

# Abschlusspräsentation Computational Intelligence Ziffernerkennung mittels neuronalem Perzeptron-Netz

Hannes Dröse

## **Projektziel**

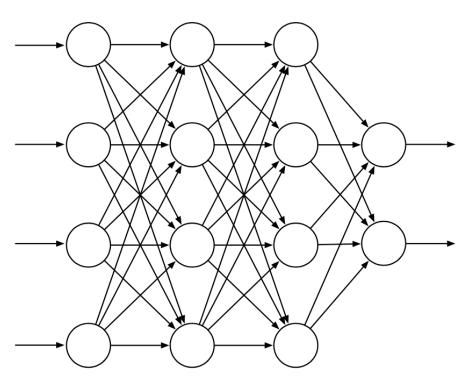




- Erkennung der Ziffern 0-9 in 7x5 Pixelraster
- Pixelwerte: 0 leer, 1 gefüllt
- kleine interaktive Anwendung mit Eingabe und Ausgabe

## Lösungsmethode





Grafik aus Sutton und Barto 2018

- neuronales Netz mit Schichten aus Perzeptronen
- vorwärtsgerichtet
- Training mittels Beispieldaten & Gradientenabstiegsverfahren

## **Berechnungen im Netz**



Informatik

Ausgabe einer Schicht:

$$\overrightarrow{y} = \phi(\overrightarrow{b} + W \cdot \overrightarrow{x})$$

Korrektur der Gewichte:

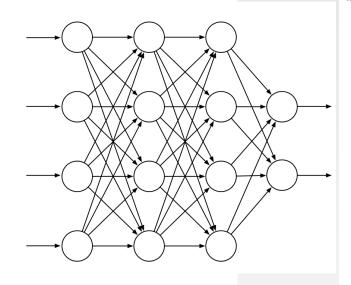
$$W_{neu} = W_{alt} - \lambda \cdot \Delta_W$$

$$\overrightarrow{b}_{neu} = \overrightarrow{b}_{alt} - \lambda \cdot \Delta_{\overrightarrow{b}}$$



$$\Delta_{\overrightarrow{b}} = \frac{\delta E}{\delta \overrightarrow{b}}$$

$$\Delta_W = \frac{\delta E}{\delta W}$$



W..Gewichtsmatrix

 $\overrightarrow{b}$ ..Biaswerte

 $\phi$ ..Aktivierungsfunktion

λ..Lernrate

E..Fehlerfunktion

#### Parameter des Netzes



- Topologie des Netzes
  - Eingabe-Schicht: 7x5 Pixel = 35 Neuronen
  - Ausgabe-Schicht: 10 Ziffern = 10 Neuronen
  - verdeckte Schichten:
    - eine Schicht reicht i.d.R. aus
    - Neuronenzahl wird experimentell ermittelt
- Aktivierungsfunktion: logistische Funktion
- Fehlerfunktion: Mean Squared Error

 $\phi(x) = \frac{1}{1 - 1 - 2^{-x}}$ 

$$E(\overrightarrow{y}, \hat{y}) = \frac{1}{n} \cdot \sum_{i} (y_i - \hat{y}_i)^2$$

Optimierung: Batch-Learning

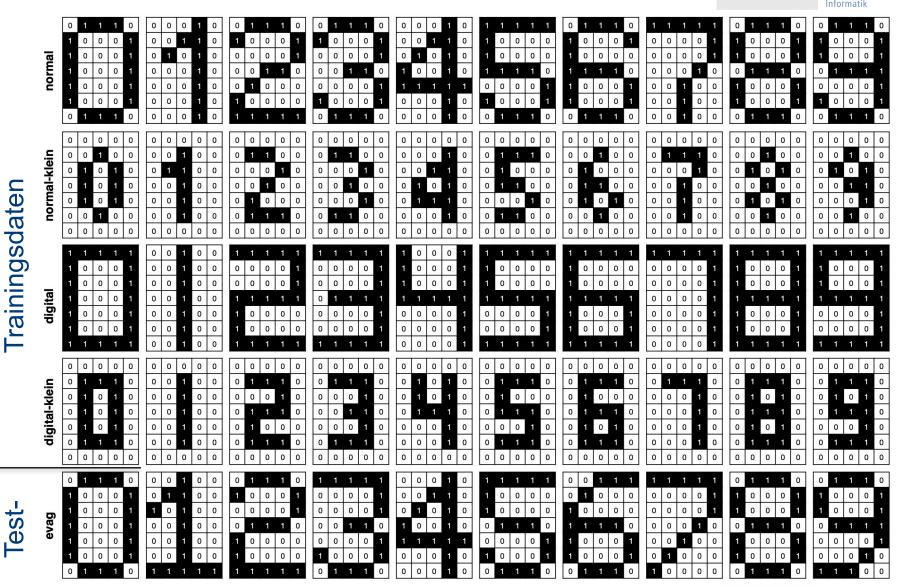
## **Umsetzung**



- Programmiersprache: Python
- Net-Module:
  - Bibliothek zum Arbeiten mit neuronalen Netzen
  - Initialisierung, Berechnung von Ergebnissen und Fehlerwerten, Training usw.
- Digits-Module:
  - lädt die Beispieldaten (CSV-Dateien in einem Ordner)
- Hauptprogramm:
  - Jupyter Notebook
  - bestimmt das optimale Netz
  - trainiert das Netz
  - generiert Grafiken und Daten zur Auswertung
- Demo-App:
  - Projektziel
  - interaktive Anwendung mit Eingabe-Raster und Ausgabe

## Beispieldaten





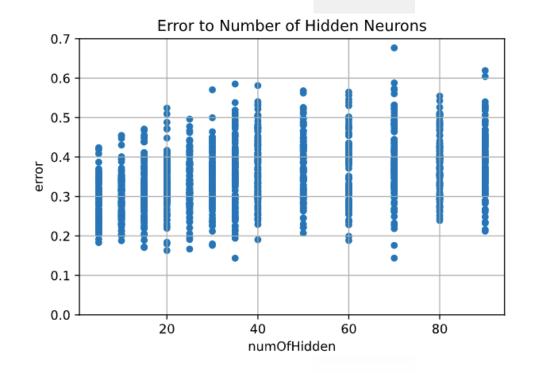
## Simulation: Ermittlung der Neuronenzahl

FACHHOCHSCHULE
 ERFURT UNIVERSITY
 OF APPLIED SCIENCES
 Angewandte

Angewandte Informatik

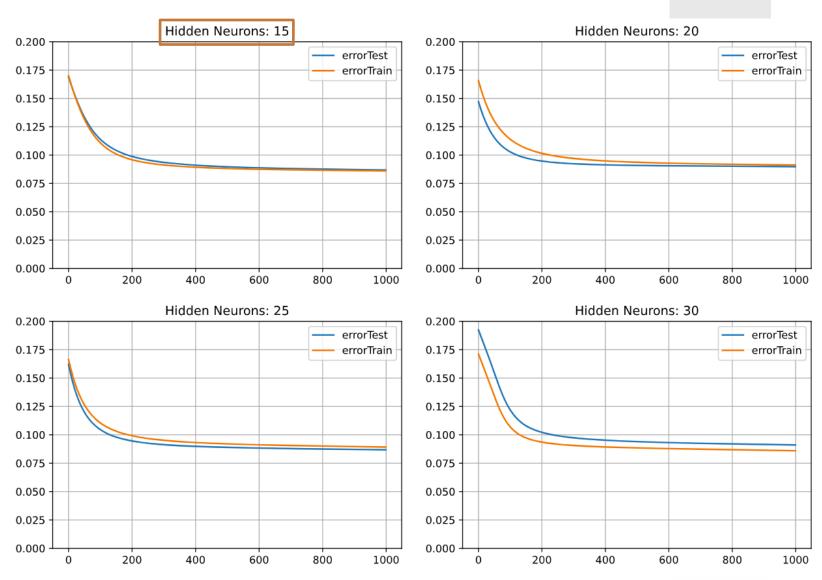
- verwendete Neuronenzahlen:5, 10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80, 90
- zu jeder Zahl wurden 100 Netze initialisiert
- anschließend wird für 1.000 Epochen trainiert

Neuronenzahl	Fehlerwert des besten Netzes
35	0,1435
70	0,1437
20	0,1629
25	0,1667
15	0,1707
30	0,1760
5	0,1830
10	0,1878
60	0,1885
40	0,1904
50	0,2071
90	0,2121
80	0,2391



#### Simulation: Ermittlung der Neuronenzahl





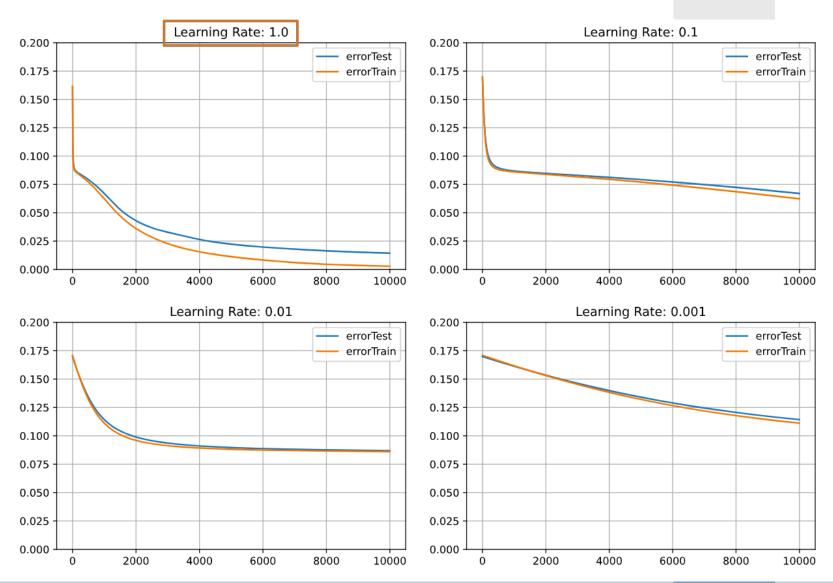
# Simulation: Ermittlung der Lernrate FI

FACHHOCHSCHULE
ERFURT UNIVERSITY
OF APPLIED SCIENCES
Angewandte
Informatik

- Vergleich von 4 verschiedenen Lernraten:
  1,0 0,1 0,01 0,001
- Training des Netzes über 10.000 Epochen

## Simulation: Ermittlung der Lernrate FIIE

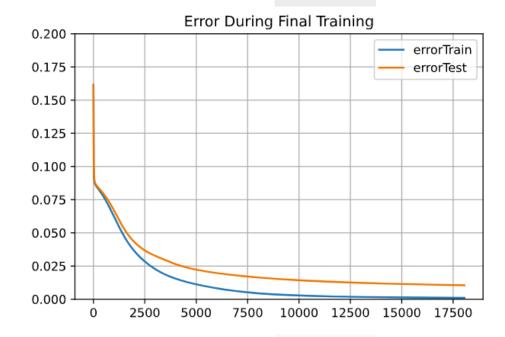




## **Simulation: Finales Training**

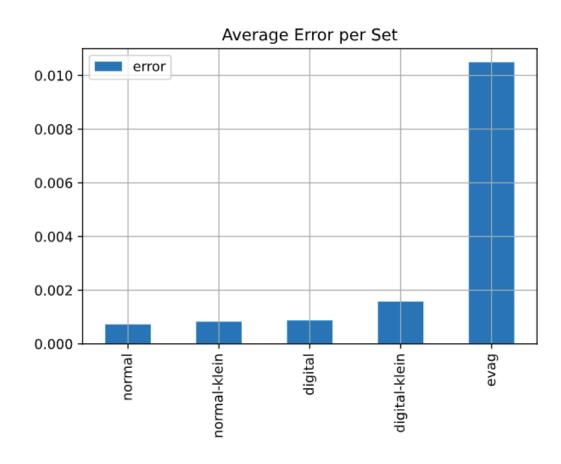


- abschließendes Training bis Fehler unter 0,001 liegt
- insgesamt: 18.052 Epochen



# **Ergebnisse**

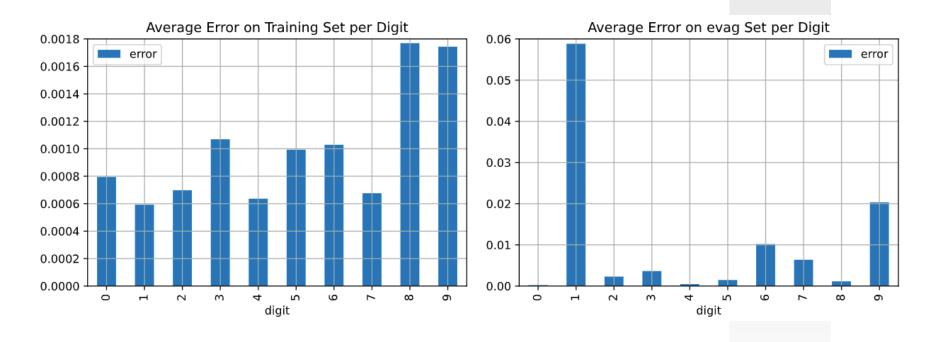




## **Ergebnisse**



Angewandte Informatik





## **Demonstration**



#### Vielen Dank für die Aufmerksamkeit

#### Quellen



- Lämmel, Uwe, und Jürgen Cleve. 2020. Künstliche Intelligenz: Wissensverarbeitung – Neuronale Netze. Carl Hanser Verlag GmbH & Co. KG.
- Papp, S., W. Weidinger, M. Meir-Huber, B. Ortner, G. Langs, und R. Wazir. 2019. Handbuch Data Science: Mit Datenanalyse und Machine Learning Wert aus Daten generieren. Carl Hanser Verlag GmbH & Co. KG.
- Patterson, Josh, und Adam Gibson. 2017. Deep Learning: A Practitioner's Approach. O'Reilly Media. https://books.google.de/books?id=qrcuDwAAQBAJ.
- Sutton, Richard S., und Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning series. MIT Press.