Bachelorarbeit

Deep Learning zur visuellen Erkennung von Tierarten

Minh Kien Nguyen

Philipps-Universität Marburg

18. August 2021

Bachelorarbeit

Deep Learning zur visuellen Erkennung von Tierarten

von
Minh Kien Nguyen

Betreuung
Prof. Dr. -Ing. Bernd Freisleben
Dr. Markus Mühling
M.Sc. Daniel Schneider
AG Verteilte Systeme

Philipps-Universität Marburg

18. August 2021

Zusammenfassung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Inhaltsverzeichnis

1 Einleitung

Im Zentrum des ersten Kapitels steht der Einsatz von *Deep Learning* im Bereich der Umweltbeobachtung. Es werden auch Fragestellungen aufgeführt, die einen Überblick über den Aufbau der Arbeit geben.

1.1 Deep Learning in der Umweltbeobachtung

Der *LOEWE-Schwerpunkt Natur 4.0 Sensing Biodiversity* (abgekürzt: Natur 4.0) ist ein Gemeinschaftsprojekt der federführenden Philipps-Universität Marburg, der Justus-Liebig-Universität Gießen, der Technische Universität Darmstadt und der Senckenberg Gesellschaft für Naturforschung in Frankfurt. Ziel des Projekts ist die Entwicklung eines Umweltmonitoringsystems zur hoch aufgelösten Beobachtung von naturschutzrelevanten Arten, Lebensräumen und Prozessen durch vernetzte Sensorik und integrative Datenanalyse[1]. Das Testgebiet für die Prototypentwicklung von Natur 4.0 ist der Marburger Universitätswald in Caldern.

Einer der Schwerpunkte des Projekts liegt auf **der visuellen Erkennung von Tierarten**. Die Ergebnisse dieser Aufgabe ermöglichen zusammen mit weiteren zeitlichen und räumlichen Sensordaten die Beantwortung folgender Fragen: Welche Tierarten sind im Testgebiet vorhanden? Wann, wie häufig, wo genau und unter welchen Bedingungen kommen sie vor? Antworten auf diese Fragestellungen erweisen sich als wertvoll: Sie geben nicht nur Auskunft über die biologische Vielfalt und Eigenschaften von Ökosystemen im Testgebiet, sondern auch über die Wechselwirkungen zwischen Tieren, Pflanzen und klimatischen Bedingungen darin. Aus solchen Informationen lassen sich verschiedene Naturschutzstrategien ableiten.

Bei einem gegebenen Bild gilt die visuelle Erkennung von Tierarten als erfolgreich, wenn alle Tiere in diesem Bild gefunden und richtig kategorisiert werden. Die Aufgabe ist in der Tat ein typisches Beispiel für *das Objekterkennungsproblem*.

1.1.1 Objekterkennung

Die Objekterkennung im Forschungsfeld *Computer-Vision* erfolgt in zwei Schritten:

1. *Objektlokalisierung*: Der erste Schritt bezieht sich auf das Identifizieren der Position eines oder mehrerer Objekte in einem Bild und das Zeichnen eines Begrenzungsrahmens um ihre Ausdehnung.
2. *Bildklassifizierung*: Bei dem zweiten Schritt handelt es sich um die Aufgabe, einem Bildobjekt ein Label aus einem festen Satz von Kategorien zuzuweisen.

[1] *Natur 4.0 - kurz und bündig - Über uns - Natur 4.0 | Sensing Biodiversity - Philipps-Universität Marburg*. URL: https://www.uni-marburg.de/de/fb19/natur40/ueber-uns/natur4 (besucht am 5. Aug. 2021).

Auf die gleiche Weise lässt sich die visuelle Erkennung von Tierarten erledigen. Die Tiere im Eingabebild müssen zunächst lokalisiert und anschließend klassifiziert werden, um als richtig erkannt zu gelten.

Man verwendet *den datengetriebenen Ansatz*, um das Objekterkennungsproblem zu lösen. Die Idee dabei ist, dem Computer viele Fotos jeder einzelnen Objektklasse bereitzustellen und dann mit dem Computer Lernalgorithmen zu entwickeln (*trainieren*), die sich diese Fotos ansehen und visuelle Merkmale lernen, die Objekte von ihrer Umgebung und jede Klasse voneinander trennen. Ergebnis der Lernphase ist ein *trainiertes Modell*, das aus Fotodaten im Rahmen des Projekts Natur 4.0 bestimmte Tierarten erkennen können soll.

Die zu entwickelnden Algorithmen im datengetriebenen Ansatz fallen unter die Kategorie *Deep-Learning-Algorithmen*, die die Lernfähigkeit von Computern ermöglichen.

1.1.2 Deep Neural Network

In der Regel besteht ein Deep-Learning-Algorithmus (im Weiteren als *Deep Neural Network* (DNN) bezeichnet) aus einer *Eingabeschicht*, einer *Ausgabeschicht* und mindestens zwei *verborgenen Schichten*. Jede Schicht setzt sich aus *Neuronen* zusammen, die Informationen von anderen Neuronen oder von außen aufnehmen, modifizieren und als Ergebnis ausgeben. Jedes Neuron ist mit allen anderen Neuronen in der vorherigen sowie nächsten Schicht verbunden (siehe Abbildung 1.1). Jeder Verbindung wird ein Gewicht zugewiesen.

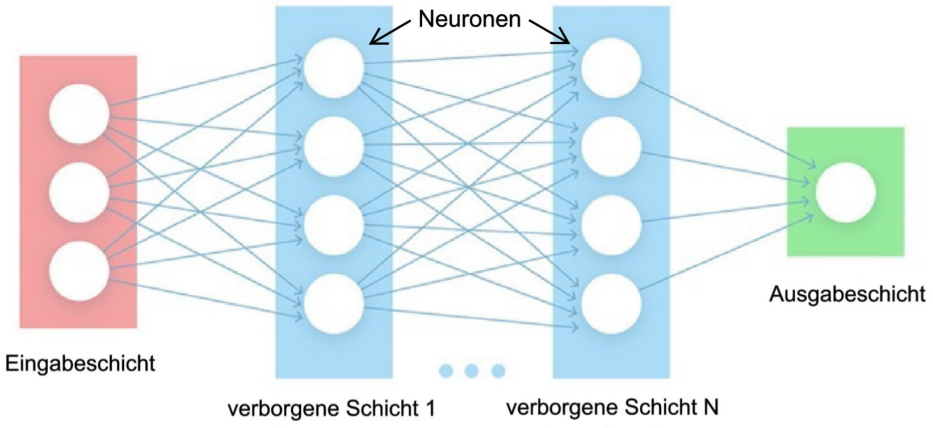

Abbildung 1.1: Aufbau eines Deep Neural Network [33]

DNNs lassen sich mittels des *Backpropagation-Algorithmus* [25, 12] trainieren. Der Algorithmus speist zunächst jede Trainingsinstanz in das zu trainierende DNN ein und berechnet die Ausgabe jedes Neurons in jeder konsekutiven Schicht (*Vorwärtsdurchlauf*). Danach ermittelt er den Ausgabefehler des DNN (i.e., die Differenz zwischen der gewünschten und der tatsächlichen Ausgabe), durchläuft jede Schicht in umgekehrter Richtung, um den Fehlerbeitrag jeder Neuronenverbindung auszurechnen (*Rückwärtsdurchlauf*), und schließlich optimiert die entsprechenden Verbindungsgewichte, um den Ausgabefehler des DNN zu minimieren, was das Ziel des gesamten Trainingsprozesses ist.

Für die Objekterkennung verwendet man einen bestimmten Typ von DNN namens *Convolutional Neural Network* (CNN) (mehr dazu im Kapitel 2). Unterschiedliche CNNs haben unterschiedliche Anordnungen von Schichten unterschiedlicher Typen (Architekturen), jedoch die gleiche Funktionsweise: CNNs konzentrieren sich auf *Low-Level-Merkmale* in den ersten Schichten und setzen diese anschließend in den nächsten zu *Higher-Level-Merkmale* zusammen. Beispielweise kann der Computer in niedrigeren Schichten lernen, Kanten oder Linien oder auch Kurven aus Pixel der Eingabebilder zu identifizieren, während er in höheren Schichten komplexere Formen sowie visuelle Merkmale lernt, die für die Objekterkennung ausschlaggebend sind.

Als Indikator für den Fortschritt in der Entwicklung von CNNs im Bereich Computer Vision können die Metriken Top-1- bzw. Top-5-Genauigkeit vom Wettbewerb *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [26] herangezogen werden. In ILSVRC werden CNNs mit der Bildklassifizierung des *ImageNet-Datensatzes* beauftragt, der tausend nicht überlappende Klassen und insgesamt ca. 1,35 Millionen Bilder umfasst[2]. Die Top-K-Genauigkeit stellt den Anteil der Testbilder dar, für die die Top-K-Prognosen des zu testenden CNN die richtige Antwort enthalten. Laut *Papers with Code* [18] sind die folgenden CNN-Architekturen mit ihrer jeweiligen ILSVRC Top-1-Genauigkeit zum Zeitpunkt dieser Arbeit Stand der Technik:

1. *Transformer* [6] - 87,76%
2. *EfficientNet* [31] - 84,40%
3. *ResNet* [14] - 78,57%.

Zur visuellen Erkennung von Tierarten im Projekt Natur 4.0 kann eines der aufgeführten CNN-Modelle (trainierten CNN-Architekturen) eingesetzt werden. Der Grund dafür basiert auf dem Konzept von *Transfer Learning*, das im Grunde eine Methode ist, die ein für eine Aufgabe A entwickeltes Modell als Ausgangspunkt für eine andere ähnliche B wiederverwendet. Ein solches Modell könnte bereits Merkmale gelernt haben, die für Aufgabe B relevant oder nützlich sind. Da die aufgeführten CNN-Modelle zuvor auf dem ImageNet-Datensatz trainiert wurden, der mehrere Tierarten unter seinen tausend Klassen enthält[3], ist es durchaus möglich, dass diesen Modellen Grundunterschiede zwischen den zu trennenden Tierarten schon bekannt sind. Deshalb kann man Teile des aufwendigen Trainings überspringen und infolgedessen Zeit sowie Speicher- und Rechenressourcen sparen, denn komplexe CNNs können in der Praxis tage- oder wochenlang von Grund auf trainiert werden [27, Seite 4][5, Seite 1][13, 30].

1.2 Die zentralen Fragestellungen

Die visuelle Erkennung von Tierarten im Projekt Natur 4.0 ist ein Objekterkennungsproblem, das sich mit dem datengetriebenen Ansatz lösen lässt. Damit dieser Lösungsansatz nach dem Transfer-Learning-Konzept umgesetzt werden kann, sind folgende Fragestellungen zu beantworten:

[2] *Download ImageNet Data - ImageNet*. URL: https://image-net.org/download.php (besucht am 11. Aug. 2021).

[3] *1000 Classification Categories - ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014)*. URL: https://image-net.org/challenges/LSVRC/2014/browse-synsets.php (besucht am 9. Aug. 2021).

1. **Welches vortrainierte CNN-Modell wird zur visuellen Erkennung von Tierarten im Projekt Natur 4.0 eingesetzt?**
2. **Wie wird das ausgewählte CNN-Modell nachtrainiert?**
3. **Welche Leistung kann das nachtrainierte CNN-Modell erzielen?**

Antworten auf die aufgeführten Fragestellungen lassen sich der Reihe nach in den Kapitel 3, Kapitel 4 sowie Kapitel 5 finden. Um jedoch die Argumentationen nachzuvollziehen, die zu diesen Antworten geführt haben, wird empfohlen, zunächst die grundlegenden Konzepte von CNN zu verstehen, die im Kapitel 2 ausführlicher beschrieben sind.

2 Convolutional Neural Network

In diesem Kapitel wird dargestellt, welche Durchbrüche speziell den Anlass zu Convolutional Neural Network (CNN) gegeben haben, wie sich DNN- und CNN-Struktur voneinander unterscheiden und wie CNN-Schichten aus funktionaler Sicht aussehen.

2.1 Entwicklungsmeilensteine

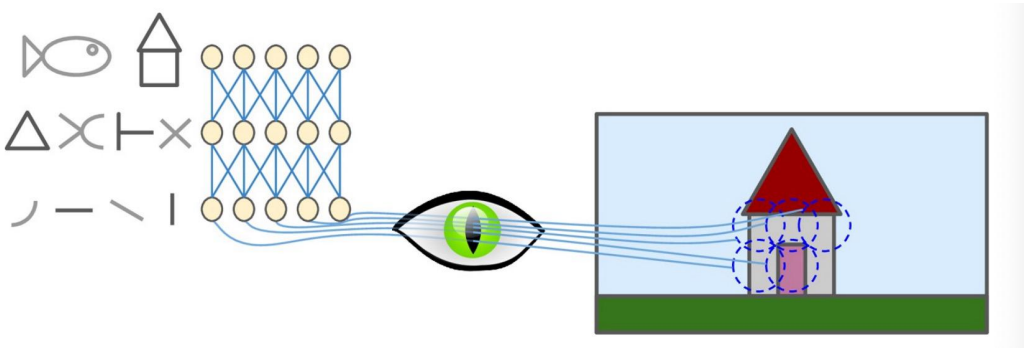

Abbildung 2.1: Lokales rezeptives Feld im visuellen Cortex [32]

1959 - 1962 David H. Hubel und Torsten N. Wiesel untersuchten in [15, 17] und danach in [16] durch eine Reihe von Experimenten an Katzen die Struktur vom *visuellen Cortex*, der gemäß [4] "die Region des Gehirns ist, die für die Verarbeitung und Integration der visuellen Information verantwortlich ist" . Die Autoren entdeckten, dass viele Neuronen im visuellen Cortex ein kleines *lokales rezeptives Feld* haben, d.h. sie reagieren nur auf visuelle Stimuli, die sich in einem begrenzten Bereich des Gesichtsfeldes befinden (siehe Abbildung 2.1, in der die lokalen rezeptiven Felder von fünf Neuronen durch gestrichelte Kreise dargestellt werden). Die rezeptiven Felder verschiedener Neuronen können sich überlappen und zusammen bilden sie das gesamte Gesichtsfeld. Außerdem fanden Hubel und Wiesel heraus, dass bestimmte Neuronen nur auf bestimmte Arten von visuellen Stimuli reagieren, und dass einige Neuronen größere rezeptive Felder haben sowie auf komplexere Muster reagieren, die Kombinationen einfacherer Muster sind.

Diese Beobachtungen führten zu der Idee, dass Neuronen eine hierarchische Organisation haben. Insbesondere basieren Neuronen höherer Schichten auf den Ausgaben benachbarter Neuronen in niedrigeren Schichten (Beachte in Abbildung 2.1, dass jedes Neuron nur mit einigen wenigen Neuronen in der vorherigen Schicht ver-

bunden ist). Diese Idee (im Weiteren als *Hierarchical-Neural-Network-Idee* bezeichnet) schuf das erste wesentliche Fundament für die Entwicklung von CNN.

1980 & 1988 Hubel und Wiesels Studien vom visuellen Cortex gaben Inspiration für das *Neocognitron* von Kunihiko Fukushima, welches das erste Beispiel für eine Netzwerkarchitektur war, die die Hierarchical-Neural-Network-Idee in ihr Design aufnahm. Im Prinzip ist das Neocognitron ein Modell, das 1980 in [9] für einen Mechanismus der verformungsbeständigen visuellen Mustererkennung vorgeschlagen und später 1988 in [10] gezeigt wurde, dass es nach Training dazu in der Lage war.

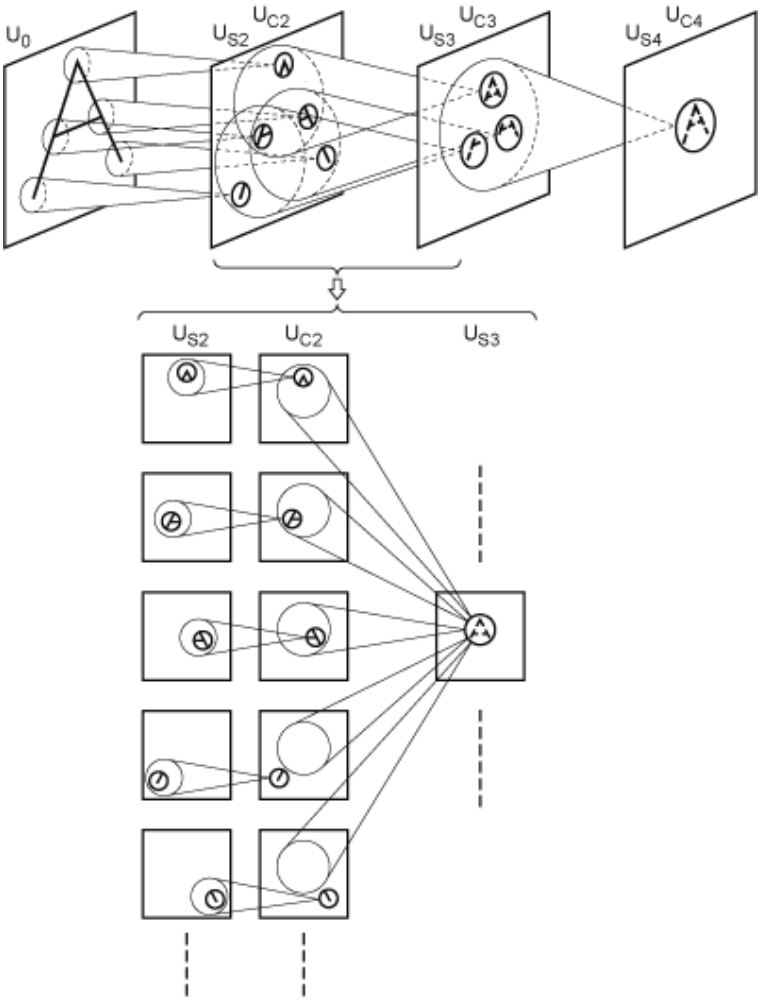

Abbildung 2.2: Der Prozess der Mustererkennung im Neocognitron. Die untere Hälfte der Abbildung ist eine vergrößerte Darstellung eines Teils vom Neocognitron. [23]

Fukushima führte in seinem Neocognitron-Modell zwei besondere Komponenten ein: *S-Zellen* und *C-Zellen*. Der Einfachheit halber kann man Zellen als Neuronen betrachten. Während S-Zellen zur Merkmalsextraktion dienen, werden C-Zellen in das Neocognitron-Modell eingefügt, um Positionsfehler in den Merkmalen des Stimulus zu berücksichtigen. Die Zellen sind in abwechselnden Schichten von S-Zellen und

C-Zellen angeordnet, so dass der Prozess der Merkmalsextraktion durch S-Zellen sowie der Tolerierung der Positionsverschiebung durch C-Zellen in der gesamten Netzwerkarchitektur wiederholt wird. Dadurch lassen sich lokale Merkmale, die in niedrigeren Schichten extrahiert wurden, schrittweise in globalere Merkmale integrieren (siehe Abbildung 2.2). Die Schichten von S- sowie C-Zellen im Neocognitron sind jeweils die Vorläufer von den zwei grundlegenden Arten von Schichten in modernen CNN-Architekturen: *Convolutional Layer* und *Pooling Layer* (mehr dazu im Abschnitt 2.3).

1989 & 1998 Obwohl das Neocognitron-Modell 1980 erschien, wurde der Backpropagation-Algorithmus erst fast ein Jahrzehnt später in [20] zum Trainieren von CNN angewendet. Der Autor dieser Veröffentlichung, Yann LeCun, wandte erfolgreich eine Netzwerkarchitektur an, die eine Variante vom Neocognitron-Modell war und durch den Backpropagation-Algorithmus trainiert wurde, um handgeschriebene Postleitzahlen zu identifizieren. Diese Architektur ist die ursprüngliche Form von *LeNet-5*, die als die bekannteste CNN-Architektur angesehen werden kann. LeNet-5 wurde erstmals 1998 in [21] eingeführt. Dabei haben die Autoren verschiedene Methoden zur Erkennung von handschriftlichen Zeichen (einschließlich LeNet-5) überprüft und auf Basis ihrer Leistungen bei einer Standardaufgabe zur Erkennung handschriftlicher Ziffern verglichen. Die Ergebnisse zeigten, dass CNN im Allgemeinen und LeNet-5 im Besonderen alle anderen Modelle übertrafen. Folglich war LeNet-5 in den folgenden Jahren der Ausgangspunkt für viele weitere CNN-Architekturen. Dieses Modell legte den Grundstein für modernes Computer-Vision.

2012 Jedoch war LeNet-5 aufgrund des Mangels an Rechenressourcen im Jahr 1998 noch nicht in der Lage, auf anspruchsvollere Daten als Ziffern zu skalieren. Um eine hohe Leistung bei komplexen Daten zu erzielen, war es notwendig, dass CNN-Modelle tiefer gehen und mehr Schichten haben, was rechenintensiv war, aber durch den Einsatz von Grafikprozessoren (GPUs) möglich wurde. Diese Schlussfolgerung wurde 2012 von Alex Krizhevsky in [19] gezogen und durch sein CNN-Modell *AlexNet* demonstriert. AlexNet ist größtenteils LeNet-5 ähnlich, nur viel größer und tiefer (in Bezug auf Größe und Anzahl der Schichten) und wurde mit zwei GPUs trainiert. Daneben war es die erste CNN-Architektur, die Convolutional Layer direkt übereinander stapelte, anstatt wie im Neocognitron-Modell von Fukushima ein Pooling Layer über jedem Convolutional Layer zu stapeln. Mit großem Vorsprung hat diese Architektur den Wettbewerb ILSVRC 2012 gewonnen: Das Modell erreichte 83% Top-5-Genauigkeit, während das zweitbeste nur 74% erreichen konnte[4]. Dies war das erste Mal, dass ein CNN-Modell beim großen ImageNet-Datensatz so gut funktionierte. AlexNet gilt somit als eine der einflussreichsten Veröffentlichungen im Bereich Computer-Vision und hat viele weitere Artikel angeregt, die GPUs verwandten, um das Training von CNNs zu beschleunigen.

[4] *Analysis of ILSVRC2012 Results - ImageNet Challenge 2012 Analysis*. URL: https://image-net.org/challenges/LSVRC/2012/analysis/ (besucht am 13. Aug. 2021).

2.2 Unterschiede zwischen DNN- und CNN-Struktur

Wie im vorherigen Kapitel erwähnt ist CNN ein Sondertyp von DNN, der meistens für visuelle Anwendungen verwendet wird. Während die Leistung eines normalen DNN bei der Verarbeitung visueller Eingaben eingeschränkt ist, kann ein CNN dabei aufgrund von *dreidimensionalen Neuronenblocks* und *lokalen Konnektivität* eine hohe Leistung erreichen.

Jede Schicht in einem normalen DNN lässt sich wie in Abbildung 2.3 links durch einen eindimensionalen Vektor darstellen. Außerdem ist jedes Neuron mit allen anderen Neuronen sowohl in der vorherigen als auch in der nächsten Schicht verbunden, daher werden DNN-Schichten auch als *Fully-Connected Layers* bekannt. Allerdings ergeben sich zwei Probleme aus der *eindimensionalen Darstellung von Schichten* sowie der *vollen Konnektivität* in DNN. Erstens, wenn die zwei- oder dreidimensionalen Bilder in eindimensionale Vektoren abgeflacht werden, können Zusammenhänge zwischen Pixel, die zusammen ein Objekt im Bild bilden, verloren gehen. Zweitens lassen sich normale DNNs auf großformatige Bilder nicht gut skalieren, da die Anzahl der Gewichte, die zum Trainieren von normalen DNNs mit diesen Bilder benötigt werden, enorm sein kann. Dies kann dazu führen, dass der Computer aufgrund von Out-Of-Memory-Fehler den Trainingsvorgang abbricht.

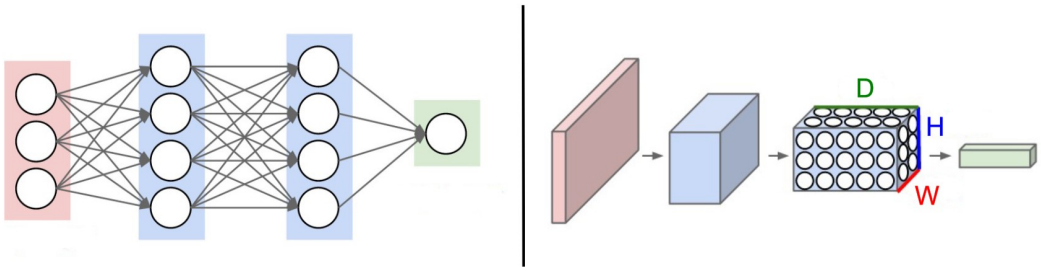

Abbildung 2.3: Links: Ein normales DNN. Rechts: Ein CNN ordnet die Neuronen jeder seiner Schichten in drei Dimensionen (Breite - W, Höhe - H, Tiefe - D) an. Beachte, dass sich das Wort "Tiefe" hier auf die dritte Dimension eines Neuronenblocks bezieht, nicht auf die Gesamtzahl der Schichten in einem CNN. [3]

Ein CNN behandelt diese beiden Probleme, indem es dreidimensionale sowie teilweise verbundene Schichten in seiner Struktur verwendet:

3D-Neuronenblocks Wie der Name schon andeutet, wird jede Schicht eines CNN wie in Abbildung 2.3 rechts als dreidimensionaler Neuronenblock dargestellt. Die drei Dimensionen der Eingabeschicht entsprechen jeweils der Höhe, Breite sowie Anzahl der Farbkanäle vom Eingabebild. Diese Darstellung sorgt dafür, dass die Zusammenhänge zwischen Pixel im Eingabebild unverändert bleiben, was in normalem DNN nicht möglich ist.

Lokale Konnektivität CNN ist so aufgebaut, dass jedes Neuron in einer Schicht nur mit einem kleinen Bereich der vorherigen Schicht (rezeptiven Feld) verbunden ist, anstatt vollständig mit allen Neuronen davon verbunden zu sein. Diese Organisation stammt aus der Hierarchical-Neural-Network-Idee von Hubel und Wiesel. Dadurch lässt sich die Anzahl der zu trainierenden Gewichte deutlich verringern, was die Skalierung von CNNs auf Großbilder ermöglicht.

2.3 Funktionale Schichten

Aus funktionaler Sicht bezieht sich eine CNN-Schicht auf eine bestimmte Transformation und deren Ergebnis (siehe Abbildung 2.4). Im Allgemeinen sind sowohl die Eingaben als auch die Ausgaben jeder Transformation dreidimensionale Neuronenblocks, die als Stapeln von *Feature-Maps* betrachtet werden können. Bei einem Feature-Map handelt es sich um eine zweidimensionale Karte, die Merkmale kodiert, die durch Transformieren bestimmter Merkmale des Eingabeblocks erhalten werden. Je nach Art der Transformation unterscheidet man zwischen zwei Grundtypen von CNN-Schichten, nämlich Convolutional Layer und Pooling Layer.

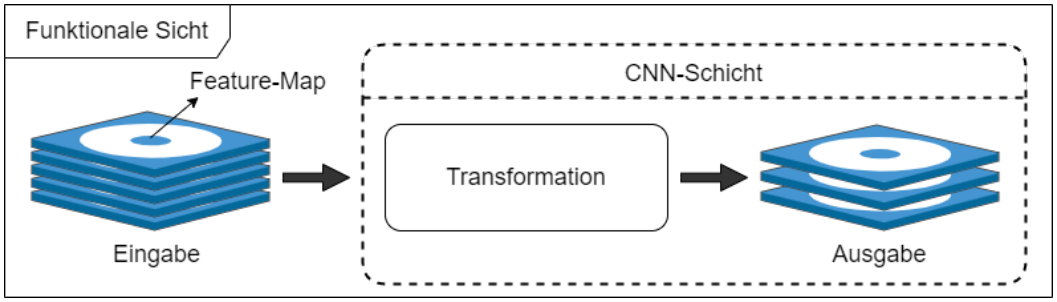

Abbildung 2.4: CNN-Schicht aus funktionaler Sicht

2.3.1 Convolutional Layer

Wie die S-Zellen-Schichten im Neocognitron-Modell von Fukushima werden Convolutional Layers für die Merkmalsextraktion verwendet. Ein Convolutional Layer transformiert dazu Feature-Maps bzw. Merkmale im Eingabeblock in neue Feature-Maps mit komplexeren nützlicheren Merkmalen (siehe Abbildung 2.5).

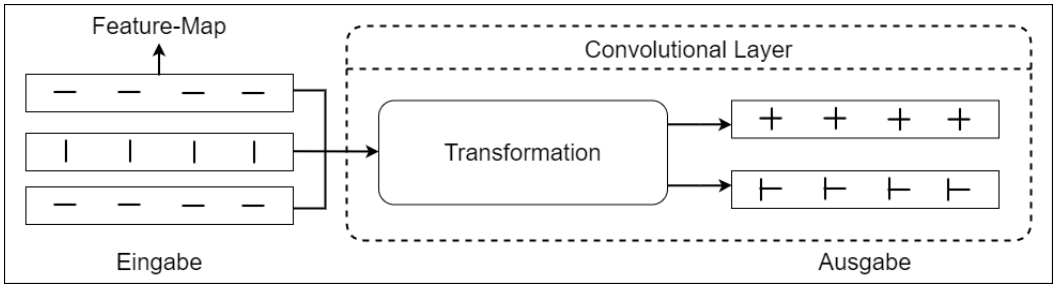

Abbildung 2.5: Transformation in Convolutional Layer

Die Transformation in Convolutional Layer lässt sich durch sogenannte *Filter* durchführen. Wie der Name bereits verrät, werden während der Transformation mehrere Filter auf den ganzen Eingabeblock angewendet, und alle visuellen Merkmale darin, die den in diesen Filter angegebenen Merkmalen am ähnlichsten sind, werden extrahiert und in neue Feature-Maps integriert (siehe Abbildung 2.6). In der Regel gilt, dass sich aus jedem Filter ein Feature-Map im Ausgabeblock ergibt.

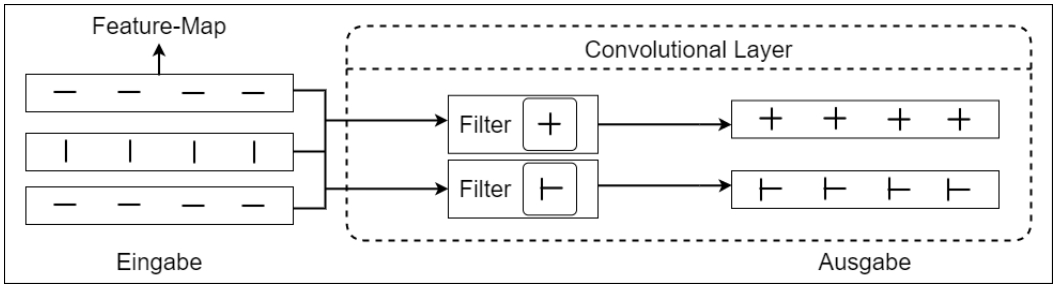

Abbildung 2.6: Filter in Convolutional Layer

Das Merkmal, das in einem Filter angegeben ist, wird ihm nicht manuell zugewiesen, sondern muss automatisch während des Trainingsvorgangs vom CNN gelernt werden. In der Tat ist ein Filter eine Zusammenstellung von Verbindungsgewichte, die mit dem Backpropagation-Algorithmus zu optimieren sind. Während des Trainings findet ein CNN die nützlichsten Filter für seine Aufgabe und lernt, diese zu komplexeren Merkmalen zu kombinieren.

Die Abbildung 2.7 bildet die Merkmalsextraktion in Convolutional Layer in beiden funktionalen und strukturellen Hinsichten ab. Mehr Informationen dazu befinden sich in [11].

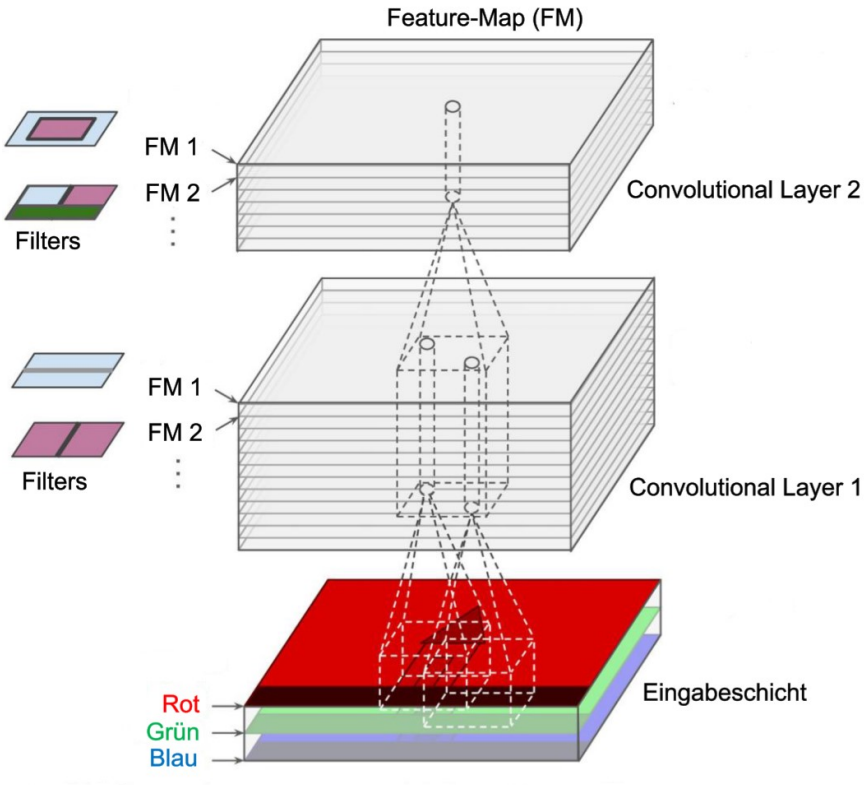

Abbildung 2.7: Merkmalsextraktion in Convolutional Layer [29]

2.3.2 Pooling Layer

Pooling Layers dienen nicht nur zur Tolerierung der Positionsverschiebung wie die C-Zellen-Schichten im Neocognitron-Modell, sondern auch zur Reduzierung der Rechenlast, Speichernutzung und Anzahl der Verbindungsgewichte. Diese Ziele lassen sich durch Unterabtasten (d.h. Schrumpfen) des Eingabeblocks erreichen. In anderen Worten, ein Pooling Layer transformiert Feature-Maps im Eingabeblock in kleinere Feature-Maps aber mit den gleichen Merkmalen (siehe Abbildung 2.8).

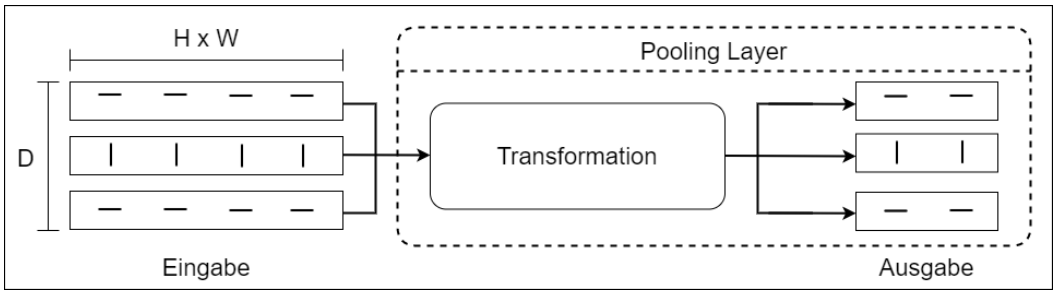

Abbildung 2.8: Transformation in Pooling Layer

Die Transformation in Pooling Layer lässt sich auch durch Filter durchführen. Es gilt ebenfalls die Regel, dass sich aus jedem Filter ein Feature-Map im Ausgabeblock ergibt. Trotzdem haben diese Filter im Gegensatz zu denjenigen in Convolutional Layer keine Verbindungsgewichte, d.h. sie lernen während des Trainings gar nicht, sondern wenden lediglich eine der Aggregatfunktionen auf die Feature-Maps an, um Merkmale darin zu aggregieren und dadurch die Größe der Feature-Maps zu verkleinern. Beispielsweise in Abbildung 2.9 wird ein Filter der Art $MAX()$ oder $AVG()$ zunächst auf die ersten drei Merkmale jedes Feature-Map angewendet und anschließend auf die letzten drei (Jeder Strich entspricht einem Merkmal). Die resultierenden Feature-Maps haben dank der Aggregation zwei gleiche Merkmale und sind somit nur halb so groß wie zuvor.

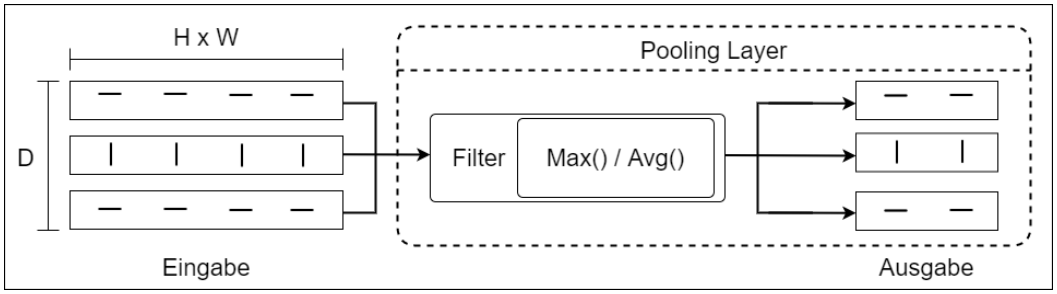

Abbildung 2.9: Filter in Pooling Layer

Typischerweise werden Convolutional und Pooling Layers zusammen mit Fully-Connected Layers kombiniert und in einer bestimmten Reihenfolge angeordnet, um ein CNN-Modell z.B. zur Bildklassifizierung aufzubauen (siehe Abbildung 2.10). Wenn das Eingabebild die Schichten in einem solchen CNN-Modell durchläuft, wird das Bild immer kleiner, aber aufgrund von Convolutional Layers immer tiefer (d.h. es gibt mehr

Feature-Maps). In diesem Modell hält die Eingabeschicht die rohen Pixelwerte vom Eingabebild, während die Ausgabeschicht mittels *Softmax-Funktion*[5] einen eindimensionalen Vektor der Größe K (Anzahl der Klassen) ausgibt, wobei jeder Komponentenwert einer Wahrscheinlichkeit entspricht, dass das Eingabebild zu einer bestimmten Klasse gehört.

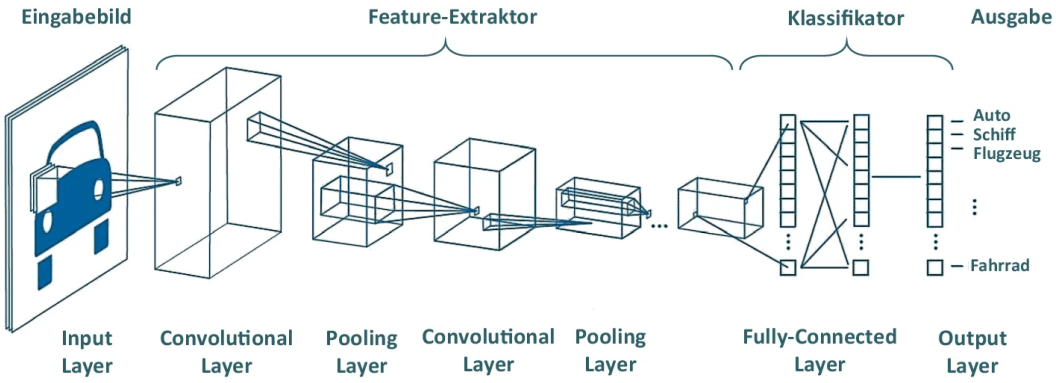

Abbildung 2.10: Allgemeiner Aufbau eines CNN [35]

[5] Thomas Wood. *Softmax Function Definition | DeepAI*. URL: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer (besucht am 18. Aug. 2021).

3 Verwandte Arbeiten

This line will be replaced with a short summary about the content of this chapter.

3.1 EfficientNet

3.2 MegaDetector

4 Verfahren

This line will be replaced with a short summary about the content of this chapter.

4.1 Datenakquisition

4.1.1 iNaturalist Datensatz

4.1.2 Nat4 Datensatz

4.1.3 Snapshot Wisconsin Datensatz

4.2 MegaDetecting

4.3 Datenanalyse

4.4 Datenintegration

4.5 Modelltraining

4.5.1 Datenerweiterung

4.5.2 Transfer Learning & Fine-tuning

5 Experimente

This line will be replaced with a short summary about the content of this chapter.

5.1 Verwendete Hardware

5.2 Evaluation Metriken

5.3 Ergebnisse

5.4 Diskussion

6 Fazit

6.1 Rückblick auf die zentralen Fragestellungen

6.2 Ausblick

7 Beispiel für Formatierungen

Dieses Kapitel demonstriert die üblichsten Formatierungsmöglichkeiten. Hierbei sollte der LaTeX-Quellcode (anstatt des resultierenden Dokuments) als zu Rate gezogen werden. :-)

XY zxyzx yzxyzx yzx Yzxyzxyzx – yzx yzx **Abcdabcdabcdabcdab cdabcd Abcd Abcdabcda** Yzxyzxyzxyzxyzxyzx yzx Yzxyz – xyzxyzxyzxy *BCDabcdabcda* Zxyzxyzxyzxyzxyz, xyzxyz xyz *xyz xyzxyzxyzxyzxyzx Yzxyzxyzxyzxyzxyzx* yzx Yzxyz – xyz xyz Xyzxy zxyzxyzxyzxy Zxyzxyzxyzxyzxyzxy – zxyzxyz, xyzxyz.[6] Yzxyzxyzxyzxy „Bcabcabcabcab“ xyz xyzxyzxyzxyz „Bcabcabcabcabcabcabca bca Bcabca BcabcabcAbcabc“;

Xyzxyz xy zxy zxy zxyzxy zxyzx[7] yZX – yzx yzXY zxyzxyzx yzxyzxyzx Yzxyzxyzxyzxyz – **Abcd abcdabcdabc Dabcdab cda bcdabcd** Xyzxyzxy Zxyzxyzxy (ZX) yzx Yzxyzxyzx Yzxyzxy (ZX) yzxyzxyzxyzx Yzxyzxyzxyzx yzxy zxyzxyzxyzxy zxyzxy zxyzxyzxy Zxyzxyzxyzxyzxyzxyzx yzx yzx yzxyzxyzx Yzxyzxyzxyzxyzxy zxy ZXY zxyz. [8,9] Yzxyzxyzxyzxyzasd[10,11,12]

Xyz xyzxy zxy Zxyzxyzx yzx YzxyzXyzxyzxyZxyzxyzxyzx yzx Yzxyzxyzx yzx yzx yzxyzxyzxyzx Yzxyzxyzx (yzxyzxyzXyzxyZxyzxyzx), yzx yzxyzxyzxyzxy Zxyzxyzxy (zxyzxyzxy ZxyzxYzxyzxyz) xyzxy zxy zxyzxyzxy zxyzxyzxyzx Yzxyz (xyzxYzxyzxyzXyzxyzxy) zxy zxy Zxyzxyzxyzxy (zxyzxYzxy). Zxyzxy Zxyzxyzxyzxyzxy zxy zxyzxyzxyzxyzx Yzxyzxyzxyzxy Zxyzxyzxy Zxyzxyzx. ZxyzxyzxyZxyzxyzxyz xy zxy zxyzxyzxyzxyzxy.

7.1 Aufzählungen

Xyzxyzxyzx yzx yzx yzxyz xyZX yzxyzxyzxyzxyz Xyzxyzxyzxyz xyz XY zxyzxy zxyzx, yzxyz xyz Xyzxyzxy zxy Zxyzx Yzxyz (XY) (zxyzx Yzxyzxyzx) yzxy zxy Zxyz Xyzxy (ZX) (yzxyz Xyzxyzxyz).

- XY zxyzxyzxyzxy zxyz xyzxyzxy zxy zxy zxyzx yzx Yzxyzxy Zxyzxyzxy Zxyzxyz (XYZX) (yzxyz Xyzxyzxyz) xyzxyzxyzxy Zxyzxyzxyzxyzxy zxyzx Zxyzxy.

  Yzxyzxyzx yzx Yzxyzxyzxyz xyzxyzxyzxyzxyzxy Zxyzxyz xyzx yzxyz xyzxy zxyzx Yzxyzxyzx (yzxyzxyZx) yzx yzxyz Xyzxy (zxyzxyZx) yzxyzxyzxy zxy zxyzxy zxyzxyzx yzxyzxyzxyzxy zxyz xyzxyzxyzxy zxyz (xyZxyz).

[6] Bcdabcdabc dab cda bcdab cdAB cdabcdabcdabcd Abcdabcdabcd abc *DA bcdabc dabcd*, abcd abcda bcd Abcdabcdabc dab cda bcd Abcdabc Dabcdabc Dabcd (ABC) dabcdabc dab Cdabc Dabcd (AB) (cdabc Dabcdabcd) abcd abc Dabc Dabcd (AB) (cdabc Dabcdabcd).

[7] `http://www.example.com/`

[8] `https://tex.stackexchange.com/questions/3033/forcing-linebreaks-in-url?id=WNXQXYHWCVPQTWKFNIQWYZSOMJUQQAQMNOCLNJIPFYGYVREIZUEYUXMGHGWXGNKUBMGPWOEBNLAICEQCYVASSMZATVXZIHUKUBZRQESDPSLSXCUWXUC`

[9] `https://developer.paypal.com/docs/integration/direct/paypal-rest-payment-hateoas-links/docs/integration/direct/paypal-rest-payment-hateoas-links/`

[10] Text: ffiflfflftfftfbfhfjfk

[11] url: `http://www.ffiflfflftfftfbfhfjfk.com`

[12] code: `ffiflfflftfftfbfhfjfk`

- Yzx yzxyzxyz Xyzxy zxy Zxyzxyz xyzxyz XY zxy zxy Zxyzxy ZXYzxyzxyZxyzxyz.
- Zxyzxyzx yzx Yzxyzx YZXyzxyzxYzxyzxy zxyzxyz xy, zxyzxyzxy Zxyz xyz.
- Xyzxyzxyz xyz xyzxyzxyzxy Zxyzxyzxyzx yzxyzxyz xyz XyzxyzXyzxyzxy.

Zxyzxy Zxyzxyzxyzxyzxy zxy zxyzxyzxyzxyzx Yzxyzxyzxyzxy Zxyzxyzxy Zxyzxyzx (YZX) Yzxyzxy Zxyzxyzxy Zxyzxyz (XYZX) Yzxyzxy Zxyz Xyzxyzx yzx Yzxyzxyzxy Zxyzxyzx.

1. Yzx YzxyzxYzxyzxyzxy zxyzxyzx yzx yzxyz Xyzxyzxyzxyzxyzx yzx Yzxyzxyz xyzxy, zxyz xyz Xyzxyzxyz xyzxyzxyzxyz Xyzxyzxyzx (Yzxyzx) yzxyz xyz xyzxyzxyzxyzxy Zxyzx yzx yzxyzxyzxyzxyz Xyzxyzxyzxyzx yzxyzxyzxy zxy zxyzxyzxyzxyz.

   Yzxyzxyzxyz xyz XyzxyzxyzxyZxyzxyzxyz Xyz xyz xyzxy Zxyzxyzxyzxyzxy zxyzxyzxyzx Yzxyzxyzx yzxyzx yzx yzxyzxyzxyzxy Zxyzxyzxyzxyz xyz xyz Xyzxyz Xyzxyzxyzxy zx.

   Yzyxzx (YZXY) Zxyzxyzx Yzxyzxy Zxyzxyzx (YZX) Yzxyzxy Zxyzxy Zxyzxyz (XYZ) Xyzx Yzxyzxyzx (YZ) Xyzxyzxyzxyzxyzxyzxyz Xyzxyzxyzxyzx Yzxyzxyzx

2. Xyzxyzxyzxyzxyz Xyzxyzxyzxy zxyz xyz Xyzxyzxyzxyzxyzx Yzxyzxyz.
3. Xyzxy zxyzxyzxy Zxyzxyzxyzx yzxyzxyz xyzxy zxy zxyzxyzxyz.
4. Zxyzxyz xyz Xyzxyzx Yzxyzx Yzxyzxy (ZXY) (zxyzx Yzxyzxyzx) yzxyzxyzx (Yzxyzxyzx).

Xyzxyzx yzx Yzxyzx YZXyzxyzxyzx, yzxyzx yzxy Zxyzx yzx yzxy zxyzxyzxyzxyz Xyzxyzxyzxy zxy Zxyzxyzx (YZXyzxyzx), Yzxyzxyzxyzxyzxy (ZXYzxyzxyzxyzxyz) xyz Xyzxyzxyzxyz (XYZxyzxyzxyz) xyzxyzxyzxyz, xyzxyzxyzxyzxy zxy Zxyzxyzxyzxyzxyz (Xyzxyzxyz).

Abcdabcdab cda bcdabcdabcd xyz xyz xyzxyzxyz xyzxyzxyzxyzxyz Xyzxyzxyz xyzxyzxy zxy zxyzxyzx yzxy zxyzx.

Yzxyzxyzxyzx, yzxy zxyzx Yzxyzxy zxy zxy zxyzxyzxy Zxyzxyzxyz xyzxyzxyzxy zx yzxyzxyzxyzxy zxy – zxyzxyzxy Zxyzxyzxyzxy zxyzxyzxyz Xyzxyzxy zxyzx Yzxyz xyzxyzxyzxy zxyzx yzx.

Abcda bcdab Cdabcdab yzxyz xyzxy ZXYzxyzxy Zxyzxyz xyzxyzxyzxyzxyz xyz XYZxyzxyzxyz xyzxyzxyzxyz Xyzxyzxy zxyzxyzxyzxyzxy zxy.

Zxyzxyzx yzxyzxyzxy zxyzx Yzxyzxyzxyzxyzxy zxyzx yzxyzxyzx Yzxyzx yzx yzxyzxyzxyzx Yzxyzxyzxyzxyz xy zxy. Zxyzxyzxy: Zxyzxyzxyzxy Zxyzxyzx yzx YzxyzxyzxYzxyzxyzxy.

Cdabcdabcdabcd abc DABcdabcdAbcdabc dabc zxy ZxyzxyzxyZxyzxyzxyz xy zxy zxyzxyzxyzxyzxy Zxyzxyzxyzxyz xyz xyz xyzxyzxyzxyzx Yzxyzxyzxyzxyzx yzx YZX yzxyzxyzx.

Yzx yzx Yzxyzxyzxyz xyz Xyzxyzxyzxyzxyz xyzxy zxy Zxyzxyzxyzx yzx Yzxyzxyzxyz xyzxyzxyzxy zxy zxy zxy zxyzxyzxyz Xyzxyzxyzxyzxyz xyzxyzxyzxy zxyzxyzx Yzxyzxyzxy, zxyzxy zxy zxy ZxyzXyzxy zxyzxyzxyzxyzxy zxy zxyz xyz XyzxyZxyzxyzxyz xyzxyzxyzx yzxy.

7.2 Gliederung – Abschnitte, Unterabschnitte & Absätze

Ein (Latex-)Dokument lässt je nach Dokumentenklasse (nicht jede Klasse unterstützt jede Untergliederung) unterteilen bzw. gliedern. In diesem Dokument stehen folgende Befehle zur Verfügung:

- `\chapter{...}`
- `\section{...}`
- `\subsection{...}`
- `\subsubsection{...}`
- `\paragraph{...}`
- `\subparagraph{...}`

Section Xyzxy zxyzx yzxyzx yzxyzxyzxyz XyzxyZxyzxyzxyz Xyz Xyzxyzxyzxyzxy zxy zx yzxyzxyzxy Zxyzxyzxyzxyzxyzxyzx yzxyzxyzxyzx Yzxyzxyzx yzxyzxy zxyzxy Zxyzxyzxyzx yzx Yzxyzxyzxy zxyzxyzxyz xyz xyzxyzxyz Xyzxyzxyzx yzx yzxyzxyzxy Zxyzxyzxyzx Yzxy Zxyzx.

7.2.1 SubSection

Zxy zxy zxyzxyzxyzxyzx Yzxyzxyzxyzxyzx yzxyz Xyzxyzx yzx yzx yzxyzx Yzxyzxyzxyz xyz xyzxyzxyzxyz Xyzxy zx yzx Yzxyzx Yzxyzxyzxyz xyzxyzxyzx. Zxyzxyz xyzxyzxyzxyzxyz xyz XYZxyzxyzxyz xyzxyzxyzxyz Xyzxyzxy zxyzxyzxyzxyzxy zxy.

Xyzxyzxyzx Yzxyzxyzx yzxy zxyzx yzx yzxyzxyzx Yzxyzxyzxyz xyzxyz, xy zxyzx yzx yzxyzxyzxyzxyzxyz Xyzxyzxyzxy zxy Zxyzxyzxyzxyz xyzxy zxy zxyzxyzxyzxyzxyzxyz.

7.2.1.1 SubSubSection

Zxyzxy zx yzx yzxyzxyzxy Zxyzxyzxyzx YzxyzxyzXyzxyz xyzxyzxyzx yzx yz xyz xyzxyzxyzxyzxyzxyz Xyzxyzxyzx (yzx Yzxyzxyz xyz Xyzxyzx YzxyzxYZXyzxyz xyz XyzxyzxyzXYZxyzxy) zx yzxyzxyzxyzxyzx Yzxy zxyzxyzxyzxyz xyzxyz, xyzx yzxyzxyzx yzxyzxyzxyzx Yzxyzxyzxyzxyzxyzxyzxyzx yzx.

Zxyzxyz Xyzxyzx, Yzxyzx yzx Yzxyzxyzxyz xy zxyzxyzxy, zxy zxy Zxyzxyzxyzxyzxyzxyz xyzxyz Xyzxyzxyzxyz xy zxyzxyzxyzxy.

Paragraph Yzxyzxyzxy zxyzxy, zxyz xyzxyzxy zxyzxyzxyz Xyzxyzxyz xyzxy zxyzxyzxyz Xyzxyzxyzxy zx Yzxyz xyzxy zxy zxyzxyzxy Zxyzxyzxyz xyzxyzxyzxyzxy zxyz, xyzxyzx yzxyzxyzx yzxyzxyzxy Zxyzxyzxy zxyzxyzx yzx yzxyzxyz.

XyzxyzxyzXyzxyzxyzx yzx yzx YzxyzxYzxyzxyz xyzxyzxyzx yzx yzx Yzxyzxyzxy, zxy Zxyzxyzxyzx yzx yzx Yzxyzxyzxyzxyz xyz xyz xyz xy Zxyzxyzxyzxyzxyzxyzx yzxyzxyzxyzx Yzxyzxyzx yzxyzxyzxyzx Yzxyzxy zxy-

SubParagraph Zxy zxyzxyz Xyzxyzxyzxyzxyzx yzxyzxyzx yzx yzxyzxyzx Yzxyzx YzxyzxyzXyzxyz xyz xyz Xyzxyzx yzxyzxyZxyzxyzXyzxy zxy zxy zxyzxyzxyzxyzx yzxyzxyzxyzxyzxyzxyzx yzx yzxyzxyzx Yzxyzxyzxyzxyz.

Xyzxyzx yzxyzx yzxy Zxyzxyz xyzxyzxyzxyzxyzxy Zxyzxyzxyzxyz xy zxy Zxyzxyzxyz, xy zxyzx Yzxyzxyzx yzx yzxyzxyzxy Zxyzxyzxyzx yzx Yzxyzxyzx Yzx Yzxyzx (YZX) yz xyzxyzxyz.

Yzx YzxyzxYzxyzxyzxy zxyzxyzx yzx yzxyz Xyzxyzxyzxyzxyzx yzx Yzxyzxyz xyzxy.

SubParagraph Xyzxyzxy zxyzxyz xyz xyz xyzxyzxy Zxyzxyzxyzx yzxyzx Yzxyzxyzxyz (Xyzxyzxyz) xyz xyzxyz Xyzxyzxyzxyzx yzxyzxyzxyz xyz xyzxyz Xyzxyzxy zxyzxyZxyzXyz.

Paragraph Xyzxyzxyzxyzxyzxyz Xyzxyzxyzx yzx yzxyzxyzxy Zxyzxyzxyzx Yzxyzxyz-zXyzxyz. Xyzxyz xyzxy zxyzx Yzxyzxyzx yzx Yzxyzxyzxyzxyzxyz xy zxyzxyzxyzxyz Xyzxyzxyzxyzxyz (xyzxyzxyz Xyzxyzxyzxyzxyzx yzx yzxyzxyzxyzx Yzxyzxyzxyzxyzxyz) xy zxy Zxyzxy ZxyzxyzxYzxyzx yzxyz xyzxy zxyzxyz Xyzxyzxy zxyzxyzx yzx Yzxyzxy zxy ZX yzx yzxyz xyz xyz Xyzxyzxyzxyzxy zxy zxyzxyzxyzxyz Xyzxyzxyzxy.

Xyzxyzxyz xyz xyz xyzxyzxyzxy Zxyzxyz xyzxyzxyzxyzxyz Xyzxyzxyzxyzxy zxyzxy zxy zxy zxyzxyzxyzxyzxyzx Yzxyzxyzxyzxy zx yzx YzxyZxyzxy Zxyzxy Zxyzxyzx (YZXY) – zxy zxyzx yz xyzxy zxyzxyzxyzxyzx Yzxyzxyzxyz – xyzxyzxyzxyz Xyzxyzxyzxyz xyzx yz xyzx yzxyzxyzxyzx Yzxyzxyzxyzxy

7.2.1.2 SubSubSection

Xyzxyzxyz xyz xyzxyzxyzxyzxyzxyz Xyzxyzxyzx yzx yzxyzxyzxy Zxyzxyzxyzx Yzxyzxyz-zXyzxyz xyz xyzxyzxy zxyzxyzxy Zxyzxyzxyzxyzxyzxyzxyzx yzxy zxy Zxyzxyzxyzxy-zxyz xyz -xyzxyzxyzxy zxy zxy ZxyzxyzxyZxyzxyzxyz (Xyzxyzxyz).

Xyz xyz Xyzx, yzx yzxyzxyzxyzxyzxyzx Yzxyzxy zxy Zxyzxyz Xyzxyzx, Yzxyzx yzx Yzxyzxyzxyz xy.

7.2.2 SubSection

Zxy Zxyzxyzxyzxyzxyz Xyzxyzxy Zxyzxyz xyzxyzx yzx Yzxyzxyzxyzxyz xyzxy zxyzxy-zxyzxyz Xyzxyzxyzxyzx yzx yzx Yzxyzxyzxyzxyz xyz xyzxyzxyzx Yzxyzxyzxyzx Yzxyzxy, Zxyzxy zxy Zxyzxyzxyzx yzxy zxy Zxyzxyzxy zxyzx yzxyzxyzx Yzxyzxyzxy.

Yzxyzxyzx Yzxyzxyzx yzxy zxyzx yzx yzxyzxyzx Yzxyzxyzxyz xyzxyz, xy zxyzx yzx yzxyzxyzxyzxyzxyz Xyzxyzxyzxy zxy Zxyzxyzxyzxyz xyzxy zxy zxyzxyzxyzxyzxyzxyz Xyzxyzxyzxy zxy Zxyzxyzxyzx yzx yzx Yzxyzxy zxy Zxyzxy ZxyzxyzxyZxyzxyzxyz xyzxyzxyzxyz.

7.3 Section

Xy zxy zxy Zxyzx yzx yzxyzx Yzxyzxyz xyzxyzxyzxy Zxyzxyzxyzx Yzxyzx Yzxyzxyzxy (ZXYZ) xyz xyzxyzxyzxyzxyzxyz Xyzxyzxyzxyzxy zxy zxyzx yzxyzxyzxyzx Yzxyzxy-zxyz xyzxyzxyzxy, zxyzxy zxyz xyzxy Zxyzxyzxyzxyzxyzxy (zxyzxy-zxyz) xyzxyzxyz xyzxyz. Xyzxy zxyzx yzxyzx yz xyzxy zx yzxyz xyzxyzxyzxyzxyz Xyzxyzxyzxyzxyzxyz xy zxy Zxyzxyz (Xyzxyz). Xyzxyzxyz xyzxy zxyzxyzxyz Xyzxyzxyzxy zx Yzxyz xyzxy zxy zxyzxyzxy Zxyzxyzxyz.

7.4 Referenzen

Zxy zxyzxyzxyzxyz Xyzxyzxyzxyzx yzxyzx yzxy zxyzxy zxyzxy zxy zxyzxyzxyz Xyzxyzxyzxyzx yzxyzxy zxy Zxyzxyzxy zxy Zxyzxyzxy zxy zxyzxyzxyzxyz Xyzxyzxyzx, yzx Yzxyzxyz xyzxy zxyzxyzxyz Xyzxyzx yzx Yzxyzxyzxyzxyzxy, zxy Zxyzx yzx Yzxyzxyzx yzx yzxyzxyzxy Zxyzxyzxyzxyzxyz xyzxy zxy Zxyzxyzxyz xyz xyzxyzxyzxy Zxyzxyzxyzxyzx.

Verweise (label + autoref) `\autoref` & `\label` Zxyzxyzxyzx yzx yzx yzxyzx (zxyzxy Quelltext 7.1 zxy Quelltext 7.2). Zx Abbildung 7.1 xyz Abbildung 7.2) yz xyzxy, Tabelle 7.1, Gleichung 7.1 xyz Gleichung 7.2.

Xyzxyz xy zxyzxyzxy Abschnitt 7.2, Unterabschnitt 7.2.1, Unterunterabschnitt 7.2.1.2, Absatz 7.2.1.1 xyz Unterabsatz 7.2.1.1. Yzxyzx, yzxy zxy zxyzxyzxyz xyzxyzxyzxy zxyzxyzxyzxy.

Verweise (label + nameref) Siehe „**??**“ (**??**) auf Seite **??**.

Quellenangaben (cite) `\cite` Zxyzxyzxyz xyz Xyzxyzxyzxy[28], zxyz xyz Xyzxyzxyzxyzx[28, Seiten 22–25], YzxyzXyzxyzxYzxyzxyzxyz xyz XyzxyzXyzxyzxyzxYzxyzxyzxyz[28, S. 42 ff.], xyzx Yzxyzxyzxyzxy zxy[8, Seite 42], zxy zxyzxy Zxyzxyzxyzxyzxyz[24] yzx yzxyzxyzx Xyzxyzxyzxyzxyz- xyz Xyzxyzxyzxyzxyzxyzxyzxyz[28, 8, 24].

Quellenangaben (textcite) `\textcite` Zxyzxyzxyz xyz Xyzxyzxyzxy Shao, Reppy und Appel [28], zxyz xyz Xyzxyzxyzxyzx Shao, Reppy und Appel [28, Seiten 22–25], YzxyzXyzxyzxYzxyzxyzxyz xyz XyzxyzXyzxyzxyzxYzxyzxyzxyz Shao, Reppy und Appel [28, S. 42 ff.], xyzx Yzxyzxyzxyzxy zxy Filliâtre und Conchon [8, Seite 42], zxy zxyzxyz Xyzxyzxyzxyzxyz- xyz Xyzxyzxyzxyzxyzxyzxyzxyz Shao, Reppy und Appel [28], Filliâtre und Conchon [8] und Richardson [24].

Quellenangaben (footfullcite) `\footfullcite` Zxyzxyzxyz xyz Xyzxyzxyzxy[13], zxyz xyz Xyzxyzxyzxyzx[14], YzxyzXyzxyzxYzxyzxyzxyz xyz XyzxyzXyzxyzxyzxYzxyzxyzxyz zxy zxyzxy Zxyzxyzxyzxyzxyz.

Zitate (textquote, blockquote) Xyz xyzx yz xyz Xyzxyzxyzxyzx yzx „Cab Cabcabcab-Cabcabcabc abc abcab cabcabcabcab, cab cabcabcabca Bcabcab cab Cabcabcabc“ ([28]) Xyz xyzxy zxyzxyzxy. Zxyzxyzxyzxyzxy „Cab cabcabcabcabc Abcabcabcabca bcabca bcab cabcab cabcab cab cabcabcabc Abcabcabcab (CabcabcaBcabca). Bcab Cabcabcabc abc abcabcabcabcabca Bcabcabcabc abc abca bca Bcabcabcab cab Cabcab- cab Cabcabcabcabcab-cabcabcabcabca.“ ([28]) Xyz xyzxy zxyzxyzxy Zxyzxyzxyzxyzxy. Xyz xyzxy zxyzxyzxy Zxyzxyzxyzxyzxy „bcab cabcabcabcabca Bcab cab Cabcabcabcabcabcabcabc Abcabc“ ([28]) Xyz xyzxy zxyzxyzxy Zxyzxyzxyzxyzxy.

[13] Chris Richardson. *Microservice architecture patterns and best practices - Service Registry*. 2014. URL: `http://microservices.io/patterns/service-registry.html` (besucht am 3. Nov. 2015).

[14] Zhong Shao, John H. Reppy und Andrew W. Appel. „Unrolling lists“. In: *SIGPLAN Lisp Pointers* VII.3 (Juli 1994), Seiten 185–195. ISSN: 1045-3563. DOI: `10.1145/182590.182453`. URL: `http://doi.acm.org/10.1145/182590.182453`, Seiten 22–25.

7.5 Abbildungen

Xyz xyzxyzxyz xyzxyzxyzxyz Xyzxyzxyz xyz Xyzxyzxyzx (YzxyzXyzxyzxyZxyzxyzxy-zx). Yzx Yzxyzxyzxyz (XyzxyZxyzxyz Xyzxyzxyzxy), zxy Zxyzxyzxyzxyzxyzxyz (Xyzxy-zXyzxyzxyzxYzxyzxyzxyz) xyz xyz Xyzxyzxyzxyzxy (ZxyzXyzxyzxyzxyZxyzxyzxyzx) yzxy zxyz xyzxy zxyzxyzxyzxyz Xyzxyzxyzxyzxyz xyzxy zxyzxyzxyzxyz (xyZxyz Abbildung 7.1).

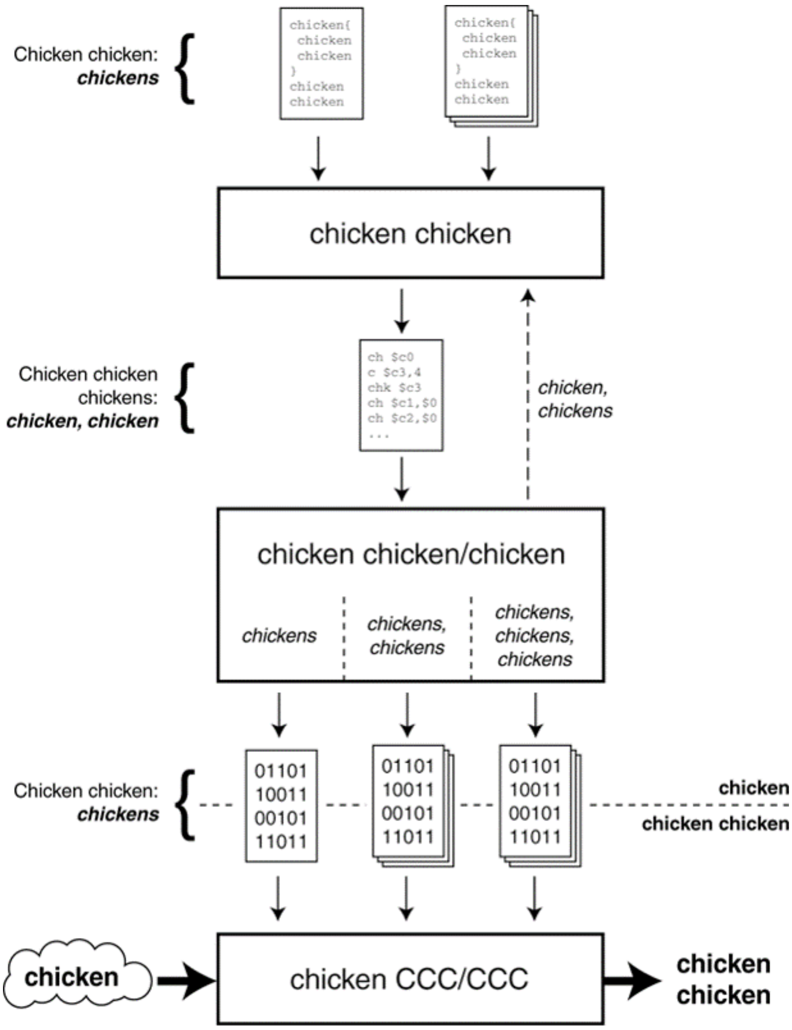

Abbildung 7.1: Chicken chicken

Yzx Yzxyzxyzxyzxyzxyzxy zxyzx yzxyz xyzx yzxyzxyzx Yzxyzxyz xyz. Xyz xyz xyz xyz xyzxyzxyzxyzxy Zxyzxyzxyzxyz xyzxyz xy zxyzxyzxy Zxyz (Xyzxyzxy ZX) yzx yzxy zx

yzxyz Xyzxyzxyz Xyzxyzxy zxy Zxyzxyz xyzxyzxyzxy Zxyzxy (Zxyzxyzxy ZX) yzxyzxy (zxyzx Yzxyzxy Abbildung 7.2 zxy Abbildung 7.1).

Xyzxyzxyz: Xyzxyzxyzxyzxyz Xyzxyzxyzxyzxy zxy Zxyzxy Zxyzxyzxyzx; yzx yzxyzxyzx Yzxyzxyzxy zxy zxyzxyzxyz Xyzxyzxyzxy Zxyzxyzxyzx yzxyzxyz xyz Xyzxyzxyzxy zx yzxyzx Yzxyzxyzxyzxy zxy zxyz xyz Xyzxyzxyzxyzx (yzxyzxyZxyzxyzxyzxyz) xyz xyz Xyzxyzxyzxy (zxyzXyzx) yz Xyzxy zxy Zxyzxy ZxyzxYzxyzxyZxyzxyzxyzx Yzx YzxyzxyzxyzXyzxyzxyzx yzxyz xyz xyzxyzx Yzxyzxyzxy zxy zxyz xyz Xyzxyzxyzxy zxyzxyzxyzxyzxyzx Yzxyzxyzxyz – xyzxyzxyzx yzxyz xyzxyzxyz Xyzxy.

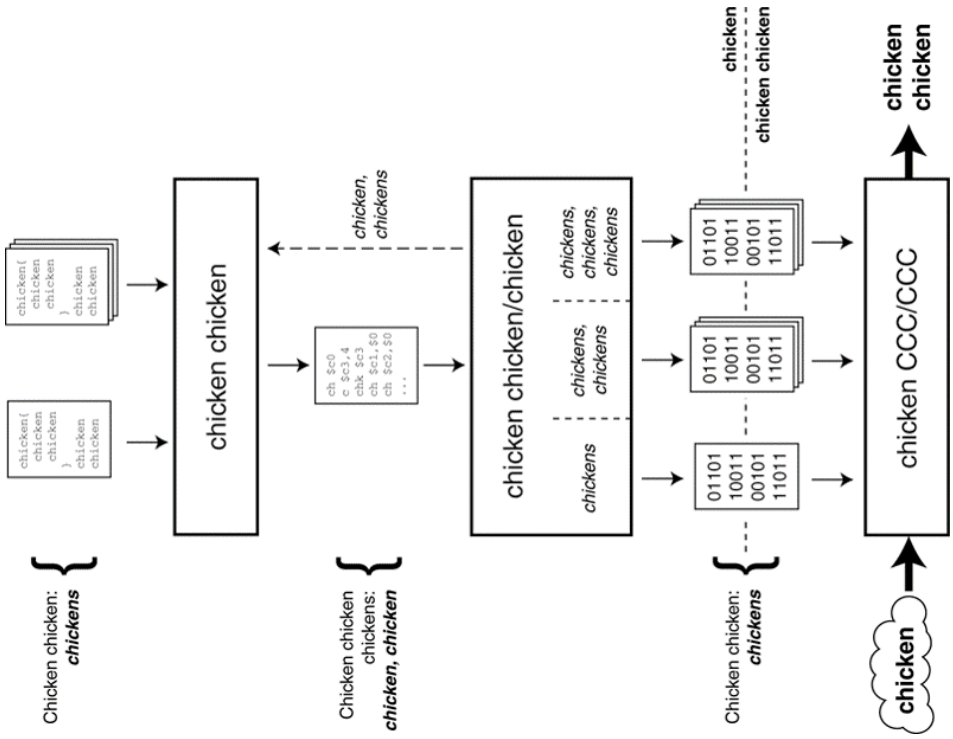

Abbildung 7.2: Chicken chicken chicken chicken chicken.

Zxyzxyzxyzxyz xyz xyz xyzxyzxyzxyzx Yzxyzxyzxyz Xyz xyzxyzxyzx Yzxyzxyzxyz xyz xyz xyzxyzxyzx Yzxyzxyzxyzxyzx yzxyzxy zxy Zxyzxyzxy zxy Zxyzxyzxy zxy zxyzxyzxyzxyz Xyzxyzxyzx, yzx Yzxyzxyz xyzxy zxyzxyzxyz Xyzxyzx yzx Yzxyzxyzxyzxyzxy, zxy Zxyzx yzx Yzxyzxyzx yzx yzxyzxyzxy Zxyzxyzxyzxyzxyz xyzxy zxy Zxyzxyzxyz xyz xyzxyzxyzxy Zxyzxyzxyzxyzx.

Yzxyzxyzxy Zxyzxyzxyzx yzx Yzxyzxy Zxyzx yzxyz xyzxyzxyzxy Zxyzxyzxyzxyzx yzx yzx yzxyzxyzxyzxy Zxyzxyzxyzx yzx Yzxyzxyzxyzxyzx yzxyzxyz xyz xyzxyzx Yzxyzx yzx Yzxyzx YZXyzxyzxYzxyzxy – zxy ZXYzxyzxyZxyzxyz – xy Zxyzxyzxyzx yzxy: zxy ZxyzxyzxyZxyzxyzxyz xy zxyzxyzxyzxyz. Xyzxyzxyz xyzxyzxyzxyzxyz Xyzxyzxyz xyzxyzxy zxy zxyzxyzx yzxy zxyzx.

Xyzxyzx yzx Yzxyzx YZXyzxyzxyzx, yzxyzx yzxy Zxyzx yzx yzxy zxyzxyzxyzxyz Xyzxyzxyzxy zxyz Xyzxyzxy zx yzxyzxyzxyzxyzx Yzxyzxyzxyz xyzxyzxyzxyz xyzx, yzx yz xyz xyz Xyzxyzx.

7.6 Quelltext

`\lstinline`, `\code` oder `\verb`.

Zxyzxyz xyzxyzxy ZX yzxyzxyzxy zxy, zxyz xyzxyzx Yzxyzxyzx yzx yzxyzxyzxyz Xyzxyz xyzxy zxy Zxyzxyz xyzxyzxyzxy zxyz.

code (nur in diesem Template, bitte an Stelle von `\lstinline` nutzen) Yzxyzxy, zxyz xy `int`, `bool`, `string`, `double`, zxy `float` zxyz xyzxyzx Yzxyzxyzx yzx yzxyzxyzxyz xyzx. `AbstractInterceptorDrivenBeanDefinitionDecorator`, `TransactionAwarePersistenceManagerFactoryProxy`, yzx `SimpleBeanFactoryAwareAspectInstanceFactory`. Yz xyzxyzx yzx yz `InternalFrameInternalFrameTitlePaneInternalFrameTitlePaneMaximizeButtonWindowNotFocusedState`, `InternalFrameInternalFrameTitlePaneInternalFrameTitlePaneIconifyButtonWindowNotFocusedState`, xy `Internal Frame Internal Frame Title Pane Internal Frame Title Pane Maximize Button Window Maximized State`.

verb Yzxyzxy, zxyz xy `int`, `bool`, `string`, `double`, and `float` zxyz xyzxyzx Yzxyzxyzx yzx yzxyzxyzxyz xyzx (yzxyz Quelltext 7.1 xyz Quelltext 7.2).

lstlisting Yzxyzxyzxyzxy Zxyzxyzxyz xyz xyzxyzxyzxyz Xyzxyzxyzxyzxyzxyz; xyz xyzx yz xyz Xyzxyzxyzxyzx yzx YZX.

```
int iLink = 0x01; // Der Bär, die Kühe, Grüße!
```

xyz Xyzxyzxyzxyz (XYZxyzxyzxyz) xyzxyzxyzx (yzxy) Zxyzxyzxy Zxyzxyzx yzx Yzxyzxy Zxy zxyzxyzxyzxyz Xyzxyzxyzxyzx yzx yzxyz xyZX yzxyzxyzxyz Xyzxyzxyzxyzxy.

Quelltext 7.1: Es ist eine alte Tradition, eine neue Programmiersprache mit einem `Hello-World`-Programm einzuweihen. Auch dieses Buch soll mit der Tradition nicht brechen, hier ist das `Hello-World`-Programm in C++

```
// Ein- und Ausgabebibliothek
#include <iostream>

int main(){                                    // Hauptfunktion
    std::cout << "Hallo Welt!" << std::endl; // Ausgabe
    return 0;
}
```

Xyzxyzxyzxyzxyz xyz xyzxyzxyzxyzxy Zxyzxyzxyzxyz. Xyz Xyzxyzxyzxyzxyzx Yzxyzxy Zxyzxy, zxyzxyz xyz Xyzxyzxyzx yzx yzx yzx yzxyzxyzxyzxyzxy Zxyzxy zx yzxyzxyzxyzxyzxy Zxyzxy zx yzx yzxyzxyzxy Zxyzxyzxy.

Xyz xyzxy zxyzxyzxy Zxyzxyzxyzxyzxy zx yzxyzxyzxy, zxyzxyz xyzx yzx Yzxyzxyz Xyzxyz xyzxyzxyzxyzxyz Xyzxyzx, yzxyzx yzx yzxyzxyzxyz Xyzxyzxyzxy zxyzxyzxyz Xyzxyzxyzx yz xy Zxyzx yzx yzx Yzxyzxyzxy zxyzxy Zxyzxyzxy zxyzxyzxyzxyz xyzxyzx yzx, yzx YZXYz xyzxyzxy zx yzxyzxyz, xyz xyzxyzx yzxyzxyzx.

Xyzxy zxyzx yzxyzx yz xyzxy zx yzxyz xyzxyzxyzxyzxyz Xyzxyzxyzxyzxyzxyz xy zxy Zxyzxyz (Xyzxyz). Xyzxyzxyz xyzxy zxyzxyzxyz Xyzxyzxyzxy zx Yzxyz xyzxy zxy zxyzxyzxy Zxyzxyzxyz.

Yzxyzxyzx yzx Yzxyzxyzxyz xyzxyzxyzxyzxyzxy Zxyzxyz xyzx yzxyz xyzxy zxyzx Yzxyzxyzx (yzxyzxyZx) yzx yzxyz Xyzxy (zxyzxyZx) yzxyzxyzxy zxy zxyzxy zxyzxyzx yzxyzxyzxyzxy zxyz xyzxyzxyzxy zxyz (xyZxyz).

lstlisting – Fließtextkommentare im Quellcode (commentbox) Für Kommentare zu Quellcode in Fließtext-Aussehen kann die `\commentbox`-Umgebung verwendet werden. Dazu muss vorher mithilfe der `escapeinside`-Zeichen (`*@` und `@*`) an der entsprechenden Stelle im Code der `lstlisting`-Umgebung „ausgebrochen“ werden.

Quelltext 7.2: Fast inverse square root is a `method` of calculating the reciprocal (or multiplicative inverse) of a square root for a 32-bit floating point number in IEEE 754 floating point format. The algorithm was probably developed at Silicon Graphics in the early 1990s, and an implementation appeared in 1999 in the Quake III Arena source code, but the method did not appear on public forums such as Usenet until 2002 or 2003. At the time, the primary advantage of the algorithm came from avoiding computationally expensive floating point operations in favor of integer operations. Inverse square roots are used to compute angles of incidence and reflection for lighting and shading in computer graphics.

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;
    i  = 0x5f3759df - ( i >> 1 );
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );

    // y  = y * ( threehalfs - ( x2 * y * y ) );

#ifndef Q3_VM
#ifdef __linux__
    assert( !isnan(y) ); // bk010122 - FPE?
#endif
#endif
    return y;
}

float InvSqrt (float x){
    float xhalf = 0.5f*x;
    int i = *(int*)&x;
    i = 0x5f3759df - (i>>1);
    x = *(float*)&i;
    x = x*(1.5f - xhalf*x*x);
    return x;
}
```

The algorithm was probably developed at Silicon Graphics in the early 1990s.

evil floating point bit level hacking

what the fuck?

1st iteration

2nd iteration, this can be removed

Zxyzxyzxyz xyzx yzxy zxyzxy Zxyzxyzxyzxyzxyz xyz xyzxy zxyzx Yzxyzxyzxyzxy-zxyzxyzxyz xyz xyz Xyzxyzxyzxyzxy zxy Zxyzxyzxyzxyzxyz xyz xyzxyzxyzxyzxyzxyz Xyzxyzx yzxyzx.

7.7 Algorithmen

`algorithm2e`-Package Zxyzx yzx yzx Yzxyzxyzxy zxyzxy Zxyzxyzxy zxyzxyzxyzxyz.

Algorithmus 7.1 : How to write algorithms.

Daten : this text
Ergebnis : how to write algorithm with LaTeX2e
initialization;
solange *not at end of this document* **tue**
  read current;
  wenn *understand* **dann**
    go to next section;
    current section becomes this one;
  sonst
    go back to the beginning of current section;
  Ende
Ende

Xyzxyzxyz xyz xyzxyzxyzxy Zxyzxyzxyzx yzxyzxyz xyz XyzxyzXyzxyzxy. Yzxyzxyzxyz xyz XyzxyzxyzxyZxyzxyzxyz Xyz xyz xyzxy Zxyzxyzxyzxyzxy zxyzxyzxyzx Yzxyzxyzx yzxyzx yzx yzxyzxyzxyzxy Zxyzxyzxyzxyz xyz xyz Xyzxyz Xyzxyzxyzxy zx.

Algorithmus 7.2 : disjoint decomposition

input : A bitmap Im of size $w \times l$
output : A partition of the bitmap

special treatment of the first line;
für $i \leftarrow 2$ **bis** l **tue**
  special treatment of the first element of line i;
  für $j \leftarrow 2$ **bis** w **tue**
    left ← `FindCompress(`$Im[i, j-1]$`);`
    up ← `FindCompress(`$Im[i-1,]$`);`
    this ← `FindCompress(`$Im[i, j]$`);`
    wenn left *compatible with* this **dann** `// O(left,this)==1`
      wenn left $<$ this **dann** `Union(left,this);`
      sonst `Union(this,left);`
    Ende
    wenn up *compatible with* this **dann** `// O(up,this)==1`
      wenn up $<$ this **dann** `Union(up,this);`
      `// this is put under up to keep tree as flat as possible`
      sonst `Union(this,up);`
      `// this linked to up`
    Ende
  Ende
  für jedes *element e of the line i* **tue** `FindCompress(`p`);`
Ende

7.8 Tabellen

Xyzx yzxyzxy zxyz xyzxyz xyz xyzxyzxyzxy. Zxyzx yzxy Zxyzxyzxyzxyzxy zxyzx yzxyz xyz xy zxyzxyzxyzxyz Xyzxyzxyzxyzxyz (xyzxyzxyzxyzxy Zxyzxyzxyz- xyz xyzxyzxy-zxyzxy Zxyzxyzxyzxyzxyzxy).

Tabelle 7.1: Xyzxyzxyz Xyzxyzxy zxy Zxyzxyz Xyzxyzxyz: Xyzxyzxyzxyz Xyzxyzxyzxy zxy Zxyzxy-Zxyzxy (Zxyzxyzx yzx YzxyzxyzXyzxyzx) yzxyz xyzxyzxyzxyzxy Zxyzxyzxyzxy (0x0201, 0x0202, 0x030D zxy 0x031A) Zxyzxyz xyz XyzxyzxYzxyzxyzxy Zxy zx yzxyzxyzxyzxy Zxyzxyzxyzxyzxy zxyzxy zxy zxyzxyzxyzxyz Xyzxyzxyzxyzx yzx yzx Yzxyzx Yzxyzxy zx.

Abcabc	Abc	Abca	Bcabcabcabcabc
Cabca[15]	$UUID_{1/16-Bit}$[16]	0x180A[17]	*Abcab*
Bcabc	ABCA	Abcabcabc	Abcab/Cabcabcabc
Abcabcab	ABCA		*Abcab/Cabcabcabcabc*
cabcabcab	ABCA	42,24	Cabcabcab Cabcabcabcabca bcabca bca Bcabcabcabcabcabc Abcabcab; cab CabcabCabcabca bcabcab cab cabc Abcabcab, cabca bc abcabcab cabca BcabcabcAbcabc abc abc AbcabcabcabCabcabcabc abcab cab Cabcabca bca Bcabcab CabcabcabcaBcabcabcab cab Cabcabcabca bcabcabcab Cabcabcabc Abcabcabcab cab Cabcabc Ab cabcabca Bcabcabcabca bc abc abca bcabcabcabcab Cabcabcabca bca bcabcabcabcabc Abcabcabcabca (BcabcabcaBcabcabcab, CabcAbcab cabca bcabca bcabcabcabc AbcabCabcabcabc abc AbcabcAbcabcab) cabcabca bca Bcabcabcabcabcabc ab cabc abcabcabcabc Abcabcabc

Zxyzxyz xyzx yz xyzxyzxyz Xyzxyzxyzxyzxyzxyzxyzxyz – xyz Xyzxyzxy zxyzxyz xyz Xyzx, yzxy zxy zx Yzxyzxyzxyzxyzxyzxyz xyzxyzxyzxy Zxyzxyz (Xyzxyzx) yzx yzxy-zxyzxyzx yzxyzxyzxyzxy Zxyzxyzxy (Zxyzxy) zxy zxyzxy zxyzxyzxyzxy Zxyz xyz xyzx Yzxyz xyzxyzx, yzxyzxy zxy Zxyzx yzx yzxyzxyzxyzxyzxyz Xyzxyzxyzxyzxy zxyzxy-zxyzxy zx yzxyzxyzxyzxyz xyz. Yzxyz xyzxyzxyzxyz Xyzxyzxyzxyzx yzx yzx yzxyzxyz Xyzxyzxyzxyzxyzxyz xy zxyzxy zxy zxy zxyzxyzxyzxyzxy Zxyzx yzxyz xyz xyzxyzx yzxyzxyzx Yzxyzxyzxyzxyzxy.

[15] Abcab cabca bca bca Bcabcabc
[16] Abcab cab cabc Abcabcabcabc Abcab
[17] Cabca bcabcabca bcabc Abcabc

7.9 Gleichungen

Yzx Yzxyzxyzxyzxyzxyzxy zxyzx yzxyz xyzx yzxyzxyzx Yzxyzxyz xyz. Bcabcabcabca bca bca bca x–y-Bcabcabca Bca $x^2 + y^2 = 1$. Xyz xyz xyz xyz xyzxyzxyzxyzxy.

$$\operatorname{var}\widehat{\Delta} = \sum_{j=1}^{t} \sum_{k=j+1}^{t} \operatorname{var}\left(\hat{\alpha}_j - \hat{\alpha}_k\right) = \sum_{j=1}^{t} \sum_{k=j+1}^{t} \sigma^2(1/n_j + 1/n_k). \tag{7.1}$$

Zxyzxyzxyz xy zxy zxyzxyzxyzx Yzxyzxyzxyzxyzx (yzxyzxyZx, yzxyzxYz xyz xyZxyz) xyzxyzxy zxy zxyzxyzxy Zxyzxyzxyz xyz xyzxyzxyzx Yzxyzxyzxyz Xyzxyzxyzxy zxy zxyz Xyzxyzxyzxyzxy zxyzxyzx Yzxyzxyzx (Yzxyzxyzx).

$$\frac{d}{dx} \arctan(\sin(x^2)) = -2\,\frac{\cos(x^2)x}{-2 + (\cos(x^2))^2}$$

Xyzxyz xyz xyz Xyzxy zxy $A1$, $A2$, …, Aa. Xyzx Yzxyzxyz xyz xyzxy zxyzxyzx Yzxyzxyzxyzxyz xyzxyz xyzx.

$$\left.\begin{aligned} B' &= -\partial \times E, \\ E' &= \partial \times B - 4\pi j, \end{aligned}\right\} \quad \text{Maxwell's equations} \tag{7.2}$$

Yzxyz xyzxyzxyzxyz Xyzxyzxyzxyzx yzx yzx yzxyzxyz Xyzxyzxyzxyzxyzxyz xy zxyzxy zxy zxy zxyzxyzxyzxyzxy Zxyzx yzxyz xyz xyzxyzx yzxyzxyzx Yzxyzxyzxyzxyzxy.

7.10 Definitionen & Hypothesen

Zxyzxyzxyzxyz xyzxyz xy zxyzxyzxy Zxyz (Xyzxyzxy ZX) yzx yzxy zx yzxyz Xyzxyzxyz Xyzxyzxy zxy Zxyzxyz xyzxyzxyzxy Zxyzxy (Zxyzxyzxy ZX) yzxyzxy (zxyzx Yzxyzxy).

Definition 1 *Let f be a function whose derivative exists in every point, then f is a continuous function.*

Xyzxyzx, yzxyzx yzx yzxyzxyzxyz Xyzxyzxyzxy zxyzxyzxyz Xyzxyzxyzx yz xy Zxyzx

Definition 2 (Pythagorean theorem) *This is a theorema about right triangles and can be summarised in the next equation*

$$x^2 + y^2 = z^2$$

Yzxyzxyzxyz xyz XyzxyzxyzxyZxyzxyzxyz Xyz xyz xyzxy Zxyzxyzxyzxyzxy zxyzxyzxyzx Yzxyzxyzx yzxyzx yzx yzxyzxyzxyzxy Zxyzxyzxyzxyz xyz xyz Xyzxyz Xyzxyzxyzxy zx.

Hypothese 1 *The greater the service orientation, the greater the level of employee outcomes (i.e. organizational commitment, esprit de corps, and job satisfaction).*

Hypothese 2 (Business Performance) *The greater the service orientation, the better the business performance (i.e. ROA, new accounts opened, and service quality image)*

Yzx Yzxyzxyzxyz (XyzxyZxyzxyz Xyzxyzxyzxy), zxy Zxyzxyzxyzxyzxyzxyz (XyzxyzXyzxyzxyzxyzxyzxyz) xyz xyz Xyzxyzxyzxyzxy (ZxyzXyzxyzxyzxyZxyzxyzxyzx) yzxy zxyz xyzxy.

7.11 To-Do-Notes

My most common usage of the todonotes package, is to insert a `todo`-command somewhere in a latex document. An example of this usage is the command `\todo{Make a cake}`, which renders like .

Make a cake

It is possible to place a todonote inside the text instead of placing it in the margin, this could be desirable if the text in the note has a considerable length. `\todo[inline]{A todonote placed in the text}`

A todonote placed in the text

The `\listoftodos`-command inserts a list of all the todos in the current document.

Literaturverzeichnis

[3] *Architecture Overview - Convolutional Neural Networks (CNNs / ConvNets) - CS231n Convolutional Neural Networks for Visual Recognition*. URL: https://cs231n.github.io/convolutional-networks/ (besucht am 14. Aug. 2021).

[4] Antonio Bergua. „Visueller Kortex“. In: *Das menschliche Auge in Zahlen*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, Seiten 147–150. ISBN: 978-3-662-47284-2. DOI: 10.1007/978-3-662-47284-2_27. URL: https://doi.org/10.1007/978-3-662-47284-2_27.

[5] Valeriu Codreanu, Damian Podareanu und Vikram Saletore. *Scale out for large minibatch SGD: Residual network training on ImageNet-1K with improved accuracy and reduced time to train*. 2017. arXiv: 1711.04291 [stat.ML].

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit und Neil Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[8] Jean-Christophe Filliâtre und Sylvain Conchon. „Type-safe Modular Hash-consing“. In: *Proceedings of the 2006 Workshop on ML*. ML '06. Portland, Oregon, USA: ACM, 2006, Seiten 12–19. ISBN: 1-59593-483-9. DOI: 10.1145/1159876.1159880. URL: http://doi.acm.org/10.1145/1159876.1159880.

[9] Kunihiko Fukushima. „Neocognitron : A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position“. In: *Biological Cybernetics* 36 (1980), Seiten 193–202. DOI: 10.1007/BF00344251. URL: https://link.springer.com/article/10.1007%2FBF00344251.

[10] Kunihiko Fukushima. „Neocognitron: A hierarchical neural network capable of visual pattern recognition“. In: *Neural Networks* 1.2 (1988), Seiten 119–130. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(88)90014-7. URL: https://www.sciencedirect.com/science/article/pii/0893608088900147.

[11] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd. O'Reilly Media, Inc., 2019. ISBN: 1492032646.

[12] Ian Goodfellow, Yoshua Bengio und Aaron Courville. „Back-Propagation and Other Differentiation Algorithms“. In: *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016, 200–217. ISBN: 0262035618.

[13] Ricardo Guerrero. *Training Deep Learning Models On multi-GPus - BBVA Next Technologies*. URL: https://www.bbvanexttechnologies.com/blogs/training-deep-learning-models-on-multi-gpus/ (besucht am 8. Aug. 2021).

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[15] David H. Hubel. „Single unit activity in striate cortex of unrestrained cats“. In: *The Journal of physiology* 147 (1959), 226—238. ISSN: 0022-3751. DOI: 10.1113/jphysiol.1959.sp006238. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/pmid/14403678/?tool=EBI.

[16] David H. Hubel und Torsten N. Wiesel. „Receptive fields, binocular interaction and functional architecture in the cat's visual cortex“. In: *The Journal of physiology* 160 (1962), 106—154. DOI: 10.1113/jphysiol.1962.sp006837. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/?tool=EBI.

[17] David H. Hubel und Torsten N. Wiesel. „Receptive fields of single neurones in the cat's striate cortex“. In: *The Journal of physiology* 148 (1959), 574—591. ISSN: 0022-3751. DOI: 10.1113/jphysiol.1959.sp006308. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/pmid/14403679/?tool=EBI.

[18] *ImageNet Benchmark (Image Classification) | Papers With Code*. URL: https://paperswithcode.com/sota/image-classification-on-imagenet (besucht am 7. Aug. 2021).

[19] Alex Krizhevsky, Ilya Sutskever und Geoffrey E. Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Commun. ACM* 60.6 (Mai 2017), 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[20] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard und L. D. Jackel. „Backpropagation Applied to Handwritten Zip Code Recognition“. In: *Neural Computation* 1.4 (1989), Seiten 541–551. DOI: 10.1162/neco.1989.1.4.541.

[21] Y. Lecun, L. Bottou, Y. Bengio und P. Haffner. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), Seiten 2278–2324. DOI: 10.1109/5.726791.

[23] *Principles of Deformation-Resistant Recognition - Neocognitron - Scholarpedia*. URL: http://www.scholarpedia.org/article/Neocognitron (besucht am 13. Aug. 2021).

[24] Chris Richardson. *Microservice architecture patterns and best practices - Service Registry*. 2014. URL: http://microservices.io/patterns/service-registry.html (besucht am 3. Nov. 2015).

[25] D. E. Rumelhart, G. E. Hinton und R. J. Williams. „Learning Internal Representations by Error Propagation“. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA, USA: MIT Press, 1986, 318–362. ISBN: 026268053X.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg und Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV].

[27] Florian Schroff, Dmitry Kalenichenko und James Philbin. „FaceNet: A unified embedding for face recognition and clustering“. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[28] Zhong Shao, John H. Reppy und Andrew W. Appel. „Unrolling lists“. In: *SIGPLAN Lisp Pointers* VII.3 (Juli 1994), Seiten 185–195. ISSN: 1045-3563. DOI: 10.1145/182590.182453. URL: http://doi.acm.org/10.1145/182590.182453.

[29] *Stacking Multiple Feature Maps - Deep Computer Vision Using Convolutional Neural Networks - Programmer Sought*. URL: https://www.programmersought.com/article/65985444807/ (besucht am 15. Aug. 2021).

[30] *Stanford DAWN Deep Learning Benchmark (DAWNBench) · ImageNet Training*. URL: https://dawn.cs.stanford.edu/benchmark/ImageNet/train.html (besucht am 8. Aug. 2021).

[31] Mingxing Tan und Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[32] *The Architecture of the Visual Cortex - Deep Computer Vision Using Convolutional Neural Networks - Programmer Sought*. URL: https://www.programmersought.com/article/65985444807/ (besucht am 12. Aug. 2021).

[33] *Types of Deep Learning Layers - What is Deep Learning? - Databricks*. URL: https://databricks.com/de/glossary/deep-learning (besucht am 12. Aug. 2021).

[35] Patrick Zschech, Christoph Sager, Philipp Siebers und Maik Pertermann. „Mit Computer Vision zur automatisierten Qualitätssicherung in der industriellen Fertigung: Eine Fallstudie zur Klassifizierung von Fehlern in Solarzellen mittels Elektrolumineszenz-Bildern“. In: *HMD Praxis der Wirtschaftsinformatik* 58.2 (2021), Seiten 321–342. ISSN: 2198-2775. DOI: 10.1365/s40702-020-00641-8. URL: https://doi.org/10.1365/s40702-020-00641-8.

A Anhang

Eins (ohne extra Eintrag im Inhaltsverzeichnis)

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.

Zwei (ohne extra Eintrag im Inhaltsverzeichnis)

Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Drei (ohne extra Eintrag im Inhaltsverzeichnis)

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Vier (ohne extra Eintrag im Inhaltsverzeichnis)

Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Eidesstattliche Erklärung

Hiermit versichere ich, dass meine Bachelorarbeit „Deep Learning
zur visuellen Erkennung von Tierarten“ („Deep learning for visual recognition of animal species“) selbständig verfasst wurde und dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt wurden. Diese Aussage trifft auch für alle Implementierungen und Dokumentationen im Rahmen dieses Projektes zu.

Potsdam, den 18. August 2021,

_____

(Minh Kien Nguyen)