Untersuchung der Auswirkungen von Quantisierung und Pruning auf Convolutional Neural Networks für Mobilgeräte

Julian Hoever

31. März 2021

Inhaltsverzeichnis

Einleitung

- Convolutional Neural Networks für die Bildverarbeitung
- Viele Architekturen sind zu groß um auf Mobilgeräten verwendet zu werden
- Optimierte Architekturen für mobile/eingebettete Systeme
  - MobileNet, MobileNetV2, MobileNetV3 und EfficientNet
  - Architekturelle Optimierungen
- Optimierungstechniken: Quantisierung und Pruning

Motivation

- Architekturen können unterschiedlich auf die Anwendung von Quantisierung und Pruning reagieren
- Gibt es Architekturen die sich besonders gut für solche Optimierungen eignen?

Grundlagen

MobileNet (2017) [HZC+17]

- Zielsetzung: Verwendung von CNN Modellen auf mobilen/eingebetteten Systemen (möglichst niedrige Latenz)
- Aufgebaut aus Depthwise Separable Convolutions

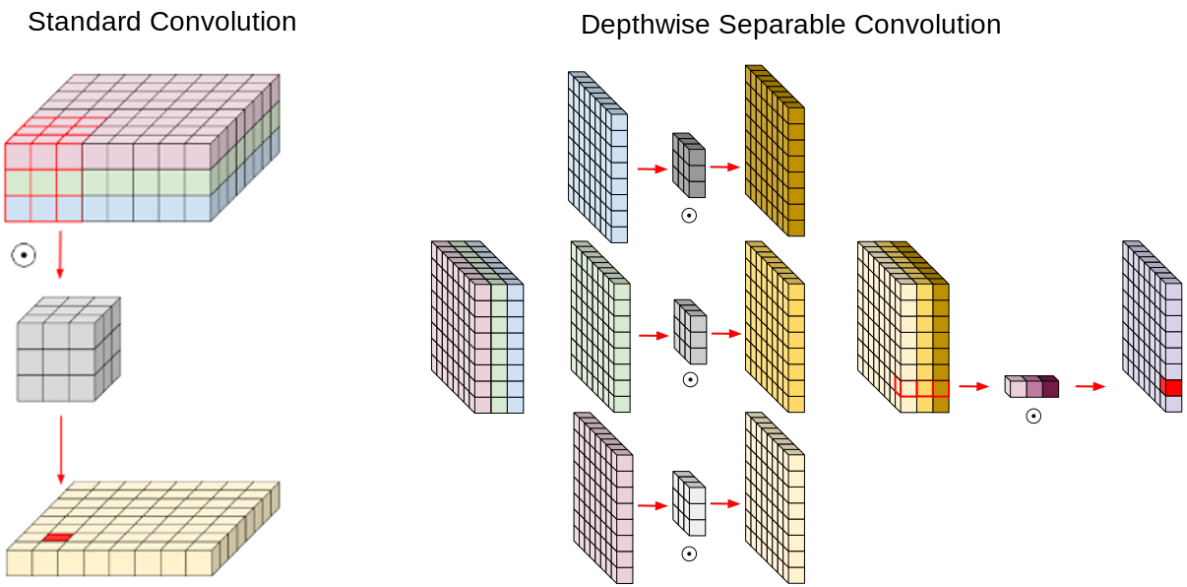

Abbildung: Schema einer Depthwise Separable Convolution [1]

[1] https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning/

MobileNetV2 (2018) [SHZ+19]

- Nachfolger von MobileNet
- Verwendet ebenfalls Depthwise Separable Convolutions
- Neu hinzugekommen: Linear Bottlenecks und Inverted Residuals
- Hauptbaustein ist der Inverted Residual Block

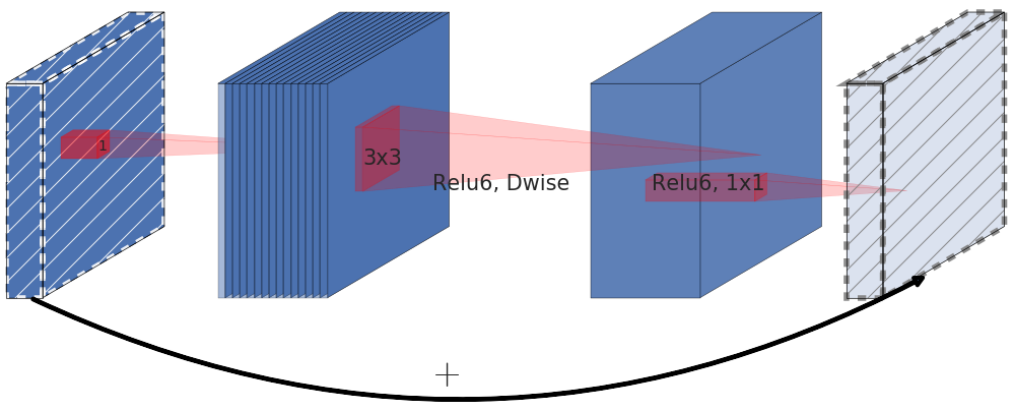

Abbildung: Inverted Residual Block mit Linear Bottleneck [SHZ+19]

MobileNetV3 Large/Small (2019) [HSC+19]

- Mittels Platform-Aware Neural Architecture Search entwickelt
  - Latenz wird auf Google Pixel Smartphones gemessen
  - MobileNetV3 Large basiert auf der MnasNet-A1 Architektur
  - Neuer Platform-Aware NAS Durchlauf für MobileNetV3 Small
- NetAdapt Algorithmus und manuellen Anpassungen
- MobileNetV3 verwendet Squeeze-And-Excitation Module

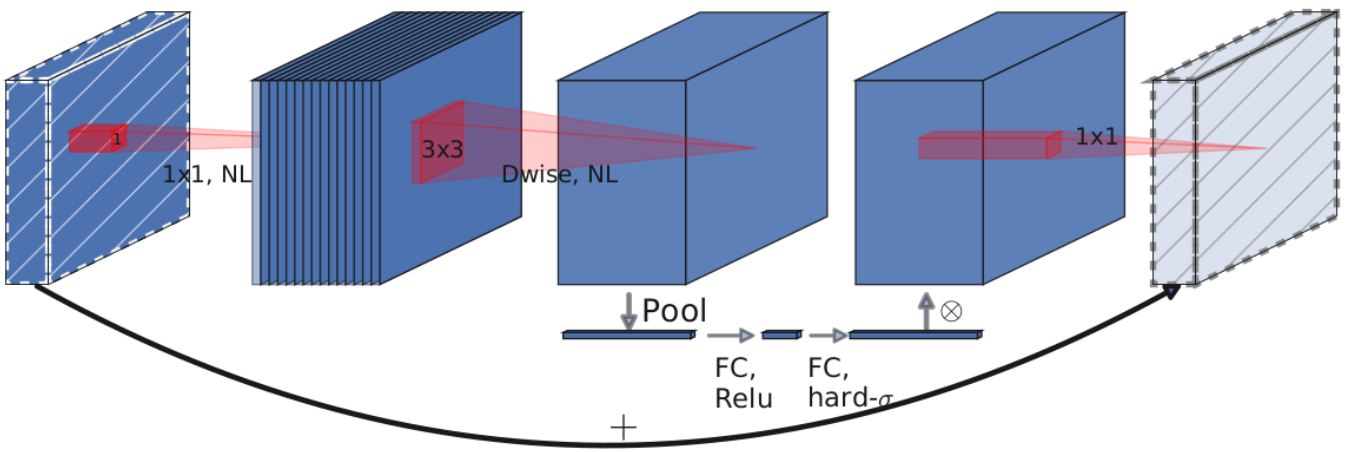

Abbildung: MobileNetV3 Building Block [HSC+19]

EfficientNet (2019) [TL20]

- Ziel: skalierbare Architektur um einen optimalen Trade-Off zwischen Genauigkeit und Ressourcenbedarf zu erreichen
- Mittels Neural Architecture Search entwickelt
  - Nicht für eine spezielle Hardware optimiert
  - Optimierung auf Gleitkommazahlen-Operationen
  - Basisarchitektur EfficientNet-B0
- Compound Scaling
- EfficientNet-B0 bis EfficientNet-B7
- Hauptbausteine ähneln dem MobileNetV3

Quantisierung

- Darstellung eines Netzwerkes verkleinern
  - Gewichte und Aktivierungen in eine Darstellung überführen, die weniger Bits benötigt.
- Post-Training Quantisierung
  - Bereits trainiertes Netzwerk wird ohne erneutes Training quantisiert.
- Quantisierungsschema von TensorFlow [JKC+17]
  - Reelle Zahl r wird durch quantisierten Wert q wie folgt dargestellt:

$$r = S(q - Z)$$

  - S, Z sind Konstanten
- In dieser Arbeit: Post-Training Float-Fallback Quantisierung

Pruning

- Nicht benötigte Verbindungen werden gekappt
  - Gewichte mit geringem Einfluss werden auf 0 gesetzt
- Magnitude Pruning [ZG17]
  - Anteil der auf 0 gesetzten Gewichte (Sparsity)
  - Training und Pruning im Wechsel
  - Schrittweise werden die kleinsten Gewichte einer Gewichtsmatrix auf 0 gesetzt.

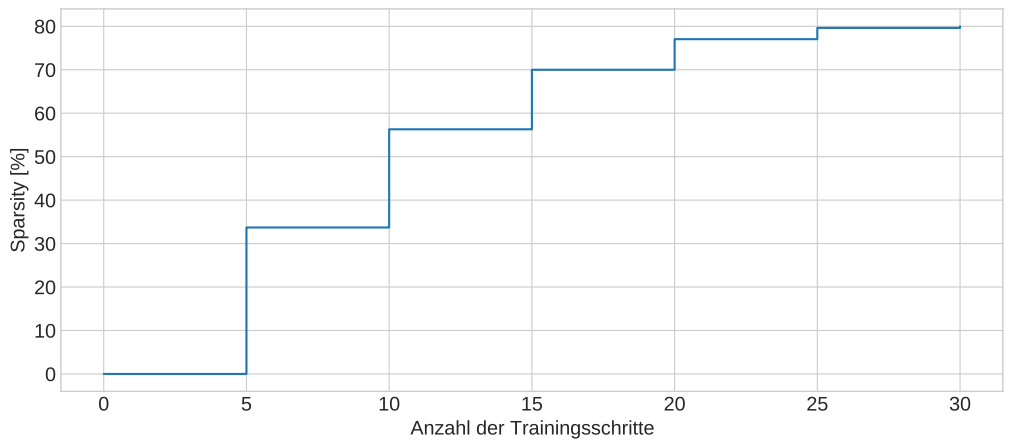

Abbildung: Schrittweises Pruning

Vorgehen

Vorgehen

- Architekturen werden auf dem CIFAR-10 Datensatz trainiert
- Pruning und Quantisierung werden auf die Architekturen angewendet
- Folgende Kombinationen von Optimierungen werden untersucht:
  - vortrainiert (keine Optimierungen)
  - vortrainiert und quantisiert
  - vortrainiert und gepruned
  - vortrainiert, gepruned und quantisiert
- TFLite FlatBuffer Format wird als einheitliches Format für die Speicherung der Modelle verwendet
- Modelle werden auf einem Raspberry Pi 4B ausgewertet

Ergebnisse

Training

- Training mittels einheitlichem Setup
- Overfitting der Architekturen
- MobileNetV3 Architekturen ließen sich schwierig trainieren
- Hohe Inferenzzeit des EfficientNet-B0 im Gegensatz zum MobileNetV3 Large

Architekturen	Parameter	Top-1	Top-3	Datei [MB]	RAM [MB]	Inferenz [µs]
MobileNet	3.2 Mio.	0.7277	0.9226	12.8453	16.1016	11803
MobileNetV2	2.3 Mio.	0.7325	0.9263	8.9153	11.2148	4961
MobileNetV3 Large	4.2 Mio.	0.6934	0.9064	16.8829	20.4141	8051
MobileNetV3 Small	1.5 Mio.	0.6687	0.9033	6.1536	8.5859	2954
EfficientNet-B0	4.1 Mio.	0.7514	0.9296	16.0860	16.6719	12453

Tabelle: Zusammenfassung verschiedener Metriken nach dem Training

Quantisierung I

- Quantisierung verkleinert die Architekturen stark
- Geringe bis teilweise keine Verluste an Genauigkeit
  - Ausnahme: MobileNetV3
- Inferenzzeit verringert sich stark (besonders bei der MobileNet Architektur)

Architekturen	Parameter	Top-1	Top-3	Datei [MB]	RAM [MB]	Inferenz [µs]
MobileNet	3.2 Mio.	0.7401	0.9312	3.6149	6.01560	3750
MobileNetV2	2.3 Mio.	0.7507	0.9411	2.8626	6.91410	2843
MobileNetV3 Large	4.2 Mio.	0.3963	0.7743	4.9440	6.93359	5259
MobileNetV3 Small	1.5 Mio.	0.6349	0.8926	1.9539	5.20310	2183
EfficientNet-B0	4.1 Mio.	0.7464	0.9324	5.1753	9.63670	8629

Tabelle: Metriken nach Anwendung von Quantisierung

Quantisierung II

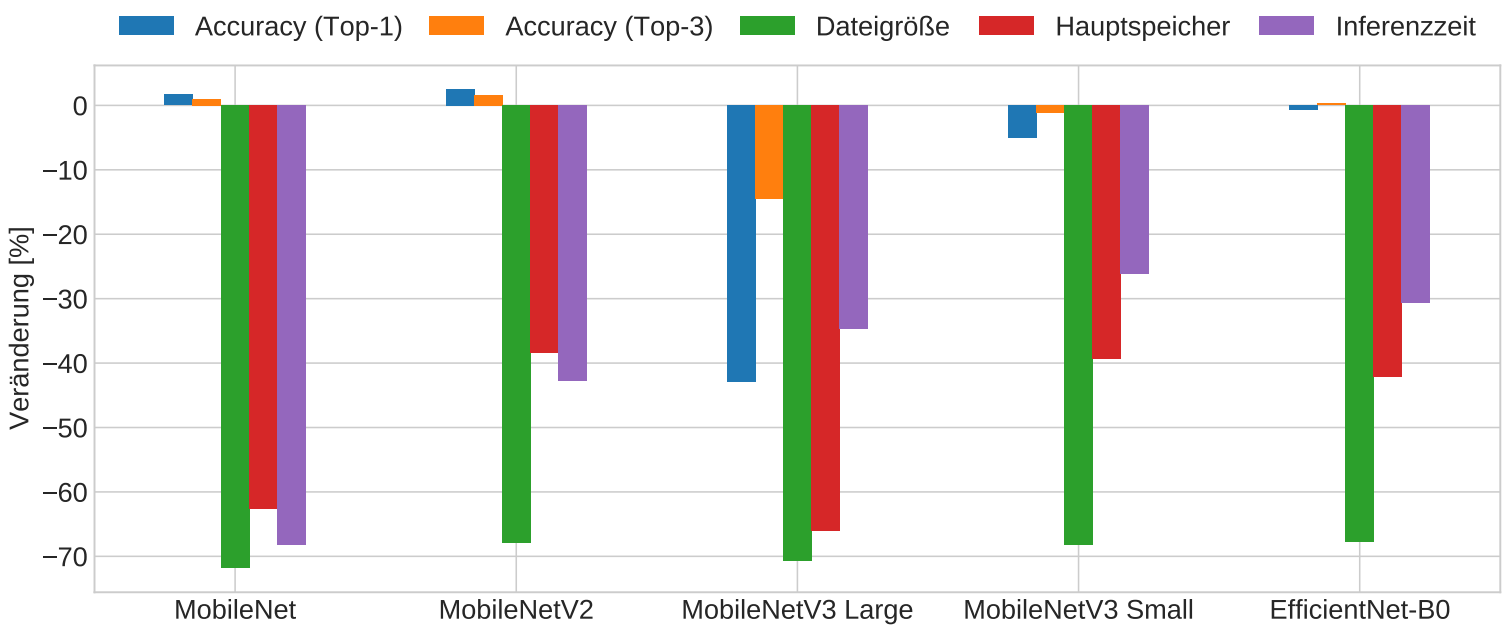

Abbildung: Veränderung der klassenweisen Top-1 Genauigkeit der geprunten MobileNet Modelle

Pruning I

- Mit zunehmender Sparsity sinkt die Genauigkeit
- MobileNet verzeichnet die kleinsten Verluste beim Pruning

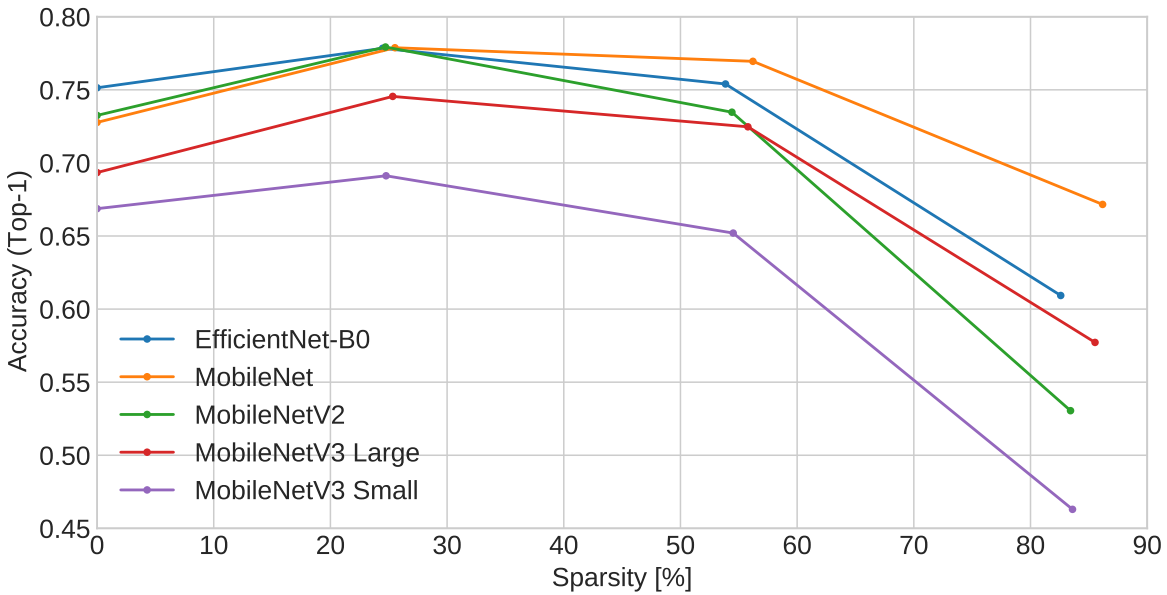

Abbildung: Genauigkeiten (Top-1) beim Pruning auf 0% (ungepruned), 30%, 60% und 90% Sparsity

Pruning II

- Bedarf an Hauptspeicher, Dateigröße und Inferenzzeit verändern sich durch das Pruning nicht
- Anwendung eines Kompressionsalgorithmus (gzip) auf die gespeicherten .tflite-Modelle
  - Anwendung: z.B. Übertragung über ein Netzwerk/Internet

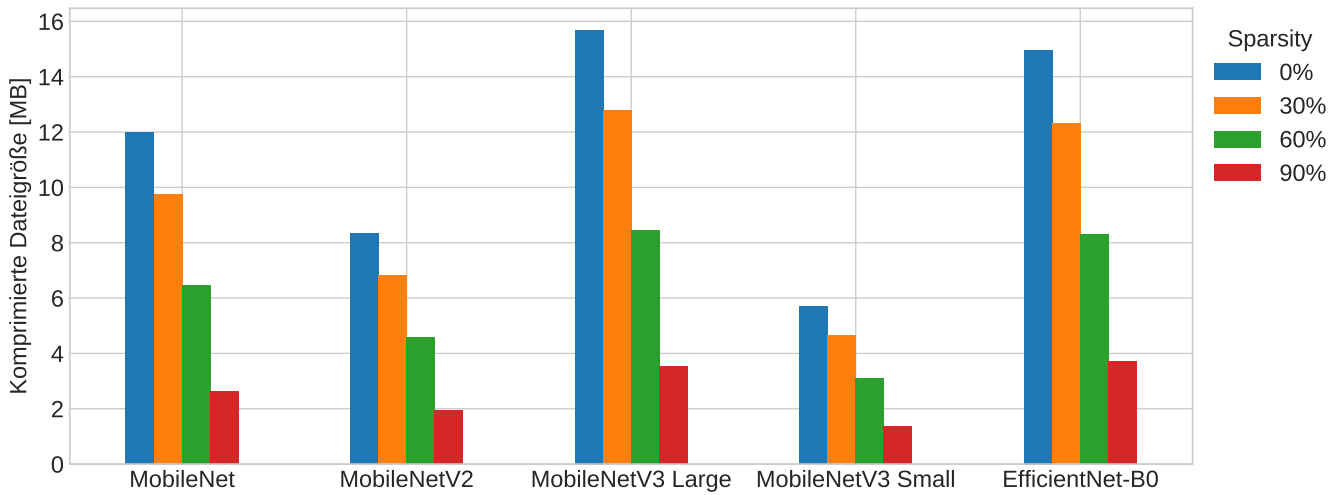

Abbildung: Dateigröße der Modelle nach Anwendung eines Kompressionsalgorithmus (gzip)

Klassenweise Genauigkeit

- Ungleichmäßige Auswirkung von Pruning auf die einzelnen Klassen [HCC+20]
- Gleiches Phänomen bei der Quantisierung

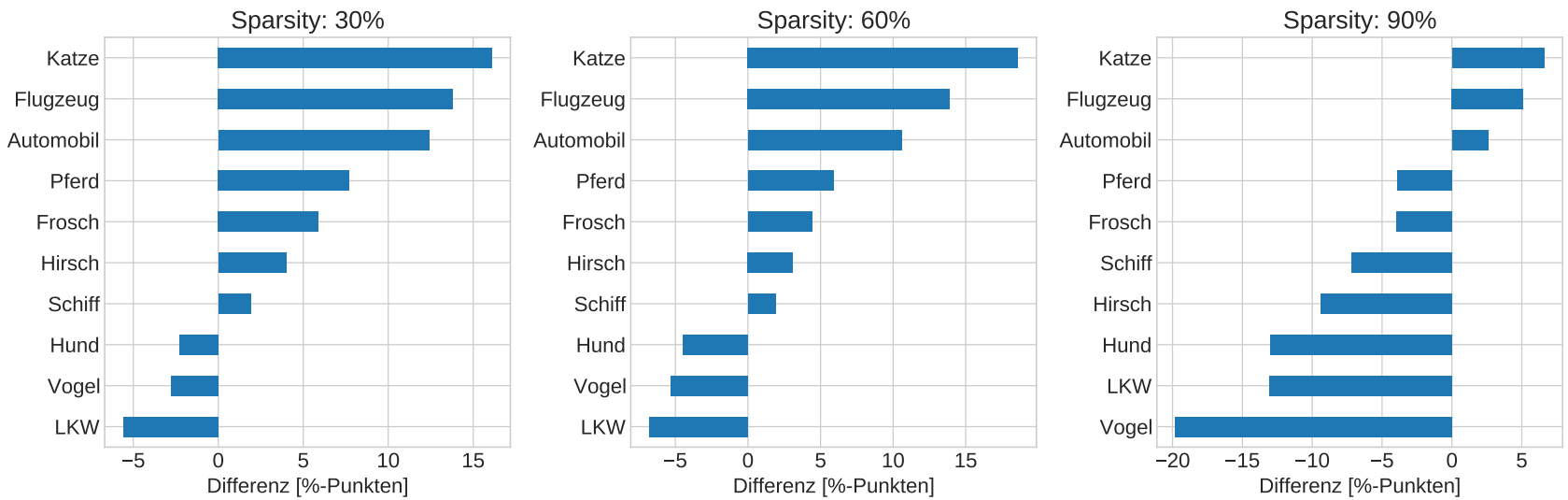

Abbildung: Veränderung der klassenweisen Top-1 Genauigkeit der geprunten MobileNet Modelle in Prozentpunkten

Fazit

Fazit

- Quantisierung sollte in mobilen/eingebetteten Anwendungsbereichen immer in Betracht gezogen werden
- Pruning ist besonders hilfreich wenn Modelle z.B. für eine Übertragung komprimiert werden sollen
- MobileNet Architektur verhält sich robust gegenüber Optimierungen
  - Hohe Kompression mit geringen Genauigkeitsverlusten
  - Deutlich verbesserte Inferenzzeit durch Quantisierung
  - Durch Kombination von Pruning, Quantisierung und gzip lässt sich die Modellgröße auf 0.9 MB verringern
- Klassenweise Genauigkeit sollte bei der Anwendung von Quantisierung/Pruning unbedingt beachtet werden

Literaturverzeichnis

Literaturverzeichnis I

[HCC+20] Sara Hooker, Aaron Courville, Gregory Clark, Yann Dauphin, and Andrea Frome.
What Do Compressed Deep Neural Networks Forget?
arXiv:1911.05248 [cs, stat], July 2020.
arXiv: 1911.05248.

[HSC+19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam.
Searching for MobileNetV3.
arXiv:1905.02244 [cs], November 2019.
arXiv: 1905.02244.

Literaturverzeichnis II

[HZC+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam.
MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.
arXiv:1704.04861 [cs], April 2017.
arXiv: 1704.04861.

[JKC+17] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko.
Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference.
arXiv:1712.05877 [cs, stat], December 2017.
arXiv: 1712.05877.

Literaturverzeichnis III

[SHZ+19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen.
MobileNetV2: Inverted Residuals and Linear Bottlenecks.
arXiv:1801.04381 [cs], March 2019.
arXiv: 1801.04381.

[TL20] Mingxing Tan and Quoc V. Le.
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.
arXiv:1905.11946 [cs, stat], September 2020.
arXiv: 1905.11946.

Literaturverzeichnis IV

[ZG17] Michael Zhu and Suyog Gupta.
To prune, or not to prune: exploring the efficacy of pruning for model compression.
arXiv:1710.01878 [cs, stat], November 2017.
arXiv: 1710.01878.