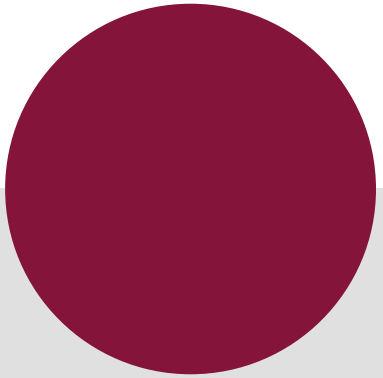
.msg value – inspired by people



SAKI 2021 Die Mathematik des Deep Learning

John Loutzenhiser
Principal IT Consultant – Public Sector

+49 (0) 173 /859 4235
John.Loutzenhiser@msg.group

Robert Pinzinger
Lead IT Consultant – Public Sector

+49 (0) 1511 225 815 5
Robert.Pinzinger@msg.group

msg systems ag
Robert-Buerkle-Str. 1, 85737 Ismaning
Germany

www.msg.group

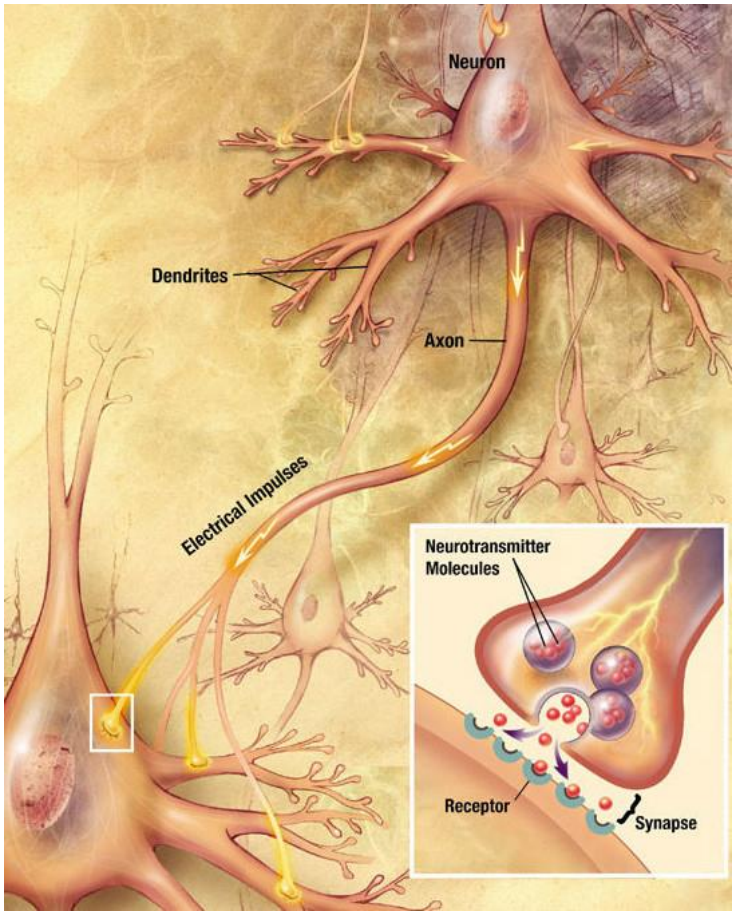
value – inspired by people

Überblick

- Biologische Motivation
- Das mathematische Neuron
- Wie trainiert man ein Netz?
- Besondere Architekturen
- Was ist Transfer-Learning?
- Adversarial Noise

Biologische Motivation

- Neuronen kommunizieren miteinander
- Die eingehenden Impulse werden kombiniert und lösen beim Überschreiten einer Schranke eine Reaktion aus



Quelle: https://en.wikipedia.org/wiki/Neuron

Das mathematische Neuron

Die Formel

- w ist ein Vektor von Gewichten
- x ist ein Vektor von Eingangsdaten
- b ist eine Zahl und bestimmt die Nullstelle
- Die Formel für z gibt an, dass die Eingangsdaten x mit Gewichten w gewichtet werden, die gewichteten Eingangsdaten werden addiert, und das Ergebnis wird um b verschoben

$$z = w^T x + b$$

Das mathematische Neuron

Der Bezug zur Geometrie

- Der Term $w^T x$ kommt in der Berechnung des Cosinus zwischen zwei Vektoren x und w vor
- $w^T x = 0$ wenn w senkrecht auf x steht

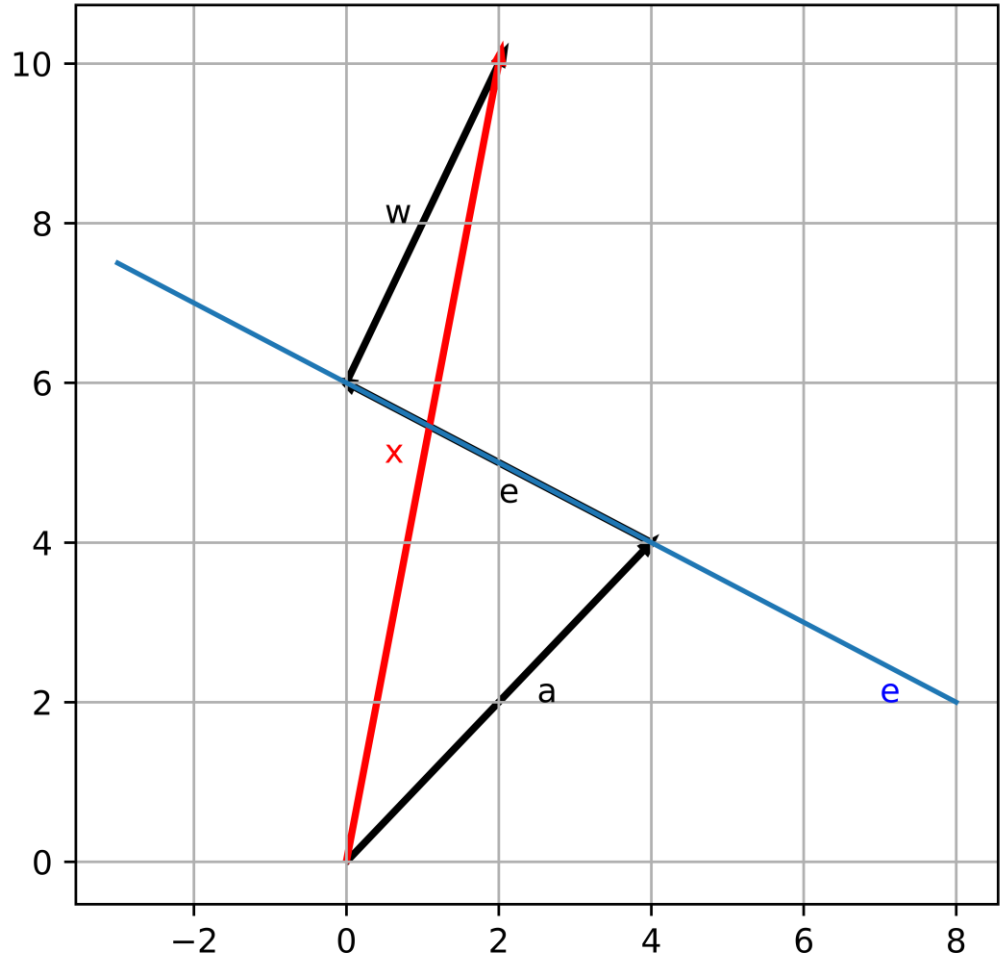
$$\cos(w, x) = \frac{w^T x}{\|w\| \|x\|}$$

Das mathematische Neuron

Der Bezug zur Geometrie (II)

- Interpretation: w ist eine Normale, die senkrecht auf einer Hyperebene e steht, und z testet, auf welcher Seite der Hyperebene e x liegt

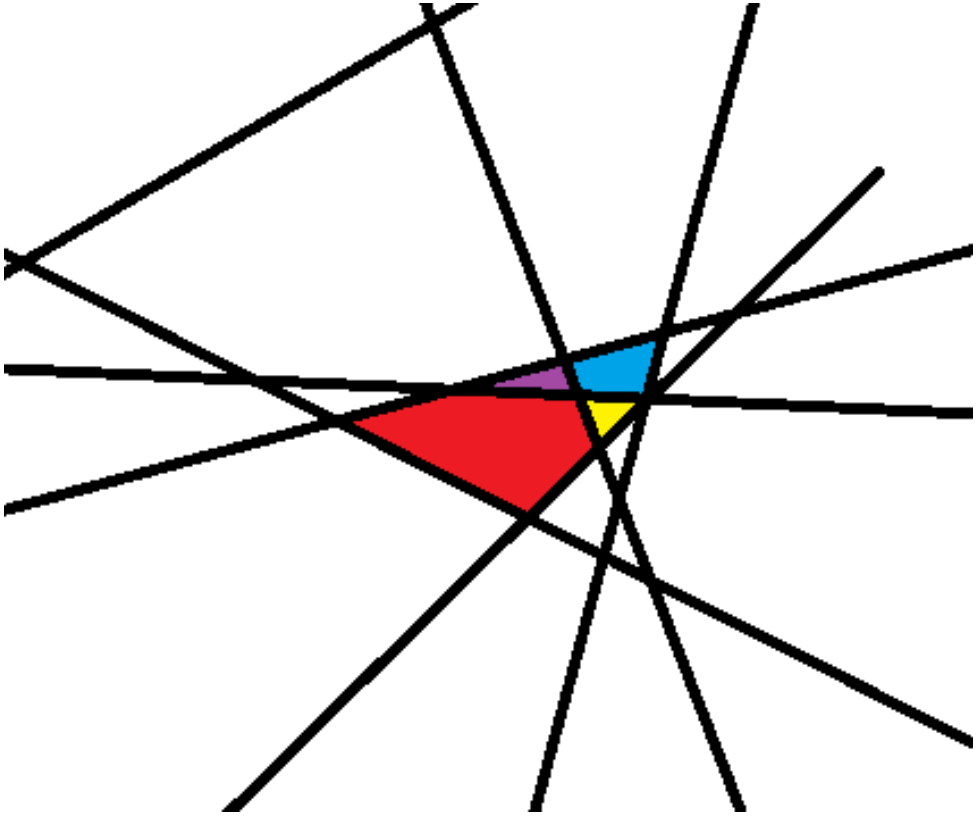
$$z = w^T x + b$$



Das mathematische Neuron

Der Bezug zur Geometrie (III)

- Mit einem Neuron kann man testen, auf welcher Seite einer Hyperebene man liegt. Mit vielen Neuronen kann man testen, auf welcher Seite vieler Hyperebenen man liegt.
- Man kann mit hinreichend vielen Neuronen jede Form abbilden (Universalapproximator)



Das mathematische Neuron

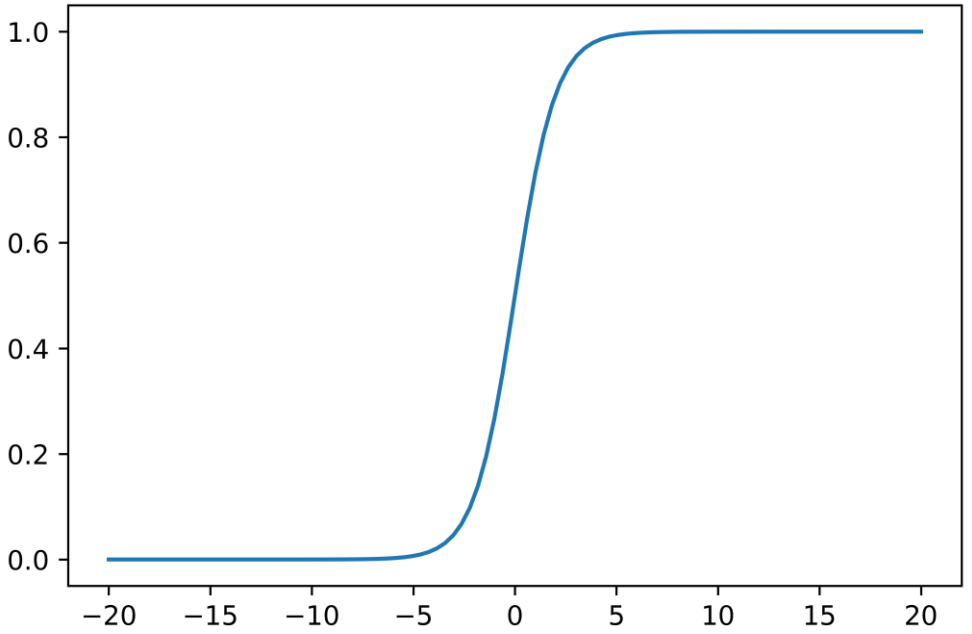
Das Problem mit der Nichtlinearität

- Problem: für nichtlineare Formen braucht man sehr viele Neuronen.
- Lösung: nichtlineare Aktivierungsfunktionen σ
- Betrachte $\sigma(z)$ anstelle von z

Das mathematische Neuron

Aktivierungsfunktionen

- Sigmoid
- Vorteil: Wertebereich zwischen 0 und 1
- Vorteil: Ableitung sehr einfach zu berechnen
- Nachteil: e^{-z} ist rechentechnisch sehr teuer
- Nachteil: Ableitung immer zwischen 0 und 1 -> verschwindende Gradienten



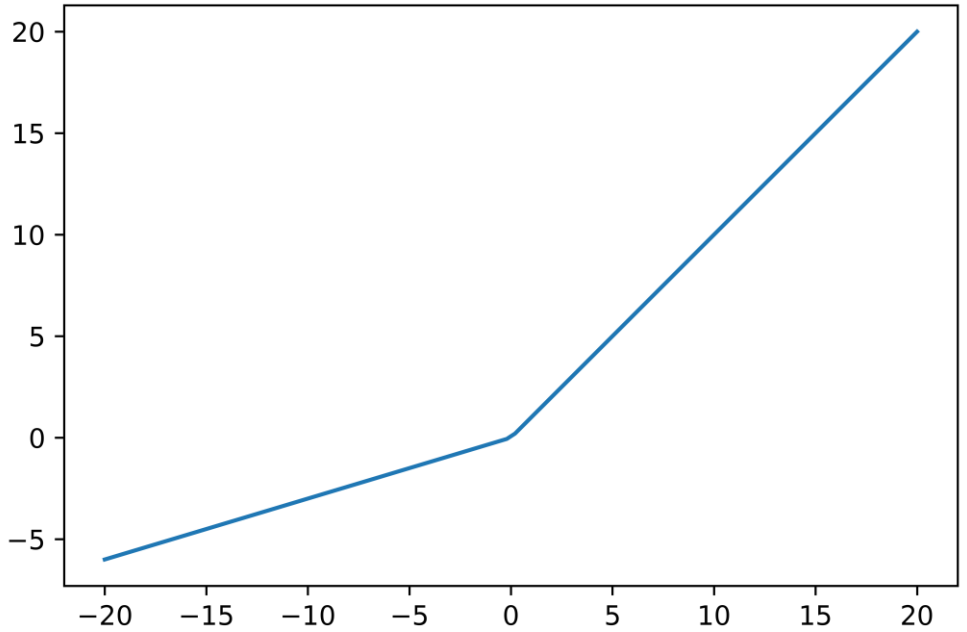
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big)$$

Das mathematische Neuron

Aktivierungsfunktionen (II)

- Leaky ReLU
- Vorteil: einfach zu berechnen
- Vorteil: Ableitung muss nicht berechnet werden
- Nachteil: unbegrenzter Wertebereich



$$\sigma(z) = \begin{cases} z \text{ wenn z>0} \\ 0{,}3z \text{ sonst} \end{cases}$$

Wie trainiert man ein Netz?

Grundlagen zum Training

- Definiere einen Abstandsbegriff
- Minimiere den Abstand zwischen den Vorhersagen $f(x)$ und den wahren Zielwerten y

Wie trainiert man ein Netz?

Abstandsfunktionen

- MSE
- Mittlere Summe der quadratischen Fehler
- Vorteil: stetig differenzierbar
- Vorteil: nach unten bei 0 beschränkt
- Nachteil: übergewichtet große Fehler und lässt kleine Fehler verschwinden (Lösung für Skalenprobleme: betrachte $log(y)$ und $log\big(f(x)\big)$)
- Eignet sich für Regressionsprobleme

$$\text{MSE} = \frac{\sum\big(y - f(x)\big)^2}{n}$$

Wie trainiert man ein Netz?

Exkurs: Shannon Entropy

- Wieviel Information steckt in einer Nachricht? Die Information entspricht der maximalen verlustfreien Kompression
- Die maximale Kompression erreicht man, indem man häufigen Ereignissen kurze Codewörter gibt
- Shannon, C. (1947): A mathematical Theory of Communication

$$E = -\sum p log(p)$$

Wie trainiert man ein Netz?

Exkurs: Kullback Leibler Cross Divergence

- Was ist der Fehler in der Kompression, wenn ich die Verteilung q an Stelle von p annehme?

$$\mathrm{KL} = \sum p \log(p) - \sum p \log(q)$$

Wie trainiert man ein Netz?

Abstandsfunktionen (II)

- Cross Entropy
- Berechne den Fehler, den man macht, wenn man die Verteilung q annimmt, die wahre Verteilung aber p ist
- Vorteil: stetig differenzierbar
- Vorteil: nach unten bei 0 beschränkt
- Nachteil: die Berechnung des Logarithmus ist rechentechnisch sehr teuer
- Nachteil: Logarithmus von 0 nicht definiert (Tensorflow)
- Eignet sich für Klassifikationsprobleme

$$CE = -\sum p log(q)$$

Wie trainiert man ein Netz?

Der Gradient

- Definiere eine Loss-Funktion, die den Abstand zwischen den Vorhersagen $f(x)$ und den wahren Werten y als Funktion der Gewichte w berechnet
- Annahme: Taylor-Approximation
- Minimiere den Loss mit dem Gradientenabstiegsverfahren (Stochastic Gradient Descent, Adam Optimizer, AdaProp, RMSProp)
- Skaliere das Update Δw mit einer Lernrate ε
- Wende die Kettenregel für die Berechnung von $\frac{\partial}{\partial w} Loss(w)$ an

$$Loss(w + \varepsilon \Delta w) \approx Loss(w) + \varepsilon \frac{\partial}{\partial w} Loss(w) \Delta w$$

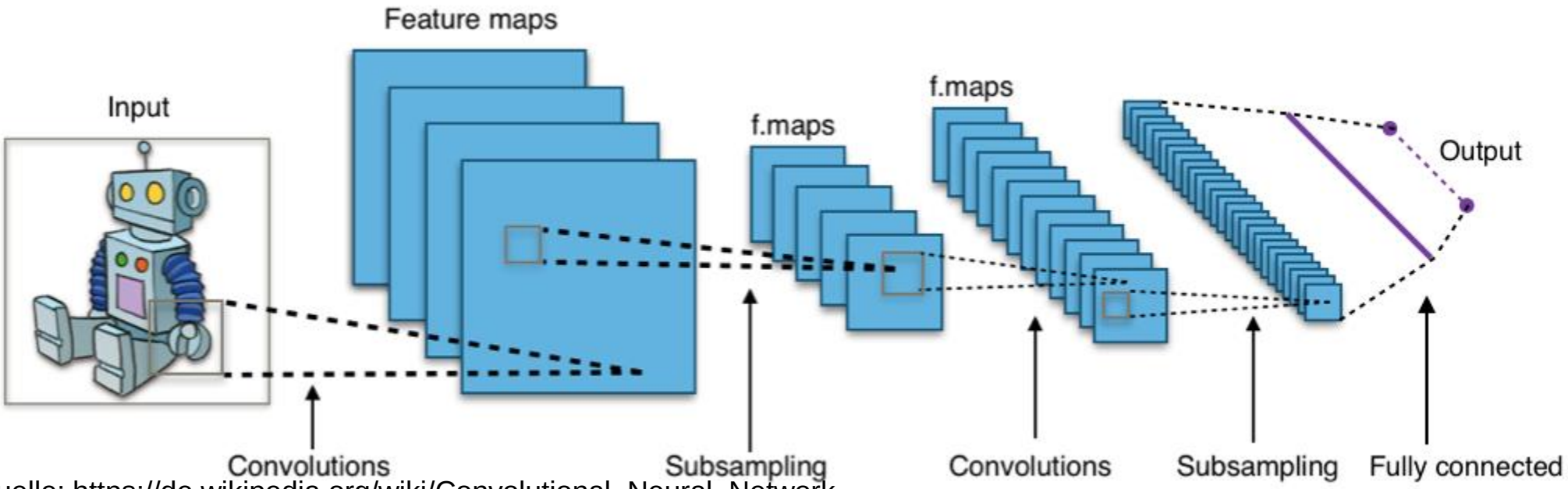
Wie trainiert man ein Netz?

Probleme

- Problem: Multiplikation von vielen Werten zwischen -1 und 1 -> Gradienten verschwinden
- Problem: viele Matrix-Vektor-Produkte -> GPUs zur Parallelisierung (CUDA)
- Problem: GPUs haben haben „wenig“ RAM und die Anbindung über PCI ist langsam -> Federated Learning (Verteilung auf viele GPUs)

Besondere Architekturen

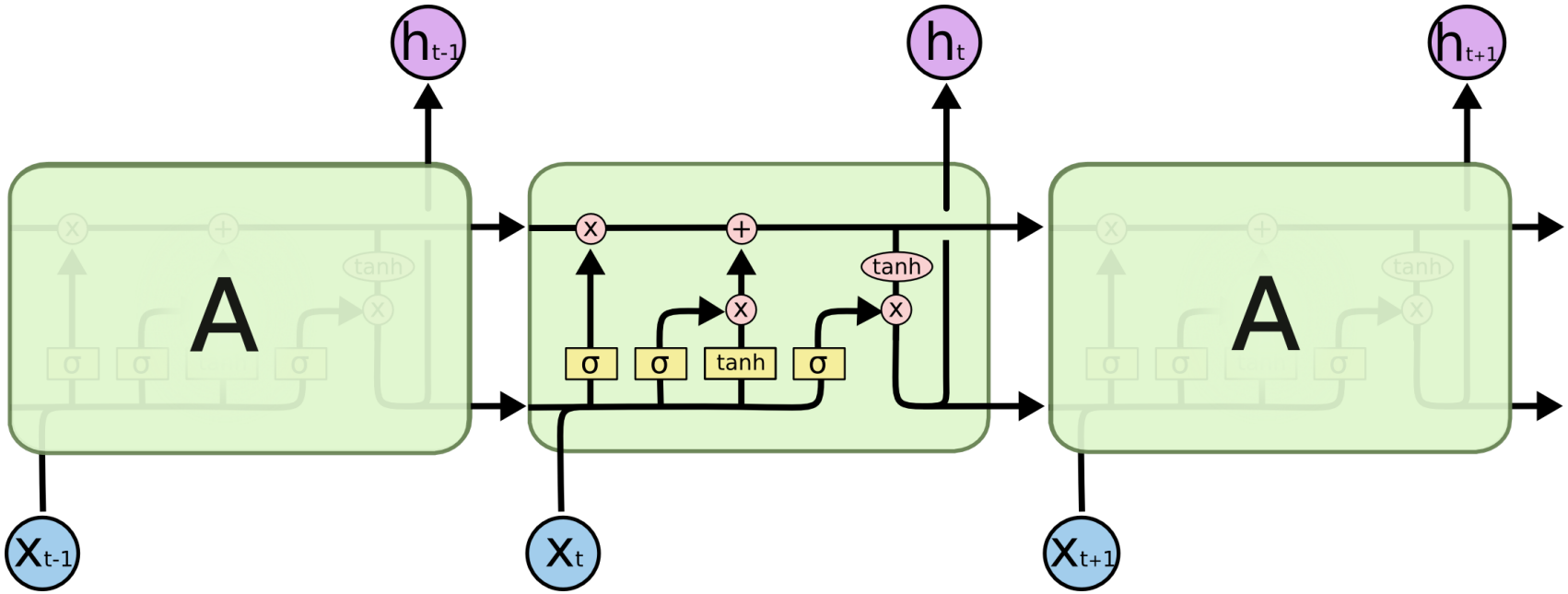
- Convolutional Neural Networks
- Standard für Bilder (1d, 2d, 3d)



Quelle: https://de.wikipedia.org/wiki/Convolutional_Neural_Network

Besondere Architekturen

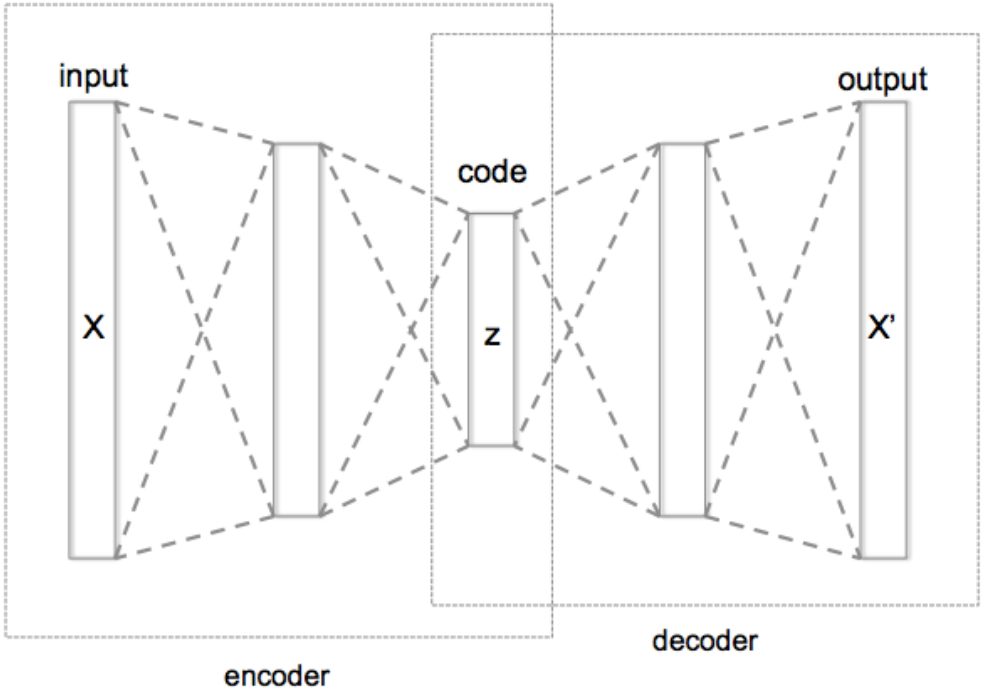
- Recurrent Neural Networks
- Standard für sequentielle Daten



Quelle: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Besondere Architekturen

- Autoencoder
- Dimensionsreduktion



Quelle: https://en.wikipedia.org/wiki/Autoencoder

Besondere Architekturen

Exkurs: Glove

- One-hot-encodiere die einzelnen Worte, aggregiere daraus Vektoren für Ein- und Ausgabe
- Benutze Autoencoder-Architektur

Jedes Wort im Satz ist in der Umgebung jedes anderen Wortes.
Jedes Wort im Satz ist in der Umgebung jedes anderen Wortes.
Jedes Wort im Satz ist in der Umgebung jedes anderen Wortes.
….
Jedes Wort im Satz ist in der Umgebung jedes anderen Wortes.
Jedes Wort im Satz ist in der Umgebung jedes anderen Wortes.
Jedes Wort im Satz ist in der Umgebung jedes anderen Wortes.

Meaning is not in things but between them (Norman Brown)

Was ist Transfer Learning?

- Interpretation: in einem tiefen neuronalen Netz ist das eigentliche Modell die letzte Schicht, und alle Schichten davor sind nur Data Preprocessing
- Transfer Learning: benutze ein bekanntes Modell M aus der Domäne, berechne zu den Trainingsdaten die Ausgabe der vorletzten Schicht von M, trainiere auf diese Daten ein spezielles Modell
- Vorteil: schnelles Training
- Vorteil: „bekannte“ Benchmark
- Nachteil: spezifische Modelle sind in der Regel besser
- Beispiele: VGG16, ELMo

Adversarial Noise

- Statistisches Rauschen in den Daten kann dazu führen, dass ein Datenpunkt auf der falschen Seite der Hyperebene eingeordnet wird
- Besonders relevant bei Bildern
- Lösungsansätze: trainiere mit Noise und füge im Betrieb Noise ein

Quelle: https://towardsdatascience.com/what-are-adversarial-examples-do-they-exist-for-humans-9572e3910219

John Loutzenhiser
Principal IT Consultant – Public Sector

+49 (0) 173 /859 4235
John.Loutzenhiser@msg.group

Robert Pinzinger
Lead IT Consultant – Public Sector

+49 (0) 1511 225 815 5
Robert.Pinzinger@msg.group

msg systems ag
Robert-Buerkle-Str. 1, 85737 Ismaning
Germany

www.msg.group

value – inspired by people