

# Long Short-Term Memory (LSTM)

We have a sequence of inputs  $x^{(\#)}$ , and we will compute a sequence of hidden states  $h^{(\#)}$  and cell states  $c^{(\#)}$ . On timestep  $t$ :

**Forget gate:** controls what is kept vs forgotten, from previous cell state

**Input gate:** controls what parts of the new cell content are written to cell

**Output gate:** controls what parts of cell are output to hidden state

**New cell content:** this is the new content to be written to the cell

**Cell state:** erase (“forget”) some content from last cell state, and write (“input”) some new cell content

**Hidden state:** read (“output”) some content from the cell

**Sigmoid function:** all gate values are between 0 and 1

$$\begin{aligned} f^{(t)} &= \sigma \left( W_f h^{(t-1)} + U_f x^{(t)} + b_f \right) \\ i^{(t)} &= \sigma \left( W_i h^{(t-1)} + U_i x^{(t)} + b_i \right) \\ o^{(t)} &= \sigma \left( W_o h^{(t-1)} + U_o x^{(t)} + b_o \right) \end{aligned}$$

$$\tilde{c}^{(t)} = \tanh \left( W_c h^{(t-1)} + U_c x^{(t)} + b_c \right)$$

$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

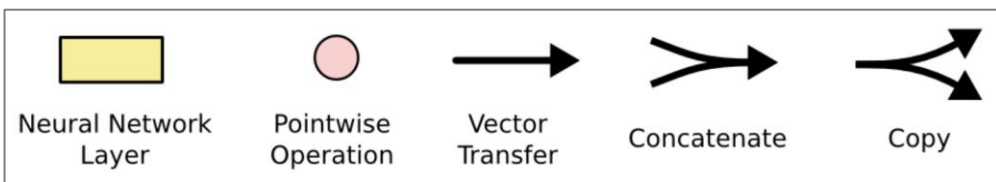
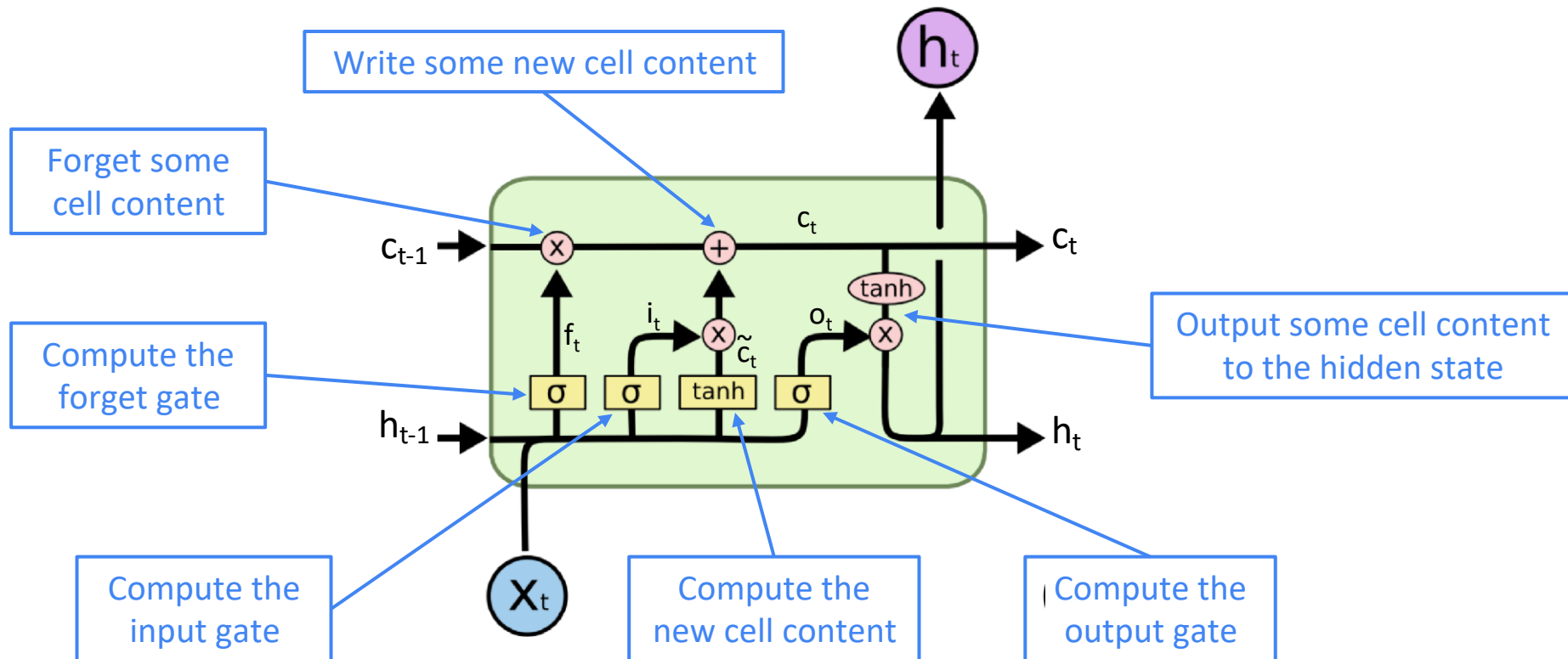
$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

Gates are applied using element-wise product

All these are vectors of same length  $n$

# Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:



# Gated Recurrent Units (GRU)

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep  $t$  we have input  $\mathbf{x}^{(t)}$  and hidden state  $\mathbf{h}^{(t)}$  (no cell state).

**Update gate:** controls what parts of hidden state are updated vs preserved

$$\mathbf{u}^{(t)} = \sigma \left( \mathbf{W}_u \mathbf{h}^{(t-1)} + \mathbf{U}_u \mathbf{x}^{(t)} + \mathbf{b}_u \right)$$

**Reset gate:** controls what parts of previous hidden state are used to compute new content

$$\mathbf{r}^{(t)} = \sigma \left( \mathbf{W}_r \mathbf{h}^{(t-1)} + \mathbf{U}_r \mathbf{x}^{(t)} + \mathbf{b}_r \right)$$

**New hidden state content:** reset gate selects useful parts of prevhidden state. Use this and current input to compute new hidden content.

$$\tilde{\mathbf{h}}^{(t)} = \tanh \left( \mathbf{W}_h (\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{U}_h \mathbf{x}^{(t)} + \mathbf{b}_h \right)$$

$$\mathbf{h}^{(t)} = (1 - \mathbf{u}^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \circ \tilde{\mathbf{h}}^{(t)}$$

**Hidden state:** update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

**How does this solve vanishing gradient?**

Like LSTM, GRU makes it easier to retain info long-term (e.g. by setting update gate to 0)

# LSTM vs GRU

- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
- Rule of thumb: LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data); Switch to GRUs for speed and fewer parameters.
  - **Note**: LSTMs can store unboundedly\* large values in memory cell dimensions, and relatively easily learn to count. (Unlike GRUs.)