Einführung in die Theorie der
Neuronalen Netze

Vorlesung 9

LSTM

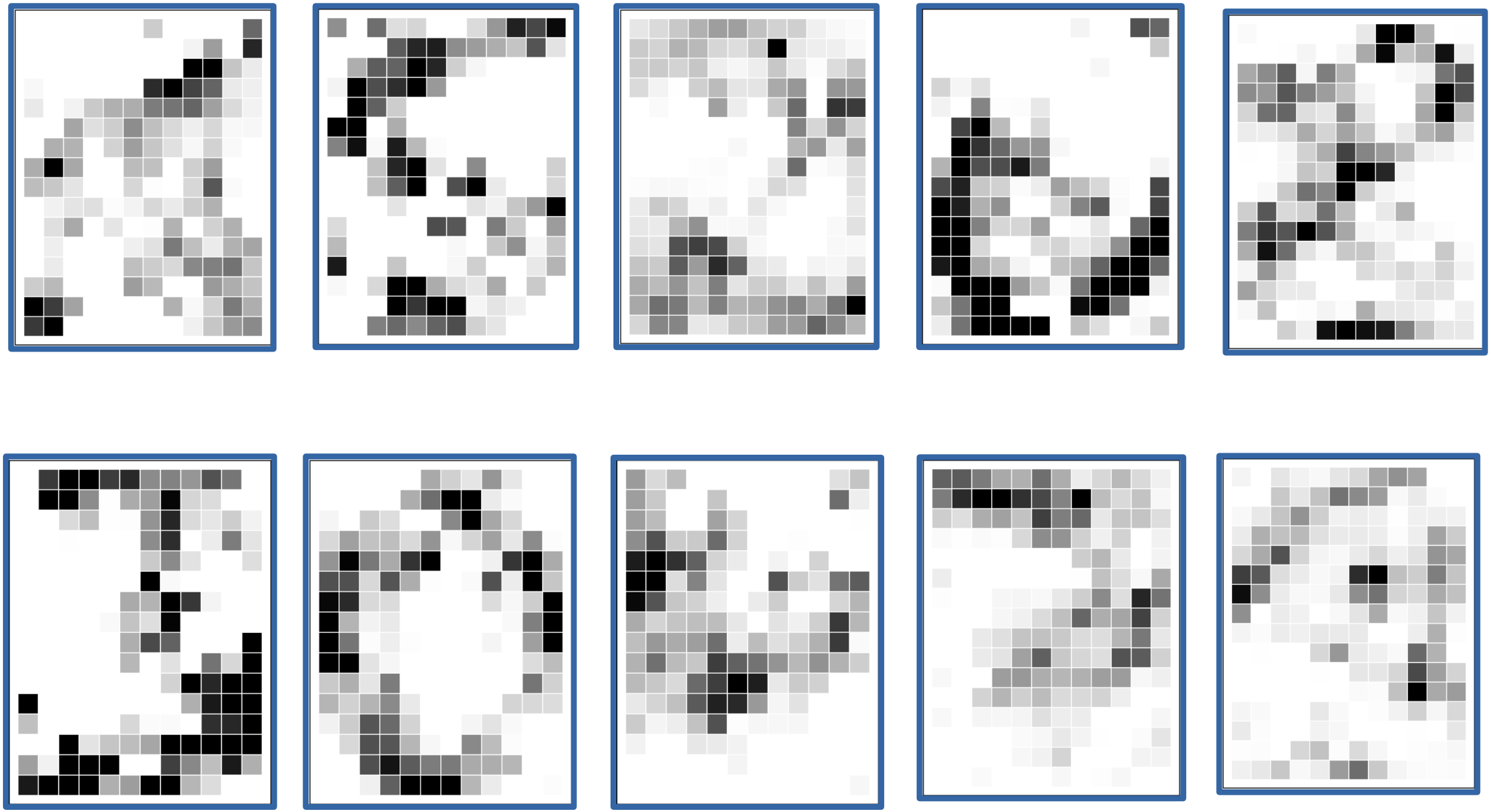
Alexander Förster
Universität Bremen

Wintersemester 2016 / 2017

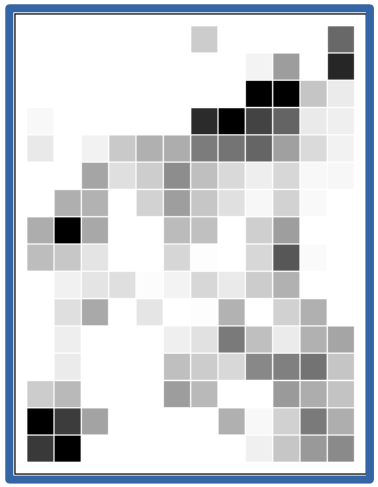
Weihnachtsaufgabenbesprechung: Wenn neuronale Netze träumen

- Diese Aufgabe baut auf die letzte Übung auf: Verwende das trainierte neuronale Netz für die Ziffernerkennung. Zum trainieren des neuronalen Netzes verwenden wir den Trick, dass beim Rückwärtsschritt die Gewichte (w) und die Ausgaben der Neuronen (o) ihre Rollen tauschen. **Nun** minimieren wir den Fehler durch Gradientenabstieg bezüglich der Eingabe.
- **Vorgehensweise**:
  - Erzeuge ein zufälliges Bild.
  - Berechne die Ausgabe des Netzes.
  - Die Ziffernklasse mit der höchsten Aktivierung wird als gewünschte Ausgabe angesehen. Diesbezüglich wird der Fehler berechnet.
  - Beim Rückwärtsschritt wird nun nicht $\partial E/\partial w$ sondern $\partial E/\partial o$ berechnet und in der Eingabeschicht auch korrigiert.
  - Erzeuge mit dem korrigierten Bild eine neue Ausgabe und wiederhole die Schritte, bis der Fehler klein genug ist.
  - Gebe eine geeignete Sequenz der Eingaben aus.

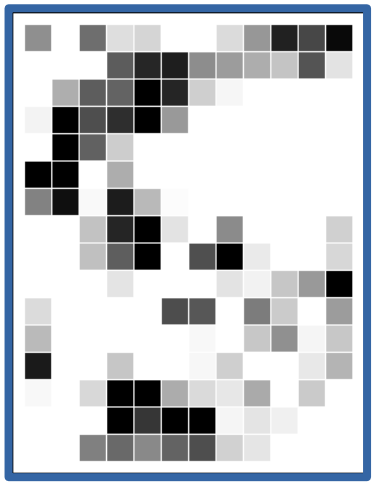
Beispieleingabenausgaben



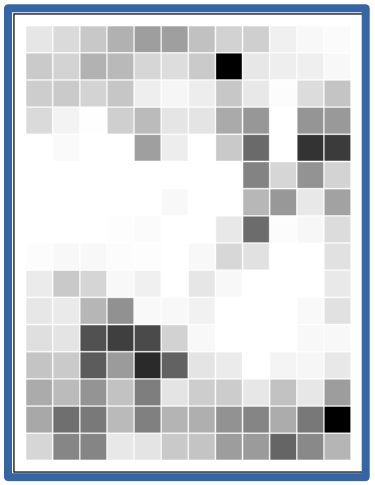
Beispieleingabenausgaben



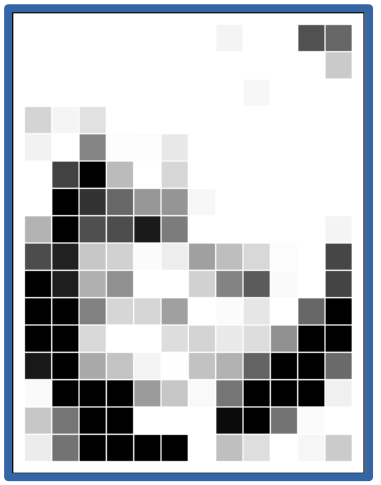
1



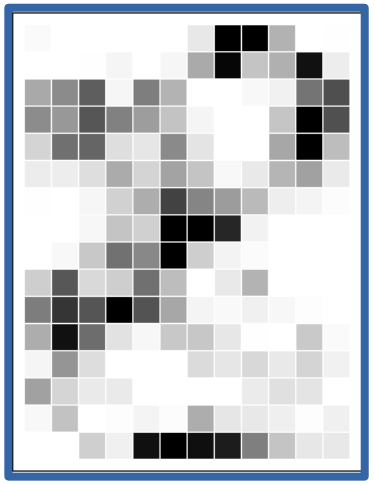
5



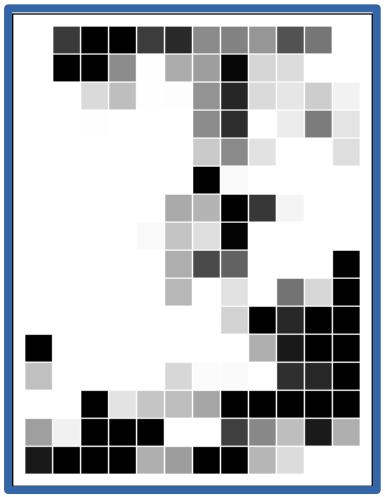
2



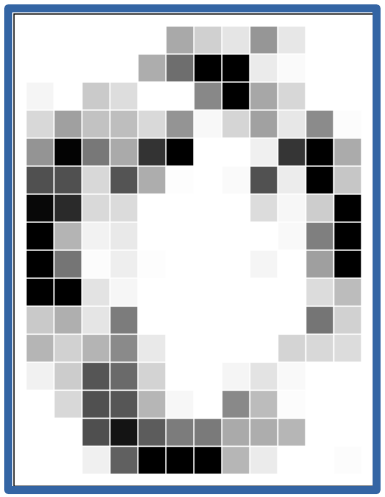
6



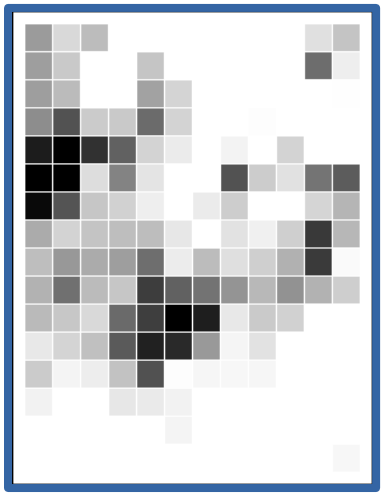
8



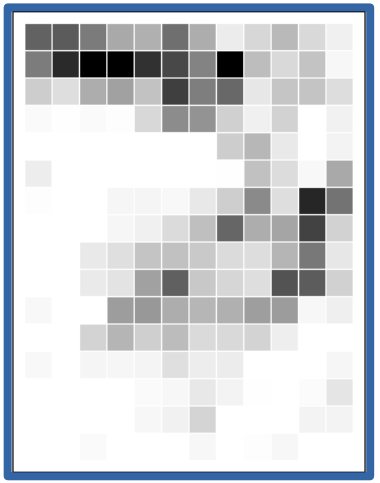
3



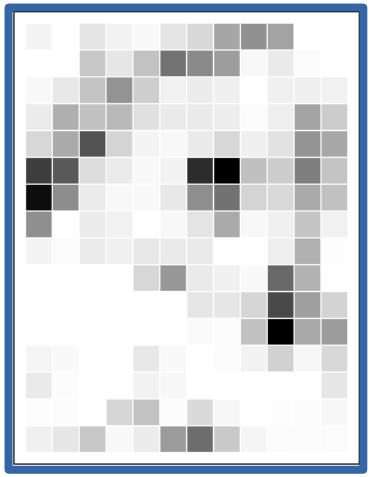
0



4

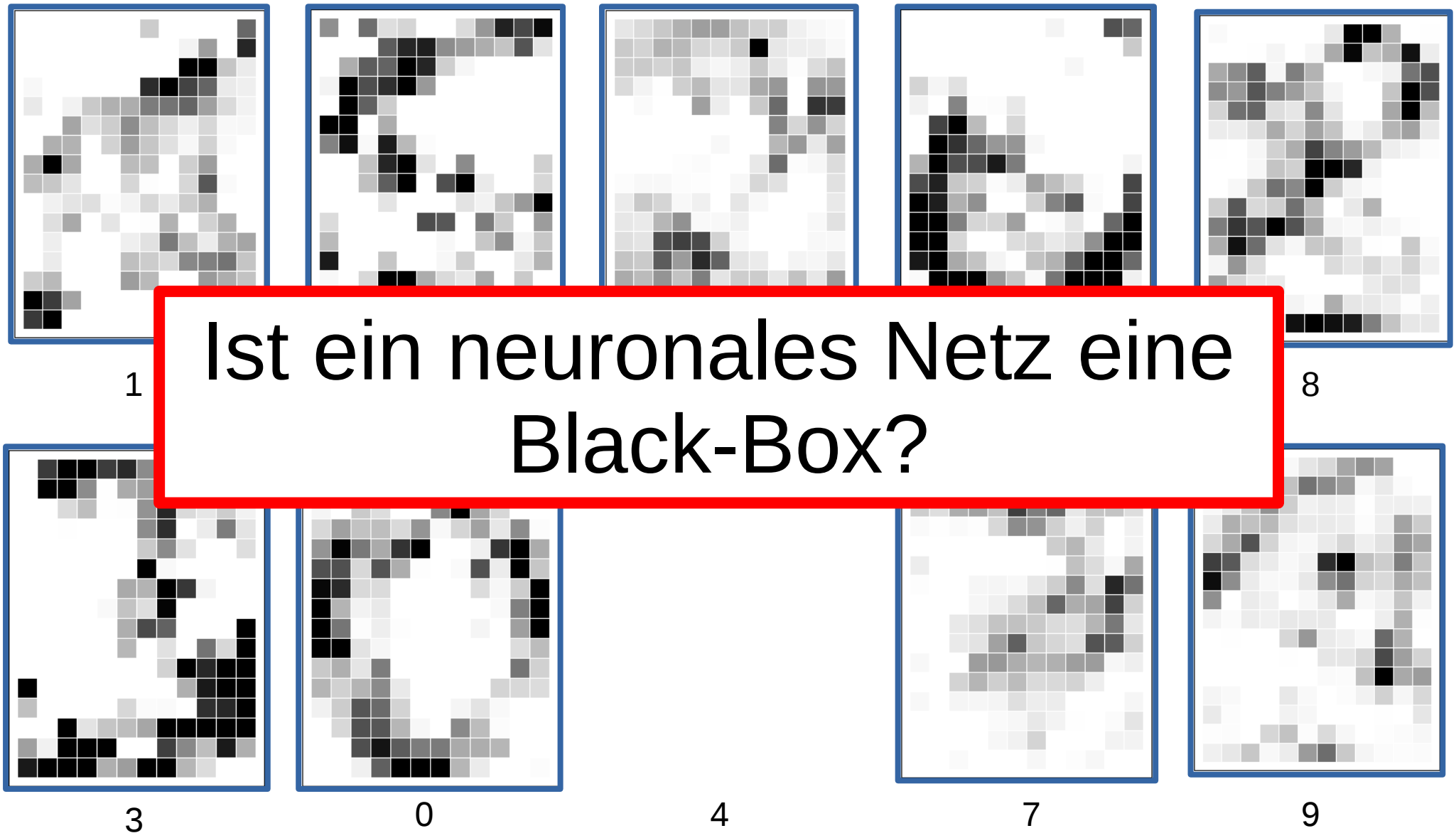


7



9

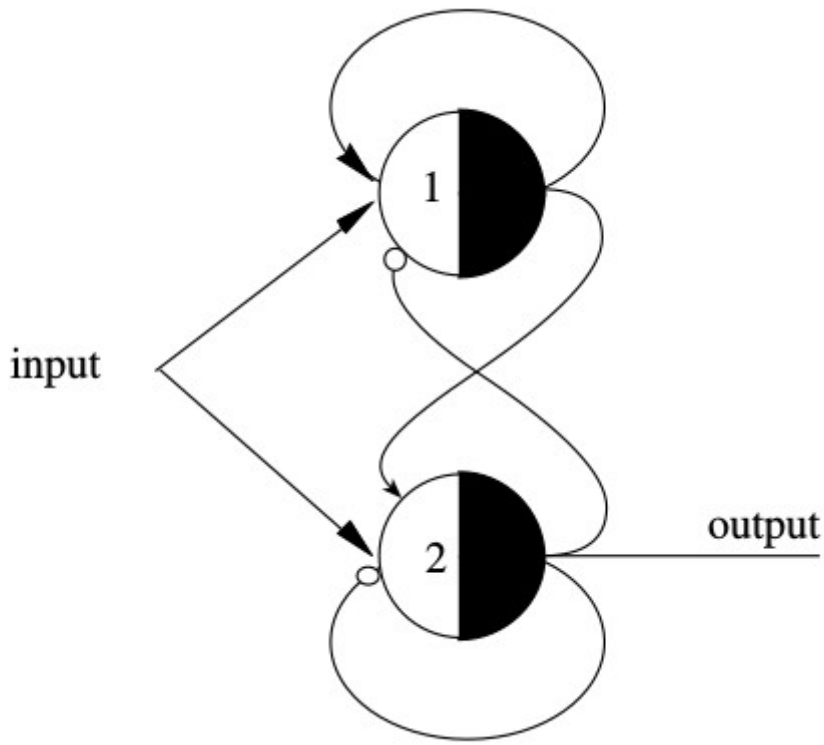
Beispieleingabenausgaben



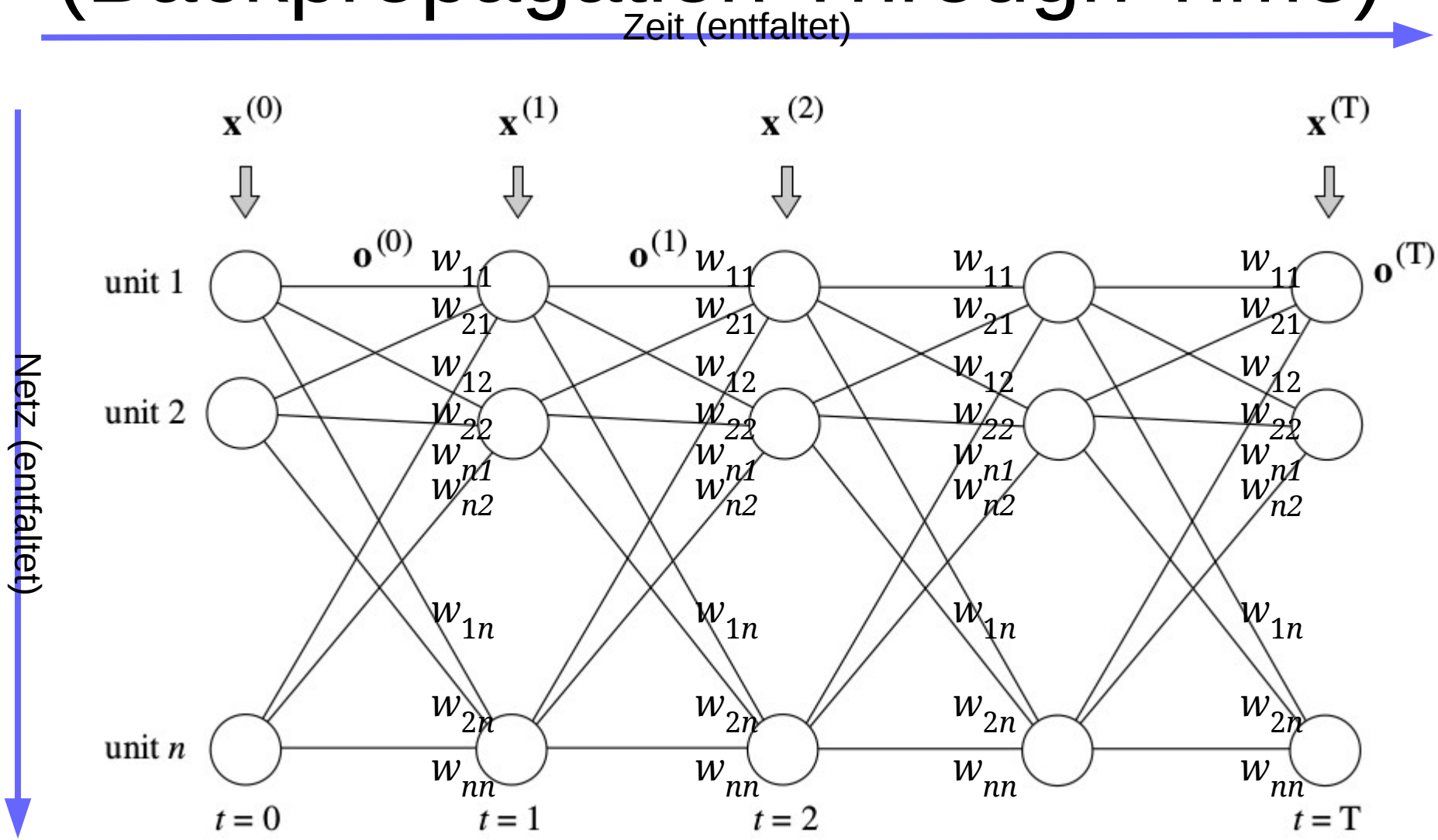
Ist ein neuronales Netz eine Black-Box?

Wiederholung: Rekurrente Netze

- Eingabe $x^{(t)}$ und Berechnung der neuen Zustände und der Ausgaben $o^{(t)}$ zum Zeitpunkt t haben Auswirkungen auf die Berechnungen zum Zeitpunkt t+1.
- Das System produziert i.d.R. unterschiedliche Ausgaben mit konstanten oder zeitlich variierenden Eingaben.
- Ein rekurrentes Netz kann als endlicher Automat aufgefasst werden.
- **Frage**: Wie kann ein solches Netz trainiert werden, um eine gewünschte Ausgabe zu erreichen.



Wiederholung BPTT (Backpropagation Through Time)



Wiederholung: Problem von BPTT

- *„Bei BPTT haben die in der Zeit **zurück geschickten Fehler** entweder die Tendenz (1) zu explodieren oder (2) zu verschwinden. Das zeitliche Verhalten des Fehlers hängt exponentiell von der Größe der Gewichte ab. Im Fall (1) oszilieren die Gewichte, im Fall (2) benötigt das Lernen extrem viel Zeit oder das Netz lernt nichts.“* (Hochreiter, S., Schmidhuber J., „Long short-term menory“, Neural Computation 9(8):1735-1780, 1997)
- Wenn das Netz nicht „tief“, also *T* klein (T<10) bleibt kann ein BPTT Netz gut lernen. Allerdings lernt es dann nichts aus der längeren Vergangenheit.

Wiederholung: Problem von BPTT

- *„Bei BPTT haben die in der Zeit **zurück geschickten Fehler** entweder die Tendenz (1) zu explodieren oder (2) zu verschwinden. Das zeitliche Verhalten des Fehlers hängt exponentiell von der Größe der Gewichte ab. Im Fall (1) oszilieren die Gewichte, im Fall (2) benötigt das Lernen extrem viel Zeit oder das Netz lernt nichts.“* (Hochreiter, S., Schmidhuber J., „Long short-term menory“, Neural Computation 9(8):1735-1780, 1997)
- Wenn das Netz nicht „tief“, also *T* klein (T<10) bleibt kann ein BPTT Netz gut lernen. Allerdings lernt es dann nichts aus der längeren Vergangenheit.
- Was können wir machen?

Nachteile von BPTT umgehen

- **Möglichkeit 1**: Keine Rekurrenz benutzen. Zeitfenster als parallele Eingabe und ein normales, recht flaches FFN.
  - Siehe letzte Vorlesung
  - Nachteil ist, dass die Vergangenheit ist relativ kurz oder die Eingabe wird recht breit und damit sind wieder sehr viele Gewichte zu lernen, wenn die Schichten vollständig verbunden sind.

Nachteile von BPTT umgehen

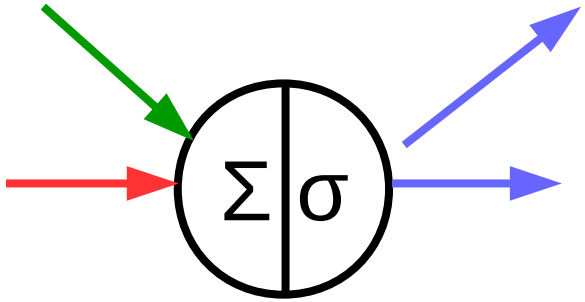
- **Möglichkeit 1**: Keine Rekurrenz benutzen. Zeitfenster als parallele Eingabe und ein normales, recht flaches FFN.
  - Siehe letzte Vorlesung
  - Nachteil ist, dass die Vergangenheit ist relativ kurz oder die Eingabe wird recht breit und damit sind wieder sehr viele Gewichte zu lernen, wenn die Schichten vollständig verbunden sind.
- **Möglichkeit 2**: Den Fehler nicht zurückgeben!
  - Dann kann er auch nicht oszillieren oder sich auslöschen.
  - Wie kann man dann mit Fehlerminimierung lernen?

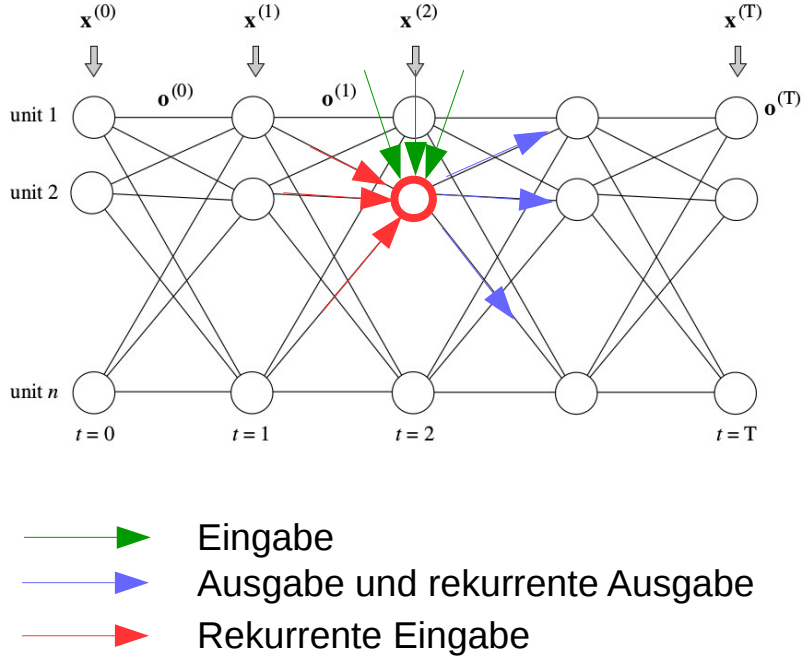
Long Short-Term Memory (LSTM)

- Von Josef (Sepp) Hochreiter entwickelt nach mathematischer Analyse (Hochreiter, 1991) der Nachteile normaler RNNs. (Hochreiter et al., 1997)
- Grundidee: „Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing **constant error flow through 'constant error carrousels' within special units.** *Multiplicative gate units* learn to open and close access to the constant error flow.“
- Es gibt unzählige Variationen die darauf aufbauen. Gleiche Grundidee aber unterschiedliche Feinheiten. LSTM ist nicht gleich LSTM. (Greff et al., 2015)

BPTT Idee

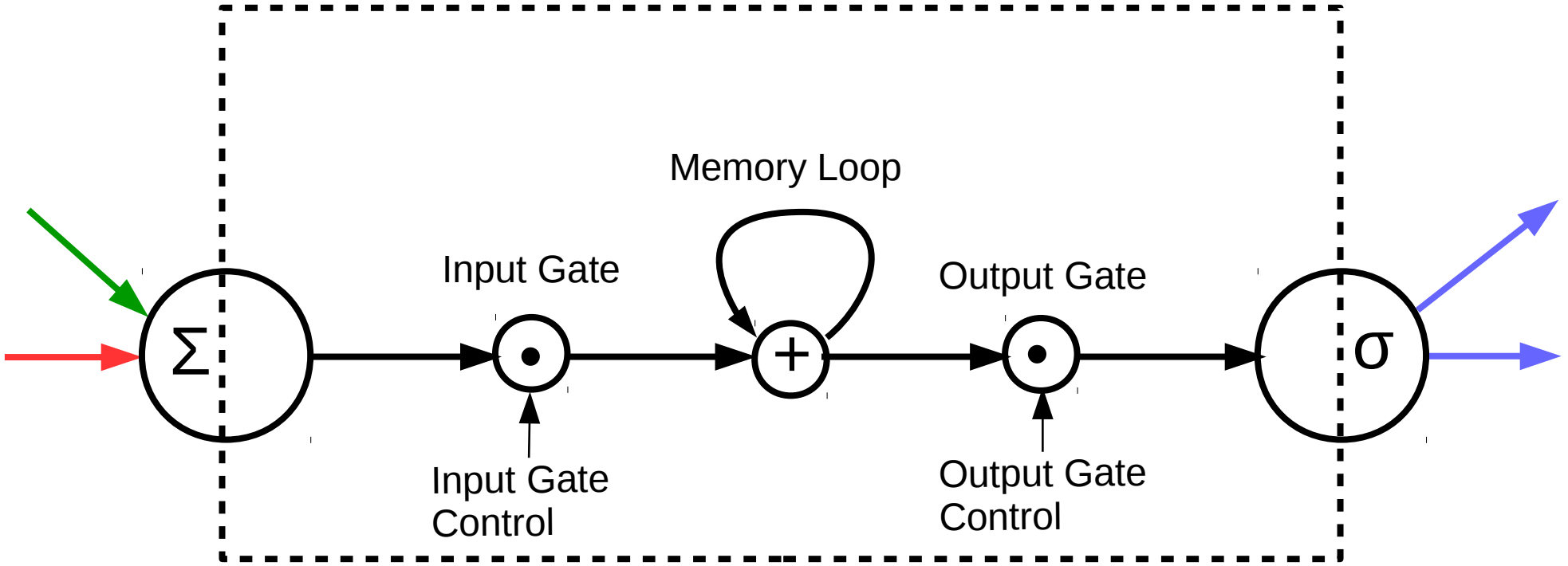
- Gewöhnlich fließt die Information über die Verbindungen durch das Netz. Es gibt keine Speicherzellen, die Information halten.





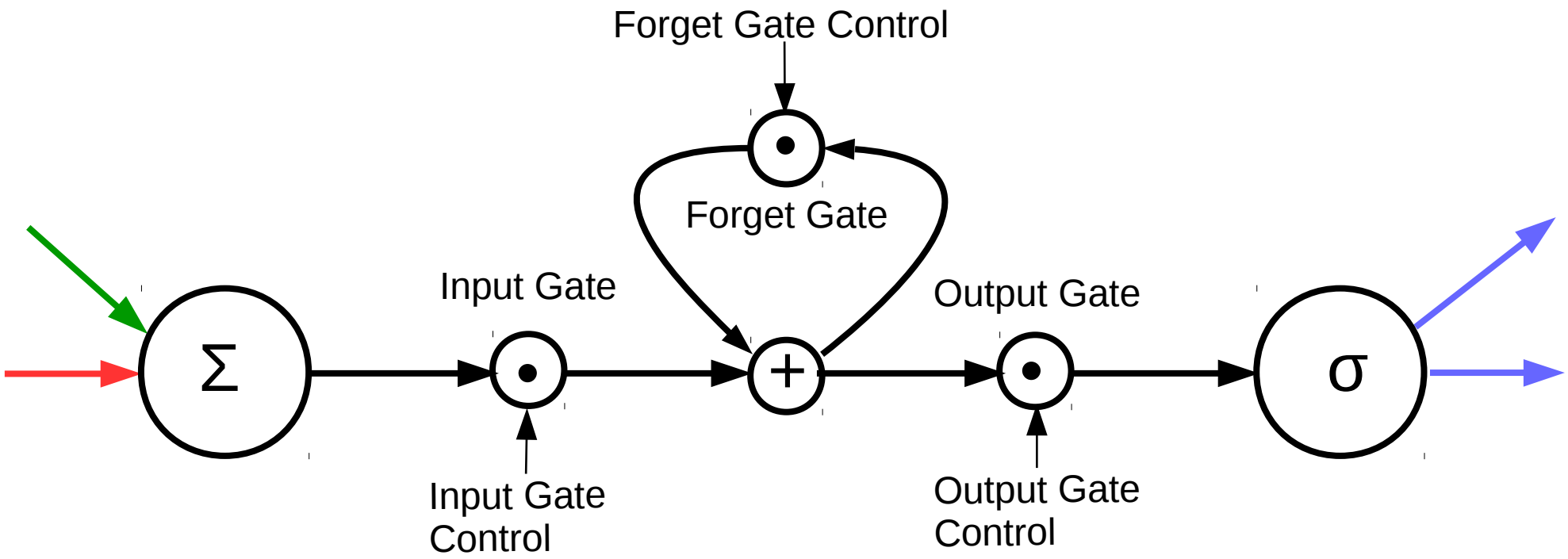
LSTM Idee

- Schalter regeln den Fluss von Information in und aus der Zelle für einen Zeitschritt, und folglich auch den Fluss von Fehlern durch die Zeit rückwärts.
- Die Langzeitinformation wird in der Schleife gespeichert und nicht mehr im ganzen Netz (über alle rekurrenten Verbindungen).
- Wir haben immer noch eine schönes „Funktionennetz“: Die Schalter sind Multiplikativ, Schleife ist additiv (alles nur Skalare)



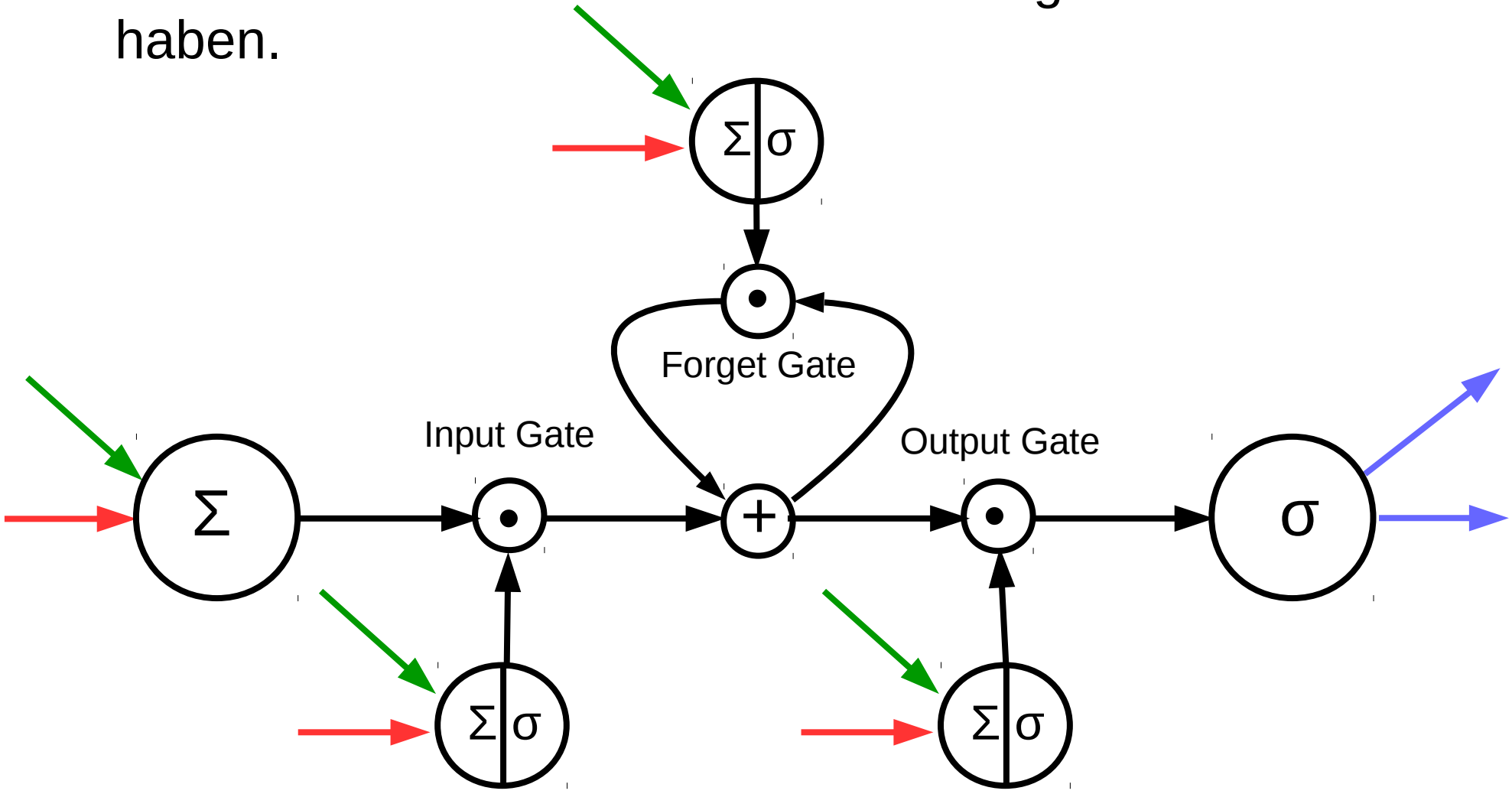
LSTM Idee

- Im Backpropagation Schritt passiert der Fehler nur die Zelle, wenn gerade die Schalter offen sind. Sonst ist der Fehler 0 und wird nicht weiter zurück gegeben.



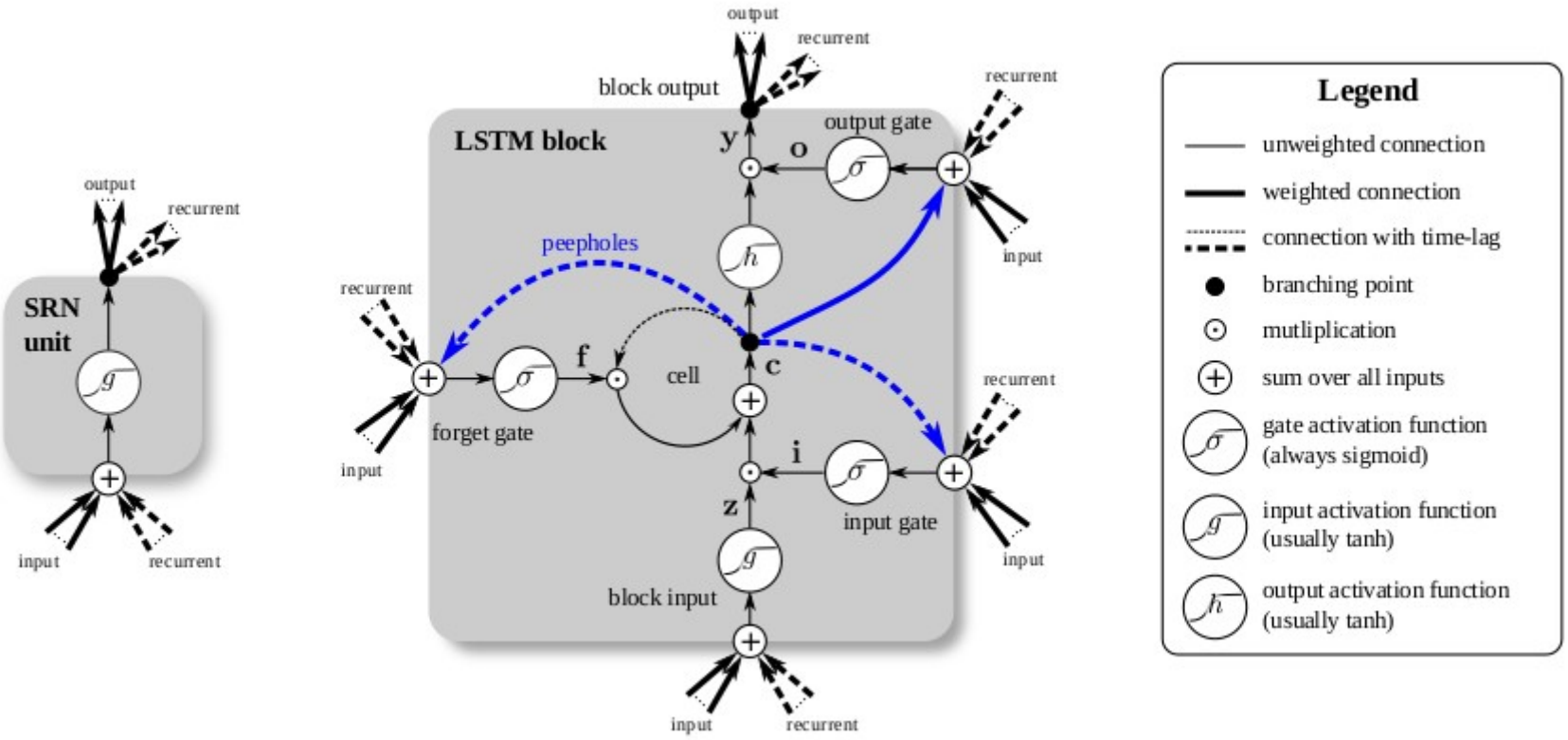
LSTM Idee

- Die Schalter werden auch durch gewöhnliche Zellen aktiviert, denen alle Eingaben zur Verfügung stehen und die selbst eine Transferfunktion und eigene Gewichte haben.



Standard LSTM Zelle

- Die Transferfunktionen von und zum CEC (Constant Error Carrousel) müssen nicht unbedingt Sigmoiden sein.
- Für die Kontrolle der Schalter kann auch die Information des aktuellen Zustands der Zelle genutzt werden (Gers et al., 2000): Durch Gucklöcher (peepholes)
- In Variationen fehlen manche Schalter oder Transferfunktionen oder Schalter sind gekoppelt.



[Quelle: Greff et al., LSTM: A Search Space Odyssey, 2015.]

LSTM Berechnungen

- Die Berechnungen sind nicht komplizierter als bei gewöhnlichen Netzen. Es gibt nur mehr Bezeichner für Zellen und Gewichtsmatrizen.
- Für die Formeln für den Rückwärtsschritt sein auf Greff et al. (2015) verwiesen.

- x^t ist der Eingabevektor zur Zeit t.
- W sind Matrizen für die Eingabegewichtung
- R sind quadratische Matrizen für die Rekurrenten Verbindungen
- p sind die Gucklöcher Gewichtsvektoren
- b sind Bias-Vektoren
- σ,g und h sind Aktivierungsfunktionen (Sigmoide und/oder Hyperbolischer Tangens)
- $\odot$ ist die punktweise Multiplikation.

$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) \qquad \textit{block input}$$
$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \qquad \textit{input gate}$$
$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) \qquad \textit{forget gate}$$
$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} \qquad \textit{cell state}$$
$$\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o) \qquad \textit{output gate}$$
$$\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t) \qquad \textit{block output}$$

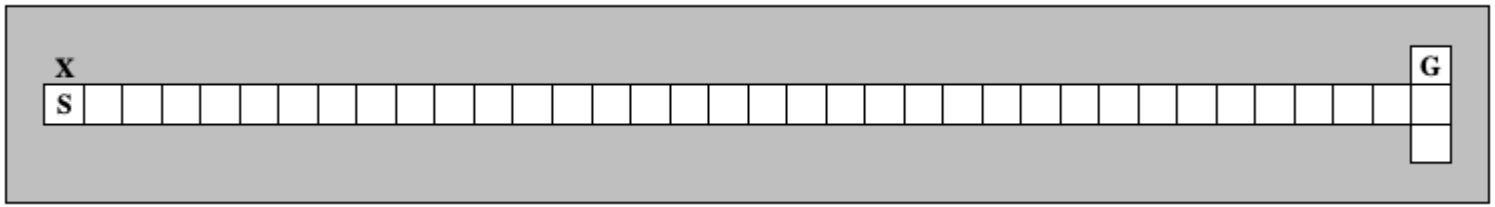
[Quelle: Greff et al., LSTM: A Search Space Odyssey, 2015.]

LSTM Beispiel: CFL

- Kontext freie Sprachen der Form
  - $a^n b^n \rightarrow$ ab, aaabbb, aaaaaabbbbbb
  - $a^n b^n c^n \rightarrow$ abc, aaaaabbbbbccccc
  - $a^n b^m B^m A^n \rightarrow$ abbbBBBA, aaaaabbBBAAAAA
- LSTMs lernen Sequenzen mit 2000,200, bzw. 1500 Zeichen (Gers, 2001).

LSTM Beispiel: T-Maze

- Klassisches Beispiel für Langzeitspeicher ist das T-Labyrinth (T-Maze).
- Das Eingabesignal am Anfang zeigt an, ob am Ende nach unten oder nach oben gegangen werden soll. Der Korridor kann sehr lang sein. In dieser Zeit muss sich der Anfangswert gemerkt werden.



[Quelle: Wierstra et al., Recurrent Policy Gradients, 2009.]

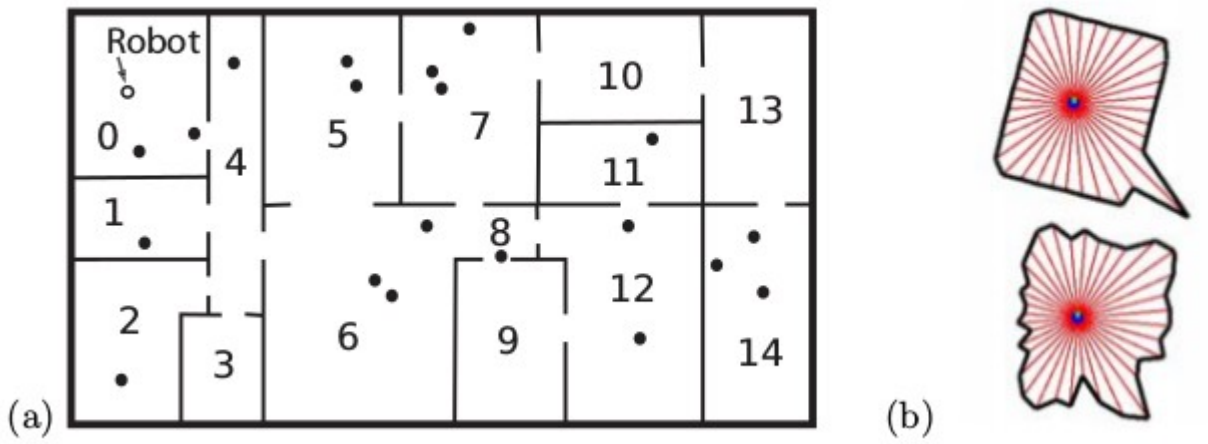
LSTM Beispiel: TORCS

- Das Fahrzeug muss möglichst schnell um den Kurs fahren. Kontrollparameter sind Gas, Bremse, Lenkung, Eingaben sind Abstand zum Rand, Winkel zur Strecke und Krümmung der Strecke.
- Um möglichst schnell zu fahren muss der Kurs auswendig gelernt werden.
- Methode Verbindet LSTM mit Reinforcement Lerning.

[Quelle: Wierstra et al., Recurrent Policy Gradients, 2009.]

LSTM Beispiel: Maps

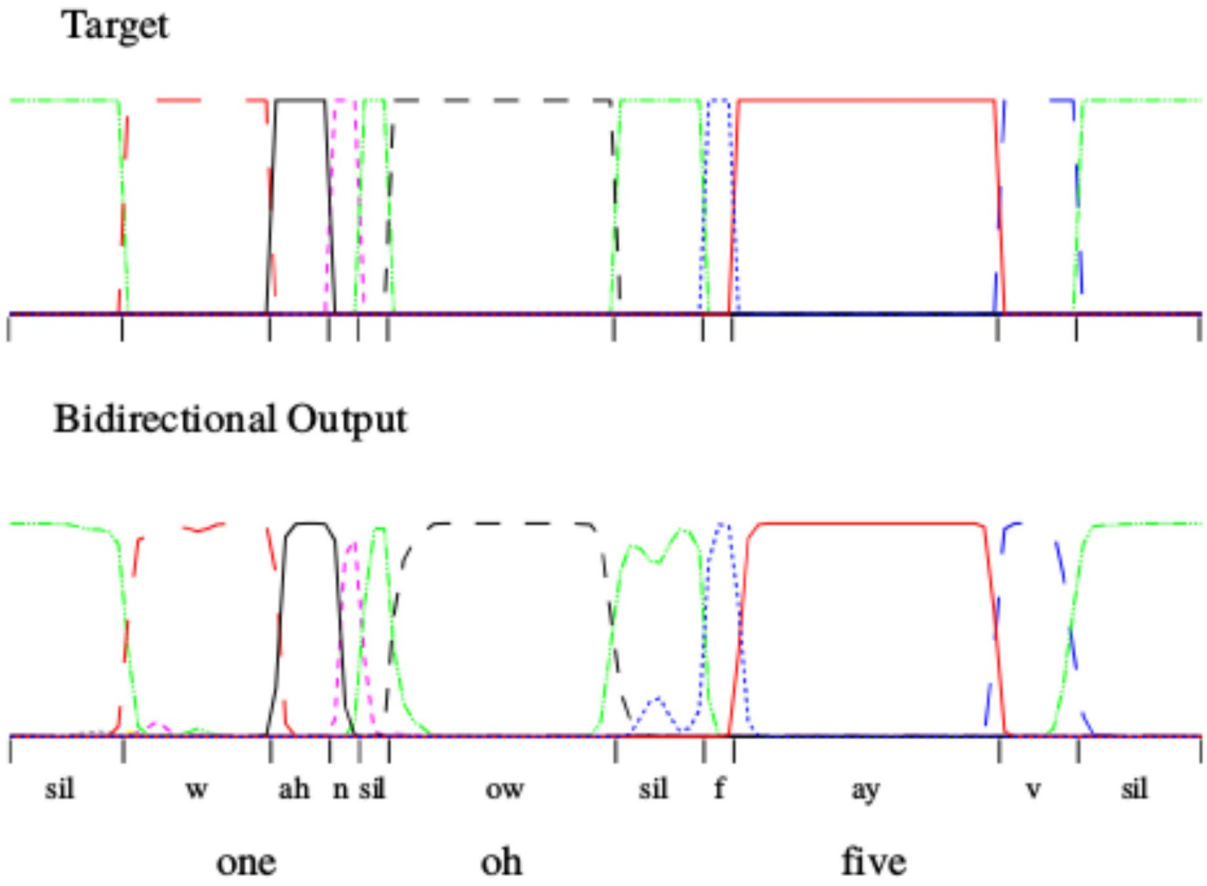
- Roboter muss sich mit verrauschtem Laserscanner in Umgebung lokalisieren. Bewegliche Objekte behindern die Sicht.
- Das LSTM Netz integriert die Sensorinformation über die Zeit und kann Übergänge zwischen zwei Räumen erkennen.



[Quelle: Förster et al., RNN-based Learning of Compact Maps for Efficient Robot Localization, 2007.]

LSTM Beispiel: Phoneme

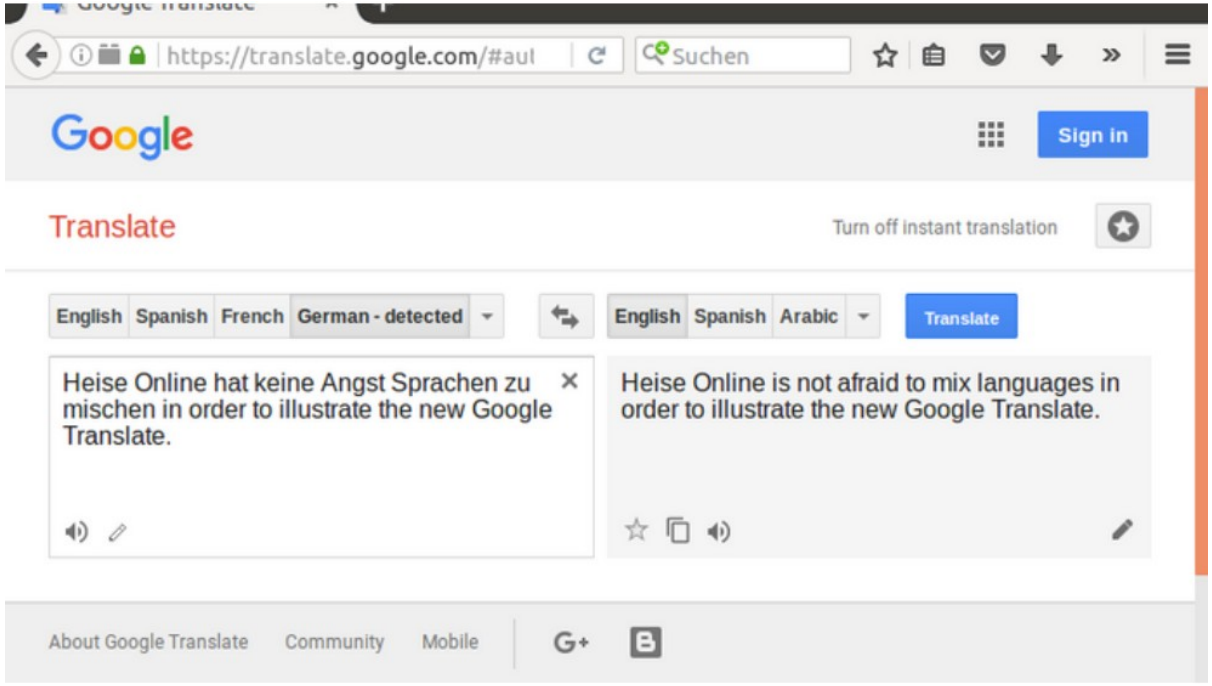
- Audiodaten werden dem B-LSTM Netz zugeführt. Es werden die Phoneme erkannt.



[Quelle: Graves et al., Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures, 2005.]

LSTM Beispiel: Google Translate

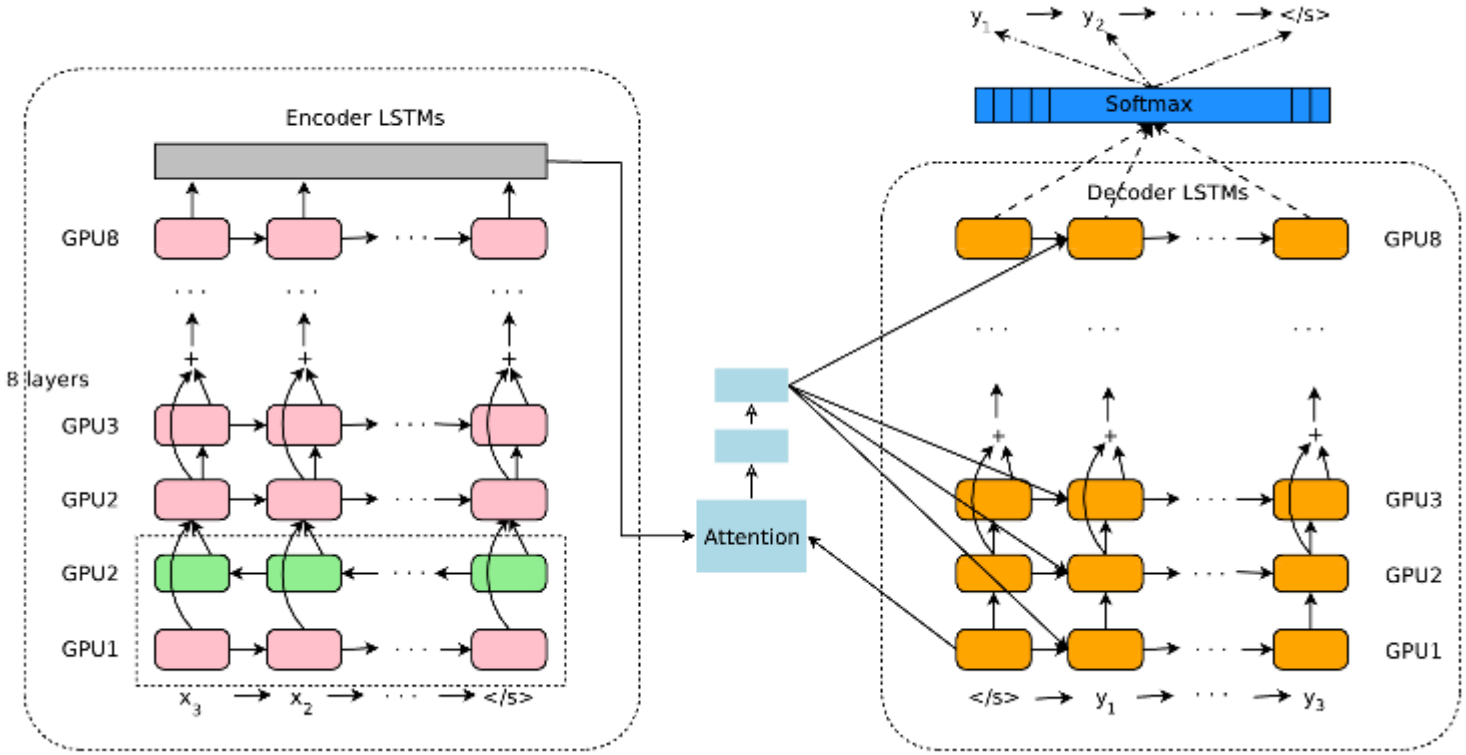
- Google Translate besteht aus drei verzahnt arbeitenden neuronalen Netzen. Zwei LSTM-Netze mit je 8 Schichten und ein normales Netz mit 3 Schichten.



[Quelle: www.heise.de, Google: Translate-KI übersetzt dank selbst erlernter Sprache, 25.11.2016.]

LSTM Beispiel: Google Translate

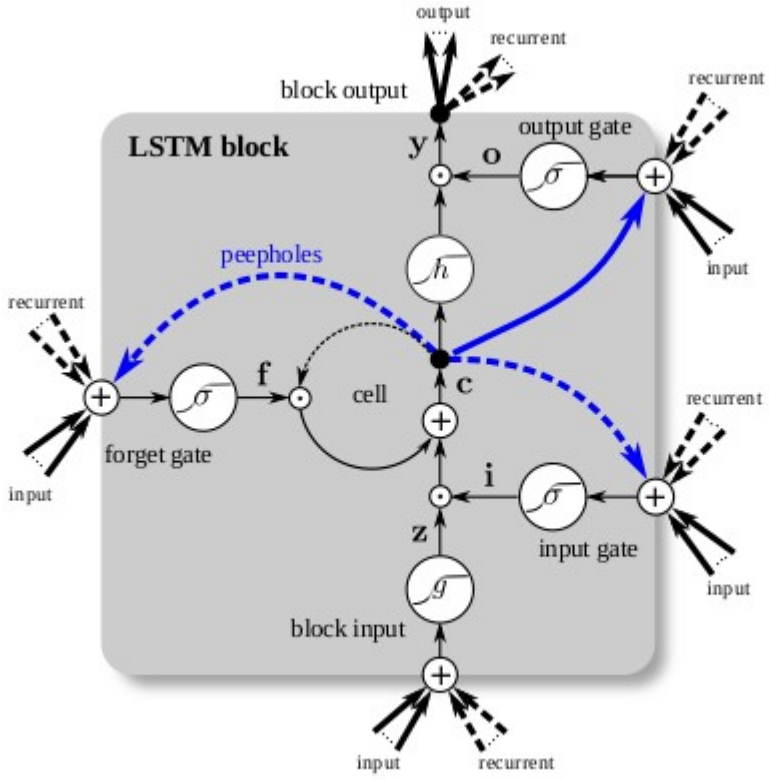
- The error strikes back! Nun gibt es ein Lernproblem nicht mit der Zeit, sondern mit den Abstraktions-Schichten. Information muss hier über die Schichten transportiert werden.
- Dies geschieht über „Residual Connections“, die die Eingabe-Information an die nächste Eingabeschicht durchreichen.



[Quelle: Wu et al., Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016]

Aufgabe

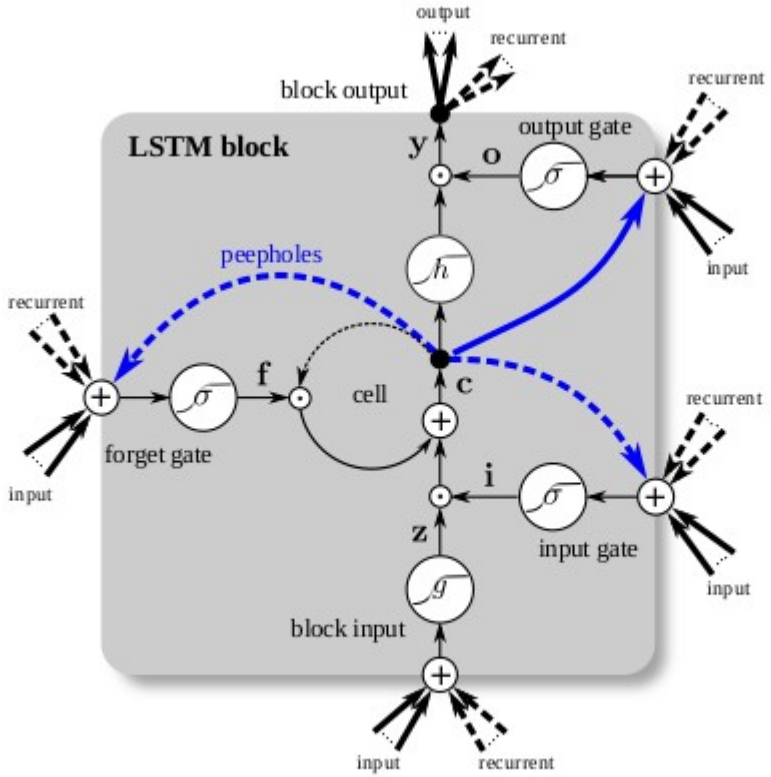
- Entwerfe **eine** LSTM-Zelle, die eine 1 am Eingang genau 10 Zeitschritte später am Ausgang ausgibt Falls zwischendurch eine neue 1 am Eingang anliegt soll die Zählung von vorne beginnen.
- Du kannst aber musst nicht jedes Feature der Zelle benutzen.
- **Verständnisfrage:** Kann das Problem auch mit einem rekurrenten McCulloch-Pitts-Netz gelöst werden? Wie wäre dort die Vorgehensweise?



[Quelle: Greff et al., LSTM: A Search Space Odyssey, 2015.]

Optional

- **Lerne** das Problem mit einer LSTM Zelle.
- Benutze dazu besser eine Bibliothek, zum Beispiel Keras (https://keras.io/), anstatt alles per Hand zu implementieren.



[Quelle: Greff et al., LSTM: A Search Space Odyssey, 2015.]

Links

- Wikipedia über LSTM mit weiteren Beispielen.
- Heise Meldung über Google Translate welches aus ANNs besteht.
- LSTM für Dummies. Von RNN zu LSTM.
- Zeitreihenvorhersage mit LSTM. Nicht die beste Anwendung, aber es wird gezeigt wie man es in Python mit Keras (einer Bibliothek für Neuronale Netze) programmiert.

Literatur

- Sepp Hochreiter, Jürgen Schmidhuber, Long Short-Term Memory, Neural Computation 9(8): 1735-1780, 1997.
- Felis Gers, Long Short-Term Memory in Recurrent Neural Networks, PhD thesis 2366, EPFL, 2001.
- Alex Graves, Jürgen Schmidhuber, Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures, IJCNN 2005.
- Daan Wierstra, Alexander Förster, Jan Peters, Jürgen Schmidhuber, Recurrent Policy Gradient, Journal of Algorithms in Cognition, Informatics and Logic, Elsevier, 6 May 2009.
- Alexander Förster, Alex Graves, Jürgen Schmidhuber, RNN-based Learning of Compact Maps for Efficient Robot Localization, ESANN 2007.
- Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber, LSTM: A Search Space Odyssey, ICML 2014. (arXiv 1503.04069v1 [cs.NE] 13 Mar 2015)
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, arXiv 1609.08144v2 [cs.CL] 8 Oct 2016.