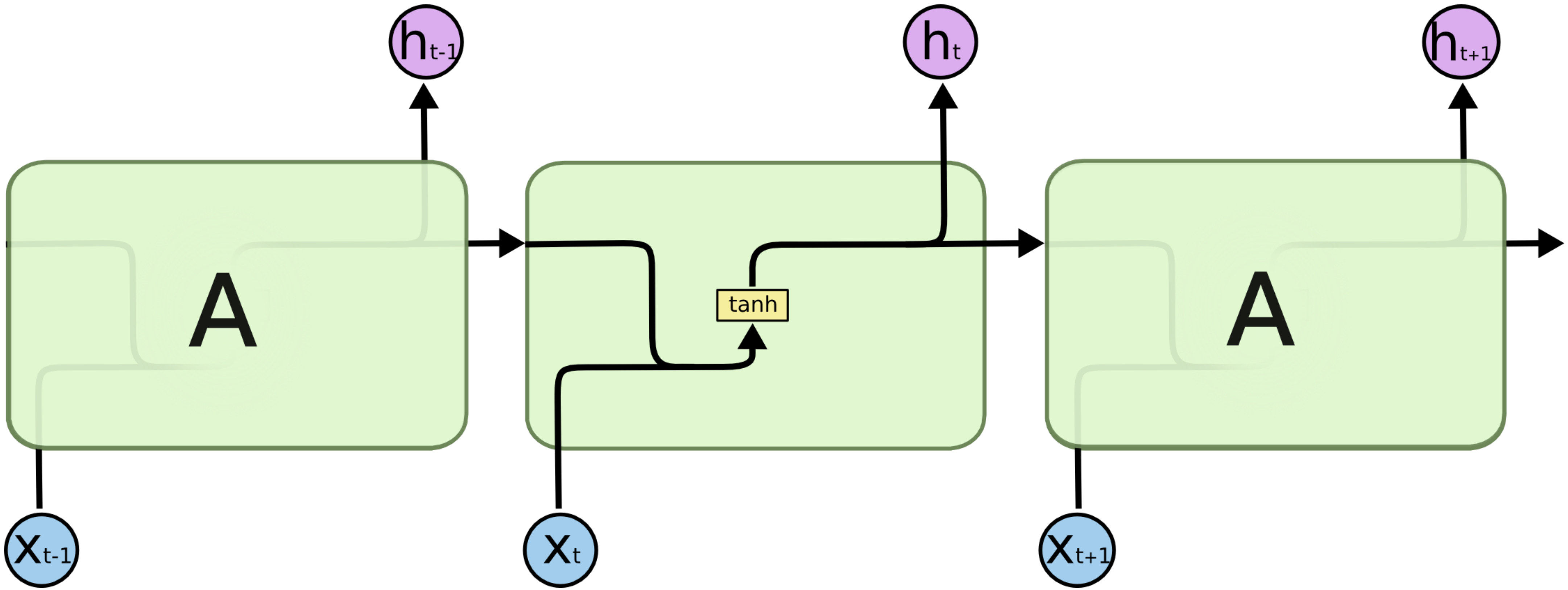
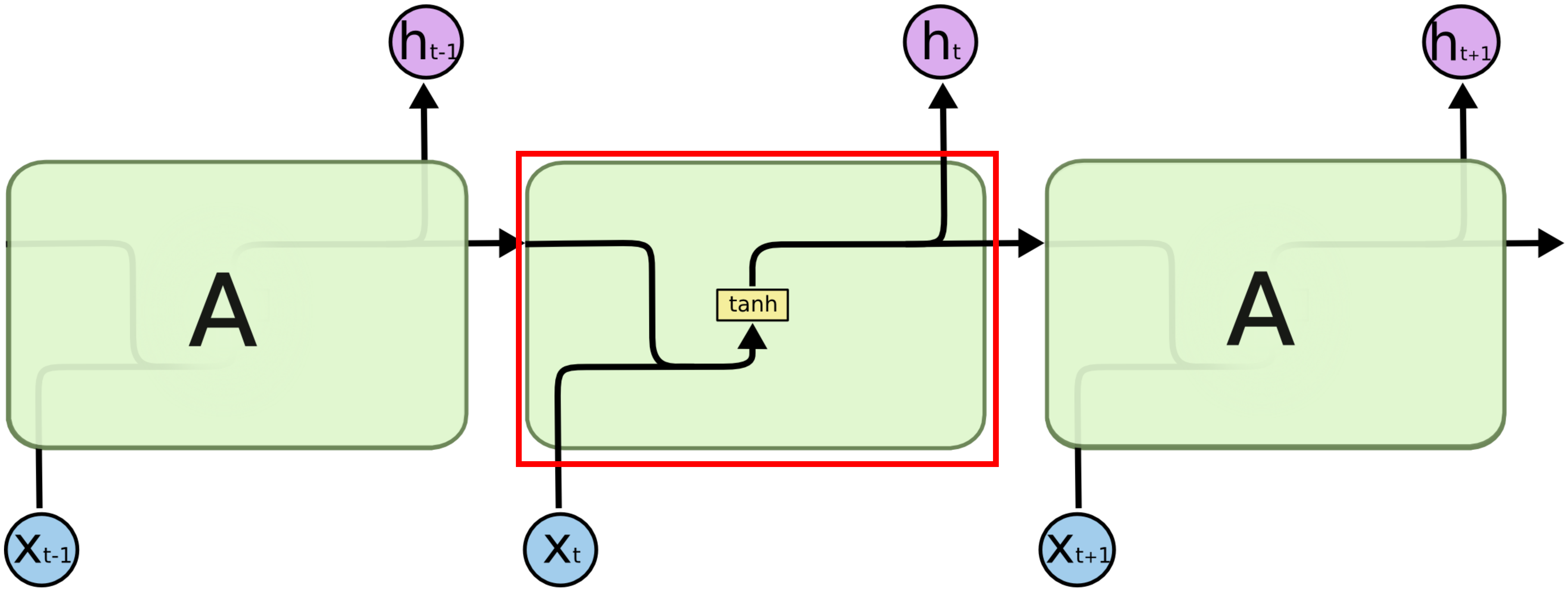
Long-Short Term Memory and RNN Architectures

Neural Networks Design And Application

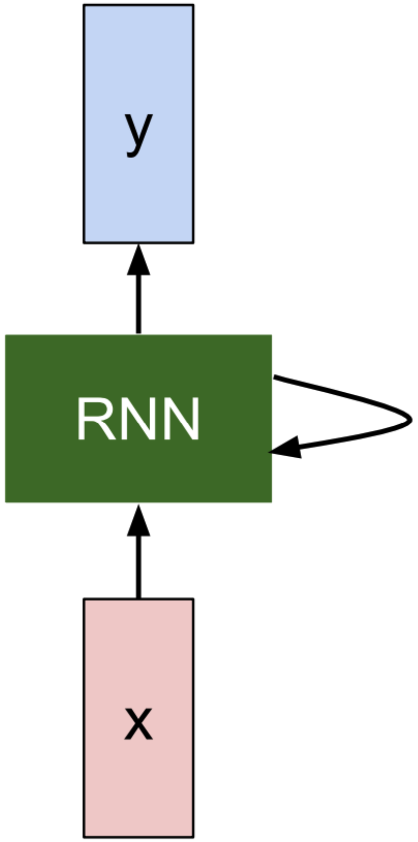
A conventional RNN



A conventional RNN



Recurrent neural networks



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

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

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

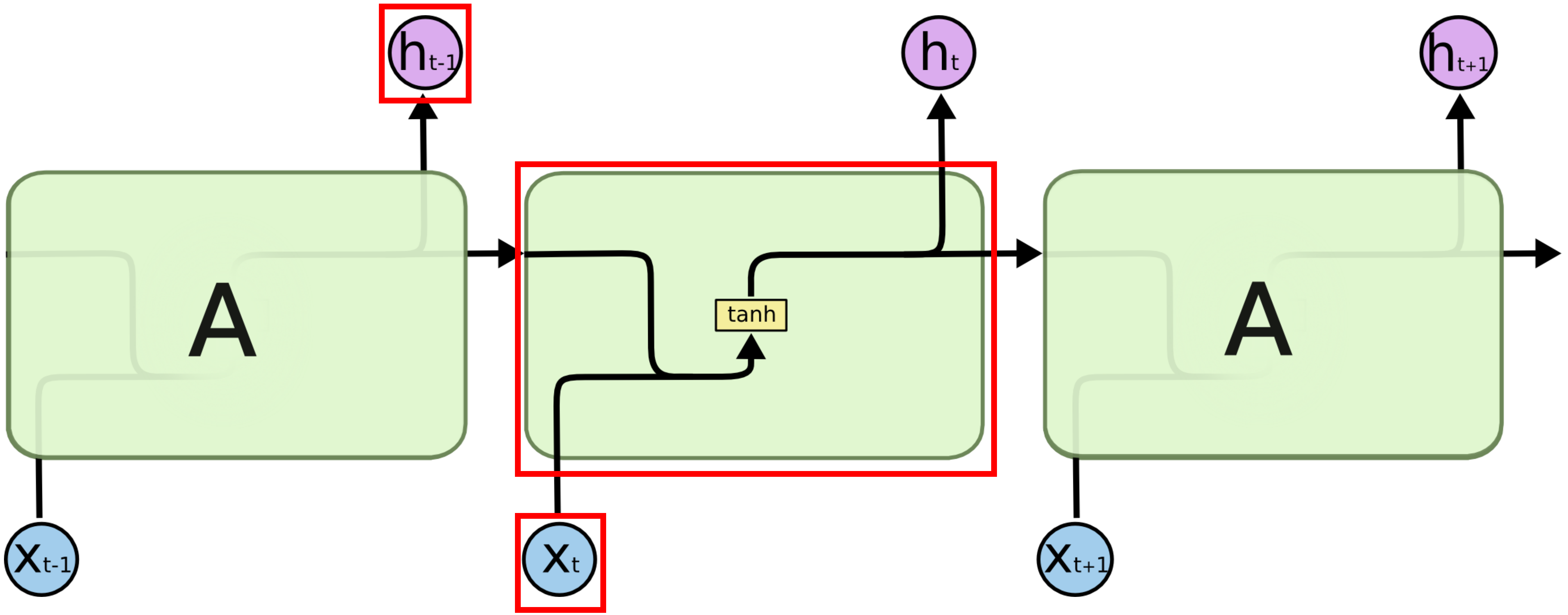
new state

old state

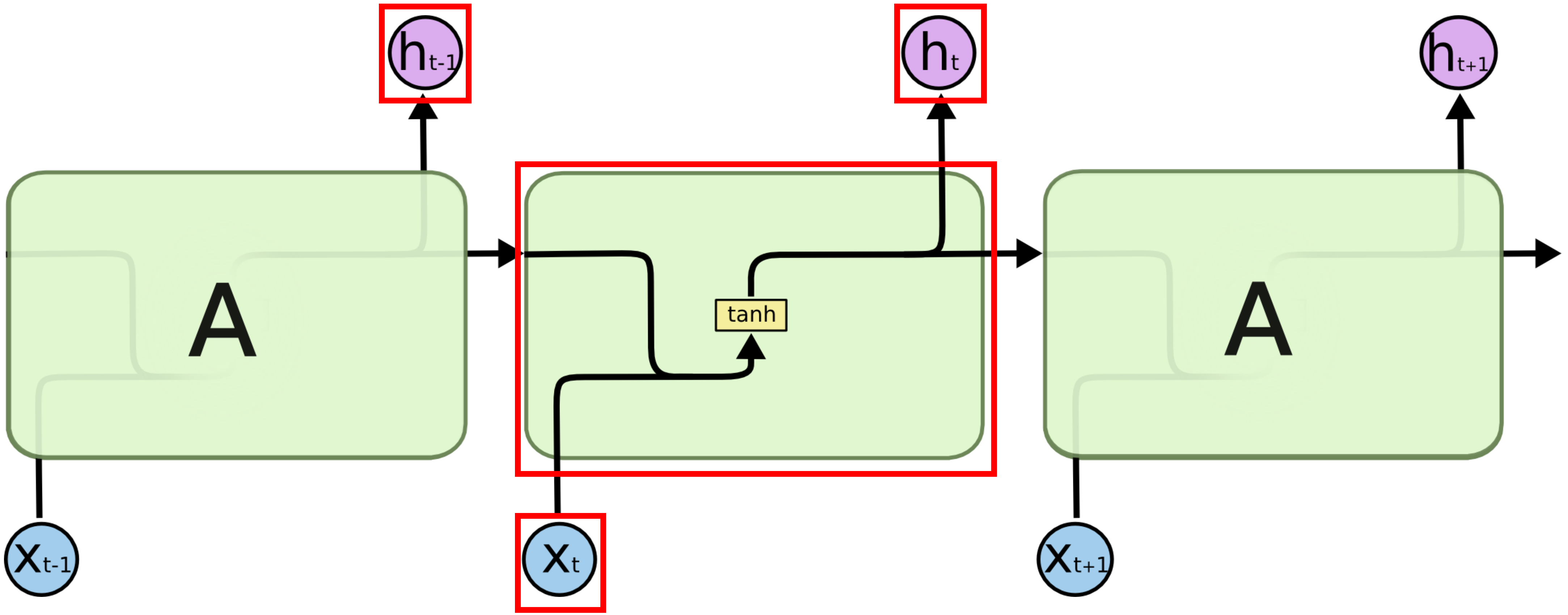
input vector at some time step

some function with parameters W

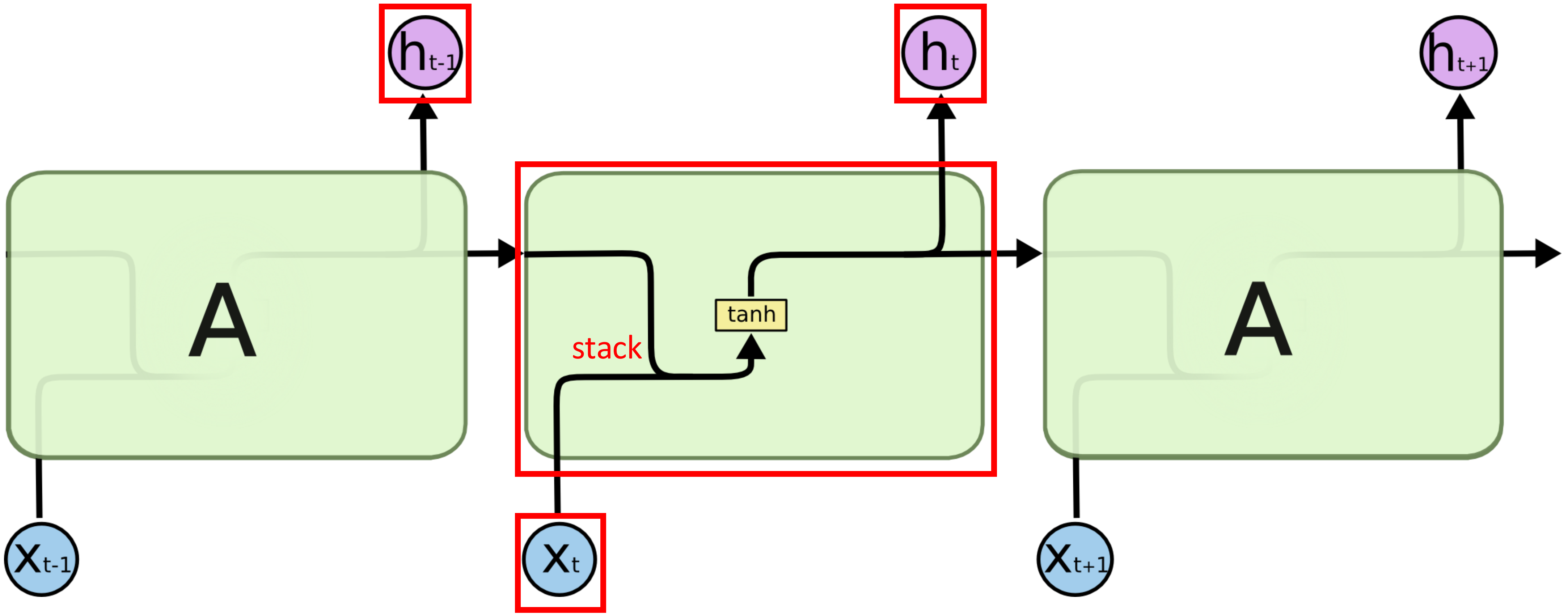
A conventional RNN



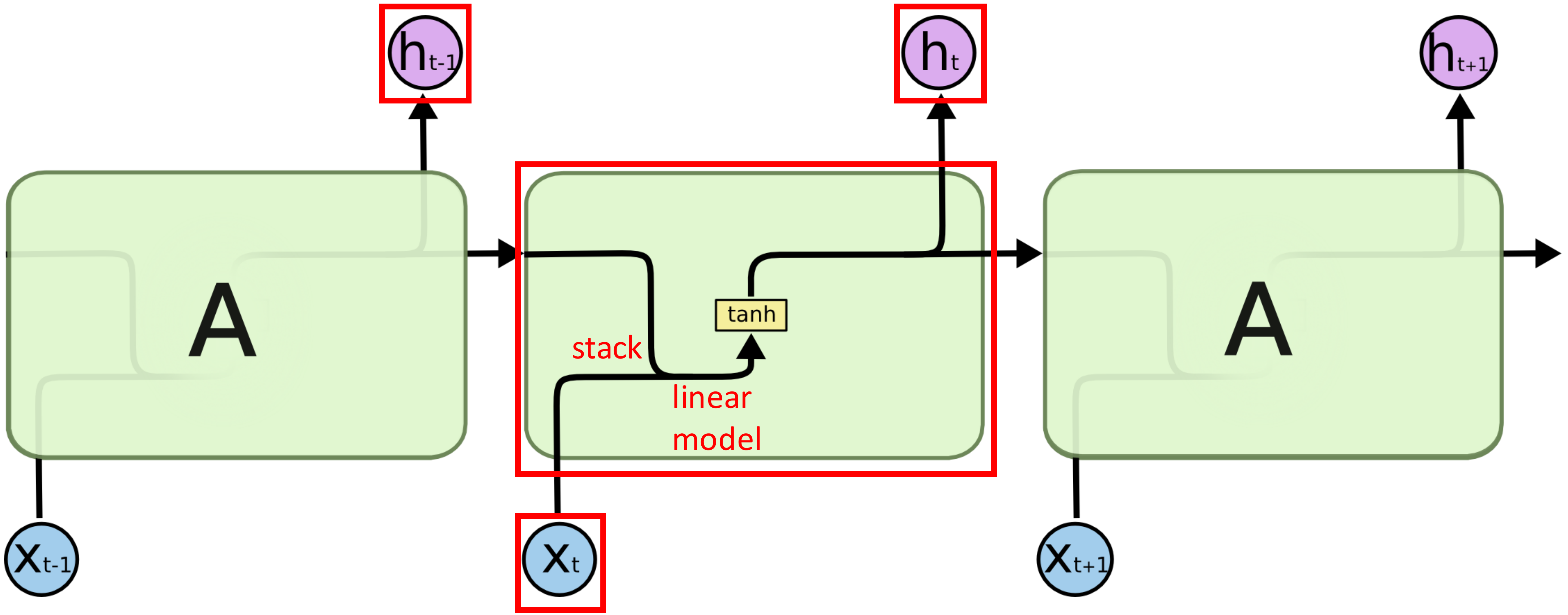
A conventional RNN



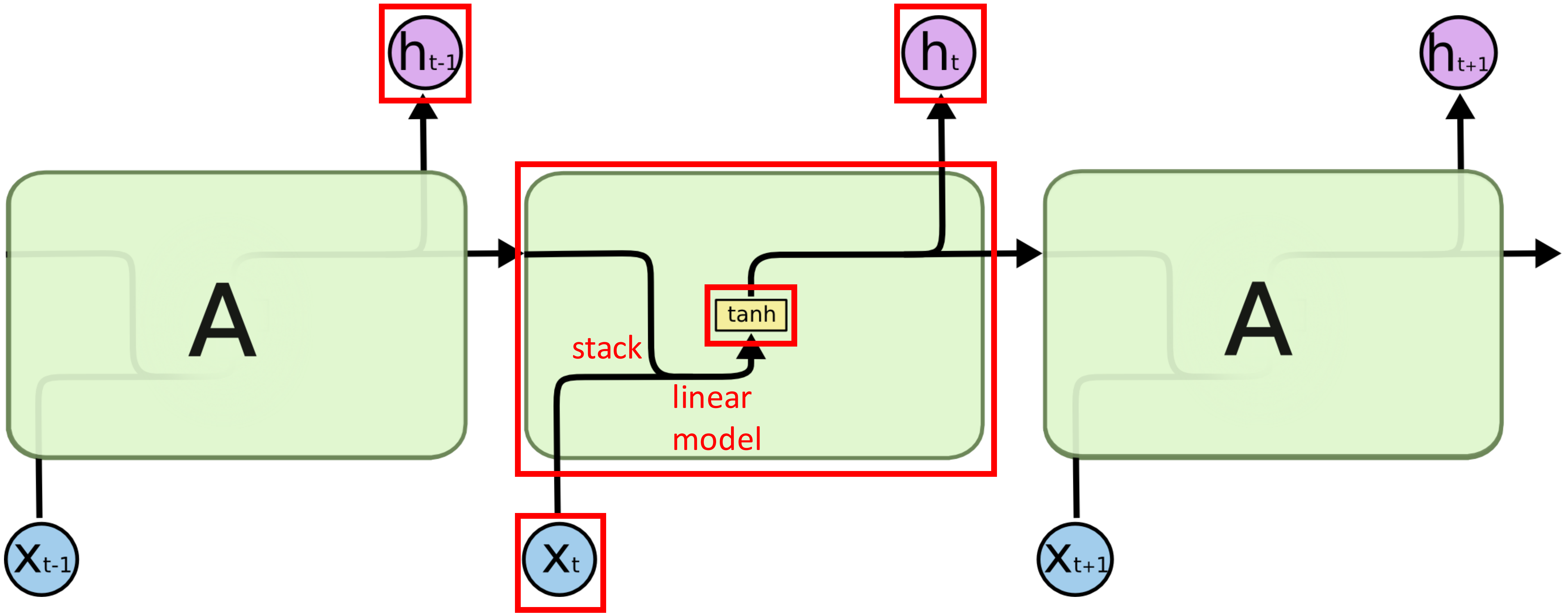
A conventional RNN



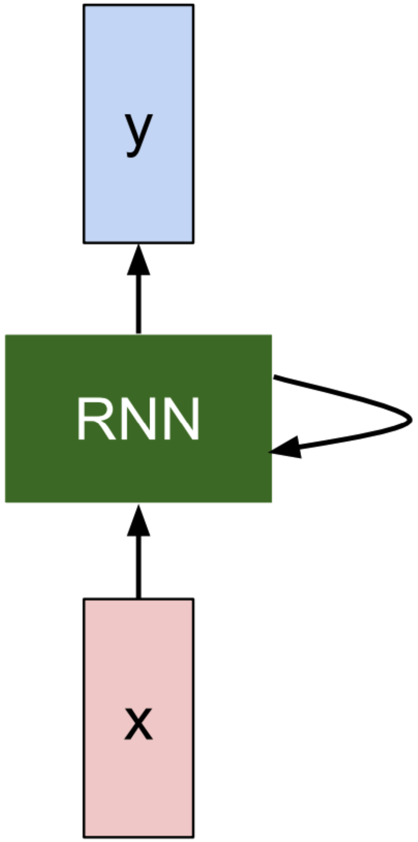
A conventional RNN



A conventional RNN



Recurrent neural networks



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

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

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

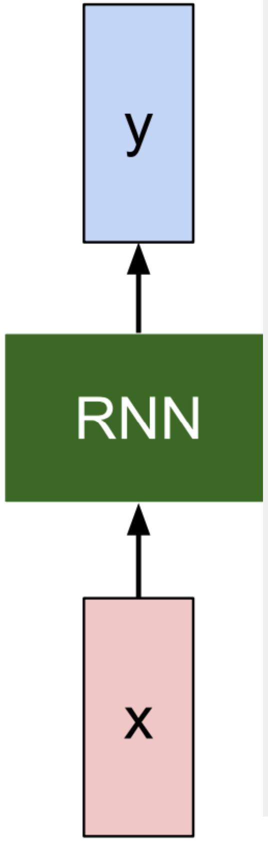
new state

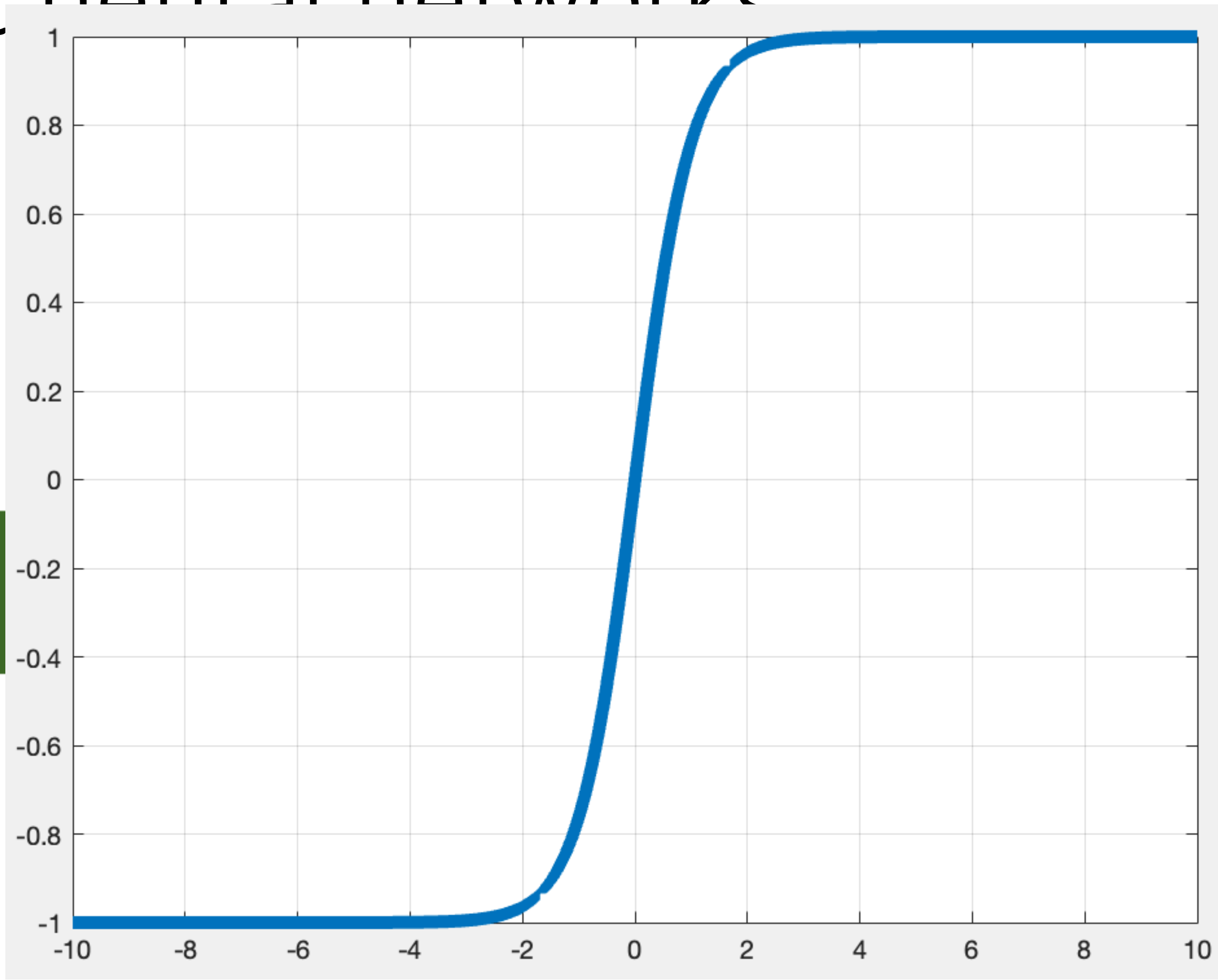
old state

input vector at
some time step

some function
with parameters W

Recurrent neural networks





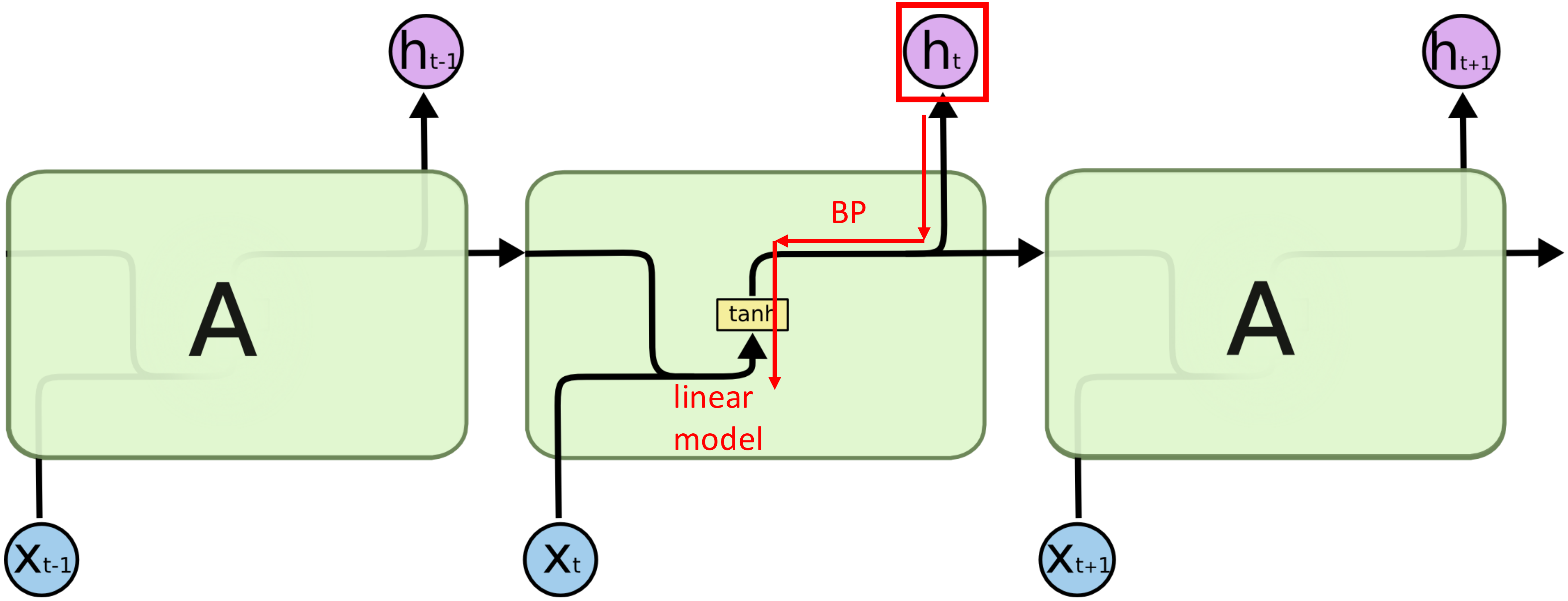
tanh function

$f = \tanh(\cdot)$

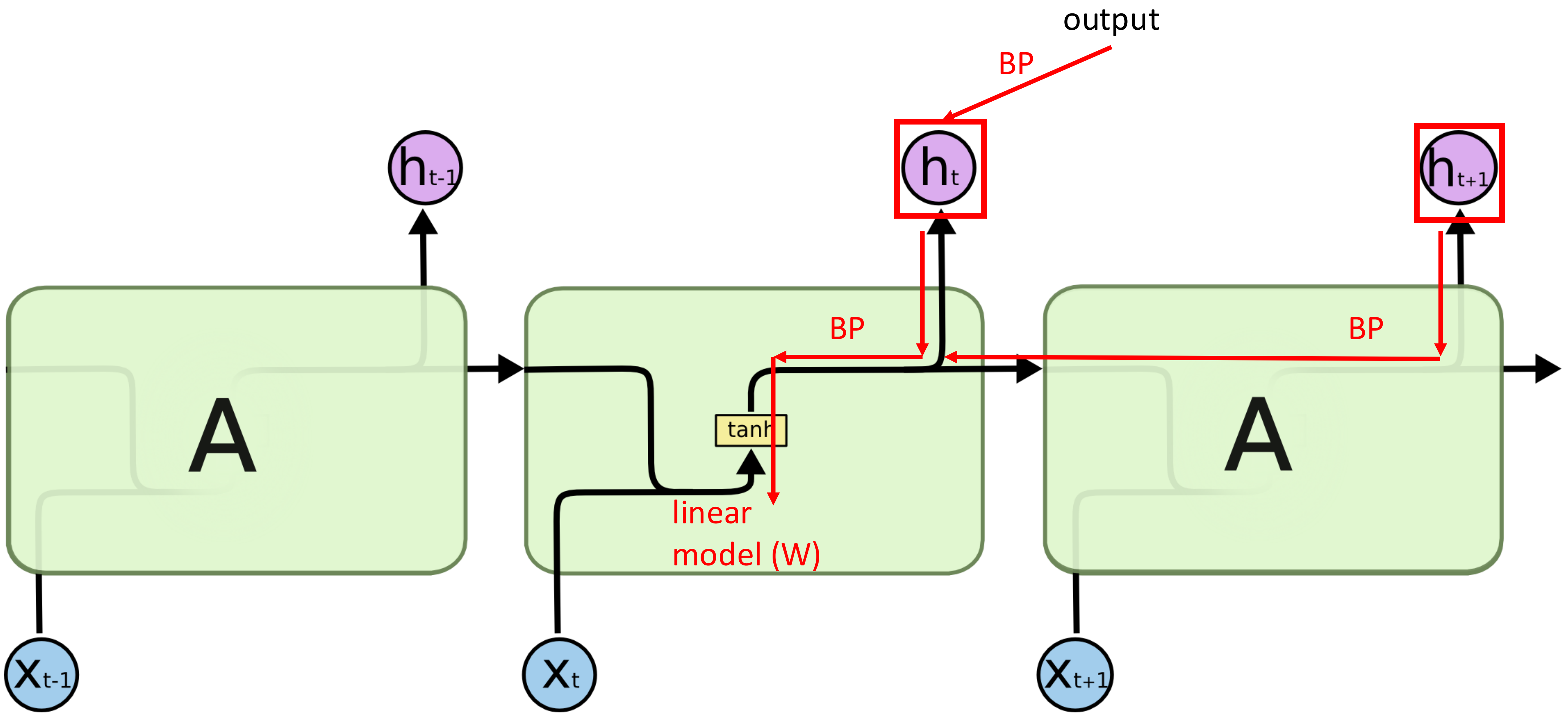
$\boxed{x_t})$

put vector at
ome time step

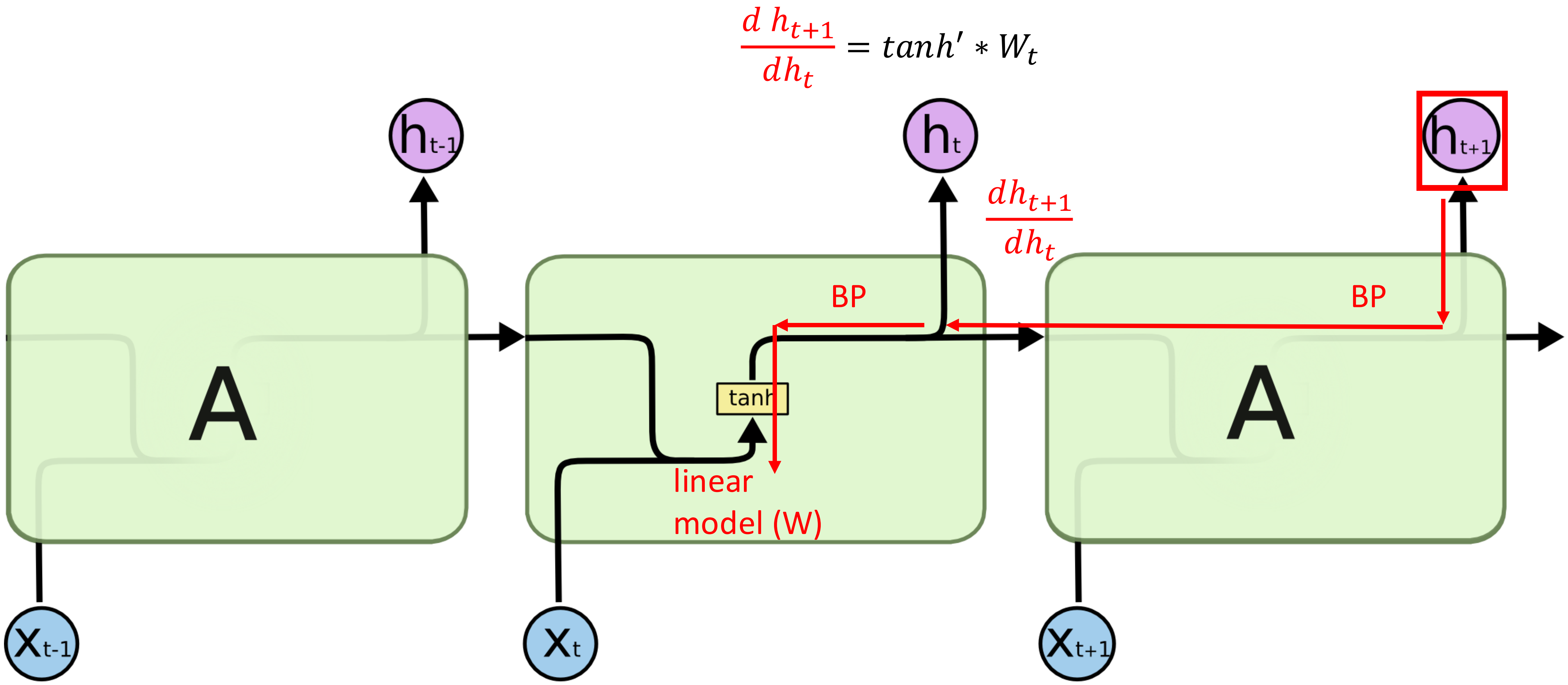
A conventional RNN



A conventional RNN



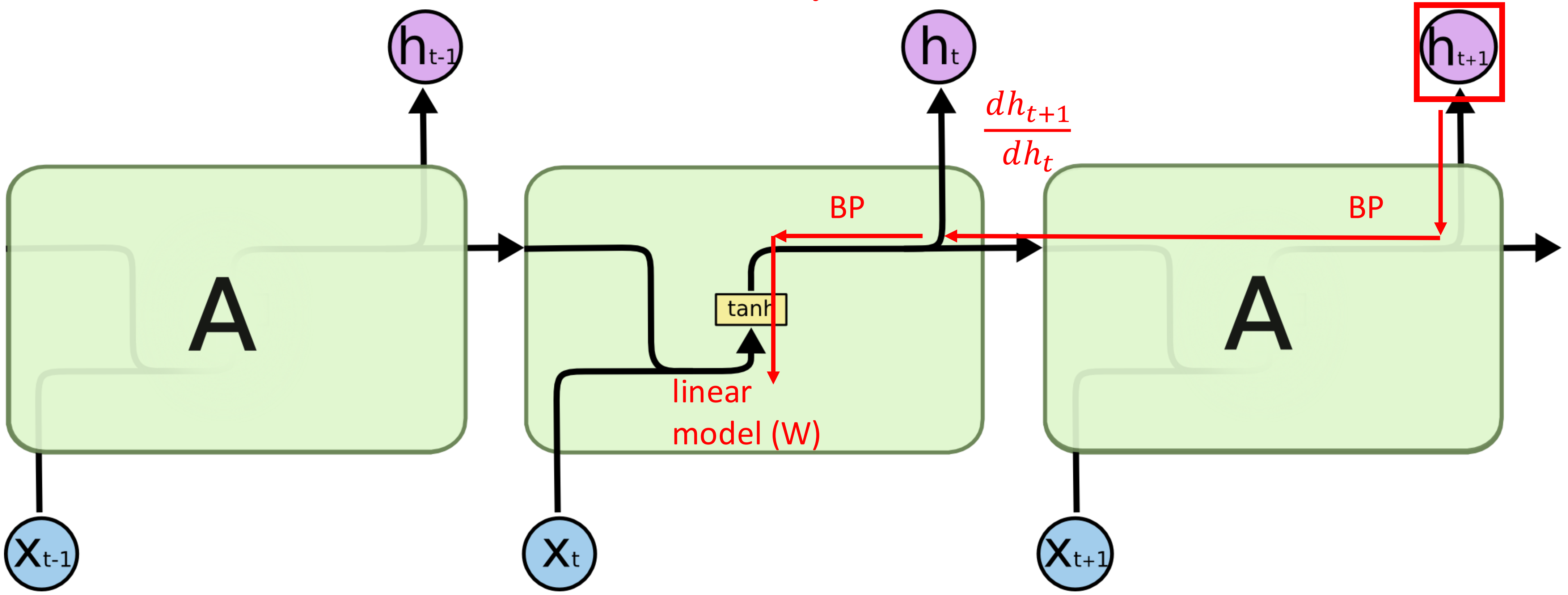
A conventional RNN



A conventional RNN

Q: what issue will we have?

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

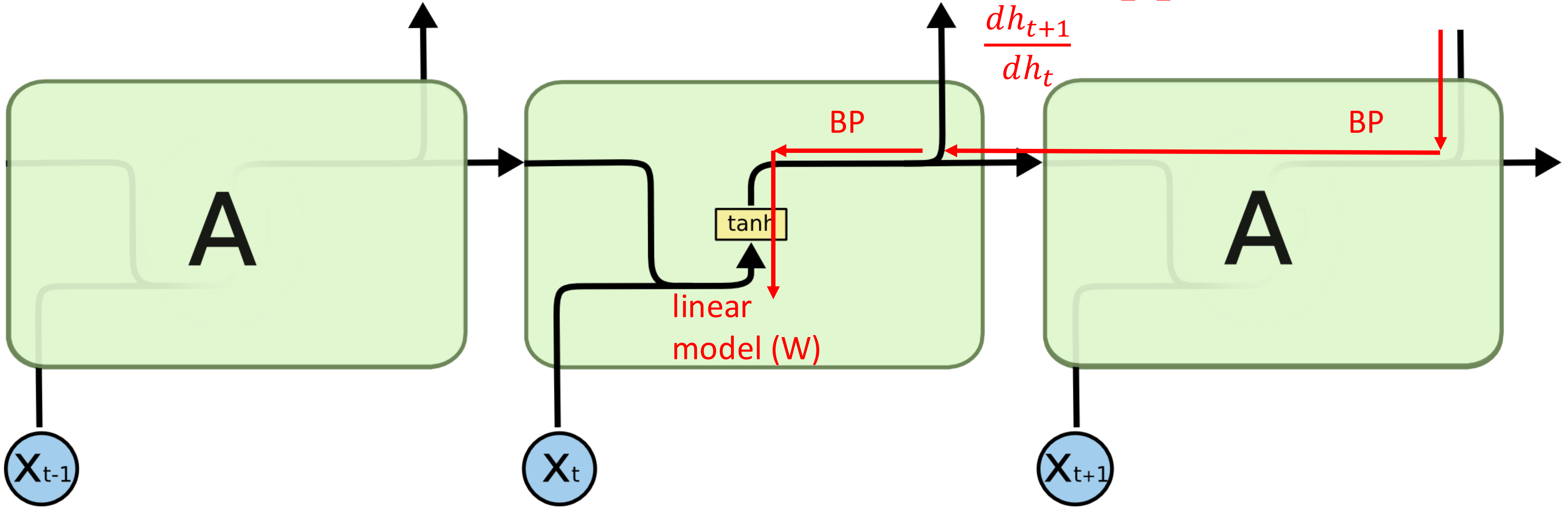


A conventional RNN

Q: what issue will we have?

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

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



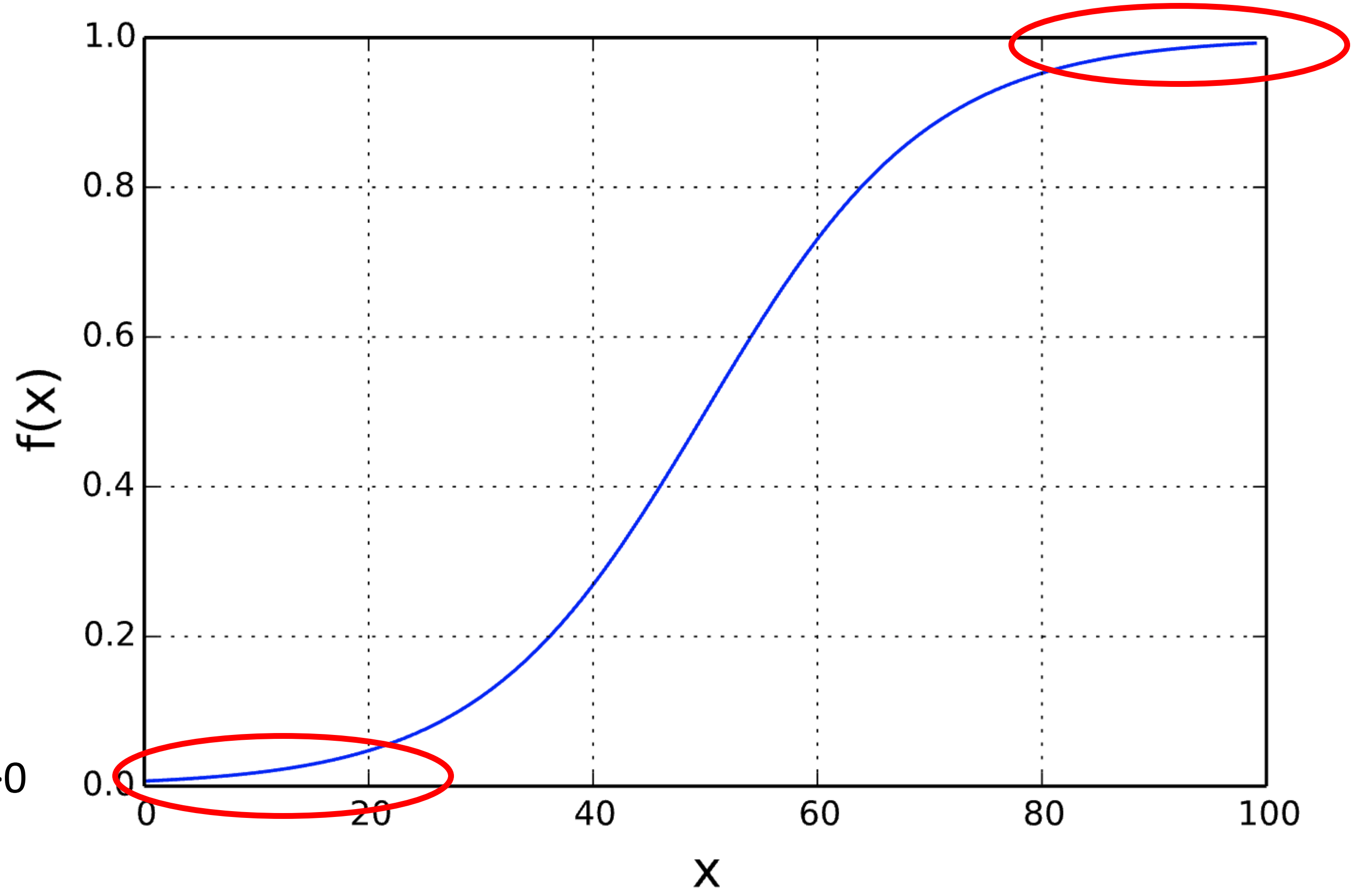
Gradient vanish

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

$$f_i \rightarrow x_i$$

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

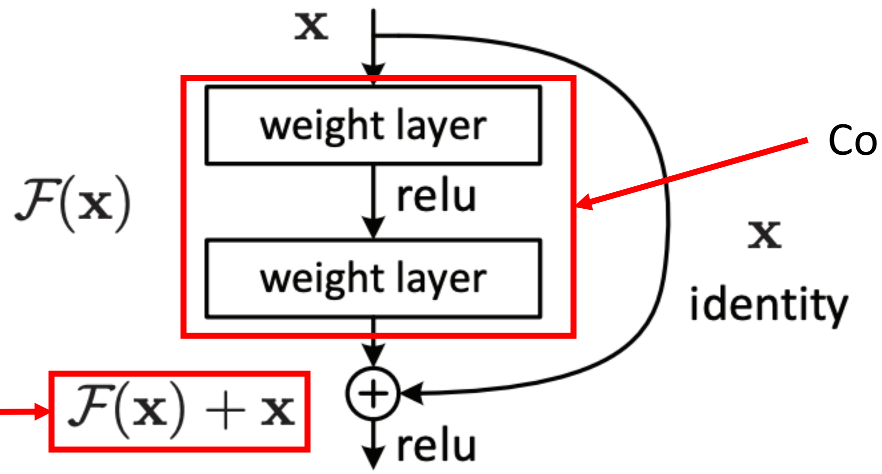
gradients->0



gradients->0

Sigmoid function

ResNet: shortcut connection



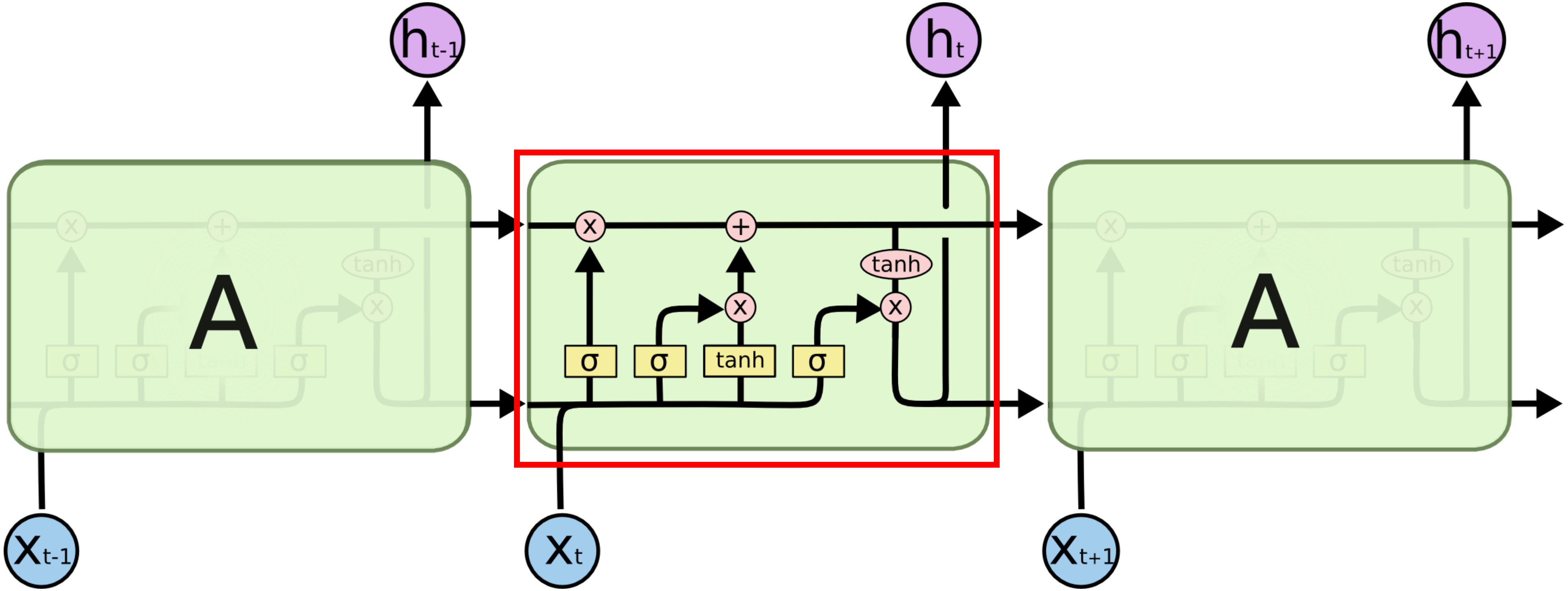
Conv layers

Addition operation

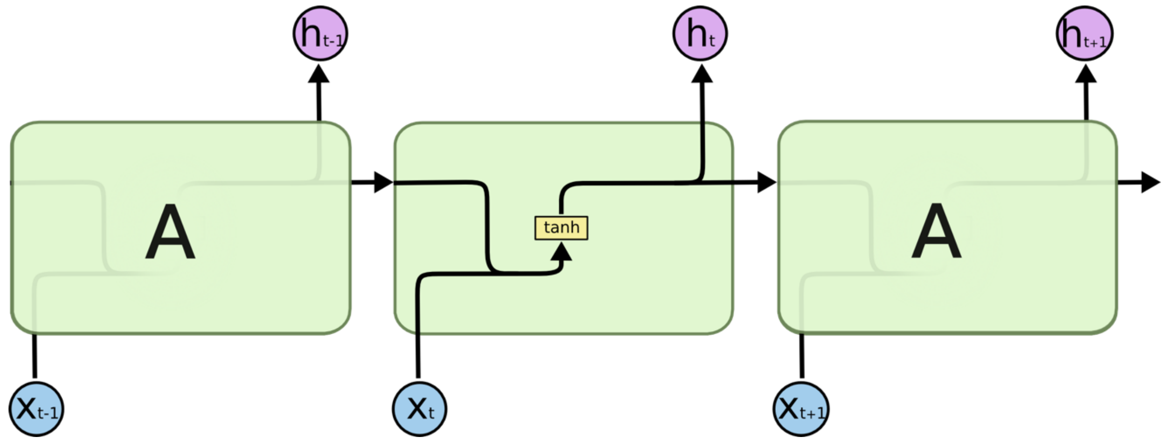
implication: same dimension

Figure 2. Residual learning: a building block.

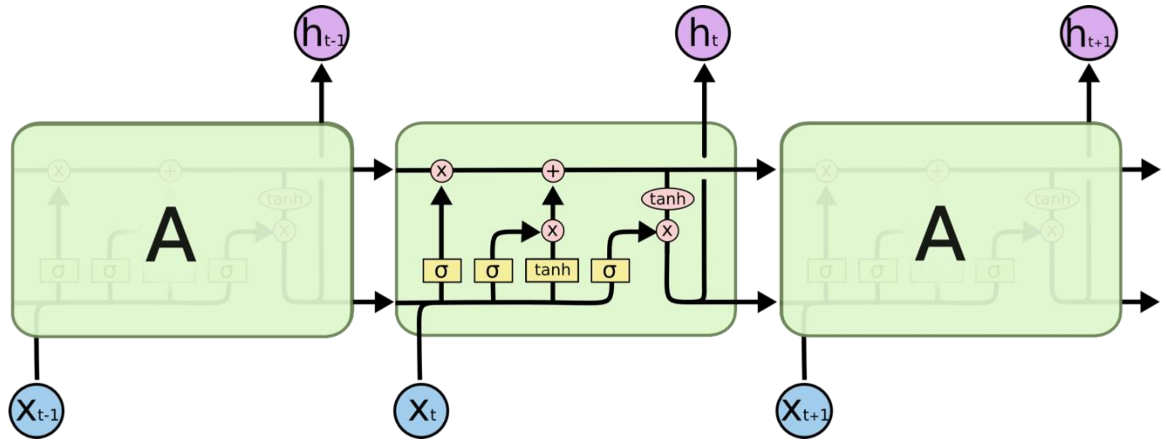
Long-short term memory (LSTM) networks



LSTM

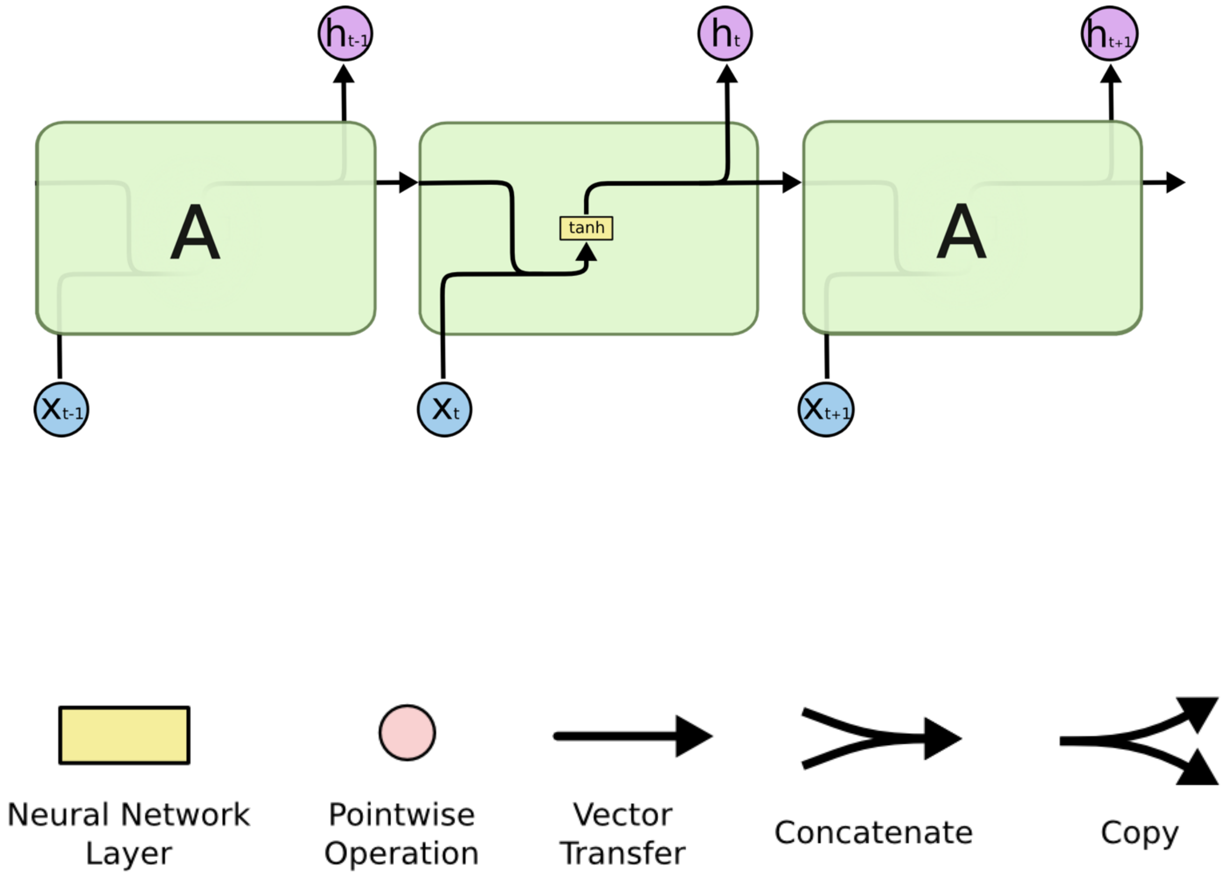


h t-1
h t
h t+1
A
tanh
A
X t-1
X t
X t+1



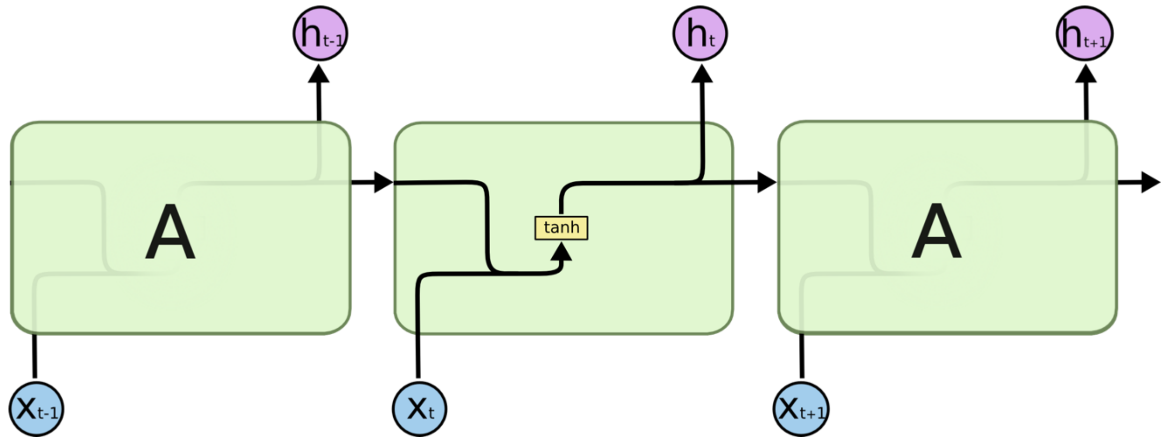
h t-1
h t
h t+1
A
x
+
tanh
x
x
σ
σ
tanh
σ
A
X t-1
X t
X t+1

LSTM

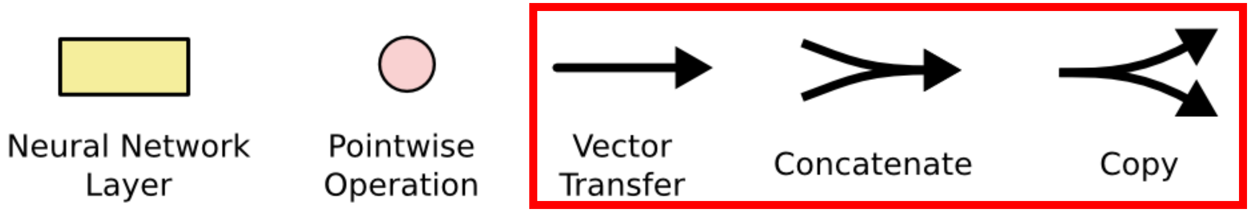


h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM

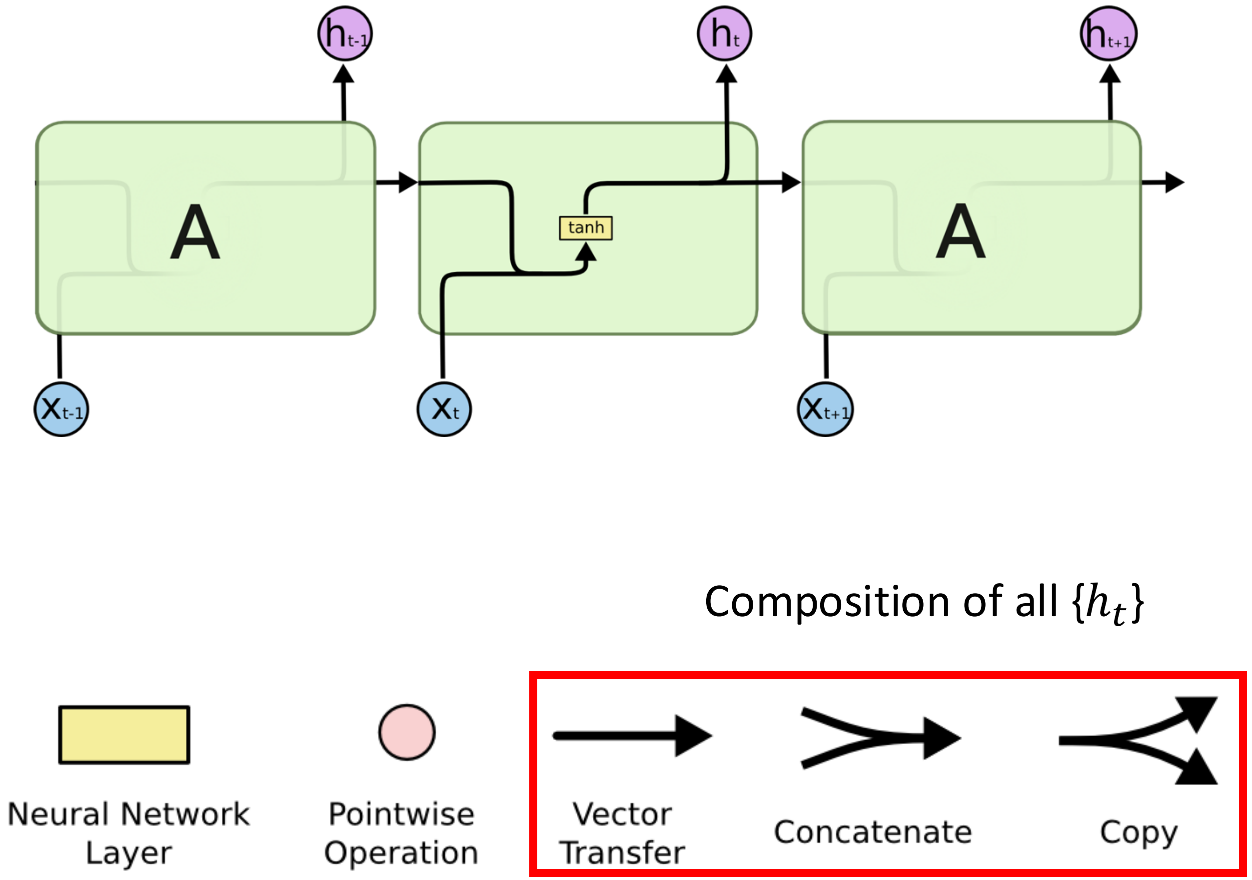


h_{t-1}
h_t
h_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}



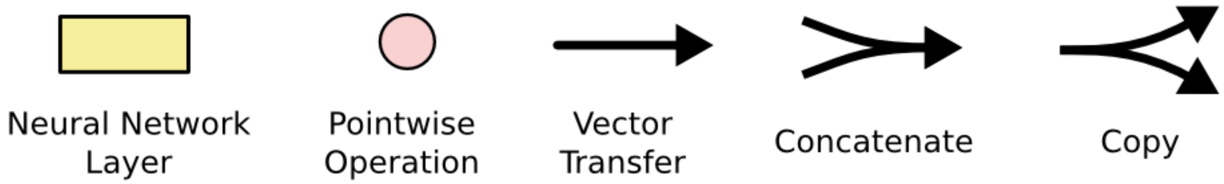
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

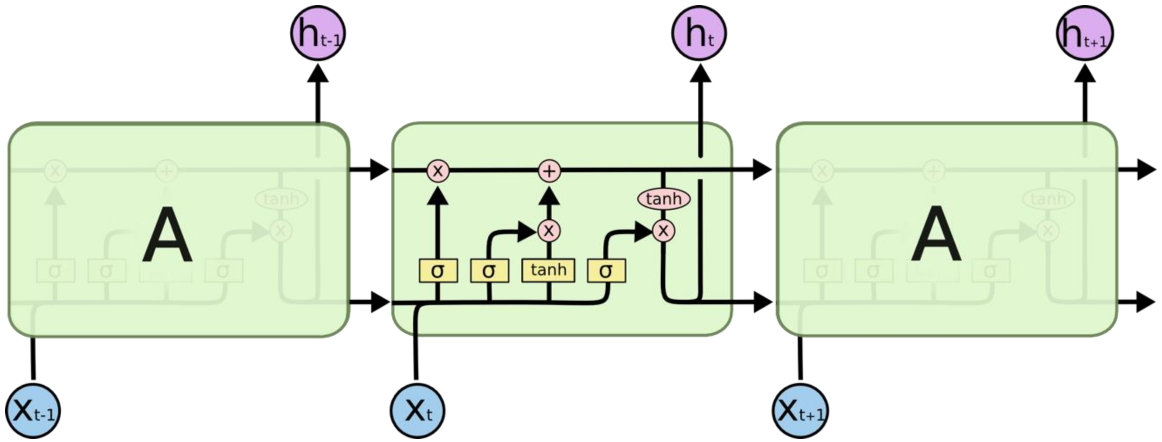
LSTM



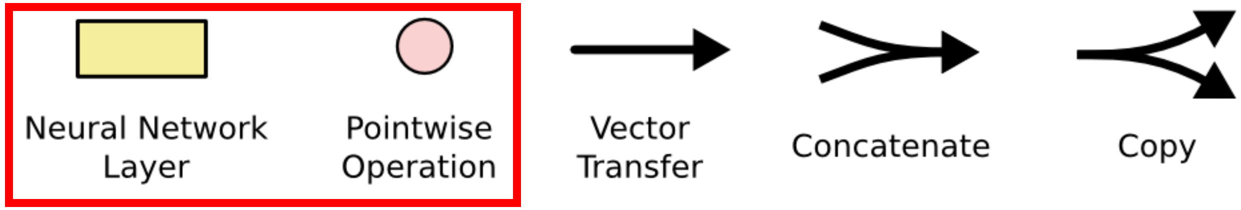
Composition of all $\{h_t\}$

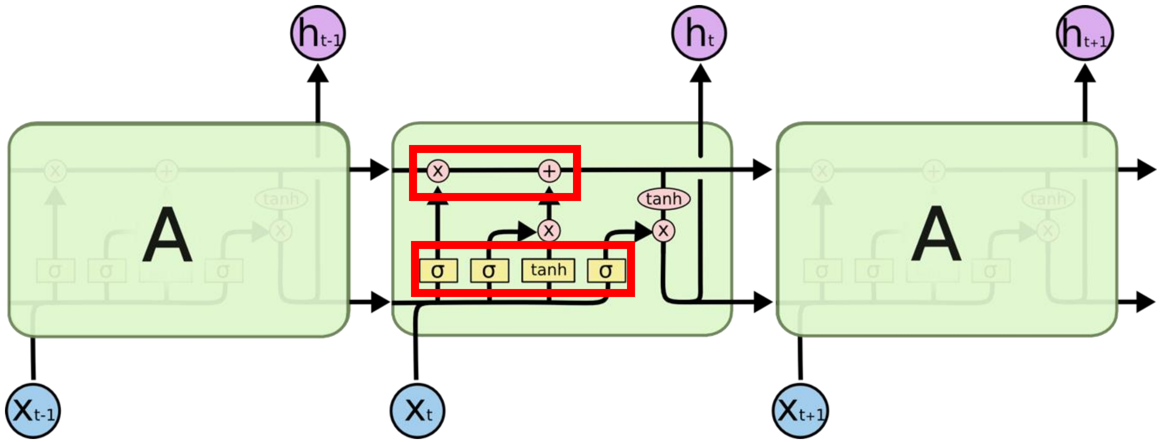
LSTM


Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

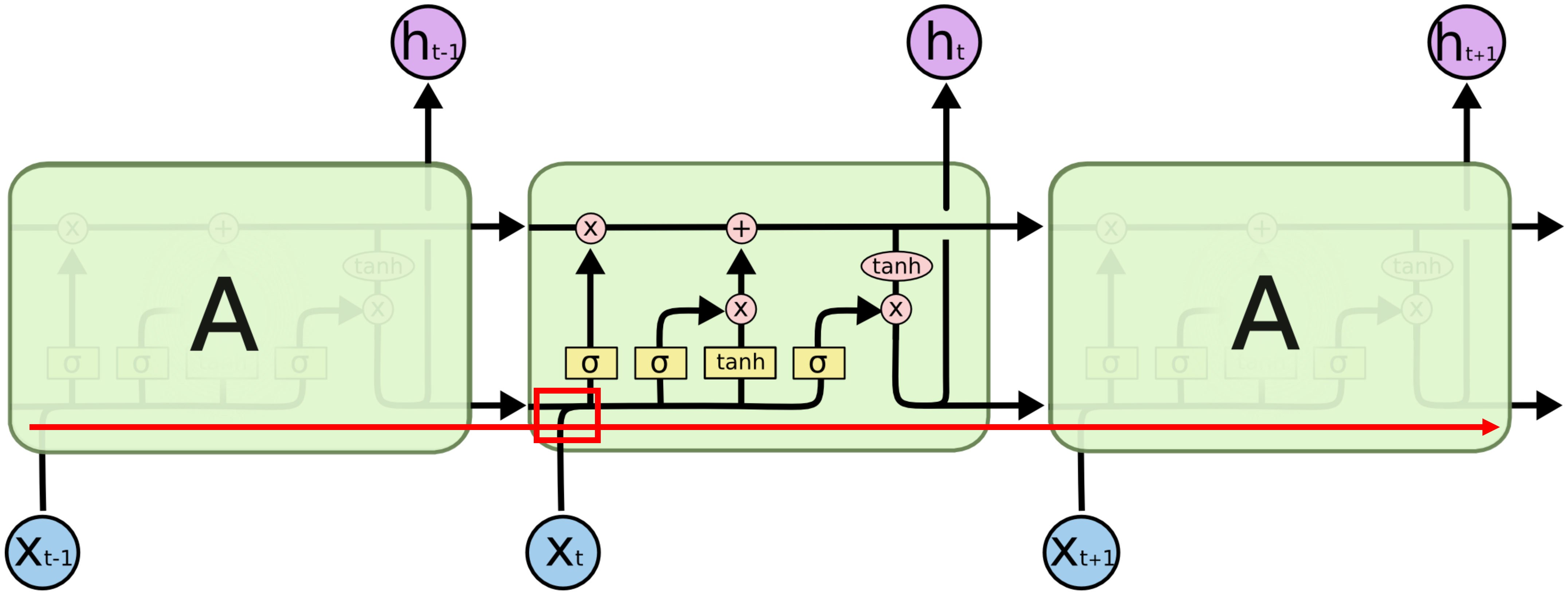

A
σ
σ
tanh
σ
tanh
A

LSTM

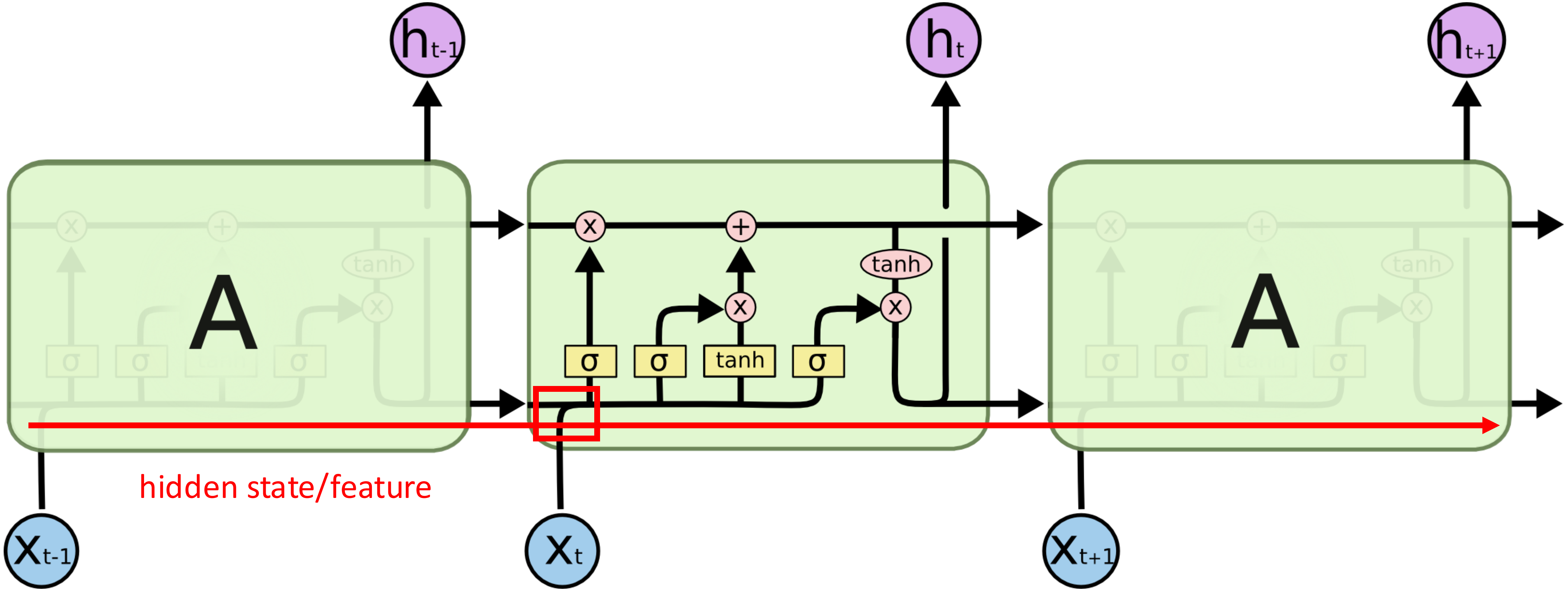

Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy


A
A

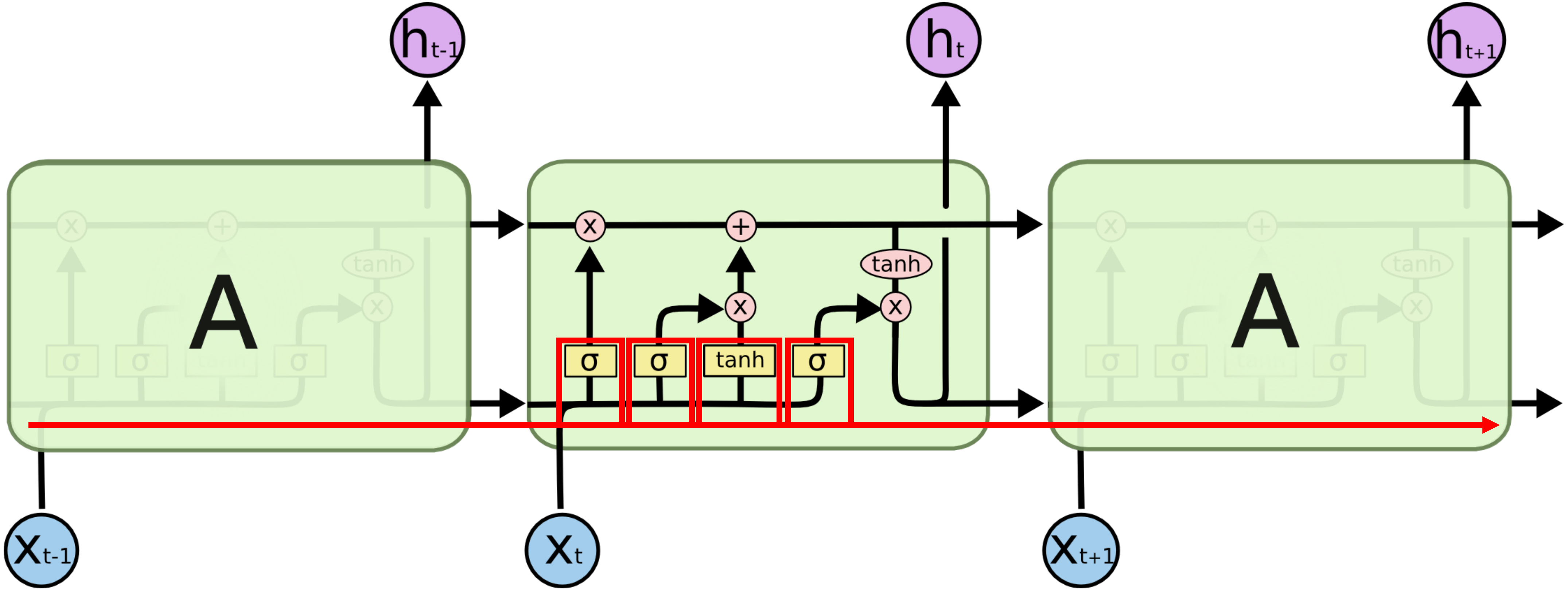
LSTM



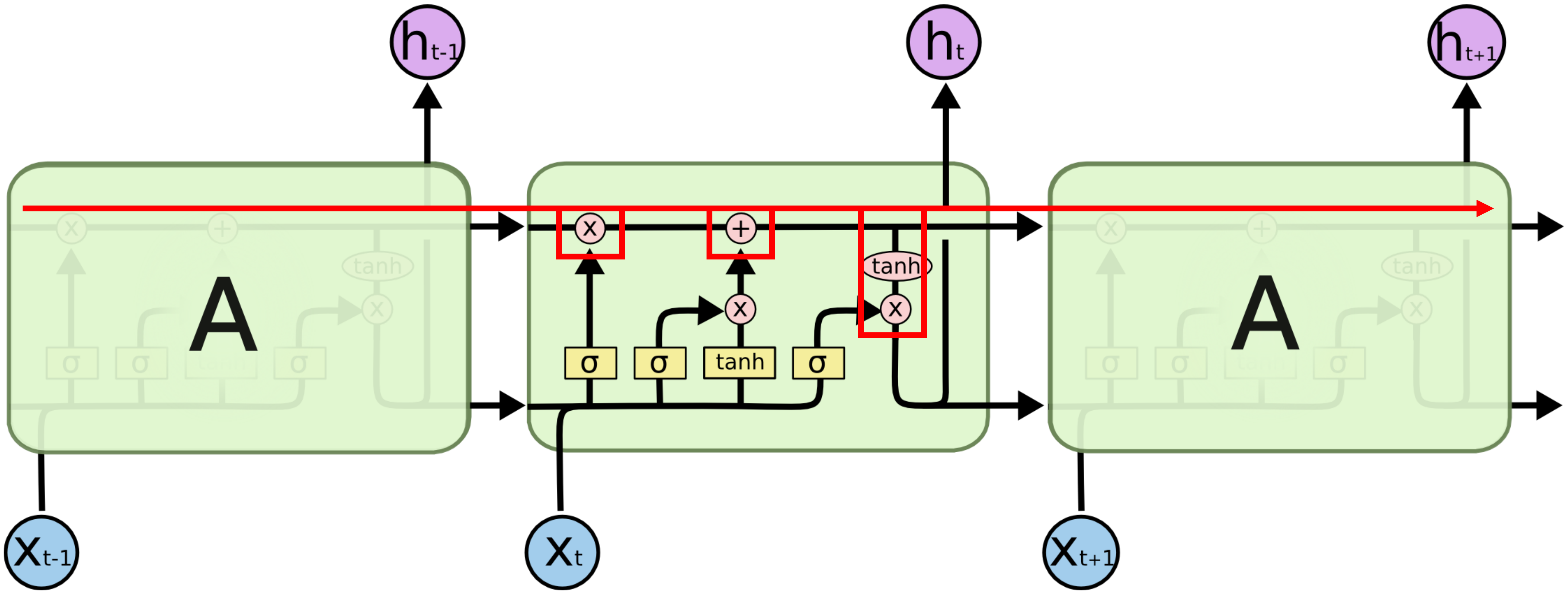
LSTM



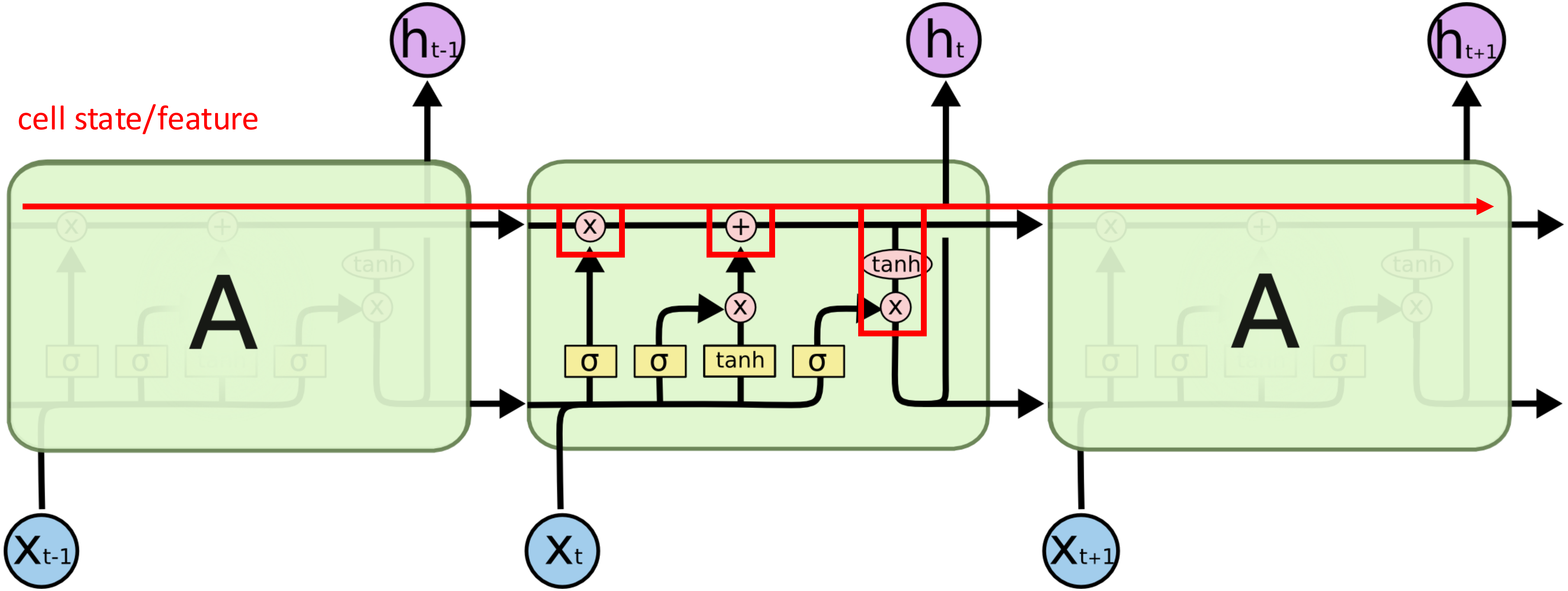
LSTM



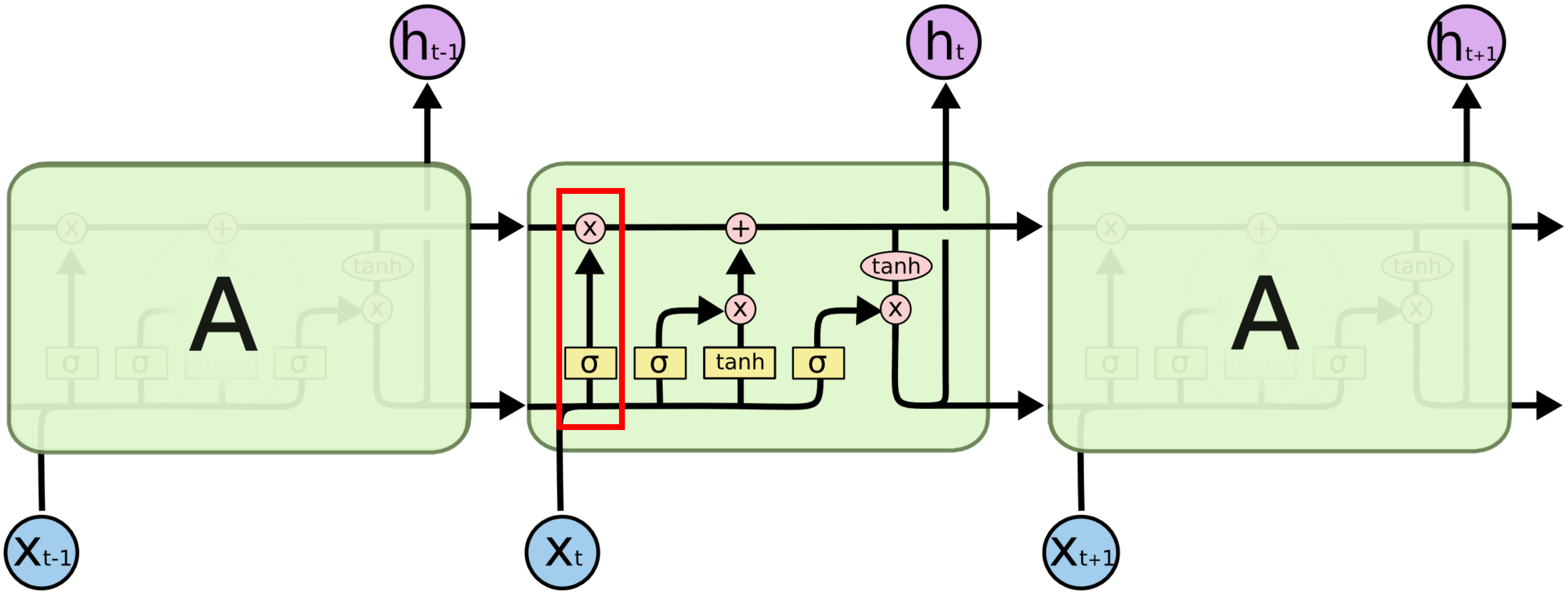
LSTM



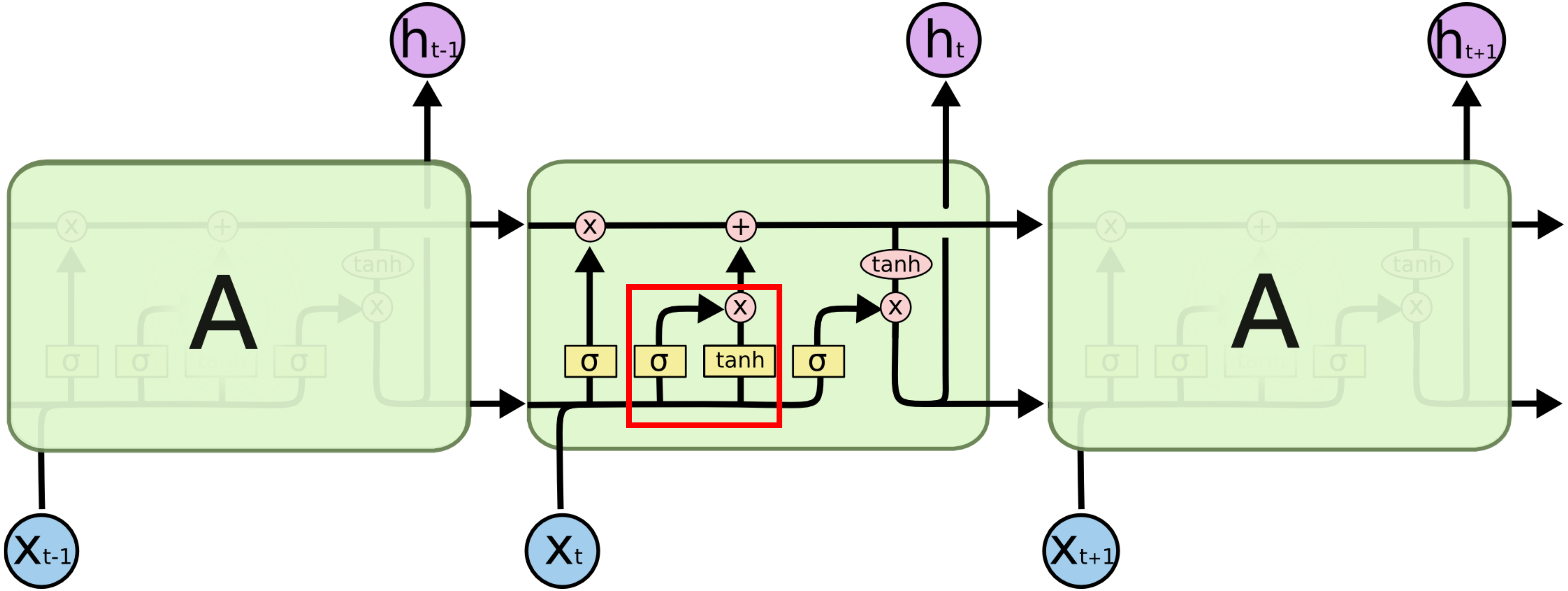
LSTM



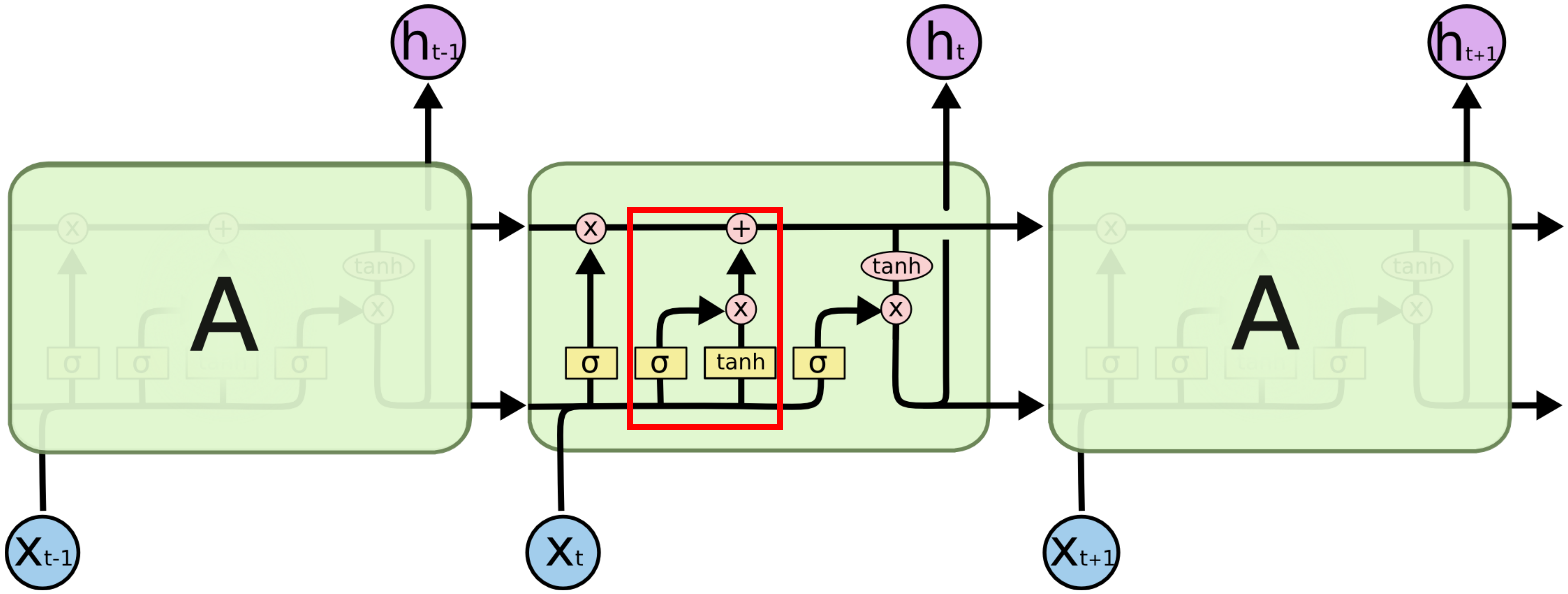
LSTM



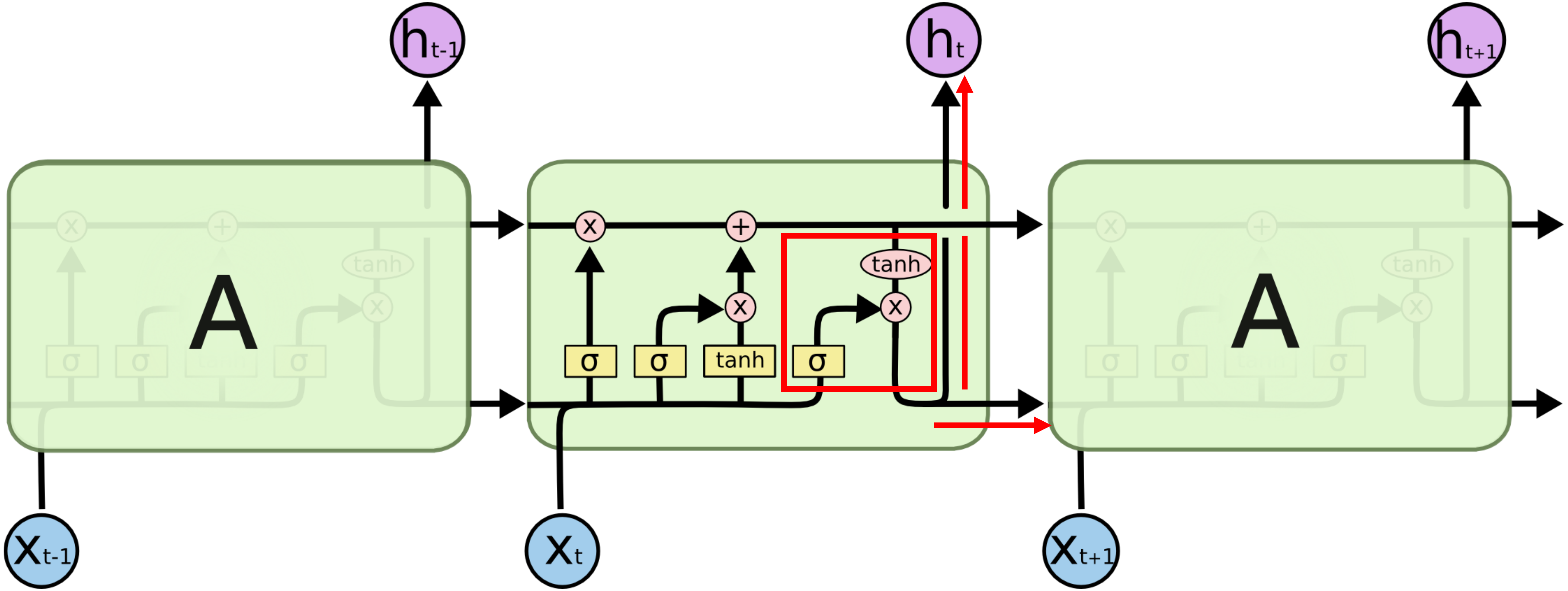
LSTM



LSTM

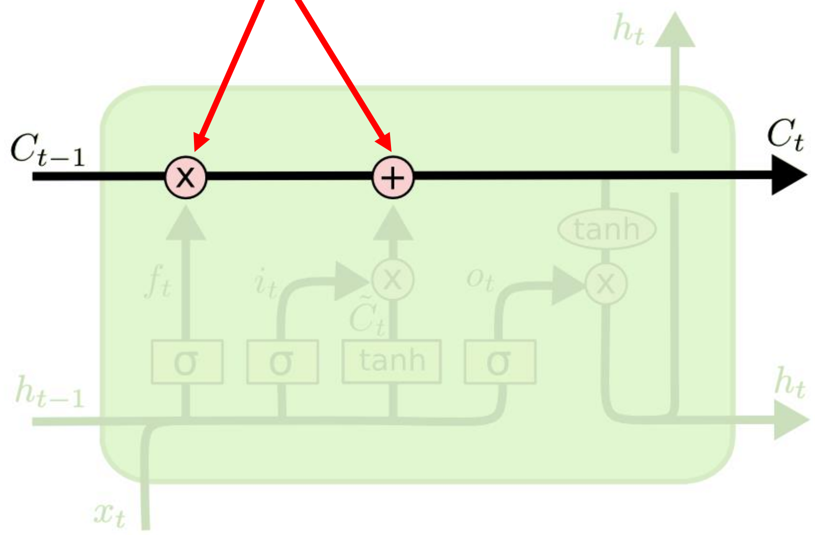


LSTM

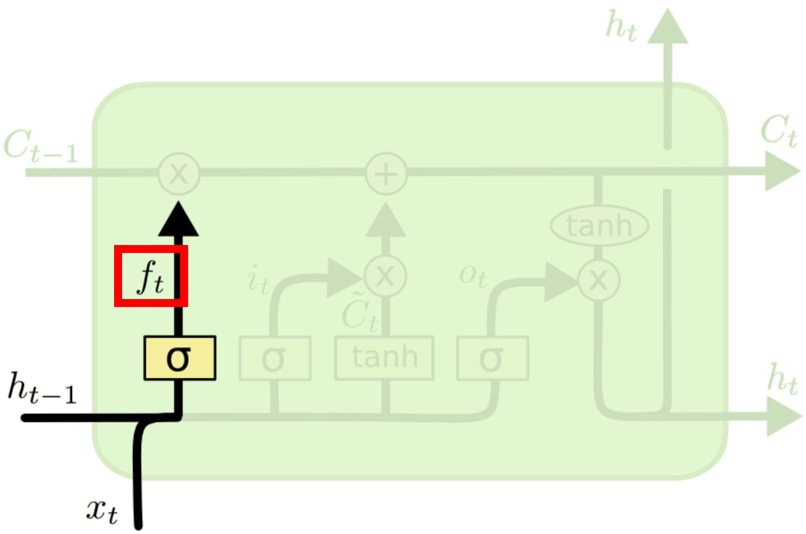


LSTM

Elementwise multiplication/addition

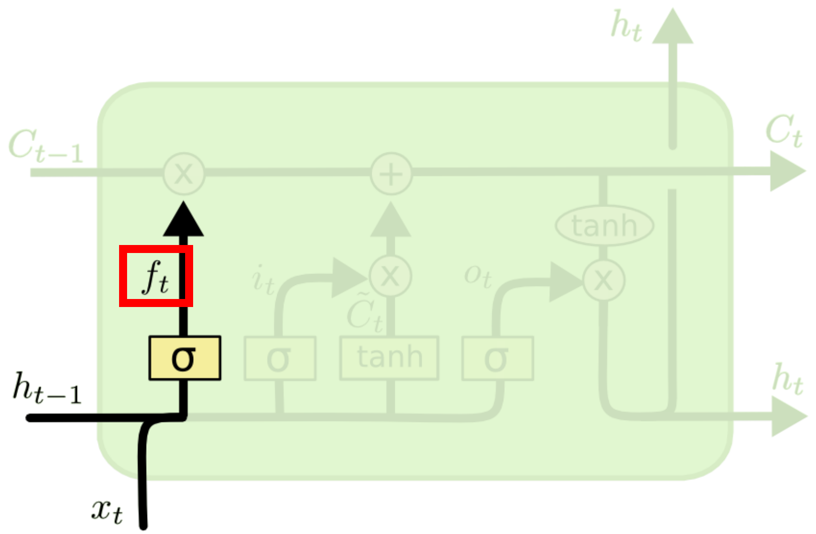


LSTM

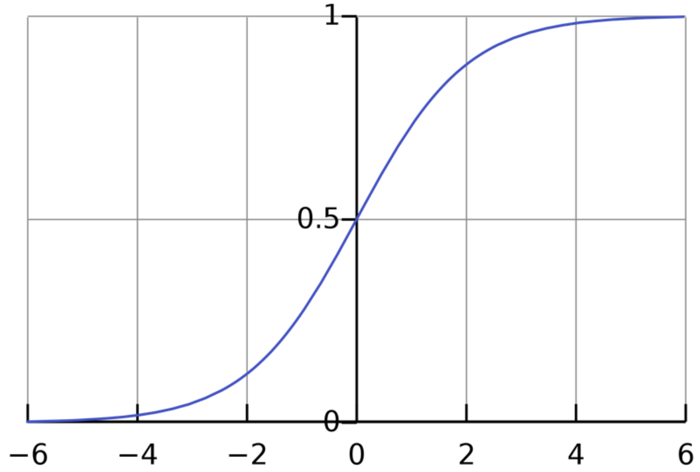


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

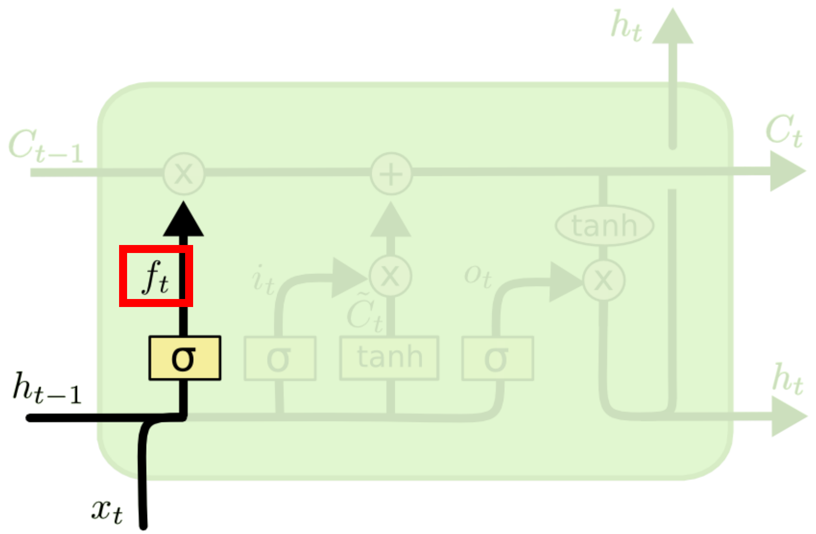
LSTM



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

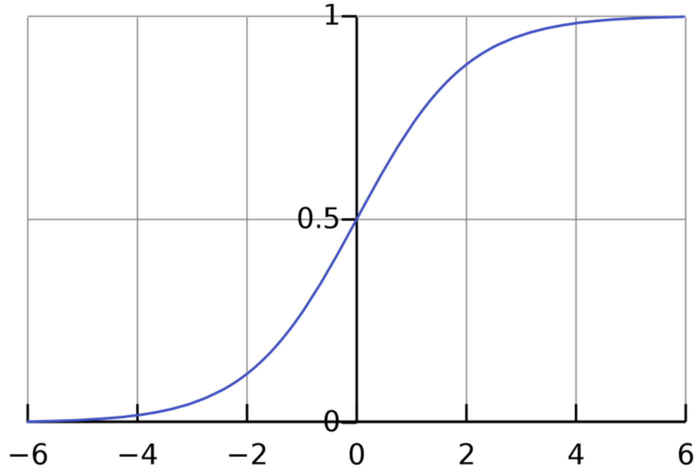


LSTM

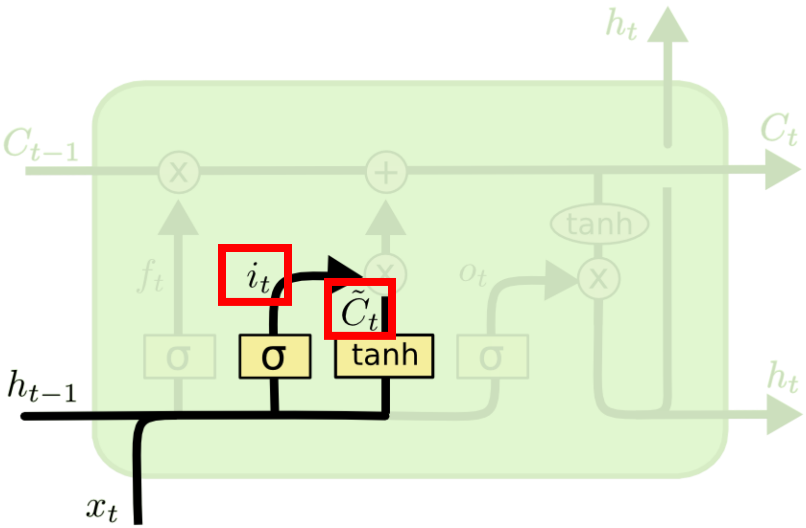


Forget gate:
Whether to erase cell

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



LSTM



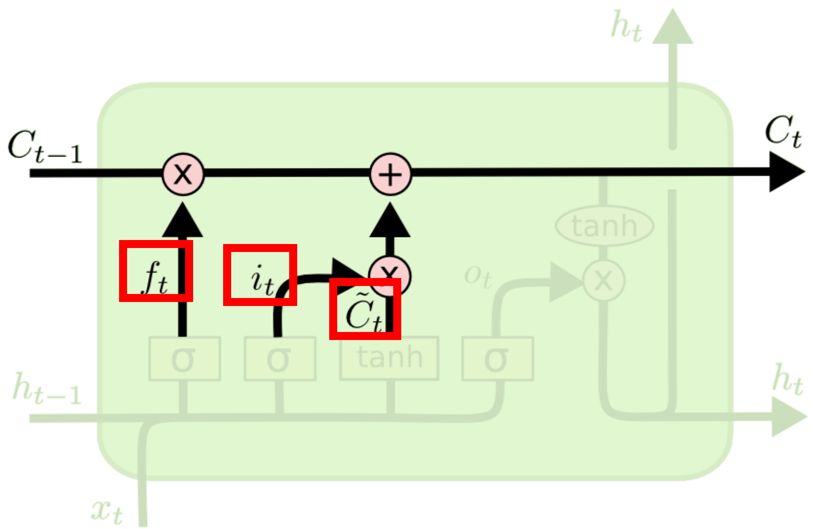
Input gate

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

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

cell input activation vector

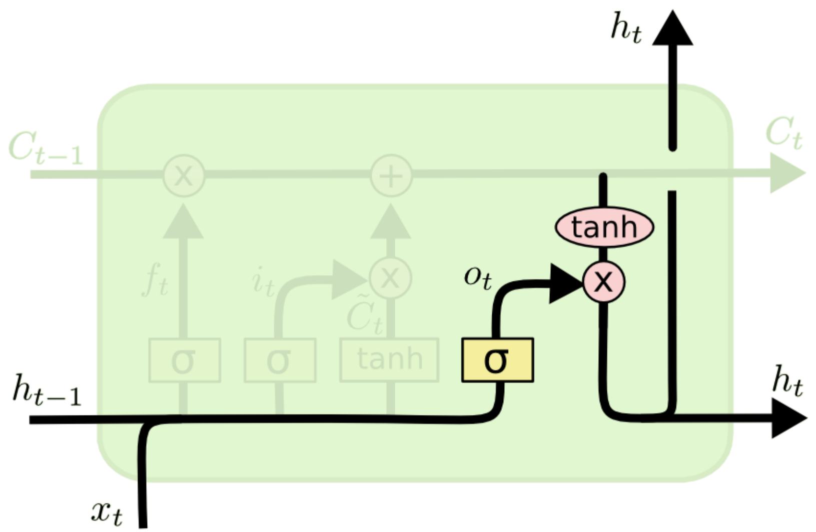
LSTM



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

Cell state

LSTM

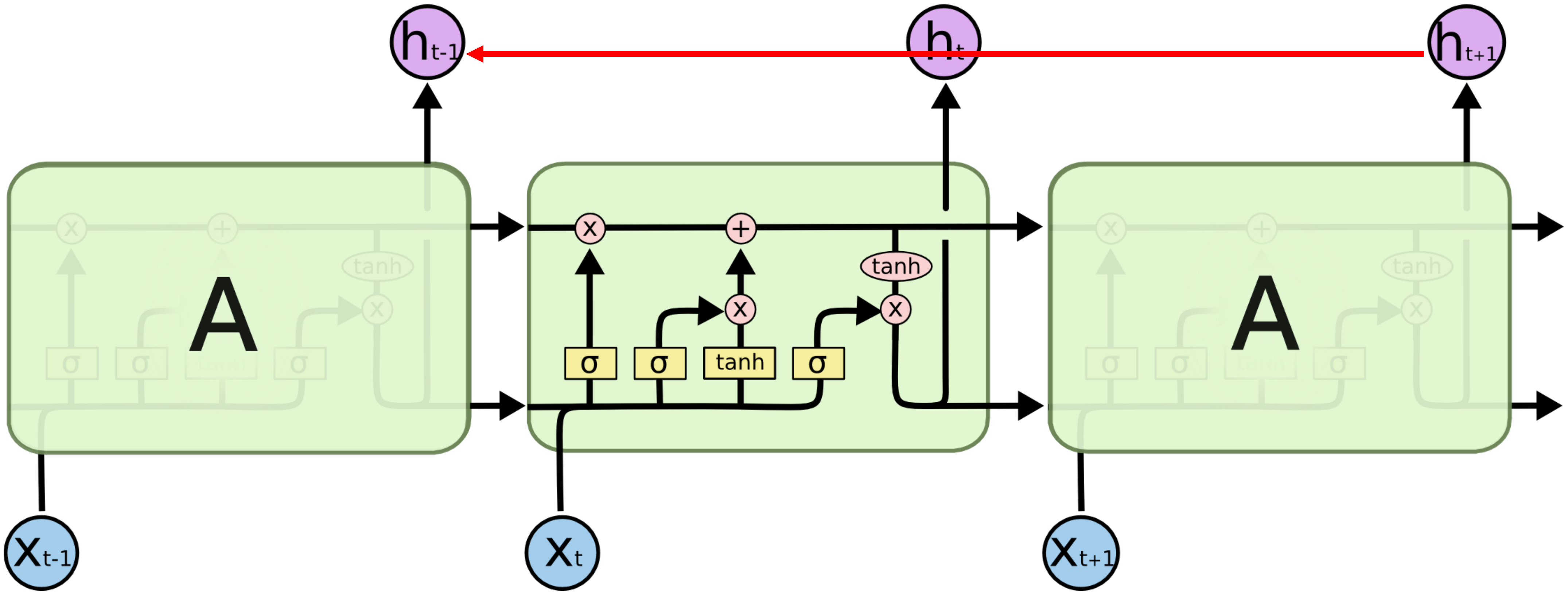


Output gate

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

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

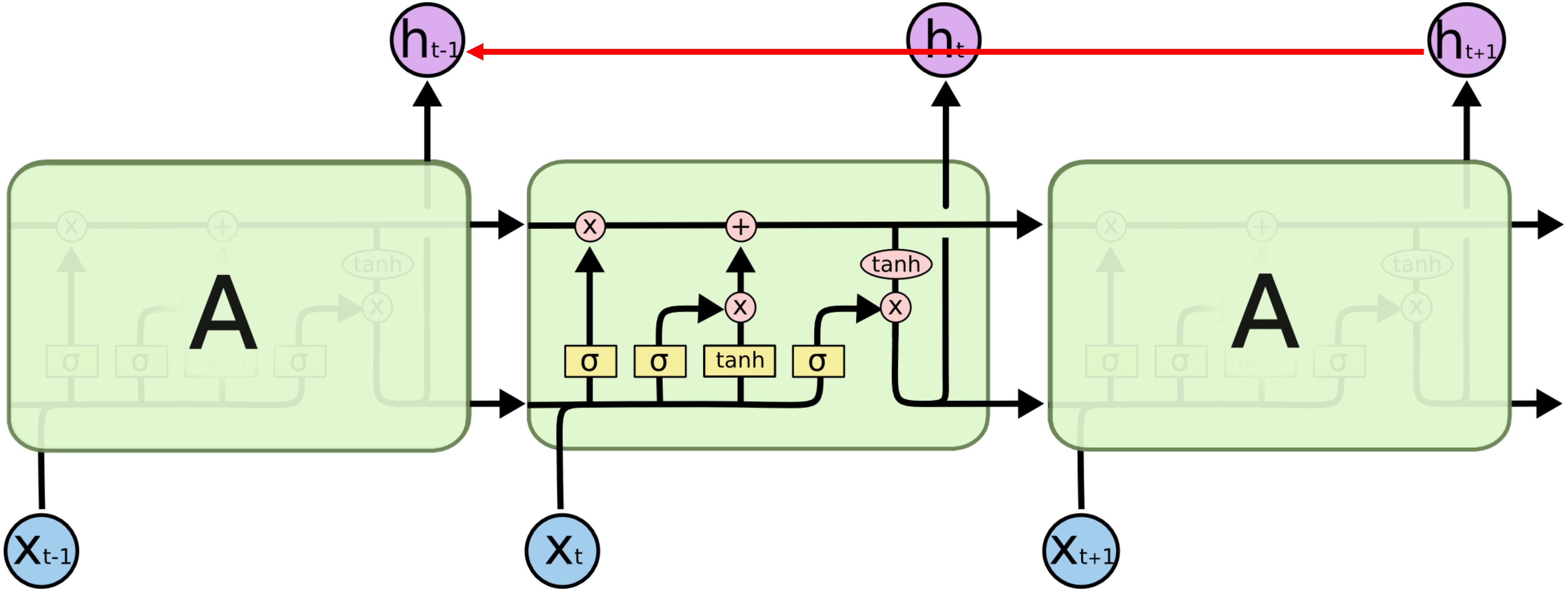
LSTM



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

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

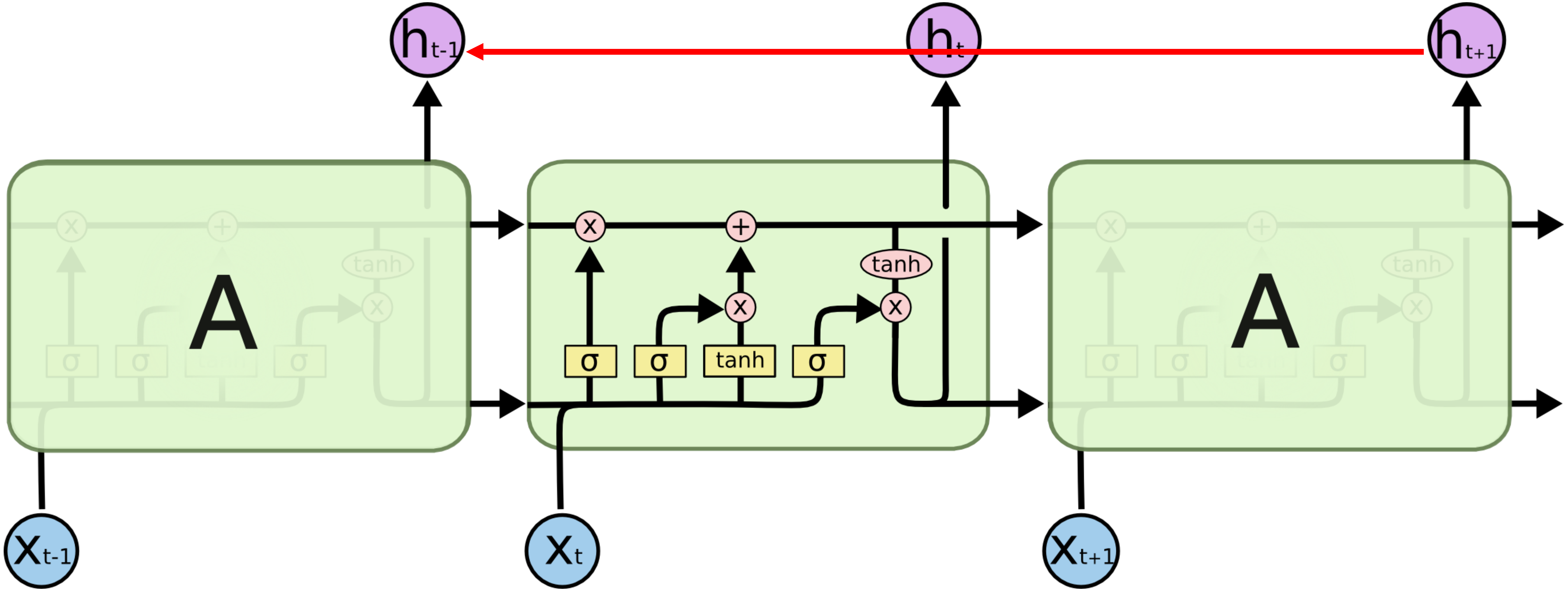
LSTM



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

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

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

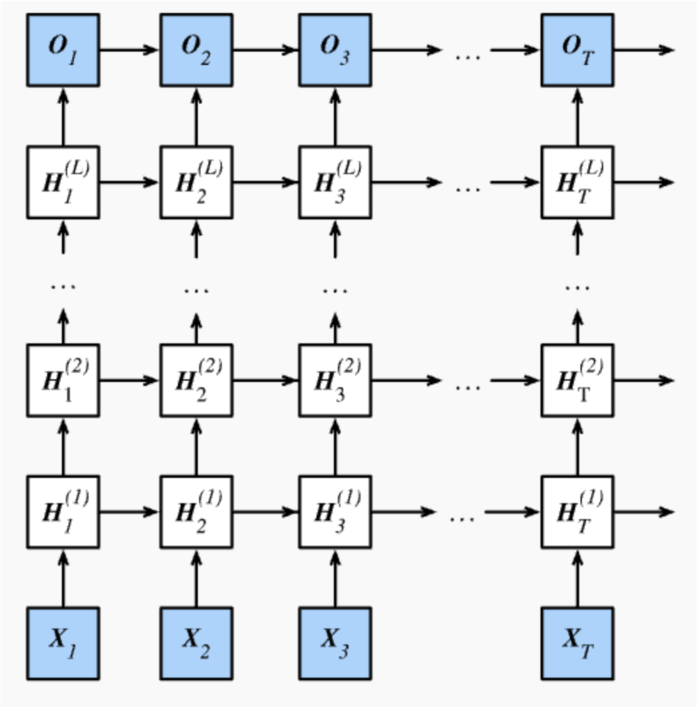


Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Deep RNN

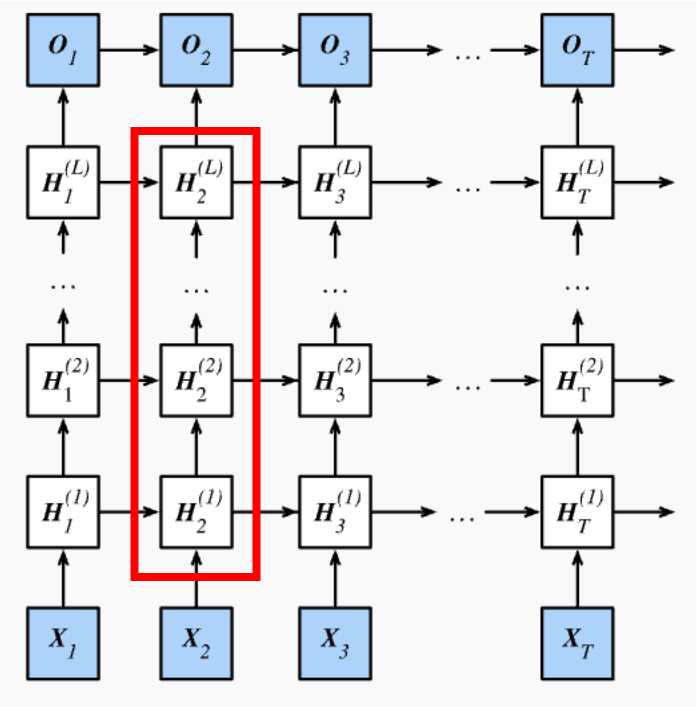
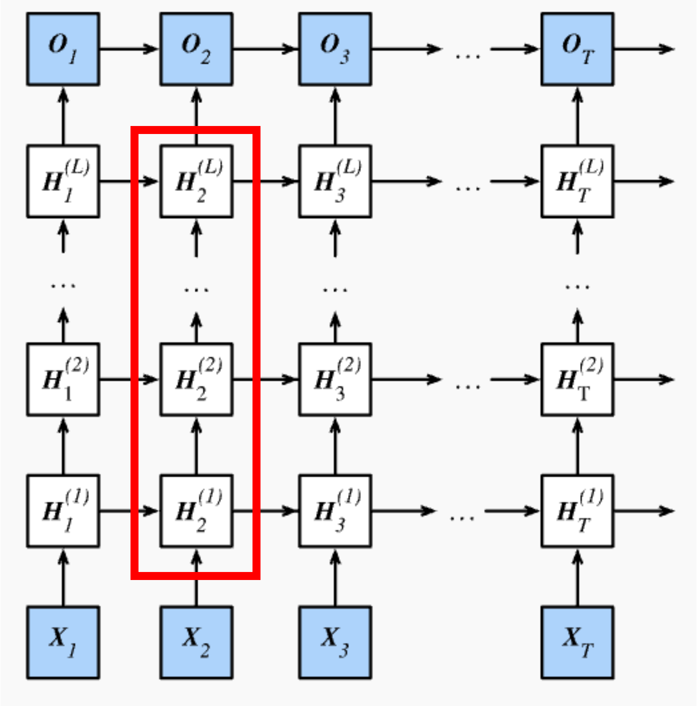


Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Deep RNN



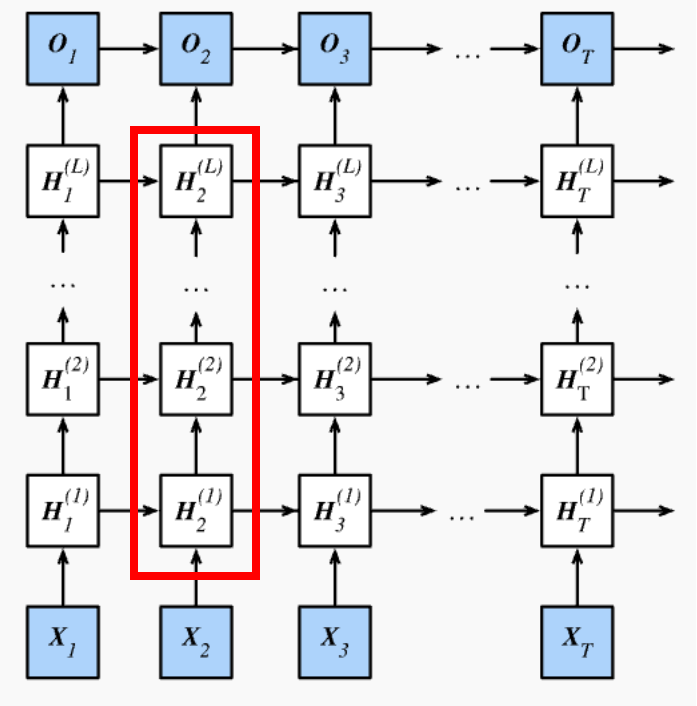
For each unit: a mapping

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

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

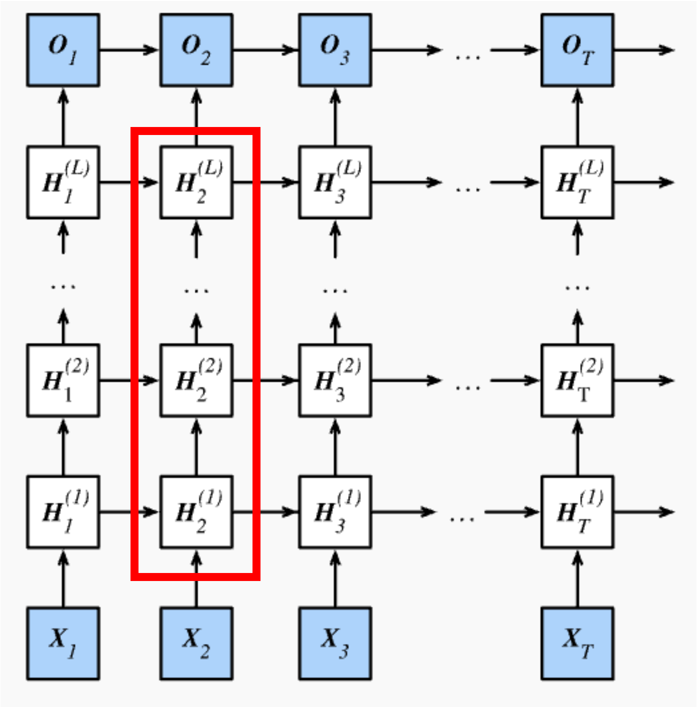
$$\text{RNN} : h_t^{l-1}, h_{t-1}^{l} \rightarrow h_t^l \qquad h_t^l \in \mathbb{R}^n$$

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

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

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

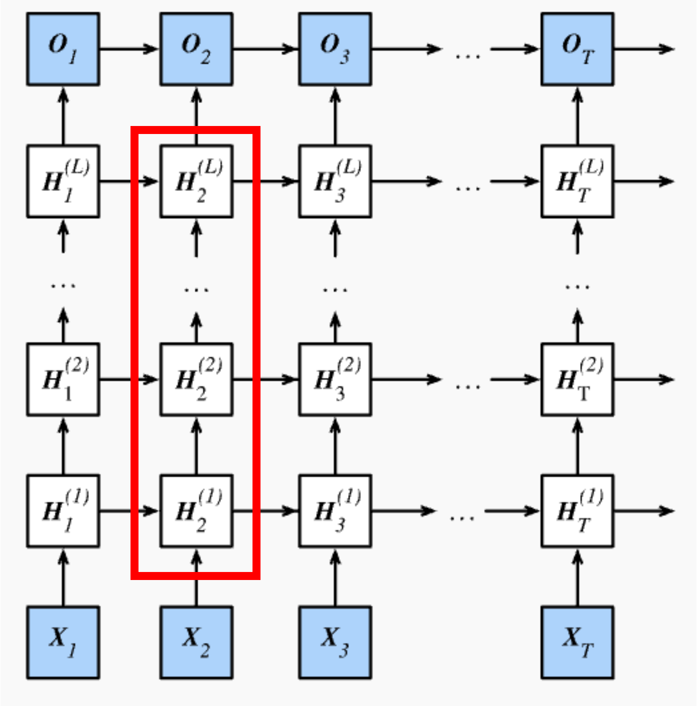
$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

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

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

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

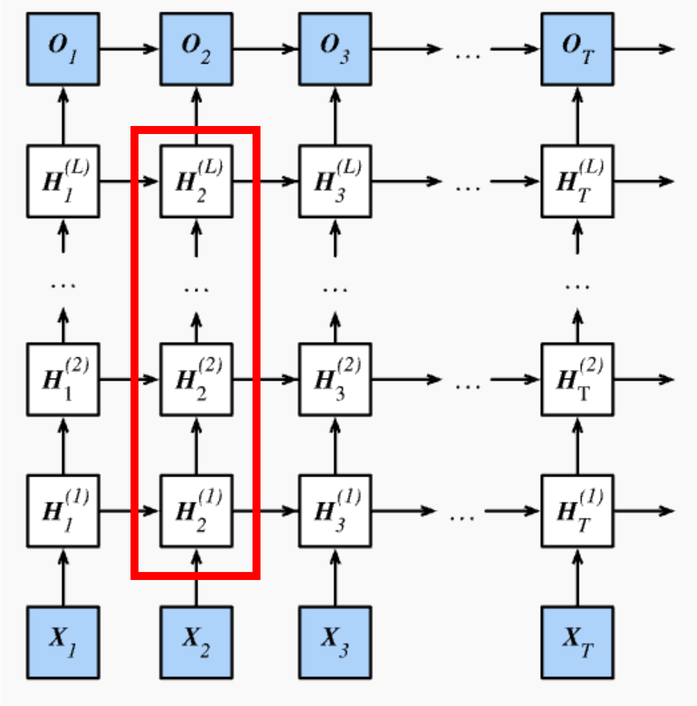
$$h_t^l = f(T_{n,n} h_t^{l-1} + T_{n,n} h_{t-1}^l), \text{ where } f \in \{\text{sigm}, \tanh\}$$

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

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

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

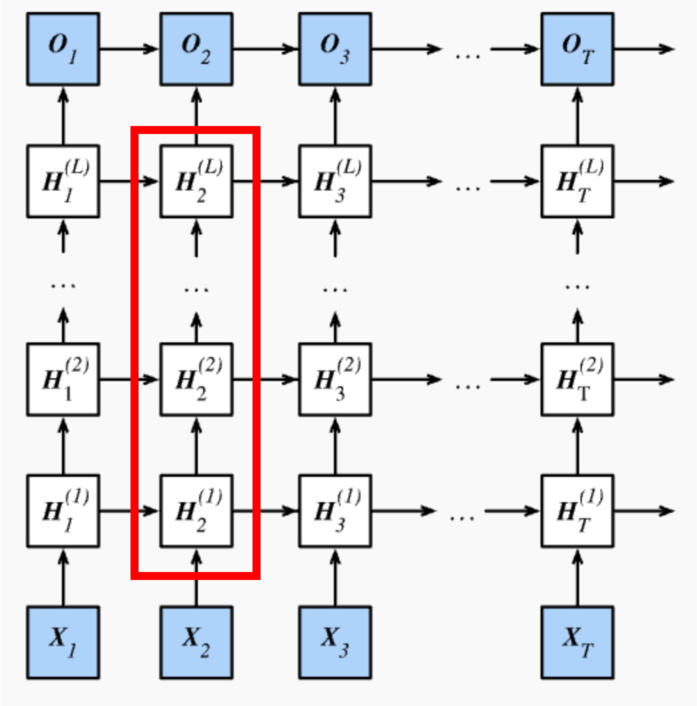
$T_{n,m} : \mathbb{R}^n \to \mathbb{R}^m$ (a mapping function)

$Wx + b$

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

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

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

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

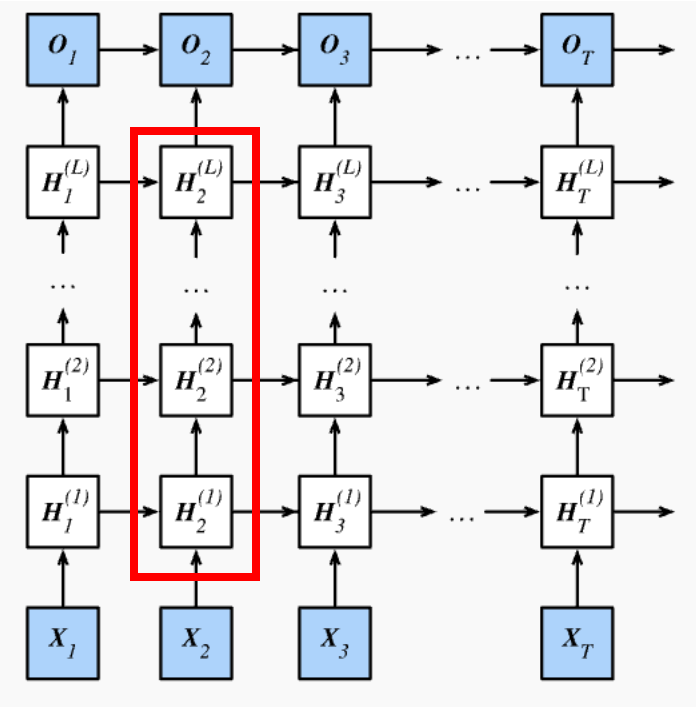
$$Wx + b$$

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

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

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

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

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

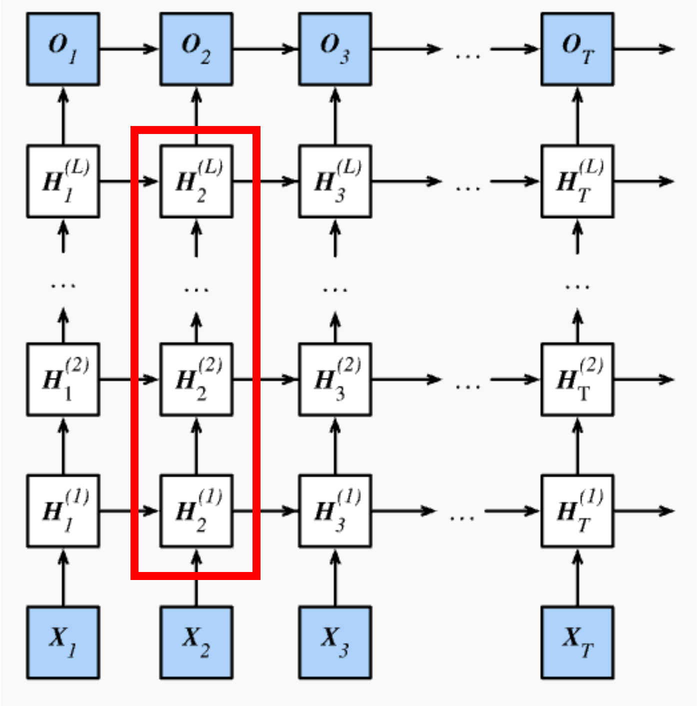
$$Wx + b$$

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

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

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

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

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

$$Wx + b$$

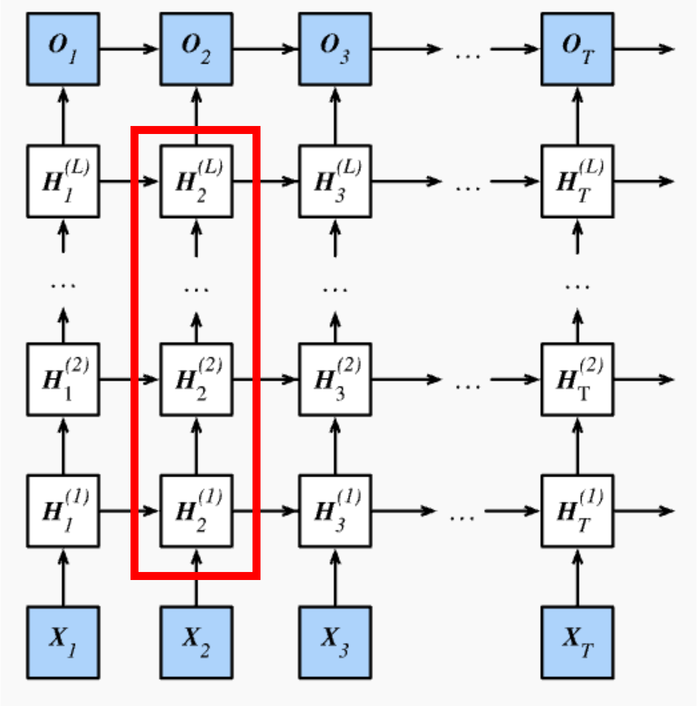
$x \in R^n, W \in R^{m \times n}, b \in R^m$

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

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



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)

$$Wx + b$$

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

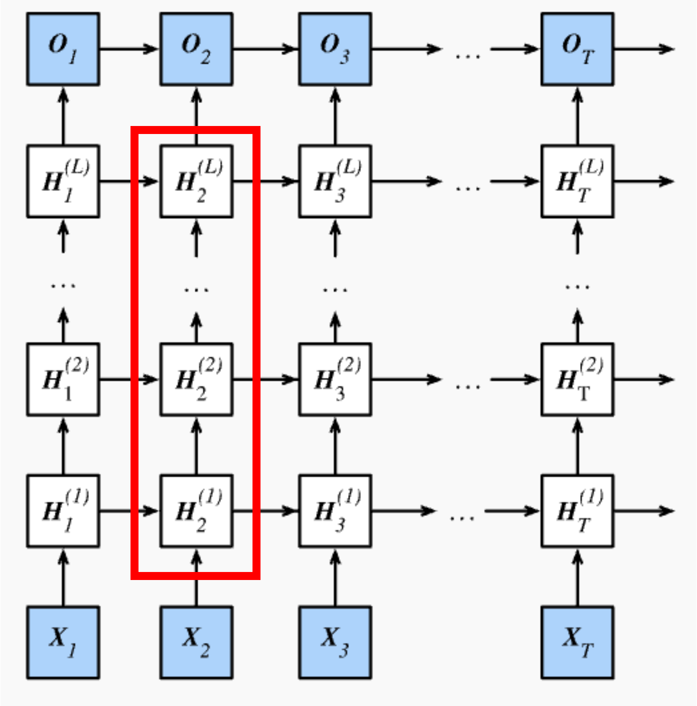
Image from Fig 8.10.1 of *Dive into Deep Learning* at https://classic.d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

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

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

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

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

$$Wx + b$$

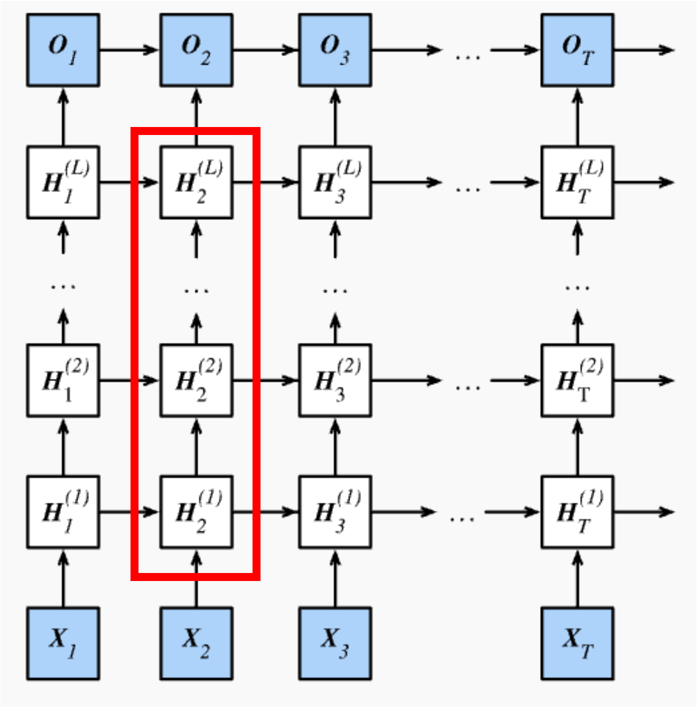
$x \in R^n, W \in R^{m \times n}, b \in R^m$

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

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



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

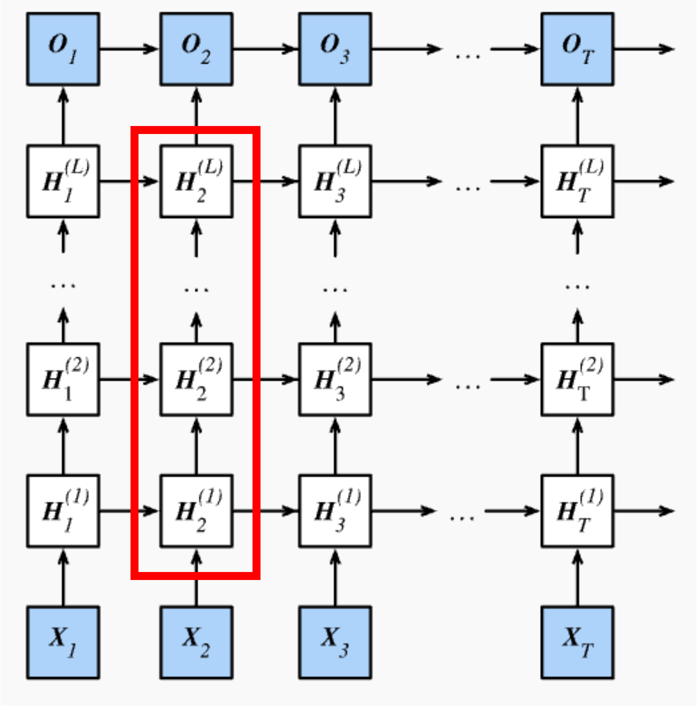
Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Deep RNN

elementwise operation

Vector feature

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



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)

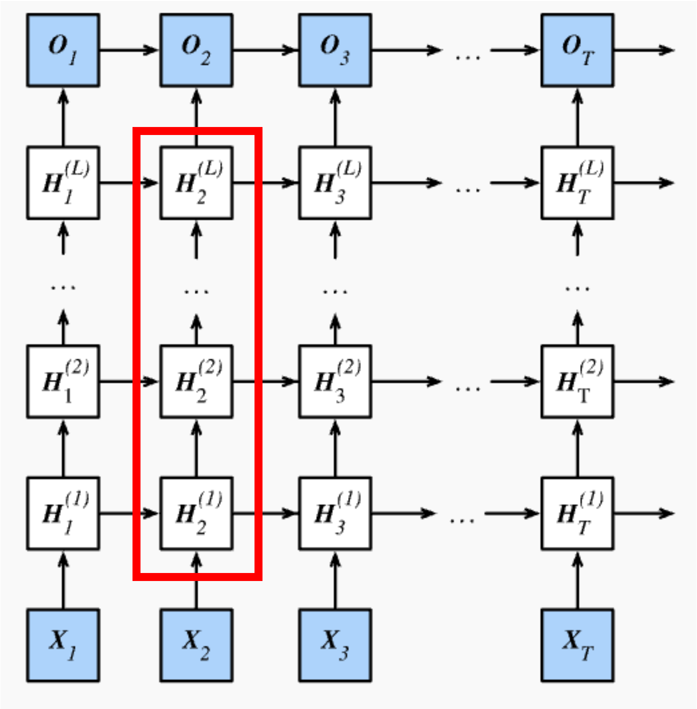
$$Wx + b$$

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

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

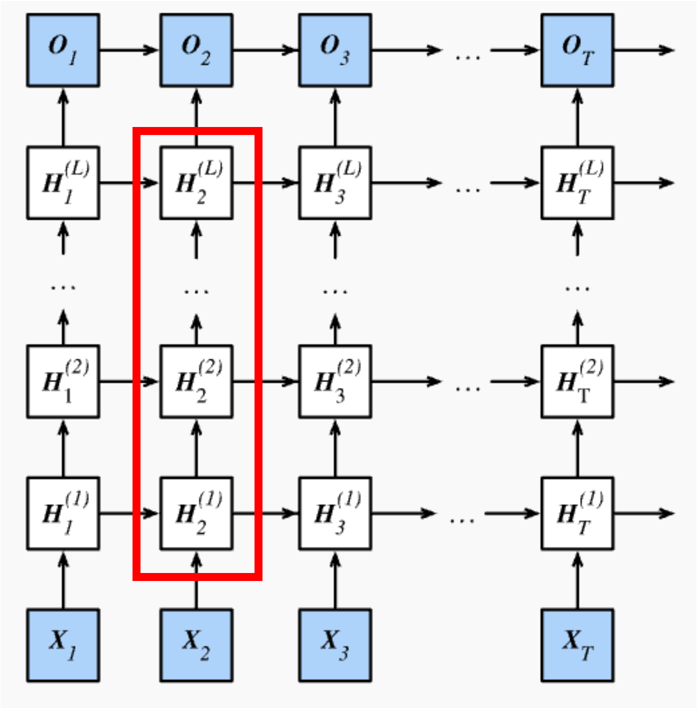


$$\text{LSTM} : h_t^{l-1}, h_{t-1}^l, c_{t-1}^l \to h_t^l, c_t^l$$

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



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

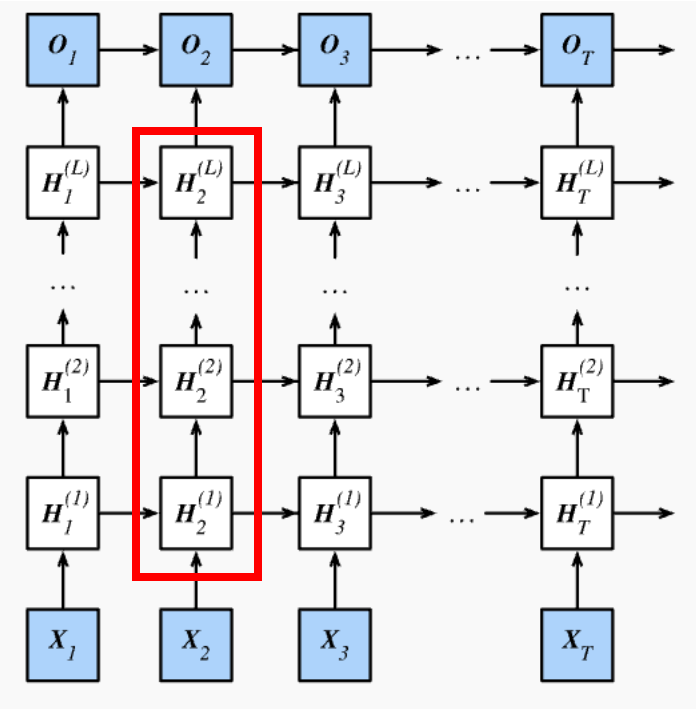
$$c_t^l \in \mathbb{R}^n$$

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

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



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

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

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

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

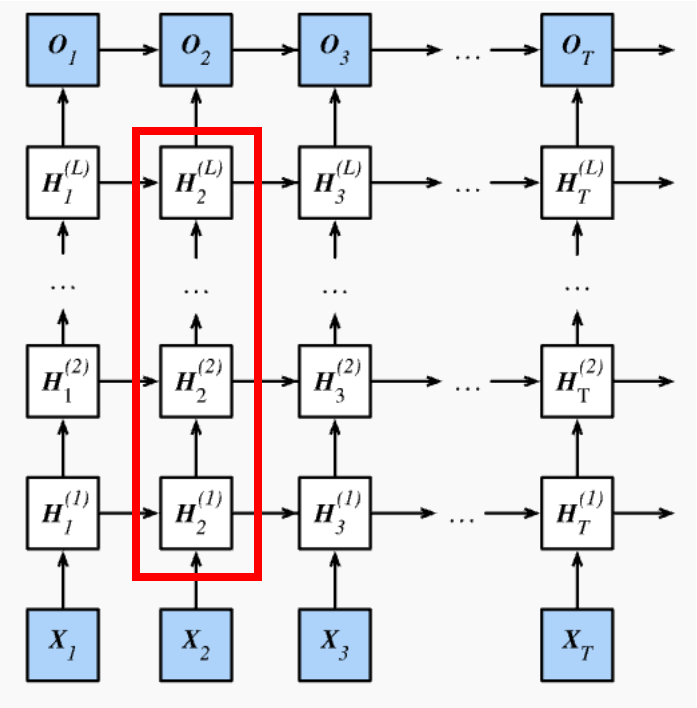
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

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

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



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

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

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

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

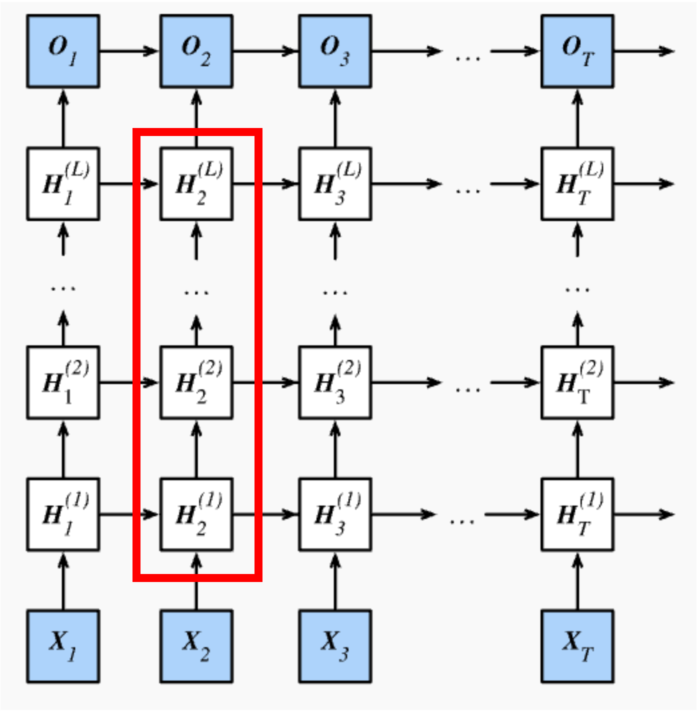
$$\boxed{c_t^l} = f \odot c_{t-1}^l + i \odot g$$

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

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



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

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

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

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

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

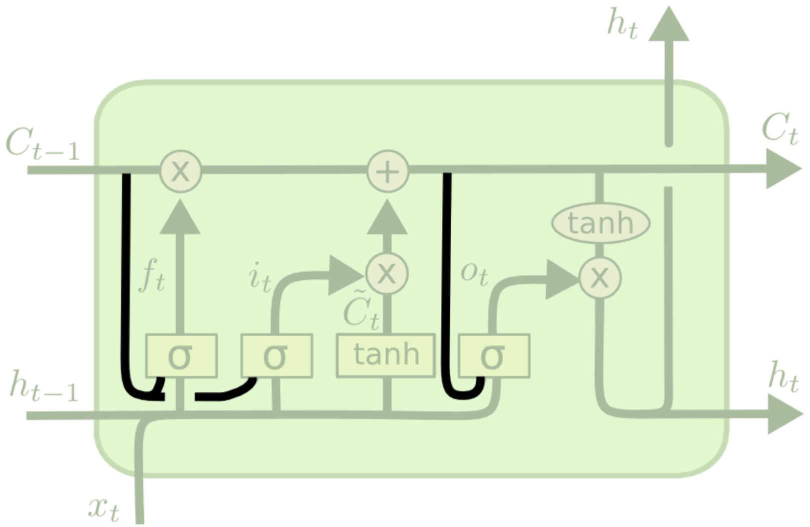
$$\boxed{h_t^l} = \boxed{o} \odot \tanh(c_t^l)$$

weights

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

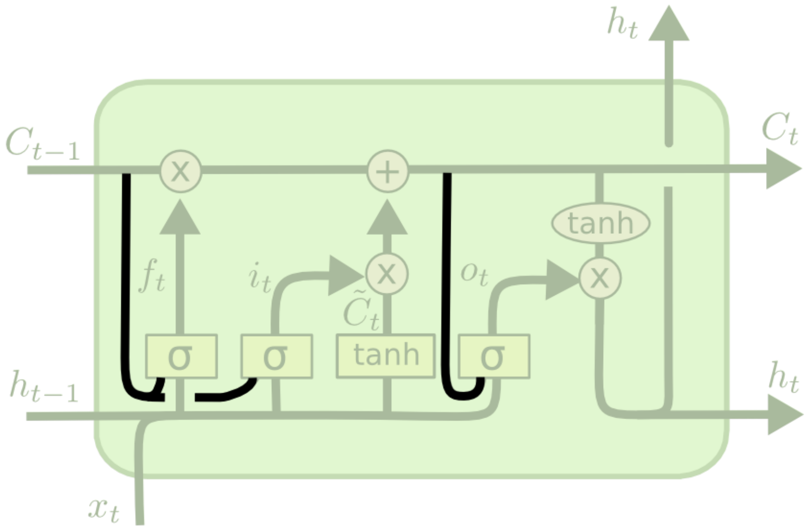
Formulations from: Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." *arXiv preprint arXiv:1409.2329* (2014).

Variant: peephole connections

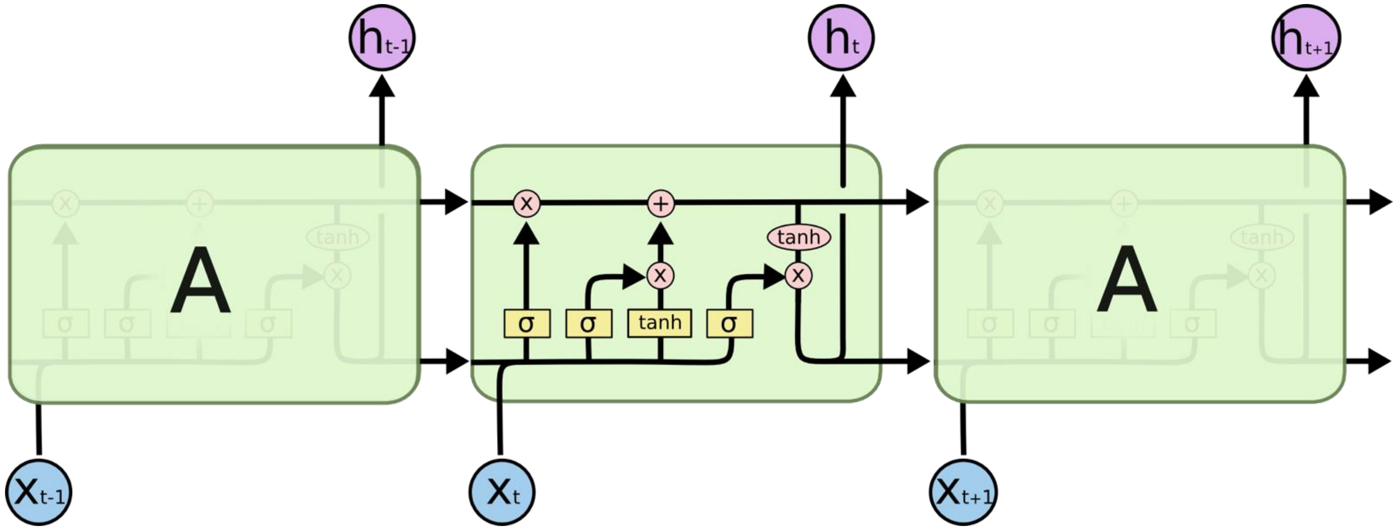


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

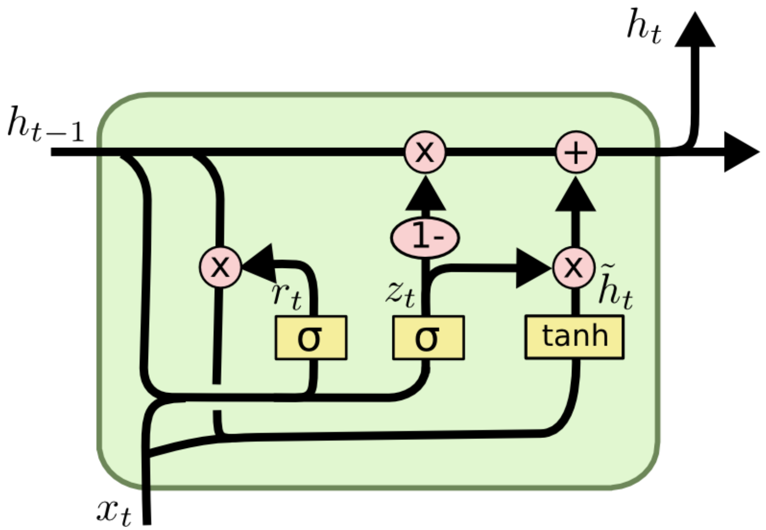
Variant: peephole connections



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



Variant: gated recurrent unit (GRU)



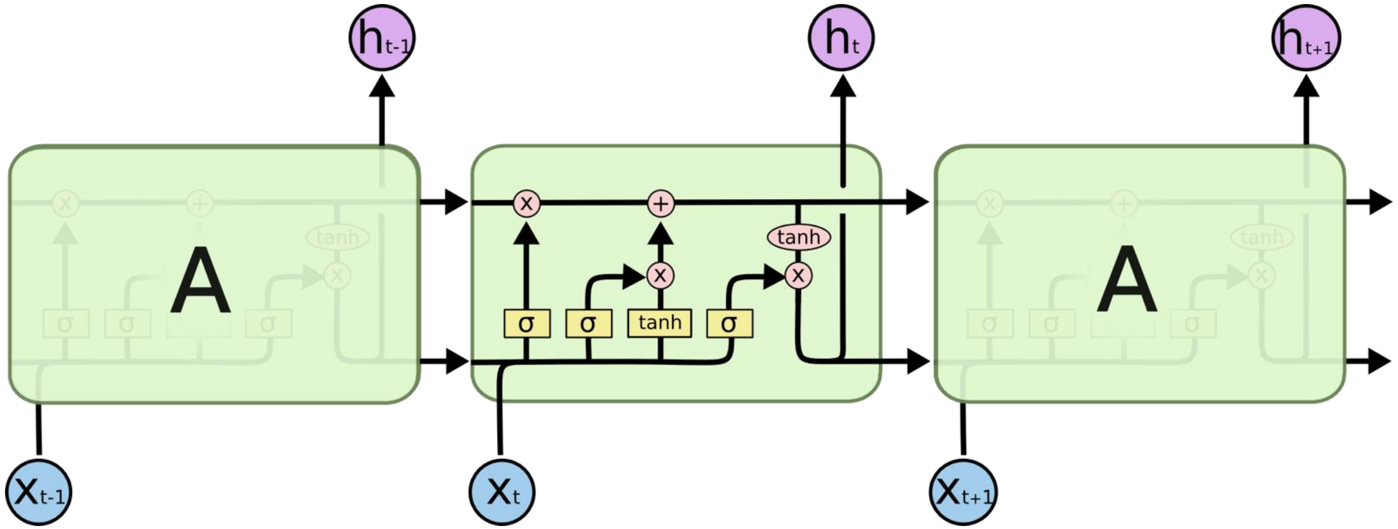
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

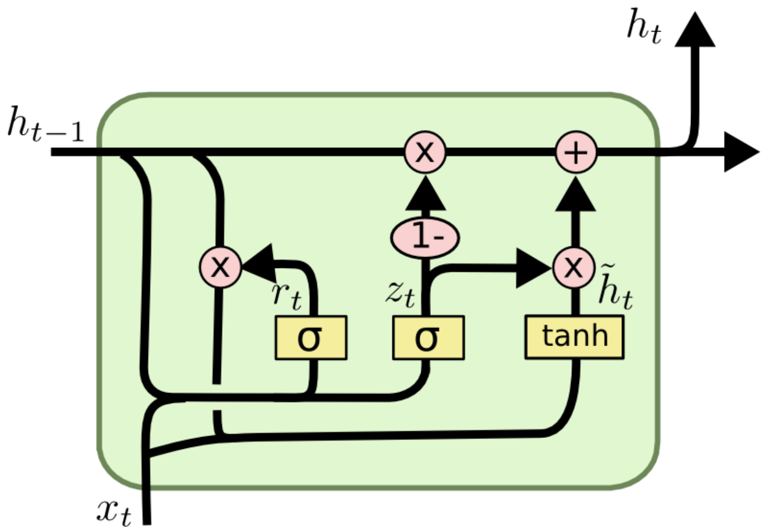
$$\boxed{\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)}$$

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

Weighted average



Variant: gated recurrent unit (GRU)



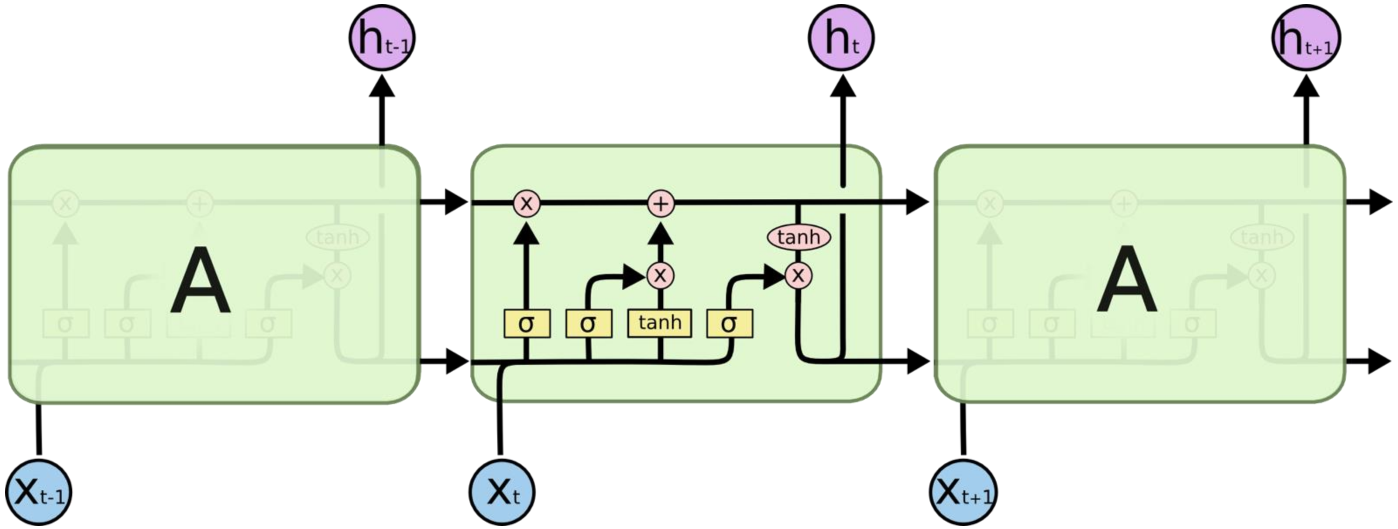
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

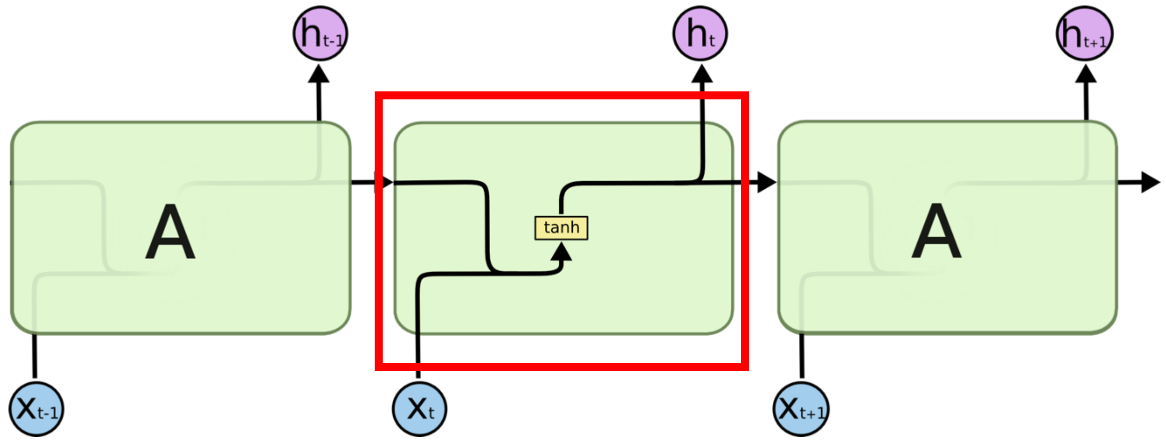
$$\boxed{\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)}$$

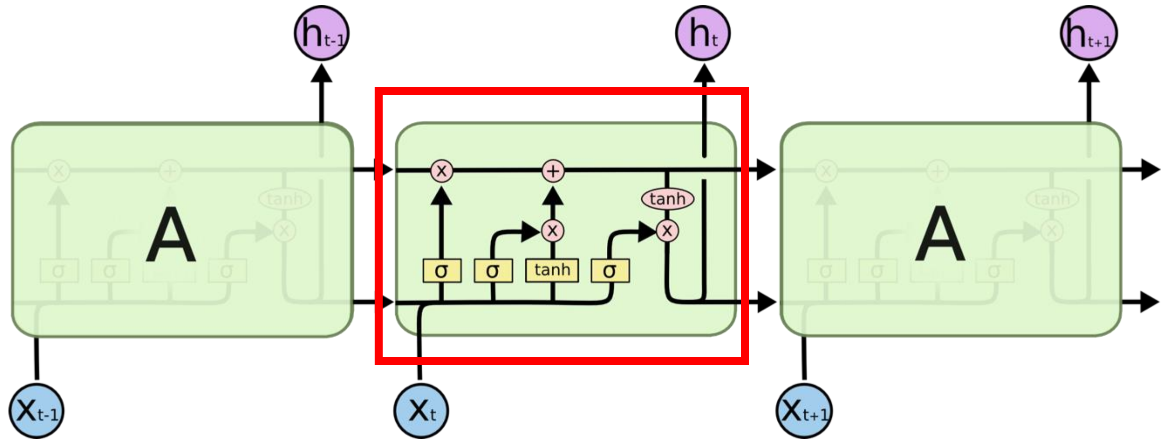
$$\boxed{h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t}$$

Weighted average



Vanilla RNN and LSTM





Generating baby names

Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacrie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa Wallon Martine Merus Jelen Candica Wallin Tel Rachene Tarine Ozila Ketia Shanne Arnande Karella Roselina Alessia Chasty Deland Berther Geamar Jackein Mellisand Sagdy Nenc Lessie Rasemy Guen Gavi Milea Anneda Margoris Janin Rodelin Zeanna Elyne Janah Ferzina Susta Pey Castina

Credit for https://github.com/wangshusen/DeepLearning/blob/master/Slides/9_RNN_5.pdf

Generating C code

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Credit for https://github.com/wangshusen/DeepLearning/blob/master/Slides/9_RNN_5.pdf

Generating academic articles

For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) =$ $\mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\dots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

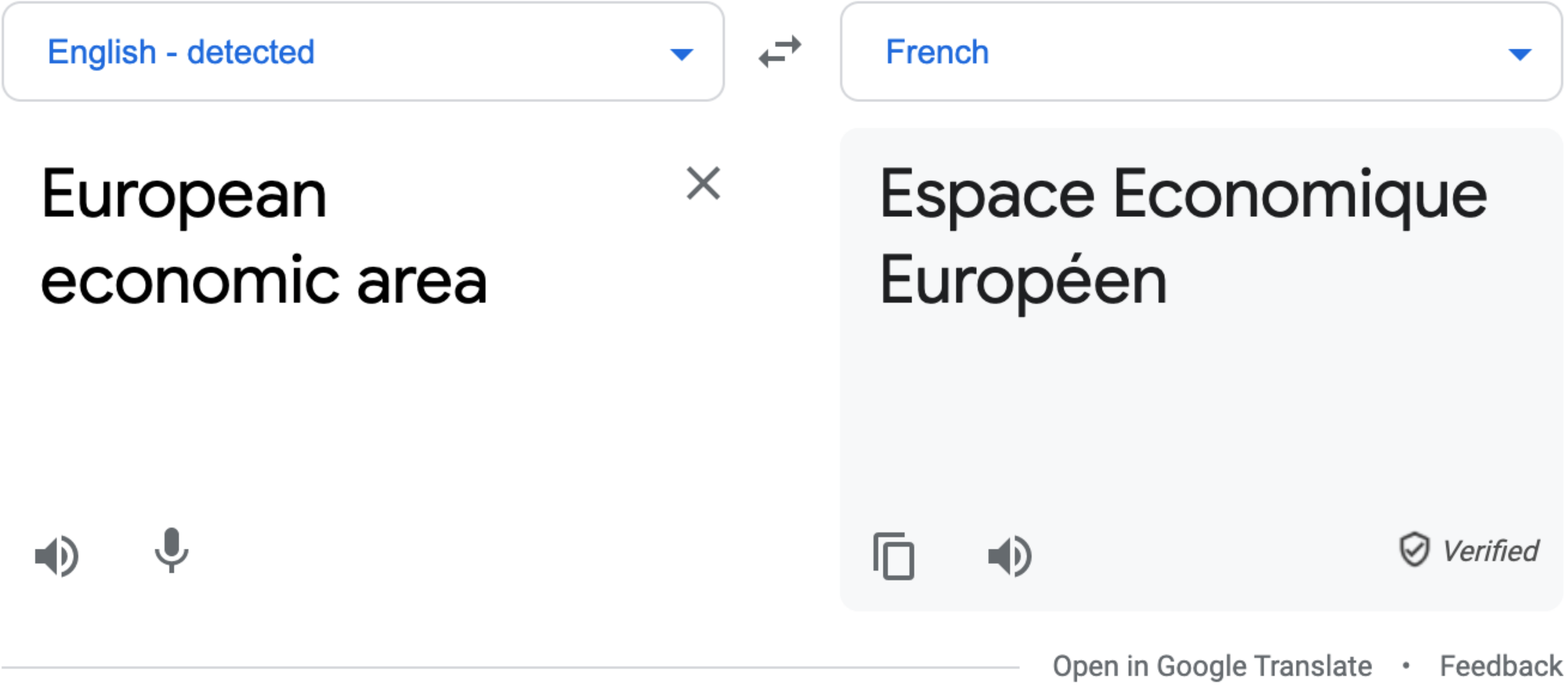
$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Credit for https://github.com/wangshusen/DeepLearning/blob/master/Slides/9_RNN_5.pdf

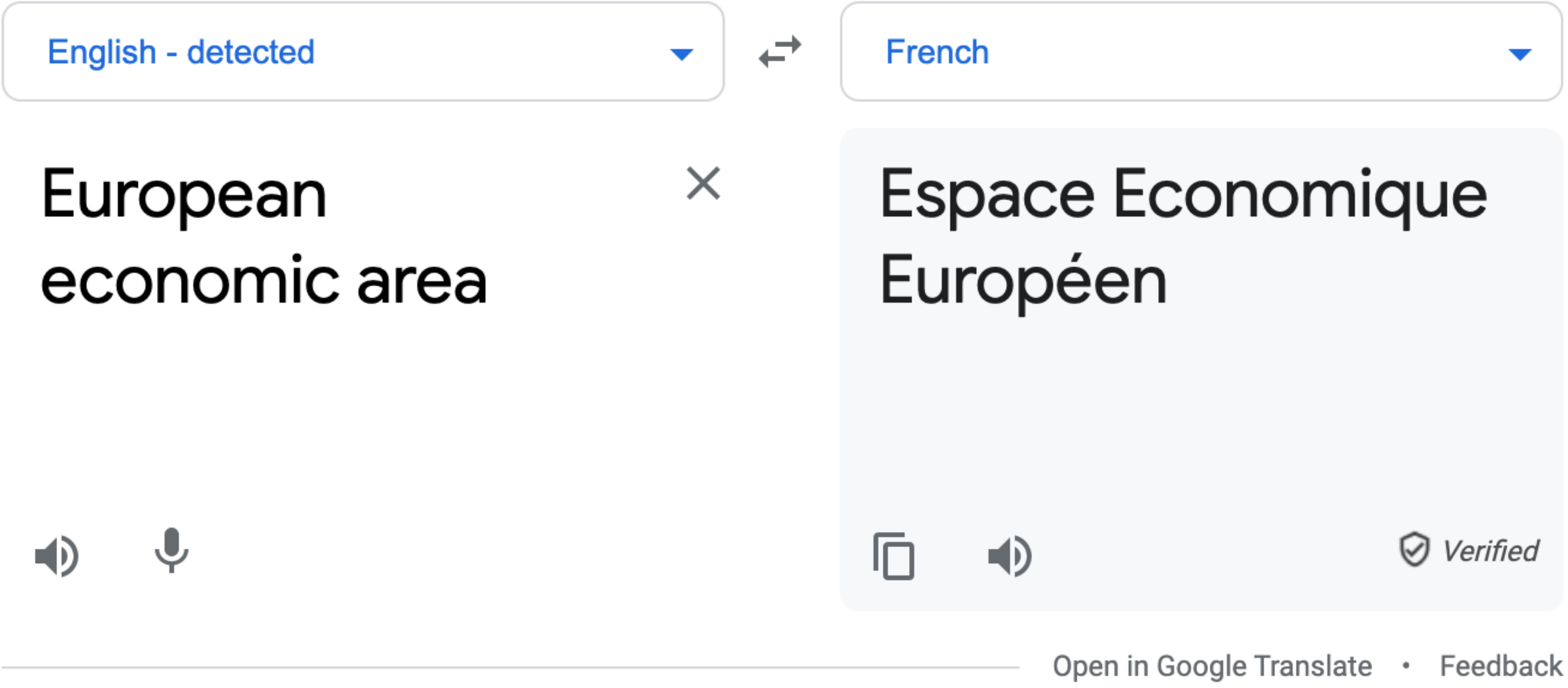
In practice

Machine translation



In practice

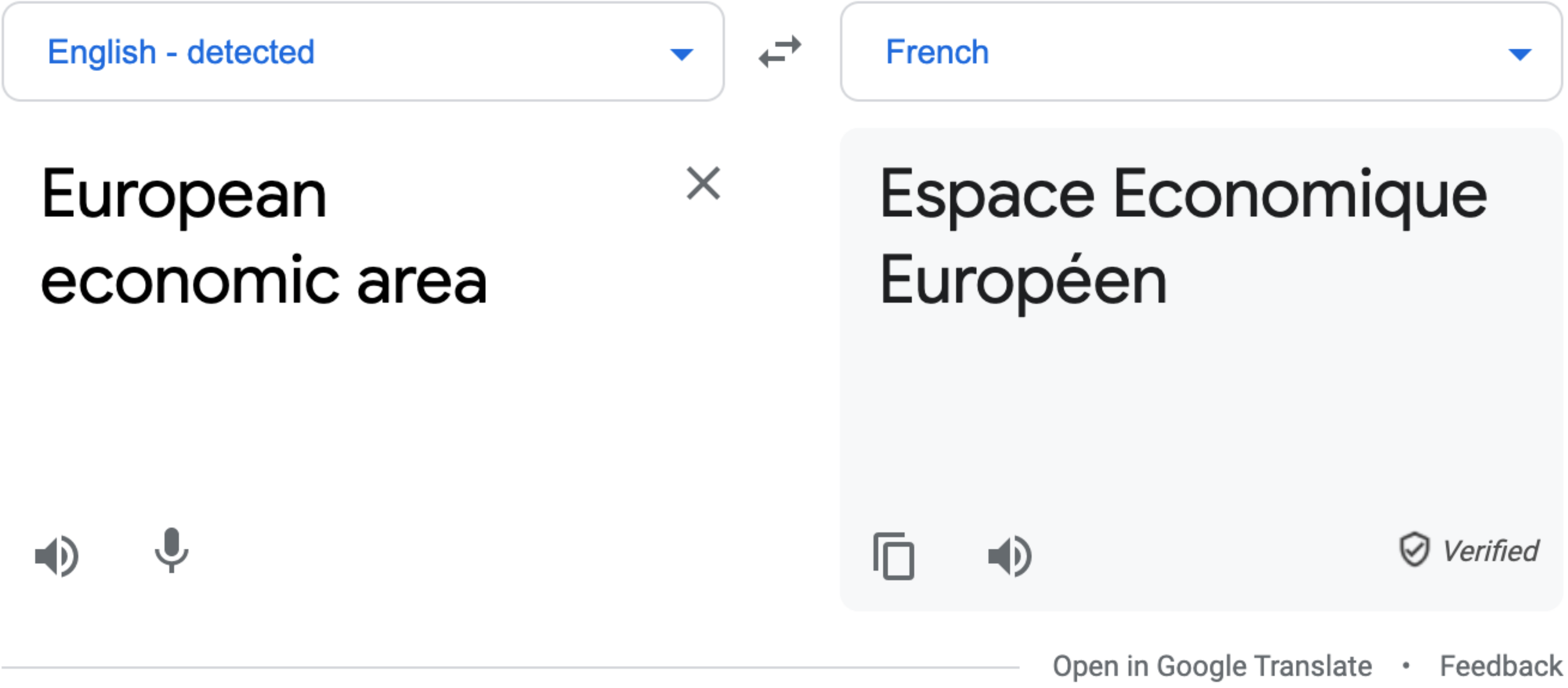
Machine translation



Q: can vanilla RNN handle machine translation?

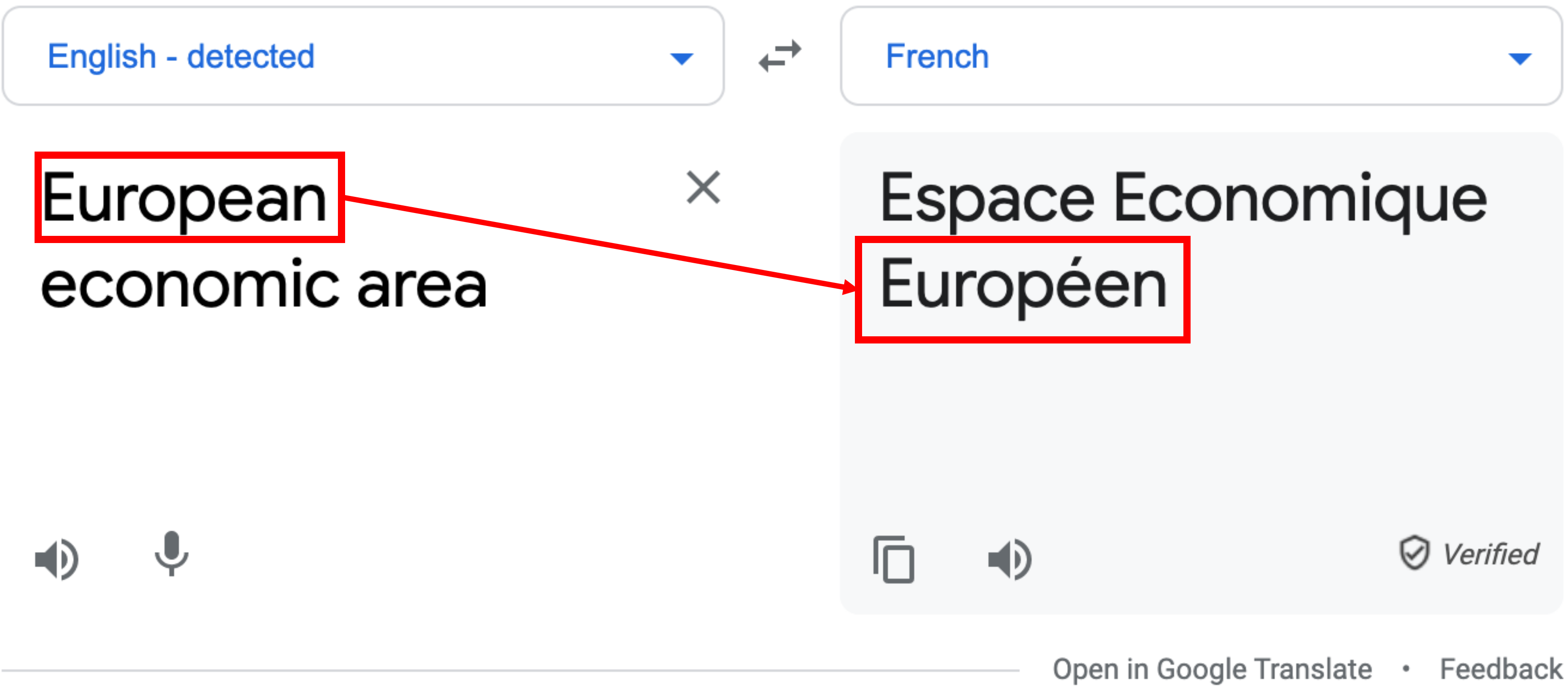
In practice

Machine translation

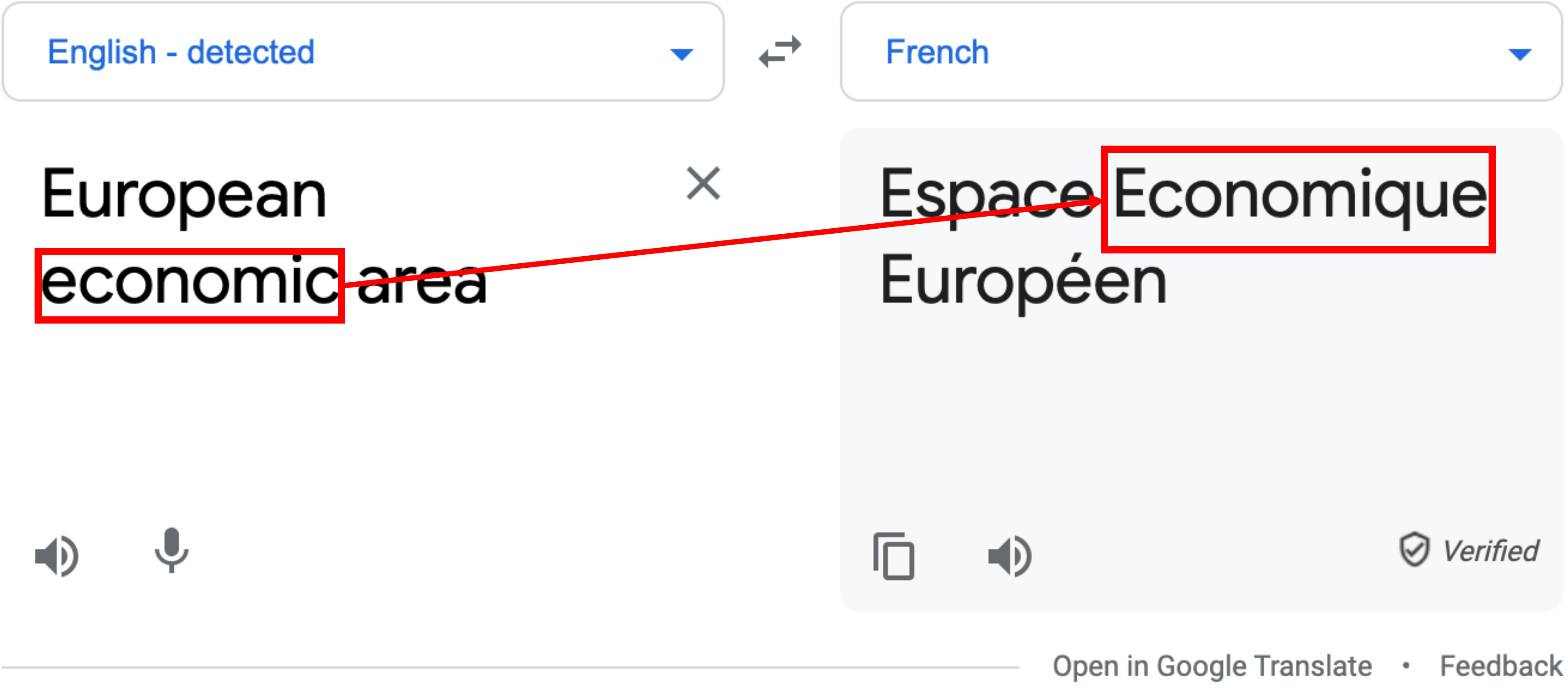


Q: can we find some correlation?

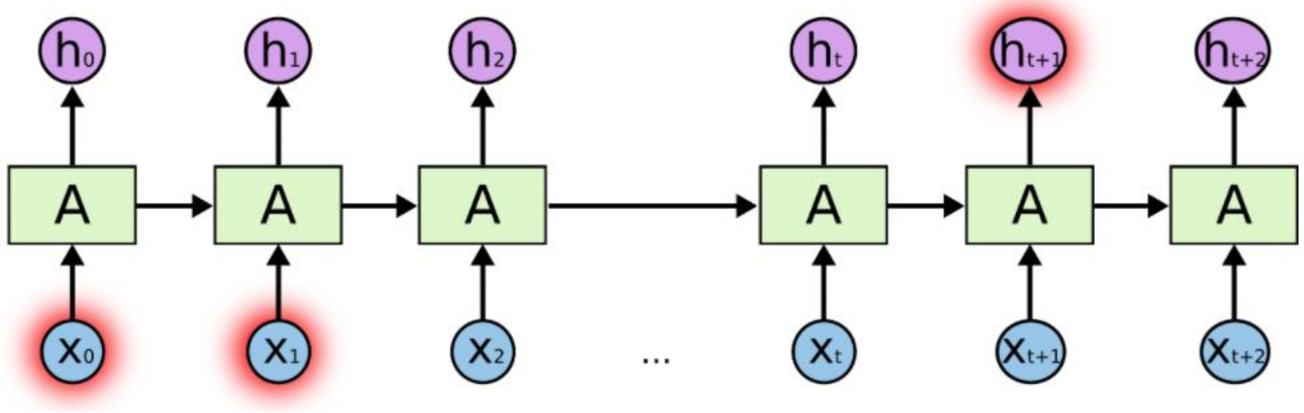
In practice


English - detected
French
European
economic area
Espace Economique
Européen
Verified
Open in Google Translate • Feedback

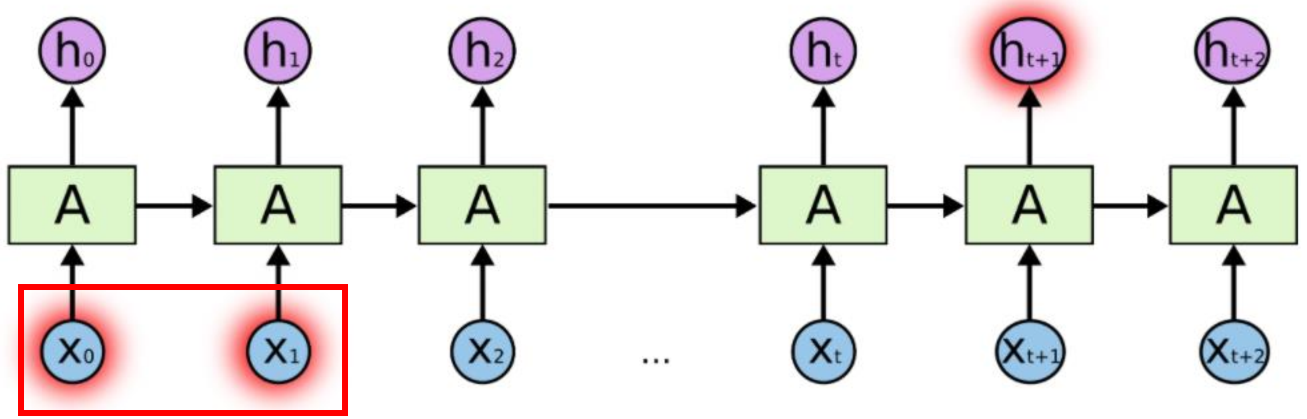
In practice


English - detected
French
European economic area
Espace Economique Européen
Verified
Open in Google Translate • Feedback

Vanilla RNN information flow

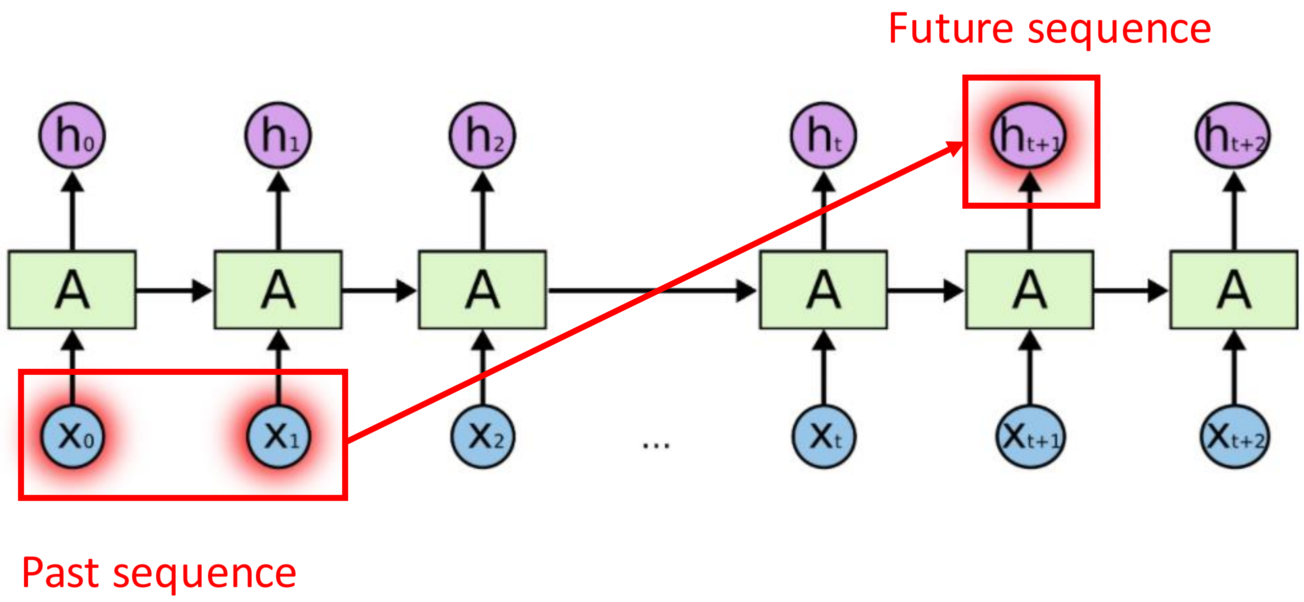


Vanilla RNN information flow

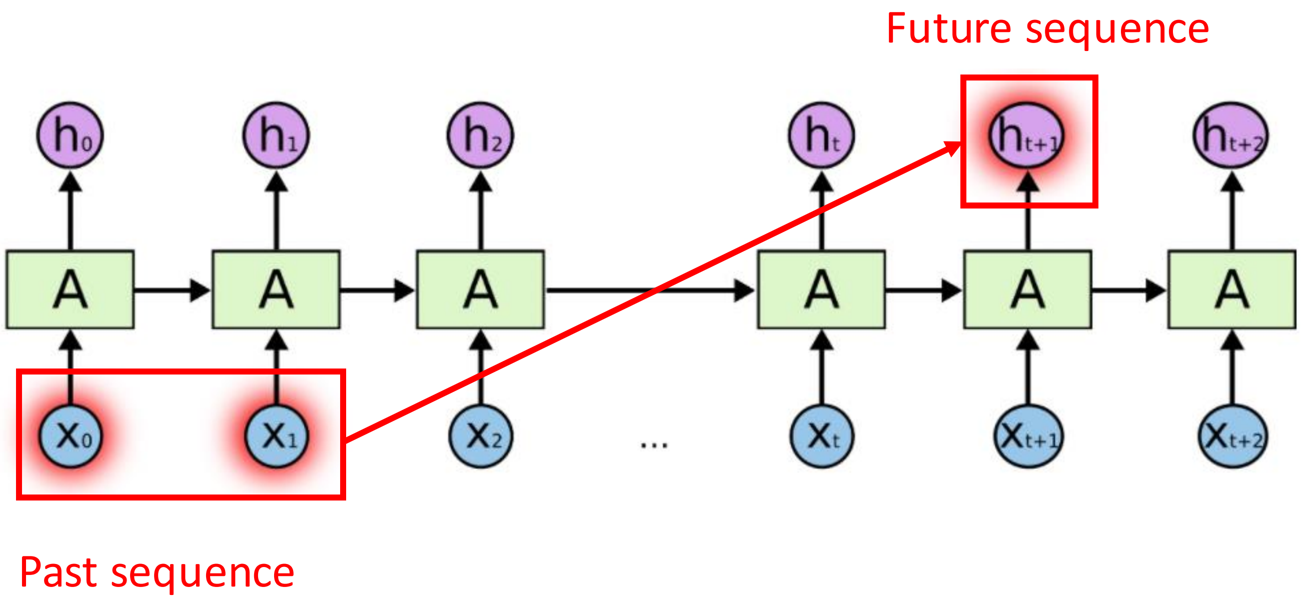


Past sequence

Vanilla RNN information flow



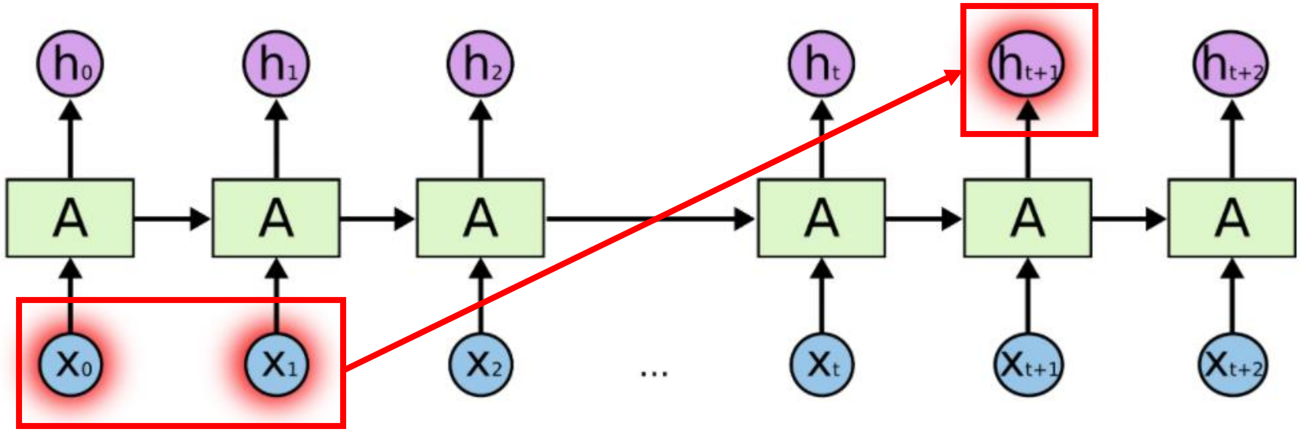
Vanilla RNN information flow



the clouds are in the sky

Vanilla RNN information flow

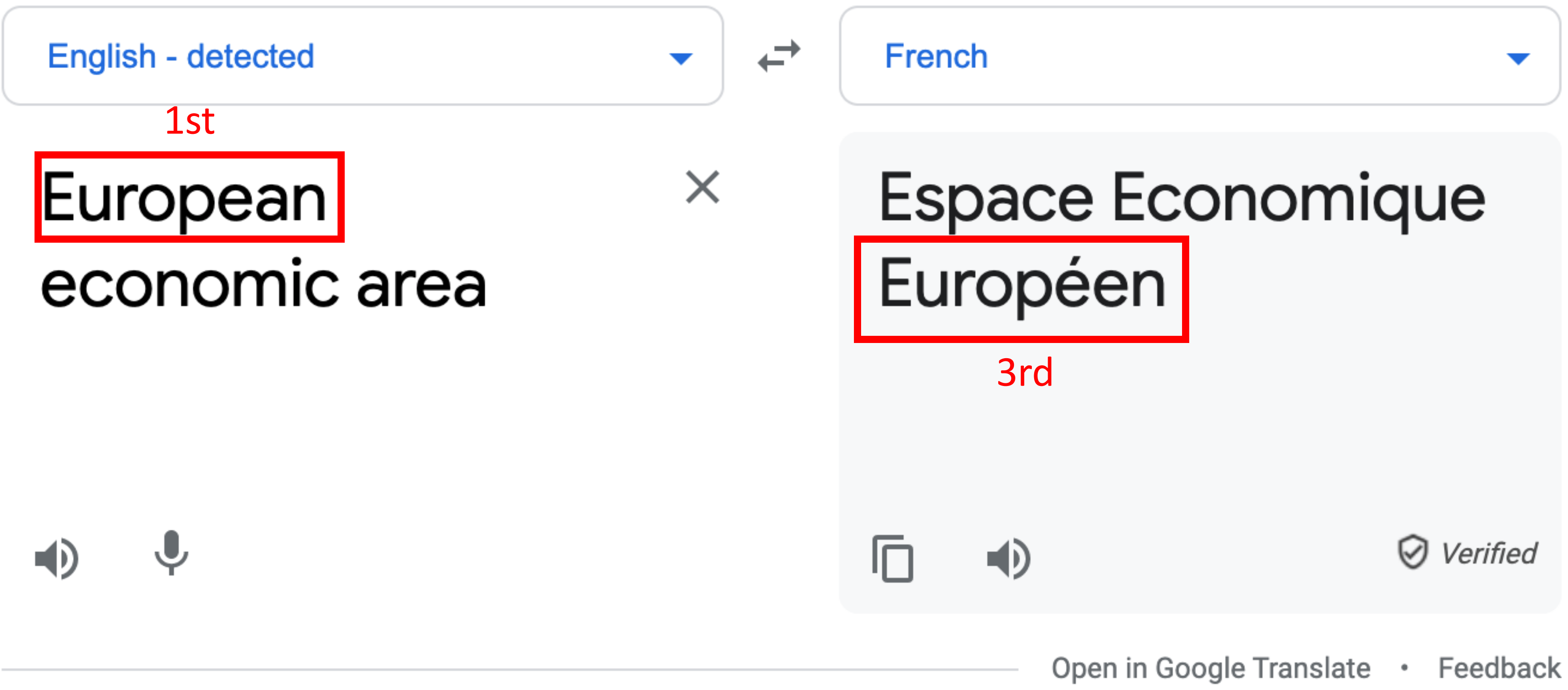
Future sequence



Past sequence

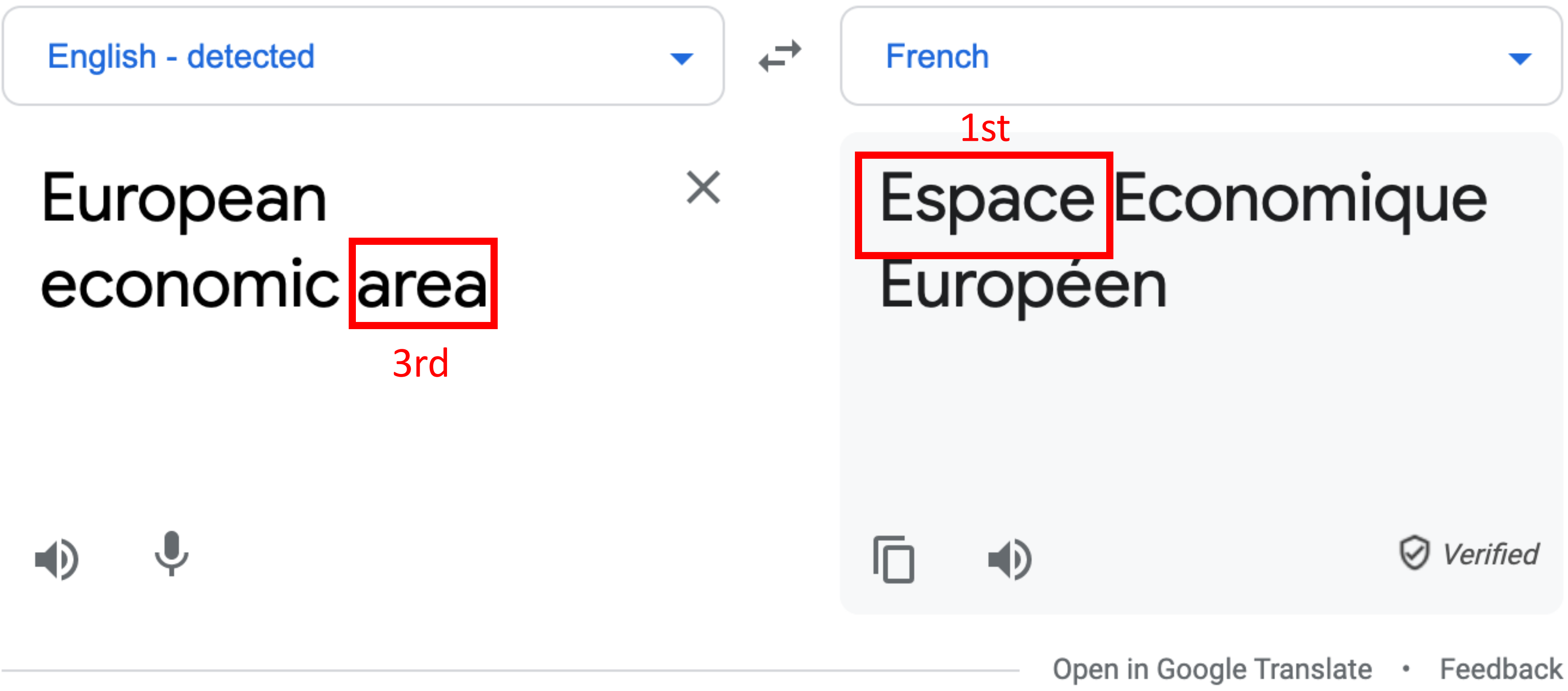
I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in France. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent *French*.

In practice



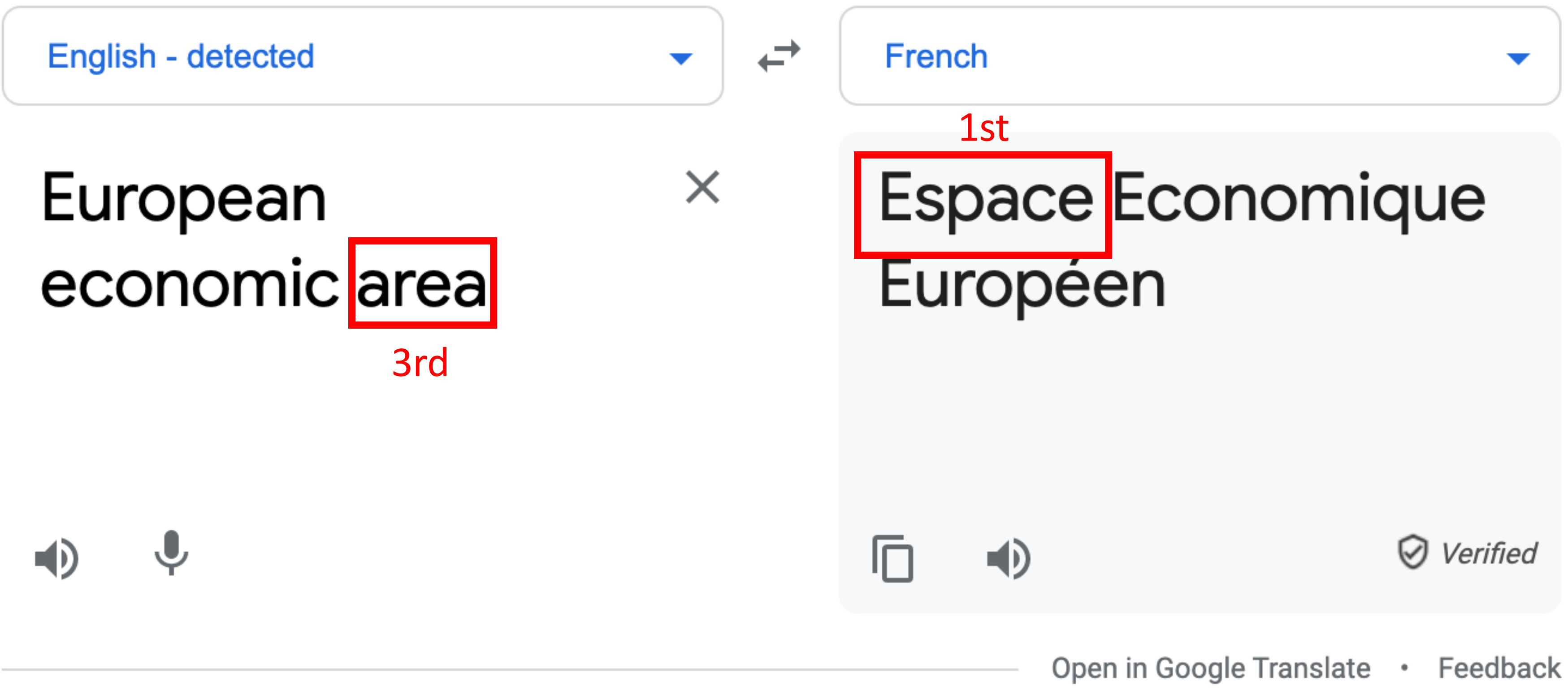
Q: is vanilla RNN able to use information flow to generate Européen?

In practice



Q: is vanilla RNN able to use information flow to generate Espace?

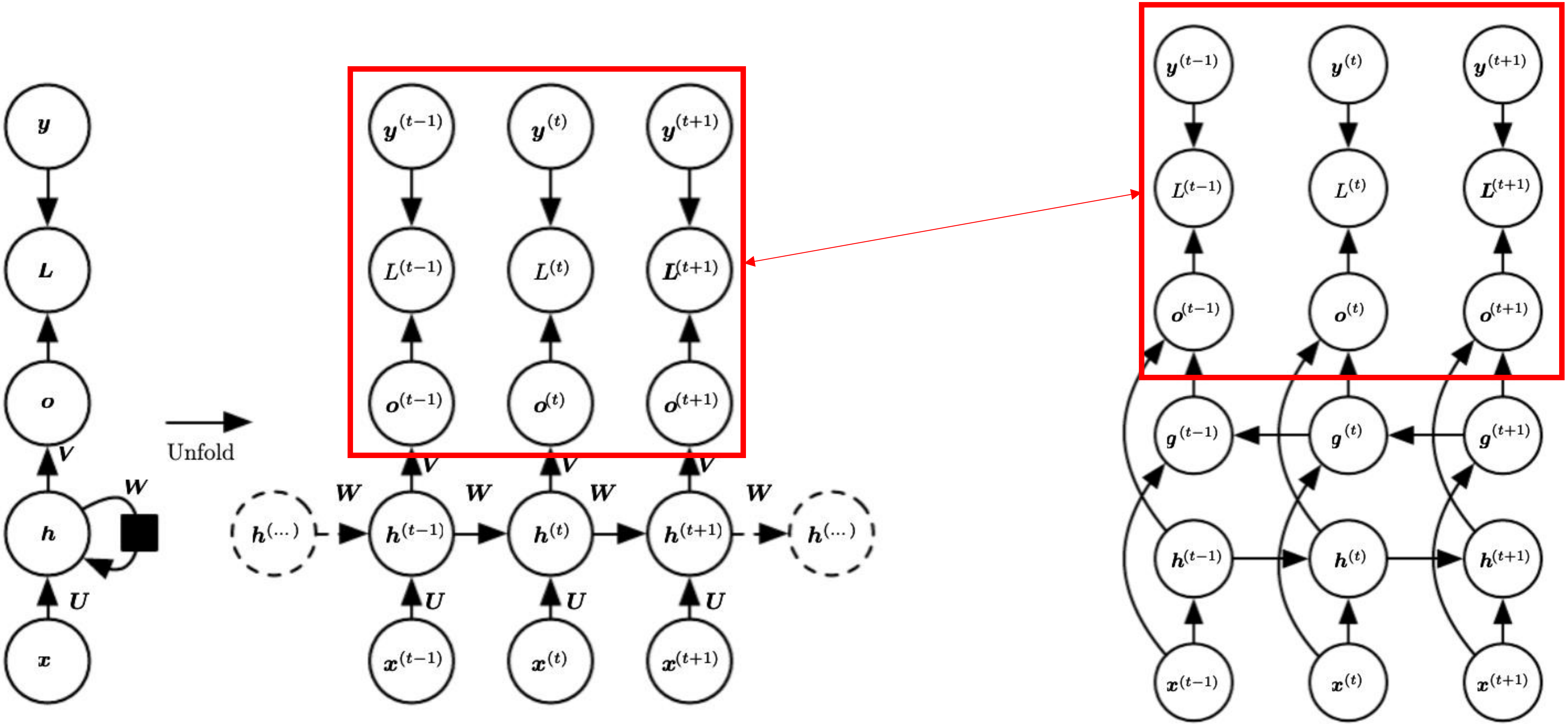
In practice



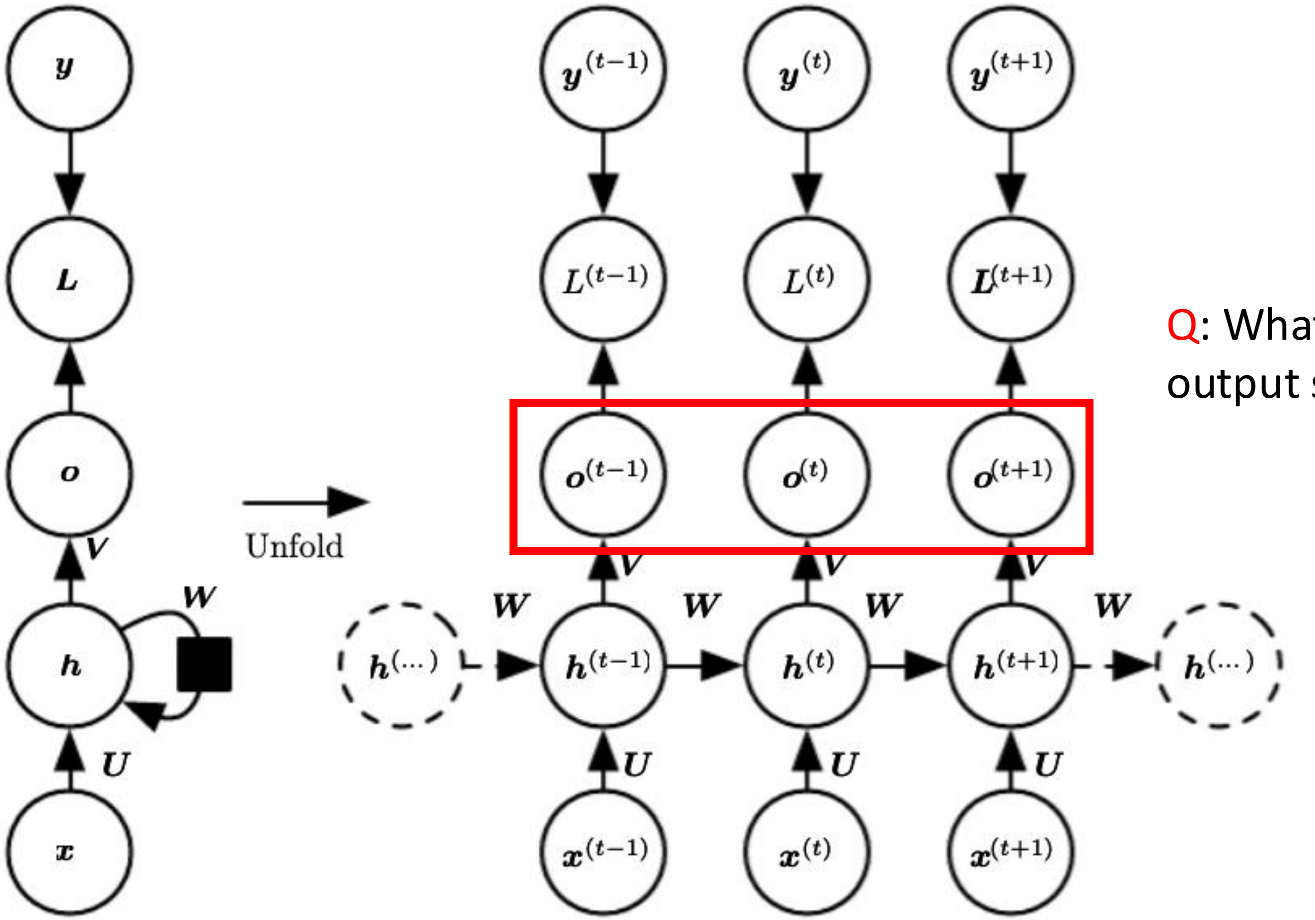
Q: is vanilla RNN able to use information flow to generate Espace?

Q: what if we need some future sequence to determine the output sequence?

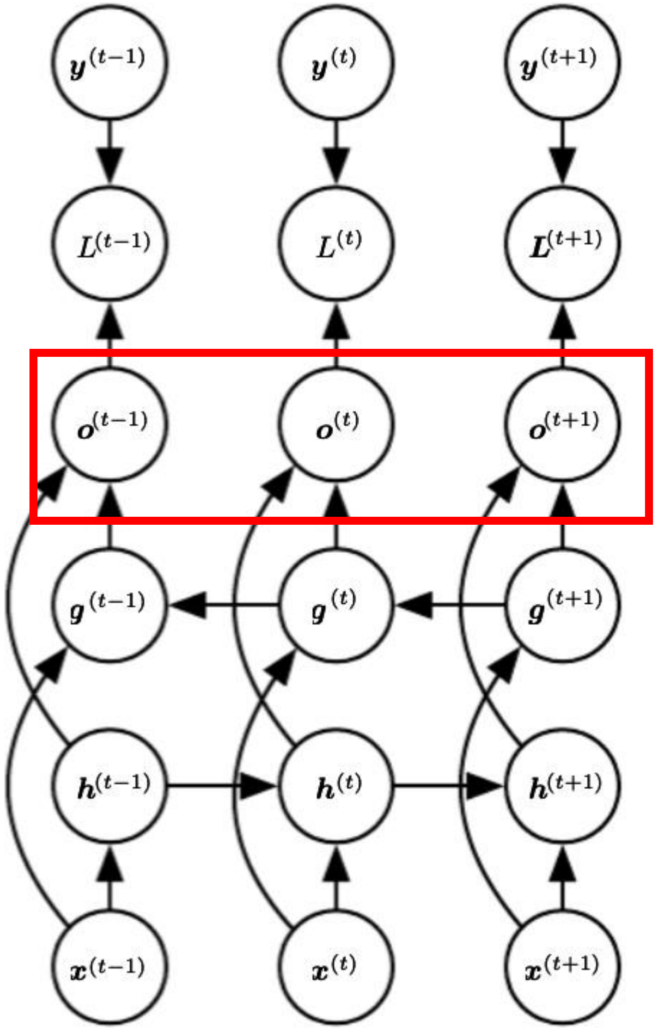
Bi-directional RNN



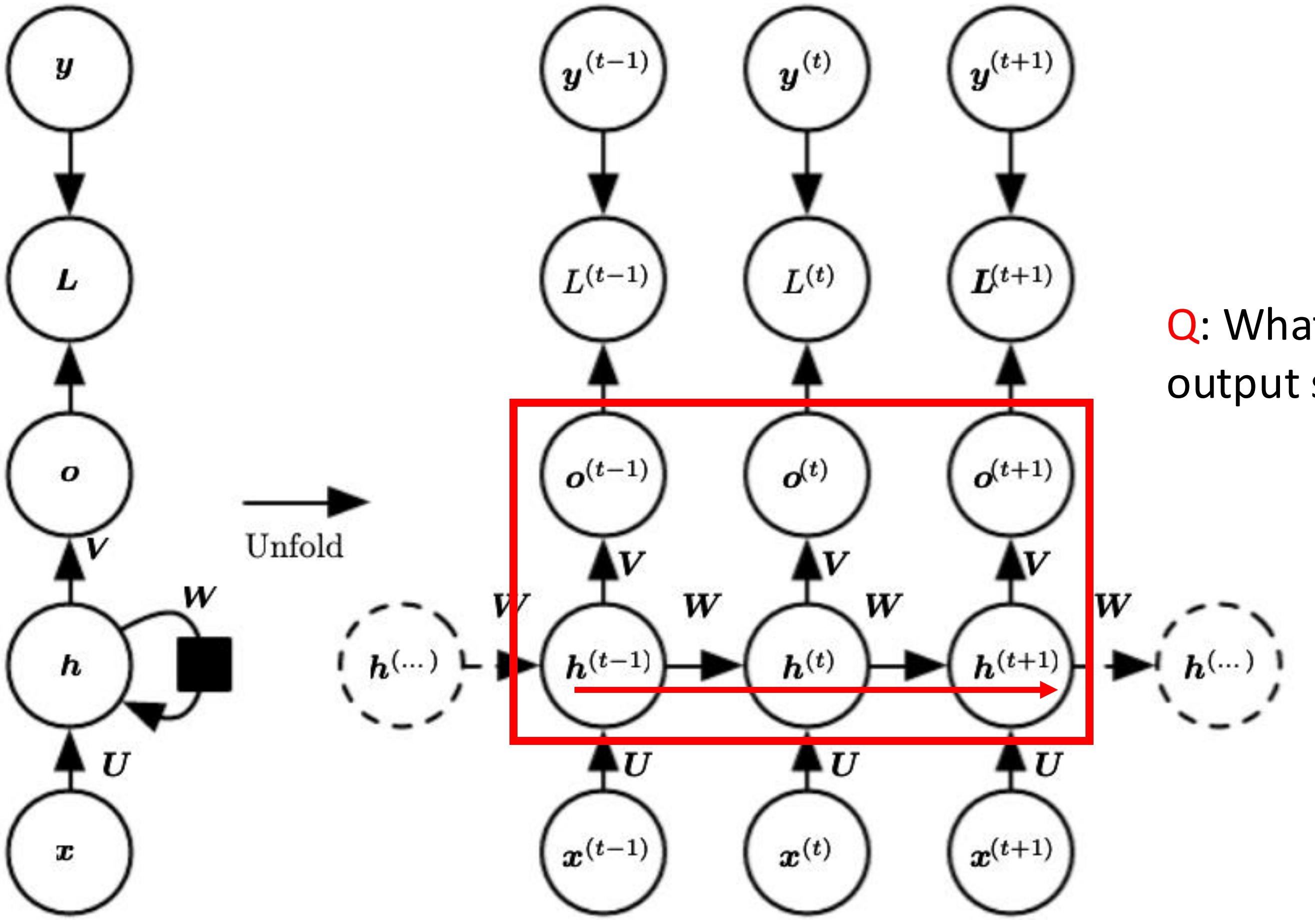
Bi-directional RNN



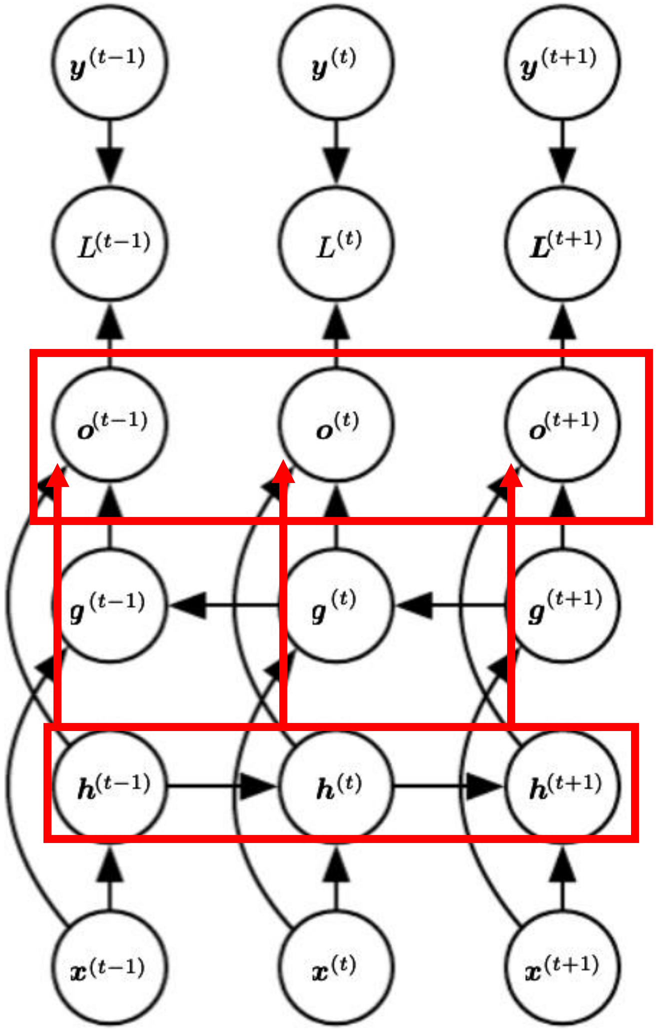
Q: What determine output sequence?



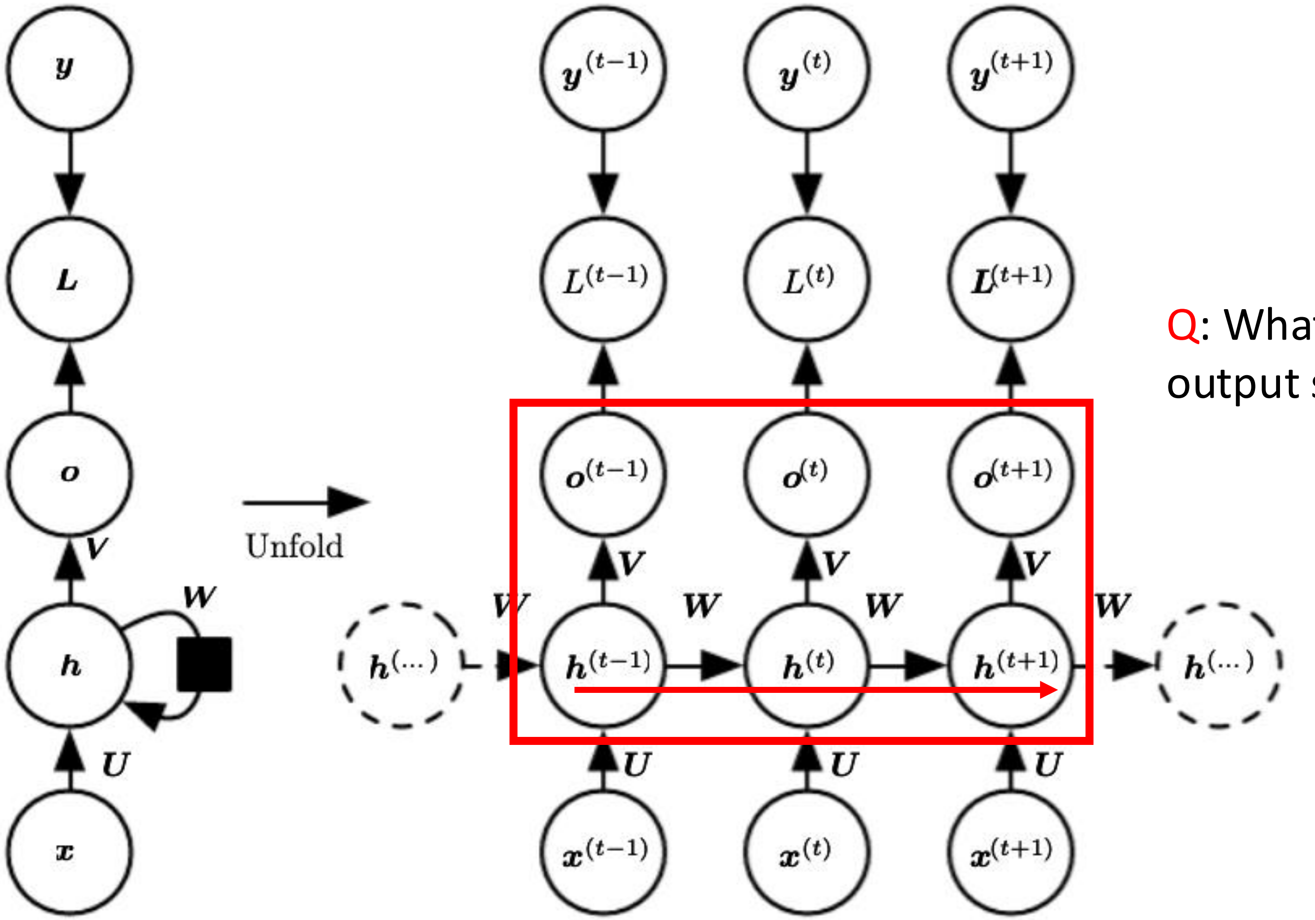
Bi-directional RNN



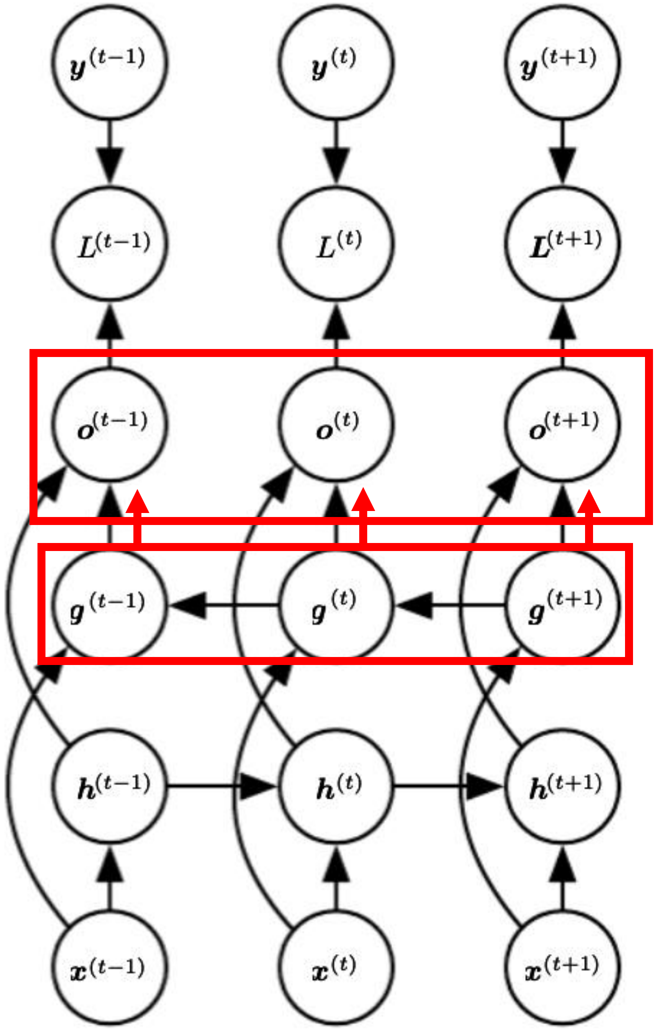
Q: What determine output sequence?



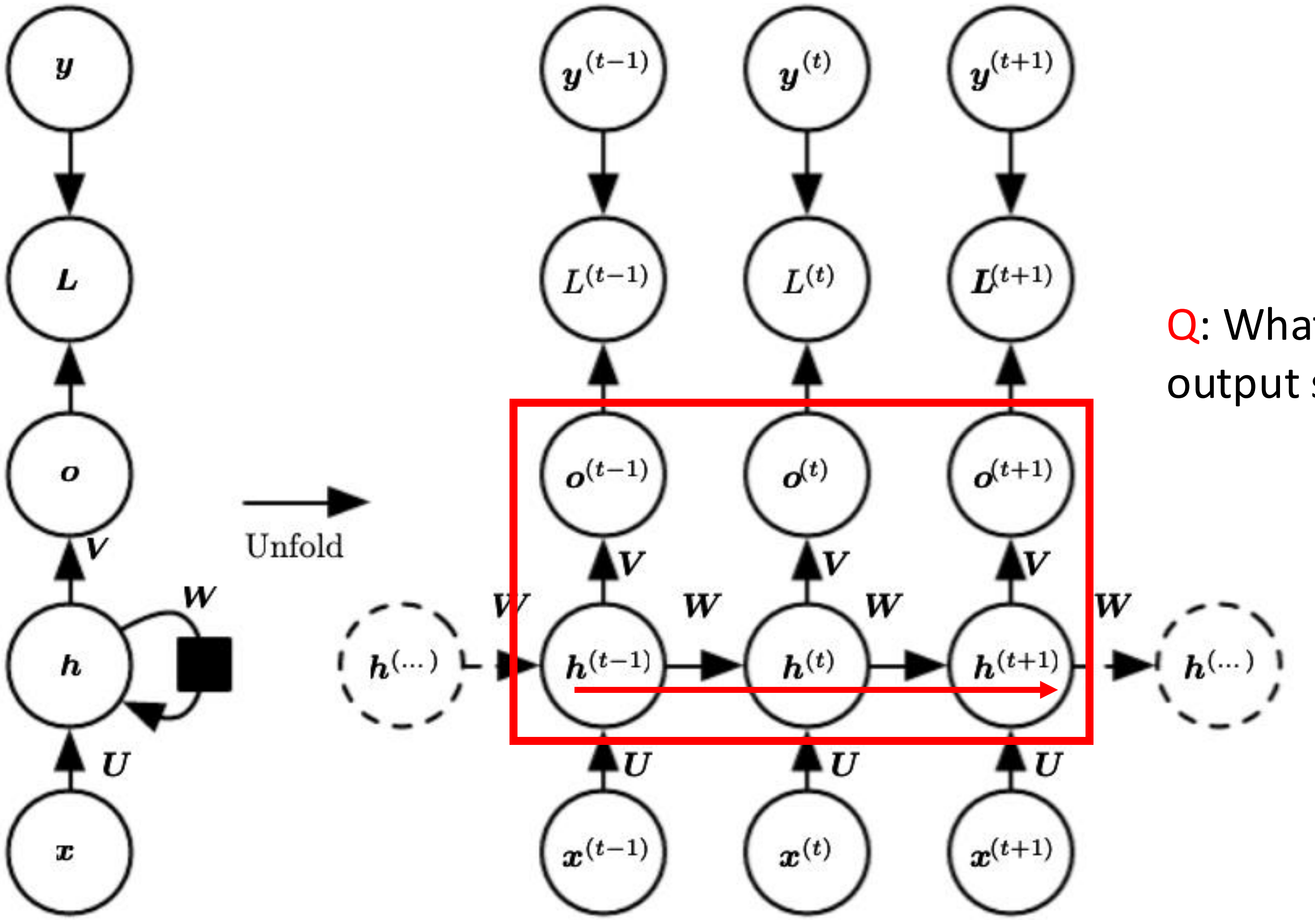
Bi-directional RNN



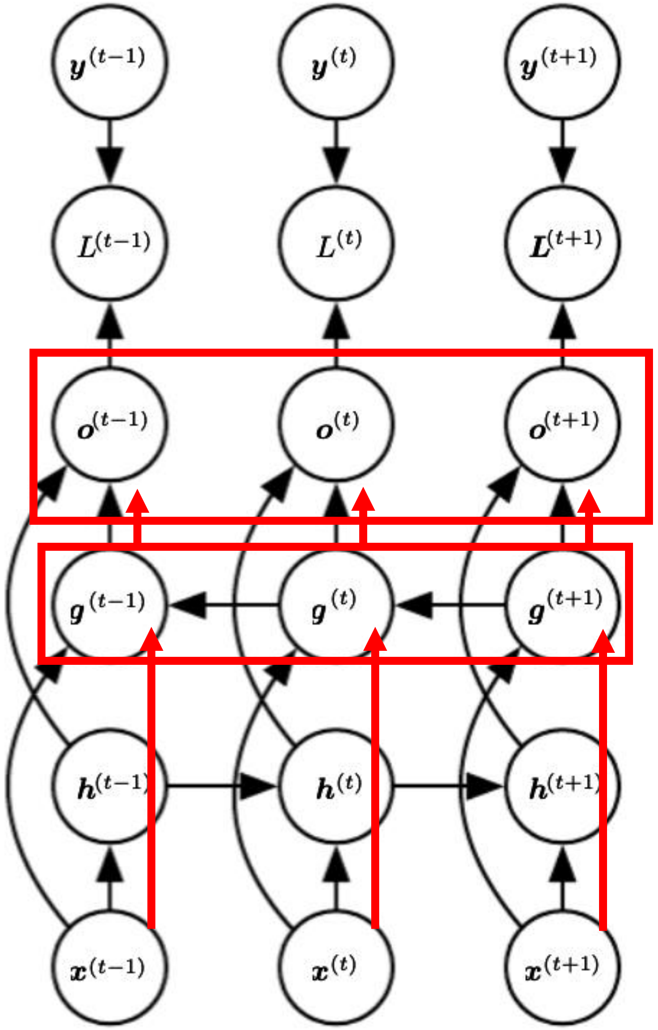
Q: What determine output sequence?



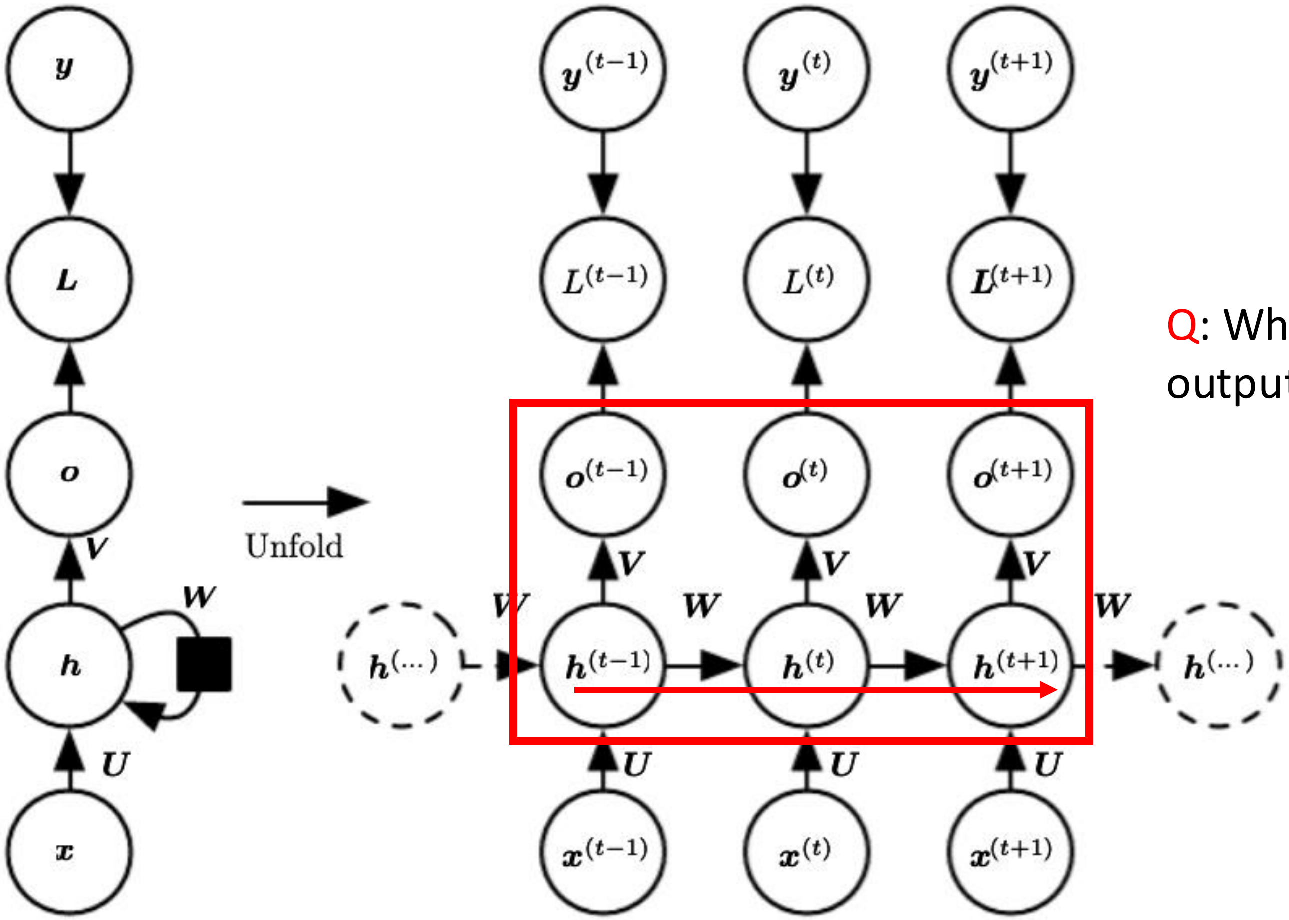
Bi-directional RNN



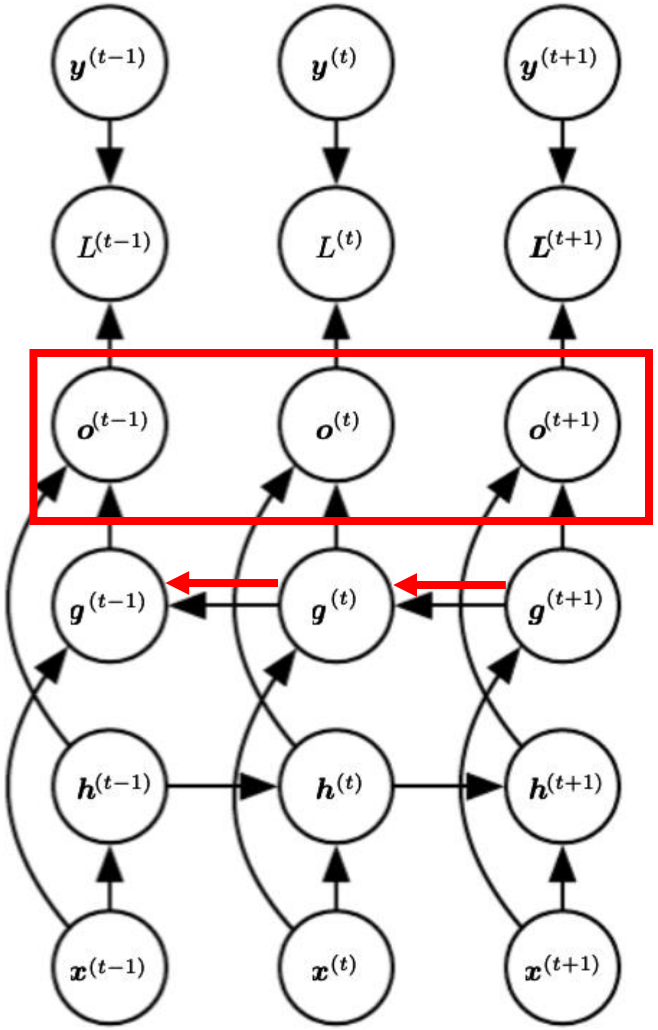
Q: What determine output sequence?



Bi-directional RNN



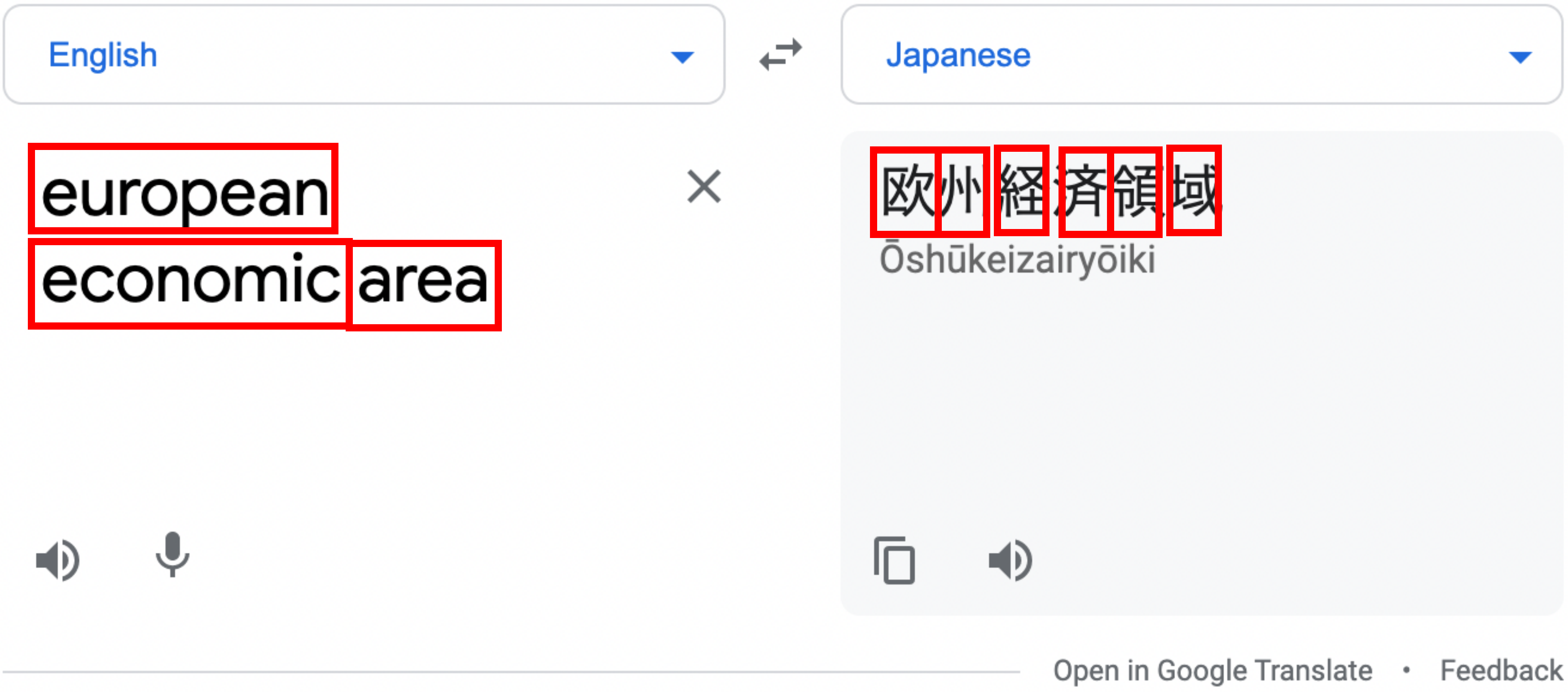
Q: What determine output sequence?



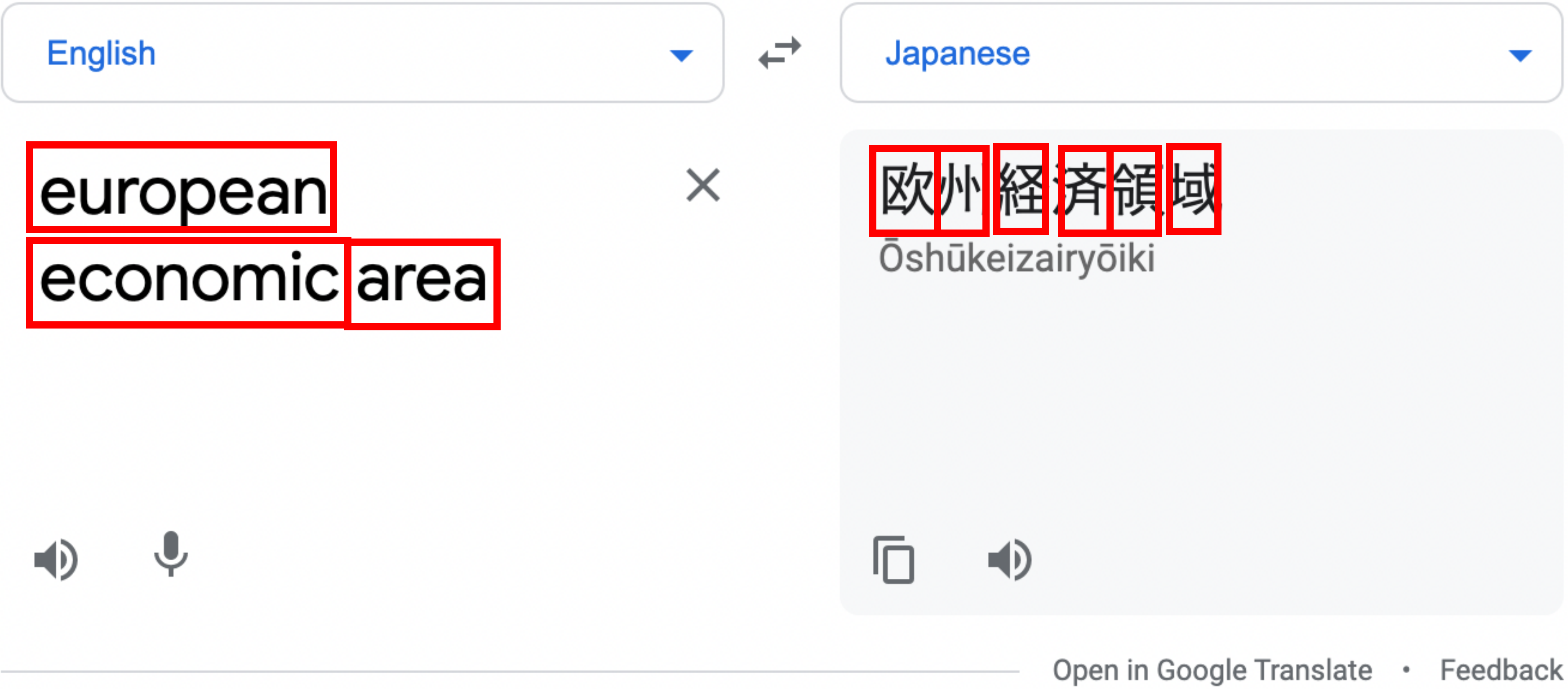
Seq2seq architecture

Seq2seq architecture

Seq2seq architecture

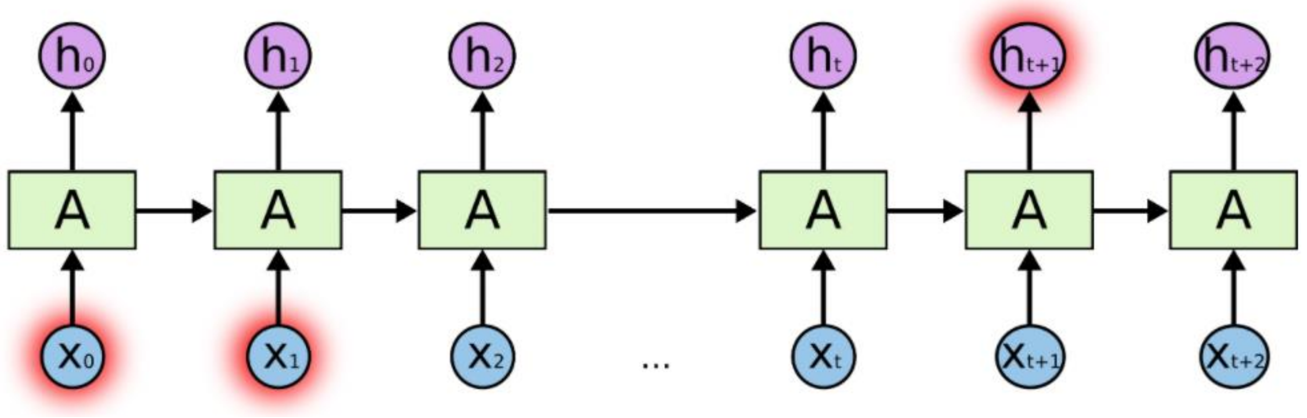


Seq2seq architecture

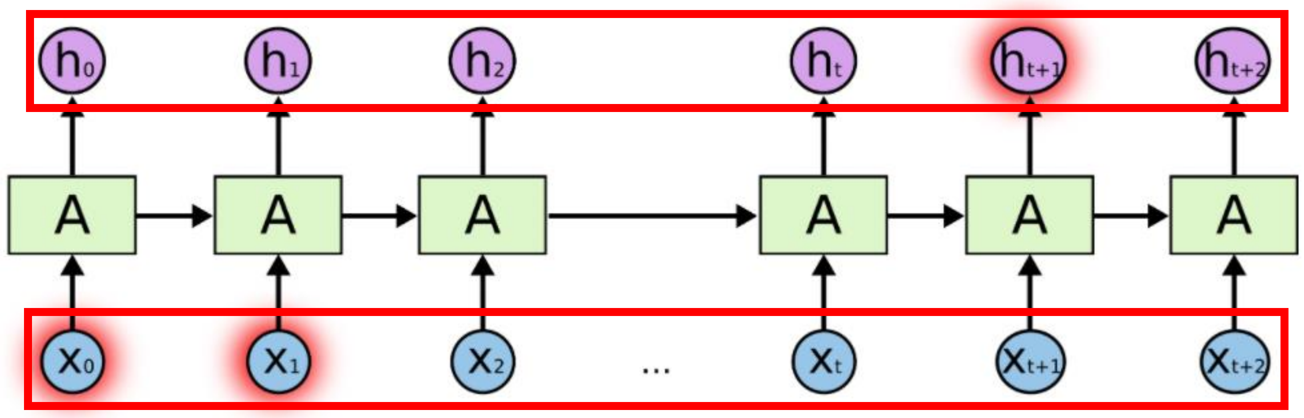


Q: is vanilla RNN able to generate an output with different length of input?

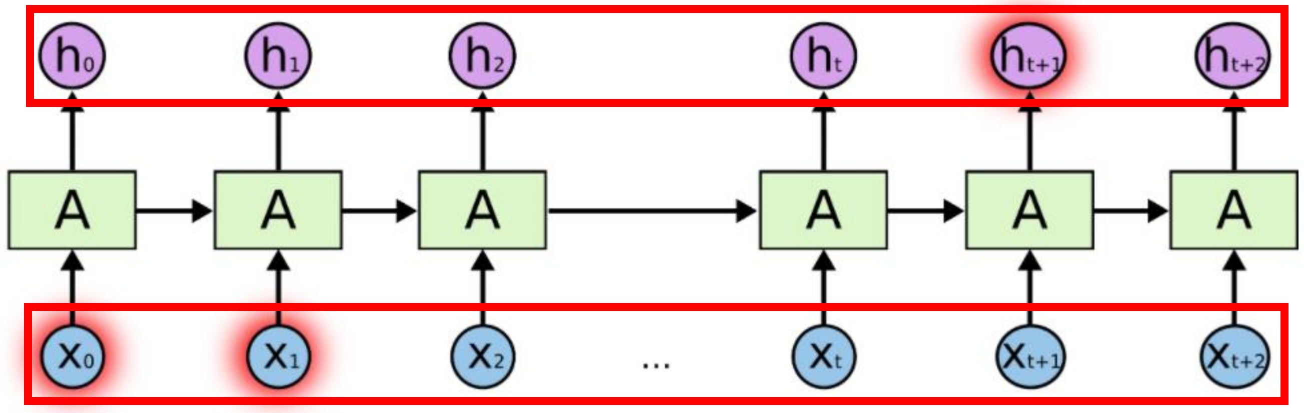
Vanilla RNN information flow



Vanilla RNN information flow

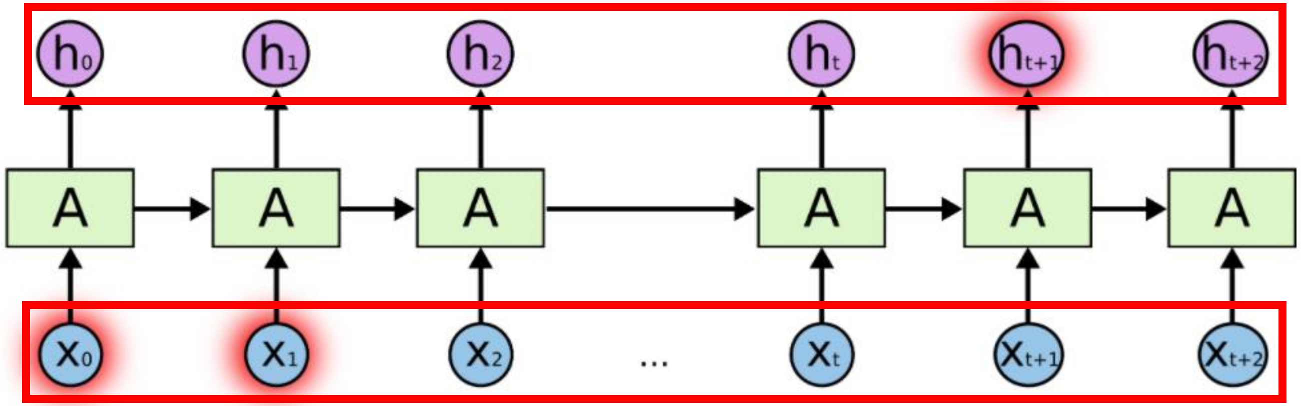


Vanilla RNN information flow



Input length = ??

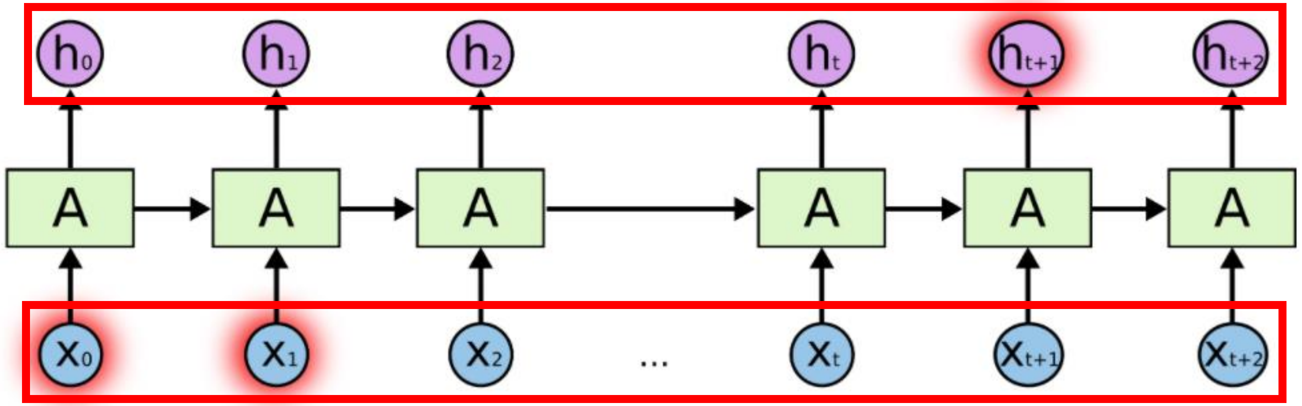
Vanilla RNN information flow



Input length = t+3

Vanilla RNN information flow

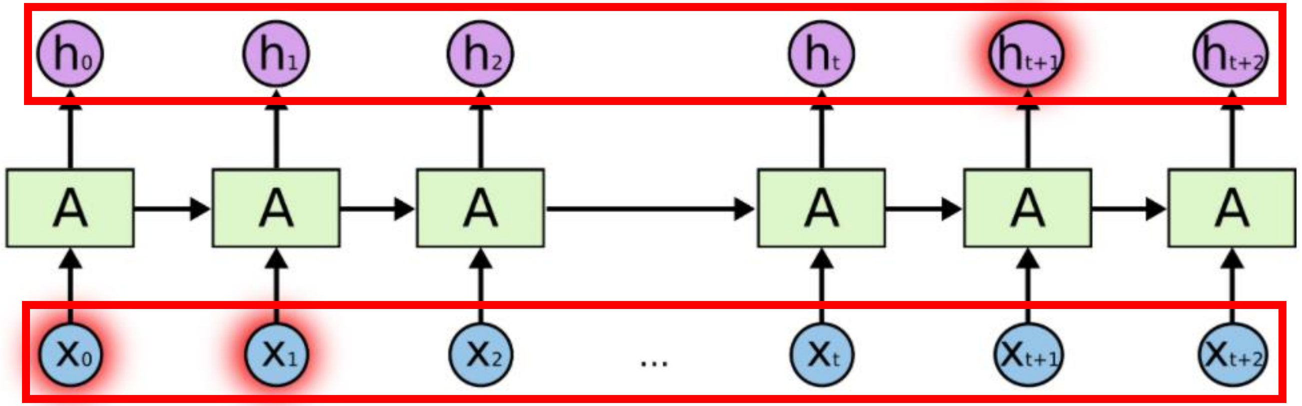
Output length = ??



Input length = t+3

Vanilla RNN information flow

Output length = t+3



Input length = t+3

Q: what if the input and output sequences are of different length?

Seq2seq RNN architecture

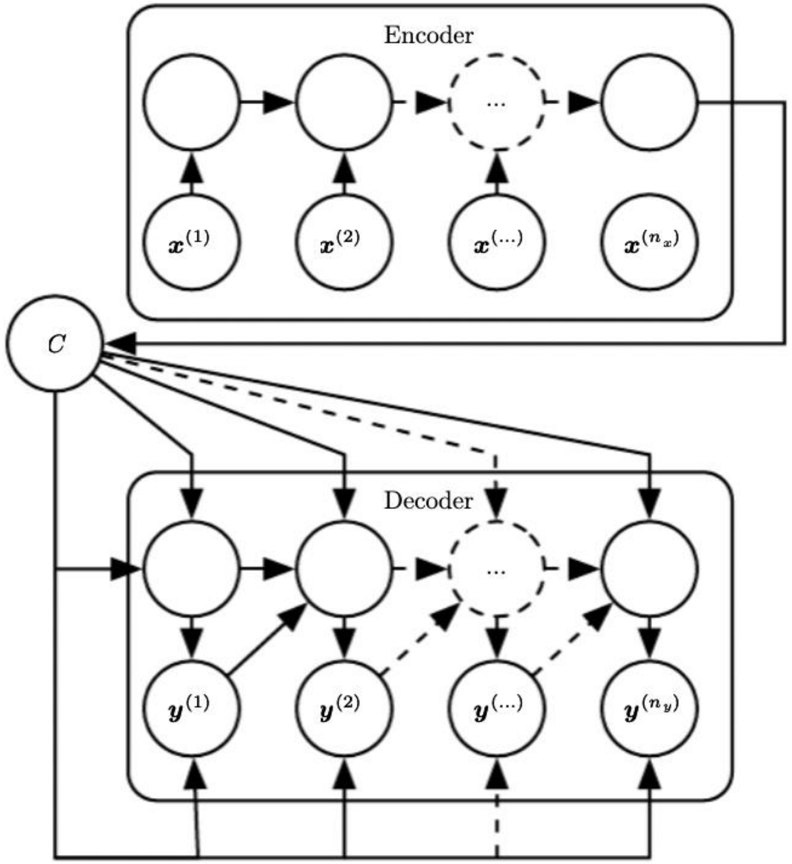
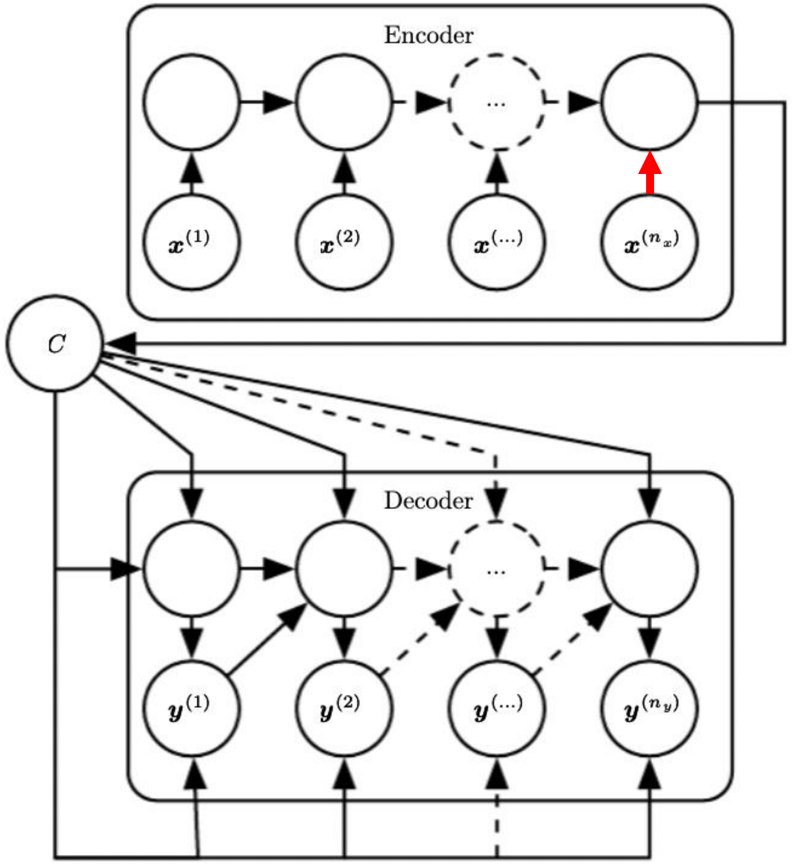


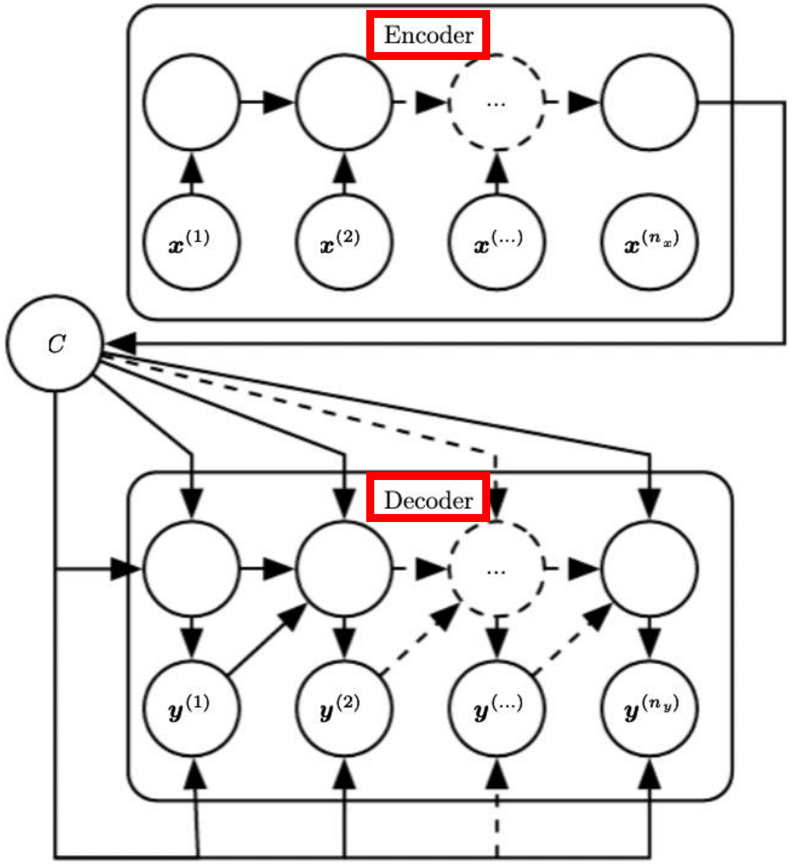
Figure 10.12 in deep learning book

Seq2seq RNN architecture

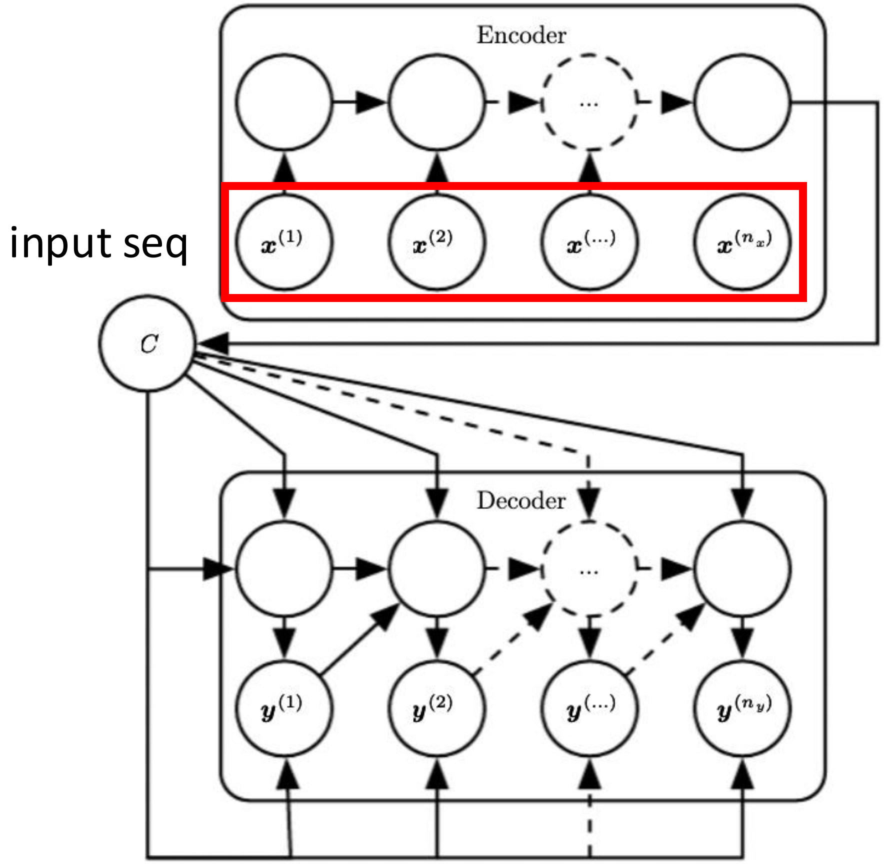


A typo in Figure 10.12 in deep learning book

Seq2seq RNN architecture

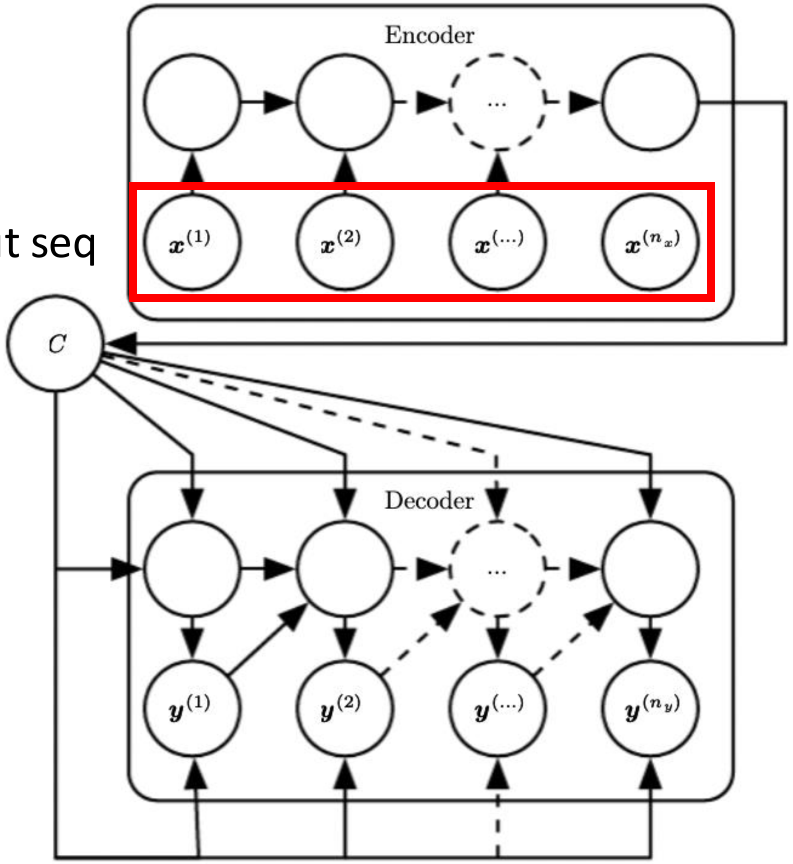


Seq2seq RNN architecture



Seq2seq RNN architecture

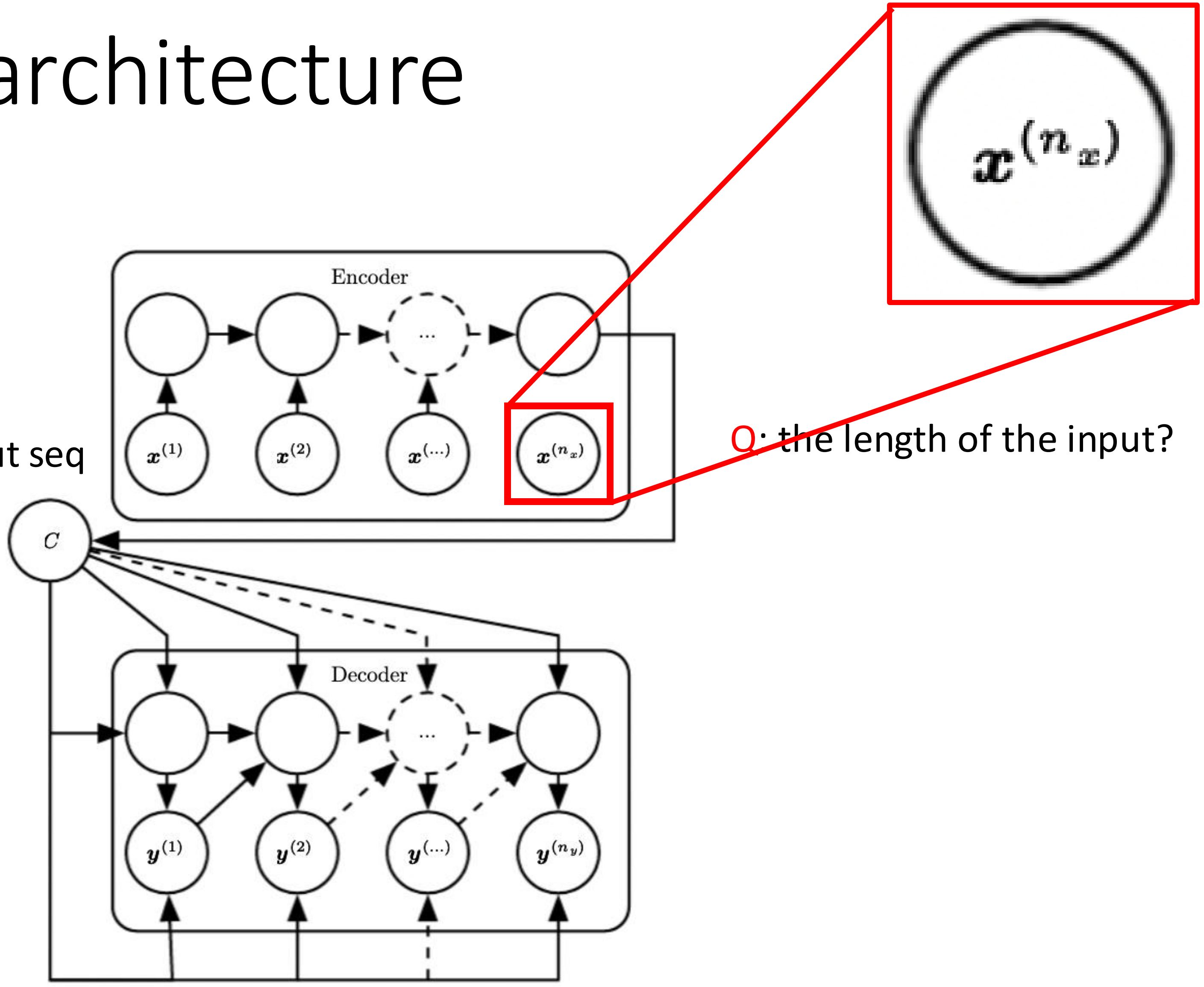
input seq



Q: the length of the input?

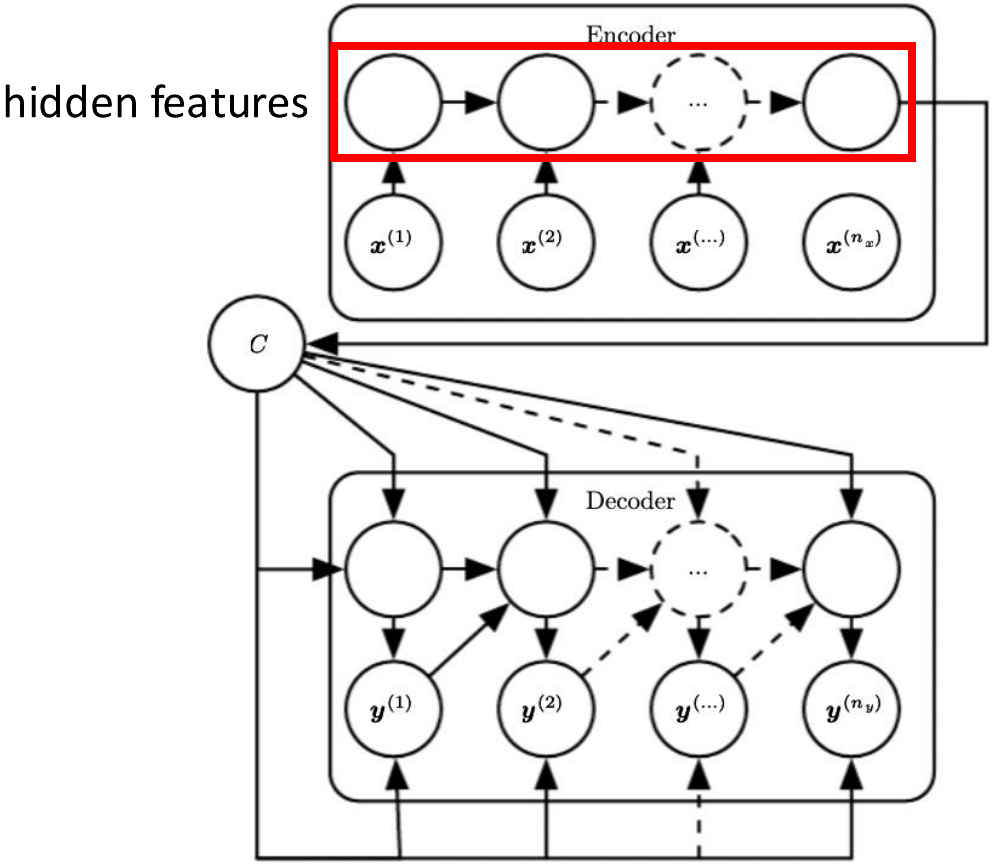
Seq2seq RNN architecture

input seq

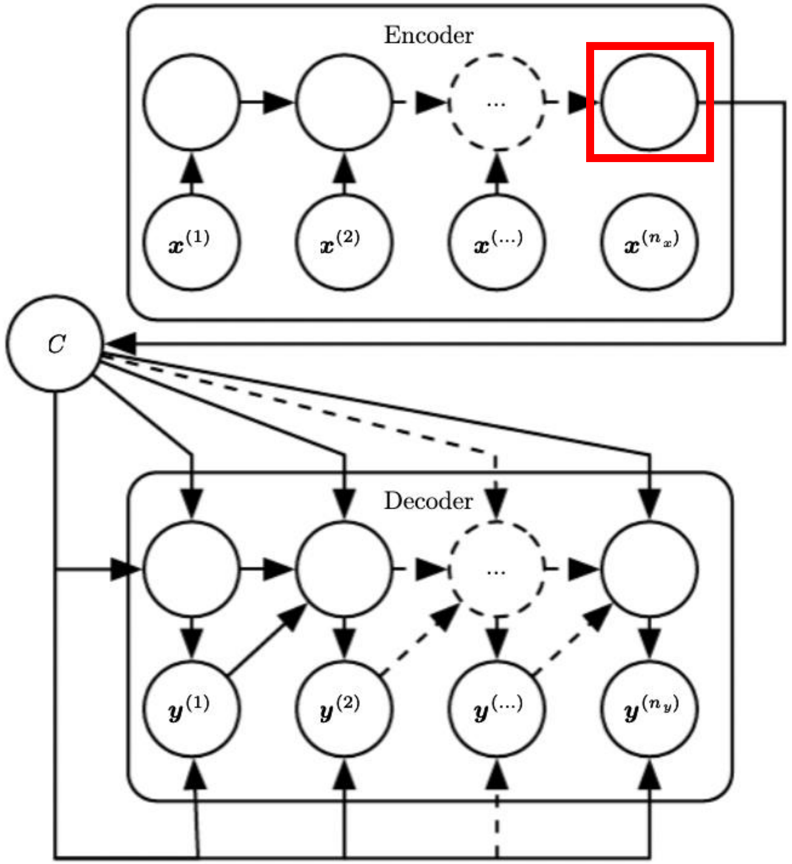


Q: the length of the input?

Seq2seq RNN architecture

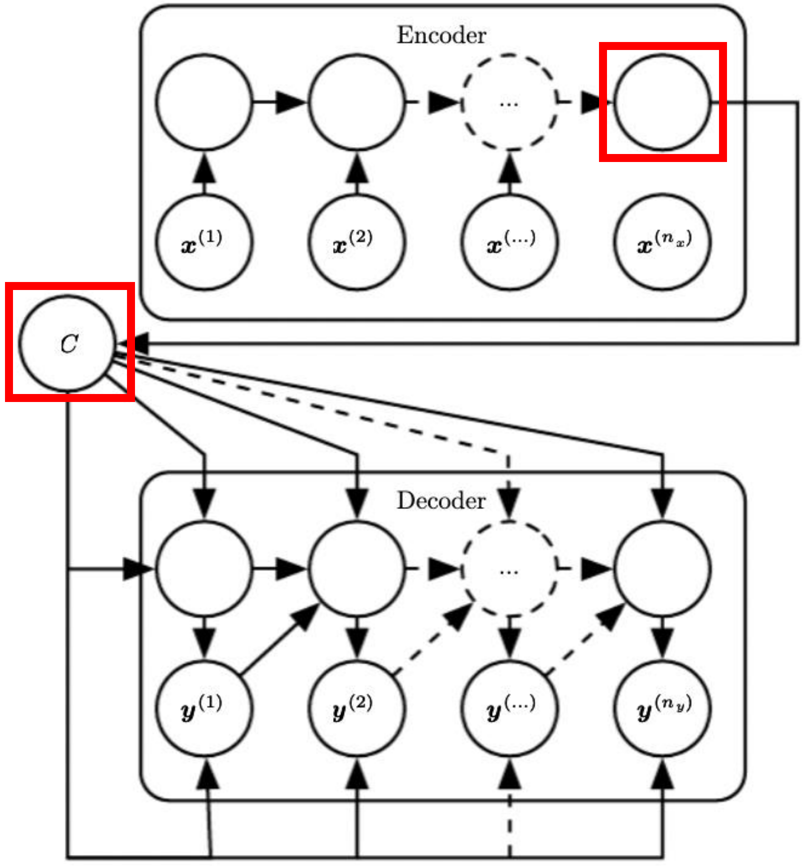


Seq2seq RNN architecture



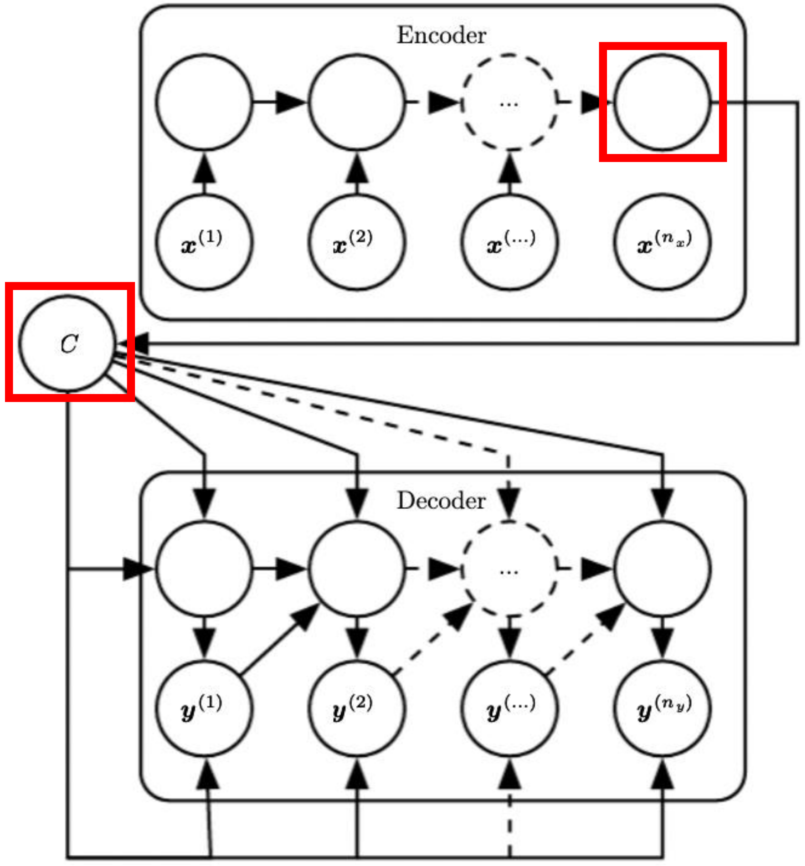
Seq2seq RNN architecture

context

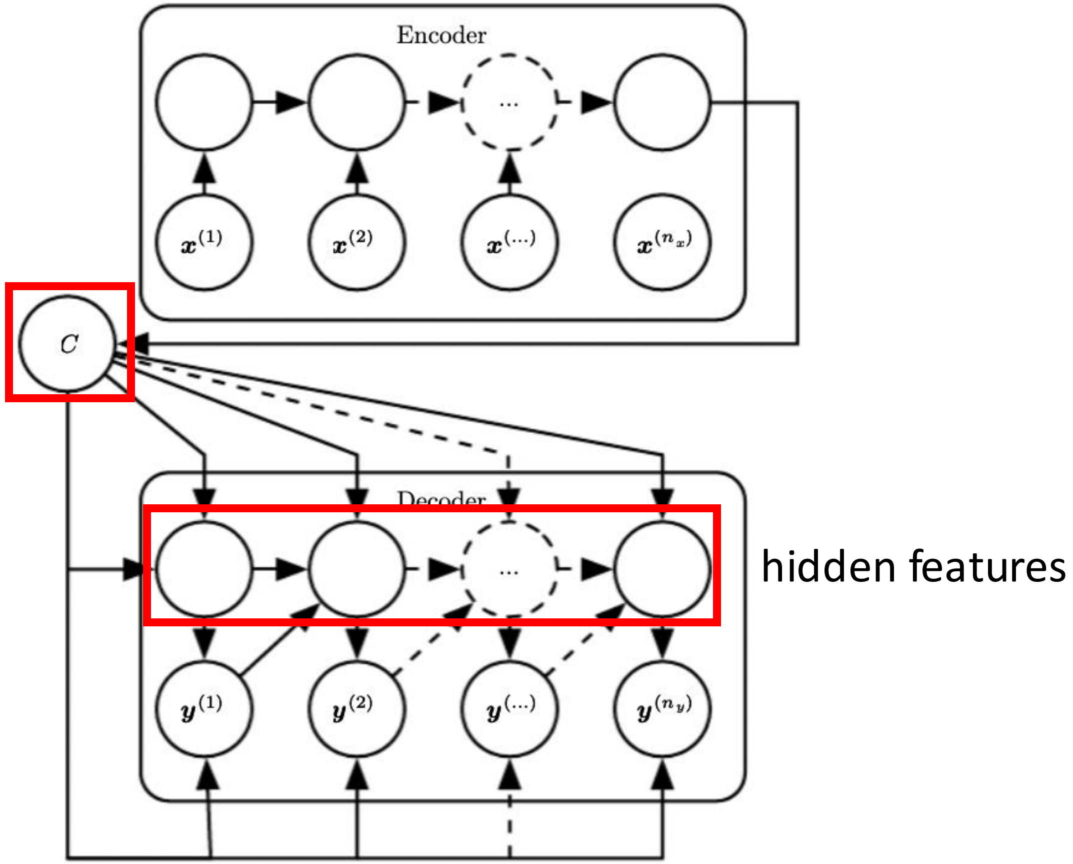


Seq2seq RNN architecture

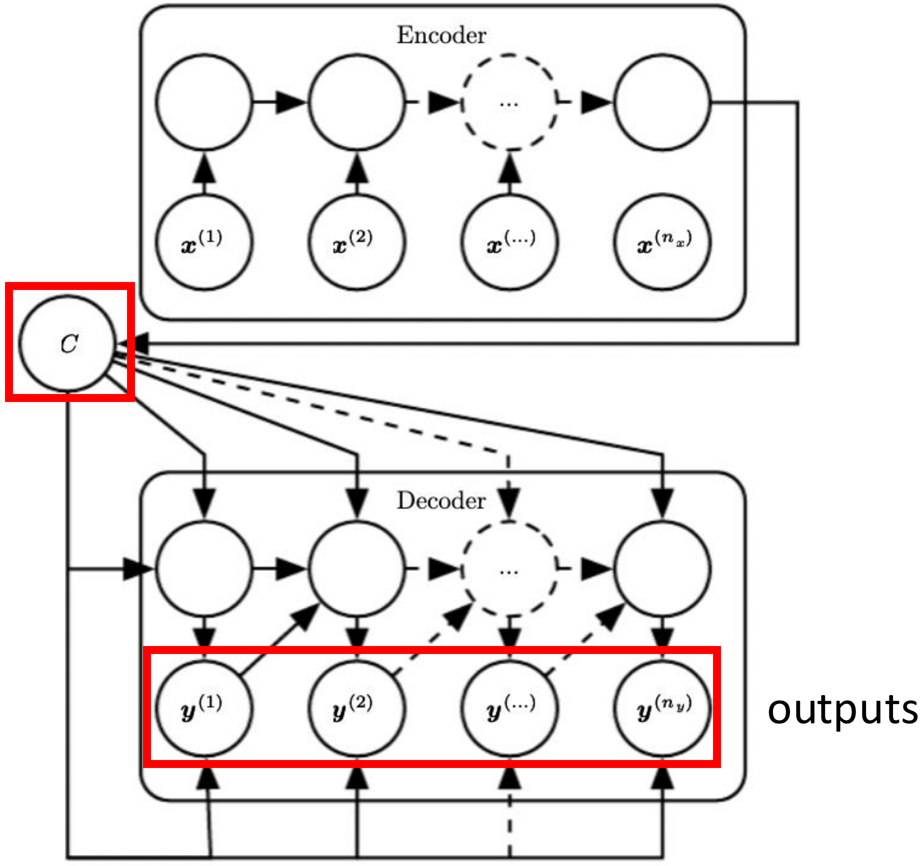
context
(feature)



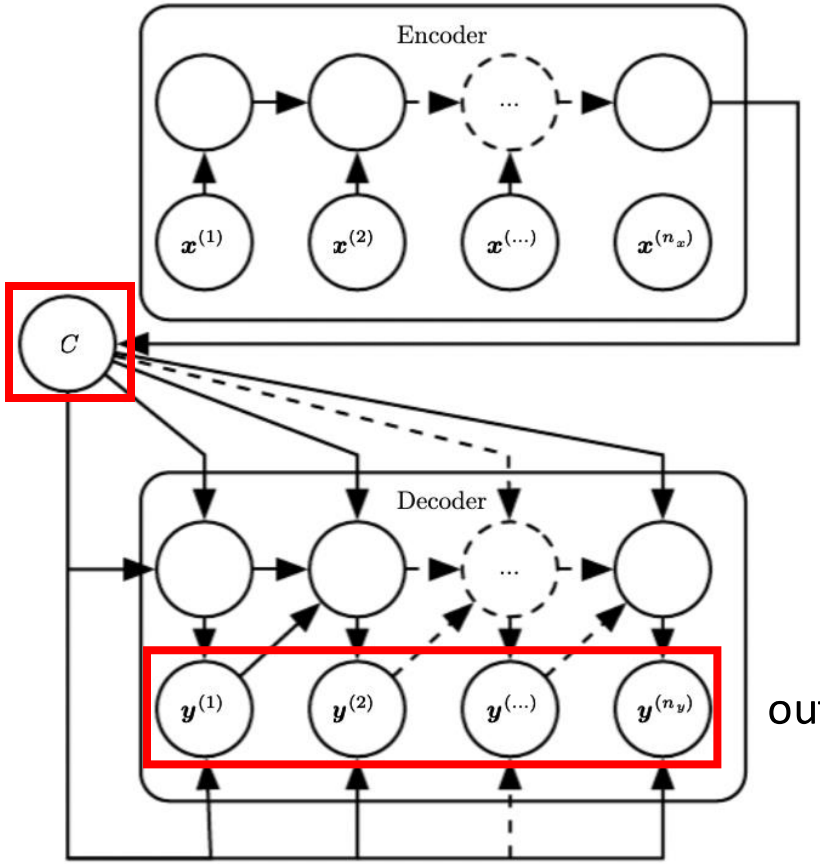
Seq2seq RNN architecture



Seq2seq RNN architecture



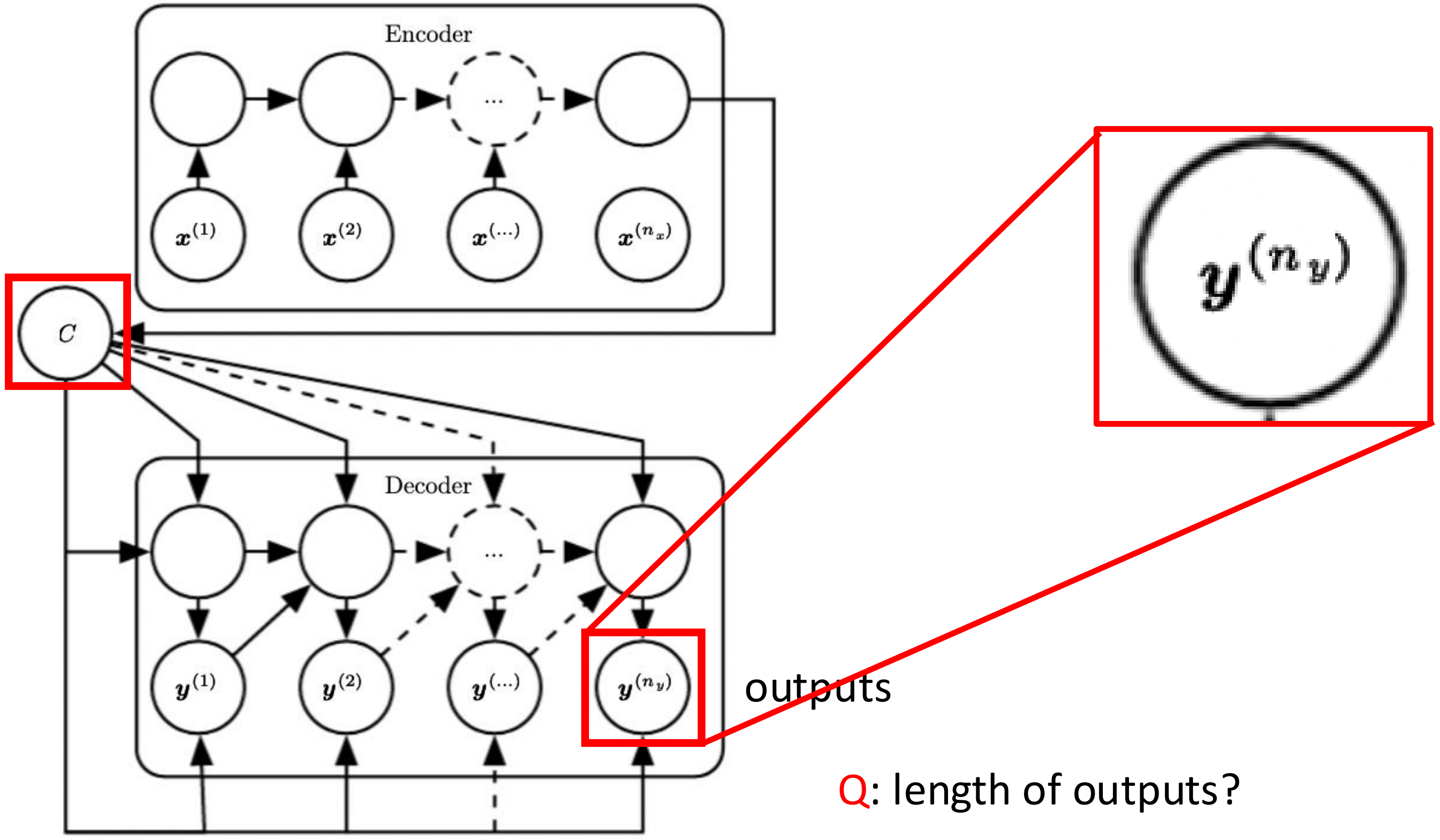
Seq2seq RNN architecture



outputs

Q: length of outputs?

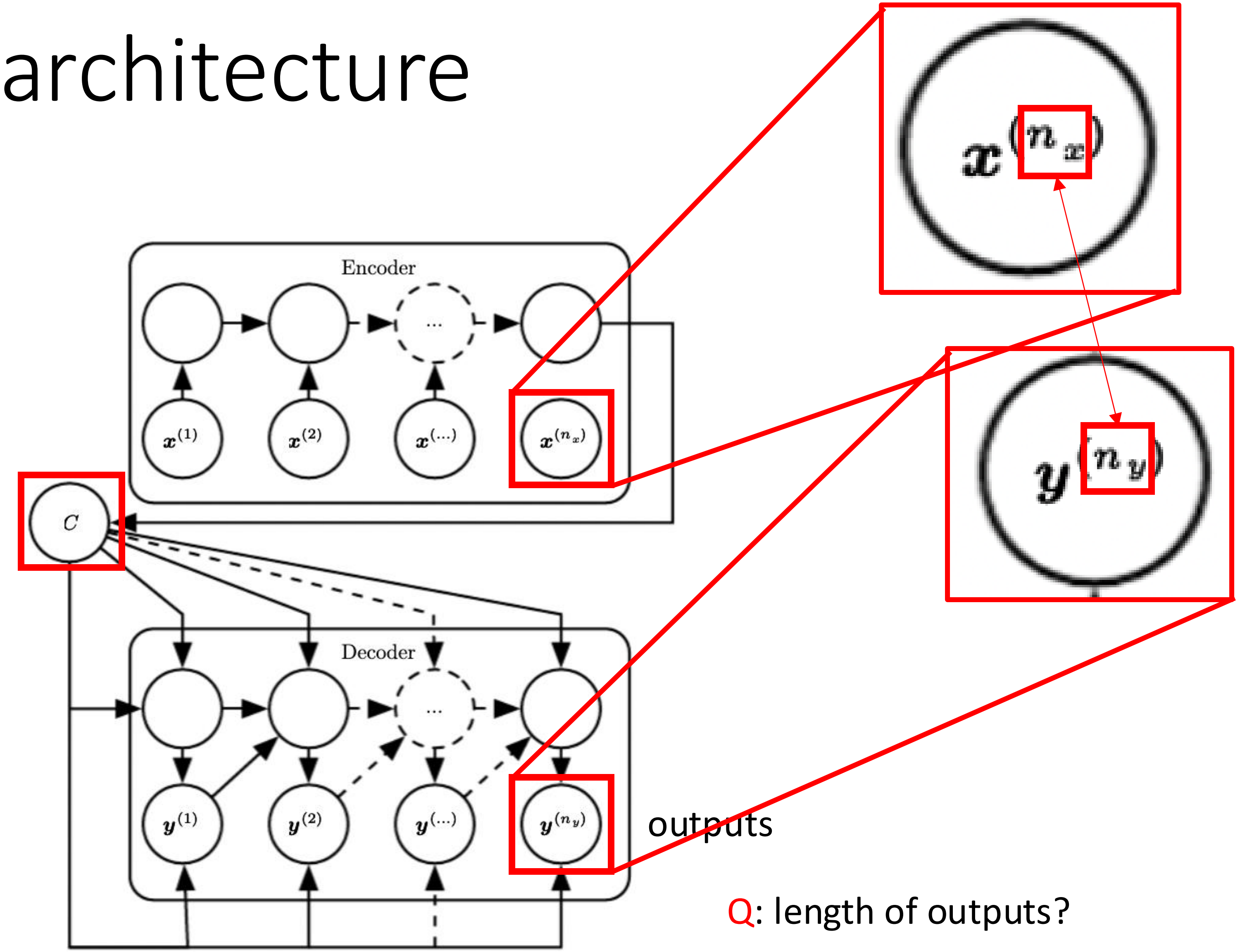
Seq2seq RNN architecture



outputs

Q: length of outputs?

Seq2seq RNN architecture



Q: length of outputs?