DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

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 Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Lukas Hörnle Marc Gökce

Abgabedatum 22.05.2023

Bearbeitungszeitraum	TODO raussuchen
Matrikelnummer	TODO
Kurs	TINF20B4
Ausbildungsfirma	CAS Software AG
1	1
	Karlsruhe
Gutachter der Studienakademie	Ralph Lausen

Erklärung

Ich versichere hiermit, dass ich meine Studienarbeitmit 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 Formulierungzu 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.

Zusammenfassung

Dieses LaTeX-Dokument kann als Vorlage für einen Praxis- oder Projektbericht, eine Studien- oder Bachelorarbeit dienen.

Zusammengestellt von Prof. Dr. Jürgen Vollmer <juergen.vollmer@dhbw-karlsruhe.de> `https://www.karlsruhe.dhbw.de`. Die jeweils aktuellste Version dieses LaTeX-Paketes ist immer auf der *FAQ-Seite* des Studiengangs Informatik zu finden: `https://www.karlsruhe.dhbw.de/inf/studienverlauf-organisatorisches.html` → *Formulare und Vorlagen.*

Stand `$Date: 2020/03/13 15:07:45 $`

Inhaltsverzeichnis

Kapitel 1

Einleitung

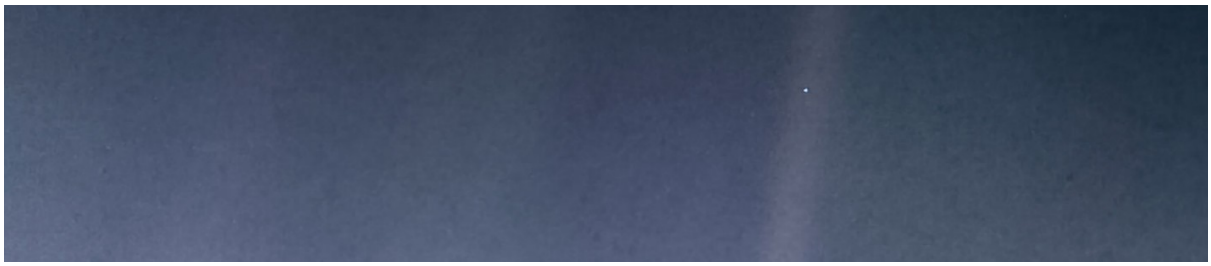

Abbildung 1.1: “The Pale Blue Dot” Feb. 14, 1990, by NASA[1].

”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

[1]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 (PPIs).

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[2][3] 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 größenunabhängig 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 (CNNs) oder Generative Adversarial Networks Generative Adversarial Network (GANs), die neue Pixel aus einem trainierten Modell erzeugen. Diese neuen Methoden sind komplex und benötigen mehr Rechenaufwand,

[2]Tech-Lib [2020]. *Image Scaling Definition*. `http://www.dante.de`.

[3]Wikipedia Contributors [2023]. *Bildskalierung Definition*. `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 schafft einen Überblick über klassische und moderne Skalierungsverfahren sowie deren ihre Vor- und Nachteile. 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 Firschungsvorhaben fokussiert sich auf die systematische Analyse unterschiedlicher Verfahren zur Skalierung von Bildern, um eine eine fundierte Empfehlung hinsichtlich der optimalen Methode abzugeben. Die Berücksichtigung von Aspekten wie Algorithmuskomplexität, 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 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.

Kapitel 2

Grundlagen der Bildverarbeitung und der Skalierung von Bildern

2.1 Einblick in die Bildverarbeitung

Historie, Entwicklung, aktueller Stand und mögliche Entwicklungen.

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. Das Verfahren der Interpolation ermöglicht es neue Pixelwerte auf Basis vorgegebener Werte zu berechnen. Die Skalierung ist eine Anpassung der Bildgröße durch das Ändern der Anzahl von Pixeln oder der Auflösung. Im Kontext der Bildverarbeitung wird Interpolation häufig verwendet, um die Größe von Bildern zu ändern, ohne dass dabei die Anzahl der Pixel verändert wird. Dazu werden neue Pixelwerte berechnet, indem vorhandene Pixelwerte interpoliert werden. Die Wahl der Interpolationsmethode hat einen großen Einfluss auf die Qualität des interpolierten Bildes. In der Bildverarbeitung gibt es verschiedene Interpolationsmethoden, wie z.B.

Nearest-Neighbor-Interpolation, Bilineare Interpolation oder Bikubische Interpolation. Skalierung hingegen verändert die Größe eines Bildes, indem die Anzahl der Pixel oder die Auflösung verändert wird. Im Gegensatz zur Interpolation wird die Anzahl der Pixel bei der Skalierung verändert, um das Bild kleiner oder größer zu machen. Auch hier hat die Wahl der Skalierungsmethode einen großen Einfluss auf die Qualität des resultierenden Bildes.

2.2.2 Bildformate

Bilder können allgemein als zweidimensionaler Array dargestellt werden. Historisch gesehen gab es jedoch viele unterschiedliche Formate für Bilder. Zunächst erschufen unterschiedliche Softwareentwickler im Bereich der Bildverarbeitung häufig ihre eigenen Formate.[1] Einheitliche Standards, wie sie heute im Einsatz sind, etablierten sich erst später. Ein Vorreiter der modernen Bildformate ist das "Portable Network Graphics Format"[2], 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.[3]

Portable Network Graphics Format

Portable Network Graphics Format (PNGs) 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 Kompression.[4] PNG unterstützt Vollfarbbilder, Grauwertbilder sowie Indexbilder.[5] Der

[1] Wilhelm Burger und Mark James Burge [2009]. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag.

[2] Thomas Boutell u. a. [o. D.] *PNG (Portable Network Graphics Format) Version 1.0*. Techn. Ber. RFC 2083.

[3] Wilhelm Burger und Mark James Burge [2009]. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag.

[4] Thomas Boutell [1997]. *Png (portable network graphics) specification version 1.0*. Techn. Ber.

[5] Wilhelm Burger u. a. [2015]. »Digitale Bilder«. In: *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*, S. 1–24.

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 kompekte Komprimmierung des PNG-Formats hat dafür gesorgt, dass der Standard im Internet eine hohe Beliebtheit genießt.[6]

JPG-Format

Joint Photographic Experts Group (JPGs) Hier noch text bittö.

Scalable Vector Graphics

> The main idea motivating (SVGs) was simple: to create a generic document-oriented solution for graphics that can be adapted to modern media "[7]

Scalable Vector Graphics (SVG) steht für ein Format, das Vektorgrafiken basierend auf () 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.[8] Da SVG auf XML basiert, kann es auch mit anderen

[6] ^**w3c_png**.

[7] Iris Fibinger [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`.

[8] Antoine Quint [2003]. »Scalable vector graphics«. In: *IEEE MultiMedia* 10.3, S. 99–102.

Webtechnologien wie HTML, (CSS) und JavaScript integriert werden.[9]

Weitere Standards

(GIFs)

Ein Format für animierte Rastergrafiken mit einer begrenzten Farbpalette von 256 Farben. Es verwendet eine verlustfreie Kompression, die aber nicht sehr effizient ist. Es ist geeignet für einfache Animationen und Grafiken mit wenigen Farben.

(TIFFs)

Ein Format für hochauflösende Rastergrafiken ohne Kompression oder mit verlustfreier Kompression. Es wird oft im Druckbereich verwendet, da es viele Optionen für Farbmanagement und Metadaten bietet. Es ist aber nicht sehr kompatibel mit Webbrowsern

(PSDs)

Ein Format für Photoshop-Dokumente, das alle Ebenen, Masken, Effekte und andere Informationen speichert. Es ermöglicht eine umfangreiche Bearbeitung von Rastergrafiken, ist aber nur mit Photoshop kompatibel.

(BMPs)

Ein Format für unkomprimierte Rastergrafiken mit hoher Qualität. Es wird selten verwendet, da es sehr große Dateien erzeugt und keine Transparenz oder andere Funktionen unterstützt.

(EPSs)

Ein Format für vektorbasierte Grafiken, das Kurven, Texte und andere Elemente

[9] Mozilla Contributors [n.d.] *SVG (Scalable Vector Graphics).* `https://developer.mozilla.org/en-US/docs/Web/SVG`. Accessed: March 9, 2023.

speichert. Es kann skaliert werden ohne Qualitätsverlust und wird oft im Druckbereich verwendet. Es ist aber nicht sehr kompatibel mit Webbrowsern oder anderen Programmen.

Weiterhin gibt es unzählige Standards um Grafiken darzustellen. Diese übersteigen jedoch den Umfang dieser Arbeit.[10][11][12][13][14][15]

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.

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

[11] 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.

[12] 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.

[13] Michael W Marcellin u. a. [2000]. »An overview of JPEG-2000«. In: *Proceedings DCC 2000. Data Compression Conference*. IEEE, S. 523–541.

[14] Encyclopedia Britannica [n.d.] *JPEG*. https://www.britannica.com/technology/JPEG. Accessed: March 9, 2023.

[15] 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.

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 Geneauigkeit 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 eine sehr schnelle Verarbeitung notwendig ist, um zeitkritische Anwendungen wie z.B. autonomes Fahren oder Augmented Reality zu realisieren. Bei diesen Aufgaben müssen Bilder in Echtzeit erfasst, verarbeitet und angezeigt werden, um eine reibungslose Funktionalität zu gewährleisten. Die Echtzeitverarbeitung von Bildern erfordert in der Regel eine hohe Rechenleistung, um die Daten schnell genug zu verarbeiten. Hierbei kommen spezielle Algorithmen zum Einsatz, die für eine effiziente Verarbeitung sorgen. 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 oft spezielle Systeme zum Einsatz, die für eine schnelle Übertragung und parallele Verarbeitung von Daten optimiert sind.[16]

2.4 Anwendungen von Skalierungsmethoden

Die Anwendung von Skalierungsmethoden bietet eine Vielzahl ungewöhnlicher, aber äußerst nützlicher Anwendungen in verschiedenen Sektoren. In der Landwirtschaft können diese Methoden dazu beitragen, die Gesundheit von Pflanzen zu überwachen und den

[16] 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.

Einsatz von Pestiziden zu optimieren. Durch die Analyse von Pflanzenbildern können frühzeitig Krankheiten erkannt und gezielte Maßnahmen ergriffen werden, um Ernteausfälle zu minimieren. In der Archäologie 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 ermöglichen Skalierungsmethoden das virtuelle Anprobieren von Kleidung, was den Kunden die Möglichkeit gibt, Passform und Aussehen der Kleidungsstücke im Voraus zu überprüfen und das Online-Shopping-Erlebnis zu verbessern. Durch die Integration von Bildverarbeitung und Augmented Reality entsteht eine innovative und interaktive Shopping-Erfahrung. Insgesamt bieten Skalierungsmethoden in diesen verschiedenen Sektoren ungewöhnliche Anwendungen, die zu Effizienzsteigerungen, Erkenntnisgewinnen und verbesserten Kundenerfahrungen führen.

Kapitel 3

Klassische Skalierungsmethoden

3.1 Pixel-Verdopplung

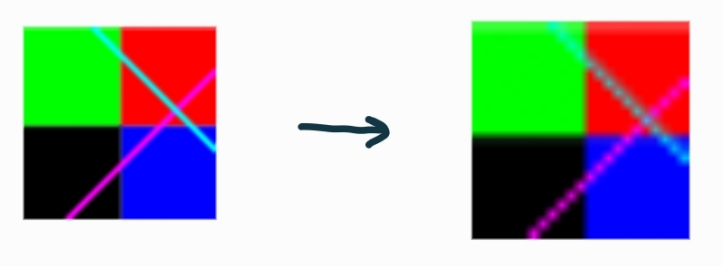

Abbildung 3.1: Beispielhafte Darstellung einer Skalierung durch PixelVerdopplung

Die Pixel-Verdopplung vergrößert das Bild, indem jeder Pixel 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 professinellen Anwendungen selten Gebrauch.[1]

[1]David C.C Wang, Anthony H Vagnucci und C.C Li [1983]. »Digital image enhancement: A survey«. 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. URL: https://www.sciencedirect.com/

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-dhbwka-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 origi
                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()
```

science/article/pii/0734189X83900610.

Die vorliegende Implementierung in Python beschreibt die Realisierung der Pixelverdopplungsklasse, welche in der Lage ist, ein größeres Bild zu erzeugen, indem die leeren Pixel mit demselben Pixelwert wie der nächste Nachbarpixel befüllt 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 "PixelVerdopplungërbt von der Klasse İmageünd besitzt einen Konstruktor, der den Pfad zum Bild und die Erweiterung "p2äls Argumente erhält. Die Methode "manipulateïst 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 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 Nearest-Neighbor-Interpolation 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.

```
import numpy as np
import cv2

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

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



image = cv2.imread('example_image.jpg')
scaled_image = nearest_neighbor_interpolation(image, 2)
cv2.imshow(image)
cv2.imshow(scaled_image)
```

[2]

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 Audgabebild kein Vielfaches ist.

[2]Nan Jiang und Luo Wang [2015]. »Quantum image scaling using nearest neighbor interpolation«. In: *Quantum Information Processing* 14, S. 1559–1571.

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.

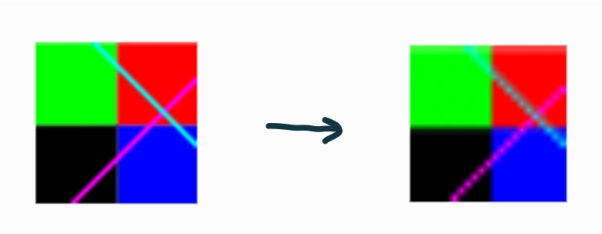

Abbildung 3.2: So sieht bilineare Verdopplung aus.

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

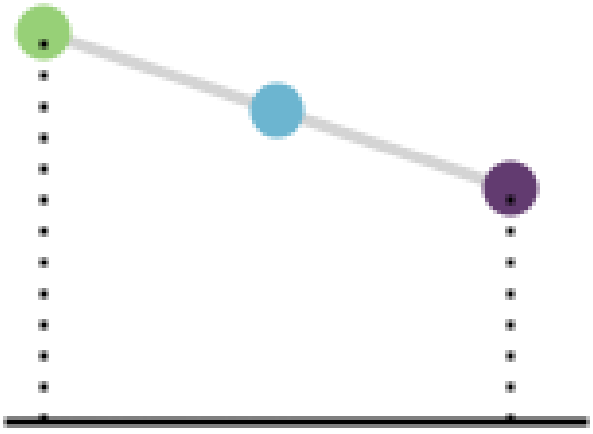

Abbildung 3.3: Graph über bilineare Skalierung.

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 Bicubic oder Lanczos eine bessere Wahl sein.

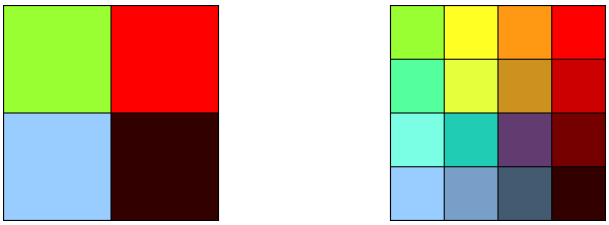

Abbildung 3.4: Beispielgrafik zur Pixelverdopplung.

```
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 substract o
        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] 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.

3.4 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.

Abbildung 3.5: Bikubische Interpolation

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.

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 hinterlasst wenige scharfe Kanten.

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

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.

Ein Beispiel für die Implementierung der bikubischen Interpolation in Python ist der folgende Code:

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 erechnet. 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.

```
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)
```

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

```
import math
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 - 1)))
                            )
                        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()
```

3.4.5 Anwendungen

Die bikubische Interpolation wird in verschiedenen Anwendungen der Bildverarbeitung eingesetzt und bietet bestimmte Vorteile 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. 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. Durch die Verwendung von zusätzlichen Pixeln und einer gewichteten 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. 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.

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:

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

[4]

Das Lanczos-Kernel hat eine kompakte Trägerfunktion. Eine kompakte Träägerformel 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. Die Interpolationsmethode verwendet diese Stützstellen als Ausgangspunkt, um eine Schätzung des Signals an anderen Orten zu berechnen. Seien $x_1, x_2, \ldots, x_n$ die Stützstellen des Signals und $y_1, y_2, \ldots, y_n$ die zugehörigen Abtastungen. Die Interpolationsfunktion $s(x)$ kann dann wie folgt berechnet werden:

[4] Claude E Duchon [1979]. »Lanczos filtering in one and two dimensions«. In: *Journal of Applied Meteorology and Climatology* 18.8, S. 1016–1022.

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

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 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 einfachere Interpolationsmethoden, da sie eine Berechnung für jeden Pixel im skalierten Bild erfordert.[5]

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
```

[5] 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.

```
import math

class LanczosInterpolation(Image):
    def __init__(self, path):
        extend = "lcz"
        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 = 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 zu Vor- und Nachteilen von Techniken in der Bildverarbeitung

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 "Treppeneffektïm 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

Erläuterung der Kriterien, die zur Bewertung der Leistung von Bildinterpolationsverfahren verwendet werden, einschließlich der visuellen Qualität, der Genauigkeit und der Berechnungseffizienz.

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.

Verfahren	Geschwindigkeit	Qualität	Brauchbarkeit
Nearest Neighbour	schnell	mäßig	blockige Ergebnisse
Bilineare Interpolation	mittel	gut	linearer Effekt
Bicubische Interpolation	langsam	sehr gut	glättender Effekt
Lanczos-Interpolation	langsam	sehr gut	glättender Effekt

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 Stapelverarbeitung.

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 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

4.1 Convolutional Neural Networks / Deep learning

4.1.1 Grundlagen von Convolutional Neural Networks (CNNs)

Convolutional Neural Networks CNNs 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. Räum.iche Daten, wie z.B. Bilder können bearbeitet, verarbeitet, erstellt und analysiert werden. CNNs bestehen aus mehreren Schichten von Neuronen, die so angeordnet sind, dass sie räumliche Beziehungen in den Daten erfassen können. Die grundlegende Idee hinter CNNs ist die Verwendung von Faltung (engl. convolution) anstelle der vollständigen Verbindung (engl. fully connected) zwischen den Schichten. Dies bedeutet, dass jedes Neuron in einer Schicht nur mit einem Teil des Eingangs verbunden ist, anstatt mit jedem Eingangsneuron. Diese Art der Verbindung spart Rechenleistung und ermöglicht eine effektivere sowie schnellere Verarbeitung von

großen Datensätzen.[1]

4.1.2 Architekturen von CNNs

Es gibt mehrere bekannte Architekturen von CNNs, darunter AlexNet, ResNet und Inception. AlexNet war das erste CNN, das auf einem großen Datensatz erfolgreich angewendet wurde. ResNet zeichnet sich durch seine Fähigkeit aus, sehr tiefe Netzwerke zu trainieren, ohne dass das Problem des Verschwindens des Gradienten auftritt. Inception wiederum ist für seine Fähigkeit bekannt, die Effizienz von CNNs durch die Verwendung von sogenannten Inception-Modulen zu erhöhen.

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 eingesetzt.[2][3]oshea2015introduction)

[1] 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]`.

[2] 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.

[3] (.

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 ????????, bei denen das Modell auf einem bereits trainierten CNN basieren kann, das auf ?????? trainiert wurde.

$$\text{Net} = \text{torchvision.models.segmentation.deeplabv3}_r esnet50(pretrained = True)$$

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 Eingabemerkmalen zu erfassen, wie z.B. das Verhalten von Objekten in einem Video. Zudem benötigen CNNs weiterhin viel Rechenleistung und Ressourcen[4]

4.2 Super Resolution

4.2.1 Grundlagen von Super Resolution

Super Resolution Super Resolution (SRs) 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 Videoanalyse Anwendung. 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

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

werden muss.[5]7115171)

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 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.

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 und der Verbesserung von Bildern für die Erkennung von Gesichtern und Objekten. In der Videoanalyse kann SR verwendet werden, um Videos zu stabilisieren, indem Bewegungsunschärfe reduziert und die Schärfe der Bilder verbessert wird. SR kann auch bei der Entschlüsselung von unscharfen und verschwommenen Bildern in Überwachungsaufnahmen 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

[5](.

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, gibt es immer noch Herausforderungen und zukünftige Forschungsziele, die erforscht werden müssen. Eine der Herausforderungen besteht darin, dass SR-Methoden häufig dazu neigen, Artefakte 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 neue SR-Methoden entwickeln, die sowohl effektiv als auch effgizient 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-Netzwerken, die weniger rechenaufwändig sind und schneller ausgeführt werden können. 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 Geselllschaft zu erfüllen.

4.3 Generative Adversarial Networks (GANs)

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 Netzwerken, einem Generator und einem Diskriminator. Der Generator erzeugt neue Daten, während der Diskriminator 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 Diskriminator 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, 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. Style Transfer ermöglicht es, das visuelle Erscheinungsbild

von Bildern zu verändern, indem der Stil von einem Bild auf ein anderes übertragen wird. GANs können auch verwendet werden, um Bilder mit höherer Auflösung oder besserer Qualität zu generieren, indem sie niedrig aufgelöste Bilder als Eingabe verwenden. In der Videoanalyse können GANs verwendet werden, um Videosequenzen zu generieren oder zu verbessern.

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 Netzwerk versucht, Bilder zu erzeugen, die von einem diskriminierenden Netzwerk nicht von echten Bildern unterschieden werden können. Das diskriminierende Netzwerk wird trainiert, um echte Bilder von den vom generativen Netzwerk 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 dasd Netzwerk Bilder erzeugt, die vom diskriminierenden Netzwerk nicht als gefälscht erkannt werden. Der Verlust des diskriminierenden Netzes wird minimiert, wenn das Netzwerk in der Lage ist, besonders zuverlässig und schneöö 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

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 errleichtert werden. Ein weiteres Problem bei der Verwendung von GANs ist, dass sie möglicherweise nicht fair sind. GANs können aufgrund ihrer Lernmethode unbewusste Vorurteile aufnehmen und in ihren generierten Ergebnissen widerspiegeln. 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 Multiscale-Skalierung

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"[6]

[6]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*

4.4.1 Grundlagen von Multiscale-Skalierung

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 Skala erhalten bleiben sollten. Das heißt, dass dieselben Merkmale oder Strukturen auf verschiedenen Skalen erkennbar sein sollten.[7]

4.4.2 Methoden zur Multiskalenanalyse

Es gibt verschiedene Techniken zur Multiskalenanalyse, darunter Wavelet-Transformation und pyramidenartige Strukturen. Wavelet-Transformation ist eine Methode zur Analyse und Synthese von Signalen oder Bildern auf verschiedenen Skalen. Dabei wird das Signal oder Bild auf verschiedene Skalen und Frequenzen zerlegt, um Informationen auf verschiedenen Skalen zu erhalten. Pyramidenartige Strukturen sind eine weitere Methode zur Multiskalenanalyse, die in der Bildverarbeitung häufig verwendet wird. Dabei wird das Signal oder Bild auf verschiedenen Skalen durch wiederholte Subsampling- und Filterungsoperationen reduziert. Auf jeder Ebene der Pyramide wird das Signal oder Bild auf eine kleinere Größe reduziert, um Informationen auf verschiedenen Skalen zu erhalten.[8]

medicine 2.1, S. 115.

[7] 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.

[8] Peter J Burt und Edward H Adelson [1987]. »The Laplacian pyramid as a compact image code«. In: *Readings in computer vision*. Elsevier, S. 671–679; Seungjun Nah, Tae Hyun Kim und Kyoung Mu Lee [2017]. »Deep multi-scale convolutional neural network for dynamic scene deblurring«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 3883–3891.

4.4.3 Methoden zur Multiskalenanalyse

Es gibt verschiedene Methoden zur Multiskalenanalyse, darunter Wavelets und Pyramiden. 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.[9]

4.4.4 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 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 (MSDN), das eine skalierbare Architektur für die Bilderkennung bietet, die auf mehreren Skalen arbeiten kann.

[9] 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.

4.4.5 Limitationen 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.[10] 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.

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

[10] 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.

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. Ein weiteres Problem ist die Interpretierbarkeit 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. 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 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. Zukünftige Forschung sollte darauf abzielen, die verschiedenen Methoden zu kombinieren und die Interpretierbarkeit von Deep-Learning-Modellen zu verbessern.

Kapitel 5

Evaluation von Skalierungsmethoden

5.1 Qualitätsmetriken zur Bewertung 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.[1]

Definition und Berechnung von PSNR! (PSNR!)

Die Definition von PSNR lautet wie folgt:

$$PSNR = 20 \log_{10} \left(\frac{Peak^2}{MSE} \right) \tag{5.1}$$

wobei Peak der höchste mögliche Wert des Signals ist, der meist auf 255 bei 8-Bit-Graustufenbildern oder 65535 bei 16-Bit-Graustufenbildern gesetzt wird. Der Mean Squared Error (MSE) ist definiert als der Durchschnitt der quadrierten Unterschiede zwischen jedem Pixel des Originalbildes und dem rekonstruierten Bild.

Formel MSE: MSE = (1/(m*n))*sum(sum((f-g).^2))

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. 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, 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 von Bildrekonstruktionen zu bewerten und zu vergleichen.[2]

[1] 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; Hamid R Sheikh und Alan C Bovik [2006]. »Image information and visual quality«. In: *IEEE Transactions on image processing* 15.2, S. 430–444; 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.

[2] 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.

5.1.2 Structural Similarity Index (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:

(5.2)

[3]

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.[4]

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.[5]

[3] Ming-Jun Chen und Alan C Bovik [2011]. »Fast structural similarity index algorithm«. In: *Journal of Real-Time Image Processing* 6, S. 281–287.

[4] 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.

[5] 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.

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-Metrik eine ungenaue Bewertung liefern.[6]

5.1.3 Mean Opinion Score (MOS)

Der Mean Opinion Score (MOSs) 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.[7]

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 () 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 wichtigeWerkzeuge für diew Bewertung und den Vergleich von Skalierungsmelthoden in der Bildverarbeitung. Es ist jedoch wichtig zu beachten, dass keine einzelne Metrik alle Aspekte der Bildqualität abdeckt. Eine umfassende Bewertung sollte mehrere Metriken

[6] 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.

[7] Hamid R Sheikh und Alan C Bovik [2006]. »Image information and visual quality«. In: *IEEE Transactions on image processing* 15.2, S. 430–444.

kombinieren und auch die subjektive Wahrnehmung[8]

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 (FLOPS). Eine genauere Messung kann durch die Verwendung von Benchmarks erreicht werden, die es ermöglichen, verschiedene Skalierungsmethoden auf derselben Hardware zu vergleichen. 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 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.[9]

[8] 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.

[9] 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; 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.

5.1.6 Root-Mean-Square Error (RMSE)

Der Root-Mean-Square Error (RMSEs) 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.[10]

5.2 Kriterien zur Auswahl der optimalen 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. 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

[10] 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, S. 347–353.

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. 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. 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.[11]

[11]Michael P Eckert und Andrew P Bradley [1998]. »Perceptual quality metrics applied to still image compression«. In: *Signal processing* 70.3, S. 177–200; 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; 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.

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 Bildverarbeitung. 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.[12]

5.2.3 Zukunftsaussichten und Herausforderungen

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

[12] 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; 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; 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.

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[13]

[13]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; 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; 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; 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

Zusammenfassung und Ausblick

6.1 Auswertung der Ergebnisse

6.2 Diskussion offener Fragen und zukünftiger Forschungsbedarf

Kapitel 7

Einführung in Deep Learning

Deep Learning ist eine Unterkategorie des maschinellen Lernens und beschäftigt sich mit der Entwicklung und Anwendung von neuronalen Netzwerken 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 in Smart Speakern wie Amazon Echo und Google Home, sowie die automatisierte Diagnose von medizinischen Bildern wie CT-Scans und MRT-Bildern.

7.1 Motivation hinter 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. 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 anzupassen und das Modell an die Daten anzupassen. Es gibt auch Techniken wie Dropout und Data Augmentation, die dazu beitragen können, Overfitting zu reduzieren. 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.

7.2 Struktur de Arbeit

Diese Studienarbeit ist in drei 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.

Im dritten Abschnitt werden die Ergebnisse und Beiträge dieses Papiers skizziert. Dies beinhaltet eine Zusammenfassung der wichtigsten Erkenntnisse und Empfehlungen für das effektive Training von CNNs.

Kapitel 8

Grundlagen von Convolutional Neural Networks

8.1 Wie lernen Maschinen?

In diesem Abschnitt werden die fundamentalen Konzepte des maschinellen Lernens und des Deep Learnings erläutert. Das maschinelle Lernen bezieht sich auf einen Prozess, bei dem maschinelle Systeme mithilfe von statistischen Modellen und Algorithmen automatisch aus Erfahrung lernen, ohne dass eine explizite Programmierung erforderlich ist. Das 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.[1]

8.1.1 Definition von maschinellem Lernen

Das maschinelle Lernen ist ein Prozess, bei dem computergestützte Systeme mithilfe von statistischen Methoden und Algorithmen aus Erfahrung lernen, ohne dass eine explizite Programmierung erforderlich ist. Im Gegensatz zur traditionellen Programmierung, bei der eine festgelegte Abfolge von Anweisungen gegeben wird, basiert das maschinelle Lernen

[1] Issam El Naqa und Martin J Murphy [2015]. *What is machine learning?* Springer.

auf der Analyse und Verarbeitung großer Mengen an Daten. Durch diesen iterativen Lernprozess können Maschinen Muster, Zusammenhänge und Regeln erkennen, um Vorhersagen zu treffen oder komplexe Aufgaben zu automatisieren. Der Kern des maschinellen Lernens liegt in der Verwendung von Algorithmen und statistischen Modellen, um aus den vorhandenen Daten zu lernen. Diese Modelle werden trainiert, indem sie mit Eingabedaten gefüttert werden und ihre internen Parameter so angepasst werden, dass sie die gewünschten Ausgabewerte erzeugen. 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.[2]

8.1.2 Deep Learning

Deep Learning ist ein Teilbereich des maschinellen Lernens, der sich auf die Verwendung neuronaler Netzwerke mit vielen Schichten konzentriert. Neuronale Netzwerke sind mathematische Modelle, die biologische Neuronen simulieren und miteinander verbundene Schichten von Neuronen enthalten. Jede Schicht nimmt Eingaben von der vorherigen Schicht entgegen und gibt Ausgaben an die nächste Schicht weiter. Der Hauptvorteil von Deep Learning liegt in der Fähigkeit, automatisch Merkmale aus den Daten zu extrahieren, anstatt dass diese manuell von einem Experten definiert werden müssen. Durch die Kombination vieler Schichten können neuronale Netzwerke komplexe Muster und abstrakte Darstellungen erfassen, wodurch sie in der Lage sind, hochdimensionale Daten effektiv zu verarbeiten. Das Training von Deep-Learning-Modellen erfolgt in der Regel mit Hilfe von 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

[2]Issam El Naqa und Martin J Murphy [2015]. *What is machine learning?* Springer; Gopinath Rebala u. a. [2019]. »Machine learning definition and basics«. In: *An introduction to machine learning*, S. 1–17.

Modells zu verbessern, genaue Vorhersagen zu treffen oder komplexe Aufgaben zu lösen. In den folgenden Abschnitten werden wir uns mit den grundlegenden Komponenten und Funktionen von Convolutional Neural Networks (CNNs) befassen, die eine spezielle Art von neuronalen Netzwerken sind, die insbesondere in der Bildverarbeitung weit verbreitet sind.[3]

8.1.3 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. 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

[3]Deep Learning [2020]. »Deep learning«. In: *High-dimensional fuzzy clustering*; Yann LeCun, Yoshua Bengio und Geoffrey Hinton [2015]. »Deep learning«. In: *nature* 521.7553, S. 436–444; Jürgen Schmidhuber [2015]. »Deep learning in neural networks: An overview«. In: *Neural networks* 61, S. 85–117.

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 einzelnen Komponenten von CNNs befassen, ihre Funktionsweise im Detail untersuchen und verschiedene Architekturen und Techniken kennenlernen, die in der Praxis weit verbreitet sind.[4]

[Graph mit 3 Kreisen: KI > ML > DL]

8.1.4 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.

[4] 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; 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; Yann LeCun u. a. [1998]. »Gradient-based learning applied to document recognition«. In: *Proceedings of the IEEE* 86.11, S. 2278–2324; Karen Simonyan und Andrew Zisserman [2014]. »Very deep convolutional networks for large-scale image recognition«. In: *arXiv preprint arXiv:1409.1556*.

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 Fähigkeiten von CNNs haben die Leistungsfähigkeit von Bildverarbeitungssystemen erheblich verbessert und zu bahnbrechenden Fortschritten in Bereichen wie autonomes Fahren, medizinische Bildgebung und Überwachungstechnologie geführt.

Anwendungen von Convolutional Neural Networks in der Computer Vision

Die Computer Vision 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, 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 und mit ihr zu interagieren.

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. 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 Anwendungen

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.

- 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.

- 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.

- Naturwissenschaften: In den Naturwissenschaften werden CNNs zur Analyse von großen Datensätzen aus Bereichen wie Astronomie, Genetik und Teilchenphysik eingesetzt. Sie können dabei helfen, Muster, Zusammenhänge und neue Erkenntnisse in komplexen wissenschaftlichen Daten zu identifizieren.

Insgesamt bieten Convolutional Neural Networks 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.

8.1.5 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:

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. Diese Filter 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.

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. Durch das Anwenden von Pooling-Operationen wird die räumliche Auflösung der Merkmalskarten reduziert, während wichtige Informationen beibehalten werden. Dies führt zu einer effizienteren Verarbeitung 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, die es ihnen ermöglichen, räumliche Informationen zu berücksichtigen, lokale Muster zu erkennen und komplexe visuelle Aufgaben effektiv zu bewältigen.

8.1.6 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

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.[5]

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.

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 CNN 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.

[5] Wolfgang Ertel und Nathanael T Black [2016]. *Grundkurs Künstliche Intelligenz*. Bd. 4. Springer.

Toleranz gegenüber Translationen, Skalierungen und Verzerrungen

CNNs sind auch in der Lage, Translationen, Skalierungen und Verzerrungen in den Eingabedaten zu tolerieren, 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 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 semantischen 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 Videodaten.

8.2 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.

8.2.1 Bildklassifikation

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.[6]

8.2.2 Objekterkennung

CNNs werden häufig für die Erkennung und Lokalisierung von Objekten in Bildern eingesetzt. 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.

8.2.3 Gesichtserkennung

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.

[6] 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.

8.2.4 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. In der 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.

8.2.5 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. Sie ermöglichen die Automatisierung und Verbesserung 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.

8.3 Architektur von CNNs

8.3.1 Input Layer

Die Input Layer eines Convolutional Neural Networks (CNNs) spielt eine entscheidende Rolle bei der Aufnahme und Vorverarbeitung der Rohdaten, die in der Regel Bilder oder Videos sind. 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. 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. Durch die Struktur der Input Layer können CNNs komplexe visuelle Muster in den Daten erkennen und in der Lage sein, komplexe Aufgaben wie Objekterkennung, Gesichtserkennung und Semantische Segmentierung effektiv zu bewältigen.[7]

8.3.2 Hidden Layers

Die Hidden Layers bilden die Hauptkomponente eines Convolutional Neural Networks (CNNs) und bestehen standardmäßig aus Convolutional-Layern, Pooling-Layern, vollständig verbundenen Layern und Aktivierungsfunktionen. Diese Schichten spielen eine entscheidende Rolle bei der 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. 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 Lokalisierungsinvarianz verbessert, da die genaue Position eines bestimmten Merkmals in den Merkmalskarten weniger wichtig wird. Jedoch erhöhen Pooling-Layer nicht unbedingt die Translationssicherheit. Während des Pooling-Prozesses kann ein gewisser Informa-

[7] John Murphy [2016]. »An overview of convolutional neural network architectures for deep learning«. In: *Microway Inc*, S. 1–22.

tionsverlust 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.[8]

8.3.3 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. 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

[8] John Murphy [2016]. »An overview of convolutional neural network architectures for deep learning«. In: *Microway Inc*, S. 1–22; Adrian Rosebrock [2021]. *Convolutional Neural Networks (CNNs) and Layer Types*. URL: https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/.

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.

8.4 Convolutional layers

8.4.1 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. 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 CNN, 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.

8.4.2 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. In einem CNN 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 CNN 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. 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 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. 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.

8.4.3 Aktivierungsfunktionen

Aktivierungsfunktionen spielen eine wichtige Rolle in der Architektur von Convolutional Neural Networks (CNNs). Sie werden auf die Ausgaben der Neuronen angewendet und führen nichtlineare Transformationen durch. 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 (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, die eine S-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.

8.4.4 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. 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. 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. 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 Netzwerkarchitektur besser erfassen.[9]

[9] Jiuxiang Gu u. a. [2018]. »Recent advances in convolutional neural networks«. In: *Pattern recognition* 77, S. 354–377; Katarzyna Janocha und Wojciech Marian Czarnecki [2017]. »On loss functions for deep neural networks in classification«. In: *arXiv preprint arXiv:1702.05659*.

Kapitel 9

Datenverarbeitung (Data Preprocessing)

Vorab: wir müssten davon nichts machen, da unser Datensatz schon vorbereitet war. Aber stellt sich die frage für uns, ob der Datensatz passt (da könntest auch eine subsection daraus bauen am Ende. Idk, using a pre-aggregated Dataset).

9.1 Was sind Daten?

9.2 Bedeutung der richtigen Daten

9.2.1 Datenbereinigung (Data Cleaning)

In der Welt der Datenanalyse und des maschinellen Lernens ist die Qualität der Daten von entscheidender Bedeutung. Daten werden oft in verschiedenen Formen und aus verschiedenen Quellen gesammelt, und sie können fehlerhaft, inkonsistent oder unvollständig sein. Um verlässliche und aussagekräftige Ergebnisse zu erzielen, ist es daher wichtig, die Daten zu bereinigen und sicherzustellen, dass sie für Analysen und Modellierung geeignet sind. Dieser Prozess wird als Datenbereinigung oder Data Cleaning bezeichnet. Die Datenbereinigung bezieht sich auf den Prozess der Entfernung von fehlerhaften, inkonsistenten

oder unvollständigen Daten aus einem Datensatz. Ziel ist es, die Qualität der Daten zu verbessern und sicherzustellen, dass sie für die weiteren Schritte der Datenanalyse und des maschinellen Lernens verwendet werden können. Eine gründliche Datenbereinigung ist entscheidend, da fehlerhafte oder inkonsistente Daten zu falschen Analysen oder Modellen führen können. Die Schritte der Datenbereinigung umfassen in der Regel das Entfernen von Duplikaten, das Behandeln von fehlenden Werten und das Korrigieren 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 Formatierungen, 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 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. Durch das Entfernen von fehlerhaften, inkonsistenten oder unvollständigen Daten wird die Zuverlässigkeit der Ergebnisse verbessert und die Basis für eine fundierte Datenanalyse geschaffen.[1]

9.2.2 Datennormalisierung (Data Normalization)

Die Datennormalisierung ist ein wichtiger Schritt in der Datenverarbeitung, der 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,

[1] Arthur D Chapman [2005]. *Principles of data quality.* GBIF; Erhard Rahm, Hong Hai Do u. a. [2000]. »Data cleaning: Problems and current approaches«. In: *IEEE Data Eng. Bull.* 23.4, S. 3–13.

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. Eine gängige Methode ist die Min-Max-Normalisierung, bei der die Daten auf einen bestimmten Wertebereich transformiert werden. Dabei werden die Daten auf einen Wertebereich zwischen 0 und 1 skaliert, wobei der 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. Eine weitere Methode ist die Z-Score-Normalisierung, 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. Eine weitere gängige Methode ist die Skalierung auf den Einheitsvektor, 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. Es ist wichtig, die Daten vor der Normalisierung sorgfältig zu analysieren und zu verstehen, um die richtige Methode auszuwählen. Insgesamt spielt die Datennormalisierung eine wichtige Rolle in der Datenverarbeitung, um die Daten auf eine einheitliche Skala oder Verteilung zu bringen. 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.

9.2.3 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. Die Datenaugmentierung hilft dabei, Überanpassung (Overfitting) zu vermeiden und die allgemeelle 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 häufig verwendetes Verfahren ist das Zufällige Zuschneiden (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. 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. 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.[2]

9.2.4 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

[2]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 10

Wie haben ist die Architektur von unserem Model

KAPITEL 10 NOCH AUSSTEHEND Sure! Let's dive deeper into the architecture of the DeepLabV3+ model with a ResNet-50 backbone.

The DeepLabV3+ model consists of two main components: the backbone network and the prediction head.

1. Backbone Network: The backbone network is responsible for extracting meaningful features from the input image. In this case, the backbone network is based on the ResNet-50 architecture. ResNet-50 is a deep CNN that has achieved excellent performance in various computer vision tasks. It consists of several convolutional layers grouped into different stages, with residual connections between them to address the vanishing gradient problem. These residual connections enable the network to learn more effectively by propagating gradients through shortcuts.

2. Prediction Head: The prediction head takes the features extracted by the backbone network and generates the final predictions. In DeepLabV3+, the prediction head employs atrous (or dilated) convolutions, which have different dilation rates to capture multi-scale information. This allows the model to have both detailed local information and contextual information.

The original DeepLabV3+ model is pretrained on the ImageNet dataset, which contains

millions of labeled images from thousands of categories. This pretraining helps the model learn generic visual features that can be fine-tuned for specific tasks.

In the provided code, the pretrained DeepLabV3+ model is loaded with the ResNet-50 backbone. This means that the backbone consists of the initial layers of ResNet-50 responsible for feature extraction. The exact layers and their configurations can be quite complex, but they typically include multiple convolutional layers, pooling layers for downsampling, and residual connections.

The modification in the code is done to the final layer of the model, which is the classifier. In the original DeepLabV3+ model, the classifier is a 1x1 convolutional layer that produces a tensor with dimensions $[\text{batch}_size, num_classes, height, width]. The number of classes in the original mode$

In the provided code, the final layer is replaced with another 1x1 convolutional layer, but with a modified number of output channels. The original number of output channels is 256, which corresponds to the number of features learned by the backbone network. However, in the modified code, the number of output channels is set to 3, indicating that the model will output a tensor with dimensions $[\text{batch}_size, 3, height, width]. This change in the number of output channels$

Overall, the DeepLabV3+ model with a ResNet-50 backbone is a powerful architecture for semantic segmentation tasks, and by modifying the final layer, the code customizes the model to classify pixels into one of three classes.

- Das ist unser code:

$\text{Net} = \text{torchvision.models.segmentation.deeplabv3}_resnet50(pretrained = True) Load net Net.classifi$
$torch.nn.Conv2d(256, 3, kernel_size = (1, 1), stride = (1, 1)) Change final layer to 3 classes Net =$
$Net.to(device)$

$\text{optimizer} = \text{torch.optim.Adam(params=Net.parameters(), lr=Learning}_Rate) Create adam optimizer$

1. Loading the Model: The code begins by loading the DeepLabV3+ model with a ResNet-50 backbone using the torchvision library. DeepLabV3+ is a popular model for semantic segmentation, which means it assigns a class label to each pixel in an input image. The ResNet-50 backbone is a type of convolutional neural network (CNN) that is known for its effectiveness in extracting features from images.

2. Modifying the Final Layer: Next, the code modifies the final layer of the loaded

model. In the original model, the final layer is a classifier that produces a probability distribution over 21 different classes (such as "person,car,dog,ëtc.). However, in this code, the final layer is replaced with a 1x1 convolutional layer. This modification changes the output of the model to produce a probability distribution over 3 classes instead of 21. The specific values for the kernel size and stride determine the behavior of the convolution operation.

3. Moving the Model to a Device: The modified model is then moved to a specified device, which could be either the CPU or a GPU. This step ensures that the computations performed by the model will take place on the selected device. Utilizing a GPU can significantly accelerate the training and inference processes for deep learning models.

4. Creating the Optimizer: An optimizer is an algorithm that adjusts the parameters of the model during the training process to minimize the loss function. In this code, an Adam optimizer is created using the torch.optim module from PyTorch. The Adam optimizer is widely used and has been shown to be effective in training deep learning models. It takes as input the parameters of the model (obtained using ‘Net.parameters()‘) and the learning rate (specified by ‘Learning$_{R}ate$‘).$Thelearningratedeterminesthestepsizeatwhichtheoptimizerupdatesthem$

By following this architecture, you have a modified DeepLabV3+ model with a ResNet-50 backbone, capable of segmenting images into one of three classes. The model's parameters will be optimized during training using the Adam optimizer and the specified learning rate.

Kapitel 11

Convolutional Neural Network Trainings Prozess

11.1 Was ist Training?

- Einführung in das Konzept des Trainings von neuronalen Netzwerken - Bedeutung des Trainingsprozesses für die Leistungsfähigkeit von künstlichen neuronalen Netzen - Warum ist Training notwendig, um ein Convolutional Neural Network (CNN) effektiv einzusetzen?

11.2 Trainingprozess

- Überblick über den allgemeinen Ablauf des Trainingsprozesses - Schritte und Phasen des Trainingsprozesses bei der Verwendung von CNNs - Datenverarbeitung und Vorverarbeitung während des Trainingsprozesses - Spezifische Aspekte des Trainingsprozesses bei der Verwendung von Convolutional Neural Networks - Bedeutung der Convolutional-Schichten und Pooling-Schichten im Trainingprozess - Anwendung von Aktivierungsfunktionen während des Trainingsprozesses für CNNs

So trainieren wir:

for itr in range(10000): Training loop print("Training [

images, ann = LoadBatch() Load taining batch

Load image images = torch.autograd.Variable(images, requires$_grad = False).to(device)$

Load annotation ann = torch.autograd.Variable(ann, requires$_grad = False).to(device)$

make prediction Pred = Net(images)['out']

Net.zero$_grad()$

Set loss function criterion = torch.nn.CrossEntropyLoss()

Calculate cross entropy loss Loss = criterion(Pred, ann.long())

Backpropogate loss Loss.backward()

Apply gradient descent change to weight optimizer.step()

Get prediction classes seg = torch.argmax(Pred[0], 0).cpu().detach().numpy()

Save model every 1000 iterations to .torch file if itr

11.3 Stochastic Gradient Descent (SGD)

- Erklärung des Stochastic Gradient Descent-Algorithmus und seiner Rolle im Training von CNNs - Unterschiede zwischen Gradient Descent und Stochastic Gradient Descent - Optimierungsverfahren und Anpassung der Lernrate im SGD-Algorithmus

11.4 Backpropagation

- Bedeutung von Backpropagation für das Training von neuronalen Netzwerken - Funktionsweise von Backpropagation im Kontext von CNNs - Verwendung von Backpropagation zur Berechnung der Gradienten während des Trainingsprozesses

11.5 Hyperparameter-Tuning

- Erklärung des Begriffs "Hyperparameteründ ihre Bedeutung für das Training von CNNs - Methoden und Strategien für das Tuning von Hyperparametern - Auswirkungen von Hyperparameter-Tuning auf die Leistungsfähigkeit von CNNs

11.6 Regularisierungstechniken

- Einführung in Regularisierungstechniken und ihre Bedeutung im Trainingsprozess - Unterschiedliche Ansätze zur Regularisierung von CNNs, z.B. L1- und L2-Regularisierung - Verwendung von Dropout und Data Augmentation als Regularisierungstechniken

https://learnopencv.com/deeplabv3-ultimate-guide/

Kapitel 12

Transfer Learning

12.1 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 nutzen, kann man ihre Architekturen und Merkmalsextraktoren verwenden und sie für eine spezifische Aufgabe abstimmen. Dadurch kann der Lernprozess beschleunigt werden.[1]

12.2 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

[1] 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; 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.

12.3 Fine-tuning von vortrainierten Modellen

Die Feinabstimmung (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, sich an die spezifischen Merkmale und Nuancen der neuen Aufgabe anzupassen. 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. 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.

12.4 Verwendung von vortrainierten Modellen als Merkmalsextraktoren

Eine alternative Methode des Transfer Learning besteht darin, das vortrainierte Modell als festen Merkmalsextraktor zu verwenden. 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. 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 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 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.

12.5 Anwendung von DeepLabV3 ResNet50

In diesem Abschnitt wird die konkrete Umsetzung des Transfer Learning in unserem Projekt durch die Verwendung des DeepLabV3 ResNet50-Modells 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 faltungsbasierter neuronaler Netzwerke zu erleichtern. Die Entscheidung, das DeepLabV3 ResNet50-Modell in unserem Projekt zu verwenden, basierte auf einer Vielzahl von Überlegungen. 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 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. 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.

Kapitel 13

Common Challenges and Solutions

13.1 Overfitting und Underfitting

In maschinellem 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 vorgestellt, um diese Probleme zu identifizieren und zu mindern.

13.1.1 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 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. 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. Eine weitere Technik zur Bekämpfung von Overfitting ist die Dropout-Methode. 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. Darüber hinaus kann das frühzeitige Beenden des Trainingsprozesses, auch bekannt als Early Stopping, 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 Muster der Trainingsdaten spezialisiert.

13.1.2 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. 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 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.

13.2 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.

13.2.1 Ursachen der vanishing Gradients

Die vanishing Gradients entstehen, wenn die Gradienten während des Backpropagation-Verfahrens in tiefen neuronalen Netzen 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.

13.2.2 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.

13.2.3 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.

13.2.4 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 und die He-Initialisierung 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 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.

13.2.5 Lösungsansätze für exploding Gradients

Um exploding Gradients zu bewältigen, ist eine gängige Methode das Gradient Clipping. 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. 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.

13.3 Gradient Descent Optimization

Das Kapitel "Gradient Descent Optimization"behandelt die Herausforderungen und Lösungen im Bereich der Optimierungsalgorithmen 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.

13.3.1 Standard-Gradientenabstieg

Der Standard-Gradientenabstieg 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.

13.3.2 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. 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.

13.3.3 Mini-Batch Gradientenabstieg

Um die Vorteile des SGD zu nutzen und gleichzeitig die Stabilität zu verbessern, wird oft der Mini-Batch Gradientenabstieg eingesetzt. 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.

13.3.4 Batch Normalisierung

Ein weiterer Ansatz zur Verbesserung des Gradientenabstiegs besteht in der Batch-Normalisierung. 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.

Kapitel 14

Tools and Frameworks for CNN Training

14.1 PyTorch

Introduce PyTorch as a popular deep learning framework known for its flexibility and dynamic computational graph. Discuss its key features, such as automatic differentiation, GPU acceleration, and extensive support for neural network architectures. Provide examples of PyTorch code snippets to demonstrate its ease of use and showcase its capabilities for CNN training.

14.2 TensorFlow

Discuss TensorFlow, one of the most widely used deep learning frameworks. Explain its static computational graph paradigm and its ability to efficiently utilize GPUs for accelerated training. Discuss TensorFlow's ecosystem, which includes high-level APIs like Keras and TensorFlow 2.0's eager execution mode. Illustrate the usage of TensorFlow through code snippets and highlight its strengths and popularity in the deep learning community.

14.3 Keras

Introduce Keras as a high-level deep learning library that runs on top of TensorFlow, CNTK, or Theano. Emphasize Keras' user-friendly interface, which simplifies the process of building, training, and evaluating neural networks. Discuss its versatility in supporting various neural network architectures, including CNNs, and its integration with popular backends. Provide examples of Keras code snippets to showcase its simplicity and accessibility.

14.4 Caffe

Discuss Caffe, a deep learning framework widely used for its efficiency and speed in CNN training. Explain Caffe's model definition language, which allows easy specification of network architectures. Discuss Caffe's pre-trained models and model zoo, which provide a vast collection of well-performing models for various tasks. Describe its popularity in computer vision applications and provide examples of Caffe code snippets.

14.5 Other Popular Frameworks

Discuss other notable deep learning frameworks that are commonly used for CNN training. Include frameworks such as MXNet, Theano, and Torch. Highlight their unique features, advantages, and any notable use cases. Briefly explain how they compare to the previously discussed frameworks.

Kapitel 15

Conclusion and Future Work

15.1 Summary of the paper

Yes, yes. Very good paper. Will show my kids and wife. They be proud. Yes, yes.

15.2 Key takeaways

15.3 Future research directions

Abkürzungsverzeichnis

PPIs Pixel per Inch 8

CNNs Convolutional neural network 8

SRs Super Resolution 41

GANs Generative Adversarial Network 8

PNGs 12

JPGs Joint Photographic Experts Group 13

SVGs 13

13

CSS 14

GIFs 14

EPSs 14

BMPs 14

PSDs 14

TIFFs 14

FLOPS 55

54

MOSs 54

RMSEs 56

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

Formelverzeichnis

Literatur

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. 46].

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. 33].

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 [siehe S. 60].

Boutell, Thomas [1997]. *Png (portable network graphics) specification version 1.0*. Techn. Ber. [siehe S. 12].

Boutell, Thomas u. a. [o. D.] *PNG (Portable Network Graphics Format) Version 1.0*. Techn. Ber. RFC 2083 [siehe S. 12].

Burger, Wilhelm und Mark James Burge [2009]. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag [siehe S. 12].

Burger, Wilhelm u. a. [2015]. »Digitale Bilder«. In: *Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java*, S. 1–24 [siehe S. 12].

Burt, Peter J und Edward H Adelson [1987]. »The Laplacian pyramid as a compact image code«. In: *Readings in computer vision*. Elsevier, S. 671–679 [siehe S. 47].

Chapman, Arthur D [2005]. *Principles of data quality.* GBIF [siehe S. 85].

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. 53].

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. 55].

Contributors, Wikipedia [2023]. *Bildskalierung Definition.* https://en.wikipedia.org/wiki/Image_scaling [siehe S. 8].

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. 32].

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. 57].

El Naqa, Issam und Martin J Murphy [2015]. *What is machine learning?* Springer [siehe S. 65, 66].

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. 15].

Encyclopedia Britannica [n.d.] *JPEG.* https://www.britannica.com/technology/JPEG. Accessed: March 9, 2023 [siehe S. 15].

Ertel, Wolfgang und Nathanael T Black [2016]. *Grundkurs Künstliche Intelligenz.* Bd. 4. Springer [siehe S. 73].

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. 13].

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. 25].

Gu, Jiuxiang u. a. [2018]. »Recent advances in convolutional neural networks«. In: *Pattern recognition* 77, S. 354–377 [siehe S. 83].

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. 59].

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. 75].

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. 49, 68].

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. 53].

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. 59].

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. 47].

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. 55].

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. 15].

Jaderberg, Max, Karen Simonyan, Andrew Zisserman u. a. [2015]. »Spatial transformer networks«. In: *Advances in neural information processing systems* 28 [siehe S. 40].

Janocha, Katarzyna und Wojciech Marian Czarnecki [2017]. »On loss functions for deep neural networks in classification«. In: *arXiv preprint arXiv:1702.05659* [siehe S. 83].

Jiang, Nan und Luo Wang [2015]. »Quantum image scaling using nearest neighbor interpolation«. In: *Quantum Information Processing* 14, S. 1559–1571 [siehe S. 22].

Kolb, Tobias [o. D.] »Entwicklung eines Convolutional Neural Network zur Handschrifterkennung«. In: *Angewandtes maschinelles Lernen–SS2019* [], S. 28 [siehe S. 40].

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. 52].

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. 68].

Learning, Deep [2020]. »Deep learning«. In: *High-dimensional fuzzy clustering* [siehe S. 67].

LeCun, Yann, Yoshua Bengio und Geoffrey Hinton [2015]. »Deep learning«. In: *nature* 521.7553, S. 436–444 [siehe S. 67].

LeCun, Yann u. a. [1998]. »Gradient-based learning applied to document recognition«. In: *Proceedings of the IEEE* 86.11, S. 2278–2324 [siehe S. 68].

Li, Hao u. a. [2018]. »Visualizing the loss landscape of neural nets«. In: *Advances in neural information processing systems* 31 [siehe S. 41].

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. 60].

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. 40].

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. 47].

Marcellin, Michael W u. a. [2000]. »An overview of JPEG-2000«. In: *Proceedings DCC 2000. Data Compression Conference*. IEEE, S. 523–541 [siehe S. 15].

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. 52].

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. 56].

Mozilla Contributors [n.d.] *SVG (Scalable Vector Graphics)*. `https://developer.mozilla.org/en-US/docs/Web/SVG`. Accessed: March 9, 2023 [siehe S. 14].

Murphy, John [2016]. »An overview of convolutional neural network architectures for deep learning«. In: *Microway Inc*, S. 1–22 [siehe S. 77, 78].

Nah, Seungjun, Tae Hyun Kim und Kyoung Mu Lee [2017]. »Deep multi-scale convolutional neural network for dynamic scene deblurring«. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 3883–3891 [siehe S. 47].

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. 7].

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. 17].

O'Shea, Keiron und Ryan Nash [2015]. *An Introduction to Convolutional Neural Networks*. arXiv: 1511.08458 [cs.NE] [siehe S. 40].

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. 96].

Prepressure [n.d.] *Prepressure Library: File Formats*. https://www.prepressure.com/library/file-formats/. Accessed: March 9, 2023 [siehe S. 15].

Quint, Antoine [2003]. »Scalable vector graphics«. In: *IEEE MultiMedia* 10.3, S. 99–102 [siehe S. 13].

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. 15].

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. 85].

Rebala, Gopinath u. a. [2019]. »Machine learning definition and basics«. In: *An introduction to machine learning*, S. 1–17 [siehe S. 66].

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. 78].

Schmidhuber, Jürgen [2015]. »Deep learning in neural networks: An overview«. In: *Neural networks* 61, S. 85–117 [siehe S. 67].

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. 52, 54].

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. 88].

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. 48].

Simonyan, Karen und Andrew Zisserman [2014]. »Very deep convolutional networks for large-scale image recognition«. In: *arXiv preprint arXiv:1409.1556* [siehe S. 68].

Tech-Lib [2020]. *Image Scaling Definition*. http://www.dante.de [siehe S. 8].

Wang, David C.C, Anthony H Vagnucci und C.C Li [1983]. »Digital image enhancement: A survey«. 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. url: https://www.sciencedirect.com/science/article/pii/0734189X83900610 [siehe S. 19].

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. 52, 57].

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. 53, 54].

whuber [Sep. 2011]. *What is Lanczos resampling?* https://gis.stackexchange.com/a/14361 [siehe S. 9].

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. 40].

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. 55].

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. 57].

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. 96].

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. 60].

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. 60].

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. 59].

Liste der ToDo's