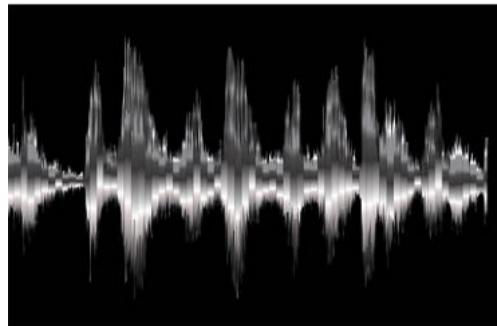
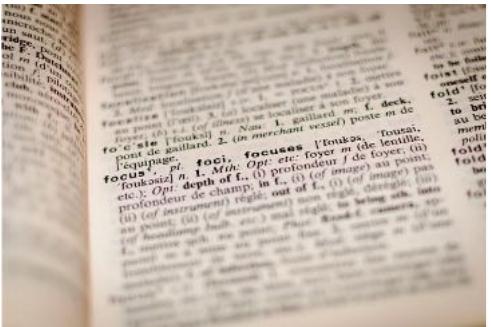


MML minor #6

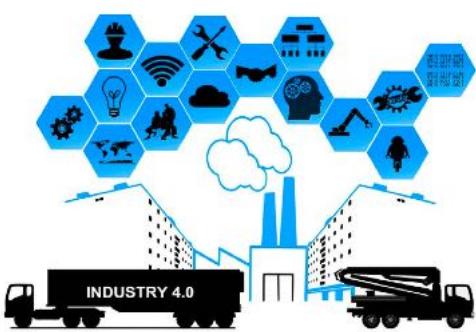
Рекуррентные нейронные сети

Последовательности повсюду

Текст, видео, аудио



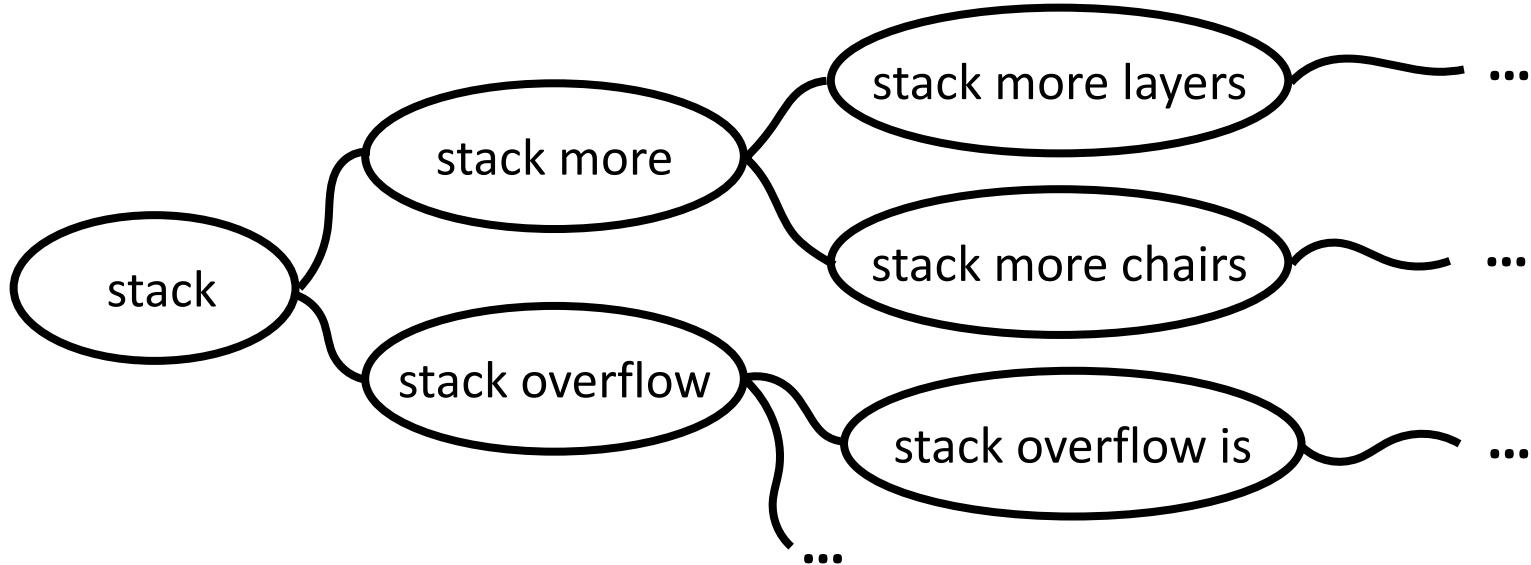
Временные ряды: финансы, медицина



Держим в голове пример: Language Model

Генеративная модель естественного языка:

$$\begin{aligned} P(\text{text}) &= P(x_0, \dots, x_n) = \\ &= P(x_0)P(x_1|x_0)P(x_2|x_0, x_1) \dots P(x_n| \dots) \end{aligned}$$



Можем семплировать предложения!

Зачем нужна модель языка?

- Машинный перевод (превратить смысл в текст)
- Чат-боты
- Распознавание речи (знаем какое следующее слово наиболее вероятно)
- ...

Как подать слово на вход нейросети

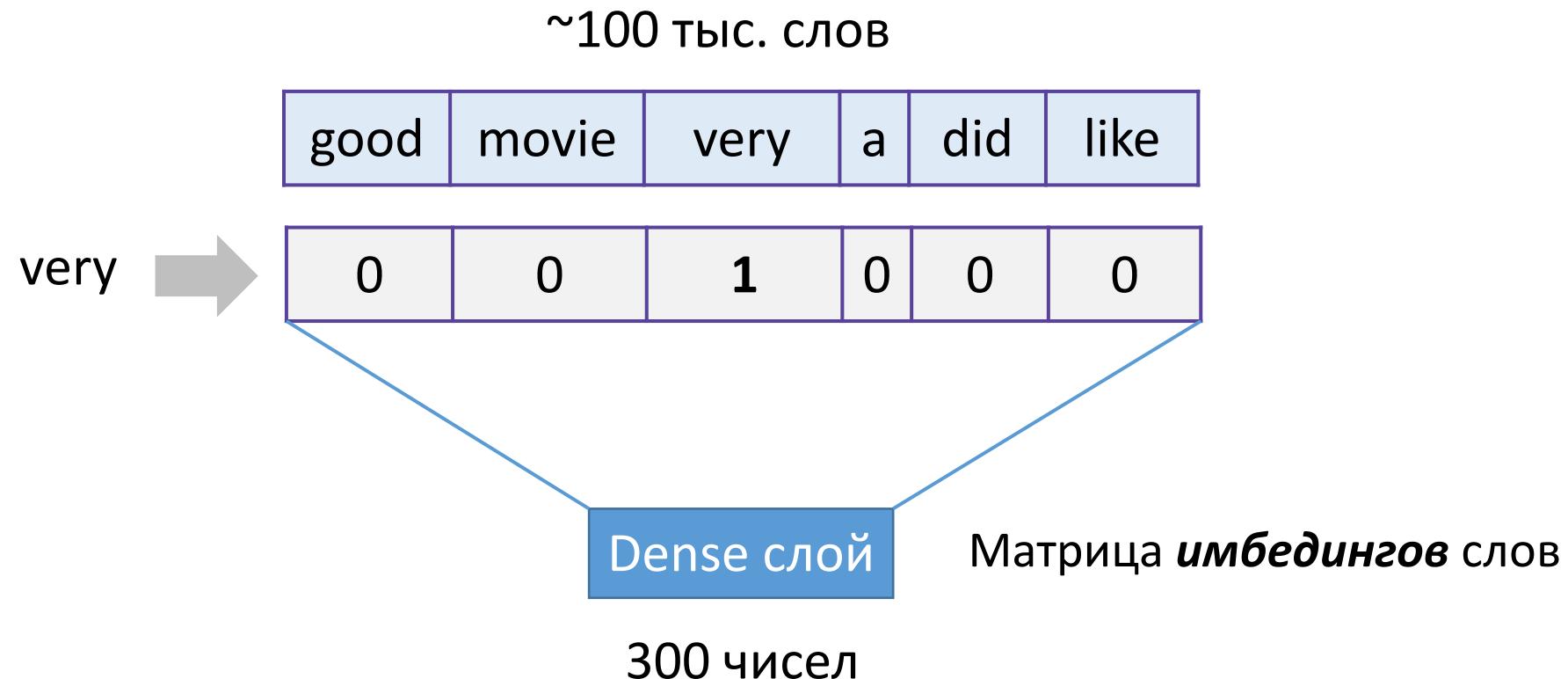
- One-hot кодирование



В чем проблема?

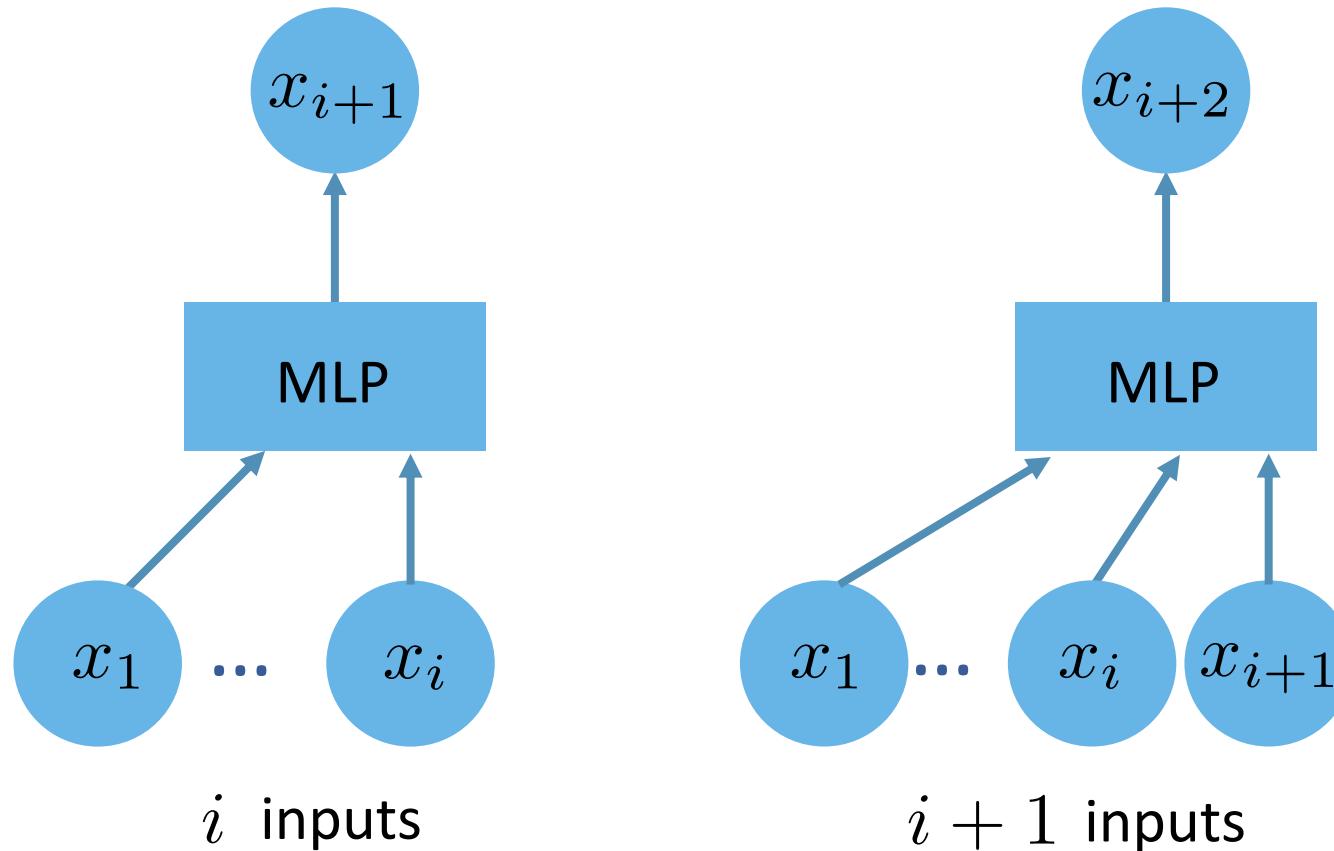
Как подать слово на вход нейросети

- Надо уменьшить размерность!



Почему бы не использовать MLP?

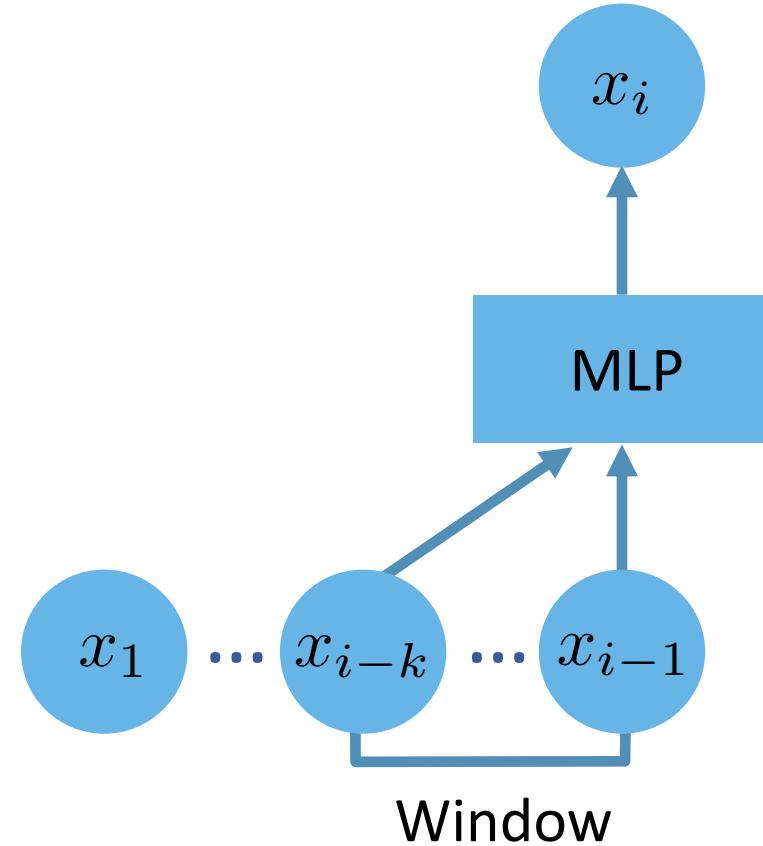
Основная проблема в разной длине входов:



Как с ней справиться?

Почему бы не использовать MLP?

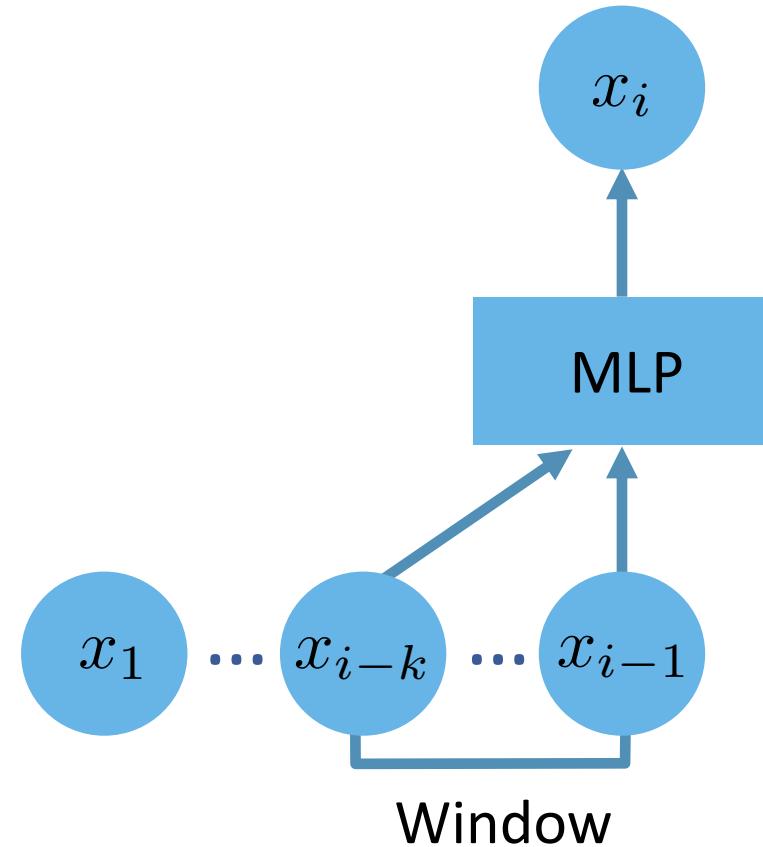
Можем использовать окно
фиксированного размера!



Почему бы не использовать MLP?

Можем использовать окно
фиксированного размера!

- Не понятно какого размера окно брать!
- Слишком много параметров в случае большого окна!

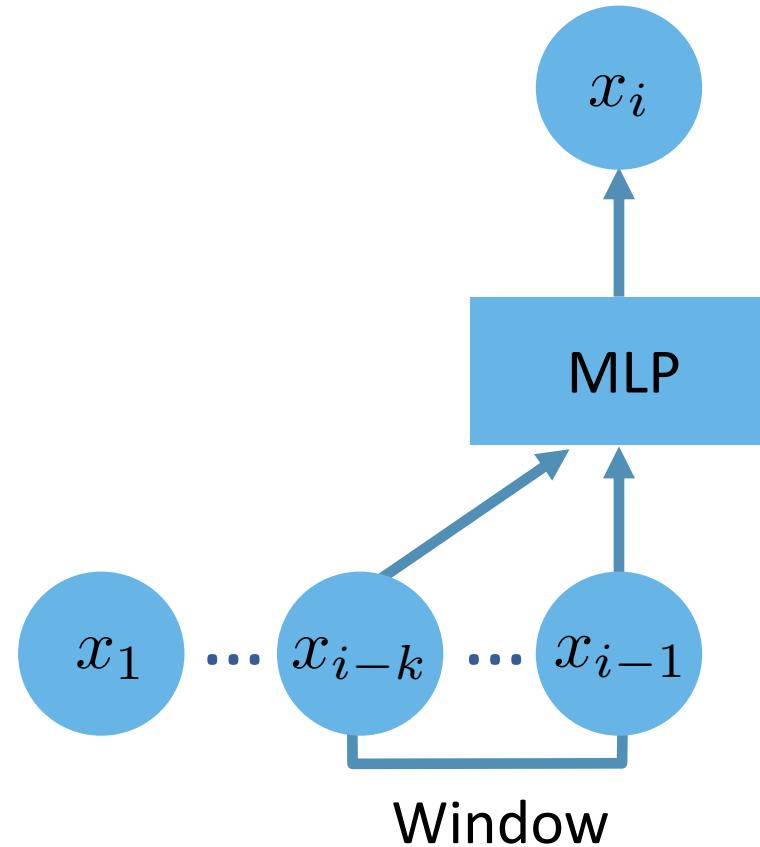


Почему бы не использовать MLP?

Можем использовать окно
фиксированного размера!

**Сколько параметров нужно
для первого скрытого слоя
MLP?**

- hidden neurons: 100
- window width: 100
- word embedding size: 100



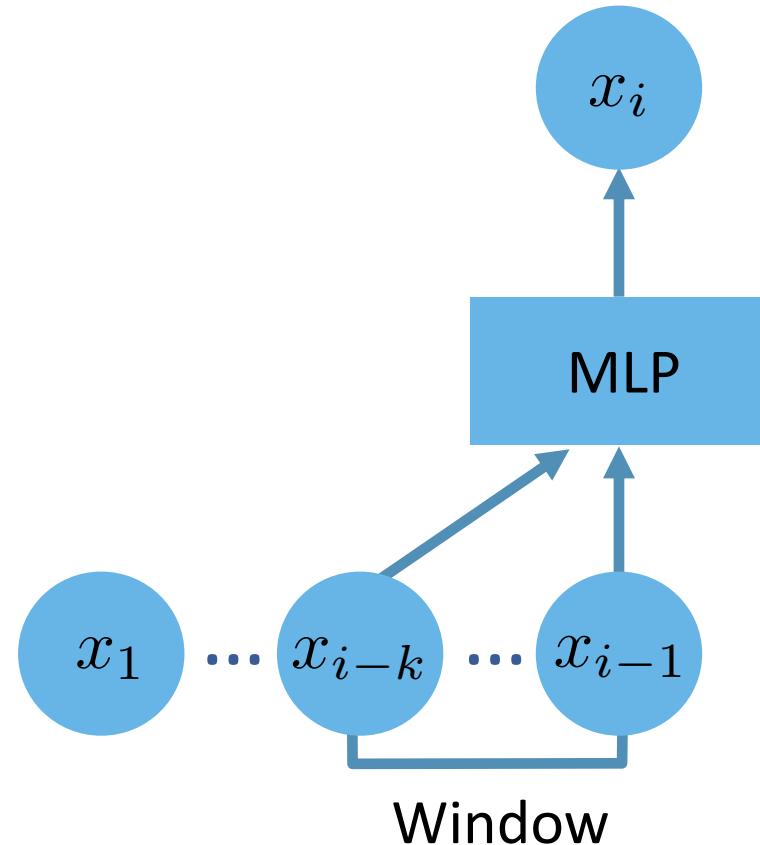
Почему бы не использовать MLP?

Можем использовать окно
фиксированного размера!

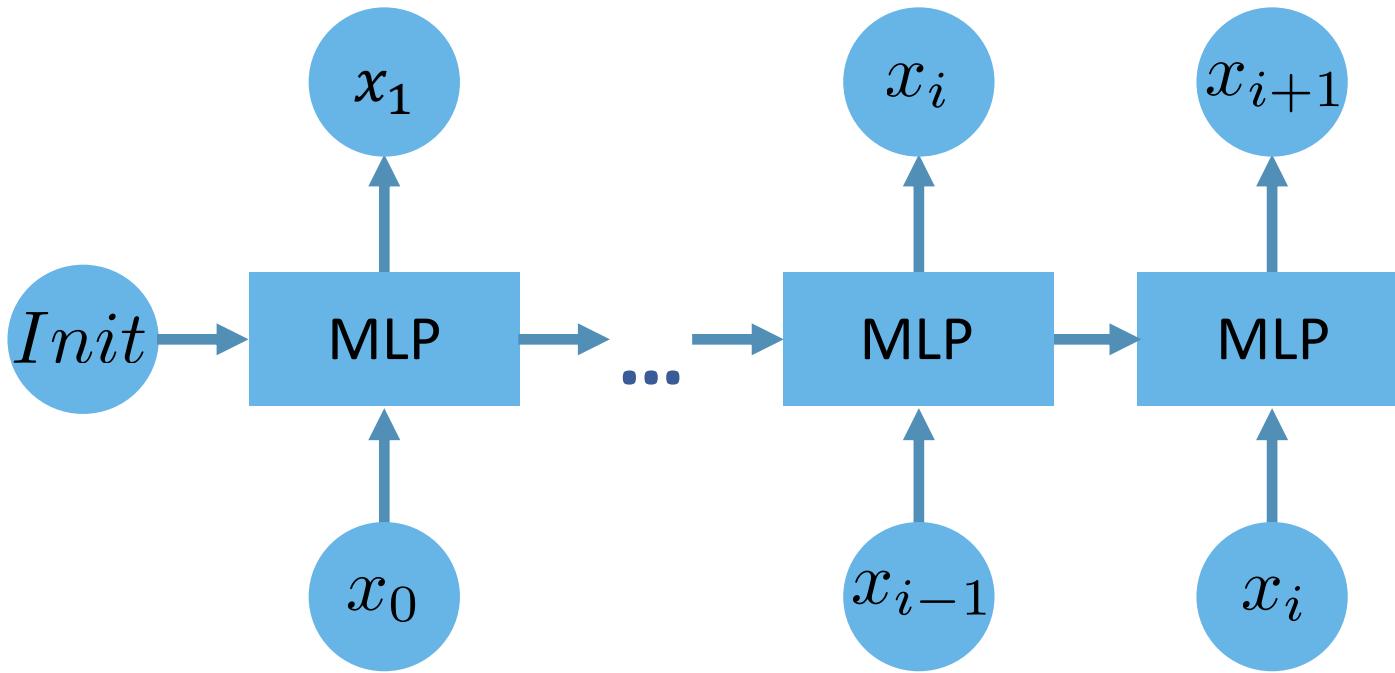
**Сколько параметров нужно
для первого скрытого слоя
MLP?**

- hidden neurons: 100
- window width: 100
- word embedding size: 100

Больше миллиона!

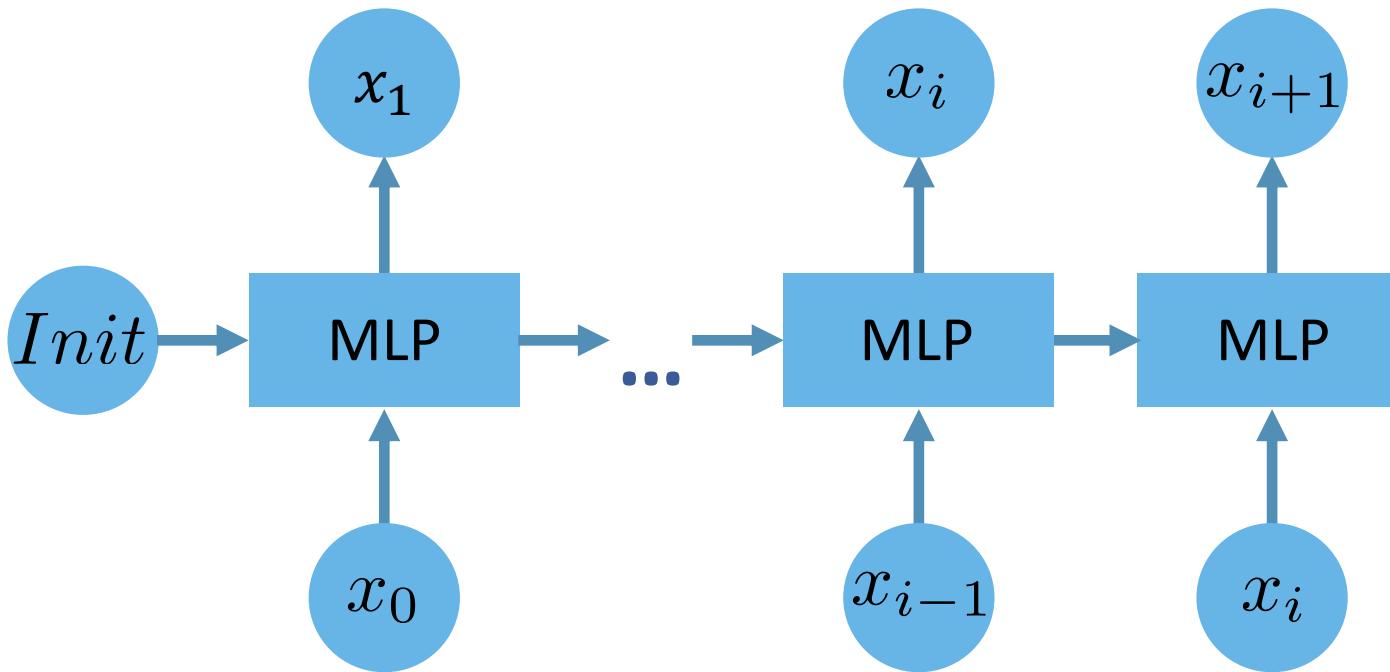


Рекуррентная архитектура (RNN)



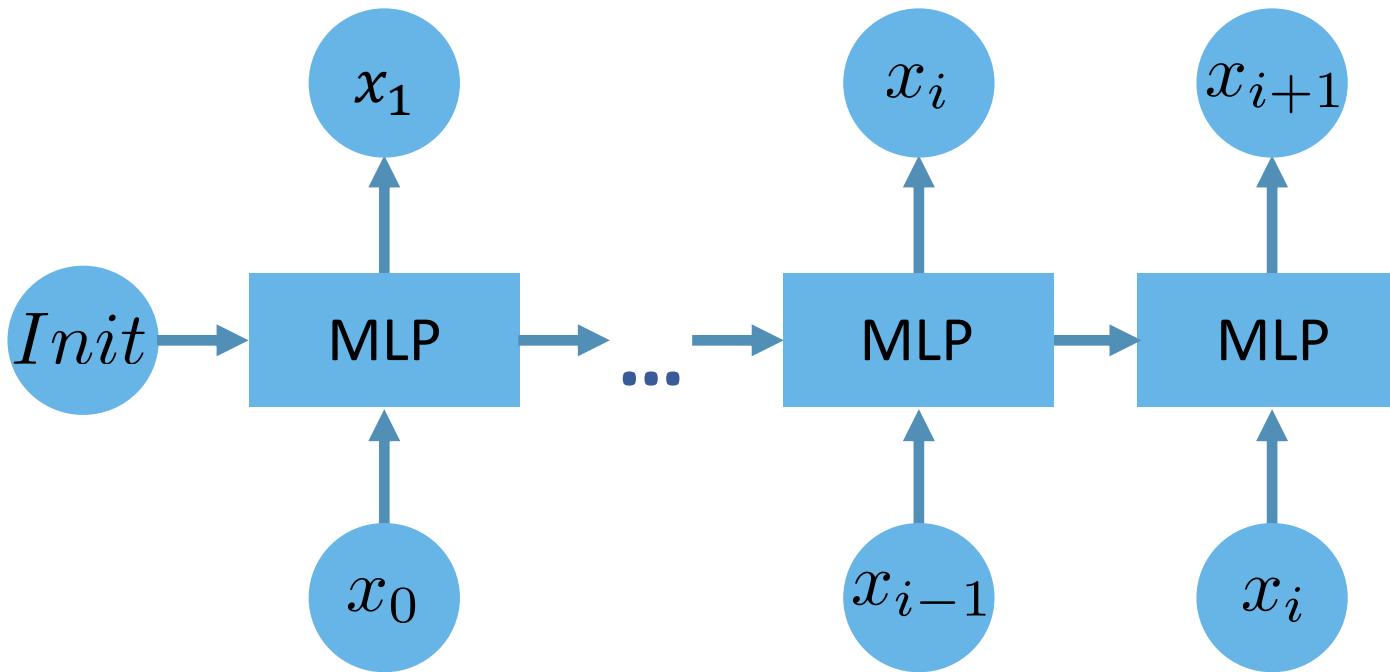
- Фиксированное количество входов у MLP!

Рекуррентная архитектура (RNN)



- В каждый момент времени используются одни и те же параметры MLP!

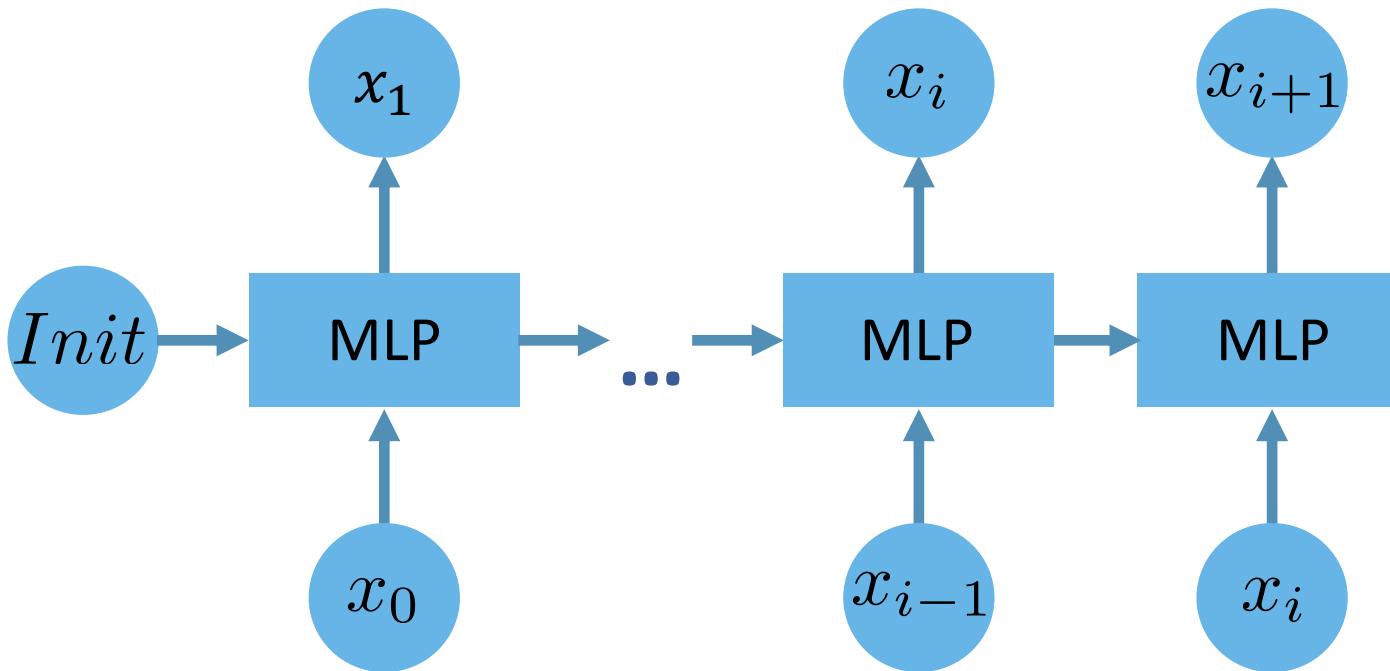
Рекуррентная архитектура (RNN)



Сколько параметров нужно для первого скрытого слоя MLP?

- hidden neurons: 100
- word embedding size: 100

Рекуррентная архитектура (RNN)

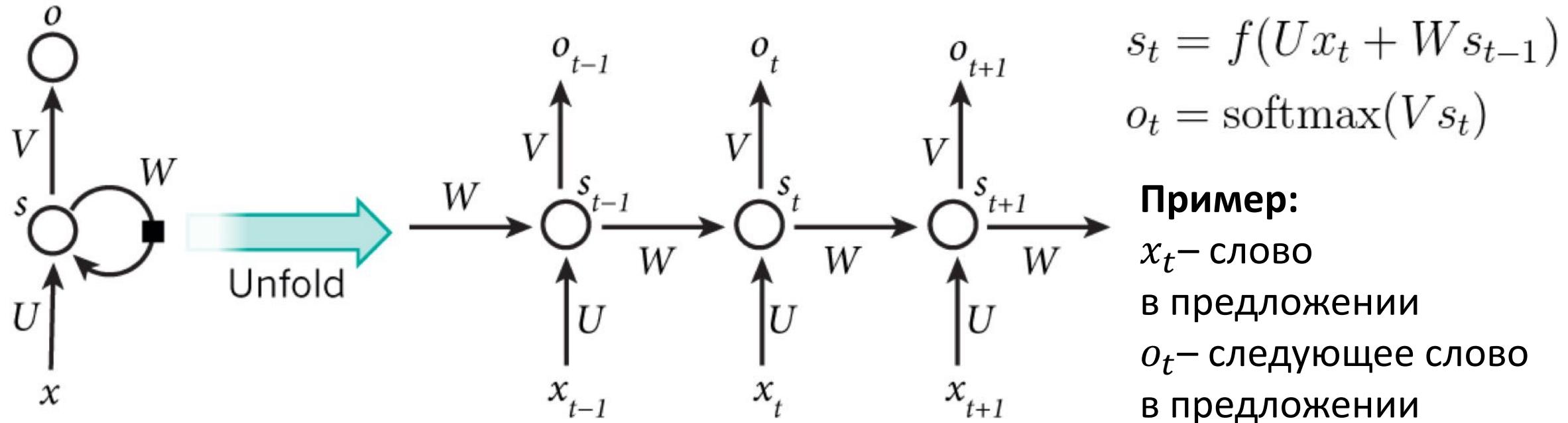


Сколько параметров нужно для первого скрытого слоя MLP?

- hidden neurons: 100
- word embedding size: 100

Всего 20100!

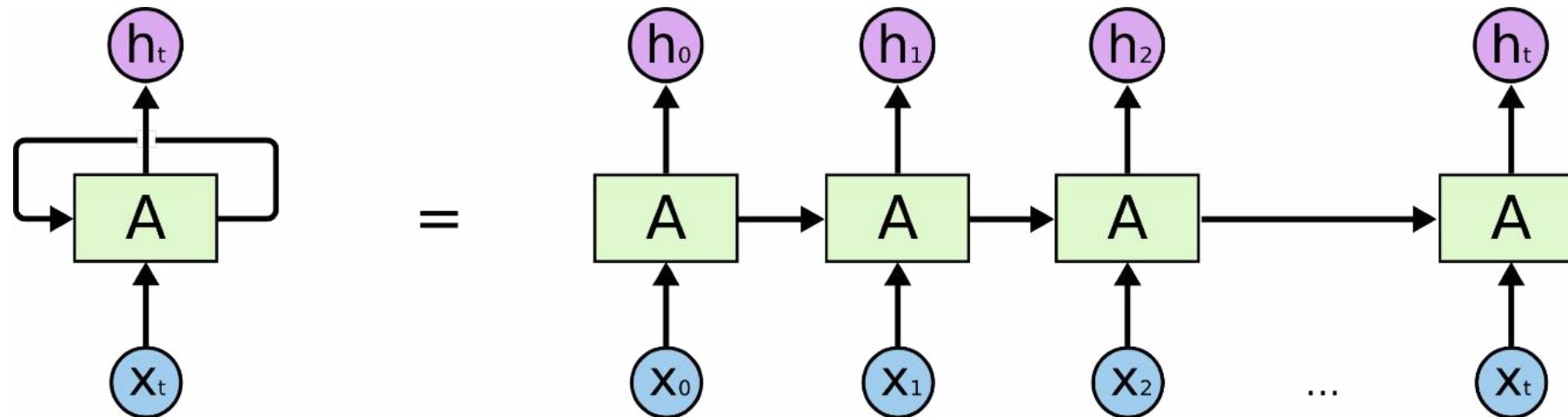
RNN для LM



- 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 свертоек CNN) суммируем градиенты по shared параметрам и делаем шаг SGD



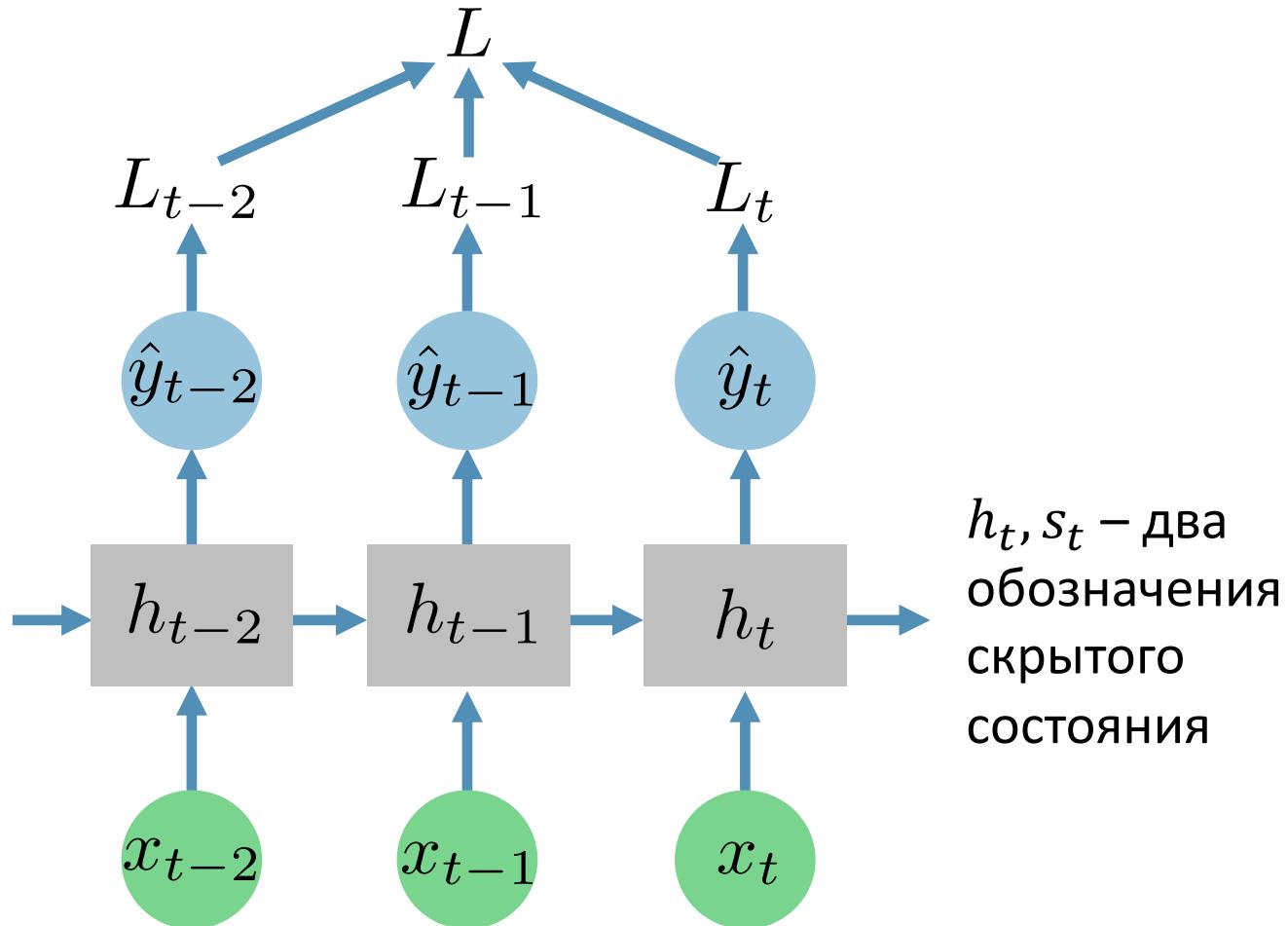
ВРТТ

На каждом шаге:

- y_t - правильный ответ
- \hat{y}_t - предсказание
- $L_t(y_t, \hat{y}_t)$ - потери

Суммарные потери:

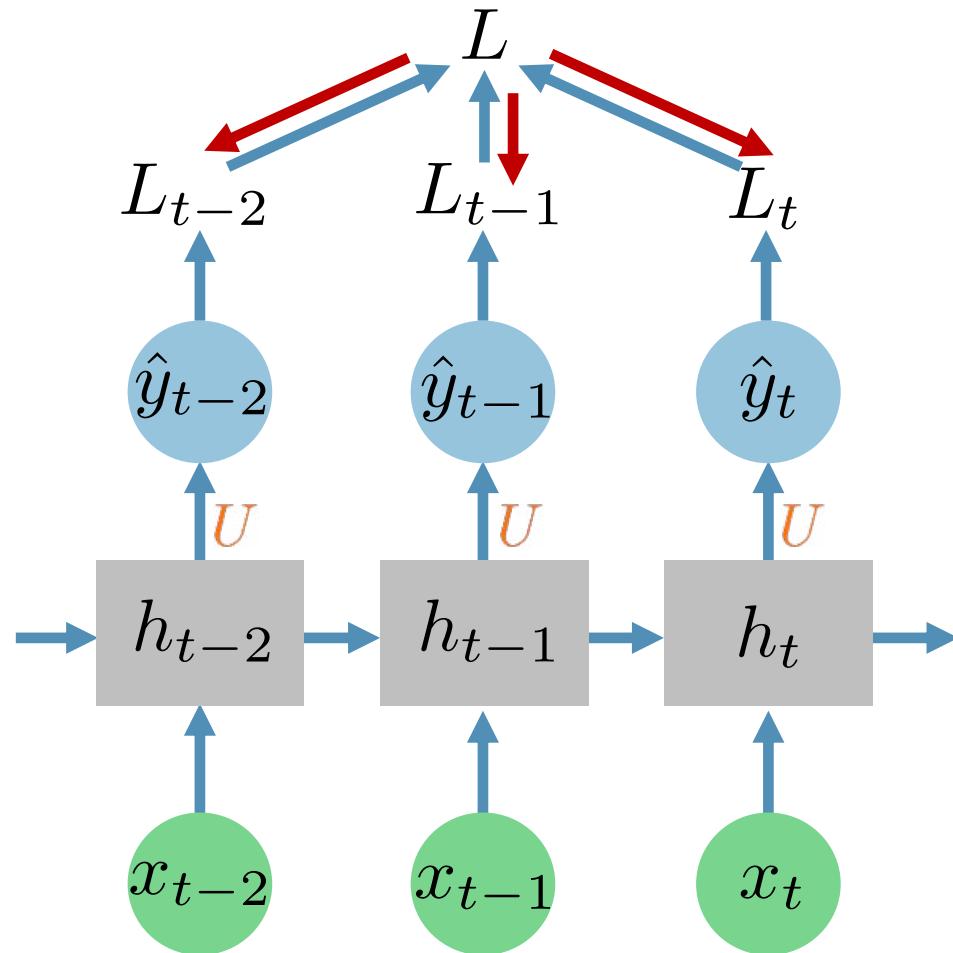
$$L = \sum_i L_i(y_i, \hat{y}_i)$$



Теперь можем запустить backprop!

BPTT

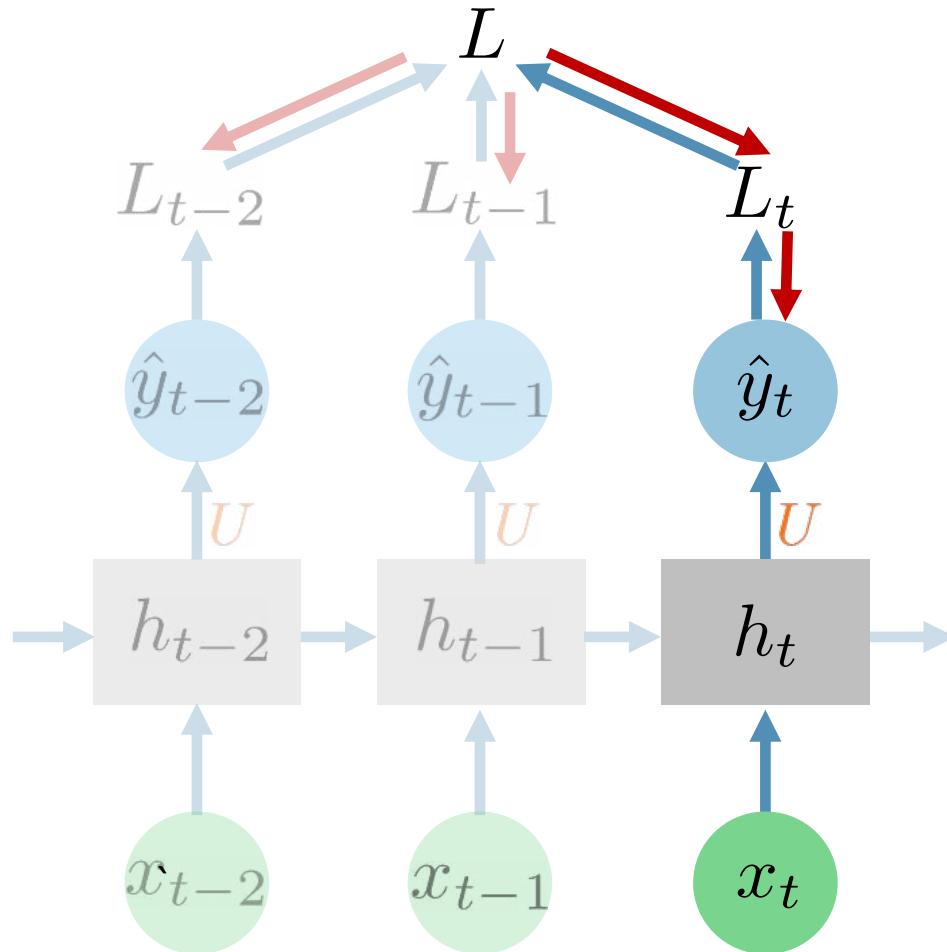
$$\frac{\partial L}{\partial U} = \sum_{i=0}^T \frac{\partial L_i}{\partial U}$$



BPTT

$$\frac{\partial L}{\partial U} = \sum_{i=0}^T \frac{\partial L_i}{\partial U}$$

$$\frac{\partial L_t}{\partial U} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial U}$$



BPTT

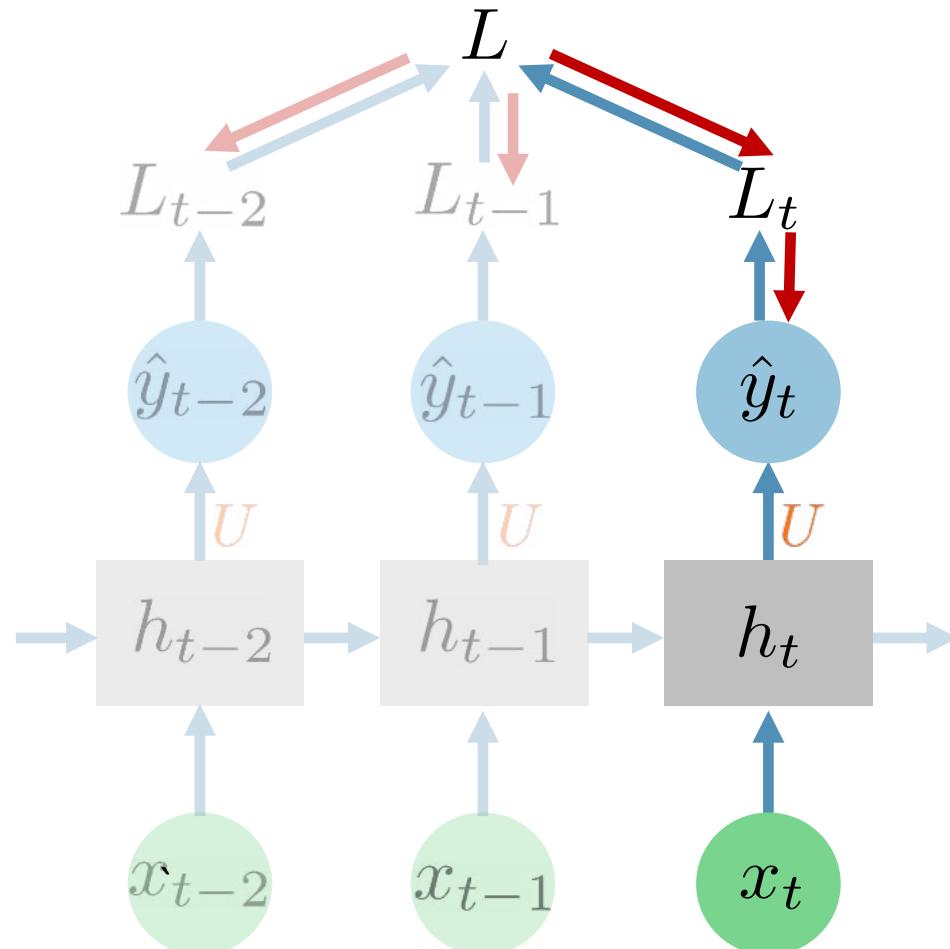
$$\frac{\partial L}{\partial U} = \sum_{i=0}^T \frac{\partial L_i}{\partial U}$$

$$\frac{\partial L_t}{\partial U} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial U}$$

$$\hat{y}_t = f_y(U h_t + b_y)$$

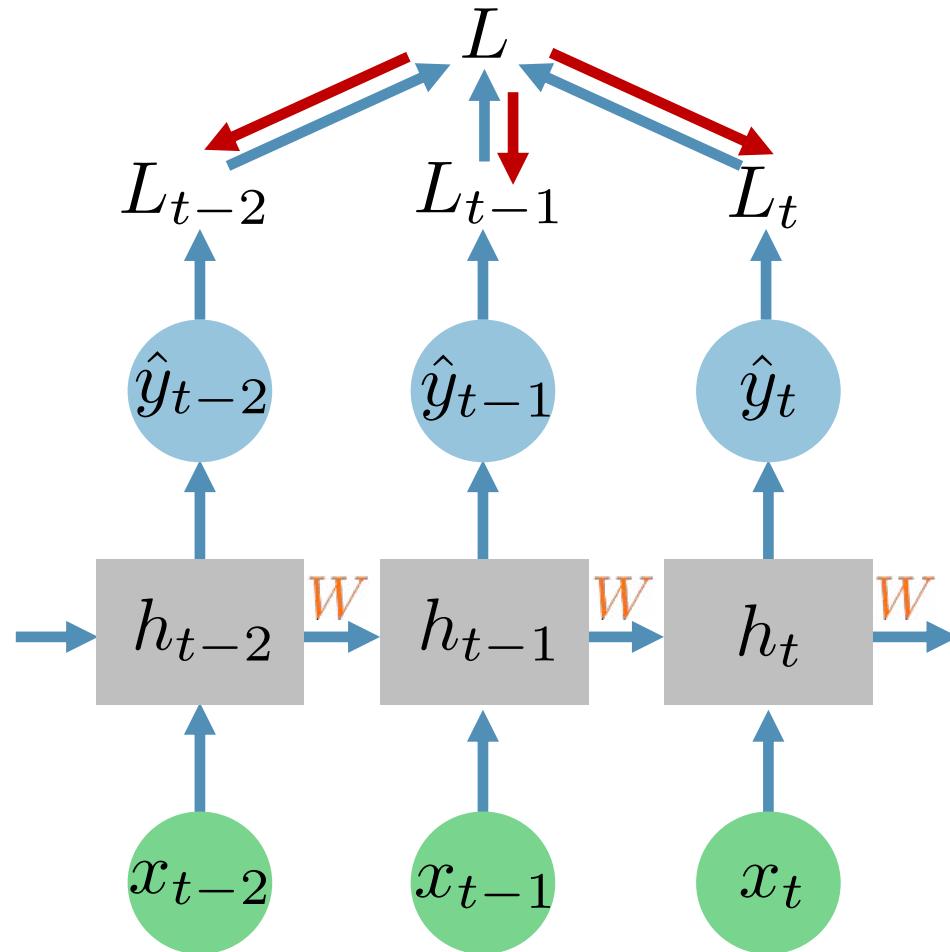
Единственное использование!

Это было просто ☺



BPTT

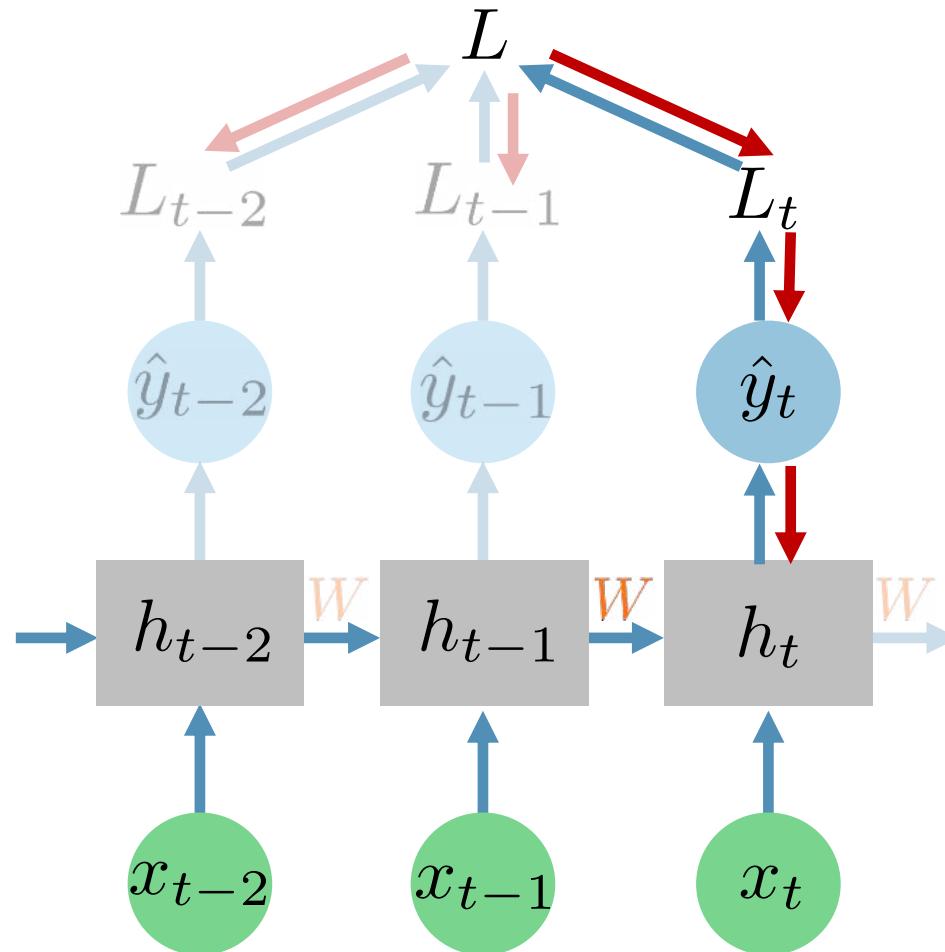
$$\frac{\partial L}{\partial W} = \sum_{i=0}^T \frac{\partial L_i}{\partial W}$$



BPTT

$$\frac{\partial L}{\partial W} = \sum_{i=0}^T \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$



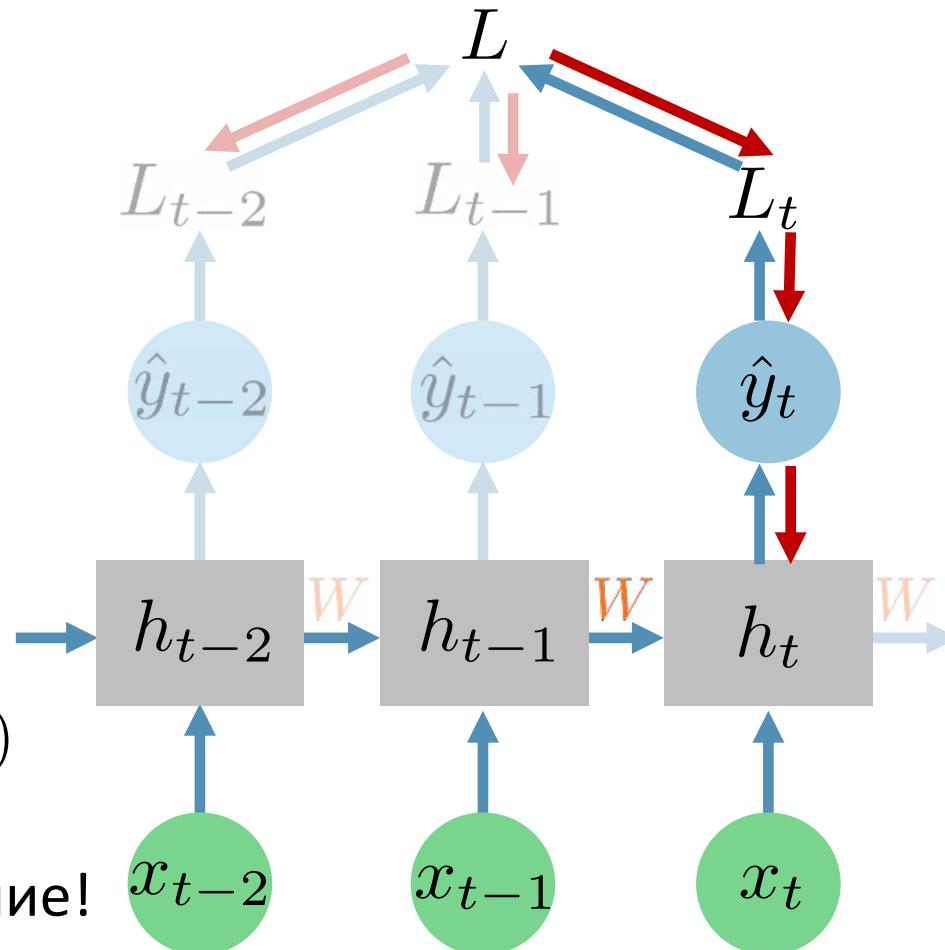
BPTT

$$\frac{\partial L}{\partial W} = \sum_{i=0}^T \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

Не единственное использование!



BPTT

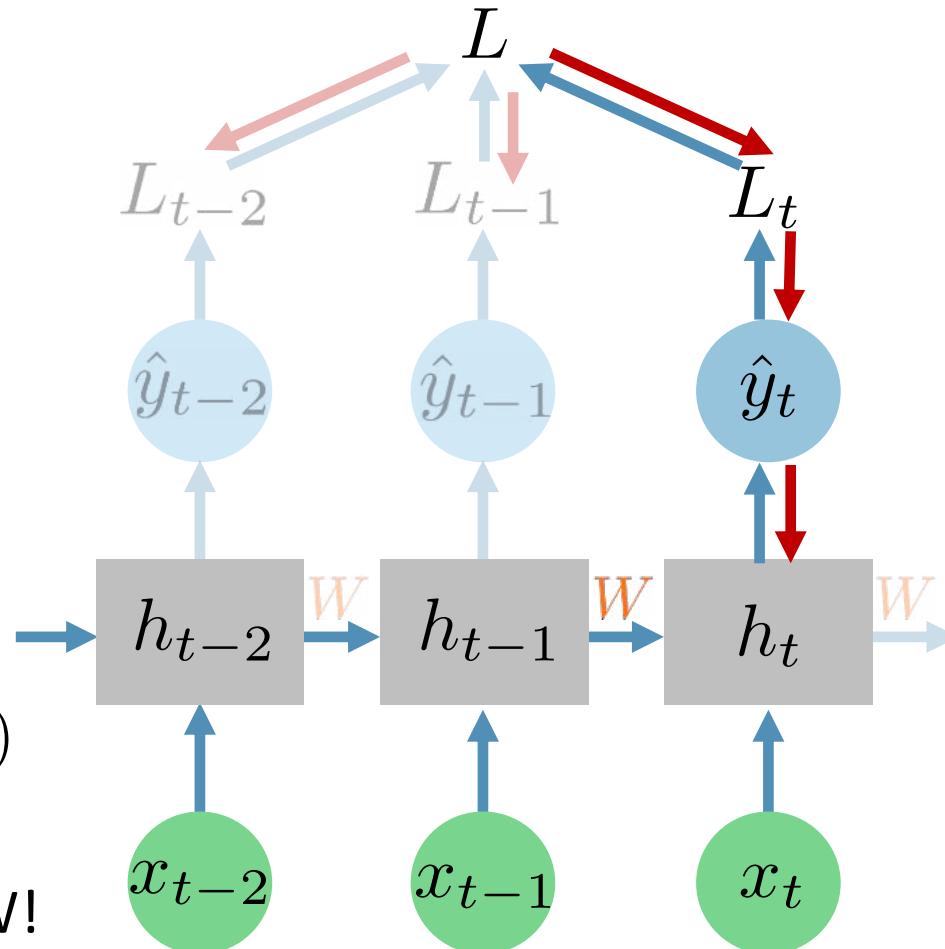
$$\frac{\partial L}{\partial W} = \sum_{i=0}^T \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

Здесь тоже используется W !

Более сложный случай!



Попробуем посчитать

- Предположим, что U, V, W, x и s – скаляры.
- Так будет проще вывести формулы 😊
- Проблема $\frac{\partial s_t}{\partial W}$ в рекуррентной формуле для s :

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

Производная для s_2

Обозначим аргумент:

$$s_1 = f(Ux_1 + Ws_0)$$
$$s_2 = f(Ux_2 + Ws_1) = f(Ux_2 + Wf(Ux_1 + Ws_0))$$

a₂

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

Потому что s_0 константа:

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

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

Результат:

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

Обозначение для $s_{j=2}$ в предположении, что s_{j-1} не зависит от W

Производная для s_3

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

Это
знаем:

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

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

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

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

Обозначение для $s_{j=3}$
в предположении, что
 s_{j-1} не зависит от W

Результат

По индукции...

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

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

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

BPTT на примере

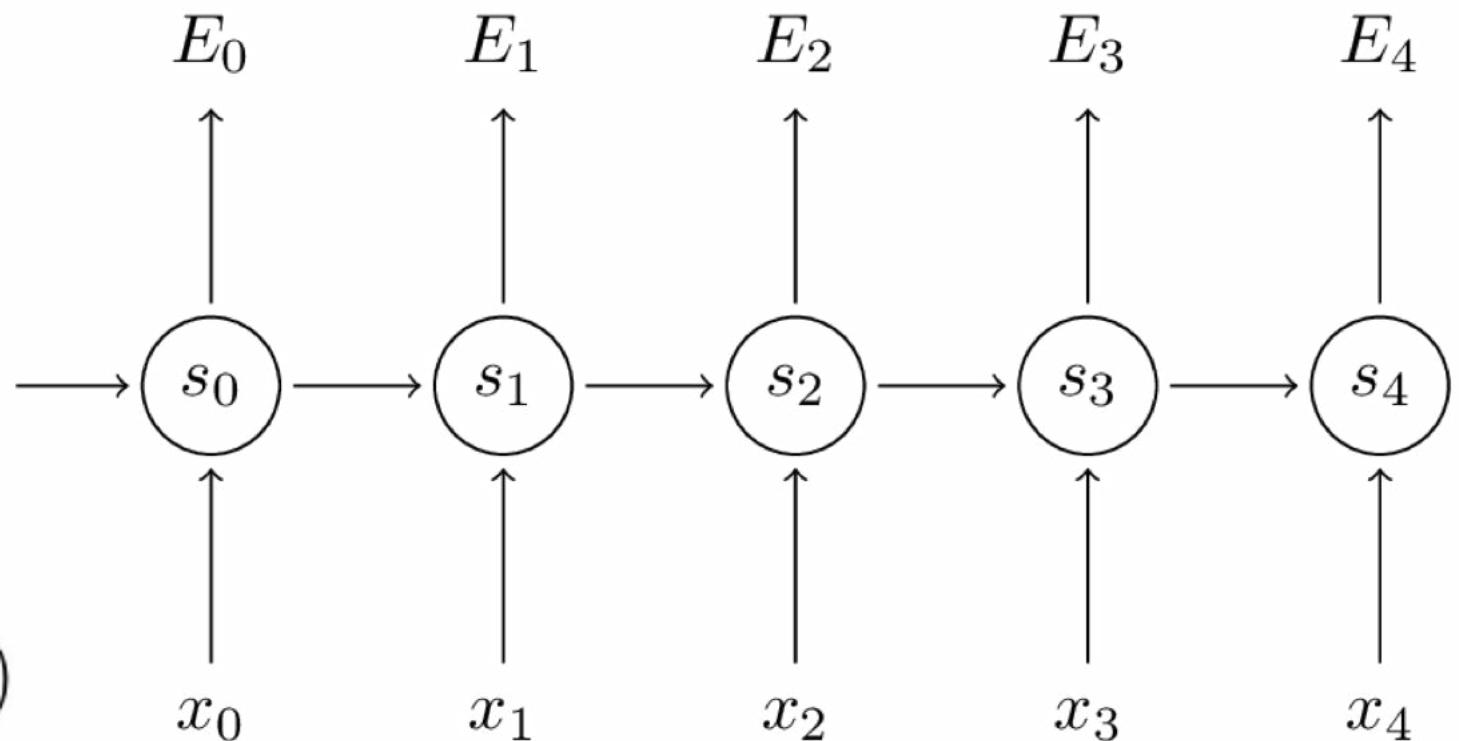
$$s_t = \tanh(Ux_t + Ws_{t-1})$$

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

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

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

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



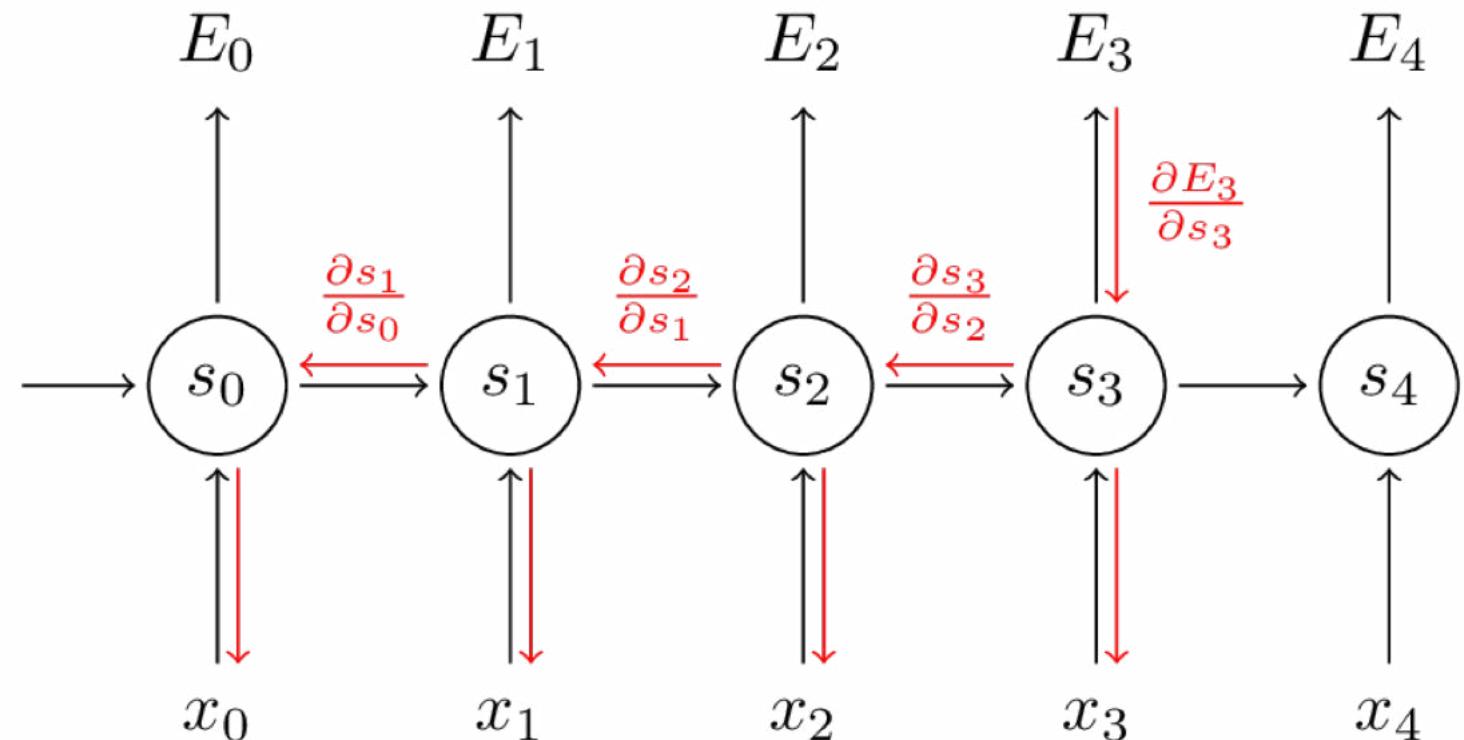
BPTT на примере

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

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

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W} =$$

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



Проблема с градиентами

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

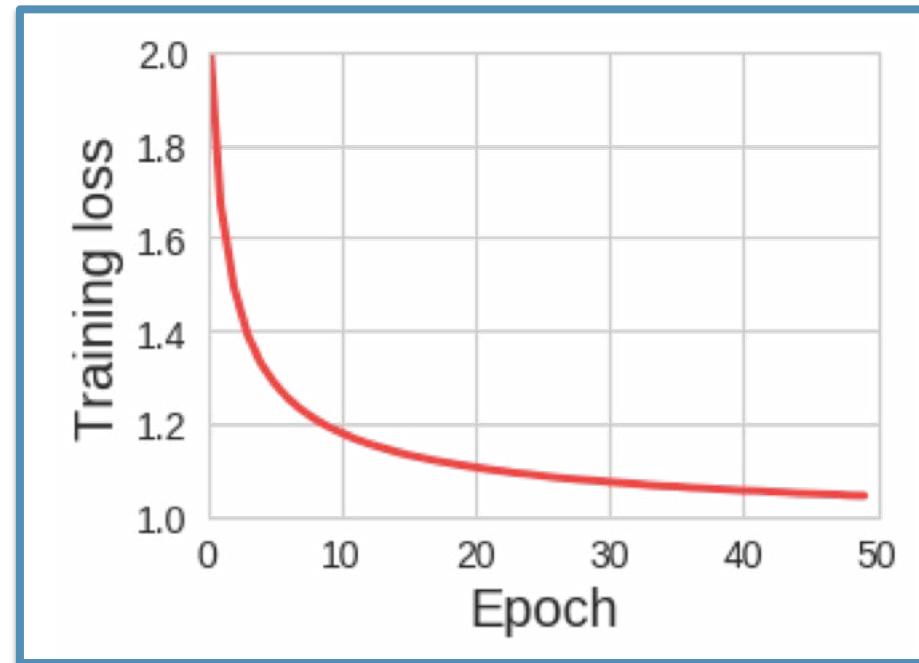
$$s_t = \tanh(Ux_t + Ws_{t-1})$$

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

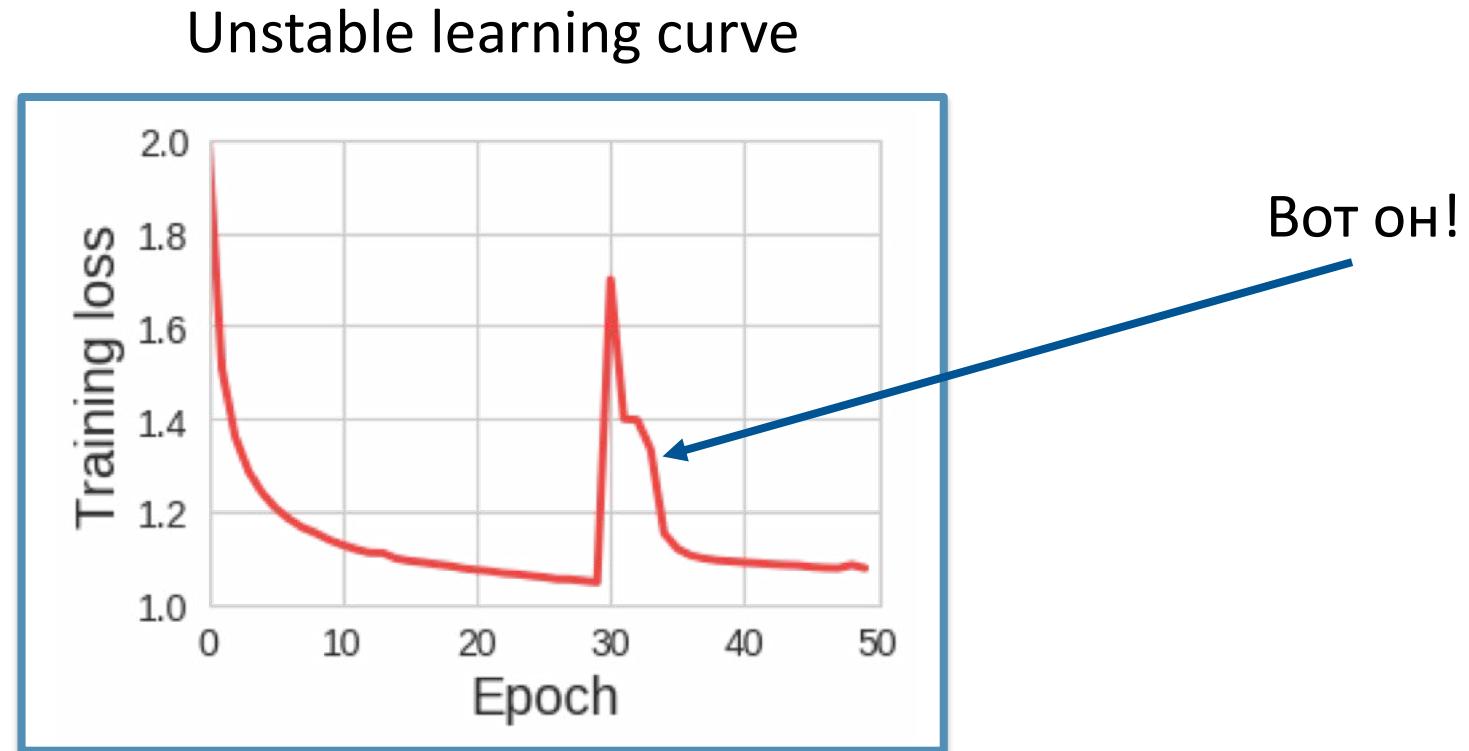
Быстро затухает
или взрывается

Взрыв легко заметить на графике

Stable learning curve



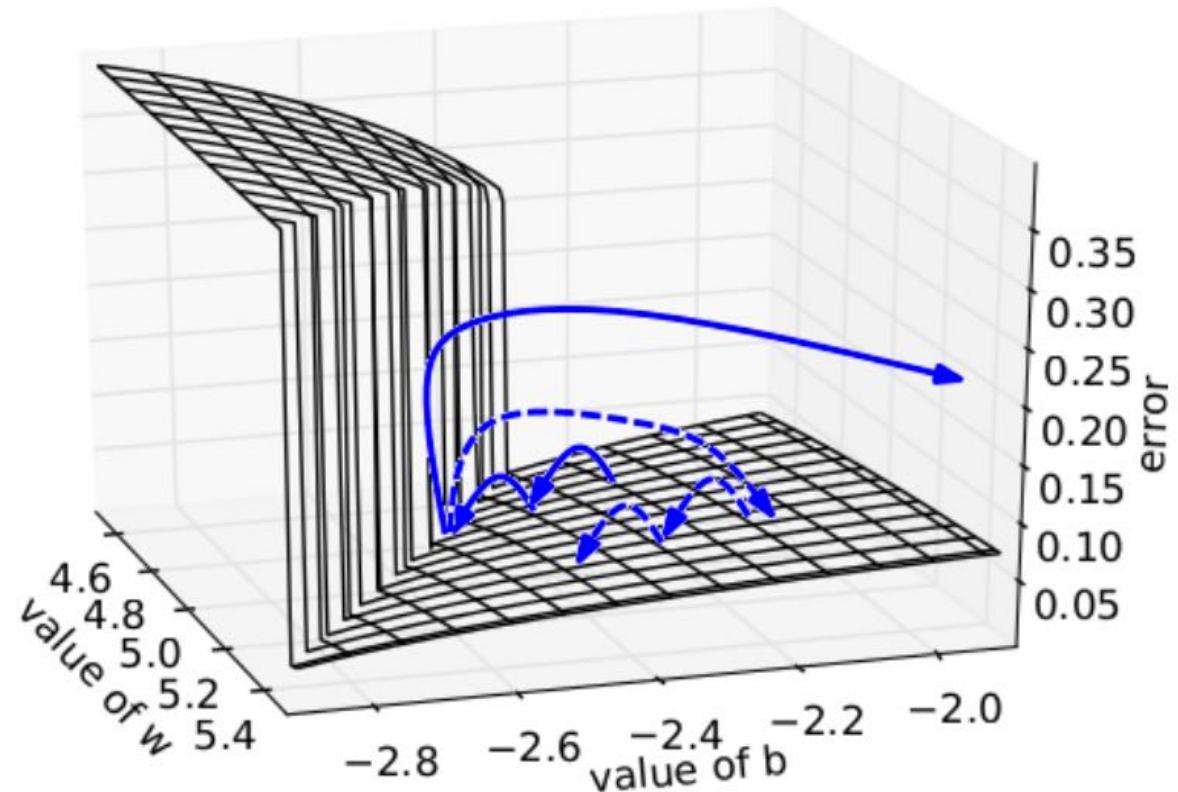
Взрыв легко заметить на графике



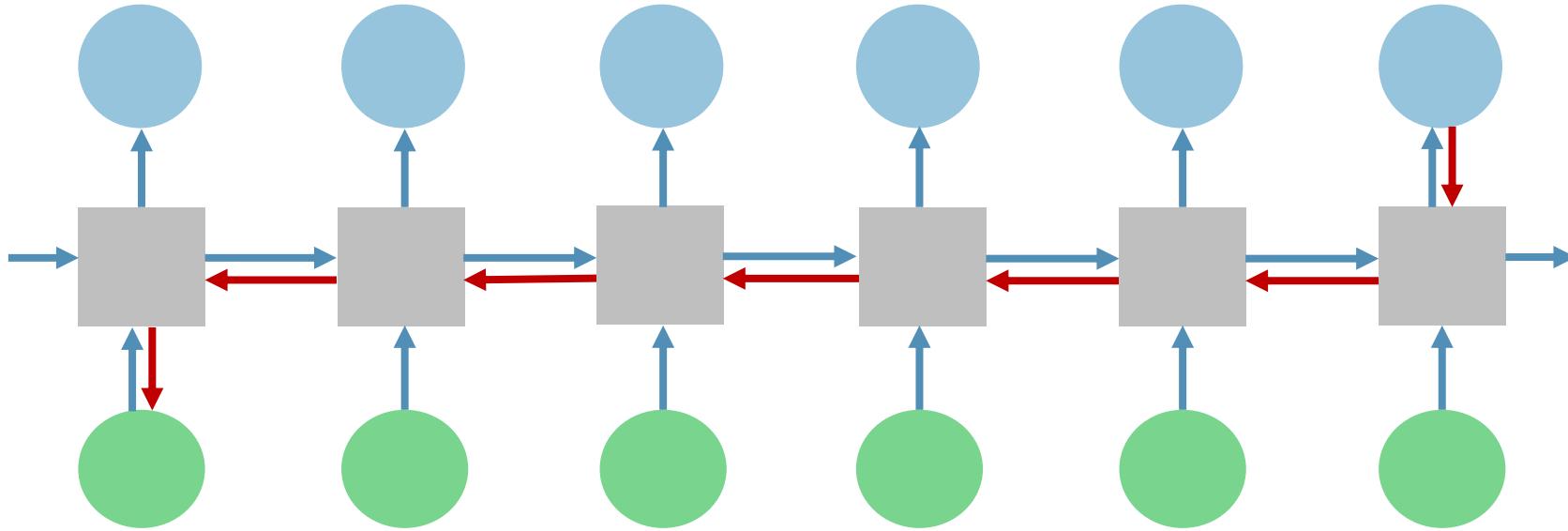
Что делать со взрывом

- Ограничим норму градиентов!

```
 $\hat{g} \leftarrow \frac{\partial \varepsilon}{\partial \theta}$ 
if  $\|\hat{g}\| \geq threshold$  then
     $\hat{g} \leftarrow \frac{threshold}{\|\hat{g}\|} \hat{g}$ 
end if
```



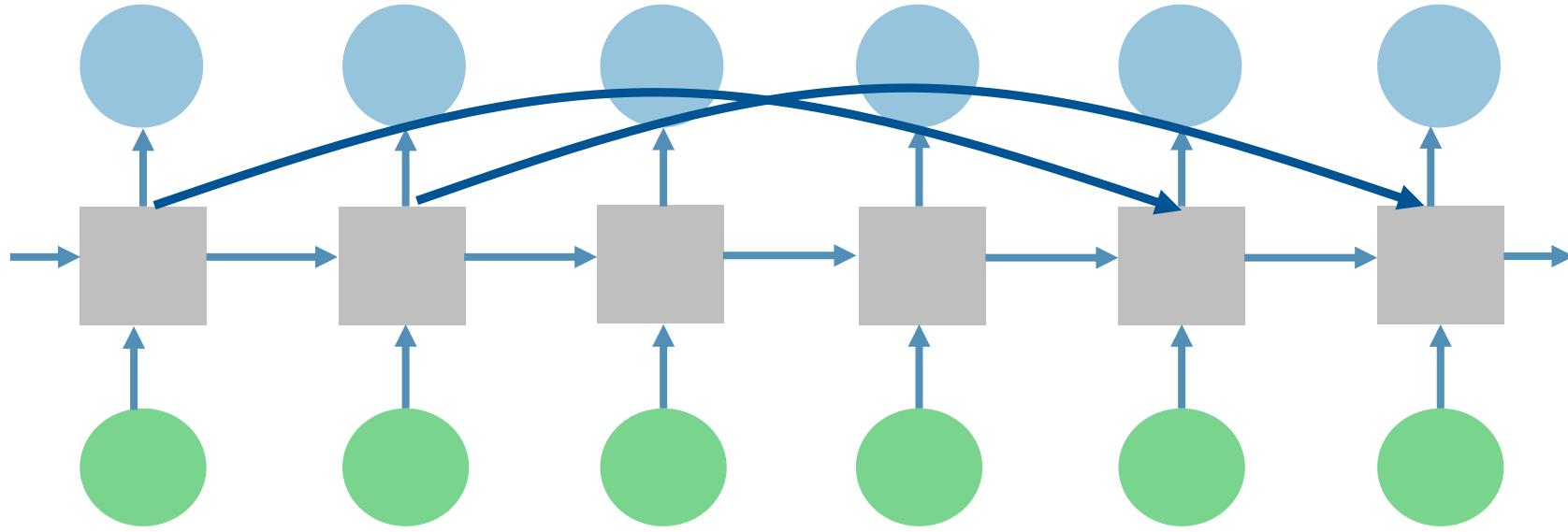
Проблема затухания



Длинные цепочки → затухающие градиенты

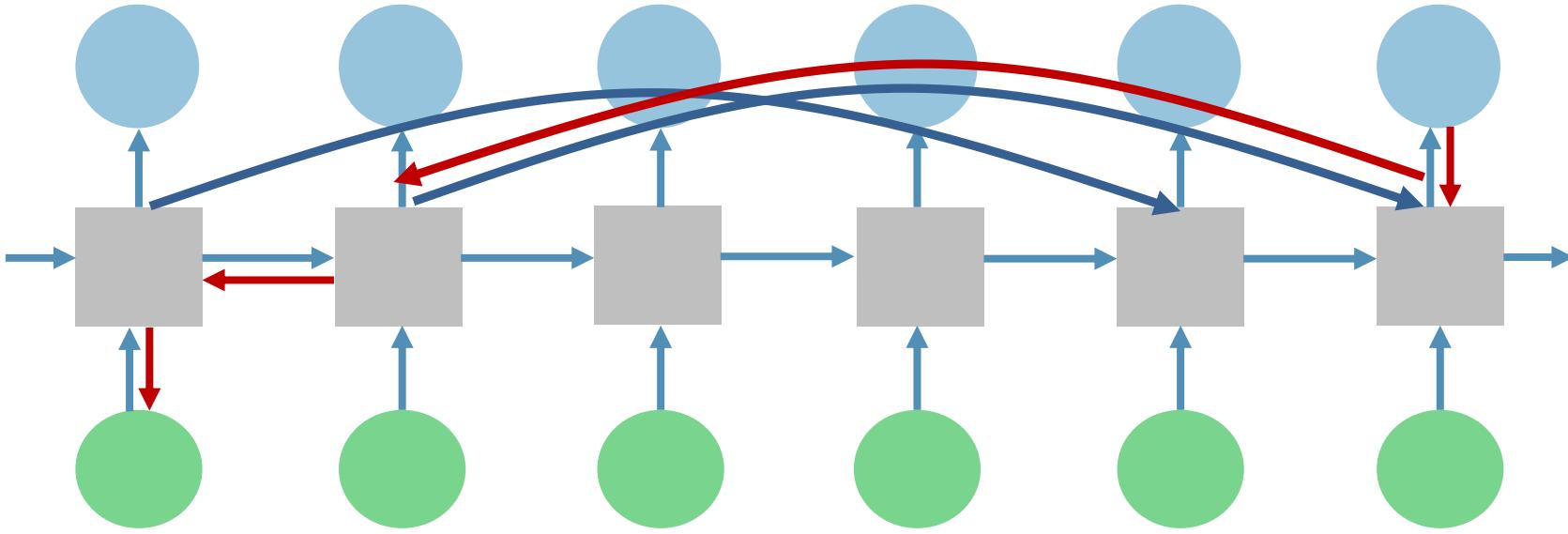
Что можно сделать?

Что делать с затуханием



Добавим более короткие пути – skip connections!

Что делать с затуханием



Градиенты распространяются дальше!

Не надо забывать про инициализацию!

$$\prod_{j=i+1}^n \frac{\partial s_j}{\partial s_{j-1}} \quad s_t = \tanh(Ux_t + Ws_{t-1})$$

Q ортогональная, если $Q^T = Q^{-1} \Rightarrow$

$\prod_i Q_i$ не будет взрываться или затухать

Инициализируем W ортогональной матрицей!

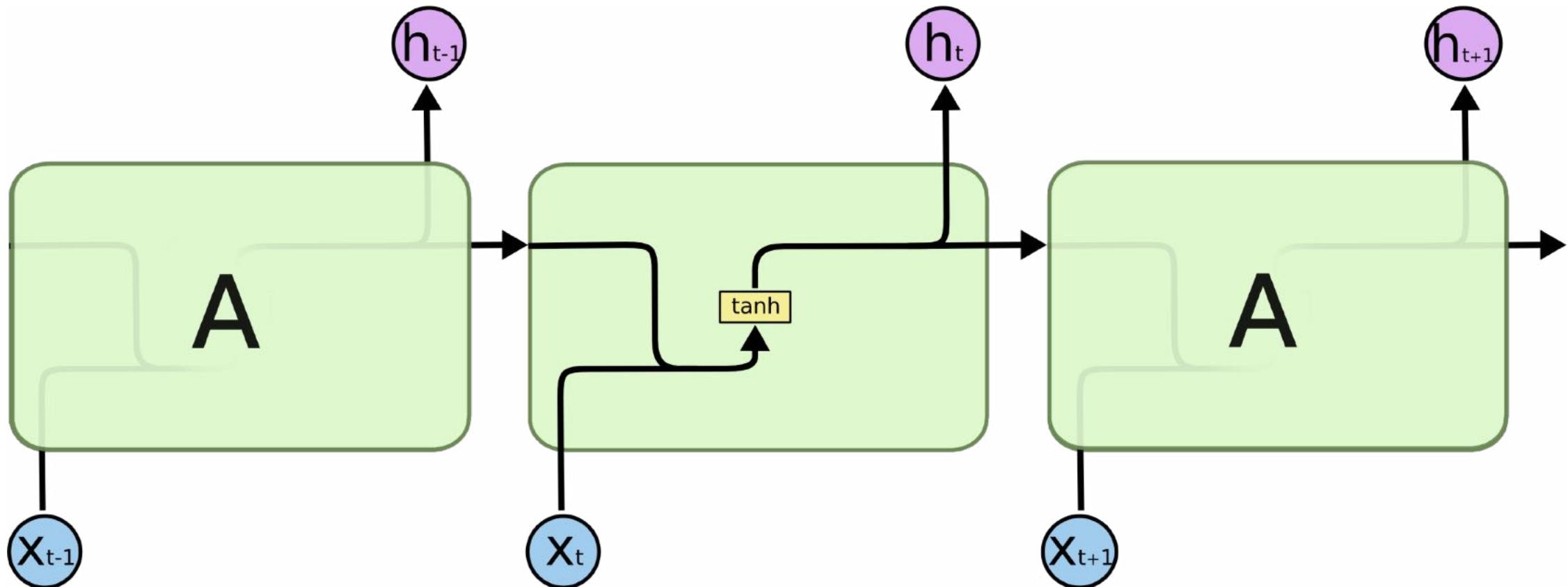
Откуда ее взять?

Не надо забывать про инициализацию!

Из SVD случайной матрицы!

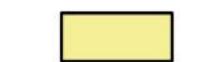
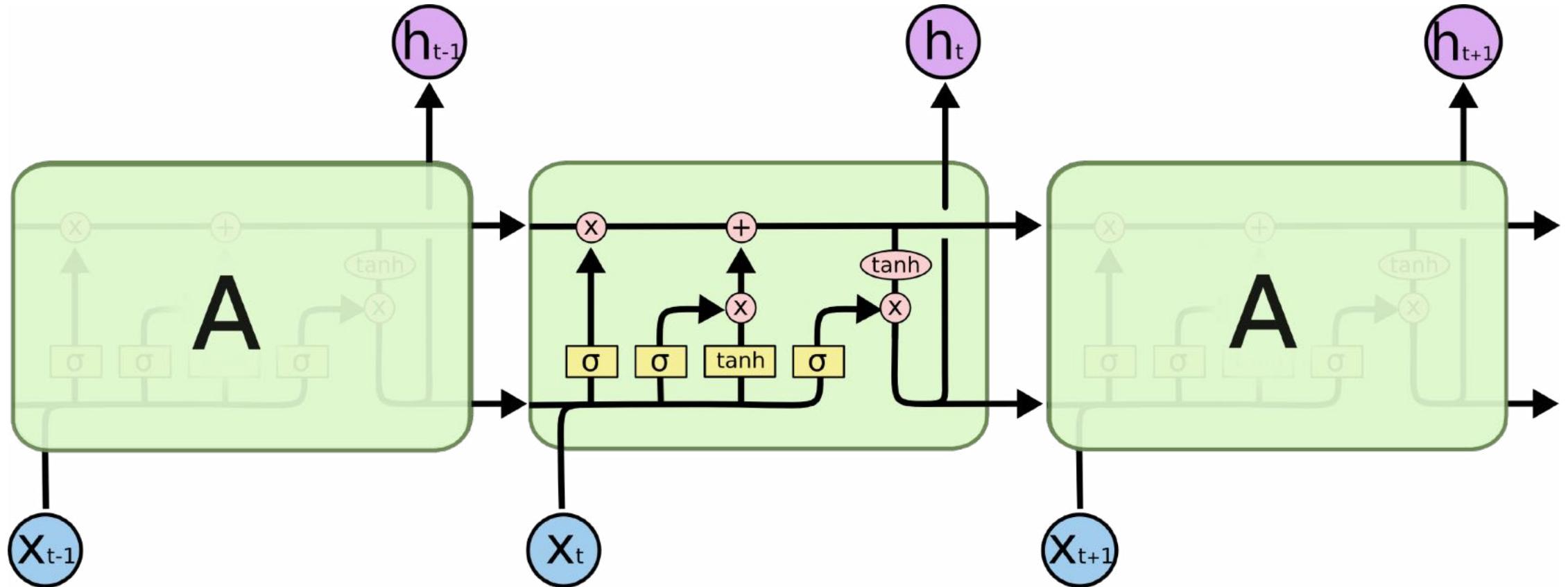
```
flat_shape = (shape[0], np.prod(shape[1:]))  
a = get_rng().normal(0.0, 1.0, flat_shape)  
u, _, v = np.linalg.svd(a, full_matrices=False)  
# pick the one with the correct shape  
q = u if u.shape == flat_shape else v  
q = q.reshape(shape)
```

Простая RNN



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

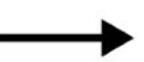
LSTM поможет затуханию!



Neural Network
Layer



Pointwise
Operation



Vector
Transfer



Concatenate

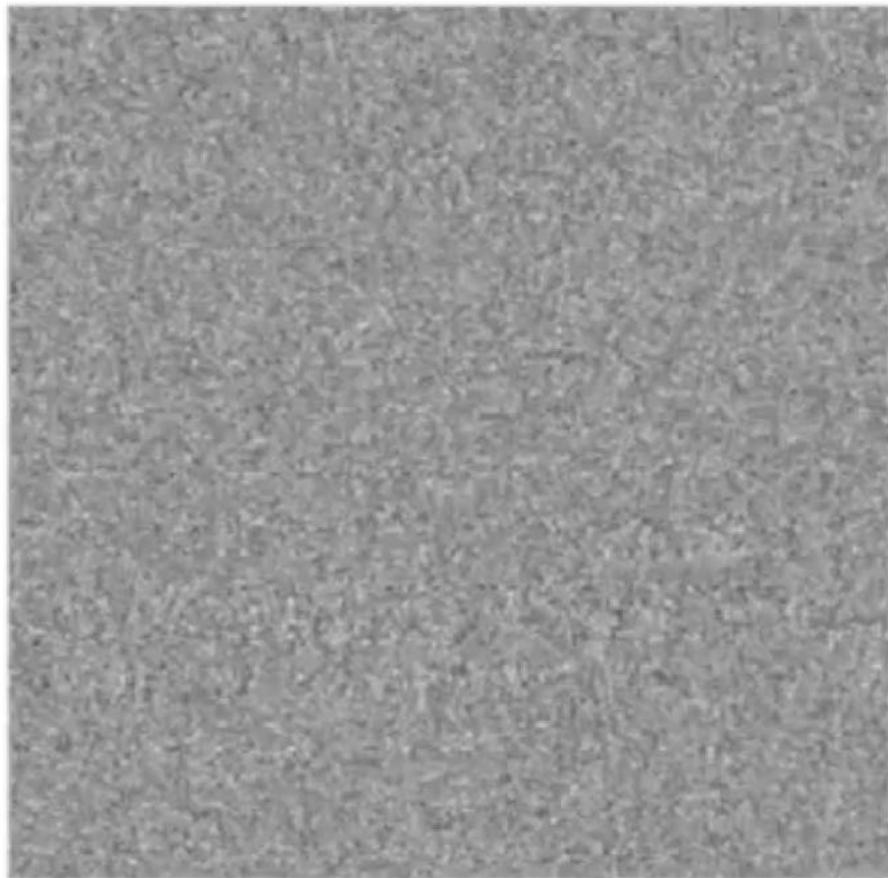


Copy

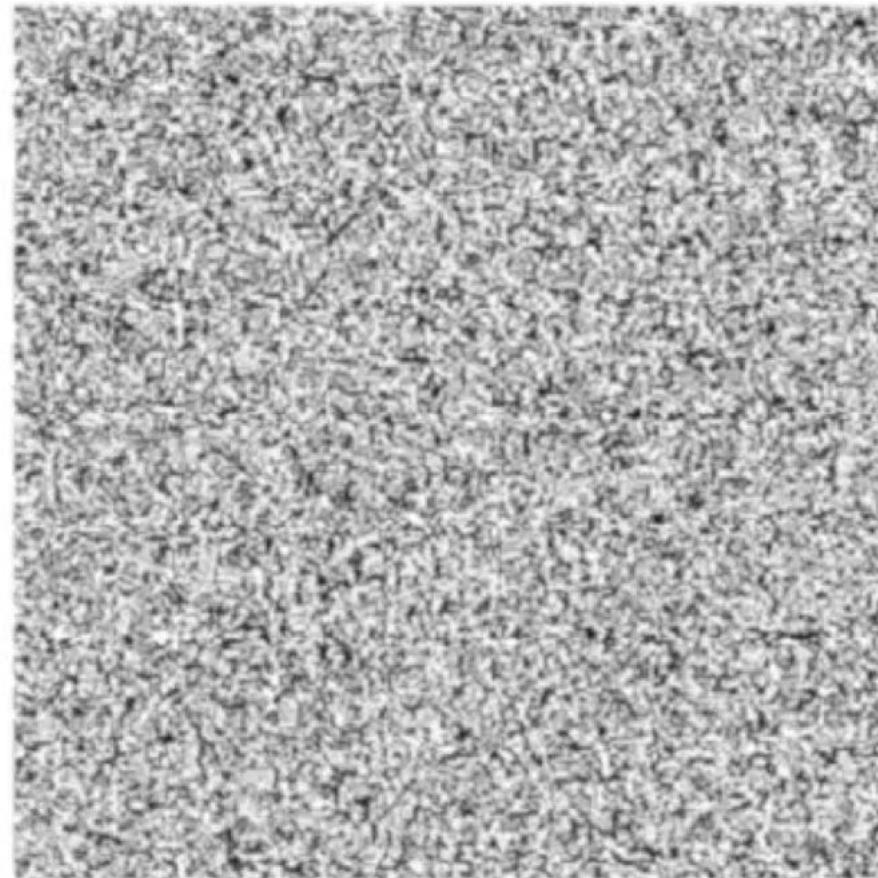
LSTM решает проблему затухания градиентов

Распространение ошибки от 128 шага

127

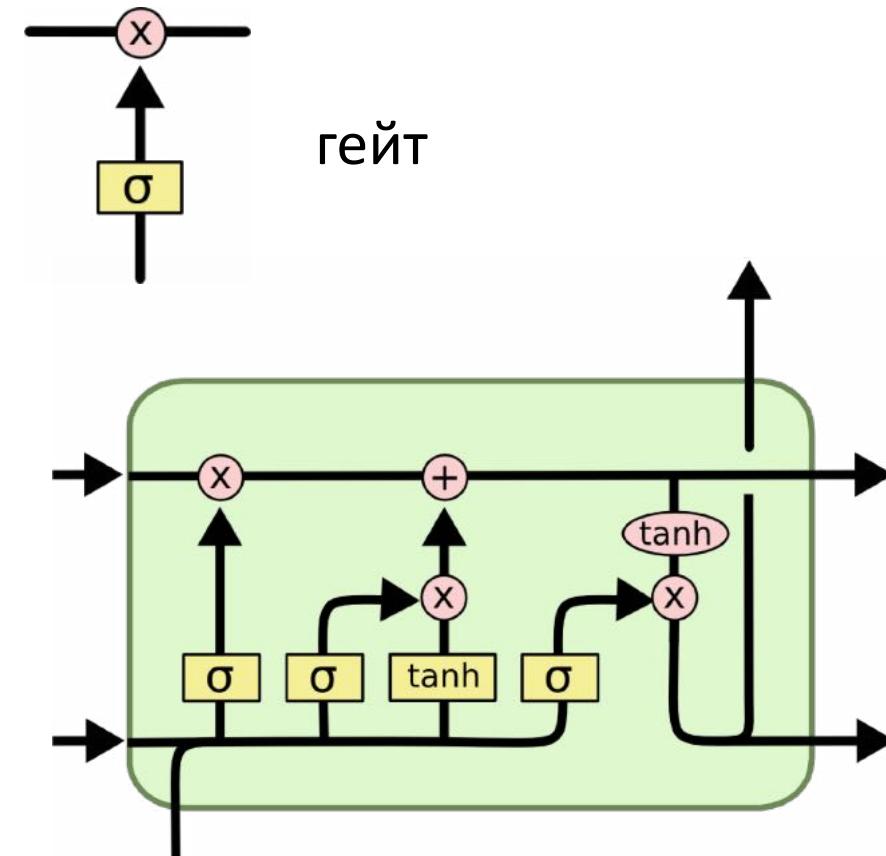
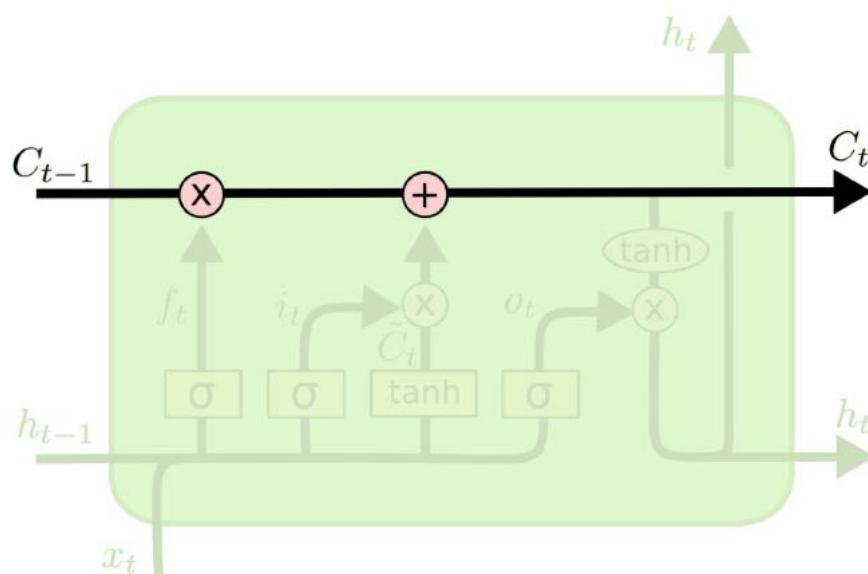


127



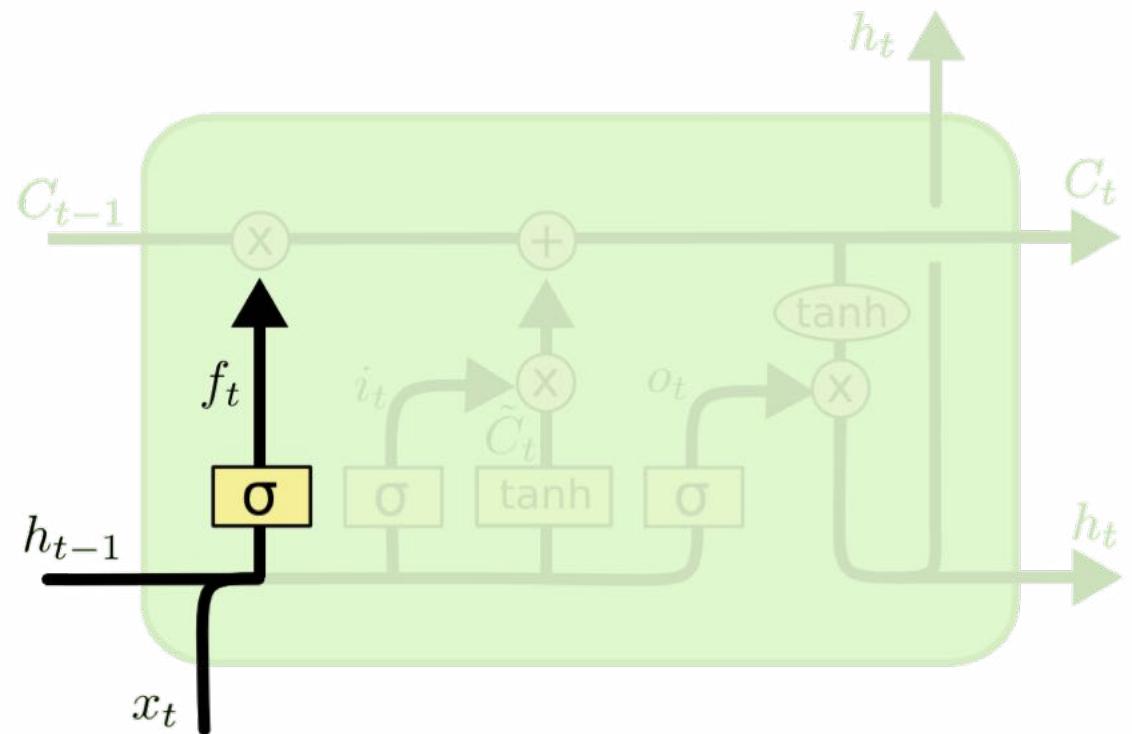
LSTM ячейка

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



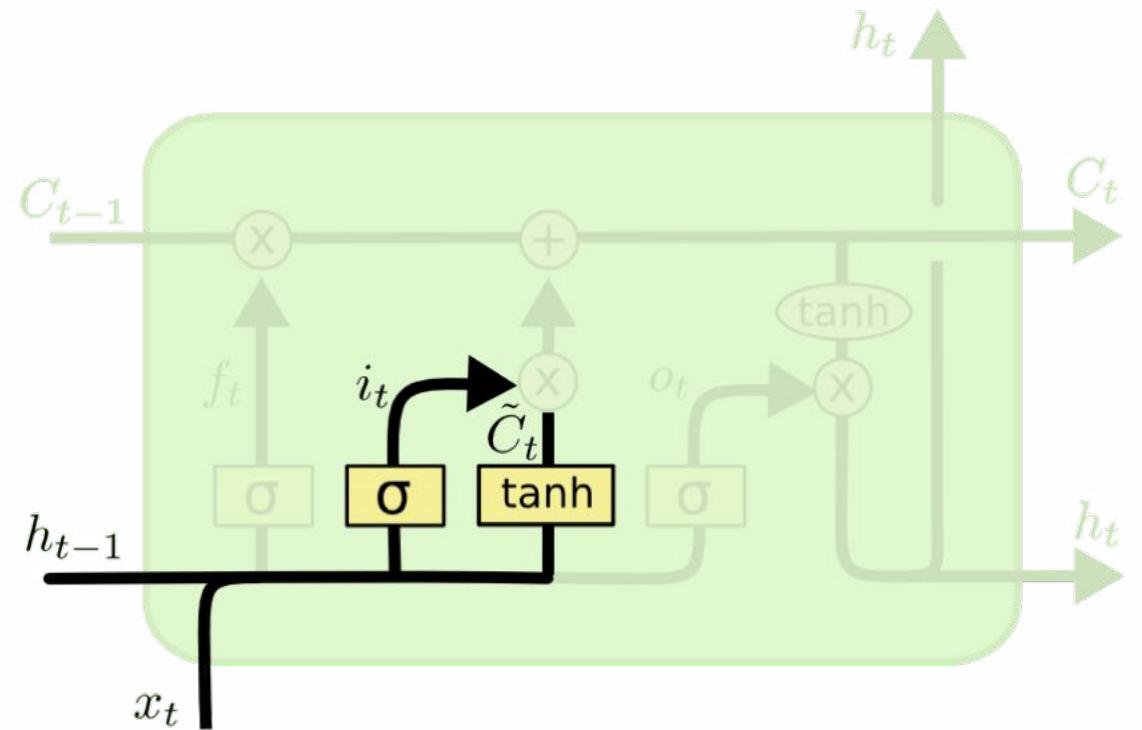
Путь для градиентов без взрывов!

LSTM forget gate



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

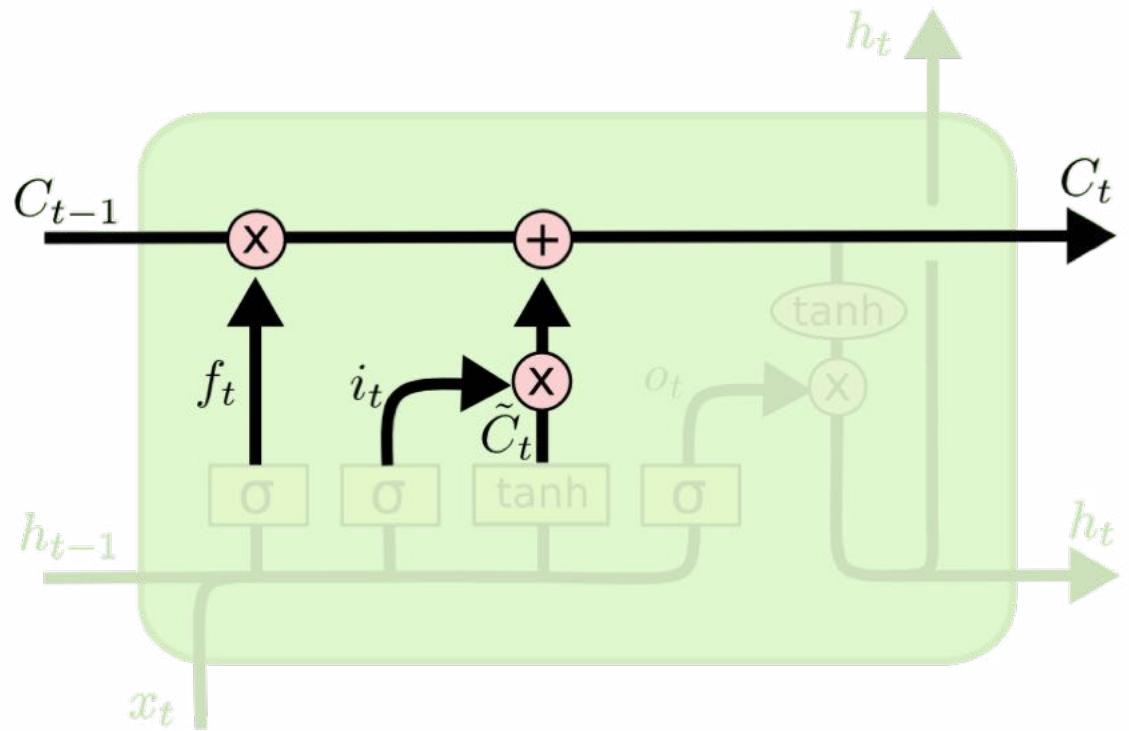
LSTM input gate и вклад ячейки в состояние



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

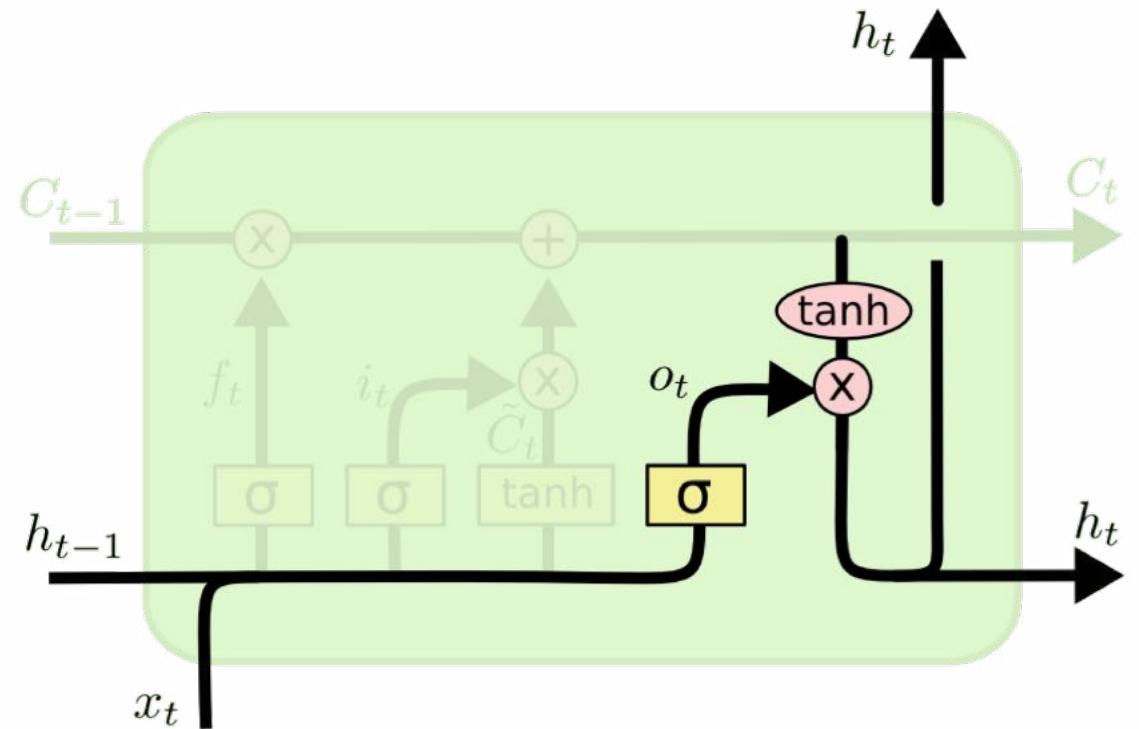
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Обновление состояния



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

Выдача скрытого состояния наружу



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

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

Пример: распознавание рукописного текста

Выход



Скрытый слой



Вход



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>