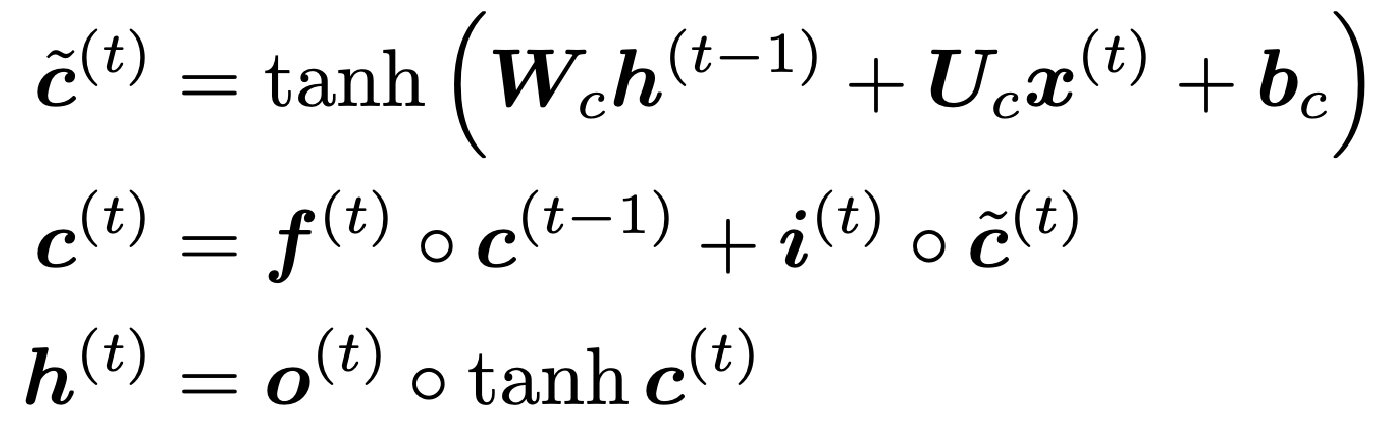
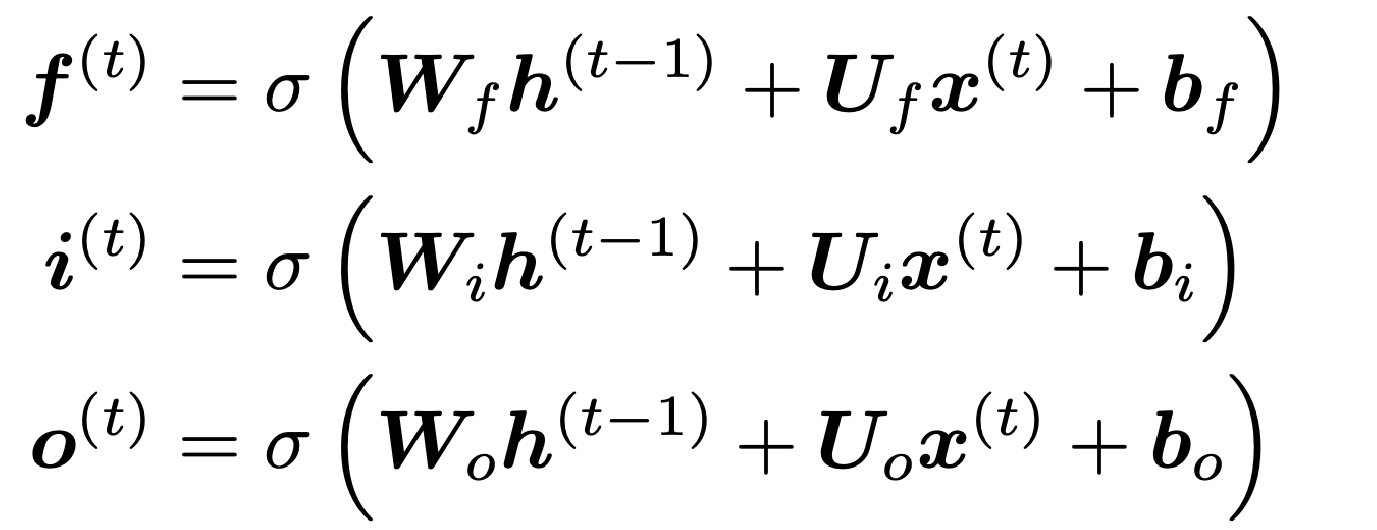
# Long Short-Term Memory (LSTM)

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



All these are vectors of same length

*n*

**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

23

Gates are applied using

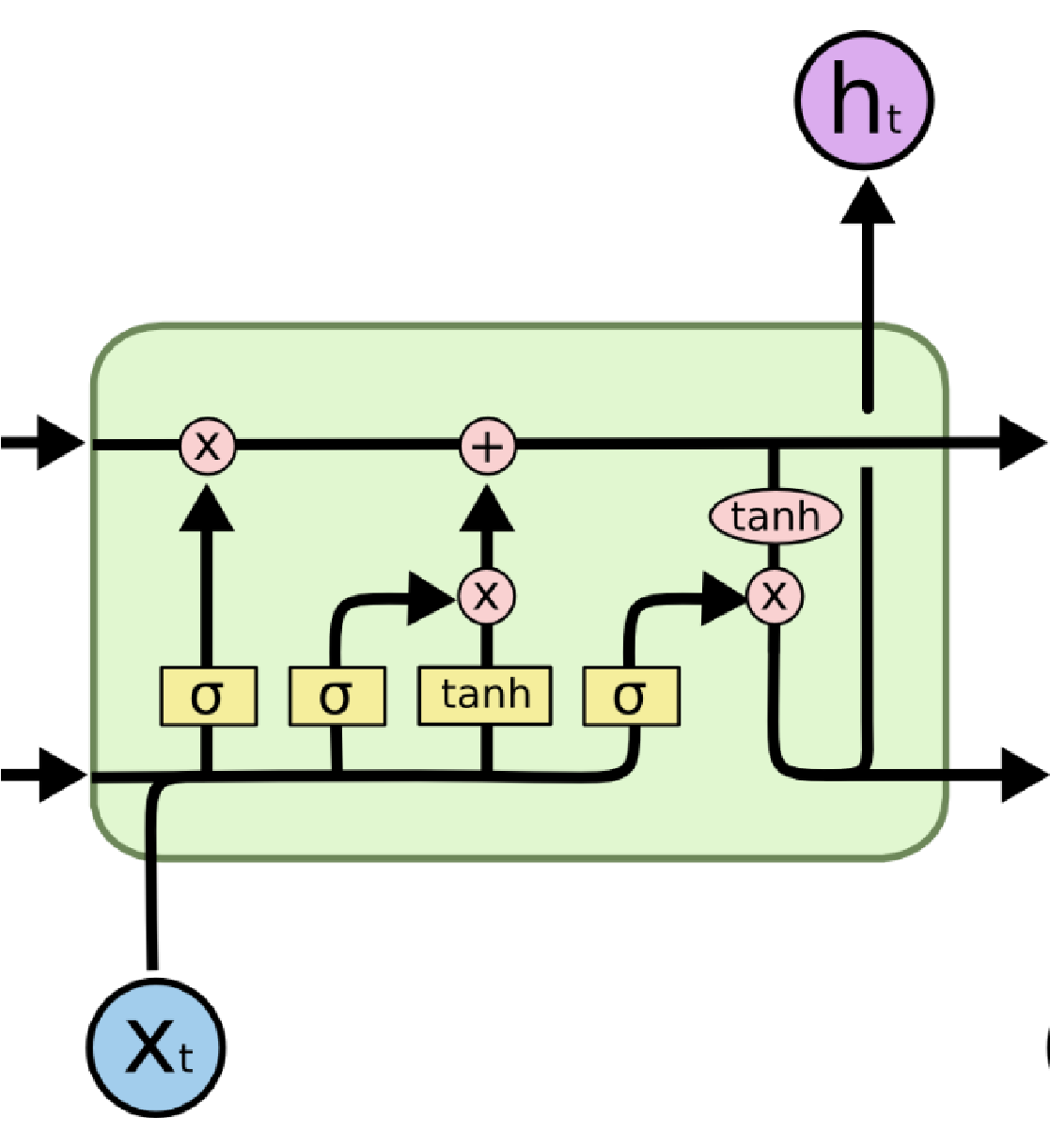
element

-

wise product

# Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:



c

t

-

1

h

t

-

1

c

t

h

t

f

t

i

t

o

t

c

t

c

t

~

Compute the

forget gate

Forget some

cell content

Compute the

input gate

Compute the

new cell content

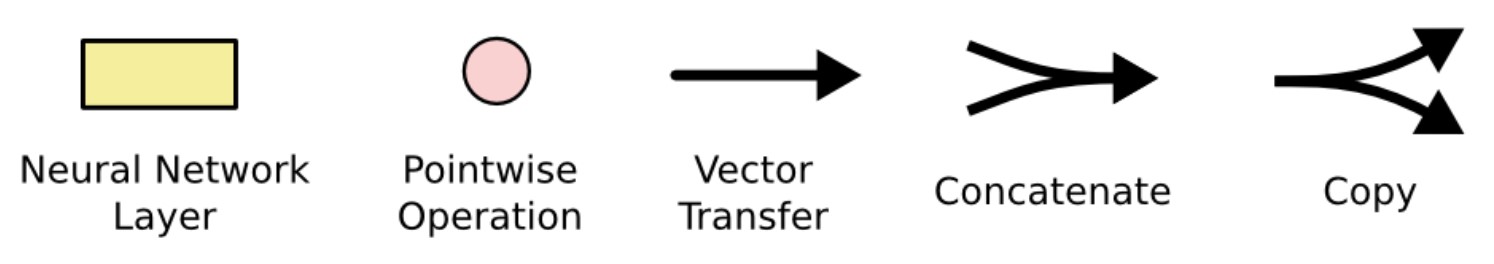
Compute the

output gate

Write some new cell content

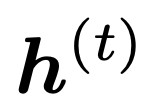
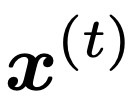
Output some cell content

to the hidden state



25 **Source:** http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# 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 and hidden state (no cell state).

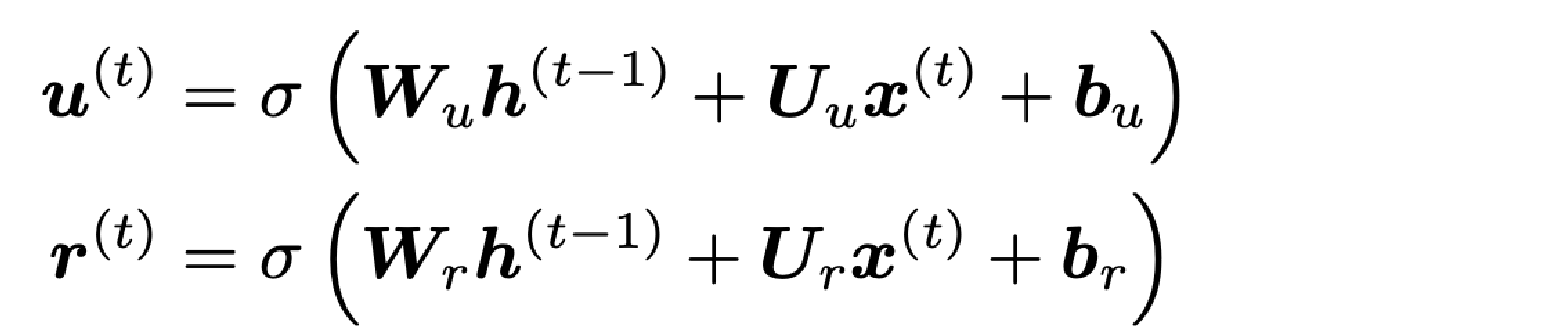
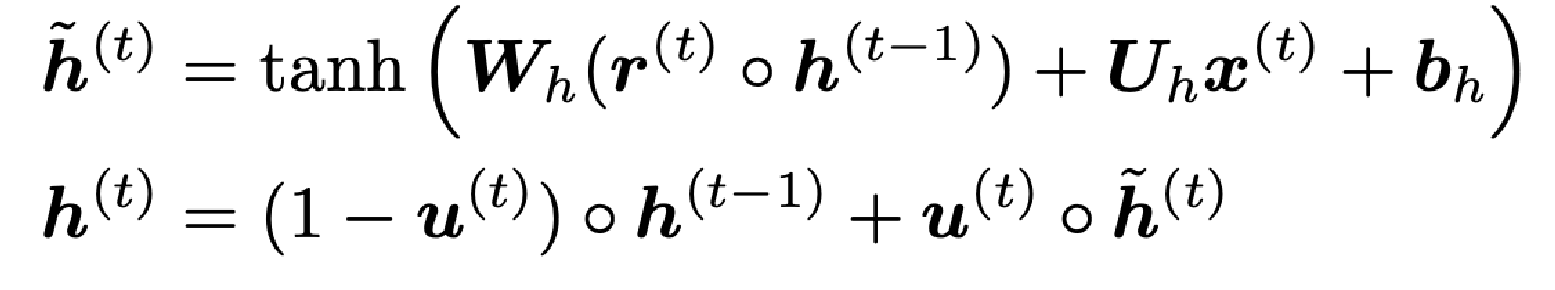
**Update gate:** controls what parts of hidden state are updated vs preserved

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

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

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



28 "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", Cho et al. 2014, https://arxiv.org/pdf/1406.1078v3.pdf

# 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.)

29 \*bounded if assuming finite precision, but still, >>1

**Source:** “On the Practical Computational Power of Finite Precision RNNs for Language Recognition”, Weiss et al., 2018. https://arxiv.org/pdf/1805.04908.pdf