

Carl-von-Bach-Gymnasium Stollberg

Jugend forscht Projekt

**Kann ein Convolutional Neural Network einen Menschen
im Klassifizieren von Verkehrsschildern übertreffen?**

Verfasser: Tom Haustein

Kurs MA12/2; Abiturjahrgang 2021

Fachbereich: Informatik/Technik

Betreuer:

Frank Rauer Carl-von-Bach-Gymnasium Stollberg
M. Sc. Peter Weissig Technische Universität Chemnitz

Abgabedatum: 25.01.2021

Kurzfassung

Mit dem Projekt sollte herausgefunden werden, ob ein Convolutional Neural Network in der Lage ist, bestimmte Aufgaben des autonomen Fahrens zu übernehmen. Dafür habe ich in meinem Versuch ein eigenes neuronales Netz trainiert, welches Verkehrsschilder richtig klassifizieren sollte. Somit kann am Ende auch verglichen werden, ob dieses neuronale Netz sogar genauer in der Klassifikation als ein Mensch ist.

Allerdings ist das Netz nur in der Lage Bilder zu klassifizieren, da es keine Bilderkennung besitzt. Das bedeutet, dass es noch nicht im Straßenverkehr einsetzbar ist, da es nur Bilder von Verkehrsschildern auf kurze Distanz richtig klassifizieren kann. Trainiert wurde das Netz mit dem GTSRB-Datensatz, der Bilder von 43 verschiedenen Verkehrsschildern aus Deutschland beinhaltet.

Letztendlich konnte das neuronale Netz durch verschiedene Trainingsmethoden und Modifikationen eine Genauigkeit von 99,11 % in dem dazugehörigen Testdatensatz erreichen. Somit ist das Netz rund 0,3 % genauer als ein Mensch, was einen ziemlich großen Unterschied darstellt und im Straßenverkehr Unfälle reduzieren könnte. Mit diesem Ergebnis erreicht das Netz eines der international besten Ergebnisse.

Somit konnte gezeigt werden, dass neuronale Netze in der Lage sind, den Menschen in bestimmten Aufgaben zu übertreffen. Ausgehend davon kann man auch sagen, dass neuronale Netze in der Lage sind, Aufgaben des autonomen Fahrens zu übernehmen.



TEAM	METHOD	TOTAL	SUBSET
[156] DeepKnowledge Seville	CNN with 3 Spatial Transformers	99.71%	99.71%
[3] IDSIA ★	Committee of CNNs	99.46%	99.46%
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%	98.97%
[1] INI-RTCV ★	Human Performance	98.84%	98.84%
[4] sermanet ★	Multi-Scale CNNs	98.31%	98.31%
[2] CAOR ★	Random Forests	96.14%	96.14%

```
Train on 39209 samples
39209/39209 [=====] - 37s 945us/sample - loss: 8.0148e-05 - acc: 1.0000
12630/12630 [=====] - 2s 175us/sample - loss: 0.0735 - acc: 0.9911
0.99105304 Batchsize: 64
NEUER BESTWERT !!!
```

Inhaltsverzeichnis

Abkürzungsverzeichnis	3
1. Einleitung	3
2. Ziel des Versuches	4
3. Der Datensatz	5
4. Aufbau des Convolutional Neural Network	6
5. Durchführung	8
6. Auswertung	12
7. Einordnung des Ergebnis	17
8. Fazit	18
9. Literaturverzeichnis	19
10. Ähnliche oder weiterführende Paper	19
11. Abbildungsverzeichnis	20
12. Unterstützungsleistung	20

Abkürzungsverzeichnis

KI	Künstliche Intelligenz
AI	artificial intelligence
KNN	Künstliche neuronale Netze
ReLU	rectified linear unit
CNN	Convolutional Neural Network
GTSRB	German Traffic Sign Recognition Benchmark

1. Einleitung

Künstliche Intelligenz und autonomes Fahren sind Themen, die viele Leute als die Themen der Zukunft ansehen. Durch dauerhafte Forschung auf beiden Gebieten entwickeln sich diese ständig weiter. Deshalb habe ich mich entschieden ein Projekt auf diesen beiden Gebieten zu beginnen, da diese mich schon immer interessierten und ich einen tieferen Einblick in diese gewinnen wollte.

Dieser Versuch ist ein Teil meiner Besonderen Lernleistung der Sekundarstufe 2. Für Jugend forscht wurde dieser noch weitergeführt und erweitert. Mit dem Versuch soll am Beispiel einer Verkehrsschilderkennung demonstriert werden, dass künstliche Intelligenz auch bestimmte Aufgaben im autonomen Fahren übernehmen und somit beispielsweise auch die Unfallrate im Straßenverkehr durch autonome Fahrzeuge gesenkt werden kann. Gleichzeitig sollte es demonstrieren, welche Möglichkeiten Convolutional Neural Networks bieten und dass diese sogar Menschen in gewissen Aufgaben übertreffen können.

Nach der Einreichung und Präsentation bei Jugend forscht wird das Projekt auch auf die Plattform „Github“ hochgeladen werden, mitsamt allen Programmen und dem trainierten neuronalen Netz im hdf5-Format. Somit können vielleicht andere Personen einen Einblick in das Thema künstliche Intelligenz und neuronale Netze gewinnen. Dazu können durch das Projekt eventuell auch Beiträge zur Forschung in diesem Bereich

geliefert werden.

Der Versuch soll sich rein mit den informatischen, mathematischen und technischen Methoden, Umsetzungen, Möglichkeiten und Problematiken beschäftigen und keinerlei Urteil über ethische Fragen zum autonomen Fahren oder zu künstlicher Intelligenz bilden.

2. Ziel des Versuches

Das Ziel dieses Versuches soll es sein, ein eigenes Convolutional Neural Network zu entwickeln, welches Verkehrsschilder klassifizieren kann. Es soll besonders hervorgehoben werden, wie es sich im Training entwickelt und welche Methoden eingesetzt werden können, um die Ergebnisse zu verbessern. Wenn es möglich ist, sollte das neuronale Netz möglichst flach sein, was bedeutet, dass es möglichst wenige Neuronenschichten besitzt. Der Grund dafür ist, dass neuronale Netze mit vielen Neuronenschichten zwar mehr Wissen „besitzen“ können als neuronale Netze mit weniger Schichten, allerdings brauchen sie dadurch auch deutlich länger zum Verarbeiten eines Inputs. Für den Einsatz im autonomen Fahren würde dies bedeuten, dass durch die längere Verarbeitungszeit weniger Bilder pro Sekunde verarbeitet und damit Gefahren erst zu spät erkannt werden können. Somit kann ein Deep Neural Network, ein neuronales Netz mit vielen Neuronenschichten, hier entscheidende Nachteile gegenüber einem flachen neuronalen Netz haben. Das letztendliche Ziel des Versuches ist es ein flaches Convolutional Neural Network zu entwickeln, welches Verkehrsschilder annähernd so gut - oder im Optimalfall auch besser - klassifizieren kann als ein Mensch.

Hierbei muss allerdings zwischen einem neuronalen Netz zur Bilderkennung und zur Bildklassifikation unterschieden werden. Die tatsächliche Verarbeitung eines Verkehrsschildes besteht aus zwei verschiedenen Schritten. Zuerst muss erkannt werden, ob sich ein Verkehrsschild im Bild befindet und an welcher Stelle. Erst danach wird der erkannte Teil des Verkehrsschildes genommen und klassifiziert. Diesen Vorgang kann man in Abbildung 1 erkennen:

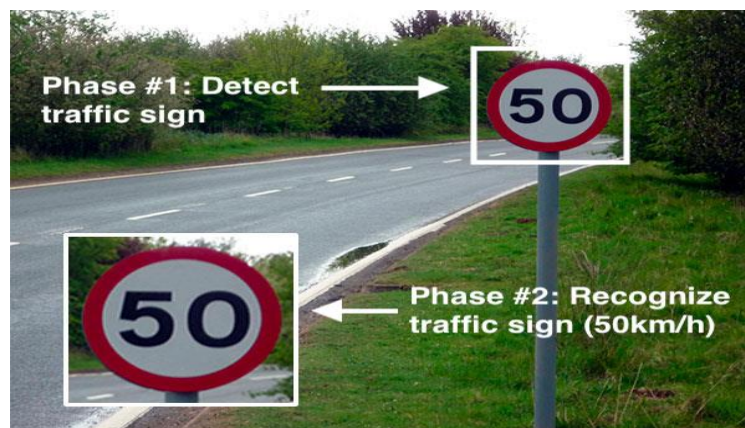


Abbildung 1: Bilderkennung und Klassifizierung

Das in diesem Versuch entwickelte neuronale Netz dient ausschließlich zur Bildklassifikation. Es ist also grundsätzlich noch nicht in der Praxis einsetzbar, da es keinen Teil zur Verkehrsschilderkennung (Siehe Phase #1 in Abbildung 1) besitzt. Ich habe mich auf die möglichst genaue Optimierung des neuronalen Netzes und bestmögliche Testergebnisse konzentriert und deshalb den Teil zur Bilderkennung vernachlässigt.

3. Der Datensatz

Für das Trainieren des neuronalen Netzes wurde einer der weltweit größten frei zugänglichen Datensätze für Verkehrsschilder genutzt, der German Traffic Sign Recognition Benchmark Datensatz oder auch kurz GTSRB-Datensatz.¹ Dieser wurde vom Institut für Neuroinformatik der Ruhr-Universität Bochum erstellt. Er enthält 51839 Bilder aus 43 verschiedenen Verkehrsschildklassen. Eine Bildklasse bzw. Verkehrsschildklasse repräsentiert hierbei Verkehrsschilder eines Typs, zum Beispiel alle Verkehrsschilder der Art „Zulässige Höchstgeschwindigkeit 50 km/h“. Unterteilt wird dabei in einen Trainingsdatensatz mit 39209 Bildern und einen Testdatensatz mit 12630 Bildern. Sowohl im Trainings- als auch im Testdatensatz sind alle Bilder in unterschiedlichen Bildgrößen vorhanden. Dabei reicht die Größe der Bilder von 15*15 Pixel bis 250*250 Pixel. Allerdings sind dabei nicht alle Bildklassen gleich oft im Datensatz vorhanden. Die Verteilung der Anzahl der Bilder in den unterschiedlichen Bildklassen kann man in folgender Grafik sehen (Siehe Abbildung 2):

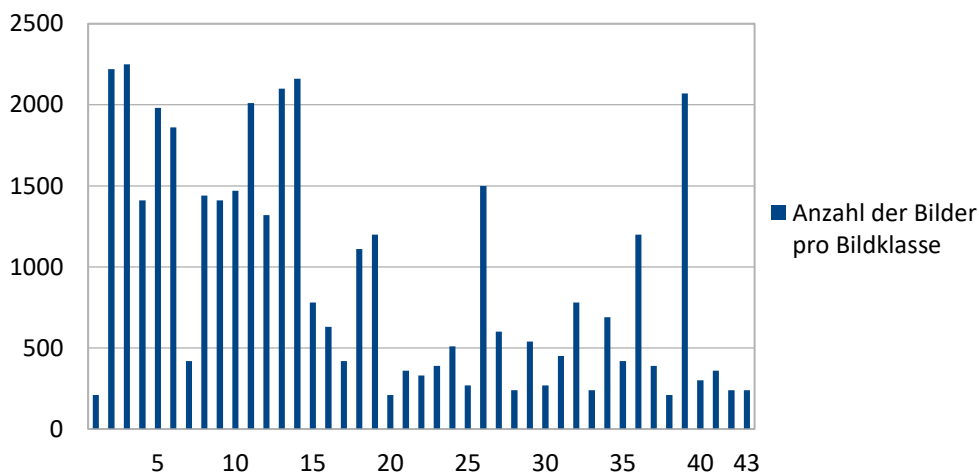


Abbildung 2: Verteilung der Bilder im Trainingsdatensatz

Dabei lässt sich gut erkennen, dass die Anzahl der Bilder sehr stark variiert. So besitzt beispielsweise Bildklasse 1, welche das 20 km/h Tempolimit Verkehrsschild darstellt, gerade einmal 210 Trainingsbilder, hingegen Bildklasse 3, welches das 50 km/h Tempolimit Verkehrsschild darstellt, 2250 Trainingsbilder. In der Regel kann man sagen, je öfter das Verkehrsschild im Straßenverkehr vorkommt, desto mehr Bilder sind im Trainingsdatensatz dazu vorhanden.

Die Verteilung der Bilder im Testdatensatz ist der Verteilung im Trainingsdatensatz sehr ähnlich. Auch hier besitzt die Bildklasse 1 die wenigsten Bilder und Bildklasse 3 die meisten Bilder. Allerdings unterscheidet sich die Anzahl der Bilder pro Bildklasse durch die geringere Gesamtanzahl an Bildern vom Trainingsdatensatz. Die Verteilung kann man in folgender Abbildung sehen (Siehe Abbildung 3):

¹<http://benchmark.ini.rub.de/?section=gtsrb&subsection=news>

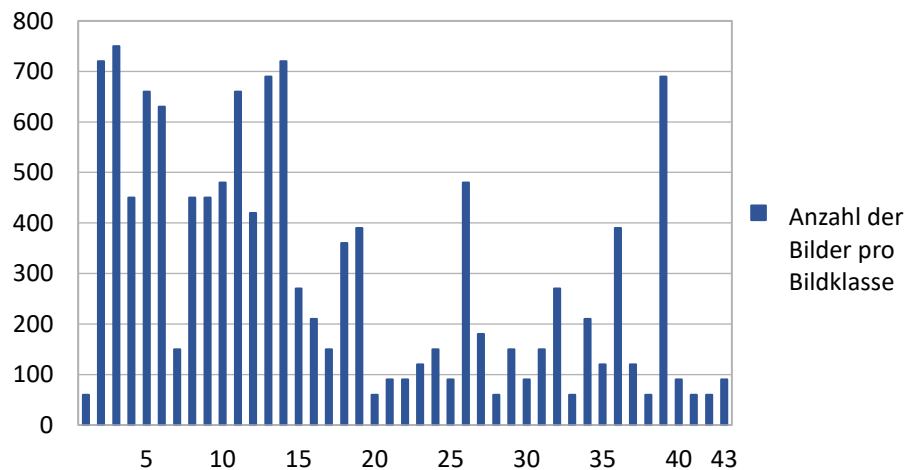


Abbildung 3: Verteilung der Bilder im Testdatensatz

Der wesentliche Unterschied zwischen Trainingsdatensatz und Testdatensatz liegt in der Qualität der Bilder. Im Trainingsdatensatz sind hauptsächlich scharfe und gut beleuchtete Bilder enthalten. Der Testdatensatz enthält deutlich unschärfere, schlecht ausgeleuchtete Bilder mit eher geringem Kontrast. Deshalb sind Bilder des Testdatensatzes sowohl für Mensch als auch CNN schwerer erkennbar. Den Unterschied kann man in zwei Beispielbildern derselben Bildklasse erkennen, wobei Abbildung 4 ein Bild aus dem Trainings- und Abbildung 5 ein Bild aus dem Testdatensatz ist:



Abbildung 4: Bild eines Verkehrsschildes aus dem Trainingsdatensatz (Bild 00005_00017)



Abbildung 5: Bild eines Verkehrsschildes aus dem Testdatensatz (Bild 00042)

4. Aufbau des Convolutional Neural Network

Für den Versuch wurde ein einfaches Convolutional Neural Network genutzt, welches aus mehreren Filtern und darauffolgenden Neuronenschichten besteht. Am Eingang befinden sich zwei Convolution Filter der Kernelgröße 3×3 mit 32 Channels und einem Stride von (1,1). Die Kernelgröße gibt die Größe des Convolution Filters an. Eine Kernelgröße von 3×3 bedeutet, dass der genutzte Filter 3×3 Einheiten groß ist. Je größer die Kernelgröße, desto stärker wird die Bildmatrix reduziert. Der Stride gibt an, wie viele Elemente in eine Richtung nach einem berechneten Element übersprungen werden. Ein höherer Stride bewirkt eine stärkere Bildreduktion. So bewirkt

ein Stride von (2,2) eine vier Mal so hohe Bildreduktion wie ein Stride von (1,1).

Beide Convolution Filter besitzen als Aktivierungsfunktion ReLU². Darauf folgt ein Pooling Filter mit einem Pool von 2*2 Pixel. Auf diesen folgt erneut ein Convolution Filter mit einer Kernelgröße von 3*3 Pixel und 64 Channels mit der Aktivierungsfunktion ReLU und ein Pooling Filter mit einem Pool von 2*2 Pixel.

Auf die Filter folgt ein Input-Layer mit 750 Neuronen, darauf ein Hidden-Layer mit 256 Neuronen und der Output-Layer mit 43 Neuronen. Die Anzahl der Neuronen im Output-Layer entspricht dabei der Anzahl der darzustellenden Bildklassen. Der Input-Layer und der Hidden-Layer besitzen als Aktivierungsfunktion ebenfalls ReLU und der Output-Layer Softmax. Die Softmax-Funktion bewirkt, dass die eingehenden Signale in die letzte Neuronenschicht in Klassenwahrscheinlichkeiten umgeformt werden. Die Bildklasse mit der höchsten Klassenwahrscheinlichkeit wird als das Ergebnis der Klassifikation gesehen.

Um Overfitting während des Trainings zu vermeiden, wurde auf den letzten Pooling Filter ein Dropout³ von 20%, auf den Input-Layer ein Dropout von 40% und auf den Hidden-Layer ein Dropout von 60% gelegt. Der starke Dropout bewirkt zwar, dass das Training deutlich länger dauert als ohne Dropout, allerdings ist das neuronale Netz gut gegen Overfitting geschützt. Overfitting bedeutet, dass sich der Fehler im Training mit dem Datensatz zwar verkleinert, allerdings der Fehler zu einem zugehörigen Testdatensatz steigt. Ursache dafür ist, dass das Netz sich zunehmend auf den Trainingsdatensatz spezialisiert und sich beispielsweise statt auf die Bildklassen, auf dessen Einzelbilder optimiert.

Der Aufbau des Netzes ist in dieser Abbildung noch einmal skizzenhaft dargestellt (Siehe Abbildung 6):

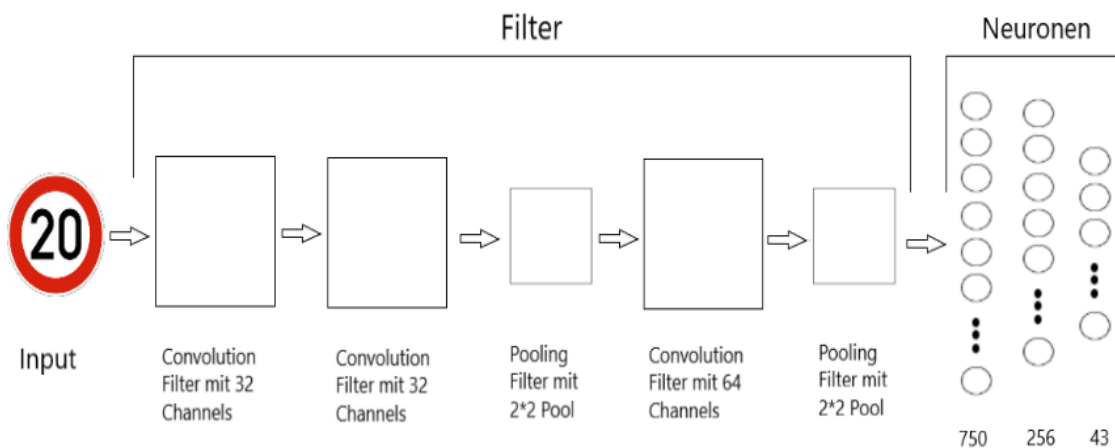


Abbildung 6: Aufbau des CNN im Versuch

Für alle Versuche wurde die Batch Size 32 gewählt. Das bedeutet, dass im Training aller 32 Bilder die Gewichtungen der Neuronen geändert werden und für diese 32 Bilder der Durchschnittswert als Verlust genutzt wird. Als Verlustfunktion wurde die Sparse Categorical Crossentropy⁴, im Deutschen auch standardmäßige

² <https://arxiv.org/pdf/1803.08375.pdf>

³ <http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf>

⁴ http://web.mit.edu/6.454/www/www_fall_2003/gew/CETutorial.pdf

kategoriale Kreuzentropie genannt, genutzt. Der Output des Netzes spiegelt dabei die Nummer der jeweiligen Bildklasse wider. Diese Verlustfunktion ist auf diese Art des Outputs spezialisiert. Diese Art der Outputvorgabe für das Training nennt man auch Label-Encoding, da für jedes Trainingsbild ein Label, also die richtige Bildklasse vorgegeben wird. Somit ist das Ergebnis der Klassifikation eines Bildes immer eine ganze Zahl von 0 bis 42, wobei jede Zahl für eine Bildklasse steht.

Für das Training wurde außerdem ein Shuffle verwendet, durch den alle Trainingsbilder in einer zufälligen Reihenfolge trainiert werden. Dieser Shuffle ist für den genutzten Datensatz dringend erforderlich, da alle Trainingsbilder den Bildklassen nach zur Liste der Trainingsbilder hinzugefügt werden. Ohne Shuffle würde das Netz zuerst alle Bilder der Bildklasse 1 lernen, anschließend alle Bilder der Bildklasse 2 usw. bis zur Bildklasse 43. Dadurch würde fast kein Trainingseffekt entstehen, da die Gewichtungen vollkommen auf die jeweilige Bildklasse ausgerichtet werden und das Netz die Gewichtungen aus dem Training der anderen Bildklassen fast vollständig annullieren und somit das meiste Gelernte vergessen würde. Mit einer zufälligen Bildreihenfolge hingegen lernt das Netz alle Bildklassen fast gleichzeitig und ändert die Gewichtungen so, dass alle 43 Bildklassen bestmöglich klassifiziert werden können. Somit ist der Shuffle hier dringend notwendig, um gute Trainingsergebnisse zu erzielen. Allerdings hat der Shuffle den Nebeneffekt, dass die Trainingsergebnisse in mehreren Versuchen stark variieren können, da diese von der Reihenfolge der Trainingsbilder beeinflusst werden.

5. Durchführung

Das Convolutional Neural Network wurde mithilfe der Bibliotheken Tensorflow und Keras in der Programmiersprache Python umgesetzt. Diese wurden von Google für das Erstellen und Trainieren von KI aus dem Bereich des maschinellen Lernens erstellt und öffentlich zugänglich gemacht. Sie bilden heute die Standardbibliotheken für das maschinelle Lernen.

Bevor das Training des CNNs beginnen kann, müssen noch einige Änderungen am Trainingsdatensatz vorgenommen werden. Dem Datensatz wird bereits eine Struktur mitgegeben, dass 43 Ordner mit den Trainingsbildern angelegt sind. In jedem der 43 Ordner befinden sich die Trainingsbilder für eine Bildklasse. Allerdings befindet sich zu den Bildern auch noch jeweils eine CSV-Datei pro Klasse mit in den Ordnern. Da diese für das Training unwichtig ist und beim Einlesen der Bilder stören würde, muss sie zuerst gelöscht werden. Weiterhin müssen die Namen der Ordner für die Bildklassen geändert werden, da diese von Grund auf zwei Nullen vor dem eigentlichen Klassennamen besitzen. Die Dateinamen sollen als Index der Bildklasse verwendet werden. Bei der Wandlung der Dateinamen in einen Integer-Wert dürfen keine führenden Nullen im Namen enthalten sein, deshalb müssen diese vorher entfernt werden.

Die genannten Änderungen können von folgendem Programm in Python übernommen werden (Siehe Abbildung 7):


```

import os
pfad="GTSRB/Final_Training/Images/"
for ordner in os.listdir(pfad):
    ordnername = int(ordner)
    ordnername = str(ordnername)
    if (ordnername != ordner):
        os.rename(pfad + ordner, pfad + ordnername)
for i in range(0,43):
    n = str(i)
    subpfad = os.path.join(pfad, n)
    for datei in os.listdir(subpfad):
        if not datei.endswith('.csv'):
            continue
        datei_mit_pfad = os.path.join(subpfad, datei)
        os.remove(datei_mit_pfad)

```

Abbildung 7: Programm zur Vorbereitung des Datensatzes

Weiterhin besitzen alle Trainings- und Testbilder bis zum jetzigen Zeitpunkt unterschiedliche Bildgrößen. Damit ein neuronales Netz mit einem Fully-Connected Layer aus Bildern lernen kann, müssen diese alle die gleiche Größe besitzen, damit ein einheitlicher Input vorliegt. Um dies zu ermöglichen, werden bei unserem neuronalen Netz alle Bilder beim Laden auf die Bildgröße von 32*32 Pixel umgewandelt.

Für das Training fehlt nun noch ein wichtiger Parameter, der Optimizer. Die Aufgabe des Optimizers ist es, die Gewichtungen des Netzes zu ändern. Dazu orientiert er sich an der Verlustfunktion und versucht den Verlust so gering wie möglich zu halten. Eine wichtige Rolle spielt hierbei auch die Lernrate. Diese gibt an, wie stark die Gewichtungen vom Optimizer verändert werden. Wenn die Lernrate zu hoch ist, werden die Gewichtungen zu stark geändert und das Netz „vergisst“ die bereits gelernten Bilder einfach wieder. Wenn die Lernrate hingegen zu niedrig ist, ändert das Netz die Gewichtungen zu wenig, wodurch der Lerneffekt zu gering wird.

Um für das bereits vorgestellte CNN den besten Optimizer zu finden, wurden drei verschiedene Optimizer in jeweils fünf Durchläufen miteinander verglichen.

Die folgende Grafik zeigt die Medianwerte für drei Durchläufe der verschiedenen Optimizer in 60 Trainingsepochen für die Genauigkeit im Testdatensatz:

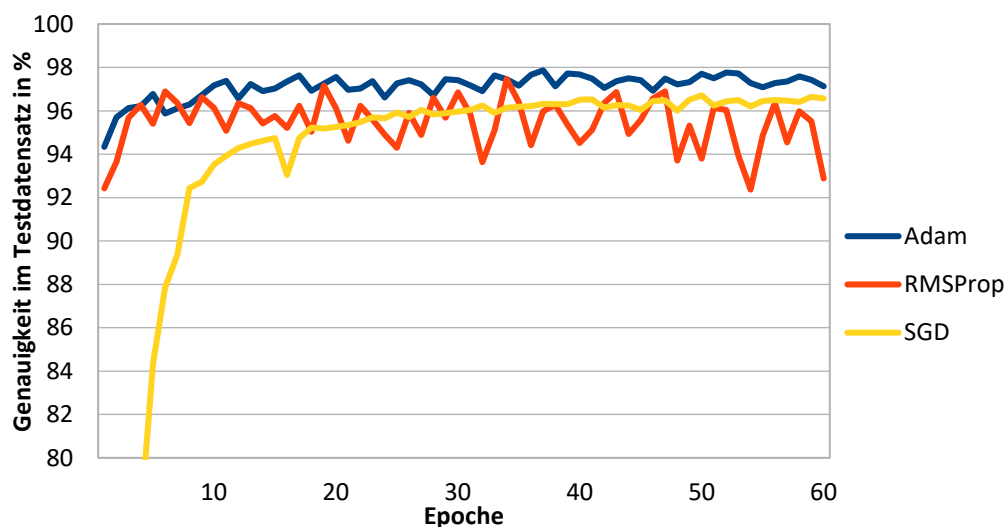


Abbildung 8: Training mit verschiedenen Optimizern

In der Grafik lässt sich erkennen, dass der Optimizer Adam⁵ am besten für das Training mit diesem Netz geeignet ist und die durchschnittlich beste Genauigkeit unter gleichen Bedingungen im Vergleich zu den anderen Optimizern erreicht. Adam hat die Besonderheit, dass er eine adaptive Lernrate besitzt und diese somit für das Training nicht vorgegeben werden muss. Dadurch müssen keine Versuche zu einer passenden Lernrate durchgeführt werden.

Somit ist das CNN nun theoretisch trainierbar, allerdings können noch einige Veränderungen zur Verbesserung des Trainings vorgenommen werden. Wie bereits in [Punkt 3](#) erwähnt, besitzt der GTSRB-Datensatz sehr unterschiedlich verteilte Bildzahlen pro Klasse im Trainingsdatensatz (Siehe Abbildung 2). Das hat zur Folge, dass einige der Bildklassen weniger stark trainiert werden als andere. Eine Möglichkeit dem etwas entgegen zu wirken ist, die Trainingsbilder aus Bildklassen mit wenigen Bildern einfach doppelt in die Liste der Trainingsbilder hinzuzufügen. Somit werden diese pro Epoche zwei Mal und somit stärker trainiert. Die Verteilung mit den doppelt hinzugefügten Trainingsbildern kann man in folgender Grafik sehen:

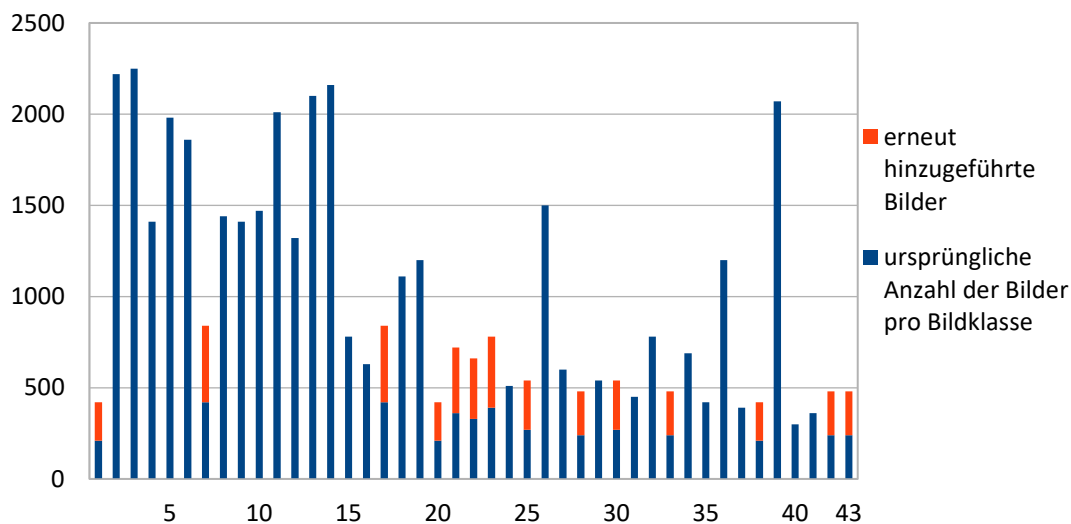


Abbildung 9: Verteilung der Bilder im Trainingsdatensatz mit Bildausgleich

Eine weitere Methode der Verbesserung der Genauigkeit ist das Hinzufügen einer Batchnormalisierung auf das gesamte Netz. Diese hat zur Folge, dass die Eingaben für jeden Layer und Filter normalisiert werden und es dadurch zu einer geringeren internen Kovariatenverschiebung⁶ kommt. Dadurch werden die Eingaben weniger stark gestreut und das Netz kann diese besser verarbeiten. In den meisten Fällen kann eine Batchnormalisierung helfen, um die Resultate etwas zu verbessern, allerdings hat sie den Nebeneffekt, dass das Training durch die Normalisierung der Eingabedaten deutlich länger dauert. Um den Effekt der Batchnormalisierung auf das Netz festzustellen, wurden fünf Durchläufe über 60 Epochen durchgeführt, einmal mit und einmal ohne Batchnormalisierung.

Die Ergebnisse kann man in folgender Grafik sehen (Siehe Abbildung 10):

⁵<https://arxiv.org/pdf/1412.6980.pdf>

⁶<https://cs.nyu.edu/~roweis/papers/invar-chapter.pdf>

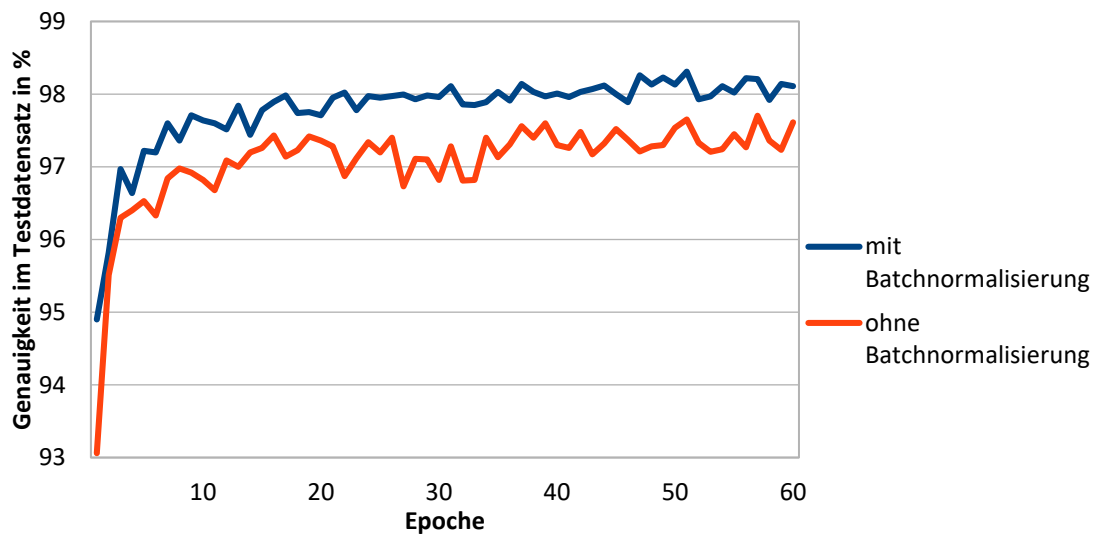


Abbildung 10: Effekt der Batchnormalisierung auf die Genauigkeit

In der Grafik lässt sich deutlich erkennen, dass das neuronale Netz mit Batchnormalisierung durchgängig eine höhere Genauigkeit in seinen Vorhersagen besitzt. Im Bereich von Epoche 50 bis Epoche 60 besitzt das neuronale Netz mit Batchnormalisierung beispielsweise im Durchschnitt eine 0,748 % höhere Genauigkeit als das neuronale Netz ohne diese. Somit bewirkt sie eine deutliche Steigerung der Genauigkeit im Testdatensatz. Um zu ermitteln, wie gut das gesamte neuronale Netz mit Ausgleich der Trainingsbilder und Batchnormalisierung ist, wurde das Netz in 80 Epochen auf die Genauigkeit im Testdatensatz getestet. Hierbei wurden 10 Durchläufe getätigt, um ein tatsächliches Bild von der Genauigkeit pro Epoche zu bekommen, da Einzelversuche durch den Shuffle immer ein Stück weit zufällig sind und erst ab 10 oder mehr Durchläufen eine wirkliche Aussagekraft besteht. Weiterhin wurde in diesem Versuch auch der Verlust pro Epoche mit betrachtet. Der Verlust ist der Fehler der Vorhersage im Vergleich zum richtigen Ergebnis und wird durch die Verlustfunktion berechnet. Je kleiner der Verlust, desto höher ist die Genauigkeit der Vorhersage.

Ausgehend von den bisher gesammelten Erkenntnissen kann man abschätzen, dass sich das neuronale Netz sehr ähnlich zu den vorherigen Versuchen (Siehe Abbildung 8, Abbildung 10) verhalten wird.

Das bedeutet, dass im Bereich von Epoche 1 bis Epoche 60 die Genauigkeit logarithmisch steigen wird. Da eine steigende Genauigkeit meist mit einem geringeren Verlust einher geht, wird dieser vermutlich exponentiell fallen. Im Bereich von 60 bis 80 Epochen Training wird die Genauigkeit wahrscheinlich nur noch sehr gering steigen oder ganz stagnieren, da mit einer hohen Epochenzahl die Gefahr des Overfittings immer größer wird. Im schlechtesten Fall fällt nach 60 Epochen bereits die Genauigkeit und der Verlust steigt wieder, da sich das Netz nur noch auf den Trainingsdatensatz optimiert und somit Bilder aus dem Testdatensatz schlechter erkennt. Bei dem Versuch konnten letztendlich folgende Ergebnisse festgestellt werden (Siehe Abbildung 11):

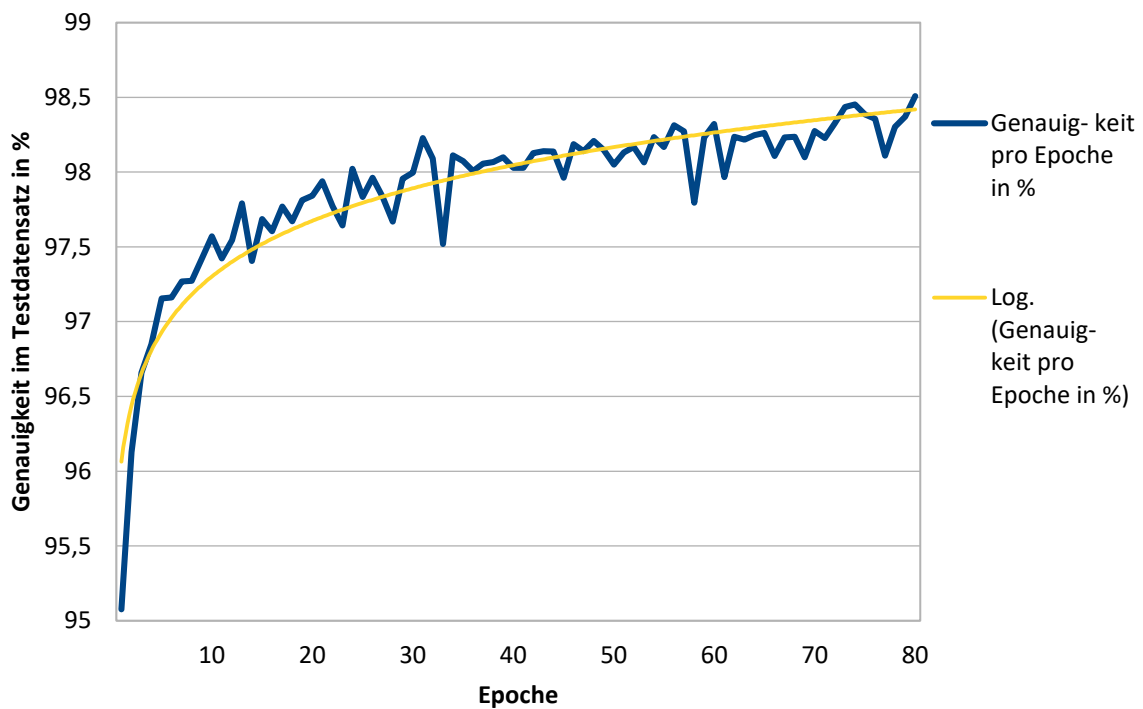


Abbildung 11: Genauigkeit im Verlauf des Trainings

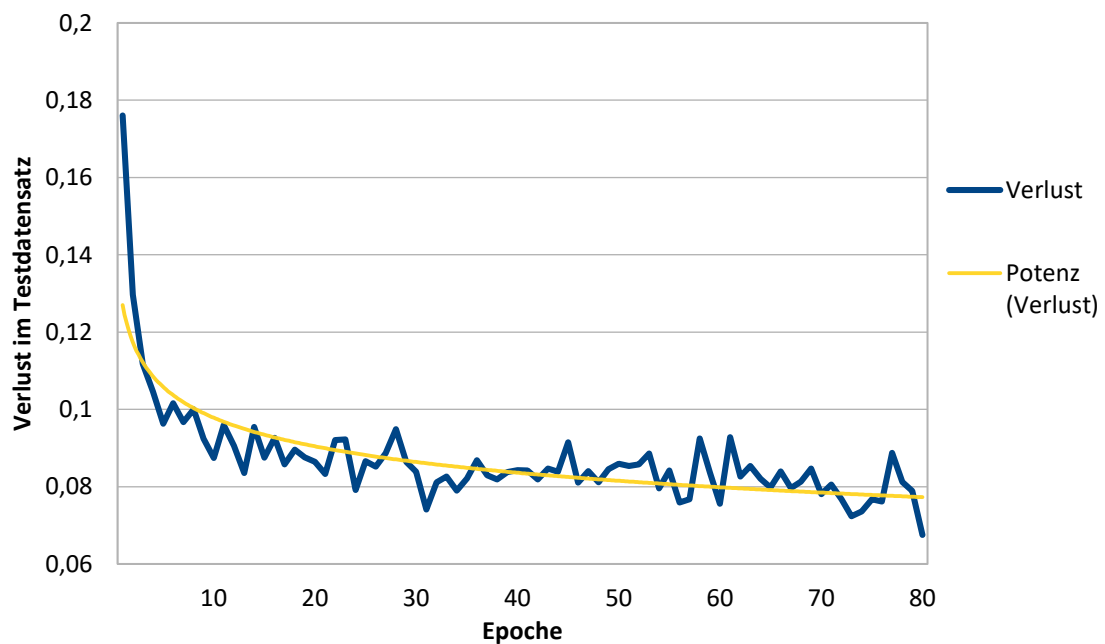


Abbildung 12: Verlust im Verlauf des Trainings

6. Auswertung

In [Abbildung 11](#) kann man die Genauigkeit im Testdatensatz pro Epoche als blaue Linie sehen. Die gelbe Linie stellt dabei eine logarithmische Trendlinie dar. Man kann deutlich erkennen, dass in den ersten Epochen die Genauigkeit sehr stark ansteigt und dann der Anstieg zunehmend schwächer wird. Ungefähr ab Epoche 30 bleibt

der Anstieg ungefähr gleich stark und ändert sich bis zur Epoche 80 nur noch sehr wenig. Damit ist der Teil meiner These, dass die Genauigkeit ab Epoche 60 aufgrund von Overfitting nicht mehr steigt, widerlegt. Somit macht sich im Bereich bis 80 Epochen noch kein Overfitting bemerkbar, was sich durch den hohen Dropout in den ersten zwei Neuronenschichten erklären lässt.

In der Abbildung kann man auch sehen, dass es einige starke Abweichungen zwischen Trendlinie und dem Graphen gibt. Im Bereich bis Epoche 30 liegen die Werte meist über der Trendlinie, bis es in Epoche 31 erst einen starken Ausschlag nach oben und kurz darauf einen starken Ausschlag nach unten in Epoche 33 gibt. Ab diesem Ausschlag liegen die Werte dann meist unter der Trendlinie. Somit ist der tatsächliche Anstieg der Genauigkeit ungefähr bis Epoche 30 höher und ab dann etwas weniger stark, als es die Trendlinie suggeriert. Um die mögliche Ursache für die starken Ausschläge bei Epoche 31 und 33 zu finden, müssen die Daten etwas genauer betrachtet werden.

Hierfür reichen allerdings nicht die Durchschnittswerte, wie sie in [Abbildung 11](#) und [Abbildung 12](#) genutzt wurden. Zur weiteren Analyse wurden die Maximalwerte, sowie die Minimal- und Medianwerte in der jeweiligen Epoche aus den 10 Durchläufen in folgende Grafik eingearbeitet:

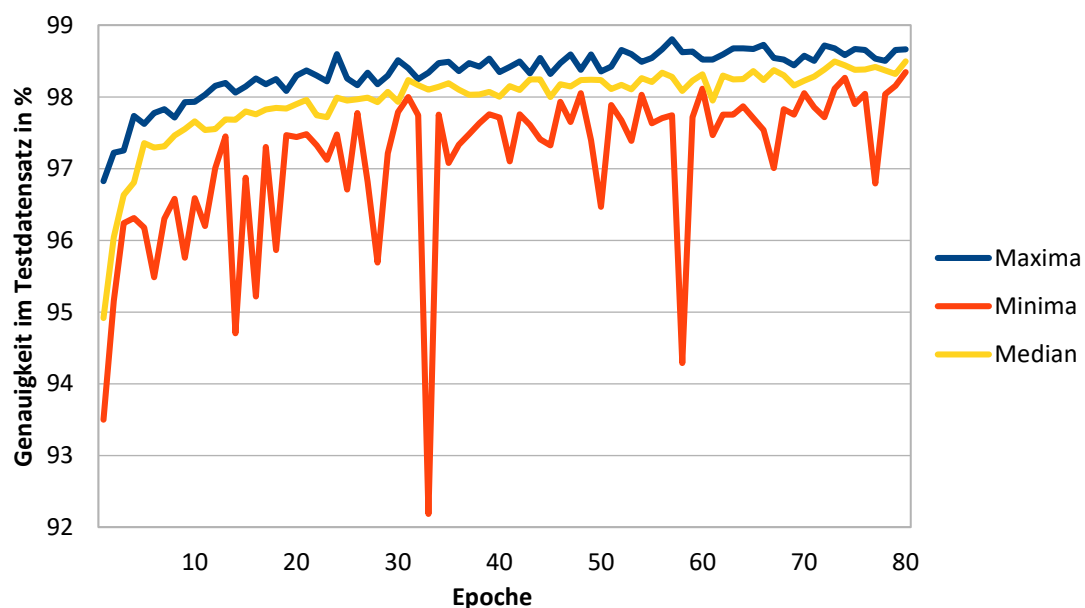


Abbildung 13: Minima-, Maxima- und Medianwerte im Verlauf des Trainings

In Abbildung 13 sind die Maximalwerte in Blau, die Minimalwerte in Rot und die Medianwerte pro Epoche in Gelb dargestellt. Man kann erkennen, dass diese Werte mit zunehmender Epochenzahl steigen. Dabei ähnelt der Verlauf der Maximalwerte annähernd dem Verlauf der Medianwerte. Der Verlauf der Minimalwerte ist hingegen sehr ungleichmäßig. Besonders fallen hierbei die Minimalwerte von Epoche 33 und Epoche 58 auf. Beide heben sich stark negativ von den anderen Minimalwerten ab. Da in Abbildung 11 der Durchschnittswert der 10 Durchläufe genutzt wurde, beeinflussen diese zwei niedrigen Minimalwerte maßgeblich den Durchschnittswert und verursachen die Ausschläge in [Abbildung 11](#). Allerdings bedeuten diese niedrigen Minimalwerte nicht, dass das neuronale Netz in Epoche 33 und in Epoche 58 immer schlecht trainiert. Dies kann man an den Medianwert dieser beiden Epochen sehen, die keinen Ausschlag im Vergleich zu den anderen Medianwerten liefern. Somit entspricht die tatsächliche Entwicklung der Genauigkeit pro Epoche annähernd der der Trendlinie. Ausgehend

davon kann man vermuten, dass sich die Werte aus [Abbildung 11](#) bei einer höheren Zahl der Durchläufe immer mehr den Medianwerten und somit auch der Trendlinie annähern werden.

Eine weitere Auffälligkeit in [Abbildung 13](#) kann man in der Differenz zwischen Medianwerten, Minimalwerten und Maximalwerten erkennen. Während in den ersten Epochen die Differenz zwischen Minimalwerten und Maximalwerten noch sehr groß ist, wird diese mit zunehmender Epochenzahl immer kleiner. Dieser Effekt lässt sich mit dem Trainingseffekt eines neuronalen Netzes erklären. Zu Beginn des Trainings werden durch den noch großen Fehler im Training die Gewichtungen des Netzes stark verändert, sodass die Reihenfolge der Trainingsbilder, die vom Shuffle bestimmt wird, einen starken Einfluss besitzt. Deshalb ist das Netz hier sehr anfällig, sowohl für die Maximalwerte als auch die Minimalwerte. Mit zunehmender Epochenzahl besitzt das Netz immer mehr „Vorwissen“ und die Gewichtungen werden weniger stark verändert. Deshalb sind die Effekte einer sehr schlechten oder besonders guten Trainingsepoche auf die Genauigkeit hier weniger stark als zu Beginn des Trainings. Den Effekt kann man ebenfalls an den Maximalwerten erkennen.

Ab Epoche 57 steigen die Maximalwerte nicht mehr, sondern nähern sich immer mehr dem Median an und fallen somit leicht. Deshalb bildet der Höchstwert aus Epoche 57 den absoluten Maximalwert aller Epochen, obwohl das neuronale Netz in den nachfolgenden Epochen eine höhere Genauigkeit in den Medianwerten besitzt. Das neuronale Netz besitzt an diesem Punkt eine Genauigkeit von 98,8 % im Testdatensatz, was bereits eine sehr hohe Genauigkeit darstellt. Die menschliche Fehlerrate in diesem Testdatensatz liegt im Vergleich dazu bei 98,84 %⁷. Somit ist das trainierte neuronale Netz nur um 0,04 % schlechter als der Mensch, also annähernd gleich stark. Hierbei ist allerdings auch von Interesse, in welcher Bildklasse dieses trainierte neuronale Netz die meisten Bilder falsch erkannt hat.

Die Anzahl der falsch klassifizierten Bilder pro Bildklasse kann man in folgender Grafik sehen:

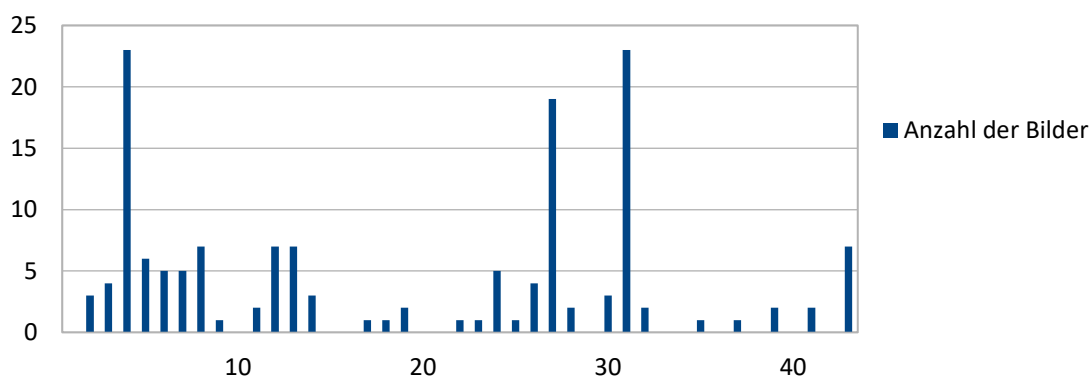


Abbildung 14: Falsch klassifizierte Bilder

Dabei kann man erkennen, dass das trainierte neuronale Netz besonders viele Bilder in Bildklasse 4, Bildklasse 27 und Bildklasse 31 falsch klassifiziert hat (Siehe [Abbildung 14](#)). Bildklasse 4 entspricht dabei dem 60 km/h Tempolimit-Schild, Bildklasse 27 dem Warnschild für Ampeln und Bildklasse 31 dem Warnschild für Schnee- und Eisglätte.

Im Folgenden sollen die Fehler in der Bildklasse 4 näher analysiert werden.

Von den 23 falsch klassifizierten Bildern wurden 21 als Bildklasse 6, eins als Bildklasse 14 und eins als Bildklasse 26 erkannt. Dabei stellt die Bildklasse 6 das 80 km/h Tempolimit-Schild dar. Das trainierte Netz

⁷ <https://christian-igel.github.io/paper/MvCBMLaFTSR.pdf>

verwechselt also noch häufig das 60 km/h Tempolimit-Schild mit dem 80 km/h Tempolimit-Schild.

In der Hoffnung, dass eine noch höhere Genauigkeit erreicht werden kann, wurde dieses bereits vortrainierte Netz noch einmal ca. 200 Epochen mit einer höheren Batch Size, 64, trainiert. Dazu wurde auch ein anderer Optimizer, Adamax⁸, genutzt, da sich dieser besser für filigranes Training eignet als Adam. Allerdings wurden bei dem Training die Trainingsbilder der Bildklasse 4 und 6 dreifach in die Liste der Trainingsbilder hinzugefügt, damit das Netz diese noch einmal intensiv trainiert und möglicherweise deshalb weniger Fehler macht. Dabei erreichte das trainierte Netz nun eine Genauigkeit von 99,105 %. Somit konnte eine Steigerung in der Genauigkeit von rund 0,3 % erzielt werden.

Nun interessiert wieder die Verteilung der falsch klassifizierten Bilder:

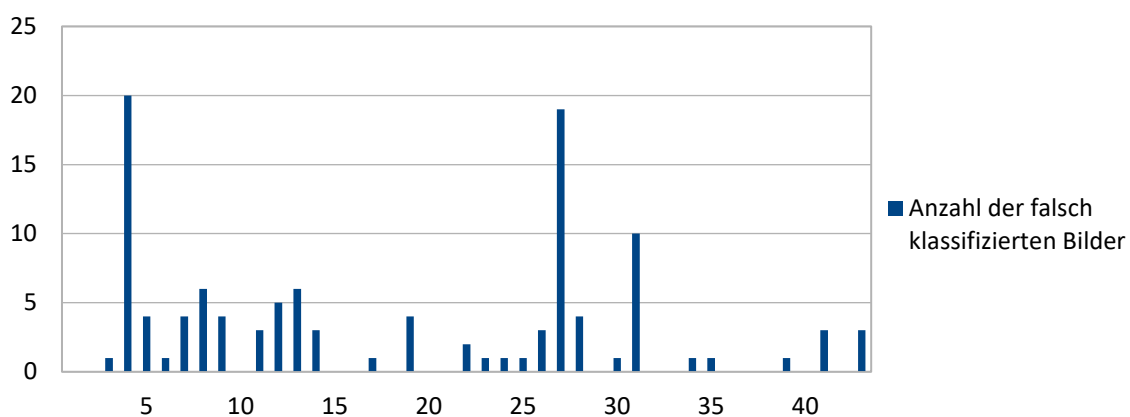


Abbildung 15: Falsch klassifizierte Bilder nach weiterem Training

Wie man in Abbildung 15 erkennen kann, hat sich die Verteilung der Fehler kaum verändert, nur die falsch klassifizierten Bilder in Bildklasse 31 hat sich stark reduziert. In Bildklasse 4 wurden nun 18 Bilder als Bildklasse 6 und jeweils ein Bild als Bildklasse 10 und 19 klassifiziert. Somit hat sich das trainierte neuronale Netz etwas verbessert, allerdings besteht das Problem mit dem Verwechseln von Bildklasse 4 und 6 immer noch, wenn auch nicht mehr so stark. Folglich hat das dreifache Hinzufügen der Trainingsbilder dieses Problem nicht beheben können. Dennoch muss man relativieren, dass das Netz von den 450 Testbildern in Bildklasse 4 430 Bilder richtig klassifiziert hat, also eine Genauigkeit von rund 96 % in dieser Bildklasse erreicht hat. Durch weiteres Training konnten in einem angemessenen Zeitrahmen keine besseren Ergebnisse erzielt werden, weshalb das Ziel einer weiteren Optimierung nicht weiterverfolgt wurde.

Zum abschließenden Test wurde dieses trainierte neuronale Netz auf einige selbst fotografierte Bilder getestet. Hierbei wurden Verkehrsschilder ausgewählt, die durch Beklebung, Farbe oder besondere Form aufgefallen sind. Die Vorhersage der Bildklasse des trainierten Netzes sowie deren Klassenwahrscheinlichkeit wurden in der jeweiligen Bildbeschreibung notiert:

⁸ <https://www.paperswithcode.com/method/adamax>



Abbildung 16: Aufgenommenes Verkehrsschild

Vorhersage: Bildklasse 14
Wahrscheinlichkeit: 100 %
tatsächliche Bildklasse: 14



Abbildung 17: Aufgenommenes Verkehrsschild

Vorhersage: Bildklasse 16
Wahrscheinlichkeit: 99,88 %
tatsächliche Bildklasse: 16



Abbildung 18: Aufgenommenes Verkehrsschild

Vorhersage: Bildklasse 16
Wahrscheinlichkeit: 99 %
tatsächliche Bildklasse: 16



Abbildung 19: Aufgenommenes Verkehrsschild

Vorhersage: Bildklasse 1
Wahrscheinlichkeit: 100 %
tatsächliche Bildklasse: 1

Hierzu kann abschließend gesagt werden, dass das trainierte neuronale Netz alle Bilder richtig klassifiziert hat, wobei es mit einem Verkehrsschild mit ausgeblancher Farbe (Siehe Abbildung 17) und einem Verkehrsschild mit einem anderen, runden Schild (Siehe Abbildung 16) dahinter keinerlei Probleme hatte. Bei dem besprühten und dem beklebten Verkehrsschild hatte das Netz ebenfalls keine Probleme beim Klassifizieren, wie man an der Wahrscheinlichkeit sehen kann. Dieses Ergebnis ist durchaus überraschend, da sich im Datensatz keinerlei Bilder von Verkehrsschildern mit Klebern oder Besprühungen vorhanden sind. Dennoch war das trainierte Netz in der Lage die Verkehrsschilder fast fehlerfrei zu erkennen. Das könnte auch daran liegen, dass die Bilder stark runterskaliert werden mussten und somit beispielsweise die Kleber in Abbildung 18 nur als Pixel erkennbar sind. Vermutlich verdecken diese Pixel markante Punkte einer bestimmten Verkehrsschildklasse nicht, wodurch für das Netz kein Problem bei der Klassifikation hatte. Bei neuronalen Netzen ist allerdings nur schwer oder gar nicht nachvollziehbar, wodurch bestimmte Entscheidungen getroffen werden, sodass hier keine eindeutige Erklärung geliefert werden kann. Dennoch zeigt dieser Effekt, dass neuronale Netze auch in der Lage sind, abstraktere Verkehrsschilder zu erkennen und somit auch vermutlich im Straßenverkehr kein Problem mit der richtigen Klassifizierung dieser hätte.

7. Einordnung des Ergebnis

Zur Einordnung des Versuchsergebnis gibt es drei Referenzpunkte. Einerseits wurde auf der Website des Neuroinstituts für Neuroinformatik Bochum eine Liste mit den Ergebnissen der ersten Runde der IJCNN 2011 veröffentlicht, für die der Datensatz als Wettbewerb erstellt wurde. Des weiteren wird auf der Seite auch eine Ergebnisliste mit Resultaten außerhalb des Wettbewerbes veröffentlicht. Hier konnten auch Teilnehmer außerhalb des Wettbewerbs ihre Ergebnisse einreichen. Eine dritte Vergleichsquelle liefert die Website „Kaggle“. Auf dieser wurde 2018 ein internationaler Wettbewerb zum GTSRB-Datensatz gestartet. Hier kann größtenteils nicht nachvollzogen werden, wie die Teilnehmer ihre Ergebnisse erreichten und ob diese tatsächlich stimmen.

Die sechs besten Ergebnisse der ersten Runde der IJCNN 2011 kann man in folgender Abbildung sehen:

TEAM	METHOD	TOTAL	SUBSET
[197] IDSIA	cnn_hog3	98.98%	98.98%
[196] IDSIA	cnn_cnn_hog3	98.98%	98.98%
[178] sermanet	EBLearn 2LConvNet ms 108 feats	98.97%	98.97%
[195] IDSIA	cnn_cnn_hog3_haar	98.97%	98.97%
[187] sermanet	EBLearn 2LConvNet ms 108 + val	98.89%	98.89%
[199] INI-RTCV	Human performance	98.81%	98.81%

Abbildung 20: Ergebnisse der IJCNN 2011

Verglichen mit dem in diesem Versuch erreichten Ergebnis von rund 99,11 % sind die Ergebnisse der IJCNN 2011 deutlich niedriger. So schlägt das in diesem Versuch trainierte neuronale Netz das beste Ergebnis der IJCNN 2011 um fast 0,1 %, was ein verhältnismäßig großer Unterschied ist. Bei dieser Betrachtung muss man allerdings auch beachten, dass sich die Technologie von neuronalen Netzen und die Rechenleistung von Computern seit 2011 stark weiterentwickelt hat. Somit ist dieses Resultat nicht sonderlich überraschend.

Anders sieht dies bei den Ergebnissen außerhalb des IJCNN-Wettbewerbs aus. Bei diesen Ergebnissen kann man sich höchstwahrscheinlich sicher sein, dass diese auch tatsächlich stimmen, da zu jedem dieser Ergebnisse ein wissenschaftliches Paper veröffentlicht wurden.

Die folgende Abbildung zeigt die sechs besten Ergebnisse aus dieser Liste:

TEAM	METHOD	TOTAL	SUBSET
[156] DeepKnowledge Seville	CNN with 3 Spatial Transformers	99.71%	99.71%
[3] IDSIA ★	Committee of CNNs	99.46%	99.46%
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%	98.97%
[1] INI-RTCV ★	Human Performance	98.84%	98.84%
[4] sermanet ★	Multi-Scale CNNs	98.31%	98.31%
[2] CAOR ★	Random Forests	96.14%	96.14%

Abbildung 21: Ergebnisse von der GTSRB-Website

In dieser Liste kann man erkennen, dass in anderen Projekten deutlich bessere Ergebnisse im Vergleich zur IJCNN und zu diesem Versuch erreicht werden konnten. Hier zeichnet sich auch bereits ein deutlicher Unterschied zwischen dem Top-Ergebnis von 99,71 % und dem Ergebnis in diesem Versuch von 99,11 %. Dennoch stellt unser Versuch in dieser Liste das drittbeste Ergebnis dar. Anders sieht dies bei den Ergebnissen von dem Wettbewerb auf „Kaggle“ aus. Diese kann man in folgender Abbildung sehen, wobei auf der linken Seite Platzierung und Teamname und auf der rechten Seite die Genauigkeit des jeweiligen Teams steht:








25	Hans K		0.99176
26	SIYUAN		0.99129
27	Mudit Pandey		0.99129
28	Shuaiyi Bu		0.99081
29	Diwakar Paliwal		0.99081
30	Deep signs		0.99065
31	Ben Ahlbrand		0.99034

Abbildung 22: Ergebnisse des Kaggle-Wettbewerbs

In dem Wettbewerb von „Kaggle“ erreichte das beste Team eine Genauigkeit von 99,75 %. Das Ergebnis des hier durchgeführten Versuches würde dabei auf Platz 28 landen und somit besser als 57 der 84 teilgenommenen Teams sein.

Somit kann man hier zusammenfassend sagen, dass mit einer Genauigkeit von rund 99,11 % ein internationales Top-Ergebnis erreicht werden konnte.

8. Fazit

Abschließend kann man zu diesem Versuch sagen, dass das neuronale Netz die Aufgabe der Verkehrsschildklassifikation ohne größere Probleme bewältigen konnte. Dabei erreichte es eine Genauigkeit im Testdatensatz von 99,11 %, welche bereits rund 0,3 % besser war als die des Menschen. Dennoch können in dem Datensatz auch höhere Genauigkeiten bis 99,71 % oder höher erreicht werden⁹. Dies wäre beispielsweise durch den Einsatz von Spatial Transformern¹⁰ möglich. Trotzdem hatte das trainierte Netz vereinzelte Probleme, wie das Verwechseln von Bildklasse 4 mit Bildklasse 6, womit es für einen Einsatz in autonomen Fahrzeugen im Straßenverkehr noch eine zu hohe Ungenauigkeit hat und somit ein Einsatz verheerende Folgen haben könnte. Weiterhin sind im realen Straßenverkehr weit mehr als die 43 verschiedenen Verkehrsschilderklassen vorhanden. Demzufolge müsste ein tatsächlich für die Verkehrsschildklassifikation einsetzbares neuronales Netz mit einem deutlich größeren Bilddatensatz trainiert werden als dem GTSRB-Datensatz. Trotzdem könnten mit dem im Versuch trainierten neuronalen Netz die am häufigsten vertretenen Verkehrsschilder auf den deutschen Straßen richtig klassifiziert werden.

⁹https://benchmark.ini.rub.de/gtsrb_results.html#156ref

¹⁰<https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf>

Mit dem Versuch konnte gezeigt werden, dass es möglich ist, bereits mit einem einfachen Convolutional Neural Network bestimmte Aufgaben des Menschen im Straßenverkehr, in diesem Fall das Klassifizieren von Verkehrsschildern, zu übernehmen. Dabei kann die eingesetzte künstliche Intelligenz sogar den Menschen in seinen Fähigkeiten übertreffen und somit Unfälle vorbeugen. Dennoch kann eine KI immer nur das anwenden, was sie im Training gelernt hat, während der Mensch auch intuitiv handeln kann.

9. Literaturverzeichnis

Schwaiger, Roland; Steinwender, Joachim (2019): „Neuronale Netze programmieren“, 1. Aufl., Bonn

Tamm, Sofia (2019,30.05.): „Einführung in Neuronale Netze“. URL: <http://www.mi.uni-koeln.de/wp-znikolic/wp-content/uploads/2019/06/10-Thamm.pdf> [29.11.2020]

Saha, Sumit (2018,15.12.): „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way“. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [29.11.2020]

Wood, Thomas: „Convolutional Neural Network“. URL: <https://deeptai.org/machine-learning-glossary-and-terms/convolutional-neural-network> [29.11.2020]

Stallkamp, Johannes; Schlipsing, Marc; Salmen, Jan (2011, 03.11.): „The German Traffic Sign Recognition Benchmark: A multi-class classification competition“. URL: https://www.ini.rub.de/upload/file/1470692848_f03494010c16c36bab9e/StallkampEtAl_GTSRB_IJCNN2011.pdf [20.01.2021]

10. Ähnliche oder weiterführende Paper

Nguyen, Hoanh (2020,19.06.): „Fast Traffic Sign Detection Approach Based on Lightweight Network and Multilayer Proposal Network“. URL: <https://downloads.hindawi.com/journals/js/2020/8844348.pdf> [21.01.2021]

Wasif Arman Haque, Samin Arefin, A.S.M. Shihavuddin, Muhammad Abul Hasan (2020,09.12.): „DeepThin: A novel lightweight CNN architecture for traffic sign recognition without GPU requirements“. URL: <https://sci-hub.st/10.1016/j.eswa.2020.114481> [21.01.2021]

Alvaro Arcos-García, Juan A. Alvarez-García, Luis M. Soria-Morillo (2017,25.07.): „Deep Neural Network for Traffic Sign Recognition Systems: An analysis of Spatial Transformers and Stochastic Optimization Methods“. URL: <https://idus.us.es/bitstream/handle/11441/80679/NEUNET-D-17-00381.pdf;jsessionid=C17E132C16F755D73EAB35D11566DDB?sequence=1> [21.01.2021]

Dan Ciresan, Ueli Meier, Jürgen Schmidhuber (2012, 13.02.): „Multi-column Deep Neural Networks for Image Classification“. URL: <https://arxiv.org/pdf/1202.2745v1.pdf> [21.01.2021]

Alexander Wong, Mohammad Javad Shafiee, Michael St. Jules (2018, 03.10.): „MicronNet: A Highly Compact Deep Convolutional Neural Network Architecture for Real-time Embedded Traffic Sign Classification“. URL: <https://arxiv.org/pdf/1804.00497v3.pdf> [21.01.2021]

Alex D. Pon, Oles Andrienko, Ali Harakeh, Steven L. Waslander (2018, 13.09.): „A Hierarchical Deep Architecture and Mini-Batch Selection Method For Joint Traffic Sign and Light Detection“. URL: <https://arxiv.org/pdf/1806.07987v2.pdf> [21.01.2021]

Mrinal Haloi (2016, 17.07.): „Traffic Sign Classification Using Deep Inception Based Convolutional Networks“. URL: <https://arxiv.org/pdf/1511.02992v2.pdf> [21.01.2021]

11. Abbildungsverzeichnis

Abbildung 1: Bilderkennung und Klassifizierung.....	4
Quelle: https://pyimagesearch.com/wpcontent/uploads/2019/11/traffic_sign_classification_phases.jpg [20.01.2021]	
Abbildung 2: Verteilung der Bilder im Trainingsdatensatz.....	5
Abbildung 3: Verteilung der Bilder im Testdatensatz.....	6
Abbildung 4: Bild eines Verkehrsschildes aus dem Trainingsdatensatz.....	6
Abbildung 5: Bild eines Verkehrsschildes aus dem Testdatensatz	6
Abbildung 6: Aufbau des CNN im Versuch	7
Abbildung 7: Programm zur Vorbereitung des Datensatzes.....	9
Abbildung 8: Training mit verschiedenen Optimizern.....	9
Abbildung 9: Verteilung der Bilder im Trainingsdatensatz mit Bildausgleich	10
Abbildung 10: Effekt der Batchnormalisierung auf die Genauigkeit.....	11
Abbildung 11: Genauigkeit im Verlauf des Trainings	11
Abbildung 12: Verlust im Verlauf des Trainings	11
Abbildung 13: Minima-, Maxima- und Medianwerte im Verlauf des Trainings	11
Abbildung 14: Falsch klassifizierte Bilder	11
Abbildung 15: Falsch klassifizierte Bilder nach weiterem Training	11
Abbildung 16: Aufgenommenes Verkehrsschild	11
Abbildung 17: Aufgenommenes Verkehrsschild	11
Abbildung 18: Aufgenommenes Verkehrsschild	11
Abbildung 19: Aufgenommenes Verkehrsschild	11
Abbildung 20: Ergebnisse der IJCNN 2011	17
Quelle: https://benchmark.ini.rub.de/gtsrb_results_ijcnn.html [20.01.2021]	
Abbildung 21: Ergebnisse von der GTSRB-Website	17
Quelle: https://benchmark.ini.rub.de/gtsrb_results.html [20.01.2021]	
Abbildung 22: Ergebnisse des Kaggle-Wettbewerbs	18
Quelle: https://www.kaggle.com/c/nyu-cv-fall-2018/leaderboard [19.01.2021]	

12. Unterstützungsleistung

Ich möchte allen danken, die mich bei meinem Projekt unterstützt haben. Besonders möchte ich bei Herrn Weissig und der Fakultät für Elektrotechnik und Informationstechnik danken, der es mir ermöglicht hat, Berechnungen über Rechner der TU Chemnitz durchzuführen. So war es möglich meinen PC etwas zu entlasten, da dieser teilweise tagelang für die Ergebnisse auf voller Auslastung rechnen musste. Dazu hat er mir viel über das Thema und wissenschaftliches Arbeiten beigebracht, sodass ich ihm einen Großteil meiner Arbeit verdanke.