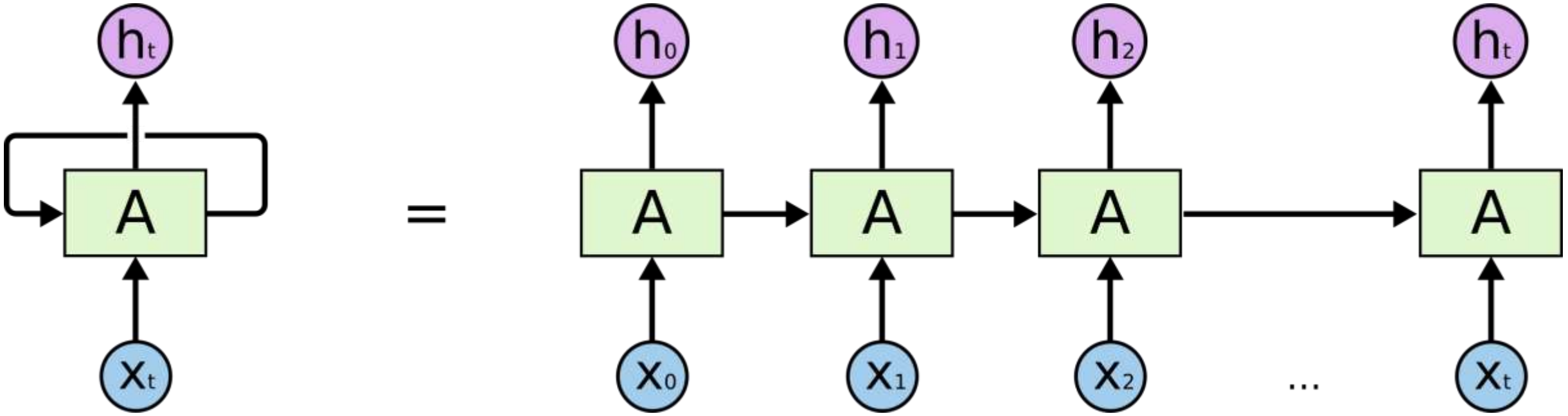
xLSTM

Extended
Long Short-Term
Memory

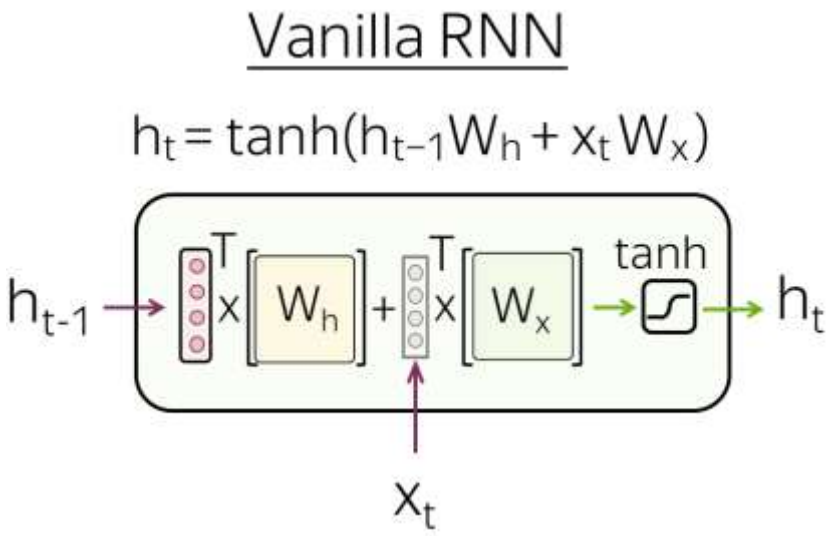
RNN

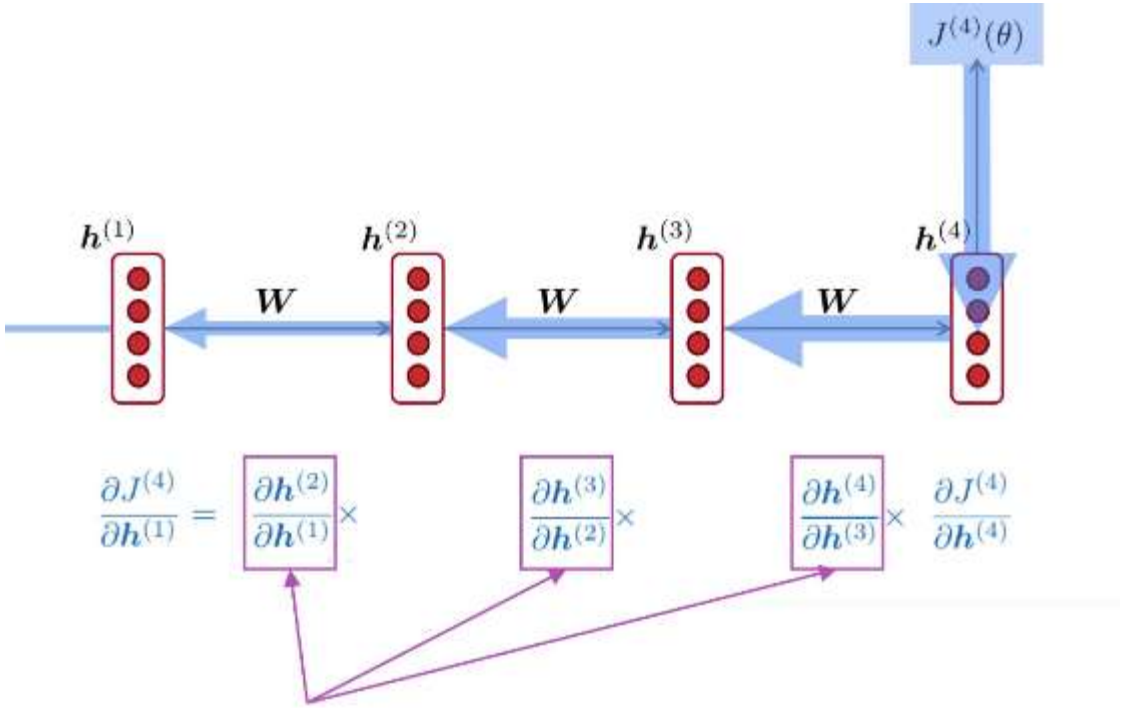


Проблемы

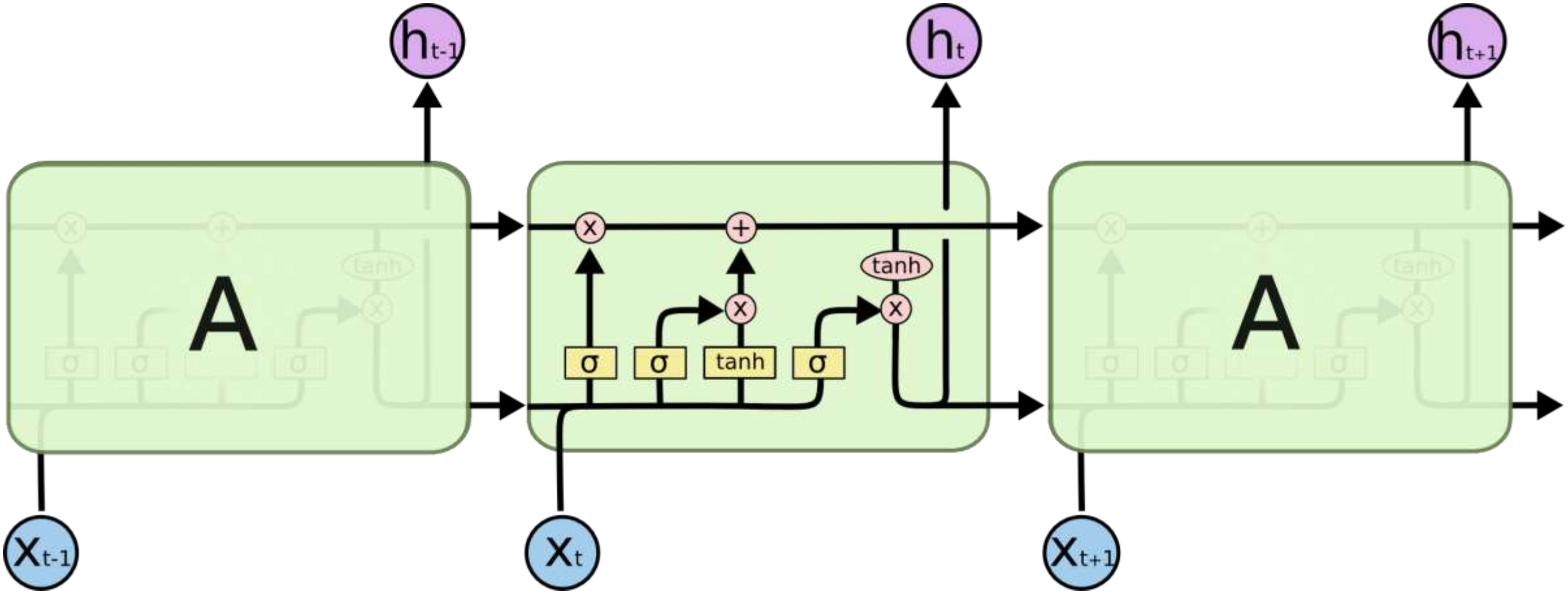
- Вектор контекста меняется на каждом шаге, поэтому не имеет возможности пройти на некотором этапе неизменным

- Затухание градиента

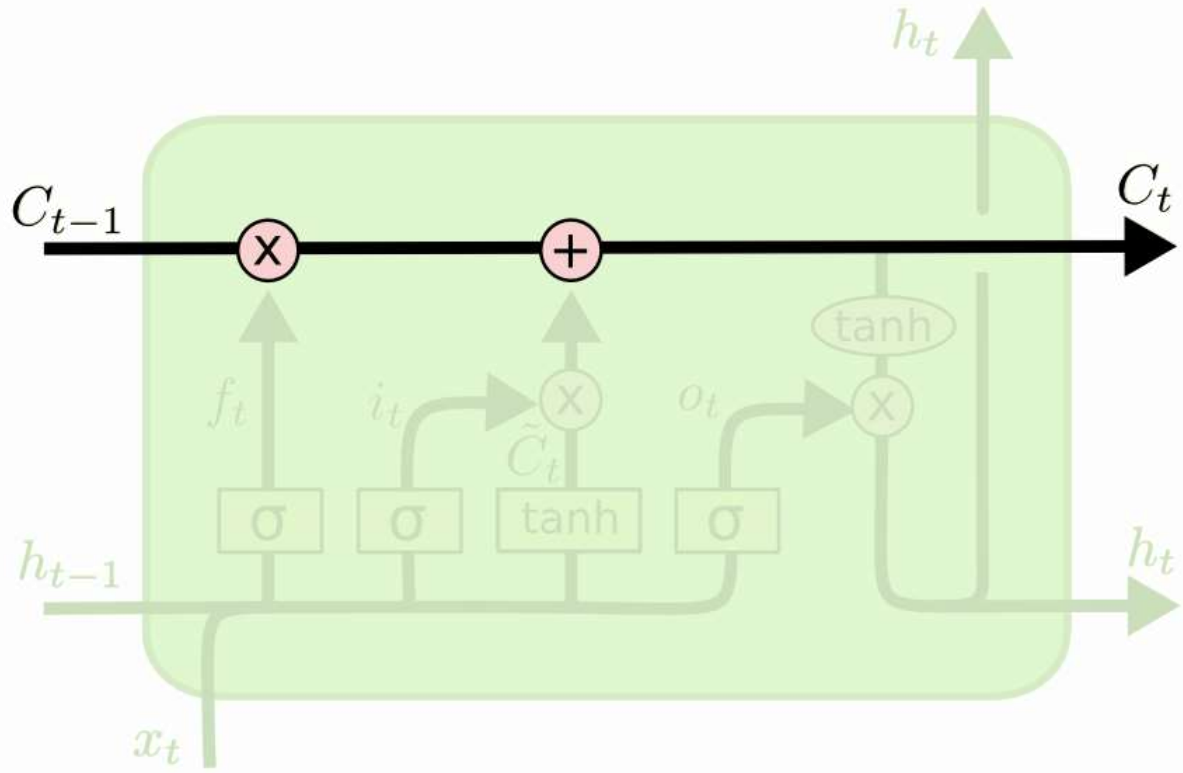




LSTM

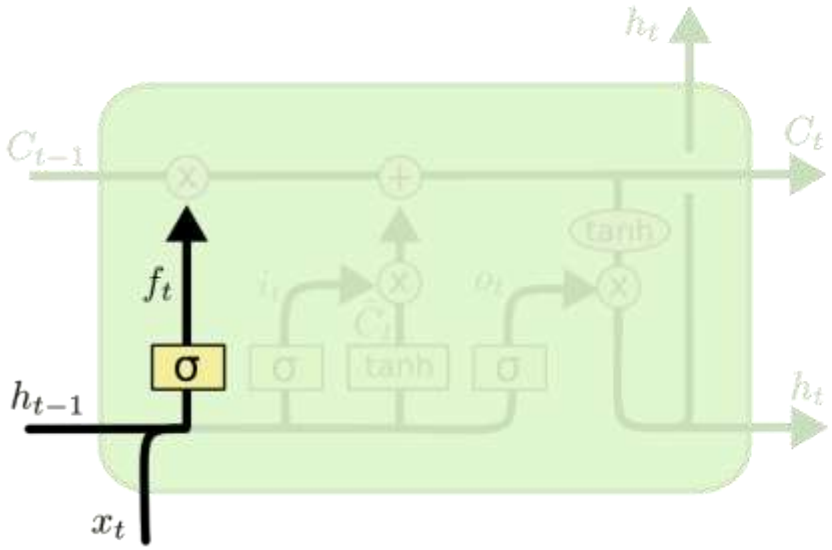


Ключевая идея



Ворота забывания

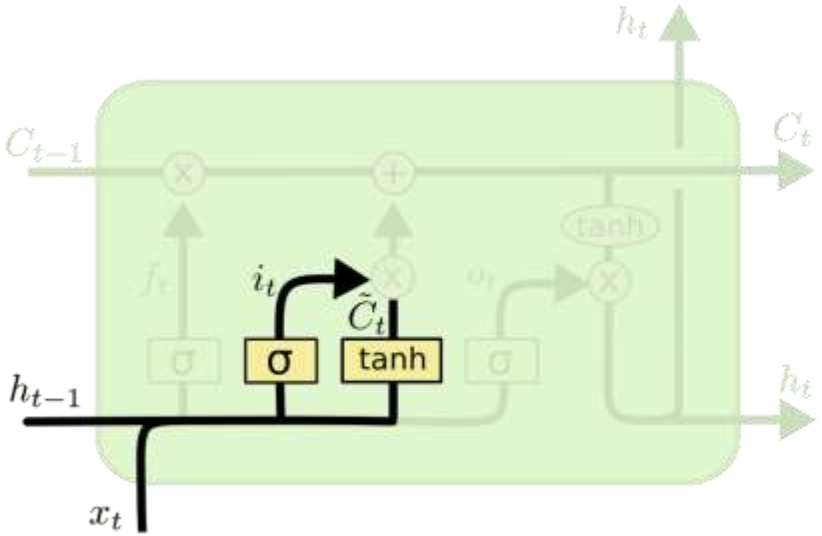
Нужно понять, что необходимо забыть, а что оставить в памяти. Эту подзадачу можно интерпретировать как бинарную классификацию. Берётся вектор краткосрочной памяти, добавляется новый ввод, всё это умножается на матрицу забывания. В результате будет получен вектор из чисел между 0 и 1. Далее путем поэлементного умножения на вектор долгосрочной памяти часть информации удалится (умножение на ноль), часть останется (умножение на 1).



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

Ворота входа

Нужно решить, что сохранить в памяти: имеется два потока (для первого используется бинарная классификация для регулирования того, какую часть второго потока добавить. После этого данные потоки перемножаются и добавляются в долгосрочную память.

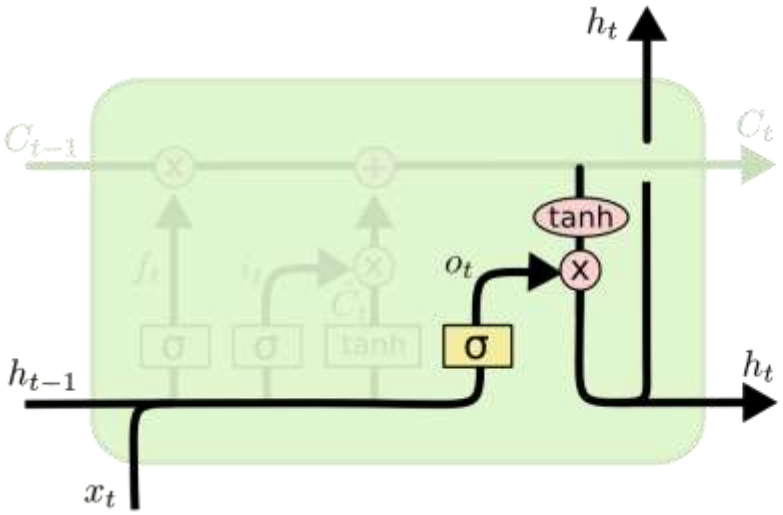


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Ворота выхода

Сколько информации из cell state следует отдавать на выход из LSTM-блока. В краткосрочную память добавляется новый вход, а затем переносится часть информации из долгосрочной памяти в краткосрочную.

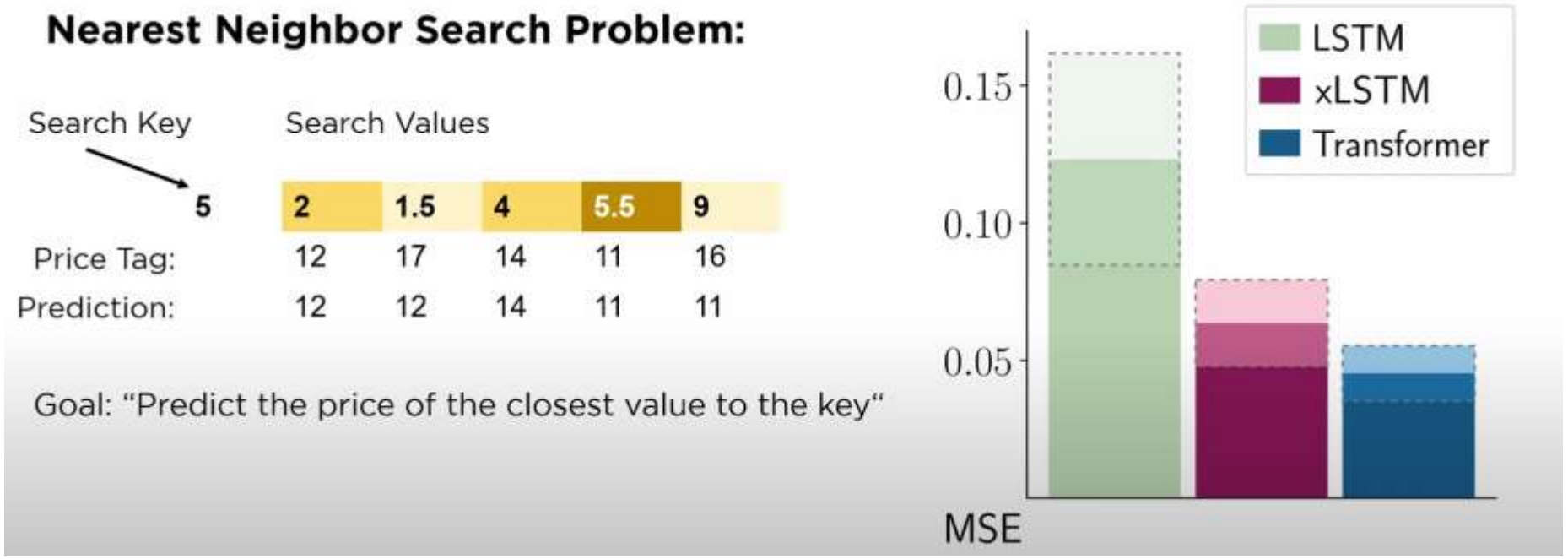


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Недостатки LSTM

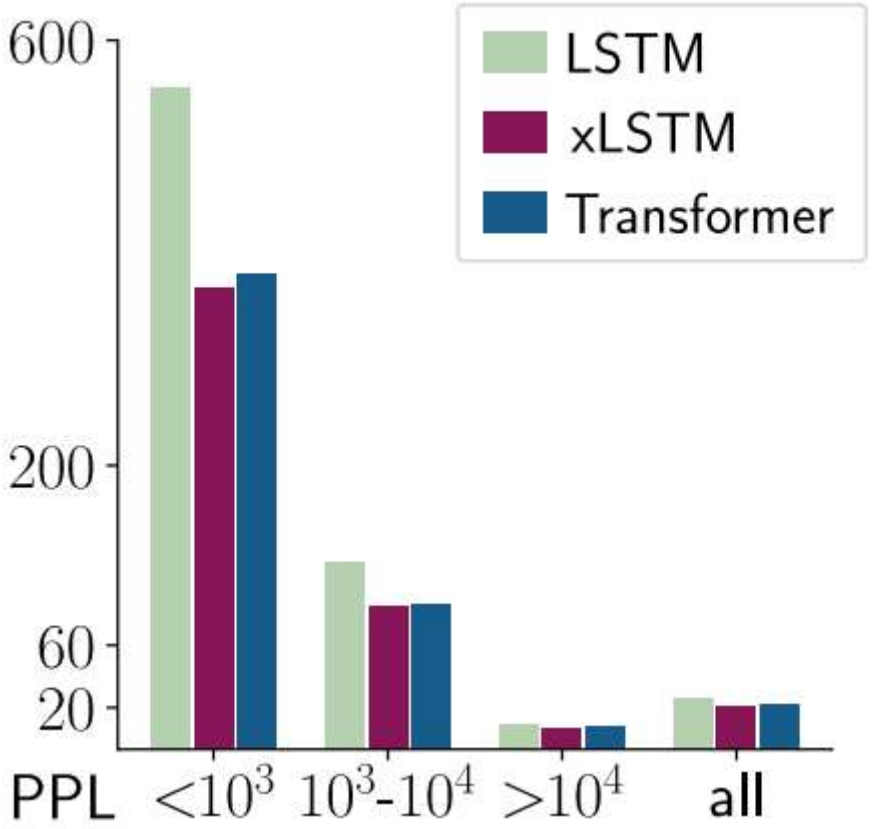
Неспособность пересмотреть решение о сохранении данных внутри ячейки памяти. Это демонстрируют на простой задаче поиска ближайшего соседа (Nearest Neighbor Search), где сначала даётся референсный вектор, а далее сканируется последовательность других векторов, и модель должна найти наиболее похожий вектор и вернуть связанное с ним значение, когда последовательность закончится. Когда в последовательности попадается ещё более подходящий вектор, то модель не справляется.



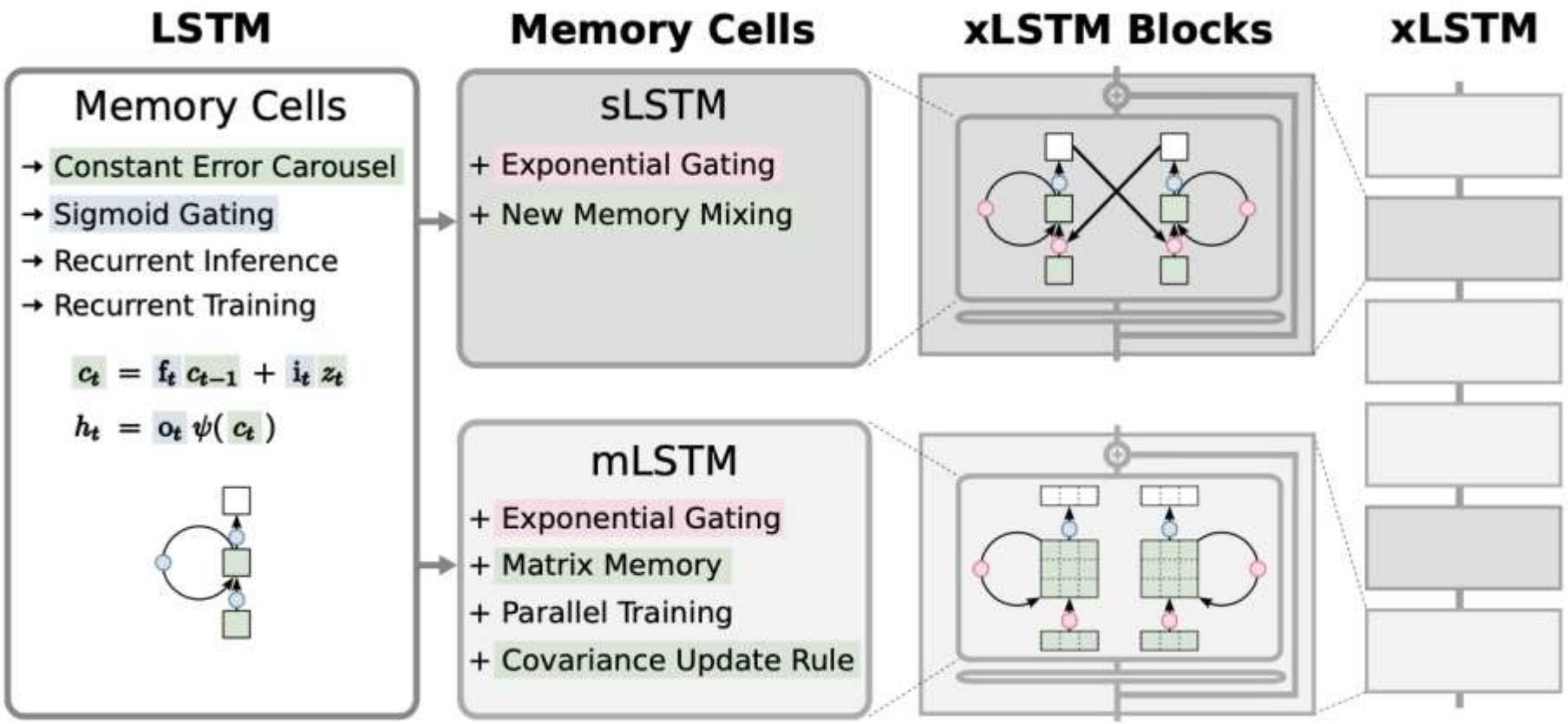
Недостатки LSTM

Ограниченная память — всё надо впихнуть внутрь скаляра, который хранится в ячейке памяти LSTM. Это демонстрируют на задаче предсказания редкого токена (Rare Token Prediction)

Плохая параллелизация в силу последовательной обработки скрытых состояний ячейки между соседними временными отсчётами (состояние зависит от предыдущего через hidden-hidden связи)



xLSTM


LSTM
Memory Cells
→ Constant Error Carousel
→ Sigmoid Gating
→ Recurrent Inference
→ Recurrent Training
$c_t = f_t\, c_{t-1} + i_t\, z_t$
$h_t = o_t\, \psi(c_t)$
Memory Cells
sLSTM
+ Exponential Gating
+ New Memory Mixing
mLSTM
+ Exponential Gating
+ Matrix Memory
+ Parallel Training
+ Covariance Update Rule
xLSTM Blocks
xLSTM

Решения в xLSTM

Exponential Gates: добавляются экспоненциальные функции активации на input и forget gate. Также появляется отдельное состояние нормализатора, и чтобы от экспоненты всё не взлетело в космос - состояние стабилизатора.

$$c_t = \mathrm{f}_t \; c_{t-1} + \mathrm{i}_t \; z_t \qquad \text{cell state} \qquad (8)$$

$$n_t = \mathrm{f}_t \; n_{t-1} + \mathrm{i}_t \qquad \text{normalizer state} \qquad (9)$$

$$h_t = \mathrm{o}_t \; \tilde{h}_t \,, \qquad \tilde{h}_t = c_t \,/\, n_t \qquad \text{hidden state} \qquad (10)$$

$$z_t = \varphi\left(\tilde{z}_t\right) \,, \qquad \tilde{z}_t = \boldsymbol{w}_z^\top \boldsymbol{x}_t + r_z h_{t-1} + b_z \qquad \text{cell input} \qquad (11)$$

$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_\mathrm{i}^\top \boldsymbol{x}_t + r_\mathrm{i} \, h_{t-1} + b_\mathrm{i} \qquad \text{input gate} \qquad (12)$$

$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right) \,, \qquad \tilde{\mathrm{f}}_t = \boldsymbol{w}_\mathrm{f}^\top \boldsymbol{x}_t + r_\mathrm{f} \, h_{t-1} + b_\mathrm{f} \qquad \text{forget gate} \qquad (13)$$

$$\mathrm{o}_t = \sigma\left(\tilde{\mathrm{o}}_t\right) \,, \qquad \tilde{\mathrm{o}}_t = \boldsymbol{w}_\mathrm{o}^\top \boldsymbol{x}_t + r_\mathrm{o} \, h_{t-1} + b_\mathrm{o} \qquad \text{output gate} \qquad (14)$$

$$m_t = \max\left(\log(\mathrm{f}_t) + m_{t-1}, \log(\mathrm{i}_t)\right) \qquad \text{stabilizer state} \qquad (15)$$

$$\mathrm{i}'_t = \exp\left(\log\left(\mathrm{i}_t\right) - m_t\right) = \exp\left(\tilde{\mathrm{i}}_t - m_t\right) \qquad \text{stabil. input gate} \qquad (16)$$

$$\mathrm{f}'_t = \exp\left(\log\left(\mathrm{f}_t\right) + m_{t-1} - m_t\right) \qquad \text{stabil. forget gate} \qquad (17)$$

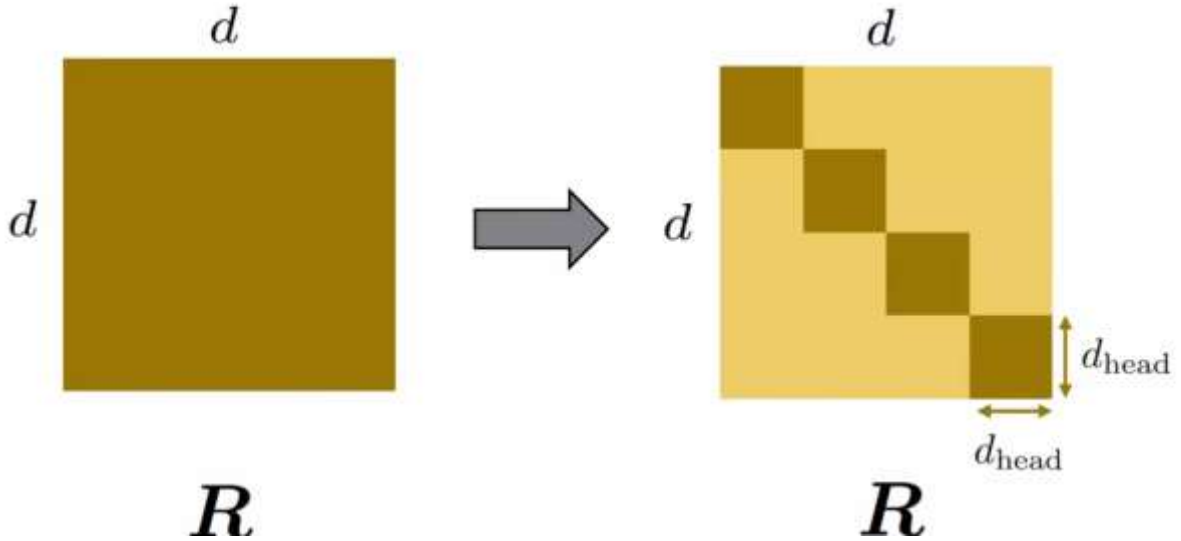
Решения в xLSTM

Scalar Pre-activations:

$$
\begin{aligned}
\tilde{z}_t &= \boldsymbol{w}_z^\top \boldsymbol{x}_t + r_z h_{t-1} + b_z \\
\tilde{\mathrm{i}}_t &= \boldsymbol{w}_\mathrm{i}^\top \boldsymbol{x}_t + r_\mathrm{i}\, h_{t-1} + b_\mathrm{i} \\
\tilde{\mathrm{f}}_t &= \boldsymbol{w}_\mathrm{f}^\top \boldsymbol{x}_t + r_\mathrm{f}\, h_{t-1} + b_\mathrm{f} \\
\tilde{\mathrm{o}}_t &= \boldsymbol{w}_\mathrm{o}^\top \boldsymbol{x}_t + r_\mathrm{o}\, h_{t-1} + b_\mathrm{o}
\end{aligned}
$$

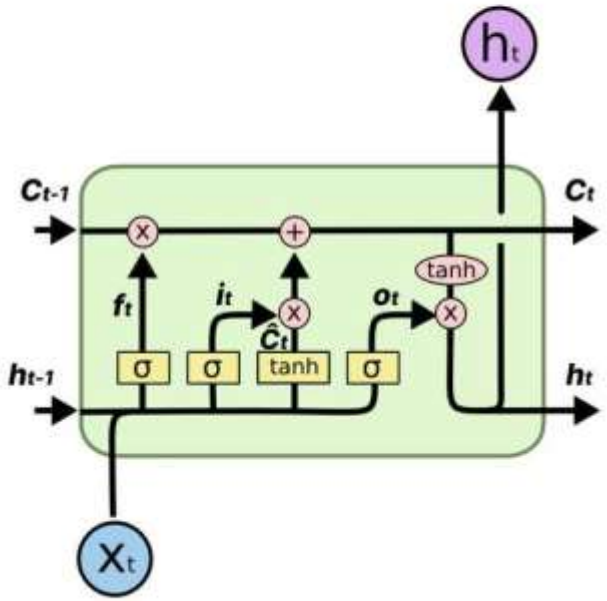
Vector Pre-activations:

$$
\begin{aligned}
\tilde{\boldsymbol{z}}_t &= \boldsymbol{W}_\boldsymbol{z}\, \boldsymbol{x}_t + \boldsymbol{R}_\boldsymbol{z}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_\boldsymbol{z} \\
\tilde{\mathbf{i}}_t &= \boldsymbol{W}_\mathbf{i}\, \boldsymbol{x}_t + \boldsymbol{R}_\mathbf{i}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_\mathbf{i} \\
\tilde{\mathbf{f}}_t &= \boldsymbol{W}_\mathbf{f}\, \boldsymbol{x}_t + \boldsymbol{R}_\mathbf{f}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_\mathbf{f} \\
\tilde{\mathbf{o}}_t &= \boldsymbol{W}_\mathbf{o}\, \boldsymbol{x}_t + \boldsymbol{R}_\mathbf{o}\, \boldsymbol{h}_{t-1} + \boldsymbol{b}_\mathbf{o}
\end{aligned}
$$

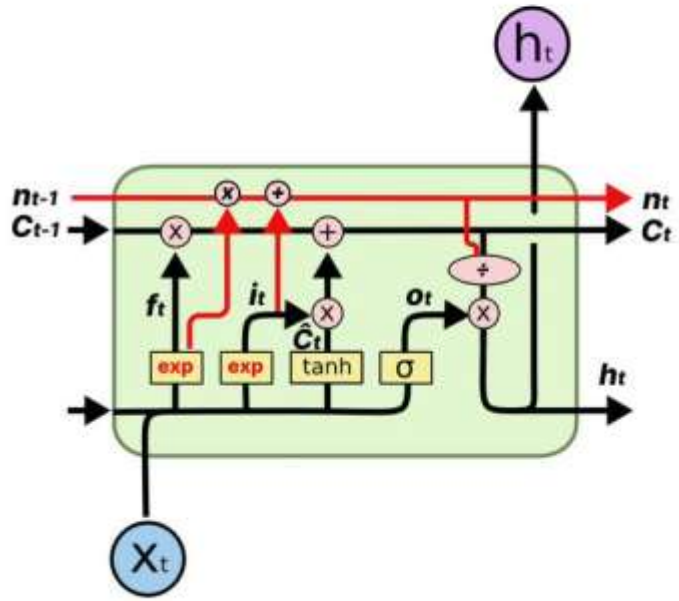


Решения в xLSTM

Блок LSTM



Блок sLSTM



Решения в xLSTM

Matrix memory: не что иное, как ассоциативная память; аналогично терминологии трансформера, создают запросы, ключи и значения qkv и получают hidden state в качестве значений v наиболее близких к запросу q. В формулу обновления ассоциативной памяти так же включены input и forget gates чтобы сделать ее более “lstm-like” это все параллелизуется

$$\boldsymbol{C}_t = \mathrm{f}_t \, \boldsymbol{C}_{t-1} + \mathrm{i}_t \, \boldsymbol{v}_t \, \boldsymbol{k}_t^\top \qquad \text{cell state} \quad (19)$$

$$\boldsymbol{n}_t = \mathrm{f}_t \, \boldsymbol{n}_{t-1} + \mathrm{i}_t \, \boldsymbol{k}_t \qquad \text{normalizer state} \quad (20)$$

$$\boldsymbol{h}_t = \mathrm{o}_t \odot \tilde{\boldsymbol{h}}_t \,, \qquad \tilde{\boldsymbol{h}}_t = \boldsymbol{C}_t \, \boldsymbol{q}_t \; / \; \max\left\{ \left| \boldsymbol{n}_t^\top \, \boldsymbol{q}_t \right|, 1 \right\} \qquad \text{hidden state} \quad (21)$$

$$\boldsymbol{q}_t = \boldsymbol{W}_q \, \boldsymbol{x}_t + \boldsymbol{b}_q \qquad \text{query input} \quad (22)$$

$$\boldsymbol{k}_t = \frac{1}{\sqrt{d}} \boldsymbol{W}_k \, \boldsymbol{x}_t + \boldsymbol{b}_k \qquad \text{key input} \quad (23)$$

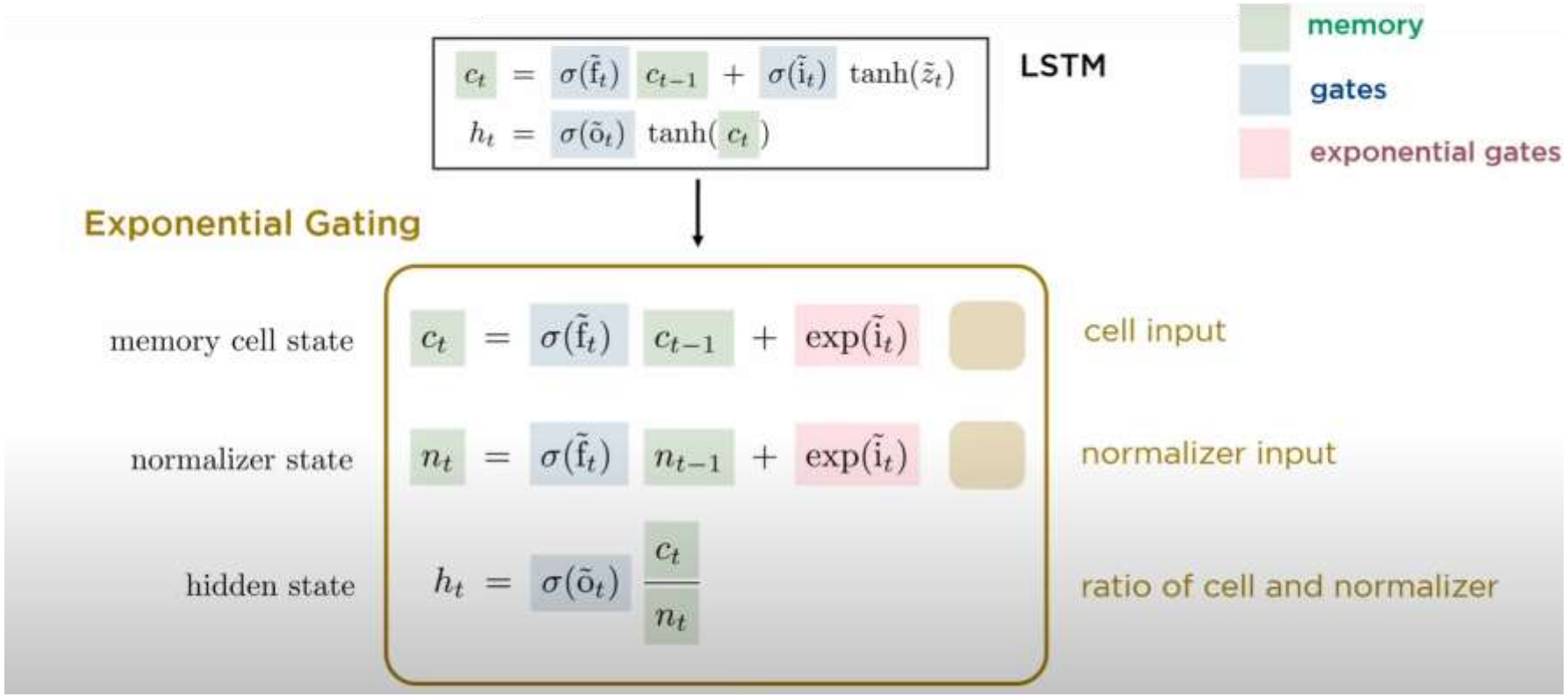
$$\boldsymbol{v}_t = \boldsymbol{W}_v \, \boldsymbol{x}_t + \boldsymbol{b}_v \qquad \text{value input} \quad (24)$$

$$\mathrm{i}_t = \exp\left(\tilde{\mathrm{i}}_t\right) \,, \qquad \tilde{\mathrm{i}}_t = \boldsymbol{w}_\mathrm{i}^\top \, \boldsymbol{x}_t + b_\mathrm{i} \qquad \text{input gate} \quad (25)$$

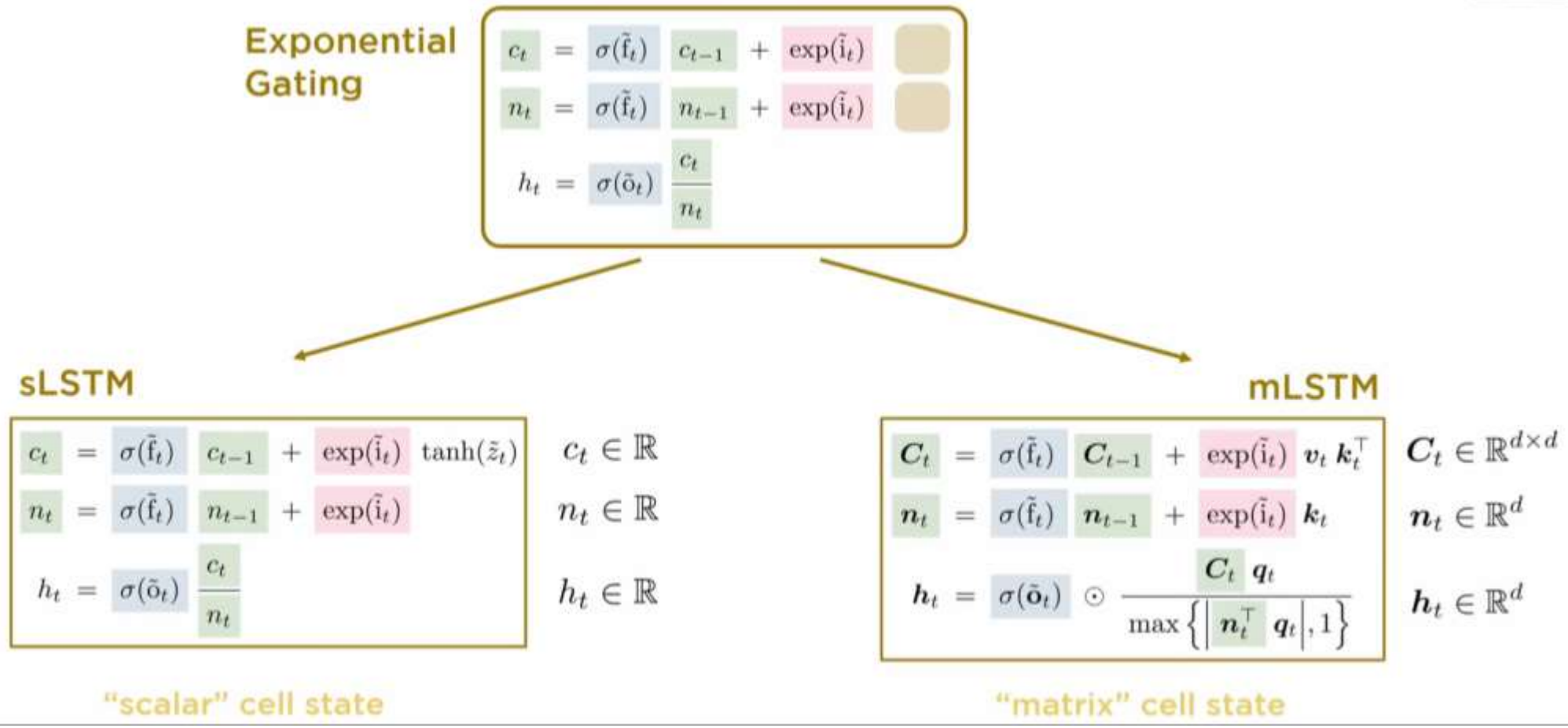
$$\mathrm{f}_t = \sigma\left(\tilde{\mathrm{f}}_t\right) \text{ OR } \exp\left(\tilde{\mathrm{f}}_t\right), \quad \tilde{\mathrm{f}}_t = \boldsymbol{w}_\mathrm{f}^\top \, \boldsymbol{x}_t + b_\mathrm{f} \qquad \text{forget gate} \quad (26)$$

$$\mathrm{o}_t = \sigma\left(\tilde{\mathbf{o}}_t\right) \,, \qquad \tilde{\mathbf{o}}_t = \boldsymbol{W}_\mathbf{o} \, \boldsymbol{x}_t + \boldsymbol{b}_\mathbf{o} \qquad \text{output gate} \quad (27)$$

Решения в xLSTM

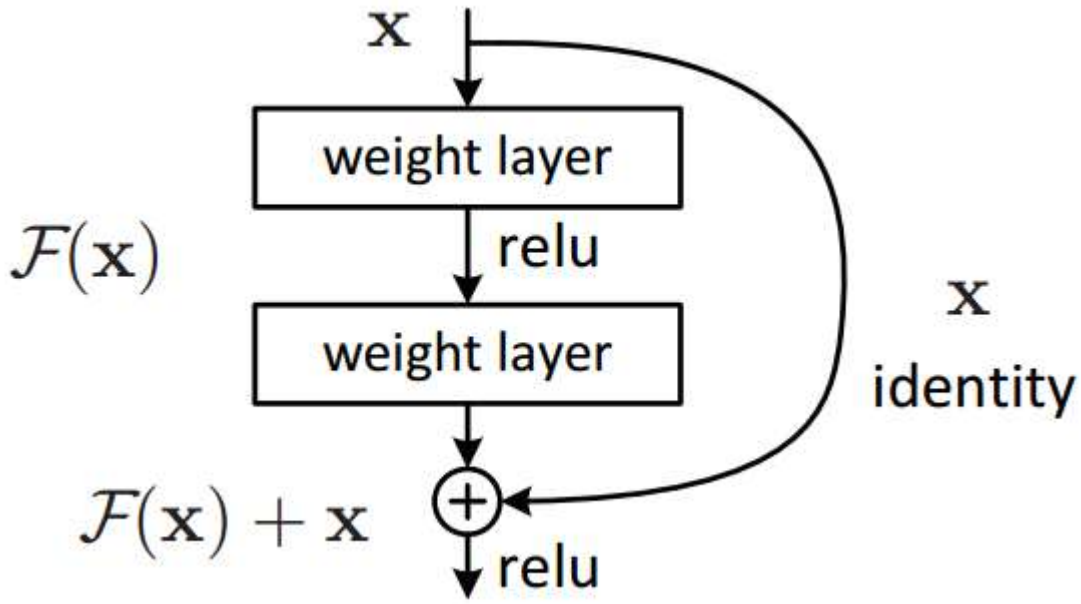


Решения в xLSTM

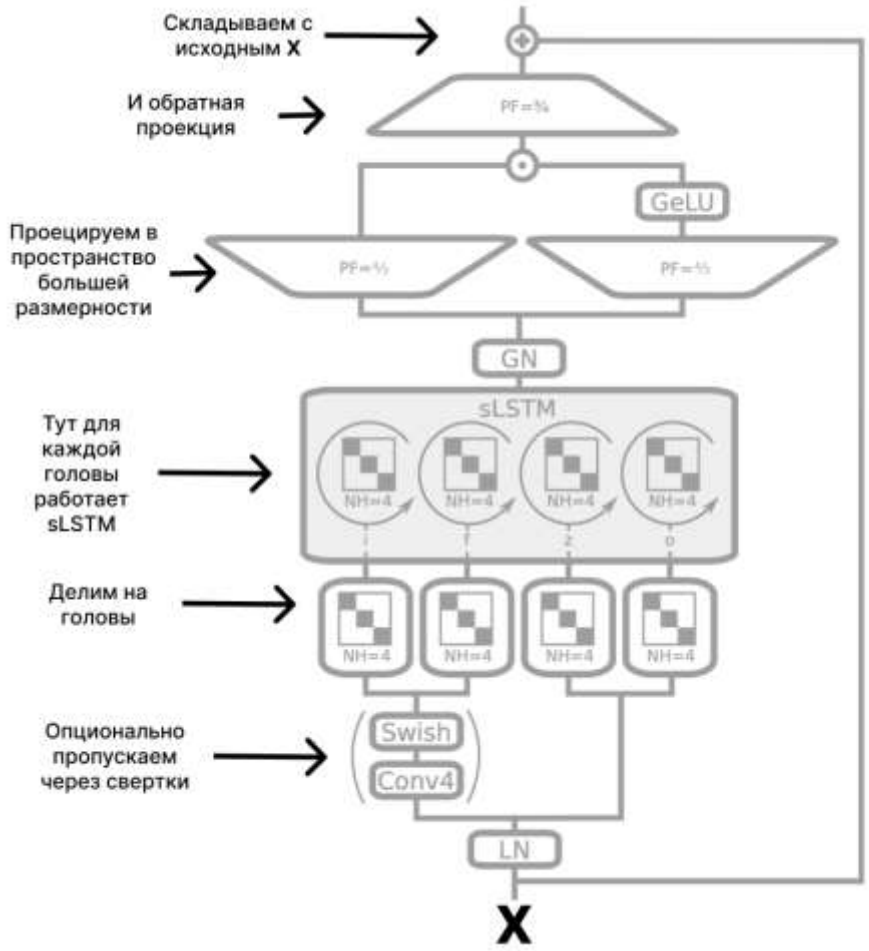


Как все собрать в xLSTM?

Дополнительно обернули каждую из структур в residual блоки

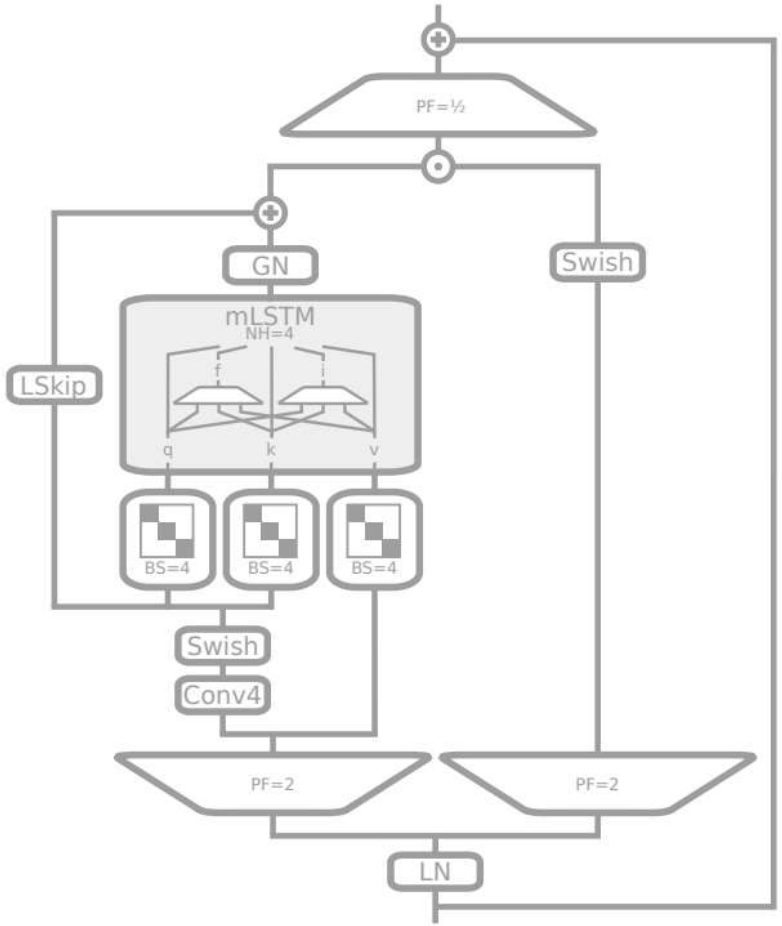


Как все собрать в xLSTM?



Входы **X** делятся на головы – на рисунке ниже их четыре. Перед этим входы опционально могут быть пропущены через несколько сверточных слоев. После деления в каждой из голов отрабатывает sLSTM. Выходы с голов затем объединяются с помощью GroupNorm, а потом проецируются в пространство большей размерности и обратно. Эта последняя часть названа **up-projection** и проделывается для того, что повысить качество histories seperating. Это умение сети разделять "линии повествования": например, понимать, к какому из персонажей относится то или иное действие.

Как все собрать в xLSTM?



Все то же самое, что и в sLSTM, но в другом порядке. Отражаем входы в пространство большей размерности → делим на головы → пропускаем через mLSTM → объединяем по GroupNorm → проецируем обратно в родную размерность → складываем со входами, чтобы получится residual → готово!

Результаты

Ablation studies on the new xLSTM components.

Model	Modification	Exponential Gating	Matrix Memory	#Params M	SlimPajama (15B) ppl ↓
LSTM	Vanilla Multi-Layer LSTM	✗	✗	607.8	2417.86
	Adding Resnet Backbone	✗	✗	506.1	35.46
	Adding Up-Projection Backbone	✗	✗	505.9	26.01
xLSTM[0:1]	Adding Exponential Gating	✓	✗	427.3	17.70
xLSTM[7:1]	Adding Matrix Memory	✓	✓	408.4	**13.48**

Результаты

Model	#Params M	SlimPajama (15B) ppl ↓
GPT-3	356	14.26
Llama	407	14.25
H3	420	18.23
Mamba	423	13.70
Hyena	435	17.59
RWKV-4	430	15.62
RWKV-5	456	16.53
RWKV-6	442	17.40
RetNet	431	16.23
HGRN	411	21.83
GLA	412	19.56
HGRN2	411	16.77
xLSTM[1:0]	409	**13.43**
xLSTM[7:1]	408	13.48