ifes Schriftenreihe

Band 26

Bianca Krol (Hrsg.)

Explainable Artificial Intelligence: Analyse und Visualisierung des Lernprozesses eines Convolutional Neural Network zur Erkennung deutscher Straßenverkehrsschilder

~

Robin Maasjosthusmann, Frank Lehrbass

FOM Hochschule
ifes
Institut für Empirie & Statistik
der FOM Hochschule
für Oekonomie & Management

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

MA Akademie Verlags- und Druck-Gesellschaft mbH
Leimkugelstraße 6, 45141 Essen
info@mav-verlag.de

Robin Maasjosthusmann, Frank Lehrbass

Explainable Artificial Intelligence: Analyse und Visualisierung des Lernprozesses eines Convolutional Neural Network zur Erkennung deutscher Straßenverkehrsschilder

ifes Institut für Empirie & Statistik
der FOM Hochschule für Oekonomie & Management

ifes Schriftenreihe
Band 26, 2021

ISBN (Print) 978-3-89275-427-5
ISBN (eBook) 978-3-89275-428-2
ISSN (Print) 2191-3366
ISSN (eBook) 2569-5355

Inhaltsverzeichnis

Abbildungsverzeichnis V
Tabellenverzeichnis VI
Formelverzeichnis VI
Abkürzungsverzeichnis VII
Symbolverzeichnis VII
1 Einleitung 9
1.1 Einführung in das Thema 9
1.2 Zielsetzung 11
1.3 Aufbau der Studie 12
2 Theoretische Grundlagen 14
2.1 Straßenverkehrsschilder 14
2.1.1 Straßenverkehrsschilder in Deutschland 14
2.1.2 German Traffic Sign Recognition Benchmark 15
2.2 Neural Networks 17
2.2.1 Aufbau 18
2.2.2 Gradient Descent 21
2.3 Convolutional Neural Networks 22
2.3.1 Biologische Grundlagen 23
2.3.2 Architektur 24
2.3.2.1 Local Receptive Fields 24
2.3.2.2 Gemeinsame Gewichte und Bias 25
2.3.2.3 Pooling 26
2.3.2.4 Softmax-Funktion 28
2.3.2.5 Spatial Transformer Networks 29
2.4 Visualisierung der Entscheidung 31
2.4.1 Activation Maximization 32
2.4.2 Saliency Map 33
2.4.3 Class Activation Mapping 34
2.4.3.1 Grad-CAM 34
2.4.3.2 Grad-Cam++ 36
3 Architektur des CNNs 37
3.1 Aufbau 37
3.1.1 Spatial Transformer Networks 37
3.1.2 Features 39
3.1.3 Classifier 41
3.1.4 Parameter 42
3.2 Hyperparameter 42

3.2.1 Aktivierungsfunktionen....43
3.2.2 Batch Size....44
3.2.3 Dropout....44
3.2.4 Epochen....45
3.2.5 Learning Rate....45
3.2.6 Lernalgorithmus....46
3.2.7 Verlustfunktion....48
4 Implementierung....49
4.1 Allgemeines....49
4.2 Aufbereitung des Datensatzes....49
4.3 Zuschneiden des Datensatzes....52
4.4 Bau des CNN....53
4.5 Visualisierungsalgorithmen....55
4.6 Rotation entlang der Achsen....55
4.7 Maskieren von Bildabschnitten....57
4.7.1 Erzeugen des Rasters....58
4.7.2 Auswahl der zu maskierenden Zellen....58
4.8 Erstellen der Stockfotos mit Aufklebern....61
4.8.1 Skalieren der verwendeten Aufkleber....63
4.8.2 Positionieren der Aufkleber....64
5 Detaillierte Auswertung und quantitative Diskussion der Ergebnisse....65
5.1 Allgemeine Performance des CNNs....65
5.2 Activation Maximization....66
5.3 Rotation entlang der Achsen....68
5.4 Manuelle Auswertung der Heatmaps....70
5.5 Einfluss der maskierten Zellen....73
5.6 Effekt realer Aufkleber auf Stockfotos....76
5.7 Reale Beispiele....79
6 Fazit....83
5.8 Ergebnisse der Studie....83
5.9 Evaluierung der Hypothesen und der Forschungsfrage....84
5.10 Konsequenzen der Forschungsergebnisse....85
5.11 Ausblick....85
7 Anhang....88
Literaturverzeichnis....97
Internetquellen....105

Abbildungsverzeichnis

Abb. 1: Anzahl Bilder pro Klasse im GTSRB Trainingsdatensatz 16
Abb. 2: Darstellung eines Perceptrons 19
Abb. 3: Darstellung der Sigmoid-, ReLU- und tanh-Funktion 20
Abb. 4: Allgemeine Darstellung eines Neural Networks mit 3 Layern 21
Abb. 5: Vereinfachter Aufbau visueller Cortex und Convolutional Neural Network 23
Abb. 6: Input Layer als Matrix (links) und Beispiel eines LRFs zwischen Input und 1. Hidden Layer 25
Abb. 7: Beispiel Max Pooling und Average Pooling 27
Abb. 8: Convolutional und Pooling Layer im Zusammenspiel 28
Abb. 9: Spatial Transformer Network 29
Abb. 10: Beispiel einer räumlichen Transformation mit Hilfe des erzeugten Gitters 30
Abb. 11: Gegenüberstellung der Verlustfunktionen anhand von Validation 45
Abb. 12: Übersicht über alle Klassen im Datensatz 51
Abb. 13: Beispiele für die Rotation entlang der Achsen 56
Abb. 14: Beispiel Verlauf der Test Accuracy beim Maskieren 57
Abb. 15: Darstellung aller zur Maskierung verwendeten Versionen eines Bildes 59
Abb. 16: Beispiele der Ergebnisse des Maskierens 61
Abb. 17: Darstellung der verwendeten Aufkleber 62
Abb. 18: Beispiele für die verschiedenen Schildergrößen mit Aufklebern 63
Abb. 19: Activation Maximization Resultate 68
Abb. 20: Verlauf der Test Accuracy bei Rotation entlang der x- bzw. z-Achse 69
Abb. 21: Gegenüberstellung Heatmaps originaler und zugeschnittener Datensatz 71
Abb. 22: Anzahl generierter Heatmaps der Visualisierungsalgorithmen 73
Abb. 23: Accuracy-Verlauf bei steigender Anzahl der maskierten Zellen 74
Abb. 24: Accuracy auf den Stockfotos mit unterschiedlicher Schildergröße 77
Abb. 25: Darstellung der Schilderklasse 00033 mit dem skalierten Fortuna-Aufkleber 78
Abb. 26: Gegenüberstellung der originalen und korrigierten Straßenschilder 79
Abb. 27: Visualisierung Feature Maps nach dem ersten Convolutional Layer 89
Abb. 28: Activation Maximization Resultate 93

Tabellenverzeichnis

Tab. 1: Übersicht der höchsten erreichten Accuracy auf den GTSRB Datensatz 17
Tab. 2: Architektur Localisation Network der STNs 38
Tab. 3: Architektur des CNNs ohne Details der STNs 41
Tab. 4: Aufklebermaße für die verschiedenen Schildergrößen 62
Tab. 5: Durchschnittliche Wahrscheinlichkeiten Klassifizierung der Stockfotos 79
Tab. 6: Übersicht Validation Loss der drei getesteten Verlustfunktionen .. 94
Tab. 7: Übersicht Validation Accuracy der drei getesteten Verlustfunktionen 95
Tab. 8: Übersicht der verwendeten Python Bibliotheken 96

Formelverzeichnis

Formel 1: Anzahl der Spalten in der Ausgabe 25
Formel 2: Berechnung des neuen Gewichts in Gradient Descent 46
Formel 3: Anwendung von Momentum in Gradient Descent 47

Abkürzungsverzeichnis

AM	Activation Maximization
CAM	Class Activation Mapping
CNN	Convolutional Neural Network
XAI	Explainable Artificial Intelligence
GD	Gradient Descent
GTSRB	German Traffic Sign Recognition Benchmark
KI	Künstliche Intelligenz
LRF	Local Receptive Field
MASTIF	Mapping and Assessing the State of Traffic InFrastructure
NN	Neural Network
RGB	Rot, Grün, Blau
SGD	Stochastic Gradient Descent
SM	Saliency Map
StVO	Straßenverkehrsordnung
STN	Spatial Transformer Network
NAG	Nesterov Accelerated Gradient Descent
SVM	Support Vector Machine
CSV	Comma Separated Values
VzKat	Katalog der Verkehrszeichen
Kfz	Kraftfahrzeuge
RFID	Radio-Frequency Identification

Symbolverzeichnis

α	Alphawert für Grad Class Activation Mapping (auch für Mapping++)
Δ	Differenz bzw. Veränderung einer Menge
δ	Kronecker-Delta
η	Learning Rate
θ	Variabler Parameter in Formel
λ	Regulierungsterm in Activation Maximation
μ	Momentum in der Anpassung der Gewichte
Φ	Parameter des generischen Kernels eines Spatial Transformer Network
σ	Aktivierungsfunktion
ω	Multiplikator im Leaky ReLU Algorithmus

∂	Partielle Ableitung
Tθ	Punkte des Gitters G
∇	Nabla-Operator für Vektor Rechnung
a	Aktiverung bzw. Ausgabesignal eines Neuron
b	Bias innerhalb eines Neural Network
C	Verlust bzw. Kosten eines Neuronal Network
e	Eulersche Zahl
G	Gitter im Spatial Transformer Network
I	Eindimensionaler Vektor eines Bildes
i	Anzahl Spalten in Ergebnismatrix
kD	Dimensionen des Kernels
L	Layer innerhalb eines Neural Network
LCAM	Ergebnismatrix Class Activation Mapping
LGrad	Ergebnismatrix Grad-Class Activation Mapping
mean	Durchschnittswert einer Menge an Objekten, z. B. Pixel
N	Anzahl von Elementen, z. B. Layer in einem Neural Network
p	Padding
s	Stride
std	Standardabweichung innerhalb einer Menge an Objekten, z. B. Pixel
T	Transponieren des Vektors
w	Gewicht innerhalb eines Neural Network
y	Wahre Klasse eines Beispiels
yˆ	Vom Convolutional Neural Network bestimmte Klasse eines Beispieles
z	Gewichtete Summe eines Neuron

1 Einleitung

Autonomes Fahren ist ohne den Einsatz von KI undenkbar. Da aber jedes Fahrzeug eine potentielle Gefahrenquelle darstellt, ist es besonders wichtig, die dabei eingesetzte KI verstehen zu können. Eine dabei bedeutsame Herausforderung ist zu begreifen, worauf die Klassifizierung deutscher Straßenverkehrsschilder mit Hilfe von KI basiert. Die nachfolgende Zielsetzung konkretisiert diese. Ein Ausblick auf den Gang der Studie folgt.

1.1 Einführung in das Thema

Durch die Verfügbarkeit autonom fahrender Fahrzeuge wird ein revolutionärer Transformationsprozess im Transportsektor erwartet.[1] Dadurch, dass sich diese Transformation sowohl auf den kommerziellen (Waren- oder Personentransport) als auch auf den privaten Sektor (private Kraftfahrzeuge (Kfz)) bezieht, wird sich ein Großteil der Teilnehmer am Straßenverkehr in absehbarer Zeit selbst in autonomen Fahrzeugen oder aber parallel zu solchen fortbewegen. Das automatisierte Fahren von Fahrzeugen wird in fünf Stufen unterteilt. Erst Stufe 5 umfasst dabei das vollständig selbstständige Fahren der Fahrzeuge und kann daher als autonomes Fahren angesehen werden. Die vorherigen Level bedienen die volle Bandbreite von funktional partikulär unterstützenden Systemen (z. B. Einparkhilfen) bis zur autonomen Lenkung bei Anwesenheit eines verantwortlichen Fahrers.[2] Bereits Mitte 2019 waren alleine in den USA mehr als 1400 autonome Fahrzeuge von über 80 Unternehmen am öffentlichen Straßenverkehr beteiligt.[3] Aktuelle Schätzungen gehen davon aus, dass bis 2045 bis zu 50 % der Neuwagenzulassungen und 40 % der mit Fahrzeugen zurückgelegten Kilometer auf autonome Fahrzeuge zurückgehen.[4]

Damit autonome Fahrzeuge sicher am Straßenverkehr teilnehmen können, müssen sie in der Lage sein, ihre Umgebung wahrzunehmen und notwendige Informationen zu extrahieren. Dies kann unter anderem mittels Kameras geschehen, welche die Umgebung durch Fotos oder Videos aufzeichnen. Die erzeugten optischen Datensätze stellen den Anfang der Verarbeitungskette auf der Softwareebene dar. Das Erkennen und Klassifizieren von Straßenverkehrsschildern (auch

[1] Vgl. McKinsey and Company, Revolution, 2021, o. S.
[2] Vgl. On-Road Automated Driving (ORAD) committee, Automatisierungsstufen, 2018, S. 19.
[3] Vgl. US Department of Transportation, Fahrzeuganzahl, 2019, o. S.
[4] Litman, T., Entwicklungsvorhersage, 2020, S. 27.

Straßenschilder oder Verkehrsschilder genannt) ist dabei von hoher Relevanz. Ein aktueller Ansatz für die Klassifizierung von Bildern ist die Verwendung von Deep Learning, insbesondere durch Neural Networks (NNs) beziehungsweise die Unterkategorie Convolutional Neural Networks (CNNs). Als Deep Learning werden Modelle bezeichnet, die aus vielen Schichten bestehen und in der Lage sind, komplizierte Strukturen zu lernen. Dieses Lernen wird durch das Anwenden des Backpropagationalgorithmus erreicht, welcher basierend auf Trainingsdaten eine Parameteroptimierung herbeiführt, wobei es eine Vielzahl möglicher Optimierungskriterien gibt.[5]

Ein großer Nachteil von Deep Learning und somit CNNs ist, dass die Entscheidungsfindung durch die hohe Anzahl von Parametern innerhalb der Modelle für Menschen nicht mehr nachvollziehbar ist. Bereits das häufig als Referenz verwendete VGG-19 Modell aus dem Jahr 2015 hat ca. 144 Millionen Parameter, welche auf Grundlage von 1,3 Millionen Bildern im Trainingsdatensatz angepasst wurden.[6] Der Mensch ist in der Lage, den Aufbau des Modelles und die zum Trainieren verwendeten Algorithmen zu verstehen, allerdings ist für ihn nicht mehr nachvollziehbar, wofür die Werte der einzelnen Parameter beziehungsweise der Zusammenschluss mehrerer Parameter stehen. Dies kann dazu führen, dass auch Modelle, die – basierend auf den ausgewählten Erfolgsmetriken – gute Ergebnisse liefern, sich an falschen Attributen orientieren. In einem Beispiel zur Unterscheidung von Huskys und Wölfen von Ribeiro et al. konnte eine hohe Accuracy auf dem Testdatensatz erreicht werden. Accuracy wird definiert als das Verhältnis der Zahl aller korrekt klassifizierten Elemente zur Anzahl aller Elemente. Allerdings wurden die Bilder im Datensatz durch die Autoren so ausgewählt, dass bei den Wölfen immer Schnee im Hintergrund zu sehen war und bei den Huskys nie. Anschließend konnten die Autoren nachweisen, dass das CNN die Tiere basierend auf dem Vorhandensein von Schnee und nicht anhand der anatomischen Merkmale der Tiere selbst klassifiziert hat.[7] In anderen Fällen führten bereits geringe Veränderungen der Bilder zu fehlerhaften Klassifizierungen.[8]

Dieses fehlende Verständnis für die Entscheidungsfindung eines Modells und die darauf basierende Unsicherheit des Gelernten bringt mehrere Nachteile mit sich. So fehlt den Nutzern vor allem bei sicherheitsrelevanten Aufgaben – wie dem

5 LeCun, Y., Bengio, Y., Hinton, G., Deep Learning, 2015, S. 1.
6 Vgl. Simonyan, K., Zisserman, A., VGG-19, 2015, S. 3-5.
7 Vgl. Ribeiro, M. T., Singh, S., Guestrin, C., Wölfe vs. Huskeys, 2016, S. 1142 f.
8 Vgl. Szegedy, C. et al., Perturbation, 2014, S. 5.

autonomen Fahren – das Vertrauen in solche Modelle.[9] Des Weiteren haben Fachexperten keine Möglichkeit, die Entscheidung des Modells nachzuvollziehen und so gegebenenfalls neue Erkenntnisse zu erlangen oder sogar auf mögliche Fehler des Modells aufmerksam zu werden.[10] Dieser Faktor verstärkt das Misstrauen der Nutzer weiter, da auch Fachexperten das Vertrauen nicht stärken können.

1.2 Zielsetzung

Um dieser Problematik entgegentreten zu können, wurden in der Vergangenheit eine Vielzahl Techniken entwickelt, welche Deep Learning Modelle transparenter und nachvollziehbarer machen sollen. Dieser Forschungsbereich wird als Explainable Artificial Intelligence (XAI) bezeichnet.[11] Für CNNs wurde eine Vielzahl von Visualisierungsalgorithmen entwickelt, welche relevante Bereiche innerhalb des Bildes markieren und somit Einblick in die Entscheidungsgrundlage des CNNs geben sollen.[12] Im Rahmen dieser Studie wird eine Auswahl dieser Algorithmen verwendet, um zu analysieren, worauf die Klassifizierung deutscher Straßenverkehrsschilder eines CNNs basiert. Aufbauend darauf wird im Anschluss ermittelt, ob die Klassifizierung des CNNs anfällig für Veränderungen der Schilderklassen ist. Dieser Aspekt ist für den Einsatz in autonomen Fahrzeugen von höchster Relevanz, da Straßenschilder über die Zeit verschiedenen äußeren Einflüssen ausgesetzt sind. So können unter anderem Schmutz, Aufkleber oder Graffiti Bereiche der Schilder verdecken. Anders formuliert lautet die Forschungsfrage: "Welche Merkmale verwendet ein CNN zur Klassifizierung von deutschen Straßenverkehrsschildern und wie anfällig für äußere Einflüsse ist dies?". Zur weitergehenden Konkretisierung des Forschungsgegenstands leiten wir zwei Hypothesen aus der so formulierten Forschungsfrage ab:

1. Es ist nicht möglich, zu ermitteln, auf welche Merkmale ein CNN zugreift, um ein Straßenverkehrsschild zu kategorisieren.
2. Es ist nicht möglich, anhand der bekannten Eigenschaften zur Kategorisierung reale Beispiele zu finden, in denen das CNN Straßenschilder fehlerhaft kategorisiert.

[9] Vgl. Holzinger, A., Plass, M. et al., Vertrauen, 2017, S. 2.
[10] Vgl. Holzinger, A., Biemann, C. et al., Fachexperten, 2017, S. 3.
[11] Adadi, A., Berrada, M., XAI, 2018, S. 52138.
[12] Yu, R., Shi, L., Visualisierungsalgorithmen, 2018, S. 147-154.

Zur Beantwortung der Forschungsfrage und der Evaluierung der Hypothesen wird zunächst ein CNN erzeugt, welches in der Lage ist, deutsche Straßenverkehrsschilder auf einem ähnlich guten Niveau wie Referenzmodelle zu klassifizieren. Das trainierte CNN wird anschließend auf eine Reihe von Testdatensätzen angewandt, die den Hypothesen entsprechend hinsichtlich ihrer Merkmalsausprägungen angepasst wurden. Es werden zwei Datensätze erzeugt, die die Anfälligkeit des CNNs auf Rotation entlang der x- und z-Achse testen. Diese können vorkommen, wenn die Halterung eines Schildes nicht mehr korrekt funktioniert oder die Schilder durch äußere Gewalt verschoben werden. Außerdem soll der Einfluss von Störfaktoren wie Aufklebern auf den Schildern getestet werden. Hierzu wird eine Auswahl an Visualisierungsalgorithmen angewendet und deren Resultate sowohl visuell als auch numerisch auf relevante Merkmale der einzelnen Schilderklassen analysiert. Anhand der ermittelten Merkmale werden die jeweiligen Testdatensätze und Stockfotos so bearbeitet, dass eine Verschlechterung der Performance des CNNs zu erwarten ist. Ist diese vorhanden, kann auf Grundlage der Größe des Verlustes bestimmt werden, inwieweit die gewählten Visualisierungsalgorithmen die relevanten Bereiche ausmachen konnten. Abschließend werden – für das CNN unbekannte – Beispiele von Straßenschildern mit Abweichungen von ihrem Sollzustand verwendet. Die Beispiele wurden hierbei basierend auf den zuvor erfolgten Analysen ausgewählt, um Hypothese 2 zu überprüfen.

1.3 Aufbau der Studie

Diese Studie gliedert sich im Folgenden in fünf Kapitel. In *Kapitel 2* werden zunächst die theoretischen Grundlagen der Thematik erläutert. Zu Beginn wird genauer auf Straßenverkehrsschilder und den verwendeten Datensatz eingegangen. Danach werden der Aufbau und die Funktionsweise von NNs und besonders CNNs beleuchtet. Abschließend werden die verwendeten Visualisierungsalgorithmen erläutert. *Kapitel 3* geht tiefer auf den Aufbau des verwendeten CNNs ein. Hierbei wird zunächst der final verwendete Aufbau beschrieben. Anschließend werden die einzelnen Designentscheidungen diskutiert und mögliche Alternativen aufgezeigt. *Kapitel 4* beschreibt die Implementierung. Grundlegend wird hierbei auf die verwendete Software sowie Bibliotheken eingegangen. Dabei wird erläutert, wie genau der Datensatz aufbereitet wurde, wie das CNN implementiert wurde und wie die Tests beziehungsweise Auswertungen umgesetzt wurden. Die Resultate der Tests werden in *Kapitel 5* dargestellt. Es wird auf jeden Test spezifisch eingegangen und die Ergebnisse diskutiert. Zum Abschluss findet

in *Kapitel 6* eine Diskussion der Studie statt. Hier wird reflektiert, ob die Forschungsfrage beantwortet werden konnte, die Hypothesen verworfen werden können oder beibehalten werden müssen. Die Studie schließt mit einem Ausblick auf weitere Forschungsbereiche im Rahmen dieser Thematik ab. Die verwendeten Codings liegen auf https://github.com/RobinMaas95/GTSRB_Visualization.

2 Theoretische Grundlagen

In diesem Kapitel soll eine kurze Übersicht über die theoretischen Grundlagen gegeben werden. Die relevanten Fachbegriffe werden definiert und eine Skizzierung der angewendeten Methoden vorgenommen. Im folgenden Abschnitt wird auf Straßenverkehrsschilder und den verwendeten Datensatz eingegangen. Anschließend werden wichtige Aspekte von NNs und CNNs erläutert und hergeleitet (Abschnitte 2.2 bzw. 2.3). Abschließend werden die verwendeten Algorithmen zur Visualisierung des Gelernten des CNNs (Abschnitt 2.4) vorgestellt.

2.1 Straßenverkehrsschilder

In diesem Abschnitt soll eine kurze Definition von Straßenverkehrsschildern sowie eine Vorstellung der in Deutschland geltenden Rechtslage hinsichtlich dieser gegeben werden. Außerdem wird ein Überblick über verfügbare Datensätze mit Schwerpunkt auf den German Traffic Sign Recognition Benchmark (GTSRB) Datensatz gegeben.

2.1.1 Straßenverkehrsschilder in Deutschland

Straßenverkehrsschilder dienen dazu, den Straßenverkehr zu regeln. Hierbei stellen sie den Verkehrsteilnehmern Warnungen, Informationen und Einzelheiten über Einschränkungen zur Verfügung. Sie gelten nach §43 der Straßenverkehrsordnung (StVO) als dringliche Verwaltungsakte oder als Rechtsverordnung. §41 Absatz 1 der StVO gibt zudem vor, dass Teilnehmer am Verkehr den Vorgaben der Straßenverkehrsschilder zu folgen haben.[13]

Alle gültigen Straßenverkehrsschilder in Deutschland werden in der StVO bzw. dem Katalog der Verkehrszeichen (VzKat) gelistet.[14] Diese werden vom Gesetzgeber fortlaufend den aktuellen Gegebenheiten angepasst und durch Novellen aktualisiert. Grundsätzlich basieren die Straßenverkehrsschilder dabei auf dem am 8. November 1968 in Wien beschlossenen *Übereinkommen über den Straßenverkehr* (engl. Originaltitel "Convention on Road Traffic").[15] Dieses wurde bis heute von 36 Nationen unterzeichnet und von 84 Nationen ratifiziert.[16]

[13] Vgl. Bundesrepublik Deutschland, StVO, 2013, o. S.
[14] Bundesrepublik Deutschland, VzKat, 2017, o. S.
[15] United Nations, Konvention Straßenverkehr, 1977, o. S. [16]
[16] United Nations, Teilnehmer Konvention, 2020, 1 f.

In der StVO sind drei Gruppen von Verkehrszeichen definiert:

- Gefahrenzeichen (§ 40 StVO)
- Vorschriftzeichen (§ 41 StVO)
- Richtzeichen (§ 42 StVO)

Neben diesen Verkehrszeichen gibt es noch Zusatzzeichen, welche zusammen mit Zeichen aus den genannten Gruppen verwendet werden. Diese sind in §39 Abs. 7 StVO aufgelistet.

2.1.2 German Traffic Sign Recognition Benchmark

Für den praktischen Teil dieser Studie wird ein Datensatz von Straßenverkehrsschildern benötigt, mit denen das CNN initial trainiert und bewertet werden kann. Es gibt eine Vielzahl von Datensätzen, die hierfür potenziell geeignet wären. Das *Mapping and Assessing the State of Traffic InFrastructure (MASTIF)* Projekt stellt drei verschiedene Datensätze aus den Jahren 2009, 2010 und 2011 zur Verfügung. Der umfangreichste ist hierbei der Datensatz aus 2009 mit 6000 Bildern.[17] Shakhuro und Konushin stellen einen Datensatz mit 104 358 verschiedenen russischen Schildern aus dem Jahre 2016 zur Verfügung.[18]

Der bisher umfangreichste Datensatz ist das *Mapillary Traffic Sign Dataset* aus dem Jahre 2020. Es enthält 100 000 Bilder mit insgesamt 325 172 erkannten Schildern. Von diesen Schildern fallen 82 724 Schilder in die im Datensatz gekennzeichneten Schilderklassen.

Die Bilder stammen dabei aus der ganzen Welt, eine grobe Verteilung liegt bei 20 % aus Nordamerika, 20 % aus Europa, 20 % aus Asien, 15 % aus Südamerika, 15 % aus Ozeanien und 10 % aus Afrika.[19]

[17] Šegvic´, S. et al., MASTIF, 2010, S. 66-73.
[18] Shakhuro, V., Konushin, A., RTSD, 2016, S. 294-300.
[19] Ertler, C. et al., Mapillary, 2020, S. 1-17.

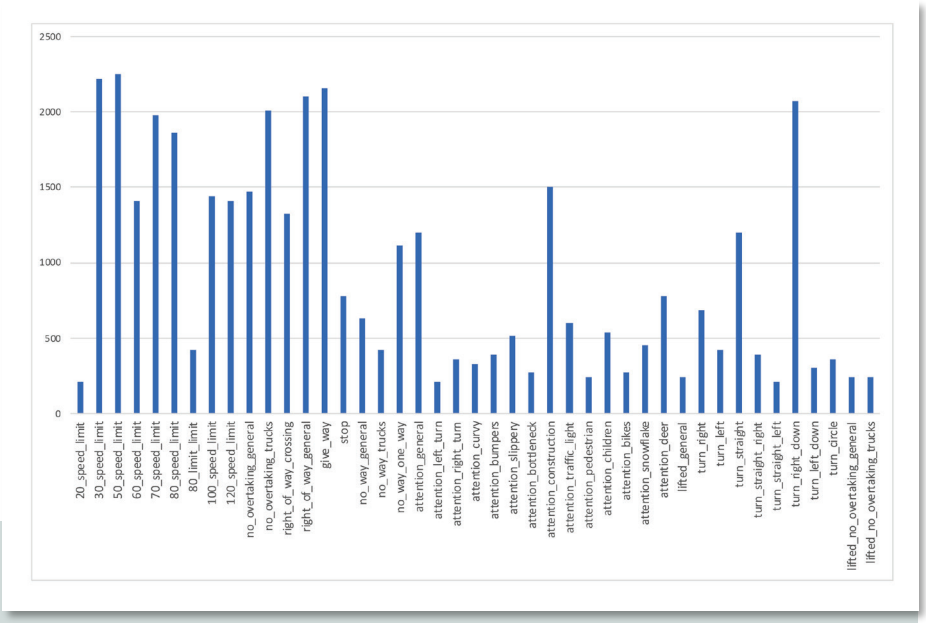


Abb. 1: Anzahl Bilder pro Klasse im GTSRB Trainingsdatensatz

Letztendlich wurde sich für den Datensatz GTSRB entschieden. Dieser umfasst 51 840 Bilder von 1700 verschiedenen deutschen Straßenverkehrsschildern, welche in 43 Klassen unterteilt wurden. Grund für die Auswahl war primär, dass dieser Datensatz die größte Anzahl deutscher Straßenverkehrsschilder bereitstellt, auf denen der Schwerpunkt dieser Studie liegt. Der Datensatz wurde im Jahre 2010 erstellt und die Maße der Bilder variieren zwischen 15x15 und 222x193 Pixel. In Abbildung 1 wird die Anzahl der Bilder pro Klasse im Trainingsdatensatz des GTSRB dargestellt. Die 43 Klassen des Datensatzes fallen alle unter die drei Gruppen der Verkehrszeichen nach § 39 Abs. 2 Satz 2 StVO. Dabei fallen 27 Klassen unter Vorschriftzeichen, 14 Klassen unter Gefahrenzeichen und 2 Klassen unter

Richtzeichen. Für den GTSRB liegt eine Vielzahl von Forschungsarbeiten vor.[20,21,22,23,24] Die fünf Einreichungen mit der höchsten Accuracy auf dem Testdatensatz sind in Tabelle 1 aufgelistet. Es ist zu erkennen, dass eine sehr hohe Accuracy erreicht wird (99,71 %).

Team	Methode	Prozentzahl (Genauigkeit)
Arcos-García, Álvarez-García, Soria-Morillo	CNN mit 3 Spatial Transformers	99,71
Ciresan, Meier, Masci, Schmidhuber	Zusammenschluss mehrerer CNNs	99,46
Gecer, Azzopardi, Petkov	Farbenbasierter COSFIRE-Filterzur Objekterkennung	98,97
Stallkamp, Schlipsing, Salmen, Igel	Durchschnittliches menschliches Abschneiden	98,84
Sermanet, LeCun	Mehrstufige CNNs	98,31

Tab. 1: Übersicht der höchsten erreichten Accuracy auf den GTSRB Datensatz, Quelle: In Anlehnung an Institut für Neuroinformatik Ruhr-Universität Bochum, Resultate GTSRB, 2019, o. S.

2.2 Neural Networks

In den letzten Jahren gab es eine Vielzahl von Fortschritten bei Computersystemen, die häufig unter dem Begriff Künstliche Intelligenz (KI) zusammengefasst werden. So konnte zum Beispiel *AlphaGo* einen der weltweit besten Spieler in dem Brettspiel Go schlagen und das System *DeepFace* ist in der Lage, menschliche Gesichter nahezu auf menschlichem Niveau erkennen zu können.[25, 26]

Systeme, die als KI bezeichnet werden, sind häufig in der Lage, Aufgaben zu bewältigen, die bisher ausschließlich durch das menschliche Gehirn bearbeitet werden konnten. Neben den oben genannten Beispielen zählt hier auch das Erkennen von Bildern oder Sprache dazu. Damit Computer diese Aufgaben erledigen können, haben Wissenschaftler sich vom menschlichen bzw. biologischen Gehirn

[20] Vgl. Arcos-García, Á., Álvarez-García, J. A., Soria-Morillo, L. M., Mehrere STN, 2018, S. 1-15.
[21] Vgl. Gecer, B., Azzopardi, G., Petkov, N., COSFIRE-Filter, 2017, S. 165-174.
[22] Ciresan, D. et al., Zusammenschluss Mehrerer CNNs, 2012, S. 333-338.
[23] Stallkamp, J. et al., GTSRB, 2012, S. 323-332.
[24] Sermanet, P., LeCun, Y., Mehrstufige CNNs, 2011, S. 2809-2813.
[25] Vgl. Spiegel, Alpha Go, 2016, o. S.
[26] Vgl. Taigman, Y. et al., Deepface, 2014, 6 ff.

inspirieren lassen. Systeme, die auf dieser Grundlage basieren und einige der Vorgehensweisen des Gehirns imitieren, werden als NNs bezeichnet.

2.2.1 Aufbau

Die grundlegenden Bausteine von NNs sind Perceptrons und wurden 1958 von Rosenblatt beschrieben.[27] Als Grundlage verwendete er hierbei unter anderem die Arbeit von McCulloch und Pitts.[28]

Ein Perceptron kann zwischen 1 und n Eingangssignalen sowie einen negativen Bias Term[29] b erhalten und hieraus ein Ausgangssignal – auch Aktivierung genannt – erzeugen. Für das Ausgangssignal werden alle Eingangssignale jeweils mit einem eigenen – als Gewicht bezeichneten – Faktor multipliziert. Die Ergebnisse werden anschließend zusammen mit dem Bias summiert. Ist die Summe kleiner oder gleich 0, ist der Ausgangswert ebenfalls 0, ansonsten beträgt der Ausgangswert 1. Der Bias stellt also den Schwellenwert dar, der von der gewichteten Summe überschritten werden muss, damit das Ausgangssignal 1 lautet.

Es ist möglich, ein Perceptron um eine Aktivierungsfunktion zu ergänzen, sodass das Ausgabesignal reelle Werte annehmen kann. Diese Erweiterung des Perceptrons wird als Neuron bezeichnet. Weit verbreitet ist die Verwendung der Sigmoid-Funktion. Bei der Anwendung einer Aktivierungsfunktion, häufig mit σ dargestellt, wird die gewichtete Summe z in die gewählte Funktion gegeben und das Ergebnis stellt das Ausgangssignal des Neurons dar.

[27] Vgl. Rosenblatt, F., Perceptron, 1958, S. 394-402.
[28] Vgl. McCulloch, W. S., Pitts, W., Nervenaktivitäten, 1943, S. 115-133.
[29] Einige Definitionen schreiben nicht vor, dass der Bias negativ sein muss. Stattdessen wird dieser im weiteren Vorgehen subtrahiert statt summiert zu werden

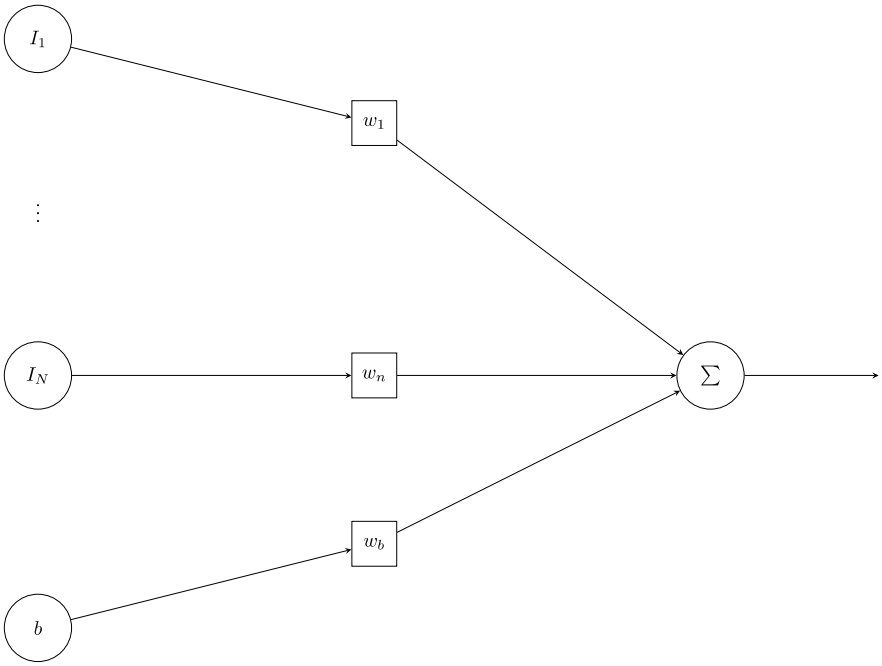


Abb. 2: Darstellung eines Perceptrons, Quelle: In Anlehnung an Nielsen, M., Deep Learning, 2015, o. S.

Neben der Sigmoid-Funktion werden in modernen NNs häufig die ReLu-Funktion oder die tanh-Funktion verwendet. Alle drei werden in Abbildung 3 dargestellt. Es ist zu erkennen, dass sowohl bei der Sigmoid-Funktion als auch bei der ReLUFunktion die untere Grenze des Wertebereichs bei 0 liegt. Bei der tanh-Funktion liegt diese bei –1. Außerdem haben nur die Sigmoid- und tanh-Funktion eine obere Grenze – beide 1 – definiert. Die ReLU-Funktion kann jeden Wert größer gleich 0 annehmen.

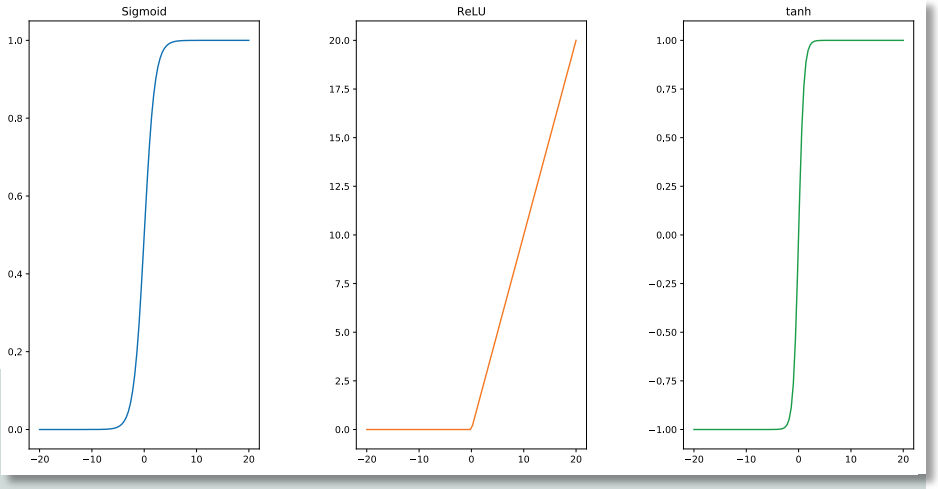


Abb. 3: Darstellung der Sigmoid-, ReLU- und tanh-Funktion

Verwendet man eine Aktivierungsfunktion, stellt der Bias keinen Schwellwert mehr dar. Er ist als einziger Term in der Berechnung des Wertes, der in die Aktivierungsfunktion gegeben wird, von den Eingangswerten unabhängig und sorgt somit für eine konstante Verschiebung. Einen tieferen Einblick in das Thema Aktivierungsfunktionen gibt Szandała in *Benchmarking Comparison of Swish vs. Other Activation Functions on CIFAR-10 Imageset*.[30]

Ein NN besteht aus mehreren verschalteten Neuronen. Diese werden in Layern angeordnet, welche typischerweise mit 1 bis *N* durchnummeriert werden. Layer 1 wird außerdem als Input Layer bezeichnet, da dieser die einzelnen Eingangswerte in das NN darstellt. Layer N wiederum wird als Output Layer bezeichnet, da er das Ergebnis des NNs ausgibt. Alle Layer dazwischen ($1 < l < N$) werden als Hidden Layer bezeichnet, da von außen weder direkt auf die Eingangssignale der Neuronen in diesen Layern Einfluss genommen werden kann, noch die Ausgabesignale dieser direkt ersichtlich sind. Jeder Layer kann dabei eine unterschiedliche Anzahl von Neuronen enthalten. Typischerweise sind die Neuronen in einem Layer mit allen Neuronen des vorherigen Layers verbunden. Ist dies der Fall, wird der Layer auch als Dense Layer bezeichnet. Eine allgemeine Darstellung eines NNs mit drei Layern ist in Abbildung 4 dargestellt.

[30] Vgl. Szandała, T., Aktivierungsfunktionen, 2021, S. 203-224.

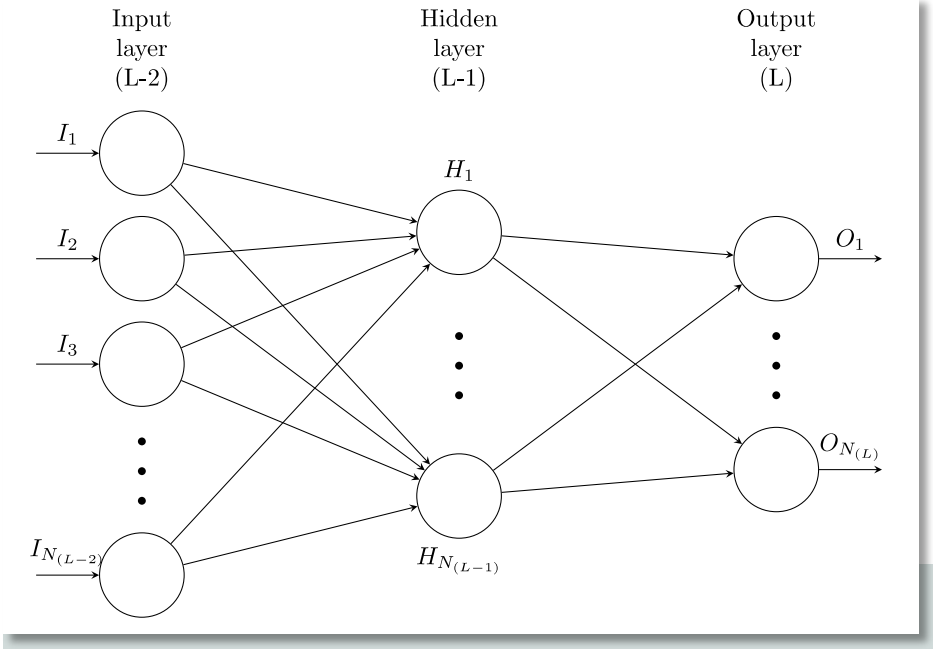


Abb. 4: Allgemeine Darstellung eines Neural Networks mit 3 Layern, Quelle: In Anlehnung an Nielsen, M., Deep Learning, 2015, o. S.

2.2.2 Gradient Descent

Im Rahmen des Lernprozesses eines NNs werden die Gewichte und Bias der einzelnen Perceptrons regelmäßig aktualisiert. Dabei liegt das Ziel darin, durch die Optimierung dieser Modellparameter den Fehler des NNs bei der Vorhersage zu minimieren.

Um diesen Fehler bestimmen zu können, wird eine Verlustfunktion definiert. Ein einfaches Beispiel einer Verlustfunktion basiert auf der mittleren quadratischen Abweichung.

Neben dieser Verlustfunktion gibt es eine Vielzahl weiterer Verlustfunktionen. Ein Vergleich einiger dieser ist in dem Paper *How much progress have we made in*

neural network training? A New Evaluation Protocol for Benchmarking Optimizers von Xiong et al. zu finden.[31]

Um den Fehler des NNs zu minimieren, muss der globale Tiefpunkt der Verlustfunktion bestimmt werden. Bei Funktionen mit einer Vielzahl von Parametern ist es zu komplex, diese Aufgabe analytisch zu lösen. Gradient Descent (GD) stellt einen iterativen Prozess dar, sich dem nächsten Minimum zu nähern. Bildlich kann dieser Vorgang mit einem Wanderer, der vom Gipfel eines Berges absteigen möchte, verglichen werden. Geht dieser von seinem aktuellen Standpunkt ein paar Schritte in die entgegengesetzte Richtung der steilsten Stelle am Berg und wiederholt diesen Vorgang immer wieder, wandert er immer weiter ins Tal.

2.3 Convolutional Neural Networks

In einem NN sind ein Großteil der Neuronen zwischen den Layern miteinander verbunden. Dies führt dazu, dass die Zahl der Parameter innerhalb des NNs schnell steigen kann. Dies ist zum Beispiel bei Bildern, bei denen jedes einzelne Pixel einen eigenen Eingabewert in das Netzwerk darstellt, der Fall. Da für jeden Parameter Berechnungen durchgeführt werden müssen, erfordert dies einen sehr hohen Trainingsaufwand.

Zusätzlich entstehen Formen und Objekte in Bildern erst durch den räumlichen Zusammenhang einiger Pixel. Daher erscheint das Vorgehen, alle Neuronen im Input Layer (Pixelwerte) mit allen Neuronen im ersten Hidden Layer zu verbinden, als wenig sinnvoll.

Aus diesen und weiteren Gründen werden heutzutage CNNs für die Klassifizierung von Bildern verwendet. Durch besondere Bausteine sorgen diese dafür, dass sowohl der räumliche Zusammenhang der Pixel betrachtet wird, als auch die Anzahl von Parametern handhabbarer ist. Basierend auf den biologischen Vorbildern – siehe nächster Abschnitt – schlugen Fukushima und Miyake im Jahre 1972 das Neocognitron vor, welches als ein Vorgänger von CNNs gesehen werden kann.[32] LeCun et al. verwendeten in ihrem Artikel *Handwritten digit recognition with a back-propagation network* von 1990 einen dem Convolutional Layer (siehe Abschnitt 2.3.2.1) ähnlichen Aufbau.[33] Der Artikel *Gradient-based learning applied*

[31] Vgl. Xiong, Y. et al., Verlustfunktionen, 2020, S. 1-16.
[32] Vgl. Fukushima, K., Miyake, S., Neocognitron, 1982, S. 267-285.
[33] Vgl. LeCun, Y., Boser, B. et al., Handschriftserkennung, 1990, S. 396-404.

to document recognition von LeCun et al. aus dem Jahre 1998 beschreibt erstmals den Aufbau eines heute üblichen CNN.[34]

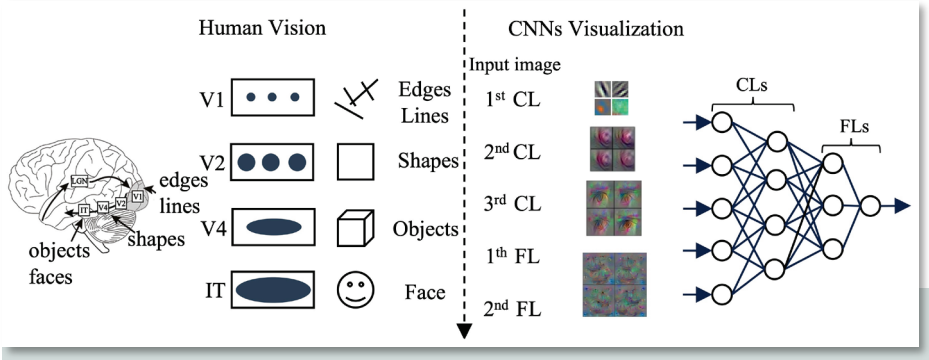


Abb. 5: Vereinfachter Aufbau visueller Cortex und Convolutional Neural Network, Quelle: Qin et al., 2018, S. 154

2.3.1 Biologische Grundlagen

CNNs basieren auf Beobachtungen der Funktionsweise des visuellen Cortex bei Menschen und Tieren. So haben Hubel und Wiesel 1959 bzw. 1962 gezeigt, dass einzelne Sehzellen nur auf Veränderungen in einzelnen Bereichen der Netzhaut reagieren. Diese Bereiche wurden von Hubel und Wiesel als Receptive Field bezeichnet. [35] [36]

In Abbildung 5 wird der vereinfachte visuelle Cortex eines Menschen dem Aufbau eines CNNs gegenübergestellt. Auf der linken Seite ist zu erkennen, dass beim visuellen Cortex verschiedene Schichten (V1, V2, V4 und IT) vorliegen, welche nacheinander geschaltet sind. Jede dieser Schichten enthält Receptive Fields, wobei diese mit jeder Schicht größer werden. Die früheren Schichten reagieren hierbei auf einfache Formen wie Linien oder Kanten und die späteren auf komplexere Formen wie ganze Objekte. Bei allen Schichten können die gesamten Elemente an jeder Stelle des Receptive Field auftreten und trotzdem für eine Aktivierung

[34] LeCun, Y., Bottou, L. et al., Gradientenbasiertes Lernen, 1998, S. 2278-2324.
[35] Vgl. Hubel, D. H., Wiesel, T. N., Receptive Field, 1959, S. 574-591.
[36] Vgl. Hubel, D. H., Wiesel, T. N., Receptive Field, 1962, S. 106-154.

sorgen. Die rechte Seite zeigt den typischen Aufbau eines CNNs, bei dem die früheren Schichten (CL) einfache und spätere Schichten (FL) komplexe Formen erkennen. [37]

2.3.2 Architektur

CNNs haben drei grundlegende Bausteine, die sie von anderen NNs unterscheiden. Diese sind Local Receptive Fields (LRFs), gemeinsame Gewichte und Bias sowie Pooling und werden nachfolgend beleuchtet.

2.3.2.1 Local Receptive Fields

Die Grundidee hinter LRFs ist – ähnlich wie beim visuellen Cortex – Neuronen zu erzeugen, die in bestimmten Bereichen auf Formen reagieren. Anders gesagt sollen die einzelnen Neuronen einen Teil des Bildes betrachten und hier Formen erkennen können. Stellt man sich das Eingangssignal als Matrix vor, lässt sich das Ganze einfacher visualisieren. Ein Bild mit 28x28 Pixel entspricht also einer 28x28 Matrix, siehe Abbildung 5 links. Um nun die LRFs nachzubilden, wird eine kleinere Matrix – typischerweise als Filter oder Kernel bezeichnet – über das Bild geschoben. Typische Maße für den Kernel sind hierbei 3x3, 4x4 oder 5x5. Alle Pixel, die vom Kernel überdeckt werden, werden in der Berechnung für den Pixelwert in der Ausgabe zusammengefasst (siehe Abbildung 5 rechts). Bei einer Matrix mit den Maßen 5x5 bedeutet dies, dass ein Pixel in der Ausgabe 25 Werte zusammenfasst. Die Werte innerhalb der Kernel-Matrix stellen dabei die Gewichte für jeden einzelnen Pixel dar. Zusätzlich hat jeder Pixel in der Ausgabe auch noch einen eigenen Bias.

Wandert der Kernel über das Bild, kann er um n Schritte nach rechts bzw. um m Schritte nach unten verschoben werden. Hierdurch wird bestimmt, welche Pixel in der Ausgabe zusammengefasst werden. Diese Schrittweite wird als Stride bezeichnet. Betrachtet man ein Bild mit 28x28 Pixeln, einen 5x5 Kernel und einen Stride von 1, besteht die Ausgabe aus 24x24 Pixeln. Dies liegt daran, dass man den Kernel von der Ausgangsposition nur 23 Mal um eins nach rechts bzw. unten schieben kann. Um zu verhindern, dass die Matrix im nachfolgenden Layer kleiner wird, kann Padding angewendet werden. Hierbei werden um das Bild herum weitere Pixel ergänzt. Typischerweise geschieht dies symmetrisch, allerdings ist es

[37] Qin, Z. et al., Visuell Cortex vs. CNN, 2018, S. 154.

auch möglich, unterschiedlich viele Reihen auf jeder Seite hinzuzufügen. Der Wert der hinzugefügten Pixel wird typischerweise auf 0 gesetzt.

Die Anzahl der Spalten in der Ausgabe kann mit Formel 1 berechnet werden.[38] Hierbei steht *i* für die Anzahl der Spalten in der Matrix, die das ursprüngliche Bild darstellt, p_{links} für die Anzahl von Spalten, die durch das Padding links hinzugefügt wurden (p_{rechts} dementsprechend für die Spalten rechts), k_D für die Dimensionen des Kernels und *s* für den Stride. Sind das Bild und der Kernel quadratisch und durch Padding wurden an allen Seiten gleich viele Spalten hinzugefügt, entspricht das Ergebnis auch der Anzahl an Zeilen in der Ausgabe. [39,40]

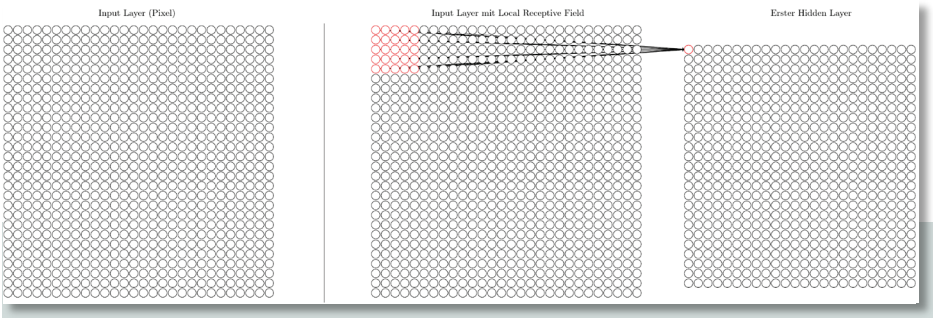


Abb. 6: Input Layer als Matrix (links) und Beispiel eines LRFs zwischen Input und 1. Hidden Layer, Quelle: In Anlehnung an Nielsen, Deep Learning, 2015, o. S.

$$o = \frac{i - k_D + p_{links} + p_{rechts}}{s} + 1 \quad (1)$$

Formel 1: Anzahl der Spalten in der Ausgabe, Quelle: In Anlehnung an Johnson, J. und Zakka, K., CS231n, 2020, o. S.

2.3.2.2 Gemeinsame Gewichte und Bias

Wie bereits erwähnt, stellen die Werte der Kernel-Matrix die Gewichte für jeden einzelnen Pixel im Bild dar. Allerdings werden außerdem die gleichen Werte innerhalb des Kernels für alle Verschiebungen verwendet. Bei einem 28x28 Layer

38 Dumoulin, V., Visin, F., Spaltenanzahl, 2018, S. 15.
39 Vgl. Nielsen, M., Determination Press, Deep Learning, 2015, o. S.
40 Johnson, J., Zakka, K., CS231n, 2020, o. S.

und einer 5x5 Matrix werden also alle 24x24 Pixel im Ausgang mit den gleichen Gewichten und Bias berechnet.

Typischerweise wird die berechnete Summe noch in eine Aktivierungsfunktion gegeben.

Das bedeutet, dass durch den Kernel überall auf dem Bild die gleichen Formen beziehungsweise Objekte erkannt werden. Man stelle sich zum Beispiel vor, dass der Kernel durch das Anpassen seiner Gewichte gelernt hat vertikale Linien zu erkennen. Dann ist es sinnvoll, dass diese vertikale Linie überall auf dem Bild erkannt werden kann. Genau dies wird durch die gemeinsamen Gewichte und Bias erreicht.

Das Element, auf welches ein Kernel reagiert, wird als Feature bezeichnet. Daher werden die Matrizen in der Ausgabe auch Feature Maps genannt. Insgesamt wird ein Layer in einem CNN, welcher Feature Maps mit gemeinsamen Gewichten erzeugt, als Convolutional Layer bezeichnet. Ein CNN soll mehr als ein Feature pro Convolutional Layer erkennen können. Daher können pro Convolutional Layer *n* verschiedene Kernel verwendet werden. Mit jedem Kernel ensteht eine neue Matrix in der Ausgabe. Die Ausgabe stellt somit eine dreidimensionale Matrix dar, bildlich gesehen kann man sich einen Stapel von zweidimensionalen Matrizen vorstellen. Grundsätzlich reagieren Kernel in Convolutional Layern, die früh im CNN vorkommen, eher auf einfache Formen wie Kanten oder Ecken. Später reagieren Convolutional Layer dann auf komplexere Formen bzw. Objekte. Auf der rechten Seite in Abbildung 5 ist dies dargestellt. Während im ersten CL einfache Linien erkannt werden, können spätere CL Objekte wie Augen erkennen.[41]

2.3.2.3 Pooling

Neben dem beschriebenen Convolutional Layer haben CNNs einen weiteren besonderen Layer. Dieser wird als Pooling Layer bezeichnet und typischerweise hinter einem Convolutional Layer angewendet.

Ein Pooling Layer vereinfacht hierbei die vorliegenden Informationen, also die Matrizen, die der vorherige Layer ausgegeben hat. Folgt der Pooling Layer auf einen Convolutional, so verwendet er die Ausgaben der Feature Maps, also die Werte, die durch die Berechnung der Aktivierungsfunktion erhalten werden.

[41] Nielsen, M., Determination Press, Deep Learning, 2015, o. S.

Beim Pooling werden mehrere Werte zusammengefasst, es wird also erneut eine Art Filter (Quadrat) über die Matrix gelegt und alle überdeckten Werte werden zusammen betrachtet. Am verbreitetsten sind hierbei das Max Pooling und das Average Pooling.

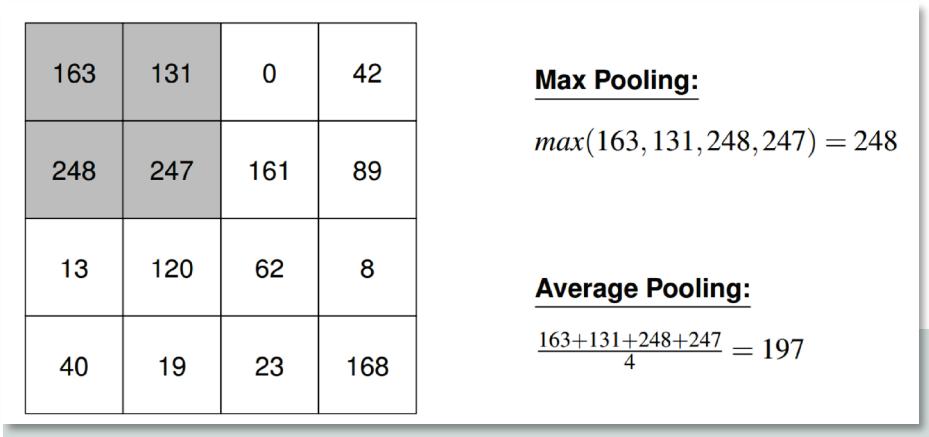


Abb. 7: Beispiel Max Pooling und Average Pooling

Beim Max Pooling wird der maximale Wert innerhalb der betrachteten Werte verwendet, während beim Average Pooling der Durchschnitt berechnet wird. Beides ist in Abbildung 7 dargestellt. Die Maße der Matrix verändern sich entsprechend der gewählten Filter Matrix, bei einer 2x2 Matrix wird die Anzahl der Spalten und Zeilen halbiert. Das Pooling wird hierbei auf jede Feature Map separat angewendet, die Anzahl der Matrizen ändert sich also nicht.

Durch Pooling wird geprüft, ob eine Form oder ein Objekt durch den vorherigen Convolutional Layer erkannt wurde. Die Grundannahme geht hierbei davon aus, dass die Information, dass dieses Element vorhanden ist, aber nicht die genaue Position relevant ist. Durch das Pooling bleibt diese Information erhalten, nur die Genauigkeit der Positionsangabe sinkt. Hierfür wird wiederum die Anzahl der Parameter stark reduziert.

In Abbildung 8 ist der Beginn eines CNN dargestellt. Hier wird ein Convolutional Layer als erster Hidden Layer verwendet und dessen Ausgaben (Feature Maps) werden als Input für den Pooling Layer (zweiter Hidden Layer) verwendet. Dessen

Ausgabe wird in einen Dense Layer gegeben, also einen Layer, der mit allen Neuronen des vorherigen Layers verbunden ist.[42]

2.3.2.4 Softmax-Funktion

CNNs, die zur Klassifizierung dienen, sind häufig so aufgebaut, dass sie für jede mögliche Klasse ein Ausgabeneuron haben. Jedes dieser Neuronen soll in diesem Fall die Wahrscheinlichkeit, mit welcher das CNN das Eingabebild dieser Klasse zuordnet, ausgeben.

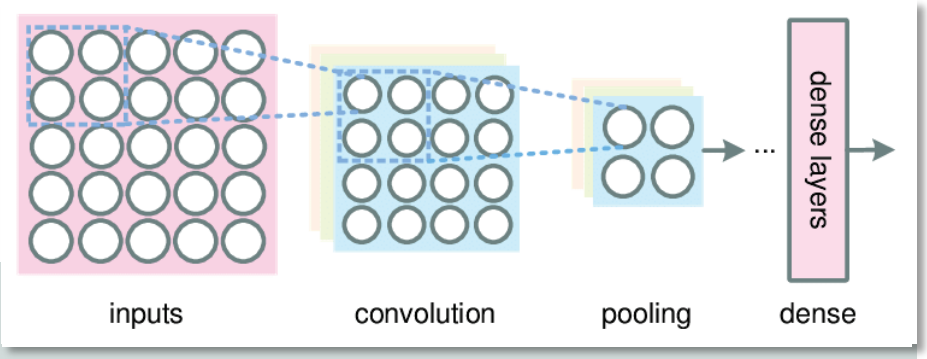


Abb. 8: Convolutional und Pooling Layer im Zusammenspiel, Quelle: Wang et al., Wireless Layer, 2017, S. 4

Die Werte innerhalb eines CNNs stellen allerdings keine Wahrscheinlichkeiten dar, sodass sie vor der Ausgabe noch normalisiert werden müssen. Hierfür wird die Softmax-Funktion angewendet. Diese normalisiert alle *k* Werte auf das Intervall (0, 1), wobei größere absolute Werte auch höhere Werte zugeordnet bekommen. Da die Summe aller *k* Werte nach der Normalisierung 1 ergibt, können die zugeordneten Werte als Wahrscheinlichkeit angesehen werden.

[42] Nielsen, M., Determination Press, Deep Learning, 2015, o. S. Dies stellt die letztendliche Codierung des Bildes darstellt, die als Vektor der softmax Funktion übergeben wird, um den Bildtyp zu entscheiden. Diese Codierung hätte man auch mit einem normalen MLP leisten können, jedoch ohne die Vorteile des CNN wie Insensitivität ggü. exakter Platzierung der Features innerhalb des Bildes.

2.3.2.5 Spatial Transformer Networks

Die Bilder in Datensätzen weisen häufig Verformungen oder Verschiebungen der relevanten Merkmale auf. Ein Beispiel hierfür ist, wenn die Straßenschilder auf den Bildern nicht immer an ähnlicher Position dargestellt werden oder sie mit leichter Rotation auftreten. Zwar fügt ein Max Pooling Layer dem CNN eine gewisse räumliche Invarianz in Bezug auf die Position von Merkmalen hinzu. Aufgrund der kleinen Bereiche, die ein Max Pooling Layer typischerweise abdeckt, ist diese allerdings nicht ausgeprägt genug, um Verluste in der Leistungsfähigkeit des CNNs zu vermeiden. Daher sind CNNs ohne weitere Hilfmittel nicht invariant gegenüber größeren räumlichen Veränderungen.[43,44]

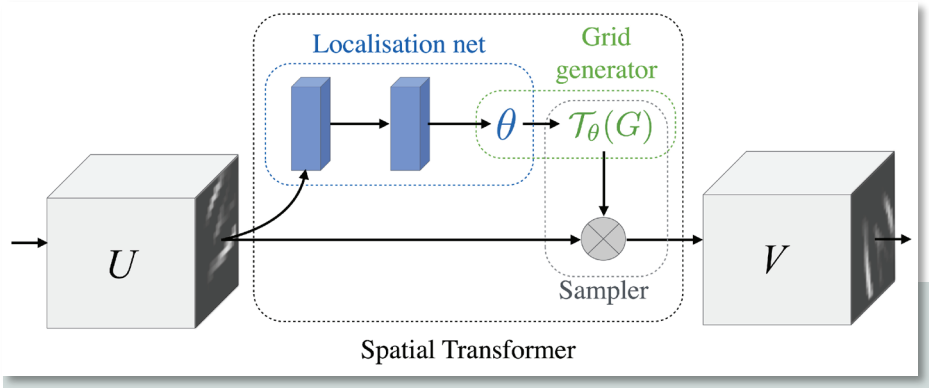


Abb. 9: Spatial Transformer Network, Quelle: Jaderberg et al., 2015, S. 3

Spatial Transformer Networks (STNs) sind eine Möglichkeit, diese Anfälligkeit gegenüber räumlichen Variationen im Datensatz teilweise auszugleichen. Ein STN stellt dabei ein Modul dar, welches an verschiedenen Stellen innerhalb eines CNNs eingebaut werden kann. Das STN lernt dabei durch einfache Vorwärtsdurchläufe und Backpropagation, welche Transformationen auf die Bilder im Datensatz angewendet werden müssen. Hierbei erzeugt es für jede Matrix, die es als Eingabe erhält, eine eigene Ausgabe. Bei Eingaben mit mehreren Feature Maps wird auf jede Feature Map die gleiche Transformation angewendet.

[43] Vgl. Cohen, T. S., Welling, M., Räumliche Veränderungen, 2015, S. 1-11.
[44] Lenc, K., Vedaldi, A., Räumliche Veränderungen, 2015, S. 991-999

Ein STN ist dabei in drei Teile unterteilt: Localisation Network, Grid Generator und Sampler (siehe Abbildung 9). Das Localisation Network erhält hierbei als Eingabe die Matrix *U* mit Breite *B*, der Höhe *H* und den Feature Maps *k* und hat die Ausgabe ϑ. Diese stellt die Parameter für die Transformation T_ϑ, die später auf die Matrix angewendet wird, dar. Die Anzahl der Parameter ϑ hängt dabei davon ab, welche Transformationen durchgeführt werden sollen. Eine Transformation, die das Zuschneiden, Verschieben, Drehen, Skalieren und Verzerren der Eingabematrix erlaubt, benötigt hierbei zum Beispiel sechs Parameter. Das Localisation Network kann jegliche Form eines NNs annehmen, sowohl als fully-connected Network als auch als CNN. Die Ausgabe des Localisation Networks wird an den Grid Generator übergeben.

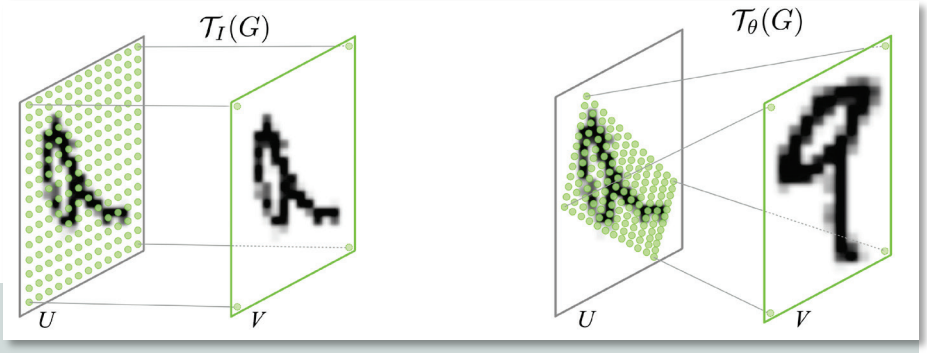


Abb. 10: Beispiel einer räumlichen Transformation mit Hilfe des erzeugten Gitters, Quelle: Jaderberg et al., 2015, S. 4

Die Ausgabepixel des STNs werden durch ein regelmäßiges Gitter $G = \{G_i\}$ mit den einzelnen Pixeln $G_i = (x^t_i, y^t_i)$ definiert. Dieses Gitter gibt die Maße für die Ausgabematrix *V* mit der angepassten Breite B^0 und Höhe H^0 sowie der gleichbleibenden Anzahl von Feature Maps *k* vor. Erzeugt wird es dabei durch den Grid Generator. Hierbei stehen (x^t_i, y^t_i) für die Zielkoordinaten im erstellten Gitter und (x^s_i, y^s_i) für die Pixelkoordinaten in der Eingabematrix. Die Koordinaten werden hierbei normiert, sodass alle Werte zwischen –1 und 1 liegen. In Abbildung 10 ist eine mögliche Transformation dargestellt, die grünen Punkte zeigen hierbei das erzeugte Gitter.

Um die Transformationen auf die Eingangsmatrix *U* durchzuführen, greift der Sampler abschließend auf diese und die Punkte des Gitters $T_\theta(G)$ zu, um die Aus-

gabematrix V zu erzeugen. Hierbei stellen die Koordinaten (x_i^s, y_i^s) in $T_\theta(G)$ räumliche Punkte dar, welchen noch Pixel der Eingabematrix zugeordnet werden müssen. Dies geschieht durch einen Kernel t.[45]

2.4 Visualisierung der Entscheidung

Es gibt eine Vielzahl von Gründen aus denen es erstrebenswert ist, ein Verständnis für die Entscheidungsgrundlage eines CNNs aufzubauen. Ein Ansatz hierfür ist es, bei einem trainierten CNN für jedes Ausgabeneuron ein Bild zu generieren, welches dessen Ausgabe maximiert. Dieses Verfahren wird Activation Maximization (AM) genannt.

Neben AM gibt es den Ansatz der Saliency Methoden. Diese versuchen, die Ausgabe des CNNs auf das Eingangsbild zurückzuspiegeln. Hierdurch soll ermöglicht werden, die Bildbereiche, die für die Aktivierung bestimmter Ausgabeneuronen verantwortlich sind, zu ermitteln. Typischerweise geschieht dies durch eine Heatmap. Bei dieser Heatmap sind Pixel, die als wichtig ermittelt wurden, besonders aktiv (z. B. höhere Werte bei einer Grauskala oder Rot bei Rot, Grün, Blau (RGB)). Wird diese Heatmap über das Originalbild gelegt, lassen sich die als relevant erkannten Bereiche des Originalbildes bestimmen.

Die im Rahmen dieser Studie angewendeten Saliency Methoden wurden basierend auf der Arbeit von Adebayo et al. ausgewählt. In dieser haben Adebayo et al. zwei Tests entwickelt, um die Vertrauenswürdigkeit von Saliency Methoden zu prüfen. Im ersten Test wurden die verschiedenen Saliency Methoden zunächst auf ein trainiertes CNN und anschließend auf ein vom Aufbau her identisches CNN mit zufällig gesetzten Gewichten angewendet. Hängt die Visualisierung tatsächlich von dem ab, was das CNN gelernt hat, sollten sich die Ergebnisse in diesem Test bei denselben Eingangsbildern unterscheiden. Im zweiten Test wurde jede Methode auf zwei vom Aufbau her identischen CNNs angewendet. Der einzige Unterschied zwischen den CNNs war, dass bei einem der CNNs die Label der Daten zufällig vertauscht wurden. Auch in diesem Test sollten sich die Ergebnisse unterscheiden. Liegt dieser Unterschied nicht vor, deutet dies darauf hin, dass die Saliency Methode die Beziehung zwischen Daten und Labeln nicht betrachtet.

[45] Vgl. Jaderberg, M. et al., STN, 2015, S. 1-9.

Verbreitete Methoden wie Guided Backpropagation[46] oder Guided GradCam[47] haben einen oder beide Tests nicht bestanden. Die im Rahmen dieser Studie verwendeten Methoden Saliency Map (SM)[48] und Grad-Class Activation Mapping (CAM)[49] wiederum konnten in beiden Tests gute Ergebnisse erzielen. Grad-CAM++ wurde nicht explizit getestet.

2.4.1 Activation Maximization

AM stellt eine Methode dar, um ein Bild als Eingabe für das CNN zu erzeugen, welches die Ausgabe eines Neurons maximiert. Dabei kann AM auf Neuronen aus jedem Layer eines CNNs angewendet werden. Weiter verbreitet ist die Anwendung auf den letzten Convolutional Layer, in welcher jedes Neuron für eine Klasse steht. In diesem Fall erzeugt AM also ein Bild, welches die Ausgabe des Neurons für die Klasse k maximiert. Anhand dieses Bildes können Rückschlüsse gezogen werden, was das CNN gelernt hat, um die Klasse k zu erkennen.

Um dieses Bild zu generieren wird ein iterativer Prozess verwendet. In diesem werden die Werte jedes Pixels im Bild in jedem Durchgang so angepasst, dass die Ausgabe des Zielneurons weiter maximiert wird. Der Prozess kann dabei in drei Schritte unterteilt werden:

1. Es wird ein Bild $x = x_0$ mit zufälligen Werten für alle Pixel generiert (Rauschen). Dieses Bild wird dann in das CNN gegeben, um einen Vorwärtsdurchlauf durchzuführen.
2. Mithilfe von Backpropagation wird die partielle Ableitung des Zielneurons bzgl. x durchgeführt. Hierbei bleiben die Gewichte und Bias im CNN unverändert.
3. Mithilfe des berechneten Gradienten werden alle Pixel in x angepasst, sodass die Aktivierung des Zielneurons weiter erhöht wird.

Der Prozess wird beendet, wenn im erzeugten Bild kein Rauschen mehr vorhanden ist. Da dies ein Zustand ist, der bei CNNs mit mehreren Layern selten erreicht wird, kann außerdem ein Schwellenwert an Rauschen definiert werden, bei welchem der Prozess stoppt.

[46] Vgl. Springenberg, J. T. et al., Guided Backpropagation, 2015, S. 1-14.
[47] Vgl. Selvaraju, R. R. et al., GradCam, 2017, S. 618-626.
[48] Vgl. Simonyan, K., Vedaldi, A., Zisserman, A., Saliency Map, 2014, S. 1-8.
[49] Vgl. Selvaraju, R. R. et al., GradCam, 2017, S. 618-626.

Ein weiteres Problem komplexer CNNs mit vielen Layern ist, dass die Muster in den späteren Layern für den Menschen zunehmend weniger interpretierbar werden.

Um die Interpretierbarkeit zu erhöhen, können verschiedene Methoden zur Regularisierung angewendet werden. Bekannte Implementierungen sind zum Beispiel l_2 Decay, Gaussian Blur, oder Mean Image Initialization.[50,51,52] Jede dieser Methoden hat ihre eigenen Effekte, l_2 Decay verhindert zum Beispiel, dass einige extreme Pixelwerte das erzeugte Bild dominieren. Die verschiedenen Methoden können auch kombiniert werden, um die verschiedenen Effekte nutzen zu können.[53]

2.4.2 Saliency Map

Während AM den Fokus auf die Visualisierung der Aktivierung einzelner Neuronen legt, betrachtet SM das gesamte CNN. Die Grundidee hinter SM ist es, die Pixel zu ermitteln, welche den größten Einfluss auf die Einordnung in eine bestimmte Klasse haben.

Nimmt man ein Bild I_0, eine Klasse c und ein trainiertes CNN mit der Bewertungsfunktion $S_c(I)$, ist das Ziel, alle Pixel nach ihrem Einfluss auf den Wert $S_c(I_0)$ zu sortieren. Im einfachen Fall einer linearen Bewertungsfunktion gibt bereits die Größe der Gewichte w an, wie wichtig die dazugehörigen Pixel für die Zuordnung zu Klasse c sind.

Die Bewertungsfunktion innerhalb eines CNNs ist allerdings eine nichtlineare Funktion, sodass dieses Prinzip nicht direkt übernommen werden kann. Geht man allerdings von dem Bild I_0 aus, kann sich dem Wert von $S_c(I)$ mit einer linearen Funktion genähert werden. Hierfür wird eine Taylorreihe erster Ordnung berechnet, wobei die Größe der Ableitung anzeigt, welche Pixel am wenigsten verändert werden müssen, um das Ergebnis der Bewertungsfunktion am stärksten zu verändern. Es wird davon ausgegangen, dass diese Pixel mit der Position des Objektes der bewerteten Klasse im Bild korrespondieren.

50 Vgl. Simonyan, K., Vedaldi, A., Zisserman, A., Saliency Map, 2014, S. 1-8.
51 Vgl. Yosinski, J. et al., Deep Visualization, 2015, S. 1-12.
52 Vgl. Wei, D. et al., Mean Image Initialization, 2015, S. 1-7.
53 Vgl. Erhan, D. et al., Activation Maximization, 2009, S. 1-14.

2.4.3 Class Activation Mapping

CAM wurde 2016 von Zhou et al. vorgestellt und wird auf bereits trainierte CNNs angewendet. Das Ziel von CAM ist es, innerhalb eines Bildes die relevanten Bereiche für die Zuordnung zu einer Klasse zu markieren. Damit CAM verwendet werden kann, muss das CNN einen bestimmten Aufbau haben. Nach dem letzten Convolutional Layer muss es einen Global Average Pooling Layer vor dem fully-connected Layer geben.

Der letzte Convolutional Layer erzeugt dabei k Feature Maps, für jeden Kernel eine. Damit hat die Ausgabe des Convolutional Layers drei Dimensionen: Die Höhe v, die Breite u und die Anzahl von Feature Maps k. Pro Feature Map wird Global Average Pooling angewendet, das heißt es wird der Mittelwert über alle Pixel in einer Feature Maps A^k gebildet.

Nachdem Global Average Pooling auf alle Feature Maps angewendet wurde, bleibt ein Vektor mit k Komponenten über. Diese werden in einen fully-connected Layer gegeben. Die Gewichte w zwischen dem Ergebnis des Global Average Pooling und dem fully-connected Layer werden dabei während des Trainingsprozesses des Netzwerkes gelernt.

Zur Erzeugung der Ergebnismatrix – bei visueller Darstellung auch Heatmap genannt – von CAM werden die gelernten Gewichte w mit den Feature Maps aus der Aktivierung des letzten Convolutional Layers A_k multipliziert. Zu beachten ist hierbei, dass CAM die Aktivierung für das aktuelle Bild und auf eine angegebene Klasse bezogen darstellt.

Das Ergebnis von CAM ist somit eine Matrix mit den Maßen der Feature Maps des letzten Convolutional Layers. Häufig wird dieses hochskaliert, sodass die Heatmap über das Originalbild gelegt werden kann.[54]

2.4.3.1 Grad-CAM

Um die Beschränkung der Architektur des CNNs durch CAM aufzuheben, haben Selvaraju et al. 2016 mit Grad-CAM die Verallgemeinerung von CAM vorgeschlagen.

Bei dieser Verallgemeinerung benötigt das CNN keinen Global Average Pooling Layer nach dem letzten Convolutional Layer mehr. Es kann eine beliebige Anzahl

[54] Zhou, B. et al., CAM, 2016, S. 1-10.

von Layern nach dem Convolutional Layer vorhanden sein, die einzige Bedingung für diese ist, dass sie differenzierbar sind. Laut den Autoren von Grad-CAM wird dabei der letzte Convolutional Layer als Grundlage für die Visualisierung verwendet, da zu erwarten ist, dass dieser das beste Gleichgewicht zwischen Interpretierbarkeit und räumlicher Auflösung bietet.

Wie bei CAM werden alle k Feature Maps gewichtet addiert, um die Ergebnismatrix zu erhalten. Der Unterschied liegt in der Berechnung der Gewichte. Während CAM die Gewichte w des CNNs verwendet, berechnet Grad-CAM eigene Gewichte α, welche auf den Gradienten basieren.

Als Grundlage für die α Werte verwendet Grad-CAM die Ausgabewerte des CNNs, bevor diese in den Softmax Layer gegeben werden (y^c). Der eigentliche Prozess von Grad-CAM kann in drei Schritte unterteilt werden.

Zunächst wird der Gradient von y^c mit Bezug auf die Feature Maps A^k berechnet. Dieser ist abhängig von dem aktuellen Eingabebild, da dieses die Grundlage für die Feature Maps und y^c liefert. Wie bei CAM ergibt ein zweidimensionales Eingabebild einen dreidimensionalen Gradienten (v, u, k). Im zweiten Schritt werden die α Werte berechnet. Jede Feature Map k erhält dabei einen Wert für die gewählte Klasse c (α_k^c). Für diesen Wert wird über die Höhe und Breite der Feature Map iteriert, um Global Average Pooling durchzuführen. Dies bedeutet, dass am Ende dieses Schrittes eine Matrix mit der Form [1,1,k] oder vereinfacht der Vektor [k] vorliegt. Jede Feature Map hat somit einen α Wert, der als Gewicht verwendet werden kann.

Im dritten Schritt wird nun die Ergebnismatrix von Grad-CAM berechnet. Diese ist die lineare Kombination der gewichteten Feature Maps, wobei das Ergebnis in die ReLU-Funktion gegeben wird. Die ReLU-Funktion sorgt dabei dafür, dass nur positive Werte einbezogen werden, da alle negativen Werte auf 0 gesetzt werden. Wie bei CAM muss die Ergebnismatrix/Heatmap hochskaliert werden, damit sie über das Eingangsbild gelegt werden kann.[55]

[55] Selvaraju, R. R. et al., GradCam, 2017, S. 618-626.

2.4.3.2 Grad-Cam++

Ein Ansatz der Weiterentwicklung von Grad-CAM ist der von Chattopadhay et al. vorgeschlagene Algorithmus Grad-CAM++. Dieser soll explizit die Beiträge der einzelnen Pixel innerhalb der Feature Maps auf die Ausgabe des CNNs abbilden. Für Details verweisen wir auf die Literatur.

3 Architektur des CNNs

In diesem Abschnitt werden alle für die Architektur des eingesetzten CNNs wesentlichen Aspekte ausführlich dargestellt. Hierbei wird im Besonderen der Komplexität der topologischen Hyperparameter Rechnung getragen.

3.1 Aufbau

Das verwendete CNN entspricht im Aufbau dem nach der Zielmetrik Accuracy performantesten CNNs aus Deep Neural Network for Traffic Sign Recognition Systems: An analysis of Spatial Transformers and Stochastic Optimization Methods von Arcos-Garcia et al..[56] Dieses belegt – Stand Dezember 2020 – den ersten Platz in der Ergebnisübersicht auf der Webseite des GTSRB. Im Folgenden wird zunächst auf STN als Komponente eingegangen, so wie von Arcos-Garcia et al. bereits umfangreich diskutiert.

3.1.1 Spatial Transformer Networks

Arcos-Garcia et al. haben in ihrem Aufsatz den Einsatz von STN innerhalb des CNNs erforscht und sind dabei zu dem Ergebnis gekommen, dass die Anwendung von drei STNs jeweils vor dem Convolutional Layer die besten Resultate für den GTSRB-Datensatz liefert. Wie beschrieben, besteht jedes STN aus drei Teilen, wobei die Architektur des Localisation Networks frei wählbar ist. Der Aufbau der drei STNs wird in Tabelle 2 nachfolgend aufgelistet.

Als Ausgabe haben alle drei STNs sechs Feature Maps mit 1x1 Pixel. Das bedeutet, es werden sechs Werte ausgegeben, welche wiederum als Parameter für die Transformation des jeweiligen STNs verwendet werden. Es reichen sechs Parameter aus, die Bilder im Eingang zuschneiden, verschieben, drehen, skalieren und verzerren zu können. Hierdurch sollten die im Datensatz und im generellen Fall von Aufnahmen von Straßenschildern auftretenden Verzerrungen ausgeglichen werden können. Hierunter fallen zum Beispiel Schilder, die um eine beliebige Achse rotiert sind oder durch Zusammenstöße mit größeren Fahrzeugen leicht verformt wurden.

[56] Arcos-García, Á., Álvarez-García, J. A., Soria-Morillo, L. M., Mehrere STN, 2018, S. 1-15.

Layer	Art	STN Layer 1			STN Layer 8			STN Layer 14		
		Feature Maps (Ausgabe)	Neuronen (Ausgabe)	Parameter	Feature Maps (Ausgabe)	Neuronen (Ausgabe)	Parameter	Feature Maps (Ausgabe)	Neuronen (Ausgabe)	Parameter
1	Eingang	3	48x48	0	200	23x23	0	250	12x12	0
2	MaxPool	3	24x24	0	200	11x11	0	150	6x6	0
3	Conv	250	24x24	19 000	150	11x11	750 150	150	6x6	937 650
4	ReLU	250	24x24	0	150	11x11	0	150	3x3	0
5	MaxPool	250	12x12	0	150	5x5	0	150	3x3	0
6	Conv	250	12x12	1 562 750	200	5x5	750 200	200	3x3	750 200
7	ReLU	250	12x12	0	200	5x5	0	200	3x3	0
8	MaxPool	250	6x6	0	200	2x2	0	200	1x1	0
9	Flatten	9000	1	0	800	1	0	200	1	0
10	Linear	250	1	2 250 250	300	1	240 300	300	1	60 300
11	ReLU	250	1	0	300	1	0	300	1	0
12	Linear	6	1	1506	6	1	1806	6	1	1806

Tab. 2: Architektur Localisation Network der STNs

Zwischen Ein- und Ausgabe liegen mehrere Layer, wobei insgesamt zwei Convolutional Layer (Layer 3 und 6) mit den dazugehörigen Aktivierungsfunktionen (Layer 4 und 7, sowie zusätzlich 11) angewendet werden. Hervorzuheben ist, dass als Aktivierungsfunktion die ReLU-Funktion verwendet wird. Dies entspricht weiterhin dem Aufbau im Paper von Arcos-Garcia et al., allerdings wurden im Rahmen der Studie auch mehrere Tests mit anderen Aktivierungsfunktionen durchgeführt. Da jedoch keine andere Aktivierungsfunktion bessere Ergebnisse als die ReLU-Funktion erzielen konnte, wurde die ReLU-Funktion beibehalten.

Als weitere Layer verwenden die Localisation Networks drei Max Pooling Layer (Layer 2, 5 und 8), eine Flatten Layer (Layer 9) und zwei Linear Layer (Layer 10 und 12). Der Flatten Layer dient dabei dazu, die Tensoren zu reduzieren, sodass sie nur noch eindimensional sind. Linear Layer stellen eine Implementierung eines fully-connected Layer mit linearer Transformation dar. Betrachtet man die Netzwerke bis Layer 11, bedeutet dies, dass bei STN 1 6912 Eingangswerte (48x48x3) auf 250 Werte, bei STN 2 105 800 Eingangswerte (23x23x200) auf 300 Werte und bei STN 3 36 000 Eingangswerte auf 300 Werte abgebildet werden. Die finale Senkung der vorhandenen Werte in Layer 12 auf sechs Werte wird in diesen Netzen durch einen fully-connected Linear Layer durchgeführt, dazu könnten theoretisch allerdings auch andere Methoden – wie z. B. k-Nearest Neighbour Classifier[57]oder Support Vector Machine (SVM)[58] – verwendet werden.

[57] Vgl. Cunningham, P., Delany, S. J., K-Nearest Neighbour Classifiers, 2020, S. 1-22.
[58] Evgeniou, T., Pontil, M., SVM, 2001, S. 249-257.

3.1.2 Features

Das gesamte CNN kann in zwei Bereiche unterteilt werden. Der erste wird hier, angelehnt an die Standards in dem verwendeten Python Framework PyTorch, als Features bezeichnet. Innerhalb dieses Bereiches befinden sich die Layer, die primär dazu dienen, die Merkmale auf den Bildern zu lernen. In der nachfolgenden Tabelle 3 ist das gesamte CNN dargestellt, der Feature-Bereich geht hierbei bis inklusive Layer 19. Die beschriebenen STNs in Layer 1, 8 und 14 gehören somit auch zu diesem Teil des CNNs.

Der Features-Bereich lässt sich in drei Blöcke unterteilen, welche die gleiche Kombination an Layer-Arten enthalten. Diese gehen von Layer 1 bis Layer 7, von Layer 8 bis Layer 13 und von Layer 14 bis Layer 19.

Zu Beginn jedes Blocks werden die aktuellen Eingaben zunächst mit einem STN transformiert. Die so aufbereiteten Werte werden anschließend mit einem Convolutional Layer mit einem 7x7 Kernel weiterverarbeitet, welcher die Merkmale lernen bzw. erkennen soll. In Abbildung 27 im Anhang ist die Visualisierung der Feature Maps vom ersten Convolutional Layer des trainierten CNNs abgebildet. Wie zu erwarten sind Ansätze von Mustern zu erkennen, allerdings bei Weitem nicht so klar wie möglicherweise erwartet. Dies kann neben der Komplexität der gelernten Schilder auch darauf zurückzuführen sein, dass die erzeugten Feature Maps nur 7x7 Pixel haben. Diese geringe Auflösung könnte dazu führen, dass die Muster schwerer zu erkennen sind.

Nach der eigentlichen diskreten Faltung wird ReLU als Aktivierungsfunktion verwendet. Wie zuvor innerhalb der STNs wurden auch hier zwei Alternativen getestet. Nach der Aktivierungsfunktion folgt ein Maxpooling Layer mit der Kernelgröße 2x2. Dieser dient dazu, die Maße der Feature Maps zu reduzieren und somit die Weiterverarbeitung auf den zusammengefassten Pixelbereichen fortzuführen. Der hierauf folgende Dropout Layer stellt den ersten Unterschied zum Aufbau von Arcos-Garcia et al. dar. In einem Folgeabschnitt wird genauer auf den Hyperparameter Dropout eingegangen. Eingefügt wurden die Dropout Layer, da die Implementierung signifikant schlechter abgeschnitten hat als die von Arcos-Garcia et al., durch sie konnte sich an diese angenähert werden. Der abschließende Layer jedes Blockes ist ein Batch-Norm Layer.

Batch-Norm wurde 2015 von Ioffe und Szegedy vorgeschlagen und normalisiert alle MiniBatches innerhalb des CNNs, sodass sie einen Mittelwert nahe 0 und eine

Standardabweichung nahe 1 haben. Dies bringt mehrere Vorteile mit sich. Zunächst sollte das CNN schneller konvergieren, was zu kürzeren Trainingszeiten führt. Allerdings steht dem der zusätzliche Rechenaufwand für die Normalisierung entgegen, sodass dies nicht der Fall sein muss. Außerdem wird das Netzwerk unempfindlicher gegenüber den initialen Gewichten, was bedeutet, dass die hier gewählten Werte nicht mehr so wichtig für den Ausgang des Trainingsprozesses sind.

Layer	Art	Feature Maps (Ausgabe)	Neuronen (Ausgabe)	Parameter	Kernel
0	Eingang	3	48x48	0	
1	STN	3	48x48	3 833 506	
2	Conv	200	46x46	29 600	7x7
3	ReLU	200	46x46	0	
4	MaxPool	200	23x23	0	2x2
6	Dropout	200	23x23	0	
7	BatchNorm	200	23x23	400	
8	STN	200	23x23	1 742 456	
9	Conv	250	24x24	800 250	4x4
10	ReLU	250	24x24	0	
11	MaxPool	250	12x12	0	2x2
12	Dropout	250	12x12	0	
13	BatchNorm	250	12x12	500	
14	STN	250	12x12	1 749 956	
15	Conv	350	13x13	1 400 350	4x4
16	ReLU	350	13x13	0	
17	MaxPool	350	6x6	0	2x2
18	Dropout	350	6x6	0	
19	BatchNorm	350	6x6	700	
20	Flatten	12 600	1	0	

Layer	Art	Feature Maps (Ausgabe)	Neuronen (Ausgabe)	Parameter	Kernel
21	Linear	400	1	5 050 400	
22	ReLu	400	1	0	
23	Linear	43	1	17 243	

Tab. 3: Architektur des CNNs ohne Details der STNs

Zusätzlich soll Batch-Norm auch höhere Learning Rates erlauben. Ioffe und Szegedy haben zusätzlich beobachtet, dass Batch-Norm ähnlich zu Dropout Layern zu einer Reduzierung von Overfitting führen und somit in einigen CNNs die Dropout Layer ersetzen kann.[59]

Da im Rahmen der hier eingesetzten CNN-Architektur die Vorteile von Batch-Norm gegenüber Dropout nicht ausgeprägt sind, wurde Dropout als auch spezifisch für diese Architektur bereits etablierter Ansatz beibehalten.

3.1.3 Classifier

Neben dem Feature-Bereich des CNNs existiert noch der Classifier-Bereich. In Tabelle 3 umfasst der Classifier-Teil die Layer 20 bis inklusive 23. Ziel des Classifier-Bereiches ist es, die Parameter der Abbildungssequenz im CNN für die Parametrisierung der abschließend eingesetzten Softmax-Funktion zu berücksichtigen. Im Fall des GTSRB entspricht die Anzahl der Ausgabeneuronen, d.h. 43, der Zahl der vorhandenen Schilderklassen. Wie in den letzten Layern der Localisation Networks in den STNs wird auch hier der Flatten Layer zur Umwandlung der mehrdimensionalen zu eindimensionalen Tensoren verwendet. Anschließend werden die nun 12 600 vorhandenen Neuronen mit zwei fully-connected Linear Layern auf zunächst 400 und anschließend 43 Neuronen reduziert. Zwischen den beiden befindet sich wiederum ein ReLU Layer als Aktivierungsfunktion.

Betrachtet man das gesamte CNN von Layer 0 bis 22, bildet dies 6912 Pixelwerte auf 400 Neuronen ab. Auch hier könnte statt des letzten Linear Layers wie innerhalb der Localisation Networks in den STNs eine andere klassifizierende Methode angewendet werden, um die 400 Neuronen auf 43 Ausgabeneuronen abzubilden.

[59] Ioffe, S., Szegedy, C., Batch Normalization, 2015, S. 448-456.

3.1.4 Parameter

Das CNN hat durch seine Architektur eine Vielzahl von lernbaren Parametern, also Gewichten. In den Tabellen 2 und 3 sind für jeden Layer die Anzahl der vorhandenen Parameter gelistet. Der Features-Bereich umfasst insgesamt 9 557 718 Parameter, während der Classifier-Bereich 5 057 643 Parameter enthält. Im Letzterem stammt der Großteil dieser Parameter aus dem ersten fully-connected Linear Layer, welcher 12 600 Neuronen mit 400 Neuronen verbindet (5 050 400 Parameter). Hier sieht man den Vorteil von CNNs: Obwohl der fully-connected Layer – welcher typischerweise die Grundlage für alle Verbindungen in NNs darstellt – erst spät im Netzwerk auftritt, hat er noch eine hohe Anzahl von Parametern. Ein NN, welches ausschließlich aus fully-connected Layern bestehen würde, hätte dementsprechend eine um mehrere Faktoren größere Zahl von Parametern als dieses CNN und wäre mit aktueller Hardware nicht mehr sinnvoll zu trainieren. Trotz aller Vorteile von CNNs im Bereich Parameterreduktion weist das erstellte CNN immer noch insgesamt 14 615 361 Parameter auf. Dies bedeutet, dass im Rahmen des Trainings des CNNs 14 615 361 Werte bei jeder Aktualisierung der Gewichte mit Backpropagation neu berechnet werden müssen. Dies stellt einen enormen Aufwand dar, welcher sich auch in der Trainingszeit widerspiegelt.[60] Auch wenn die Anwendung eines bereits trainierten CNNs deutlich weniger anspruchsvoll ist als das Trainieren an sich, sollte dieser Rechenaufwand im Auge behalten werden, vor dies bei der Verwendung in Fahrzeugen in minimaler Zeit geschehen muss.

3.2 Hyperparameter

Ein CNN hat einige Parameter, welche nicht im Rahmen des Trainingsprozesses durch Backpropagation oder ähnliche automatische Verfahren bestimmt werden können. Diese Parameter werden als Hyperparameter bezeichnet und müssen vor Beginn des Lernprozesses festgelegt werden. In diesem Abschnitt werden die verwendeten Hyperparameter beschrieben, ebenso wie Alternativen, die im Rahmen des Aufbaus des CNNs getestet und verworfen wurden. Die einzelnen Hyperparameter werden hierbei alphabetisch aufgelistet.

[60] Im Rahmen dieser Studie wurde Google Colab mit einer Nvidia Tesla P100-PCIE mit 16GB Speicher verwendet. Eine Trainingsepoche (Batch Size 50) benötigte hierbei im Durchschnitt zwischen 3,5 und 4 Minuten.

3.2.1 Aktivierungsfunktionen

Wie im vorherigen Abschnitt beschrieben, wird im gesamten CNN die ReLU-Funktion als Aktivierungsfunktion verwendet. Auch wenn dies dem Aufbau des CNNs von Arcos-Garcia et al. entspricht, wurden noch weitere Aktivierungsfunktionen getestet.

Als naheliegende Alternative zur ReLU-Funktion wurde zunächst die Leaky ReLUFunktion angewendet. Die Leaky ReLU-Funktion setzt anders als die ReLU-Funktion nicht alle negativen Werte auf 0, sondern arbeitet bei diesen mit einem kleinen, negativen Parameter ω (z. B. $\omega = 0{,}001$) als Multiplikator. Die Grundidee hierbei ist, dass es im Training mit der ReLU-Funktion durch das Abschneiden negativer Werte dazu kommen kann, dass Neuronen in künftigen Trainingsdurchgängen ignoriert werden. Entscheidend ist hierbei, dass ein Neuron auf den Wert 0 gesetzt werden kann. Diese Möglichkeit ist im Normalfall gewollt, allerdings kann es bei einzelnen Neuronen dazu kommen, dass sie zwischenzeitlich negative, aber sonst überwiegend positive Werte erzeugen. Durch den einmalig negativen Wert und somit das Setzen auf 0 geht der positive Einfluss im CNN allerdings verloren. Dieser Effekt wird auch dying ReLU genannt, da unter Umständen eine Vielzahl von Neuronen betroffen sein kann. Hierdurch kann die Lernfähigkeit des Netzes erheblich beschränkt werden.[61]

In den Tests mit Leaky ReLU-Funktion konnten keine Verbesserungen der Performance des CNNs festgestellt werden. Es ist daher davon auszugehen, dass im CNN kein dying ReLU Problem vorliegt oder es durch leaky ReLU-Funktion nicht entschieden verhindert wird.

Als weitere Alternative wurde die tanh-Funktion getestet. Diese wurde der Sigmoid-Funktion vorgezogen, da zwar beide einen geschlossenen Wertebereich haben, die tanhFunktion zeitgleich aber um den Nullpunkt zentriert ist. Mit dem geschlossenen Wertebereich steht die tanh-Funktion hierbei im Kontrast zu ReLU-Funktion, welches im positiven Bereich nach oben offen ist. Da die Ergebnisse mit der tanh-Funktion nicht besser als zuvor waren, aber der Optimierungsprozess deutlich langsamer, wurde entschieden, weiterhin die ReLU-Funktion zu verwenden. Der trägere Optimierungsprozess im Zusammenspiel mit GD bzw. in dieser Studie SGD deckt sich dabei mit anderen Publikationen, zum Beispiel ImageNet

[61] Maas, A. L., Hannun, A. Y., Ng, A. Y., Leaky ReLU, 2013, S. 1-6.

Classification with Deep Convolutional Neural Networks von Krizhevsky et al., bei denen die ReLU-Funktion beim Training ungefähr sechsmal schneller war.[62]

3.2.2 Batch Size

In SGD und anderen Lernalgorithmen werden die Gewichtsanpassungen nicht am Ende einer Epoche durchgeführt, sondern bereits nach einer fest definierten Anzahl von Beispielen. Diese Gruppe von Beispielen wird als Batch bezeichnet und die Anzahl an Beispielen innerhalb dieser als Batch Size. Im Rahmen dieser Studie wurden verschiedene Batch Sizes, die das Vielfache von 50 darstellen, getestet. Innerhalb der Tests hat das CNN die besten Resultate erzeugt, wenn die Batch Size auf 50 festgelegt wurde.

3.2.3 Dropout

Innerhalb des Trainings eines NNs besteht die Gefahr, dass ein Netzwerk sich zu sehr an die Trainingsdaten anpasst. Es lernt hierbei die spezifischen Unterschiede der Trainingsdaten statt, wie erhofft, generalisierbare Merkmale für die einzelnen Klassen. Dieses als Overfitting bezeichnete Problem führt dazu, dass das CNN bei unbekannten Eingabewerten schlechter abschneidet, als es könnte, wenn kein Overfitting vorliegen würde. Eine Methode, um dies zu verhindern, sind Dropouts. Bei diesen werden in jedem Durchgang ausschließlich während des Trainings zufällige Neuronen und ihre Gewichte innerhalb des Netzwerkes auf 0 gesetzt. Hierdurch kann das CNN keine zu große Abhängigkeit von einzelnen Pfaden durch das Netz bilden, da Teile hiervon immer wieder wegfallen. Dropouts werden häufig – wie auch in dieser Studie – durch Dropout Layer integriert. Diese stellen Layer in dem CNN dar, die nur die Aktivierungen der Eingänge weiterleiten, welche nicht durch die Dropout-Funktion ausgewählt wurden. Bei Dropouts stellt die Wahrscheinlichkeit, mit denen ein Neuron auf 0 gesetzt wird, den Hyperparameter dar.[63]

Im Rahmen der Tests wurde die Dropout-Rate als letzter Hyperparameter ermittelt. Mit den anderen, bereits festgesetzten Hyperparametern wurden für die Dropout-Raten Tests im Bereich von 0 bis 0,5 durchgeführt. Das beste Ergebnis wurde bei einer Dropout-Rate von 0,45 erreicht.

62 Vgl. Krizhevsky, A., Sutskever, I., Hinton, G. E., Tanh vs. ReLU, 2017, S. 84-90.
63 Srivastava, N. et al., Dropout, 2014, S. 1929-1958.

3.2.4 Epochen

Die Anzahl der Epochen bestimmt, wie häufig das CNN alle Trainingsdaten durchlaufen soll, bis das Training beendet wird. Desto mehr Epochen ein Netzwerk trainieren kann, umso besser können seine Resultate werden. Allerdings erreicht es hierbei irgendwann einen Zustand, in dem das Anpassen der Gewichte zu keiner relevanten Verbesserung mehr führt.

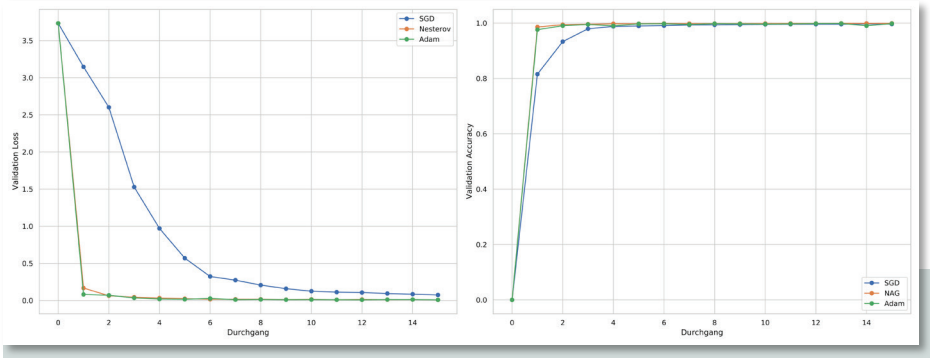


Abb. 11: Gegenüberstellung der Verlustfunktionen anhand von Validation

In Abbildung 11 lässt sich dies bei den späteren Epochen beobachten; die einzelnen Graphen verlaufen nahezu horizontal, es wird also kaum noch Verbesserung erzielt. Im Rahmen dieser Studie wurden die CNNs über 15 Epochen trainiert, wobei eine Abwägung zwischen erreichbarem Verlust bzw. erreichbarer Accuarcy und der Trainingszeit getroffen wurde.

3.2.5 Learning Rate

Die Learning Rate steuert als Schrittweite die Geschwindigkeit, mit welcher die Gewichte im Rahmen des Trainings des CNNs angepasst werden. Sie muss schrittweise verkleinert werden, um die Konvergenz des Lernalgorithmus zu garantieren.

Sie hat dabei Einfluss auf zwei Faktoren des Trainingsprozesses. Zunächst kann sie die Trainingszeit verkürzen oder verlängern. Eine kleinere Learning Rate führt hierbei zu längeren Trainingszeiten, da mehr Schritte benötigt werden, bis das CNN konvergiert. Auf der anderen Seite kann die Learning Rate die erreichte Accuracy dadurch beeinflussen, dass sie durch die Schrittgröße dafür sorgen kann,

dass die Optimierung in einem Tal hängen bleibt oder auch den Tiefpunkt nicht erreichen kann. Letzteres kann dadurch geschehen, dass die Schrittweite zu groß ist, um von den erreichbaren Positionen den Tiefpunkt zu erreichen. Dieser wird von allen Seiten aus immer überschritten. Die Gefahr hierfür steigt mit einer höheren Learning Rate.

Somit ist die Auswahl der Learning Rate eine Wahl zwischen einer höheren Learning Rate, welche den Lernprozess beschleunigen würde, und einer niedrigeren Learning Rate, welche zu besseren Ergebnissen führen könnte. Da es aber zeitgleich zu nicht optimalen Ergebnissen kommen kann, wenn die Learning Rate zu klein ist, ist die Auswahl der Learning Rate nicht trivial.

Um die verwendete Learning Rate festzulegen, wurden mit einer beschränkten Anzahl von Epochen mehrere Learning Rates im Bereich von $1 \cdot 10^{-4}$ bis $1 \cdot 10^{-1}$ angewendet. Im Zusammenspiel mit dem verwendeten Lernalgorithmus hat hierbei eine Learning Rate von 0,001 am besten abgeschnitten. Dies deckt sich mit den Ergebnissen von Arcos-Garcia et al.

3.2.6 Lernalgorithmus

Um die lernbaren Parameter des Netzwerkes anzupassen, wird auf einen Lernalgorithmus zurückgegriffen. Neben dem weitverbreiteten GD bzw. SGD gibt es noch eine Vielzahl von anderen Lernalgorithmen; im Rahmen dieser Studie wurden noch zwei weitere Alternativen in die Tests mit einbezogen. Die erste Alternative stellt eine Variation von GD dar. Wie bereits erläutert, berechnet GD den neuen Wert eines Gewichts dadurch, dass vom bisherigen Gewicht w_t-1 der ermittelte Gradient, hier für eine bessere Übersichtlichkeit mit g_t-1 bezeichnet, subtrahiert wird. Der Gradient wird hierbei zusätzlich noch durch die Learning Rate η skaliert.

$$w_t = w_t-1 \; -\eta g_t-1 \tag{2}$$

Formel 2: Berechnung des neuen Gewichts in Gradient Descent, in Anlehnung an Rumelhart, D. E., et al., Backpropagation, 1968, S. 535

Es kann in Funktionsbereichen, in denen viele lokale Minima und Maxima existieren, dazu kommen, dass GD der falschen negativen Steigung folgt und somit einen Schritt in eine nicht optimale Richtung wählt. Dies kann durch einen Momentumterm ausgeglichen werden. Die Grundidee dahinter ist, dass auch die vorherige Gewichtsanpassung noch mit in Betracht gezogen wird. Zeigt diese in

die gleiche Richtung wie die neu Berechnete, wird der Schritt vergrößert. Ändert sich die Richtung allerdings, wird die Schrittweite gedämpft, was zu einem glatteren Optimierungsverlauf führt. Das Momentum stellt hierbei einen weiteren Hyperparameter dar, welcher angibt, wie stark der Einfluss der vorherigen Anpassung sein soll. In der nachfolgenden Formel stellt μ diesen Hyperparameter dar, während Δw_t die vorhergehende Gewichtsanpassung darstellt.[64]

$$\Delta w_t+1 = -\mathbf{g}_t + \mu \Delta w_t \quad (3)$$

Formel 3: Anwendung von Momentum in Gradient Descent, in Anlehnung an Rumelhart, D. E., et al., Backpropagation, 1968, S. 535

Eine weitere Verfeinerung stellt der Nesterov Accelerated Gradient Descent (NAG) dar.[65] Hierbei wird der Gradient berechnet nachdem der Momentum Term addiert wurde. Für diesen Lernalgorithmus liegt zudem ein Konvergenzbeweis vor.

Die dritte getestete Alternative des Lernalgorithmus ist Adam. Dieser baut auf den beiden Lernalgorithmen AdaGrad[66] und RMSProp[67] auf und berechnet eine adaptive Lernrate für jeden Parameter. Hierdurch können zu Beginn des Trainings größere Schritte, später in der Nähe des (lokalen) Optimums kleinere Schritte vollzogen werden. Zur Berechnung verwendet Adam dabei den exponentiell verfallenden Durchschnitt der vergangenen Gradienten. Für die mathematische Herleitung wird aufgrund des Umfangs auf das Paper von Kingma und Ba verwiesen.[68]

Alle drei Lernalgorithmen wurden auf den GTSRB-Datensatz angewendet. In der vorigen Abbildung 11 sind die Verläufe des Validation Loss und der Validation Accuracy dargestellt. Die genauen Werte sind im Anhang in den Tabellen 6 und 7 zu finden. Beim Training wurde eine Dropout Rate von 0,01 innerhalb des Netzwerkes verwendet. Das Momentum für NAG wurde auf 0,9 und die Betas von Adam auf 0,9 und 0,999 festgelegt. Bei SGD und NAG lag die Learning Rate bei 0,001, für Adam bei 0,0001. Es ist zu erkennen, dass von den drei Lernalgorithmen Adam am schnellsten lernt; so liegen hier nach der ersten Epoche – sowohl beim Loss als auch bei der Accuracy – die besten Ergebnisse vor. NAG holt diesen Vorsprung allerdings schon nach der zweiten Epoche auf, anschließend bleiben beide auf einem ähnlichen Niveau. SGD schneidet am schlechtesten ab; es

64 Vgl. Rumelhart, D. E., Hinton, G. E., Williams, R. J., Backpropagation, 1986, S. 533-536.
65 Nesterov, Y., Nesterov-Momentum, 1983, S. 372-376.
66 Duchi, J., Hazan, E., Singer, Y., AdaGrad, 2011, S. 2121-2159.
67 Hinton, G., RMSProp, 2011, o. S.
68 Kingma, D. P., Ba, J., Adam, 2015, S. 1-15.

braucht acht Epochen, um bei der Accuracy zu den anderen beiden aufzuschließen. Bei dem Validation Loss erreicht SGD die Werte von Adam und NAG gar nicht. Da alle drei Lernalgorithmen eine sehr hohe Validation Accuracy erreichen, lässt sich erkennen, dass die Struktur des CNNs und die Beschaffenheit des Datensatzes zu einer gut optimierbaren Funktion führen. Dass die beiden Lernalgorithmen mit einer Implementierung des Momentums schneller und besser lernen, spricht allerdings dafür, dass es viele lokale Minima und Maxima gibt. Hier zeigt sich die unterschiedliche Robustheit von SGD auf der einen Seite und Adam sowie NAG auf der anderen Seite in Bezug auf lokale Minima. Die hohe erreichte Validation Accuracy von SGD zeigt zwar, dass er sich in diesen nicht festgelaufen hat, allerdings haben die Schritte in diese Bereiche Zeit und Rechenleistung gekostet. Betrachtet man die Werte in den Tabellen 6 und 7, erreicht NAG in beiden marginal bessere Werte als Adam. Da die Trainingszeit zudem kürzer ist, wurde sich dafür entschieden, NAG als Lernalgorithmus zu verwenden.

3.2.7 Verlustfunktion

Damit eine Lernalgorithmus angewendet werden kann, um die Gewichte innerhalb des Netzes anzupassen, muss der Fehler, den das CNN erzeugt hat, berechnet werden. Hierfür wird eine Verlustfunktion verwendet. Die einfachste Verlustfunktion basiert auf dem mittleren quadratischen Fehler. Neben dieser wurde die Cross-Entropy verwendet.

4 Implementierung

Dieses Kapitel gibt einen Überblick über die zum Implementieren des praktischen Bereichs dieser Studie verwendete Programmiersprache sowie die Spezifika der Bibliotheken, die eingesetzt wurden.

4.1 Allgemeines

Im Rahmen dieser Studie wurden alle praktischen Arbeitsschritte mit der Programmiersprache Python 3.8 umgesetzt. Hierzu gehören das Auf- und Vorbereiten des Datensatzes, das Erstellen und Trainieren des CNNs, das Erzeugen der Visualisierungen des Gelernten sowie die weiteren Auswertungen und Tests. Der verwendete Code ist auf GitHub[69] verfügbar. In diesem Abschnitt wird auf die wichtigsten Arbeitsschritte und verwendeten Bibliotheken eingegangen. Eine Übersicht über die verwendeten Bibliotheken ist in Tabelle 8 im Anhang zu finden. Ausgeführt wurde der Code auf Google Colab[70], einer Plattform zum Ausführen von Jupyter Notebooks. Diese stellen eine interaktive Umgebung zum Ausführen von Python bereit.[71] Google Colab verwendet als Betriebssystem – Stand Dezember 2020 – Ubuntu in Version 18.04.05 LTS. Die erzielten Ergebnisse basieren somit auf einem Unix-Kernel. Es wurde festgestellt, dass sich die Ergebnisse auf einem Windows 10 (Version 20H2, Build 19043.685) System nicht 1:1 reproduzieren lassen. Selbst die Anwendung des bereits trainierten CNNs erzielt schlechtere Ergebnisse. Auf anderen Unix-basierten Systemen (MacOS 11.0.1 und Ubuntu 20.04 LTS) konnten die Ergebnisse wiederum reproduziert werden. Dies könnte auf die zur Implementierung des CNNs verwendete Bibliothek PyTorch zurückzuführen sein. In ihrer Dokumentation wird bemerkt, dass die Reproduzierbarkeit über verschiedene Plattformen nicht garantiert werden kann.[72]

4.2 Aufbereitung des Datensatzes

Der vom Institut für Neuroinformatik der Ruhr-Universtität Bochum bereitgestellte GTSRB Datensatz besteht aus einem Trainings- und einem Testdatensatz. Vor dem Trainieren des CNNs wurden einige vorbereitende Schritte durchgeführt.

69 https://github.com/RobinMaas95/GTSRB_Visualization
70 Google, Colab, 2020, o. S.
71 IPython development team, Jupyter Notebook, 2020, o. S. .
72 Torch Contributors, Reproduzierbarkeit, 2020, o. S.

Die in diesem Abschnitt beschriebenen Schritte sind dabei ausschließlich auf den Trainingsdatensatz angewendet worden. Ziel dieser Schritte ist es, die Anzahl der unterschiedlichen Beispiele pro Klasse innerhalb des Datensatzes zu erhöhen und die Anzahl der Beispiele pro Klasse auszugleichen. Ersteres führt dazu, dass das CNN mehr Beispiele pro Klasse hat, wodurch es diese besser generalisieren können sollte. Zweiteres sollte zu einer besseren Accuracy des CNNs führen, da keine Klasse aufgrund ihres überproportionalen Auftretens im Trainingsdatensatz bevorzugt wird.

Das Ausgleichen der unterschiedlichen Quantitäten der Beispiele in den einzelnen Klassen durch die Vervielfältigung von Beispielen wird als Oversampling bezeichnet. Buda et al. haben festgestellt, dass Oversampling Alternativen wie Undersampling (Entfernen von Beispielen in Klassen mit vielen Beispielen) meistens überlegen ist. Außerdem konnten Sie nachweisen, dass Oversampling den Rückstand der kleineren Klassen vollständig ausgleichen kann. Zuletzt konnten sie zeigen, dass Oversampling bei CNNs nicht zu Overfitting führt.[73]

In Abbildung 12 ist jede Klasse im Datensatz dargestellt. Bei den Bildern handelt es sich in diesem Fall um Stockfotos und nicht um Beispiele aus dem Datensatz selbst. Die Stockfotos entstammen hierbei dem VzKat und sind gemeinfrei.[74] Um die Anzahl von Beispielen innerhalb der Klassen zu erhöhen, wurden die Beispiele einiger Klassen gespiegelt. Hierbei kann eine gespiegelte Klasse durch das Spiegeln an der horizontalen oder vertikalen Achse entweder ein neues Beispiel für sich selbst darstellen oder aber auch eine andere Klasse ergeben. Die Klassen 11, 12, 13, 15, 17, 18, 22, 26, 30, 35 und 40 ergeben sich bei horizontaler Spiegelung selbst. Das Gleiche gilt für die Klassen 1, 5, 12, 15 und 17 bei vertikaler Spiegelung. Zuletzt ergeben folgende Klassenpaare sich gegenseitig: 19+20, 33+34, 36+37 und 38+39. Um die Spiegelung durchzuführen, wurden die beiden Funktionen fliplr und flipud der Bibliothek numpy (Version 1.19.0) verwendet.[75]

[73] Buda, M., Maki, A., Mazurowski, M. A., Upsampling, 2018, S.249-259.
[74] Bundesrepublik Deutschland, VzKat, 2017, o. S.
[75] NumPy community, Numpy, 2020, 559 f.

Abb. 12: Übersicht über alle Klassen im Datensatz

Nach der Spiegelung der Beispiele hat die Klasse mit den meisten Beispielen 4438 Bilder und die Klasse mit den wenigsten 209. Für das Upsampling wurde die Anzahl von 4500 Bildern als Ziel ausgewählt, da dies den Rückstand der kleineren Klasse vollständig ausgleicht.

Im letzten Vorbereitungsschritt wurden die Bilder aus allen Klassen mit der *rotate*-Funktion der Bibliothek Pillow (Version 8.0.1) entlang der x-Achse rotiert.[76] Hierbei wurde die Rotation basierend auf einer Normalverteilung mit dem Erwartungswert 0 Grad und einer Standardabweichung von 20 Grad durchgeführt. Diese Verteilung wurde basierend auf der Annahme ausgewählt, dass Schilder mit leichten Rotationen bei realen Beispielen häufiger vorkommen sollten als solche mit stärkeren Rotationen. Bilder mit zu starker Rotation – hier als 20 Grad angenommen – sollten sehr selten auftreten, da Schilder, die derart stark rotiert sind, ver-

[76] Clark, A., Pillow, 2020, S. 65.

mutlich wieder von den Straßenmeistereien o.a. gerichtet werden. Um den Einfluss der STNs evaluieren zu können, wurde darauf verzichtet, im Trainingsdatensatz auch Beispiele mit einer Rotation entlang der z-Achse einzuführen. Da die STNs somit nicht mit dieser Transformation trainiert wurden, sollte ein Unterschied in der Anfälligkeit für die Rotation an den beiden Rotationsachsen feststellbar sein.

4.3 Zuschneiden des Datensatzes

Bei der Auswertung der ersten Ergebnisse der Visualisierungsalgorithmen wurde festgestellt, dass das CNN im Original-Datensatz häufig den Hintergrund als wichtigsten Bereich für seine Entscheidung verwendet. Hierauf wird in der späteren Auswertung der durchgeführten Versuche genauer eingegangen.

Der GTSRB stellt die Koordinaten für die genauen Positionen der Schilder innerhalb der einzelnen Bilder bereit. Daher wurde entschieden, die Bilder so zuzuschneiden, dass die Hintergrundfläche auf den Bildern minimiert wird. Der Prozess des Zuschneidens ist dabei in die folgenden Schritte unterteilbar und wurde auf die Trainingsdaten und den Testdatensatz angewendet:

- Auslesen der Anmerkung

Im GTSRB gibt es für jeden Ordner eine Comma Separated Values (CSV) Datei, in der verschiedene Informationen zu jedem Bild innerhalb des Ordners enthalten sind. Für das Zuschneiden der Bilder sind die Spalten Roi.X1, Roi.Y1, Roi.X2 und Roi.Y2 relevant. Diese vier Werte geben die Koordinaten eines Rechtecks an, welches das Schild umfasst.

- Ausschneiden des Rechtecks

Nachdem die Position des Rechtecks mit dem Schild innerhalb des Bildes bekannt ist, werden durch die *crop*-Funktion der Bibliothek Pillow alle Pixel außerhalb des Rechtecks entfernt. Die *crop*-Funktion bekommt hierfür die vier Koordinaten aus dem vorherigen Schritt übergeben.[77]

- Skalieren der Bilder

[77] Vgl. Clark, A., Pillow, 2020, S. 59.

Für die Eingabe in das CNN werden Bilder mit dem Format 48x48 Pixel benötigt. Um dieses Format nach dem Zuschneiden der Bilder zu erhalten, werden sie mittels der Pillow *resize*-Funktion und einem Lanczos-Filter[78] hochskaliert. Zwar ist dieser im Vergleich zu anderen Algorithmen langsamer, allerdings liefert der Skalierungsprozess dafür sehr gute Ergebnisse.[79]

Aufgrund der Form der Schilder kann ein Rechteck diese nicht perfekt erfassen, ohne noch Hintergrund zu enthalten. Daher ist auch im zugeschnittenen Datensatz noch etwas Hintergrund vorhanden. Dieser macht allerdings – im Vergleich zu den Schildern an sich – nun deutlich weniger Fläche als zuvor aus.

4.4 Bau des CNN

Das CNN ist mit Hilfe von PyTorch (Version 1.7.0), einer Bibliothek für maschinelles Lernen, implementiert worden. Zusätzlich wurde PyTorch Lightning (Version 1.1.0) verwendet. Dies ist ein Wrapper um PyTorch, welcher eine einheitliche Struktur für den Modellaufbau sowie zusätzliche Funktionen bereitstellt. Die Layer innerhalb des CNNs sind somit die PyTorch Implementationen der in Tabelle 3 genannten Layer-Arten.

PyTorch Lightning bringt eine *Trainer*-Methode mit sich, welche viele Aufgaben innerhalb des Trainings und Testens eines CNNs abstrahiert. Zu diesen Aufgaben zählen unter anderem die Anpassung der Gewichte, das Berechnen der Trainings/Validierungs und Test Accuracy, sowie das Speichern der Gewichte zur Sicherung des Gelernten. Um diese Aufgaben ausführen zu können, muss das erstellte Modell gewisse Methoden implementiert haben. Im Folgenden sind die – im Rahmen dieser Studie relevanten – Methoden beschrieben.

- forward

Gibt an, wie ein Vorwärtsdurchlauf innerhalb des CNNs aussieht. Im Rahmen der Studie wird die Eingabe hierbei zunächst in den Features-Bereich gegeben und dessen Ergebnis anschließend in den Classifier-Bereich. Dadurch wird die Eingabe verarbeitet und das Netzwerk erzeugt die zugehörige Vorhersage.

- configure_optimizer

Einstellung des zu verwendenden Lernalgorithmus. Hier wird NAG angewendet.

[78] Vgl. Duchon, C. E., Lanczos Filter, 1979, S. 1016-1022.
[79] Clark, A., Pillow, 2020, S. 23.

- training_step/validation_step/test_step

Beschreibt wie ein Schritt in einer der drei Phasen aussehen soll. In allen drei Phasen wird der Verlust berechnet, nachdem das Netzwerk die Eingabe erhalten und seine Zuordnung in eine Klasse durchgeführt hat (siehe forward). In dieser Methode wird definiert, welche Verlustfunktion in welcher Phase verwendet werden soll (hier überall Cross Entropy).

- validation_epoch_end/test_epoch_end

Am Ende einer Epoche kann auf alle Ergebnisse innerhalb dieser zugegriffen werden. Im Rahmen dieser Studie werden hier der durchschnittliche Verlust und die durchschnittliche Accuracy berechnet. Diese Werte unterliegen geringeren Schwankungen als die Resultate der einzelnen Vorwärtsdurchläufe und eignen sich damit besser, um den Lernfortschritt des Netzwerkes nachzuvollziehen. validation_epoch_end wird hierbei während des Trainings – am Ende einer Trainingsepoche – aufgerufen. test_epoch_end wird nur aufgerufen, wenn die Testmethode des Trainers aufgerufen wird.

- train_dataloader/val_dataloader/test_dataloader

In diesen Methoden wird definiert, wie die Bilder als Eingabe für das CNN geladen werden. Zusätzlich können innerhalb dieser Methoden Transformationen auf die geladenen Bilder durchgeführt werden. Bei dem erstellten CNN werden die Bilder für die Trainingsdurchläufe zufällig einer Verschiebung (Faktor 0,2), Scherung (Bereich von 20 Grad) oder Skalierung (Faktor 0,8 - 1,2) ausgesetzt. Außerdem wird jedes Bild normalisiert. Mittelwert und Standardabweichung aller Bilder im Trainingsdatensatz werden dazu verwendet. Dabei werden für beide drei Werte übergeben, jeweils einer für jede Feature Map im RGB-Eingangsbild.[80]

Wie anhand der Namen einiger Methoden des Trainers zu erkennen ist, unterscheidet PyTorch zwischen Trainings-, Validierungs- und Testdatensatz. [81]

Um den noch fehlenden Validation-Datensatz zu erzeugen, wurde die Python Bibliothek split-folders (Version 0.4.3) verwendet. Diese teilt die zuvor zufällig sortierten Bilder innerhalb der einzelnen Ordner anhand des vorgesehenen Training-Validierung-Splits auf. Für diese Studie wurde dabei ein Verhältnis von 70 % Training zu 30 % Prozent Validation ausgewählt. Daraus ergibt sich, dass für jede Klasse 3150 Bilder zum Training und 1350 Bilder zum Validieren genutzt werden.

[80] Vgl. Torch Contributors, Normalize, 2020, o. S.
[81] Vgl. Falcon, W., PyTorch Lightning, 2020, S. 79-109.

Um eine Reproduzierbarkeit trotz der zufälligen Sortierung vor dem Aufteilen herzustellen, kann ein Random Seed übergeben werden.

Wird die Funktion mehrfach mit dem gleichen Random Seed aufgerufen, sortiert sie die Beispiele vor dem Aufteilen immer gleich. Im Rahmen der vorliegenden Studie wurde der Seed auf 2020 gesetzt.[82]

4.5 Visualisierungsalgorithmen

Um den Lernprozess des trainierten CNNs zu visualisieren, wurden vier Visualisierungsalgorithmen angewendet. Diese sind die beschriebenen Algorithmen AM, SM, Grad-CAM und Grad-CAM++.

Für alle vier Algorithmen existiert bereits eine Vielzahl von Implementierungen, die kompatibel mit PyTorch Modellen sind. Daher wird auf eine Auswahl dieser Algorithmen zurückgegriffen und auf eine eigene Implementierung verzichtet.

Für AM und SM wurde auf die Bibliothek nn_interpretability[83] zurückgegriffen. Diese enthält die Implementierung einer Vielzahl von Visualisierungsalgorithmen und ist über GitHub verfügbar.[84] Zwar enthält diese Bibliothek auch eine Implementierung von Grad-CAM, allerdings nicht von Grad-CAM++. Daher wurde hierfür die Bibliothek SmoothGradCAMplusplus[85] von Yuchi Ishikawa gewählt, welche beide zur Verfügung stellt. SmoothGradCAMplusplus ist ebenfalls über GitHub verfügbar.[86]

4.6 Rotation entlang der Achsen

Es kann vorkommen, dass Straßenschilder durch äußere Einflüsse ihre vorgesehene Ausrichtung verändern. So kann zum Beispiel eine Befestigung beschädigt werden, was dazu führt, dass das Schild nach links oder rechts rotiert. Diese Rotation wird im Folgenden als Rotation entlang der x-Achse bezeichnet. Außerdem kann das Schild oder der ganze Mast – hier vor allem bei temporären Schildern, welche nicht fest im Boden eingelassen sind – einseitig nach hinten verschoben

82 Filter, J., Split-Folders, 2020, o. S.
83 Verwendete Commit: 3f494ae1835baa1389097c0afc71671393e50ca2
84 Vgl. hans66hsu, Nn_interpretability, 2020, o. S.
85 Verwendete Commit: e728e5f6b1fde4fb416efb2509e1583e1cdc0a8a
86 Ishikawa, Y., SmoothGradCAMplusplus, 2019, o. S.

werden. Hierdurch steht die Schildfläche nicht mehr waagerecht zum Straßenverkehr, dies wird im Folgenden als Rotation entlang der z-Achse bezeichnet.

Um zu überprüfen, welchen Einfluss eine Rotation entlang der x- oder z-Achse auf die Leistungsfähigkeit des CNNs hat, wurden die Bilder des Datensatzes künstlich rotiert, um neue Datensätze zu erzeugen. Für die Rotation entlang der x-Achse wurde, wie bei der Vorbereitung des Trainingsdatensatzes, die *rotate*-Funktion der Bibliothek Pillow verwendet. Um die Bilder entlang der z-Achse zu rotieren, wurde wiederum die Funktion *ImageTransformer.rotate_along_axis* der Bibliothek rotate_3d[87] von Hou-Ning Hu verwendet. Die Bibliothek ist auf GitHub verfügbar.[88]

Insgesamt wurden für jede der Rotationsachsen jeweils 180 neue Datensätze erzeugt. Der Wertebereich der Rotation liegt bei –90 bis 90 Grad. In Abbildung 13 ist jeweils ein Beispiel für die Rotation entlang der x-Achse (b) und z-Achse (c) dargestellt. Beide wurden hierbei um 45 Grad rotiert. Zum Vergleich ist in (a) das Originalbild abgebildet.

(a) Originalbild | (b) Rotation entlang der x-Achse | (c) Rotation entlang der z-Achse

Abb. 13: Beispiele für die Rotation entlang der Achsen

87 Verwendete Commit: 614e0acf9b772007b517caca8921fe6167bab0d0

88 Vgl. Hu, H.-N., Rotate_3d, 2017, o. S.

4.7 Maskieren von Bildabschnitten

Der erste Test, der auf den Ergebnissen der Visualisierungsalgorithmen beruht, ist das Maskieren einzelner Bildbereiche. Zum Maskieren wird ein Bereich hierbei komplett auf Weiß gesetzt. Hierdurch gehen alle Informationen in ihm verloren. Die grundlegende Idee hierhinter ist, dass nach und nach mehr Fläche im Bild maskiert wird. Zwischen den Maskierungsschritten kann die Test Accuracy ermittelt werden, um zu bewerten, welchen Einfluss das Fehlen der maskierten Informationen hat. Um die zu maskierenden Flächen zu bestimmen, wird auf die erzeugten Heatmaps der Visualisierungsalgorithmen zurückgegriffen.

Durch diesen Test sollen zwei Erkenntnisse gewonnen werden. Zunächst kann abgeleitet werden, wie gut die einzelnen Visualisierungsalgorithmen die relevantesten Bereiche innerhalb der Bilder erkennen können. Funktionieren sie wie erwartet, sollte ein ähnlicher Verlauf wie nachfolgend in Abbildung 14 dargestellt erreicht werden. Der Verlauf zeichnet sich durch ein starkes Abfallen der Accuracy zu Beginn ab, welches später abflacht. Die zweite Erkenntnis ist, wie anfällig die Klassifizierung der Straßenschilder für äußere Einflüsse ist. Hier kann beobachtet werden, wie groß die maskierte Fläche sein muss, um relevanten Einfluss auf die Klassifizierung zu haben.

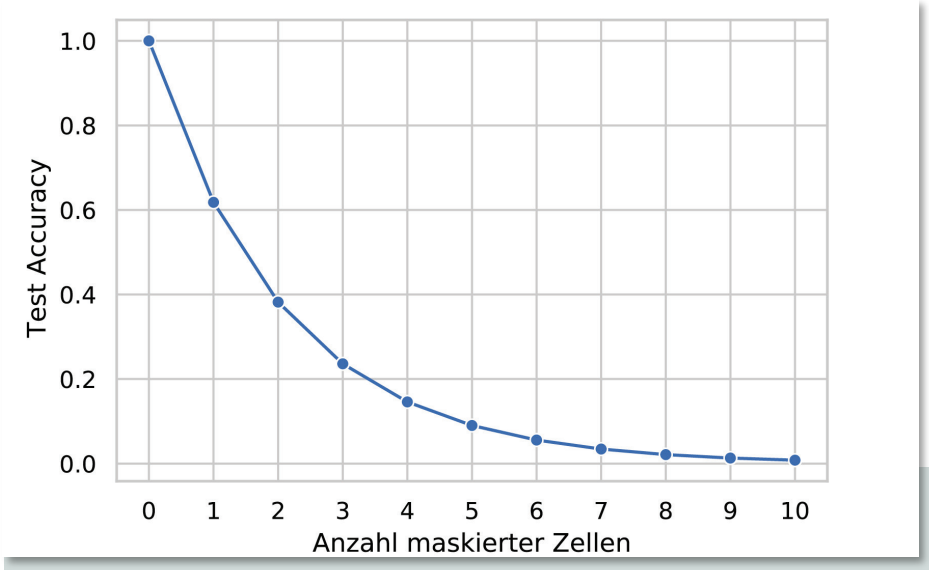


Abb. 14: Beispiel Verlauf der Test Accuracy beim Maskieren

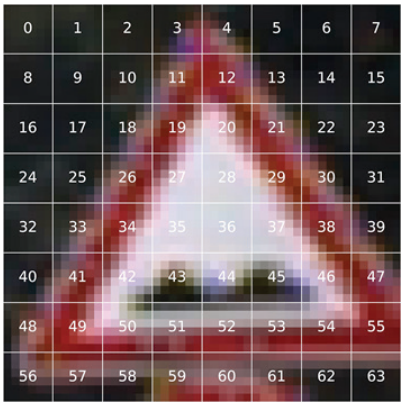
4.7.1 Erzeugen des Rasters

Um ein Bild schrittweise maskieren zu können, muss dieses in Bereiche unterteilt werden. Eine Möglichkeit wäre, die kleinstmöglichen Bereiche, also die Pixel, zu verwenden. Hierfür müssten diese Pixel zunächst anhand ihrer zugeordneten Relevanz – basierend auf den Ergebnissen der Visualisierungsalgorithmen – sortiert werden. Anschließend würde pro Schritt immer eine definierte Anzahl von Pixeln maskiert werden. Es kann allerdings theoretisch vorkommen, dass die in der Reihenfolge aufeinander folgenden Pixel auf dem Bild weit verstreut sind. Da Aufkleber oder andere Verschmutzungen auf einem Schild allerdings zusammenhängende Bereiche bedecken, würde dies nicht den zu testenden Szenarien dieser Studie entsprechen. Daher wurde entschieden, eigene Bereiche zu definieren. Dies wurde durch ein Raster verwirklicht, welches über das Bild gelegt wird und 6x6 Pixel zu einer Zelle zusammenfasst. Bei einer Ausgangsdimension von 48x48 Pixel entstehen somit 64 Zellen. In Abbildung 15 wird ein Beispielbild dargestellt, wobei in (a) das unbearbeitete Bild zu sehen ist und in (b) das generierte Raster über das Bild gelegt wurde.

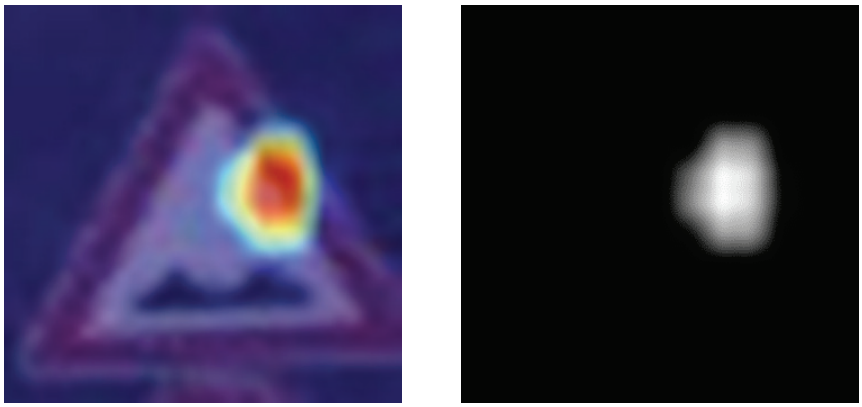
4.7.2 Auswahl der zu maskierenden Zellen

Nachdem das Bild in 64 Zellen unterteilt wurde, muss als nächstes bestimmt werden, als wie relevant die Bildbereiche der einzelnen Zellen auf Grundlage der Visualisierungsalgorithmen bewertet werden.

(a) Original



(b) Raster



(c) RGB Heatmap (d) Grauskala Heatmap

Abb. 15: Darstellung aller zur Maskierung verwendeten Versionen eines Bildes

Die Visualisierungsalgorithmen erzeugen RGB Heatmaps, wobei Rot für die höchste Aktivierung steht (siehe 19 (c)). Für die Auswertung wurden diese Heatmaps in eine Grauskala umgewandelt. Bei dieser stellt der Wert 0 keine Aktivierung und der Wert 1 maximale Aktivierung dar. Wird die grauskalierte Heatmap als Bild ausgegeben, entspricht der Wert 1 der Farbe Weiß (siehe 19 (d)). Grund für die Umwandlung ist, dass die Anzahl von Feature Maps so von 3 auf 1 reduziert wird. Hierdurch kann in den weiteren Schritten Arbeits- und Rechenaufwand gespart werden, da alle Schritte nur ein- statt dreimal ausgeführt werden müssen. Außerdem entfällt der finale Schritt, das Zusammenfassen der Ergebnisse der drei Feature Maps. Im Rahmen der Tests wurde hierbei festgestellt, dass die Methode, mit welcher die auf RGB basierenden Heatmaps in eine Grauskala umgewandelt werden, einen starken Einfluss auf die späteren Ergebnisse hat. So ist der Effekt des Maskierens einzelner Zellen basierend auf dem SM Algorithmus deutlich geringer, wenn dessen Heatmaps – basierend auf der Implementierung in der Bibliothek nn_interpretability – während des Speicherns durch die *imsave*-Funktion der Bibliothek matplotlib (Version 3.1.3) umgewandelt werden.[89] Erst wenn, wie bei Grad-CAM und Grad-CAM++, die *rgb2gray*-Funktion der Bibliothek scikit_image (Version 0.16.2) zur Umwandlung vor dem Abspeichern verwendet wird, zeigt das Maskieren der Zellen das erwarte Verhalten.[90]

[89] Vgl. Hunter, J. et al., Imsave, 2020, o. S.
[90] Vgl. scikit-image development team, Rgb2gray, 2019, o. S.

Mit den vorliegenden grauskalierten Heatmaps wurden zwei Vorgehensweisen zur Bewertung der Relevanz der einzelnen Zellen getestet. In Variante 1 wurde die Anzahl von Pixeln, deren Wert größer als 0,0 (=keine Aktivierung) ist, innerhalb jeder Zelle gezählt. Anschließend wurden die Zellen anhand dieser Summe sortiert, wobei die höchste Anzahl als am relevantesten angesehen wird. Der Nachteil dieser Vorgehensweise ist, dass eine Zelle mit vielen sehr schwach aktivierten Pixeln höher einsortiert wird als eine Zelle mit wenigen stark aktivierten Pixeln. Außerdem ist die Reihenfolge bei Zellen mit der gleichen Anzahl an aktivierten Zellen uneindeutig. Da es pro Zelle nur 36 Pixel gibt, kann dieser Fall häufiger auftreten.

Aufgrund der Probleme von Variante 1 wurde Variante 2 getestet. Bei dieser wird für jede Zelle die Summe aller Pixel mit einer Aktivierung größer 0,0 berechnet. Hierfür wurden die aktivierten Pixel zunächst nach Höhe der Aktivierung absteigend sortiert. Anschließend wurde über diese Liste iteriert und der Aktivierungswert des aktuellen Pixels wurde zu der Summe der Zelle, in der er sich befindet, addiert. Anzumerken ist hierbei, dass für eine kürzere Laufzeit maximal 576 Pixel pro Bild betrachtet wurden. Dies entspricht der – für die avisierte Maskierung von bis zu zehn Zellen im Test – benötigten Anzahl an Pixel plus 60 % Reserve. Die Reserve ist deshalb vorhanden, weil nicht sichergestellt ist, dass die ersten 360 Pixel ausschließlich in zehn Zellen liegen. Abschließend werden die Zellen nach Höhe ihrer Summe absteigend sortiert. Beide Nachteile von Variante 1 werden durch diese Vorgehensweise eliminiert oder reduziert. So wird eine Zelle mit nur einem sehr stark aktivierten Pixel (z. B. 0,99) als relevanter eingestuft als eine Zelle, in der zwar alle Pixel aktiviert sind, diese allerdings nur sehr schwach (z. B. 36*0,01=0,36). Außerdem ist die Wahrscheinlichkeit, dass zwei oder mehr Zellen die gleiche Summe enthalten, deutlich geringer, als dass sie gleich viele aktivierte Zellen enthalten.

In Tests konnten die Maskierungen auf Grundlage von Variante 2 höhere Verluste der Test Accuracy erreichen. Daher wurde sich dazu entschieden, diese im weiteren Vorgehen zu verwenden. In beiden Varianten kann es dazu kommen, dass ein Bild keine zehn verschiedenen Zellen hat, die aktivierte Pixel enthalten. In diesem Fall enthält die Liste der relevanten Zelle für das Bild dementsprechend weniger Elemente. In Abbildung 16 sind beispielhaft Ergebnisse mit einer, fünf und zehn maskierten Zellen dargestellt.

(a) Eine maskierte Zelle (b) Fünf maskierte Zellen (c) Zehn maskierte Zellen

Abb. 16: Beispiele der Ergebnisse des Maskierens

4.8 Erstellen der Stockfotos mit Aufklebern

Nachdem durch das Maskieren einzelner Zellen die generelle Anfälligkeit des CNNs für das Fehlen von Informationen auf den Schildern überprüft wurde, soll mit dem nächsten Versuch ein realistischeres Szenario getestet werden. Hierbei wird auf Aufkleber zurückgegriffen, die in Düsseldorf an Schildern angebracht vorgefunden wurden. Diese wurden fotografiert und ausgemessen. Anhand ihrer Maße und der fest definierten Größen der Verkehrsschilder in Deutschland ist es möglich, die Aufkleber skaliert auf Stockfotos zu platzieren. Die hierdurch entstehenden Datensätze ermöglichen es zu überprüfen, ob die Größe von sich bereits in Verwendung befindenden Aufklebern ausreicht, um die Leistungsfähigkeit des CNNs zu beeinflussen. Hierbei wurde auf Stockfotos zurückgegriffen, um den äußeren Einfluss zu minimieren und so möglichst genau den Einfluss der Aufkleber erkennbar zu machen.

Die zwei ausgemessenen Aufkleber sind in Abbildung 17 nachfolgend abgebildet. Der unter (a) dargestellte Aufkleber hat eine Breite von 185 mm und eine Höhe von 89,25 mm. Im Folgenden wird dieser als Fortuna-Aufkleber bezeichnet. Als zweiter Aufkleber wurde ein etwas kleinerer Aufkleber ausgewählt, dieser ist unter (b) abgebildet und ist 70 mm breit und hoch. Er wird im Folgenden als Bochum-Aufkleber bezeichnet.

(a) Fortuna-Aufkleber, 185mm x 89,25mm (BxH)

(b) Bochum-Aufkleber, 70mm x 70mm (BxH)

Abb. 17: Darstellung der verwendeten Aufkleber

Schildart	Schildmaße (mm)	Umrechnung (px/mm)	Bochum (px)	Fortuna Höhe (px)	Fortuna Breite (px)
Rund (klein)	420	1,905	133	170	352
Rund (mittel)	600	1,333	93	119	247
Rund (groß)	750	1,067	75	95	197
Dreieck (klein)	630	1,270	89	113	235
Dreieck (mittel)	900	0,889	62	79	164
Dreieck (groß)	1260	0,635	44	57	117
Achteck (mittel)	900	0,889	62	79	164
Achteck (groß)	1050	0,762	53	68	141
Quadratisch (klein)	420	1,905	133	170	352
Quadratisch (mittel)	600	1,333	93	119	247
Quadratisch (groß)	840	0,952	67	85	176

Tab. 4: Aufklebermaße für die verschiedenen Schildergrößen

4.8.1 Skalieren der verwendeten Aufkleber

Die in Deutschland verwendeten Straßenverkehrsschilder existieren in verschiedenen Größen. Die verwendete Größe orientiert sich hierbei an der zulässigen Höchstgeschwindigkeit am Standort des jeweiligen Schildes. Für dreieckige, quadratische und rechteckige Schilder wird hierbei zwischen 0 km/h - 50 km/h, 50 km/h - 100 km/h und mehr als 100 km/h unterschieden. Bei runden Schildern liegt die Staffelung bei 0 km/h - 20 km/h, 20 km/h - 80 km/h und mehr als 80 km/h. In Tabelle 22 sind die verschiedenen Größen der vier Schilderformen, welche im Datensatz vorhanden sind, aufgelistet. Anzumerken ist hierbei, dass für das achteckige Stop-Schild nur zwei Größen existieren.[91,92]

Anhand der bekannten Maße der Aufkleber, Schilder und der Stockfotos (600x600 Pixel) ist es nun möglich, die Aufkleber zu skalieren. Hierbei sollen sie so angepasst werden, dass sie dem Orginalaufkleber auf den verschiedenen Schildergrößen entsprechen. Das bedeutet, die Aufkleber müssen auf den Beispielbildern für die drei Schildergrößen von groß nach klein immer weniger Fläche des Schildes verdecken. In Abbildung 18 sind anhand eines Stockfotos die drei verschiedenen Schildergrößen mit Aufkleber dargestellt.

(a) Klein (b) Mittel (c) Groß

Abb. 18: Beispiele für die verschiedenen Schildergrößen mit Aufklebern

91 Vgl. Bundesrepublik Deutschland, Verkehrszeichengröße, 2001, o. S.
92 Vgl. Bundesrepublik Deutschland, VzKat, 2017, o. S.

4.8.2 Positionieren der Aufkleber

Nachdem die Aufkleber nun in den verschiedenen Größen vorliegen, müssen sie noch auf den Stockfotos platziert werden. Für die Platzierung wurde entschieden, erneut auf die Ergebnisse der Visualisierungsalgorithmen zurückzugreifen. Die Aufkleber sollen dabei so platziert werden, dass ihre obere linke Ecke an der oberen linken Ecke der Zelle anliegt, die als die relevanteste Zelle für das Schild bestimmt wurde. Um diese für jedes Schild zu bestimmen, wurden zwei Methoden getestet. Beide Methoden iterieren dabei für ein Schild über alle erzeugten Heatmaps für das Schild und sortieren die Zellen innerhalb dieser anhand desselben Vorgehens, das im vorherigen Abschnitt beschrieben wurde. Bei Methode 1 wurde anschließend ermittelt, welche Zelle am häufigsten die am stärksten aktivierte Zelle auf den Heatmaps war. In Variante 2 erhalten die Zellen Punkte anhand ihrer Platzierung in der Aktivierungsrangfolge. Die Zelle mit dem höchsten Aktivierungswert erhält dabei 10 Punkte, die mit dem zweithöchsten 9 Punkte und so weiter. Die erhaltenen Punkte werden für alle Bilder eines Schildes addiert und abschließend wurde die Zelle mit der höchsten Gesamtpunktzahl ausgewählt. Bei Tests hat sich ergeben, dass die Methoden teilweise unterschiedliche Zellen auswählen. Der Einfluss der Aufkleber auf die Klassifizierung des CNNs ist allerdings bei beiden Methoden identisch. Daher wurde sich für die simplere Methode – Variante 1 – entschieden.

Um die Aufkleber auf den Stockfotos zu platzieren, wurde erneut auf die Bibliothek Pillow zurückgegriffen. Diese bietet mit der *paste*-Funktion die Möglichkeit, ein Bild in ein anderes Bild einzufügen. Hierfür müssen beide Bilder zunächst mit Pillow geöffnet werden und anschließend die *paste*-Funktion auf dem Objekt des Stockfotos aufgerufen werden. Als Parameter werden der Aufkleber und die Zielkoordinaten für die obere linke Ecke übergeben.[93] Die Koordinaten wurden zuvor anhand der Eigenschaft des Rasters (8x8 Zellen, jede Zelle 6 Pixel breit und hoch) berechnet.

[93] Vgl. Clark, A., Pillow, 2020, S. 62.

5 Detaillierte Auswertung und quantitative Diskussion der Ergebnisse

In diesem Kapitel werden die Resultate der Anwendung des optimierten CNNs vorgestellt. Außerdem wird die Resilienz des Klassifikationsprozesses im Hinblick auf verschiedene reale und modellierte Störeinflüsse diskutiert. Hierbei wird zunächst auf die Performance des CNNs im Hinblick auf den GTSRB-Datensatz eingegangen und anschließend der Effekt der einzelnen Modifikationen evaluiert. Danach wird anhand der Stockfotos analysiert, inwieweit das CNN bei nahezu idealen Bedingungen abschneidet. Abschließend werden reale Beispiele mit klar erkennbaren Modifikationen sowie bearbeitete Versionen dieser Schilder – auf denen die Makel behoben wurden – in das CNN gegeben. Hiermit soll aufgezeigt werden, inwieweit das CNN bei unbekannten Schildern trotz Makel noch korrekte Ergebnisse erzielen kann und überprüft werden, ob wirklich diese Makel der Grund für mögliche fehlerhafte Zuordnungen sind.

5.1 Allgemeine Performance des CNNs

Wie zuvor beschrieben, entspricht das verwendete CNN dem Aufbau des CNNs von Arcos-Garcia et al.. Dessen angegebene Genauigkeit liegt bei 99,71 %.[94] Das in dieser Studie verwendete CNN konnte allerdings nur eine Test Accuracy von 97,75 % auf dem Originaldatensatz – ohne Zuschnitt der Bilder – erreichen (97,69 % auf dem zugeschnittenen Datensatz). Die Validation Accuracy erreichte hierbei einen Wert von 99,1 %. Anhand der hohen Accuracy beim Validierungsdatensatz ist ersichtlich, dass das CNN kaum noch weiter lernen wird. Es ist davon auszugehen, dass der Fehler innerhalb des Trainingsdatensatzes mindestens so gering ist wie beim Validierungsdatensatz, wenn nicht sogar noch geringer. Daher hat das CNN keinen großen Anreiz mehr, die Gewichte anzupassen. Dass die Gewichte kaum noch angepasst werden, lässt sich in Abbildung 11 erkennen. In den späteren Epochen verlaufen die Graphen nahezu horizontal. Dies bedeutet, dass die Gewichte nur sehr marginal verändert werden, sodass die Ausgabe des CNNs nahezu identisch bleibt. Es wurden im Rahmen der Studie weitere Architekturen sowie die beschriebenen Variationen der Hyperparameter getestet. Allerdings konnten keine spürbar besseren Resultate erzielt werden. Lediglich eine Implementierung mit einem vorgeschalteten Network Deconvolution[95] Layer konnte marginal besser abschneiden. Aufgrund der deutlich erhöhten Rechenleistung für

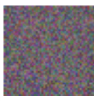
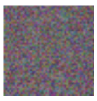
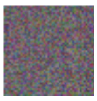
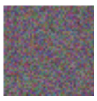
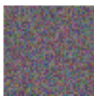
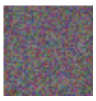
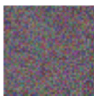
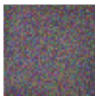
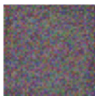
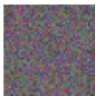
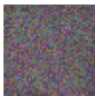
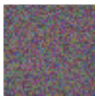
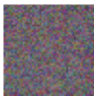
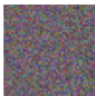
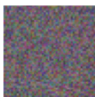
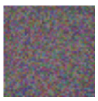
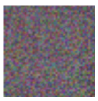
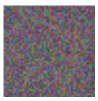
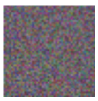
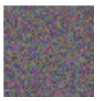
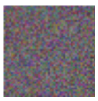
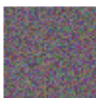
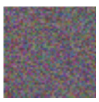
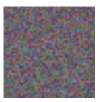
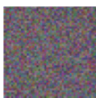
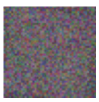
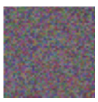
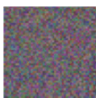
[94] Vgl. Institut für Neuroinformatik der Ruhr-Universität Bochum, Resultate GTSRB, 2019, o. S.
[95] Vgl. Ye, C. et al., Deconvolution, 2020, S. 1-20.

eine Verbesserung im Nachkommabereich wurde sich allerdings gegen die Verwendung dieser Architektur entschieden. Zusammenfassend lässt sich sagen, dass die vorgenommene Implementierung nahezu den gleichen Reifegrad wie die Referenzlösung von Arcos-Garcia et al. bietet und somit eine Basis für die weitergehenden Betrachtungen darstellt, die hier von wesentlichem Interesse sind.

5.2 Activation Maximization

In Abbildung 19 sind die durch den AM Algorithmus erzeugten Bilder abgebildet. Zusätzlich sind sie in größerer Form im Anhang in Abbildung 28 zu finden. Jedes der erzeugten Bilder stellt dabei eine generierte Eingabe dar, welche zu einer maximalen Aktivierung des Zielneurons der zugehörigen Klasse führt. Im Folgenden werden die Aktivierungen als Prozentwerte angegeben. Dies ist möglich, da die Ausgabe des CNNs durch den Softmax Layer auf das Intervall 0 bis 1 normalisiert wurde. Die durchschnittlich erreichte Aktivierung über alle Klassen hinweg liegt bei 99,535 %. Mit 98,890 % hat die Klasse 00022 die niedrigste und mit 99,791 % die Klasse 00026 die höchste erreichte Wahrscheinlichkeit. Es ist anzumerken, dass die dargestellten Werte nicht immer durch das Ausführen des AM Algorithmus zu erreichen sind. Dadurch, dass mit einem Rauschen als Startbild begonnen wird, startet die Optimierung bei jedem Ausführen an unterschiedlichen Startpunkten. Dies und die schrittweise Optimierung, welche nicht das Erreichen des absoluten Minimums der Verlustfunktion garantiert, kann zu schwankenden Resultaten führen. Die verwendeten Werte sind die maximale Aktivierung jeder Klasse, die innerhalb von zehn Durchgängen erreicht wurde.

Auch wenn die Aktivierungswerte sehr hoch sind, lässt sich auf den Bildern keine Ähnlichkeit zu einer der Straßenschilderklassen erkennen. Für das menschliche Auge sieht es weiterhin nach zufälligem Rauschen aus. Dieses Ergebnis entspricht den Erwartungen, weil es sich auch schon in anderen Anwendungsfällen offenbart hat. Dadurch, dass der AM Algorithmus mit einem Rauschen als Ausgangsbild beginnt und anschließend die einzelnen Pixelwerte lokal optimiert, ohne dass mehrere Pixel kombiniert angepasst werden (kontextbasierte Optimierung), ist das Entstehen erkennbarer Formen sehr unwahrscheinlich. Die Ergebnisse des AM Algorithmus zeigen eindrucksvoll, dass das CNN zwar in der Lage ist, die einzelnen Straßenschilderklassen zu klassifizieren, für die eindeutige Zuordnung eines Bildes in eine Klasse allerdings keinerlei Ähnlichkeit im Sinne menschlicher Wahrnehmung zu der Klasse im Bild vorhanden sein muss.

Klasse: 00000	Klasse: 00001	Klasse: 00002	Klasse: 00003	Klasse: 00004	Klasse: 00005	Klasse: 00006
						
Wahrscheinlichkeit: 98.890	Wahrscheinlichkeit: 99.173	Wahrscheinlichkeit: 99.180	Wahrscheinlichkeit: 99.206	Wahrscheinlichkeit: 99.267	Wahrscheinlichkeit: 99.328	Wahrscheinlichkeit: 99.348
Klasse: 00007	Klasse: 00008	Klasse: 00009	Klasse: 00010	Klasse: 00011	Klasse: 00012	Klasse: 00013
						
Wahrscheinlichkeit: 99.384	Wahrscheinlichkeit: 99.393	Wahrscheinlichkeit: 99.432	Wahrscheinlichkeit: 99.435	Wahrscheinlichkeit: 99.457	Wahrscheinlichkeit: 99.500	Wahrscheinlichkeit: 99.504
Klasse: 00014	Klasse: 00015	Klasse: 00016	Klasse: 00017	Klasse: 00018	Klasse: 00019	Klasse: 00020
						
Wahrscheinlichkeit: 99.513	Wahrscheinlichkeit: 99.515	Wahrscheinlichkeit: 99.516	Wahrscheinlichkeit: 99.517	Wahrscheinlichkeit: 99.534	Wahrscheinlichkeit: 99.539	Wahrscheinlichkeit: 99.578
Klasse: 00021	Klasse: 00022	Klasse: 00023	Klasse: 00024	Klasse: 00025	Klasse: 00026	Klasse: 00027
						
Wahrscheinlichkeit: 99.579	Wahrscheinlichkeit: 99.584	Wahrscheinlichkeit: 99.585	Wahrscheinlichkeit: 99.597	Wahrscheinlichkeit: 99.607	Wahrscheinlichkeit: 99.612	Wahrscheinlichkeit: 99.630
Klasse: 00028	Klasse: 00029	Klasse: 00030	Klasse: 00031	Klasse: 00032	Klasse: 00033	Klasse: 00034
						
Wahrscheinlichkeit: 99.634	Wahrscheinlichkeit: 99.654	Wahrscheinlichkeit: 99.657	Wahrscheinlichkeit: 99.661	Wahrscheinlichkeit: 99.669	Wahrscheinlichkeit: 99.673	Wahrscheinlichkeit: 99.680

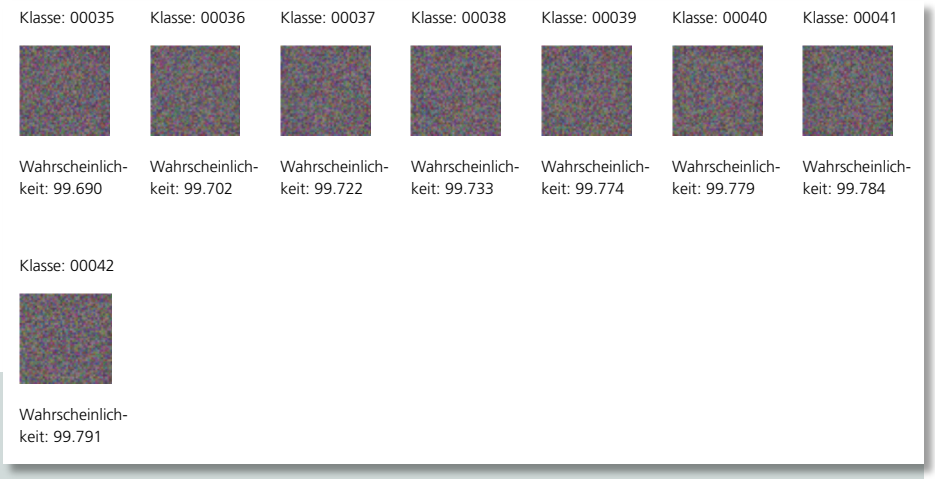
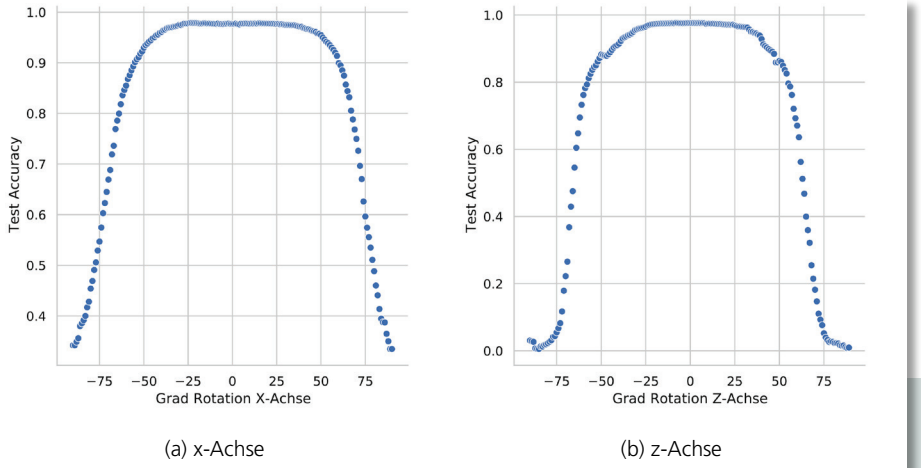


Abb. 19: Activation Maximization Resultate

5.3 Rotation entlang der Achsen

Um zu prüfen, inwieweit das CNN für Rotationen entlang der x- oder z-Achse der Schilder anfällig ist, wurden jeweils 180 neue Datensätze erzeugt. In Abbildung 20 (a) ist der Verlauf der Test Accuracy bei der Rotation entlang der x-Achse dargestellt. Es ist eine Glockenform ohne deutliche Ausreißer erkennbar. Die Symmetrie des Leistungsabfalls in beide Richtungen der Rotation entspricht den Erwartungen. Durch das Vorhandensein der STNs innerhalb des CNNs konnte außerdem davon ausgegangen werden, dass es ein Plateau um 0 Grad Rotation gibt, in dem kaum Accuracy Verlust vorliegt. Dies bestätigt sich. Keine oder nur geringe Rotation entlang der x-Achse haben kaum Einfluss auf die Leistungsfähigkeit des CNNs. Erst bei größerer Rotation steigt die Fehlerquote des CNNs. Die implementierten STNs funktionieren also wie erwartet. Der Verlust der Accuracy überschreitet die 1 %-Marke erst ab einer Rotation um –35 Grad bzw. +40 Grad. Dies stellt den doppelten Rotationswert, mit welcher das CNN trainiert wurde, dar. Eine mehr als 10 % schlechtere Accuracy tritt erst ab –58 Grad bzw. +63 Grad auf. Die niedrigste Accuracy tritt bei ±90 Grad auf und liegt immer noch bei knapp über 33,4 %. Dies zeigt, dass eine starke Rotation durchaus Einfluss auf das Erkennen der Schilder hat, allerdings immer noch ein beachtlicher Anteil der Schilder korrekt zugeordnet wird. Dies erstaunt wenn man bedenkt, dass mehrere

Schilderklassen richtungsweisende Zeichen enthalten, welche durch eine starke Rotation eine andere Klasse darstellen können oder komplett ungültig werden.



(a) x-Achse (b) z-Achse

Abb. 20: Verlauf der Test Accuracy bei Rotation entlang der x- bzw. z-Achse

In Abbildung 20 (b) wird der gleiche Verlauf für die Rotation entlang der z-Achse dargestellt. Auch hier ist eine Glockenform zu erkennen, allerdings ist diese nicht so sauber wie erwartet. Im Bereich von ±50 Grad lassen sich einige Ausreißer erkennen, bei denen trotz stärkerer Rotation eine bessere Accuracy vorliegt als bei den Vorgängern. Auch am unteren linken Fuß der Glocke – also bei den Accuracy-Werten mit der stärksten Rotation im negativen Bereich – sind die Ergebnisse besser als bei geringerer Rotation. Da der Accuracy-Wert hier allerdings nah an Null Prozent liegt, lässt sich dies auf zufällige Treffer zurückführen. Insgesamt ist die Glocke weniger breit als die der Rotation entlang der x-Achse. Dies zeigt, dass der Ansatz zur Unterscheidbarkeit des Einflusses von Rotationen, im Training nur eine der Rotationen mitzutrainieren, erfolgreich war. Allerdings tritt der Verlust von mehr als einem Prozent der Test Accuracy erst ab –25 Grad bzw. +26 Grad auf. Diese Werte sind bereits höher als die Werte, mit welchen die Rotation mit in den Trainingsdatensatz geflossen wäre. Außerdem entfernt die Rotation entlang der z-Achse – im Gegensatz zur Rotation an der x-Achse – tatsächlich Informationen aus dem Bild. Daher ist es wahrscheinlich, dass die Glocke auch mit angepasstem Trainingsdatensatz schmaler wäre. Dies lässt sich auch an der Accuracy, welche bei stärkerer Rotation noch erreicht wird, erkennen. Während

die Rotation entlang der x-Achse selbst bei ±90 Grad noch über 33,4 % lag, fällt sie bei der Rotation entlang der z-Achse bereits bei –69 Grad bzw. +67 Grad unter diesen Wert. Die niedrigste erreichte Accuracy liegt bei lediglich 0,5 %.

Zusammenfassend lässt sich sagen, dass eine Rotation entlang der x- und z-Achse nachweislich Einfluss auf die Accuracy des CNNs haben kann. Allerdings benötigt eine deutliche Verschlechterung dieser bereits so starke Rotationen, dass diese in der Realität nur selten auftreten sollten. Möglicherweise sind dann auch die Straßen selbst derart stark deformiert, so dass sich dieses Problem relativiert.

5.4 Manuelle Auswertung der Heatmaps

Im Rahmen der Studie wurden die Algorithmen SM, Grad-CAM und Grad-CAM++ verwendet um Heatmaps zu erzeugen. Diese Heatmaps sollen die Bereiche auf einem Bild markieren, welche besonders relevant für die Klassifizierung durch das CNN sind.

Zunächst wurden die Algorithmen auf eine Version des CNNs angewendet, welche mit den Originalbildern des GTSRB-Datensatzes trainiert wurde. Dabei wurden die Heatmaps auf Grundlage der Bilder aus dem Test-Datensatz erzeugt, da diese für das CNN unbekannt sind bzw. sie nicht im Rahmen des Trainings verwendet wurden. In der oberen Zeile der Abbildung 21 sind einige Heatmaps des originalen Datensatzes dargestellt. Es ist zu erkennen, dass die als relevant markierten Bereiche (rot) primär neben den Schildern liegen. Eine mögliche Erklärung hierfür ist, dass die Bilder im GTSRB durch eine Videoaufnahme während des Vorbeifahrens an den Schildern entstanden sind.[96] Dies bedeutet, dass der Hintergrund auf Serien von Bildern einer Klasse gleich bleibt. Zusätzlich verändert sich das Schild im Vordergrund durch die Vorbeifahrt stärker als der weiter entfernte Hintergrund. Hierdurch kann es für das CNN effektiver sein, auf den Hintergrund statt auf das Schild selbst zurückzugreifen, um die Klasse zu bestimmen.

[96] Vgl. Stallkamp, J. et al., GTSRB, 2012, S. 323-332.

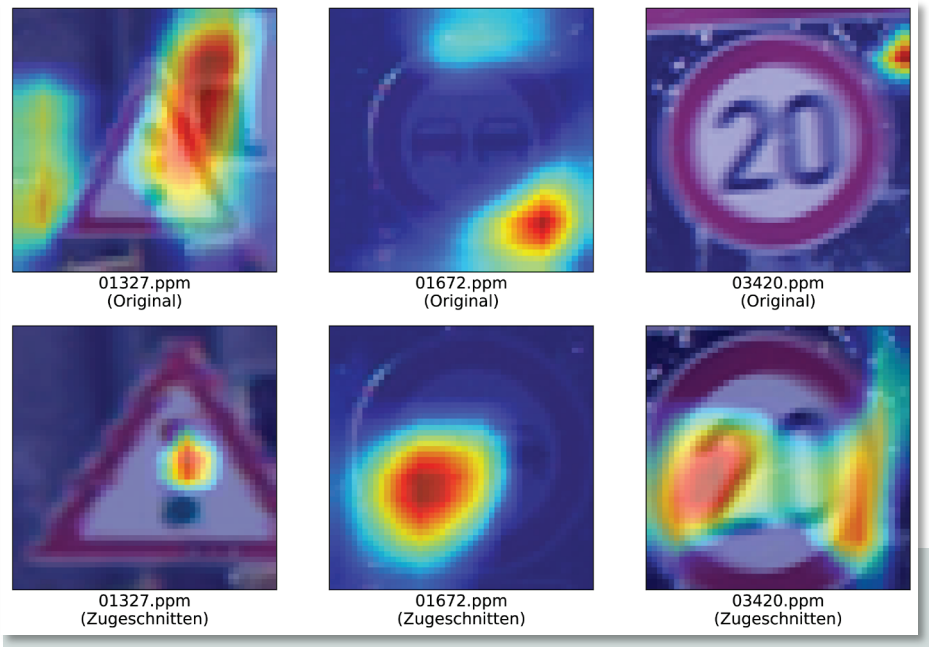


Abb. 21: Gegenüberstellung Heatmaps originaler und zugeschnittener Datensatz

Da dies kein erstrebenswertes Vorgehen des CNNs ist, wurde im Rahmen dieser Studie entschieden, die Bilder im Datensatz zuzuschneiden. Hierdurch wird die Menge der Informationen im Hintergrund begrenzt, sodass das CNN dazu gezwungen ist, sich stärker auf das Schild selbst zu fokussieren. In der unteren Zeile der Abbildung 21 sind die Heatmaps für die zugeschnittenen Bilder der gleichen Beispiele wie in der oberen Zeile dargestellt. Es ist zu erkennen, dass nun Bereiche auf dem Schild als relevant markiert werden.

Es ist anzumerken, dass durch das Zuschneiden der Bilder eine Verbesserung aufgetreten ist. Trotzdem wird bei vielen Bildern der Hintergrund noch mit einbezogen. Dies ist besonders auffällig bei den Schilderklassen mit einer dreieckigen Form. Hier ist allerdings nicht klar, ob dies weiterhin ein ungewolltes Verhalten ist. Auf der einen Seite könnte eine mögliche Erklärung sein, dass die dreieckige Form der Schilder deutlich weniger Fläche des rechteckigen Bildes ausfüllt als die anderen drei Schilderformen. Dadurch stehen dem CNN hier mehr Informationen im Hintergrund zur Verfügung, als es bei den anderen Schilderklassen der Fall ist. Auf der anderen Seite könnte die Fläche neben den Schildern hier auch markiert

sein, weil das CNN durch das Fehlen von Schilderattributen an diesen Stellen erkannt hat, dass ein dreieckiges Schild vorliegen muss. Dies wäre ein akzeptables Vorgehen, um die möglichen Schilderklassen für ein Schild auf dem Bild einzuschränken, da es ja um die äussere Form geht.

Zum einen könnten die Bereiche der Aktivierungen innerhalb der Heatmaps damit verglichen werden, wie ein Mensch die Schilderklassen identifizieren würde. Zum anderen könnte eine Analyse auf numerischer Basis durch das Maskieren ausgewählter Zellen innerhalb der Bilder stattfinden. Dies wird nachfolgend geleistet (siehe Abschnitt 4.7.2 und Abschnitt 5.5).

Vergleicht man die Erwartungshaltungen mit den Analysen, ergibt sich ein heterogenes Bild. Zum einen treten viele Ähnlichkeiten auf. So sind die runde Form sowie der äußere rote Kreis für einen Menschen relevante Merkmale aller Schilderklassen, die eine Geschwindigkeitsbegrenzung vorgeben. Die Heatmaps dieser Klassen zeigen häufig Aktivierungen am Rand der Schilder. Diese Bereiche könnten sowohl zum Erkennen der runden Form als auch des roten Randes verwendet werden. Auf der anderen Seite wurden andere vermutete Merkmale, vom CNN gar nicht verwendet. So wurden die Ziffern der Geschwindigkeitsbegrenzungen häufig an anderen Stellen erkannt, als vermutet. Ein Beispiel hierfür ist die Klasse 00000 (Zulässige Höchstgeschwindigkeit 20). Hier wurde der untere Strich der 2 der Zahl 20 als relevant vermutet, da dieser die 2 von den anderen Ziffern unterscheidet. Die Heatmaps dieser Klasse tendieren allerdings eher zum oberen Bereich der 2.

Zusammenfassend lässt sich sagen, dass das CNN – basierend auf der Ausgabe der Visualisierungsalgorithmen – durchaus auf Merkmale zurückgreift, die denen ähneln, die von Menschen verwendet werden oder welche zumindest für den Menschen nachvollziehbar sind. Letztere sind häufig Merkmale, die erst sinnvoll erscheinen, wenn man die markierten Bereiche mit anderen Klassen vergleicht. Als Beispiel hierfür kann das bereits beschriebene Fehlen jeglicher Schildermerkmale an den Seiten der dreieckigen Schilder gesehen werden. Hierdurch fallen alle Klassen, deren Schilder nicht dreieckig sind, als mögliche Kandidaten raus. Am Ende stellen diese Schlussfolgerungen auf die relevanten Merkmale allerdings nur Vermutungen dar. Zwar zeigen die Visualisierungsalgorithmen den Bereich an, welcher für das CNN für die Klassifizierung relevant war, geben allerdings keine Auskunft darüber, welches Merkmal genau dort verwendet wurde.

5.5 Einfluss der maskierten Zellen

Für die drei Visualisierungsalgorithmen SM, Grad-CAM und Grad-CAM++ wurden jeweils zehn neue Datensätze erstellt. Bei jedem dieser Datensätze ist eine Zelle mehr pro Bild – basierend auf den Heatmaps des jeweiligen Algorithmus – maskiert (siehe Abschnitt 4.7). Nicht jeder Algorithmus konnte hierbei für jedes Bild im Test-Datensatz eine Heatmap mit Aktivierungen erzeugen.

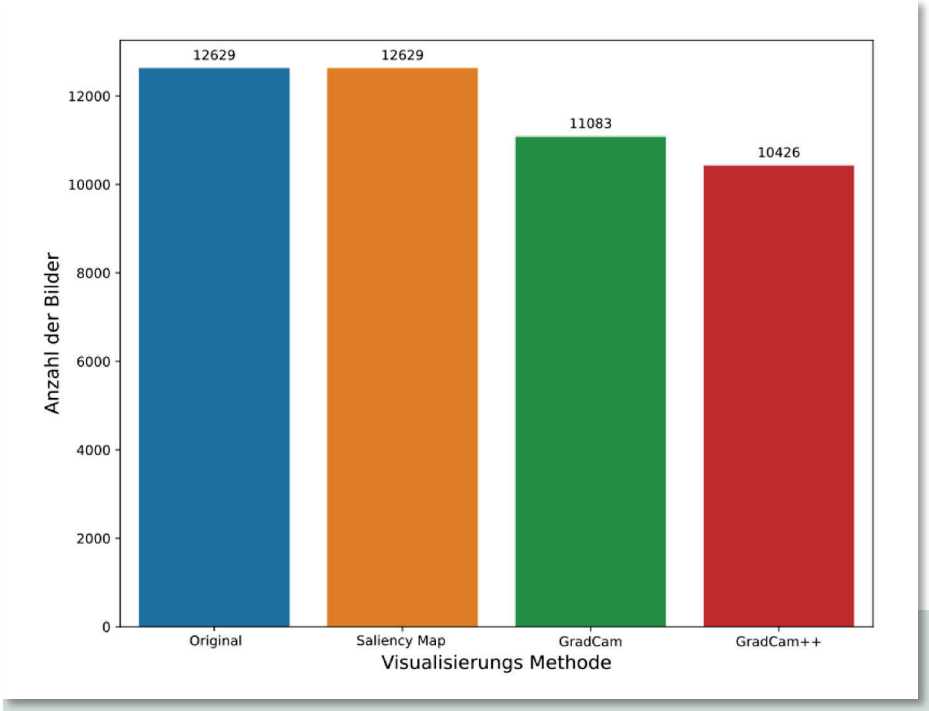


Abb. 22: Anzahl generierter Heatmaps der Visualisierungsalgorithmen

In Abbildung 22 ist eine Übersicht über die Anzahl der erzeugten Heatmaps mit Aktivierung dargestellt. Als Vergleichswert wurde außerdem die Anzahl der Beispiele im Testdatensatz mit aufgenommen. Es ist zu erkennen, dass ausschließlich SM für jedes Bild eine Heatmap mit Aktivierung erzeugen konnte. Grad-CAM konnte mit 11 083 Heatmaps für 87,76 % der Bilder eine Heatmap erstellen und Grad-CAM++ mit 10 426 nur für 82,55 %.

Um die Vergleichbarkeit der verschiedenen Algorithmen zu erhöhen, wurde entschieden, weitere Datensätze zu erzeugen. Diese enthalten die Heatmaps, welche durch SM und Grad-CAM erzeugt wurden. Allerdings nur für die Bilder, die auch im Grad-CAM++ Datensatz auftauchen. Ohne dieses Vorgehen wäre es möglich, dass Grad-CAM++ in den Tests scheinbar besser abschneidet als die anderen Algorithmen, da er mehr nicht eindeutige Bilder ausgelassen hat.

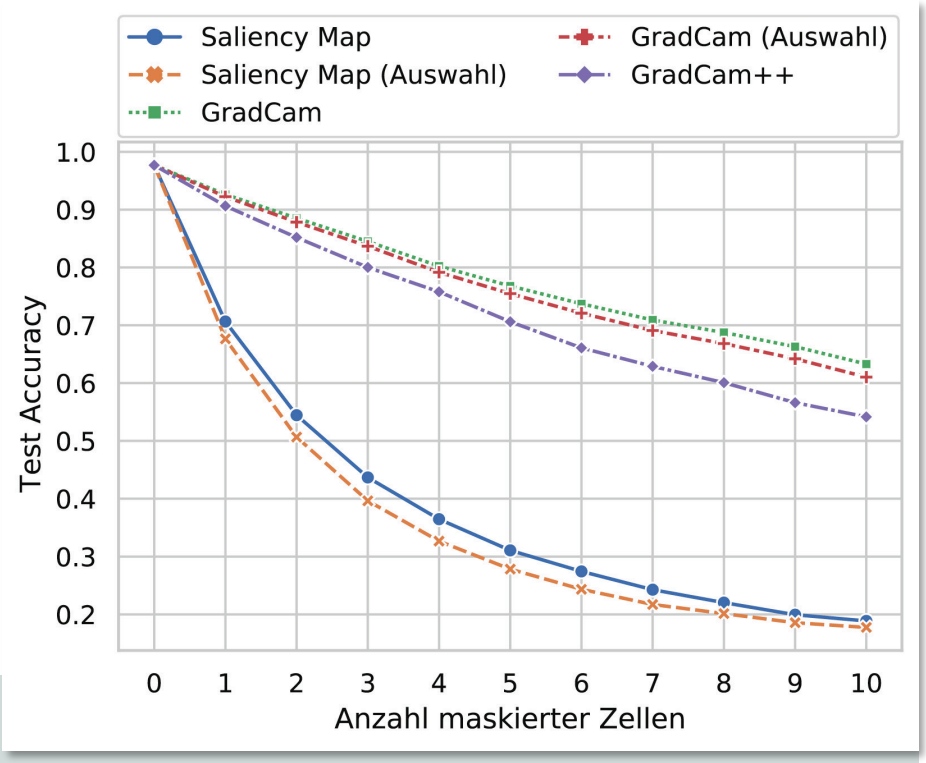


Abb. 23: Accuracy-Verlauf bei steigender Anzahl der maskierten Zellen

In Abbildung 23 ist der Verlauf der Accuracy aller erzeugten Datensätze mit einer steigenden Anzahl maskierter Zellen dargestellt. Zunächst ist ersichtlich, dass alle Datensätze ohne maskierte Zellen die gleiche Ausgangs-Accuracy erreichen (97,69 %). Außerdem ist erkennbar, dass die Datensätze, die lediglich die oben erläuterte Auswahl an Bildern enthalten (markiert durch Auswahl), besser abschneiden als ihre vollständigen Versionen. Grad-CAM++ scheitert also tatsäch-

lich daran, für Bilder, in denen die relevanten Bereiche weniger deutlich sind, Heatmaps zu erstellen. Es ist deutlich erkennbar, dass die maskierten Zellen basierend auf Grad-CAM und Grad-CAM++ deutlich weniger Einfluss auf die Accuracy nehmen, als jene, die auf SM basieren. Hierbei erreicht SM bereits mit zwei maskierten Zellen eine niedrigere Accuracy als jeder der CAM-Algorithmen mit zehn maskierten Zellen. Es ist fraglich, warum die auf CAM basierenden Algorithmen lediglich eine Reduzierung der Accuracy auf minimal 54,16 % erreichen können. Eine mögliche Erklärung ist, dass diese Algorithmen meistens nur einzelne Bereiche des Bildes mit Aktivierungen markiert haben, während SM nahezu das gesamte Bild mit einer Aktivierung von größer als 0 versehen hat. Auch wenn Letzteres für den Großteil eines Bildes nur minimal die Null überschreitet, liegen hier trotzdem mehr Informationen vor. Diesem Ansatz widerspricht allerdings, dass auch für nahezu alle Heatmaps von Grad-CAM und Grad-CAM++ zehn Zellen maskiert werden konnten. Hätte es nur Aktivierungen in z. B. fünf Zellen gegeben, wären auch nur fünf Zellen maskiert worden. Das heißt, auch diese Algorithmen haben meistens genug Bereiche für das Maskieren von zehn Zellen markiert.

Das Maskieren einer Zelle – basierend auf SM – führt bereits zu einer Reduzierung der Accuracy um 27,04 % bzw. 30,04 % (Auswahl). Dieser signifikante Verlust in der Accuracy wird dabei bereits dadurch erreicht, dass mit 6x6 Pixel einer Zelle gerade mal 1,56 % der 48x48 Pixel des gesamten Bildes maskiert werden. Durch das Maskieren mit zehn Zellen (15,62 % der Bildfläche) sinkt die Accuracy auf 18,86 % bzw. 17,74 % (Auswahl).

Auch wenn dies einen deutlich geringeren Wert als die Ausgangs-Accuracy darstellt, ist der Wert immer noch fast zehnmal so hoch wie die Wahrscheinlichkeit, durch zufälliges Raten bei 43 Klassen richtig zu liegen. Dies bedeutet, dass, selbst wenn dem CNN die wichtigsten 15 % Bildfläche für die Klassifizierung nicht zur Verfügung stehen, dieses immer noch Ansatzpunkte für diese hat. Pro Klasse werden also nicht nur einzelne signifikante Bereiche gelernt, sondern eine breitere Auswahl dieser.

Es ist außerdem noch anzumerken, dass das Maskieren der Zellen durch eine Anpassung der enthaltenen Pixel auf den identischen RGB-Wert für Weiß ((255, 255, 255) durchgeführt wurde. Da einige Schilder weiße Zeichen enthalten und das CNN gegebenenfalls auf weiße Flächen an bestimmten Bereichen des Bildes prüft, könnte es zu zusätzlicher Störung des CNNs gekommen sein. Nicht nur, dass die wahren Informationen des Bildes innerhalb der Zelle nicht zur Verfügung stehen,

sondern auch, dass stattdessen manipulative Informationen diese ersetzen. Es wurde überlegt, dies durch eine andere Farbe beim Maskieren zu verhindern. Allerdings können reale Beschädigungen wie Aufkleber diesen Effekt ebenfalls zufällig oder sogar durch bewusste Manipulation erzielen.[97] Daher wurde sich gegen eine andere Farbe entschieden.

Alle Verläufe entsprechen dem erhofften Verlauf. Auch wenn die auf dem CAM-Algorithmus basierenden Verläufe nicht so steil sind wie die Verläufe basierend auf SM, erreichen sie trotzdem noch signifikante Verschlechterungen der Accuracy. Auch bei ihnen führen die ersten maskierten Zellen zu deutlich größeren Verlusten in der Accuracy als die späteren (z. B. 7,05 % zu 2,46 % bei Grad-CAM++). Dies bedeutet, dass beide erwarteten Ergebnisse durch das Maskieren der Zellen – Visualisierungsalgorithmen können die wichtigeren Bereiche zur Klassifizierung korrekt erkennen und das CNN ist anfällig für externe Manipulation – bestätigt wurden (siehe Abschnitt 4.7)

5.6 Effekt realer Aufkleber auf Stockfotos

Das Anbringen von realen Aufklebern auf Stockfotos soll zeigen, wie anfällig das CNN für bereits auf Straßenschildern angebrachte Aufkleber ist. Diese können – im Verhältnis zum Schild – größer oder kleiner als eine maskierte Zelle sein. Allerdings bringen sie durch ihr eigenes Design mehr Einflüsse mit als ein komplettes Maskieren. Es wurde auf Stockfotos zurückgegriffen, um die Größe anderer Einflüsse als der der Aufkleber klein zu halten. Da die einzelnen Eigenschaften aller Schilder auf den Stockfotos perfekt zu erkennen sind, sollte das CNN ohne Aufkleber keine Probleme mit der Klassifizierung haben. Außerdem ist auf den Stockfotos keinerlei Hintergrund vorhanden. Sollte es also doch zu Fehlklassifizierungen durch das CNN kommen, wäre eine mögliche Erklärung, dass für das CNN weiterhin relevante Informationen im Hintergrund existieren.

Wie in Abschnitt 4.8 erläutert, gibt es verschiedene Schildergrößen. Die ausgemessenen Aufkleber wurden dementsprechend skaliert, sodass ihr Einfluss bei jeder der Schildergrößen getestet werden konnte. In Abbildung 24 sind die Resultate dieser Tests dargestellt. Wie erwartet liegt die Accuracy ohne Aufkleber (None) bei 100 %. Ebenfalls wie erwartet erreicht der Fortuna-Aufkleber eine größere Reduktion der Accuracy. Da dieser größer als der Bochum-Aufkleber ist,

[97] Vgl. Gu, T., Dolan-Gavitt, B., Garg, S., BadNets, 2019, S. 1-13.

verdeckt er auch mehr Informationen des Schildes. Überraschend ist, dass die Accuracy beim Fortuna-Aufkleber beim Wechsel von der mittleren Schildergröße auf die kleinste Schildergröße wieder steigt. Da der Aufkleber hier im Verhältnis zum Schild am größten ist, verdeckt er bei der kleinsten Schildergröße mehr Informationen als bei der mittleren Schildergröße. Dies sollte eigentlich dazu führen, dass die Accuracy schlechter wird, zumindest aber gleich bleibt.

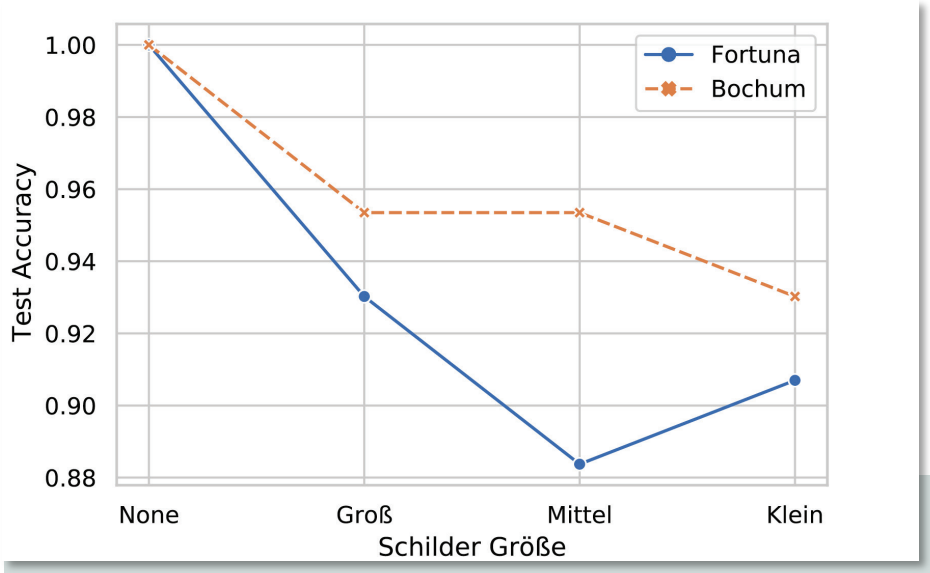


Abb. 24: Accuracy auf den Stockfotos mit unterschiedlicher Schildergröße

Betrachtet man die Prozentzahlen, ist erkennbar, dass bei dem Fortuna-Aufkleber bei der größten Schildergröße drei Schilderklassen falsch klassifiziert wurden (93,02 %). Bei der mittleren Schildergröße steigt dieser Wert auf fünf (88,37 %), um dann bei der kleinsten Schildergröße wieder auf vier (90,7 %) zu sinken. Es wurde also trotz des verhältnismäßig größeren Aufklebers eine Schilderklasse mehr korrekt klassifiziert. Bei dieser Schilderklasse handelt es sich um Klasse 00033 (Gefahrenstelle Fahrtrichtung rechts). In Abbildung 25 ist diese Klasse in den drei Schildergrößen mit dem skalierten Fortuna-Aufkleber dargestellt. Nur die mittlere Version wurde fehlerhaft klassifiziert, sie wurde fälschlicherweise der Klasse 00039 (Vorgeschriebene Vorbeifahrt links vorbei) zugeordnet. Die korrekte Klasse 00033 liegt hierbei bei der zugeordneten Wahrscheinlichkeit mit 37,4 % auf Platz 2 (00039 hat eine Wahrscheinlichkeit von 62,2 %). Bei der kleinsten

Schildergröße ändert sich dies zugunsten der korrekten Klasse (74,8 % zu 24,8 %), womit es hier sogar eindeutiger ist als bei der größten Schildergröße. Bei dieser liegt zwar auch eine korrekte Klassifizierung vor, allerdings nur mit 58,6 % zu 41,1 %. Eine mögliche Erklärung könnte sein, dass der Aufkleber auf dem größten Schild noch nicht genug verdeckt, um die Klassifizierung zu Klasse 00039 zu kippen. Währenddessen könnte sich die weiße Schrift innerhalb des Aufklebers bei der kleinsten Schildergröße zufällig dort befinden, wo das CNN Teile des weißen Pfeils der Schilderklasse 00033 erwartet.

Abb. 25: Darstellung der Schilderklasse 00033 mit dem skalierten Fortuna-Aufkleber

Führt man die gleiche Betrachtung wie für den Fortuna-Aufkleber für den Bochum-Aufkleber durch, erkennt man, dass bei der größten und mittleren Schildergröße zwei (95,34 %) und bei der kleinsten Schildergröße drei (93,02 %) Schilderklassen falsch klassifiziert wurden. Dieser Verlauf entspricht eher dem Erwarteten, dass die Aufkleber größeren Einfluss haben, wenn die Schildergrößen geringer werden. Der Bochum-Aufkleber bedeckt bei den Schildergrößen in den verschiedenen Formen der Schilder von groß nach klein zwischen 0,42 % und 0,81 %, 0,57 % und 1,36 %, sowie 0,57 % und 2,76 %. Er liegt hiermit teilweise deutlich unter den 1,5 %, die durch das Maskieren einer einzelnen Zelle im vorherigen Versuch bedeckt wurden. Dies könnte erklären, warum sein Einfluss nicht so stark ist.

In Tabelle 5 sind die durchschnittlichen Wahrscheinlichkeitswerte für die durch das CNN bestimmten Schilderklassen dargestellt. Hierbei wird sowohl zwischen korrekt und fehlerhaft klassifiziert als auch zwischen Bochum-Aufkleber und Fortuna-Aufkleber unterschieden. Es ist zu erkennen, dass die durchschnittliche Wahrscheinlichkeit bei den korrekt klassifizierten Klassen sehr hoch liegt (mindes-

tens 98,12 %). Bei den fehlerhaft klassifizierten Klassen liegt diese deutlich niedriger (maximal 77,23 %). Mit einem Schwellenwert könnte man also tatsächlich – zumindest auf Stockfotos – ableiten, ob das CNN durch äußere Einflüsse verunsichert wurde oder nicht.

Schildergröße / Klassifizierung	Groß	Mittel	Klein	Groß	Mittel	Klein
	Bochum			Fortuna		
Korrekt	0,9887	0,9910	0,9793	0,9812	0,9952	0,9365
Falsch	0,5275	0,6757	0,6670	0,7718	0,7723	0,6952

Tab. 5: Durchschnittliche Wahrscheinlichkeiten Klassifizierung der Stockfotos

5.7 Reale Beispiele

Für diesen Test wurden vier Straßenschilder in Düsseldorf fotografiert. Zusätzlich wurde ein Straßenschild aus einem Foto eines Artikels der Leipziger Volkszeitung verwendet.[98] Diese Schilder weisen eine oder mehrere Beschädigungen auf, sodass sie nicht mehr vollständig dem Sollzustand ihrer Klassen entsprechen. Diese Beschädigungen sind das Ausbleichen der Farben auf einem Schild, die Verformung eines Schildes sowie ein oder mehrere Aufkleber. In der oberen Zeile der Abbildung 26 sind die beschädigten Schilder abgebildet. Die Mängel an den Schildern wurden zusätzlich mittels Bildbearbeitung entfernt. Die korrigierten Versionen werden in der unteren Zeile der Abbildung 26 dargestellt.

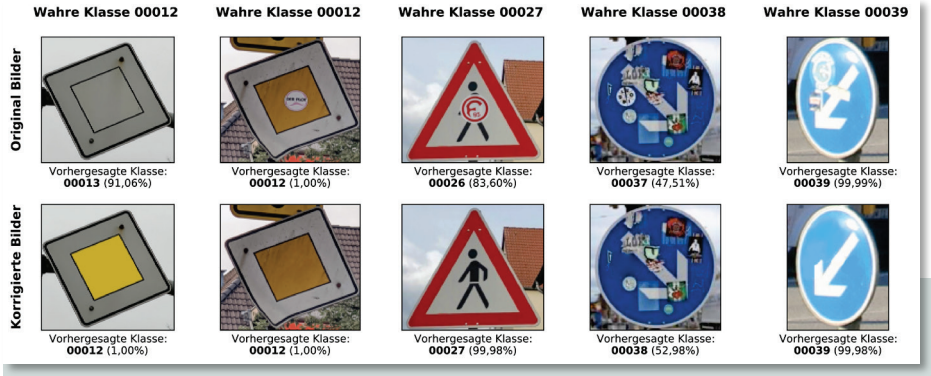


Abb. 26: Gegenüberstellung der originalen und korrigierten Straßenschilder

98 Vgl. Heinze, J., Vorbeifahrt rechts, 2019, o. S.

Jede Spalte der Abbildung enthält dabei die zusammengehörigen Bilder eines Schildes. Unter jedem Bild ist die vom CNN zugeordnete Klasse mit der jeweiligen Wahrscheinlichkeit angegeben. Wie in Abbildung 26 zu erkennen, wurden von den beschädigten Schildern zwei Schilder korrekt und drei Schilder fehlerhaft klassifiziert.

Das Schild in der ersten Spalte der Abbildung stellt ein ausgeblichenes Schild der Klasse 00012 (Vorfahrtsstraße) dar. Statt der korrekten Klasse 00012 wurde die Klasse 00013 (Vorfahrt gewähren) mit 91,06 % Wahrscheinlichkeit zugeordnet. Diese entspricht weder der eigentlichen Form des Schildes – dreieckig gegenüber viereckig –, noch passt der äußere rote Rand. Allerdings ist die Klasse 00013 eine von nur zwei Klassen innerhalb des Datensatzes, welche eine durchgehende weiße Fläche mittig auf dem Schild hat. Die andere Klasse ist 00015 (Verbot für Fahrzeuge aller Art). Klasse 00015 ist bei den zugeordneten Wahrscheinlichkeiten auf Platz 3 (1,45 %), knapp hinter der tatsächlichen Klasse 00012 (4,27 %). Alle anderen Klassen haben eine Wahrscheinlichkeit kleiner als 1 %. Es scheint also, als ob das Ersetzen der gelben Fläche in der Mitte des Schildes durch eine weiße Fläche genügt, um das CNN zu einer fehlerhaften Klassifizierung zu bringen. Um dies zu überprüfen, wurde die mittige Fläche in der korrigierten Version gelb gefärbt. Dies führt tatsächlich zur korrekten Klassifizierung mit einer Wahrscheinlichkeit von 100 %.

In der zweiten Spalte ist dieselbe Schilderklasse mit einer Verformung im unteren Bereich des Schildes und zusätzlich einem Aufkleber in der Mitte des Schildes abgebildet. In diesem Fall wurde bereits die beschädigte Version vom CNN korrekt klassifiziert. Das CNN gibt hierbei eine Wahrscheinlichkeit von 100 % an. Bedenkt man die Erkenntnisse aus der ersten Spalte mit, scheint es so, als ob der Aufkleber nicht genügend gelbe Fläche abdecken würde bzw. er nicht an den relevanten Stellen ist. Betrachtet man die erzeugten Heatmaps oder die Ergebnisse durch das Maskieren, scheint der Fokus des CNNs eher auf den Rändern der gelben Fläche bzw. der Ecken dieser zu liegen. Da der Aufkleber annähernd zentral angebracht ist, überdeckt er diese Bereiche nicht. Wie erwartet verändert das Entfernen des Aufklebers die Klassifizierung und Wahrscheinlichkeit nicht. Der tatsächlich verformte Teil des Schildes – die weiße untere Spitze – wird in nahezu keiner der Heatmaps als besonders relevant markiert. Daher ist es nicht verwunderlich, dass die Verformung an dieser Stelle keinerlei Einfluss auf das CNN hat. Auf das Beheben dieses Makels wurde aufgrund der bereits erreichten 100 % Wahrscheinlichkeit verzichtet.

Die dritte Spalte enthält ein Schild der Klasse 00027 (Gefahrenstelle Fußgänger) mit einem Aufkleber mittig auf dem Schild. Das CNN hat das beschädigte Schild fehlerhafterweise der Klasse 00026 (Gefahrenstelle Lichtzeichenanlage) mit einer Wahrscheinlichkeit von 83,60 % zugeordnet. In diesem Fall wurde fünf Klassen eine Wahrscheinlichkeit von über einem Prozent zugeordnet. Die korrekte Klasse 00027 liegt mit 1,55 % nur auf Platz 3. Deutlich höher wurde mit 11,59 % die Klasse 00024 (Gefahrenstelle: Einseitig verengte Fahrbahn (rechts)) eingeordnet. In diesem Fall ist nicht klar ersichtlich, warum die beiden falschen Klassen der korrekten Klasse vorgezogen wurden. Hier passen zwar die grundlegende Form und Farbgebung, allerdings trifft dies auch auf viele andere Klassen zu, von denen nur zwei noch über ein Prozent Wahrscheinlichkeit zugeordnet bekommen haben. Eine Möglichkeit wäre, dass diese beiden Schilder weiße Flächen innerhalb ihrer Symbole haben. Die weißen Flächen des Aufklebers könnten hierbei zu einer Erkennung dieser geführt haben. Für die Klasse 00026 spricht zudem, dass der rote Rand des Aufklebers als Teil des roten Kreises der Ampel auf dieser Schilderklasse interpretiert worden sein könnte. Hinzu kommt, dass der Aufkleber einen großen Teil der Figur auf dem Schild überdeckt. So wurde in der Analyse der Heatmaps für die Klasse 00027 der rechte Arm der Figur als relevant ermittelt. Dieser ist komplett vom Aufkleber überdeckt, sodass er als Identifizierungspunkt wegfällt. Nachdem der Aufkleber entfernt und die fehlenden Teile des Männchens nachgemalt wurden, klassifiziert das CNN das Bild mit einer Wahrscheinlichkeit von 99,98 % korrekt.

Als letztes fehlerhaft klassifiziertes Schild ist in Spalte vier ein Schild der Klasse 00038 (Vorgeschriebene Vorbeifahrt rechts vorbei) dargestellt. Dieses ist mit einer Vielzahl von Aufklebern beklebt und wird mit einer niedrigen Wahrscheinlichkeit von 47,51 % der falschen Klasse 00037 (Vorgeschriebene Fahrtrichtung links/geradeaus) zugeordnet. Erneut liegt die korrekte Klasse nur auf Platz drei, mit einer Wahrscheinlichkeit von 9,68 %. Auf Platz zwei liegt die Klasse 00039 (Vorgeschriebene Vorbeifahrt links vorbei) mit 34,20 %. Alle anderen Klassen haben eine Wahrscheinlichkeit von kleiner als ein Prozent. Betrachtet man die für das CNN in Frage kommenden Schilder, erkennt man, dass alle drei sehr ähnlich sind. Bei allen drei liegt ein rundes blaues Schild vor, auf dem sich ein weißes Zeichen befindet. Die zwei falschen Klassen 00037 und 00039 unterscheiden sich primär dadurch von Klasse 00038, dass sie im linken unteren Viertel des Schildes noch weiße Elemente haben. Die Klasse 00037 hat neben ihrem senkrechten Pfeil noch den Pfeil nach links. Die Klasse 00039 besteht aus einem nach links unten zeigenden Pfeil, dessen Pfeilspitze im relevanten Bereich ist. Durch einen der Aufkleber

lässt sich eine weiße Fläche links neben dem Pfeil des Schildes finden. Entfernt man diesen einen Aufkleber (siehe untere Zeile), fällt Klasse 00037 auf 44,26 % und die Klasse 00039 auf 2,64 %. Zeitgleich steigt die korrekte Klasse 00038 auf 52,98 %. Das CNN klassifiziert das Schild also korrekt. Tatsächlich hat der weiße Aufkleber also entscheidend auf die Klassifizierung des CNNs eingewirkt. Zwar ist die Wahrscheinlichkeit mit 52,98 % für die korrekte Klasse nicht sehr hoch, allerdings sind auch noch viele störende Aufkleber vorhanden. Entfernt man diese auch, erhält man eine Wahrscheinlichkeit von 99,99 %.

In der letzten Spalte der Abbildung 26 ist ein Bild der Klasse 00039 (Vorgeschriebene Vorbeifahrt links vorbei) dargestellt. Dieses wird - trotz der Präsenz - dreier Aufkleber bereits mit 99,99 % Wahrscheinlichkeit korrekt klassifiziert. Betrachtet man alle Schilderklassen, gibt es nicht wirklich eine Klasse, die an den Stellen der Aufkleber Elemente hat und ansonsten ähnlich zu Klasse 00039 ist. Maximal Klasse 00034 (Vorgeschriebene Fahrtrichtung links) hat in dem Bereich des Schildes ein Element und ist ein blaues, rundes Schild. Allerdings ist das Element der Klasse 00034 ein weißer Pfeil, während der Aufkleber grün ist. Tatsächlich senkt das Entfernen der Aufkleber die Wahrscheinlichkeit minimal auf 99,98 %; warum dies der Fall ist, ist nicht ersichtlich. Eine mögliche Begründung, warum in beiden Versionen des Schildes nicht 100 % Wahrscheinlichkeit erreicht werden, ist, dass das Schild seitlich fotografiert wurde. Das heißt, es liegt eine Rotation entlang der z-Achse vor. Wie in Abschnitt 5.3 beleuchtet, führt dies zu einer Verschlechterung der Leistungsfähigkeit des CNNs. Diese Verschlechterung ist in diesem Fall zwar nur minimal, dies lässt sich aber dadurch begründen, dass das Schild immer noch sehr gut erkennbar ist.

Zusammenfassend lässt sich sagen, dass das CNN trotz Beschädigungen an den einzelnen Schildern noch gut bei der Klassifizierung abschneidet. Zwei der Beispiele wurden trotz Beschädigungen mit einer sehr hohen Wahrscheinlichkeit korrekt zugeordnet. Bei einem anderen Beispiel hat das Entfernen von nur einem von vielen Aufklebern genügt, um zu einer korrekten Klassifizierung zu kommen. Die beiden Beispiele der Klasse 00012 zeigen zudem, dass das CNN zwar für großflächige Änderungen anfällig sein kann, kleine bis mittelgroße Beschädigungen allerdings nicht unbedingt Einfluss haben. Neben der Leistungsfähigkeit des CNNs haben sich einige Erkenntnisse über den Lernprozess des CNNs, welche durch die Analysen und Evaluierungen in den vorherigen Abschnitten gewonnen wurden, bestätigt. So konnten zum Beispiel relevante und nicht relevante Bereiche für die Klassifizierung erneut bestätigt werden.

6 Fazit

Zum Abschluss der Studie fasst dieses Kapitel alle Ergebnisse zusammen. Basierend darauf wird evaluiert, ob die in Abschnitt 1.2 aufgestellten Hypothesen verworfen werden können. Außerdem wird die im gleichen Abschnitt gestellte Forschungsfrage beantwortet. Nach einem kritischen Blick auf die Vorgehensweise und Methodik werden Konsequenzen, die aus den Ergebnissen gezogen werden können, aufgezeigt. Zum Abschluss wird ein Ausblick auf weitere Themen gegeben, die für den Forschungsgegenstand der Studie relevant sind.

5.8 Ergebnisse der Studie

Im Rahmen der Studie konnte ein CNN erstellt werden, welches im Bezug auf die Maßzahl Accuracy ein ähnliches Niveau wie die besten veröffentlichten Modelle für den GTSRB- Datensatz erreicht. Dadurch stellt dieses CNN eine gute Grundlage für die durchgeführten Tests dar. Auch die ausgewählten Visualisierungsalgorithmen konnten erfolgreich angewendet werden. Mittels AM konnte gezeigt werden, dass die quantitativ maßgeblichen Merkmale in den klassifizierten Bildern nicht der intuitiven Erwartungshaltung geometrischer Primitive der entsprechenden Schilderklassen entsprachen.

Im Kontext XAI ist hier ein Anknüpfungspunkt zu grundlegenden Diskrepanzen in humaner und maschineller Kognition gegeben. Alle drei implementierten Saliency Methoden sind in der Lage, für die Klassifizierung relevante Bereiche innerhalb eines Bildes zu markieren. Die von den Visualisierungsalgorithmen erzeugte Gewichtung der markierten Bereiche konnte plausibilisiert werden. Das Maskieren von als besonders relevant erkannten Bereichen hat hierbei einen stärkeren Einfluss auf die Accuracy als das Maskieren von als weniger relevant erkannten Bereichen.

Die besten Ergebnisse konnten mit dem SM-Algorithmus erzielt werden. Das auf ihm basierende Maskieren konnte nicht nur konsistent bei gleicher maskierter Fläche tiefere Accuracy-Werte erzielen, sondern auch eine deutlich geringere minimale Accuracy als die anderen beiden Algorithmen erreichen. Basierend auf den durch SM erzeugten Heatmaps konnte neben der numerischen Analyse des Maskierens auch eine subjektive Analyse der relevanten Bereiche durchgeführt werden. Die menschliche Betrachtung spielt nach wie vor die Rolle einer wichtigen Referenz in Klassifikationsaufgaben und assoziierten Problemen und ergänzt die maschinelle Herangehensweise.

Die dadurch ermittelten Erkenntnisse konnten vom Testdatensatz, auf Basis dessen sie erzeugt wurden, erfolgreich auf weitere Datensätze übertragen werden. So konnte anhand von Stockfotos und real existierender und skalierter Aufkleber gezeigt werden, dass die Platzierung der Aufkleber – basierend auf der numerischen Analyse – auf ansonsten perfekten Ausgangsbildern bei mehreren Klassen zu Fehlklassifizierung führt. Auch die auf der subjektiven Analyse basierende Auswahl von Schildern aus Düsseldorf konnte aufzeigen, dass Störfaktoren an als relevant erkannten Bereichen starken bzw. an als nicht relevant erkannten Bereichen keinen Einfluss auf die Klassifizierung haben. Neben dem Einfluss von Störfaktoren auf den Schildern konnte ebenfalls der Einfluss von Rotationen entlang der x- bzw. z-Achse der Schilder ermittelt werden. Das CNN ist hierbei in der Lage, bei beiden Achsen eine Rotation von mindestens 25 % in beide Richtungen ohne nennenswerte Verluste der Accuracy zu verarbeiten.

5.9 Evaluierung der Hypothesen und der Forschungsfrage

In Abschnitt 1.2 wurden zwei Hypothesen aufgestellt. Hypothese 1 kann eindeutig verworfen werden. Anhand der Visualisierungsalgorithmen konnten die relevanten Pixel für die Klassifizierung ermittelt werden. Sowohl der numerische Ansatz, welcher in der Lage ist, für jedes Bild einzeln die relevanten Bereiche zu bestimmen, als auch der auf die Schilderklassen abstrahierende subjektive Ansatz sind in der Lage, die Pixelbereiche hierbei auf eine höhere Abstraktionsebene (Zellen bzw. Attribute) zu heben.

Auch Hypothese 2 kann grundsätzlich verworfen werden. So können anhand der ermittelten Merkmale nicht nur Schilder mit Störfaktoren bestimmt werden, die fehlerhaft klassifiziert werden. Es ist auch möglich, Schilder zu bestimmen, welche Störfaktoren enthalten, aber trotzdem korrekt klassifiziert werden. Allerdings haben die Versuche auch gezeigt, dass das CNN sich bei keiner Klasse ausschließlich auf kleine Bereiche fokussiert. So müssen mittel- bis großflächige oder mehrere Störfaktoren vorliegen, um die Klassifizierung des CNN zu stören.

Zusammenfassend können die Resultate der Arbeit die Forschungsfrage beantworten. Relevante Merkmale konnten mittels Visualisierungsalgorithmen, basierend auf numerischer und subjektiver Analyse, herausgearbeitet werden. Auch die Anfälligkeit der Klassifizierung für äußere Einflüsse konnte anhand des Verlustes der Accuracy nachgewiesen werden. Hierbei können sowohl das Maskieren einzelner Zellen als auch die verwendeten Aufkleber auf die bedeckte Fläche der

Schilder umgerechnet werden. Dies liefert einen guten Anhaltspunkt, welchen Einfluss verschiedene Größen äußerer Störfaktoren auf die Klassifizierung haben.

5.10 Konsequenzen der Forschungsergebnisse

Anhand der Ergebnisse dieser Arbeit können einige Konsequenzen für die Anwendung der Klassifizierung von Straßenschildern im Bereich des Straßenverkehrs und insbesondere des autonomen Fahrens gezogen werden. Zunächst zeigen die Ergebnisse, dass CNNs in der Lage sind, hohe Accuracy-Werte zu erzielen. Dies zeigt, dass sie grundsätzlich geeignet sind, Teil der maschinellen Straßenschildererkennung zu sein. Allerdings ist auch ersichtlich, dass die Fehlerquote zu hoch ist, um die Analyse einzelner Bilder als einzige Methode zu verwenden. So wäre die Anzahl der fehlerhaft erkannten Schilder sogar mit der auf dem Trainingsdatensatz erreichten Validation Accuracy von 99,1 % bei täglich Millionen von Anwendungsfällen am Straßenverkehr immer noch zu hoch. Hier sollten weitere Methoden entwickelt werden, sodass eine Redundanz vorliegt. Da beim Fahren eine Vielzahl von Bildern aus verschiedenen Winkeln und Entfernungen eines Schildes erzeugt werden kann, wäre auch ein Zusammenfassen mehrerer Klassifizierungen für ein Schild denkbar. Außerdem wurde aufgezeigt, dass äußere Störfaktoren deutliche Einflüsse auf die Accuracy eines CNNs für diesen Einsatzzweck haben. Dies bedeutet, dass in Zukunft deutlich mehr darauf geachtet werden muss, die Schilder frei von Störfaktoren zu halten. Dies ist natürlich mit steigenden Kosten in der Wartung des Straßenverkehrnetzes verbunden.

5.11 Ausblick

Basierend auf den im Rahmen dieser Arbeit erzielten Erkenntnissen ergeben sich diverse Optionen zur weitergehenden Betrachtung des maschinellen Klassifikationsprozesses von Straßenverkehrsschildern.

Bei der Auswahl der Visualisierungsalgorithmen ist eine breitere Anlage der Betrachtung denkbar. So erzeugen Grad-CAM und Grad-CAM++ relativ ähnliche Resultate und eine Komplementierung mit Verfahren, die sich konzeptionell unterscheiden, insbesondere mit nichtlokalen Optimierern, ist ein naheliegender nächster Schritt. Auch unabhängig von grundlegend anders operierenden Verfahren könnte man eine größere Auswahl der eingesetzten Verfahren validieren.

Auch die Möglichkeit zur Kontrastierung der Ergebnisse von humaner und maschineller Kognition im Rahmen der Forschungsfrage benötigt weitere Forschung.

Zwar ist das manuelle Maskieren von allen ca. 12 500 Bildern im Testdatensatz mit einem Arbeitsaufwand verbunden, welcher im Rahmen dieser Studie nicht geleistet werden konnte, ein Vergleich mit einem Auszug ist jedoch denkbar. Dies gilt ebenso für die Positionierung der Aufkleber auf den Stockfotos. Während sich der Aufwand auf der technischen Ebene leicht abgrenzen lässt, sollte bei diesem Punkt aber berücksichtigt werden, dass für einen quantitativen Vergleich mit humaner Kognition experimentelle Verfahren der Kognitionspsychologie genutzt werden müssten, sich hier also eine Erweiterung der Forschungsfrage in den transdisziplinären Kontext ergäbe.

Aus einer hardwarebezogenen Perspektive ist davon auszugehen, dass die zunehmende Auflösung zukünftiger Kamerasysteme in diesem Bereich einen erheblichen Einfluss auf die Prozessperformanz haben könnte. So weist der GTSRB-Datensatz eine Vielzahl von verschiedenen Bildergrößen auf, welche im Bereich von mehreren Dutzend bis maximal mehreren hundert Pixel Kantenlänge liegen. Diese Auflösungen sind als eher gering anzusehen, sodass es möglich erscheint, dass mit höher auflösenden Aufnahmen andere Ergebnisse erzielt werden. Dies erzeugt im Zusammenspiel mit der beschränkten Rechenleistung der Bordcomputer von Fahrzeugen und der zeitkritischen Natur der Klassifizierungsaufgabe von Straßenverkehrsschildern ein Spannungsfeld. Da mit höherer Auflösung auch die Zahl der Parameter innerhalb des Netzwerkes und somit die benötigte Rechenleistung steigt, muss eine Auflösung gefunden werden, die einen Mittelweg zwischen Zuverlässigkeit und Rechenaufwand liefert.

Um diese Auflösung finden zu können, wird ein Datensatz mit höher aufgelösten Bildern benötigt. Eine Analyse der bereits existierenden Datensätze kann darüber Auskunft geben, ob einer oder mehrere hiervon als Grundlage nutzbar sind. Ist dies nicht der Fall, muss ein neuer Datensatz erzeugt werden. Ebenfalls ein neuer Datensatz wird vermutlich für die im vorherigen Abschnitt aufgeworfene Frage, ob das Zusammenfassen mehrerer Klassifizierungen eines Schildes aus unterschiedlichen Positionen einen positiven Einfluss auf die Zuverlässigkeit hat, benötigt.

Des Weiteren kann der Klassenumfang des Datensatzes weiter erhöht werden. So gibt es alleine in Deutschland Dutzende weiterer Schilder, die erkannt werden müssen. Daneben kann auch der regionale Fokus verschoben werden und betrachtet werden, ob die Schilder in anderen Ländern besser oder schlechter klassifiziert werden können. Da Fahrzeuge nicht ländergebunden sind, ist auch zu

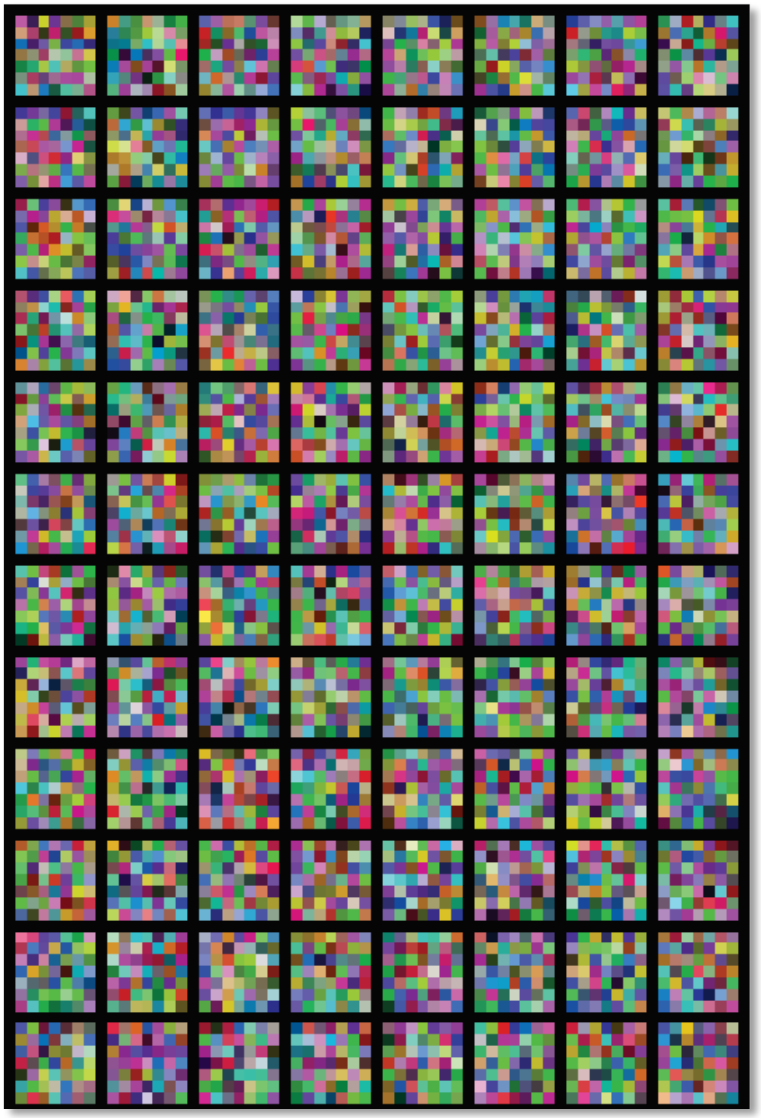
überprüfen, inwieweit ein CNN mit ähnlichen – aber nicht identischen – Formen der Straßenschilder in verschiedenen Ländern zurechtkommt.

Auch die genauere Differenzierung der Klassifizierungsleistung von CNNs kann hilfreiche Informationen liefern. So kann genauer betrachtet werden, welche Klassen fehleranfälliger sind als andere und wie ein CNN für diese optimiert werden kann. Hierbei könnte ein Fokus darauf liegen, besonders sicherheitsrelevante Schilder (z. B. *Vorfahrt gewähren* oder *Gefahrenstellen*) zu ermitteln und die Fehlerquote hier zu minimieren.

Außerdem soll auch noch das bereits erwähnte Zusammenspiel mit weiteren Methoden zur Schildererkennung genannt werden. Hier können sinnvolle Kombinationen von vorhandenen Sensoren bei bereits fürs autonome Fahren konstruierten Fahrzeugen ermittelt werden. Einen weiteren Forschungsansatz stellt das Erforschen von Alternativen zur optischen Distribution der Informationen von Straßenverkehrsschildern dar, zum Beispiel mittels 5G oder Radio-Frequency Identification (RFID).

7 Anhang

Anhang 1: Ergänzende Abbildungen



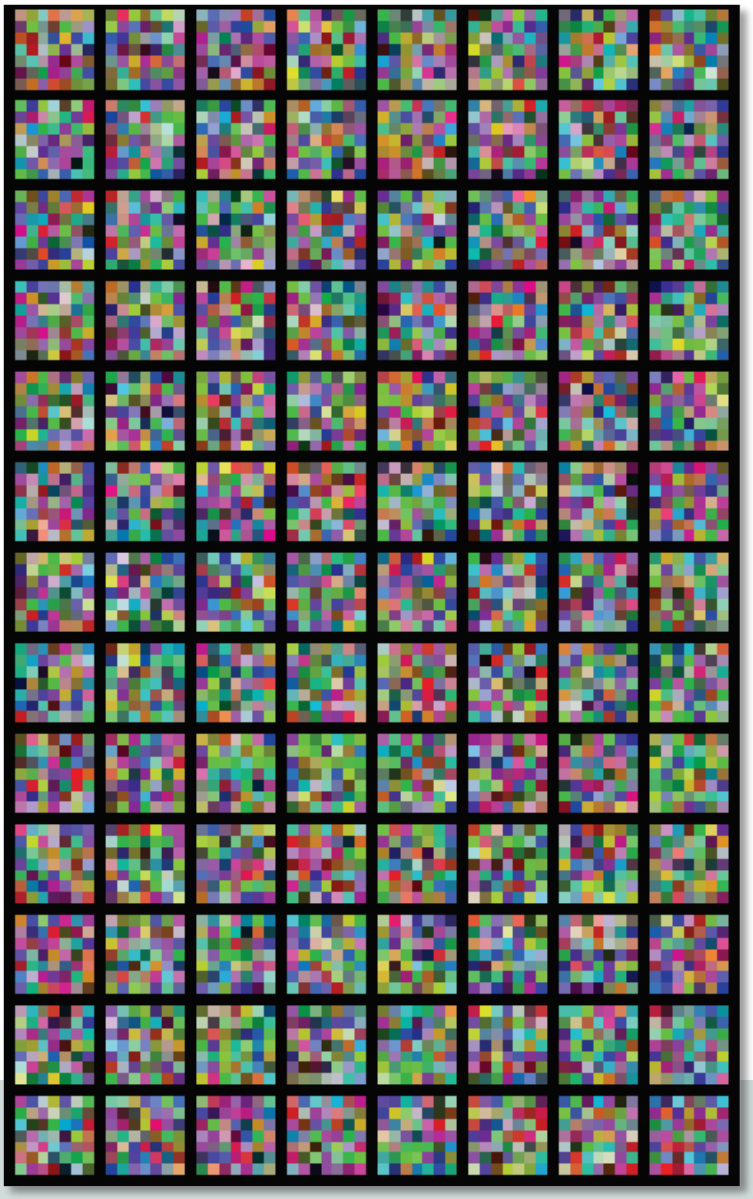
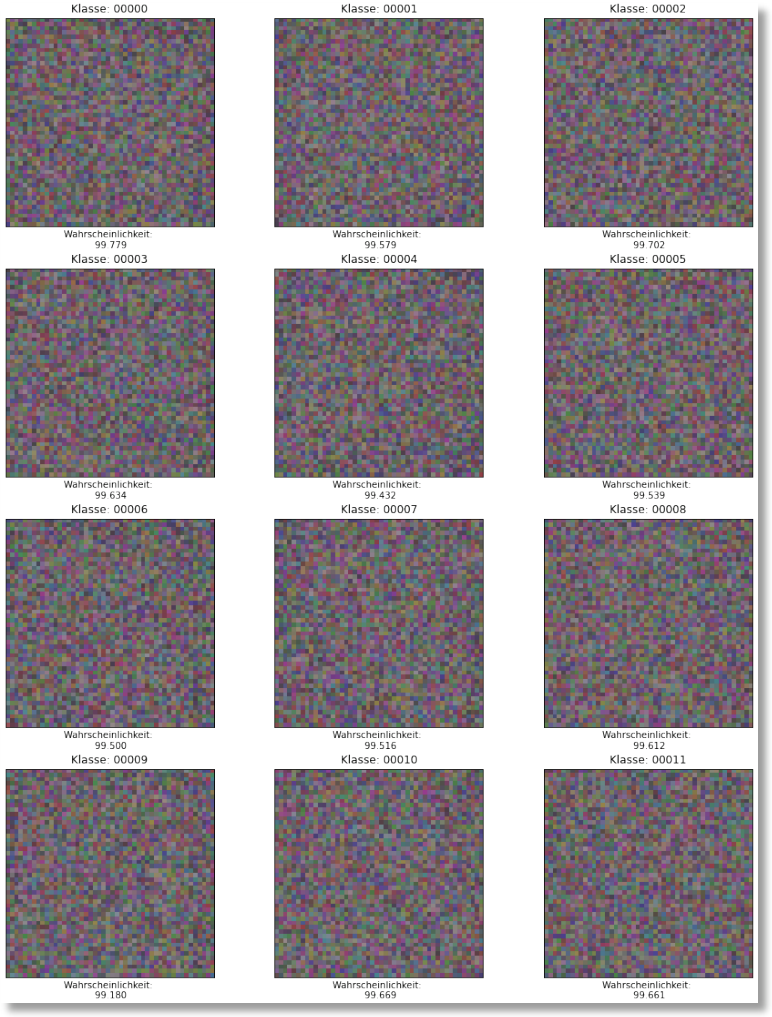
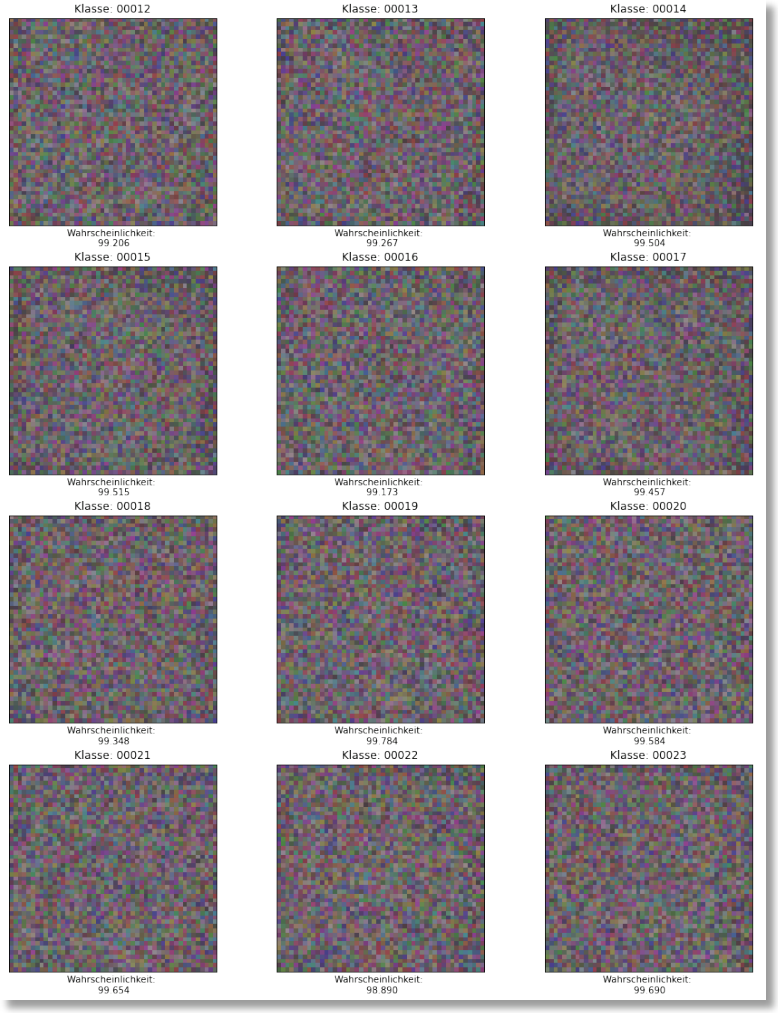
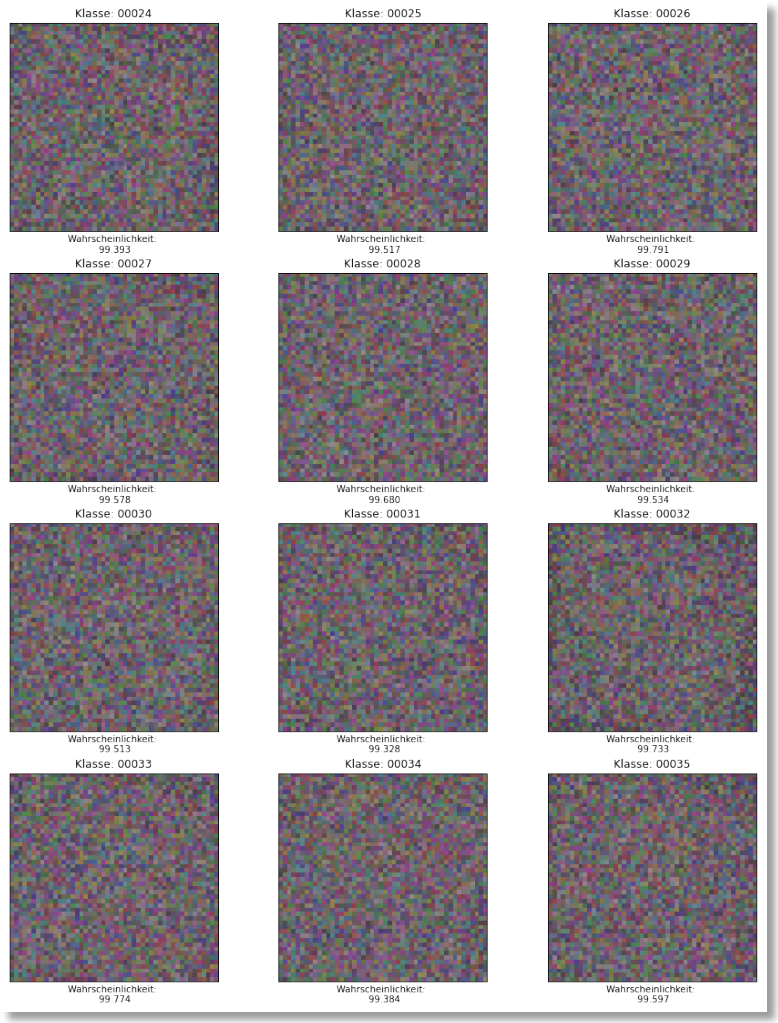


Abb. 27: Visualisierung Feature Maps nach dem ersten Convolutional Layer


Klasse: 00000
Wahrscheinlichkeit:
99.779
Klasse: 00001
Wahrscheinlichkeit:
99.579
Klasse: 00002
Wahrscheinlichkeit:
99.702
Klasse: 00003
Wahrscheinlichkeit:
99.634
Klasse: 00004
Wahrscheinlichkeit:
99.432
Klasse: 00005
Wahrscheinlichkeit:
99.539
Klasse: 00006
Wahrscheinlichkeit:
99.500
Klasse: 00007
Wahrscheinlichkeit:
99.516
Klasse: 00008
Wahrscheinlichkeit:
99.612
Klasse: 00009
Wahrscheinlichkeit:
99.180
Klasse: 00010
Wahrscheinlichkeit:
99.669
Klasse: 00011
Wahrscheinlichkeit:
99.661


Klasse: 00012
Wahrscheinlichkeit:
99.206
Klasse: 00013
Wahrscheinlichkeit:
99.267
Klasse: 00014
Wahrscheinlichkeit:
99.504
Klasse: 00015
Wahrscheinlichkeit:
99.515
Klasse: 00016
Wahrscheinlichkeit:
99.173
Klasse: 00017
Wahrscheinlichkeit:
99.457
Klasse: 00018
Wahrscheinlichkeit:
99.348
Klasse: 00019
Wahrscheinlichkeit:
99.784
Klasse: 00020
Wahrscheinlichkeit:
99.584
Klasse: 00021
Wahrscheinlichkeit:
99.654
Klasse: 00022
Wahrscheinlichkeit:
98.890
Klasse: 00023
Wahrscheinlichkeit:
99.690


Klasse: 00024
Wahrscheinlichkeit:
99.393
Klasse: 00025
Wahrscheinlichkeit:
99.517
Klasse: 00026
Wahrscheinlichkeit:
99.791
Klasse: 00027
Wahrscheinlichkeit:
99.578
Klasse: 00028
Wahrscheinlichkeit:
99.680
Klasse: 00029
Wahrscheinlichkeit:
99.534
Klasse: 00030
Wahrscheinlichkeit:
99.513
Klasse: 00031
Wahrscheinlichkeit:
99.328
Klasse: 00032
Wahrscheinlichkeit:
99.733
Klasse: 00033
Wahrscheinlichkeit:
99.774
Klasse: 00034
Wahrscheinlichkeit:
99.384
Klasse: 00035
Wahrscheinlichkeit:
99.597

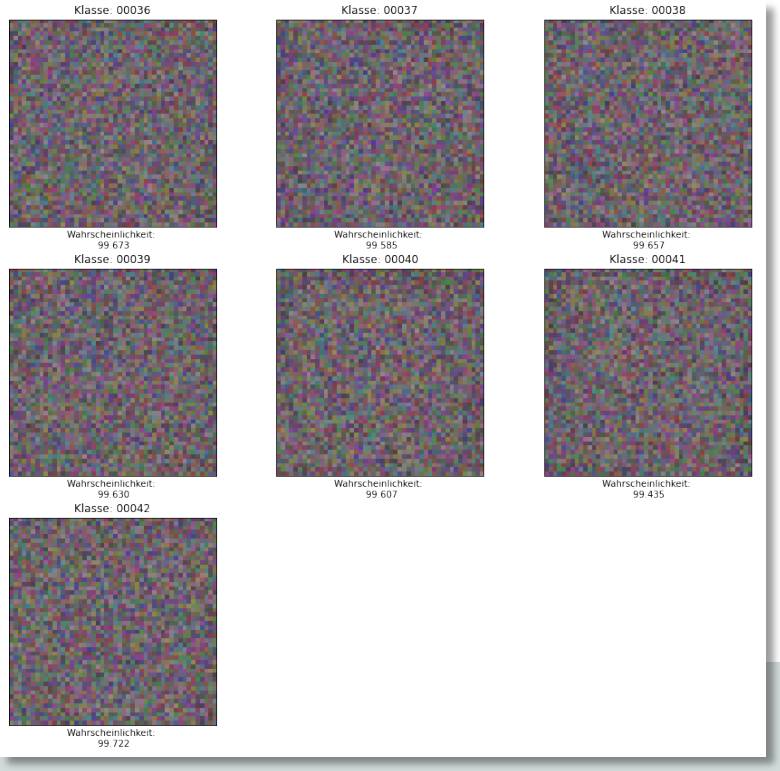


Abb. 28: Activation Maximization Resultate

Anhang 2: Ergänzende Tabellen

Durchgang	SGD	SGD mit Nesterov	Adam
0	3,7324	3,7324	3,7324
1	3,146013021	0,1675	0,0815
2	2,601388454	0,0642	0,0702
3	1,528146625	0,0443	0,0347
4	0,970382869	0,033	0,0198
5	0,570398092	0,0244	0,0163
6	0,323 388 755	0,0162	0,0272
7	0,273 840 308	0,0167	0,011
8	0,207 257 628	0,0168	0,0136
9	0,159 013 554	0,012	0,0104
10	0,125 317 514	0,0155	0,012
11	0,112 791 747	0,0097	0,009 98
12	0,107 366 696	0,0137	0,008 57
13	0,093 795 754	0,0111	0,0125
14	0,084 938 601	0,0124	0,0126
15	0,074 537 322	0,0081	0,008 33

Tab. 6: Übersicht Validation Loss der drei getesteten Verlustfunktionen

Durchgang	SGD	SGD mit Nesterov	Adam
0	0	0	0
1	0,8154	0,9857	0,9761
2	0,933	0,9941	0,9905
3	0,9795	0,9959	0,995
4	0,9879	0,9974	0,9899
5	0,9895	0,9976	0,997
6	0,9907	0,9985	0,9975
7	0,9931	0,998	0,9949
8	0,9935	0,9982	0,9977
9	0,9944	0,9982	0,9971
10	0,9951	0,9983	0,9963
11	0,9957	0,9987	0,997
12	0,9958	0,9987	0,9982
13	0,9958	0,9991	0,9986
14	0,9972	0,9988	0,9903
15	0,9963	0,9991	0,998

Tab. 7: Übersicht Validation Accuracy der drei getesteten Verlustfunktionen

Name	Version/Commit	Dokumentation (URL)
jupyterlab	2.2.9	Webseite
matplotlib	3.1.3	Webseite
nn_interpretability	3f494ae1835baa1389097c0afc71671393e50ca2	GitHub
numpy	1.19.0	Webseite
pandas	1.0.1	Webseite
Pillow	8.0.1	Webseite
pytorch	1.7.0	Webseite
pytorch-lightning	1.1.0	Webseite
pytorch-vision	0.8.1	Webseite
requests	2.22.0	Webseite
rotate_3d	614e0acf9b772007b517caca8921fe6167bab0d0	GitHub
scikit_image	0.16.2	Webseite
scikit_learn	0.23.2	Webseite
seaborn	0.11.0	Webseite
Smooth-GradCAMplusplus	e728e5f6b1fde4fb416efb2509e1583e1cdc0a8a	GitHub
split-folders	0.4.3	GitHub
tensorflow	2.4.0	Webseite

Tab. 8: Übersicht der verwendeten Python Bibliotheken

Literaturverzeichnis

Adadi, Amina, Berrada, Mohammed (XAI, 2018): Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI), in: IEEE Access, 6 (2018), S. 52139– 52160, [Zugriff: 2021-01-06]

Adebayo, Julius, Gilmer, Justin, Muelly, Michael, Goodfellow, Ian, Hardt, Moritz, Kim, Been (Überprüfung Saliency Maps, 2018): Sanity Checks for Saliency Maps, in: Bengio,

S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Hrsg.), Advances in Neural Information Processing Systems, Bd. 31, o. O.: Curran Associates, Inc., 2018, S. 1–11, [Zugriff: 2020-12-07]

Arcos-García, Álvaro, Álvarez-García, Juan A., Soria-Morillo, Luis M. (Mehrere STN, 2018): Deep Neural Network for Traffic Sign Recognition Systems: An Analysis of

Spatial Transformers and Stochastic Optimisation Methods, in: Neural Networks, 99 (2018), S. 1–15

Buda, Mateusz, Maki, Atsuto, Mazurowski, Maciej A. (Upsampling, 2018): A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks, in: Neural Networks, 106 (2018), S. 249–259

Bundesrepublik Deutschland (Verkehrszeichengröße, 2001): Allgemeine Verwaltungsvorschrift Zur Straßenverkehrs-Ordnung, o. O., 2001-01-26, URL: http : / / www . verwaltungsvorschriften-im-internet.de/bsvwvbund_26012001_S3236420014.htm [Zugriff: 2020-12-21]

Bundesrepublik Deutschland (StVO, 2013): Straßenverkehrs-Ordnung, o. O., 2013-03-06, URL: https://www.gesetze-im-internet.de/stvo_2013/StVO.pdf [Zugriff: 2021-01-02]

Bundesrepublik Deutschland (VzKat, 2017): Katalog Der Verkehrszeichen, o. O., 2017-06-20, URL: http://www.vzkat.de/2017/VzKat.htm [Zugriff: 2020-12-21]

Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V. N. (Grad-Cam++, 2018): Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), o. O., 2018, S. 839–847

Cires¸an, Dan, Meier, Ueli, Masci, Jonathan, Schmidhuber, Jürgen (Zusammenschluss Mehrerer CNNs, 2012): Multi-Column Deep Neural Network for Traffic Sign Classification, in: Neural Networks, 32 (2012), S. 333–338, [Zugriff: 2020-10-19]

Cohen, Taco S., Welling, Max (Räumliche Veränderungen, 2015): Transformation Properties of Learned Visual Representations, in: Bengio, Yoshua, LeCun, Yann (Hrsg.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, o. O., 2015, S. 1–11

Duchi, John, Hazan, Elad, Singer, Yoram (AdaGrad, 2011): Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, in: Journal of Machine Learning

Research, 12 (2011), Nr. 61, S. 2121–2159

Duchon, Claude E. (Lanczos Filter, 1979): Lanczos Filtering in One and Two Dimensions, in: Journal of Applied Meteorology and Climatology, 18 (1979), Nr. 8, S. 1016–1022

Erhan, Dumitru, Bengio, Y., Courville, Aaron, Vincent, Pascal (Activation Maximization, 2009): Visualizing Higher-Layer Features of a Deep Network, in: Technical Report, Univeristé de Montréal (2009), Nr. 1341, S. 4–6, [Zugriff: 2020-10-22]

Evgeniou, Theodoros, Pontil, Massimiliano (SVM, 2001): Support Vector Machines: Theory and Applications, in: Paliouras, Georgios, Karkaletsis, Vangelis, Spyropoulos, Constantine D., Goos, G., Hartmanis, J., van Leeuwen, J. (Hrsg.), Machine Learning and Its Applications, Bd. 2049, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, S. 249–257, [Zugriff: 2020-12-11]

Fukushima, Kunihiko, Miyake, Sei (Neocognitron, 1982): Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition, in: Competition and Cooperation in Neural Nets, o. O.: Springer, 1982, S. 267–285

Gecer, Baris, Azzopardi, George, Petkov, Nicolai (COSFIRE-Filter, 2017): Color-Blob-

Based COSFIRE Filters for Object Recognition, in: Image and Vision Computing, 57 (2017), S. 165–174, [Zugriff: 2020-10-19]

Goodfellow, Ian, Bengio, Yoshua, Courville, Aaron (Softmax, 2016): Deep learning, Cambridge, Massachusetts: MIT Press, 2016, [Zugriff: 2020-12-07]

Hinton, Geoffrey (RMSProp, 2011): Neural Networks for Machine Learning - Lecture 6a

Overview of Mini-Batch Gradient Descent, Vorlesung, Vorlesung, Toronto, CA, 2011,

URL: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf [Zugriff: 2020-12-13]

Holzinger, Andreas, Biemann, Chris, Pattichis, Constantinos S., Kell, Douglas B. (Fachexperten, 2017): What Do We Need to Build Explainable AI Systems for the Medical Domain?, in: CoRR, abs/1712.09923 (2017), S. 1–28

Holzinger, Andreas, Plass, Markus, Holzinger, Katharina, Crisan, Gloria Cerasela, Pintea, Camelia-Mihaela, Palade, Vasile (Vertrauen, 2017): A Glass-Box Interactive Machine Learning Approach for Solving NP-Hard Problems with the Human-in-the-Loop, in: CoRR, abs/1708.01104 (2017), S. 1–26

Hubel, D. H., Wiesel, T. N. (Receptive Field, 1959): Receptive Fields of Single Neurones in the Cat's Striate Cortex, in: The Journal of Physiology, 148 (1959), Nr. 3, S. 574– 591, [Zugriff: 2020-10-17]

Hubel, David H, Wiesel, Torsten N (Receptive Field, 1962): Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex, in: The Journal of physiology, 160 (1962), Nr. 1, S. 106–154

Ioffe, Sergey, Szegedy, Christian (Batch Normalization, 2015): Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in: Bach, Francis, Blei, David (Hrsg.), Proceedings of the 32nd International Conference on Machine Learning, Bd. 37, Proceedings of Machine Learning Research, Lille, France: PMLR, 2015-07-07/2015-07-09, S. 448–456

Jaderberg, Max, Simonyan, Karen, Zisserman, Andrew, kavukcuoglu, koray (STN, 2015): Spatial Transformer Networks, in: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (Hrsg.), Advances in Neural Information Processing Systems, Bd. 28, o. O.: Curran Associates, Inc., 2015, S. 1–9

Kingma, Diederik P., Ba, Jimmy (Adam, 2015): Adam: A Method for Stochastic Optimization, in: Bengio, Yoshua, LeCun, Yann (Hrsg.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, o. O., 2015, S. 1–15

Krizhevsky, Alex, Sutskever, Ilya, Hinton, Geoffrey E. (Tanh vs. ReLU, 2017): ImageNet Classification with Deep Convolutional Neural Networks, in: Commun. ACM, 60 (2017), Nr. 6, S. 84–90, [Zugriff: 2020-12-11]

LeCun, Yann, Bengio, Yoshua, Hinton, Geoffrey (Deep Learning, 2015): Deep Learning, in: Nature, 521 (2015), Nr. 7553, S. 436–444, [Zugriff: 2021-01-05]

LeCun, Yann, Boser, Bernhard, Denker, John, Henderson, Donnie, Howard, R., Hubbard, Wayne, Jackel, Lawrence (Handschriftserkennung, 1990): Handwritten Digit Recognition with a Back-Propagation Network, in: Touretzky, D. (Hrsg.), Advances in Neural Information Processing Systems, Bd. 2, o. O.: Morgan-Kaufmann, 1990, S. 396–404

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, Haffner, Patrick (Gradientenbasiertes Lernen, 1998): Gradient-Based Learning Applied to Document Recognition, in: Proceedings of the IEEE, 86 (1998), Nr. 11, S. 2278–2324

Lenc, Karel, Vedaldi, Andrea (Räumliche Veränderungen, 2015): Understanding Image Representations by Measuring Their Equivariance and Equivalence, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), o. O., 2015-06, S. 991–999

Litman, Todd (Entwicklungsvorhersage, 2020): Autonomous Vehicle Implementation Predictions: Implications for Transport Planning, o. O.: Victoria Transport Policy Institute, 2020, S. 1–45, URL: https://www.vtpi.org/avip.pdf. [Zugriff: 2021-01-06]

Maas, Andrew L., Hannun, Awni Y., Ng, Andrew Y. (Leaky ReLU, 2013): Rectifier Nonlinearities Improve Neural Network Acoustic Models, in: ICML 2013, Workshop on Deep Learning for Audio, Speech and Language Processing, o. O., 2013, S. 1–6, [Zugriff: 2020-12-11]

McCulloch, Warren S., Pitts, Walter (Nervenaktivitäten, 1943): A Logical Calculus of the Ideas Immanent in Nervous Activity, in: Bulletin of Mathematical Biophysics, 5 (1943), Nr. 4, S. 115–133, [Zugriff: 2020-10-11]

Nesterov, Yu (Nesterov-Momentum, 1983): A Method of Solving a Convex ProgramMing Problem with Convergence Rate O(1/Sqr(k)), in: Soviet Mathematics Doklady, (1983), Nr. 27, S. 372–376

Qin, Zhuwei, Yu, Fuxun, Liu, Chenchen, Chen, Xiang, ‚George Mason University, 4400 University Dr, Fairfax, VA 22030, USA, ‚Clarkson University, 8 Clarkson Ave, Potsdam, NY 13699, USA (Visuell Cortex vs. CNN, 2018): How Convolutional Neural Networks See the World — A Survey of Convolutional Neural Network Visualization

Methods, in: Mathematical Foundations of Computing, 1 (2018), Nr. 2, S. 149–180, [Zugriff: 2020-10-17]

Ribeiro, Marco Tulio, Singh, Sameer, Guestrin, Carlos (Wölfe vs. Huskeys, 2016): "Why Should I Trust You?": Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco California USA: ACM, 2016-08-13, S. 1142–1143, [Zugriff: 2021-01-05]

On-Road Automated Driving (ORAD) committee (Automatisierungsstufen, 2018): Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, J3016_201806, o. O.: SAE International, 2018-06-15, S. 1–35, URL: https://www.sae.org/content/j3016_201806 [Zugriff: 2021-01-05]

Robbins, Herbert, Monro, Sutton (SGD, 1951): A Stochastic Approximation Method, in: The annals of mathematical statistics (1951), S. 400–407

Rosenblatt, F. (Perceptron, 1958): The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. In: Psychological Review, 65 (1958), Nr. 6, S. 386–408, [Zugriff: 2020-10-11]

Rumelhart, David E, Hinton, Geoffrey E, Williams, Ronald J (Backpropagation, 1986): Learning Representations by Back-Propagating Errors, in: nature, 323 (1986), Nr. 6088, S. 533–536

Šegvic´, Siniša, Brkic´, Karla, Kalafatic´, Zoran, Stanisavljevic´, Vladimir, Ševrovic´, Marko, Budimir, Damir, Dadic´, Ivan (MASTIF, 2010): A Computer Vision Assisted Geoinformation Inventory for Traffic Infrastructure, in: 13th International IEEE Conference on Intelligent Transportation Systems, IEEE, o. O., 2010, S. 66–73

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D. (GradCam, 2017): Grad-Cam: Visual Explanations from Deep Networks via Gradient-Based Localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), Los Alamitos, CA, USA: IEEE Computer Society, 2017-10, S. 618–626

Sermanet, Pierre, LeCun, Yann (Mehrstufige CNNs, 2011): Traffic Sign Recognition with Multi-Scale Convolutional Networks, in: The 2011 International Joint Conference on

Neural Networks, 2011 International Joint Conference on Neural Networks (IJCNN 2011 - San Jose), San Jose, CA, USA: IEEE, 2011-07, S. 2809–2813, [Zugriff: 2020-10-19]

Shakhuro, Vladislav, Konushin, Anton (RTSD, 2016): Russian Traffic Sign Images Dataset, in: Computer Optics, 40 (2016), Nr. 2, S. 294–300

Simonyan, Karen, Vedaldi, Andrea, Zisserman, Andrew (Saliency Map, 2014): Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, in: Bengio, Yoshua, LeCun, Yann (Hrsg.), 2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings, International Conference on Learning Representations, Banff, AB, Canada, 2014, S. 1–8

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, Riedmiller, Martin A. (Guided Backpropagation, 2015): Striving for Simplicity: The All Convolutional Net, in: Bengio, Yoshua, LeCun, Yann (Hrsg.), 3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings, International Conference on Learning Representations, San Diego, CA, USA, 2015, S. 1–14

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov, Ruslan (Dropout, 2014): Dropout: A Simple Way to Prevent Neural Networks from Overfitting, in: Journal of Machine Learning Research, 15 (2014), Nr. 56, S. 1929–1958, [Zugriff: 2020-06-26]

Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C. (GTSRB, 2012): Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition, in: Neural Networks (2012), S. 323–332

Szandała, Tomasz (Aktivierungsfunktionen, 2021): Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks, in: Bhoi, Akash Kumar, Mallick, Pradeep Kumar, Liu, Chuan-Ming, Balas, Valentina E. (Hrsg.), Bio-Inspired Neurocomputing, Singapore: Springer Singapore, 2021, S. 203–224

Taigman, Yaniv, Yang, Ming, Ranzato, Marc'Aurelio, Wolf, Lior (Deepface, 2014): Deepface: Closing the Gap to Human-Level Performance in Face Verification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, o. O., 2014, S. 1701–1708

Toro-Vizcarrondo, Carlos, Wallace, T. D. (Mittlere Quadratischer Abweichung, 1968): A Test of the Mean Square Error Criterion for Restrictions in Linear Regression, in: Journal of the American Statistical Association, 63 (1968), Nr. 322, S. 558–572

United Nations (Konvention Straßenverkehr, 1977): Convention on Road Traffic, Wien, Österreich, 1977-05-21, [Zugriff: 2020-10-21]

Wei, Donglai, Zhou, Bolei, Torrabla, Antonio, Freeman, William T. (Mean Image Initialization, 2015): Understanding Intra-Class Knowledge inside CNN, in: CoRR, abs/1507.02379 (2015), S. 1–7

Ye, Chengxi, Evanusa, Matthew, He, Hua, Mitrokhin, Anton, Goldstein, Tom, Yorke, James A., Fermüller, Cornelia, Aloimonos, Yiannis (Deconvolution, 2020): Network Deconvolution, in: 8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, OpenReview.net, 2020, S. 1–20

Yosinski, Jason, Clune, Jeff, Nguyen, Anh Mai, Fuchs, Thomas J., Lipson, Hod (Deep Visualization, 2015): Understanding Neural Networks through Deep Visualization, in: CoRR, abs/1506.06579 (2015), S. 1–12

Yu, Rulei, Shi, Lei (Visualisierungsalgorithmen, 2018): A User-Based Taxonomy for Deep Learning Visualization, in: Visual Informatics, 2 (2018), Nr. 3, S. 147–154, [Zugriff: 2021-01-06]

Zhou, Bolei, Khosla, Aditya, Lapedriza, Agata, Oliva, Aude, Torralba, Antonio (CAM, 2016): Learning Deep Features for Discriminative Localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA: IEEE, 2016-06, S. 2921–2929, [Zugriff: 2020-06-16]

Internetquellen

Clark, Alex (Pillow, 2020): Pillow (PIL Fork) Documentation, <https : / / buildmedia.readthedocs . org / media / pdf / pillow / latest / pillow . pdf> (2020-12-20) [Zugriff: 2020-12-20]

Cunningham, Padraig, Delany, Sarah Jane (K-Nearest Neighbour Classifiers, 2020): KNearest Neighbour Classifiers: 2nd Edition (with Python Examples), arXiv: 2004. 04523 [cs, stat], <https://arxiv.org/abs/2004.04523> (2020-04-29) [Zugriff: 2020-12-15]

Dumoulin, Vincent, Visin, Francesco (Spaltenanzahl, 2018): A Guide to Convolution Arithmetic for Deep Learning, arXiv: 1603.07285 [cs, stat], <https://arxiv.org/abs/ 1603.07285> (2018-01-11) [Zugriff: 2020-12-16]

Ertler, Christian, Mislej, Jerneja, Ollmann, Tobias, Porzi, Lorenzo, Neuhold, Gerhard, Kuang, Yubin (Mapillary, 2020): The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale, arXiv: 1909.04422 [cs], <https://arxiv.org/abs/ 1909.04422> (2020-05-07) [Zugriff: 2020-12-21]

Falcon, William (PyTorch Lightning, 2020): PyTorch Lightning Documentation, <https:// pytorch-lightning.readthedocs.io/en/latest/> (2020-12-19) [Zugriff: 2020-12-19]

Filter, Johannes (Split-Folders, 2020): Jfilter/Split-Folders, <https://github.com/jfilter/splitfolders> (2020-11-01) [Zugriff: 2020-12-18]

Google (Colab, 2020): Colaboratory, <https://research.google.com/colaboratory/faq.html> (2020-12-16) [Zugriff: 2020-12-16]

Gu, Tianyu, Dolan-Gavitt, Brendan, Garg, Siddharth (BadNets, 2019): BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain, arXiv: 1708 . 06733 [cs], <https://arxiv.org/abs/1708.06733> (2019-03-11) [Zugriff: 2020-12-17]

hans66hsu (Nn_interpretability, 2020): Hans66hsu/Nn_interpretability, <https : / / github . com/hans66hsu/nn_interpretability> (2020-10-14) [Zugriff: 2020-12-20]

Heinze, Josephine (Vorbeifahrt rechts, 2019): Stadt Leipzig testet Anti-GraffitiBeschichtung für Verkehrsschilder, <https://www.lvz.de/Leipzig/Lokales/StadtLeipzig-testet-Anti-Graffiti-Beschichtung-fuer-Verkehrsschilder> (2019-03-16) [Zugriff: 2021-01-01]

Hu, Hou-Ning (Rotate_3d, 2017): Eborboihuc/Rotate_3d, <https://github.com/eborboihuc/ rotate_3d> (2017-05-22) [Zugriff: 2020-12-20]

Hunter, John, Dale, Darren, Firing, Eric, Droettboom, Michael, Matplotlib development team (Imsave, 2020): Imsave - Matplotlib 3.1.3, <https : / / matplotlib. org / 3 . 1 . 3 / api/_as_gen/matplotlib.pyplot.imsave.html> (2020-02-09) [Zugriff: 2021-01-05]

Institut für Neuroinformatik der Ruhr-Universität Bochum (Resultate GTSRB, 2019):

GTSRB Results, <https://benchmark.ini.rub.de/gtsrb_results.html> (2019-05-10) [Zugriff: 2020-09-26]

IPython development team (Jupyter Notebook, 2020): The Jupyter Notebook — IPython, <https://ipython.org/notebook.html> (2020-12-16) [Zugriff: 2020-12-16]

Ishikawa, Yuchi (SmoothGradCAMplusplus, 2019): Yiskw713/SmoothGradCAMplusplus, <https : / / github. com / yiskw713 / SmoothGradCAMplusplus> (2019-08-29) [Zugriff: 2020-12-20]

Johnson, Justin, Zakka, Kevin (CS231n, 2020): CS231n Convolutional Neural Networks for Visual Recognition, <https://cs231n.github.io/convolutional-networks/> (2020-04-11) [Zugriff: 2020-09-26]

McKinsey and Company (Revolution, 2021): Autonomous Driving | MCFM | McKinsey & Company, <https://www.mckinsey.com/features/mckinsey-center-for-future-mobility/ overview/autonomous-driving> (2021-01-05) [Zugriff: 2021-01-05]

Nielsen, Michael, Determination Press (Deep Learning, 2015): Neural networks and deep learning, <http://neuralnetworksanddeeplearning.com> (2015) [Zugriff: 2020-09-26]

NumPy community (Numpy, 2020): NumPy Reference, <https://numpy.org/doc/stable/> (2020-06-29) [Zugriff: 2020-12-20]

scikit-image development team (Rgb2gray, 2019): Rgb2gray - Scikit-Image 0.16.2, <https:// scikit-image.org/docs/0.16.x/api/skimage.color.html?highlight=rgb2gray#rgb2gray> (2019-10-14) [Zugriff: 2021-01-04]

Simonyan, Karen, Zisserman, Andrew (VGG-19, 2015): Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv: 1409.1556 [cs], <https://arxiv. org/abs/1409.1556> (2015-04-10) [Zugriff: 2020-12-23]

Spiegel (Alpha Go, 2016): Google-Computer Alpha Go besiegt Lee Sodol 4:1 - DER SPIEGEL - Netzwelt, <https://www.spiegel.de/netzwelt/gadgets/alphago-besiegt-leesedol-mit-4-zu-1-a-1082388.html> (2016-03-15) [Zugriff: 2020-10-11]

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, Fergus, Rob (Perturbation, 2014): Intriguing Properties of Neural Networks, arXiv: 1312 . 6199, <https : / / arxiv. org / abs / 1312 . 6199> (2014-02-19) [Zugriff: 2021-01-06]

Torch Contributors (Cross-Entropy, 2020): Crossentropyloss, <https : / / pytorch . org / docs / stable / generated / torch . nn . CrossEntropyLoss. html> (2020-11-18) [Zugriff: 2020-12-16]

Torch Contributors (Normalize, 2020): Normalize, <https : / / pytorch . org / docs / stable / torchvision/transforms.html#torchvision.transforms.Normalize> (2020-10-16) [Zugriff: 2020-12-20]

Torch Contributors (Reproduzierbarkeit, 2020): Reproducibility — PyTorch 1.7.0 Documentation, <https://pytorch.org/docs/stable/notes/randomness.html#reproducibility> (2020-10-13) [Zugriff: 2020-12-23]

United Nations (Teilnehmer Konvention, 2020): United Nations Treaty Collection, <https: / / treaties. un . org / Pages / ViewDetailsIII . aspx ? src = TREATY & mtdsg _ no = XI - B 19&chapter=11&Temp=mtdsg3&lang=en> (2020-10-21) [Zugriff: 2020-10-21]

US Department of Transportation (Fahrzeuganzahl, 2019): UBER Elevate Symposium | US Department of Transportation, <https://www.transportation.gov/briefing-room/uberelevate-symposium> (2019-06-11) [Zugriff: 2021-01-05]

Xiong, Yuanhao, Liu, Xuanqing, Lan, Li-Cheng, You, Yang, Si, Si, Hsieh, Cho-Jui (Verlustfunktionen, 2020): How Much Progress Have We Made in Neural Network Training? A New Evaluation Protocol for Benchmarking Optimizers, arXiv: 2010.09889 [cs.LG], <https://arxiv.org/abs/2010.09889> (2020-10-19) [Zugriff: 2020-12-22]

kostenloser Download
unter **fom-ifes.de**

Hernes, D. / Lehrbass, F. / Maucy, K. (2021): Big Data basierte Analyse des Einflusses traditioneller und neuartiger Faktoren auf Mietpreise in Düsseldorf, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 25, 2021, ISSN (eBook) 2569-5355, ISBN (eBook) 978-3-89275-426-8

Lehrbass, F. (2021): Deep Learning Diagnostics – How to Avoid Being Fooled by TensorFlow, PyTorch, or MXNet with the Help of Modern Econometrics, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 24, 2021, ISSN (eBook) 2569-5355, ISBN (eBook) 978-3-89275-424-4

Lehrbass, F. / Wörndl, F. (2021): Was treibt die Renditen von Hedgefonds? Eine empirische Untersuchung ausgewählter Hedgefonds Strategien, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 23, 2021, ISSN (eBook) 2569-5355, ISBN (eBook) 978-3-89275-422-0

Kladroba, A. / Friz, K. / Buchmann, T. / Wolf, P. (2020): Netzwerk- und Outputmessung – Indikatorik für transformative Technologiefelder (NEO-Indikatorik), in: Krol, B. / Kladroba, A. (Hrsg.), ifes Schriftenreihe, Band 22, 2020, ISSN (eBook) 2569-5355, ISBN (eBook) 978-3-89275-420-6

Bähren, T. / Maasjosthusmann, R. / Walter, A. / Lehrbass, F. (2020): Praktische Umsetzung von Business Analytics im Mediensektor: Predictive Analytics im Filmgeschäft, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 21, 2020, ISSN (eBook) 2569-5355, ISBN (eBook) 978-3-89275-418-3

Kladroba, A. (2019): Der Einfluss mathematischer Methoden auf das Ergebnis von Mannschaftswettkämpfen: Eine Simulationsrechnung, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 20, 2019, ISSN (eBook) 2569-5355, ISBN (eBook) 978-3-89275-416-9

Raasch, A. / Lehrbass, F. (2019): Investmentstrategien im Rahmen von Übernahmen börsennotierter Gesellschaften – Merger Arbitrage und Maschinelles Lernen, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 19, 2019, ISSN 2191-3366, ISBN 978-3-89275-413-8

Hagemann, D. / Lehrbass, F. (2018): Prognosemodelle für Länderrisiken: Logit- und Deep Learning-Methoden im Vergleich, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 18, 2018, ISSN 2191-3366, ISBN 978-3-89275-411-4

Graalmann, M.-P. / Lehrbass, F. (2018): Eignung von Varianz-Kovarianz-Ansätzen und Copula-Modellen zur Risikoaggregation in bankaufsichtlichen Risikotragfähigkeitskonzepten, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 17, 2018, ISSN 2191-3366, ISBN 978-3-89275-409-1

Cox, P. / Lehrbass, F. (2018): Determinanten der Replikationsgüte von Exchange Traded Funds, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 16, 2018, ISSN 2191-3366, ISBN 978-3-89275-407-7

Lehrbass, F. / Scheipers, N. (2017): Determinanten der Höhe von Wirtschaftsprüfungshonoraren am Beispiel von gelisteten Unternehmen im Prime Standard, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 15, 2017, ISSN 2191-3366, ISBN 978-3-89275-406-0

Schwarz, J. (2017): Ergebnisse der Analyse von Studienabbrüchen, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 14, 2017, ISSN 2191-3366, ISBN 978-3-89275-405-3

Lehrbass, F. (2016): Risikomessung für den globalen Kohlehandel: Einfache und fortgeschrittene Verfahren nebst Backtesting sowie ein Vergleich mit IFRS 7, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 13, 2016, ISSN 2191-3366, ISBN 978-3-89275-404-6

Godbersen, H. (2016): Die Means-End Theory of Complex Cognitive Structures – Entwicklung eines Modells zur Repräsentation von verhaltensrelevanten und komplexen Kognitionsstrukturen für die Wirtschafts- und Sozialwissenschaften, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 12, 2016, ISSN 2191-3366, ISBN 978-3-89275-403-9

Seng, A. / Landherr, G. (2015): Vielfalt leben und Vielfalt gestalten – Diversity Management in der Lehre, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 11, 2015, ISSN 2191-3366, ISBN 978-3-89275-402-2

Gansser, O. A. / Schutkin, A. (2014): Studie zur Validierung der Persönlichkeitsmerkmale Abenteuerlust und Routineverhalten, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 10, 2014, ISSN 2191-3366, ISBN 978-3-89275-401-5

Gansser, O. A. (2014): Marketingplanung als Instrument zur Krisenbewältigung, in: Krol, B. (Hrsg.), ifes Schriftenreihe, Band 9, 2014, ISSN 2191-3366, ISBN 978-3-89275-400-8

Runia, P. M. / Wahl, F. / Rüttgers, C. (2013): Das Markenimage von Hersteller- und Handelsmarken: Eine empirische Analyse der Imagekomponenten von Körperpflegemarken auf der Grundlage eines Markenidentitätskonzeptes, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 8, 2013, ISSN 2191-3366

Naskrent, J. / Rüttgers, C. (2013): Sportmonitor Essen 2013: Eine empirische Analyse über das Image regionaler Sportvereine und ihre Sponsoring- und Promotionangebote, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 7, 2013, ISSN 2191-3366

Seng, A. / Fiesel, L. / Rüttgers, C. (2013): Akzeptanz der Frauenquote, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 6, 2013, ISSN 2191-3366

Naskrent, J. / Rüttgers, C. (2012): Wahrnehmung von Werbung mit Sportereignisbezug: Eine empirische Analyse der Einschätzung von Sponsoring und Ambush-Marketing im Rahmen der Fußball-Europameisterschaft und der Olympischen Spiele im Jahr 2012, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 5, 2012, ISSN 2191-3366

Seng, A. / Fiesel, L. / Krol, B. (2012): Erfolgreiche Wege der Rekrutierung in Social Networks, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 4, 2012, ISSN 2191-3366

Heinemann, S. / Krol, B. (2011): Nachhaltige Nachhaltigkeit: Zur Herausforderung der ernsthaften Integration einer angemessenen Ethik in die Managementausbildung, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 2, 2011, ISSN 2191-3366

Hermeier, B. / Rettig, P. / Krol, B. (2010): Marken- und Produktmanagement durch Nutzung von Sportgroßereignissen: Möglichkeiten und Grenzen für Industrie und Handel, in: Krol, B. (Hrsg.), KCS Schriftenreihe, Band 1, 2010, ISSN 2191-3366

ISBN (Print) 978-3-89275-427-5 ISSN (Print) 2191-3366
ISBN (eBook) 978-3-89275-428-2 ISSN (eBook) 2569-5355

Institut für Empirie & Statistik
der FOM Hochschule
für Oekonomie & Management

FOM Hochschule

FOM. Die Hochschule. Für Berufstätige.

Die mit bundesweit über 57.000 Studierenden größte private Hochschule Deutschlands führt seit 1993 Studiengänge für Berufstätige durch, die einen staatlich und international anerkannten Hochschulabschluss (Bachelor/Master) erlangen wollen.

Die FOM ist der anwendungsorientierten Forschung verpflichtet und verfolgt das Ziel, adaptionsfähige Lösungen für betriebliche bzw. wirtschaftsnahe oder gesellschaftliche Problemstellungen zu generieren. Dabei spielt die Verzahnung von Forschung und Lehre eine große Rolle: Kongruent zu den Masterprogrammen sind Institute und KompetenzCentren gegründet worden. Sie geben der Hochschule ein fachliches Profil und eröffnen sowohl Wissenschaftlerinnen und Wissenschaftlern als auch engagierten Studierenden die Gelegenheit, sich aktiv in den Forschungsdiskurs einzubringen.

Weitere Informationen finden Sie unter **fom.de**

ifes

Zunehmende Digitalisierung erfordert und ermöglicht datenbasierten Erkenntnisgewinn und fundiertes unternehmerisches Handeln. Um aus den allgegenwärtigen Daten die richtigen Schlüsse zu ziehen, ist überall eine kritische Methodenkompetenz erforderlich. Der wissenschaftliche Fokus der ifes-Akteure liegt dabei in den Bereichen der empirischen Unternehmens-, Markt- und Konsumentenforschung, der angewandten Statistik, des Data Minings und der Finanzstatistik.

Das ifes verfolgt das Ziel, empirische Kompetenzen an der FOM zu bündeln und die angewandte Forschung im empirischen Bereich der Hochschule weiter voranzutreiben. Damit nimmt das ifes eine zentrale Stellung im Bereich der Entwicklung und Unterstützung der Methodenausbildung in der Lehre der Bachelor- und Masterstudiengänge sowie im Promotionsprogramm der FOM ein.

Weitere Informationen finden Sie unter **fom-ifes.de**

SCIENCE BLOG

Im Forschungsblog werden unter dem Titel „FOM forscht“ Beiträge und Interviews rund um aktuelle Forschungsthemen und -aktivitäten der FOM Hochschule veröffentlicht.

Besuchen Sie den Blog unter fom-blog.de