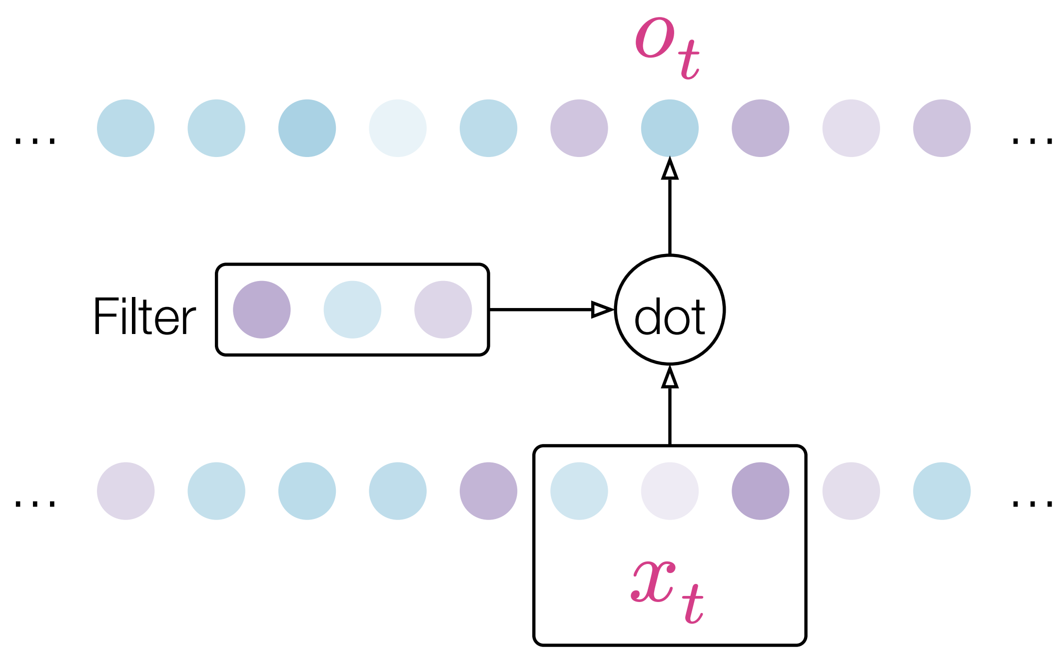
Deep Learning for Time Series

Session 2: ConvNets and Recurrent architectures

Romain Tavenard

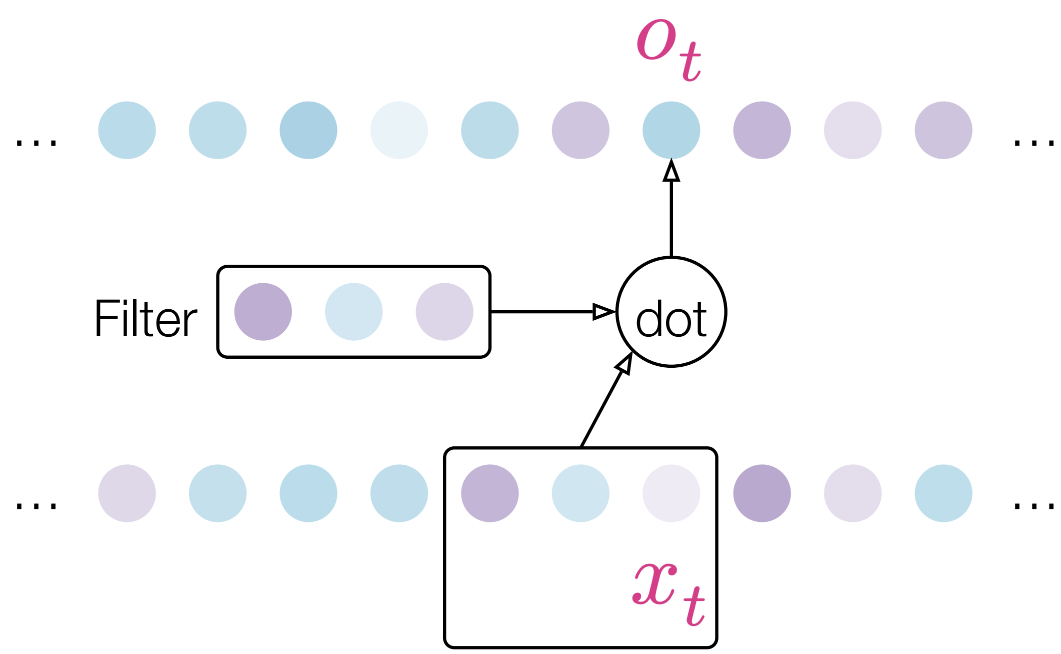
Convolutional architectures

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



Causal convolutions

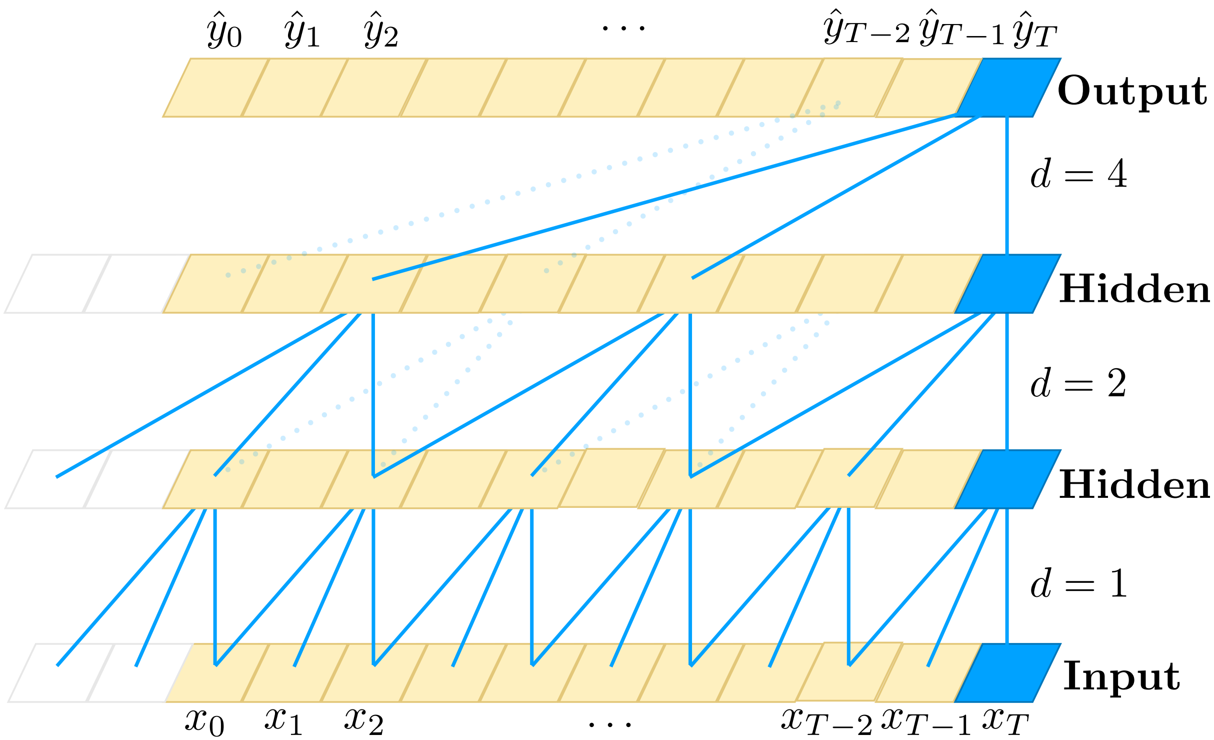
- Forecasting tasks: cannot access the future
- Causal convolution: convolve on past information alone (asymmetric window)



Temporal Convolution Network (TCN)

- Main idea: cascade dilated causal convolutions

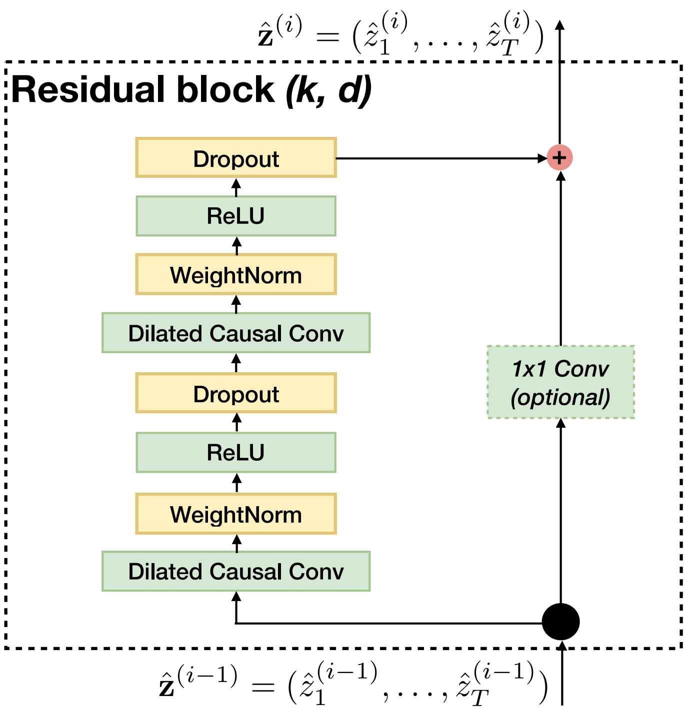
  $\Rightarrow$ Larger receptive field



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

- Additional improvements:
  - Residual connections

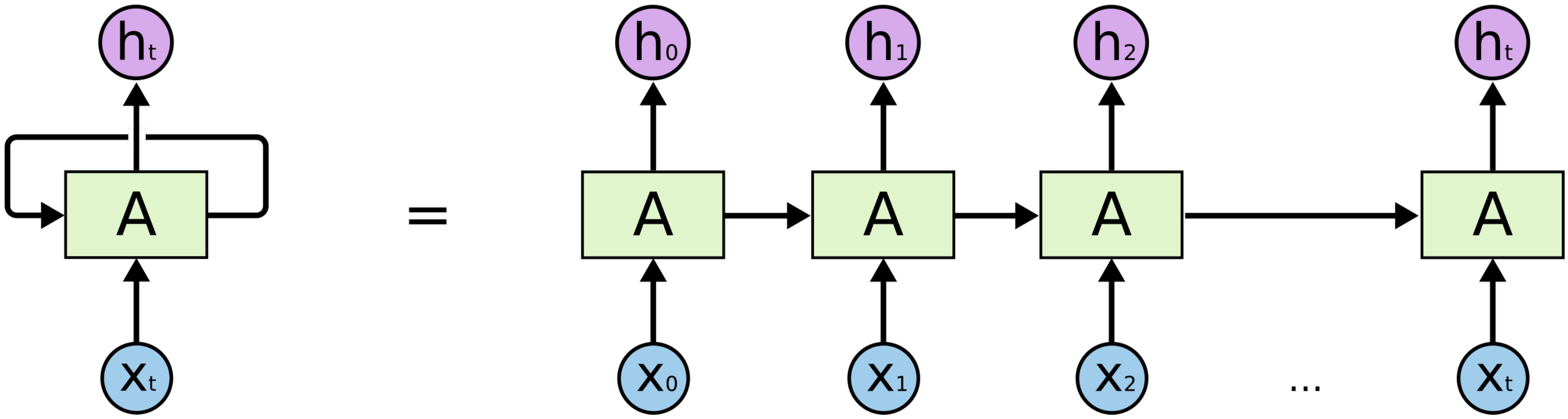
    $\Rightarrow$ Multi-resolution analysis
  - Normalization+Dropout layers



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

Recurrent architectures

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

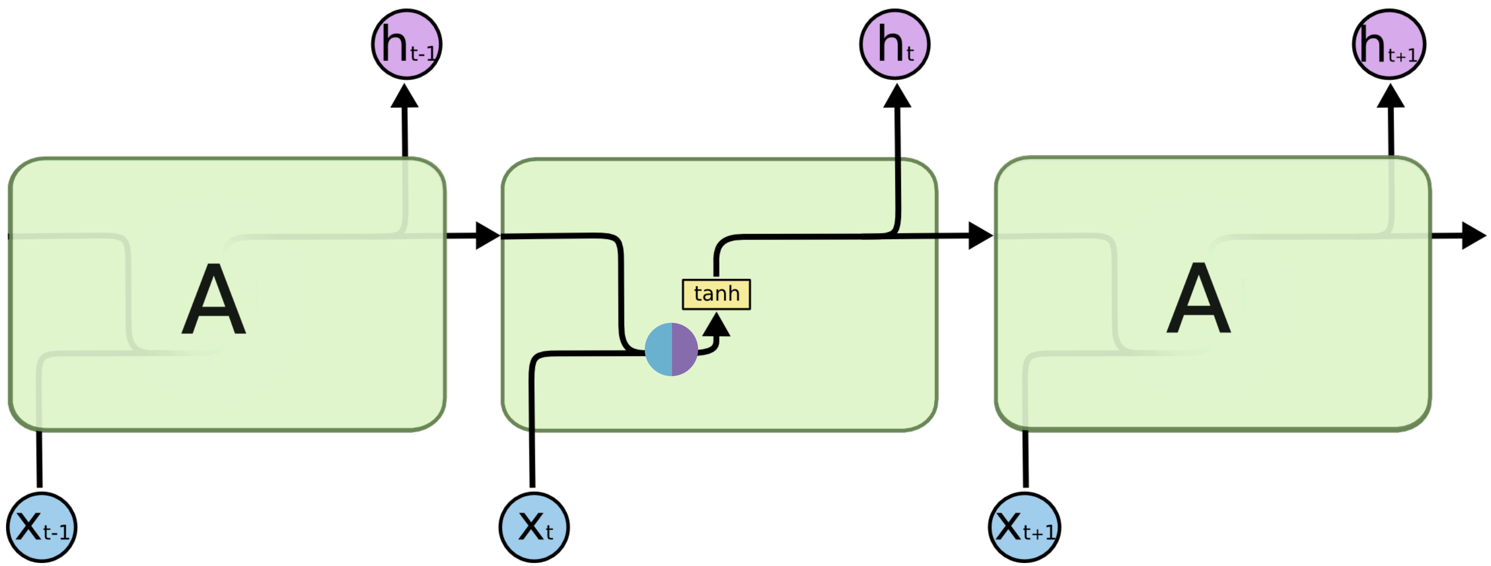


Source: Christopher Olah's blog

"Vanilla" RNN

- Hidden state is computed as:

$$h_t = \varphi(\bullet)$$



Source: Christopher Olah's blog

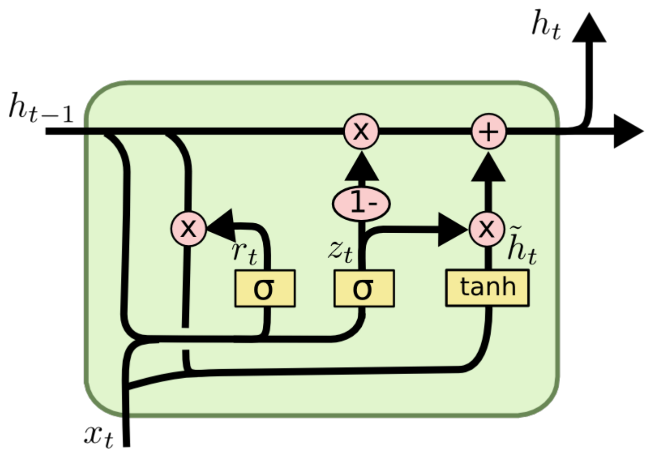
● x_t ● h_{t-1} ● Linear mixing of x_t and h_{t-1}

“Vanilla” RNN properties

- 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)

Gated Recurrent Unit (GRU)

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



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

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

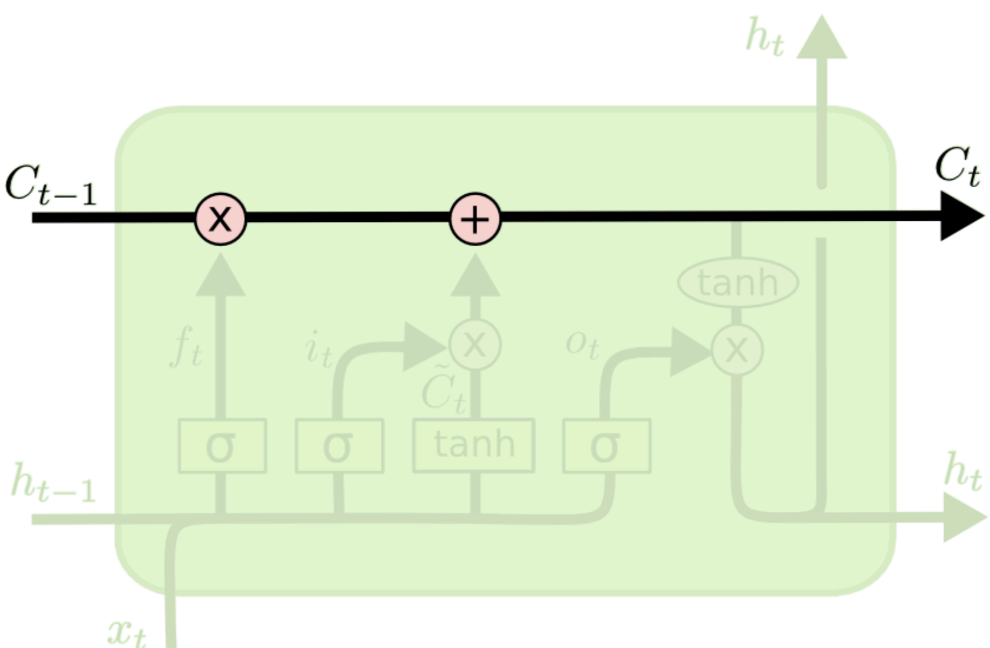
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

Source: Christopher Olah's blog

Long Short Term Memory (LSTM)

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

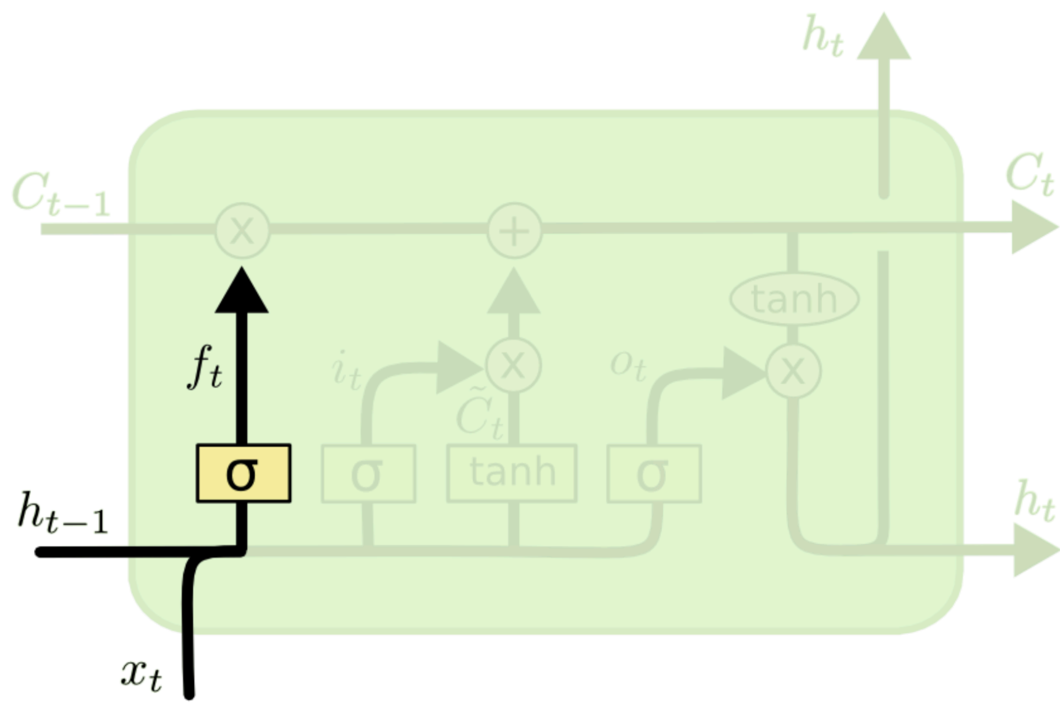


Source: Christopher Olah's blog

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

Long Short Term Memory (LSTM)

- **Forget gate**: $f_t = \sigma(◑)$

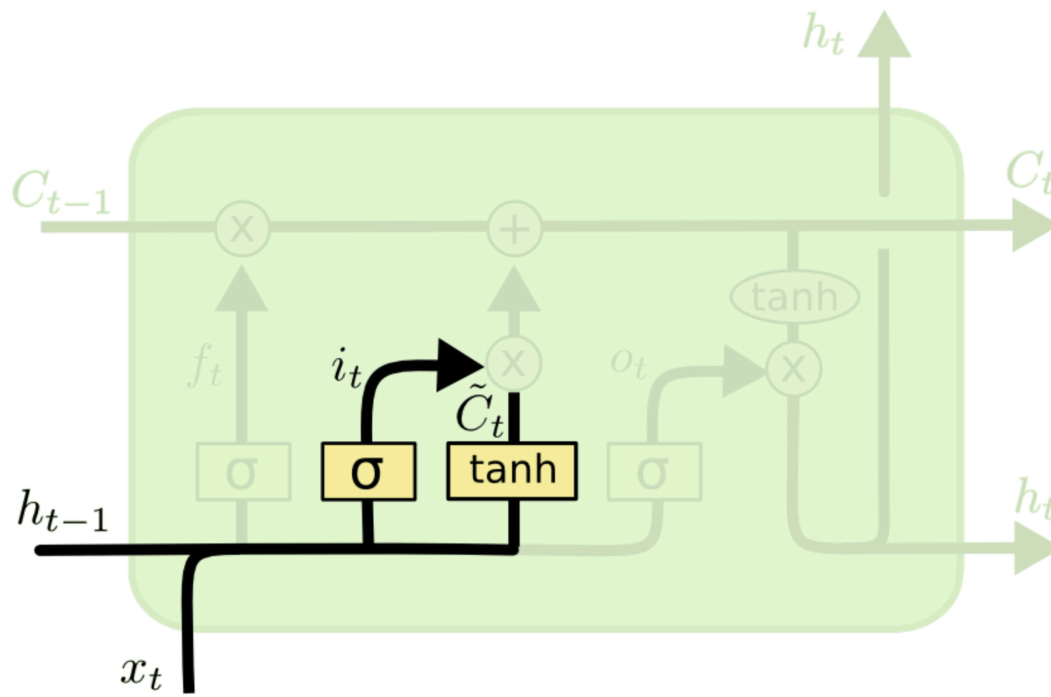


Source: Christopher Olah's blog

● x_t ● h_{t-1} ◑ Linear mixing of x_t and h_{t-1}

Long Short Term Memory (LSTM)

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

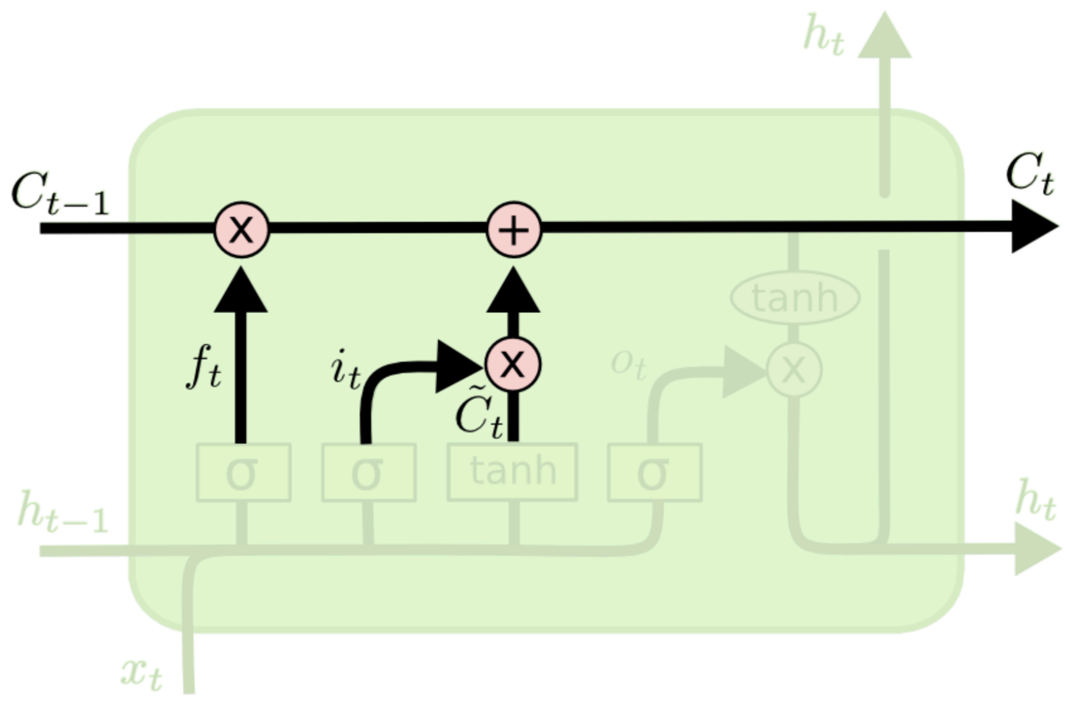


Source: Christopher Olah's blog

x_t h_{t-1} Linear mixing of x_t and h_{t-1}

Long Short Term Memory (LSTM)

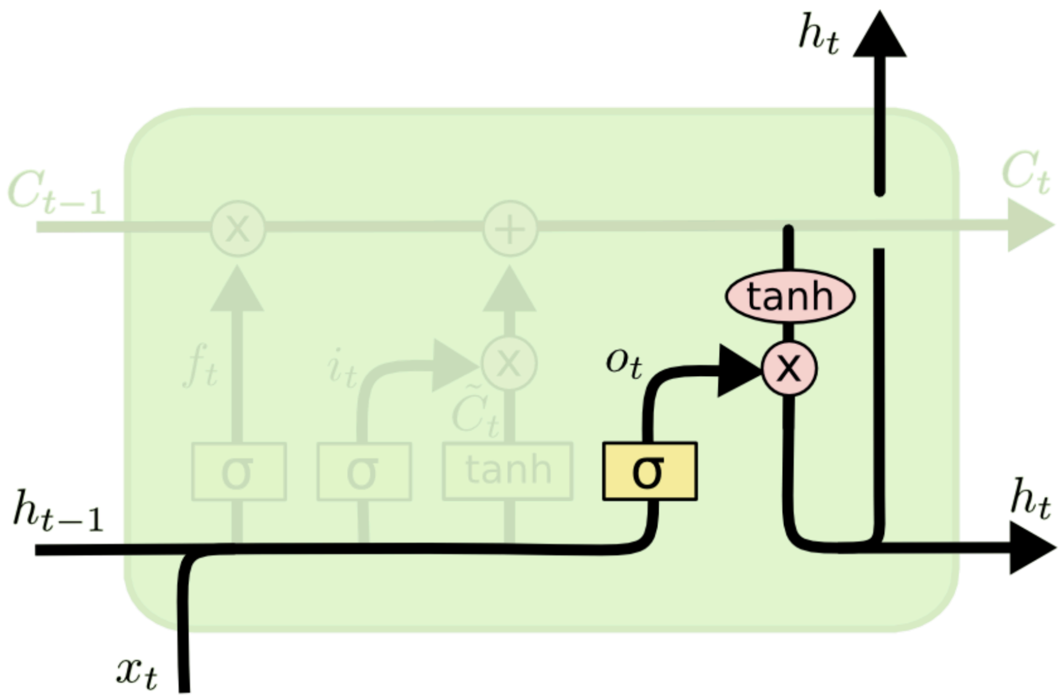
- **C_t update rule**: $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$



Source: Christopher Olah's blog

Long Short Term Memory (LSTM)

- **Output gate**: $o_t = \sigma(\ )$
- **Hidden state update rule**: $h_t = o_t \odot \varphi(C_t)$

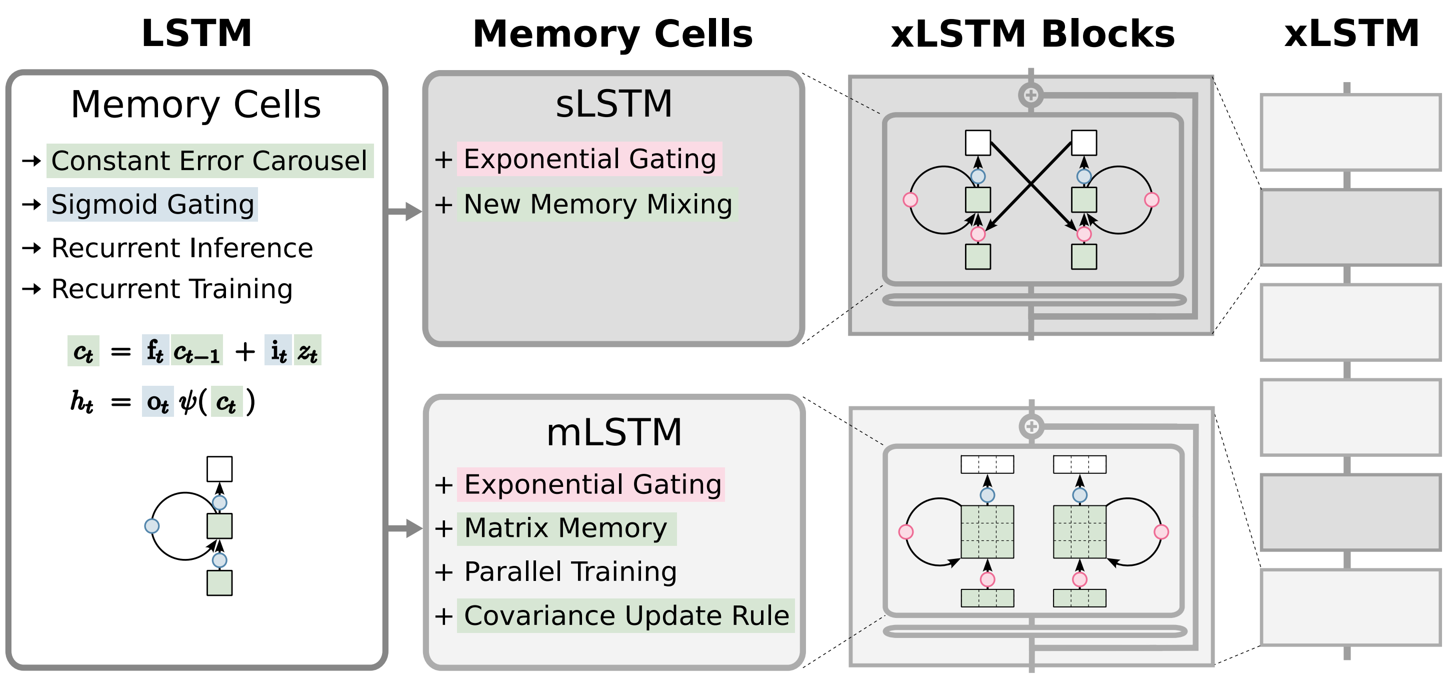


Source: Christopher Olah's blog

x_t h_{t-1} Linear mixing of x_t and h_{t-1}

xLSTM

- A "modern" LSTM variant
  - Made of sLTSM and mLSTM layers
  - Embedded in blocks with normalization layers, residual connections, *à la* Transformer



Source: "xLSTM: Extended Long Short-Term Memory" by Beck et al., NeurIPS 2024

xLSTM

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

xLSTM: focus on sLSTM layers

- Exponential activation for input and forget gates:

$$i_t = \exp(\text{◐})$$

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

- Multi-head: keep separate linear mixings per head

● x_t ● h_{t-1} ◐ Linear mixing of x_t and h_{t-1}

xLSTM: focus on mLSTM layers

- Exponential activation as in sLSTM
- Memory store

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

$$\tilde{h_t} = C_t q_t \qquad \text{(up to normalization)}$$

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

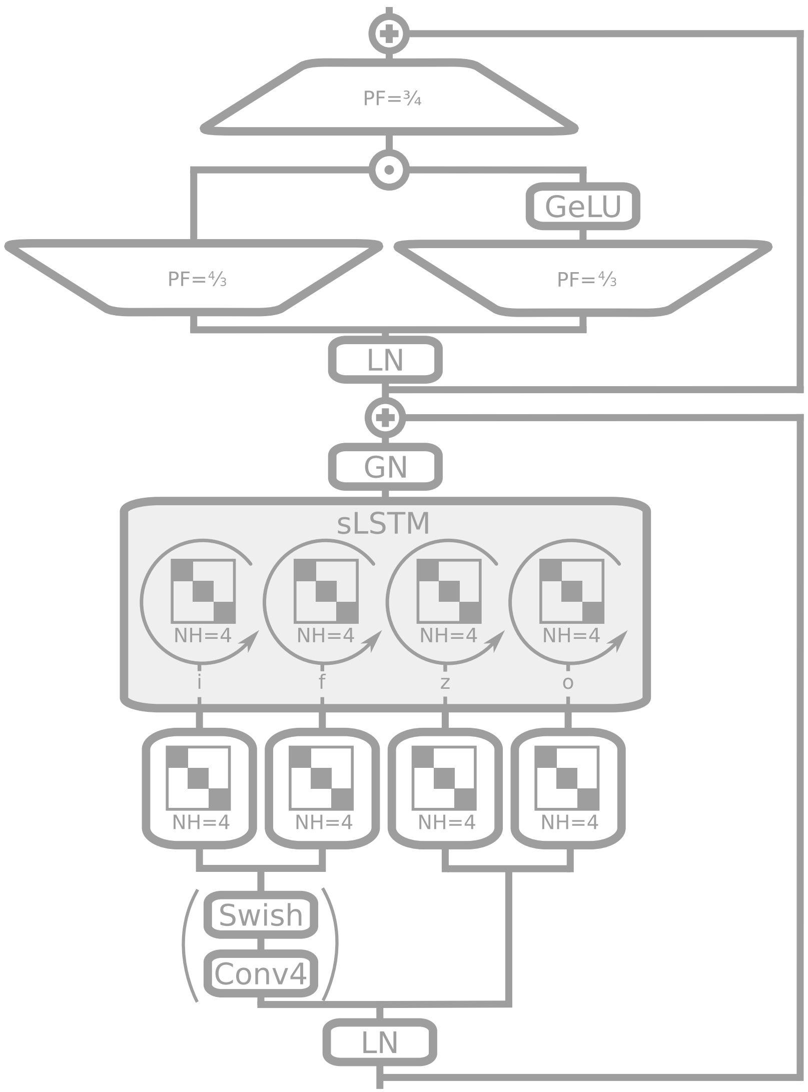
$$i_t = \exp(●)$$

$$f_t = \max(\exp(●), \sigma(●))$$

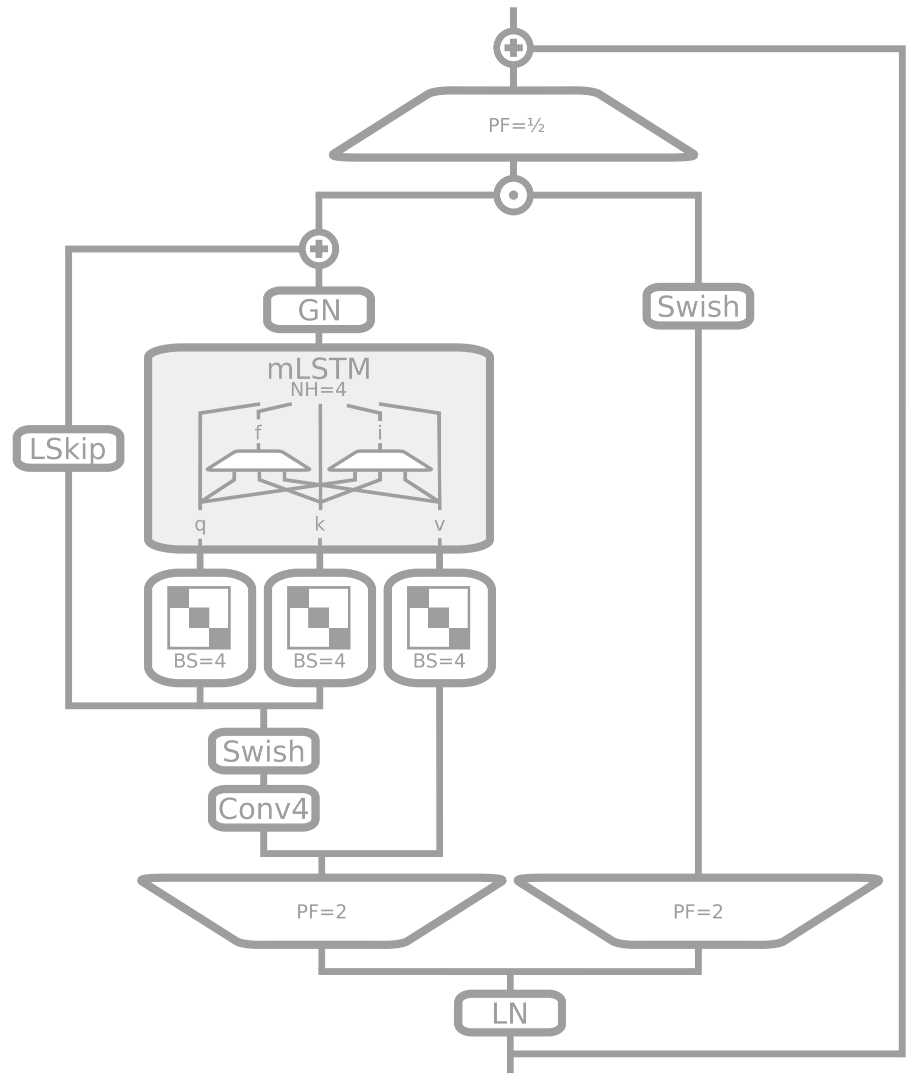
$$o_t = \sigma(●)$$

● x_t ● h_{t-1} ◐ Linear mixing of x_t and h_{t-1}

xLSTM: Building blocks



An sLSTM block



An mLSTM block