TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Technische Informatik, *Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur*

Künstliche Neuronale Netze

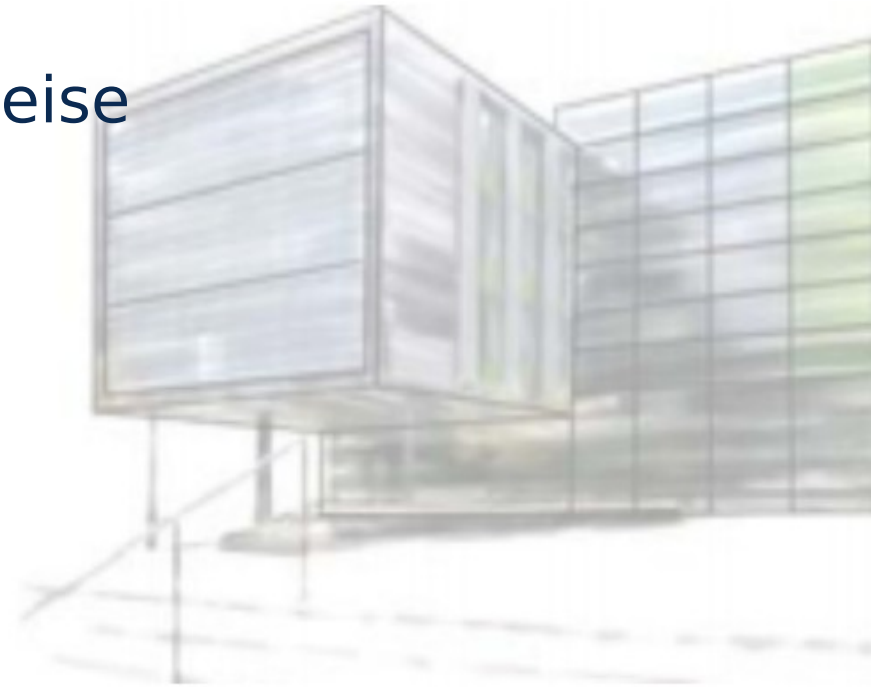
Hauptseminar

Martin Knöfel

Dresden, 16.11.2017

Gliederung

1. Motivation
2. Aufbau und Funktionsweise
3. Hardwarebedarf
4. Fazit



Künstliche Neuronale Netze

- Gehirnmodell
  (Simulation von Neuronalen Netzen)
- Machine Learning
- Künstliche Intelligenz
- Strukturierter Aufbau

Simulation des Gehirnmodells

- Aufbau: Neuronen Netz
- Funktion: Simulation jedes Neurons
- Aufgabe: Lösen von Problemen
  (Kognition – umgestalten von Informationen)
- Problem: Lernverfahren

Machine Learning

- System sammelt Wissen aus Daten
- „Big Data“
- Früher: Formalisierungen und Algorithmen
  (kompliziert für kognitive Aufgaben)
- Heute: Künstliche Neuronale Netze [GBC16]

Anwendungen

- Modellbildung (Prognose, Kennlinien)
- Bildvearbeitung (allg. Mustererkennung)
- Automatisierung (Qualitätskontrolle)
- Medizin (Diagnostik)

Anwendungen

Georg-Schumann-Bau, TU Dresden [Hb3]

Anwendungen

Georg-Schumann-Bau umgewandelt mit dem AI Painter [AIP]

Neuron

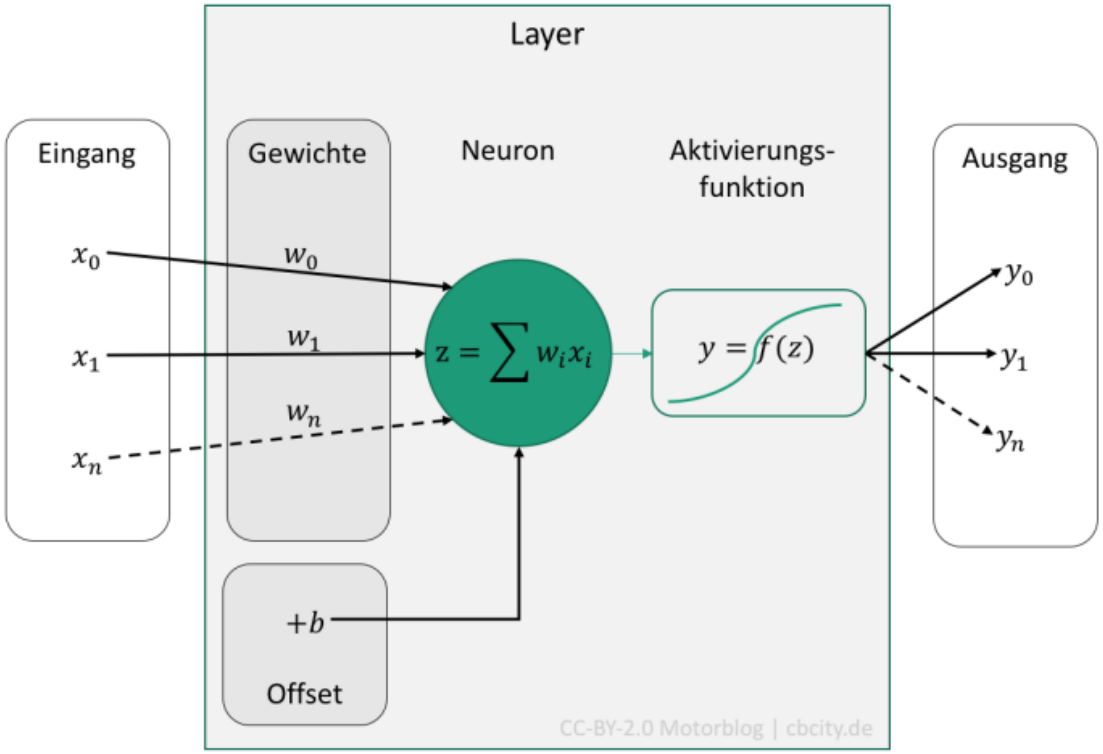


Abbildung: Aufbau eines Neurons [BAL16]

Arten von Neuron

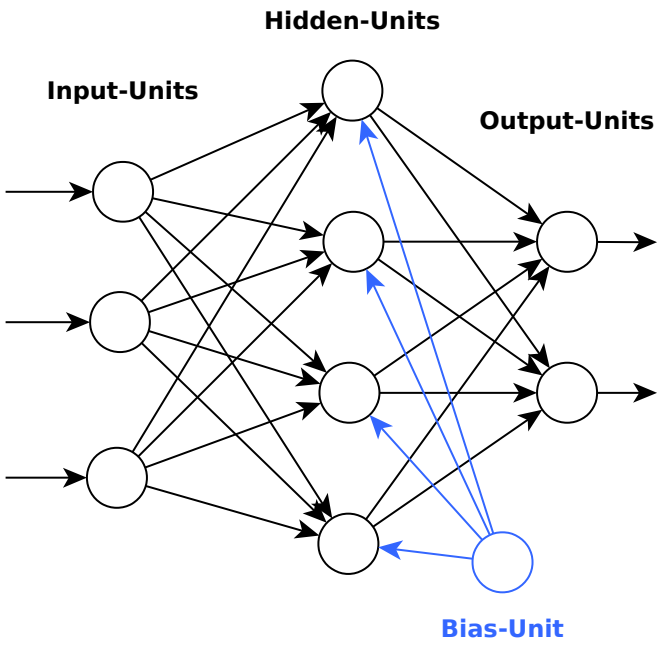


Abbildung: Netzbeispiel

Input-Units:
- nur ein Eingang
- Propagierungsfunktion entfällt

Hidden-Units:
- Ein- und Ausgabe unbekannt (deshalb „versteckte“ Knoten)

Output-Units:
- Ausgabe des Systems

Bias-Units:
- Schwellwert (Verschiebung der Aktivierungsfunktion)

Aktivierungsfunktionen

- Linear (mit und ohne Schwellwert)
- Binäre Schwellenwertfunktion (eindeutige Aktivität)
- Sigmoid Funktion
- Tangens Hyperbolicus (Tanh)
- Rectifier Linear Unit (ReLu)

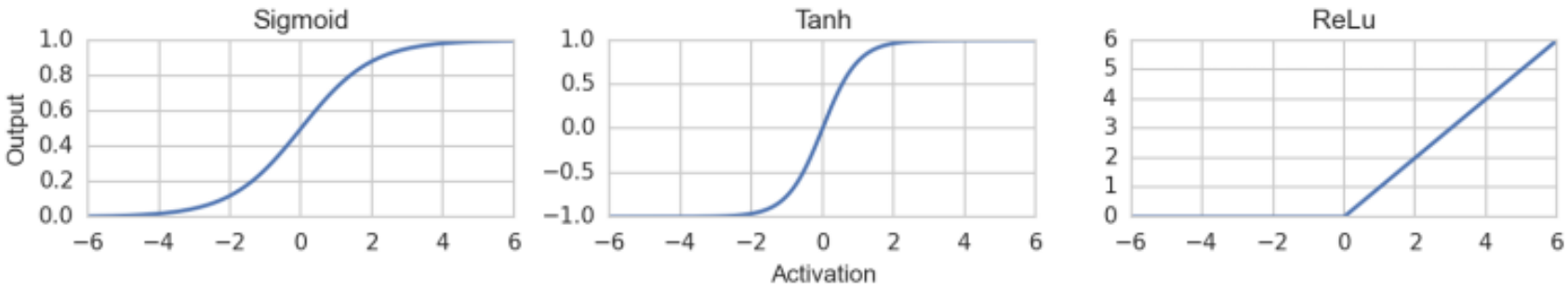


Abbildung: Aktivierungsfunktionen [BAL16]

Aktivierungsfunktionen

- Monoton wachsend
- Stetigkeit (Differenzierbarkeit → Lernverfahren)

- Obere Grenze
- Untere Grenze

} vermeidet Dominoeffekt bei zu starken Aktivitätslevel

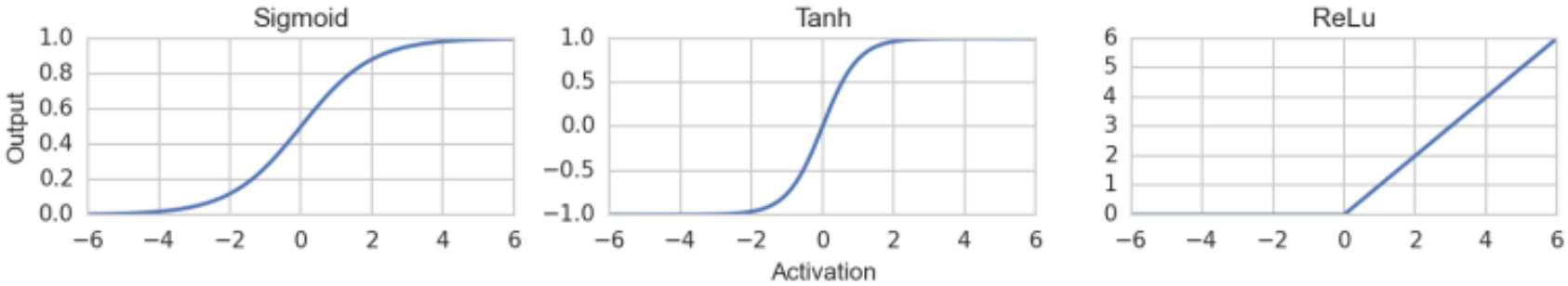


Abbildung: Aktivierungsfunktionen [BAL16]

Netztopologie

- Gewichte enthalten das Wissen
- Anordnung in Schichten (Parallelität)
- Rückkopplungen:
  - ohne („feedforward“ Netze)
  - mit („rekurrente“ Netze)

Lernverfahren

- Gewichte enthalten das Wissen
- Das Lernverfahren ist abhängig vom Aufbau des Netzes

3 Arten von Lernverfahren:

- Überwachtes Lernen („supervised Learning“)
- Unüberwachtes Lernen („unsupervised Learning“)
- Bestärkendes Lernen („reinforcement Learning“)

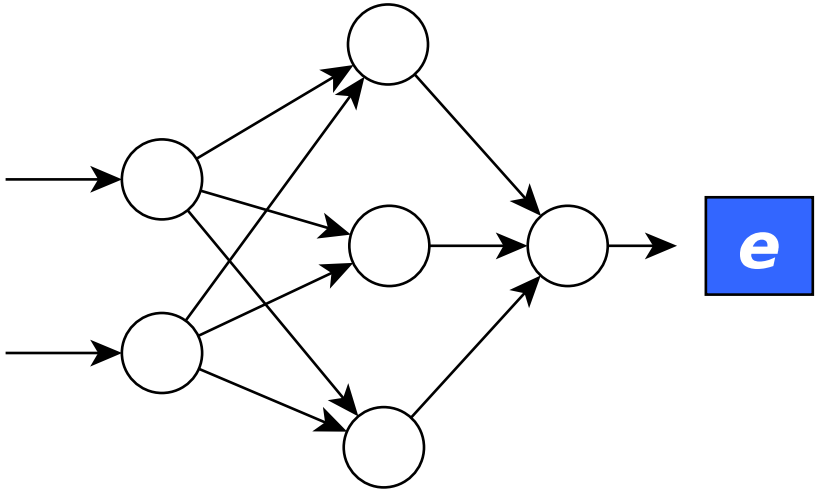
Backpropagation

- Überwachtes Lernen
- „feedforward“ Netze
- Gradientenabstiegsverfahren

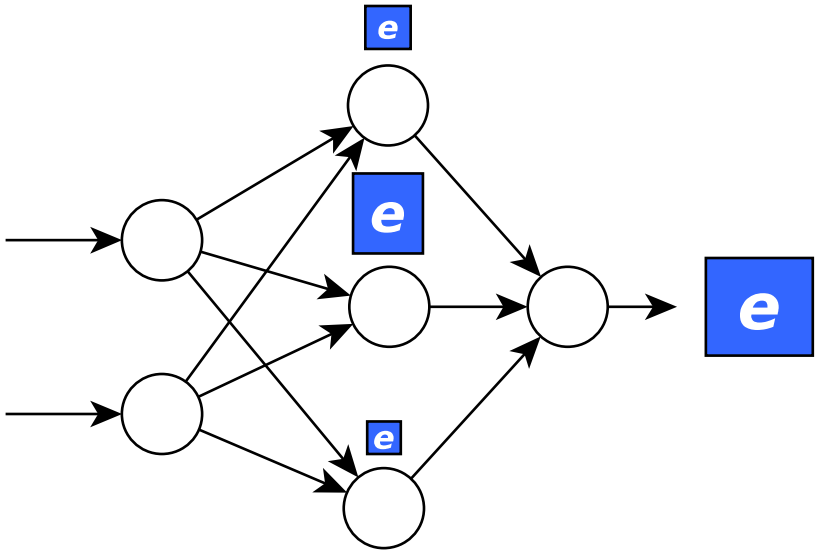
Algorithmus:

1) Berechnung der Netzausgabe
2) Ermitteln des Fehlerwertes (Fehlerfunktion)
3) Aufteilung des Fehlerterms
   → negativer Gradient
4) Modifizieren der Gewichte

Backpropagation



Backpropagation



Bedarf an Beschleunigern

- Parallelität (Schichten)
- Hoher Rechenaufwand:
  - Propagationsfunktion
  - Aktivierungsfunktion
  - Lernverfahren (besonders aufwändig)

FPGA Beispiel

- AlexNet: Convolutional Neural Network
- Programmiert mit Cuda
- Klassifizierung von Bildern

System	Throughput	Power	Throughput/Watt
Arria 10-115 (FP32, @275Mhz)	600 img/s	~31W	19,4 img/s/W
Arria 10-115 (FP32, @350Mhz)	800 img/s	~37W	21,6 img/s/W
Arria 10-115 (FP16, @350Mhz)	1200 img/s	~36W	33,3 img/s/W
2x Arria 10-115	2400 img/s	~75W	32 img/s/W
Caffe on NVIDIA Titan X	1000 img/s	~250W	4 img/s/W
cuDNN4 on NVIDIA Titan X	3216 img/s	~227W	14,2 img/s/W

Tabelle: AlexNet Vergleichsanalyse [BAL16]

FPGA Vorteile

- Niedriger Stromverbrauch
- Hohes Potential für die Beschleunigung von Lernverfahren
- Platzsparender (Industrie)

System	Throughput	Power	Throughput/Watt
Arria 10-115 (FP32, @275Mhz)	600 img/s	~31W	19,4 img/s/W
Arria 10-115 (FP32, @350Mhz)	800 img/s	~37W	21,6 img/s/W
Arria 10-115 (FP16, @350Mhz)	1200 img/s	~36W	33,3 img/s/W
2x Arria 10-115	2400 img/s	~75W	32 img/s/W
Caffe on NVIDIA Titan X	1000 img/s	~250W	4 img/s/W
cuDNN4 on NVIDIA Titan X	3216 img/s	~227W	14,2 img/s/W

Tabelle: AlexNet Vergleichsanalyse [BAL16]

KNN Vorteile

- Kein Modell (Formalisierung) notwendig
- Großer Anwendungsbereich
- Hohe Fehlertoleranz
- Widerstandsfähigkeit (Rauschen, fehlender Input)
- Generalisierungsfähigkeit

KNN Nachteile

- „Black Box“ - empirische Forschung
  - Prototyping
- Hoher Rechenaufwand
- Hoher Datenbedarf (Lernen)
- Lange Trainingszeit

Vielen Dank für Ihre Aufmerksamkeit!

Quellen

[Bal16] Paul Balzer. Neuronale Netze einfach erklärt. 2016.
url: www.cbcity.de/tutorial-neuronale-netze-einfach-erklaert.

[Jen16] Intel FPGA: Bill Jenkins. Machine Learning on FPGAs: Neural Networks. 2016.
url: https://de.slideshare.net/embeddedvision/accelerating-deep-learning-using-altera-fpgas-a-presentation-from-intel

[GBC16] Ian Goodfellow, Yoshua Bengio und Aaron Courville. Deep Learning. MIT Press, 2016.
url: www.deeplearningbook.org.

[Hb3] Photograph by: Hullbr3ach. Georg-Schumann-Bau. 2006.
url: https://commons.wikimedia.org/wiki/File:TU-Dresden-Georg-Schumann-Bau.jpg

[AIP] AI Painter
url: www.instapainting.com/ai-painter

Frameworks

- Theano
- MatConvnet
- Cuda-convnet
- TensorFlow
- Intel DAAL
- Torch
- Caffe

KNN Typen (Beispiele)

Kohonennetze
- Funktionen approximieren
- Inverse 2D-Kinematik (auch Wegfindung)
- Mustererkennung

Convolutional Neural Network
- Bilderkennung, Spracherkennung

Perceptron (einschichtig, mehrschichtig)

Zahlenbeispiel

- Handschrifterkennung 28x28 Pixel
- 784 Input-Units
- 2x 16 Hidden-Units

1. Schicht:
→ 16 x 784 = 11968 Multiplikationen und Additionen