

Архитектуры нейросетей для обработки последовательностей

Андрей Белов, МФТИ, ABBYY

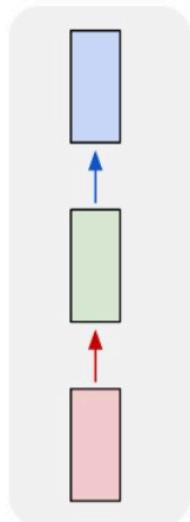
План лекции

- Введение
- Обучение RNN
- LSTM, GRU
- Варианты использования RNN
- Recursive Neral Networks

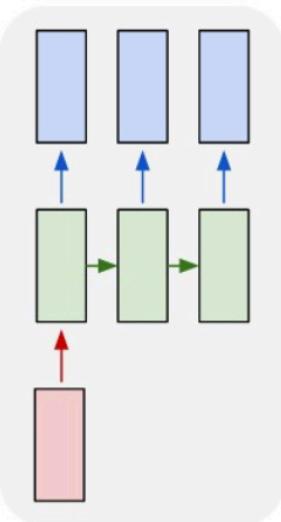
Введение

Примеры работы с последовательностями

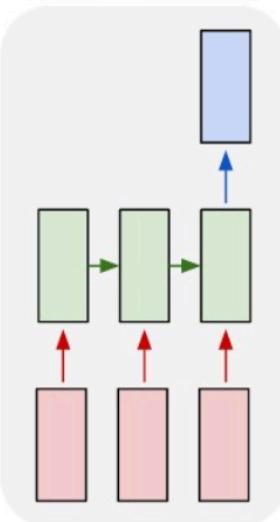
one to one



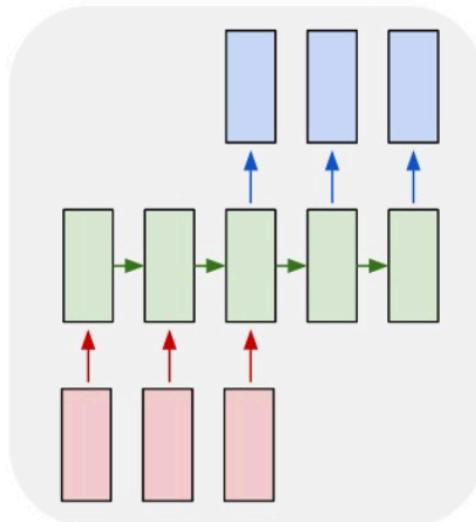
one to many



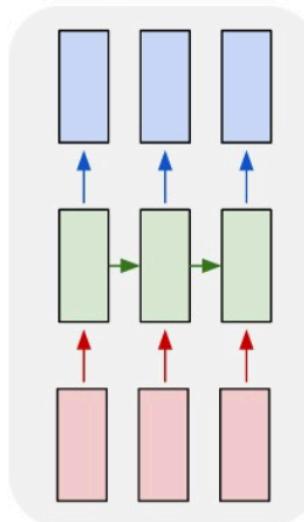
many to one



many to many



many to many



Обычная
feed forward
архитектура

Image
captioning

Sentiment
analysis

Machine Translation

POS tagging,
Video Processing

Чем плохи полносвязные слои для обработки последовательностей?

Fully Convolutional:

$$relu(\mathbf{W} \cdot \mathbf{x} + \mathbf{b})$$

Flatten:

$$[1 \ 0 \ 0 \ 0 \ \dots | \ 0 \ 1 \ 0 \ 0 \ \dots | \ 0 \ 0 \ 1 \ 0 \ \dots | \ 0 \ 0 \ 0 \ 1 \ \dots]$$

One-hot encoding:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \dots \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \dots \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \dots \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ \dots \end{bmatrix}$$

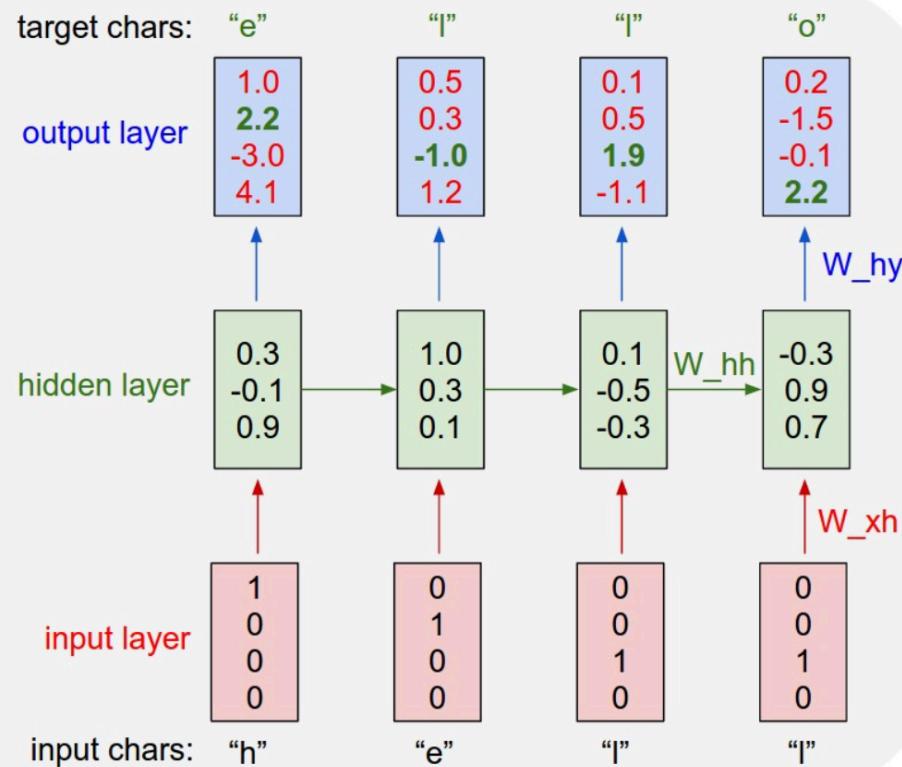
The camera is perfect

Parameter sharing

- Хотим воспользоваться тем, что во входных данных есть некоторая структура (последовательность) и применять одинаковые преобразования для каждого элемента этой структуры
- Варианты решения:
 - Использовать свертки
 - Использовать RNN

Пример: предсказание следующего символа в тексте

Vanilla RNN:



Внутреннее состояние:

$$\begin{aligned} h_t &= f(x_t, h_{t-1}) = \tanh\left(W \cdot \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix}\right) \\ &= \tanh(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t) \end{aligned}$$

$$h_0 = 0$$

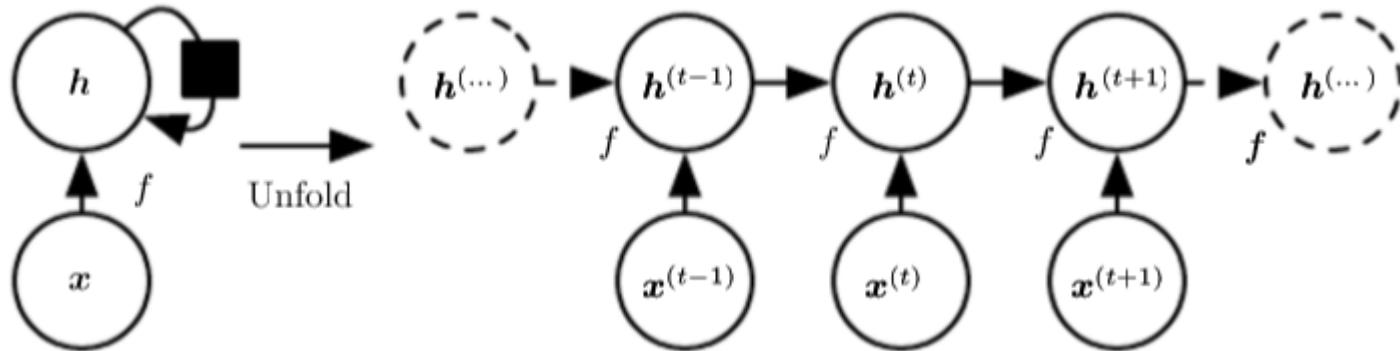
Выходной слой:

$$y_t = W_{hy} \cdot h_t$$

Генерируем y , пока модель не вернет символ конца текста/предложения.

Рекуррентная архитектура

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta})$$



Из графа видно, что значение $h(t)$ зависит от всех предшествующих $x(1) \dots x(t)$:

$$\begin{aligned}\mathbf{h}^{(t)} &= g^{(t)} (\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t-2)}, \dots, \mathbf{x}^{(2)}, \mathbf{x}^{(1)}) \\ &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta})\end{aligned}$$

Свойства рекуррентных нейросетей

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta})$$

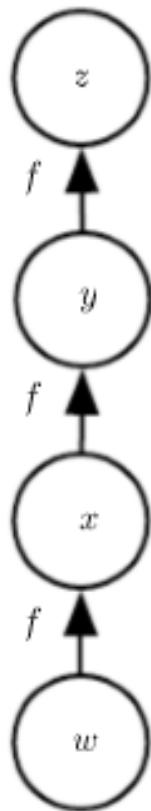
- Теоретически значение $\mathbf{h}(t)$ зависит от всех предшествующих $\mathbf{x}(1) \dots \mathbf{x}(t)$
 - Но есть потери: бесконечная последовательность $\mathbf{x}(t)$ кодируется вектором \mathbf{h} конечного размера
- Одна и та же функция f применяется для всех элементов последовательности
- Размер модели во время вывода не зависит от длины последовательности
- Любая функция, вычислимая машиной тьюринга, моделируется RNN фиксированного размера

Обучение RNN

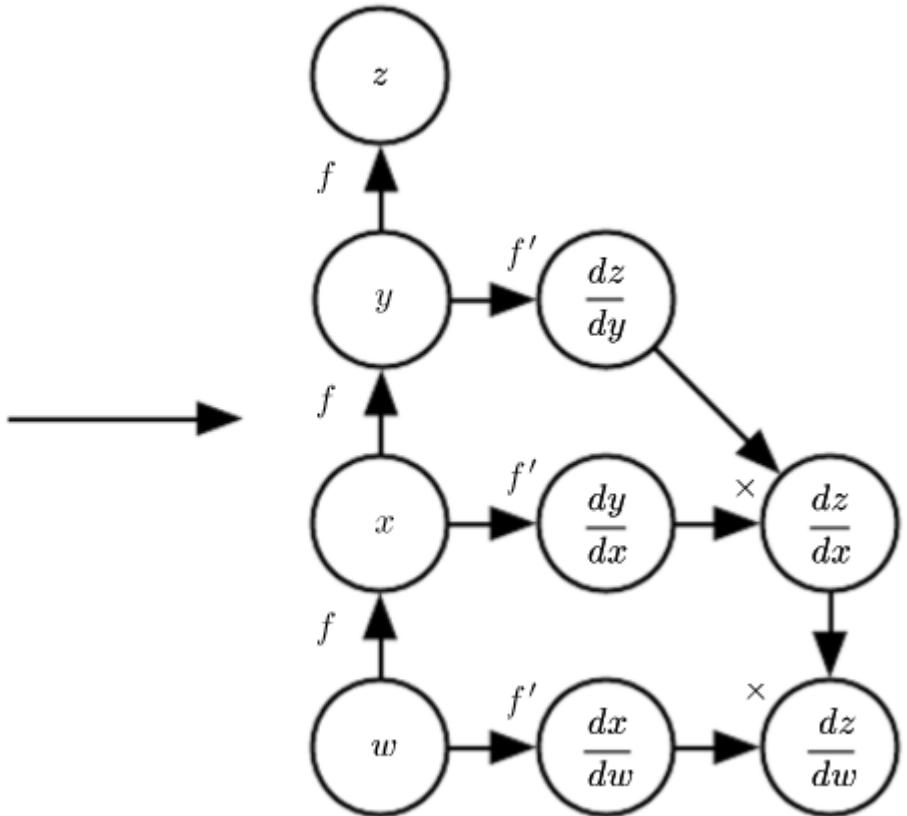
Напоминание backprop

- Есть функция потерь L
- Вычисляем значение функции потерь L на батче
- Вычисляем градиент функции потерь по весам dL/dW
- Смещаем веса алгоритмом оптимизации в сторону уменьшения функции потерь

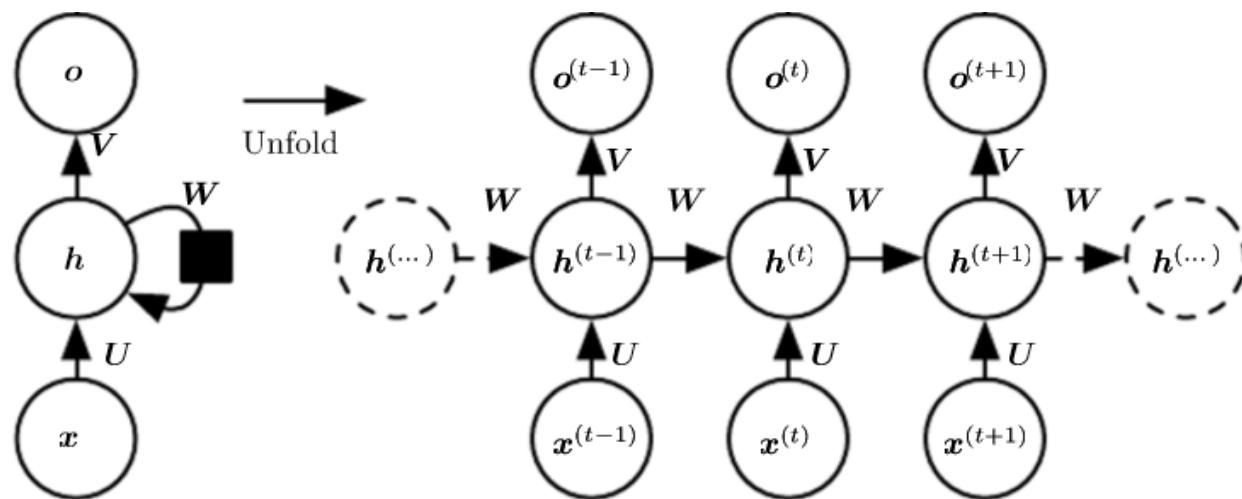
Напоминание: граф вычислений для нахождения производных



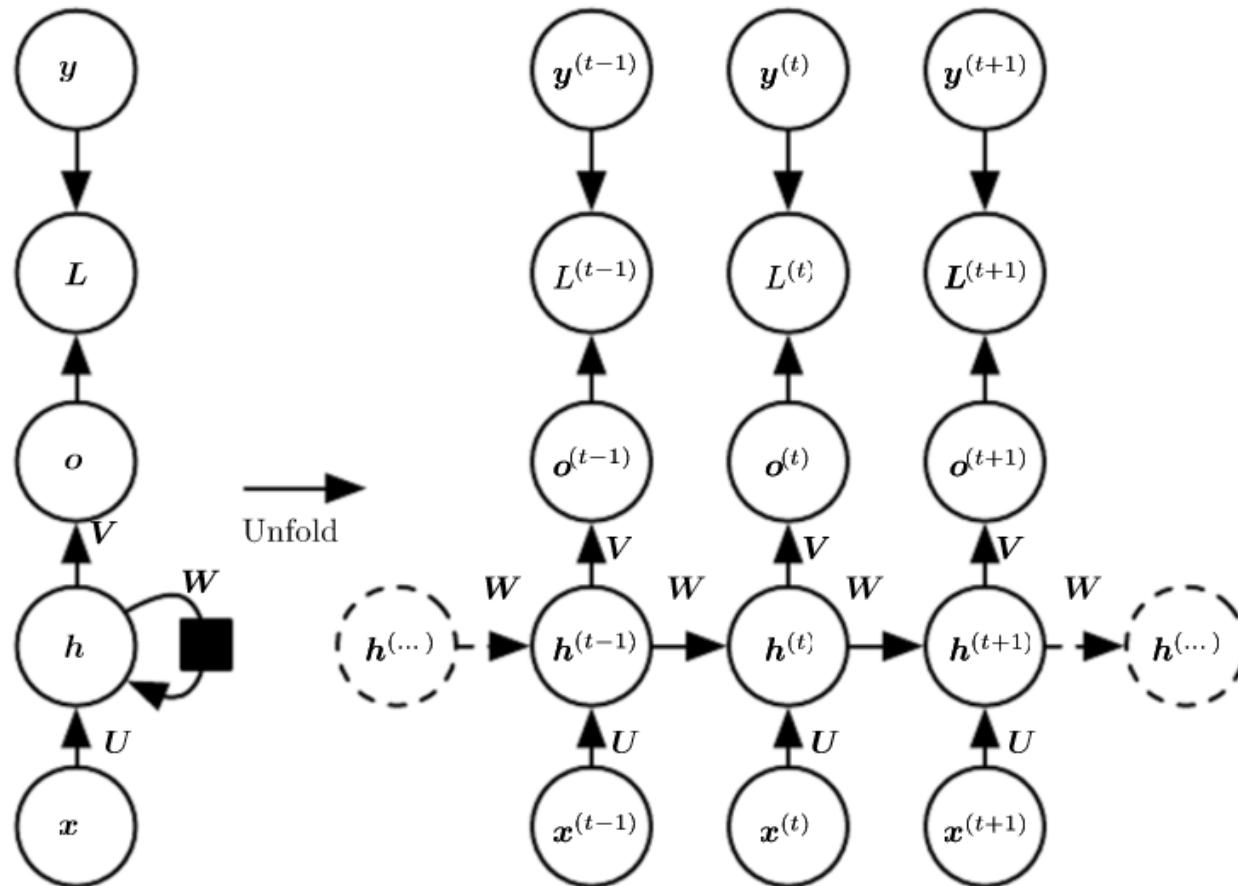
$$\begin{aligned}\frac{\partial z}{\partial w} &= \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial w} \\ &= f'(y) f'(x) f'(w) \\ &= f'(f(f(w))) f'(f(w)) f'(w)\end{aligned}$$



Обучение RNN

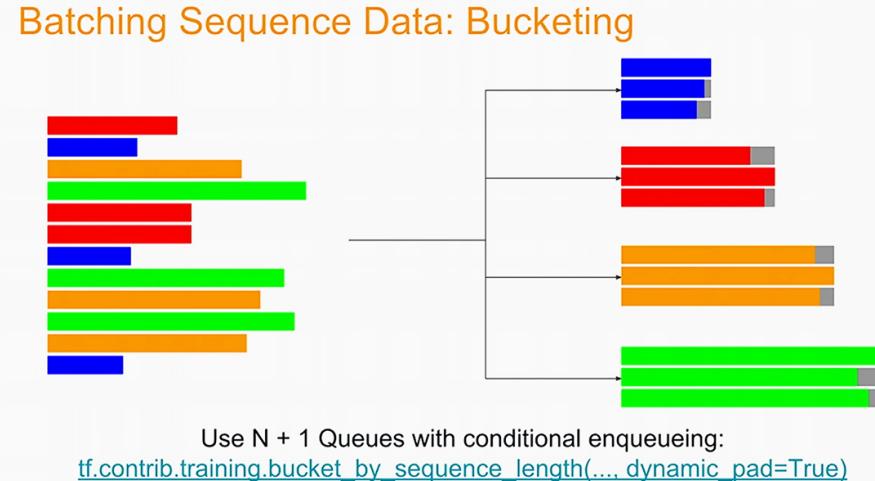
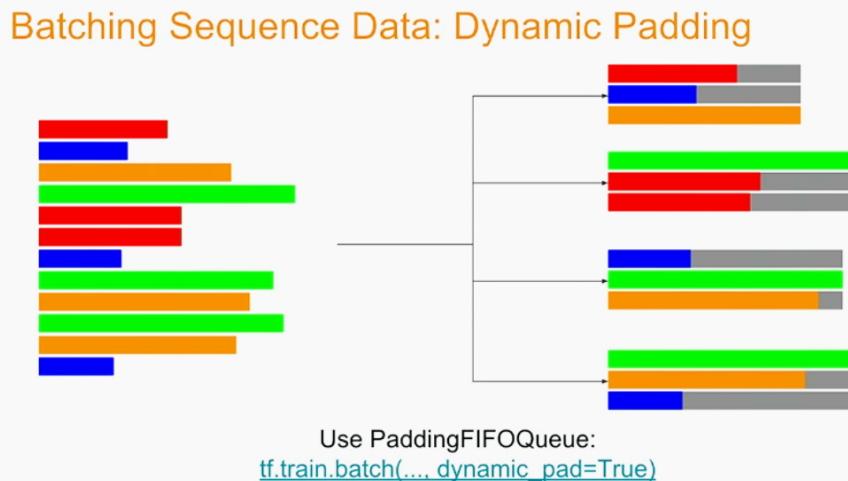


Обучение RNN

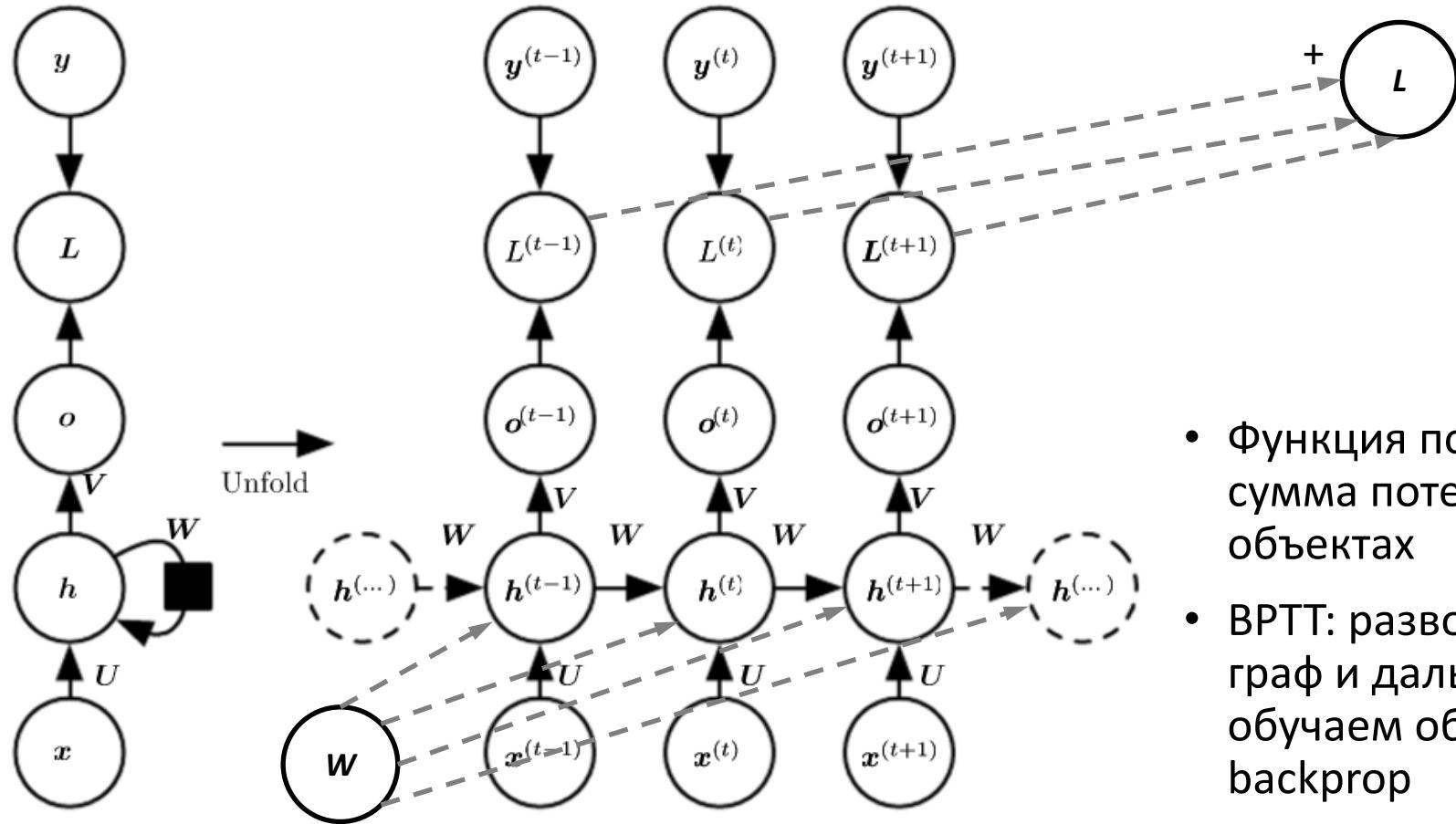


Формирование батча

- Если в бэтч помещается несколько последовательностей, но они разной длины
 - Забиваем padding-ами в начале/конце
 - Для эффективности в бэтч можно набирать последовательности примерно равной длины (можно делать в tensorflow)



Обучение RNN

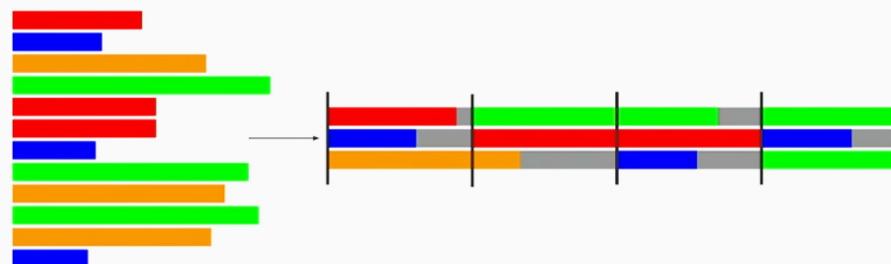


- Функция потерь – сумма потерь на всех объектах
- ВРТТ: разворачиваем граф и дальше обучаем обычным backprop

Формирование батча

- Последовательность может быть слишком длинной
 - Режем последовательность на подпоследовательности (батчи) длиной T
 - Прогоняем feedforward первом батче, запоминаем внутреннее состояние $h(T)$
 - Делаем backward pass
 - На следующем батче в качестве начального внутреннего состояния $h(0)$ передаем последнее состояние $h(T)$, из предыдущего батча

Batching Sequence Data:
Truncated BPTT via State Saver



Uses Barrier + Queues, you **must** call `save_state` each training step:
[`tf.contrib.training.batch_sequences_with_states\(...\)`](#)

Back propagation through time

- Алгоритм:
 - Разворачиваем вычислительный граф во времени
 - Считаем градиенты dL/dW
 - Обновляем веса
- Проблемы
 - Получается очень глубокая сеть
 - forward/backward pass не параллелируется по шагам t

Затухание/взрыв градиентов (vanishing/exploding)

- собственный вектор матрицы W – ненулевой вектор x , такой что $Wx = \lambda x$, где λ – собственное число.
- Матрицу W можно разложить на собственные вектора: $W = Q\Lambda Q^\top$
- Тогда, опустив нелинейности внутреннее состояние на шаге t можно записать как

$$h^{(t)} = W^\top h^{(t-1)}$$

$$h^{(t)} = (W^t)^\top h^{(0)}$$

$$h^{(t)} = Q^\top \Lambda^t Q h^{(0)}$$

- Матрица Λ^t может принимать очень «неприятные» значения
- Напомним, что $d(W_2 * h(x, W_1)) / dW_1 = W_2$

Затухание/взрыв градиентов в рекуррентных нейросетях

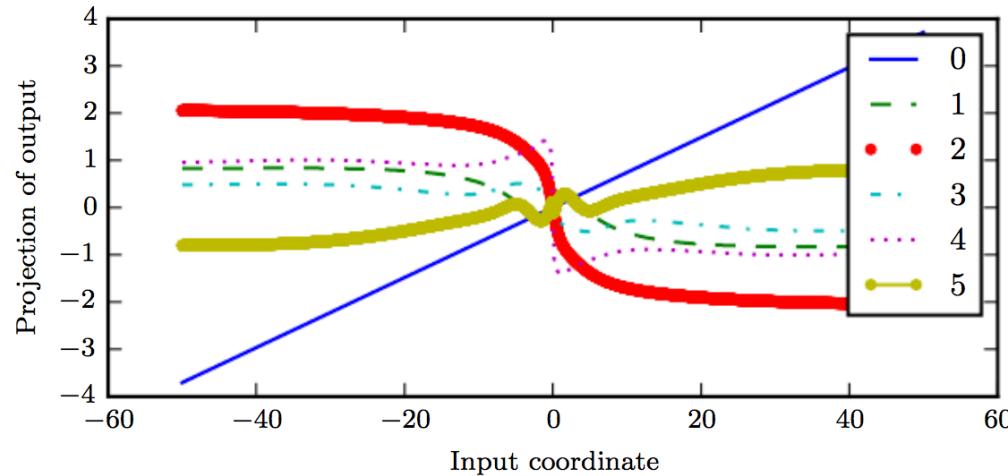
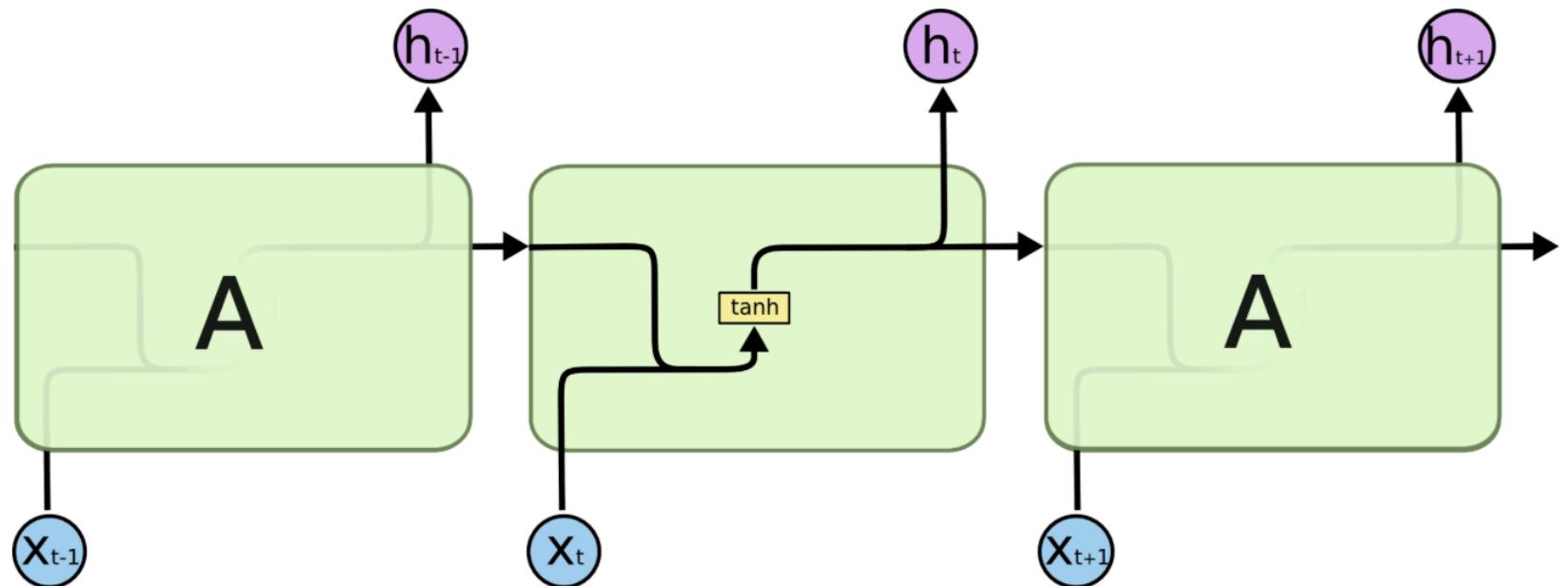


Figure 10.15: Repeated function composition. When composing many nonlinear functions (like the linear-tanh layer shown here), the result is highly nonlinear, typically with most of the values associated with a tiny derivative, some values with a large derivative, and many alternations between increasing and decreasing. Here, we plot a linear projection of a 100-dimensional hidden state down to a single dimension, plotted on the y -axis. The x -axis is the coordinate of the initial state along a random direction in the 100-dimensional space. We can thus view this plot as a linear cross-section of a high-dimensional function. The plots show the function after each time step, or equivalently, after each number of times the transition function has been composed.

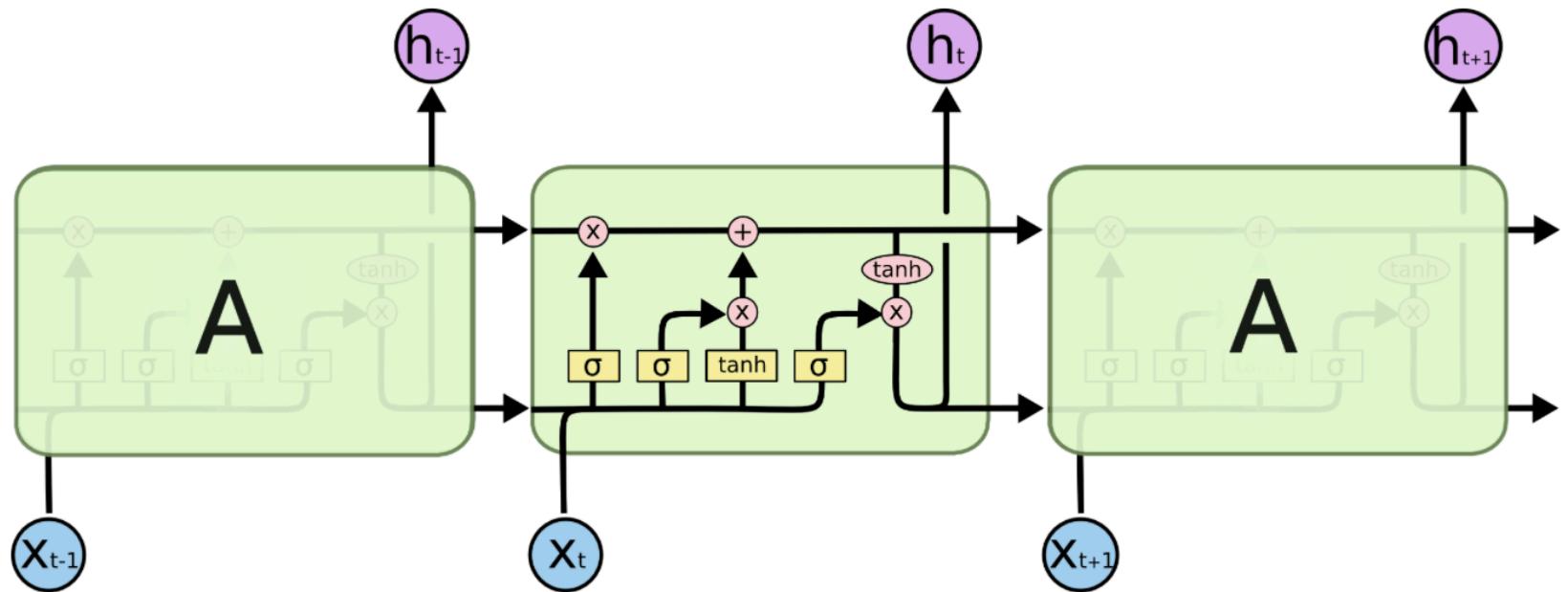
LSTM и GRU

Еще раз: Vanilla RNN

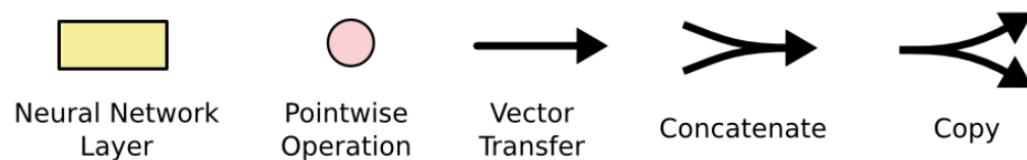


The repeating module in a standard RNN contains a single layer.

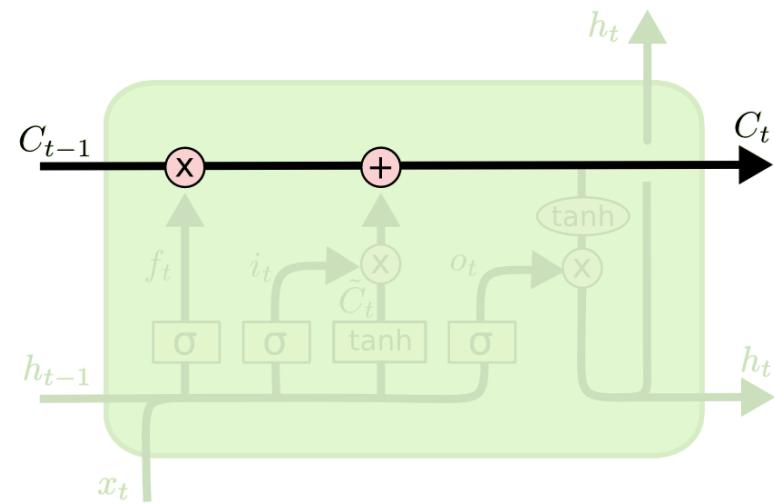
LSTM – Long Short-Term Memory



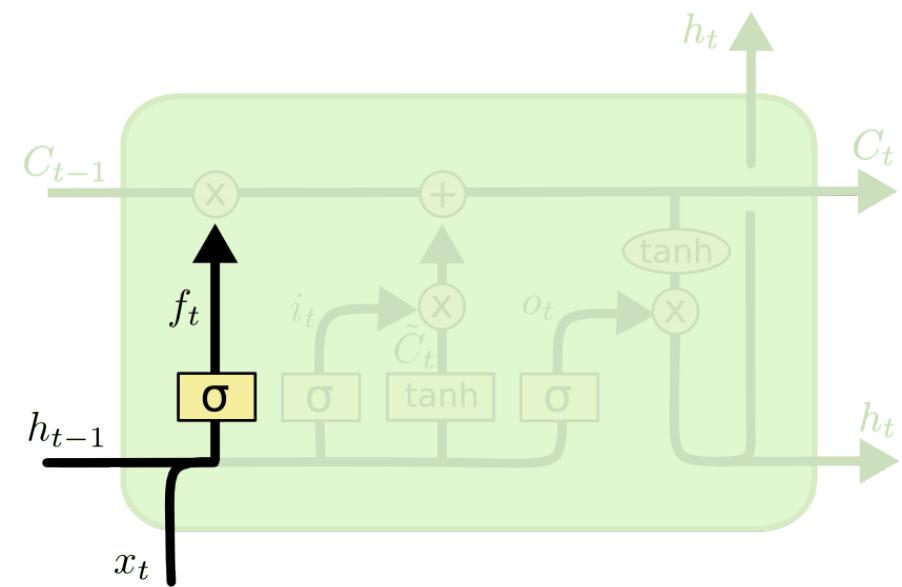
The repeating module in an LSTM contains four interacting layers.



LSTM: Long term memory

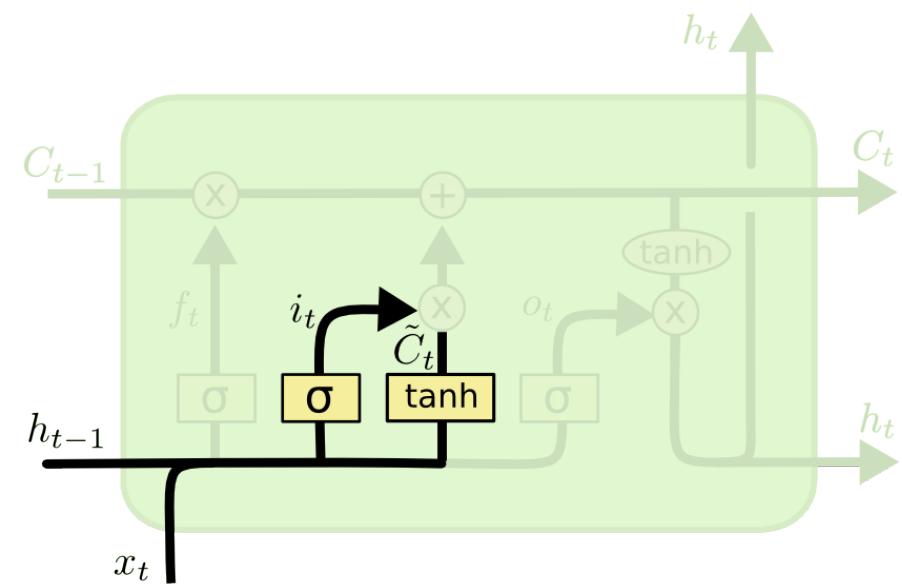


LSTM: Forget gate



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

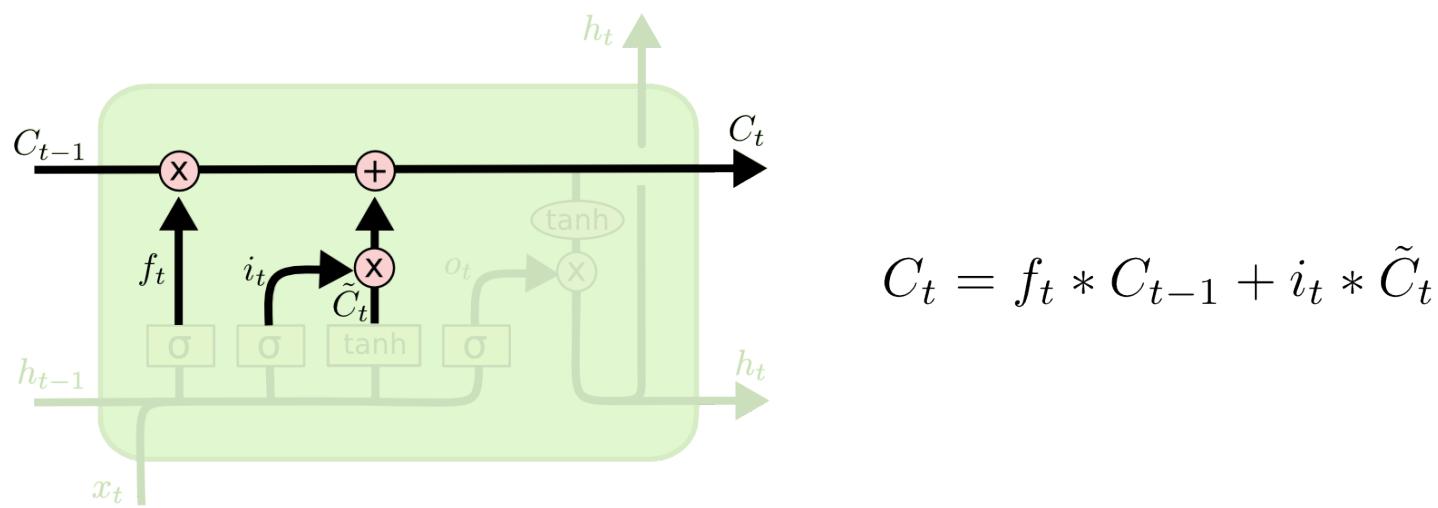
LSTM: Input gate



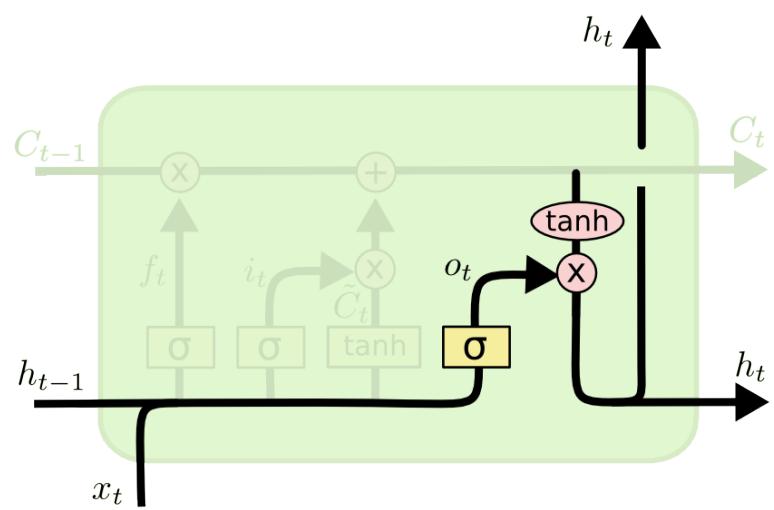
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

LSTM: Запись в память

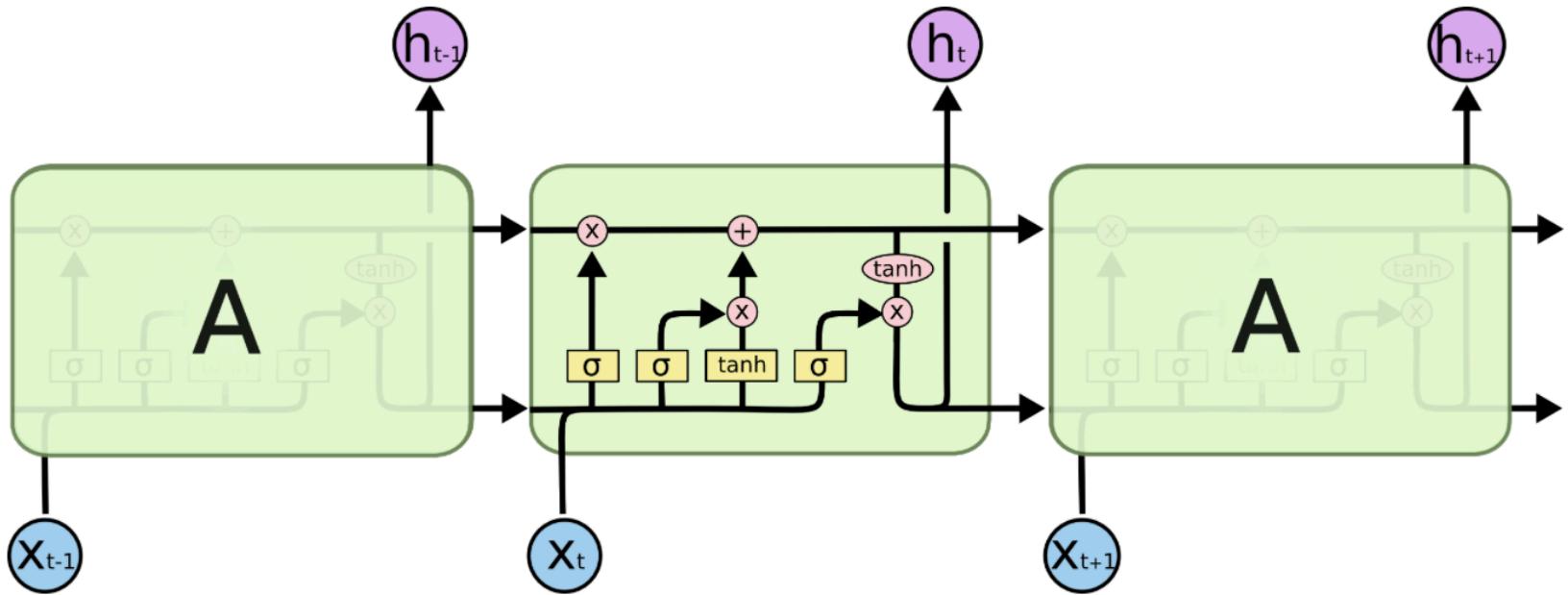


LSTM: Output gate



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

LSTM: все формулы вместе



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

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

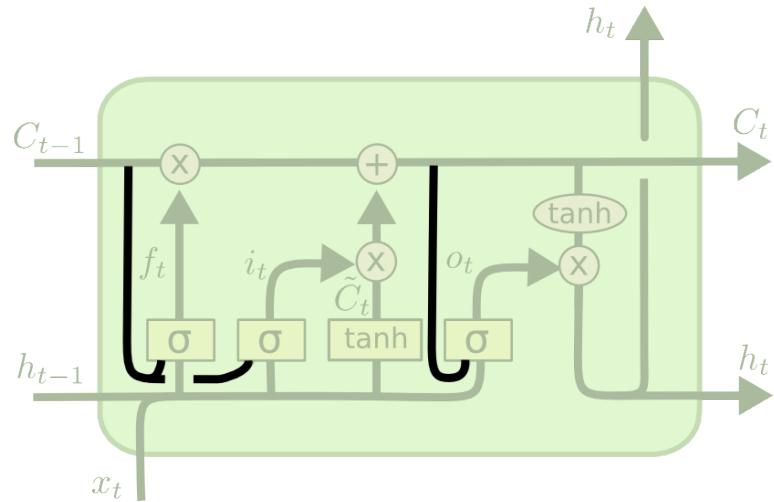
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

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

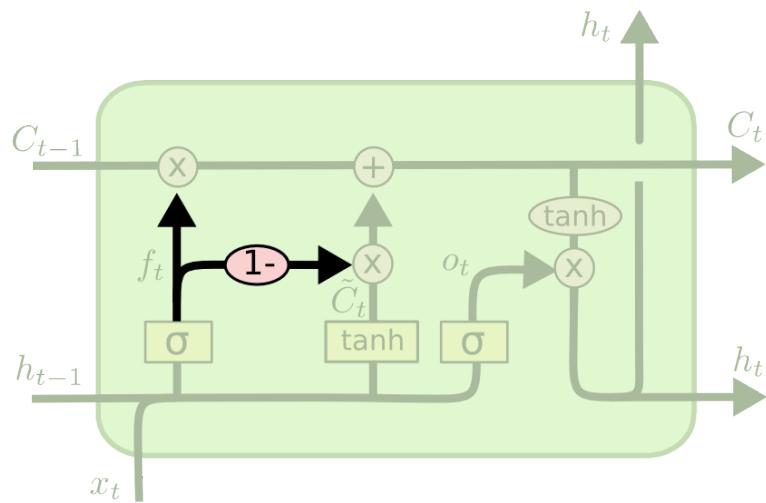
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

Другие вариации LSTM

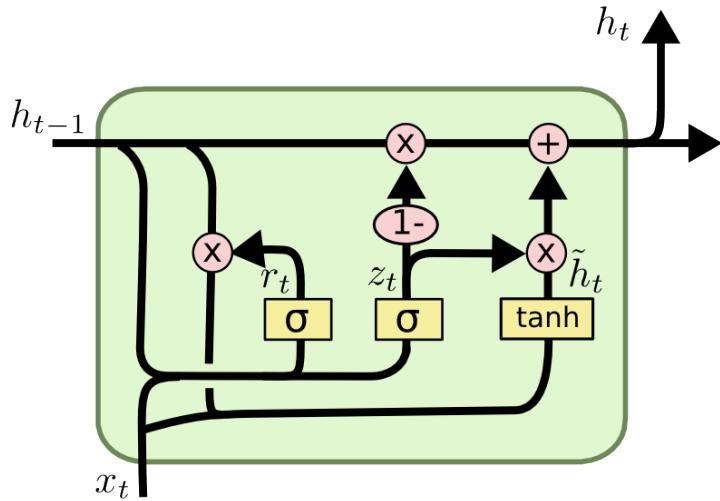


$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$



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

GRU – Gated Recurrent Unit



$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

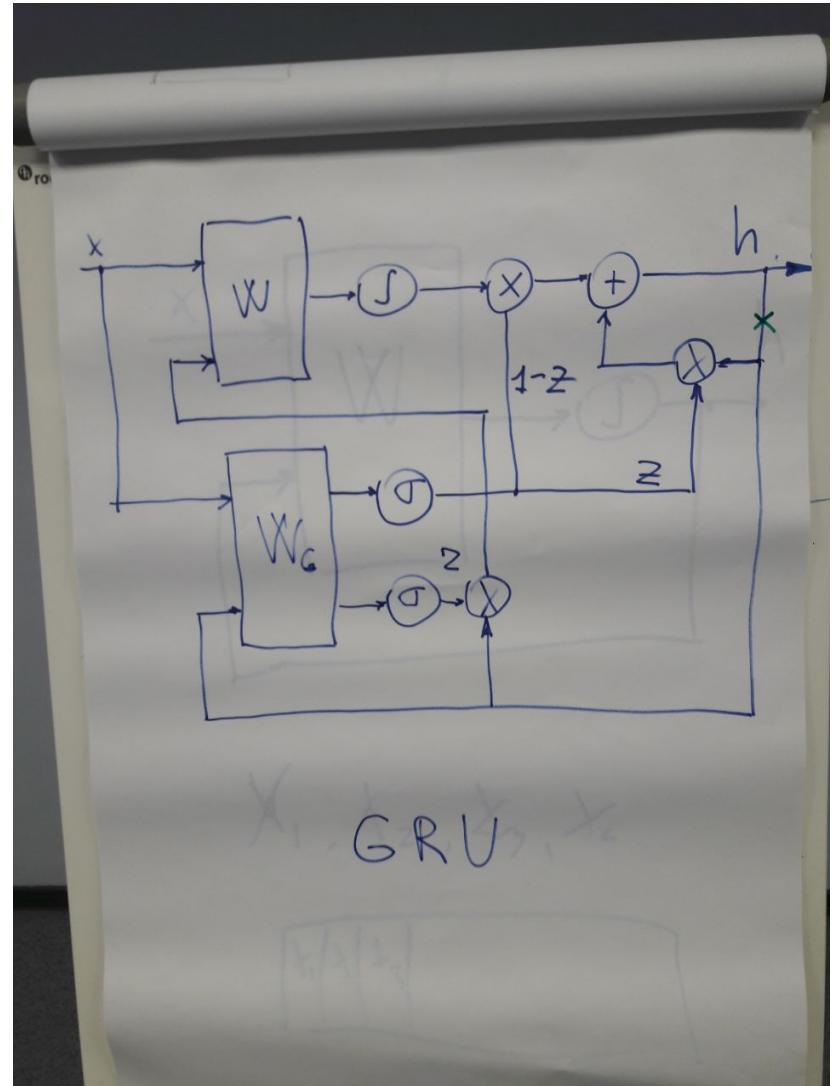
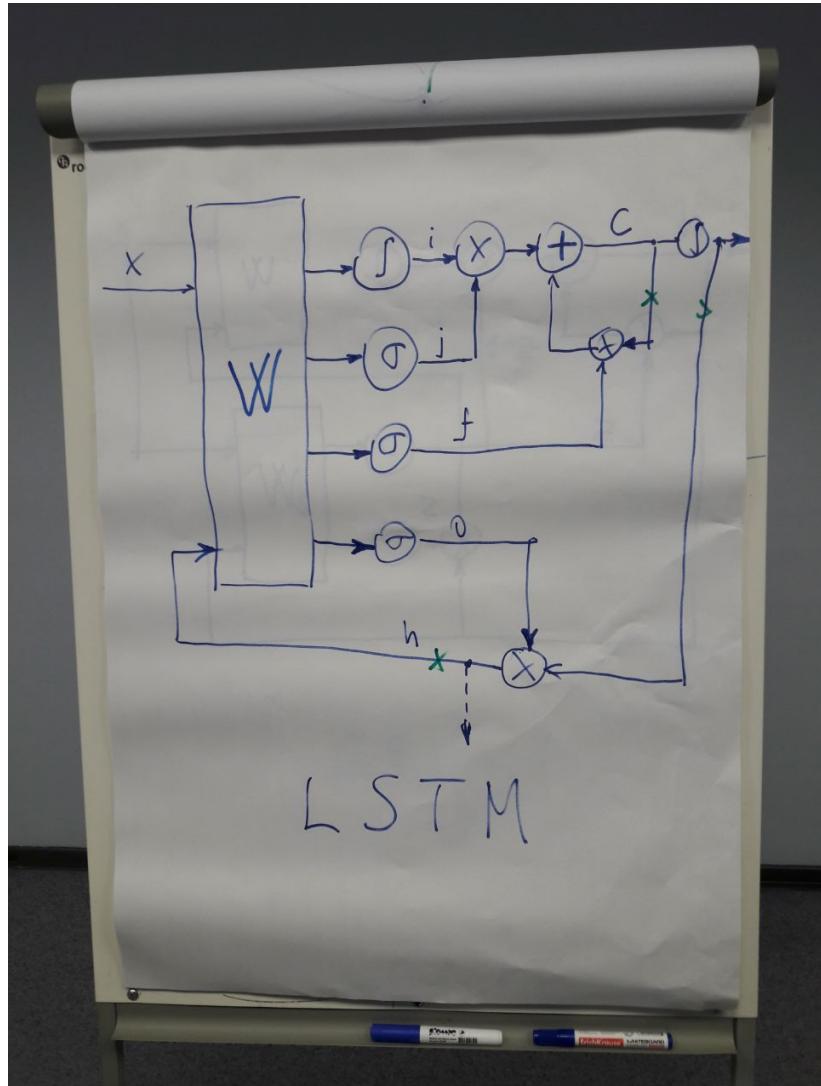
$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Более простая модель:

- Объединяем input и forget gate
- Объединяем cell state и hidden state
- Еще небольшие изменения

LSTM и GRU «по Анисимовичу»

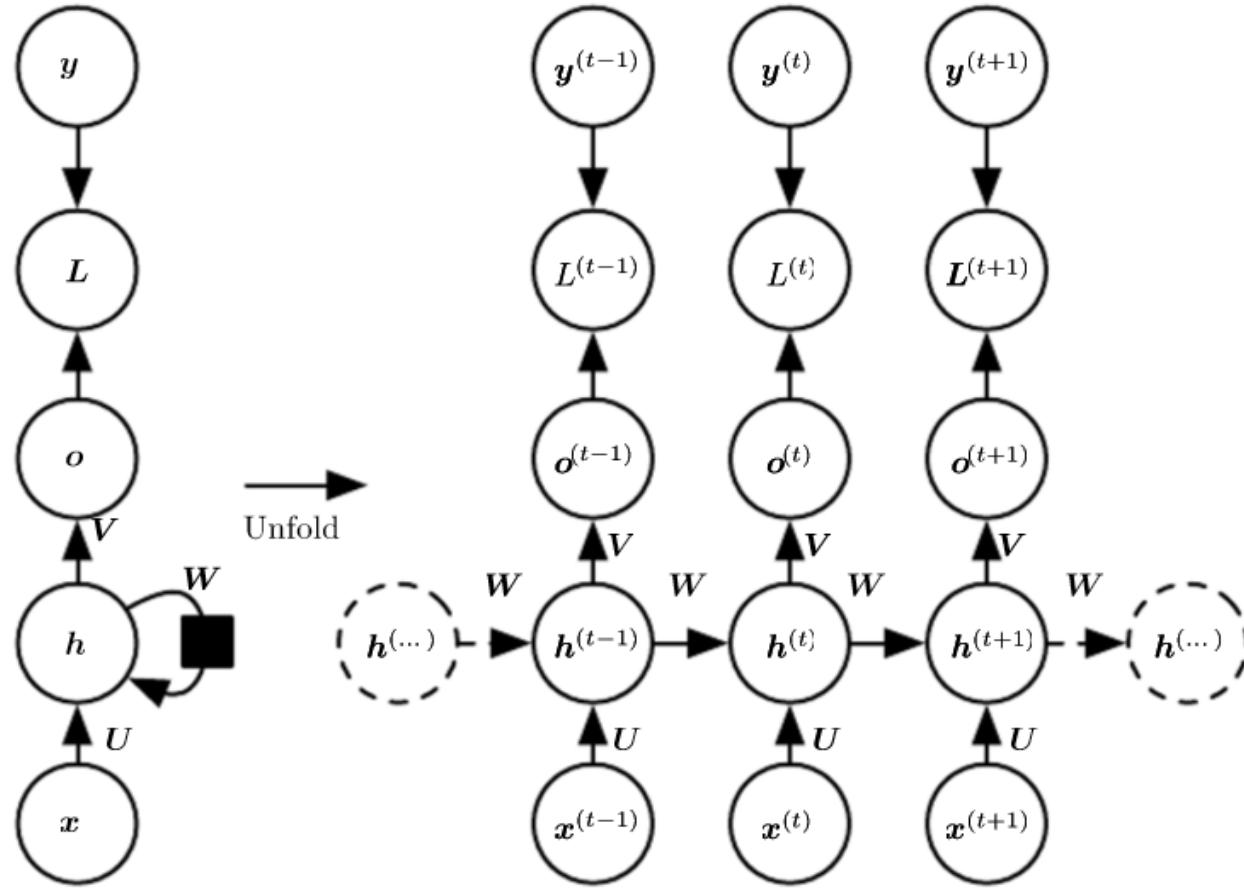


Другие вариации LSTM и GRU

- Проведены эксперименты по перебору возможных архитектур рекуррентных ячеек:
 - Обычно лучше всех работает LSTM с инициализацией forget-gate в единицу (ничего не забываем)
 - Без нелинейностей внутри ячеек все плохо
 - Без forget gate все очень плохо

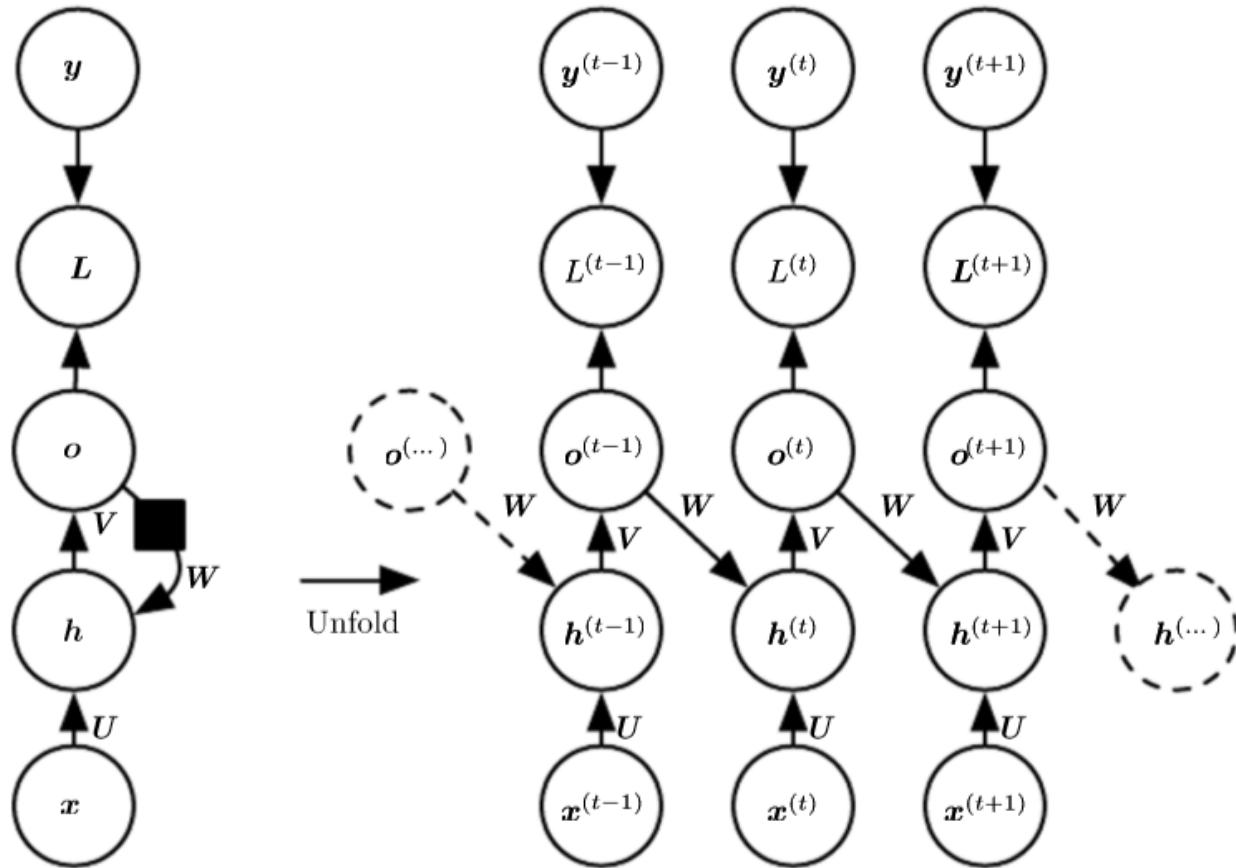
Варианты использования RNN

Варианты рекуррентных сетей



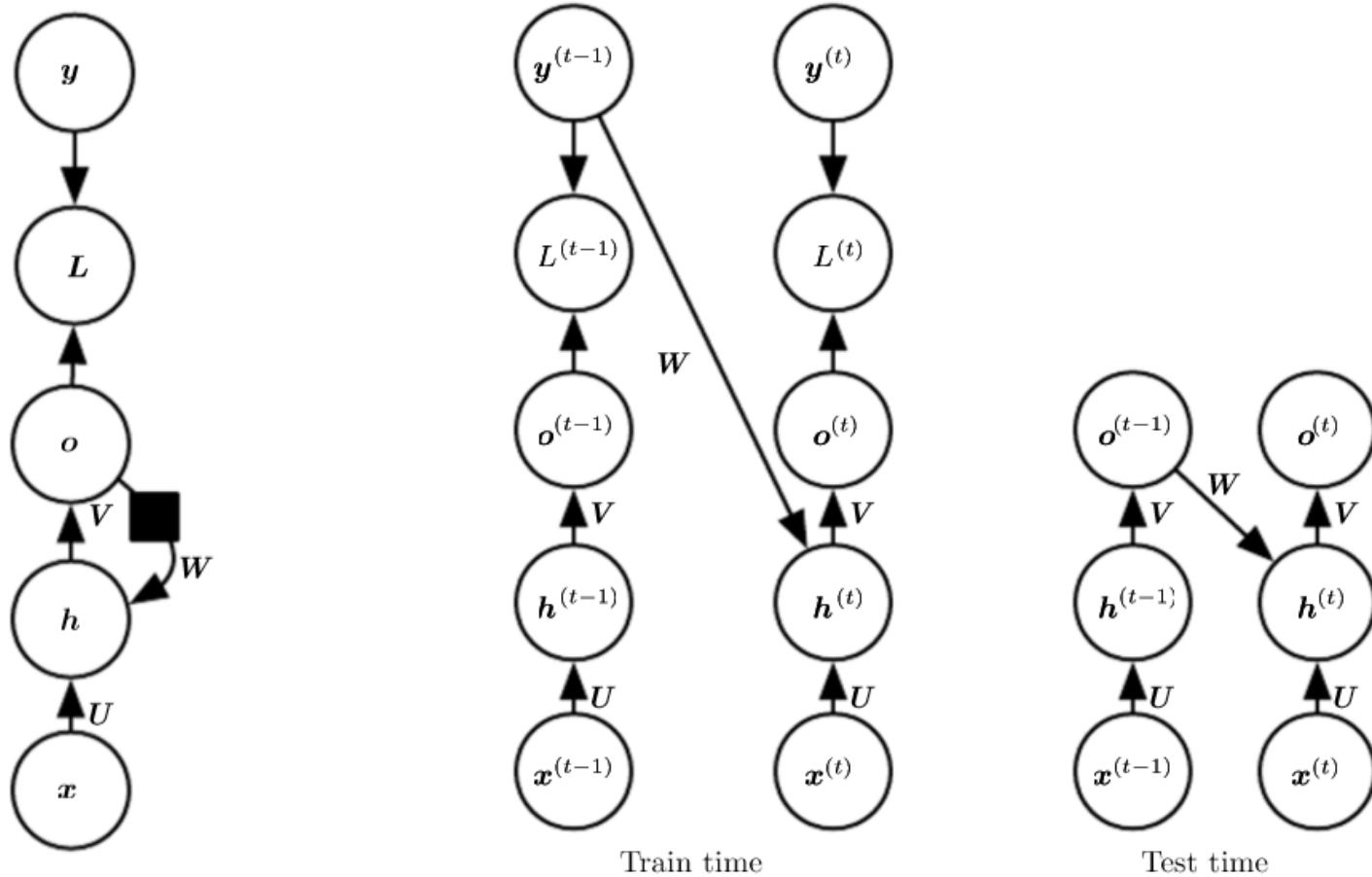
Сеть возвращает ответ на каждом временном шаге и имеет рекуррентные связи между внутренними состояниями

Варианты рекуррентных сетей



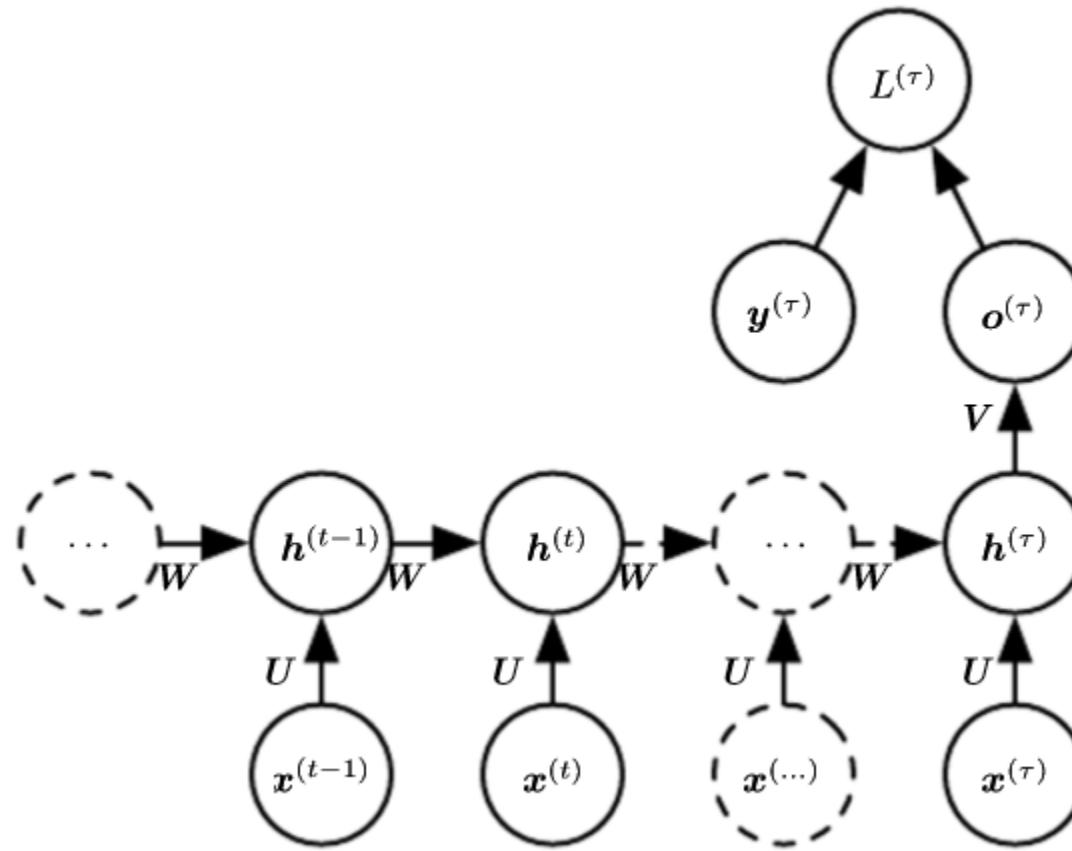
Сеть возвращает ответ на каждом временном шаге и имеет рекуррентную связь от выходного слоя к внутреннему слою

Teacher Forcing and Networks with Output Recurrence



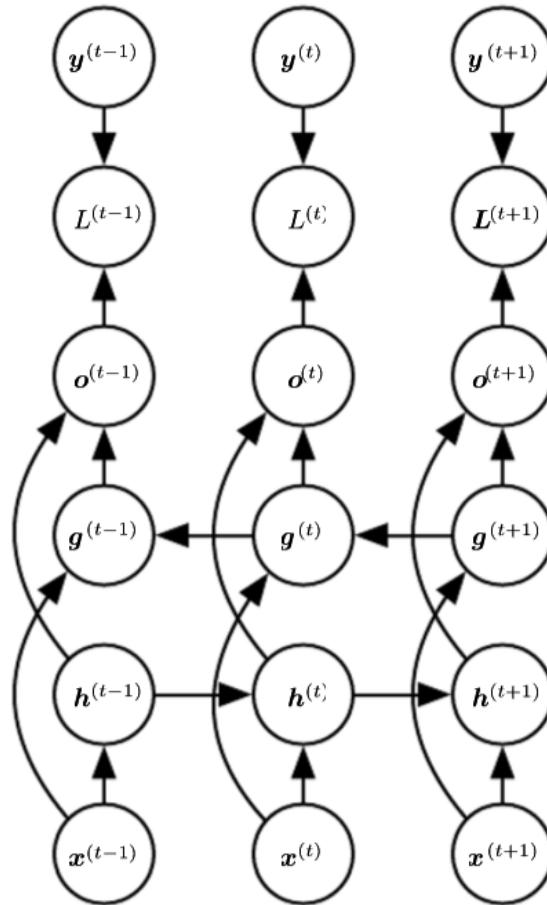
Сети, которые имеют рекуррентную связь от выходного слоя можно обучать более эффективно, передавая вместо обратной связи ответы из обучающей выборки.

Варианты рекуррентных сетей

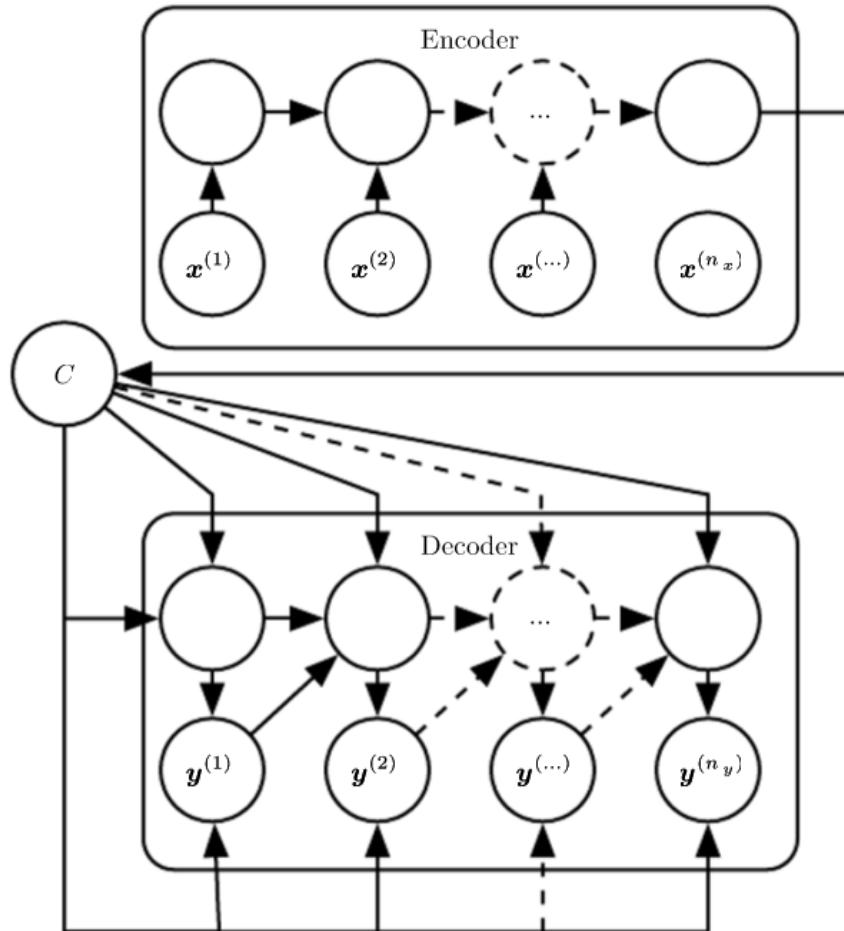


Сеть возвращает ответ после обработки всей последовательности и имеет рекуррентную связь между внутренними состояниями

Двунаправленная (Bidirectional) рекуррентная нейросеть

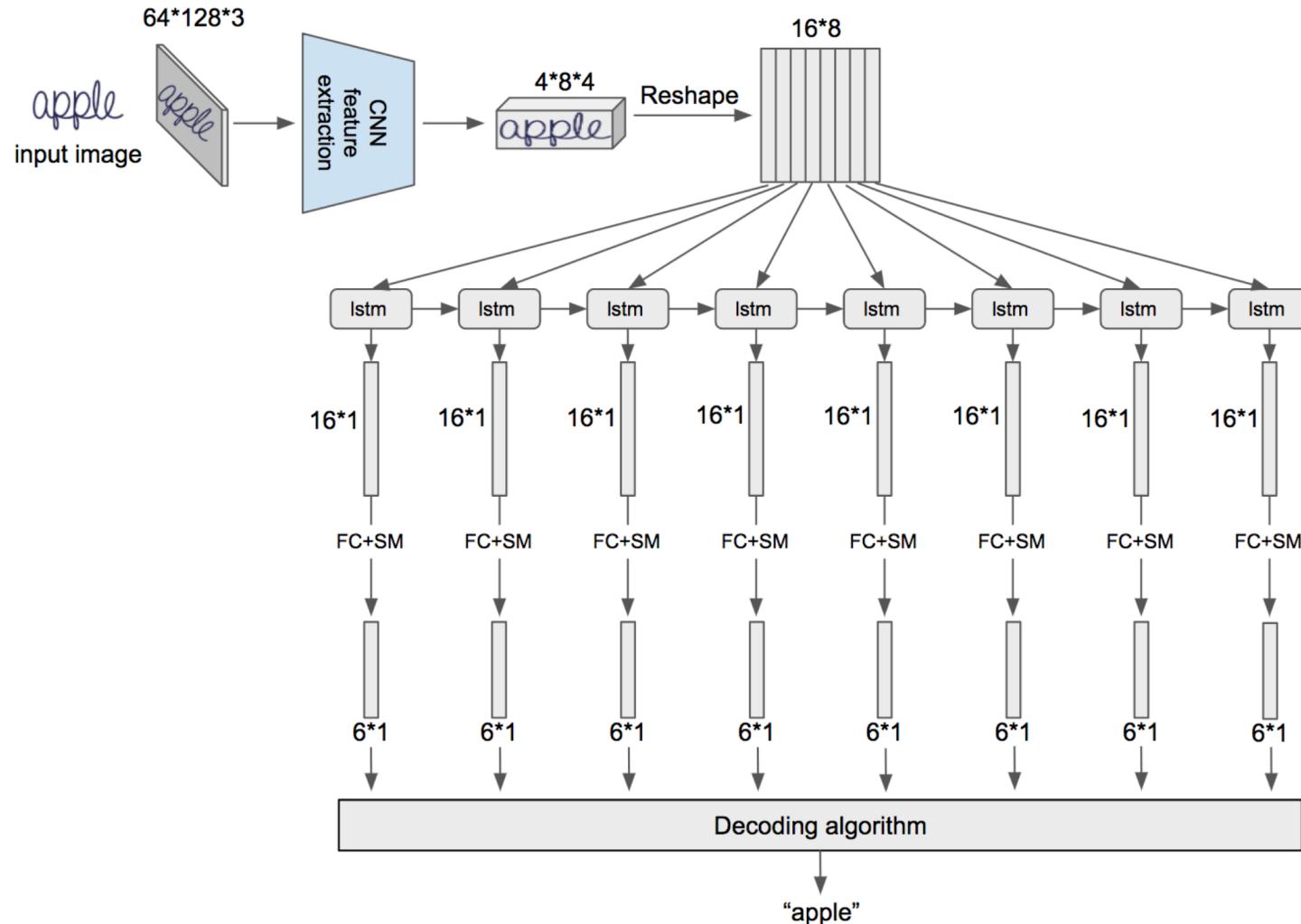


Encoder-Decoder Sequence-to-Sequence Architecture

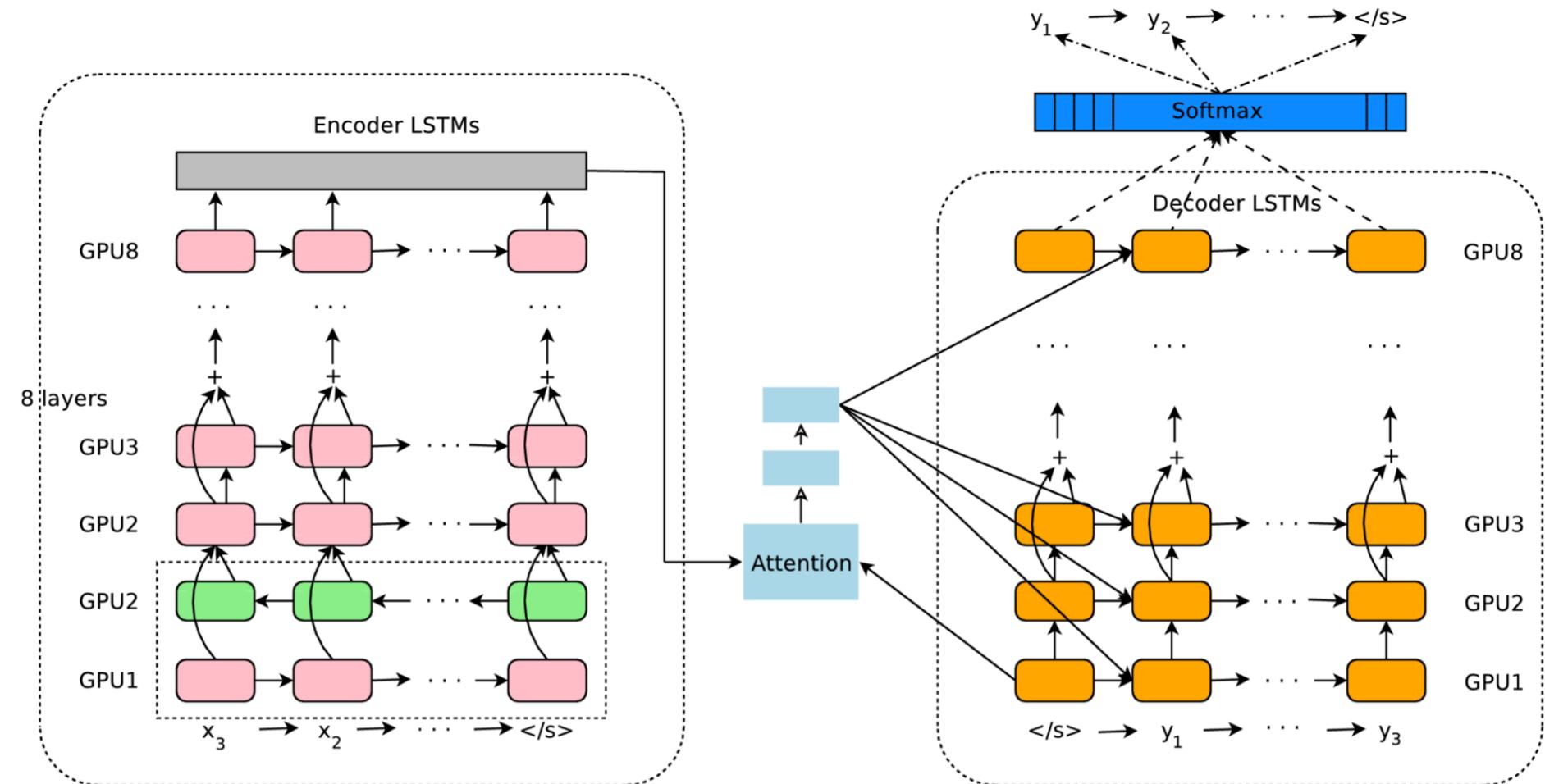


- Считываем последовательность в контекст C , затем декодируем его в выходную последовательность
- Проблема: Размера контекста C может быть недостаточно для обобщения всей последовательности

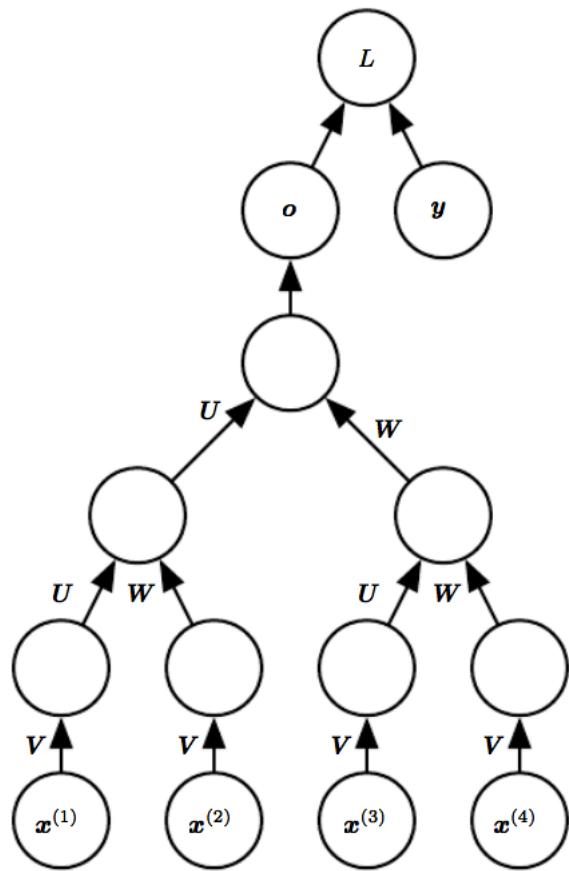
Optical Character Recognition



Machine Translation



Recursive Neural Networks



- Глубина сети $O(\log(t))$, а не $O(t)$
- Благодаря этому лучше находятся долгосрочные зависимости
- Архитектура может быть:
 - сбалансированным бинарным деревом
 - продиктована предметной областью (например, дерево синтаксического разбора предложения)

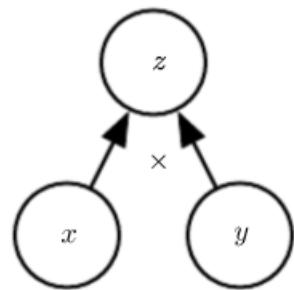
Источники

- Deep Learning – Ian Goodfellow, Yoshua Bengio and Aaron Courville – глава Sequence Modelling
<http://www.deeplearningbook.org/>
- Simon says - LSTM - что, как и почему
<https://www.youtube.com/watch?v=wYI7RZz4Rz0>
- <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- <http://proceedings.mlr.press/v37/jozefowicz15.pdf>
- <http://proceedings.mlr.press/v37/jozefowicz15.pdf>

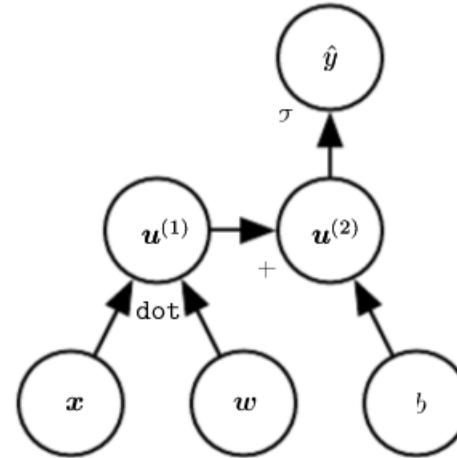
Спасибо за внимание

Дополнительные слайды
по backprop и
вычислительным графикам

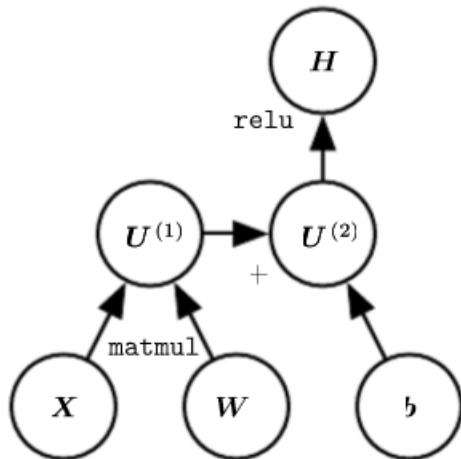
Графы вычислений



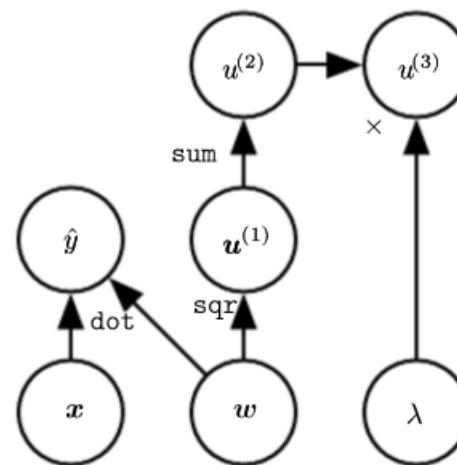
$$z = xy$$



$$\hat{y} = \sigma(x^\top w + b)$$



$$\mathbf{H} = \max\{0, \mathbf{X}\mathbf{W} + \mathbf{b}\}$$



$$\lambda \sum_i w_i^2.$$

Chain Rule (производная сложной функции)

Пусть $y = g(x)$ $z = f(g(x)) = f(y)$

Тогда:

$$\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$$

Для случая $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

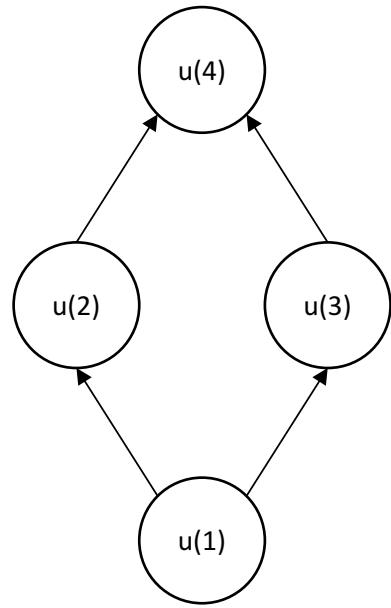
Или в матричной форме:

$$\nabla_x z = \left(\frac{\partial y}{\partial x} \right)^\top \nabla_y z$$

Для случая тензоров X и Y (j пробегает по всем индексам тензора):

$$\nabla_X z = \sum_j (\nabla_X Y_j) \frac{\partial z}{\partial Y_j}$$

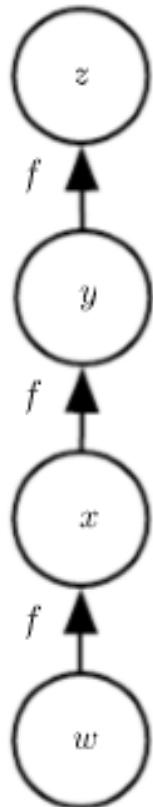
Идея алгоритма backprop



$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}$$

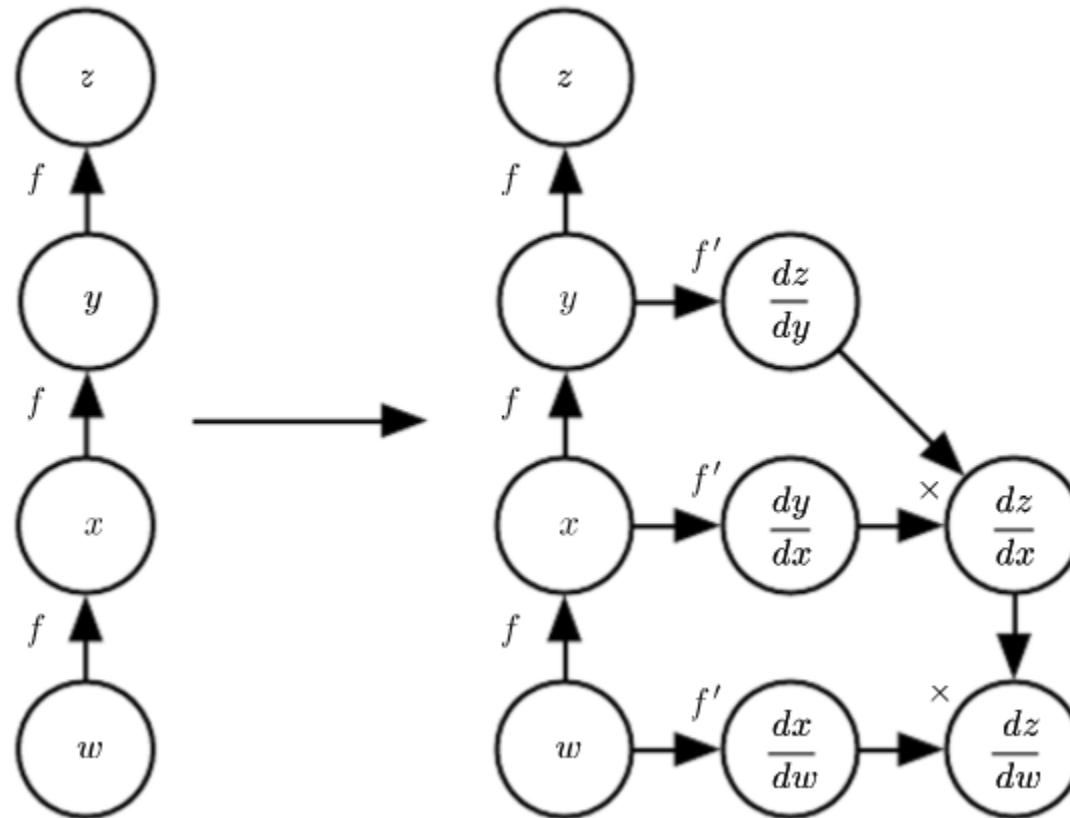
Суммирование по всем детям $u^{(j)}$

Граф вычислений для нахождения производных



$$\begin{aligned}\frac{\partial z}{\partial w} &= \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial w} \\ &= f'(y) f'(x) f'(w) \\ &= f'(f(f(w))) f'(f(w)) f'(w)\end{aligned}$$

Граф вычислений для нахождения производных



Граф вычислений для MLP с одним внутренним слоем

