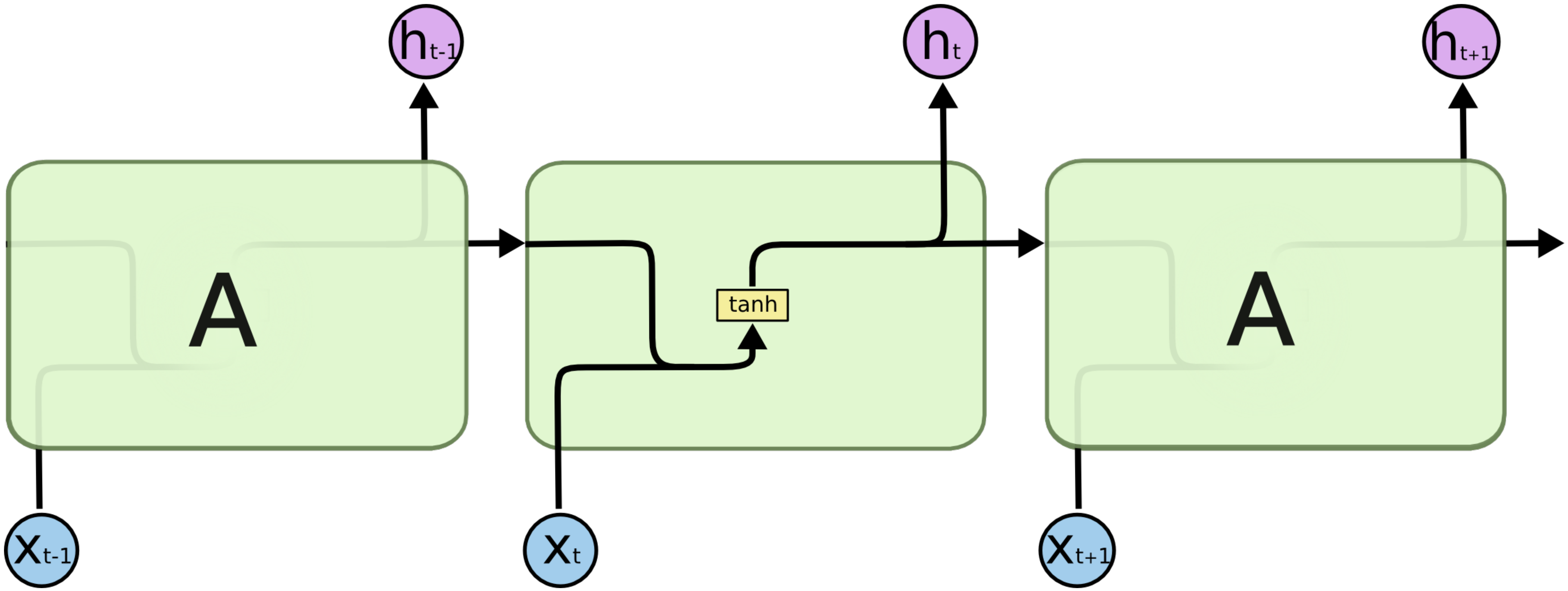
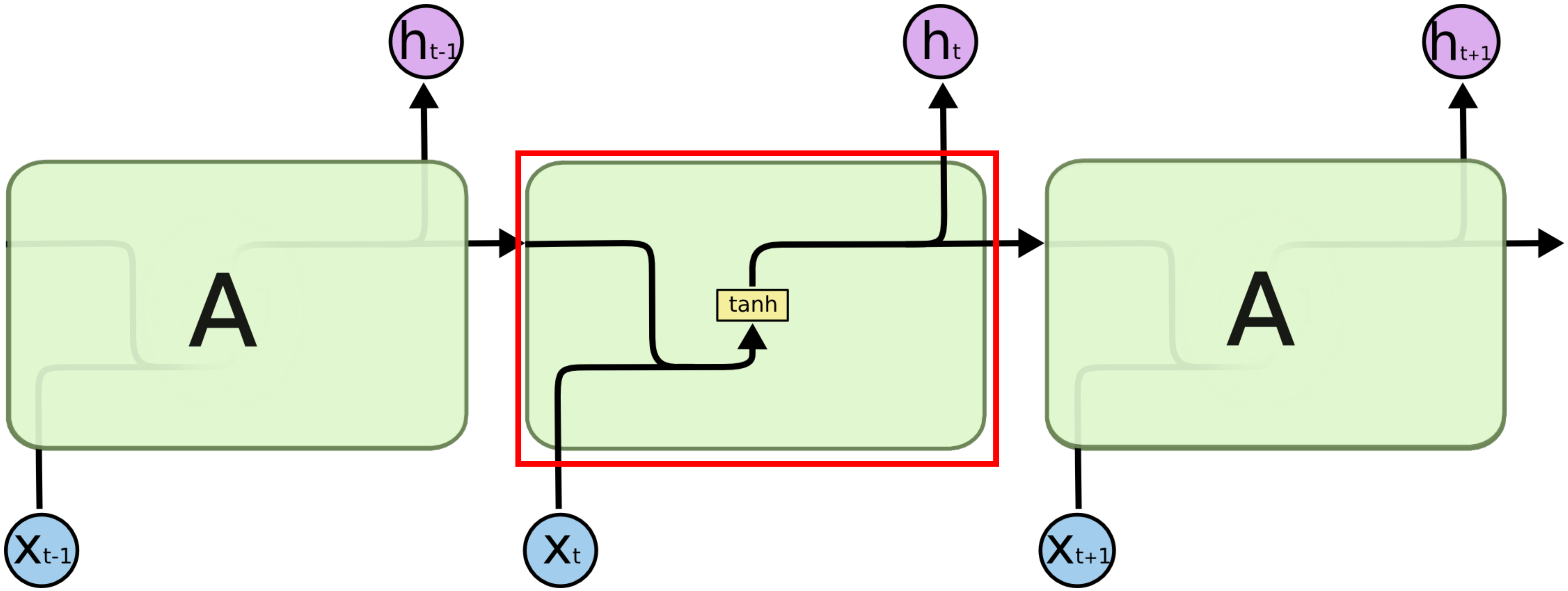
Long-Short Term Memory

Neural Networks Design And Application

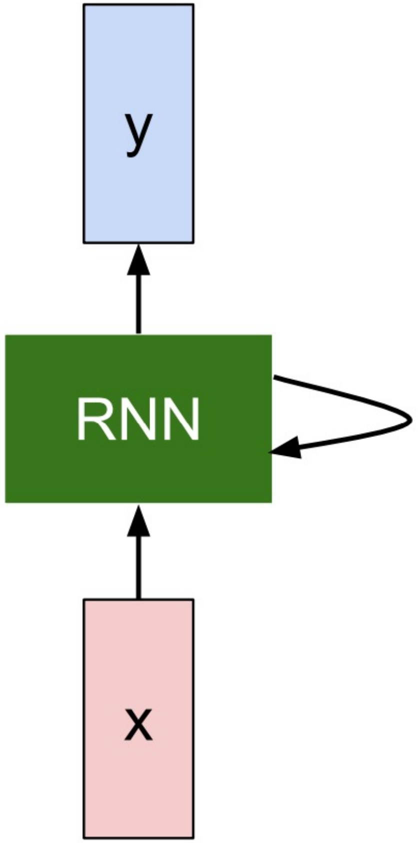
A conventional RNN



A conventional RNN



Recurrent neural networks



$$h_t = f\left[W * \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right]$$

$$f = \tanh(\cdot)$$

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

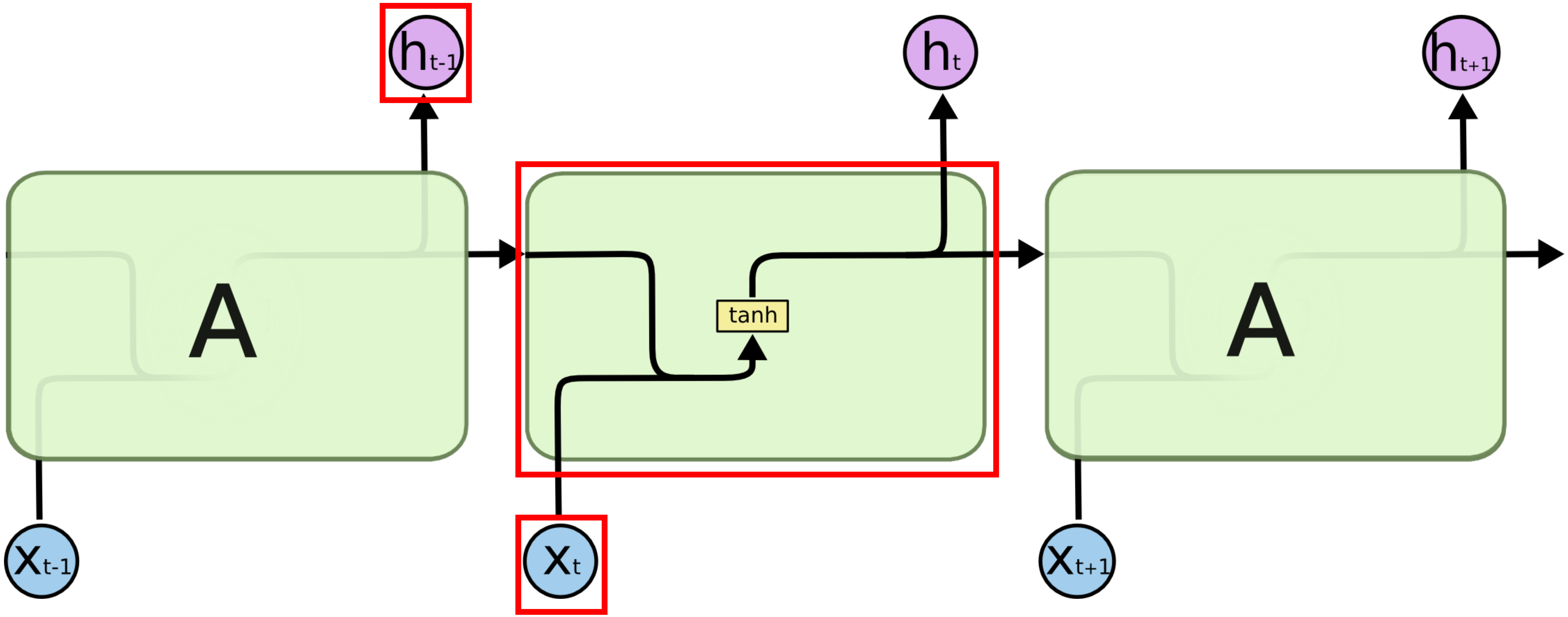
new state

some function
with parameters W

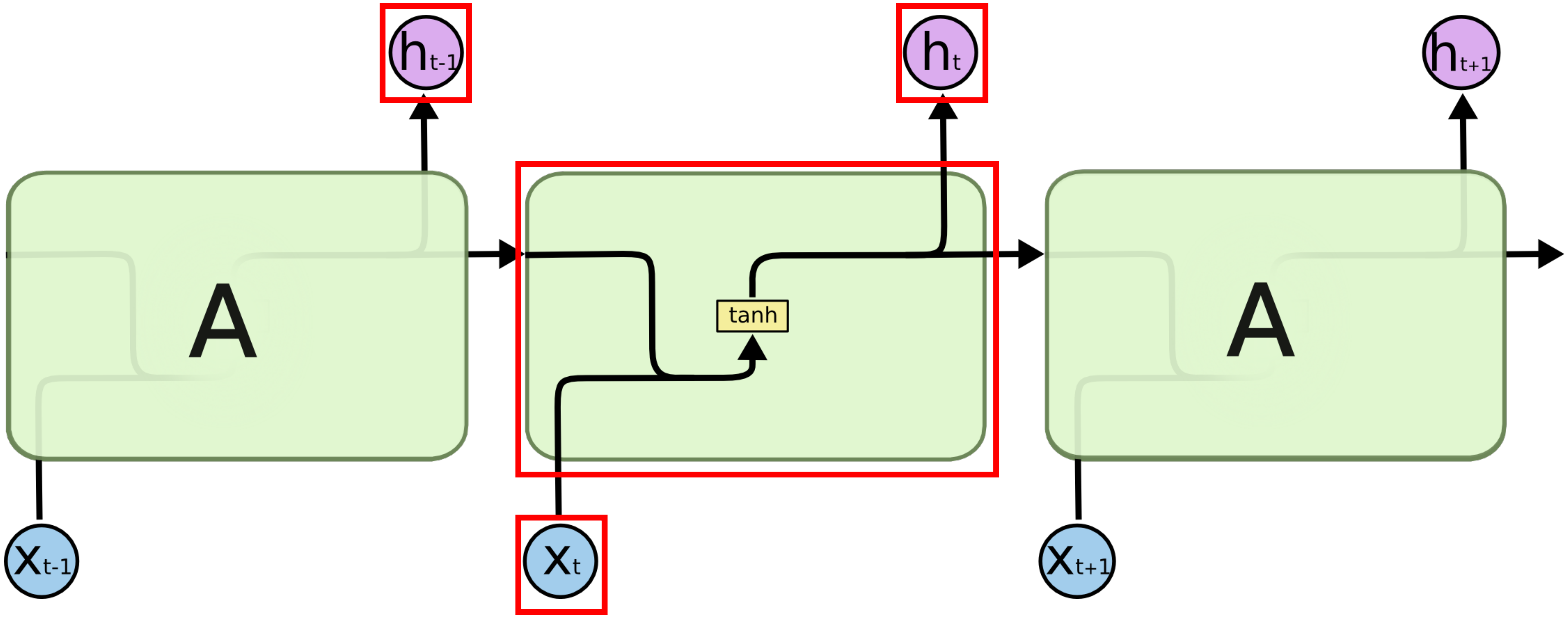
old state

input vector at
some time step

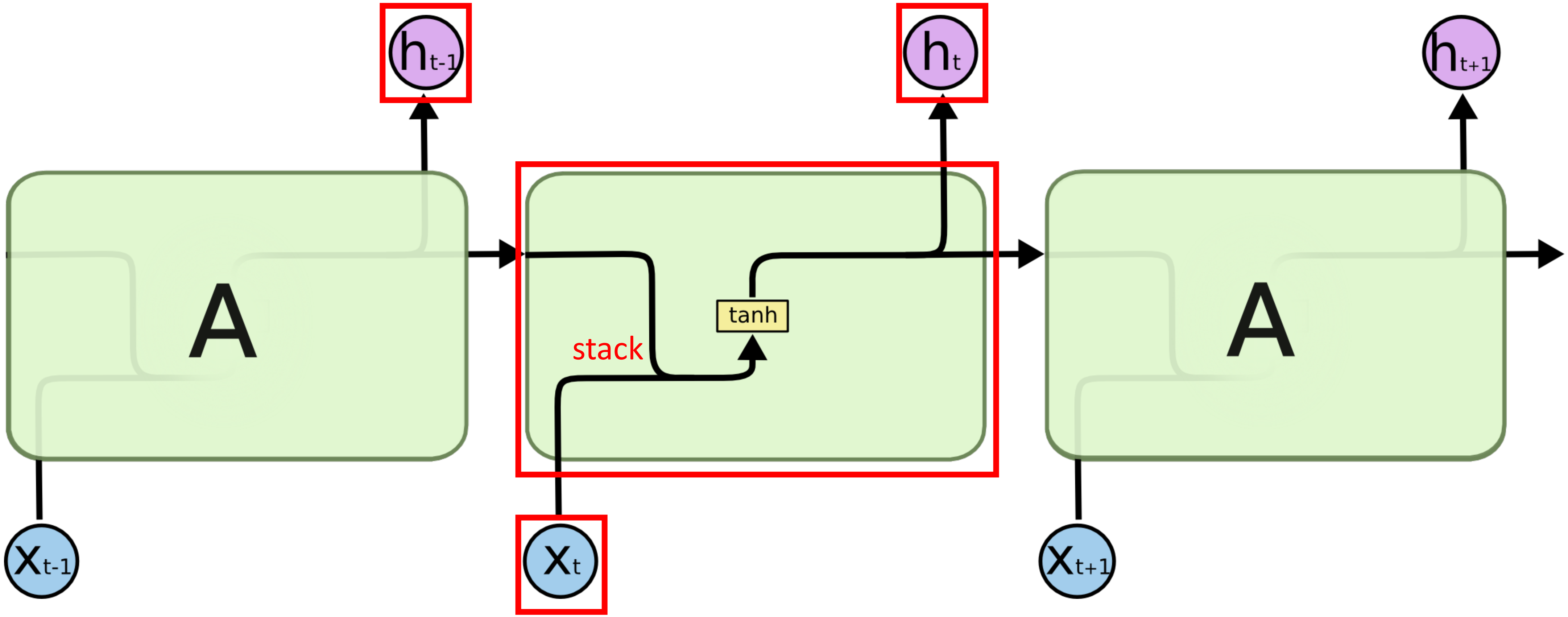
A conventional RNN



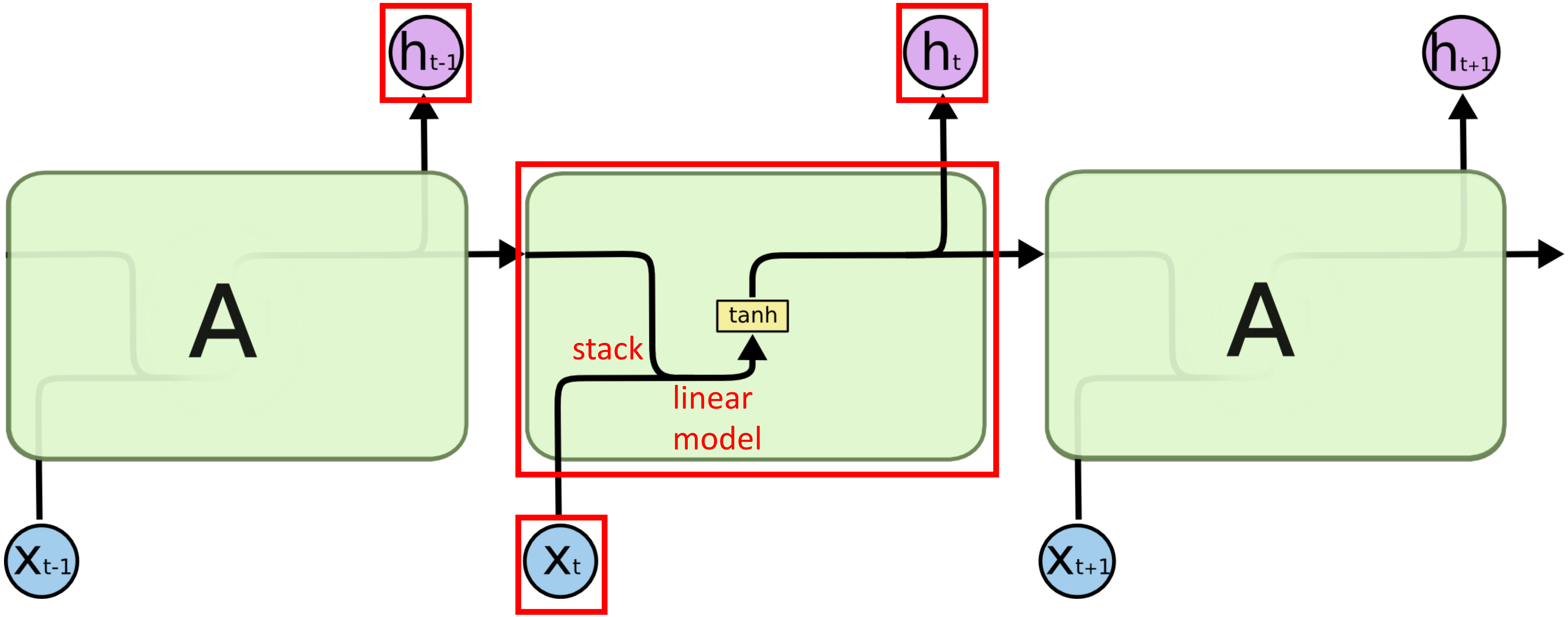
A conventional RNN



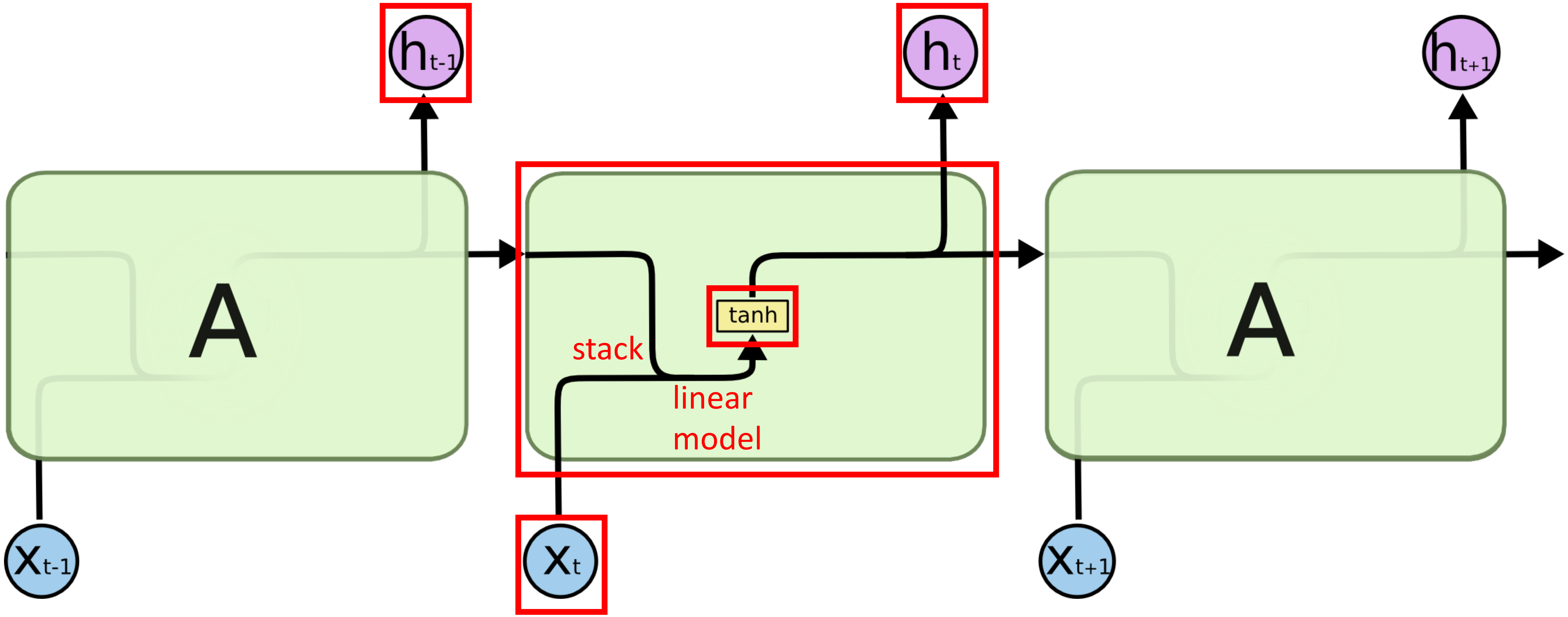
A conventional RNN



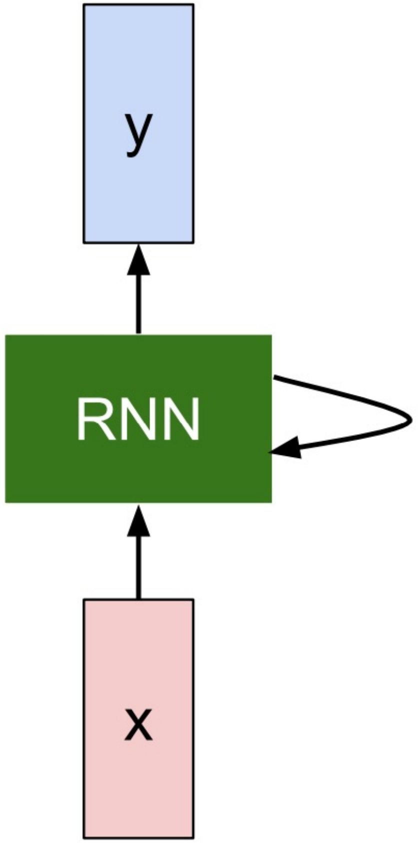
A conventional RNN



A conventional RNN



Recurrent neural networks



$$h_t = f\left[W * \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right] \qquad f = \tanh(\cdot)$$

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

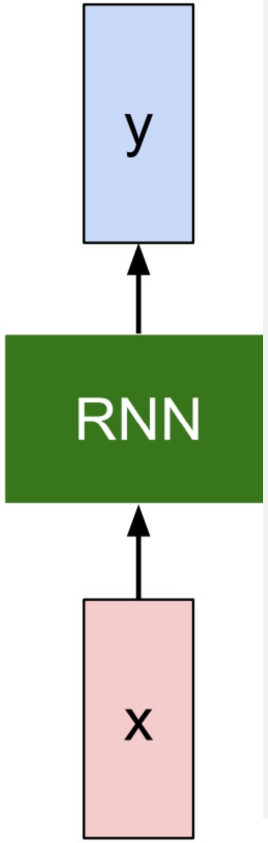
new state

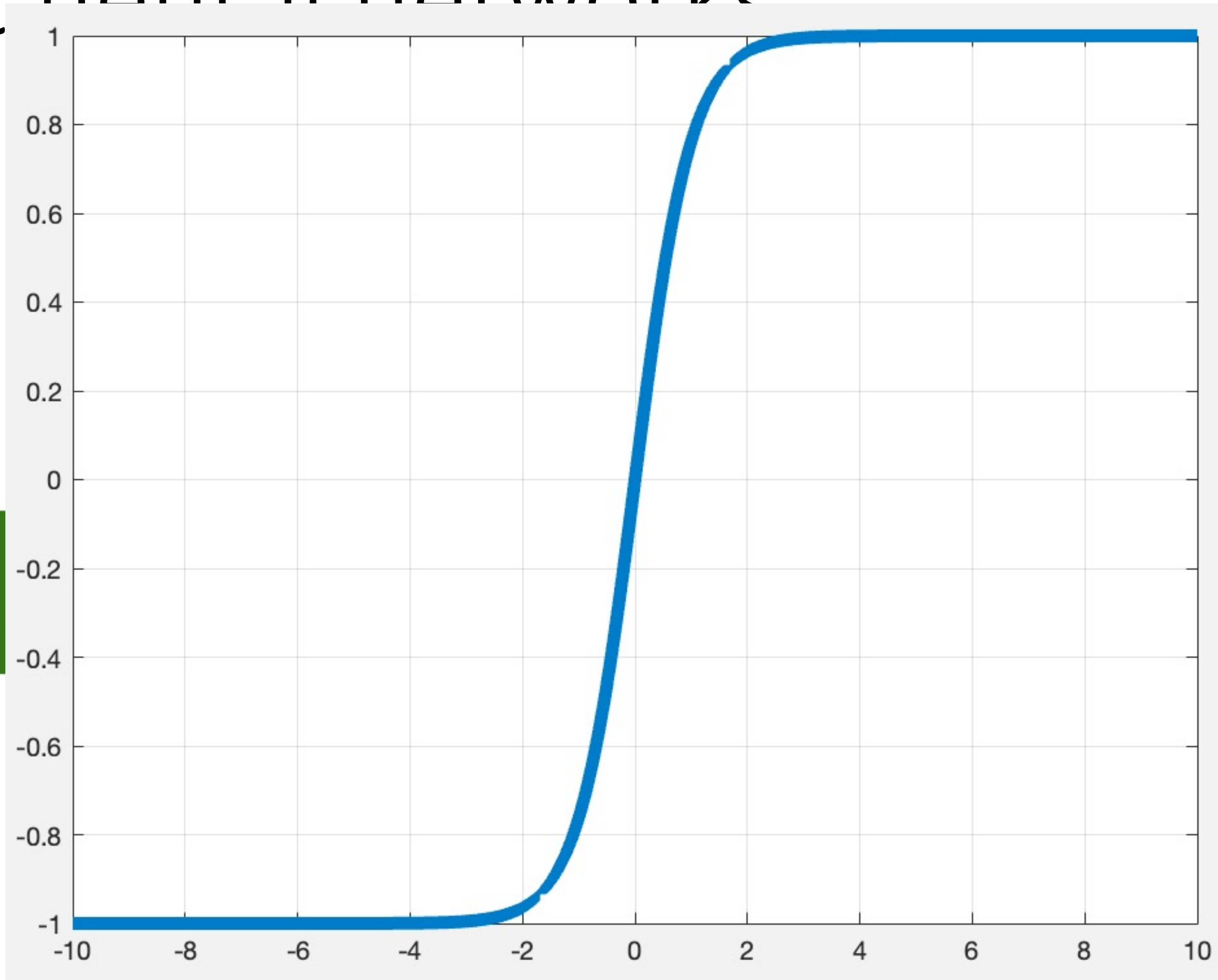
old state

input vector at
some time step

some function
with parameters W

Recurrent neural networks





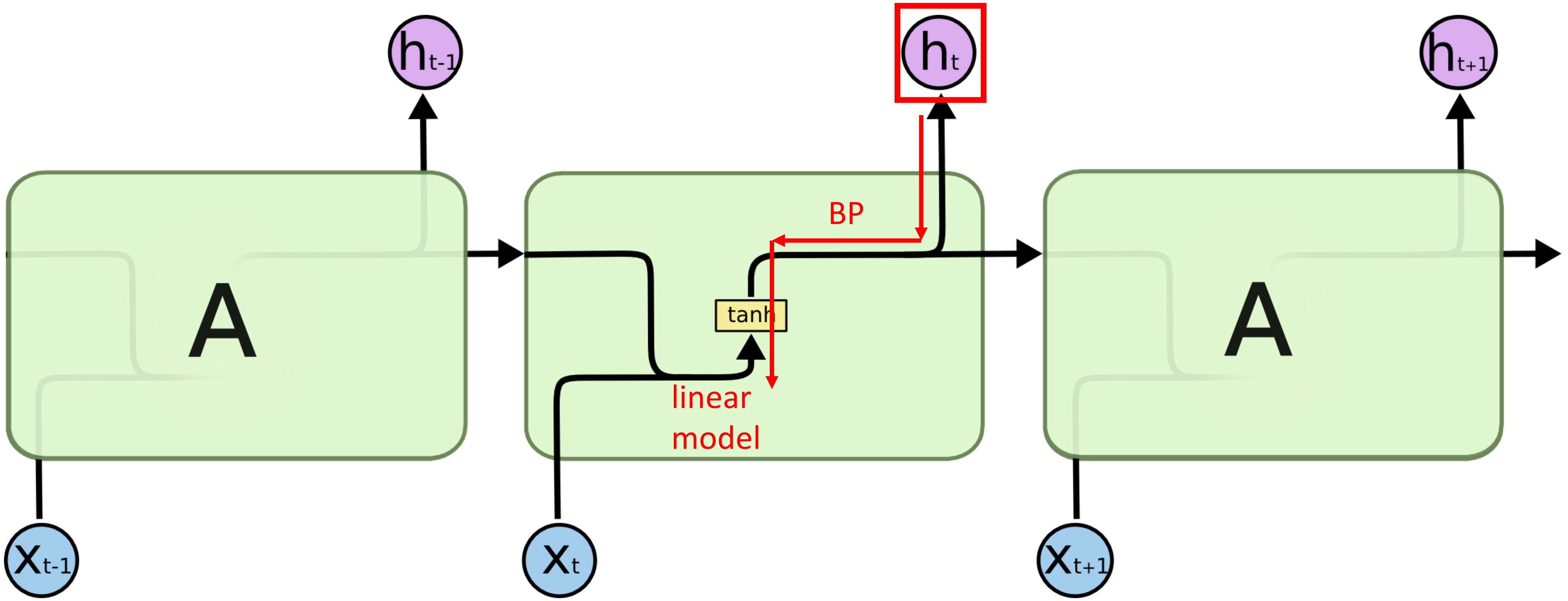
tanh function

$f = \tanh(\cdot)$

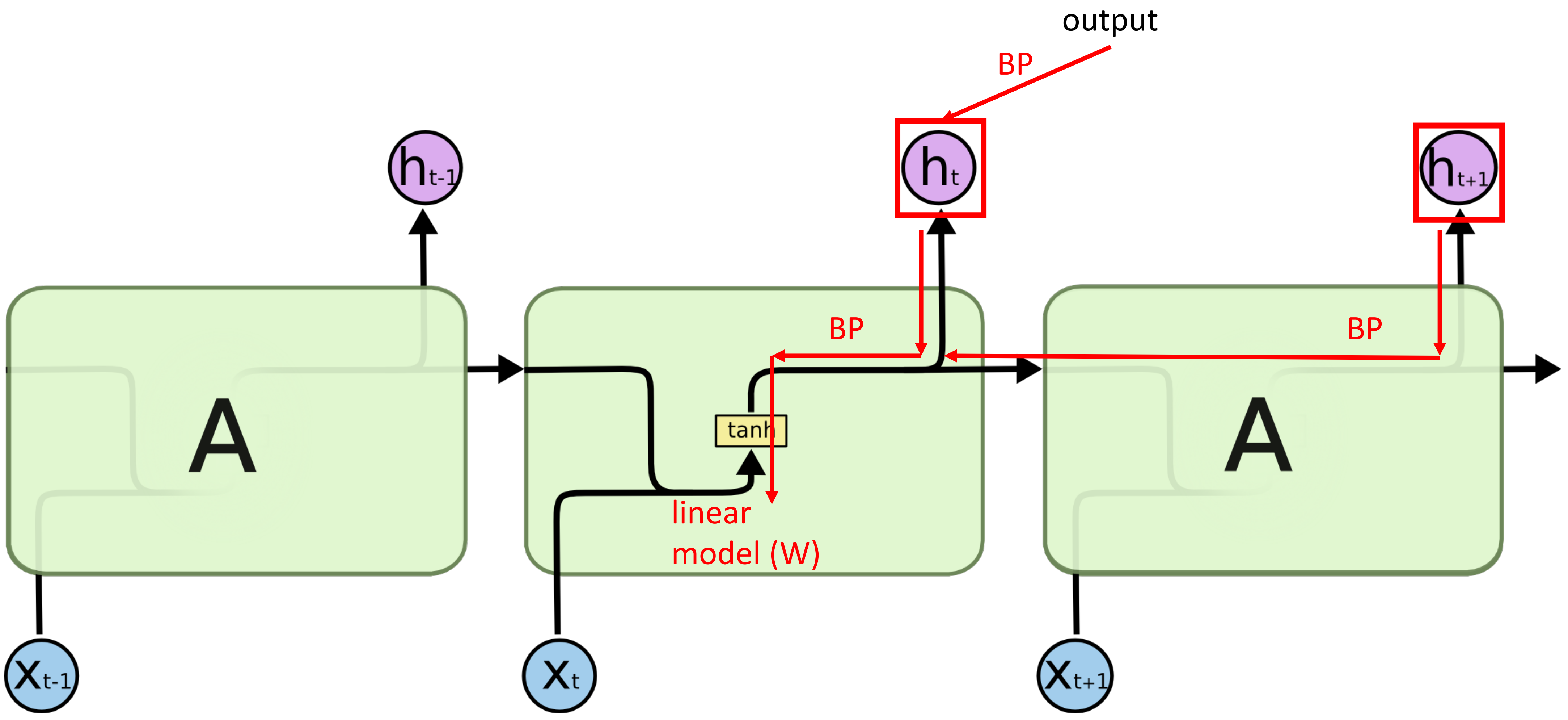
$\boxed{x_t})$

put vector at
ome time step

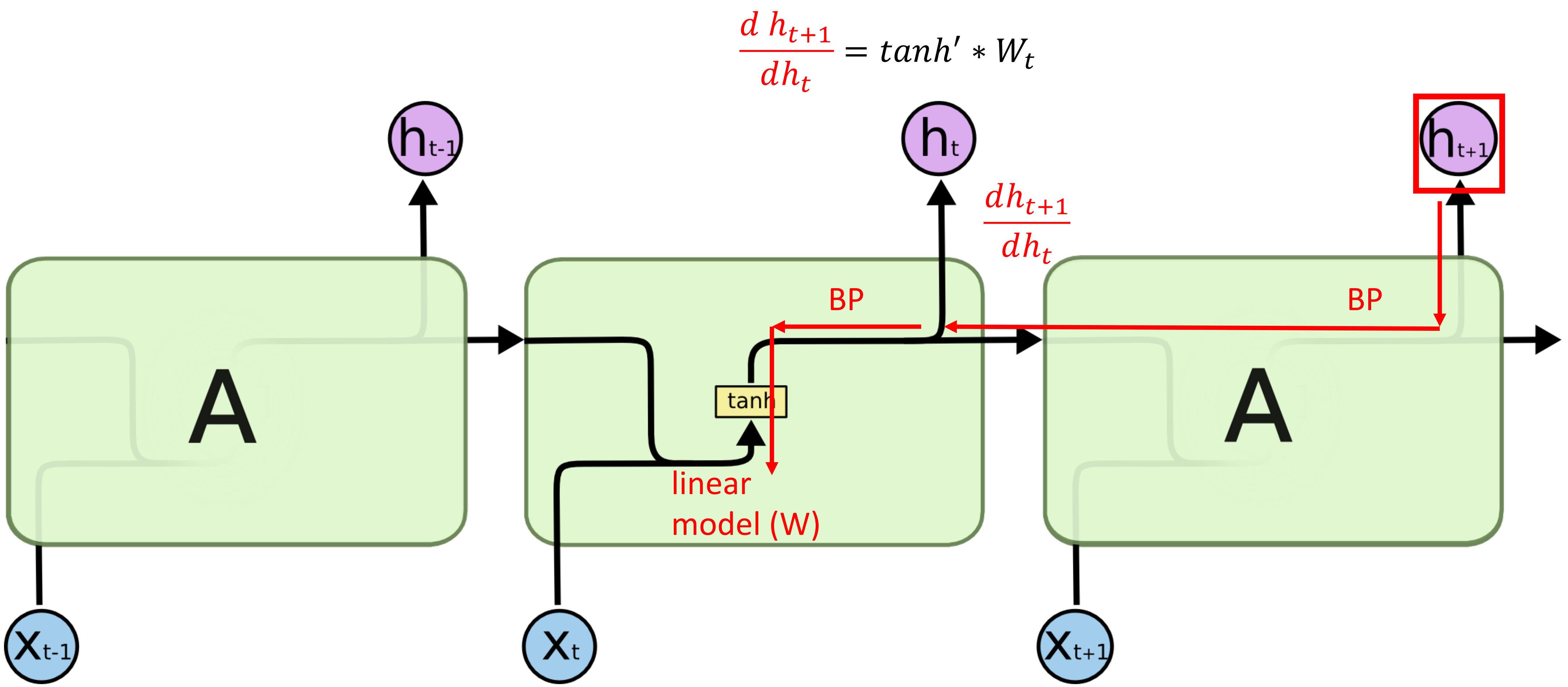
A conventional RNN



A conventional RNN



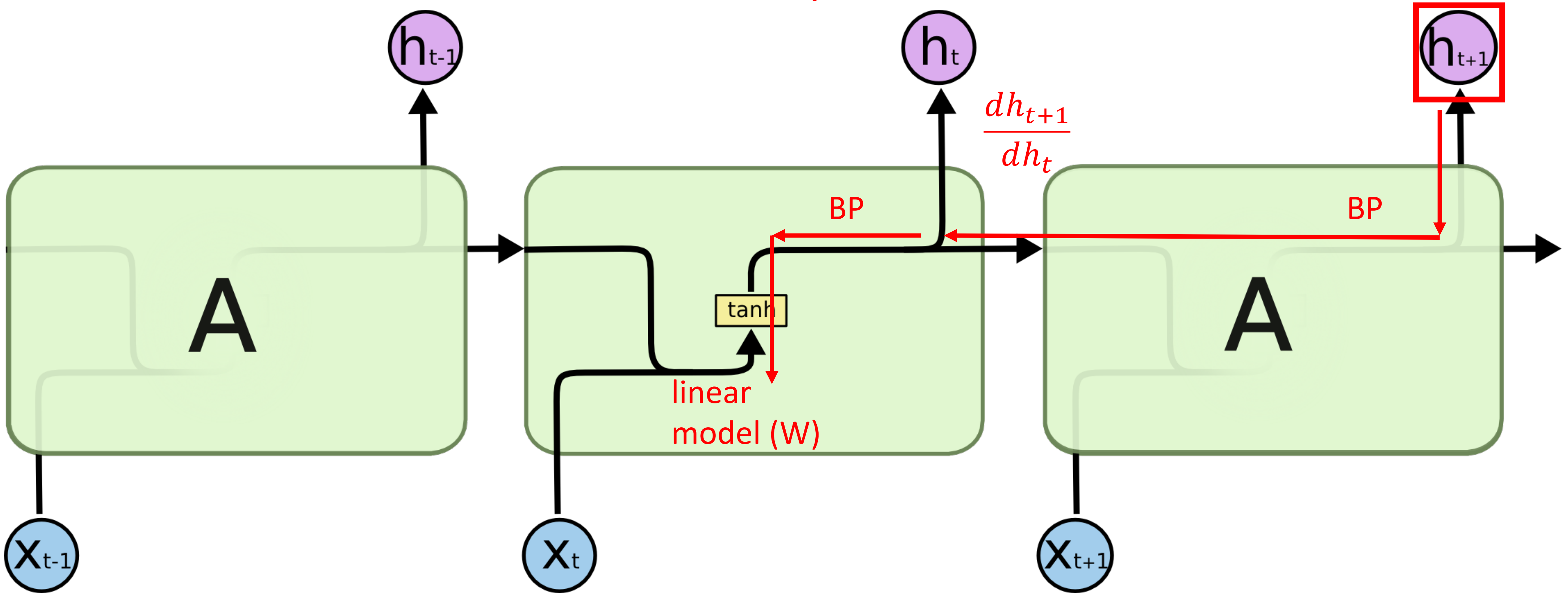
A conventional RNN



A conventional RNN

Q: what issue will we have?

$$\frac{d\ h_{t+T}}{dh_t} = tanh' * tanh' * \cdots * \prod W_t$$

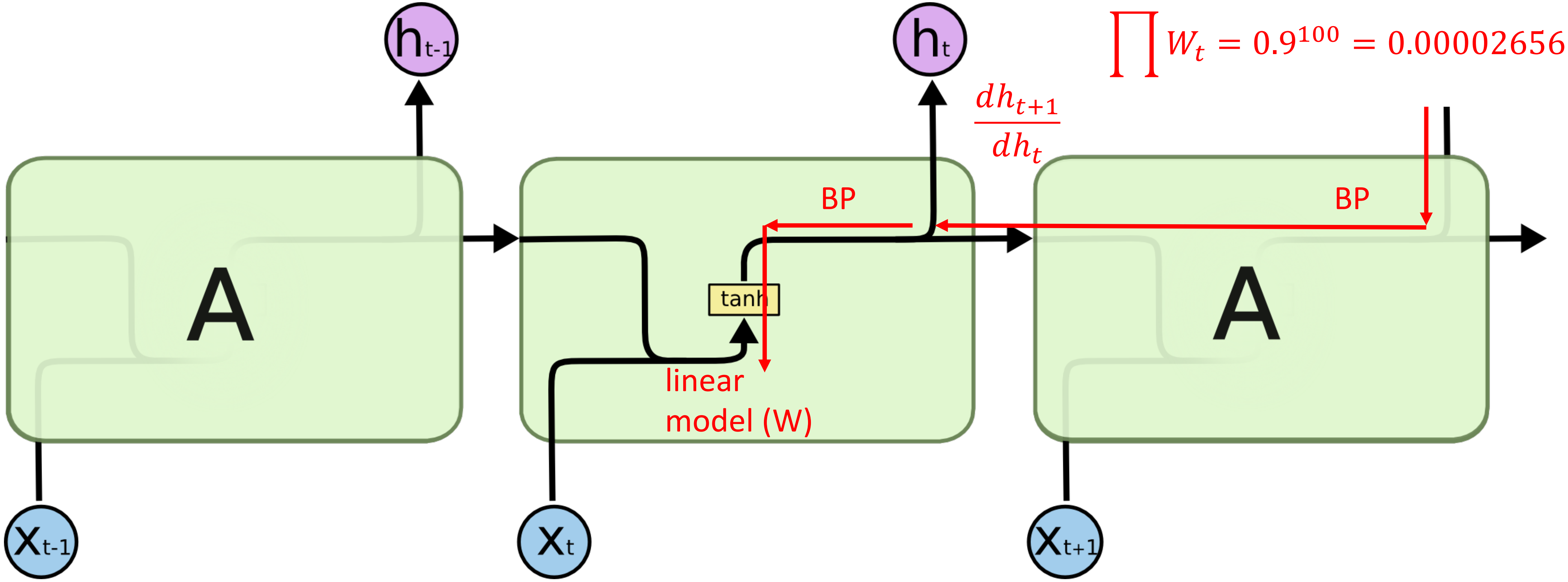


A conventional RNN

Q: what issue will we have?

$$\frac{d\ h_{t+T}}{dh_t} = tanh' * tanh' * \cdots * \prod W_t$$

$$\prod W_t = 0.9^{100} = 0.000026561$$

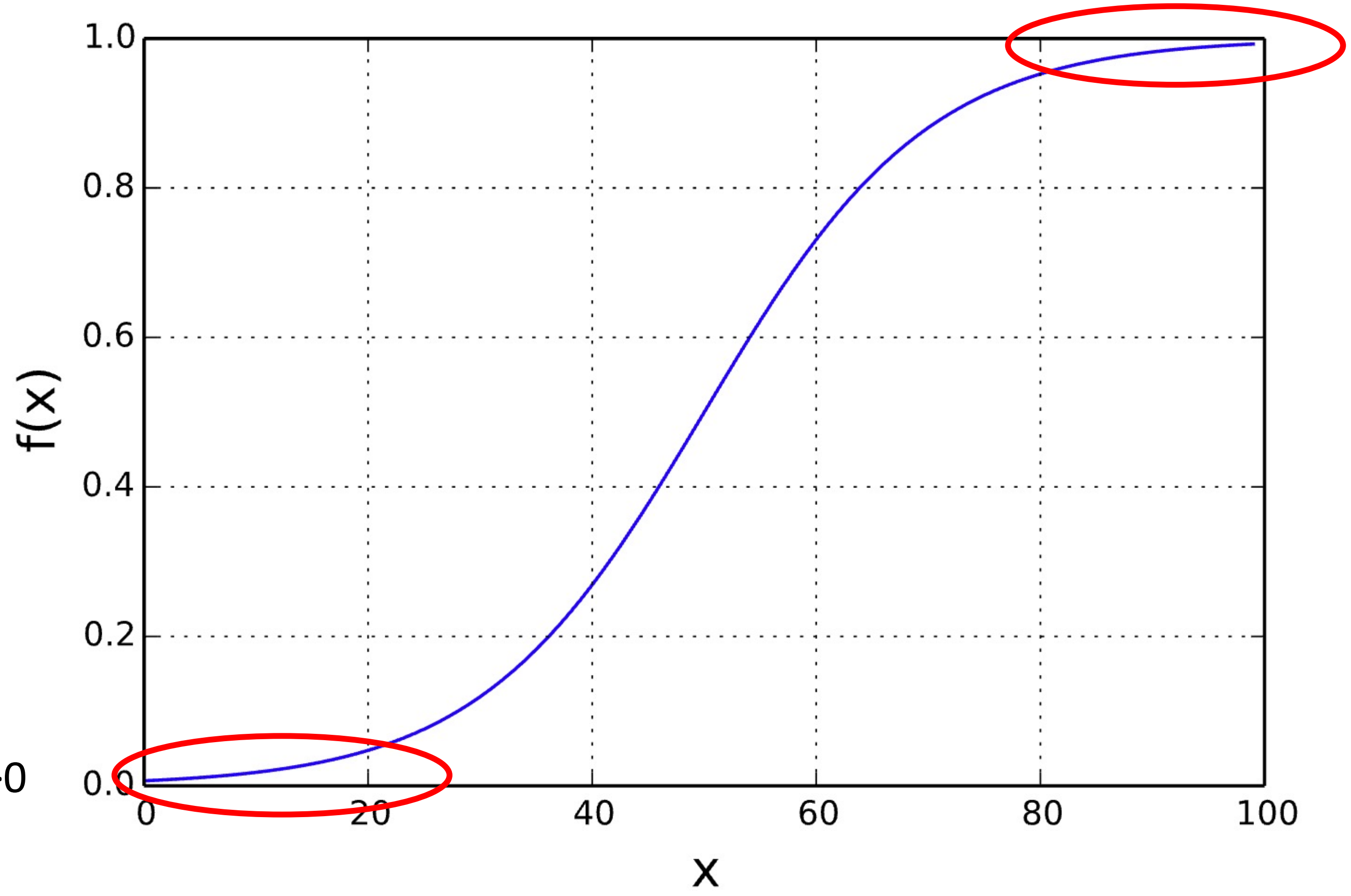


Gradient vanish

$$f_n\left(\dots\left(f_2\left(f_1(x)\right)\right)\right) \rightarrow ?$$

$$f_i \rightarrow x_i$$

$$\frac{dx_n}{dx_1} = \frac{dx_n}{dx_{n-1}} \cdot \dots \cdot \frac{dx_2}{dx_1} \cdot \frac{dx_1}{dx}$$



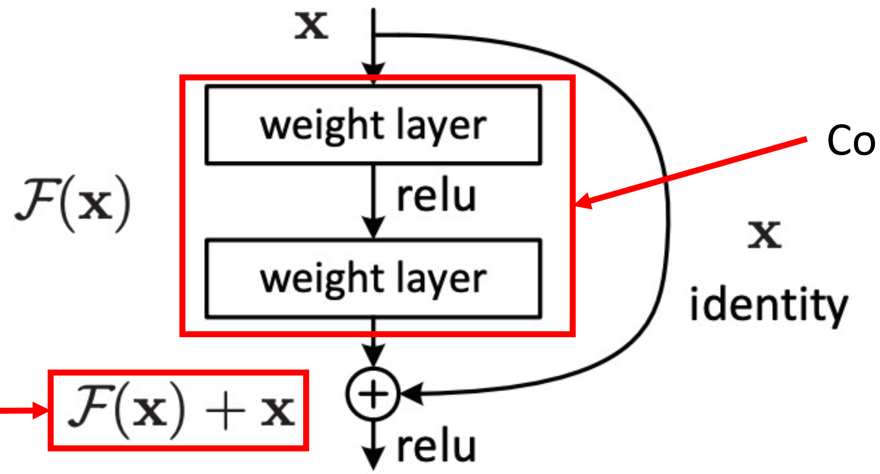
gradients->0

gradients->0

Sigmoid function

ResNet: shortcut connection

Conv layers

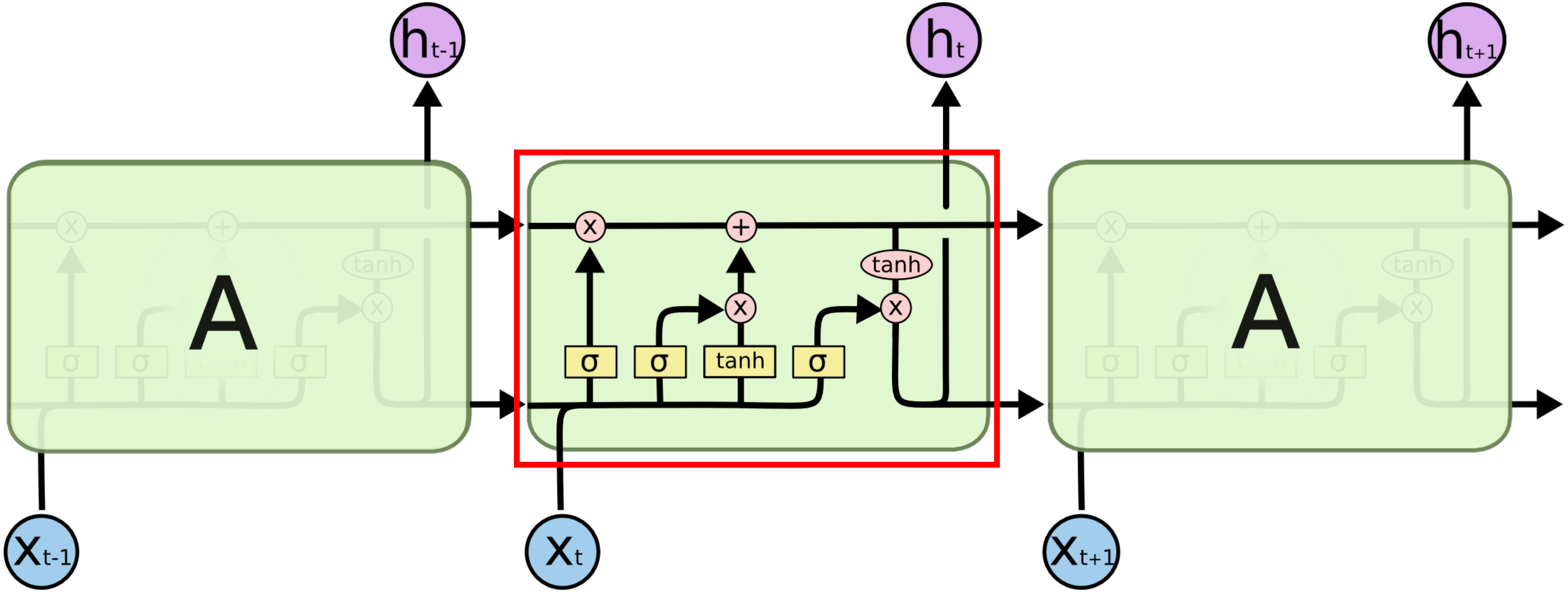


Addition operation

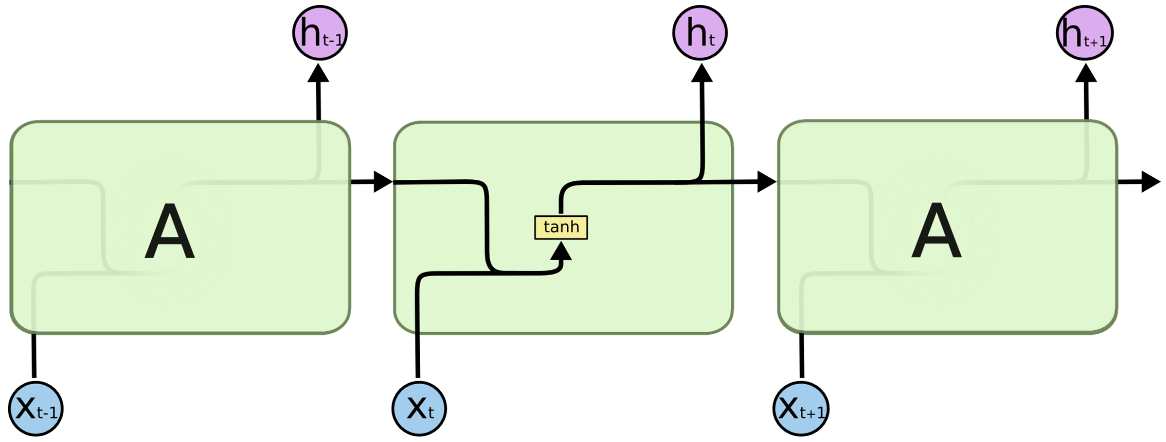
implication: same dimension

Figure 2. Residual learning: a building block.

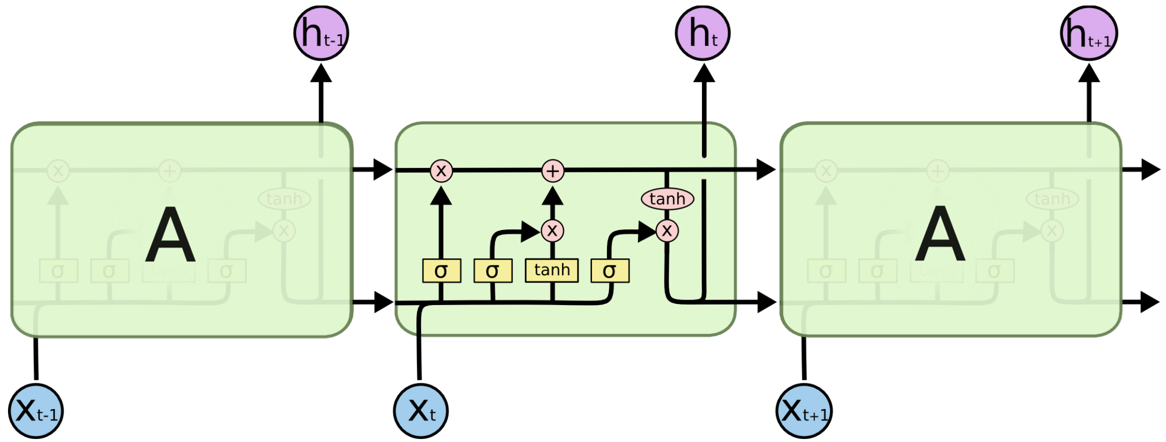
Long-short term memory (LSTM) networks



LSTM

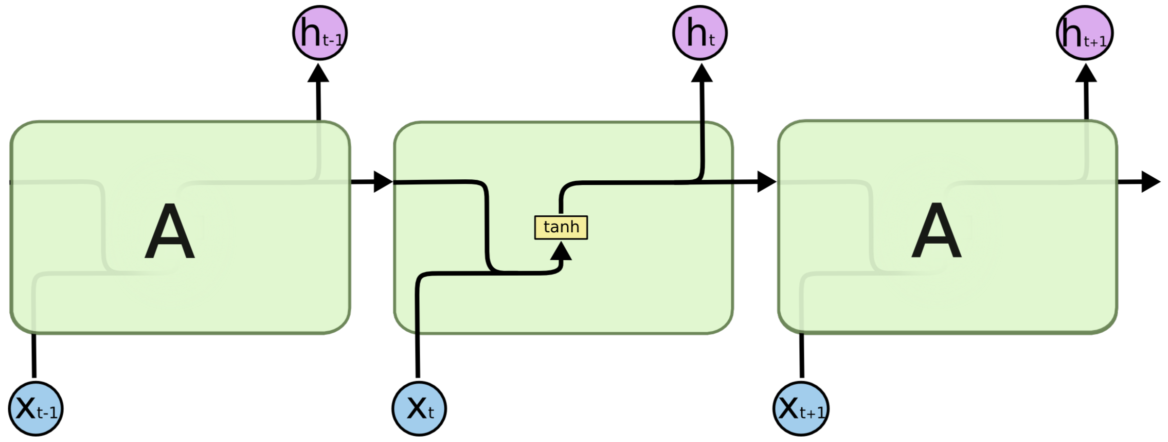


h_{t-1}
h_t
h_{t+1}
A
tanh
A
x_{t-1}
x_t
x_{t+1}

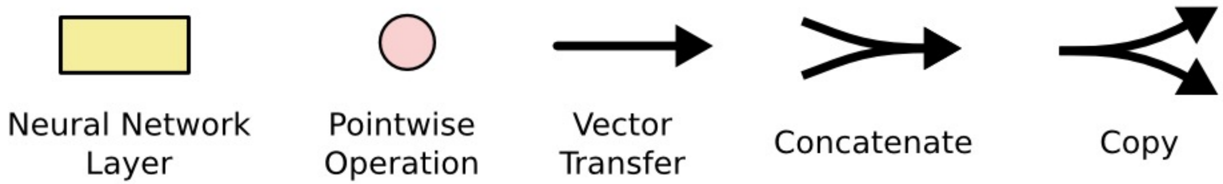


h_{t-1}
h_t
h_{t+1}
A
tanh
σ
σ
tanh
σ
A
x_{t-1}
x_t
x_{t+1}

LSTM

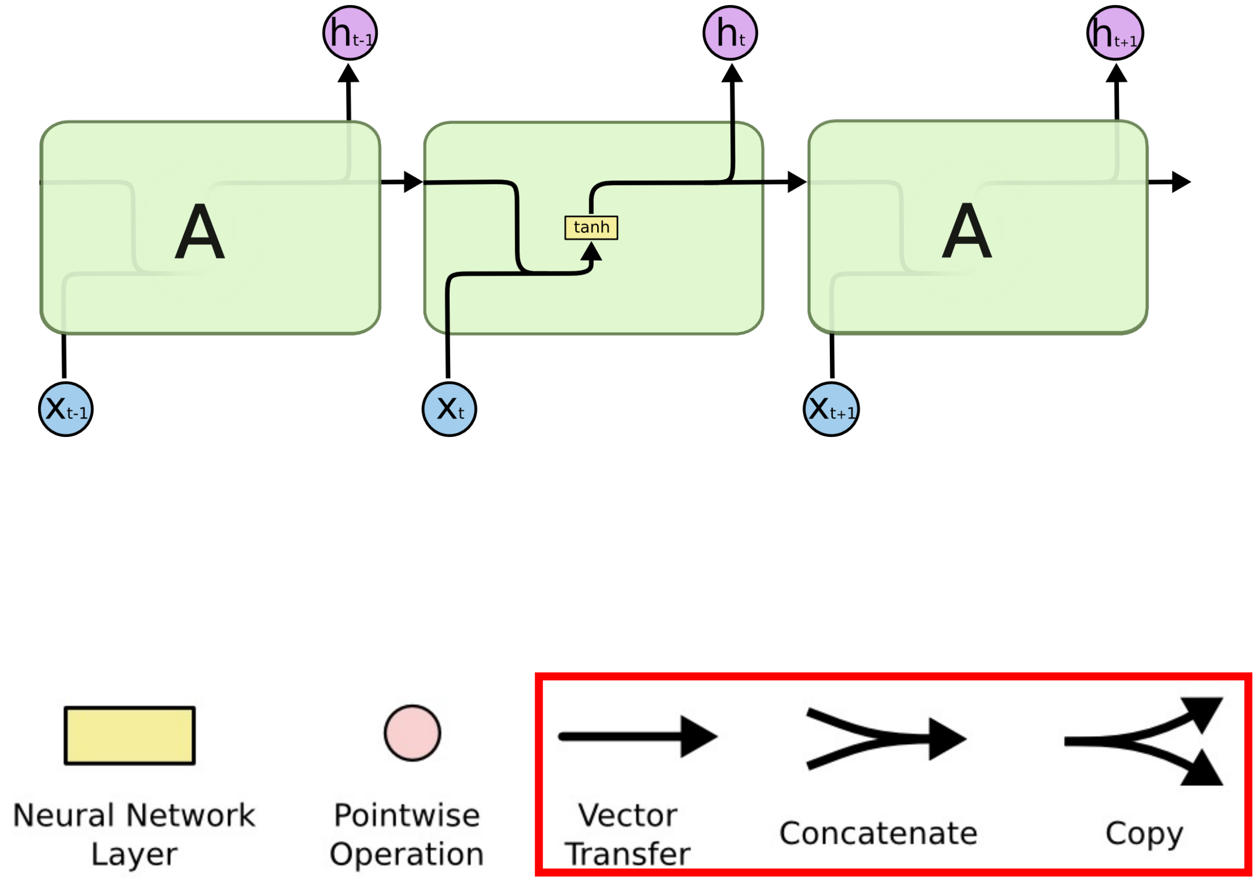


A
tanh
A
h_{t-1}
h_t
h_{t+1}
X_{t-1}
X_t
X_{t+1}

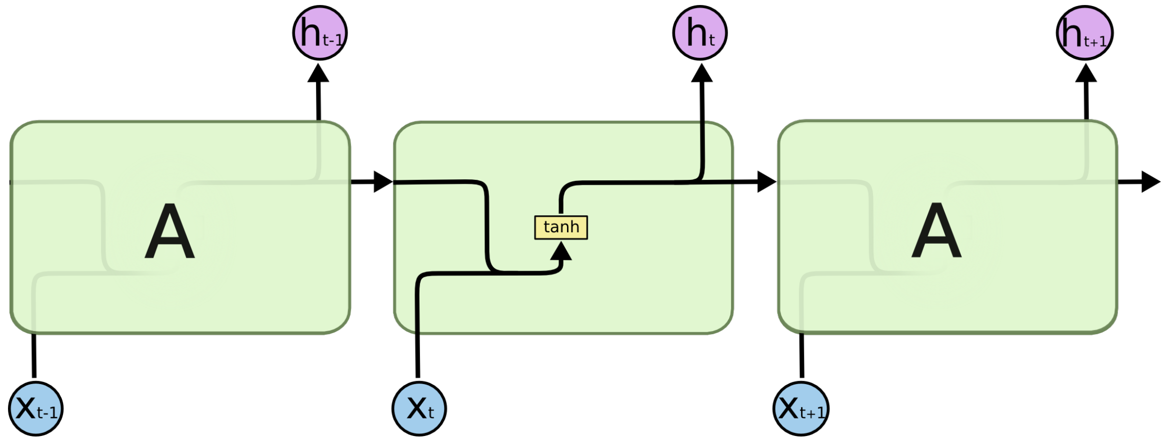


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

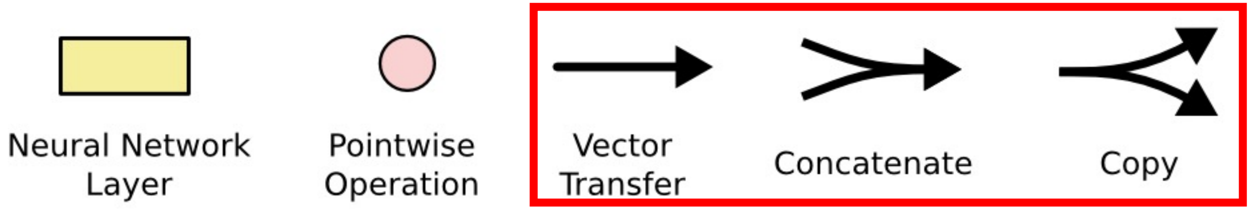
LSTM


ht-1
ht
ht+1
A
tanh
A
Xt-1
Xt
Xt+1
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

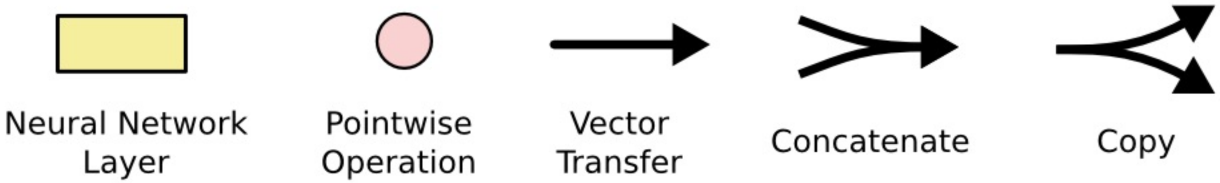
LSTM

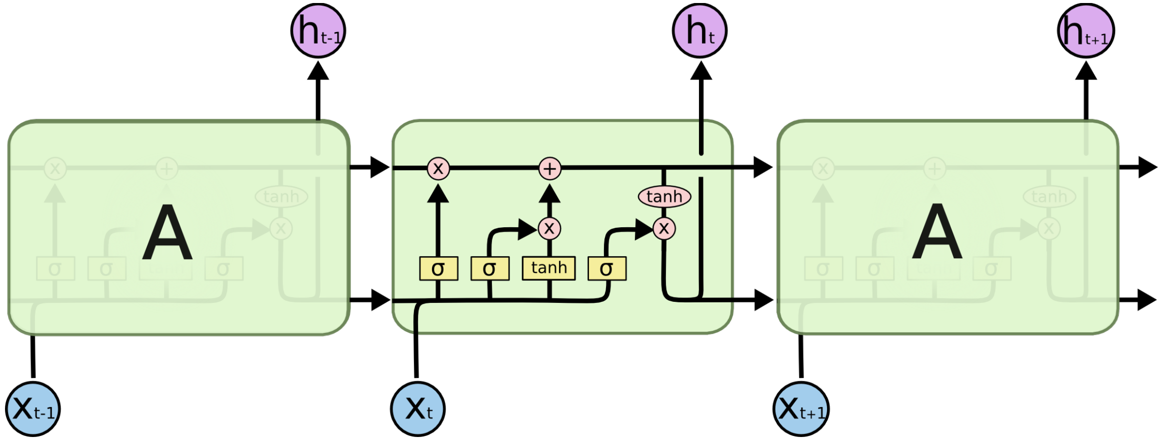


Composition of all $\{h_t\}$

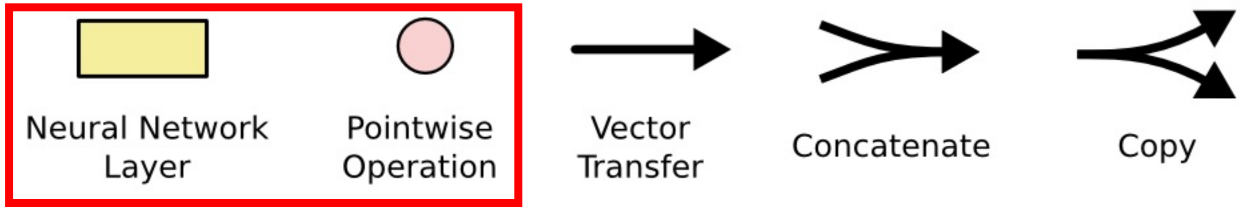


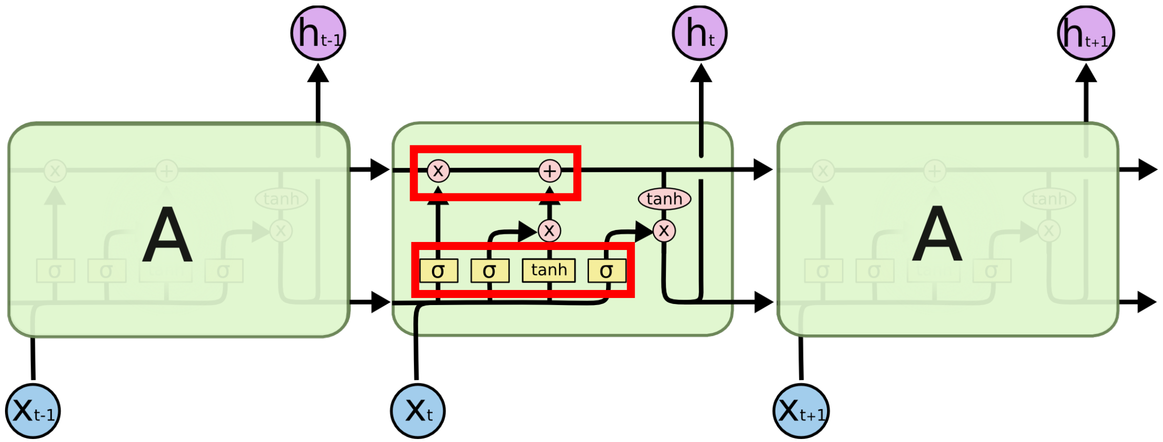
LSTM


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

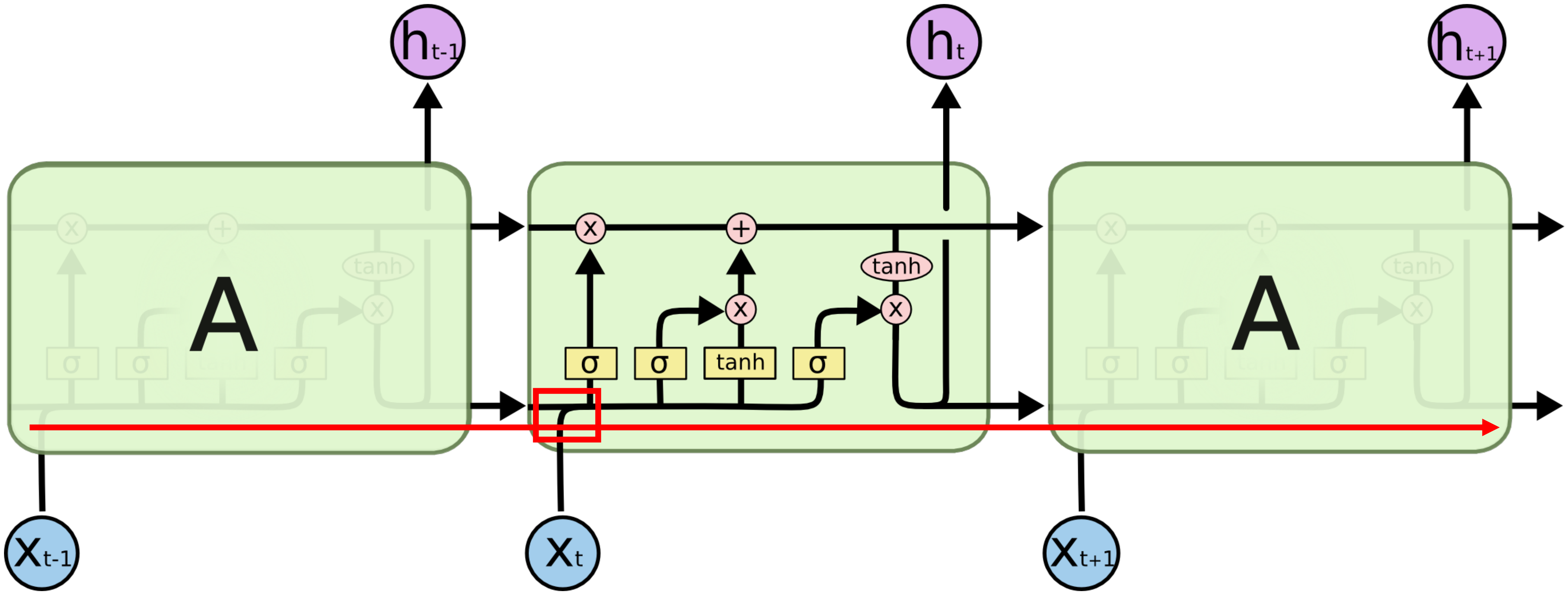

h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}

LSTM

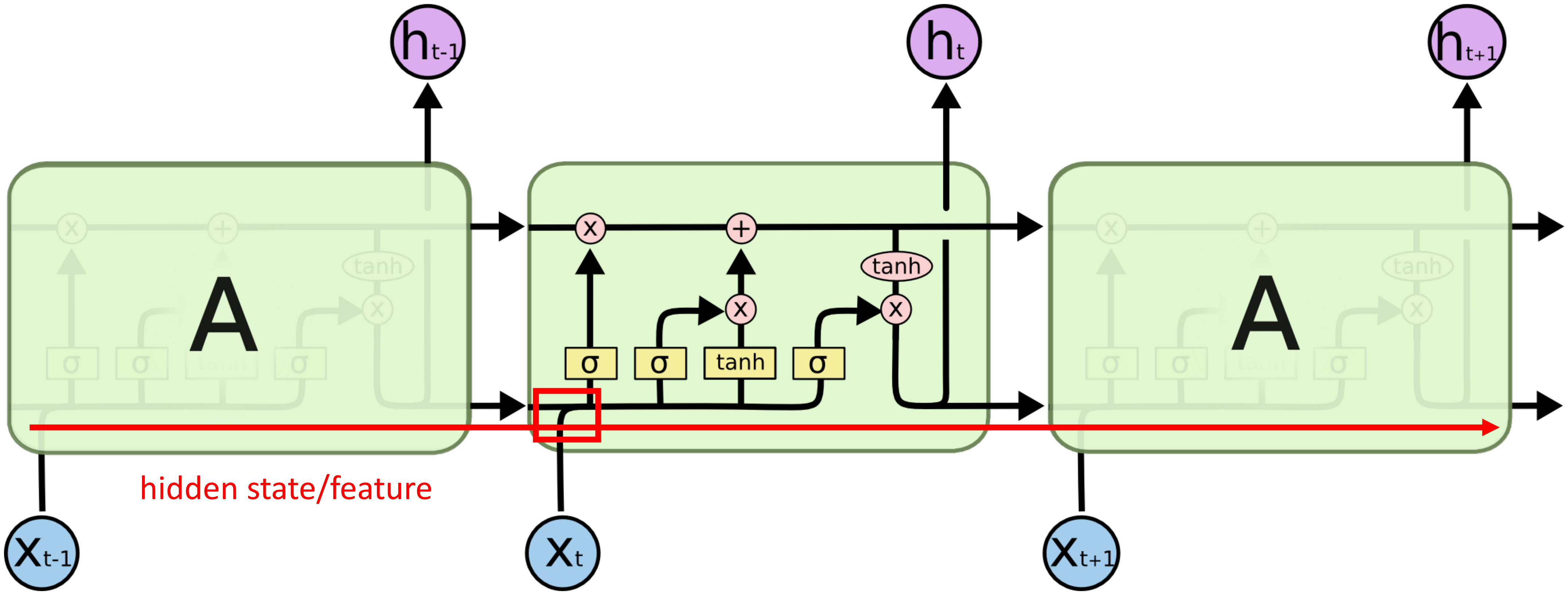

Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy


A
A
tanh
σ
σ
tanh
σ

LSTM

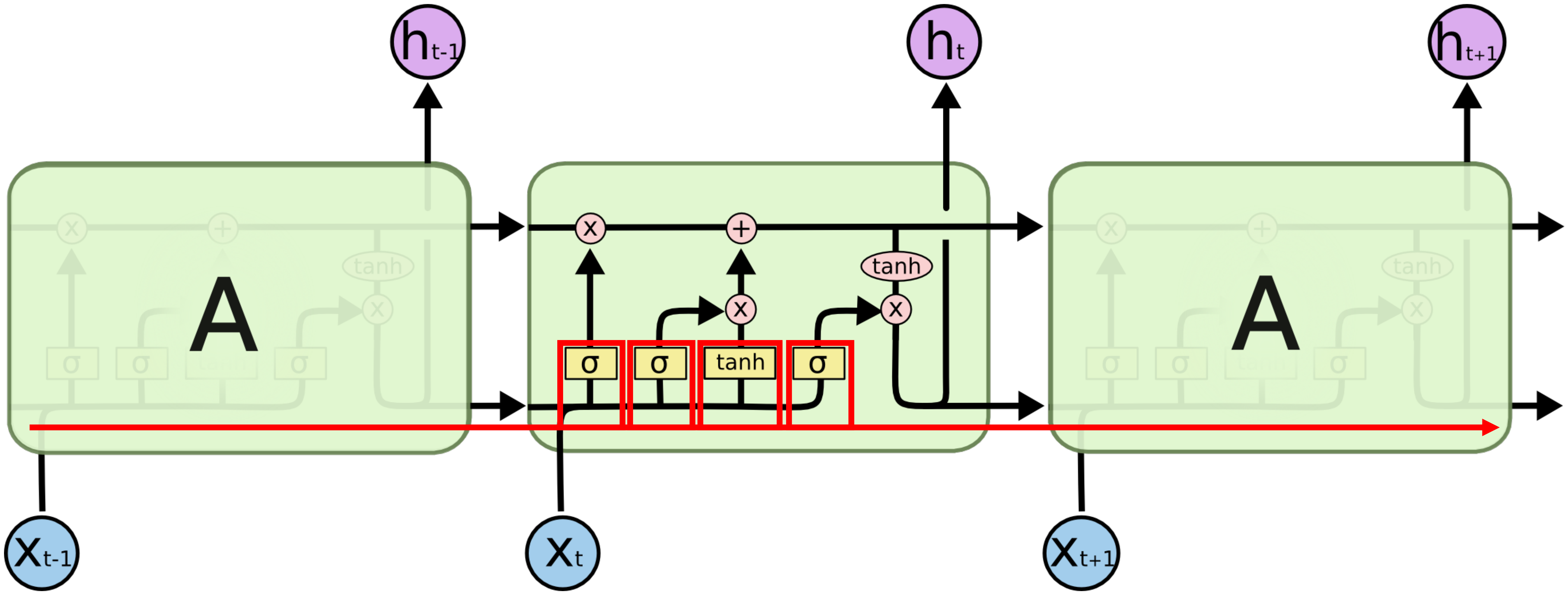


LSTM

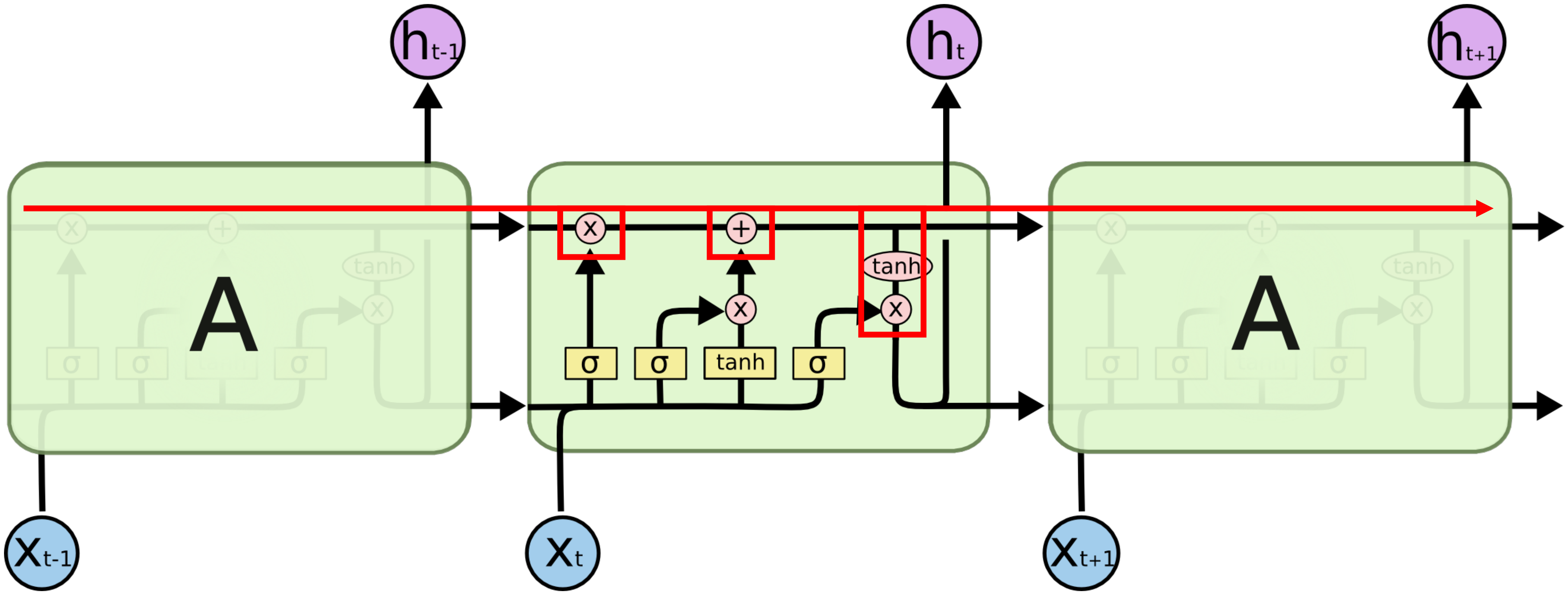


h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
x
+
x
x
hidden state/feature
x_{t-1}
x_t
x_{t+1}

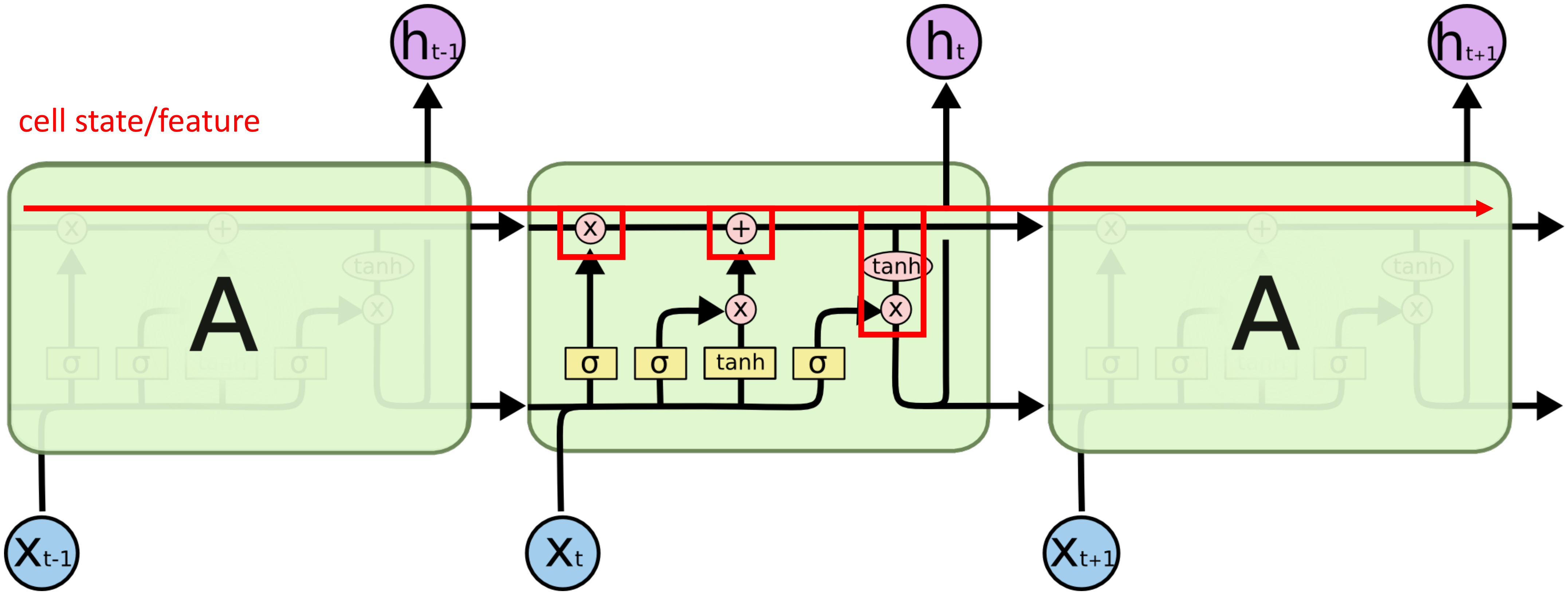
LSTM



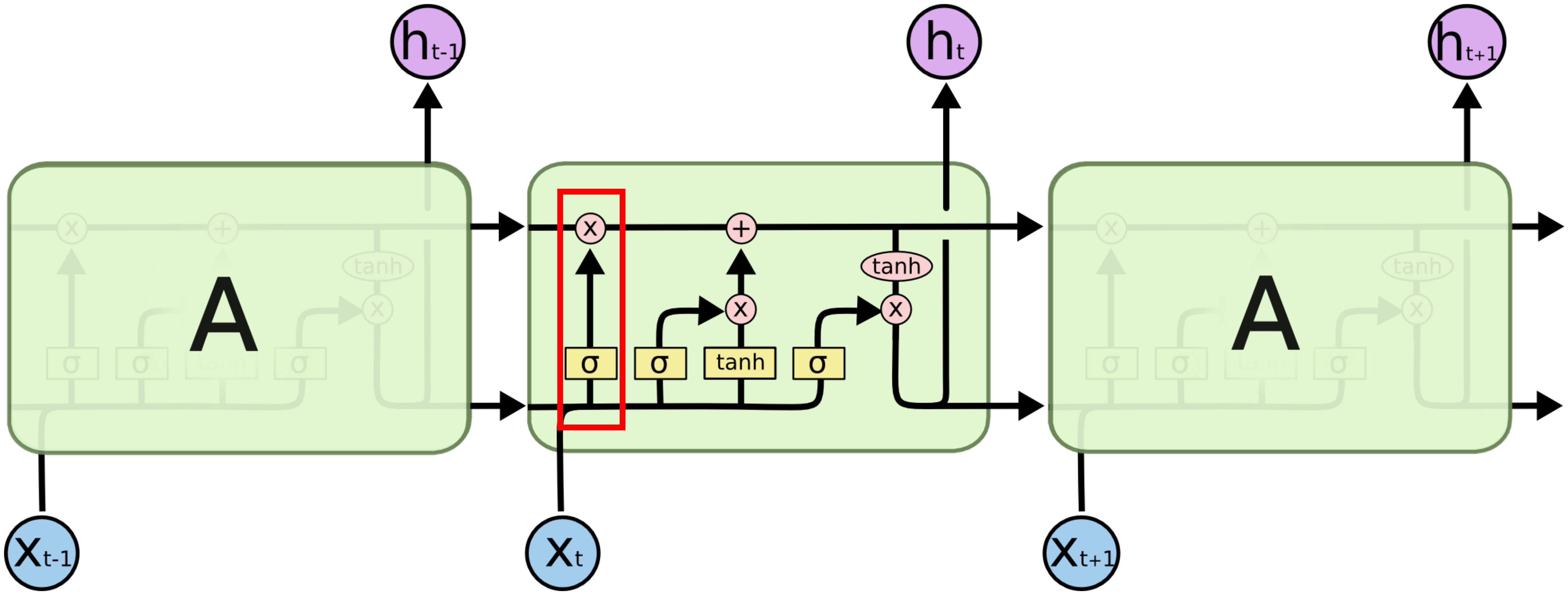
LSTM



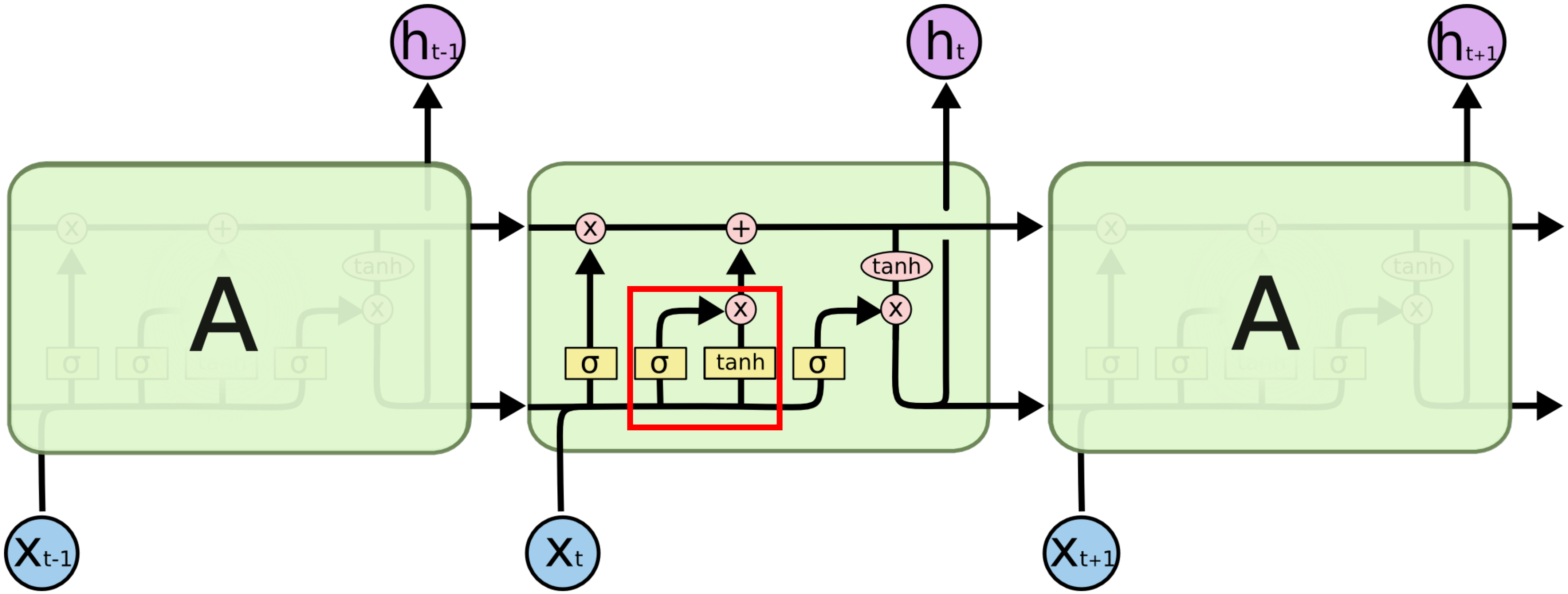
LSTM



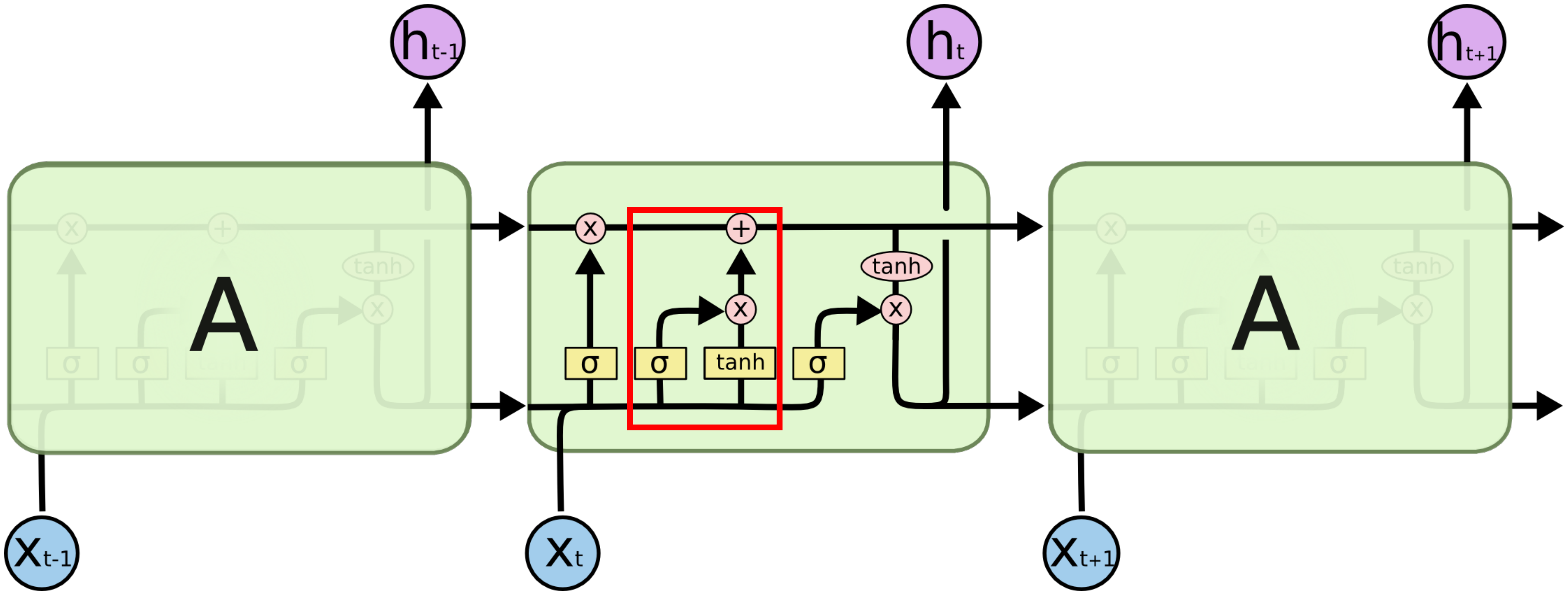
LSTM



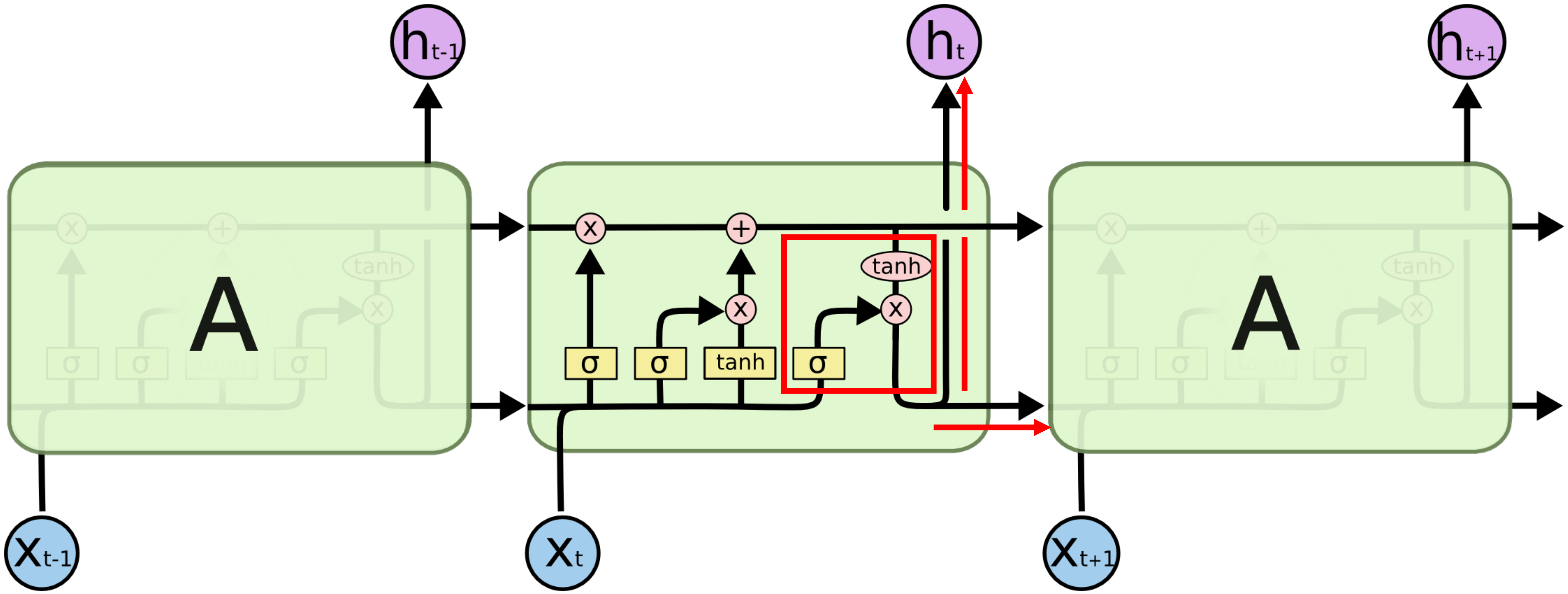
LSTM



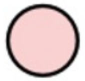
LSTM



LSTM

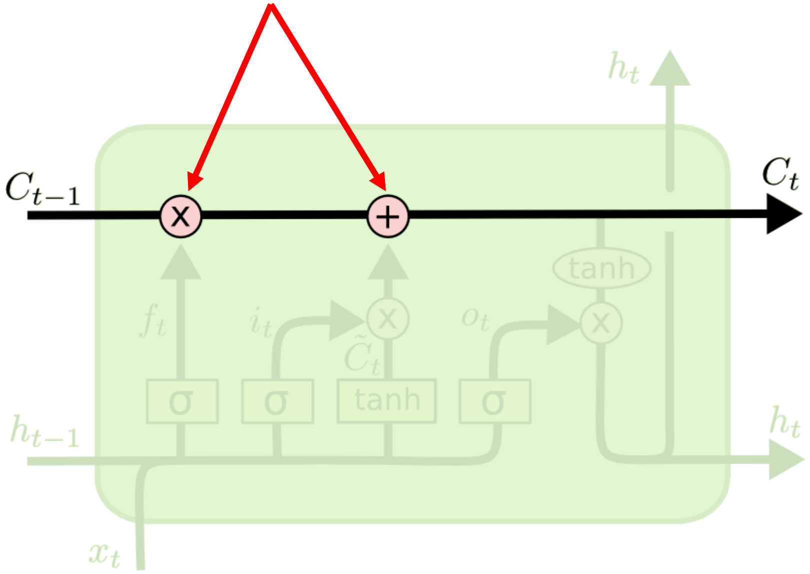


LSTM

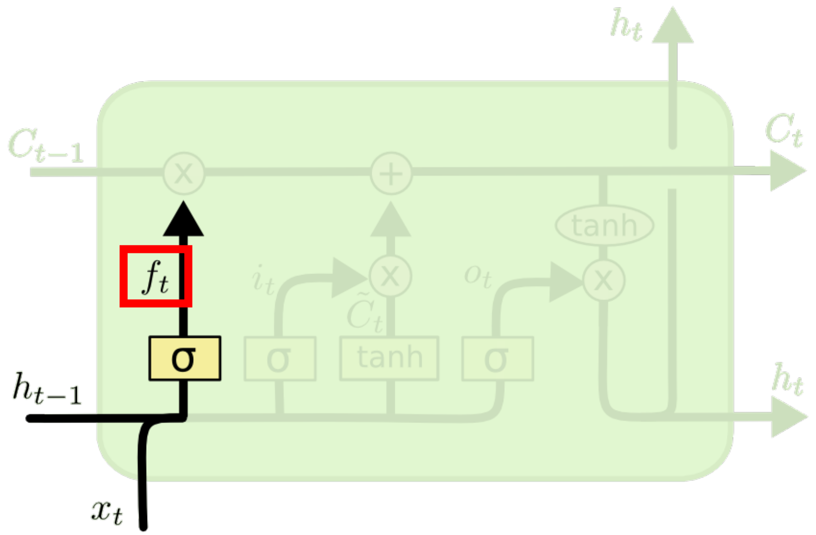


Pointwise Operation

Elementwise multiplication/addition

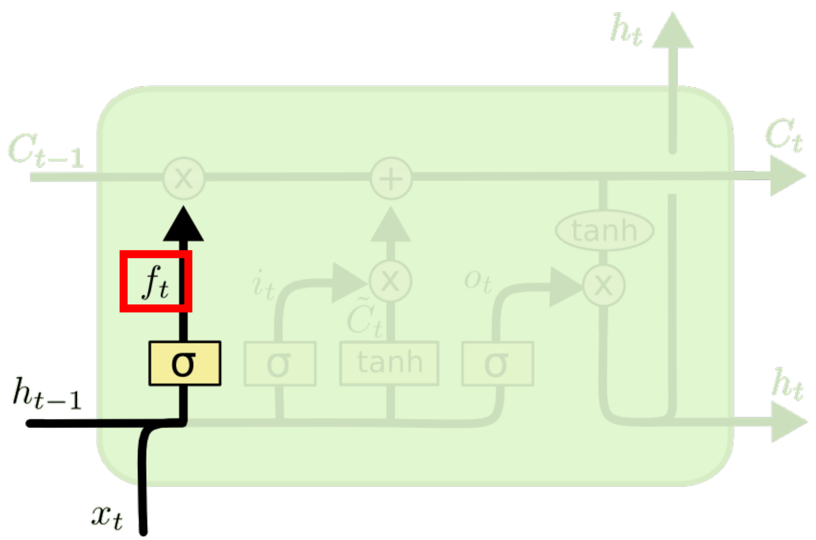


LSTM

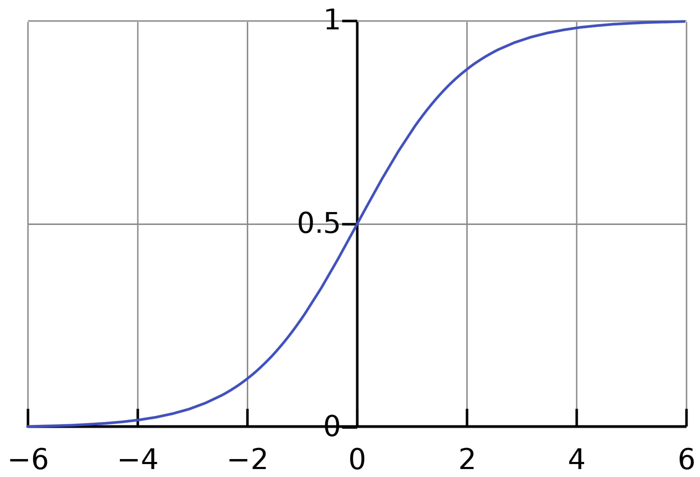


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

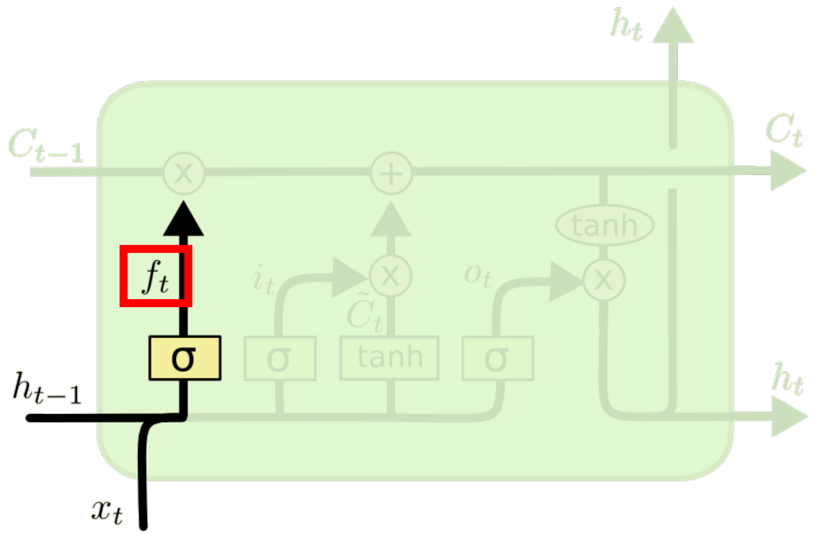
LSTM



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

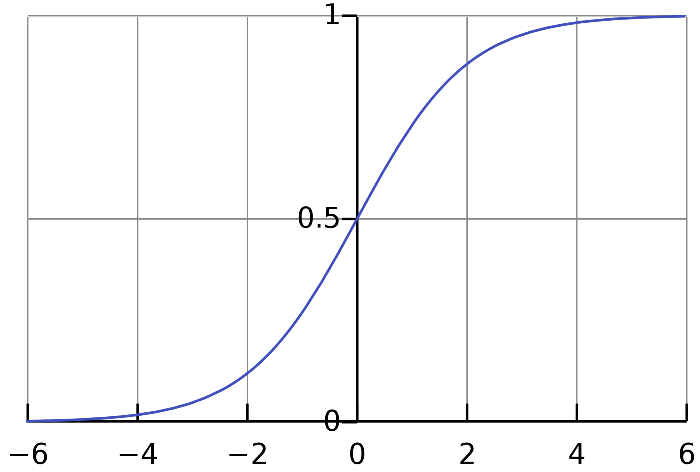


LSTM

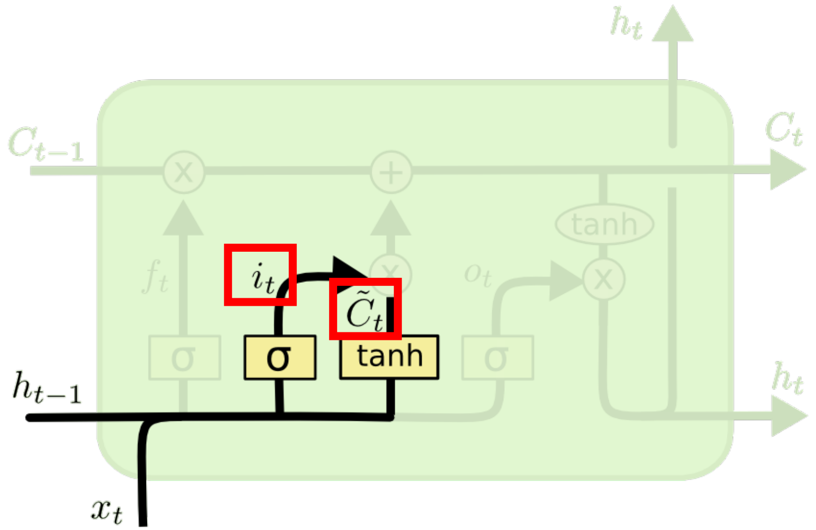


Forget gate:
Whether to erase cell

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



LSTM



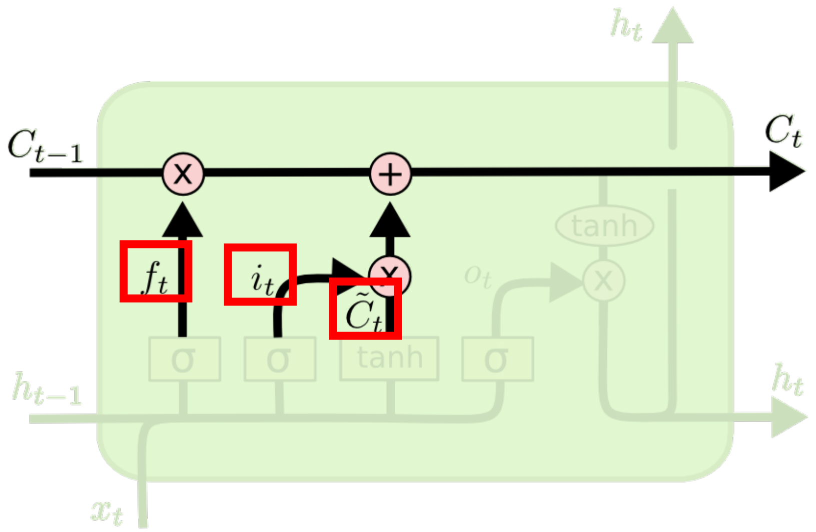
Input gate

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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

cell input activation vector

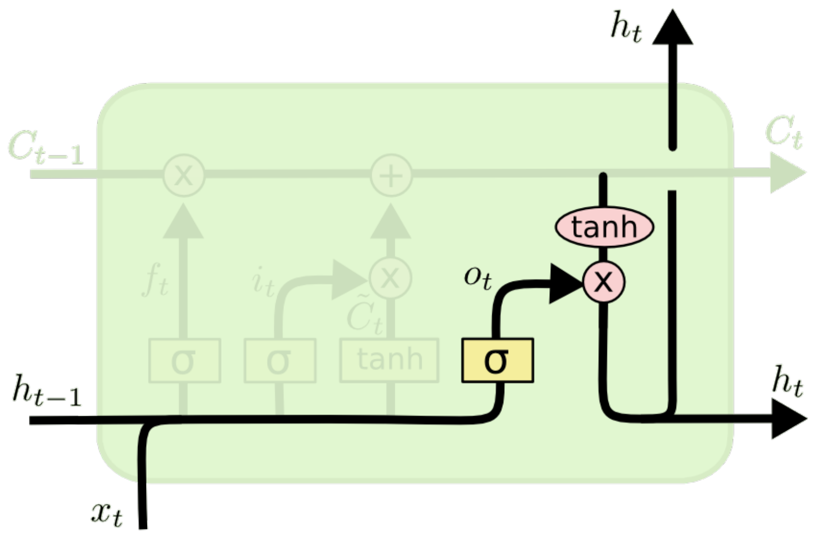
LSTM



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

Cell state

LSTM

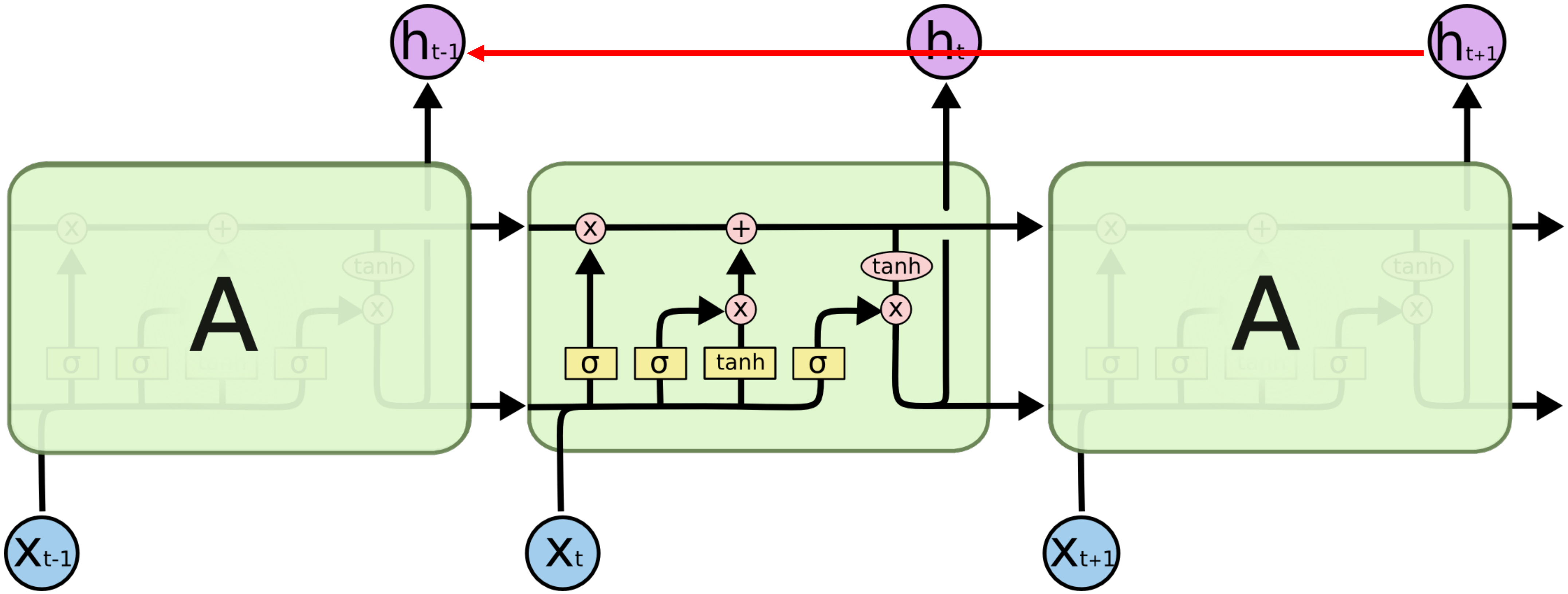


Output gate

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

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

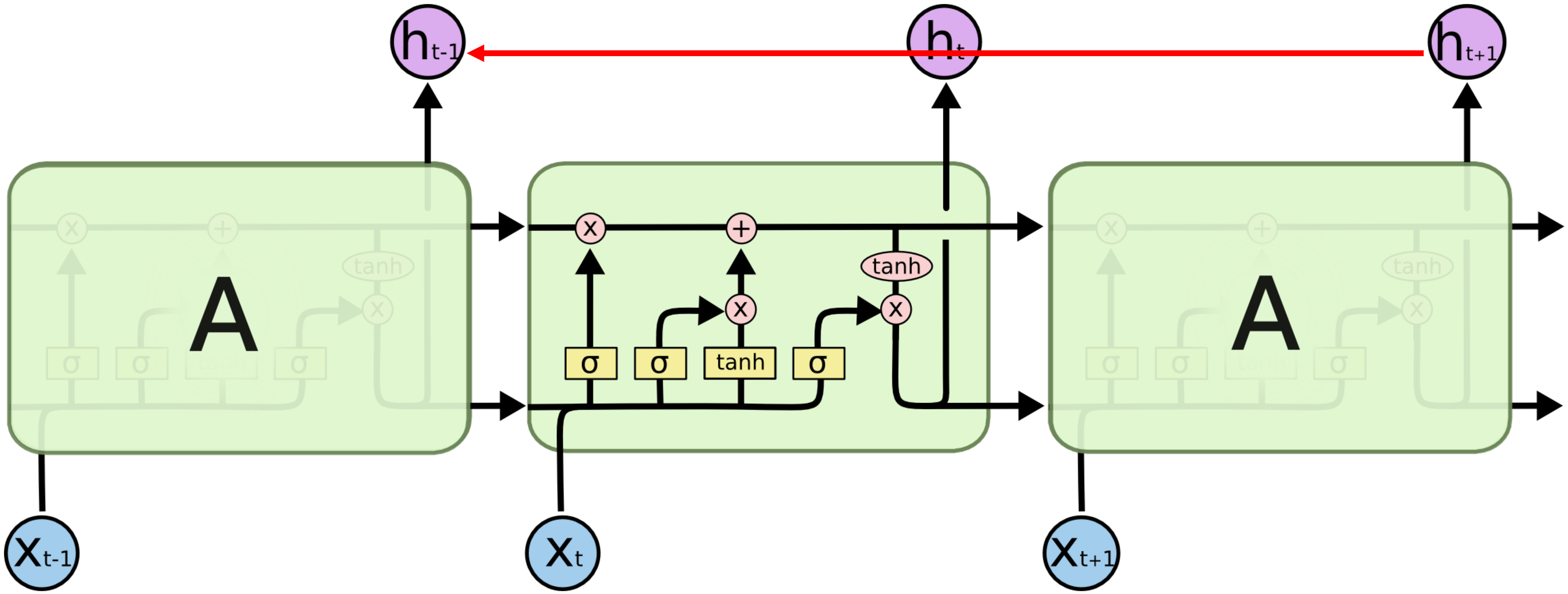
LSTM



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

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

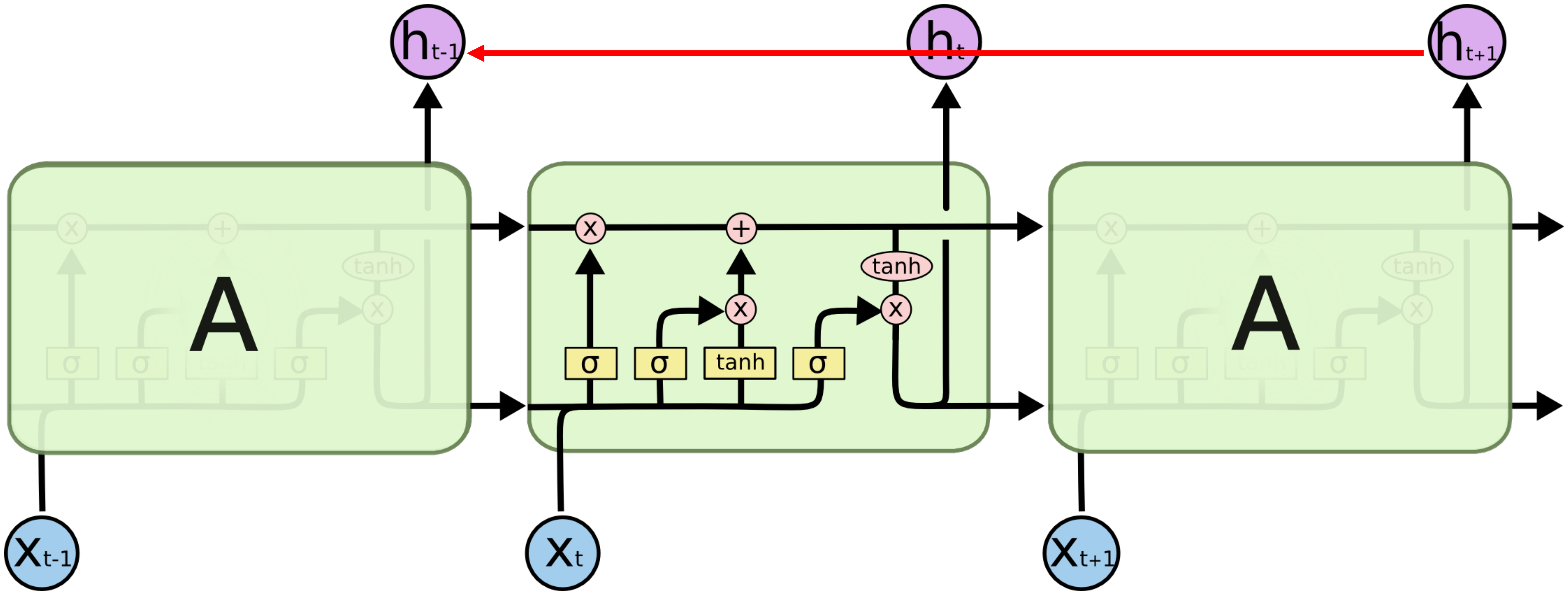
LSTM



$$C_t = f_t * \boxed{C_{t-1}} + i_t * \tilde{C}_t = f_t * f_{t-1} * C_{t-2} + others$$

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

LSTM



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t = f_t * f_{t-1} * C_{t-2} + others$$

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$ Not multiplication of weights $\prod W_t$

Deep RNN

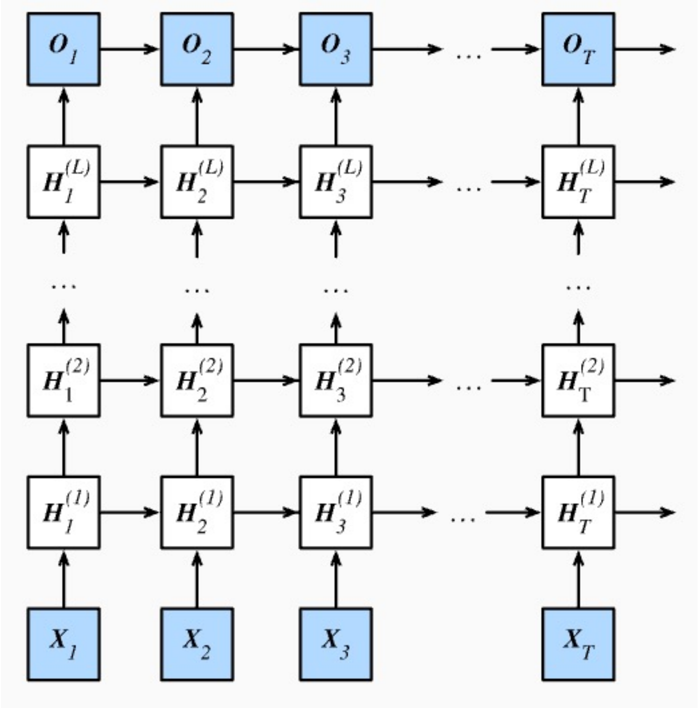


Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Deep RNN

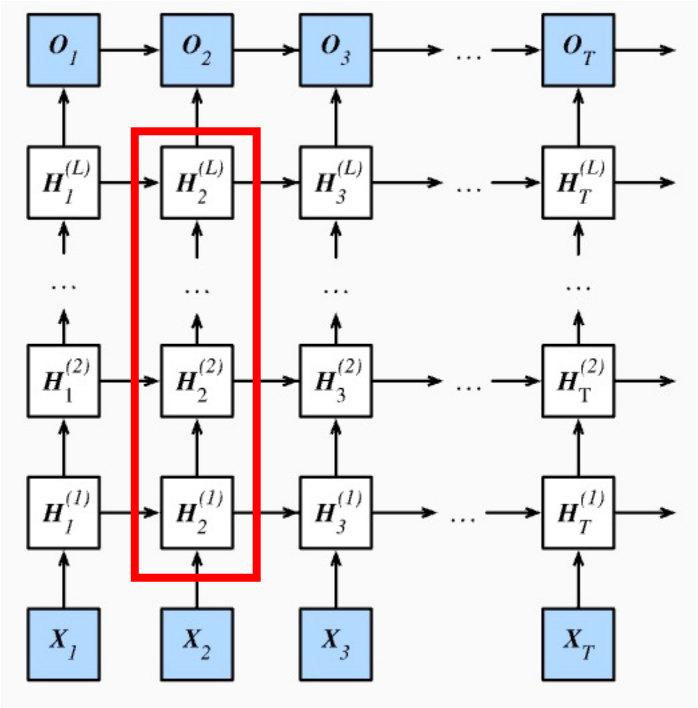
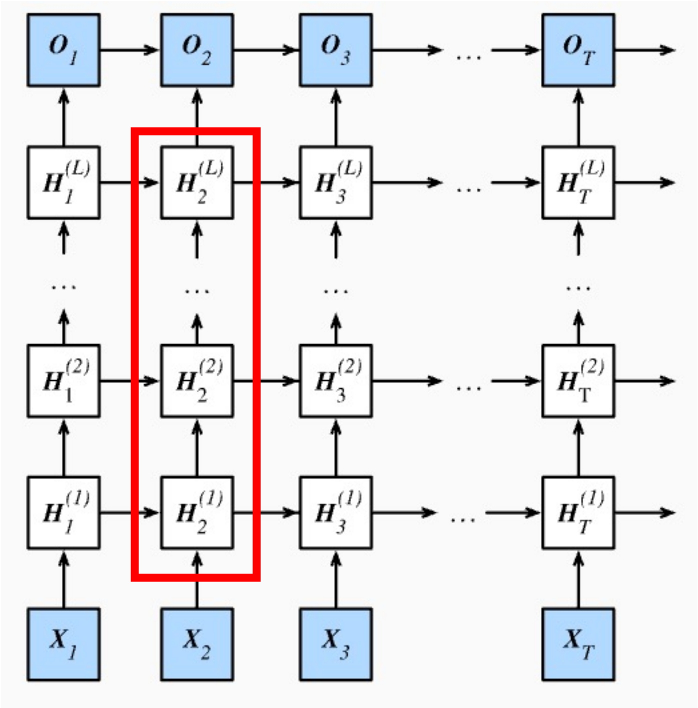


Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Deep RNN



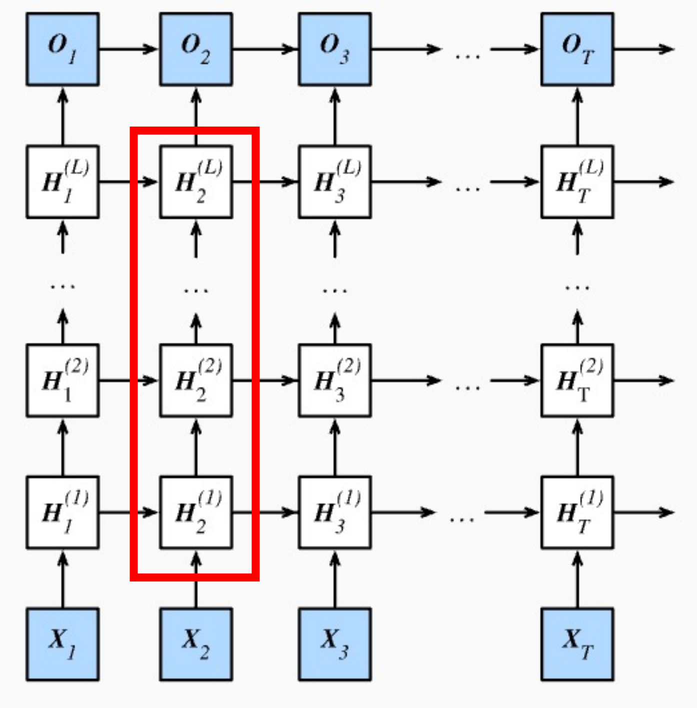
For each unit: a mapping

$$\text{RNN}: h_t^{l-1}, h_{t-1}^{l} \rightarrow h_t^{l} \qquad h_t^l \in \mathbb{R}^n$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

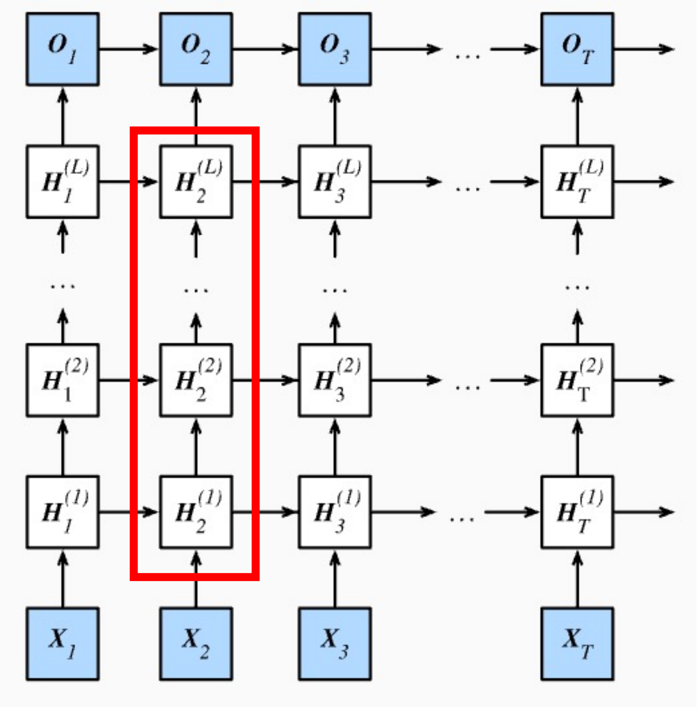
$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \rightarrow h_t^l \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

$$\mathrm{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^l \qquad h_t^l \in \mathbb{R}^n$$

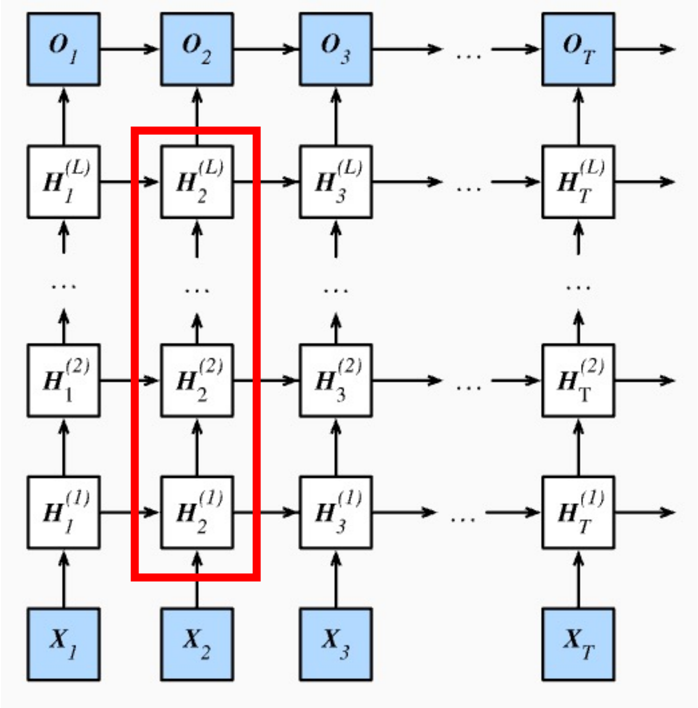
$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\mathrm{sigm}, \tanh\}$$

$$T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

$$\text{RNN}: h_t^{l-1}, h_{t-1}^{l} \to h_t^l \qquad h_t^l \in \mathbb{R}^n$$

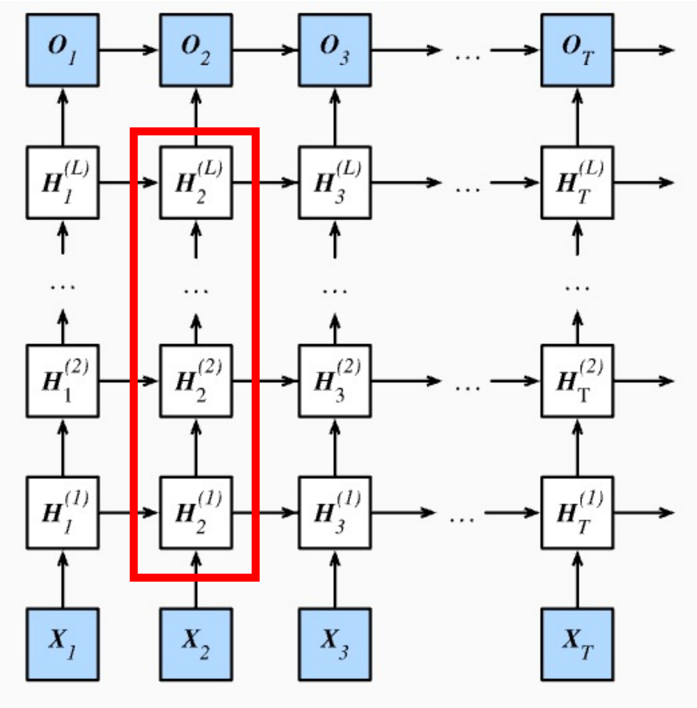
$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$ (a mapping function)

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^l \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

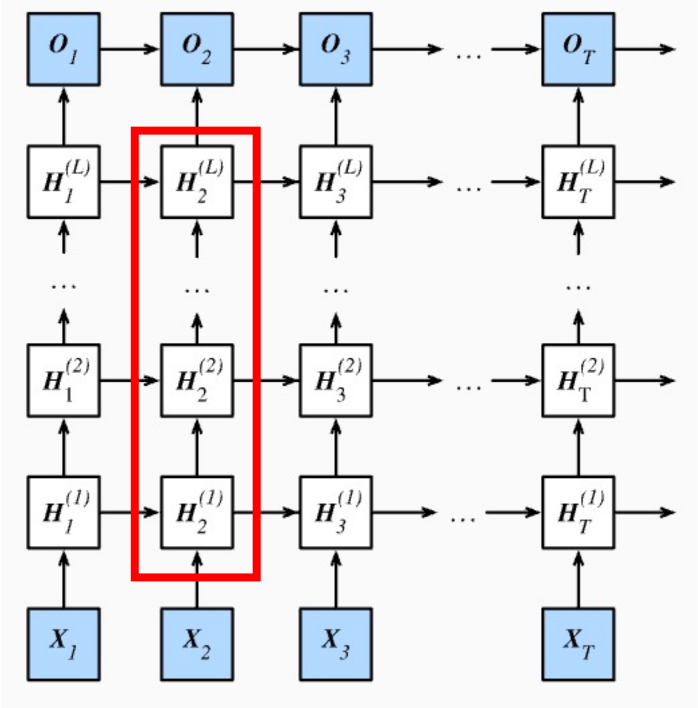
$T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$ (a mapping function)

$Wx + b$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^l \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$ (a mapping function)

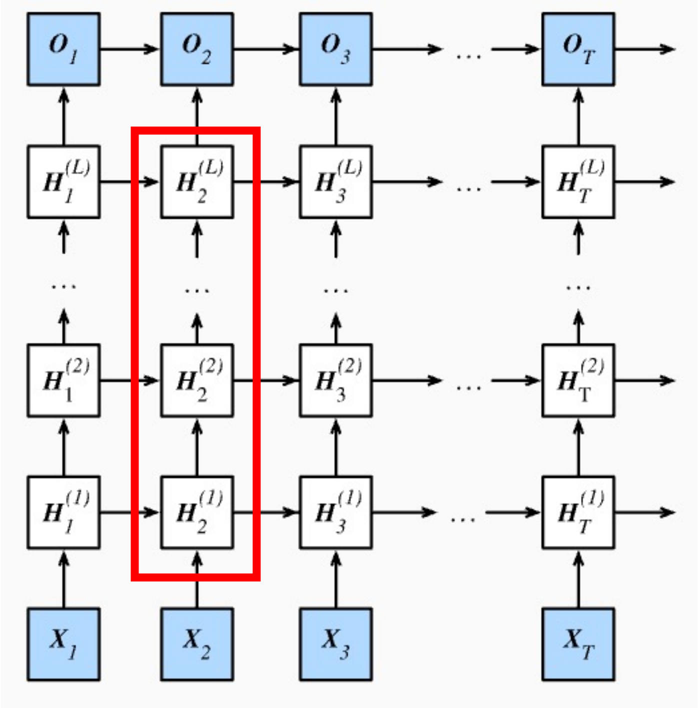
$$Wx + b$$

$x \in R^n, W \in R^{m \times n}, b \in R^m$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \rightarrow h_t^l \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$$T_{n,m} : \boxed{\mathbb{R}^n \rightarrow \mathbb{R}^m}$$ (a mapping function)

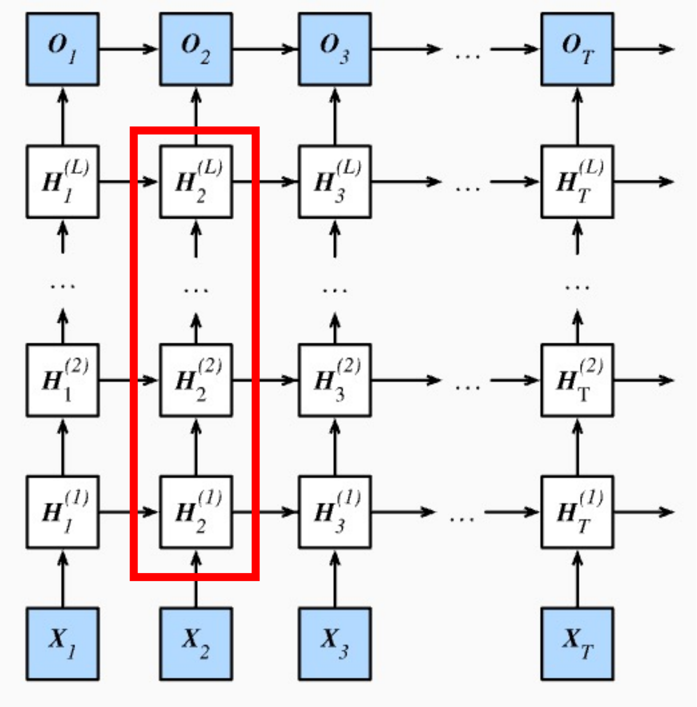
$$Wx + b$$

$$x \in R^n, W \in R^{m \times n}, b \in R^m$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^{l} \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(\boxed{T_{n,n}} h_t^{l-1} + \boxed{T_{n,n}} h_{t-1}^{l}), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$$T_{n,m} : \boxed{\mathbb{R}^n \to \mathbb{R}^m}$$ (a mapping function)

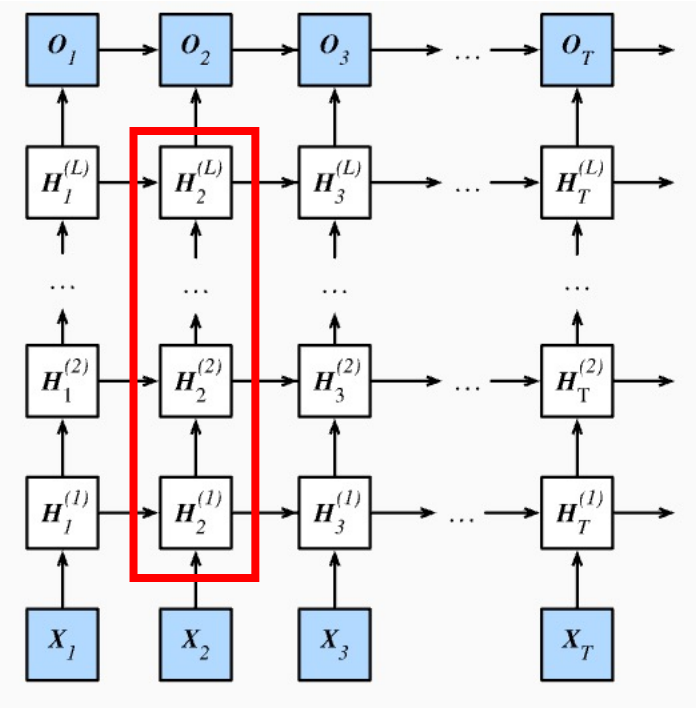
$$Wx + b$$

$x \in R^n, W \in R^{m \times n}, b \in R^m$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN



$$h_t = f\left[\text{W} * \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right]$$

For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^{l} \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(\boxed{T_{n,n}} h_t^{l-1} + \boxed{T_{n,n}} h_{t-1}^{l}), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$$T_{n,m} : \boxed{\mathbb{R}^n \to \mathbb{R}^m}$$ (a mapping function)

$$Wx + b$$

$x \in R^n, W \in R^{m\times n}, b \in R^m$

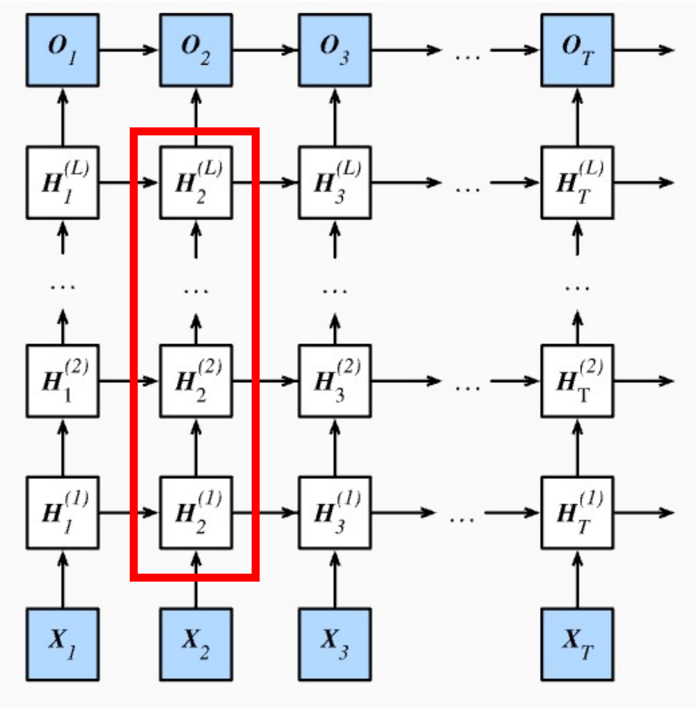
Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN

$$h_t = f\left[W * \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right]$$

$f = \tanh(\cdot)$



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \rightarrow h_t^{l} \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(\boxed{T_{n,n}} h_t^{l-1} + \boxed{T_{n,n}} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$$T_{n,m} : \boxed{\mathbb{R}^n \rightarrow \mathbb{R}^m}$$ (a mapping function)

$$Wx + b$$

$$x \in R^n, W \in R^{m \times n}, b \in R^m$$

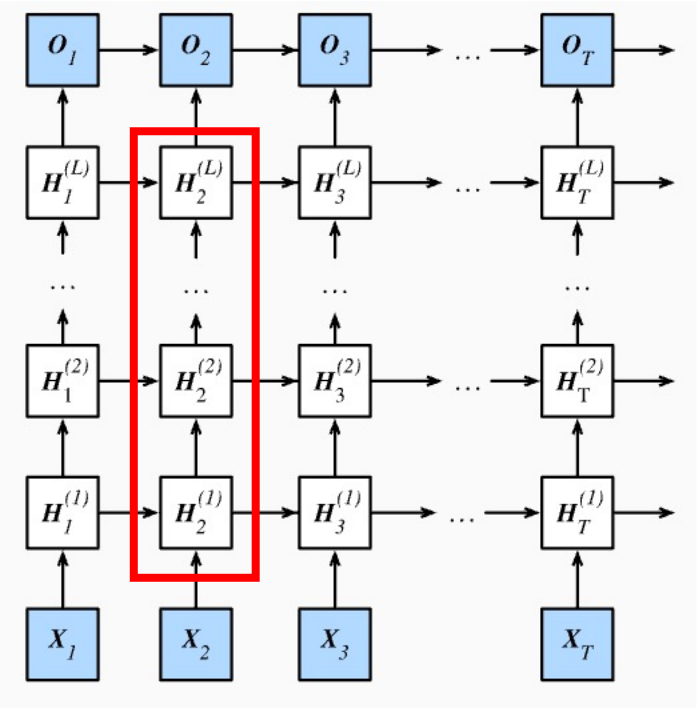
Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN

$$h_t = f\left[\overbrace{W * \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}}^{\text{Vector feature}}\right]$$

$f = \tanh(\cdot)$



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^l \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(\boxed{T_{n,n}} h_t^{l-1} + \boxed{T_{n,n}} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$$T_{n,m} : \boxed{\mathbb{R}^n \to \mathbb{R}^m}$$ (a mapping function)

$$Wx + b$$

$$x \in R^n, W \in R^{m \times n}, b \in R^m$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

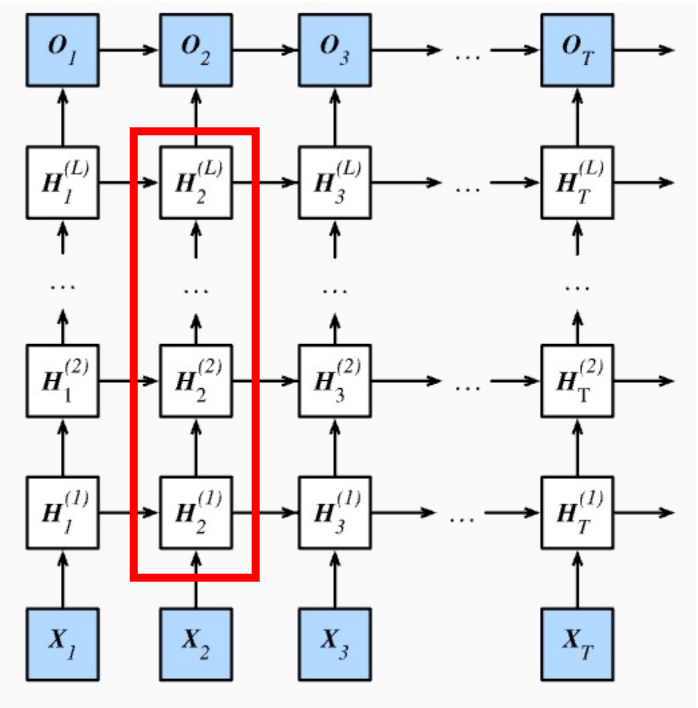
Deep RNN

elementwise operation

Vector feature

$$h_t = f\left[W * \overbrace{\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}}^{\text{Vector feature}} \right]$$

$$f = \tanh(\cdot)$$



For each unit: a mapping

$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \to h_t^l \qquad h_t^l \in \mathbb{R}^n$$

$$h_t^l = f(\boxed{T_{n,n}} h_t^{l-1} + \boxed{T_{n,n}} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

$$T_{n,m} : \boxed{\mathbb{R}^n \to \mathbb{R}^m}$$ (a mapping function)

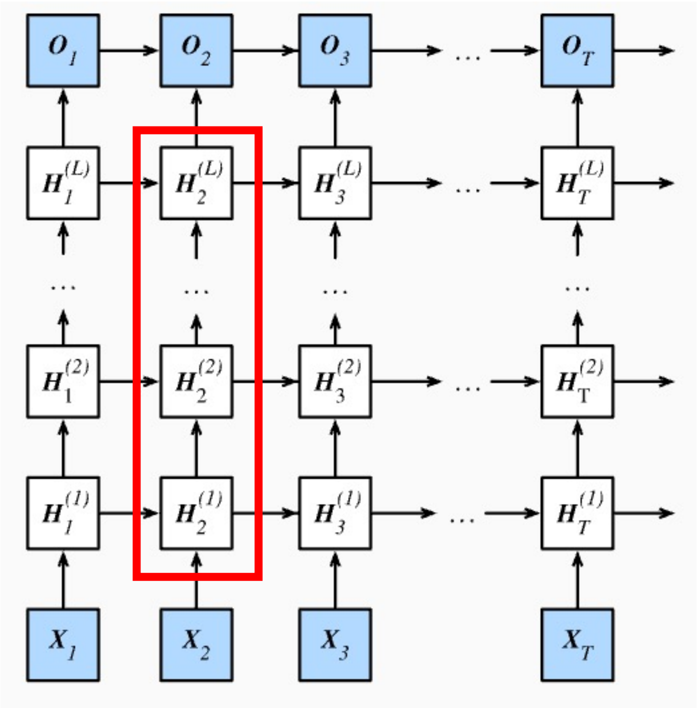
$$Wx + b$$

$$x \in R^n, W \in R^{m \times n}, b \in R^m$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep LTSM

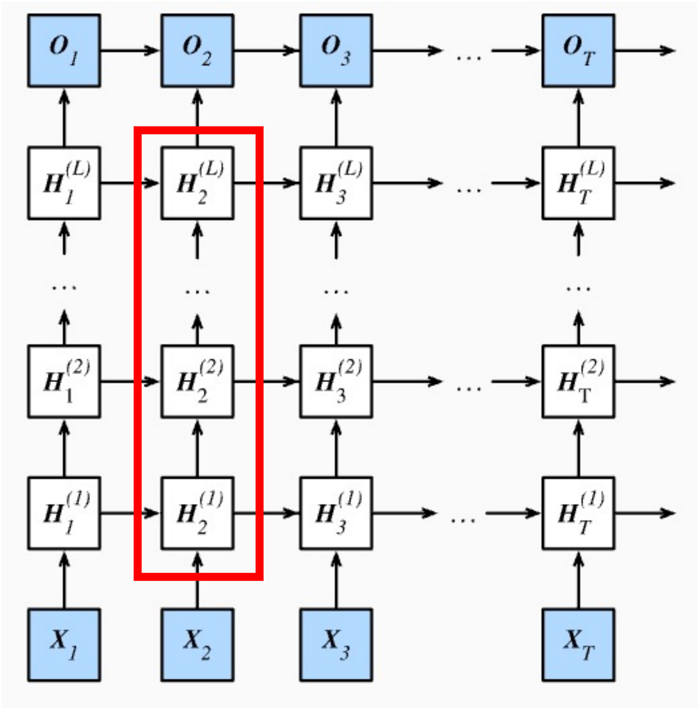


$$\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep LTSM



$$\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l$$

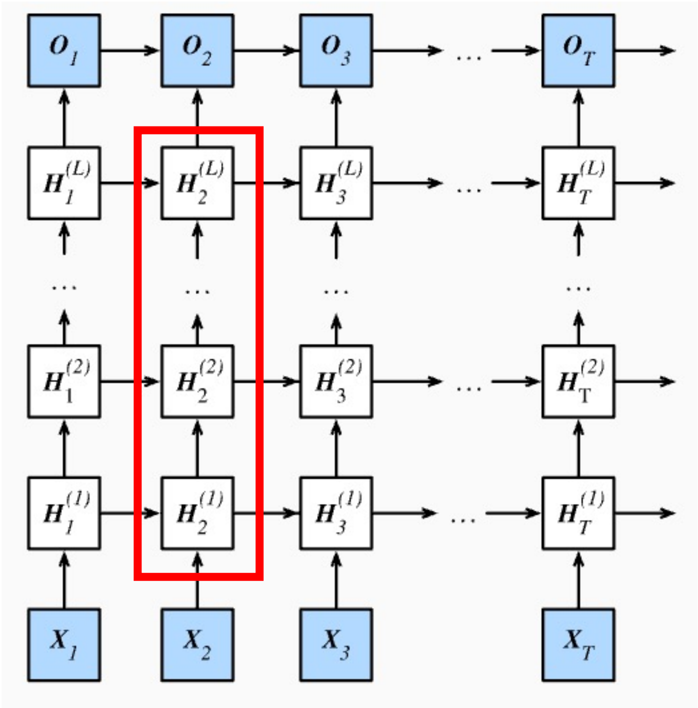
$$c_t^l \in \mathbb{R}^n$$

$$h_t^l \in \mathbb{R}^n$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep LTSM



$$c_t^l \in \mathbb{R}^n$$

$$h_t^l \in \mathbb{R}^n$$

$$\mathrm{LSTM}: h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow h_t^l, c_t^l$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

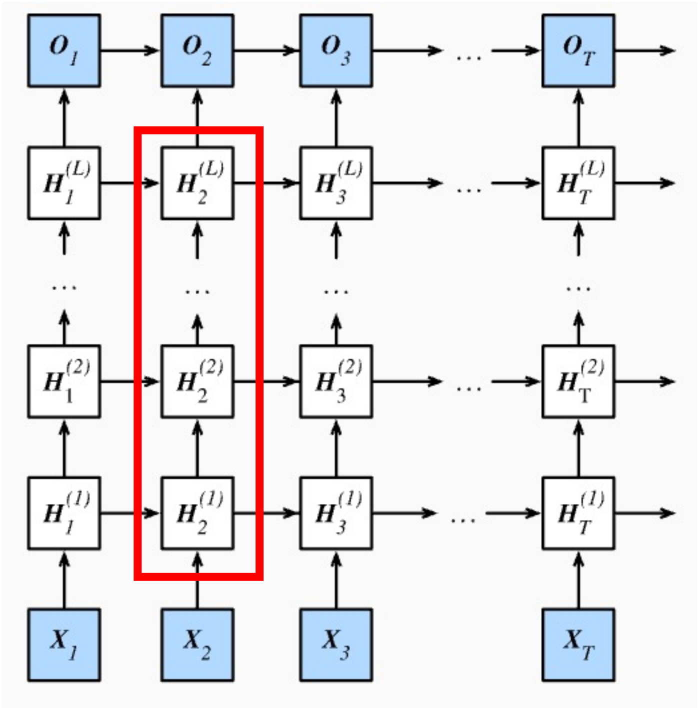
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep LTSM



$$c_t^l \in \mathbb{R}^n$$

$$h_t^l \in \mathbb{R}^n$$

$$\mathrm{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \rightarrow \boxed{h_t^l, c_t^l}$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \mathrm{sigm} \\ \mathrm{sigm} \\ \mathrm{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

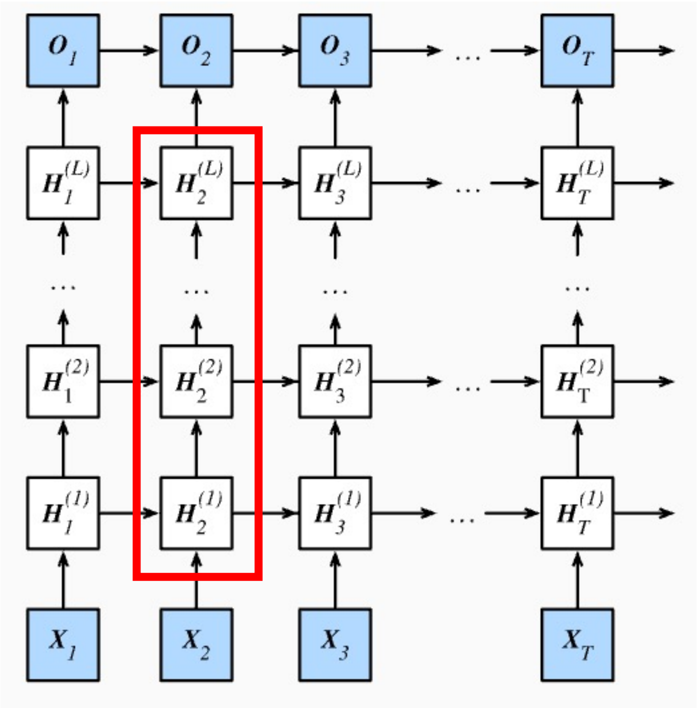
$$\boxed{c_t^l} = f \odot c_{t-1}^l + i \odot g$$

$$\boxed{h_t^l} = o \odot \tanh(c_t^l)$$

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep LTSM



$$\text{LSTM} : h_t^{l-1}, h_{t-1}^{l}, c_{t-1}^{l} \rightarrow \boxed{h_t^l, c_t^l}$$

$$c_t^l \in \mathbb{R}^n$$

$$h_t^l \in \mathbb{R}^n$$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} T_{2n,4n} \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^{l} \end{pmatrix}$$

$$\boxed{c_t^l} = \boxed{f} \odot c_{t-1}^l + \boxed{i} \odot g$$

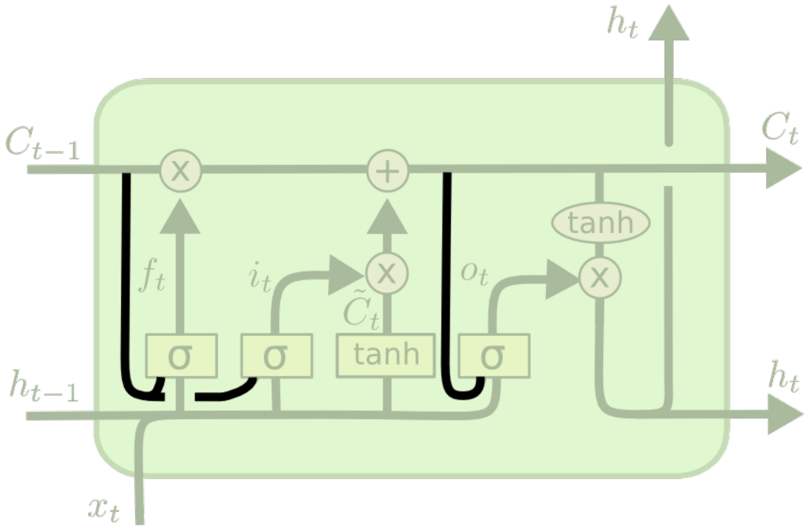
$$\boxed{h_t^l} = \boxed{o} \odot \tanh(c_t^l)$$

weights

Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Variants: peephole connections

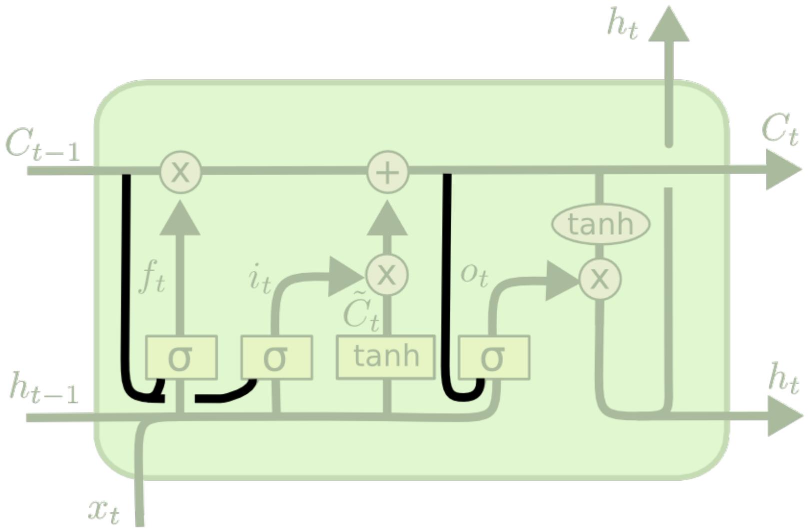


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$

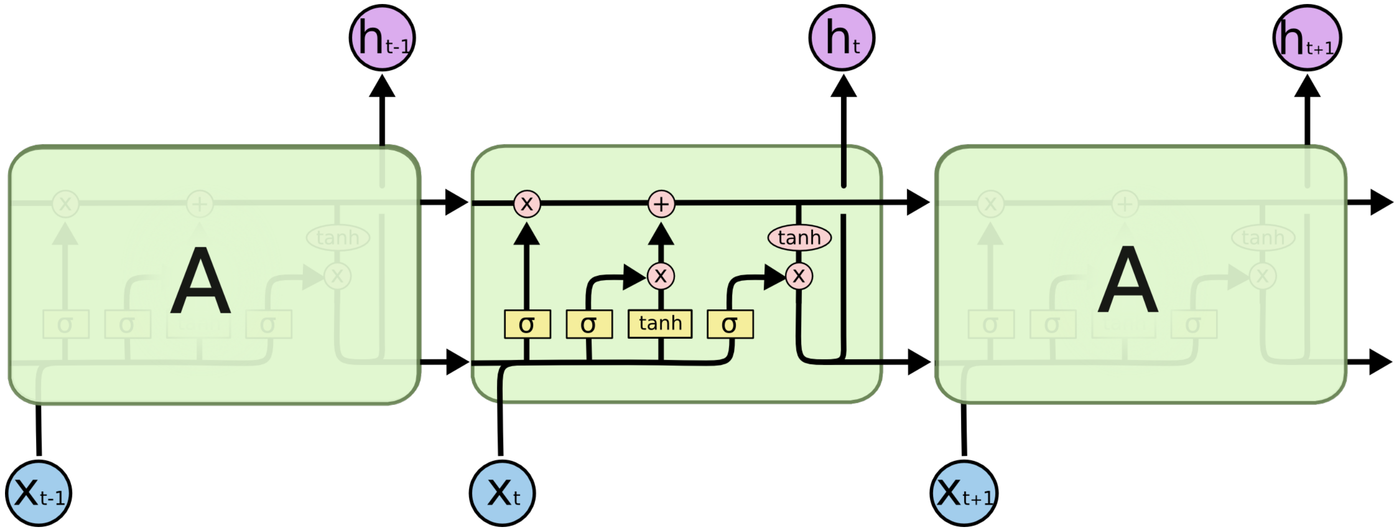
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$

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

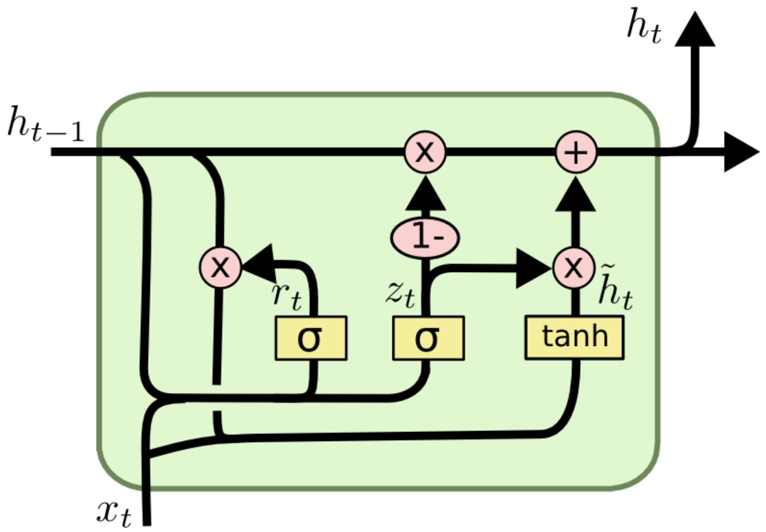
Variants: peephole connections



$$f_t = \sigma\left(W_f \cdot [\mathbf{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\mathbf{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\mathbf{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$



Variant: gated recurrent unit (GRU)



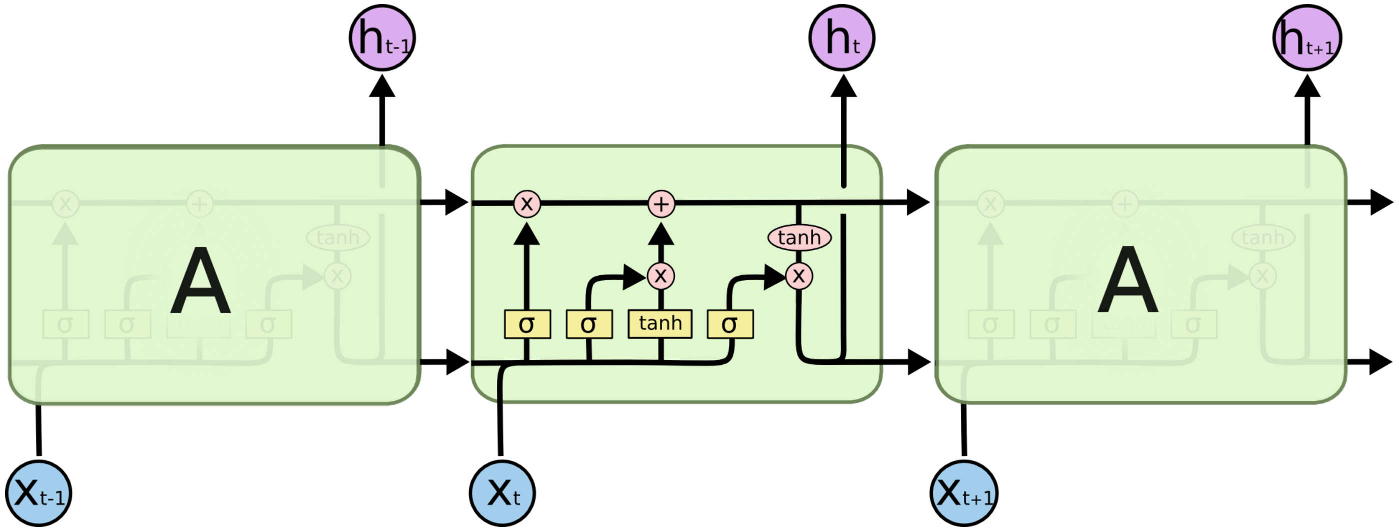
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

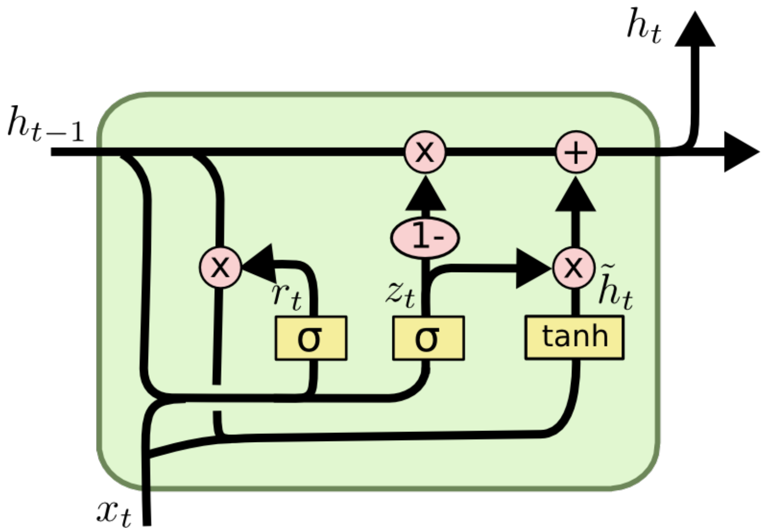
$$\boxed{\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)}$$

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

Weighted average



Variant: gated recurrent unit (GRU)



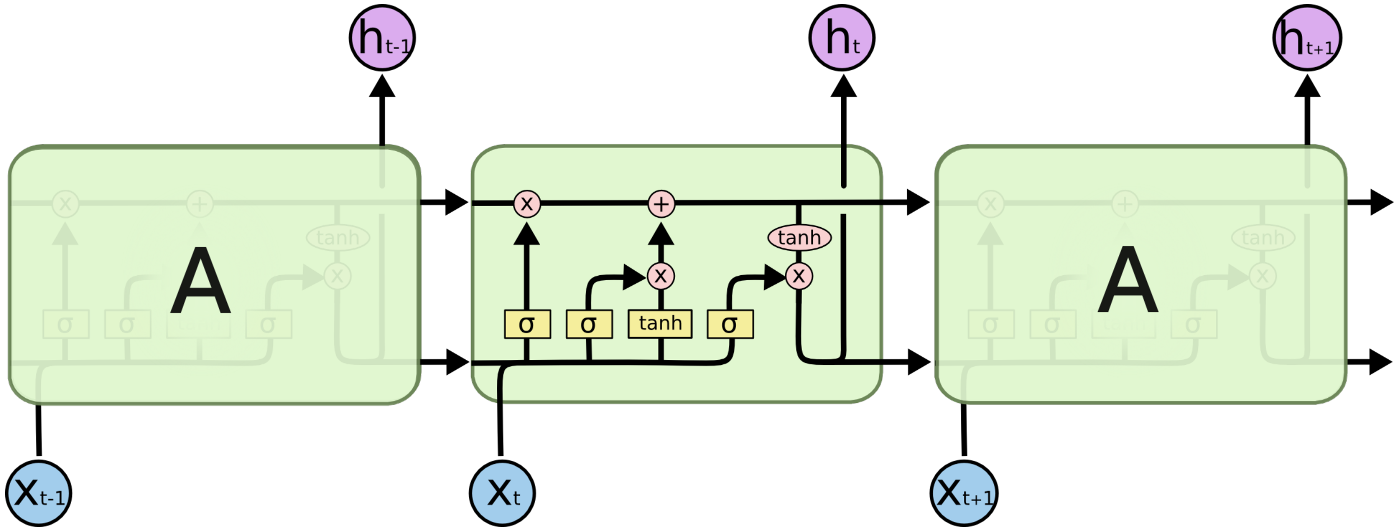
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

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

Weighted average



Reading

- Deep learning book
  - Section 10.2.2, 10.7, 10.10
- Blog: understanding LSTM
  - https://colah.github.io/posts/2015-08-Understanding-LSTMs/