Recurrent Neural Networks

Directed Reading Course (Part Two)

Arman Afrasiyabi
Electrical Engineering and Computer Engineering Department
Université Laval

UNIVERSITÉ LAVAL

Content

- Motivation
- Vanilla Recurrent Neural Network (RNNs)
- Bidirectional RNN (Bi-RNN)
- Problems of Vanilla RNNs
- Gated RNNs
    - Gated Recurrent Units (GRU)
    - Long Short-Term Memory(LSTM)
- Examples:
    - Encoder-Decoder RNNs
    - Attention based RNNs

Motivation

The problem of language model

$$P(\mathrm{w}_1, \ldots, \mathrm{w}_m)$$

Traditional methods

$$p(\mathrm{w}_2 \mid \mathrm{w}_1) = \frac{\mathrm{count}(\mathrm{w}_1, \mathrm{w}_2)}{\mathrm{count}(\mathrm{w}_1)}$$

$$p(\mathrm{w}_3 \mid \mathrm{w}_1, \mathrm{w}_2) = \frac{\mathrm{count}(\mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3)}{\mathrm{count}(\mathrm{w}_1, \mathrm{w}_2)}$$

$$\vdots$$

Motivation

Markov assumption

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1})$$
$$\approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

Motivation

Markov assumption

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1})$$

$$\approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1})$$

Necessary…

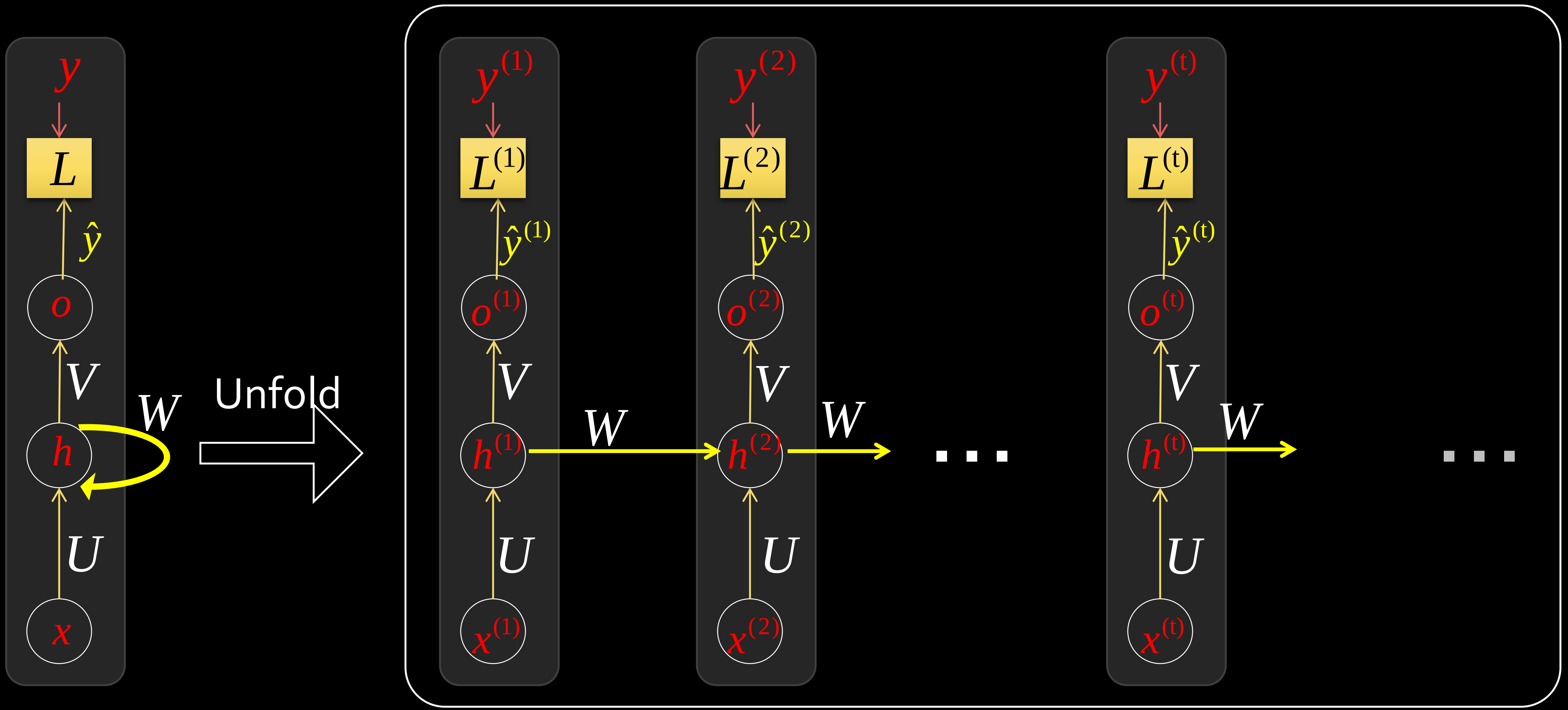
Incorrect in many cases!

Recurrent Neural Networks (RNNs)

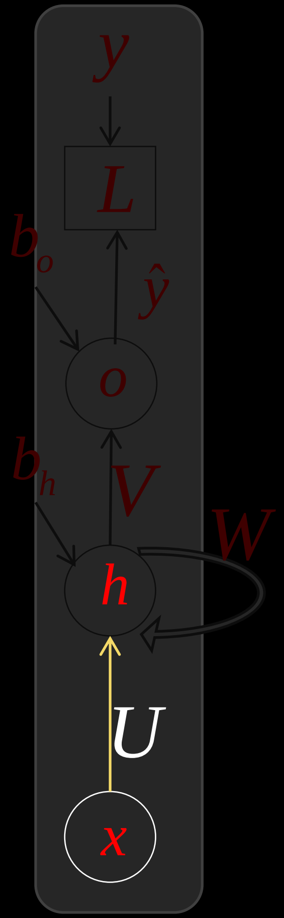
RNNs are a type of neural networks with the following goal:

$$\hat{P}\left(x_{t+1} = \hat{y} \mid x_t, \ldots, x_1\right)$$

Vanilla Recurrent Neural Networks

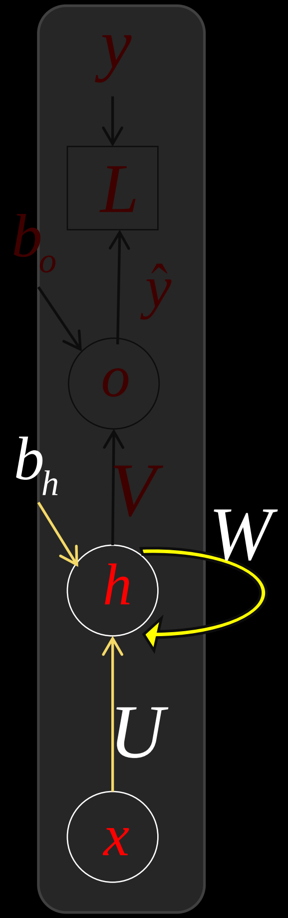


Forward Pass



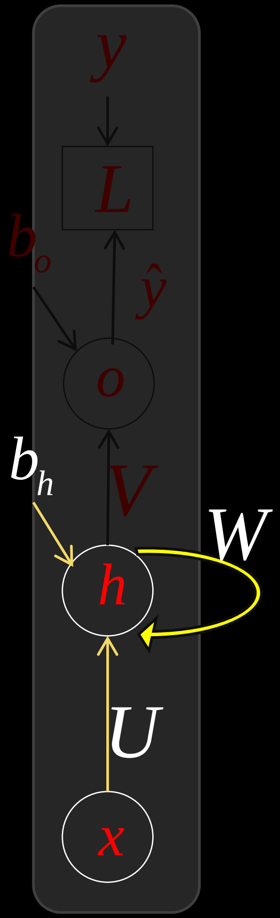
$$Ux^{(t)}$$

Forward Pass



$$h^{(t)} = \varphi\left(Ux^{(t)} + Wh^{(t-1)} + b_h\right)$$

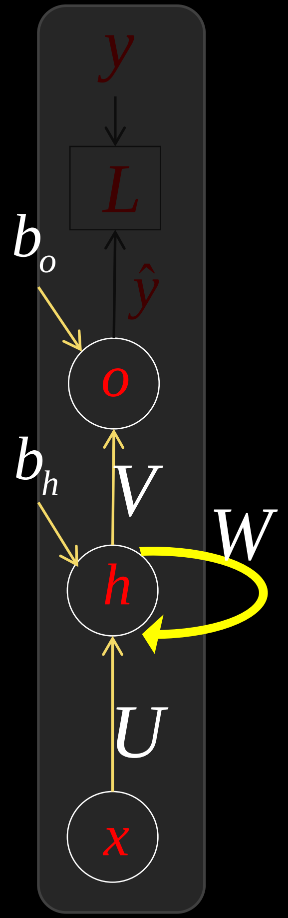
Forward Pass



$$h^{(t)} = \varphi\left(Ux^{(t)} + Wh^{(t-1)} + b_h\right)$$

$$\begin{aligned}(\mathrm{n}\times 1) &= \varphi\left((\mathrm{n}\times m)(m\times 1) + (\mathrm{n}\times n)(n\times 1) + (\mathrm{n}\times 1)\right)\\ &= \varphi\left((\mathrm{n}\times 1) + (\mathrm{n}\times 1) + (\mathrm{n}\times 1)\right)\\ &= \varphi\left(\mathrm{n}\times 1\right)\end{aligned}$$

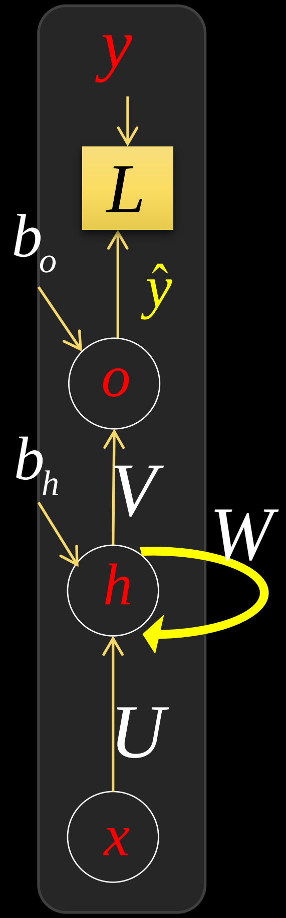
Forward Pass



$$h^{(t)} = \varphi\left(Ux^{(t)} + Wh^{(t-1)} + b_h\right)$$

$$o^{(t)} = Vh^{(t)} + b_o$$

Forward Pass



$$h^{(t)} = \varphi\left(Ux^{(t)} + Wh^{(t-1)} + b_h\right)$$

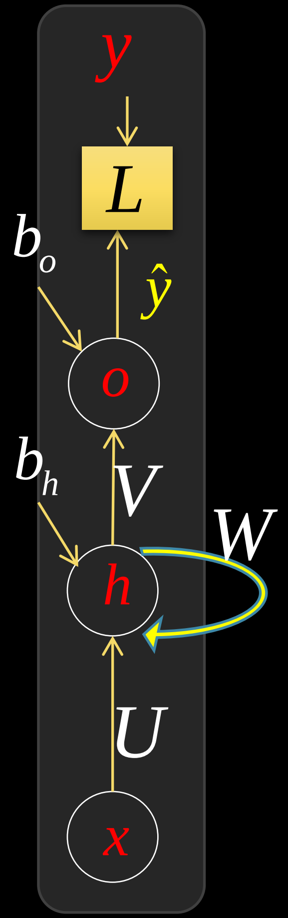
$$o^{(t)} = Vh^{(t)} + b_o$$

Example:

$$\hat{y}^{(t)} = \mathrm{softmax}(o^{(t)})$$

$$\mathrm{L} = \log-\mathrm{likelihood}$$

Forward Pass



$$h^{(t)} = \varphi\left(Ux^{(t)} + Wh^{(t-1)} + b_h\right)$$

$$o^{(t)} = Vh^{(t)} + b_o$$

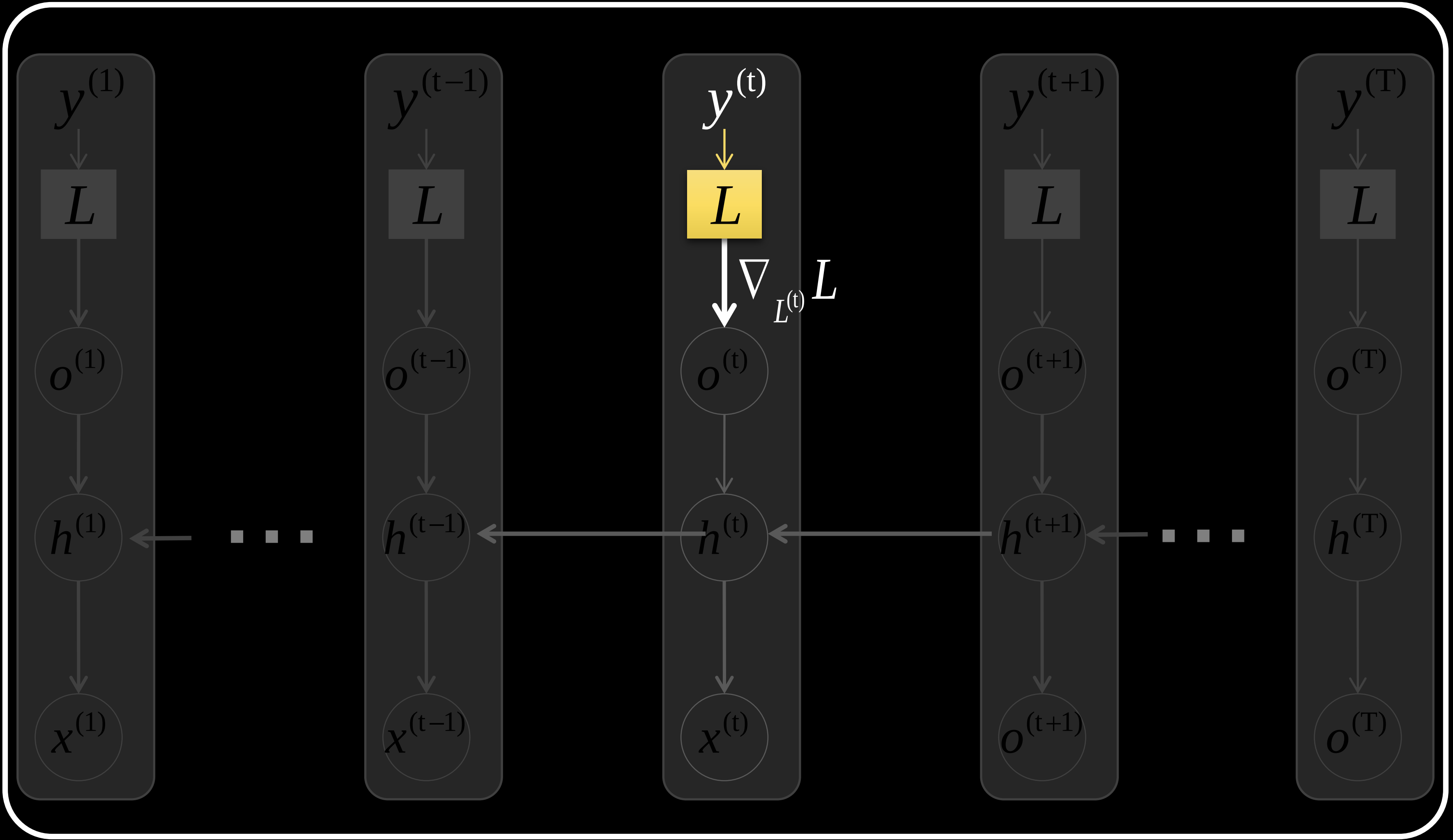
Example:

$$\hat{y}^{(t)} = \text{softmax}(o_t)$$
$$\text{L} = \text{log} - \text{likelihood}$$

In some cases $\hat{y}^{(t)}$ is optional ☺

Back-Propagation Through Time (BPTT)

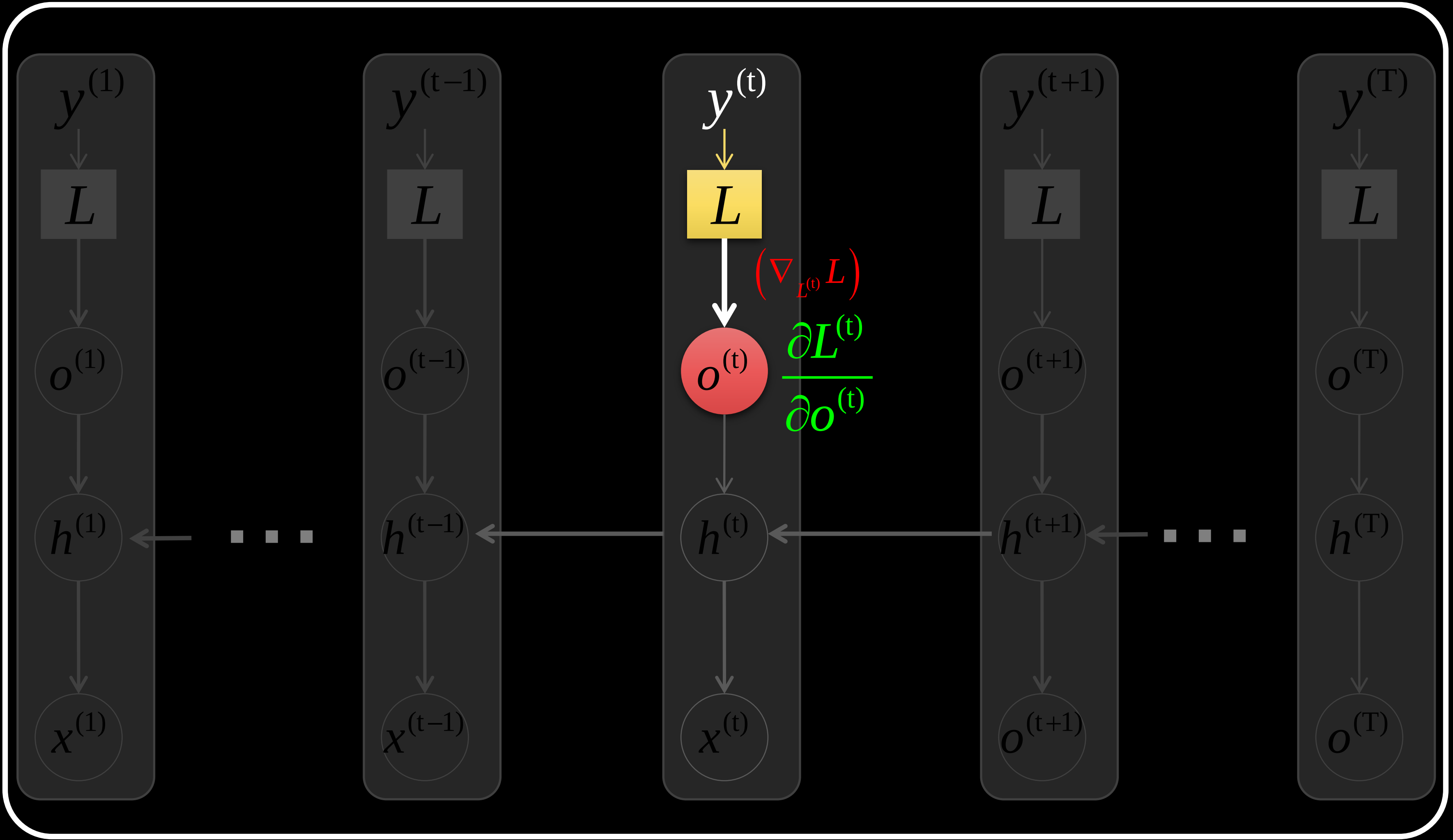
$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$



Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right) \frac{\partial L^{(t)}}{\partial o^{(t)}}$$

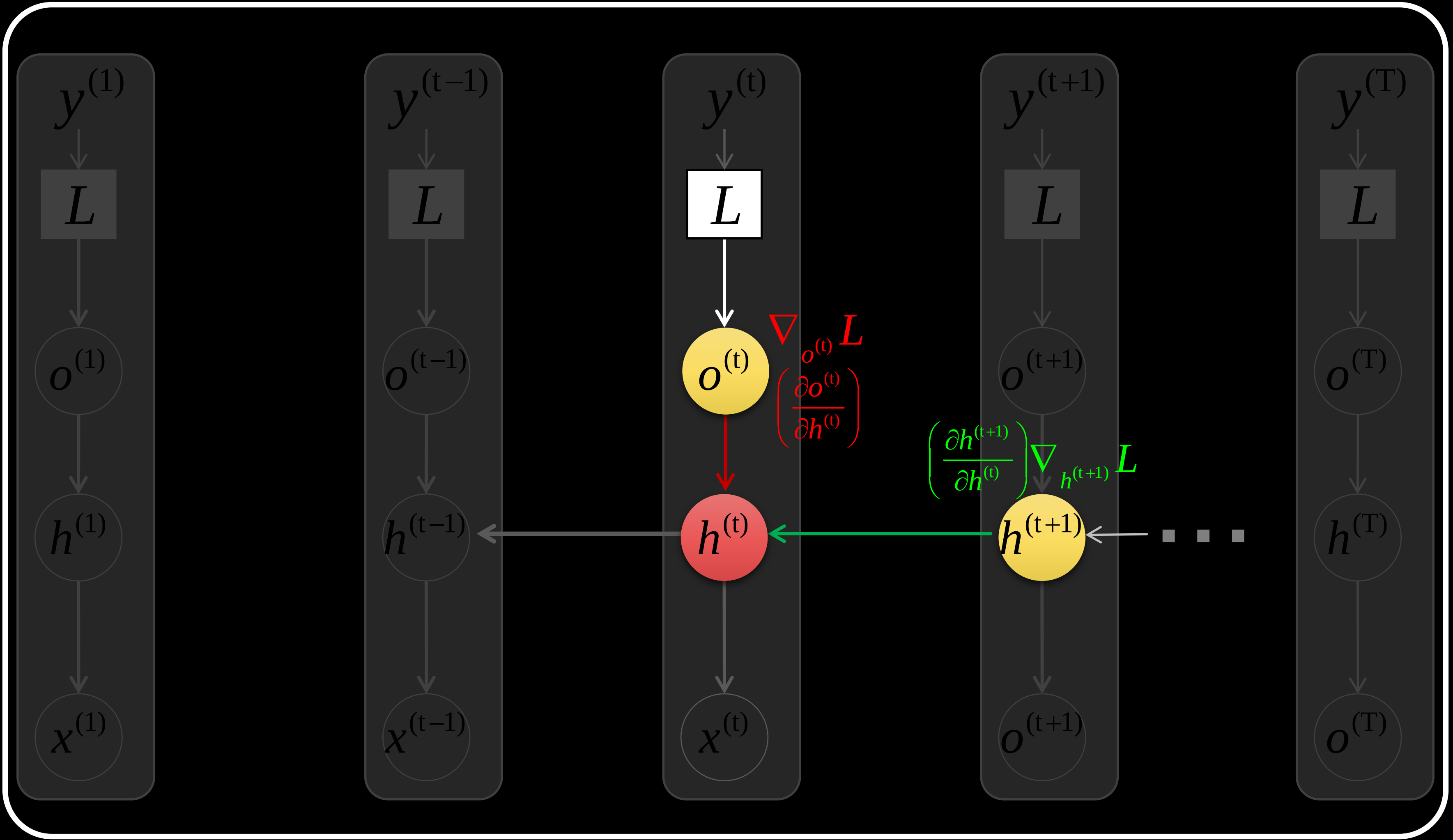


Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right) \frac{\partial L^{(t)}}{\partial o_i^{(t)}}$$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right) \nabla_{o^{(t)}} L + \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right) \nabla_{h^{(t+1)}} L$$



Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

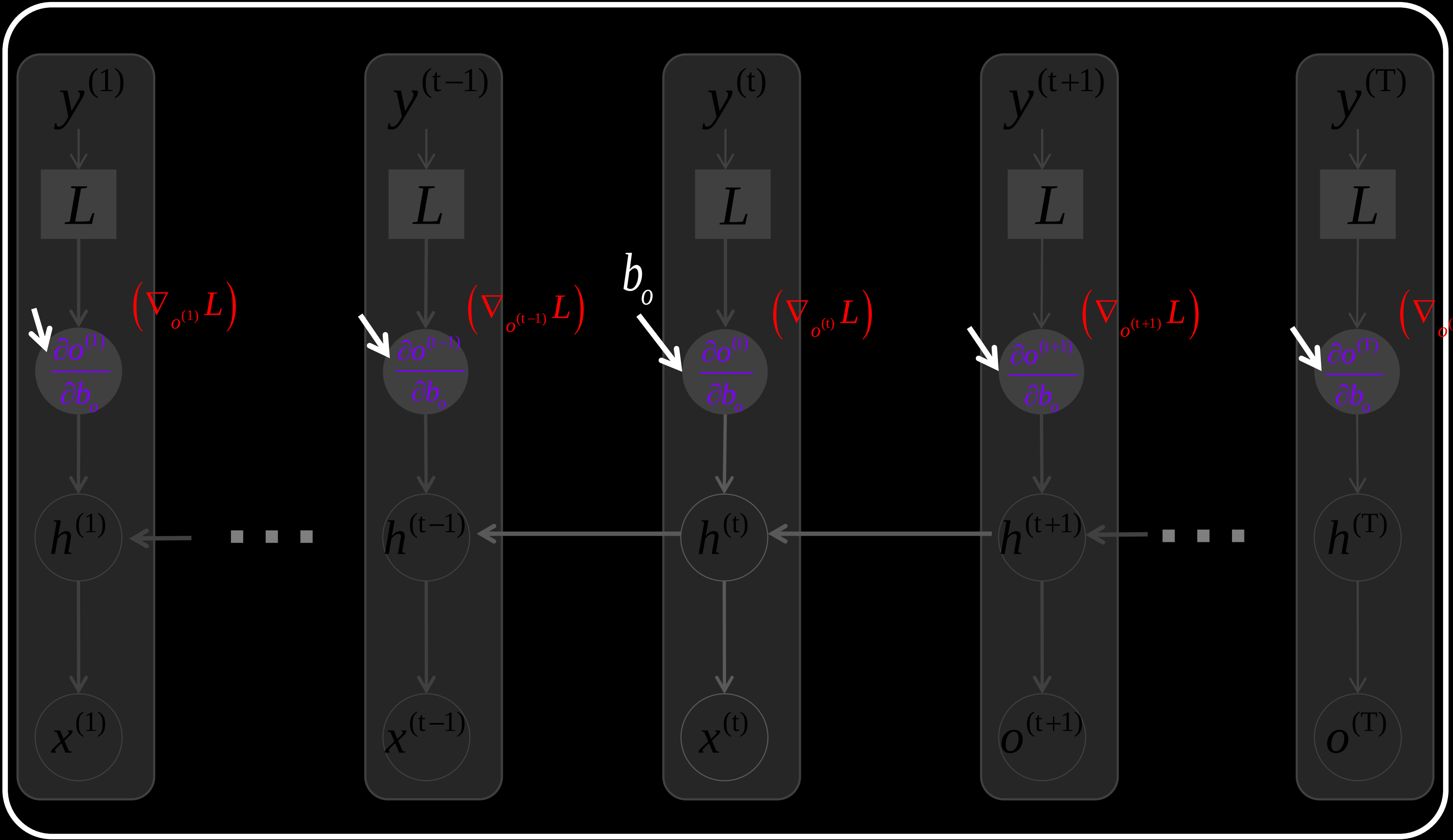
$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right)\frac{\partial L^{(t)}}{\partial o_i^{(t)}}$$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)\nabla_{h^{(t+1)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)\nabla_{o^{(t)}} L$$

$$\nabla_{b_o} L = \sum_{t=1}^{T}\left(\frac{\partial o^{(t)}}{\partial b_o}\right)\left(\nabla_{o^{(t)}} L\right)$$

biases

Weights



Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right)\frac{\partial L^{(t)}}{\partial o_i^{(t)}}$$

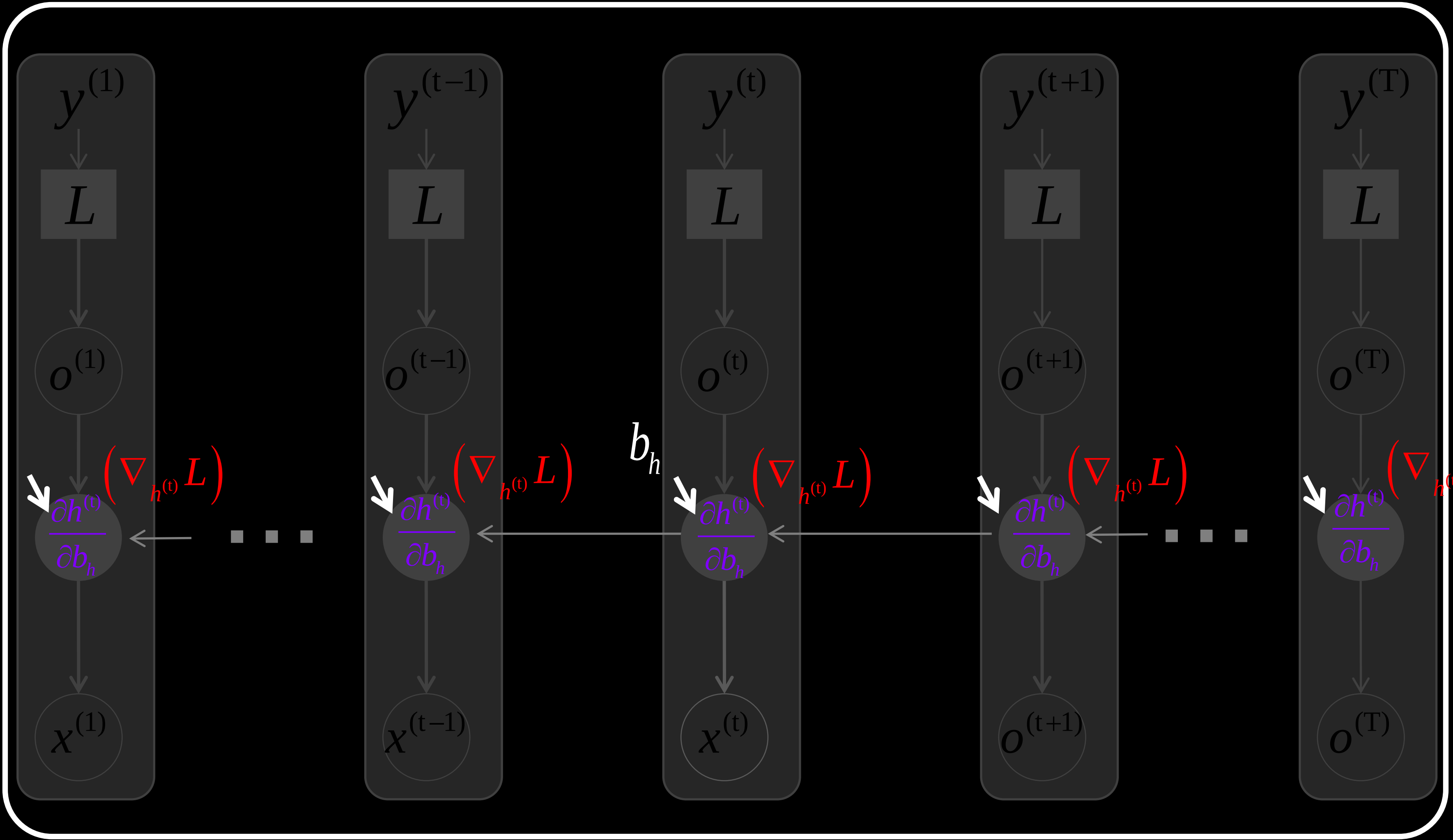
$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right)\nabla_{h^{(t+1)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right)\nabla_{o^{(t)}} L$$

biases

$$\nabla_{b_o} L = \sum_{t=1}^{T}\left(\frac{\partial o^{(t)}}{\partial b_o}\right)\left(\nabla_{o^{(t)}} L\right)$$

$$\nabla_{b_h} L = \sum_{t=1}^{T}\left(\frac{\partial h^{(t)}}{\partial b_h}\right)\left(\nabla_{h^{(t)}} L\right)$$

Weights



Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right) \frac{\partial L^{(t)}}{\partial o_i^{(t)}}$$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right) \nabla_{h^{(t+1)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right) \nabla_{o^{(t)}} L$$

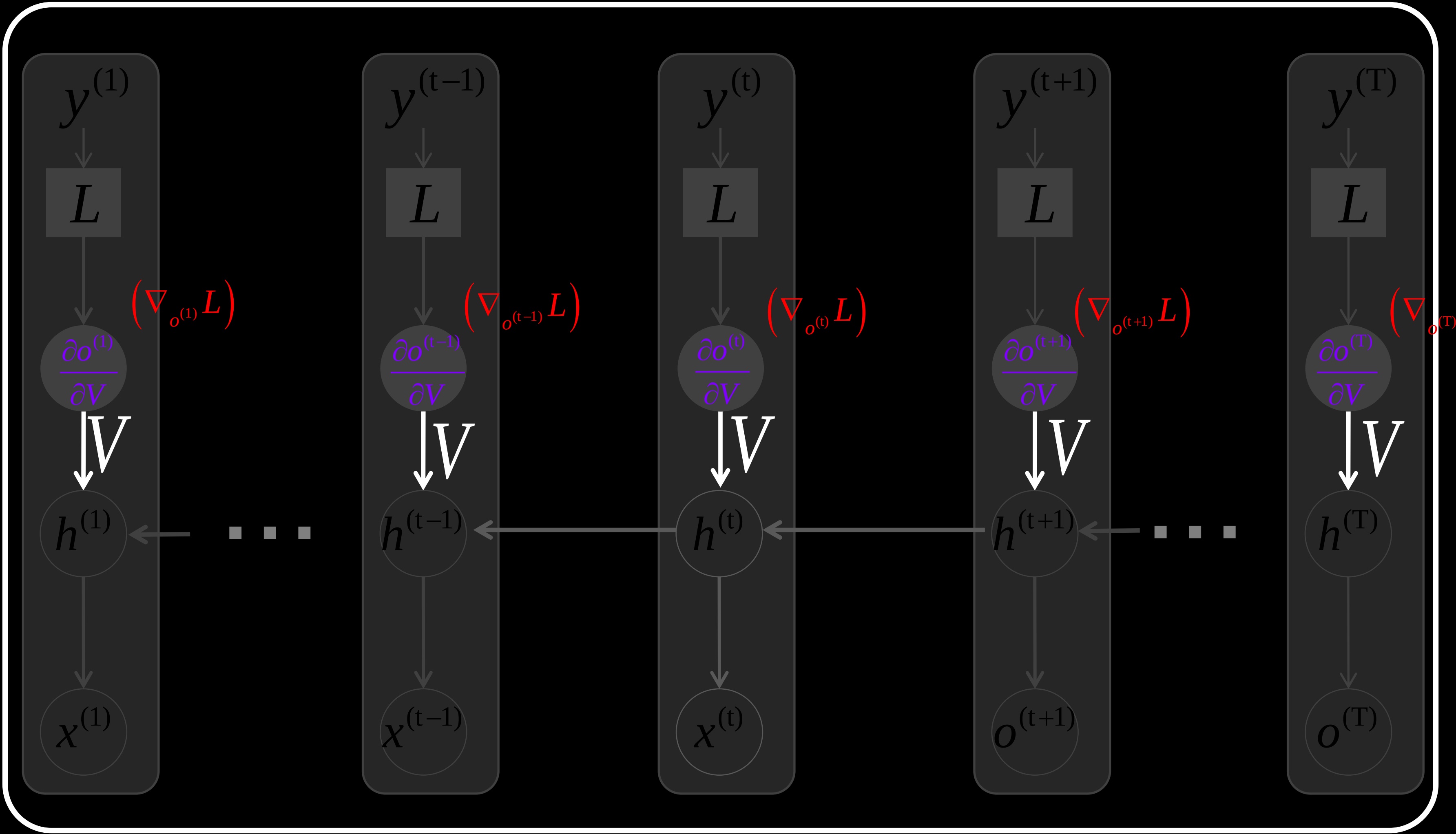
biases

$$\nabla_{b_o} L = \sum_{t=1}^{T} \frac{\partial o^{(t)}}{\partial b_o} \left(\nabla_{o^{(t)}} L\right)$$

$$\nabla_{b_h} L = \sum_{t=1}^{T} \left(\frac{\partial h^{(t)}}{\partial b_h}\right) \left(\nabla_{h^{(t)}} L\right)$$

Weights

$$\nabla_{V} L = \sum_{t=1}^{T} \left(\frac{\partial o^{(t)}}{\partial V}\right) \left(\nabla_{o^{(t)}} L\right)$$



Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right) \frac{\partial L^{(t)}}{\partial o_i^{(t)}}$$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right) \nabla_{h^{(t+1)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right) \nabla_{o^{(t)}} L$$

biases

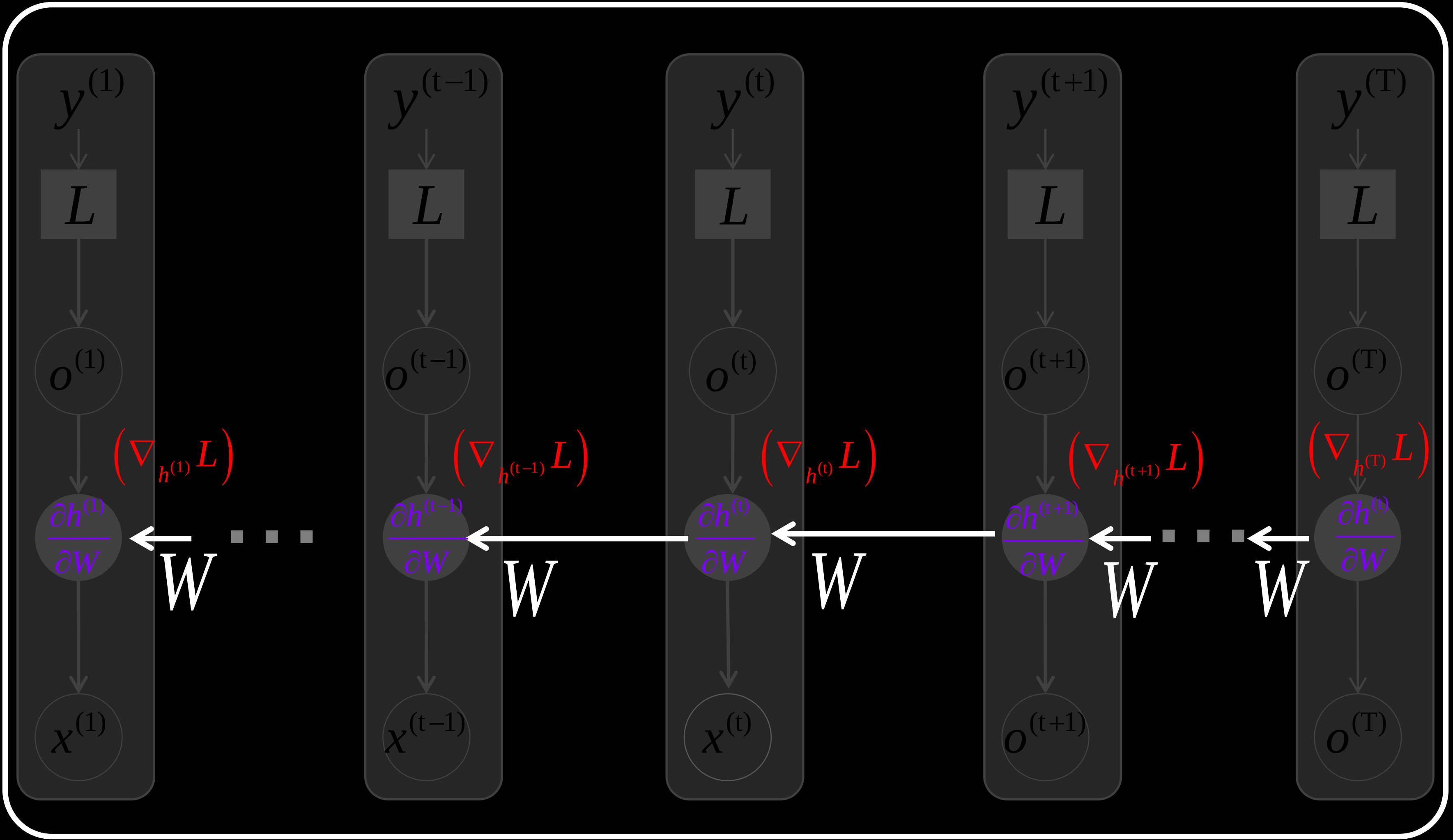
$$\nabla_{b_o} L = \sum_{t=1}^{T} \frac{\partial o^{(t)}}{\partial b_o} \left(\nabla_{o^{(t)}} L\right)$$

$$\nabla_{b_h} L = \sum_{t=1}^{T} \left(\frac{\partial h^{(t)}}{\partial b_h}\right) \left(\nabla_{h^{(t)}} L\right)$$

Weights

$$\nabla_{V} L = \sum_{t=1}^{T} \left(\frac{\partial o^{(t)}}{\partial V}\right) \left(\nabla_{o^{(t)}} L\right)$$

$$\nabla_{W} L = \sum_{t=1}^{T} \left(\frac{\partial h^{(t)}}{\partial W}\right) \left(\nabla_{h^{(t)}} L\right)$$



Back-Propagation Through Time (BPTT)

$$\nabla_{L^{(t)}} L = \frac{\partial L}{\partial L^{(t)}}$$

$$\nabla_{o^{(t)}} L = \left(\nabla_{L^{(t)}} L\right) \frac{\partial L^{(t)}}{\partial o_i^{(t)}}$$

$$\nabla_{h^{(t)}} L = \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}}\right) \nabla_{h^{(t+1)}} L + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}}\right) \nabla_{o^{(t)}} L$$

biases

$$\nabla_{b_o} L = \sum_{t=1}^{T} \frac{\partial o^{(t)}}{\partial b_o} \left(\nabla_{o^{(t)}} L\right)$$

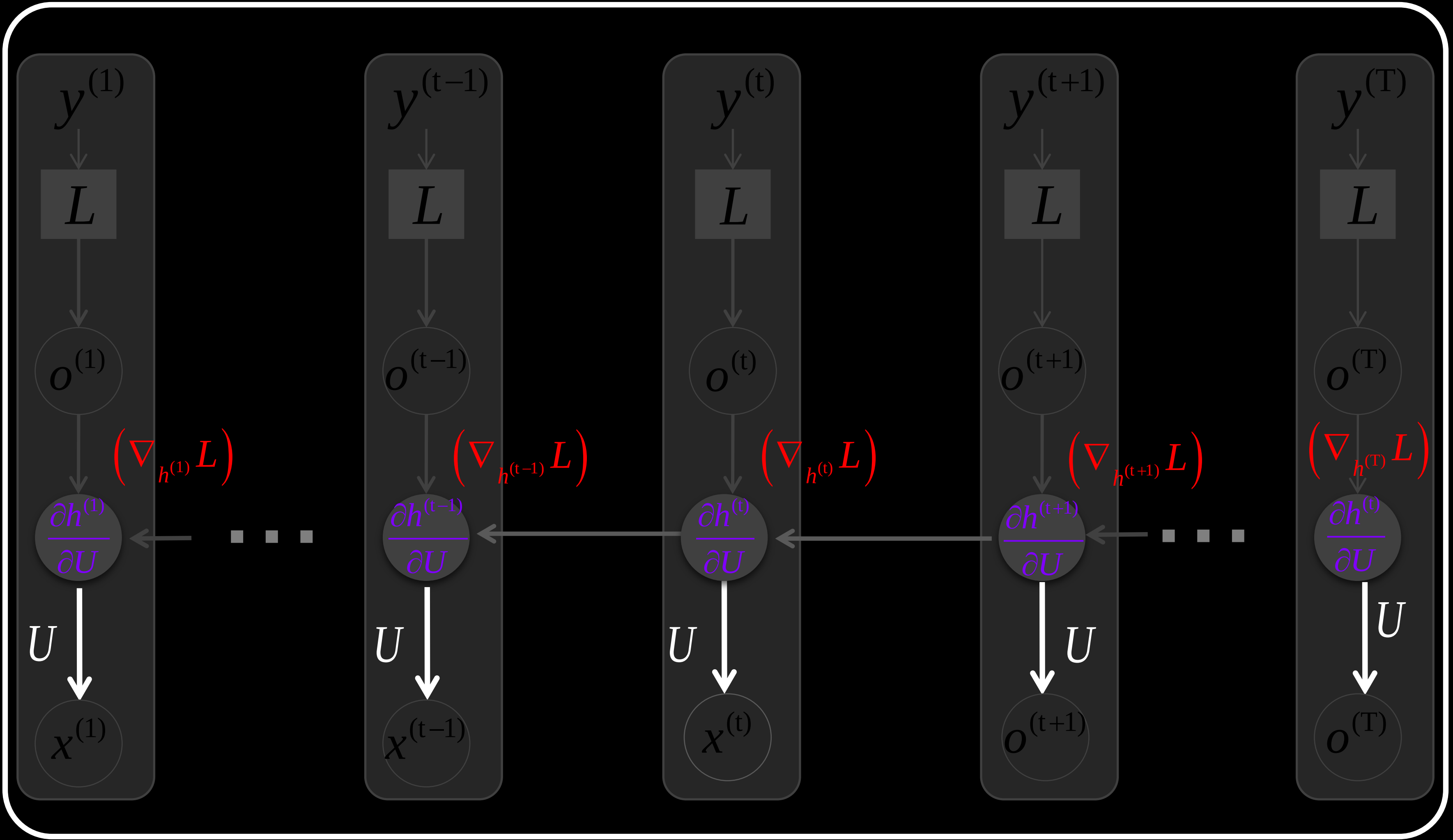
$$\nabla_{b_h} L = \sum_{t=1}^{T} \left(\frac{\partial h^{(t)}}{\partial b_h}\right) \left(\nabla_{h^{(t)}} L\right)$$

Weights

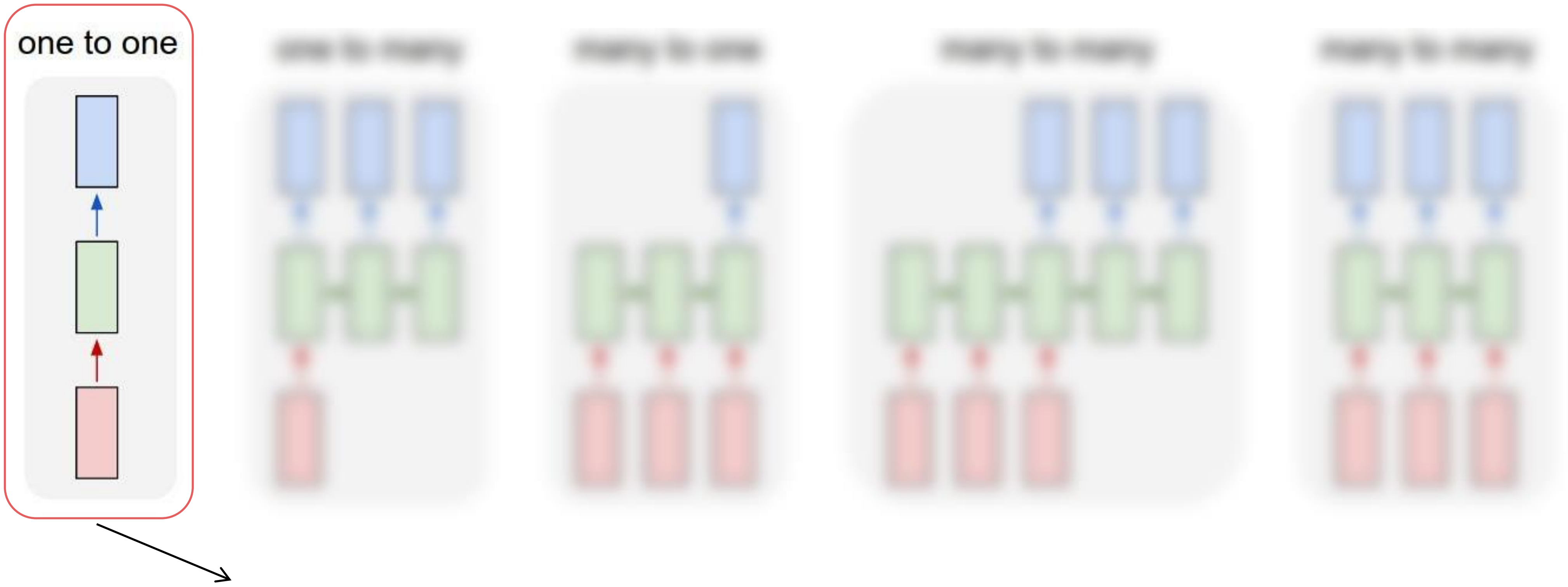
$$\nabla_{V} L = \sum_{t=1}^{T} \left(\nabla_{o^{(t)}} L\right) \left(\frac{\partial o^{(t)}}{\partial V}\right)$$

$$\nabla_{W} L = \sum_{t=1}^{T} \left(\nabla_{h^{(t)}} L\right) \left(\frac{\partial h^{(t)}}{\partial W}\right)$$

$$\nabla_{U} L = \sum_{t=1}^{T} \left(\frac{\partial h^{(t)}}{\partial U}\right) \left(\nabla_{h^{(t)}} L\right)$$



RNN: one to one



Vanilla Neural Networks

RNN: one to many

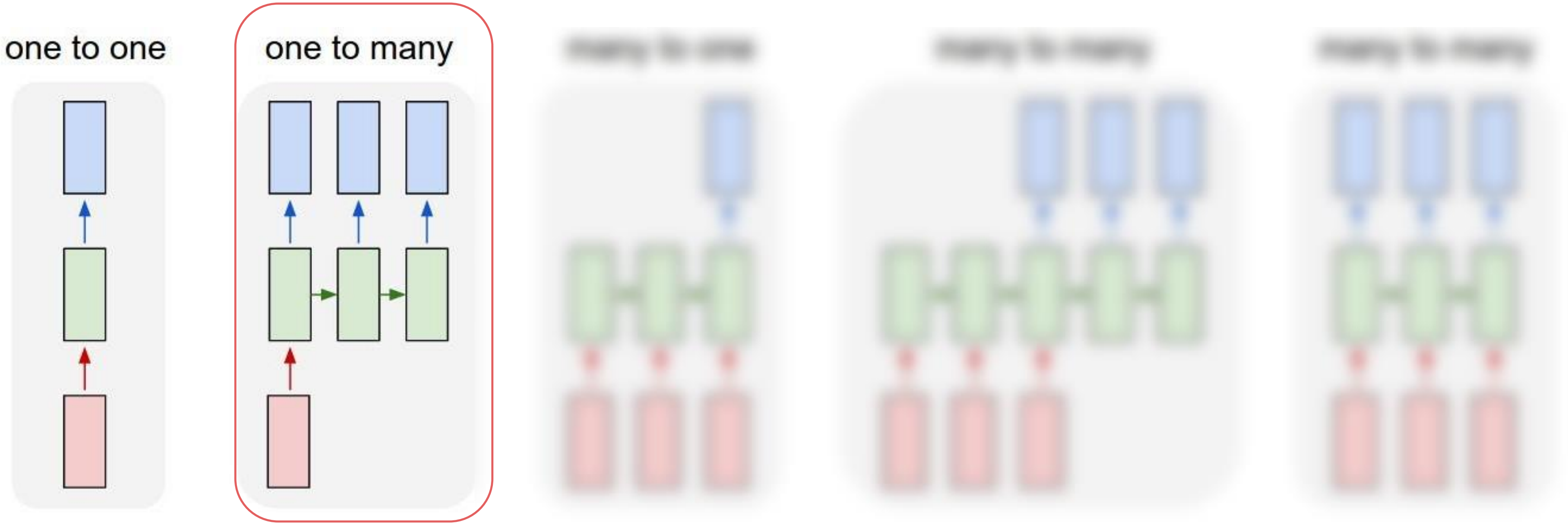
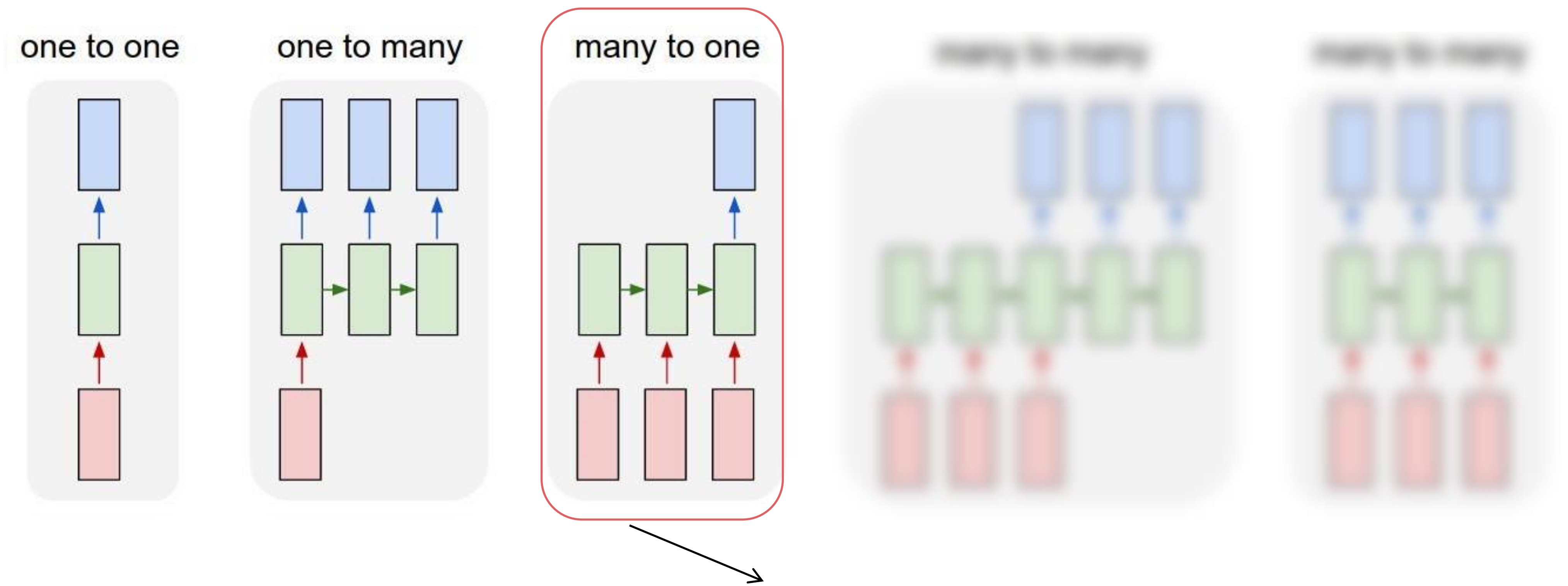


Image Captioning
image -> sequence of words

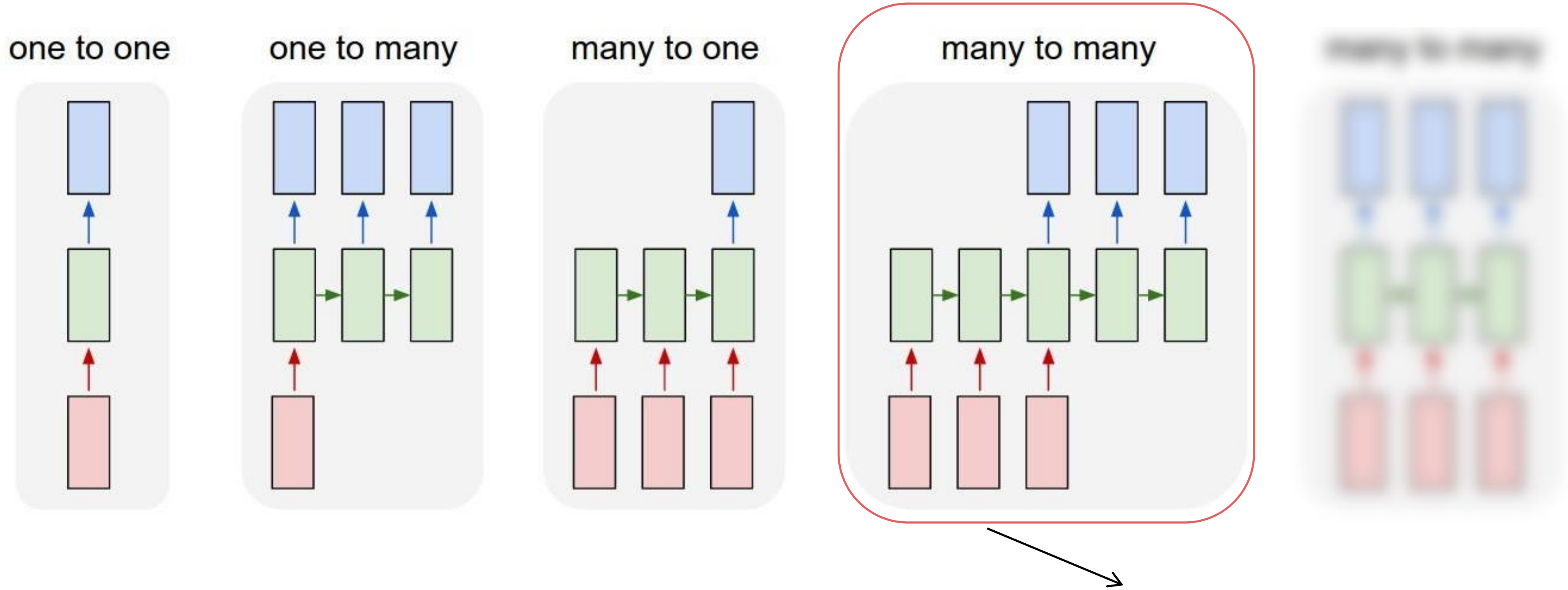
Slides: Fei-Fei Li & Justin Johnson & Serena Yeung

RNN: many to one



Sentiment Classification
sequence of words -> sentiment

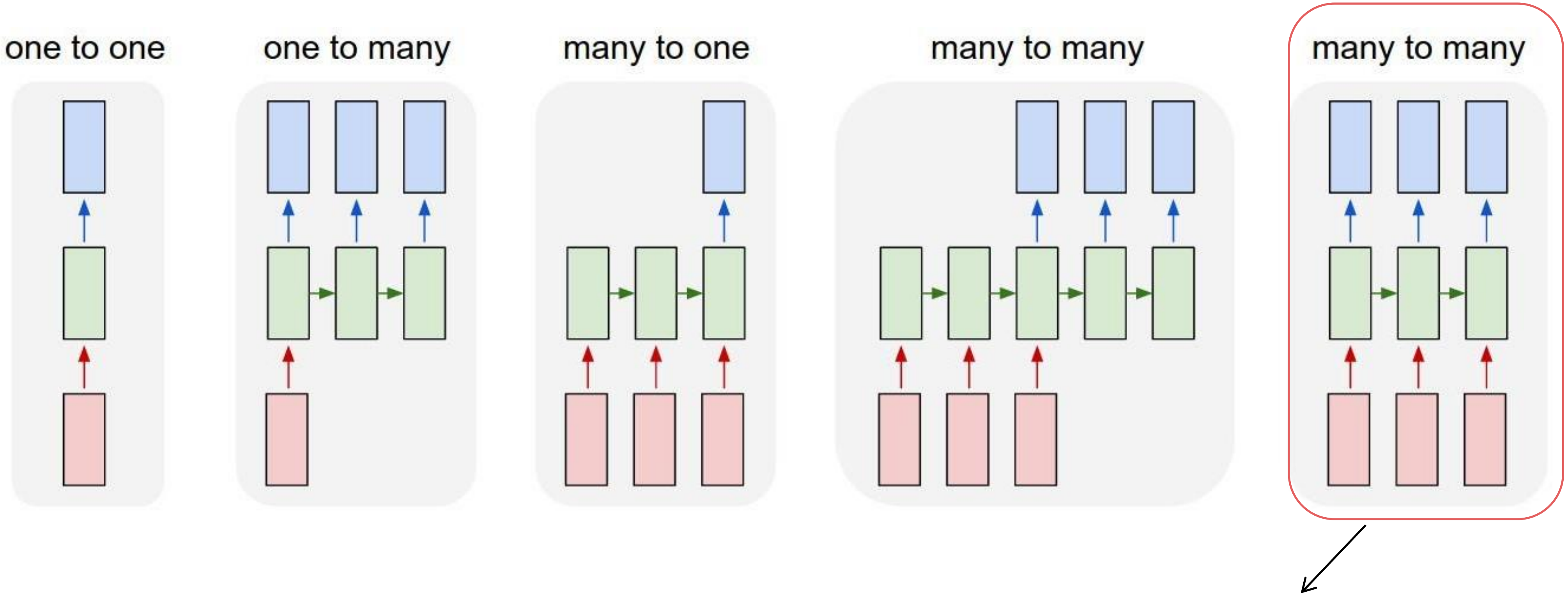
RNN: many to many



Machine Translation
seq of words -> seq of words

Slides: Fei-Fei Li & Justin Johnson & Serena Yeung

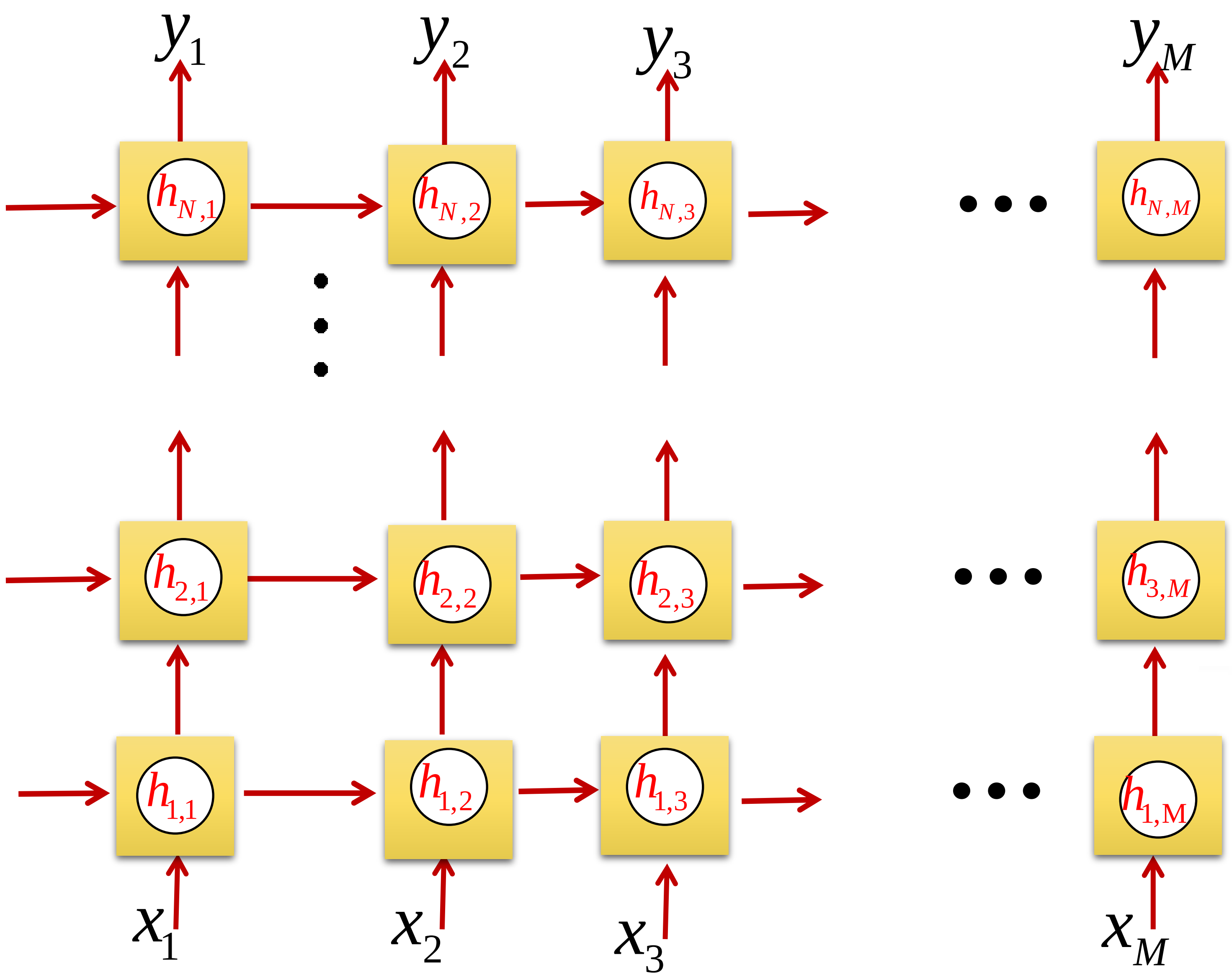
RNN: many to many



Video classification on frame level

Slides: Fei-Fei Li & Justin Johnson & Serena Yeung

Deep RNNs



y_1
y_2
y_3
y_M
$h_{N,1}$
$h_{N,2}$
$h_{N,3}$
$h_{N,M}$
$h_{2,1}$
$h_{2,2}$
$h_{2,3}$
$h_{3,M}$
$h_{1,1}$
$h_{1,2}$
$h_{1,3}$
$h_{1,M}$
x_1
x_2
x_3
x_M

Bidirectional RNNs

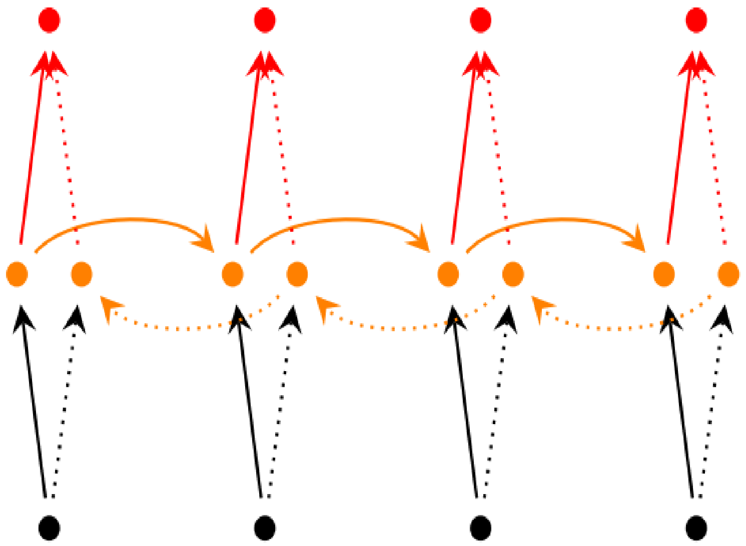
Vanilla RNNs

looks into the one side of sequence (left or past) to predict the next output.

Bidirectional RNNs (BiRNNs)

can focus on both past and future (right side of the sequence).

Bidirectional RNNs

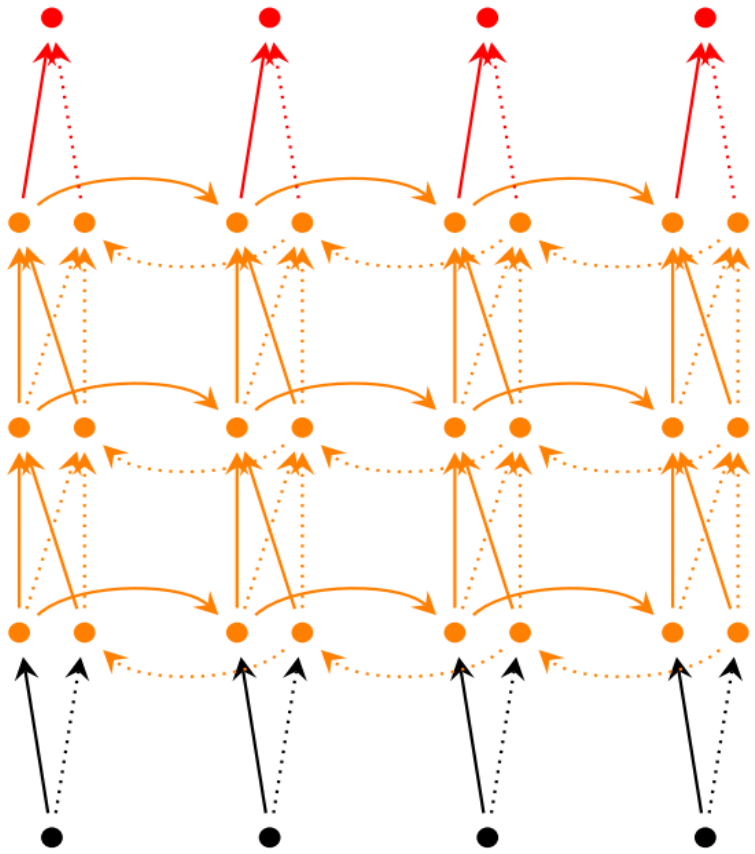


$$\overrightarrow{h}_t = f\left(\overrightarrow{U}x_t + \overrightarrow{W}h_{t-1} + \overrightarrow{b}_h\right)$$

$$\overleftarrow{h}_t = f\left(\overleftarrow{U}x_t + \overleftarrow{W}h_{t+1} + \overleftarrow{b}_h\right)$$

$$\hat{y} = g(\mathrm{V}\, h_t + \mathrm{b}_o) = \mathrm{g}(\mathrm{V}[\overrightarrow{h}_t ; \overleftarrow{h}_t] + \mathrm{b}_o)$$

Deep Bidirectional RNNs



$$\vec{h}_t^{(i)} = f\left(\vec{U}^{(i)} h_t^{(i-1)} + \vec{W} h_{t-1}^{(i)} + \vec{b}_h^{(i)}\right)$$

$$\overleftarrow{h}_t^{(i)} = f\left(\overleftarrow{U}^{(i)} h_t^{(i-1)} + \overleftarrow{W} h_{t-1}^{(i)} + \overleftarrow{b}_h^{(i)}\right)$$

$$\hat{y}_{(t)} = g(\mathrm{V}\, h_t + \mathrm{b}_o) = \mathrm{g}(\mathrm{U}[\vec{h}_t^{(\mathrm{L})}; \overleftarrow{h}_t^{(\mathrm{L})}] + \mathrm{b}_o)$$

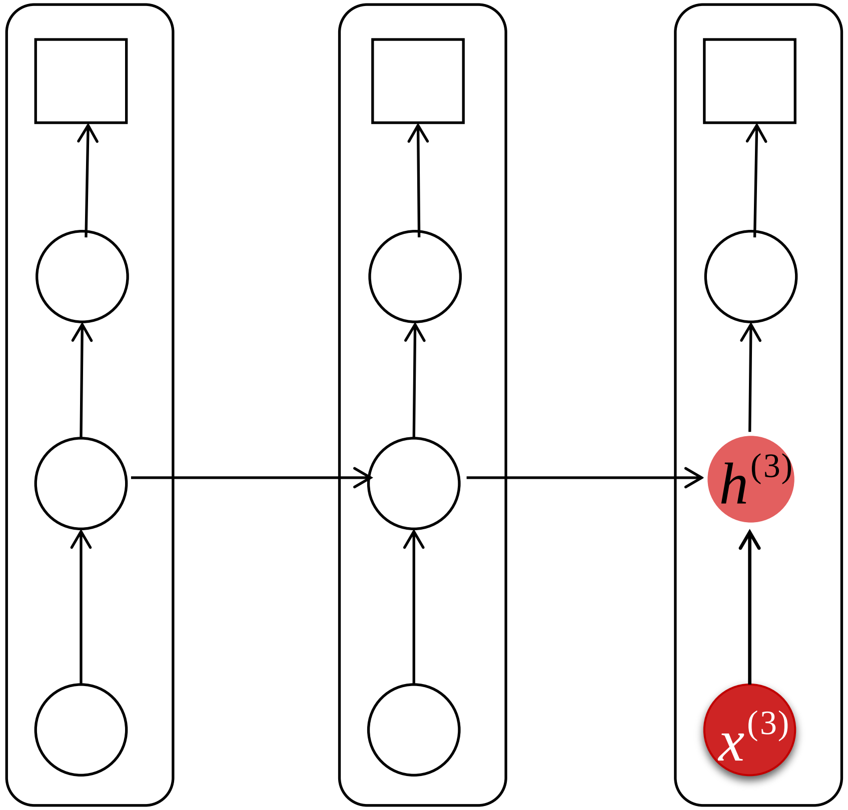
Two Problems of Vanilla RNNS

A) Information Morphing

B) Vanishing/Exploding of the Gradient

Problem 1: Information morphing

Suppose $x^{(3)}$ is important and we must remember it.

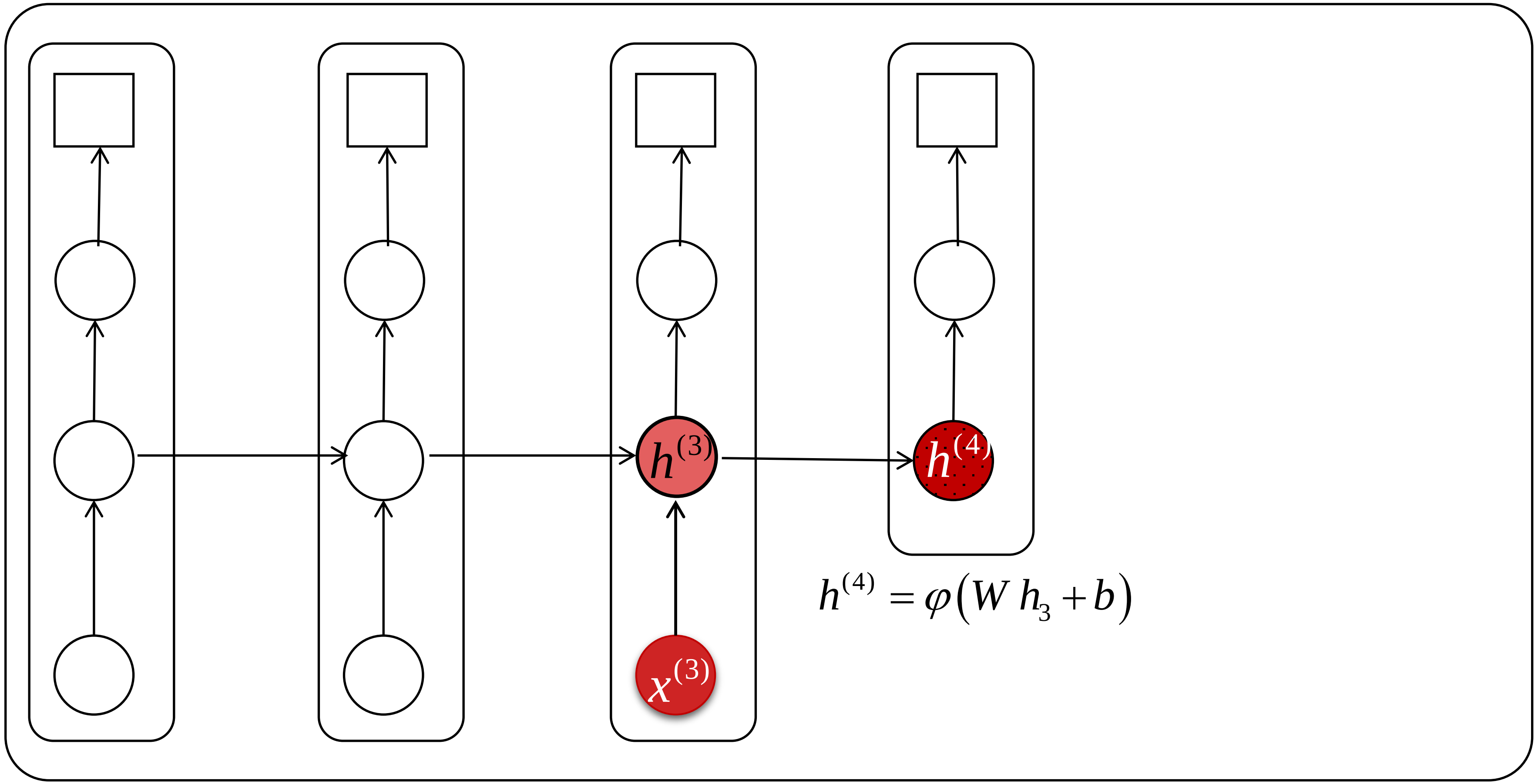


Goal:

$$h_3 = h_4 = \ldots = h_{t+1}$$

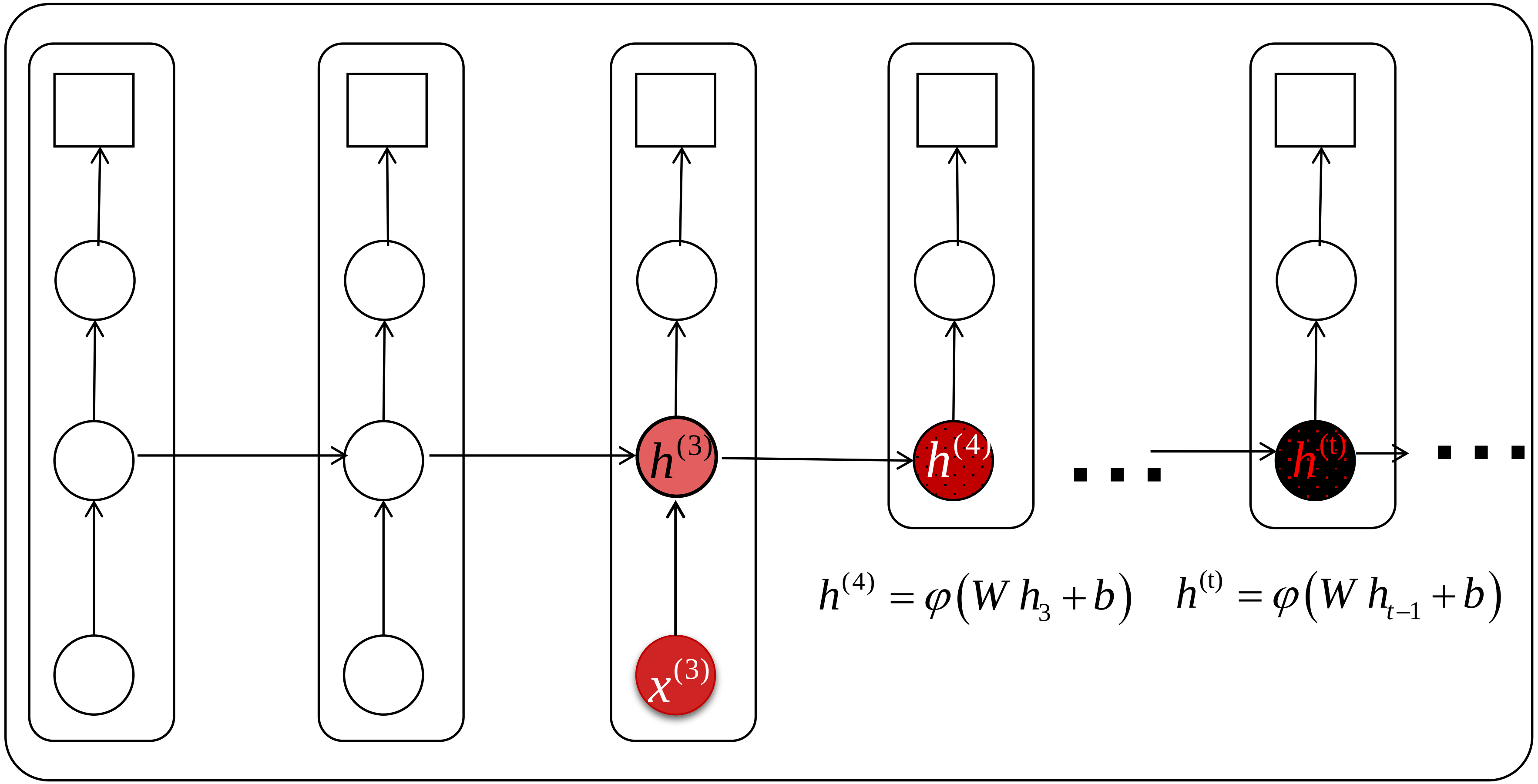
Problem 1: Information morphing

$h_{(4)}$ contains this information.



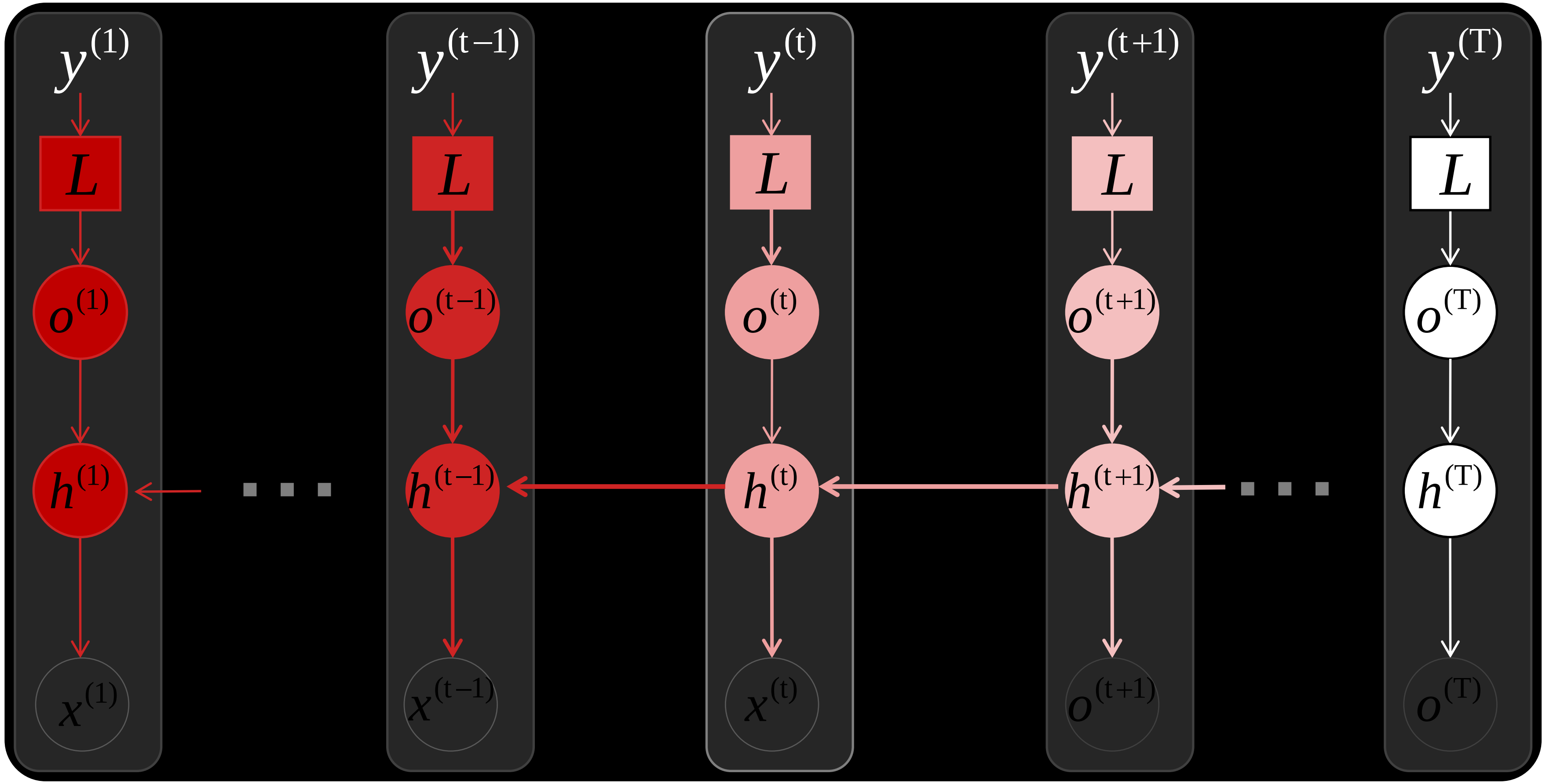
Problem 1: Information morphing

But, this information is washing away as time precedes…



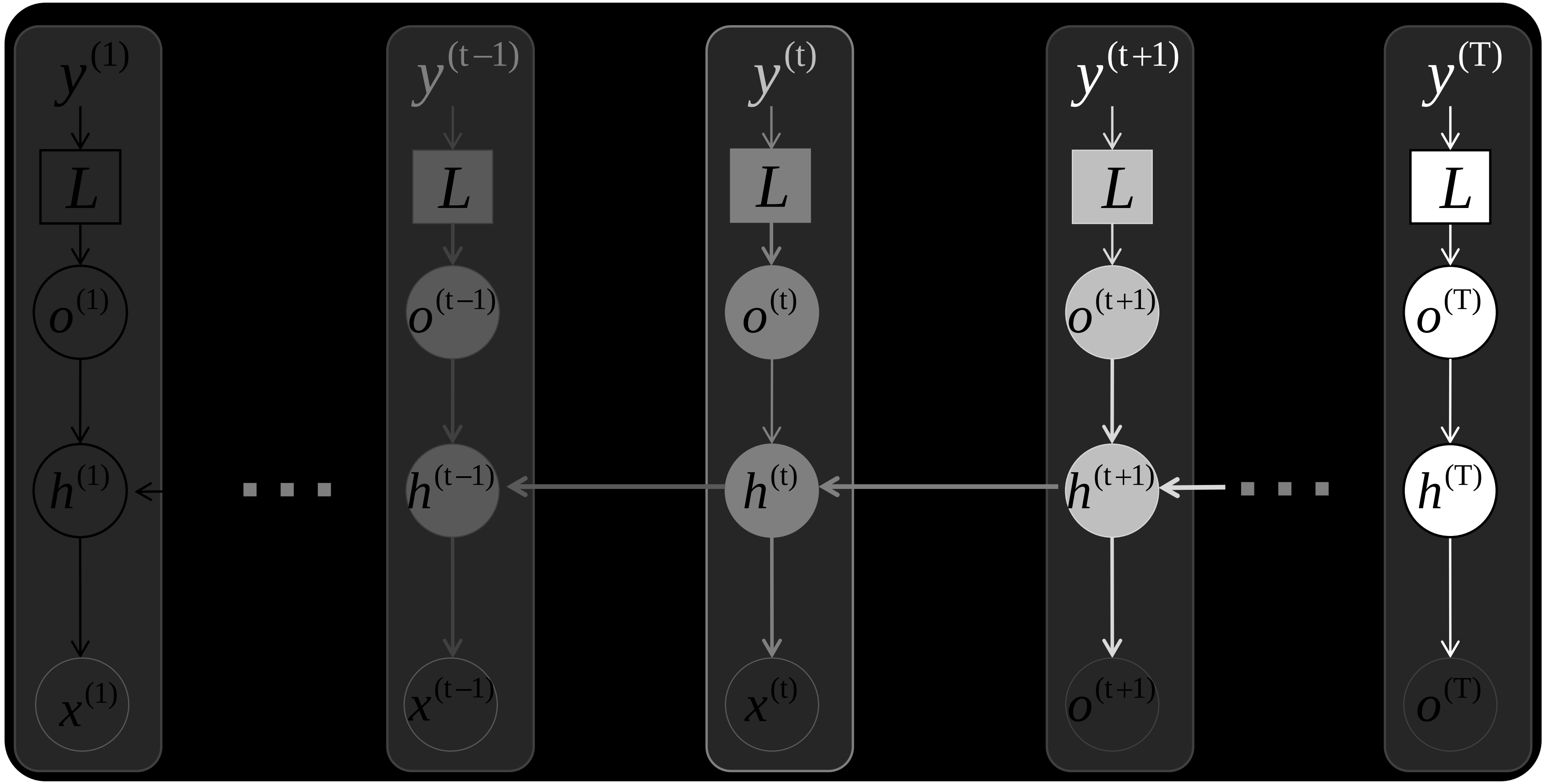
Problem 2: Exploding Gradient

If the weights are big, the gradients grow exponentially.



Problem 2: Vanishing Gradient

If the weights are small, the gradients shrink exponentially.

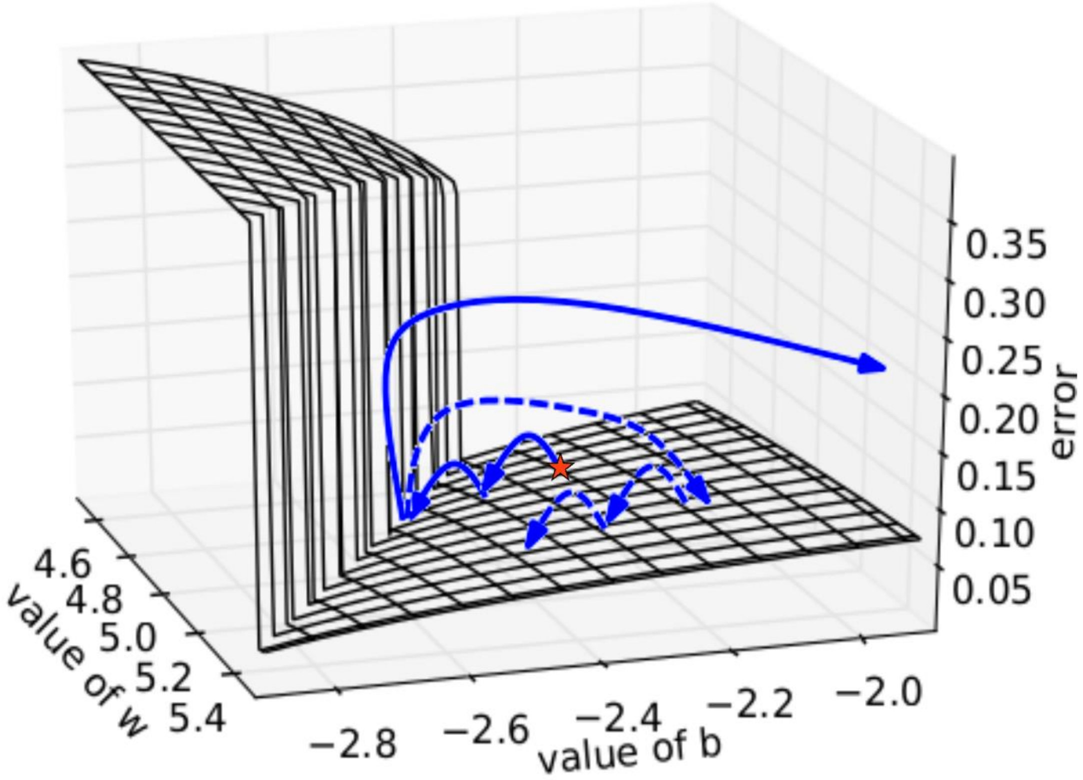


Two obvious solutions

- Better weight initialization (instead of random).
- Rectified Linear Unit (RLU) as activation function.

Gradient clipping

Simple heuristic solution that clips gradients to a small number whenever they explode.



[Thomas Mikolov et al. , Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio, 2013]

Questions?

Can We Change the Structure?

Let's reformulated our vanilla RNN in the following form:

$$\begin{pmatrix} h^{(t)} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix} = \mathit{function}\left[\begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \quad \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix}\right]$$

Can We Change the Structure?

Let's reformulated our vanilla RNN in the following form:

$$\begin{pmatrix} h^{(t)} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix} = \underbrace{function}\left[\begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix}\right]$$

Memory

The Source of problem

Memory's Properties

Memory should has three properties:

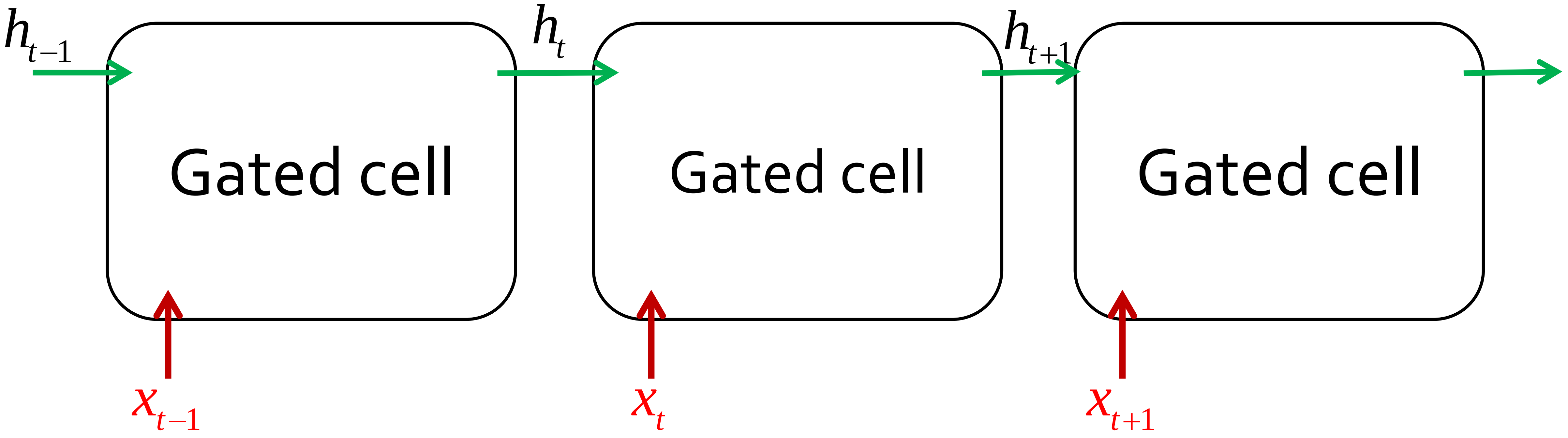
I) Forgettablility

II) Writability

III) Readability

These ideas of memory is used to develop robust RNNS such as GRU and LSTM.

These ideas of memory is used to develop robust gated RNN cells such as GRU and LSTM.



Before, explaining GRU
Let’s talk about its intuition.

Linear operation on Memory

Memory should be a linear combination of Forgettablility and Writability.

$$\begin{pmatrix} h^{(t)} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix} = \begin{pmatrix} \text{Clearning} \\ \text{Memory} \end{pmatrix} + \begin{pmatrix} \text{Writing} \\ \text{Memory} \end{pmatrix}$$

Clearing and Writing on Memory

We need two separate parts to clear the unnecessary information from the memory and writing the new information.

$$\begin{pmatrix} h^{(t)} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix} = \begin{pmatrix} \text{Clearning} \\ \text{Memory} \end{pmatrix} + \begin{pmatrix} \text{Writing} \\ \text{Memory} \end{pmatrix}$$

$$\begin{pmatrix} \text{Clearning} \\ \text{Memory} \end{pmatrix} = \begin{pmatrix} f^{(t)} \\ \text{forget gate} \\ \text{binary vector} \end{pmatrix} \otimes \begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}$$

$$\begin{pmatrix} \text{Writing} \\ \text{Memory} \end{pmatrix} = \begin{pmatrix} s^{(t)} \\ \text{store gate} \\ \text{binary vector} \\ \text{at time } t \end{pmatrix} \otimes \begin{pmatrix} \tilde{h}^{(t)} \\ \text{candidate} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix}$$

$$f^{(t)}, s^{(t)}, \tilde{h}^{(t)}$$

Clearing Memory

$$\begin{pmatrix} \text{Clearning} \\ \text{Memory} \end{pmatrix} = \begin{pmatrix} f^{(t)} \\ \text{forget gate} \\ \text{binary vector} \end{pmatrix} \otimes \begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}$$

$$f^{(t)} = \sigma_f \left[\begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix} \right]$$

Writing on Memory

$$\begin{pmatrix}\text{Writing}\\ \text{Memory}\end{pmatrix} = \begin{pmatrix} s^{(t)} \\ \text{store gate} \\ \text{binary vector} \\ \text{at time } t \end{pmatrix} \otimes \begin{pmatrix} \tilde{h}^{(t)} \\ \text{candidate} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix}$$

$$s^{(t)} = \sigma_s \left[\begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix} \right]$$

$$\begin{pmatrix} \tilde{h}^{(t)} \\ \text{candidate} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix} = \varphi \left[\begin{pmatrix} r^{(t)} \\ \text{read gate} \\ \text{at time } t \end{pmatrix} \otimes \begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix} \right]$$

Writing on Memory

$$\begin{pmatrix}\text{Writing}\\ \text{Memory}\end{pmatrix} = \begin{pmatrix} s^{(t)} \\ \text{store gate} \\ \text{binary vector} \\ \text{at time } t \end{pmatrix} \times \begin{pmatrix} \tilde{h}^{(t)} \\ \text{candidate} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix}$$

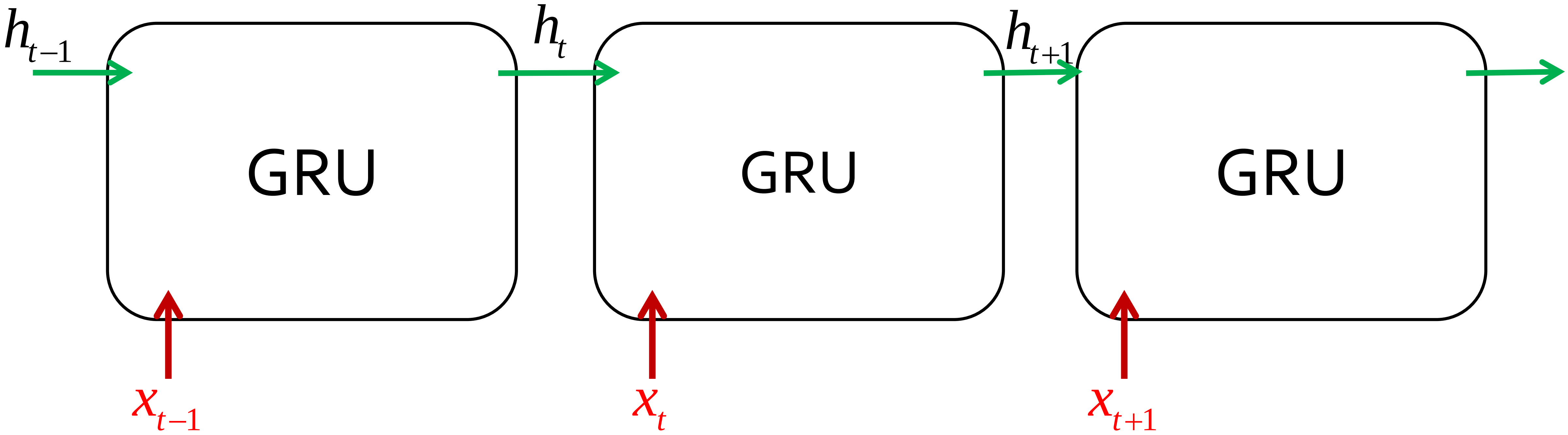
$$s^{(t)} = \sigma_f \left[\begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix} \right]$$

$$\begin{pmatrix} \tilde{h}^{(t)} \\ \text{candidate} \\ \text{hid. state} \\ \text{at time } t \end{pmatrix} = \tanh \left[\begin{pmatrix} r^{(t)} \\ \text{read gate} \\ \text{at time } t \end{pmatrix} \otimes \begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix} \right]$$

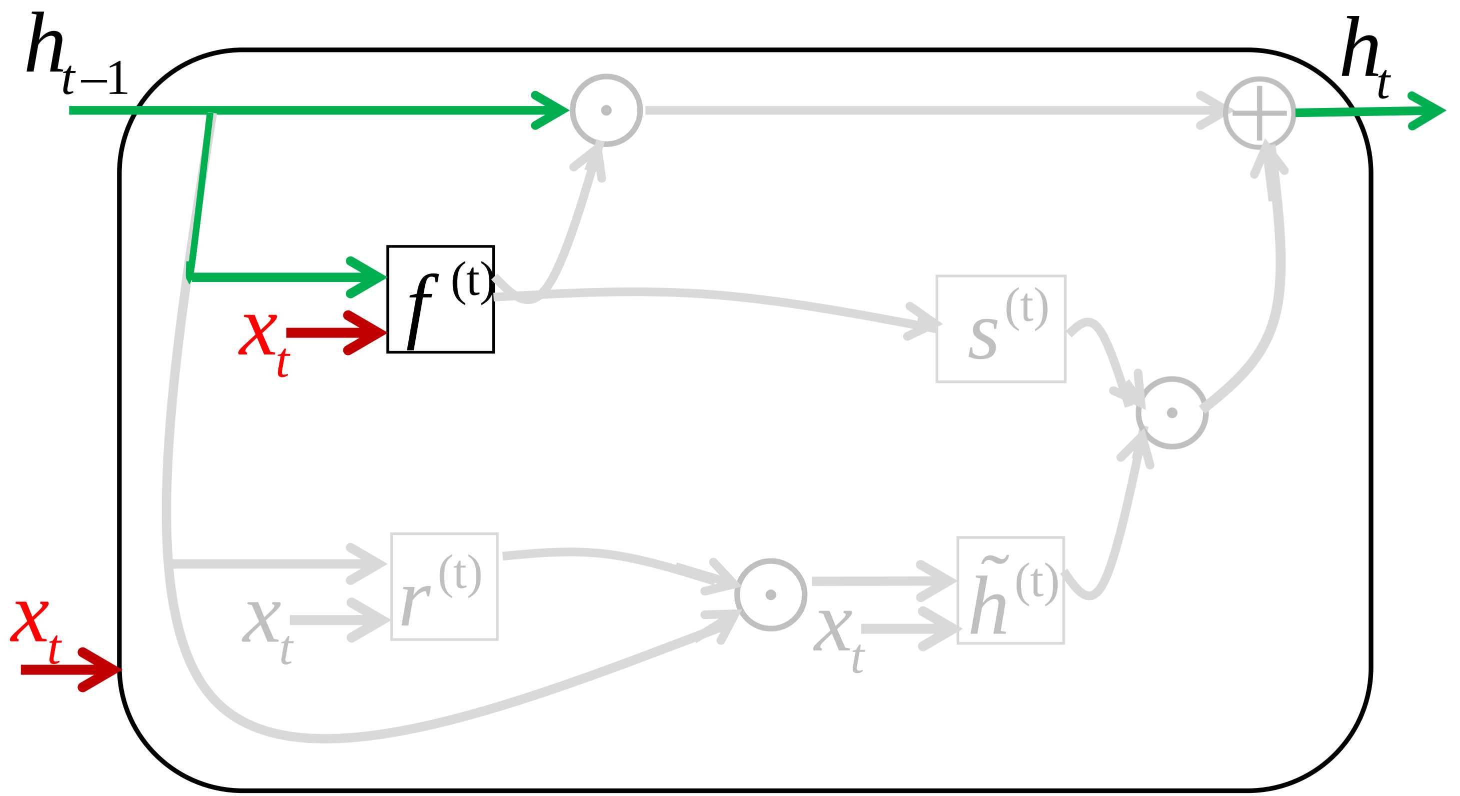
$$r^{(t)} = \sigma_f \left[\begin{pmatrix} h^{(t-1)} \\ \text{hid. state} \\ \text{at time } t-1 \end{pmatrix}, \begin{pmatrix} x^{(t)} \\ \text{input} \\ \text{at time } t \end{pmatrix} \right]$$

Gated Recurrent Unit-(GRU)

[Cho et al., 2014]

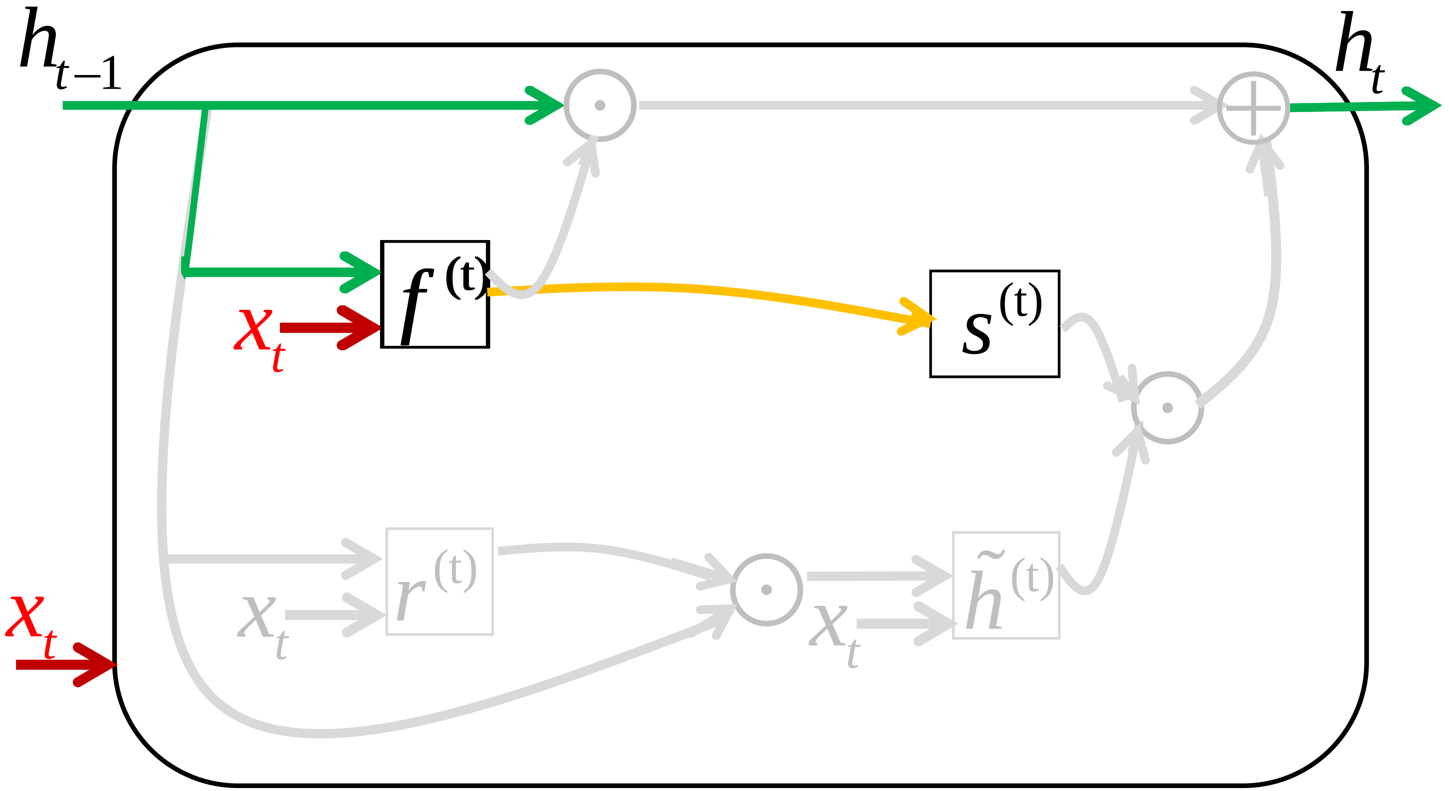


GRU: Forget gate



$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

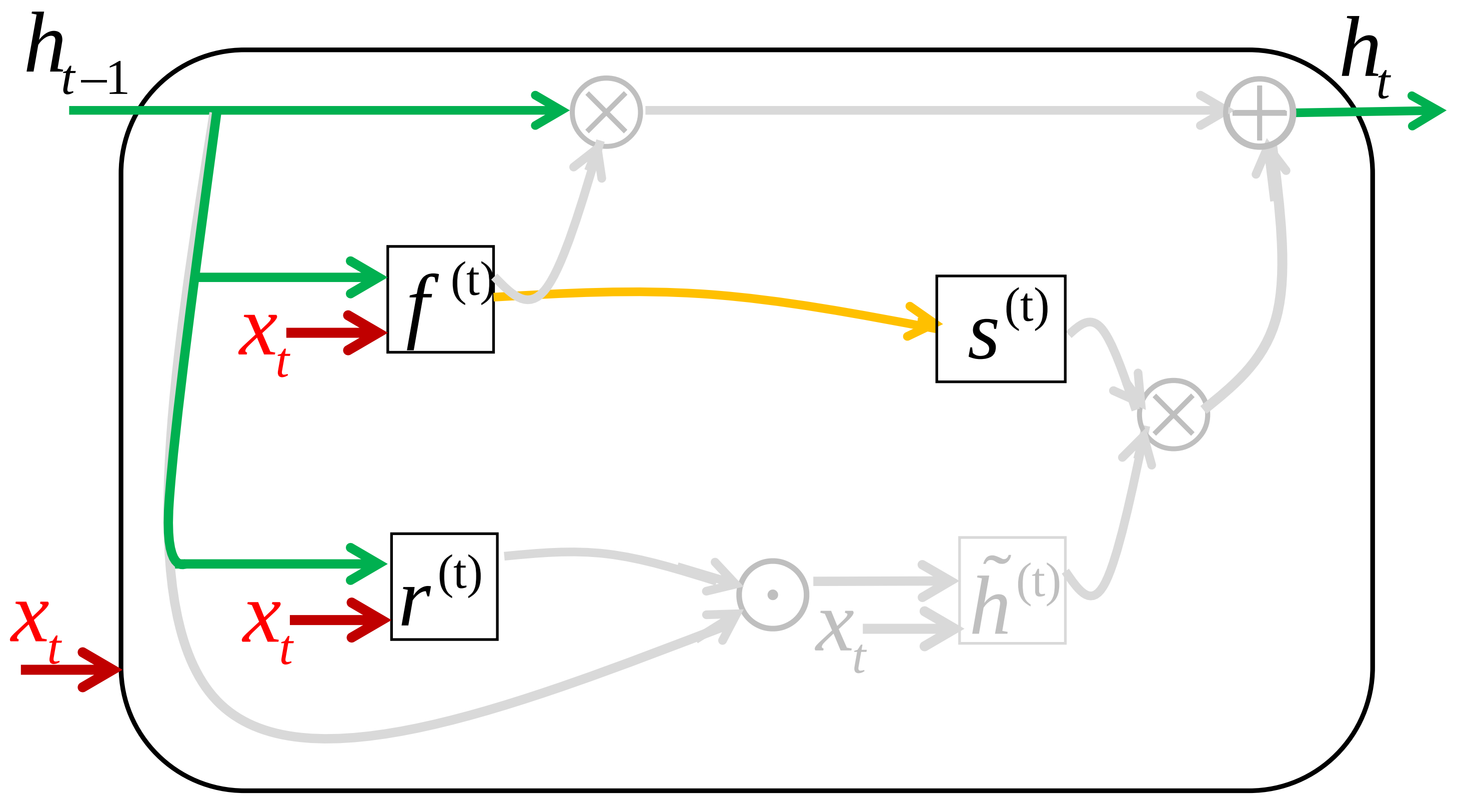
GRU: Forget gate and Store gate



$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$s_t = 1 - f_t$$

GRU: Reading gate

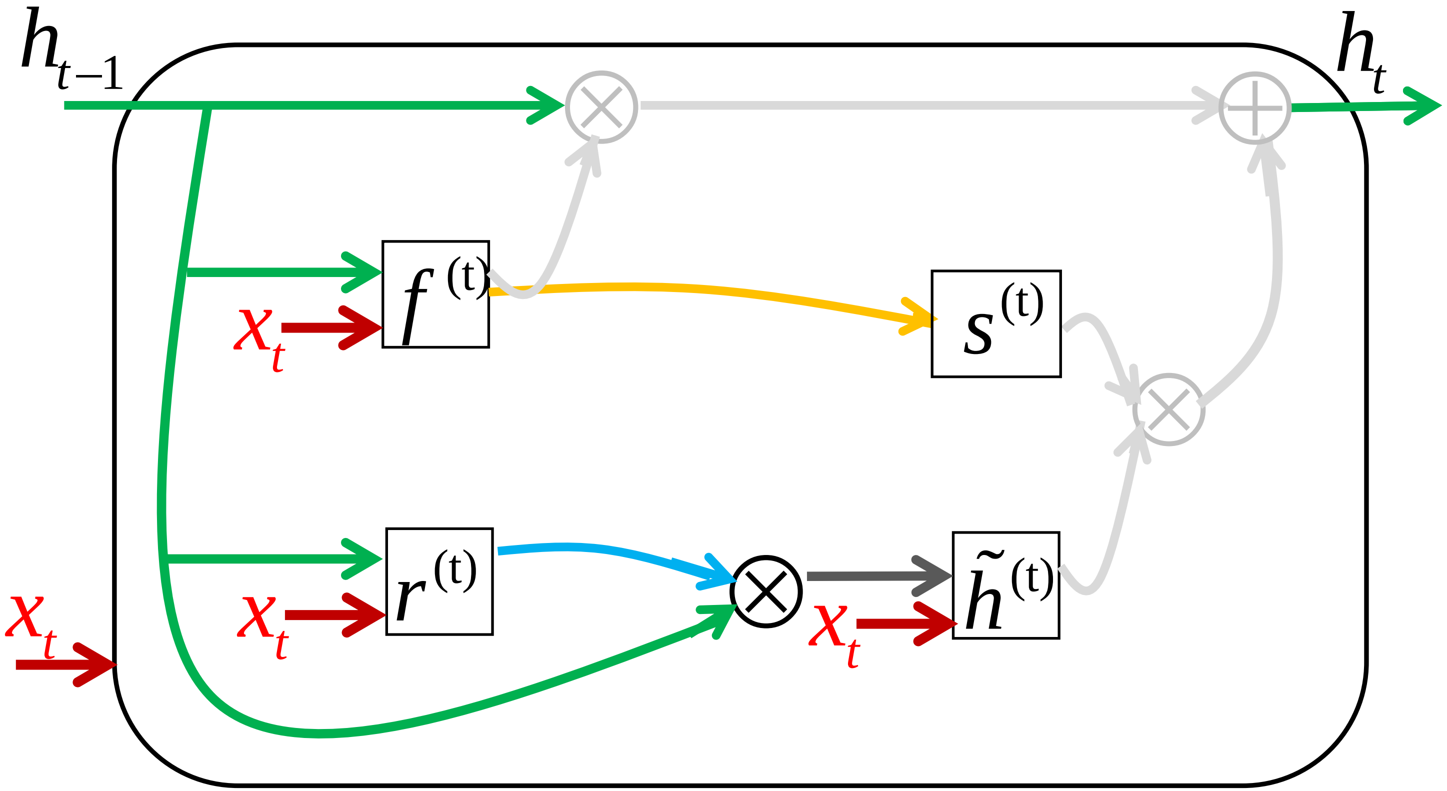


$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$s_t = 1 - f_t$$

$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$$

GRU: Candidate Hidden State



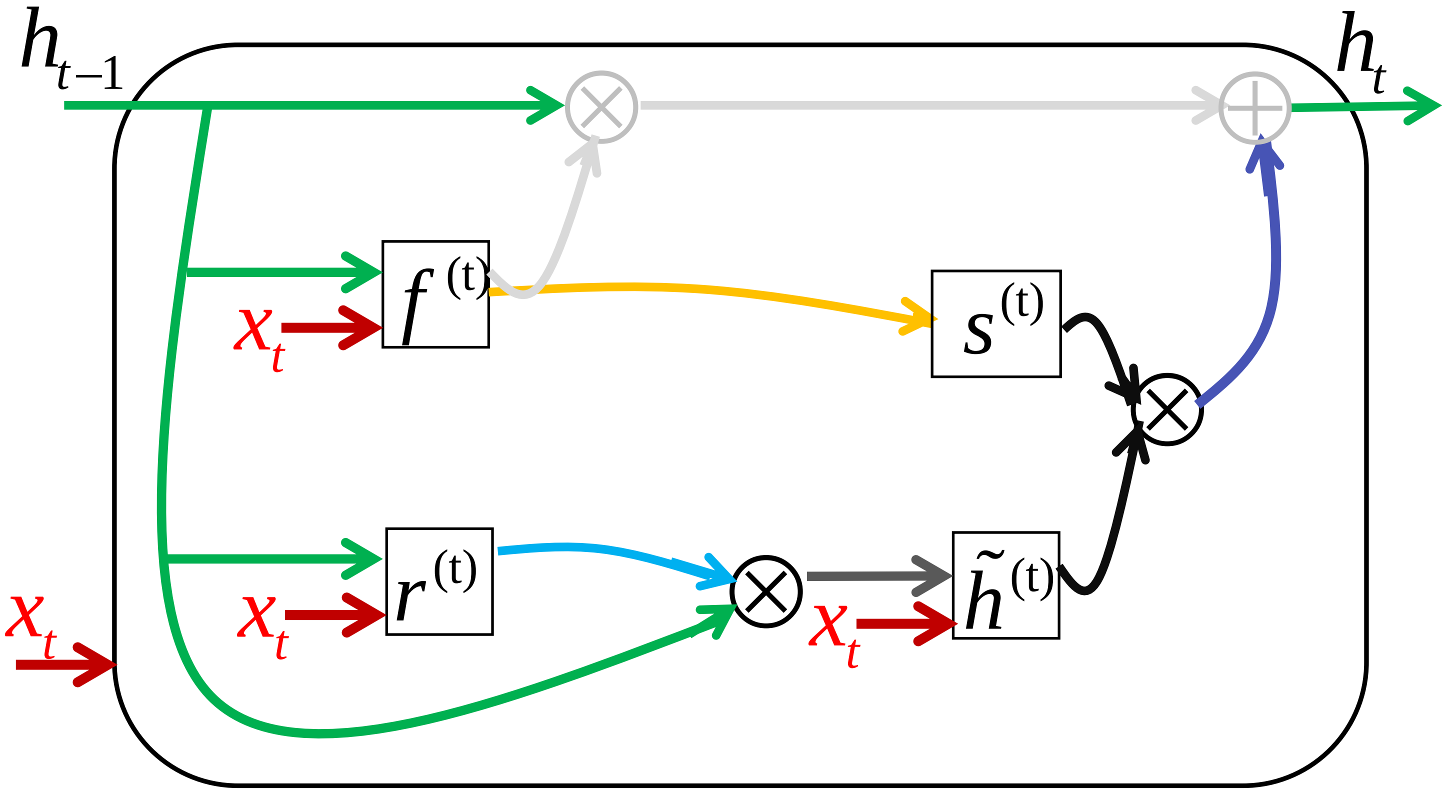
$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$s_t = 1 - f_t$$

$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$$

$$\tilde{h}_t = \tanh\left(W(r_t \otimes h_{t-1}) + U x_t + b\right)$$

GRU: Hidden State



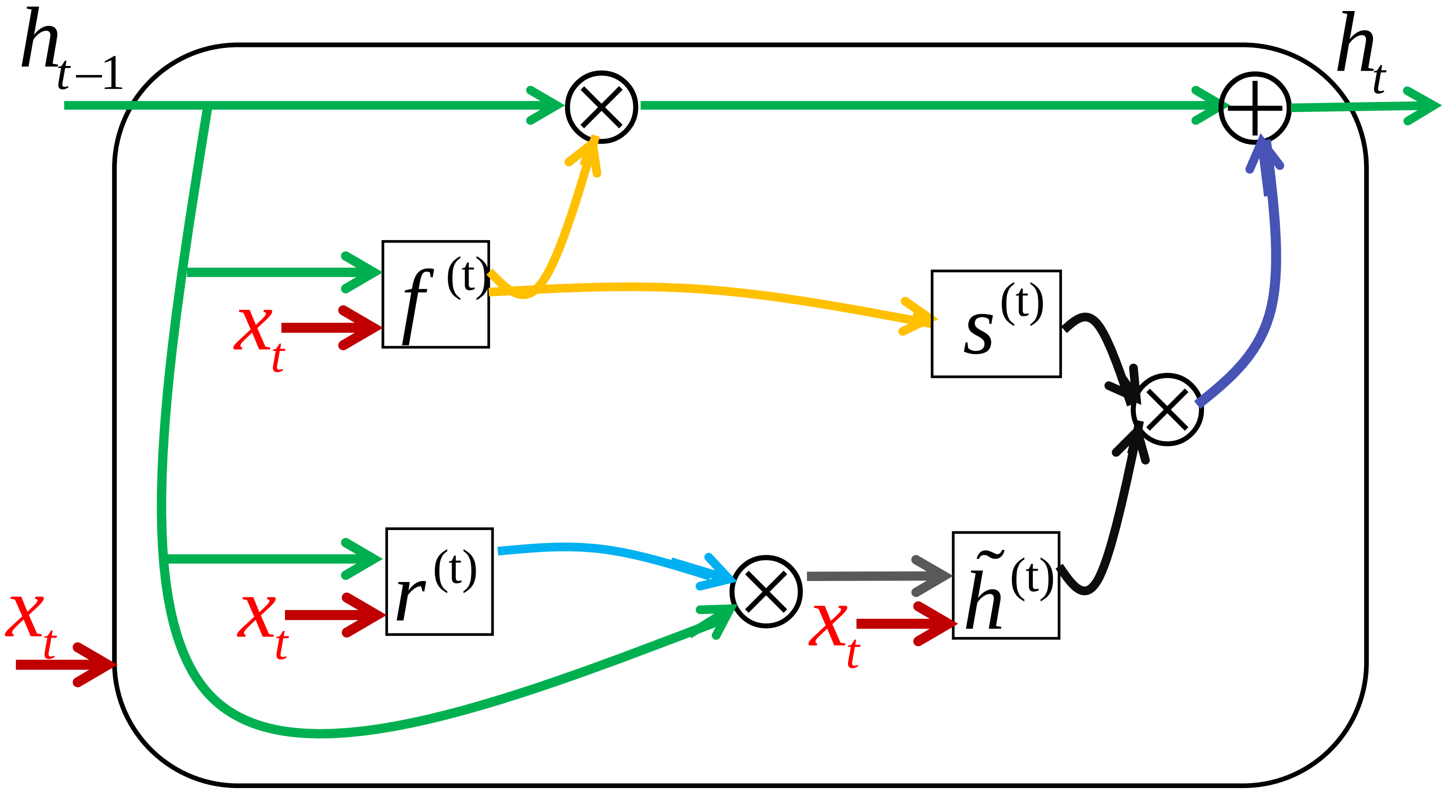
$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

$$s_t = 1 - f_t$$

$$r_t = \sigma(U_r x_t + W_r h_{t-1} + b_r)$$

$$\tilde{h}_t = \tanh\left(W(r_t \otimes h_{t-1}) + U x_t + b\right)$$

GRU: Memory Update



$$f_t = \sigma(\mathrm{U}_f \mathrm{x}_t + \mathrm{W}_f \mathrm{h}_{t-1} + \mathrm{b}_f)$$

$$\mathrm{s}_t = 1 - f_t$$

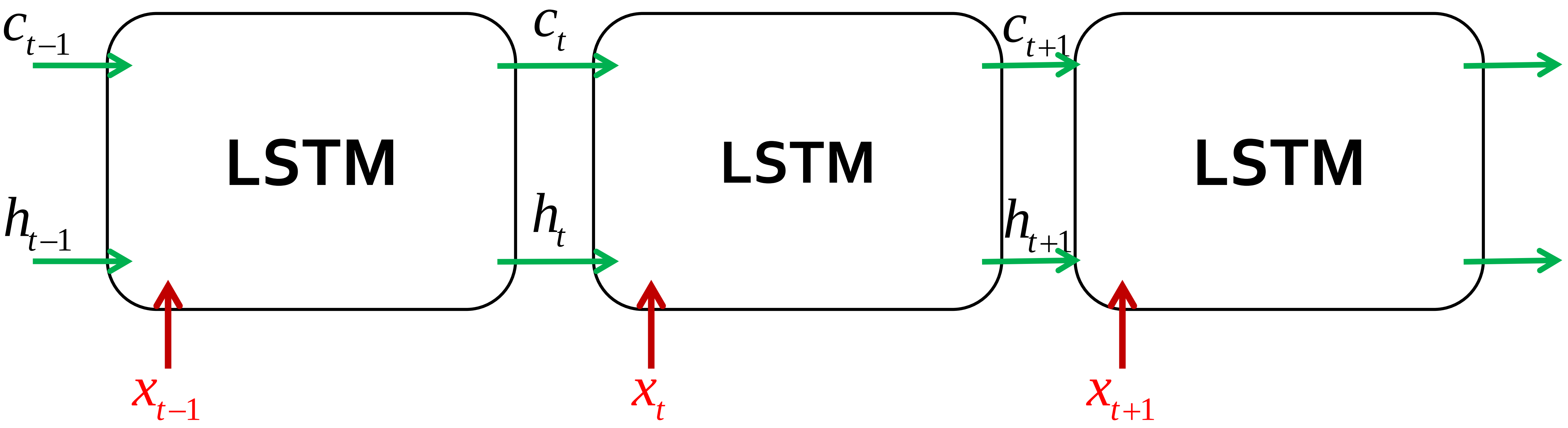
$$r_t = \sigma(\mathrm{U}_r \mathrm{x}_t + \mathrm{W}_r \mathrm{h}_{t-1} + \mathrm{b}_r)$$

$$\tilde{h}_t = \tanh\left(\mathrm{W}(\mathrm{r}_t \otimes \mathrm{h}_{t-1}) + Ux_t + b\right)$$

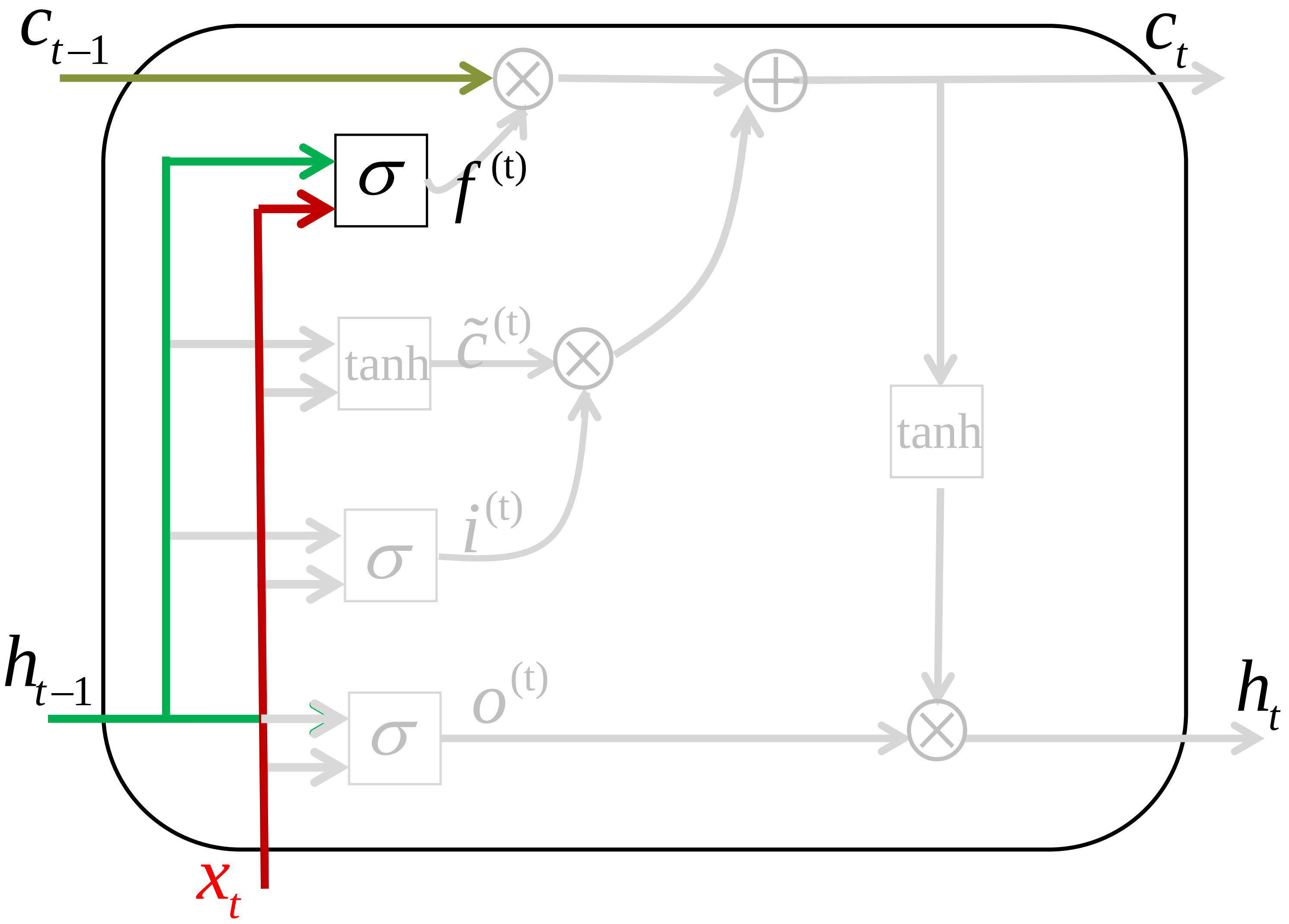
$$h_t = \left(f_t \otimes \mathrm{h}_{t-1}\right) + \left(\mathrm{s}_t \otimes \tilde{h}_t\right)$$

[Sepp Hochreiter and Jürgen Schmidhuber, 1997]

Long Short-Term Memory (LSTM)

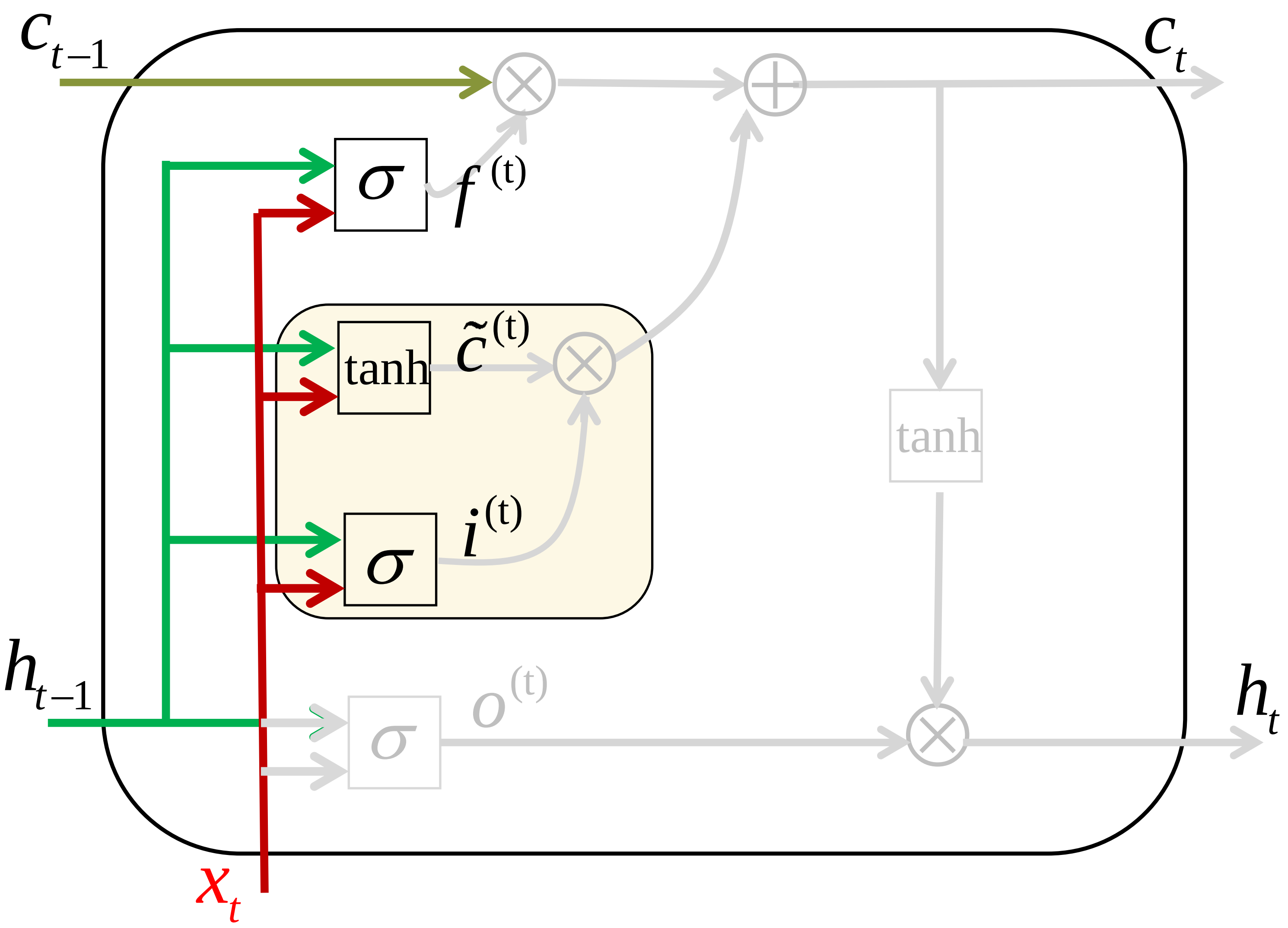


LSTM- Forget Gate



$$f^{(t)} = \sigma(U_f x^{(t)} + W_f h^{(t-1)} + b_f)$$

LSTM- Store Gate

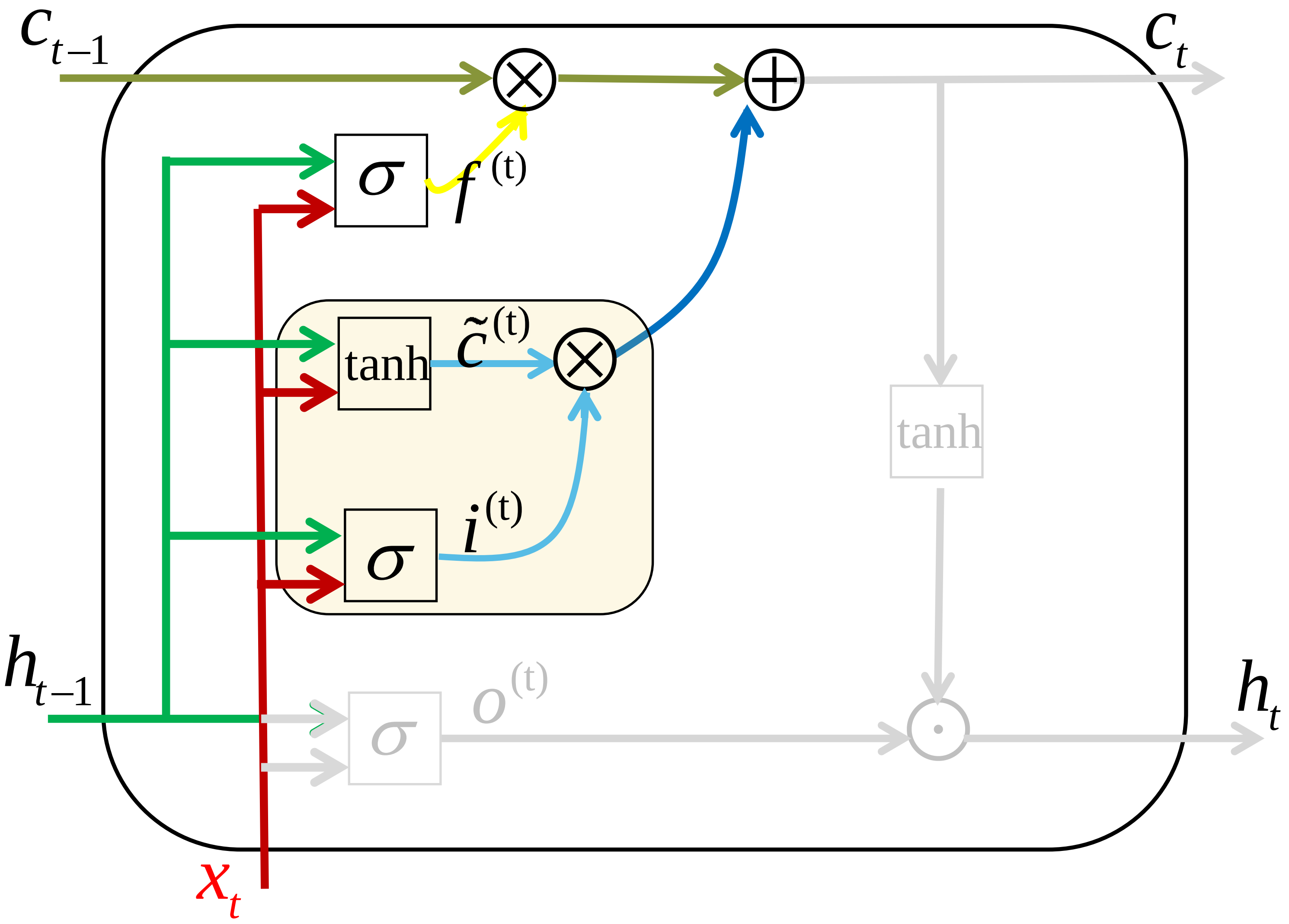


$$f^{(t)} = \sigma(U_f x^{(t)} + W_f h^{(t-1)} + b_f)$$

$$i^{(t)} = \sigma(U_i x^{(t)} + W_i h^{(t-1)} + b_i)$$

$$\tilde{c}^{(t)} = \tanh\left(W h^{(t-1)} + U x^{(t)} + b\right)$$

LSTM- Memory Update



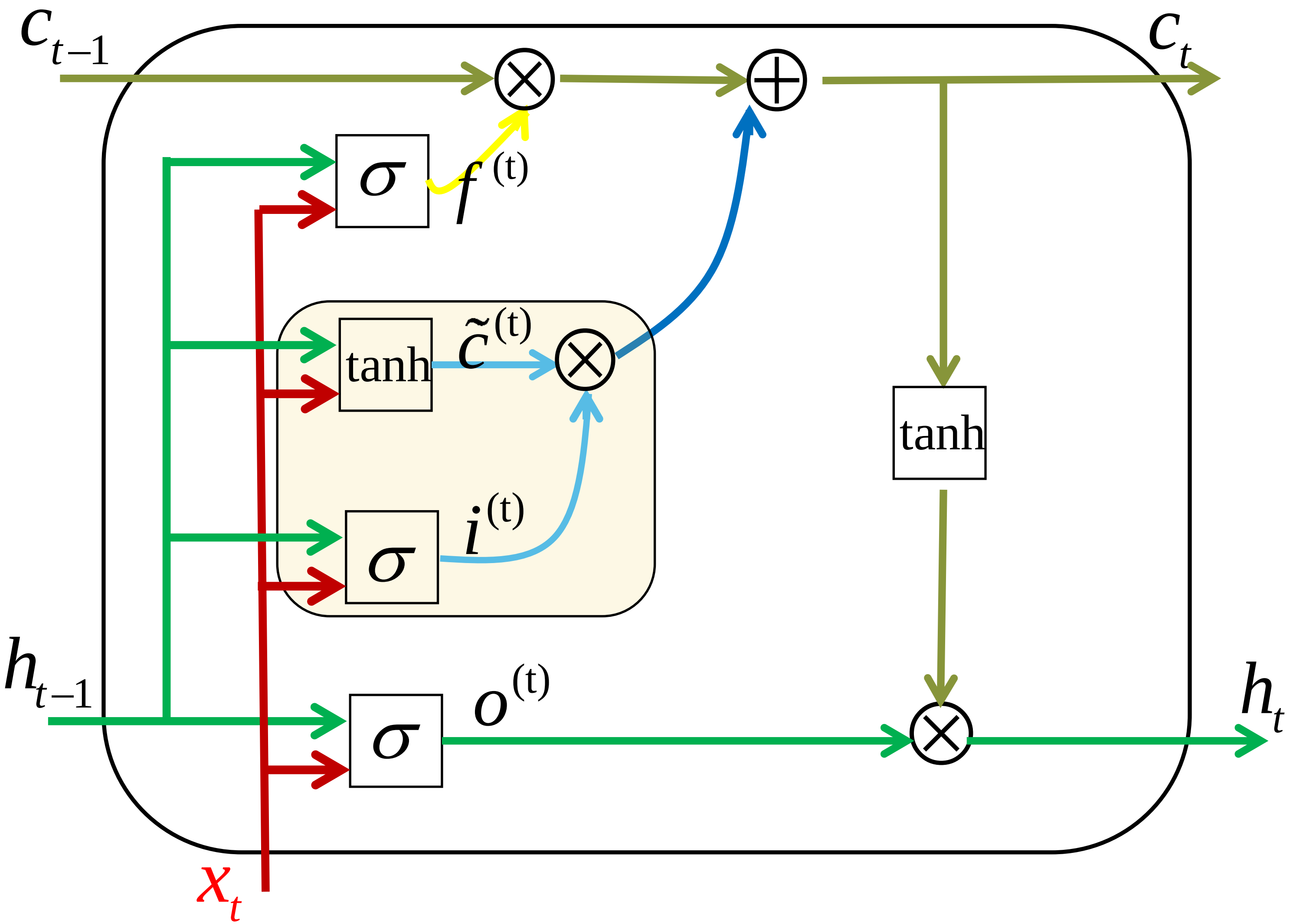
$$f^{(t)} = \sigma(\mathrm{U}_f \mathrm{x}^{(t)} + \mathrm{W}_f \mathrm{h}^{(t-1)} + \mathrm{b}_f)$$

$$i^{(t)} = \sigma(\mathrm{U}_i \mathrm{x}^{(t)} + \mathrm{W}_i \mathrm{h}^{(t-1)} + \mathrm{b}_i)$$

$$\tilde{c}^{(t)} = \tanh\left(\mathrm{W}\mathrm{h}^{(t-1)} + U\,\mathrm{x}^{(t)} + b\right)$$

$$c^{(t)} = \left(f^{(t)} \circ \tilde{c}^{(t-1)}\right) + \left(i \circ \tilde{c}^{(t)}\right)$$

LSTM- Output Gate



$$f^{(t)} = \sigma(U_f x^{(t)} + W_f h^{(t-1)} + b_f)$$

$$i^{(t)} = \sigma(U_i x^{(t)} + W_i h^{(t-1)} + b_i)$$

$$\tilde{c}^{(t)} = \tanh\left(W h^{(t-1)} + U x^{(t)} + b\right)$$

$$c^{(t)} = \left(f^{(t)} \circ \tilde{c}^{(t-1)}\right) + \left(i \circ \tilde{c}^{(t)}\right)$$

$$o^{(t)} = \sigma(U_o x^{(t)} + W_o h^{(t-1)} + b_o)$$

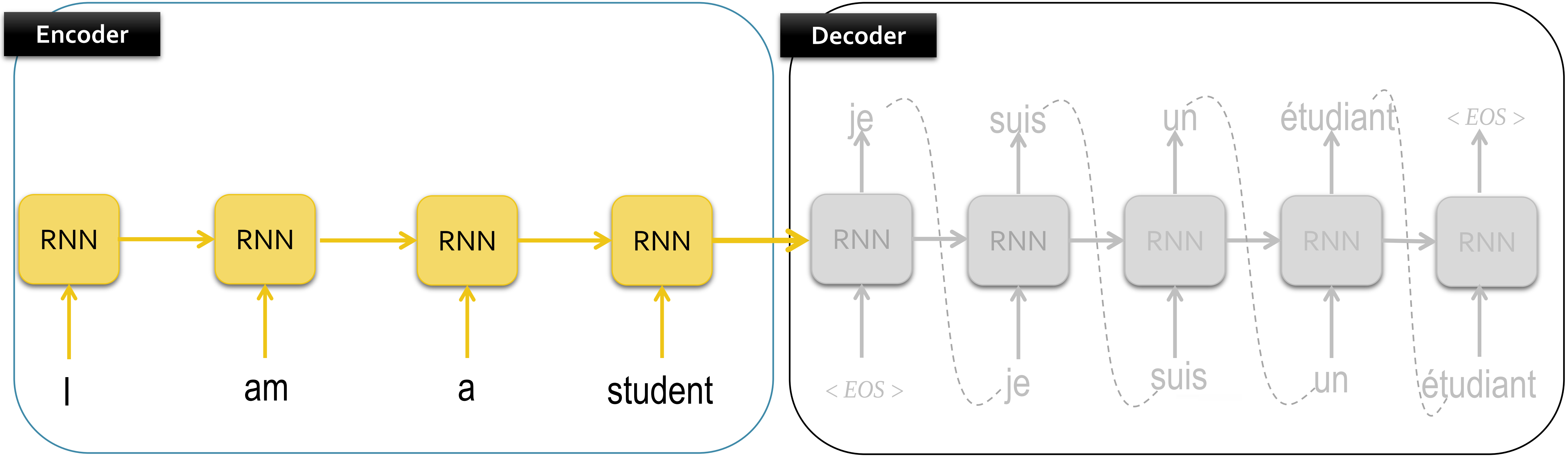
$$h^{(t)} = o^{(t)} \circ \tanh(c^{(t)})$$

$$y^{(t)} = h^{(t)}$$

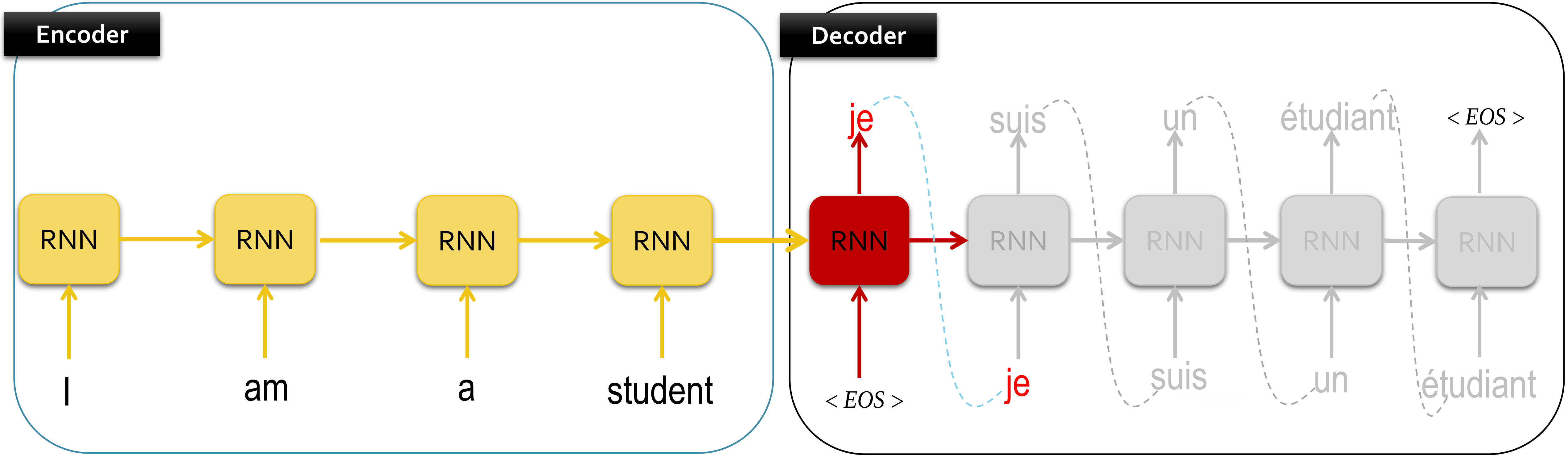
Questions?

Some examples of RNN

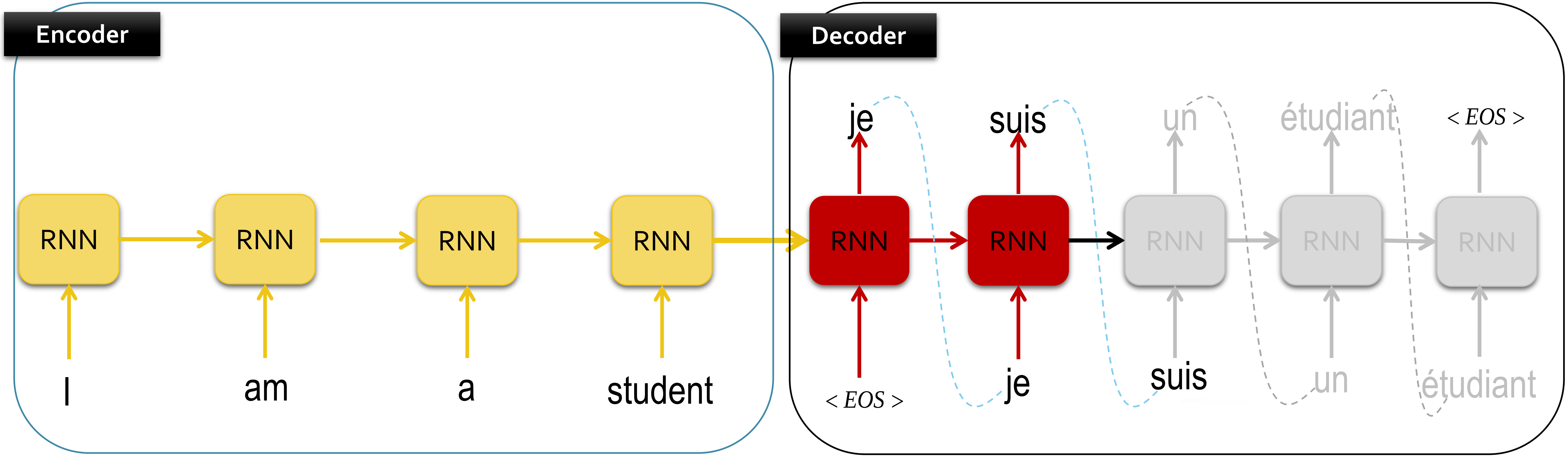
Sequence to Sequence



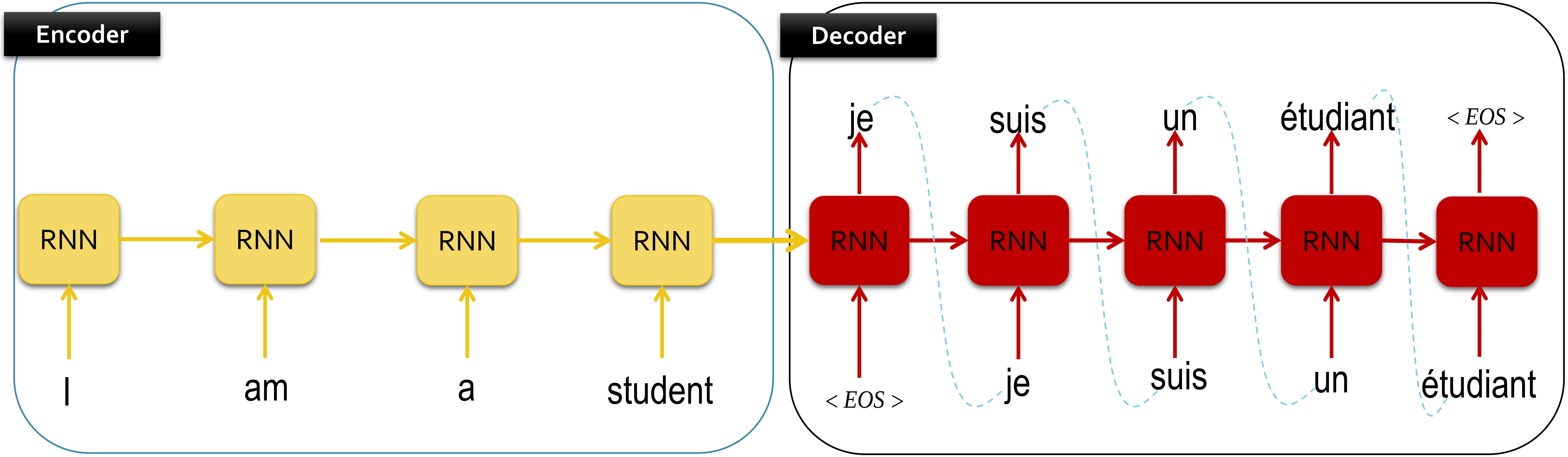
Sequence to Sequence



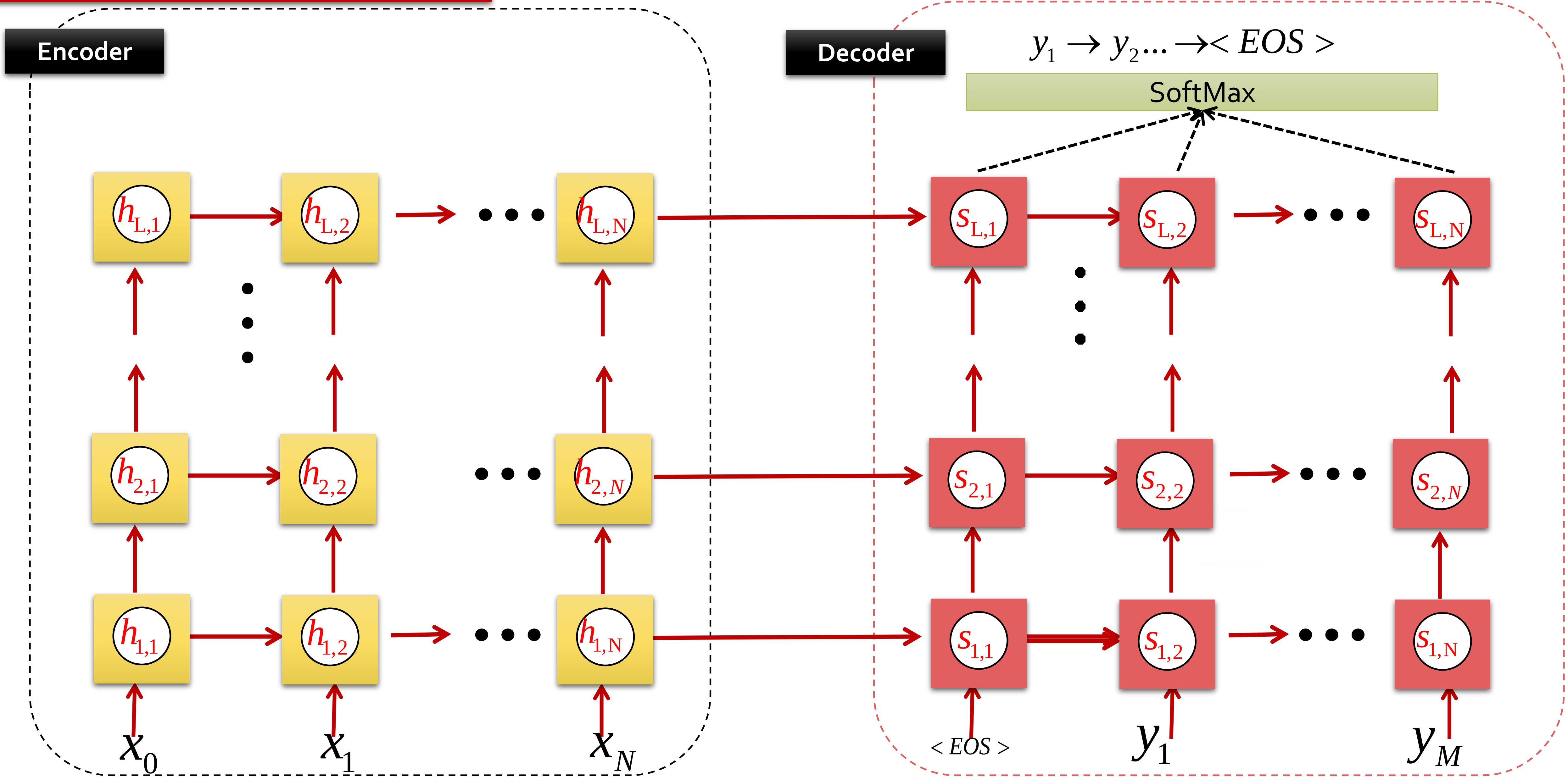
Sequence to Sequence



Sequence to Sequence

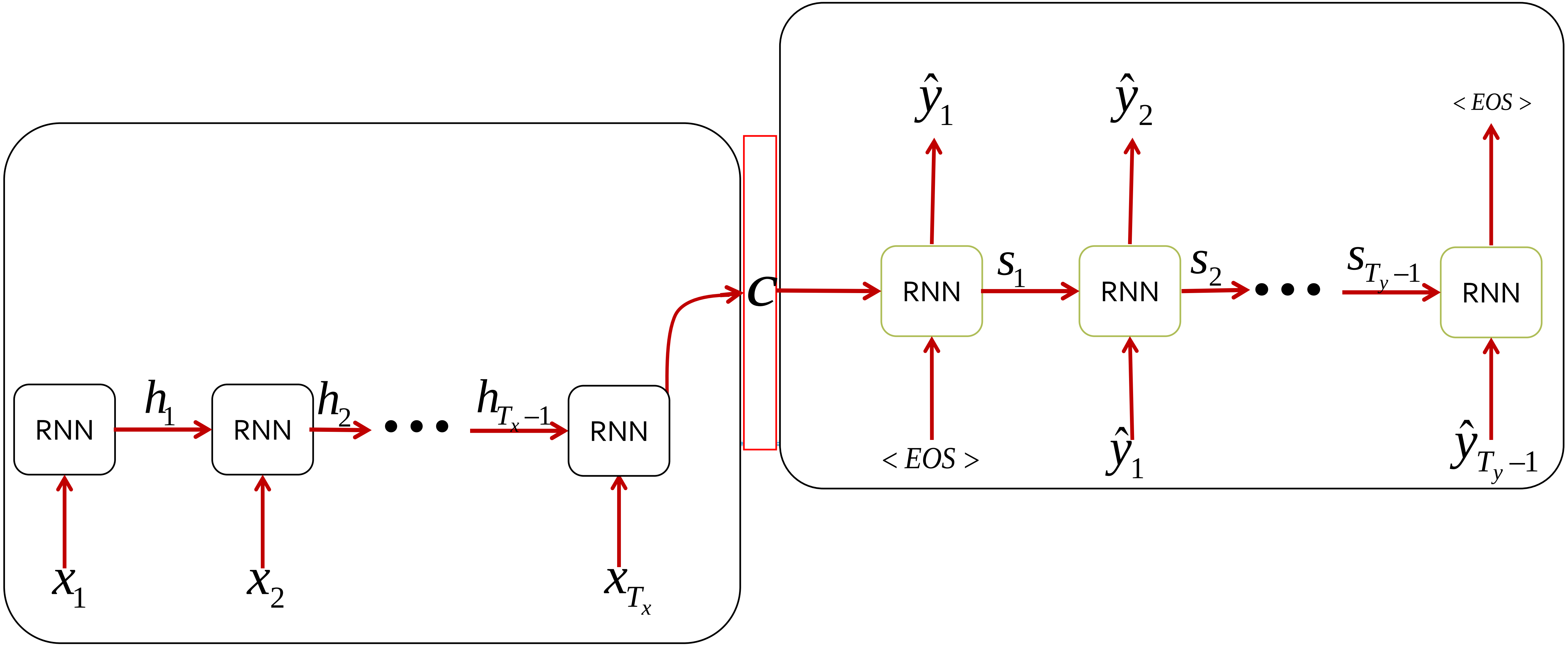


Deep Sequence to Sequence



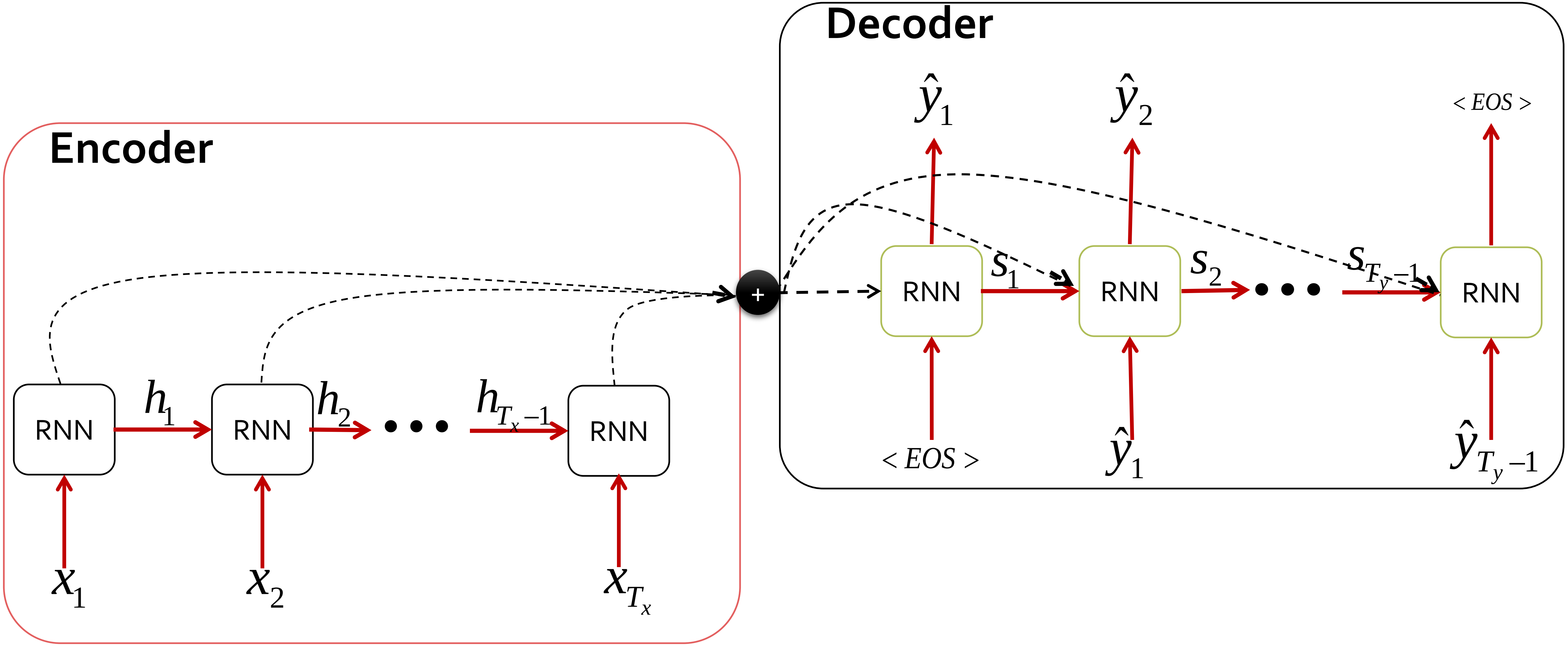
Classic Encoder-Decoder

- The last hidden state summarizes the entire input sentence into C.



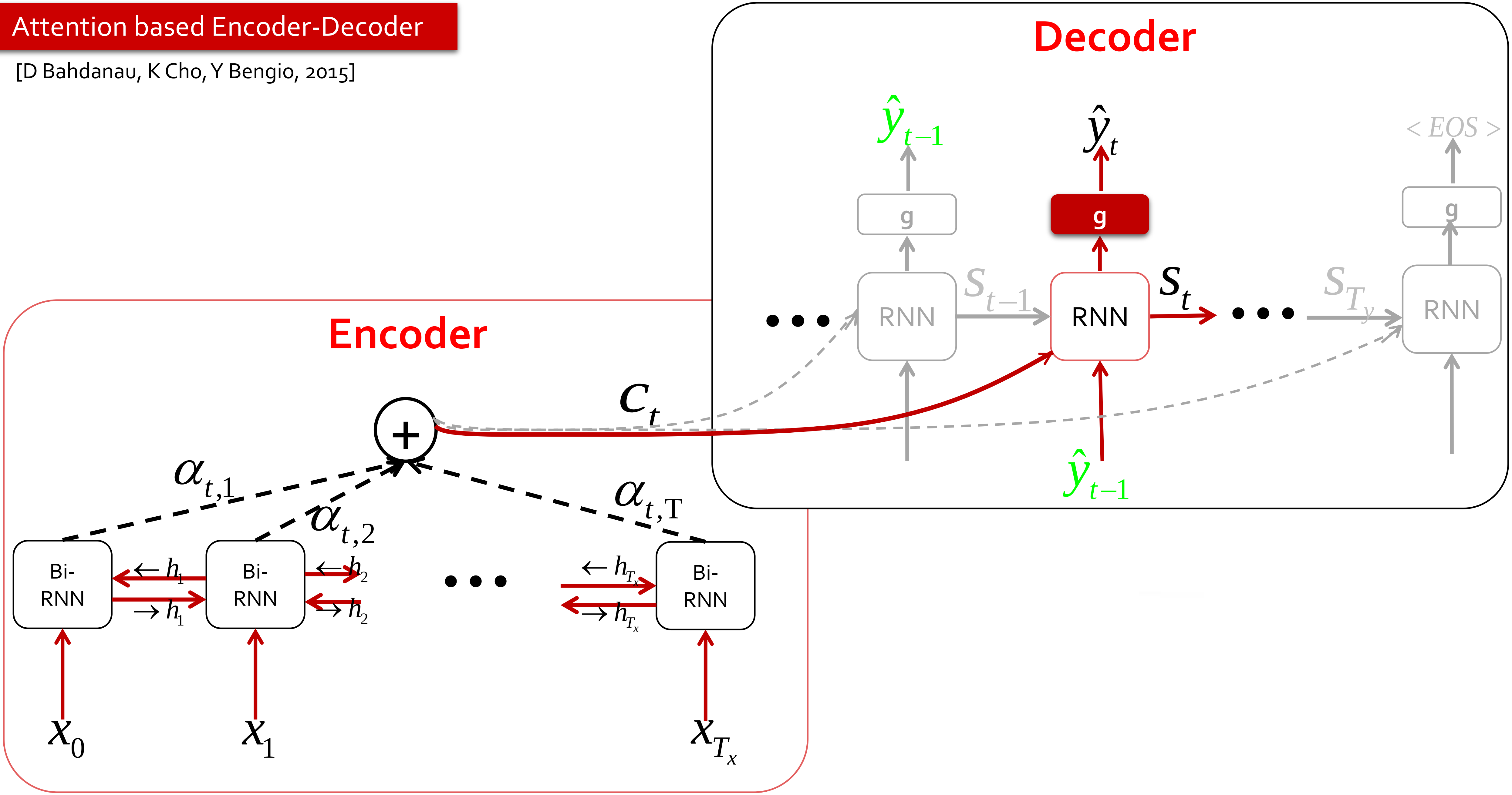
Attention based Encoder-Decoder

[D Bahdanau, K Cho, Y Bengio, 2015]



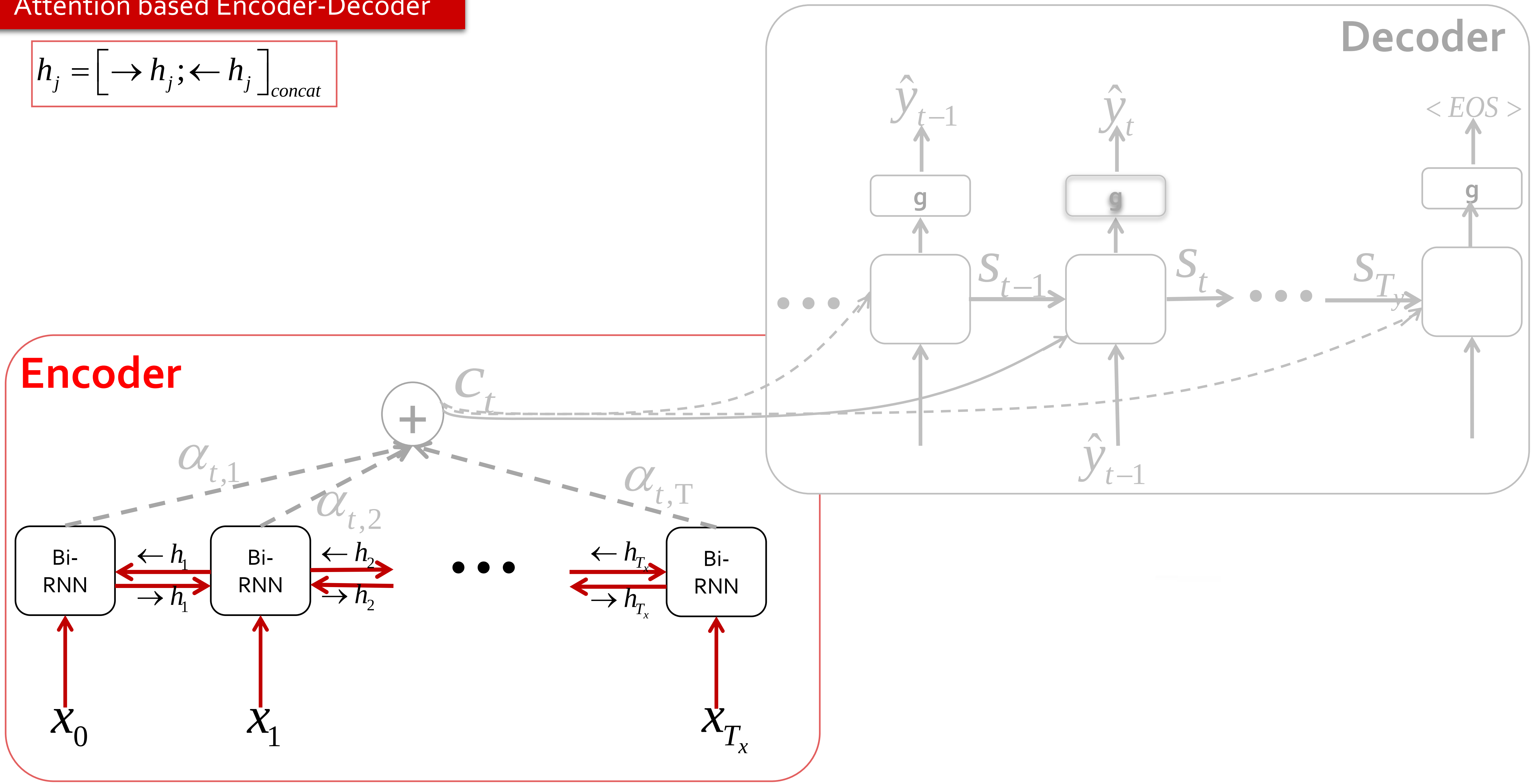
Attention based Encoder-Decoder

[D Bahdanau, K Cho, Y Bengio, 2015]



Attention based Encoder-Decoder

$$h_j = [\rightarrow h_j; \leftarrow h_j]_{concat}$$

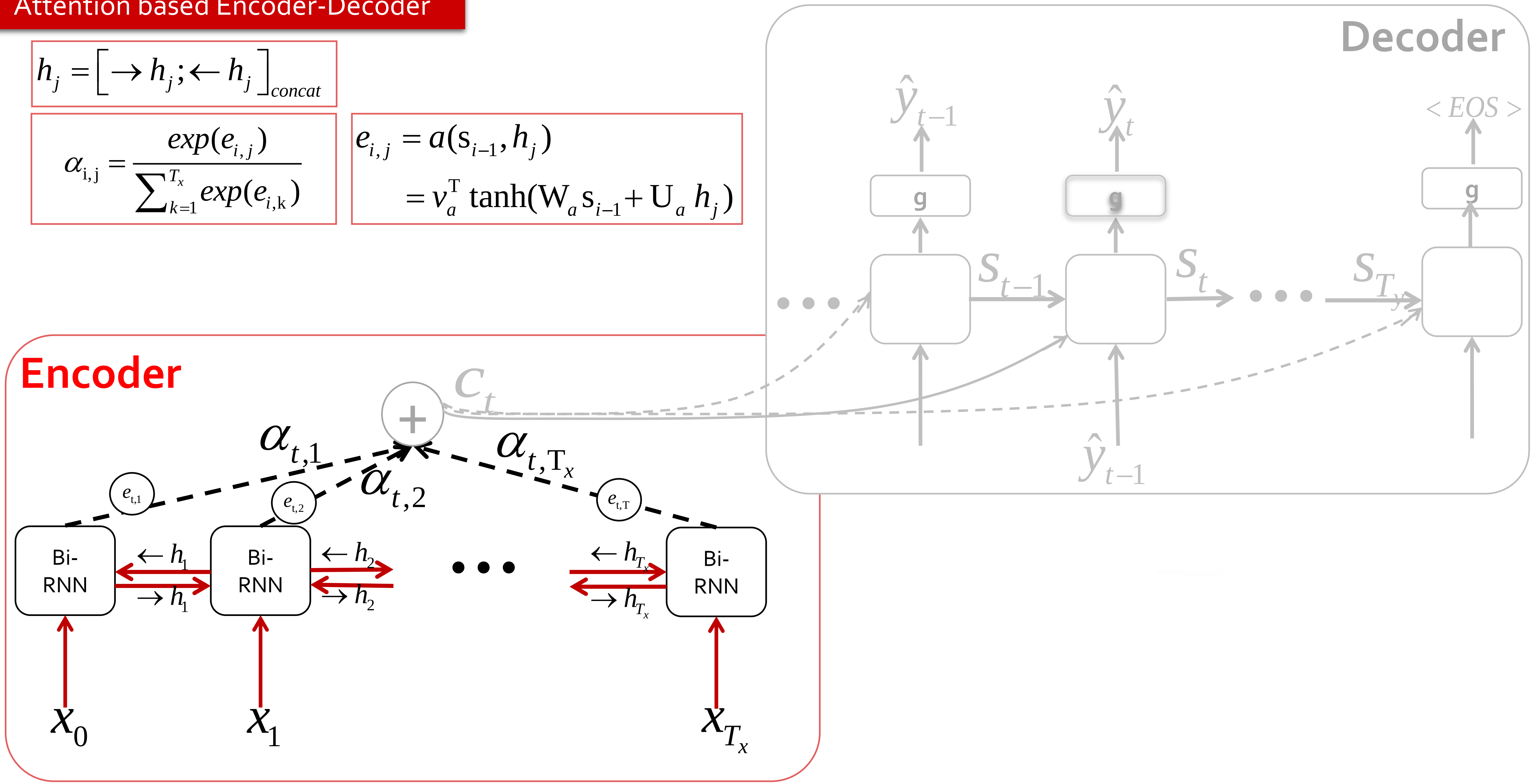


Attention based Encoder-Decoder

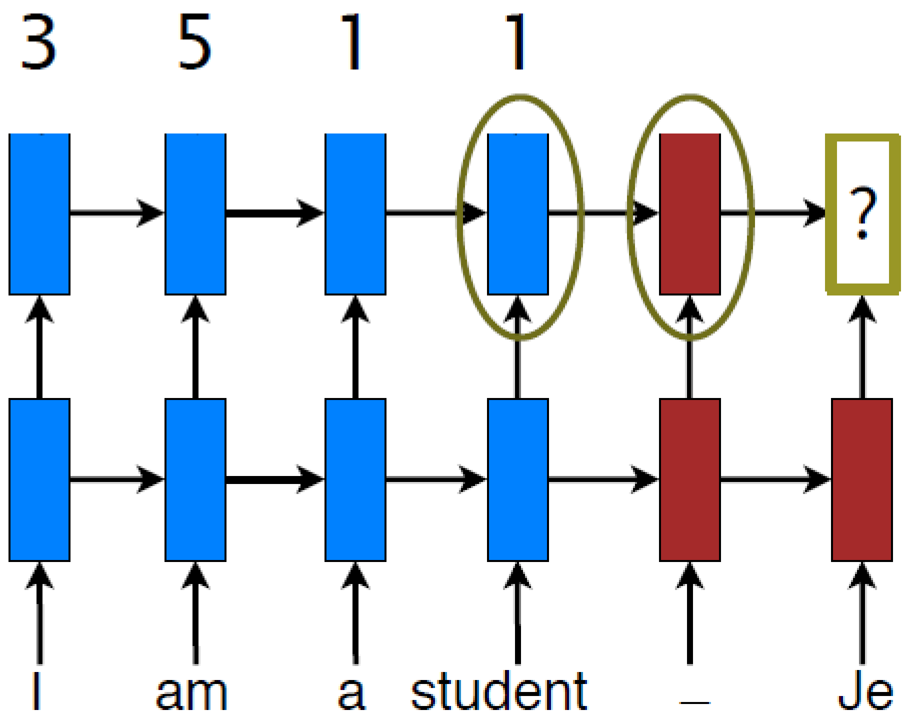
$$h_j = [\rightarrow h_j ; \leftarrow h_j]_{concat}$$

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{T_x} exp(e_{i,k})}$$

$$e_{i,j} = a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$



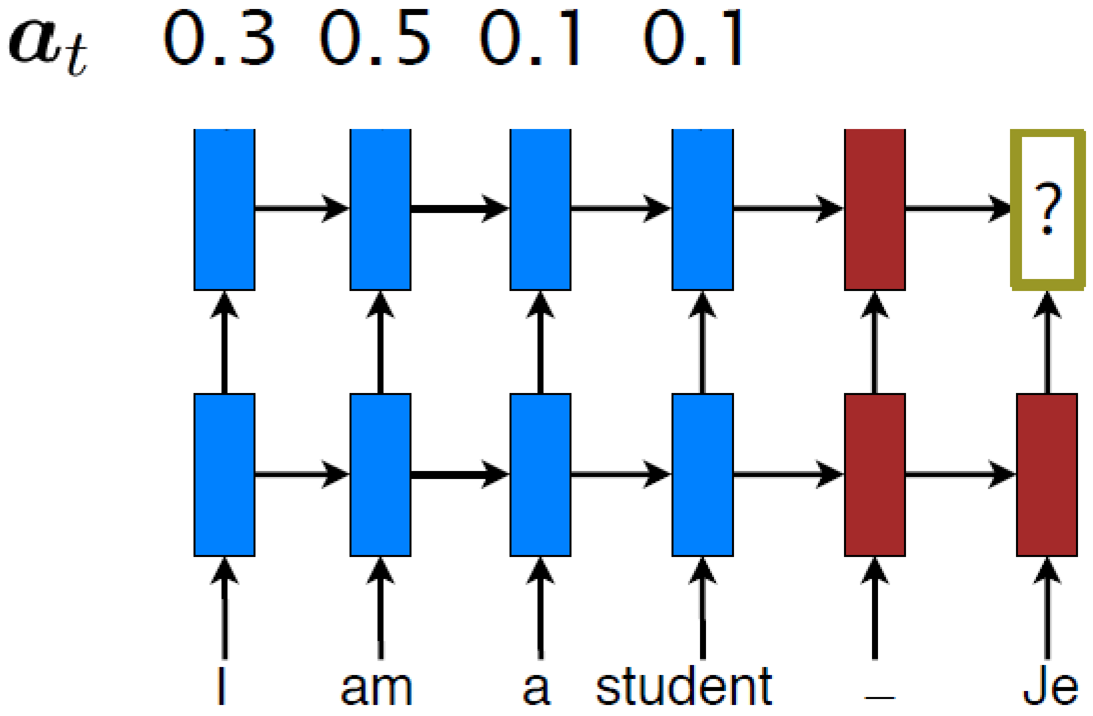
Attention based Encoder-Decoder-Computing the e



Slide: from NLP- Stanford (Christopher Manning and Richard Socher)

Attention based Encoder-Decoder- Normalization

$$\boldsymbol{a}_t(s) = \frac{e^{\mathrm{score}(s)}}{\sum_{s'} e^{\mathrm{score}(s')}}$$



Slide: from NLP- Stanford (Christopher Manning and Richard Socher)

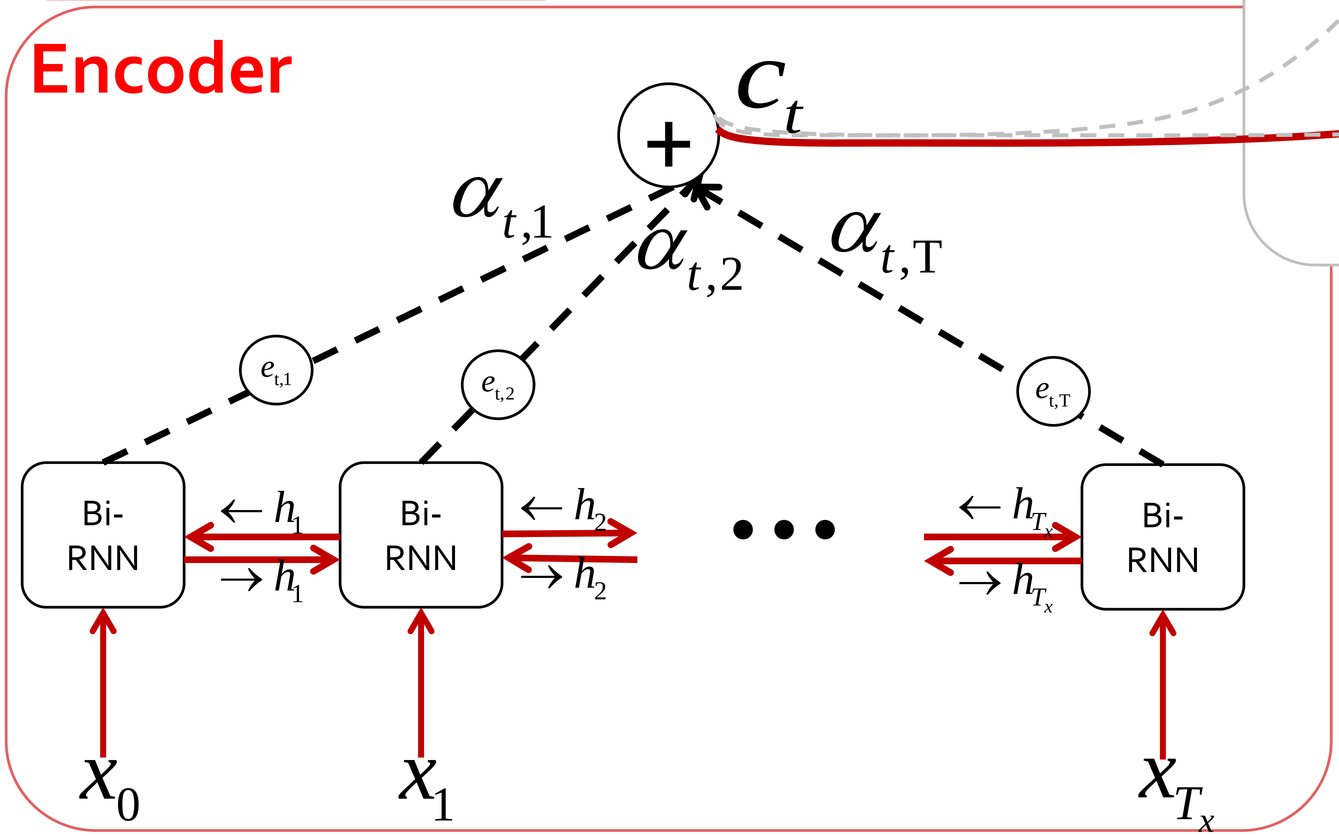
Attention based Encoder-Decoder

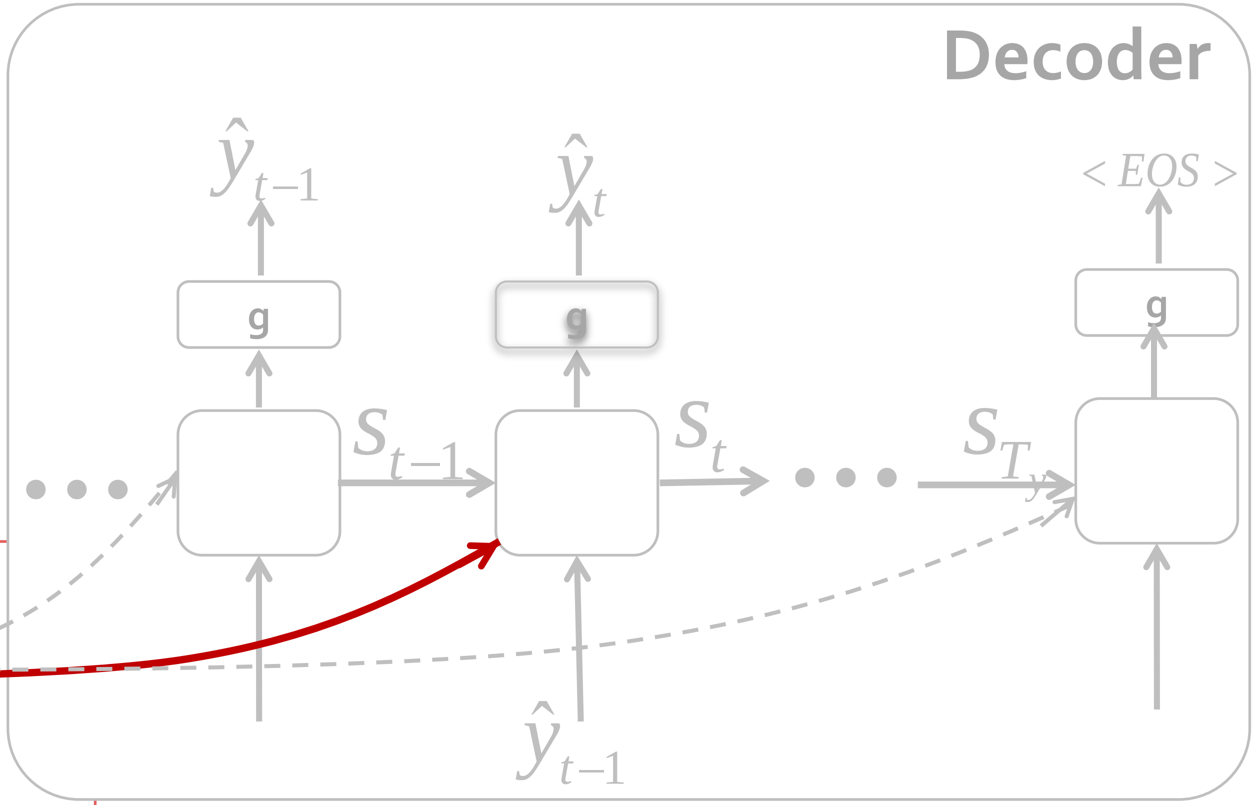
$$h_j = [\rightarrow h_j ; \leftarrow h_j]_{concat}$$

$$\alpha_{i,j} = \frac{exp(e_{i,j})}{\sum_{k=1}^{T_x} exp(e_{i,k})}$$

$$e_{i,j} = a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

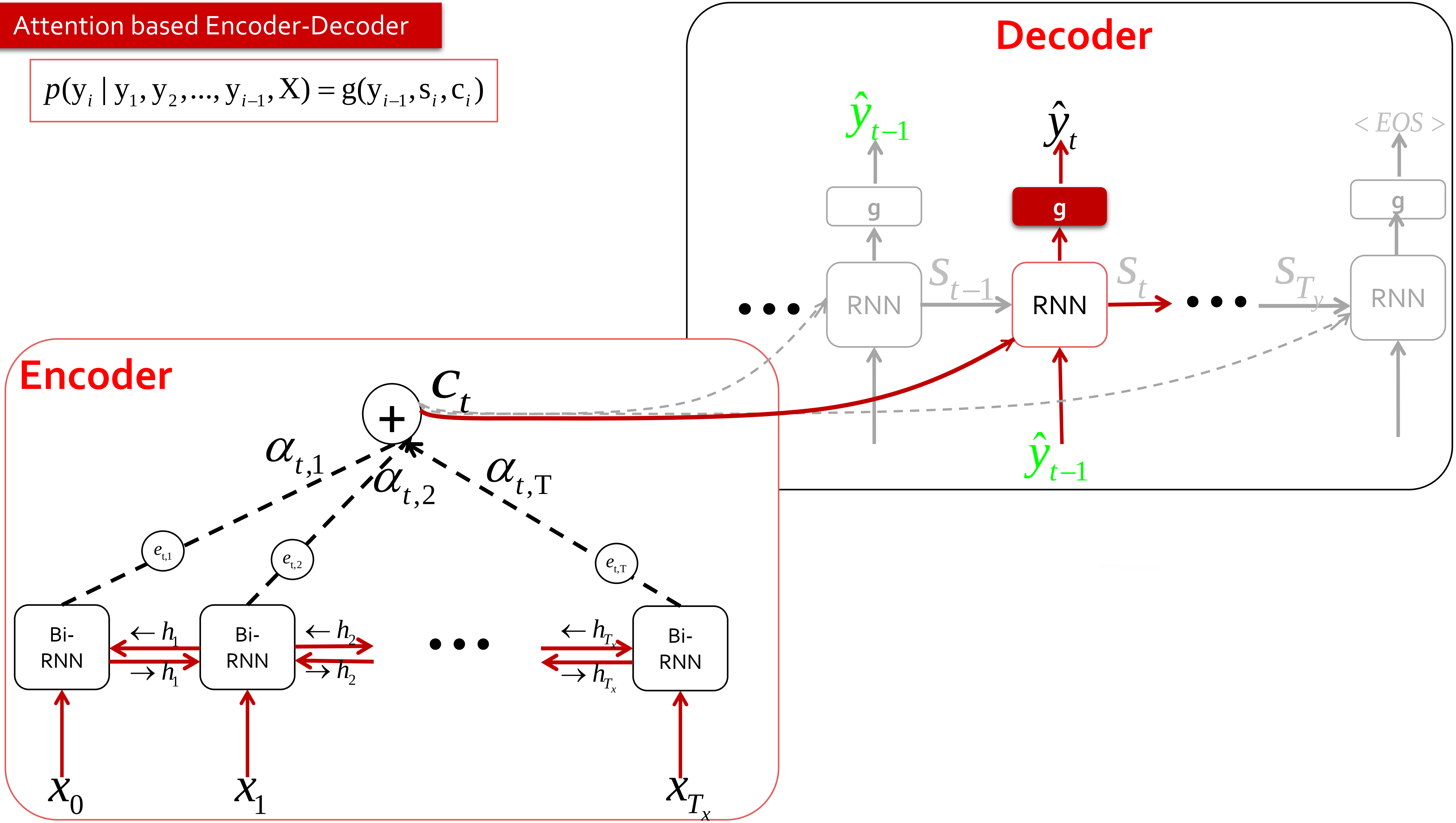
$$c_i = \sum_{k=1}^{T_x} \alpha_{i,j} h_j$$





Attention based Encoder-Decoder

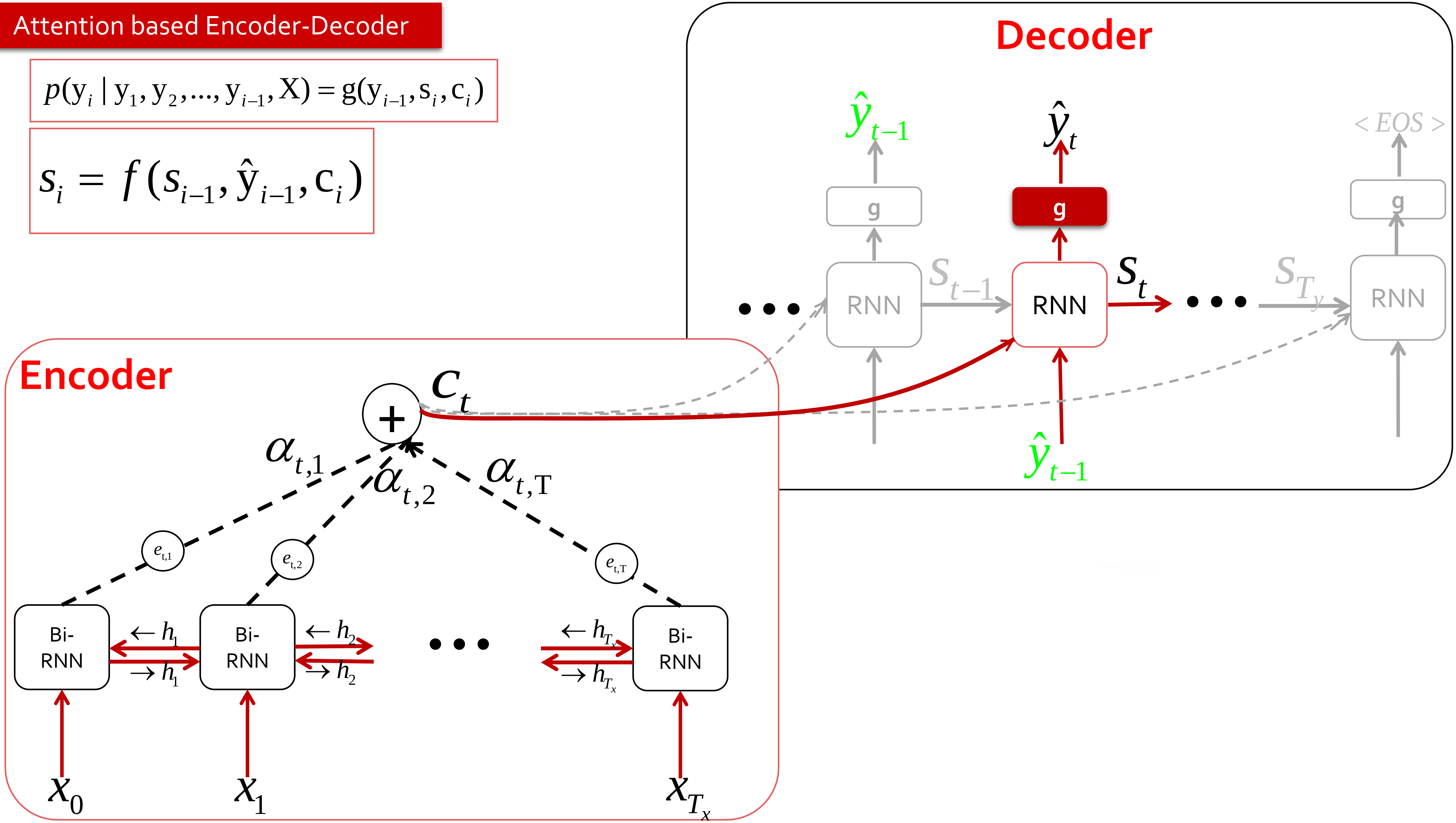
$$p(y_i \mid y_1, y_2, \ldots, y_{i-1}, X) = g(y_{i-1}, s_i, c_i)$$



Attention based Encoder-Decoder

$$p(y_i \mid y_1, y_2, \ldots, y_{i-1}, X) = g(y_{i-1}, s_i, c_i)$$

$$s_i = f(s_{i-1}, \hat{y}_{i-1}, c_i)$$



Thank **You!**

Questions?