

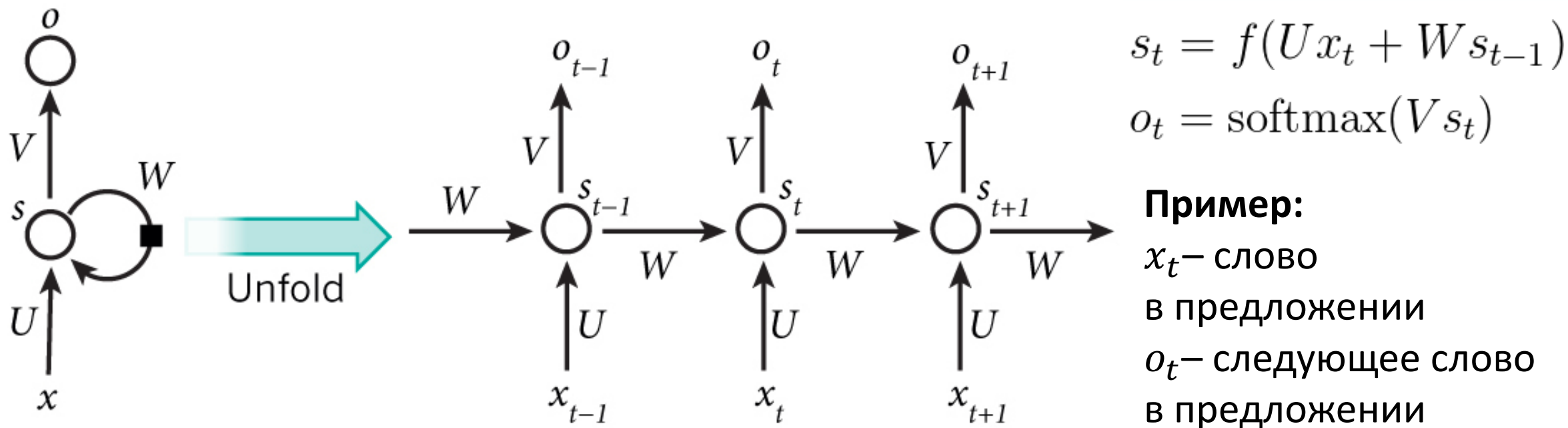
MML minor #7

Рекуррентные нейронные сети
(продолжение)

Recurrent Neural Networks (RNN)

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

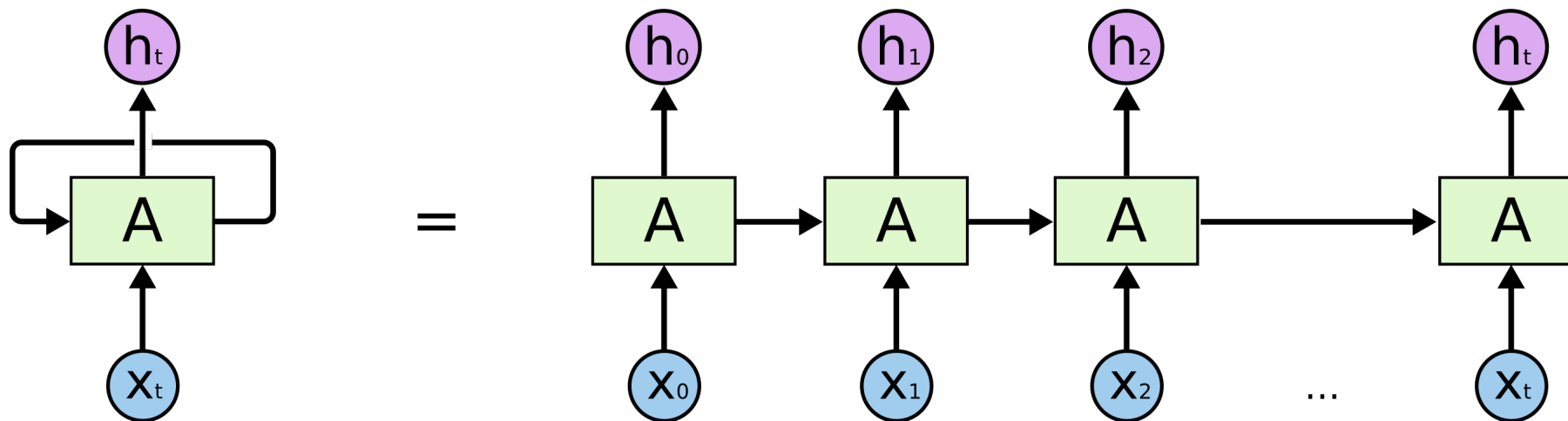
Как устроена простая RNN



- 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

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



Посчитаем производную для 2 шагов

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

a_2

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

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

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

Обозначение
в предположении,
что s_1 не зависит от W

Посчитаем производную для 3 шагов

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

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

$$\frac{\partial s_3}{\partial s_2}$$

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

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

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

Обозначение
в предположении,
что s_2 не зависит от W

Индукцией легко показать, что...

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

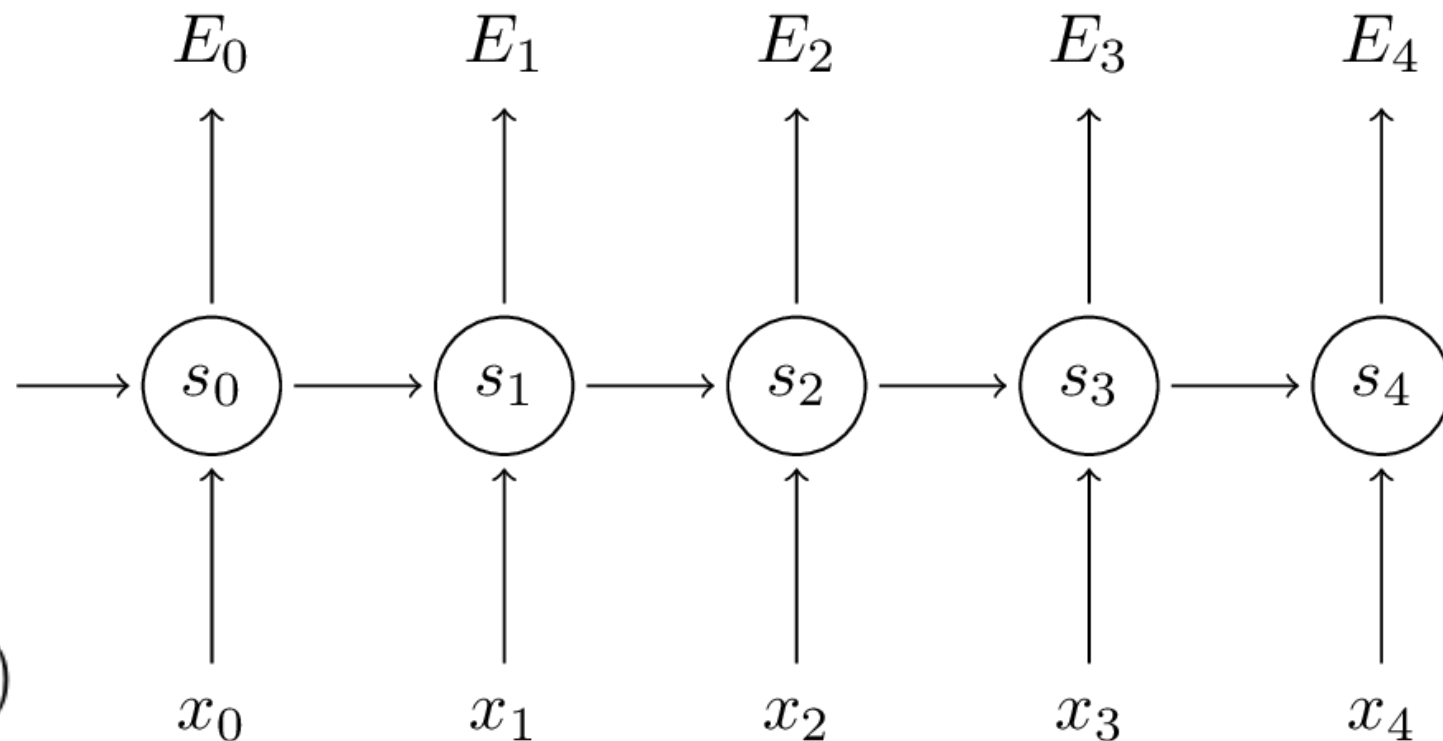
$$\frac{\partial s_3}{\partial W} = \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1} \frac{\partial s_1}{\partial W_*} + \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial W_*} + \frac{\partial s_3}{\partial W_*}$$

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

ВРТТ на примере

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

$$\hat{y}_t = \text{softmax}(Vs_t)$$



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

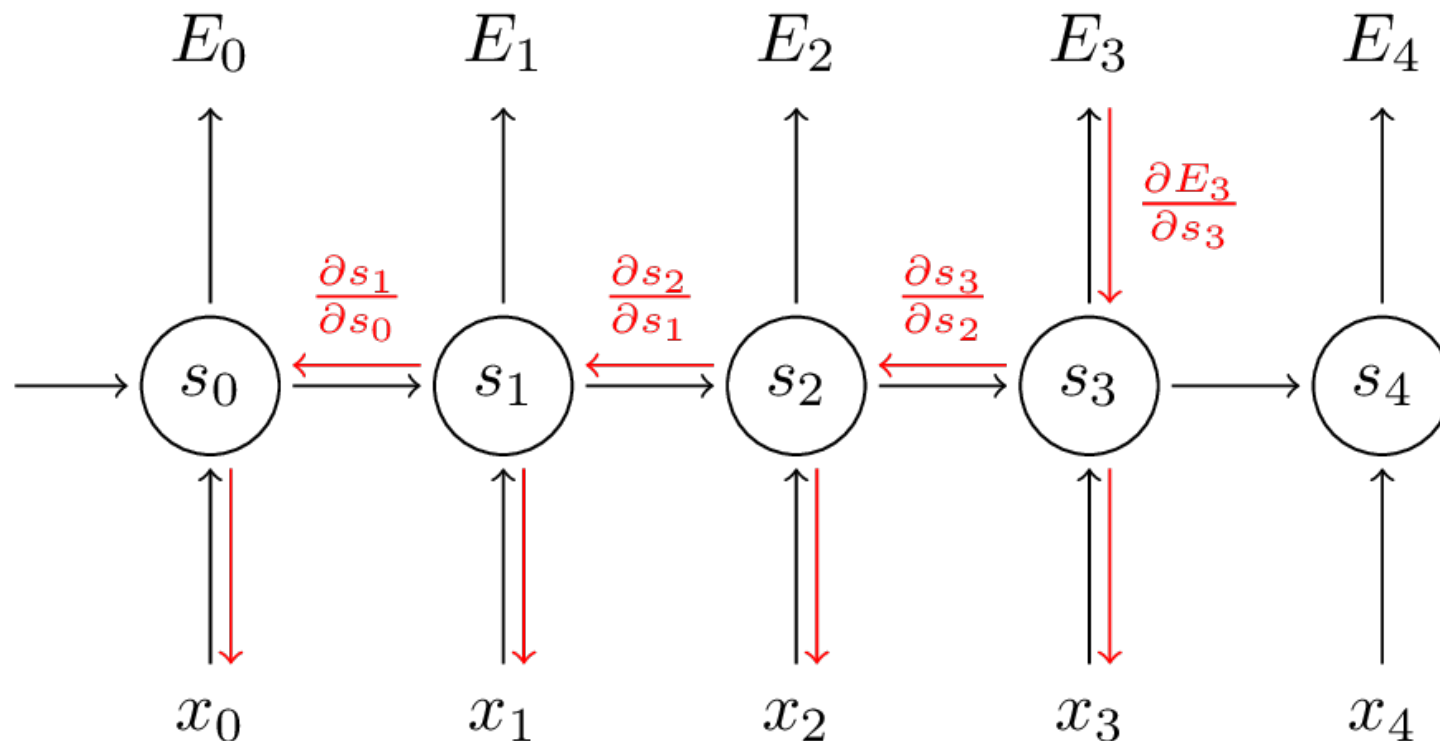
$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$

$$= - \sum_t y_t \log \hat{y}_t$$

BPTT на примере

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

$$\hat{y}_t = \text{softmax}(Vs_t)$$



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

Проблема с затухающими градиентами

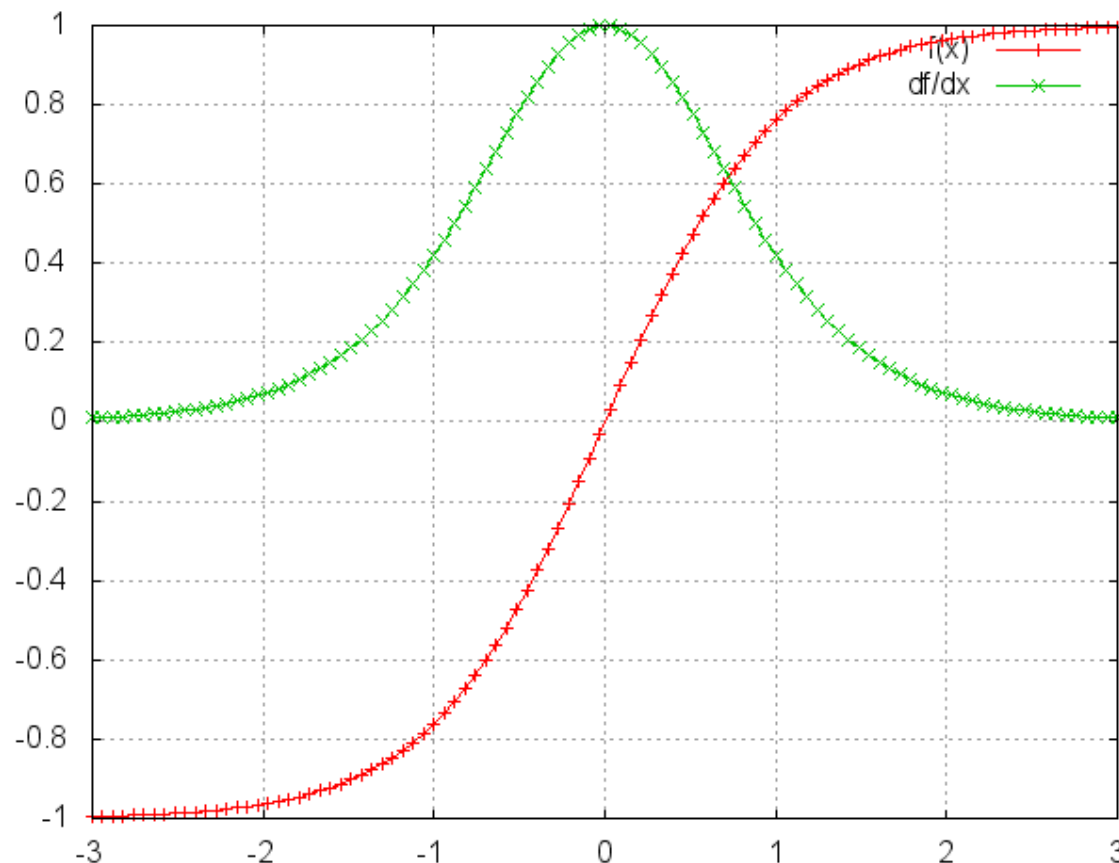
$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \sum_{i=1}^3 \left(\prod_{j=i+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_i}{\partial W_*}$$

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

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

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

Быстро убывает
или взрывается



Gradient clipping

- Градиенты могут взорваться – ограничим их норму

```

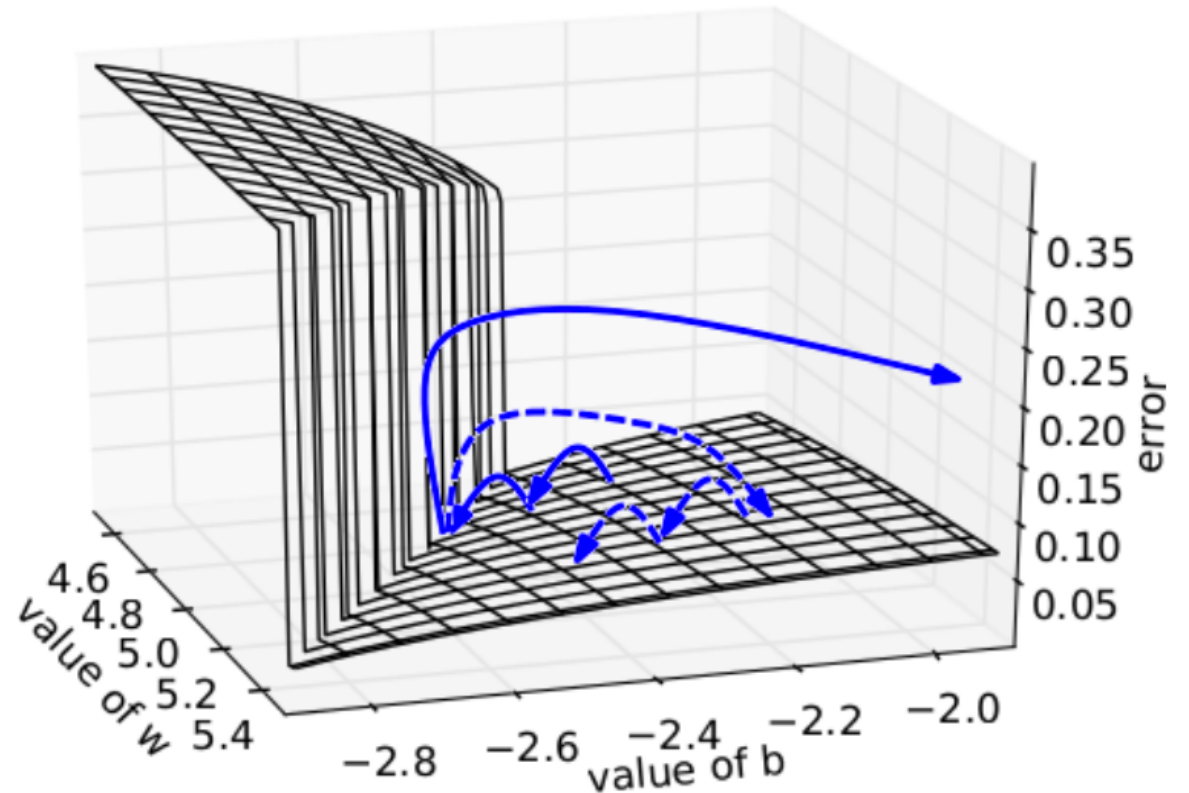
$$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$$


if  $\|\hat{\mathbf{g}}\| \geq threshold$  then

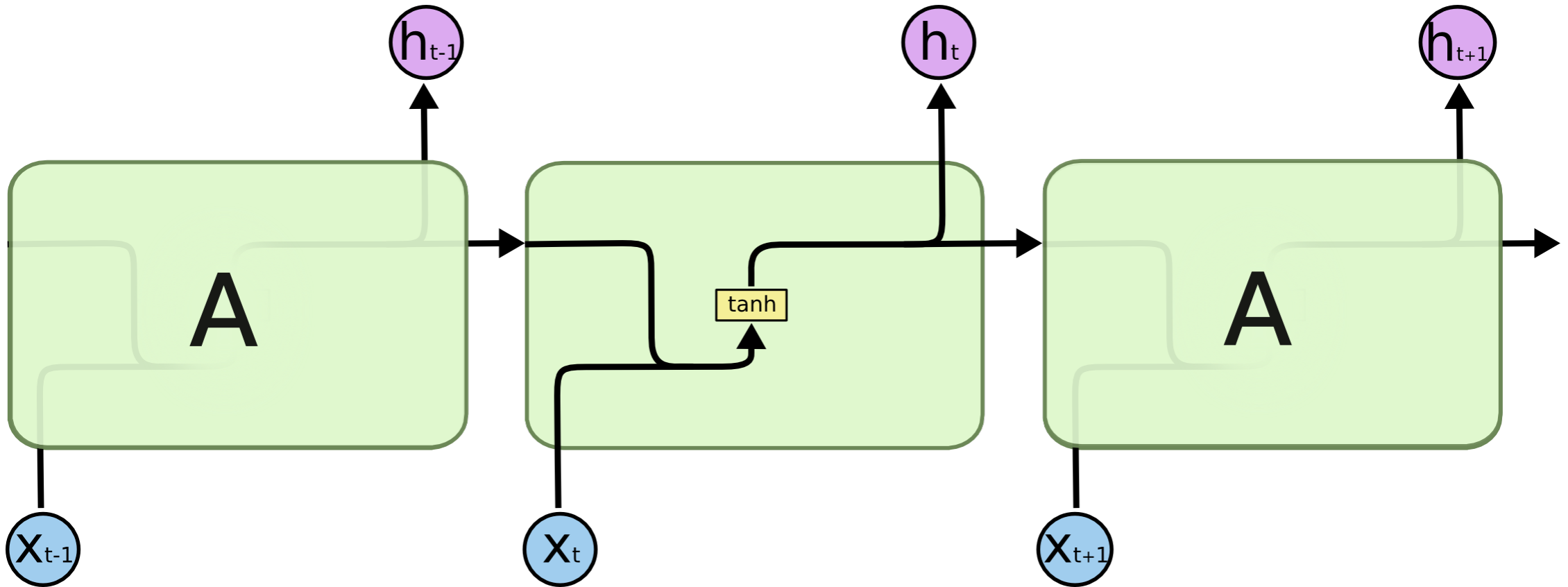

$$\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$$


end if


```

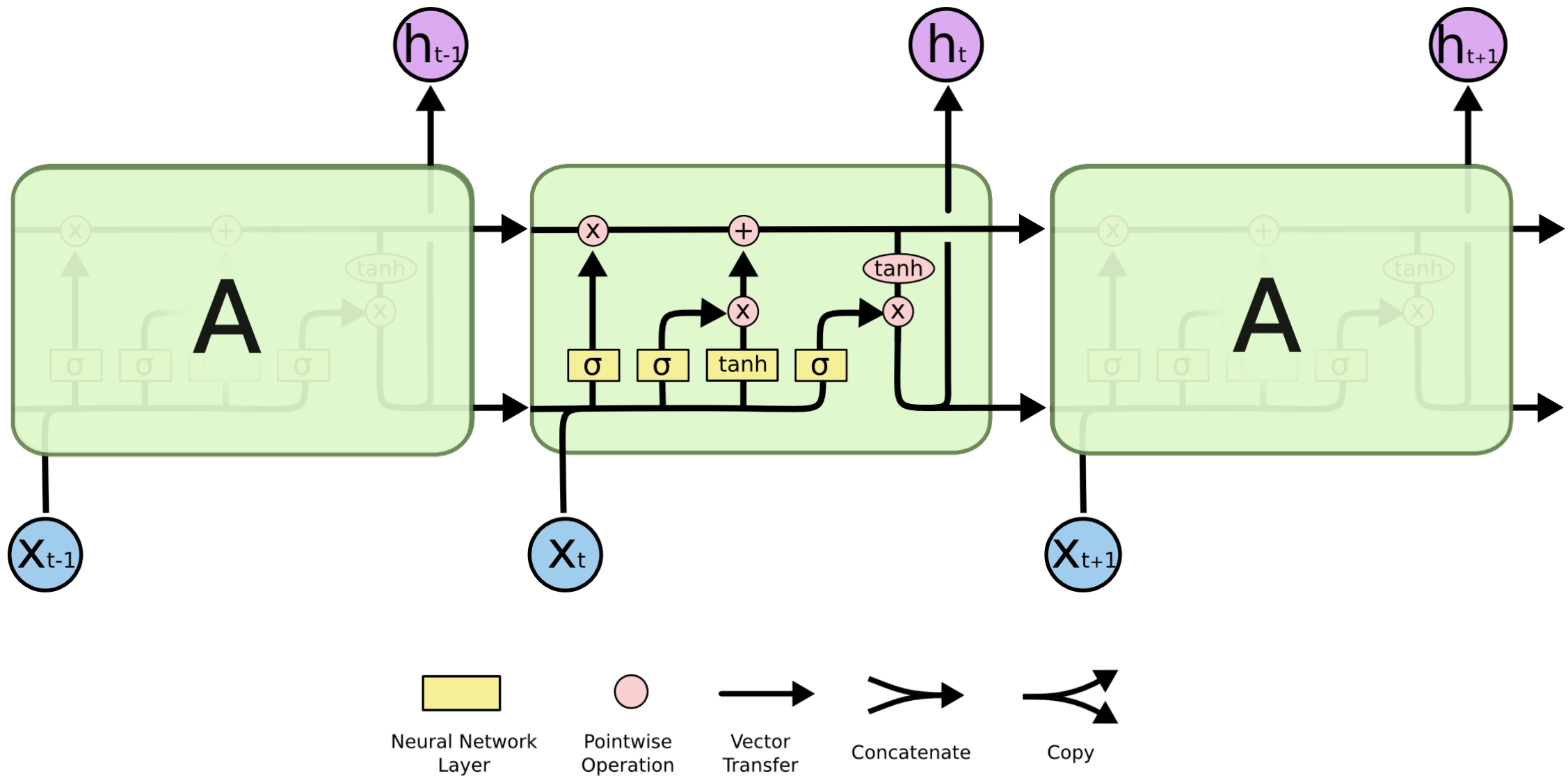


Простая RNN плохо работает



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

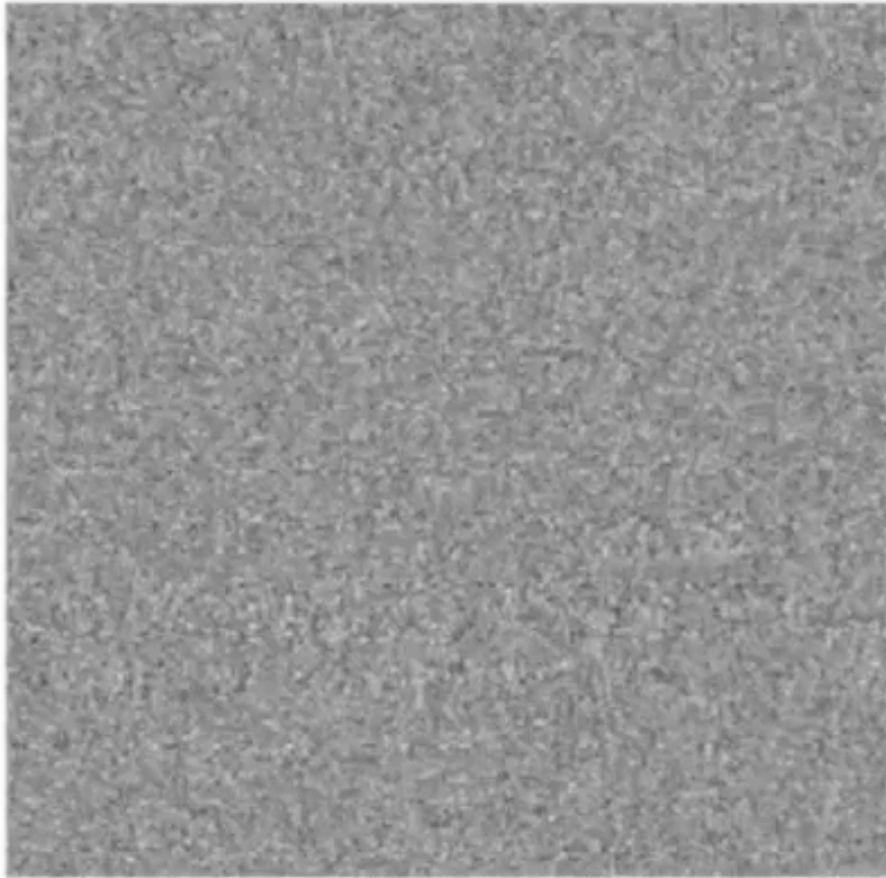
Спасет LSTM



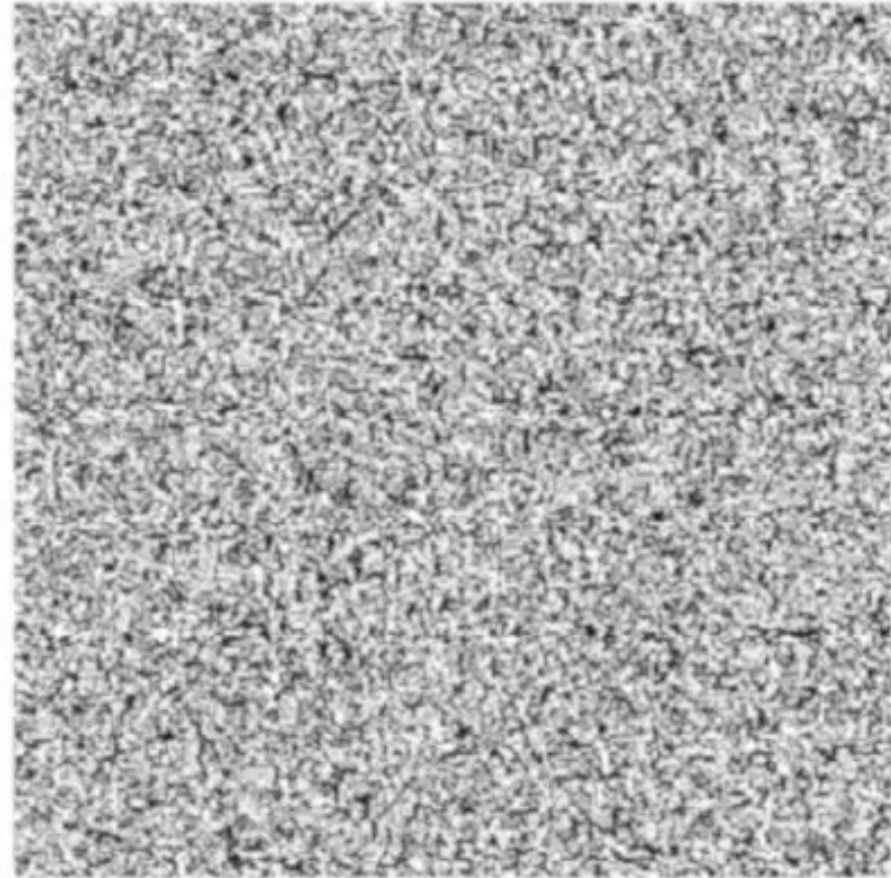
LSTM решает проблему затухания градиентов

Распространение ошибки на 128 шаге

127

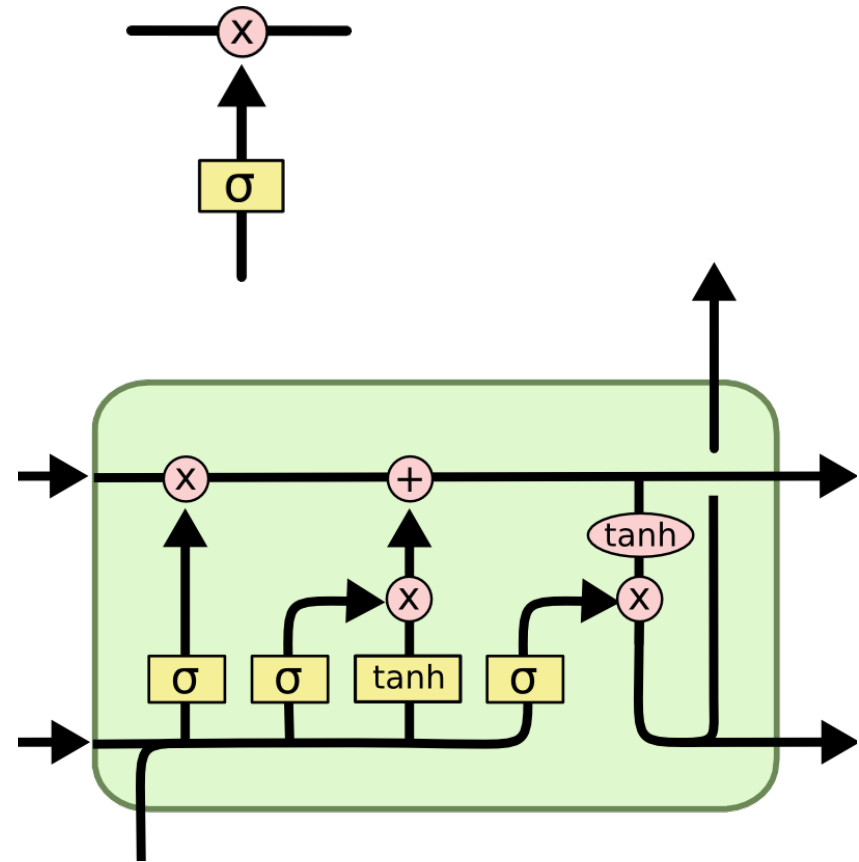
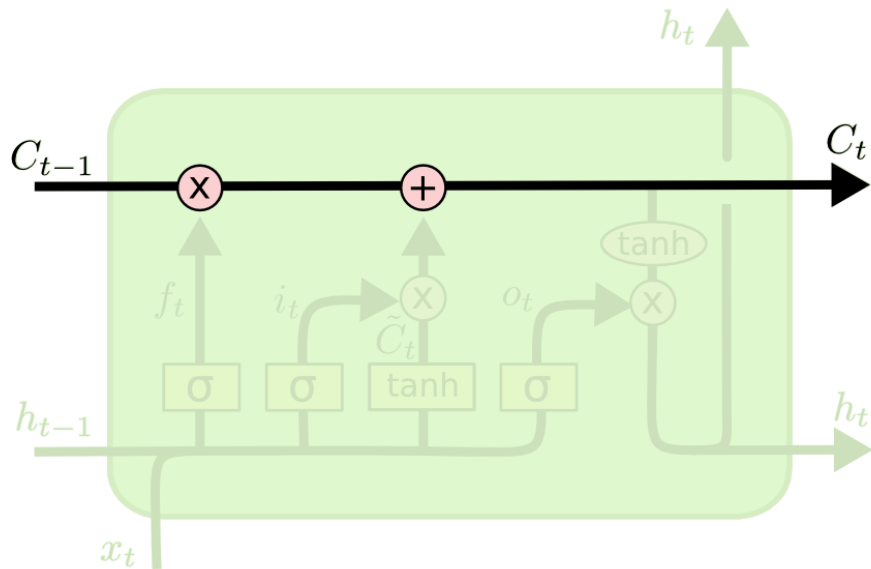


127

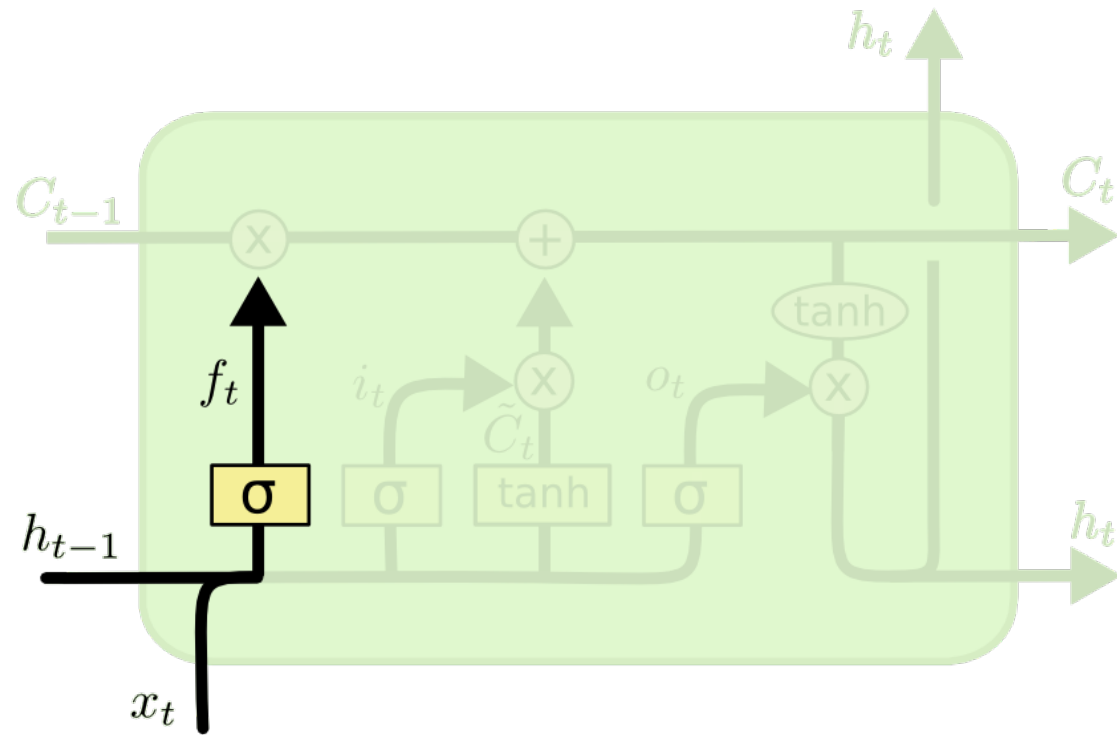


LSTM cell state

- 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

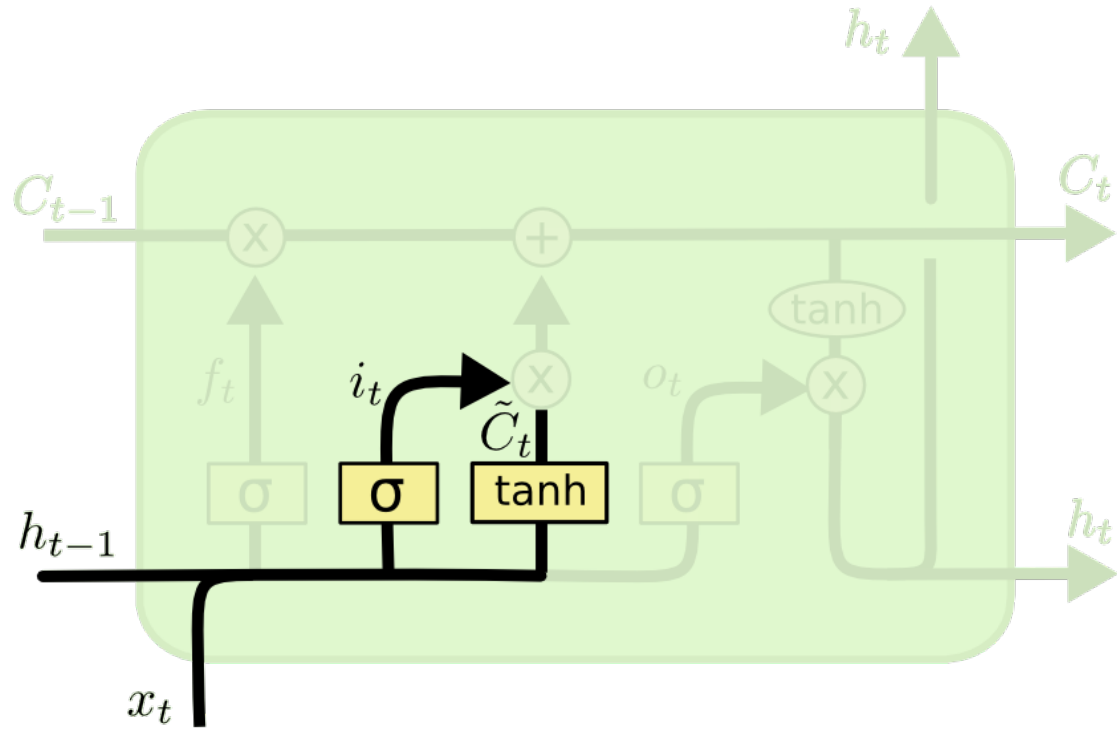


LSTM forget gate



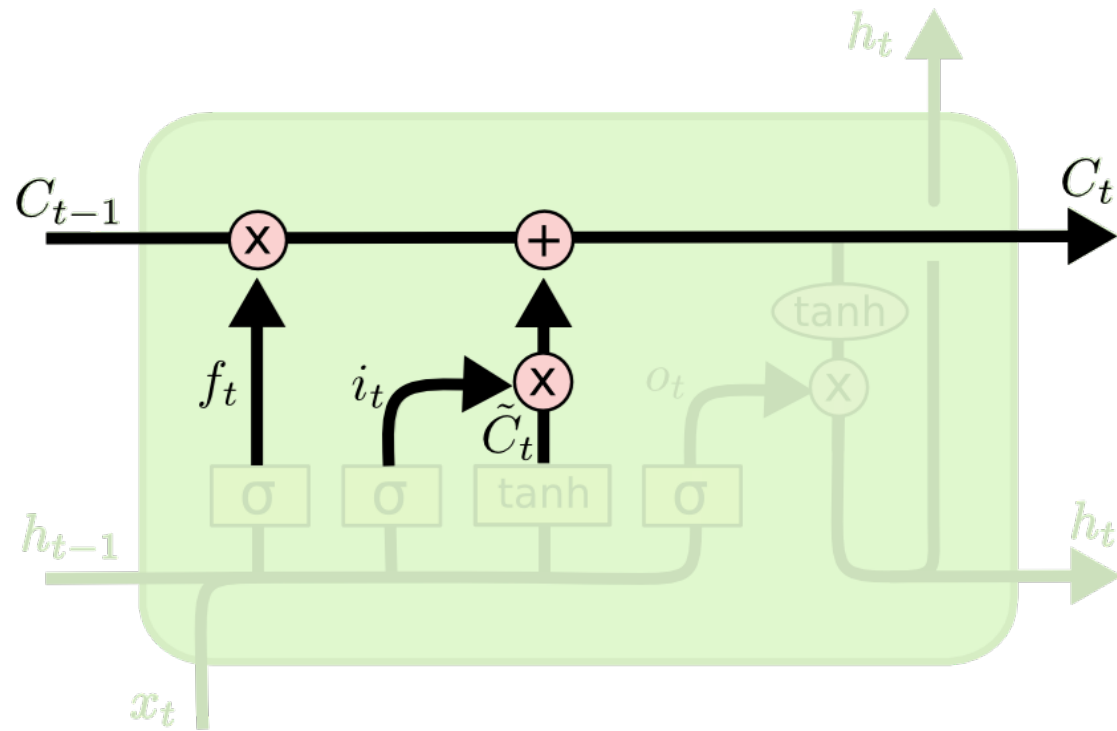
$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM input gate и вклад ячейки в состояние



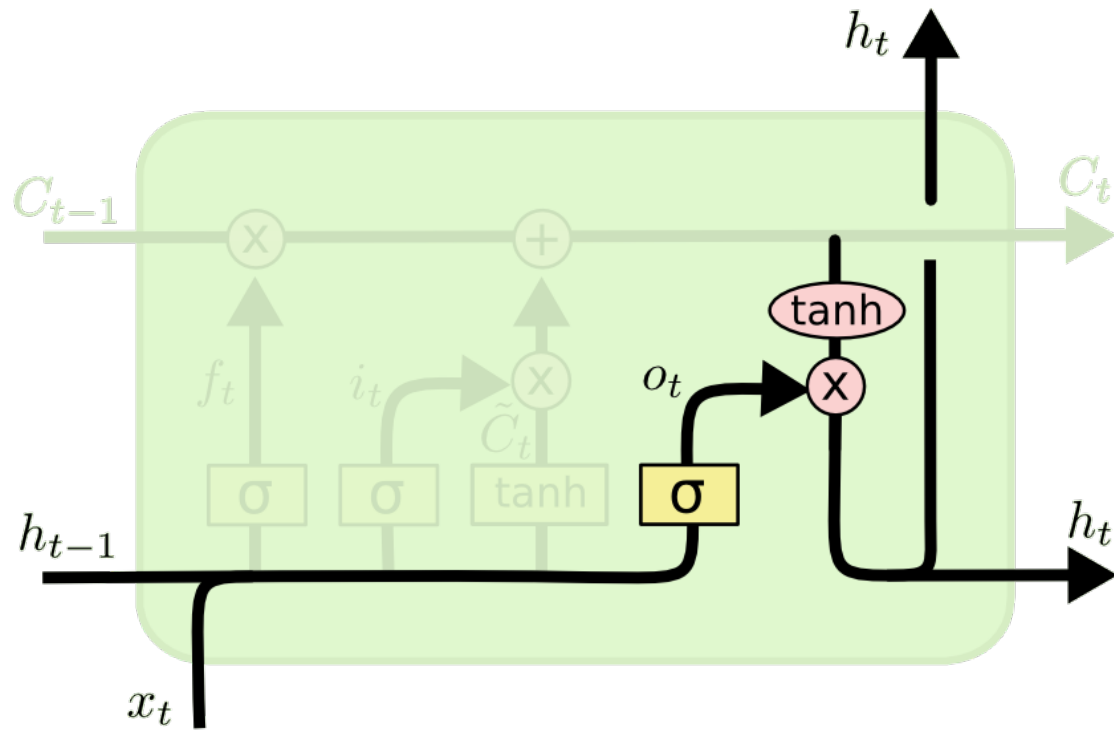
$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Обновление состояния



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

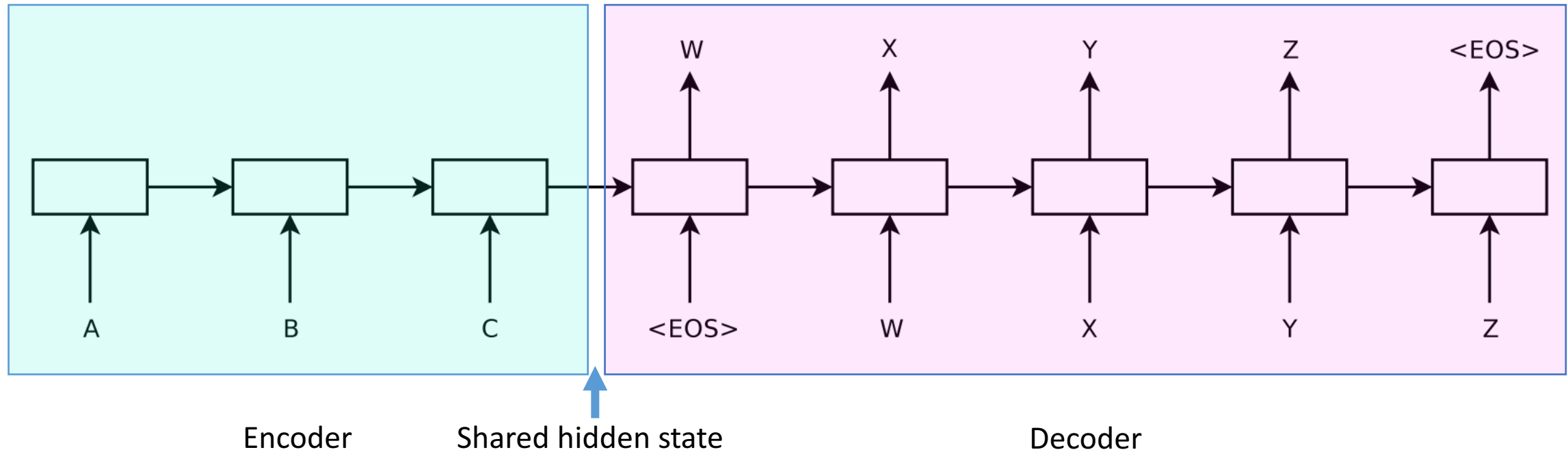
Выдача скрытого состояния наружу



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

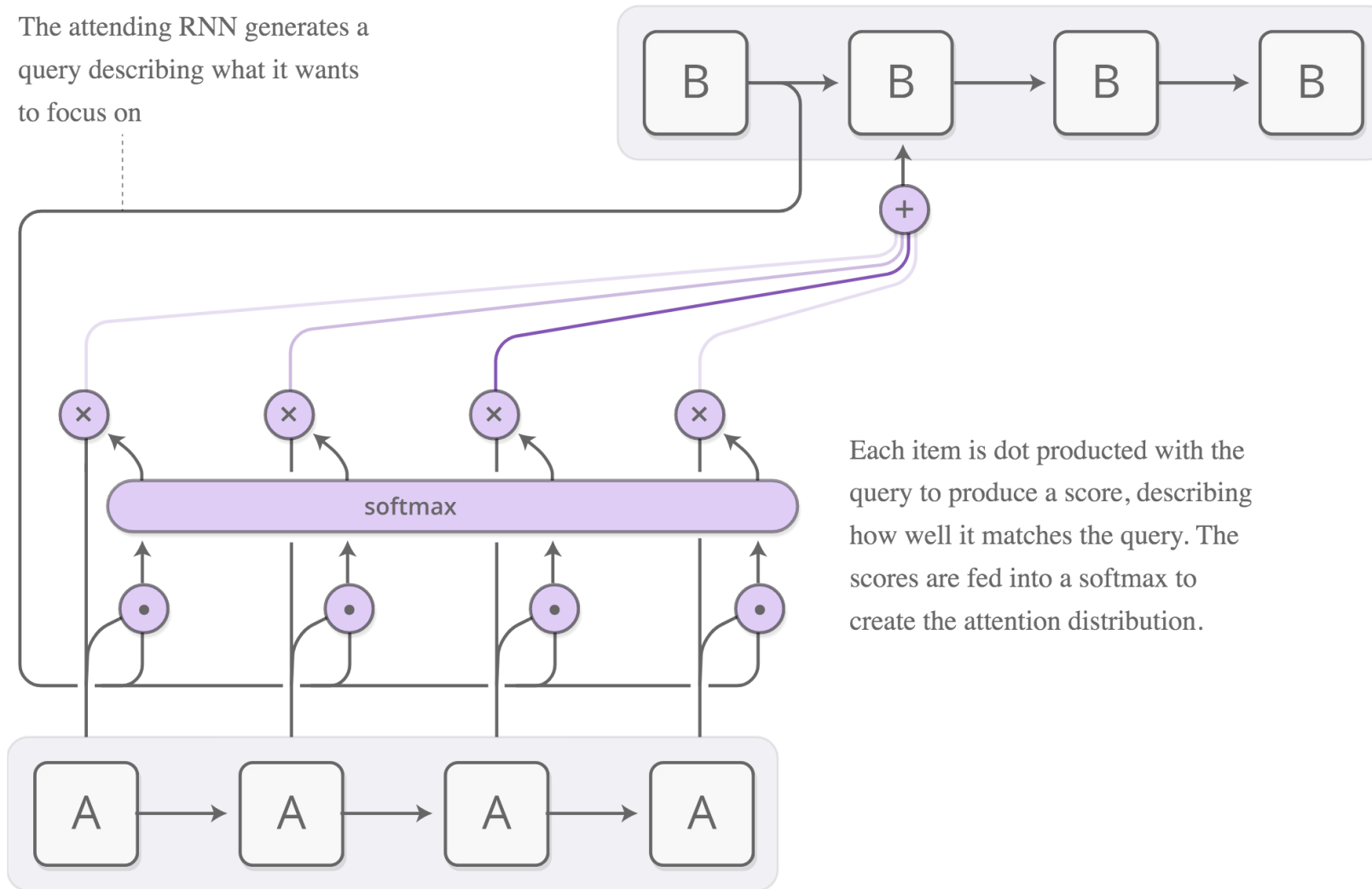
$$h_t = o_t * \tanh (C_t)$$

Encoder-decoder (seq2seq)

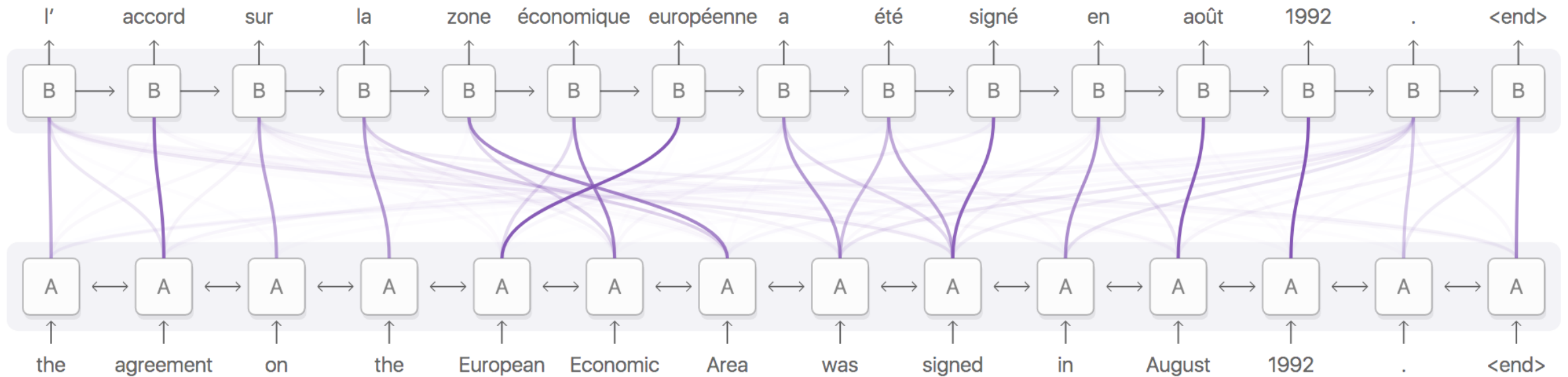


- Слабое место – связь между encoder и decoder только через последний скрытый слой encoder

Поможет attention

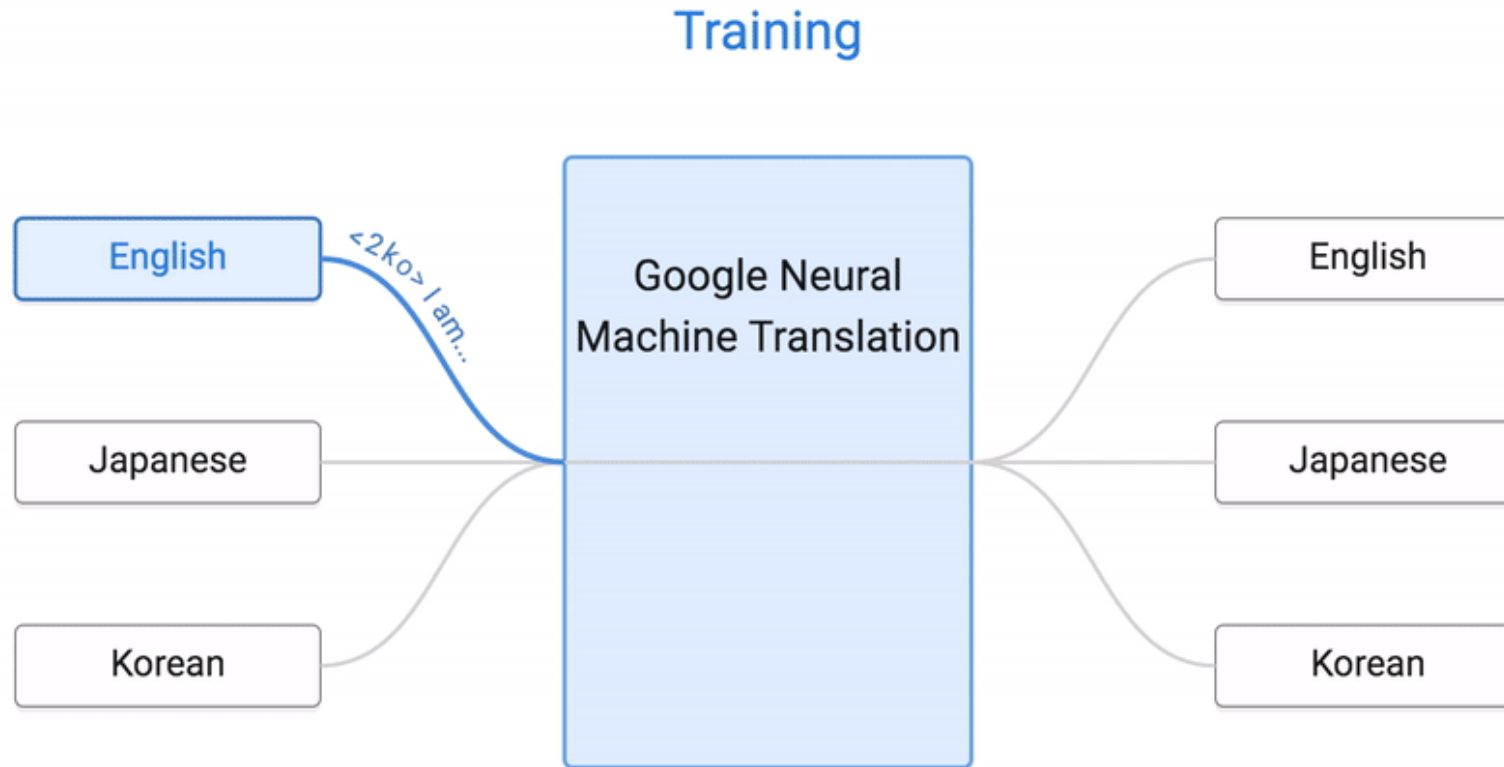


Машинный перевод (seq2seq + attention)



Multi-language перевод от Google

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



Распознавание голоса (seq2seq + attention)

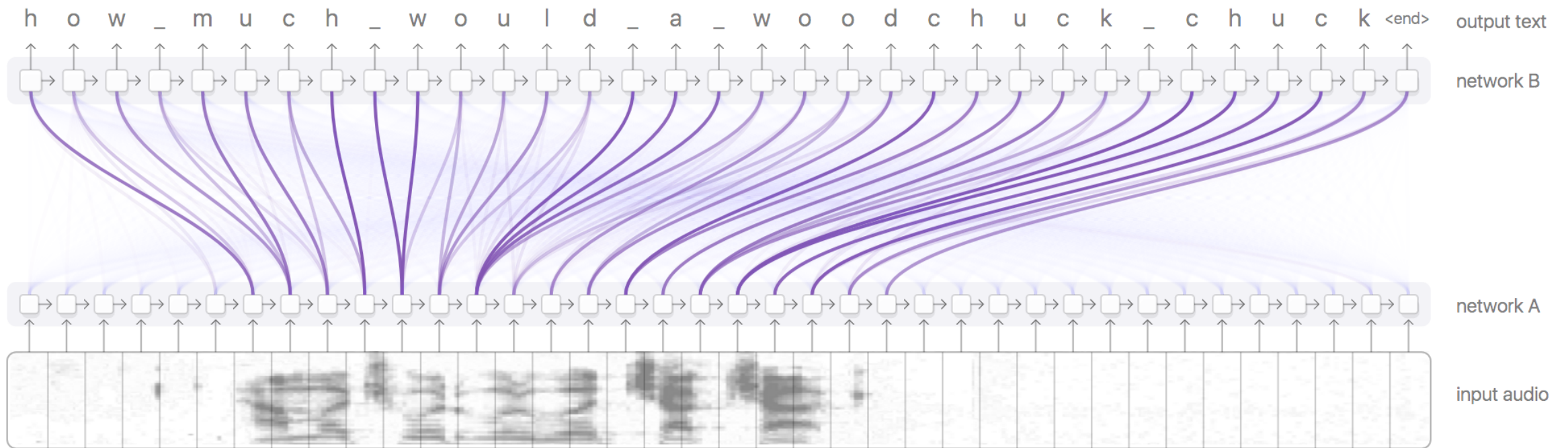
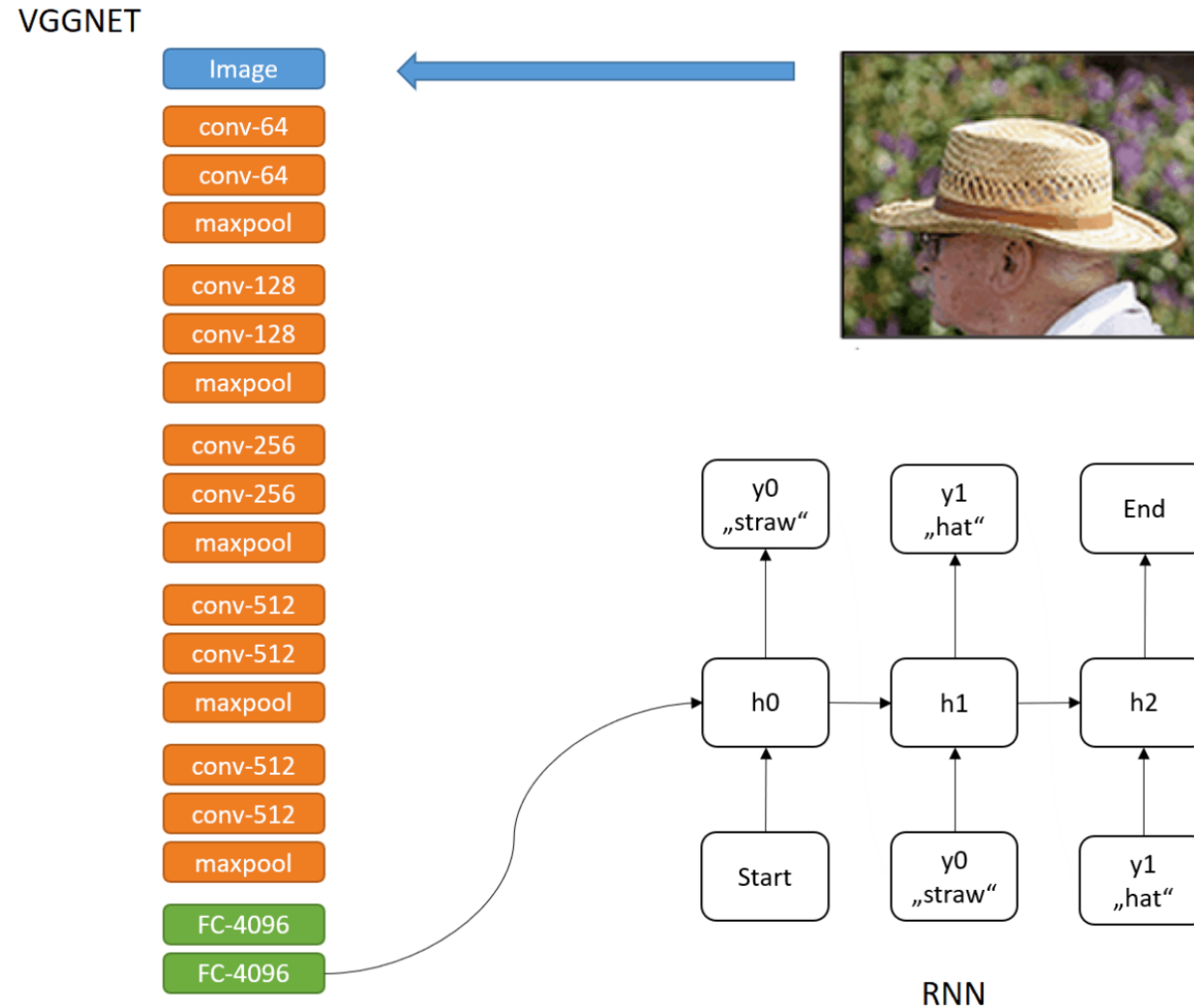
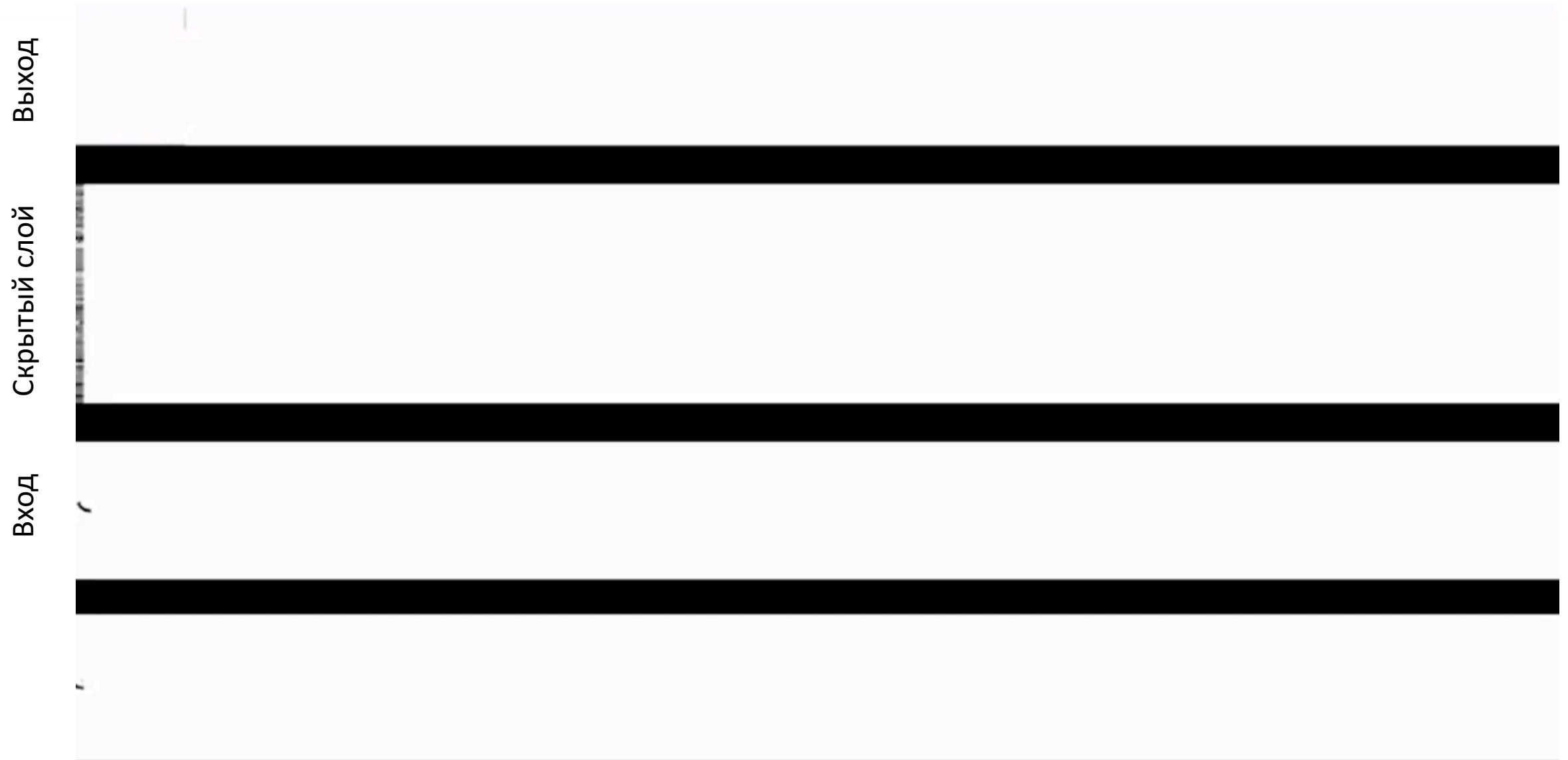


Image captioning (CNN encoder – RNN decoder)



Пример: распознавание рукописного текста



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>