Lecture 9. Recurrent Neural Networks

Alex Avdyushenko

Kazakh-British Technical University

November 5, 2022

Five-minutes block

Five-minutes block

- Write down several names of neural network optimization methods
- Describe a couple of regularization methods for training neural networks
- Draw (or write down) how the skip-connection block works

Disadvantages of Convolutional Neural Networks

Or why we need recurrent ones :)

Disadvantages of Convolutional Neural Networks

Or why we need recurrent ones :)

- the input is only fixed-dimensional vectors (e.g. 28×28 images)

Disadvantages of Convolutional Neural Networks

Or why we need recurrent ones :)

- the input is only fixed-dimensional vectors (e.g. 28×28 images)
- the output is also a fixed dimension (for example, probabilities of 1000 classes)

Disadvantages of Convolutional Neural Networks

Or why we need recurrent ones :)

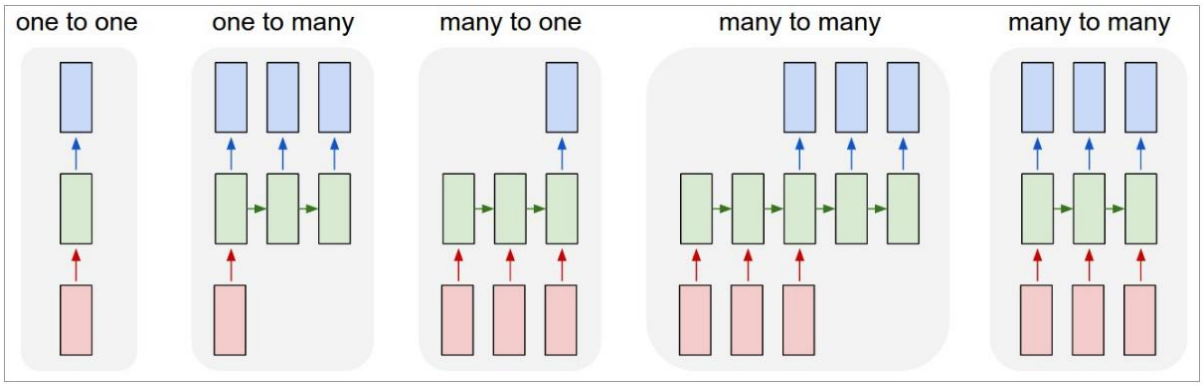
- the input is only fixed-dimensional vectors (e.g. 28×28 images)
- the output is also a fixed dimension (for example, probabilities of 1000 classes)
- fixed number of computational steps (i.e. network architecture)

Disadvantages of Convolutional Neural Networks

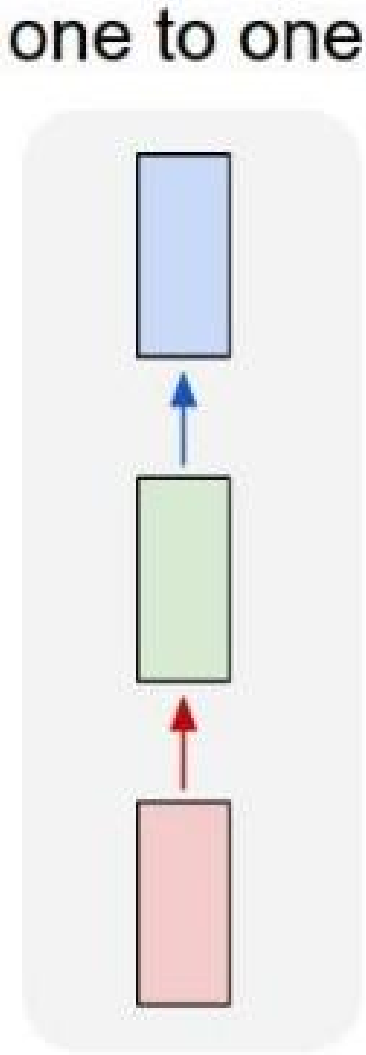
Or why we need recurrent ones :)

- the input is only fixed-dimensional vectors (e.g. 28×28 images)
- the output is also a fixed dimension (for example, probabilities of 1000 classes)
- fixed number of computational steps (i.e. network architecture)

A. Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks



Architectures of Recurrent Networks



Vanilla Neural Networks

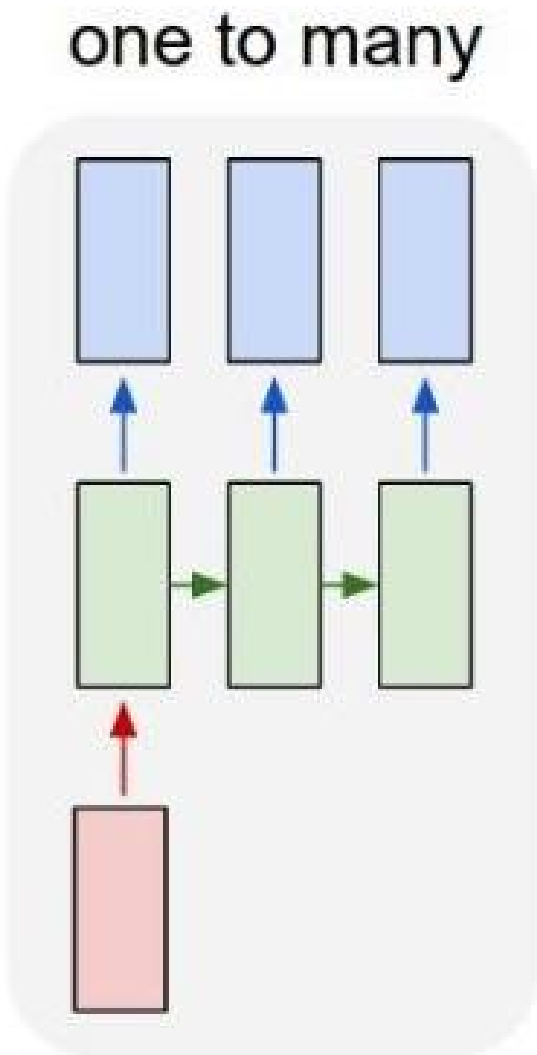
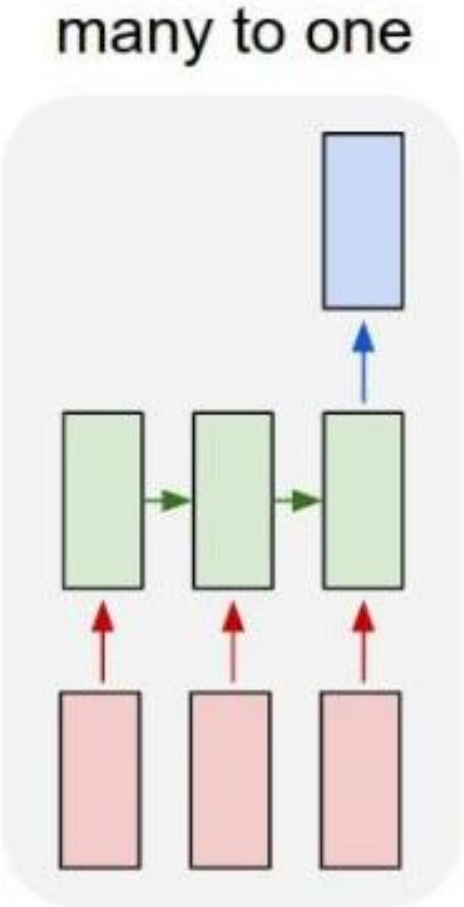
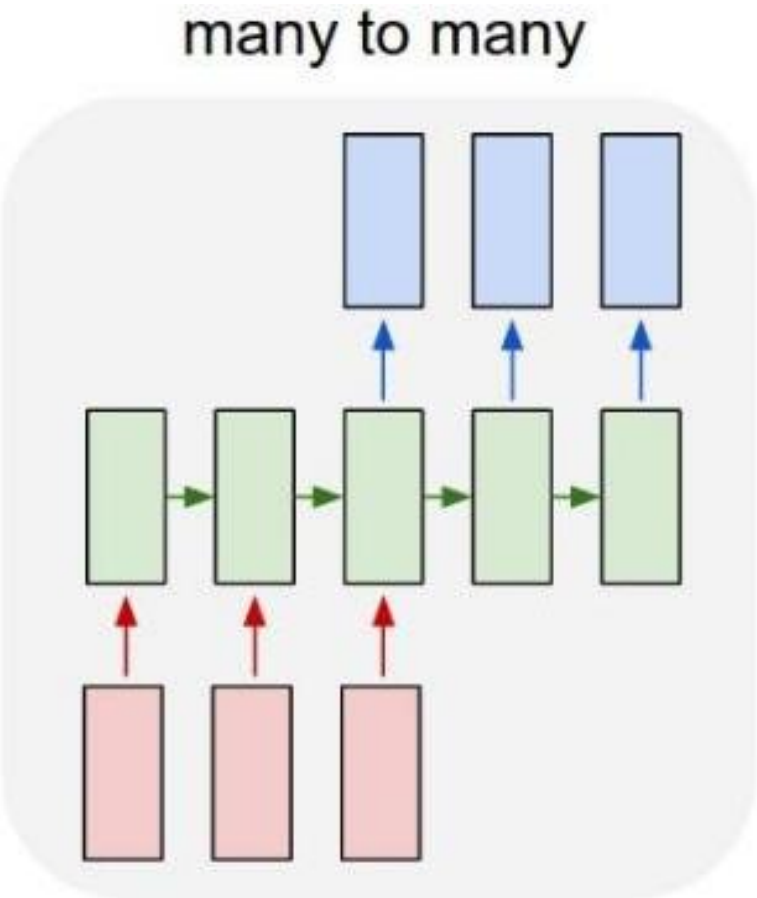


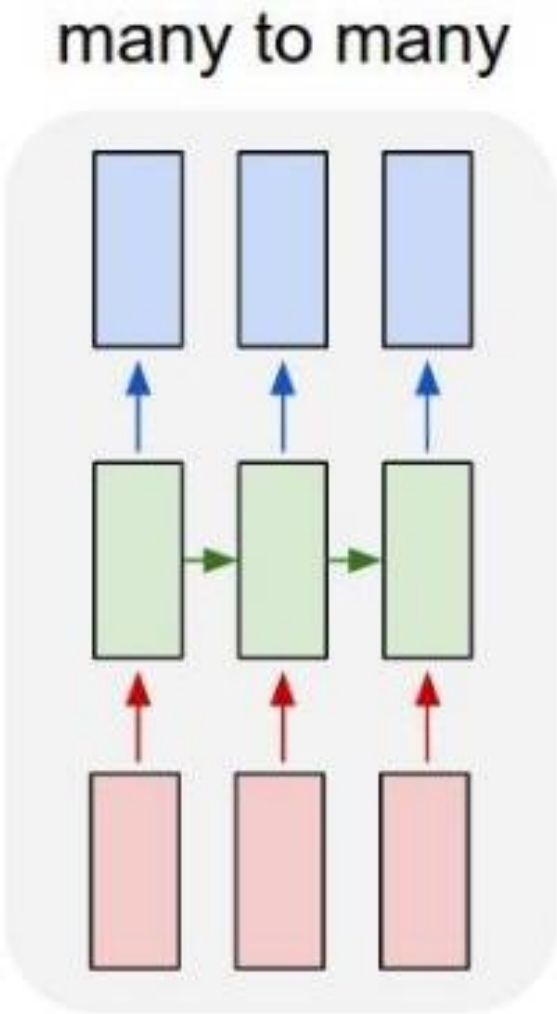
Image Captioning
image $\rightarrow$ (sequence of words)



Sentiment Classification
(sequence of words) $\rightarrow$ sentiment

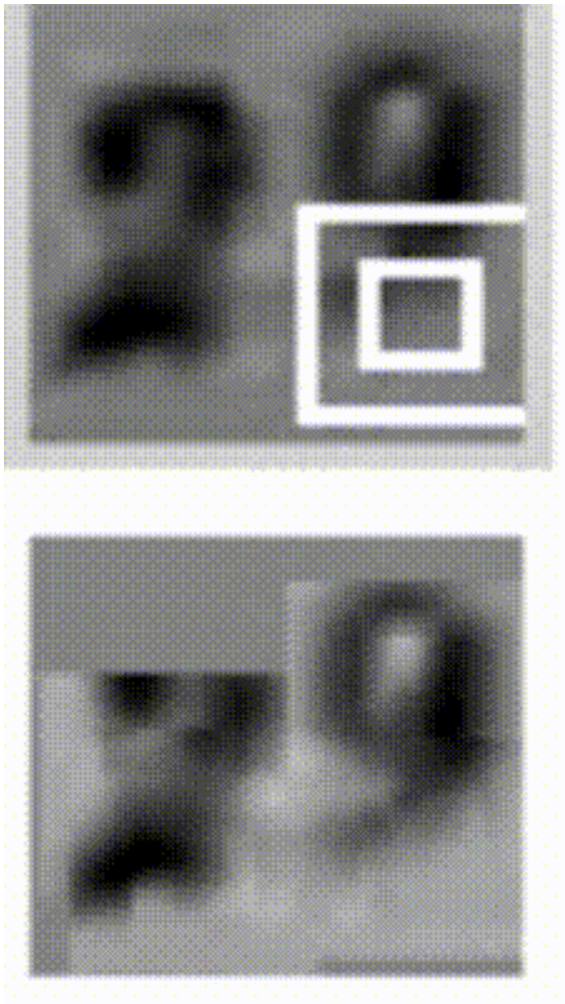


Machine Translation
(seq of words) $\rightarrow$ (seq of words)



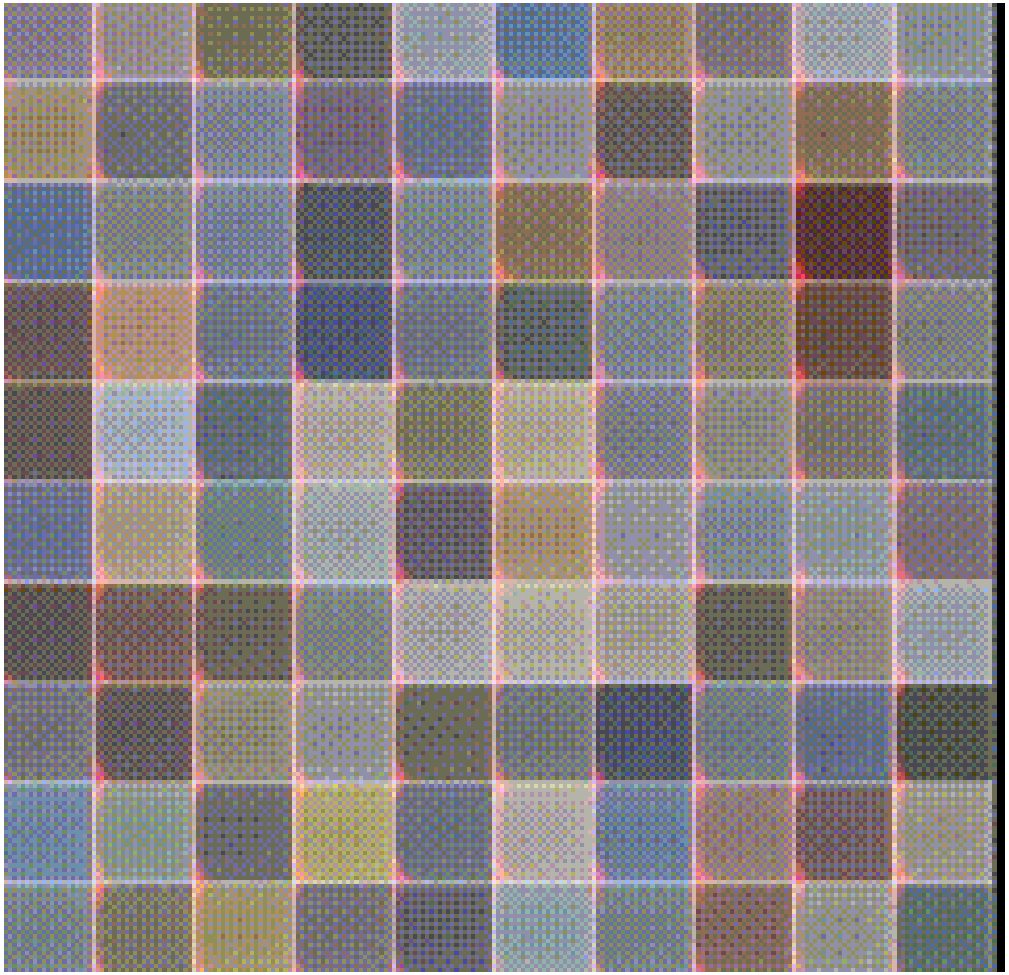
Video Classification
(on frame level)

Sequential Processing of Fixed Input



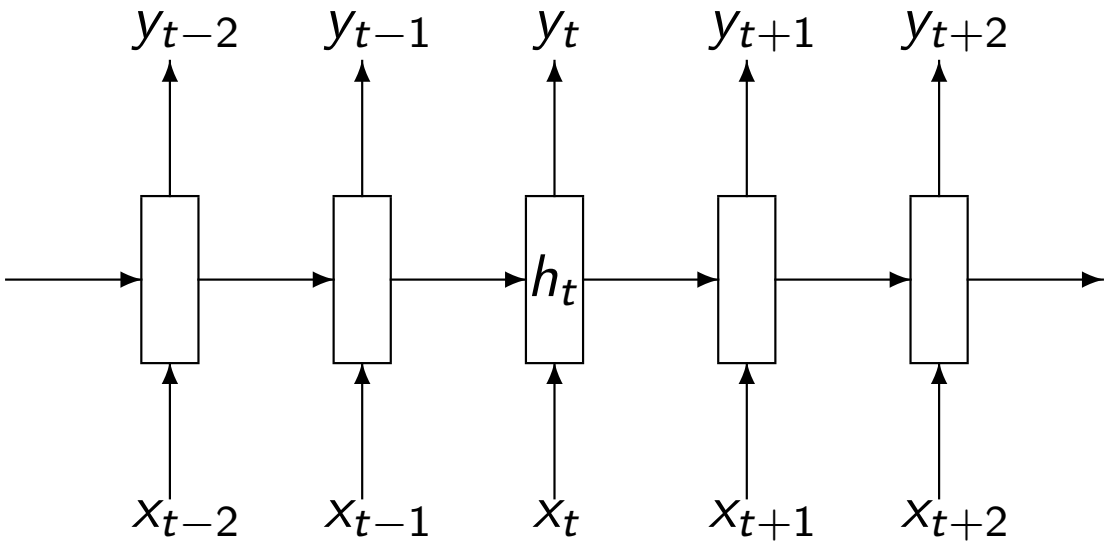
J. Ba, V. Mnih, K. Kavukcuoglu. Multiple Object Recognition with Visual Attention

Sequential Generation of Fixed Output



K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, D. Wierstra. DRAW: A Recurrent Neural Network For Image Generation

Recurrent Neural Network scheme



Recurrent Neural Network

We process the sequence of vectors x with **one and the same** function with parameters:

$$h_t = f_W(h_{t-1}, x_t)$$

f_W is a function parameterized by W
x_t — next input vector
h_t — hidden state

Recurrent Neural Network

We process the sequence of vectors x with **one and the same** function with parameters:

$$h_t = f_W(h_{t-1}, x_t)$$

f_W is a function parameterized by W
x_t — next input vector
h_t — hidden state

Question

What function can we take as f_W?

Vanilla Recurrent Neural Network

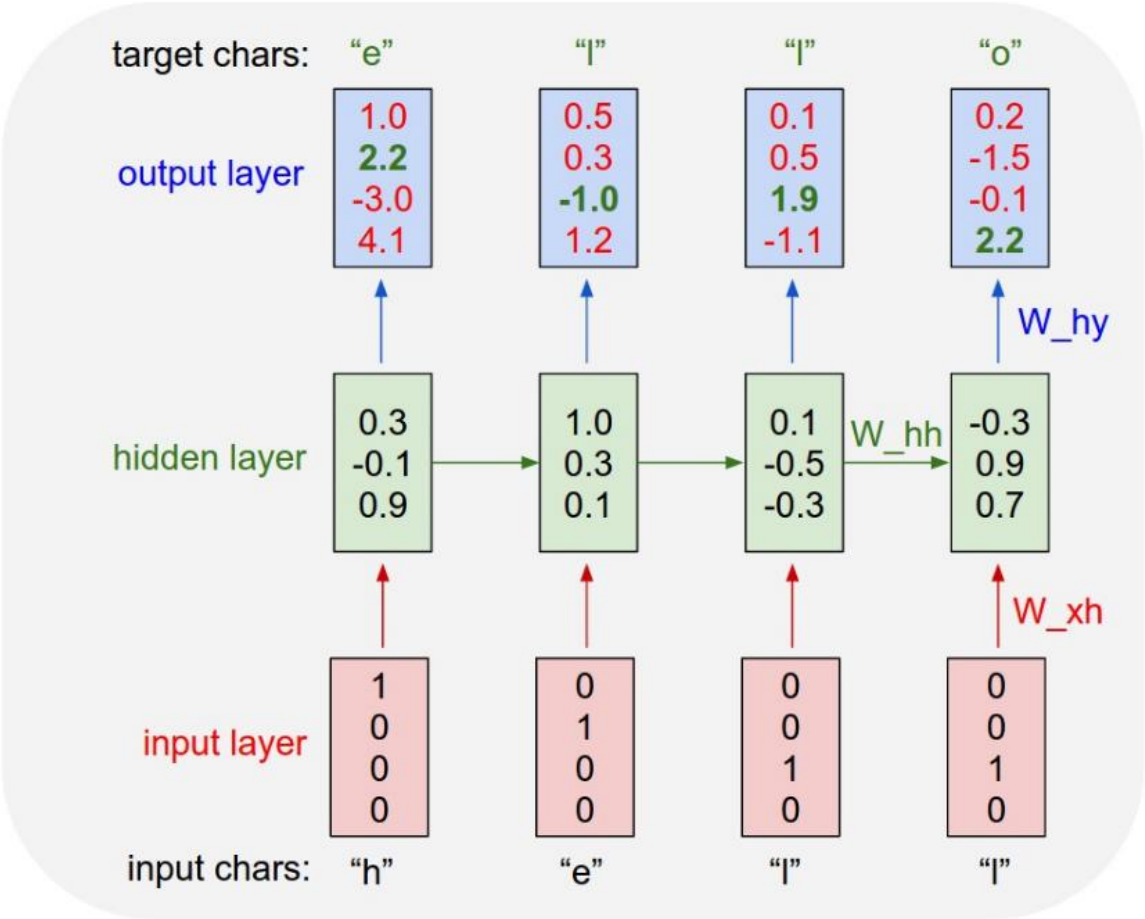
$$h_t = f_W(h_{t-1}, x_t)$$

As a function f_W we set a linear transformation with a non-linear component-wise "sigmoid":

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$
$$y_t = W_{hy} h_t$$

Character level model example

The entire four-letter dictionary: $[h, e, l, o]$ and word "hello" as train:



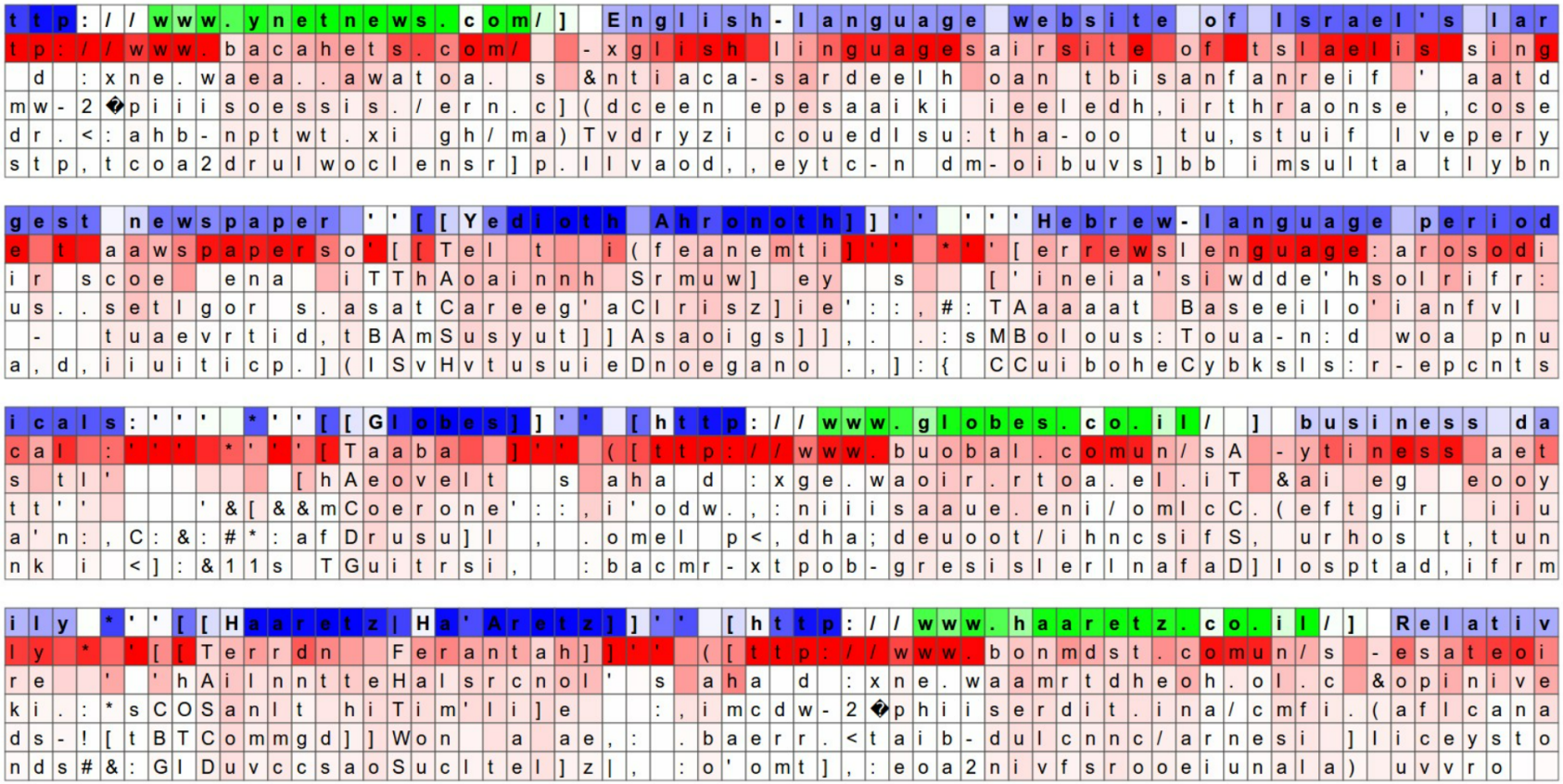
Softmax is also applied to the values of the output layer to obtain the loss function

Demo

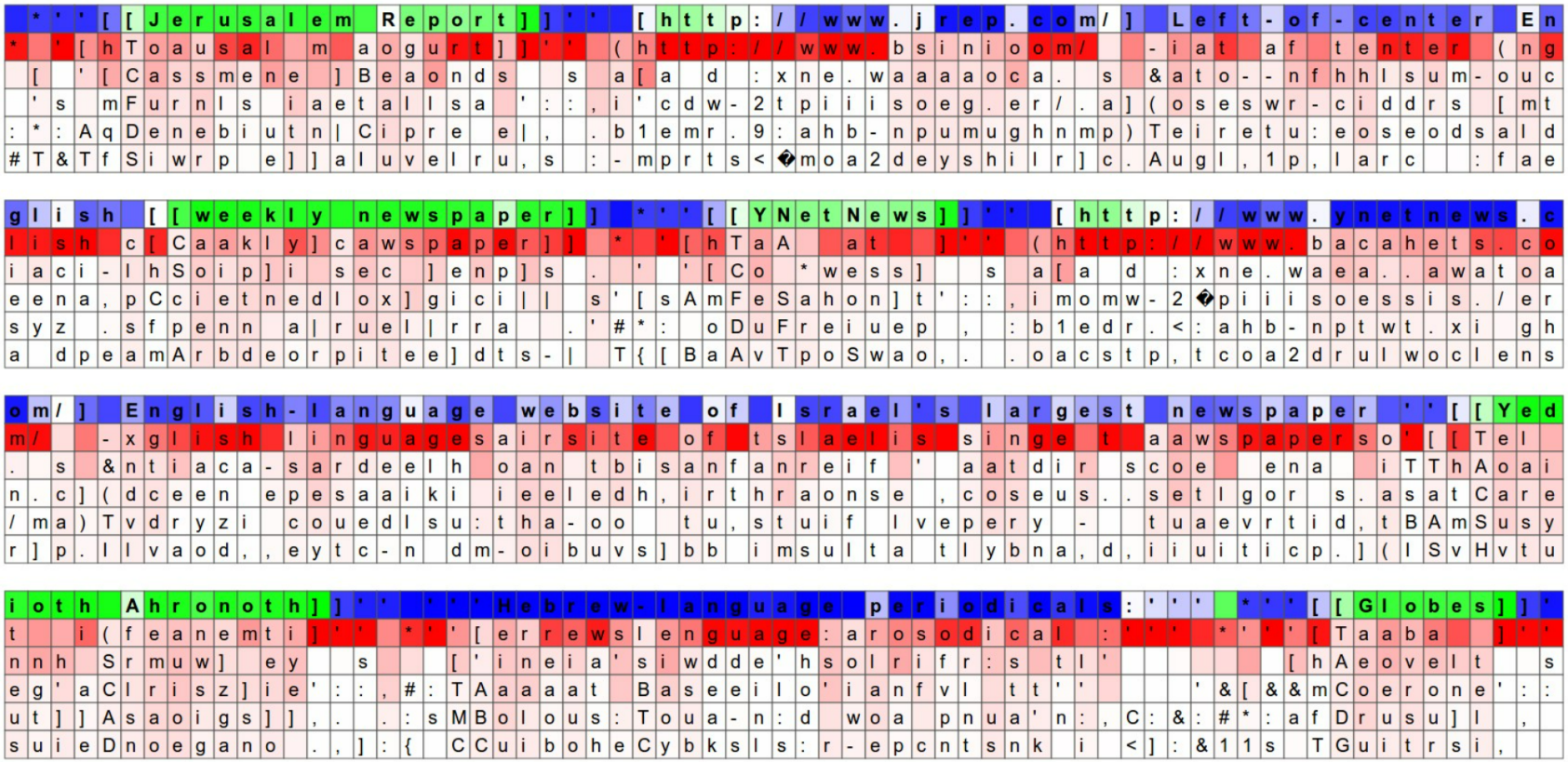
numpy implementation by Karpathy

Let's get to grips with the code in jupyter notebook!

How does it work?

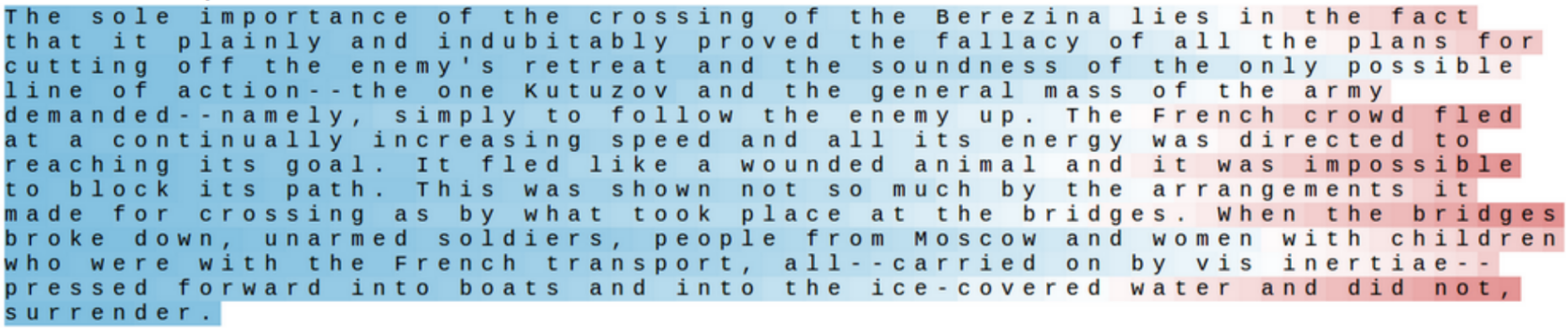


The neuron highlighted in this image seems to get very excited about URLs and turns off outside of the URLs. The LSTM is likely using this neuron to remember if it is inside a URL or not.

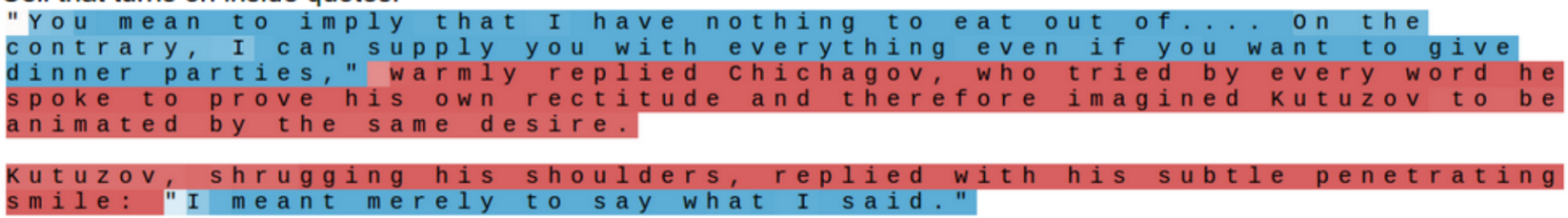


The highlighted neuron here gets very excited when the RNN is inside the [[ ]] markdown environment and turns off outside of it. Interestingly, the neuron can't turn on right after it sees the character "[", it must wait for the second "[" and then activate. This task of counting whether the model has seen one or two "[" is likely done with a different neuron.

Cell sensitive to position in line:



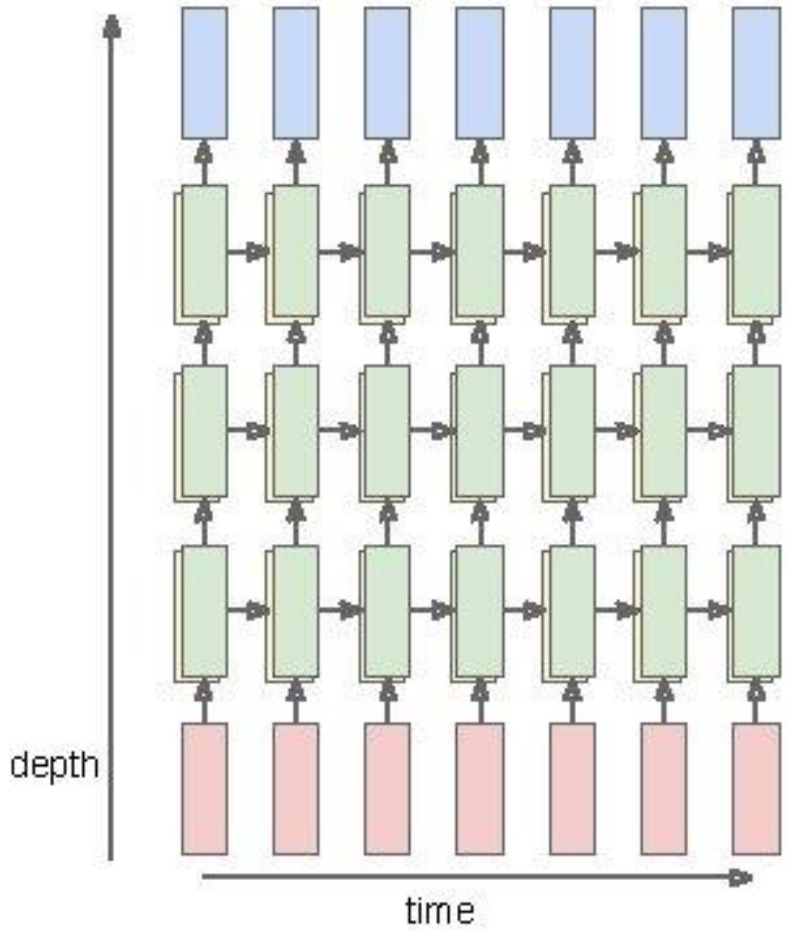
Cell that turns on inside quotes:



Deep recurrent networks

$$h_t^\ell = \tanh W^\ell \begin{pmatrix} h_t^{\ell-1} \\ h_{t-1}^\ell \end{pmatrix}$$

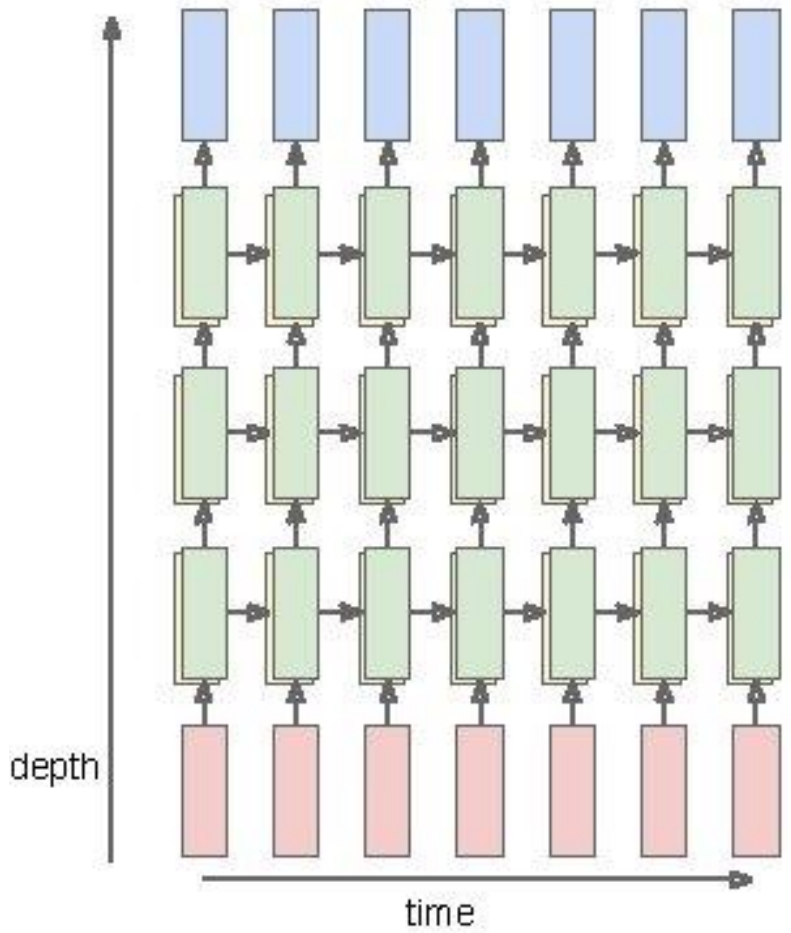
$$h \in \mathbb{R}^n, \qquad W^\ell[n \times 2n]$$



Deep recurrent networks

$$h_t^\ell = \tanh W^\ell \begin{pmatrix} h_t^{\ell-1} \\ h_{t-1}^\ell \end{pmatrix}$$

$$h \in \mathbb{R}^n, \qquad W^\ell[n \times 2n]$$



Question

What is the main problem with vanilla RNN?

Long short-term memory (LSTM)

$$W^\ell[4n \times 2n]$$

$$\begin{pmatrix} i \\ f \\ o \\ c'_t \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^\ell \begin{pmatrix} h_t^{\ell-1} \\ h_{t-1}^{\ell} \end{pmatrix}$$

$$c_t^\ell = f \odot c_{t-1}^\ell + i \odot c'_t$$
$$h_t^\ell = o \odot \tanh(c_t^\ell)$$

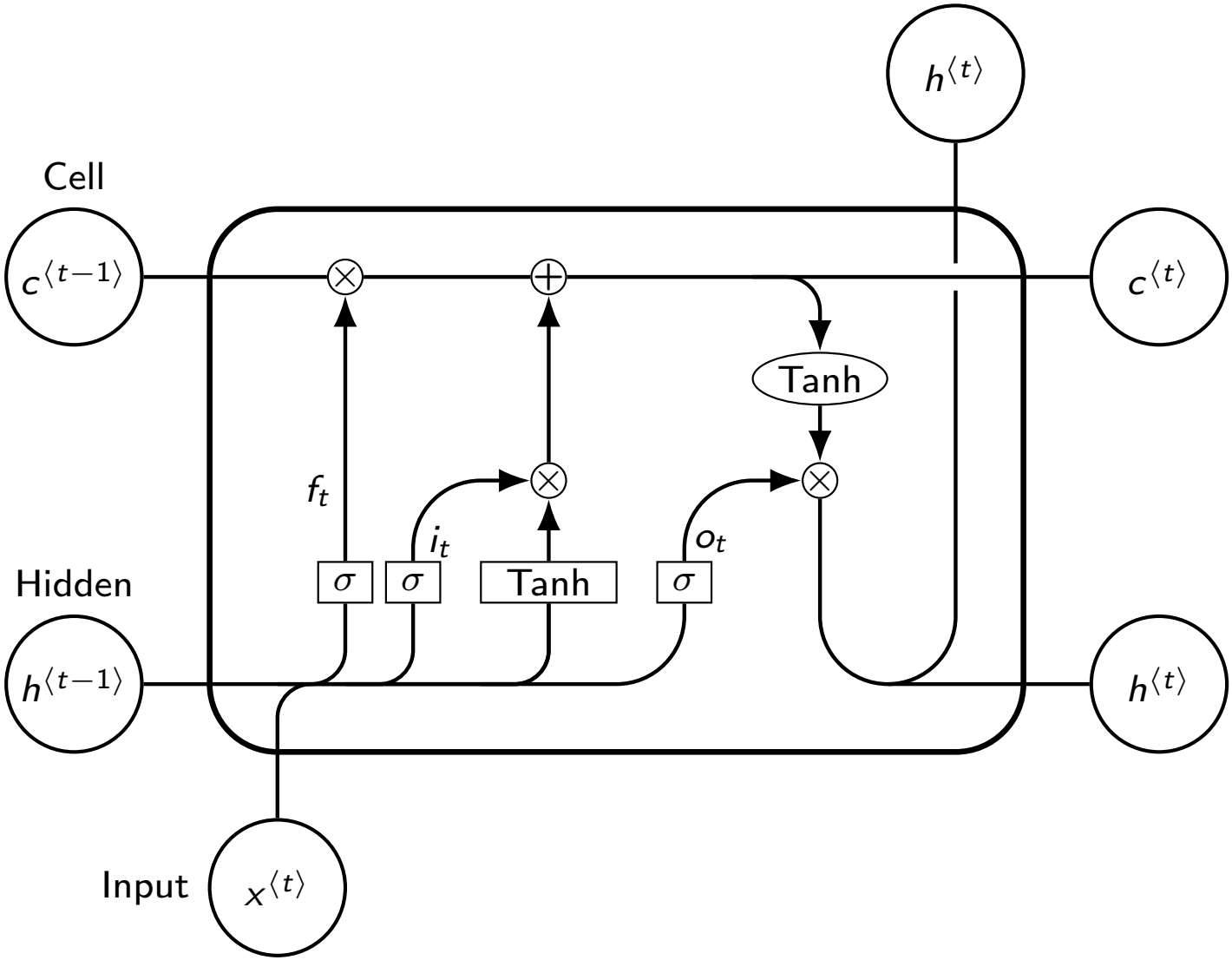
$\odot$ — component-wise product

LSTM: Motivation and Schema

The network should remember the context for a long time. Which context? Network learns itself. To do this, the vector c_t is introduced, which is the state vector of the network at the moment t.

$$
\begin{aligned}
c'_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c'}) && \text{candidate cell state} \\
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) && \text{input gate} \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) && \text{forget gate} \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) && \text{output gate}
\end{aligned}
$$

$$
\begin{aligned}
c_t &= f_t \odot c_{t-1} + i_t \odot c'_t && \text{cell state} \\
h_t &= o_t \odot \tanh(c_t) && \text{block output}
\end{aligned}
$$


$h^{\langle t \rangle}$
Cell
$c^{\langle t-1 \rangle}$
$c^{\langle t \rangle}$
Tanh
f_t
i_t
o_t
Hidden
σ
σ
Tanh
σ
$h^{\langle t-1 \rangle}$
$h^{\langle t \rangle}$
Input
$x^{\langle t \rangle}$

LSTM: forget gate

$$f_t = \sigma(W_{xf} x_t + W_{hf} h_{t-1} + b_f)$$

Example: Removing the Gender of an Actor When Generating Text

LSTM: input/candidate gates

$$c'_t = \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_{c'}) \quad \text{candidate cell state}$$
$$i_t = \sigma(W_{xi} x_t + W_{hi} h_{t-1} + b_i) \quad \text{input gate}$$

Example: Adding the Gender of an Actor When Generating Text

LSTM: updating the state of a memory cell

$$c_t = f_t \odot c_{t-1} + i_t \odot c'_t \qquad \text{(cell state)}$$

The new state c_t is obtained by the sum of the previous state c_{t-1} with the filter f_t and the vector of candidate values c'_t with the filter i_t

LSTM: output generation

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad \text{output gate}$$
$$h_t = o_t \odot \tanh(c_t) \quad \text{block output}$$

GRU: Gated Recurrent Unit

$$
\begin{aligned}
u_t &= \sigma(W_{xu}x_t + W_{hu}h_{t-1} + b_u) \\
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
h'_t &= \tanh(W_{xh'}x_t + W_{hh'}(r_t \odot h_{t-1})) \\
h_t &= (1 - u_t) \odot h'_t + u_t \odot h_{t-1}
\end{aligned}
$$

Only h_t is used, vector c_t is not introduced.
Update-gate instead of input and forget.
The reset gate determines how much memory to move forward from the previous step.

Disadvantages of Vanilla RNN

$$h_t = f_W(h_{t-1}, x_t)$$

As a function f_W we set a linear transformation with a non-linear component-wise "sigmoid":

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$
$$y_t = W_{hy} h_t$$

Disadvantages of Vanilla RNN

$$h_t = f_W(h_{t-1}, x_t)$$

As a function f_W we set a linear transformation with a non-linear component-wise "sigmoid":

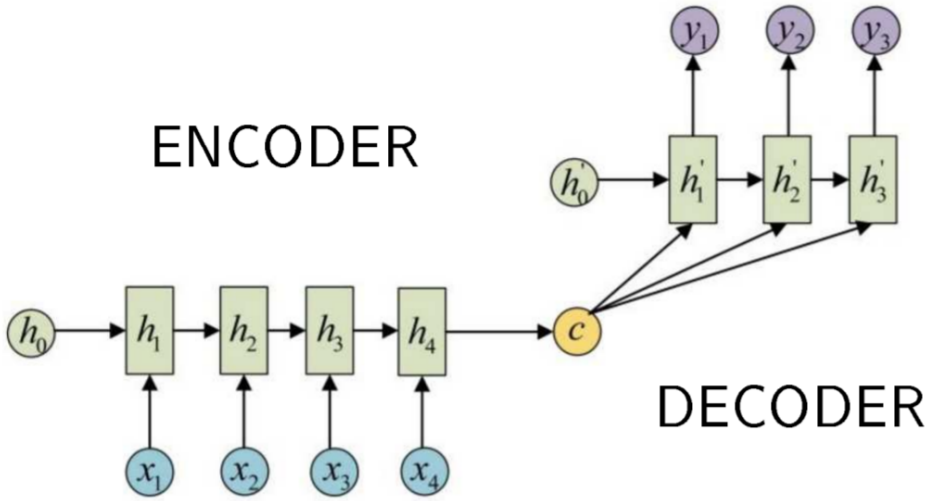
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$
$$y_t = W_{hy} h_t$$

Disadvantages

1. input and output signal lengths must match
2. "reads" the input only from left to right, does not look ahead
3. therefore is not suitable for machine translation, question answering tasks and others

RNN for sequence synthesis (seq2seq)

$X = (x_1, \ldots, x_n)$ — input sequence
$Y = (y_1, \ldots, y_m)$ — output sequence
$c \equiv h_n$ encodes all information about X to synthesize Y



$$h_i = f_{in}(x_i, h_{i-1})$$
$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$
$$y_t = f_y(h'_t, y_{t-1})$$

Summary

- Recurrent neural networks — a simple, powerful and flexible approach to solving various machine learning problems
- Vanilla RNNs are simple, but still not good enough
- So you need to use LSTM or GRU, and seq2seq architecture
- LSTM prevents zeroing gradients
- Clipping helps with «explosion of gradients»
- We need deeper understanding, both theoretical and practical — there are many open questions

Summary

- Recurrent neural networks — a simple, powerful and flexible approach to solving various machine learning problems
- Vanilla RNNs are simple, but still not good enough
- So you need to use LSTM or GRU, and seq2seq architecture
- LSTM prevents zeroing gradients
- Clipping helps with «explosion of gradients»
- We need deeper understanding, both theoretical and practical — there are many open questions

What else can you see?

- Lecture 10 of the course «CS231n» by Andrej Karpathy at Stanford
- How to train neural networks?
- Similar lecture of the School of Data Analysis course (in Russian)
- Fresh interview of Andrej Karpathy by Lex Fridman