CS 4973 & 6463 :

Advanced Topics in Data Science

Deep Learning

UTSA - Computer Science - Fall 2019

Recurrent Neural Networks

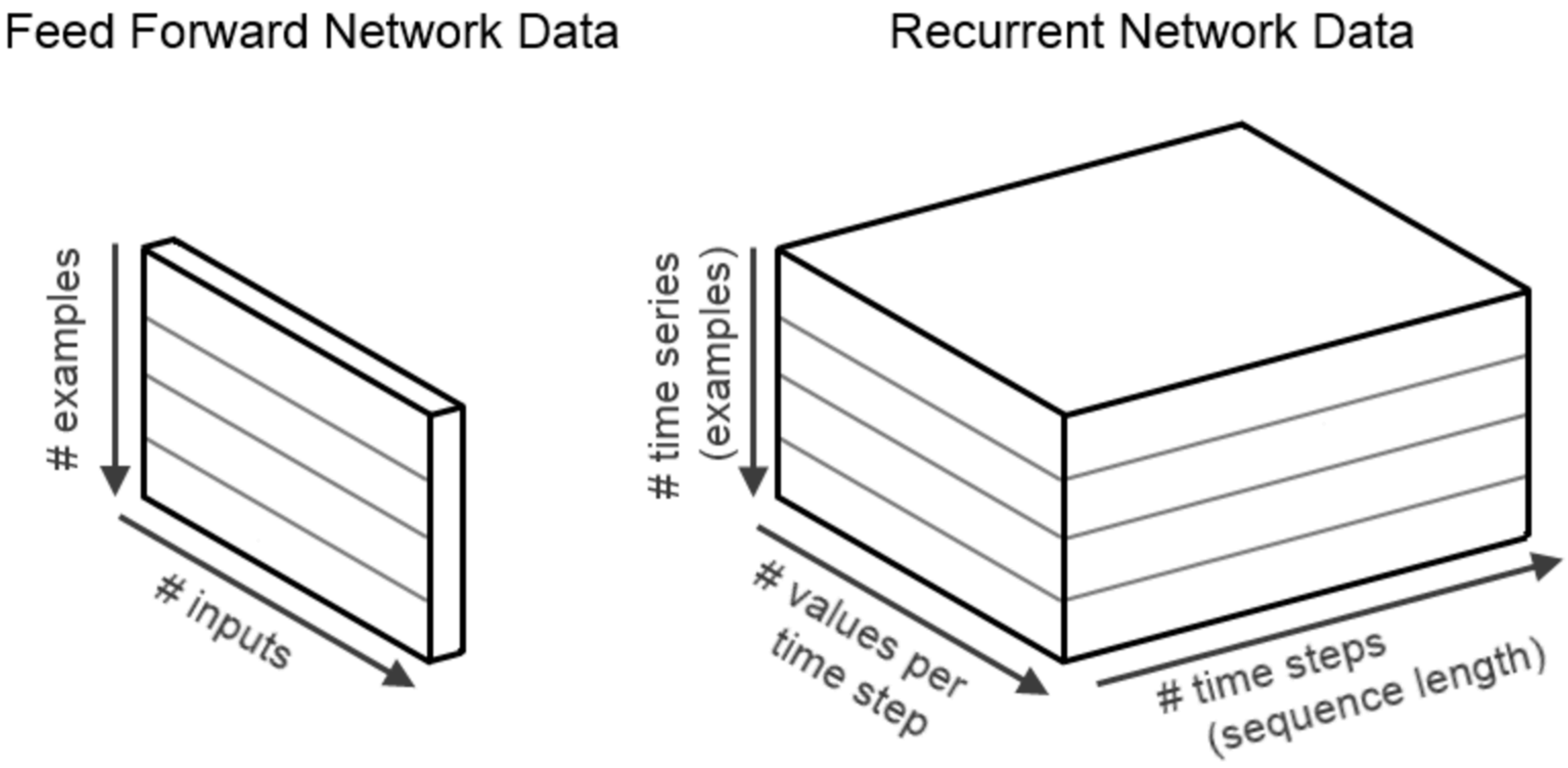
RNNs

- Problem: **inputs of variable size**
    - *Image data can be normalized. Text is more difficult..*

- Solution: **recurrent neural networks**
    - *Varying length sequences handled by **stateful neurons***

RNNs: Motivating Applications

- Time-series data
- Language processing
    - Speech recognition
    - Text generation
- Sensor data
- Genome sequences
- Stock prediction



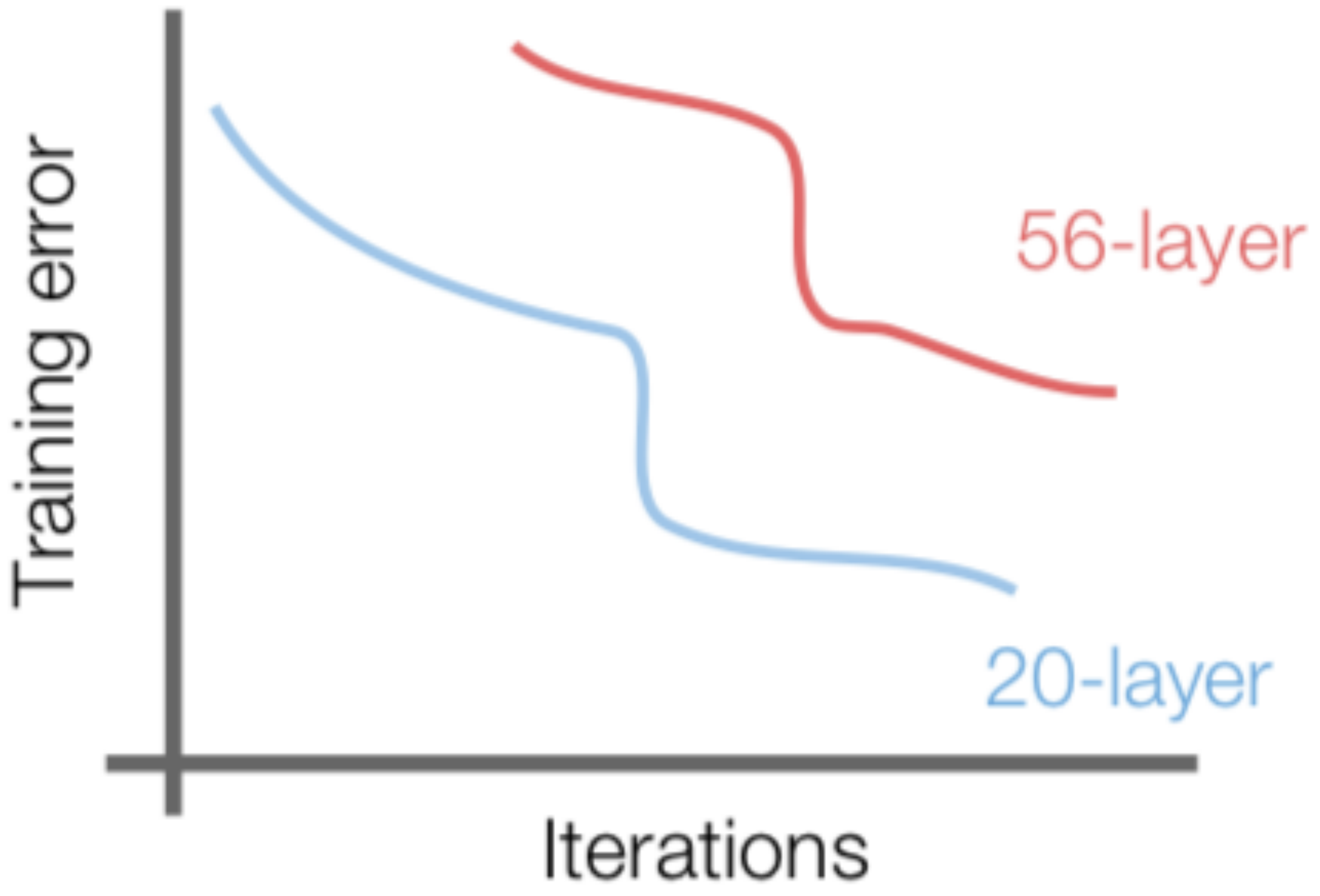
Beyond CNNs

Problem: **Deeper models are hard to optimize.**
Solution: **Construct deep model from learned layers of shallow model.**

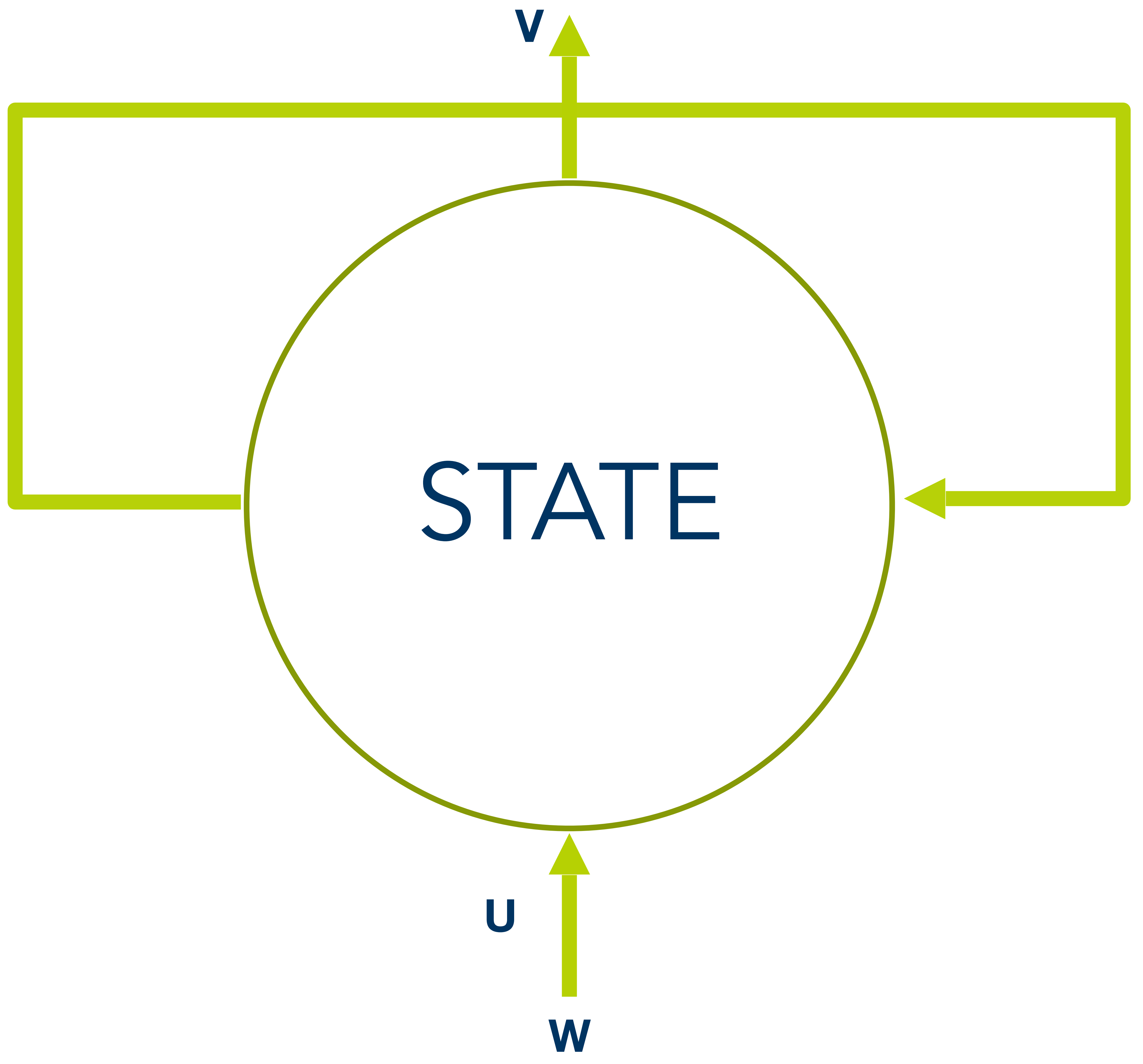
- Stacking deeper layers on CNNs may result in **worsening performance** in both training and testing.

- Is this due to overfitting?

No! See the pattern?



RNNs: Architecture



U encodes the input

[s x r]

W updates the info from the previous state

[s x s]

U+W is passed into the activation function

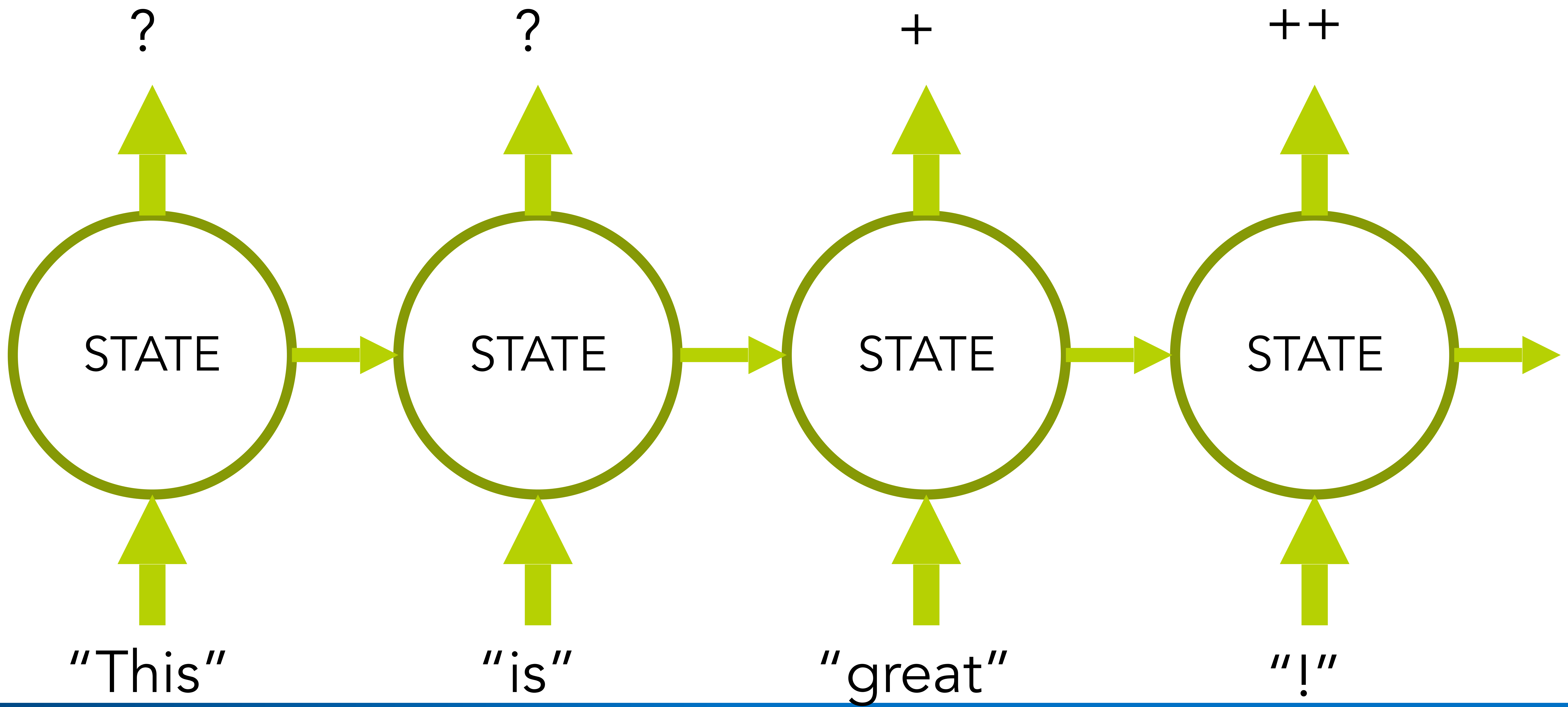
V is [t x s]

Let:
r = dim of input vector
s = dim of hidden state
t = dim output vector

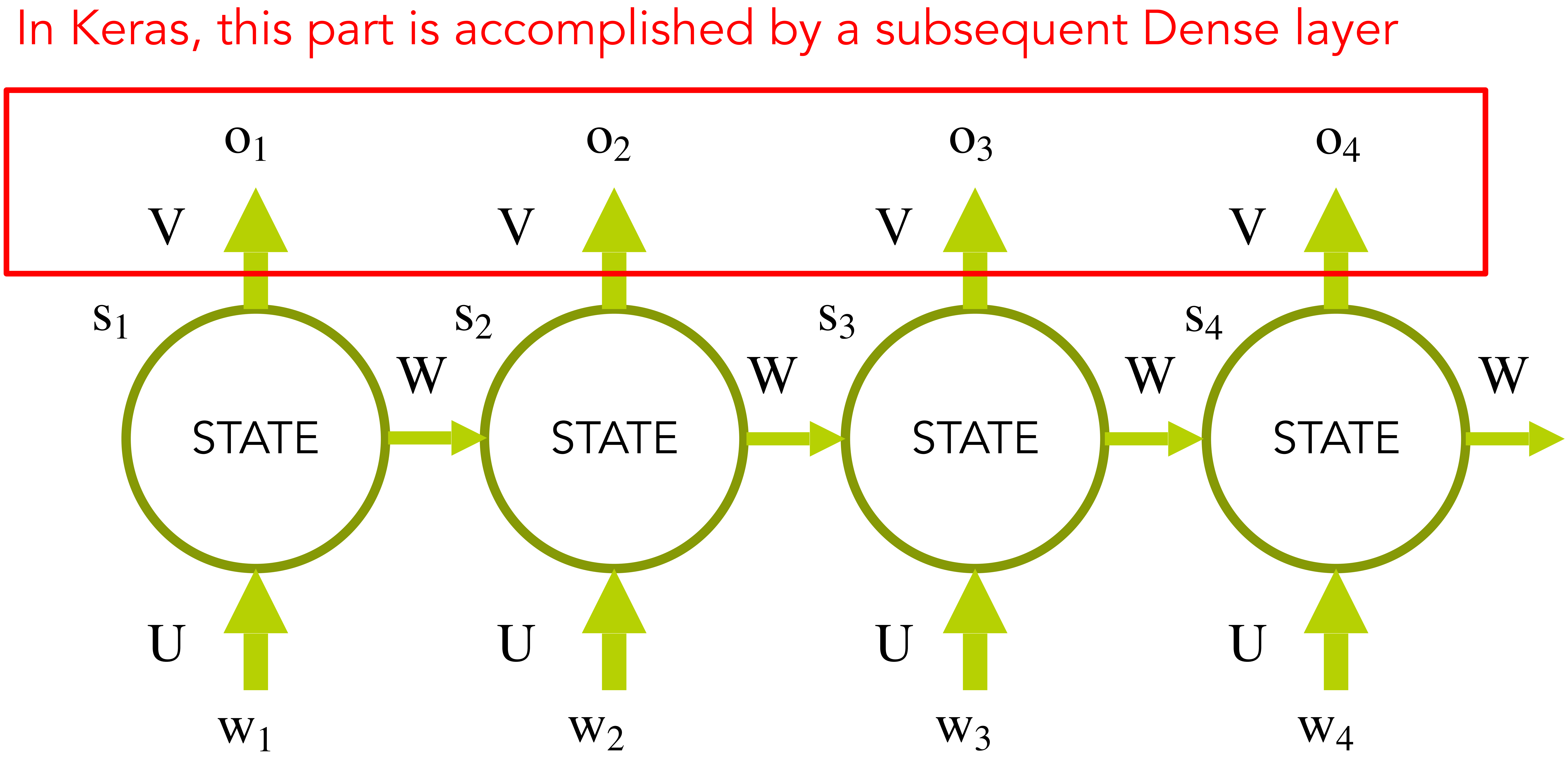
"Unfolding" an RNN

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

"Unrolling" the RNN

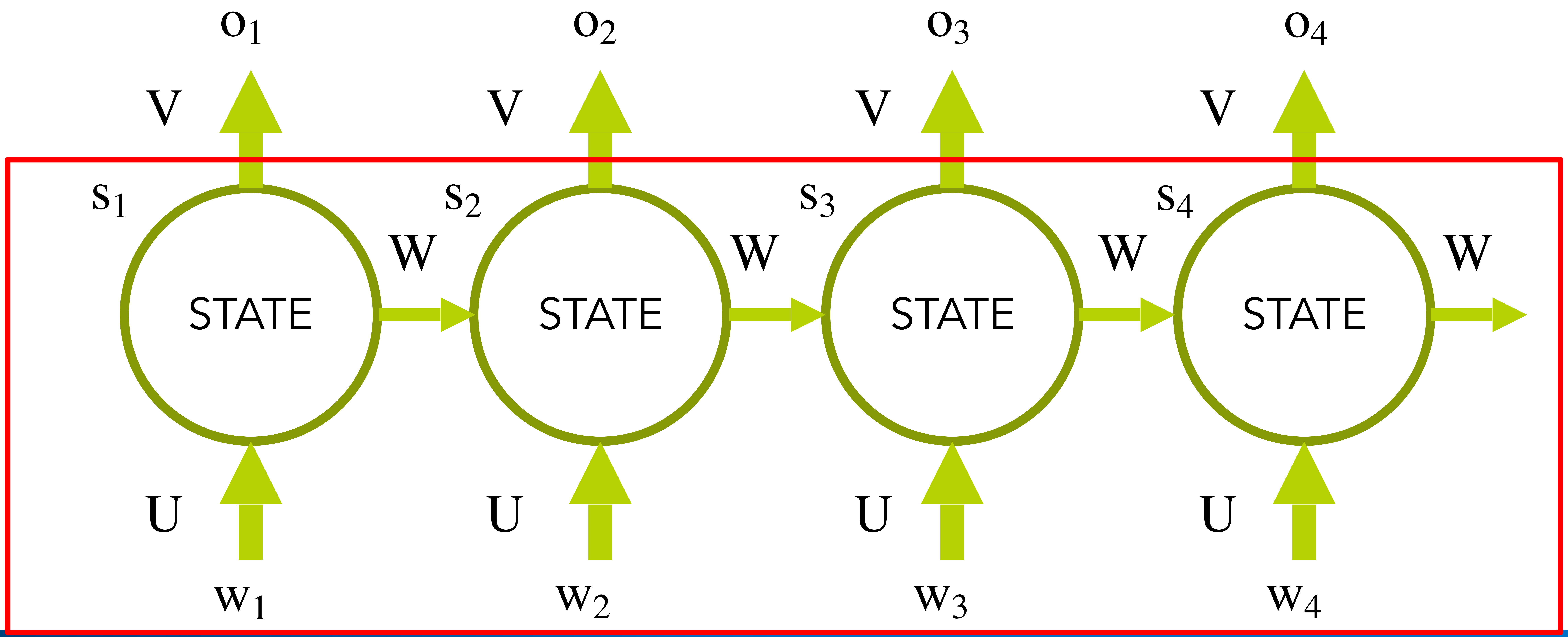


"Unrolling" the RNN



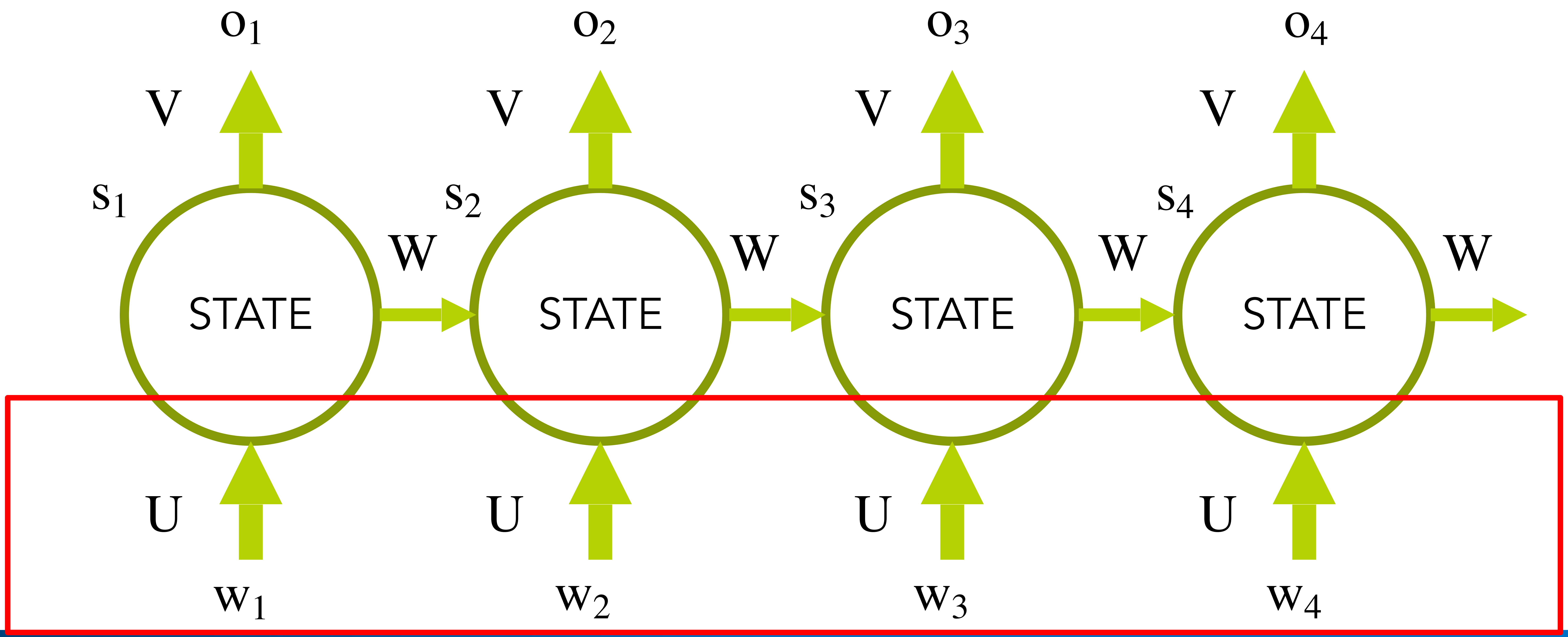
"Unrolling" the RNN

This part is the core RNN



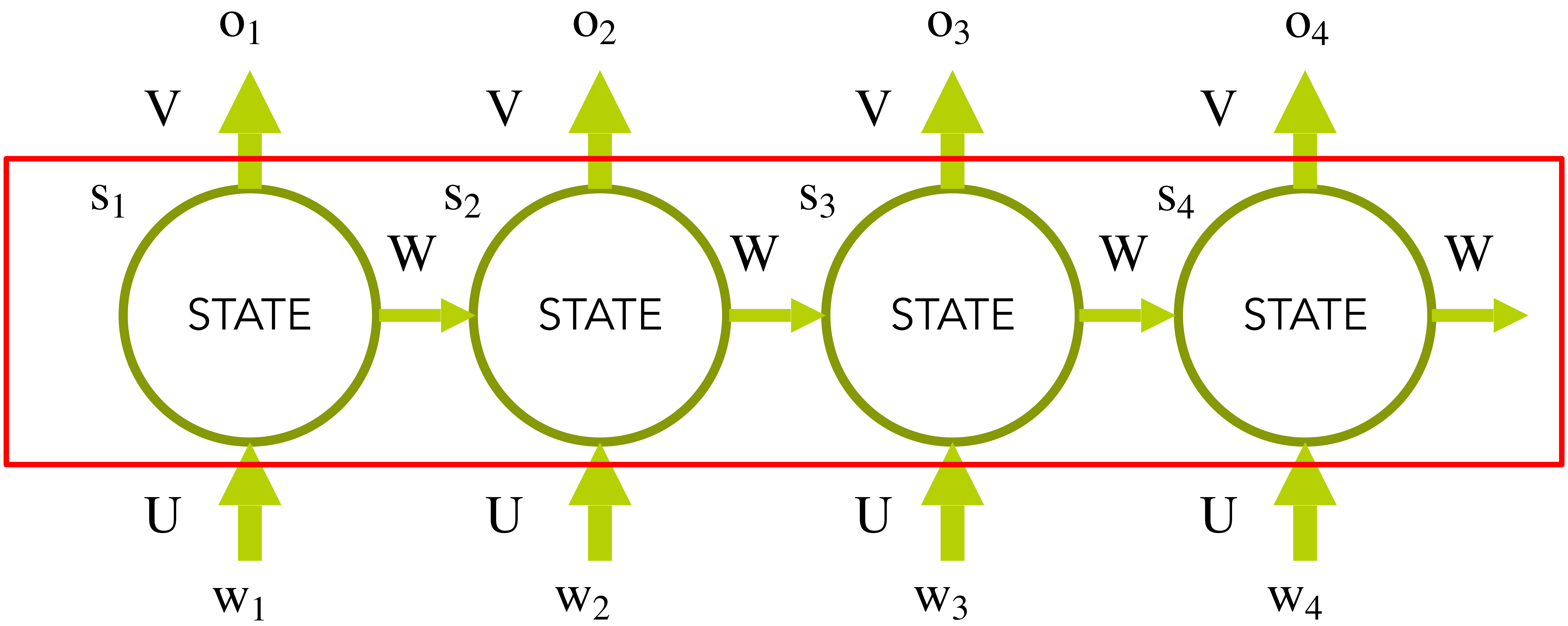
“Unrolling” the RNN

Keras calls this part the “kernel” (e.g. `kernel_initializer`,…)



"Unrolling" the RNN

Keras calls this part "recurrent" (`recurrent_initializer,...`)

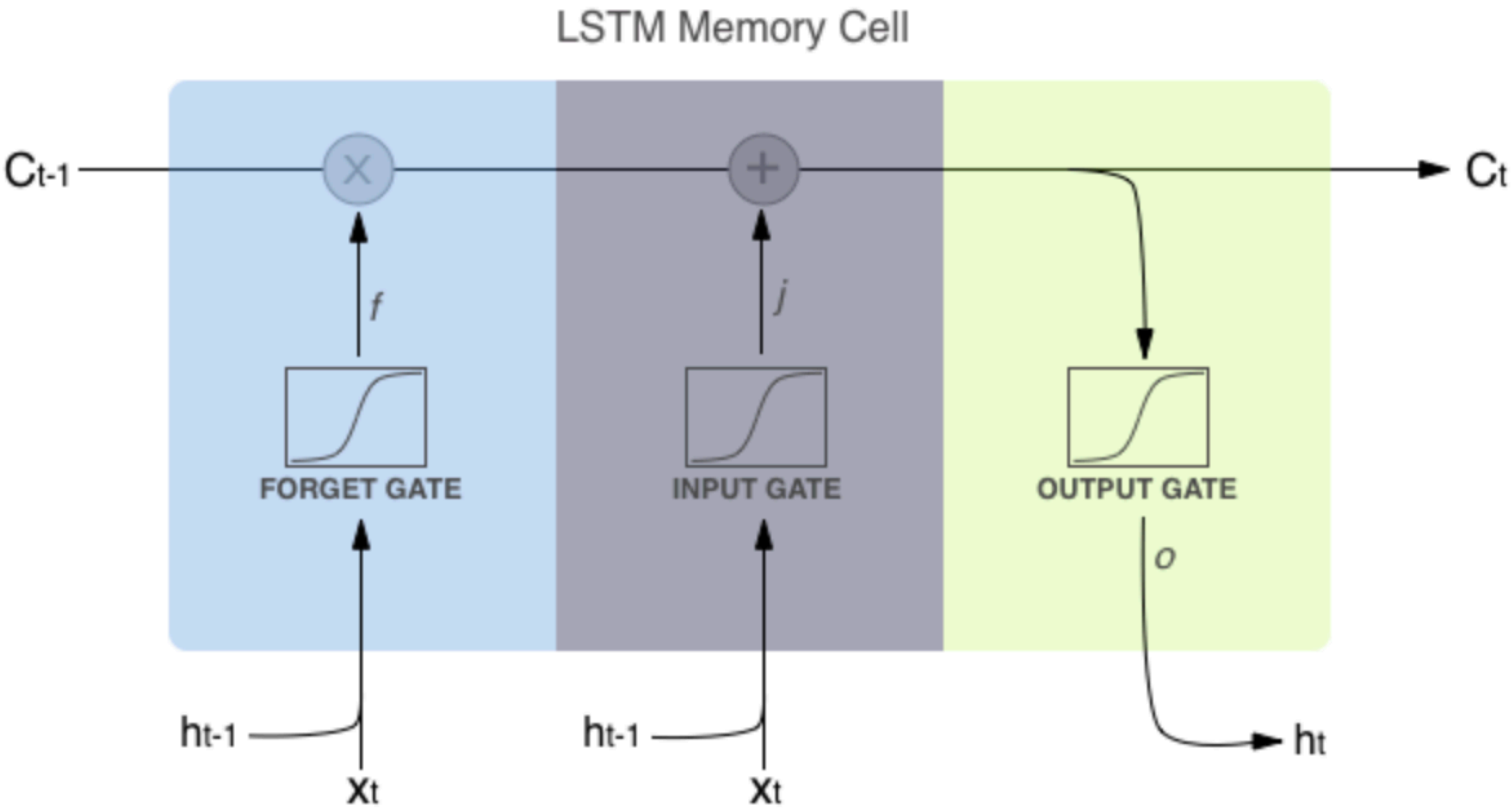


RNNs with LSTM

- Problem: **state transitions**
  - *It is hard to keep info from the distant past in current memory (without reinforcement)*

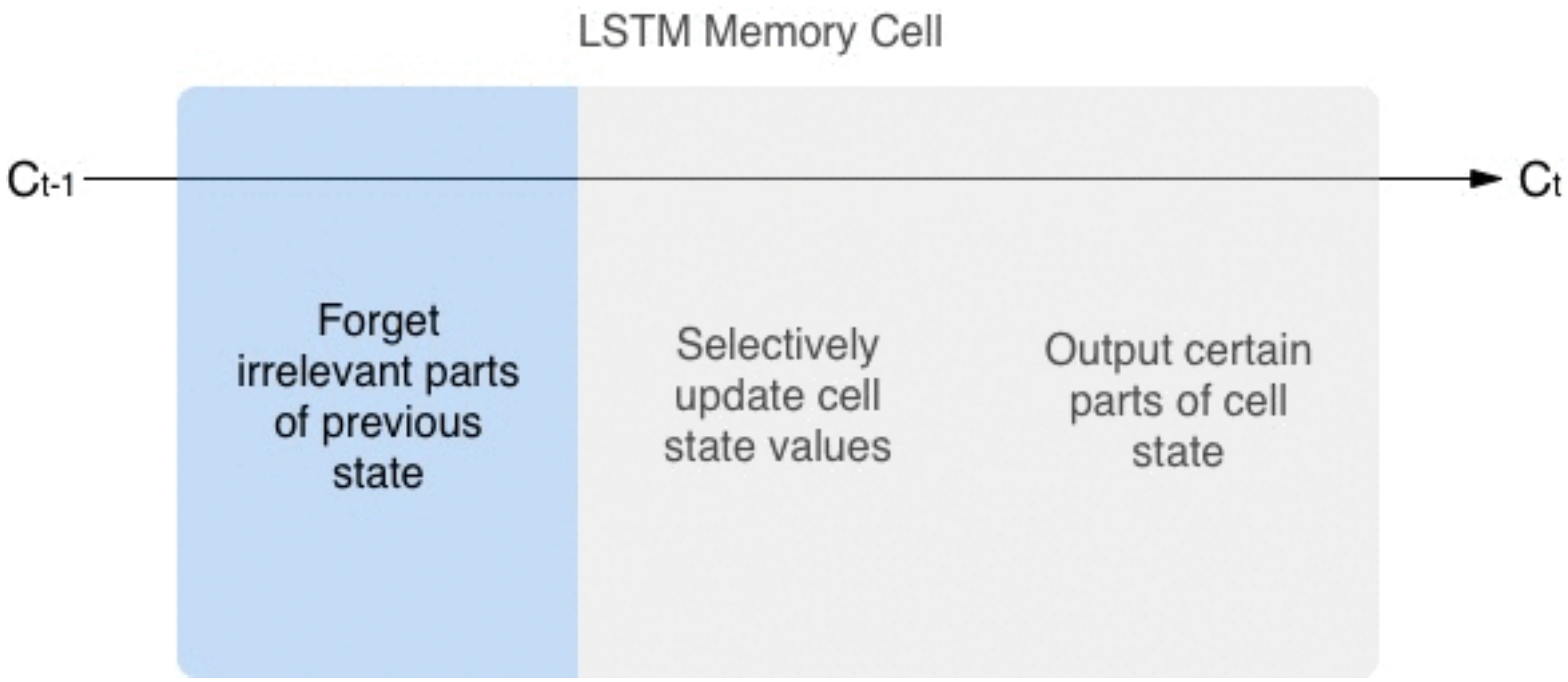
- Solution: **long short-term memory**

LSTM

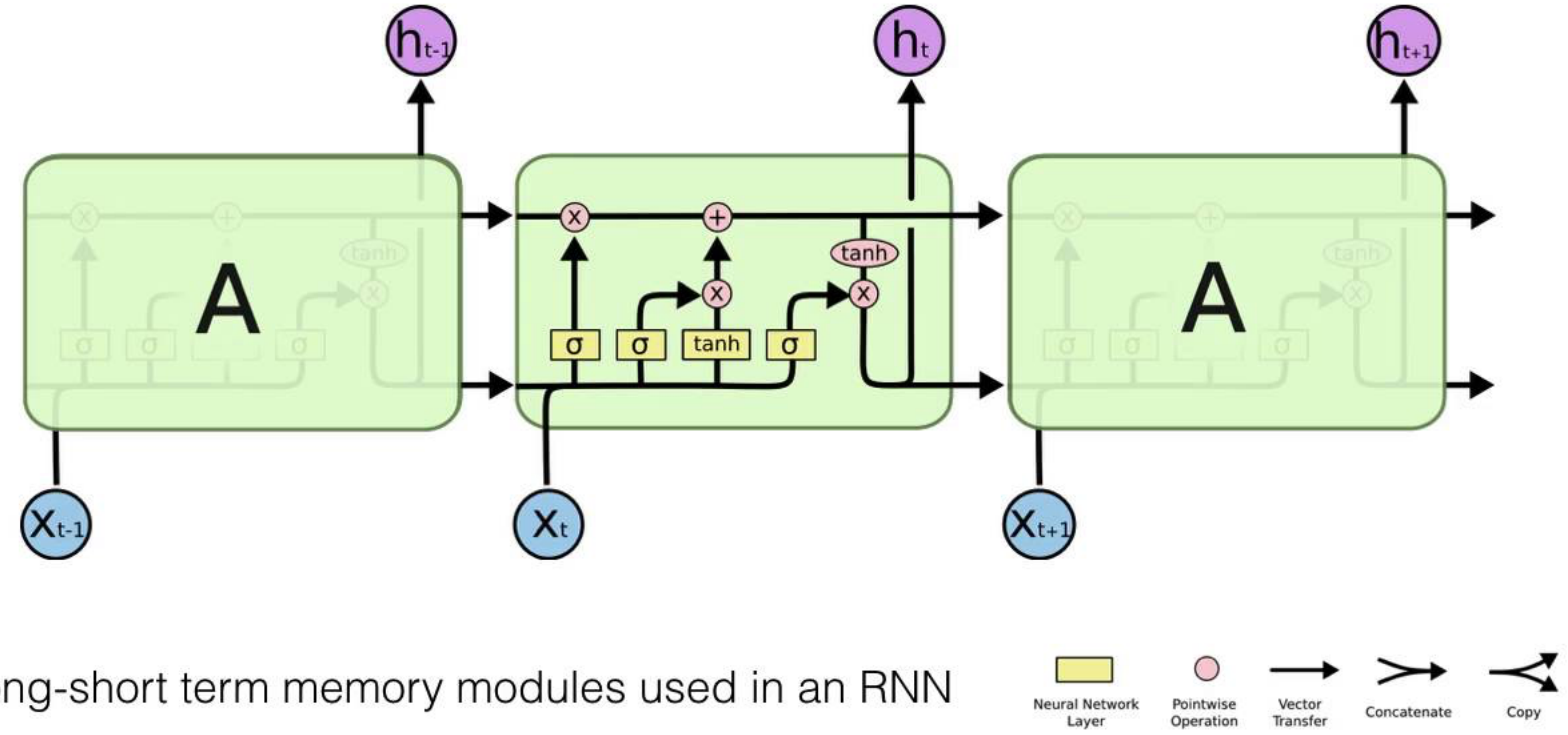


http://harinisuresh.com/2016/10/09/lstms/

LSTM


LSTM Memory Cell
C_{t-1}
C_t
Forget irrelevant parts of previous state
Selectively update cell state values
Output certain parts of cell state

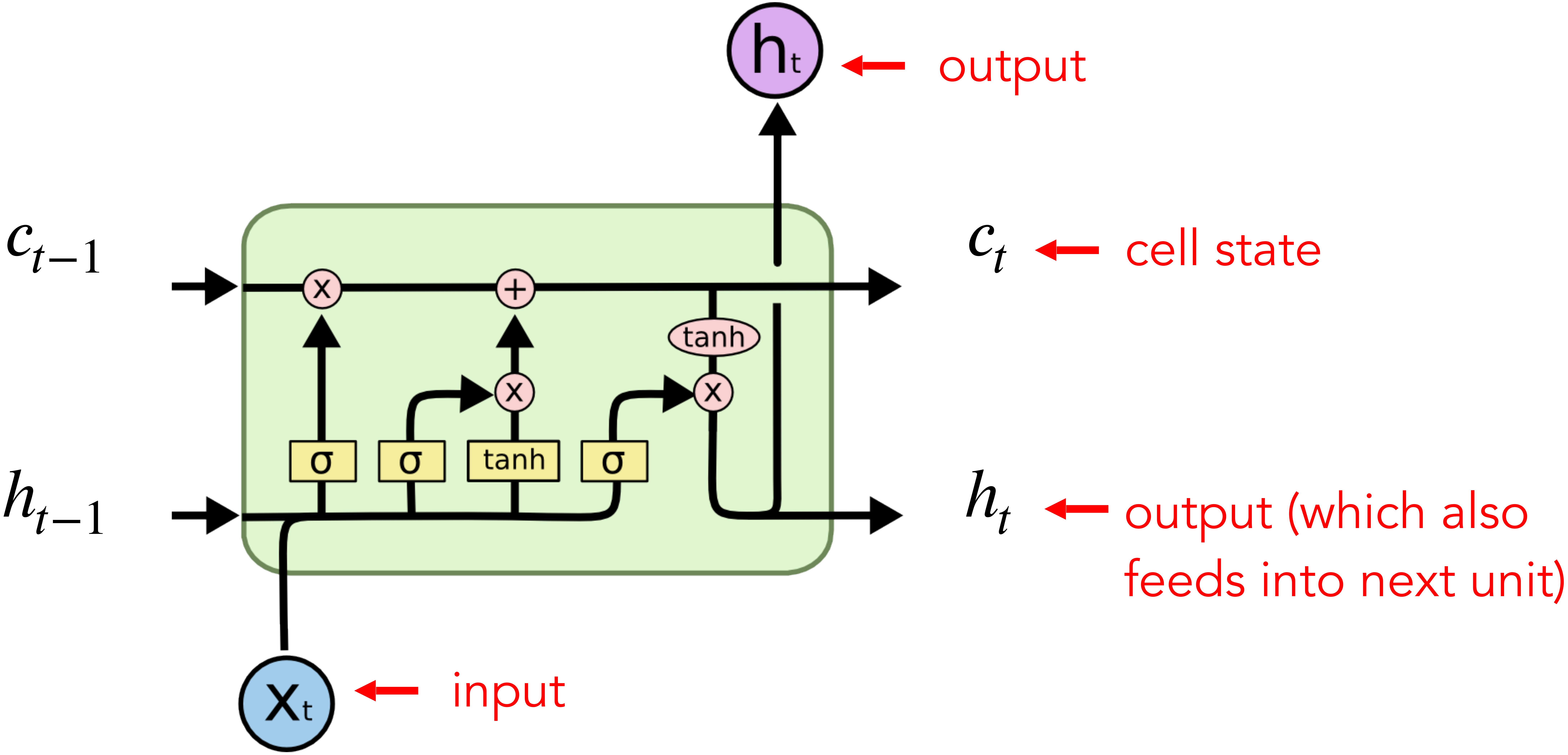
LSTM



long-short term memory modules used in an RNN

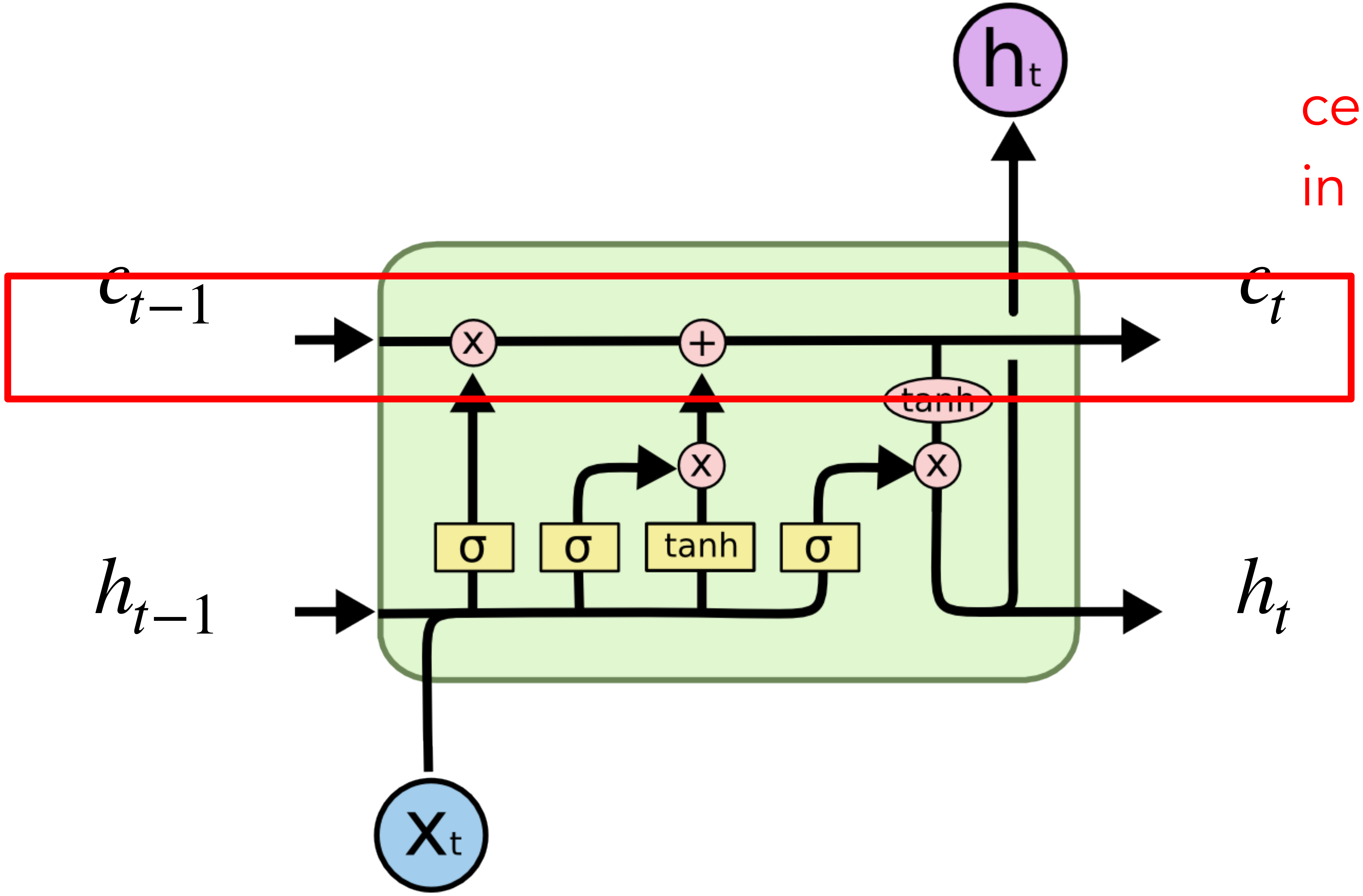
http://colah.github.io/posts/2015-08-Understanding-LSTMs/ Eugenio Culurciello © 2016

LSTM diagram



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

LSTM diagram

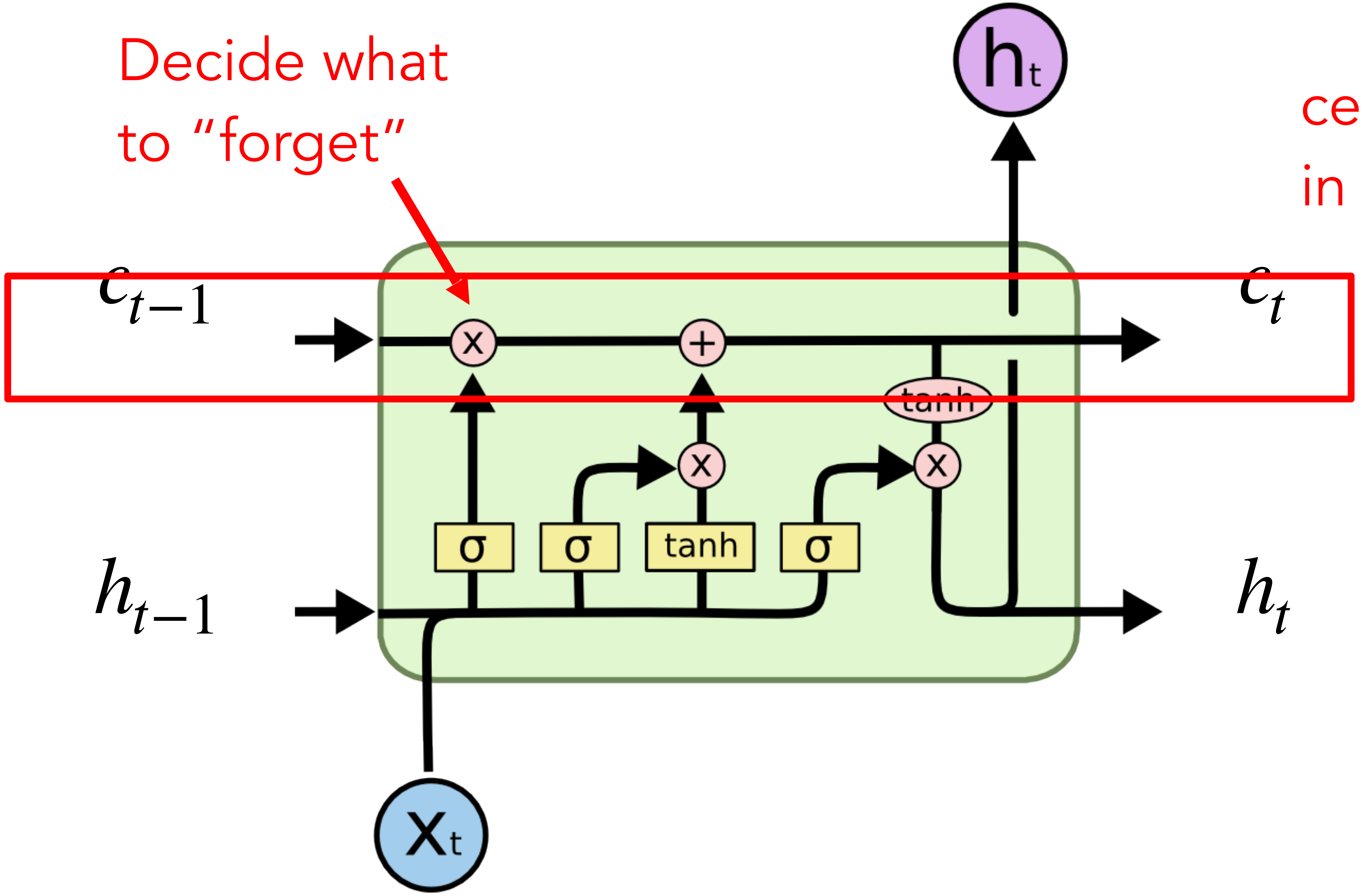


cell state gets updated in two stages

LSTM diagram

Decide what to "forget"

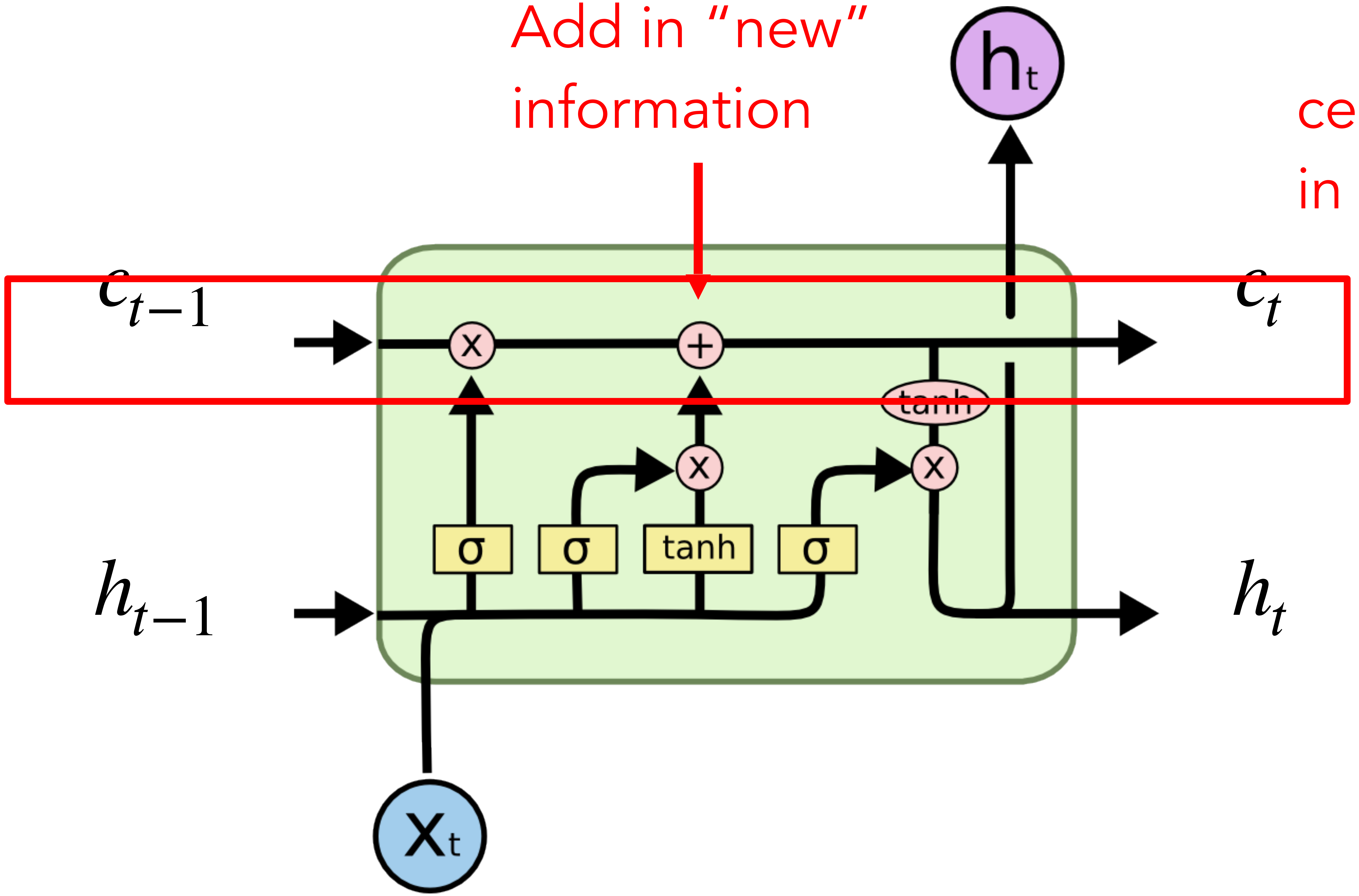
cell state gets updated in two stages



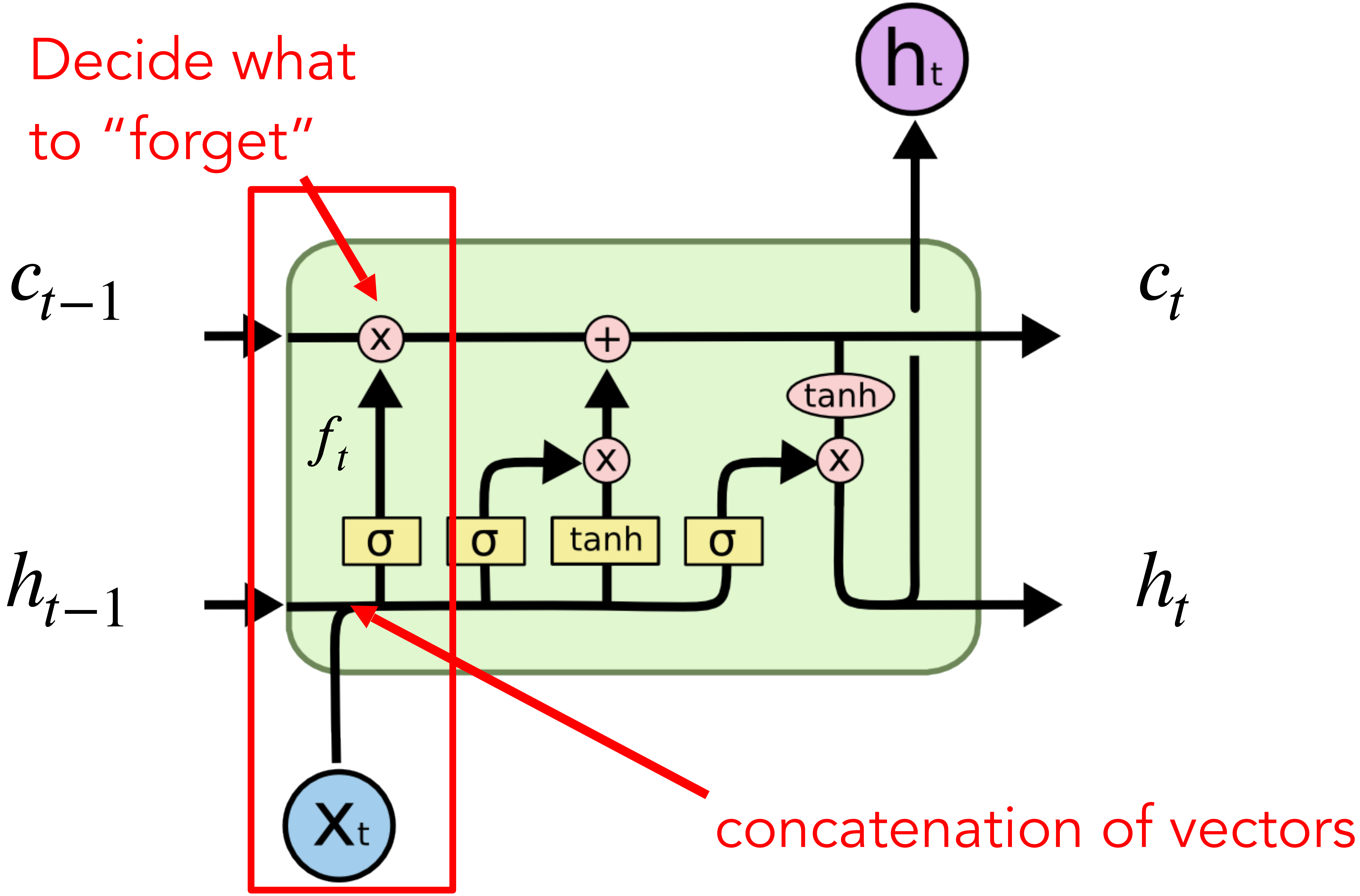
LSTM diagram

Add in "new" information

cell state gets updated in two stages



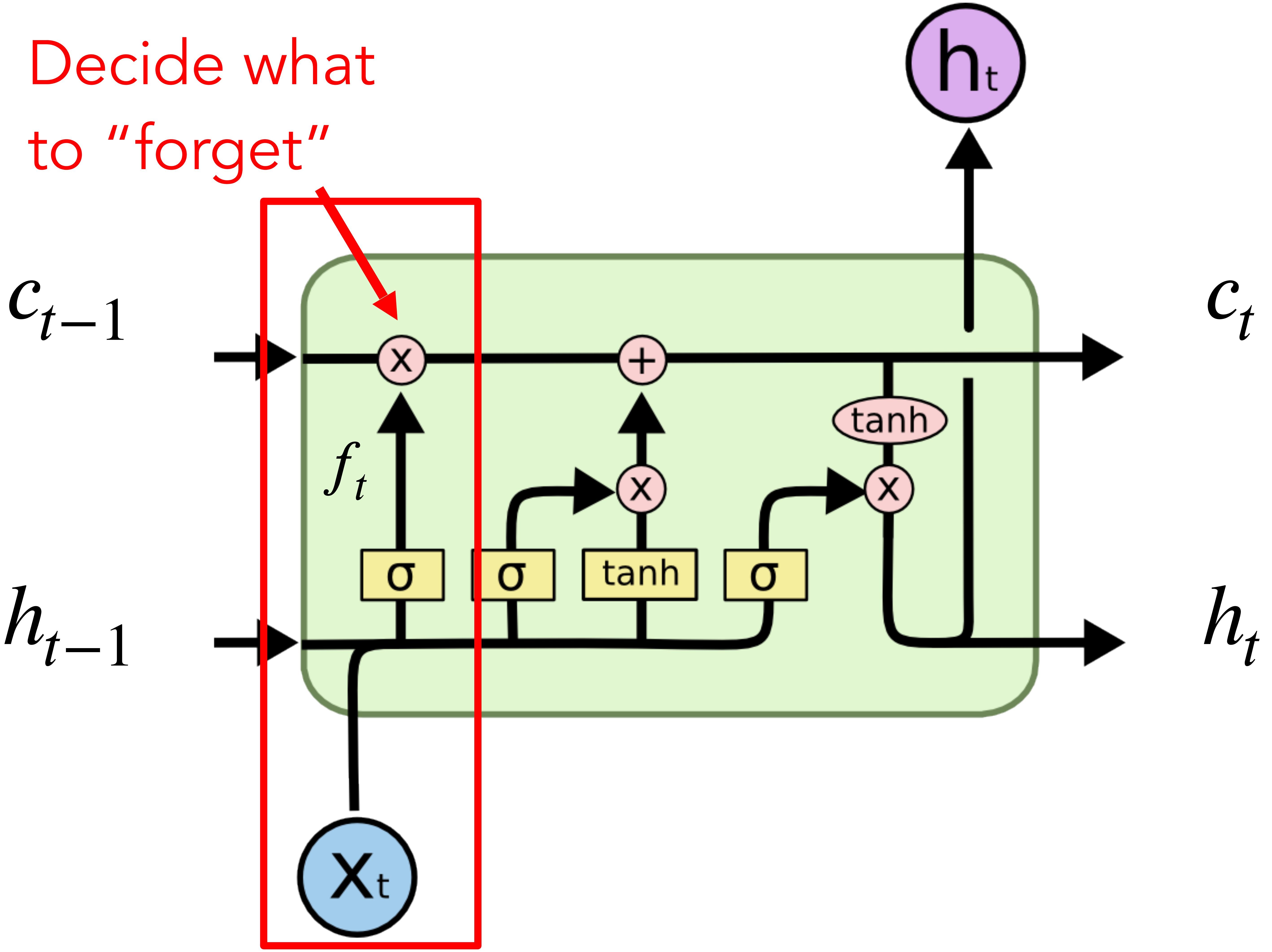
LSTM diagram



$[h_{t-1}, x_t]$

LSTM diagram

Decide what to "forget"

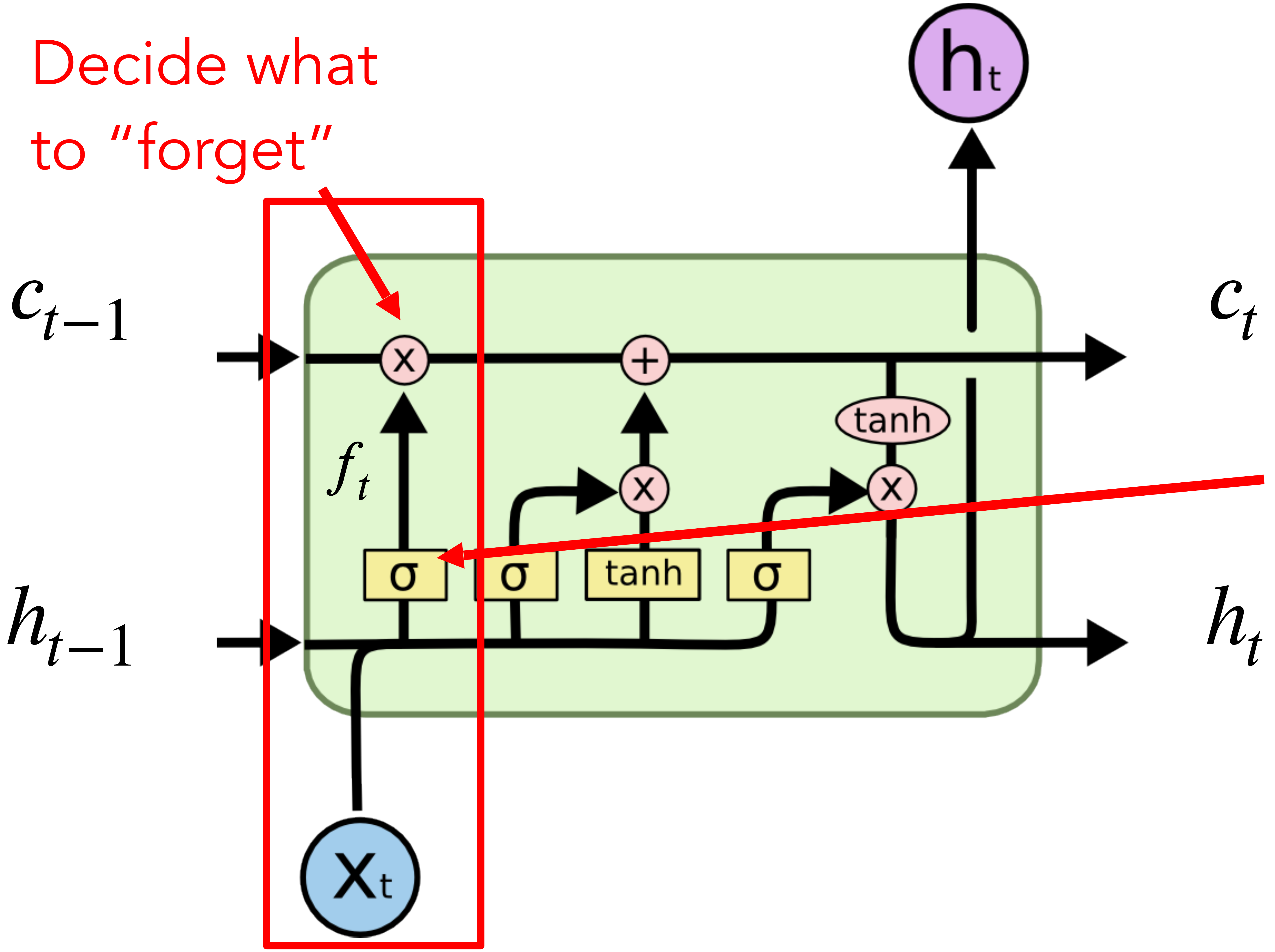


$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right)$$

based on previous output and current input

LSTM diagram

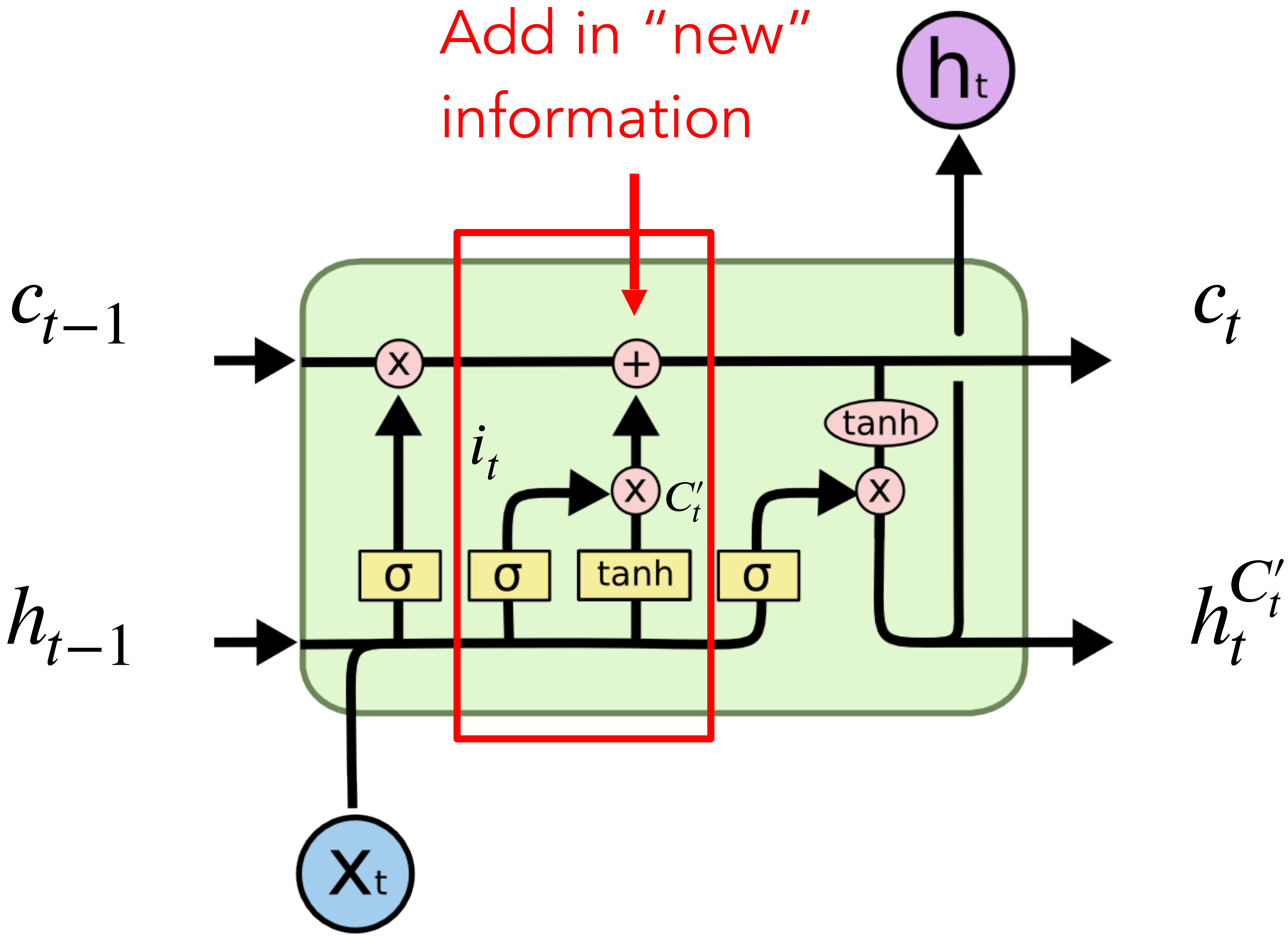
Decide what to "forget"



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

based on previous output and current input

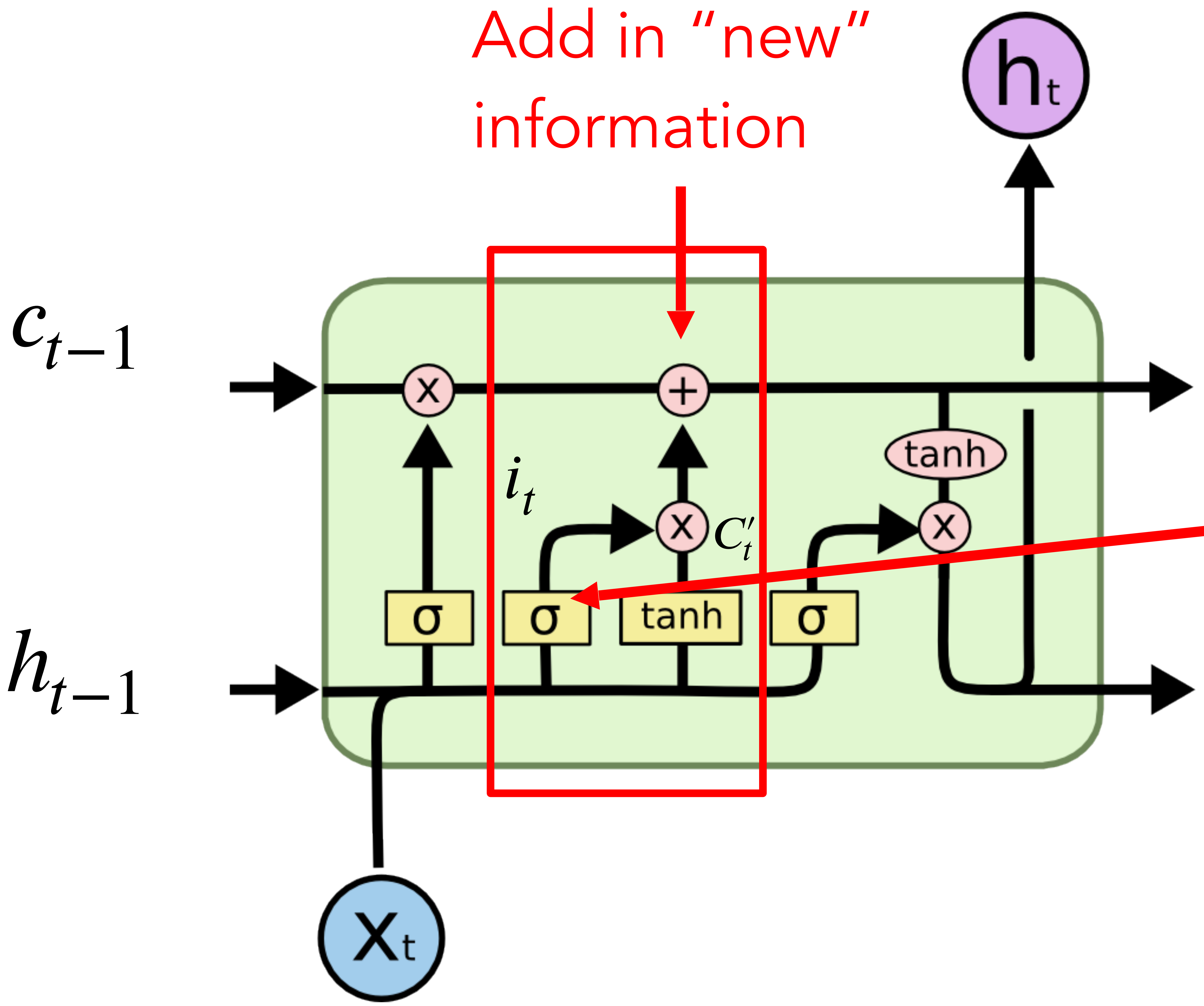
LSTM diagram



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

$$C'_t = tanh\left(W_C\left[h_{t-1}, x_t\right] + b_C\right)$$

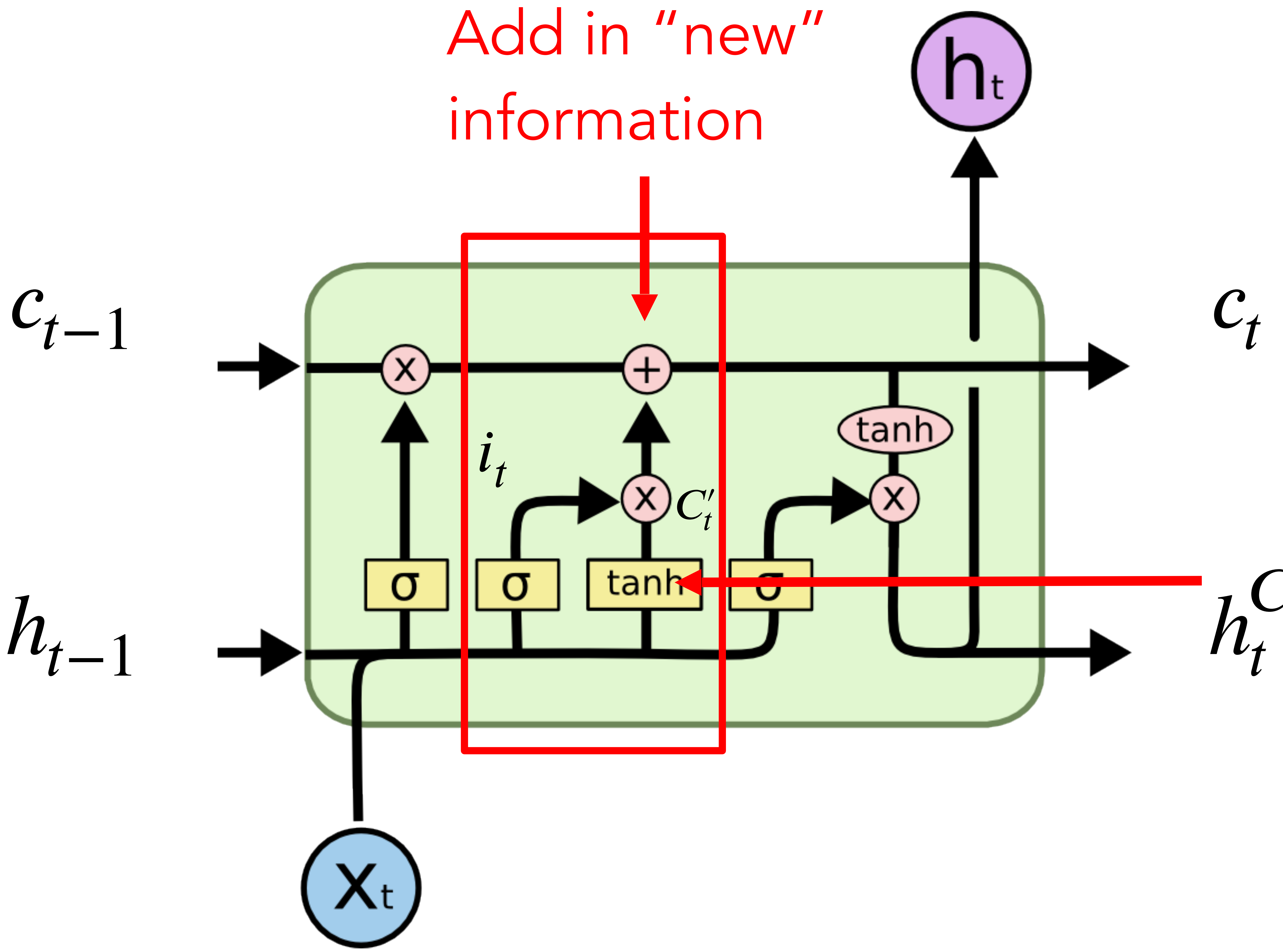
LSTM diagram



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

$$C'_t = tanh\left(W_C\left[h_{t-1}, x_t\right] + b_C\right)$$

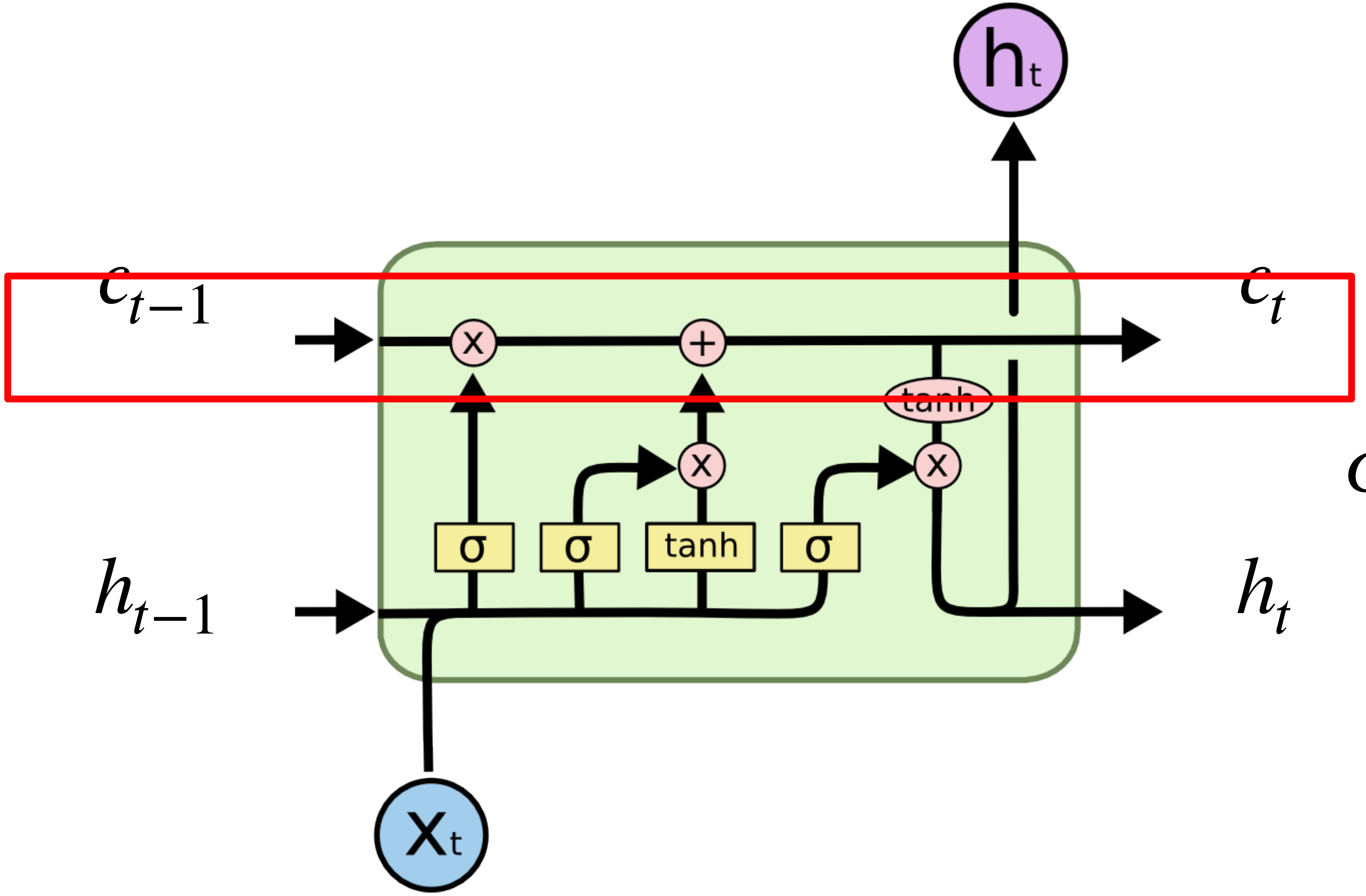
LSTM diagram



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

$$C'_t = tanh\left(W_C\left[h_{t-1}, x_t\right] + b_C\right)$$

LSTM diagram



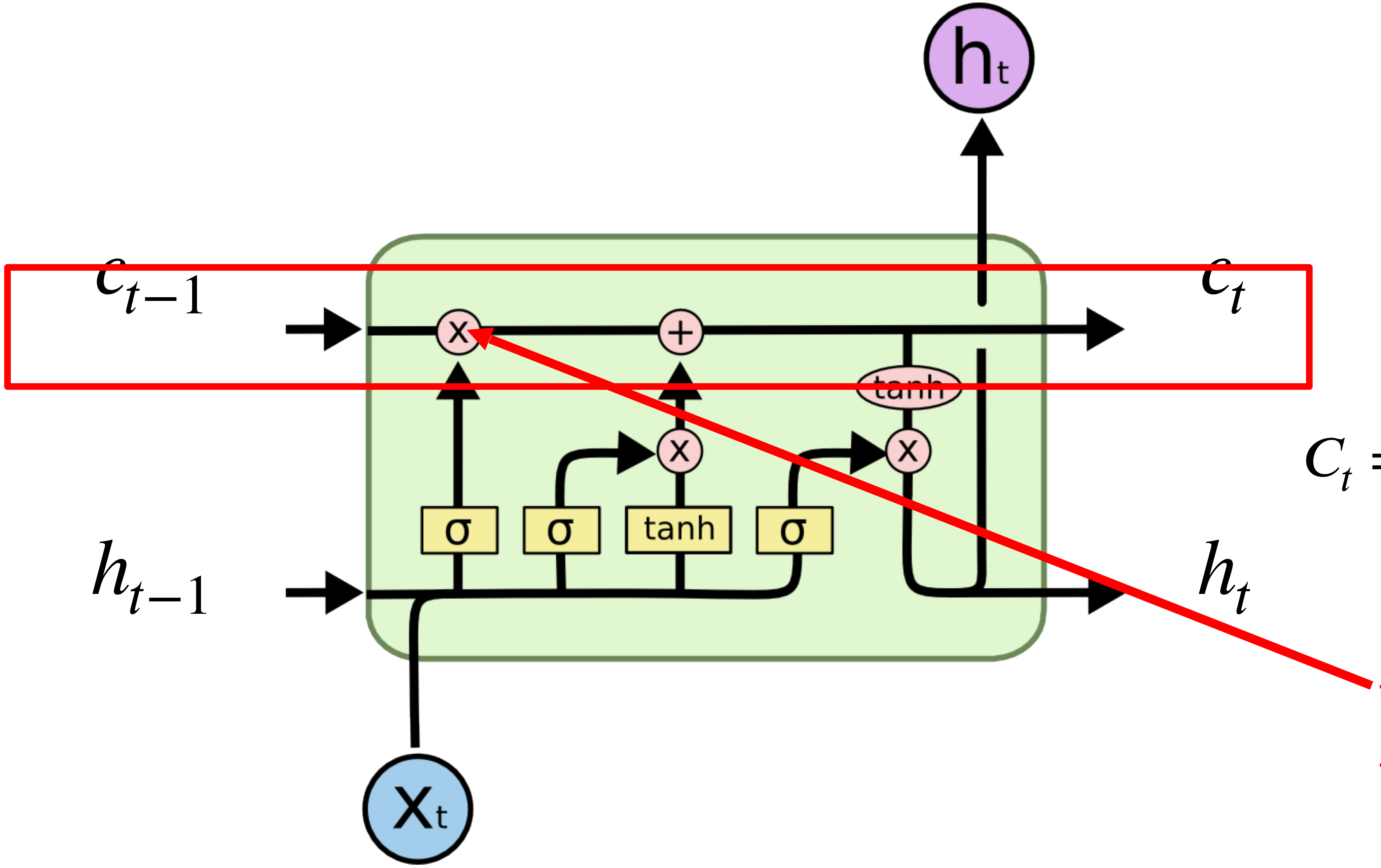
Note: 'x' represents element-wise multiplication

$$C_t = f_i * C_{t-1} + i_t * C'_t$$

forget the old (or not)

add the new (or not)

LSTM diagram



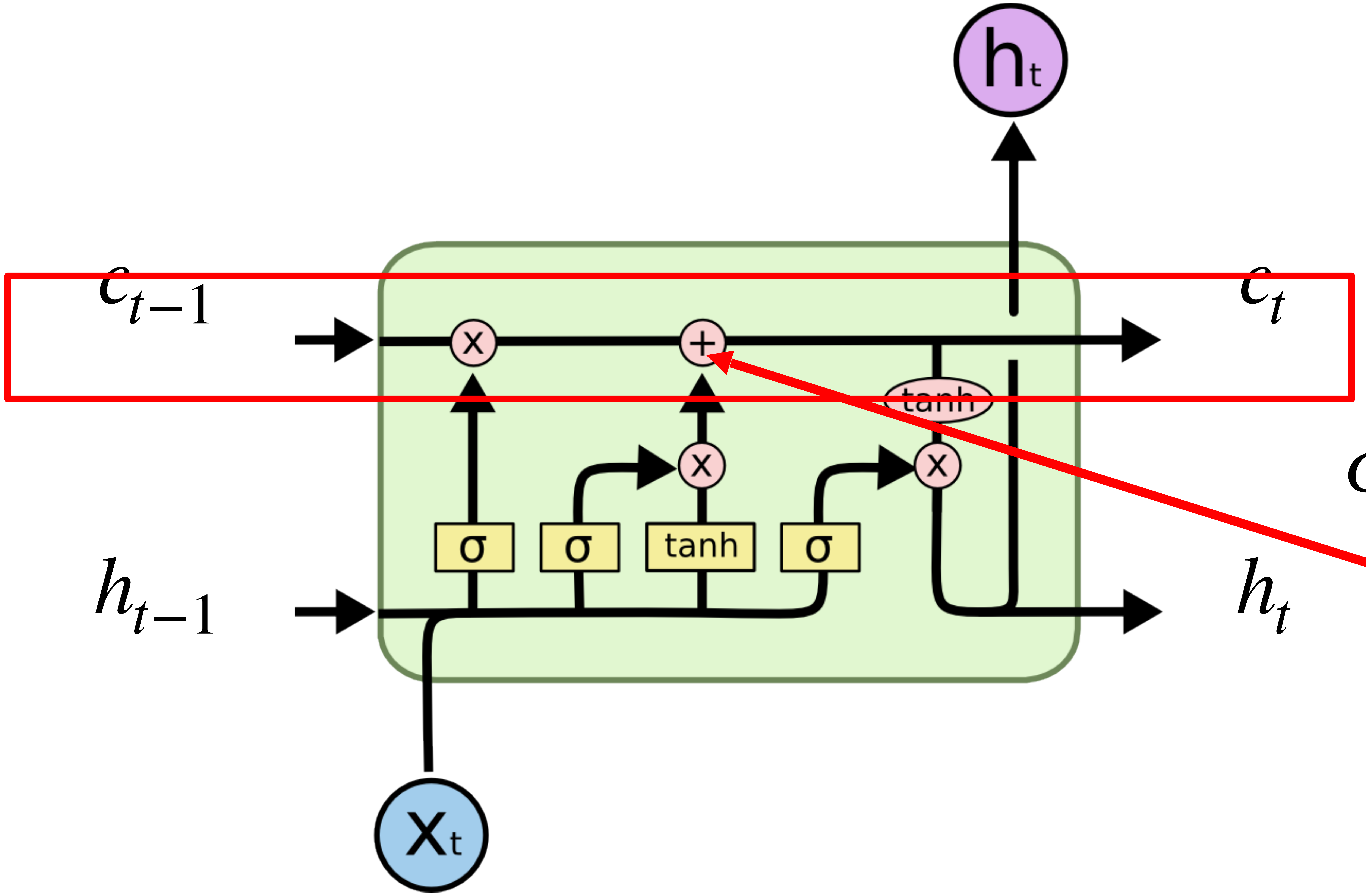
Note: 'x' represents element-wise multiplication

$$C_t = f_i * C_{t-1} + i_t * C'_t$$

forget the old (or not)

add the new (or not)

LSTM diagram



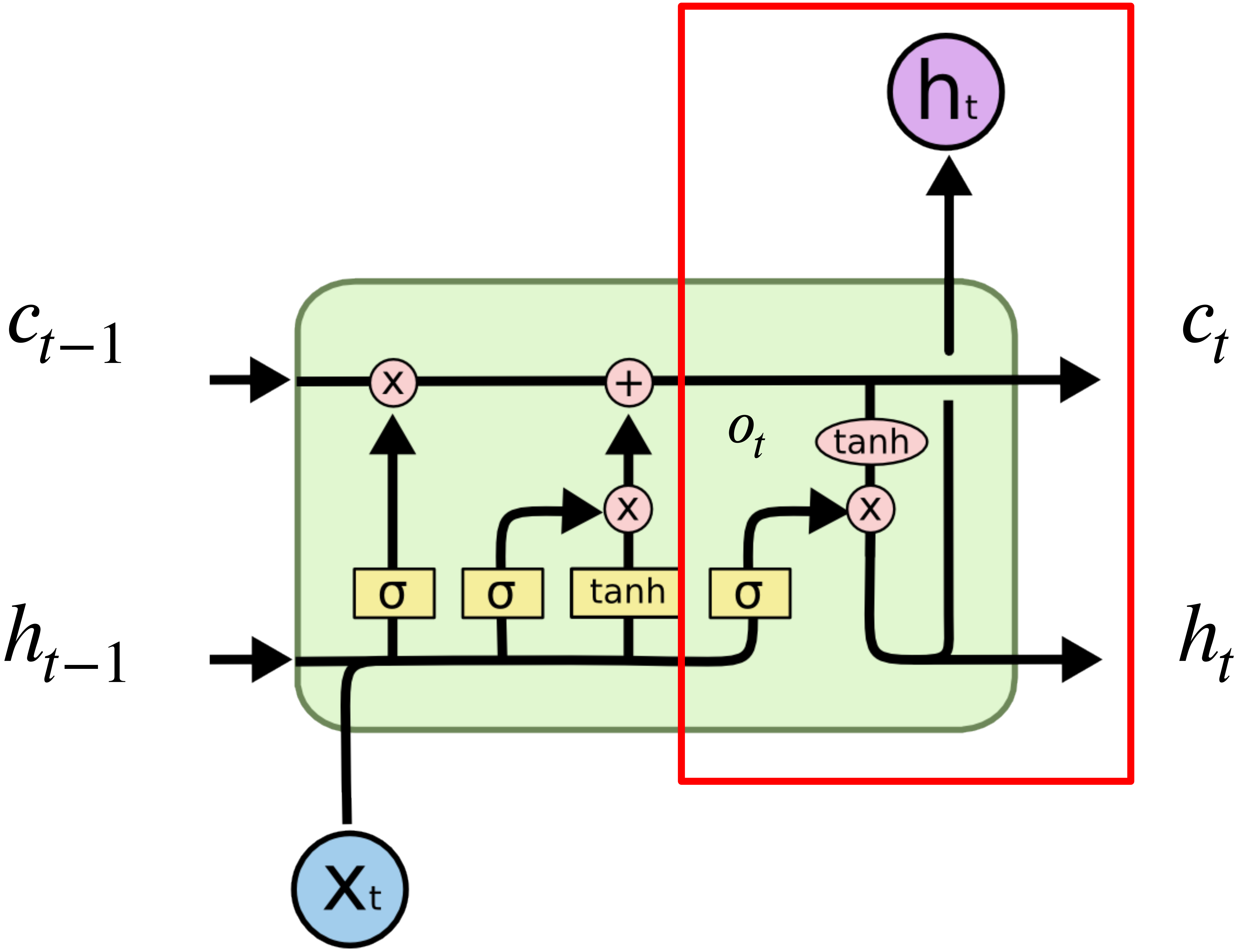
Note: '' represents element-wise multiplication

$$C_t = f_i * C_{t-1} + i_t * C_t'$$

forget the old (or not)

add the new (or not)

LSTM diagram

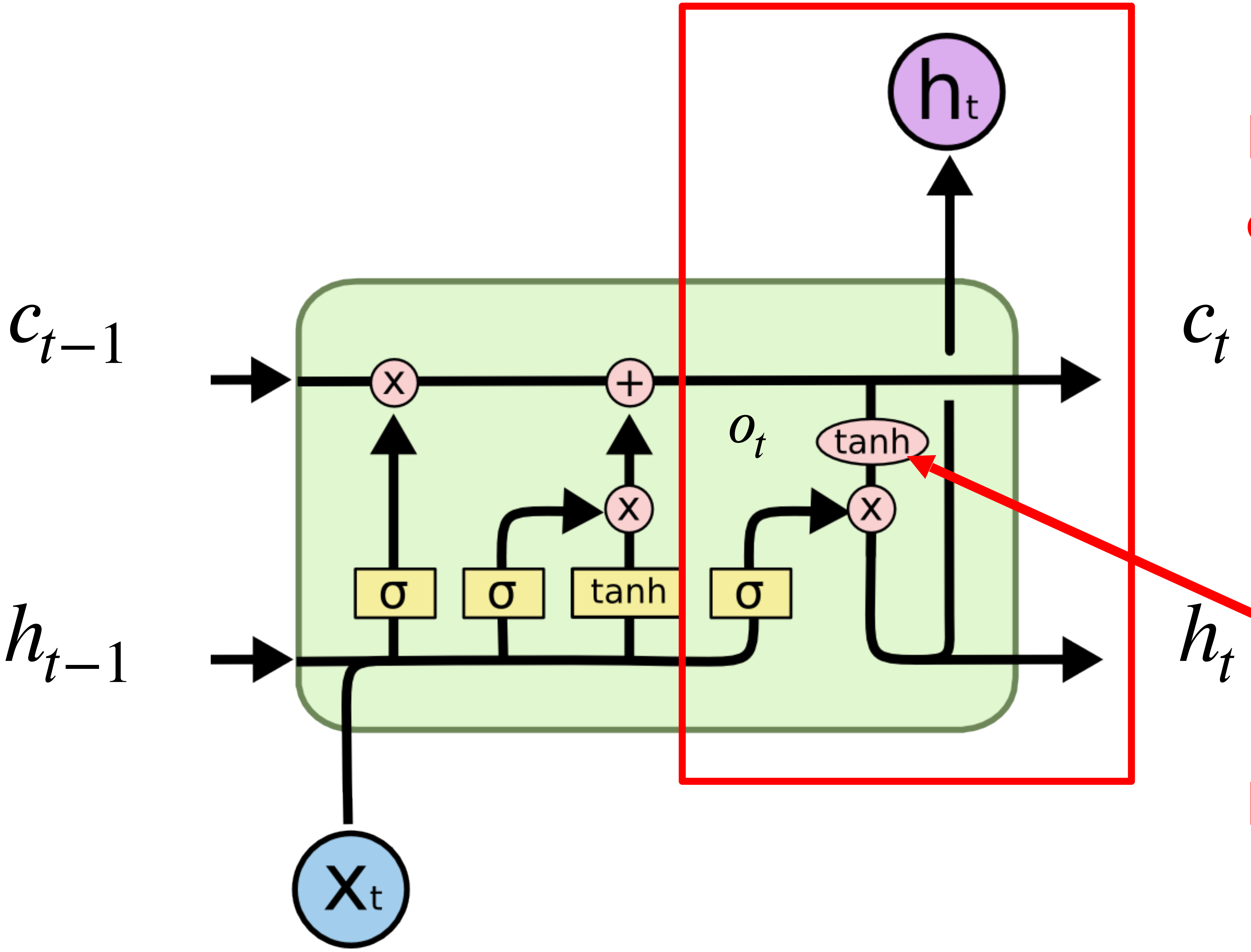


Final stage computes the output

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

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

LSTM diagram



Final stage computes the output

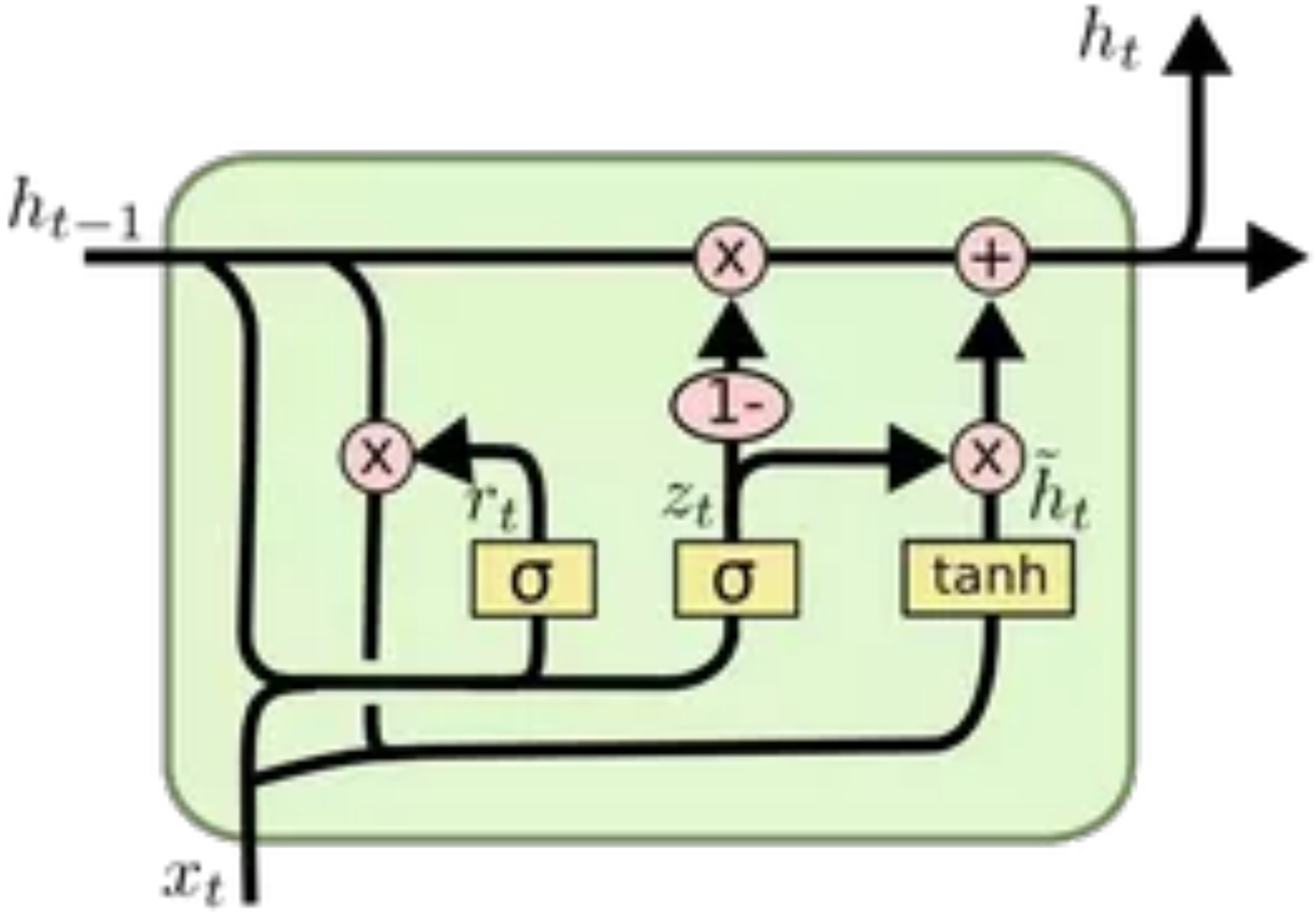
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Note: No weights here

Gated Recurrent Unit (GRU)

- Gated Recurrent Unit (GRU)



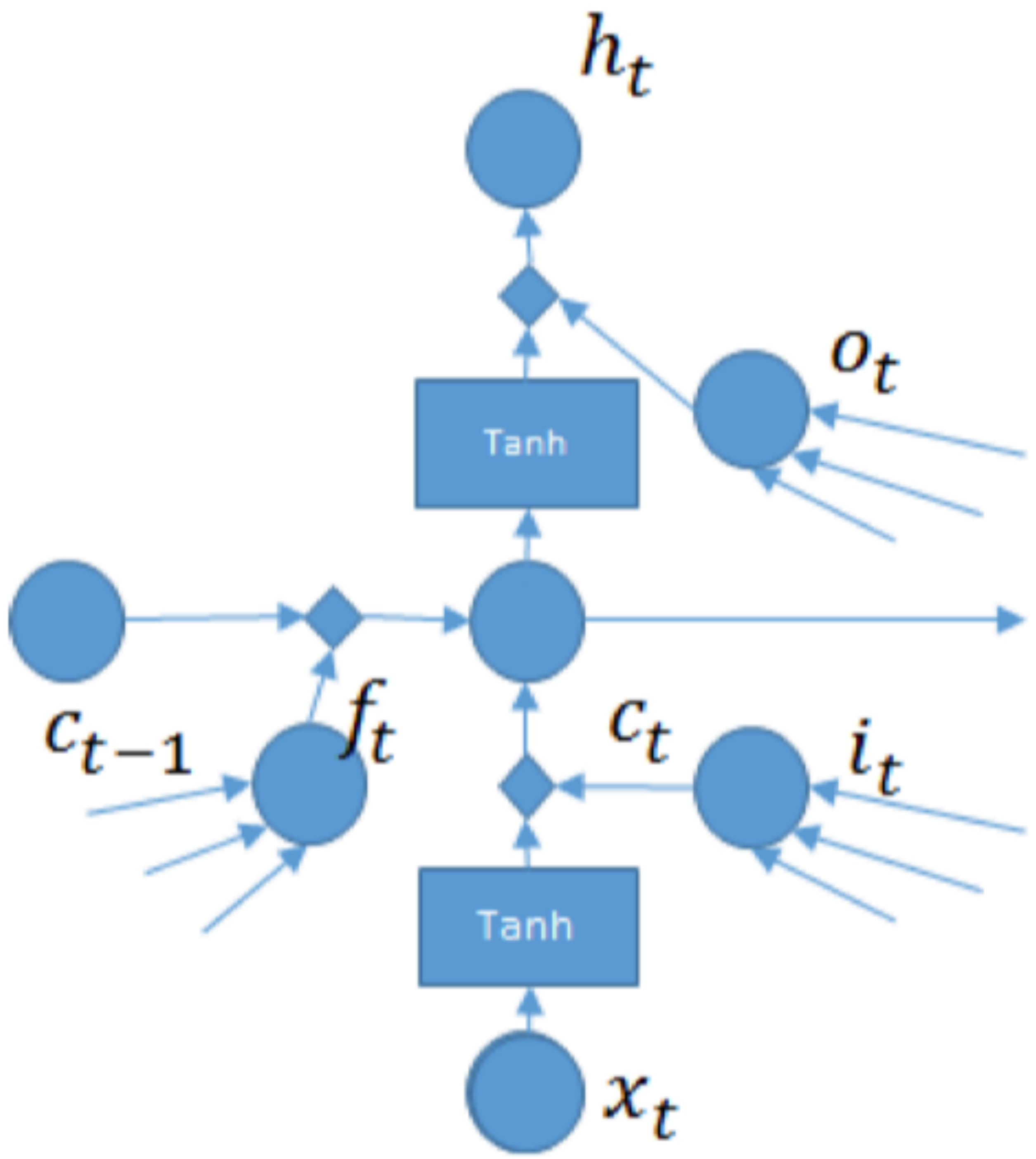
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

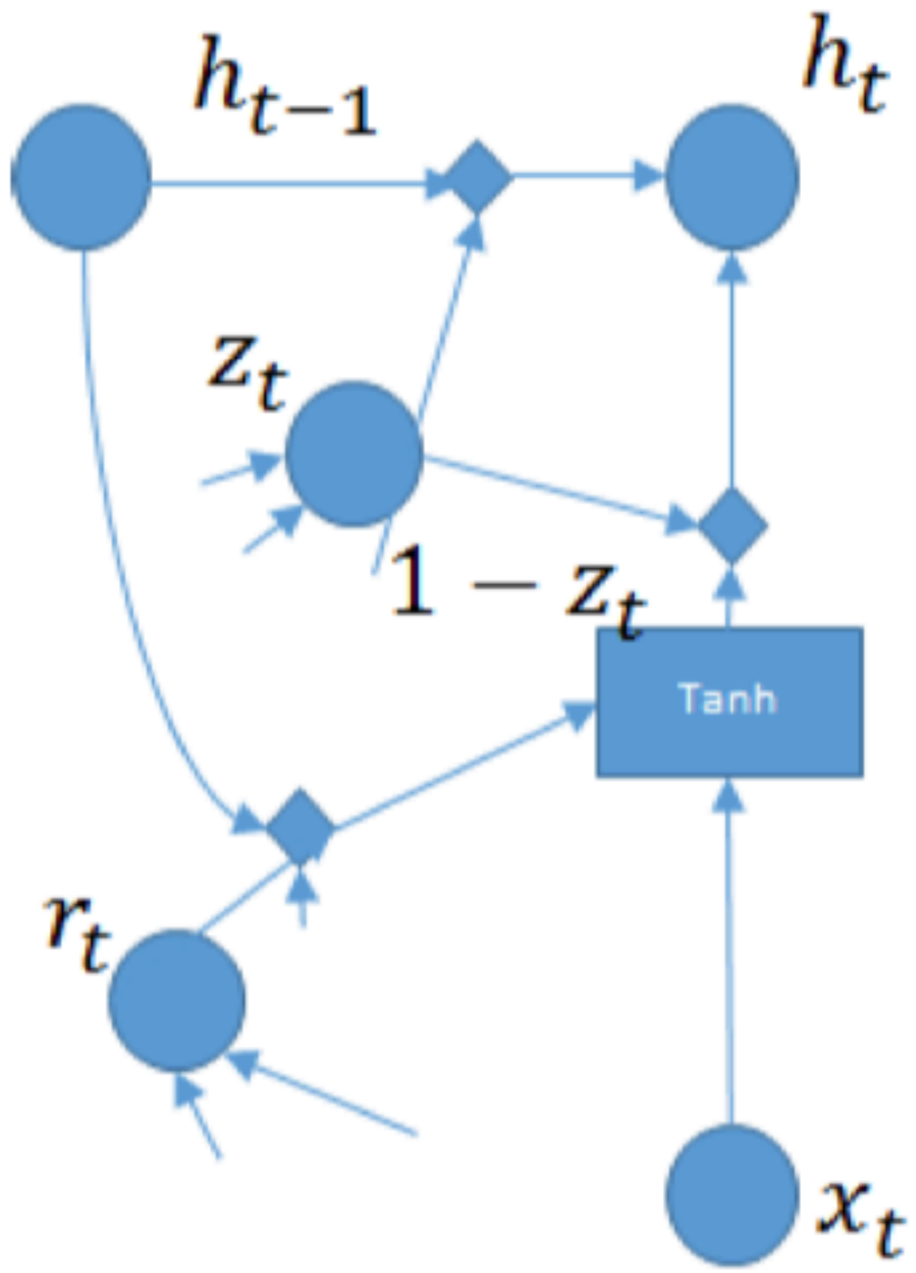
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

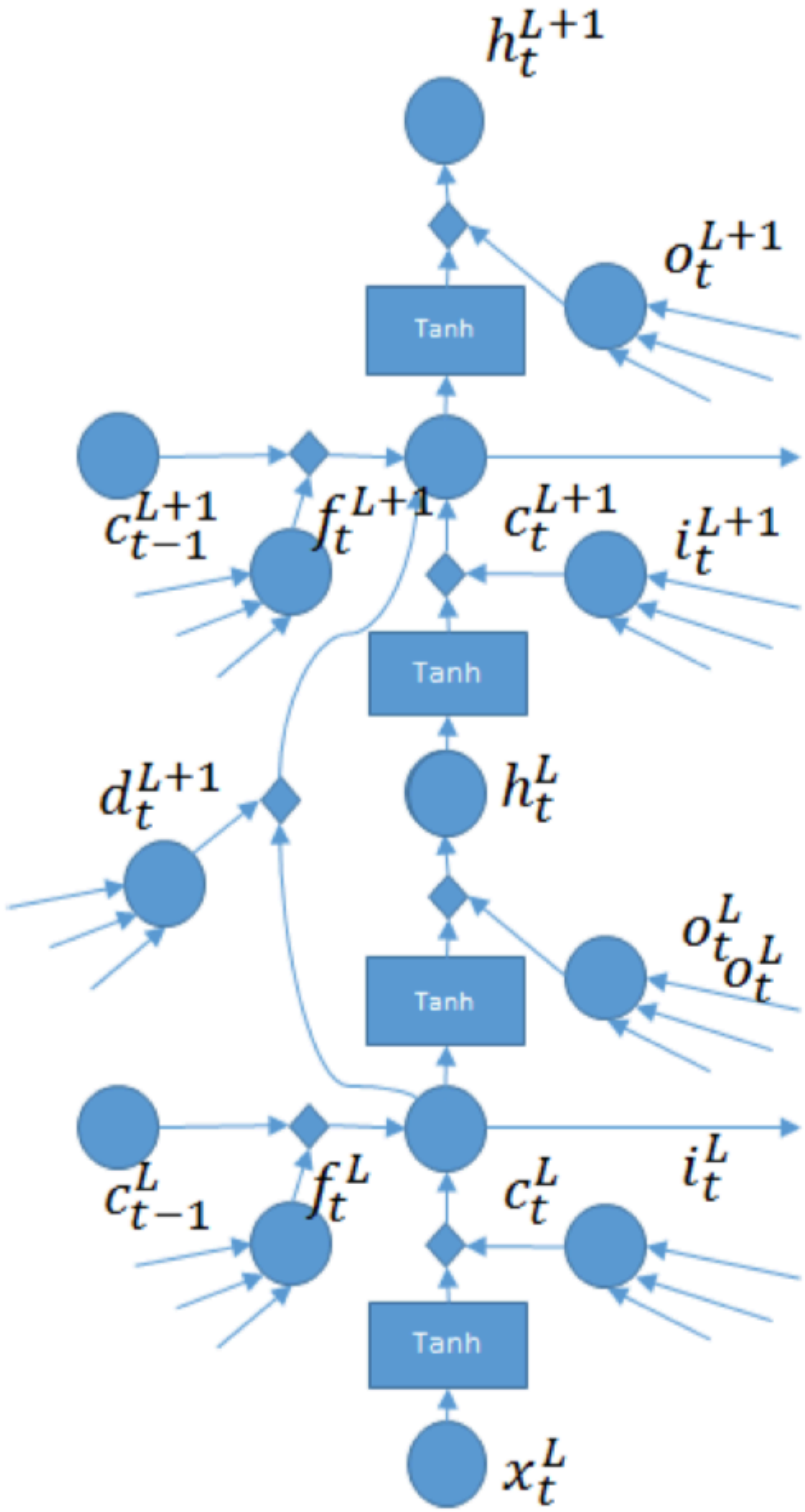
Variations of LSTM



LSTM



GRU

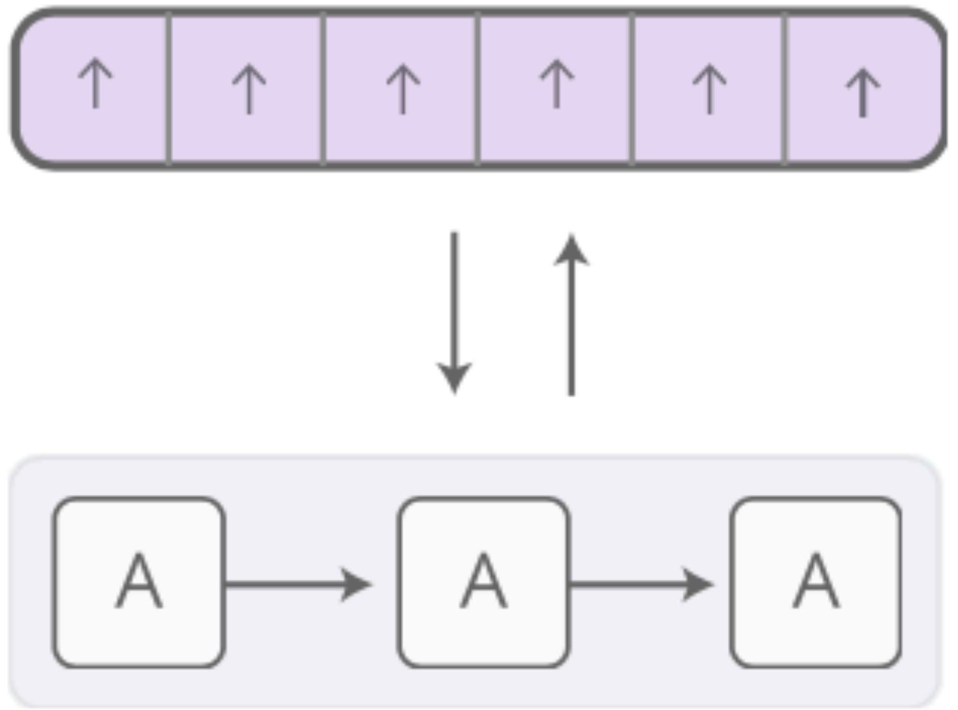


Depth-Gabed LSTM

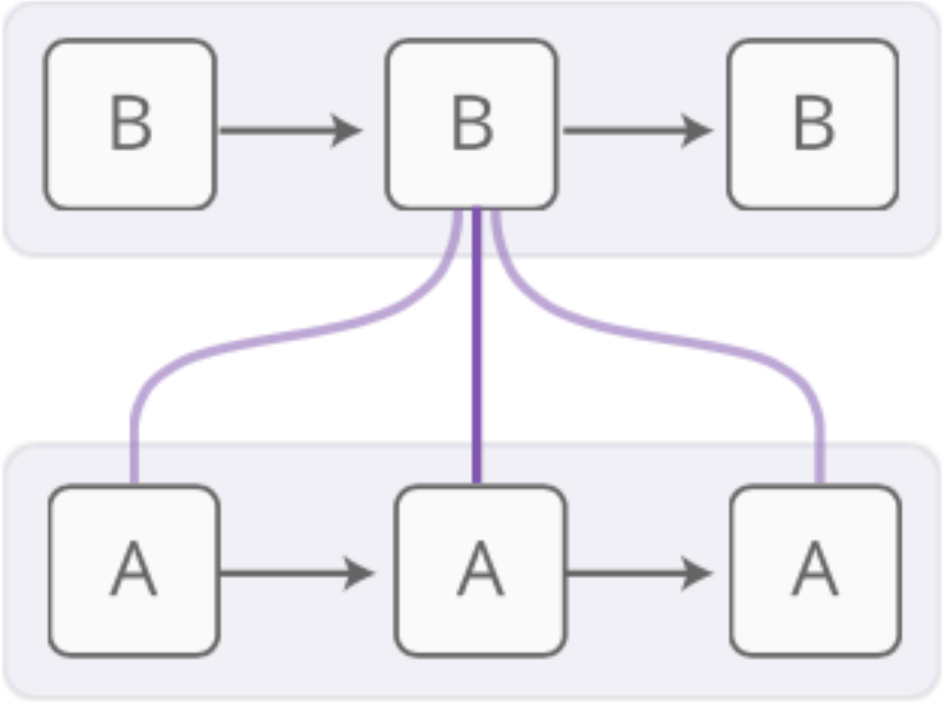
https://arxiv.org/pdf/1508.03790v2.pdf

Improving RNNs

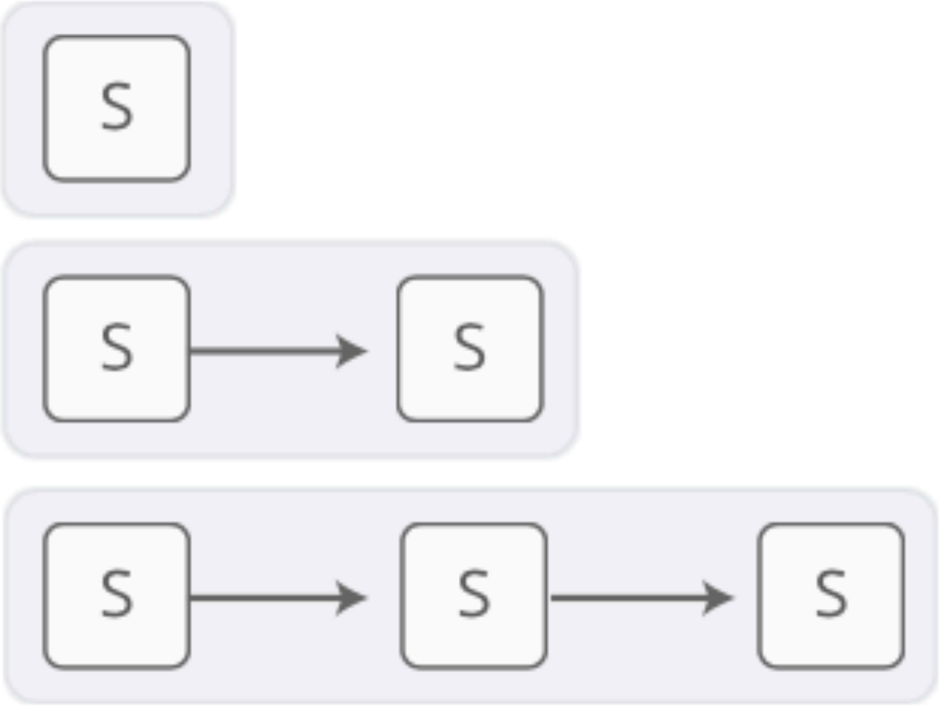
Augmenting RNNs



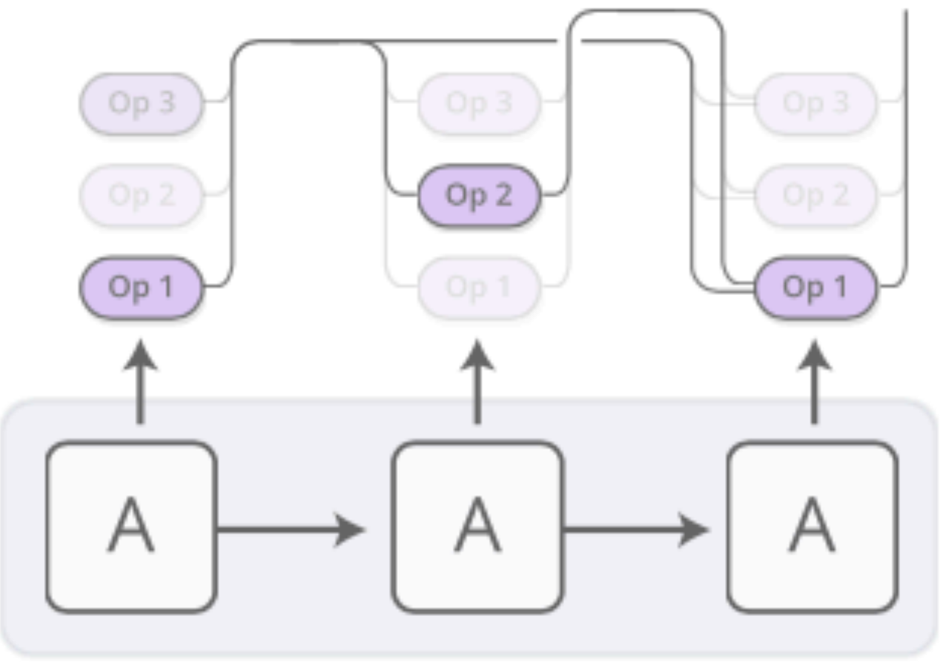
Neural Turing Machines
have external memory that they can read and write to.



Attentional Interfaces
allow RNNs to focus on parts of their input.



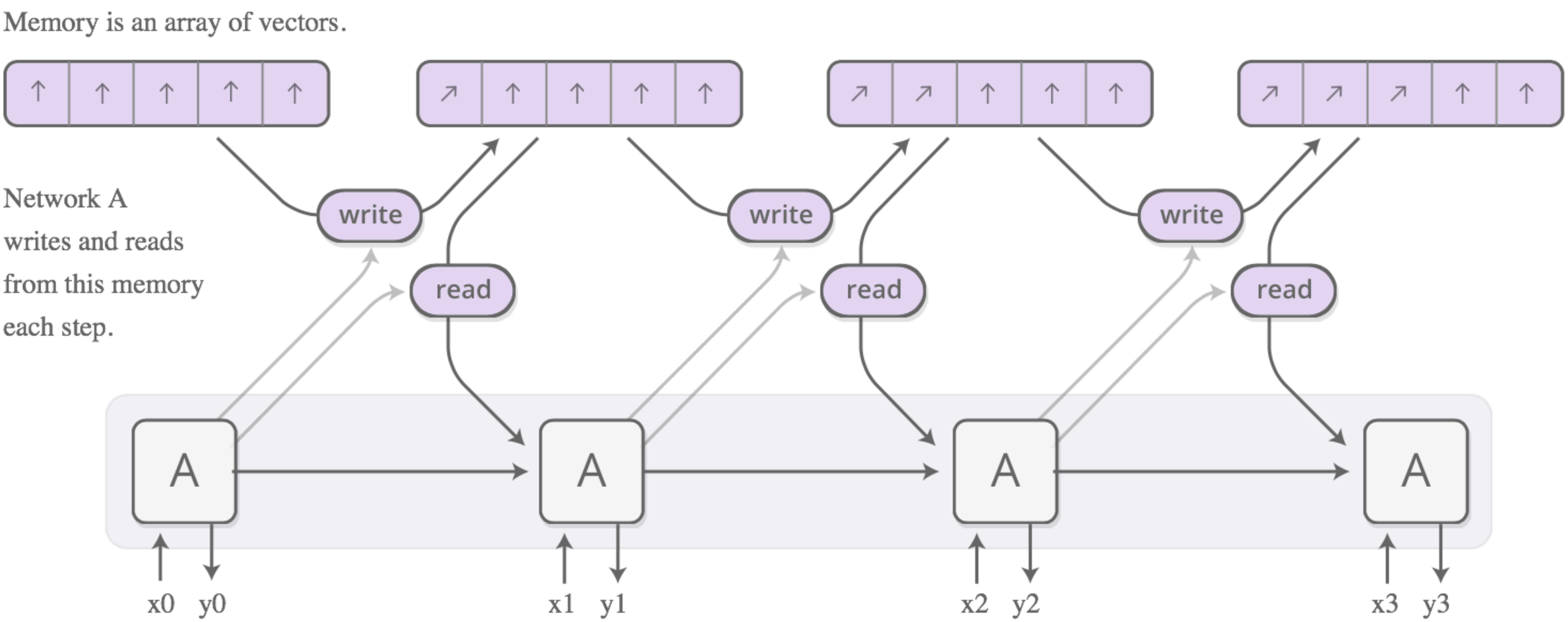
Adaptive Computation Time
allows for varying amounts of computation per step.



Neural Programmers
can call functions, building programs as they run.

https://distill.pub/2016/augmented-rnns/

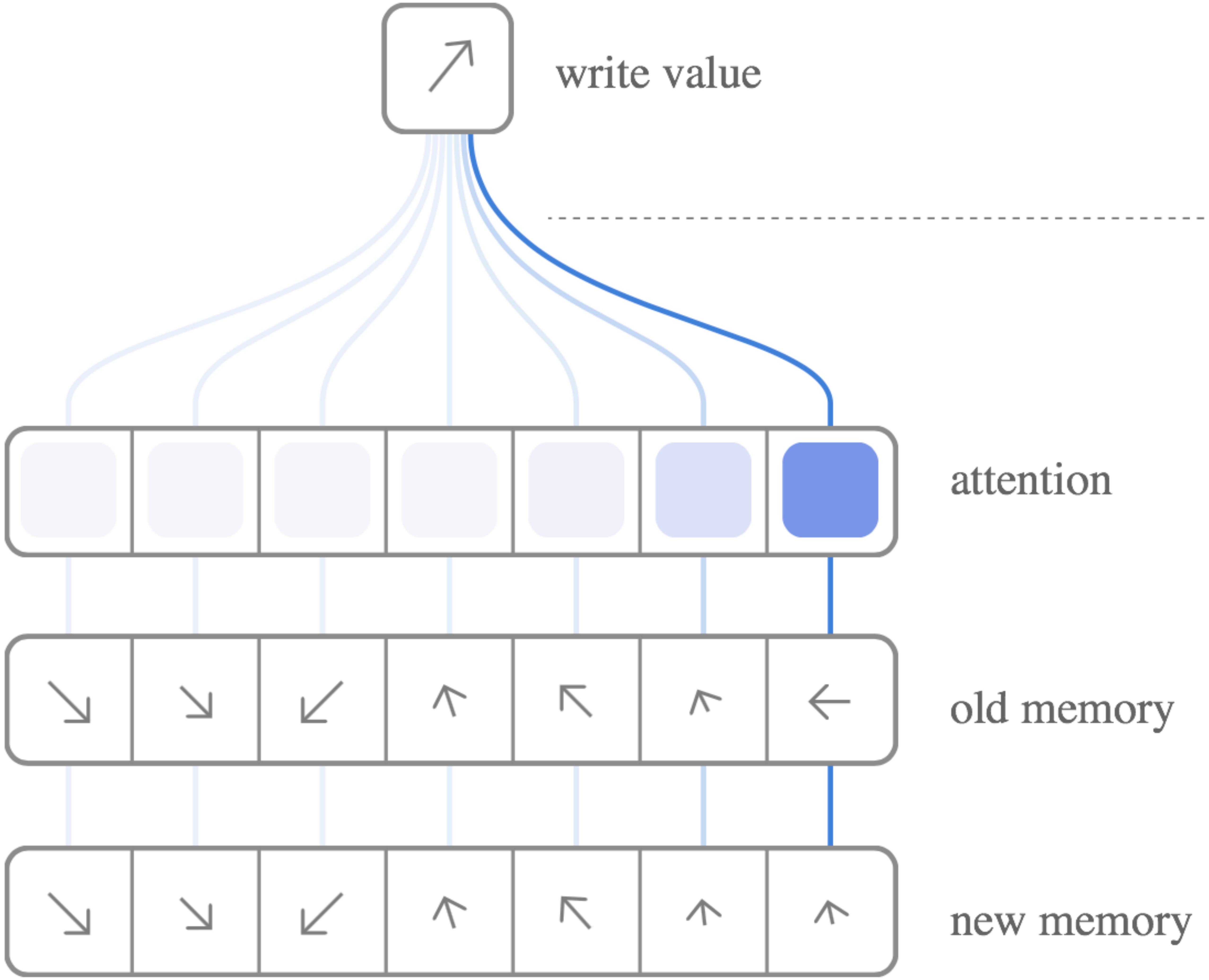
Neural Turing Machines



Graves et al, 2014

https://distill.pub/2016/augmented-rnns/

Neural Turing Machines



Instead of writing to one location, we write everywhere, just to different extents.

The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

$$M_i \leftarrow a_i w + (1 - a_i) M_i$$

[Graves et al, 2014](https://distill.pub/2016/augmented-rnns/)

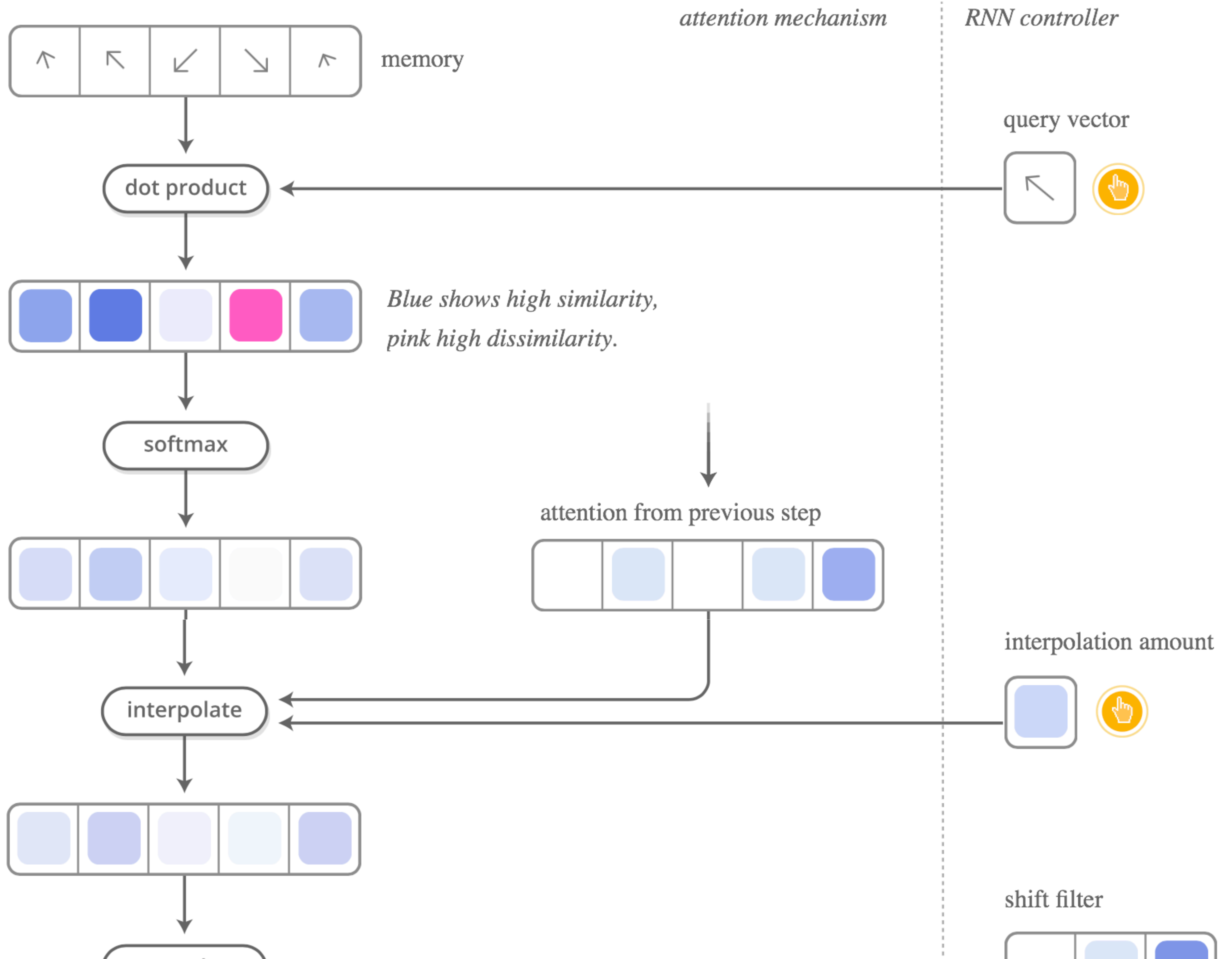
https://distill.pub/2016/augmented-rnns/

First, the controller gives a query vector and each memory entry is scored for similarity with the query.

The scores are then converted into a distribution using softmax.

Next, we interpolate the attention from the previous time step.

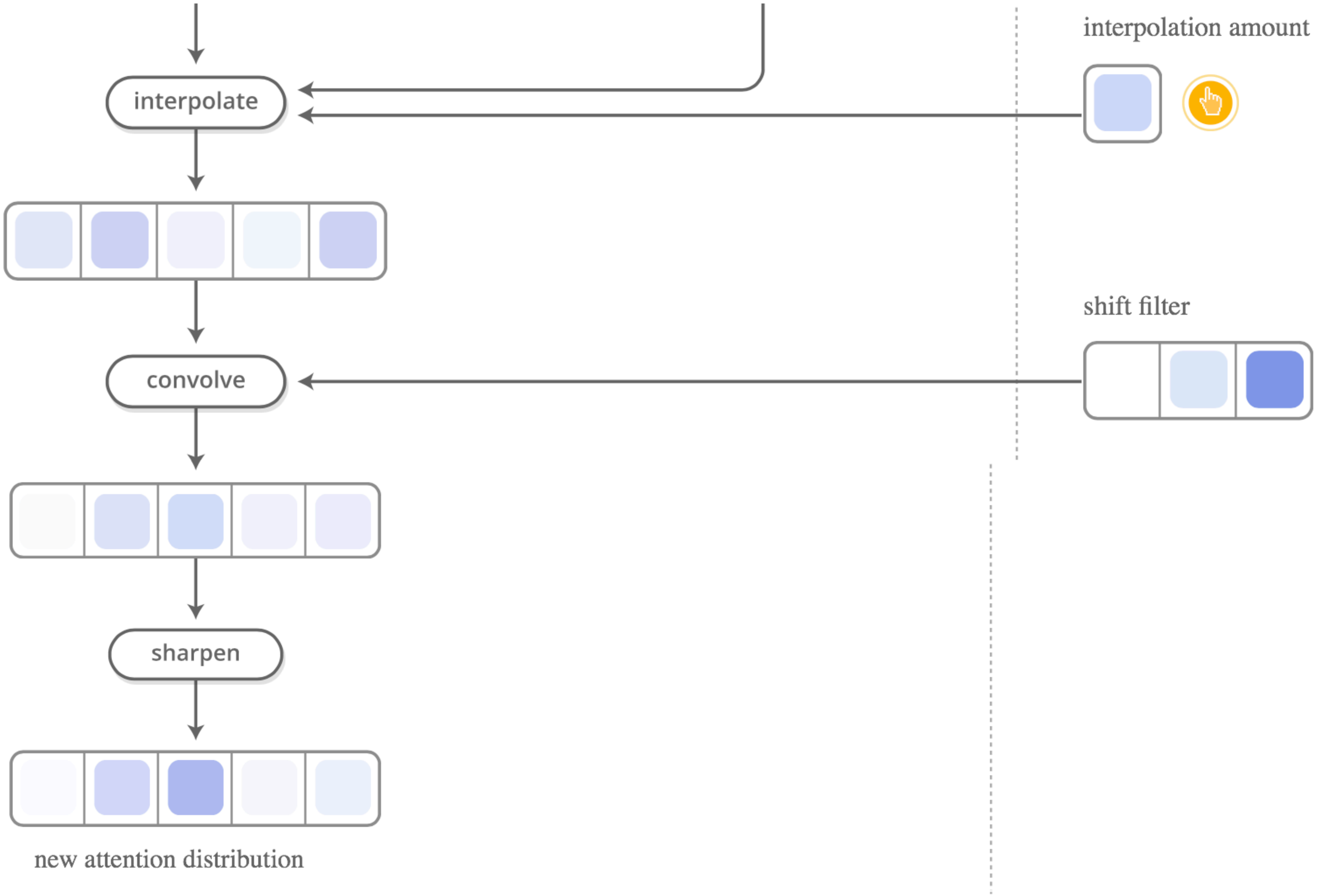
We convolve the attention with a shift filter—this allows the



Next, we interpolate the attention from the previous time step.

We convolve the attention with a shift filter—this allows the controller to move its focus.

Finally, we sharpen the attention distribution. This final attention distribution is fed to the read or write operation.



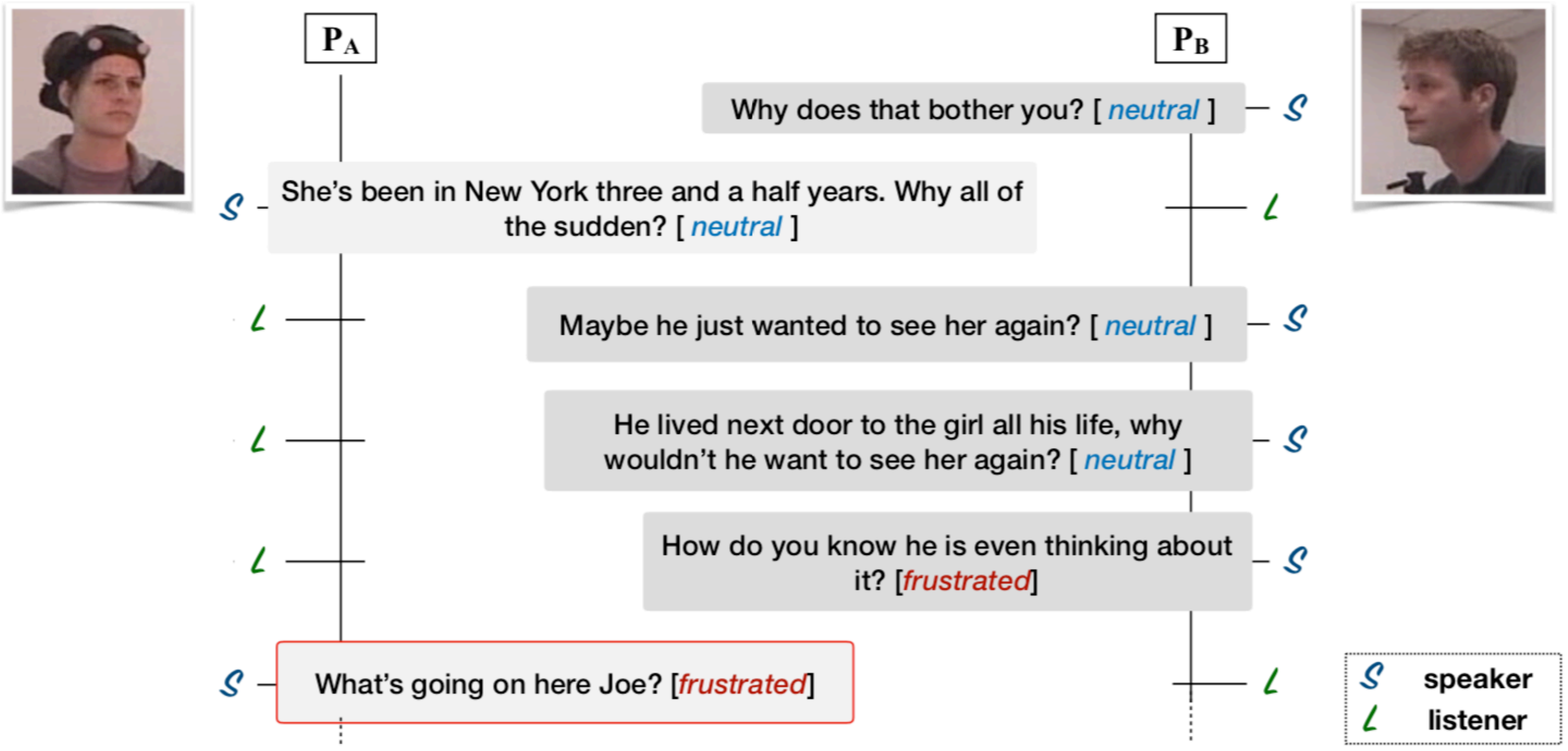
Graves et al, 2014

https://distill.pub/2016/augmented-rnns/

DialogueRNN:
An Attentive RNN for Emotion Detection in Conversations

AAAI 2019

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika , Rada Mihalcea, Alexander Gelbukh, Erik Cambria



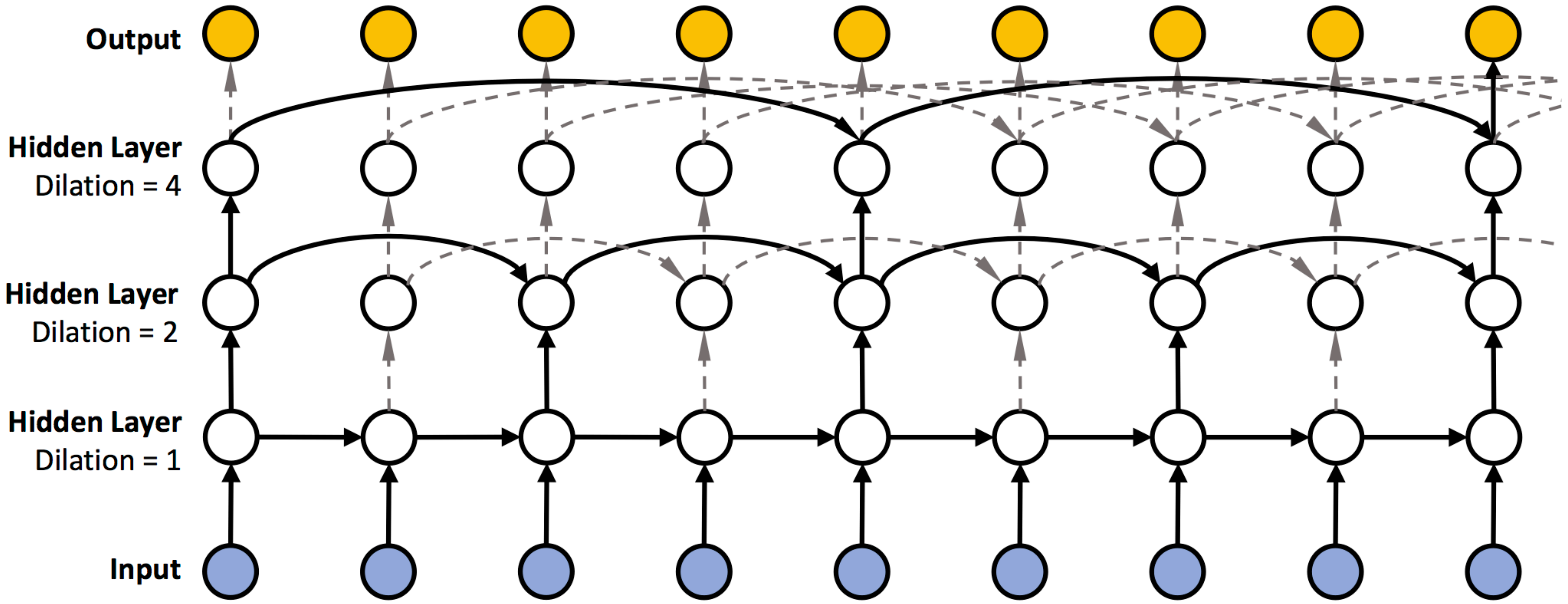
https://aaai.org/ojs/index.php/AAAI/article/view/4657

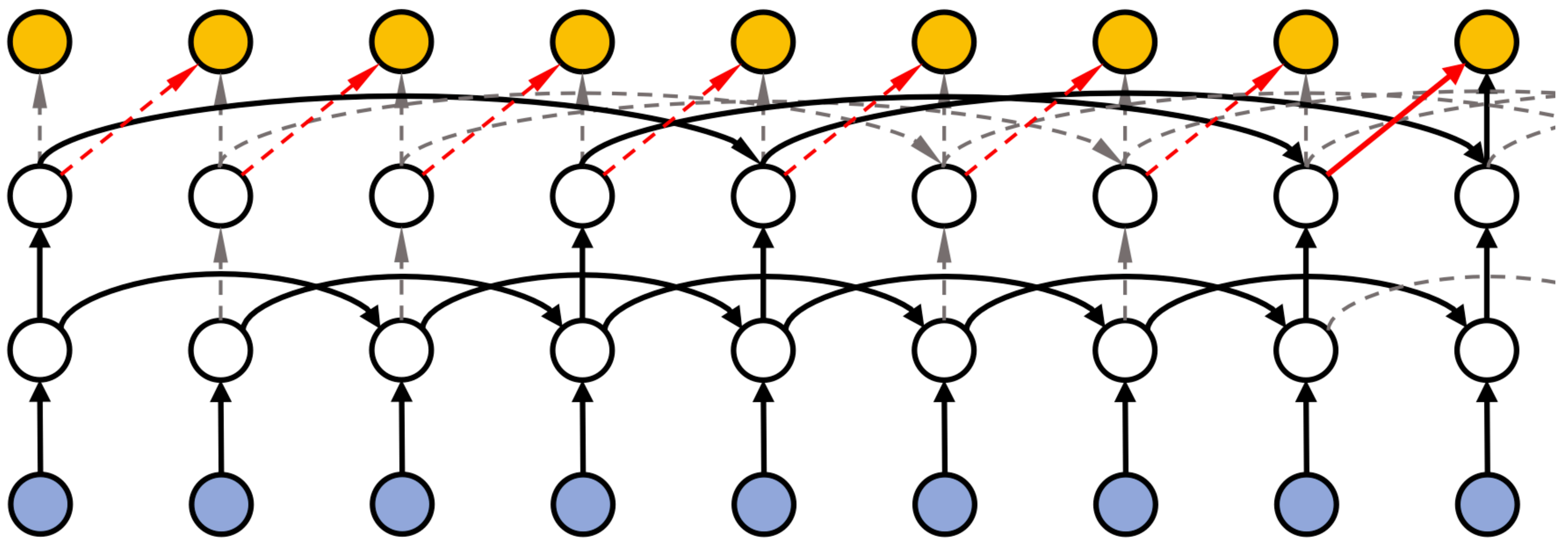
This week!

Dilated Recurrent Neural Networks

NeurIPS 2017

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui,
Michael Witbrock, Mark Hasegawa-Johnson, Thomas S. Huang





http://papers.nips.cc/paper/6613-dilated-recurrent-neural-networks.pdf

Next week!