

Section 15. Recurrent Neural Networks (RNN)

목차

- 섹션 14. Convolution Neural Network (CNN)
- **섹션 15. Recurrent Neural Network (RNN)**
- 섹션 16. Attention과 Transformer

Objective

학습 목표

- RNN의 구성에 대한 이해
- RNN의 작동원리에 대한 이해
- RNN의 Backpropagation
- RNN의 Gradient Vanishing 문제
- LSTM의 구성
- GRU의 구성
- RNN 계열의 모델이 가진 한계 파악

15-1. 시계열 데이터 (Time series data)

Recurrent Neural Networks

Introduction

- 시계열 데이터 (Sequential Data)는 어떻게 모델링 할 수 있을까?
- 시계열 데이터의 특징:
 - Order matters (순서)
 - Variable Length (배열의 길이가 변동적이다.)
 - Context (맥락)

Recurrent Neural Networks

Problem setup (문제 정의)

Studying Deep Learning is really 

- 어떻게 하면 문장에서 다음에 올 단어를 예측할 수 있을까?
- 즉, 에 들어갈 단어를 맞추도록 어떻게 학습할 수 있을까?

Recurrent Neural Networks

종류

- Recurrent Neural Network (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Network (GRU)

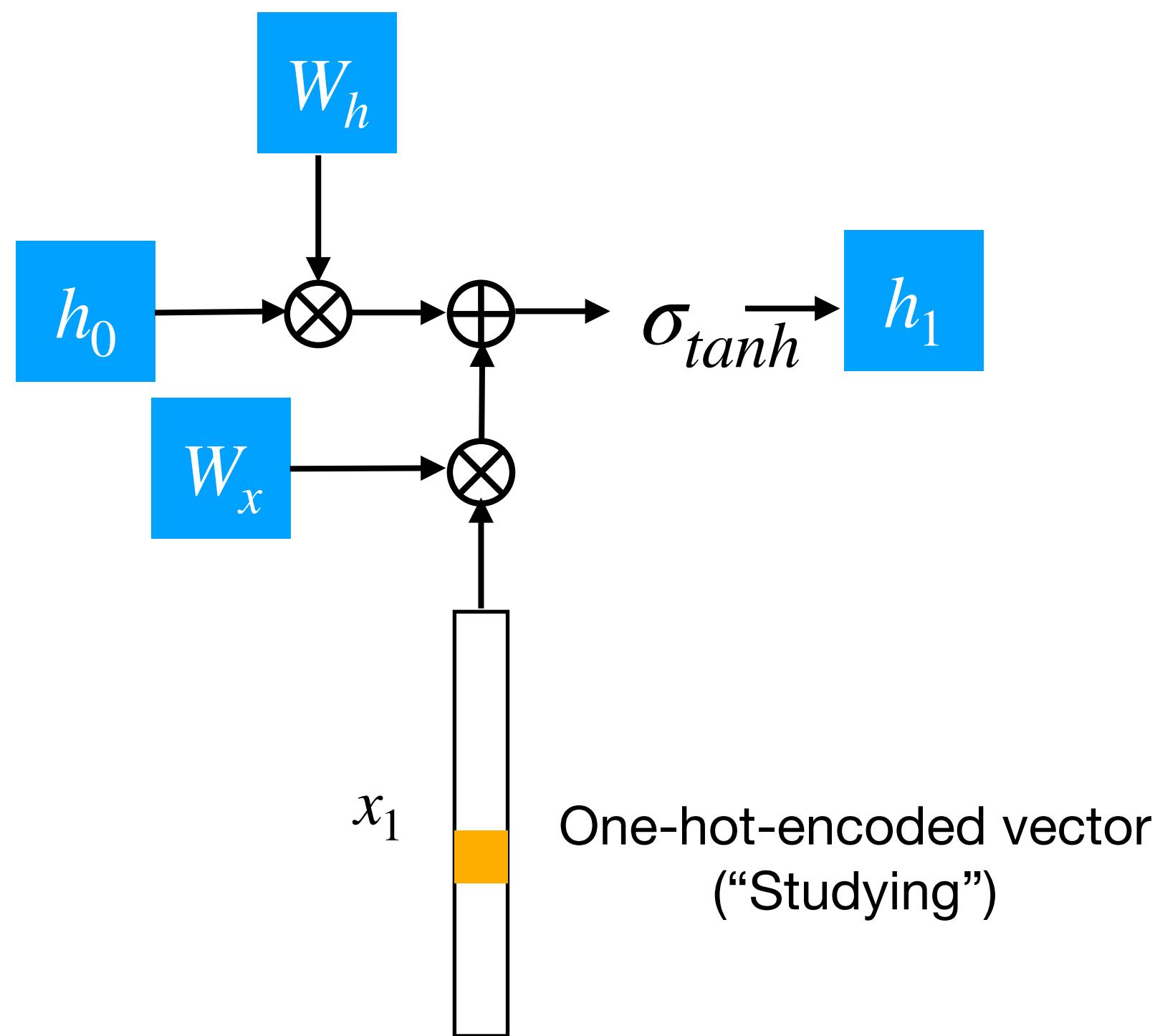
Copyright©2023. Acadential. All rights reserved.

15-2. RNN과 RNN의 forward propagation

Recurrent Neural Networks

RNN

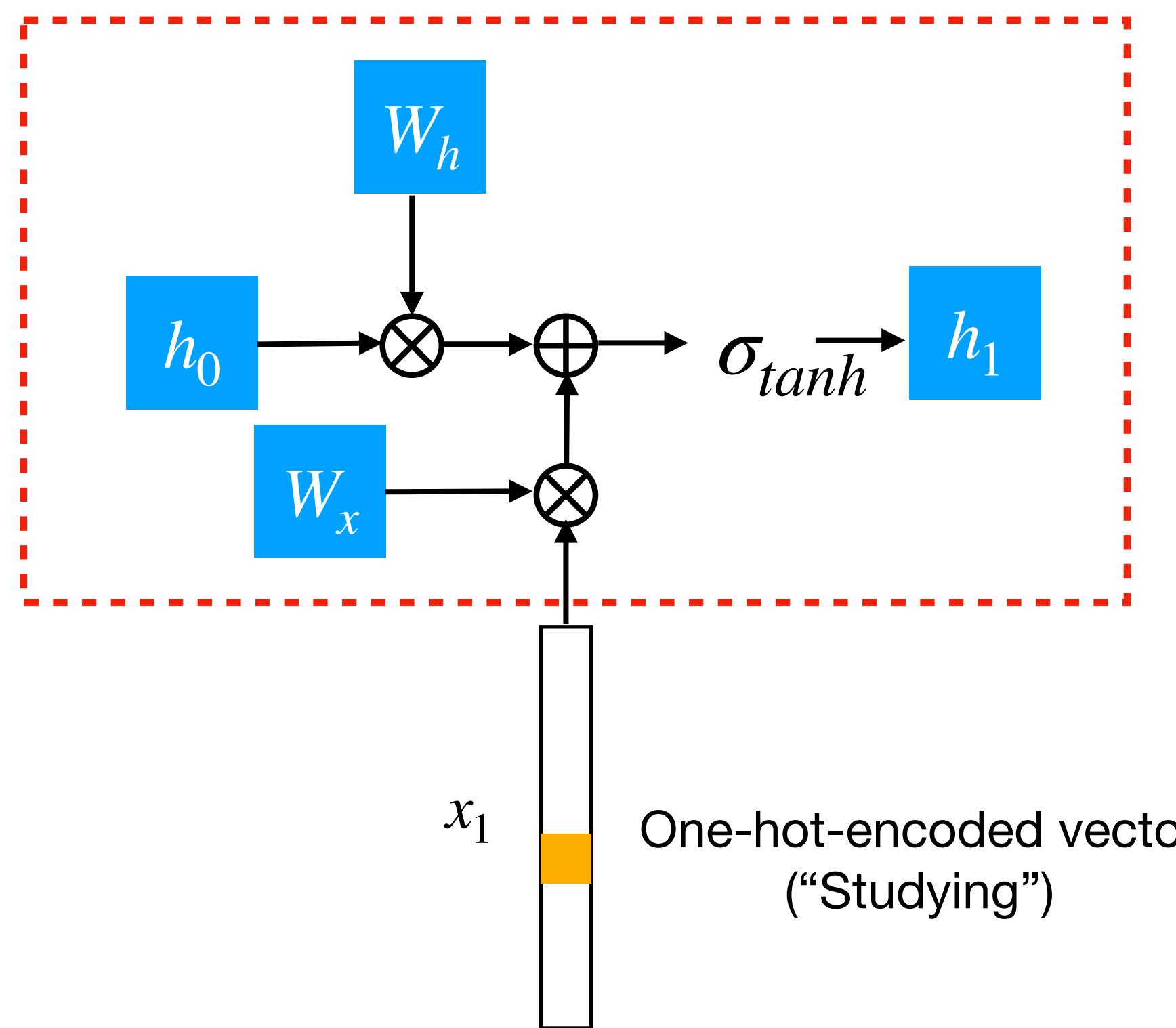
- RNN:
 - 배열의 맥락 (context)을 저장해두는 hidden state h_t 을 활용한다.



Recurrent Neural Networks

RNN

- RNN:
 - 배열의 맥락 (context)을 저장해두는 hidden state h_t 을 활용한다.



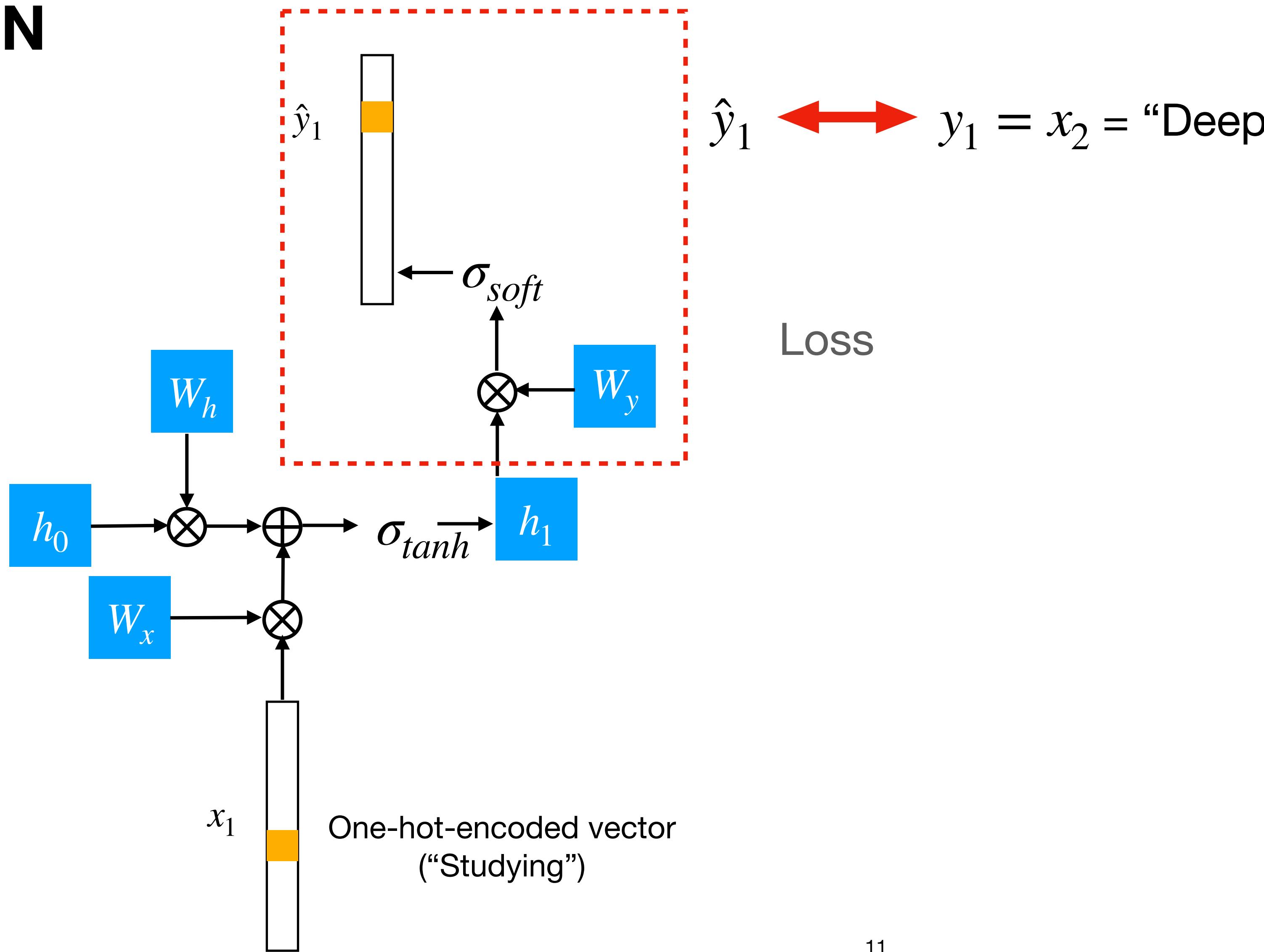
$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

h_t 은 time-step t 까지의 맥락을 저장해두는
hidden state이다!

\otimes : matrix multiplication
 \oplus : element-wise addition
 σ_{tanh} : tanh

Recurrent Neural Networks

RNN



$$p(\hat{y}_t) = \text{softmax}(W_y h_t)$$

즉, Hidden state h_t 으로부터
 y_t 을 예측.

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

\otimes : matrix multiplication

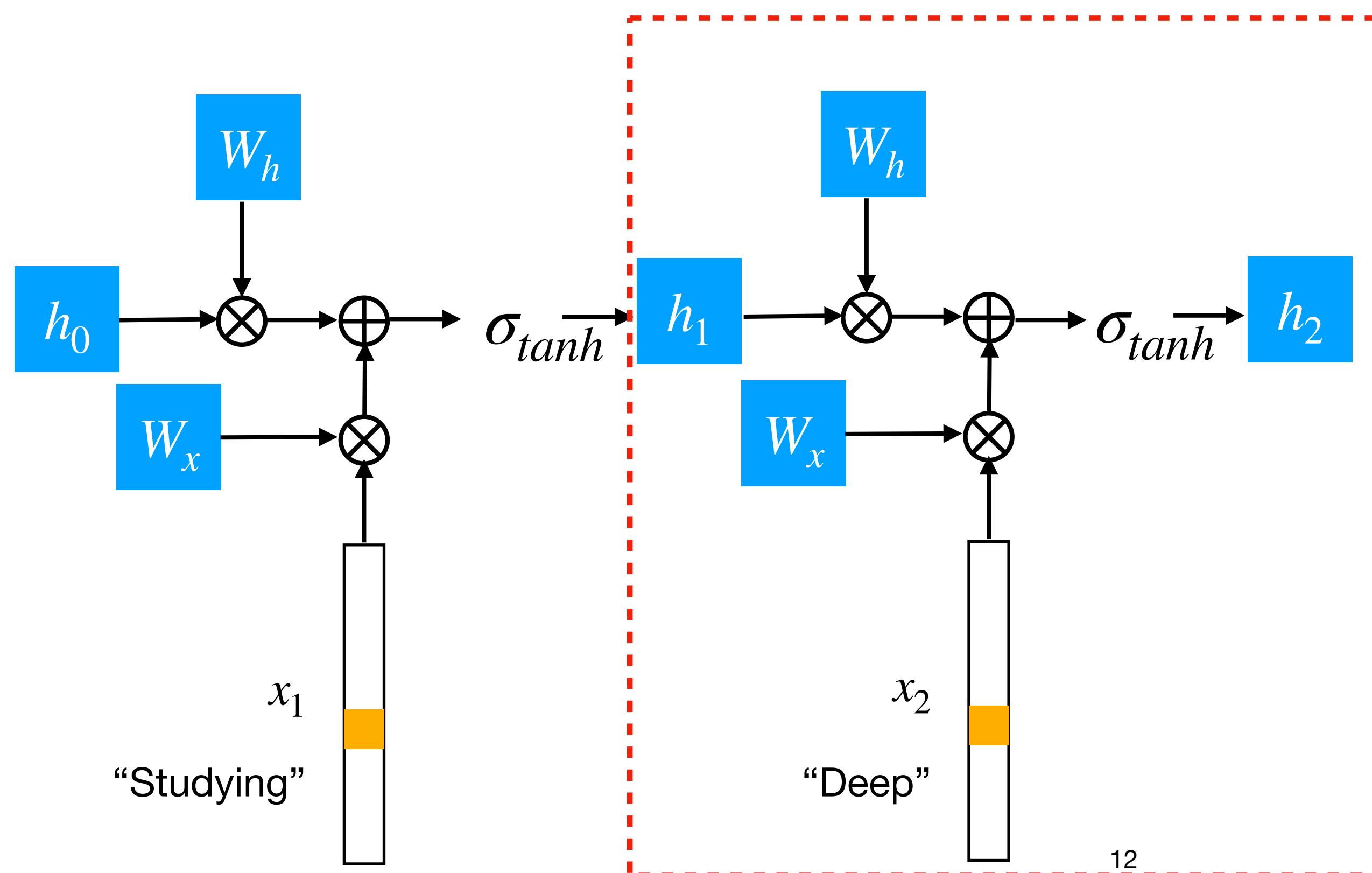
\oplus : element-wise addition

σ_{tanh} : tanh

Recurrent Neural Networks

RNN

Copyright©2023. Acadential. All rights reserved.



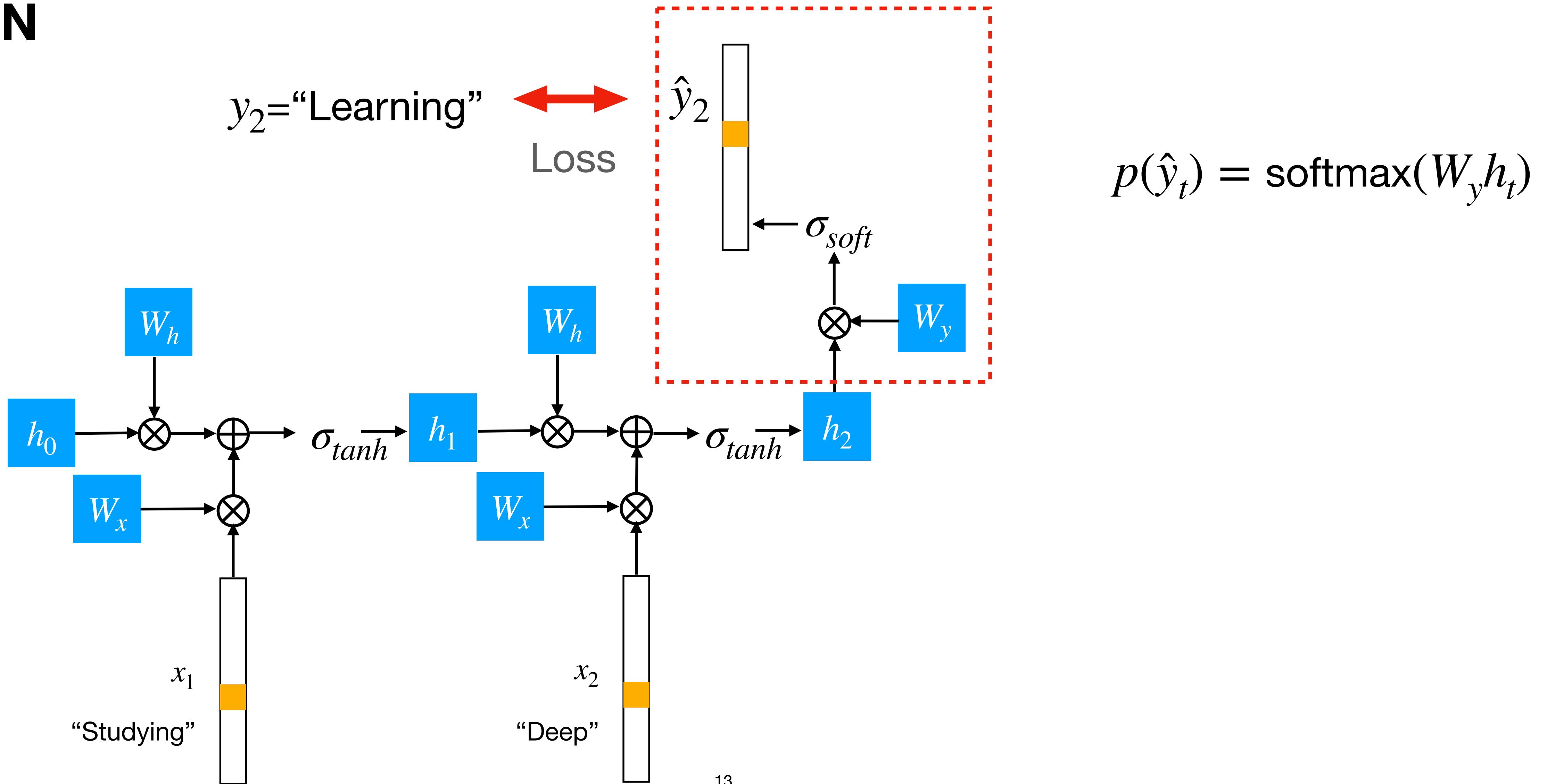
$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

h_1 과 x_2 을 입력받아서 h_2 을 계산한다!

그리고 weight W_h , W_x , W_y 은 t 와 관계없이 모두 동일하다 (즉, shared parameter)!

Recurrent Neural Networks

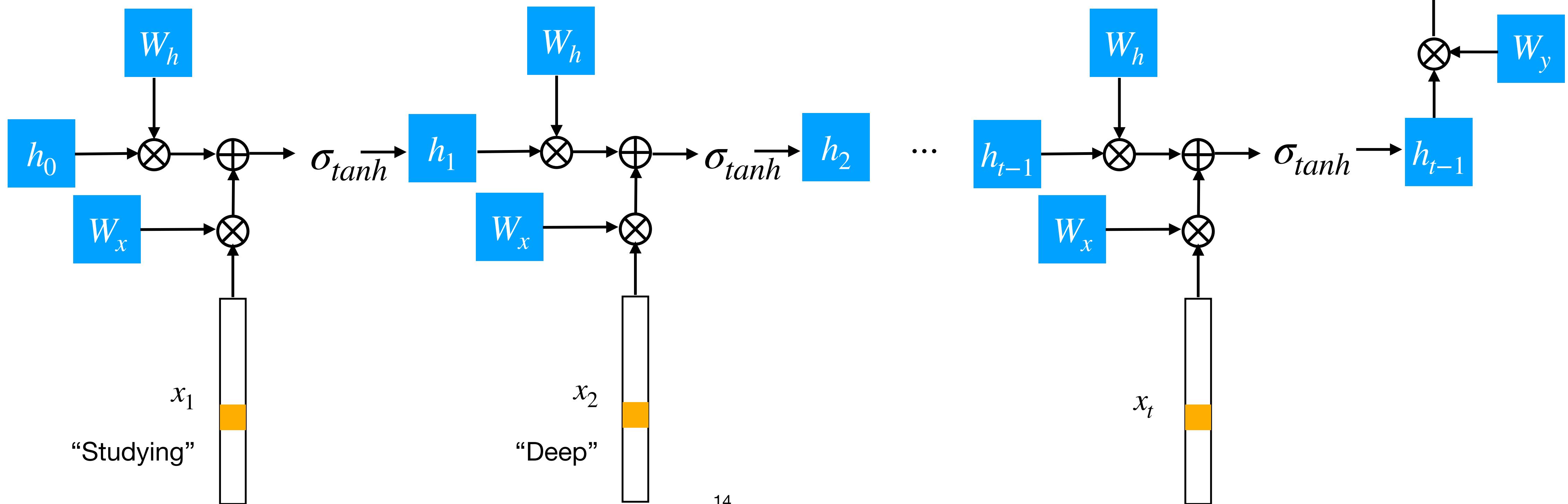
RNN



Recurrent Neural Networks

RNN

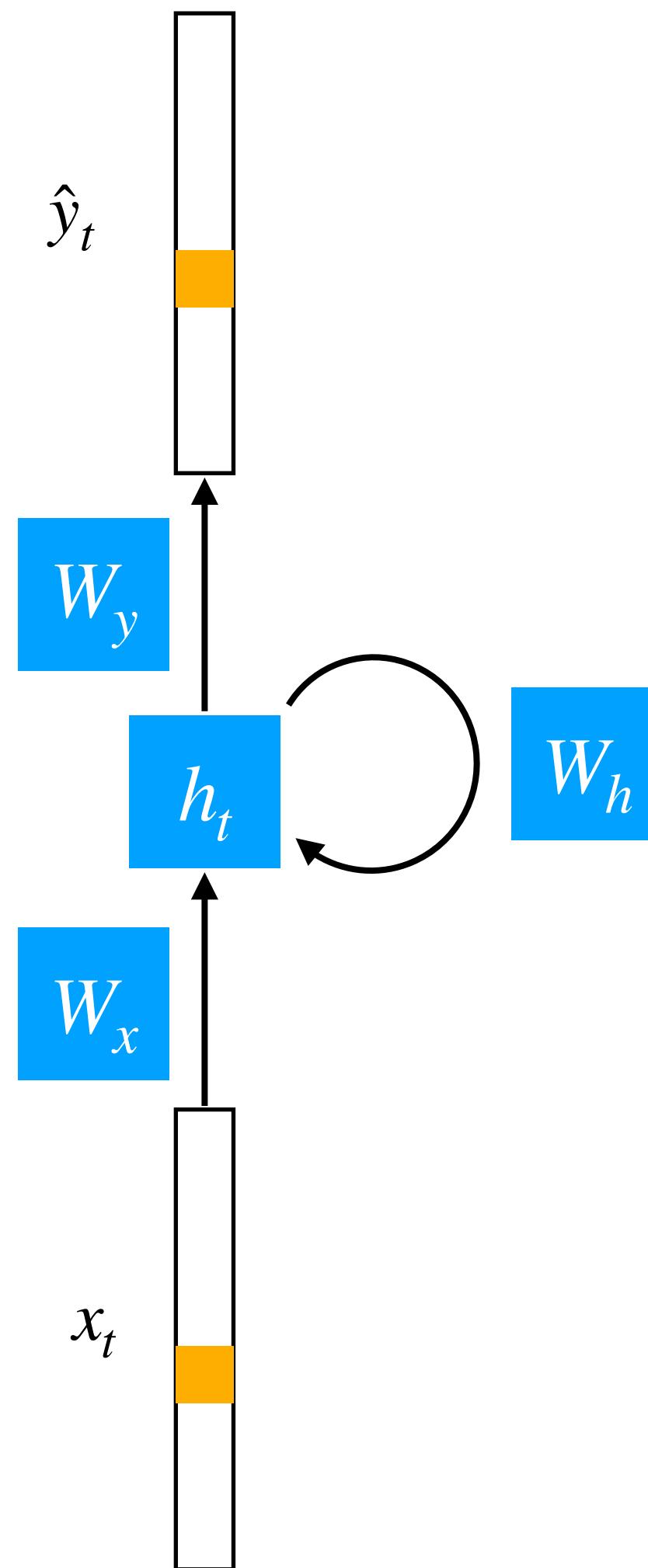
- 해당 과정이 계속 반복된다!
- 이전 RNN unit에서 계산된 hidden state h_{t-1} 은 다음 RNN unit의 입력이 된다!
- 다음 장에서 이것을 좀 더 concisely하게 visualize해보자.



Recurrent Neural Networks

Copyright©2023. Acadential. All rights reserved.

RNN

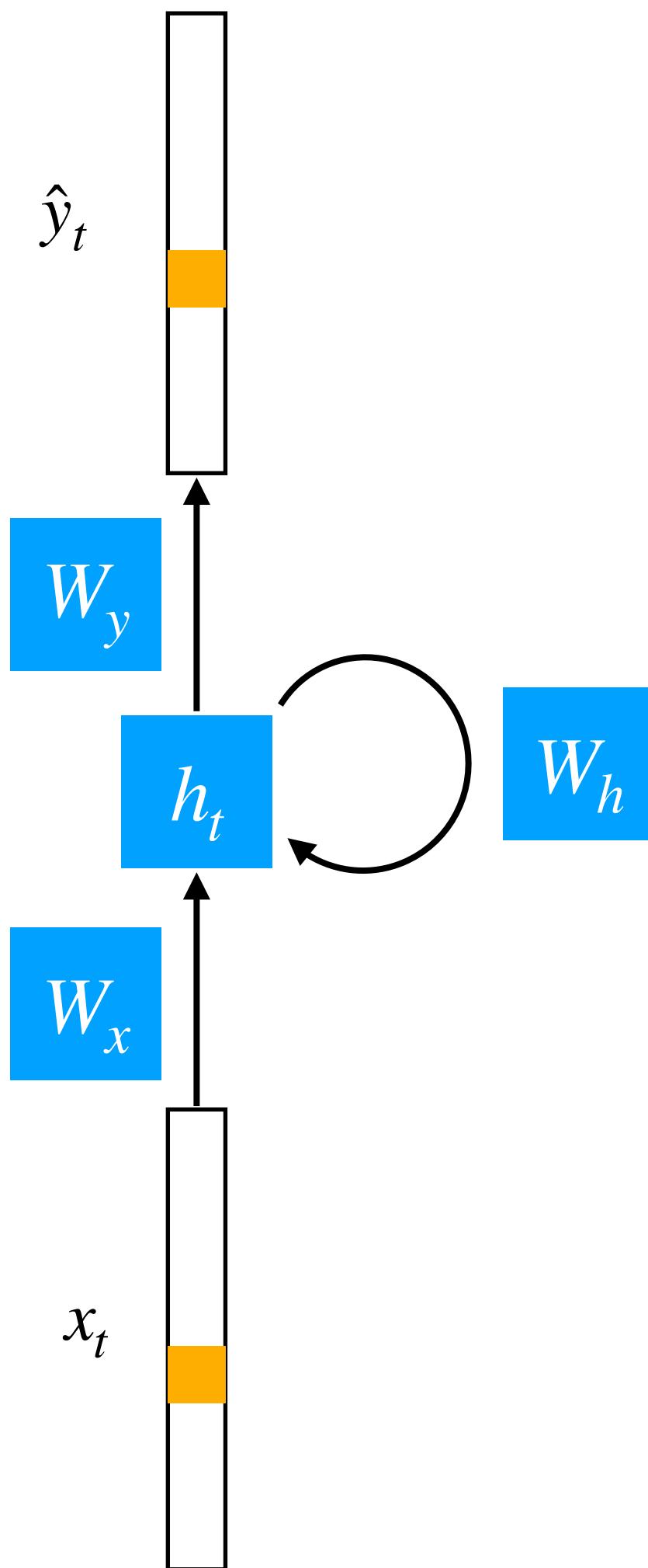


W_x, W_h, W_y 은 time-step과 무관하게 동일하다!
(shared parameter이다!)

즉, 동일한 RNN unit들이다.

Recurrent Neural Networks

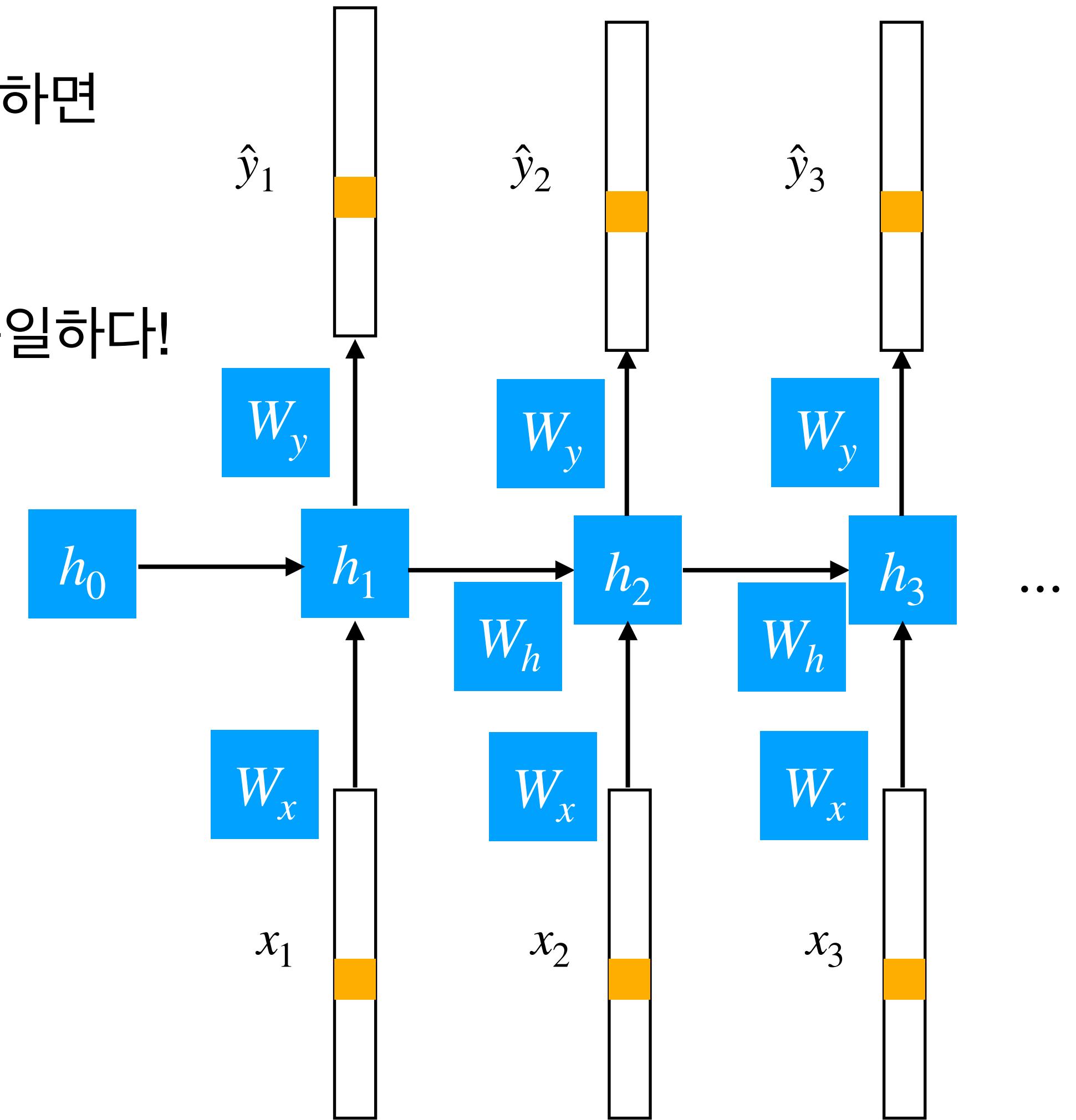
RNN



RNN을 time-step에 대해서 풀어서 표현하면
오른쪽과 같다.

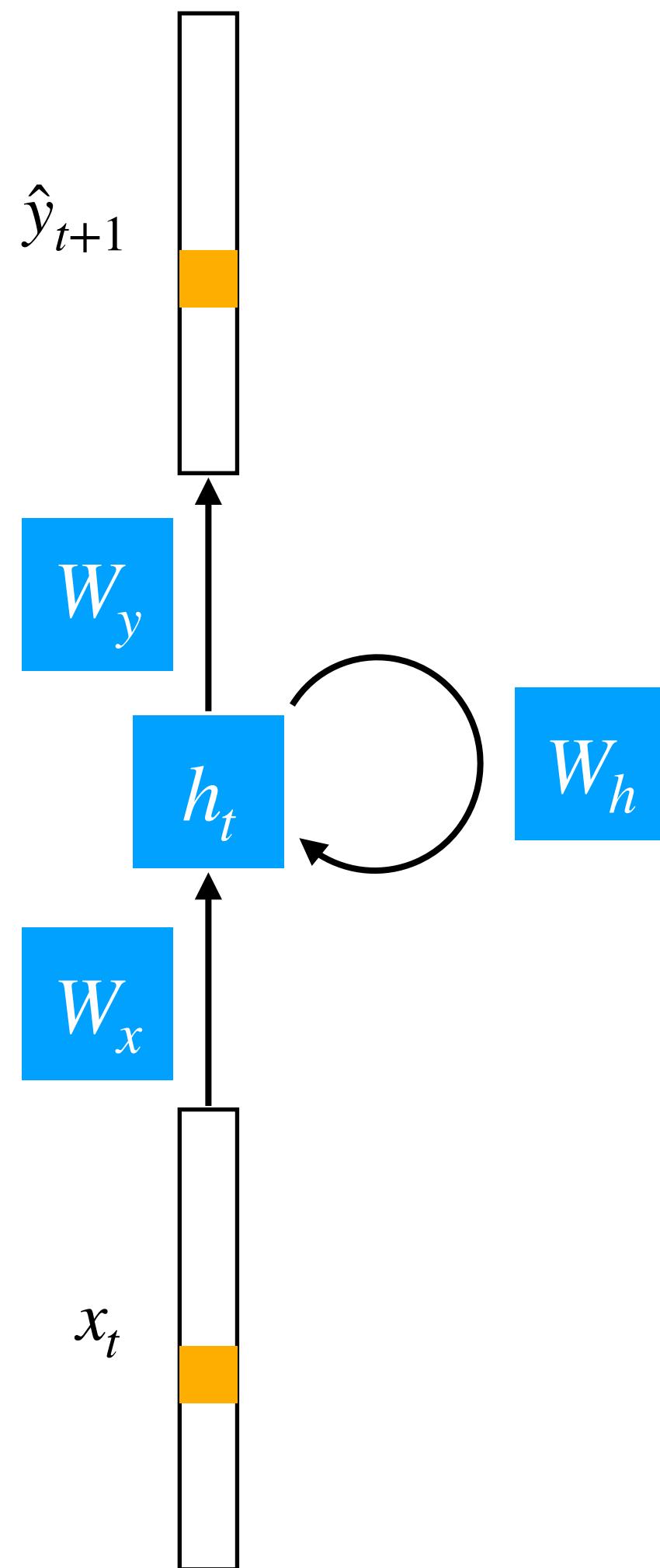
W_x, W_h, W_y 은 time-step과 무관하게 동일하다!
(shared parameter이다!)

즉, 동일한 RNN unit들이다.



Recurrent Neural Networks

RNN

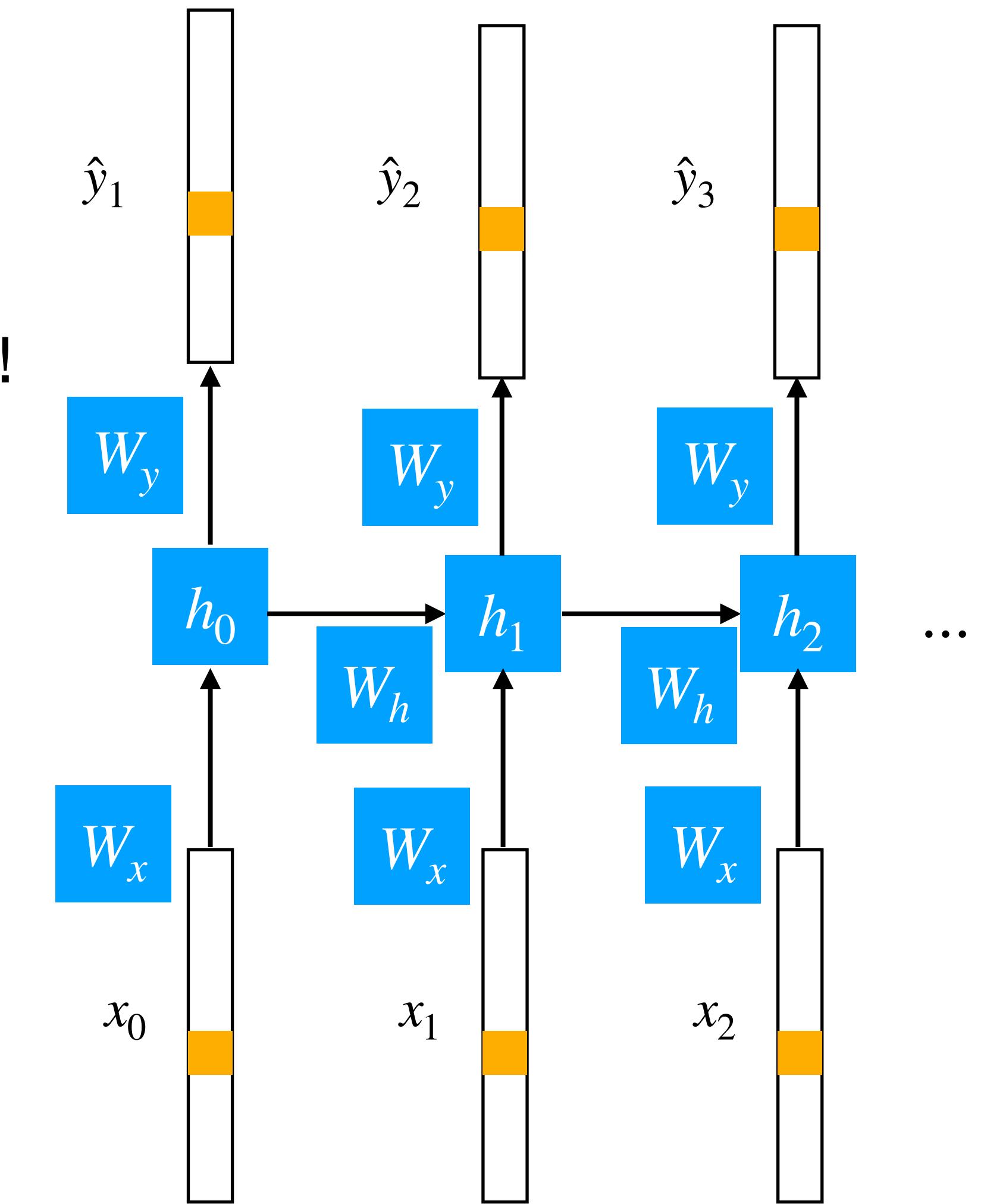


RNN을 time-step에 대해서 풀어서 표현하면
오른쪽과 같다.

W_x, W_h, W_y 은 time-step과 무관하게 동일하다!
(shared parameter이다!)

즉, 동일한 RNN unit들이다.

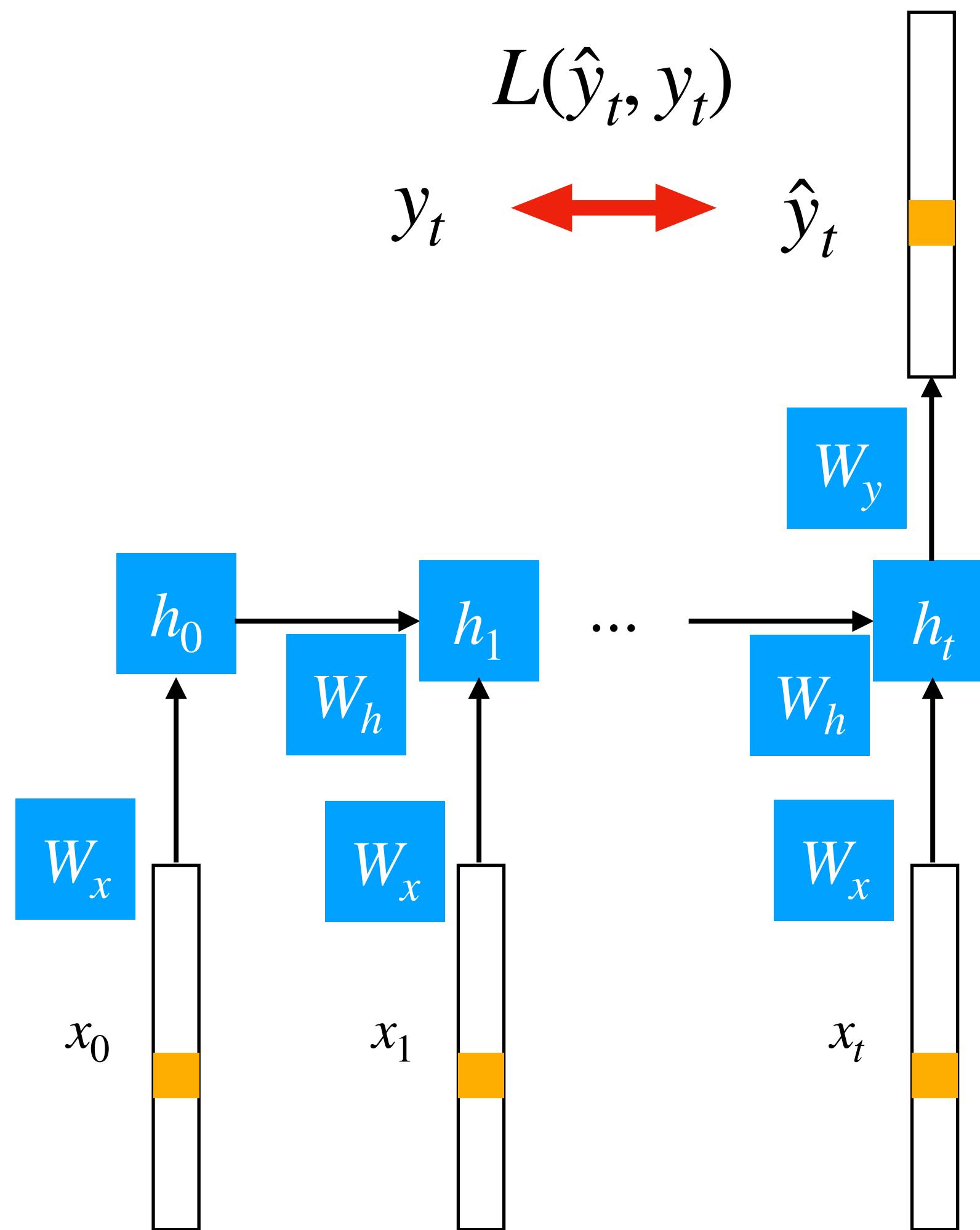
그렇다면 RNN에 대한
backpropagation은 어떻게 할까?



15-3. RNN의 Backpropagation

Recurrent Neural Networks

Differentiating RNN

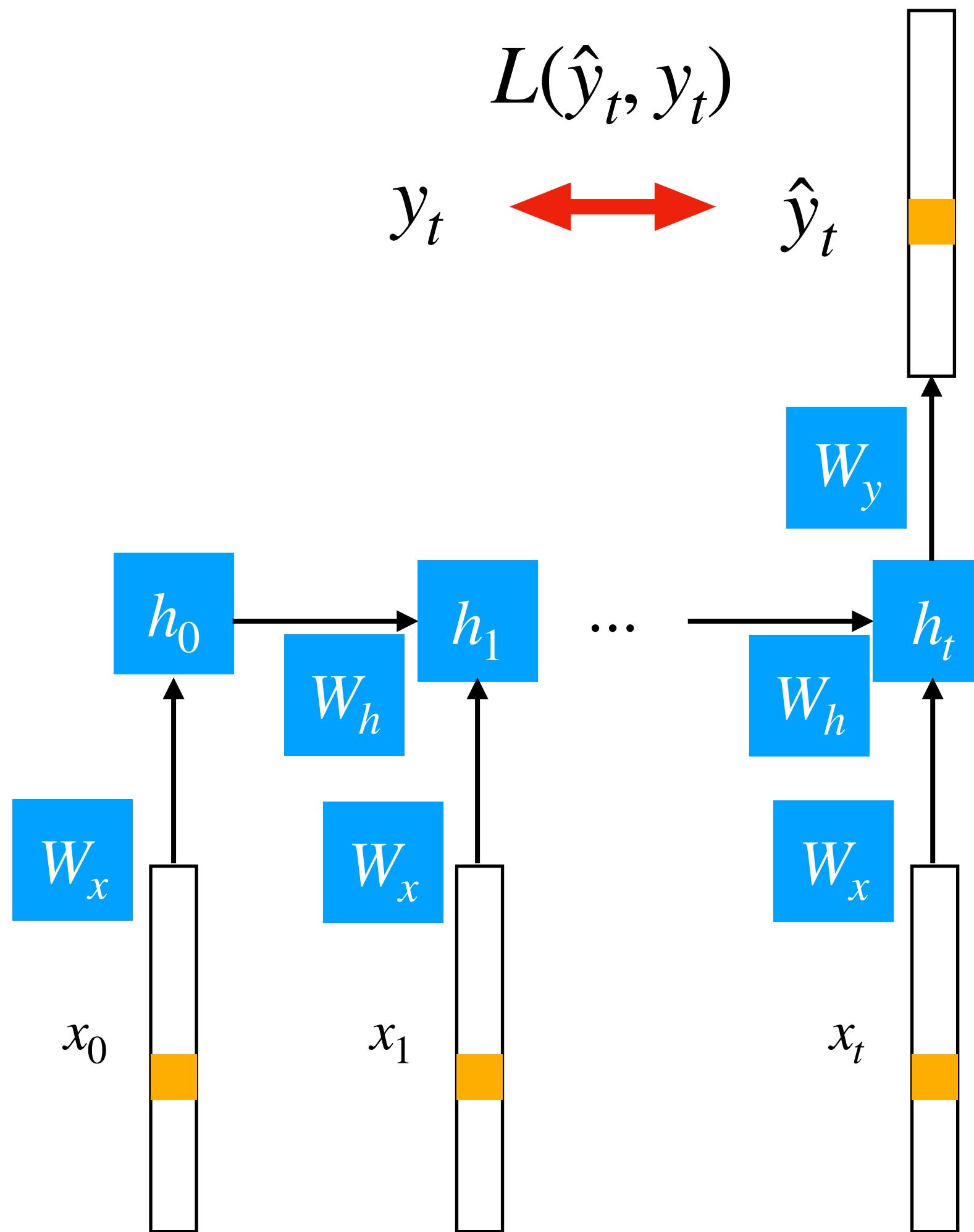


Loss $L(\hat{y}_t, y_t | x_0, \dots x_t)$ 일 때,

RNN의 Weight W_y, W_h, W_x 에 대한 경사는 어떻게 구할까?

Recurrent Neural Networks

Differentiating RNN



Loss $L(\hat{y}_t, y_t | x_0, \dots, x_t)$ 일 때,

RNN의 Weight W_y, W_h, W_x 에 대한 경사는 어떻게 구할까?

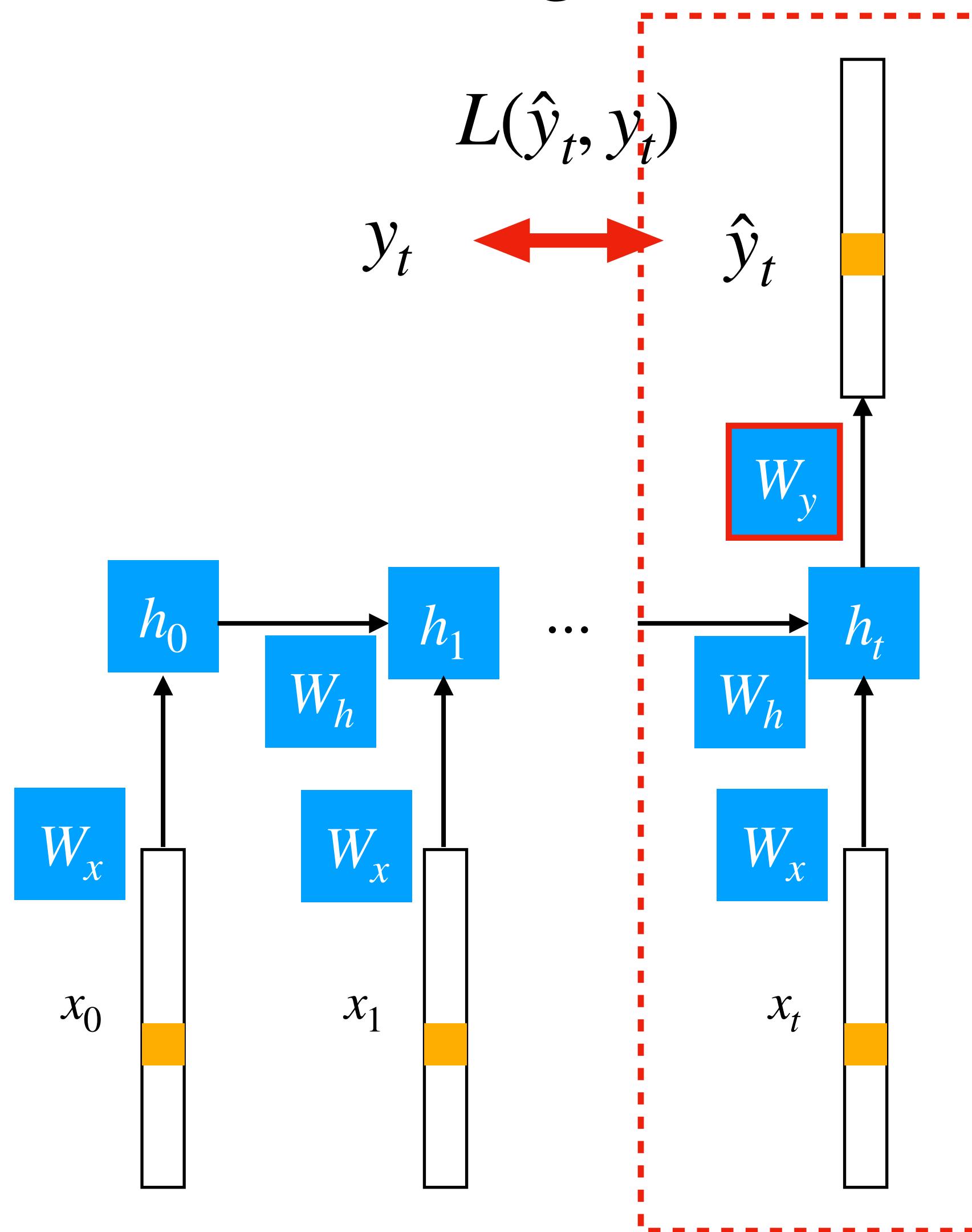
즉, 어떻게 하면

$\frac{\partial L}{\partial W_y}, \frac{\partial L}{\partial W_h}, \frac{\partial L}{\partial W_x}$ 을 구할까?

먼저 W_y 에 대한 Loss gradient인 $\frac{\partial L}{\partial W_y}$ 을 구해보자

Recurrent Neural Networks

Differentiating RNN



W_y 에 대한 Loss gradient인 $\frac{\partial L}{\partial W_y}$ 을 구하려면...

Computation graph 상에서 W_y 에서 L 을 잇는 경로는:

$$W_y \rightarrow \hat{y}_t \rightarrow L$$

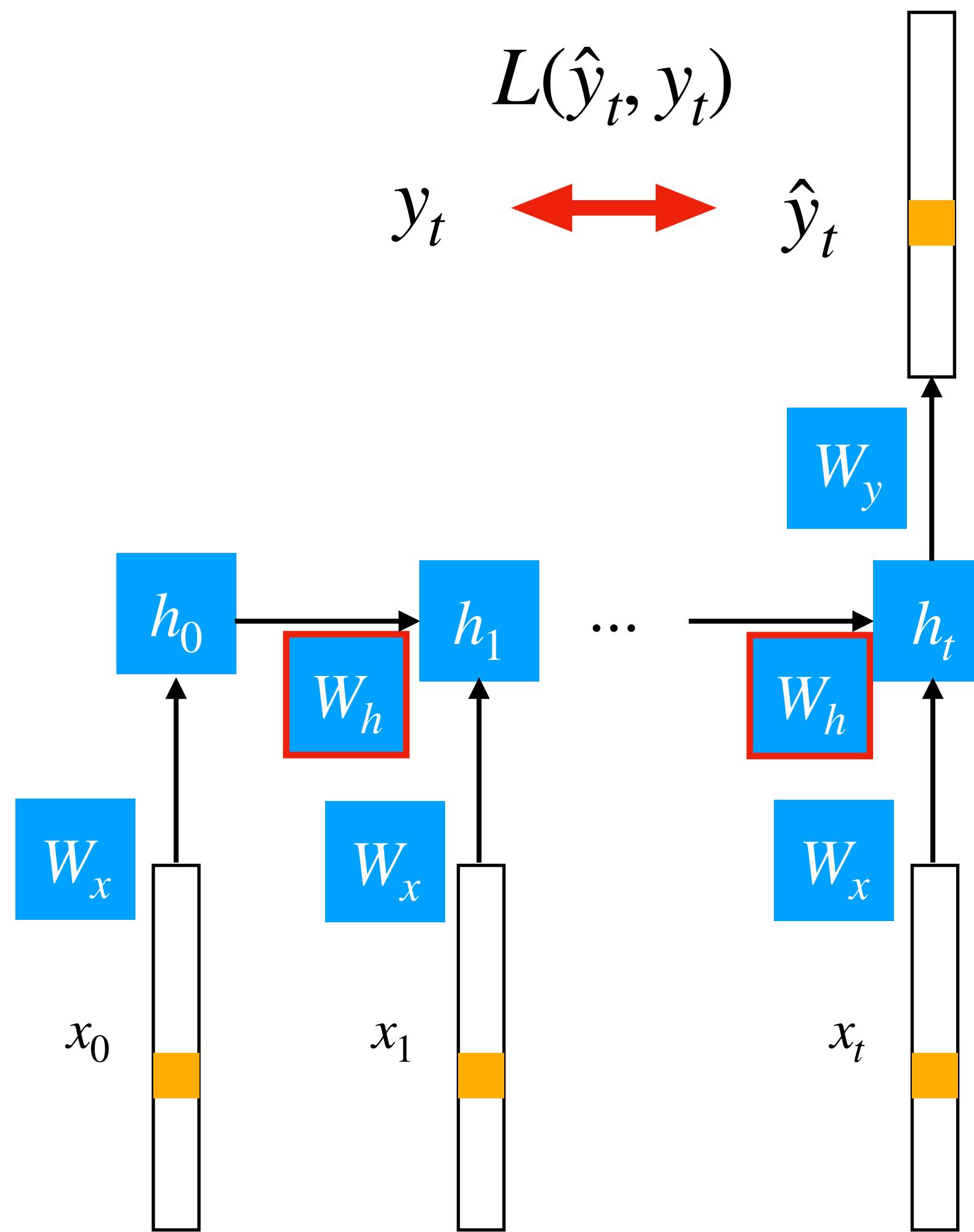
따라서:

$$\frac{\partial L}{\partial W_y} = \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial W_y}$$

그렇다면 W_h 에 대한 Loss gradient인 $\frac{\partial L}{\partial W_h}$ 은 어떻게 구할까?

Recurrent Neural Networks

Differentiating RNN



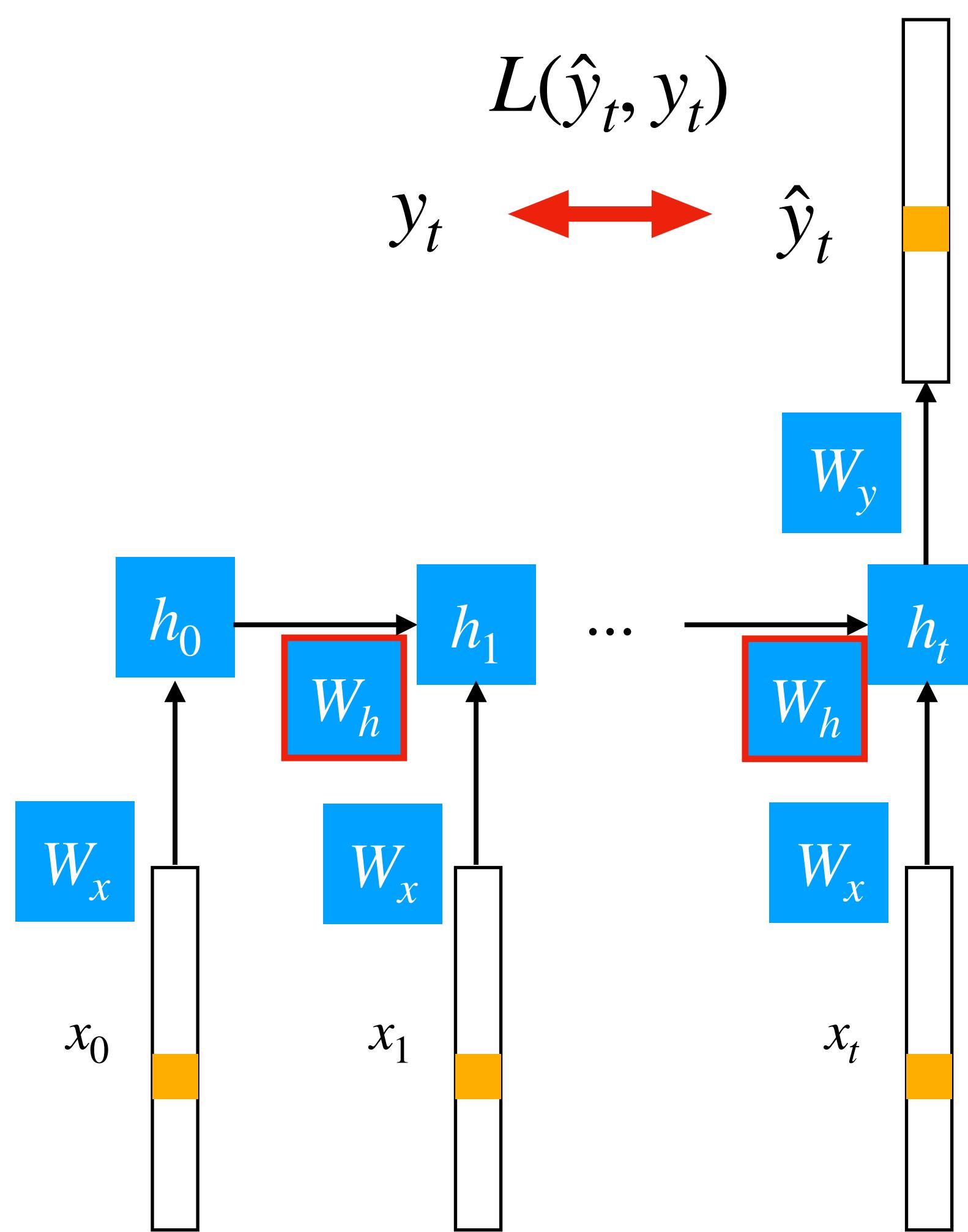
W_h 에 대한 Loss gradient인 $\frac{\partial L}{\partial W_h}$ 을 구해보자:

Loss $L(\hat{y}_t, y_t | x_0, \dots, x_t)$

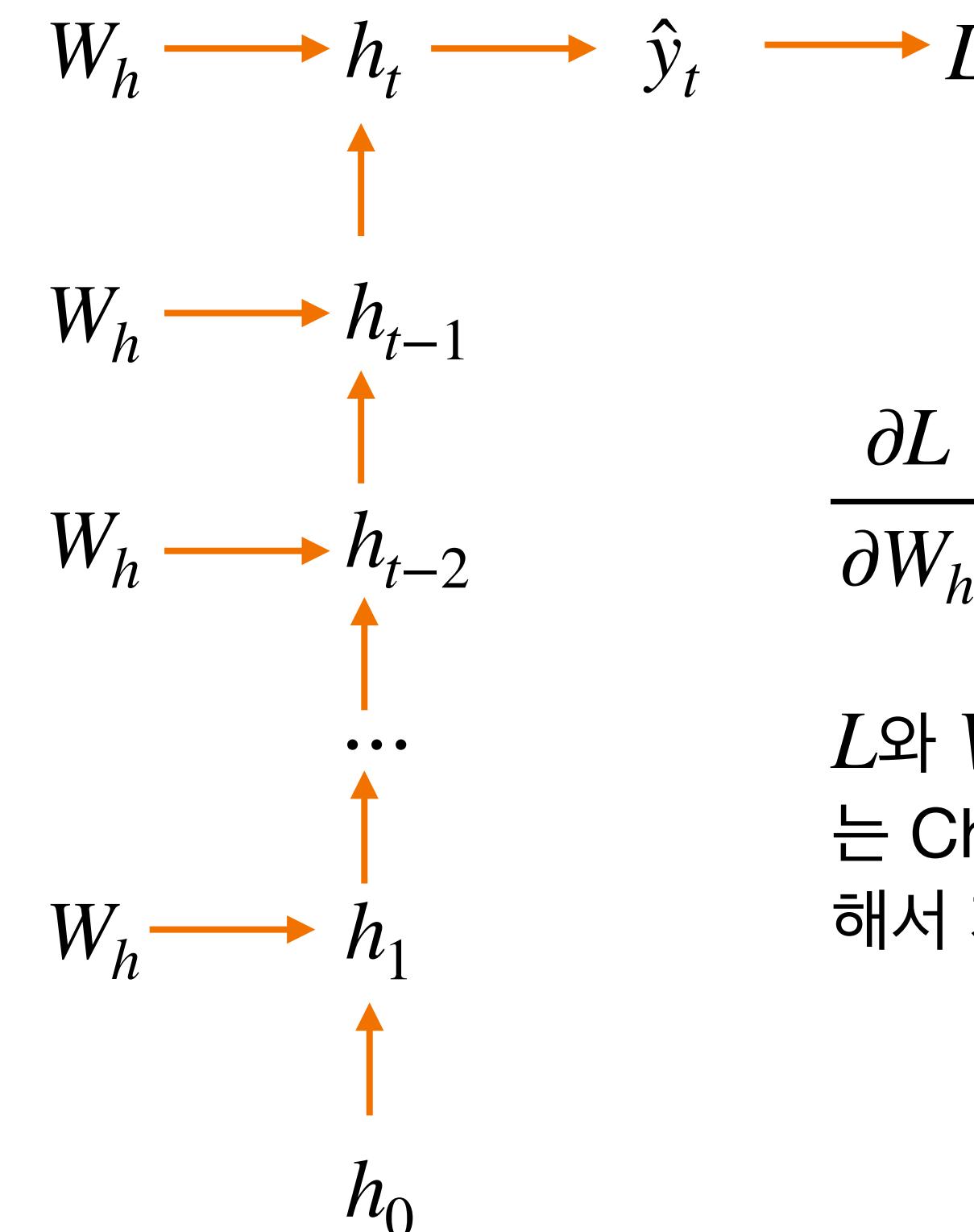
Computation graph 상에서 각 W_h 을 L 과 잇는 경로는
어떻게 될까?

Recurrent Neural Networks

Differentiating RNN



W_h 와 L 을 잇는 경로만 포함한 Computation Graph은 다음과 같다:

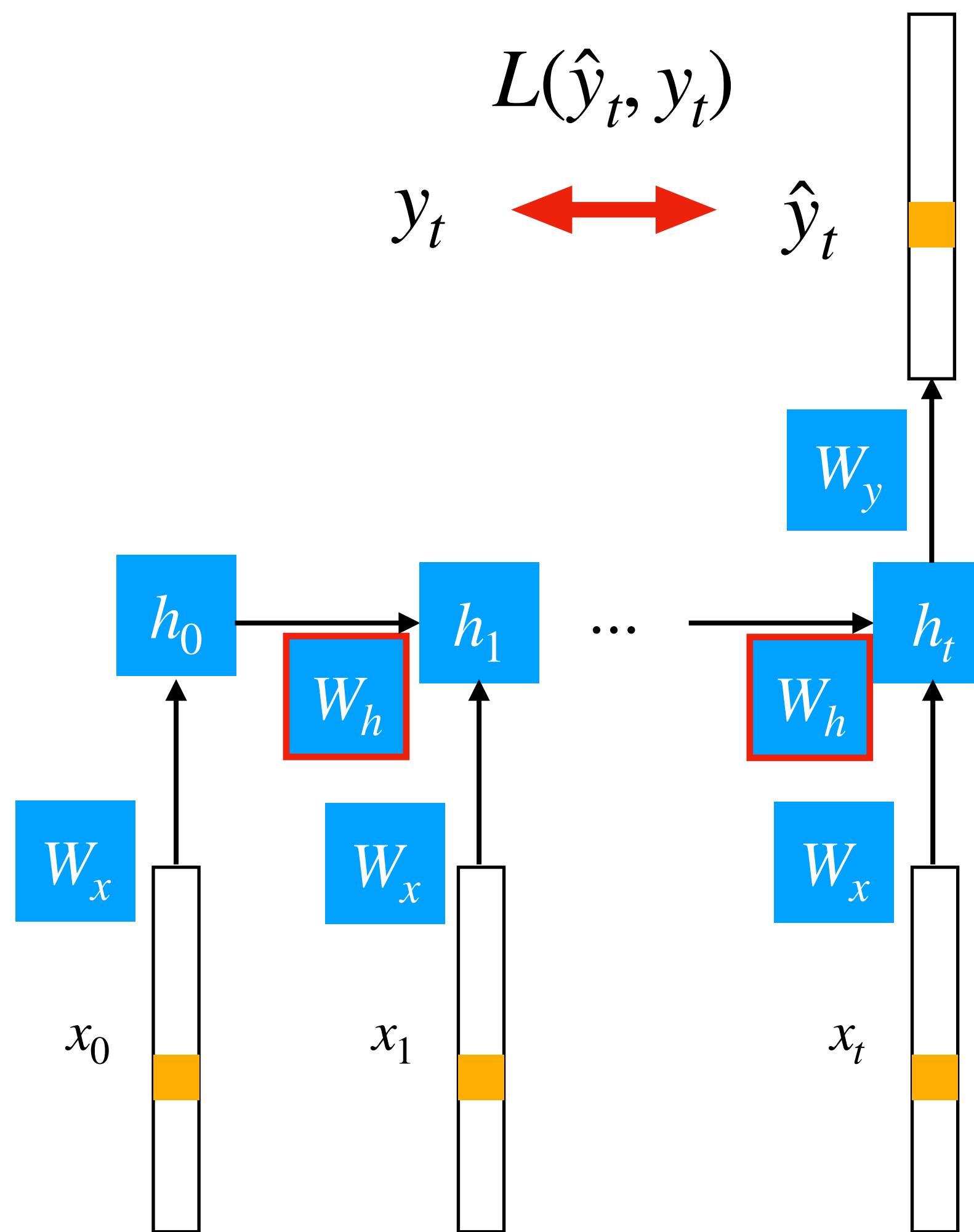


$\frac{\partial L}{\partial W_h}$ 을 구하려면:

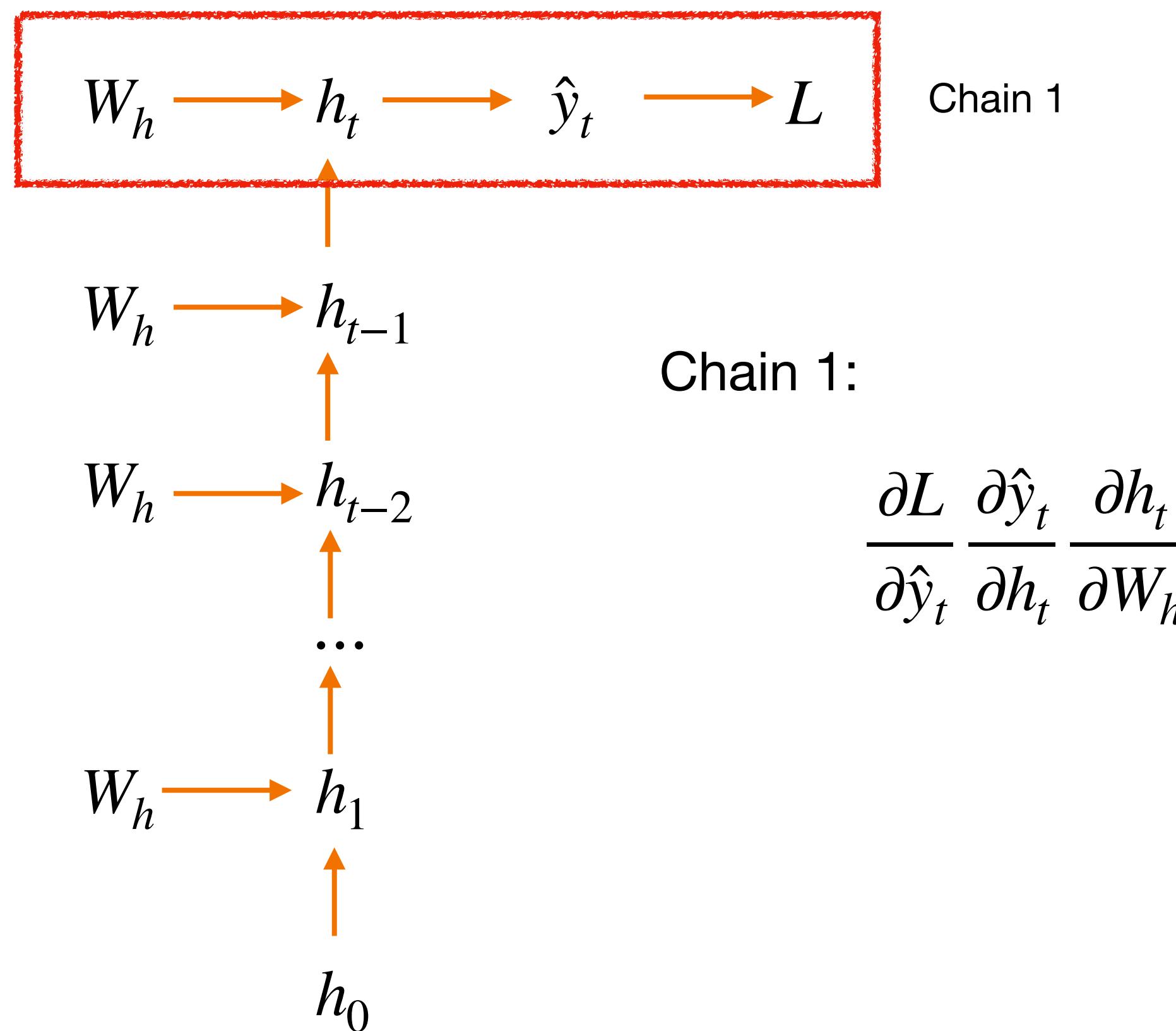
L 와 W_h 을 잇는 각 경로에 해당되는 Chain of derivative들을 다 더 해서 계산.

Recurrent Neural Networks

Differentiating RNN



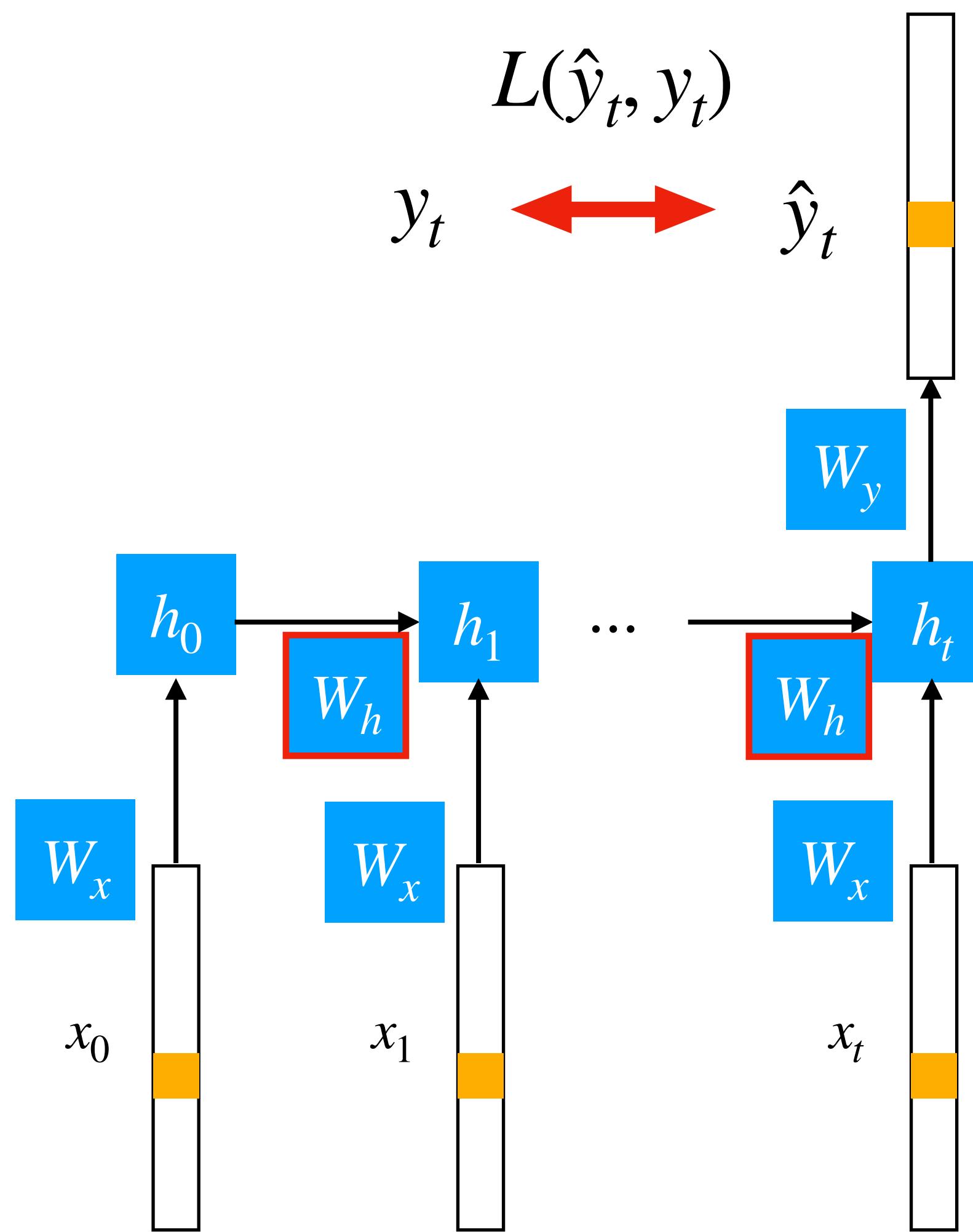
Computation graph은 다음과 같다:



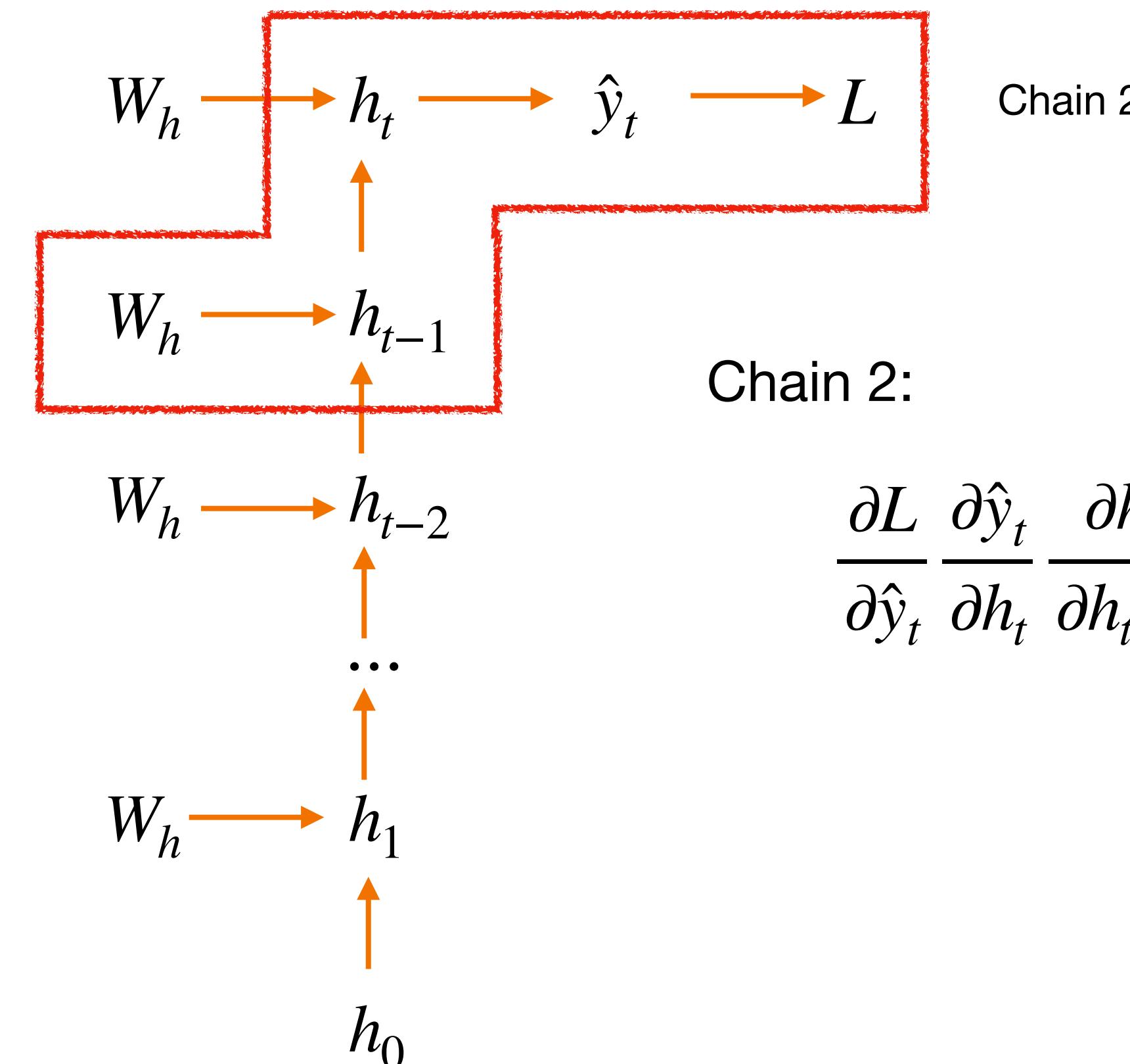
$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W_h}$$

Recurrent Neural Networks

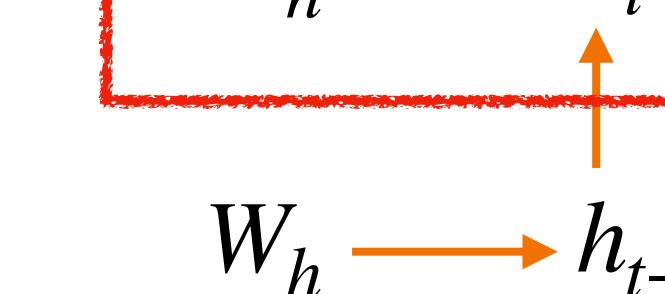
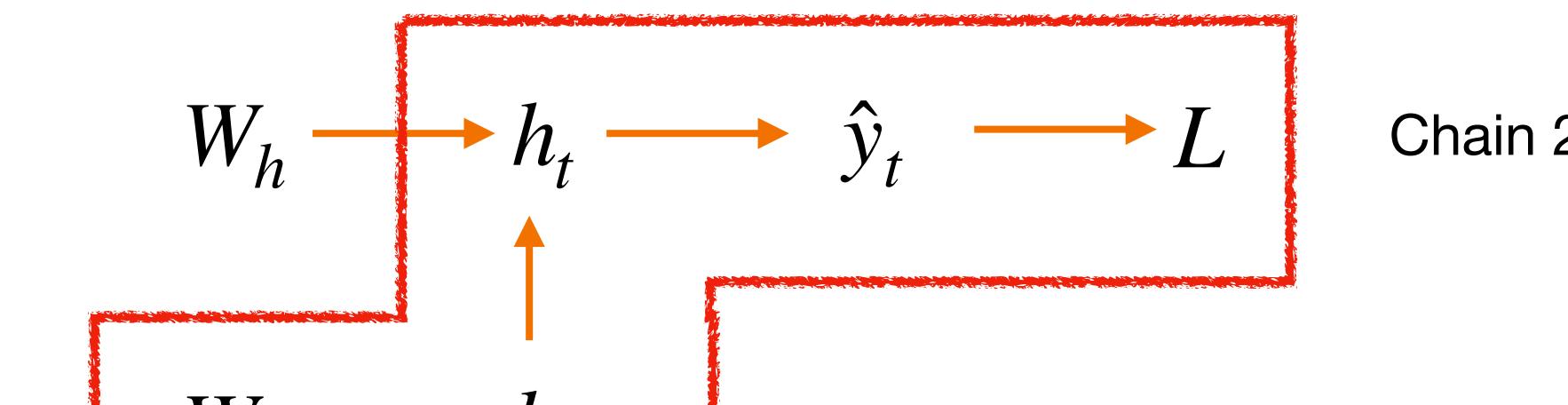
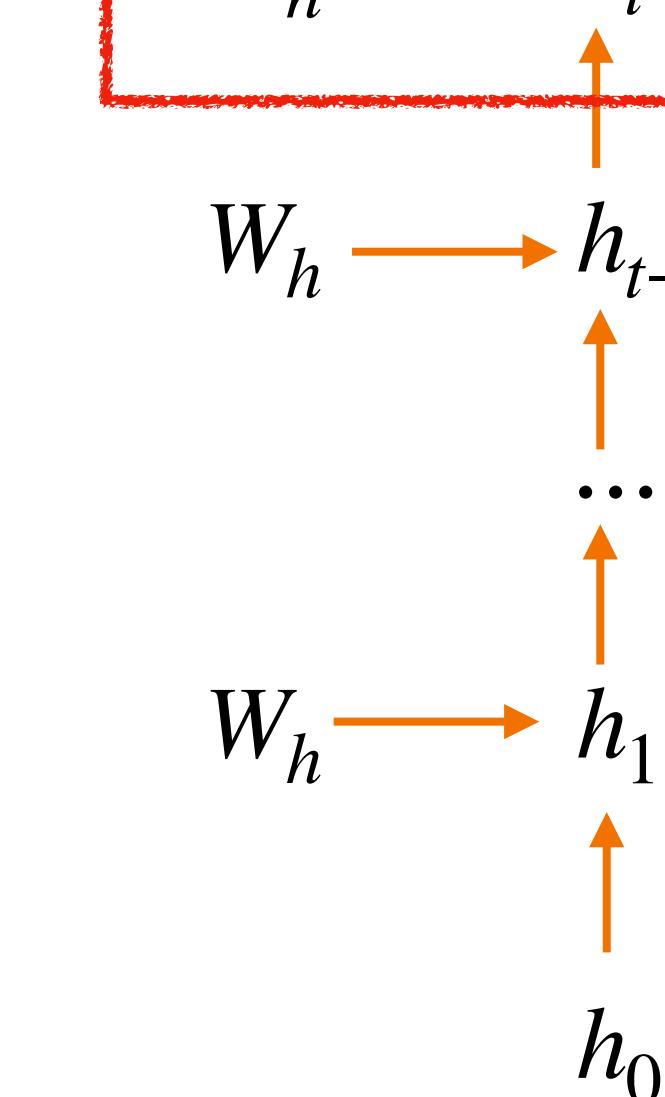
Differentiating RNN



Computation graph은 다음과 같다:

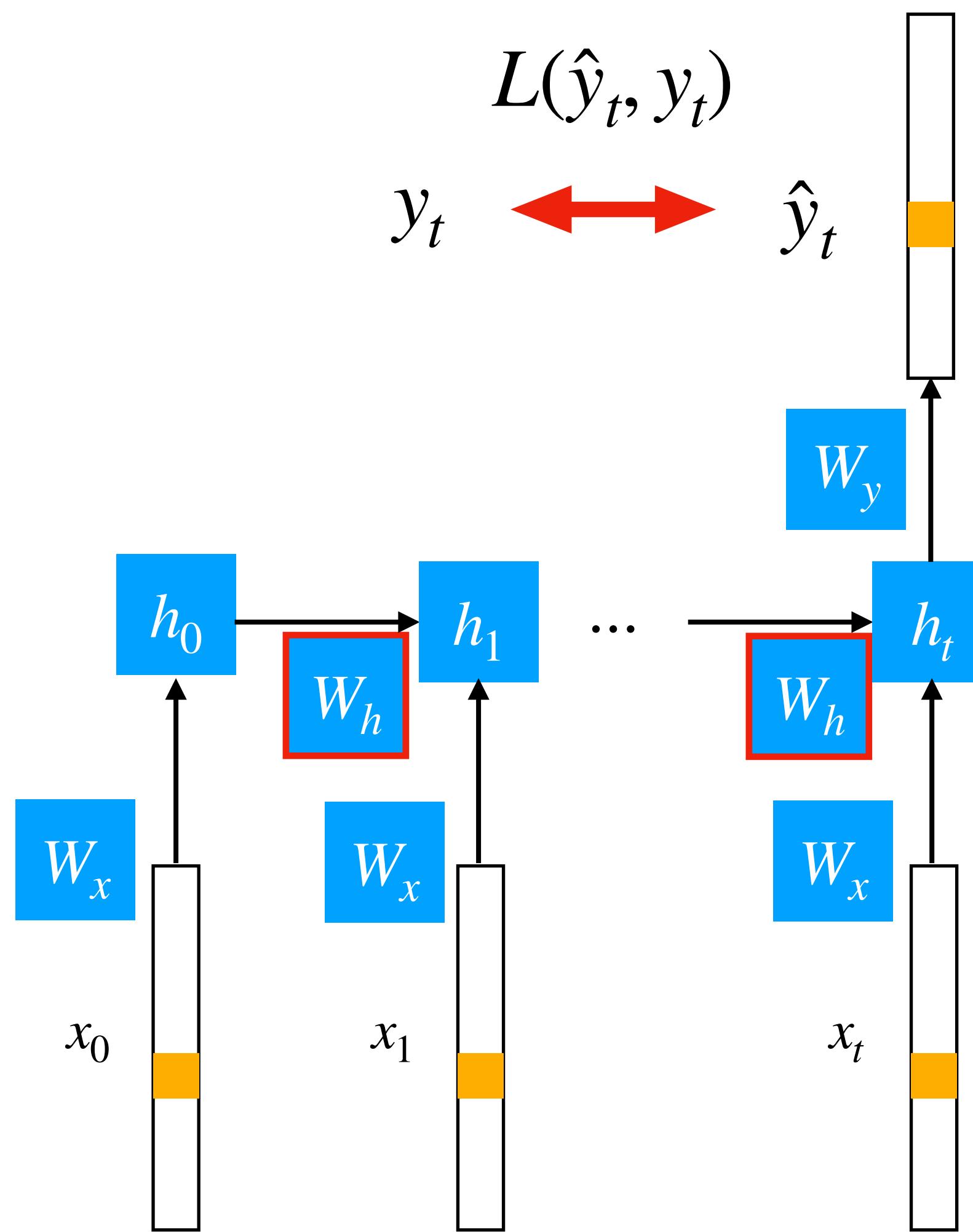


$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_h}$$

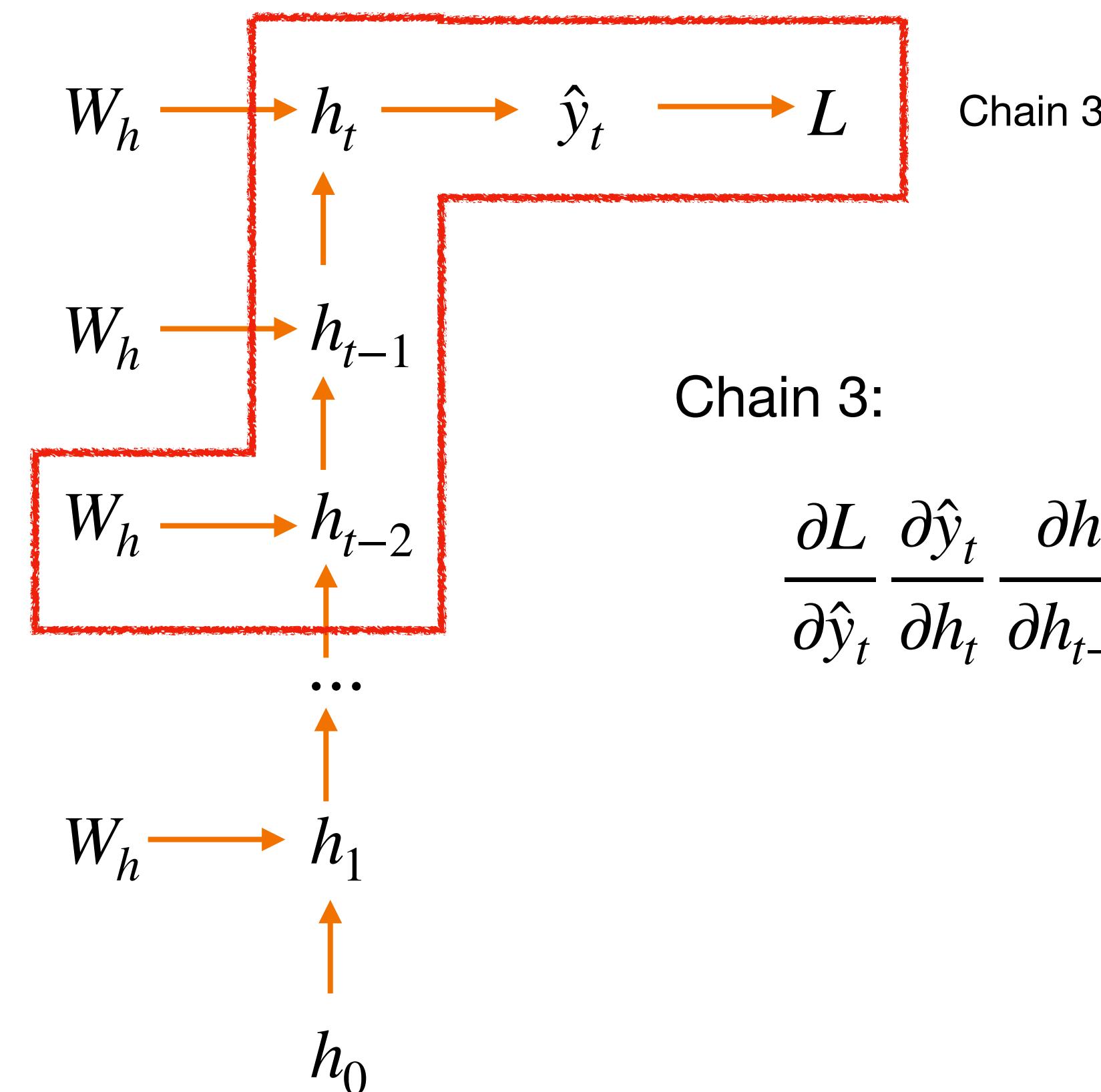


Recurrent Neural Networks

Differentiating RNN



Computation graph은 다음과 같다:

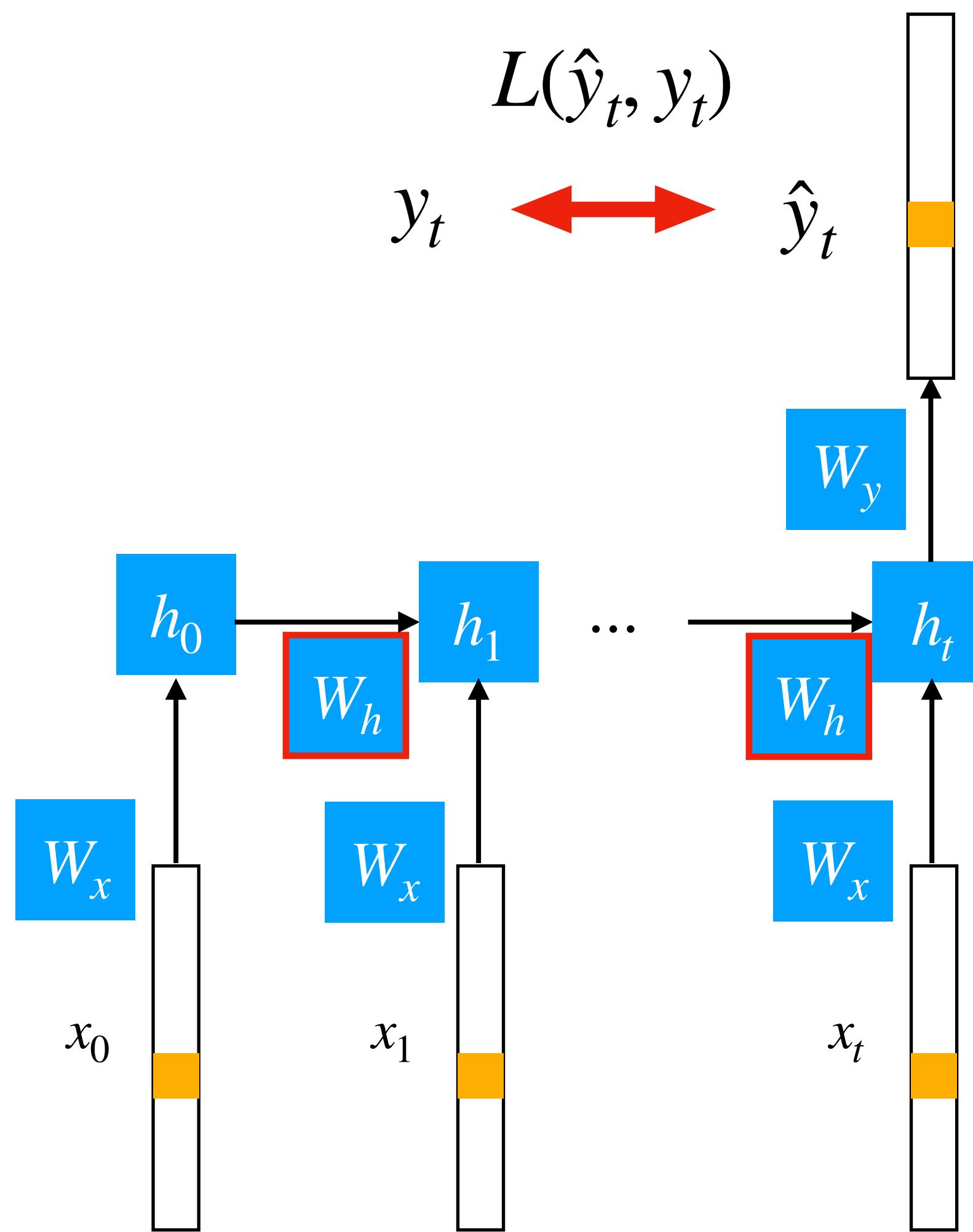


Chain 3:

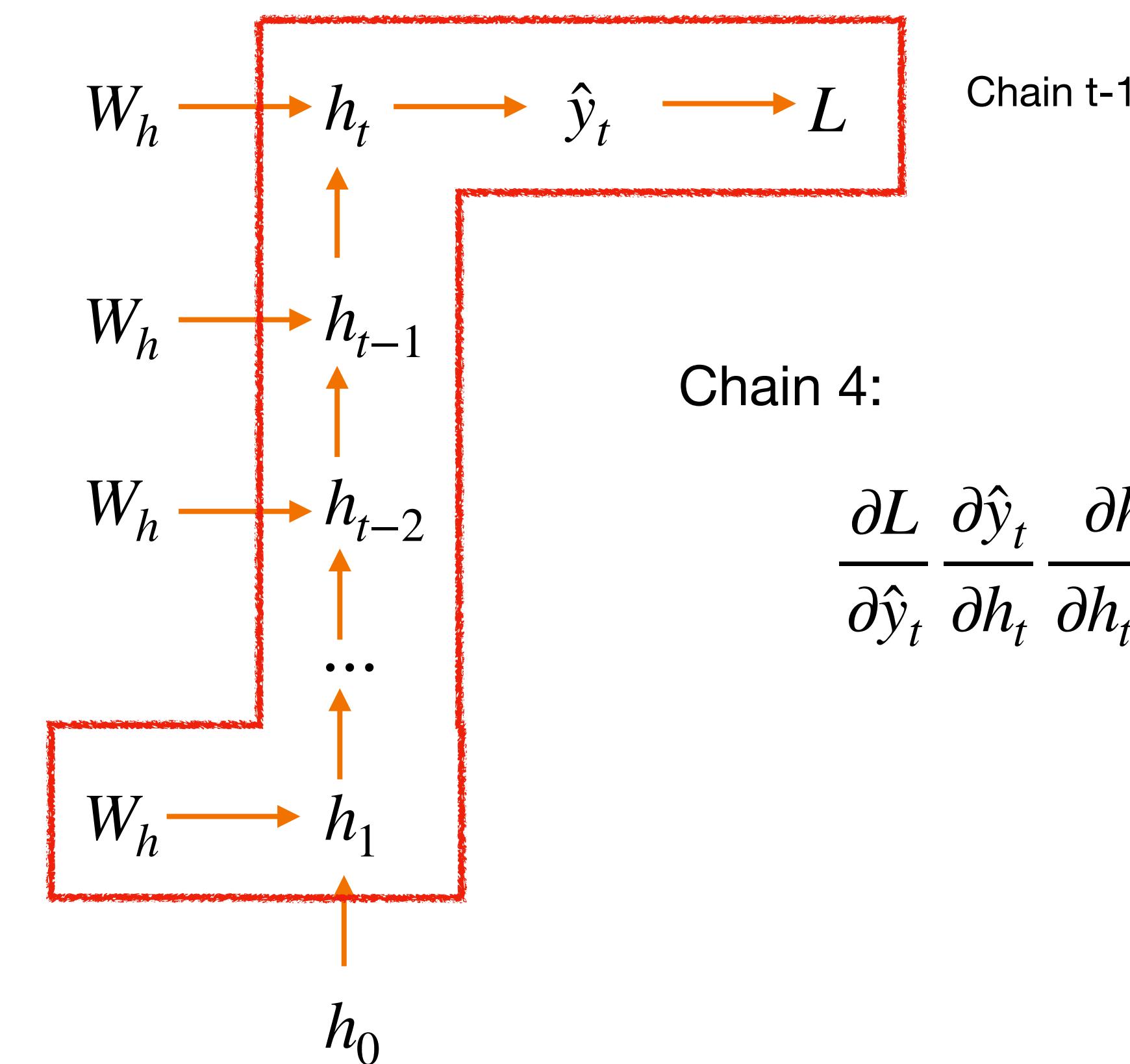
$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_h}$$

Recurrent Neural Networks

Differentiating RNN

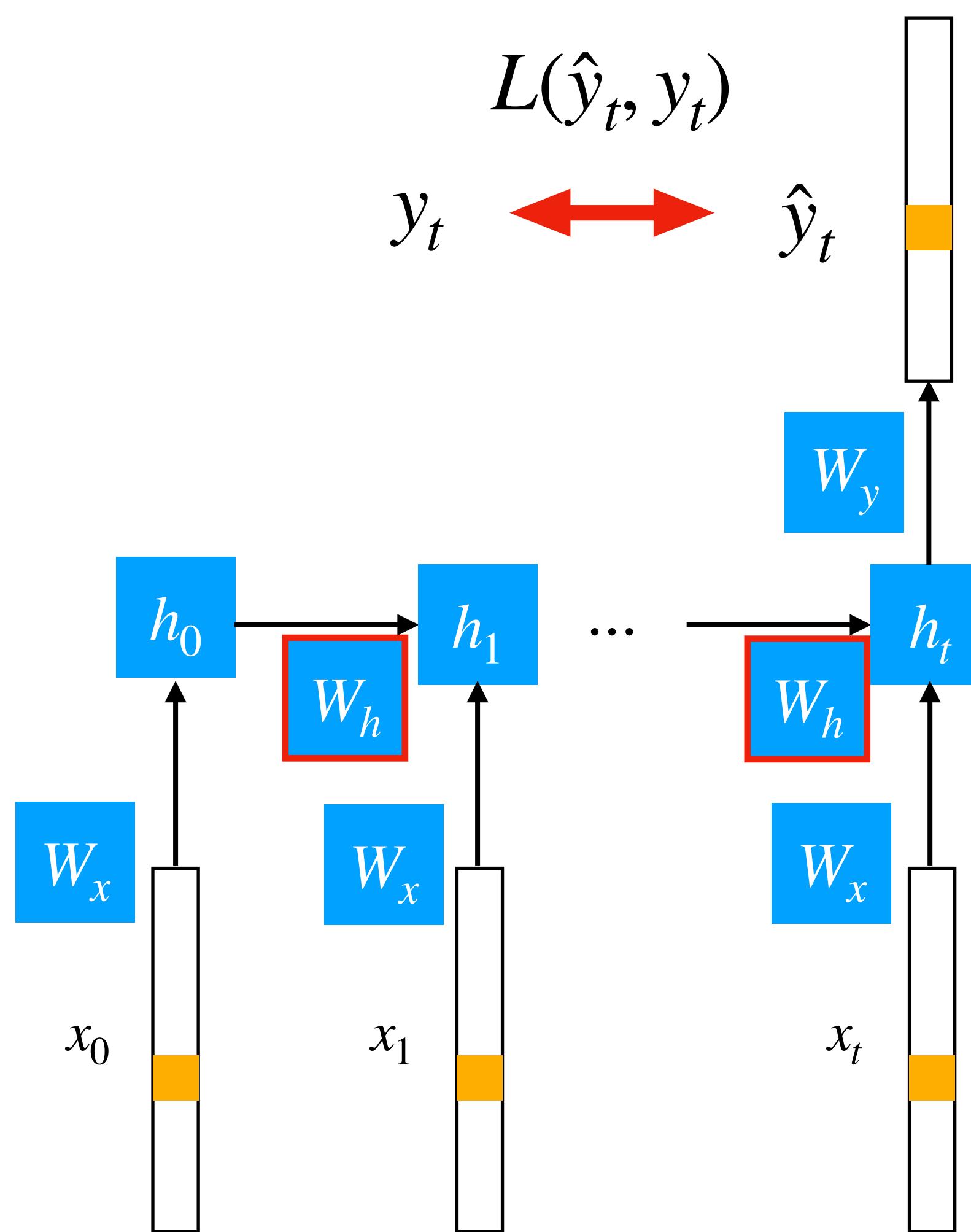


Computation graph은 다음과 같다:

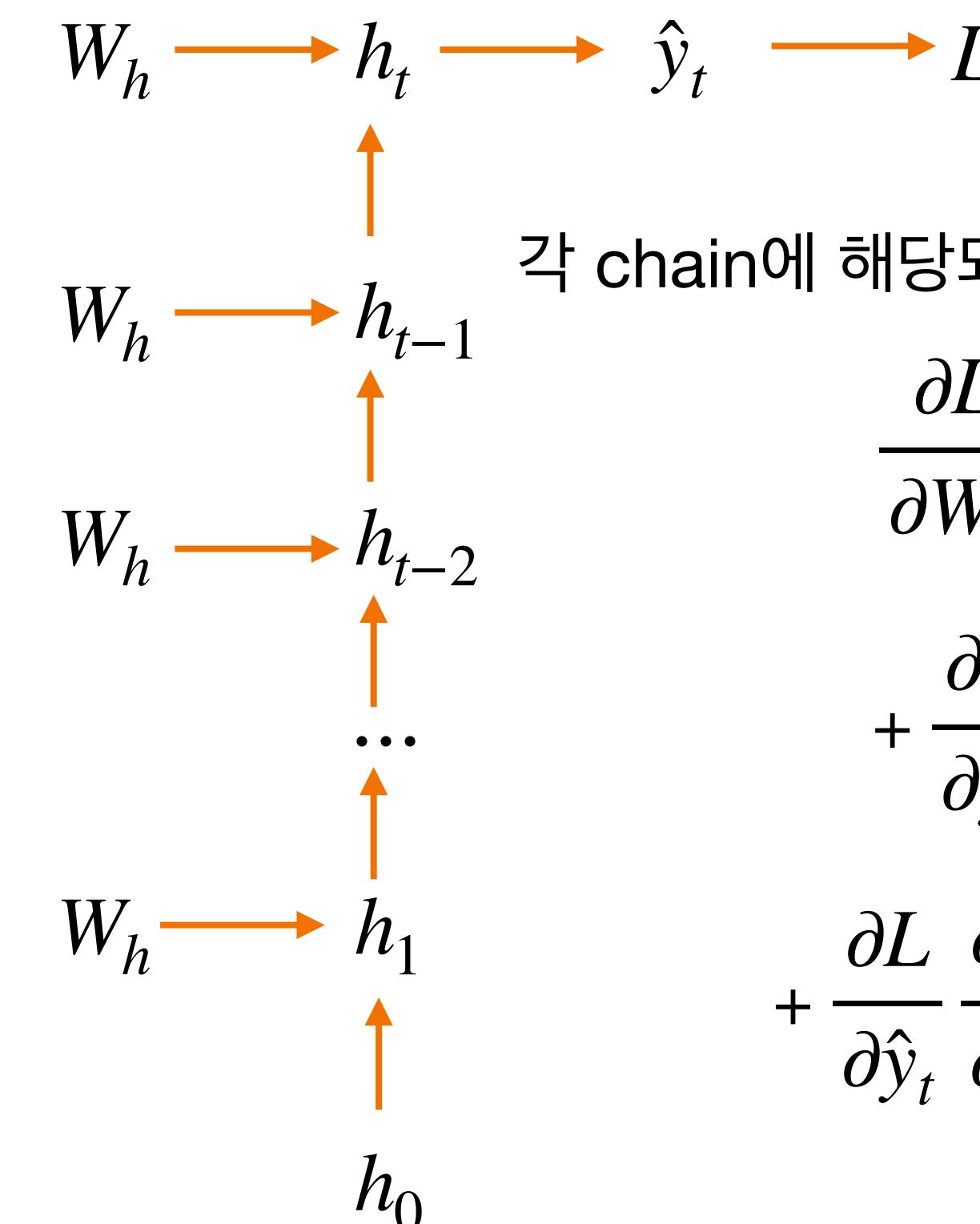


Recurrent Neural Networks

Differentiating RNN



Computation graph은 다음과 같다:

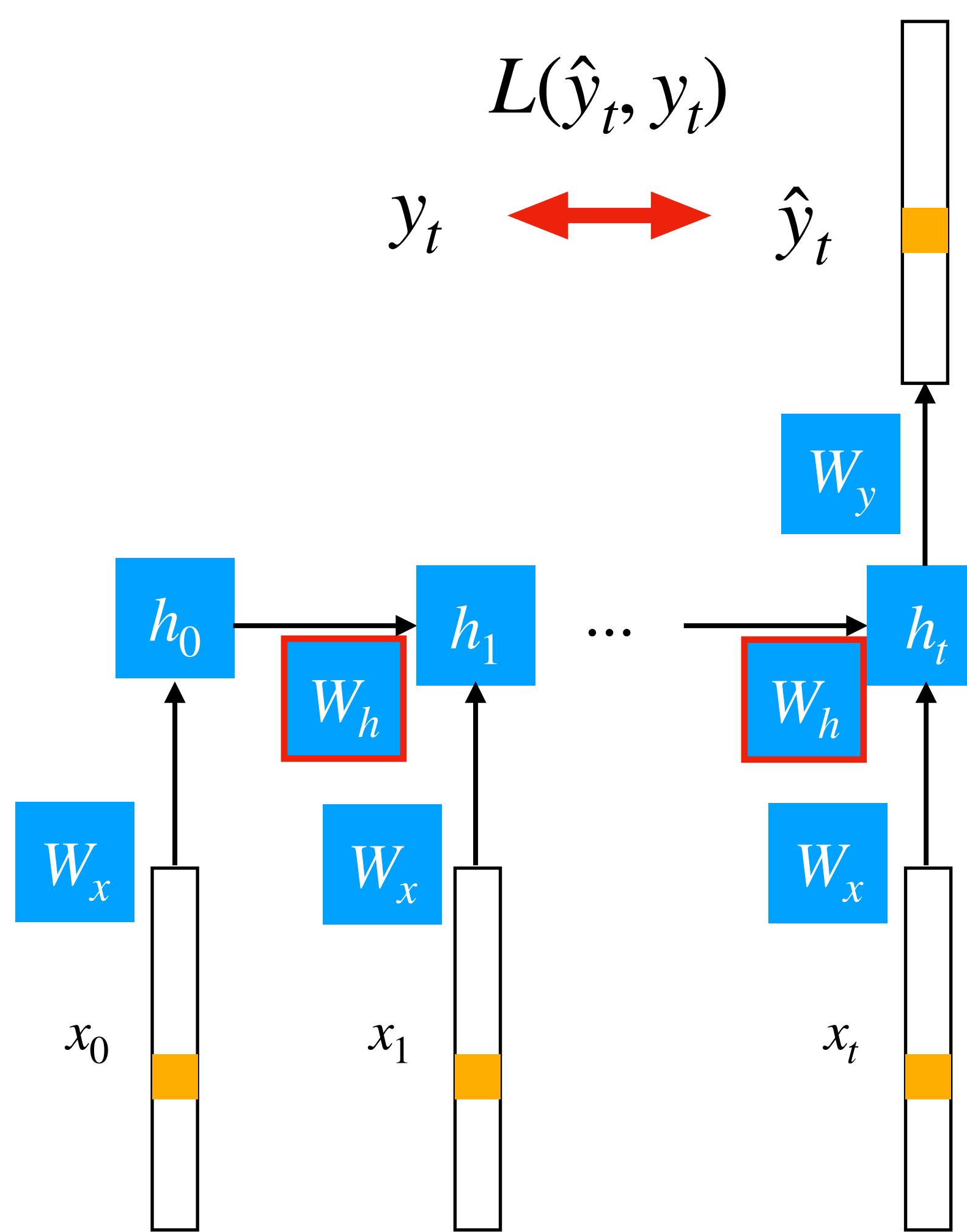


각 chain에 해당되는 derivative를 모두 다 더하면:

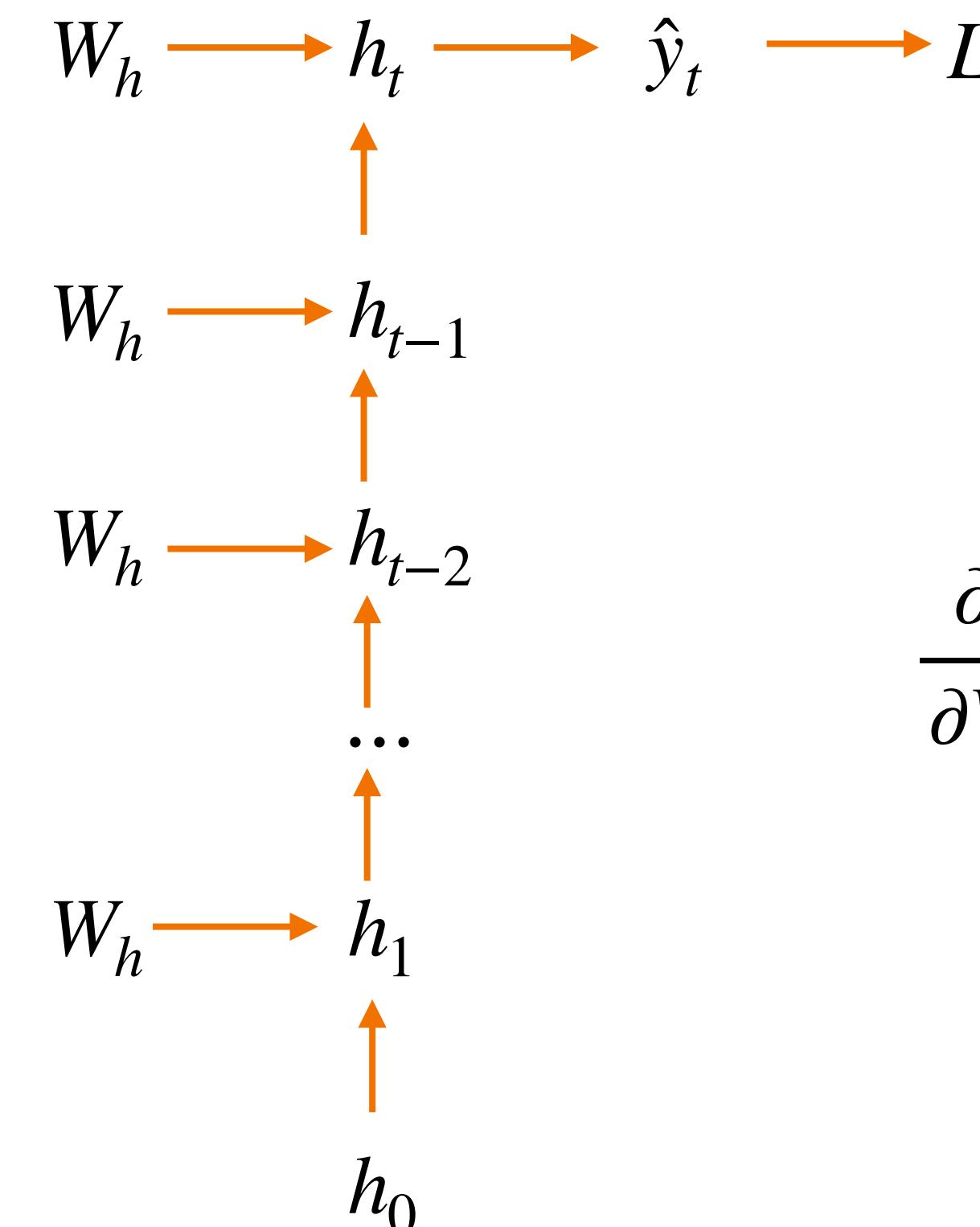
$$\begin{aligned} \frac{\partial L}{\partial W_h} &= \frac{\partial L}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial h_T} \frac{\partial h_T}{\partial W_h} && \text{1번째 Chain} \\ &+ \frac{\partial L}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial W_h} && \text{2번째 Chain} \\ &+ \frac{\partial L}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial h_{T-2}} \frac{\partial h_{T-2}}{\partial W_h} && \text{3번째 Chain} \\ &\dots \\ &+ \frac{\partial L}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial h_T} \frac{\partial h_T}{\partial h_{T-1}} \dots \frac{\partial h_1}{\partial W_h} && \text{t번째 Chain} \end{aligned}$$

Recurrent Neural Networks

Differentiating RNN



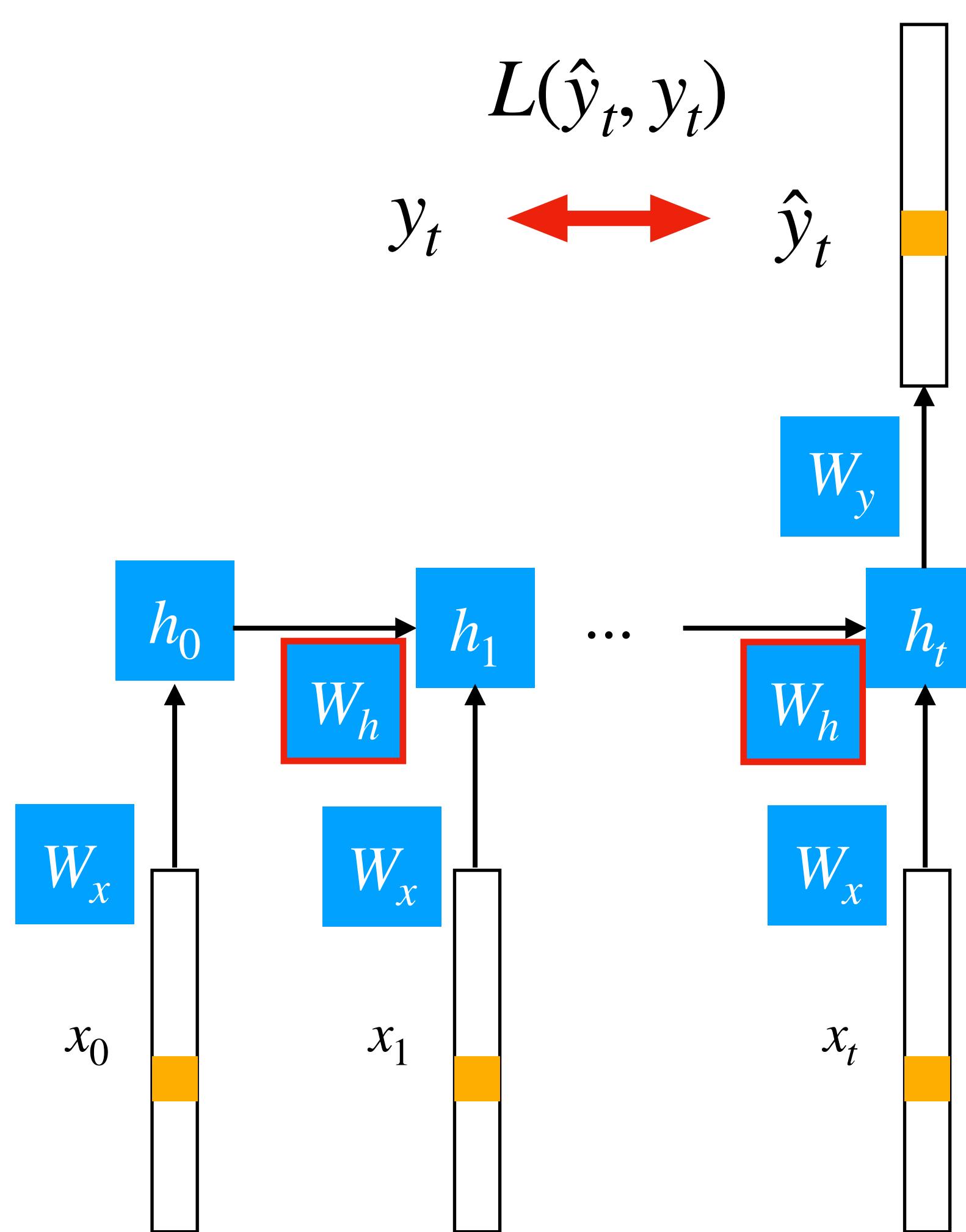
Computation graph은 다음과 같다:



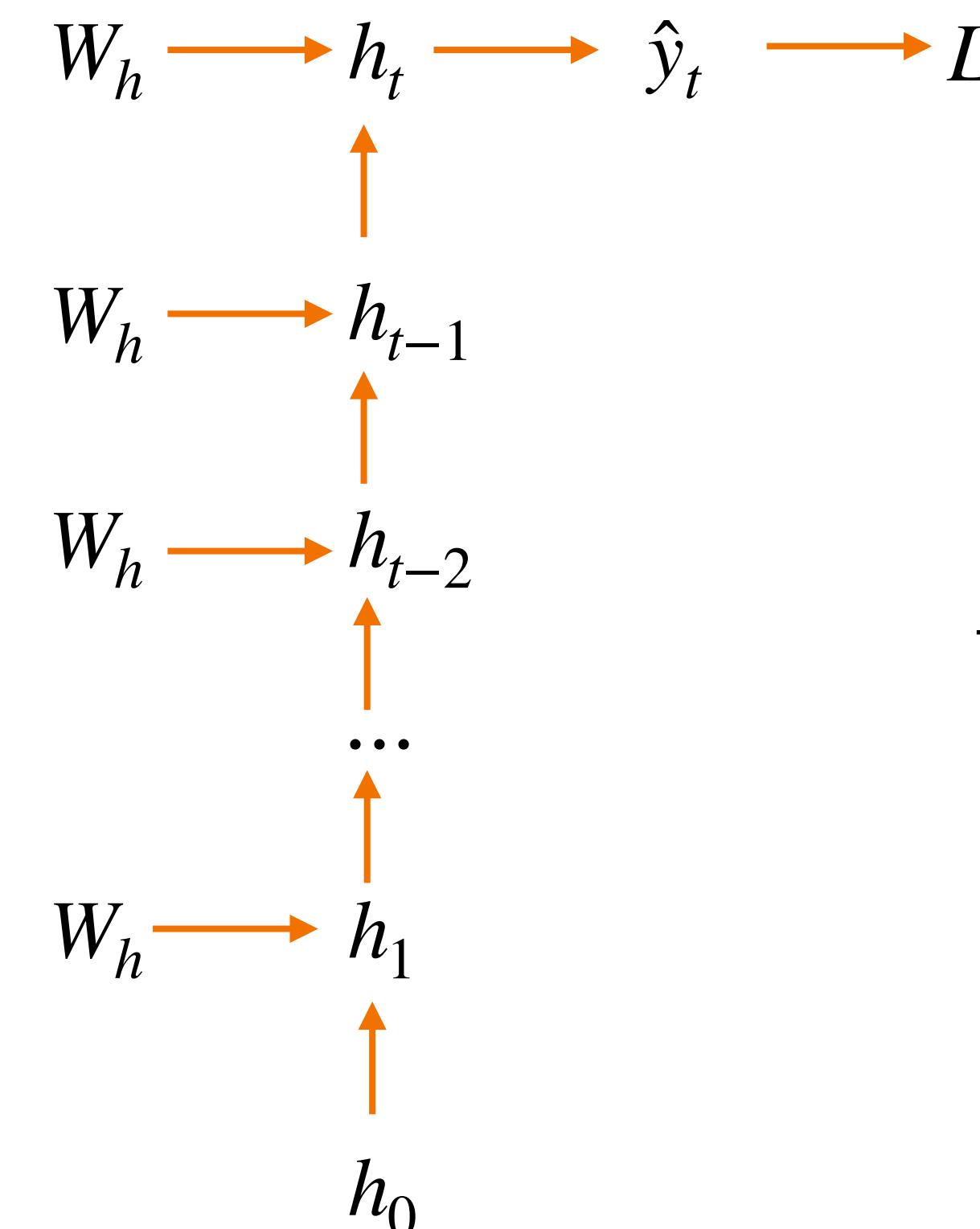
$$\frac{\partial L}{\partial W_h} = \sum_{i=1}^t \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \dots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial W_h}$$

Recurrent Neural Networks

Differentiating RNN



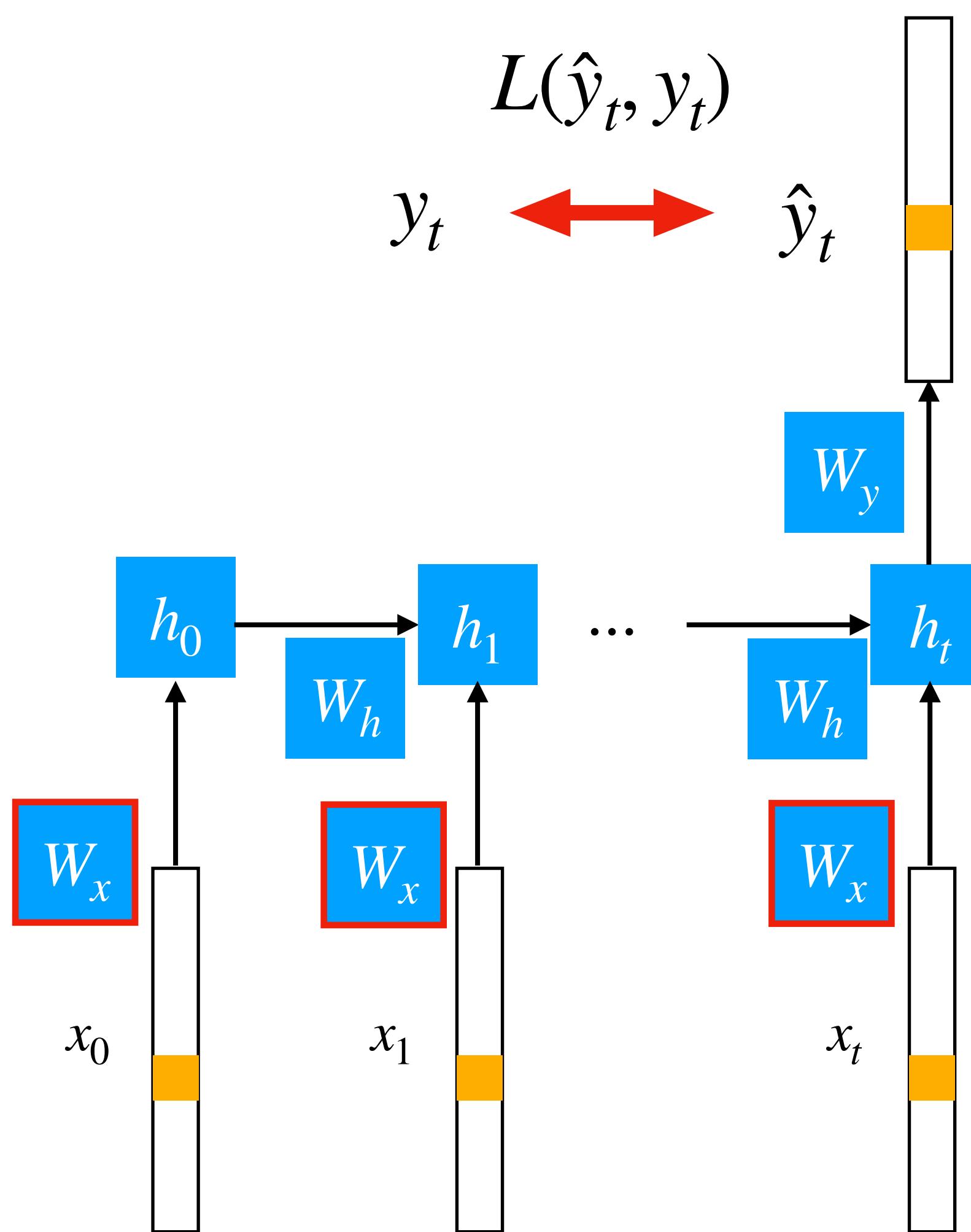
Computation graph은 다음과 같다:



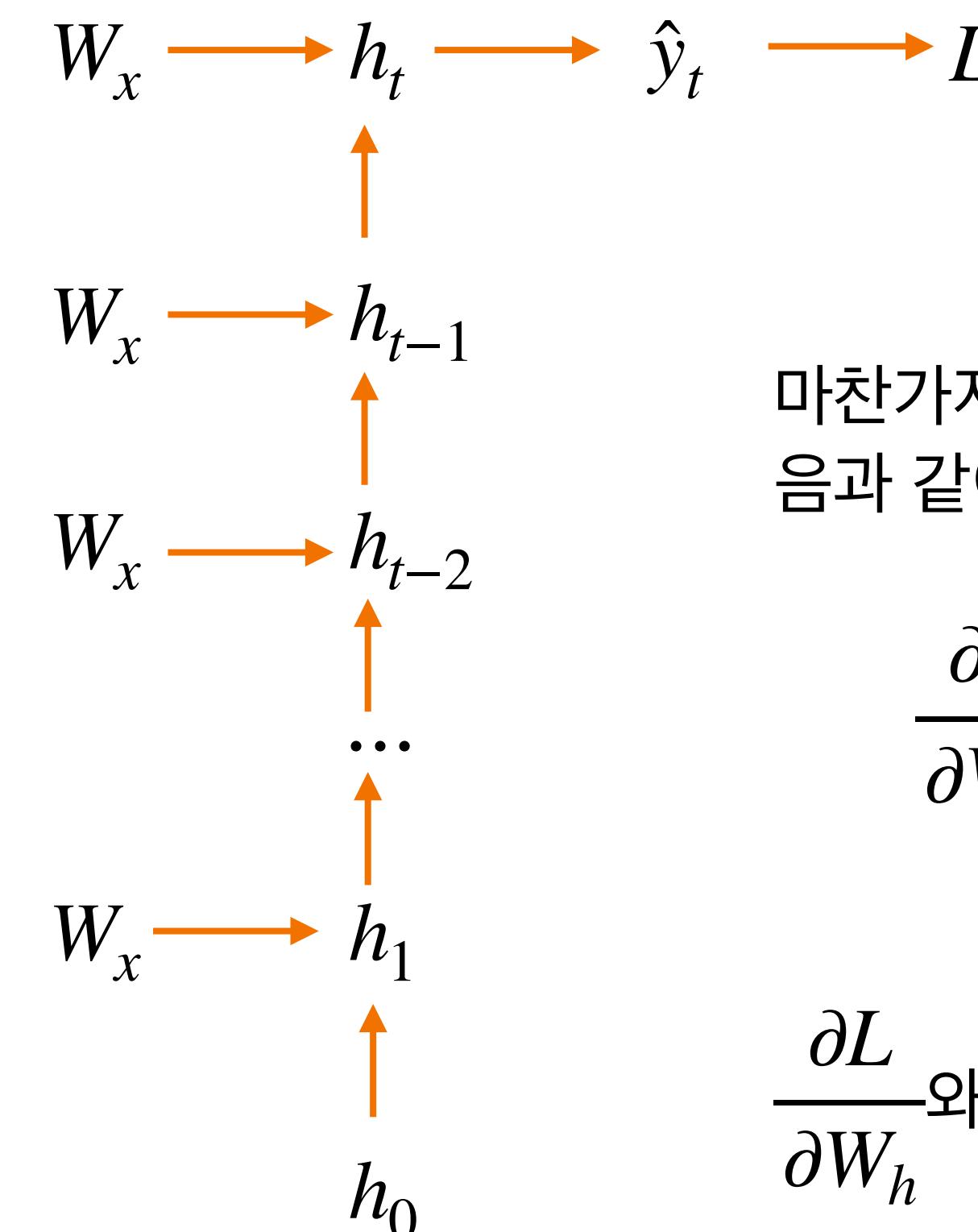
그렇다면 $\frac{\partial L}{\partial W_x}$ 은 어떻게 구할까?

Recurrent Neural Networks

Differentiating RNN



Computation graph은 다음과 같다:



마찬가지로 Computational Graph을 참고하면 다음과 같이 구할 수 있다!

$$\frac{\partial L}{\partial W_x} = \sum_{i=1}^t \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \dots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial W_x}$$

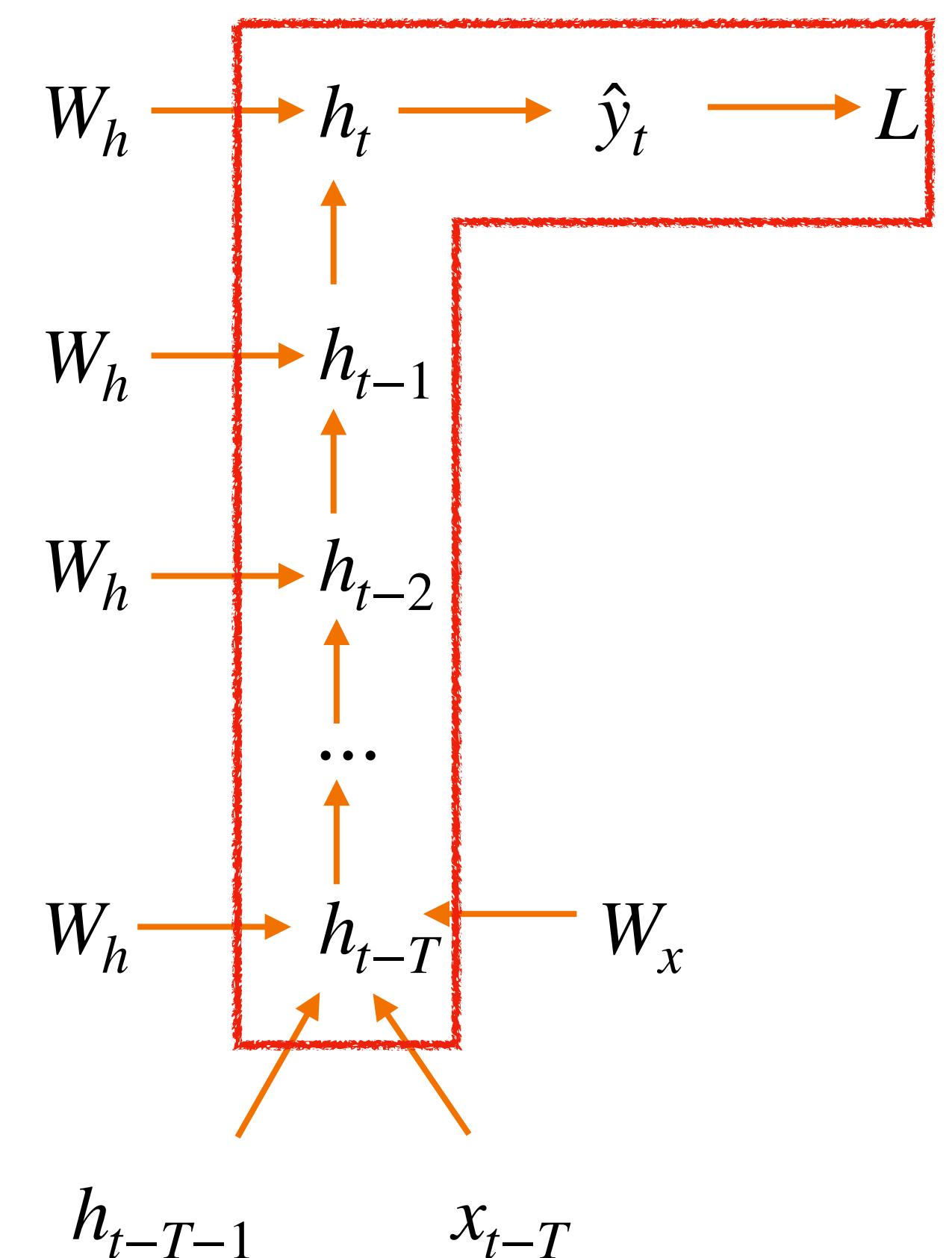
$\frac{\partial L}{\partial W_h}$ 와 유사하다!

15-4. RNN의 Vanishing Gradient 문제

Recurrent Neural Networks

Vanishing Gradient Problem in RNN

Sequence 길이가 길어질수록 RNN에서는 어떤 문제점이 발생하는가?



Recurrent Neural Networks

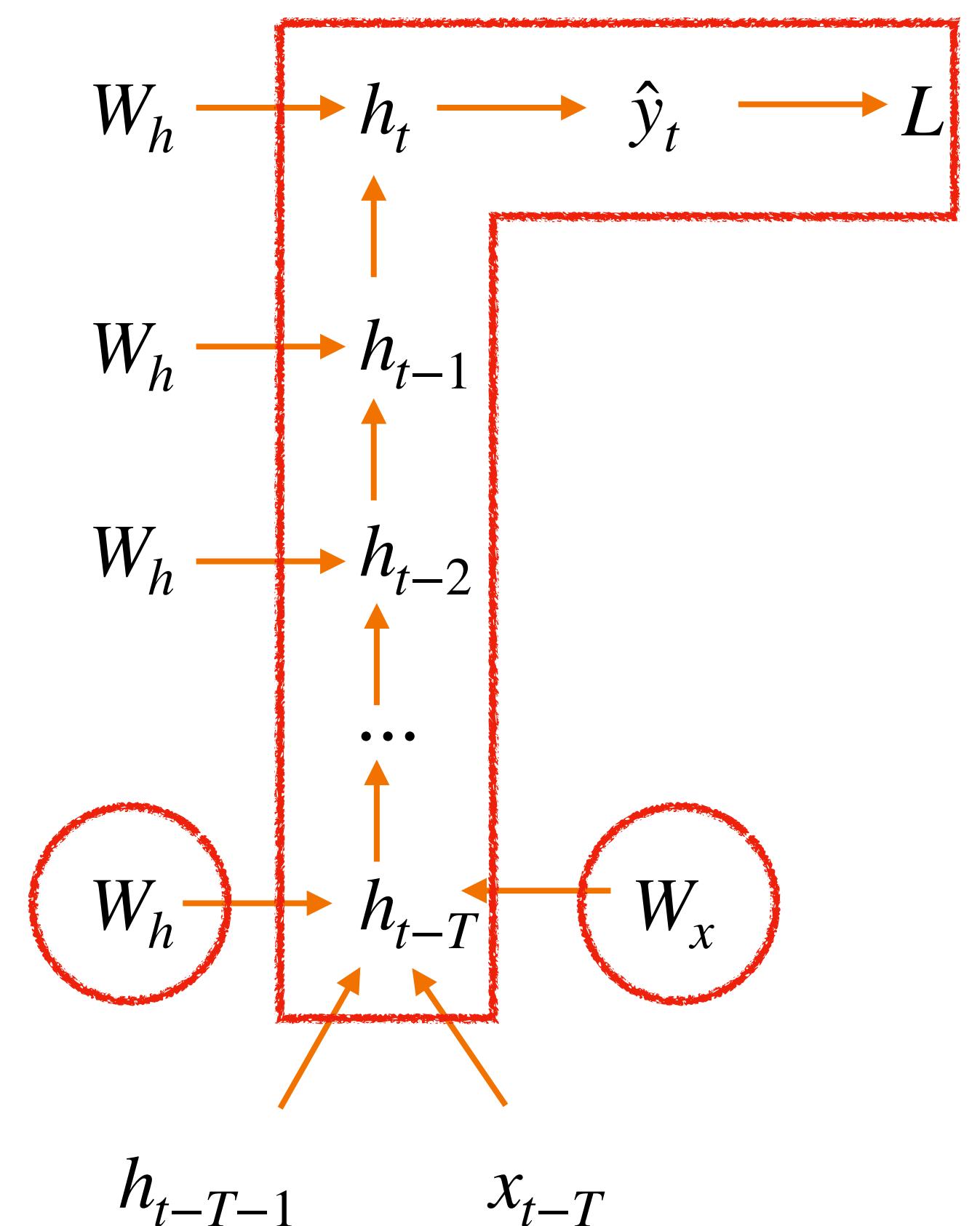
Vanishing Gradient Problem in RNN

Sequence 길이가 길어질수록 RNN에서는 어떤 문제점이 발생하는가?

$t = T$ 번째 입력으로부터 비롯되는:

$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

여기서 $W = W_h$ or W_x



Recurrent Neural Networks

Vanishing Gradient Problem in RNN

Sequence 길이가 길어질수록 RNN에서는 어떤 문제점이 발생하는가?

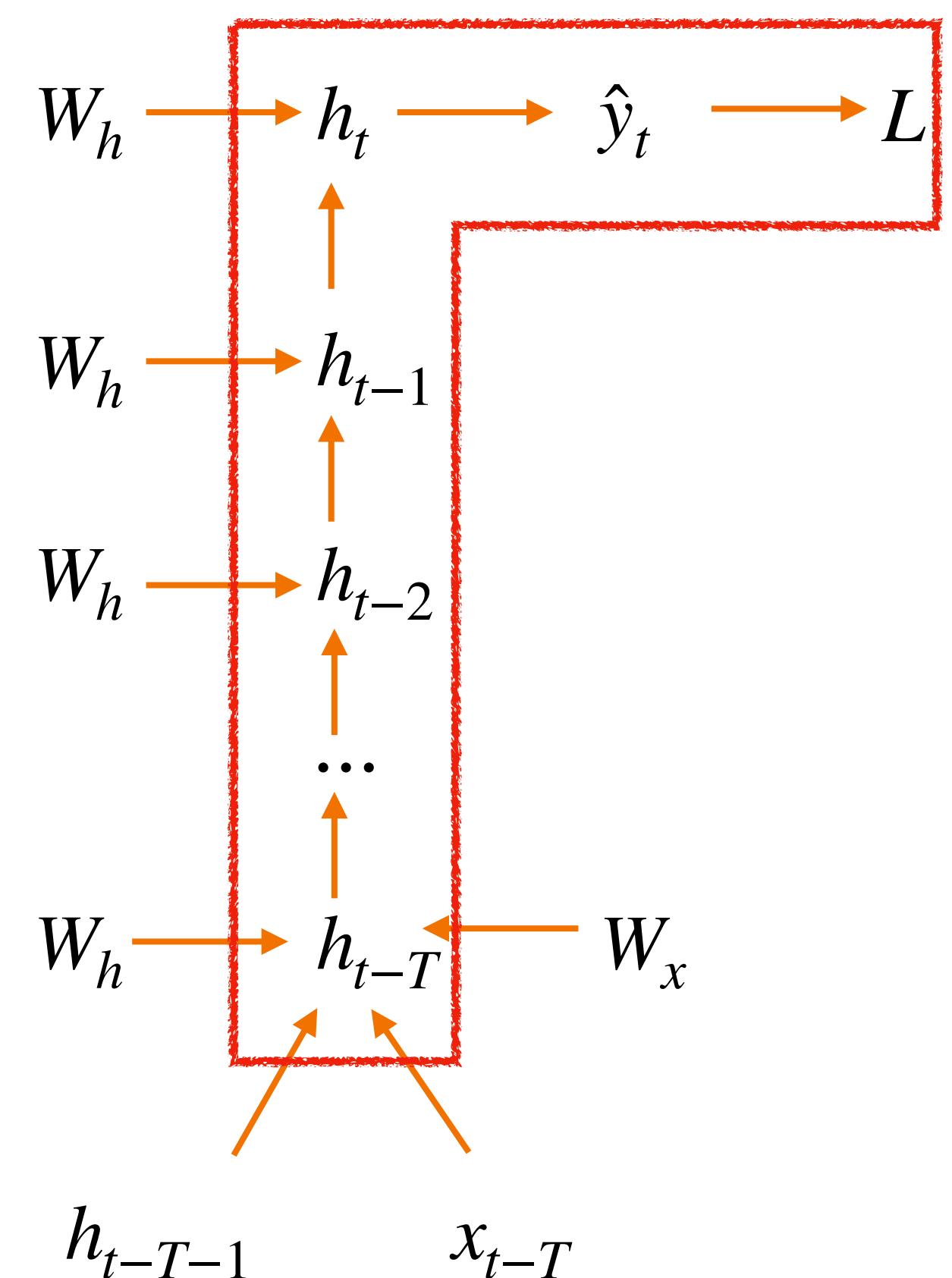
$t = T$ 번째 입력으로부터 비롯되는:

$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

여기서 $W = W_h$ or W_x

참고로 RNN에서 사용되는 activation function은 tanh function이다.

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$



Recurrent Neural Networks

Vanishing Gradient Problem in RNN

Sequence 길이가 길어질수록 RNN에서는 어떤 문제점이 발생하는가?

$t = T$ 번째 입력으로부터 비롯되는:

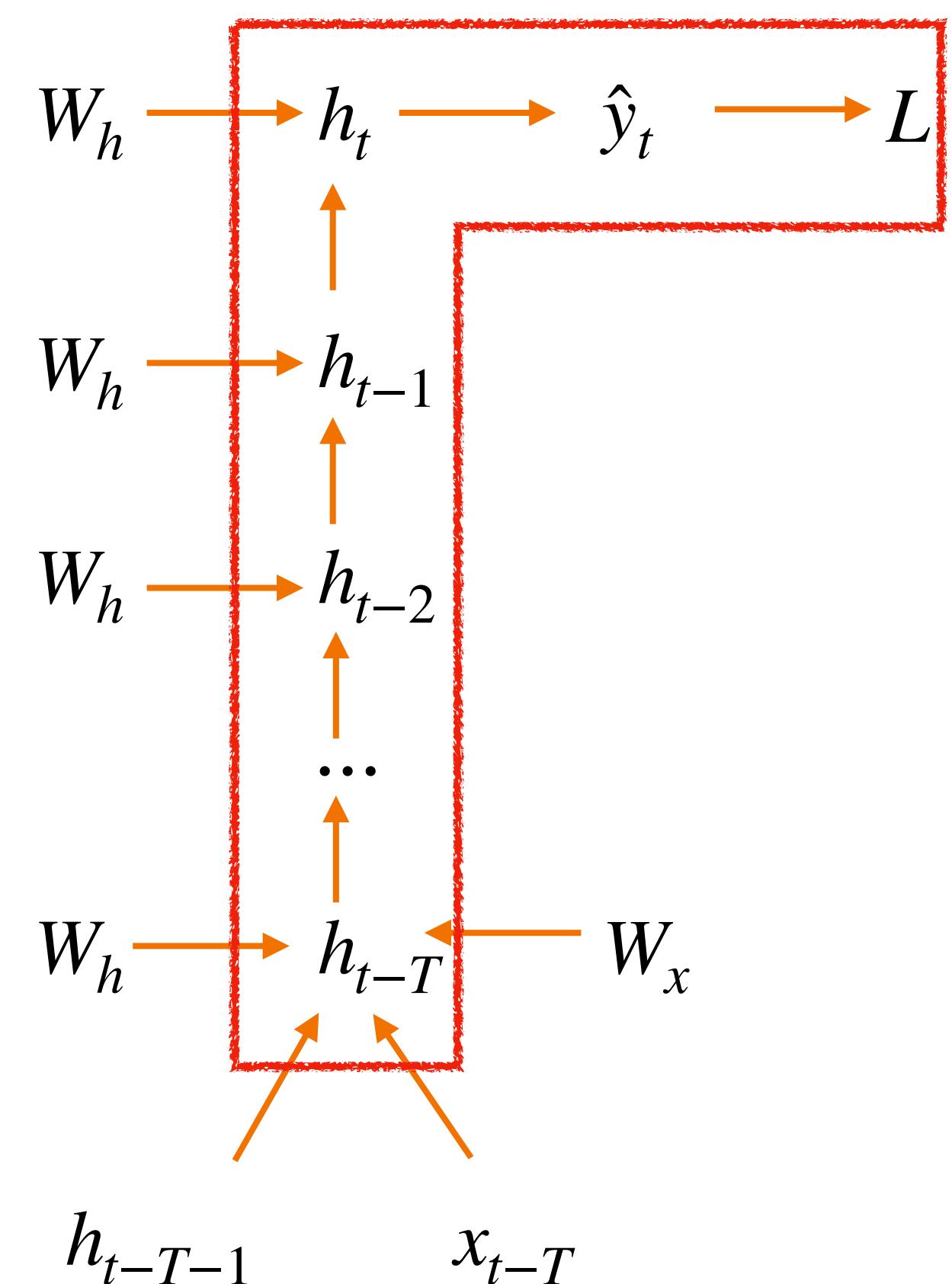
$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

$\frac{\partial h_t}{\partial h_{t-T}}$

여기서 $W = W_h$ or W_x

참고로 RNN에서 사용되는 activation function은 tanh function이다.

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$



Recurrent Neural Networks

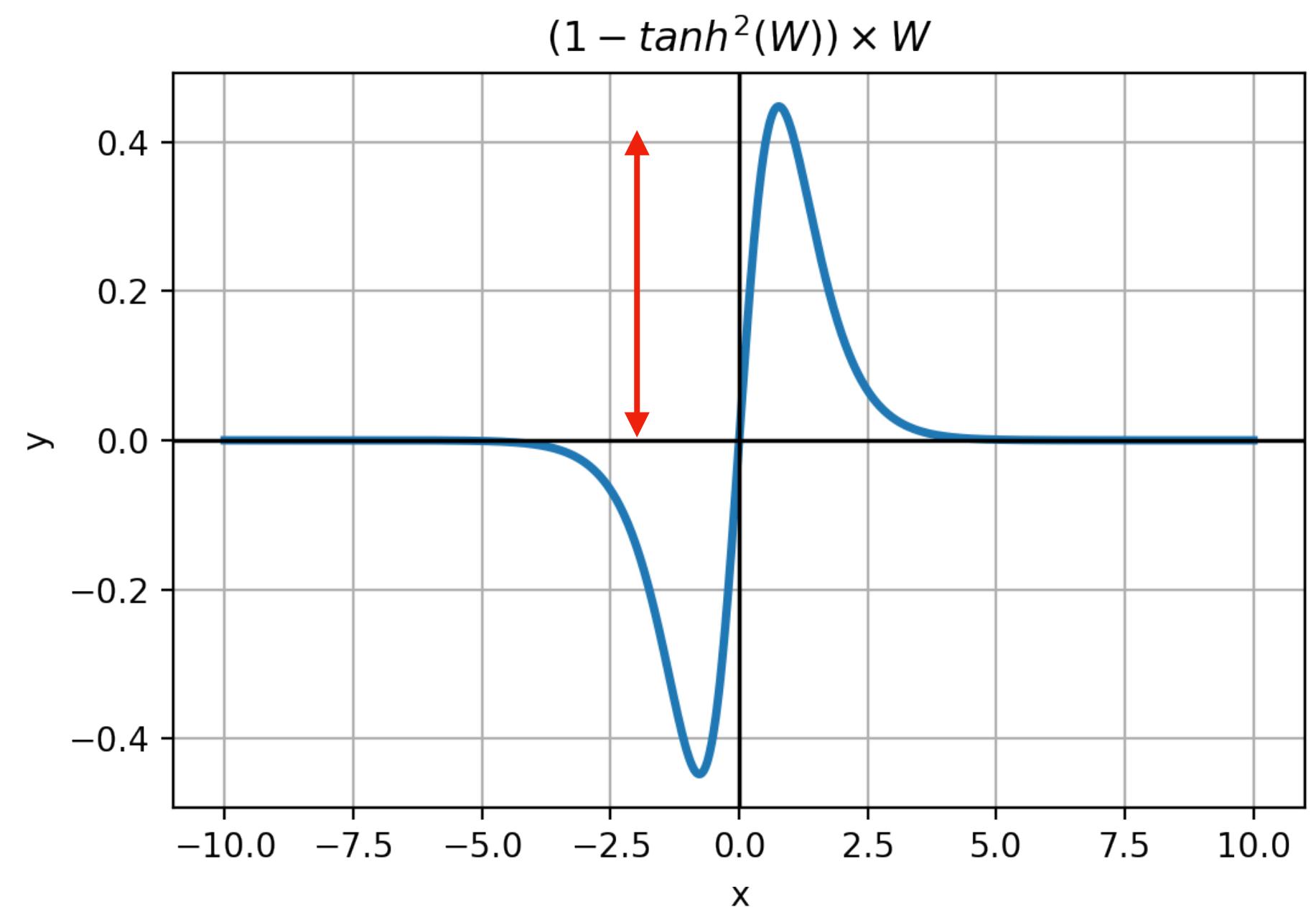
Vanishing Gradient Problem in RNN

참고로 RNN에서 사용되는 activation function은 tanh function이다.

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

이에 따라:

$$\begin{aligned} \frac{\partial h_t}{\partial h_{t-T}} &= \left(1 - \tanh^2(W_h h_{t-1} + W_x x_t) \right) W_h \frac{\partial h_{t-1}}{\partial h_{t-T}} \\ &= \frac{\partial h_t}{\partial h_{t-1}} \quad \text{magnitude가 1보다 작다!} \end{aligned}$$



Recurrent Neural Networks

Vanishing Gradient Problem in RNN

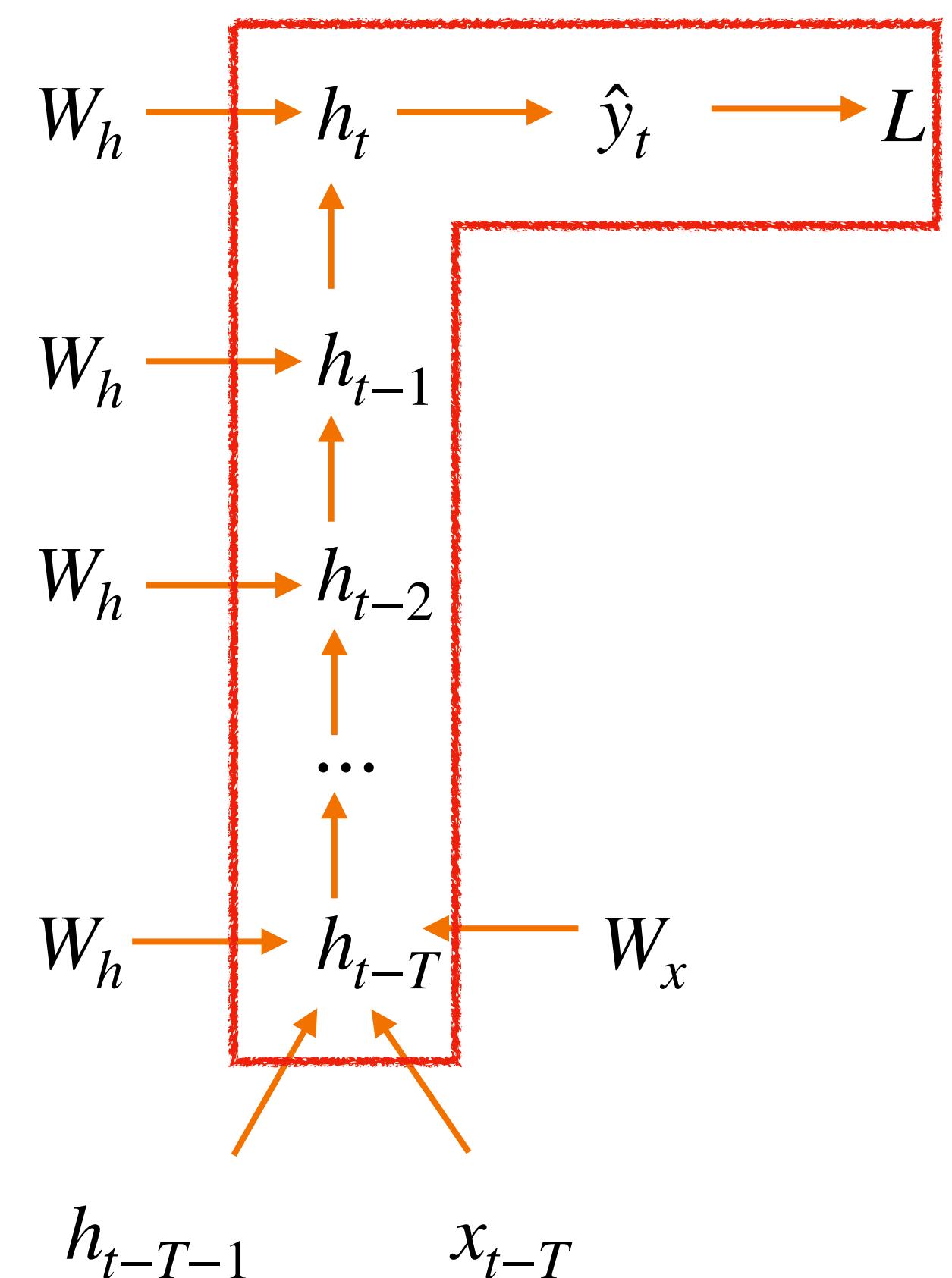
Sequence 길이가 길어질수록 RNN에서는 어떤 문제점이 발생하는가?

$t = T$ 번째 입력으로부터 비롯되는 partial derivative은:

$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

magnitude가 1보다 작다!

따라서 T 가 크면 클수록 1보다 작은 $\frac{\partial h_{i+1}}{\partial h_i}$ 을 여러 번 곱하게 되고 위 식은 0에 수렴하게 된다!



Recurrent Neural Networks

Vanishing Gradient Problem in RNN

Sequence 길이가 길어질수록 RNN에서는 어떤 문제점이 발생하는가?

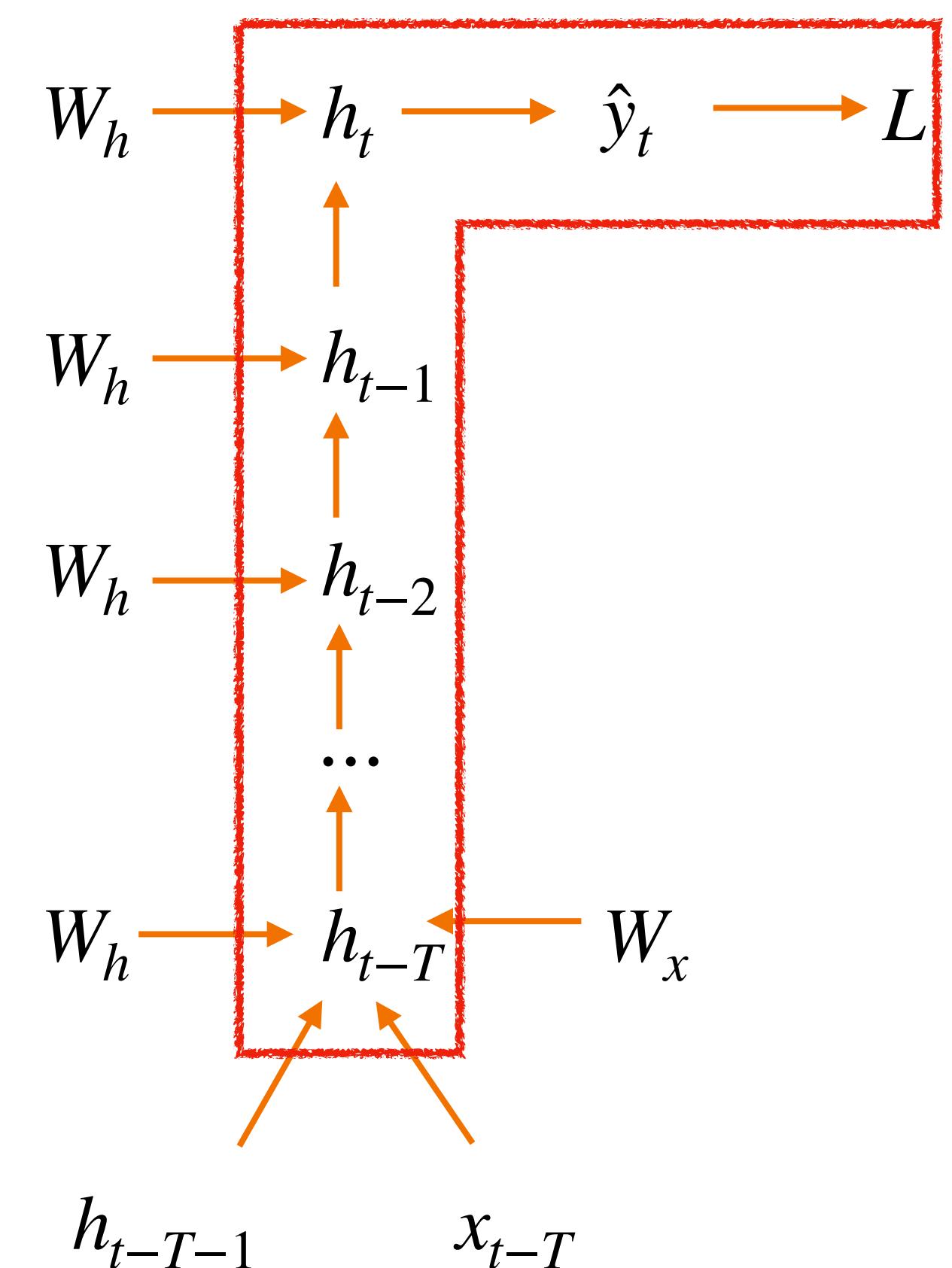
$t = T$ 번째 입력으로부터 비롯되는 partial derivative은:

$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

magnitude가 1보다 작다!

따라서 T 가 크면 클수록 1보다 작은 $\frac{\partial h_{i+1}}{\partial h_i}$ 을 여러 번 곱하게 되고 위 식은 0에 수렴하게 된다!

즉, Vanishing Gradient의 문제 발생



Recurrent Neural Networks

Vanishing Gradient Problem in RNN

Vanishing Gradient의 문제점을 해결하기 위해서는 어떤 해결방안이 있을까?

Long-Short Term Memory (LSTM), Gated Recurrent Units (GRU)

먼저 LSTM부터 살펴보자.

15-5. Long Short Term Memory (LSTM)

Recurrent Neural Networks

LSTM

LSTM의 구성 요소

- c_t : cell state
- h_t : hidden state
- $f(x, h_{t-1})$: forget gate (0~1 사이의 값)
- $o(x, h_{t-1})$: output gate (0~1 사이의 값)
- $i(x, h_{t-1})$: input gate (0~1 사이의 값)
- h'_t : candidate hidden state
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

LSTM의 구성 요소

- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
- $f(x, h_{t-1})$: forget gate
- $o(x, h_{t-1})$: output gate
- $i(x, h_{t-1})$: input gate
- h'_t : candidate hidden state
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

Copyright©2023. Acadential. All rights reserved.

LSTM의 구성 요소

- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
(time step t에서의 예측값 \hat{y}_t 을 출력하기 위해 필요한 특징/정보를 내포한 hidden state)
- $f(x, h_{t-1})$: forget gate
- $o(x, h_{t-1})$: output gate
- $i(x, h_{t-1})$: input gate
- h'_t : candidate hidden state
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

Copyright©2023. Acadential. All rights reserved.

LSTM의 구성 요소

- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
(time step t에서의 예측값 \hat{y}_t 을 출력하기 위해 필요한 특징/정보를 내포한 hidden state)
- $f(x, h_{t-1})$: forget gate
(c_{t-1} 을 얼마나 c_t 에 반영할지 조절하는 gate)
- $o(x, h_{t-1})$: output gate
- $i(x, h_{t-1})$: input gate
- h'_t : candidate hidden state
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

Copyright©2023. Acadential. All rights reserved.

LSTM의 구성 요소

- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
(time step t에서의 예측값 \hat{y}_t 을 출력하기 위해 필요한 특징/정보를 내포한 hidden state)
- $f(x, h_{t-1})$: forget gate
(c_{t-1} 을 얼마나 c_t 에 반영할지 조절하는 gate)
- $o(x_t, h_{t-1})$: output gate
(c_t 을 얼마나 h_t 에 반영할지 조절하는 gate)
- $i(x, h_{t-1})$: input gate
- h'_t : candidate hidden state
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

Copyright©2023. Acadential. All rights reserved.

LSTM의 구성 요소

- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
(time step t에서의 예측값 \hat{y}_t 을 출력하기 위해 필요한 특징/정보를 내포한 hidden state)
- $f(x, h_{t-1})$: forget gate
(c_{t-1} 을 얼마나 c_t 에 반영할지 조절하는 gate)
- $o(x_t, h_{t-1})$: output gate
(c_t 을 얼마나 h_t 에 반영할지 조절하는 gate)
- $i(x, h_{t-1})$: input gate
(c'_t 을 얼마나 c_t 에 반영할지 조절하는 gate)
- h'_t : candidate hidden state
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

Copyright©2023. Acadential. All rights reserved.

LSTM의 구성 요소

- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
(time step t에서의 예측값 \hat{y}_t 을 출력하기 위해 필요한 특징/정보를 내포한 hidden state)
- $f(x, h_{t-1})$: forget gate
(c_{t-1} 을 얼마나 c_t 에 반영할지 조절하는 gate)
- $o(x_t, h_{t-1})$: output gate
(c_t 을 얼마나 h_t 에 반영할지 조절하는 gate)
- $i(x, h_{t-1})$: input gate
(c'_t 을 얼마나 c_t 에 반영할지 조절하는 gate)
- h'_t : candidate hidden state
(현재 time step t에서의 후보 hidden state, h_t 에 반영됨)
- c'_t : candidate cell state

Recurrent Neural Networks

LSTM

Copyright©2023. Acadential. All rights reserved.

LSTM의 구성 요소

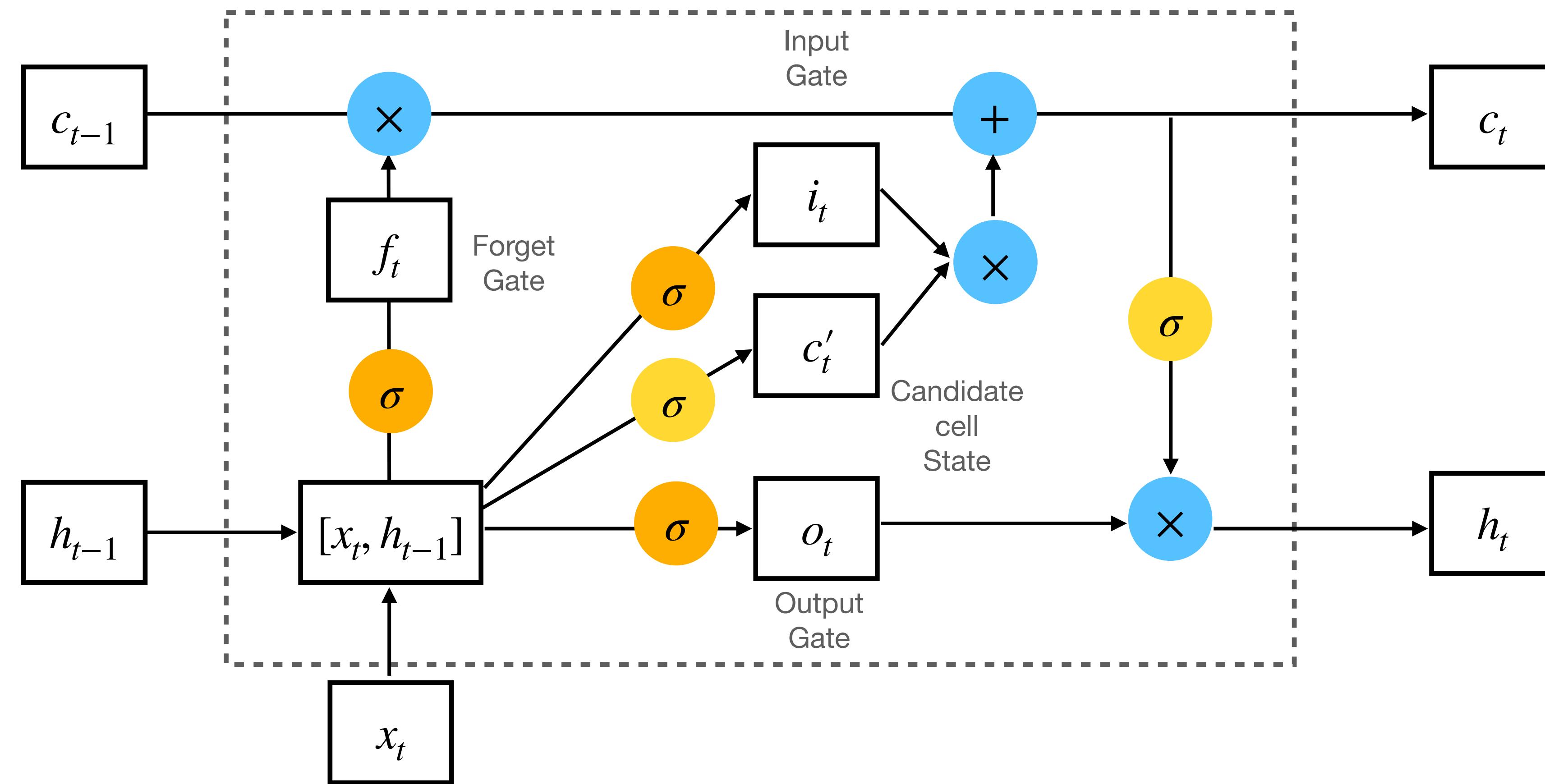
- c_t : cell state
(배열의 전체적인 맥락을 담은 state) (일종의 “memory”의 역할을 한다!)
- h_t : hidden state
(time step t에서의 예측값 \hat{y}_t 을 출력하기 위해 필요한 특징/정보를 내포한 hidden state)
- $f(x, h_{t-1})$: forget gate
(c_{t-1} 을 얼마나 c_t 에 반영할지 조절하는 gate)
- $o(x_t, h_{t-1})$: output gate
(c_t 을 얼마나 h_t 에 반영할지 조절하는 gate)
- $i(x, h_{t-1})$: input gate
(c'_t 을 얼마나 c_t 에 반영할지 조절하는 gate)
- h'_t : candidate hidden state
(현재 time step t에서의 후보 hidden state, h_t 에 반영됨)
- c'_t : candidate cell state
(현재 time step t에서의 후보 cell state, c_t 에 반영됨)

Recurrent Neural Networks

LSTM Architecture and Diagram

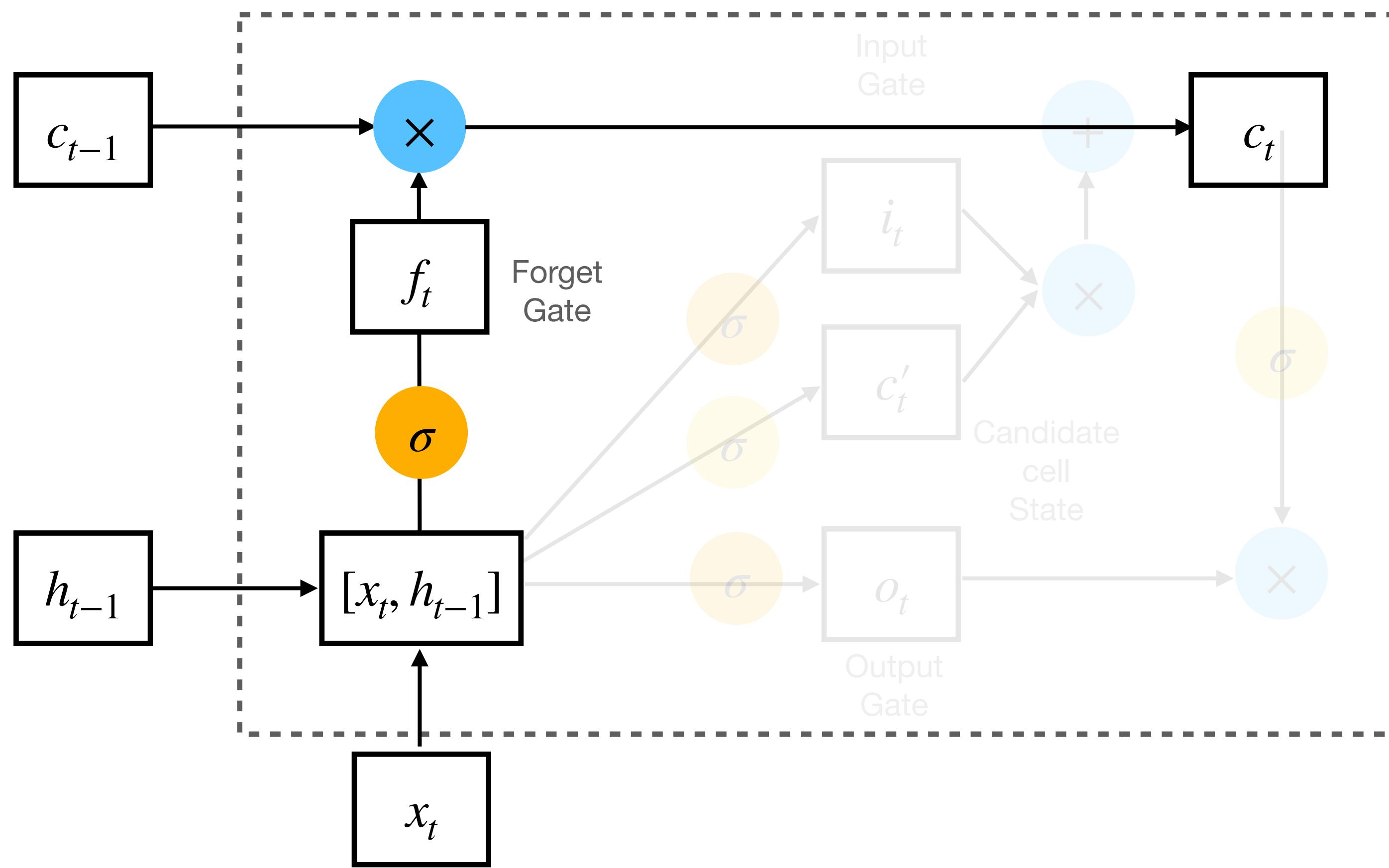
Copyright©2023. Acadential. All rights reserved.

- σ : tanh activation
- σ : sigmoid activation



Recurrent Neural Networks

LSTM: Forget Gate



Forget gate은 다음과 같이 계산된다:

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

이것은 다음 cell state을 구할 때 사용된다:

$$c_t = \sigma(f_t \cdot c_{t-1} + i_t \cdot c'_t)$$

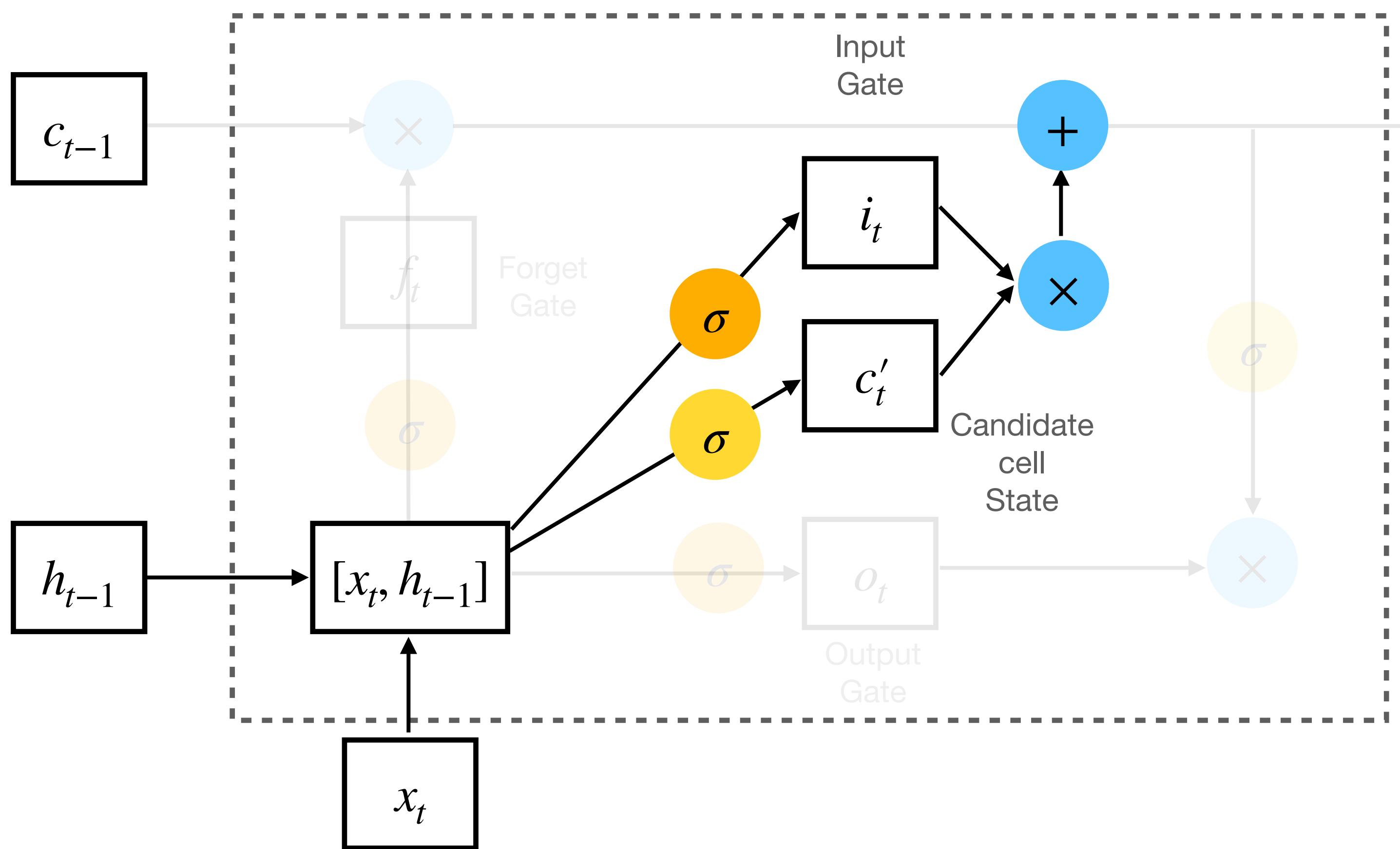
f_t 은 이전 cell state c_{t-1} 을 얼마나 c_t 에 반영할지 “조절”하는 값이다.

“ $f_t = 0$ ”: forget c_{t-1}

“ $f_t = 1$ ”: retain c_{t-1}

Recurrent Neural Networks

LSTM: Input Gate



Input gate은 다음과 같이 계산된다:

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

그리고 candidate cell state은 다음과 같이 계산된다:

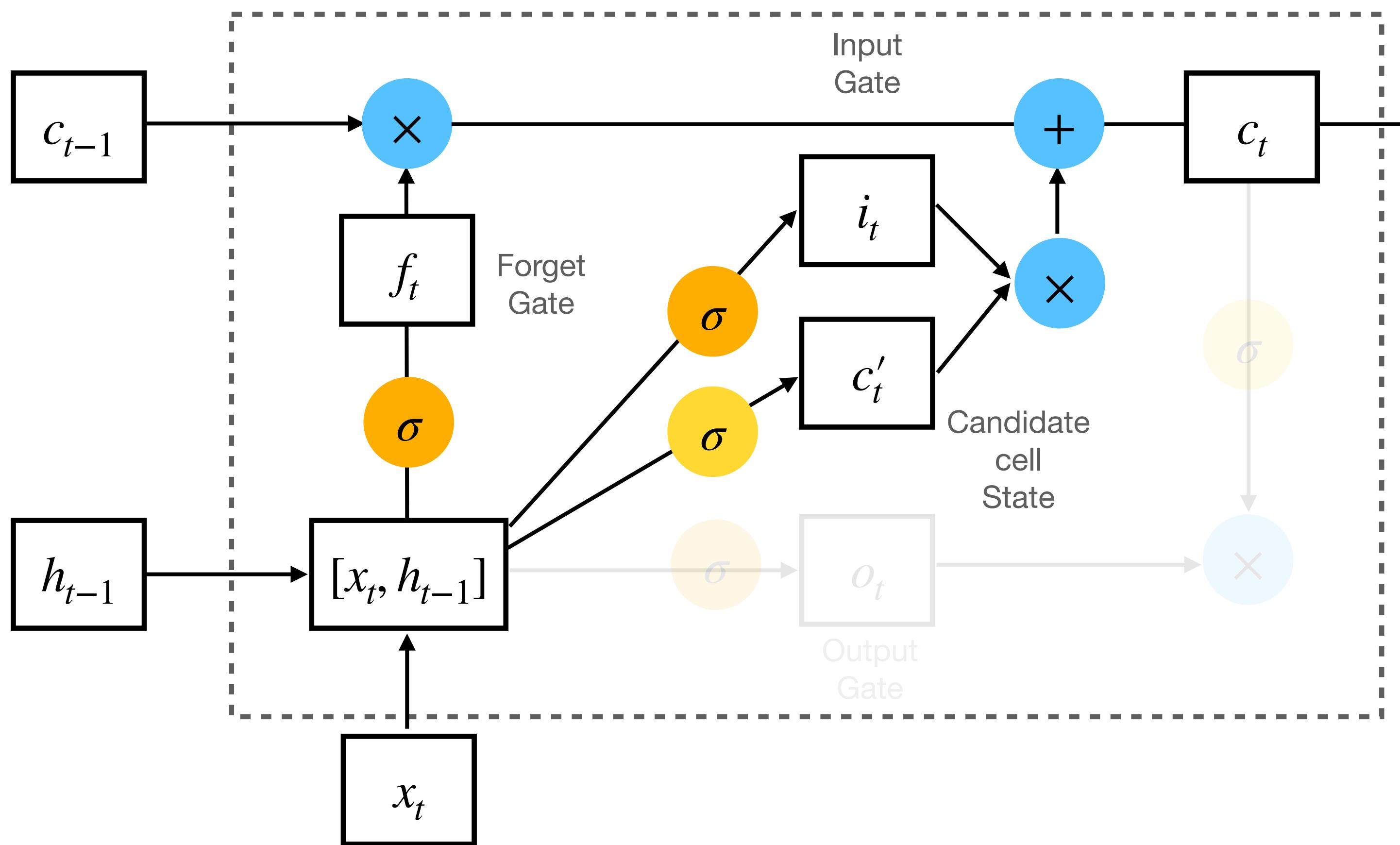
$$c'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

candidate cell state은 h_{t-1} 과 x_t 을 나타내는 feature이다!

Recurrent Neural Networks

LSTM: Input Gate

그리고 현재 cell state은:



$$c_t = \sigma(f_t \cdot c_{t-1} + i_t \cdot c'_t)$$

즉, i_t 은 얼마나 candidate cell state c'_t 을 c_t 에 반영할지 조절하는 값이다.

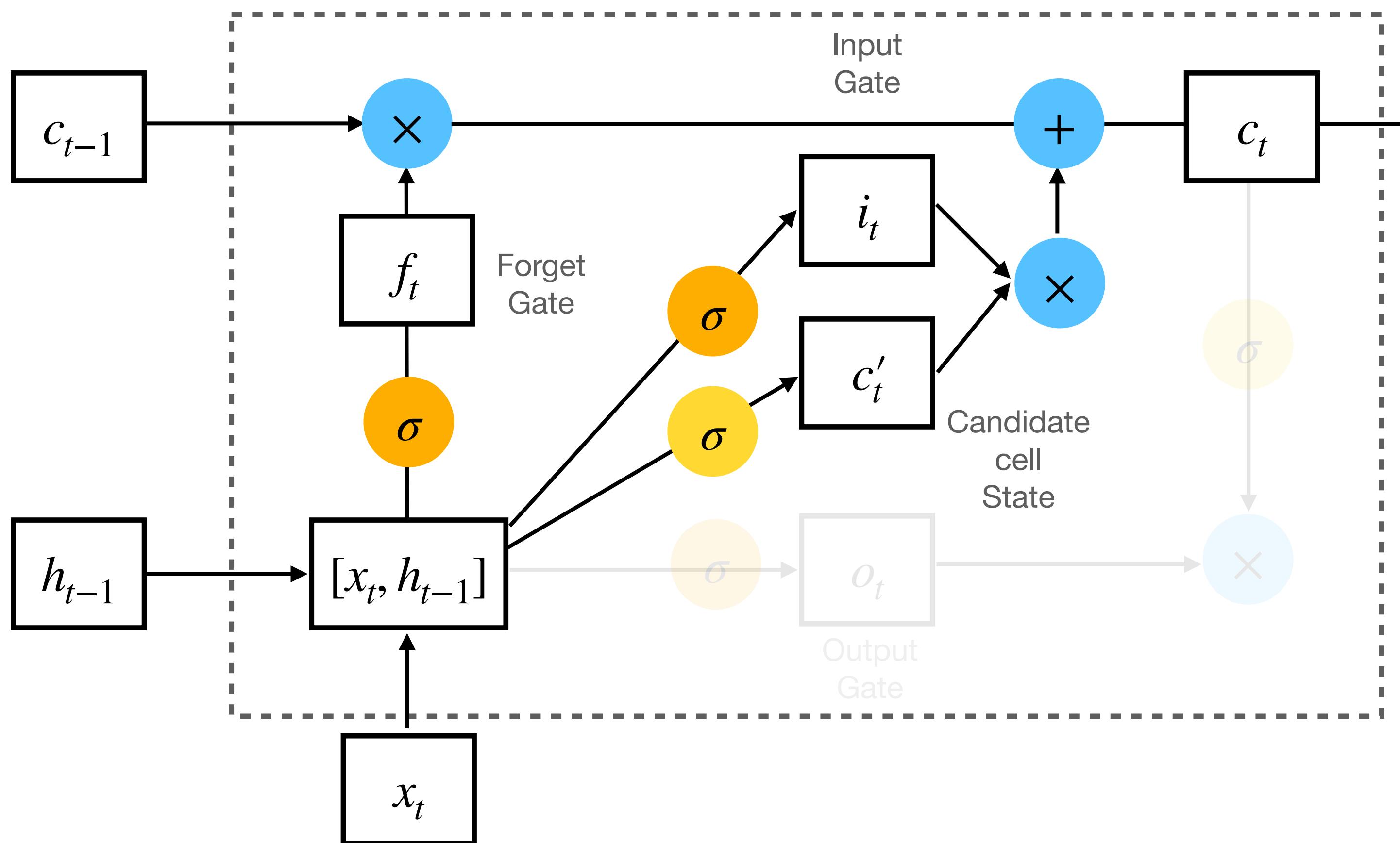
“ $i_t = 0$ ”: ignore current candidate cell state

“ $i_t = 1$ ”: retain current candidate cell state

Recurrent Neural Networks

LSTM: Input Gate

그리고 현재 cell state은:



$$c_t = \sigma(f_t \cdot c_{t-1} + i_t \cdot c'_t)$$

즉, i_t 은 얼마나 candidate cell state c'_t 을 c_t 에 반영할지 조절하는 값이다.

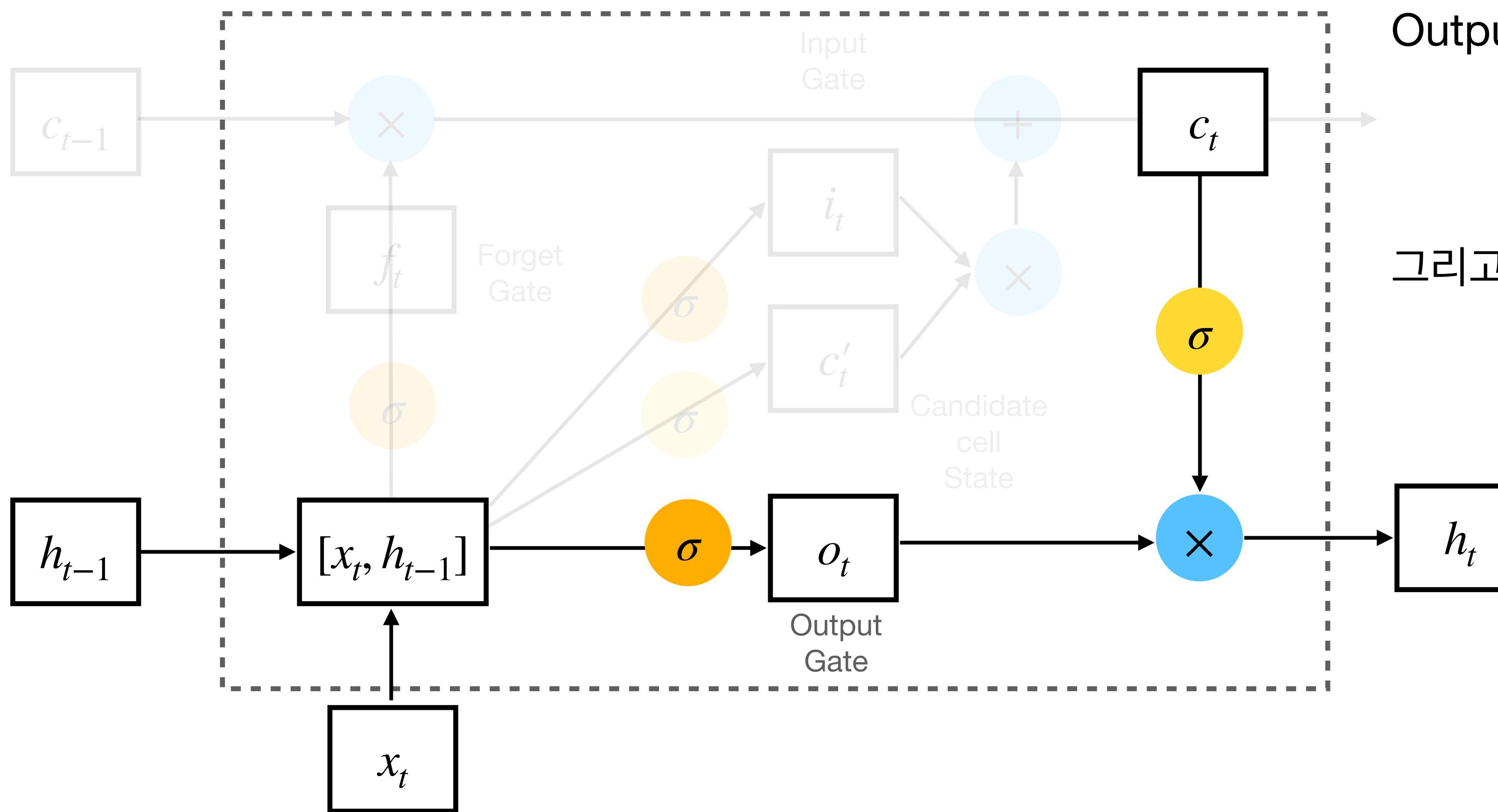
“ $i_t = 0$ ”: ignore current candidate cell state

“ $i_t = 1$ ”: retain current candidate cell state

f_t 가 클수록 이전 “memory”가 더 많이 반영되고, i_t 가 클수록 현재의 input x 와 이전 hidden state h_{t-1} 가 더 많이 반영되는 것이다!

Recurrent Neural Networks

LSTM: Output Gate



Output gate 은 다음과 같이 계산된다:

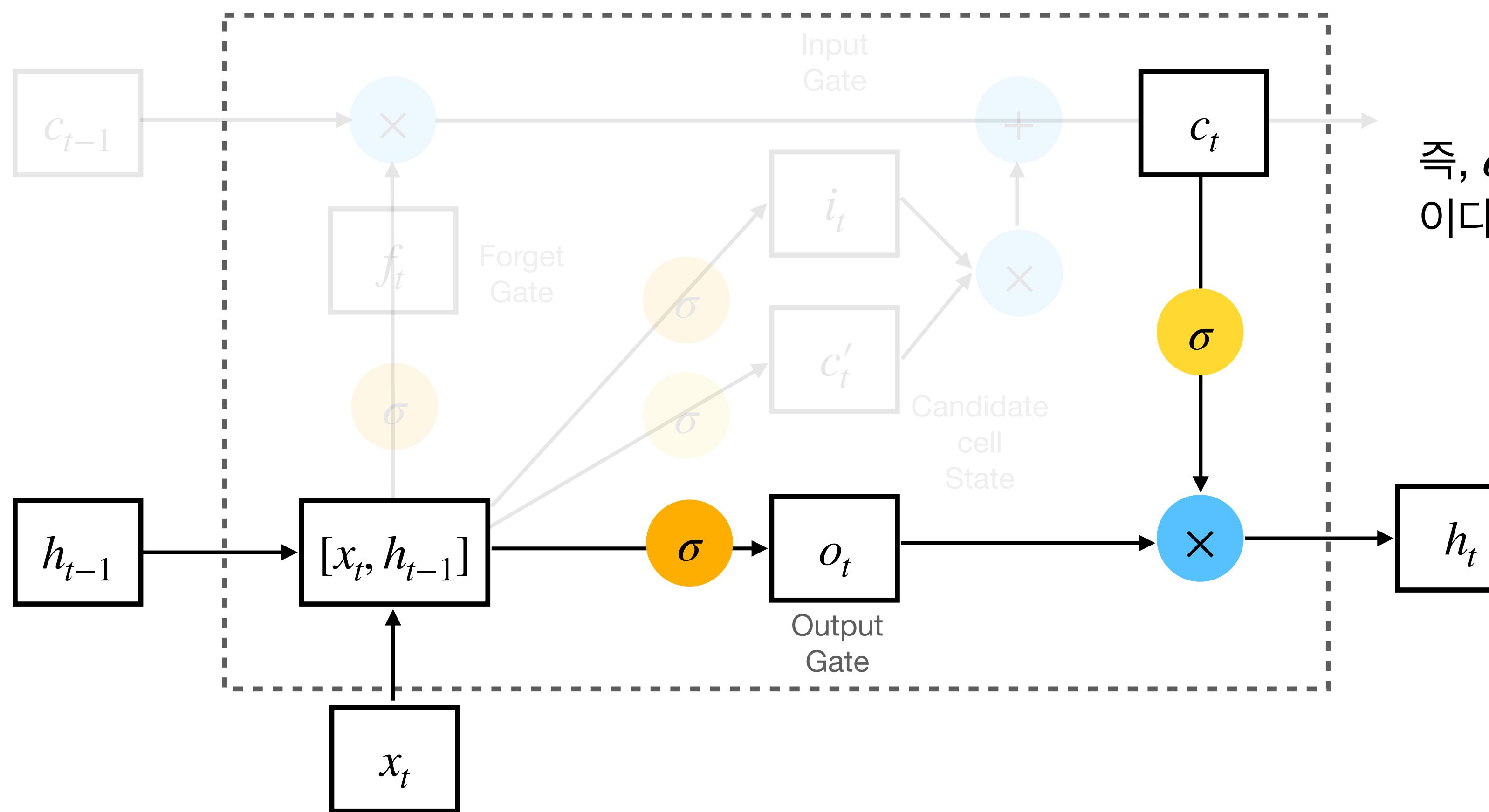
$$o_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o)$$

그리고 hidden state은 다음과 같다:

$$h_t = o_t \cdot \tanh(c_t)$$

Recurrent Neural Networks

LSTM: Output Gate

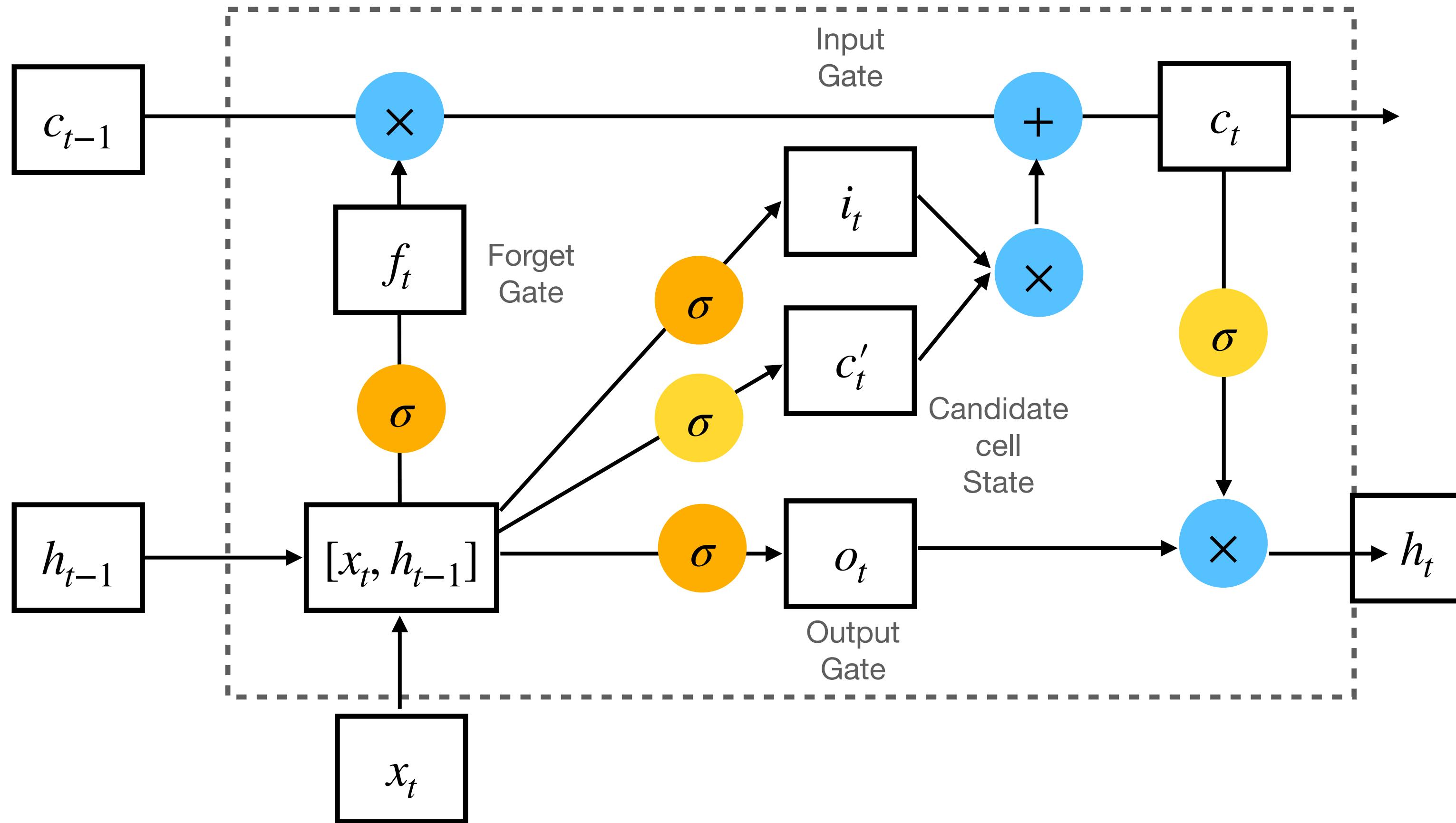


$$h_t = o_t \cdot \tanh(c_t)$$

즉, o_t 은 얼마나 c_t 을 h_t 에 반영할지 조절하는 값이다.

Recurrent Neural Networks

LSTM의 구성



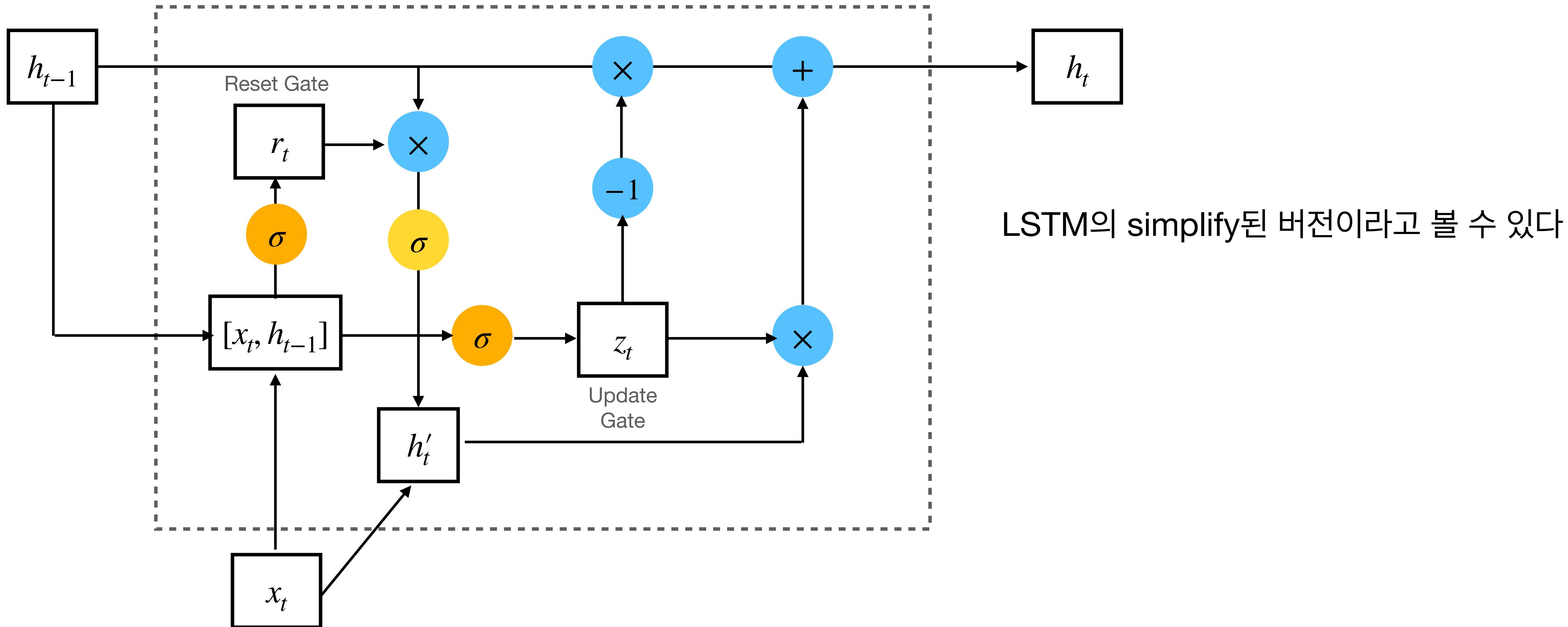
Copyright©2023. Acadential. All rights reserved.

	공식	역할
Forget Gate f_t	$\sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$	c_{t-1} 을 얼마나 c_t 에 반영할지 결정
Input Gate i_t	$\sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$	후보 cell state c'_t 을 얼마나 c_t 에 반영할지 결정
Output Gate o_t	$\sigma(W_o \cdot [x_t, h_{t-1}] + b_o)$	현재 cell state c_t 을 얼마나 hidden state에 반영할지 결정
Candidate Cell State c'_t	$\tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$	h_t 후보 cell state
Cell State c_t	$\sigma(f_t \cdot c_{t-1} + i_t \cdot c'_t)$	time-step t까지의 맥락을 담은 State (일종의 Memory의 역할을 한다)
Hidden State h_t	$o_t \cdot \tanh(c_t)$	예측값 \hat{y}_t 을 출력하기 위해 필요한 정보를 담은 hidden state

15-6. Gated Recurrent Units (GRU)

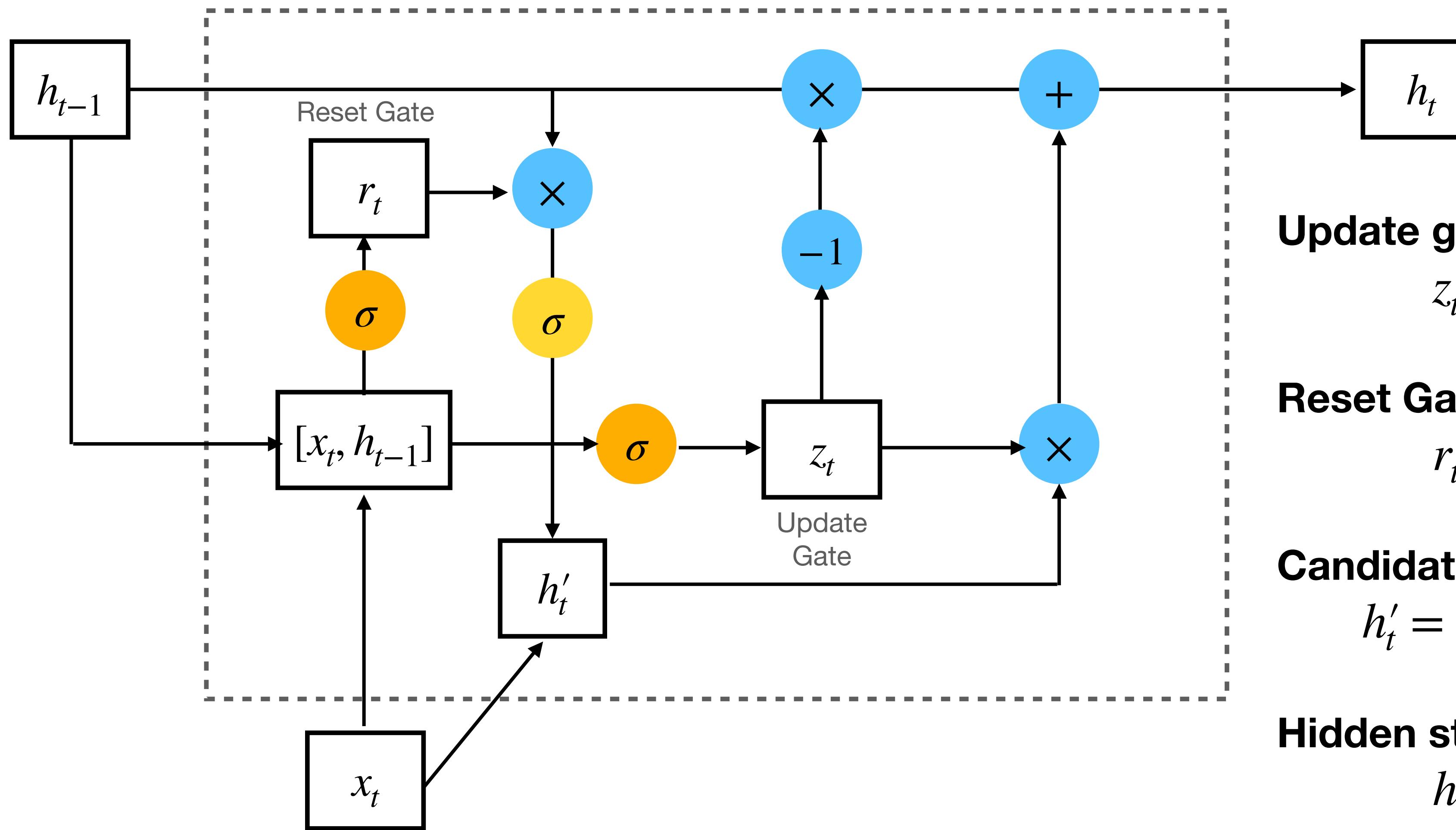
Recurrent Neural Networks

Gated Recurrent Units (GRUs)



Recurrent Neural Networks

GRU



Update gate

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

Reset Gate

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

Candidate hidden state:

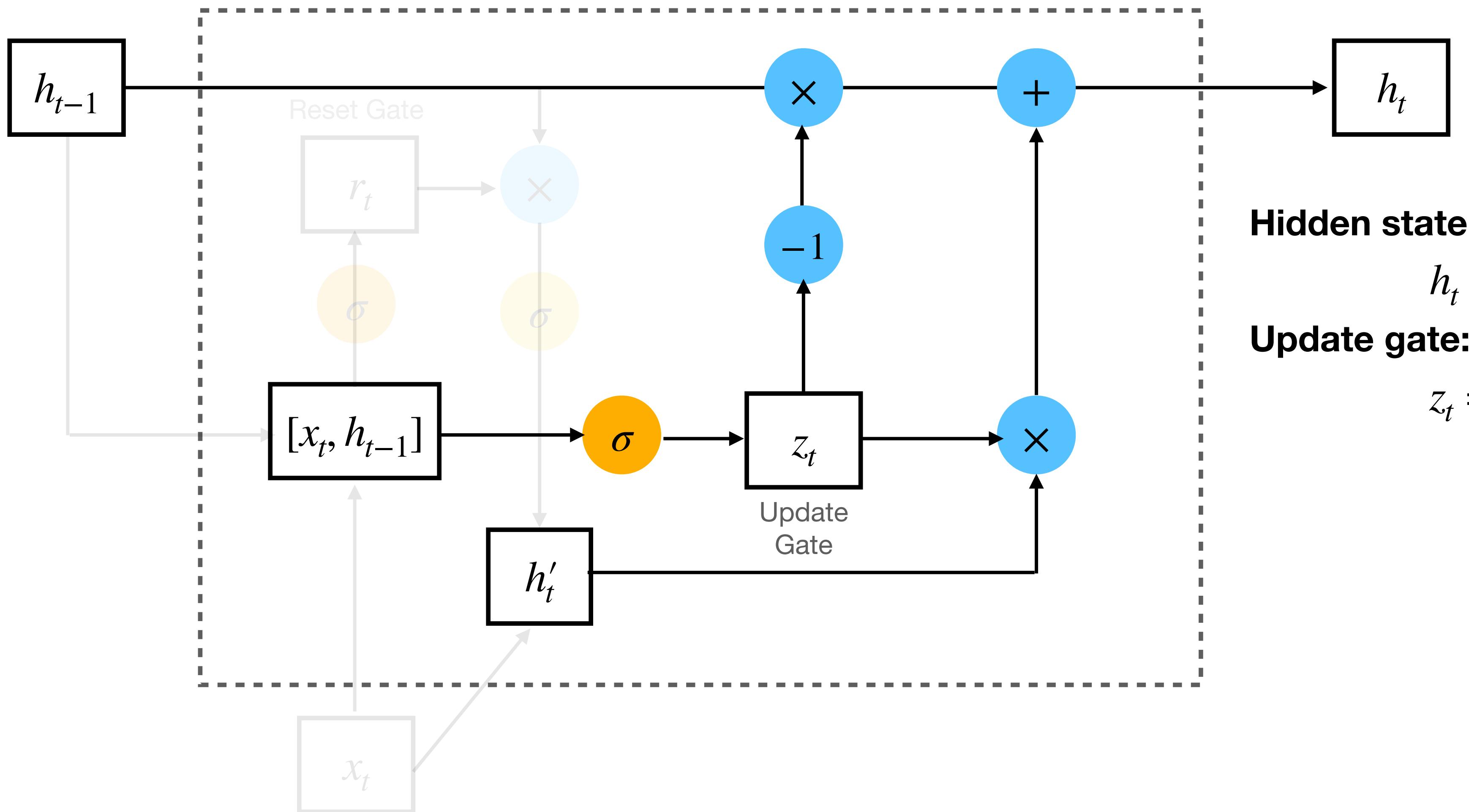
$$h'_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$$

Hidden state

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot h'_t$$

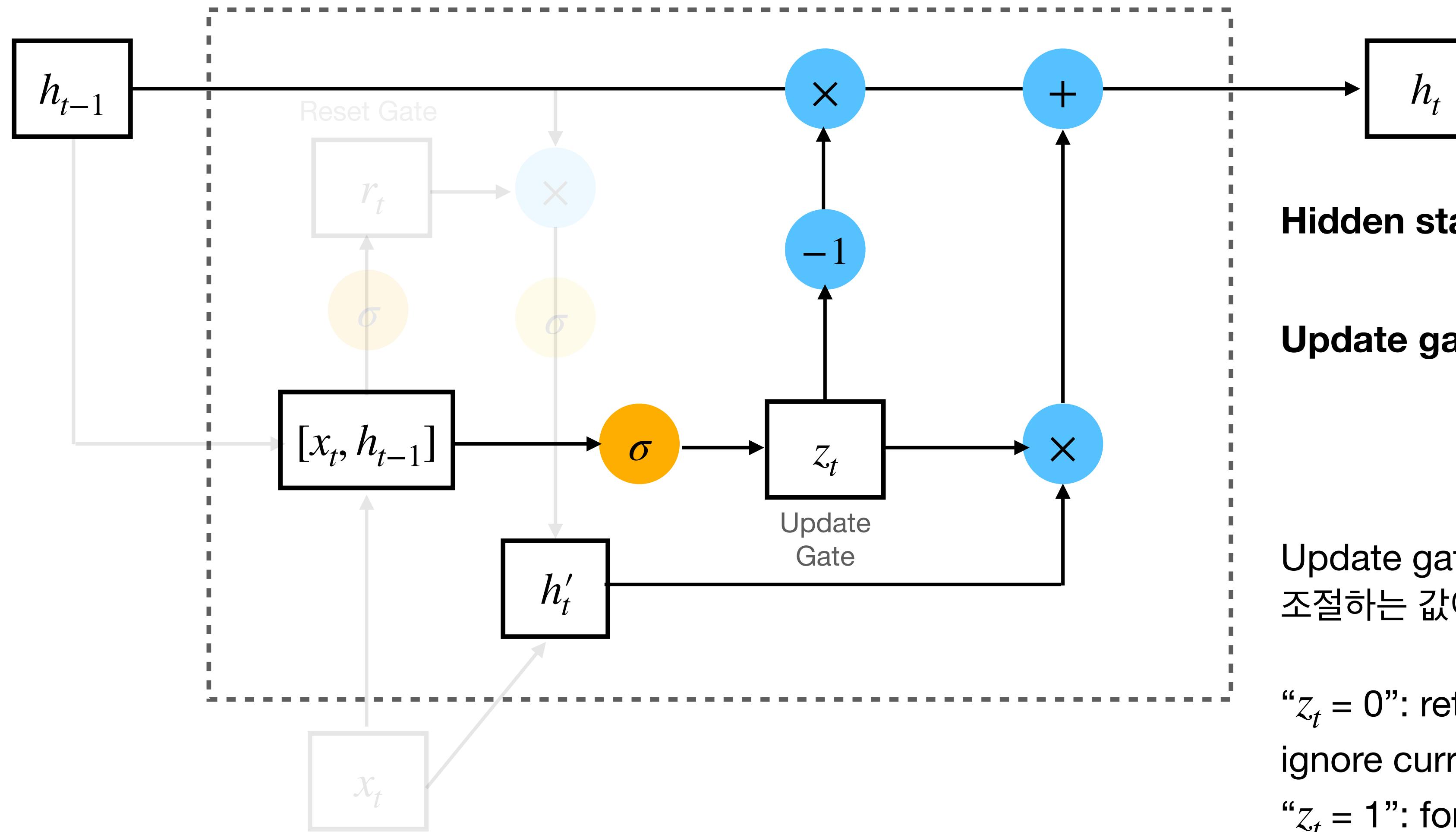
Recurrent Neural Networks

GRU



Recurrent Neural Networks

GRU



Hidden state:

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot h'_t$$

Update gate:

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

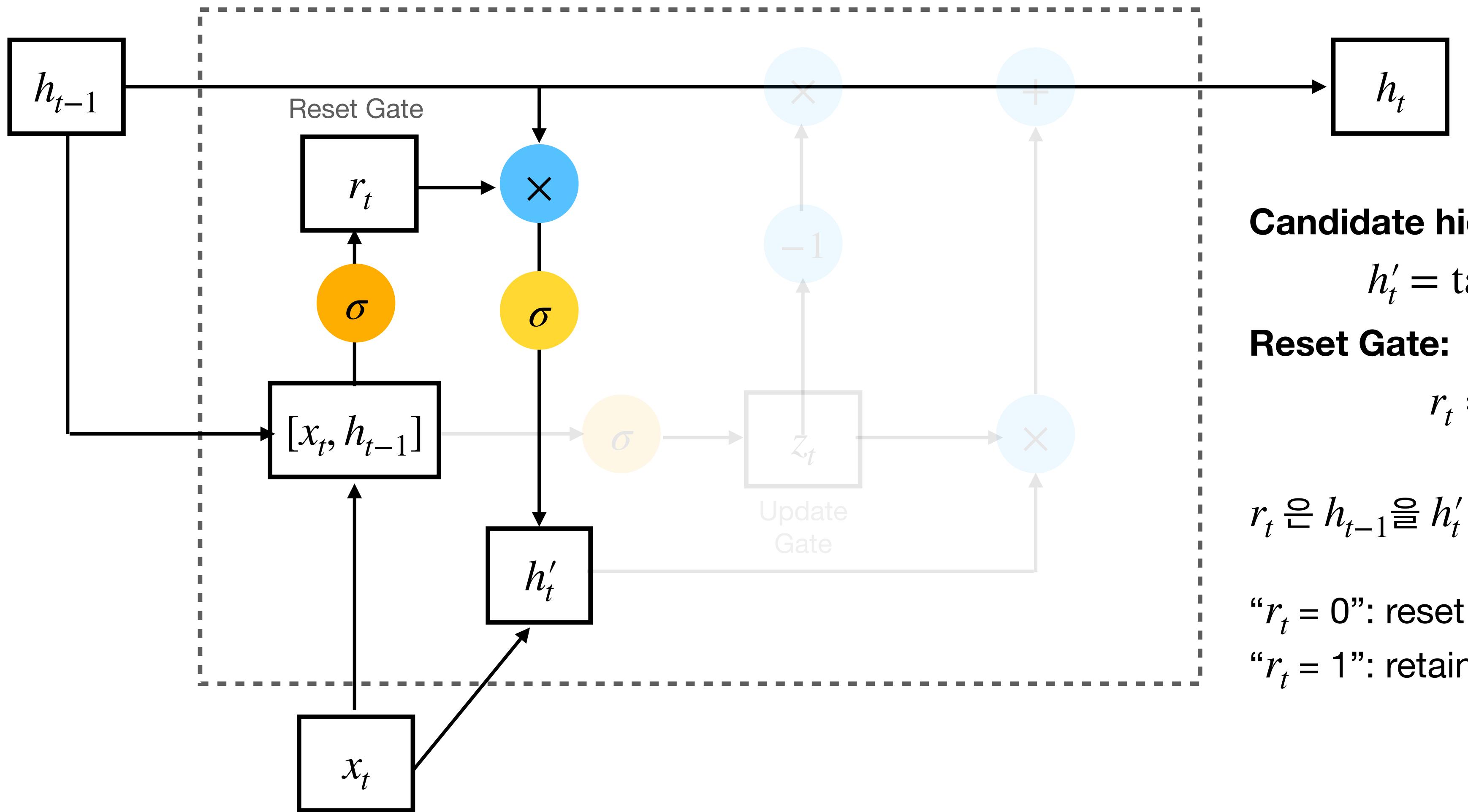
Update gate z_t 은 h_{t-1} 와 h'_t 을 h_t 에 얼마나 반영할지 조절하는 값이다.

“ $z_t = 0$ ”: retain previous hidden state h_{t-1} and ignore current candidate hidden state h'_t .

“ $z_t = 1$ ”: forget previous hidden state h_{t-1} and update using current candidate hidden state h'_t .

Recurrent Neural Networks

GRU



Candidate hidden state:

$$h'_t = \tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$$

Reset Gate:

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

r_t 은 h_{t-1} 을 h'_t 에 얼마나 반영할지 조절하는 값이다.

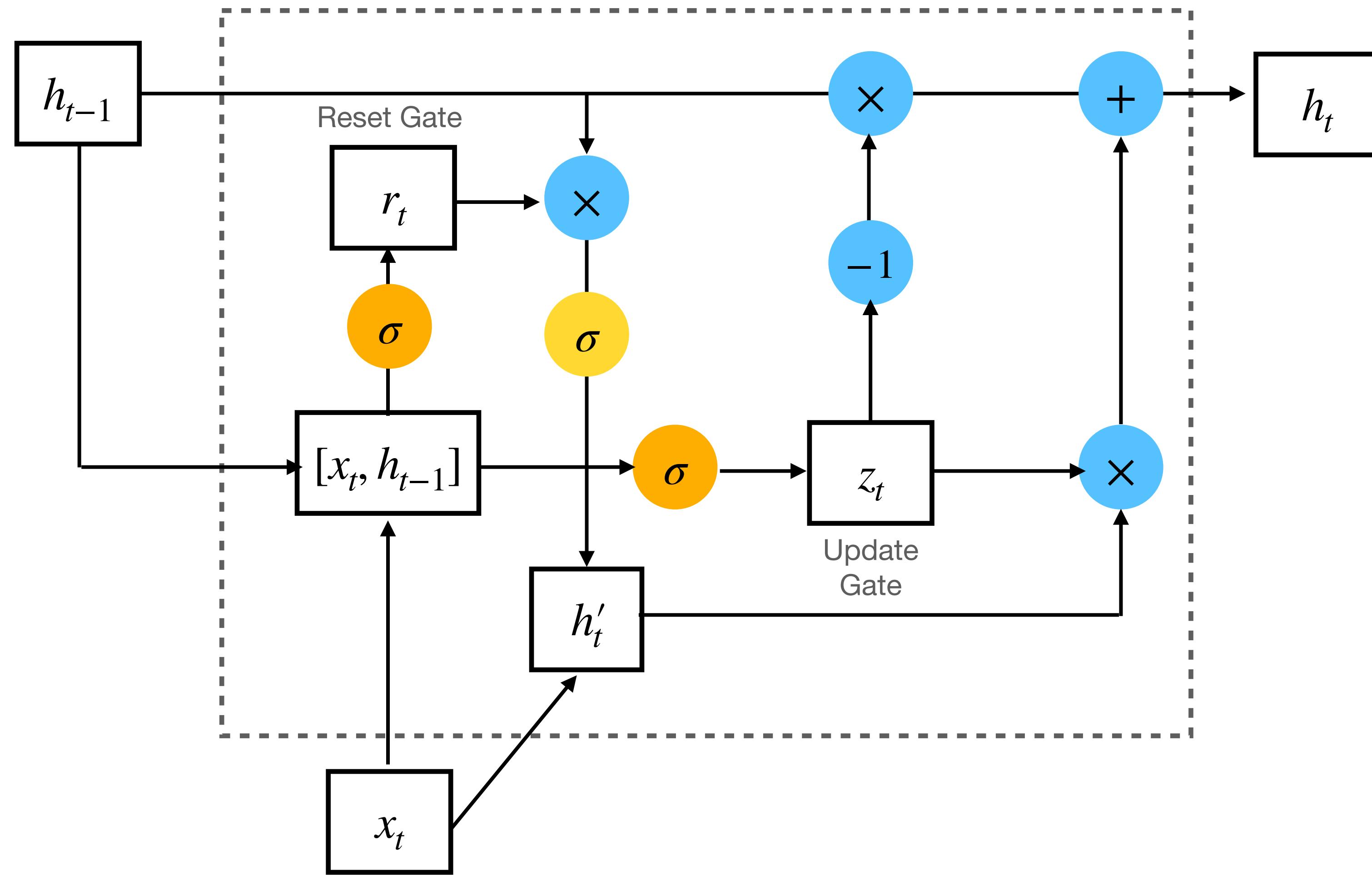
“ $r_t = 0$ ”: reset previous hidden state.

“ $r_t = 1$ ”: retain previous hidden state.

Recurrent Neural Networks

GRU

Copyright©2023. Acadential. All rights reserved.



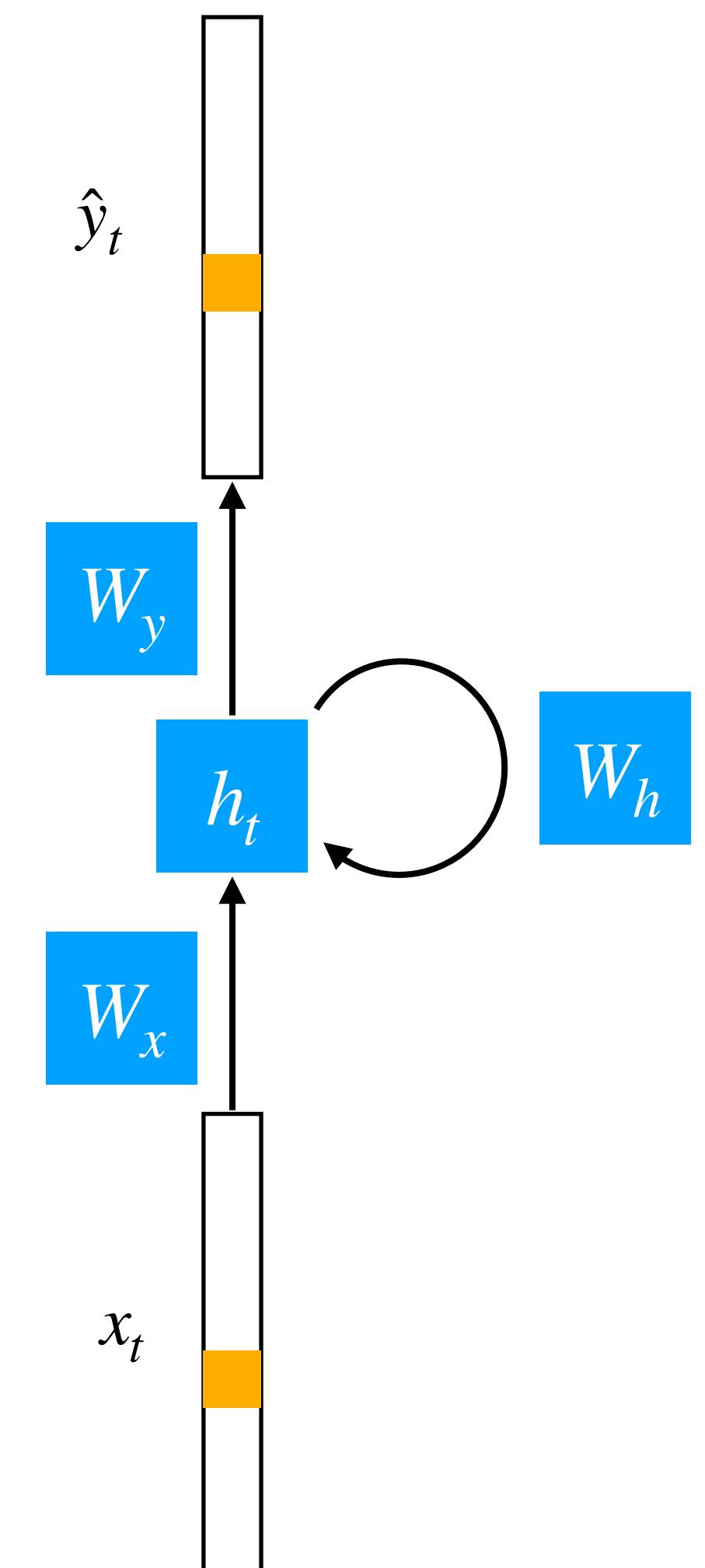
	공식	역할
Update Gate	$\sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$	h_{t-1} 와 h'_t 을 h_t 에 얼마나 반영할지 조절하는 값.
Reset Gate	$\sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$	h_{t-1} 을 h_t 에 얼마나 반영할지 조절하는 값.
Candidate Hidden State	$\tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$	현재 time-step의 input값이 반영된 candidate hidden state.
Hidden State	$(1 - z_t) \cdot h_{t-1} + z_t \cdot h'_t$	RNN와 LSTM의 Hidden state의 역할과 동일

15-7. RNN 계열의 모델의 한계

Recurrent Neural Networks

RNN 계열의 모델의 한계

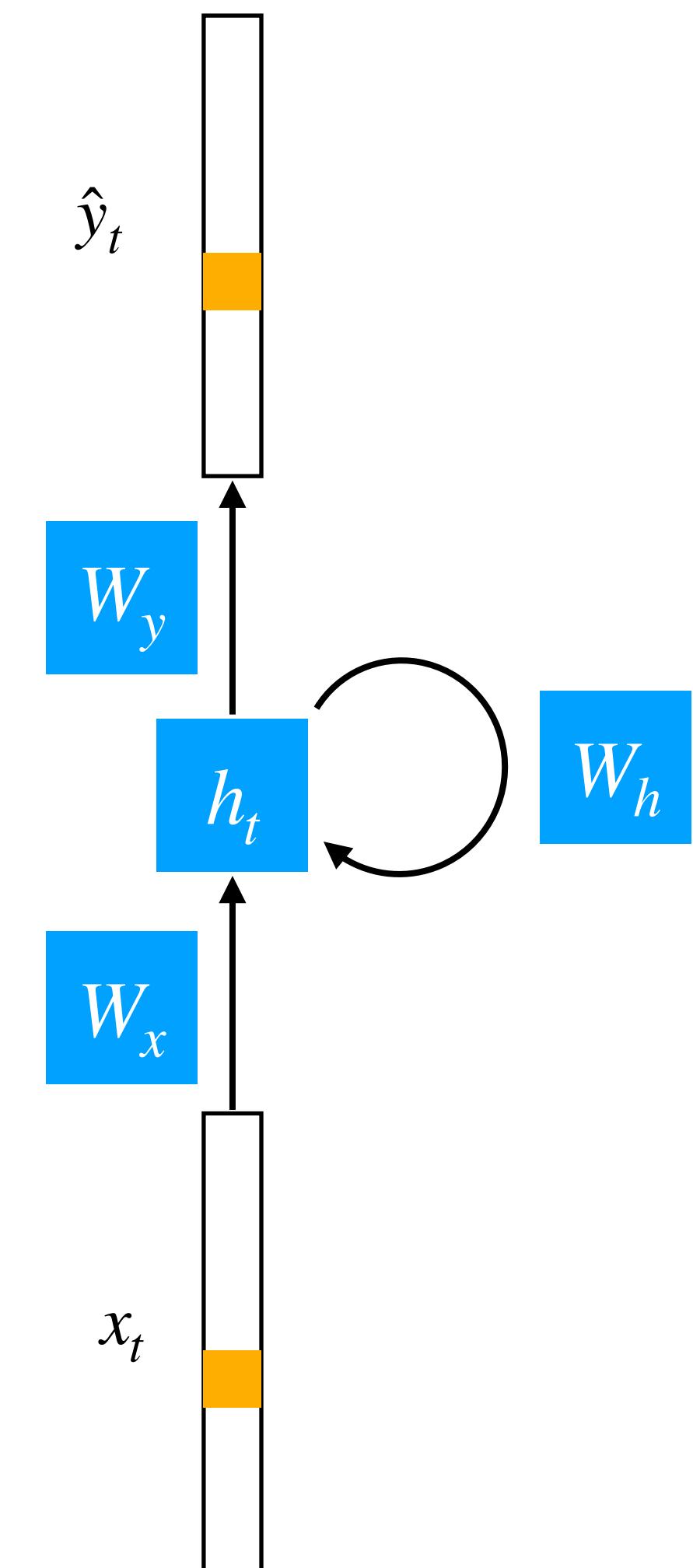
1. LSTM과 GRU가 vanilla RNN보다는 Vanishing Gradient의 문제가 덜하지만 여전히 Vanishing Gradient의 문제점이 존재한다.



Recurrent Neural Networks

RNN 계열의 모델의 한계

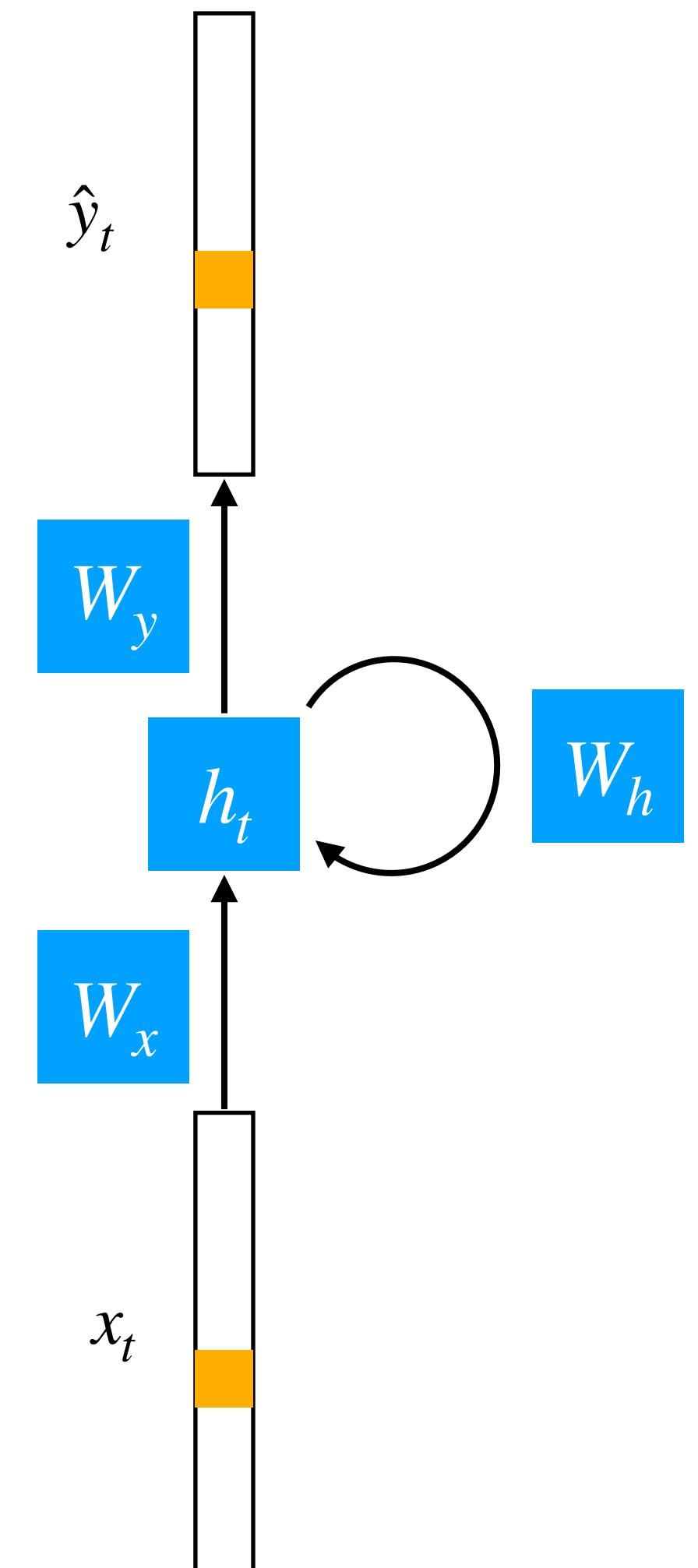
1. LSTM과 GRU가 vanilla RNN보다는 Vanishing Gradient의 문제가 덜하지만 여전히 Vanishing Gradient의 문제점이 존재한다.
2. 전체 sequence의 맥락을 한정된 “memory”인 hidden state에 저장해야한다.
 - 문장이 복잡하거나 매우 긴 경우 충분한 맥락을 담아내기 어려울 수 있다.



Recurrent Neural Networks

RNN 계열의 모델의 한계

1. LSTM과 GRU가 vanilla RNN보다는 Vanishing Gradient의 문제가 덜하지만 여전히 **Vanishing Gradient**의 문제점이 존재한다.
2. 전체 sequence의 맥락을 한정된 “memory”인 hidden state에 저장해야한다.
3. Sequence 길이가 길수록 backpropagation으로 traverse해야하는 unit의 개수도 늘어난다.
 - 인풋 배열상에서 멀리 떨어져 있는 두 입력값 간의 직접적인 interaction을 모델링하기 어려움.
 - RNN 계열의 모델들은 각 time step에 대해서 병렬적으로 계산하지 못함.



Recurrent Neural Networks

RNN 계열의 모델의 한계

- RNN 계열의 모델이 인풋 배열 상의 두 단어들의 상호작용을 모델링하려면?
- 해당 단어들 사이에 있는 단어들을 모두 처리해야한다!
- 아래의 예시의 경우 “나는”과 “등산을”간에 서로 어떤 관계인지 모델링하려면?

나는

관악산에

가서

...

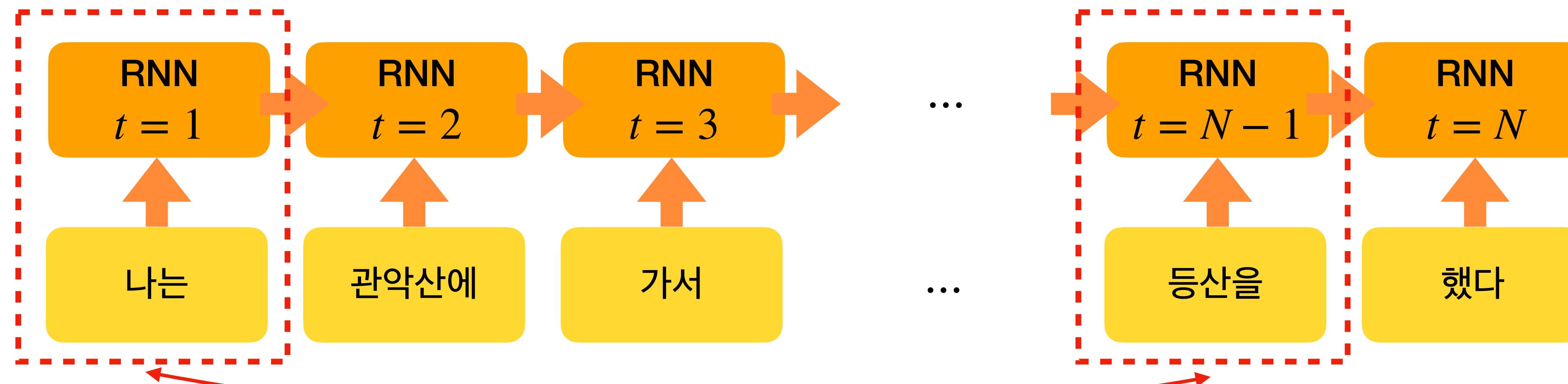
등산을

했다

Recurrent Neural Networks

RNN 계열의 모델의 한계

- 총 **N-1개의 RNN unit**을 통과해야 이를 모델링 할 수 있다! (Direct interaction을 modeling하기 어려움)
- 즉, 중간에 놓인 단어들 때문에 “**의미가 희석**” 될 수 있다.



Recurrent Neural Networks

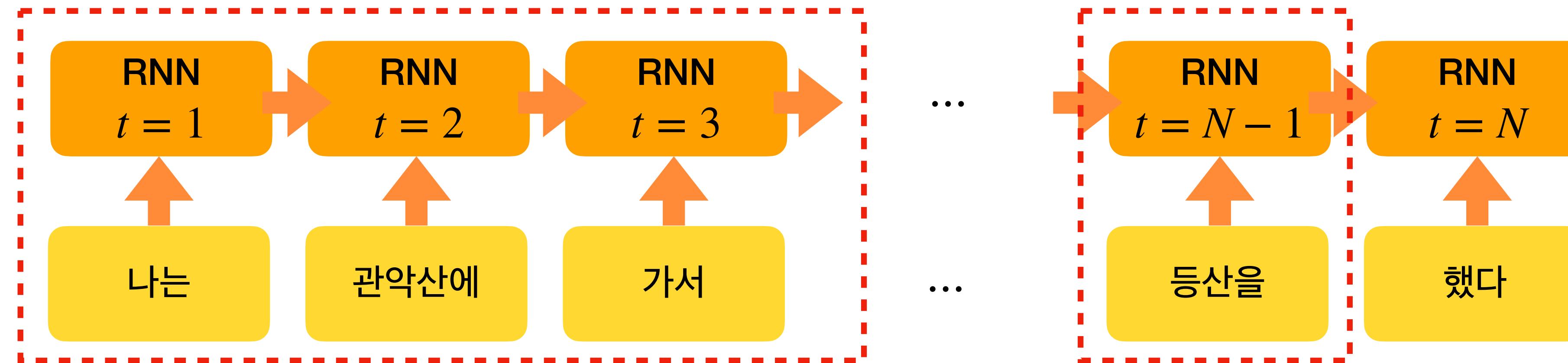
RNN 계열의 모델의 한계

- $t = t'$ 의 RNN unit을 계산하기 위해서는 앞선 $t < t'$ RNN unit들에 대해서 차례대로 계산해야한다!
- 즉, Parallelizable (병렬적)으로 계산하는게 어렵다.

Recurrent Neural Networks

RNN 계열의 모델의 한계

- $t = t'$ 의 RNN unit을 계산하기 위해서는 앞선 $t < t'$ RNN unit들에 대해서 차례대로 계산해야한다!
- 즉, Parallelizable (병렬적)으로 계산하는게 어렵다 → 따라서 병렬화에 특화되어 있는 GPU에 효율적이지 않다!



이것을 구하기 위해서는 $t < (N - 1)$ 에 대해서 차례대로 계산해야함!

Recurrent Neural Networks

RNN 계열의 모델의 한계

즉, RNN은 다음과 같은 한계를 가졌다:

1. Vanishing Gradient 문제를 완전히 해소하지 못한다.
2. Sequence의 맥락을 전부 다 hidden vector 상에 내포해야 한다.
3. 단어들 간의 Direct interaction을 modeling하지 못한다.
4. Computation을 병렬화하지 못한다. → GPU에 비효율적이다.

이를 해결하기 위한 방안으로 “Attention”이 제안됨!

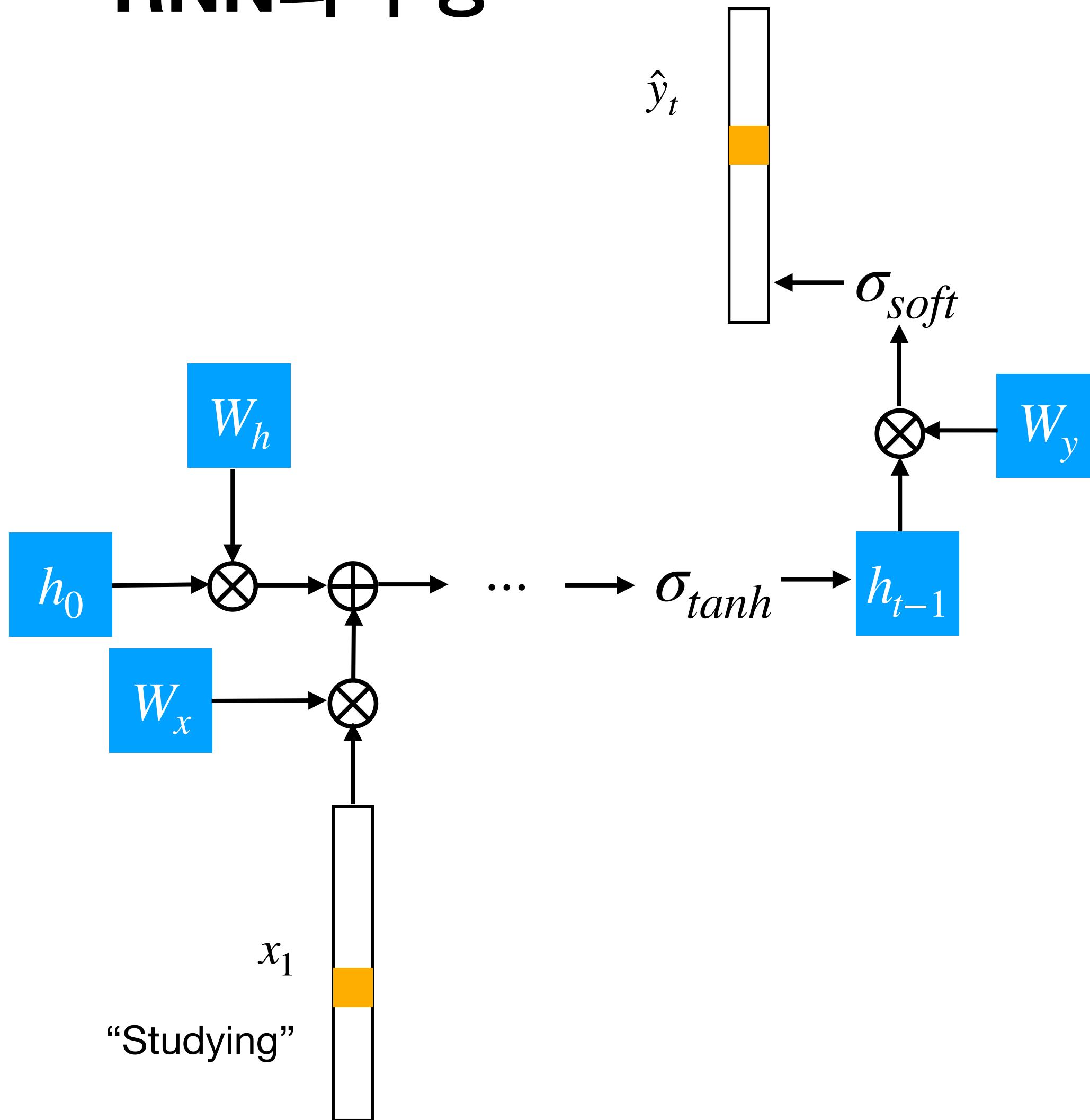
15-8. PyTorch로 만들어보는 RNN, LSTM, GRU

15-9. RNN, GRU, LSTM을 활용한 NLP 프로젝트

15-10. Section 15 요약

Section Summary

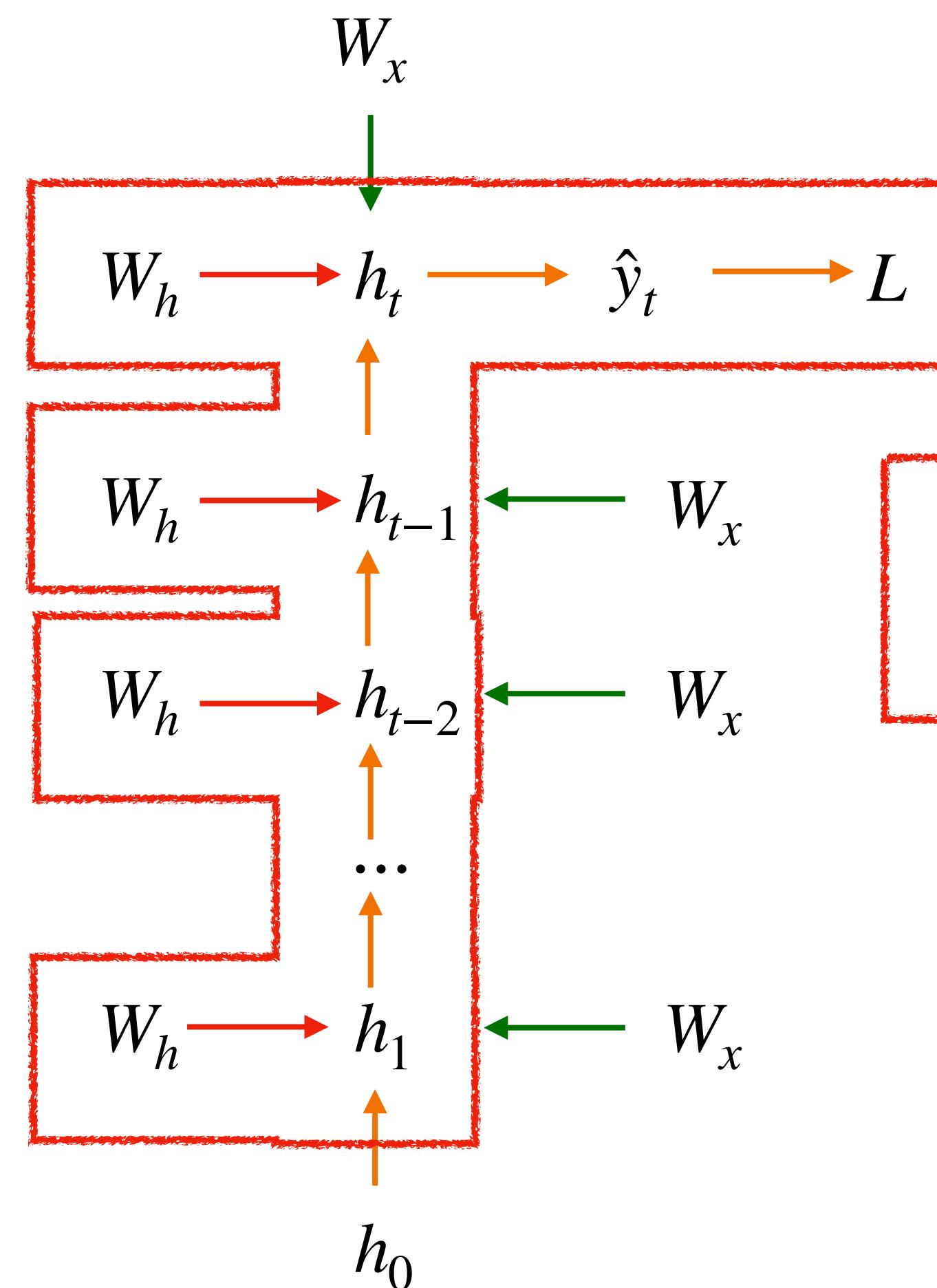
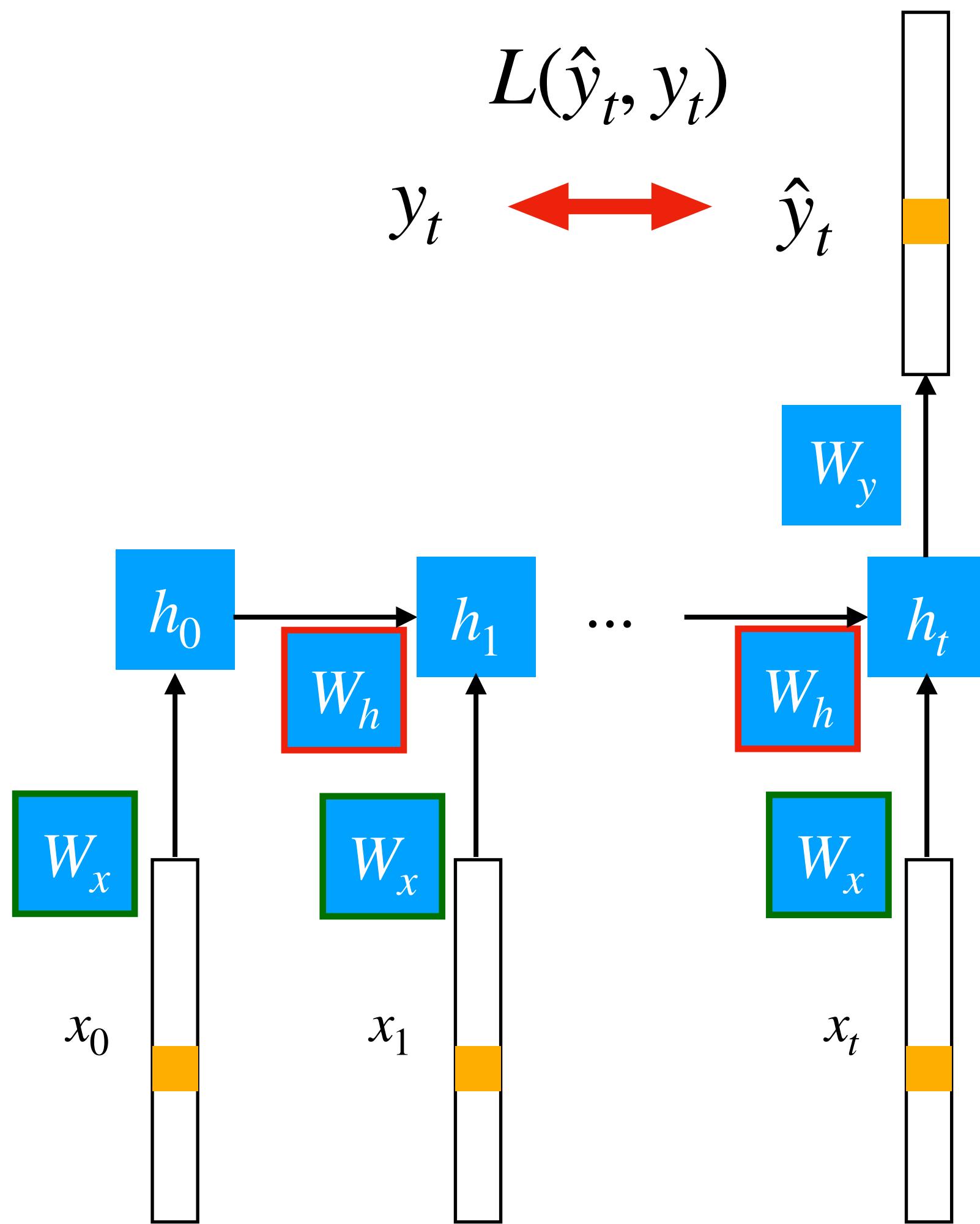
RNN의 구성



공식	역할
hidden state $h_t = \tanh(W_h h_{t-1} + W_x x_t)$	time-step t까지의 맥락을 저장 해두는 hidden state
predicted probability $p(\hat{y}_t) = \text{softmax}(W_y h_t)$	t-th step의 Hidden state 으로부터 \hat{y}_t 을 예측

Section Summary

RNN의 Backpropagation

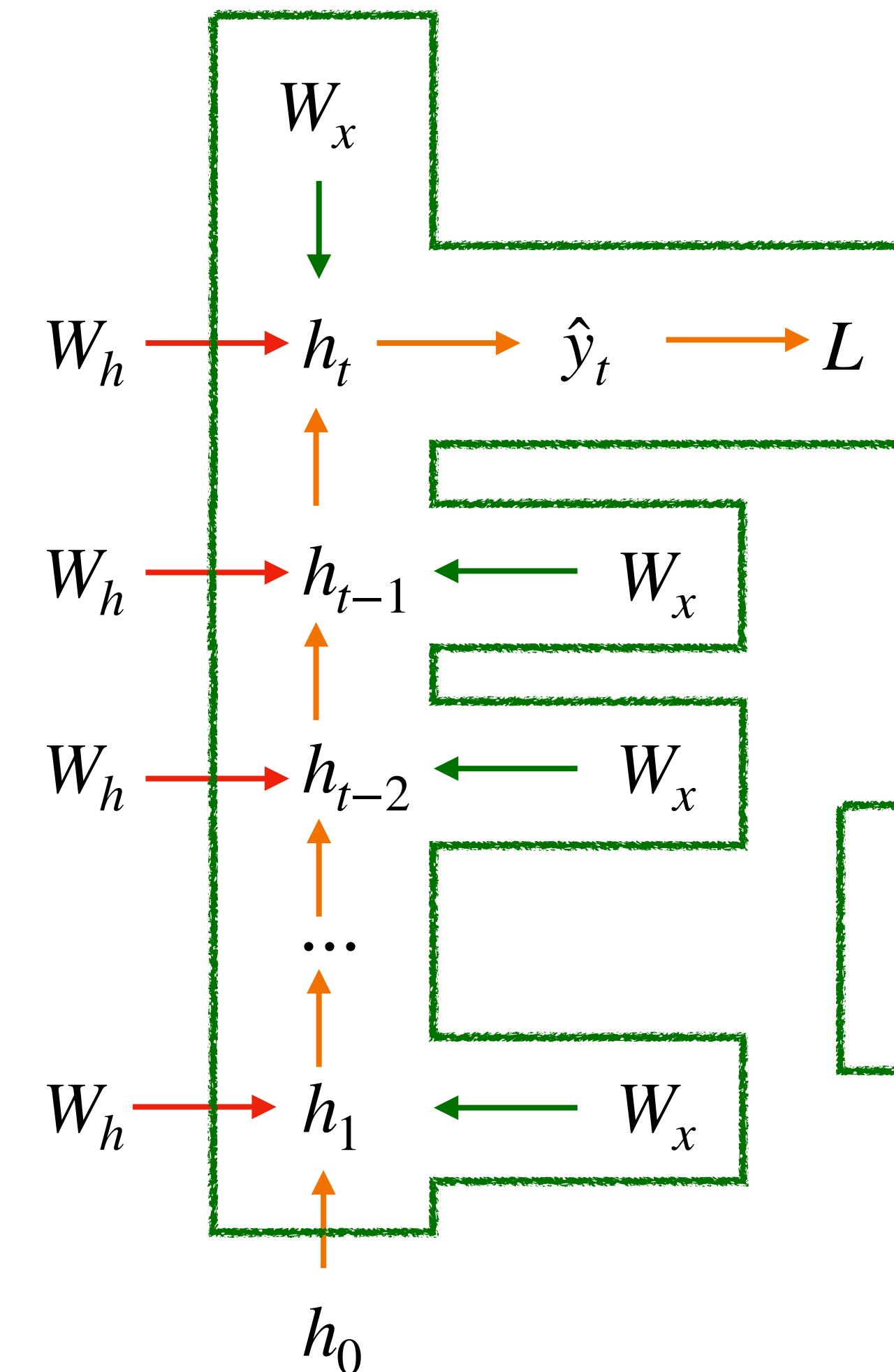
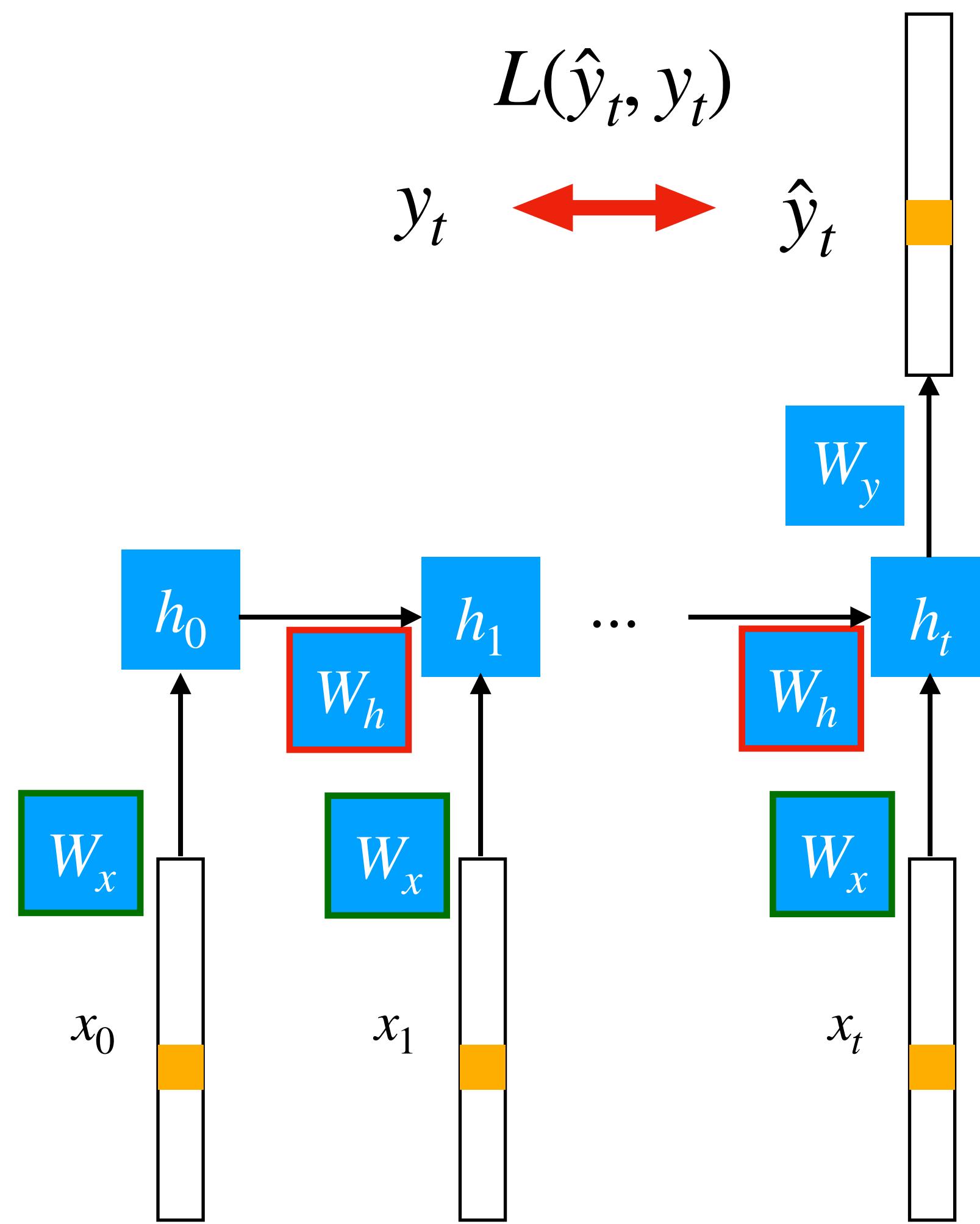


$$\frac{\partial L}{\partial W_h} = \sum_{i=1}^t \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \dots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial W_h}$$

$$\frac{\partial L}{\partial W_x} = \sum_{i=1}^t \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \dots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial W_x}$$

Section Summary

RNN의 Backpropagation



$$\frac{\partial L}{\partial W_h} = \sum_{i=1}^t \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \dots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial W_h}$$

$$\frac{\partial L}{\partial W_x} = \sum_{i=1}^t \frac{\partial L}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_{t-1}} \dots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial W_x}$$

Section Summary

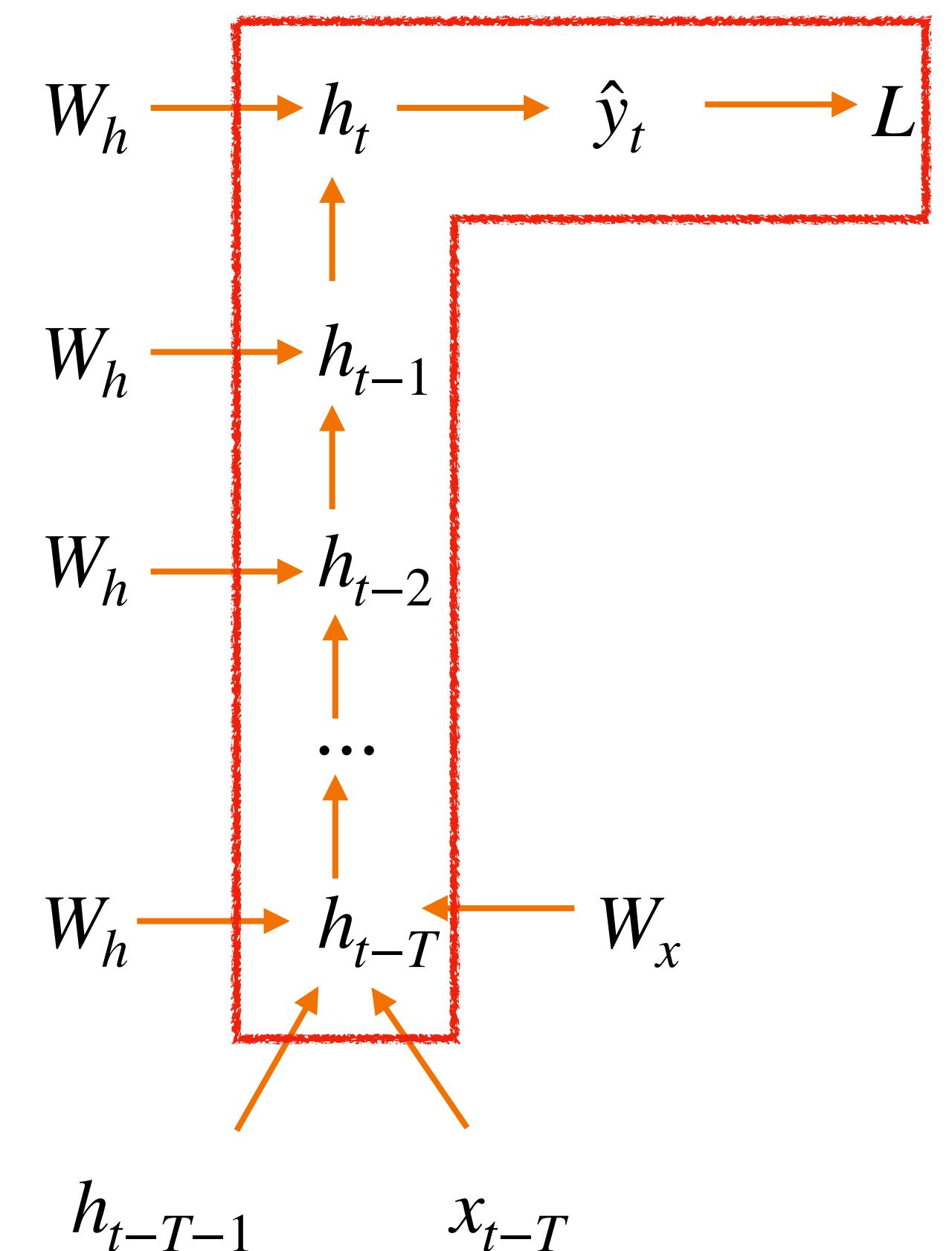
Vanishing Gradient Problem in RNN

$t = T$ 번째 입력으로부터 비롯되는 partial derivative은:

$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

- $W = W_h$ or W_x
- $h_t = \tanh(W_h h_{t-1} + W_x x_t)$

$$\left(1 - \tanh^2(W_h h_{t-1} + W_x x_t) \right) W_h$$



Section Summary

Vanishing Gradient Problem in RNN

$t = T$ 번째 입력으로부터 비롯되는 partial derivative은:

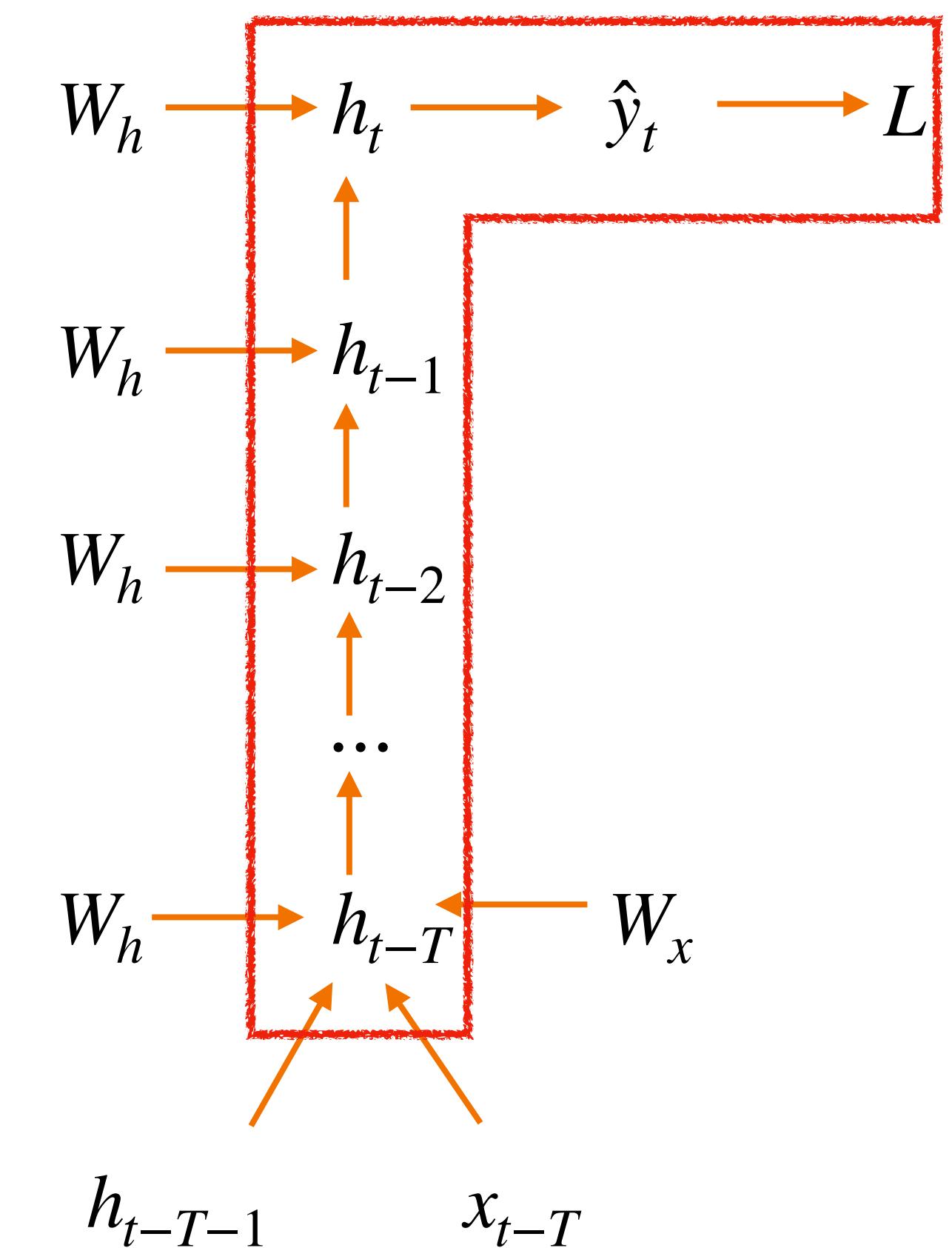
$$\frac{\partial L}{\partial \hat{y}_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{t-T+1}}{\partial h_{t-T}} \frac{\partial h_{t-T}}{\partial W}$$

- $W = W_h$ or W_x
- $h_t = \tanh(W_h h_{t-1})$

$$\left(1 - \tanh^2(W_h h_{t-1} + W_x x_t) \right) W_h$$

$$\left| \left(1 - \tanh^2(W_h h_{t-1} + W_x x_t) \right) W_h \right| < 1$$

따라서, 여러 번 곱할시 Gradient가 '0'으로 Vanish한다!

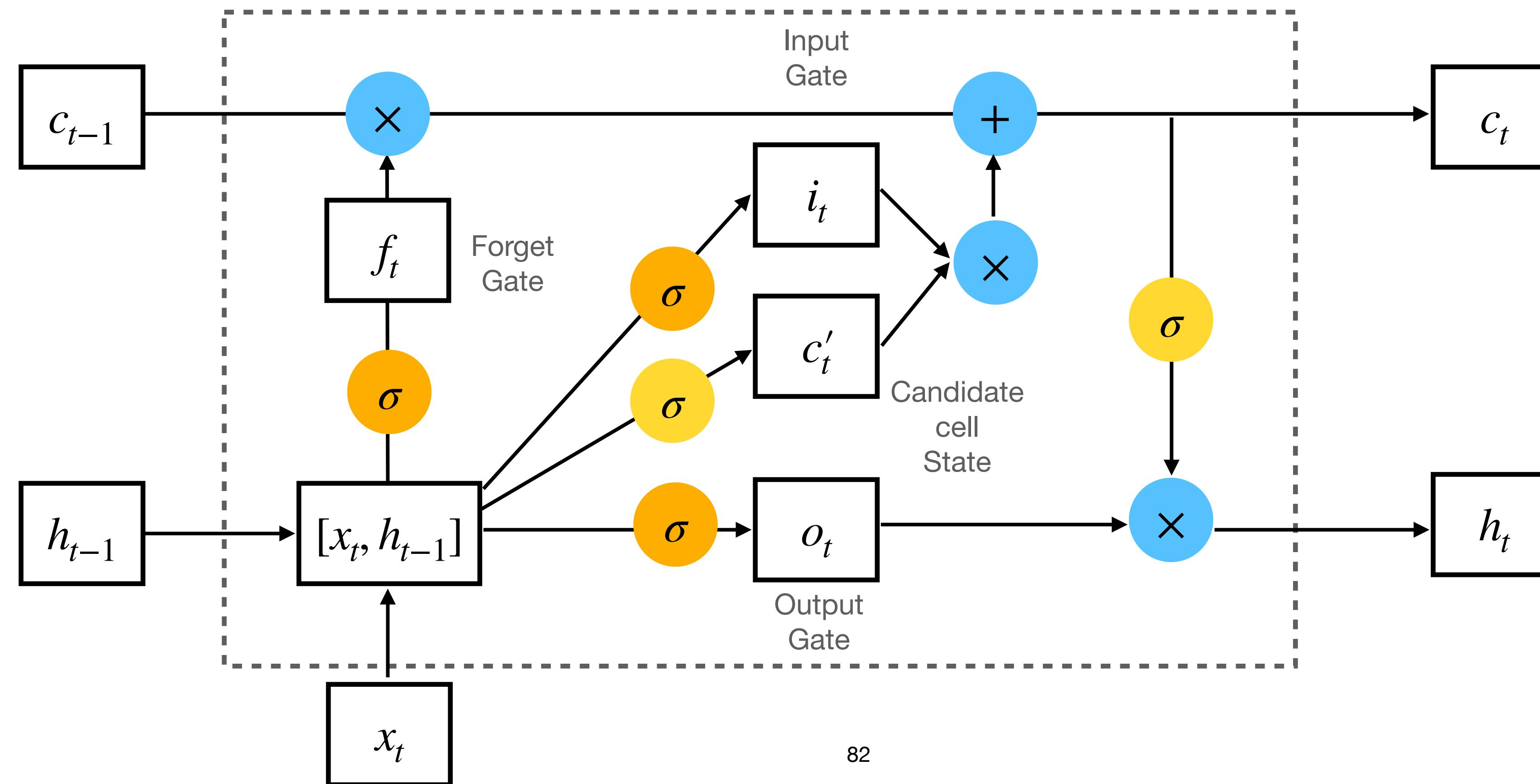


Section Summary

LSTM의 구성

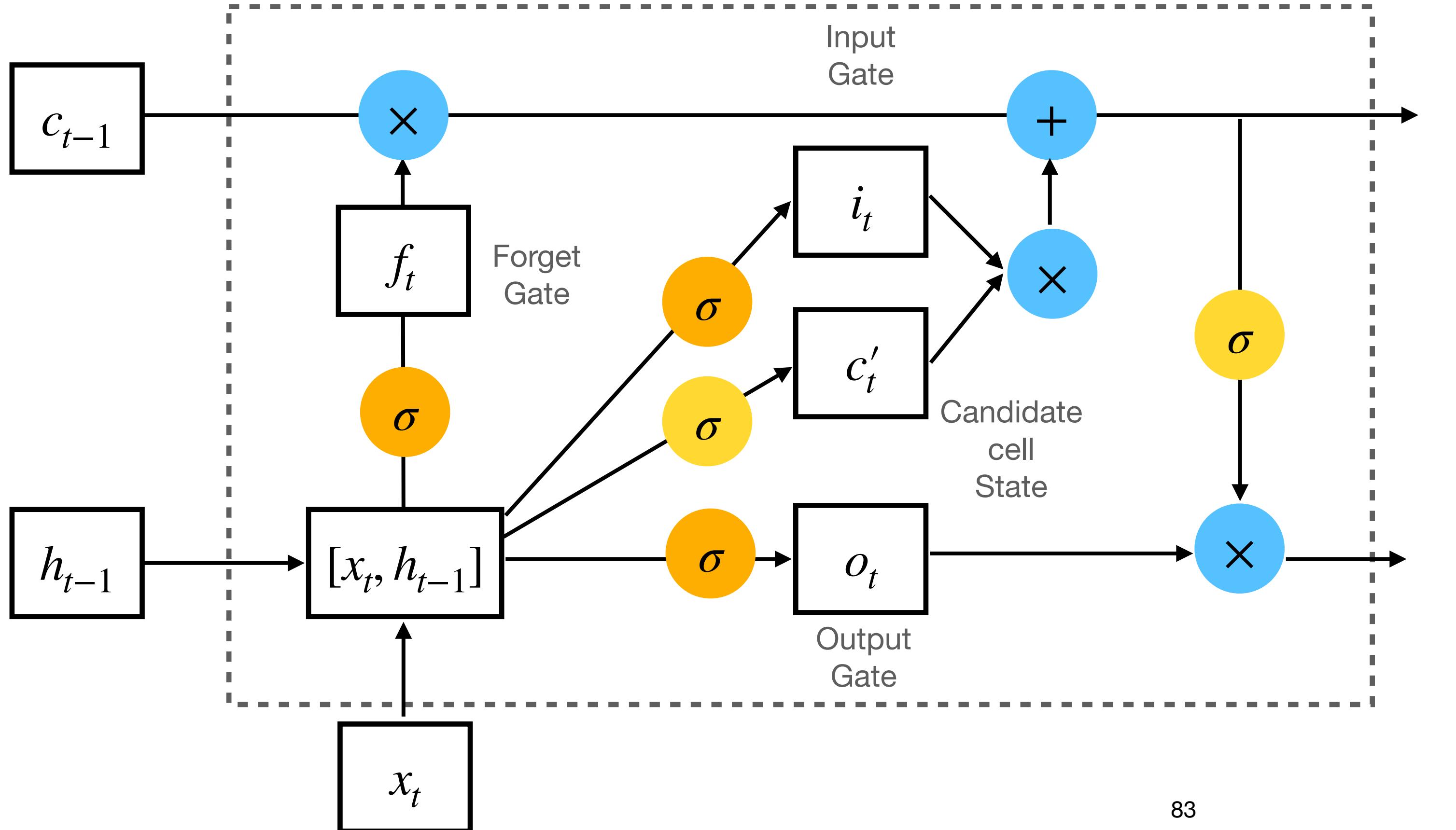
Copyright©2023. Acadential. All rights reserved.

- σ : tanh activation
- σ : sigmoid activation



Section Summary

LSTM의 구성

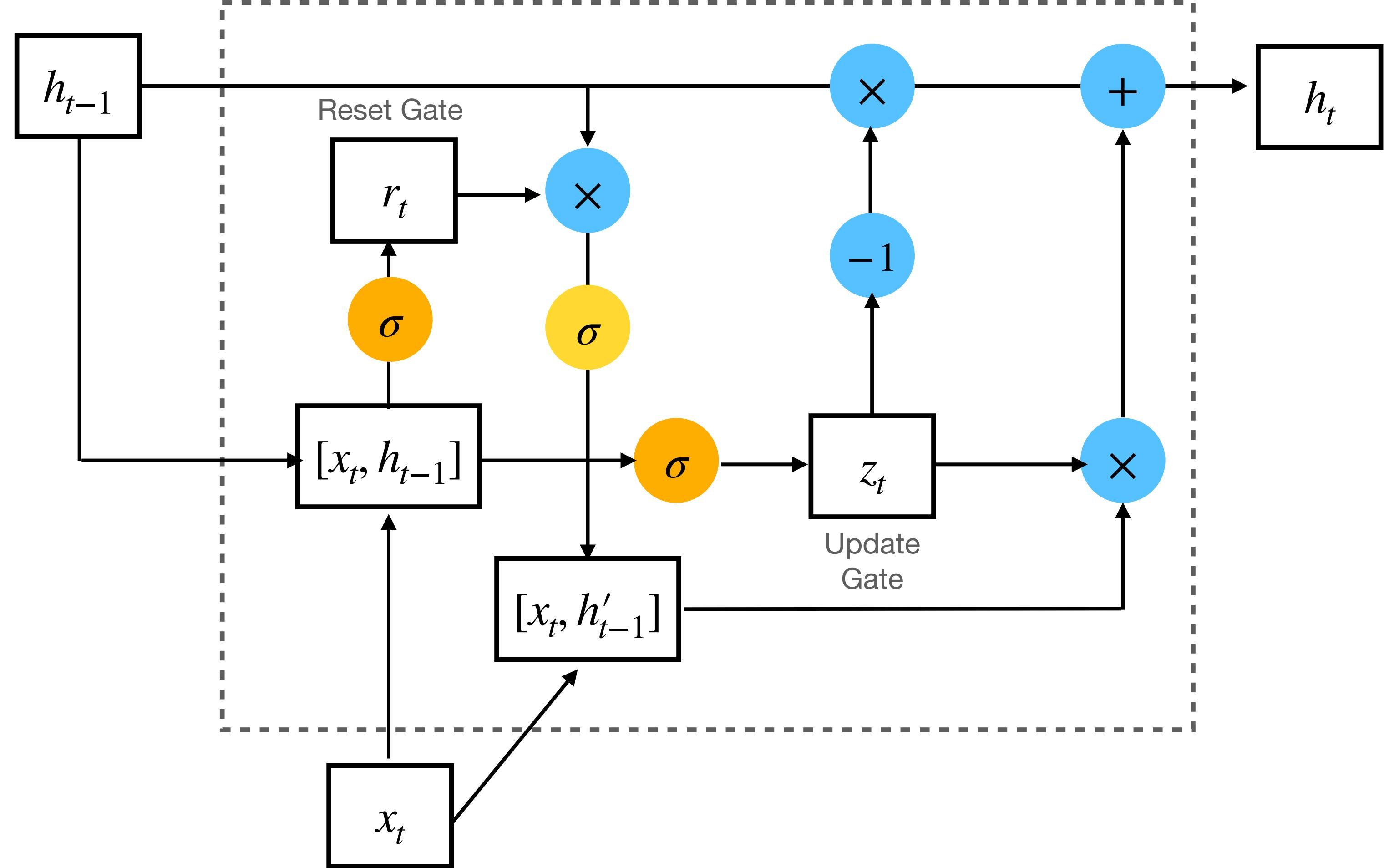


Copyright©2023. Acadential. All rights reserved.

	공식	역할
Forget Gate f_t	$\sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$	c_{t-1} 을 얼마나 c_t 에 반영할지 결정
Input Gate i_t	$\sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$	후보 cell state c'_t 을 얼마나 c_t 에 반영할지 결정
Output Gate o_t	$\sigma(W_o \cdot [x_t, h_{t-1}] + b_o)$	현재 cell state c_t 을 얼마나 hidden state에 반영할지 결정
Candidate Cell State c'_t	$\tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$	h_t 후보 cell state
Cell State c_t	$\sigma(f_t \cdot c_{t-1} + i_t \cdot c'_t)$	time-step t까지의 맥락을 담은 State (일종의 Memory의 역할을 한다)
Hidden State h_t	$o_t \cdot \tanh(c_t)$	예측값 \hat{y}_t 을 출력하기 위해 필요한 정보를 담은 hidden state

Section Summary

GRU



	공식	역할
Update Gate	$\sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$	h_{t-1} 와 h'_t 을 h_t 에 얼마나 반영할지 조절하는 값.
Reset Gate	$\sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$	h_{t-1} 을 h_t 에 얼마나 반영할지 조절하는 값.
Candidate Hidden State	$\tanh(W_h \cdot [r_t \cdot h_{t-1}, x_t] + b_h)$	현재 time-step의 input값이 반영된 candidate hidden state.
Hidden State	$(1 - z_t) \cdot h_{t-1} + z_t \cdot h'_t$	RNN와 LSTM의 Hidden state의 역할과 동일

Section Summary

RNN 계열의 모델의 한계

즉, RNN은 다음과 같은 한계를 가졌다:

1. Vanishing Gradient 문제를 완전히 해소하지 못한다.
2. Sequence의 맥락을 전부 다 hidden vector 상에 내포해야 한다.
3. 단어들 간의 Direct interaction을 modeling하지 못한다.
4. Computation을 병렬화하지 못한다. → GPU에 비효율적이다.

이를 해결하기 위한 방안으로 “Attention”이 제안됨!

Next Up!

Next Up!

Attention

- 중간에 위치한 단어들을 거치지 않고 두 단어 간의 “Direct Interaction”을 모델링할 수 있을까?

Attention!

다음 챕터에서 살펴보자!

