DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Prüfungsleistung Digitale Bildverarbeitung

Projektarbeit

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Dario Villani

Abgabedatum:	4. Januar 2024
Matrikelnummer:	5805250
Kurs:	TFE21-2
Dozent:	Mark Schutera

Erklärung

gemäß Ziffer 1.1.14 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017 in der Fassung vom 24.07.2023.

Ich versichere hiermit, dass ich meine Projektarbeit mit dem Thema:

Prüfungsleistung Digitale Bildverarbeitung

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Weingarten, den 4. Januar 2024

_____

Dario Villani

Projektarbeit Digitale Bildverarbeitung

In der vorliegenden Projektarbeit soll aufbauend auf dem bereits vorhandenen Netz aus [Sch] ein optimiertes neuronales Netz für das MNIST-Dataset erstellt werden. Dieses weiterentwickelte Netz bildet die Basis für Untersuchungen der Auswirkung diverser Hyperparameter. Dabei wird das Basis-Netz mit verschiedenen Konfigurationen der Hyperparameter trainiert und ausgewertet. Auch der Einfluss von Data Augmentation auf die Performance des Netzes ist Teil der Arbeit. Das daraus folgende Netz mit der besten Performance wird daraufhin detaillierter analysiert, um Schwächen und weiteres Optimierungspotenzial ausfindig zu machen.
Die Dokumentation beschreibt die Vorgehensweise, Hintergründe und nähere Erläuterungen zu dem Code aus dem Notebook (Notebook_Dario_Villani).

Basis-Netz

Um eine solide Basis für weitere Analysen zu schaffen, wird zunächst das vorhandene Netz angepasst und optimiert. Das Basisnetz soll eine Accuracy von 99% erreichen, dabei sollen die Parameter in model.compile und model.fit beibehalten werden.
Um diese Aufgabe zu erfüllen eignet sich ein Convolutional Neural Network (CNN). Convolutional Networks sind spezialiserte Netze, die sich für die Verarbeitung gitterartiger Strukturen, wie beispielsweise 2D Bilderdaten eignen. [GBC16]
Die Architektur des Netzes orientiert sich am LeNet-5, dabei handelt es sich um ein

Convolutional Netz, dass von LeCun et. al [Lec+98] vorgestellt wurde und seitdem ein häufig verwendetes Netz für Ziffererkennung ist. Das Netz besteht aus zwei convolutional layers und zwei pooling layers, welche abwechselnd aufeinander folgen. Danach folgt eine weitere convolutional layer und zwei vollverbundene Schichten. Um die Generalisierung zu verbessern, Overfittig zu reduzieren und dadurch mehr Robustheit zu schaffen [GBC16], wird Dropout in das Netz eingebaut. Die Parameter zur Konfiguration der einzelnen Schichten orientieren sich an einer Architektur aus [KAP20], da diese bei einer Analyse einer ähnlichen Struktur eine gute Performance erzielen konnten.
Durch diesen Aufbau erreicht das Netz nach 10 Epochen eine Accuracy von ca. 99%, allerdings besitzt das Netz auch mehr Parameter als das Ausgangsnetz Marvin. Mit dem vorhandenen Basisnetz kann nun der Einfluss von verschiedenen Hyperparametern und die Performance bei unterschiedlichen Datensätzen (durch data augmentation) des MNIST-Datensatzes untersucht werden.

Hyperparameter Tuning

Hyperparameter Tuning ist der Vorgang die Hyperparameter für das Netz anzupassen und die Kombination zu finden, welche bei dem vorhandenen Datensatz die beste Performance aufweist. In diesem Abschnitt sollen die Einflüsse verschiedener Hyperparameter auf die Performance (gemessen in der Accuracy des Netzes) des neuronalen Netzes analysiert werden. Genauer wird der Einfluss der Hyperparameter Optimizer, Batch-Size und verschiedener Aktivierungs-Funktionen untersucht. Um den Einfluss der Hyperparameter zu analysieren wird der automatisierte Ansatz Grid Search verwendet. Grid Search eignet sich für die Hyperparameter-Optimierung weniger Hyperparameter. Bei der Grid-Search Methodik wird jede Kombination der Hyperparameter als Netz trainiert und anschließend das Netz mit der besten Performance ausgewählt. [GBC16] Bei der Betrachtung der verschiedenen Kombinationen an Hyperparametern wird sich auf eine Auswahl von drei Hyperparametern mit jeweils zwei Instanzen beschränkt, um den Trainingsaufwand einzugrenzen. Dadurch ergeben sich acht mögliche Netzkonfigurationen.

Die Hyperparameter samt ihrer unterschiedlichen Ausführungen werden in einer Matrix notiert. Anhand dieser Matrix werden alle möglichen Kombinationen der verschiedenen Hyperparameter angelegt. Um alle Hyperparameterkombinationen zu vergleichen, wird das Netz mit allen daraus entstandenen Konfigurationen trainiert. Der Trainingsvorgang der unterschiedlichen Netze wird drei mal ausgeführt, um zufällige Ergebnisse einzugrenzen. Eine häufigere Ausführung wäre sinnvoll, erhöht jedoch die Zeit für den Trainingsvorgang enorm. Die Accuracy aller Netze wird über die drei Durchgänge gemittelt. Anhand der gemittelten Accuracies wird das Netz ausgewählt, dass für die vorhandenen Trainingsvorgänge die beste Performance aufweist.

Anhand der durchgeführten Trainings ergeben sich acht Mittelwerte der Genauigkeiten, siehe Abbildung 1. Die drei besten Kombinationen sind unten nach ihrer Performance mit den jeweiligen Hyperparametern aufgelistet. Das Netz mit der besten performance wird im Code und im weiteren Text mit best_model bezeichnet. Im Anhang befindet sich eine Auflistung aller acht Konfigurationen mit ihrer Performance.

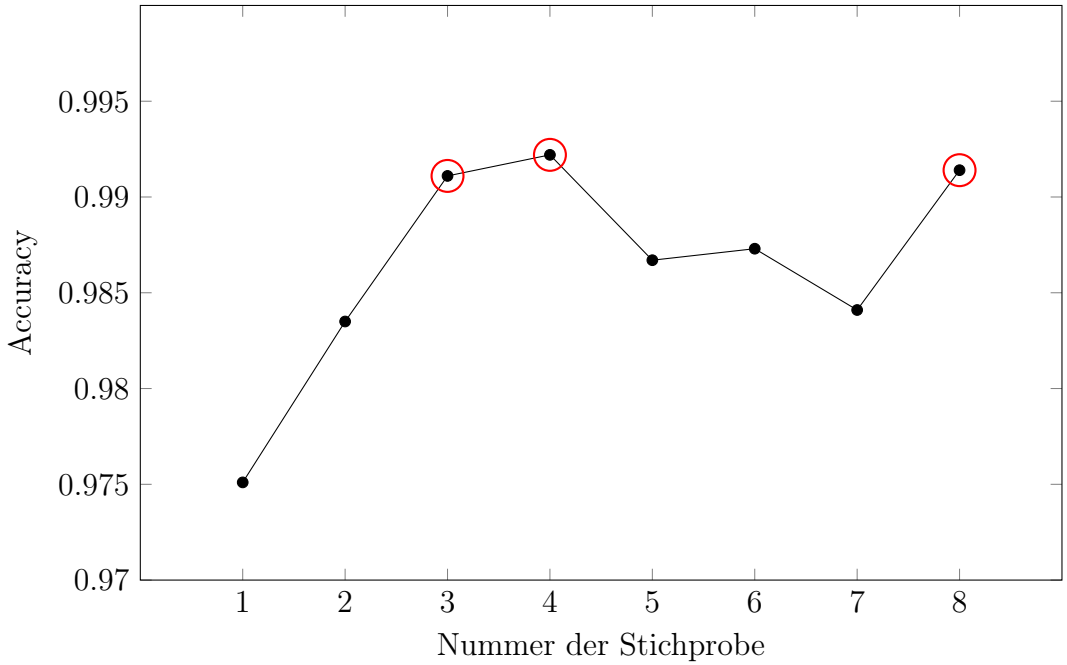

Abbildung 1: Genauigkeiten der Netze mit unterschiedlichen Hyperparameterkonfigurationen (Top 3 mit rotem Kreis hervorgehoben)

Tabelle 1: Top 3 Parameterkombinationen basierend auf durchschnittlicher Genauigkeit

Platz	Accuracy	Optimizer	Activation	Batch Size
1	99.22%	adam	relu	64
2	99.14%	rmsprop	relu	64
3	99.11%	adam	relu	8

Anhand der Ergebnisse der durchgeführen Analyse können für die vorliegenden Daten folgende Aussage getroffen werden:

- Ein Netz mit der Aktivierungsfunktion relu erreicht eine höhere Accuracy als ein Netz mit der Aktivierungsfunktion tanh.
- Eine größere Batch-Size erhöht die Performance des Netzes.
- Der verwendete Optimizer hat einen geringeren Einfluss auf die Performance als die anderen Parameter des betrachteten Netzes.

Dataset augmentation

Nachdem durch das Hyperparameter-Tuning aus den betrachteten Netzen das beste Netz ausgewählt wurde, wird im nächsten Abschnitt der Einfluss von Data Augmentation auf die Performance des Netzes untersucht. Data Augmentation ist eine Methode den Datensatz der Trainingsdaten durch Veränderungen der Daten in ihrer Dimension (z.B. Verschiebungen oder Rotationen) zu vergrößern. Dadurch kann beim Trainingsvorgang eine bessere Generalisierung und dadurch eine bessere Performance bei den Testdaten verursacht werden [GBC16]. Dafür wird das Netz zuerst mit dem normalen Dataset trainiert. Danach wird das Dataset mittels data augmentation verändert und das Netz wird erneut trainiert. Die data augmentation wird mit dem ImageDataGenerator umgesetzt.

Die Trainingsbilder werden mit einem maximalen Rotationsbereich von 20° rotiert um ein breiteres Dataset zu erstellen. Wichtig ist hier die Rotation nicht zu groß zu wählen, da es ansonsten zu Verfälschungen im Dataset kommen kann (eine um 180° rotierte 6 entspricht einer 9). Deshalb wird der Rotationsbereich auf 20° beschränkt. Anhand der beiden Durchläufe sieht man, dass der Trainingsvorgang mit dem durch data augmentation veränderten Dataset eine bessere Accuracy hervorbringt (99.13% statt 99.09%).

best_model analysieren

In diesem Abschnitt soll das best_model detaillierter analysiert werden. Das Netz hat bei erneuter Ausführung bei den Testdaten einen Loss von 0.0298 und eine Accuracy von 0.9925. Anhand der Lernkurve sieht man, dass das Netz auf den Validierungsdaten anfangs eine bessere Accuracy hat und sich dann während des Trainingsvorgangs der Trainings-Accuracy annähert. Die Abbildungen auf die sich in diesem Kapitel bezogen werden, befinden sich im Anhang.
Die Confusion-Matrix zeigt an, wie oft die Ziffern vom Netz korrekt vorhergesagt wurden und wie oft Verwechslungen hervorgegangen sind. Durch Analyse der Confusion-Matrix kann man folgende Aussagen für das betrachtete Netz treffen:

- Die Ziffer 1 wird am besten erkannt und am seltesten falsch vorhergesagt.
- Die Ziffer 9 wird am schlechtesten vorhergesagt.
- Die häufigste Verwechslung von zwei Zahlen ist die Vorhersage einer 9 als eine 7, welche insgesamt acht mal vorkommt.

Zusätzlich werden die Ziffern geplottet, die das Netz falsch zugeordnet hat. Bei einer ersten Betrachtung sieht man, dass viele dieser Ziffern recht ungenau ausgeschrieben sind, also häufig nur schlecht ausgeprägte Erkennungsmerkmale haben oder sehr stark

rotiert sind. Die falsch vorhergesagten Ziffern sind in dem vorhandenen Notebook abgebildet.

Zusammenfassung

Zusammengefasst konnte durch die Optimierung des vorhandenen Netzes ein Basis-Netz mit einer Accuracy von 99% geschaffen werden. Ausgehend von diesem Basis-Netz konnte aus einer Auswahl verschiedener Hyperparameterkonfigurationen durch Grid Search, die nach den Testdurchgängen beste Kombination, mit der besten Performance ausgewählt werden. Mit Dataset Augmentation konnte ein neuer Trainingsdatensatz geschaffen werden, mit welchem das Netz bei den vorhandenen Durchläufen eine bessere Performance bei dem Testdatensatz erreicht. Im letzten Schritt wurde das Netz detaillierter analysiert und es konnten Stärken und Schwächen bei der Ziffererkennung ausgemacht werden. Trotz der vorgenommenen Optimierungen sieht man anhand der geplotteten Ziffern, die das Netz nicht korrekt erkennt, dass das Netz noch sehr viel Verbesserungspotenzial hat. Mögliche Ansätze und Schritte für eine mögliche Weiterentwicklung sind:

- Veränderungen in der Architektur, also mehr oder weniger Layer einbauen (je nachdem, ob der Fokus auf einer hohen Accuracy oder der Reduzierung der Parameter des Netzes liegt) und die Parameter der einzelnen Schichten verändern, um dadurch die Performance des Netzes zu verbessern.

- Weitere Hyperparameter (Epochen-Anzahl, learn-rate...) untersuchen und eine größere Anzahl an Konfigurationen vergleichen. Dabei kann auch ein anderer Hyperparameter-Optimierungs-Ansatz, wie der in [GBC16] vorgestellte Ansatz (Model-Based Hyperparameter Optimization) verwendet werden.

- Auch die Ausweitung der Data-Augmentation birgt weiteres Potenzial, in dem der Trainingsdatensatz weiter vergrößert wird und die Ziffern auch in anderen Dimensionen (z.B.Verschiebungen oder Verzeichnungen) verändert werden.

Literaturverzeichnis

[GBC16] Ian J. Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[KAP20] Shivam S. Kadam, A. C. Adamuthe und Ashwini Patil. „CNN Model for Image Classification on MNIST and Fashion-MNIST Dataset“. In: *Journal of scientific research* (2020). URL: https://api.semanticscholar.org/CorpusID:226435631.

[Lec+98] Y. Lecun u. a. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.

[Sch] Mark Schutera. *DeepDive*. https://github.com/schutera/DeepDIve.

Anhang

Tabelle 1: Genauigkeitswerte der verschiedenen Hyperparameterkonfigurationen

Platz	Accuracy	Optimizer	Activation	Batch Size
1	99.22%	adam	relu	64
2	99.14%	rmsprop	relu	64
3	99.11%	adam	relu	8
4	98.73%	rmsprop	tanh	64
5	98.67%	rmsprop	tanh	8
6	98.53%	adam	tanh	64
7	98.41%	rmsprop	relu	8
8	97.51%	adam	tanh	8

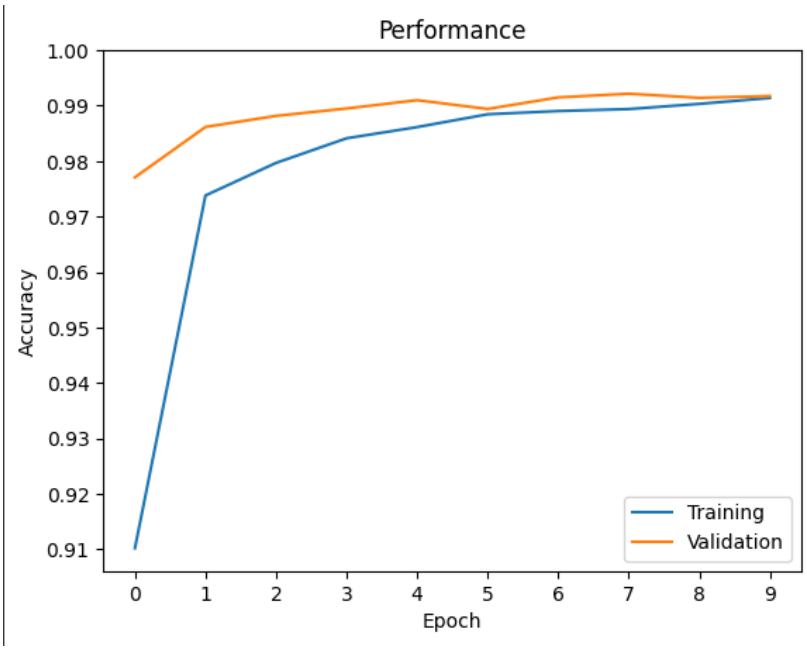

Abbildung 1: Lernkurve best_model

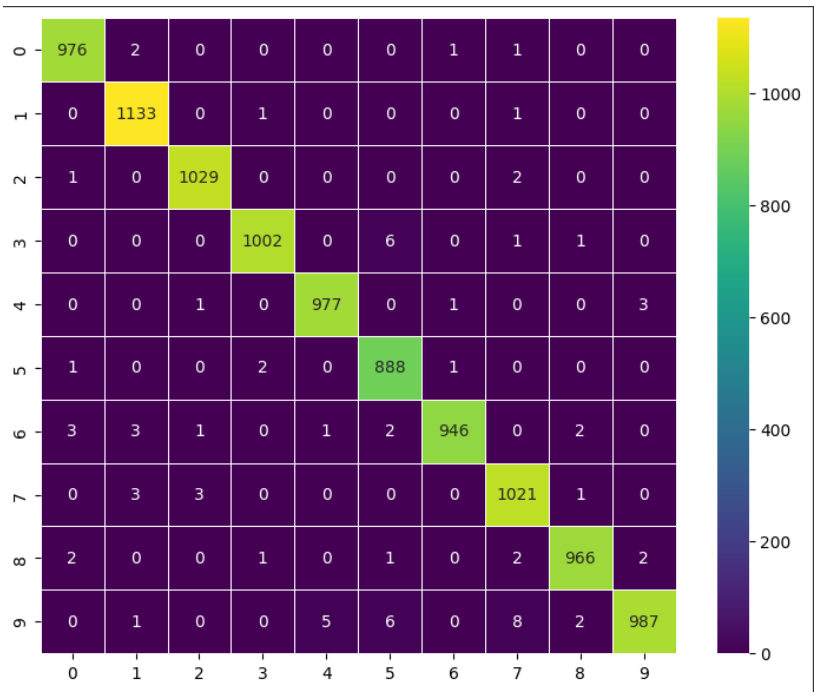

Abbildung 2: Confusion Matrix best_model