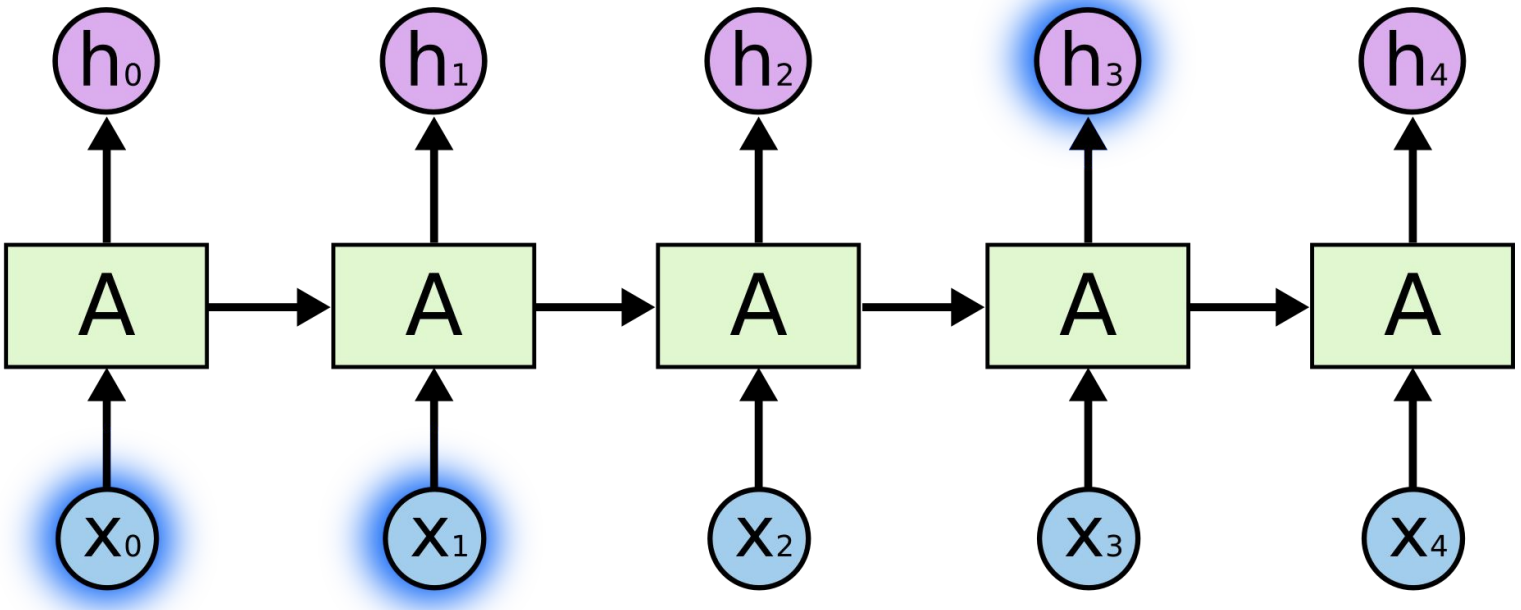
Рекуррентные нейронные сети. Часть 2.

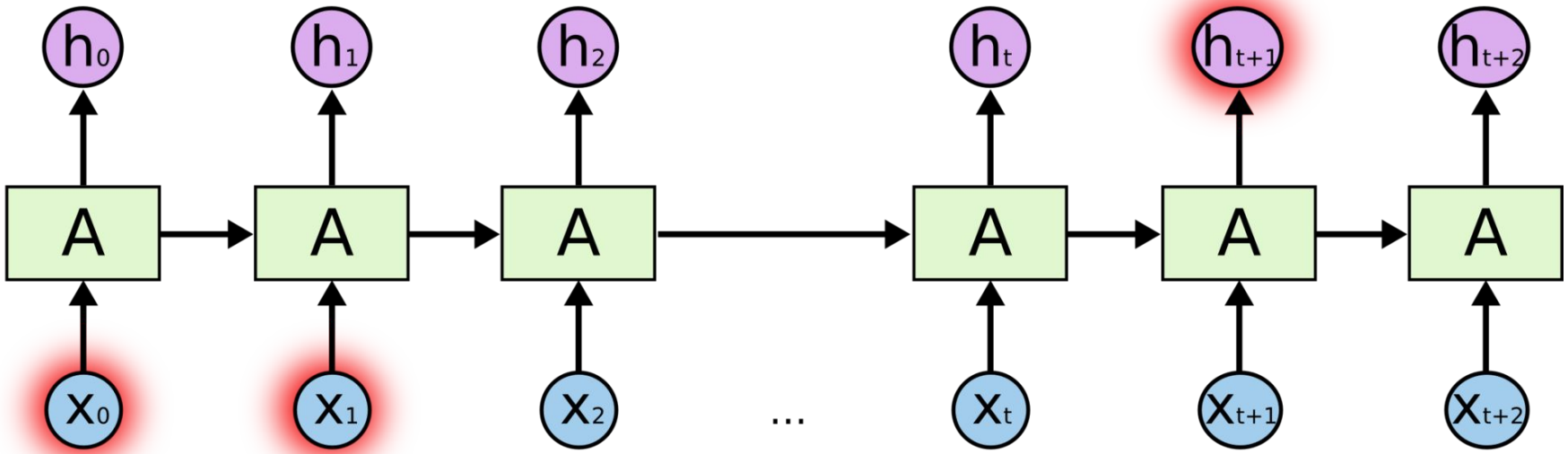
Павел Остяков
pavelosta@gmail.com

Проблема долговременных зависимостей

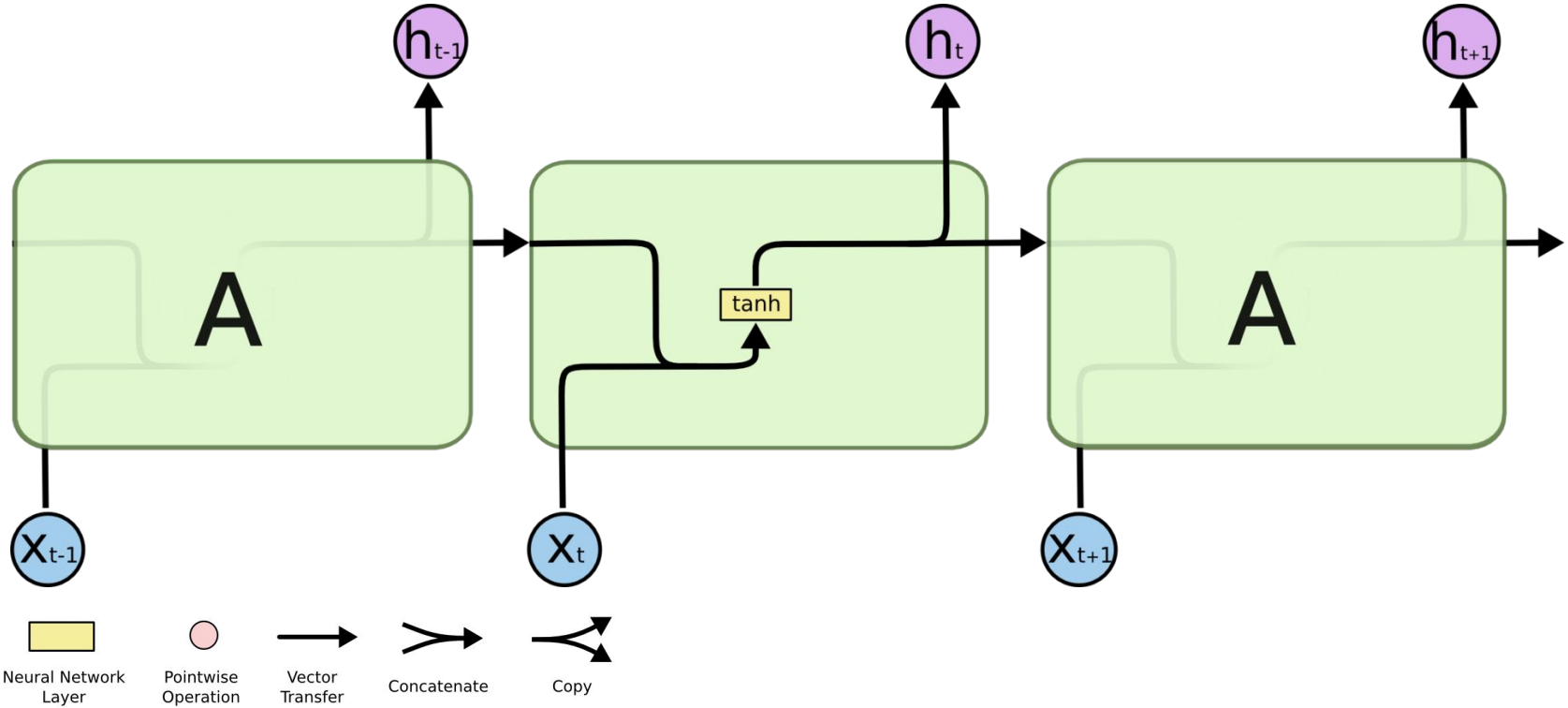


Изображения здесь и далее взяты из
http://colah.github.io/posts/2015-08-Understanding-LSTMs

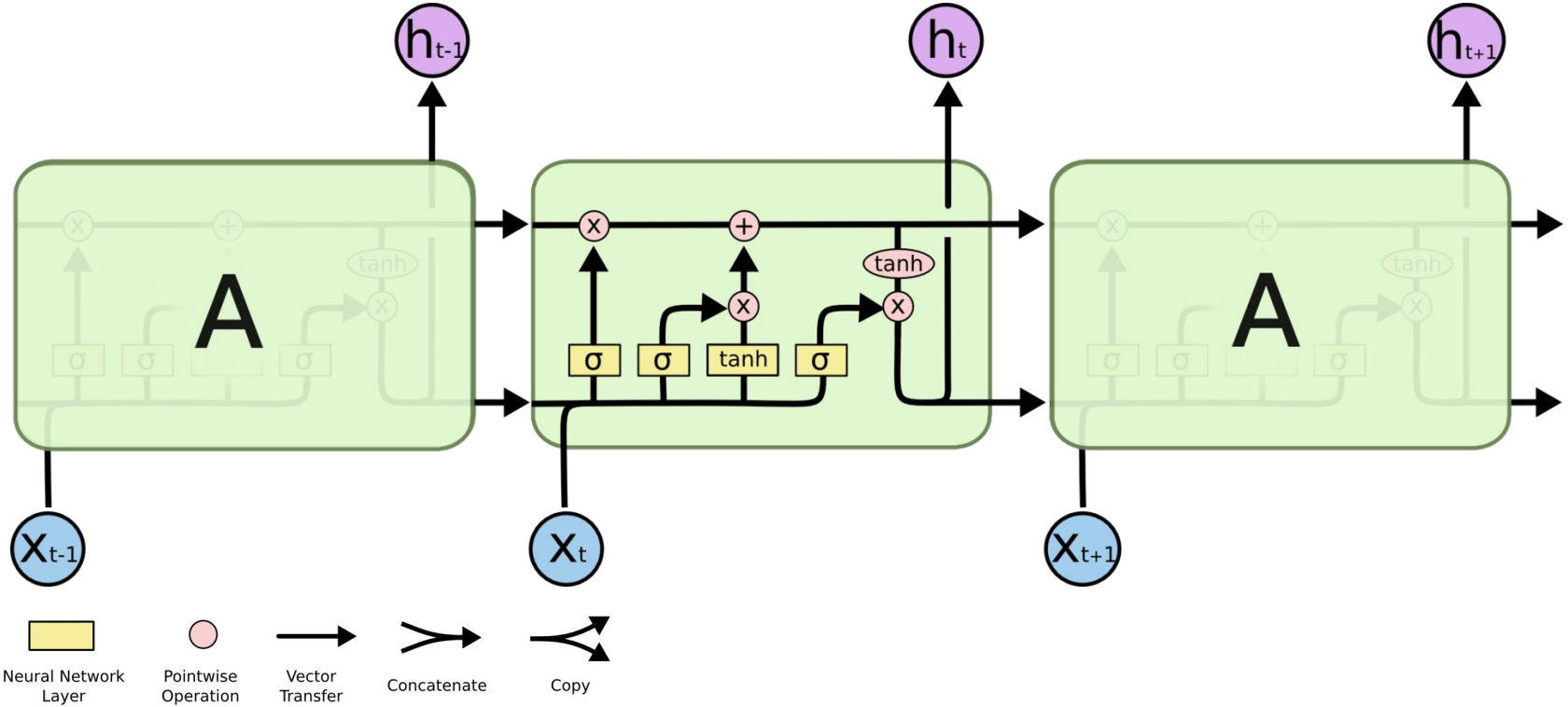
Проблема долговременных зависимостей



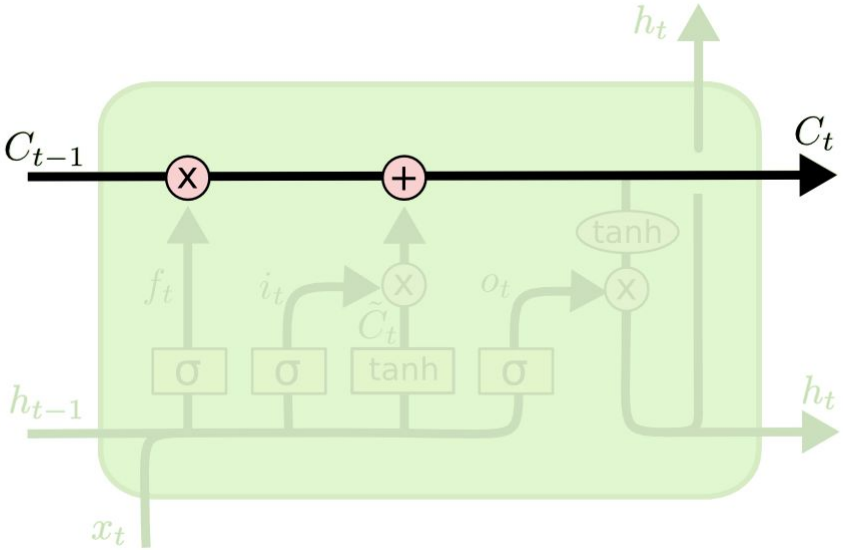
Стандартный модуль RNN

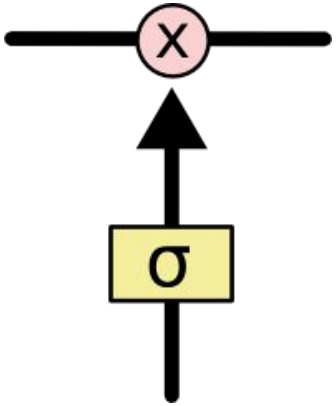


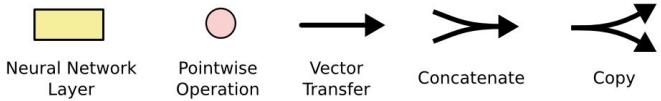
Модуль LSTM


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

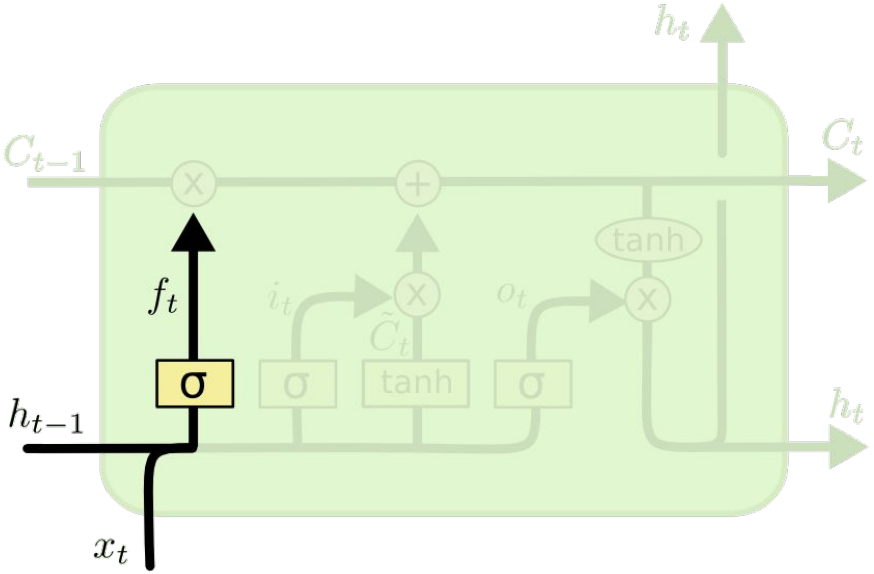
Модуль LSTM. Основная идея



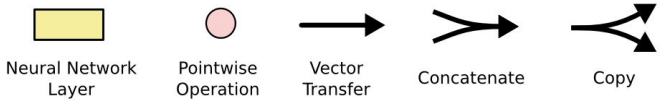




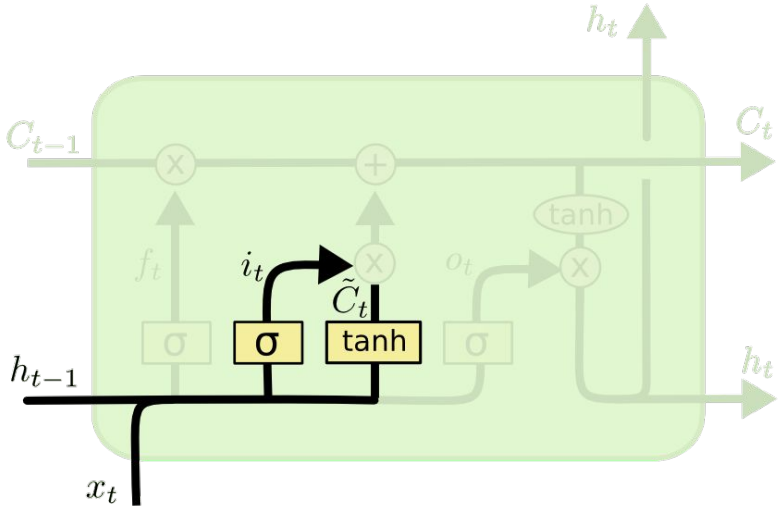
Модуль LSTM. “Слой забывания”



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

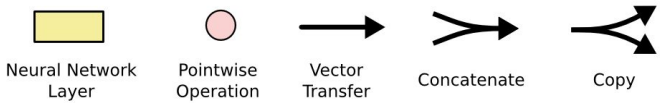


Модуль LSTM. Слой запоминания

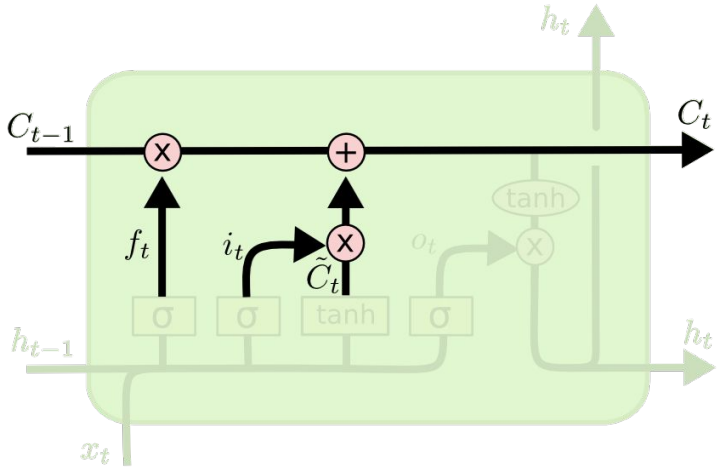


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

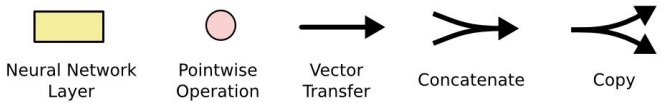
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



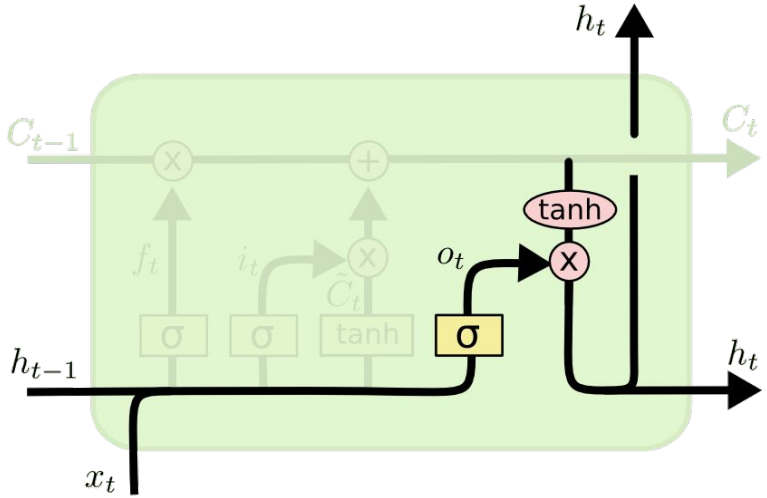
Модуль LSTM. Обновление скрытого состояния



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

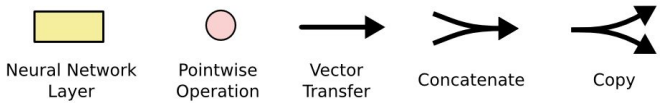


Модуль LSTM. Выход ячейки

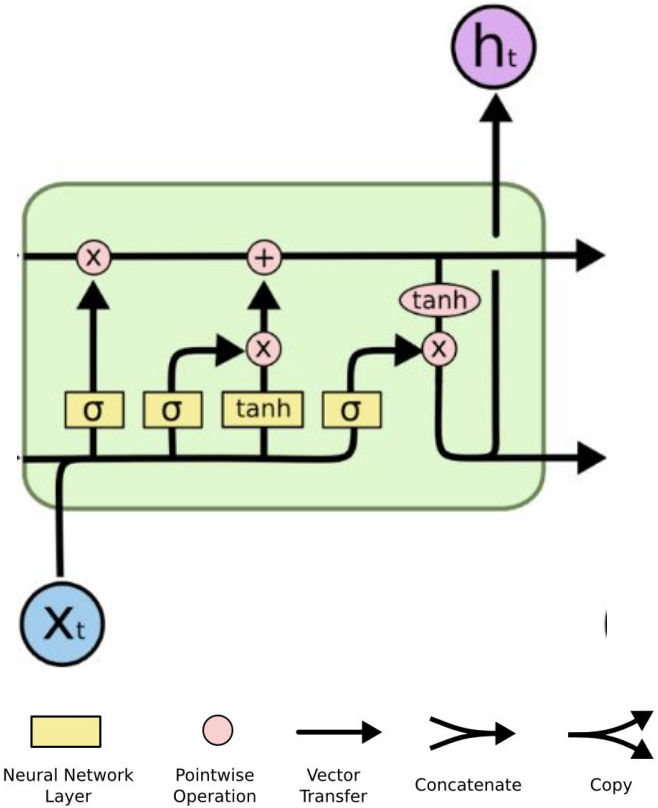


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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



Модуль LSTM



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

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

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

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

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

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

Проблемы LSTM

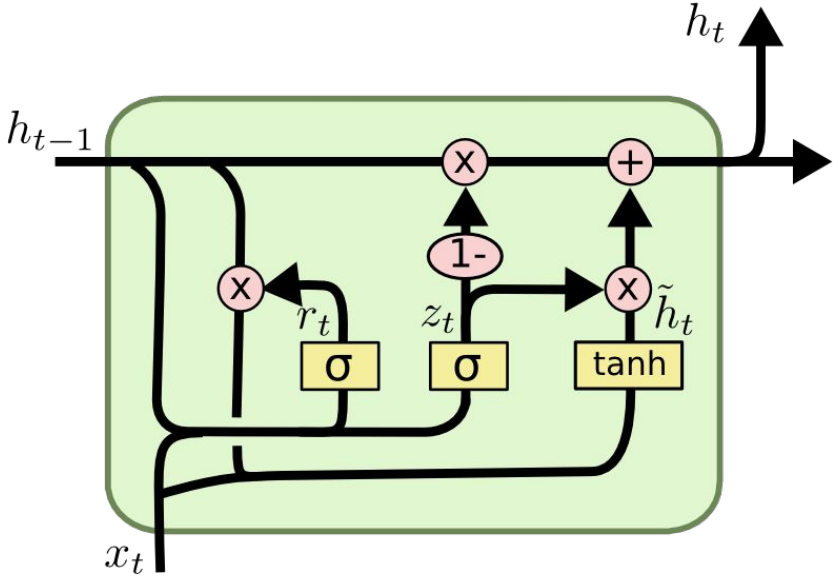
- Очень медленно обучаются

```
171/2083 [=>............................] - ETA: 391648s -
172/2083 [=>............................] - ETA: 390988s -
173/2083 [=>............................] - ETA: 390274s -
174/2083 [=>............................] - ETA: 389576s -
175/2083 [=>............................] - ETA: 389055s -
176/2083 [=>............................] - ETA: 388459s -
177/2083 [=>............................] - ETA: 387942s -
```

- Быстро переобучаются

```
ETA: 65s - loss: 0.4846 - acc: 0.8287
ETA: 43s - loss: 0.4845 - acc: 0.8288
ETA: 21s - loss: 0.4843 - acc: 0.8288
38439s - loss: 0.4842 - acc: 0.8289 - val_loss: 1.3238 - val_acc: 0.5104
```

Модуль GRU

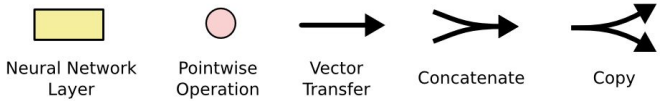


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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



Модуль GRU. Особенности

- Обучаются быстрее, чем LSTM
- Меньше переобучаются
- Зачастую помогают достичь такого же (или даже лучше) качества

Модуль SRU

$$\tilde{\mathbf{x}}_t = \mathbf{W}\mathbf{x}_t$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{b}_f)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \tilde{\mathbf{x}}_t$$
$$\mathbf{h}_t = g(\mathbf{c}_t)$$

Здесь g - некоторая функция активации

https://arxiv.org/pdf/1709.02755.pdf

Модуль SRU.

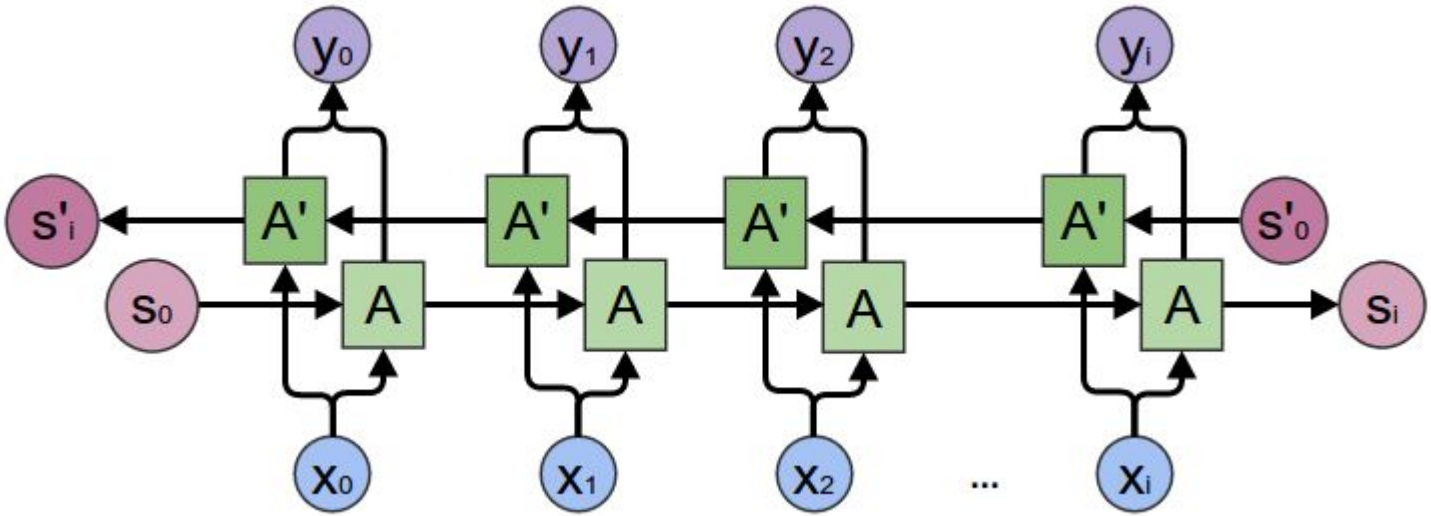
$$
\begin{aligned}
\tilde{\mathbf{x}}_t &= \mathbf{W}\mathbf{x}_t \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{b}_f) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{b}_r) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + (1 - \mathbf{f}_t) \odot \tilde{\mathbf{x}}_t \\
\mathbf{h}_t &= \mathbf{r}_t \odot g(\mathbf{c}_t) + (1 - \mathbf{r}_t) \odot \mathbf{x}_t
\end{aligned}
$$

SRU. Результаты

OpenNMT default setup	# layers	Size		Test BLEU	Time in RNNs
Klein et al. (2017)	2	-	-	17.60	
Klein et al. (2017) + BPE	2	-	-	19.34	
cuDNN LSTM (wd $= 0$)	2	85m	10m	18.04	149 min
cuDNN LSTM (wd $= 10^{-5}$)	2	85m	10m	19.99	149 min
Our setup					
cuDNN LSTM	2	84m	9m	19.67	46 min
cuDNN LSTM	3	88m	13m	19.85	69 min
cuDNN LSTM	5	96m	21m	20.45	115 min
SRU	3	81m	6m	18.89	12 min
SRU	5	84m	9m	19.77	20 min
SRU	6	85m	10m	20.17	24 min
SRU	10	91m	16m	20.70	40 min

Table 5: English-German translation results (Section 4.4). We list the total number of parameters and the number excluding word embeddings. Our setup disables $\mathbf{h}_{t-1}$ input, which significantly reduces the training time. Timings are performed on a single Nvidia Titan X Pascal GPU.

Bidirectional RNN



https://hardfish82.github.io/posts/2015-09-NN-Types-FP/

Дополнительные трюки

- Сортировка батчей по длине
- Recurrent dropout
- Batch Normalization
- Data augmentation
- Residual connections

Ссылки

- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://arxiv.org/abs/1412.3555
- https://arxiv.org/pdf/1709.02755.pdf
- https://arxiv.org/pdf/1507.06228.pdf
- https://hardfish82.github.io/posts/2015-09-NN-Types-FP/