

Evaluation verschiedener  
Bildverarbeitungsmethoden und neuronaler  
Netze sowie Implementierung eines  
Bildverarbeitungsverfahrens zur  
Segmentierung und Auswertung von  
Füllständen in Bildern

## STUDIENARBEIT

für die Prüfung zum  
Bachelor of Science  
des Studienganges Angewandte Informatik

an der  
Dualen Hochschule Baden-Württemberg Karlsruhe

von

**Lukas Hörnle & Marc Gökce**

Abgabedatum 22.05.2023

Bearbeitungszeitraum	300 Stunden x 2
Matrikelnummer	6828354 & 4587590
Kurs	TINF20B4
Ausbildungsfirma	CAS Software AG & 1&1 Karlsruhe
Gutachter der Studienakademie	Ralph Lausen

## **Erklärung**

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema: »Evaluation verschiedener Bildverarbeitungsmethoden und neuronaler Netze sowie Implementierung eines Bildverarbeitungsverfahrens zur Segmentierung und Auswertung von Füllständen in Bildern« selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt. \_\_\_\_\_

---

Ort      Datum

Unterschrift

*Sofern vom Dualen Partner ein Sperrvermerk gewünscht wird, ist folgende Formulierung zu verwenden:*

## **Sperrvermerk**

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung vom Dualen Partner vorliegt.

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>8</b>
<b>2 Grundlagen der Bildverarbeitung und der Skalierung von Bildern</b>	<b>12</b>
2.1 Einblick in die Bildverarbeitung . . . . .	12
2.2 Skalierung von Bildern . . . . .	14
2.2.1 Arten der Skalierungen: Interpolation und Skalierung . . . . .	14
2.2.2 Bildformate . . . . .	15
2.2.3 Wichtige Aspekte der Skalierung . . . . .	20
2.3 Echtzeitverarbeitung von Bildern . . . . .	21
2.4 Anwendungen von Skalierungsmethoden . . . . .	23
<b>3 Klassische Skalierungsmethoden</b>	<b>24</b>
3.1 Pixel-Verdopplung . . . . .	24
3.2 Nearest-Neighbor-Interpolation . . . . .	26
3.3 Bilineare Interpolation . . . . .	27
3.4 Bikubische Interpolation . . . . .	30
3.4.1 Mathematische Grundlagen . . . . .	30
3.4.2 Algorithmische Implementierung . . . . .	31
3.4.3 Analyse der Leistung . . . . .	32
3.4.4 Implementation . . . . .	32
3.4.5 Anwendungen . . . . .	35
3.4.6 Erweiterungen und Variationen . . . . .	36

<b>INHALTSVERZEICHNIS</b>	<b>3</b>
3.5 Lanczos-Interpolation . . . . .	36
3.5.1 Mathematische Grundlage . . . . .	37
3.5.2 Praktische Anwendung . . . . .	38
3.6 Zusammenfassung von Bildverarbeitungstechniken . . . . .	40
3.6.1 Pixelverdopplung . . . . .	40
3.6.2 Nearest Neighbour Interpolation . . . . .	40
3.6.3 Bilineare Interpolation . . . . .	41
3.6.4 Bikubische Interpolation . . . . .	41
3.6.5 Lanczos Interpolation . . . . .	41
3.7 Analyse der Leistung . . . . .	41
3.7.1 Vergleich und Zusammenfassung . . . . .	45
<b>4 Fortgeschrittene Skalierungsmethoden</b>	<b>46</b>
4.1 Convolutional Neural Networks / Deep learning . . . . .	47
4.1.1 Grundlagen von Convolutional Neural Networks (CNNs) . . . . .	47
4.1.2 Architekturen von CNNs . . . . .	47
4.1.3 Anwendungen von CNNs . . . . .	48
4.1.4 Transfer Learning mit CNNs . . . . .	49
4.1.5 Limitationen von CNNs und aktuelle Forschungsziele . . . . .	50
4.2 Super Resolution . . . . .	51
4.2.1 Grundlagen von Super Resolution . . . . .	51
4.2.2 Super Resolution-Methoden auf Basis von Deep Learning . . . . .	51
4.3 Generative Adversarial Networks . . . . .	55
4.3.1 Grundlagen von Generative Adversarial Networks (GANs) . . . . .	55
4.3.2 Architekturen von GANs . . . . .	55
4.3.3 Anwendungen von GANs . . . . .	56
4.3.4 Training von GANs und Evaluierung von generierten Ergebnissen	59
4.3.5 Ethische und soziale Implikationen von GANs . . . . .	60
4.4 Multiskalenskalierung . . . . .	61
4.4.1 Grundlagen von Multiskalenskalierung . . . . .	61

<b>INHALTSVERZEICHNIS</b>	<b>4</b>
4.4.2 Methoden zur Multiskalenanalyse . . . . .	62
4.4.3 Anwendungen von Multiscale-Skalierung . . . . .	62
4.4.4 Limitierungen und zukünftige Forschungsziele von Multiscale-Skalierung	63
4.5 Vor- und Nachteile der fortgeschrittenen Methoden . . . . .	64
<b>5 Evaluation von Skalierungsmethoden</b>	<b>66</b>
5.1 Qualitätsmetriken von Skalierungsmethoden . . . . .	66
5.1.1 Peak Signal-to-Noise Ratio (PSNR) . . . . .	66
5.1.2 Structural Similarity Index Measure (SSIM) . . . . .	68
5.1.3 Mean Opinion Score . . . . .	69
5.1.4 Peak Signal-to-Noise Ratio der Y-Komponente (PSNR-Y) . . . .	70
5.1.5 Computational Speed . . . . .	70
5.1.6 Root-Mean-Square Error . . . . .	71
5.2 Kriterien zur Auswahl der Skalierungsmethode . . . . .	72
5.2.1 Bildvergleich und visuelle Bewertung . . . . .	72
5.2.2 Effektivität und Effizienz . . . . .	73
5.2.3 Zukunftsaussichten und Herausforderungen . . . . .	74
<b>6 Einführung in Deep Learning</b>	<b>77</b>
6.1 Motivation der Studie von CNNs und deren Training . . . . .	78
6.2 Struktur der Arbeit . . . . .	79
<b>7 Grundlagen von Convolutional Neural Networks</b>	<b>81</b>
7.1 Wie lernen Maschinen? . . . . .	81
7.1.1 Definition von maschinellem Lernen . . . . .	81
7.1.2 Deep Learning . . . . .	82
7.1.3 Grundlagen von Convolutional Neural Networks . . . . .	83
7.1.4 Definition und Anwendungen von Convolutional Neural Networks (CNNs)	85

7.1.5	Wie unterscheiden sich CNNs von anderen neuronalen Netzwerkarchitekturen? . . . . .	88
7.1.6	Warum sind Convolutional Neural Networks besonders nützlich für Bild- und Videodaten? . . . . .	91
7.2	Anwendungen von CNNs . . . . .	93
7.2.1	Bildklassifikation . . . . .	94
7.2.2	Objekterkennung . . . . .	94
7.2.3	Gesichtserkennung . . . . .	95
7.2.4	Natural Language Processing (NLP) . . . . .	96
7.2.5	Weitere Anwendungen . . . . .	96
7.3	Architektur von CNNs . . . . .	97
7.3.1	Input Layer . . . . .	97
7.3.2	Hidden Layers . . . . .	98
7.3.3	Output Layers . . . . .	99
7.4	Convolutional layers . . . . .	100
7.4.1	Pooling Layers . . . . .	100
7.4.2	Fully Connected Layers . . . . .	102
7.4.3	Overfitting und Dropout . . . . .	102
7.4.4	Aktivierungsfunktionen . . . . .	103
7.4.5	Verlustfunktionen . . . . .	106
<b>8</b>	<b>Datenverarbeitung (Data Preprocessing)</b> . . . . .	<b>108</b>
8.1	Was sind Daten? Natur der Daten . . . . .	108
8.2	Die Bedeutung von qualitativ hochwertigen Daten . . . . .	109
8.3	Datennormalisierung (Data Normalization) . . . . .	110
8.4	Datenaugmentierung (Data Augmentation) . . . . .	112
8.5	Fazit . . . . .	114
<b>9</b>	<b>Architektur des DeepLabV3+ Modells mit ResNet-50 Backbone</b> . . . . .	<b>115</b>
9.1	Backbone-Netzwerk . . . . .	115

<b>INHALTSVERZEICHNIS</b>	<b>6</b>
9.2 Prediction-Head . . . . .	116
9.3 Laden des Modells . . . . .	117
9.4 Modifikation der letzten Schicht . . . . .	118
9.5 Verschieben des Modells auf ein Gerät . . . . .	118
9.6 Erstellen des Optimierers . . . . .	118
<b>10 Convolutional Neural Network Trainings Prozess</b>	<b>119</b>
10.1 Was ist Training? . . . . .	119
10.2 Trainingsprozess . . . . .	120
10.3 Stochastic Gradient Descent (SGD) . . . . .	121
10.4 Backpropagation . . . . .	122
10.5 Hyperparameter-Tuning . . . . .	123
10.6 Regularisierungstechniken . . . . .	124
<b>11 Transfer Learning</b>	<b>126</b>
11.1 Einführung in Transfer Learning . . . . .	126
11.2 Pre-Trained Models . . . . .	127
11.3 Fine-tuning von vortrainierten Modellen . . . . .	128
11.4 Verwendung von vortrainierten Modellen . . . . .	130
11.5 Anwendung von DeepLabV3 ResNet50 . . . . .	131
<b>12 Herausforderungen und Lösungen im Bereich Deep Learning</b>	<b>134</b>
12.1 Overfitting und Underfitting . . . . .	134
12.1.1 Overfitting . . . . .	134
12.1.2 Underfitting . . . . .	136
12.2 Vanishing and Exploding Gradients . . . . .	137
12.2.1 Ursachen der vanishing Gradients . . . . .	137
12.2.2 Ursachen der exploding Gradients . . . . .	138
12.2.3 Negative Auswirkungen der vanishing und exploding Gradients . .	138
12.2.4 Lösungsansätze für vanishing Gradients . . . . .	139

<b>INHALTSVERZEICHNIS</b>	<b>7</b>
12.2.5 Lösungsansätze für exploding Gradients . . . . .	140
12.3 Gradient Descent Optimization . . . . .	140
12.3.1 Standard-Gradientenabstieg . . . . .	141
12.3.2 Stochastischer Gradientenabstieg (SGD) . . . . .	141
12.3.3 Mini-Batch Gradientenabstieg . . . . .	142
12.3.4 Batch Normalisierung . . . . .	142
<b>13 Tools and Frameworks for CNN Training</b>	<b>143</b>
13.1 PyTorch . . . . .	143
13.2 TensorFlow . . . . .	145
13.3 Keras . . . . .	146
13.4 Caffe . . . . .	147
13.5 Weitere beliebte Frameworks . . . . .	149
<b>14 Schlusswort</b>	<b>150</b>
14.1 Zusammenfassung und Fazit der Arbeit . . . . .	150
14.2 Perspektiven für zukünftige Forschungen . . . . .	151
14.2.1 Optimierung der Bildverarbeitungsmethoden . . . . .	151
14.2.2 Integration fortschrittlicher neuronaler Netze . . . . .	152
14.2.3 Erweiterung des Anwendungsbereichs . . . . .	152
14.2.4 Evaluation auf größeren und vielfältigeren Datensätzen . . . . .	152
14.2.5 Integration von Echtzeitverarbeitung und Echtzeitfeedback . . . . .	153
14.2.6 Integration von KI-gestütztem Lernen . . . . .	153
14.2.7 Berücksichtigung von Datenschutz und Sicherheit . . . . .	154
14.2.8 Langzeitstudien zur Zuverlässigkeit und Stabilität . . . . .	154
<b>Anhang</b>	<b>161</b>
<b>Index</b>	<b>161</b>
<b>Literaturverzeichnis</b>	<b>161</b>

# Kapitel 1

## Einleitung



Abbildung 1.1: “The Pale Blue Dot” Feb. 14, 1990, by NASA<sup>1</sup>.

”Ein Bild sagt mehr aus als tausend Worte.“ Dieses bekannte Sprichwort drückt aus, wie mächtig Bilder als Kommunikationsmittel sind. Bilder beinhalten Informationen, vermitteln Emotionen, erzählen Geschichten und sind ein Fenster in die Vergangenheit. Bilder sind in der modernen Gesellschaft omnipräsent und im Alltag digital als auch analog unentbehrlich. Bilder unterscheiden sich je nach Aufnahme in den verschiedenen Eigenschaften ihrer Speicherung und Darstellung. Zwei dieser Eigenschaften sind die Größe

<sup>1</sup>NASA/JPL-CALTECH [Feb. 1990]. *The Pale Blue Dot is a photograph of Earth taken Feb. 14, 1990, by NASA’s Voyager 1 at a distance of 6 billion kilometers from the Sun. The image inspired the title of scientist Carl Sagan’s book, ”Pale Blue Dot: A Vision of the Human Future in Space,” in which he wrote: ”Look again at that dot. That’s here. That’s home. That’s us.“ <https://solarsystem.nasa.gov/resources/536/voyager-1s-pale-blue-dot/>*

und die Auflösung eines Bildes. Die Größe eines digitalen Bildes gibt an, wie viele Pixel es enthält, während die Auflösung eines Bildes angibt, wie viele Pixel pro Flächeneinheit vorhanden sind Pixel per Inch (PPI).

Die Größe und die Auflösung eines Bildes haben Einfluss auf seine Qualität und seinen Speicherplatzbedarf. Um ein Bild für einen bestimmten Zweck zu nutzen, muss es häufig in seiner Größe und oder Auflösung verändert werden. Der Vorgang zur Veränderung der Größe und Auflösung wird als Bildskalierung bezeichnet<sup>23</sup> und ist eine grundlegende Operation in der digitalen Bildverarbeitung. Bildskalierung ist eine grundlegende Methode der Bildverarbeitung. Bildskalierung erlaubt die Änderung der Größe eines digitalen Bildes. Eine Gute Bildskalierung misst sich an ihren Eigenschaften in den Bereichen Rechenaufwand und Qualitätsverlust. Besonders wichtig ist der Qualitätsverlust, wenn man Bilder größer skaliert. Ein geeignetes Modell um diesen Prozess zu erklären ist das übertragen einer Zeichnung von einem kleinen Papier auf eine große Leinwand. Wird die Zeichnung lediglich unbedacht vergrößert, wird diese unscharf und verliert an Details. Das Ziel einer guten Bildskalierung ist es, diesen Effekt zu verhindern und die Zeichnung großenunabhängig scharf und detailreich darzustellen.

Es gibt viele verschiedene Methoden, um die Größe eines Bildes zu ändern. Klassische Methoden verwenden Interpolationstechniken, die neue Pixel aus den vorhandenen Pixeln berechnen. Diese Methoden sind schnell und stellen einen geringen Rechenaufwand in Kombination mit geringer Komplexität dar. Jedoch kommt es mit diesen Algorithmen oft zu Qualitätsverlusten oder der Erzeugung von Artefakten. Moderne Anwendungen zur Skalierung von Bildern verwenden Deep-Learning-Techniken wie Convolutional Neural Networks Convolutional neural network (CNN) oder Generative Adversarial Networks Generative Adversarial Network (GAN), die neue Pixel aus einem trainierten Modell erzeugen. Diese neuen Methoden sind komplex und benötigen mehr Rechenaufwand,

---

<sup>2</sup>TECH-LIB [2020]. *Image Scaling Definition*. <http://www.dante.de>.

<sup>3</sup>Wikipedia CONTRIBUTORS [2023]. *Bildskalierung Definition*. [https://en.wikipedia.org/wiki/Image\\_scaling](https://en.wikipedia.org/wiki/Image_scaling).

können allerdings die Qualität des Bildes verbessern oder kreative Effekte erstellen.

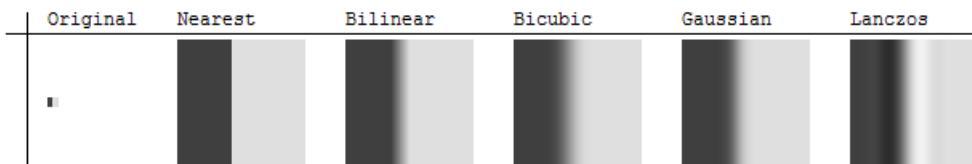


Abbildung 1.2: Verschiedene Beispiele von upscaling Algorithmen[WHUBER 2011].

Die Bildskalierung hat heute viele Anwendungen in verschiedenen Bereichen wie beispielweise Webdesign, Fotografie, Druck oder Videotechnik. Es gibt auch in modernen Anwendungen verschiedene Arten von Skalierungsverfahren, die sich in ihrer Funktionsweise und ihrem Ergebnis unterscheiden. Diese Arbeit soll einen Überblick über klassische und moderne Skalierungsverfahren sowie deren ihre Vor- und Nachteile schaffen. Diese werden anhand von Beispielen inszeniert. Zuletzt wird basierend auf der Evaluierung der verschiedenen Verfahren eine Empfehlung für die beste Skalierungsmethode für verschiedene Bildtypen gegeben.

In dieser Studienarbeit wird das zentrale Anliegen verfolgt mittels einer akribischen Untersuchung das optimale Gleichgewicht zwischen Komplexität, Rechenaufwand und Resultaten zu ermitteln. Darüber hinaus erfolgt eine umfassende Erläuterung der Bewertungskriterien, die bei der Evaluierung solcher Verfahren Anwendung finden. Das Forschungsvorhaben fokussiert sich auf die systematische Analyse unterschiedlicher Verfahren zur Skalierung von Bildern, um eine fundierte Empfehlung hinsichtlich der optimalen Methode abzugeben. Die Berücksichtigung von Aspekten wie Algorithmuskomplexität<sup>4</sup>, Rechenressourcen und erzielten Ergebnissen ist dabei von essenzieller Bedeutung. Ferner werden präzise Kriterien erörtert, die zur objektiven Bewertung und Vergleichbarkeit der verschiedenen Skalierungsmethoden herangezogen werden können. Diese Studienarbeit

---

<sup>4</sup>Ian CHIVERS u. a. [2015]. »An introduction to Algorithms and the Big O Notation«. In: *Introduction to Programming with Fortran: With Coverage of Fortran 90, 95, 2003, 2008 and 77*, S. 359–364.

strebt an, durch eine methodische Herangehensweise das Potenzial verschiedener Bildskalierungsmethoden zu evaluieren, um eine optimale Lösung zu finden, die ein ausgewogenes Verhältnis zwischen algorithmischer Komplexität, Ressourcenverbrauch und qualitativen Resultaten bietet. Zusätzlich wird eine umfassende Beschreibung der Kriterien angestrebt, welche zur Beurteilung und Vergleichbarkeit dieser Methoden genutzt werden können. Hierzu werden zuerst die wichtigsten Konzepte der digitalen Bildverarbeitung und der Skalierung von Bildern erklärt und einige Beispiele für ihre Anwendung aufgezeigt. Danach werden die traditionellen Skalierungsmethoden vorgestellt und ihre Stärken sowie Schwächen verglichen. Anschließend evaluiert diese Arbeit die neueren Skalierungsmethoden und vergleicht ihre Stärken sowie Schwächen. Abschließend bewertet diese Studienarbeit die verschiedenen Methoden mit verschiedenen Maßstäben für die Bildqualität und gibt eine Empfehlung für die Auswahl einer passenden Methode. Die Arbeit fasst sämtliche in ihr erarbeiteten Ergebnisse zusammen und bespricht deren Bedeutung bezüglich Möglichkeiten und Einschränkungen.

In dieser Studienarbeit wird anschließend das Konzept der Convolutional Neural Networks (CNN) eingeführt, welches im Kontext von Machine Learning / Deep Learning zur Bildanalyse und Bildverarbeitung eingesetzt wird. Das Ziel besteht darin, das Potenzial verschiedener CNN-Architekturen und CNN-Verfahren zu evaluieren und eine optimale Lösung zu finden, die ein ausgewogenes Verhältnis zwischen algorithmischer Komplexität, Rechenaufwand und qualitativen Ergebnissen bietet. Es werden dabei Bewertungskriterien erläutert, die zur objektiven Beurteilung und Vergleichbarkeit der verschiedenen CNN-Methoden herangezogen werden können. Durch eine methodische Herangehensweise sollen die Stärken und Schwächen der CNN-Verfahren untersucht werden, um eine fundierte Empfehlung für ihre Anwendung zu geben.

# Kapitel 2

## Grundlagen der Bildverarbeitung und der Skalierung von Bildern

### 2.1 Einblick in die Bildverarbeitung

Die Bildverarbeitung hat eine faszinierende Geschichte der Entwicklung und Innovation durchlaufen. In den letzten Jahrzehnten hat sie sich zu einem zentralen Bereich der Künstlichen Intelligenz entwickelt. In diesem Abschnitt werfen wir einen Blick auf die historische Entwicklung und den aktuellen Stand der Bildverarbeitung sowie mögliche zukünftige Entwicklungen.

Die Anfänge der Bildverarbeitung reichen zurück bis in die 1960er Jahre, als die theoretischen Grundlagen gelegt wurden<sup>1</sup>. In den 1980er Jahren erfolgte ein weiterer Boom mit dem Aufkommen von Expertensystemen<sup>2</sup>. Doch erst in den letzten Jahren, insbesondere seit 2010, hat die Bildverarbeitung einen revolutionären Fortschritt erlebt, der auf den Durchbrüchen im Bereich des maschinellen Lernens beruht<sup>3</sup>.

Die Geschichte der Künstlichen Intelligenz und der Bildverarbeitung ist eng miteinander

---

<sup>1</sup> Allen NEWELL [1963]. *Learning, Generality and Problem-Solving*. Rand Corporation.

<sup>2</sup> Avron BARR, Edward A FEIGENBAUM und Paul R COHEN [1981]. *The handbook of artificial intelligence*. Bd. 1. William Kaufmann.

<sup>3</sup> Dhanesh RAMACHANDRAM und Graham W TAYLOR [2017]. »Deep multimodal learning: A survey on recent advances and trends«. In: *IEEE signal processing magazine* 34.6, S. 96–108.

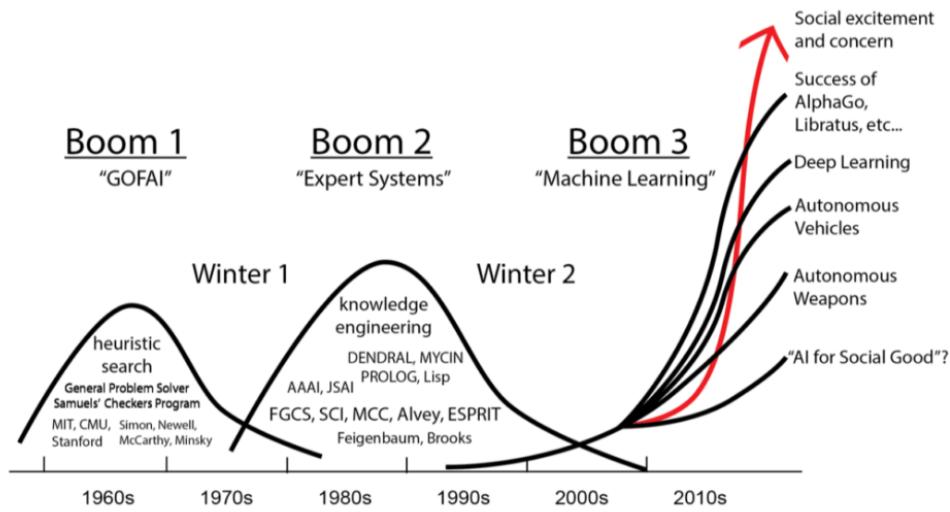


Abbildung 2.1: Die 3 Booms in der Geschichte von KI. Vorlesung DHBW Karlsruhe, TINF20, 2023 – D. Münch

verknüpft. Bereits in den 1940er und 1950er Jahren wurden erste Modelle menschlicher Neuronen entwickelt, um die Funktionsweise des Gehirns besser zu verstehen<sup>4</sup>. In den 1950er Jahren wurde die Künstliche Intelligenz als akademisches Fachgebiet etabliert, und es wurden erste Programme wie der "General Problem Solver" entwickelt.

In den 1960er Jahren wurde mit dem ADaptive LInear NEuron<sup>5</sup> (ADALINE) ein adaptives Mustererkennungssystem entwickelt, das als Meilenstein in der Bildverarbeitung gilt. In den folgenden Jahrzehnten wurden zahlreiche mathematische Theorien und Methoden wie Hidden-Markov-Modelle<sup>6</sup> (HMMs) und Data-Mining entwickelt, um die Bildverarbeitung weiter voranzutreiben.

Allerdings gab es auch Herausforderungen zu bewältigen. In der Vergangenheit waren

<sup>4</sup>Warren S McCULLOCH und Walter PITTS [1943]. »A logical calculus of the ideas immanent in nervous activity«. In: *The bulletin of mathematical biophysics* 5, S. 115–133.

<sup>5</sup>Donald F SPECHT [1990]. »Probabilistic neural networks and the polynomial adaline as complementary techniques for classification«. In: *IEEE Transactions on Neural Networks* 1.1, S. 111–121.

<sup>6</sup>Sean R EDDY [2004]. »What is a hidden Markov model?« In: *Nature biotechnology* 22.10, S. 1315–1316.

die Trainingsdaten begrenzt und die Sensorik nicht ausgereift genug<sup>7</sup>. Zudem waren die Rechenleistungen der Computer nicht ausreichend, um komplexe Aufgaben in Echtzeit zu bewältigen. Die Kombinatorik und Komplexität der Problemstellungen, wie beispielsweise die Gesichtserkennung, stellten weitere Schwierigkeiten dar.

In den letzten Jahren kam es zu einem regelrechten Revival der Künstlichen Intelligenz und der Bildverarbeitung<sup>8</sup>. Durch den Einsatz von Deep Learning und insbesondere von Deep Convolutional Neural Networks wie dem bahnbrechenden AlexNet im Jahr 2012 wurden große Fortschritte erzielt. So konnte beispielsweise im Jahr 2016 AlphaGo den Weltmeister im Spiel Go besiegen<sup>9</sup>. Allerdings brachten diese Entwicklungen auch neue Herausforderungen mit sich, wie beispielsweise die Entstehung von DeepFakes im Jahr 2017<sup>10</sup>.

Der aktuelle Stand der Bildverarbeitung ist geprägt von einer beeindruckenden Leistungsfähigkeit und einer breiten Anwendungspalette. Doch die Möglichkeiten sind noch lange nicht ausgeschöpft. Zukünftige Entwicklungen könnten die Bildverarbeitung in Bereichen wie der Medizin, der Sicherheitstechnik und der Automatisierung weiter vorantreiben und neue Perspektiven eröffnen.

## 2.2 Skalierung von Bildern

### 2.2.1 Arten der Skalierungen: Interpolation und Skalierung

Die Interpolation und die Skalierung von Bildern oder Bildbereichen sind wichtige Konzepte der Bildverarbeitung. Skalierung ist eine Anpassung der Bildgröße, wobei Interpolation genutzt wird, um neue Pixelwerte auf Basis vorgegebener Werte zu berechnen.

Die Wahl der Interpolationsmethode hat einen großen Einfluss auf die Qualität des interpolierten Bildes. In der Bildverarbeitung gibt es verschiedene Interpolationsmetho-

---

<sup>7</sup>Ray KURZWEIL u. a. [1990]. *The age of intelligent machines*. Bd. 580. MIT press Cambridge.

<sup>8</sup>Md Zahangir ALOM u. a. [2018]. »The history began from alexnet: A comprehensive survey on deep learning approaches«. In: *arXiv preprint arXiv:1803.01164*.

<sup>9</sup>Demis HASSABIS [2017]. *Artificial intelligence: chess match of the century*.

<sup>10</sup>Jan KIETZMANN u. a. [2020]. »Deepfakes: Trick or treat?« In: *Business Horizons* 63.2, S. 135–146.

den, wie z.B. Nearest-Neighbor-Interpolation, Bilineare Interpolation oder Bikubische Interpolation.

## 2.2.2 Bildformate

Herkömmliche Bildformate sind rasterbasiert und repräsentieren Bilder als ein Raster von einzelnen Punkten oder Pixeln. Sie ermöglichen eine detaillierte Darstellung, sind jedoch möglicherweise anfällig für Qualitätsverluste bei der Skalierung. Auf der anderen Seite verwenden andere Formate, wie Vektorgrafiken, mathematische Beschreibungen von Linien und Formen, was zu einer verlustfreien Skalierung führen kann und eine hohe Flexibilität bei der Bearbeitung bietet, sind jedoch in der Regel weniger detailreich. In diesem Teil unserer Arbeit werden wir uns hauptsächlich auf die Verarbeitung von Raster-basierten Bildern umfassen.

Historisch gesehen gab es jedoch viele unterschiedliche Formate für Bilder. Zunächst erschufen unterschiedliche Softwareentwickler im Bereich der Bildverarbeitung häufig ihre eigenen Formate.<sup>11</sup> Einheitliche Standards, wie sie heute im Einsatz sind, etablierten sich erst später. Ein Vorreiter der modernen Bildformate ist das "Portable Network Graphics Format"<sup>12</sup>, das 1985 in den USA vorgestellt wurde. Moderne Dateiformate zur Speicherung von Bildern werden anhand der Art des Bildes sowie der Kriterien Speicherbedarf und Kompression, Kompatibilität und ihrem Anwendungsbereich bewertet.<sup>13</sup>

### Portable Network Graphics Format

Portable Network Graphics (PNG) Format setzt einen besonderen Fokus auf eine geringe Komplexität und eine einfache Implementierung des Standards. Der Standard kann frei von jedem genutzt werden. Des weiteren profitiert das Format von verlustfreier Kompres-

---

<sup>11</sup>Wilhelm BURGER und Mark James BURGE [2009]. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag.

<sup>12</sup>Thomas BOUTELL u. a. [o. D.] *PNG (Portable Network Graphics Format)* Version 1.0. Techn. Ber. RFC 2083.

<sup>13</sup>Wilhelm BURGER und Mark James BURGE [2009]. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag.

sion.<sup>14</sup> PNG unterstützt Vollfarbbilder, Grauwertbilder, Transparentz sowie Indexbilder.<sup>15</sup> Der PNG-Algorithmus komprimiert Bilder, indem er mehrere Techniken, einschließlich Filterung und Huffman-Codierung anwendet. Zunächst wird das Bild in Blöcke von 16 x 16 Pixeln aufgeteilt und dann wird auf jedem Block ein Filter angewendet, um Redundanzen zu entfernen. Anschließend wird das Ergebnis der Filterung Huffman-codiert, um eine effiziente Darstellung der Daten zu erreichen. Characteristisch für PNG-Dateien ist auch die Möglichkeit, transparente Flächen einzubauen. Der Standard verwendet eine spezielle Methode, um Transparenz darzustellen. Diese wird als Alpha-Kanal bezeichnet und ermöglicht es, transparente sowie halbtransparente Bilder zu erstellen. Die kompakte Komprimierung des PNG-Formats hat dafür gesorgt, dass der Standard im Internet eine hohe Beliebtheit genießt.<sup>16</sup>

### **JPG-Format**

Das Joint Photographic Experts Group (JPG)-Format, auch bekannt als Joint Photographic Experts Group (JPEG), ist eines der am weitesten verbreiteten und bekanntesten Bildformate. Es wurde von der gleichnamigen Expertengruppe entwickelt und 1992 veröffentlicht.

JPEG basiert auf einer verlustbehafteten Kompressionsmethode, bei der Bildinformationen reduziert werden, um die Dateigröße zu verringern. Dies ermöglicht eine effiziente Speicherung und Übertragung von Bildern. Die Kompressionsrate kann je nach gewünschter Qualität und Dateigröße angepasst werden.

Das JPEG-Format eignet sich besonders gut für fotografische Bilder und komplexe Szenen mit vielen Farbverläufen und Details. Es unterstützt eine hohe Farbtiefe und kann Millionen von Farben darstellen. Aufgrund der verlustbehafteten Kompression können jedoch bei hohen Kompressionsraten Artefakte und Qualitätsverluste sichtbar werden.

---

<sup>14</sup>Thomas BOUTELL [1997]. *Png (portable network graphics) specification version 1.0*. Techn. Ber.

<sup>15</sup>Wilhelm BURGER u. a. [2015]. »Digitale Bilder«. In: *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*, S. 1–24.

<sup>16</sup>W3C [2003]. *Portable Network Graphics (PNG) Specification (Second Edition)*. <https://www.w3.org/TR/png/>. Accessed: March 9, 2023.

Das JPG-Format ist weit verbreitet und wird von den meisten Bildbetrachtungs- und Bildbearbeitungsprogrammen unterstützt. Es ist auch das bevorzugte Format für Bilder im Internet, da es eine gute Balance zwischen Qualität und Dateigröße bietet. Es ist jedoch wichtig, die Kompressionsrate sorgfältig zu wählen, um ein optimales Ergebnis zu erzielen und Qualitätsverluste zu minimieren.

Insgesamt ist das JPG-Format eine beliebte Wahl für die Speicherung und Weitergabe von fotografischen Bildern, insbesondere wenn es auf eine effiziente Dateigröße und eine breite Kompatibilität ankommt.

## Scalable Vector Graphics

The main idea motivating Scalable Vector Graphics (SVG) was simple: to create a generic document-oriented solution for graphics that can be adapted to modern media [FIBINGER 2002]

Scalable Vector Graphics (SVG) steht für ein Format, das Vektorgrafiken basierend auf eXtensible Markup Language (XML) darstellt. Im Gegensatz zu Rastergrafiken, wie z.B. PNG, die aus Pixeln bestehen und bei Vergrößerung an Schärfe verlieren, sind Vektorgrafiken vektorbasiert und behalten ihre Qualität bei beliebiger Skalierung. Der Standard ermöglicht eine besonders effiziente Speicherung von Bildern. SVG ist außerdem ein offenes Format und unterstützt Interaktivität, Animation und Skripting.<sup>17</sup> Da SVG auf XML basiert, kann es auch mit anderen Webtechnologien wie HTML, Cascading Style Sheets (CSS) und JavaScript integriert werden.<sup>18</sup>

## Weitere Standards

Im Bereich der Grafikformate gibt es eine Vielzahl von Standards, die verschiedene Anforderungen und Einsatzzwecke erfüllen. Neben den bereits erwähnten Formaten gibt es noch weitere wichtige Standards, die im Folgenden näher erläutert werden:

<sup>17</sup>Antoine QUINT [2003]. »Scalable vector graphics«. In: *IEEE MultiMedia* 10.3, S. 99–102.

<sup>18</sup>MOZILLA CONTRIBUTORS [n.d.] *SVG (Scalable Vector Graphics)*. <https://developer.mozilla.org/en-US/docs/Web/SVG>. Accessed: March 9, 2023.

### **Graphics Interchange Format (GIF)**

Das GIF-Format steht für Graphics Interchange Format und ist ein Format für animierte Rastergrafiken. Es zeichnet sich durch eine begrenzte Farbpalette von 256 Farben aus. GIF verwendet eine verlustfreie Kompression, die jedoch nicht besonders effizient ist. Es eignet sich gut für einfache Animationen und Grafiken mit wenigen Farben. Aufgrund dieser Eigenschaft, und der frühen Adoption ist GIF besonders für Webanwendungen und den Austausch von einfachen Grafiken im Internet beliebt.

### **Tagged Image File Format (TIFF)**

Das TIFF-Format, kurz für Tagged Image File Format, ist ein Format für hochauflösende Rastergrafiken. Es bietet verschiedene Optionen für Farbmanagement und Metadaten und wird häufig im Druckbereich verwendet. Im Gegensatz zu JPEG oder GIF unterstützt TIFF sowohl verlustfreie als auch unkomprimierte Kompressionsmöglichkeiten. Dadurch bleiben die Bildqualität und die Details erhalten. Allerdings ist TIFF aufgrund seiner Größe und der begrenzten Browserkompatibilität weniger geeignet für die Verwendung im Web.

### **Photoshop Document (PSD)**

Das PSD-Format steht für Photoshop-Dokument und wird von der Software Adobe Photoshop verwendet. Es ist ein proprietäres Format, das alle Ebenen, Masken, Effekte und andere Informationen einer Rastergrafik speichert. PSD ermöglicht umfangreiche Bearbeitungsmöglichkeiten, ist jedoch nur mit Photoshop kompatibel. Das Format wird häufig von Grafikdesignern und professionellen Bildbearbeitern verwendet, um hochwertige Rastergrafiken zu erstellen und zu bearbeiten, als auch Vektorgrafiken.

### **Bitmap (BMP)**

Das BMP-Format, kurz für Bitmap, ist ein unkomprimiertes Rastergrafikformat. Es speichert Bilddaten pixelgenau und unterstützt keine Komprimierungstechniken. Dadurch entstehen sehr große Dateien, weshalb BMP heutzutage eher selten verwendet wird. Zudem bietet das Format keine Transparenz- oder andere erweiterte

Funktionen. Es war jedoch eines der ersten weit verbreiteten Formate und wird in einigen spezifischen Anwendungsbereichen noch genutzt.

### Encapsulated PostScript (EPS)

Das EPS-Format, kurz für Encapsulated PostScript, ist ein Format für vektorbasierte Grafiken. Es speichert Kurven, Texte und andere Elemente anhand mathematischer Beschreibungen. EPS ermöglicht eine verlustfreie Skalierung der Grafiken ohne Qualitätsverlust und wird häufig im Druckbereich verwendet. Allerdings ist EPS aufgrund seiner begrenzten Browser- und Programmkompatibilität weniger geeignet für die Verwendung im Web oder mit anderen Softwareanwendungen.

Neben den hier genannten Standards gibt es noch viele weitere Grafikformate, die je nach spezifischen Anforderungen und Einsatzszenarien verwendet werden. Da die Vielfalt der Grafikformate den Umfang dieser Arbeit übersteigt, wurden hier nur einige der bedeutendsten und bekanntesten Formate vorgestellt. Jedes Format hat seine eigenen Vorteile und Nachteile, abhängig von Faktoren wie Bildqualität, Dateigröße, Kompatibilität und Verwendungszweck.<sup>19,20,21,22,23,24</sup>

---

<sup>19</sup>PREPRESSURE [n.d.] *Prepressure Library: File Formats*. <https://www.prepressure.com/library/file-formats/>. Accessed: March 9, 2023.

<sup>20</sup>IONOS [n.d.] *Graphic File Formats: Which Formats Are Important?* <https://www.ionos.com/digitalguide/websites/web-design/graphic-file-formats-which-formats-are-important/>. Accessed: March 9, 2023.

<sup>21</sup>Majid RABBANI und Rajan JOSHI [2002]. »An overview of the JPEG 2000 still image compression standard«. In: *Signal processing: Image communication* 17.1, S. 3–48.

<sup>22</sup>Michael W MARCELLIN u. a. [2000]. »An overview of JPEG-2000«. In: *Proceedings DCC 2000. Data Compression Conference*. IEEE, S. 523–541.

<sup>23</sup>ENCYCLOPEDIA BRITANNICA [n.d.] *JPEG*. <https://www.britannica.com/technology/JPEG>. Accessed: March 9, 2023.

<sup>24</sup>ELSEVIER [n.d.] *Artwork and media instructions*. <https://www.elsevier.com/authors/policies-and-guidelines/artwork-and-media-instructions/artwork-overview>. Accessed: March 9, 2023.

### 2.2.3 Wichtige Aspekte der Skalierung

#### Segmentierung

Die Segmentierung von Bildern ist ein wichtiges Verfahren der Bildverarbeitung, da es ermöglicht, ein Bild in sinnvolle Regionen zu unterteilen. Hierbei können verschiedene Verfahren, wie Schwellenwert- oder Clustering-Methoden eingesetzt werden. Die Genauigkeit der Segmentierung hängt dabei maßgeblich von der Komplexität des Bildes und der gewählten Methode ab. Eine erfolgreiche Segmentierung kann für viele Anwendungen von Nutzen sein. Die automatischen Erkennung von Gesichtern oder die Identifizierung von Verkehrszeichen auf Straßenbildern sind die populärsten Beispiele. Jedoch ist es oft schwer genaue Grenzen zwischen Objekten in Bildern mit komplexen Strukturen zu erkennen.

#### Klassifizierung

Die Klassifizierung von Bildinhalten ist ein weiterer wichtiger Aspekt der Bildverarbeitung. Hierbei werden Bilder in automatisch bestimmte Kategorien eingeteilt. Beispielanwendungen inkludieren das Erkennen von Tierarten auf Naturfotos oder das Identifizieren von Gesichtern auf Fotos. Hierfür können verschiedene Techniken verwendet werden. Beispiele sind Deep Learning oder Entscheidungsbaum-Algorithmen. Eine erfolgreiche Klassifizierung kann für viele Anwendungen genutzt werden, wie zum Beispiel zur Automatisierung von Aufgaben oder im maschinellen Lernen. Die Genauigkeit einer Klassifizierung wird von Faktoren, wie zum Beispiel von der Qualität der Trainingsdaten oder der Komplexität der verwendeten Klassifikationsmethode beeinflusst.

#### Objekterkennung und -verfolgung

Die Objekterkennung und -verfolgung ist ein wichtiger Aspekt der Bildverarbeitung, der oft in Anwendungen wie der Überwachung und Robotik genutzt wird. Hier geht es darum, Objekte in Bildern oder Videos zu erkennen und ihre Bewegungen zu verfolgen. Es können verschiedene Techniken eingesetzt werden, wie zum Beispiel Hintergrundsubtraktion oder optische Flussberechnung. Eine erfolgreiche Objekterkennung und -verfolgung kann für

viele Anwendungen genutzt werden, wie zum Beispiel in der Videoüberwachung oder bei der Steuerung von autonomen Fahrzeugen. Allerdings gibt es auch Herausforderungen bei der Objekterkennung und -verfolgung, wie zum Beispiel die Bewältigung von Hintergrundrauschen oder die Verfolgung von Objekten bei hoher Geschwindigkeit.

### 3D-Bildverarbeitung

In der 3D-Bildverarbeitung werden dreidimensionale Bilder und Modelle analysiert und verarbeitet. Die Anwendungsfelder dieser Technologien reichen von medizinischen Umgebungen bis hin zur industriellen Fertigung. In bildgebenden medizinischen Verfahren werden 3D-Bilder für die Diagnose von Krankheiten und Verletzungen verwendet. In der industriellen Fertigung werden 3D-Modelle für die Qualitätssicherung und die Fehlererkennung verwendet.

### Bildkompression

Da Bilder in der Regel große Datenmengen erzeugen, die für die Übertragung und Speicherung unpraktisch sind, benötigt es oft eine Bildkompression. Durch Kompressionstechniken wie z.B. die JPEG-Kompression kann die Größe von Bildern erheblich reduziert werden ohne dabei große Qualitätsverluste zu erleiden.

## 2.3 Echtzeitverarbeitung von Bildern

Die Echtzeitverarbeitung von Bildern stellt eine Herausforderung in der Bildverarbeitung dar, da sie eine sehr schnelle Verarbeitung erfordert, um zeitkritische Anwendungen wie autonomes Fahren oder Augmented Reality in Echtzeit zu realisieren.<sup>25,26</sup> Bei diesen Aufgaben müssen Bilder in Echtzeit erfasst, verarbeitet und angezeigt werden, um eine reibungslose Funktionalität zu gewährleisten.

---

<sup>25</sup>Y. ZHANG und Q. LIU [2018]. »Scalable Image Processing Techniques for Real-Time Applications«. In: *IEEE Transactions on Image Processing*.

<sup>26</sup>C. LI und Z. WANG [2020]. »Convolutional Neural Networks for Image Scaling in Real-Time Applications«. In: *Journal of Visual Communication and Image Representation*.

Unter Echtzeitverarbeitung versteht man die Fähigkeit, Daten sofort und unmittelbar zu verarbeiten, ohne wahrnehmbare Verzögerungen. In Bezug auf die Bildverarbeitung bedeutet dies, dass die Verarbeitung innerhalb eines bestimmten Zeitrahmens erfolgen muss, um den Anforderungen der Anwendung gerecht zu werden. In der Regel ist dies eine Echtzeitverarbeitung in Echtzeit oder nahezu Echtzeit.

Die Echtzeitverarbeitung von Bildern erfordert in der Regel eine hohe Rechenleistung, um die Daten schnell genug zu verarbeiten. Hierbei kommen spezielle Algorithmen und Techniken zum Einsatz, die für eine effiziente Verarbeitung sorgen. Beispielsweise werden häufig spezialisierte Hardware wie Grafikprozessoren (GPUs) oder FPGA (Field Programmable Gate Array)-basierte Systeme verwendet, um die Verarbeitungsgeschwindigkeit zu erhöhen.

Eine weitere Herausforderung bei der Echtzeitverarbeitung von Bildern ist die Echtzeit-Kommunikation zwischen den verschiedenen Komponenten des Systems. Hierbei müssen Daten schnell und zuverlässig ausgetauscht werden, um Verzögerungen zu vermeiden. Hierfür kommen häufig spezielle Systeme zum Einsatz, die für eine schnelle Übertragung und parallele Verarbeitung von Daten optimiert sind. Eine effiziente Datenübertragung und Kommunikation zwischen den einzelnen Komponenten des Systems ist entscheidend, um eine Echtzeitverarbeitung zu ermöglichen.<sup>27</sup>

Insgesamt erfordert die Echtzeitverarbeitung von Bildern eine Kombination aus leistungsstarker Hardware, effizienten Algorithmen und einer effektiven Datenkommunikation, um die Anforderungen zeitkritischer Anwendungen zu erfüllen. Durch den Fortschritt in der Hardware- und Softwaretechnologie wird die Echtzeitverarbeitung von Bildern immer besser und ermöglicht somit eine Vielzahl von spannenden Anwendungen.

---

<sup>27</sup>Carl-Werner OEHLRICH u. a. [1992]. »Ein Transputersystem mit verteiltem Bildspeicher für Echtzeit-Computergrafik und Bildverarbeitung«. In: *Parallele Datenverarbeitung mit dem Transputer: 3. Transputer-Anwender-Treffen TAT'91, Aachen, 17.–18. September 1991*. Springer, S. 170–177.

## 2.4 Anwendungen von Skalierungsmethoden

Die Anwendung von Skalierungsmethoden bietet eine Vielzahl ungewöhnlicher, aber äußerst nützlicher Anwendungen in verschiedenen Sektoren. In der Landwirtschaft können Skalierungsmethoden, wie beispielsweise das Herunterskalieren von Pflanzenbildern und ihre Analyse mithilfe von Convolutional Neural Networks (CNN), dazu beitragen, die Gesundheit von Pflanzen zu überwachen und den Einsatz von Pestiziden zu optimieren<sup>28</sup>. In der Archäologie<sup>29</sup> ermöglichen Skalierungsmethoden die digitale Rekonstruktion verwitterter oder beschädigter Artefakte. Dies eröffnet Forschern neue Einblicke in die Geschichte vergangener Zivilisationen und ermöglicht die Erstellung interaktiver virtueller Museen. In der Modeindustrie werden Skalierungsmethoden eingesetzt, um eine möglichst präzise und detailgetreue Skalierung von Kleiderbildern zu ermöglichen. Dadurch können Kunden Kleidungsstücke virtuell anprobieren, um im Voraus die Passform und das Aussehen zu überprüfen. Dies kann dazu beitragen, das Online-Shopping-Erlebnis für Kunden zu verbessern<sup>30</sup>. Insgesamt bieten Skalierungsmethoden in vielen verschiedenen Sektoren ungewöhnliche Anwendungen, die zu Effizienzsteigerungen, Erkenntnisgewinnen und verbesserten Kundenerfahrungen führen.

---

<sup>28</sup>Alina GRANWEHR und Verena HOFER [2021]. »Analysis on digital image processing for plant health monitoring«. In: *Journal of Computing and Natural Science* 1, S. 1.

<sup>29</sup>Jeroen DE REU u. a. [2014]. »On introducing an image-based 3D reconstruction method in archaeological excavation practice«. In: *Journal of Archaeological Science* 41, S. 251–262.

<sup>30</sup>Frederic CORDIER, Hyewon SEO und Nadia MAGNENAT-THALMANN [2003]. »Made-to-measure technologies for an online clothing store«. In: *IEEE Computer graphics and applications* 23.1, S. 38–48.

# Kapitel 3

## Klassische Skalierungsmethoden

### 3.1 Pixel-Verdopplung

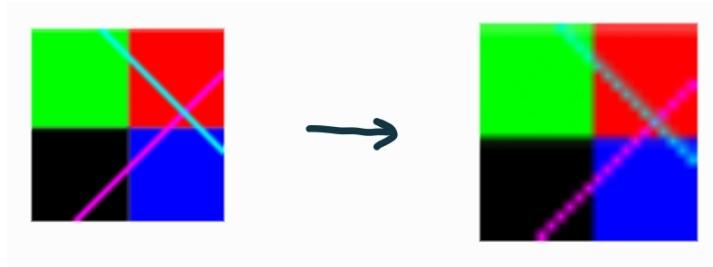


Abbildung 3.1: Beispielhafte Darstellung einer Skalierung durch Pixelverdopplung

Die Pixel-Verdopplung vergrößert das Bild, indem jeder Pixel mit ganzen Zahlen dupliziert wird. Diese Methode kann schnell und einfach umgesetzt werden, indem jeder Pixelwert einfach auf den Nachbarpixel übertragen wird. Wenn Bilder mit dieser Methode stark vergrößert werden, ergeben sich oft pixelige und unscharfe Ausgaben, da die Details nicht wirklich vorhanden sind, sondern nur durch die Duplizierung von Pixeln aufgefüllt werden. Aus diesem Grund wird Pixel-Verdopplung oft als eine minderwertige Skalierungsmethode betrachtet und findet in professionellen Anwendungen selten Gebrauch<sup>1</sup>.

<sup>1</sup> »Digital image enhancement: A survey« [1983]. In: *Computer Vision, Graphics, and Image Processing* 24.3, S. 363–381. ISSN: 0734-189X. DOI: [https://doi.org/10.1016/0734-189X\(83\)90061-0](https://doi.org/10.1016/0734-189X(83)90061-0).

Eine beispielhafte Implementierung in Python sieht folgendermaßen aus:

Algorithmus 3.1: Python-Klasse zur Pixelverdopplung und Bildmanipulation: Implementierung des Algorithmus mit der Pillow-Bibliothek und Erklärung des Codes von PixelVerdopplung: [https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/pixel\\_verdopplung.py](https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/pixel_verdopplung.py).

```
class PixelVerdopplung(Image):
    def __init__(self, path):
        extend = "p2"
        super().__init__(path, extend)

    def manipulate(self, new_size):
        super().manipulate(new_size)

        for y in range(self.new_height):
            for x in range(self.new_width):
                x_old = int(x / (self.new_width / self.width))
                y_old = int(y / (self.new_height / self.height))

                # Check that x_old and y_old are within bounds of original image
                if x_old >= self.width:
                    x_old = self.width - 1
                if y_old >= self.height:
                    y_old = self.height - 1

                old_pixel = self.img.getpixel((x_old, y_old))
                self.newImg.putpixel((x, y), old_pixel)

    return self.save()
```

Die vorliegende Implementierung in Python beschreibt die Realisierung der Pixelver-

dopplungsklasse, welche in der Lage ist, ein größeres Bild zu erzeugen, indem die leeren Pixel mit demselben Pixelwert wie der nächste Nachbarpixel gefüllt werden. Der Code ist darauf ausgelegt, eine einfache Möglichkeit bereitzustellen, um ein Bild auf eine höhere Auflösung zu skalieren, wodurch fehlende Details ausgeglichen werden können. Dabei wird eine lineare Interpolation auf der Basis der Nachbarpixel durchgeführt, um das neue Bild zu generieren. Die Klasse “PixelVerdopplung” erbt von der Klasse “Image” und besitzt einen Konstruktor, der den Pfad zum Bild und die Erweiterung “p2” als Argumente erhält. Die Methode “manipulate” ist dafür zuständig, das Bild auf die gewünschte Größe zu skalieren und zu manipulieren. Innerhalb der Methode werden Schleifen durchlaufen, um jeden neuen Pixel im manipulierten Bild zu generieren. Die Koordinaten des jeweiligen alten Pixels werden durch Division der neuen Koordinaten durch die Skalierungsfaktoren berechnet und auf den nächstgelegenen Integer gerundet. Um sicherzustellen, dass die berechneten Koordinaten innerhalb der Grenzen des ursprünglichen Bildes liegen, werden sie in einem nächsten Schritt auf den maximalen Index des Bildes zurückgesetzt, falls sie außerhalb liegen sollten. Anschließend wird der Pixelwert des entsprechenden alten Pixels abgerufen und als neuer Pixel an der berechneten Stelle im neuen Bild platziert. Die Implementierung dieses Algorithmus stellt eine einfache Möglichkeit dar, um ein Bild auf eine höhere Auflösung zu skalieren, wodurch ein besseres visuelles Ergebnis erzielt werden kann. Dabei ist darauf zu achten, dass die lineare Interpolation eine höhere Laufzeit und Speicheranforderungen aufweist als andere Interpolationsmethoden.

## 3.2 Nearest-Neighbor-Interpolation

Die Nearest Neighbor Interpolation (NNI) ist eine weitere Methode zur Skalierung von Bildern. Es wird für jedes Pixel im Ausgabebild der am nächsten liegende Pixel im Eingabebild ausgewählt und der Farbwert des ausgewählten Pixels wird als Farbwert des entsprechenden Pixels im Ausgabebild verwendet. Die Verwendung von NNI ist einfach und schnell zu implementieren. Aufgrund ihrer geringen Komplexität ist sie daher sehr beliebt. Die Methode eignet sich besonders gut für die Vergrößerung von Bildern mit

großen, einheitlichen Bereichen oder harten Kanten.

```
def nearest_neighbor_interpolation(image, scale_factor):
    new_size = (int(image.shape[1] * scale_factor), int(image.shape[2] * scale_factor))

    scaled_image = np.zeros(new_size + (image.shape[2],), dtype=np.uint8)

    for i in range(new_size[0]):
        for j in range(new_size[1]):
            x = int(i / scale_factor)
            y = int(j / scale_factor)
            scaled_image[j, i] = image[y, x]

    return scaled_image
```

<sup>2</sup>

Bei der Verkleinerung von Bildern erleiden diese jedoch oft einen Qualitätsverlust. Hier kommt es zu Unschärfe und Blockbildung. Dieser Effekt verstärkt sich, wenn das Verhältniss zwischen Quellbild und Ausgabebild kein Vielfaches ist.

### 3.3 Bilineare Interpolation

Die bilineare Interpolation ist eine weit verbreitete Methode zur Skalierung von Bildern. Sie basiert auf dem Konzept der linearen Interpolation und wird häufig in Grafikanwendungen und Bildverarbeitungsalgorithmen eingesetzt. Bei der bilinearen Interpolation werden zwei Pixelwerte linear interpoliert, um den Wert eines Zwischenpunkts zu berechnen. Dieser Ansatz führt zu einer relativ schnellen Berechnung und liefert ästhetisch ansprechende Ergebnisse.

Der Prozess der bilinearen Interpolation bezieht sich auf die Berechnung der Distanz zwischen dem zu interpolierenden Punkt und den umliegenden vier Pixeln. Diese Distanz

---

<sup>2</sup>Nan JIANG und Luo WANG [2015]. »Quantum image scaling using nearest neighbor interpolation«. In: *Quantum Information Processing* 14, S. 1559–1571.

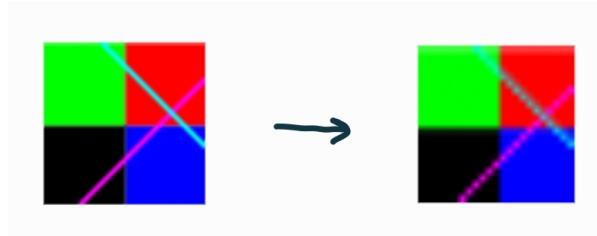


Abbildung 3.2: Bilineare Verdopplung

wird linear verwendet, um gewichtete Durchschnittswerte der umgebenden Pixel zu erzeugen. Dieser Ansatz ermöglicht eine glattere Darstellung von Zwischenwerten im Vergleich zu einfacheren Interpolationsmethoden wie der nächsten Nachbar-Interpolation.

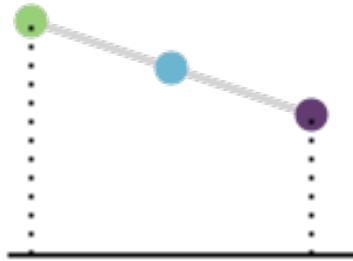


Abbildung 3.3: Graph über bilineare Skalierung.

Die bilineare Interpolation wird aufgrund ihrer Einfachheit und Effizienz häufig eingesetzt. Sie ist schnell zu implementieren und kann auf unterschiedliche Bildgrößen angewendet werden. Viele Bildbearbeitungssoftware und Grafikbibliotheken nutzen bilineare Interpolation als Standardmethode für die Skalierung von Bildern. Dennoch hat die bilineare Interpolation einige Nachteile, insbesondere bei extremen Skalierungen. Bei sehr großen oder kleinen Skalierungsfaktoren können Artefakte auftreten, die zu einer verminderten Bildqualität führen. In solchen Fällen können fortgeschrittenere Interpolationsverfahren wie Bikubische oder Lanczos eine bessere Wahl sein.

Algorithmus 3.2: [https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bilinear\\_interpolation.py](https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bilinear_interpolation.py)

```
for y in range(self.new_height):
    for x in range(self.new_width):
```

```

x_old = x / (self.new_width / self.width)
y_old = y / (self.new_height / self.height)

# Find the surrounding pixels
x1 = int(x_old)
x2 = min(x1 + 1, self.width - 1)
y1 = int(y_old)
y2 = min(y1 + 1, self.height - 1)

# Check if x2 and y2 are out of bounds, if they are subtract one
if x2 == self.width - 1:
    x2 = x1
    x1 -= 1
if y2 == self.height - 1:
    y2 = y1
    y1 -= 1

# Find the weights
w1 = (x2 - x_old) * (y2 - y_old)
w2 = (x_old - x1) * (y2 - y_old)
w3 = (x2 - x_old) * (y_old - y1)
w4 = (x_old - x1) * (y_old - y1)

# Get the pixel values of the surrounding pixels
p1 = self.img.getpixel((x1, y1))
p2 = self.img.getpixel((x2, y1))
p3 = self.img.getpixel((x1, y2))
p4 = self.img.getpixel((x2, y2))

```

```

# Interpolate the pixel value
new_pixel = (
    int(w1 * p1[0] + w2 * p2[0] + w3 * p3[0] + w4 * p4[0]),
    int(w1 * p1[1] + w2 * p2[1] + w3 * p3[1] + w4 * p4[1]),
    int(w1 * p1[2] + w2 * p2[2] + w3 * p3[2] + w4 * p4[2])
)
self.newImg.putpixel((x, y), new_pixel)

```

3

## 3.4 Bikubische Interpolation

### 3.4.1 Mathematische Grundlagen

Die bikubische Interpolation ist ein mathematisches Verfahren zur Schätzung der Werte einer kontinuierlichen Funktion an einer gegebenen Stelle, indem eine Funktion mit kubischen Polynomen verwendet wird, die durch benachbarte Funktionswerte verläuft. Dabei wird das umliegende Gebiet untersucht und die Werte werden basierend auf der Stichprobentheorie geschätzt.

Die bikubische Interpolationsformel ist eine Erweiterung der Bilinearinterpolation auf vier umliegende Pixel und verwendet eine Funktion, die durch benachbarte Funktionswerte verläuft. Die resultierende Funktion ist stetig differenzierbar und besitzt glatte partielle Ableitungen.

Im Vergleich zu anderen Interpolationsverfahren wie der bilinearen Interpolation und der Lanczos-Interpolation hat die bikubische Interpolation den Vorteil, dass sie eine höhere Genauigkeit bei der Schätzung von Pixelwerten bietet. Ein Nachteil ist jedoch, dass sie im Allgemeinen höhere Rechenaufwendungen erfordert.

---

<sup>3</sup>K.T. GRIBBON und D.G. BAILEY [2004]. »A novel approach to real-time bilinear interpolation«. In: *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, S. 126–131. DOI: 10.1109/DELTA.2004.10055.

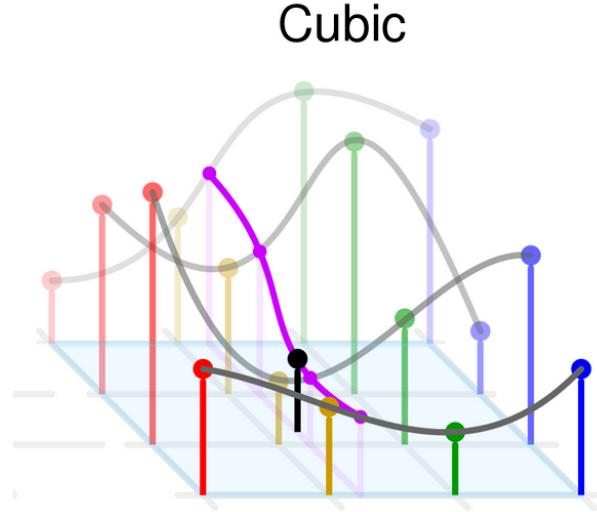


Abbildung 3.4: Bikubische Interpolation

Die Fourier-Analyse und die Fourier-Transformationen spielen eine wichtige Rolle bei der Bildinterpolation, da sie es ermöglichen, die Funktion in den Frequenzraum zu transformieren und damit eine effektivere Interpolation zu erreichen.

### 3.4.2 Algorithmische Implementierung

Die bikubische Interpolation erfolgt durch die Auswahl von umgebenden Pixeln und deren Gewichte sowie der Berechnung der neuen Pixelwerte.

Sie verwendet eine 4x4-Matrix umliegender Pixel, um den Wert an einem bestimmten Punkt zu berechnen. Die Formel zur Berechnung lautet:

Die Koeffizienten  $a_{ij}$  werden basierend auf den umliegenden Pixelwerten und ihren Ableitungen berechnet. Dieses Verfahren ermöglicht eine glatte Interpolation zwischen den Pixeln und hinterlässt wenige scharfe Kanten.

$$f(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{i,j} * x^i * y^j \quad (3.1)$$

Eine Möglichkeit zur Optimierung der Leistung besteht darin, Lookup-Tabellen vorzubereiten, Parallelisierungstechniken zu verwenden und die Speicherverwaltung zu optimieren. Grenzfälle und Randbedingungen können bei der bikubischen Interpolation auftreten und

müssen effektiv behandelt werden. Die algorithmische Implementierung der bikubischen Interpolation kann mit anderen Interpolationsverfahren wie der bilinearen Interpolation und der Lanczos-Interpolation verglichen werden.

Unsere Implementierung der bikubischen Interpolation in Python ist im Abschnitt 3.4.4.

### 3.4.3 Analyse der Leistung

Zur Bewertung der Leistung von Bildinterpolationsverfahren werden verschiedene Kriterien verwendet, einschließlich der visuellen Qualität, der Genauigkeit und der Berechnungseffizienz. Die Leistung der bikubischen Interpolation kann mit anderen Interpolationstechniken anhand von Testbildern und Datensätzen verglichen werden. Dabei werden verschiedene Parameter wie Bildgröße, Auflösung und Inhalt untersucht, um ihre Auswirkungen auf die Leistung der bikubischen Interpolation zu analysieren.

### 3.4.4 Implementation

Algorithmus 3.3: Python-Klasse zur Bikubischen interpolation [https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bicubic\\_interpolation.py](https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bicubic_interpolation.py).

```
from backend.image import Image

def __cubic(x, a=-0.5):
    abs_x = abs(x)
    if abs_x <= 1:
        return (a + 2) * (abs_x ** 3) - (a + 3) * (abs_x ** 2) + 1
    elif abs_x < 2:
        return a * (abs_x ** 3) - (5 * a) * (abs_x ** 2) + (8 * a) *
    abs_x - (4 * a)
    return 0
```

```

def __get_weight( distance ):
    abs_distance = abs( distance )
    if abs_distance <= 1:
        return __cubic( abs_distance )
    return 0

class BicubicInterpolation( Image ):
    def __init__( self , path ):
        extend = "biCu"
        super().__init__( path , extend )

    def manipulate( self , new_size ):
        super().manipulate( new_size )

        for y in range( self.new_height ):
            for x in range( self.new_width ):
                px = x * self.width / self.new_width
                py = y * self.height / self.new_height

                ix = math.floor( px )
                iy = math.floor( py )

                dx = px - ix
                dy = py - iy

                weights = []
                for j in range( -1 , 3 ):
                    for i in range( -1 , 3 ):

```

```

        weight = _get_weight(i - dx) * _get_weight(dy - j)
        weights.append(weight)

    total_weight = sum(weights)
    normalized_weights = [w / total_weight for w in weights]

    new_pixel = [0, 0, 0]
    for j in range(-1, 3):
        for i in range(-1, 3):
            ox = ix + i
            oy = iy + j
            if ox < 0 or oy < 0 or ox >= self.width or
            oy >= self.height:
                pixel_value = list(
                    self.img.getpixel((min(max(ox, 0), self.width - 1),
                                      min(max(oy, 0), self.height)))
                )
            else:
                pixel_value = list(self.img.getpixel((ox, oy)))

            weight_index = (j + 1) * 4 + (i + 1)
            weight = normalized_weights[weight_index]
            for k in range(3):
                new_pixel[k] += weight * pixel_value[k]

    self.newImg.putpixel((x, y), (int(new_pixel[0]),
                                   int(new_pixel[1]),
                                   int(new_pixel[2])))
    return self.save()

```

Die Methode ‘manipulate‘ der Klasse ‘BicubicInterpolation‘ implementiert die bikubische Interpolation für das Vergrößern von Bildern. Der wichtigste Teil des Codes ist

die Schleife, die über jedes Pixel einen Kernel berechnet, der die jeweiligen Gewichte der benachbarten Pixel errechnet. In den genannten Zeilen wird ein 4x4-Kern um das aktuelle Pixel herum gebildet und für jeden Pixel im Kern werden die Gewichte berechnet. Dabei wird die ‘`_get_weight`’-Funktion aufgerufen, welche auf der Basis einer kubischen Kurve das Gewicht für einen bestimmten Abstand berechnet. Die berechneten Gewichte werden in eine Liste ‘weights’ hinzugefügt. Diese Gewichte dienen später bei der Interpolation des aktuellen Pixels als multiplikative Faktoren für die umliegenden Pixel im Originalbild.

### 3.4.5 Anwendungen

Die bikubische Interpolation wird in verschiedenen Anwendungen der Bildverarbeitung eingesetzt und bietet bestimmte Vorteile<sup>4</sup> gegenüber der bilinearen Interpolation. Ein Bereich, in dem die bikubische Interpolation besser geeignet sein kann, ist die Bildskalierung und -größenänderung. Durch die Verwendung von zusätzlichen benachbarten Pixeln in der Interpolationsmethode kann die bikubische Interpolation feinere Details beibehalten und glattere Übergänge erzeugen, was zu hochwertigeren skalierten Bildern führen kann.

Ein weiterer Anwendungsbereich ist die Bildrotation und -transformation<sup>5</sup>. Bei der bikubischen Interpolation werden nicht nur benachbarte Pixel, sondern auch deren benachbarte Pixel berücksichtigt, wodurch eine bessere Anpassung an die gewünschten Rotationen oder Transformationen ermöglicht wird. Dies kann zu weniger Verzerrungen und einem insgesamt realistischeren Ergebnis führen.

Die bikubische Interpolation kann auch in der Bildentrauschung und -wiederherstellung nützlich sein<sup>6</sup>. Durch die Verwendung von zusätzlichen Pixeln und einer gewichteten

---

<sup>4</sup>Donya KHALEDYAN u. a. [2020]. »Low-cost implementation of bilinear and bicubic image interpolation for real-time image super-resolution«. In: *2020 IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE, S. 1–5.

<sup>5</sup>Pankaj S PARSANIA und Paresh V VIRPARIA [2016]. »A comparative analysis of image interpolation algorithms«. In: *International Journal of Advanced Research in Computer and Communication Engineering* 5.1, S. 29–34.

<sup>6</sup>Zhou WANG u. a. [2004a]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE transactions on image processing* 13.4, S. 600–612.

Interpolation können Rauschanteile reduziert und verloren gegangene Informationen wiederhergestellt werden. Dies kann insbesondere in der medizinischen Bildgebung und der Satellitenbildgebung von Vorteil sein, wo genaue und zuverlässige Bilder erforderlich sind.

Wichtig zu beachten ist, dass die Eignung der bikubischen Interpolation von verschiedenen Faktoren abhängt, einschließlich der Natur des Eingangsbildes, des gewünschten Ausgabekontexts und der verfügbaren Rechenleistung. In einigen Fällen kann die bikubische Interpolation aufgrund ihrer erhöhten Berechnungskosten und möglichen Unschärfen weniger geeignet sein<sup>7</sup>. Eine sorgfältige Bewertung der spezifischen Anforderungen und der verfügbaren Optionen ist daher umso entscheidender, um die optimale Interpolationsmethode für eine bestimmte Anwendung zu bestimmen.

### 3.4.6 Erweiterungen und Variationen

Erläuterung der verschiedenen Erweiterungen und Variationen der bikubischen Interpolation, wie z. B. Super-Resolution-Techniken, Multiskalen- und pyramidenbasierte Interpolation und adaptive Interpolationstechniken. Diskussion der Vor- und Nachteile dieser Varianten und ihrer Eignung für verschiedene Bildtypen und Anwendungen. Erkundung potenzieller künftiger Forschungsrichtungen in diesem Bereich, z. B. auf Deep Learning basierende Ansätze, ungleichmäßige und unregelmäßige Abtastverfahren sowie mehrdimensionale und mehrkanalige Interpolation.

## 3.5 Lanczos-Interpolation

Die Lanczos-Interpolation ist eine Methode zur Rekonstruktion von Werten im Bild aus diskreten Abtastungen. In diesem Abschnitt wird die mathematische Grundlage und die praktische Anwendung der Lanczos-Interpolation kurz erläutert.

---

<sup>7</sup>K.T. GRIBBON und D.G. BAILEY [2004]. »A novel approach to real-time bilinear interpolation«. In: *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, S. 126–131. DOI: 10.1109/DELTA.2004.10055.

### 3.5.1 Mathematische Grundlage

Die Lanczos-Interpolation basiert auf der Idee, ein kontinuierliches Signal  $f(x)$  durch eine Summe von gewichteten Basisfunktionen zu approximieren. Dieses Signal können zum Beispiel Farbwerte in einem Bild sein. Die Basisfunktionen werden durch das sogenannte Lanczos-Kernel definiert:

$$L(x) = \begin{cases} \frac{a \cdot \sin(\pi \cdot x) \cdot \sin(\pi \cdot x/a)}{(\pi \cdot x)^2} & \text{wenn } |x| < a \\ 0 & \text{sonst} \end{cases} \quad (3.2)$$

<sup>8</sup>

Das Lanczos-Kernel hat eine kompakte Trägerfunktion. Eine kompakte Trägerformel bedeutet, sie ist nur in einem begrenzten Bereich von Null verschieden. In der Praxis hat dies den Vorteil, dass das Signalrauschen in Bereichen außerhalb des Bereichs im Signalraum reduziert wird und somit eine bessere Interpolation des Signals erreicht werden kann. Die Gewichtungen der Basisfunktionen werden durch die Interpolationskoeffizienten bestimmt, die durch die diskreten Abtastungen des Signals berechnet werden.

Die Lanczos-Interpolation wird in der Regel auf gleichmäßig verteilten Stützstellen angewendet<sup>9,10</sup>. Die Interpolationsmethode verwendet diese Stützstellen als Ausgangspunkt, um eine Schätzung des Signals an anderen Orten zu berechnen. Seien  $x_1, x_2, \dots, x_n$  die Stützstellen des Signals und  $y_1, y_2, \dots, y_n$  die zugehörigen Abtastungen. Die Interpolationsfunktion  $s(x)$  kann dann wie folgt berechnet werden:

$$s(x) = \sum_{i=1}^n y_i \cdot L(x - x_i) \quad (3.3)$$

Hierbei ist  $L(x - x_i)$  die Lanczos-Kernel-Funktion, die den Beitrag des i-ten Stützpunkts zum interpolierten Signal an der Position x angibt. Die Gewichtung erfolgt durch Multiplikation der Abtastung  $y_i$  mit dem Wert des Lanczos-Kernels für den Abstand

---

<sup>8</sup>Claude E DUCHON [1979]. »Lanczos filtering in one and two dimensions«. In: *Journal of Applied Meteorology and Climatology* 18.8, S. 1016–1022.

<sup>9</sup>Louis KOMZSIK [2003]. *The Lanczos method: evolution and application*. SIAM.

<sup>10</sup>WHUBER [Sep. 2011]. *What is Lanczos resampling?* <https://gis.stackexchange.com/a/14361>.

zwischen der Stützstelle  $x_i$  und dem Zielpunkt x.

Die Wahl des Parameters  $a$  beeinflusst die Schärfe der Interpolation. Ein kleinerer Wert von  $a$  führt zu einer breiteren Trägerfunktion und einer glatteren Interpolation, während ein größerer Wert von  $a$  zu einer schärferen Trägerfunktion und einer detailreicheren Interpolation führt. Es ist wichtig zu beachten, dass bei zu großen Werten von  $a$  sogenannte Ringing-Artefakte auftreten können, bei denen sich unerwünschte Oszillationen um Kanten oder Kontrastübergänge im Bild bilden.

Insgesamt bietet die Lanczos-Interpolation eine effektive Methode zur Skalierung von Bildern mit Erhaltung von Details und Schärfe. Sie ist jedoch rechenaufwändiger als einfache Interpolationsmethoden, da sie eine Berechnung für jeden Pixel im skalierten Bild erfordert.<sup>11</sup>

### 3.5.2 Praktische Anwendung

Die Lanczos-Interpolation findet Anwendung in vielen Bereichen der Bildverarbeitung. Ein Anwendungsbeispiel ist die Upsampling von digitalen Bildern, um eine höhere Auflösung zu erreichen.

Die praktische Umsetzung der Lanczos-Interpolation erfordert die Berechnung der Interpolationskoeffizienten und die Bestimmung der Stützstellen des Signals. In der Regel werden hierfür spezielle Algorithmen eingesetzt, die auf der effizienten Berechnung der Basisfunktionen basieren.

```
from backend.image import Image
import math

class LanczosInterpolation(Image):
    def __init__(self, path):
        extend = "lcz"
```

---

<sup>11</sup>A.H. BENTBIB u. a. [2016]. »A global Lanczos method for image restoration«. In: *Journal of Computational and Applied Mathematics* 300, S. 233–244. ISSN: 0377-0427. DOI: <https://doi.org/10.1016/j.cam.2015.12.034>. URL: <https://www.sciencedirect.com/science/article/pii/S0377042715006469>.

```

super().__init__(path, extend)

def lanczos_kernel(self, x, a=2):
    if x == 0:
        return 1
    elif abs(x) >= a:
        return 0
    else:
        return math.sin(math.pi * x) * math.sin(math.pi * x / a) /
(math.pi ** 2 * x ** 2)

def manipulate(self, new_size):
    super().manipulate(new_size)

    for i in range(self.new_width):
        for j in range(self.new_height):
            x, y = i * self.width / self.new_width, j *
self.height / self.new_height
            u, v = math.floor(x), math.floor(y)
            s, t = x - u, y - v

            pixel = (0, 0, 0)
            weight_sum = 0

            for m in range(u - 2, u + 3):
                for n in range(v - 2, v + 3):
                    if 0 <= m < self.width and 0 <= n < self.height:
                        weight = self.lanczos_kernel(s - (m - u)) *
self.lanczos_kernel(t - (n - v))
                        pixel = (pixel[0] + weight * self.image[s][t][0],
                                pixel[1] + weight * self.image[s][t][1],
                                pixel[2] + weight * self.image[s][t][2])
                        weight_sum += weight
```
```

```

        pixel = tuple([p + weight * self.img.getpixel(
(m, n))[i] for i, p in enumerate(pixel)])
        weight_sum += weight
        if weight_sum > 0:
            pixel = tuple([int(p / weight_sum) for p in pixel])
            self.newImg.putpixel((i, j), pixel)

    return self.save()

```

## 3.6 Zusammenfassung von Bildverarbeitungstechniken

In der Bildverarbeitung gibt es verschiedene Techniken, die verwendet werden können, um ein Bild zu bearbeiten oder zu manipulieren. In diesem Abschnitt werden wir uns mit einigen gängigen Techniken zur Interpolation von Bildern beschäftigen, nämlich Pixelverdopplung, Nearest Neighbour Interpolation, Bilineare Interpolation, Bikubische Interpolation und Lanczos Interpolation. Wir werden jeweils auf die Vor- und Nachteile dieser Techniken eingehen.

### 3.6.1 Pixelverdopplung

Bei der Pixelverdopplung wird jedes Pixel im Bild einfach dupliziert, um ein größeres Bild zu erzeugen. Diese Methode ist einfach und schnell, aber sie führt oft zu einer Verzerrung des Bildes und kann zu einer verschlechterten Bildqualität führen.

### 3.6.2 Nearest Neighbour Interpolation

Die Nearest Neighbour Interpolation ist eine einfache Interpolationsmethode, bei der jeder neue Pixelwert durch den nächstgelegenen Pixelwert im Originalbild bestimmt wird. Diese

Methode ist einfach und schnell, aber sie führt oft zu einem "Treppeneffekt" Bild, da die Pixelwerte nicht kontinuierlich interpoliert werden.

### 3.6.3 Bilineare Interpolation

Die bilineare Interpolation ist eine Methode, bei der die neuen Pixelwerte aus einer bilinearen Funktion berechnet werden, die aus den vier nächstgelegenen Pixeln im Originalbild abgeleitet wird. Diese Methode führt oft zu einer glatteren Interpolation als die Nearest Neighbour Interpolation, aber es kann immer noch zu einem "Verwischen" des Bildes kommen.

### 3.6.4 Bikubische Interpolation

Die bikubische Interpolation ist eine Methode, bei der die neuen Pixelwerte aus einer bikubischen Funktion berechnet werden, die aus den sechzehn nächstgelegenen Pixeln im Originalbild abgeleitet wird. Diese Methode führt oft zu einer noch glatteren Interpolation als die bilineare Interpolation, aber sie kann auch zu einer Überbetonung von Bildstrukturen führen.

### 3.6.5 Lanczos Interpolation

Die Lanczos Interpolation ist eine Methode, bei der die neuen Pixelwerte aus einer Lanczos-Funktion berechnet werden, die aus einer begrenzten Anzahl von nächstgelegenen Pixeln im Originalbild abgeleitet wird. Diese Methode führt oft zu einer sehr glatten Interpolation und reduziert das Rauschen im Bild, aber sie kann auch zu einer gewissen Unschärfe im Bild führen.

## 3.7 Analyse der Leistung

Es wird erläutert, welche Kriterien zur Beurteilung von Bildinterpolationsverfahren verwendet werden, darunter die visuelle Qualität, die Genauigkeit und die Effizienz der

Berechnung.

Untersuchung der Auswirkungen verschiedener Parameter wie Bildgröße, Auflösung und Inhalt auf die Leistung der bikubischen Interpolation. Diskussion der potenziellen Einschränkungen und Kompromisse, die mit der Verwendung der bikubischen Interpolation verbunden sind, sowie der Faktoren, die ihre Leistung in verschiedenen Kontexten beeinflussen können.

| Interpolation Typ | Subjektive Einschätzung | Evaluation   | Verarbeitungszeit |
|-------------------|-------------------------|--------------|-------------------|
| Nearest Neighbour | offensichtliches Mosaik | zu Verpixelt | 5s                |
| Bilinear          | unscharf, nicht scharf  | Schlechte    | 6s                |
| Bicubic           | verschwommen, schärfer  | Bessere      | 8s                |
| BC-Spline         | relativ klar, scharf    | Gute         | 17s               |

Tabelle 3.1: Kontrasttabelle der subjektiven Bewertung mit verschiedenen Interpolationsmethoden [HAN 2013]

Die vorliegende Tabelle 3.2 zeigt den Kontrast der PSNR bei verschiedenen Interpolationsmethoden. Die getesteten Bilder wurden mit der Nearest Neighbour-, Bilinear- und Bicubic Interpolation sowie Cubic B-Spline verarbeitet. Die Tabelle gibt die PSNR-Werte für jedes getestete Bild an. Höhere PSNR-Werte deuten auf eine bessere Bildqualität hin. Die Bicubische Interpolation zeigt die höchsten PSNR-Werte, gefolgt von der Cubic B-Spline-Interpolation. Die Nearest-Neighbour-Interpolation weist die niedrigsten PSNR-Werte auf.

Der vorliegende Graph 3.5<sup>12</sup> zeigt die durchschnittliche Laufzeit verschiedener Interpolationsalgorithmen. Es ist zu beachten, dass die Komplexität der verschiedenen Algorithmen variiert und somit auch ihre durchschnittliche Laufzeit erheblich unterschiedlich ist. Die Nearest Neighbour Interpolation weist die geringste durchschnittliche Laufzeit auf von 5 Sekunden, gefolgt von der Bilinearen Interpolation mit 6 Sekunden. Der Algorithmus der

<sup>12</sup>Dianyuan HAN [2013]. »Comparison of commonly used image interpolation methods«. In: *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. Atlantis Press, S. 1556–1559.

| Tested Image | Nearest Neighbour | Bilinear  | Bicubic  | Cubic B-Spline |
|--------------|-------------------|-----------|----------|----------------|
| image 1      | 19.1112           | 23.3023   | 23.4204  | 22.5028        |
| image 2      | 16.0059           | 18.4308   | 18.5899  | 17.9319        |
| image 3      | 20.0614           | 25.3736   | 25.6634  | 24.3505        |
| image 4      | 15.9497           | 18.7056   | 18.9392  | 18.1074        |
| image 5      | 17.1252           | 20.2387   | 20.9872  | 18.6582        |
| image 6      | 18.15564          | 22.137185 | 22.24938 | 21.37766       |
| image 7      | 14.40531          | 16.58772  | 16.73091 | 16.13871       |
| image 8      | 17.05219          | 21.56756  | 21.81389 | 20.697925      |
| image 9      | 12.75976          | 14.96448  | 15.15136 | 14.48592       |
| image 10     | 12.8439           | 15.179025 | 15.7404  | 13.99365       |

Tabelle 3.2: Contrast table of SNR with different interpolation method[HAN 2013]

Bicubischen Interpolation zeigt im Vergleich dazu eine längere durchschnittliche Laufzeit von 8 Sekunden (60% langer als Nearest Neighbour). Die Unterschiede zeigen sich Bedeutend bei Anwendungen, die Echtzeitverarbeitung erfordern oder bei großen Datenmengen oder in Webapplikationen.

Die Beurteilung von Bildinterpolationsverfahren erfolgt anhand diverser Kriterien, welche ihre Leistung charakterisieren. Zu den frequent verwendeten Kriterien zählen die visuelle Qualität, die Genauigkeit sowie die Berechnungseffizienz. Die visuelle Qualität bezieht sich auf die subjektive Wahrnehmung der Bildqualität nach der Interpolation. Wesentliche Aspekte der visuellen Qualität beinhalten die Schärfe der Details, das Vorhandensein von Artefakten (z.B. Unschärfe oder Kantenartefakte) und die allgemeine Ästhetik des interpolierten Bildes. Die Genauigkeit eines Interpolationsverfahrens betrifft dessen Fähigkeit, die tatsächlichen Werte oder Strukturen des Originalbildes möglichst präzise zu reproduzieren. Eine präzise Interpolation minimiert den Informationsverlust und wahrt die inhärente Qualität des Bildes. Die Berechnungseffizienz betrachtet die Laufzeit- und Ressourcenanforderungen des Interpolationsverfahrens. Zügige und effiziente Methoden ermöglichen eine reibungslose Verarbeitung großer Datenmengen in Echtzeit oder bei der

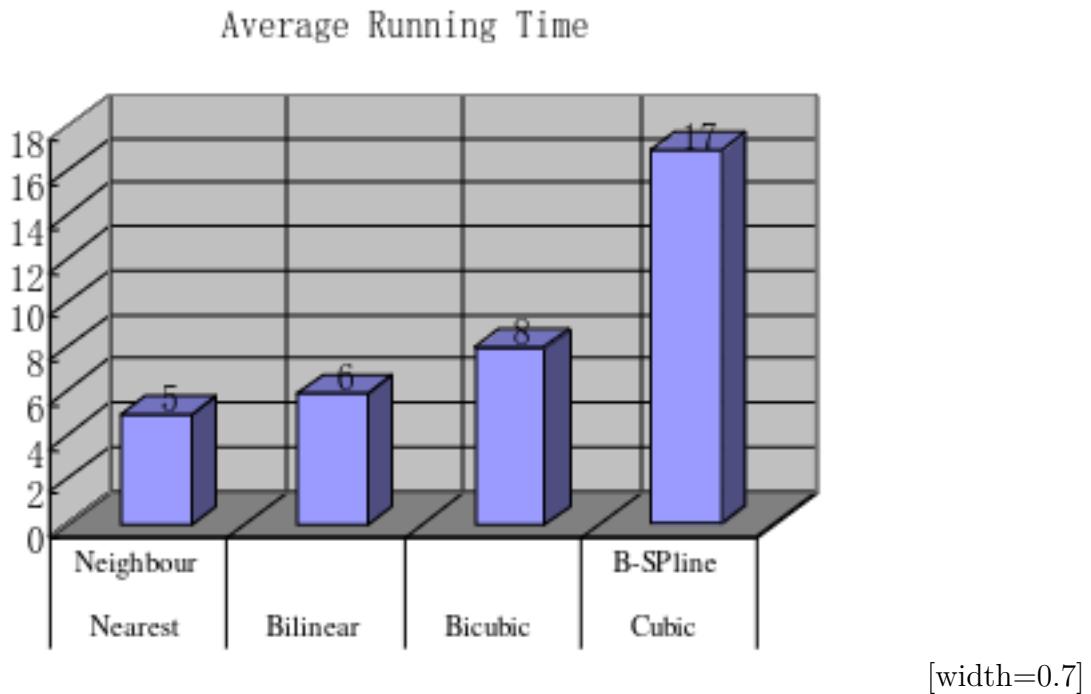


Abbildung 3.5: Durchschnittliche Laufzeit verschiedener Interpolationsalgorithmen in Sekunden[HAN 2013].

Stapelverarbeitung<sup>13</sup>.

Beispiele aus dem wirklichen Leben umfassen Ladezeiten für Webseiten, medizinische Bildgebung oder Fernerkundung. In wissenschaftlichen Anwendungen wie der medizinischen Bildgebung oder der Fernerkundung ist es von besonderer Bedeutung, dass präzise Messungen und Analysen durchführbar sind. Eine präzise Interpolation minimiert den Informationsverlust und wahrt die inhärente Qualität des Bildes. Die Berechnungseffizienz spielt eine ausschlaggebende Rolle in Anwendungen wie der Videokodierung oder der Echtzeitbildverarbeitung.

Es ist essenziell zu beachten, dass die Leistung der Interpolationsverfahren von diversen Faktoren abhängt, einschließlich der Bildgröße, der Auflösung, des Inhalts und

---

<sup>13</sup>Yongqiang CAI, Qianxiao LI und Zuowei SHEN [2019]. »A quantitative analysis of the effect of batch normalization on gradient descent«. In: *International Conference on Machine Learning*. PMLR, S. 882–890.

des Kontextes der Anwendung. Ein Verfahren, welches in einer bestimmten Situation gute Ergebnisse erzielt, kann in einer anderen Situation weniger erfolgreich sein. Deshalb ist es wichtig, die spezifischen Anforderungen und Beschränkungen einer Anwendung zu berücksichtigen, um das am besten geeignete Interpolationsverfahren auszuwählen.

### **3.7.1 Vergleich und Zusammenfassung**

Insgesamt gibt es keine "beste" Methode zur Interpolation von Bildern, da jede Methode ihre eigenen Vor- und Nachteile hat. Die Wahl der Methode hängt von den spezifischen Anforderungen und Einschränkungen ab, die für die jeweilige Anwendung gelten. Die Wahl einer geeigneten Interpolationsmethode kann jedoch die Bildqualität erheblich verbessern und zu einer effektiveren Bildverarbeitung führen.

# Kapitel 4

## Fortgeschrittene Skalierungsmethoden

In diesem Kapitel werden fortgeschrittene Skalierungsmethoden behandelt. Unter Beachtung der Erhaltung der Bildqualität werden Convolutional Neural Networks (CNNs) sowie Deep Learning zur Bildskalierung betrachtet. Des Weiteren wird die Super Resolution vorgestellt. Super Resolution ist eine Methode zur Generierung hochauflösender Bilder aus niedrigauflösenden Versionen, einschließlich der Wiederherstellung von verlorenen Details.

Ein weiterer Ansatz, ist der Einsatz von Generative Adversarial Networks (GANs), die durch den Wettbewerb zwischen einem Generator-Netz und einem Diskriminator-Netz hochrealistische Bilder erzeugen können. Außerdem gibt es die Multiskalenskalierungstechnik, bei der verschiedene Skalierungsfaktoren angewendet werden, um eine ausgewogene Balance zwischen Details und Vergrößerung zu erreichen. Abschließend werden die Vor- und Nachteile dieser fortgeschrittenen Methoden vorgestellt, um ein umfassendes Verständnis für ihre Anwendungsmöglichkeiten und Einschränkungen in der Bildverarbeitung zu entwickeln.

## 4.1 Convolutional Neural Networks / Deep learning

### 4.1.1 Grundlagen von Convolutional Neural Networks (CNNs)

Convolutional Neural Networks CNN sind eine Art von Deep-Learning-Modell, welches besonders im Hinblick auf die Verarbeitung von Daten mit räumlicher Struktur den aktuellen Stand der Technik darstellt. Räumliche Daten, wie z.B. Bilder können bearbeitet, verarbeitet, erstellt und hauptsächlich analysiert werden. CNNs bestehen aus mehreren Schichten von Neuronen, die so angeordnet sind, dass sie Merkmale extrahieren und Objekte Klassifizieren können. Die grundlegende Idee hinter CNNs ist die Verwendung von Faltung (engl. convolution) zur Merkmalsextraktion sowie pooling layers. Pooling layers sparen Rechenleistung (Abb. 7.4.1) und ermöglicht eine effektivere sowie schnellere Verarbeitung von großen Datensätzen<sup>1,2</sup>.

### 4.1.2 Architekturen von CNNs

Es gibt mehrere bekannte Architekturen von CNNs, darunter AlexNet<sup>3</sup>, ResNet<sup>4</sup> und Inception<sup>5</sup>.

AlexNet<sup>6</sup> war das erste CNN, das auf einem großen Datensatz erfolgreich angewendet

---

<sup>1</sup>Zewen LI u. a. [Dez. 2022]. »A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects«. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12, S. 6999–7019. DOI: 10.1109/tnnls.2021.3084827. URL: {<https://doi.org/10.1109/tnnls.2021.3084827>}.

<sup>2</sup>Tobias KOLB [o. D.] »Entwicklung eines Convolutional Neural Network zur Handschrifterkennung«. In: *Angewandtes maschinelles Lernen–SS2019* [], S. 28; Keiron O'SHEA und Ryan NASH [2015]. *An Introduction to Convolutional Neural Networks*. arXiv: 1511.08458 [cs.NE].

<sup>3</sup>Neena ALOYSIUS und M. GEETHA [Apr. 2017]. »A review on deep convolutional neural networks«. In: *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE. DOI: 10.1109/iccsp.2017.8286426. URL: <https://doi.org/10.1109/iccsp.2017.8286426>.

<sup>4</sup>Pengcheng YUAN u. a. [2020]. *HS-ResNet: Hierarchical-Split Block on Convolutional Neural Network*. DOI: 10.48550/ARXIV.2010.07621. URL: <https://arxiv.org/abs/2010.07621>.

<sup>5</sup>Md Zahangir ALOM u. a. [2017]. *Inception Recurrent Convolutional Neural Network for Object Recognition*. DOI: 10.48550/ARXIV.1704.07709. URL: <https://arxiv.org/abs/1704.07709>.

<sup>6</sup>Neena ALOYSIUS und M. GEETHA [Apr. 2017]. »A review on deep convolutional neural networks«. In: *2017 International Conference on Communication and Signal Processing (ICCSP)*. IEEE. DOI:

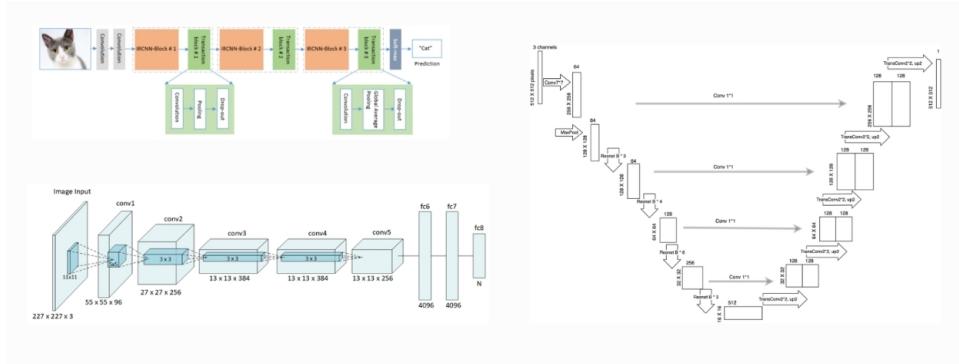


Abbildung 4.1: Verschiedene Architekturen von Image Networks. Oben links vereinfacht Alex net, unten links inception net, rechts Abbildung der Res net Architektur.

wurde. ResNet zeichnet sich durch seine Fähigkeit aus, sehr tiefe Netze zu trainieren, ohne dass das Problem des Verschwindens des Gradienten auftritt<sup>7</sup>. Inception wiederum ist für seine Fähigkeit bekannt, die Effizienz von CNNs durch die Verwendung von sogenannten Inception-Modulen zu erhöhen<sup>8</sup>.

### 4.1.3 Anwendungen von CNNs

CNNs haben zahlreiche Anwendungen, darunter Bildklassifizierung, Objekterkennung und Gesichtserkennung. Bei der Bildklassifizierung werden Bilder automatisch in verschiedene Kategorien eingeteilt. Beispielsweise können Bilder in Klassen wie Hunde, Katzen oder Autos eingeteilt werden. Bei der Objekterkennung wird das Modell darauf trainiert, bestimmte Objekte in einem Bild zu erkennen, wie z.B. Personen oder Straßenschilder. Die Gesichtserkennung wird oft zur Identifikation von Personen in Sicherheitsanwendungen

10.1109/icccsp.2017.8286426. URL: <https://doi.org/10.1109/icccsp.2017.8286426>.

<sup>7</sup>Pengcheng YUAN u.a. [2020]. *HS-ResNet: Hierarchical-Split Block on Convolutional Neural Network*.

DOI: 10.48550/ARXIV.2010.07621. URL: <https://arxiv.org/abs/2010.07621>.

<sup>8</sup>Md Zahangir ALOM u.a. [2017]. *Inception Recurrent Convolutional Neural Network for Object Recognition*. DOI: 10.48550/ARXIV.1704.07709. URL: <https://arxiv.org/abs/1704.07709>.

eingesetzt.<sup>9,10</sup>

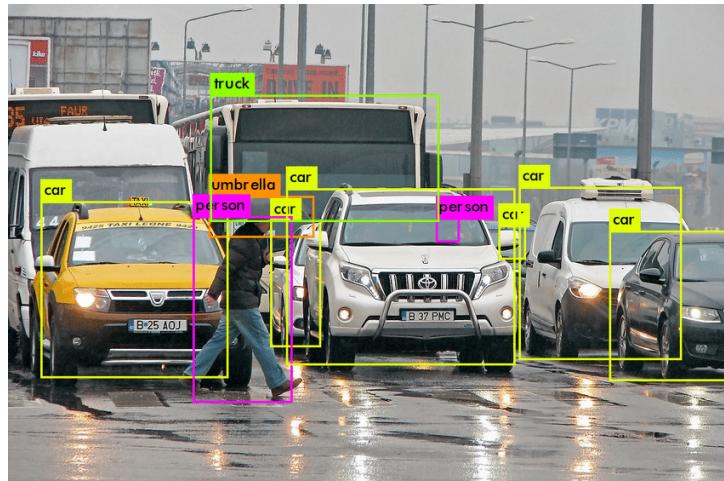


Abbildung 4.2: Beispielbild eines Frames mit Bounding Boxes

#### 4.1.4 Transfer Learning mit CNNs

Transfer Learning ist eine Technik, bei der ein bereits trainiertes CNN auf eine neue Aufgabe angewendet wird, ohne es von Grund auf neu zu trainieren. Dies ist nützlich, wenn man nur über begrenzte Trainingsdaten verfügt oder wenn das Trainieren eines neuen Modells zu viel Zeit oder Ressourcen in Anspruch nimmt. Ein Beispiel hierfür ist Erkennung von Autos, bei denen das Modell auf einem bereits trainierten CNN basieren kann, das auf z.B. Yolo v8 trainiert wurde<sup>11</sup>.

<sup>9</sup>Max JADERBERG, Karen SIMONYAN, Andrew ZISSERMAN u. a. [2015]. »Spatial transformer networks«. In: *Advances in neural information processing systems* 28; Ziwei LIU u. a. [2015]. »Deep learning face attributes in the wild«. In: *Proceedings of the IEEE international conference on computer vision*, S. 3730–3738; Bichen WU u. a. [2017]. »SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, S. 129–137.

<sup>10</sup>[https://www.researchgate.net/figure/Object-detection-in-a-dense-scene\\_fig4\\_329217107](https://www.researchgate.net/figure/Object-detection-in-a-dense-scene_fig4_329217107)

<sup>11</sup>Huy Hoang NGUYEN u. a. [Jan. 2021]. »YOLO Based Real-Time Human Detection for Smart Video Surveillance at the Edge«. In: *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*. IEEE. DOI: 10.1109/icce48956.2021.9352144. URL: <https://doi.org/10.1109/icce48956.2021.9352144>

In unserem Projekt haben wir das Modell resnet50 zur Trainingsgrundlage genommen, um unser eigenes Modell zu entwickeln. Wir haben den Code, den wir verwendet haben, um dieses Training durchzuführen, im Folgenden aufgeführt:

```
Net = torchvision.models.segmentation.deeplabv3_resnet50(pretrained=True)
```

Dieser Code<sup>12</sup> ermöglichte es uns, die Leistungsfähigkeit von resnet50 auszunutzen und es an unsere spezifischen Anforderungen anzupassen. Durch das Training unseres Modells konnten wir die gewünschten Ergebnisse erzielen und wichtige Erkenntnisse gewinnen, ohne ein Model von Grund auf zu trainieren.

#### 4.1.5 Limitationen von CNNs und aktuelle Forschungsziele

Obwohl CNNs sehr erfolgreich bei der Verarbeitung von Bildern sind, haben sie auch einige Limitationen. Zum Beispiel sind sie nicht gut geeignet, um komplexe Abhängigkeiten zwischen verschiedenen Eingabemerkmale zu erfassen, wie z.B. das Verhalten von Objekten in einem Video. Zudem benötigen CNNs weiterhin viel Rechenleistung und Ressourcen<sup>13</sup>

Facebooks Segment Anything Model (SAM) ist ein neues und Open-Source Modell, das sowohl interaktive als auch automatische Segmentierung durchführen kann. Die Benutzeroberfläche des Modells ermöglicht es, es auf flexible Weise zu verwenden, was eine Vielzahl von Segmentierungsaufgaben durch einfache Ingenieursarbeiten möglich macht. Das Modell wurde auf einem Datensatz von 11 Millionen Bildern und 1,1 Milliarden Masken trainiert und hat eine starke Null-Schuss-Leistung bei einer Vielzahl von Segmentierungsaufgaben<sup>14</sup>.

---

icce48956.2021.9352144.

<sup>12</sup>[https://github.com/studienarbeit-cnn-dhwka-2022/Segmentation\\_cnn/blob/main/cnn.ipynb](https://github.com/studienarbeit-cnn-dhwka-2022/Segmentation_cnn/blob/main/cnn.ipynb)

ipynb

<sup>13</sup>Hao Li u. a. [2018]. »Visualizing the loss landscape of neural nets«. In: *Advances in neural information processing systems* 31.

<sup>14</sup><https://research.facebook.com/publications/segment-anything/>



Abbildung 4.3: SAM: A generalized approach to segmentation

## 4.2 Super Resolution

### 4.2.1 Grundlagen von Super Resolution

Super Resolution Super Resolution (SR) ist eine Technik, um aus einer niedrig aufgelösten Eingabe ein hochauflösendes Bild zu generieren. Dies wird oft als Upscaling bezeichnet und findet in vielen Anwendungen wie der Bildrekonstruktion und Videoverarbeitung Anwendungen. Die Grundidee hinter SR ist, dass hochauflösende Informationen in einem niedrig aufgelösten Bild versteckt sein können. Die Herausforderung besteht darin, diese Informationen zu extrahieren und in ein hochauflösendes Bild zu integrieren. SR ist somit ein Problem der inversen Bildgebung, bei dem eine hohe Auflösung aus einer niedrigen Auflösung abgeleitet werden muss.<sup>15,16</sup>

### 4.2.2 Super Resolution-Methoden auf Basis von Deep Learning

Super Resolution-Methoden auf Basis von Deep Learning haben in den letzten Jahren viel Aufmerksamkeit erhalten und sind derzeit der Stand der Technik für SR. Diese

<sup>15</sup>Chao DONG u. a. [2016]. »Image Super-Resolution Using Deep Convolutional Networks«. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2, S. 295–307. doi: 10.1109/TPAMI.2015.2439281.

<sup>16</sup>J. KIM und S. LEE [2019]. »Deep Learning-Based Scaling Methods for Image Super-Resolution«. In: *Pattern Recognition*.

Methoden verwenden Convolutional Neural Networks (CNNs) zur Verarbeitung von Bildern und zur Generierung von hochauflösenden Bildern. Es gibt verschiedene Arten von SR-Methoden auf Basis von Deep Learning, darunter Single-Image Super Resolution (SISR) und Multi-Image Super Resolution (MISR). SISR-Methoden verwenden nur ein niedrig aufgelöstes Bild als Eingabe, während MISR-Methoden mehrere Bilder verwenden, um ein hochauflösendes Bild zu generieren.

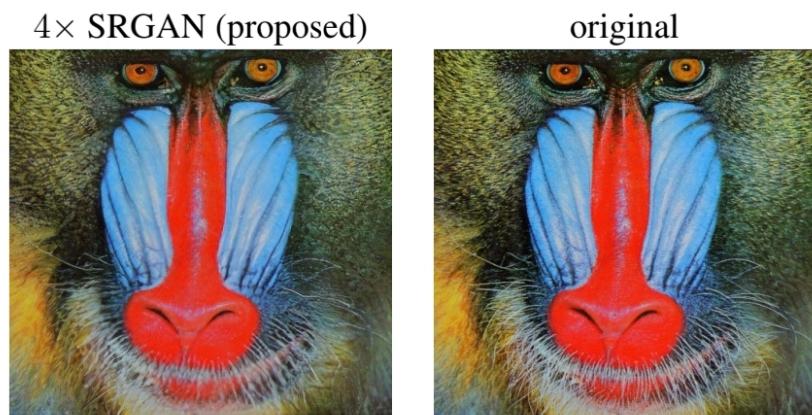


Abbildung 4.4: Super resolution links, original rechts

## Anwendungen von SR

SR hat viele Anwendungen in der Bild- und Videoanalyse, einschließlich der Rekonstruktion von Bildern aus medizinischen Scans, der Verbesserung von Bildern für die forensische Analyse<sup>17</sup> und der Verbesserung von Bildern für die Erkennung von Gesichtern und Objekten. In der Videoanalyse kann SR verwendet werden, um Videos zu skalieren<sup>18</sup>, indem Bewegungsunschärfe reduziert und die Schärfe der Bilder verbessert wird. SR kann auch bei der Entschlüsselung von unscharfen und verschwommenen Bildern in

<sup>17</sup>Ashish GEHANI und John REIF [2007]. »Super-Resolution Video Analysis for Forensic Investigations«. In: *Advances in Digital Forensics III*. Hrsg. von Philip CRAIGER und Sujeet SHENOI. New York, NY: Springer New York, S. 281–299. ISBN: 978-0-387-73742-3.

<sup>18</sup>Ning FANG und Zongqian ZHAN [2022]. »High-resolution optical flow and frame-recurrent network for video super-resolution and deblurring«. In: *Neurocomputing* 489, S. 128–138.

Überwachungsaufnahmen<sup>19</sup> helfen.

### Evaluierung von SR-Methoden

Die Evaluierung von SR-Methoden ist eine wichtige Aufgabe, um die Qualität und Effektivität der generierten Bilder zu bestimmen. Die gängigen Evaluierungsmethoden umfassen die Verwendung von visuellen Qualitätsmetriken wie Peak Signal-to-Noise Ratio (PSNR) und Structural Similarity Index Measure (SSIM). Es gibt auch speziellere Evaluierungsmethoden wie die Verwendung von Perceptual Quality Assessment (PQA)-Maßnahmen, die menschliche Wahrnehmungseigenschaften berücksichtigen, um die Qualität der generierten Bilder zu bestimmen.

### Herausforderungen und zukünftige Forschungsziele von Super Resolution

Obwohl SR-Methoden auf Basis von Deep Learning vielversprechende Ergebnisse erzielt haben<sup>20</sup>, gibt es immer noch Herausforderungen und zukünftige Forschungsziele, die erforscht werden müssen. Eine der Herausforderungen besteht darin, dass SR-Methoden häufig dazu neigen, Artefakte<sup>21</sup> in den generierten Bildern zu erzeugen, insbesondere bei der Verwendung von sehr hohen Upscaling-Faktoren. Dies kann die visuelle Qualität der generierten Bilder beeinträchtigen und die Anwendbarkeit von SR-Methoden in bestimmten Szenarien einschränken.

Eine weitere Herausforderung besteht darin, dass SR-Methoden häufig sehr rechenaufwändig sind, insbesondere wenn sie auf großen Datensätzen oder in Echtzeit angewendet werden müssen. Die benötigten Ressourcen sind teuer. Dies kann die praktische Anwendbarkeit von SR-Methoden in einigen Anwendungen einschränken. Zukünftige Forschungsziele könnten sich darauf konzentrieren, diese Herausforderungen zu überwinden, indem sie

---

<sup>19</sup>Seiichi GOHSHI [2015]. »Real-time super resolution algorithm for security cameras«. In: *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*. Bd. 5. IEEE, S. 92–97.

<sup>20</sup>Zhou WANG u. a. [2004a]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE transactions on image processing* 13.4, S. 600–612.

<sup>21</sup>Jiaming WANG u. a. [2022]. »From artifact removal to super-resolution«. In: *IEEE Transactions on Geoscience and Remote Sensing* 60, S. 1–15.

neue SR-Methoden entwickeln, die sowohl effektiv als auch effizient sind. Eine mögliche Lösung wäre die Verwendung von Generative Adversarial Networks (GANs) zur Verbesserung der visuellen Qualität der generierten Bilder und zur Reduzierung von Artefakten. Eine weitere mögliche Lösung wäre die Entwicklung von neuartigen Architekturen von Deep Learning-Netzen, die weniger rechenaufwändig sind und schneller ausgeführt werden können<sup>22,23</sup>.

### Deep Learning Super Sampling

Eine vielversprechende Lösung zur Beschleunigung von Deep Learning ist Nvidia's DLSS-3 (Deep Learning Super Sampling 3, Stand 2023 Mai 17)<sup>24</sup>. DLSS ist eine fortschrittliche Technologie, die von NVIDIA entwickelt wurde und es ermöglicht, hochauflösende Grafiken in Echtzeit zu rendern, indem es auf KI-Methoden basiert und durch Hardware (GPU) beschleunigt wird. DLSS bietet nicht nur schnellere Renderzeiten, sondern ermöglicht auch eine verbesserte Leistung auf Grafikkarten, da weniger rechenintensive Berechnungen erforderlich sind. Durch extensive Nutzung der CUDA cores, welche immer imposanter in Grafikkarten eingebaut werden, ist das Berechnen von Hohchauflösenden Bildern schneller mit jeder Generation.

Insgesamt bleibt SR ein aktives Forschungsfeld mit großem Potenzial für Anwendungen in der Bild- und Videoanalyse. Mit weiteren Fortschritten in der Forschung können SR-Methoden immer leistungsfähiger und praktischer werden, um die Bedürfnisse der Industrie und der Gesellschaft zu erfüllen.

---

<sup>22</sup>Chao DONG, Chen Change LOY und Xiaou TANG [2016]. *Accelerating the Super-Resolution Convolutional Neural Network*. arXiv: 1608.00367 [cs.CV].

<sup>23</sup>Jangsoo PARK, Jongseok LEE und Donggyu SIM [Sep. 2020]. »Low-complexity CNN with 1D and 2D filters for super-resolution«. In: *Journal of Real-Time Image Processing* 17.6, S. 2065–2076. DOI: 10.1007/s11554-020-01019-1. URL: <https://doi.org/10.1007/s11554-020-01019-1>.

<sup>24</sup><https://www.nvidia.com/en-us/geforce/news/dlss3-ai-powered-neural-graphics-innovations/>

## 4.3 Generative Adversarial Networks

### 4.3.1 Grundlagen von Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) sind ein leistungsstarkes Framework für das Training von Deep Learning-Modellen zur Generierung von Daten. GANs bestehen aus zwei miteinander konkurrierenden neuronalen Netzen: einem Generator und einem Diskriminatoren.

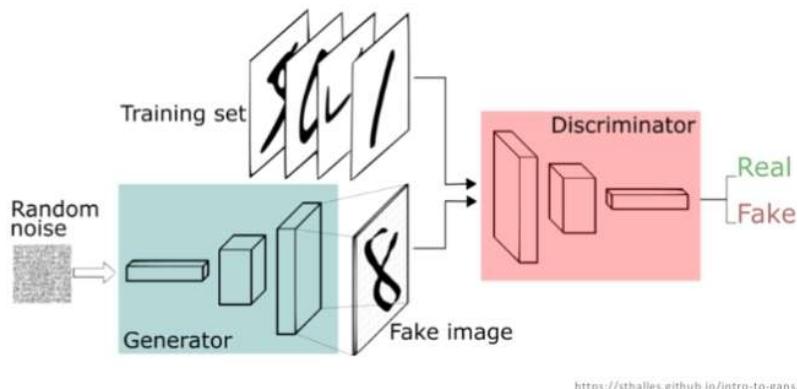


Abbildung 4.5: Der Generator erzeugt neue Daten, während der Diskriminatoren versucht, zwischen den vom Generator erzeugten Daten und den echten Daten zu unterscheiden.

Im Laufe des Trainings passt sich der Generator kontinuierlich an und verbessert seine Fähigkeit, realistische Daten zu generieren, während der Diskriminatoren gleichzeitig verbessert wird, um zwischen den generierten und echten Daten zu unterscheiden.

### 4.3.2 Architekturen von GANs

Es gibt verschiedene Architekturen von GANs, die für verschiedene Arten von Anwendungen geeignet sind. In Beispiel ist das Deep Convolutional GAN (DCGAN), das speziell für die Generierung von Bildern entwickelt wurde. DCGAN nutzt Convolutional Neural Networks (CNNs) und Transposed Convolutional Neural Networks, um Bilder zu generieren, die visuell realistisch aussehen und strukturell konsistent sind. Ein weiteres Beispiel ist

das CycleGAN<sup>25</sup>, das für die Bildübersetzung zwischen verschiedenen Domänen verwendet werden kann. CycleGAN nutzt einen Generator und einen Diskriminator sowie zusätzliche Cycle-Verlustfunktionen, um die Transformationen zwischen den Bildern in verschiedenen Domänen zu erlernen.

### 4.3.3 Anwendungen von GANs

GANs finden Anwendungen in verschiedenen Bereichen wie der Bildgenerierung, Style Transfer, der Verbesserung von Bildern und der Videoanalyse. Zum Beispiel können GANs verwendet werden, um realistisch aussehende Bilder von Gesichtern, Landschaften oder anderen Objekten zu generieren.



Abbildung 4.6: Eingabe in das Modell: A small cactus wearing a straw hat and neon sunglasses in the Sahara desert.: <https://Imagen.research.google/>

Style Transfer ermöglicht die Veränderung des visuellen Erscheinungsbildes von Bildern durch die Übertragung des Stils von einem Bild auf ein anderes. Dabei wird ein vortrainiertes neuronales Netz verwendet, um Merkmale des Stils und Inhalts eines Bildes zu erfassen und zu kombinieren. Diese Technik hat Anwendungen in der Bildbearbeitung, Kunstgenerierung und visuellen Effekterstellung gefunden.

<sup>25</sup>Jun-Yan ZHU u. a. [2017]. »Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks«. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*.

Durch Style Transfer kann der visuelle Stil eines Bildes auf ein anderes übertragen werden, wobei ein vortrainiertes neuronales Netz die Merkmale des Stils und Inhalts erfasst. Dies ermöglicht künstlerische Effekte und Anpassungen in der Bildbearbeitung und eröffnet neue Möglichkeiten für die kreative Gestaltung von visuellen Inhalten.

Wie im dargestellten Bild zu sehen ist, wird das ursprüngliche Bild mithilfe des Stils des Gemäldes übertragen, wobei die Konturen des Hundes weiterhin erkennbar bleiben.



Abbildung 4.7: [https://www.tensorflow.org/tutorials/generative/style\\_transfer](https://www.tensorflow.org/tutorials/generative/style_transfer)

GANs können auch verwendet werden, um Bilder mit höherer Auflösung oder besserer Qualität zu generieren, indem sie niedrig aufgelöste Bilder als Eingabe verwenden. GANs werden immer öfters in Bildverarbeitungstools eingesetzt um fehlende Bereiche zu füllen, oder zum erweitern von Bildern (siehe 4.3.3)<sup>26</sup>.

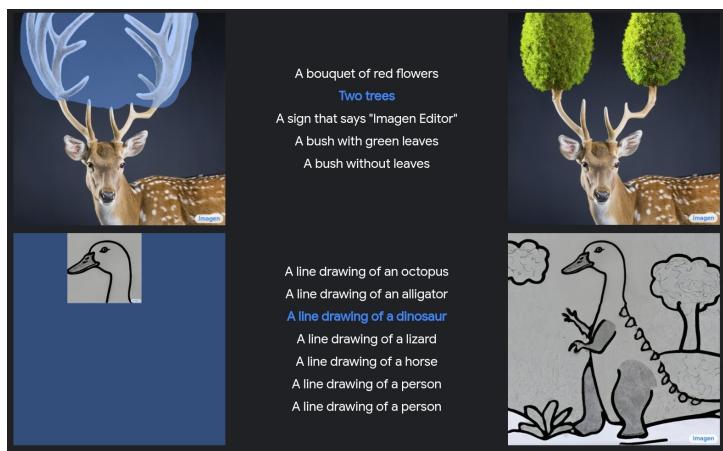


Abbildung 4.8: Füllen von markierten Bereichen eines Bildes durch Verwendung von Imagen und Erweitern von Bildern mithilfe von Imagen, Google.

<sup>26</sup><https://imagen.research.google/>

In der Forschung können GANs verwendet werden, um Videosequenzen zu generieren oder zu verbessern. Ein modernes Beispiel wäre Facebooks Errungenschaften durch Make-A-Video<sup>27</sup>, siehe 4.9.



Abbildung 4.9: Generierung kurzer Videos, anhand von einfachen textuellen Beschreibungen.

Da wir leider nicht in Hogwarts leben<sup>28</sup>, können Videos *nicht* auf PDFs angezeigt werden, noch weniger auf Papier. Die Eingaben für das Modell sind wie folgt: Oben Links: A dog wearing a superhero cape flying through the sky. Oben rechts: A spaceship landing on Mars. Unten links: An artist's brush painting on a canvas close up, highly detailed. Unten rechts: A horse drinking water.

<sup>27</sup><https://ai.facebook.com/blog/generative-ai-text-to-video/>

<sup>28</sup>Interaktive Zeitung aus Harry Potter: <https://youtu.be/xaBEFqFVSE8>

#### 4.3.4 Training von GANs und Evaluierung von generierten Ergebnissen

Das Training von GANs ist eine Herausforderung, da es sich um ein adversariales Lernverfahren handelt. Das bedeutet, dass es zwei Netze gibt, die sich gegenseitig trainieren und verbessern.

Das generative Netz versucht, Bilder zu erzeugen, die von einem diskriminierenden Netz nicht von echten Bildern unterschieden werden können. Das diskriminierende Netz wird trainiert, um echte Bilder von den vom generativen Netz generierten Bildern zu unterscheiden. Das Training von GANs erfolgt durch die Minimierung einer Verlustfunktion, die als GAN-Verlust bezeichnet wird.

Der GAN-Verlust besteht aus zwei Komponenten: dem Verlust des generativen Netzes und dem Verlust des diskriminierenden Netzes. Der Verlust des generativen Netzes wird minimiert, wenn das Netz Bilder erzeugt, die vom diskriminierenden Netz nicht als gefälscht erkannt werden. Der Verlust des diskriminierenden Netzes wird minimiert, wenn das Netz in der Lage ist, besonders zuverlässig und schnell zwischen echten und generierten Bildern zu unterscheiden. Die Evaluierung von generierten Ergebnissen ist eine wichtige Aufgabe bei der Arbeit mit GANs.

Es gibt verschiedene Methoden zur Bewertung von GANs, wie beispielsweise die visuelle Bewertung, die qualitative Bewertung und die quantitative Bewertung. Die visuelle Bewertung beinhaltet das Betrachten der generierten Bilder, um zu beurteilen, ob sie realistisch aussehen oder nicht. Die qualitative Bewertung beinhaltet die Verwendung von Bewertungsskalen, um die Qualität der generierten Bilder zu bewerten. Die quantitative Bewertung beinhaltet die Verwendung von Metriken wie der Inception Score oder der Frechet Inception Distance, um die Qualität der generierten Bilder zu bewerten.

#### 4.3.5 Ethische und soziale Implikationen von GANs

*There are several ethical challenges facing text-to-image research broadly. (...) First, downstream applications of text-to-image models are varied and may impact society in complex ways.*

<https://Imagen.research.google/>

Obwohl GANs eine vielversprechende Technologie sind, gibt es auch ethische und soziale Implikationen, die berücksichtigt werden müssen. Ein Problem bei der Verwendung von GANs ist, dass sie zur Erzeugung gefälschter Bilder oder Videos verwendet werden können. Dies kann zu Fälschungen und Manipulationen führen, die negative Auswirkungen auf die Gesellschaft haben können. Auch Rufschädigung kann durch GANs erleichtert werden. Ein weiteres Problem bei der Verwendung von GANs ist, dass sie möglicherweise nicht fair sind.



Abbildung 4.10: Beispiel für Vorurteile in GANs [ADAMS 2023]

GANs können aufgrund ihrer Lernmethode unbewusste Vorurteile aufnehmen und in ihren generierten Ergebnissen widerspiegeln (siehe Bild 4.10). Dies kann zu diskriminierenden Ergebnissen führen, die unfaire Entscheidungen unterstützen. Es ist wichtig,

dass bei der Verwendung von GANs Ethik und soziale Verantwortung berücksichtigt werden. Es sollten Maßnahmen ergriffen werden, um sicherzustellen, dass GANs fair und ethisch korrekt arbeiten. Zum Beispiel können spezielle Algorithmen entwickelt werden, um unbewusste Vorurteile zu minimieren. Weiterhin können Regierungsbehörden und andere Organisationen Maßnahmen ergreifen, um den Missbrauch von GANs zu verhindern. Das finden eines Kompromiss aus Forschung und politischer Einschränkung übersteigt jedoch den Rahmen dieser Arbeit.

## 4.4 Multiskalenskalierung

In vielen Anwendungen der Bildverarbeitung und Computergrafik ist es notwendig, Objekte oder Strukturen in verschiedenen Größenordnungen zu analysieren oder darzustellen. Multiscale-Skalierung bezeichnet Techniken und Vorgehen, die es ermöglichen, Objekte oder Signale auf verschiedenen Skalen zu untersuchen oder zu manipulieren. Dabei können sowohl lokale als auch globale Eigenschaften eines Objekts berücksichtigt werden. Diese Möglichkeiten komplementieren häufig Implementierungen von künstlichen Intelligenzen

*Machine learning and multiscale modeling mutually complement one another.<sup>29</sup>*

### 4.4.1 Grundlagen von Multiskalenskalierung

Multiscale-Skalierung ist ein Konzept aus der Signal- und Bildverarbeitung, das auf der Idee basiert, dass Signale und Bilder aus verschiedenen Skalen von Strukturen aufgebaut sind. In der Praxis bedeutet dies, dass ein Signal oder Bild auf verschiedene Skalen abgetastet oder transformiert wird, um Informationen auf verschiedenen Größenskalen zu erhalten. Ein grundlegendes Konzept der Multiscale-Skalierung ist die Skaleninvarianz. Das bedeutet, dass die Informationen in einem Signal oder Bild unabhängig von der

<sup>29</sup>Mark ALBER u. a. [2019]. »Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences«. In: *NPJ digital medicine* 2.1, S. 115.

Skala erhalten bleiben sollten. Das heißt, dass dieselben Merkmale oder Strukturen auf verschiedenen Skalen erkennbar sein sollten.<sup>30</sup>

#### 4.4.2 Methoden zur Multiskalenanalyse

Es gibt verschiedene Techniken zur Multiskalenanalyse, darunter Wavelet-Transformation<sup>31</sup> und pyramidenartige<sup>32</sup> Strukturen. Wavelets basieren auf der Zerlegung von Signalen in unterschiedliche Frequenzbänder, wodurch eine Multiskalenanalyse ermöglicht wird. Eine Wavelet-Transformation kann auf ein Signal angewendet werden, um es in seine Hoch- und Niederfrequenzkomponenten zu zerlegen. Diese Komponenten können dann unabhängig voneinander verarbeitet werden, um eine Analyse auf verschiedenen Skalen durchzuführen. Pyramiden sind eine weitere Methode zur Multiskalenanalyse, die auf der Idee der rekursiven Unterteilung von Signalen in immer feinere Skalen basiert. Dabei wird ein Signal in eine Pyramide aus verschiedenen Ebenen unterteilt, wobei jede Ebene eine andere Skala darstellt. Die unterste Ebene enthält das Originalsignal, während die oberen Ebenen eine immer gröbere Approximation des Signals enthalten.<sup>33</sup>

#### 4.4.3 Anwendungen von Multiscale-Skalierung

Multiscale-Skalierung hat viele Anwendungen in der Bildverarbeitung und Computer Vision. Ein Beispiel ist die Texturanalyse, bei der Texturen auf verschiedenen Skalen analysiert werden, um Muster und Strukturen zu identifizieren. Multiscale-Skalierung wird auch in der Bildkompression verwendet, um Bilder auf verschiedene Auflösungen zu

---

<sup>30</sup>Gao HUANG u. a. [2017]. »Densely connected convolutional networks«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 4700–4708; Stephane G MALLAT [1989]. »A theory for multiresolution signal decomposition: the wavelet representation«. In: *IEEE transactions on pattern analysis and machine intelligence* 11.7, S. 674–693.

<sup>31</sup>Jean-Luc STARCK, Fionn D MURTAGH und Albert BIJAOUI [1998]. *Image processing and data analysis: the multiscale approach*. Cambridge University Press.

<sup>32</sup>David I SHUMAN, Mohammad Javad FARAJI und Pierre VANDERGHEYNST [2015]. »A multiscale pyramid transform for graph signals«. In: *IEEE Transactions on Signal Processing* 64.8, S. 2119–2134.

<sup>33</sup>Eero P SIMONCELLI und Edward H ADELSON [1996]. »Noise removal via Bayesian wavelet coring«. In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Bd. 1. IEEE, S. 379–382.

skalieren und so Speicherplatz zu sparen. In jüngerer Zeit wurde die Multiskalenanalyse auch in Verbindung mit Deep Learning eingesetzt, um Modelle zu entwickeln, die auf verschiedenen Skalen arbeiten können. Ein Beispiel ist das Multiscale Dense Network Multiscale Dense Network (MSDN), das eine skalierbare Architektur für die Bilderkennung bietet, die auf mehreren Skalen arbeiten kann.

#### **4.4.4 Limitierungen und zukünftige Forschungsziele von Multiscale-Skalierung**

Obwohl die Multiskalenanalyse in vielen Bereichen der Bildverarbeitung und Computer Vision erfolgreich eingesetzt wurde, gibt es auch einige Limitationen. Eines der Hauptprobleme ist die Herausforderung, die richtige Skala für eine bestimmte Aufgabe zu wählen.

In einigen Fällen kann dies schwierig sein, da verschiedene Skalen unterschiedliche Informationen enthalten. Eine weitere Herausforderung besteht darin, die Multiskalenanalyse mit Deep Learning-Methoden zu integrieren. Während einige Fortschritte in diesem Bereich gemacht wurden, gibt es immer noch Raum für Verbesserungen, um die Multiskalenanalyse nahtlos in Deep Learning-Architekturen zu integrieren<sup>34</sup>.

Zukünftige Forschungsrichtungen könnten sich auf die Entwicklung von verbesserten Methoden zur Skalierung und Multiskalenanalyse konzentrieren, die eine höhere Genauigkeit und Effizienz ermöglichen. Darüber hinaus könnten Forscher daran arbeiten, die Integration der Multiskalenanalyse in Deep Learning-Architekturen weiter zu verbessern, um noch bessere Ergebnisse zu erzielen.

---

<sup>34</sup>Kaiming HE u. a. [2016]. »Deep residual learning for image recognition«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 770–778.

## 4.5 Vor- und Nachteile der fortgeschrittenen Methoden

Im Hinblick auf die Bildqualität sind Deep-Learning-basierte Methoden wie Super Resolution und GANs den traditionellen Methoden wie der Bilinear-Interpolation überlegen. Diese Methoden können hochwertige Bilder mit hoher Auflösung und Details erzeugen, die den Originalbildern nahe kommen. Insbesondere die GANs erlauben die Erzeugung von realistisch aussehenden Bildern, die schwer von echten Bildern zu unterscheiden sind. Ein weiterer Vorteil der fortgeschrittenen Methoden ist ihre Fähigkeit zur Generierung von neuen Inhalten. So können GANs und Style Transfer genutzt werden, um neue Bilder zu erzeugen, die auf den Stil und Inhalt anderer Bilder basieren. Dies bietet neue kreative Möglichkeiten in der Kunst und Design.

Jedoch stellen diese fortgeschrittene Methoden auch Herausforderungen bei der Anwendung dar. Beispielsweise benötigen Deep-Learning-Methoden große Datensätze und Rechenleistung, um optimal zu funktionieren<sup>35</sup>. Ein weiteres Problem ist die Interpretierbarkeit<sup>36</sup> der Ergebnisse, da es schwierig sein kann, zu verstehen, wie ein Modell zu seinen Ergebnissen gekommen ist. Die Anwendung von fortgeschrittenen Skalierungsmethoden hat auch Auswirkungen auf die Leistung und Effizienz von Systemen<sup>37</sup>. So können Deep-Learning-basierte Methoden in Echtzeit-Anwendungen, wie beispielsweise autonomen Fahrzeugen, zu Verzögerungen führen. Es ist daher wichtig, dass die Anforderungen an Leistung und Effizienz bei der Auswahl der Skalierungsmethode berücksichtigt werden.

Es gibt moderne Architekturen und Techniken für höchst effiziente und leistungsstarke Ergebnisse. Zukunftsaussichten für fortgeschrittene Skalierungsmethoden liegen in der

---

<sup>35</sup>Juncheng LI, Zehua PEI und Tieyong ZENG [2021]. *From Beginner to Master: A Survey for Deep Learning-based Single-Image Super-Resolution*. arXiv: 2109.14335 [eess.IV].

<sup>36</sup>Sheng HE u. a. [2023]. »Segmentation ability map: Interpret deep features for medical image segmentation«. In: *Medical Image Analysis* 84, S. 102726.

<sup>37</sup>Hongda WANG u. a. [Dez. 2018]. »Deep learning enables cross-modality super-resolution in fluorescence microscopy«. In: *Nature Methods* 16.1, S. 103–110. DOI: 10.1038/s41592-018-0239-0. URL: <https://doi.org/10.1038/s41592-018-0239-0>.

Kombination verschiedener Methoden. So können beispielsweise GANs und Super Resolution zusammen genutzt werden, um qualitativ hochwertige und realistisch aussehende Bilder zu erzeugen. Ein weiteres Forschungsgebiet ist die Verbesserung der Interpretierbarkeit und Erklärbarkeit von Deep-Learning-Modellen.

Insgesamt bieten fortgeschrittene Skalierungsmethoden in der Bildverarbeitung und Computergrafik viele Vorteile, jedoch müssen auch die Herausforderungen bei der Anwendung und die Auswirkungen auf die Leistung und Effizienz von Systemen berücksichtigt werden<sup>38</sup>. Zukünftige Forschung sollte darauf abzielen, die verschiedenen Methoden zu kombinieren und die Interpretierbarkeit von Deep-Learning-Modellen zu verbessern.

---

<sup>38</sup>Christina KLÜVER und Jürgen KLÜVER [2022]. »Chancen und Herausforderungen beim Einsatz neuronaler Netzwerke als Methoden der Künstlichen Intelligenz oder des Maschinellen Lernens in KMU«. In: *Digitalisierung und Nachhaltigkeit – Transformation von Geschäftsmodellen und Unternehmenspraxis*. Springer Berlin Heidelberg, S. 121–148. DOI: 10.1007/978-3-662-65509-2\_8. URL: [https://doi.org/10.1007/978-3-662-65509-2\\_8](https://doi.org/10.1007/978-3-662-65509-2_8).

# Kapitel 5

## Evaluation von Skalierungsmethoden

### 5.1 Qualitätsmetriken von Skalierungsmethoden

Die Evaluation von Skalierungsmethoden in der Bildverarbeitung bedarf einer facettenreichen Palette an Qualitätsmetriken, welche eine präzise Analyse und Vergleichbarkeit unterschiedlicher Methoden ermöglichen. Im Rahmen dieser Arbeit werden ausgewählte sowie zentrale Qualitätsmetriken für die Bewertung von Skalierungsmethoden erörtert und diskutiert.

#### 5.1.1 Peak Signal-to-Noise Ratio (PSNR)

Das Peak Signal-to-Noise Ratio PSNR ist eine der am häufigsten verwendeten Metriken zur Bewertung der Qualität von Bildern. Es misst die Qualität einer Bildrekonstruktion, indem es den Unterschied zwischen einem Originalbild und einem rekonstruierten Bild berechnet und diesen Unterschied durch das maximale Signal (Peak) und das Rauschen (Noise) im Originalbild teilt. Je höher der PSNR-Wert, desto geringer ist der Unterschied zwischen Original- und Rekonstruktionsbildern und desto besser ist die Qualität der Rekonstruktion.<sup>1</sup>

---

<sup>1</sup>Zhou WANG u. a. [2004a]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE transactions on image processing* 13.4, S. 600–612.

## Definition und Berechnung von PSNR

Die Definition von PSNR lautet wie folgt:

$$PSNR = 20 \log_{10} \left( \frac{MAX_I}{\sqrt{MSE}} \right) \quad (5.1)$$

Dabei ist  $MAX_I$  der maximal mögliche Pixelwert des Bildes. Bei der Darstellung von Pixeln mit 8 Bits pro Abtastwert ist dieser Wert 255. Allgemeiner ausgedrückt: Wenn die Samples mit linearer PCM mit B Bits pro Sample dargestellt werden, ist  $MAX_I 2^B - I^{22}$ .

$$MSE = \frac{1}{n * m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (I(i, j) - K(i, j))^2 \quad (5.2)$$

Der Mean Squared Error Mean Squared Error (MSE) ist definiert als der Durchschnitt der quadrierten Unterschiede zwischen jedem Pixel des Originalbildes und dem rekonstruierten Bild. Je höher der PSNR-Wert, desto geringer ist der MSE und desto besser ist die Qualität der Rekonstruktion.

## Anwendung von PSNR bei der Bewertung von Bildqualität

Obwohl PSNR eine weit verbreitete Methode zur Bewertung der Qualität von Bildern ist, hat sie auch ihre Limitationen<sup>3</sup>. Zum einen berücksichtigt sie nur die Fehler zwischen Original- und Rekonstruktionsbildern und vernachlässigt andere Faktoren wie Bildverzerrungen, die durch Komprimierung oder Filterung entstehen können. Zum andern ist der PSNR-Wert nicht sensitiv für menschlich wahrgenommene Verzerrungen<sup>4</sup>, wie z.B. Farbverschiebungen oder Artefakte. Trotz dieser Limitationen bleibt PSNR eine wichtige Metrik in der Bildverarbeitung und wird oft in der Praxis verwendet, um die Qualität

---

<sup>2</sup>WIKIPEDIA [2023a]. *Peak signal-to-noise ratio — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-May-2023]. URL: <http://en.wikipedia.org/w/index.php?title=Peak%20signal-to-noise%20ratio&oldid=1145027401>.

<sup>3</sup>Hamid R SHEIKH und Alan C BOVIK [2006]. »Image information and visual quality«. In: *IEEE Transactions on image processing* 15.2, S. 430–444.

<sup>4</sup>Anish MITTAL, Anush Krishna MOORTHY und Alan Conrad BOVIK [2012]. »No-reference image quality assessment in the spatial domain«. In: *IEEE Transactions on image processing* 21.12, S. 4695–4708.

von Bildrekonstruktionen zu bewerten und zu vergleichen.<sup>5</sup>

### 5.1.2 Structural Similarity Index Measure (SSIM)

Der Structural Similarity Index SSIM ist eine Metrik zur Bewertung der strukturellen Ähnlichkeit zwischen einem Originalbild und einem rekonstruierten Bild. Im Gegensatz zum PSNR berücksichtigt der SSIM nicht nur die Pixelwerte, sondern auch die Struktur und Textur des Bildes. Der SSIM berechnet die Ähnlichkeit zwischen den beiden Bildern anhand von drei Faktoren: Helligkeit, Kontrast und Struktur. Die Formel zur Berechnung von SSIM ist wie folgt:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (5.3)$$

In dieser Formel repräsentieren  $x$  und  $y$  zwei verglichene Bilder. Die SSIM misst die strukturelle Ähnlichkeit zwischen diesen Bildern.  $\mu_x$  und  $\mu_y$  sind die Mittelwerte von  $x$  und  $y$ , während  $\sigma_x^2$  und  $\sigma_y^2$  die Varianzen von  $x$  und  $y$  darstellen.  $\sigma_{xy}$  repräsentiert die Kovarianz zwischen  $x$  und  $y$ . Die Konstanten  $c_1$  und  $c_2$  sind kleine Werte, die zur Stabilisierung der Division hinzugefügt werden<sup>6,7</sup>.

#### Anwendung von SSIM bei der Bewertung von Bildqualität

SSIM wird häufig verwendet, um die Qualität von Bildrekonstruktionen zu bewerten. Es hat sich gezeigt, dass SSIM besser als PSNR die wahrgenommene Bildqualität wiederspiegelt. Dies liegt daran, dass SSIM die strukturelle Ähnlichkeit zwischen den beiden Bildern berücksichtigt, während PSNR nur die Differenz der Pixelweite betrachtet.

---

<sup>5</sup>Jari KORHONEN und Junyong YOU [2012]. »Peak signal-to-noise ratio revisited: Is simple beautiful?« In: *2012 Fourth International Workshop on Quality of Multimedia Experience*. IEEE, S. 37–38.

<sup>6</sup>Ming-Jun CHEN und Alan C BOVIK [2011]. »Fast structural similarity index algorithm«. In: *Journal of Real-Time Image Processing* 6, S. 281–287.

<sup>7</sup>WIKIPEDIA [2023b]. *Structural similarity — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-May-2023]. URL: <http://en.wikipedia.org/w/index.php?title=Structural%20similarity&oldid=1142355078>.

### Vorteile von SSIM im Vergleich zu PSNR

Im Vergleich zum PSNR hat SSIM mehrere Vorteile. Zum einen berücksichtigt es die strukturelle Ähnlichkeit zwischen den beiden Bildern, was zu einer besseren Bewertung der wahrgenommenen Bildqualität führt. Zum anderen ist SSIM in der Lage, Verzerrungen zu erkennen, die durch Kompression oder andere Arten von Bildverarbeitung verursacht werden, während PSNR dies nicht tut.<sup>8</sup>

### Limitationen von SSIM

Obwohl SSIM eine bessere Metrik zur Bewertung der Bildqualität als PSNR darstellt, hat es auch seine Limitationen. SSIM ist anfällig für Helligkeits- und Kontrastunterschiede zwischen den beiden Bildern und kann bei der Bewertung von stark komprimierten Bildern ungenau sein. Auch bei der Anwendung auf Bilder mit unterschiedlichen Strukturen kann die SSIM eine ungenaue Bewertung liefern.<sup>9</sup>

#### 5.1.3 Mean Opinion Score

Der Mean Opinion Score (MOS) ist eine subjektive Metrik, die die wahrgenommene Qualität einer Bildrekonstruktion misst. MOS basiert auf der Bewertung durch menschliche Beobachter, die gebeten werden, die Qualität von Original- und Rekonstruktionsbildern auf einer Skala von 1 bis 5 oder 1 bis 10 zu bewerten. MOS ist eine wichtige Metrik, da sie die subjektive Wahrnehmung der Qualität eines Bildes durch den Betrachter berücksichtigt.<sup>10</sup>

<sup>8</sup>Alain HORÉ und Djemel ZIOU [2010]. »Image Quality Metrics: PSNR vs. SSIM«. In: *2010 20th International Conference on Pattern Recognition*, S. 2366–2369. doi: 10.1109/ICPR.2010.579.

<sup>9</sup>Zhou WANG u. a. [2004b]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE Transactions on Image Processing* 13.4, S. 600–612. ISSN: 1941-0042. doi: 10.1109/TIP.2003.819861.

<sup>10</sup>Hamid R SHEIKH und Alan C BOVIK [2006]. »Image information and visual quality«. In: *IEEE Transactions on image processing* 15.2, S. 430–444.

### 5.1.4 Peak Signal-to-Noise Ratio der Y-Komponente (PSNR-Y)

Die Y-Komponente im YCbCr-Farbraum enthält die Helligkeitsinformationen des Bildes. Das Peak Signal-to-Noise Ratio der Y-Komponente<sup>11</sup> Peak Signal-to-Noise Ratio Y (PSNR-Y) ist eine spezielle Version des PSNR, die nur die Helligkeitsinformationen des Originalbildes und des rekonstruierten Bildes berücksichtigt. PSNR-Y ist eine wichtige Metrik zur Bewertung der Qualität von Skalierungsmethoden für Graustufen- oder Schwarz-Weiß-Bilder. Diese Qualitätsmetriken sind wichtige Werkzeuge für die Bewertung und den Vergleich von Skalierungsmethoden in der Bildverarbeitung. Es ist jedoch wichtig zu beachten, dass keine einzelne Metrik alle Aspekte der Bildqualität abdeckt. Eine umfassende Bewertung sollte mehrere Metriken kombinieren und auch die subjektive Wahrnehmung<sup>12</sup>

### 5.1.5 Computational Speed

Die Wahl einer Skalierungsmethode hängt nicht nur von der Bildqualität, sondern auch von der benötigten Rechenleistung ab. Die Computational Speed ist daher ein wichtiger Faktor bei der Auswahl einer Skalierungsmethode. Es gibt verschiedene Methoden zur Messung von Computational Speed, wie z.B. die Messung der benötigten Zeit, um ein Bild zu skalieren oder die Berechnung von Floating Point Operations Per Second (FLOPS). Eine genauere Messung kann durch die Verwendung von Benchmarks erreicht werden, die es ermöglichen, verschiedene Skalierungsmethoden auf derselben Hardware zu vergleichen<sup>13</sup>. Jedoch muss bei der Messung der Computational Speed berücksichtigt werden, dass die Leistung des verwendeten Computers die Ergebnisse beeinflussen kann. Ein schnellerer Computer kann eine Methode schneller ausführen als ein langsamerer Computer. Daher

---

<sup>11</sup>Xiwu SHANG u. a. [2018]. »Color-sensitivity-based combined PSNR for objective video quality assessment«. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.5, S. 1239–1250.

<sup>12</sup>Q. HUANG, L. YU und S. WANG [2010]. »A New Image Quality Assessment Method for Peak Signal-to-Noise Ratio (PSNR) in YUV Color Space«. In: *IEEE Transactions on Consumer Electronics* 56.4, S. 2317–2322. DOI: 10.1109/TCE.2010.5682097.

<sup>13</sup>Danny HERNANDEZ und Tom B BROWN [2020]. »Measuring the algorithmic efficiency of neural networks«. In: *arXiv preprint arXiv:2005.04305*.

ist es wichtig, die Messungen auf einem vergleichbaren Computer durchzuführen und die Ergebnisse zu normalisieren. Ein wichtiger Trade-off besteht zwischen der Bildqualität und der Computational Speed. Eine Methode, die eine höhere Bildqualität liefert, benötigt in der Regel mehr Rechenleistung und ist daher langsamer als eine Methode mit niedrigerer Bildqualität. Es ist daher wichtig, die gewünschte Bildqualität und die verfügbare Rechenleistung abzuwägen und eine Methode zu wählen, die den Anforderungen am besten entspricht. Es können auch Techniken wie progressiver Skalierung verwendet werden, um eine gute Bildqualität mit einer angemessenen Geschwindigkeit zu erreichen.<sup>14,15</sup>

### 5.1.6 Root-Mean-Square Error

Der Root Mean Squared Error (RMSE) ist eine Metrik zur Bewertung der Qualität von Bildrekonstruktionen. Ein Nachteil von RMSE ist, dass er nur den Fehler zwischen Pixelwerten betrachtet und keine Berücksichtigung von strukturellen Unterschieden im Bild nimmt. Daher wird RMSE oft in Kombination mit anderen Metriken wie PSNR und SSIM verwendet, um ein umfassenderes Bild der Bildqualität zu erhalten. Abschließend ist zu sagen, dass beider Bewertung von Bildqualität die Wahl der Metrik von der spezifischen Anwendung abhängt. Während PSNR und SSIM für die Bewertung von Bildern in vielen Anwendungen ausreichend sein können, kann in anderen Fällen die subjektive Wahrnehmung des menschlichen Betrachters durch MOS eine bessere Metrik sein. Darüber hinaus ist bei der Wahl einer Skalierungsmethode auch die Messung von Computational Speed und das Abwägen von Trade-offs zwischen Bildqualität und Geschwindigkeit von entscheidender Bedeutung.<sup>16</sup>

---

<sup>14</sup>Jiajun XU, Xudong JIANG und Zulin ZHANG [2017]. »Efficient and accurate image super-resolution via recurrent mixture density network«. In: *IEEE Transactions on Image Processing* 26.7, S. 3187–3201. DOI: 10.1109/TIP.2017.2699923.

<sup>15</sup>Min Joon CHOI, Sang Heon LEE und Sang Jun LEE [2019]. »Lightweight and Efficient Deep Neural Network for Super-Resolution with Mobile Devices«. In: *Electronics* 8.4, S. 423. DOI: 10.3390/electronics8040423.

<sup>16</sup>M MORAD, AI CHALMERS und PR O'REGAN [1996]. »The role of root-mean-square error in the geo-transformation of images in GIS«. In: *International Journal of Geographical Information Science* 10.3,

## 5.2 Kriterien zur Auswahl der Skalierungsmethode

### 5.2.1 Bildvergleich und visuelle Bewertung

Die Evaluation von Skalierungsmethoden in der Bildverarbeitung erfordert eine umfassende und präzise Analyse verschiedener Kriterien, um die optimale Methode auszuwählen. Ein wesentlicher Aspekt bei dieser Bewertung ist der Bildvergleich und die visuelle Bewertung, die auf der subjektiven Wahrnehmung von menschlichen Beobachtern beruhen (siehe sec5.1.3). Die visuelle Bewertung von Bildern durch menschliche Beobachter ermöglicht eine direkte und subjektive Beurteilung der Bildqualität basierend auf wahrgenommenen visuellen Eigenschaften. Verschiedene Methoden des Bildvergleichs werden eingesetzt, wie beispielsweise der Side-by-Side-Vergleich, bei dem zwei Bilder direkt miteinander verglichen werden, oder der Triple-Stimulus-Test, bei dem ein Originalbild mit zwei rekonstruierten Bildern verglichen wird. Diese Methoden des Bildvergleichs und der visuellen Bewertung werden auch bei der Evaluierung von Skalierungsmethoden angewendet. Durch den Vergleich der rekonstruierten Bilder mit dem Originalbild können verschiedene Skalierungsmethoden anhand ihrer visuellen Qualität beurteilt und verglichen werden. dies ermöglicht eine umfangreiche Analyse und eine direkte Gegenüberstellung der unterschiedlichen Methoden. Es ist jedoch von entscheidender Bedeutung, die Limitationen und Vorbehalte im Zusammenhang mit der visuellen Bewertung von Bildern zu berücksichtigen. Die subjektive Wahrnehmung kann von Person zu Person variieren und wird auch von anderen Faktoren wie Beleuchtung, Betrachtungsabstand und individuellen Präferenzen beeinflusst<sup>17</sup>. Darüber hinaus können komplexe visuelle Verzerrungen nicht immer präzise durch die visuelle Bewertung erfasst werden. Daher sollten bei der Bewertung von Skalierungsmethoden auch objektive Metriken und weitere Evaluierungskriterien einbezogen werden. Insgesamt spielt der Bildvergleich und die visuelle Bewertung eine bedeutende Rolle bei der Evaluierung von Skalierungsmethoden in der Bildverarbeitung.

---

S. 347–353.

<sup>17</sup> Robert C STREIJL, Stefan WINKLER und David S HANDS [2016]. »Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives«. In: *Multimedia Systems* 22.2, S. 213–227.

Durch die Kombination der subjektiven visuellen Bewertung mit objektiven Metriken können fundierte Entscheidungen getroffen werden, um die optimale Methode gemäß den spezifischen Anforderungen der Anwendung auszuwählen. Die sorgfältige Berücksichtigung der genannten Limitationen trägt dazu bei, zuverlässige und aussagekräftige Ergebnisse zu erzielen und die Qualität von Bildrekonstruktionen bestmöglich zu bewerten.<sup>18,19,20</sup>

### 5.2.2 Effektivität und Effizienz

Die Bewertung von Skalierungsmethoden in der Bildverarbeitung erfordert eine umfassende Betrachtung sowohl der Effektivität, also der Bildqualität, als auch der Effizienz, insbesondere der Computational Speed. Es besteht ein inhärenter Trade-off zwischen der erzielten Bildqualität und der benötigten Rechenleistung, der bei der Auswahl der optimalen Skalierungsmethode berücksichtigt werden muss. Die Effektivität einer Skalierungsmethode bezieht sich auf die erzielte Bildqualität der rekonstruierten Bilder. Eine Methode, die eine höhere Bildqualität liefert, wird in der Regel auch mehr Rechenleistung benötigen und somit langsamer sein als eine Methode mit geringerer Bildqualität. Daher ist es von großer Bedeutung, die gewünschte Bildqualität und die verfügbare Rechenleistung sorgfältig abzuwägen, um die beste Skalierungsmethode zu wählen.

Die Evaluierung der Trade-offs zwischen Effektivität und Effizienz erfordert eine gründliche Analyse der Leistungsmerkmale verschiedener Skalierungsmethoden. Hierbei können Kosten-Nutzen-Analysen hilfreich sein, um die Auswirkungen der gewählten Methode auf die Bildqualität und die erforderliche Rechenleistung abzuschätzen. Indem die Kosten in Bezug auf die erzielte Bildqualität bewertet werden, kann die optimale Methode entsprechend den spezifischen Anforderungen des Anwendungsbereichs ausgewählt werden. Die Relevanz von Effektivität und Effizienz variiert je nach Anwendungsbereich der Bildverar-

---

<sup>18</sup>Peng YE und David DOERMANN [2012]. »No-reference image quality assessment using visual codebooks«. In: *IEEE Transactions on Image Processing* 21.7, S. 3129–3138.

<sup>19</sup>Zhou WANG u. a. [2004a]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE transactions on image processing* 13.4, S. 600–612.

<sup>20</sup>Michael P ECKERT und Andrew P BRADLEY [1998]. »Perceptual quality metrics applied to still image compression«. In: *Signal processing* 70.3, S. 177–200.

| Model    | PLCC   | MAE    | RMS    | SRCC   | KRCC   | Computation time |
|----------|--------|--------|--------|--------|--------|------------------|
| PSNR     | 0.9062 | 7.4351 | 9.8191 | 0.8804 | 0.6886 | 1                |
| SSIM     | 0.9253 | 6.9203 | 8.8069 | 0.9014 | 0.7246 | 22.65            |
| MS-SSIM  | 0.8945 | 8.1969 | 10.384 | 0.8619 | 0.6605 | 48.49            |
| SSIMplus | 0.9732 | 4.3192 | 5.3451 | 0.9349 | 0.7888 | 7.83             |

Tabelle 5.1: Leistungsvergleich zwischen PSNR, SSIM, MS-SSIM, VQM, PQR-Tek, DMOS-Tek, JND-VC, DMOS-VC und SSIMplus einschließlich aller Geräte<sup>22</sup>.

beitung. In einigen Fällen, wie beispielsweise medizinischen Bildgebungsverfahren oder hochauflösenden Videoanwendungen, ist eine hohe Bildqualität von größter Bedeutung, selbst wenn dies mit einem höheren Rechenaufwand einhergeht. In anderen Szenarien, wie in Echtzeit-Anwendungen oder mobilen Geräten, kann die Effizienz und die schnelle Verarbeitung der Bilder priorisiert werden, auch wenn dies zu einer gewissen Einbuße der Bildqualität führt. Insgesamt ist es von entscheidender Bedeutung, sowohl die Effektivität als auch die Effizienz bei der Wahl der besten Skalierungsmethode zu berücksichtigen. Die richtige Balance zwischen Bildqualität und erforderlicher Rechenleistung zu finden, ermöglicht die optimale Nutzung von Ressourcen und die Erfüllung der spezifischen Anforderungen des jeweiligen Anwendungsbereichs der Bildverarbeitung.<sup>23,24,25</sup>

### 5.2.3 Zukunftsansichten und Herausforderungen

Die kontinuierliche Entwicklung von Technologien stellt neue Herausforderungen bei der Evaluierung von Skalierungsmethoden in der Bildverarbeitung dar. Fortschritte in der

<sup>23</sup>Y. ZHANG u. a. [2019]. »Efficiency-Driven Image Scaling Method Selection Based on Deep Neural Networks«. In: *IEEE Transactions on Image Processing* 28.1, S. 424–439. DOI: 10.1109/TIP.2018.2889740.

<sup>24</sup>K. GU u. a. [2019]. »A Comprehensive Study of Image Upsampling Quality Metrics«. In: *IEEE Transactions on Image Processing* 28.3, S. 1316–1331. DOI: 10.1109/TIP.2019.2906826.

<sup>25</sup>Y. HU u. a. [2020]. »An Efficient Edge-Guided Image Upsampling Method Based on Local Frequency Estimation«. In: *IEEE Transactions on Image Processing* 29, S. 430–442. DOI: 10.1109/TIP.2019.2896251.

Hardware, wie leistungsstärkere Prozessoren und spezialisierte Beschleuniger, können die Rechenleistung erhöhen und somit neue Möglichkeiten für komplexe Skalierungsalgorithmen eröffnen. Gleichzeitig führen neue Technologien wie Virtual Reality, Augmented Reality und 8K-Bildschirme zu höheren Anforderungen an die Bildqualität und erfordern präzisere Bewertungsmethoden. Spezifische Anwendungsbereiche, wie die medizinische Bildgebung oder die Videokompression, stellen weitere Herausforderungen bei der Evaluierung von Skalierungsmethoden dar. In der medizinischen Bildgebung ist die genaue Beurteilung der Bildqualität von entscheidender Bedeutung für die richtige Diagnosestellung und Behandlungsplanung. Hier müssen spezielle Metriken und Evaluierungsverfahren entwickelt werden, um den Anforderungen dieses Bereichs gerecht zu werden. Ebenso erfordert die Videokompression eine sorgfältige Evaluierung der Skalierungsmethoden, um eine ausreichende Qualität der komprimierten Videos zu gewährleisten und gleichzeitig die Effizienz der Kompression zu maximieren. Ein vielversprechendes Potenzial für die zukünftige Evaluierung von Skalierungsmethoden liegt in der Anwendung von KI-basierten Evaluierungsmethoden. Durch den Einsatz von maschinellem Lernen und neuronalen Netzwerken können automatisierte Bewertungsalgorithmen entwickelt werden, die die menschliche Wahrnehmung von Bildqualität besser simulieren. Diese KI-basierten Ansätze können eine schnellere und objektivere Evaluierung ermöglichen und den Entwicklungsprozess von Skalierungsmethoden beschleunigen. Bei der Bewertung von Skalierungsmethoden sollten jedoch auch ethische und soziale Implikationen berücksichtigt werden. Die Entwicklung von immer leistungsfähigeren Skalierungsmethoden kann auch potenzielle Risiken mit sich bringen, wie zum Beispiel die Manipulation von Bildern oder die Schaffung von gefälschten Inhalten. Es ist daher wichtig, entsprechende Schutzmechanismen zu entwickeln, um den Missbrauch solcher Technologien zu verhindern und die Privatsphäre und Integrität von Personen zu wahren. Zusammenfassend lassen sich die Zukunftsaussichten für die Evaluierung von Skalierungsmethoden als vielversprechend betrachten. Durch die Berücksichtigung technologischer Entwicklungen, die Bewältigung spezifischer Herausforderungen in verschiedenen Anwendungsbereichen, die Nutzung von KI-basierten Evaluierungsmethoden und die Beachtung ethischer und sozialer Implikationen können wir die Qualität

und Effizienz von Skalierungsmethoden weiter verbessern und gleichzeitig sicherstellen, dass sie verantwortungsvoll und zum Wohl der Gesellschaft eingesetzt werden.<sup>26,27,28</sup>

---

<sup>26</sup>Kai ZHANG u.a. [2015]. »Learning a single convolutional super-resolution network for multiple degradations«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 3262–3270. DOI: 10.1109/CVPR.2015.7298958.

<sup>27</sup>Yochai BLAU und Tomer MICHAELI [2018]. »The perception-distortion tradeoff«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 6228–6237. DOI: 10.1109/CVPR.2018.00651.

<sup>28</sup>Wei LIU u.a. [2016]. »SSD: Single shot multibox detector«. In: *European conference on computer vision*. Springer, S. 21–37. DOI: 10.1007/978-3-319-46448-0\_2; Richard ZHANG, Phillip ISOLA und Alexei A. EFROS [2018]. »Split-brain autoencoders: Unsupervised learning by cross-channel prediction«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 1058–1067. DOI: 10.1109/CVPR.2018.00120.

# Kapitel 6

## Einführung in Deep Learning

Deep Learning ist eine Unterkategorie des maschinellen Lernens und beschäftigt sich mit der Entwicklung und Anwendung von neuronalen Netzen mit mehreren Schichten, um komplexe Probleme zu lösen. Es verwendet künstliche Intelligenz, um aus großen Datenmengen Muster zu erkennen und Entscheidungen zu treffen. In der heutigen Welt hat Deep Learning Anwendungen in Bereichen wie Spracherkennung, Bilderkennung, Robotik, Medizin und Autonomen Fahrzeugen gefunden. Die Geschichte des Deep Learning reicht zurück bis in die 1940er Jahre, aber erst in den 2000er Jahren wurden die Algorithmen und Computerleistung ausreichend entwickelt, um Deep Learning erfolgreich anzuwenden. Im Jahr 2012 gewann ein Deep Learning-Modell namens AlexNet den ImageNet-Wettbewerb, was als Durchbruch für die Anwendung von Deep Learning in Computer Vision gilt. Traditionelles maschinelles Lernen verwendet in der Regel flache neuronale Netzwerke, die nur eine Schicht haben, um Daten zu analysieren. Im Gegensatz dazu verwenden Deep-Learning-Modelle mehrere Schichten, um komplexere Merkmale der Daten zu identifizieren. Dies ermöglicht eine höhere Genauigkeit und Effektivität bei der Lösung komplexer Probleme. Beispiele für reale Anwendungen von Deep Learning sind unter anderem die Bilderkennung in sozialen Medien, die Spracherkennung<sup>1</sup> in Smart Speakern wie Amazon Echo und Google Home, sowie die automatisierte Diagnose von medizinischen Bildern wie

---

<sup>1</sup>SPACERIDE [2023]. *So funktioniert Spracherkennung mit Deep Learning*. URL: <https://www.spaceride.de/blog/so-funktioniert-spracherkennung-mit-deep-learning>.

CT-Scans<sup>2</sup> und MRT-Bildern.

## 6.1 Motivation der Studie von CNNs und deren Training

Traditionelle maschinelle Lernalgorithmen sind nicht effektiv bei der Verarbeitung komplexer Datentypen wie Bildern. Hier kommen Convolutional Neural Networks (CNNs) ins Spiel. CNNs sind eine spezielle Art von neuronalem Netzwerk, die speziell für die Bilderkennung und Computer Vision entwickelt wurden. CNNs arbeiten durch die Verwendung von Filtern, die über das Eingabebild geschoben werden, um Merkmale wie Kanten, Farben und Texturen zu extrahieren. Diese Merkmale werden dann von Schichten von Neuronen verwendet, um die Merkmale zu kombinieren und eine Vorhersage zu treffen. Zum Beispiel können 2 Kantenfilter kombiniert werden um Eckenfilter zu bauen. Eine der größten Herausforderungen beim Training von CNNs ist das Overfitting, bei dem das Modell zu stark auf die Trainingsdaten optimiert wird und dadurch bei neuen Daten schlecht abschneidet. Eine weitere Herausforderung ist das Problem der verschwindenden Gradienten, bei dem die Gradienten, die zur Anpassung der Gewichte verwendet werden, während des Trainings immer kleiner werden und das Modell nicht mehr lernen kann. Um diese Probleme zu lösen, werden Optimierungsalgorithmen wie der stochastische Gradientenabstieg verwendet, um die Gewichte des Modells und das Modell an die Daten anzupassen. Es gibt auch Techniken wie Dropout und Data Augmentation (siehe sec 8.4), die dazu beitragen können, Overfitting und Underfitting zu minimieren. Es ist wichtig, CNNs zu verstehen und effektiv zu trainieren, um die volle Leistungsfähigkeit von Deep Learning in der Bilderkennung und Computer Vision zu nutzen.

---

<sup>2</sup>Vruddhi SHAH u. a. [2021]. »Diagnosis of COVID-19 using CT scan images and deep learning techniques«. In: *Emergency radiology* 28, S. 497–505.

## 6.2 Struktur der Arbeit

Dieser Teil der Studienarbeit ist in zwei Hauptabschnitte unterteilt. Im ersten Abschnitt wird eine Einführung in Deep Learning gegeben, einschließlich der Bedeutung von Deep Learning in der heutigen Welt, der Geschichte und Entwicklung des Deep Learning, der Unterschiede zwischen Deep Learning und traditionellem maschinellem Lernen und Beispielen für reale Anwendungen von Deep Learning.

Im zweiten Abschnitt wird die Motivation hinter der Studie von CNNs und deren Training erläutert. Dabei werden die Einschränkungen von traditionellen maschinellen Lernalgorithmen bei der Verarbeitung komplexer Datentypen wie Bildern diskutiert und die Bedeutung von CNNs bei der Bilderkennung und Computer Vision erklärt. Weiterhin werden die Herausforderungen beim Training von CNNs wie Overfitting und verschwindende Gradienten sowie die Rolle von Optimierungsalgorithmen wie dem stochastischen Gradientenabstieg beim Training von CNNs beschrieben.

Dieser Teil der Studienarbeit ist in zwei Hauptabschnitte unterteilt. Im ersten Abschnitt wird eine Einführung in Deep Learning gegeben, einschließlich der Bedeutung von Deep Learning in der heutigen Welt, der Geschichte und Entwicklung des Deep Learning, der Unterschiede zwischen Deep Learning und traditionellem maschinellem Lernen und Beispielen für reale Anwendungen von Deep Learning.

Im zweiten Abschnitt wird die Motivation hinter der Studie von CNNs und deren Training erläutert. Dabei werden die Einschränkungen von traditionellen maschinellen Lernalgorithmen bei der Verarbeitung komplexer Datentypen wie Bildern diskutiert und die Bedeutung von CNNs bei der Bilderkennung und Computer Vision erklärt. Weiterhin werden die Herausforderungen beim Training von CNNs wie Overfitting und verschwindende Gradienten sowie die Rolle von Optimierungsalgorithmen wie dem stochastischen Gradientenabstieg beim Training von CNNs beschrieben.

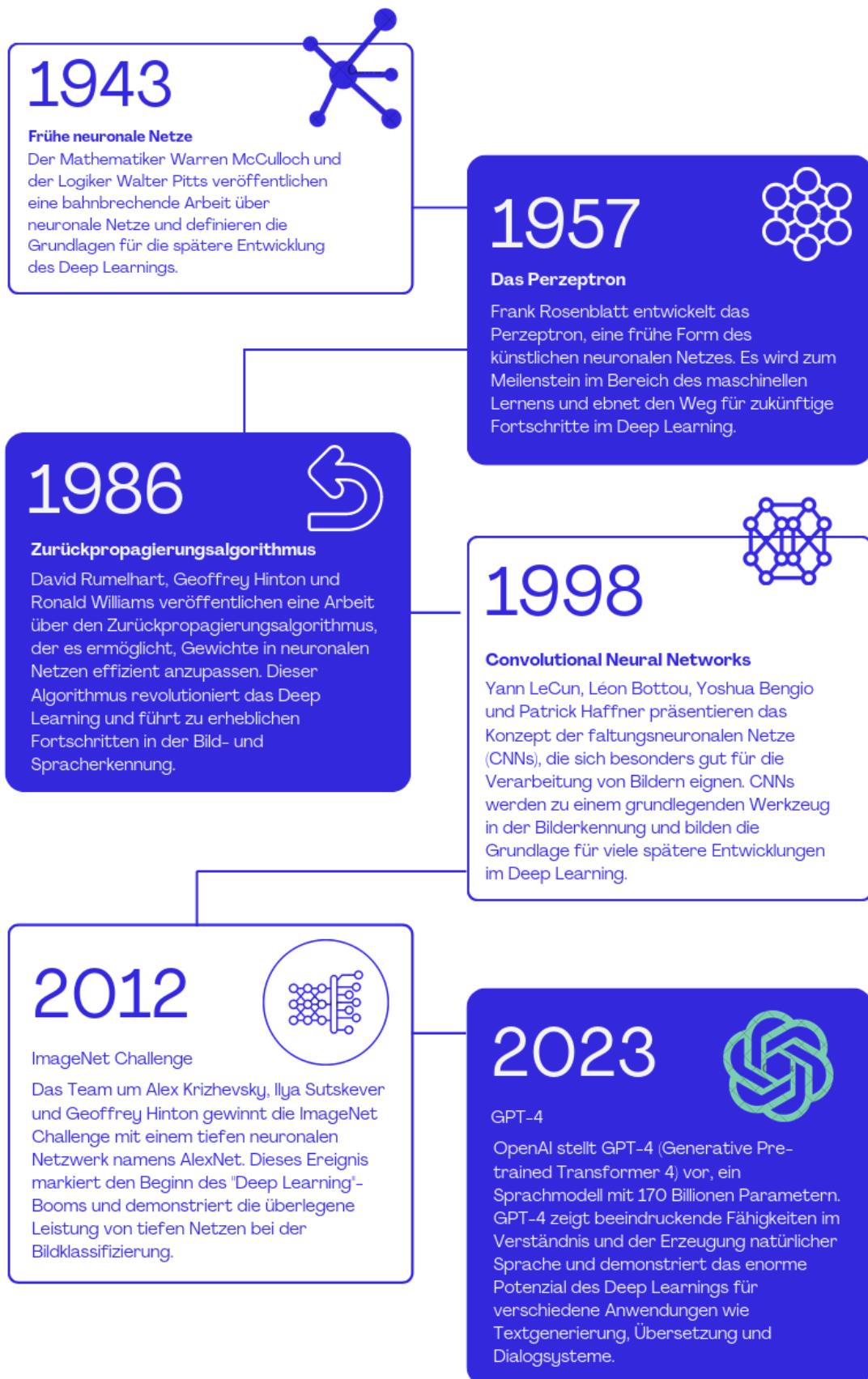


Abbildung 6.1: Timeline der Geschichte von Deep Learning

# Kapitel 7

## Grundlagen von Convolutional Neural Networks

### 7.1 Wie lernen Maschinen?

In diesem Abschnitt werden die fundamentalen Konzepte des maschinellen Lernens und des Deep Learnings erläutert. Maschinelles Lernen bezieht sich auf einen Prozess, bei dem maschinelle Systeme mithilfe von statistischen Modellen und Algorithmen selbstständig aus Daten lernen, ohne dass eine explizite Programmierung von Regeln oder Wissen erforderlich ist<sup>1</sup>. Deep Learning stellt einen spezifischen Teilbereich des maschinellen Lernens dar und konzentriert sich auf die Anwendung von neuronalen Netzwerken mit zahlreichen Schichten, um komplexe abstrakte Repräsentationen zu erlernen, ohne vorherige Merkmalsextraktion manuell beizufügen.<sup>2</sup>

#### 7.1.1 Definition von maschinellem Lernen

Das maschinelle Lernen ist ein Prozess, bei dem computergestützte Systeme mithilfe von statistischen Methoden und Algorithmen aus großen Datenmengen lernen, gestützt von einer expliziten Programmierung von Regeln und Merkmalsextraktionen erforderlich

---

<sup>1</sup>Xiang MA u. a. [2023]. *Deep Image Feature Learning with Fuzzy Rules*. arXiv: 1905.10575 [cs.CV].

<sup>2</sup>Issam EL NAQA und Martin J MURPHY [2015]. *What is machine learning?* Springer.

ist. Im Gegensatz zur traditionellen Programmierung, bei der eine festgelegte Abfolge von Anweisungen gegeben wird, basiert das maschinelle Lernen auf der Analyse und Verarbeitung großer Mengen an Daten. Durch diesen iterativen Lernprozess können Deep Learning Modelle Muster, Zusammenhänge und Regeln erkennen, um Vorhersagen zu treffen oder komplexe Aufgaben zu automatisieren. Diese Modelle werden trainiert, indem sie mit Eingabedaten gefüttert werden und ihre internen Parameter so angepasst werden, dass sie die gewünschten Ausgabewerte erzeugen.



Abbildung 7.1: Primitive Abstraktion eines Modellierungsprozesses für maschinelles Lernen

Der Trainingsprozess erfolgt durch die Optimierung von Modellparametern unter Verwendung von mathematischen Optimierungsmethoden wie dem Gradientenabstieg. Durch wiederholtes Training und Feinabstimmung der Modelle können sie kontinuierlich verbessert werden, um präzisere Vorhersagen zu generieren oder überlegene Leistungen bei bestimmten Aufgaben zu erzielen.<sup>34</sup>

### 7.1.2 Deep Learning

Deep Learning ist ein Teilbereich des maschinellen Lernens, der sich auf die Nutzung neuronaler Netzwerke mit mehreren Schichten konzentriert. Neuronale Netzwerke sind mathematische Modelle, welche biologische Neuronen simulieren und Schichten von verbundenen Neuronen enthalten. Jede Schicht empfängt Eingaben von der vorherigen Schicht und leitet Ausgaben an die nächste Schicht weiter.

<sup>3</sup>Gopinath REBALA u. a. [2019]. »Machine learning definition and basics«. In: *An introduction to machine learning*, S. 1–17.

<sup>4</sup>Issam EL NAQA und Martin J MURPHY [2015]. *What is machine learning?* Springer.

Ein Hauptvorteil des Deep Learnings gegenüber allgemeinerem Maschinellem Lernen, liegt in der automatischen Extraktion von Merkmalen aus den Trainingsdaten, ohne dass ein Experte (Mensch) diese manuell definieren muss. Durch die Kombination mehrerer Schichten können neuronale Netzwerke komplexe Muster und abstrakte Darstellungen erfassen, wodurch sie hochdimensionale Daten effektiv verarbeiten können.

Das Training von Deep-Learning-Modellen erfolgt üblicherweise mit großen Datensätzen und leistungsstarken GPUs, um die erforderlichen Berechnungen effizient durchführen zu können. Durch das iterative Training der Netzwerke werden die Gewichtungen und Parameter der einzelnen Neuronen angepasst, um die Fähigkeit des Modells zur präzisen Vorhersage oder Lösung komplexer Aufgaben zu verbessern.

Im weiteren Verlauf werden die grundlegenden Komponenten und Funktionen von Convolutional Neural Networks (CNNs) erläutert, die eine spezielle Art von neuronalen Netzwerken darstellen und insbesondere in der Bildverarbeitung weit verbreitet sind.<sup>5,6,7</sup>

### 7.1.3 Grundlagen von Convolutional Neural Networks

Convolutional Neural Networks (CNNs) sind eine spezielle Art von neuronalen Netzwerken, die in der Lage sind, Muster und Merkmale in Bildern effektiv zu erkennen. Sie zeichnen sich durch ihre Fähigkeit aus, lokale Verbindungen und Gewichtungen zwischen den Neuronen herzustellen und so räumliche Informationen in den Daten zu berücksichtigen. Die grundlegende Architektur eines CNNs besteht aus mehreren Schichten, darunter Convolutional Layers, Pooling Layers und Fully Connected Layers. In den Convolutional Layers werden Filter verwendet, um lokale Muster und Merkmale zu extrahieren, indem sie über das Eingabebild verschoben werden und die Faltungsoperation durchgeführt wird. Diese Filter können beispielsweise Kanten, Texturen oder spezifische Formen erfassen.

---

<sup>5</sup>Yann LECUN, Yoshua BENGIO und Geoffrey HINTON [2015]. »Deep learning«. In: *nature* 521.7553, S. 436–444.

<sup>6</sup>Jürgen SCHMIDHUBER [2015]. »Deep learning in neural networks: An overview«. In: *Neural networks* 61, S. 85–117.

<sup>7</sup>Deep LEARNING [2020]. »Deep learning«. In: *High-dimensional fuzzy clustering*.

Nach den Convolutional Layers folgen Pooling Layers, die dazu dienen, die räumliche Dimension der Merkmale zu reduzieren und die wichtigsten Informationen beizubehalten. Typische Pooling-Operationen umfassen das Max-Pooling, bei dem der maximale Wert in einem Bereich ausgewählt wird, und das Average-Pooling, bei dem der Durchschnittswert berechnet wird. Die Ausgabe der Pooling Layers wird dann an die Fully Connected Layers weitergeleitet, die eine traditionelle neuronale Netzwerkstruktur aufweisen. Diese Schichten dienen dazu, die erfassten Merkmale zu klassifizieren oder eine Vorhersage zu treffen, indem sie die Gewichtungen der Neuronen anpassen und die Aktivierungsfunktionen anwenden. Die Stärke von CNNs liegt in ihrer Fähigkeit, hierarchische Merkmale in den Daten zu erfassen. Die früheren Schichten lernen einfache Merkmale wie Kanten und Ecken, während die späteren Schichten komplexere Merkmale und abstrakte Darstellungen lernen. Durch das Training des Netzwerks mit großen Datensätzen können CNNs die Gewichtungen ihrer Neuronen so anpassen, dass sie die gewünschte Aufgabe, wie beispielsweise die Klassifizierung von Bildern, mit hoher Genauigkeit erfüllen können. CNNs haben eine breite Anwendungspalette in der Bildverarbeitung und sind in der Lage, komplexe Aufgaben wie Objekterkennung, Gesichtserkennung, Semantische Segmentierung und Bildgenerierung zu bewältigen. Ihre Leistungsfähigkeit beruht auf der Fähigkeit, automatisch relevante Merkmale aus den Daten zu extrahieren und sie in einer hierarchischen Weise zu verarbeiten. In den nächsten Abschnitten werden wir uns genauer mit den

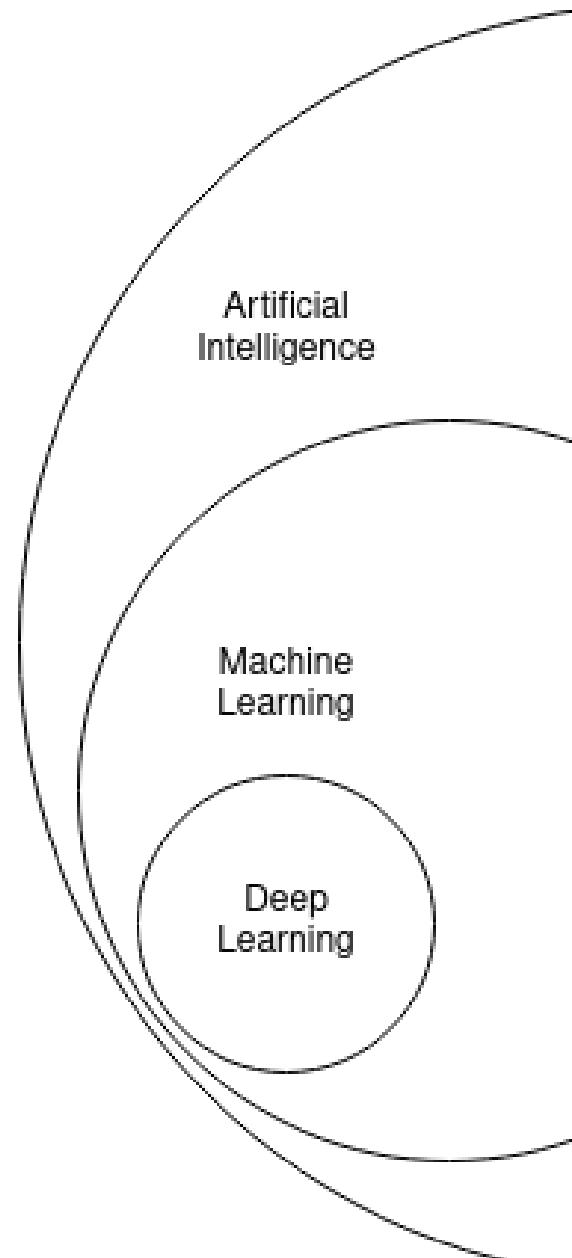


Abbildung 7.2: Einordnung der Begriffe Künstliche Intelligenz, Maschine Learning und Deep Learning.

einzelnen Komponenten von CNNs befassen, ihre Funktionsweise im Detail untersuchen und verschiedene Architekturen und Techniken kennenlernen, die in der Praxis weit verbreitet sind.<sup>8,9,10</sup>

### 7.1.4 Definition und Anwendungen von Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) sind eine Art von künstlichen neuronalen Netzwerken, die speziell für die Verarbeitung von Daten mit räumlicher Struktur entwickelt wurden. Sie zeichnen sich durch ihre Fähigkeit aus, hierarchische Muster und Merkmale in komplexen Daten wie Bildern, Videos oder Tonaufnahmen zu erkennen. CNNs haben in den letzten Jahren enorme Fortschritte in verschiedenen Bereichen der Informatik gemacht, insbesondere in der Bildverarbeitung, Computer Vision und Mustererkennung.

#### Einsatz von Convolutional Neural Networks in der Bildverarbeitung

Convolutional neural networks (CNNs) spielen eine entscheidende Rolle bei der automatischen Analyse und Verarbeitung von visuellen Daten in der Bildverarbeitung. Durch ihre einzigartige Architektur sind sie in der Lage, Bilder in ihre Bestandteile zu zerlegen und räumliche Merkmale wie Kanten, Formen und Texturen zu extrahieren. Dies geschieht durch die Kombination mehrerer Convolutional Layers und Pooling Layers, wodurch komplexe visuelle Muster erlernt und hochdimensionale Daten effektiv verarbeitet werden können. Als Ergebnis haben sich zahlreiche Anwendungen wie Objekterkennung, Gesichtserkennung, semantische Segmentierung und Bildgenerierung entwickelt. Die fortschrittlichen

---

<sup>8</sup>Yann LECUN u. a. [1998]. »Gradient-based learning applied to document recognition«. In: *Proceedings of the IEEE* 86.11, S. 2278–2324.

<sup>9</sup>Alex KRIZHEVSKY, Ilya SUTSKEVER und Geoffrey E HINTON [2012]. »Imagenet classification with deep convolutional neural networks«. In: *Advances in neural information processing systems*, S. 1097–1105.

<sup>10</sup>Kaiming HE u. a. [2016]. »Deep residual learning for image recognition«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 770–778; Karen SIMONYAN und Andrew ZISSERMAN [2014]. »Very deep convolutional networks for large-scale image recognition«. In: *arXiv preprint arXiv:1409.1556*.

Fähigkeiten von CNNs haben die Leistungsfähigkeit von Bildverarbeitungssystemen erheblich verbessert und zu bahnbrechenden Fortschritten in Bereichen wie autonomes Fahren<sup>11</sup>, medizinische Bildgebung<sup>12</sup> und Überwachungstechnologie<sup>13</sup> geführt.

### Anwendungen von Convolutional Neural Networks in der Computer Vision

Die Computer Vision<sup>14</sup> ist ein interdisziplinäres Forschungsfeld, das sich mit der Entwicklung von Algorithmen und Techniken zur Erfassung, Interpretation und Verarbeitung von visuellen Informationen befasst. In diesem Bereich haben Convolutional neural networks (CNNs) eine Revolution ausgelöst (siehe 6.1), da sie die Fähigkeit besitzen, automatisch relevante visuelle Merkmale aus Bildern oder Videosequenzen zu extrahieren. Durch das Training mit großen Datensätzen können CNNs lernen, Objekte und Szenen zu erkennen, Gesichter zu identifizieren, Gesten zu verstehen und komplexe visuelle Aufgaben zu lösen. Die Anwendungen von CNNs in der Computer Vision sind vielfältig und umfassen Bereiche wie automatische Fahrzeugerkennung, Augmented Reality, Robotik und intelligente Überwachungssysteme. Durch den Einsatz von CNNs wurden die Grenzen der Computer Vision erweitert und Computer sind nun in der Lage, die visuelle Welt um uns herum besser zu verstehen.

### Mustererkennung und Klassifikation mit Convolutional Neural Networks

Ein weiterer bedeutender Anwendungsbereich von Convolutional neural networks (CNNs) liegt in der Mustererkennung und Klassifikation. Durch das Training mit annotierten Datensätzen sind CNNs in der Lage, Muster und Zusammenhänge in den Daten zu erkennen und verschiedene Klassen oder Kategorien von Objekten zu unterscheiden.

---

<sup>11</sup>Yu HUANG und Yue CHEN [2020]. *Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies*. arXiv: 2006.06091 [cs.CV].

<sup>12</sup>Aidan BOYD, Adam CZAJKA und Kevin BOWYER [2020]. *Deep Learning-Based Feature Extraction in Iris Recognition: Use Existing Models, Fine-tune or Train From Scratch?* arXiv: 2002.08916 [cs.CV].

<sup>13</sup>Scott ROBBINS [2022]. »Machine Learning, Mass Surveillance, and National Security: Data, Efficacy, and Meaningful Human Control«. In: *The Palgrave Handbook of National Security*, S. 371–388.

<sup>14</sup>George STOCKMAN und Linda G SHAPIRO [2001]. *Computer vision*. Prentice Hall PTR.

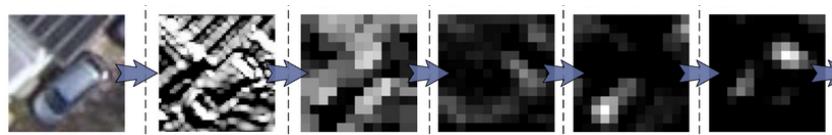


Abbildung 7.3: Einsicht in die Vorhergehensweise eines Neuronalen Netzen, und dessen automatische Merkmalsextraktion au mehreren Schichten.<sup>16</sup>

Dies ermöglicht die automatische Klassifizierung von Bildern, Texten, Tonaufnahmen und vielen anderen Datenarten. In den Bereichen der Texterkennung, Spracherkennung, Sprachübersetzung und anderen Bereichen der Mustererkennung haben CNNs bahnbrechende Fortschritte erzielt. Sie demonstrieren bemerkenswerte Fähigkeiten zur Bewältigung komplexer Klassifikationsaufgaben mit hoher Genauigkeit und haben somit die Effizienz und Genauigkeit von maschinellen Lernsystemen in erheblichem Maße verbessert.

### Weitere Anwendungsbeispiele

Neben den oben genannten Bereichen finden Convolutional Neural Networks auch in vielen anderen Bereichen Anwendung. Hier sind einige Beispiele:

- Medizinische Bildgebung: CNNs werden verwendet, um medizinische Bilder wie Röntgenaufnahmen, MRT-Scans und CT-Scans zu analysieren und Krankheiten zu diagnostizieren. Sie können Tumore, Anomalien oder andere gesundheitliche Zustände identifizieren, was Ärzten bei der genauen Diagnose und Behandlung unterstützt<sup>17</sup>.
- Sprachverarbeitung: CNNs können in der automatischen Spracherkennung und Sprachsynthese eingesetzt werden. Sie können Audiosignale analysieren und Transkriptionen von gesprochener Sprache generieren. Dies ermöglicht Anwendungen wie Sprachsteuerungssysteme, Sprachassistenten und Untertitelungsdienste.<sup>18</sup>

<sup>17</sup>Tawsifur RAHMAN u. a. [2020]. »Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray«. In: *Applied Sciences* 10.9, S. 3233.

<sup>18</sup>Nishtha H TANDEL, Harshadkumar B PRAJAPATI und Vipul K DABHI [2020]. »Voice recognition and voice comparison using machine learning techniques: A survey«. In: *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, S. 459–465.

- Finanzanalyse: CNNs werden zur Vorhersage von Finanzmärkten und zur Erkennung von betrügerischen Transaktionen eingesetzt. Sie können komplexe Muster in historischen Finanzdaten erkennen und Modelle entwickeln, um zukünftige Trends oder Risiken vorherzusagen.<sup>19</sup>
- Naturwissenschaften: In den Naturwissenschaften werden CNNs zur Analyse von großen Datensätzen aus Bereichen wie Astronomie<sup>20</sup>, Genetik und Teilchenphysik eingesetzt. Sie können dabei helfen, Muster, Zusammenhänge und neue Erkenntnisse in komplexen wissenschaftlichen Daten zu identifizieren. Im Beispiel von Astromoie, können CNNs eingesetzt werden um seltene Objektiv-Konfigurationen zu finden.

Insgesamt bieten Convolutional neural networks (CNNs) eine vielfältige Bandbreite an Anwendungen, die von der Bildverarbeitung über die Computer Vision bis hin zur Mustererkennung reichen. Ihre Fähigkeit, komplexe Muster in hochdimensionalen Daten zu erfassen, hat die Möglichkeiten der automatisierten Datenverarbeitung und -analyse erweitert und damit zahlreiche innovative Lösungen in verschiedenen Bereichen ermöglicht.

### 7.1.5 Wie unterscheiden sich CNNs von anderen neuronalen Netzwerkarchitekturen?

Convolutional neural networks (CNNs) stellen eine spezifische Architektur von künstlichen neuronalen Netzwerken dar, die sich in einigen wesentlichen Punkten von anderen Netzwerkarchitekturen unterscheiden. Im Folgenden werden zwei wichtige Unterschiede hervorgehoben:

---

<sup>19</sup>Jou-Fan CHEN u. a. [Nov. 2016]. »Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks«. In: *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE. DOI: 10.1109/ccbd.2016.027. URL: <https://doi.org/10.1109/ccbd.2016.027>.

<sup>20</sup>Andrew DAVIES, Stephen SERJEANT und Jane M BROMLEY [2019]. »Using convolutional neural networks to identify gravitational lenses in astronomical images«. In: *Monthly Notices of the Royal Astronomical Society* 487.4, S. 5263–5271.

### Räumlich-sensitive Neuronen und feste Gewichtungen

Im Gegensatz zu vollständig verbundenen neuronalen Netzwerken verwenden CNNs räumlich-sensitive Neuronen und feste Gewichtungen, um lokale Muster in den Eingabedaten zu erkennen. Bei einem vollständig verbundenen Netzwerk sind alle Neuronen einer Schicht mit allen Neuronen der nächsten Schicht verbunden. Dies führt dazu, dass die räumliche Struktur der Daten nicht berücksichtigt wird und das Netzwerk möglicherweise nicht in der Lage ist, lokale Muster und Zusammenhänge effektiv zu erfassen. CNNs hingegen verwenden sogenannte Filter oder Kernel, die über das Eingabebild verschoben werden, um lokale Merkmale zu extrahieren.

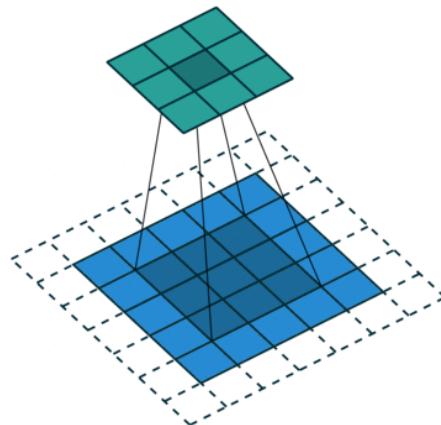


Abbildung 7.4: Sobel Kernel.

Diese Filter<sup>21</sup> haben feste Gewichtungen, die während des Trainings angepasst werden, um auf spezifische Muster zu reagieren. Durch die Verwendung räumlich-sensitiver Neuronen und fester Gewichtungen sind CNNs in der Lage, lokalisierte Merkmale wie Kanten, Texturen und spezifische Formen zu erfassen.

---

<sup>21</sup>Sumang GOEL [Juni 2020]. *Understanding Convolutional Neural Network*. Medium. URL: <https://medium.com/@sumangoel151/understanding-convolutional-neural-network-76e465f65ef3>.

## Pooling-Operationen zur Dimensionalitätsreduktion und Erhöhung der Robustheit

Ein weiterer Unterschied besteht darin, dass CNNs Pooling-Operationen verwenden, um die Dimensionalität der Eingabedaten zu reduzieren und die Robustheit gegenüber Translationen zu erhöhen. Nach den Convolutional Layers folgen in einem CNN typischerweise Pooling Layers. In diesen Schichten werden lokal aggregierte Informationen aus den vorherigen Schichten extrahiert, indem beispielsweise der maximale Wert (Max-Pooling) oder der Durchschnittswert (Average-Pooling) innerhalb eines bestimmten Bereichs ausgewählt wird.

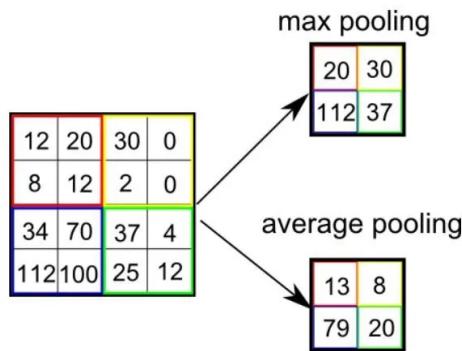


Abbildung 7.5: Max und average Pooling.

Durch das Anwenden von Pooling-Operationen wird die räumliche Auflösung der Merkmalskarten reduziert, während wichtige Informationen beibehalten werden. Dies führt zu einer effizienteren Verarbeitung<sup>22</sup> der Daten und einer erhöhten Robustheit gegenüber kleinen Translationen in den Eingabedaten. Diese Eigenschaft ist besonders vorteilhaft, wenn die genaue Position der Merkmale in den Daten nicht von entscheidender Bedeutung ist. Die Verwendung räumlich-sensitiver Neuronen und fester Gewichtungen sowie die Integration von Pooling-Operationen sind charakteristische Merkmale von Convolutional neural networks (CNNs), die es ihnen ermöglichen, räumliche Informationen zu berücksichtigen, lokale Muster zu erkennen und komplexe visuelle Aufgaben effektiv zu

<sup>22</sup>Mujastia Feliati MUHDALIFAH [2022]. »Pooling comparison in CNN architecture for Javanese script classification«. In: *International Journal of Informatics and Computation* 3.2, S. 15–22.

bewältigen.

### 7.1.6 Warum sind Convolutional Neural Networks besonders nützlich für Bild- und Videodaten?

Convolutional neural networks (CNNs) haben sich als äußerst nützlich und effektiv bei der Verarbeitung von Bild- und Videodaten erwiesen. Im Folgenden werden einige Gründe dafür erläutert:

#### Automatische Merkmalsextraktion

Convolutional neural networks (CNNs) haben die Fähigkeit, automatisch Merkmale aus Bildern und Videos zu extrahieren, ohne dass eine manuelle Merkmalsextraktion erforderlich ist. Traditionelle Ansätze zur Verarbeitung von visuellen Daten erforderten oft die Definition und Konstruktion von spezifischen Merkmalsvektoren durch Experten. Im Gegensatz dazu können CNNs lernen, hierarchische Merkmale direkt aus den Rohdaten zu extrahieren. Durch die Kombination von Faltungs- und Pooling-Schichten sind CNNs in der Lage, lokale Muster wie Kanten, Texturen und Formen zu erkennen, die zur Identifizierung von Objekten und zur Klassifizierung von Bildern wesentlich sind.<sup>23</sup>

#### Lernen räumlicher Hierarchien von Merkmalen

Die Verwendung von faltenden Schichten ermöglicht es CNNs, räumliche Hierarchien von Merkmalen zu lernen, was sie besonders effektiv für die Objekterkennung und Klassifizierung macht. Durch die schrittweise Anwendung von Faltung und Pooling in aufeinanderfolgenden Schichten können CNNs komplexe visuelle Konzepte erfassen. Anfängliche Schichten lernen einfache Merkmale wie Kanten und Farbkontraste, während tiefere Schichten komplexere Merkmale wie Formen, Strukturen und Objekte auf höheren Abstraktionsebenen erkennen können. Diese Hierarchie von Merkmalen ermöglicht es CNNs, Objekte mit hoher Genauigkeit zu identifizieren und Bilder korrekt zu klassifizieren.

---

<sup>23</sup>Wolfgang ERTEL und Nathanael T BLACK [2016]. *Grundkurs Künstliche Intelligenz*. Bd. 4. Springer.

## Anpassung auf weitere Anwendungsbereiche

Ein weiterer Vorteil von CNNs ist ihre Fähigkeit, ohne viel Aufwand und vortrainiertes Wissen auf verschiedene Anwendungsbereiche angepasst zu werden. Ein bereits vortrainiertes Modell für die Segmentierung von Menschen kann beispielsweise für die Erkennung von Objekten wiederverwendet werden. Indem man das Netzwerk auf neue Daten feinabstimmt (Fine-Tuning), kann es auf spezifische Aufgaben oder Domänen angepasst werden, ohne von Grund auf neu trainiert werden zu müssen. Dies spart Zeit und Ressourcen und ermöglicht es, CNNs für eine Vielzahl von Bild- und Videodaten-Anwendungen einzusetzen.

## Toleranz gegenüber Translationen, Skalierungen und Verzerrungen

CNNs sind auch in der Lage, Translationen, Skalierungen und Verzerrungen in den Eingabedaten zu tolerieren<sup>24</sup>, was für die Verarbeitung von Bildern und Videos von Vorteil ist. Aufgrund der Verwendung von Faltungsschichten und Pooling-Operationen sind CNNs invariant gegenüber kleinen räumlichen Verschiebungen in den Eingabedaten. Dies bedeutet, dass ein Objekt, das sich leicht innerhalb eines Bildes bewegt oder skaliert oder verzerrt wird, immer noch von einem CNN erkannt und klassifiziert werden kann. Dies ist besonders vorteilhaft für Anwendungen wie die Objekterkennung in Videos oder die Klassifizierung von Bildern, bei denen die Position, Größe oder Ausrichtung der Objekte variieren können. Durch die Toleranz<sup>25</sup> (siehe Tabelle 7.1) gegenüber solchen Variationen wird die Robustheit und Zuverlässigkeit von CNNs in der Verarbeitung von Bild- und Videodaten verbessert. Sie können auch mit unterschiedlichen Auflösungen von Bildern umgehen und sind in der Lage, Informationen auf verschiedenen Skalenebenen zu extrahieren. Diese Fähigkeiten machen CNNs zu einem effektiven Werkzeug für eine Vielzahl von Bild- und Videodaten-Anwendungen.

Von der automatischen Erkennung von Objekten und Gesichtern bis hin zur semanti-

---

<sup>24</sup>Joosung LEE u. a. [2020]. *AD-VO: Scale-Resilient Visual Odometry Using Attentive Disparity Map*. arXiv: 2001.02090 [cs.CV].

<sup>25</sup>Joosung LEE u. a. [2020]. *AD-VO: Scale-Resilient Visual Odometry Using Attentive Disparity Map*. arXiv: 2001.02090 [cs.CV].

| RGB-VO |               |                  | D-VO   |               |                  | AD-VO  |               |                  |
|--------|---------------|------------------|--------|---------------|------------------|--------|---------------|------------------|
|        | skip-ordering | no skip-ordering |        | skip-ordering | no skip-ordering |        | skip-ordering | no skip-ordering |
| Seq    | Trans%        | Rot[deg/m]       | Trans. | Rot.          | Trans.           | Trans. | Rot.          | Trans.           |
| 8      | 14.39         | 0.0452           | 16.41  | 0.0547        | 13.36            | 0.0478 | 9.21          | 0.0341           |
| 9      | 9.59          | 0.0397           | 13.7   | 0.0417        | 8.79             | 0.0365 | 7.43          | 0.032            |
| 10     | 19.18         | 0.0438           | 18.48  | 0.0954        | 14.36            | 0.0671 | 7.26          | 0.0317           |
| avg    | 13.83         | 0.0438           | 16.02  | 0.0562        | 12.44            | 0.0474 | 8.59          | 0.0334           |
| std    | 2.718         | 0.0023           | 1.407  | 0.0147        | 1.995            | 0.0083 | 0.868         | 0.001            |

Tabelle 7.1: Leistung der Algorithmen: Translations- und Rotationsfehler von Testsequenzen unter Verwendung des KITTI Devkit. AD-VO funktioniert zuverlässig in verschiedenen Fahrumgebungen. 'D' steht für die Verwendung einer Disparitätskarte und 'A' bedeutet, dass das Modell einen Attention-Block anpasst.

schen Segmentierung und Bildgenerierung haben CNNs die Grenzen der Bildverarbeitung und Computer Vision erweitert und bahnbrechende Fortschritte in Bereichen wie autonomes Fahren, medizinische Bildgebung und Überwachungstechnologie ermöglicht. Insgesamt sind Convolutional Neural Networks aufgrund ihrer automatischen Merkmalsextraktion, des Lernens räumlicher Hierarchien von Merkmalen, der Anpassungsfähigkeit auf verschiedene Anwendungsbereiche und der Toleranz gegenüber Translationen, Skalierungen und Verzerrungen besonders nützlich für die Verarbeitung und Analyse von Bild- und Videodateien.

## 7.2 Anwendungen von CNNs

Convolutional neural networks (CNNs) finden in vielen Anwendungsgebieten Anwendung, insbesondere in der Bildverarbeitung und Mustererkennung. Durch ihre Fähigkeit, komplexe Merkmale zu erkennen und Bilder automatisch zu klassifizieren, haben sie bahnbrechende Fortschritte in verschiedenen Bereichen erzielt.

### 7.2.1 Bildklassifikation

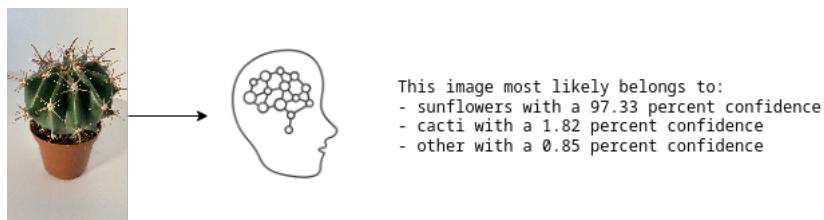


Abbildung 7.6: Modell zur Bildklassifizierung

Bildklassifikation ist eine der grundlegenden Anwendungen von CNNs. Mit Hilfe von trainierten Modellen können CNNs Bilder automatisch in verschiedene Kategorien klassifizieren, wie beispielsweise Hund vs. Katze, Auto vs. Fahrrad usw. Diese Fähigkeit beruht auf der Nutzung tieferer Schichten in einem CNN. Durch diese tieferen Schichten können komplexe Merkmale wie Texturen, Formen und Strukturen erkannt werden, was zu einer verbesserten Bildklassifikation führt. Indem CNNs lernen, spezifische Merkmale auf verschiedenen Abstraktionsebenen zu erfassen, können sie komplexe visuelle Muster in Bildern effizient identifizieren und analysieren.<sup>26</sup>

### 7.2.2 Objekterkennung

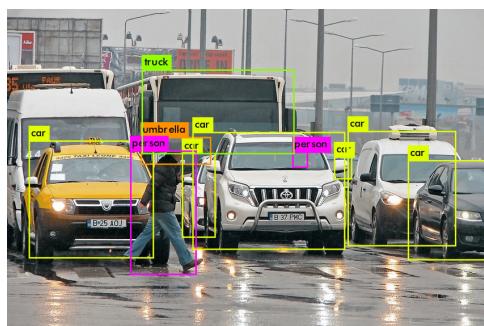


Abbildung 7.7: Modell zur Objekterkennung.

---

<sup>26</sup>Tianmei GUO u. a. [2017]. »Simple convolutional neural network on image classification«. In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, S. 721–724.

Convolutional neural networks (CNNs) werden häufig für die Erkennung und Lokalisierung von Objekten in Bildern eingesetzt (siehe Bild 7.7<sup>27</sup>). Durch die Nutzung tieferer Schichten können CNNs komplexe Merkmale wie Texturen, Formen und Strukturen erkennen, was die Genauigkeit der Bildklassifikation verbessert. Zusätzlich ermöglicht der Einsatz von Region Proposal Networks (RPNs) den CNNs, die genauen Begrenzungsrahmen der erkannten Objekte zu berechnen.

### 7.2.3 Gesichtserkennung

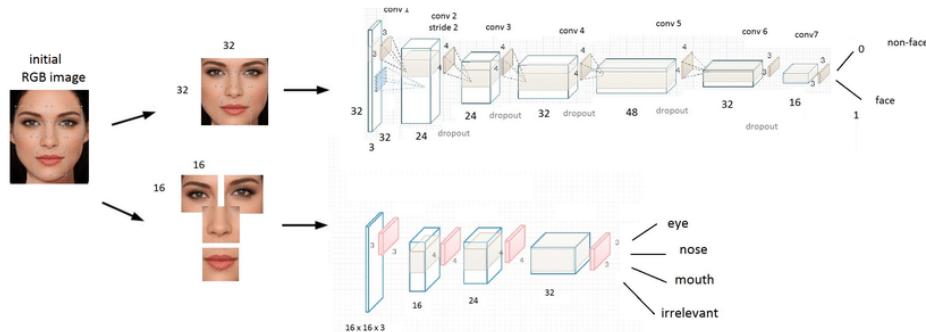


Abbildung 7.8: Modell zur Gesichtserkennung.

Convolutional neural networks (CNNs) haben in der Gesichtserkennung signifikante Fortschritte gemacht und werden häufig für die Identifizierung von Personen in Bildern und Videos eingesetzt. Durch den Einsatz von CNNs können Gesichtsmerkmale wie Augen, Nase und Mund erkannt und zur Identifizierung von Personen verwendet werden. Diese Technologie findet Anwendung in Zugangskontrollen, Überwachungssystemen und biometrischer Authentifizierung.

---

<sup>27</sup>Danai TRIANTAFYLLOU und Anastasios TEFAS [Okt. 2017]. »A Fast Deep Convolutional Neural Network for Face Detection in Big Visual Data«. In: S. 61–70. ISBN: 978-3-319-47897-5. DOI: 10.1007/978-3-319-47898-2\_7.

### 7.2.4 Natural Language Processing (NLP)

Obwohl CNNs hauptsächlich für die Verarbeitung von Bildern entwickelt wurden, können sie auch in bestimmten Anwendungen des Natural Language Processing (NLP) eingesetzt werden<sup>28</sup>. In der Natural Language Processing (NLP) können CNNs zur Klassifizierung von Texten, Sentimentanalyse, maschinellen Übersetzung und Textgenerierung eingesetzt werden.

Durch die Anwendung von Faltungsschichten auf Textsequenzen können CNNs relevante Merkmale extrahieren und wichtige Informationen für die Klassifizierung liefern.

### 7.2.5 Weitere Anwendungen

Neben den oben genannten Anwendungen finden CNNs in vielen anderen Bereichen Anwendung, wie zum Beispiel in der medizinischen Bildgebung, bei autonomen Fahrzeugen und der Spracherkennung<sup>29</sup>. Sie ermöglichen die Automatisierung und Verbes-

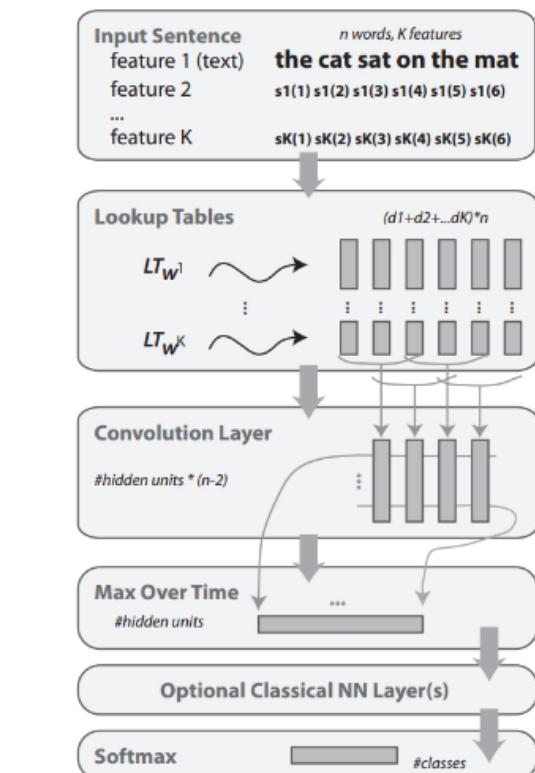


Abbildung 7.9: Eine verallgemeinerte Deep-NN-Architektur für natürliche Sprachverarbeitung (NLP). [COLLOBERT und WESTON 2008]

<sup>28</sup>Wei WANG und Jianxun GANG [2018]. »Application of convolutional neural network in natural language processing«. In: *2018 international conference on information Systems and computer aided education (ICISCAE)*. IEEE, S. 64–70.

<sup>29</sup>Ronan COLLOBERT und Jason WESTON [2008]. »A unified architecture for natural language processing: Deep neural networks with multitask learning«. In: *Proceedings of the 25th international conference on Machine learning*, S. 160–167.

serung verschiedener Aufgaben, indem sie Muster und Zusammenhänge in den Daten erkennen. Mit der kontinuierlichen Weiterentwicklung von CNNs eröffnen sich ständig neue Anwendungsmöglichkeiten in verschiedenen Branchen.

## 7.3 Architektur von CNNs

### 7.3.1 Input Layer

Die Input Layer eines Convolutional neural networks (CNNs) wird bei der Aufnahme und Vorverarbeitung der Rohdaten, die in der Regel Bilder oder Videos sind benutzt. Sie ist die erste Schicht des Netzwerks und ermöglicht den Datenfluss in das CNN. Die Eingabeschicht besteht aus einer Anordnung von Neuronen, wobei jedes Neuron mit einem bestimmten Bereich des Eingabebildes verbunden ist.

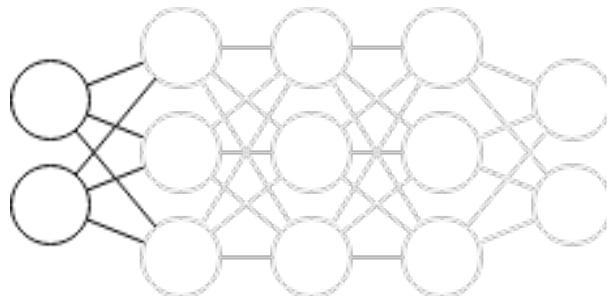


Abbildung 7.10: Input layers

Diese Verbindungen stellen sicher, dass jedes Neuron Informationen aus einem begrenzten lokalen Bereich des Bildes erhält. Auf diese Weise kann das CNN räumliche Informationen über die Daten berücksichtigen und lokalisierte Merkmale erfassen. Die Größe der Input Layer ist normalerweise fest vorgegeben, um eine konsistente Verarbeitung der Daten sicherzustellen. Für RGB-Bilder beträgt die typische Größe beispielsweise 244x244x3, wobei die ersten beiden Dimensionen die Breite und Höhe des Bildes repräsentieren und die letzte Dimension die Farbkanäle (Rot, Grün, Blau) darstellt. Die Input Layer spielt eine wichtige Rolle bei der Einführung der Daten in das CNN und ermöglicht

die weitere Verarbeitung und Extraktion von Merkmalen in den nachfolgenden Schichten.<sup>30</sup>

### 7.3.2 Hidden Layers

Die Hidden Layers<sup>31</sup> bilden die Hauptkomponente eines Convolutional neural networks (CNNs) und bestehen standardmäßig aus Convolutional-Layern, Pooling-Layern, vollständig verbundenen Layern und Aktivierungsfunktionen. Diese Schichten ermöglichen die Verarbeitung und Extraktion von Merkmalen aus den Eingabedaten. Die Convolutional-Layer führen Faltungsoperationen auf den Eingabedaten durch, um lokale Muster und Merkmale zu extrahieren. Durch die Anwendung von Filtern auf die Eingabedaten werden wichtige Informationen hervorgehoben und irrelevante Details unterdrückt. Dies ermöglicht es dem CNN, Merkmale wie Kanten, Texturen und Formen zu erkennen, die für die spätere Klassifizierung oder Segmentierung von entscheidender Bedeutung sind.

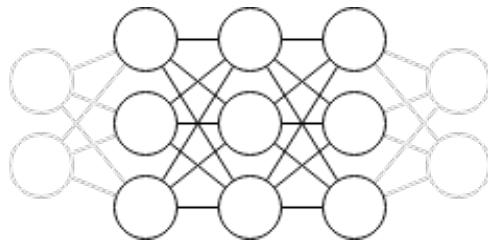


Abbildung 7.11: Hidden layers

Die Pooling-Layer sind für die Reduzierung der Dimensionalität der Daten verantwortlich. Sie ermöglichen es, die Größe der Merkmalskarten zu verkleinern und die Anzahl der Parameter im Modell zu reduzieren. Der Hauptzweck des Pooling besteht darin, die Rechenkomplexität des Modells zu verringern, indem weniger Merkmale beibehalten werden. Durch das Zusammenfassen von Informationen wird auch die Lokalisierungsinviananz verbessert, da die genaue Position eines bestimmten Merkmals in den Merkmalskarten

---

<sup>30</sup>John MURPHY [2016]. »An overview of convolutional neural network architectures for deep learning«. In: *Microway Inc*, S. 1–22.

<sup>31</sup>Adrian ROSEBROCK [2021]. *Convolutional Neural Networks (CNNs) and Layer Types*. URL: <https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/>.

weniger wichtig wird. Jedoch erhöhen Pooling-Layer nicht unbedingt die Translationssicherheit. Während des Pooling-Prozesses kann ein gewisser Informationsverlust auftreten, da nur die dominanten Merkmale beibehalten werden. Dadurch können Feinheiten und detaillierte Informationen verloren gehen, die möglicherweise für die genaue Klassifizierung oder Segmentierung von Objekten von Bedeutung sind. Insgesamt sind Pooling-Layer eine wichtige Komponente von CNNs, da sie die Dimensionalität reduzieren und die Rechenressourcen effizienter nutzen können. Es ist jedoch wichtig, sie mit Bedacht einzusetzen, abhängig von den Anforderungen der spezifischen Aufgabe und dem Trade-off zwischen Dimensionsreduktion und dem Erhalt wichtiger Informationen. Die Auswahl der richtigen Pooling-Strategie und -Parameter ist entscheidend, um ein optimales Gleichgewicht zwischen Dimensionalitätsreduktion und Informationsbewahrung zu erreichen.<sup>32</sup>

### 7.3.3 Output Layers

Die Output Layers bilden die letzte Schicht eines Convolutional neural networks (CNNs) und sind für die Generierung der endgültigen Vorhersage, Klassifizierung oder Segmentierung verantwortlich. Die Ausgabeschicht ist entscheidend für die Interpretation der durch das CNN ermittelten Merkmale und ermöglicht die Ausgabe der gewünschten Ergebnisse. Ähnlich wie bei den Hidden Layers führen Convolutional-Layer auch in der Ausgabeschicht Faltungsoperationen auf den vorverarbeiteten Eingabedaten durch.

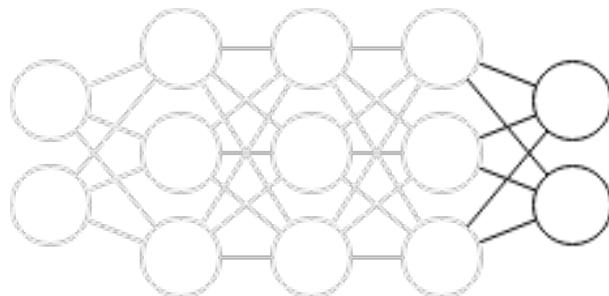


Abbildung 7.12: Output layers

---

<sup>32</sup>John MURPHY [2016]. »An overview of convolutional neural network architectures for deep learning«. In: *Microway Inc*, S. 1–22.

Diese Operationen dienen dazu, die relevanten Merkmale zu extrahieren und sie für die weitere Verarbeitung und Interpretation verfügbar zu machen. Die in den vorherigen Schichten erlernten Merkmale werden hier zusammengeführt, um eine aussagekräftige Vorhersage oder Klassifizierung zu ermöglichen. Die Form der Ausgabeschicht variiert je nach Anwendungsfall. Bei binärer Klassifizierung besteht die Ausgabeschicht in der Regel aus einem einzelnen Neuron, das eine Wahrscheinlichkeit oder Entscheidung für eine der beiden Klassen ausgibt. Bei Multi-Klassen-Klassifizierung hingegen werden mehrere Neuronenausgaben verwendet, um die Wahrscheinlichkeiten oder Entscheidungen für jede einzelne Klasse zu liefern. In unserem spezifischen Projekt ist die Ausgabeschicht ein weiteres Bild, das genau die gleiche Größe wie das Eingangsbild hat. Dies ermöglicht die Segmentierung des Eingangsbildes in verschiedene Bereiche oder die Generierung eines verarbeiteten Ausgabebildes mit spezifischen Eigenschaften. Die Ausgabeschicht stellt somit das Endergebnis des CNNs dar und ist entscheidend für die Interpretation und Verwendung der ermittelten Merkmale. Durch eine passende Auswahl der Ausgabeschicht können wir die gewünschten Ergebnisse erzielen, sei es eine Klassifizierung, Segmentierung oder die Generierung eines verarbeiteten Ausgabebildes.

## 7.4 Convolutional layers

### 7.4.1 Pooling Layers

Die Pooling Layers spielen eine wichtige Rolle in der Architektur von Convolutional neural networks (CNNs). Ihr Hauptzweck besteht darin, die Dimensionalität der Daten zu reduzieren, indem sie die Aktivierungswerte in bestimmten Bereichen zusammenfassen. Dies ermöglicht eine effizientere Verarbeitung der Daten und eine Verbesserung der räumlichen Invarianz gegenüber kleinen Translationen. In der Regel werden in CNNs zwei gängige Pooling-Operationen verwendet: das Max-Pooling und das Average-Pooling. Beim Max-Pooling wird der maximale Aktivierungswert in einem bestimmten Bereich ausgewählt und als Ergebnis verwendet. Beim Average-Pooling hingegen wird der Durchschnitt der Aktivierungswerte in diesem Bereich berechnet. Diese Operationen ermöglichen eine Reduktion

der Merkmalsdimensionen und helfen dabei, relevante Informationen beizubehalten.

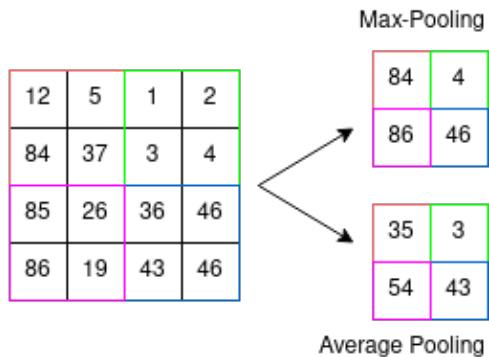


Abbildung 7.13: max und average Pooling.

In den meisten CNN-Architekturen werden Max-Pooling-Layers eingesetzt, da sie in der Regel bessere Ergebnisse erzielen. Durch die Auswahl des maximalen Aktivierungswerts wird sichergestellt, dass die dominanten Merkmale erhalten bleiben und die Netzwerkperformance verbessert wird. Pooling-Schichten bieten auch den Vorteil, die Anzahl der Parameter im Netzwerk zu reduzieren. Durch das Zusammenfassen der Informationen wird die Anzahl der zu lernenden Parameter verringert, was zu einer effizienteren Verarbeitung führt und Overfitting reduzieren kann. Ein weiterer Vorteil von Pooling-Layern besteht darin, dass sie die räumliche Invarianz gegenüber kleinen Translationen fördern. Da nur die relevanten Merkmale beibehalten werden, wird die Position dieser Merkmale in den Merkmalskarten weniger wichtig. Dies ermöglicht dem CNNs, auf ähnliche Merkmale in verschiedenen Bereichen eines Bildes zu reagieren und eine gewisse Robustheit gegenüber Translationen zu erreichen. Zusammenfassend kann gesagt werden, dass Pooling Layers eine wichtige Komponente in der Architektur von CNNs sind. Sie helfen dabei, die Dimensionalität zu reduzieren, die Anzahl der Parameter zu verringern und die räumliche Invarianz gegenüber Translationen zu verbessern. Durch die sorgfältige Auswahl und Platzierung von Pooling-Schichten können wir die Effizienz und Leistungsfähigkeit des CNNs steigern.

### 7.4.2 Fully Connected Layers

Fully Connected Layers, auch bekannt als vollständig verbundene Schichten, sind traditionelle neuronale Netzwerk-Schichten, bei denen alle Neuronen mit allen Neuronen der vorherigen Schicht verbunden sind. Diese Schichten spielen eine wichtige Rolle in der Architektur von Convolutional neural networks (CNNs) und dienen normalerweise am Ende des Netzwerks, um die extrahierten Merkmale zu klassifizieren.

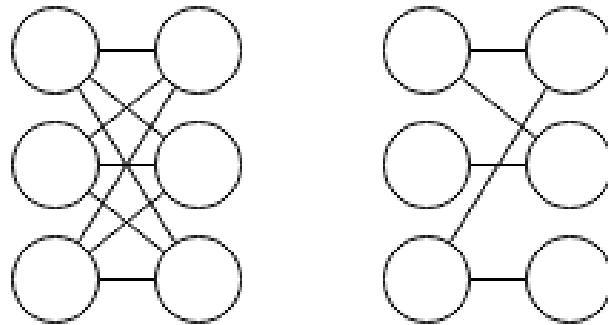


Abbildung 7.14: Links fully connected, rechts not fully connected layers.

In einem CNNs werden vor den vollständig verbundenen Schichten in der Regel Convolutional Layers und Pooling Layers verwendet, um Merkmale aus den Eingabedaten zu extrahieren und die Dimensionalität zu reduzieren. Diese Merkmale werden dann in den vollständig verbundenen Schichten verwendet, um eine endgültige Klassifizierung oder Vorhersage durchzuführen. Die Anzahl der Neuronen in den vollständig verbundenen Schichten hängt von der Anzahl der Klassen oder der spezifischen Aufgabe ab. Wenn beispielsweise ein CNNs zur Klassifizierung von Bildern in 10 verschiedene Kategorien verwendet wird, wird die Anzahl der Neuronen in der letzten vollständig verbundenen Schicht in der Regel auf 10 festgelegt, um eine Vorhersage für jede Klasse zu ermöglichen.

### 7.4.3 Overfitting und Dropout

Es ist wichtig anzumerken, dass Overfitting ein häufiges Problem in neuronalen Netzwerken ist, insbesondere wenn die Anzahl der Parameter hoch ist. Um Overfitting zu minimieren

und die Generalisierungsfähigkeit<sup>33</sup> des Modells zu verbessern, kann Dropout eingesetzt werden. Dropout ist eine Technik, bei der während des Trainings zufällig bestimmte Neuronen und ihre Verbindungen in den vollständig verbundenen Schichten deaktiviert werden. Dadurch wird die Vernetzung der Neuronen reduziert und das Modell wird robuster gegenüber Overfitting.

Data Augmentation ist eine effektive Methode zur Minimierung von Overfitting und Verbesserung der Generalisierungsfähigkeit von Modellen. Durch Rotationen können Bilder gedreht werden, um verschiedene Blickwinkel zu simulieren und die Robustheit gegenüber unterschiedlichen Ausrichtungen zu erhöhen. Shifting-Techniken ermöglichen das Verschieben von Bildern in verschiedene Richtungen, um die Positionierung der Objekte im Bild zu variieren und die Fähigkeit des Modells zu verbessern, sie unabhängig von ihrer genauen Position zu erkennen. Diese Transformationen helfen dabei, mehr Vielfalt in die Trainingsdaten einzuführen und das Modell darauf vorzubereiten, mit verschiedenen Blickwinkeln, Positionen und Perspektiven umzugehen, wodurch die Gefahr von Overfitting verringert und die Fähigkeit zur Generalisierung gesteigert wird.

Zusammenfassend sind Fully Connected Layers ein wesentlicher Bestandteil der Architektur von CNNs. Sie ermöglichen die Klassifizierung der extrahierten Merkmale und spielen eine entscheidende Rolle bei der Ausführung verschiedener Aufgaben. Die Anzahl der Neuronen in den vollständig verbundenen Schichten wird entsprechend der spezifischen Anforderungen der Aufgabe festgelegt. Durch den Einsatz von Techniken wie Dropout kann das Modell vor Overfitting geschützt werden und eine verbesserte Generalisierungsfähigkeit erzielen.

#### 7.4.4 Aktivierungsfunktionen

Aktivierungsfunktionen spielen eine wichtige Rolle in der Architektur von Convolutional neural networks (CNNs). Sie werden auf die Ausgaben der Neuronen angewendet und

<sup>33</sup>Panissara THANAPOL u. a. [2020]. »Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition«. In: *2020-5th International Conference on Information Technology (InCIT)*. IEEE, S. 300–305.

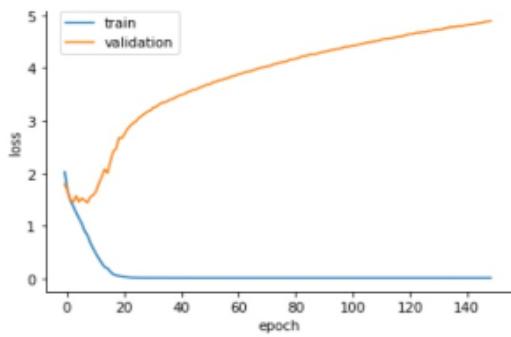


Abbildung 7.15: Overfitting und geringe Generalisierung in alleinigen CNN.

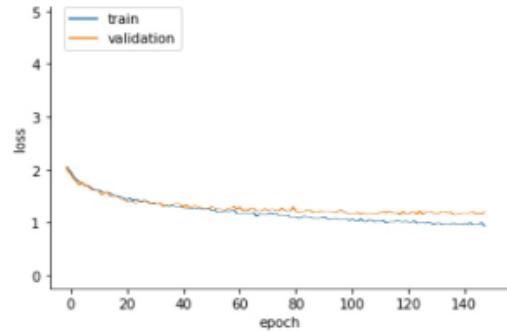


Abbildung 7.16: Reduktion von Overfitting und Verbesserung der Generalisierung durch Anwendung von Verschiebung in Breite und Höhe zusammen mit Dropout bei Convolutional Neural Networks (CNN)

führen nichtlineare Transformationen durch<sup>34</sup>. Diese Funktionen ermöglichen es dem Netzwerk, komplexe Zusammenhänge zu modellieren und eine höhere Ausdrucksfähigkeit zu erreichen. In CNNs werden verschiedene Aktivierungsfunktionen verwendet, von denen einige besonders häufig anzutreffen sind.

Eine gängige Aktivierungsfunktion ist die ReLU<sup>35</sup> (Rectified Linear Unit), die für positive Eingaben den Wert beibehält und negative Eingaben auf Null setzt. Die ReLU-Funktion ist bekannt für ihre Einfachheit und Effektivität bei der Behandlung von Vanishing-Gradient-Problemen, bei denen die Gradienten während des Trainings verschwinden können.

Eine weitere gängige Aktivierungsfunktion ist die Sigmoid-Funktion<sup>36</sup>, die eine S-

<sup>34</sup>Wang HAO u. a. [Dez. 2020]. »The Role of Activation Function in CNN«. In: *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*. IEEE. DOI: 10.1109/itca52113.2020.00096. URL: <https://doi.org/10.1109/itca52113.2020.00096>.

<sup>35</sup>Yingying WANG u. a. [2020]. »The influence of the activation function in a convolution neural network model of facial expression recognition«. In: *Applied Sciences* 10.5, S. 1897.

<sup>36</sup>Yingying WANG u. a. [2020]. »The influence of the activation function in a convolution neural network

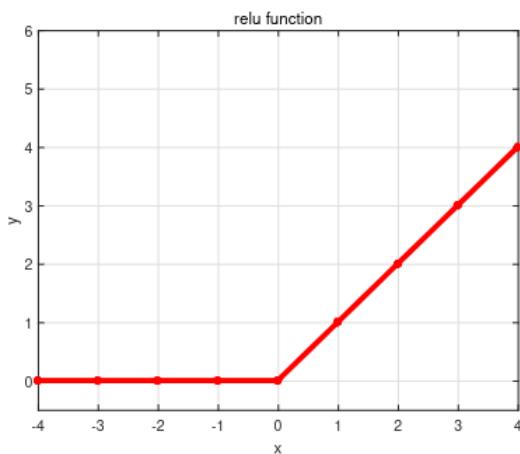


Abbildung 7.17: ReLu Aktivierungsfunktion.

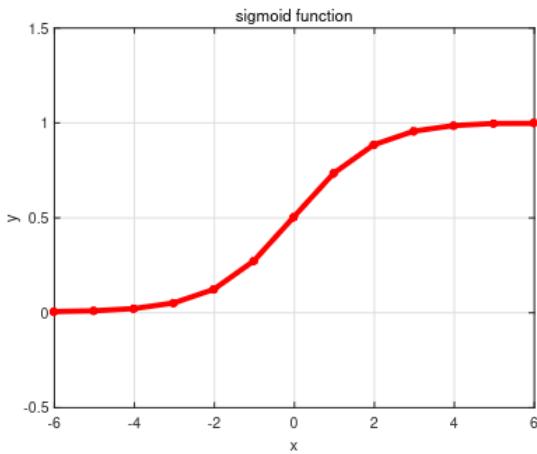


Abbildung 7.18: Sigmoid Aktivierungsfunktion.

förmige Kurve erzeugt. Sie komprimiert den Wertebereich der Ausgabe auf den Bereich zwischen 0 und 1, wodurch die Funktion gut zur Modellierung von Wahrscheinlichkeiten geeignet ist. Die Sigmoid-Funktion wird oft in binären Klassifizierungsaufgaben verwendet. Die tanh-Funktion (Hyperbolic Tangent) ist eine weitere Aktivierungsfunktion, die eine S-förmige Kurve erzeugt, aber im Vergleich zur Sigmoid-Funktion eine symmetrische Ausgabe mit einem Wertebereich zwischen -1 und 1 liefert. Die tanh-Funktion wird oft in mehrklassigen Klassifizierungsaufgaben eingesetzt. Die Verwendung von Aktivierungsfunktionen ermöglicht es CNNs, nichtlineare Zusammenhänge zu erlernen und komplexe Muster in den Daten zu erkennen. Durch die Einführung von Nichtlinearität können CNNs flexibler auf verschiedene Eingabemuster reagieren und ihre Fähigkeit zur Modellierung komplexer Zusammenhänge verbessern. Zusammenfassend spielen Aktivierungsfunktionen eine wesentliche Rolle in der Architektur von CNNs. Sie ermöglichen nichtlineare Transformationen der Neuronausgaben und helfen dabei, die Fähigkeit des Netzwerks zur Modellierung komplexer Zusammenhänge zu verbessern. Die Auswahl der geeigneten Aktivierungsfunktion hängt von der Art der Aufgabe und den spezifischen Anforderungen des Modells ab.

---

model of facial expression recognition». In: *Applied Sciences* 10.5, S. 1897.

### 7.4.5 Verlustfunktionen

Verlustfunktionen sind ein wesentlicher Bestandteil der Architektur von Convolutional neural networks (CNNs). Sie dienen dazu, den Unterschied zwischen den Vorhersagen des Modells und den tatsächlichen Werten der Daten zu messen. Die Wahl einer geeigneten Verlustfunktion ist entscheidend, um das CNN für die spezifische Aufgabe zu optimieren und die Genauigkeit der Vorhersagen zu verbessern<sup>37</sup>. In der Klassifizierung werden häufig Verlustfunktionen wie die Cross-Entropy-Loss-Funktion verwendet. Diese Funktion misst den Abstand zwischen den Vorhersagen des Modells und den tatsächlichen Klassenlabels der Daten. Sie wird häufig in Multi-Klassen-Klassifizierungsaufgaben eingesetzt und ermöglicht es dem Netzwerk, die Wahrscheinlichkeiten für jede Klasse zu modellieren und den Fehler basierend auf den Abweichungen von den richtigen Klassenlabels zu berechnen.

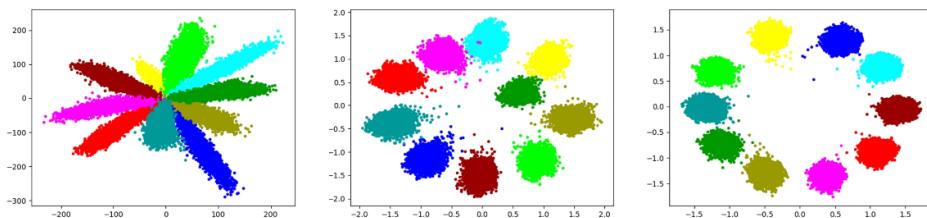


Abbildung 7.19: Links Softmax-2D, mittig Center-Loss, rechts PEDCC-loss.

Die Verlustfunktionen dienen als Grundlage für die Berechnung des Fehlersignals und die Aktualisierung der Gewichte im Netzwerk während des Trainingsprozesses. Durch die Minimierung der Verlustfunktion kann das CNN lernen, die optimalen Gewichtungen zu finden und die Genauigkeit der Vorhersagen zu verbessern. Es ist wichtig zu beachten, dass die Wahl der Verlustfunktion von der spezifischen Aufgabe abhängt. In einigen Fällen können andere Verlustfunktionen wie die Mean Squared Error (MSE) oder die Binary Cross-Entropy verwendet werden, je nach den Anforderungen der Aufgabe und der Art der Daten.

<sup>37</sup>Qiuyu ZHU u. a. [2019]. »A new loss function for CNN classifier based on predefined evenly-distributed class centroids«. In: *IEEE Access* 8, S. 10888–10895.

Algorithmus 7.1: Trainingsverlustverlauf: Unsere Loss-Funktion nimmt kontinuierlich ab, was auf eine verbesserte Modellleistung hinweist.

```

Training [ 0/10000] ..... 0 ) Loss= 1.1146412
Training [ 1/10000] ..... 1 ) Loss= 1.0547378
Training [ 2/10000] ..... 2 ) Loss= 1.1281569

Training [7078/10000] ..... 7078 ) Loss= 0.073117875
Training [7079/10000] ..... 7079 ) Loss= 0.05900609
Training [7080/10000] ..... 7080 ) Loss= 0.040167235

```

Die Verlustfunktionen spielen eine zentrale Rolle in der Architektur von CNNs, da sie den Unterschied zwischen den Vorhersagen und den tatsächlichen Werten der Daten messen. Sie dienen als Leitfaden für die Optimierung des Modells während des Trainingsprozesses und helfen dabei, die Genauigkeit der Vorhersagen zu verbessern. Die Auswahl der richtigen Verlustfunktion ist entscheidend, um die Leistung des CNNs zu maximieren und die gestellte Aufgabe erfolgreich zu lösen. Indem wir die Architektur und die verschiedenen Schichten eines Convolutional Neural Networks verstehen, können wir die Funktionsweise und die Fähigkeiten dieser leistungsstarken neuronalen Netzarchitektur besser erfassen.<sup>38,39</sup>

---

<sup>38</sup>Katarzyna JANOWSKA und Wojciech Marian CZARNECKI [2017]. »On loss functions for deep neural networks in classification«. In: *arXiv preprint arXiv:1702.05659*.

<sup>39</sup>Jiuxiang GU u. a. [2018]. »Recent advances in convolutional neural networks«. In: *Pattern recognition* 77, S. 354–377.

# Kapitel 8

## Datenverarbeitung (Data Preprocessing)

Im Folgenden wird die Datenverarbeitung, insbesondere das Data Preprocessing, in all ihren wichtigen Aspekten behandelt. Hierbei werden die Schritte der Datenbereinigung, Datennormalisierung und Datenaugmentierung erläutert. Diese grundlegenden Vorverarbeitungsschritte sind unverzichtbar, um hochwertige Daten zu erhalten, die für Analysen und Modellierungszwecke geeignet sind.

### 8.1 Was sind Daten? Natur der Daten

Eine präzise Definition von Daten ist von zentraler Bedeutung, um ein umfassendes Verständnis der Datenverarbeitung zu gewährleisten. Daten repräsentieren Informationen oder Fakten, die aus verschiedenen Quellen und in unterschiedlichen Formen gesammelt werden. Sie dienen als Grundlage (Ground truth) für Analysen und Modellbildung, um Muster und Erkenntnisse zu identifizieren, die zur Lösung von Problemen oder zur Gewinnung von Informationen dienen.

## 8.2 Die Bedeutung von qualitativ hochwertigen Daten

Im Bereich der Datenanalyse und des maschinellen Lernens ist die Qualität der Daten von entscheidender Bedeutung. Daten können fehlerhaft, inkonsistent oder unvollständig sein, da sie aus unterschiedlichen Quellen annotiert werden und in verschiedenen Formaten stammen. Die Resultate und Ergebnisse welche die Modelle zurückgegeben, stehen in direkter Korrelation zur Qualität der genutzten Trainingsdaten<sup>1</sup>. Um zuverlässige und aussagekräftige Ergebnisse zu erzielen, ist es unerlässlich, die Daten zu bereinigen und sicherzustellen, dass sie für Analysen und Modellierungszwecke geeignet sind. Dieser kritische Prozess wird als Datenbereinigung oder Data Cleaning bezeichnet.

Die Datenbereinigung beinhaltet den Prozess, fehlerhafte, inkonsistente oder unvollständige Daten aus einem Datensatz zu entfernen. Das Hauptziel besteht darin, die Datenqualität zu verbessern und sicherzustellen, dass sie für die weiteren Schritte der Datenanalyse und des maschinellen Lernens verwendet werden können. Eine sorgfältige Datenbereinigung ist von entscheidender Bedeutung, da fehlerhafte oder inkonsistente Daten zu fehlerhaften Analysen oder Modellen führen können<sup>2</sup>. Typische Schritte der Datenbereinigung umfassen die Entfernung von Duplikaten, die Behandlung von fehlenden Werten und die Korrektur von inkonsistenten Dateneinträgen. Duplikate können die Analyseergebnisse verfälschen und sollten daher entfernt werden. Fehlende Werte sind ein häufiges Problem in Datensätzen und müssen angemessen behandelt werden, entweder durch das Auffüllen der fehlenden Werte oder das Entfernen der betroffenen Datensätze. Inkonsistente Dateneinträge, wie zum Beispiel unterschiedliche Schreibweisen oder Formatisierungen, sollten korrigiert werden, um eine einheitliche und konsistente Datenbasis zu gewährleisten.

Die Bedeutung der richtigen Datenbereinigung kann nicht unterschätzt werden. Sie trägt maßgeblich zur Zuverlässigkeit und Genauigkeit der Analyseergebnisse bei und bildet

<sup>1</sup>Erhard RAHM, Hong Hai Do u. a. [2000]. »Data cleaning: Problems and current approaches«. In: *IEEE Data Eng. Bull.* 23.4, S. 3–13.

<sup>2</sup>Arthur D CHAPMAN [2005]. *Principles of data quality*. GBIF.

die Grundlage für fundierte Entscheidungen. Eine unzureichende Datenbereinigung kann zu falschen Schlussfolgerungen führen und das Vertrauen in die Analyse oder das Modell beeinträchtigen.

Insgesamt ist die Datenbereinigung ein wesentlicher Schritt in der Datenverarbeitung, der sicherstellt, dass die Daten von hoher Qualität sind und für Analysen und Modellierungszwecke geeignet sind.

## 8.3 Datennormalisierung (Data Normalization)

Die Datennormalisierung ist ein wichtiger Schritt in der Datenverarbeitung, welcher dazu dient, die Daten auf eine einheitliche Skala oder Verteilung zu transformieren. Oftmals enthalten Datensätze verschiedene Merkmale mit unterschiedlichen Skalen oder Einheiten. Durch die Normalisierung der Daten werden diese in einen bestimmten Wertebereich gebracht, um Probleme aufgrund von Skalenunterschieden oder unterschiedlichen Einheiten zu vermeiden. Der Prozess der Datennormalisierung spielt eine entscheidende Rolle in der Vorverarbeitung von Daten, da er die Grundlage für viele statistische Analysen und maschinelle Lernverfahren bildet.

Durch die Normalisierung der Daten können Muster und Zusammenhänge besser erkannt werden, da keine Verzerrungen aufgrund unterschiedlicher Skalen auftreten. Darüber hinaus kann die Normalisierung die Leistung von Algorithmen verbessern und die Konvergenz während des Trainingsprozesses beschleunigen. Es gibt verschiedene Methoden zur Datennormalisierung, die je nach Anwendungsfall und Art der Daten verwendet werden können.

**Min-Max-Normalisierung** Eine gängige Methode ist die Min-Max-Normalisierung<sup>3</sup>, bei der die Daten auf einen bestimmten Wertebereich transformiert werden. Dabei werden die Daten auf einen Wertebereich zwischen 0 und 1 skaliert, wobei der

<sup>3</sup>S. Gopal Krishna PATRO und Kishore Kumar SAHU [März 2015]. »Normalization: A Preprocessing Stage«. In: *IARJSET*, S. 20–22. DOI: 10.17148/iarjset.2015.2305. URL: <https://doi.org/10.17148/iarjset.2015.2305>.

kleinste Wert auf 0 und der größte Wert auf 1 abgebildet wird. Diese Methode eignet sich gut, wenn die genaue Verteilung der Daten nicht von großer Bedeutung ist.

**Z-Score-Normalisierung** Eine weitere Methode ist die Z-Score-Normalisierung<sup>4</sup>, bei der die Daten auf ihre Standardabweichung transformiert werden. Hierbei werden die Daten so verschoben und skaliert, dass sie einen Mittelwert von 0 und eine Standardabweichung von 1 haben. Diese Methode berücksichtigt die Verteilung der Daten und ist insbesondere dann nützlich, wenn die Daten einer Normalverteilung folgen.

**Einheitsvektor** Eine weitere gängige Methode ist die Skalierung auf den Einheitsvektor<sup>5</sup>, bei der die Daten auf eine Länge von 1 normiert werden. Diese Methode wird häufig in maschinellen Lernalgorithmen verwendet, bei denen die Richtung der Daten von Bedeutung ist, aber nicht die genaue Länge. Die Auswahl der geeigneten Normalisierungsmethode hängt von der Art der Daten und den Anforderungen des spezifischen Anwendungsfalls ab.

Insgesamt spielt die Datennormalisierung eine wichtige Rolle in der Datenverarbeitung, um die Daten auf eine einheitliche Skala oder Verteilung zu bringen. Es ist wichtig, die Daten vor der Normalisierung sorgfältig zu analysieren und zu verstehen, um die richtige Methode auszuwählen. Durch die Anwendung geeigneter Normalisierungsmethoden können Verzerrungen aufgrund von Skalenunterschieden oder unterschiedlichen Einheiten vermieden werden, was zu besseren Analysen und Modellen führt. Es ist entscheidend, die Daten sorgfältig zu analysieren und die richtige Normalisierungsmethode entsprechend den Anforderungen des Anwendungsfalls auszuwählen.

---

<sup>4</sup>Nanyi FEI u. a. [Okt. 2021]. »Z-Score Normalization, Hubness, and Few-Shot Learning«. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv48922.2021.00021. URL: <https://doi.org/10.1109/iccv48922.2021.00021>.

<sup>5</sup>Wei TIAN u. a. [Apr. 2013]. »Auto-normalization algorithm for robotic precision drilling system in aircraft component assembly«. In: *Chinese Journal of Aeronautics* 26.2, S. 495–500. DOI: 10.1016/j.cja.2013.02.029. URL: <https://doi.org/10.1016/j.cja.2013.02.029>.

## 8.4 Datenaugmentierung (Data Augmentation)

Die Datenaugmentierung bezieht sich auf den Prozess der künstlichen Erweiterung des Trainingsdatensatzes durch Anwendung von Transformationen oder Manipulationen auf die vorhandenen Daten. Sie spielt eine wichtige Rolle in der Datenverarbeitung, insbesondere bei begrenzten Datenmengen oder wenn das Modell eine größere Vielfalt an Beispielen lernen soll.

Durch die Datenaugmentierung wird die Varianz im Datensatz erhöht, indem verschiedene Transformationen auf die Daten angewendet werden. Dadurch erhält das Modell Zugang zu einer größeren Bandbreite an Datenmustern und kann robuster und vielfältiger werden<sup>6</sup>. Die Datenaugmentierung hilft dabei, Überanpassung<sup>7</sup> (Overfitting) zu vermeiden und die allgemeine Fähigkeit des Modells zur Verallgemeinerung auf neue Daten zu verbessern. Es gibt verschiedene Techniken der Datenaugmentierung, die je nach Anwendungsfall und Art der Daten angewendet werden können:

Ein etwas moderneres Verfahren ist das Zufällige Zuschneiden<sup>8</sup> (Random Cropping), bei dem zufällige Ausschnitte aus den Bildern genommen werden, um den Trainingsdatensatz zu erweitern. Dadurch wird das Modell in der Lage, Objekte in unterschiedlichen Positionen und Größen zu erkennen. Ähnlich dazu kann Zufälliges Löschen (random Erasing) von Bereichen aus Bildern angewendet werden, um Lückenhafte Daten im Datensatz zu nutzen. Das Zufällige Löschen (Random Erasing) von Bereichen aus Bildern ermöglicht es, Lücken in den Daten zu nutzen und die Modelle auf unvollständige oder teilweise verdeckte Objekte vorzubereiten. Es trägt zur Verbesserung der Robustheit und Allgemeinheit der

---

<sup>6</sup>Luke TAYLOR und Geoff NITSCHKE [Nov. 2018]. »Improving Deep Learning with Generic Data Augmentation«. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. doi: 10.1109/ssci.2018.8628742. URL: <https://doi.org/10.1109/ssci.2018.8628742>.

<sup>7</sup>Panissara THANAPOL u. a. [2020]. »Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition«. In: *2020-5th International Conference on Information Technology (InCIT)*. IEEE, S. 300–305.

<sup>8</sup>Zhun ZHONG u. a. [Apr. 2020]. »Random Erasing Data Augmentation«. In: *Proceedings of the AAAI Conference on Artificial Intelligence 34.07*, S. 13001–13008. doi: 10.1609/aaai.v34i07.7000. URL: <https://doi.org/10.1609/aaai.v34i07.7000>.

Bilderkennungsmodelle bei.

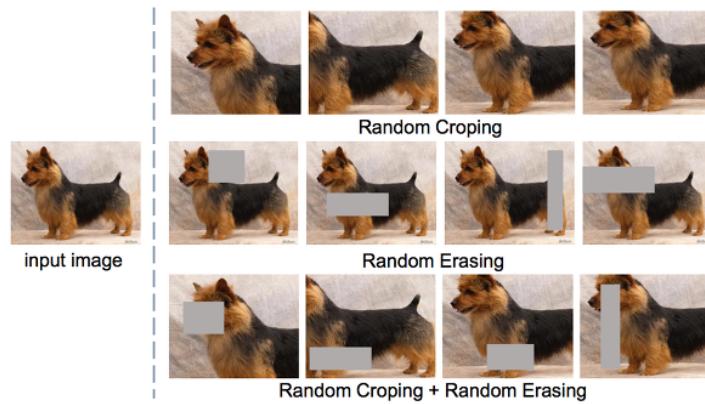


Abbildung 8.1: Verschiedene Methoden und Kombinationen zur Datenaugmentation.

Eine weitere Technik ist das Horizontale Spiegeln (Horizontal Flipping), bei dem die Bilder horizontal gespiegelt werden. Dadurch werden die Daten umgekehrt und das Modell lernt, Objekte aus verschiedenen Blickwinkeln zu erkennen. Diese Technik ist besonders nützlich, wenn die Orientierung der Objekte in den Bildern nicht von Bedeutung ist. Das Hinzufügen von Rauschen (Noise Addition) ist eine weitere Methode der Datenaugmentierung, bei der Rauschen in die Daten eingefügt wird. Dies kann helfen, das Modell widerstandsfähiger gegenüber Störungen zu machen und es auf den Umgang mit realen Daten vorzubereiten. Weitere Techniken umfassen das Skalieren, Drehen, Farbveränderungen und das Hinzufügen von Text oder Objekten zu den Bildern.

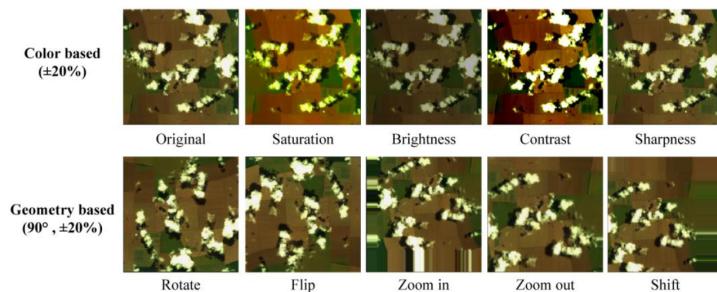


Abbildung 8.2: Klassische Methoden und Kombinationen zur Datenaugmentation.

Diese Methoden ermöglichen es, den Trainingsdatensatz zu diversifizieren und sicherzustellen, dass das Modell auf verschiedene Situationen und Variationen vorbereitet ist.

Insgesamt spielt die Datenaugmentierung eine bedeutende Rolle in der Datenverarbeitung, da sie die Qualität und Vielfalt der Trainingsdaten verbessert. Durch die Anwendung verschiedener Transformationen auf die Daten kann das Modell robustere und leistungsfähigere Modelle erstellen. Es ist wichtig, die richtigen Techniken der Datenaugmentierung entsprechend der Domäne und des Anwendungsfalls auszuwählen, um die Leistung des Modells zu verbessern.<sup>9</sup>

## 8.5 Fazit

Die richtige Vorverarbeitung der Daten ist entscheidend für den Erfolg von Modellen und Analysen. Durch Datenbereinigung, Datennormalisierung und Datenaugmentierung kann die Qualität und Aussagekraft der Daten verbessert werden, was zu besseren Ergebnissen führt. Die Datenbereinigung ermöglicht es, fehlerhafte oder unvollständige Daten zu entfernen oder zu korrigieren, um eine solide Grundlage für präzise und verlässliche Analysen und Modelle zu schaffen.

Obwohl in diesem Projekt der verwendete Datensatz bereits vorbereitet ist, ist es dennoch wichtig, zu prüfen, ob solch einer den Anforderungen des spezifischen Problems gerecht wird. Dies könnte die Überlegung beinhalten, ob ein vorgefertigter Datensatz verwendet werden kann oder ob Anpassungen erforderlich sind, um sicherzustellen, dass die Daten die relevanten Informationen für die Analysen enthalten.

---

<sup>9</sup>Conor SHORTEN und Taghi M. KHOSHGOFTAAR [2019]. »Data Augmentation for Deep Learning: A Comprehensive Survey«. In: *Journal of Big Data* 6.1, S. 1–48. doi: 10.1186/s40537-019-0197-0.

# Kapitel 9

## Architektur des DeepLabV3+ Modells mit ResNet-50 Backbone

Das DeepLabV3+ Modell ist eine leistungsstarke Architektur für die semantische Segmentation von Bildern. In diesem Kapitel werden wir uns näher mit der Architektur des Modells befassen, insbesondere mit dem ResNet-50 Backbone.

### 9.1 Backbone-Netzwerk

Das Backbone-Netzwerk ist für die Extraktion aussagekräftiger Merkmale aus dem Eingangsbild verantwortlich. In diesem Fall basiert das Backbone-Netzwerk auf der ResNet-50 Architektur. ResNet-50 ist ein CNN, das in verschiedenen Computer Vision Aufgaben herausragende Leistung erzielt hat. Es besteht aus mehreren Faltungsschichten, die in verschiedene Stufen gruppiert sind, wobei Residualverbindungen zwischen ihnen verwendet werden, um das Problem des verschwindenden Gradienten zu lösen. Diese Residualverbindungen ermöglichen es dem Netzwerk, effektiver zu lernen, indem sie Gradienten über Verbindungen mit geringerer Tiefe propagieren.

## 9.2 Prediction-Head

Der Prediction-Head nimmt die von dem Backbone-Netzwerk extrahierten Merkmale und generiert die endgültigen Vorhersagen. Im DeepLabV3+ Modell verwendet der Prediction-Head atrous (oder dilatierte) Faltungen mit unterschiedlichen Dilationsraten, um mehrskalige Informationen zu erfassen. Dadurch kann das Modell sowohl detaillierte lokale Informationen als auch Kontextinformationen berücksichtigen.

Das ursprüngliche DeepLabV3+ Modell wird auf dem ImageNet-Datensatz vorgenutzt, der Millionen gelabelten Bildern aus Tausenden Kategorien enthält. Dieses Vortraining hilft dem Modell, generische visuelle Merkmale zu erlernen, die für spezifische Aufgaben feinabgestimmt werden können.

Im bereitgestellten Code wird das vorgenutzte DeepLabV3+ Modell mit dem ResNet-50 Backbone geladen. Das bedeutet, dass das Backbone die anfänglichen Schichten von ResNet-50 umfasst, die für die Merkmalsextraktion verantwortlich sind. Die genauen Schichten und ihre Konfigurationen sind oft sehr komplex, in der Regel enthalten sie mehrere Faltungsschichten, Pooling-Schichten zur Verkleinerung der Auflösung und Residualverbindungen.

Die Modifikation im Code betrifft die letzte Schicht des Modells, den Klassifizierer. Im ursprünglichen DeepLabV3+ Modell handelt es sich dabei um eine 1x1-Faltungsschicht, die einen Tensor mit den Dimensionen [Batchgröße, Anzahl der Klassen, Höhe, Breite] erzeugt. Die Anzahl der Klassen im Originalmodell beträgt 21, da es auf dem COCO-Datensatz trainiert wurde, der Objekte aus 21 verschiedenen Kategorien enthält.

Im bereitgestellten Code wird die letzte Schicht jedoch durch eine andere 1x1-Faltungsschicht mit einer geänderten Anzahl von Ausgabekanälen ersetzt. Die ursprüngliche Anzahl von Ausgabekanälen beträgt 256, was der Anzahl der vom Backbone-Netzwerk erlernten Merkmale entspricht. In der modifizierten Version des Codes wird jedoch die Anzahl der Ausgabekanäle auf 3 gesetzt. Dies weist darauf hin, dass das Modell einen Tensor mit den Dimensionen [Batchgröße, 3, Höhe, Breite] ausgibt. Diese Änderung der Anzahl der Ausgabekanäle erfolgt, um das Modell für eine spezifische Aufgabe mit drei Klassen anstelle der ursprünglichen 21 Klassen anzupassen.

Insgesamt handelt es sich bei dem DeepLabV3+ Modell mit ResNet-50 Backbone um eine leistungsstarke Architektur für semantische Segmentierungsaufgaben. Durch die Modifikation der letzten Schicht passt der Code das Modell an, um Pixel in eine von drei Klassen zu klassifizieren.

**Code-Beispiel:**

```
Net = torchvision.models.segmentation.deeplabv3_resnet50(pretrained=True)
# Modell laden
Net.classifier[4] = torch.nn.Conv2d(
    256,
    3,
    kernel_size=(1, 1),
    stride=(1, 1)
) # Letzte Schicht auf 3 Klassen andern
Net = Net.to(device)

optimizer = torch.optim.Adam(params=Net.parameters(), lr=Learning_Rate)
# Adam-Optimizer erstellen
```

## 9.3 Laden des Modells

Der Code beginnt damit, das DeepLabV3+ Modell mit ResNet-50 Backbone mithilfe der torchvision-Bibliothek zu laden. DeepLabV3+ ist ein beliebtes Modell für die semantische Segmentierung, d.h., es weist jedem Pixel in einem Eingangsbild eine Klassenbezeichnung zu. Das ResNet-50 Backbone ist ein Typ von Faltungsneuronalem Netzwerk (CNN), das für seine Effektivität bei der Extraktion von Merkmalen aus Bildern bekannt ist.

## 9.4 Modifikation der letzten Schicht

Anschließend wird die letzte Schicht des geladenen Modells modifiziert. Im Originalmodell ist die letzte Schicht ein Klassifizierer, der eine Wahrscheinlichkeitsverteilung über 21 verschiedene Klassen (wie ‐Person‐, ‐Auto‐, ‐Hund‐, usw.) erzeugt. In diesem Code wird die letzte Schicht jedoch durch eine 1x1-Faltungsschicht ersetzt. Diese Modifikation ändert die Ausgabe des Modells so, dass eine Wahrscheinlichkeitsverteilung über 3 Klassen anstelle von 21 erzeugt wird. Die spezifischen Werte für die Kernelgröße und den Stride bestimmen das Verhalten der Faltung.

## 9.5 Verschieben des Modells auf ein Gerät

Das modifizierte Modell wird dann auf Hardware verschoben, welche zum Beispiel eine CPU oder eine GPU sein kann. Dieser Schritt stellt sicher, dass die Berechnungen, die das Modell durchführt, auf dem ausgewählten Gerät durchgeführt werden. Die Nutzung einer GPU kann die Schulung und Inferenz von Deep Learning Modellen erheblich beschleunigen.

## 9.6 Erstellen des Optimierers

Ein Optimierer ist ein Algorithmus, der die Parameter des Modells während des Trainingsprozesses anpasst, um die Verlustfunktion zu minimieren. In diesem Code wird ein Adam-Optimierer erstellt, der das Modellparameter (erhalten durch `Net.parameters()`) und die Lernrate (angegeben durch `Learning_Rate`) als Eingabe verwendet. Die Lernrate bestimmt die Schrittgröße, mit der der Optimierer die Modellparameter basierend auf den berechneten Gradienten während der Rückwärtspropagation aktualisiert.

Indem Sie dieser Architektur folgen, haben Sie ein modifiziertes DeepLabV3+ Modell mit ResNet-50 Backbone, das Bilder in eine von drei Klassen segmentieren kann. Die Modellparameter werden während des Trainings mit dem Adam-Optimierer und der angegebenen Lernrate optimiert.

# Kapitel 10

## Convolutional Neural Network Trainings Prozess

### 10.1 Was ist Training?

Bevor wir uns mit dem Trainingsprozess von Convolutional Neural Networks (CNNs) befassen, ist es sinnvoll, das Konzept des Trainings von neuronalen Netzwerken kurz aufzufrischen. Training bezieht sich auf den Prozess des Anpassens der Gewichte und Bias-Werte eines neuronalen Netzwerks, um eine bestimmte Aufgabe zu erlernen. Im Fall von CNNs besteht das Ziel darin, das Netzwerk auf eine bestimmte Bildklassifizierung oder ein anderes visuelles Erkennungsproblem vorzubereiten.

Der Trainingsprozess ist von entscheidender Bedeutung für die Leistungsfähigkeit von CNNs. Während des Trainings lernt das Netzwerk, relevante Merkmale aus den Trainingsdaten zu extrahieren und Muster zu erkennen. Durch die Optimierung der Gewichte und Bias-Werte kann das Netzwerk lernen, geeignete Entscheidungen zu treffen und präzise Vorhersagen zu treffen.

## 10.2 Trainingsprozess

Der Trainingsprozess eines CNNs lässt sich grob in verschiedene Schritte und Phasen unterteilen. Zunächst werden die Trainingsdaten geladen und gegebenenfalls vorverarbeitet, um eine optimale Eingabe für das Netzwerk zu gewährleisten. Dies kann Aufgaben wie das Skalieren der Bilder, das Normalisieren der Daten oder das Anwenden von Data Augmentation-Techniken umfassen. Während des Trainingsprozesses werden die Eingabedaten durch das CNN propagiert, wobei die Convolutional-Schichten und Pooling-Schichten verwendet werden, um Merkmale auf verschiedenen Abstraktionsebenen zu extrahieren. Aktivierungsfunktionen wie die ReLU-Funktion werden angewendet, um die Nichtlinearität des Netzwerks zu erhöhen.

Ein entscheidender Schritt ist die Berechnung des Verlusts (Loss), der den Unterschied zwischen den vom Netzwerk vorhergesagten Ausgaben und den tatsächlichen Labels misst. Die Wahl des geeigneten Verlustmaßes hängt von der spezifischen Aufgabe ab, beispielsweise der Kreuzentropie-Verlust für Klassifizierungsaufgaben. Um den Verlust zu minimieren und die Gewichte anzupassen, wird der Backpropagation-Algorithmus angewendet. Dabei werden die Gradienten der Verlustfunktion bezüglich der Gewichte berechnet und anschließend mittels eines Optimierungsverfahrens, wie zum Beispiel dem Stochastic Gradient Descent (SGD), verwendet, um die Gewichte in die richtige Richtung zu aktualisieren.

Der Trainingsprozess wird in der Regel über mehrere Epochen durchgeführt, wobei jede Epoche eine Durchlauf der gesamten Trainingsdaten umfasst. Dies ermöglicht es dem Netzwerk, schrittweise zu lernen und seine Leistung zu verbessern. Es ist auch üblich, das Netzwerk regelmäßig auf einem separaten Validierungsdatensatz zu testen, um die Überanpassung (Overfitting) zu vermeiden.

## 10.3 Stochastic Gradient Descent (SGD)

Der Stochastic Gradient Descent<sup>1</sup> (SGD) ist ein Optimierungsverfahren, das eine zentrale Rolle im Trainingsprozess von CNNs spielt. Im Gegensatz zum Gradientenabstiegsverfahren (Gradient Descent), bei dem der Verlust über den gesamten Trainingsdatensatz berechnet wird, verwendet SGD eine zufällige Teilmenge der Daten, um den Gradienten zu approximieren. Eine Vereinfachte Formel von SGD ist folgende Formel, mit  $x_{i+1}$  als neuer Wert,  $x_i$  der alte Wert und  $\alpha_i * d_i$  als berechnung des Absteigst ( $\alpha_i$  Schrittweite und  $d_i$  als steilster Abstieg):

$$x_{i+1} = x_i + \alpha_i * d_i \quad (10.1)$$

Der Einsatz von SGD hat verschiedene Vorteile. Erstens ermöglicht das Verfahren eine schnellere Berechnung des Gradienten, da nur eine Teilmenge der Daten betrachtet wird. Zweitens macht die zufällige Auswahl der Daten das Verfahren robuster gegenüber lokalen Minima, da das Netzwerk unterschiedliche Datenmuster im Verlauf des Trainingsprozesses betrachtet.

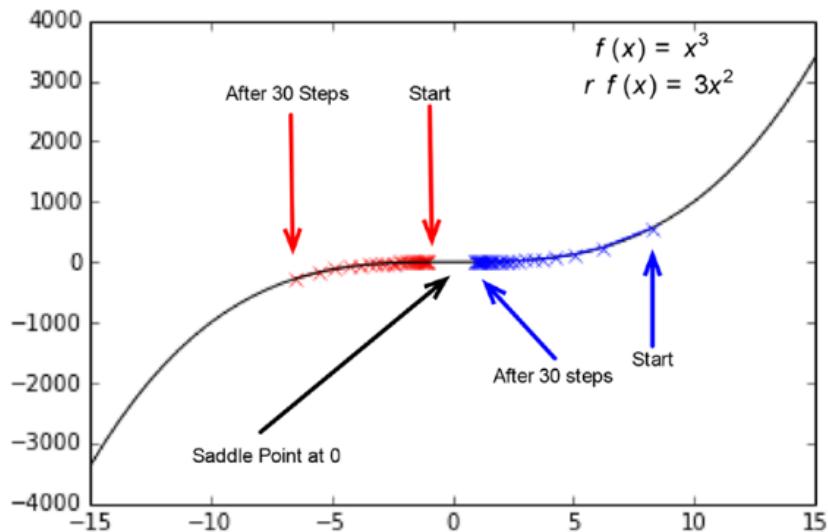


Abbildung 10.1: Beispiel von Stochastic Gradient Descent mit 30 Lernschritten.

---

<sup>1</sup>Shun-ichi AMARI [1993]. »Backpropagation and stochastic gradient descent method«. In: *Neurocomputing* 5.4-5, S. 185–196.

Die Anpassung der Lernrate ist ein wichtiger Aspekt im SGD-Algorithmus. Die Lernrate bestimmt die Größe der Aktualisierungen der Gewichte und beeinflusst somit die Konvergenzgeschwindigkeit des Netzwerks. Eine zu hohe Lernrate kann zu instabilen oder schlecht konvergierenden Lösungen führen, während eine zu niedrige Lernrate den Trainingsprozess verlangsamen kann. Es ist daher oft erforderlich, die Lernrate im Laufe des Trainings anzupassen, beispielsweise durch den Einsatz von Lernrateplanern oder adaptiven Methoden wie Adam.

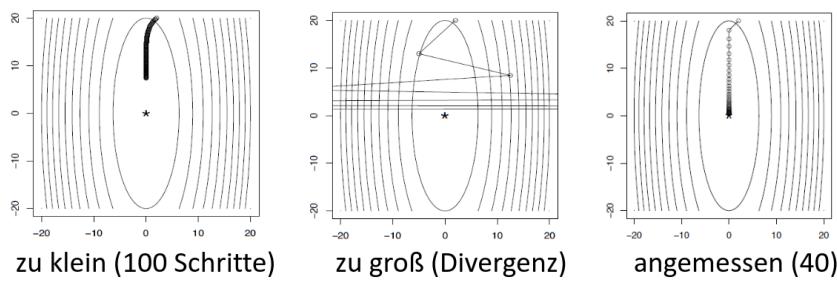


Abbildung 10.2: Gradientenabstiegsverfahren mit verschiedenen Schrittweiten

## 10.4 Backpropagation

Backpropagation<sup>2</sup> ist ein wesentlicher Bestandteil des Trainingsprozesses von neuronalen Netzwerken, einschließlich CNNs. Es handelt sich um ein Verfahren zur Berechnung der Gradienten der Verlustfunktion bezüglich der Gewichte des Netzwerks.

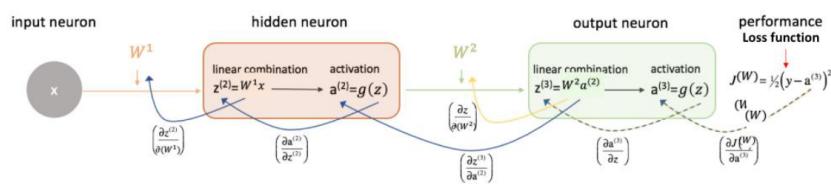


Abbildung 10.3: Backpropagation. Quelle: Vorlesung KI DHBW

Der Backpropagation-Algorithmus funktioniert, indem er die Fehlerinformationen

<sup>2</sup>Shun-ichi AMARI [1993]. »Backpropagation and stochastic gradient descent method«. In: *Neurocomputing* 5.4-5, S. 185–196.

vom Ausgabeneuron zurück durch das Netzwerk propagiert. Dabei werden die partiellen Ableitungen der Verlustfunktion nach den Gewichten in den Schichten berechnet. Dies ermöglicht es, den Gradienten des Verlusts bezüglich der Gewichte zu bestimmen und somit die Gewichte entsprechend anzupassen.

Dank der effizienten Berechnung des Backpropagation-Algorithmus ist es möglich, CNNs mit vielen Schichten zu trainieren. Die Gradienten werden schichtweise berechnet, wodurch eine effektive Ausbreitung der Fehler ermöglicht wird. Dieser Prozess des Gradientenabstiegs ermöglicht es dem Netzwerk, seine Gewichte so anzupassen, dass der Verlust minimiert wird.

## 10.5 Hyperparameter-Tuning

Hyperparameter<sup>3</sup> sind Parameter, die nicht direkt aus den Daten gelernt werden, sondern vor dem Trainingsprozess festgelegt werden müssen. Sie haben einen erheblichen Einfluss auf die Leistungsfähigkeit des CNNs und müssen sorgfältig ausgewählt werden.

Das Tuning von Hyperparametern beinhaltet die Suche nach den besten Werten für diese Parameter, um eine optimale Leistung des CNNs zu erzielen. Dies kann durch manuelles Ausprobieren verschiedener Hyperparameterkombinationen oder durch den Einsatz von automatisierten Methoden wie Grid Search oder Bayesian Optimization erfolgen<sup>4</sup>.

Beispiele für Hyperparameter sind die Lernrate, die Anzahl der Schichten und Filter im Netzwerk, die Größe des Mini-Batches, die Dropout-Rate und die Regularisierungspараметer. Das richtige Tuning dieser Hyperparameter kann dazu beitragen, eine bessere Generalisierungsfähigkeit des Netzwerks zu erreichen und Überanpassung zu vermeiden.

---

<sup>3</sup>Rafael G MANTOVANI u. a. [2016]. »Hyper-parameter tuning of a decision tree induction algorithm«. In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, S. 37–42.

<sup>4</sup>Shun-ichi AMARI [1993]. »Backpropagation and stochastic gradient descent method«. In: *Neurocomputing* 5.4-5, S. 185–196.

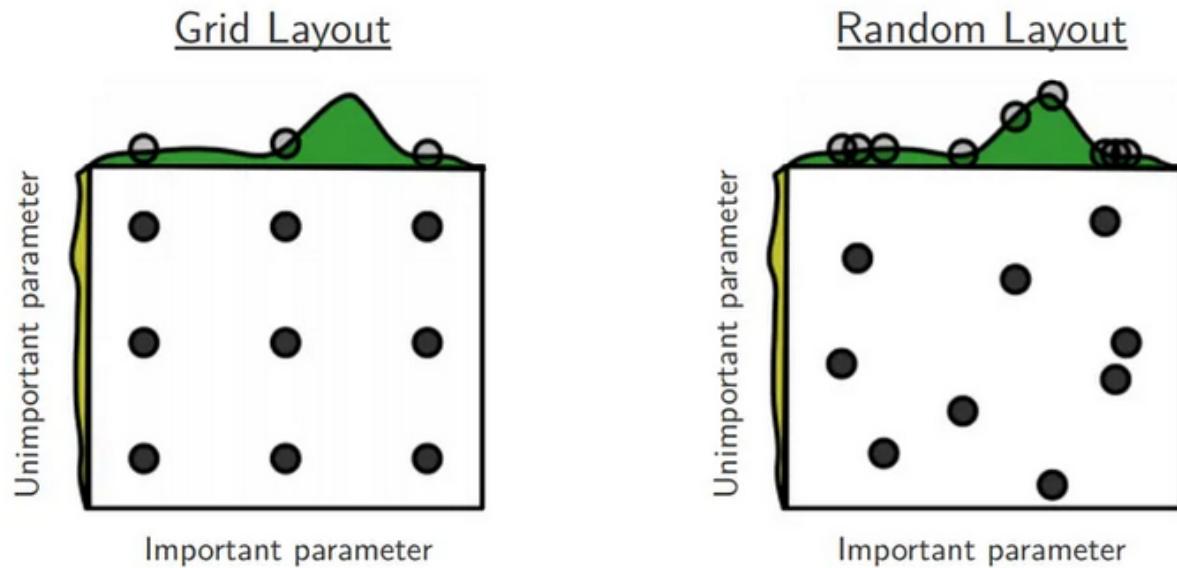


Abbildung 10.4: Hyperparameter tuning.

## 10.6 Regularisierungstechniken

Regularisierungstechniken<sup>5</sup> sind Methoden, die während des Trainingsprozesses angewendet werden, um die Überanpassung des Netzwerks an die Trainingsdaten zu reduzieren und die allgemeine Leistungsfähigkeit zu verbessern.

Eine gängige Regularisierungstechnik ist die L1- und L2-Regularisierung<sup>6</sup>, bei der ein Regularisierungsterm zur Verlustfunktion hinzugefügt wird, der die Gewichte des Netzwerks beeinflusst. Dies hilft, die Gewichte zu reduzieren und die Modellkomplexität zu verringern.

Ein weiteres Regularisierungsverfahren ist Dropout<sup>7</sup>, bei dem während des Trainings zufällig einige Neuronen deaktiviert werden. Dadurch wird das Netzwerk gezwungen,

<sup>5</sup>Golnaz GHIASI, Tsung-Yi LIN und Quoc V LE [2018]. »Dropblock: A regularization method for convolutional networks«. In: *Advances in neural information processing systems* 31.

<sup>6</sup>Belal IBRAHIM HAIRAB u. a. [2023]. »Anomaly Detection of Zero-Day Attacks Based on CNN and Regularization Techniques«. In: *Electronics* 12.3, S. 573.

<sup>7</sup>Alex LABACH, Hojjat SALEHINEJAD und Shahrokh VALAEE [2019]. *Survey of Dropout Methods for Deep Neural Networks*. arXiv: 1904.13310 [cs.NE].

redundante Merkmale zu lernen und erhöht die Robustheit gegenüber Überanpassung.

Data Augmentation ist eine weitere Regularisierungstechnik, bei der die Trainingsdaten künstlich erweitert werden, indem sie transformiert oder mit Rauschen versehen werden. Dies ermöglicht es dem Netzwerk, mehr Variationen der Daten zu sehen und generalisierbarere Merkmale zu lernen.

Die Anwendung von Regularisierungstechniken während des Trainingsprozesses trägt dazu bei, die Leistungsfähigkeit des CNNs zu verbessern und die Überanpassung an die Trainingsdaten zu reduzieren (siehe sec. 12.1.1).

# Kapitel 11

## Transfer Learning

### 11.1 Einführung in Transfer Learning

Transfer Learning ist eine leistungsstarke Technik im Bereich des Deep Learnings, die es uns ermöglicht, das Wissen von vorgefertigten Modellen zu nutzen und es auf neue Aufgaben anzuwenden. Dabei werden die erlernten Repräsentationen oder Parameter eines Modells wiederverwendet und auf eine andere verwandte Aufgabe übertragen. Diese Herangehensweise hat in den letzten Jahren aufgrund ihrer Fähigkeit, Zeit und Rechenressourcen zu sparen und dennoch hohe Leistung bei neuen Aufgaben zu erzielen, erhebliche Aufmerksamkeit und Beliebtheit erlangt. Die grundlegende Idee hinter Transfer Learning ist, dass Modelle, die auf großen und vielfältigen Datensätzen wie ImageNet trainiert wurden, allgemeine Merkmale gelernt haben, die für eine Vielzahl von visuellen Erkennungsaufgaben nützlich sind. Anstatt den Lernprozess für eine neue Aufgabe mit begrenzten Daten von Grund auf zu beginnen, können wir unser Modell mit den vorab trainierten Gewichten aus einer verwandten Aufgabe initialisieren. Dadurch besitzt das Modell bereits Wissen über niedrig eingestufte Merkmale, Formen und Muster, was in der neuen Aufgabe von Vorteil sein kann. Einer der Hauptvorteile von Transfer Learning besteht darin, das Problem des Datenmangels zu umgehen. Das Sammeln und Markieren großer Datenmengen für jede spezifische Aufgabe kann zeitaufwändig und kostspielig sein. Durch die Nutzung von Transfer Learning können wir jedoch auf die große Menge an markierten Daten zurückgreifen,

die für das Vortraining verfügbar sind, und somit den Bedarf an einem großen markierten Datensatz für die Ziel-Aufgabe verringern. Dies ist besonders vorteilhaft in Bereichen, in denen das Erlangen von markierten Daten herausfordernd oder nicht praktikabel ist. Ein weiterer Vorteil von Transfer Learning liegt in seiner Fähigkeit zur Generalisierung auf neuen Aufgaben. Indem wir Wissen von vortrainierten Modellen übertragen, nutzen wir effektiv die erlernten Repräsentationen, die aussagekräftige Merkmale erfassen. Diese Fähigkeit zur Generalisierung ermöglicht es dem Modell, auch mit begrenzten Trainingsdaten für die Ziel-Aufgabe gute Leistungen zu erbringen. Es trägt auch zur Reduzierung von Overfitting bei, da das Modell bereits aus einem vielfältigen Datensatz gelernt hat und robuste und diskriminierende Merkmale entwickelt hat. Darüber hinaus ermöglicht Transfer Learning, von der Expertise und den Forschungsanstrengungen zu profitieren, die in die Entwicklung vortrainierter Modelle investiert wurden. Viele fortschrittliche Modelle und Architekturen wurden auf groß angelegten Datensätzen vortrainiert und erreichen hohe Genauigkeit bei verschiedenen Benchmark-Aufgaben. Indem man diese vortrainierten Modelle als Ausgangspunkt nutzt, kann man ihre Architekturen und Merkmalsextraktoren verwenden und sie für eine spezifische Aufgabe abstimmen. Dadurch kann der Lernprozess beschleunigt werden.<sup>1,2</sup>

## 11.2 Pre-Trained Models

Ein wichtiger Bestandteil des Transfer Learnings sind vortrainierte Modelle. Diese vortrainierten Modelle sind neuronale Netzwerkmodelle, die auf großen Datensätzen trainiert wurden, in der Regel für eine andere Aufgabe als die aktuelle. Durch das Training auf umfangreichen Datensätzen haben diese Modelle allgemeine Merkmale und Muster erlernt, die für eine Vielzahl von verwandten Aufgaben nützlich sein können. Vortrainierte Modelle sind das Ergebnis umfangreicher Trainingsverfahren auf großen Rechenressourcen, um

---

<sup>1</sup>Sinno J. PAN und Qiang YANG [2010]. »A Survey on Transfer Learning«. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, S. 1345–1359. DOI: 10.1109/TKDE.2009.191.

<sup>2</sup>Jason YOSINSKI u. a. [2014]. »How Transferable are Features in Deep Neural Networks?« In: *Advances in Neural Information Processing Systems*, S. 3320–3328.

komplexe Muster in den Daten zu erkennen. Typischerweise werden vortrainierte Modelle auf großen Bilderkennungsdatensätzen wie ImageNet trainiert, die Millionen von Bildern mit verschiedenen Klassen umfassen. Während des Trainings lernen diese Modelle, verschiedene visuelle Merkmale wie Kanten, Formen, Texturen und Objekte zu erkennen und zu extrahieren. Durch diese umfangreiche Vorarbeit können vortrainierte Modelle als Ausgangspunkt für andere Aufgaben dienen, indem sie bereits gelernte Merkmale zur Verfügung stellen. Die Idee hinter der Verwendung vortrainierter Modelle im Transfer Learning besteht darin, das Wissen und die Merkmale, die in den vortrainierten Modellen enthalten sind, auf neue Aufgaben zu übertragen. Anstatt ein Modell von Grund auf neu zu trainieren, können wir ein vortrainiertes Modell verwenden und es an die spezifischen Anforderungen unserer Aufgabe anpassen. Da vortrainierte Modelle bereits eine gewisse Vorstellung von visuellen Merkmalen haben, können sie uns dabei unterstützen, auch mit begrenzten Daten gute Ergebnisse zu erzielen. Bei der Verwendung vortrainierter Modelle müssen wir jedoch beachten, dass die vortrainierte Aufgabe und die aktuelle Aufgabe zusammenpassen sollten. Wenn die vortrainierte Aufgabe ähnliche Merkmale oder Konzepte wie die aktuelle Aufgabe beinhaltet, ist die Wahrscheinlichkeit höher, dass das Transfer Learning erfolgreich ist. Zum Beispiel könnte ein vortrainiertes Modell, das auf Bilderkennung trainiert wurde, als Ausgangspunkt für eine Aufgabe der Objekterkennung dienen, da beide Aufgaben visuelle Merkmale nutzen.

### 11.3 Fine-tuning von vortrainierten Modellen

Die Feinabstimmung<sup>3</sup> (Fine-tuning) ist ein wichtiger Schritt im Transfer Learning, der es ermöglicht, ein vortrainiertes Modell für eine spezifische Aufgabe anzupassen. Bei der Feinabstimmung wird das vortrainierte Modell zunächst als Ausgangspunkt verwendet und anschließend werden die Gewichte des Modells mithilfe eines kleineren, auf die spezifische Aufgabe zugeschnittenen Datensatzes aktualisiert. Dieser Prozess erlaubt es dem Modell,

---

<sup>3</sup>Tagrid ALSHALALI und Darsana JOSYULA [2018]. »Fine-tuning of pre-trained deep learning models with extreme learning machine«. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, S. 469–473.

sich an die spezifischen Merkmale und Nuancen der neuen Aufgabe anzupassen<sup>4</sup>. Der erste Schritt bei der Feinabstimmung besteht darin, das vortrainierte Modell zu laden, das auf einer großen, allgemeinen Aufgabe trainiert wurde. Dieses Modell verfügt bereits über ein gewisses Verständnis von visuellen Merkmalen, das durch das Training auf umfangreichen Datensätzen erworben wurde. Anschließend werden die oberen Schichten des Modells, die für die spezifische Aufgabe weniger relevant sind, eingefroren, um zu verhindern, dass sie während des Trainings aktualisiert werden. Dies ermöglicht es uns, die vortrainierten Merkmale beizubehalten, während wir die Gewichte der unteren Schichten des Modells anpassen. Der nächste Schritt besteht darin, ein kleineres, auf die spezifische Aufgabe zugeschnittenes Datenset zu verwenden, um das Modell zu trainieren. Da die Anzahl der verfügbaren Daten möglicherweise begrenzt ist, ist es wichtig, Overfitting zu vermeiden und gleichzeitig das Modell an die spezifischen Merkmale der neuen Aufgabe anzupassen. Durch die Verwendung eines kleineren Datensatzes können wir die Rechenressourcen effizienter nutzen und den Trainingsprozess beschleunigen. Während des Trainingsprozesses werden die Gewichte der unteren Schichten des Modells aktualisiert, um die Merkmale der neuen Aufgabe besser zu erfassen. Da die oberen Schichten des Modells eingefroren sind, bleiben die bereits erlernten Merkmale erhalten, während die Gewichte der unteren Schichten an die neuen Daten angepasst werden. Dadurch kann das Modell spezifische Muster und Merkmale der neuen Aufgabe erlernen, während es gleichzeitig von den allgemeinen Merkmalen des vortrainierten Modells profitiert. Die Feinabstimmung bietet mehrere Vorteile<sup>5</sup>. Erstens ermöglicht sie eine schnellere Konvergenz, da das Modell bereits über eine gute initiale Gewichtung verfügt und somit weniger Trainingsiterationen benötigt werden. Zweitens kann die Leistung des Modells durch die Anpassung an die spezifischen Merkmale der neuen Aufgabe verbessert werden. Indem das Modell auf die Besonderheiten der neuen Aufgabe abgestimmt wird, kann es genauere Vorhersagen treffen und bessere Ergebnisse erzielen.

<sup>4</sup>Xu HAN u. a. [2021]. »Pre-trained models: Past, present and future«. In: *AI Open* 2, S. 225–250.

<sup>5</sup>Ahmadreza JEDDI, Mohammad Javad SHAFIEE und Alexander WONG [2020]. *A Simple Fine-tuning Is All You Need: Towards Robust Deep Learning Via Adversarial Fine-tuning*. arXiv: 2012.13628 [cs.CV].

## 11.4 Verwendung von vortrainierten Modellen

Eine alternative Methode des Transfer Learning besteht darin, das vortrainierte Modell als festen Merkmalsextraktor zu verwenden<sup>6</sup>. Dabei werden die Faltungsschichten des vortrainierten Modells genutzt, um Merkmale aus den Eingabedaten zu extrahieren, die anschließend einem neuen Klassifikator oder Regressor zugeführt werden. Dieser Ansatz bietet verschiedene Vorteile, wie zum Beispiel die Möglichkeit, vortrainierte Modelle auf kleineren Datensätzen einzusetzen. Bei dieser Methode werden die Gewichte des vortrainierten Modells beibehalten und die oberen Schichten eingefroren, um sicherzustellen, dass die bereits erlernten Merkmale nicht verändert werden.

Die Faltungsschichten des Modells dienen dann als Merkmalsextraktoren, die relevante Informationen aus den Eingabedaten extrahieren. Diese Merkmale werden anschließend als Eingabe für einen neuen Klassifikator oder Regressor verwendet, der auf die spezifische Aufgabe abgestimmt ist. Der Vorteil dieser Vorgehensweise liegt darin, dass das vortrainierte Modell bereits über ein umfangreiches Wissen verfügt, das auf großen Datensätzen trainiert wurde. Indem wir diese vortrainierten Merkmalsextraktoren verwenden, können wir von dem bereits erlernten Wissen profitieren, ohne das gesamte Modell neu trainieren zu müssen. Dies ist insbesondere dann vorteilhaft, wenn wir nur über einen begrenzten Datensatz verfügen, da das Training eines Modells von Grund auf mit wenigen Daten schwierig sein kann.

Ein weiterer Vorteil besteht darin, dass die Merkmalsextraktionsschichten des vortrainierten Modells bereits gute Ergebnisse bei der Erkennung allgemeiner Merkmale erzielen<sup>7</sup>. Dies ermöglicht es uns, auch auf kleineren Datensätzen aussagekräftige Merkmale zu extrahieren und sie als Eingabe für den Klassifikator oder Regressor zu verwenden. Somit können wir die Vorteile des vortrainierten Modells nutzen, um bessere Ergebnisse

<sup>6</sup>Souleyman CHAIB u. a. [2017]. »Deep feature extraction and combination for remote sensing image classification based on pre-trained CNN models«. In: *Ninth international conference on digital image processing (ICDIP 2017)*. Bd. 10420. SPIE, S. 712–716.

<sup>7</sup>Kaichao YOU u. a. [2021]. »Logme: Practical assessment of pre-trained models for transfer learning«. In: *International Conference on Machine Learning*. PMLR, S. 12133–12143.

auf unseren spezifischen Aufgaben zu erzielen. Es ist wichtig zu beachten, dass die Verwendung vortrainierter Modelle als Merkmalsextraktoren bestimmte Einschränkungen hat. Da die oberen Schichten des Modells eingefroren<sup>8</sup> sind, können sie nicht auf die spezifischen Merkmale der neuen Aufgabe angepasst werden. Daher ist diese Methode besonders geeignet, wenn die Merkmale der neuen Aufgabe bereits in den vortrainierten Schichten erfasst werden können. In solchen Fällen bietet die Nutzung der vortrainierten Merkmalsextraktoren eine effiziente Möglichkeit, um genaue Vorhersagen zu treffen.

## 11.5 Anwendung von DeepLabV3 ResNet50

In diesem Abschnitt wird die konkrete Umsetzung des Transfer Learning in unserem Projekt durch die Verwendung des DeepLabV3<sup>9</sup> ResNet50-Modells<sup>10</sup> erläutert. DeepLabV3 ist eine hochmoderne Architektur, die für semantische Segmentierungsaufgaben entwickelt wurde und für ihre herausragende Leistung in der Bildverarbeitung und Szenenanalyse bekannt ist. Das ResNet50 dient als grundlegende Netzwerkstruktur innerhalb des DeepLabV3-Frameworks und nutzt Residual-Lernen, um das Training tiefer faltungs-basierter neuronaler Netzwerke zu erleichtern. Die Entscheidung, das DeepLabV3 ResNet50-Modell in unserem Projekt zu verwenden, basierte auf einer Vielzahl von Überlegungen<sup>11</sup>. Die Architektur des Modells hat sich insbesondere im Bereich der semantischen Segmentierung als äußerst leistungsstark erwiesen, was eine wesentliche Aufgabe in unseren Projektzielen darstellt. Durch den Einsatz des DeepLabV3 ResNet50-Modells kann man das umfangreiche Training und die erlernten Repräsentationen nutzen, die das Modell auf großen

---

<sup>8</sup>Hassan ALHuzali, Tianlin Zhang und Sophia Ananiadou [2021]. »Predicting Sign of Depression via Using Frozen Pre-trained Models and Random Forest Classifier.« In: *CLEF (Working Notes)*, S. 888–896.

<sup>9</sup>Liang-Chieh Chen u. a. [2017]. *Rethinking Atrous Convolution for Semantic Image Segmentation*. arXiv: 1706.05587 [cs.CV].

<sup>10</sup>Kaiming He u. a. [2016]. »Deep residual learning for image recognition.« In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 770–778.

<sup>11</sup>Basierend auf einer umfangreichen Prüfung verschiedener Faktoren wurde eine fundierte und vielfältig informierte Entscheidung getroffen. Unter Berücksichtigung des Videos <https://www.youtube.com/watch?v=aTYVdK3GNxI> wurden geschickt inspirierende Aspekte in die Entscheidungsfindung eingebunden.

Datensätzen erfasst hat. Dadurch kann das Netzwerk feine semantische Details in unserem spezifischen Anwendungsbereich erkennen. Die Verwendung eines vortrainierten Modells wie DeepLabV3 ResNet50 bietet eine Vielzahl von Vorteilen. Ein bemerkenswerter Vorteil besteht darin, dass man das Wissen, das durch vorherige Modelle erlangt wurde, in das Projekt übertragen kann, ohne ein neuronales Netzwerk von Grund auf trainieren zu müssen. Das vortrainierte Modell enthält eine Vielzahl von allgemeinen Merkmalen und Mustern, die während der umfangreichen Exposition des Modells gegenüber vielfältigen visuellen Daten während seiner Trainingsphase erlernt wurden. Durch die Nutzung dieser erlernten Merkmale kann man die Entwicklungsdauer des Projekts verkürzen und die Rechenbelastung, die mit dem Training eines Modells von Grund auf verbunden ist, verringern.

Darüber hinaus bietet das DeepLabV3 ResNet50-Modell eine solide Grundlage für die Feinabstimmung, die es ermöglicht, das Netzwerk an spezifische Aufgaben anzupassen. Bei der Feinabstimmung werden die vortrainierten Gewichte des Netzwerks beibehalten und an die Feinheiten des Ziel-Datensatzes angepasst. Dieser Prozess ermöglicht eine schnellere Konvergenz des Lernprozesses des Modells, verkürzt die Gesamttrainingszeit und verbessert die endgültige Leistung bei der Segmentierungsaufgabe. Die Feinabstimmung ermöglicht es auch, das im vortrainierten Modell vorhandene Wissen an die spezifische Anwendungsdomäne anzupassen. Dabei können die allgemeinen Merkmale, die vom Modell extrahiert wurden, verfeinert und an die Feinheiten der Zielaufgabe angepasst werden. Zusammenfassend bietet die Einbindung des DeepLabV3 ResNet50-Modells in das Projekt mittels Transfer Learning eine solide Grundlage, um eine präzise semantische Segmentierung zu erreichen. Indem man auf das vorhandene Wissen des Modells zurückgreift und die Vorteile der Feinabstimmung nutzt, kann man die Fachkenntnisse von DeepLabV3 ResNet50 nutzen und sie effektiv auf die einzigartige Anwendungsdomäne anpassen. Diese Vorgehensweise beschleunigt nicht nur den Entwicklungsprozess des Projekts, sondern führt auch zu einer hochwertigen Lösung, indem das Modell von seinem umfangreichen Training mit vielfältigen visuellen Daten profitiert.

Ein entscheidender Aspekt bei der Anwendung des DeepLabV3 ResNet50-Modells

besteht darin, dass die vortrainierten Gewichte des Modells eingefroren bleiben, um die extrahierten Merkmale beizubehalten. Die Faltungsschichten des Modells werden verwendet, um Merkmale aus den Eingabedaten zu extrahieren, die dann in einen neuen Klassifizierer oder Regressor eingespeist werden können. Dieser Ansatz ermöglicht es, die Vorteile des vortrainierten Modells als leistungsstarken Feature-Extraktor zu nutzen, während man gleichzeitig einen neuen, auf die spezifische Aufgabe abgestimmten Klassifizierer oder Regressor trainiert. Die Verwendung von vortrainierten Modellen als Feature-Extraktoren bietet eine Reihe von Vorteilen<sup>12</sup>. Insbesondere kann man auf kleineren Datensätzen arbeiten, da die vortrainierten Modelle bereits allgemeine Merkmale gelernt haben, die auf verschiedene Aufgaben übertragbar sind. Dadurch wird der Bedarf an großen Trainingsdatensätzen reduziert, was sowohl die Datenbeschaffung als auch die Rechenressourcen erleichtert. Darüber hinaus ermöglicht die Verwendung von vortrainierten Modellen als Feature-Extraktoren eine schnellere Entwicklung von Modellen, da der Schwerpunkt auf der Anpassung des Klassifizierers oder Regressors liegt, anstatt das gesamte Modell von Grund auf neu zu trainieren.

Insgesamt bietet die Anwendung des DeepLabV3 ResNet50-Modells als Feature-Extraktor eine effektive Methode des Transfer Learning, um hochwertige Ergebnisse in der semantischen Segmentierungsaufgabe zu erzielen. Durch die Kombination der Stärken des vortrainierten Modells und der Anpassung an die spezifische Anwendungsdomäne kann man die Effizienz verbessern und gleichzeitig genaue und zuverlässige Ergebnisse erzielen.

---

<sup>12</sup>Souleyman CHAIB u. a. [2017]. »Deep feature extraction and combination for remote sensing image classification based on pre-trained CNN models«. In: *Ninth international conference on digital image processing (ICDIP 2017)*. Bd. 10420. SPIE, S. 712–716.

# Kapitel 12

## Herausforderungen und Lösungen im Bereich Deep Learning

### 12.1 Overfitting und Underfitting

Beim maschinellen Lernen, insbesondere im Bereich des Deep Learning, treten häufig die Phänomene des Overfittings und Underfittings auf. Diese Konzepte spielen eine entscheidende Rolle bei der Modellierung und Bewertung neuronaler Netze. In diesem Kapitel werden die Herausforderungen des Overfittings und Underfittings erläutert sowie Lösungsansätze<sup>1</sup> vorgestellt, um diese Probleme zu identifizieren und zu mindern.

#### 12.1.1 Overfitting

Overfitting tritt auf, wenn ein Modell zu stark an die Trainingsdaten angepasst ist und die Fähigkeit verliert, auf neuen, unbekannten Daten zu generalisieren. Das Modell erlernt spezifische Muster und Rauschen in den Trainingsdaten, die nicht repräsentativ für die gesamte Datenmenge sind. Dies führt zu einer übermäßigen Komplexität des Modells und einer schlechten Leistung auf neuen Daten. Die Ursachen für Overfitting können

---

<sup>1</sup>Kai ZHANG u. a. [2015]. »Learning a single convolutional super-resolution network for multiple degradations«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 3262–3270. DOI: 10.1109/CVPR.2015.7298958.

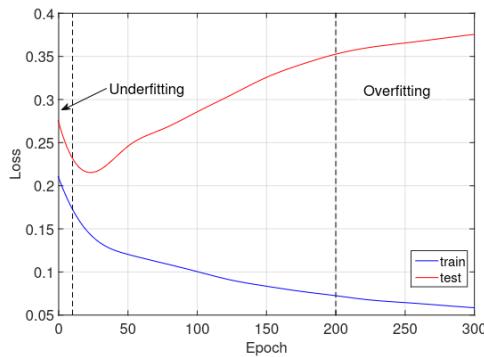


Abbildung 12.1: Beispiel von Under- und Overfitting Probleme

vielfältig sein. Eine mögliche Ursache ist eine zu große Anzahl von Parametern im Modell, die zu einer Überanpassung an die Trainingsdaten führen. Weitere Gründe können unzureichende Datenmengen, das Fehlen von Regularisierungstechniken oder das Vorhandensein von Ausreißern in den Trainingsdaten sein. Die Auswirkungen von Overfitting sind vielfältig und können zu suboptimalen Modellleistungen führen<sup>2</sup>. Das Modell kann hohe Trainingsgenauigkeit aufweisen, jedoch eine niedrige Testgenauigkeit erzielen, da es nicht in der Lage ist, das Gelernte auf neue Daten zu verallgemeinern. Dies kann zu falschen Vorhersagen und einer geringen Robustheit des Modells führen. Um Overfitting zu identifizieren und zu vermeiden, stehen verschiedene Techniken zur Verfügung. Eine häufig verwendete Methode ist die Regularisierung, die die Komplexität des Modells reduziert und eine bessere Generalisierung ermöglicht. Beispiele für Regularisierungsmethoden sind L1- und L2-Regularisierung, die den Verlustfunktionen Strafterme hinzufügen, um große Gewichtungen zu minimieren. Dadurch wird eine bessere Kontrolle über die Modellkomplexität erreicht. Darüber hinaus kann das frühzeitige Beenden des Trainingsprozesses, auch bekannt als Early<sup>3</sup>, dazu beitragen, Overfitting zu verhindern. Hierbei wird das Training gestoppt, wenn die Leistung auf einem Validierungsdatensatz nicht mehr verbessert wird. Dadurch wird verhindert, dass das Modell zu lange trainiert wird und sich auf spezifische

<sup>2</sup>Tomaso POGGIO u. a. [2018]. *Theory of Deep Learning III: explaining the non-overfitting puzzle*. arXiv: 1801.00173 [cs.LG].

<sup>3</sup>Yingbin BAI u. a. [2021]. »Understanding and improving early stopping for learning with noisy labels«. In: *Advances in Neural Information Processing Systems* 34, S. 24392–24403.

Muster der Trainingsdaten spezialisiert.

### 12.1.2 Underfitting

Underfitting tritt auf, wenn das Modell nicht in der Lage ist, die Muster und Zusammenhänge in den Daten zu erfassen. Es handelt sich um das Gegenteil von Overfitting, bei dem das Modell zu einfach oder zu grob ist, um die Komplexität der Daten angemessen zu modellieren. Infolgedessen kann das Modell weder die Trainingsdaten noch neue Daten gut generalisieren. Die Gründe für Underfitting können verschiedene Ursachen haben. Eine mögliche Ursache ist die Verwendung eines zu einfachen Modells, das nicht in der Lage ist, die inhärente Komplexität der Daten abzubilden. Dies kann auch auf eine unzureichende Anzahl von Parametern oder Merkmalen im Modell zurückzuführen sein. Darüber hinaus kann ein Mangel an relevanten Merkmalen oder eine unangemessene Datenpräparation zu Underfitting führen. Die Auswirkungen von Underfitting sind ähnlich wie bei Overfitting ungünstig. Das Modell zeigt sowohl eine niedrige Trainingsgenauigkeit als auch eine niedrige Testgenauigkeit, da es nicht in der Lage ist, die Muster in den Daten adäquat zu erfassen. Das Modell verliert an Informationsgehalt und seine Vorhersagen sind ungenau und wenig zuverlässig. Um Underfitting zu beheben, stehen verschiedene Ansätze zur Verfügung. Ein erster Ansatz besteht darin, die Komplexität des Modells zu erhöhen, indem beispielsweise die Anzahl der Parameter oder die Netzwerkarchitektur erhöht wird. Dies ermöglicht es dem Modell, die zugrundeliegenden Muster in den Daten besser zu erfassen. Dropout ist eine Technik zur Bekämpfung von Underfitting<sup>4</sup>. Hierbei werden zufällig einige Neuronen während des Trainings deaktiviert, um eine gewisse Redundanz im Modell zu erzeugen. Dies führt zu einer Verbesserung der Generalisierungsfähigkeit des Modells und verringert die Abhängigkeit einzelner Neuronen. Eine weitere Möglichkeit, Underfitting zu reduzieren, besteht darin, zusätzliche relevante Merkmale in das Modell einzubeziehen oder vorhandene Merkmale besser zu transformieren. Dies kann dazu beitragen, die Modellkapazität zu erhöhen und eine bessere Passung der Daten zu ermöglichen. Die Anwendung von Ensemble-Methoden, bei denen mehrere Modelle

---

<sup>4</sup>Zhuang LIU u. a. [2023]. *Dropout Reduces Underfitting*. arXiv: 2303.01500 [cs.LG].

kombiniert werden, kann ebenfalls dazu beitragen, Underfitting zu bekämpfen. Durch die Kombination der Vorhersagen mehrerer Modelle kann die Gesamtleistung verbessert und eine bessere Anpassung an die Daten erzielt werden. Es ist wichtig, die richtige Balance zwischen Overfitting und Underfitting zu finden, um ein optimales Modell zu erhalten. Dies erfordert eine sorgfältige Modellauswahl und Hyperparameter-Optimierung. Eine gute Modellbewertung und Validierung auf unabhängigen Testdatensätzen sind ebenfalls entscheidend, um die Qualität des Modells zu gewährleisten.

## 12.2 Vanishing and Exploding Gradients

Die Probleme der verschwindenden (vanishing) und explodierenden (exploding) Gradienten stellen eine Herausforderung beim Training von tiefen neuronalen Netzen dar. Diese Phänomene können zu einer schlechten Modellleistung und einer ineffizienten Lernkonvergenz führen. In diesem Kapitel werden die Ursachen dieser Probleme sowie Lösungsansätze zur Bewältigung der vanishing und exploding Gradients diskutiert.

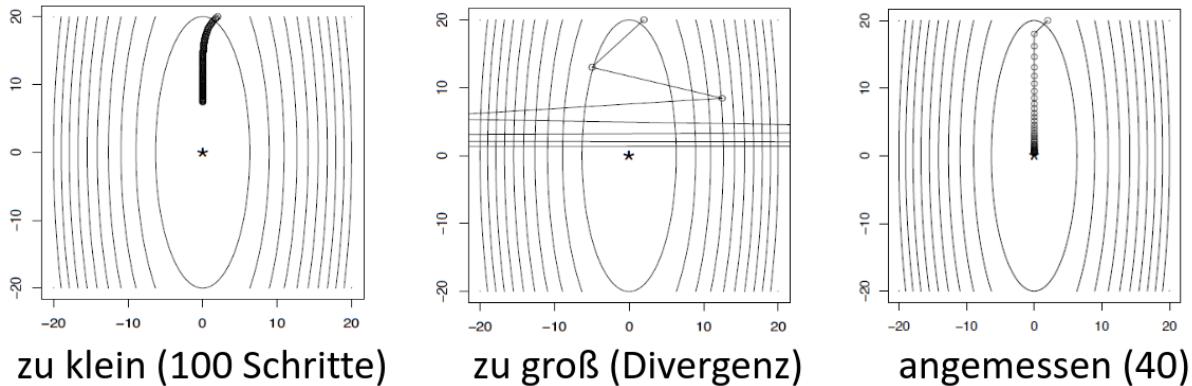


Abbildung 12.2: Beispiel von Vanishing- und Exploding Gradient Probleme

### 12.2.1 Ursachen der vanishing Gradients

Die vanishing Gradients entstehen, wenn die Gradienten während des Backpropagation-Verfahrens in tiefen neuronalen Netzen (deep neural networks) mit jedem weiteren Layer

immer kleiner werden. Dies tritt auf, wenn die Ableitungen der Aktivierungsfunktionen, die zur Berechnung der Gradienten verwendet werden, Werte kleiner als 1 haben. Die Multiplikation kleiner Zahlen in aufeinanderfolgenden Schichten führt zu exponentiell abnehmenden Gradienten. Ein weiterer Faktor, der vanishing Gradients verursachen kann, ist die Tiefe des neuronalen Netzes. Je tiefer das Netzwerk ist, desto mehr Schichten müssen durchlaufen werden, um den Fehler zurückzupropagieren. Da die Gradienten mit jeder Schicht multipliziert werden, wird ihre Größe exponentiell kleiner, was zu einem Verschwinden der Gradienten in den frühen Schichten führt.

### 12.2.2 Ursachen der exploding Gradients

Im Gegensatz zu den vanishing Gradients treten bei den exploding Gradients Probleme auf, wenn die Gradienten während des Backpropagation-Verfahrens in tiefen neuronalen Netzen immer größer werden. Dies geschieht, wenn die Ableitungen der Aktivierungsfunktionen Werte größer als 1 aufweisen. Die Multiplikation großer Zahlen in aufeinanderfolgenden Schichten führt zu exponentiell ansteigenden Gradienten. Die exploding Gradients können auch durch eine falsche Wahl der Lernrate verursacht werden. Wenn die Lernrate zu hoch gewählt wird, können die Gradienten im Laufe des Trainings stark ansteigen und die Gewichte des Modells unkontrolliert verändern.

### 12.2.3 Negative Auswirkungen der vanishing und exploding Gradients

Die vanishing und exploding Gradients können erhebliche negative Auswirkungen auf den Trainingsprozess und die Modellleistung haben. Bei den vanishing Gradients kann die Lernkonvergenz verlangsamt werden, da die Gradienten in den frühen Schichten zu klein werden, um signifikante Gewichtsaktualisierungen zu bewirken. Dadurch kann das Modell Schwierigkeiten haben, komplexe Muster und Zusammenhänge in den Daten zu erfassen. Bei den exploding Gradients können instabile Gewichtsaktualisierungen auftreten, die zu einer übermäßigen Anpassung des Modells an die Trainingsdaten führen können. Dies

kann zu einem Verlust der Generalisierungsfähigkeit des Modells führen, da es zu stark auf die Trainingsdaten spezialisiert ist.

### 12.2.4 Lösungsansätze für vanishing Gradients

Um vanishing Gradients zu adressieren, stehen verschiedene Lösungsansätze zur Verfügung. Eine Methode besteht darin, spezielle Initialisierungstechniken für die Gewichte zu verwenden, die eine angemessene Skalierung ermöglichen. Hierbei haben sich insbesondere die Xavier-Initialisierung<sup>5</sup> und die He-Initialisierung<sup>6,7</sup> als effektiv erwiesen. Diese Techniken berücksichtigen die Anzahl der Eingangs- und Ausgangsneuronen eines Layers und stellen sicher, dass die Gewichte in einem geeigneten Bereich initialisiert werden, um das Verschwinden der Gradienten zu verhindern.

Eine weitere Lösung für vanishing Gradients besteht in der Verwendung geeigneter Aktivierungsfunktionen. Die Rectified Linear Unit (ReLU) und die Leaky ReLU-Funktion<sup>8</sup> haben sich in der Praxis als wirksame Wahl erwiesen, um das Problem der verschwindenden Gradienten anzugehen. Diese Aktivierungsfunktionen ermöglichen eine nicht-lineare Transformation der Eingaben und verhindern das Sättigen der Gradienten in den tiefen Schichten des Netzwerks.

---

<sup>5</sup>Xavier GLOROT und Yoshua BENGIO [2010]. »Understanding the difficulty of training deep feedforward neural networks«. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Hrsg. von Yee Whye TEH und Mike TITTERINGTON. Bd. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, S. 249–256. URL: <https://proceedings.mlr.press/v9/glorot10a.html>.

<sup>6</sup>Jo SCHLEMPER u. a. [2017]. »A deep cascade of convolutional neural networks for MR image reconstruction«. In: *Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings* 25. Springer, S. 647–658.

<sup>7</sup>Kaiming HE u. a. [2015]. »Delving deep into rectifiers: Surpassing human-level performance on imagenet classification«. In: *Proceedings of the IEEE international conference on computer vision*, S. 1026–1034.

<sup>8</sup>Jiajun XU, Xudong JIANG und Zulin ZHANG [2017]. »Efficient and accurate image super-resolution via recurrent mixture density network«. In: *IEEE Transactions on Image Processing* 26.7, S. 3187–3201. DOI: 10.1109/TIP.2017.2699923.

### 12.2.5 Lösungsansätze für exploding Gradients

Um exploding Gradients zu bewältigen, ist eine gängige Methode das Gradient Clipping<sup>9</sup>. Dabei wird die Größe der Gradienten während des Trainings begrenzt, um zu verhindern, dass sie einen bestimmten Schwellenwert überschreiten. Durch die Anwendung des Gradient Clippings wird die Stabilität des Trainingsprozesses verbessert und das Risiko von instabilen Gewichtsaktualisierungen reduziert. Eine weitere Möglichkeit, exploding Gradients zu vermeiden, besteht darin, die Lernrate anzupassen<sup>10</sup>. Es kann hilfreich sein, adaptive Optimierungsalgorithmen wie den Adam-Optimizer zu verwenden, der automatisch die Lernrate für jedes Gewicht anpasst. Durch die dynamische Anpassung der Lernrate werden große Gewichtsaktualisierungen vermieden und die Konvergenz des Modells verbessert.

## 12.3 Gradient Descent Optimization

Der Abschnitt "Gradient Descent Optimization" behandelt die Herausforderungen und Lösungen im Bereich der Optimierungsalgorithmen<sup>11</sup> für das Gradientenabstiegsverfahren, die in der Praxis bei der Schulung tiefer neuronaler Netze häufig eingesetzt werden. Dieser Abschnitt bietet einen Überblick über die gängigsten Optimierungsalgorithmen und deren Auswirkungen auf die Effizienz und Konvergenzgeschwindigkeit des Trainingsprozesses.

---

<sup>9</sup>Xiangyi CHEN, Steven Z WU und Mingyi HONG [2020]. »Understanding gradient clipping in private sgd: A geometric perspective«. In: *Advances in Neural Information Processing Systems* 33, S. 13773–13782.

<sup>10</sup>Yuchuan QIAO u. a. [2015]. »Fast automatic step size estimation for gradient descent optimization of image registration«. In: *IEEE transactions on medical imaging* 35.2, S. 391–403.

<sup>11</sup>Sebastian RUDER [2016]. »An overview of gradient descent optimization algorithms«. In: *arXiv preprint arXiv:1609.04747*.

### 12.3.1 Standard-Gradientenabstieg

Der Standard-Gradientenabstieg<sup>12</sup> ist der grundlegende Optimierungsalgorithmus, der zur Aktualisierung der Gewichte in neuronalen Netzen verwendet wird. Bei diesem Verfahren wird der Gradient der Verlustfunktion für jeden einzelnen Trainingsdatensatz berechnet und die Gewichte entsprechend angepasst. Obwohl der Standard-Gradientenabstieg einfach zu implementieren ist, hat er einige Herausforderungen. Ein Problem des Standard-Gradientenabstiegs ist seine langsame Konvergenzgeschwindigkeit. Da der Gradient für jeden Trainingsdatensatz separat berechnet wird, kann dies zu einer ineffizienten Aktualisierung der Gewichte führen. Darüber hinaus besteht die Gefahr von lokalen Minima oder Plateaus, in denen das Modell stecken bleiben kann und Schwierigkeiten hat, sich weiter zu verbessern.

### 12.3.2 Stochastischer Gradientenabstieg (SGD)

Eine verbreitete Verbesserung des Standard-Gradientenabstiegs ist der stochastische Gradientenabstieg (SGD). Beim SGD wird der Gradient nicht für jeden einzelnen Trainingsdatensatz berechnet, sondern nur für eine zufällig ausgewählte Teilmenge der Daten, auch als Mini-Batch bezeichnet<sup>13</sup>. Dadurch wird der Trainingsprozess beschleunigt, da weniger Berechnungen erforderlich sind. Der SGD bietet jedoch auch Herausforderungen. Er kann zu einer instabilen Aktualisierung der Gewichte führen, da die zufällige Auswahl der Mini-Batches zu einer großen Varianz in den Gradientenschätzungen führen kann. Dies kann zu einer schlechten Konvergenz und einer inkonsistenten Modellleistung führen.

<sup>12</sup>Anjar WANTO u. a. [2018]. »Analysis of Standard Gradient Descent with GD Momentum And Adaptive LR for SPR Prediction«. In:

<sup>13</sup>Sarit KHIRIRAT, Hamid Reza FEYZMAHDAVIAN und Mikael JOHANSSON [2017]. »Mini-batch gradient descent: Faster convergence under data sparsity«. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, S. 2880–2887.

### 12.3.3 Mini-Batch Gradientenabstieg

Um die Vorteile des SGD zu nutzen und gleichzeitig die Stabilität zu verbessern, wird oft der Mini-Batch Gradientenabstieg eingesetzt<sup>14</sup>. Beim Mini-Batch Gradientenabstieg werden die Gradienten auf der Grundlage einer Mini-Batch-Größe berechnet, die größer als eins ist, aber kleiner als die gesamte Trainingsdatenmenge. Dadurch werden sowohl die Effizienz als auch die Stabilität des Trainingsprozesses verbessert.

### 12.3.4 Batch Normalisierung

Ein weiterer Ansatz zur Verbesserung des Gradientenabstiegs besteht in der Batch-Normalisierung<sup>15</sup>. Dieses Verfahren normalisiert die Eingabeaktivierungen eines Netzwerks innerhalb eines Mini-Batches, um die interne Covariate Shift-Problematik zu reduzieren. Die Batch-Normalisierung verbessert die Konvergenz des Modells, indem sie die Gradientenflüsse stabilisiert und die Abhängigkeit von der Initialisierung der Gewichte verringert.

---

<sup>14</sup>Sarit KHIRIRAT, Hamid Reza FEYZMAHDAVIAN und Mikael JOHANSSON [2017]. »Mini-batch gradient descent: Faster convergence under data sparsity«. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, S. 2880–2887.

<sup>15</sup>Yongqiang CAI, Qianxiao LI und Zuowei SHEN [2019]. »A quantitative analysis of the effect of batch normalization on gradient descent«. In: *International Conference on Machine Learning*. PMLR, S. 882–890.

# Kapitel 13

## Tools and Frameworks for CNN Training

### 13.1 PyTorch

PyTorch ist ein beliebtes Deep-Learning-Framework, das für seine Flexibilität und dynamischen Berechnungsgraphen bekannt ist. Es ermöglicht Entwicklern, ihre neuronalen Netze auf einfache und intuitive Weise zu definieren und zu trainieren. Eine der Schlüsselfunktionen von PyTorch ist die automatische Differentiation, die es ermöglicht, den Gradienten automatisch zu berechnen und so das Training von Modellen zu erleichtern. Es folgt ein Beispiel, das den Prozess des Aufbaus eines einfachen CNNs mit PyTorch veranschaulicht:

```
import torch
import torch.nn as nn

# Definition des Convolutional Neural Networks (CNN)
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1)
```

```
    self.relu = nn.ReLU()
    self.maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
    self.fc = nn.Linear(16 * 14 * 14, 10)

def forward(self, x):
    out = self.conv1(x)
    out = self.relu(out)
    out = self.maxpool(out)
    out = out.view(out.size(0), -1)
    out = self.fc(out)
    return out

# Erzeugung einer Beispiel-Eingabe
input = torch.randn(1, 1, 28, 28)

# Instanziierung des CNNs
model = CNN()

# Vorwärtsdurchlauf (Forward pass)
output = model(input)

# Ausgabe der Vorhersagen
print(output)
```

Dieses Beispiel zeigt, wie einfach es ist, ein CNN mit PyTorch zu definieren. Das CNN besteht aus Convolutional-Layer, Aktivierungsfunktionen (ReLU), Max-Pooling und einem Fully-Connected-Layer. Durch den Aufruf der ‘forward’-Methode des Modells mit einer Eingabe kann eine Vorhersage generiert werden.

## 13.2 TensorFlow

TensorFlow ist ein führendes Deep-Learning-Framework, das für seine Flexibilität, Skalierbarkeit und Effizienz bekannt ist. Es verwendet einen statischen Berechnungsgraphen, der eine effiziente Ausführung auf CPUs und GPUs ermöglicht. TensorFlow bietet auch eine breite Palette von Werkzeugen und APIs, darunter Keras, eine benutzerfreundliche High-Level-API. Das folgende Beispiel zeigt die Verwendung von TensorFlow für das Training eines CNNs:

```
import tensorflow as tf
from tensorflow.keras import layers

# Definition des Convolutional Neural Networks (CNN)
model = tf.keras.Sequential([
    layers.Conv2D(32, kernel_size=3, activation='relu', input_shape=(28,
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(10, activation='softmax')

])

# Kompilierung des Modells
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Laden der Daten und Training des Modells
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train[... , tf.newaxis] / 255.0
x_test = x_test[... , tf.newaxis] / 255.0
```

```
model.fit(x_train, y_train, batch_size=32, epochs=5)
```

```
# Auswertung des Modells
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test Accuracy:', test_acc)
```

In diesem Beispiel wird ein CNN mit TensorFlow und der Keras-API definiert. Das Modell besteht aus Convolutional-Layern, Pooling-Layern, einer Flatten-Schicht und einem Fully-Connected-Layer. Das Modell wird mit dem Adam-Optimizer und der Sparse Categorical Crossentropy als Verlustfunktion kompiliert. Anschließend wird das Modell mit den MNIST-Daten trainiert und ausgewertet.

### 13.3 Keras

Keras ist eine benutzerfreundliche Deep-Learning-Bibliothek, die auf TensorFlow, CNTK oder Theano aufbaut. Sie bietet eine einfache und intuitive Schnittstelle zum Entwerfen, Trainieren und Evaluieren von neuronalen Netzwerken. Hier ist ein Mini-Beispiel, das die Verwendung von Keras für das Training eines CNNs zeigt:

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Definition des Convolutional Neural Networks (CNN)
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
    input_shape=(28, 28, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

```
# Kompilierung des Modells
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Laden der Daten und Training des Modells
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train = x_train[... , None] / 255.0
x_test = x_test[... , None] / 255.0

model.fit(x_train, y_train, batch_size=32, epochs=5)

# Auswertung des Modells
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test Accuracy:', test_acc)
```

In diesem Beispiel wird ein CNN mit Keras definiert. Das Modell besteht aus Convolutional-Layern, Pooling-Layern, einer Flatten-Schicht und einem Fully-Connected-Layer. Das Modell wird mit dem Adam-Optimizer und der sparse kategorischen Kreuzentropie als Verlustfunktion kompiliert. Anschließend wird das Modell mit den MNIST-Daten trainiert und ausgewertet.

## 13.4 Caffe

Caffe ist ein Deep-Learning-Framework, das für seine Effizienz und Geschwindigkeit beim Training von CNNs bekannt ist. Es verwendet eine eigene Modellierungssprache, mit der Netzwerkarchitekturen einfach spezifiziert werden können. Hier ist ein Mini-Beispiel, das die Verwendung von Caffe für das Training eines CNNs zeigt:

```
# Modeldefinition in Caffe
model = """
```

```
name: "SimpleNet"
layer {
    name: "data"
    type: "Input"
    top: "data"
    input_param { shape: { dim: 1 dim: 3 dim: 224 dim: 224 } }
}
layer {
    name: "conv1"
    type: "Convolution"
    bottom: "data"
    top: "conv1"
    convolution_param {
        num_output: 64
        kernel_size: 3
        stride: 1
        pad: 1
    }
}
layer {
    name: "relu1"
    type: "ReLU"
    bottom: "conv1"
    top: "conv1"
}
layer {
    name: "pool1"
    type: "Pooling"
    bottom: "conv1"
```

```
top: "pool1"
pooling_param {
    pool: MAX
    kernel_size: 2
    stride: 2
}
"""
# Training des Modells mit Caffe
# ...
```

Dieses Beispiel zeigt die Modelldefinition eines einfachen CNNs in Caffe. Das Modell besteht aus Eingabe-, Convolutional-, ReLU-, Pooling-, Fully-Connected- und Softmax-Schichten. Die Definition erfolgt über eine spezifische Syntax. Anschließend kann das Modell mit Caffe trainiert werden.

## 13.5 Weitere beliebte Frameworks

Neben den oben genannten Frameworks gibt es noch weitere beliebte Deep-Learning-Frameworks wie MXNet, Theano und Torch. Diese Frameworks bieten ebenfalls leistungsstarke Funktionen und werden in verschiedenen Anwendungsfällen eingesetzt. MXNet zeichnet sich beispielsweise durch seine Skalierbarkeit aus, Theano war eines der ersten Frameworks, das GPU-Beschleunigung unterstützte, und Torch bietet eine flexible und dynamische Berechnungsgraphen-Engine. Jedes Framework hat seine eigenen Stärken und Vorzüge, die Auswahl kann je nach den Anforderungen und Präferenzen des Anwenders und der Problemstellung variieren.

# Kapitel 14

## Schlusswort

### 14.1 Zusammenfassung und Fazit der Arbeit

Das Ziel dieser Arbeit war es, einen umfassenden Überblick über Skalierungsmethoden in der Bildverarbeitung zu geben. In den Grundlagen wurden verschiedene Arten der Skalierung und wichtige Aspekte behandelt. Anschließend wurden klassische Skalierungsmethoden wie Pixelverdopplung (3.1), Nearest-Neighbor-Interpolation (3.2), bilineare Interpolation (3.3), Bikubische Interpolation (3.4) und Lanczos-Interpolation<sup>3.5</sup> untersucht und deren Vor- und Nachteile analysiert (3.7).

Darüber hinaus wurden fortgeschrittene Skalierungsmethoden wie Convolutional neural network (CNN) (4.1), Super Resolution(4.2), Generative Adversarial Network (GAN) (4.3) und Multiscale-Skalierung (4.4) vorgestellt. Es wurde auf die Grundlagen, Architekturen, Anwendungen und Limitationen dieser Methoden eingegangen (4.5).

Die Evaluation von Skalierungsmethoden wurde anhand verschiedener Qualitätsmetriken wie Peak Signal-to-Noise Ratio (PSNR) (5.1.1), Structural Similarity Index Measure (SSIM) (5.1.2) und Mean Opinion Score (MOS) (5.1.3) durchgeführt. Zudem wurden Kriterien zur Auswahl der geeigneten Skalierungsmethode diskutiert (5.2).

Ein Schwerpunkt der Arbeit lag auf der Einführung in Deep Learning (6) und speziell Convolutional Neural Networks (7). Die Architektur von CNNs (7.3), deren Anwendungen (7.3) und der Trainingsprozess wurden detailliert erläutert. Des Weiteren wurde die

Bedeutung von Datenverarbeitung (8), einschließlich Datennormalisierung (8.3) und Datenaugmentierung (8.4), betont.

Abschließend wurden gängige Herausforderungen und Lösungen im Bereich des Deep Learning diskutiert, wie Overfitting (12.1.1), Exploding und Vanishing Gradients (12.2) sowie Gradient Descent Optimization (12.3). Verschiedene Tools und Frameworks für das Training von CNNs, wie PyTorch (13.1), TensorFlow (13.2), Keras (13.3) und Caffe (13.4), wurden vorgestellt.

Zusammenfassend liefert diese Arbeit einen umfassenden Einblick in Skalierungsmethoden und deren Anwendungen in der Bildverarbeitung. Zukünftige Forschungsrichtungen könnten sich auf die Verbesserung der Skalierungsgenauigkeit, die Effizienz der Methoden und die Anwendung auf spezifische Domänen konzentrieren.

## 14.2 Perspektiven für zukünftige Forschungen

In Anbetracht der vorliegenden Forschungsergebnisse und des implementierten Bildverarbeitungsverfahrens zur Segmentierung und Auswertung von Füllständen in Bildern eröffnen sich mehrere vielversprechende Perspektiven für zukünftige Forschungen. Diese Perspektiven können zur weiteren Verbesserung der Bildverarbeitungsmethoden und neuronalen Netze sowie zur Erweiterung des Anwendungsbereichs beitragen. Im Folgenden werden einige relevante Themen für zukünftige Forschungen skizziert:

### 14.2.1 Optimierung der Bildverarbeitungsmethoden

Obwohl die in dieser Studienarbeit verwendeten Bildverarbeitungsmethoden gute Ergebnisse bei der Füllstandserkennung gezeigt haben, gibt es noch Raum für Verbesserungen. Zukünftige Forschungen könnten darauf abzielen, neue Bildverarbeitungstechniken zu entwickeln oder vorhandene Methoden weiter zu optimieren, um die Genauigkeit und Effizienz der Füllstandserkennung zu steigern. Die Verwendung fortschrittlicher Algorithmen zur Bildverbesserung, Rauschunterdrückung und Kantenextraktion könnte beispielsweise zu präziseren Segmentierungsergebnissen führen. Darüber hinaus könnten adaptive Methoden

untersucht werden, die sich an verschiedene Arten von Behältern und Beleuchtungsbedingungen anpassen können.

### 14.2.2 Integration fortschrittlicher neuronaler Netze

Die vorgestellten neuronalen Netzwerkarchitekturen haben sich als wirksam erwiesen, aber es gibt noch viele weitere Architekturen und Ansätze, die in Betracht gezogen werden könnten. Zukünftige Forschungen könnten den Einsatz fortschrittlicher neuronaler Netze wie Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) oder Transformer-Netzwerken für die Füllstandserkennung untersuchen. Diese Netze könnten in Kombination mit Transfer Learning oder generativen Modellen wie Generative Adversarial Networks (GANs) eingesetzt werden, um bessere Ergebnisse zu erzielen und die Robustheit gegenüber verschiedenen Szenarien und Datenqualitäten zu verbessern.

### 14.2.3 Erweiterung des Anwendungsbereichs

Die Anwendung des entwickelten Bildverarbeitungsverfahrens wurde auf die Segmentierung und Auswertung von Füllständen in Bildern beschränkt. Zukünftige Forschungen könnten jedoch den Anwendungsbereich erweitern und das Verfahren auf andere ähnliche Aufgaben anwenden. Beispielsweise könnten ähnliche Techniken zur Erkennung und Klassifizierung von Objekten in Bildern eingesetzt werden, oder das Verfahren könnte auf andere Bildverarbeitungsaufgaben wie Gesichtserkennung, medizinische Bildgebung oder autonome Fahrzeuge angewendet werden. Eine solche Erweiterung würde dazu beitragen, die Einsatzmöglichkeiten des entwickelten Verfahrens zu erweitern und dessen praktischen Nutzen zu maximieren.

### 14.2.4 Evaluation auf größeren und vielfältigeren Datensätzen

Eine weitere vielversprechende Perspektive für zukünftige Forschungen besteht darin, das entwickelte Bildverarbeitungsverfahren auf größeren und vielfältigeren Datensätzen zu evaluieren. Die in dieser Studienarbeit verwendeten Datensätze waren begrenzt und

möglicherweise nicht repräsentativ für alle potenziellen Anwendungsfälle. Zukünftige Forschungen könnten die Performance des Verfahrens auf umfangreicheren Datensätzen mit unterschiedlichen Behältertypen, Füllstandsniveaus und Beleuchtungsbedingungen untersuchen. Dies würde dazu beitragen, die Robustheit und Allgemeingültigkeit des Verfahrens zu validieren und mögliche Einschränkungen oder Herausforderungen bei der Anwendung auf verschiedene Szenarien aufzudecken.

#### **14.2.5 Integration von Echtzeitverarbeitung und Echtzeitfeedback**

Die Echtzeitverarbeitung von Bildern und das Echtzeitfeedback sind für viele Anwendungen entscheidend. Zukünftige Forschungen könnten darauf abzielen, das entwickelte Bildverarbeitungsverfahren so zu optimieren, dass es in Echtzeit arbeitet und sofortiges Feedback zu den Füllstandsergebnissen liefert. Dies würde den praktischen Nutzen des Verfahrens erheblich erhöhen und es für Anwendungen in Echtzeitüberwachungssystemen, automatisierten Prozessen oder reaktiven Steuerungssystemen geeignet machen. Die Integration von fortschrittlichen Technologien wie GPUs, FPGAs oder spezialisierten Hardwareplattformen könnte erforscht werden, um die Verarbeitungsgeschwindigkeit zu verbessern und die Latenzzeiten zu minimieren.

#### **14.2.6 Integration von KI-gestütztem Lernen**

Die Integration von KI-gestütztem Lernen in das entwickelte Bildverarbeitungsverfahren könnte einen weiteren Forschungsbereich darstellen. Zukünftige Forschungen könnten untersuchen, wie das Verfahren durch die kontinuierliche Analyse und das Lernen aus den erfassten Füllstandsinformationen verbessert werden kann. Die Anwendung von Reinforcement Learning könnte es dem Verfahren ermöglichen, sich an neue Gegebenheiten, Veränderungen in den Behälterbedingungen oder individuelle Präferenzen anzupassen. Durch die Integration von KI-gestütztem Lernen könnte das Verfahren adaptiver und selbstlernender werden, was zu einer kontinuierlichen Verbesserung der Leistung und der

Anpassungsfähigkeit führen würde.

### **14.2.7 Berücksichtigung von Datenschutz und Sicherheit**

Ein wichtiger Aspekt bei der weiteren Erforschung und Anwendung von Bildverarbeitungsmethoden und neuronalen Netzen ist die Berücksichtigung von Datenschutz und Sicherheit. Zukünftige Forschungen sollten den Schutz personenbezogener Daten und die Vermeidung von Missbrauch oder unerwünschter Überwachung in Betracht ziehen. Die Entwicklung von Datenschutzrichtlinien, Anonymisierungstechniken oder Privatsphäreschutzmechanismen könnte erforscht werden, um sicherzustellen, dass das entwickelte Verfahren ethischen Richtlinien und gesetzlichen Bestimmungen entspricht.

### **14.2.8 Langzeitstudien zur Zuverlässigkeit und Stabilität**

Um die Langzeitzuverlässigkeit und Stabilität des entwickelten Bildverarbeitungsverfahrens zu gewährleisten, könnten zukünftige Forschungen Langzeitstudien durchführen. Durch die kontinuierliche Überwachung und Bewertung der Leistung des Verfahrens über einen längeren Zeitraum hinweg könnten potenzielle Auswirkungen von Umweltbedingungen, Alterungseffekten oder Änderungen in den Einsatzumgebungen ermittelt werden. Dies würde helfen, die praktische Anwendbarkeit des Verfahrens im Langzeitbetrieb zu bewerten und gegebenenfalls Optimierungen oder Wartungsmaßnahmen vorzunehmen.

Insgesamt bieten diese Perspektiven für zukünftige Forschungen spannende Möglichkeiten zur Weiterentwicklung und Verbesserung der Bildverarbeitungsmethoden und neuronalen Netze zur Segmentierung und Auswertung von Füllständen in Bildern. Die vorgeschlagenen Forschungsthemen können zur Weiterentwicklung des Fachgebiets beitragen und den praktischen Nutzen dieser Technologien in verschiedenen Anwendungsdomänen erweitern.

# Abkürzungsverzeichnis

|               |                                                |    |
|---------------|------------------------------------------------|----|
| <b>PPI</b>    | Pixel per Inch . . . . .                       | 9  |
| <b>CNN</b>    | Convolutional neural network . . . . .         | 9  |
| <b>SR</b>     | Super Resolution . . . . .                     | 51 |
| <b>GAN</b>    | Generative Adversarial Network . . . . .       | 9  |
| <b>PNG</b>    | Portable Network Graphics . . . . .            | 15 |
| <b>JPG</b>    | Joint Photographic Experts Group . . . . .     | 16 |
| <b>JPEG</b>   | Joint Photographic Experts Group . . . . .     | 16 |
| <b>SVG</b>    | Scalable Vector Graphics . . . . .             | 17 |
| <b>XML</b>    | eXtensible Markup Language . . . . .           | 17 |
| <b>CSS</b>    | Cascading Style Sheets . . . . .               | 17 |
| <b>GIF</b>    | Graphics Interchange Format . . . . .          | 18 |
| <b>EPS</b>    | Encapsulated PostScript . . . . .              | 19 |
| <b>BMP</b>    | Bitmap . . . . .                               | 18 |
| <b>PSD</b>    | Photoshop Document . . . . .                   | 18 |
| <b>TIFF</b>   | Tagged Image File Format . . . . .             | 18 |
| <b>NNI</b>    | Nearest Neighbor Interpolation . . . . .       | 26 |
| <b>FLOPS</b>  | Floating Point Operations Per Second . . . . . | 70 |
| <b>PSNR-Y</b> | Peak Signal-to-Noise Ratio Y . . . . .         | 70 |

## 14.2. PERSPEKTIVEN FÜR ZUKÜNFTIGE FORSCHUNGEN 156

|             |                                               |    |
|-------------|-----------------------------------------------|----|
| <b>MOS</b>  | Mean Opinion Score . . . . .                  | 69 |
| <b>RMSE</b> | Root Mean Squared Error . . . . .             | 71 |
| <b>MSDN</b> | Multiscale Dense Network . . . . .            | 63 |
| <b>PSNR</b> | Peak Signal-to-Noise Ratio . . . . .          | 4  |
| <b>MSE</b>  | Mean Squared Error . . . . .                  | 67 |
| <b>SSIM</b> | Structural Similarity Index Measure . . . . . | 4  |
| <b>NLP</b>  | Natural Language Processing . . . . .         | 96 |

# Abbildungsverzeichnis

8figure.1.1

|     |                                                                                                                                                                                           |    |
|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1.2 | Verschiedene Beispiele von upscaling Algorithmen[WHUBER 2011]. . . . .                                                                                                                    | 10 |
| 2.1 | Die 3 Booms in der Geschichte von KI. Vorlesung DHBW Karlsruhe,<br>TINF20, 2023 – D. Münch . . . . .                                                                                      | 13 |
| 3.1 | Beispielhafte Darstellung einer Skalierung durch Pixelverdopplung . . . . .                                                                                                               | 24 |
| 3.2 | Bilineare Verdopplung . . . . .                                                                                                                                                           | 28 |
| 3.3 | Graph über bilineare Skalierung. . . . .                                                                                                                                                  | 28 |
| 3.4 | Bikubische Interpolation . . . . .                                                                                                                                                        | 31 |
| 3.5 | Durchschnittliche Laufzeit verschiedener Interpolationsalgorithmen in Se-<br>kunden[HAN 2013]. . . . .                                                                                    | 44 |
| 4.1 | Verschiedene Architekturen von Image Networks. Oben links vereinfachtest<br>Alex net, unten links inception net, rechts Abbildung der Res net Architektur. .                              | 48 |
| 4.2 | Beispielbild eines Frames mit Bounding Boxes . . . . .                                                                                                                                    | 49 |
| 4.3 | SAM: A generalized approach to segmentation . . . . .                                                                                                                                     | 51 |
| 4.4 | Super resolution links, original rechts . . . . .                                                                                                                                         | 52 |
| 4.5 | Der Generator erzeugt neue Daten, während der Diskriminator versucht,<br>zwischen den vom Generator erzeugten Daten und den echten Daten zu<br>unterscheiden. . . . .                     | 55 |
| 4.6 | Eingabe in das Modell: A small cactus wearing a straw hat and neon<br>sunglasses in the Sahara desert.: <a href="https://Imagen.research.google/">https://Imagen.research.google/</a> . . | 56 |

|                                                                                                                                                 |     |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 4.7 <a href="https://www.tensorflow.org/tutorials/generative/style_transfer">https://www.tensorflow.org/tutorials/generative/style_transfer</a> | 57  |
| 4.8 Füllen von markierten Bereichen eines Bildes durch verwendung von Imagen und Erweitern von Bildern mithilfe von Imagen, Google.             | 57  |
| 4.9 Generierung kurzer Videos, anhand von einfachen textuellen Beschreibungen.                                                                  | 58  |
| 4.10 Beispiel für Vorurteile in GANs [ADAMS 2023]                                                                                               | 60  |
| 6.1 Timeline der Geschichte von Deep Learning                                                                                                   | 80  |
| 7.1 Primitive Abstraktion eines Modellierungsprozesses für maschinelles Lernen                                                                  | 82  |
| 7.2 Einordnung der Begriffe Künstliche Intelligenz, Maschine Learning und Deep Learning.                                                        | 84  |
| 7.3 Einsicht in die Vorhergehensweise eines Neuronalen Netzen, und dessen automatische Merkmalsextraktion au mehreren Schichten. <sup>1</sup>   | 87  |
| 7.4 Sobel Kernel.                                                                                                                               | 89  |
| 7.5 Max und average Pooling.                                                                                                                    | 90  |
| 7.6 Modell zur Bildklassifizierung                                                                                                              | 94  |
| 7.7 Modell zur Objekterkennung.                                                                                                                 | 94  |
| 7.8 Modell zur Gesichtserkennung.                                                                                                               | 95  |
| 7.9 Eine verallgemeinerte Deep-NN-Architektur für natürliche Sprachverarbeitung (NLP). [COLLOBERT und WESTON 2008]                              | 96  |
| 7.10 Input layers                                                                                                                               | 97  |
| 7.11 Hidden layers                                                                                                                              | 98  |
| 7.12 Output layers                                                                                                                              | 99  |
| 7.13 max und average Pooling.                                                                                                                   | 101 |
| 7.14 Links fully connected, rechts not fully connected layers.                                                                                  | 102 |
| 7.15 Overfitting und geringe Generalisierung in alleinigen CNN.                                                                                 | 104 |

---

<sup>1</sup>Hao Li u. a. [2018]. »Visualizing the loss landscape of neural nets«. In: *Advances in neural information processing systems* 31.

|                                                                                                                                                                                                |     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 7.16 Reduktion von Overfitting und Verbesserung der Generalisierung durch Anwendung von Verschiebung in Breite und Höhe zusammen mit Dropout bei Convolutional Neural Networks (CNN) . . . . . | 104 |
| 7.17 ReLu Aktivierungsfunktion. . . . .                                                                                                                                                        | 105 |
| 7.18 Sigmoid Aktivierungsfunktion. . . . .                                                                                                                                                     | 105 |
| 7.19 Links Softmax-2D, mittig Center-Loss, rechts PEDCC-loss. . . . .                                                                                                                          | 106 |
|                                                                                                                                                                                                |     |
| 8.1 Verschiedene Methoden und Kombinationen zur Datenaugmentation. . .                                                                                                                         | 113 |
| 8.2 Klassische Methoden und Kombinationen zur Datenaugmentation. . .                                                                                                                           | 113 |
|                                                                                                                                                                                                |     |
| 10.1 Beispiel von Stochastic Gradient Descent mit 30 Lernschritten. . . . .                                                                                                                    | 121 |
| 10.2 Gradientenabstiegsverfahren mit verschiedenen Schrittweiten . . . . .                                                                                                                     | 122 |
| 10.3 Backpropagation. Quelle: Vorlesung KI DHBW . . . . .                                                                                                                                      | 122 |
| 10.4 Hyperparameter tuning. . . . .                                                                                                                                                            | 124 |
|                                                                                                                                                                                                |     |
| 12.1 Beispiel von Under- und Overfitting Probleme . . . . .                                                                                                                                    | 135 |
| 12.2 Beispiel von Vanishing- und Exploding Gradient Probleme . . . . .                                                                                                                         | 137 |

# Tabellenverzeichnis

|                                                                                                                                                                                                                                                                                                           |    |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.1 Kontrasttabelle der subjektiven Bewertung mit verschiedenen Interpolationsmethoden [HAN 2013] . . . . .                                                                                                                                                                                               | 42 |
| 3.2 Contrast table of SNR with different interpolation method[HAN 2013] . .                                                                                                                                                                                                                               | 43 |
| 5.1 Leistungsvergleich zwischen PSNR, SSIM, MS-SSIM, VQM, PQR-Tek, DMOS-Tek, JND-VC, DMOS-VC und SSIMplus einschließlich aller Geräte <sup>2</sup> .                                                                                                                                                      | 74 |
| 7.1 Leistung der Algorithmen: Translations- und Rotationsfehler von Testsequenzen unter Verwendung des KITTI Devkit. AD-VO funktioniert zuverlässig in verschiedenen Fahrumgebungen. 'D' steht für die Verwendung einer Disparitätskarte und 'A' bedeutet, dass das Modell einen Attention-Block anpasst. | 93 |

---

<sup>2</sup>Abdul REHMAN, Kai ZENG und Zhou WANG [2015]. URL: <https://ece.uwaterloo.ca/~z70wang/publications/HVEI15.pdf>.

# Liste der Algorithmen

|     |                                                                                                                                                                                                                                                                                                                                                                                                                  |     |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 3.1 | Python-Klasse zur Pixelverdopplung und Bildmanipulation: Implementierung des Algorithmus mit der Pillow-Bibliothek und Erklärung des Codes von PixelVerdopplung: <a href="https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/pixel_verdopplung.py">https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/pixel_verdopplung.py</a> . | 25  |
| 3.2 | <a href="https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bilinear_interpolation.py">https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bilinear_interpolation.py</a> . . . . .                                                                                                                                                | 28  |
| 3.3 | Python-Klasse zur Bikubischen interpolation <a href="https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bicubic_interpolation.py">https://github.com/studienarbeit-cnn-dhwka-2022/Code/blob/main/backend/skalierungsmethoden/bicubic_interpolation.py</a> . . . . .                                                                                                      | 32  |
| 7.1 | Trainingsverlustverlauf: Unsere Loss-Funktion nimmt kontinuierlich ab, was auf eine verbesserte Modellleistung hinweist. . . . .                                                                                                                                                                                                                                                                                 | 106 |

# Literatur

ADAMS, Robert [2023]. *Midjourney Smells Of Racism: The Neural Network Created A Provocative Image - Gadget Tendency.* <https://gadgettendency.com/midjourney-smells-of-racism-the-neural-network-created-a-provocative-image/>. [Accessed 21-May-2023] [siehe S. 60].

ALBER, Mark u. a. [2019]. »Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences«. In: *NPJ digital medicine* 2.1, S. 115 [siehe S. 61].

ALHUZALI, Hassan, Tianlin ZHANG und Sophia ANANIADOU [2021]. »Predicting Sign of Depression via Using Frozen Pre-trained Models and Random Forest Classifier.« In: *CLEF (Working Notes)*, S. 888–896 [siehe S. 131].

ALOM, Md Zahangir u. a. [2017]. *Inception Recurrent Convolutional Neural Network for Object Recognition*. DOI: 10.48550/ARXIV.1704.07709. URL: <https://arxiv.org/abs/1704.07709> [siehe S. 47, 48].

ALOM, Md Zahangir u. a. [2018]. »The history began from alexnet: A comprehensive survey on deep learning approaches«. In: *arXiv preprint arXiv:1803.01164* [siehe S. 14].

ALOYSIUS, Neena und M. GEETHA [Apr. 2017]. »A review on deep convolutional neural networks«. In: *2017 International Conference on Communication and Signal Processing (ICCPSP)*. IEEE. DOI: 10.1109/iccsp.2017.8286426. URL: <https://doi.org/10.1109/iccsp.2017.8286426> [siehe S. 47].

- ALSHALALI, Tagrid und Darsana JOSYULA [2018]. »Fine-tuning of pre-trained deep learning models with extreme learning machine«. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, S. 469–473 [siehe S. 128].
- AMARI, Shun-ichi [1993]. »Backpropagation and stochastic gradient descent method«. In: *Neurocomputing* 5.4-5, S. 185–196 [siehe S. 121–123].
- BAI, Yingbin u. a. [2021]. »Understanding and improving early stopping for learning with noisy labels«. In: *Advances in Neural Information Processing Systems* 34, S. 24392–24403 [siehe S. 135].
- BARR, Avron, Edward A FEIGENBAUM und Paul R COHEN [1981]. *The handbook of artificial intelligence*. Bd. 1. William Kaufmann [siehe S. 12].
- BENTBIB, A.H. u. a. [2016]. »A global Lanczos method for image restoration«. In: *Journal of Computational and Applied Mathematics* 300, S. 233–244. ISSN: 0377-0427. DOI: <https://doi.org/10.1016/j.cam.2015.12.034>. URL: <https://www.sciencedirect.com/science/article/pii/S0377042715006469> [siehe S. 38].
- BLAU, Yochai und Tomer MICHAELI [2018]. »The perception-distortion tradeoff«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 6228–6237. DOI: [10.1109/CVPR.2018.00651](https://doi.org/10.1109/CVPR.2018.00651) [siehe S. 76].
- BOUTELL, Thomas [1997]. *Png (portable network graphics) specification version 1.0*. Techn. Ber. [siehe S. 16].
- BOUTELL, Thomas u. a. [o. D.] *PNG (Portable Network Graphics Format) Version 1.0*. Techn. Ber. RFC 2083 [siehe S. 15].
- BOYD, Aidan, Adam CZAJKA und Kevin BOWYER [2020]. *Deep Learning-Based Feature Extraction in Iris Recognition: Use Existing Models, Fine-tune or Train From Scratch?* arXiv: 2002.08916 [cs.CV] [siehe S. 86].
- BURGER, Wilhelm und Mark James BURGE [2009]. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag [siehe S. 15].

- BURGER, Wilhelm u. a. [2015]. »Digitale Bilder«. In: *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*, S. 1–24 [siehe S. 16].
- CAI, Yongqiang, Qianxiao LI und Zuowei SHEN [2019]. »A quantitative analysis of the effect of batch normalization on gradient descent«. In: *International Conference on Machine Learning*. PMLR, S. 882–890 [siehe S. 44, 142].
- CHAIB, Souleyman u. a. [2017]. »Deep feature extraction and combination for remote sensing image classification based on pre-trained CNN models«. In: *Ninth international conference on digital image processing (ICDIP 2017)*. Bd. 10420. SPIE, S. 712–716 [siehe S. 130, 133].
- CHAPMAN, Arthur D [2005]. *Principles of data quality*. GBIF [siehe S. 109].
- CHEN, Jou-Fan u. a. [Nov. 2016]. »Financial Time-Series Data Analysis Using Deep Convolutional Neural Networks«. In: *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE. DOI: 10.1109/ccbd.2016.027. URL: <https://doi.org/10.1109/ccbd.2016.027> [siehe S. 88].
- CHEN, Liang-Chieh u. a. [2017]. *Rethinking Atrous Convolution for Semantic Image Segmentation*. arXiv: 1706.05587 [cs.CV] [siehe S. 131].
- CHEN, Ming-Jun und Alan C BOVIK [2011]. »Fast structural similarity index algorithm«. In: *Journal of Real-Time Image Processing* 6, S. 281–287 [siehe S. 68].
- CHEN, Xiangyi, Steven Z WU und Mingyi HONG [2020]. »Understanding gradient clipping in private sgd: A geometric perspective«. In: *Advances in Neural Information Processing Systems* 33, S. 13773–13782 [siehe S. 140].
- CHIVERS, Ian u. a. [2015]. »An introduction to Algorithms and the Big O Notation«. In: *Introduction to Programming with Fortran: With Coverage of Fortran 90, 95, 2003, 2008 and 77*, S. 359–364 [siehe S. 10].
- CHOI, Min Joon, Sang Heon LEE und Sang Jun LEE [2019]. »Lightweight and Efficient Deep Neural Network for Super-Resolution with Mobile Devices«. In: *Electronics* 8.4, S. 423. DOI: 10.3390/electronics8040423 [siehe S. 71].

- COLLOBERT, Ronan und Jason WESTON [2008]. »A unified architecture for natural language processing: Deep neural networks with multitask learning«. In: *Proceedings of the 25th international conference on Machine learning*, S. 160–167 [siehe S. 96].
- CONTRIBUTORS, Wikipedia [2023]. *Bildskalierung Definition*. [https://en.wikipedia.org/wiki/Image\\_scaling](https://en.wikipedia.org/wiki/Image_scaling) [siehe S. 9].
- CORDIER, Frederic, Hyewon SEO und Nadia MAGNENAT-THALMANN [2003]. »Made-to-measure technologies for an online clothing store«. In: *IEEE Computer graphics and applications* 23.1, S. 38–48 [siehe S. 23].
- DAVIES, Andrew, Stephen SERJEANT und Jane M BROMLEY [2019]. »Using convolutional neural networks to identify gravitational lenses in astronomical images«. In: *Monthly Notices of the Royal Astronomical Society* 487.4, S. 5263–5271 [siehe S. 88].
- DE REU, Jeroen u. a. [2014]. »On introducing an image-based 3D reconstruction method in archaeological excavation practice«. In: *Journal of Archaeological Science* 41, S. 251–262 [siehe S. 23].
- »Digital image enhancement: A survey« [1983]. In: *Computer Vision, Graphics, and Image Processing* 24.3, S. 363–381. ISSN: 0734-189X. DOI: [https://doi.org/10.1016/0734-189X\(83\)90061-0](https://doi.org/10.1016/0734-189X(83)90061-0) [siehe S. 24].
- DONG, Chao, Chen Change LOY und Xiaou Tang TANG [2016]. *Accelerating the Super-Resolution Convolutional Neural Network*. arXiv: 1608.00367 [cs.CV] [siehe S. 54].
- DONG, Chao u. a. [2016]. »Image Super-Resolution Using Deep Convolutional Networks«. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2, S. 295–307. DOI: 10.1109/TPAMI.2015.2439281 [siehe S. 51].
- DUCHON, Claude E [1979]. »Lanczos filtering in one and two dimensions«. In: *Journal of Applied Meteorology and Climatology* 18.8, S. 1016–1022 [siehe S. 37].
- ECKERT, Michael P und Andrew P BRADLEY [1998]. »Perceptual quality metrics applied to still image compression«. In: *Signal processing* 70.3, S. 177–200 [siehe S. 73].

- EDDY, Sean R [2004]. »What is a hidden Markov model?« In: *Nature biotechnology* 22.10, S. 1315–1316 [siehe S. 13].
- EL NAQA, Issam und Martin J MURPHY [2015]. *What is machine learning?* Springer [siehe S. 81, 82].
- ELSEVIER [n.d.] *Artwork and media instructions*. <https://www.elsevier.com/authors/policies-and-guidelines/artwork-and-media-instructions/artwork-overview>. Accessed: March 9, 2023 [siehe S. 19].
- ENCYCLOPEDIA BRITANNICA [n.d.] *JPEG*. <https://www.britannica.com/technology/JPEG>. Accessed: March 9, 2023 [siehe S. 19].
- ERTEL, Wolfgang und Nathanael T BLACK [2016]. *Grundkurs Künstliche Intelligenz*. Bd. 4. Springer [siehe S. 91].
- FANG, Ning und Zongqian ZHAN [2022]. »High-resolution optical flow and frame-recurrent network for video super-resolution and deblurring«. In: *Neurocomputing* 489, S. 128–138 [siehe S. 52].
- FEI, Nanyi u. a. [Okt. 2021]. »Z-Score Normalization, Hubness, and Few-Shot Learning«. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE. DOI: 10.1109/iccv48922.2021.00021. URL: <https://doi.org/10.1109/iccv48922.2021.00021> [siehe S. 111].
- FIBINGER, Iris [2002]. *SVG. Scalable Vector Graphics.: Praxiswegweiser und Referenz für den neuen Vektorgrafikstandard. Für Fortgeschrittene*. Markt+Technik-Verl. ISBN: 3827262399, 9783827262394. URL: <http://gen.lib.rus.ec/book/index.php?md5=1a655693c0779b14c519078151dfdf31> [siehe S. 17].
- GEHANI, Ashish und John REIF [2007]. »Super-Resolution Video Analysis for Forensic Investigations«. In: *Advances in Digital Forensics III*. Hrsg. von Philip CRAIGER und Sujeet SHENOI. New York, NY: Springer New York, S. 281–299. ISBN: 978-0-387-73742-3 [siehe S. 52].

- GHIASI, Golnaz, Tsung-Yi LIN und Quoc V LE [2018]. »Dropblock: A regularization method for convolutional networks«. In: *Advances in neural information processing systems* 31 [siehe S. 124].
- GLOROT, Xavier und Yoshua BENGIO [2010]. »Understanding the difficulty of training deep feedforward neural networks«. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Hrsg. von Yee Whye TEH und Mike TITTERINGTON. Bd. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, S. 249–256. URL: <https://proceedings.mlr.press/v9/glorot10a.html> [siehe S. 139].
- GOEL, Sumang [Juni 2020]. *Understanding Convolutional Neural Network*. Medium. URL: <https://medium.com/@sumangoel151/understanding-convolutional-neural-network-76e465f65ef3> [siehe S. 89].
- GOHSI, Seiichi [2015]. »Real-time super resolution algorithm for security cameras«. In: *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*. Bd. 5. IEEE, S. 92–97 [siehe S. 53].
- GRANWEHR, Alina und Verena HOFER [2021]. »Analysis on digital image processing for plant health monitoring«. In: *Journal of Computing and Natural Science* 1, S. 1 [siehe S. 23].
- GRIBBON, K.T. und D.G. BAILEY [2004]. »A novel approach to real-time bilinear interpolation«. In: *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*, S. 126–131. DOI: 10.1109/DELTA.2004.10055 [siehe S. 30, 36].
- GU, Jiuxiang u. a. [2018]. »Recent advances in convolutional neural networks«. In: *Pattern recognition* 77, S. 354–377 [siehe S. 107].
- GU, K. u. a. [2019]. »A Comprehensive Study of Image Upsampling Quality Metrics«. In: *IEEE Transactions on Image Processing* 28.3, S. 1316–1331. DOI: 10.1109/TIP.2019.2906826 [siehe S. 74].

- GUO, Tianmei u. a. [2017]. »Simple convolutional neural network on image classification«. In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, S. 721–724 [siehe S. 94].
- HAN, Dianyuan [2013]. »Comparison of commonly used image interpolation methods«. In: *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. Atlantis Press, S. 1556–1559 [siehe S. 42–44].
- HAN, Xu u. a. [2021]. »Pre-trained models: Past, present and future«. In: *AI Open* 2, S. 225–250 [siehe S. 129].
- HAO, Wang u. a. [Dez. 2020]. »The Role of Activation Function in CNN«. In: *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*. IEEE. DOI: 10.1109/itca52113.2020.00096. URL: <https://doi.org/10.1109/itca52113.2020.00096> [siehe S. 104].
- HASSABIS, Demis [2017]. *Artificial intelligence: chess match of the century* [siehe S. 14].
- HE, Kaiming u. a. [2016]. »Deep residual learning for image recognition«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 770–778 [siehe S. 63, 85, 131].
- [2015]. »Delving deep into rectifiers: Surpassing human-level performance on imagenet classification«. In: *Proceedings of the IEEE international conference on computer vision*, S. 1026–1034 [siehe S. 139].
- HE, Sheng u. a. [2023]. »Segmentation ability map: Interpret deep features for medical image segmentation«. In: *Medical Image Analysis* 84, S. 102726 [siehe S. 64].
- HERNANDEZ, Danny und Tom B BROWN [2020]. »Measuring the algorithmic efficiency of neural networks«. In: *arXiv preprint arXiv:2005.04305* [siehe S. 70].
- HORÉ, Alain und Djemel ZIOU [2010]. »Image Quality Metrics: PSNR vs. SSIM«. In: *2010 20th International Conference on Pattern Recognition*, S. 2366–2369. DOI: 10.1109/ICPR.2010.579 [siehe S. 69].

- HU, Y. u. a. [2020]. »An Efficient Edge-Guided Image Upsampling Method Based on Local Frequency Estimation«. In: *IEEE Transactions on Image Processing* 29, S. 430–442. DOI: 10.1109/TIP.2019.2896251 [siehe S. 74].
- HUANG, Gao u. a. [2017]. »Densely connected convolutional networks«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 4700–4708 [siehe S. 62].
- HUANG, Q., L. YU und S. WANG [2010]. »A New Image Quality Assessment Method for Peak Signal-to-Noise Ratio (PSNR) in YUV Color Space«. In: *IEEE Transactions on Consumer Electronics* 56.4, S. 2317–2322. DOI: 10.1109/TCE.2010.5682097 [siehe S. 70].
- HUANG, Yu und Yue CHEN [2020]. *Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies*. arXiv: 2006.06091 [cs.CV] [siehe S. 86].
- IBRAHIM HAIRAB, Belal u. a. [2023]. »Anomaly Detection of Zero-Day Attacks Based on CNN and Regularization Techniques«. In: *Electronics* 12.3, S. 573 [siehe S. 124].
- IONOS [n.d.] *Graphic File Formats: Which Formats Are Important?* <https://www.ionos.com/digitalguide/websites/web-design/graphic-file-formats-which-formats-are-important/>. Accessed: March 9, 2023 [siehe S. 19].
- JADERBERG, Max, Karen SIMONYAN, Andrew ZISSERMAN u. a. [2015]. »Spatial transformer networks«. In: *Advances in neural information processing systems* 28 [siehe S. 49].
- JANOCHA, Katarzyna und Wojciech Marian CZARNECKI [2017]. »On loss functions for deep neural networks in classification«. In: *arXiv preprint arXiv:1702.05659* [siehe S. 107].
- JEDDI, Ahmadreza, Mohammad Javad SHAFIEE und Alexander WONG [2020]. *A Simple Fine-tuning Is All You Need: Towards Robust Deep Learning Via Adversarial Fine-tuning*. arXiv: 2012.13628 [cs.CV] [siehe S. 129].

- JIANG, Nan und Luo WANG [2015]. »Quantum image scaling using nearest neighbor interpolation«. In: *Quantum Information Processing* 14, S. 1559–1571 [siehe S. 27].
- KHALEDYAN, Donya u. a. [2020]. »Low-cost implementation of bilinear and bicubic image interpolation for real-time image super-resolution«. In: *2020 IEEE Global Humanitarian Technology Conference (GHTC)*. IEEE, S. 1–5 [siehe S. 35].
- KHIRIRAT, Sarit, Hamid Reza FEYZMAHDAVIAN und Mikael JOHANSSON [2017]. »Minibatch gradient descent: Faster convergence under data sparsity«. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, S. 2880–2887 [siehe S. 141, 142].
- KIETZMANN, Jan u. a. [2020]. »Deepfakes: Trick or treat?« In: *Business Horizons* 63.2, S. 135–146 [siehe S. 14].
- KIM, J. und S. LEE [2019]. »Deep Learning-Based Scaling Methods for Image Super-Resolution«. In: *Pattern Recognition* [siehe S. 51].
- KLÜVER, Christina und Jürgen KLÜVER [2022]. »Chancen und Herausforderungen beim Einsatz neuronaler Netzwerke als Methoden der Künstlichen Intelligenz oder des Maschinellen Lernens in KMU«. In: *Digitalisierung und Nachhaltigkeit – Transformation von Geschäftsmodellen und Unternehmenspraxis*. Springer Berlin Heidelberg, S. 121–148. DOI: 10.1007/978-3-662-65509-2\_8. URL: [https://doi.org/10.1007/978-3-662-65509-2\\_8](https://doi.org/10.1007/978-3-662-65509-2_8) [siehe S. 65].
- KOLB, Tobias [o. D.] »Entwicklung eines Convolutional Neural Network zur Handschrifterkennung«. In: *Angewandtes maschinelles Lernen–SS2019* [], S. 28 [siehe S. 47].
- KOMZSIK, Louis [2003]. *The Lanczos method: evolution and application*. SIAM [siehe S. 37].
- KORHONEN, Jari und Junyong YOU [2012]. »Peak signal-to-noise ratio revisited: Is simple beautiful?« In: *2012 Fourth International Workshop on Quality of Multimedia Experience*. IEEE, S. 37–38 [siehe S. 68].

- KRIZHEVSKY, Alex, Ilya SUTSKEVER und Geoffrey E HINTON [2012]. »Imagenet classification with deep convolutional neural networks«. In: *Advances in neural information processing systems*, S. 1097–1105 [siehe S. 85].
- KURZWEIL, Ray u. a. [1990]. *The age of intelligent machines*. Bd. 580. MIT press Cambridge [siehe S. 14].
- LABACH, Alex, Hojjat SALEHINEJAD und Shahrokh VALAEE [2019]. *Survey of Dropout Methods for Deep Neural Networks*. arXiv: 1904.13310 [cs.NE] [siehe S. 124].
- LEARNING, Deep [2020]. »Deep learning«. In: *High-dimensional fuzzy clustering* [siehe S. 83].
- LECUN, Yann, Yoshua BENGIO und Geoffrey HINTON [2015]. »Deep learning«. In: *nature* 521.7553, S. 436–444 [siehe S. 83].
- LECUN, Yann u. a. [1998]. »Gradient-based learning applied to document recognition«. In: *Proceedings of the IEEE* 86.11, S. 2278–2324 [siehe S. 85].
- LEE, Joosung u. a. [2020]. *AD-VO: Scale-Resilient Visual Odometry Using Attentive Disparity Map*. arXiv: 2001.02090 [cs.CV] [siehe S. 92].
- LI, C. und Z. WANG [2020]. »Convolutional Neural Networks for Image Scaling in Real-Time Applications«. In: *Journal of Visual Communication and Image Representation* [siehe S. 21].
- LI, Hao u. a. [2018]. »Visualizing the loss landscape of neural nets«. In: *Advances in neural information processing systems* 31 [siehe S. 50].
- LI, Juncheng, Zehua PEI und Tieyong ZENG [2021]. *From Beginner to Master: A Survey for Deep Learning-based Single-Image Super-Resolution*. arXiv: 2109.14335 [eess.IV] [siehe S. 64].
- LI, Zewen u. a. [Dez. 2022]. »A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects«. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12, S. 6999–7019. DOI: 10.1109/tnnls.2021.3084827. URL: {<https://doi.org/10.1109/tnnls.2021.3084827>} [siehe S. 47].

- LIU, Wei u. a. [2016]. »SSD: Single shot multibox detector«. In: *European conference on computer vision*. Springer, S. 21–37. doi: 10.1007/978-3-319-46448-0\_2 [siehe S. 76].
- LIU, Zhuang u. a. [2023]. *Dropout Reduces Underfitting*. arXiv: 2303.01500 [cs.LG] [siehe S. 136].
- LIU, Ziwei u. a. [2015]. »Deep learning face attributes in the wild«. In: *Proceedings of the IEEE international conference on computer vision*, S. 3730–3738 [siehe S. 49].
- MA, Xiang u. a. [2023]. *Deep Image Feature Learning with Fuzzy Rules*. arXiv: 1905.10575 [cs.CV] [siehe S. 81].
- MALLAT, Stephane G [1989]. »A theory for multiresolution signal decomposition: the wavelet representation«. In: *IEEE transactions on pattern analysis and machine intelligence* 11.7, S. 674–693 [siehe S. 62].
- MANTOVANI, Rafael G u. a. [2016]. »Hyper-parameter tuning of a decision tree induction algorithm«. In: *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, S. 37–42 [siehe S. 123].
- MARCELLIN, Michael W u. a. [2000]. »An overview of JPEG-2000«. In: *Proceedings DCC 2000. Data Compression Conference*. IEEE, S. 523–541 [siehe S. 19].
- MCCULLOCH, Warren S und Walter PITTS [1943]. »A logical calculus of the ideas immanent in nervous activity«. In: *The bulletin of mathematical biophysics* 5, S. 115–133 [siehe S. 13].
- MITTAL, Anish, Anush Krishna MOORTHY und Alan Conrad BOVIK [2012]. »No-reference image quality assessment in the spatial domain«. In: *IEEE Transactions on image processing* 21.12, S. 4695–4708 [siehe S. 67].
- MORAD, M, AI CHALMERS und PR O'REGAN [1996]. »The role of root-mean-square error in the geo-transformation of images in GIS«. In: *International Journal of Geographical Information Science* 10.3, S. 347–353 [siehe S. 71].

- MOZILLA CONTRIBUTORS [n.d.] *SVG (Scalable Vector Graphics)*. <https://developer.mozilla.org/en-US/docs/Web/SVG>. Accessed: March 9, 2023 [siehe S. 17].
- MUHDALIFAH, Mujastia Feliati [2022]. »Pooling comparison in CNN architecture for Java-nese script classification«. In: *International Journal of Informatics and Computation* 3.2, S. 15–22 [siehe S. 90].
- MURPHY, John [2016]. »An overview of convolutional neural network architectures for deep learning«. In: *Microway Inc*, S. 1–22 [siehe S. 98, 99].
- NASA/JPL-CALTECH [Feb. 1990]. *The Pale Blue Dot is a photograph of Earth taken Feb. 14, 1990, by NASA's Voyager 1 at a distance of 6 billion kilometers from the Sun. The image inspired the title of scientist Carl Sagan's book, "Pale Blue Dot: A Vision of the Human Future in Space," in which he wrote: "Look again at that dot. That's here. That's home. That's us."* <https://solarsystem.nasa.gov/resources/536/voyager-1s-pale-blue-dot/> [siehe S. 8].
- NEWELL, Allen [1963]. *Learning, Generality and Problem-Solving*. Rand Corporation [siehe S. 12].
- NGUYEN, Huy Hoang u. a. [Jan. 2021]. »YOLO Based Real-Time Human Detection for Smart Video Surveillance at the Edge«. In: *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*. IEEE. DOI: 10.1109/icce48956.2021.9352144. URL: <https://doi.org/10.1109/icce48956.2021.9352144> [siehe S. 49].
- OEHLRICH, Carl-Werner u. a. [1992]. »Ein Transputersystem mit verteiltem Bildspeicher für Echtzeit-Computergrafik und Bildverarbeitung«. In: *Parallele Datenverarbeitung mit dem Transputer: 3. Transputer-Anwender-Treffen TAT'91, Aachen, 17.–18. September 1991*. Springer, S. 170–177 [siehe S. 22].
- O'SHEA, Keiron und Ryan NASH [2015]. *An Introduction to Convolutional Neural Networks*. arXiv: 1511.08458 [cs.NE] [siehe S. 47].

- PAN, Sinno J. und Qiang YANG [2010]. »A Survey on Transfer Learning«. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, S. 1345–1359. DOI: 10.1109/TKDE.2009.191 [siehe S. 127].
- PARK, Jangsoo, Jongseok LEE und Donggyu SIM [Sep. 2020]. »Low-complexity CNN with 1D and 2D filters for super-resolution«. In: *Journal of Real-Time Image Processing* 17.6, S. 2065–2076. DOI: 10.1007/s11554-020-01019-1. URL: <https://doi.org/10.1007/s11554-020-01019-1> [siehe S. 54].
- PARSANIA, Pankaj S und Paresh V VIRPARIA [2016]. »A comparative analysis of image interpolation algorithms«. In: *International Journal of Advanced Research in Computer and Communication Engineering* 5.1, S. 29–34 [siehe S. 35].
- PATRO, S.Gopal Krishna und Kishore Kumar SAHU [März 2015]. »Normalization: A Preprocessing Stage«. In: *IARJSET*, S. 20–22. DOI: 10.17148/iarjset.2015.2305. URL: <https://doi.org/10.17148/iarjset.2015.2305> [siehe S. 110].
- POGGIO, Tomaso u. a. [2018]. *Theory of Deep Learning III: explaining the non-overfitting puzzle*. arXiv: 1801.00173 [cs.LG] [siehe S. 135].
- PREPRESSURE [n.d.] *Prepressure Library: File Formats*. <https://www.prepressure.com/library/file-formats/>. Accessed: March 9, 2023 [siehe S. 19].
- QIAO, Yuchuan u. a. [2015]. »Fast automatic step size estimation for gradient descent optimization of image registration«. In: *IEEE transactions on medical imaging* 35.2, S. 391–403 [siehe S. 140].
- QUINT, Antoine [2003]. »Scalable vector graphics«. In: *IEEE MultiMedia* 10.3, S. 99–102 [siehe S. 17].
- RABBANI, Majid und Rajan JOSHI [2002]. »An overview of the JPEG 2000 still image compression standard«. In: *Signal processing: Image communication* 17.1, S. 3–48 [siehe S. 19].
- RAHM, Erhard, Hong Hai DO u. a. [2000]. »Data cleaning: Problems and current approaches«. In: *IEEE Data Eng. Bull.* 23.4, S. 3–13 [siehe S. 109].

- RAHMAN, Tawsifur u. a. [2020]. »Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray«. In: *Applied Sciences* 10.9, S. 3233 [siehe S. 87].
- RAMACHANDRAM, Dhanesh und Graham W TAYLOR [2017]. »Deep multimodal learning: A survey on recent advances and trends«. In: *IEEE signal processing magazine* 34.6, S. 96–108 [siehe S. 12].
- REBALA, Gopinath u. a. [2019]. »Machine learning definition and basics«. In: *An introduction to machine learning*, S. 1–17 [siehe S. 82].
- REHMAN, Abdul, Kai ZENG und Zhou WANG [2015]. URL: <https://ece.uwaterloo.ca/~z70wang/publications/HVEI15.pdf>.
- ROBBINS, Scott [2022]. »Machine Learning, Mass Surveillance, and National Security: Data, Efficacy, and Meaningful Human Control«. In: *The Palgrave Handbook of National Security*, S. 371–388 [siehe S. 86].
- ROSEBROCK, Adrian [2021]. *Convolutional Neural Networks (CNNs) and Layer Types*. URL: <https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/> [siehe S. 98].
- RUDER, Sebastian [2016]. »An overview of gradient descent optimization algorithms«. In: *arXiv preprint arXiv:1609.04747* [siehe S. 140].
- SCHLEMPER, Jo u. a. [2017]. »A deep cascade of convolutional neural networks for MR image reconstruction«. In: *Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, June 25–30, 2017, Proceedings* 25. Springer, S. 647–658 [siehe S. 139].
- SCHMIDHUBER, Jürgen [2015]. »Deep learning in neural networks: An overview«. In: *Neural networks* 61, S. 85–117 [siehe S. 83].
- SHAH, Vruddhi u. a. [2021]. »Diagnosis of COVID-19 using CT scan images and deep learning techniques«. In: *Emergency radiology* 28, S. 497–505 [siehe S. 78].

- SHANG, Xiwu u. a. [2018]. »Color-sensitivity-based combined PSNR for objective video quality assessment«. In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.5, S. 1239–1250 [siehe S. 70].
- SHEIKH, Hamid R und Alan C BOVIK [2006]. »Image information and visual quality«. In: *IEEE Transactions on image processing* 15.2, S. 430–444 [siehe S. 67, 69].
- SHORTEN, Conor und Taghi M. KHOSHGOFTAAR [2019]. »Data Augmentation for Deep Learning: A Comprehensive Survey«. In: *Journal of Big Data* 6.1, S. 1–48. DOI: 10.1186/s40537-019-0197-0 [siehe S. 114].
- SHUMAN, David I, Mohammad Javad FARAJI und Pierre VANDERGHEYNST [2015]. »A multiscale pyramid transform for graph signals«. In: *IEEE Transactions on Signal Processing* 64.8, S. 2119–2134 [siehe S. 62].
- SIMONCELLI, Eero P und Edward H ADELSON [1996]. »Noise removal via Bayesian wavelet coring«. In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Bd. 1. IEEE, S. 379–382 [siehe S. 62].
- SIMONYAN, Karen und Andrew ZISSERMAN [2014]. »Very deep convolutional networks for large-scale image recognition«. In: *arXiv preprint arXiv:1409.1556* [siehe S. 85].
- SPACERIDE [2023]. *So funktioniert Spracherkennung mit Deep Learning*. URL: <https://www.spaceride.de/blog/so-funktioniert-spracherkennung-mit-deep-learning> [siehe S. 77].
- SPECHT, Donald F [1990]. »Probabilistic neural networks and the polynomial adaline as complementary techniques for classification«. In: *IEEE Transactions on Neural Networks* 1.1, S. 111–121 [siehe S. 13].
- STARCK, Jean-Luc, Fionn D MURTAGH und Albert BIJAOUI [1998]. *Image processing and data analysis: the multiscale approach*. Cambridge University Press [siehe S. 62].
- STOCKMAN, George und Linda G SHAPIRO [2001]. *Computer vision*. Prentice Hall PTR [siehe S. 86].

- STREIJL, Robert C, Stefan WINKLER und David S HANDS [2016]. »Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives«. In: *Multi-media Systems* 22.2, S. 213–227 [siehe S. 72].
- TANDEL, Nishtha H, Harshadkumar B PRAJAPATI und Vipul K DABHI [2020]. »Voice recognition and voice comparison using machine learning techniques: A survey«. In: *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, S. 459–465 [siehe S. 87].
- TAYLOR, Luke und Geoff NITSCHKE [Nov. 2018]. »Improving Deep Learning with Generic Data Augmentation«. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. DOI: 10.1109/ssci.2018.8628742. URL: <https://doi.org/10.1109/ssci.2018.8628742> [siehe S. 112].
- TECH-LIB [2020]. *Image Scaling Definition*. <http://www.dante.de> [siehe S. 9].
- THANAPOL, Panissara u. a. [2020]. »Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition«. In: *2020-5th International Conference on Information Technology (InCIT)*. IEEE, S. 300–305 [siehe S. 103, 112].
- TIAN, Wei u. a. [Apr. 2013]. »Auto-normalization algorithm for robotic precision drilling system in aircraft component assembly«. In: *Chinese Journal of Aeronautics* 26.2, S. 495–500. DOI: 10.1016/j.cja.2013.02.029. URL: <https://doi.org/10.1016/j.cja.2013.02.029> [siehe S. 111].
- TRIANTAFYLLIDOU, Danai und Anastasios TEFAS [Okt. 2017]. »A Fast Deep Convolutional Neural Network for Face Detection in Big Visual Data«. In: S. 61–70. ISBN: 978-3-319-47897-5. DOI: 10.1007/978-3-319-47898-2\_7 [siehe S. 95].
- W3C [2003]. *Portable Network Graphics (PNG) Specification (Second Edition)*. <https://www.w3.org/TR/png/>. Accessed: March 9, 2023 [siehe S. 16].

- WANG, Hongda u. a. [Dez. 2018]. »Deep learning enables cross-modality super-resolution in fluorescence microscopy«. In: *Nature Methods* 16.1, S. 103–110. DOI: 10.1038/s41592-018-0239-0. URL: <https://doi.org/10.1038/s41592-018-0239-0> [siehe S. 64].
- WANG, Jiaming u. a. [2022]. »From artifact removal to super-resolution«. In: *IEEE Transactions on Geoscience and Remote Sensing* 60, S. 1–15 [siehe S. 53].
- WANG, Wei und Jianxun GANG [2018]. »Application of convolutional neural network in natural language processing«. In: *2018 international conference on information Systems and computer aided education (ICISCAE)*. IEEE, S. 64–70 [siehe S. 96].
- WANG, Yingying u. a. [2020]. »The influence of the activation function in a convolution neural network model of facial expression recognition«. In: *Applied Sciences* 10.5, S. 1897 [siehe S. 104].
- WANG, Zhou u. a. [2004a]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE transactions on image processing* 13.4, S. 600–612 [siehe S. 35, 53, 66, 73].
- WANG, Zhou u. a. [2004b]. »Image quality assessment: from error visibility to structural similarity«. In: *IEEE Transactions on Image Processing* 13.4, S. 600–612. ISSN: 1941-0042. DOI: 10.1109/TIP.2003.819861 [siehe S. 69].
- WANTO, Anjar u. a. [2018]. »Analysis of Standard Gradient Descent with GD Momentum And Adaptive LR for SPR Prediction«. In: [Siehe S. 141].
- WHUBER [Sep. 2011]. *What is Lanczos resampling?* <https://gis.stackexchange.com/a/14361> [siehe S. 10, 37].
- WIKIPEDIA [2023a]. *Peak signal-to-noise ratio — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-May-2023]. URL: <http://en.wikipedia.org/w/index.php?title=Peak%20signal-to-noise%20ratio&oldid=1145027401> [siehe S. 67].
- [2023b]. *Structural similarity — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-May-2023]. URL: <http://en.wikipedia.org/w/index.php?title=Structural%20similarity&oldid=1142355078> [siehe S. 68].

- WU, Bichen u. a. [2017]. »Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, S. 129–137 [siehe S. 49].
- XU, Jiajun, Xudong JIANG und Zulin ZHANG [2017]. »Efficient and accurate image super-resolution via recurrent mixture density network«. In: *IEEE Transactions on Image Processing* 26.7, S. 3187–3201. DOI: 10.1109/TIP.2017.2699923 [siehe S. 71, 139].
- YE, Peng und David DOERMANN [2012]. »No-reference image quality assessment using visual codebooks«. In: *IEEE Transactions on Image Processing* 21.7, S. 3129–3138 [siehe S. 73].
- YOSINSKI, Jason u. a. [2014]. »How Transferable are Features in Deep Neural Networks?« In: *Advances in Neural Information Processing Systems*, S. 3320–3328 [siehe S. 127].
- YOU, Kaichao u. a. [2021]. »Logme: Practical assessment of pre-trained models for transfer learning«. In: *International Conference on Machine Learning*. PMLR, S. 12133–12143 [siehe S. 130].
- YUAN, Pengcheng u. a. [2020]. *HS-ResNet: Hierarchical-Split Block on Convolutional Neural Network*. DOI: 10.48550/ARXIV.2010.07621. URL: <https://arxiv.org/abs/2010.07621> [siehe S. 47, 48].
- ZHANG, Kai u. a. [2015]. »Learning a single convolutional super-resolution network for multiple degradations«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 3262–3270. DOI: 10.1109/CVPR.2015.7298958 [siehe S. 76, 134].
- ZHANG, Richard, Phillip ISOLA und Alexei A. EFROS [2018]. »Split-brain autoencoders: Unsupervised learning by cross-channel prediction«. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, S. 1058–1067. DOI: 10.1109/CVPR.2018.00120 [siehe S. 76].

- ZHANG, Y. und Q. LIU [2018]. »Scalable Image Processing Techniques for Real-Time Applications«. In: *IEEE Transactions on Image Processing* [siehe S. 21].
- ZHANG, Y. u. a. [2019]. »Efficiency-Driven Image Scaling Method Selection Based on Deep Neural Networks«. In: *IEEE Transactions on Image Processing* 28.1, S. 424–439. DOI: 10.1109/TIP.2018.2889740 [siehe S. 74].
- ZHONG, Zhun u. a. [Apr. 2020]. »Random Erasing Data Augmentation«. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.07, S. 13001–13008. DOI: 10.1609/aaai.v34i07.7000. URL: <https://doi.org/10.1609/aaai.v34i07.7000> [siehe S. 112].
- ZHU, Jun-Yan u. a. [2017]. »Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks«. In: *Computer Vision (ICCV), 2017 IEEE International Conference on* [siehe S. 56].
- ZHU, Qiuyu u. a. [2019]. »A new loss function for CNN classifier based on predefined evenly-distributed class centroids«. In: *IEEE Access* 8, S. 10888–10895 [siehe S. 106].