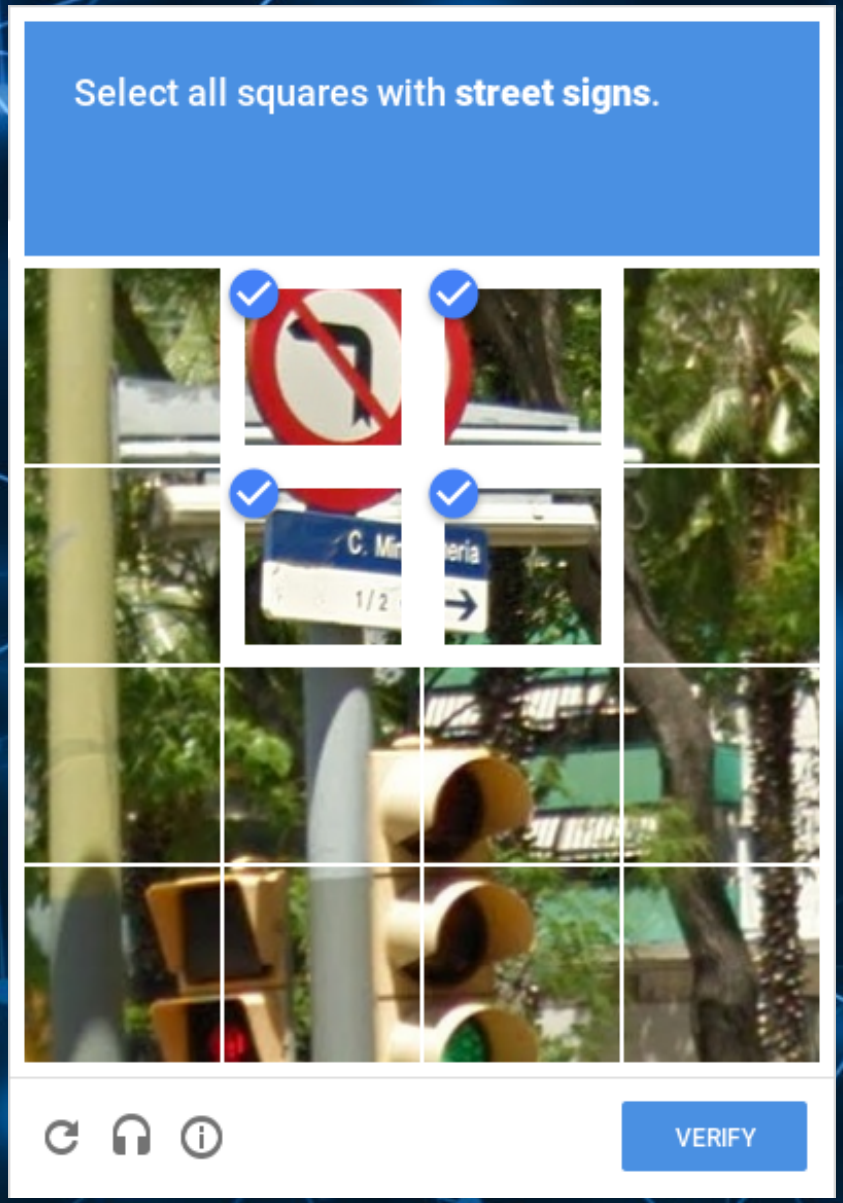
Neural Networks

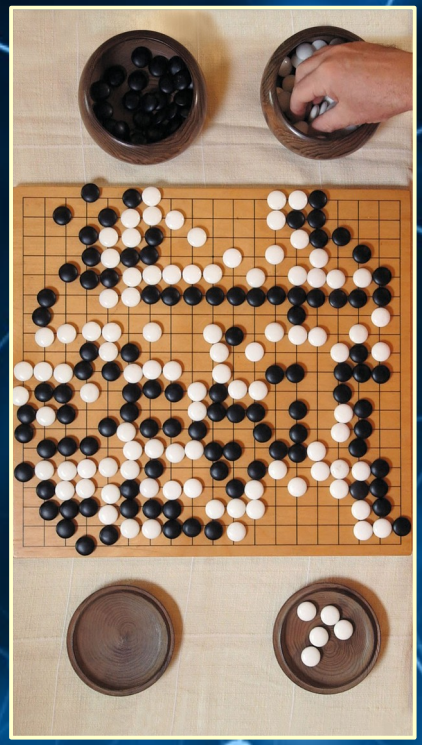
mit 

Anwendungsbeispiele

Anwendungsbeispiele



Anwendungsbeispiele



Anwendungsbeispiele

WaveNet

ist ein Voice Synthesizer und ein Projekt von Google DeepMind.

WaveNet basiert auf generativen Text-To-Speech Modellen.

https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Anwendungsbeispiele

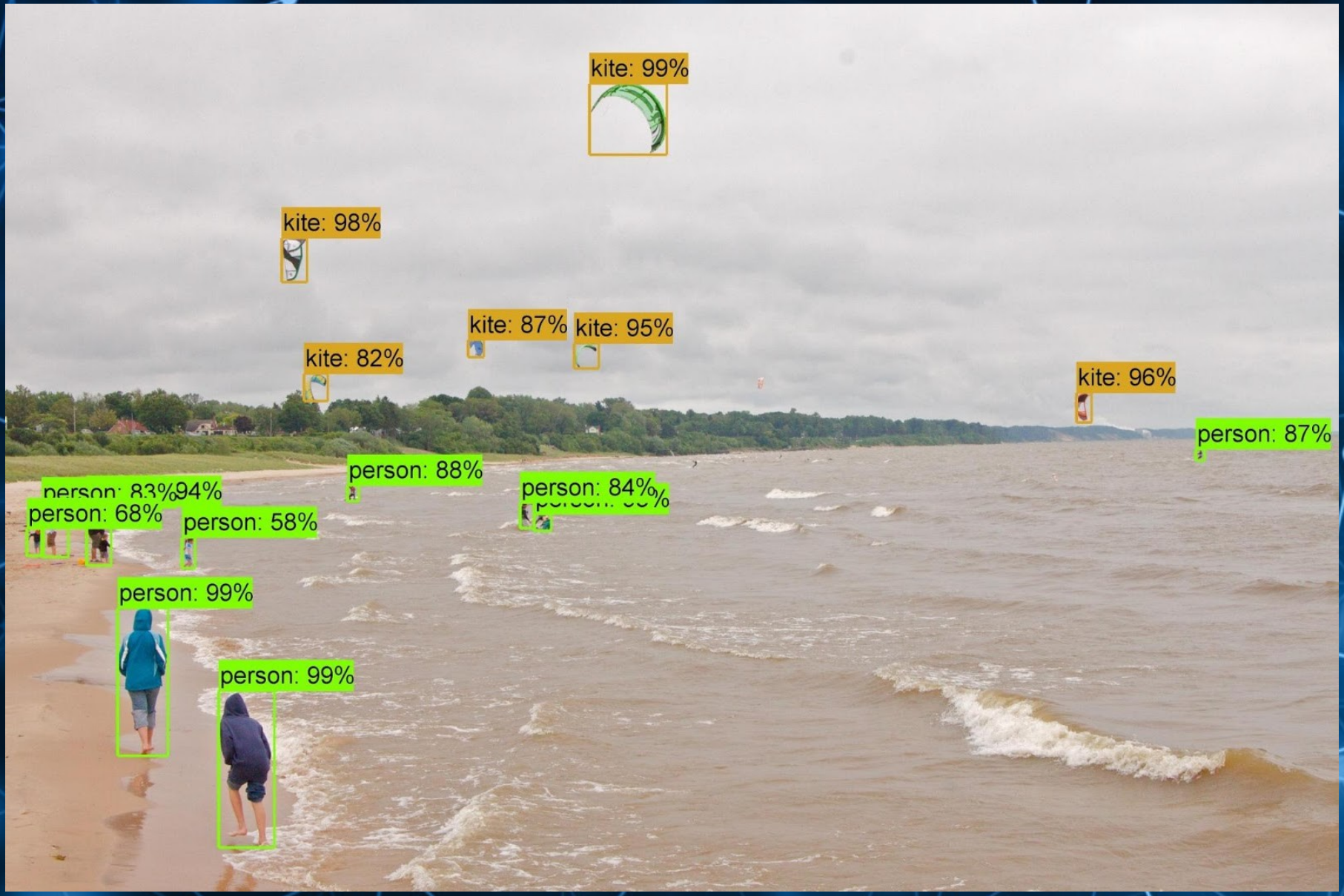
Syntaxnet

- TensorFlow Toolkit für Natural Language Understanding (NLU)
- 99% POS und 96% der Syntax Sunction Assignment
- F-Score = 0.94

Word2vec (Modellgruppe)

- für Word Embeddings
- mit nur 2 Layern sehr flach
- Erzeugt einen Vektorraum basierend auf gegebenen Korpora

Anwendungsbeispiele



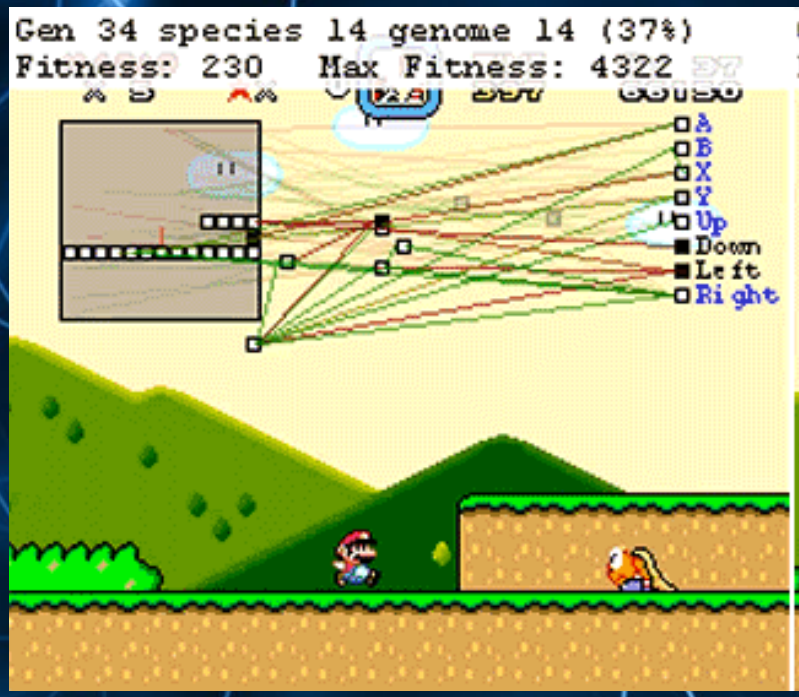
Ebenfalls von Google: **AutoML & NASNet**
http://automl.info/
https://github.com/tensorflow/models/tree/master/research/slim/nets/nasnet

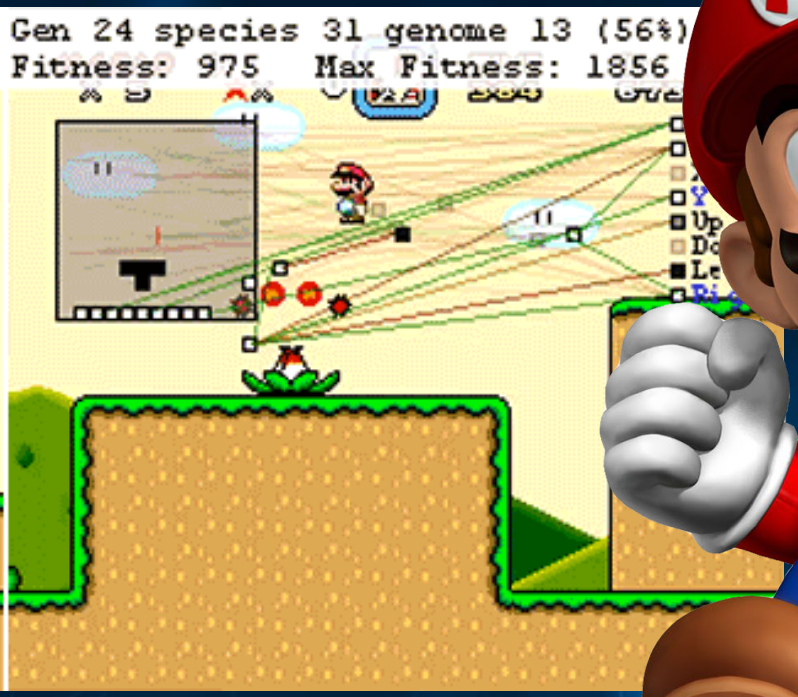
Anwendungsbeispiele

NVIDIA

http://research.nvidia.com/publication/2017-12_Unsupervised-Image-to-Image-Translation

Anwendungsbeispiele





MarI/O

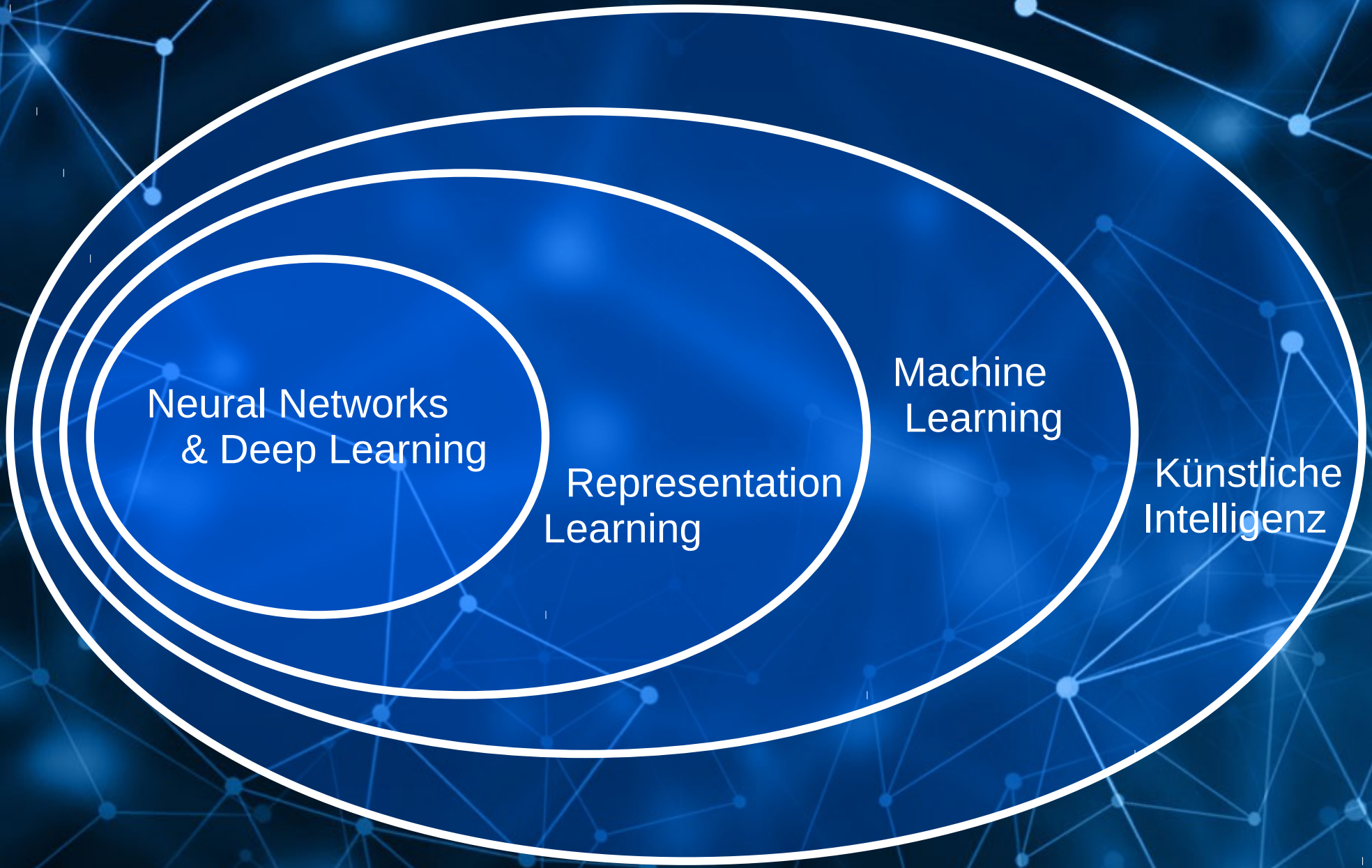
Neural Networks in Kombination mit evolutionären Algorithmen

https://www.youtube.com/watch?v=qv6UVOQ0F44

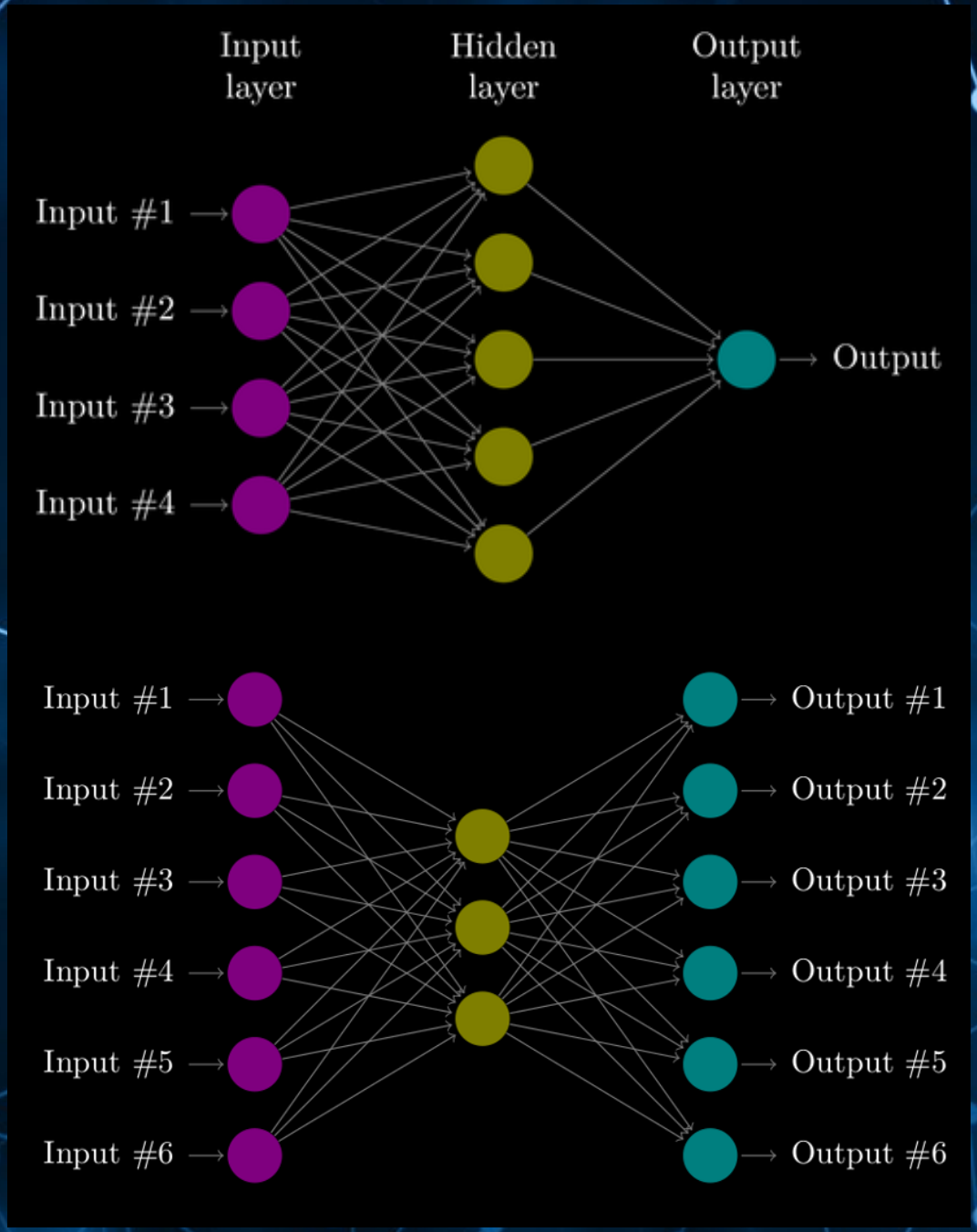
Wie funktioniert das?

Wie funktioniert das?

Was ist ein Neural Network?



Typische Darstellung



Was ist ein Neural Network?

- „Universelle Funktions-Abschätzungs-Maschine“

Was ist ein Neural Network?

- „Universelle Funktions-Abschätzungs-Maschine“
- Programmatische Nachbildung eines organischen Gehirns

Was ist ein Neural Network?

- „Universelle Funktions-Abschätzungs-Maschine“
- Programmatische Nachbildung eines organischen Gehirns
- Lernfähig und veränderbar gemäß der Hebbschen Lernregel „Neurons that fire together wire together“

Was ist ein Neural Network?

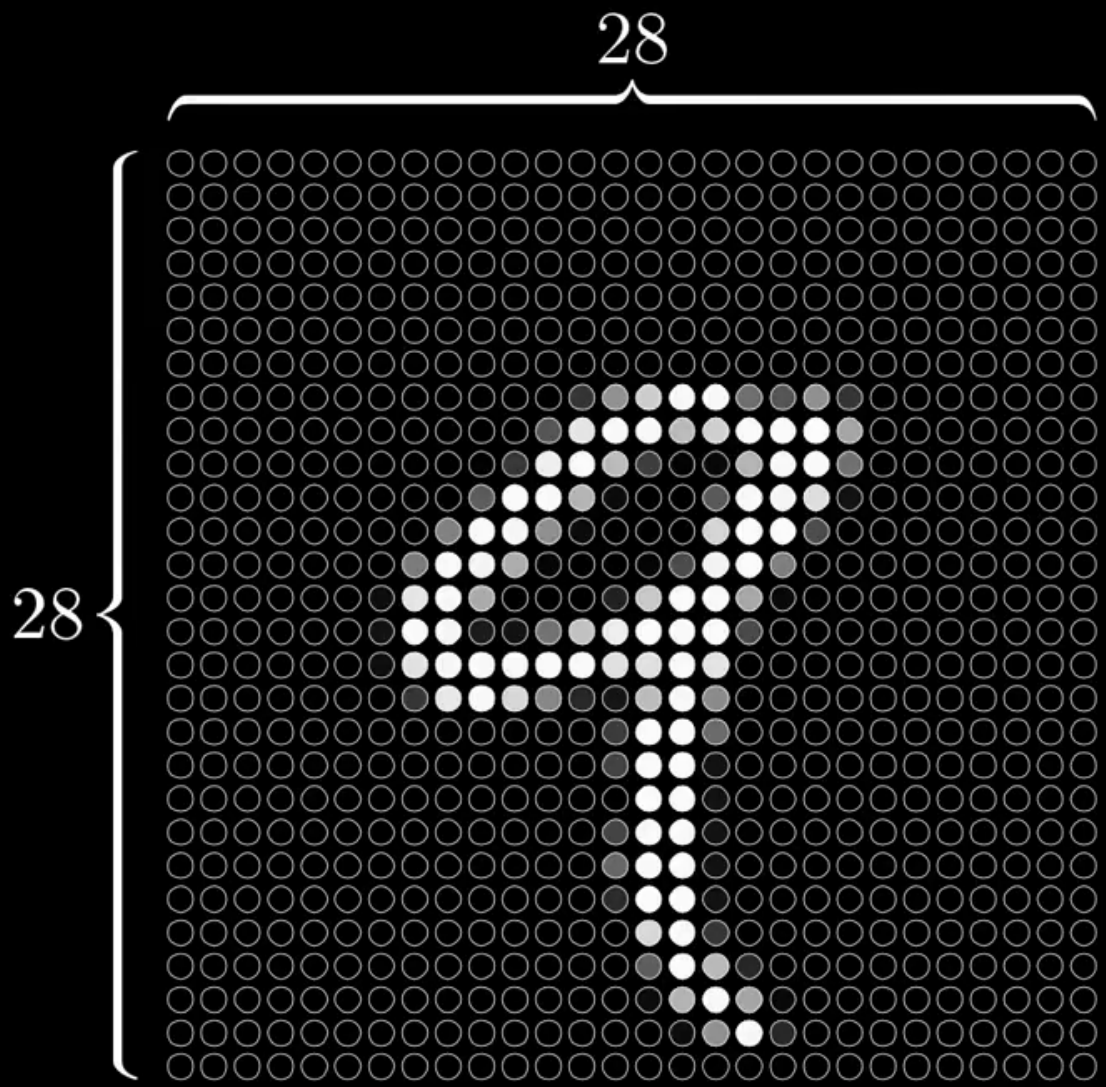
- „Universelle Funktions-Abschätzungs-Maschine“
- Programmatische Nachbildung eines organischen Gehirns
- Lernfähig und veränderbar gemäß der Hebbschen Lernregel „Neurons that fire together wire together“
- Auch Konzepte wie Kurzzeit-/Langzeitgedächtnis sind realisierbar, bekannt als „Long Short-Term Memory“ (LSTM)

Das Neuron

Das Neuron

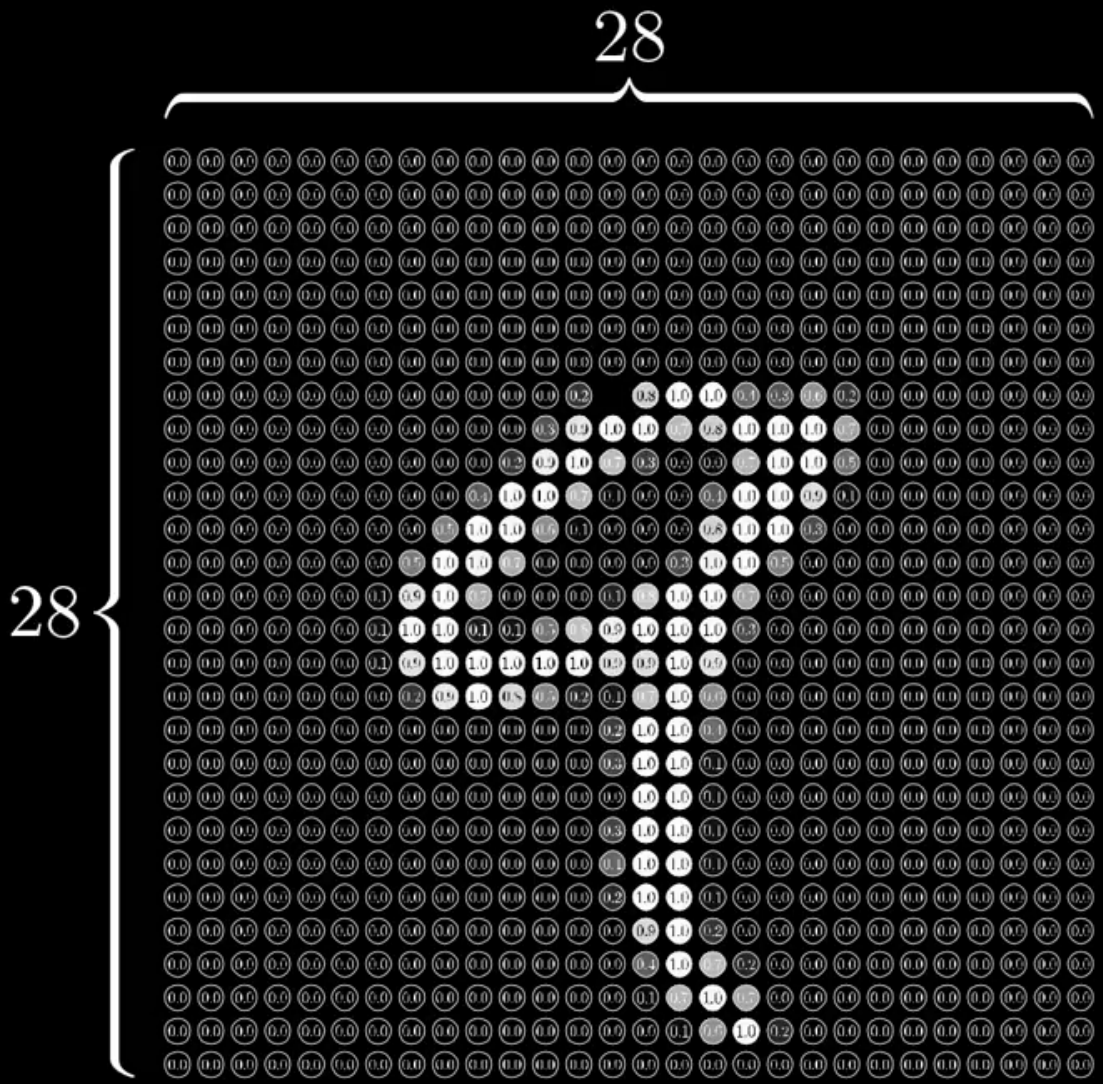
- Einheit, die sich einen Wert merkt
- Dieser Wert („Activation“) ist zwischen 0 und 1
- Der Wert ist das Resultat der Aktivierungsfunktion

Das Neuron



$$28 \times 28 = 784$$

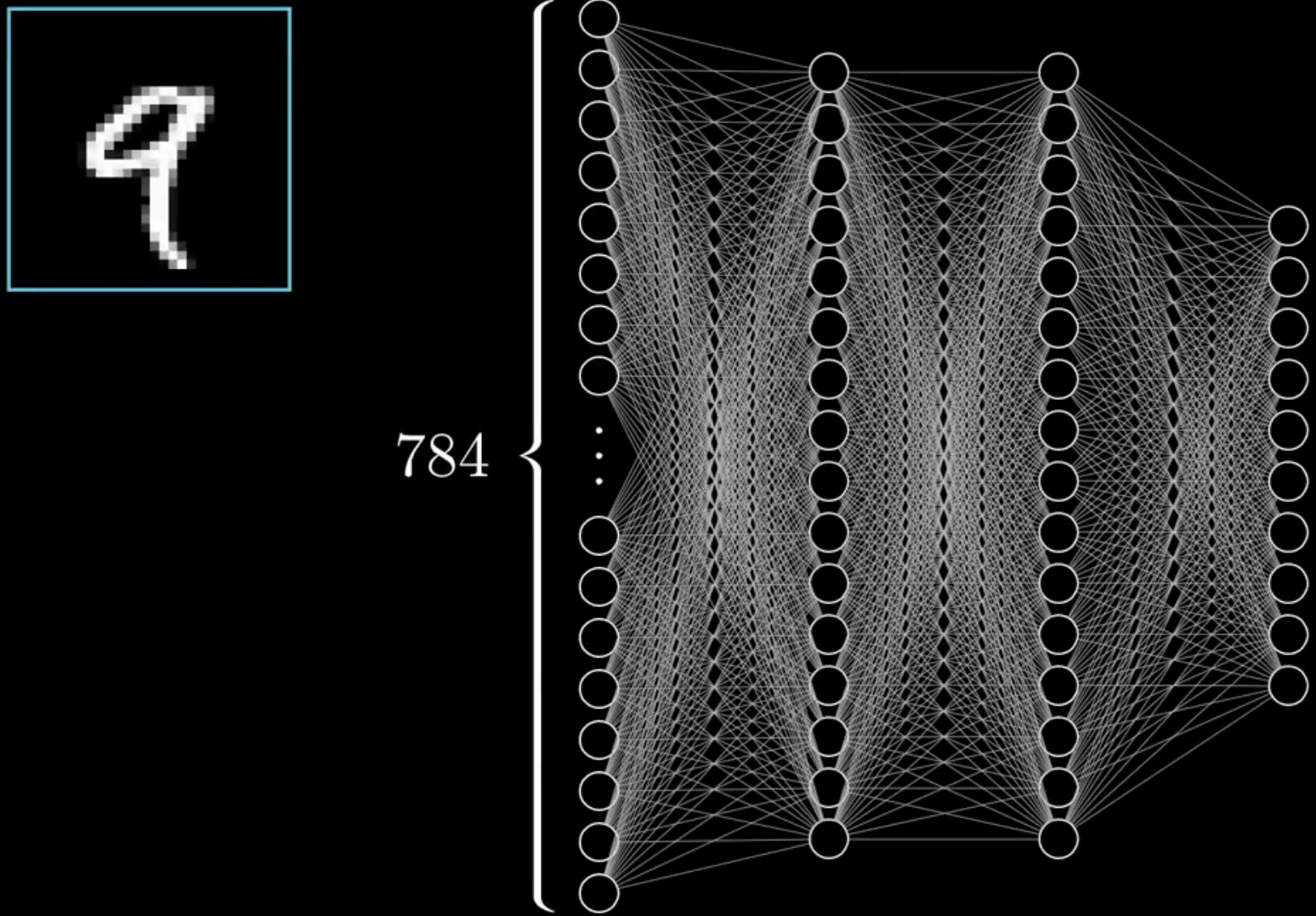
Das Neuron



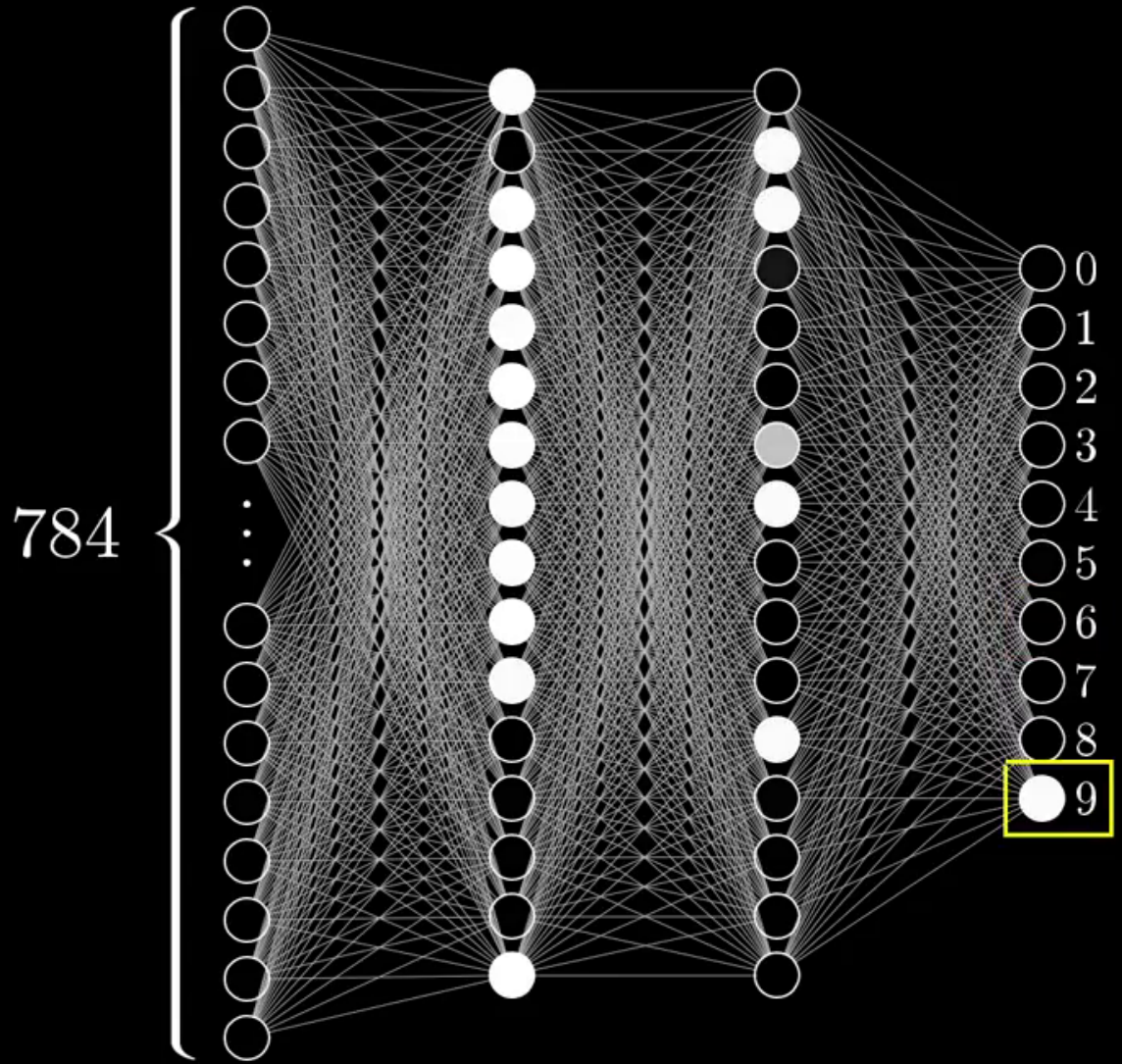
$$28 \times 28 = 784$$

0.58

Das Neuron

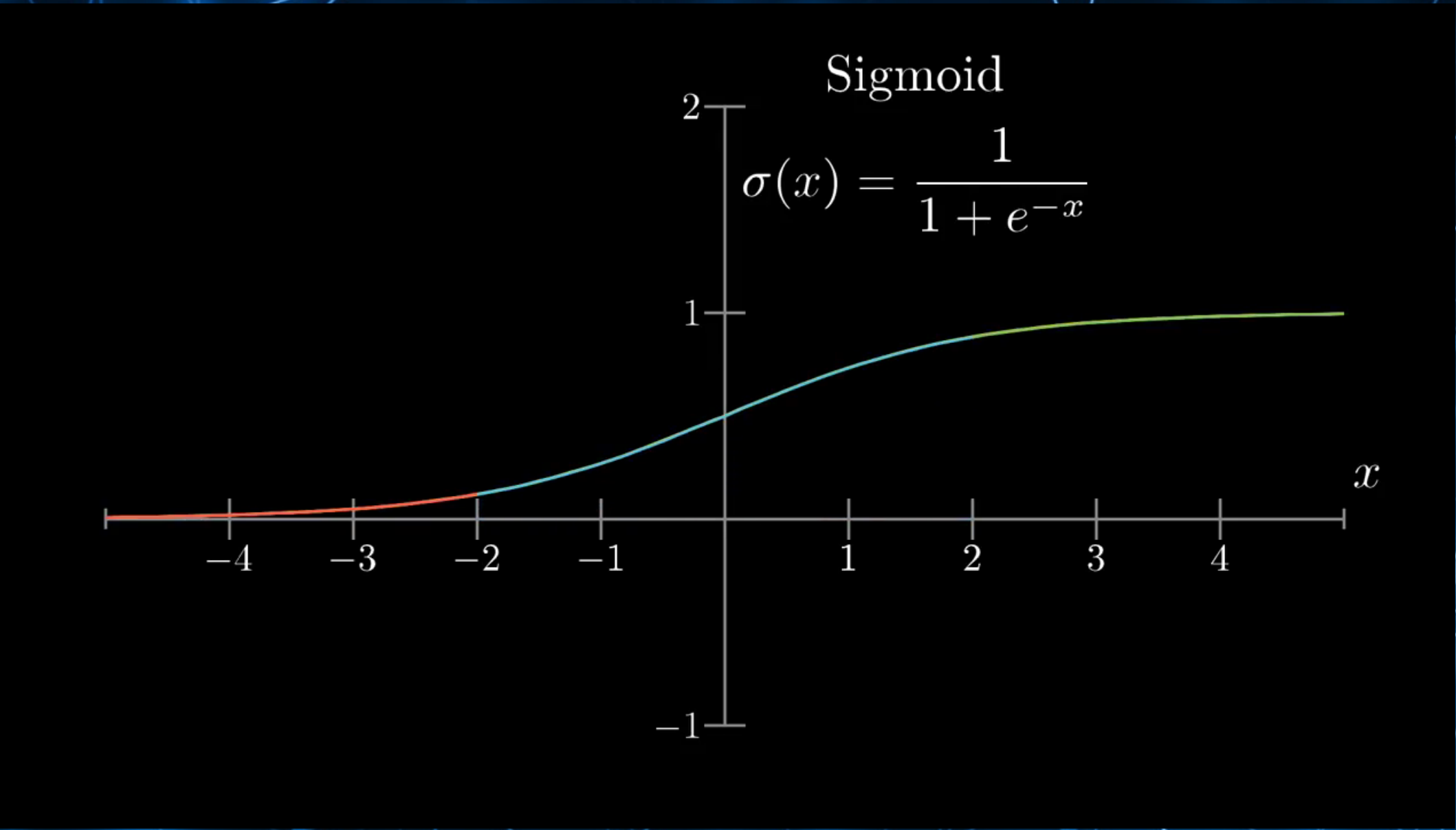


Das Neuron

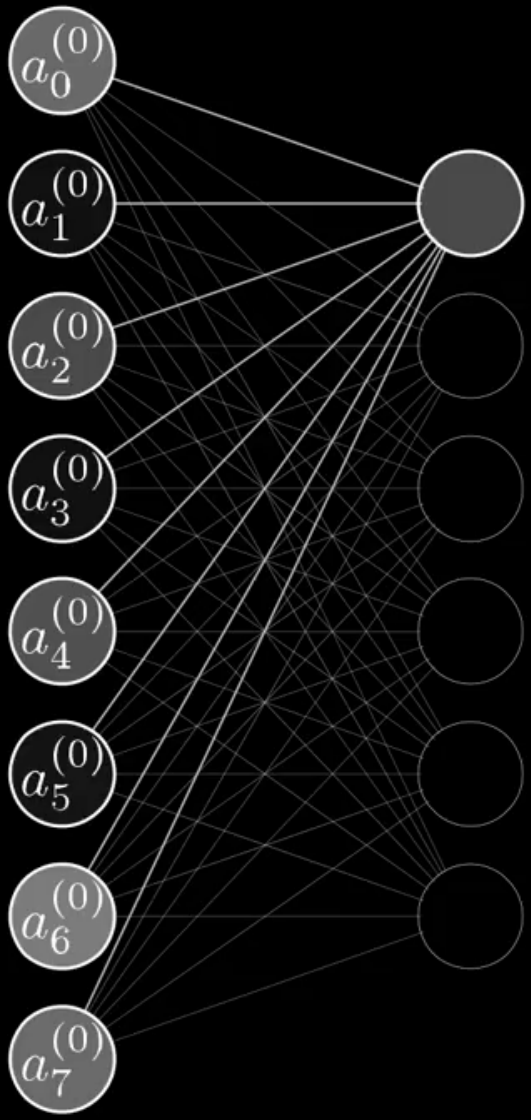


Die Aktivierungsfunktion

Die Aktivierungsfunktion



Die Aktivierungsfunktion



Sigmoid

$$a_0^{(1)} = \sigma\left(w_{0,0}\, a_0^{(0)} + w_{0,1}\, a_1^{(0)} + \cdots + w_{0,n}\, a_n^{(0)} + b_0 \right)$$

Bias

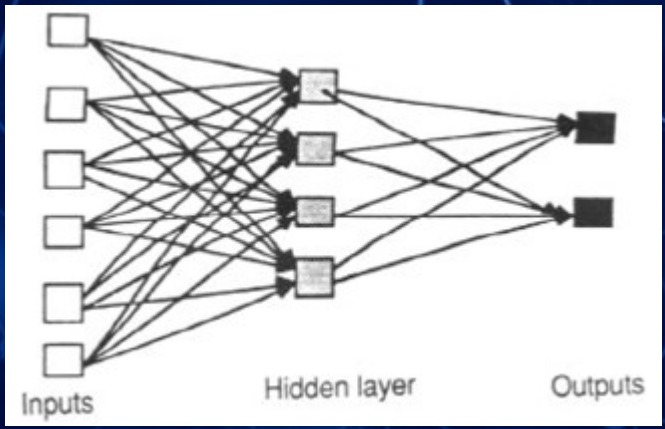
$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

Die Aktivierungsfunktion

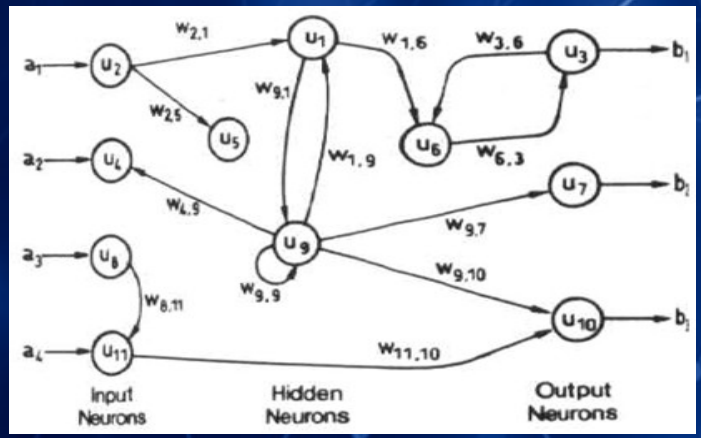
$$\mathbf{a}^{(1)} = \sigma\left(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b}\right)$$

Feedforward ↔ Recurrent

- Fixe Input-Größe
- Keine Feedback-Loops im Network möglich
- **Anwendungsbeispiele:**
  - Pattern Recognition
  - Word Embedding



- Kann Input-Sequenzen unbekannter Länge einlesen (z.B. Sätze)
- Loops im Network möglich
- Wird Schritt für Schritt „gefüttert und geupdatet“
- **Anwendungsbeispiele:**
  - Sentiment Analyse
  - Syntax Parsing



Supervised ↔ Unsupervised

- Normalfall
- Annotierte Trainingsdaten
  - z.B. [(1, “ungerade“), (2, “gerade“), (3, “ungerade“), (4, “gerade“), …]
- **Anwendungsbeispiele:**
- Classification (Ergebnis ist Kategorie):
  - Die geg. Zahl ist **gerade** / **ungerade**
  - Der Hund in dem Foto ist ein **Pudel** / **Schäferhund** / **Dackel** / **Rottweiler**
- Regression (Ergebnis ist ein Wert):
  - Die Person in dem Foto wiegt **x kg**
  - Das Haus mit geg. Daten kostet **x €**

- Meist deutlich komplexer
- Trainingsdaten sind nicht annotiert
- Das NN ermittelt angemessene Outputs eigenständig während dem Trainingsprozess
- **Anwendungsbeispiele:**
- Clustering:
  - Onlinekunden gemäß Shopping-verhalten sinnvoll gruppieren
- Association:
  - Kunden die X kaufen, kaufen auch Y

Denkbare Neuronale Netze

	Supervised	Unsupervised
Feedforward	✓	✓
Recurrent	✓	✓

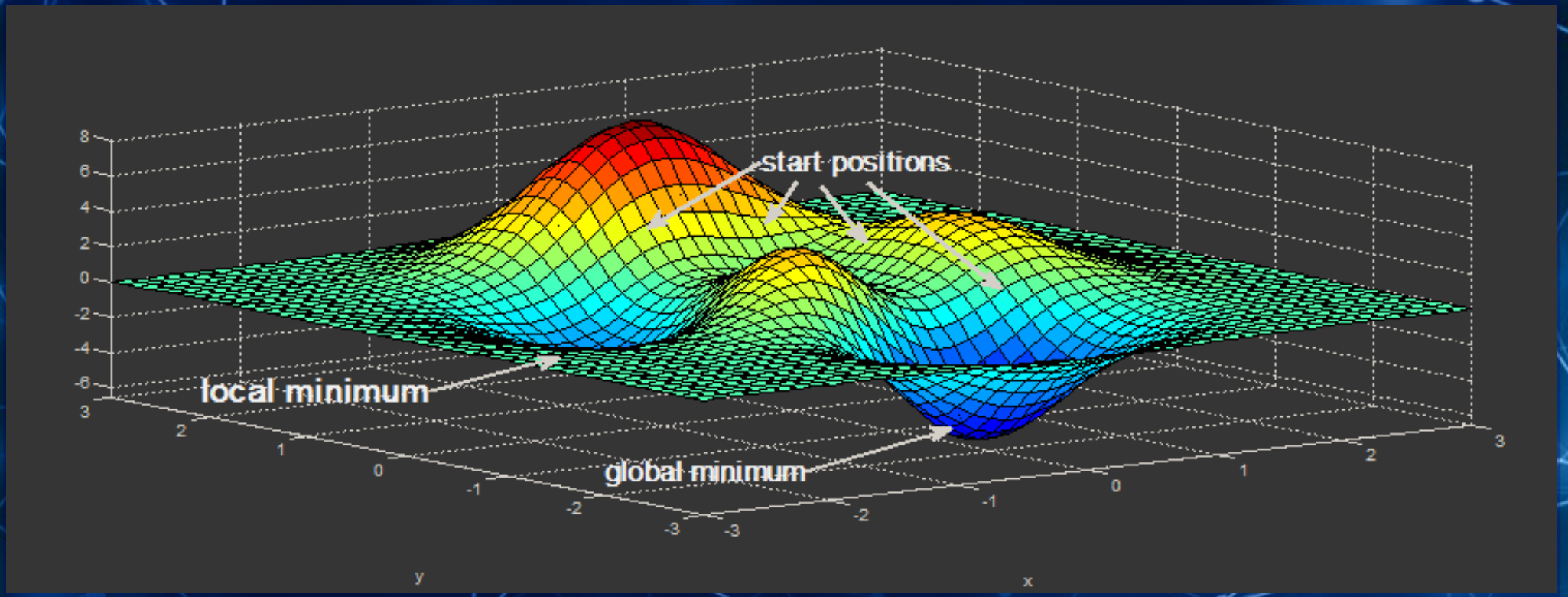
Kurzer Rückblick

- Neuron
- Kanten mit Gewichten
- Aktivierungsfunktion
- Feedforward ↔ Recurrent
- Supervised ↔ Unupervised

Der Trainings-Algorithmus

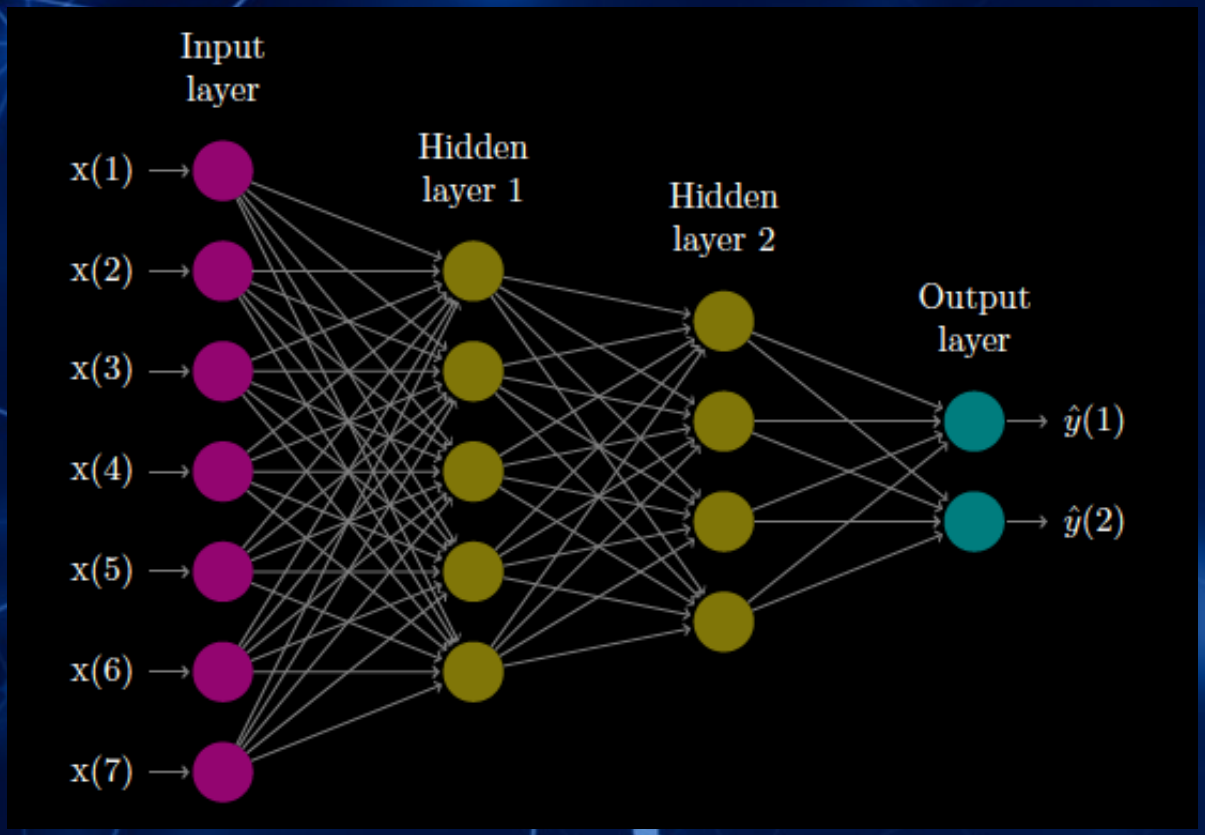
Die Mathematik neuronaler Netze

Optimierungsproblem → Suche nach globalem Minimum



… in beliebig vielen Dimensionen

Die Mathematik neuronaler Netze



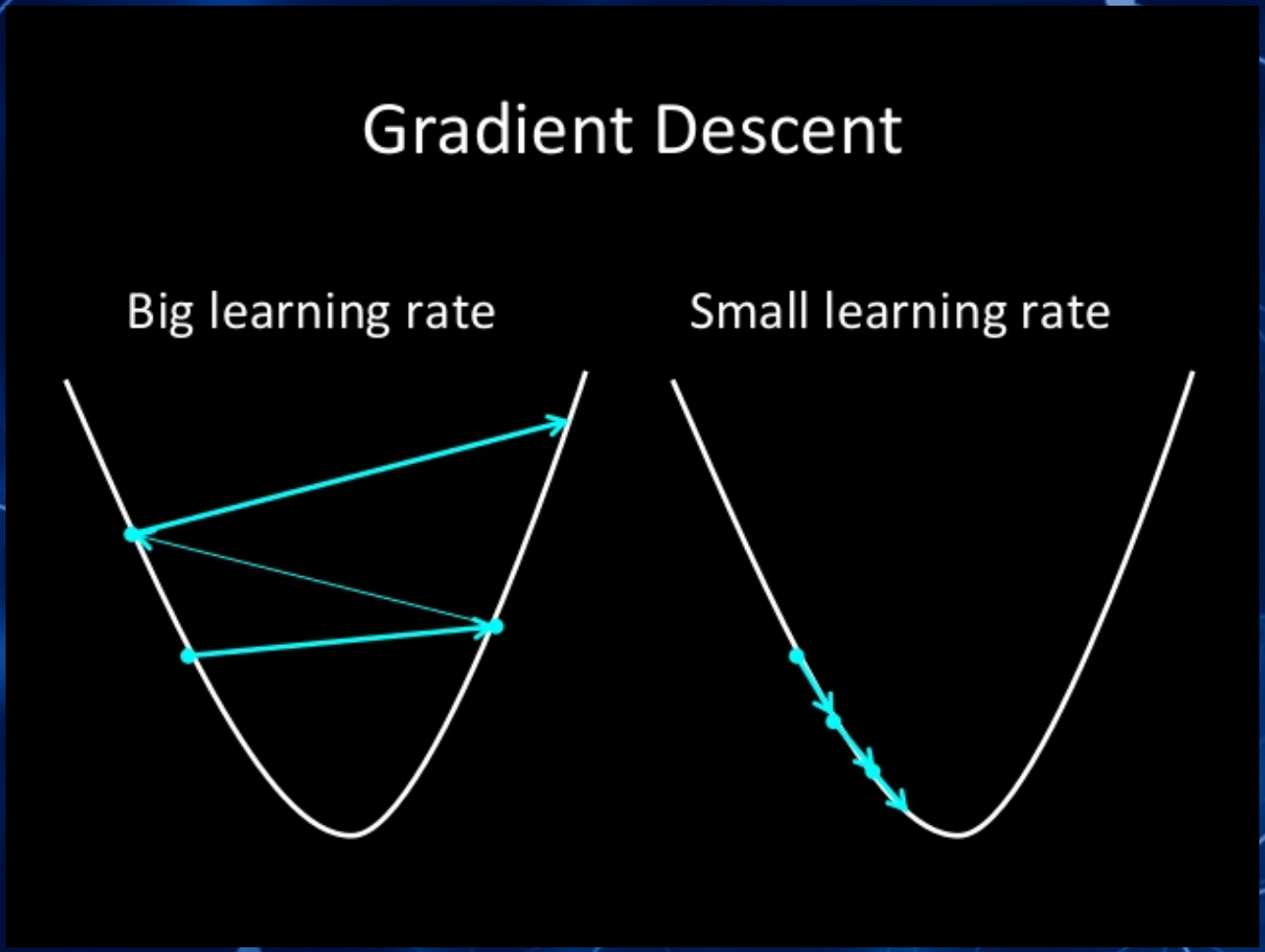
Ein Neural Network mit

- 7 Features
- 2 Hidden Layers
- 2 Output States

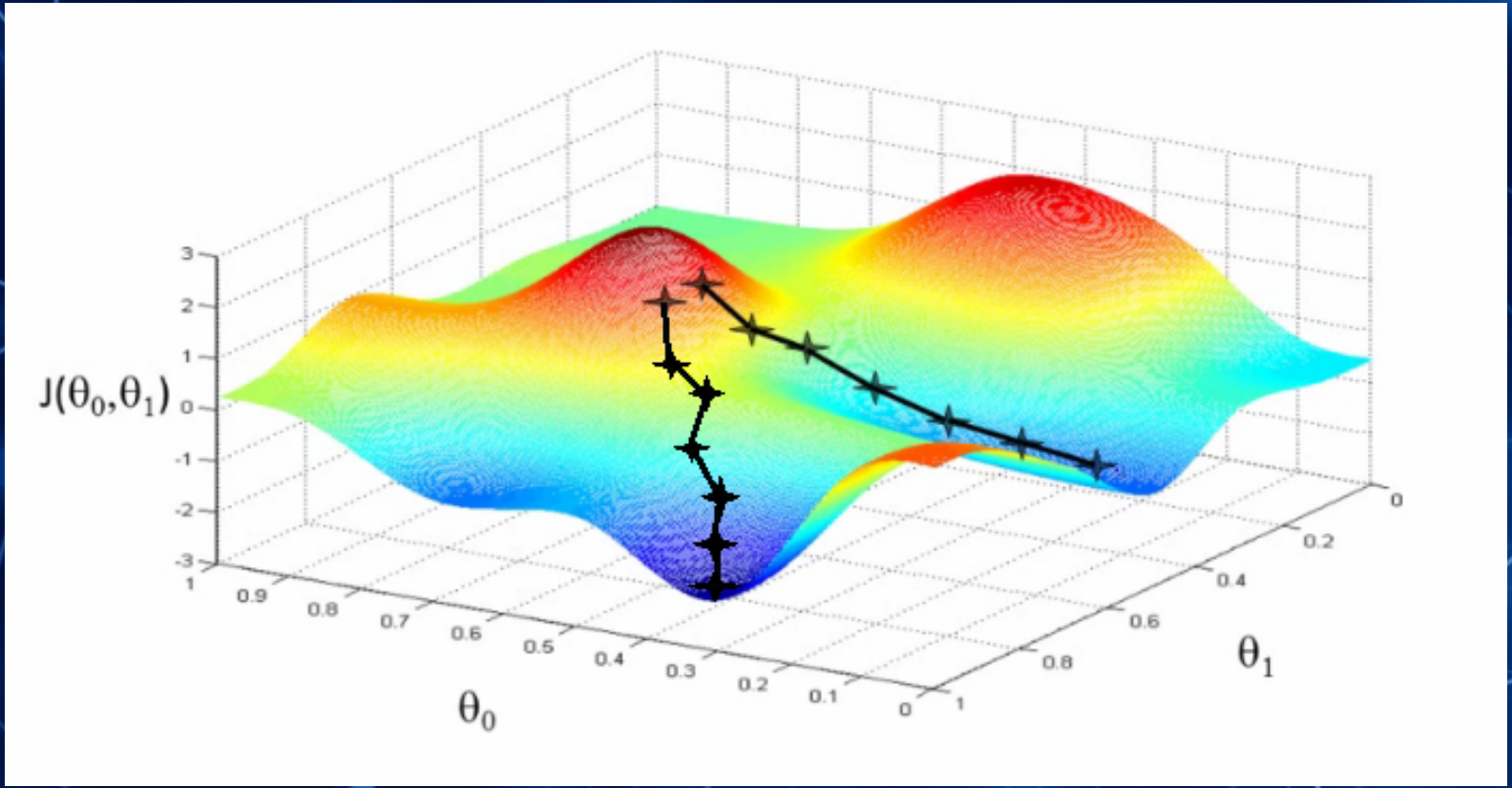
➔ Lernen / Trainieren bedeutet hier:

Optimiere die Gewichte an jeder Kante sodass der Out-Of-Sample Fehler (d.h. die Abweichung vom theoretisch perfekten Output) minimiert wird.

Gradient Descent

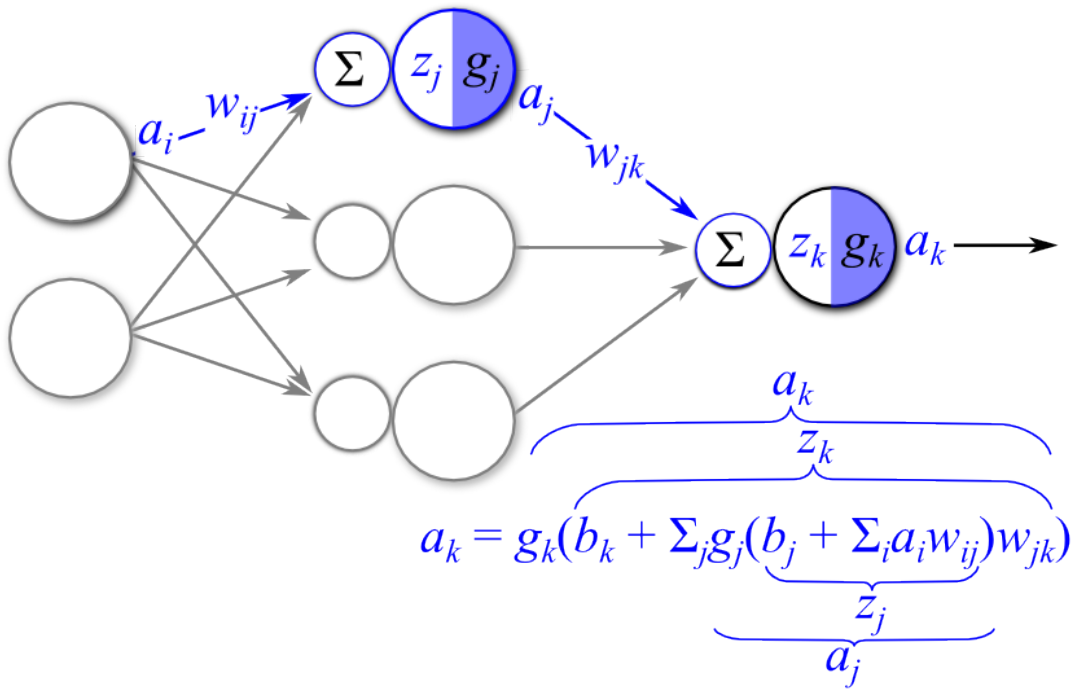


Gradient Descent

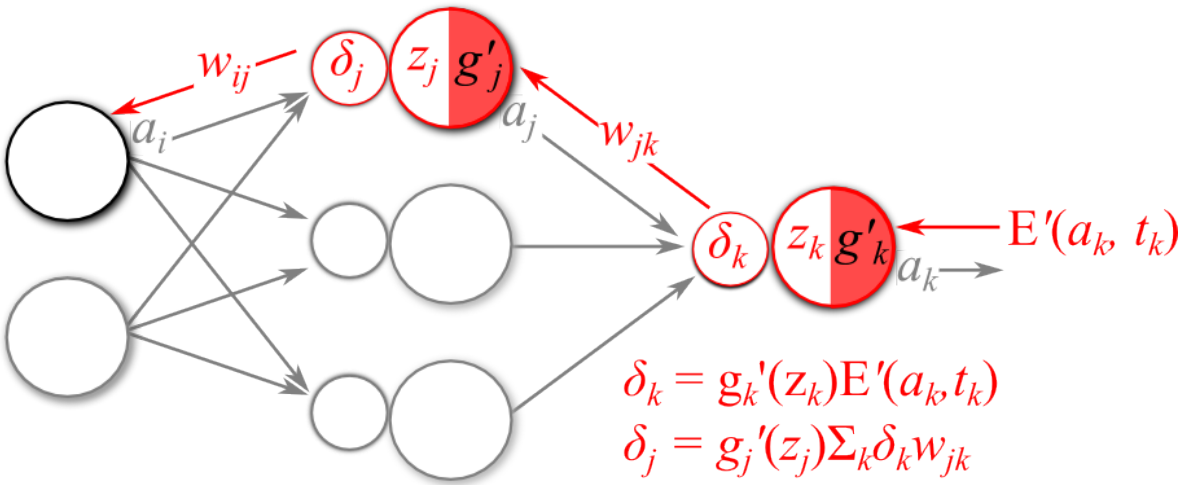


Backpropagation (Fehlerrückführung)

I. Forward-propagate Input Signal

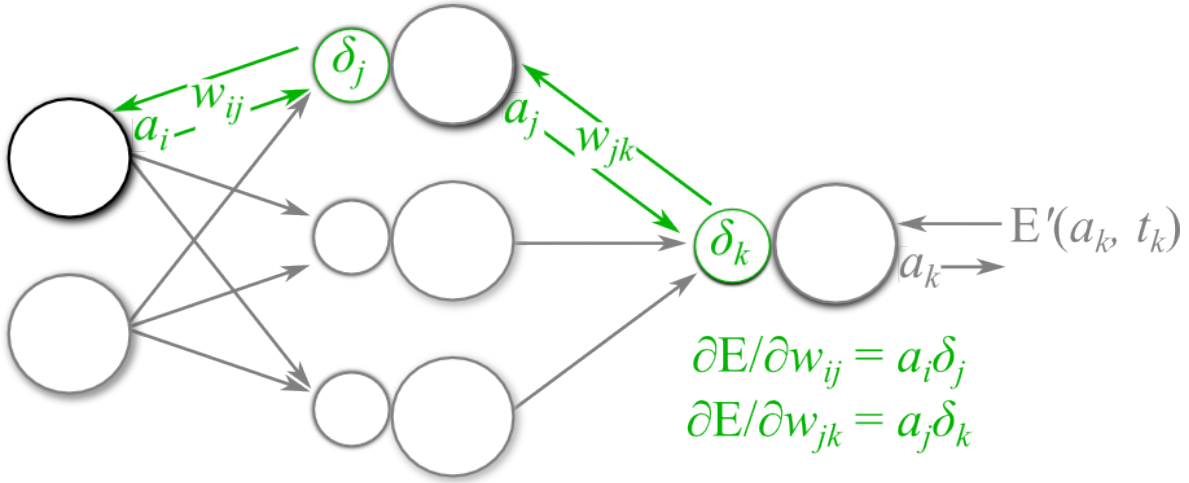


II. Back-propagate Error Signals



Backpropagation (Fehlerrückführung)

III. Calculate Parameter Gradients



$\partial E/\partial w_{ij} = a_i \delta_j$

$\partial E/\partial w_{jk} = a_j \delta_k$

IV. Update Parameters

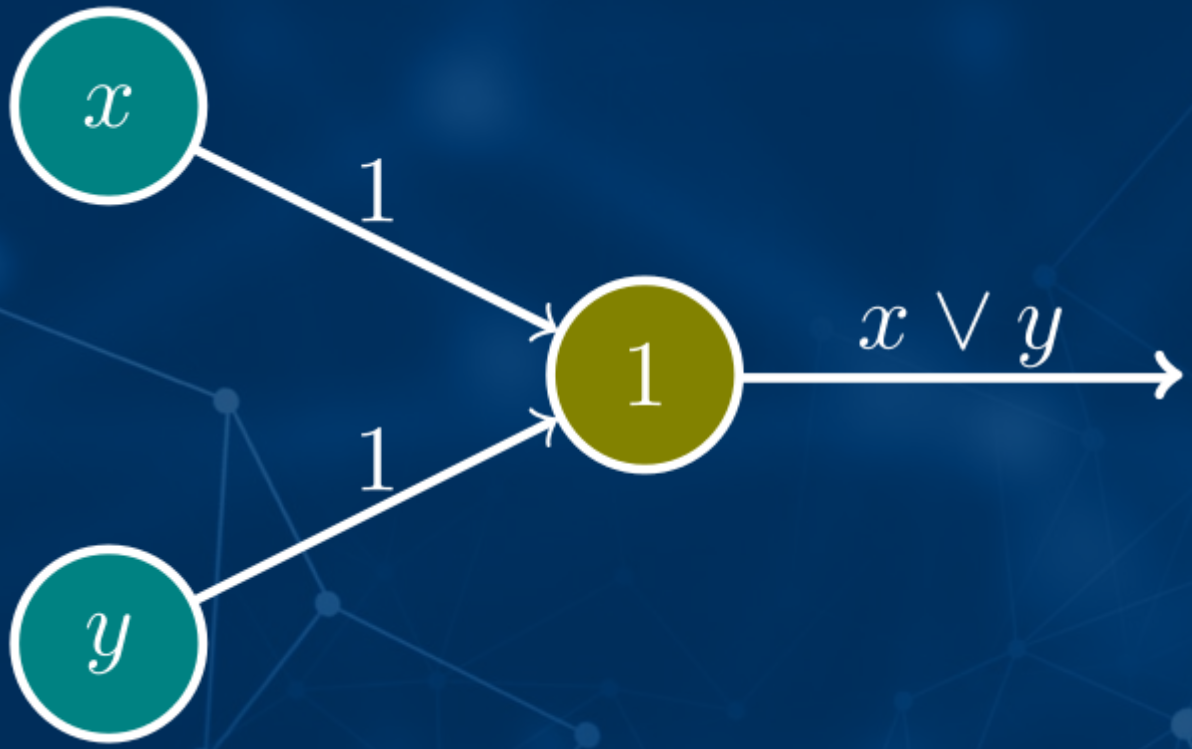
$w_{ij} = w_{ij} - \eta(\partial E/\partial w_{ij})$

$w_{jk} = w_{jk} - \eta(\partial E/\partial w_{jk})$

for learning rate η

Patient Zero: Das Perzeptron

Einfachstes denkbares NN mit nur 1 Schicht



https://de.wikipedia.org/wiki/Perzeptron

Praktischer Teil

Unser Werkzeug:

Was ist PyTorch?

- Imperative Programmierung mit allen gängigen Python-Statements
- GPU Nutzung (keine manuelle Grafikkartenprogrammierung notwendig)
- Hohe Abstraktionsebene, daher wenig Mathematik für den User
- ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ Sehr einfach zu debuggen ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥ ♥

Companies & Universities developing PyTorch

Übung 1: Gradient Descent

Beschreibung

In dieser Übung sind einfache Daten von x-Werten [1, 2, 3] und y-Werten [2, 4, 6] gegeben.

Wir verwenden ein sehr einfaches NN und trainieren es mit dem Ziel, die (unbekannte) lineare Funktion zu emulieren, mit der die y-Werte aus den x-Werten erzeugt wurden.

Aufgaben

- Mache dich mit der Programmstruktur vertraut
- Lassen sich durch die Verwendung von mehr Trainingsdaten bessere Ergebnisse erzielen?
- Lassen sich durch die Veränderung der Lernrate bessere Ergebnisse erzielen?
- Lassen sich durch mehr Trainingsdurchläufe bessere Ergebnisse erzielen?
- Verändere die x- & y-Daten entsprechend einer beliebigen anderen linearen Funktion (z.B. $f(x) = 4x$). Wie reagiert das NN?
- Warum kann dieses NN z.B. keine quadratische Funktion $f(x) = x^2$ erlernen?

exercise_1.py

Übung 2 A: NN als Klasse

Beschreibung

In dieser Übung implementieren wir eine neue Klasse, die von PyTroch‘s mitgelieferter Module-Klasse erbt. Wir nutzen eine Instanz dieser neuen Klasse und trainieren diese dazu, anhand von gegebenen Gesundheitsdaten festzustellen, ob eine Person Diabetiker ist/wird oder nicht.

Die Daten liegen vor in Form von data/diabetes.csv:

Aufgaben

- Mache dich mit der Struktur der Trainings-Daten vertraut (data/diabetes.py)
- Mache dich mit der Struktur des Networks und der Klasse vertraut
- Lassen sich durch eine größere Anzahl von Layern bessere Ergebnisse erzielen?
- Lassen sich durch tiefere Layer bessere Ergebnisse erzielen?
- Wo wird die Lernrate festgelegt und welche Rolle spielt sie?

exercise_2a.py

Übung 2 B

Beschreibung

Ein NN soll trainiert werden, das Geschlecht zu einem gegebenen Namen auszugeben.

Gegeben ist die folgende Liste als data/names_original.csv:

Quentin,m
Jean,m
Carissa,f
Jillian,f
Gerald,m
…
…

Aufgaben

- Lässt sich das NN aus Übung 2 A auf diese neue Aufgabenstellung anpassen?
- Wie könnte man den Trainings-Input modifizieren, um bessere Ergebnisse zu erreichen?

exercise_2b.py

Übung 3: Textgenerierung

Beschreibung

In dieser Übung trainieren wir ein Recurrent Neural Network dazu, eigenständig Texte zu generieren.

Wir implementieren ein LSTM-basiertes Sprachmodell (d.h. mit Lang- & Kurzzeitgedächtnis). Dieses trainieren wir mit verschiedenen Korpora und lassen es auf Grundlage der darin erkannten Muster selber Texte erzeugen.

Doch anstatt einfach nur einen „Zufallsgenerator“ auf eine Wortmenge anzuwenden, Trainieren wir das NN auf der Ebene von Buchstaben. Das NN betrachtet also lediglich Buchstabenfolgen und lernt selbstständig Regeln zu erkennen, wie z.B. dass gewisse Buchstaben in Gruppierungen auftreten (Wörter), dass diese Gruppierungen mit bestimmten Regeln zueinander positioniert sind (Grammatik) und dass gelegentlich Kommas/Punkte auftreten (Syntax).

Aufgaben

- Mache dich mit der Kommandozeilenbedienung des Tolls vertraut
  https://github.com/mcleonard/pytorch-charRNN
- Trainiere das Network mit eigenen Dateien oder mit den gegebenen Trainingsdaten im data-Verzeichnis
- Verwende das Trainierte Network um Texte zu generieren

exercise_3.py


Fragen?

Vielen Dank!

Vielen Dank!

Alternative Deeplearning Frameworks

- Tensorflow
- Keras
- MXnet
- Caffe
- Caffe2
- Theano
- Torch

Links & Literaturverzeichnis

Zum Weiterlesen:

PyTorch allgemein: http://pytorch.org/tutorials/beginner/pytorch_with_examples.html

Natural Language Processing with Pytorch (N-Gram Language Modeling, Word Embeddings, …):
https://github.com/rguthrie3/DeepLearningForNLPInPytorch/blob/master/Deep%20Learning%20for%20Natural%20Language%20Processing%20with%20Pytorch.ipynb

Übungen:

Übung 1:	Codebasis von https://github.com/hunkim/PyTorchZeroToAll
Übung 2 A:	Codebasis von https://github.com/hunkim/PyTorchZeroToAll
Übung 2 B:	Namensdaten von https://www.ssa.gov/OACT/babynames/
Übung 3:	Codebasis von https://github.com/mcleonard/pytorch-charRNN

Texte für Übung 3 (20.01.2018):

tx_DE_1_Schneewitchen.txt	http://gutenberg.spiegel.de/buch/-6248/150
tx_DE_2_Mathemathische_Grundlagen_1.txt	http://www.coli.uni-saarland.de/~saurer/lehre/mg1/mg1.html
tx_DE_3_Der_Erlkönig.txt	http://www.literaturwelt.com/werke/goethe/erlkoenig.html
tx_DE_4_Oh_Tannenbaum.txt	http://www.labbe.de/liederbaum/index.asp?themaid=10&titelid=641
tx_DE_5_Pulb_Fiction_Synopsis.txt	http://www.filmstarts.de/kritiken/10126.html
tx_DE_6_Das_Sandmännchen.txt	https://www.golyr.de/sandmaennchen/songtext-sandmann-lieber-sandmann-635416.html
tx_DE_7_Coli_Studiengang_Beschreibung.txt	https://www.uni-saarland.de/campus/studium/studienangebot/az/c/coli.html
tx_EN_1_Declaration_of_Independence.txt	https://en.wikipedia.org/wiki/United_States_Declaration_of_Independence
tx_EN_2_Twitter_Terms_Of_Service.txt	https://twitter.com/en/tos
tx_EN_3_Wikipedia_Tennis.txt	https://en.wikipedia.org/wiki/Tennis
tx_EN_4_Mark Twain_A_Ghost Story.txt	http://www.fairytalescollection.com/MarkTwain/AGhostStory.aspx
tx_EN_5_Sleeping_Beauty.txt	http://www.fairytalescollection.com/TheGrimmBrothers/SleepingBeauty.aspx
tx_EN_6_The_Ugly_Ducking.txt	http://www.fairytalescollection.com/HansChristianAndersen/TheUglyDuckling.aspx
tx_EN_7_Nvidia.txt	https://blogs.nvidia.com/blog/2018/01/22/ai-heart-arrhythmia/

Links & Literaturverzeichnis

Grafiken & Medien (20.01.2018) in Reihenfolge der Verwendung:

Folie 1
http://pytorch.org/static/img/pytorch-logo-light.svg

Folie 3
https://i.redd.it/il81sjck8vgy.png
https://i.imgflip.com/174t2v.jpg

Folie 4
https://cdn.technologyreview.com/i/images/alpha.gox1200.jpg?sw=1200
https://cdn2.techworld.com/cmsdata/slideshow/3636755/go_game_nature_video_youtube_thumb650.jpg
https://pbs.twimg.com/media/DMgqXotWAAcLxgQ.jpg

Folie 5
https://deepmind.com/static/v0.0.0/images/deepmind_logo.png

Folie 7
https://4.bp.blogspot.com/-evE8fEJZOVM/WfuG7lOk8-I/AAAAAAAACIo/iBszpNVEnT4Y2MtzH2_Pd_igdASc1AFbQCLcBGAs/s1600/image2.jpg

Folie 8
https://www.youtube.com/watch?v=mmj3iRIQw1k
https://www.youtube.com/watch?v=9VC0c3pndbI

Folie 9
https://i.kinja-img.com/gawker-media/image/upload/s--SCKUU_aC--/c_fit,fl_progressive,q_80,w_636/1298342456783405227.gif
https://vignette1.wikia.nocookie.net/supermarioglitchy4/images/4/4a/1.png/revision/latest?cb=20140728061754

Folie 11
https://yourdorn.files.wordpress.com/2017/11/df2071e1084c273fe910863f1b2cfde86f938c161259d95e8fd0ee5d893b0c6f.jpg

Folie 13
https://ethervision.net/wp-content/uploads/2014/01/neural-network.png
https://edouardfouche.com/img/neural-based-outlier-discovery/autoencoder.png

Links & Literaturverzeichnis

Grafiken & Medien (20.01.2018) in Reihenfolge der Verwendung:

Folie 20 - 27
https://www.youtube.com/watch?v=aircAruvnKk
https://www.youtube.com/watch?v=IHZwWFHWa-w
https://www.youtube.com/watch?v=Ilg3gGewQ5U
https://www.youtube.com/watch?v=tIeHLnjs5U8

Folie 28
http://www.cs.iusb.edu/~danav/teach/c463/forward_nn.gif
http://www.cs.iusb.edu/~danav/teach/c463/rec_nn.gif

Folie 33
http://sf.anu.edu.au/~vvv900/gaussian/ts/Freiburg2013_3D-example.png

Folie 34
https://matthewdixon2015.files.wordpress.com/2016/09/screen-shot-2016-09-17-at-12-23-09-pm.png

Folie 35
https://media.licdn.com/mpr/mpr/AAEAAQAAAAAAAA0oAAAAJDUzMTBlMDdjLWM0ZmMtNDJkNS1hODk3LTAzYTllMDUwZmY1OQ.jpg

Folie 36
http://blog.datumbox.com/wp-content/uploads/2013/10/gradient-descent.png

Folie 37
https://theclevermachine.files.wordpress.com/2014/09/fprop_bprop5.png

Folie 38
https://theclevermachine.files.wordpress.com/2014/09/fprop_bprop5.png

Folie 39
https://upload.wikimedia.org/wikipedia/commons/thumb/4/4b/Perceptron-or-task.svg/639px-Perceptron-or-task.svg.png

Folie 40
http://www.imagenspng.com.br/wp-content/uploads/2015/07/minions-02.png

Folie 41
http://pytorch.org/static/img/pytorch-logo-light.svg

Folie 43
Screenshot von http://pytorch.org/