DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Optimierung des neuronalen Netzes

Dokumentation

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Moritz Kramer

Abgabedatum:	5. Januar 2024
Matrikelnummer:	6317166
Kurs:	TFE21-2
Dozent der Dualen Hochschule:	Mark Schutera

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des Moduls Bildverarbeitung und Digitale Systeme ist ein neuronales Netz, welches Ziffern von 0-9 erkennen soll, zu optimieren. Hierbei wird das TensorFlow-Framework, welches durch die Keras-Bibliothek unterstützt wird, verwendet. Die Aufgabe umfasst das Entwerfen, Trainieren und anschließende Testen des neuronalen Netzes. Trainiert wird mit 60 000 handgeschriebenen Ziffern aus dem MNIST-Datensatz im Format von 28x28 Pixeln. Mit weiteren 10 000 Ziffern wird das Netz getestet. In der folgenden Dokumentation wird zunächst auf die vorliegende "Basisversion"des neuronalen Netzes und dessen Ergebnisse eingegangen, bevor Ziele bezüglich einer verbesserten Genauigkeit (accuracy) des Netzes formuliert werden. Weiterhin werden die zur Verbesserung der Genauigkeit angewandten Methoden aufgezeigt und die Ergebnisse des optimierten Netzes dargestellt.

2 Basisversion und Zielsetzung

Basisversion des Netzes

Die in der Basisversion implementierte Netzarchitektur (marvin) stellt ein sequentielles Modell mit drei verschiedenen Layern dar. Lediglich mit einem Flatten-, Dense- und Dropout-Layer. Diese Architektur soll als eine Grundlage zur weiteren Optimierung des Netzes dienen. Das sogenannte Multilayer Perceptron (MLP), das aus einer Eingabeschicht, einer versteckten Schicht und einer Ausgabeschicht besteht, sieht dabei folgendermaßen aus:

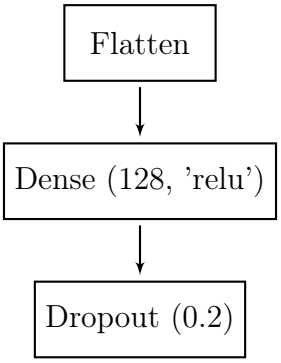

Abbildung 2.1: MLP-Architektur

Das Modell trainiert dabei 50 Epochen mit einer Batch-Size von 516.

Zielsetzung

In der optimierten Version des neuronalen Netzes soll mindestens eine Genauigkeit von $99,5\,\%$ erreicht werden. Dabei gibt es keine Anforderungen mit wie wenigen Parametern oder Labels gearbeitet werden.

3 Umsetzung

Dieses Kapitel befasst sich mit den angewandten Methoden, um die anfangs definierten Zielsetzungen zu erfüllen. Das Ziel, eine Genauigkeit von mindestens $99,5\ \%$, erfordert dabei Anpassungen der Modellarchitektur des vorliegenden neuronalen Netzes. Aktuelle Untersuchungen bezüglich der Bilderkennung zeigen, dass vor allem durch sogenannte Convolutional Neural Networks (CNNs) Erfolge zu verzeichnen sind [Sco+23]. Ein Convolutional Layer dient zur Erkennung lokaler Muster in Bildern und verschiebt einen Filter über das gesamte Bild, um aus diesem Merkmale zu extrahieren. Dabei ist ein Neuron eines Layers nicht mit jedem Neuron des vorherigen Layers verbunden. Die gemeinsame Nutzung der Gewichtung (shared weights) des Kernel-Filters wird dazu eingesetzt, identische Merkmale an unterschiedlichen Positionen im Bild zu erkennen. Gleichzeitig wird das Netz resistenter gegen Transformationen des Bildes gemacht, welche die Position des Merkmals innerhalb des Bilds verändern. Dies lässt die Performance des neuronalen Netzes steigern währenddessen die zu erlernenden Parameter reduziert werden können [Mei19]. Ein weiterer Bestandteil von Convolutional Neural Networks sind Pooling Layer, die die Dimensionalität Feature-Maps zu reduzieren. Dabei werden häufig sogenannte MaxPooling2D-Layer implementiert, welche den maximalen Wert aus einem Bereich der Feature-Map extrahieren. Um Overfitting zu reduzieren werden mehrere Dropout-Layer implementiert. Zum Schluss des CNNs werden die räumlichen Merkmale in Form von 2D-Matrizen durch ein Flatten-Layer in eindimensionale Vektoren umgewandelt, welche anschließend als Eingabe des vollständig verbundenen Dense-Layers dienen. Das Dense-Layer erzeugt schließlich die endgültige Wahrscheinlichkeitsverteilung der Klassen.
Nach [Tha22] kann die Dimensionsreduktion anstatt durch die MaxPooling2D-Layer auch durch Conv2D-Layer mit einer Schrittlänge (strides) größer als 1 erreicht werden, was lernfähige Pooling-Layer ermöglichen soll. Dabei können Conv2D-Layer mit Strides genutzt werden, um die räumliche Abmessung der Daten zu verringern und gleichzeitig relevante Merkmale zu extrahieren. Eine Implementation solcher lernfähiger Pooling-Layer führen jedoch zu keiner verbesserten Genauigkeit des vorliegenden neuronalen Netzes. Weiterhin wird versucht, durch Data Augmentation mittels zufälligem Drehen, Skalieren und Verschieben des Trainingsdatensatzes eine deutlich größere Anzahl an Trainingsdaten zu generieren, um schlussendlich eine höhere Genauigkeit zu

erzielen. Die Methode der Data Augmentation findet aber aufgrund von nicht behebbaren Implementationsfehlern in diesem CNN keine Anwendung. Die finale Architektur sieht folgendermaßen aus:

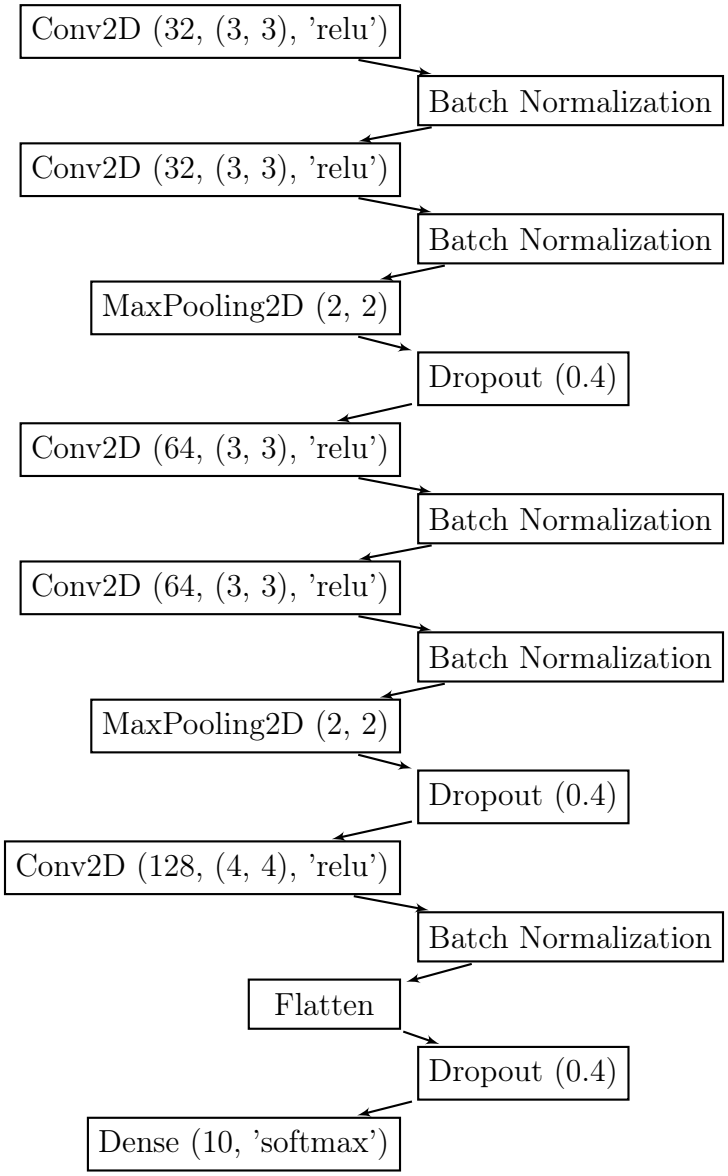

Abbildung 3.1: CNN-Architektur

Weiterhin werden die Hyperparameter Epochen von 50 auf 15 und Batch-Size von 516 auf 32 verändert. Grund dafür sind nebe einer verbesserten Stabilität und Leistung vor allem sehr lange resultierende Laufzeiten des CNNs bei zunehmenden Epochen.

4 Ergebnis und Auswertung

In Abbildung 4.1 sind die Ergebnisse der Trainingsläufe und Testläufe beider neuronalen Netze dargestellt. Dabei werden jeweils die Trainings- und Test-Accuracys über die Epochen geplottet. Vergleicht man die beiden Darstellungen wird deutlich, dass das Marvin Netz erheblichere Unterschiede zwischen der Trainings- und Test-Accuracy aufweist. Die Trainings-Accuracy steigt nicht über 80 %, die finale Test-Accuracy beträgt 92, 52 %. Beim optimierten Loki Netz hingegen nähert sich die Trainings-Accuracy bei zunehmenden Epochen der Test-Accuracy, welche final 99, 60 % erreicht, weiter an. Somit ist die anfangs definierte Zielsetzung von einer Test-Accuracy mit mindestens 99, 50 % erreicht. Wie auch in den Jupyter Notebooks zu finden, bieten Confusion Matrizen einen

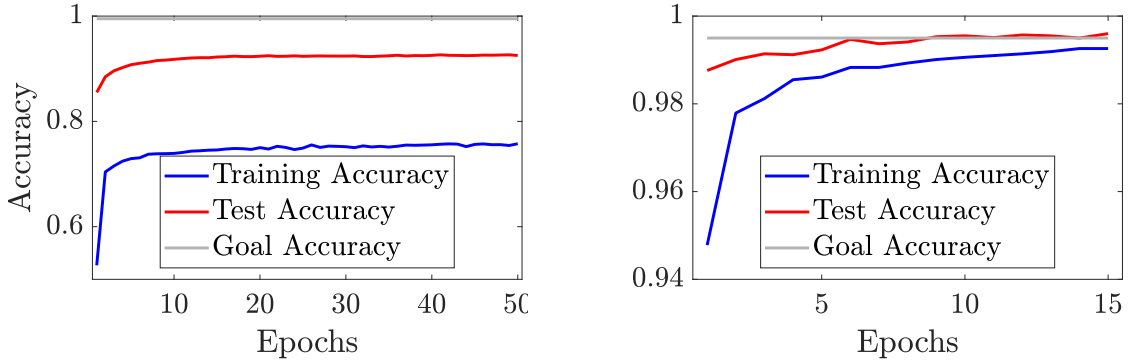

Abbildung 4.1: Links: Accuracy Basisversion Marvin, Rechts: Accuracy optimierte Version Loki

weiteren Überblick über die Leistung des neuronalen Netzes im Bezug auf die Genauigkeit. Betrachtet man die Confusions Matrix beider Netze (Abb.4.2) lässt sich aus diesen herauslesen, dass die einzelnen Ziffern deutlich besser vorhergesagt werden können. Die maximale Anzahl der Verwechslung zweier Ziffern, bei Marvin 43-mal, konnte erheblich beim Loki Netz auf 6-mal reduziert werden.
Das Loki Netz bietet dabei weiterhin einen Anhaltspunkt zur fortführenden Optimierung der Genauigkeit bei der Vorhersage handgeschriebener Ziffern. Die Methode der Data Augmentation kann weiter verfolgt werden, nachdem diese im Rahmen dieser Arbeit nicht erfolgreich implementiert werden kann. Durch weitere Optimierungen der ver-

wendeten Layer, Hyperparameter und Hinzufügen neuer Layer können weitere Untersuchungen zur Verbesserung der Stabilität und Leistung des neuronalen Netzes angestellt werden.

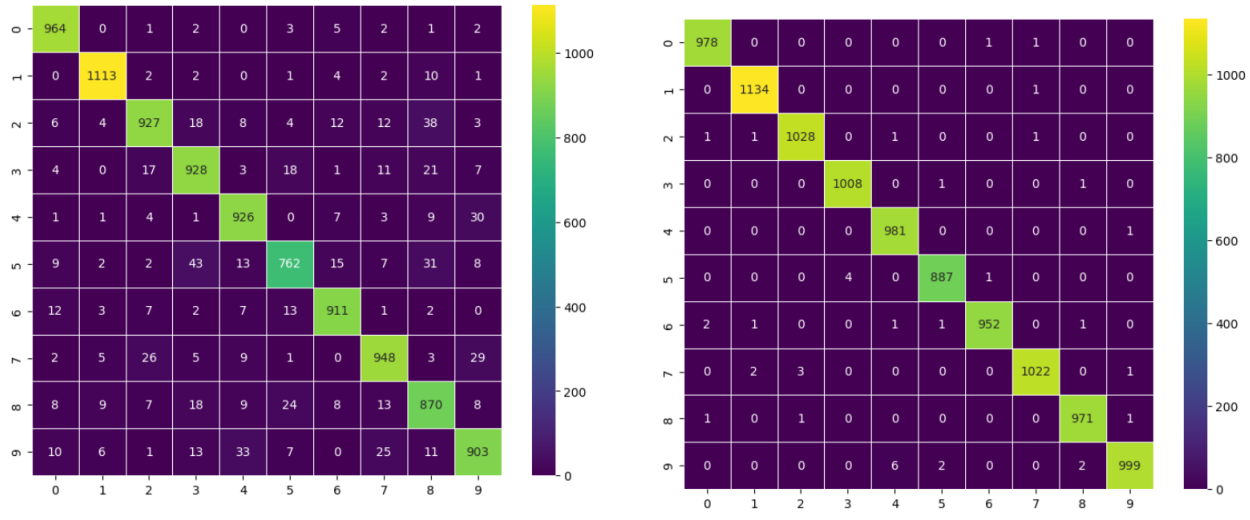

Abbildung 4.2: Links: Confusion Matrix Basisversion Marvin, Rechts: Confusion Matrix optimierte Version Loki

Literatur

[Mei19] Nadine Meißler. *Visualisierung der internen Prozesse Künstlicher Neuronaler Netze in Virtual Reality*. Hrsg. von Hochschule Düsseldorf. 2019.

[Sco+23] Riccardo Scodellaro u. a. *Training Convolutional Neural Networks with the Forward-Forward Algorithm*. Hrsg. von Journal of Machine Leraning Research. 2023.

[Tha22] Hari Thapliyaal. *MNIST with 99.75% Accuracy*. Hrsg. von kaggle. 2022. URL: `https://www.kaggle.com/code/harithapliyal/mnist-with-99-75-accuracy`.

Abbildungsverzeichnis