Recurrent Neural Networks

Marcin Walkowski 2022

ETI
Katedra Architektury Systemów Komputerowych
amazon

Plan for today

- Sequence models motivation
- RNNs
- LSTMs
- **Kahoot time!**

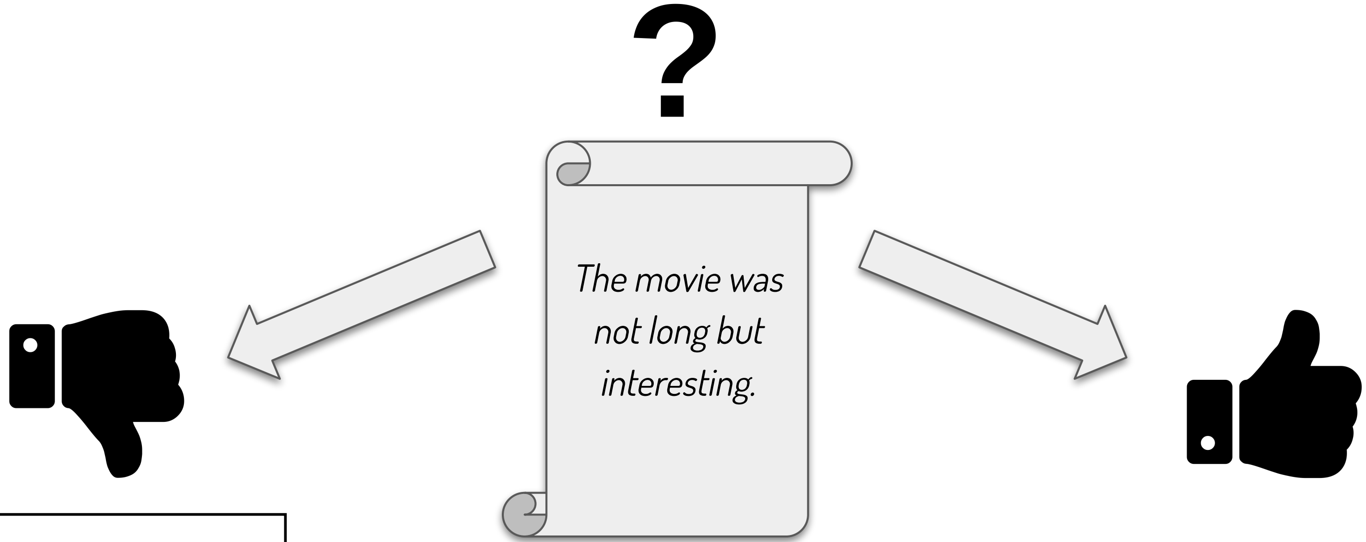
Resources

- Andrew Ng - Sequence Models
- Leo Dirac - LSTM is dead. Long Live Transformers!
- Christopher Olah - Understanding LSTM Networks

Motivation - Sequence data tasks

● Sentiment Classification	variable ↦ fixed size
● ECG Classification	variable ↦ fixed size
● Machine Translation	variable ↦ variable size
● ASR and TTS	variable ↦ variable size
● Music Generation	fixed ↦ variable size

Motivation - Sentiment Classification



Motivation - Bag-of-words

“The movie was not long but interesting.”

{'the': 1, 'movie': 1, 'song': 0, 'was': 1, 'is': 0, 'not': 1, 'long': 1, 'but': 1, 'interesting': 1}

“I like to sleep and I like to eat”

{'i': 2, 'like': 2, 'to': 2, 'sleep': 1, 'and': 1, 'eat': 1}

Motivation - Bag-of-words

- Fixed-length vector equal to dictionary size
- Information about the frequency
- No information about the order

Motivation - Bag-of-words

- Fixed-length vector equal to dictionary size
- Information about the frequency
- No information about the order

"The movie was not long but interesting."
"The movie was long but not interesting."

{'the': 1, 'movie': 1, 'song': 0, 'was': 1, 'is': 0, 'not': 1, 'long': 1, 'but': 1, 'interesting': 1}

Order matters!

Motivation - Bag-of-words

- n-grams are potential solution
- Vector size grows exponentially

The PWN spelling dictionary has approx. 140 thousands words.
By using bigrams, we get 19.6 billion values.

RNN

Recurrent Neural Network

RNN

$$f\left(x^{<1>}, x^{<2>}, \ldots, x^{<T_x>}\right)?$$

- How to compute a function for variable-length data?

RNN

$$f\left(x^{<1>}, x^{<2>}, \ldots, x^{<T_x>}\right)?$$

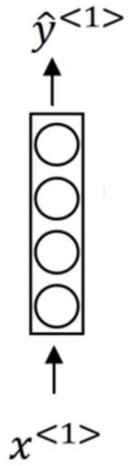
- How to compute a function for variable-length data?
- Recursively. Final activation is the final output

RNN - Word Representation

$$\begin{bmatrix} the \\ movie \\ was \\ not \\ long \\ but \\ interesting \end{bmatrix} \qquad was \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

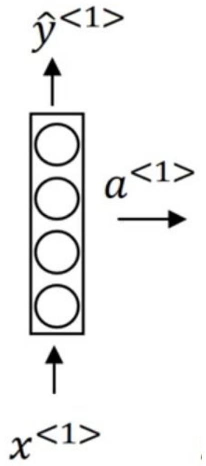
$$x^{<t>}$$

RNN - Diagram

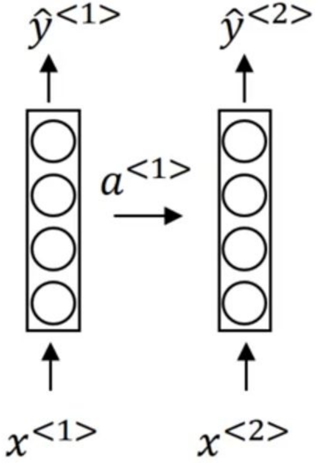


Source of diagrams in the next sections:
https://www.coursera.org/learn/nlp-sequence-models

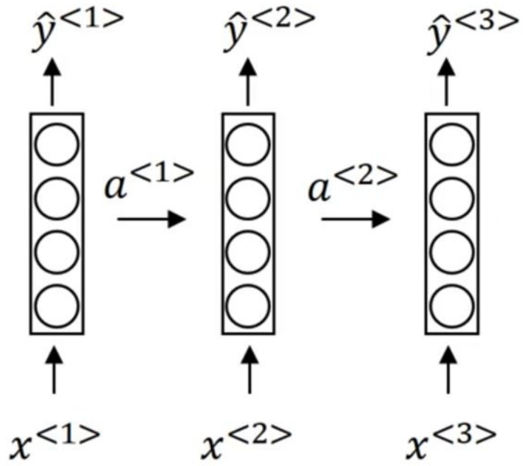
RNN - Diagram



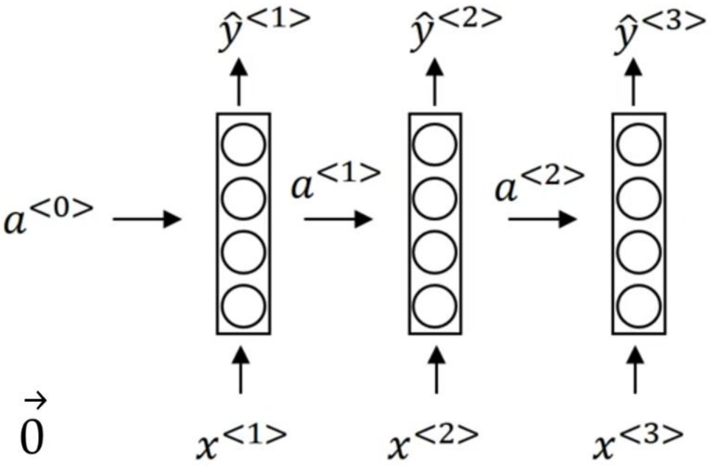
RNN – Diagram



RNN - Diagram

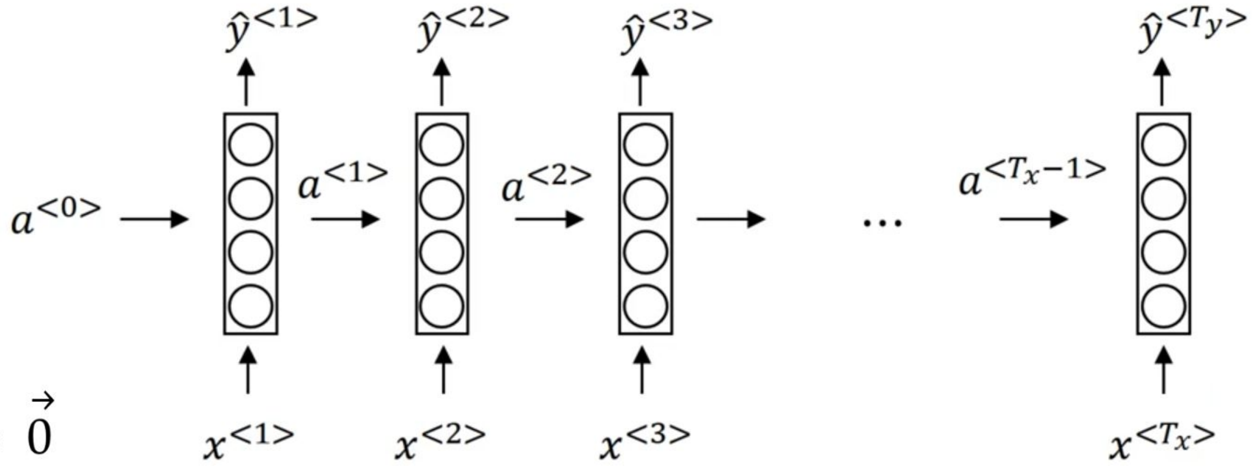


RNN - Diagram



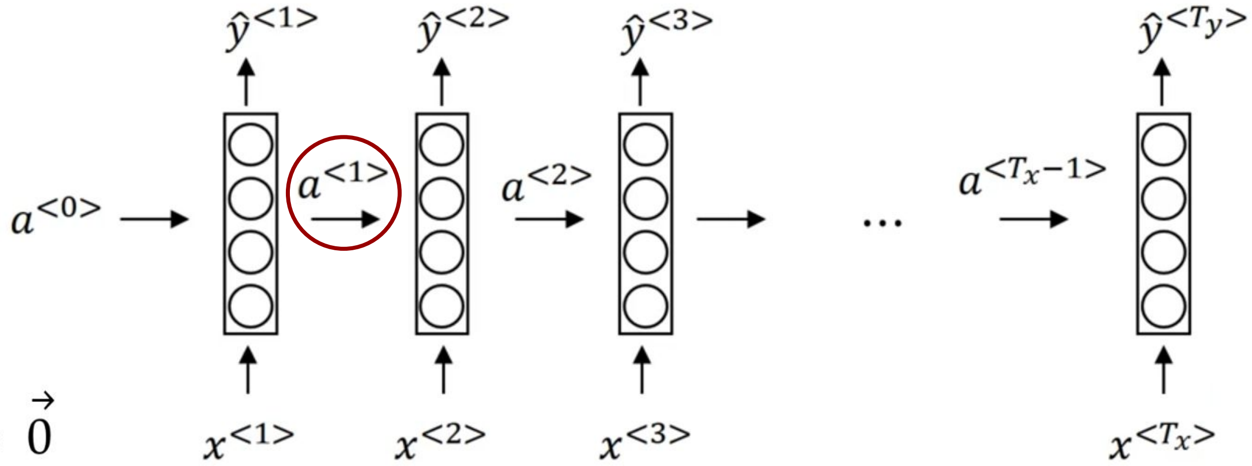
$$a^{<0>} = \vec{0}$$

RNN - Diagram



$$a^{<0>} = \vec{0}$$

RNN - Diagram

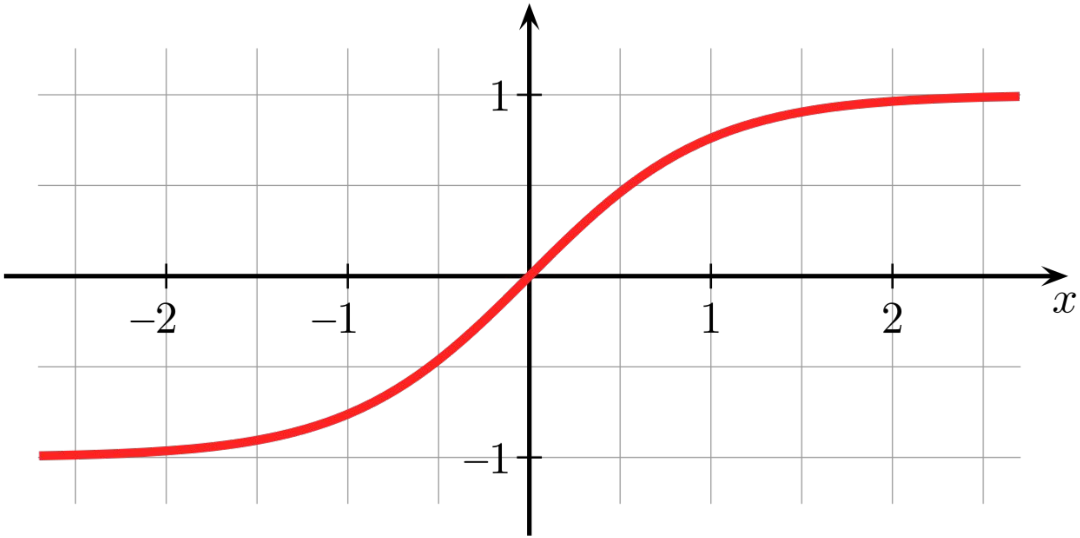


$$a^{<0>} = \vec{0}$$

$$a^{<1>} = g\left(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a\right)$$

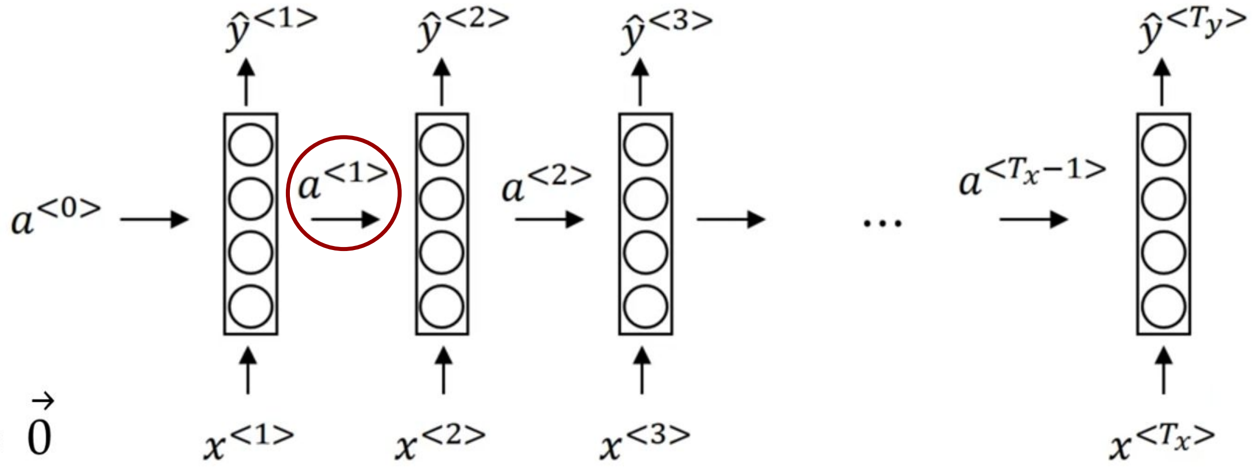
Hyperbolic Tangent

$$g(x) = tanh(x)$$



Source:
https://en.wikipedia.org/wiki/Hyperbolic_functions

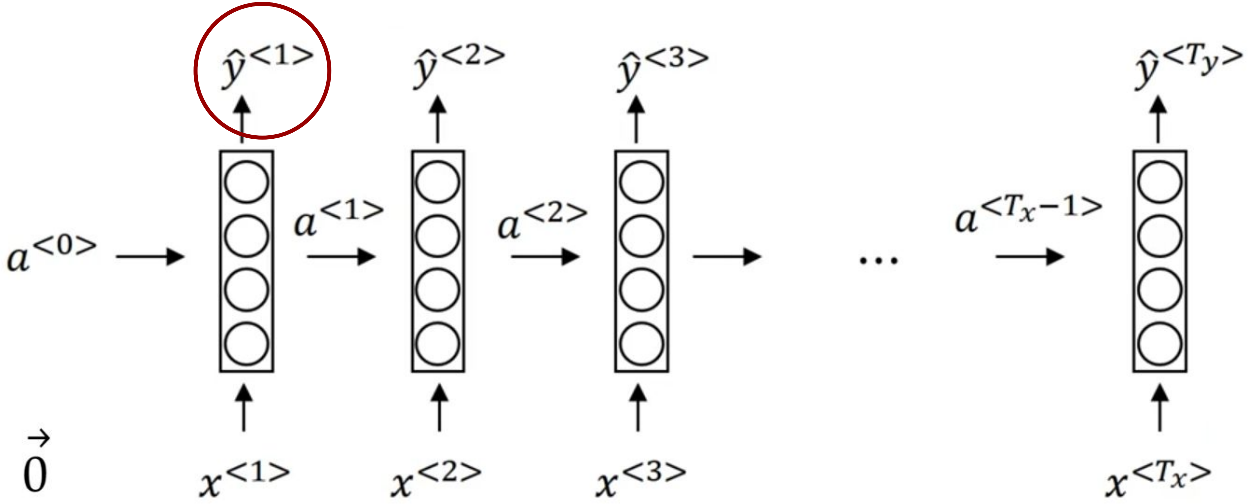
RNN - Diagram



$$a^{<0>} = \vec{0}$$

$$a^{<1>} = g\left(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a\right)$$

RNN – Diagram

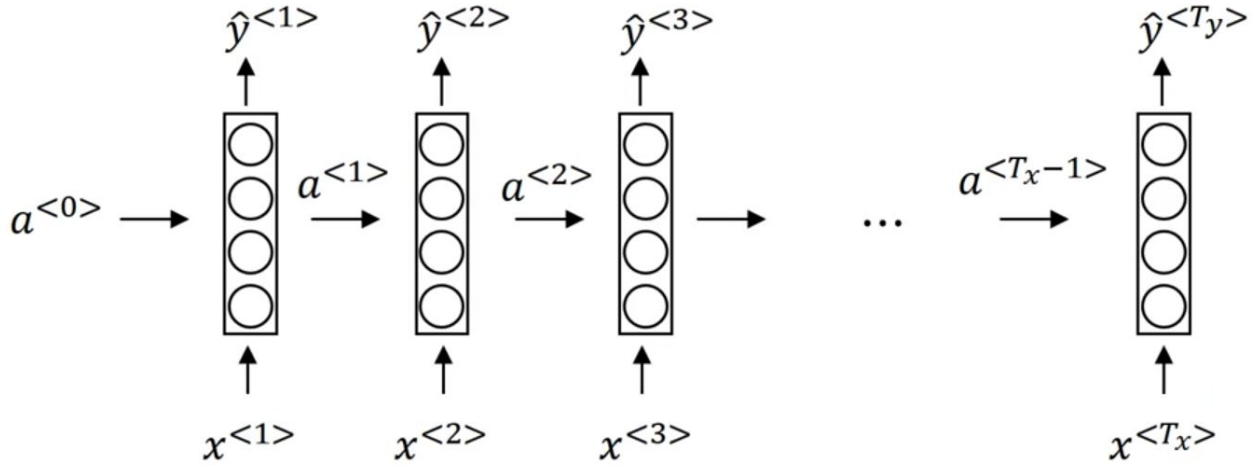


$$a^{<0>} = \vec{0}$$

$$a^{<1>} = g\left(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a\right)$$

$$\hat{y}^{<1>} = g\left(W_{ya}a^{<1>} + b_y\right)$$

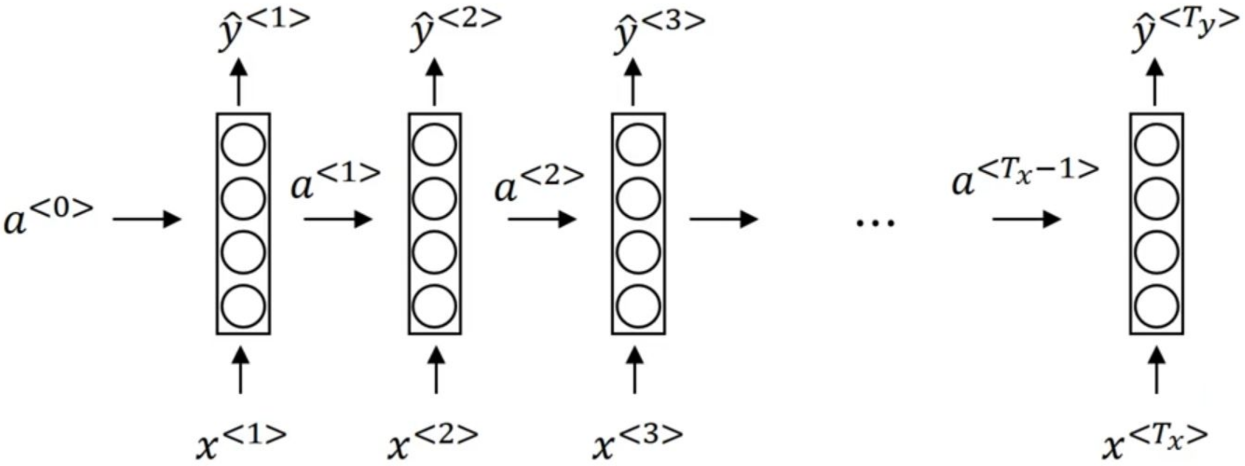
RNN - Diagram



$$a^{<t>} = g\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(W_{ya}a^{<t>} + b_y\right)$$

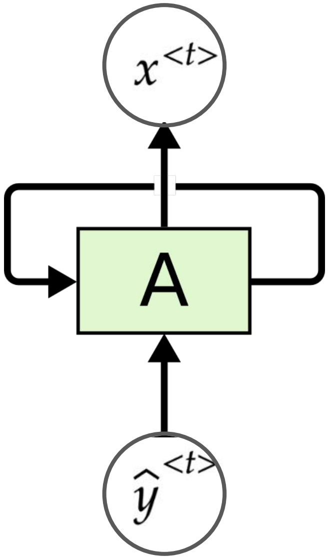
RNN - Diagram



$$a^{<t>} = g\left(\underline{W_{aa}}a^{<t-1>} + \underline{W_{ax}}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(\underline{W_{ya}}a^{<t>} + b_y\right)$$

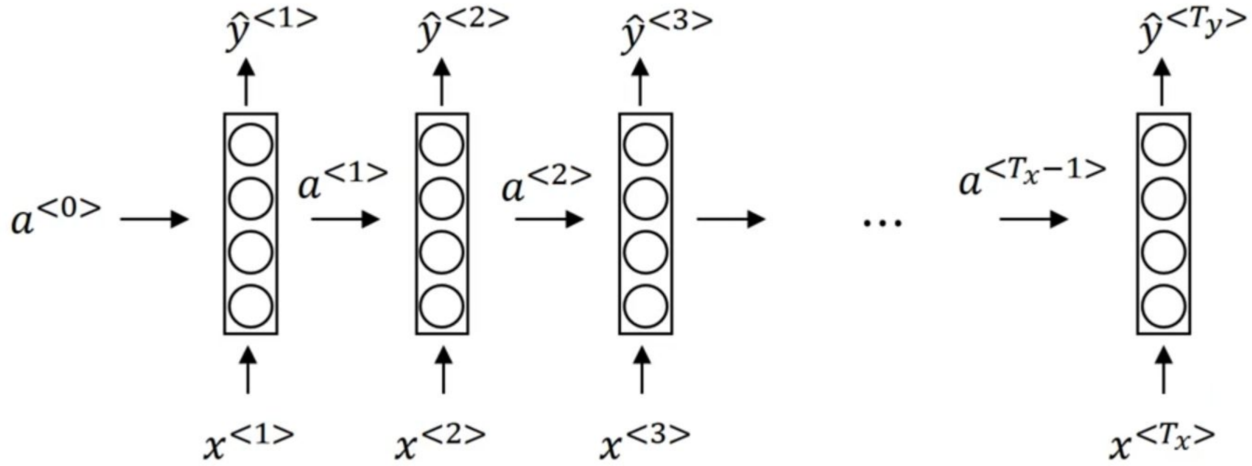
RNN - Diagram



$$a^{<t>} = g\left(\underline{W_{aa}}a^{<t-1>} + \underline{W_{ax}}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(\underline{W_{ya}}a^{<t>} + b_y\right)$$

RNN - Diagram

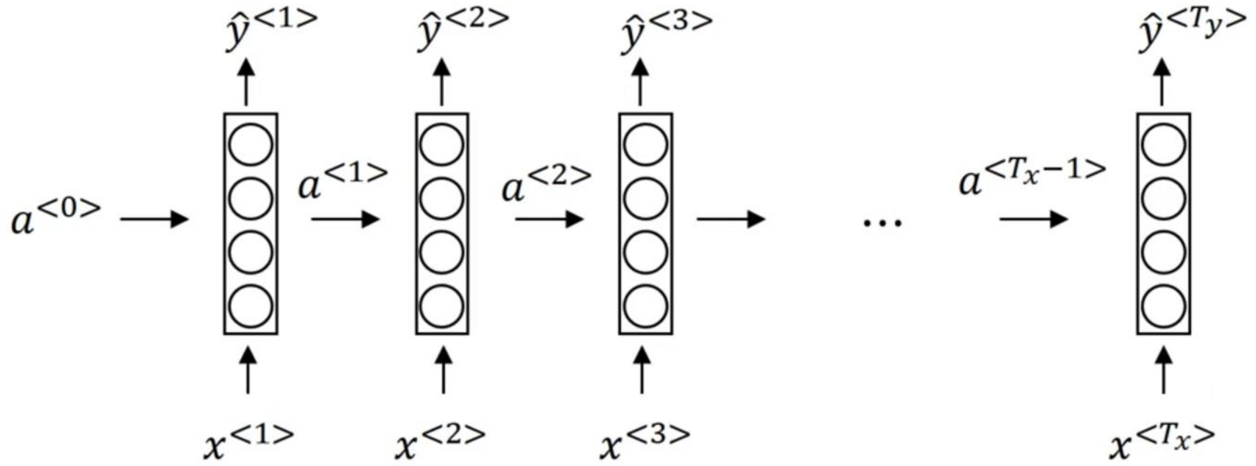


$$a^{<t>} = g\left(\underline{W_{aa}}a^{<t-1>} + \underline{W_{ax}}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(\underline{W_{ya}}a^{<t>} + b_y\right)$$

RNN - Diagram

$$Loss\left(\hat{y}^{<T_y>},\ y^{<T_y>}\right)$$

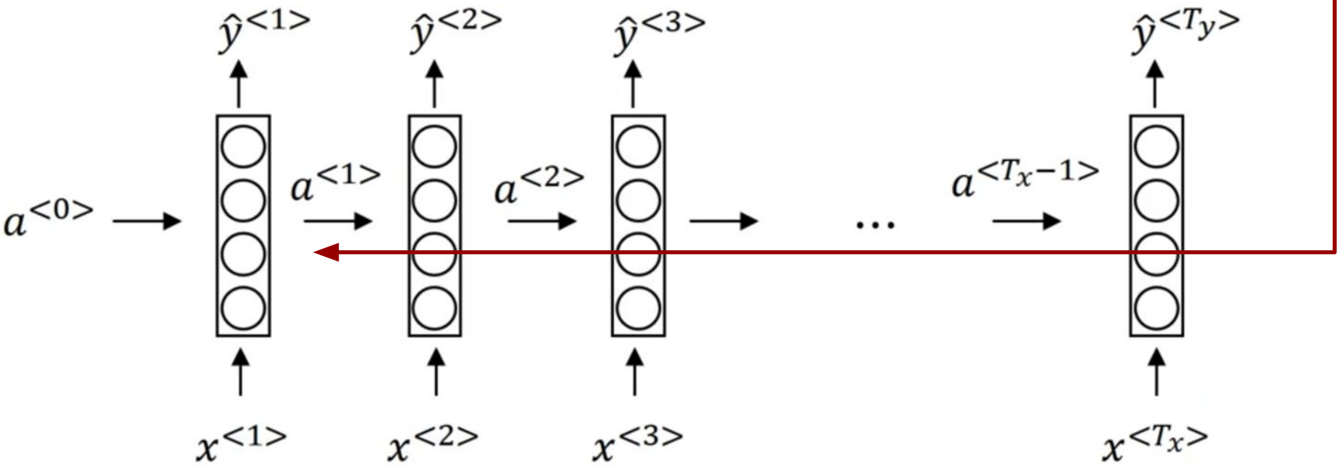


$$a^{<t>} = g\left(\underline{W_{aa}}a^{<t-1>} + \underline{W_{ax}}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(\underline{W_{ya}}a^{<t>} + b_y\right)$$

RNN - Backpropagation Through Time

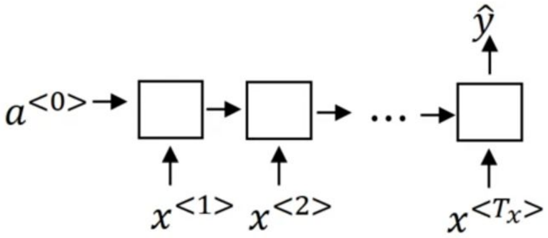
$$Loss\left(\hat{y}^{<T_y>},\ y^{<T_y>}\right)$$



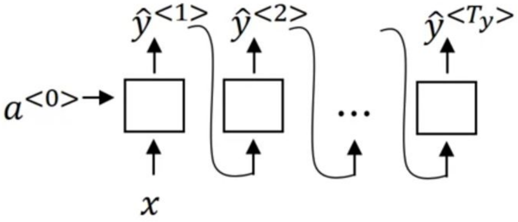
$$a^{<t>} = g\left(\underline{W_{aa}}a^{<t-1>} + \underline{W_{ax}}x^{<t>} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(\underline{W_{ya}}a^{<t>} + b_y\right)$$

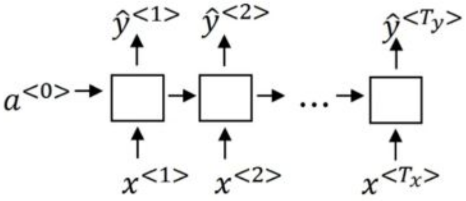
RNN Types & Variations



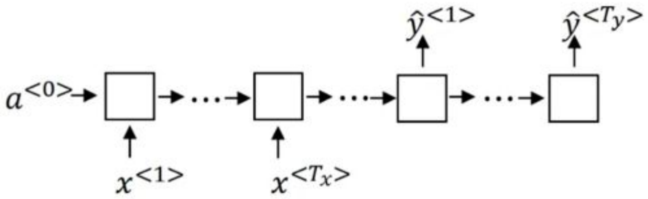
Many to one



One to many

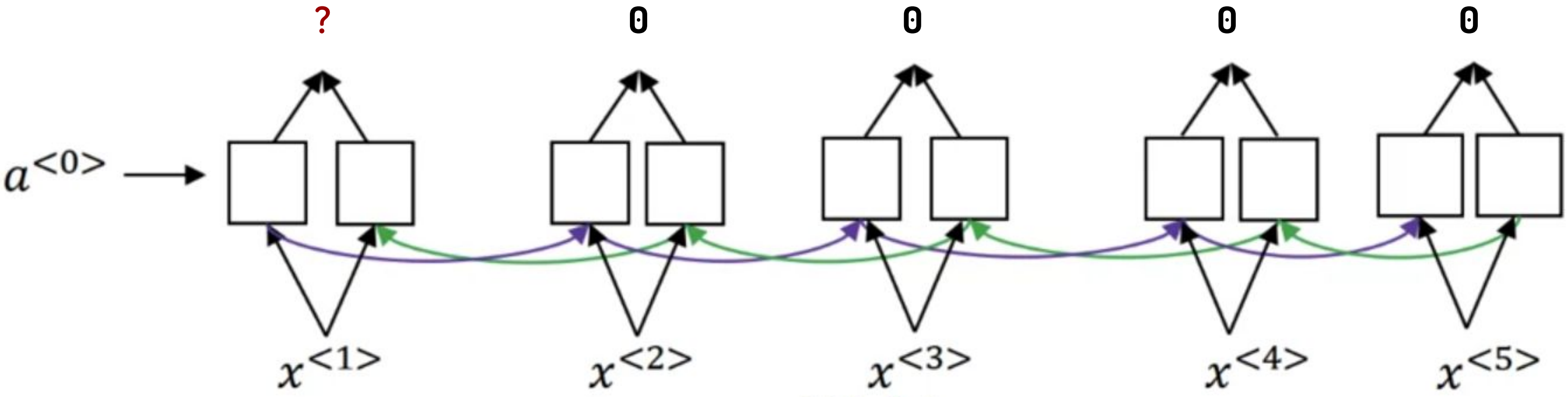


Many to many



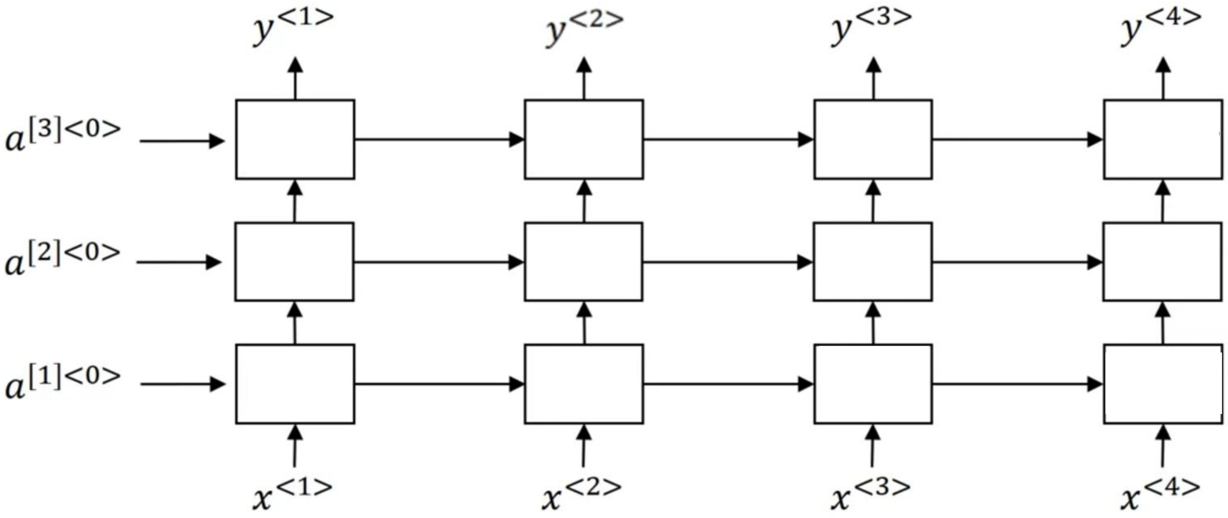
Many to many

RNN Types & Variations – Bidirectional

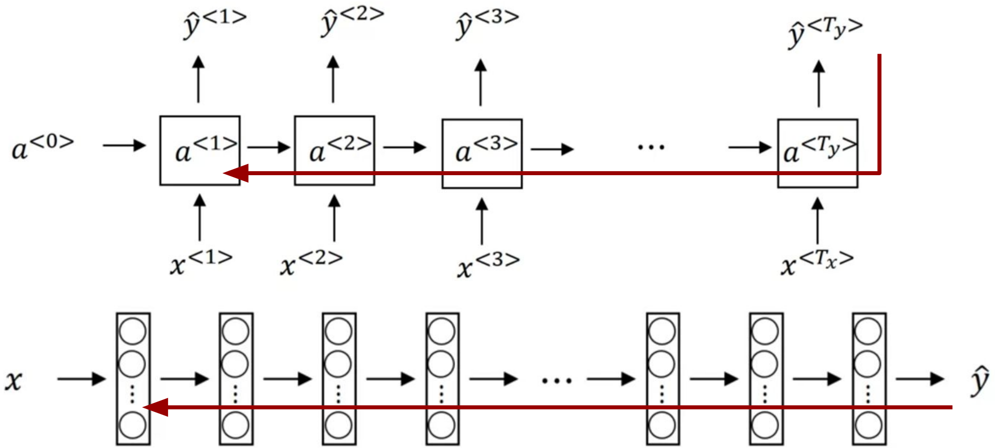


	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	$x^{<5>}$
1.	Teddy	bears	are	on	sale
2.	Teddy	Roosevelt	was	a	president

RNN Types & Variations – Deep



RNN Problems



RNN Problems - Vanishing and Exploding Gradients

- **0.9 ** 100 = 2.6561e-05**
- **0.9 ** 300 = 1.8739e-14**
- Gradient Clipping - solution to explosion

$$a^{<t>} = g\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right)$$
$$\hat{y}^{<t>} = g\left(W_{ya}a^{<t>} + b_y\right)$$

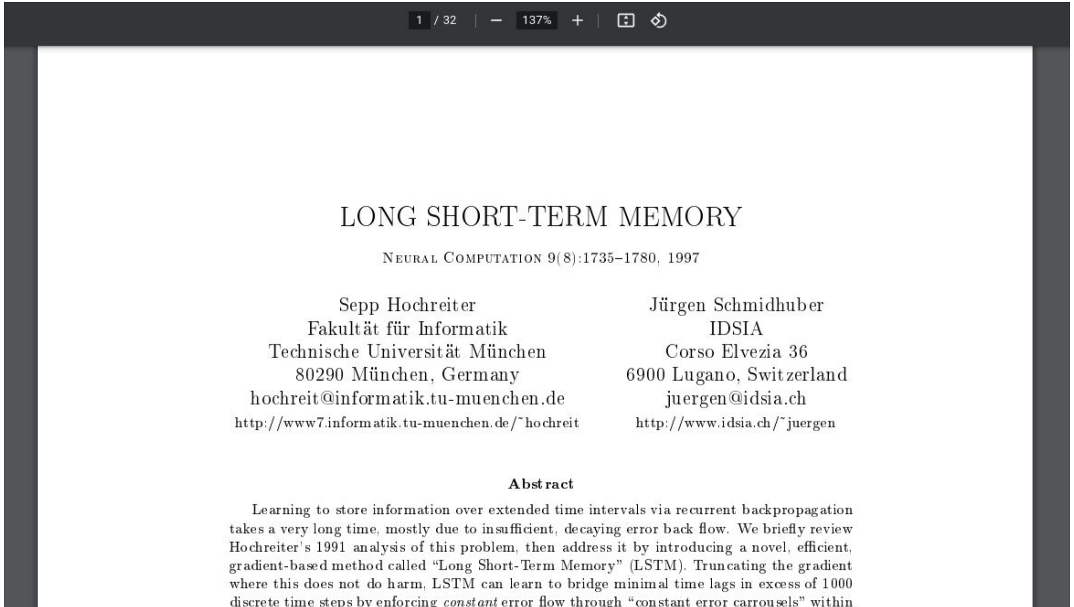
LSTM

Long **S**hort-**T**erm **M**emory

LSTM

- A kind of RNN
- Addressing vanishing gradients

Source: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory


1 / 32 – 137% +

LONG SHORT-TERM MEMORY

NEURAL COMPUTATION 9(8):1735–1780, 1997

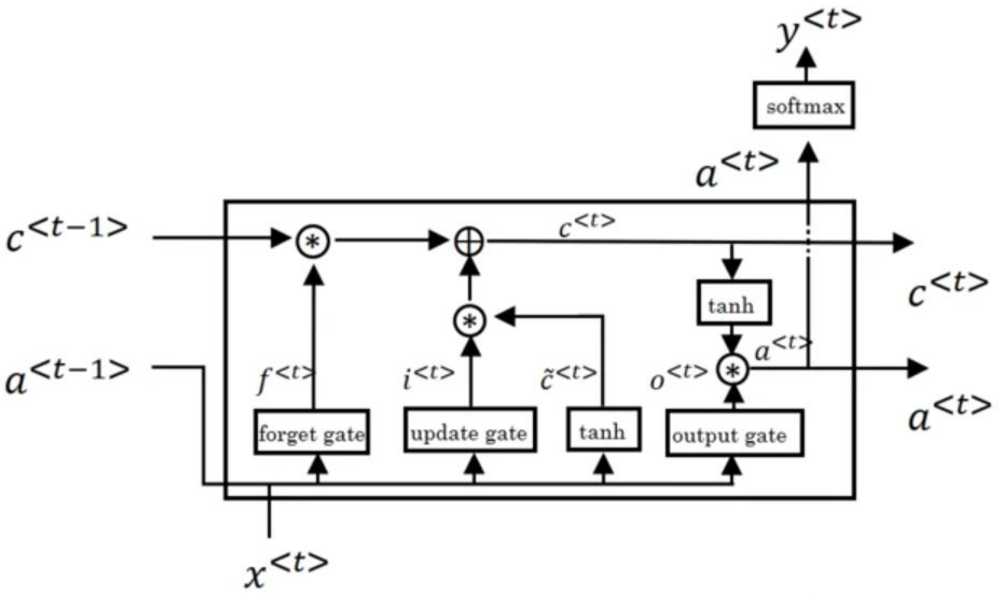
Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

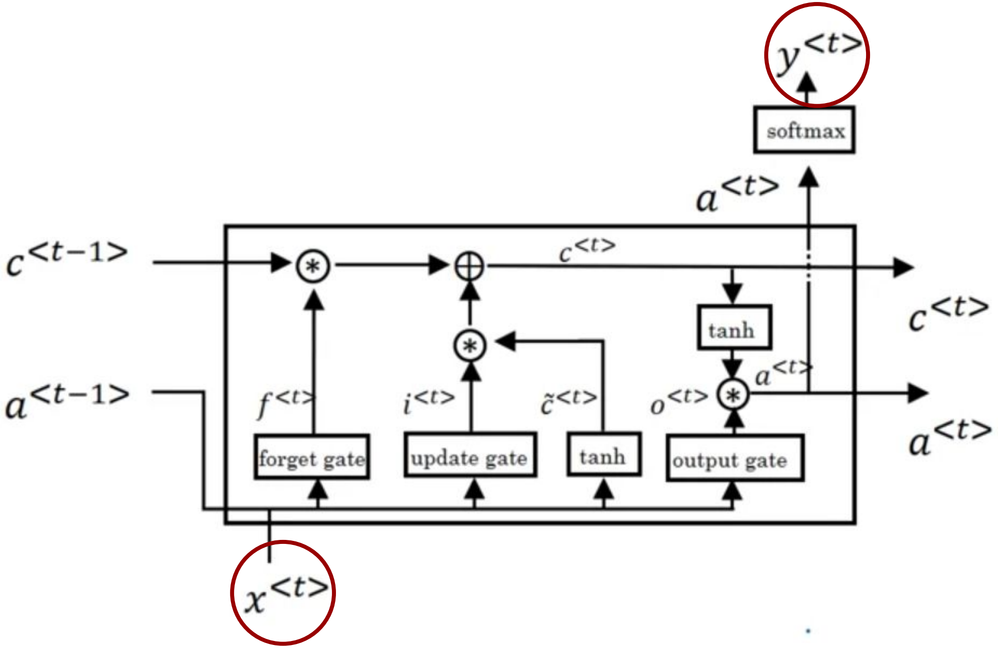
Abstract

Learning to store information over extended time intervals via recurrent backpropagation takes a very long time, mostly due to insufficient, decaying error back flow. We briefly review Hochreiter's 1991 analysis of this problem, then address it by introducing a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM). 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

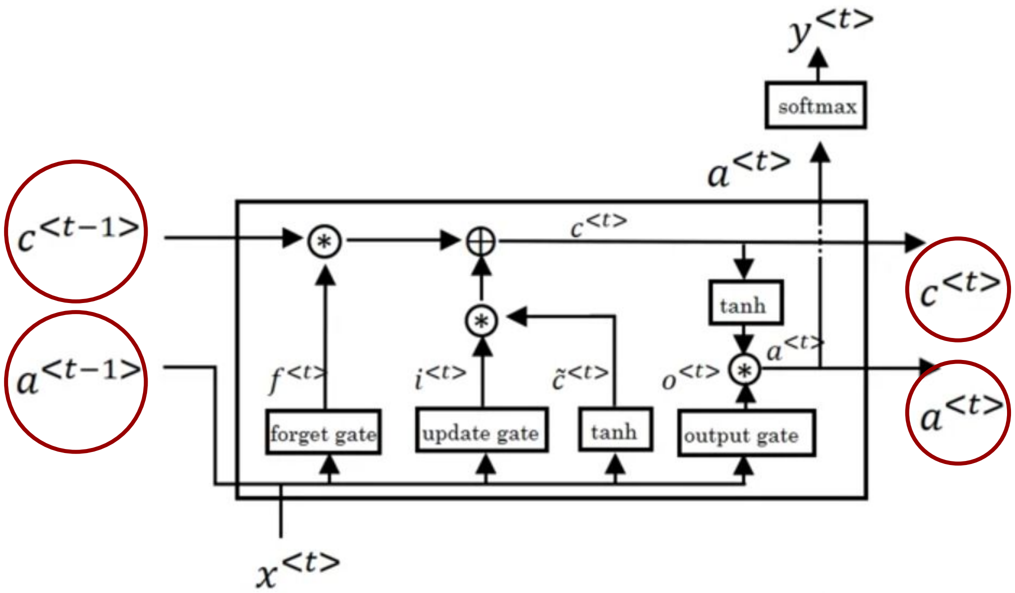
LSTM - Diagram



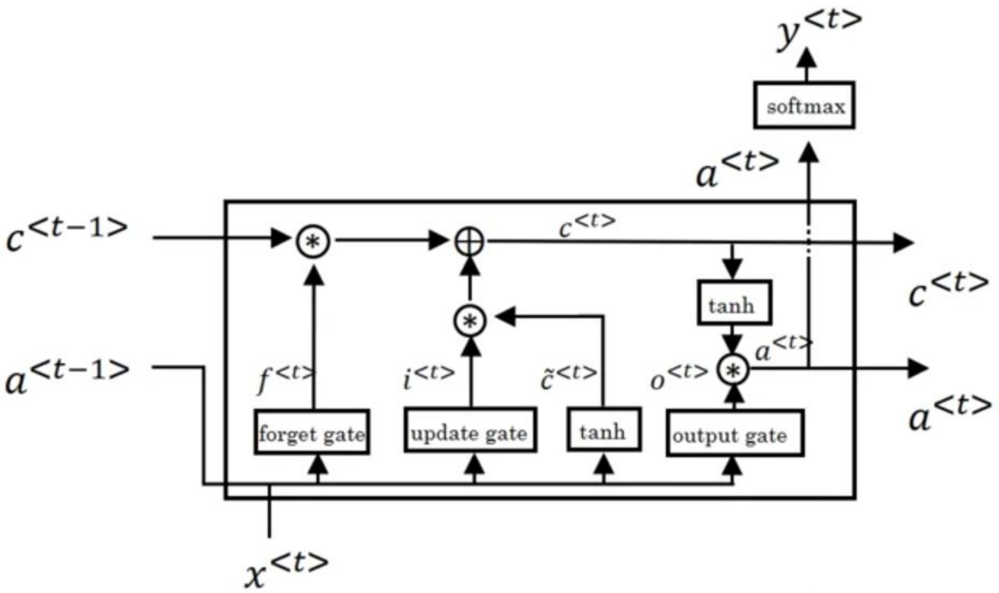
LSTM - Diagram



LSTM - Diagram

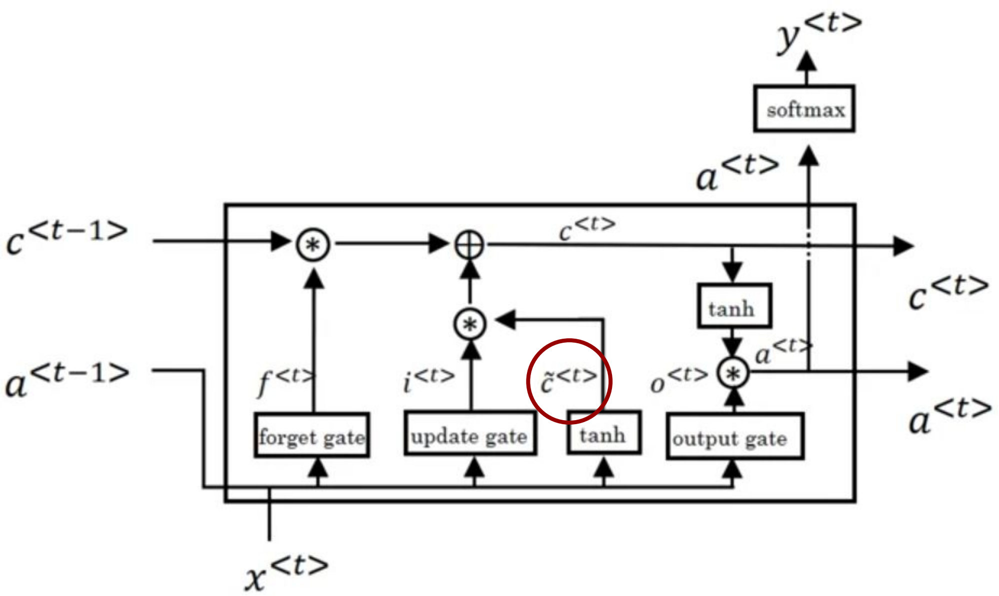


LSTM - Diagram



LSTM - Diagram

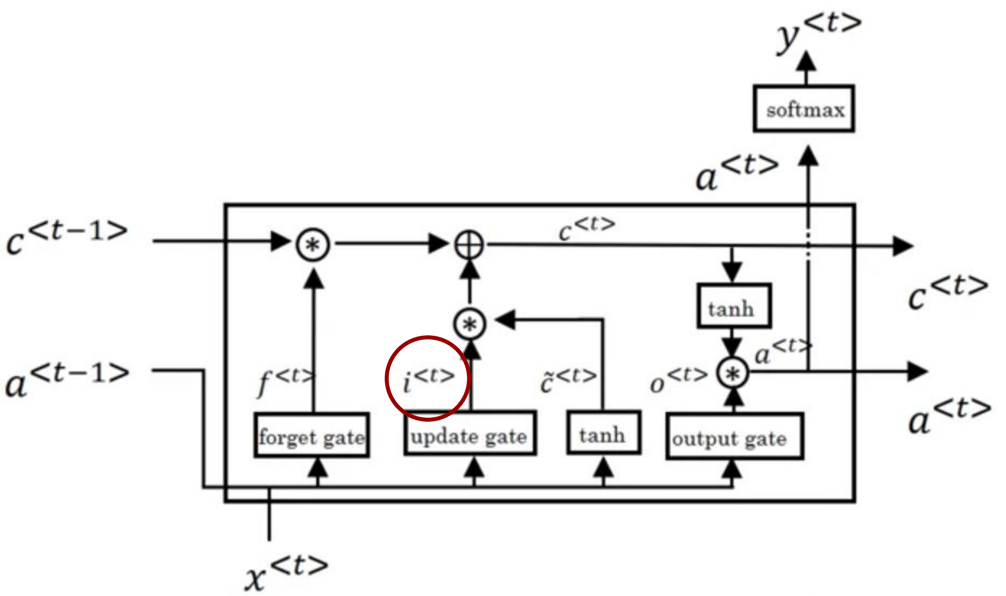
$$\tilde{c}^{<t>} = tanh(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c)$$



LSTM - Diagram

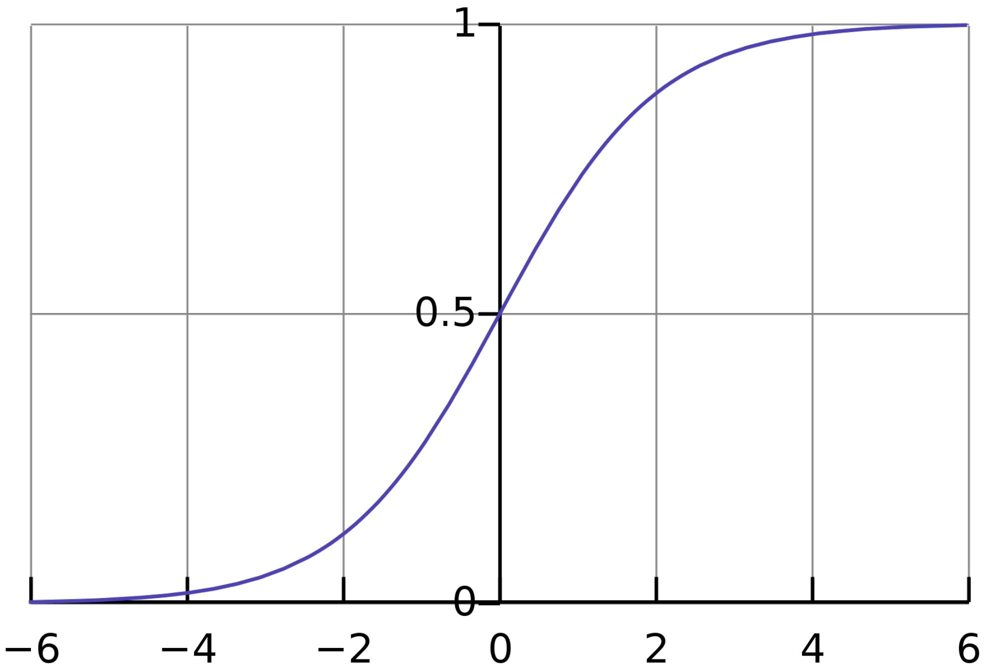
$$\tilde{c}^{<t>} = tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c\right)$$

$$\Gamma_u = \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u\right)$$



Sigmoid

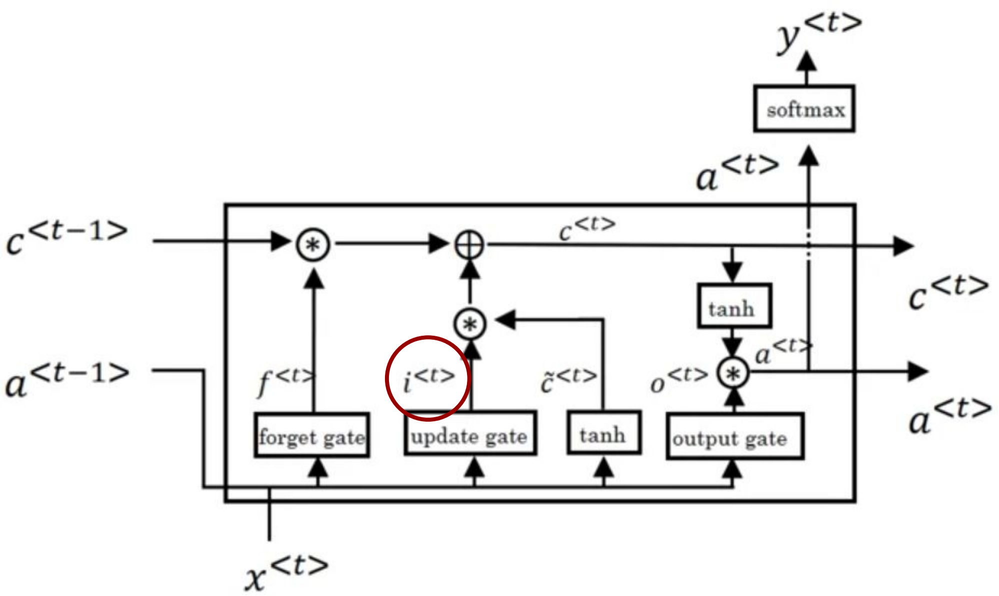
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Source:
https://en.wikipedia.org/wiki/Sigmoid_function

LSTM - Diagram

$$\tilde{c}^{<t>} = tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c\right)$$
$$\Gamma_u = \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u\right)$$

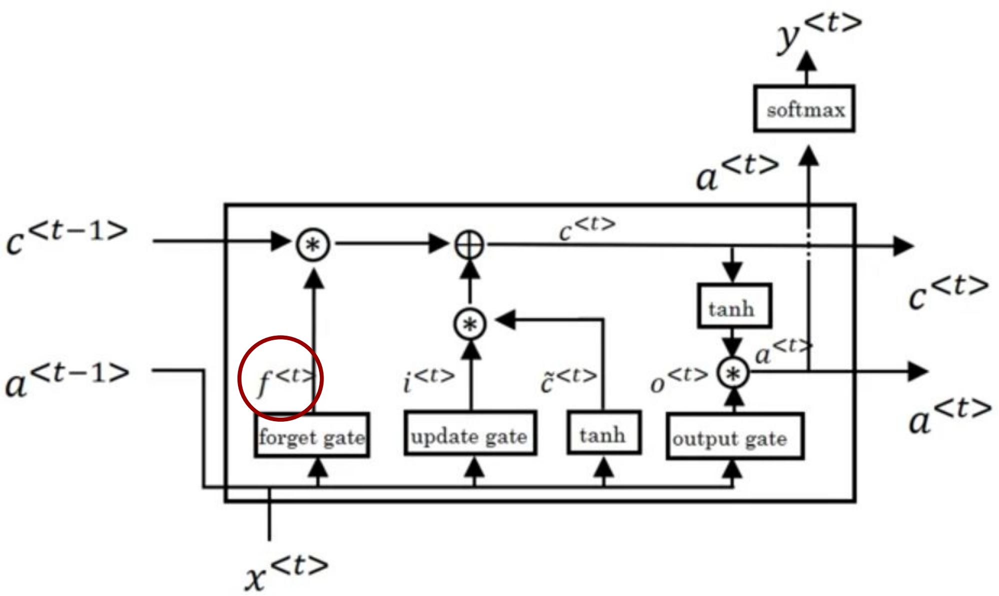


LSTM - Diagram

$$\tilde{c}^{<t>} = tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c\right)$$

$$\Gamma_u = \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u\right)$$

$$\Gamma_f = \sigma\left(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + b_f\right)$$



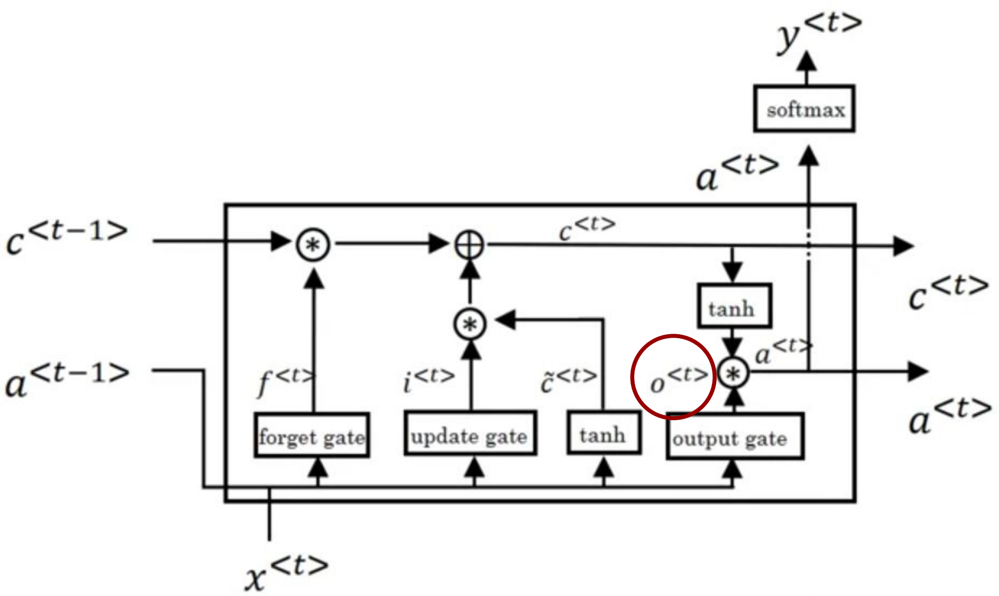
LSTM - Diagram

$$\tilde{c}^{<t>} = tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c\right)$$

$$\Gamma_u = \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u\right)$$

$$\Gamma_f = \sigma\left(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + b_f\right)$$

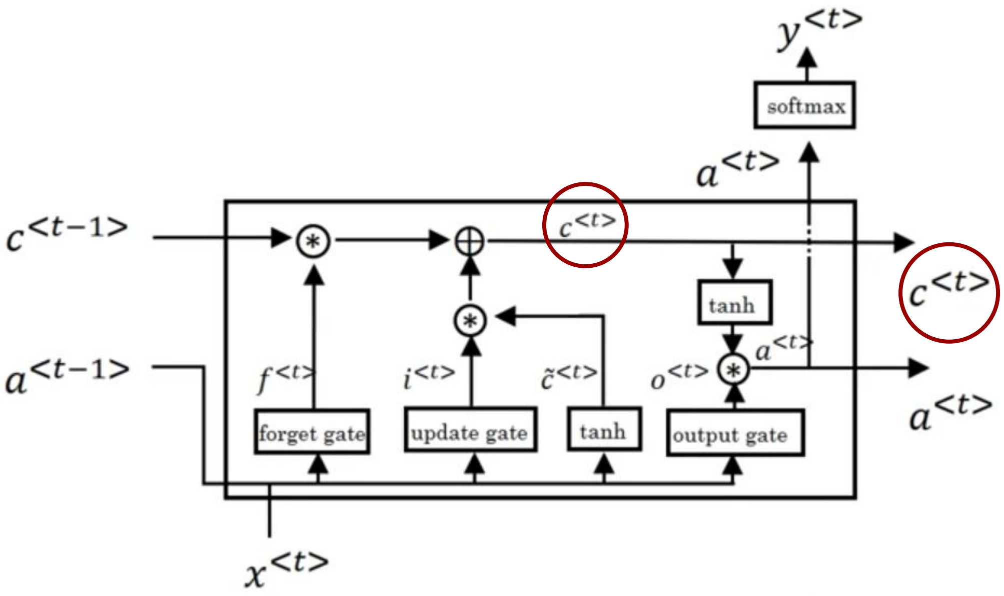
$$\Gamma_o = \sigma\left(W_{oa}a^{<t-1>} + W_{ox}x^{<t>} + b_o\right)$$



LSTM - Diagram

$$\tilde{c}^{<t>} = tanh(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c)$$
$$\Gamma_u = \sigma(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u)$$
$$\Gamma_f = \sigma(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + b_f)$$
$$\Gamma_o = \sigma(W_{oa}a^{<t-1>} + W_{ox}x^{<t>} + b_o)$$

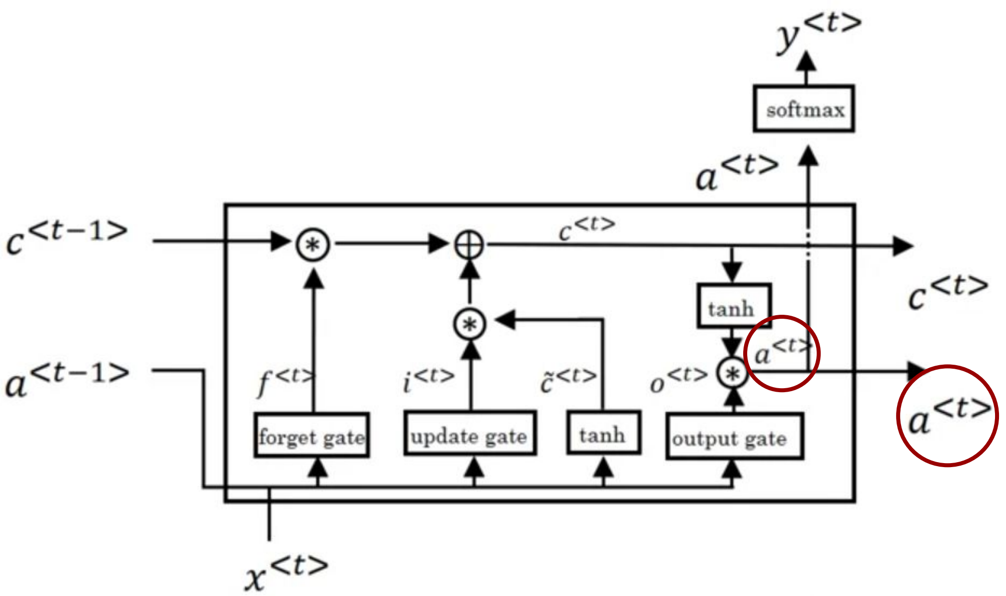
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$



LSTM - Diagram

$$\tilde{c}^{<t>} = \tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c\right)$$
$$\Gamma_u = \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u\right)$$
$$\Gamma_f = \sigma\left(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + b_f\right)$$
$$\Gamma_o = \sigma\left(W_{oa}a^{<t-1>} + W_{ox}x^{<t>} + b_o\right)$$

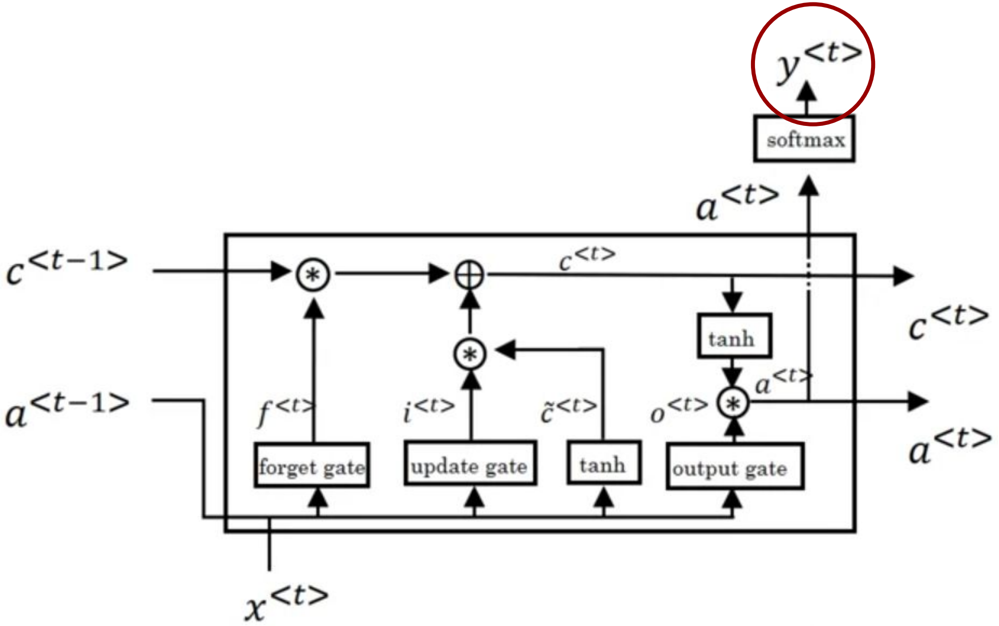
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$



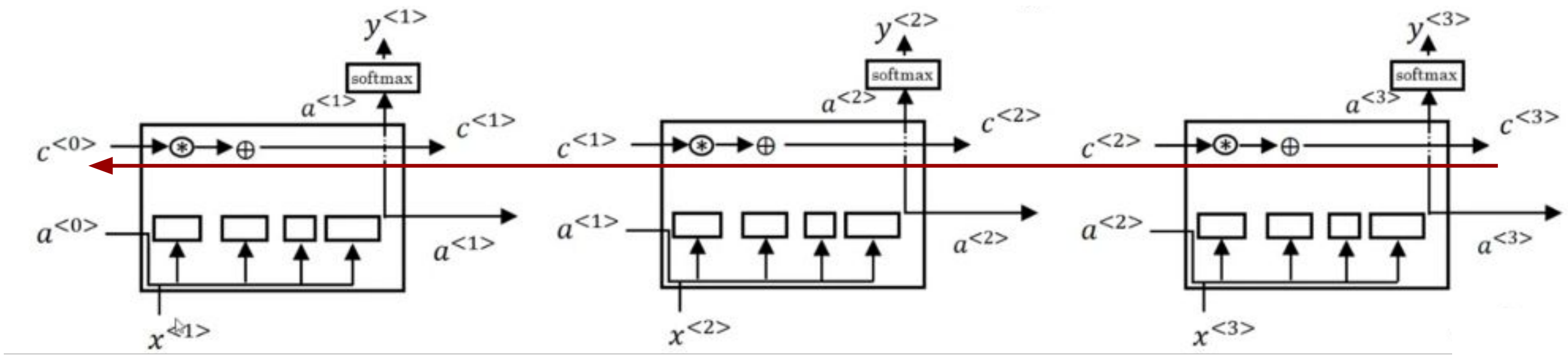
LSTM - Diagram

$$\tilde{c}^{<t>} = tanh\left(W_{ca}a^{<t-1>} + W_{cx}x^{<t>} + b_c\right)$$
$$\Gamma_u = \sigma\left(W_{ua}a^{<t-1>} + W_{ux}x^{<t>} + b_u\right)$$
$$\Gamma_f = \sigma\left(W_{fa}a^{<t-1>} + W_{fx}x^{<t>} + b_f\right)$$
$$\Gamma_o = \sigma\left(W_{oa}a^{<t-1>} + W_{ox}x^{<t>} + b_o\right)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * tanh\, c^{<t>}$$



LSTM - Diagram



LSTM - Problems

- Difficult to train
  - Very long gradient paths
- Transfer learning doesn't really work
  - New dataset for every task
- Slow
  - Parallelization is impossible

Transformer

- Proposed in 2017 addressing machine translation
- Based on attention mechanism
- Parallelization is now possible
- **Transfer learning works**

👉 https://jalammar.github.io/illustrated-transformer/

Source:
https://transformers.fandom.com/pl/wiki/Bumblebee

Kahoot time!

Thank you & Q&A