

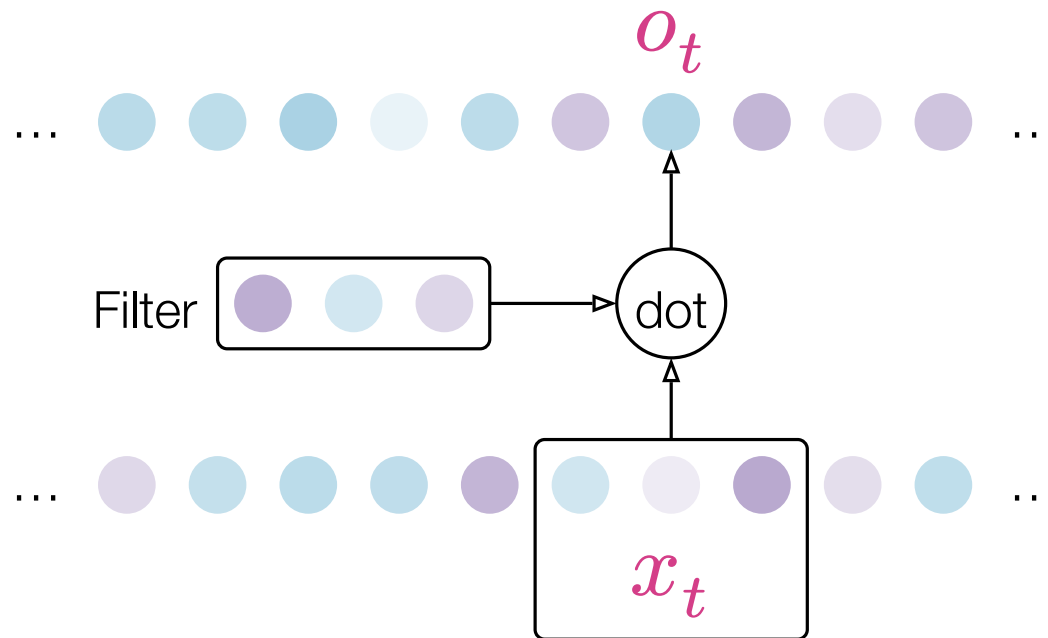
Deep Learning for Time Series

Session 2: ConvNets and Recurrent architectures

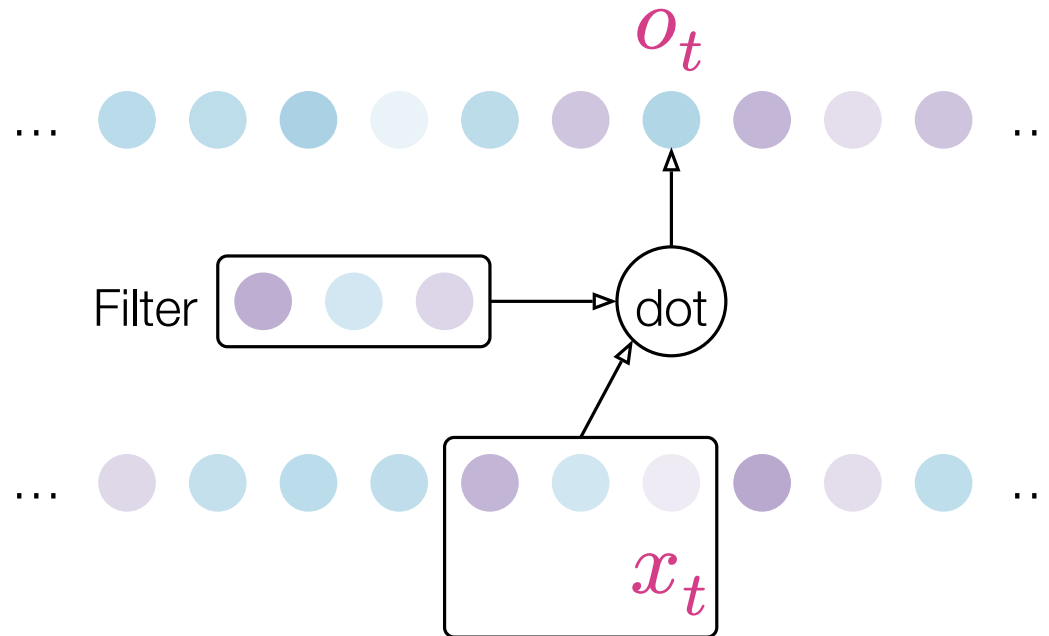
Romain Tavenard

Convolutional architectures

- Basic time series processing: 1d convolutions (over time)
- Limited receptive field: co-localization matters



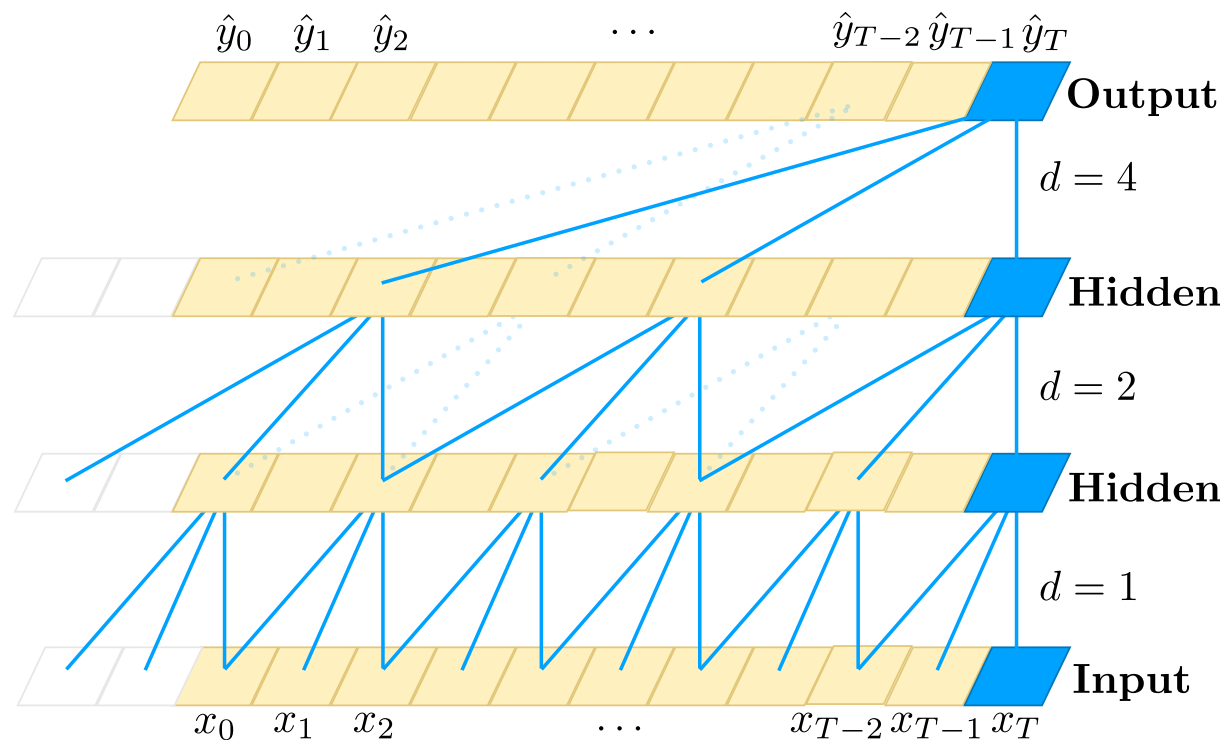
- Forecasting tasks: cannot access the future
- Causal convolution: convolve on past information alone (asymmetric window)



Temporal Convolution Network (TCN)

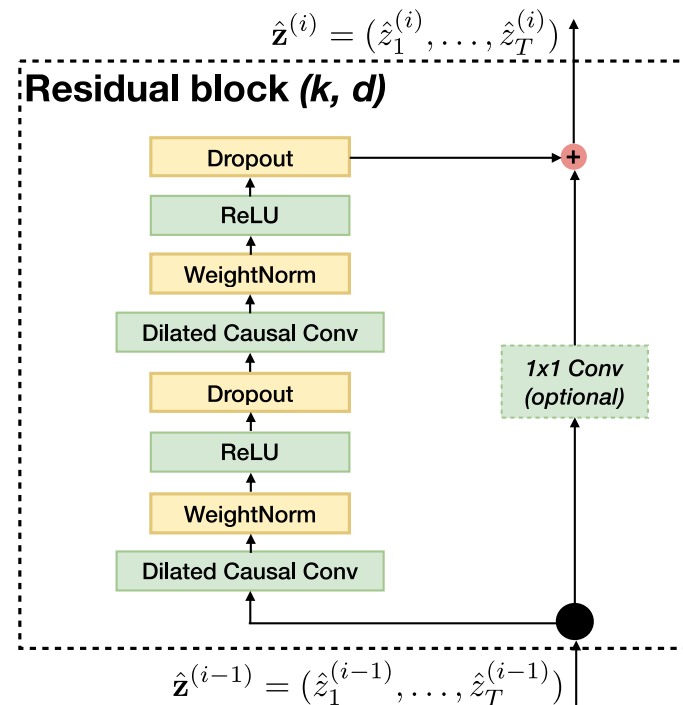
Convolutional architectures

- Main idea: cascade dilated causal convolutions
⇒ Larger receptive field



Source: “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”, Bai et al., arXiv 2018

- Additional improvements:
 - Residual connections
⇒ Multi-resolution analysis
 - Normalization+Dropout layers

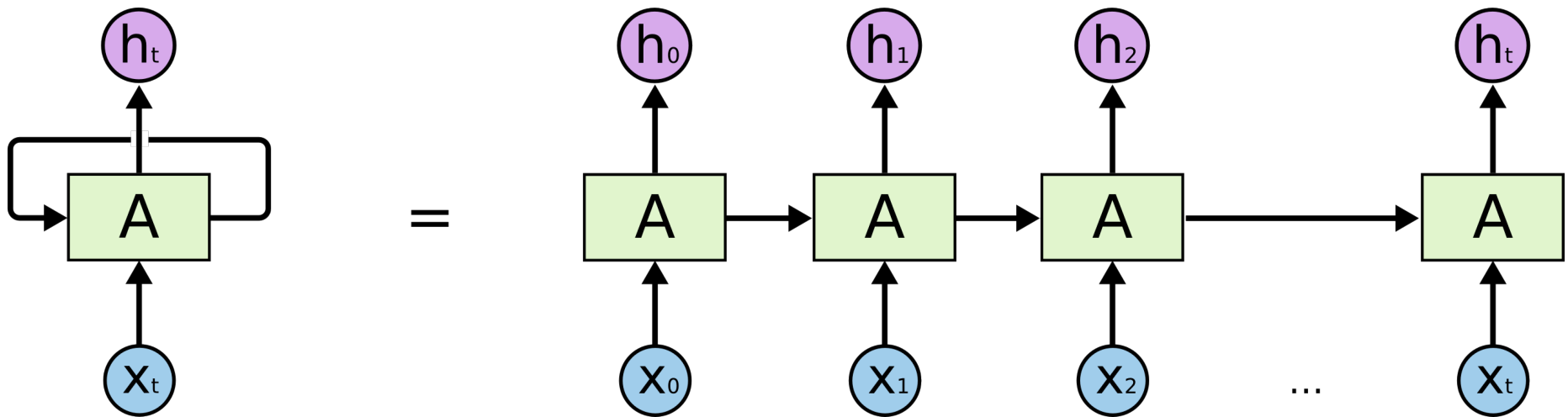


Source: “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”, Bai et al., arXiv 2018

Recurrent architectures

Recurrent Neural Networks (RNNs) Recurrent architectures

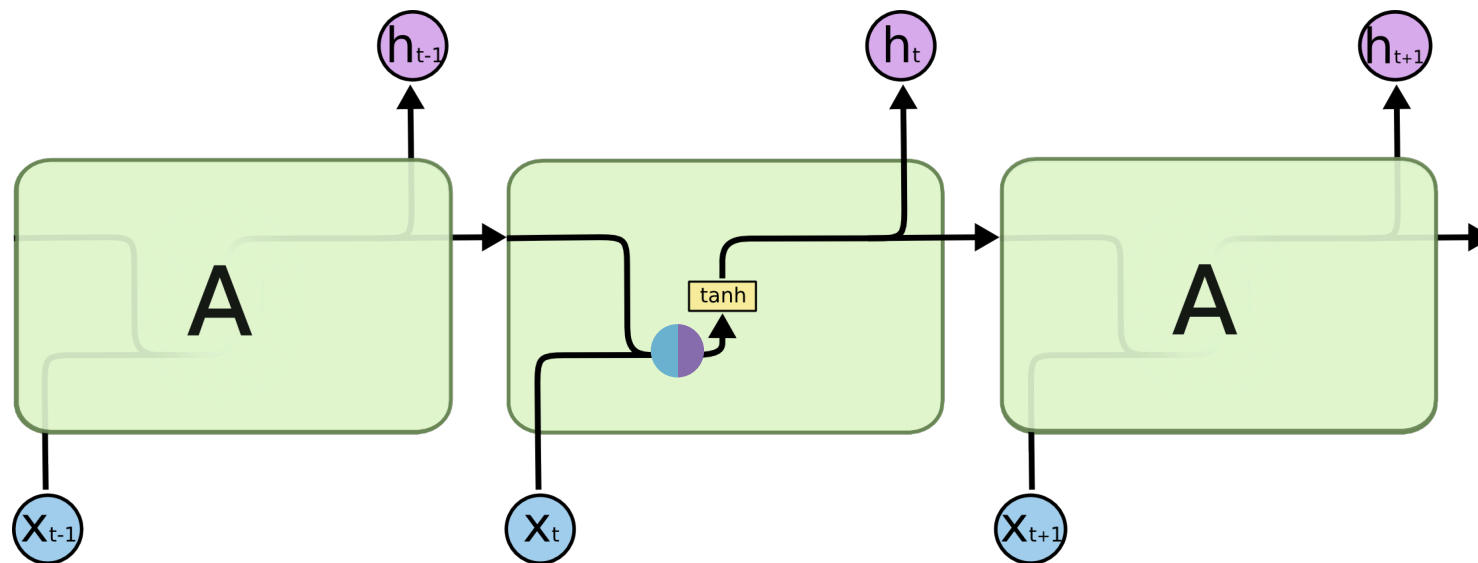
- Very flexible model (any length, let the model learn its memory needs, ...)



Source: [Christopher Olah's blog](#)

- Hidden state is computed as:

$$h_t = \varphi(\text{Linear combination of } x_t \text{ and } h_{t-1})$$

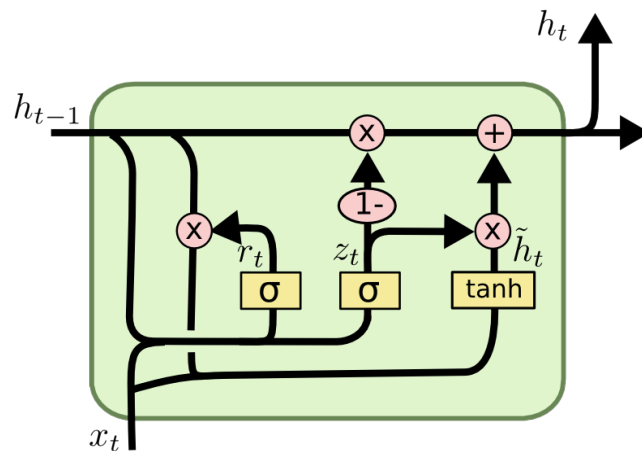


Source: [Christopher Olah's blog](#)

● x_t ● h_{t-1} ● Linear combination of x_t and h_{t-1}

- Very flexible model (any length, let the model learn its memory needs, ...)
- Difficult to learn in practice
 - Slow (lack of parallelism)
 - Vanishing gradients (hard to learn long-term dependencies) or exploding gradients (if φ is unbounded)

- At each time step, keep only part of the information
 - Through **gating mechanism**



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

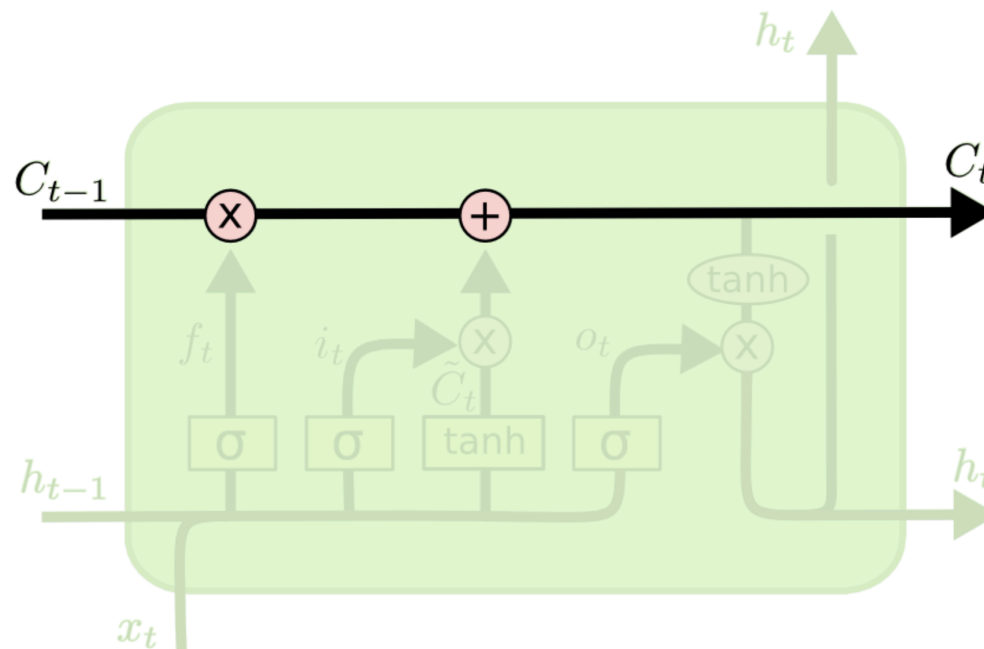
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

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

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

Source: [Christopher Olah's blog](#)

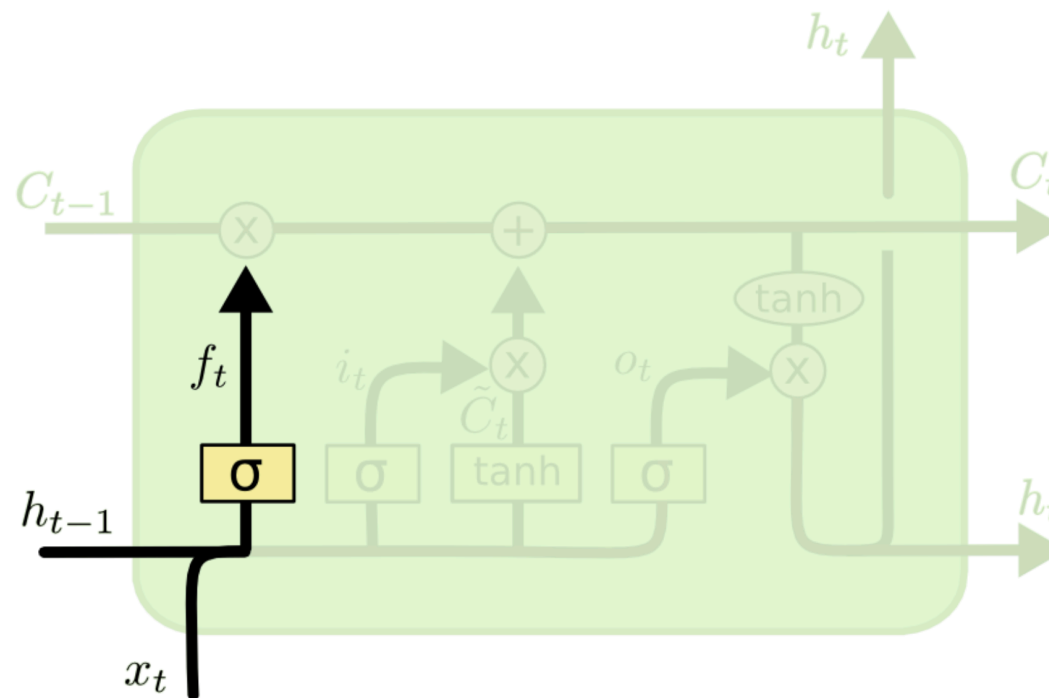
- Similar ideas as in GRUs, but:
 - an additional *cell state* C_t



Source: [Christopher Olah's blog](#)

- input and forget gates are made independent (in place of z_t in GRU)

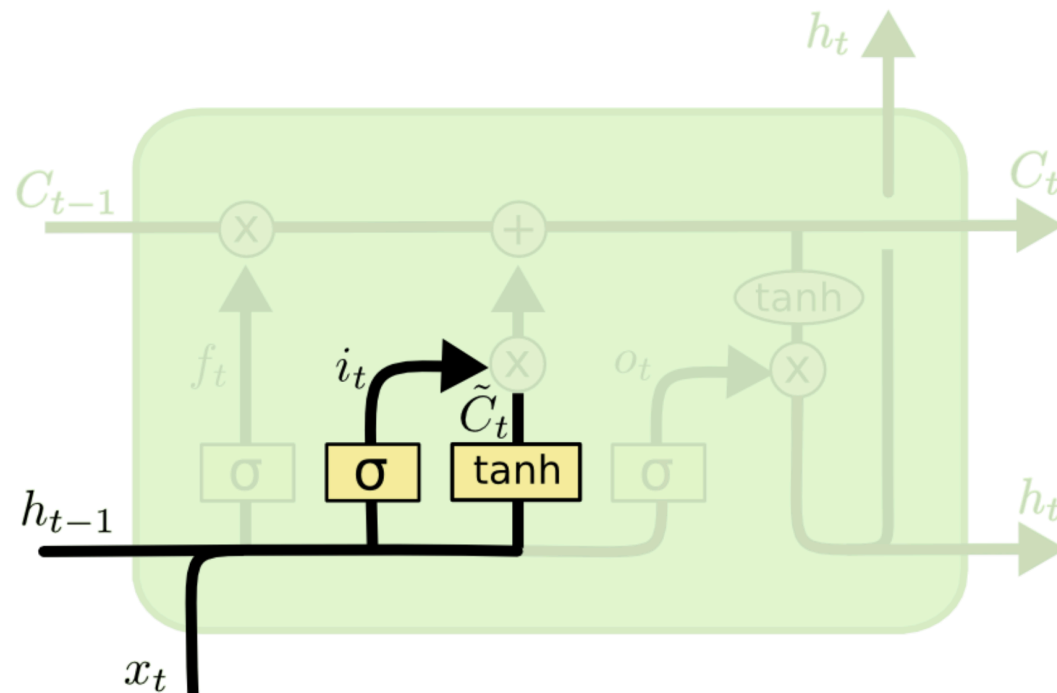
- **Forget gate:** $f_t = \sigma(\text{●})$



Source: [Christopher Olah's blog](#)

● x_t ● h_{t-1} ● Linear combination of x_t and h_{t-1}

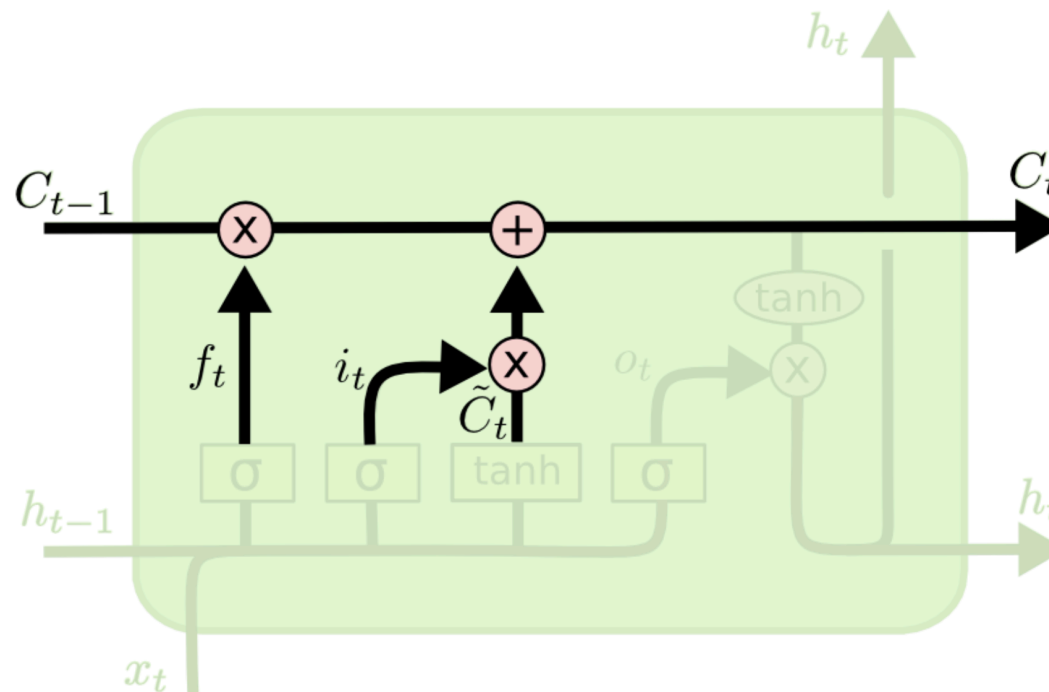
- **Input gate:** $i_t = \sigma(\text{●})$
- **Suggested C_t update:** $\tilde{C}_t = \varphi(\text{●})$



Source: [Christopher Olah's blog](#)

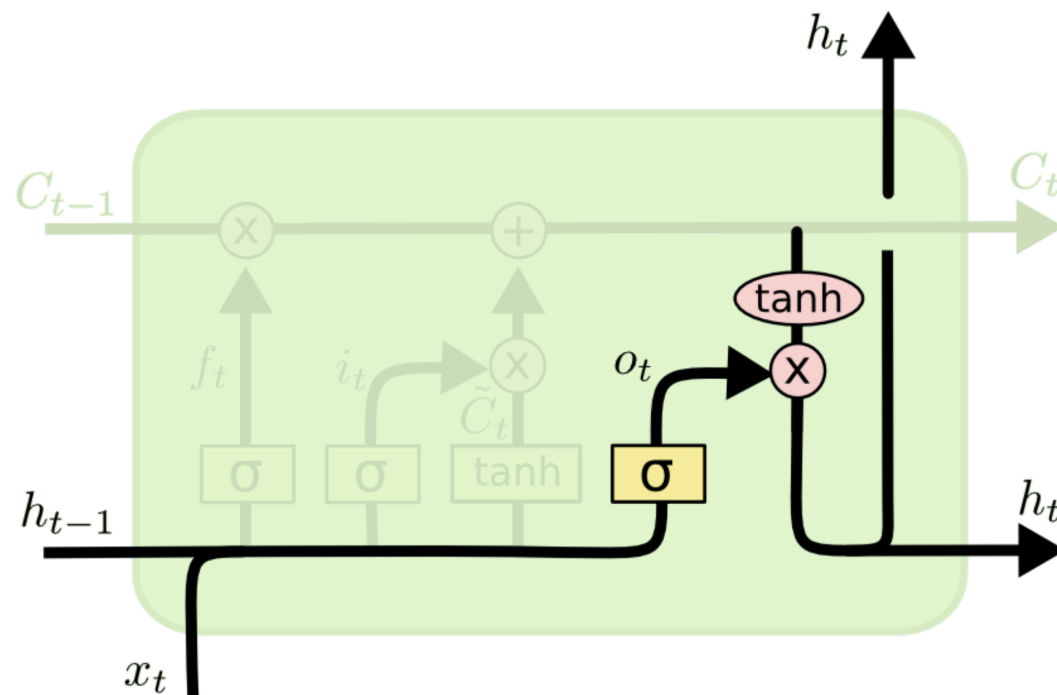
● x_t ● h_{t-1} ● Linear combination of x_t and h_{t-1}

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



Source: [Christopher Olah's blog](#)

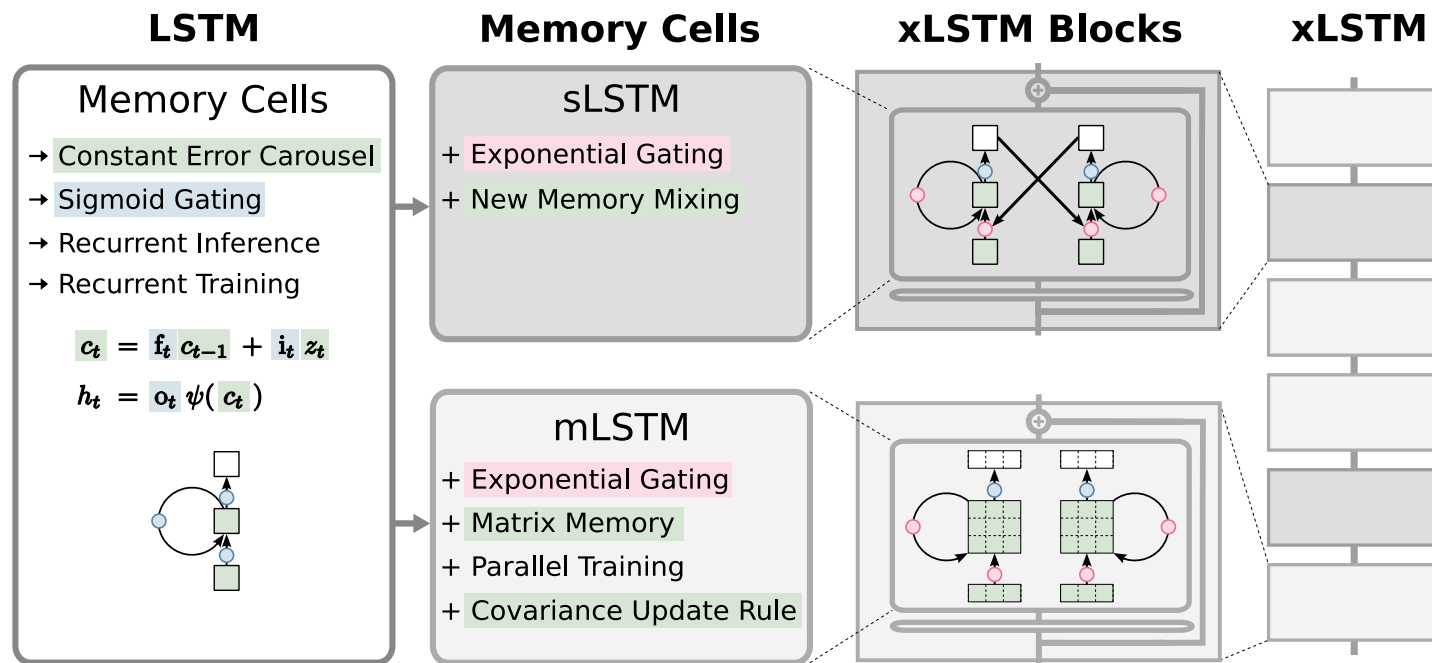
- **Output gate:** $o_t = \sigma(\text{Linear combination of } x_t \text{ and } h_{t-1})$
- **Hidden state update rule:** $h_t = o_t \odot \varphi(C_t)$



Source: [Christopher Olah's blog](#)

● x_t ● h_{t-1} ● Linear combination of x_t and h_{t-1}

- ## connections, *à la* Transformer



Source: “xLSTM: Extended Long Short-Term Memory” by Beck et al., NeurIPS

2024

- What's “new”?
 - In both sLSTM and mLSTM layers:
 - Exponential activation (to face vanishing gradients)
 - In sLSTM only:
 - Multi-head
 - In mLSTM only:
 - Novel memory store
 - Drop recurrence for gate computations: better parallelism

- Exponential activation for input and forget gates:

$$i_t = \exp(\text{●})$$

$$f_t = \max(\exp(\text{●}), \sigma(\text{●}))$$

- Multi-head: keep separate linear combinations per head

● x_t ● h_{t-1} ● Linear combination of x_t and h_{t-1}

- Exponential activation as in sLSTM
- Memory store

$$C_t = f_t \odot C_{t-1} + i_t \odot v_t k_t^\top$$

$$\tilde{h}_t = C_t q_t \quad (\text{up to normalization})$$

- Simplified case (no gate): similar to QKV in self-attention
- Drop recurrence for gate computations: better parallelism

$$i_t = \exp(\text{●})$$

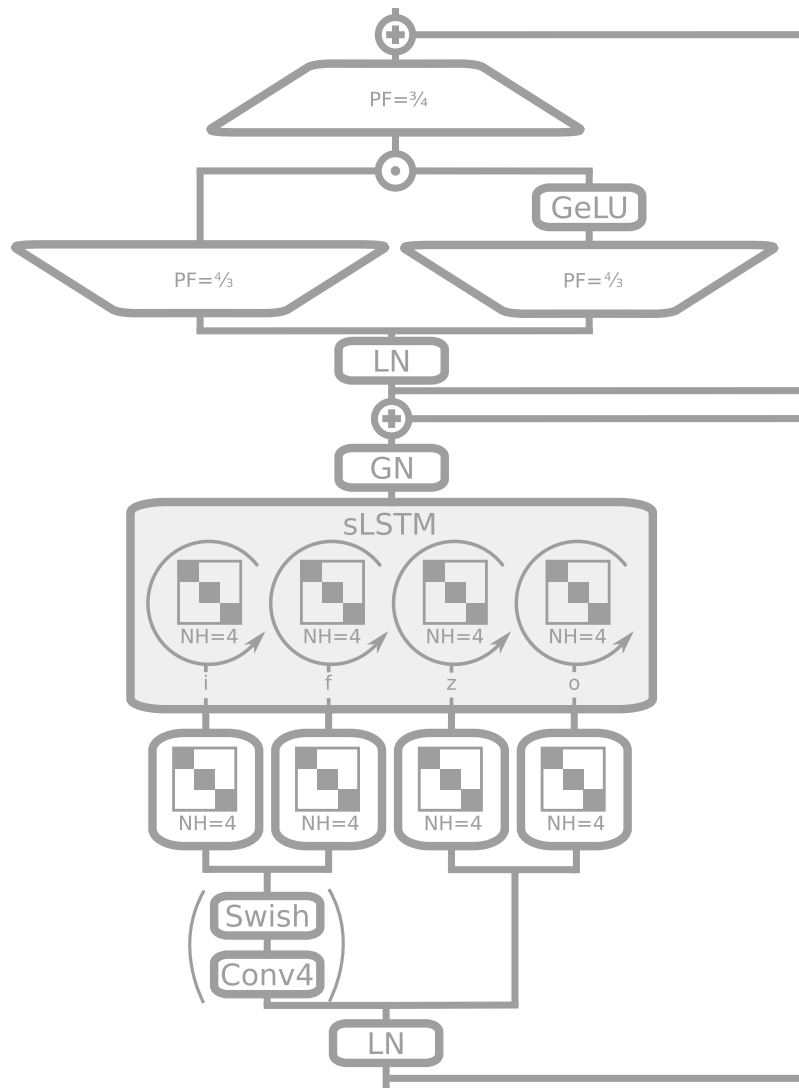
$$f_t = \max(\exp(\text{●}), \sigma(\text{●}))$$

$$o_t = \sigma(\text{●})$$

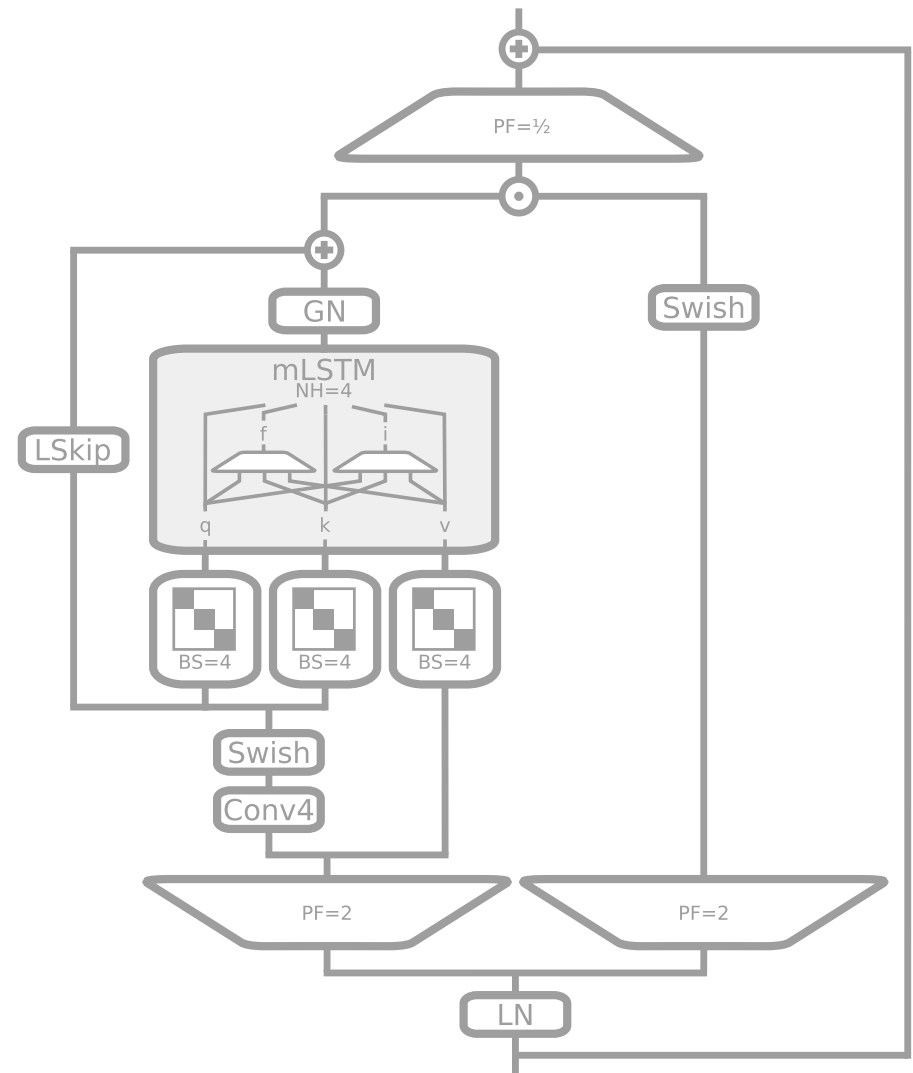
● x_t ● h_{t-1} ● Linear combination of x_t and h_{t-1}

xLSTM: Building blocks

Recurrent architectures



An sLSTM block



An mLSTM block