Lecture 14

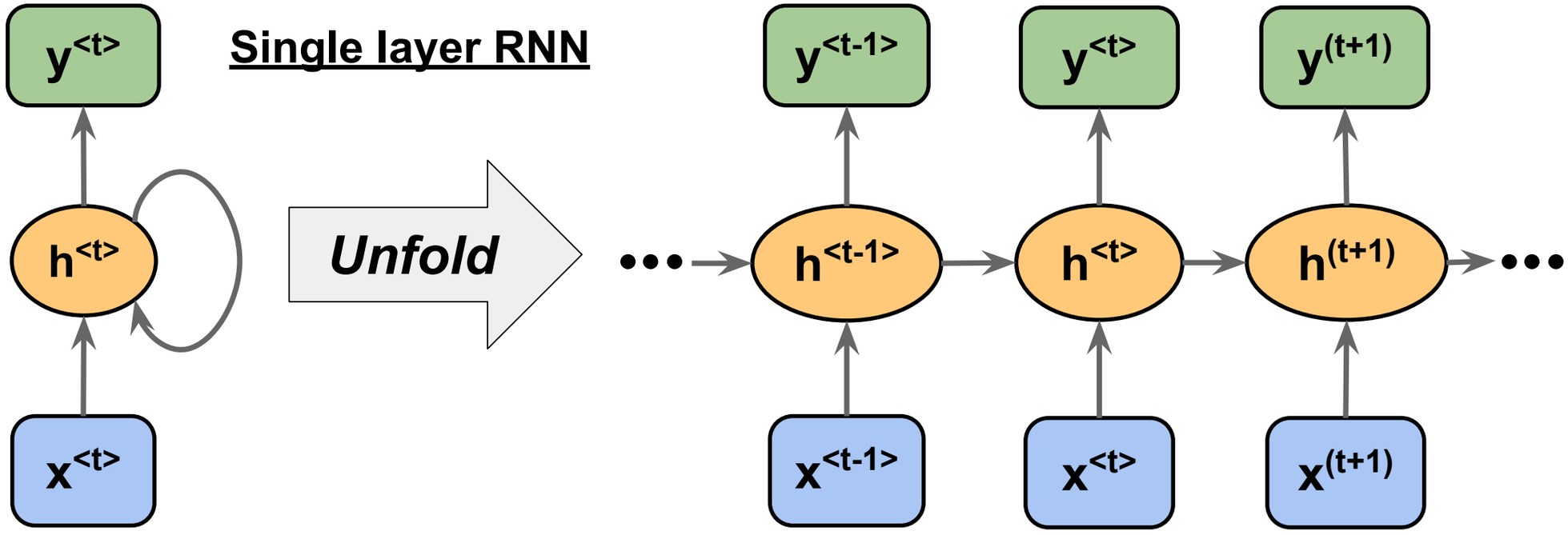
Introduction to Recurrent Neural Networks (Part 2)

STAT 479: Deep Learning, Spring 2019

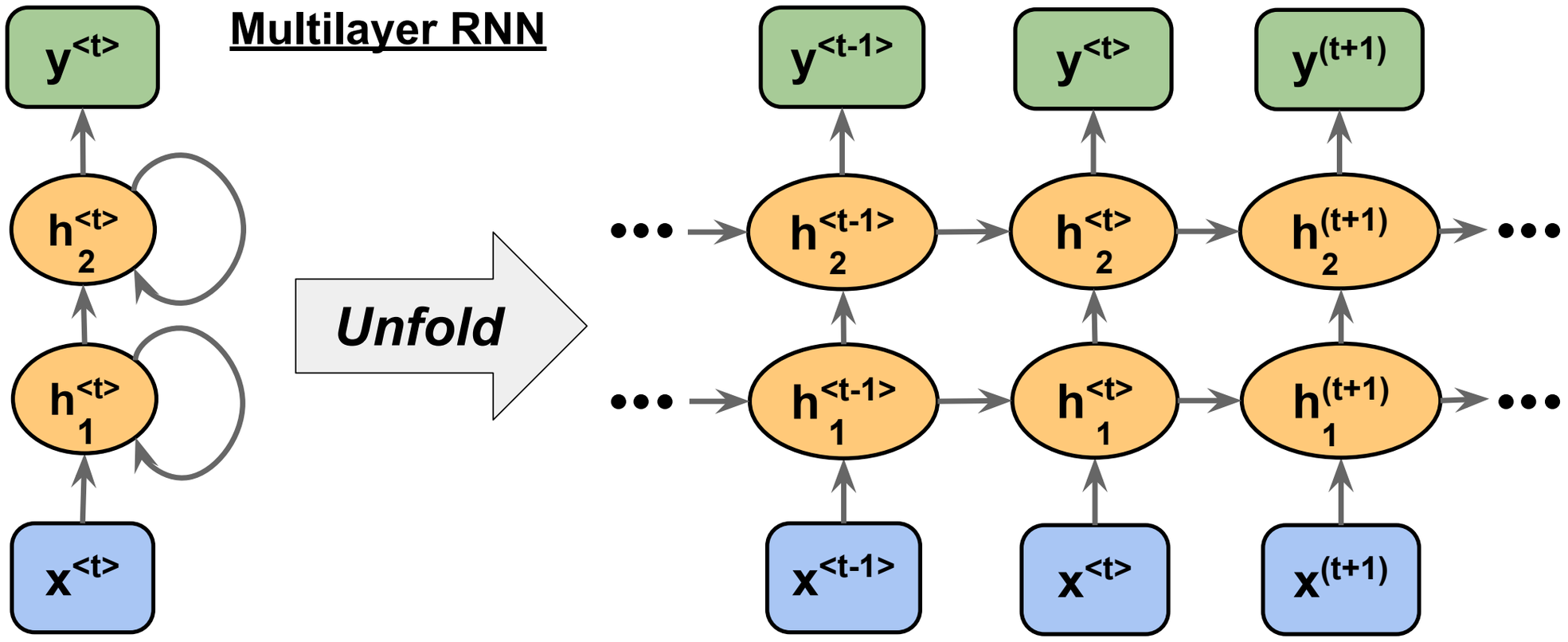
Sebastian Raschka

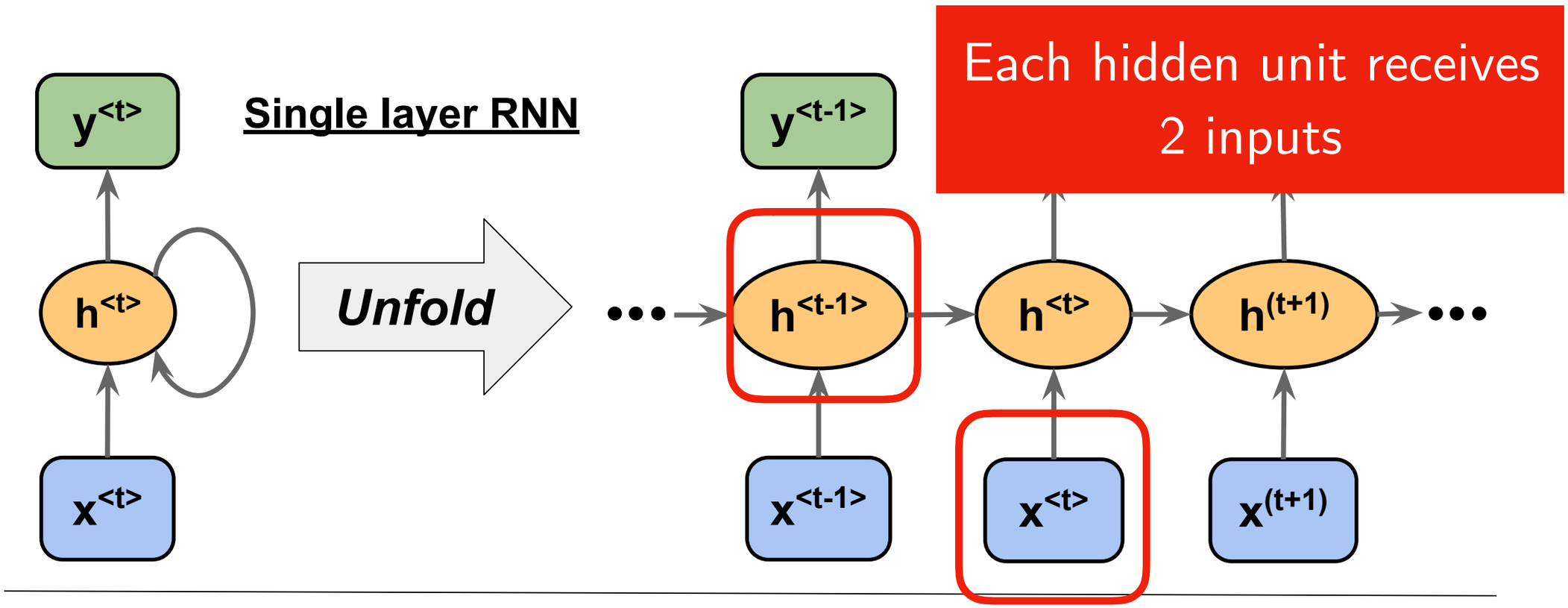
http://stat.wisc.edu/~sraschka/teaching/stat479-ss2019/

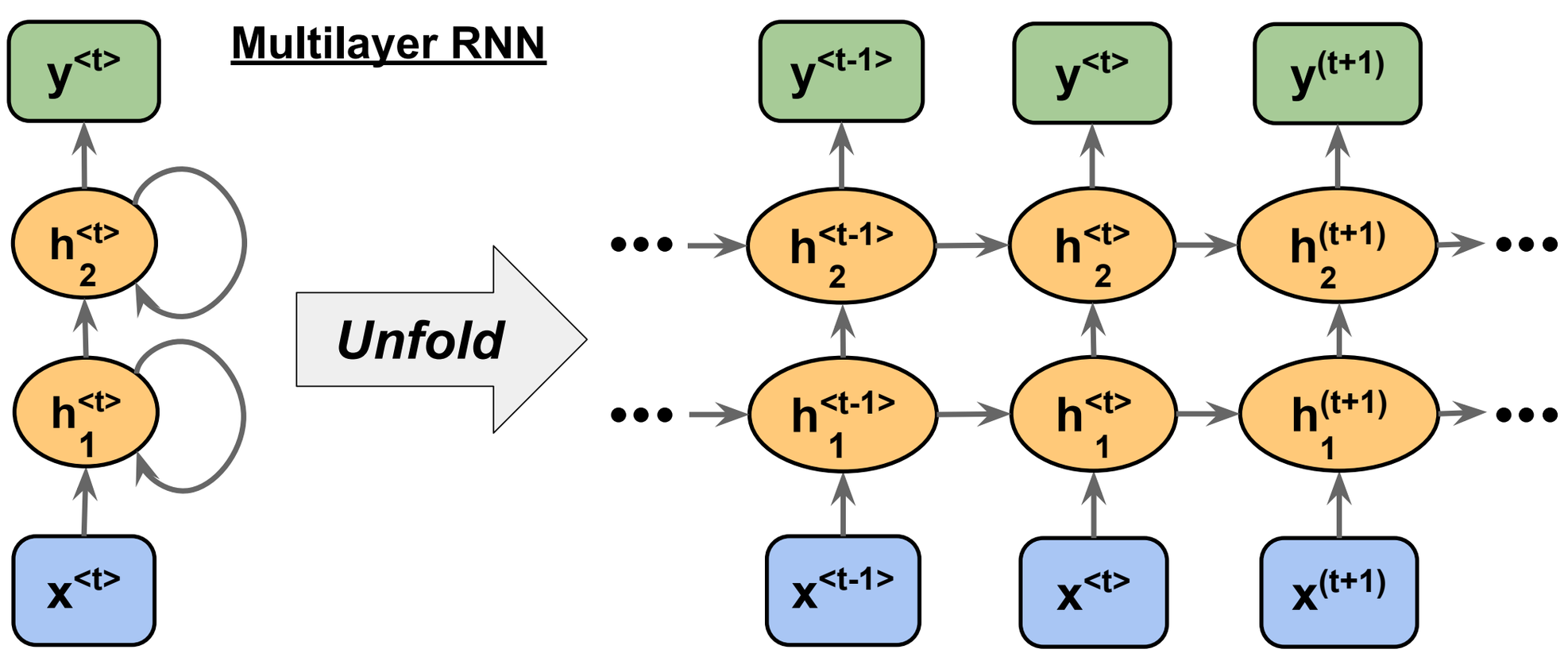
Single layer RNN



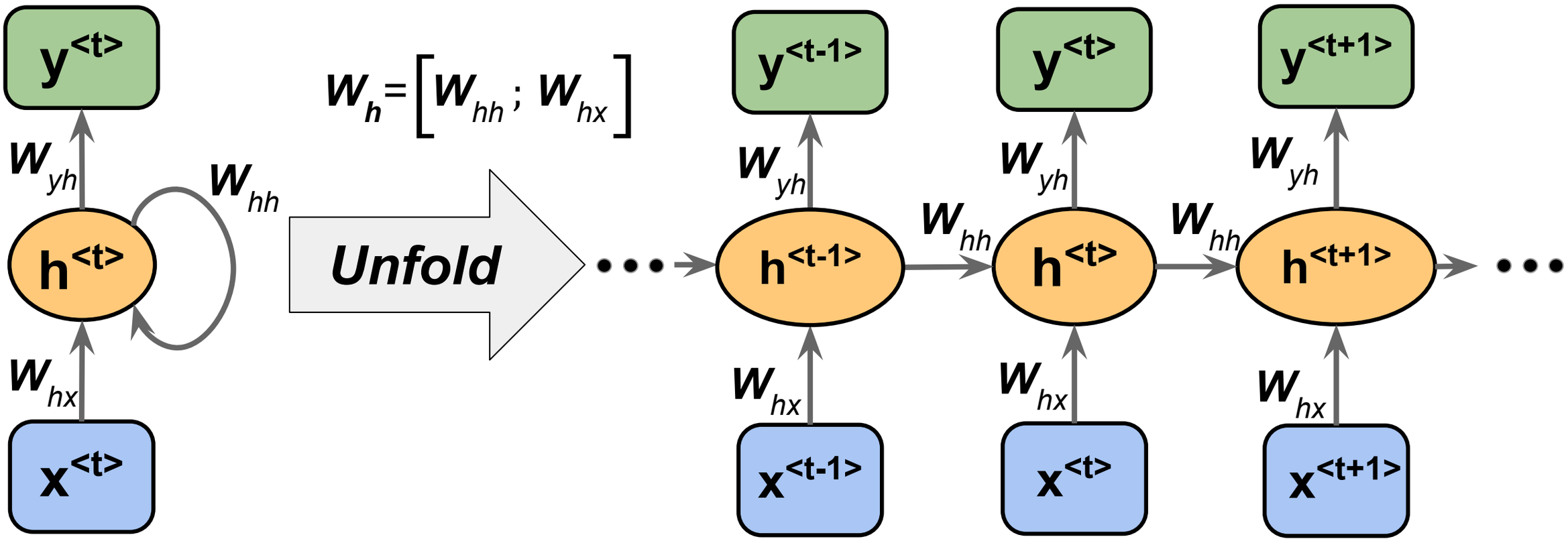
Multilayer RNN



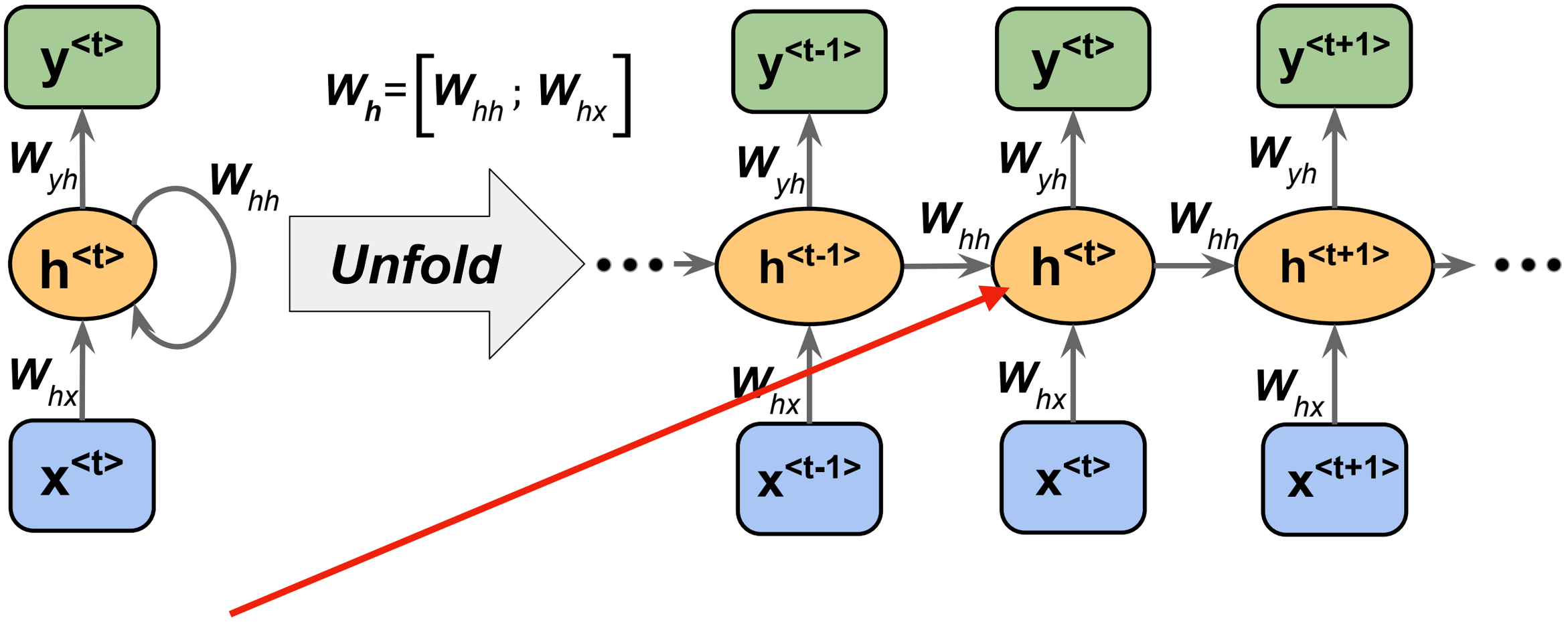

Single layer RNN
Unfold
Each hidden unit receives 2 inputs
y<t>
h<t>
x<t>
y<t-1>
h<t-1>
x<t-1>
h<t>
x<t>
h(t+1)
x(t+1)


Multilayer RNN
Unfold
y<t>
h2<t>
h1<t>
x<t>
y<t-1>
y<t>
y(t+1)
h2<t-1>
h2<t>
h2(t+1)
h1<t-1>
h1<t>
h1(t+1)
x<t-1>
x<t>
x(t+1)

Weight matrices in a single-hidden layer RNN



Weight matrices in a single-hidden layer RNN



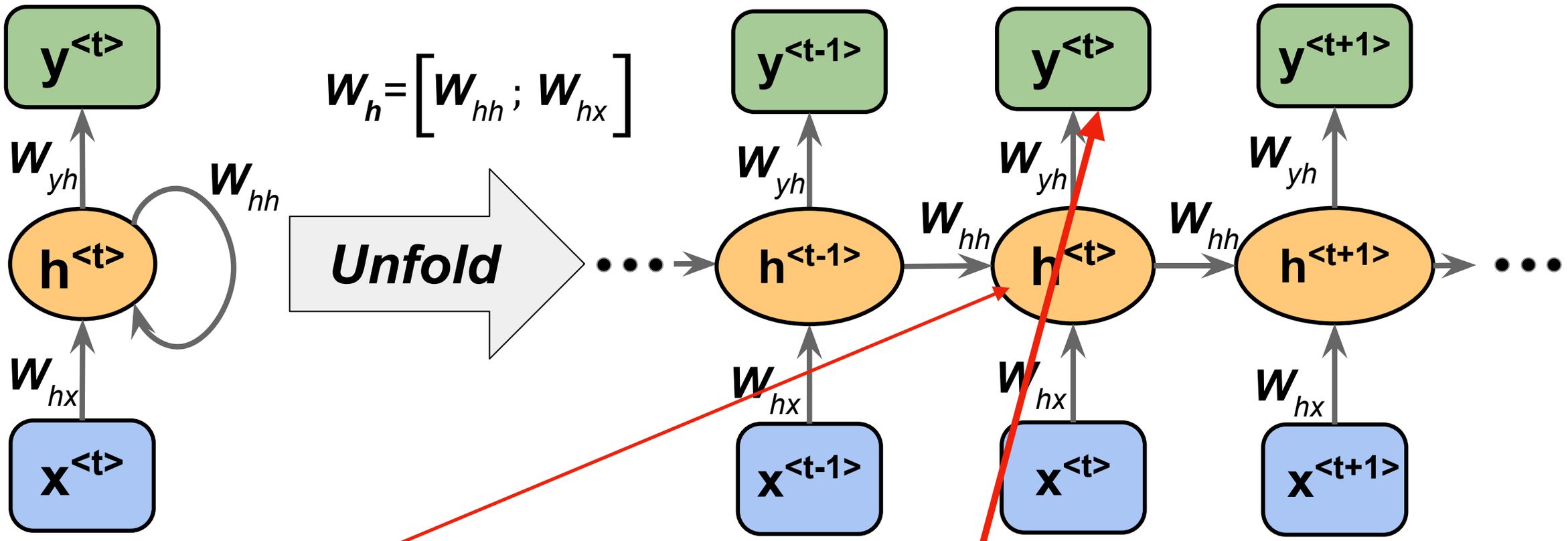
Net input:

$$\mathbf{z}_h^{\langle t \rangle} = \mathbf{W}_{hx}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{hh}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_h$$

Activation:

$$\mathbf{h}^{\langle t \rangle} = \sigma_h\left(\mathbf{z}_h^{\langle t \rangle}\right)$$

Weight matrices in a single-hidden layer RNN



Net input:

$$\mathbf{z}_h^{\langle t \rangle} = \mathbf{W}_{hx}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{hh}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_h$$

Activation:

$$\mathbf{h}^{\langle t \rangle} = \sigma_h\left(\mathbf{z}_h^{\langle t \rangle}\right)$$

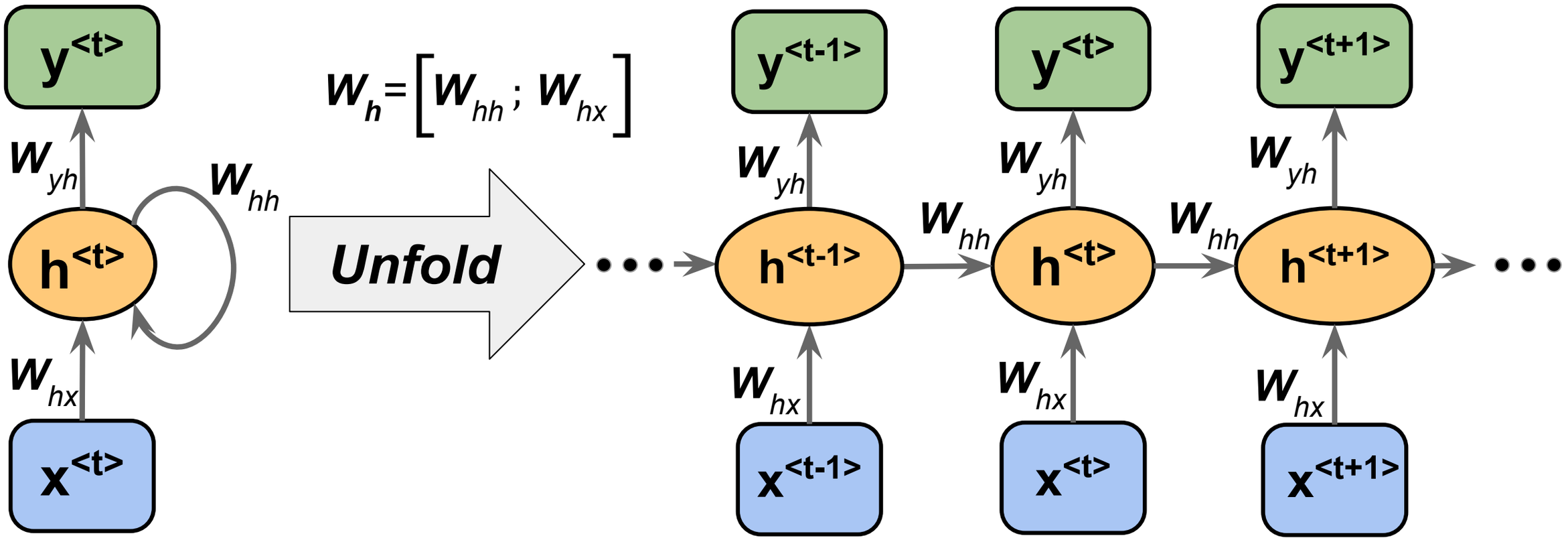
Net input:

$$\mathbf{z}_y^{\langle t \rangle} = \mathbf{W}_{yh}\mathbf{h}^{\langle t \rangle} + \mathbf{b}_y$$

Output:

$$\mathbf{y}^{\langle t \rangle} = \sigma_y\left(\mathbf{z}_y^{\langle t \rangle}\right)$$

Backpropagation through time

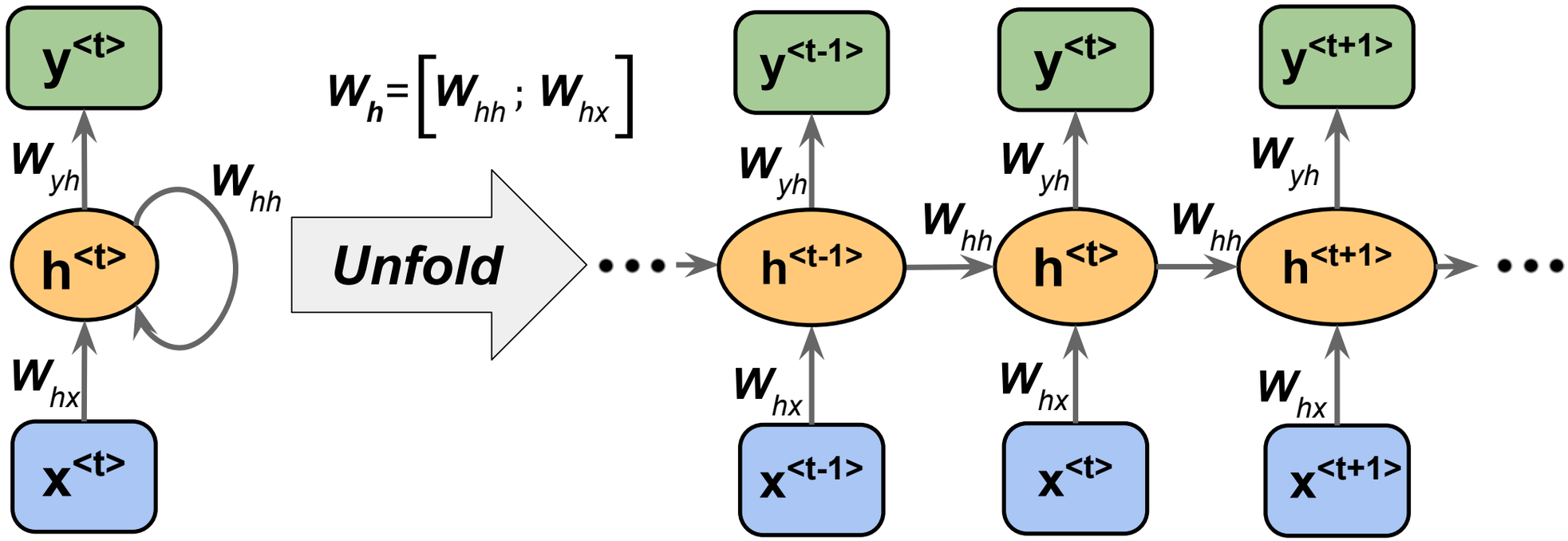


Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

The loss is computed as the sum over all time steps:

$$L = \sum_{t=1}^{T} L^{(t)}$$

Backpropagation through time

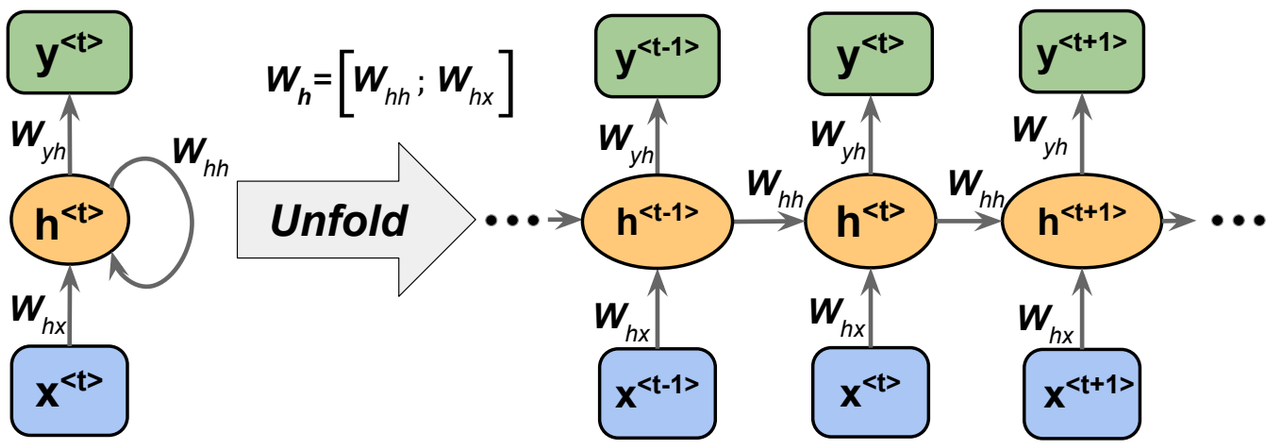


Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)}$$

$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

Backpropagation through time



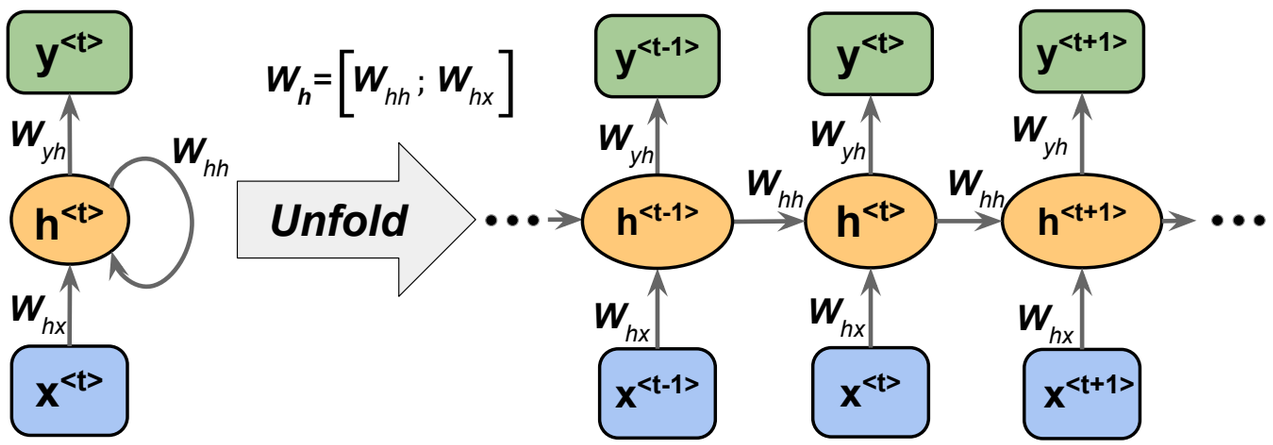
Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)} \qquad \frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

computed as a multiplication of adjacent time steps:

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

Backpropagation through time



Werbos, Paul J. "Backpropagation through time: what it does and how to do it." *Proceedings of the IEEE* 78, no. 10 (1990): 1550-1560.

$$L = \sum_{t=1}^{T} L^{(t)} \qquad \frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^{t} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

computed as a multiplication of adjacent time steps:

This is very problematic:
Vanishing/Exploding gradient problem!

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

A good resource that explains *backpropation through time* nicely:

Boden, Mikael. "A guide to recurrent neural networks and backpropagation." *the Dallas project* (2002).

2 Solutions to the vanishing/exploding gradient problem

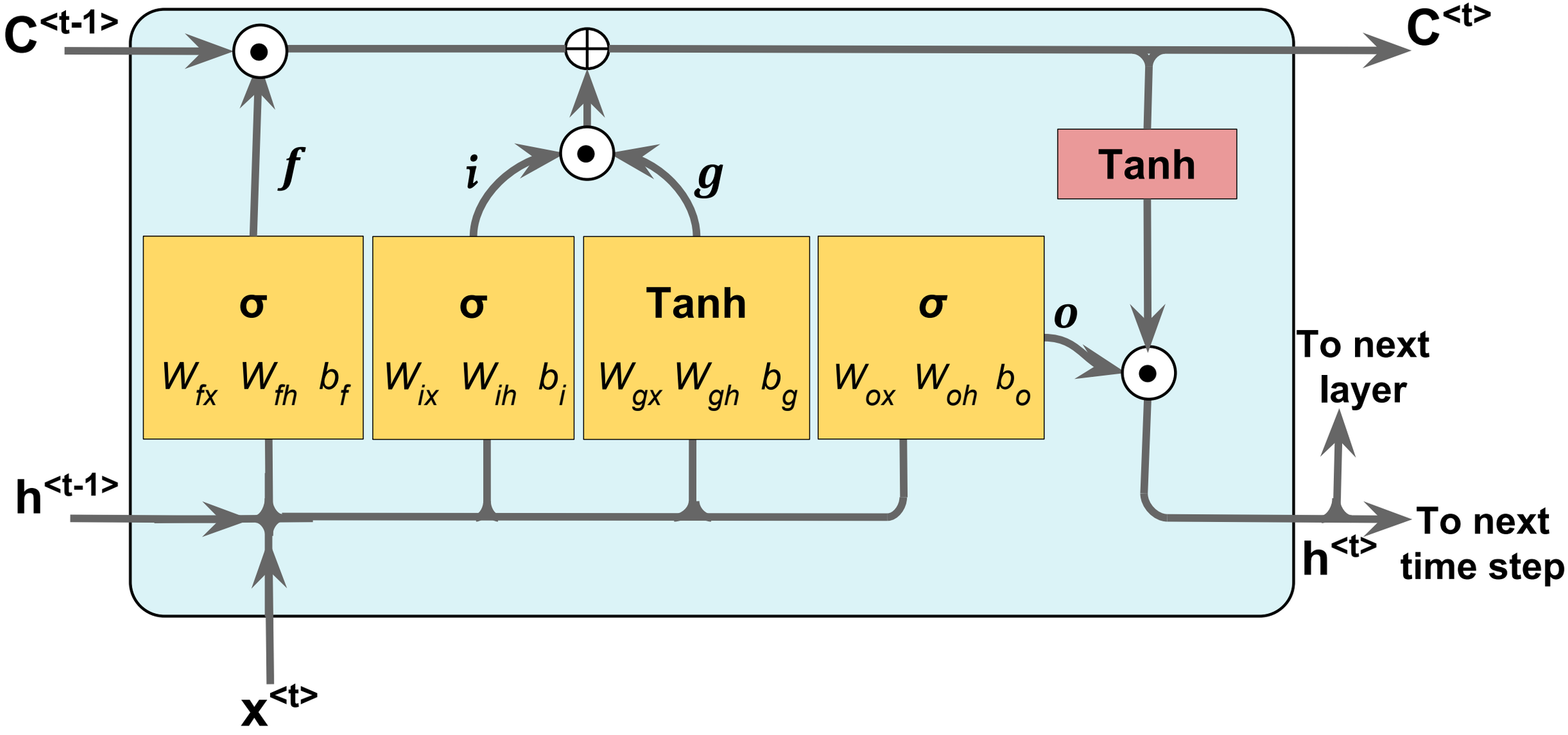
1) Truncated backpropagation through time (TBPTT)
   - simply limits the number of time steps the signal can backpropagate after each forward pass. E.g., even if the sequence has 100 elements/steps, we may only backpropagate through 20 or so

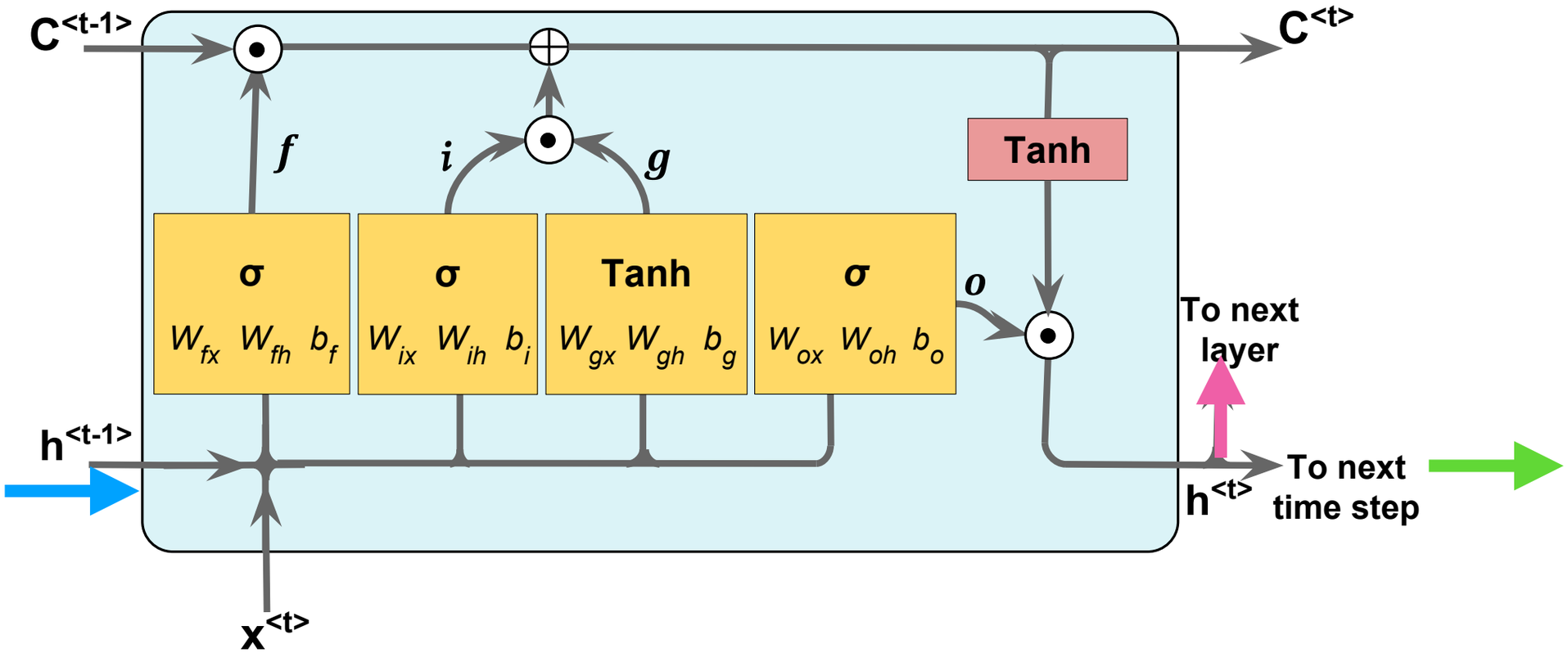
2) Long short-term memory (LSTM)

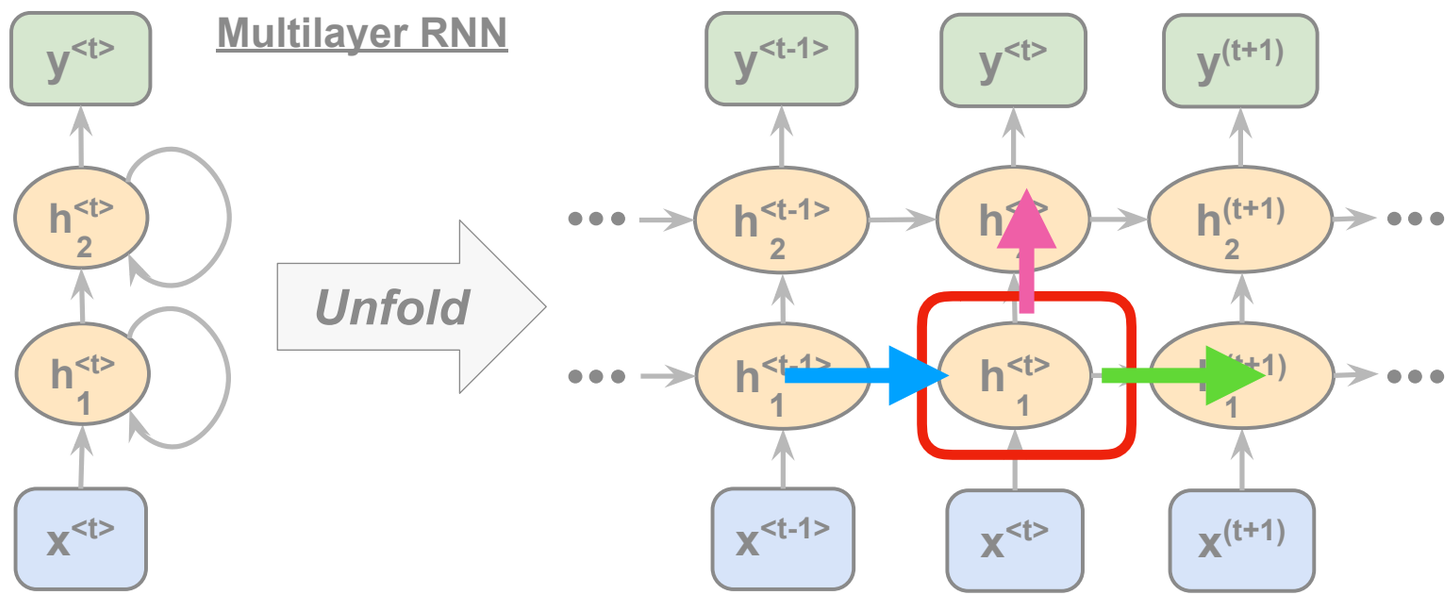
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9, no. 8 (1997): 1735-1780.

Long-short term memory (LSTM)

LSTM cell:



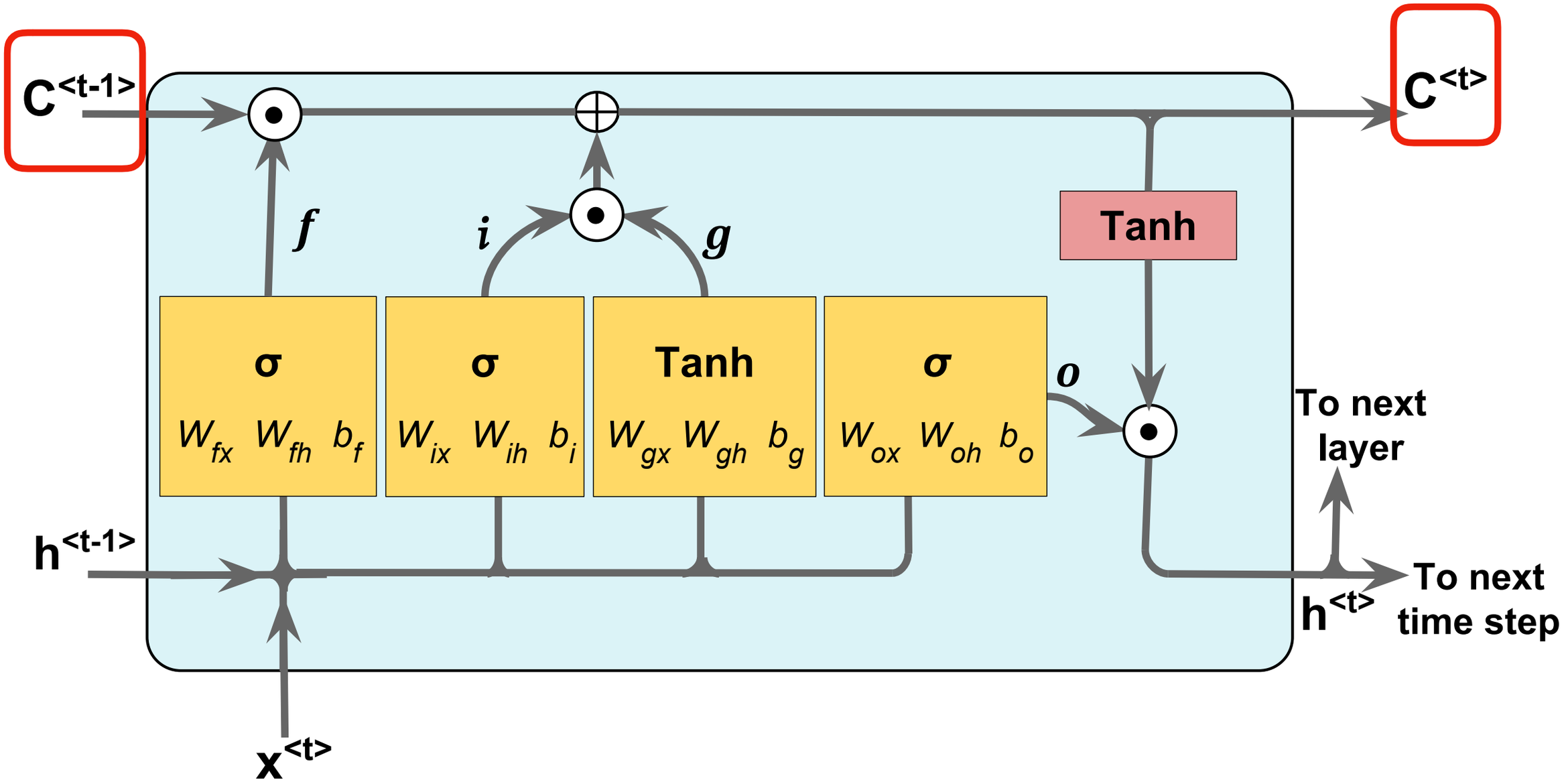

C<t-1>
C<t>
f
i
g
Tanh
σ
σ
Tanh
σ
o
W_fx W_fh b_f
W_ix W_ih b_i
W_gx W_gh b_g
W_ox W_oh b_o
To next layer
h<t-1>
To next time step
h<t>
x<t>


Multilayer RNN
y<t>
h2<t>
h1<t>
x<t>
Unfold
y<t-1>
y<t>
y(t+1)
h2<t-1>
h2(t+1)
h1<t-1>
h1<t>
h1(t+1)
x<t-1>
x<t>
x(t+1)

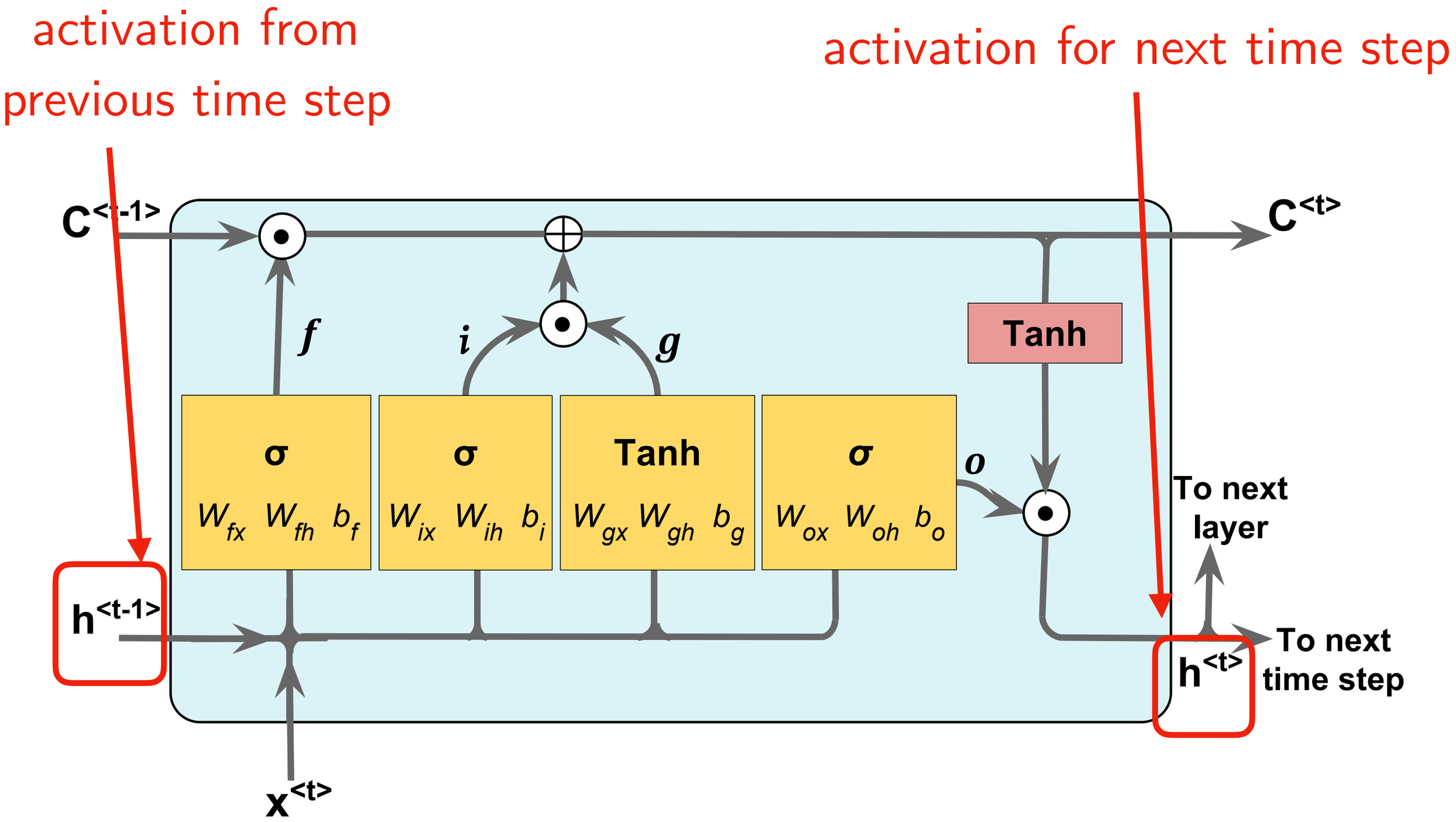
Long-short term memory (LSTM)

Cell state at previous time step

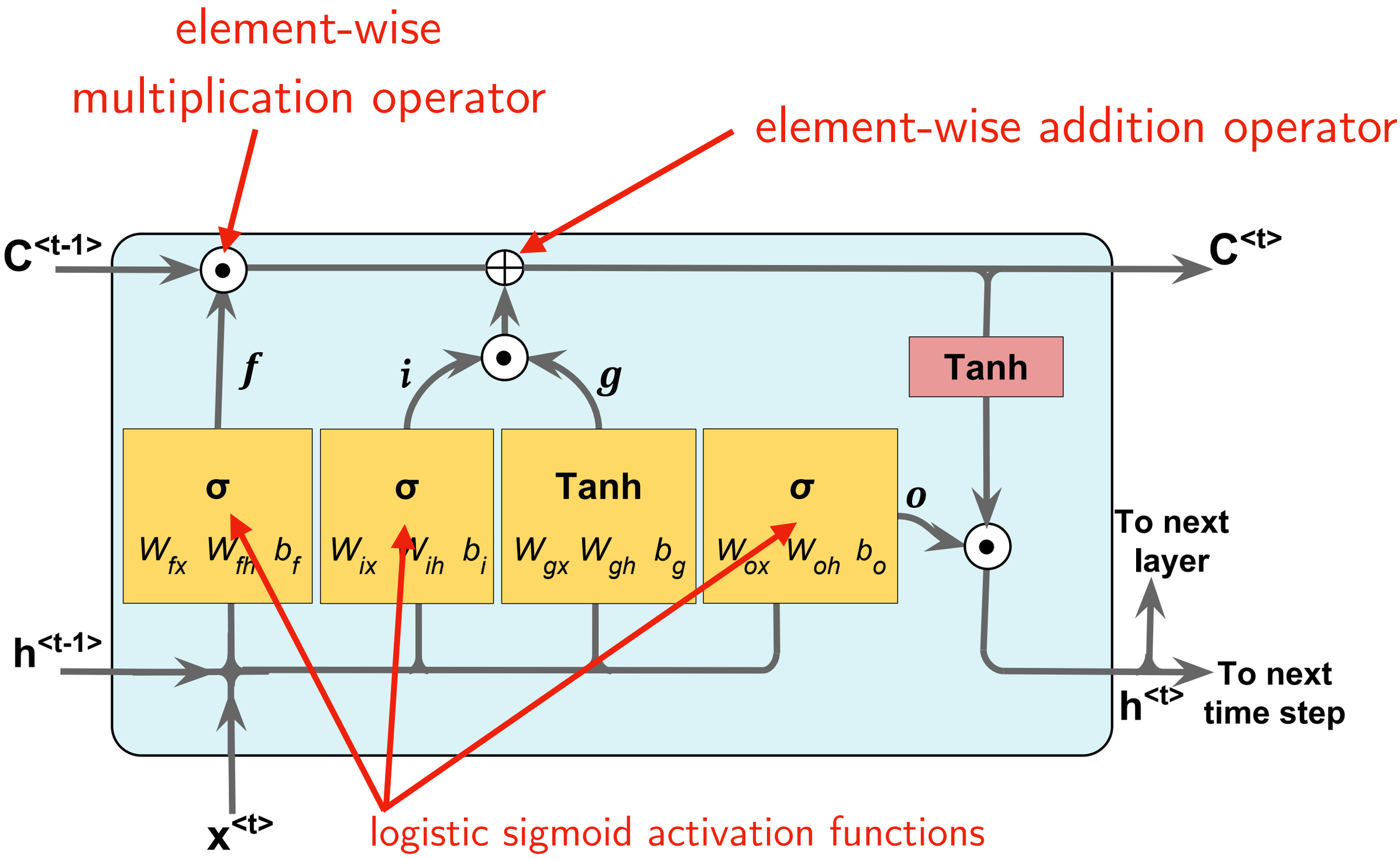
Cell state at current time step



Long-short term memory (LSTM)



Long-short term memory (LSTM)

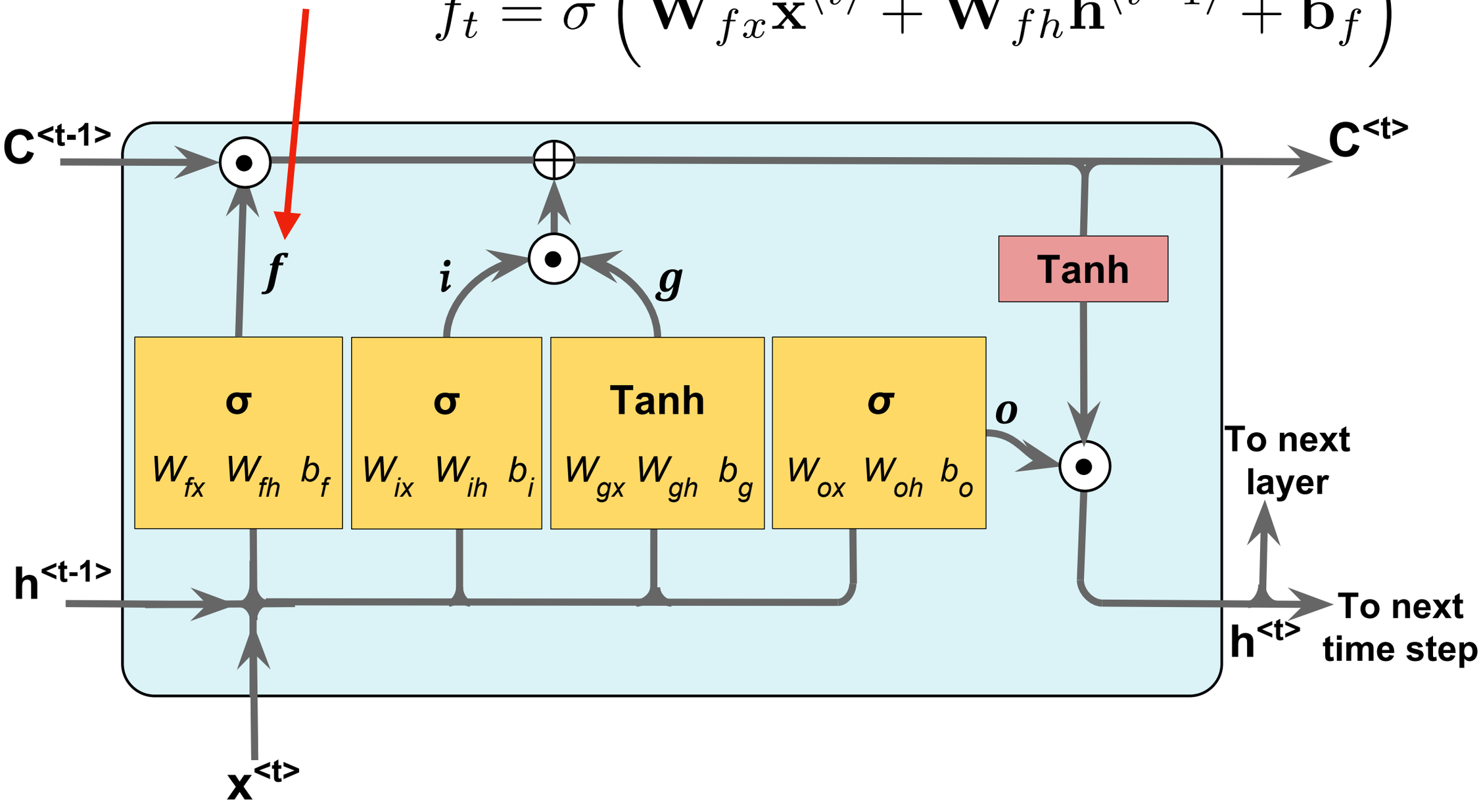


Long-short term memory (LSTM)

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." (1999): 850-855.

"Forget Gate": controls which information is remembered, and which is forgotten; can reset the cell state

$$f_t = \sigma\left(\mathbf{W}_{fx}\mathbf{x}^{\langle t\rangle} + \mathbf{W}_{fh}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_f\right)$$

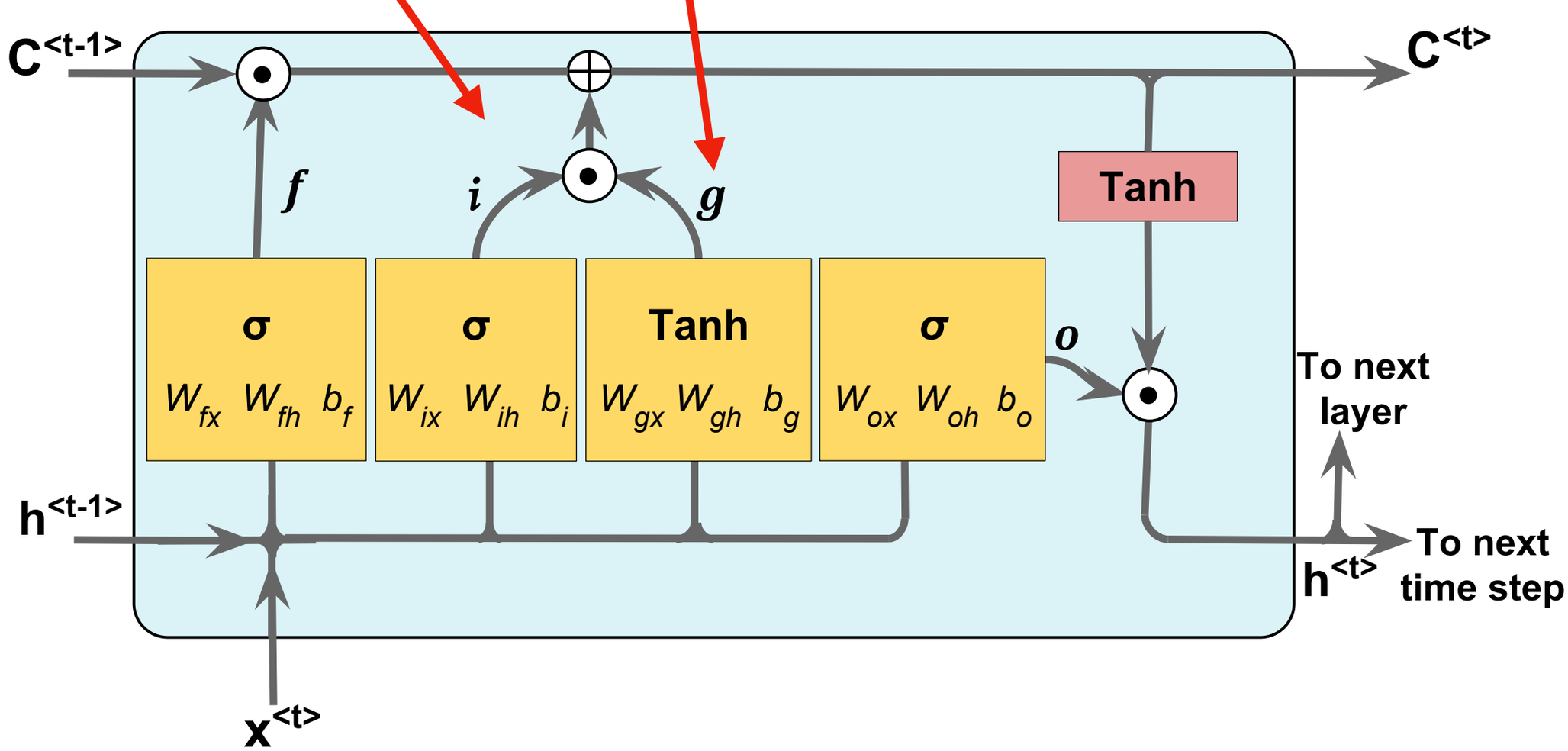


Long-short term memory (LSTM)

"Input Gate": $\mathbf{i}_t = \sigma\left(\mathbf{W}_{ix}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{ih}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_i\right)$

"Input Node":

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{gx}\mathbf{x}^{\langle t \rangle} + \mathbf{W}_{gh}\mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_g\right)$$



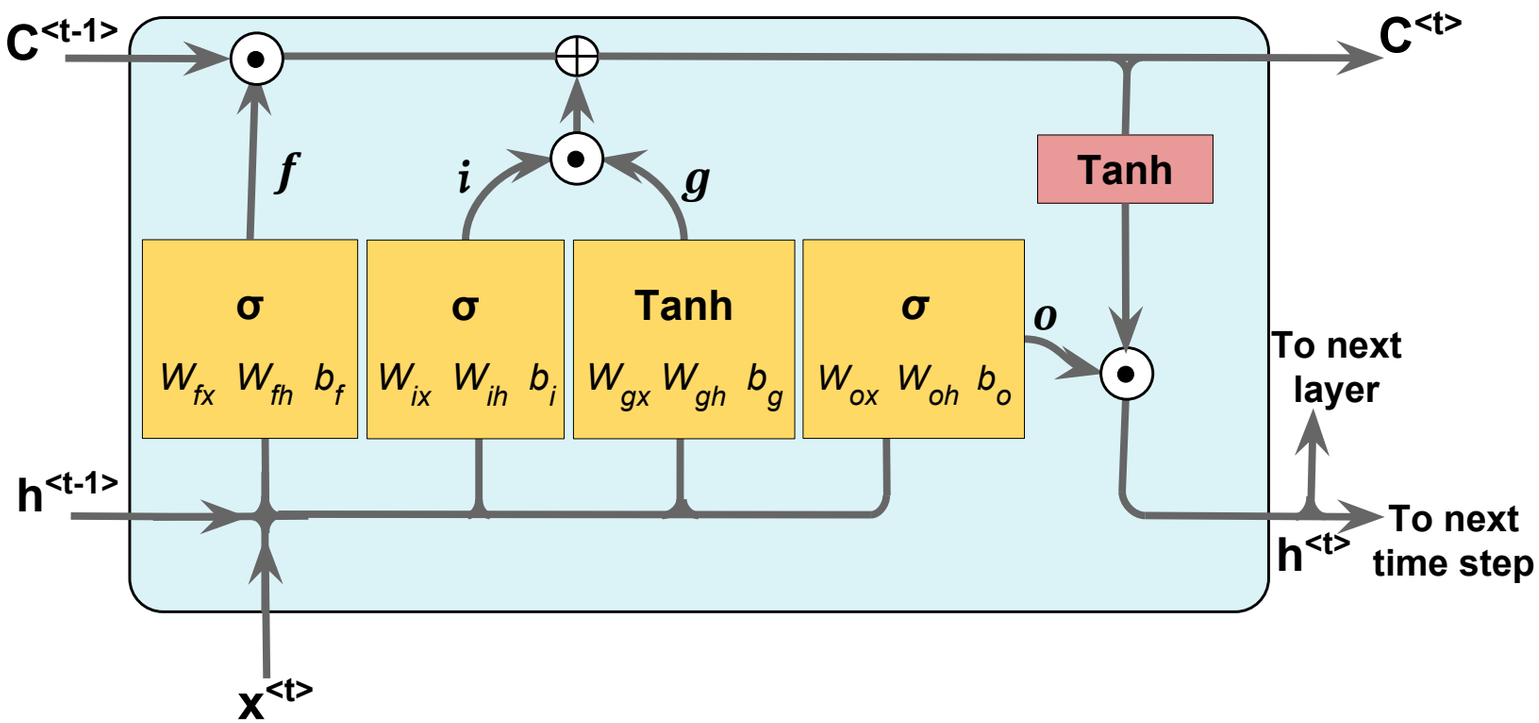
Long-short term memory (LSTM)

Brief summary of the gates so far ...

Forget Gate Input Node Input Gate

$$C^{\langle t \rangle} = \Big(C^{\langle t-1 \rangle} \odot f_t \Big) \oplus (i_t \odot g_t)$$

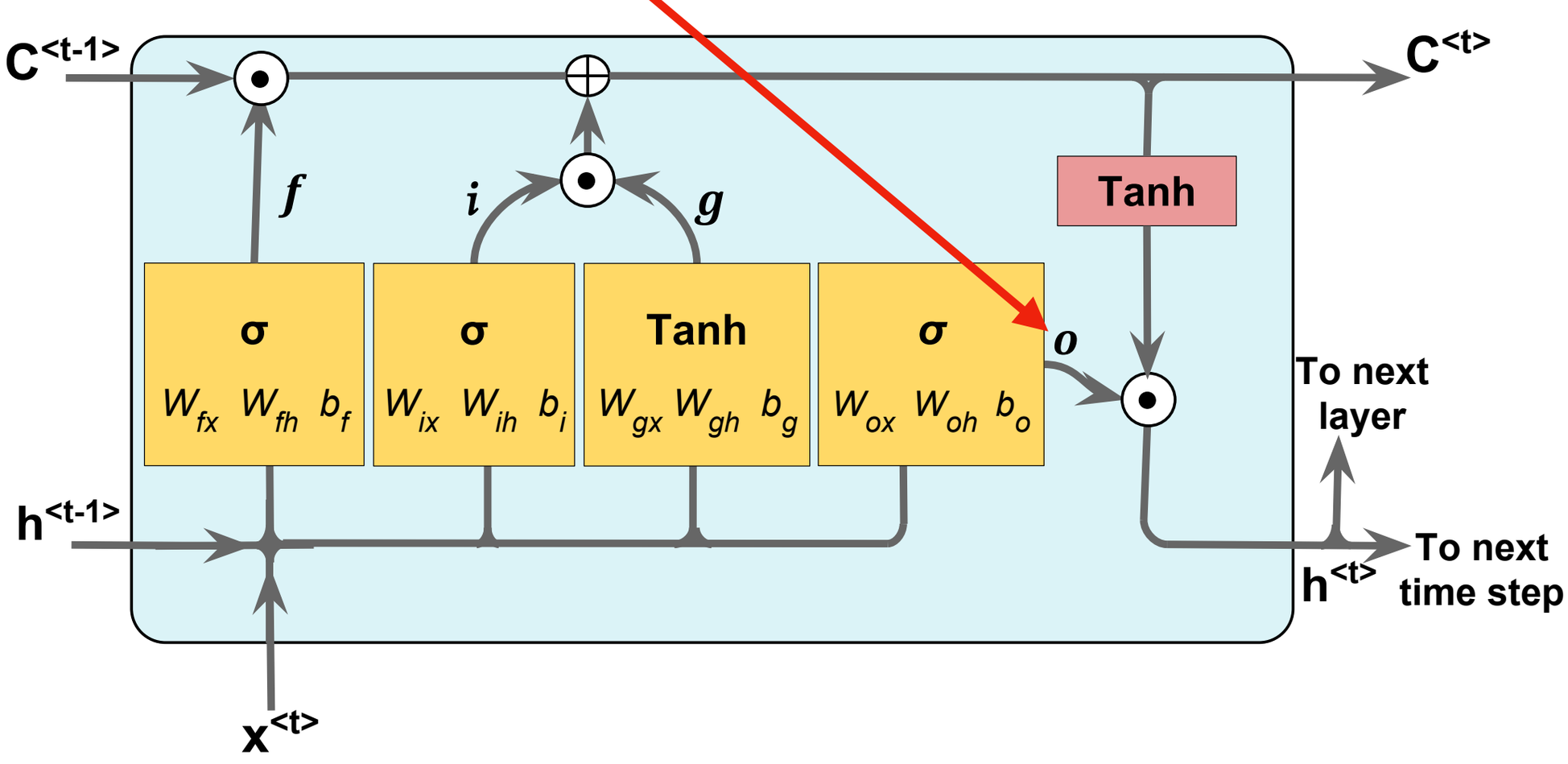
For updating the cell state



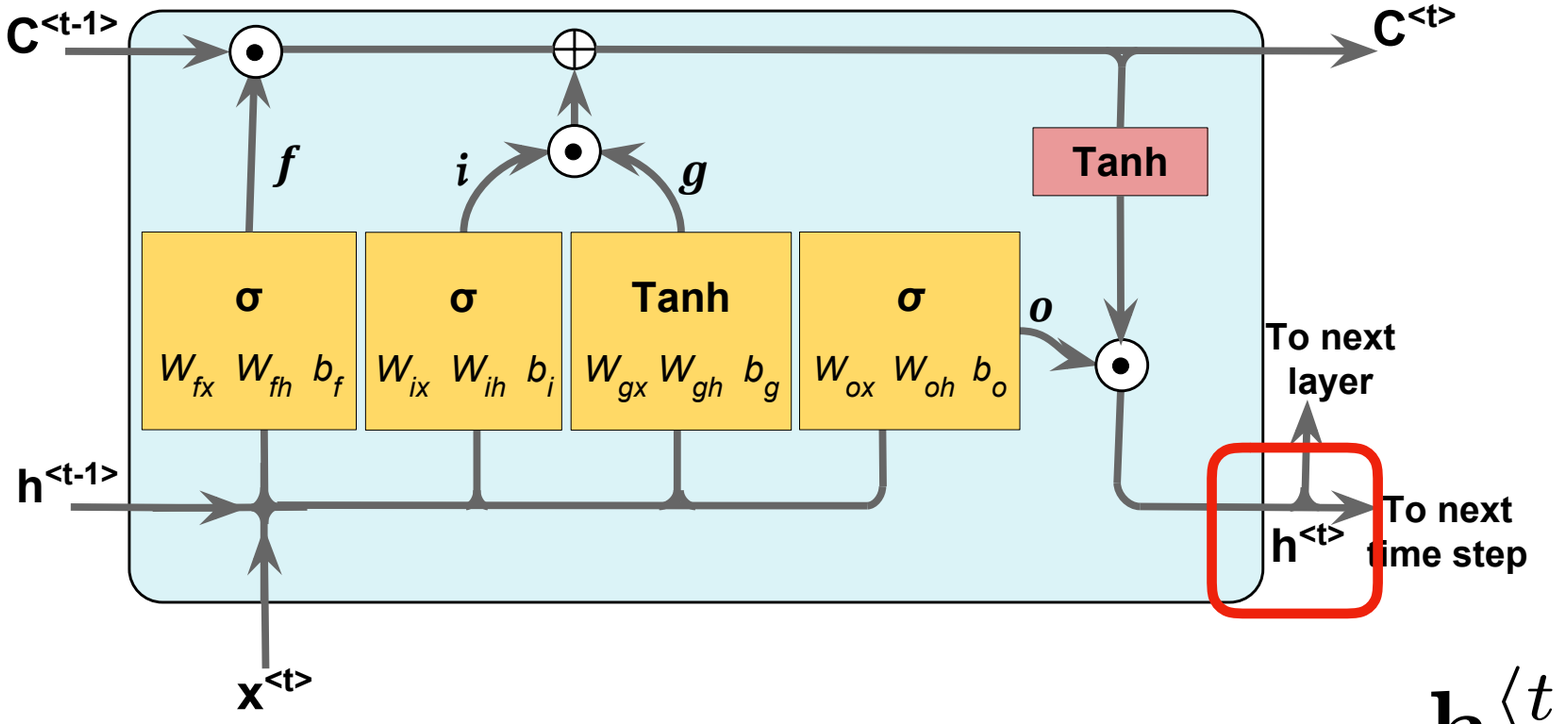
Long-short term memory (LSTM)

Output gate for updating the values of hidden units:

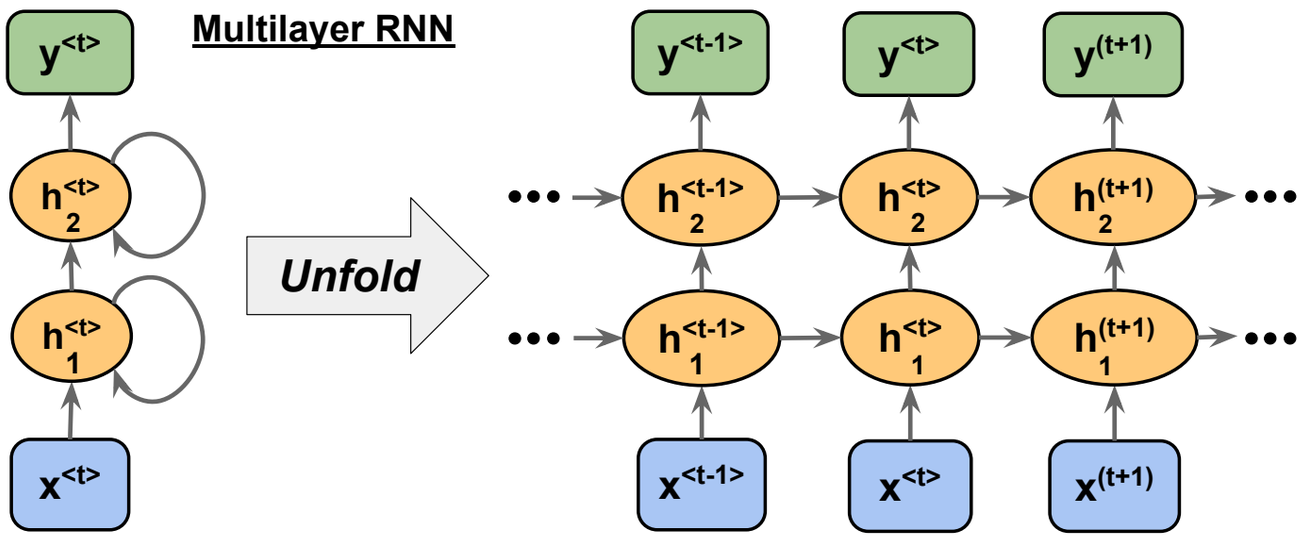
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{ox}\mathbf{x}^{\langle t\rangle} + \mathbf{W}_{oh}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_o\right)$$



Long-short term memory (LSTM)



$$\mathbf{h}^{\langle t \rangle} = \mathbf{o}_t \odot \tanh\left(\mathbf{C}^{\langle t \rangle}\right)$$

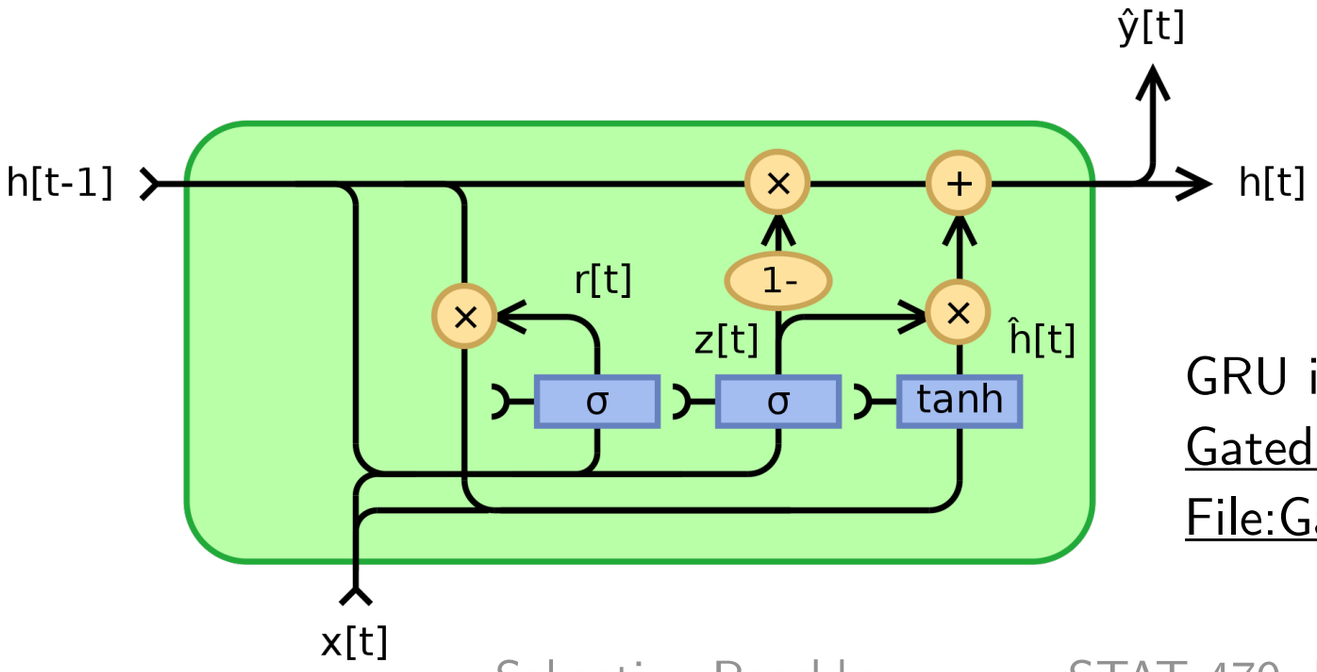


Long-short term memory (LSTM)

- Still popular and widely used today
- A recent, related approach is the Gated Recurrent Unit (GRU)

  Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).
- Nice article exploring LSTMs and comparing them to GRUs

  Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." In *International Conference on Machine Learning*, pp. 2342-2350. 2015.



GRU image source: https://en.wikipedia.org/wiki/Gated_recurrent_unit#/media/File:Gated_Recurrent_Unit,_base_type.svg

##

Why language models could be smarter than you think:
...LSTMs are smarter than you think, say researchers...
Researchers with the Cognitive Neuroimaging Unit at the 'NeuroSpin Center', as well as Facebook AI Research and the University of Amsterdam, have analyzed how LSTMs keep track of certain types of information, when tested for their ability to model language structures at longer timescales. The analysis is meant to explore "whether these generic sequence-processing devices are discovering genuine structural properties of language in their training data, or whether their success can be explained by opportunistic surface-pattern-based heuristics", the authors write.

The researchers study a pre-trained model "composed of a 650-dimensional embedding layer, two 650-dimensional hidden layers, and an output layer with vocabulary size 50,000". They evaluate this model on a set of what they call 'number-agreement tasks' where they test subject-verb agreement in increasingly challenging setups (eg, a simple case is looking at network activations for 'the boy greets the guy', with harder ones taking the form of things like 'the boy most probably greets the guy' and 'the boy near the car kindly greets the guy', and so on).

Neurons that count, sometimes together: During analysis, they noticed the LSTM had developed "two 'grandmother' cells to carry number features from the subject to the verb across the intervening material". They found that these cells were sometimes used to help the network decide in particularly tricky cases: "The LSTM also possesses a more distributed mechanism to predict number when subject and verb are close, with the grandmother number cells only playing a crucial role in more difficult long-distance cases".

They also discovered a cell that "encodes the presence of an embedded phrase separating the main subject-verb dependency, and has strong efferent connections to the long-distance number cells, suggesting that the network relies on genuine syntactic information to regulate agreement-feature percolation".

Why this matters: "Strikingly, simply training an LSTM on a language-model objective on raw corpus data brought about single units carrying exceptionally specific linguistic information. Three of these units were found to form a highly interactive local network, which makes up the central part of a 'neural' circuit performing long-distance number agreement", they write. "Agreement in an LSTM language-model cannot be entirely explained away by superficial heuristics, and the networks have, to some extent, learned to build and exploit structure-based syntactic representations, akin to those conjectured to support human-sentence processing"

The most interesting thing about all of this is the apparent sophistication that emerges as we train these networks. It seems to inherently support some of the ideas outlined by Richard Sutton (covered in Import AI #138) about the ability for relatively simple networks to - given sufficient compute - develop very sophisticated capabilities.

Read more: The emergence of number and syntax units in LSTM language models (Arxiv).
##

Source: https://jack-clark.net/2019/03/25/making-better-healthcare-ai-systems-via-audio-de-identification-teaching-drones-to-help-humans-fight-fires-and-why-language-models-could-be-smarter-than-you-think/

Next part: Training Word-RNN for sentiment classification and a Character-RNN for text generation