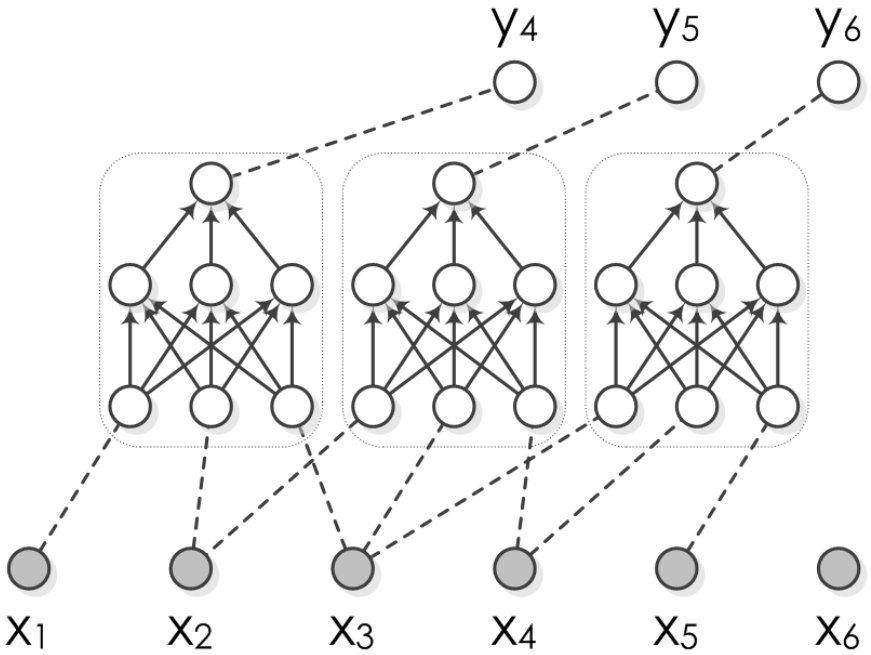
Стандартные Feed-Forward сети

- Используем стандартные сети прямого распространения
- Применим одну и ту же сеть по скользящему окну
- Проблемы?



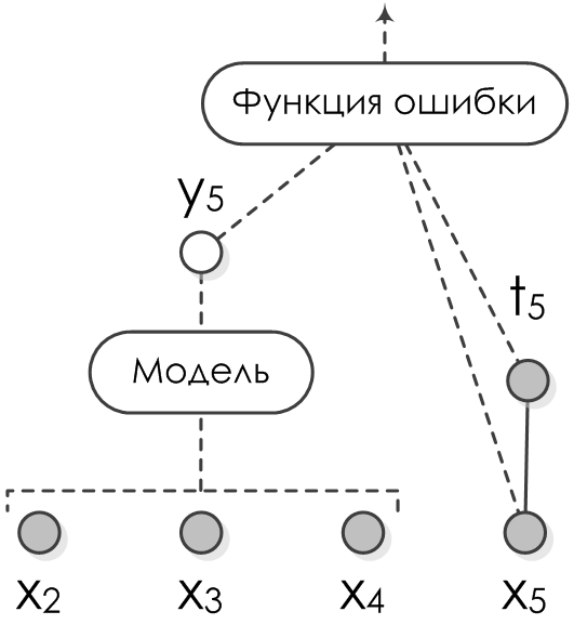
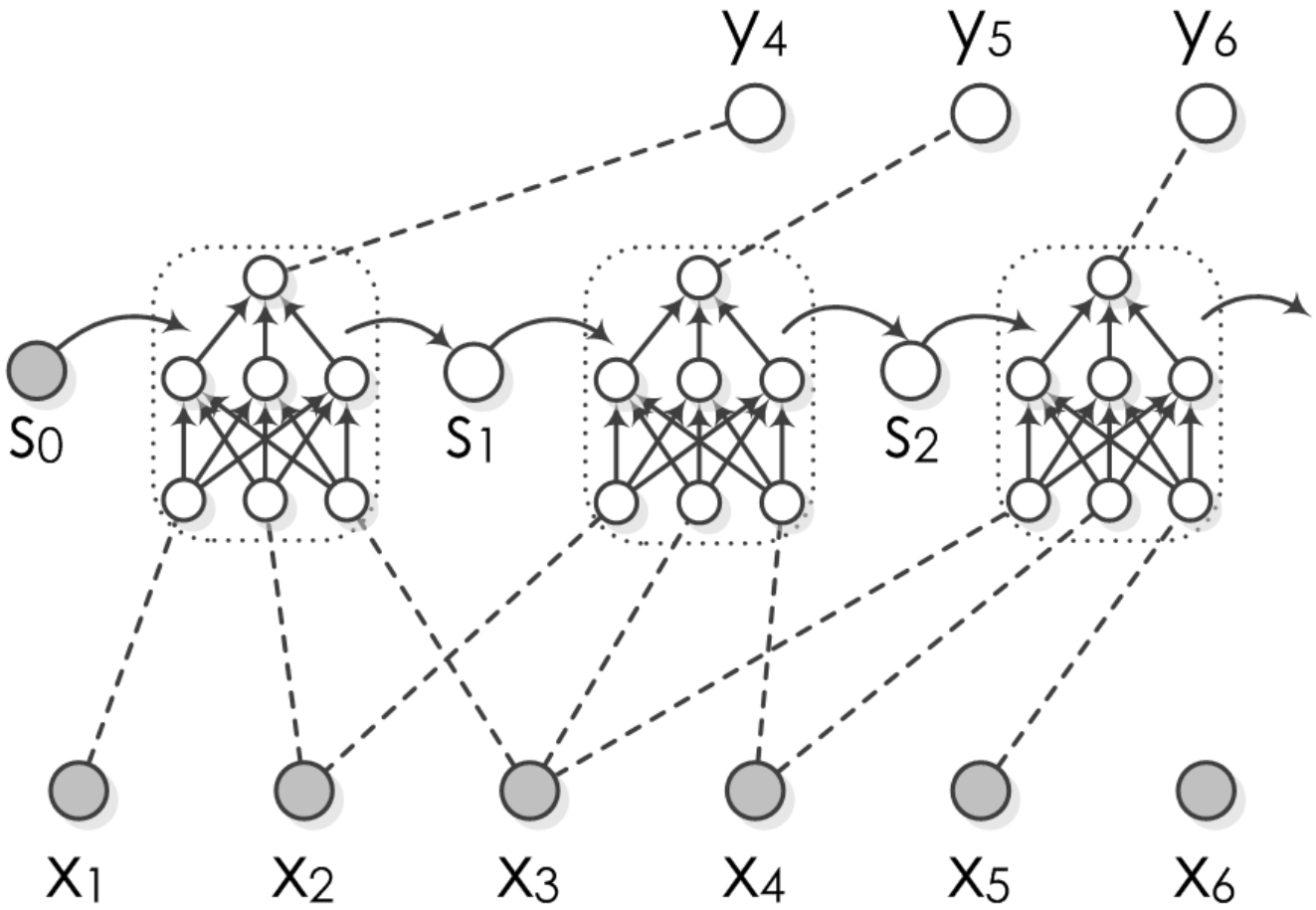
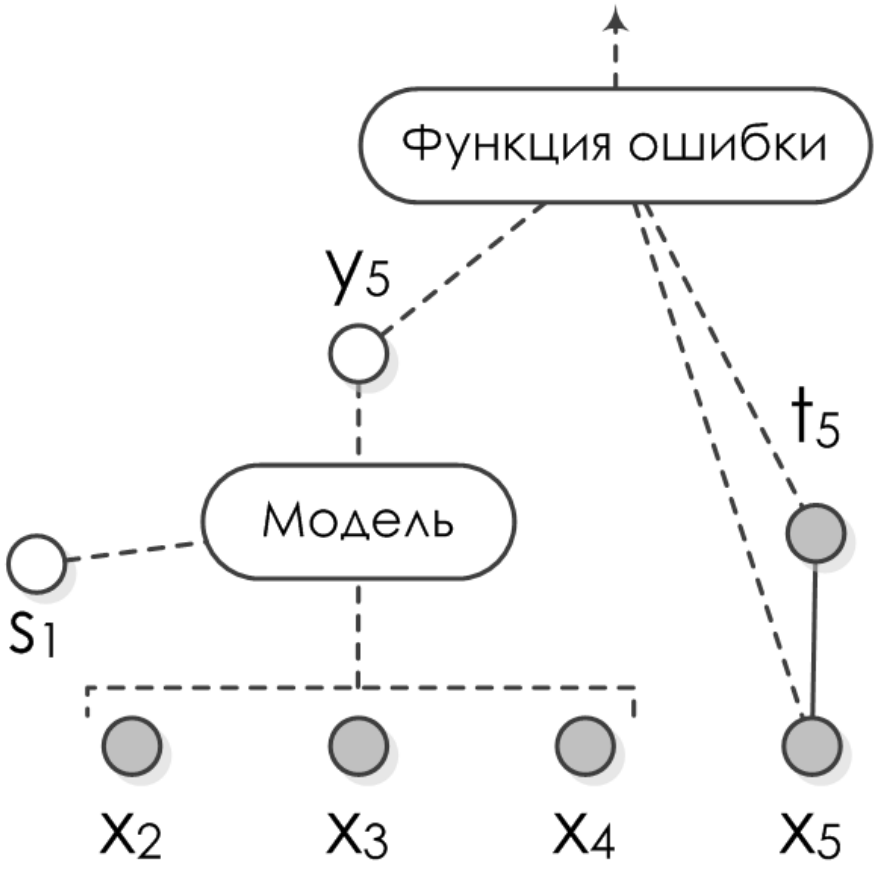


Image credit: https://logic.pdmi.ras.ru/~sergey/teaching/mlhse17/38-dl5-rnn.pdf

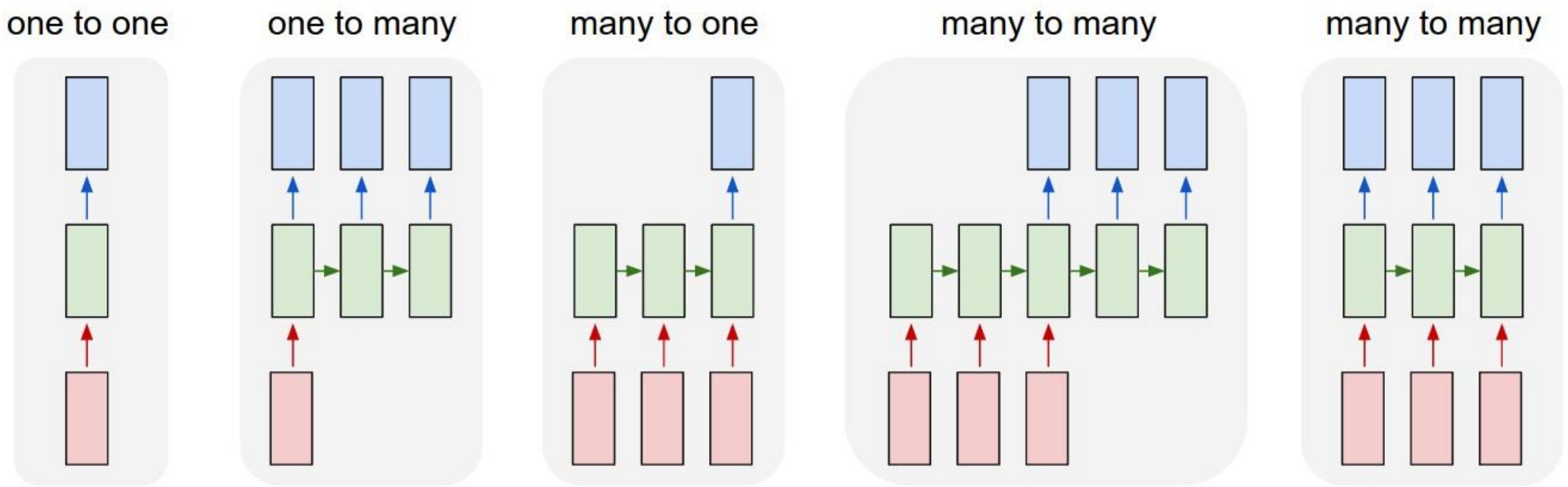
Рекуррентные сети - идея



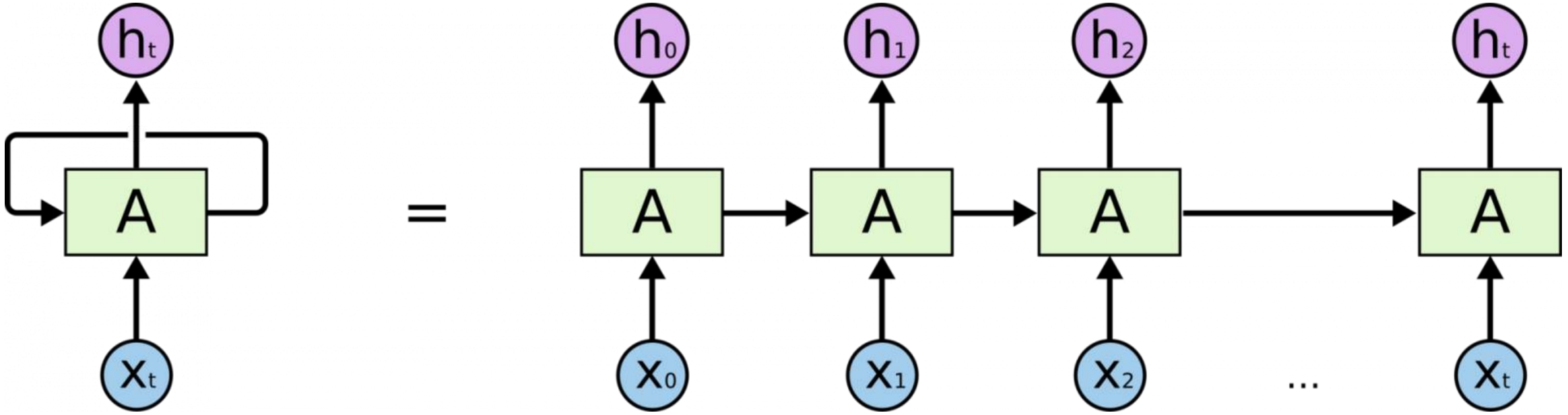


Типы задач

- Временные ряды (котировки, объёмы, температура, …)
- Сигналы (технические, музыка, …)
- Текст

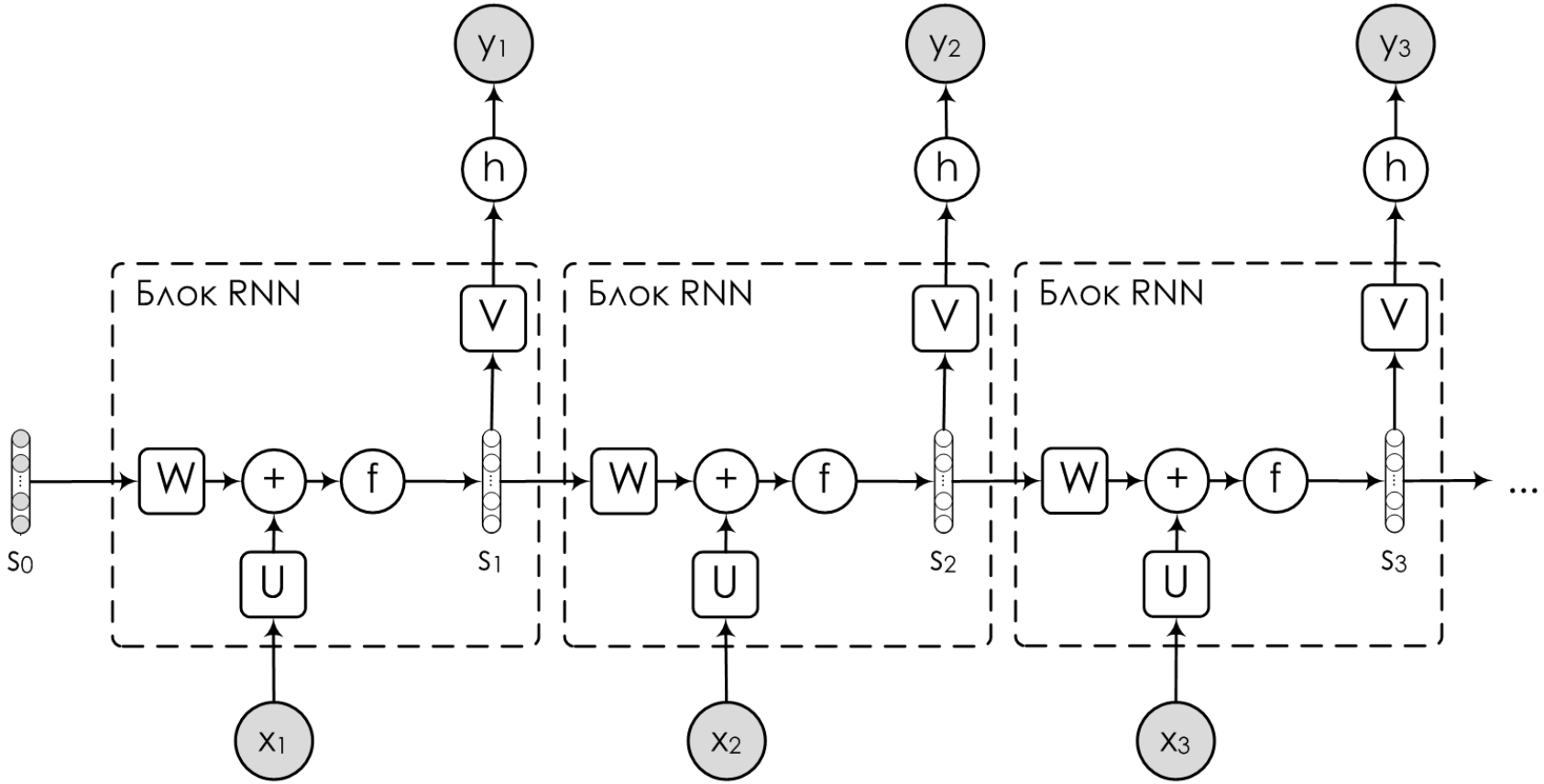


Vanilla RNN



- Как хранить состояние?
- Как обновлять состояние?
- Как предобрабатывать данные?
- Как вычислять градиенты?

Vanilla RNN (Simple RNN)

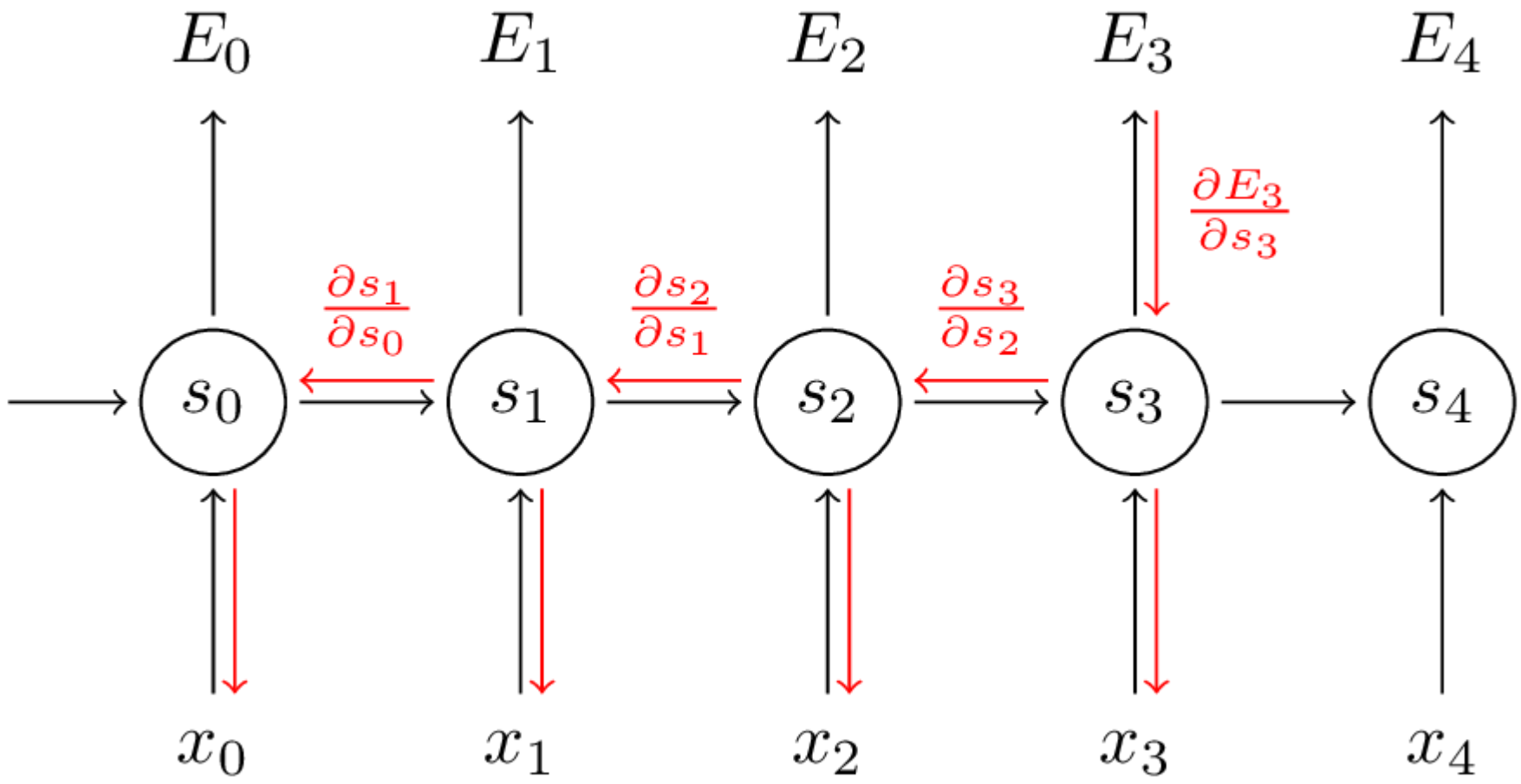


$a_t = b + W\, s_{t-1} + U\, x_t,$
$s_t = f(a_t),$
$o_t = c + V\, s_t,$
$y_t = h(o_t),$

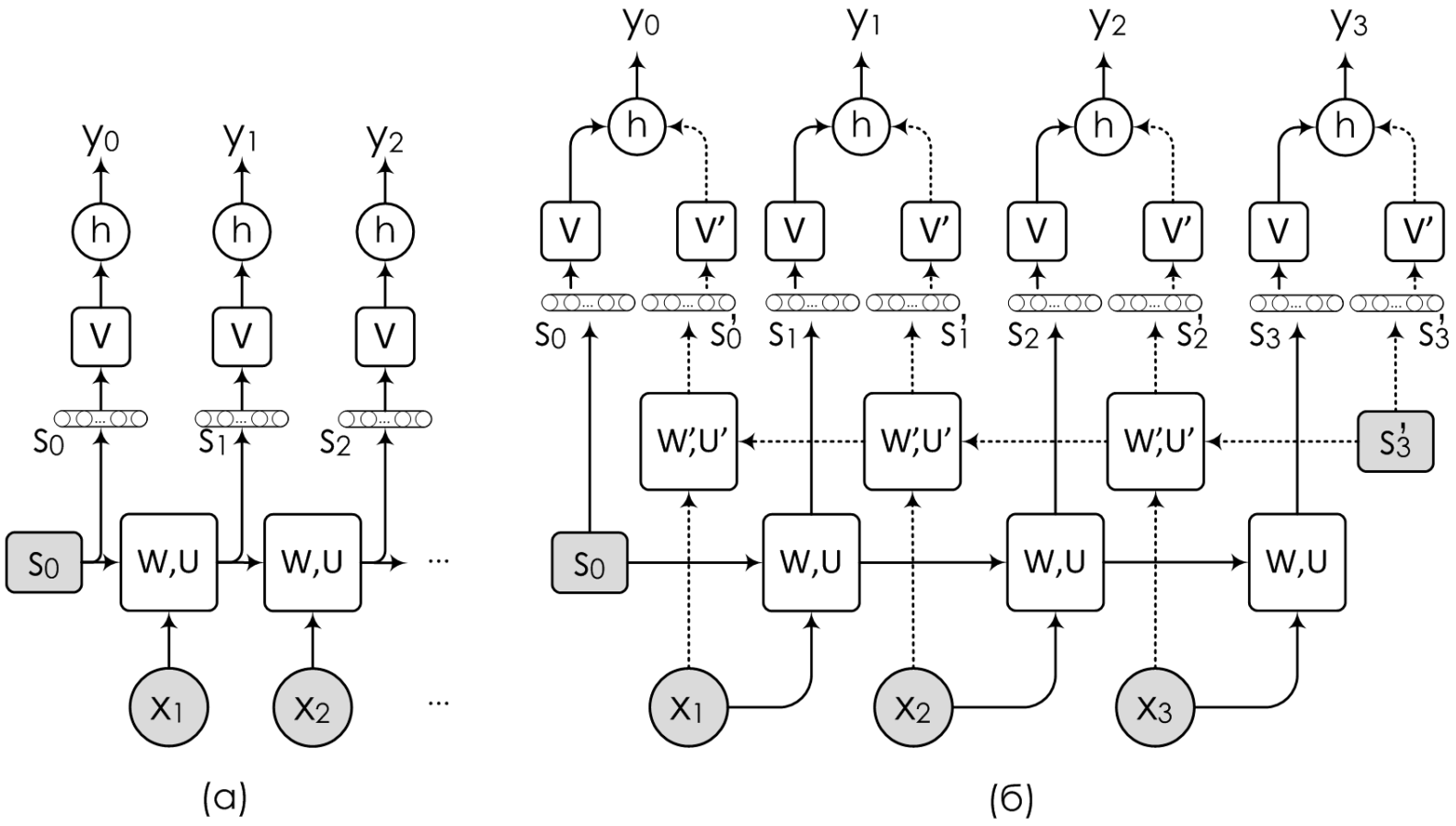
Предобработка данных

- Для большинства архитектур важно отнормировать данные, иначе будут серьёзные проблемы с градиентами
- Используйте стандартные методы, их достаточно (MinMaxScaler, StandardScaler)
- Если в датасете много рядов, каждый необходимо нормировать отдельно

BPTT – BackPropagation Through Time

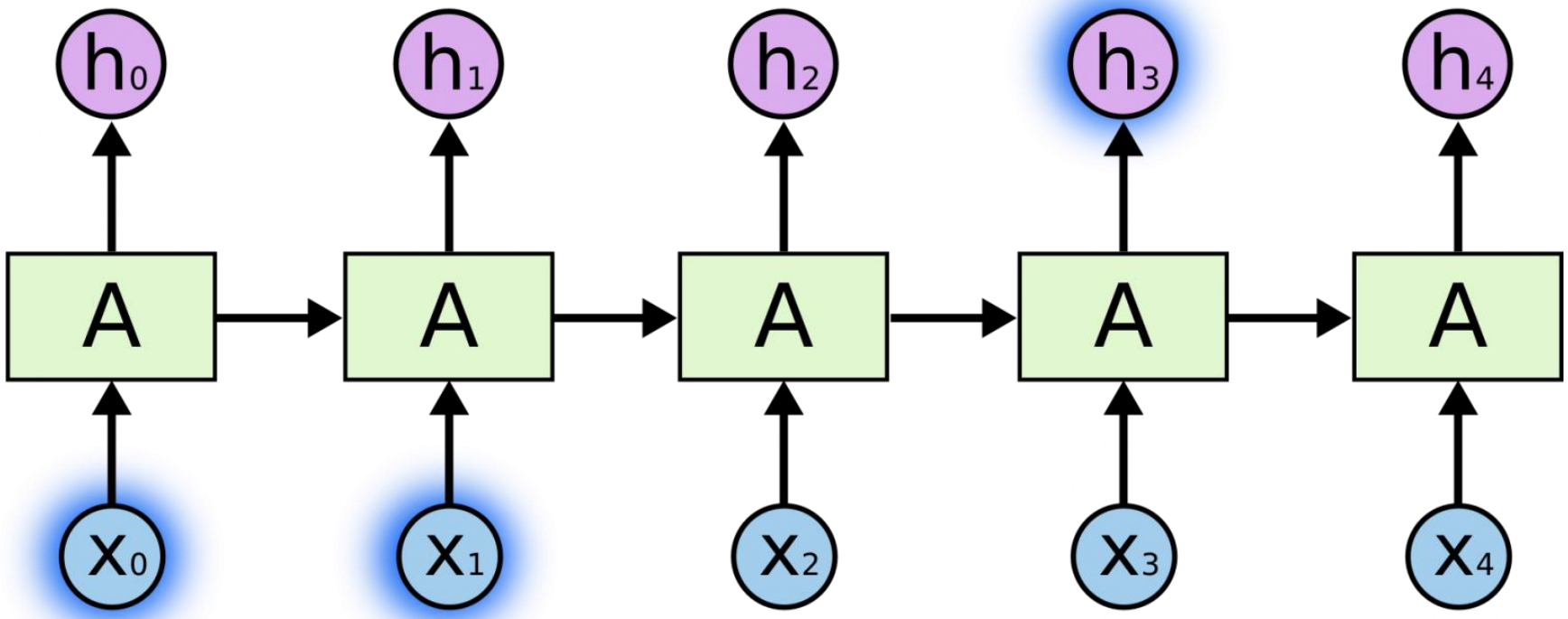


Двунаправленные архитектуры

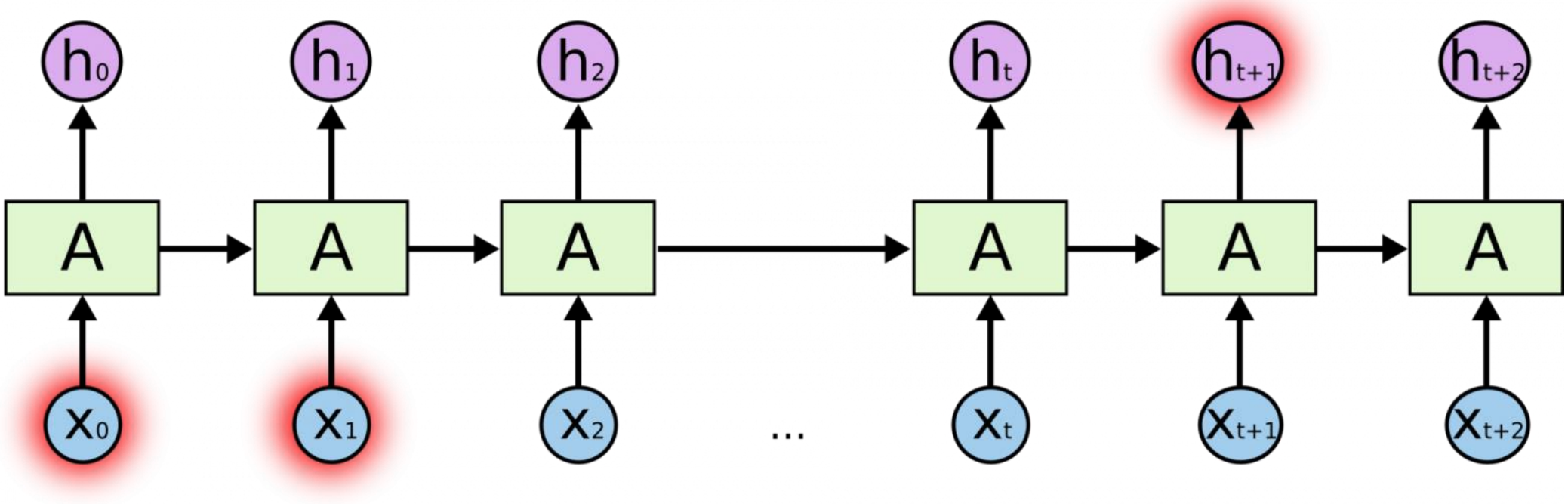


$$\mathbf{s}_t = \sigma\left(\mathbf{b} + W\mathbf{s}_{t-1} + U\mathbf{x}_t\right),$$
$$\mathbf{s}'_t = \sigma\left(\mathbf{b}' + W'\mathbf{s}'_{t+1} + U'\mathbf{x}_t\right),$$
$$o_t = c + V\mathbf{s}_t + V'\mathbf{s}'_t,$$
$$\mathbf{y}_t = h\left(o_t\right).$$

Память в RNN



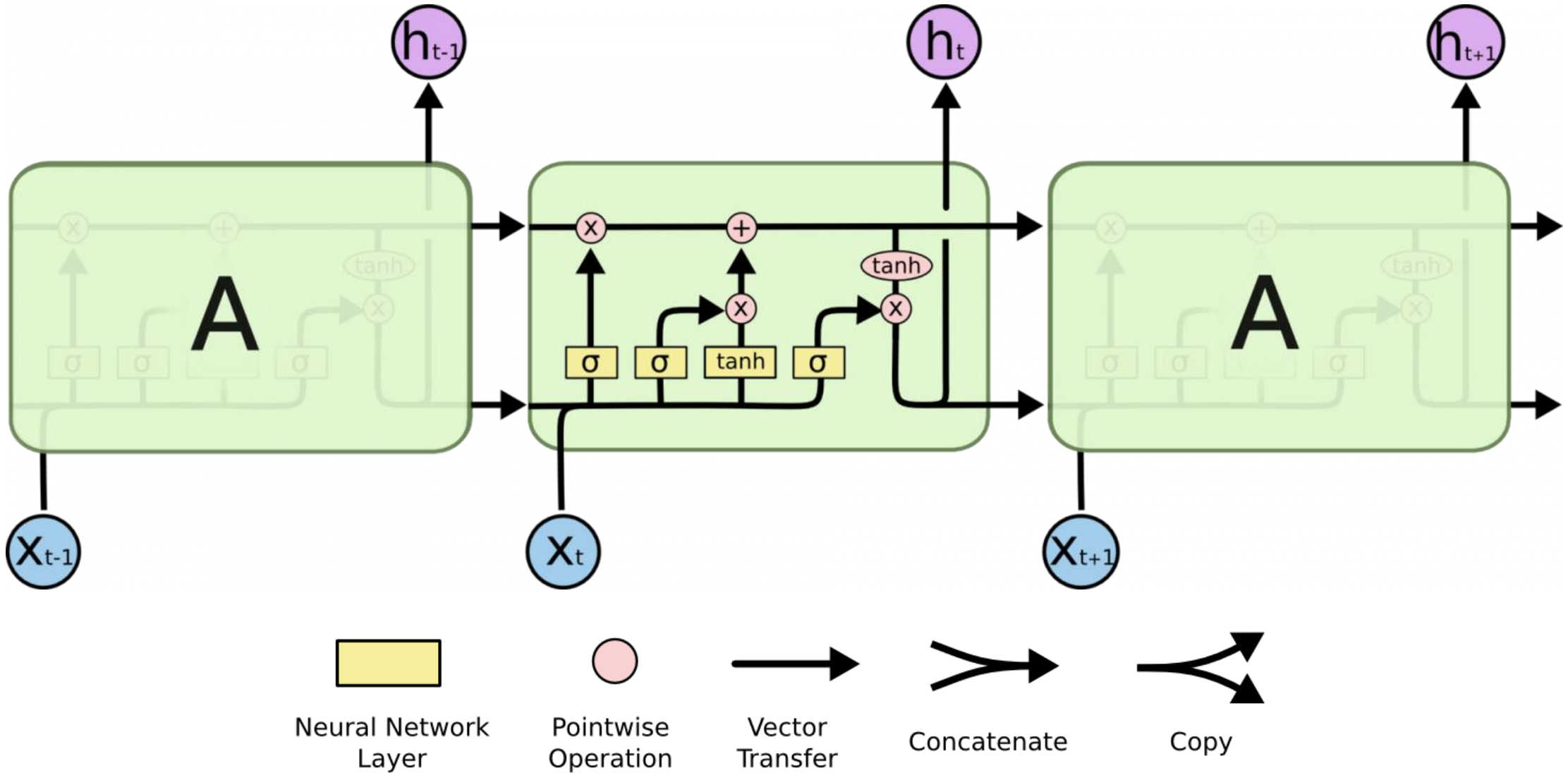
Долговременная память в RNN



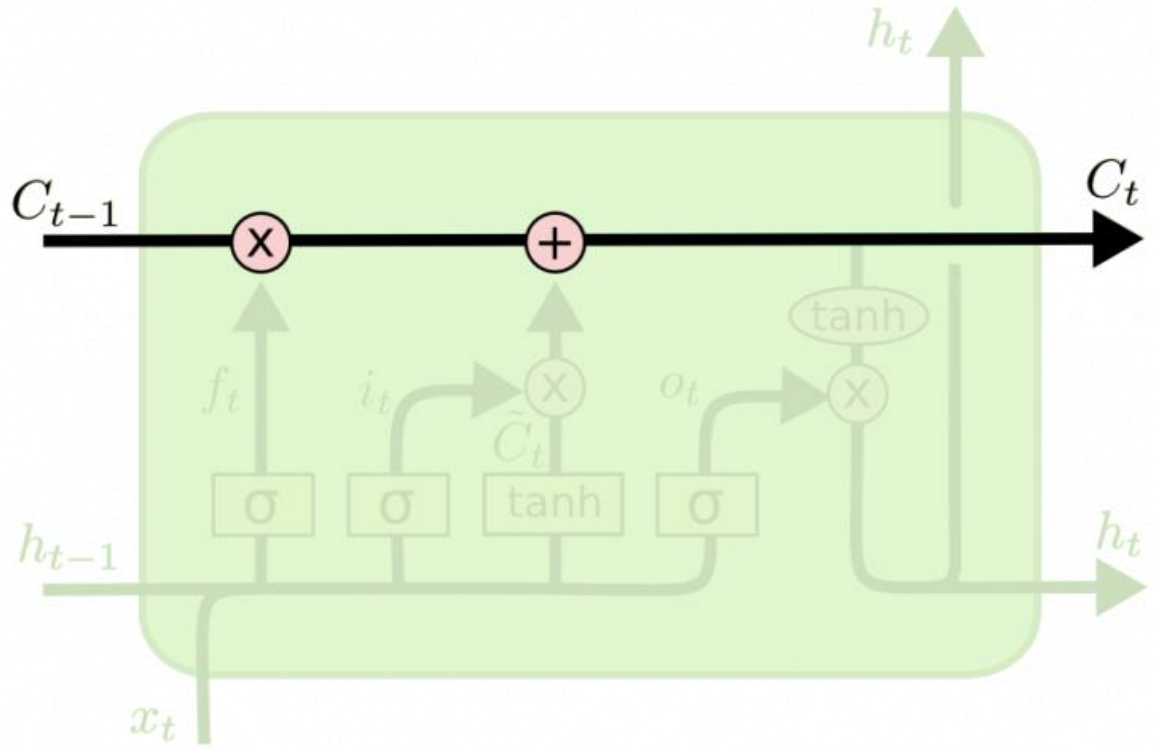
Память в RNN

- RNN способна запоминать последовательности
- Её память ограничена, притом весьма коротким горизонтом
- Градиенты затухают экспоненциально по времени
- Для более эффективной памяти нужны архитектурные изменения

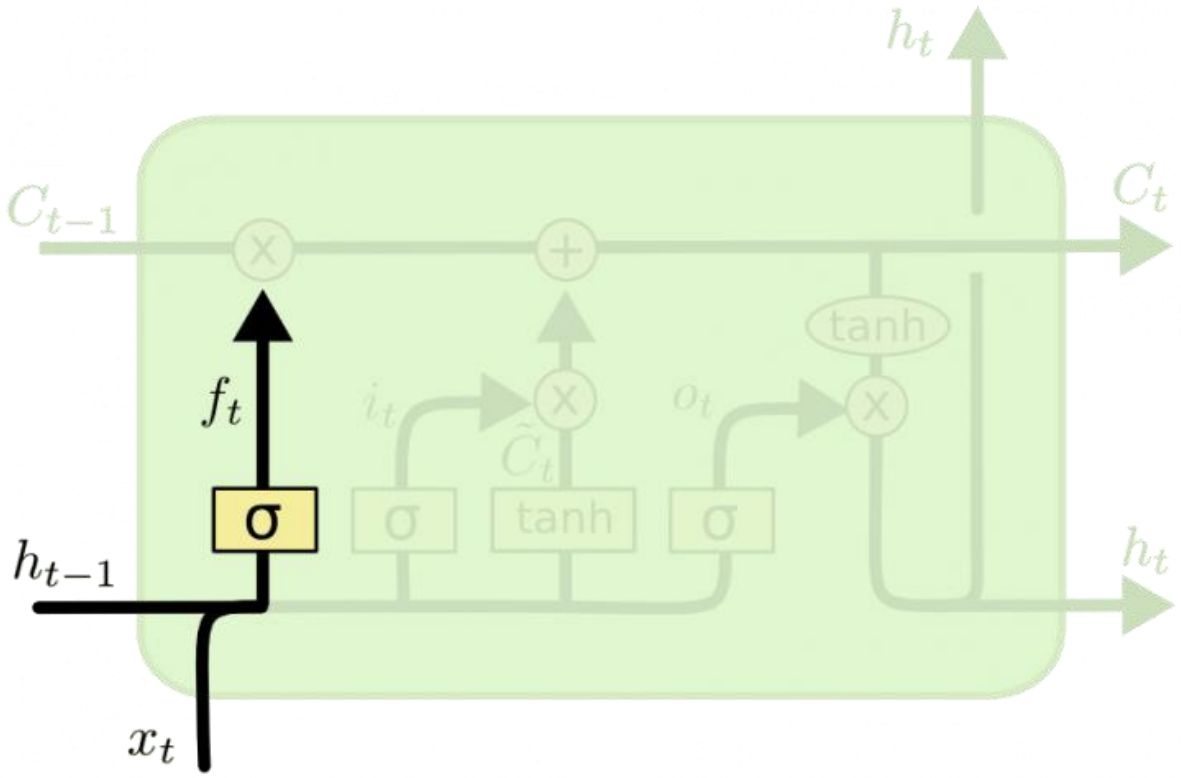
LSTM


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

Состояние ячейки

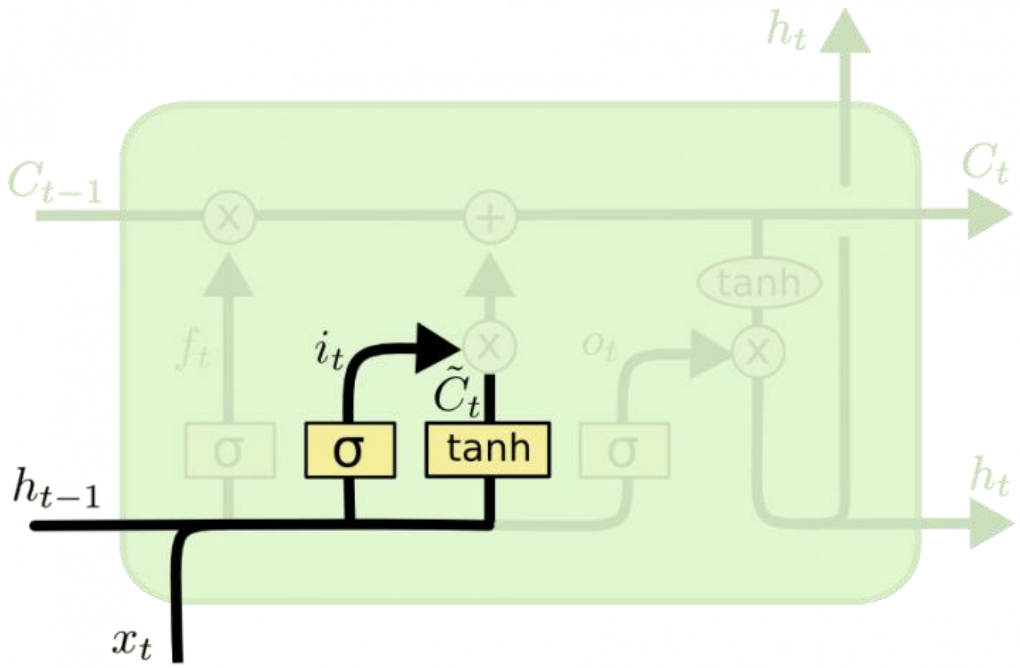


Forget gate



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

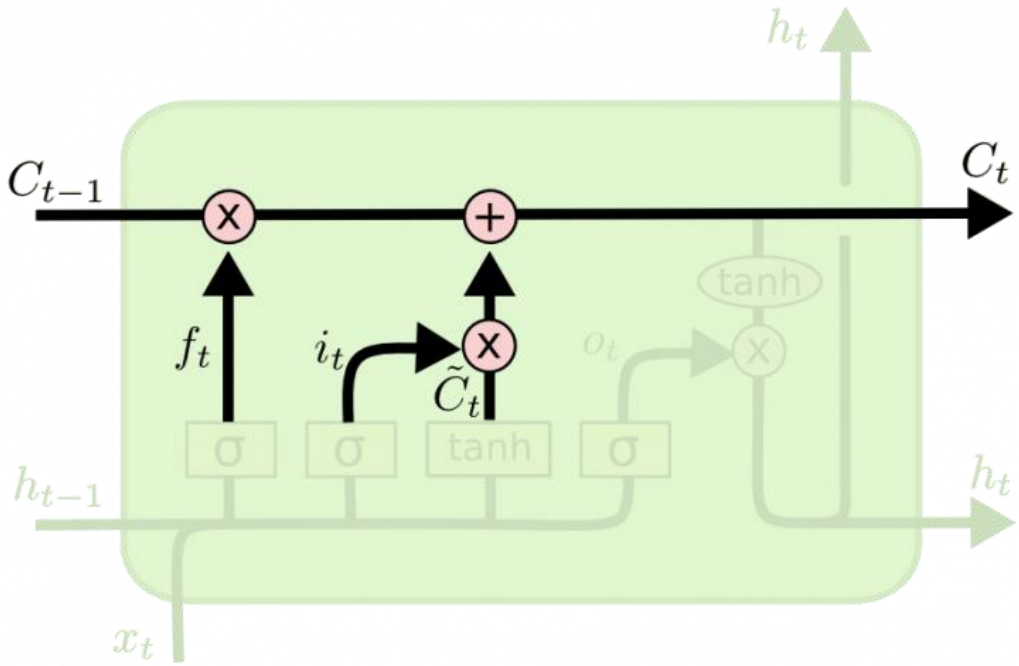
Input gate



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

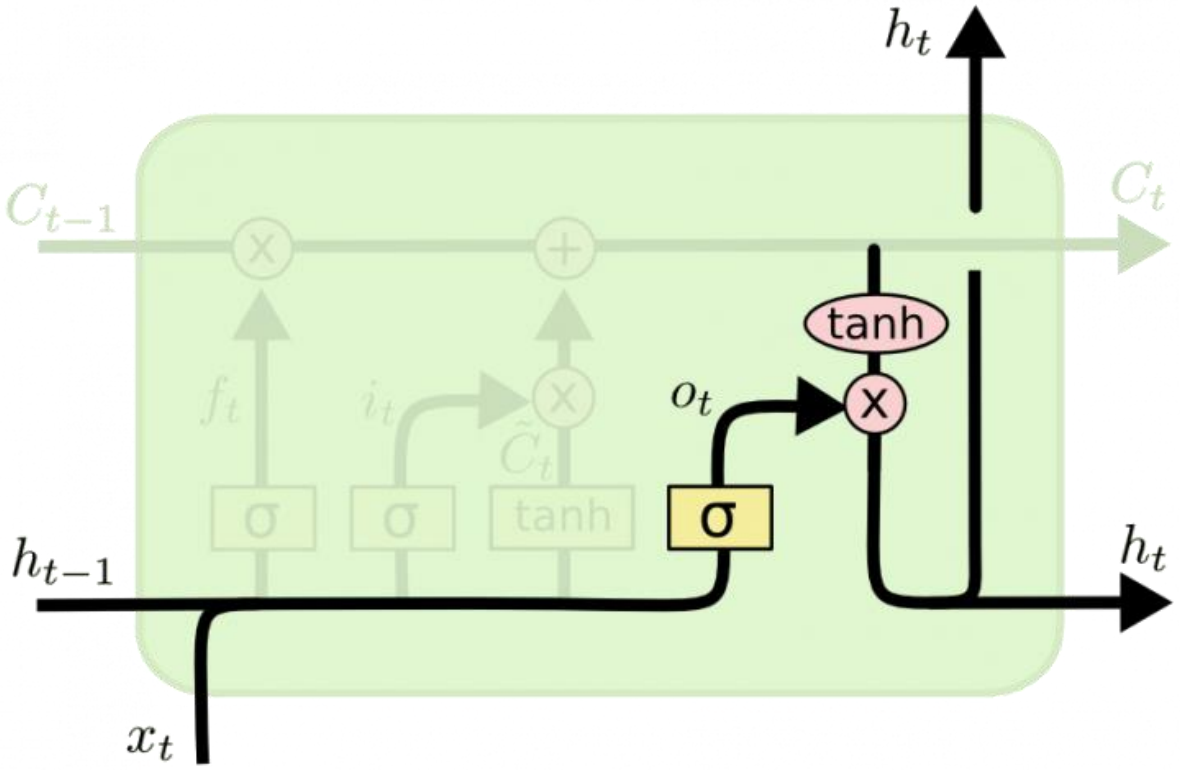
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Обновление состояния



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

Output gate



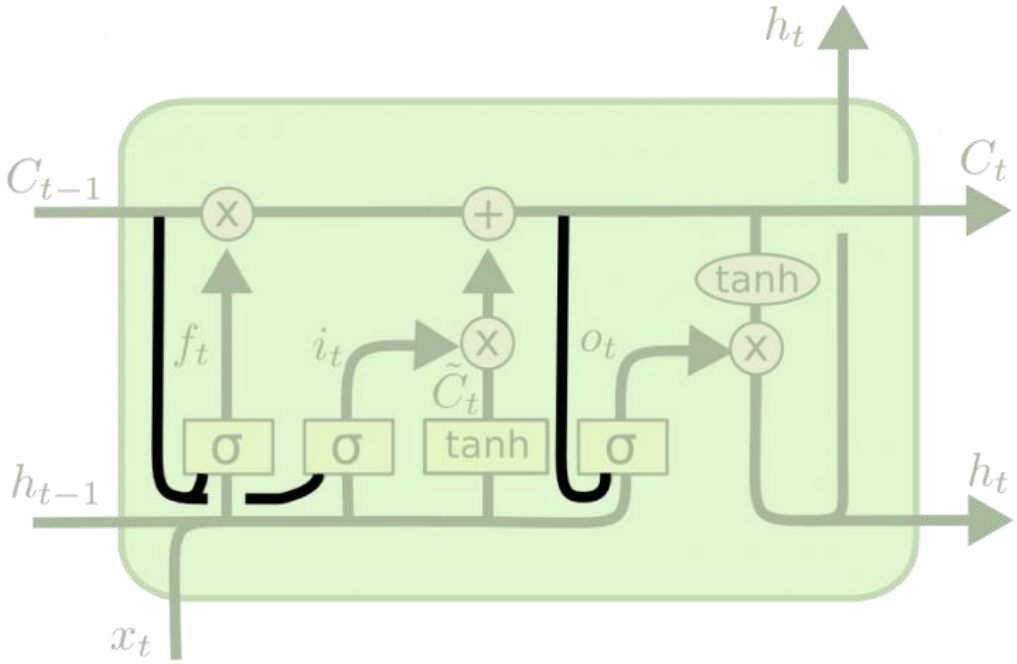
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Архитектура LSTM

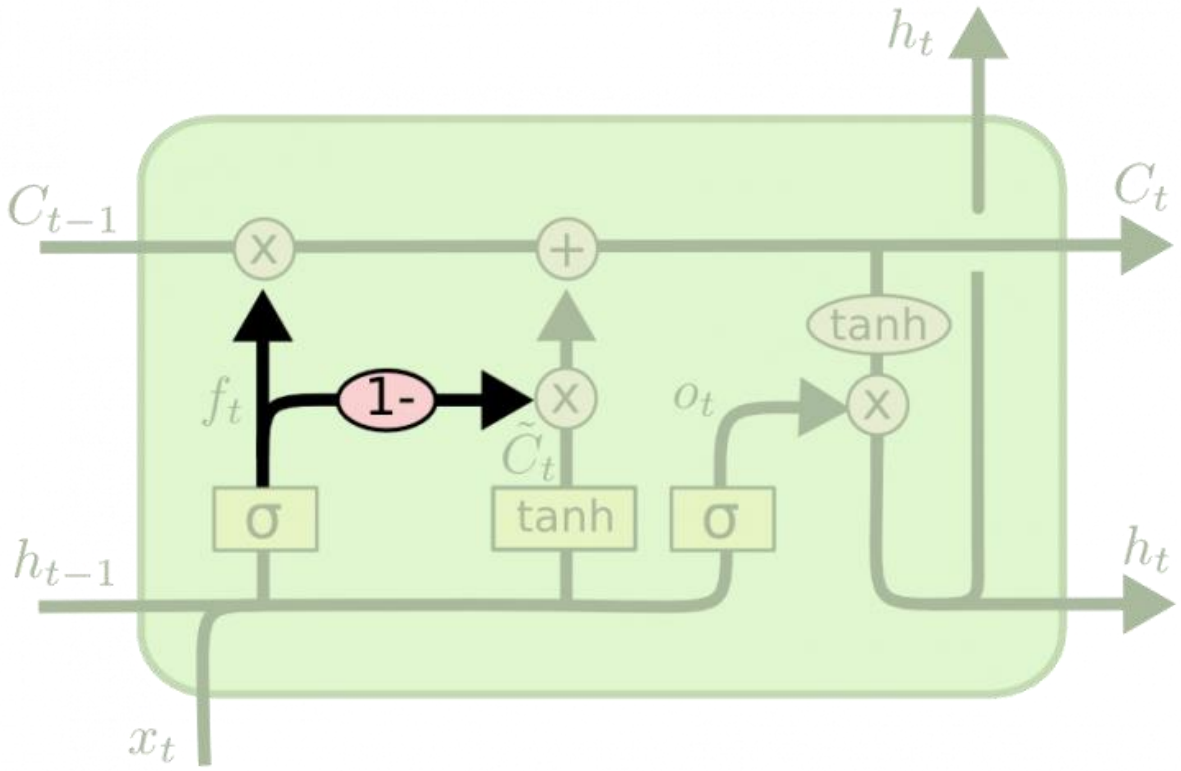
- Преимущества:
  - Есть явный механизм хранения и обновления памяти
  - Присутсвует карусель константной ошибки (“constant error carousel”), что решает проблему затухания градиентов
  - Архитектура хорошо себя зарекомендовала для сложных зависимостей
- Недостатки:
  - Архитектура эвристична (хотя что в DL не эвристично?)
  - Много весов (особенно с модификациями), долго обучать
  - Не решает проблему взрыва градиента (можно решить отдельно)
  - Есть исследования, что можно упростить архитектуру без снижения качества (Greff 2015, LSTM: A Search Space Odyssey)

Peepholes connections (aka skip-connections)



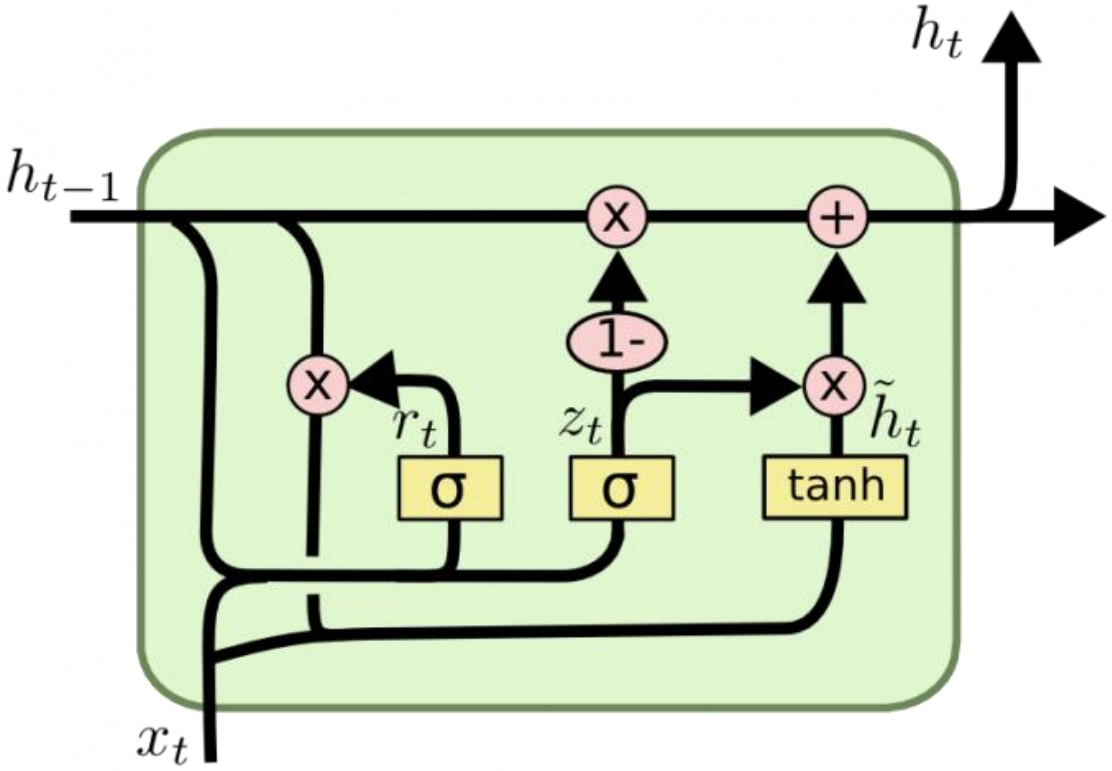
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Объединение гейтов



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

GRU



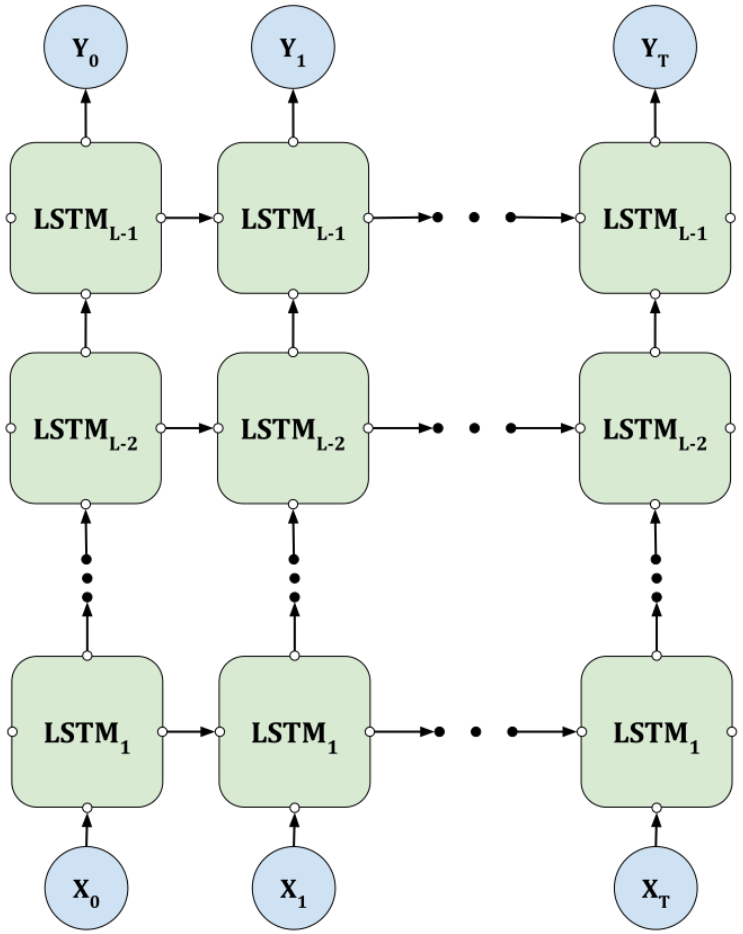
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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

Многослойные модели



Итоги

- RNN просты и верны идейно, но плохо сохраняют зависимости
- LSTM решают проблему памяти, но существенно сложнее
- LSTM это просто эвристика без чёткого обоснования
- Существует много модификаций LSTM, которые в целом не дают прироста качества
- LSTM можно аппроксимировать более простыми моделями (GRU)
- На базе рекуррентных сетей далее строится механизм внимания (Attention) и трансформеры

Image credit

- https://karpathy.github.io/2015/05/21/rnn-effectiveness/
- https://habr.com/ru/companies/wunderfund/articles/331310/
- https://logic.pdmi.ras.ru/~sergey/teaching/mlhse18/14-rnn.pdf
- http://vbystricky.ru/2021/05/rnn_lstm_gru_etc.html