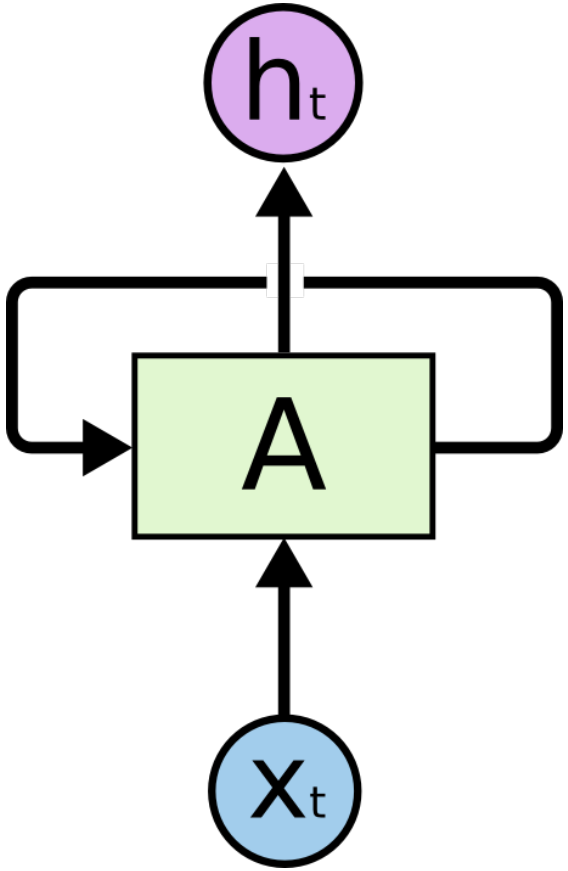
Recurrent Neural Networks

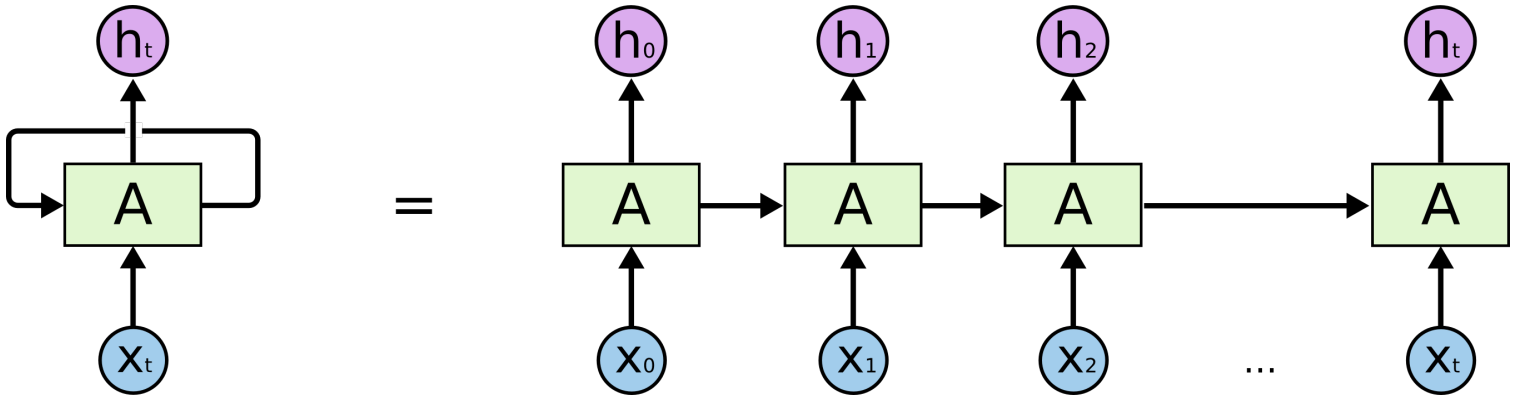
Einfache RNNs



- RNNs haben einen Feedback loop, welche den Output zurück ins Netzwerk führen
- Input: $X_t,\ h_{t-1}$
- Output: h_t
- t wird als Zeitschritt bezeichnet, aber muss nicht unbedingt mit physikalischer Zeit übereinstimmen

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

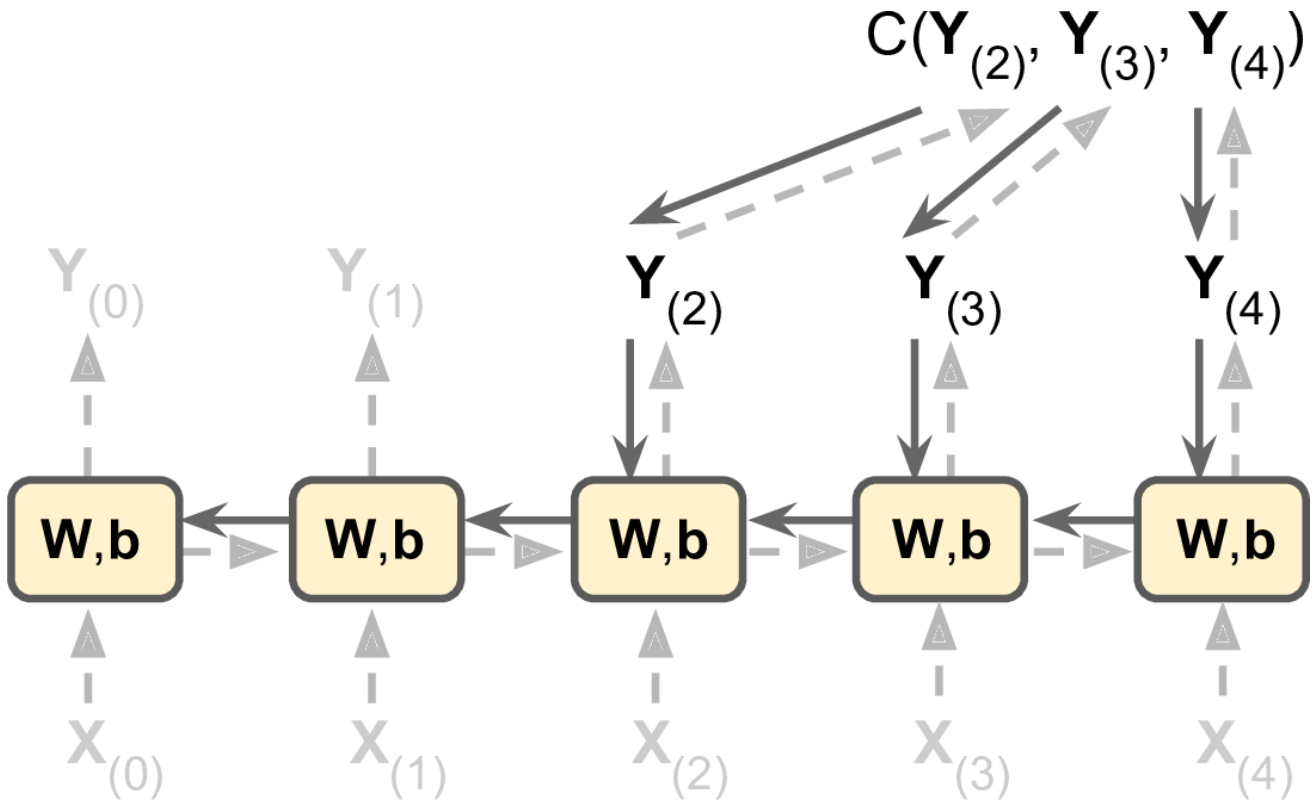
RNNs in der Zeit aufrollen



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

- RNNs können in der Zeit aufgerollt werden
- Das aufgerollte Netzwerk ist ähnlich zu einem tiefen feedforward Netzwerk, insbesondere ändert sich Backpropagation nicht wesentlich (BPTT)
- Zum ersten Zeitschritt existiert noch kein hidden state -> Typischerweise werden hier Nullen verwendet (Hinweis auf Stateless vs. Stateful)

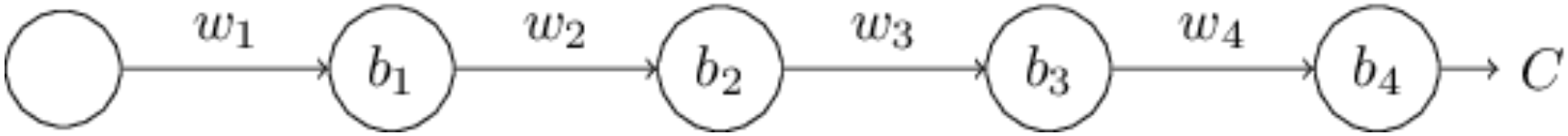
RNNs in der Zeit aufrollen



Source: Aurelion Geron, Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow, 2nd ed.

- Backpropagation through time (BPTT)
- Aufgerollte RNNs können mit normaler Backpropagation trainiert werden
- Einige Outputs können von der Loss function ignoriert werden
- Vanishing/Exploding Gradient Problem:
  - Ebenso wie tiefe feedforward Netze haben auch RNNs Probleme mit instabilen Gradienten

Vanishing / Exploding Gradient Problem



- Betrachte ein 4 Layer feedforward Netz mit einem Neuron pro Layer
- Output des Netzwerks: a_4
- Output der j-ten Layer: $a_j = \sigma(z_j), \text{ mit } z_j = w_j a_{j-1} + b_j$
- Backpropagation gibt für den Gradienten in der ersten Layer:

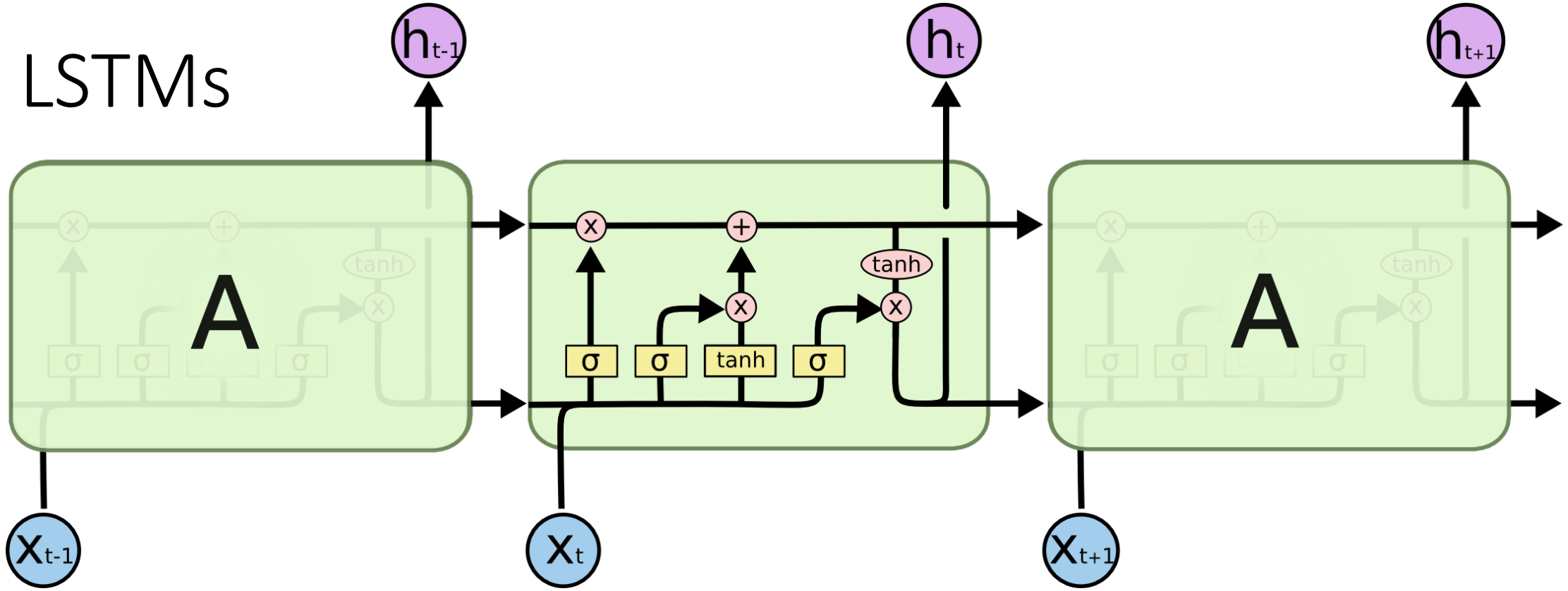
$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \cdot w_2 \cdot \sigma'(z_2) \cdot w_3 \cdot \sigma'(z_3) \cdot w_4 \cdot \sigma'(z_4) \cdot \frac{\partial C}{\partial a_4}$$

- Produkt mit vielen Faktoren -> instabiler Gradient, denn
  - Exponentieller Anstieg was Faktoren > 1
  - Exponentieller Abfall für Faktoren < 1
- Für RNNs ist $w = w_1 = w_2 = w_3 = w_4$

Tips und Tricks gegen instabile Gradienten

- Gute Initialisierung von Gewichten
- Dropout (in Keras *dropout* und *recurrent_dropout*)
- Batch Normalization
- Layer Normalization
- Gradient clipping

LSTMs

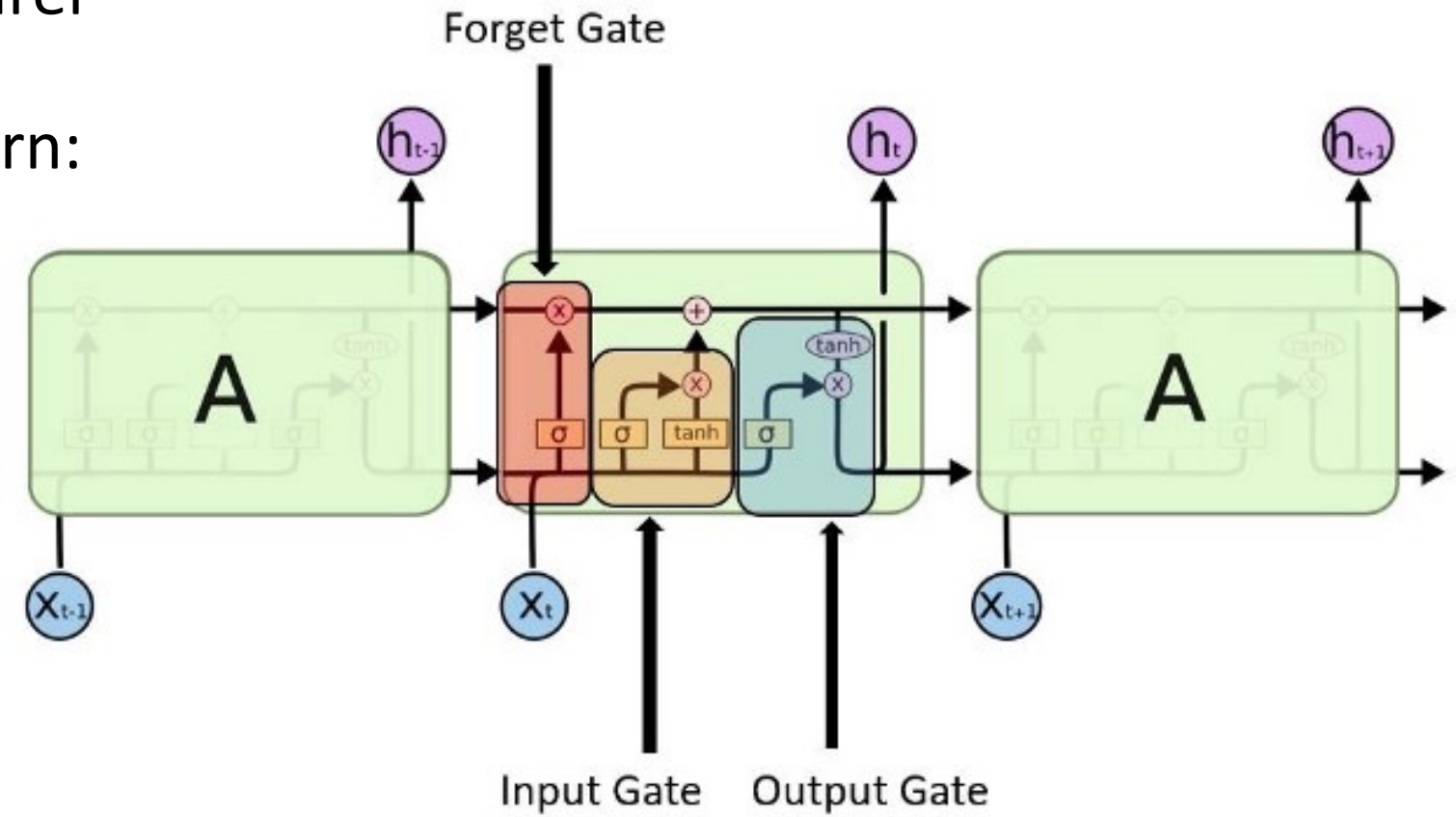


Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

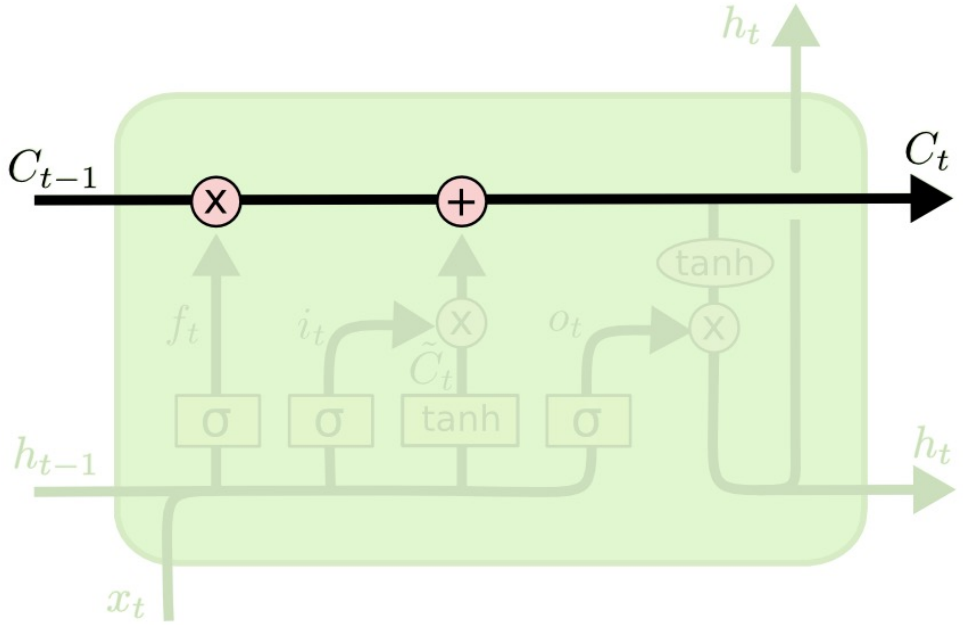
- LSTM (Long Short-Term Memory) (Hochreiter & Schmidhuber, 1997)
- Besitzen einen cell state zum Speichern von Informationen, um das Short-Term Memory Problem in den Griff zu kriegen

Cell State

- Die LSTM Zelle besitzt drei Gates welche den Informationsfluss steuern:
  - Forget Gate
  - Input Gate
  - Output Gate

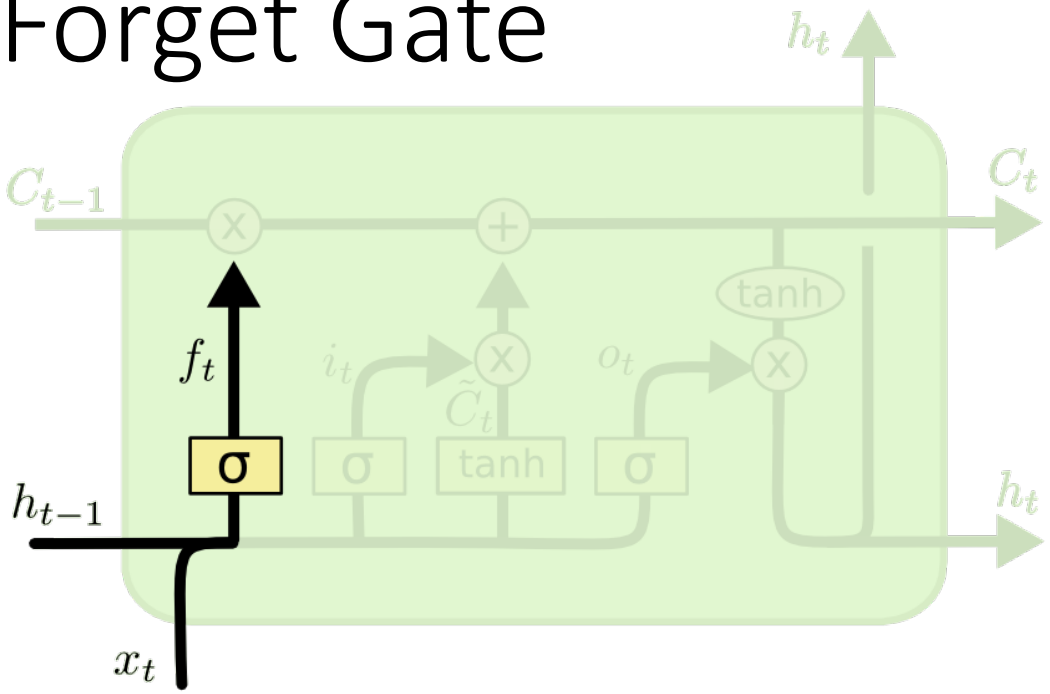


Cell State



- Cell state durchläuft die LSTM Zelle und dient als "Gedächtnis"
- Zwei Operationen:
  - Forget Gate: Multiplikation mit [0, 1] (steuert welche Informationen vergessen werden sollen)
  - Input Gate: Addition (steuert welche Informationen hinzugefügt werden sollen)

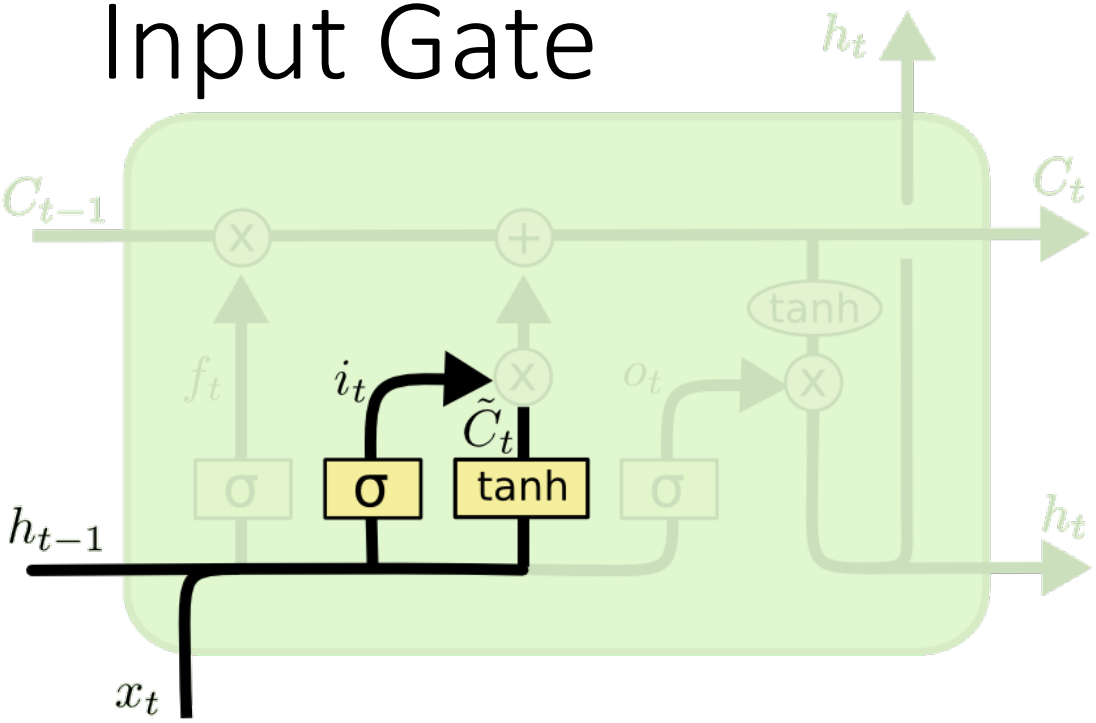
Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- Input: x_t, h_{t-1}
- Sigmoid bildet auf Intervall [0, 1] ab
- 0 = vergiss alles
- 1 = erhalte alles

Input Gate

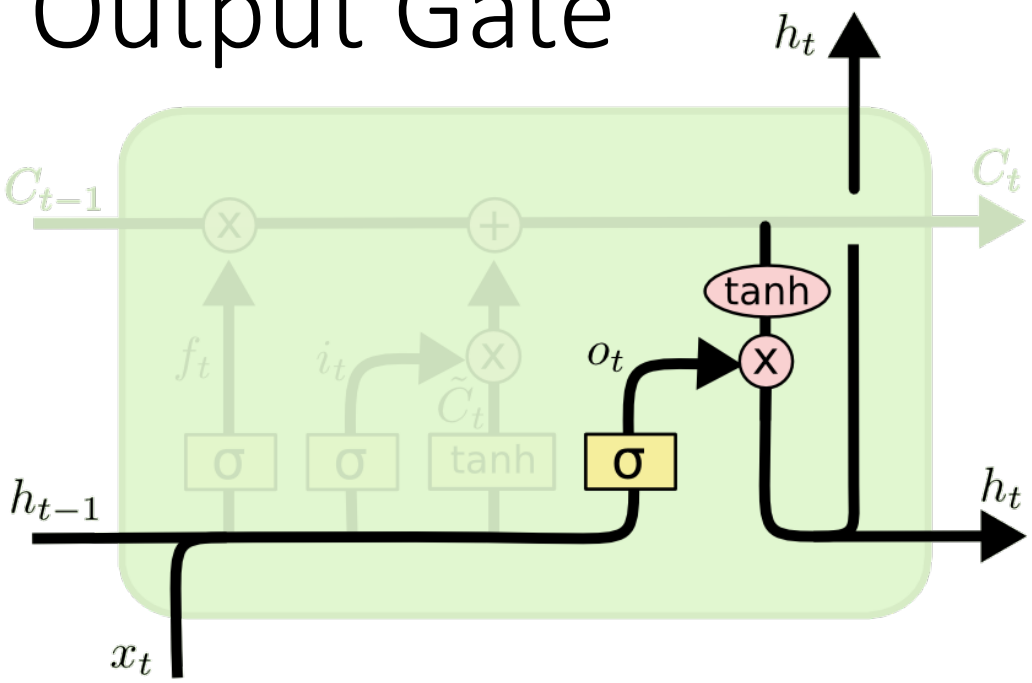


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- Input: x_t, h_{t-1}
- i_t Steuert welche Teile des cell state ein update kriegen
- Addition auf den cell state steuert welche neuen Informationen hinzugefügt werden sollen

Output Gate



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

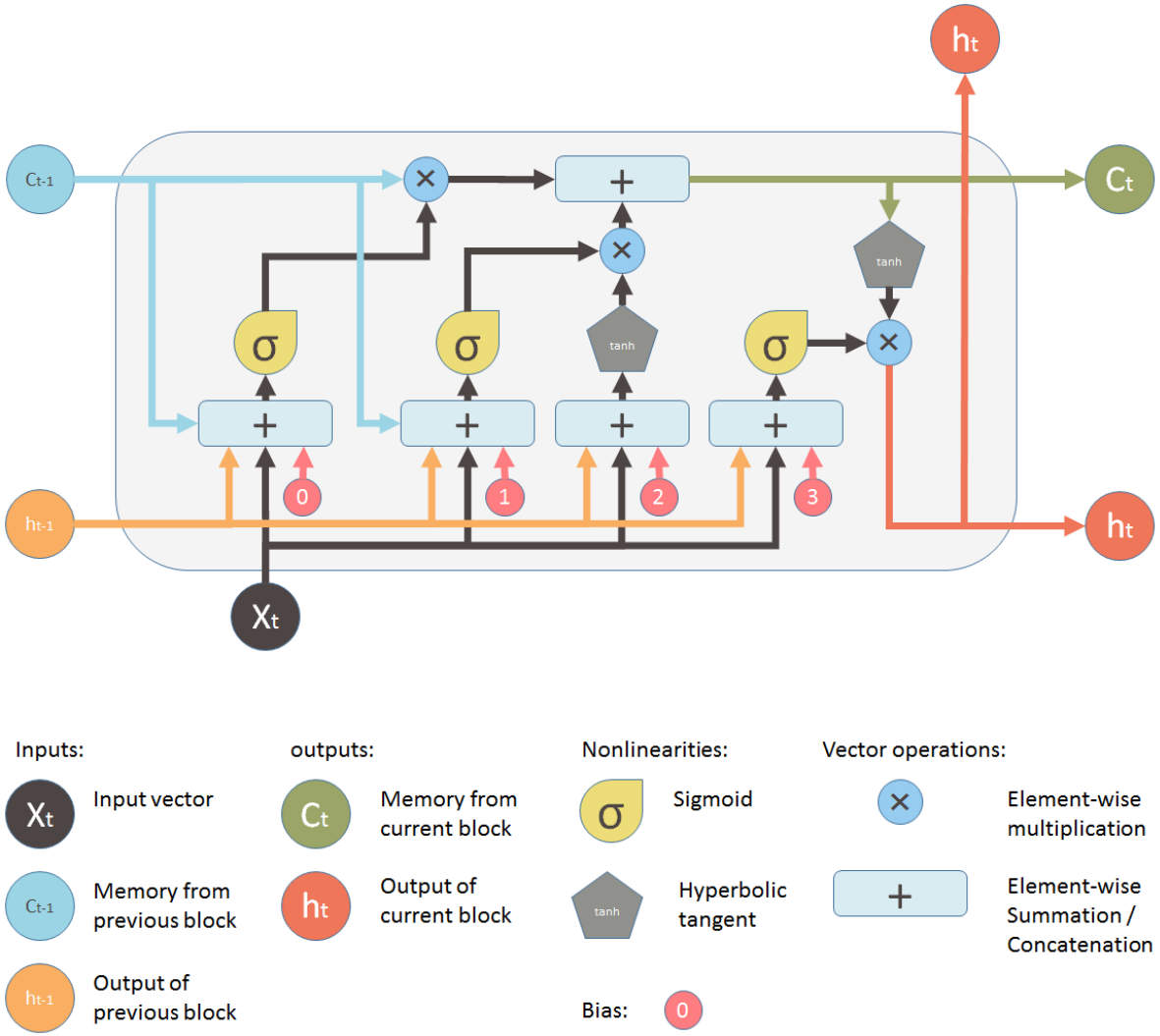
$$h_t = o_t * \tanh\left(C_t\right)$$

- Input: C_t, x_t, h_{t-1}
- O_t steuert welche Teile des cell states in den Output kommen

Anwendungen von RNNs

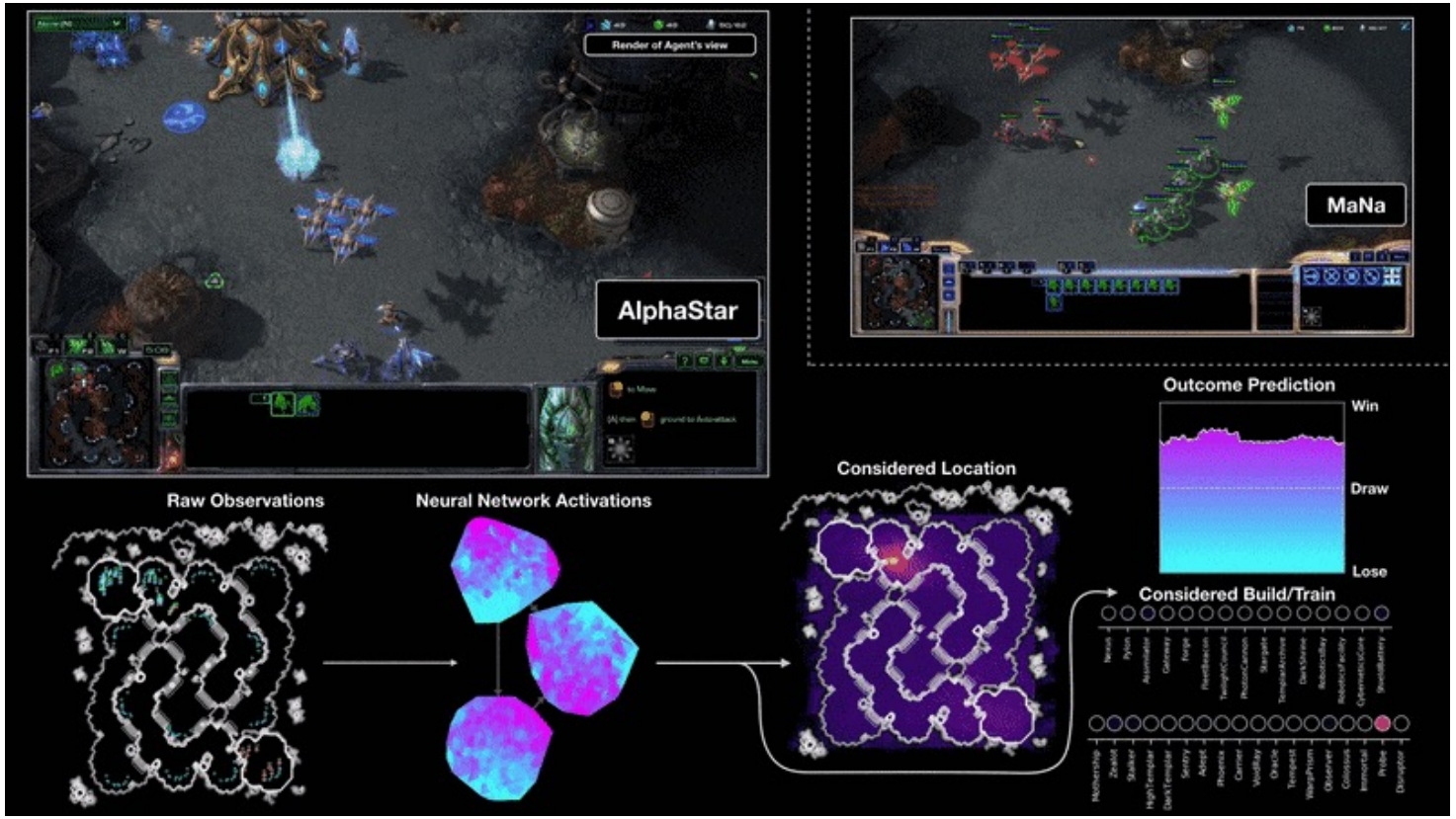
- Zeitreihenprognose
- Machine Translation
- Spracherkennung
- Text Generation
- Erkennung von handgeschriebener Schrift
- ...

LSTM cells Wrap-Up



- Drei Gates (Sigmoids):
  - Forget gate: kontrolliert was vom cell state gelöscht wird
  - Input gate: kontrolliert was zum cell state hinzugefügt wird
  - Output gate: kontrolliert welcher Teil des Cell states als Output ausgegeben wird
- Peephole Connections (Gers & Schmidhuber, 2000):
  - Erweiterung um drei sog. Peephole Verbindungen: Ergänzt den Cell state C_{t-1} zum Input des Forget und Input Gates und C_t zum Input des Output Gates
  - Das dargestellte LSTM hat Peephole connections zum Forget und Input Gate

Alphastar



- DeepMind stellt 2019 eine KI vor, welche professionelle Starcraft II Spieler schlägt
- > 99.8% offiziell gerankter menschlicher Spieler
- Tiefes NN mit LSTM
- Zunächst Supervised Training mit aufgezeichneten Spielen
- Danach Reinforcement Learning = agent vs agent games

Textgeneration

Andrej Karpathy blog

About

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

There's something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for Image Captioning. Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I've in fact reached the opposite conclusion). Fast forward about a year: I'm training RNNs all the time and I've witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me. This post is about sharing some of that magic with you.

> *We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"*

By the way, together with this post I am also releasing code on Github that allows you to train character-level language models based on multi-layer LSTMs. You give it a large chunk of text and it will learn to generate text like it one character at a time. You can also use it to reproduce my experiments below. But we're getting ahead of ourselves; What are RNNs anyway?

VIOLA:

Why, Salisbury must find his flesh and thought

That which I am not aps, not a man and in fire,

To show the reining of the raven and the wars

To grace my hand reproach within, and not a fair are hand,

That Caesar and my goodly father's world;

When I was heaven of presence and our fleets,

We spare with hours, but cut thy council I am great,

Murdered and by thy master's ready there

My power to give thee but so much as hell:

Some service in the noble bondman here,

Would show him to her wine.

KING LEAR:

O, if you were a feeble sight, the courtesy of your law,

Your sight and several breath, will wear the gods

With his heads, and my hands are wonder'd at the deeds,

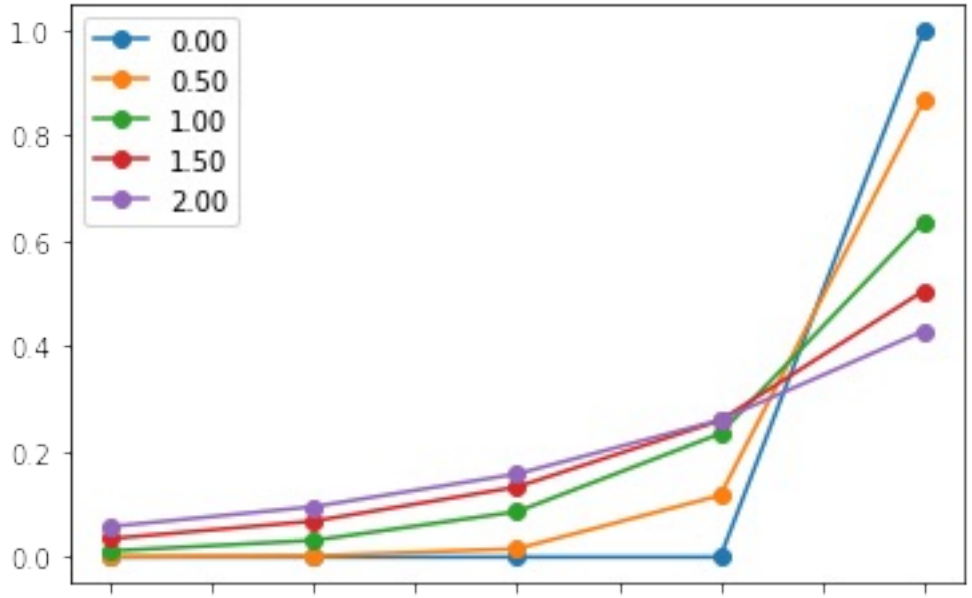
So drop upon your lordship's head, and your opinion

Shall be against your honour.

Textgeneration

Seltene Wörter können Folgefehler hervorrufen → Deshalb werden Tricks verwendet um natürlich klingenden Text zu generieren, der gleichzeitig nicht „degeneriert“.

Temperature sampling:
Niedrige Temperatur → Hohe Wahrscheinlichkeiten werden höher, Kleine kleiner.
Hohe Temperatur → Kleine Wahrscheinlichkeiten werden größer, Große kleiner.



Source: https://towardsdatascience.com/how-to-sample-from-language-models-682bceb97277