- sequence ←
- recurrent neuron.
- long short-term memory (LSTM)*

In-Class 21: Recurrent Neural Network (RNN)

[SCS4049] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

5-fold cross validation

⇒ 1000개의 sample
↳ 5개로 쪼갰음.

input = 8 x 1000

output = 1 x 1000 $\in \{0,1,2\}^{1000}$

label:
output = 0 인 애들 500개
= 1 인 애들 250개
= 2 인 애들 250개.

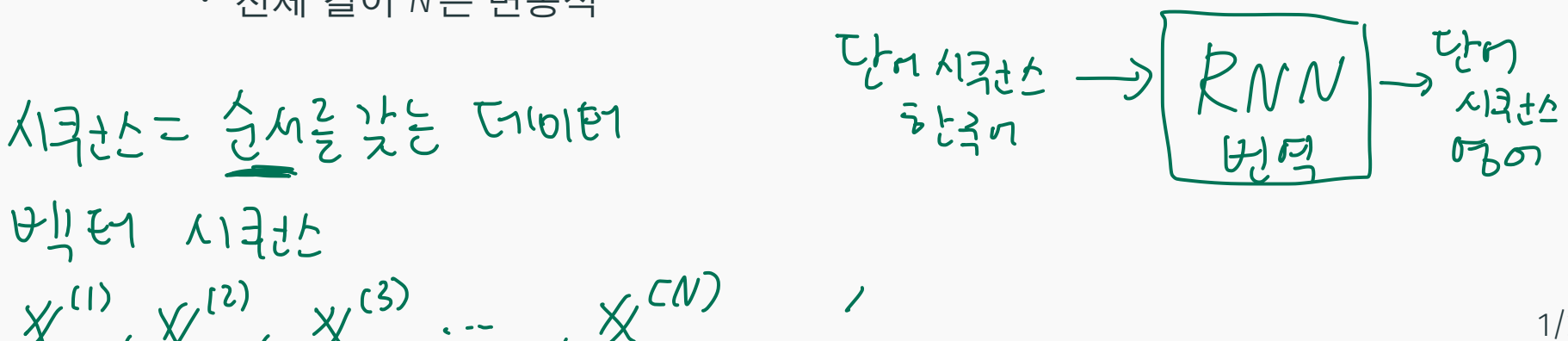
Introduction of recurrent neural network (RNN)

- Rumelhart, et. al., Learning Internal Representations by Error Propagation (1986)
- $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}, \mathbf{x}^{(t+1)}, ..., \mathbf{x}^{(N)}\}$와 같은 sequence를 처리하는데 특화
- RNN 은 긴 시퀀스를 처리할 수 있으며, 길이가 변동적인 시퀀스도 처리 가능
- Parameter sharing: 전체 sequence에서 파라미터를 공유함

CNN은 시퀀스를 다룰 수 없나?

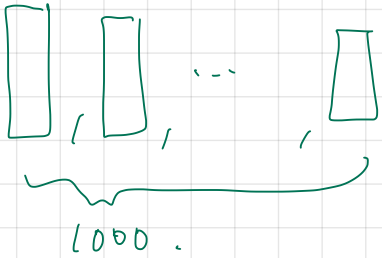
- 시간 축으로 1-D convolution
  $\Rightarrow$ 시간 지연 신경망도 가능하지만 깊이가 얕음 (shallow)
- RNN은 깊은 구조(deep)가 가능함

- Data type
  - 일반적으로 $\mathbf{x}_t$는 시계열 데이터 (time series, temporal data)
  - 여기서 $t = 1, 2, ..., N$은 time step 또는 sequence 내에서의 순서
  - 전체 길이 N은 변동적

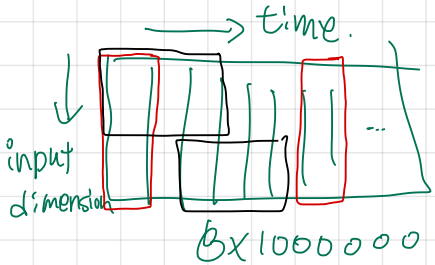


배치의 시퀀스.

B x 1000000



→ 2D 이미지



→ CNN.

Recurrent neuron

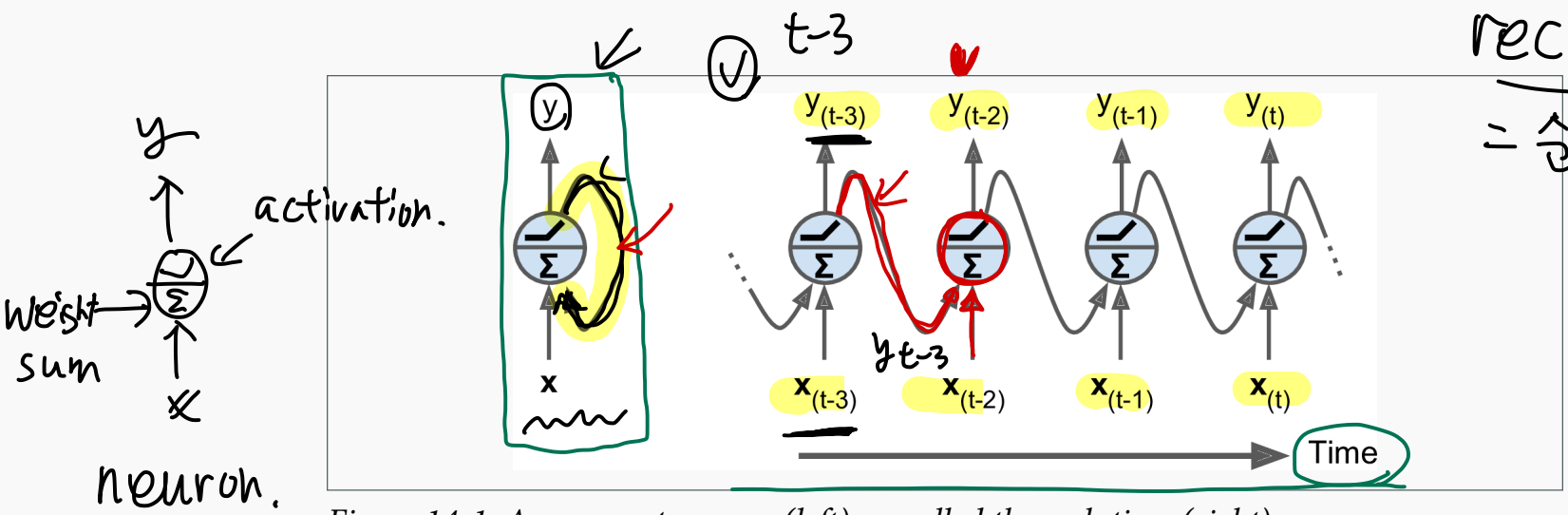


Figure 14-1. A recurrent neuron (left), unrolled through time (right)

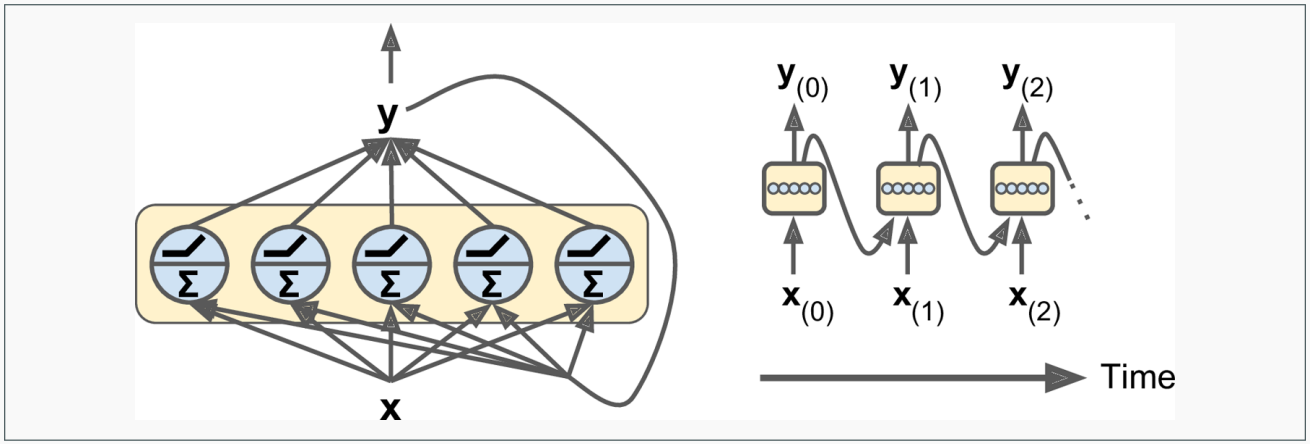


Figure 14-2. A layer of recurrent neurons (left), unrolled through time (right)

- Unfold or unroll: 순환적(recurrent, recursive) 계산을 순환 없이 반복적인 구조의 그래프로 변환

Recurrent neuron

Output of a single recurrent neuron for a single instance

$$\mathbf{y}^{(t)} = \phi\left(\mathbf{x}^{(t)\,T} \cdot \mathbf{w}_x + \mathbf{y}^{(t-1)\,T} \cdot \mathbf{w}_y + b\right) \tag{1}$$

Outputs of a layer of recurrent neurons for all instances in a minibatch

$$\mathbf{Y}^{(t)} = \phi\left(\mathbf{X}^{(t)} \cdot \mathbf{W}_x + \mathbf{Y}^{(t-1)} \cdot \mathbf{W}_y + \mathbf{b}\right) \tag{2}$$

$$= \phi\left(\begin{bmatrix} \mathbf{X}^{(t)} & \mathbf{Y}^{(t-1)} \end{bmatrix} \cdot \mathbf{W} + \mathbf{b}\right) \text{ with } \mathbf{W} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_y \end{bmatrix} \tag{3}$$

Memory cells and input/output sequences

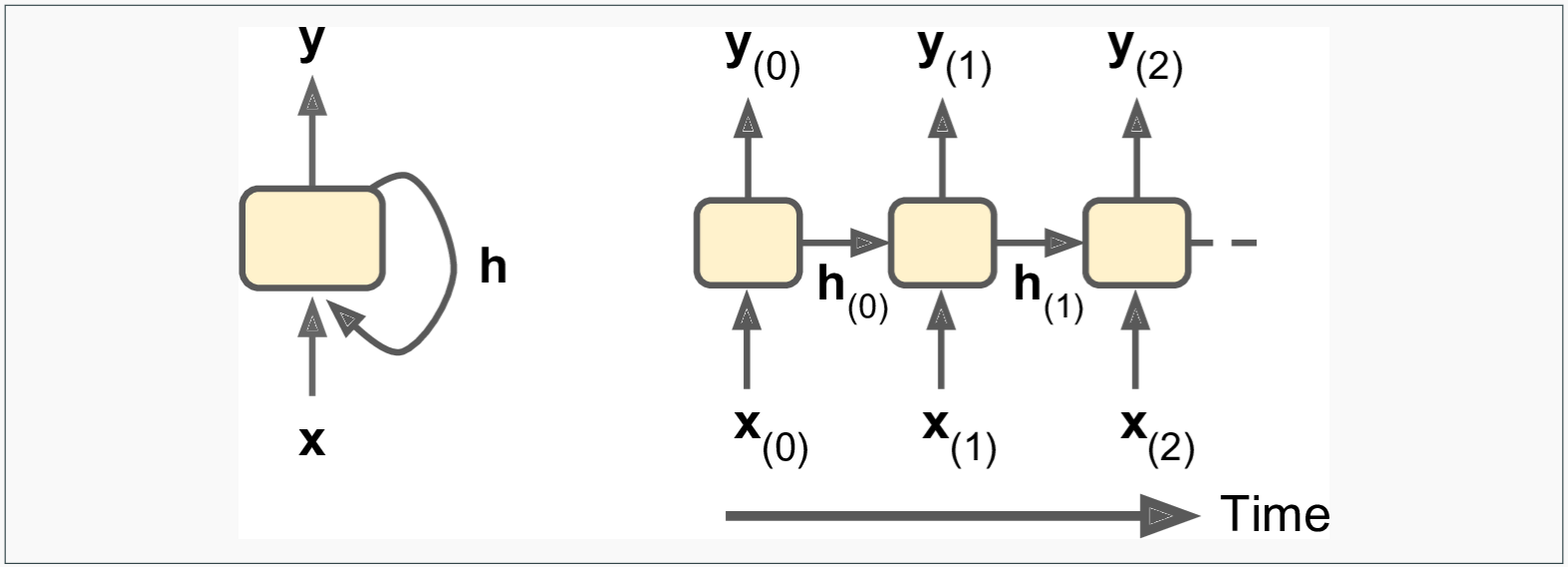


Figure 14-3. A cell's hidden state and its output may be different

Memory cells and input/output sequences

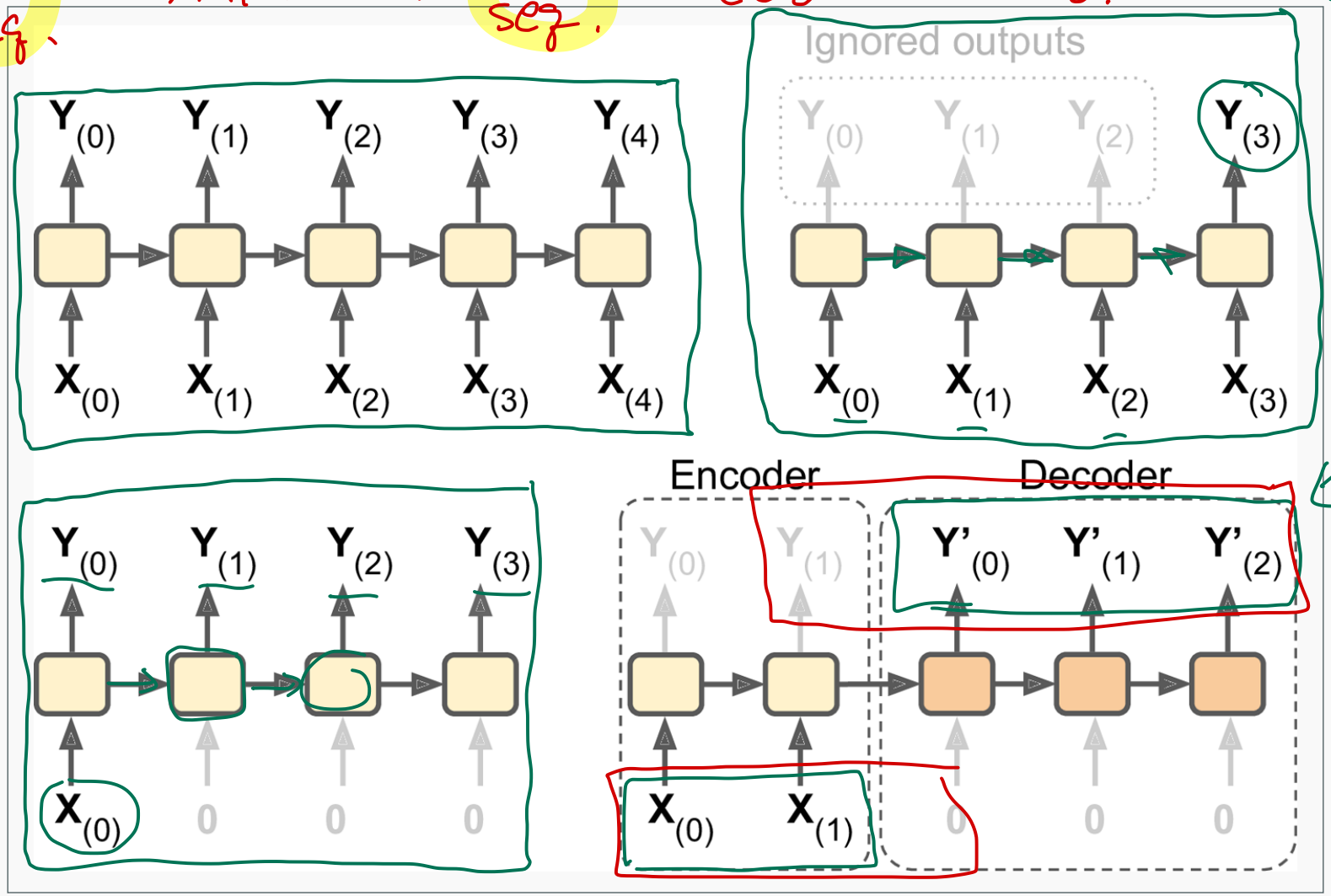


Figure 14-4. Seq to seq (top left), seq to vector (top right), vector to seq (bottom left), delayed seq to seq (bottom right)

An encoder-decoder network for machine translation

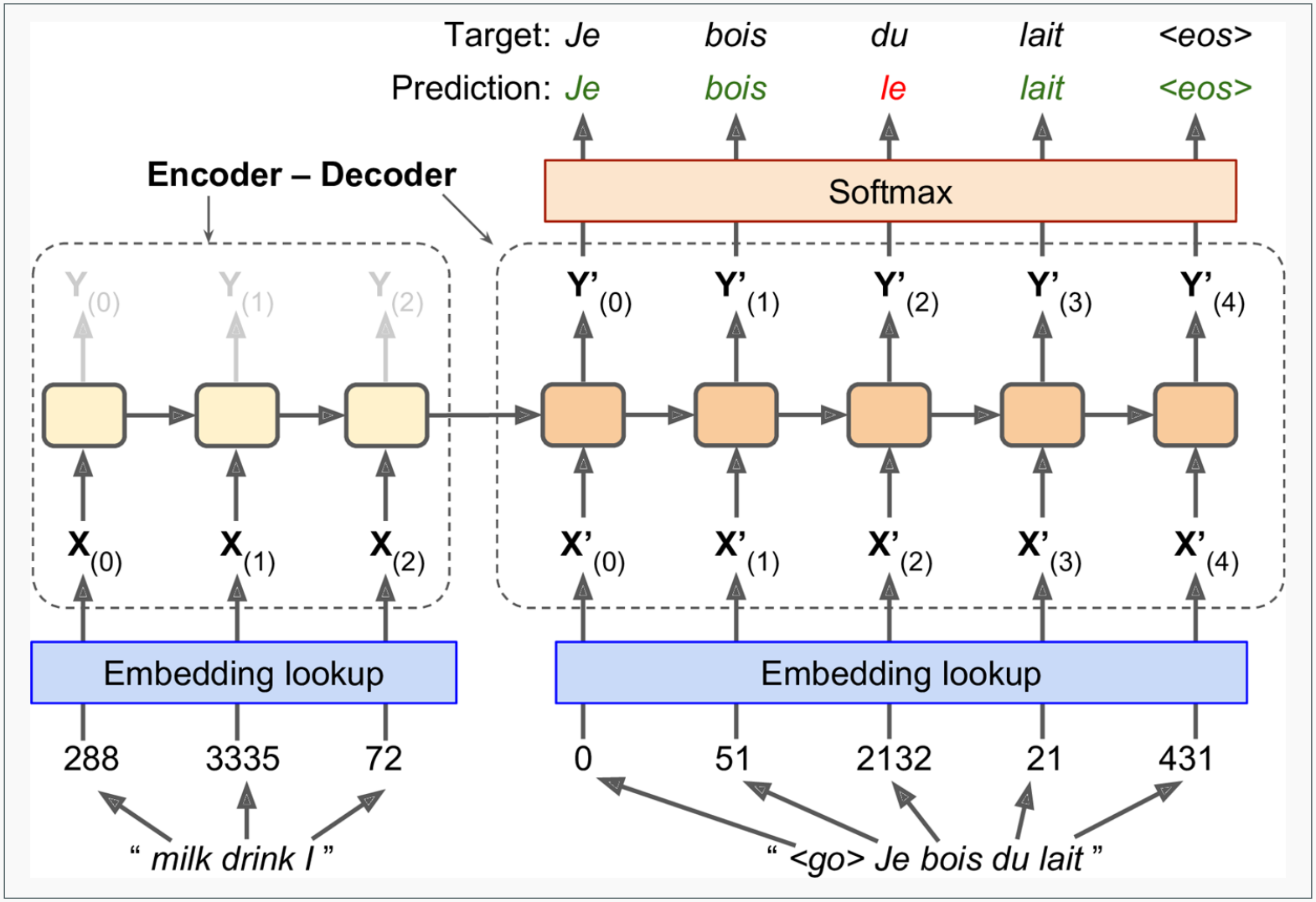
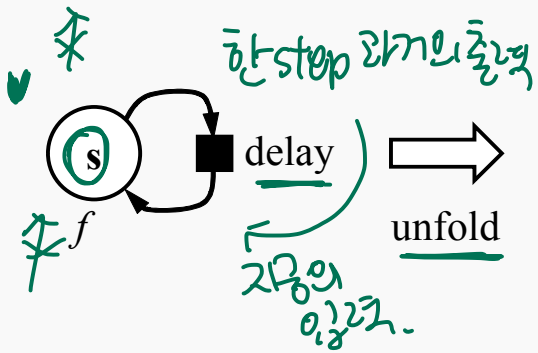


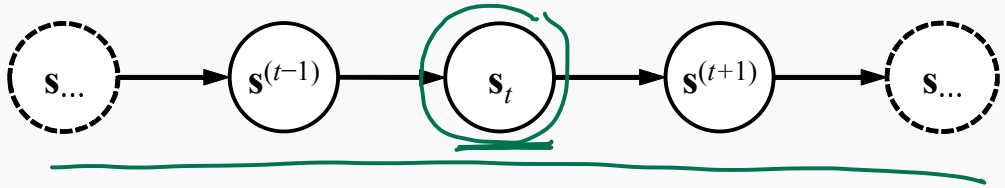
Figure 14-15. A simple machine translation model

Recurrent neuron with hidden unit

Recurrent neuron

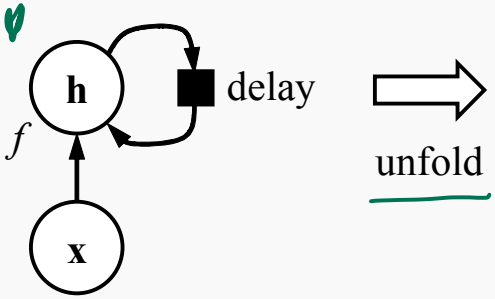
$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}; \boldsymbol{\theta}) \tag{4}$$

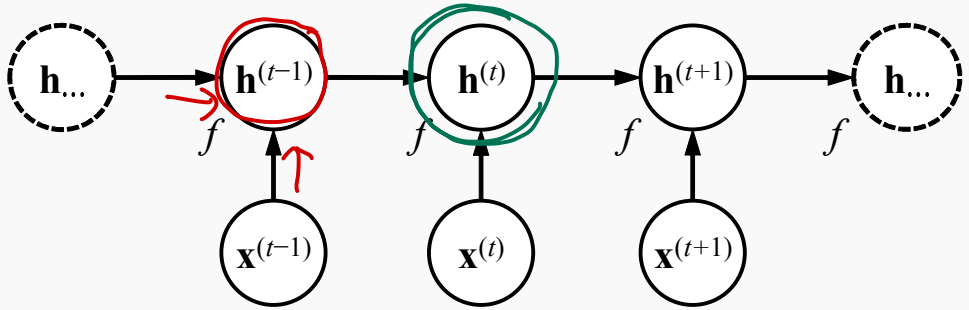




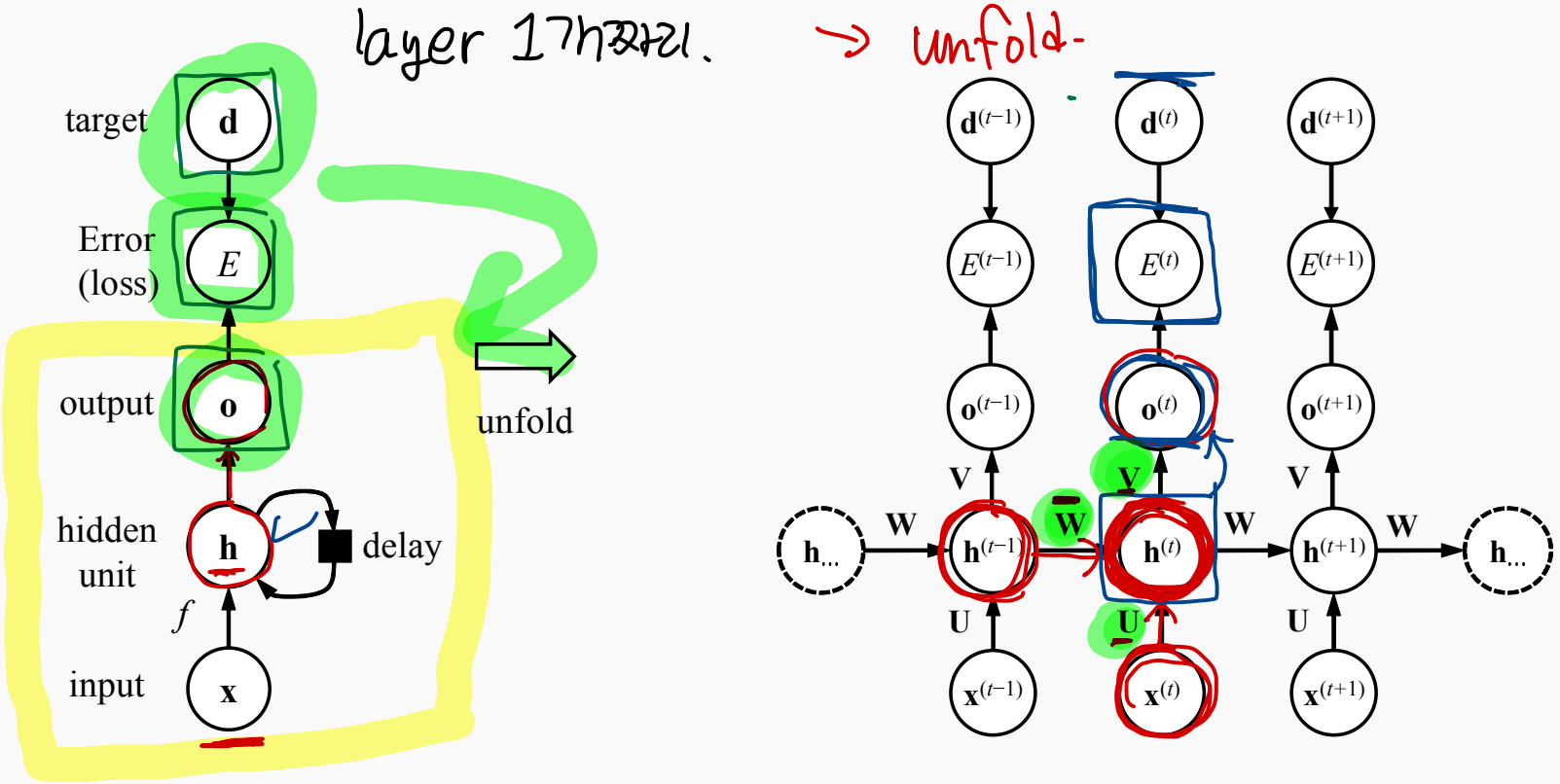
Recurrent neuron with hidden unit

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta}) = g^{(t)}(\mathbf{x}^{(t)}, \mathbf{x}^{(t-1)}, \ldots, \mathbf{x}^{(2)}, \mathbf{x}^{(1)}) \tag{5}$$





Graph representation of RNN



- 가중치 매트릭스 $\mathbf{U}, \mathbf{V}, \mathbf{W}$ or $\mathbf{W}_{xh}, \mathbf{W}_{hy}, \mathbf{W}_{hh}$
- Error E = 예측값과 참값 사이의 cross entropy with $\mathbf{y} = \mathrm{softmax}(\mathbf{o})$

Various architectures of RNN

one to one

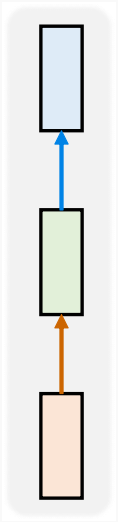


Image Classification (Vanilla Neural Networks)

one to many

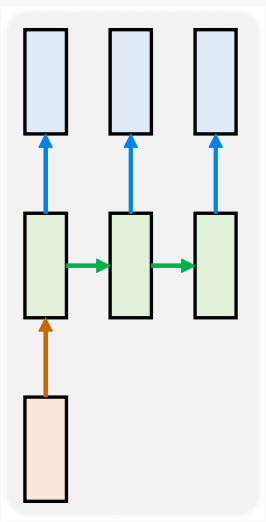
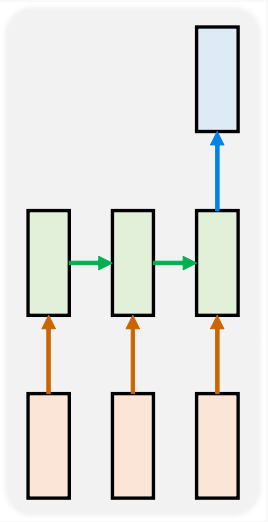


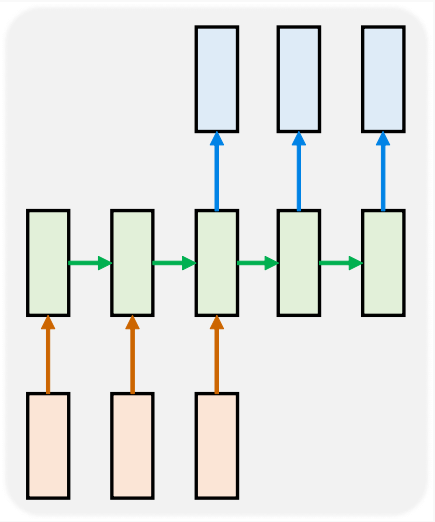
Image Captioning (image → seq. of words

many to one



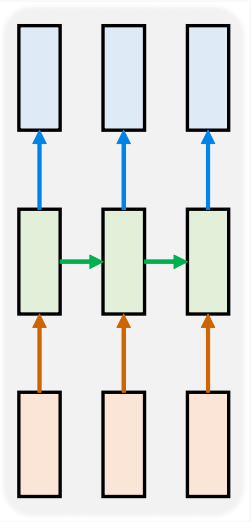
Sentiment Analysis (seq. of words → sentiment)

many to many



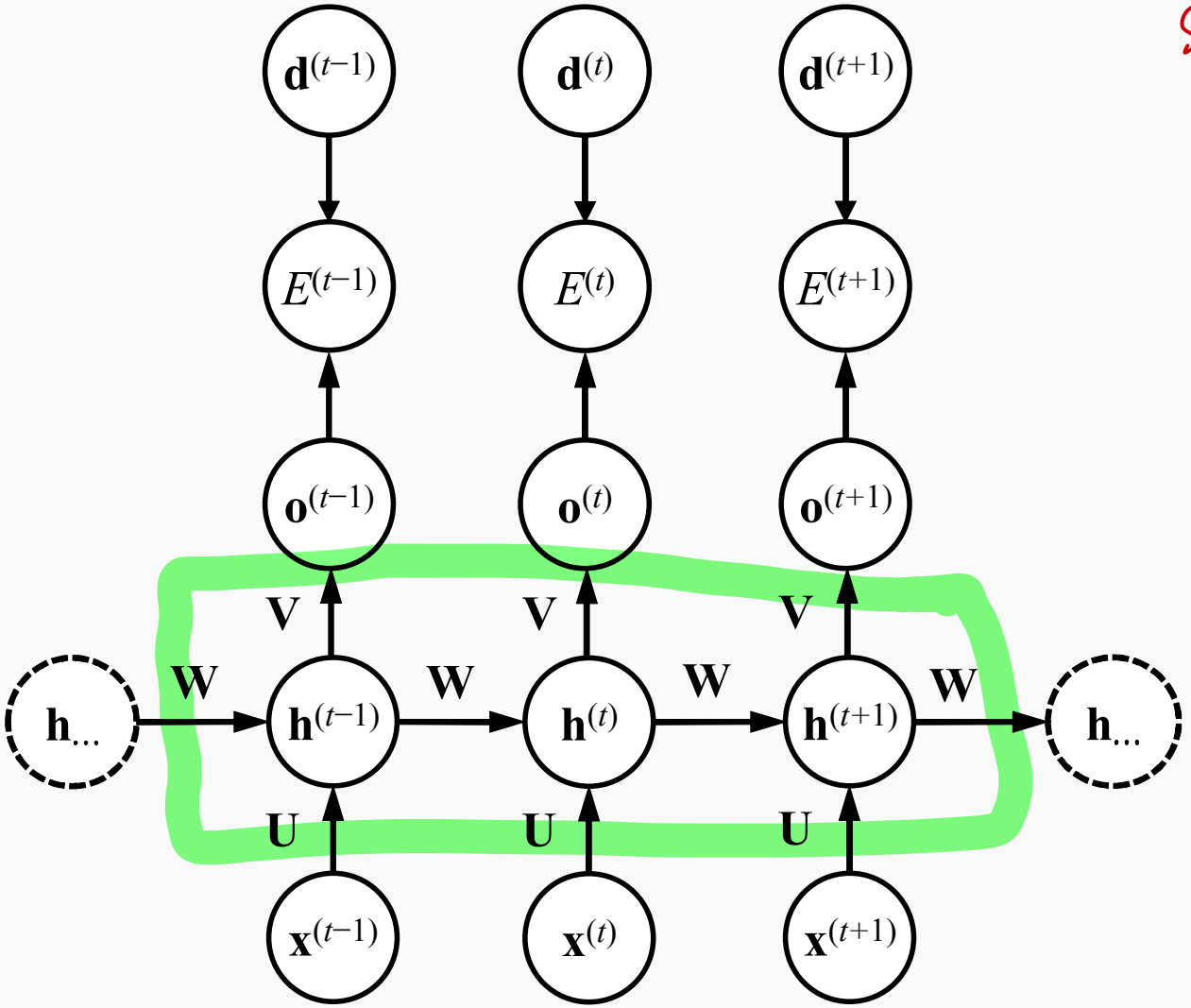
Machine Translation (seq. of words → seq. of words)

many to many



Synced Sequence (video classification on frame level)

Forward propagation



Shallow RNN.

Forward propagation

Assuming the initial state $\mathbf{h}^{(0)} = \mathbf{0}$

$$\mathbf{a}^{(t)} = \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} + \mathbf{b} \tag{6}$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)}) \tag{7}$$

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)} + \mathbf{c} \tag{8}$$

$$\mathbf{y}^{(t)} = \mathrm{softmax}(\mathbf{o}^{(t)}) \tag{9}$$

Alternatively

$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}_{hh}\mathbf{h}^{(t-1)} + \mathbf{W}_{xh}\mathbf{x}^{(t)}) \tag{10}$$

$$\mathbf{y}^{(t)} = \mathrm{softmax}(\mathbf{W}_{hy}\mathbf{h}^{(t)}) \tag{11}$$

Error (loss) function: cross entropy

$$E^{(t)}(\mathbf{y}^{(t)}, \mathbf{d}^{(t)}) = -\mathbf{d}^{(t)} \log \mathbf{y}^{(t)} = -\sum_{i=1}^{D} d_t^{(t)} \log y_i^{(t)} \tag{12}$$

$$\Rightarrow E(\mathbf{y}, \mathbf{d}) = -\sum_{t=1}^{N} \mathbf{d}^{(t)} \log \mathbf{y}^{(t)} \tag{13}$$

Deep RNNs

Deep RNN

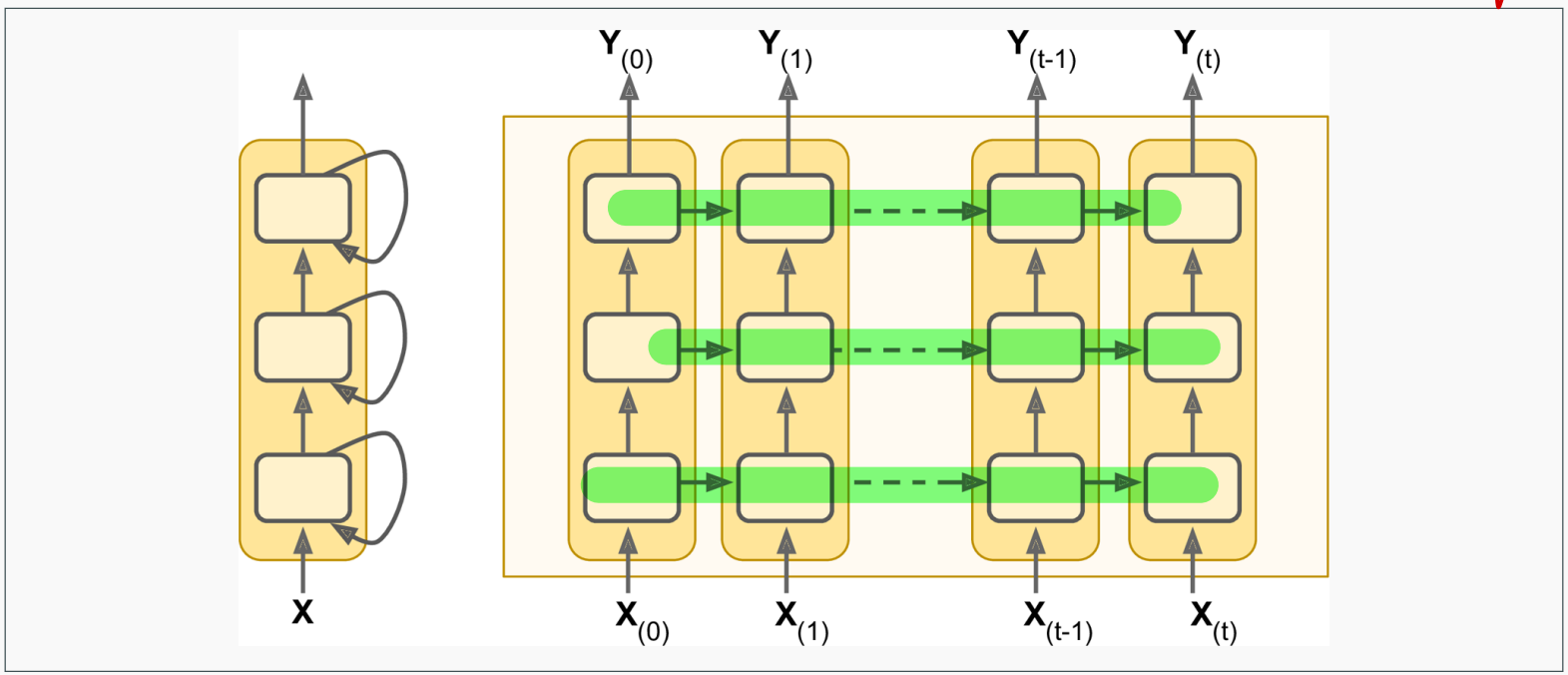
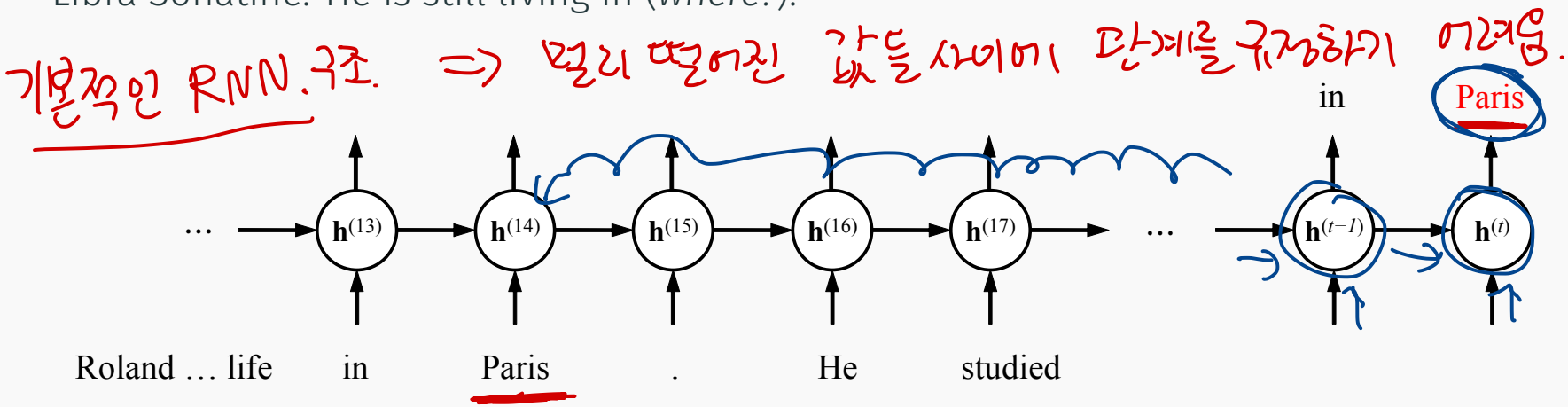


Figure 14-12. Deep RNN (left), unrolled through time (right)

Long-term dependency problem

Roland Dyens was born in Tunisia and lived most of his life in Paris.

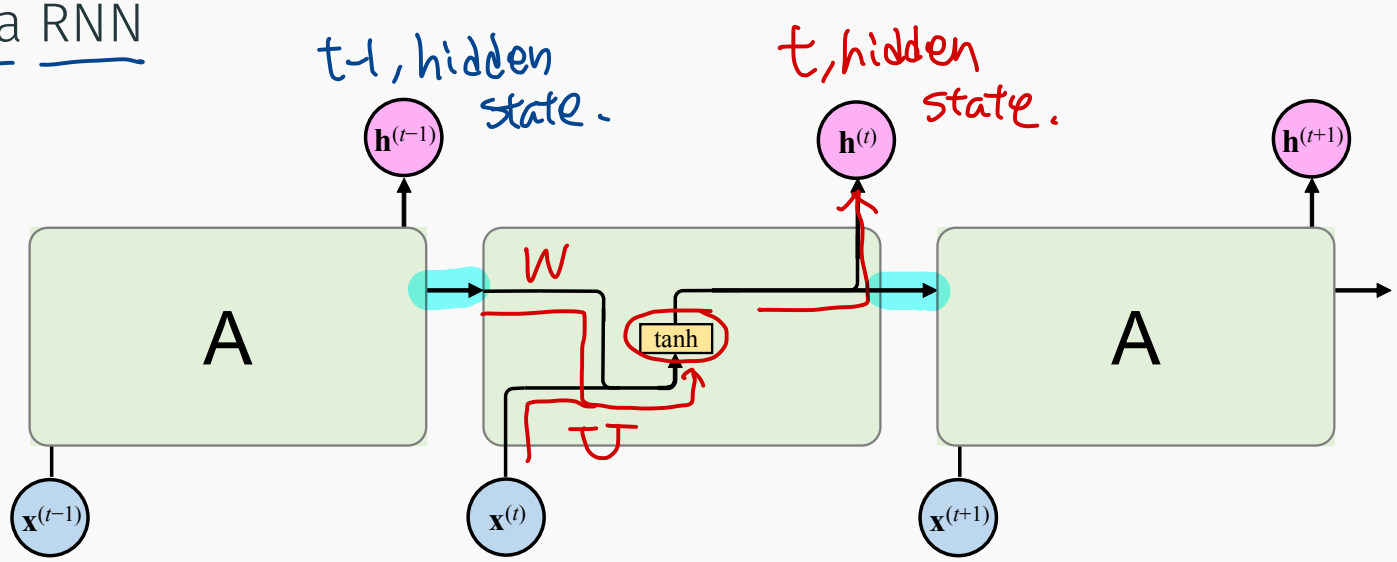
He studied with Spanish classical guitarist Alberto Ponce and with Désiré Dondeyne. As a performer, Dyens was known for improvisation. Sometimes he opened his concerts with an improvised piece, and he might improvise the program itself, without planning or announcing beforehand what he would be playing. He said that a journalist once told him he had the hands of a classical musician and the head of a jazz musician. He played Bach suites and he played with jazz musicians at the Arvika Festival in Sweden. A heavy metal band did a version of the third movement of his Libra Sonatine. He is still living in (*where?*).



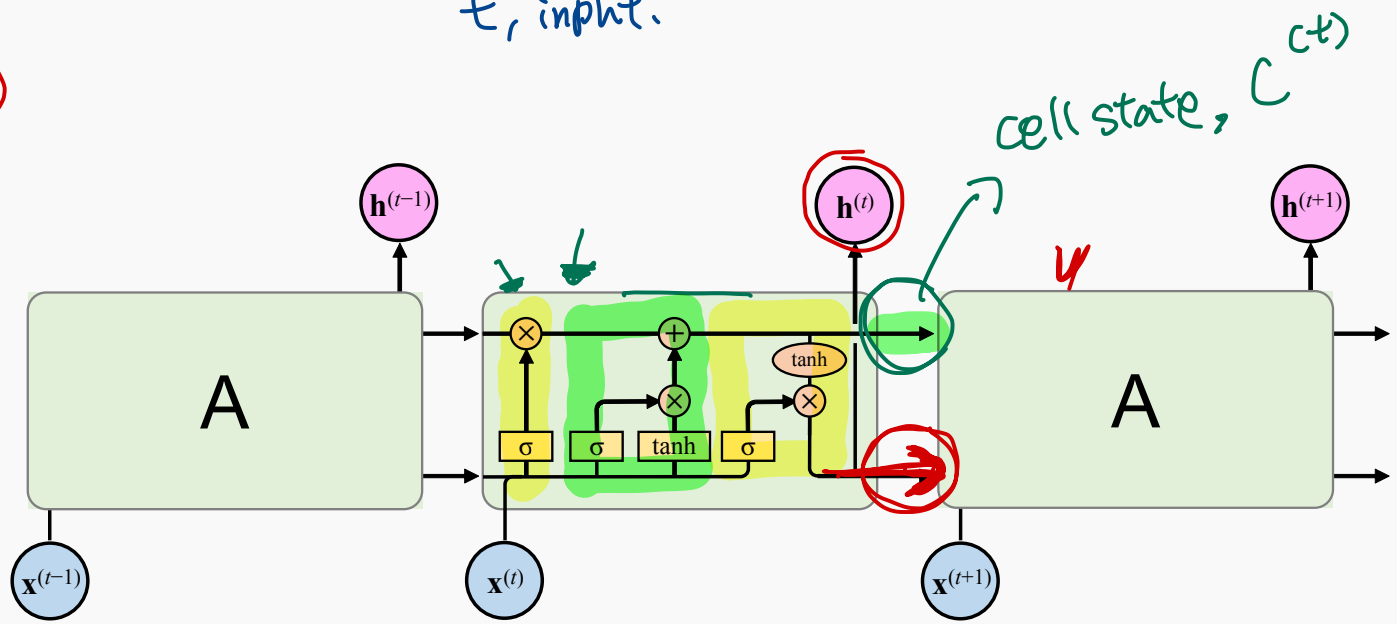
질문 단어로 부터 'Paris' 는 116 시간 스탭이나 멀리 떨어져 있다. ⟹ Gradient 는 그렇게까지 멀리 역전파되지 못한다. (**Vanishing/Exploding Gradient**)

Long short-term memory (LSTM)

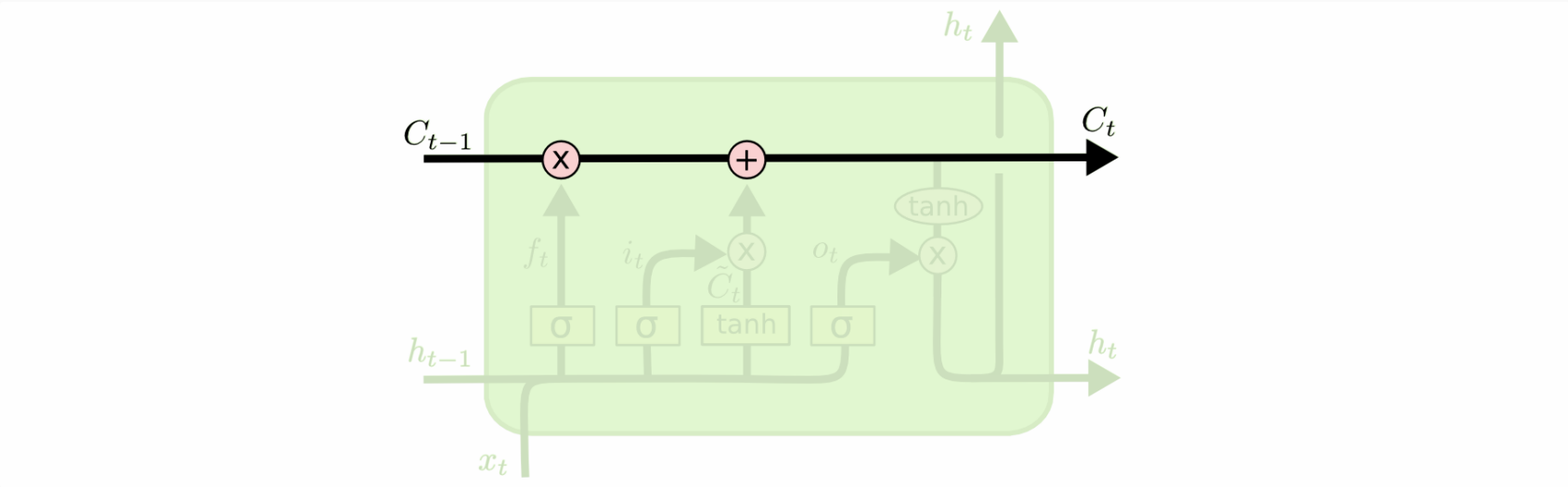
Vanilla RNN



LSTM

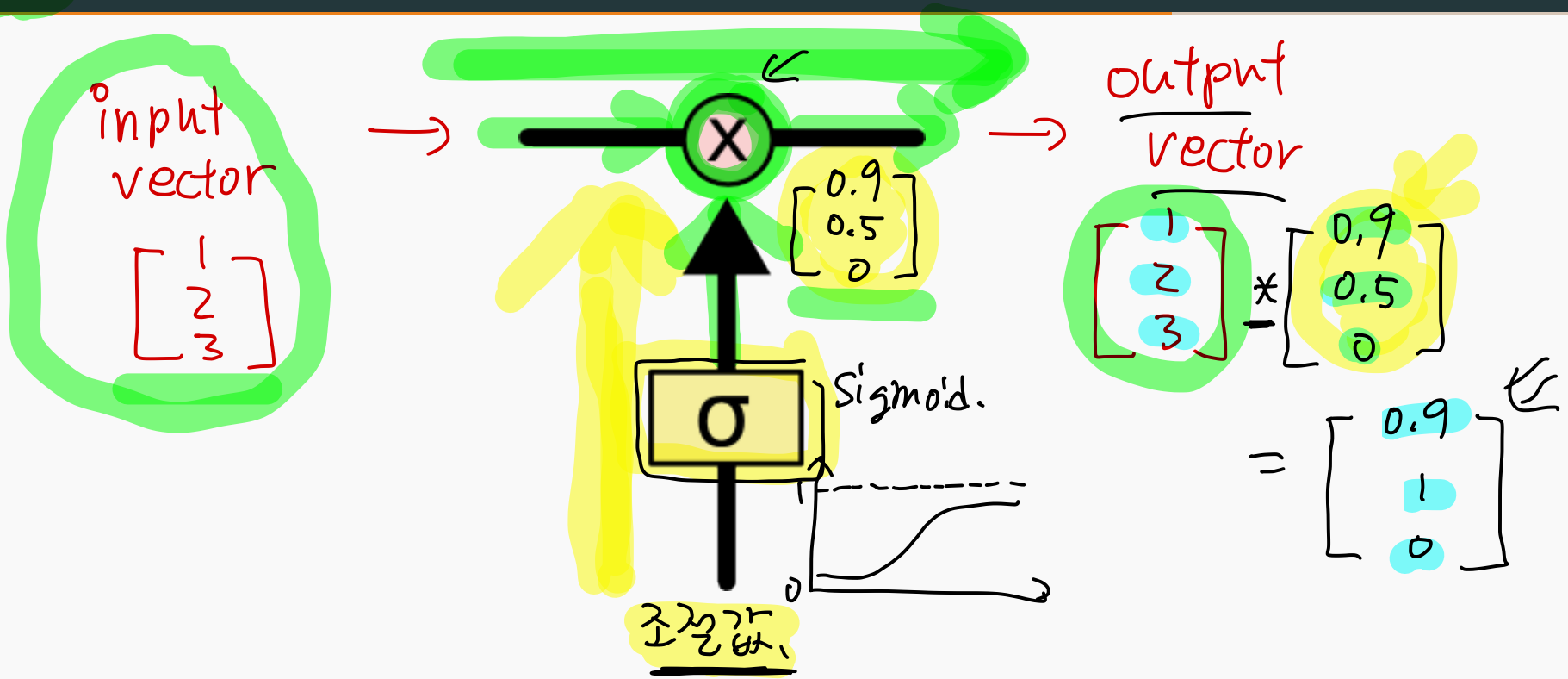


Cell state



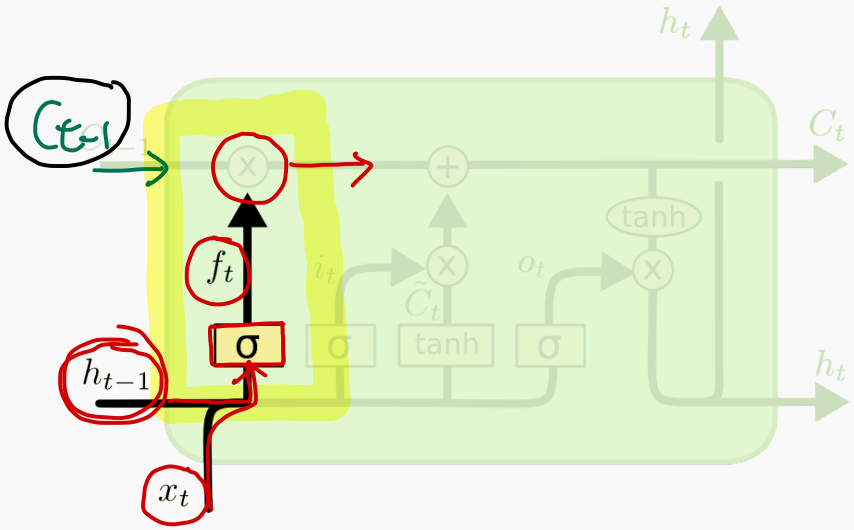
- 컨베이어 벨트 같은 통로가 있어서, 정보가 전체 연결망을 그대로 흘러다닐 수 있음
- 정보가 변경 없이 그대로 흘러다닐 수 있음
- Cell state에 정보를 지우거나 더할 수 있음

Gate



- 정보의 흐름을 제어하는 gate
- Sigmoid 함수 σ와 pointwise multiplication으로 이루어져 있음
- Sigmoid의 출력에 따라
  - $\sigma = 0$일 때, 흐름을 중단함
  - $\sigma = 1$일 때, 흐름을 모두 통과

Forget gate layer

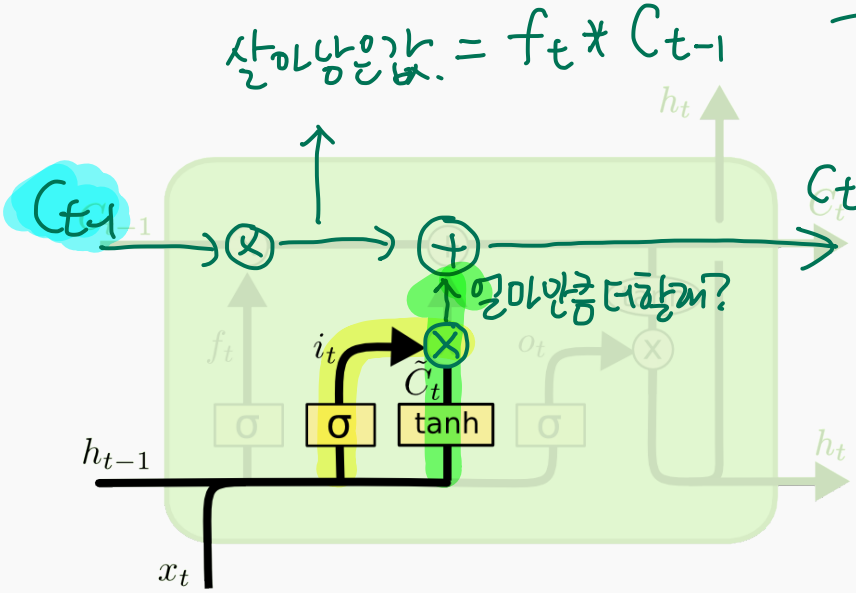


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

= 직전 cell state C_{t-1}를 얼마나 살릴거냐를 결정하는 값.

- Output of sigmoid $f_t \in [0, 1]$
  - 1: completely keep this
  - 0: completely get rid of this

Input gate layer



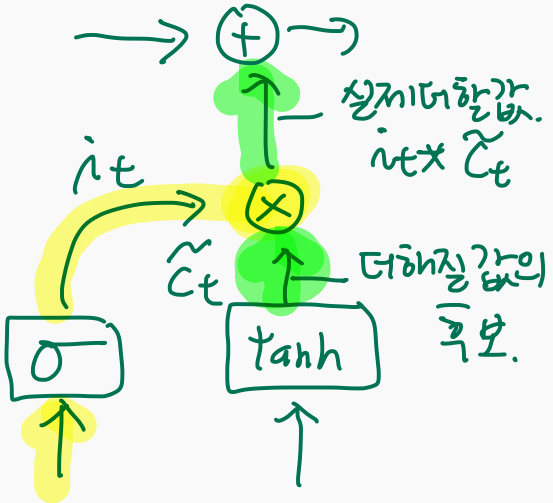
살아남은값 / 새로더해진값

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

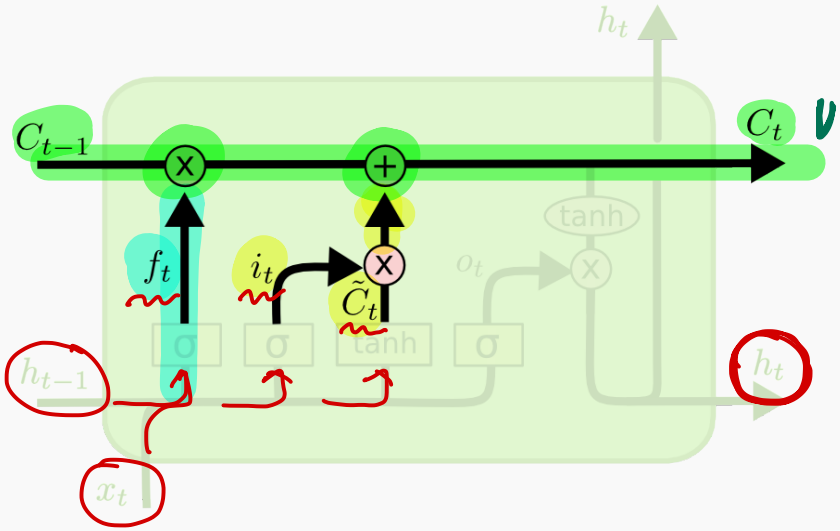
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

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

- 어떤 새로운 정보를 입력시킬 지 결정
- 입력 게이트는 어떤 값을 업데이트 할 지 결정
- tanh 층은 새로운 값의 후보가 되는 벡터를 생성



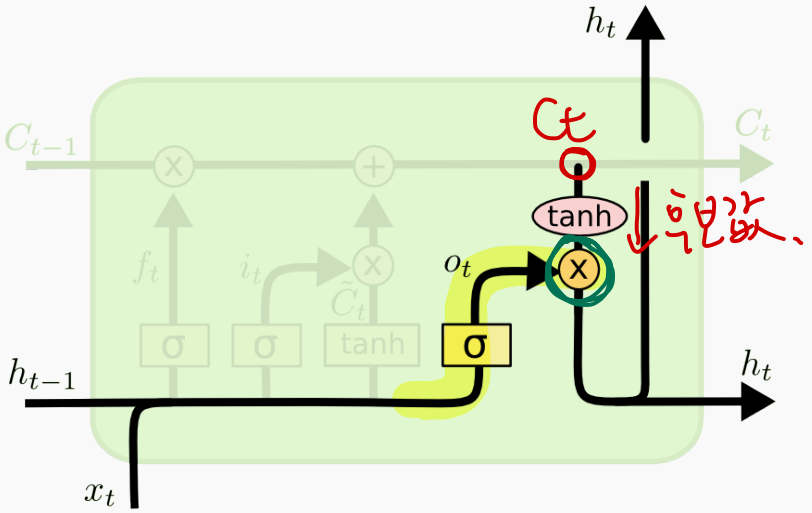
Update cell state



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

- Cell state가 업데이트 되어 다음 시간 스텝으로 넘어가는 과정
- 동시에 새로운 hidden state를 출력하기 위해 $\tanh$로 전달됨

Output hidden state



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

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

- 현재의 cell state 중에서 출력 h로 내보낼 값을 선택
- 이 출력은 윗 층과 다음 시간 스텝으로 동시에 값을 보냄

Appendix

Reference and further reading

- “Chap 14 | Recurrent Neural Networks” of A. Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow
- “Lecture 11 | Recurrent Neural Networks” of Kwang Il Kim, Machine Learning (2019)
- Christopher Olah, Understanding LSTM Networks (link)