



**HOCHSCHULE
MITTWEIDA**
University of Applied Sciences

Einführung in Transformer Netzwerke

Der Weg vom einfachen neuronalen Netz zu Transformer
Netzwerken

Thomas Davies, B.Eng., B.Sc.

hs-mittweida.de

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Kunst



Abbildung: Bild von Jason Allen. Sieger des *State Fair Fine Arts* Kunstwettbewerbs

Kunst

Jason Allen verwendete Künstliche Intelligenz

The image shows a screenshot of a news website, likely CNN Business, displaying several headlines related to AI-generated art. At the top left, there is a smaller box with the headline "The Ban Hammer Comes Out for AI-Generated Art". Below it, the main navigation bar includes links for Markets, Tech, Media, Success, Perspectives, and Videos. The main content area features two large, side-by-side news articles. The article on the left is titled "AI won an art contest, and artists are furious" and is by Rachel Metz, published on September 3, 2022. It includes a small profile picture of the author. The article on the right is titled "AI art banned from art communities as websites are flooded with generative pics" and is also by Rachel Metz, published on September 15, 2022. This second article includes a "Fear & Greed Index" graphic showing a value of 62, with a note that "Greed is driving the US market". Both articles mention the artist Jason Allen and his AI-generated artwork.

The Ban Hammer Comes Out for AI-Generated Art

By Adrianna Nine on September 15, 2022 at 9:18 am | [Comments](#)

CNN BUSINESS Markets Tech Media Success Perspectives Videos

Markets →

	DOW	S&P 500	NASDAQ
	33,779.97	3,956.27	11,050.38
	0.10% ▲	0.23% ▼	0.86% ▼

Fear & Greed Index →

Latest Market News →

Greed is driving the US market

America faces a possible rail strike

Retailers warn of 'self-inflicted' e

Bob Iger named Disney CEO again

AI won an art contest, and artists are furious

By Rachel Metz, CNN Business
Published 10:54 AM EDT, Sat September 3, 2022

AI Ethics Left Hanging When AI Wins Art Contest And Human Artists Are Fuming

The Battle Lines Over AI Art

AI art banned from art communities as websites are flooded with generative pics

Should AI generated art be banned?

Abbildung: Die Kunstwelt steht Kopf. Überschriften von Forbes, CNN, etc.

Vom Text zum Bild

Die Fähigkeiten von Dall-E und co.



Abbildung: Dall-E bitte mache mir: *Teddy bears working on new AI research underwater with 1990s technology*

Vom Text zum Bild

Die Fähigkeiten von Dall-E und co.



Abbildung: Unbekannter Input, aber wer wollte nicht mal Homer Simpson in echt sehen?

Warte...

Transformer können auch programmieren!

```
1  <button style="background-color: #yellow; border-color: #orange; border-width:2">
2    Donald Trump
3
4  </button>
```



Donald Trump

Abbildung: Bitte programmiere mir: *A button with the color of donald trump's hair*

So! Fangen wir an?

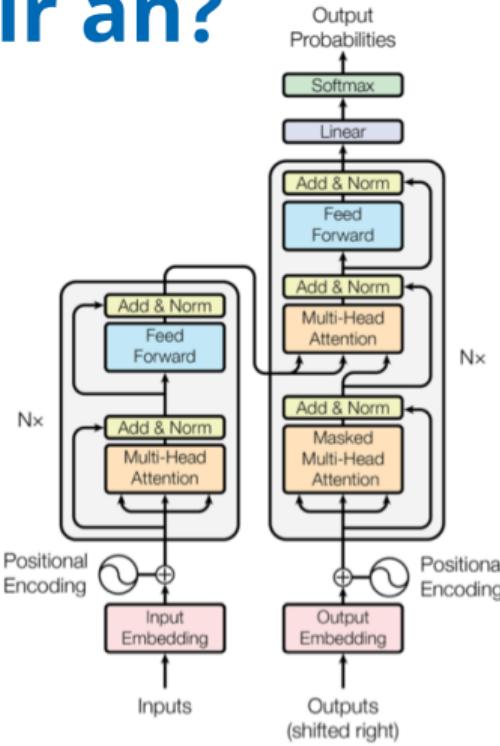


Abbildung: Transformer Diagramm (von Attention is all you need paper)

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Neuronale Netzwerke

Was passiert?

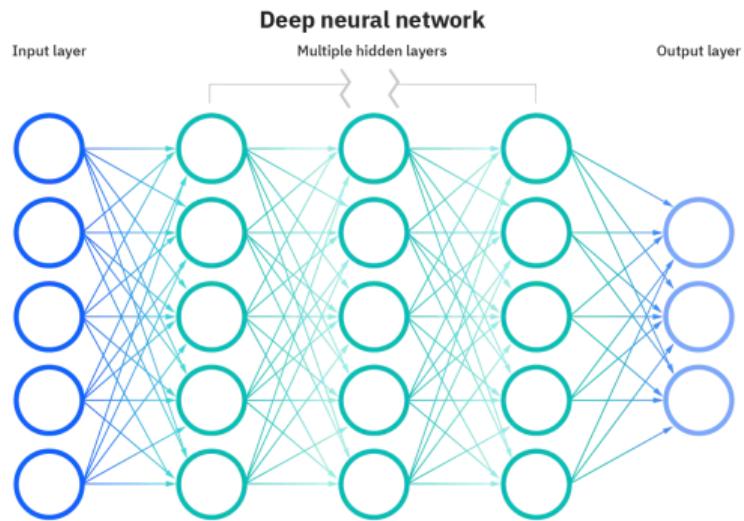


Abbildung: Deep Neural Network (von IBM Webseite)

Partielle Ableitung

Sehen wir uns eine multivariate Funktion $f(x,y)$ als schnelles Beispiel an.

$$f(x,y) = x^2 + x - xy^3 + y^2 \quad (1)$$

Wir können eine multivariate Funktion nach einer Variablen ableiten, indem wir andere alle anderen Variablen als Konstante behandeln. Angenommen wir wollen f nach x ableiten, also

$$\frac{\partial f}{\partial x} = 2x + 1 - y^3 \quad (2)$$

Partielle Ableitung

$$f(x,y) = x^2 + x - xy^3 + y^2 \quad (3)$$

Wir können f aber auch nach y ableiten, also

$$\frac{\partial f}{\partial y} = 2xy^2 + 2y \quad (4)$$

Optimierung Kostenfunktion

Gut, dass wir partiell ableiten können

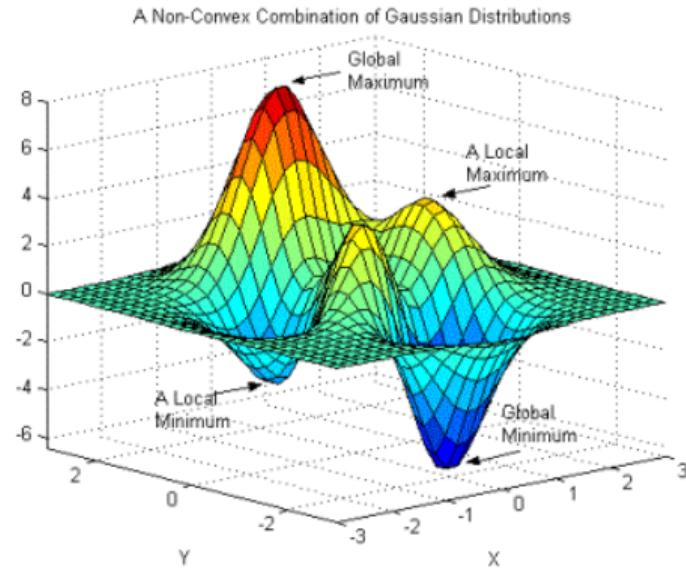


Abbildung: Fiktiver Plot einer Kostenfunktion. (von Wikipedia)

Warum brauchen wir so ein Zeug?

Der Gradient wird häufig mit dem Nabla-Operator gekennzeichnet. Der Gradient eines Vektors w wird als ∇w gekennzeichnet.

Wir wollen unsere Gewichte so anpassen, dass sie den Fehler minimieren, also bewegen wir diese entgegen der Richtung des stärksten Aufstiegs.

Dieses Prinzip nennt sich **Gradient Descent**. Wenn wir zufällige Datenpunkte dem Netz übergeben, wird es zum **Stochastic Gradient Descent** Verfahren.

So sieht das mit dem Gradienten aus

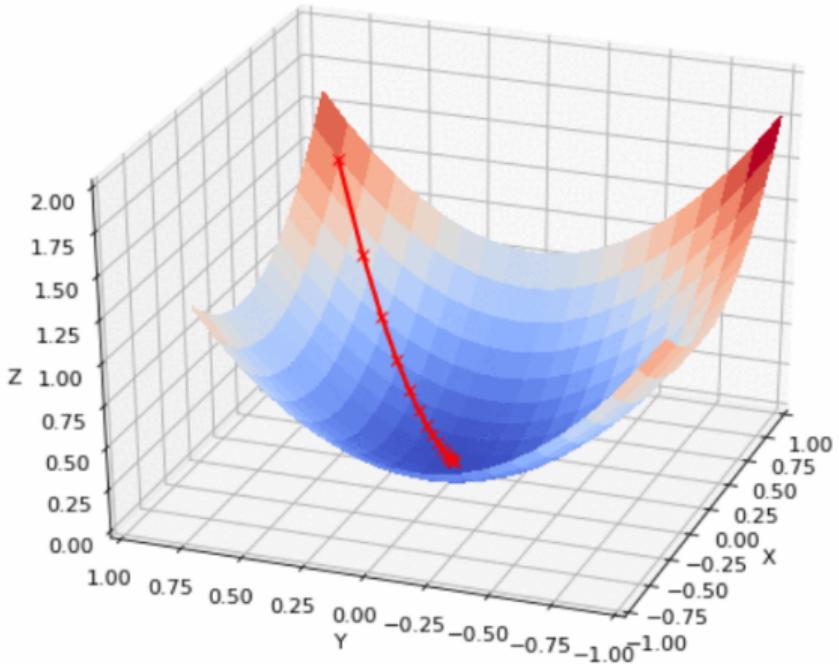


Abbildung: Gradienten Abstieg (von blog.paperspace.com)

Abschließend ein schönes Beispiel

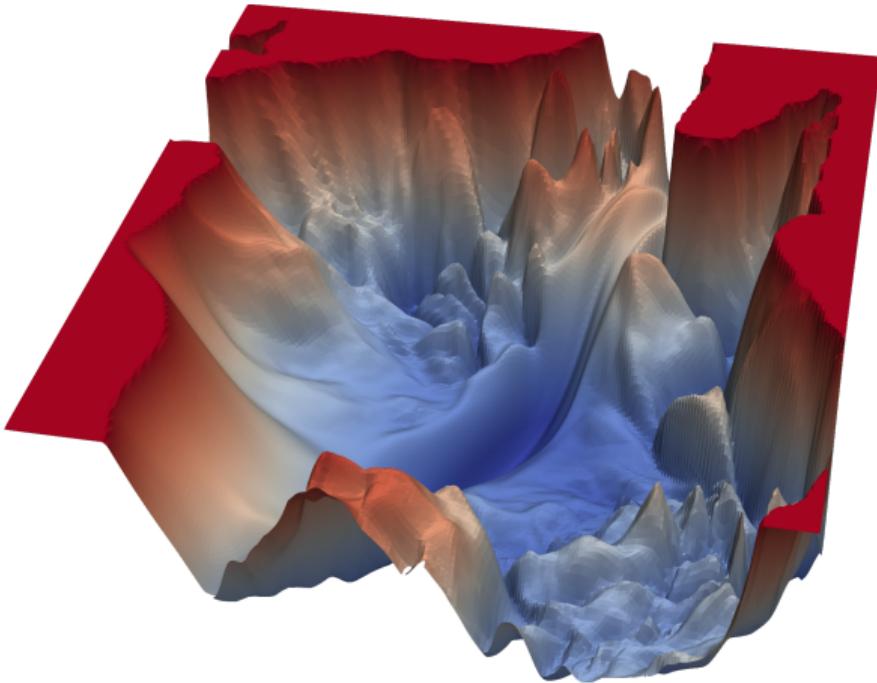
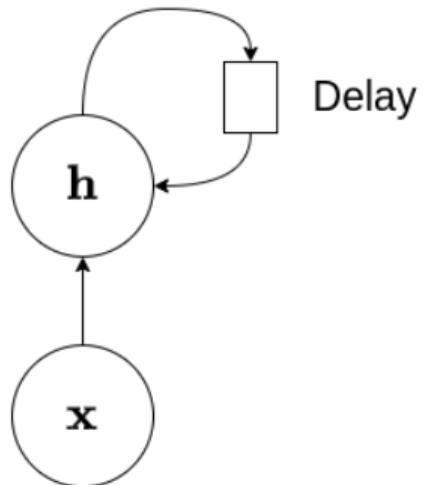


Abbildung: Plot einer Loss-Funktion (von <https://www.cs.umd.edu/tomg/projects/landscapes/>)

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
- 3. Rekurrente Neuronale Netzwerke (RNN)**
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Diagramm für ein RNN



Hier haben wir eine Sequenz von Vektoren, also

$$\begin{aligned}\mathbf{x} &= (\mathbf{x}_i)_{i=1}^{n \in N} \\ &= (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)\end{aligned}$$

Abbildung: Einfaches RNN in kompakter Darstellung

Definition RNN

Definition

Ein rekurrentes Netzwerk ist ein Netzwerk, welches

- ① als Graph dargestellt einen Kreis enthält
- ② beliebig viele Signalpunkt \mathbf{x}_t verarbeiten kann
- ③ die Zustände der *hidden Neurons* als rekursive Funktion dargestellt werden können.

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \Theta) \quad (5)$$

Hier steht Θ für unsere lernbaren Parameter.

RNN Ausgerollt

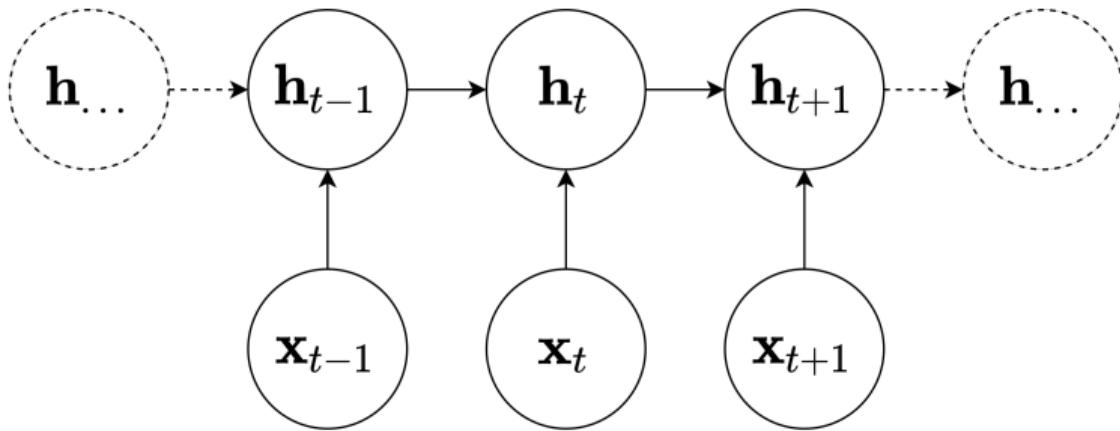


Abbildung: RNN wo jeder Signalpunkt ein eigener Eingabepunkt ist.

Immer noch kryptisch?

Zeit für ein Beispiel!

Beispiel

Sei \mathbf{x} ein Zeitreihensignal der Länge $t = 3$, also $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. Wir verwenden die Definition unserer hidden Neuronen.

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \quad (6)$$

Wir verzichten für das Beispiel auf Θ . Der letzte Knoten $t = 3$ lässt sich demnach wie folgt berechnen.

$$\mathbf{h}_3 = f(\mathbf{h}_2, \mathbf{x}_3) \quad (7)$$

Beispiel (Fortsetzung $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$)

Wenn wir unsere Funktion rekursiv zurückverfolgen für alle Eingabewerte, erhalten wir

$$\begin{aligned}\mathbf{h}_3 &= f(\mathbf{h}_2, \mathbf{x}_3) \\ &= f\left(f\left(\underbrace{f(\mathbf{h}_0, \mathbf{x}_1), \mathbf{x}_2}_{=\mathbf{h}_1}, \mathbf{x}_3\right)\right) \\ &\quad \underbrace{\qquad\qquad\qquad}_{=\mathbf{h}_2}\end{aligned}$$

Was ist dieses Θ ?

In Θ stecken unsere trainierbaren Parameter drin.

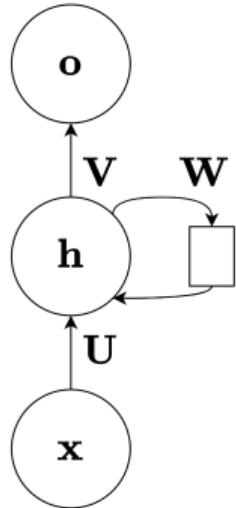


Abbildung: Erweiterung unseres Einführungskonzepts eines RNN

$$\mathbf{a}_t = \mathbf{b} + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t$$

$$\mathbf{h}_t = \tanh(\mathbf{a}_t)$$

$$\mathbf{o}_t = \mathbf{c} + \mathbf{V}\mathbf{h}_t$$

$$\hat{\mathbf{y}}_t = \text{softmax}(\mathbf{o}_t)$$

mit

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Wir brauchen noch eine Kostenfunktion

Mit der Kostenfunktion wollen wir den Fehler berechnen, den unser Modell in seiner aktuellen Konfiguration produziert. Dafür verwenden wir den *negative log-likelihood* bzw. die *cross-entropie*.

$$\begin{aligned} L\left((\mathbf{x}_i)_{i=0}^n, (\mathbf{y}_i)_{i=0}^n\right) &= \sum_t^n L_t \\ &= - \sum_t^n \log p(\mathbf{y}_t | (\mathbf{x}_i)_{i=0}^t) \end{aligned}$$

RNN mit Kostenfunktion

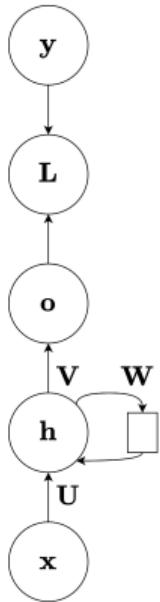


Abbildung: RNN mit Output und Kostenfunktion

Backpropagation...

über die Zeit hinweg

Da wir mit Signaldaten arbeiten, hat die Backpropagation auch einen ganz besonderen Namen. Hier handelt es sich um

Backpropagation Through Time (BPTT)

Der Gradient

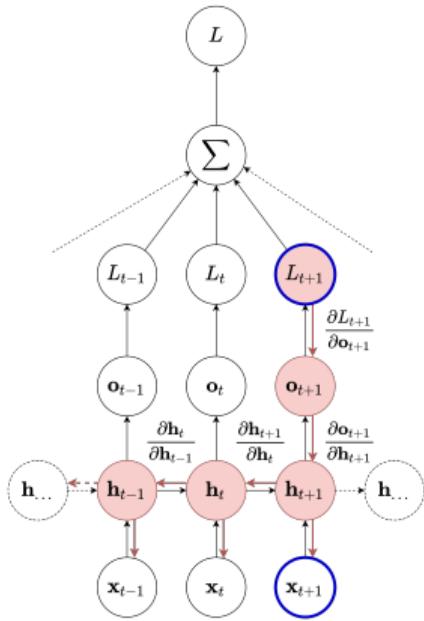
Wir wollen wieder den Gradienten für unsere trainierbaren Parameter berechnen.

$$\Delta \mathbf{W} = \frac{\partial L(t', t)}{\partial \mathbf{W}}$$

Hierbei sagen wir, dass wir den Loss zwischen Zeitpunkten t' und t mit $t' \leq t$ berechnen wollen. Daher berechnen wir

$$\frac{\partial L_t}{\partial \mathbf{W}} = \frac{\partial L_t}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}}$$

Backpropagation Through Time



$$\frac{\partial L_t}{\partial \mathbf{W}} = \frac{\partial L_t}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}}$$

Abbildung: Backpropagation im RNN

Achtung! Vanishing/Exploding Gradient

Der **Vanishing/Exploding Gradient** beschreibt, dass die Veränderung eines trainierbaren Parameters unter bestimmten Bedingungen extrem hoch (exploding) oder verschwindend klein (vanishing) sein kann.

RNNs leiden unter diesem Phänomen. Wir schauen uns an warum.

Vanishing/Exploding Gradient

Wir erweitern unsere Formel um die Idee, dass wir mehr als einen Zeitschritt zurückpropagieren können.

$$\frac{\partial L_t}{\partial \mathbf{W}} = \begin{cases} \frac{\partial L_t}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}} & \text{if } t - t' = 1 \\ \frac{\partial L_t}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \left(\prod_{k=t'+1}^t \frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}} \right) \frac{\partial \mathbf{h}_{t'}}{\partial \mathbf{W}_{t'}} & \text{if } t - t' > 1 \end{cases}$$

Vanishing/Exploding Gradient

Für unsere Untersuchung ist folgender Ausdruck von besonderem Interesse

$$\prod_{k=t'+1}^t \frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}}$$

Wir lösen daher zunächst den Ausdruck im Produkt.

$$\begin{aligned}\frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}} &= \frac{\partial \tanh(\mathbf{a}_k)}{\partial \mathbf{h}_{k-1}} && \text{(definition } \mathbf{h}_k\text{)} \\ &= \frac{\partial \tanh(\mathbf{b} + \mathbf{W}\mathbf{h}_{k-1} + \mathbf{Ux}_t)}{\partial \mathbf{h}_{k-1}} && \text{(definition } \mathbf{a}_k\text{)} \\ &= \mathbf{W} \cdot \frac{d}{d\mathbf{h}_{k-1}} \tanh(\mathbf{a}_k) && \text{(differentiation chain rule)}\end{aligned}$$

Vanishing/Exploding Gradient

Jetzt setzen wir unser gefundenen Ausdruck ein

$$\begin{aligned} \prod_{k=t'+1}^t \frac{\partial \mathbf{h}_k}{\partial \mathbf{h}_{k-1}} &= \prod_{k=t'+1}^t \mathbf{W} \cdot \frac{d}{d \mathbf{h}_{k-1}} \tanh(\mathbf{a}_k) \\ &= \mathbf{W}^{t-t'} \prod_{k=t'+1}^t \frac{d}{d \mathbf{h}_{k-1}} \tanh(\mathbf{a}_k) \end{aligned}$$

Es gibt verschiedene RNN

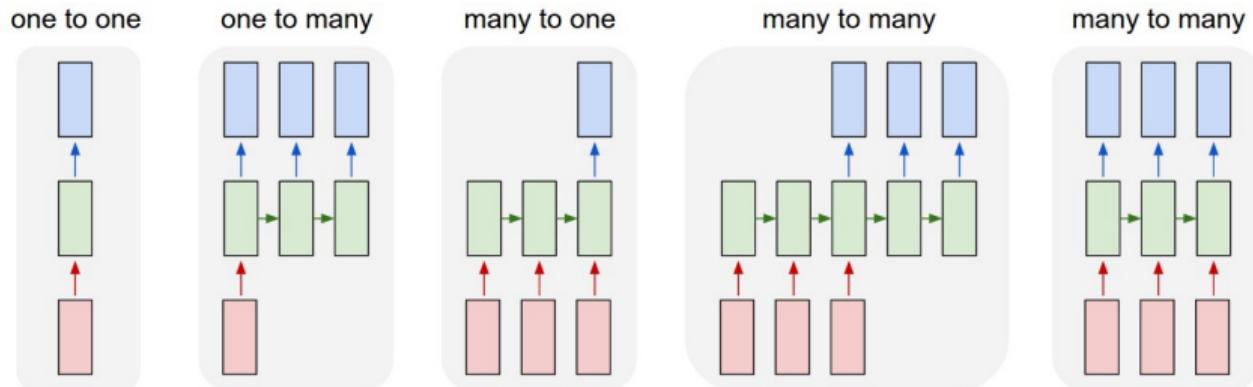


Abbildung: RNN Typen (von <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>)

Fazit zu RNN

- ① Zeit- und Raumkomplexität $\mathcal{O}(t)$ für sowohl Training als auch Backpropagation
- ② Vanishing/Exploding Gradient beeinträchtigt Stabilität der Netze für Daten ab ca. 8 Signalzeitpunkten

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
- 4. Long Short Term Memory Netzwerke (LSTM)**
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Einführung LSTM

Wir wollen den Vanishing/Exploding Gradient erheblich reduzieren!

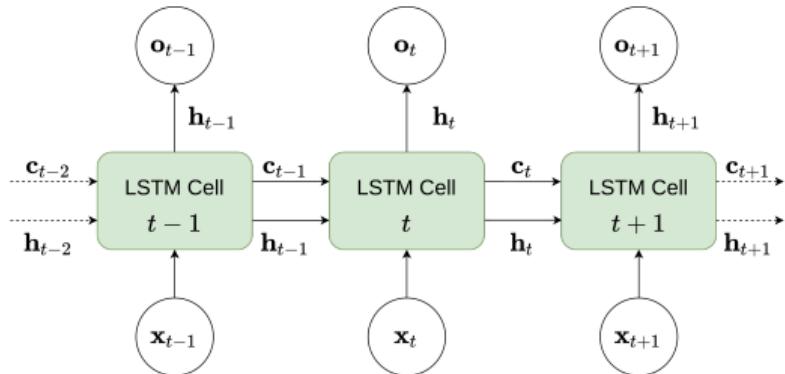


Abbildung: LSTM Netzwerk ausgerollt

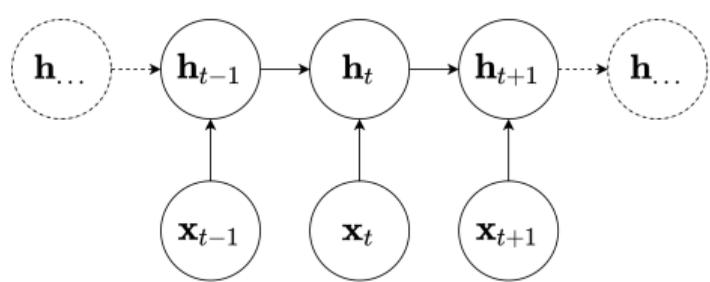


Abbildung: RNN ausgerollt

Die LSTM-Zelle

Gates sind hier der entscheidende Unterschied

→ f_t Forget Gate

→ i_t Input Gate

→ u_t Output Gate

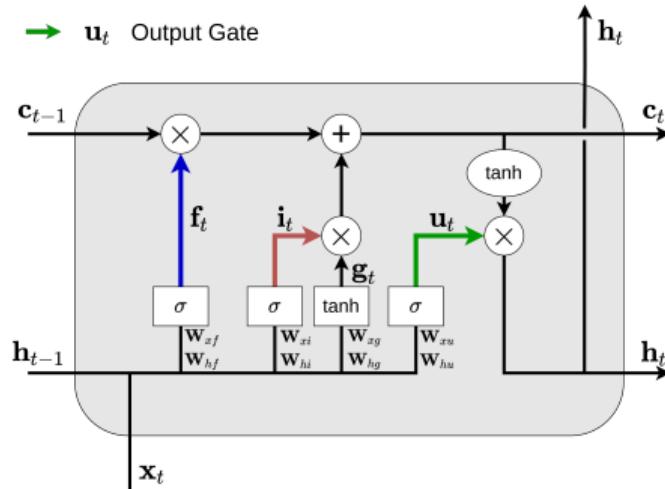


Abbildung: Ein vollständiger Blick in eine LSTM Zelle

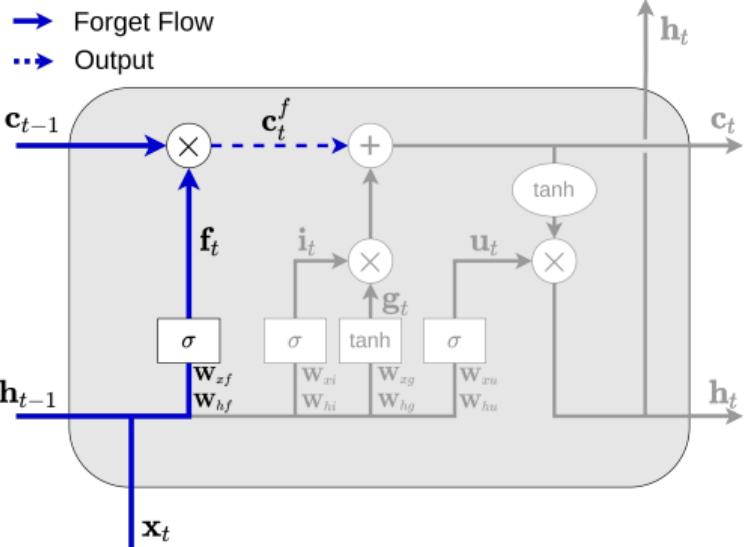
Wie funktioniert ein Gate?

Ein Gate $\mathbf{v} \in \mathbb{R}^n$ ist ein Vektor dessen Werte v_i zwischen 0 und 1 liegen. Also $v_i \in [0, 1]$.

An den Punkten wo das Symbol \otimes zu sehen ist, wird eine punktweise Multiplikation \odot zwischen dem Gate und dem einkommenden Signal durchgeführt. Sei das einkommende Signal der Zellzustand $\mathbf{c} \in \mathbb{R}^n$, dann haben wir am Gate \mathbf{v}

$$\mathbf{c} \odot \mathbf{v} = \begin{pmatrix} c_1 \cdot v_1 \\ c_2 \cdot v_2 \\ \dots \\ c_n \cdot v_n \end{pmatrix}$$

Das Forget-Gate

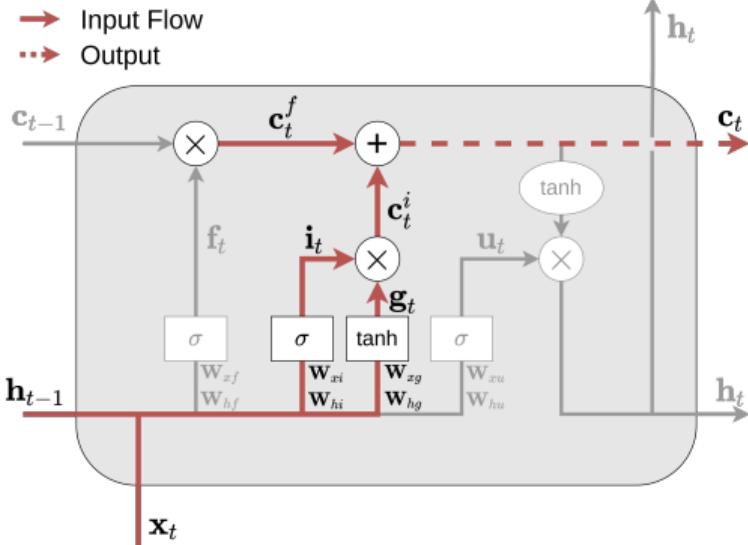


Definition

Das Forget-Gate f_t bestimmt wie viel des Gedächtniszustands der vorherigen Zelle $t - 1$ in die neue Zelle übernommen werden soll. Es wird berechnet

$$f_t = \sigma(\mathbf{W}_{xf} \mathbf{x}_t + \mathbf{W}_{hf} \mathbf{h}_{t-1} + \mathbf{b}_f)$$

Das Input-Gate



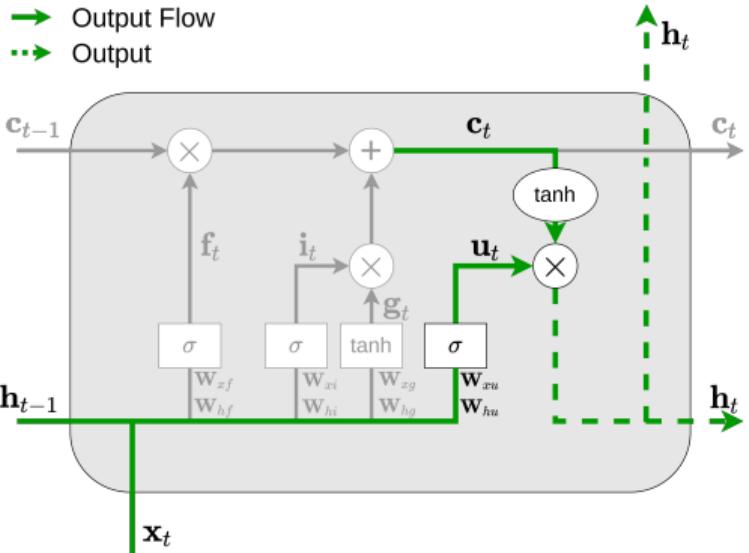
Definition

Das Input-Gate i_t bestimmt wie viel der Verarbeitung g_t des neuen inputs x_t in den neuen Zellzustand einfließen soll. Es wird berechnet

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

Das Output-Gate

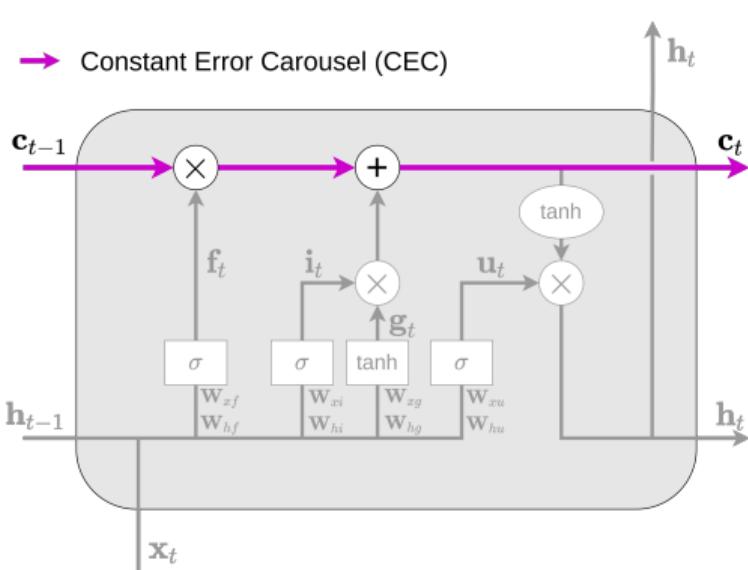


Definition

Das Output-Gate bestimmt wie viel der Aktivierung des Zellzustands $\tanh(\mathbf{c}_t)$ in den Ausgabezustand \mathbf{h}_t fließen soll. Es wird berechnet

$$\mathbf{u}_t = \sigma \left(\mathbf{W}_{xo} \mathbf{x}_t + \mathbf{W}_{ho} \mathbf{h}_{t-1} + \mathbf{b}_o \right)$$

Das Constant-Error-Carousel (CEC)



Definition

Das Constant-Error-Carousel ist die Verbindung zwischen zwei hidden Neurons über eine lineare Einheit, sodass ein konstanter Error-Fluss garantiert werden kann.

Beschränkte Backpropagation

Ein weiteres Konzept ist die BPTT in der Spanne der Zeit zu beschränken. Dies wird auch verwendet um in gewöhnlichen RNN dem Vanishing/Exploding Gradient entgegen zu wirken.

Definition (Fehler Beschränkte BPTT)

Sei $\hat{t} > 0$ eine Konstante an Zeitschritten. Wir nennen BPTT Fehler-beschränkt, wenn wir die Backpropagation nach \hat{t} Zeitschritten stoppen.

Wie trainiert man sowas?

Wir berechnen wieder den Loss und führen einen Gradientenabstieg durch

$$\frac{\partial L_t}{\partial \mathbf{W}} = \frac{\partial L_t}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{c}_t}{\partial \mathbf{h}_t} \left(\prod_{k=t'+1}^t \frac{\partial \mathbf{c}_k}{\partial \mathbf{c}_{k-1}} \right) \frac{\partial \mathbf{c}_{t'}}{\partial \mathbf{W}_{t'}} \quad (8)$$

$$\frac{\partial L_t}{\partial \mathbf{W}} = \frac{\partial L_t}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_t} \left(\prod_{k=t'+1}^t \frac{\partial \mathbf{c}_k}{\partial \mathbf{c}_{k-1}} \right) \frac{\partial \mathbf{c}_{t'}}{\partial \mathbf{W}_{t'}} \quad (9)$$

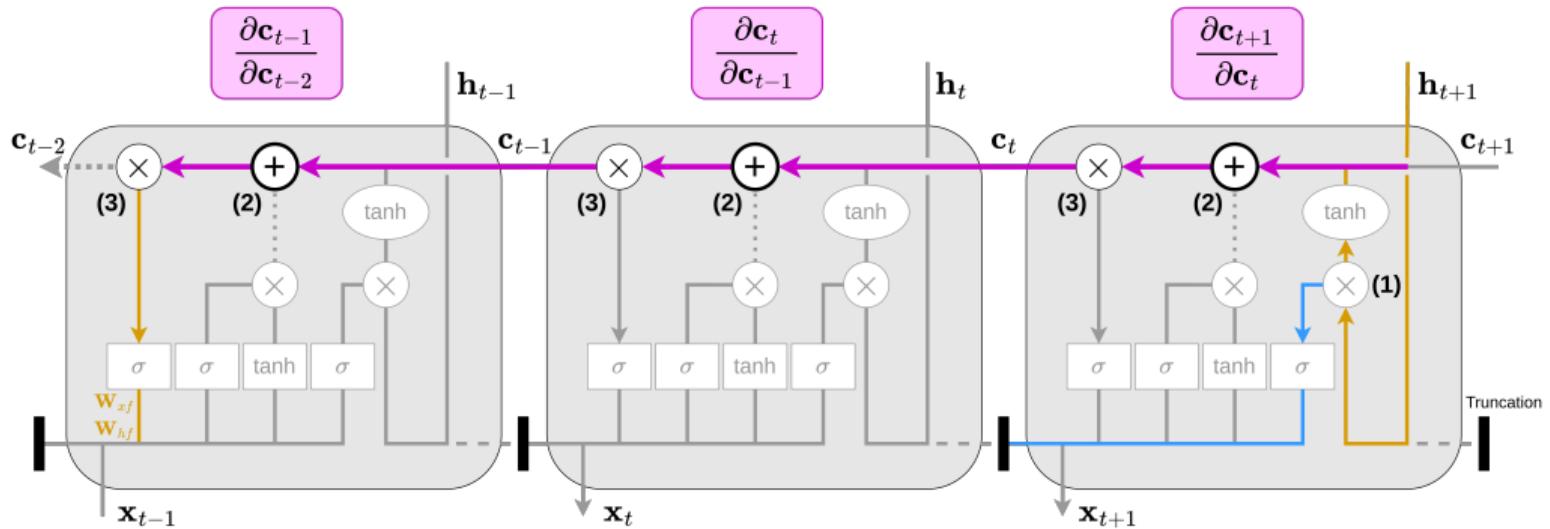


Abbildung: Partielle Ableitung nach den Forget Matrizen \mathbf{W}_f

Aber...

wie löst das unseren Vanishing/Exploding Gradient?

Im RNN lag das Problem im Produkt $\prod_{k=t'+1}^t \frac{\partial \mathbf{c}_k}{\partial \mathbf{c}_{k-1}}$ begraben, was wir jetzt genauer ansehen

$$\begin{aligned}\frac{\partial \mathbf{c}_{t+1}}{\partial \mathbf{c}_t} &= \frac{\partial \mathbf{c}_t^f}{\partial \mathbf{c}_t} + \frac{\partial \mathbf{c}_t^i}{\partial \mathbf{c}_t} && \text{(definition } \mathbf{c}_t\text{)} \\ &= \frac{\partial \mathbf{c}_t \odot \mathbf{f}_{t+1}}{\partial \mathbf{c}_t} + \frac{\partial \mathbf{i}_{t+1} \odot \mathbf{g}_{t+1}}{\partial \mathbf{c}_t} && \text{(definition } \mathbf{c}_t^f \& \mathbf{f}_{t+1}\text{)} \\ &= \underbrace{\frac{\partial \mathbf{c}_t \odot \mathbf{f}_{t+1}}{\partial \mathbf{c}_t}}_{\text{input path}} + \underbrace{\frac{\partial \mathbf{i}_{t+1} \odot \mathbf{g}_{t+1}}{\partial \mathbf{c}_t}}_{\text{Differentiation Summenregel}}\end{aligned}$$

Fortsetzung Analyse Gradient

$$\begin{aligned} &= \mathbf{c}_t \cdot \underbrace{\frac{\partial \mathbf{f}_{t+1}}{\partial \mathbf{c}_t}}_{=0} + \mathbf{f}_{t+1} \cdot \underbrace{\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_t}}_{=1} && \text{(differentiation product rule)} \\ &= \mathbf{f}_{t+1} \end{aligned}$$

Also wissen wir

$$\prod_{k=t'+1}^t \frac{\partial \mathbf{c}_k}{\partial \mathbf{c}_{k-1}} = \prod_{k=t'+1}^t \mathbf{f}_k$$

Somit verschwindet die sich potenzierende Matrix \mathbf{W} .

Fazit LSTM

- ① LSTM sind deutlich robuster gegenüber dem Vanishing/Exploding Gradient
- ② Die Zeit- und Raumkomplexität von Forward- und Backwardpass ist immernoch linear
- ③ Momentan produzieren wir für jeden Eingabe-Punkt einen Ausgabe-Punkt. Wir benötigen eine Lösung, dass Variabel lange Ausgaben entstehen können

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Encoder-Decoder Netzwerke

Eigenschaften von Encoder-Decoder Netzwerken:

- die Ausgabesequenz $(y_i)_{i=1}^{l \in \mathbb{N}}$ kann unterschiedlich lang zur Eingabesequenz $(x_i)_{i=1}^{n \in \mathbb{N}}$ sein
- es wird ein Kontext-Vektor $(z_i)_{i=1}^{m \in \mathbb{N}}$ erzeugt
- es besteht aus LSTM Netzwerken, in seiner Urfassung besteht der Encoder und der Decoder jeweils aus einem eigenen LSTM Netzwerk

Encoder-Decoder Netzwerke

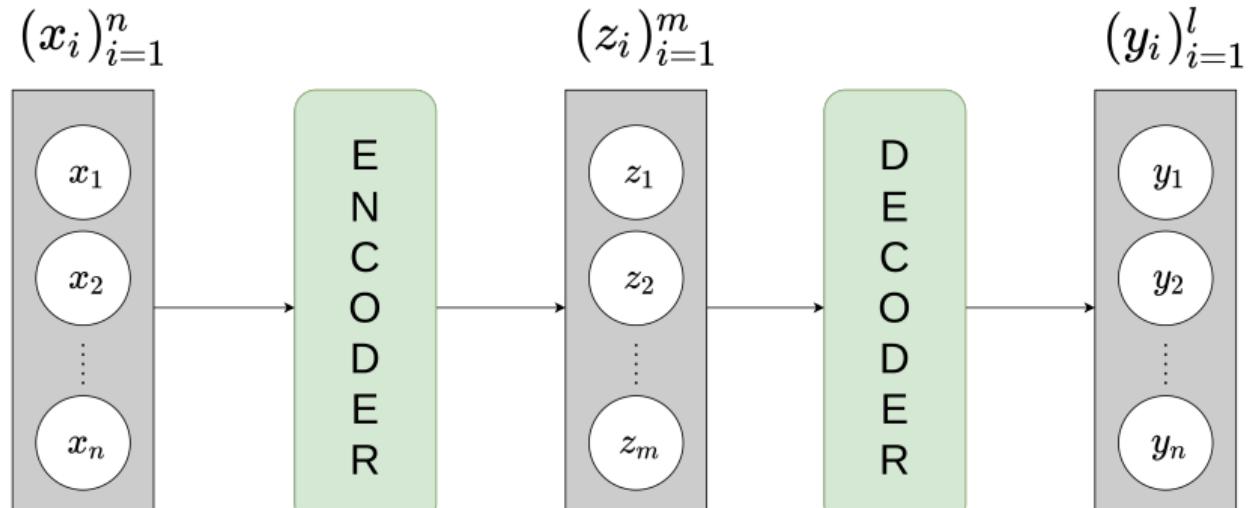


Abbildung: Prinzipielle Idee

Encoder-Decoder Netzwerke

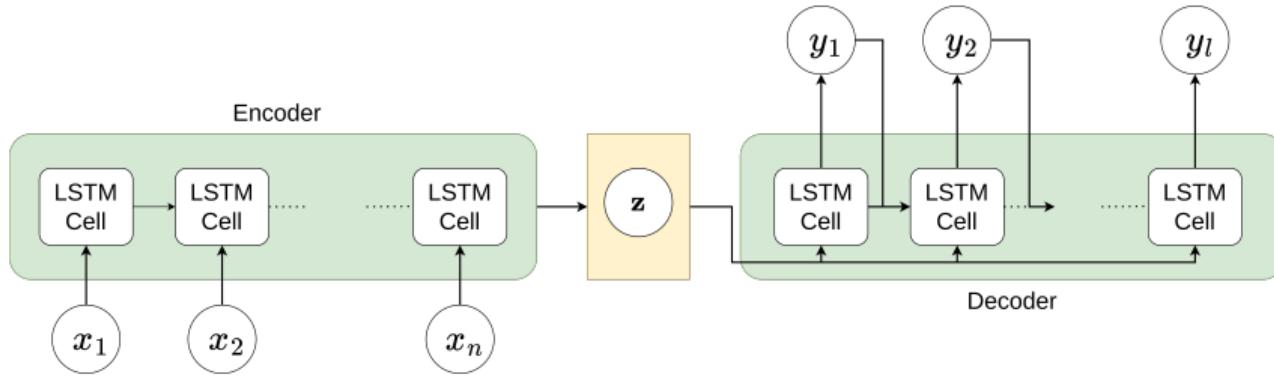


Abbildung: Encoder-Decoder mit mehr Detail

$$\mathbf{h}_t = f_1(x_t, \mathbf{h}_{t-1})$$

$$\mathbf{z} = \mathbf{h}$$

$$\mathbf{d}_{t'} = f_2(\mathbf{d}_{t'-1}, y_{t'-1}, \mathbf{z})$$

$$y_{t'} = g(\mathbf{d}_{t'}, y_{t'-1}, \mathbf{z})$$

Fazit Encoder-Decoder Netzwerke

- ① Komplexität vergleichbar mit klassischen RNN
- ② Kontext-Vektor z muss den Inhalt der gesamten Eingabe erfassen. Vor allem der Zusammenhang eines Zeitpunktes zu allen anderen Zeitpunkten. Begrenzte Größe des Kontextvektors erschwert die Erfassung der Informationen.

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Attention Einführung

Die Idee von Attention ist den Zusammenhang eines Zeitpunktes mit anderen Zeitpunkten zu erfassen.

- jedem Signalpunkt x_i wird ein Gewicht a_i ermittelt
- Attention ist in der Lage zu visualisieren, welche Teile für die Vorhersage wichtig waren
- Attention erlaubt ein Stück weit Interpretierbarkeit von den subsymbolischen neuronalen Netzen

Beispiel von Attention in Bildern



Abbildung: Klassifikation von Bildern

Beispiel von Attention in Texten

Task: Hotel location

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this can't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this can't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this can't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Abbildung: Sentiment-Analyse von Hotel Reviews

Lasst uns das mal formalisieren

Definition (Einfache Attention)

Attention ist eine Funktion, die einen **Query q** nimmt und zu einer **Key Matrix K** ein Gewicht-Vektor α ausgibt. Dabei gibt α_i das Gewicht von Query q zu einem der Keys $k_i \in K$ an. Wir definieren Attention wie folgt

$$\alpha = \text{attention}(q, K) = g(f_e(q, K))$$

wo f_e die Energie-Funktion ist.

$$e = f_e(q, K)$$

Energie Funktionen im Überblick

Es gibt eine Vielzahl möglicher Energiefunktionen. Hier ein paar ausgewählte Beispiele.

$$f_e(\mathbf{q}, \mathbf{K}) = \text{similarity}(\mathbf{q}, \mathbf{K}) \quad (\text{similarity})$$

$$f_e(\mathbf{q}, \mathbf{K}) = \mathbf{q}^T \mathbf{K} \quad (\text{multiplicative or dot})$$

$$f_e(\mathbf{q}, \mathbf{K}) = \frac{\mathbf{q}^T \mathbf{K}}{\sqrt{n_k}} \quad (\text{scaled multiplicative})$$

wobei n_k für die Größe des Vektor $\mathbf{k} \in \mathbf{K}^{n_k \times d_k}$ ist.

Vielleicht hilft ein Diagramm...

es besser zu verstehen.

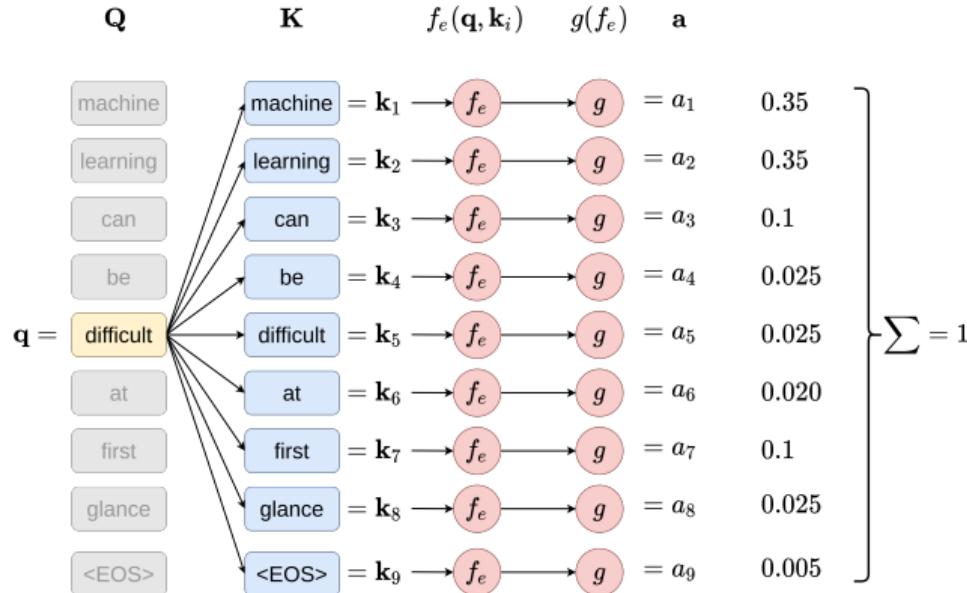


Abbildung: Attention am Beispiel

Visualisierung des Signalflusses

Wo kommen K und q eigentlich her?

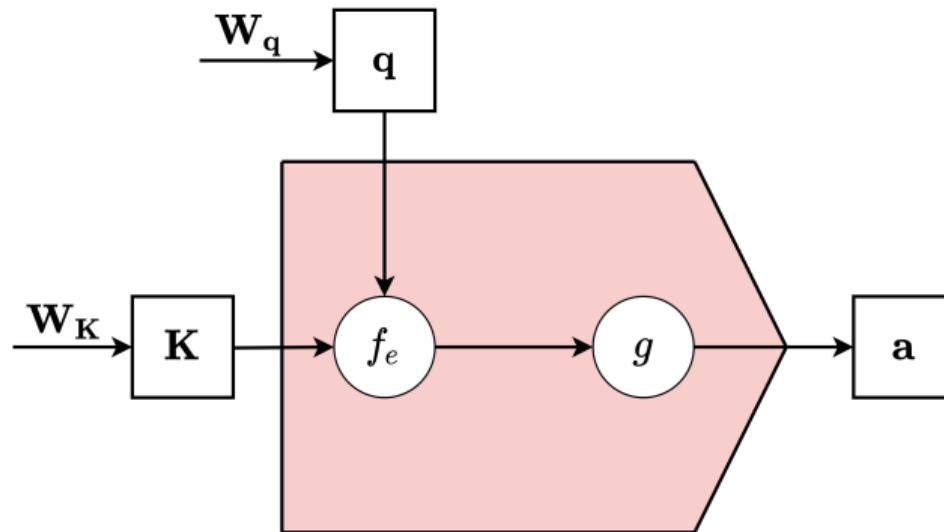


Abbildung: Flussdiagramm für einfache Attention

Wenn es einfache Attention gibt...

Definition (Erweiterte Attention)

Erweiterte Attention nimmt als zusätzlichen Parameter die Value Matrix $V^{n_k \times d_k}$

$$\mathbf{z} = \text{attention}(\mathbf{q}, \mathbf{K}, \mathbf{V})$$

wobei wir die Verarbeitung wie folgt definieren

$$\mathbf{z} = \sum_{\mathbf{r}_i \in \mathbf{R}} \mathbf{r}_i$$

$$\mathbf{r}_i = \alpha_i \mathbf{v}_i$$

$$\mathbf{R}[:, i] = \mathbf{r}_i$$

Diagramm zur erweiterten Attention

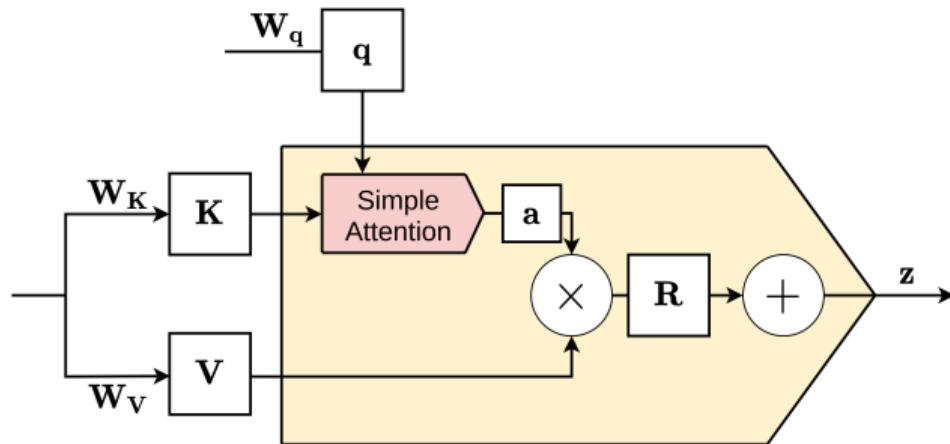


Abbildung: Flussdiagramm für erweitertes Attention

Gliederung - Zwischenstand

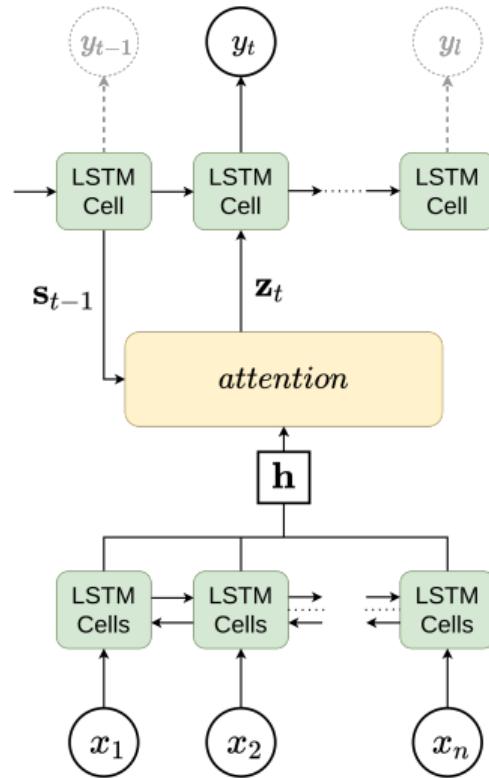
1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Was machen wir nun mit Attention?

Attention kann die Probleme von der reinen Encoder-Decoder Architektur kontern

- Problem war, der Kontext Vektor z aus Encoder-Decoder muss den vollständigen Zusammenhang zwischen allen Signalpunkten lernen
- Attention verbessert die Performance den Zusammenhang zwischen Signalpunkten zu lernen

Lasst uns Attention reinbauen

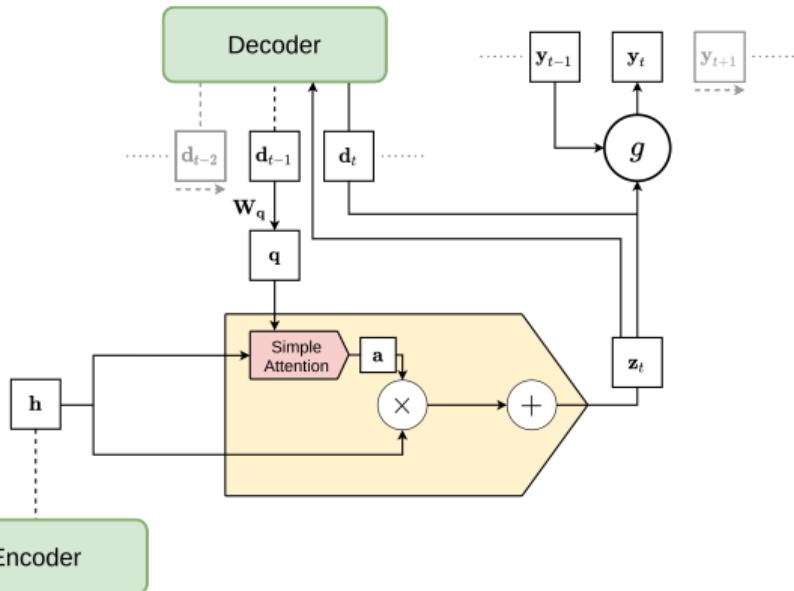


Was verändert sich...

gegenüber dem traditionellen Encoder-Decoder

Component	Traditional Encoder-Decoder	Encoder-Decoder with Attention
Encode	$\mathbf{h}_i = f(x_i, \mathbf{h}_{i-1})$	$\mathbf{h}_i = f(x_i, \mathbf{h}_{i-1})$
Context	$\mathbf{z} = \mathbf{h}$	$\mathbf{z}_t = \sum_{i=1}^{d_h} a_{it} \mathbf{h}_i$ $a_{it} = g(e_{it})$ $e_{it} = f_e(d_{t-1}, \mathbf{h}_i)$
Decode	$\mathbf{d}_t = f_2(\mathbf{d}_{t-1}, y_{t-1}, \mathbf{z})$	$\mathbf{d}_t = f(\mathbf{d}_{t-1}, y_{t-1}, \mathbf{z}_t)$
Generate	$y_t = g(\mathbf{d}_t, y_{t-1}, \mathbf{z})$	$y_t = g(\mathbf{d}_t, y_{t-1}, \mathbf{z}_t)$

Schauen wir uns das mal genau an



Component	ED with Attention
Encode	$\mathbf{h}_i = f(x_i, \mathbf{h}_{i-1})$
Context	$\mathbf{z}_t = \sum_{i=1}^{d_h} a_{it} \mathbf{h}_i$
	$a_{it} = g(e_{it})$
	$e_{it} = f_e(d_{t-1}, \mathbf{h}_i)$
Decode	$\mathbf{d}_t = f(\mathbf{d}_{t-1}, y_{t-1}, \mathbf{z}_t)$
Generate	$y_t = g(\mathbf{d}_t, y_{t-1}, \mathbf{z}_t)$

Fazit Encoder-Decoder mit Attention

- Attention löst es den Zusammenhang zwischen allen Zeitpunkten mitzulernen
- Attention kann uns visualisieren, wo der Fokus/das Gewicht bei der Arbeit des Modells liegt
- Nach wie vor haben wir rekursive Strukturen, weshalb wir immernoch linearen Aufwand in Speicheraufwand und Rechenzeit haben

Gliederung - Zwischenstand

1. Motivation - Transformer Netzwerke
2. Grundlagen in Neuronale Netzwerke
3. Rekurrente Neuronale Netzwerke (RNN)
4. Long Short Term Memory Netzwerke (LSTM)
5. Encoder-Decoder Netzwerke
6. Attention in Neuronalen Netzwerken
7. Encoder-Decoder mit Attention
8. Transformer Netzwerke

Zurück zum Transformer!

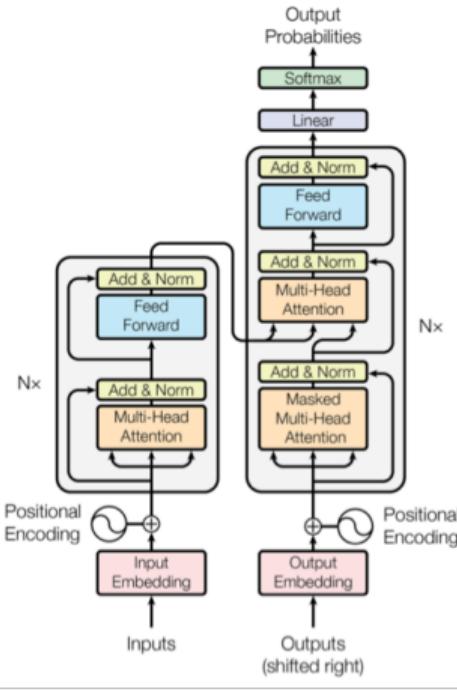


Abbildung: Transformer Diagramm (von Attention is all you need paper)

Bibliographie I

-  I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*.
Adaptive Computation and Machine Learning series, MIT Press, 2016.
-  S. S. Haykin, *Neural networks and learning machines*.
Upper Saddle River, NJ: Pearson Education, third ed., 2009.
-  N. Mohajerin and S. L. Waslander, "State initialization for recurrent neural network modeling of time-series data," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2330–2337, 2017.
-  A. Graves, *Long Short-Term Memory*, pp. 37–45.
Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

Bibliographie II

-  R. C. Staudemeyer and E. R. Morris, "Understanding LSTM - a tutorial into long short-term memory recurrent neural networks," *CoRR*, vol. abs/1909.09586, 2019.
-  Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.
-  A. Kag and V. Saligrama, "Training recurrent neural networks via forward propagation through time," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 5189–5200, PMLR, 18–24 Jul 2021.

Bibliographie III

-  A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," 2015.
-  S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.
-  F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," vol. 12, pp. 2451–71, 10 2000.
-  H. Tang and J. R. Glass, "On training recurrent networks with truncated backpropagation through time in speech recognition," *CoRR*, vol. abs/1807.03396, 2018.

Bibliographie IV

-  T. Parr and J. Howard, "The matrix calculus you need for deep learning," *CoRR*, vol. abs/1802.01528, 2018.
-  J. Bayer, P. van der Smagt, and J. Schmidhuber, *Learning Sequence Representations*.
Universitätsbibliothek der TU München, 2015.
-  A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.
-  R. Dale, "Gpt-3: What's it good for?," *Natural Language Engineering*, vol. 27, no. 1, p. 113–118, 2021.

Bibliographie V

-  A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022.
-  K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.
-  M. Henaff, A. Szlam, and Y. LeCun, "Orthogonal rnns and long-memory tasks," *CoRR*, vol. abs/1602.06662, 2016.
-  I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.

Bibliographie VI

-  S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, "An attentive survey of attention models," 2019.
-  I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014.
-  A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4291–4308, oct 2021.
-  H. Guo, X. Fan, and S. Wang, "Human attribute recognition by refining attention heat map," *Pattern Recognition Letters*, vol. 94, pp. 38–45, 2017.

Bibliographie VII

-  D. Britz, A. Goldie, M.-T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," 2017.
-  M. Phuong and M. Hutter, "Formal algorithms for transformers," 2022.
-  D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 1 ed., 2000.