MML minor #6

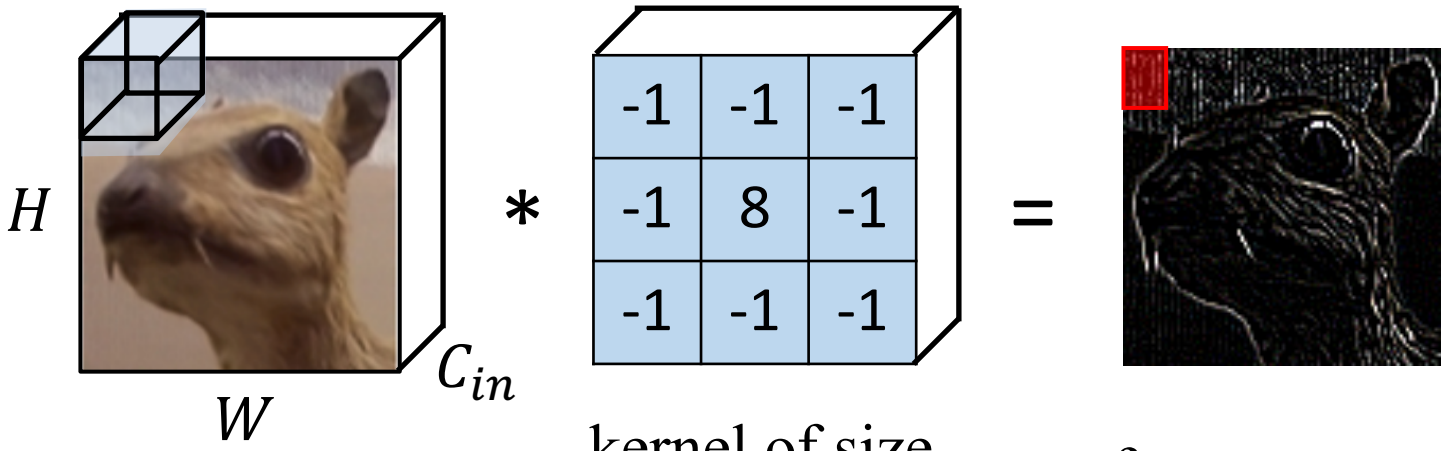
Свёрточные (*продолжение*) и рекуррентные нейронные сети

A color image input

- Let's say we have a color image as an input, which is W$\times H\times C_{in}$ **tensor** (multidimensional array), where
- W – is an image width,
- H – is an image height,
- C_{in} – is a number of input channels (e.g. 3 RGB channels).

A color image input

- Let's say we have a color image as an input, which is W× H×C_{in} **tensor** (multidimensional array), where
- W – is an image width,
- H – is an image height,
- C_{in} – is a number of input channels (e.g. 3 RGB channels).

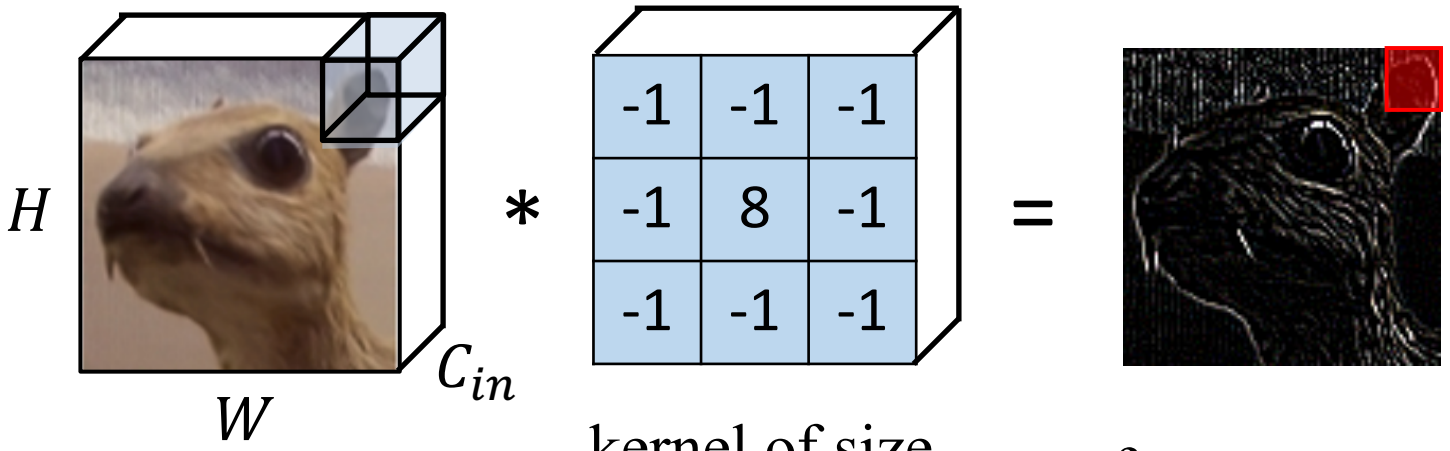


kernel of size W_k× H_k×C_{in}

feature map

A color image input

- Let's say we have a color image as an input, which is W× H ×C_{in} **tensor** (multidimensional array), where
- W – is an image width,
- H – is an image height,
- C_{in} – is a number of input channels (e.g. 3 RGB channels).

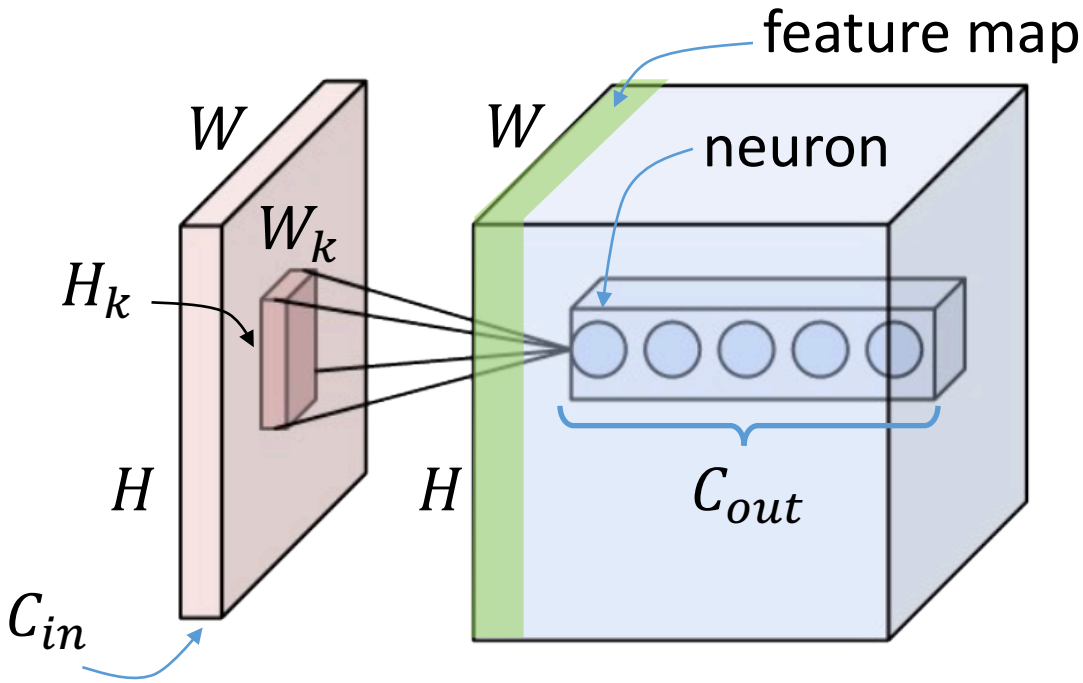


kernel of size W_k× H_k ×C_{in}

feature map

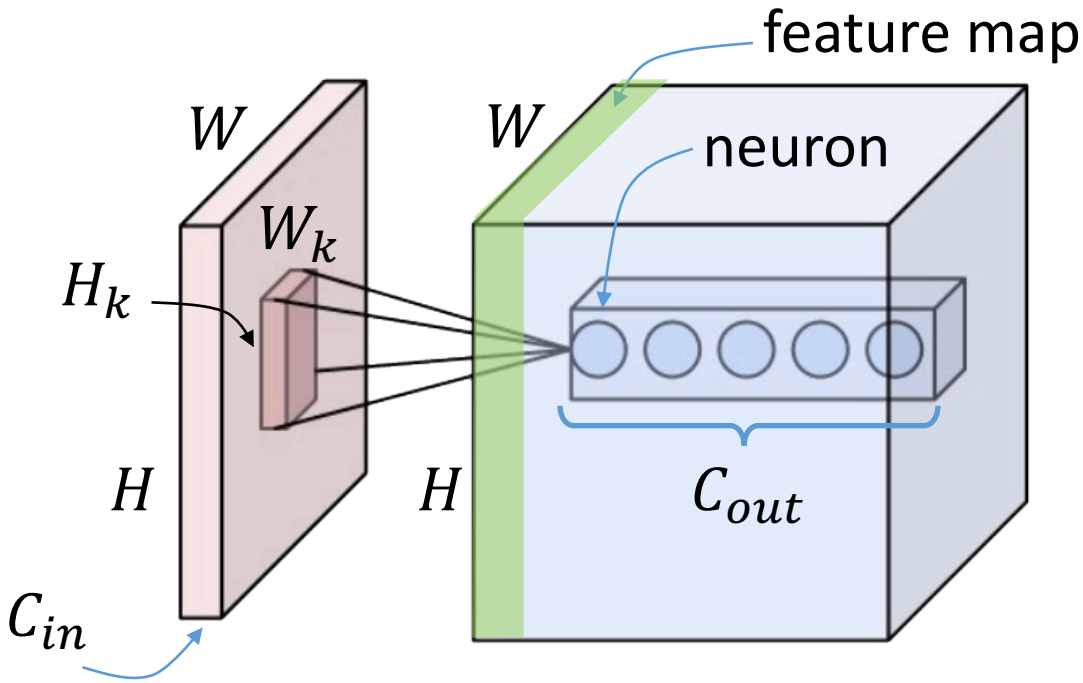
One kernel is not enough!

- We want to train C_{out} kernels of size $W_k \times H_k \times C_{in}$.
- Having a stride of 1 and enough zero padding we can have W$\times H \times C_{out}$ output neurons.



One kernel is not enough!

- We want to train C_{out} kernels of size $W_k \times H_k \times C_{in}$.
- Having a stride of 1 and enough zero padding we can have W$\times H \times C_{out}$ output neurons.



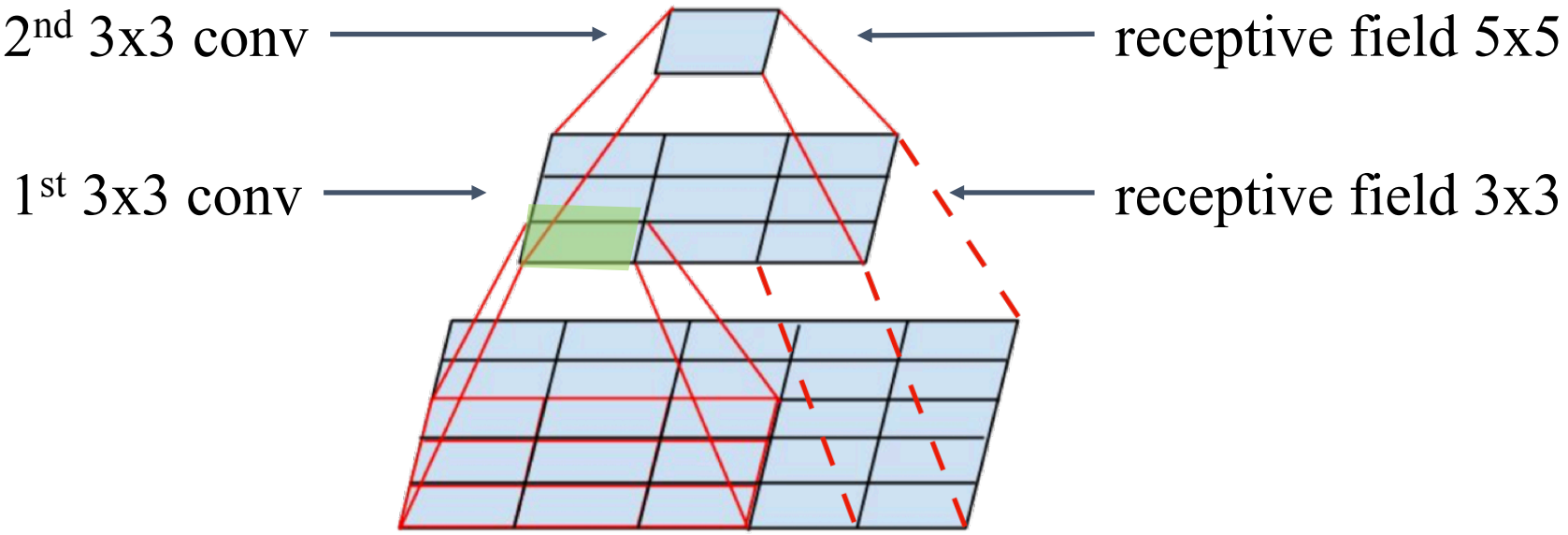
- Using $(W_k * H_k * C_{in} + 1) * C_{out}$ parameters.

One convolutional layer is not enough!

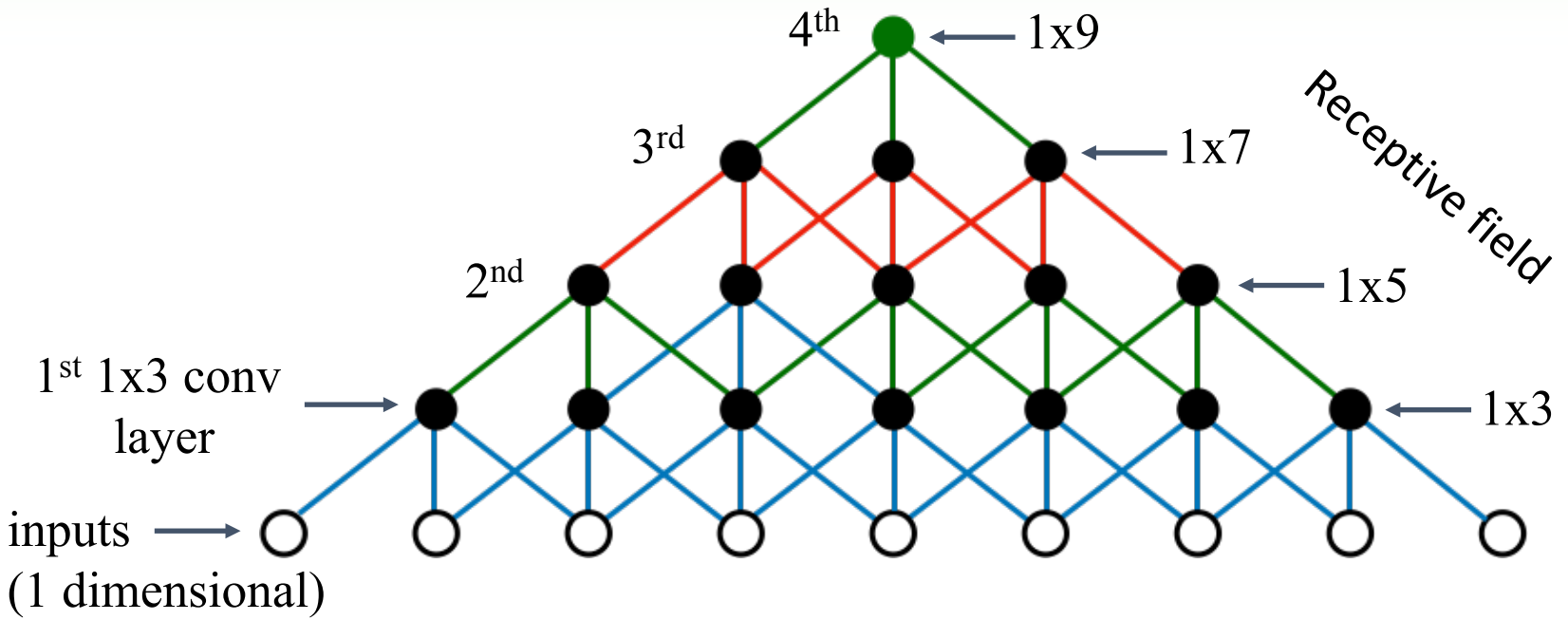
- Let's say neurons of the 1st convolutional layer look at the patches of the image of size 3x3.
- What if an object of interest is bigger than that?
- We need a 2nd convolutional layer on top of the 1st!

One convolutional layer is not enough!

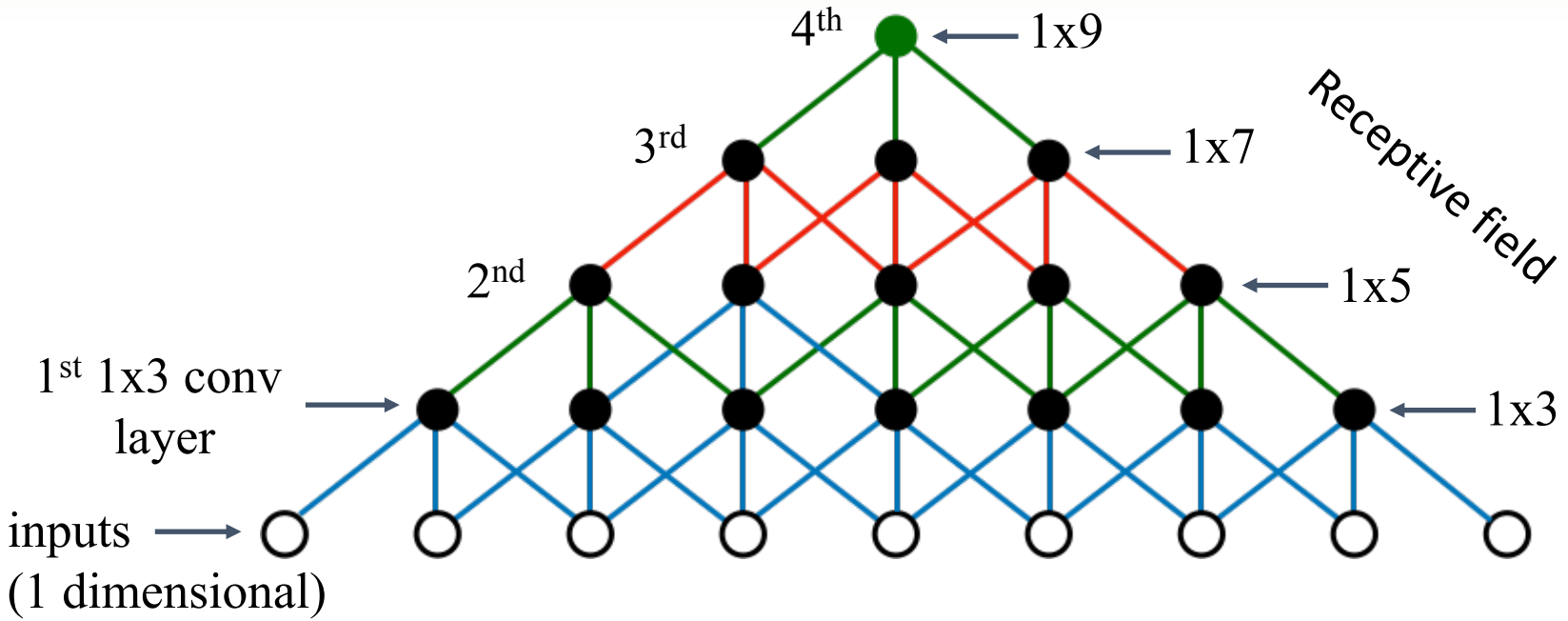
- Let's say neurons of the 1st convolutional layer look at the patches of the image of size 3x3.
- What if an object of interest is bigger than that?
- We need a 2nd convolutional layer on top of the 1st!



Receptive field after N convolutional layers



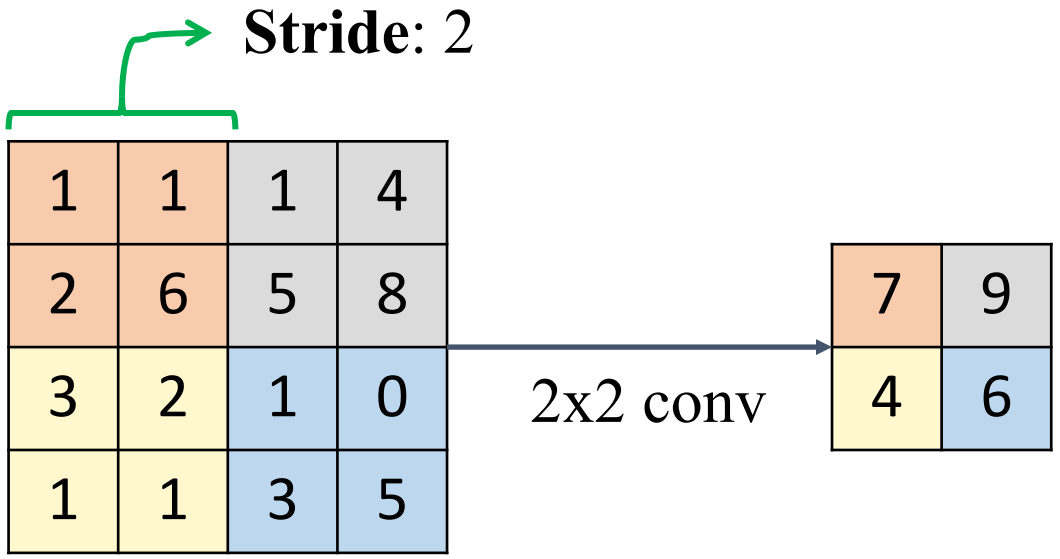
Receptive field after N convolutional layers



- If we stack N convolutional layers with the same kernel size 3x3 the receptive field on N-th layer will be $\boldsymbol{2N + 1 \times 2N + 1}$.
- It looks like we need to stack a lot of convolutional layers!
  To be able to identify objects as big as the input image **300x300** we will need **150** convolutional layers!

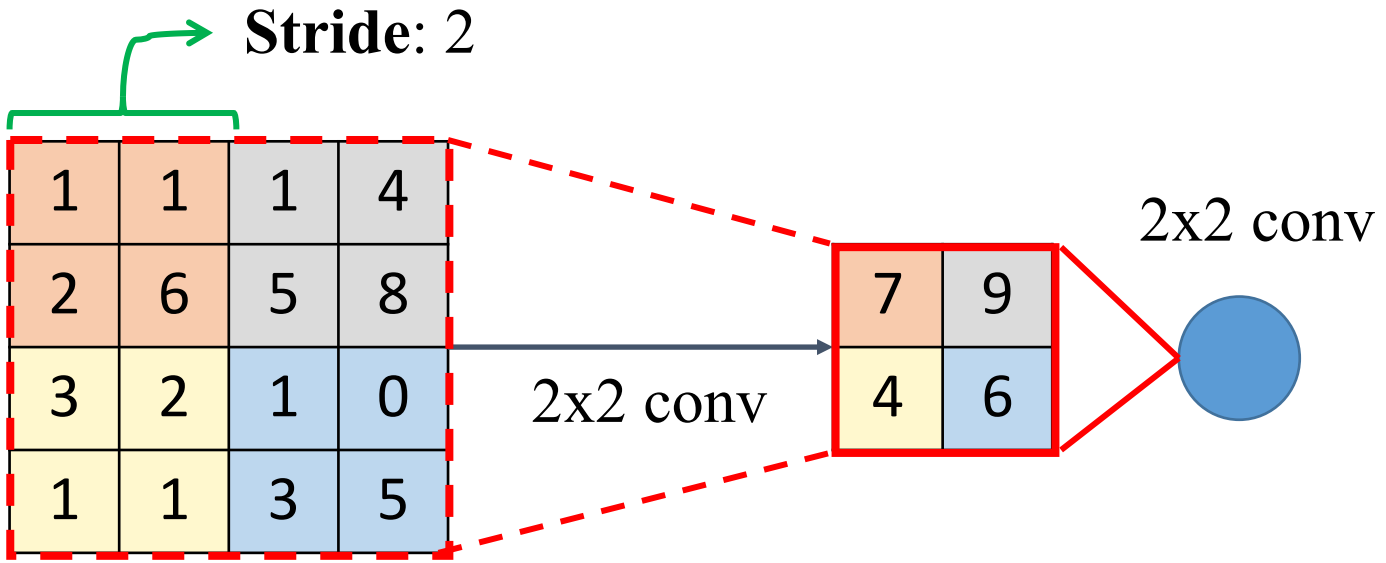
We need to grow receptive field faster!

- We can increase a **stride** in our convolutional layer to reduce the output dimensions!



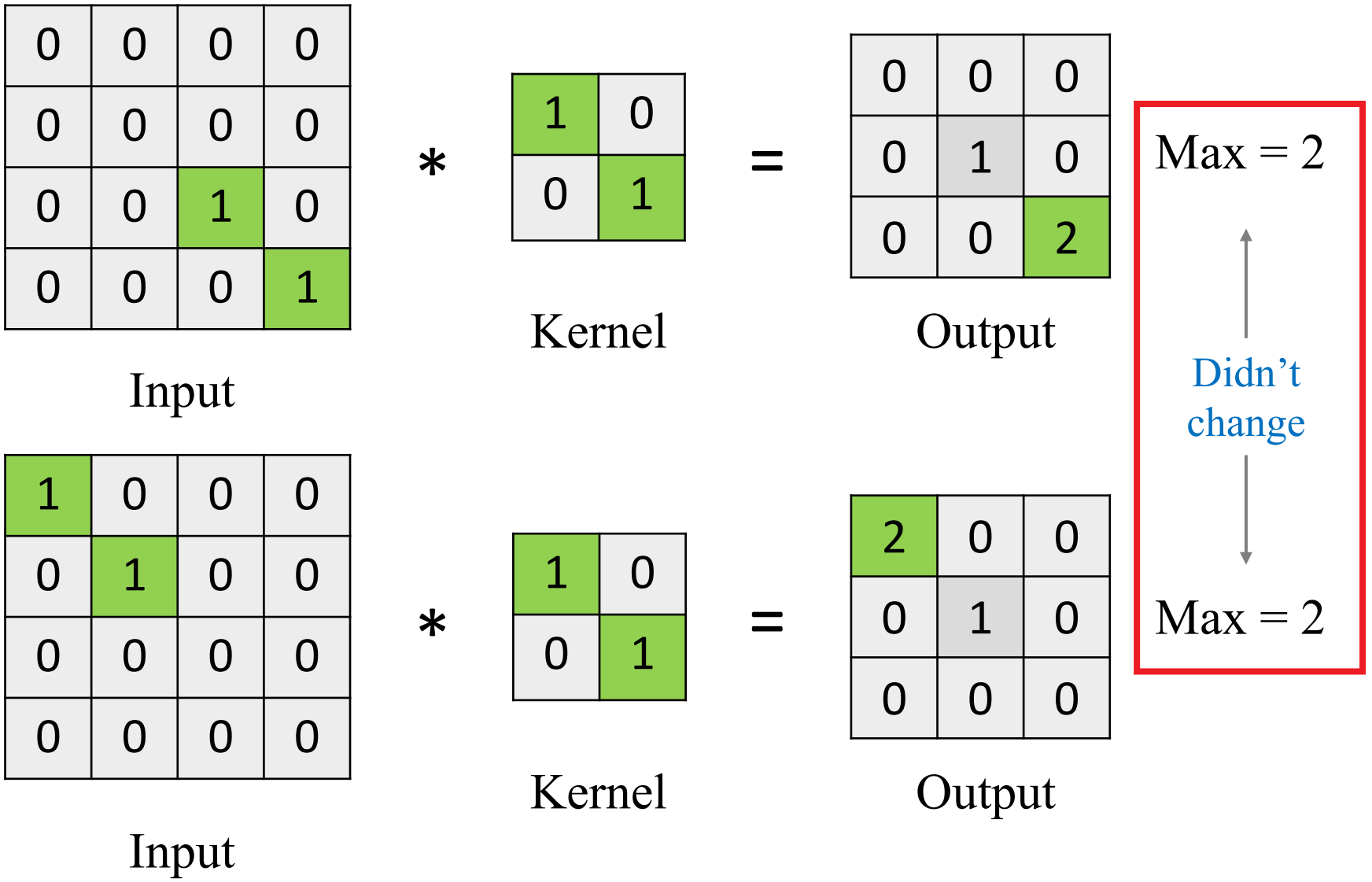
We need to grow receptive field faster!

- We can increase a **stride** in our convolutional layer to reduce the output dimensions!



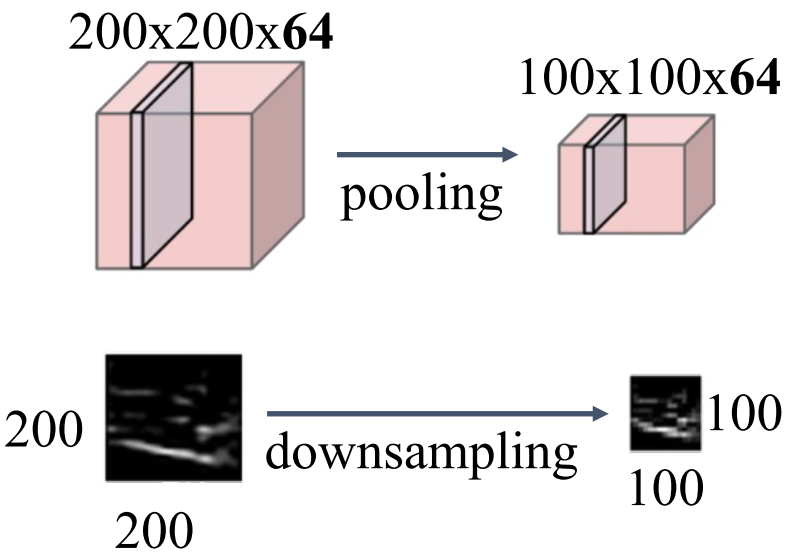
Further convolutions will effectively **double** their receptive field!

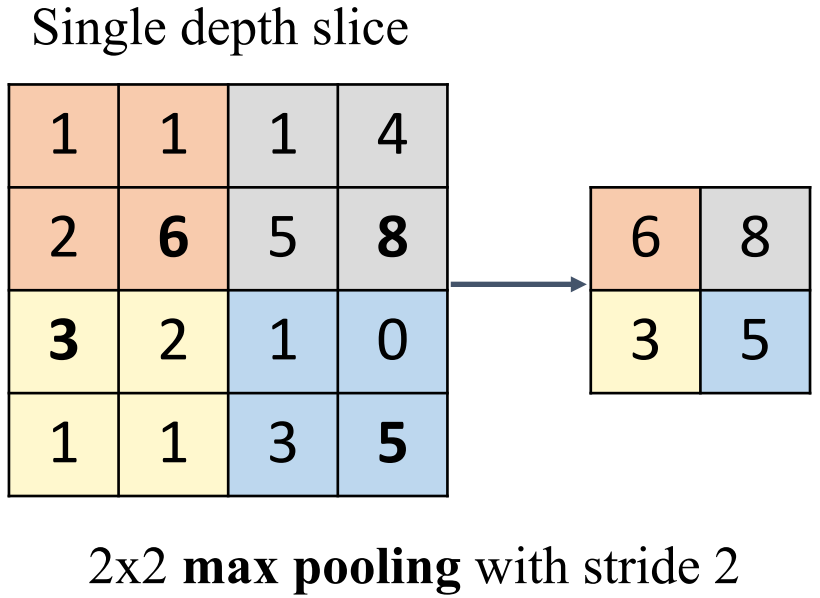
How do we maintain translation invariance?



Pooling layer will help!

- This layer works like a convolutional layer but doesn't have kernel, instead it calculates **maximum** or **average** of input patch values.



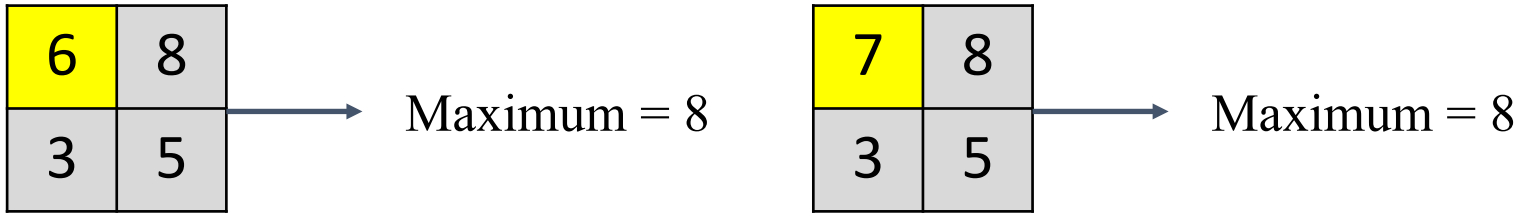


Backpropagation for max pooling layer

Strictly speaking: maximum is not a differentiable function!

Backpropagation for max pooling layer

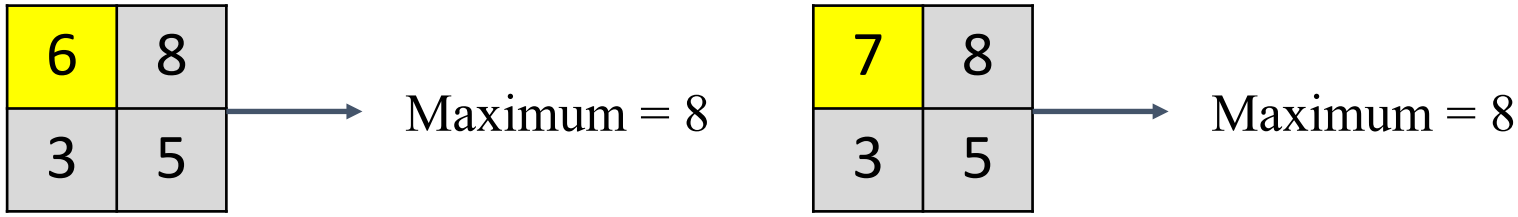
Strictly speaking: maximum is not a differentiable function!



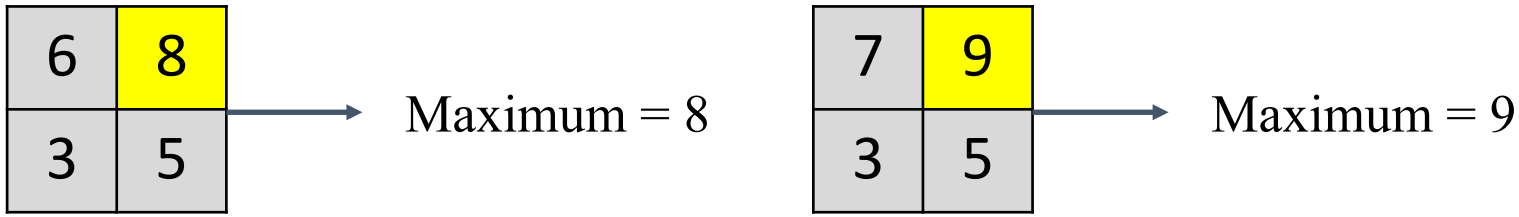
There is no gradient with respect to non maximum patch neurons,
since changing them slightly
does not affect the output.

Backpropagation for max pooling layer

Strictly speaking: maximum is not a differentiable function!



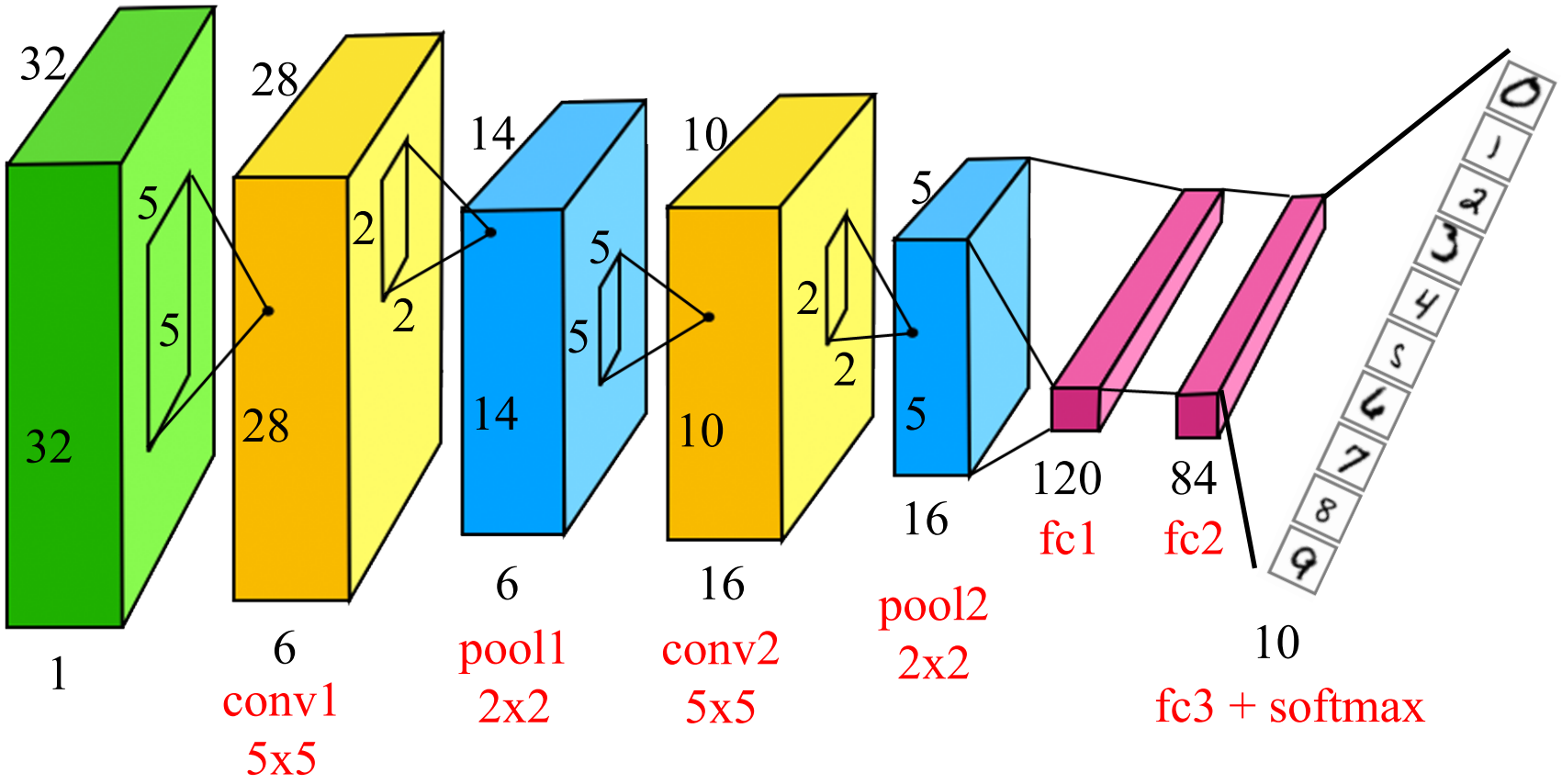
There is no gradient with respect to non maximum patch neurons,
since changing them slightly
does not affect the output.



For the maximum patch neuron we have a gradient of 1.

Putting it all together into a simple CNN

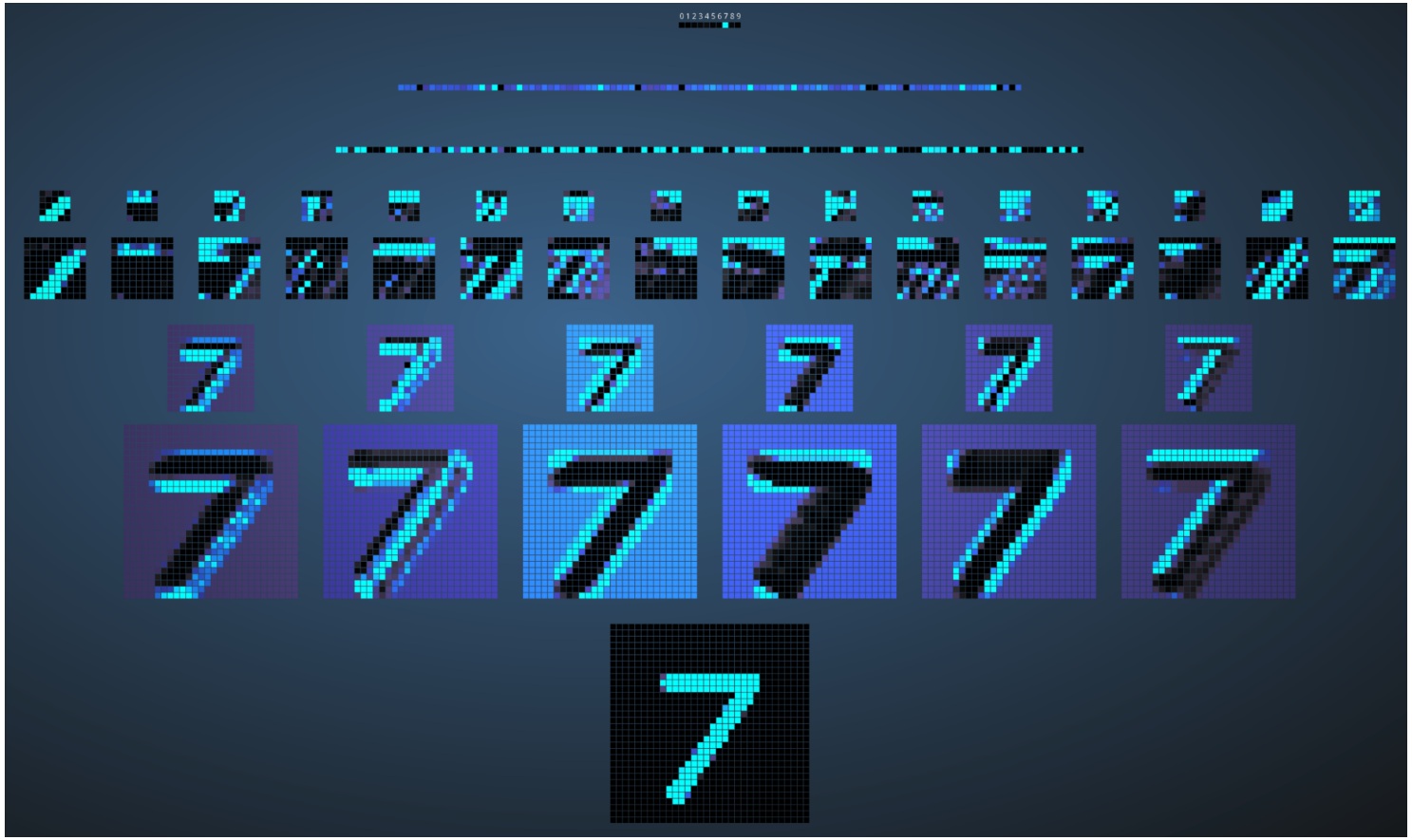
- LeNet-5 architecture (1998) for handwritten digits recognition on MNIST dataset:



http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

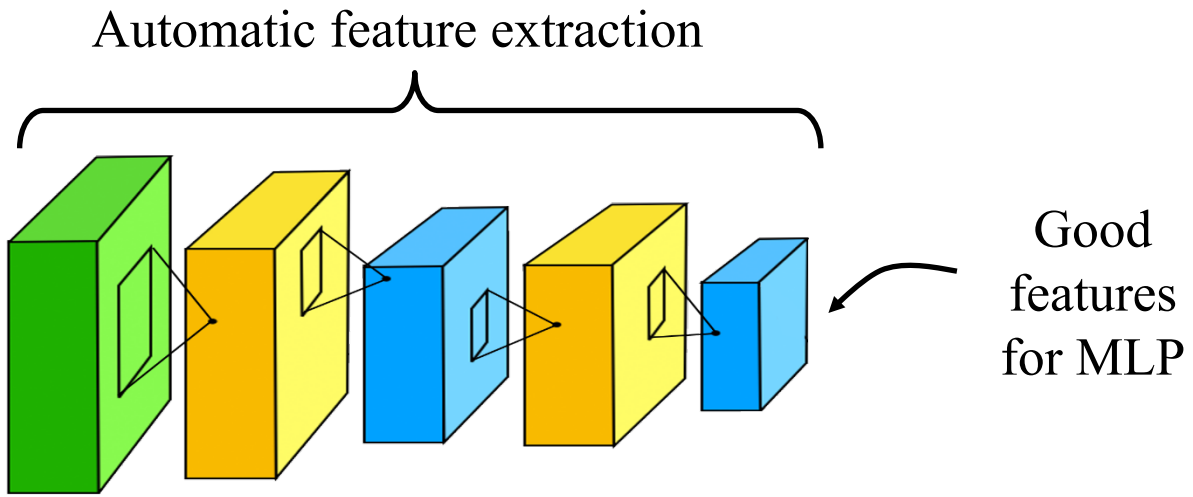
Визуализация обученной сети для MNIST

- http://scs.ryerson.ca/~aharley/vis/conv/flat.html



Learning deep representations

- Neurons of deep convolutional layers learn complex representations that can be used as features for classification with MLP.



Inputs that provide highest activations:

http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

Современные применения: перенос стиля

Картинка | Стиль | Картинка со стилем

https://github.com/luanfujun/deep-photo-styletransfer

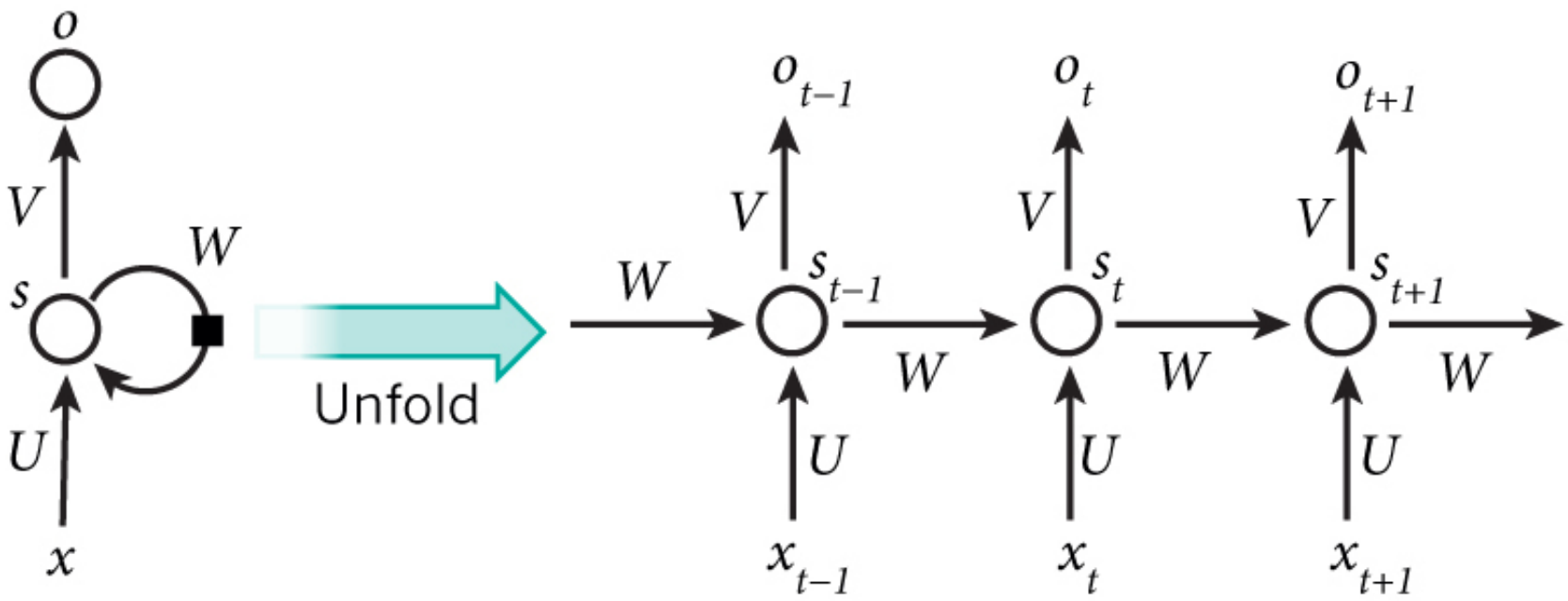
Еще примеры применений

- https://quickdraw.withgoogle.com
- https://affinelayer.com/pixsrv/

Recurrent Neural Networks (RNN)

- Работают с последовательностями
  - Слов в предложении
  - Букв в предложении
  - Отсчетов в аудио сигнале (амплитуда, частота)
  - Пикселей изображения
  - …

Как устроена простая RNN



$$s_t = f(Ux_t + Ws_{t-1})$$

$$o_t = \mathrm{softmax}(Vs_t)$$

Пример:

x_t– слово в предложении

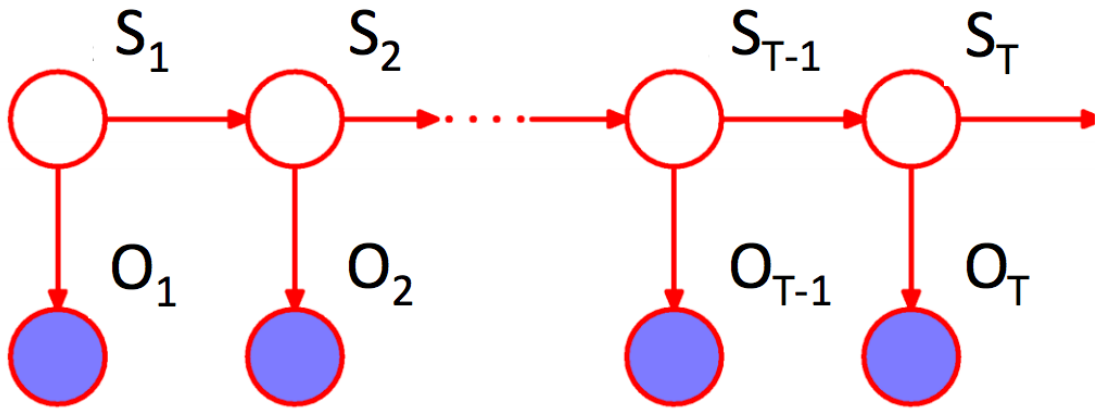
o_t– следующее слово в предложении

- Работает **одинаково** для каждого элемента последовательности x_t, но вычисления зависят от предыдущих элементов x_t
- Можно сказать, что у RNN есть память (**скрытое состояние** s_t), в которой хранится информация о предыдущих элементах последовательности

А как же Hidden Markov Model (HMM)?

$O_t \in \{y_1, y_2, \ldots, y_K\}$

$S_t \in \{1, \ldots, I\}$



Пример:

s_t – POS tag

o_t – слово в предложении

$$p(S_1, \ldots, S_T, O_1, \ldots, O_T) = \prod_{t=1}^{T} p(O_t|S_t) \prod_{t=1}^{T} p(S_t|S_{t-1})$$

1st order Markov assumption on hidden states $\{S_t\}$ t = 1, …, T

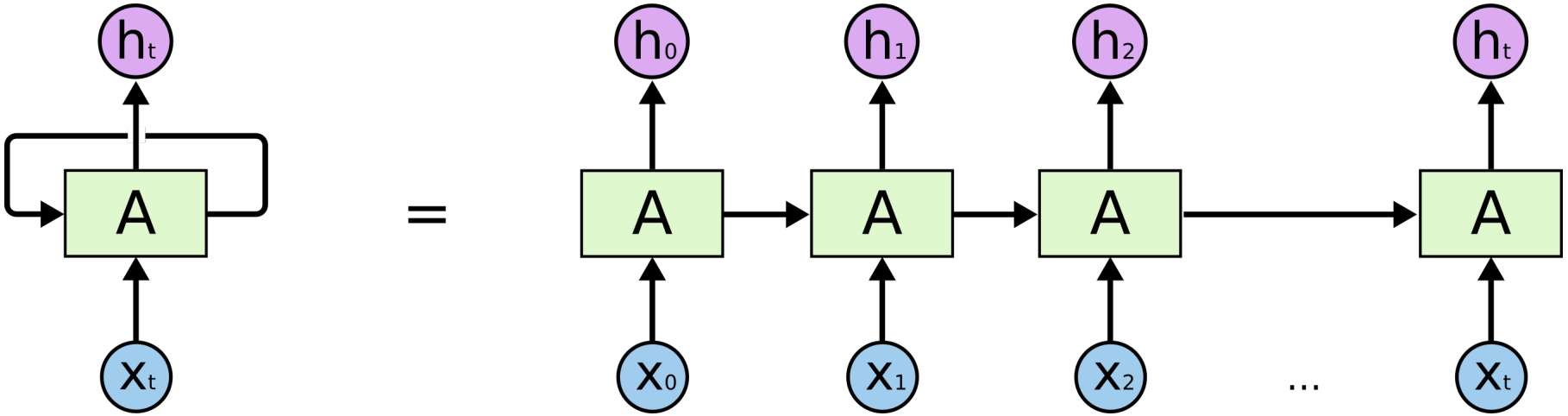
(can be extended to higher order).

http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/

http://www.cs.cmu.edu/~aarti/Class/10701/slides/Lecture17.pdf

Backpropagation through time (BPTT)

- Если развернуть сеть по времени t, то получим обычную feed-forward сеть с shared параметрами
- Применяем backpropagation, считаем градиенты для каждого параметра
- Далее (так же, как в случае сверток CNN) суммируем градиенты по shared параметрам и делаем шаг SGD



https://en.wikipedia.org/wiki/Backpropagation_through_time

Посчитаем производную для 2 шагов

$$s_1 = f(Ux_1 + Ws_0)$$

a_2

$$s_2 = f(Ux_2 + Ws_1) = f\big(Ux_2 + Wf(Ux_1 + Ws_0)\big)$$

$$\frac{\partial s_2}{\partial W} = \frac{\partial f}{\partial a_2}\left(W\frac{\partial s_1}{\partial W} + s_1\right) = \frac{\partial f}{\partial a_2}W\frac{\partial s_1}{\partial W} + \frac{\partial f}{\partial a_2}s_1$$

$$\frac{\partial s_1}{\partial W} = \frac{\partial s_1}{\partial W_*}$$

$$\frac{\partial s_2}{\partial s_1} \qquad \frac{\partial s_2}{\partial W_*}$$

$$\frac{\partial s_2}{\partial W} = \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_*} + \frac{\partial s_2}{\partial W_*}$$

Обозначение
в предположении,
что s_1 не зависит от w

Посчитаем производную для 3 шагов

$$s_3 = f(Ux_3 + Ws_2)$$

$$\frac{\partial s_2}{\partial W} = \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_*} + \frac{\partial s_2}{\partial W_*}$$

$$\frac{\partial s_3}{\partial s_2} \qquad\qquad \frac{\partial s_3}{\partial W_*}$$

$$\frac{\partial s_3}{\partial W} = \frac{\partial f}{\partial a_3}\left(W\frac{\partial s_2}{\partial W} + s_2\right) = \frac{\partial f}{\partial a_3}W\left(\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_*} + \frac{\partial s_2}{\partial W_*}\right) + \frac{\partial f}{\partial a_3}s_2$$

$$\frac{\partial s_3}{\partial W} = \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_*} + \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W_*} + \frac{\partial s_3}{\partial W_*}$$

Обозначение
в предположении,
что s_2 не зависит от w

Индукцией легко показать, что…

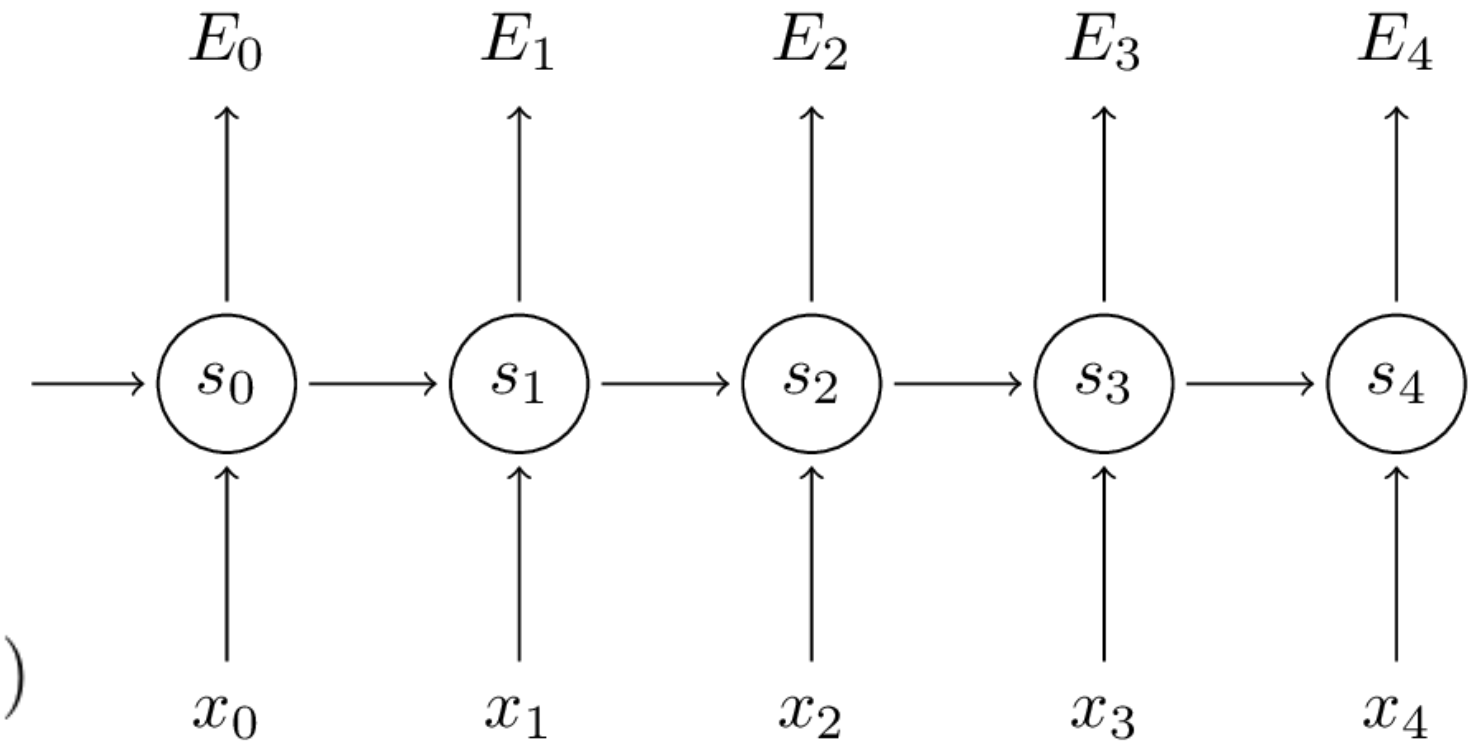
$$\frac{\partial s_2}{\partial W} = \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_*} + \frac{\partial s_2}{\partial W_*}$$

$$\frac{\partial s_3}{\partial W} = \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W_*} + \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W_*} + \frac{\partial s_3}{\partial W_*}$$

$$\frac{\partial s_k}{\partial W} = \sum_{i=1}^{k}\left(\prod_{j=i+1}^{k}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_i}{\partial W_*}$$

BPTT на примере

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = \mathrm{softmax}(Vs_t)$$

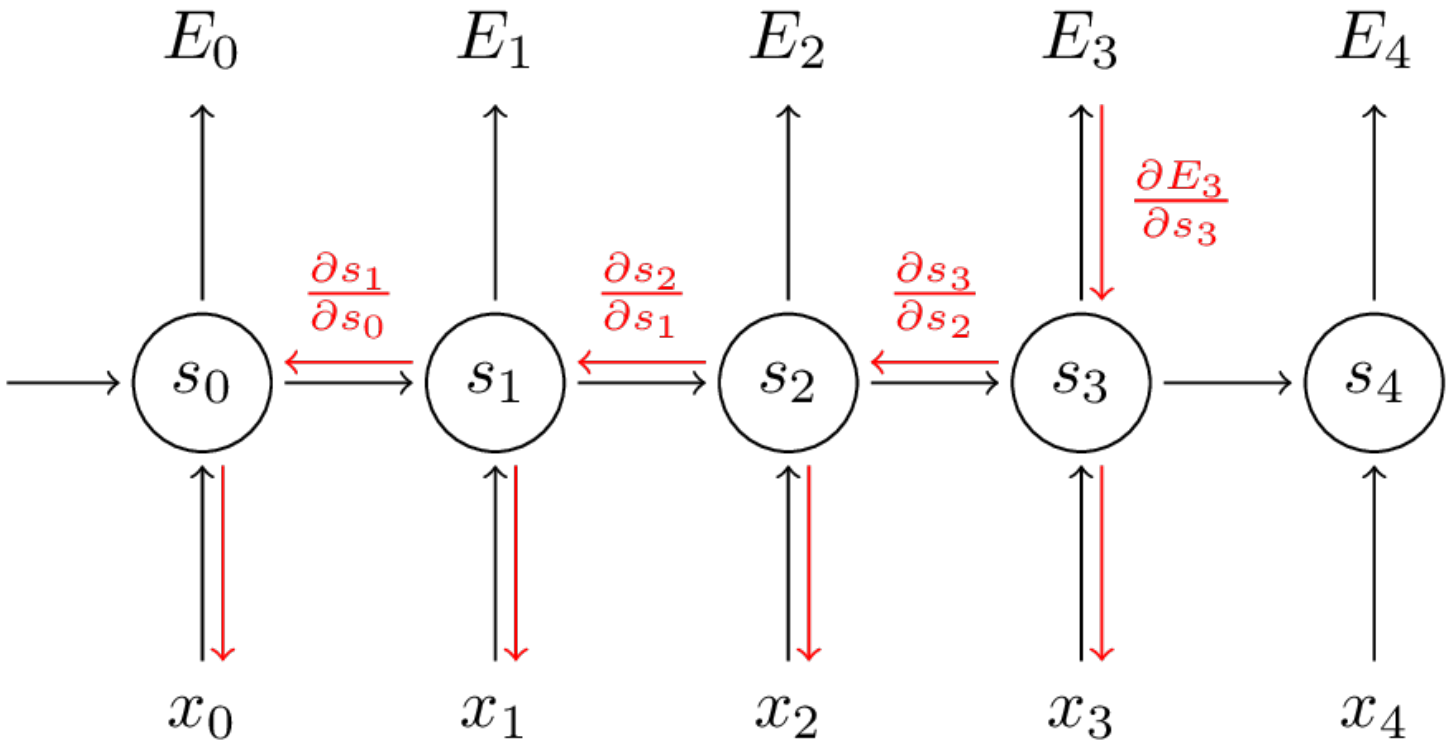


$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$
$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$
$$= -\sum_t y_t \log \hat{y}_t$$

http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

BPTT на примере

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = \mathrm{softmax}(Vs_t)$$



$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W} =$$
$$= \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\sum_{i=1}^{3}\left(\prod_{j=i+1}^{3}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_i}{\partial W_*}$$

http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

Проблема с затухающими градиентами

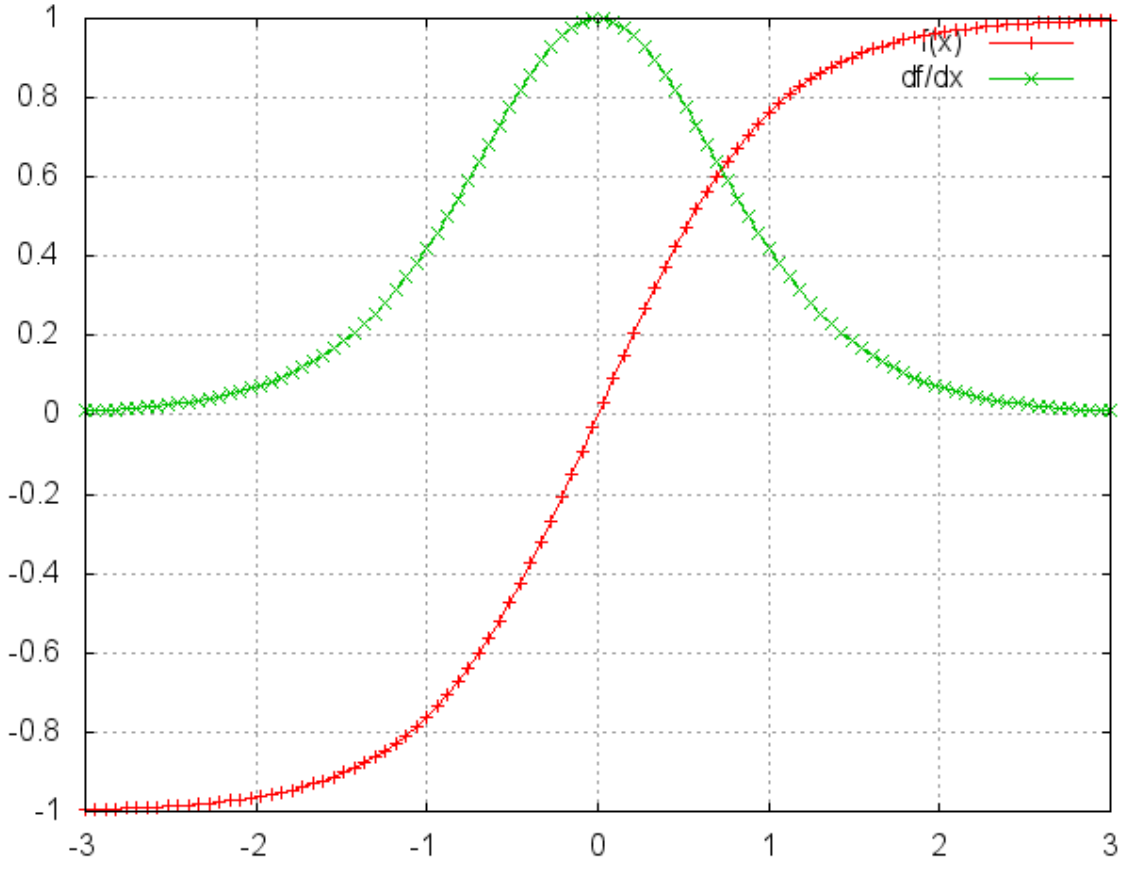
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\sum_{i=1}^{3}\left(\prod_{j=i+1}^{3}\frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_i}{\partial W_*}$$

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\frac{\partial s_j}{\partial s_{j-1}} = \frac{\partial f}{\partial a_j}W \qquad f(x) = \tanh(x)$$

$$\prod_{j=i+1}^{3}\frac{\partial s_j}{\partial s_{j-1}}$$

Быстро убывает
или взрывается

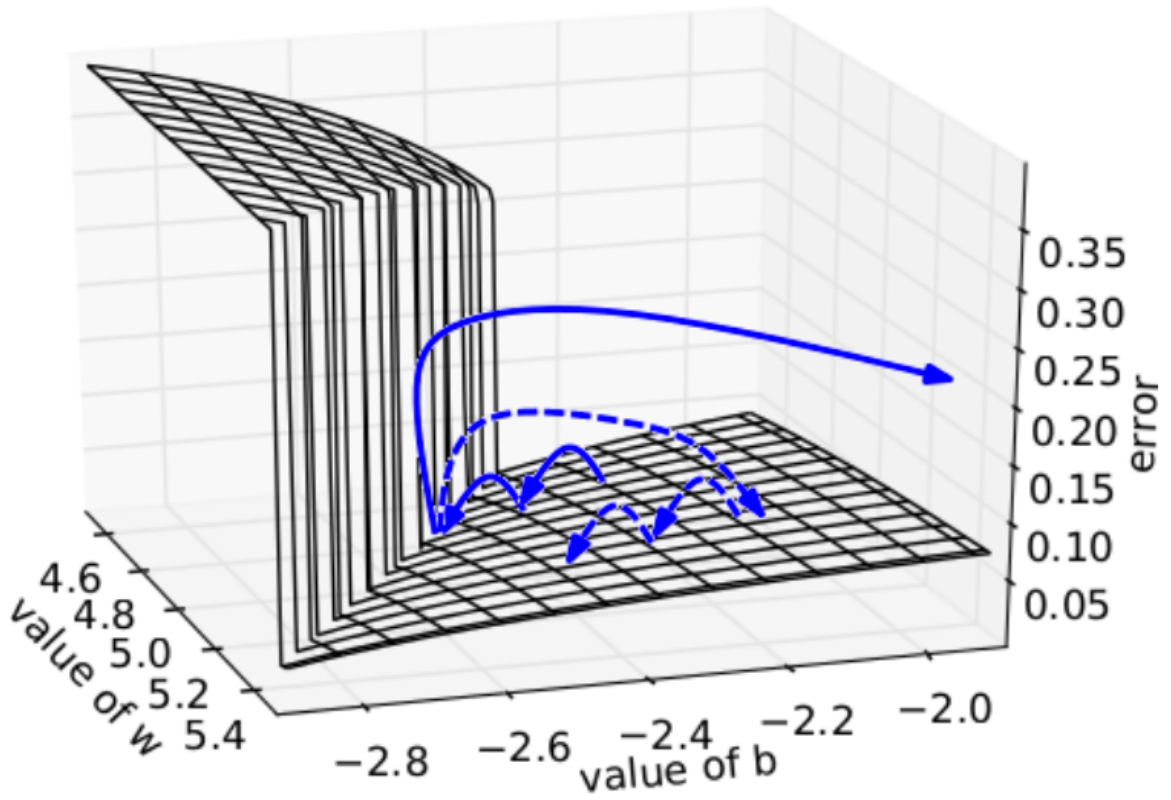


http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

Gradient clipping

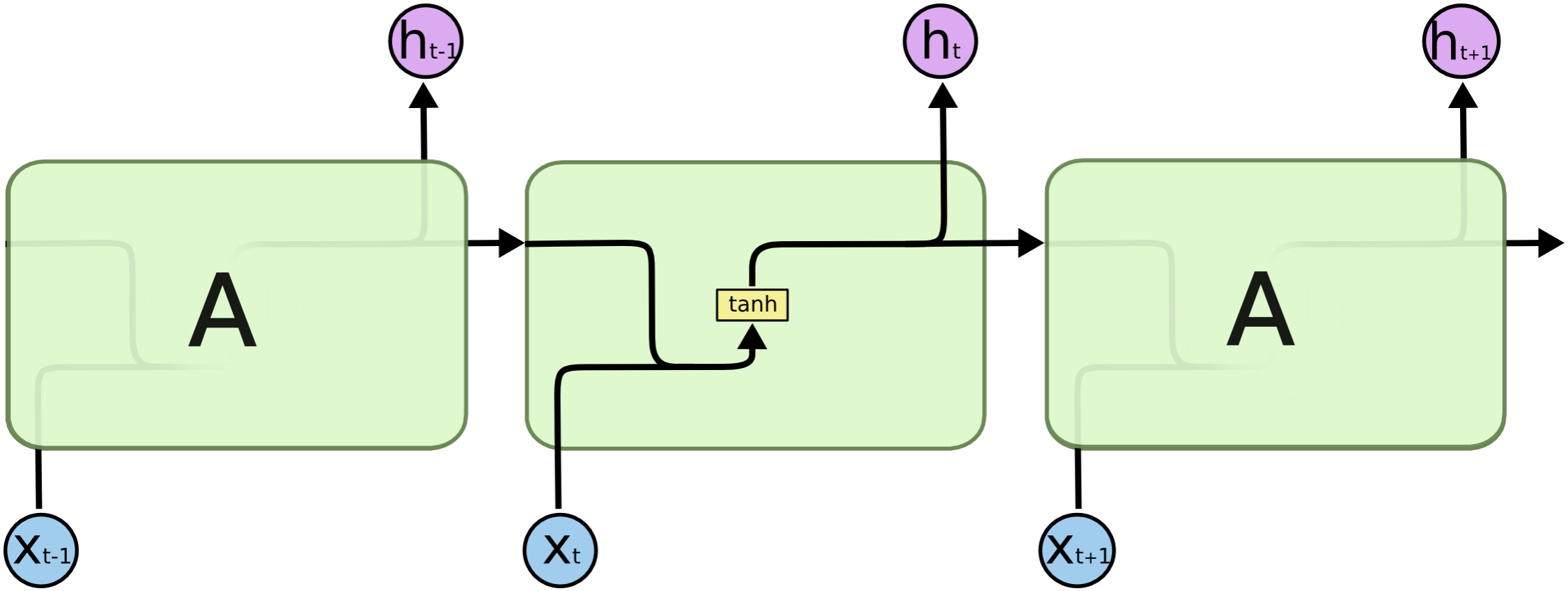
- Градиенты могут взорваться – ограничим их норму

$$
\begin{aligned}
&\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta} \\
&\textbf{if } \|\hat{\mathbf{g}}\| \geq threshold \textbf{ then} \\
&\quad \hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}} \\
&\textbf{end if}
\end{aligned}
$$



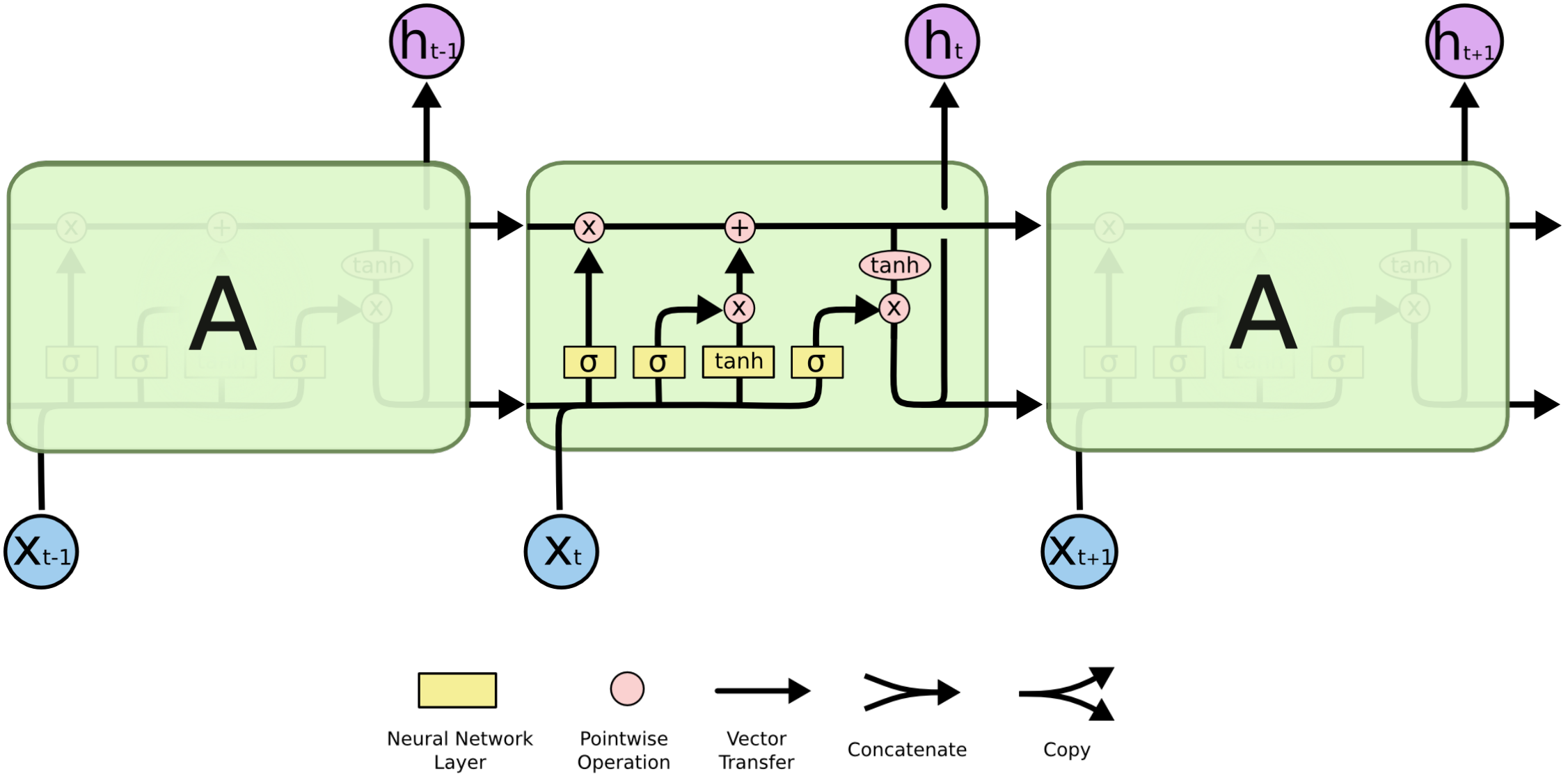
https://arxiv.org/pdf/1211.5063.pdf

Простая RNN плохо работает



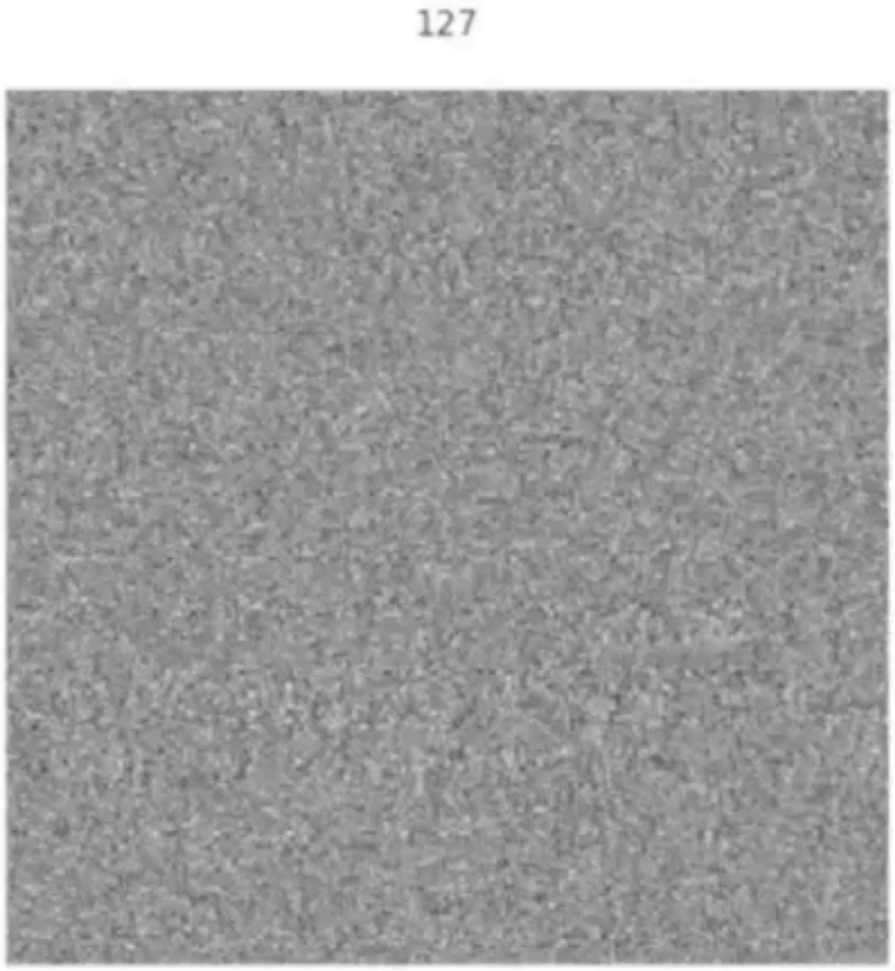
$$h_t = \tanh(Ux_t + Wh_{t-1})$$

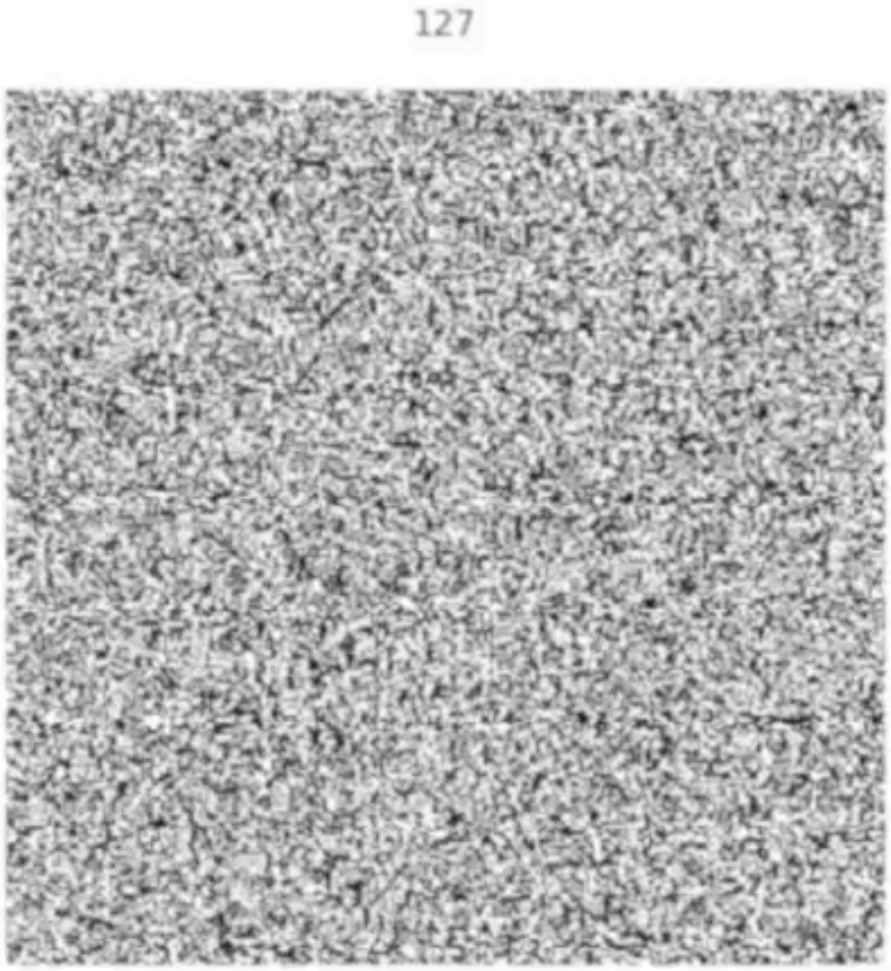
Спасет LSTM


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM решает проблему затухания градиентов

Распространение ошибки на 128 шаге

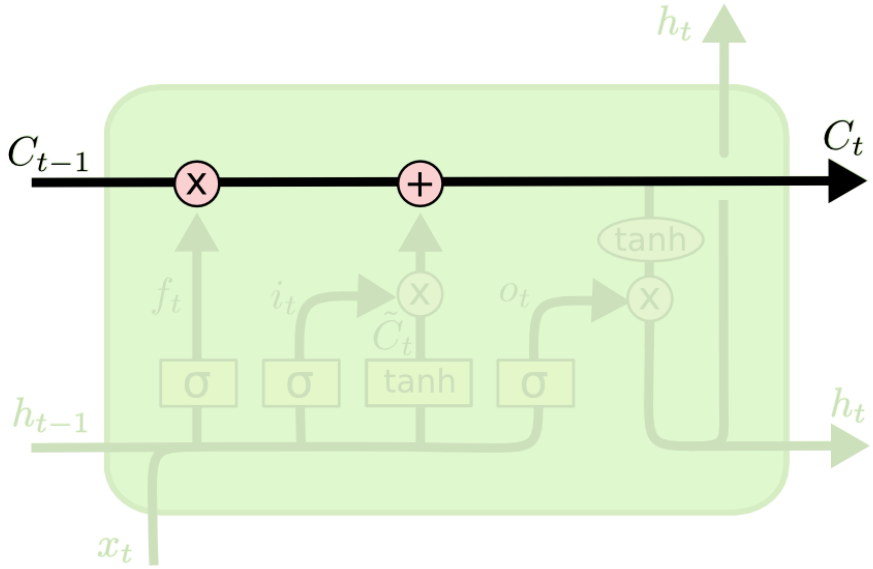


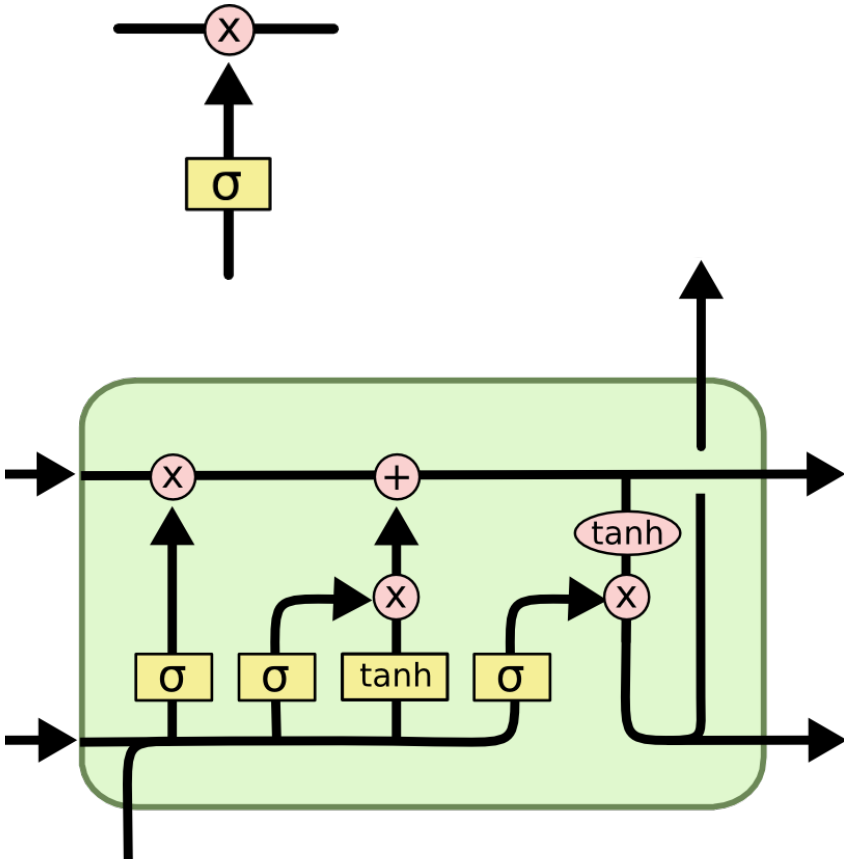


https://imgur.com/gallery/vaNahKE

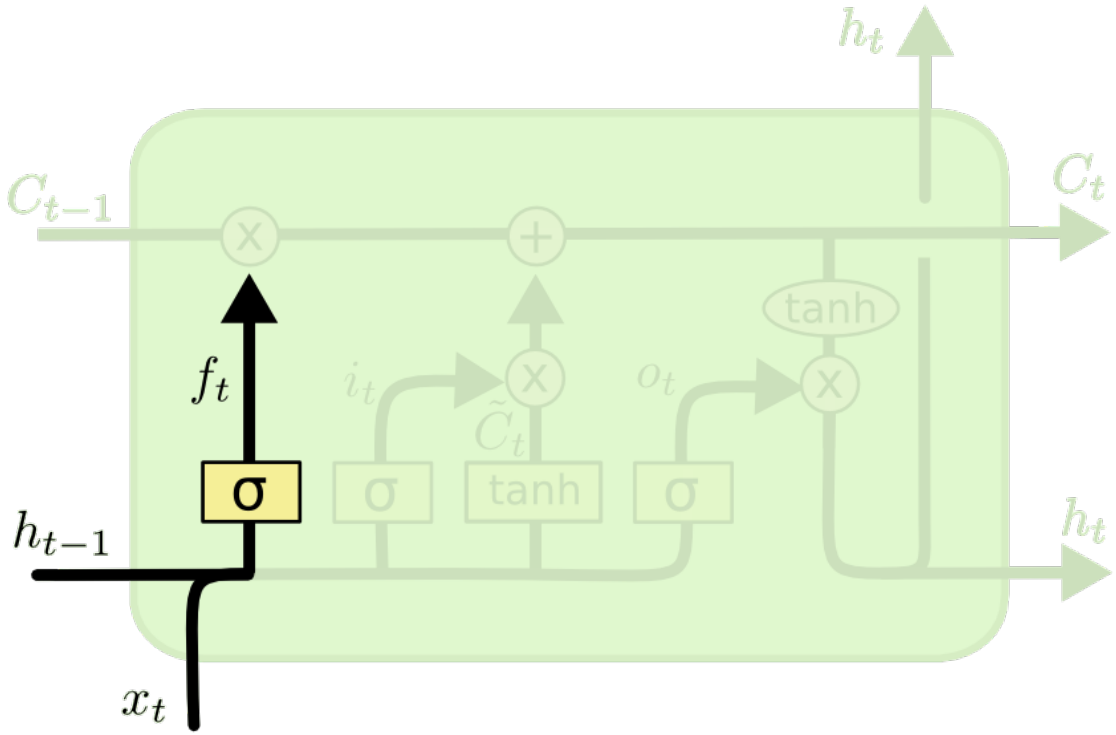
LSTM cell state

- C_t проходит через все ячейки, LSTM может забывать или добавлять информацию в C_t
- Gates учат маски для забывания (**forget**), добавления (**input**) и вывода (**output**) вектора состояний C_t



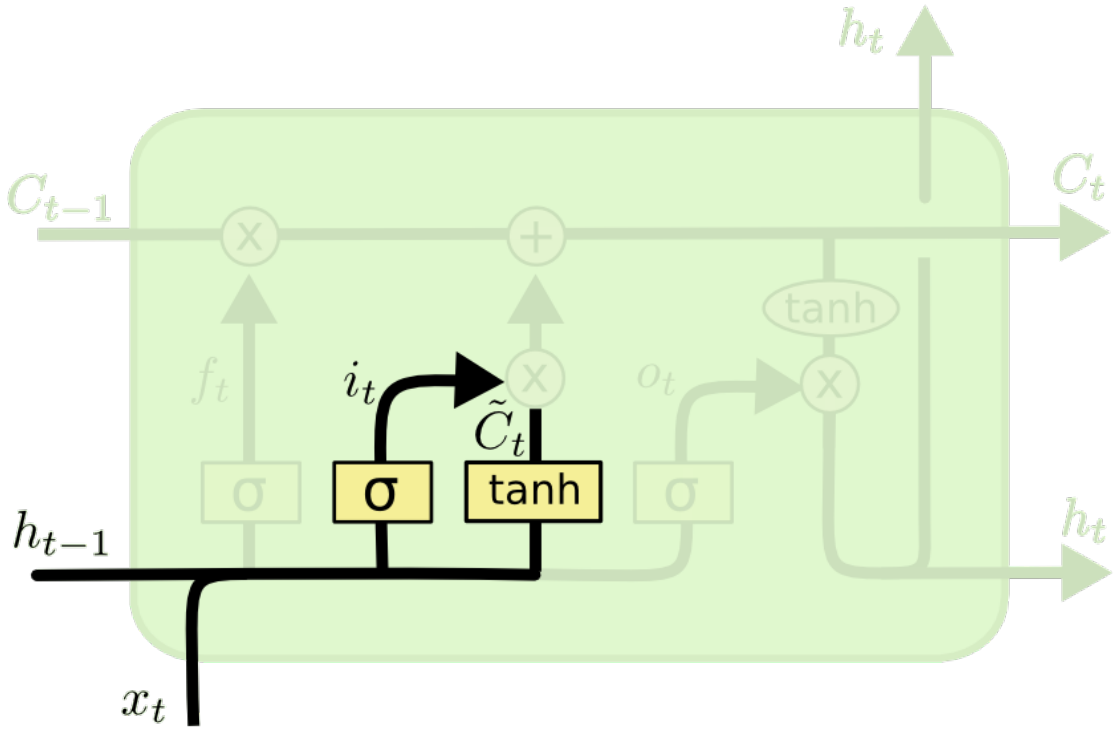


LSTM forget gate



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

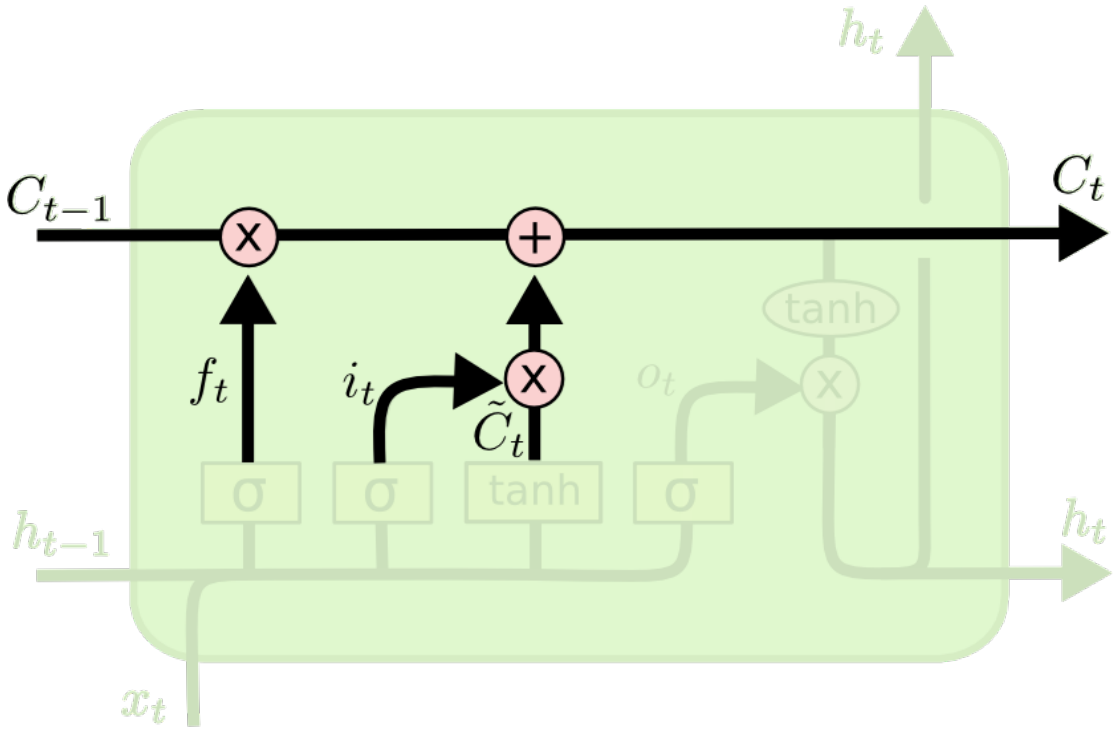
LSTM input gate и вклад ячейки в состояние



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

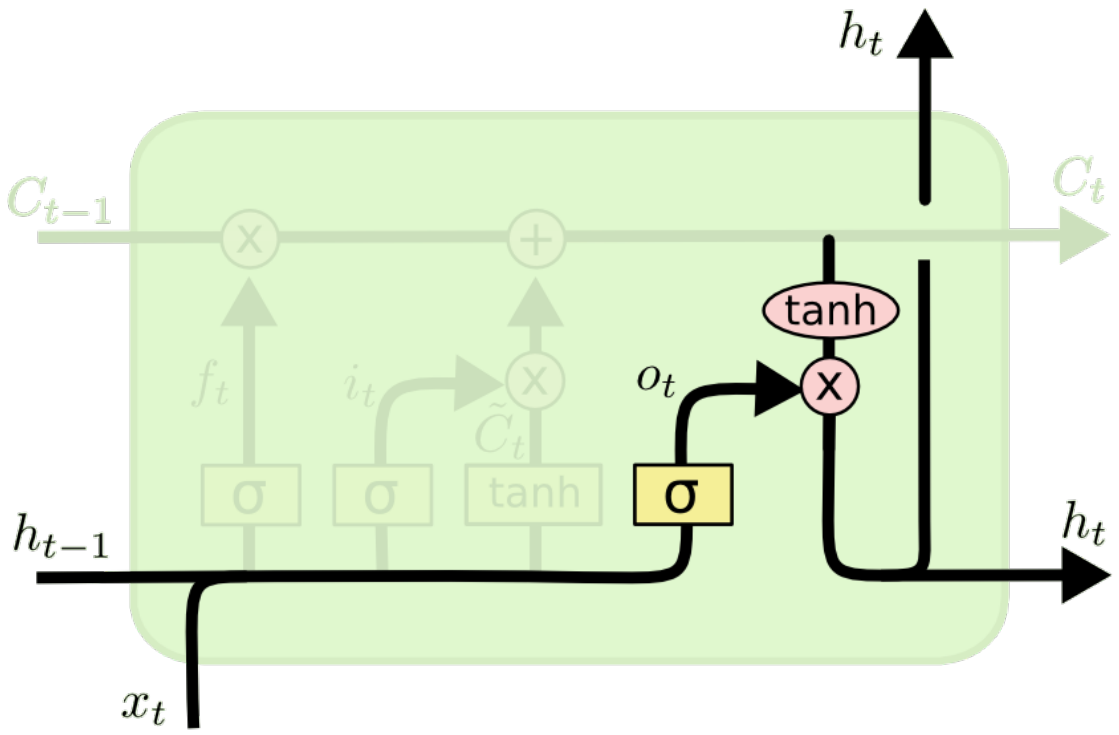
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Обновление состояния



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

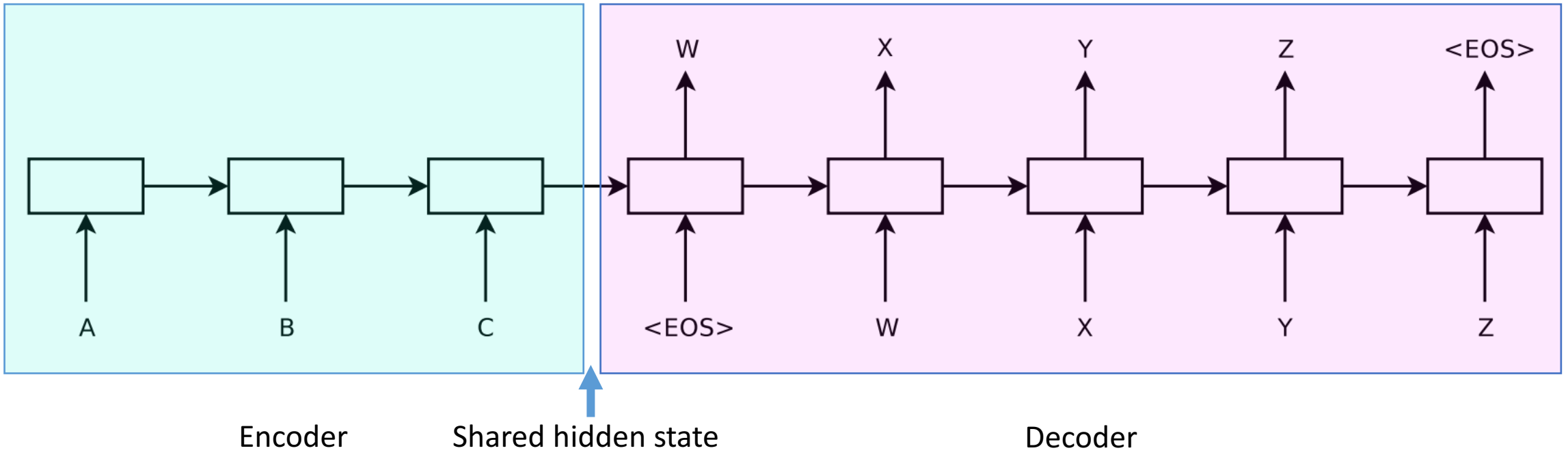
Выдача скрытого состояния наружу



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

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

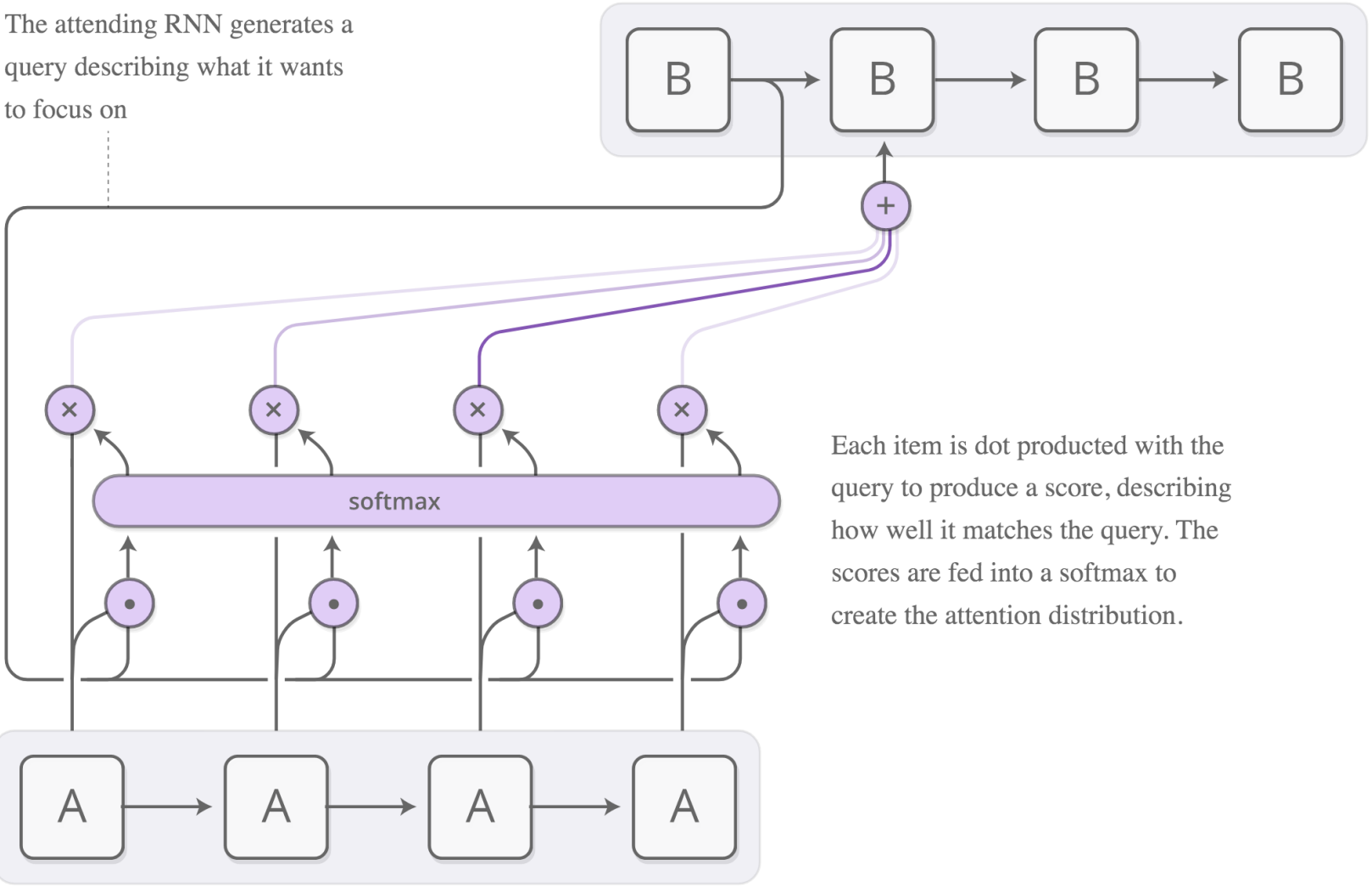
Encoder-decoder (seq2seq)



- Слабое место – связь между encoder и decoder только через последний скрытый слой encoder

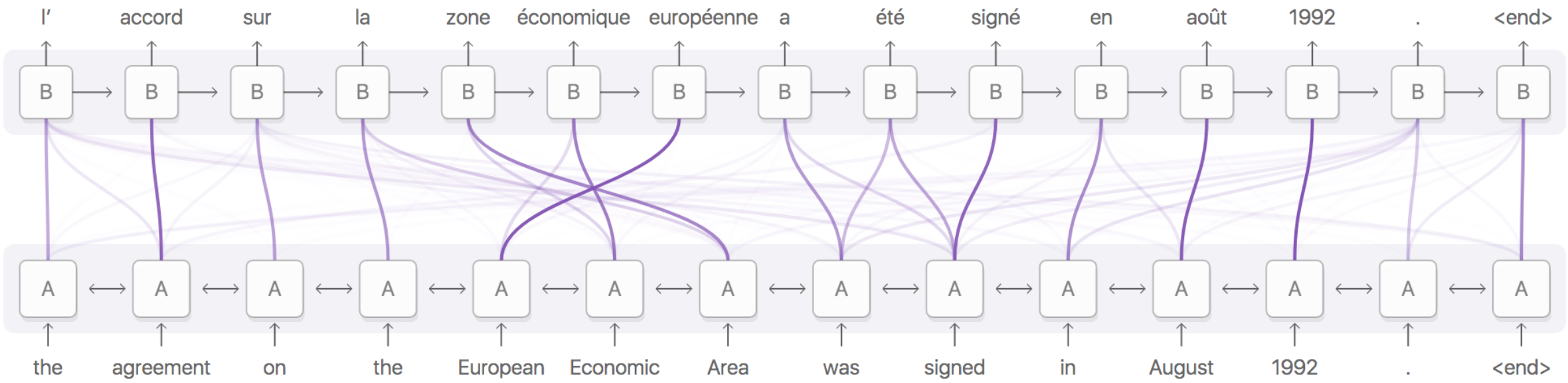
https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Поможет attention



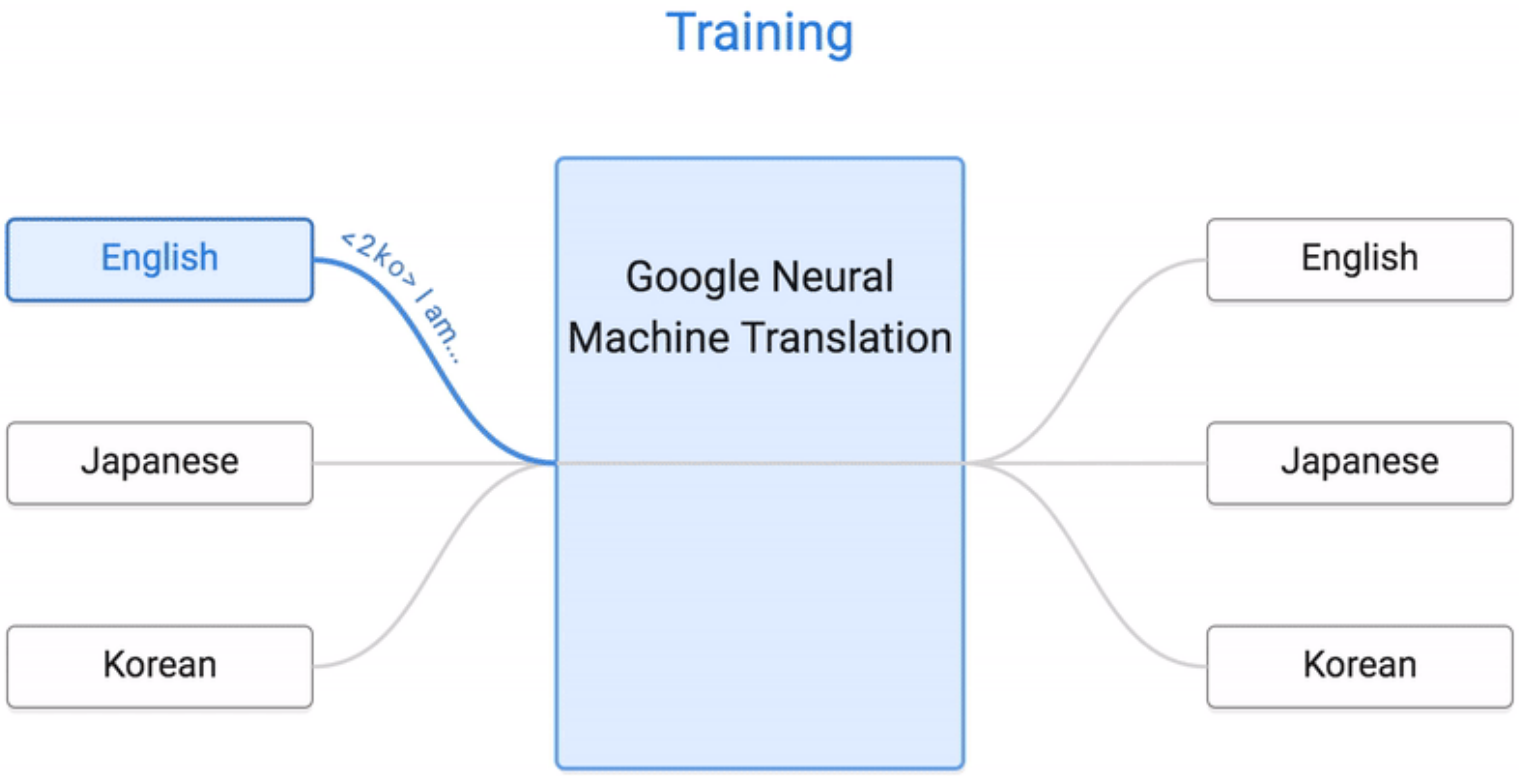
http://distill.pub/2016/augmented-rnns/#attentional-interfaces

Машинный перевод (seq2seq + attention)



Multi-language перевод от Google

- Обучение encoder и decoder для каждого языка на разных парах языков одновременно



https://research.googleblog.com/2016/11/zero-shot-translation-with-googles.html

Распознавание голоса (seq2seq + attention)

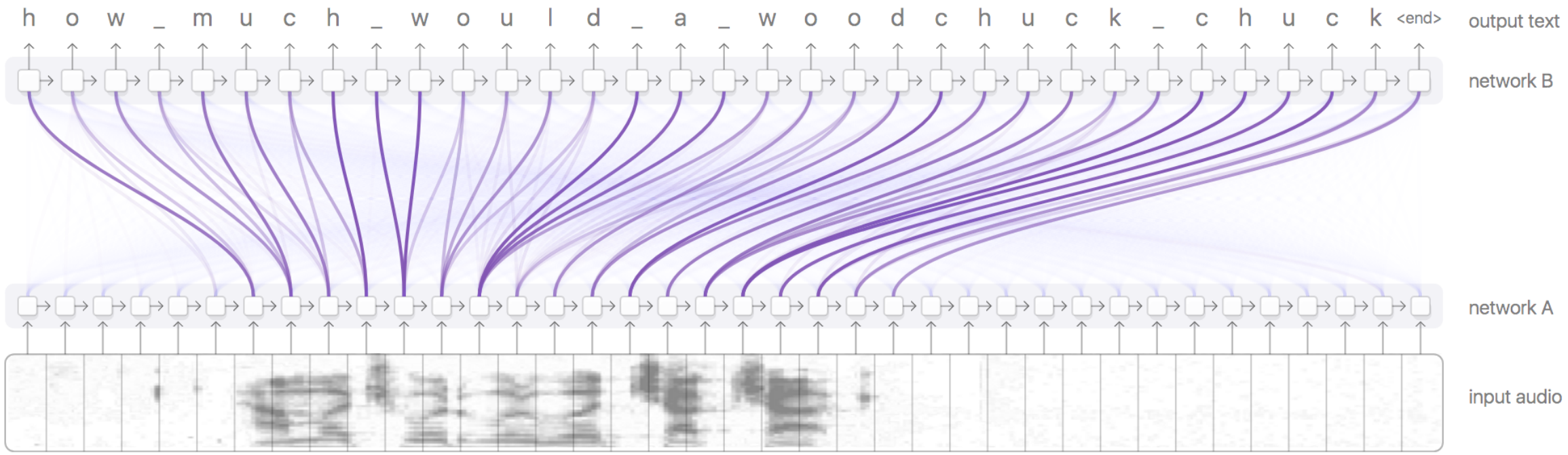
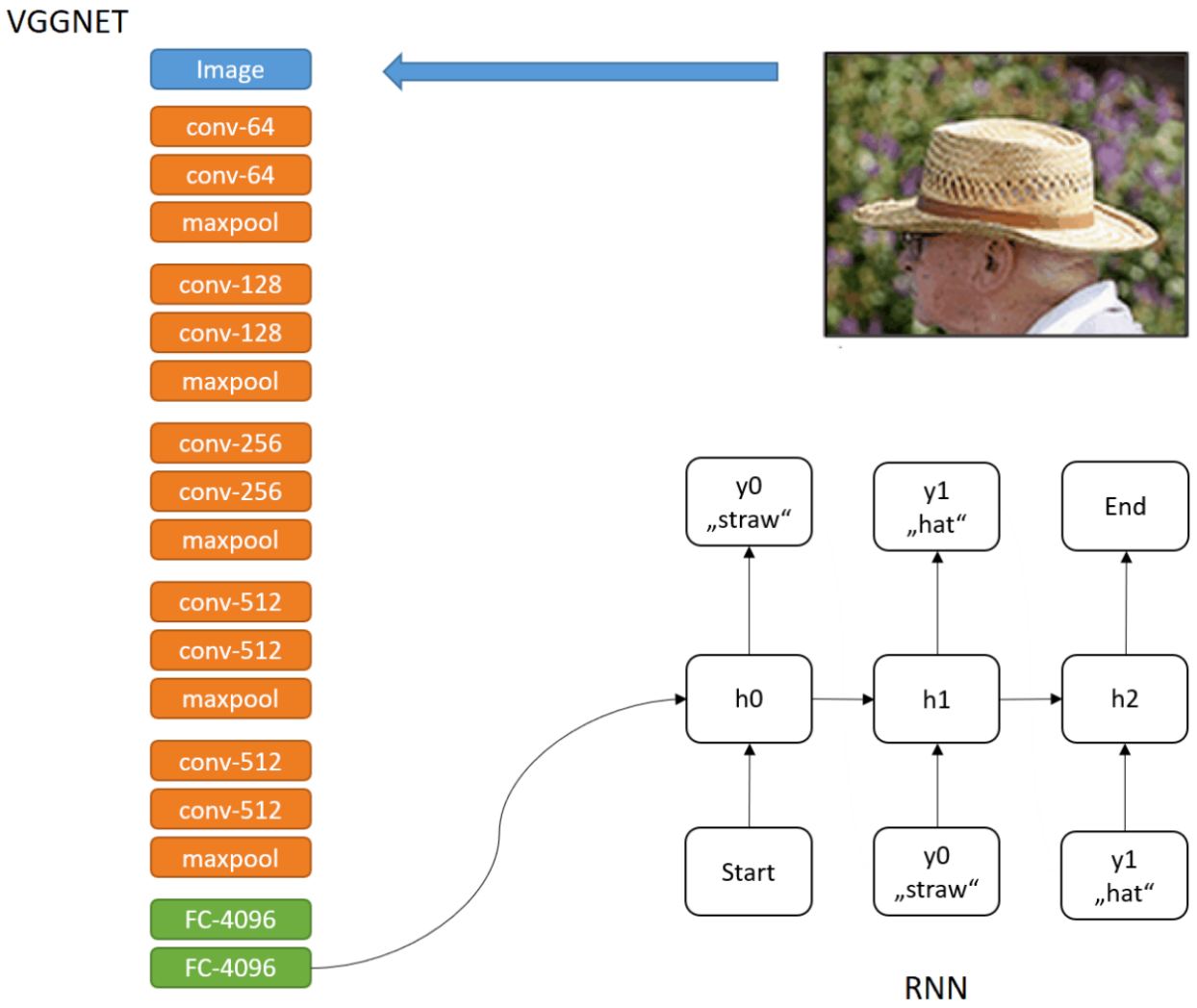


Image captioning (CNN encoder – RNN decoder)



https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN

Пример: распознавание рукописного текста

https://youtu.be/mLxsbWAYIpw

Ссылки

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf
- http://distill.pub/2016/augmented-rnns/
- Multilingual Neural Machine Translation https://arxiv.org/abs/1611.04558
- https://deepmind.com/blog/wavenet-generative-model-raw-audio/
- https://www.tensorflow.org/tutorials/recurrent
- https://www.tensorflow.org/tutorials/seq2seq