Ausgangsgröße der Faltung der Eingangsgröße. Der Sobel-Operator wird als Kernel verwendet [KVB88]. . . . .	13
3.7	(a) 1-dilated convolution mit 3x3 rezeptivem Feld, (b) 2-dilated convolution mit 7x7 rezeptivem Feld, (c) 4-dilated convolution mit 15x15 rezeptivem Feld .	13
3.8	Pooling dient der Dimensionsreduktion unter Beibehaltung der aussagekräftigsten Merkmale, wobei Methoden wie Max-Pooling und Average-Pooling zum Einsatz kommen. . . . .	14
3.9	Mask R-CNN Architektur . . . . .	15
3.10	Architektur des ResNet-34. ResNet Netze bestehen aus mehreren Ebenen. Jede Ebene extrahiert Merkmale auf einer anderen Abstraktionsebene. . . . .	16
3.11	Das in dieser Arbeit verwendete Annotation Format, abgespeichert als JSON Datei. Datensätze in COCO Format können mit den meisten Frameworks nativ zum Training Neuronaler Netze verwendet werden. . . . .	20
5.1	Befehle, um die Position der Manipulatoren im Grob- und Feinbereich auszulesen und zu ändern. Auszug dem Nanocontrol Software Manual. . . . .	23
5.2	Die Implementierung der Schnittstelle des Nanocontrols in Python ermöglicht eine flexible Steuerung der Manipulatoren. . . . .	24
5.3	Die SemAPI wird von ZEISS in Form einer ocx-Datei zur Verfügung gestellt. Die Sem-Klasse bietet eine abstrahierte und angepasste Form der API, die speziell für die effiziente Erstellung von Datensätzen geeignet ist. . . . .	25

5.4	Messspitzen, dargestellt bei unterschiedlichen Vergrößerungen. Bei einer hohen Vergrößerung sind die Spitzen klar erkennbar. Je kleiner die Vergrößerung, desto schwerer ist es, den Apex der Spitze zu lokalisieren. . . . .	26
5.5	Vier vordefinierte Pfade für das Zurückziehen der Messspitzen. Der Verfahrweg wird mit der Vergrößerung skaliert. Die Probe wird zyklisch verfahren. . . . .	29
5.6	Eine Szene aufgenommen mit verschiedenen Parametern. Deutlich zu erkennen ist der Unterschied zwischen dem SE2 Detektor in (a), (c) und (e) und dem InLens Detektor in (b) und (d). Die Probenstruktur ist sehr unterschiedlich. . . . .	30
5.7	Bild und zugehörige annotierte Maske. Die HEX-Farbwerte sind hier in die Spitzen eingezeichnet. . . . .	31
5.8	Darstellung der Verzeichnisstruktur des Datensatzes vor und nach der Konvertierung. Die Anzahl der erstellten Ordner hängt von der Wahl der Parameter des Konvertierungsskripts ab. . . . .	33
5.9	Skript-Parameter für die Konvertierung von Datensätzen. . . . .	33
5.10	Anzahl der Instanzen jeder Spitzengruppe von „750img_merged“. Dies ermöglicht einen Vergleich zwischen den Klassenverteilungen in den Trainings- und Validierungsdaten. . . . .	34
5.11	Die verwendete Mask R-CNN Architektur, der Keypoint-Zweig wird angefügt. Die Keypoints werden als eine One-Hot Maske ausgegeben. . . . .	36
5.12	Der Verlauf der Lernrate über die Epochen, logarithmisch dargestellt. Lernrate der Aufwärmepochen in Blau, konstante Lernrate in Rot. . . . .	38
6.1	Gesamter Fehler der Modelle über die Trainingsepochen. Bei Iteration 1800 beginnt das training des Backbones. . . . .	40
6.2	Entwicklung der AP <sup><i>IoU=.75</i></sup> für die verschiedenen Backbones während der Trainingsepisoden. . . . .	41
6.3	Darstellung der AP <sup><i>OKS=.75</i></sup> der Keypoint-Vorhersage für die verschiedenen Backbones über die Trainingsepisoden. . . . .	42
6.4	Visualisierung der Detektionen des ResNet-50-FPN-Netzes mit einem Konfidenzschwellenwert von 90%. . . . .	43
6.5	AP <sup>.75</sup> -Werte für die Bounding-Box-Vorhersage, die Segmentierung und die Keypoint-Erkennung. Für die Modelle, welche die Spitzen anhand ihrer Richtung unterscheiden. . . . .	44
6.6	AP <sup>.75</sup> -Werte der Vorhersagen, getrennt nach den acht Klassen und die Anzahl der Instanzen pro Klasse im Trainingsdatensatz. Es wird eine mögliche Korrelation zwischen der Menge der Trainingsdaten und der Vorhersagegenauigkeit untersucht. . . . .	45
6.7	Acht Spitzen beim Kontaktieren einer Probe. Aufgenommen bei 100.000-facher Vergrößerung. Die Segmentierung der jeweiligen Netze ist über das Bild gelegt. . . . .	47
6.8	Segmentierung und Keypoints der verschiedenen Netzvarianten. Aufgenommen bei 2.000-facher Vergrößerung. Der zentrale Bereich ist vergrößert dargestellt. . . . .	48
6.9	Zwei gekreuzte Nadeln, dieses Szenario tritt bei der Reinigung der Spitzen auf. . . . .	49

# Tabellenverzeichnis

3.1	Übersicht der für diese Arbeit relevanten Average Precision (AP) Metriken . . . . .	20
5.1	Auszug aus den genutzten analogen Parametern des ZEISS GeminiSEM. . . . .	28
5.2	Auszug aus den genutzten digitalen Parametern des ZEISS GeminiSEM. . . . .	28
5.3	Zusammenfassung der Datensätze. Alle Datensätze haben ein Train-Val-Verhältnis von 0.9 . . . . .	34
5.4	Genutzte vortrainierte Modelle aus dem Detectron2 Model Zoo. . . . .	37
6.1	Mittelwert der letzten 200 Epochen von AP, $AP^{IoU=.5}$ , $AP^{IoU=.75}$ der Segmentation für die verschiedenen Modelle. . . . .	41
6.2	AP, $AP^{OKS=.5}$ , $AP^{OKS=.75}$ Werte der Keypoint-Vorhersage für die verschiedenen Modelle. . . . .	42
6.3	Falsch-negativ und falsch-positiv Raten von Mask R-CNN mit verschiedenen Backbones. . . . .	43
6.4	Vergleich der Werte $AP_{BB}^{IoU=.75}$ , $AP_S^{IoU=.75}$ , $AP_K^{OKS=.75}$ des auf ein oder acht Klassen trainierten Modells. Aufgeteilt nach Backbone-Architektur. . . . .	44
6.5	Berechnete Korrelationskoeffizienten zwischen der Anzahl der Trainingsbilder pro Spitze und der Genauigkeit des Modells sowie zwischen der Eckposition der Spitzen und der Genauigkeit. . . . .	45

# Literaturverzeichnis

- [ADIP21] Apicella, Andrea, Francesco Donnarumma, Francesco Isgrò und Roberto Prevete: *A survey on modern trainable activation functions*. Neural Networks, 138:14–32, 2021.
- [Ado] Adobe Inc.: *Adobe Photoshop*.
- [AI423] AI4Life: *BioImage.IO datasets*. <https://bioimage.io/#/?type=dataset>, 2023. Letzter Zugriff: 21. August 2023.
- [AMAZ17] Albawi, Saad, Tareq Abed Mohammed und Saad Al-Zawi: *Understanding of a convolutional neural network*. In: *2017 International Conference on Engineering and Technology (ICET)*, Seiten 1–6, 2017.
- [Ato23] Atoum, I. A.: *Adaptive Rectified Linear Unit (Arelu) for Classification Problems to Solve Dying Problem in Deep Learning*. International Journal of Advanced Computer Science and Applications, 14(2):97–102, 2023.
- [BGO<sup>+</sup>16] Bewley, Alex, Zongyuan Ge, Lionel Ott, Fabio Ramos und Ben Upcroft: *Simple online and realtime tracking*. In: *2016 IEEE International Conference on Image Processing (ICIP)*, Seiten 3464–3468, 2016.
- [Blo62] Block, H. D.: *The Perceptron: A Model for Brain Functioning. I.* Rev. Mod. Phys., 34:123–135, Jan 1962.
- [Bra00] Bradski, G.: *The OpenCV Library*. Dr. Dobb’s Journal of Software Tools, 2000.
- [Cla15] Clark, Alex: *Pillow (PIL Fork) Documentation*, 2015.
- [CM20] Cutkosky, Ashok und Harsh Mehta: *Momentum Improves Normalized SGD*. In: III, Hal Daumé und Aarti Singh (Herausgeber): *Proceedings of the 37th International Conference on Machine Learning*, Band 119 der Reihe *Proceedings of Machine Learning Research*, Seiten 2260–2268. PMLR, 13–18 Jul 2020.
- [COC23] COCO Consortium: *COCO Dataformat*. <https://cocodataset.org>, 2023. Letzter Zugriff: 21. August 2023.
- [DDS<sup>+</sup>09] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li und Li Fei-Fei: *Imagenet: A large-scale hierarchical image database*. In: *2009 IEEE conference on computer vision and pattern recognition*, Seiten 248–255. Ieee, 2009.
- [EAAAM22] Elharrouss, Omar, Younes Akbari, Noor Almaadeed und Somaya Al-Maadeed: *Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches*, 2022.

- [Eij21] Eijenstein, Chris: *image-to-coco-json-converter*. <https://github.com/chrise96/image-to-coco-json-converter>, 2021. Letzter Zugriff: 21. August 2023.
- [FK21] Frei, M. und F.E. Kruis: *FibeR-CNN: Expanding Mask R-CNN to improve image-based fiber analysis*. Powder Technology, 377:974–991, jan 2021.
- [Fre22] Frei, Max: *FibeR-CNN*. <https://github.com/maxfrei750/FibeR-CNN>, 2022. Letzter Zugriff: 21. August 2023.
- [G<sup>+</sup>] Gillies, Sean et al.: *Shapely: manipulation and analysis of geometric objects*, 2007–.
- [GD98] Gardner, M.W und S.R Dorling: *Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences*. Atmospheric Environment, 32(14):2627–2636, 1998.
- [GDDM16] Girshick, Ross, Jeff Donahue, Trevor Darrell und Jitendra Malik: *Region-Based Convolutional Networks for Accurate Object Detection and Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1):142–158, 2016.
- [GDG<sup>+</sup>17] Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesołowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia und Kaiming He: *Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour*, 2017.
- [Gir15] Girshick, Ross: *Fast R-CNN*. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [Goo23] Google Inc.: *Colaboratory*. <https://research.google.com/colaboratory/faq.html>, 2023. Letzter Zugriff: 21. August 2023.
- [Har21] Hartung, Roman: *Unglaubliche Präzision - Wir untersuchen einen einzelnen TSMC 7nm Transistor Kleindiek Teil 1/2*. <https://www.youtube.com/watch?v=zY5WC9kMS0g&t>, 2021. Letzter Zugriff: 21. August 2023.
- [Haw04] Hawkins, Douglas M.: *The Problem of Overfitting*. Journal of Chemical Information and Computer Sciences, 44(1):1–12, 2004. PMID: 14741005.
- [HBDS16] Hosang, Jan, Rodrigo Benenson, Piotr Dollar und Bernt Schiele: *What Makes for Effective Detection Proposals?* IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(4):814–830, apr 2016.
- [HGDG17] He, Kaiming, Georgia Gkioxari, Piotr Dollar und Ross Girshick: *Mask R-CNN*. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [HZRS16a] He, Kaiming, Xiangyu Zhang, Shaoqing Ren und Jian Sun: *Deep Residual Learning for Image Recognition*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [HZRS16b] He, Kaiming, Xiangyu Zhang, Shaoqing Ren und Jian Sun: *Identity Mappings in Deep Residual Networks*, 2016.

- [iA93] Amari, Shun ichi: *Backpropagation and stochastic gradient descent method*. Neurocomputing, 5(4):185–196, 1993.
- [Ink16] Inkson, B.J.: *2 - Scanning electron microscopy (SEM) and transmission electron microscopy (TEM) for materials characterization*. In: Hübschen, Gerhard, Iris Altpeter, Ralf Tschuncky und Hans-Georg Herrmann (Herausgeber): *Materials Characterization Using Nondestructive Evaluation (NDE) Methods*, Seiten 17–43. Woodhead Publishing, 2016.
- [KB17] Kingma, Diederik P. und Jimmy Ba: *Adam: A Method for Stochastic Optimization*, 2017.
- [KC20] Kandel, Ibrahem und Mauro Castelli: *The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset*. ICT Express, 6(4):312–315, 2020.
- [Kle23] Kleindiek, Stephan: *Kleindiek Nanotechnik GmbH*. <https://www.nanotechnik.com/>, 2023. Letzter Zugriff: 21. August 2023.
- [KVB88] Kanopoulos, Nick, Nagesh Vasanthavada und Robert L Baker: *Design of an image edge detection filter using the Sobel operator*. IEEE Journal of solid-state circuits, 23(2):358–367, 1988.
- [LDG<sup>+</sup>16] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan und Serge Belongie: *Feature Pyramid Networks for Object Detection*, 2016.
- [LJH<sup>+</sup>19] Liu, Liyuan, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao und Jiawei Han: *On the Variance of the Adaptive Learning Rate and Beyond*, 2019.
- [LLX<sup>+</sup>18] Liu, Jing, Weifu Li, Chi Xiao, Bei Hong, Qiwei Xie und Hua Han: *Automatic Detection and Segmentation of Mitochondria from SEM Images using Deep Neural Network*. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Seiten 628–631, 2018.
- [Mat23] MathWorks: *Getting Started with Mask R-CNN for Instance Segmentation*. <https://de.mathworks.com/help/vision/ug/getting-started-with-mask-r-cnn-for-instance-segmentation.html>, 2023. Letzter Zugriff: 21. August 2023.
- [NVF20] NVIDIA, Péter Vingermann und Frank H.P. Fitzek: *CUDA, release: 10.2.89*, 2020.
- [NVIa] NVIDIA: *NVIDIA Jetson*. <https://developer.nvidia.com/embedded/faq>. Letzter Zugriff: 21. August 2023.
- [NVIb] NVIDIA: *NVIDIA TensorRT*. <https://developer.nvidia.com/tensorrt>. Letzter Zugriff: 21. August 2023.
- [ON15] O’Shea, Keiron und Ryan Nash: *An Introduction to Convolutional Neural Networks*, 2015.

- [ONN21] ONNX Runtime developers: *ONNX Runtime*. <https://onnxruntime.ai/>, 2021. Letzter Zugriff: 21. August 2023.
- [OP17] Orhan, A. Emin und Xaq Pitkow: *Skip Connections Eliminate Singularities*, 2017.
- [Pan22] Pandey, Abhishek Kumar: *Convolution, Padding, Stride, and Pooling in CNN*. <https://medium.com/analytics-vidhya/convolution-padding-stride-and-pooling-in-cnn-13dc1f3ada26>, 2022. Letzter Zugriff: 21. August 2023.
- [Pap23a] Papers with Code: *Papers with Code - Computer Vision*. <https://paperswithcode.com/area/computer-vision>, 2023. Letzter Zugriff: 21. August 2023.
- [Pap23b] Papers with Code: *Papers with Code - Datasets*. <https://paperswithcode.com/datasets>, 2023. Letzter Zugriff: 21. August 2023.
- [PGM<sup>+</sup>19] Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai und Soumith Chintala: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In: *Advances in Neural Information Processing Systems 32*, Seiten 8024–8035. Curran Associates, Inc., 2019.
- [Qay22] Qayyum, Rafay: *Introduction To Pooling Layers In CNN*. <https://towardsai.net/p/1/introduction-to-pooling-layers-in-cnn>, 2022. Letzter Zugriff: 21. August 2023.
- [Rak19] Rakhecha, Aditya: *Understanding Learning Rate*. <https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate>, 2019. Letzter Zugriff: 21. August 2023.
- [RHGS15] Ren, Shaoqing, Kaiming He, Ross B. Girshick und Jian Sun: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. CoRR, abs/1506.01497, 2015.
- [RHW86] Rumelhart, David E., Geoffrey E. Hinton und Ronald J. Williams: *Learning representations by back-propagating errors*. Nature, 323(6088):533–536, 1986.
- [Ros58] Rosenblatt, F.: *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, 65(6):386–408, 1958.
- [SGS15] Srivastava, Rupesh Kumar, Klaus Greff und Jürgen Schmidhuber: *Training Very Deep Networks*, 2015.

- [SMDH13] Sutskever, Ilya, James Martens, George Dahl und Geoffrey Hinton: *On the importance of initialization and momentum in deep learning*. In: Dasgupta, Sanjoy und David McAllester (Herausgeber): *Proceedings of the 30th International Conference on Machine Learning*, Band 28 der Reihe *Proceedings of Machine Learning Research*, Seiten 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.
- [SRP19] Shah, Furqan A., Krisztina Ruscsák und Anders Palmquist: *50 years of scanning electron microscopy of bone—a comprehensive overview of the important discoveries made and insights gained into bone material properties in health, disease, and taphonomy*. *Bone Research*, 7(1):15, 2019.
- [SXLW19] Sun, Ke, Bin Xiao, Dong Liu und Jingdong Wang: *Deep High-Resolution Representation Learning for Human Pose Estimation*. In: *CVPR*, 2019.
- [VdWSNI<sup>+</sup>14] Walt, Stefan Van der, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart und Tony Yu: *scikit-image: image processing in Python*. *PeerJ*, 2:e453, 2014.
- [VGO<sup>+</sup>20] Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt und SciPy 1.0 Contributors: *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. *Nature Methods*, 17:261–272, 2020.
- [WBL23] Wang, Chien-Yao, Alexey Bochkovskiy und Hong-Yuan Mark Liao: *YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors*. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seiten 7464–7475, June 2023.
- [WBP17] Wojke, Nicolai, Alex Bewley und Dietrich Paulus: *Simple Online and Real-time Tracking with a Deep Association Metric*. In: *2017 IEEE International Conference on Image Processing (ICIP)*, Seiten 3645–3649. IEEE, 2017.
- [Wei23] Weisstein, Eric W.: *Convolution*. <https://mathworld.wolfram.com/Convolution.html>, 2023. Letzter Zugriff: 21. August 2023.
- [WKM<sup>+</sup>19] Wu, Yuxin, Alexander Kirillov, Francisco Massa, Wan-Yen Lo und Ross Girshick: *Detectron2*. <https://github.com/facebookresearch/detectron2>, 2019. Letzter Zugriff: 21. August 2023.
- [WMZT22] Wang, Qi, Yue Ma, Kun Zhao und Yingjie Tian: *A Comprehensive Survey of Loss Functions in Machine Learning*. *Annals of Data Science*, 9(2):187–212, 2022.

- [WSC<sup>+</sup>19] Wang, Jingdong, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu und Bin Xiao: *Deep High-Resolution Representation Learning for Visual Recognition*. TPAMI, 2019.
- [Yin19] Ying, Xue: *An Overview of Overfitting and its Solutions*. Journal of Physics: Conference Series, 1168(2):022022, feb 2019.
- [YK16] Yu, Fisher und Vladlen Koltun: *Multi-scale context aggregation by dilated convolutions*. In: *International Conference on Learning Representations*, 2016.
- [YKF17] Yu, Fisher, Vladlen Koltun und Thomas Funkhouser: *Dilated Residual Networks*. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [ZAWJ07] Zhou, Weilie, Robert Apkarian, Zhong Lin Wang und David Joy: *Fundamentals of Scanning Electron Microscopy (SEM)*, Seiten 1–40. Springer New York, New York, NY, 2007.
- [ZEI23] ZEISS: *ZEISS GeminiSEM*. <https://www.zeiss.com/microscopy/de/produkte/sem-und-fib-sem/sem/die-geminisem-produktfamilie.html>, 2023. Letzter Zugriff: 21. August 2023.