DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Digitale Bildverarbeitung und Mustererkennung

Dokumentation des Programmentwurfs

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Marvin Auwärter

Erstellungsdatum:	2. Januar 2024
Bearbeitungszeitraum:	25.10.2023 - 05.01.2024
Matrikelnummer:	8463154
Kurs:	TFE21-2
Gutachter der Dualen Hochschule:	Mark Schutera

1 Festlegung der Ziele für die Performance des Netzes

Dieses Dokument ist eine Ergänzung zum ebenfalls abgegebenen Programmentwurf. Sie soll Ideen bei der Verbesserung des Modells „Marvin“ darstellen und geht auf die verwendeten Methoden und Ergebnisse ein. Bevor die Umsetzung der Software erläutert wird, sind in diesem Abschnitt die zu Beginn gesetzten Ziele aufgelistet. Das beschriebene Modell soll eine gute Balance zwischen einer hohen Genauigkeit, sowie wenigen Parametern und Labeln zu erreichen. Dabei wurden folgende Ziele gesetzt, um eine Vorstellung des zu Erreichenden zu erhalten: Die Validation Accuracy soll einen Wert über 98% erreichen, die Anzahl der Parameter soll unter 4000 liegen, während die Anzahl der Trainingsdaten (Label) auf 4000 festgelegt wurden.

2 Schichten des Netzes

Meine Architektur zur Zifferererkennung besteht aus neun Layern. Als Netztyp wurde ein Convolutional Neural Network (CNN) gewählt, da diese Art an Netzen aufgrund ihres Aufbaus mit verschiedenen Filtern in jeder Schicht gut für die Bilderkennung geeignet ist. Sie eignet sich damit besonders für die Mustererkennung beim MNIST-Datensatz. [Mat23] Die Hyperparameter der einzelnen Layer wurden, aufgrund der geringen Erfahrung mit neuronalen Netzen, in den meisten Fällen durch Experimentieren bestimmt.

1. Das Modell beginnt mit einem Convolutional-Layer („Conv2D“), um Muster und Merkmale des Labels zu erkennen. Dabei werden Faltungen auf dem Eingabebild durchgeführt. Diese Schicht hat 16 Filter der Größe 3x3. Die genutzte Aktivierungsfunktion ist „ReLU (Rectified Linear Unit)“. Des Weiteren ist in dieser Schicht die Größe der Eingabebilder festgeschrieben. Diese liegt bei 28x28 Pixeln.

2. An dieser Stelle ist ein Batch-Normalization Layer eingebaut. Diese Schicht normalisiert die Aktivierung der vorherigen Schicht. Dadurch wird das Training stabilisiert.

3. Es folgt ein „Max Pooling“-Layer mit einem Fenster der Größe 2x2. Dies reduziert die Dimensionen und somit die Anzahl der Parameter. Außerdem konzentriert sich das Modell dadurch nur auf die relevantesten Informationen.

4. Um die Validation Accuracy nachhaltig zu erhöhen, wird derselbe Ablauf mit anderen Hyperparametern wiederholt. Dieser „Conv2D“-Layer hat 8 Filter der Größe 3x3. Die genutzte Aktivierungsfunktion ist wieder „ReLU“.

5. Auch an dieser Stelle wird zur Stabilisierung des Trainings wieder ein Batch-Normalization Layer eingebaut.

6. Daneben folgt wieder ein „Max Pooling“-Layer mit einem Fenster der Größe 2x2.

7. Im nächsten Schritt wird ein „Dropout“-Layer eingebaut. Dabei werden zufällig 50 % der Neuronen während des Trainings deaktiviert, um Overfitting bestmöglich einzuschränken. Mit dem Overfitting ist das Auswendiglernen der Trainingsdaten gemeint. Dieses wirkt sich negativ auf das Erkennen der Testdaten aus.

8. An dieser Stelle werden die Daten im „Flatten“-Layer von einer Feature Map in einen eindimensionalen Vektor umgewandelt, um sie als Eingabe für den „Dense“-Layer verwenden zu können.

9. In diesem „Dense“-Layer werden die Bilder klassifiziert und zu den möglichen Ergebnissen, bei MNIST den Ziffern von 0 bis 9, zugeordnet. Zum Schluss wird dann die wahrscheinlichste Ziffer als Antwort des Netzes ausgegeben.

3 Aufbau des Trainings

Beim Testen zu Beginn zum experimentellen Festlegen der Hyperparameter wurde nur mit 30 Epochen trainiert, um die Folgen der Änderungen in kurzer Zeit erkennen zu können. Bei der Verfeinerung bis zum finalen Modell wurde dann ein Training von bis zu 500 Epochen benutzt, um das Maximum des Netzes auszureizen und zu sehen, wie lange das Netz sich noch verbessern kann bis es einen Grenzwert erreicht. Falls das Modell diesen erreicht hat wird zur Vermeidung von unnötiger Rechenzeit der „Early Stopping Callback“ mit einer Patience von 50 benutzt. Das bedeutet, dass das Training beendet wird, sobald die Validation Accuracy in 50 aufeinanderfolgenden Epochen nicht mehr verbessert hat. Duch diese Einschränkung ist es in keinem Trainingsdurchlauf passiert, dass die 500 Epochen wirklich auch durchlaufen wurden. Mit „Shuffle“ wurde sichergestellt, dass die Trainingsdaten vor jeder Epoche gemischt werden. Dies hilft bei der Generalisierung des Modells und verhindert wie der Dropout Layer das Auswendiglernen der Trainingsdaten. Daneben wurde mit „Validation Split“ein Teil des Datensatzes als Validierungsdatensatz verwendet, um bereits während des Trainings das Risiko von Overfitting weiter zu verringern. Für die „Batch Size“ wurde mit 32 ein dafür gewöhnlicher Wert festgelegt.

4 Ergebnisse des Trainings

Das Ergebnis des Trainings lautet:
Bei einer Anzahl von 3426 Parametern und 4000 Trainingsdaten wurde eine Validation Accuracy von 98,12% erreicht.

```
Epoch 90/500
125/125 [==============================] - 1s 9ms/step - loss: 0.0630 - accuracy: 0.9760 - val_loss: 0.0675 - val_accuracy: 0.9812
```

Abbildung 1: Ergebnis der besten Epoche des Trainings

In allen drei Kategorien wurden die Ziele erreicht. Dennoch macht das Modell ein paar kleinere Fehler. Die größten Probleme liegen bei der Ziffer 7. Dabei wird häufig eine 2 ausgegeben. Daneben wird auch die Ziffer 8 oft als 9 erkannt. Am besten erkennt das Modell die Ziffer 1. In 10.000 Testlabels wurde diese Ziffer nur 10 mal nicht erkannt. Die folgende Matrix fasst die Ergebnisse zusammen.

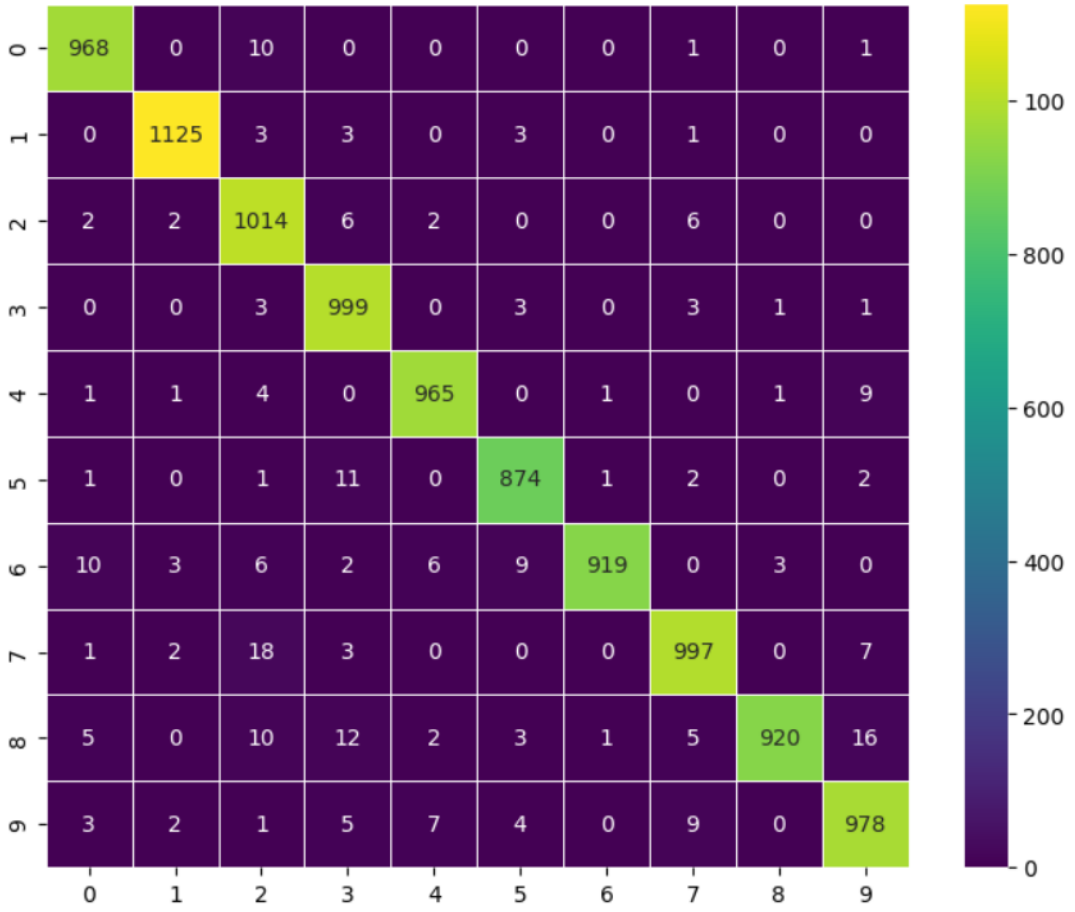

Abbildung 2: Anzeige der Ergebnisse bei den verschiedenen Ziffern

Der Grund für die Schwierigkeiten von „Marvin“ bei den Ziffer 7 und 8 ist auf die Ähnlichkeit mancher Bilder anderer Ziffern in MNIST zurückzuführen. Für eine bessere Genauigkeit könnte man mehr als 4000 Trainingsdaten verwenden oder ein komplexeres Modell mit mehr Layern und Parametern erschaffen.

Literatur

[Mat23] Mathworks. *Was ist ein Convolutional Neural Network?* 2023. URL: https://de.mathworks.com/discovery/convolutional-neural-network-matlab.html#:~:text=CNNs%20sind%20besonders%20n%C3%BCtzlich%2C%20um,und%20Signaldaten%20sehr%20effektiv%20sein..