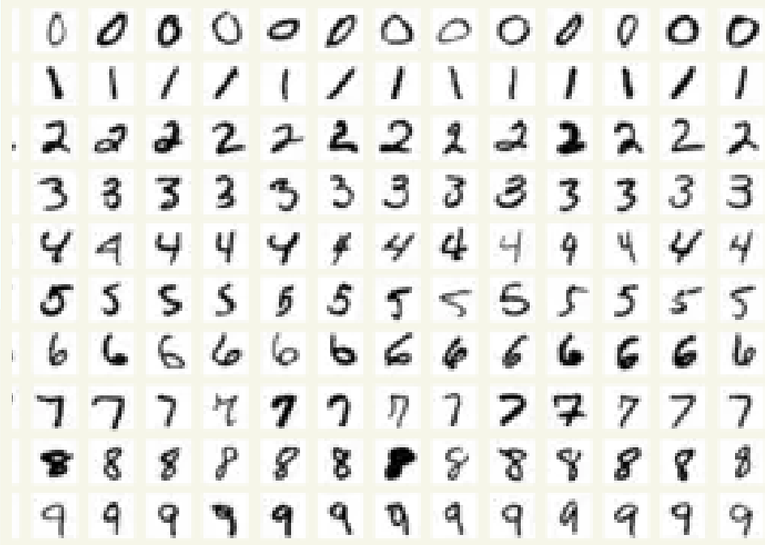
DENİZ OZDEMİRLİ

KLASSIFIZIERUNG DER HANDGESCHRIEBENEN ZIFFERN MIT MULTILAYER PERZEPTRON



Der MNIST Datensatz enthält 70.000 Bilder von handgeschriebenen Ziffern. In dieser Forschung werden wir mehrschichtige Perceptrons mit verschiedenen Konfigurationen und Aktivierungsfunktionen erstellen, um handgeschriebene Ziffern in diesem Datensatz zu klassifizieren.

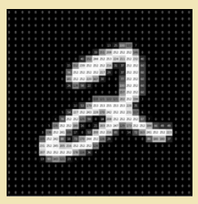
DATENSATZ

Die MNIST-Datenbank ist ein Datensatz mit handgeschriebenen Ziffern. Sie enthält 60.000 Trainingsdaten und 10.000 Testdaten. Jedes Bild besteht aus 28x28 Pixeln, von denen jedes einen Wert von 0-255 mit seinem Graustufenwert enthält.

Die Trainings- und Testlabels sind eindimensional. Sie liegen im Zahlenformat von 0 bis 9 vor. Dies wird als spärliche skalare Darstellung der Labels bezeichnet. Sie ist nicht für die Ausgabeschicht des neuronalen Netzes geeignet, die Wahrscheinlichkeiten pro Klasse ausgibt.

SHAPES BEFORE

Training images: (60000, 28, 28)
Training labels: (60000,)
Test images: (10000, 28, 28)
Test Labels: (10000,)



PREPROCESS

Bevor ich diese Daten in mein Netzwerk einspeise, müssen die Beschriftungen mit einem Hotcode codiert und die Bilder umgestaltet und normalisiert werden.

Das umgeformte Bild ist ein eindimensionales Array, das 784 (28 * 28 Pixel) Elemente enthält. Jedes Pixel (Element) im umgeformten Bild wird als Eingangsmerkmal für ein MLP-Modell betrachtet. Die ursprünglichen Pixelwerte reichen von 0 bis 255. Ich habe sie also durch 255 geteilt, um sie in den Bereich von 0,0 bis 1,0 zu bringen.

Ich habe jedes Label in einen One-Hot-Vektor umgewandelt.Wenn das Label beispielsweise 4 ist, lautet der entsprechende One-Hot-Vektor [0,0,0,0,1,0,0,0,0,0]

SHAPES AFTER

Training images: (784, 60000)
Training labels: (10, 60000)
Test images: (784, 10000)
Test Labels: (10, 10000)

ONE HOT VEKTOR
RESHAPE
NORMALIZE

SOFTMAX

Ich habe Softmax in der Ausgabeschicht verwendet. Die Ausgabe eines Neurons deuten Sie als die Wahrscheinlichkeit, dass die Eingabe x der entsprechenden Klasse angehört.

Ich habe wegen Softmax eine leicht modifizierte Version des Cross-Entropy Verlustes verwendet, um die Training- und Testverlust zu berechnen (k ist die Anzahl der Klassen) :

$$L = \sum_{i=1}^{k} (y_i \log (\hat{y}_i)$$

EXPERIMENTE

NETWORK 1

- Layer Configurations : [784,128,10]
- Activations : [relu, softmax]
- Epoche : 400
- Learning Rate : 0.2

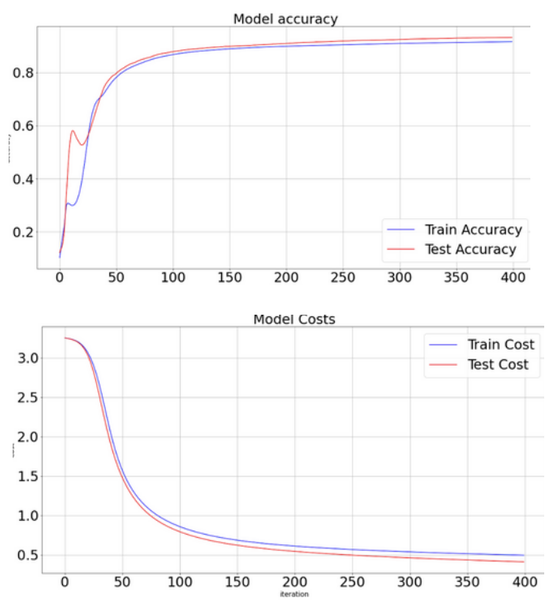
NETWORK 2

- Layer Configurations : [784,512,128,10]
- Activations : [relu, relu, softmax]
- Epoche : 400
- Learning Rate : 0.2

NETWORK 3

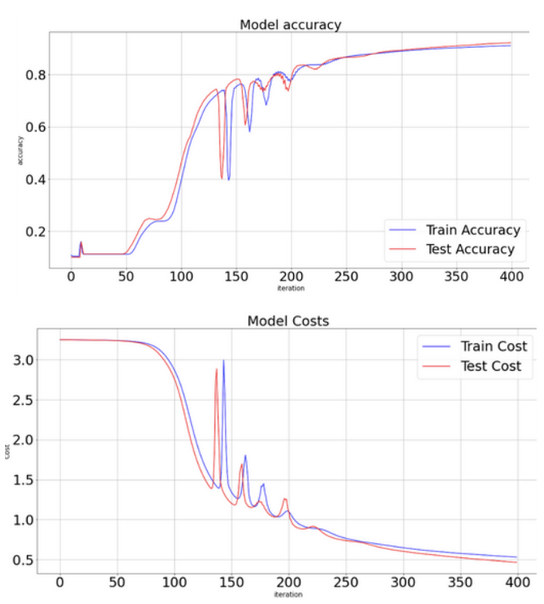
- Layer Configurations : [784,128,10]
- Activations : [sigmoid, softmax]
- Epoche : 400
- Learning Rate : 0.2

NETWORK 1



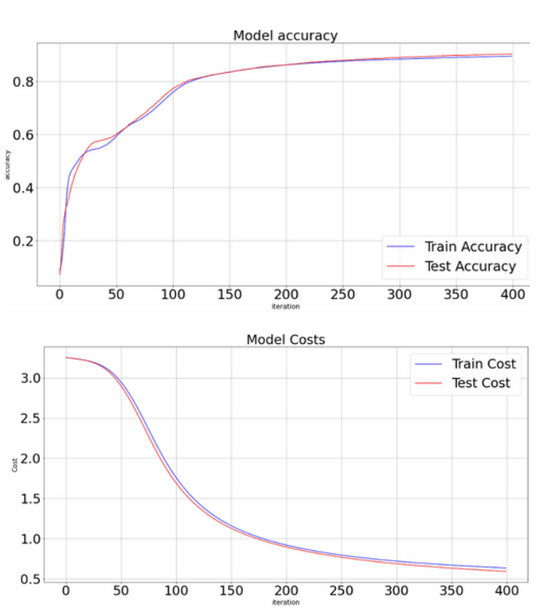
ERFOLGSRATE 91.9%

NETWORK 2



ERFOLGSRATE 91%

NETWORK 3

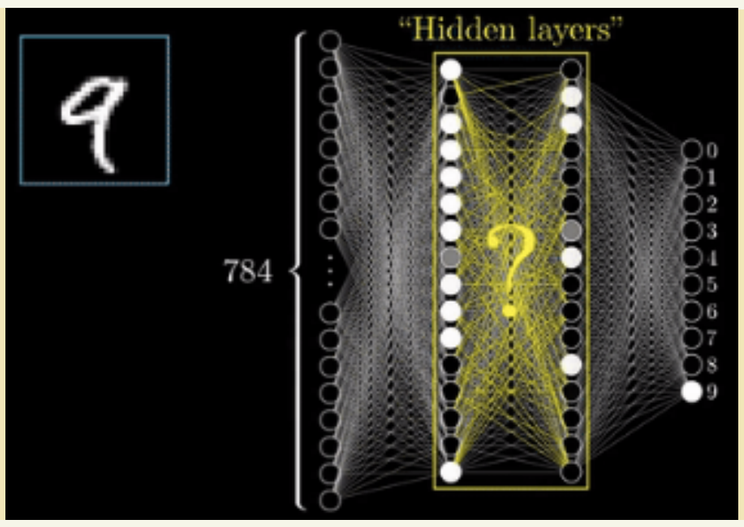


ERFOLGSRATE 90%

OVERFIT

Eine Überanpassung des Modells ist erkennbar, wenn die Genauigkeit der Trainingsdaten höher ist als die Genauigkeit der Validierungsdaten.Keines meiner Modelle zeigte eine Überanpassung.

Wenn sich eines meiner Modelle zu stark anpasst, würde ich den Dropout-Algorithmus verwenden. Dieser Algorithmus setzt einen Teil der Eingaben (in diesem Fall die Pixelwerte) auf 0.



FAZIT

Nach 3 verschiedenen Experimenten wurde festgestellt, dass die Sigmoidfunktion langsamer ist als die Relu-Funktion.

Mit diesen 3 Modellen wurden fast die gleichen Erfolgsquoten erzielt.In der Konfiguration des erfolgreichsten Modells wurden die Relu-Funktion und eine versteckte Schicht mit 128 Neuronen verwendet.

Als Ergebnis konnten handgeschriebene Ziffern erfolgreich mit MLP klassifiziert werden.