



Karlsruhe Institute of Technology
Fakultät für Elektrotechnik



Institut für Technik der
Informationsverarbeitung (ITIV)

Bewertung und Vergleich verschiedener Arten faltender künstlicher neuronaler Netze zur Regression der Picklänge kardiovaskulärer Implantate basierend auf Kamerabildern und Deep Learning

Bachelorarbeit von

cand. el. Emilio Rivera Cedeño

30. April 2021

Institutsleitung: Prof. Dr.-Ing. Dr. h.c. J. Becker
Prof. Dr.-Ing. Eric Sax
Prof. Dr. rer. nat W. Stork

Betreuer: M. Sc. Benedikt Haas

Zusammenfassung

Die Sicherung der Qualität bei Stents ist von großer Bedeutung sowohl für den Hersteller, als auch für den Patienten. Hierbei ist der Vorgang der visuellen Inspektion aufwendig und fehleranfällig, weshalb der Vorgang automatisiert werden soll, um die Reproduzierbarkeit der Prüfung zu verbessern und ein genaueres Prüfergebnis zu erzielen. Als grundlegende Qualitätsmerkmale gelten in der Geometrie von Stents der Flechtwinkel und die Picklänge.

In der vorliegenden Arbeit wurden unterschiedliche faltende neuronale Netze implementiert und trainiert, um eine Regression der betrachteten Picklänge im Stent durchzuführen. Diese wurden darauffolgend getestet, sodass die Ergebnisse bewertet und verglichen werden konnten. Hierfür war die Erstellung eines Datensatzes, dessen Erweiterung durch Augmentation und die Vorverarbeitung der Bilddaten notwendig.

Durch die Evaluation konnte ein neuronales Netz gewonnen werden, das in der Lage ist, die Länge des mittleren Picks eines Bildes mit einer Genauigkeit von 1,764 Pixel oder 0,077 Millimeter innerhalb von 18,9 Millisekunden zu bestimmen. Im Anschluss der Arbeit ist es möglich, eine vereinfachte Form der Fehlerlokalisierung durchzuführen.

Urheberrecht

ARM®, AMBA®, AXI™, Cortex™, TrustZone™, SecurCore™, DSTREAM™ und weitere im Text erwähnte ARM-Produkte sowie die entsprechenden Logos sind Marken oder eingetragene Marken der Advanced RISC Machines Ltd.

Xilinx®, Zynq™ und weitere im Text erwähnte Xilinx-Produkte sowie die entsprechenden Logos sind Marken oder eingetragene Marken der Xilinx Inc.

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit (bzw. Masterarbeit) selbstständig und unter Beachtung der Regeln zur Sicherung guter wissenschaftlicher Praxis im Karlsruher Institut für Technologie (KIT) in der aktuellen Fassung angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und wörtlich oder inhaltlich übernommene Stellen als solche kenntlich gemacht.

Karlsruhe, den 30. April 2021

Emilio Rivera Cedeño

Inhaltsverzeichnis

1 Einleitung	1
1.1 Zielsetzung	1
1.2 Aufbau der Arbeit	2
2 Grundlagen	3
2.1 Stents	3
2.1.1 Anwendungsgebiete	4
2.1.2 Herstellungsverfahren	4
2.1.3 Picklänge	5
2.2 Histogrammausgleich	5
2.2.1 Adaptiver Histogrammausgleich	6
2.2.2 Kontrastbegrenzter adaptiver Histogrammausgleich	6
2.3 Deep Learning	7
2.3.1 Training eines neuronalen Netzes	8
2.3.2 Faltende neuronale Netze	13
2.4 Netzwerkarchitekturen	20
2.4.1 AlexNet	21
2.4.2 VGG	21
2.4.3 ResNet	22
2.4.4 Inception-v4	23
2.4.5 Inception-ResNet-v2	23
2.4.6 Xception	24
2.4.7 MobileNetV3	24
2.4.8 EfficientNet	24
3 Verwandte Arbeiten	27
3.1 Bildbasierte Analyse von Geflechten	27
3.2 Visuelle Inspektion von Stents	28
3.3 Stents4Tomorrow	29
4 Konzept	31
4.1 Aufbau des Projekts	31
4.2 Berechnung der Picklänge	32
4.3 Auswahl der faltenden neuronalen Netzen	32
4.4 Erfassung der Trainingsdaten	33

5 Implementierung	35
5.1 Trainingsdaten	35
5.2 Aufbereitung der Trainingsdaten	36
5.2.1 Bildvorverarbeitung	37
5.2.2 Aufteilen und Normalisieren der Daten	38
5.3 Implementierung der faltenden neuronalen Netze	39
5.4 Reproduzierbarkeit	40
5.5 Fehlerlokalisierung	40
6 Bewertung und Vergleich der Ergebnisse	41
6.1 Training	41
6.2 Bewertung und Vergleich der Ergebnisse	43
6.2.1 Fehler	43
6.2.2 Rechenzeit pro Bild	43
6.2.3 Trainingsdauer	45
6.2.4 Fehler zu Rechenzeit	45
6.3 Bestimmung der Abweichung	45
6.4 Fehlererkennung	47
6.5 Mögliche Fehlerquellen	47
6.6 Bewertung des Systems	48
7 Fazit und Ausblick	51
A Anhang	63

Kapitel 1

Einleitung

Nach Angaben des Statistischen Bundesamtes starben in Deutschland im Jahre 2019 195.690 Menschen an einer Herzerkrankung [Stab]. Dies entspricht etwa ein Fünftel der Sterbefälle in diesem Jahr [Staa]. Um das Risiko eines Herzinfarktes zu reduzieren, werden seit 1988 Stents eingesetzt [Sig17].

Stents sind kardiovaskuläre Implantate, welche in Gefäße eingefügt werden, um eine Verstopfung bzw. ein Zusammenfallen derselben zu verhindern. Aus diesem Grund muss der Stent gewisse Qualitätsanforderungen erfüllen wie zum Beispiel die geometrischen Verhältnisse dieser Spiralprothesen. Diese werden aktuell stichprobenartig nach der Herstellung durch einen Menschen mittels visueller Inspektion geprüft. Allerdings ist dieser Vorgang zeitaufwendig, für den Inspizierenden anstrengend und daher fehleranfällig. Zudem ist dieser Ablauf in der Produktion ineffizient, weil der Stent erst nach seiner Fertigung geprüft werden kann.

Um jeden Stent überprüfen zu können und den möglichen menschlichen Fehler zu beseitigen, soll der Prozess der geometrischen Inspektion automatisiert werden. Eine zur Herstellung parallel geführte Überprüfung nach Defekten würde außerdem das sofortige Ausgleichen von Ungenauigkeiten erlauben. Zudem bietet dieser Ansatz die Möglichkeit, alle Stents zu prüfen. Dementsprechend soll, basierend auf Kamerabildern und Deep Learning, ein effektives System zur Fehlererkennung in Stents entwickelt werden.

1.1 Zielsetzung

Ziel dieser Arbeit ist die Überprüfung der Stentgeometrie anhand der Picklängen auf erfassten Kamerabildern. Hierbei soll immer derjenige Pick analysiert werden, der sich in der Bildmitte befindet. Im Spezifischen sollen verschiedene Arten faltender neuronaler Netze evaluiert und anschließend miteinander verglichen werden. Dies schließt eine ge-

eignete Bildvorverarbeitung sowie die Definition geeigneter Vergleichsparameter mit ein. Abschließend soll eine Lokalisierung des Fehlers im Bild möglich sein.

1.2 Aufbau der Arbeit

Aus diesem Grund setzt sich Kapitel 2 mit den für das Verständnis dieser Arbeit notwendigen Grundlagen auseinander. Dazu zählt ein Überblick über Stents, der Histogrammausgleich und der Aufbau und Funktionsweise faltender neuronaler Netze. Darauffolgend wird der aktuelle Stand von Methoden zur Defekterkennung in Stents und Geflechten in Abschnitt 3 vorgestellt. Des Weiteren wird der grundlegende Aufbau des Projektes Stents4Tomorrow [Pro] in Kapitel 4 dargestellt und die im Rahmen dieser Arbeit relevanten Schritte hervorgehoben. Zudem wird die Berechnung der Picklänge erläutert und die Auswahl der faltenden neuronalen Netze erklärt. Darüber hinaus werden die einzelnen Schritte der Datenvorverarbeitung und Implementierung der neuronalen Netze in Kapitel 5 beschrieben und begründet. Im Übrigen werden die erworbenen Ergebnisse in Abschnitt 6 bewertet und miteinander verglichen. Hierfür werden diese vorgestellt und ausgewertet, um die faltenden neuronalen Netze anhand festgelegter Vergleichsfaktoren zu vergleichen. Anschließend werden die Ergebnisse in Kapitel 7 zusammengefasst und die nächsten Schritte im Sinne eines Ausblicks erläutert.

Kapitel 2

Grundlagen

Dieses Kapitel stellt die grundlegenden Konzepte über Stents und faltende neuronale Netze vor, die für ein Verständnis dieser Arbeit benötigt werden. Der erste Unterabschnitt beginnt mit einer Einführung zu Stents und deren Einsatzgebiete in der Medizin. Des Weiteren wird das Herstellungsverfahren und die Geometrie des Aufbaus eines Stents erläutert.

Das Kapitel beginnt mit einen kurzen Einblick in einige Verfahren des Histogrammausgleichs aus der Bildverarbeitung und wird mit den Grundlagen zu faltenden neuronalen Netzen fortgeführt. Hierbei wird sowohl in die Bausteine, als auch in die daraus bestehenden Netzwerkarchitekturen eingegangen. Zudem wird der Vorgang des Trainings eines neuronalen Netzes konkretisiert.

2.1 Stents

Am 12. Juni 1986 veränderten Ulrich Sigwart und Jacques Puel, mit der ersten Anwendung eines Stents bei Menschen, die Behandlung der ischämischen Herzkrankheit. Es handelte sich um ein selbstexpandierendes geflochtenes Gittergewebe aus 16 Drähten mit einem Durchmesser von $60 \mu\text{m}$. Wie in Abbildung 2.1 zu sehen ist, war der Stent an einen Abgabekatheter durch eine einziehbare Plastikmembran gebunden. [Sig17]

Diese Art von Prothesen dienen als Instrument zum Offenhalten von Gefäßen oder Hohlorganen. Meistens bestehen diese aus Kunststoffen oder Metallen. Stents stabilisieren verengte Gefäße durch das Erweitern dieser und vermeiden einen erneuten Verschluss oder eine Verengung. Die Oberfläche des Gefäßes wird geglättet und Ablagerungen werden vom Stent gegen die Innenwand gepresst. Dies führt zu einer Verbesserung des Blutflusses im betroffenen Bereich. [Mei]

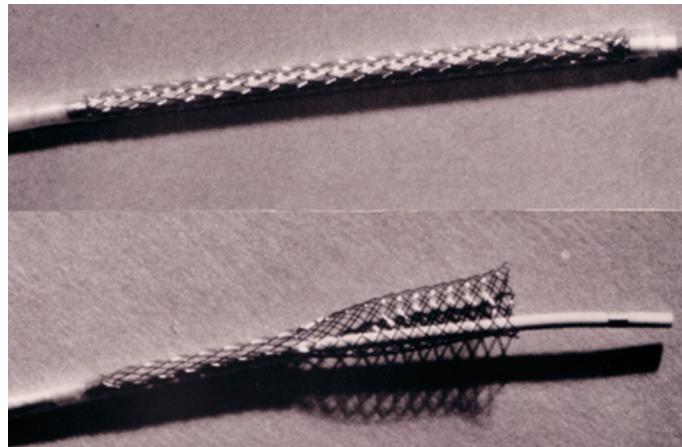


Abbildung 2.1: Der erste selbstexpandierbare Stent. Oben am Abgabekatheter gebunden. Unten zum Teil bereitgestellt durch Zurückziehen der Membran [Sig17].

2.1.1 Anwendungsgebiete

Die Anwendung von Stents befindet sich bei der Aufdehnung verschlossener Gefäße oder Hohlorgane. Meist werden sie in den Koronararterien eingebaut, um diese zu stabilisieren. Auch Halsschlagadern, hirnversorgende Arterien, die Hauptschlagader sowie andere periphere Blutgefäße gehören zu den Einsatzgebieten von Stents. [Mei]

Zudem wird diese Kategorie von Prothesen auch in weiteren Bereichen eingesetzt. In der Speiseröhre und im Dickdarm werden Stents bei fortgeschrittenem Krebs verwendet. Ebenso finden sie im Harnleiter, in der Prostata, im Gallengang und seit wenigen Jahren auch in den Augen diverse Anwendung.

2.1.2 Herstellungsverfahren

Eine schnelle Methode zur Produktion von Stents ist das Schneiden mit Lasern. Die thermische Energie des Lasers wird absorbiert, wodurch das Werkstück an definierten Stellen aufgewärmt und die zu entfernenden Stellen in einen geschmolzenen, verdampfen oder chemisch veränderten Status transformiert werden. Anhand der Strömung eines Hochdruckgases können diese Anteile entfernt werden [GC18]. Da diese Methode für Strukturen eines kleineren Durchmessers ungeeignet ist, werden im Rahmen dieser Arbeit ausschließlich Stents untersucht, die durch einen Flechtprozess gefertigt werden. Eine Maschine zum Flechten von Stents ist in Abbildung 2.2 zu sehen.

Auf einer ringförmigen Struktur werden mehrere Klöppel montiert, die jeweils eine Spule mit Flechtmaterial tragen. Beim Flechtvorgang dreht sich diese Struktur, sodass die Drähte um die Mitte gewickelt werden. Zudem drehen sich nebeneinanderstehende Klöppel in entgegengesetzte Richtungen, um die Gitterstruktur der Stents zu erzielen. [Kyo14]

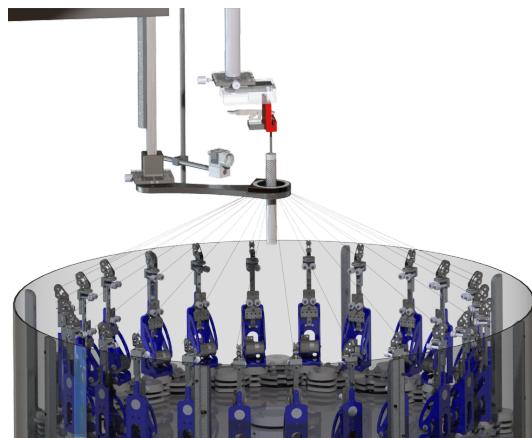


Abbildung 2.2: Aufbau einer Flechtmaschine, entnommen aus [Pro]

2.1.3 Picklänge

Ein Stent kann als Gitter bestehend aus Rautenstrukturen aufgefasst werden. Eine solche Struktur, welche aus den Überkreuzungspunkten des Drahtes gebildet wird, wird als Pick bezeichnet. Die Picklänge entspricht in diesem Fall dem Abstand zwischen der oberen und unteren Ecke eines Picks. Diese Länge ist in Abbildung 2.3 dargestellt.

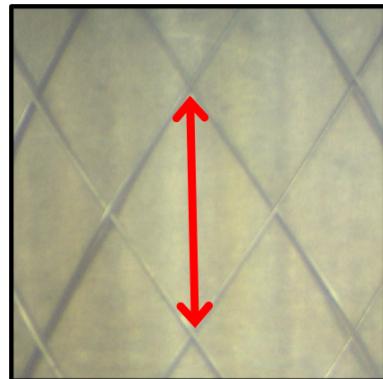


Abbildung 2.3: Veranschaulichung eines einzelnen Picks. Picklänge in rot visualisiert. Entnommen aus [Pro].

2.2 Histogrammausgleich

Der Histogrammausgleich ist ein Verfahren der Bildverarbeitung zur Kontrastverbesserung. Ziel ist es, ein Bild zu erhalten, bei dem die Bildwerte gleich verteilt sind [BLF16]. Ein Beispiel hierfür ist in Abbildung 2.4 dargestellt.

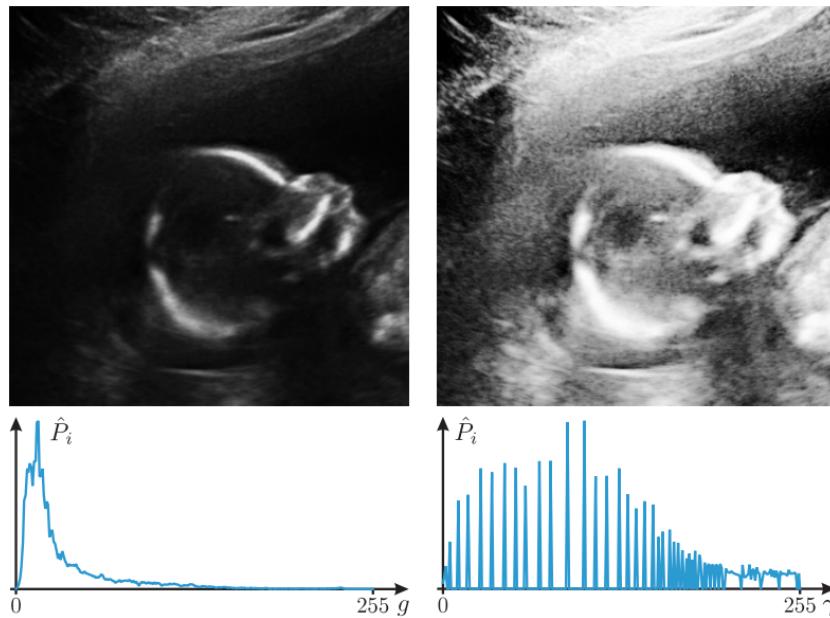


Abbildung 2.4: Beispiel eines Histogrammausgleichs: links, ursprüngliches Bild; rechts, Ergebnis des Histogrammausgleichs. Darunter sind die zugehörigen Histogramme dargestellt. Entnommen aus [BLF16].

2.2.1 Adaptiver Histogrammausgleich

Beim adaptiven Histogrammausgleich (engl. Adaptive Histogram Equalization) oder AHE wird diese Methode verbessert, sodass auch bei Bildern mit deutlich helleren oder dunkleren Regionen gute Ergebnisse erzielt werden können. Es werden mehrere Histogramme berechnet, um die Beleuchtungswerte im Bild neu zu verteilen. Jedes Pixel ändert sich basierend auf dem Histogramm der Region, von der es umgeben wird. Für die Anwendung an der am Rand liegenden Pixel wird ein sogenanntes Padding angewendet. In diesem Fall wird das Bild um die am Rand gespiegelten Werte erweitert. [PAA⁺87]

2.2.2 Kontrastbegrenzter adaptiver Histogrammausgleich

Der kontrastbegrenzte adaptive Histogrammausgleich (engl. Contrast-Limited Adaptive Histogram Equalization (CLAHE)) ist eine Modifikation vom AHE. Abbildung 2.5 zeigt ein Beispiel für diese Methode. Hierbei wird die Kontrastverstärkung als die Ableitung der Transformationsfunktion, die Ein- und Ausgangsintensität verbindet, definiert. Der Ableitungswert eins entspricht keiner Verstärkung und höhere Werte führen zu höheren Verstärkungen. Um die Verstärkung zu begrenzen, wird das Histogramm an einer vordefinierten Höhe beschnitten. Somit wird die Steigung der Transformationsfunktion limitiert. Der Punkt, ab dem das Histogramm beschnitten wird, wird durch den Grenzwert der Steigung für die Funktion bestimmt. [PAA⁺87]



Abbildung 2.5: Beispiel für die Anwendung von CLAHE: links, Originalbild mit Nebel; rechts, Bild nach der Anwendung von CLAHE. Entnommen aus [YMA14].

2.3 Deep Learning

Laut dem Wörterbuch der University of Cambridge ist Deep Learning eine Art künstlicher Intelligenz, die Algorithmen verwendet, welche auf der Funktionsweise des menschlichen Gehirns basieren [Cam]. Diese neuronale Perspektive wird durch zwei Hauptideen vorangetrieben. Eine davon ist das Konzept der Nachkonstruktion der rechnerischen Prinzipien hinter dem Gehirn und die Duplizierung der Funktionalität dessen, um Intelligenz zu erschaffen. Zum anderen sollen Modelle des Deep Learnings nicht nur technischen Anwendungen dienen, sondern auch zu einem besseren Verstand über das Zerebrum und den Prinzipien, die der menschlichen Intelligenz zugrunde liegen, beitragen. [GBCB16]

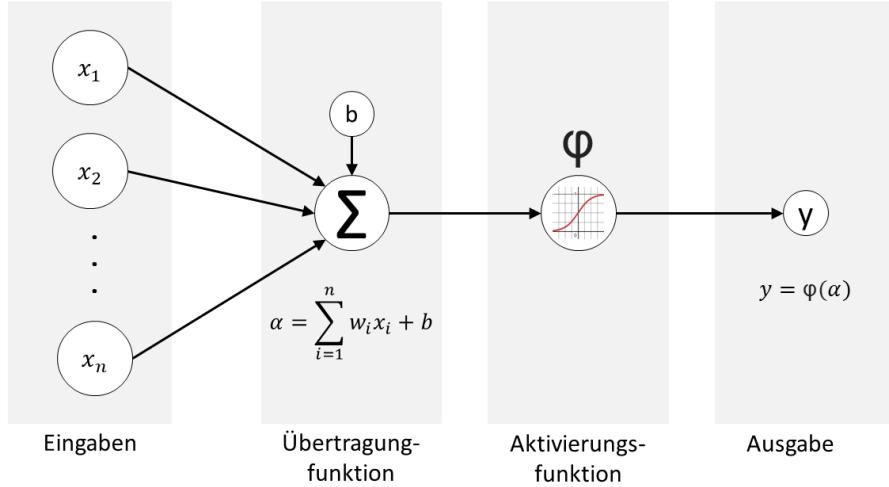


Abbildung 2.6: Schematischer Aufbau eines künstlichen neuronalen Netzes, das als einschichtiges Perzepron-Netz modelliert wird. In Anlehnung an [DN20].

Diese Modelle werden mithilfe künstlicher Neuronen gebildet, die auf mehreren Ebenen miteinander verbunden sind. Hierbei wird jede Nervenzelle als sogenanntes Perzepron (siehe Abbildung 2.6) modelliert, bei dem die Dendriten des biologischen Neurons durch eine Schicht mit Eingaben gestaltet werden. Das Perikaryon wird durch die Übertragungs-

funktion, das Axon durch die Ausgabe und die inhibitorischen bzw. exzitatorischen Eigenarten der Synapse werden durch die Gewichte repräsentiert. Zur erwähnten Eingabe gehört auch ein Bias, welches zur Summe aller Eingaben addiert wird, bevor die Gleichung 2.1 in die Aktivierungsfunktion eingegeben wird. Diese bestimmt den Schwellenwert, bei dem die Nervenzelle aktiviert wird [DN20]. Somit war das von Frank Rosenblatt entworfene Perzepron das erste Modell, welches in der Lage war, anhand von Beispielen die Gewichte der einzelnen Kategorien zu definieren [Ros57].

$$\alpha = \sum_{i=1}^n w_i x_i + b \quad (2.1)$$

Mithilfe solcher künstlichen Neuronen kann ein neuronales Netz wie in Abbildung 2.7 aufgebaut werden. Das abgebildete Netzwerk besteht aus drei Schichten: die Eingabeschicht, eine Zwischenschicht und eine Ausgabeschicht. Eine Eingabe wird in das Netz über die zwei Neuronen der ersten Schicht eingespeist und an die Zwischenschicht weitergeleitet. Von dort wird sie an die Ausgabeschicht geleitet, an der die Ausgabe des Netzwerks dargestellt wird.

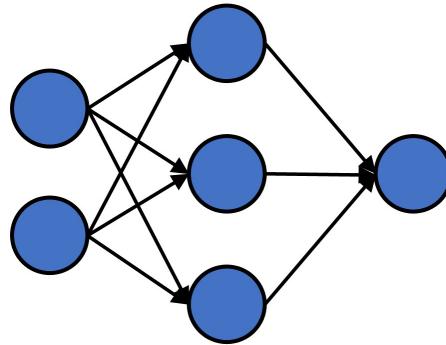


Abbildung 2.7: Aufbau eines neuronalen Netzes mit zwei Neuronen für die Eingabe und einem für die Ausgabe

2.3.1 Training eines neuronalen Netzes

Das Training eines neuronalen Netzes ist der Prozess, bei dem die Parameter so gesetzt werden, dass die Differenz zwischen den Ist- und den Soll-Werten möglichst minimiert wird. Die Ausgabe des Netzes, also der Ist-Wert, wird mit der gewünschten Ausgabe (Soll-Wert), welche als Label bezeichnet wird, verglichen. [YNDT18]

Zu Beginn des Trainings wird ein Stapel (engl. batch) von Bildern vom Eingang zum Ausgang des Netzes vorwärts propagiert. Darauffolgend wird die Leistung des Modells mithilfe einer Fehlerfunktion berechnet. Diese misst die Differenz zwischen den Netzausgaben und den Labels. Im Anschluss werden die erlernbaren Parameter des Netzwerks

anhand des Backpropagation-Algorithmus [RHW85] und dem Gradientenverfahren (siehe Abschnitt 2.3.1.3) für Optimierungsprobleme angepasst. [YNDT18]

Der Ablauf des Trainings eines neuronalen Netzes wird in drei Phasen aufgeteilt, wodurch es drei unterschiedliche Datensätze benötigt: einen Test-, einen Validierungs- und einen Testdatensatz. Die Trainingsdaten werden verwendet, um das Modell zu trainieren. In der Validierungsphase wird bestimmt, wie gut das neuronale Netz trainiert wurde. Zuletzt wird der Testdatensatz für die Bewertung des Generalisierungsfehlers des Modells verwendet. [HTF09]

2.3.1.1 Mittlerer absoluter Fehler

Der mittlere absolute Fehler (engl. Mean Absolute Error, kurz MAE) gibt den Durchschnitt aller Absolutbeträge der einzelnen Abweichungen zwischen der Netzausgabe und dem Label an. Für n Testinstanzen wird dieser wie folgt berechnet:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |(y_i - \lambda(x_i))|, \quad (2.2)$$

wobei y_i und $\lambda(x_i)$ den Soll- und Ist-Wert für x_i darstellen. [SW11]

2.3.1.2 Mittlerer quadratischer Fehler

Beim mittleren quadratischen Fehler (engl. Mean Squared Error, kurz MSE) wird die quadratische Abweichung der Netzausgaben zum Soll-Wert über n Testinstanzen wie folgt bestimmt:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \lambda(x_i))^2. \quad (2.3)$$

Die Bedeutungen für y_i , $\lambda(x_i)$ und x_i sind analog zu 2.3.1.1. [SW11]

Der Unterschied zwischen den zwei vorgestellten Verlustfunktionen wird in Abbildung 2.8 anhand zweier beispielhafter Verläufe aufgezeigt. Der mittlere absolute Fehler steigt linear mit einem zunehmenden Fehler an. Im Gegensatz dazu steigt die Funktion beim MSE exponentiell mit einer Zunahme des Fehlers. Dadurch bestraft der mittlere quadratische Fehler große Abweichungen mehr als kleine.

2.3.1.3 Gradientenverfahren

Das Gradientenverfahren (engl. gradient descent) ist ein Optimierungsalgorithmus für das iterative Anpassen der erlernbaren Parameter eines neuronalen Netzes, um den Verlust des Netzwerks möglichst zu minimieren. Der Gradient der Verlustfunktion gibt die Richtung

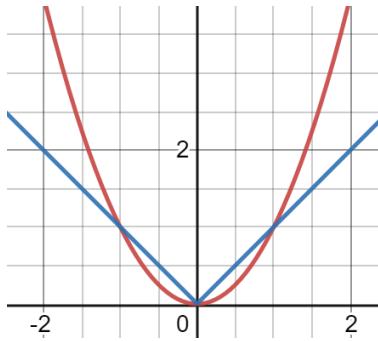


Abbildung 2.8: Beispielhafter Verlauf für den mittleren absoluten Fehler (blau) und den mittleren quadratischen Fehler (rot).

an, in der die negative Steigung der Funktion am größten ist. So werden die Filterkerne und Gewichte in diese Richtung des Gradienten angepasst. Die Schrittgröße wird anhand der Lernrate, ein positiver Skalar, bestimmt. Eine Darstellung dieses Verfahrens wird in Abbildung 2.9 dargestellt. Bei einer zu großen Lernrate kann die Verlustfunktion steigen anstatt zu sinken. Andernfalls wird das Training eines Netzes bei einer zu kleinen Lernrate langsamer und es besteht die Gefahr, bei einem hohen Fehler dauerhaft stecken zu bleiben [GCB16]. [YNDT18]

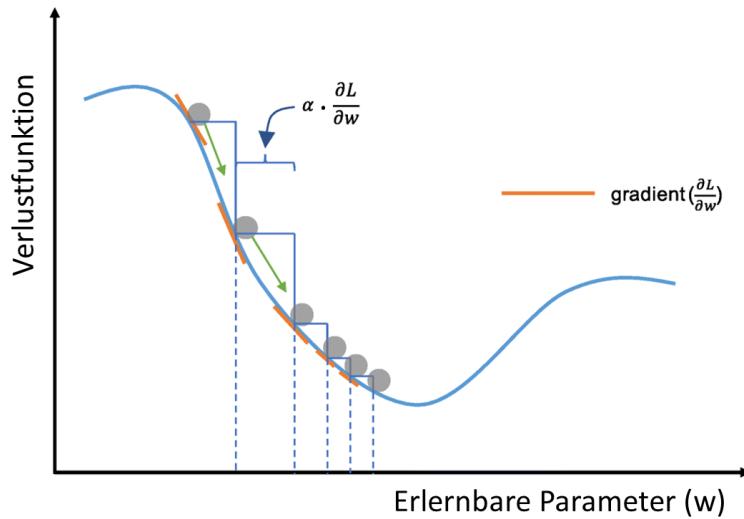


Abbildung 2.9: Verlauf des Gradientenverfahrens mit einer Lernrate α , einer Verlustfunktion L und den Parametern w . Ursprüngliches Bild aus [YNDT18].

Eine solche Aktualisierung der Parameter wird in Gleichung 2.4 formuliert. In diesem Fall steht w für einen einzelnen Parameter, α für die Lernrate und L für die Verlustfunktion.

$$w := w - \alpha \cdot \frac{\partial L}{\partial w}. \quad (2.4)$$

2.3.1.4 Adam

Der Optimierungsalgorithmus Adam ist eine Weiterentwicklung des Gradientenverfahrens. Dieses Verfahren berechnet einzelne anpassungsfähige Lernraten für unterschiedliche Parameter und basiert auf zwei anderen Algorithmen und entnimmt deren Vorteile; AdaGrad und RMSProp. Adaptive Gradient (AdaGrad) stellt eine feste Lernrate pro Parameter fest und verbessert die Leistung bei sogenannten "sparse"-Gradienten [DHS11]. Laut Elibol et al. [ELJ20] sind damit Gradienten gemeint, die eine geringe Anzahl an großen Koordinaten besitzen. Root Mean Square Propagation (RMSProp) verwendet den Durchschnitt des quadratischen Gradienten, um die Parameter anzupassen [HSS12]. Adam ist rechnerisch effizient, benötigt wenig Speicher und ist gut für Anwendungen mit vielen Daten und Parametern geeignet. [KB14]

Die Parameteranpassungen werden wie folgt durchgeführt:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (2.5)$$

mit

$$\begin{aligned} g_t &= \nabla_{\theta} f_t(\theta_{t-1}) \\ m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned} \quad (2.6)$$

α steht für die Lernrate mit einem Wert von 1e-3 in der Veröffentlichung. ϵ ist eine sehr kleine Zahl, üblicherweise 1e-8 or 1e-10, um eine Nulldivision zu vermeiden. β_1 und β_2 sind die exponentiellen Abnahmeraten für die Momentenschätzwerte erster und zweiter Ordnung und betragen üblicherweise 0,9 bzw. 0,999. Nach der Berechnung des Gradienten g_t zum Zeitpunkt t werden die verzerrten Schätzwerte für das statische Moment m_t und für das Trägheitsmoment v_t aktualisiert. Darauffolgend wird die Verzerrung dieser Werte korrigiert und es ergeben sich \hat{m}_t und \hat{v}_t . Aus diesen Anteilen kann der angepasste Parameter mit Gleichung 2.5 bestimmt werden. [GBCB16]

2.3.1.5 Regularisierung

Neuronale Netze werden nach dem Training auf Bilder oder Daten, die in dem Modell noch nie zuvor eingegeben wurden, getestet. Somit kann untersucht werden, wie gut ein Modell Informationen generalisieren kann. Ein Netzwerk mit einer guten Generalisierungsfähigkeit kann gute Ergebnisse auf diesen Testdaten erzielen. Das sogenannte Overfitting tritt

auf, wenn ein Modell die Trainingsdaten zwar gut verarbeiten kann, aber nicht die Testdaten. In diesem Fall hat sich das neuronale Netz viel zu gut an die Trainingsdaten angepasst und kann noch nie gesehene Muster deshalb schlecht verarbeiten. Abbildung 2.10 zeigt den Verlauf eines solchen Trainings. Im Gegensatz dazu wird bei einer schlechten Leistung sowohl im Trainings-, als auch im Testdatensatz von Underfitting gesprochen. Um dieses fundamentale Problem des Deep Learnings zu bekämpfen, gibt es mehrere Methoden die angewendet werden können, wie Daten-Augmentation, Verwendung von mehr Trainingsdaten, Regularisierung, Batch Normalization (siehe 2.3.2.5) oder Reduzieren der Modellkomplexität [SL19]. Im Folgenden wird auf einige Techniken der Regularisierung tiefer eingegangen. [YNDT18]

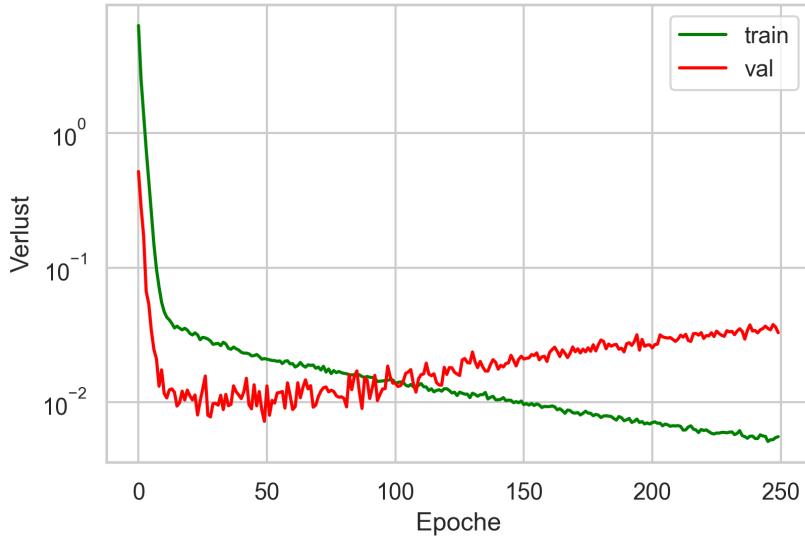


Abbildung 2.10: Beispielhafter Verlauf eines Trainings mit Overfitting. Der Trainingsloss sinkt über den gesamten Vorgang, demgegenüber fängt der Validierungsloss ab etwa der 50. Epoche an zu steigen.

Eine verbreitete Methode ist das Early Stopping, welches für ein vorzeitiges Abbrechen des Trainingvorgangs, um Overfitting zu vermeiden, sorgt. Jedes Mal, wenn der Fehler auf dem Validierungssatz verringert wird, wird eine Kopie der Parameter gespeichert. Dieser Algorithmus beendet das Training, sobald das Modell nach einer vorgegebenen Anzahl an Iterationen keine besseren Ergebnisse auf die Validierungsdaten erzielt. Das Modell mit den besten Parametern wird zurückgegeben. [GBCB16]

Das Dropout ist ein weiteres Instrument der Regularisierung. Beim Dropout wird während des Trainings ein Neuron und dessen Verbindungen mit einer vorgegebenen Ausfallwahrscheinlichkeit p deaktiviert, wodurch das Neuron von der Modellarchitektur temporär

ausgeschlossen wird. Während der Testphase sind alle Neuronen und Verbindungen aktiv, doch die Gewichte werden mit p skaliert. Hiermit wird eine übermäßige Abhängigkeit des Netzes von bestimmten Verbindungen vermieden. Dadurch wird anhand dieser Methode ein Ensemble trainiert, welches aus mehreren Teilnetzen besteht. Das Ensemble enthält alle möglichen Netzwerke, die durch Entfernen von Einheiten mit der Ausfallwahrscheinlichkeit gefertigt werden können. [SHK⁺14]

Darüber hinaus sind die L1- und L2-Regularisierungen ein weiteres Mittel, um Overfitting zu vermeiden. Bei den Dabei handelt es sich um Funktionen, die zur Verlustfunktion hinzugefügt werden. Die L1-Regularisierung entspricht der Summe der Beträge, also die L1-Normen, der Parameter. Dies führt zu einer Lösung, bei der einige Parameter einen optimalen Wert von Null haben. Die Lösung ist also dünnbesetzter, wodurch sich das Netzwerk auf die Erkennung bedeutsamer Merkmale begrenzt. Bei der L2-Regularisierungsfunktion wird die L2-Norm der Parameter betrachtet. Dies verhindert das Entstehen von Gewichten mit zu hohen Werten. Formal sind die Bestrafungen der L1- und L2-Regularisierungen für eine Gewichtung w_k in Gleichung 2.7 und Gleichung 2.8 definiert. [RR19]

$$\Omega(W^{(p)})_{l1} = |w| = \sum_{k=1}^m |w_k| \quad (2.7)$$

$$\Omega(W^{(p)})_{l2} \equiv \|W\|_2^2 \quad (2.8)$$

2.3.2 Faltende neuronale Netze

Faltende neuronale Netze (kurz CNN) sind vorwärtsgekoppelte Netzwerkarchitekturen, weil sie durch unidirektionale Verbindungen zwischen den Neuronen ausgezeichnet werden. Dementsprechend führt kein Pfad aus dem Ausgang eines Neurons zum Eingang dessen oder zu einer vorherigen Schicht des Netzes; der Aufbau stellt einen azyklischen Graphen dar. Da sie von der von Hubel und Wiesel 1959 [HW68] erforschten Funktionsweise der Informationsverarbeitung im visuellen Kortex inspiriert sind, eignen sie sich besonders für Bilder. In dieser Region des Gehirns werden globale Probleme in mehrere kleinere und daher leichter zu lösende Schritte verwandelt. [DN20]

Eine der ersten Implementierungen eines CNN ist das 1989 von LeCun et al. [LBD⁺90] publizierte und neun Jahre später in [LBBH98] verbesserte LeNet-5 Modell. Seitdem sind mehrere Architekturen und somit auch Schichten entworfen worden, mit dem Ziel, Schwierigkeiten beim Trainieren der Netze zu beseitigen und deren Effektivität zu steigern. [GWK⁺18]

2.3.2.1 Convolutional Layer

Das Convolutional Layer (engl. Convolutional Neural Network, kurz Conv) ist ein grundlegender Bestandteil aller CNN-Architekturen, weil es für das Extrahieren von Eigenschaften sorgt. Dies basiert auf der mathematischen Operation der Faltung [YNDT18]. Die Faltung innerhalb der Schicht wird typischerweise wie in Gleichung 2.9 notiert.

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (2.9)$$

Hierbei wird ein zweidimensionales Bild I mit einem ebenfalls zweidimensionalen Filterkern gefalten und ergibt somit eine sogenannte Feature Map (siehe Abbildung 2.11) [GCB16]. Der Filter wird auf die gesamte Eingabe angewendet. Durch die Wiederholung des Vorgangs mit unterschiedlichen Kernen kann eine willkürliche Anzahl an Feature Maps erzeugt werden, welche unterschiedliche Eigenschaften der Eingabe darstellen. So können unterschiedliche Feature Maps beispielsweise Augen, Mund und Nase auf Bildern von Gesichtern hervorheben. Üblicherweise werden Filter der Größe 3×3 verwendet, doch diese lässt sich modifizieren. Der Abstand zwischen zwei Positionen an denen der Filterkern angewendet wird, wird durch den Stride festgelegt. [YNDT18]

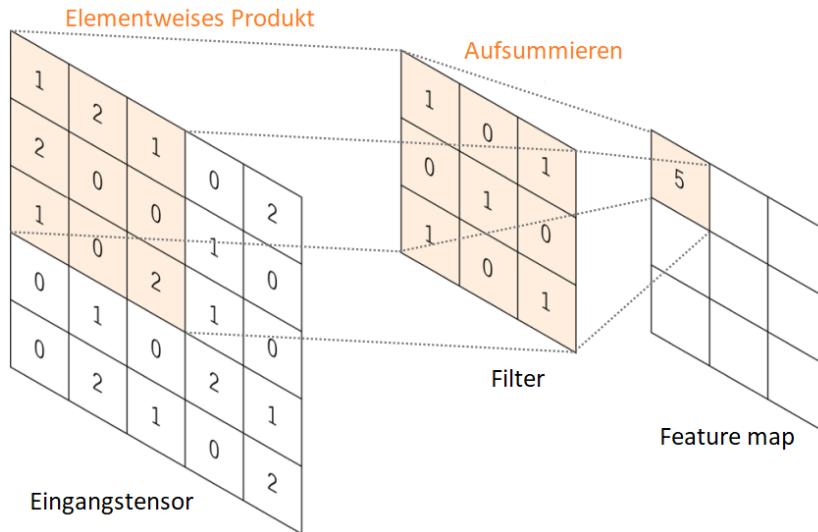


Abbildung 2.11: Beispiel einer Faltungsoperation mit einem 3×3 Filter. Entnommen und neu beschriftet aus [YNDT18].

Da die durch die Anwendung des Filters entstandene Feature Map eine reduzierte Höhe und Breite gegenüber dem ursprünglichen Bild aufweist, kann beispielsweise zero-Padding benutzt werden. An den Rändern der Eingabe werden Reihen und Spalten mit Nullen hinzugefügt, sodass die Dimension des Tensors, welches eine bestimmte Anzahl von Vektoren auf einen Vektor abbildet, beibehalten wird. [ON15]

2.3.2.2 Aktivierungsfunktionen

Ausgaben einer linearen Operation wie die Faltung werden darauffolgend in eine nichtlineare Aktivierungsfunktion eingesetzt. Die Auswahl der Funktion hat einen signifikanten Einfluss auf das Training und die Leistung des Modells [RZL17]. Im Rahmen dieser Arbeit sind folgende drei Funktionen relevant.

Die Sigmoid-Aktivierungsfunktion aus Gleichung 2.10 wird oftmals auch als logistische Funktion bezeichnet. Es handelt sich um eine begrenzt differenzierbare reelle Funktion, welche für reelle Eingabewerte definiert ist und Werte zwischen null und eins ausgibt. [NIGM18]

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.10)$$

Eine der bekanntesten Aktivierungsfunktionen ist die in Gleichung 2.11 aufgezeigte Rectified Linear Unit (ReLU) [KSH12]. Künstliche neuronale Netze mit ReLU sind einfacher zu optimieren, als Netze, welche zuvor veröffentlichten Funktionen verwenden, weil die mathematischen Operationen der Funktion simpler sind [RZL17].

$$f(x) = \max(0, x) \quad (2.11)$$

Das Exponential Linear Unit (ELU) wurde 2015 eingeführt (siehe Gleichung 2.12) und erlaubt ein schnelleres und präziseres Lernen in Netzen. Im Vergleich zur ReLU-Funktion können ELUs auch negative Werte anzeigen, beschleunigen das Training und verbessern die Generalisierung der Netze. [CUH15]

$$f(x) = \begin{cases} x & , x > 0 \\ \alpha(\exp(x) - 1) & , x \leq 0 \end{cases} \quad \text{für } \alpha > 0 \quad (2.12)$$

2.3.2.3 Pooling Layer

Ziel des Pooling Layers ist, die Dimensionalität der Eingabe zu reduzieren und gleichzeitig möglichst keinen Informationsverlust zu erleiden [AMAZ17]. Eine Invarianz gegen kleine Verschiebungen und Verzerrungen wird eingeführt und die Anzahl der erlernbaren Parameter sinkt. Ähnlich zur Faltung, sind Stride, Padding und die Größe des Filters Hyperparameter beim Pooling. Damit ist ein Parameter gemeint, dessen Wert zur Steuerung des Trainings verwendet wird. Eine der meist verbreiteten Formen ist das Max Pooling, welche in Abbildung 2.12 angewendet wird, bei dem der höchste Wert innerhalb des Bereiches übergeben wird. Eine weitere Variante ist das Average Pooling, bei dem der durchschnittliche Wert aller Werte im Bereich übernommen wird. [YNDT18]

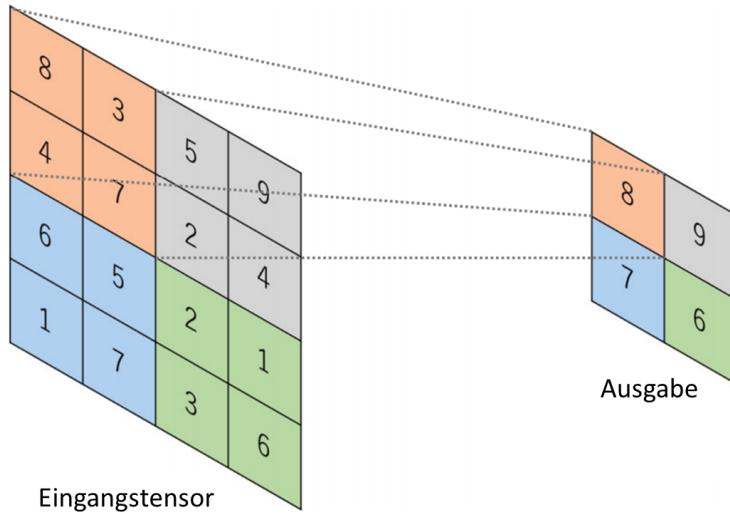


Abbildung 2.12: Beispiel zur Anwendung von Max Pooling. Ursprüngliches Bild aus [YNDT18].

2.3.2.4 Fully Connected Layer

Beim Fully Connected Layer (auch Dense Layer genannt) sind alle Neuronen direkt mit allen Neuronen der zwei benachbarten Schichten verbunden. Ein solcher Aufbau wird in Abbildung 2.13 aufgezeigt. Dies ist ähnlich zur Form in der die Nervenzellen beim einem traditionellen neuronalen Netz angeordnet sind. Da dieser Aufbau viele Parameter einbezieht, können sie eine lange Zeit beim Trainieren beanspruchen. [AMAZ17]

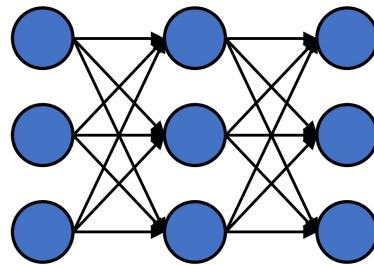


Abbildung 2.13: Aufbau eines Fully Connected Layers.

2.3.2.5 Batch Normalization Layer

Das von Ioffe et al. [IS15] entworfene Batch Normalisation Layer (kurz BN) hat die Aufgabe die Eingänge in das Layer zu normalisieren. Darunter wird eine Transformation der Werte in einen bestimmten Bereich mit Erwartungswert null und Varianz eins verstanden. Dies wird durch die Anwendung von Gleichungen 2.13, 2.14, 2.15 und 2.16 auf eine

Datenmenge $\mathcal{B} = \{x_1, \dots, x_m\}$ erzielt. [IS15]

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad (2.13)$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m m(x_i - \mu_{\mathcal{B}})^2 \quad (2.14)$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad (2.15)$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad (2.16)$$

Als erstes wird $\mu_{\mathcal{B}}$, der Durchschnitt aller Werte berechnet und daraus wird $\sigma_{\mathcal{B}}^2$, die Varianz der Eingaben, bestimmt (Gleichung 2.14). Daraufhin werden die Werte normalisiert und die transformierten Ausgaben ergeben sich in Gleichung 2.16 aus einer Skalierung und Verschiebung im Zielbereich. Hierbei sind γ und β erlernbare Parameter und ϵ eine Konstante mit einem sehr geringen Wert, um Division durch Null zu vermeiden und somit numerische Stabilität zu gewährleisten. [IS15]

Batch Normalization reduziert das Risiko eines Overfittings und verbessert den Verlauf von Gradienten über das Netz. Dies erlaubt die Verwendung höherer Lernraten, wodurch die Trainingszeiten der neuronalen Netze reduziert werden. Zudem sinken Batch Normalization Layern das Bedürfnis nach Dropout [GWK⁺18]. Für eine lineare Regression wird außerdem durch dessen Verwendung eine bessere Generalisierung laut Lathuiliere et al. [LMAPH19] erzielt.

2.3.2.6 Inception Layer

Das in [SLJ⁺15] eingeführte Inception Layer besteht aus einer parallelen Anordnung von drei Convolutional Layern und einem Pooling Layer mit den Filtergrößen 1×1 , 3×3 , 5×5 und 3×3 . Diese Operationen werden zusammen für die gleiche Eingabe ausgeführt und die jeweiligen Ausgänge werden gemeinsam verkettet. Der entsprechende Aufbau ist in Abbildung 2.14 zu sehen. [SLJ⁺15]

Aufgrund des Rechenaufwands von 5×5 Convolutional Layern, kann die Architektur aus Abbildung 2.14 mit 1×1 Convolutional Schichten zum Aufbau in Abbildung 2.15 erweitert werden. Somit sinkt die benötigte Rechenleistung und es wird eine kleinere Dimension der Ausgabe erzielt. [SLJ⁺15]

Die Weiterentwicklung dieses Moduls hat zu einer Diversifikation bei den Varianten geführt. So entstanden die Inception-A, Inception-B, Inception-C, Reduction-A und Reduction-B Module. Im Vergleich zu den zwei Jahre älteren Versionen haben diese einen geringeren Trainingsaufwandaufwand. [SIVA17]

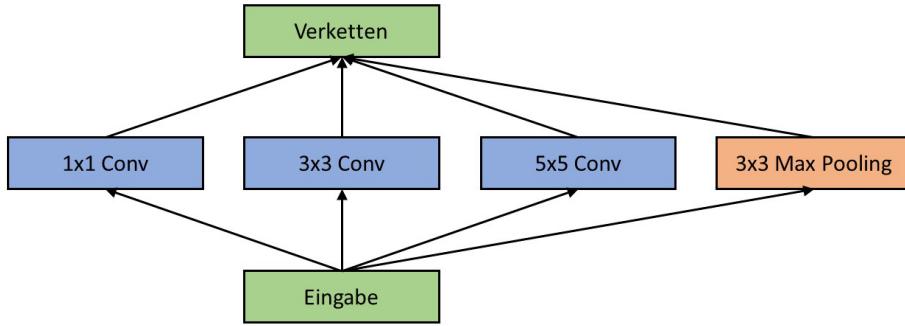


Abbildung 2.14: Aufbau des Inception Layer nach [SLJ⁺15] ohne 1x1 Convolution.

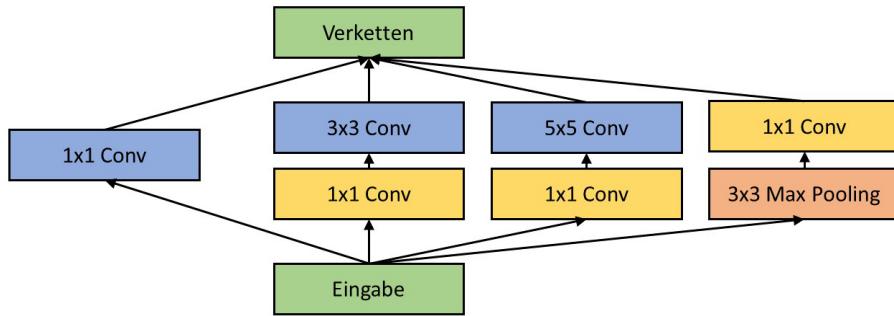


Abbildung 2.15: Aufbau des Inception Layer nach [SLJ⁺15] mit 1x1 Convolution.

2.3.2.7 Separable Convolution Layer

Das Separable Convolution Layer ist, vom Aufbau her, ähnlich zu dem eines Inception Moduls, welches nur eine Größe von Filtern verwendet und keinen Pooling Layer besitzt. Dies kann als gesamte 1×1 Convolution, die von weiteren einzelnen Convolutional Layern gefolgt wird, gesehen werden. Die nachfolgenden Schichten arbeiten auf nichtüberlappendenden Segmenten der Ausgabe. Beide Operationen werden von einer ReLU-Aktivierungsfunktion gefolgt. Ein Beispiel für eine solche Architektur ist in Abbildung 2.16 aufgezeigt. [Cho17]

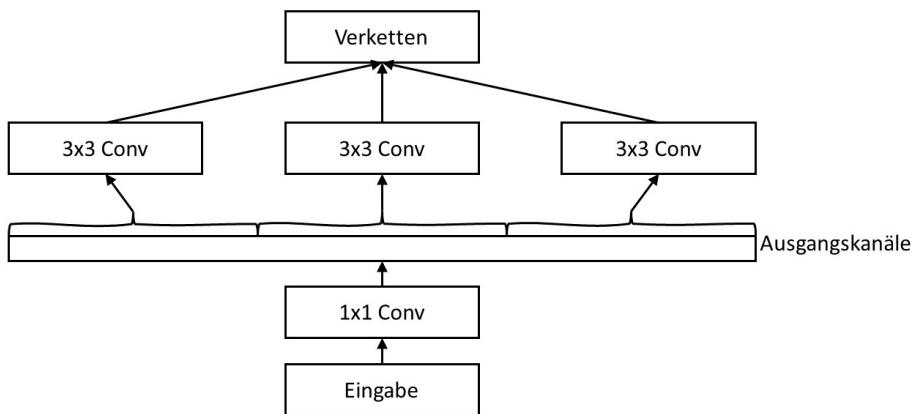


Abbildung 2.16: Beispiel eines Separable Convolution Layer nach [Cho17]

2.3.2.8 Residual Layer

Beim sogenannten Residual Layer werden mehrere Convolutional und Batch Normalization Layer hintereinander gestellt und es wird eine sogenannte "Identitätsverknüpfungsverbindung", die auch als Abkürzung (engl. Shortcut) bezeichnet wird, eingeführt. Shortcut-Verbindungen überspringen eine oder mehrere Schichten. In diesem Fall wird diese Verbindung so verwendet, dass ihr Ausgang mit dem Ausgang der gestapelten Schichten addiert wird. Hierbei müssen beide Größen von der gleichen Dimension sein. Die Struktur ist in Abbildung 2.17(a) dargestellt. [HZRS16a]

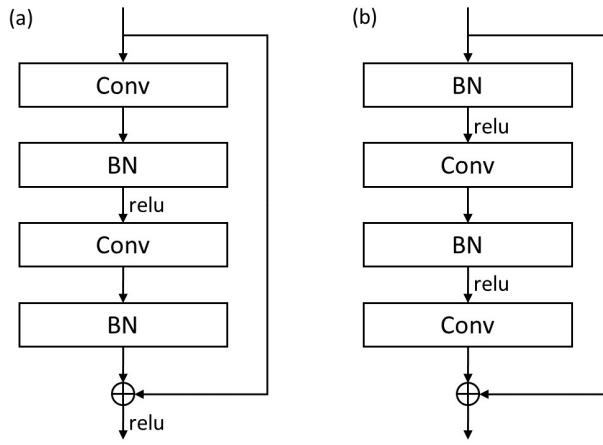


Abbildung 2.17: (a) Aufbau des originellen Residual Layers nach [HZRS16a]; (b) Aufbau der Weiterentwicklung eines Residual Layers [HZRS16b].

Ein Jahr nach der Einführung des Residual Layers führte He et. al [HZRS16b] eine verbesserte Version ein. Ihr Aufbau ist in Abbildung 2.17(b) aufgezeigt. Hiermit wurde der Fehler auf den CIFAR-10-Datensatz, welches aus 60.000 Bildern aus 10 Klassen besteht [KH⁺09], von 7,61% auf 4,92% verringert. [HZRS16b]

2.3.2.9 Bottleneck Layer

Das Bottleneck Layer (siehe Abbildung 2.18) ist die nächste Entwicklung des ursprünglichen Residual Layers und besteht aus drei hintereinanderliegenden Convolutional Layer mit Filtern der Größen 1×1 , 3×3 und 1×1 . Die Schichten mit den 1×1 Filtern reduzieren erst und erhöhen dann die Dimensionen, somit hat die mittlere Schicht mit dem 3×3 Filter eine kleinere Ein- und Ausgangsdimension. Dieser Aufbau erinnert an einen Flaschenhals (engl. Bottleneck), woran der Name des Filters angelehnt ist. Die parameterfreie Verzweigung, die an den zwei Enden mit den höchsten Dimensionen verbunden ist, führt außerdem zu einem geringeren Zeitverbrauch. [HZRS16a]

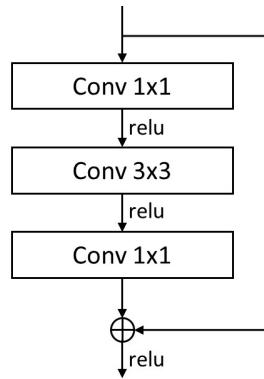


Abbildung 2.18: Aufbau eines Bottleneck Layers nach [HZRS16a].

2.3.2.10 Inverted Residual Layer

Ein Inverted Residual Layer, auch MBCConv Layer genannt, besteht aus einer 1×1 Convolution, einer 3×3 Depthwise Convolution (kurz DepthConv), einer weiteren 1×1 Convolution und einer Shortcut-Verbindung vom Eingang des Layers bis zum Ausgang der zweiten 1×1 Convolution. In Abbildung 2.19 ist dieser Aufbau zu sehen. Die Depthwise Convolution in der Mitte funktioniert ähnlich zu einer normalen Convolution. Der Unterschied liegt darin, dass der Filter auf den einzelnen Informationskanälen angewendet wird. Erst wird die Eingabe und der Filter in den Kanälen gespalten. Daraufhin wird der Filter angewendet und zuletzt werden die Ausgaben zusammengestellt. Diese Operation vermindert die Anzahl an Parametern. [SHZ⁺18]

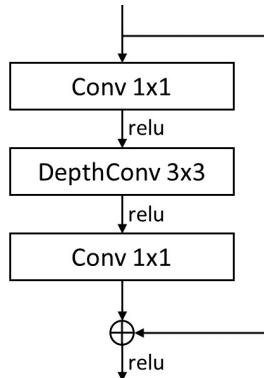


Abbildung 2.19: Zusammenstellung des Inverted Residual Layer [SHZ⁺18].

2.4 Netzwerkarchitekturen

Seit der Einführung von CNNs wurden viele unterschiedliche Architekturen für diverse Zwecke entworfen. Somit ist im Laufe der Zeit die Komplexität, aber auch die Leistung

gestiegen. Dies ist am Beispiel von ImageNet ersichtlich. ImageNet ist ein Datensatz bestehend aus 14.197.122 annotierten Bildern aus unterschiedlichen Kategorien. In den letzten zehn Jahren ist die Genauigkeit des Modells mit dem besten Ergebnis von 50,9% auf 90,2% gestiegen (siehe Abbildung 2.20) [ima]. Im Folgenden werden einige der in den letzten Jahren entworfenen Architekturen nach deren Veröffentlichungsjahr aufgezeigt.

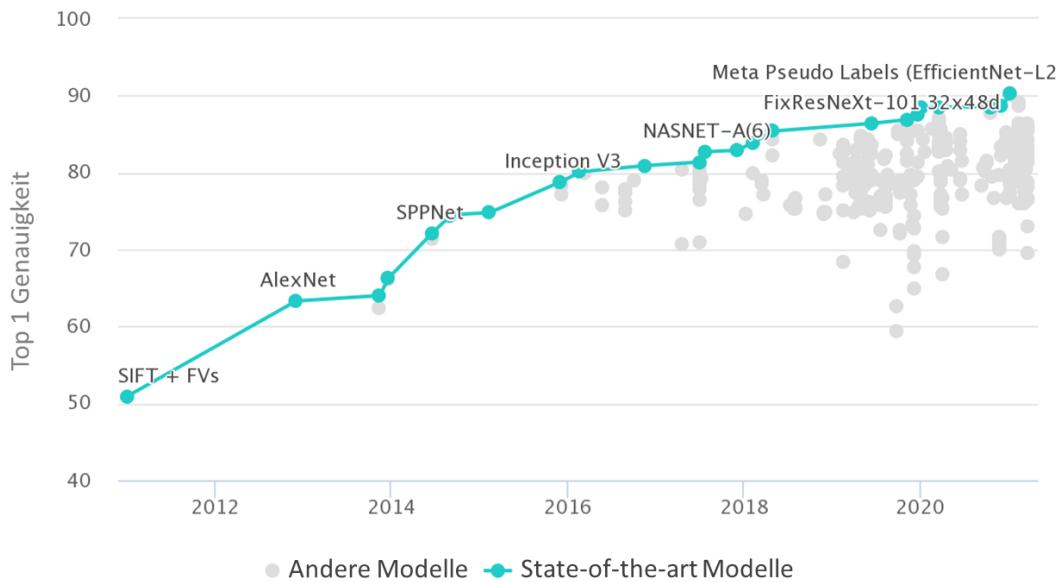


Abbildung 2.20: Rangliste für Modellergebnisse auf ImageNet über die Zeit [ima].

2.4.1 AlexNet

Die von Krizhevsky et al. [KSH12] entworfene AlexNet-Architektur ist ein tieferes und breiteres Modell als das von LeCun [LBBH98] veröffentlichte LeNet-5 CNN. AlexNet erreichte 2012 den State of the Art bei der Erkennungsgenauigkeit von Bildern und stellte einen Durchbruch in den Bereichen des Maschinellen Lernens und der Computer Vision dar [ATY⁺18].

2.4.2 VGG

Das Visual Geometry Group (VGG) wurde von der Arbeitsgruppe, die diesen Namen an der University of Oxford trägt, entworfen. Es ist eine Weiterentwicklung des AlexNet und ist nach den gleichen Prinzipien aufgebaut, wie die Netze aus Ciresan et al. [CMM⁺11] und Krizhevsky et al. [KSH12]. Die zwei Varianten des Netzes, das VGG16 und das VGG19 unterscheiden sich in der Anzahl an enthaltenen Convolutional Layern, also in der Tiefe des faltenden neuronalen Netzes. Hierbei steht die Zahl am Ende der Namen für die Anzahl an Schichten mit Parametern, die Teil der Architekturen sind. [SZ14]

Ein großer Unterschied zu den neuronalen Netzen die bis 2013 den State of the Art repräsentierten, ist die Größe des Filters im ersten Convolutional Layer. Im Gegensatz zu den Filtern der Größe 11×11 mit einem Stride von vier in [KSH12], wird im gesamten Netz eine 3×3 Größe mit Stride eins bevorzugt.

Laut Krizhevsky et al. [SZ14] besitzen zwei der 3×3 Convolutional Layer ein effektiveres Empfangsfeld als das eines 5×5 Layers; drei dieser Schichten besitzen das Empfangsfeld einer 7×7 Schicht, wie in [ZF14]. Dies führt zu einer starken Reduzierung der Anzahl an Parametern. Für einen Stapel mit drei Convolutional Layern mit Filtern der Größe 3×3 mit C Informationskanälen ergeben sich

$$3(3^2C^2) = 27C^2 \quad (2.17)$$

Parameter. Im Vergleich dazu, beträgt die Anzahl an Parametern bei nur einer Schicht mit 7×7 Filtern

$$7^2C^2 = 49C^2. \quad (2.18)$$

2.4.3 ResNet

Seit der Einführung der AlexNet-Architektur gehen neue neuronale Netze immer tiefer. So stieg die Anzahl an enthaltenen Convolutional Layern im Laufe der Jahre. 2012 hatte AlexNet [KSH12] fünf Convolutional Layer, drei Jahre später ist die Anzahl auf 16 mit dem VGG [SZ14] gestiegen.

Um eine Erhöhung der Tiefe des Netzes zu erzielen, ist es allerdings nicht ausreichend, weitere Schichten aufeinander zu stapeln. Bei tiefen Netzen tritt nämlich das sogenannte verschwindende Gradientenproblem [Tak17] auf, weshalb sie schwieriger zu trainieren sind. Der Gradient wird auf frühere Schichten zurück propagiert und kann, infolge wiederholten Multiplikationen, unendlich klein werden. Da das Netz tiefer und tiefer geht, wird dessen Leistung gesättigt oder nimmt schnell ab. Dies ist anhand des Beispiels aus Abbildung 2.21 ersichtlich, bei dem das tiefere der beiden Netze eine schlechtere Leistung auf dem CIFAR-10-Datensatz erbrachte.

Um dieses Problem zu beseitigen, werden Identitätsverknüpfungsverbindungen im Residual Layer eingeführt. Diese Schichten gelten als grundlegende Bausteine der Residual Networks, oder ResNets. Die Resnet50 und ResNet101 Architekturen nutzen das Bottleneck Layer, welches eine Weiterentwicklung des Residual Layers ist. [HZRS16a]

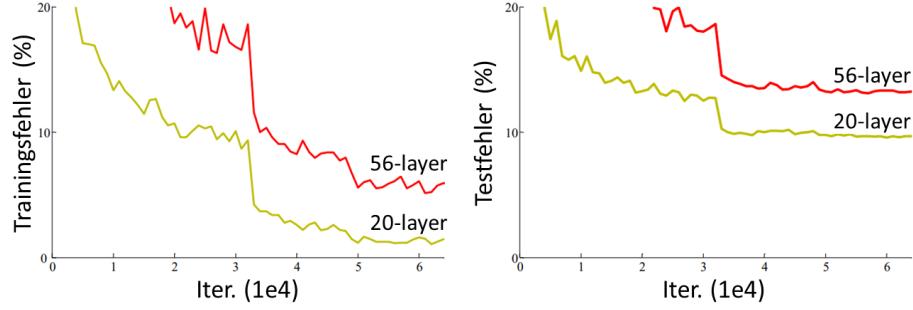


Abbildung 2.21: Trainingsfehler (links) und Testfehler (rechts) auf dem CIFAR-10-Datensatz mit ”einfachen” Netzen mit 20 und 56 Schichten. Entnommen und Beschriftung übersetzt aus [HZRS16a].

2.4.4 Inception-v4

Inception-v4 [SIVA17] ist die vierte Generation der von Google entworfenen Inception-Architektur. Das Inception Layer wurde zum ersten Mal 2014 von Szegedy et al. [SLJ⁺15] mit Inception-v1, oder GoogLeNet als Huldigung an Yann LeCuns LeNet-5 [LBBH98], eingeführt. In diesem Jahr gewann diese Architektur die Large Scale Visual Recognition Challenge (kurz: ILSVRC2014). Inception-v2 [SVI⁺16] implementierte Batch Normalization und eretzte die 5×5 Convolutional Layern durch zwei 3×3 Schichten, um die Anzahl an Parametern zu reduzieren. In der gleichen Veröffentlichung wurde ebenfalls die Inception-v3-Architektur [SVI⁺16] eingeführt.

Die Prämisse für Inception-v4 ist das Erschaffen einheitlicherer Module. Zudem waren einige Module der früheren Entwicklung komplizierter als nötig. Hierfür wurde der ”Stem”, welcher die Operation vor den Inception-Blöcken darstellt, verändert. Zudem wurden die drei Hauptmodule Inception-A, -B und -C aktualisiert und es wurden sogenannte ”grid-reduction” Module eingeführt, um die Breite und Höhe des Gitters zu verändern. [SIVA17]

2.4.5 Inception-ResNet-v2

Inception-ResNet-v2 ist ein Convolutional Neural Network, welches auf der Inception-Architektur basiert und die Identitätsverknüpfungsverbindung der ResNets einbaut. Um dies umzusetzen, müssen Ein- und Ausgang nach der Convolution von der gleichen Dimension sein. Dafür werden Convolutional Layer mit Filtern der Größe 1×1 nach den ursprünglichen Convolutions eingesetzt. In den Inception-ResNet-A, -B und -C Modulen wird das Pooling Layer durch den Shortcut ersetzt. [SIVA17]

2.4.6 Xception

Die 2017 veröffentlichte Xception Architektur ist von der Inception-Architektur [SVI¹⁶] inspiriert worden. Ein wichtiger Bestandteil von Xception ist die Einführung des Separable Convolution Layer. Der Aufbau von Xception besteht aus drei grundlegenden Teilen; dem Eingang, dem mittleren Durchfluss, welcher acht Mal wiederholt wird, und dem Ausgang. [Cho17]

2.4.7 MobileNetV3

MobileNetV3 ist eine Weiterentwicklung des MobileNet aus [HZA¹⁷] und wurde für die Benutzung mit CPUs von Mobiltelefonen entworfen. Das neuronale Netz besteht aus mehreren Inverted Residual Layern (siehe Abschnitt 2.3.2.10) und führt eine neue Aktivierungsfunktion, "h-swish", ein. [HSC¹⁹]

$$f(x) = x \cdot \sigma(\beta x), \text{ mit } \sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.19)$$

Die ursprüngliche Swish-Funktion wurde 2017 von Ramachandran et al. [RZL17] eingeführt und ist in Gleichung 2.19 beschrieben. Hierbei ist β ein änderbarer Parameter. Laut [HSC¹⁹] stellt h-swish eine Verbesserung der ursprünglichen Swish-Funktion dar, da die Rechenzeit verringert wird. Diese wird in Formel 2.20 aufgezeigt.

$$\text{h-swish}(x) = x \frac{\min(\max(x + 3, 0), 6)}{6} \quad (2.20)$$

2.4.8 EfficientNet

Die Grundidee hinter der Architektur von EfficientNets ist das sogenannte Compound Scaling. Anstatt Netze anhand der Tiefe (Anzahl der Schichten), Breite (Anzahl der Informationskanäle) oder Bildgröße zu skalieren, werden alle drei Dimensionen so verändert, dass sie ein Gleichgewicht bilden. Somit kann eine bessere Genauigkeit und Effizienz bei den neuronalen Netzen erzielt werden. Diese Methode kann auch bei anderen Architekturen angewendet werden. So wurde gezeigt, dass das Compound Scaling bei MobileNetV1, MobileNetV2 und ResNet-50 zu besseren Ergebnissen auf dem Imagenet-Datensatz führte. [TL19]

$$\begin{aligned} \text{Tiefe: } d &= \alpha^\phi \\ \text{Breite: } w &= \beta^\phi \\ \text{Auflösung: } r &= \gamma^\phi \end{aligned} \quad (2.21)$$

$$\forall \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2, \alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

Compound Scaling verwendet einen Koeffizienten ϕ , um die Breite, Tiefe und Auflösung gemeinsam zu bestimmen. Die Gleichungen aus 2.21 beschreiben die skalierten Attribute. Der benutzerdefinierte Parameter ϕ bestimmt die zusätzlichen verfügbaren Ressourcen und α , β und γ bestimmen die Verteilung dieser Ressourcen über der Tiefe d , Breite w und Auflösung r .

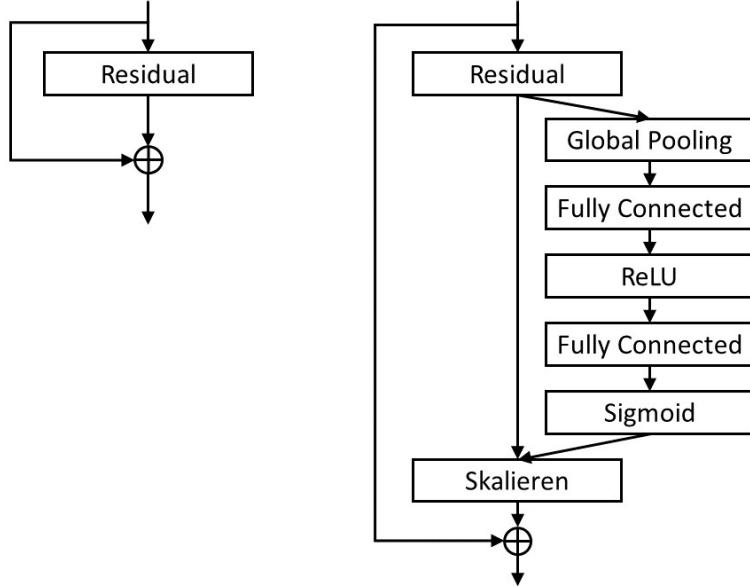


Abbildung 2.22: Aufbau eines Residual Layers (links) und dem SE-ResNet Modul (rechts) nach [HSS18].

Der Hauptbaustein des EfficientNet ist das Inverted Residual Layer, welches zusätzlich noch durch Squeeze-and-Excitation (SE) optimiert wird [TL19]. Der SE-Block besitzt am Eingang eine Convolution. Mithilfe des Poolings wird jeder Informationskanal auf einen einzigen Wert zusammengepresst. Das darauffolgende Fully Connected Layer mit einer ReLU als Aktivierungsfunktion fügt eine Nichtlinearität ein und das zweite Fully Connected Layer glättet die Funktion. Dieses Layer verwendet die Sigmoid-Funktion. Am Ende wird jede Feature Map der Faltungsoperation anhand des Ergebnisses der hinzugefügten Schichten skaliert. Abbildung 2.22 zeigt die Umsetzung für ein Residual Layer. Diese zusätzliche Struktur benötigt unter 1% an zusätzlicher Rechenleistung und kann zu jedem neuronalen Netzwerk hinzugefügt werden. [HSS18]

Kapitel 3

Verwandte Arbeiten

Bisher wurde kein vollautomatisiertes System zur geometrischen Überprüfung der Stentstruktur konzipiert. In diesem Kapitel wird an erster Stelle die Forschung im Bereich der bildbasierten Analyse von Geflechten vorgestellt. Darauffolgend wird auf den aktuellen Stand der automatisierten Inspektion von Stents eingegangen, welche größtenteils auf Verfahren der Bildverarbeitung basiert. Im Anschluss wird auf das Projekt Stents4Tomorrow [Pro] eingegangen und der bisherige Fortschritt erläutert.

3.1 Bildbasierte Analyse von Geflechten

Die bildbasierte Analyse von Geflechten wurde bereits von mehrfachen Studien untersucht, was zu diversen Lösungsansätzen geführt hat. So basieren die von Phoenix [Pho78], Zhang et al. [ZBB99] und Guyader et al. [GGH13] entworfenen Systeme auf geometrische Beziehungen im Geflecht. Die von Alpyildiz [Alp12] veröffentlichte Arbeit schlägt dreidimensionale Garnwege im Geflecht vor, welche für die Simulation der Geflechtgeometrie notwendig sind. Darüber hinaus entwickelten Hunt et al. [HC19] ein Computer-Vision-System für die Echtzeitvermessung des Flechtwinkels in der Produktion. Das System verwendet eine 2D diskrete Fourier-Transformation für die Verarbeitung der Bilder und erzielt eine Genauigkeit von $\pm 1\text{-}2$ Grad bei der Vermessung von einzelnen Winkeln. [HC19] Liu et al. [LK19] implementierten ein ähnliches System unter der Verwendung des Canny-Algorithmus [Can86] und der Hough-Transformation [DH72].

Xiao et al. untersuchten die Vermessung der Picklänge und des Flechtwinkels mit unterschiedlichen Ansätzen in mehreren Studien. Beim ersten Ansatz [XPZ⁺18] wurden vielfältige Kantendetektoren eingesetzt, um die Ecken im Bild zu ermitteln. Diese Herangehensweise wurde darauffolgend in [XPZ⁺19] durch die Einführung des "local edge extreme"-Algorithmus ersetzt. In einer weiteren Veröffentlichung [PXG⁺19] wurde für die Vermessung der oben genannten Größen eine auf die sogenannte "gray projection" ba-

sierte Methode eingeführt, mit der eine hohe Genauigkeit erzielt wurde. Darüber hinaus wurde ein Faster R-CNN-Modell in [XPG⁺20] trainiert, um Picklängen und Flechtwinkel in Bildern, wie in Abbildung 3.1 aufgezeigt, zu lokalisieren und messen.

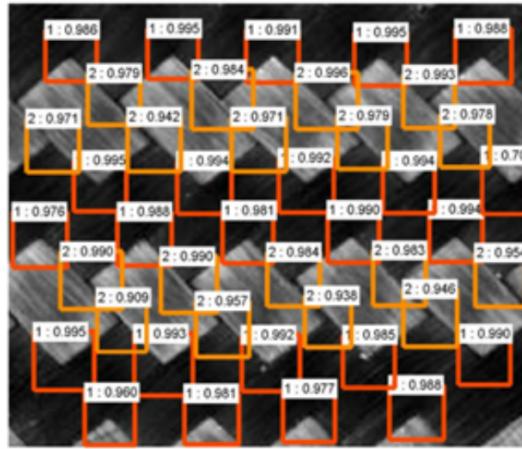


Abbildung 3.1: Testergebnisse der Anwendung eines Faster R-CNN auf Bilder von Ge flechten. Entnommen aus [XPG⁺20].

3.2 Visuelle Inspektion von Stents

Es wurden bereits einige Untersuchungen und Studien im Bereich der visuellen Inspektion von Stents durchgeführt. Eine von Farmer [Far05] durchgeführte Studie führte ein Bildverarbeitungssystem ein, um die Drähte und Kanten des Stents zu untersuchen. Das System wird von zwei Sensorenpaketen betrieben. Das erste Paket wird für die Überprüfung der Drähte und Kanten verwendet, wohingegen das zweite Paket die Fläche zwischen den Drähten betrachtet. Das entworfene System verfehlte 0,08% der Defekte, identifizierte 8% der Defekte falsch positiv und konnte mit einer Geschwindigkeit von 18 mm pro Minute und sechs Sekunden betrieben werden. [Far05]

In der von Ibraheem und Binder [IB09] veröffentlichten Arbeit, wird eine Zeilenkamera verwendet, um ein hochauflösendes Bild des gesamten Stents aufzunehmen. Zur Erkennung von Fehlern werden unterschiedliche Verfahren aus der Kantendetektion und Bildsegmentierung angewendet. Somit werden bogenförmige Segmente an den Kreisbögen des Stents im Bild angepasst. Anhand dieser Anpassung können die Mittelpunkte und Radien der Bögen bestimmt werden. Aus den Abweichungen zwischen den Werten können lokale Fehler detektiert werden. [IB09]

In einer von Bermudez et al. [BLC⁺17c] durchgeführten Studie werden Bilder, wie in Abbildung 3.2 zu sehen, mithilfe einer Mikroskopanordnung, einem dreifachen Beleuchtungssystem und einer Rotationsbühne aufgenommen. Anhand einer Segmentierung kann

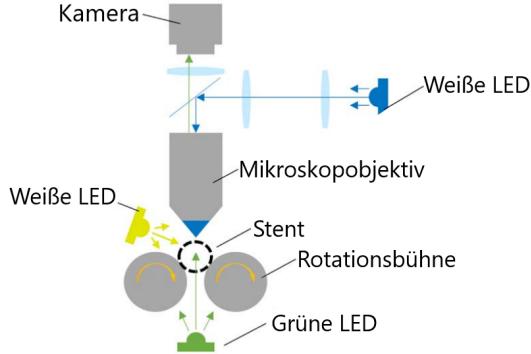


Abbildung 3.2: Aufbau des von Bermudez et al. 2016 entworfenen Systems mit einem dreifachen Beleuchtungssystem. Entnommen und übersetzt aus [BLC⁺17c].

der Draht des Stents vom Hintergrund des Bildes isoliert werden (siehe Abbildung 3.3). Durch die Anwendung morphologischer Operatoren wird das segmentierte Bild weiterverarbeitet, um Kanten sichtbar zu machen. Im Anschluss werden die Drahtbreite und Kantenrundheit auf geometrische Fehler untersucht [BLC⁺17c]. Eine weitere Veröffentlichung stellte ein verbessertes System mit einem verbesserten Vermessungssystem vor. Hierfür werden sogenannte Confocal und Coherence Scanning Interferometry (kurz CSI [Gro11]) angewendet, um die Form, Textur und Defekte der Oberfläche von Stents anhand von 3D-Messtechnik evaluieren zu können [BLC⁺17b]. Des Weiteren wird in einer darauffolgenden Studie ein Klassifikator eingeführt, welcher Defekte von Stents an der Oberfläche oder Kanten in unterschiedliche Arten einstufen kann [BLC⁺17a].

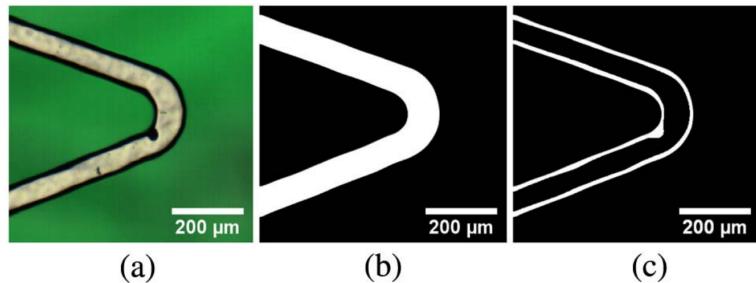


Abbildung 3.3: Stent mit einem Rissdefekt: (a) ursprüngliches Bild, (b) Maske der Oberfläche, (c) Maske der Kanten. Entnommen aus [BLC⁺17c].

3.3 Stents4Tomorrow

Im Rahmen des Stents4Tomorrow-Projekts [Pro] soll der Prozess der visuellen Inspektion von Stents während der Produktion automatisieren. Ein wichtiger Bestandteil ist die Erkennung von Fehlern im Geflecht. Houssem [EK19] setzte sich sowohl mit der

Berechnung des Flechtwinkels als auch mit der Erkennung von Unregelmäßigkeiten im Flechtmuster auseinander. Ähnlich wie in [HC19] wurden die Hough- und 2D diskrete Fourier-Transformation angewendet und miteinander verglichen. Hierbei führte die zweidimensionale diskrete Fourier-Transformation zu besseren Ergebnissen bei Bildern aller Flechtwinkel. Zur Überprüfung des Flechtmusters wurde die U-Net-Architektur implementiert und mit einem Convolutional Autoencoder verglichen. U-Net zeigte eine höhere Leistung vor. [EK19]

Zusammen mit dem Flechtwinkel ist die Picklänge ein wichtiger Bestandteil der Geometrie von Stents. Aus diesem Grund untersuchte Schorle [Sch20] die Vermessung der Picklänge anhand unterschiedlicher Methoden. Durch die Anwendung faltender neuronaler Netze konnte eine Genauigkeit von 0,267 Millimetern bei der Vermessung der durchschnittlichen Länge von Picks auf einer Linie erreicht werden. [Sch20]

Neben der zuverlässigen Fehlererkennung im Stent ist die Effizienz des Systems bei der Korrektur der erkannten Fehler wichtig. Djamal [Dja20] setzte sich mit der Optimierung dieses Vorgangs auseinander. Somit wurden sechs Abläufe zur Fehlerkorrektur entworfen. Bei den zwei nicht-optimierten Vorgängen wurden zwei optimale Ansätze gefunden. Zudem konnte eine optimale Lösung für die Genauigkeit und für die Ablaufzeit konzipiert werden. [Dja20]

Der Grundbestandteil der Fehlerkorrektur ist das entsprechende System, um diese auszuführen. In [KB20] wird das Konzept eines kamerabasierten Korrektursystems zur Verbesserung des Flechtprozesses von Stents vorgestellt. Die Funktionsweise des Algorithmus des Systems ist hierbei in drei grundlegende Schritte aufgeteilt: Erst wird die Picklänge vermessen, darauffolgend wird über die Korrektur entschieden und im Anschluss wird diese spezifiziert. Das vorgestellte System zeichnet sich durch die Verwendung mehrerer arbeitsteiliger Convolutional Neural Networks aus. Diese sind in einer Programmlogik miteinander verknüpft, sodass Messfehler besser eingegrenzt werden können. [KB20]

Kapitel 4

Konzept

Im folgenden Kapitel wird der grundlegende Aufbau dieser Arbeit vorgestellt und die im Rahmen dieser Arbeit relevanten Schritte hervorgehoben. Außerdem wird das Verfahren zur Berechnung der Picklänge erläutert und auf die Auswahl der Netzwerkarchitekturen und das Erfassen der Bilddaten eingegangen.

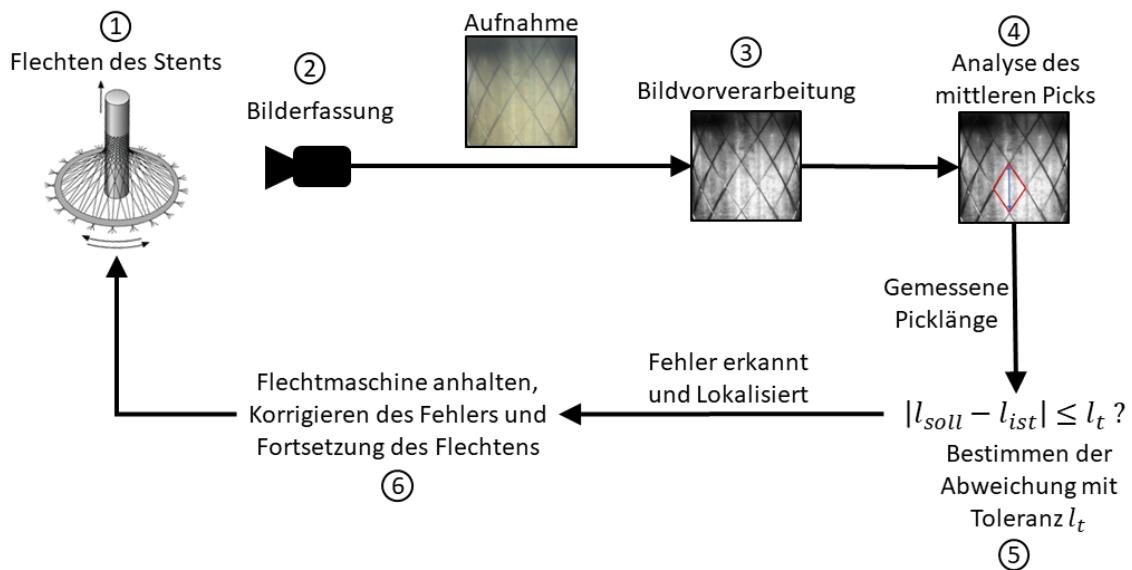


Abbildung 4.1: Aufbau des Gesamtsystems zur automatisierten Fehlererkennung und -korrektur von Stents basierend auf Kamerabildern

4.1 Aufbau des Projekts

Der grundlegende Aufbau des Systems für die automatische Fehlererkennung von Stents in der Produktion ist in Abbildung 4.1 aufgezeigt. Die Stents werden bei einem vorgegebenen Flechtwinkel von einer Flechtmaschine hergestellt. Der Verlauf dieses Prozesses

soll mit einer Kamera überwacht werden. Nach einer Vorverarbeitung der Bilder wird die Picklänge des mittleren Picks bestimmt und anhand der gewünschten Länge die Abweichung zwischen diesen zwei Werten berechnet. Mittels eines festgelegten Toleranzwertes wird entschieden, ob ein Fehler vorliegt oder nicht. Sollte ein Fehler im Geflecht vorhanden sein, wird die Flechtmaschine angehalten, der Defekt korrigiert und das Flechten im Anschluss fortgesetzt. Im Rahmen dieser Arbeit wurden drei dieser Schritte betrachtet: die Bildvorverarbeitung der Aufnahmen, deren Analyse, um die Länge des mittleren Picks zu bestimmen und die Bestimmung der Abweichung der Netzausgaben.

4.2 Berechnung der Picklänge

Es gibt mehrere Ansätze, mit denen eine Pickanalyse durchgeführt werden kann. Schorle [Sch20] untersuchte Methoden der Bildverarbeitung des klassischen maschinellen Lernens und faltende neuronale Netze, um den Mittelwert der Längen aller Picks in der Mitte des Bildes zu vermessen. Wie in Abbildung 4.2 dargestellt, kann trotz eines korrekten Mittelwertes ein Fehler in der Struktur des Stents vorliegen. Aus diesem Grund wird in dieser Arbeit nur der sich in der Mitte des Bildes befindende Pick analysiert. Aus den in [Sch20] untersuchten Methoden wurden mit faltenden neuronalen Netzen die besten Ergebnisse erreicht. Deshalb werden sie im Rahmen dieser Arbeit für die Vermessung der Picklänge eingesetzt.

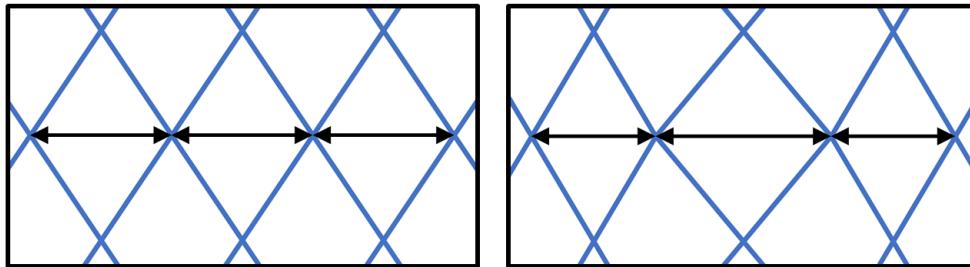


Abbildung 4.2: Beispiel für zwei Zeilen von Picks mit gleicher durchschnittlicher Picklänge, fehlerfrei (links) und fehlerhaft (rechts)

4.3 Auswahl der faltenden neuronalen Netzen

Um möglichst vielfältige Netzwerkarchitekturen zu analysieren, wurden die in Kapitel 2.4 beschriebenen Modelle implementiert und optimiert. Die Auswahl enthält sowohl aktuelle Architekturen wie EfficientNet, als auch Netze wie AlexNet, welche einen einfachen Aufbau haben. Zudem variieren die Einsatzgebiete der gewählten Architekturen.

Bei der Anwendung faltender neuronaler Netze mit Bildern gibt es drei große Anwendungsbereiche. Die meist verbreitete ist die Bildklassifizierung, welche Bilder in unterschiedliche Kategorien einstuft. Weiterhin wird die Objekterkennung angewendet, um beliebige Objekte innerhalb eines Bildes mit einem sogenannten Begrenzungskasten zu erkennen und einzurahmen. Im Vergleich dazu wird bei der semantischen Segmentierung jedem Pixel eine Kategorie zugeordnet. So kann das Bild in mehrere Bereiche oder Cluster aufgeteilt werden. Einige Beispiele zu diesen Bereichen sind in Abbildung 4.3 aufgezeigt. In der Netzauswahl befinden sich Netze, die auch für weitere Anwendungen eingesetzt werden. Dazu zählen die Maschinenübersetzung, bei der Texte aus einer Sprache in eine andere umgewandelt werden können und die Superauflösung, welche sich mit der Verbesserung der Auflösung von Bildern auseinandersetzt.

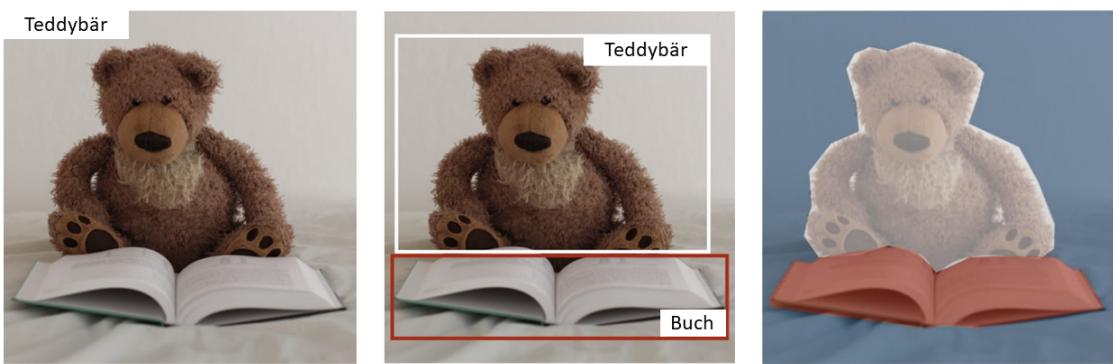


Abbildung 4.3: Beispiele für die drei größten Anwendungsbereiche von CNNs: Klassifizierung (links), Objekterkennung (Mitte), Segmentierung (rechts). Ursprüngliche Bilder aus [AA18].

4.4 Erfassung der Trainingsdaten

Das Trainieren von neuronalen Netzen erfordert einen Datensatz. Hierfür wurden Bilder von Stents inklusive der zugehörigen Picklängen als Labels verwendet. Die vom Projekt Stents4Tomorrow [Pro] zur Verfügung gestellten Bilder zeigen mehrere Segmente von Stents mit unterschiedlichen Flechtwinkeln. Diese Bilder der Größe 3088×2064 enthalten Informationen, wie der Hintergrund, welche für die Vermessung der Picklänge irrelevant sind. Zudem wird ein sehr langer Abschnitt des Stents gezeigt, wodurch nicht immer ein Pick in der Mitte des Bildes zu sehen ist. Aus diesem Grund wurden zur Verfügung gestellte vorgeschnittene Bilder der Größe 450×450 verwendet, um die neuronalen Netze zu trainieren. Ein Beispiel für eines der ursprünglichen Bilder und ein vorgeschnittenes Bild ist in Abbildung 4.4 zu sehen. Als Label für die Bilder wird die Länge des mittleren Picks im Bild in Pixel angegeben.

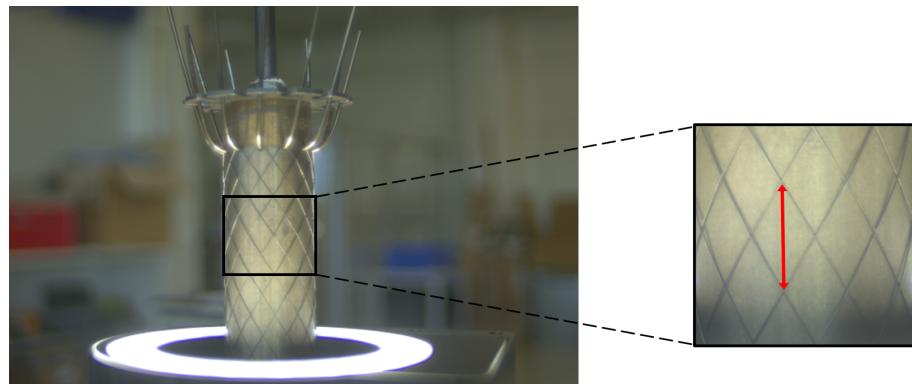


Abbildung 4.4: Ursprüngliches Bild der Größe 3088×2064 (links) und vorgeschnittene Variante (rechts). Die Länge des sich in der Mitte des in der Bildmitte befindlichen Picks ist rot eingezeichnet.

Kapitel 5

Implementierung

Die Realisierung der Bildvorverarbeitung, Analyse des mittleren Picks und Bestimmung der Abweichung werden im folgenden Kapitel vorgestellt. Im ersten Abschnitt wird in die Aufbereitung und Aufteilung des Datensatzes für das Trainieren und Testen der neuronalen Netze eingegangen. Darauffolgend wird auf die Implementierung der faltenden neuronalen Netze eingegangen und die Auswahl der Hyperparameter erklärt. Anschließend wird im dritten Abschnitt dieses Kapitels eine vereinfachte Form der Fehlerlokalisierung aufgebaut.

5.1 Trainingsdaten

Wie in Abschnitt 4.4 beschrieben, wurden vorgeschnittene Bilder der Größe 450×450 verwendet; insgesamt standen 23.527 Bilder zur Verfügung. Die dazugehörigen Labels der Bilder wurden in Zusammenarbeit mithilfe eines Python-Skriptes erstellt. Hierbei wurden Bilder, bei denen der Stent nicht sichtbar war, kein Pick im Mittelpunkt lag oder sich dieser zwischen zwei Picks befand, aussortiert. Insgesamt betrifft dies 7.586 Aufnahmen. Beispiele dafür sind in Abbildung 5.1 dargestellt.

Zudem wurden im Anschluss weitere 11.150 Bilder aussortiert. Aufnahmen, bei denen die Lichtverhältnisse suboptimal waren, wurden entfernt, weil die Sichtbarkeit des mittleren Picks somit beeinflusst wurde und er nicht immer klar erkennbar war. Bilder von Stents, bei denen einer der Drähte gerissen war, wurden ebenfalls nicht übernommen. Abbildung 5.2 zeigt drei Beispiele für die zuvor genannten Fälle.

Nach dem Aussortieren ergab sich ein Datensatz mit 4.791 Bildern von Stents mit Flechtwinkeln von 30, 40, 50, 60 und 70 Grad. Tabelle 5.1 zeigt die Aufteilung der Anzahl an verwendbaren und aussortierten Bilddaten.

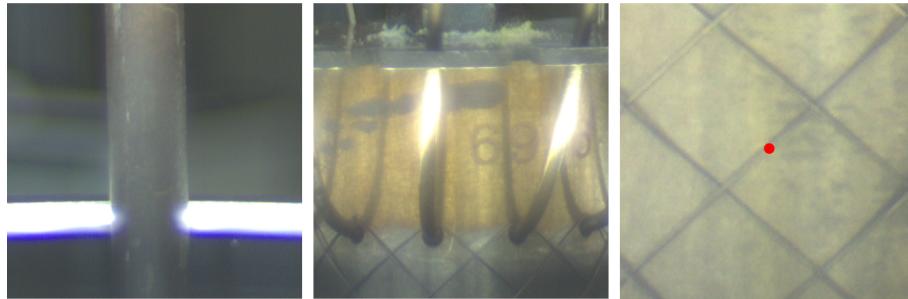


Abbildung 5.1: Beispiele für ungeeignete Bilder für das neuronale Netz: kein Stent im Bild sichtbar (links), kein Pick in der Mitte des Bildes (Mitte), kein eindeutiger Pick in der Mitte (rot markiert) des Bildes (rechts). Bilder aus [Pro] entnommen.



Abbildung 5.2: Weitere Beispiele für aussortierte Bilder: ohne Licht (links), starke Lichtbestrahlung (Mitte), Draht gerissen (rechts). Bilder aus [Pro] entnommen.

Gesamte Bilder	23.527
Ungeeignetes Licht	4.140
Draht gerissen	7.010
Kein Stent oder Pick sichtbar	7.586
Verwendbare Bilder	4.791

Tabelle 5.1: Aufteilung der ursprünglichen Aufnahmen in verwendbare und ungeeignete Bilder

5.2 Aufbereitung der Trainingsdaten

Wie bereits beschrieben, konnten 4.791 geeignete Bilder aus den ursprünglichen Daten entnommen werden. Um den Datensatz zu erweitern, wurden diese verwendbaren Bilder durch das Drehen um 180° augmentiert. Da sich dadurch der zuvor markierte Pick aus dem Mittelpunkt des Bildes verschieben kann, wurde das Skript zum Labeln der Bilder erweitert. Somit konnten zuvor falsch gesetzte Labels gelöscht und Labels, bei denen sich

der mittlere Pick nicht änderte für das augmentierte Bild übernommen werden. Hierdurch entstanden 3.892 weitere verwendbare Bilder inklusive Labels. Die Übersicht der Zusammensetzung des gesamten Datensatzes ist in Tabelle 5.2 dargestellt.

Datensatz	Anzahl Bilder
Vorgeschnittene Daten	4.791
Augmentierte Daten	3.892
Gesamter Datensatz	8.683

Tabelle 5.2: Übersicht über die vorgeschnittenen, augmentierten und den daraus entstandenen gesamten Daten

Eine detailliertere Darstellung vom Aufbau des Datensatzes wird durch Tabelle 5.3 aufgezeigt. Es ist ersichtlich, dass keiner der Flechtwinkel einen mehrheitlichen Anteil des Datensatzes darstellt, wodurch eine Gleichverteilung der Flechtwinkel in den unterschiedlichen Datensätzen gewährleistet wird.. Im gesamten Datensatz betragen die kleinste und größte Picklänge 40,0 und 219,06 Pixel. Zudem entspricht ein Millimeter 23 Pixel in den Bildern.

	30 Grad	40 Grad	50 Grad	60 Grad	70 Grad
Vorgeschnittene Bilder	880	1.142	988	878	903
Augmentierte Bilder	816	974	845	657	600
Gesamte Bilder	1696	2116	1833	1535	1503
Prozentanteil	19,5	24,4	21,1	17,7	17,3

Tabelle 5.3: Aufteilung der vorgeschnittenen und augmentierten Daten nach Flechtwinkel

5.2.1 Bildvorverarbeitung

Um die Erkennung gewisser Merkmale im Bild deutlicher zu machen, wurden die Bilder entsprechend vorverarbeitet. Hierfür wurde ein Histogrammausgleich durchgeführt, konkret der kontrastbegrenzte adaptive Histogrammausgleich aus Abschnitt 2.2.2. Die Implementierung wurde mit der Python-Bibliothek OpenCV [BK00] durchgeführt. Die Anwendung des gewöhnlichen Histogrammausgleichs, AHE und CLAHE aus Kapitel 2.2 wird in den Abbildungen 5.3, 5.4, 5.5 und 5.6 visualisiert.

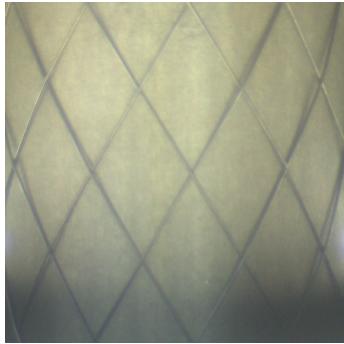


Abbildung 5.3: Ursprüngliches Bild

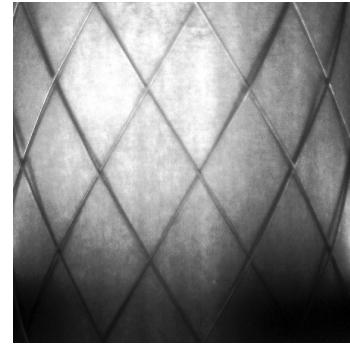


Abbildung 5.4: Histogrammausgleich

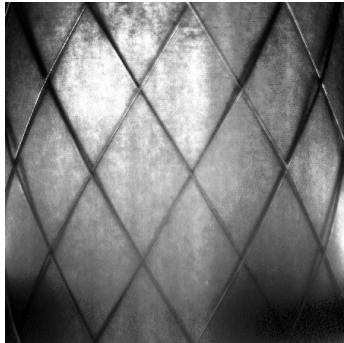


Abbildung 5.5: AHE

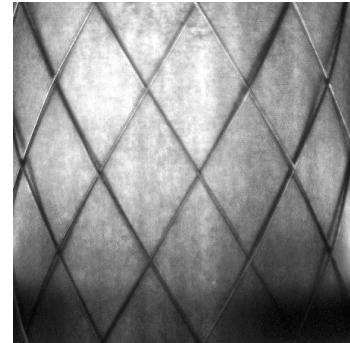


Abbildung 5.6: CLAHE

5.2.2 Aufteilen und Normalisieren der Daten

Wie in Tabelle 5.2 aufgezeigt, besteht der gesamte Datensatz aus 8.683 verwendbaren Bildern. Um die neuronalen Netze zu trainieren, werden ein Trainings- und ein Validierungsdatensatz benötigt. Zusätzlich ist ein Testdatensatz für die Evaluierung der einzelnen Netze notwendig. Aus diesem Grund wurde der Datensatz, wie in Tabelle 5.4 dargestellt, in drei kleinere Datensätze aufgeteilt. Dabei enthält der Trainingsdatensatz 70% der Daten. Die Datensätze für die Validierung und das Testen bestehen jeweils aus 15% der Bilder und Labels.

Datensatz	Anzahl Bilder
Training	6.078
Validierung	1.302
Test	1.303

Tabelle 5.4: Aufteilung des gesamten Datensatzes in drei kleinere Datensätze

Wie in [Jo19] wurden die Daten anhand einer sogenannten Min-Max-Normalization zwischen den Werten Null und Eins gesetzt. Der MinMax-Scaler wurde erst auf den Inhalt

des Trainingsdatesatzes angepasst und danach auf alle drei Datensätze angewendet. Um die Bilder und Labels aus diesen Datensätzen an die Netzwerkarchitekturen zu übergeben, wurde die Dataset API aus der Tensorflow-Bibliothek [ABC⁺16] verwendet, sodass eine Datenpipeline aufgebaut werden konnte.

Beim ersten Schritt wurden die Datensätze in TFRecord-Dateien umgewandelt. Dies hat den Vorteil, dass sie eine niedrigere Speicherkapazität benötigen und die Informationen der einzelnen Bilder zusammen mit ihren zugehörigen Labels hinterlegt werden. Zudem ist somit eine ununterbrochene Verbindung zum Laufwerk, auf dem die gesamten Daten gespeichert sind, nicht notwendig.

Um die Daten an den neuronalen Netzen zu übergeben, werden die einzelnen TFRecord-Dateien eingelesen. Mittels der Dataset API werden die eingelesenen Daten in einen Datensatz überführt. Danach werden die Bilder dekodiert und gecastet, um auf den Datentyp float32 umgewandelt zu werden. Die Werte der Labels werden ebenfalls auf den gleichen Datentyp umgewandelt. Im Anschluss wird der Datensatz aufgrund der Leistung des verwendeten Rechners in Batches der Größe acht aufgeteilt. Vorteil dieses Vorgehen ist, dass Bilder und Labels erst dekodiert und verarbeitet werden, wenn diese benötigt werden.

5.3 Implementierung der faltenden neuronalen Netze

Für die Implementierung der in Kapitel 4.3 angedeuteten künstlichen faltenden neuronalen Netzen wurde die Tensorflow-Bibliothek mit der Keras API [C⁺15] verwendet. Da mehrere der Modelle bereits in Keras eingebaut und auf dem ImageNet-Datensatz trainiert sind, konnten die vortrainierten Parameter übernommen werden. Dieser Vorgang wird als Transfer Learning gekennzeichnet und basiert auf der Prämisse, dass Menschen ihr vorher erlerntes Wissen anwenden können, um neue Probleme schneller und besser zu lösen; im Deep Learning funktioniert dies analog [PY09].

Als Aktivierungsfunktion wurde die von der Veröffentlichung des Netzes vordefinierte Funktion verwendet. In Fällen, bei denen diese nicht vorgegeben war, wurde die Exponential Linear Unit (ELU) angewendet, da diese laut [CUH15] nicht nur die Trainingszeit verkürzt, sondern die Leistung der Netze im Vergleich zur Anwendung von anderen Aktivierungsfunktionen wie ReLU verbessert. Zudem wurden alle drei in Kapitel 2.3.1.5 beschriebenen Methoden der Regularisierung angewendet: Early Stopping, Dropout und L1L2-Regularisierung. Bei der Anwendung des Dropouts wurde die Rate aus den Veröffentlichungen, sofern diese vorgegeben war, übernommen. In den meisten Fällen war diese nicht angegeben, weshalb der Wert auf 0,5 gesetzt wurde. Laut Srivastava et al. [SHK⁺14] ist diese Rate für eine breite Auswahl an Netzwerken und Aufgaben nahezu optimal. Für das Training der Netze wurde der Adam Optimierungsalgorithmus mit

einer Anfangslernrate von $1 \cdot 10^{-6}$ gewählt. Zudem wurde das Mean Squared Error als Fehlerfunktion gewählt.

5.4 Reproduzierbarkeit

Im Rahmen des wissenschaftlichen Arbeitens ist die Reproduzierbarkeit eines der wichtigsten Prinzipien. Damit ist die Wiederholbarkeit bzw. Wiedererzeugbarkeit bestimmter Ergebnisse unter gleichen Umständen gemeint. Um dies zu ermöglichen, wurde ein sogenannter Seed verwendet. Dies ist ein beliebiger Wert, welcher Zufallsoperationen innerhalb des Codes steuert. Somit werden bei jeder Ausführung die gleichen drei Datensätze erstellt.

Für das Training der neuronalen Netze wurden zu Beginn die Seeds zur den Numpy-, Tensorflow- und Random-Bibliotheken gesetzt. Des Weiteren wurde die Seed auch zur Initialisierung der Parameter im Netz verwendet. Da im Vorgang des Trainings nicht nur die CPU, sondern auch die GPU des Rechners verwendet wird, kann dies zu Abweichungen bei der Ausführung des Codes führen. Hierdurch werden nämlich gewisse Bibliotheken der GPU verwendet, wodurch weitere Quellen für Zufallswerte eingeführt werden, welche nicht gesteuert werden können.

5.5 Fehlerlokalisierung

Bezogen auf die Zielsetzung der Arbeit in Abschnitt 1.1, soll eine Lokalisierung eines erkannten Fehlers im Anschluss möglich sein. Da die Lokalisierung einzelner Picks im Stent ein breites Thema ist, wurde dies nur oberflächlich untersucht. Deshalb wurde nur die Bestimmung eines Fehlers anhand eines Toleranzwertes untersucht. Es wurde ein kleines Programm geschrieben, welches eine willkürliche Anzahl an Bilder inklusive Labels und einen beliebigen Toleranzwert in Pixel entgegennimmt. Demnach werden die einzelnen Ausgaben des Netzes mit den Labels verglichen, um den Wert der Abweichung zu berechnen. Ist dieser Wert größer als die gesetzte Toleranz, wird die Abweichung in Pixel und der Pfad zum entsprechenden Bild in einer Ausgabedatei eingetragen.

Kapitel 6

Bewertung und Vergleich der Ergebnisse

In diesem Kapitel werden die Ergebnisse des vorher erläuterten Systems und den unterschiedlichen Architekturen faltender neuronaler Netze analysiert. An erster Stelle wird auf den Vorgang des Trainings eines neuronalen Netzes eingegangen. Darauffolgend werden die Netzwerkarchitekturen miteinander verglichen. Hierfür wurden drei unterschiedliche Vergleichsparameter festgelegt, auf welche näher eingegangen wird. Zudem werden mögliche Fehlerquellen betrachtet. Im Anschluss des Kapitels wird das System bewertet.

6.1 Training

In der Regel sieht der Trainingsverlauf eines neuronalen Netzes wie in Abbildung 6.1 aus. Zu Beginn des Trainings liegt der Fehler auf einem hohen Wert, welcher innerhalb weniger Epochen stark gesenkt wird. Dabei werden die Verbesserungen im Verlauf immer kleiner, sodass diese Kurve abflacht und einen konvergenten Verlauf annimmt. Da sich der Fehler des Validierungsvorgangs ab einem bestimmten Zeitpunkt nicht mehr verbessert, wird das Training anhand des Early Stoppings angehalten. Die Anzahl an Epochen, die abgewartet werden, kann beliebig gewählt werden. Beispielsweise wurde dieser Wert im Verlauf des Trainings eines AlexNet, welcher in Abbildung 2.10 dargestellt ist, deutlich höher gesetzt, um die Auswirkungen des Overfittings zu zeigen.

Am logarithmischen Verlauf ist ebenfalls ersichtlich, dass das Netz bessere Ergebnisse in der Validierung als im Training erzielt. Dafür gibt es zwei Gründe. Zum einen wird das Dropout nicht im Validierungs- und Testvorgang angewendet, sondern nur während des Trainings. Zum anderen wird der Verlust im Training während den entsprechenden Epochen gemessen und aktualisiert. Im Gegensatz dazu wird der Validierungsfehler nach jeder Epoche berechnet, sodass die Parameter zu diesem Zeitpunkt auf angepassten Werten liegen.

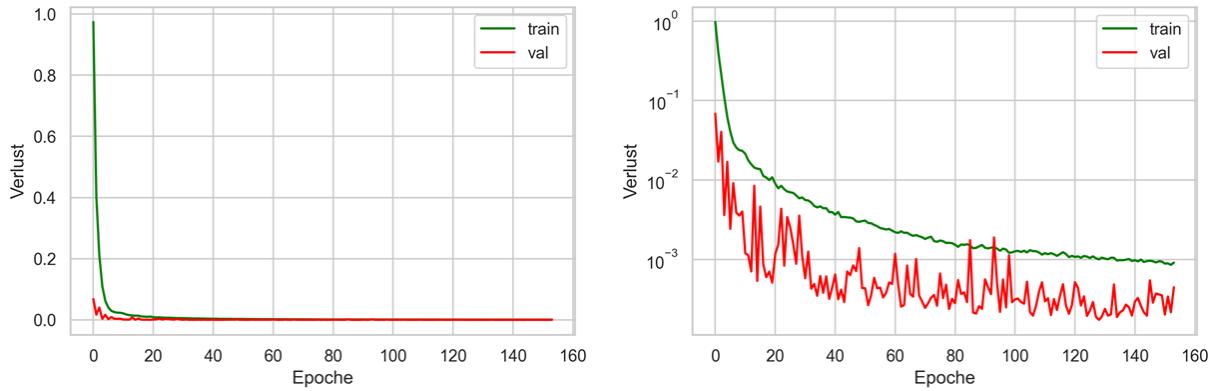


Abbildung 6.1: Verläufe des Trainingsvorgangs der VGG19-Architektur in linearer (links) und logarithmischer (rechts) Darstellung

Die Funktionsweise eines trainierten faltenden neuronalen Netzes kann mithilfe der Ausgaben aus den einzelnen Layern aufgezeigt werden. Ein vortrainiertes VGG19 erzeugt nach dem ersten Convolutional Layer 64 Feature Maps (siehe Abschnitt 2.4.2). Zehn die bei einer Eingabe des Bildes aus 5.3 entstandenen Feature Maps sind in Abbildung 6.2 abgebildet.

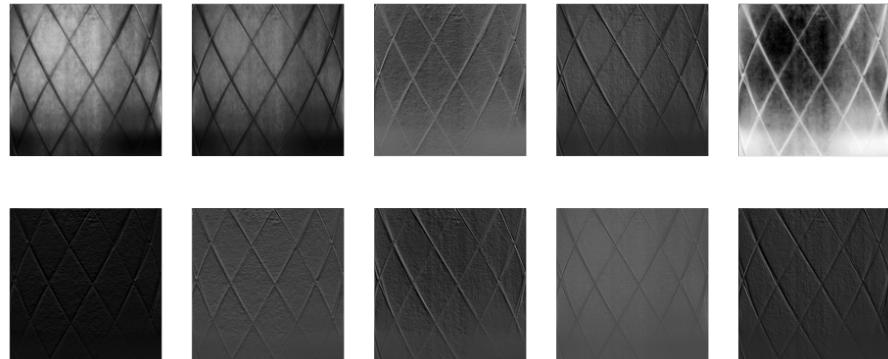


Abbildung 6.2: Zehn der 64 Feature Maps, welche durch das erste Convolutional Layer einer trainierten VGG19-Architektur entstehen

Durch die Anwendung der unterschiedlichen Filter aus dem Convolutional Layer werden die Kanten, also die Drähte des Stents, hervorgehoben. In den darauffolgenden Schritten wird die Dimensionalität der Eingabe durch die Anwendung weiterer Filter reduziert. Ebenso werden die Drähte im Bild verdeutlicht und der Hintergrund homogenisiert, wie in den Beispielen aus Abbildung 6.3 gezeigt.

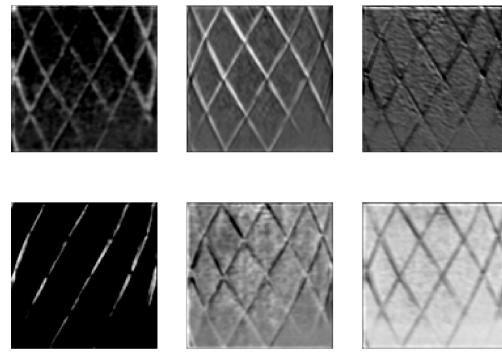


Abbildung 6.3: Sechs der 256 Feature Maps aus dem vierten und letzten Convolutional Layer im dritten Block des VGG19

6.2 Bewertung und Vergleich der Ergebnisse

Insgesamt wurden drei grundlegende Vergleichsfaktoren gewählt: Der Fehler, welcher die durchschnittliche Abweichung zwischen der Netzausgabe und dem Soll-Wert darstellt, die benötigte Rechenzeit pro Bild und die gesamte Dauer des Trainings der Netze. Im Folgenden werden die Trainings- und Testergebnisse der einzelnen Netze miteinander bezüglich den zuvor genannten Faktoren verglichen. Eine Übersicht mit allen faltenden neuronalen Netzen und den Vergleichsgrößen wird in Anhang A in Tabelle A.1 aufgezeigt.

6.2.1 Fehler

Der Fehler ist das wichtigste Vergleichskriterium unter den neuronalen Netzen, da es sich auf die Genauigkeit dieser bezieht. Um Fehler möglichst richtig zu erkennen, muss das Netzwerk die Länge des Picks möglichst zuverlässig bestimmen können.

Aus Abbildung 6.4 kann abgelesen werden, dass das VGG19 in dieser Kategorie das beste Ergebnis erreicht. Die durchschnittliche Abweichung von 1,76 Pixel entspricht etwa 0,077 mm. Zudem schneidet das AlexNet, die älteste Architektur aus der Auswahl, mit Abstand am schlechtesten ab. Die neueste Architektur, das EfficientNet B2, erreicht in diesem Fall eine durchschnittliche Leistung bei 3,24 Pixel.

6.2.2 Rechenzeit pro Bild

Als allgemeine Randbedingung wurde vorgegeben, dass das Gesamtsystem zur Fehlererkennung und -Korrektur in der Lage sein muss mindestens zehn Picks pro Sekunde zu inspizieren. Da das Bild durch alle Schichten der neuronalen Netze gespeist werden muss, hat die Anzahl an Schichten einen Einfluss auf die Dauer dieses Vorgangs. Abbildung

6.5 zeigt die benötigte Rechenzeit für ein Bild je Netz, wobei diese nach dem Veröffentlichungsjahr sortiert sind. Das AlexNet besitzt die niedrigste Anzahl an Schichten (sieben) und analysiert Bilder am schnellsten. Im Gegensatz dazu ist das Inception-ResNet-v2 das tiefste Netz und verbraucht die meiste Zeit zur Analyse. Trotz dieser Korrelation und des Einflusses der Anzahl an Schichten auf die Rechenzeit, sind beide Werte nicht proportional zueinander. Trotz der zweitniedrigsten Anzahl an Schichten verbraucht beispielsweise das VGG16 viel Zeit für die Analyse der Picklänge.

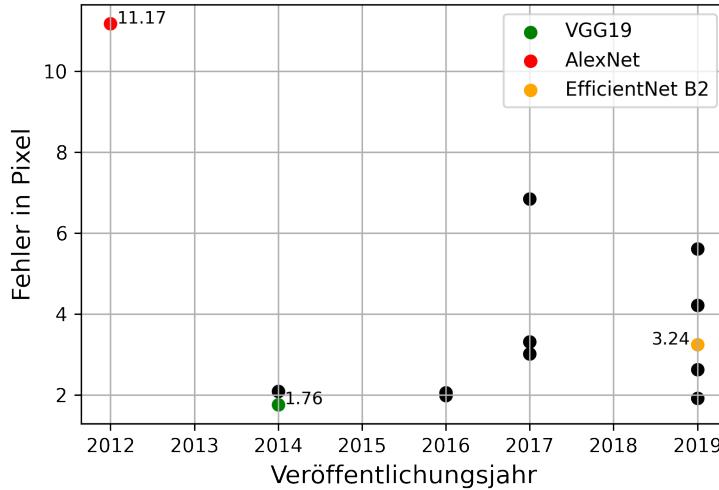


Abbildung 6.4: Vergleich des Fehlers nach Veröffentlichungsjahr der Netzwerkarchitekturen

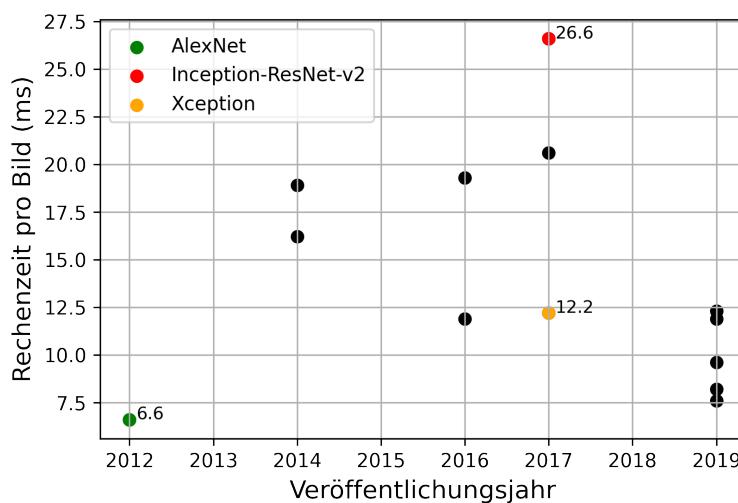


Abbildung 6.5: Vergleich der Rechenzeit pro Bild nach Veröffentlichungsjahr der Netzwerkarchitekturen

Nach Angaben des Industriepartners muss das Gesamtsystem zur Fehlererkennung und -korrektur in der Lage sein, zehn Picks innerhalb einer Sekunde zu inspizieren. Obwohl alle in diesem Abschnitt vorgestellten Werte unter dieser Schranke liegen, muss betrachtet werden, dass dies das Erfassen von Bildern, deren Vorverarbeitung, die Lokalisierung von Picks und weitere Schritte des Verfahrens nicht miteinbezieht.

6.2.3 Trainingsdauer

Der dritte und letzte Vergleichsfaktor unter den Netzen ist die benötigte Trainingsdauer. Allerdings hat diese keinen Einfluss auf die Leistung der neuronalen Netzwerke. Aufgrund seiner geringen Anzahl an Schichten schneidet hier die AlexNet-Architektur am besten ab. Unter den restlichen Modellen schwankt die Trainingsdauer zwischen zehn und zwanzig Stunden. Eine Übersicht der beim Fehler und Rechenzeit hervorgehobenen Netze, aus den Abschnitten 6.2.1 und 6.2.2, ist in Tabelle A.1 aufgezeigt.

6.2.4 Fehler zu Rechenzeit

Da der Fehler und die Rechenzeit die wichtigsten Faktoren sind, wurden diese ebenfalls, wie in Abbildung 6.6 zu sehen, miteinander verglichen. Anhand des Graphens ist ersichtlich, dass das VGG19 zwar den niedrigsten Fehler macht, sich dennoch unter den langsamsten Netzen bei der Vermessung der Picklänge befindet. Im Gegensatz dazu führt das MobileNetV3 large zu einer leicht höheren durchschnittlichen Abweichung von etwa 0,16 Pixel. Dafür wird die Länge eines Picks mehr als doppelt so schnell wie beim VGG19 vermessen. Bei der small Variante der MobileNetV3-Architektur beträgt diese Zeit etwas weniger als bei der large Variante. Allerdings ist der Unterschied beim erzeugten Fehler im Vergleich zum MobileNetV3 large deutlich größer.

6.3 Bestimmung der Abweichung

Zu Beginn des Flechtvorgangs sind sowohl der Soll-Wert l_{soll} für die Länge der Picks, als auch ein beliebiger Toleranzwert l_t angegeben. Darauffolgend wird, aus der Analyse eines Picks durch ein faltendes neuronales Netz, die vom System gemessene Picklänge ausgegeben, welche dem Ist-Wert l_{ist} entspricht. Somit kann die Abweichung l_{diff} der Netzausgabe durch den Betrag der Differenz zwischen dem Soll- und Ist-Wert wie in Gleichung 6.1 berechnet werden.

$$l_{diff} = |l_{soll} - l_{ist}| \quad (6.1)$$

Die Werte der Abweichungen eines Netzes können als Histogramm dargestellt werden.

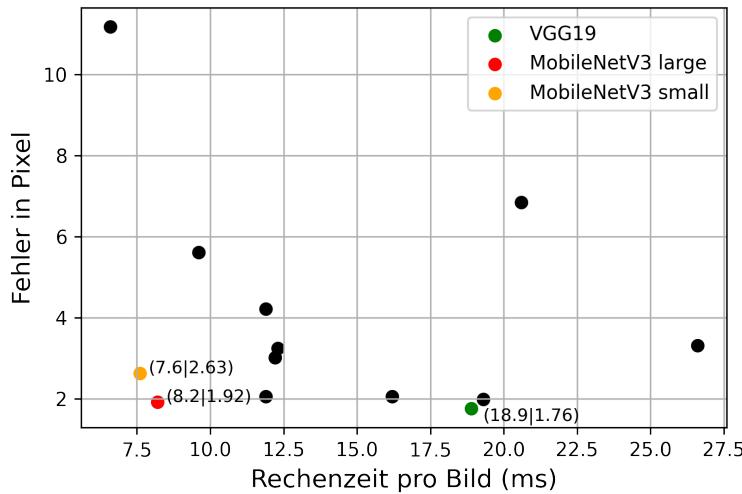


Abbildung 6.6: Vergleich des Fehlers nach der benötigten Rechenzeit pro Bild

Ein solches Diagramm ist in Abbildung 6.7 für das Testen einer trainierten VGG19-Architektur auf den Testdatensatz abgebildet. Für die meisten Bilder beträgt die Abweichung der Netzausgaben weniger als einen Pixel. Da es bei allen Netzen oftmals zu Ausgaben kommt, welche eine große Differenz aufweisen, wurde dies in Abschnitt 6.5 näher untersucht.

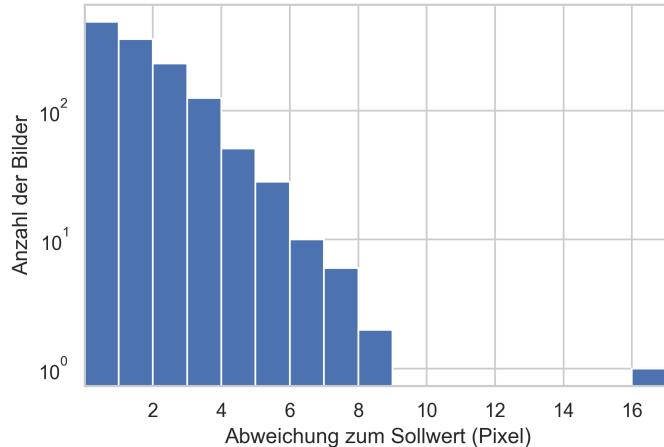


Abbildung 6.7: Aufteilung der Abweichungen bei den Ausgaben eines trainierten VGG19 auf den Testdatensatz

Werden die Abweichungen der fünf Architekturen mit dem besten Fehler näher betrachtet, wie in Abbildung 6.8 aufgezeigt, fällt auf, dass ein besserer Fehler nicht mit einem niedrigeren Intervall zwischen der besten und schlechtesten Vermessung zusammenhängt.

So liegen sowohl die größten, als auch die niedrigsten Abweichungen beim ResNet101V2 und VGG16 niedriger als bei neuronalen Netzen wie das VGG19 und MobileNetV3 large, welche einen besseren Fehler erzielen.

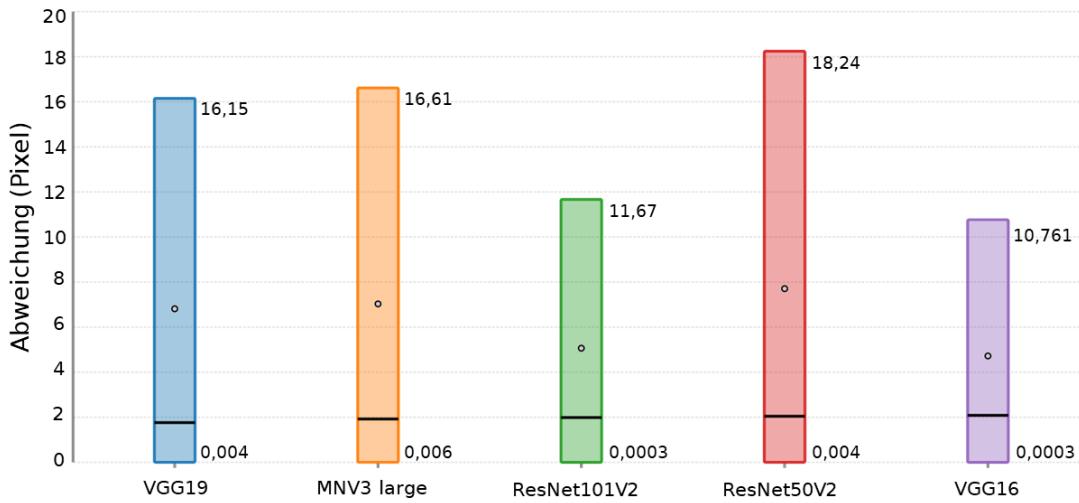


Abbildung 6.8: Aufteilung der Abweichungen unter den fünf Netzen mit dem besten Fehler

6.4 Fehlererkennung

Die errechnete Abweichung kann mit dem Toleranzwert verglichen werden, um über einen potentiellen Defekt im Stent zu entscheiden. Sollte diese kleiner oder gleich dem Wert der Toleranz l_t wie in Gleichung 6.2 sein, liegt diese noch im Toleranzbereich, sodass der Pick den Qualitätsanforderungen entspricht. Andernfalls entspricht die vom Netz gemessene Picklänge nicht den vorgegebenen Anforderungen, sodass der Pick als fehlerhaft empfunden wird.

$$l_{diff} \leq l_t \quad (6.2)$$

Je nachdem wie der Toleranzwert festgelegt wird, wird die Leistung des Systems beeinflusst. Ist dieser zu hoch, werden weniger Defekte gemeldet, wodurch das System weniger falsch negative Ergebnisse anzeigt und die Produktivität gesteigert wird. Auf der anderen Seite können Defekte dadurch übersehen werden, womit die Genauigkeit und Zuverlässigkeit der Überprüfung sinkt.

6.5 Mögliche Fehlerquellen

Im Rahmen dieses Abschnitts sollen Gründe für Abweichungen zwischen den Soll- und Ist-Werten erklärt werden. Für Abweichungen einer hohen Größenordnung wurden zwei

Gründe festgestellt. Zum einen sind Teile des mittleren Picks auf einigen Bildern schlecht erkennbar. Im betrachteten Pick aus Abbildung 6.9 links, ist die obere Ecke der Struktur schlecht erkennbar. Aus diesem Bild ergab sich die größte Abweichung mit dem VGG19, nämlich 16,2 Pixel.

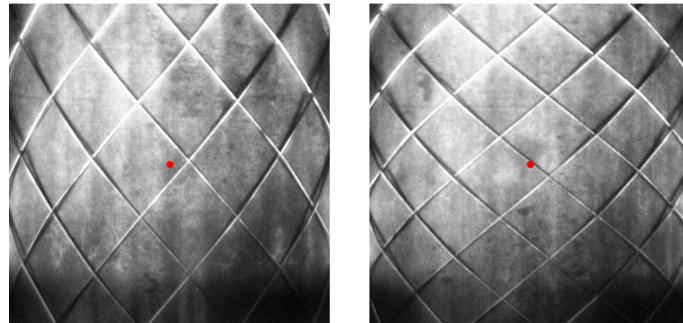


Abbildung 6.9: Beispiele für Bilder, welche zu großen Abweichungen bei der Netzausgabe geführt haben. Der Mittelpunkt der Bilder ist rot markiert.

Der zweite und meist verbreitete Grund ist rechts in Abbildung 6.9 dargestellt. Ein zwischen zwei Picks positionierter Mittelpunkt des Bildes führte oftmals zu großen Abweichungen bei den Netzen.

Abweichungen einer geringen Größe können aufgrund ungenauer Labels entstehen. In Anbetracht der breiten Anzahl an Bildern, von denen die Labels in Zusammenarbeit erstellt wurden, und dem unvermeidbaren menschlichen Fehler, sind Ungenauigkeiten in den Labels unumgänglich. Somit ist es möglich, dass die Netze zwar die richtigen Eigenschaften der Bilder erlernen, sich aber Abweichungen infolge ungenauer Labels bilden.

6.6 Bewertung des Systems

Bezogen auf das Projekt Stents4Tomorrow [Pro] sind drei wichtige Größen für die Genauigkeit des Systems zur automatisierten Korrektur von Fehlern in Stents vorgegeben. Ein optimales System muss in der Lage sein, Picklängen mit einer Genauigkeit von $\pm 0,01$ Millimeter zu berechnen. Bei einer guten Leistung verdoppelt sich dieser Wert und ein ausreichendes Ergebnis bedingt eine Genauigkeit von 0,05 Millimeter. Diese Werte werden als l_1 , l_2 und l_3 gekennzeichnet.

Abbildung 6.10 zeigt den prozentualen Anteil an Bildern aus dem Testdatensatz, welche von unterschiedlichen Netzen bei einem gewissen Toleranzwert richtig vermessen werden. Hierbei sind die drei zuvor erläuterten Vorgaben ebenso im Graphen eingetragen. Hieraus kann abgelesen werden, dass die zwei Netze mit dem besten Fehler, das VGG19 und MobileNetV3, ab einer Toleranz von etwa sieben Pixeln nahe an die 100% kommen. Die

aus der Auswahl neueste Architektur, das EfficientNet B2, benötigt etwa 11,5 Pixel. Das AlexNet hingegen erreicht, trotz einer Toleranz von 12 Pixeln, was mehr als einen halben Millimeter entspricht, eine Genauigkeit von ungefähr 65%.

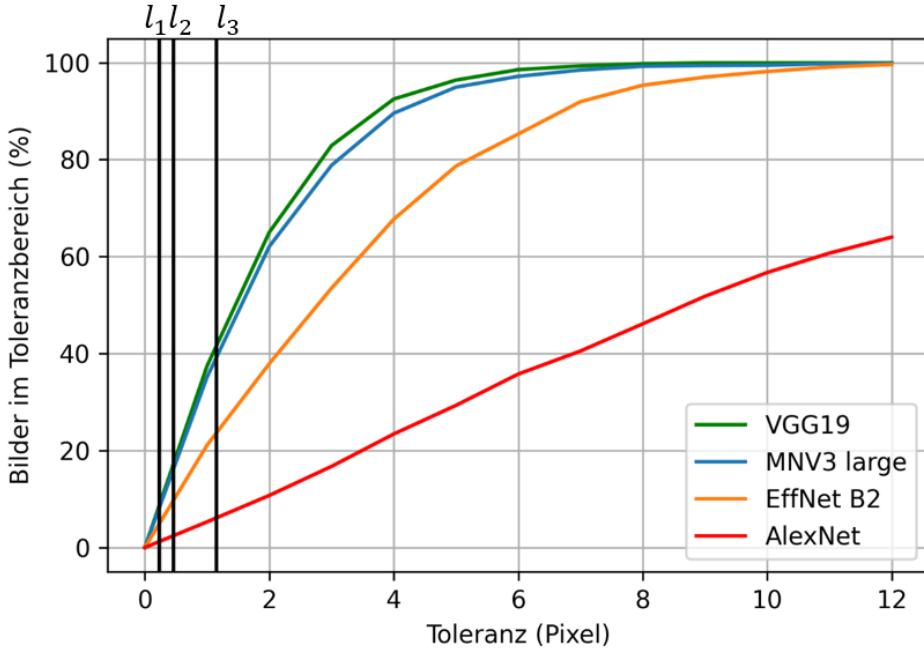


Abbildung 6.10: Prozentanteil der Ausgaben mehrerer Netze je nach gesetzter Toleranz. Die drei vorgegebenen Größen $l_1 = 0,23$ Pixel, $l_2 = 0,46$ Pixel und $l_3 = 1,15$ Pixel sind eingezeichnet.

Werden diese Ergebnisse nun mit den Zielwerten verglichen, wird deutlich, dass die l_1 - und l_2 -Werte aktuell noch nicht erreicht werden können. Die Akkurate des aktuell genauesten Netzes liegt hinsichtlich dieser Werte bei etwa 8,1% und 16,8%. Selbst eine ausreichend gute Vermessung der Picklänge kann aktuell nur in ungefähr 43,3% der Fälle erreicht werden.

Kapitel 7

Fazit und Ausblick

In dieser Arbeit wurden neun Architekturen und einige ihrer Varianten, also dreizehn faltende neuronale Netze bezüglich einer Regression der Picklänge kardiovaskulärer Implantate untersucht. Hierfür wurden zur Verfügung gestellte Daten vorverarbeitet und augmentiert. Darauffolgend wurden die einzelnen neuronalen Netze implementiert, angepasst, trainiert und daraufhin evaluiert. Basierend auf den Evaluationsergebnissen, wurden die Netze hinsichtlich einer Regression der Picklänge der Implantate miteinander verglichen.

Aus der Bewertung der neuronalen Netze und dem anschließenden Vergleich hat das VGG19 die besten Ergebnisse in Bezug auf die durchschnittliche Abweichung erzielt. Diese beträgt 1,76 Pixel oder 0,077 Millimeter, sodass der Fehler bei der kleinsten und der größten Picklänge des Datensatzes 4,4 % bzw. 0,8 % beträgt. Überdies beträgt der Fehler des MobileNetV3 large 0,16 Pixel mehr, doch die erforderliche Rechenzeit, um die Länge eines Picks zu berechnen, wird mehr als halbiert. Hinsichtlich einer zuverlässigen und zugleich zeitlich effizienten Überprüfung der Stentgeometrie kann dieser Faktor im weiteren Verlauf des Projekts von großer Bedeutung sein.

Die in Abschnitt 6.6 erläuterten Zielanforderungen an das System sind unter den aktuellen Umständen nicht erfüllbar. Gründe dafür sind der menschliche Fehler bei der Erstellung der Labels für die Bilddaten, die verfügbare Rechenleistung und die Auflösung der aktuellen zugeschnittenen Aufnahmen.

Im nächsten Schritt des Projektes ist die Durchführung einer Hyperparameteroptimierung mit den besten faltenden neuronalen Netzen sinnvoll. Daraus können die Ergebnisse dieser Arbeit weiterhin verbessert werden. Aufgrund des verfügbaren Rechners und der begrenzten Zeit für dessen Verwendung war eine solche Optimierung der Hyperparameter nicht möglich.

Außerdem kann der Datensatz mit weiteren Flechtwinkeln erweitert und diversifiziert wer-

den, da dieser aktuell aus Bildern mit nur fünf unterschiedlichen Flechtwinkeln besteht. Dadurch bilden sich unter den Ausgaben der trainierten Netzwerkarchitekturen fünf unterschiedliche Cluster.

Zudem können Bilder mit unterschiedlichen Lichtverhältnissen und Hintergründen hinzugefügt werden, um die Fähigkeit der Generalisierung von den neuronalen Netzen zu steigern und somit deren Leistung zu verbessern, sodass die Längen der einzelnen Picks genauer und zuverlässiger bestimmt werden können. Für die Umsetzung der angesetzten Ideen bezüglich des Datensatzes kann die Erzeugung künstlicher Bilddaten inklusive Labels untersucht werden. Zu diesem Zweck können beispielsweise Generative Adversarial Networks [GPAM⁺14], Generative Teaching Networks [SRL⁺20] oder weitere Verfahren erprobt werden.

Literaturverzeichnis

- [AA18] AMIDI, Afshine ; AMIDI, Shervine: *Convolutional Neural Networks Cheat Sheet*. 2018
- [ABC⁺16] ABADI, Martín ; BARHAM, Paul ; CHEN, Jianmin ; CHEN, Zhifeng ; DAVIS, Andy ; DEAN, Jeffrey ; DEVIN, Matthieu ; GHEMAWAT, Sanjay ; IRVING, Geoffrey ; ISARD, Michael u. a.: Tensorflow: A system for large-scale machine learning. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, S. 265–283
- [Alp12] ALPYILDIZ, Tuba: 3D geometrical modelling of tubular braids. In: *Textile research journal* 82 (2012), Nr. 5, S. 443–453
- [AMAZ17] ALBAWI, Saad ; MOHAMMED, Tareq A. ; AL-ZAWI, Saad: Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)* Ieee, 2017, S. 1–6
- [ATY⁺18] ALOM, Md Z. ; TAHA, Tarek M. ; YAKOPCIC, Christopher ; WESTBERG, Stefan ; SIDIKE, Paheding ; NASRIN, Mst S. ; VAN ESESN, Brian C. ; AW-WAL, Abdul A S. ; ASARI, Vijayan K.: The history began from alexnet: A comprehensive survey on deep learning approaches. In: *arXiv preprint arXiv:1803.01164* (2018)
- [BK00] BRADSKI, Gary ; KAEHLER, Adrian: OpenCV. In: *Dr. Dobb's journal of software tools* 3 (2000)
- [BLC⁺17a] BERMUDEZ, Carlos ; LAGUARTA, Ferran ; CADEVALL, Cristina ; MATILLA, Aitor ; IBÁÑEZ, Sergi ; ARTIGAS, Roger: Automated stent defect detection and classification with a high numerical aperture optical system. In: *Automated Visual Inspection and Machine Vision II* Bd. 10334 International Society for Optics and Photonics, 2017, S. 103340C
- [BLC⁺17b] BERMUDEZ, Carlos ; LAGUARTA, Ferran ; CADEVALL, Cristina ; MATILLA, Aitor ; IBÁÑEZ, Sergi ; ARTIGAS, Roger: Optical stent inspection of surface

- texture and coating thickness. In: *Photonic Instrumentation Engineering IV* Bd. 10110 International Society for Optics and Photonics, 2017, S. 1011007
- [BLC⁺17c] BERMUDEZ, Carlos ; LAGUARTA, Ferran ; CADEVALL, Cristina ; MATILLA, Aitor ; IBAÑEZ, Sergi ; ARTIGAS, Roger: Stent optical inspection system calibration and performance. In: *Applied optics* 56 (2017), Nr. 9, S. D134–D141
- [BLF16] BEYERER, Jürgen ; LEÓN, Fernando P. ; FRESE, Christian: *Automatische Sichtprüfung: Grundlagen, Methoden und Praxis der Bildgewinnung und Bildauswertung*. Springer-Verlag, 2016
- [C⁺15] CHOLLET, François u. a.: *keras*. 2015
- [Cam] CAMBRIDGE DICTIONARY: *deep learning*. – <https://dictionary.cambridge.org/de/worterbuch/englisch/deep-learning> (Abgerufen am: 14. Februar 2021, 19:22)
- [Can86] CANNY, John: A computational approach to edge detection. In: *IEEE Transactions on pattern analysis and machine intelligence* (1986), Nr. 6, S. 679–698
- [Cho17] CHOLLET, François: Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, S. 1251–1258
- [CMM⁺11] CIRESAN, Dan C. ; MEIER, Ueli ; MASCI, Jonathan ; GAMBARDELLA, Luca M. ; SCHMIDHUBER, Jürgen: Flexible, high performance convolutional neural networks for image classification. In: *Twenty-second international joint conference on artificial intelligence*, 2011
- [CUH15] CLEVERT, Djork-Arné ; UNTERTHINER, Thomas ; HOCHREITER, Sepp: Fast and accurate deep network learning by exponential linear units (elus). In: *arXiv preprint arXiv:1511.07289* (2015)
- [DH72] DUDA, Richard O. ; HART, Peter E.: Use of the Hough transformation to detect lines and curves in pictures. In: *Communications of the ACM* 15 (1972), Nr. 1, S. 11–15
- [DHS11] DUCHI, John ; HAZAN, Elad ; SINGER, Yoram: Adaptive subgradient methods for online learning and stochastic optimization. In: *Journal of machine learning research* 12 (2011), Nr. 7

- [Dja20] DJAMAL, Aulia J.: *Conceptual Comparison and Evaluation of Various Strategies for the Automated Correction of Errors in the Braiding Process of Cardiovascular Implants Based on Camera Images and Deep Learning*, Karlsruhe Institute of Technology, Bachelor's Thesis, 2020. – Institut für Technik der Informationsverarbeitung
- [DN20] DERU, Matthieu ; NDIAYE, Alassane: *Deep Learning mit TensorFlow, Keras und TensorFlow.js*. Rheinwerk Verlag, 2020
- [EK19] EDDINE KHATRCHI, Houssem: *Intelligente visuelle Inspektion von Flechtmustern durch maschinelles Lernen zur Flechterkennung*, Karlsruhe Institute of Technology, Master's Thesis, 2019. – Institut für Technik der Informationsverarbeitung
- [ELJ20] ELIBOL, Melih ; LEI, Lihua ; JORDAN, Michael I.: Variance reduction with sparse gradients. In: *arXiv preprint arXiv:2001.09623* (2020)
- [Far05] FARMER, Jason: *Automated Vision-Based Inspection System for Stents*, Worcester Polytechnic Institute, Diss., 2005
- [GBCB16] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron ; BENGIO, Yoshua: *Deep learning*. MIT press Cambridge, 2016
- [GC18] GUERRA, Antonio J. ; CIURANA, Joaquim: *Stent's Manufacturing Field: Past, Present, and Future Prospects*. IntechOpen, 2018
- [GGH13] GUYADER, G ; GABOR, A ; HAMELIN, P: Analysis of 2D and 3D circular braiding processes: Modeling the interaction between the process parameters and the pre-form architecture. In: *Mechanism and Machine Theory* 69 (2013), S. 90–104
- [GPAM⁺14] GOODFELLOW, Ian J. ; POUGET-ABADIE, Jean ; MIRZA, Mehdi ; XU, Bing ; WARDE-FARLEY, David ; OZAIR, Sherjil ; COURVILLE, Aaron ; BENGIO, Yoshua: Generative adversarial networks. In: *arXiv preprint arXiv:1406.2661* (2014)
- [Gro11] GROOT, Peter de: Coherence scanning interferometry. In: *Optical measurement of surface topography*. Springer, 2011, S. 187–208
- [GWK⁺18] GU, Jiuxiang ; WANG, Zhenhua ; KUEN, Jason ; MA, Liyang ; SHAHROUDY, Amir ; SHUAI, Bing ; LIU, Ting ; WANG, Xingxing ; WANG, Gang ; CAI, Jianfei u. a.: Recent advances in convolutional neural networks. In: *Pattern Recognition* 77 (2018), S. 354–377

- [HC19] HUNT, Alexander J. ; CAREY, Jason P.: A machine vision system for the braid angle measurement of tubular braided structures. In: *Textile Research Journal* 89 (2019), Nr. 14, S. 2919–2937
- [HSC⁺19] HOWARD, Andrew ; SANDLER, Mark ; CHU, Grace ; CHEN, Liang-Chieh ; CHEN, Bo ; TAN, Mingxing ; WANG, Weijun ; ZHU, Yukun ; PANG, Ruoming ; VASUDEVAN, Vijay u. a.: Searching for mobilenetv3. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, S. 1314–1324
- [HSS12] HINTON, Geoffrey ; SRIVASTAVA, Nitish ; SWERSKY, Kevin: Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. In: *Cited on* 14 (2012), Nr. 8
- [HSS18] HU, Jie ; SHEN, Li ; SUN, Gang: Squeeze-and-excitation networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, S. 7132–7141
- [HTF09] HASTIE, Trevor ; TIBSHIRANI, Robert ; FRIEDMAN, Jerome: *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009
- [HW68] HUBEL, David H. ; WIESEL, Torsten N.: Receptive fields and functional architecture of monkey striate cortex. In: *The Journal of physiology* 195 (1968), Nr. 1, S. 215–243
- [HZC⁺17] HOWARD, Andrew G. ; ZHU, Menglong ; CHEN, Bo ; KALENICHENKO, Dmitry ; WANG, Weijun ; WEYAND, Tobias ; ANDREETTO, Marco ; ADAM, Hartwig: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: *arXiv preprint arXiv:1704.04861* (2017)
- [HZRS16a] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 770–778
- [HZRS16b] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing ; SUN, Jian: Identity mappings in deep residual networks. In: *European conference on computer vision* Springer, 2016, S. 630–645
- [IB09] IBRAHEEM, Issa ; BINDER, Alfred: Automated Inspection System of Stent. In: *4th European Conference of the International Federation for Medical and Biological Engineering* Springer, 2009, S. 952–957

- [ima] *Image Classification on ImageNet.* – <https://paperswithcode.com/sota/image-classification-on-imagenet> (Abgerufen am 31. März 2021, 21:22)
- [IS15] IOFFE, Sergey ; SZEGEDY, Christian: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning* PMLR, 2015, S. 448–456
- [Jo19] JO, Jun-Mo: Effectiveness of normalization pre-processing of big data to the machine learning performance. In: *The Journal of the Korea institute of electronic communication sciences* 14 (2019), Nr. 3, S. 547–552
- [KB14] KINGMA, Diederik P. ; BA, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014)
- [KB20] KHAN-BLOUKI, Valentin: *Konzeptionierung eines kamerabasierten Deep Learning Systems zur automatisierten Vermessung kardiovaskulärer Implantate mit dem Ziel einer Korrektur des Flechtprozesses*, Karlsruhe Institute of Technology, Bachelor's Thesis, 2020. – Institut für Technik der Informationsverarbeitung
- [KH⁺09] KRIZHEVSKY, Alex ; HINTON, Geoffrey u. a.: Learning multiple layers of features from tiny images. (2009)
- [KSH12] KRIZHEVSKY, Alex ; SUTSKEVER, Ilya ; HINTON, Geoffrey E.: Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems* 25 (2012), S. 1097–1105
- [Kyo14] KYOSEV, Yordan: *Braiding technology for textiles: Principles, design and processes*. Elsevier, 2014
- [LBBH98] LECUN, Yann ; BOTTOU, Léon ; BENGIO, Yoshua ; HAFFNER, Patrick: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324
- [LBD⁺90] LECUN, Yann ; BOSER, Bernhard E. ; DENKER, John S. ; HENDERSON, Donnie ; HOWARD, Richard E. ; HUBBARD, Wayne E. ; JACKEL, Lawrence D.: Handwritten digit recognition with a back-propagation network. In: *Advances in neural information processing systems*, 1990, S. 396–404
- [LK19] LIU, YS ; KYOSEV, Yordan: Automatic analysis the braiding angle of the braided fabrics using image processing. In: *Fibres and Textiles* 26 (2019), S. 63–68

- [LMAPH19] LATHUILIÈRE, Stéphane ; MESEJO, Pablo ; ALAMEDA-PINEDA, Xavier ; HORAUD, Radu: A comprehensive analysis of deep regression. In: *IEEE transactions on pattern analysis and machine intelligence* 42 (2019), Nr. 9, S. 2065–2081
- [Mei] MEIER, Dr. O.: *Stent.* – <https://www.qualitaetskliniken.de/behandlungen/stent/> (Abgerufen am: 05. Februar 2021, 13:58)
- [NIGM18] NWANKPA, Chigozie ; IJOMAH, Winifred ; GACHAGAN, Anthony ; MARS-HALL, Stephen: Activation functions: Comparison of trends in practice and research for deep learning. In: *arXiv preprint arXiv:1811.03378* (2018)
- [ON15] O’SHEA, Keiron ; NASH, Ryan: An introduction to convolutional neural networks. In: *arXiv preprint arXiv:1511.08458* (2015)
- [PAA⁺87] PIZER, Stephen M. ; AMBURN, E P. ; AUSTIN, John D. ; CROMARTIE, Robert ; GESELOWITZ, Ari ; GREER, Trey ; HAAR ROMENY, Bart ter ; ZIMMERMAN, John B. ; ZUIDERVELD, Karel: Adaptive histogram equalization and its variations. In: *Computer vision, graphics, and image processing* 39 (1987), Nr. 3, S. 355–368
- [Pho78] PHOENIX, SL: Mechanical response of a tubular braided cable with an elastic core. In: *Textile Research Journal* 48 (1978), Nr. 2, S. 81–91
- [Pro] PROJEKTKOORDINATOR: MARC BRÄUNER, ADMEDES GMBH: *Stents4Tomorrow.* <https://www.zukunft-der-wertschoepfung.de/de/projekte.php?PN=11051057> (Abgerufen am: 13. April 2021, 09:47),
- [PXG⁺19] PEI, Lei ; XIAO, Zhitao ; GENG, Lei ; WU, Jun ; ZHANG, Fang ; SUN, Ying: Surface parameters measurement for braided composite preform based on gray projection. In: *Journal of Engineered Fibers and Fabrics* 14 (2019), S. 1558925019887621
- [PY09] PAN, Sinno J. ; YANG, Qiang: A survey on transfer learning. In: *IEEE Transactions on knowledge and data engineering* 22 (2009), Nr. 10, S. 1345–1359
- [RHW85] RUMELHART, David E. ; HINTON, Geoffrey E. ; WILLIAMS, Ronald J.: Learning internal representations by error propagation / California Univ San Diego La Jolla Inst for Cognitive Science. 1985. – Forschungsbericht
- [Ros57] ROSENBLATT, Frank: *The perceptron, a perceiving and recognizing automation Project Para.* Cornell Aeronautical Laboratory, 1957

- [RR19] RAHANGDALE, Ashwini ; RAUT, Shital: Deep neural network regularization for feature selection in learning-to-rank. In: *IEEE Access* 7 (2019), S. 53988–54006
- [RZL17] RAMACHANDRAN, Prajit ; ZOPH, Barret ; LE, Quoc V.: Searching for activation functions. In: *arXiv preprint arXiv:1710.05941* (2017)
- [Sch20] SCHORLE, Felix: *Automatische Sichtprüfung von kardiovaskulären Implantaten mit Neuronalen Netzen*, Karlsruhe Institute of Technology, Bachelor's Thesis, 2020. – Institut für Technik der Informationsverarbeitung
- [SHK⁺14] SRIVASTAVA, Nitish ; HINTON, Geoffrey ; KRIZHEVSKY, Alex ; SUTSKEVER, Ilya ; SALAKHUTDINOV, Ruslan: Dropout: a simple way to prevent neural networks from overfitting. In: *The journal of machine learning research* 15 (2014), Nr. 1, S. 1929–1958
- [SHZ⁺18] SANDLER, Mark ; HOWARD, Andrew ; ZHU, Menglong ; ZHMOGINOV, Andrey ; CHEN, Liang-Chieh: Mobilenetv2: Inverted residuals and linear bottlenecks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, S. 4510–4520
- [Sig17] SIGWART, Ulrich: The Stent Story: how it all started.... In: *European heart journal* 38 (2017), Nr. 28, S. 2171–2172
- [SIVA17] SZEGEDY, Christian ; IOFFE, Sergey ; VANHOUCKE, Vincent ; ALEMI, Alexander: Inception-v4, inception-resnet and the impact of residual connections on learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence* Bd. 31, 2017
- [SL19] SALMAN, Shaeke ; LIU, Xiuwen: Overfitting mechanism and avoidance in deep neural networks. In: *arXiv preprint arXiv:1901.06566* (2019)
- [SLJ⁺15] SZEGEDY, Christian ; LIU, Wei ; JIA, Yangqing ; SERMANET, Pierre ; REED, Scott ; ANGUELOV, Dragomir ; ERHAN, Dumitru ; VANHOUCKE, Vincent ; RABINOVICH, Andrew: Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, S. 1–9
- [SRL⁺20] SUCH, Felipe P. ; RAWAL, Aditya ; LEHMAN, Joel ; STANLEY, Kenneth ; CLUNE, Jeffrey: Generative teaching networks: Accelerating neural architecture search by learning to generate synthetic training data. In: *International Conference on Machine Learning* PMLR, 2020, S. 9206–9216

- [Staa] STATISTISCHES BUNDESAMT: *Anzahl der Gestorbenen nach Kapiteln der ICD-10 und nach Geschlecht für 2019.* – https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Gesundheit/Todesursachen/Tabellen/gestorbene_anzahl.html (Abgerufen am: 5. März 2021, 10:12)
- [Stab] STATISTISCHES BUNDESAMT: *Sterbefälle durch Herz-Kreislauf-Erkrankungen insgesamt 2019.* – <https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Gesundheit/Todesursachen/Tabellen/sterbefaelle-herz-kreislauf-erkrankungen-insgesamt.html> (Abgerufen am: 5. März 2021, 10:46)
- [SVI⁺16] SZEGEDY, Christian ; VANHOUCKE, Vincent ; IOFFE, Sergey ; SHLENS, Jon ; WOJNA, Zbigniew: Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, S. 2818–2826
- [SW11] SAMMUT, Claude ; WEBB, Geoffrey I.: *Encyclopedia of machine learning*. Springer Science & Business Media, 2011
- [SZ14] SIMONYAN, Karen ; ZISSERMAN, Andrew: Very deep convolutional networks for large-scale image recognition. In: *arXiv preprint arXiv:1409.1556* (2014)
- [Tak17] TAKI, Masato: Deep residual networks and weight initialization. In: *arXiv preprint arXiv:1709.02956* (2017)
- [TL19] TAN, Mingxing ; LE, Quoc: Efficientnet: Rethinking model scaling for convolutional neural networks. In: *International Conference on Machine Learning* PMLR, 2019, S. 6105–6114
- [XPG⁺20] XIAO, Zhitao ; PEI, Lei ; GENG, Lei ; SUN, Ying ; ZHANG, Fang ; WU, Jun: Surface Parameter Measurement of Braided Composite Preform Based on Faster R-CNN. In: *Fibers and Polymers* 21 (2020), Nr. 3, S. 590–603
- [XPZ⁺18] XIAO, Zhitao ; PEI, Lei ; ZHANG, Fang ; GENG, Lei ; WU, Jun ; TONG, Jun ; XI, Jiangtao ; OGUNBONA, Philip O.: Measurement of surface parameters of three-dimensional braided composite preform based on curvature scale space corner detector. In: *Textile Research Journal* 88 (2018), Nr. 23, S. 2641–2653
- [XPZ⁺19] XIAO, Zhitao ; PEI, Lei ; ZHANG, Fang ; SUN, Ying ; GENG, Lei ; WU, Jun ; TONG, Jun ; WEN, Jia: Surface parameters measurement of braided preform based on local edge extreme. In: *The Journal of The Textile Institute* 110 (2019), Nr. 4, S. 535–542

- [YMA14] YADAV, Garima ; MAHESHWARI, Saurabh ; AGARWAL, Anjali: Contrast limited adaptive histogram equalization based enhancement for real time video system. In: *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* IEEE, 2014, S. 2392–2397
- [YNDT18] YAMASHITA, Rikiya ; NISHIO, Mizuho ; DO, Richard Kin G. ; TOGASHI, Kaori: Convolutional neural networks: an overview and application in radiology. In: *Insights into imaging* 9 (2018), Nr. 4, S. 611–629
- [ZBB99] ZHANG, Q ; BEALE, D ; BROUGHTON, RM: Analysis of circular braiding process, Part 1: theoretical investigation of kinematics of the circular braiding process. (1999)
- [ZF14] ZEILER, Matthew D. ; FERGUS, Rob: Visualizing and understanding convolutional networks. In: *European conference on computer vision* Springer, 2014, S. 818–833

Anhang A

Anhang

Architektur	Fehler (Pixel)	Rechenzeit (ms)	Trainingsdauer
VGG19	1,764	18,9	15h 48m 28s
MobileNetV3 large	1,924	8,2	14h 5m 24s
ResNet101V2	1,988	19,3	18h 15m 18s
ResNet50V2	2,045	11,9	13h 24m 31s
VGG16	2,085	16,2	8h 40m 55s
MobileNetV3 small	2,634	7,6	22h 4m 32s
Xception	3,020	12,2	12h 30m 33s
EfficientNet B2	3,242	12,3	16h 56m 49s
Inception-ResNet-v2	3,311	26,6	20h 32m 46s
EfficientNet B1	4,206	11,9	10h 52m 11s
EfficientNet B0	5,612	9,6	7h 26m 50s
Inception-v4	6,840	20,6	12h 7m 27s
AlexNet	11,167	6,6	4h 55m 54s

Tabelle A.1: Vergleich der Trainings- und Testergebnisse aller Architekturen. Betrachtet wird der Fehler, welcher die durchschnittliche Abweichung zwischen der Netzausgabe und dem Soll-Wert darstellt, die Rechenzeit pro Bild und die gesamte Trainingsdauer der einzelnen Netze.

Tabellenverzeichnis

5.1	Aufteilung der ursprünglichen Aufnahmen in verwendbare und ungeeignete Bilder	36
5.2	Übersicht über die vorgeschnittenen, augmentierten und den daraus entstandenen gesamten Daten	37
5.3	Aufteilung der vorgeschnittenen und augmentierten Daten nach Flechtwinkel	37
5.4	Aufteilung des gesamten Datensatzes in drei kleinere Datensätze	38
A.1	Vergleich der Trainings- und Testergebnisse aller Architekturen. Betrachtet wird der Fehler, welcher die durchschnittliche Abweichung zwischen der Netzausgabe und dem Soll-Wert darstellt, die Rechenzeit pro Bild und die gesamte Trainingsdauer der einzelnen Netze.	63

Abbildungsverzeichnis

2.1	Der erste selbstexpandierbare Stent. Oben am Abgabekatheter gebunden. Unten zum Teil bereitgestellt durch Zurückziehen der Membran [Sig17].	4
2.2	Aufbau einer Flechtmaschine, entnommen aus [Pro]	5
2.3	Veranschaulichung eines einzelnen Picks. Picklänge in rot visualisiert. Entnommen aus [Pro].	5
2.4	Beispiel eines Histogrammausgleichs: links, ursprüngliches Bild; rechts, Ergebnis des Histogrammausgleichs. Darunter sind die zugehörigen Histogramme dargestellt. Entnommen aus [BLF16].	6
2.5	Beispiel für die Anwendung von CLAHE: links, Originalbild mit Nebel; rechts, Bild nach der Anwendung von CLAHE. Entnommen aus [YMA14].	7
2.6	Schematischer Aufbau eines künstlichen neuronalen Netzes, das als einschichtiges Perzeptron-Netz modelliert wird. In Anlehnung an [DN20].	7
2.7	Aufbau eines neuronalen Netzes mit zwei Neuronen für die Eingabe und einem für die Ausgabe	8
2.8	Beispielhafter Verlauf für den mittleren absoluten Fehler (blau) und den mittleren quadratischen Fehler (rot).	10
2.9	Verlauf des Gradientenverfahrens mit einer Lernrate α , einer Verlustfunktion L und den Parametern w. Ursprüngliches Bild aus [YNDT18].	10
2.10	Beispielhafter Verlauf eines Trainings mit Overfitting. Der Trainingsloss sinkt über den gesamten Vorgang, demgegenüber fängt der Validierungsloss ab etwa der 50. Epoche an zu steigen.	12
2.11	Beispiel einer Faltungsoperation mit einen 3×3 Filter. Entnommen und neu beschriftet aus [YNDT18].	14
2.12	Beispiel zur Anwendung von Max Pooling. Ursprüngliches Bild aus [YNDT18].	16
2.13	Aufbau eines Fully Connected Layers.	16
2.14	Aufbau des Inception Layer nach [SLJ ⁺ 15] ohne 1x1 Convolution.	18
2.15	Aufbau des Inception Layer nach [SLJ ⁺ 15] mit 1x1 Convolution.	18
2.16	Beispiel eines Separable Convolution Layer nach [Cho17]	18

2.17 (a) Aufbau des originellen Residual Layers nach [HZRS16a]; (b) Aufbau der Weiterentwicklung eines Residual Layers [HZRS16b].	19
2.18 Aufbau eines Bottleneck Layers nach [HZRS16a].	20
2.19 Zusammenstellung des Inverted Residual Layer [SHZ ⁺ 18].	20
2.20 Rangliste für Modellergebnisse auf ImageNet über die Zeit [ima].	21
2.21 Trainingsfehler (links) und Testfehler (rechts) auf dem CIFAR-10-Datensatz mit "einfachen" Netzen mit 20 und 56 Schichten. Entnommen und Beschriftung übersetzt aus [HZRS16a].	23
2.22 Aufbau eines Residual Layers (links) und dem SE-ResNet Modul (rechts) nach [HSS18].	25
 3.1 Testergebnisse der Anwendung eines Faster R-CNN auf Bilder von Geflechten. Entnommen aus [XPG ⁺ 20].	28
3.2 Aufbau des von Bermudez et al. 2016 entworfenen Systems mit einem dreifachen Beleuchtungssystem. Entnommen und übersetzt aus [BLC ⁺ 17c]. . .	29
3.3 Stent mit einem Rissdefekt: (a) ursprüngliches Bild, (b) Maske der Oberfläche, (c) Maske der Kanten. Entnommen aus [BLC ⁺ 17c].	29
 4.1 Aufbau des Gesamtsystems zur automatisierten Fehlererkennung und -korrektur von Stents basierend auf Kamerabildern	31
4.2 Beispiel für zwei Zeilen von Picks mit gleicher durchschnittlicher Picklänge, fehlerfrei (links) und fehlerhaft (rechts)	32
4.3 Beispiele für die drei größten Anwendungsbereiche von CNNs: Klassifizierung (links), Objekterkennung (Mitte), Segmentierung (rechts). Ursprüngliche Bilder aus [AA18].	33
4.4 Ursprüngliches Bild der Größe 3088×2064 (links) und vorgeschnittene Variante (rechts). Die Länge des sich in der Mitte des in der Bildmitte befindlichen Picks ist rot eingezeichnet.	34
 5.1 Beispiele für ungeeignete Bilder für das neuronale Netz: kein Stent im Bild sichtbar (links), kein Pick in der Mitte des Bildes (Mitte), kein eindeutiger Pick in der Mitte (rot markiert) des Bildes (rechts). Bilder aus [Pro] entnommen.	36
5.2 Weitere Beispiele für aussortierte Bilder: ohne Licht (links), starke Lichtbestrahlung (Mitte), Draht gerissen (rechts). Bilder aus [Pro] entnommen. .	36
5.3 Ursprüngliches Bild	38
5.4 Histogrammausgleich	38
5.5 AHE	38
5.6 CLAHE	38

6.1	Verläufe des Trainingsvorgangs der VGG19-Architektur in linearer (links) und logarithmischer (rechts) Darstellung	42
6.2	Zehn der 64 Feature Maps, welche durch das erste Convolutional Layer einer trainierten VGG19-Architektur entstehen	42
6.3	Sechs der 256 Feature Maps aus dem vierten und letzten Convolutional Layer im dritten Block des VGG19	43
6.4	Vergleich des Fehlers nach Veröffentlichungsjahr der Netzwerkarchitekturen	44
6.5	Vergleich der Rechenzeit pro Bild nach Veröffentlichungsjahr der Netzwerkarchitekturen	44
6.6	Vergleich des Fehlers nach der benötigten Rechenzeit pro Bild	46
6.7	Aufteilung der Abweichungen bei den Ausgaben eines trainierten VGG19 auf den Testdatensatz	46
6.8	Aufteilung der Abweichungen unter den fünf Netzen mit dem besten Fehler	47
6.9	Beispiele für Bilder, welche zu großen Abweichungen bei der Netzausgabe geführt haben. Der Mittelpunkt der Bilder ist rot markiert.	48
6.10	Prozentanteil der Ausgaben mehrerer Netze je nach gesetzter Toleranz. Die drei vorgegebenen Größen $l_1 = 0,23$ Pixel, $l_2 = 0,46$ Pixel und $l_3 = 1,15$ Pixel sind eingezeichnet.	49

