Vergleich von implementierten Keras-Architekturen für eine Klassifizierung von Verkehrsschildern anhand des GTSRB-Datensatzes

Samet Dursun, Hanne Raum, Alican Kapusuz und Anton Paul Seymour Elmiger

Zusammenfassung—In dieser Ausarbeitung werden verschiedene Architekturen, die mit der GTSRB Datenbank trainiert wurden, nach ihren Genauigkeiten und ihren Geschwindigkeiten bewertet. Zu zeigen ist, ob es mit einfachen Methoden möglich ist ein Netz zu implementieren, welches mit den besten Ergebnissen der Benchmark konkurrieren kann. Um dieses Ziel zu erreichen, werden bewährte Netz-Architekturen eingesetzt. Durch das Implementieren der in Keras integrierten Netzwerke, war es auch möglich in relativ kurzer Zeit verschiedene Konfigurationen zahlreicher Netzwerke miteinander zu vergleichen. Dabei wurden die Netze jeweils mit und ohne vortrainierten Gewichten des *ImageNet* Datensatzes geladen und mit einer Vielzahl an Epochen ausgeführt. Dabei stellt sich heraus, dass die Xception Architektur durch *fine-tuning* annähernd *state of the art* erreicht.

Index Terms—German Traffic Sign Recognition Benchmark, GTSRB, Bildgestützte Automatisierung, Convolutional Neural Networks (CNN), Keras, Transfer Learning

1 Einleitung

Obwohl in Deutschland ein Abwärtstrend bei den Verkehrstoten pro Jahr zu verzeichnen ist, sind im Jahr 2019 immer noch 3046 Menschen umgekommen [1]. Mit der Entwicklung von autonom fahrenden Autos, sowie dem Einsatz und der Weiterentwicklung von Fahrassitenzsystem, ist zu hoffen, dass Unfälle, die aufgrund der Ermüdung des Fahrenden, der Überschreitung der Geschwindigkeitsbegrenzung, der Unaufmerksamkeit oder der Überforderung bei komplexen Gefahrensituation geschehen, künftig vermieden werden können [3], [2]. Unabdingbar dafür ist eine zuverlässige Klassifikation von Vehrkehrschildern.
Um die Güte des Klassifikationsansatzes einschätzen zu können, wurde im Folgenden die *German Traffic Sign Recognition Benchmark (GTSRB)* Datenbank benutzt. Im Rahmen des IJCNN 2011 Wettbewerbs wurden zahlreiche neuronale Netzwerke, die mit der GTSRB Datenbank trainiert wurden, nach ihrer Genauigkeit bewertet. Um jedoch effizient und schnell ein Deep Neural Network zu implementieren bieten sich bereits vorimplementierte und trainierte Netze an, wie sie beispielsweise in Bibliotheken wie Keras zu finden sind [4].
Nun stellt sich die Frage, ob mit einem bereits implementiertem Netz ein vergleichsweises gutes Ergebnis erzielt werden kann. Um diese Frage zufriedenstellend zu beantworten, werden die folgende Modelle *ResNet50*, *ResNet152V2*, *Xception*, *VGG16*, *VGG19*, *MobileNetV2* und *InceptionV3* unter verschiedenen Ansätzen trainiert und ausgewertet. Dabei wird unter anderen untersucht, ob ein Transfer Learning Ansatz, also ein Import der Gewichtungen aus dem Training der Netze mit dem *ImageNet* Datensatz, oder ein bloßes importierten der Architektur der Netze sinnvoller ist. Zusätzlich wird die Geschwindigkeit während des Training und während der Vorhersage einzelner Bilder untersucht, sowie bei ausgewählten Netzen der Einfluss der Tiefe auf die Genauigkeit der Ergebnisse.
Im Folgenden wird nach einen kurzen Überblick über den aktuellen Stand der Technik in Abschnitt 2 das Konzept in Abschnitt 3 detailliert beschrieben. In Abschnitt 4 wird die exakte Durchführung und der Ablauf der Implementierungen detailliert beschrieben. In Abschnitt 5 werden die Ergebnisse vorgestellt und diskutiert und letztendlich werden im Abschnitt 6 die wichtigsten Ergebnisse zusammengefasst und ein kurzer Ausblick gegeben.

2 Stand der Technik

Das von Akatsuka et al. entwickelte System [5] zählt zu den ersten Versuchen zur automatischen Verkehrsschilderkennung und seitdem wurden verschiedene Ansätze und Methode entwickelt, die auf klassischem maschinellen Lernen und kürzlich auf Deep Learning beruhen.
Obwohl die maschinelles Lernen basierten Ansätze, wie Support Vector Machines (SVM) [6] und Random Forest Klassifikatoren [7], bei der Verkehrsschilderkennung gute Resultate liefern, müssen sie mit bestimmten Merkmalsextraktoren wie z.B. HOG, SIFT usw. kombiniert werden.
Der gängigste Deep Learning Ansatz im Bereich maschinelles Sehen ist die Convolutional Neural Networks (CNN). Aufgrund der steigenden Rechenleistung, Erreichbarkeit der großen Datensätze und deren hohen Genauigkeit werden in letzter Zeit CNN basierte Klassifikationsmodelle bevorzugt. Deren turmartige Struktur macht diese Netzwerke leichter implementierbar und genauer, da die tiefen

Schichten ermöglichen die Daten selbständig auf sehr hohen Niveau zu abstrahieren und zu verstehen [2].
Vor der Veröffentlichung von *GTSRB* wurden verschiedene Architekturen für Verkehrsschilderkennung mit unterschiedlichen Datensätzen trainiert, was die Vergleichbarkeit dieser Architekturen erschwert. Aus diesem Grund verwenden wir in dieser Ausarbeitung auch den *GTSRB* Datensatz. In der *GTSRB* erzielte mit $99,71\%$ bei der Klassifikation von Verkehrsschildern ein Multi-column Deep Neural Network einer der besten Ergebnisse. Dieses Netzwerk besteht aus 2 Convolutional-Schichten, gefolgt von Max-Pooling-Schichten. Am Ende werden 2 vollständig verbundene verborgene Schichten verwendet, um die Ausgabe an eine letzte vollständig verknüpfte Schicht mit 6 Neuronen zur Durchführung der Klassifizierung weiterzuleiten [8].
Das LeNet-5 Netzwerk zählt auch als einer der besten CNN Architekturen für die Verkehrsschilderkennung. Es besteht aus 7 Schichten, 3 Faltungsschichten, gefolgt von Unterabtastungsschichten, einer vollständig verbundenen Schicht und der endgültigen Ausgabeschicht, die aus euklidischen RBF-Einheiten besteht [9].
Außer der bisher erläuterten Netzwerke enthält Keras vortrainierte Netzwerke zur Bildklassifikation, die mit ImageNet-Daten trainiert werden. Sie sind nämlich *ResNet50*, *ResNet152V2* [11] , *Xception* [12], *VGG16* [10], *VGG19*, *MobileNetV2* [13] und *InceptionV3* [14]. Diese vortrainierten Netzwerke sind in der Lage, jedes Bild zu klassifizieren, das in 1000 Bildkategorien fällt und sie können auch für Transfer Learning eingesetzt werden. Da diese Netzwerke komplett in der Keras Bibliothek integriert sind, können sie schnell und nahtlos implementiert werden. Daher ist der Vergleich von diesen Architekturen relativ einfacher.

3 KONZEPT

Die Netze *ResNet50*, *ResNet152V2*, *Xception*, *VGG16*, *VGG19*, *MobileNetV2* und *InceptionV3* unterscheiden sich alle in ihrer Architektur sei es durch shortcuts/identity mapping im Falle des (*ResNet*), verschiedene Arten der Faltung (bspw. *Xception*), ein Aufbau mit sogenannten Inception Modulen (bspw. *InceptionV3*) oder der Tiefe des Netzwerkes (bspw. *VGG16* und *VGG19*). Detaillierte Informationen sind der angegebenen Literatur zu entnehmen. Alle angebenden Netze wurden mit dem Datensatz *ImageNet* bereits trainiert und dessen Gewichtung gespeichert. Daraus ergeben sich drei Möglichkeiten die Netze aus der Keras Bibliothek zu importieren und auf den *GTSRB* Datensatz zu trainieren:

1) Die Netze werden ohne vortrainierte Gewichte geladen
2) Die Netze werden mit vortrainierten Gewichten des *ImageNet* Datensatzes geladen, jedoch werden nur die letzten Klassifizierungs Layer trainiert
3) Die Netze werden mit vortrainierten Gewichten des *ImageNet* Datensatzes geladen und alle Layer werden trainiert (*fine tuning*)

Im Anschluss werden die Netze nach folgenden Kriterien ausgewertet:

1) Genauigkeit der Klassifikation bei den Testdaten des *GTSRB* Datensatzes mit folgernder Unterscheidung:
   a) allgemeine Genauigkeit gemittelt über alle 43 Klassen des Testdatensatzes
2) Ermittlung der Geschwindigkeit:
   a) beim Training der Netze
   b) beim Testen der Netze

Das Konzept der Auswertung wurde ausgelegt, um einen möglichst guten Vergleich der Netze unter praktischen Gesichtspunkten zuzulassen. Um mögliche Faktoren bedingt durch die Hardware auszuschließen, wurde für einen Vergleich der einzelnen Geschwindigkeiten die Netze bei 2 nur auf einem Rechner ausgeführt.

4 EXPERIMENTE

4.1 Durchführung

Im Folgenden wird beschrieben, wie die einzelnen Architekturen trainiert und ausgewertet werden. Um für Vergleichbarkeit zu sorgen, muss sowohl die Hardware, als auch die Software beim Training dieselbe bleiben.

4.1.1 Hardware

Trainiert werden die neuronalen Netze auf einer NVIDIA RTX 2070S mit 8GB V-RAM. Der Prozessor des Systems ist ein AMD Ryzen 5 3600 mit 16GB RAM.

4.1.2 Software

Framework für die Experimente ist Keras mit einem Tensorflow-Backend (Version 2.3). Das Netzwerk wird auf der GPU mit CUDA 10.1 und CUDNN 7 trainiert.

4.1.3 Ablauf

Zu Beginn wird der Trainings-Datensatz geladen. Da die Größe der Bilder variiert, werden diese auf 128x128 Pixel skaliert. Zusätzlich werden die Pixelwerte der einzelnen Farbkanäle von 0-255 auf 0-1 skaliert.
Die Bilder werden in eine Trainings- und eine Validierungs-Gruppe mit einem Verhältnis von 80/20 aufgeteilt. Ziel der Aufteilung ist die Performance des Netzes anhand von nicht gesehenen Bildern nach jeder Epoche evaluieren zu können. Gleichzeitig werden die Bilder in Batches mit der Größe 64 zusammengefasst. Die Größe der Batches hat vorrangig einen Einfluss auf die Geschwindigkeit des Trainings, wird jedoch limitiert durch den V-RAM.
Anschließend werden die Architekturen der unterschiedlichen neuronalen Netze in Keras geladen. Diese sind bereits in der Bibliothek integriert. Für die Experimente werden diese Architekturen unter drei verschiedenen Konfigurationen geladen (Abschnitt 3). Da der *ImageNet* Datensatz 1000 Klassen besitzt und der GTSRB Datensatz nur aus 43 Klassen besteht, wird jeweils das letzte Layer entfernt und durch ein Dense-Layer mit 43 Knoten ersetzt.
Die Learning-Rate wird auf 0,001 gesetzt und es werden jeweils 40 Epochen berechnet. Die Anzahl der Epochen wurde experimentell bestimmt. Nach 40 Epochen konvergiert die Validation-Genauigkeit. Diese Konvergenz wird beispielhaft für drei neuronale Netze in Abbildung 1 dargestellt. Da die Klassen des *GTSRB* Datensatzes nicht balanciert (Abb. 2) sind muss dies beim Training beachtet werden. Diese

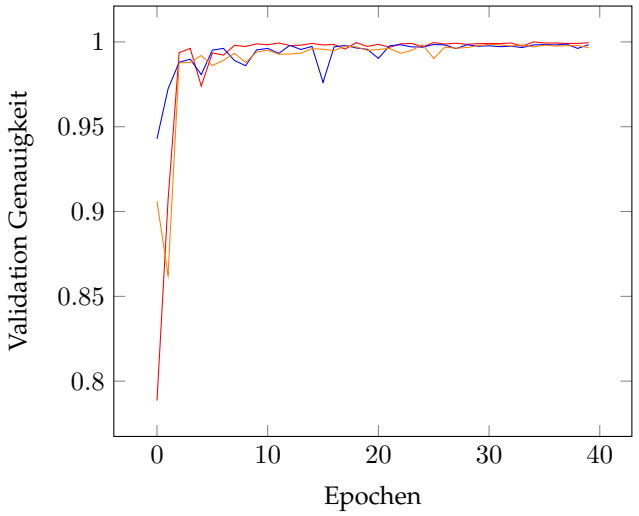

Abbildung 1. Konvergenz der Validation Genauigkeit

Netz	Ablauf	Genauigkeit	Ausführungsdauer (ms/Bild)
InceptionV3	nicht vortrainiert	95,43%	0,893
	vortrainiert locked base	69,43%	0,783
	vortrainiert	96,92%	0,924
MobileNetV2	nicht vortrainiert	96,51%	0,499
	vortrainiert locked base	75,80%	**0,378**
	vortrainiert	98,27%	0,447
ResNet50v2	nicht vortrainiert	97,14%	0,760
	vortrainiert locked base	77,29%	0,748
	vortrainiert	98,04%	0,785
ResNet152v2	nicht vortrainiert	96,76%	1,743
	vortrainiert locked base	75,56%	1,629
	vortrainiert	97,05%	1,814
VGG16	nicht vortrainiert	5,70%	0,947
	vortrainiert locked base	66,89%	0,822
	vortrainiert	97,24%	0,824
VGG19	nicht vortrainiert	5,70%	0,896
	vortrainiert locked base	60,39%	0,849
	vortrainiert	95,19%	0,848
Xception	nicht vortrainiert	97,99%	0,904
	vortrainiert locked base	70,68%	0,909
	vortrainiert	**98,83%**	0,943

Tabelle 1
Genauigkeiten der Klassifikation und ihre Ausführungszeiten pro Bild

Gewichtung der Klassen verstärkt den Einfluss von Verkehrszeichen, welche seltener auftreten, auf das Netzwerk. Nach dem Training gibt es einen eigenen Datensatz für das Testen des Modells.

5 Diskussion

Nach erfolgreichem Ablauf der Experimente, sind in Tabelle 1 die Ergebnisse der Genauigkeiten der Klassifikation und ihre Ausführungszeiten pro Bild für die angewendeten Netze aufgelistet.
Für die Abläufe ohne vortrainierte Gewichte erfolgt mit zwei Ausnahmen eine Klassifizierung mit Genauigkeiten zwischen 95% und 98%. Im Vergleich zu den angegebenen Resultaten der GTSRB würden die Genauigkeiten auf der Rangfolge zwischen den sechsten und achten Plätzen liegen. Bei den ersten fünf Rängen handelt es sich um CNN-Methoden basierte Algorithmen und der *Human Performance* auf dem vierten Rang mit $98,84\%$. Für diese Abläufe werden die zwei Ausnahmen *VGG16*, *VGG19* mit je $5,7\%$ nicht zur Kenntnis genommen, da dies wahrscheinlich einer zufälligen Klassifikationsgenauigkeit entspricht. Bei den Abläufen der Netze, bei denen nur die letzten Klassifizierungs-Layer vortrainiert werden, erfolgt die Klassifizierung nur mit einer Genauigkeit von 60% bis 78%. Deshalb wird diese Herangehensweise nicht für die Anwendung vorgeschlagen. Für das Training mit vortrainierten Gewichten des *ImageNet* Datensatzes, wo alle Layer trainiert werden, erhalten wir Klassifizierungen mit zufrieden stellenden Genauigkeiten. Für das Netz *Xception* liegt die Genauigkeit bei $98,83\%$, was auf der GTRSB Platzierung dem fünften Rang entsprechen würde.
In den Abbildungen 3 bis 5 sind die Ergebnisse der Genauigkeit über die Ausführungszeit pro Bild für alle angewendeten Netze dargestellt. Diese Darstellung vereinfacht die Wahl des Netzes für Anwendungen, die neben einer hohen Genauigkeit auch eine hohe Geschwindigkeit der Klassifizierung voraussetzen. Auffällig sind hier die Ergebnisse des *ResNet15v2* Netzes, dessen Ausführungszeiten deutlich über denen der anderen Netze liegen.
Die Frage, ob mit einem bereits implementiertem Netz ein zufriedenstellendes gutes Ergebnis erzielt werden kann, kann bestätigt werden. Zu betrachten sind jedoch auch die Zeiten, die benötigt werden um die Netze zu trainieren. In Tabelle 2 sind die Trainingszeiten für die gewählten Netze dargestellt. Für das *Xception*-Netz liegt die Trainingsdauer für 40 Epochen mit der gegebenen Hardware bei 1 Stunde 18 Minuten. Wir haben anhand eines Versuchslaufs angenommen, dass die Genauigkeit bei 40 Epochen bereits konvergiert. Nun könnte man weitere Ablaufe für bestimmte Netze mit mehr Epochen testen, um zu bestimmen, bei welcher Anzahl die Konvergenzgrenze des Ergebnisses liegt.
In Abbildung 6 ist die Confusion-Matrix für das *Xception*-Netz zu sehen. Zu sehen ist, dass nur wenige auffällige Falsch-Klassifizierungen sich häufen. In Abbildung 7 ist ein Beispiel für die Falscherkennung des *Baustellen* Schildes als *einmalige Vorfahrt* Schild gezeigt. Hier kann angenommen werden, dass aufgrund der Skalierung der Bilder die Bildqualität zu stark gelitten hat.
Kritisch zu betrachten ist die Vielzahl von sehr ähnlichen Bildern im Traings Datensatz. Dies kann dazu führen, dass das Netz ein bestimmtes Bild bevorzugt, da es sehr häufig im Datensatz vorkommt. Desweiteren ist die Validations Genauigkeit nicht aussagekräfig beim Training, da manche sehr ähnliche Bilder sowohl im Trainings, als auch im Validations Datensatz auftreten und somit dem Netz bereits bekannt sind. Auch im Test Datensatz gibt es viele sehr ähnliche Bilder, sodass ein falsch klassifiziertes Bild gleich mehrfach gewichtet wird und umgekehrt. Dies ist auch sichtbar in der Abbildung 7.

6 Schluss

In dieser Ausarbeitung werden verschiedene Architekturen, die mit der GTSRB Datenbank trainiert wurden, nach ihrer Genauigkeit und Geschwindigkeit bewertet. Um dieses Ziel zu erreichen, wurden die vortrainierten Modelle in Keras

Model	Trainingsdauer mit/ohne Vortrainieren		Trainingsdauer mit locked base vortrainiert	
	gesamt	pro Epoche	gesamt	pro Epoche
InceptionV3	00:38:00	00:00:57	00:16:00	00:00:24
MobileNetV2	00:32:00	00:00:48	00:12:00	00:00:18
ResNet50v2	00:53:00	00:01:20	00:20:00	00:00:30
ResNet152v2	01:51:00	00:02:47	00:41:00	00:01:02
VGG16	01:00:00	00:01:30	00:25:00	00:00:38
VGG19	01:10:00	00:01:45	00:28:00	00:00:42
Xception	01:18:00	00:01:57	00:24:00	00:00:36

Tabelle 2
Trainingsdauer der Netze

verwendet. Durch Implementierung von der in Keras integrierten Netzwerken war es auch möglich in relativ kurzer Zeit verschiedene Konfigurationen zahlreicher Netzwerke miteinander zu vergleichen.
In der ersten Konfiguration wurden die Netze ohne vortrainierte Gewichte, in der zweiten wurden nur ihre letzte Klassifizierungsschicht und in der dritten alle Schichten mit vortainierten Gewichten trainiert. Nach der Durchführung der Experimente wird festgestellt, dass bereits implementierte Netzwerke Genauigkeiten über 95% erreichen können. Desweiteren ermöglicht die Trainingszeit von weniger als 2 Stunden eine Optimierung der Hyperparameter wie der Learning-Rate. Beispielsweise konnte durch eine Verringerung der Learning-Rate bei längerem Training das *Xception* Netzwerk eine Genauigkeit von $99,15\%$ erreicht werden. Außerdem bietet die geringeren Trainingszeiten der vortrainierten Netze auch den folgenden Vorteil an: Die Eignung der zahlreichen Netzwerke für bestimmte Anwendungsfälle wird zügig dargestellt und demnach kann das passende Netzwerk ausgewählt und anschließend optimiert werden.
Um die Klassifizierung von Verkehrsschildern zu verbessern kann der Datensatz durch Augmentation erweitert werden. In der Ausweitung dieser Ausarbeitung können die vortrainierten Modelle mit einem Datensatz aus europäischen Verkehrsschildern trainiert und die Auswirkung der Intravarianz von Klassen auf die Genauigkeit untersucht werden.

ANHANG A

Klassenhäufigkeit im Datensatz. (Abb. 2)
Grafische Darstellungen für Genauigkeit über Geschwindigkeit der Netze. (Abb. 3-5)

ANHANG B

Confusion-Matrix aller Klassen für *Xception*. (Abb. 7)

LITERATUR

[1] *https://www.destatis.de/DE/Themen/Gesellschaft-Umwelt/Verkehrsunfaelle/Grafik/Interaktiv/verkehrsunfaelle-getoetete-jahr.html*,
[2] Sourajit Saha, Sharif Amit Kamran, Ali Sabbir *Total Recall: Understanding Traffic Signs Using Deep Convolutional Neural Network* 21st International Conference of Computer and Information Technology (ICCIT) 2018, p.1-6,
[3] Smith, Andrew P, *A UK Survey of Driving Behaviour, Fatigue, Risk Taking and Road Traffic Accidents*, BMJ Open 6.8 (2016)
[4] https://keras.io/api/applications/
[5] H. Akatsuka and S. Imai, *Road signposts recognition system*, SAE transactions, Vol 96, No. 1, pp. 936-943, Feb. 1987.
[6] S. M. Bascón, J. A. Rodríguez, S. L. Arroyo, A. F. Caballero, and F. López-Ferreras, *An optimization on pictogram identification for the road-sign recognition task using SVMs*, Comput. Vis. Image Understand., vol. 114, no. 3, pp. 373–383, 2010.
[7] A. Ellahyani, M. El Ansari, and I. El Jaafari, *Traffic sign detection and recognition based on random forests*, Applied Soft Computing, vol. 46, pp. 805–815, 2016
[8] Cireşan, Dan, Ueli Meier, Juergen Schmidhuber, *Multi-Column Deep Neural Networks for Image Classification*, CVPR 2012, p. 3642-3649
[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[10] Karen Simonyan, Andrew Zisserman *Very deep convolutional networks for large-scale image recognition*
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun *Deep Residual Learning for Image Recognition*
[12] Francois Chollet, *Xception: Deep Learning with Depthwise Separable Convolutions*
[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* Andrew G. HowardMenglong ZhuBo ChenDmitry KalenichenkoWeijun WangTobias WeyandMarco Andreetto-Hartwig Adam, arXiv:1704.04961
[14] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlensshlens, Zbigniew Wojna *Rethinking the Inception Architecture for Computer Vision* Computer Vision and Pattern Recognition, arXiv:1512.00567

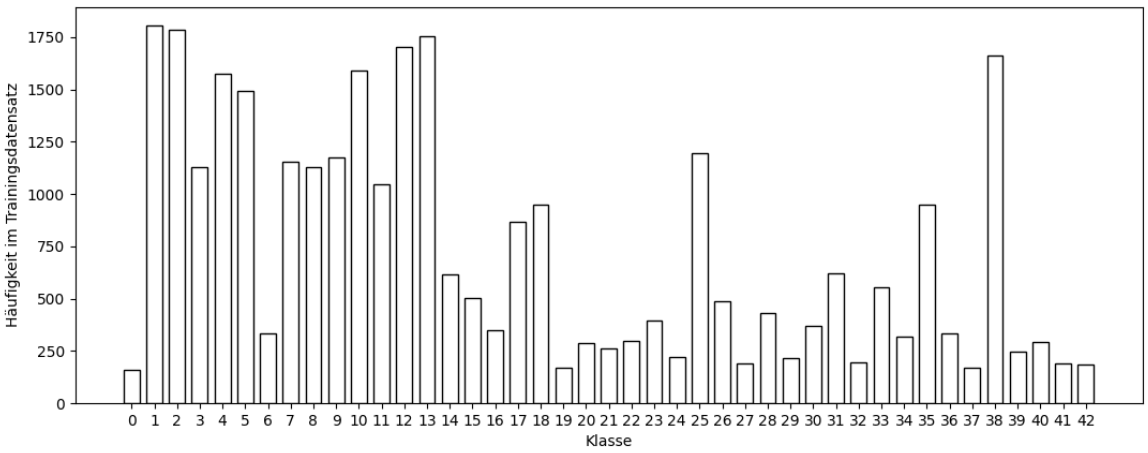

Abbildung 2. Häufigkeiten der Klassen im Datensatz

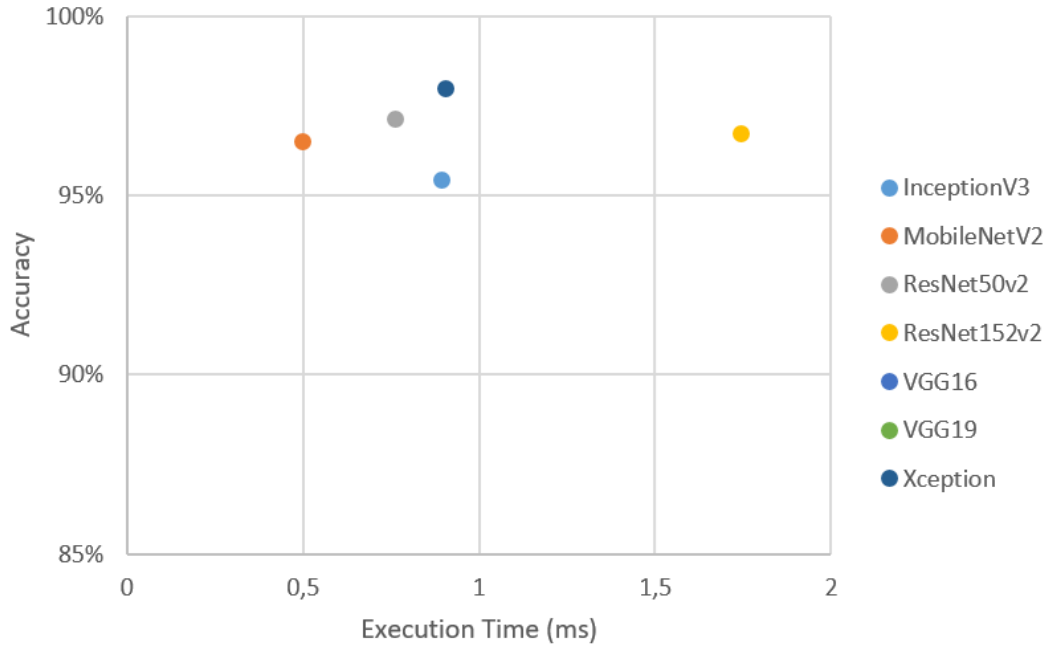

Abbildung 3. Genauigkeit über Geschwindigkeit für Netze ohne vortrainierten Gewichte

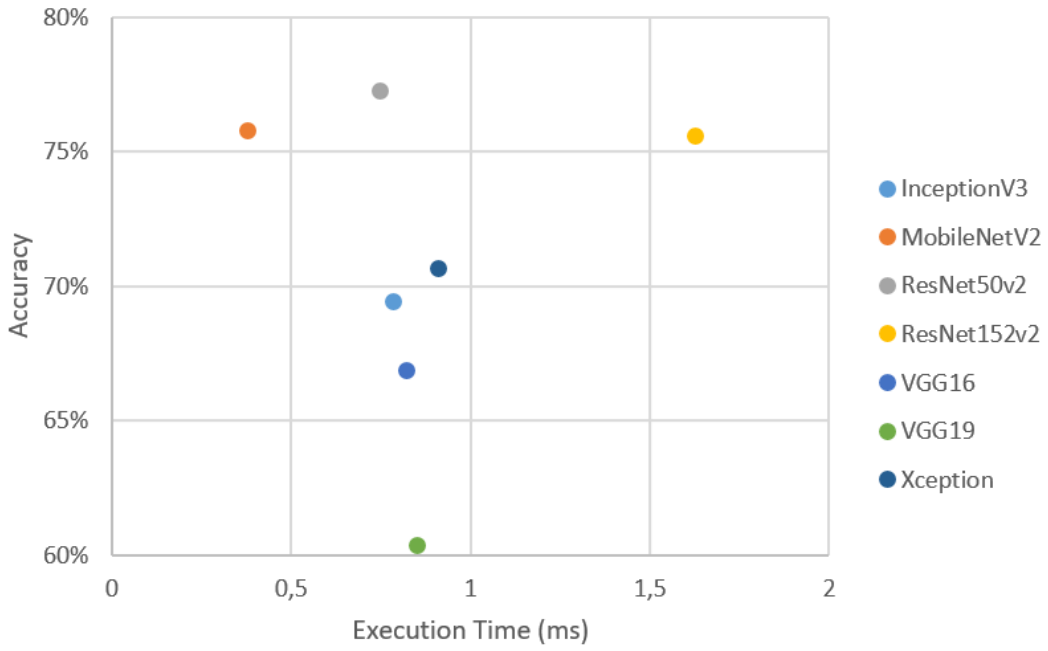

Abbildung 4. Genauigkeit über Geschwindigkeit für Netze mit teils vortrainierte Gewichte

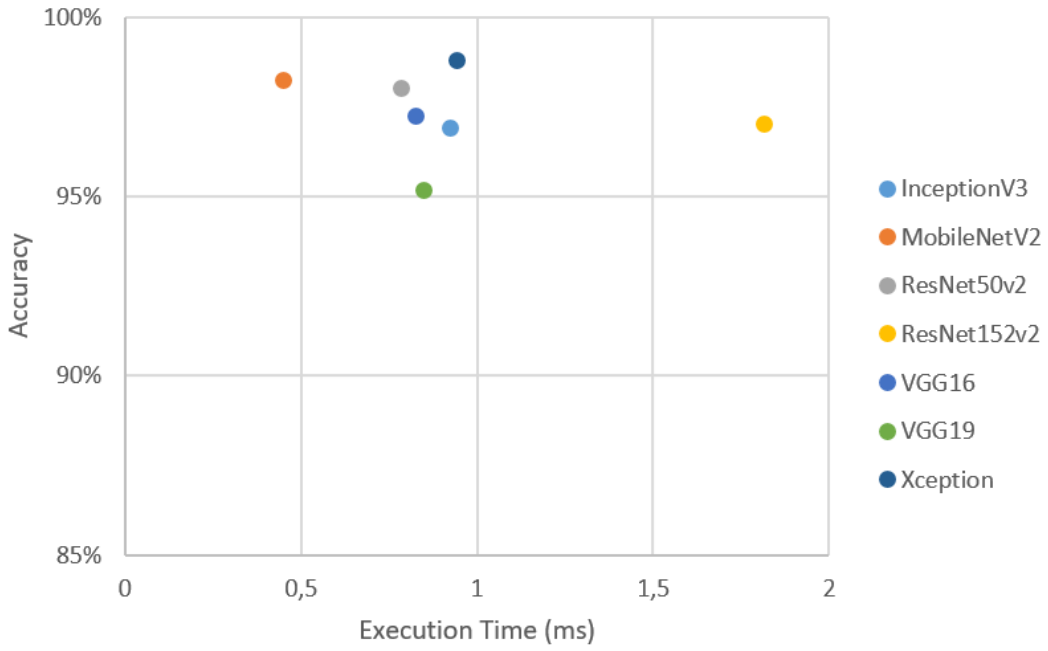

Abbildung 5. Genauigkeit über Geschwindigkeit für Netze mit vortrainierten Gewichte

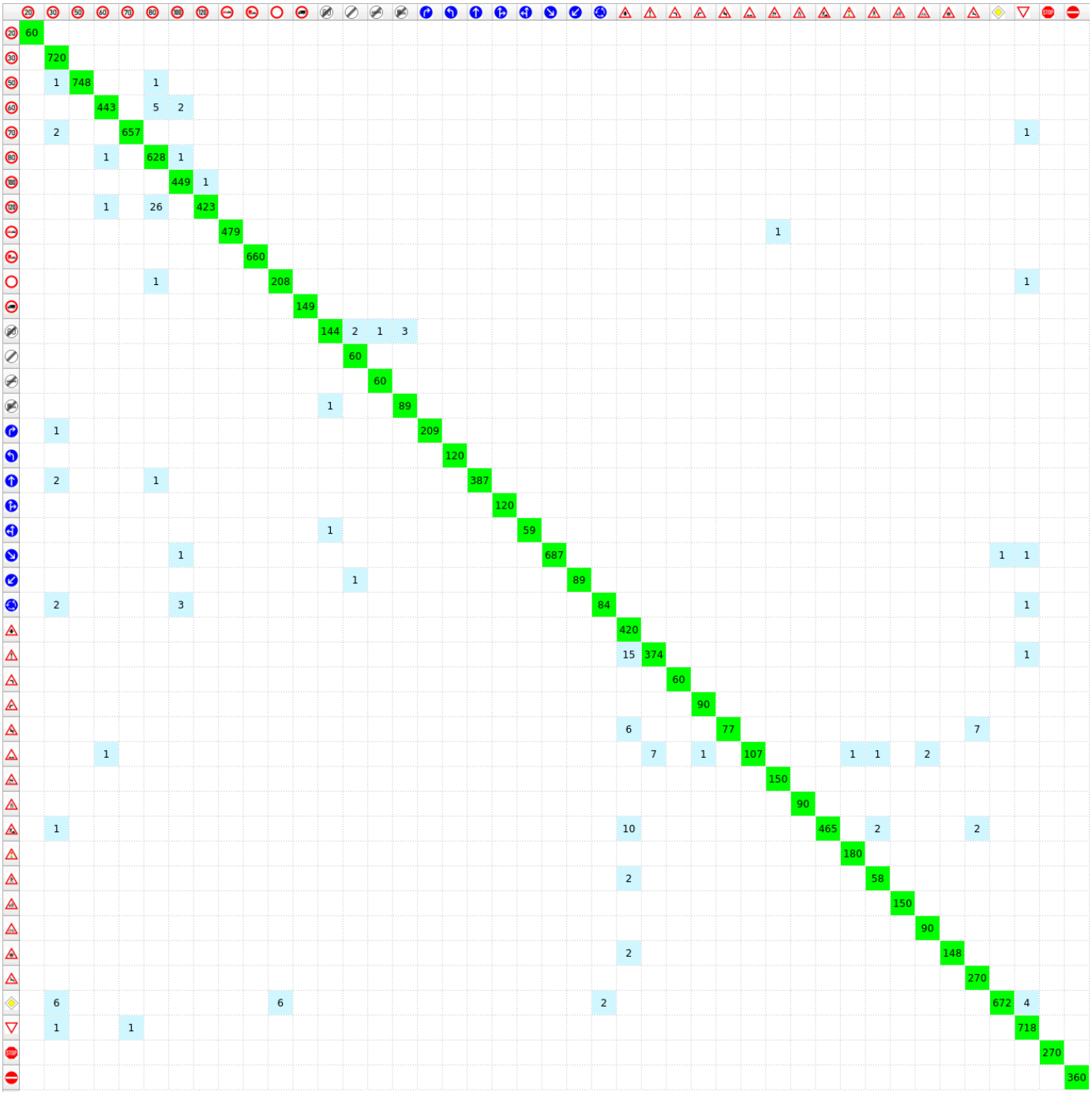

Abbildung 6. Confusion-Matrix aller Klassen für *Xception*

Abbildung 7. Fehlerhafte Klassifizierung