Bericht zur Implementierung eines neuronalen Netzwerks zur MNIST-Klassifikation

Luna Schätzle

December 17, 2024

Aufgabenbeschreibung

In Aufgabe 5 haben wir ein neuronales Netzwerk implementiert, um die MNIST-Datenbank zu klassifizieren. Die MNIST-Datenbank besteht aus 60.000 Trainings- und 10.000 Testbildern. Jedes Bild umfasst 28×28 Pixel und stellt eine handgeschriebene Ziffer von 0 bis 9 dar. Das Ziel war es, ein neuronales Netzwerk zu erstellen, das diese Ziffern zuverlässig klassifizieren kann.

Die Aufgabe erforderte die Implementierung eines vollständigen Workflows, einschlieSSlich der Datenvorverarbeitung, dem Design der Netzwerkarchitektur, dem Trainieren des Modells, der Leistungsevaluierung und der Verwendung des trainierten Modells für Vorhersagen.

Datenladen

Der erste Schritt bestand darin, den MNIST-Datensatz für das Modell zu laden. Die Funktion `load_data()` wurde implementiert, um den Datensatz zu laden und die Trainings- sowie Testdaten zurückzugeben.

Code-Implementierung

```
1 import numpy as np
2 import tensorflow as tf
3 from tensorflow.keras.datasets import mnist
4
```

```
# MNIST-Datensatz laden
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Datensatzdimensionen ausgeben
print(f"x_train shape: {x_train.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_test shape: {y_test.shape}")
```

Datensatzanalyse

Der MNIST-Datensatz besteht aus Graustufenbildern, bei denen die Pixelwerte zwischen 0 und 255 liegen. Um die Verarbeitung für das neuronale Netzwerk zu vereinfachen, wurden die Pixelwerte auf den Bereich [0, 1] normalisiert, indem sie durch 255 geteilt wurden. Diese Normalisierung stellt sicher, dass das Netzwerk effizient arbeiten kann, ohne auf Probleme durch groSSe Eingabewerte zu stoSSen.

Visualisierung

Ein Beispielbild aus dem Trainingsdatensatz wird unten dargestellt, zusammen mit Beispielbildern aus jeder Ziffernkategorie:

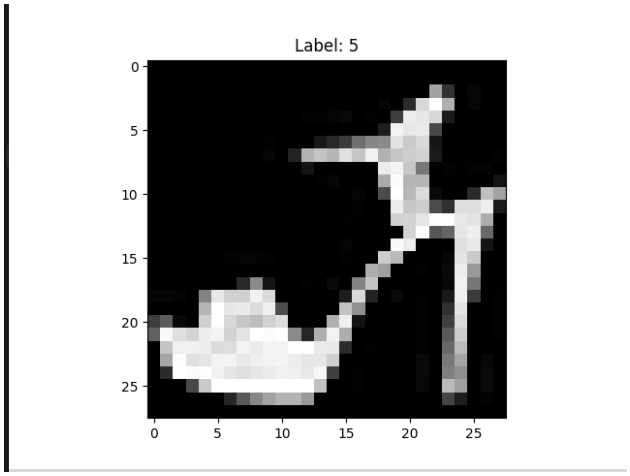

Figure 1: Beispielbild aus dem Trainingsdatensatz

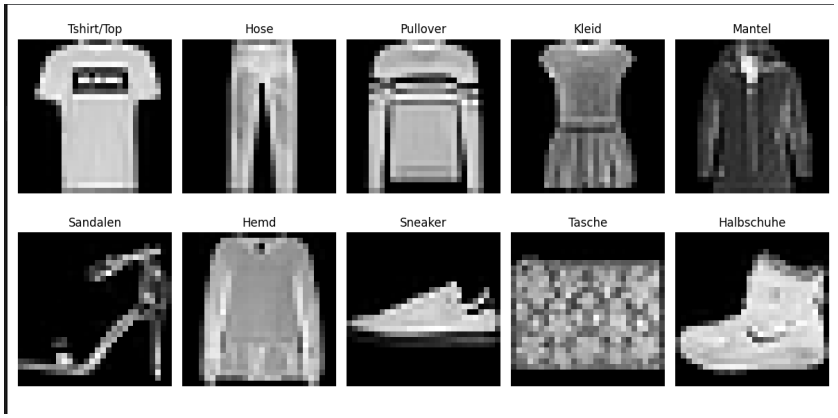

Figure 2: Beispielbilder aus allen Ziffernkategorien

Modellarchitektur

Das neuronale Netzwerk wurde mithilfe von Keras implementiert. Die Architektur umfasst:

- Eine Eingabeschicht für 28×28 aufbereitete Pixel.
- Sechs dichte (vollständig verbundene) Schichten mit ReLU-Aktivierungsfunktion.
- Eine finale Ausgabeschicht mit Softmax-Aktivierung für 10 Klassen.

Code-Implementierung

```
from tensorflow.keras import models, layers

model = models.Sequential([
    layers.Input(shape=(28 * 28,)),
    layers.Dense(512, activation='relu'),
    layers.Dense(256, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(24, activation='relu'),
    layers.Dense(10, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

Modelzusammenfassung

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	401,920
dense_1 (Dense)	(None, 256)	131,328
dense_2 (Dense)	(None, 128)	32,896
dense_3 (Dense)	(None, 24)	3,096
dense_4 (Dense)	(None, 10)	250
dense_5 (Dense)	(None, 10)	110

Total params: 569,600 (2.17 MB)
Trainable params: 569,600 (2.17 MB)
Non-trainable params: 0 (0.00 B)

Trainingsprozess

Das Modell wurde über 40 Epochen mit dem Adam-Optimierer und der Verlustfunktion "Sparse Categorical Cross-Entropy" trainiert. Genauigkeit und Verlust während Training und Validierung wurden überwacht.

Trainingsausgabe

Der Trainingsprozess zeigte schrittweise Verbesserungen der Genauigkeit und Reduzierungen des Verlusts, wie in Abbildung 3 dargestellt.

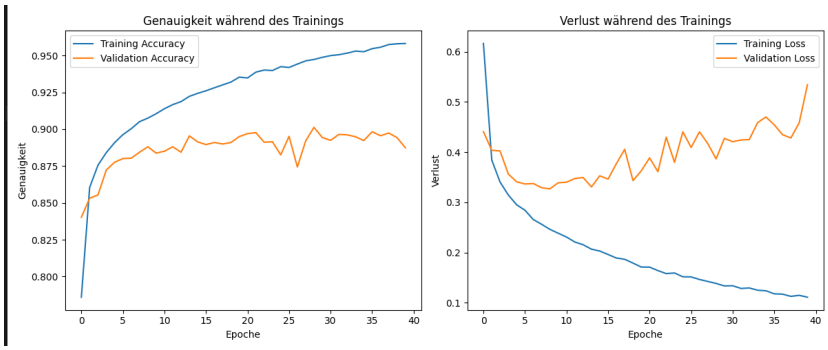

Figure 3: Trainingsgenauigkeit und Verlust über die Epochen

Evaluation

Nach dem Training wurde das Modell auf dem Testdatensatz evaluiert. Die Ergebnisse waren:

- Testverlust: 0.5342
- Testgenauigkeit: 88,74%

Fazit

Dieses Projekt hat erfolgreich ein neuronales Netzwerk zur MNIST-Ziffernklassifikation implementiert. Mit einer Testgenauigkeit von 88,74% zeigt das Modell eine solide Leistung.

Source Code

Der vollständige Quellcode für dieses Projekt ist auf GitHub unter folgendem Link verfügbar: https://github.com/Luna-Schaetzle/INFI_Informations_Systeme/tree/main/Aufgabe_5

References

1. MNIST Dataset: http://yann.lecun.com/exdb/mnist/
2. MNIST Fashion Dataset: https://github.com/zalandoresearch/fashion-mnist?tab=readme-ov-file
3. Keras Documentation: https://keras.io/
4. TensorFlow Documentation: https://www.tensorflow.org/
5. GitHub Repository: https://github.com/Luna-Schaetzle/INFI_Informations_Systeme/tree/main/Aufgabe_5
6. Python Documentation: https://docs.python.org/3/

Contact Information

For any questions or clarifications, please contact me at
Luna P. I. Schätzle
Email: luna.schaetzle@gmail.com
Website: https://luna-schaetzle.xyz
GitHub: https://github.com/Luna-Schaetzle