Machine Learning

RNN

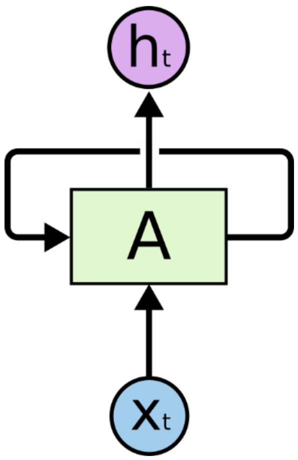
김선녕(sykim.lecture@gmail.com)

RNN

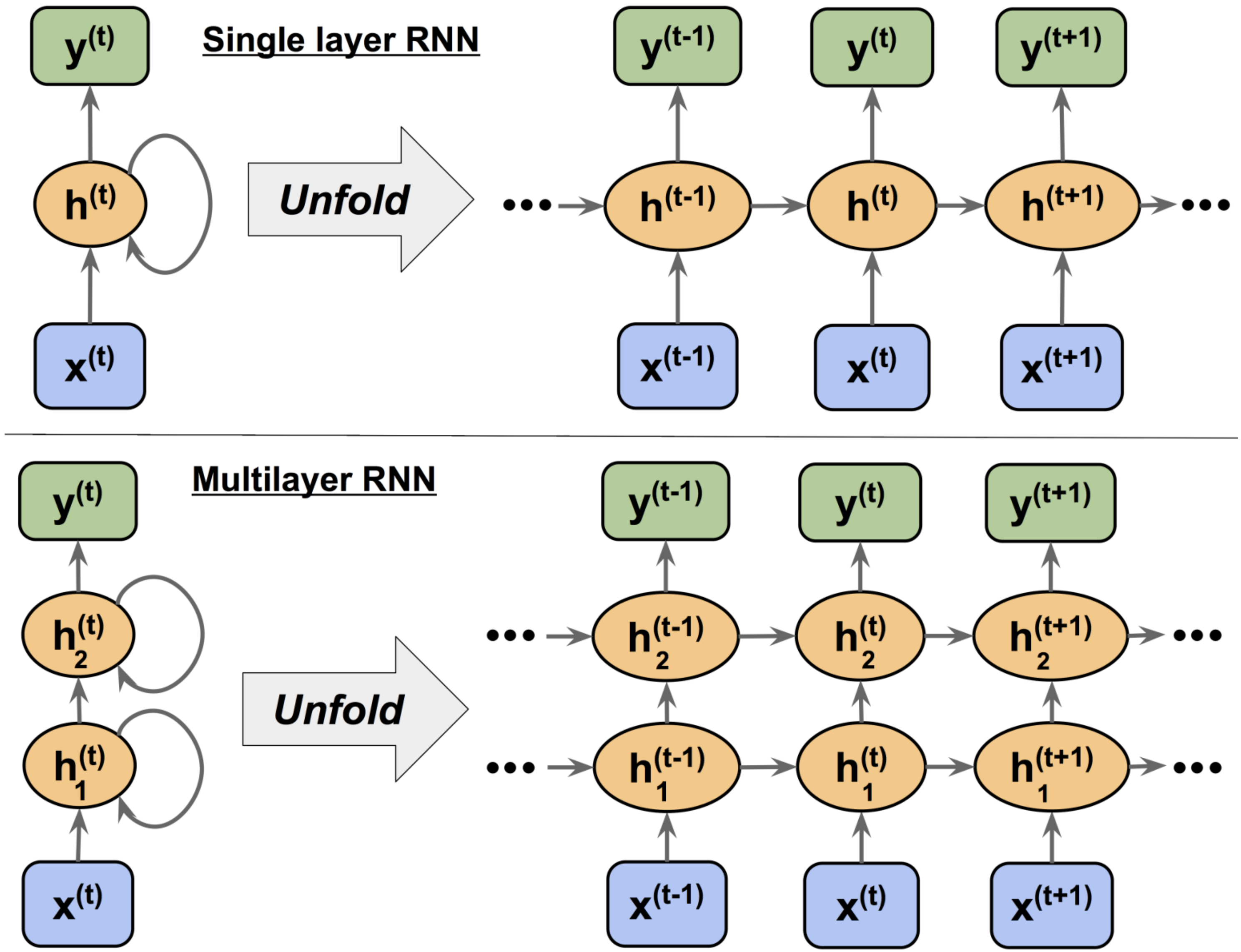
LSTM

개요

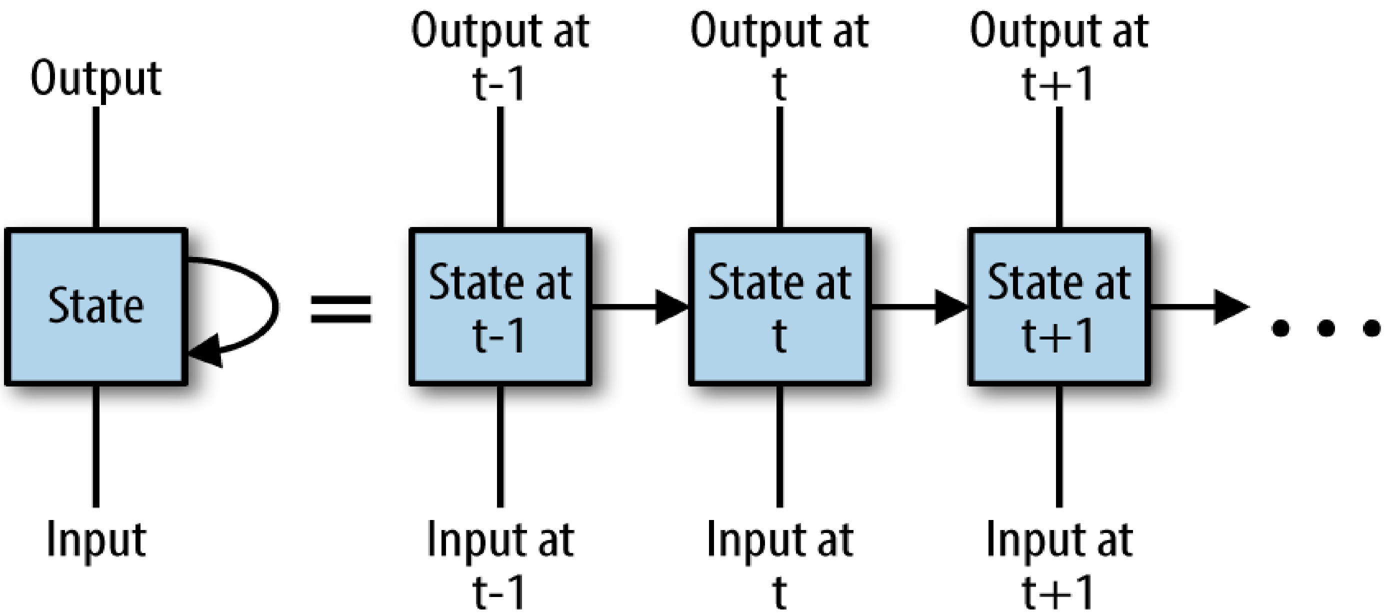
- 순차적인 데이터(sequential data)를 학습하여 분류 또는 예측을 수행
  - Recurrent Neural Networks have loops.
- 각 레이어마다 파라미터(Parameter)들이 독립적이었으나 RNN은 공유.
- 과거의 데이터가 미래에 영향을 줄 수 있는 구조
- 응용
  - 음성, 자연어 문장, 동영상, 주가 변동등의 시계열(time series)데이터를 분석하여 분류 및 예측
  - 자율주행 시스템에서 차의 이동 경로를 예측
  - 문장, 문서, 오디오 샘플을 입력 받을 수 있고, 자동번역, 스피치 투 텍스트같은 자연어 처리에 매우 유용
  - 기계번역, 음성 인식, 필기체 인식, 영상 주석달기, 동영상에서 행동인식, 작곡 및 작사 등 다양한 응용분야에서 활용



RNN 구조



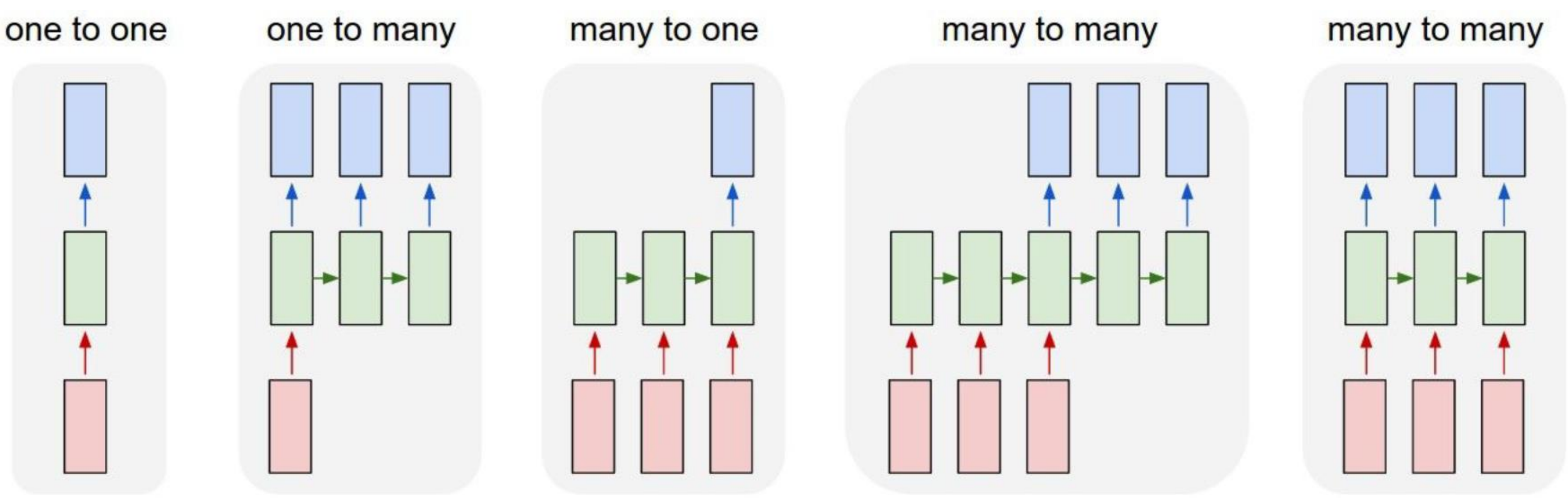
ref : python machine learning 2nd Ed



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Process Sequences

- **one to one** : from fixed-sized input to fixed-sized output(Vanilla mode)
  - e.g. image classification
- **one to many** : sequence output
  - e.g. image captioning takes an image and outputs a sentence of words
- **many to one** : sequence input
  - e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment
- **many to many** : sequence input and sequence output
  - e.g. machine translation: an RNN reads a sentence in English and then outputs a sentence in French
- **many to many** : synced sequence input and output
  - e.g. video classification where we wish to label each frame of the video

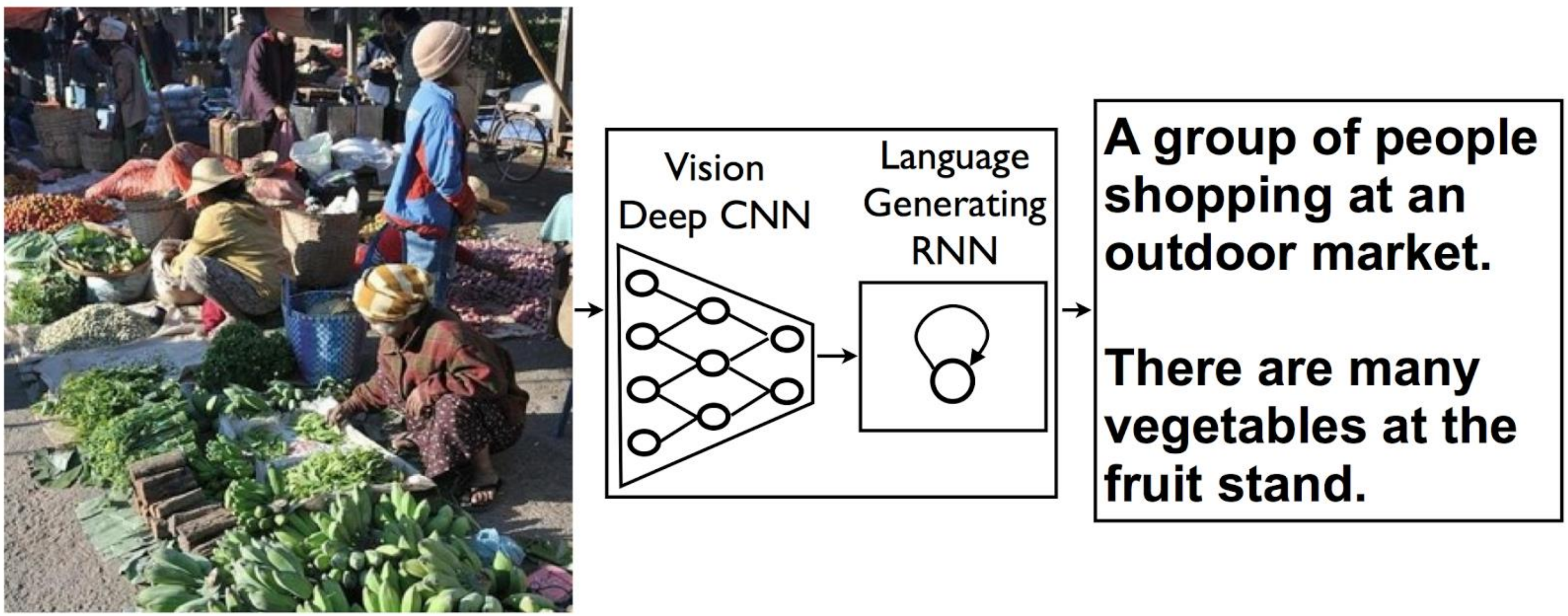


red : input vectors, blue : output vectors, green : RNN's state

http://cs231n.stanford.edu/

Example : one to many

- CNN과 RNN 조합
- $image\ captioning(one\ to\ many)$



http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

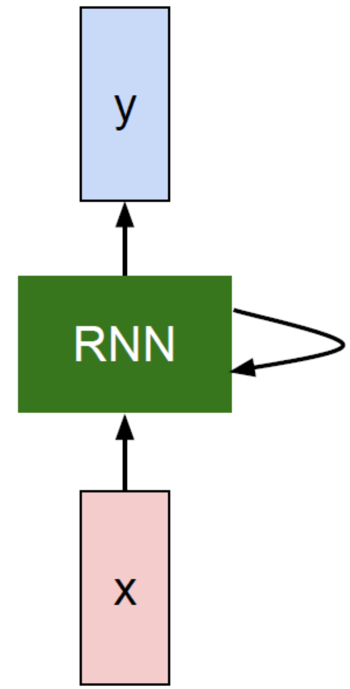
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

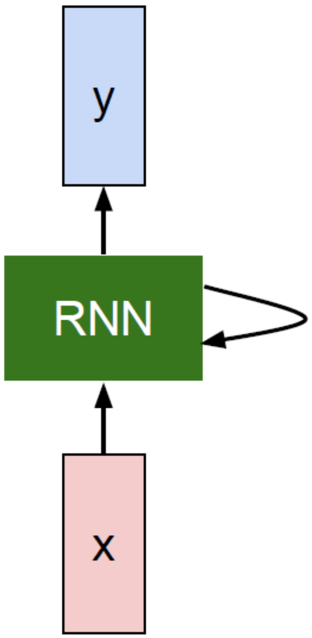
input vector at some time step

some function with parameters W



(Simple) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:



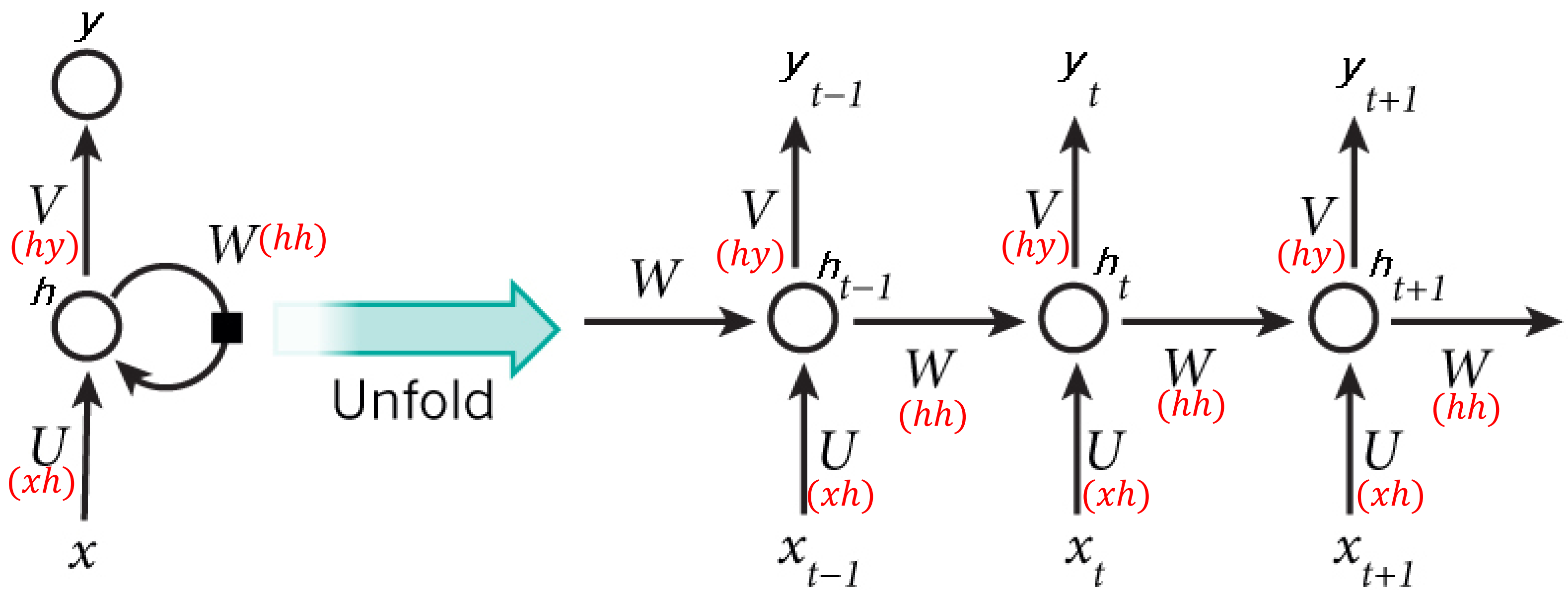
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$

Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

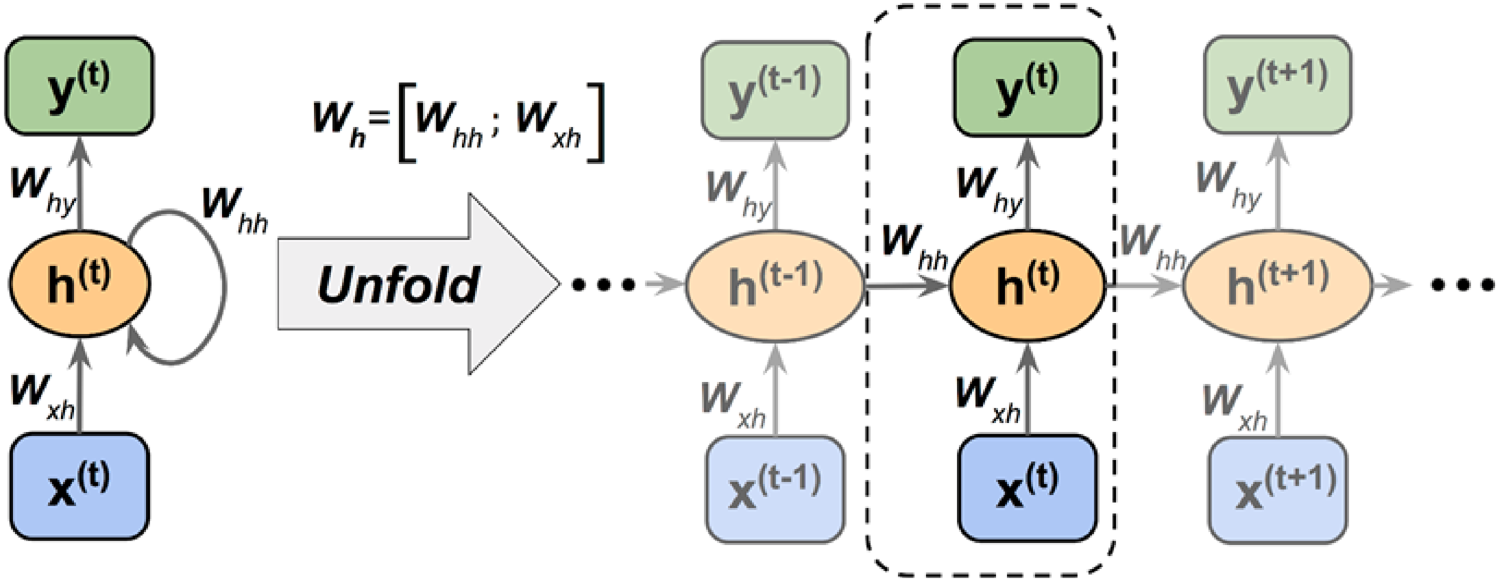
RNN의 기본



A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

- x_t : t에서의 입력 값
- h_t : t에서의 *hidden state*. 네트워크의 메모리(과거 시간 스텝들에서 일어난 정보).
  - $t-1$의 *hidden state*값과 현재 t의 입력값(x_t)에 의해 계산.
  - $h_t = tanh(Ux_t + Ws_{t-1})$
  - 최초(s_{-1})는 0으로 초기화
- y_t : t에서의 출력 값. 현재시간 t의 메모리만 의존
  - $y_t = softmax(Vh_t)$
- 모든 시간 스텝에 대해 파라미터 값 공유 (U, V, W)

활성화 출력계산



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

$$\downarrow$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$

$$fw_h{}^t = W_{xh} x^t + W_{hh} h^{t-1} + b_h$$

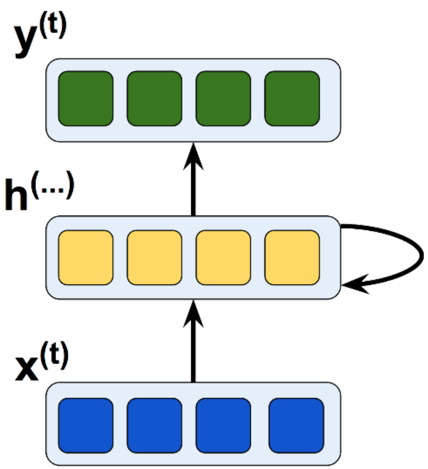
$$h^t = \phi_h\left(fw_h{}^t\right) = \phi_h(W_{xh} x^t + W_{hh} h^{t-1} + b_h)$$

$$W_h = [W_{xh}\ ;\ W_{hh}]$$

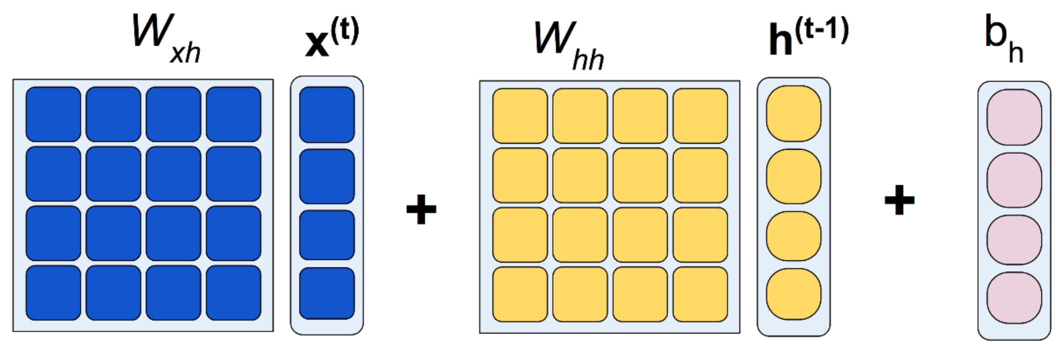
$$\boldsymbol{h^t} = \phi_h\left([W_{xh}\ ;\ W_{hh}]\begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} + b_h\right)$$

$$y^t = \phi_y\left(W_{hy}\boldsymbol{h^t} + b_y\right)$$

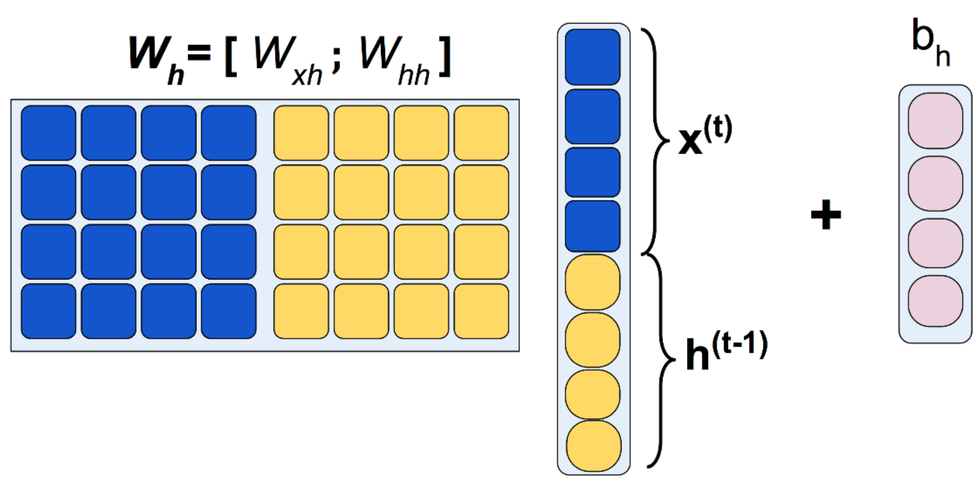
활성화 출력계산



Formulation 1: $\mathbf{h}^{(t)} = \phi_h\left(W_{xh}\mathbf{x}^{(t)} + W_{hh}\mathbf{h}^{(t-1)} + b_h \right)$



Formulation 2: $\mathbf{h}^{(t)} = \phi_h\left(\boldsymbol{W_h} \left[\mathbf{x}^{(t)}; \mathbf{h}^{(t-1)}\right]^T + b_h \right)$



Final Output: $\mathbf{y}^{(t)} = \phi_y\left(W_{hy}\, \mathbf{h}^{(t)} + b_y\right)$

ref : python machine learning 2nd Ed

- m개의 단어로 이루어져 있는 문장이 있다면 특정데이터셋에서 이 문장이 나타날 확률

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

(Ex) He went to buy some chocolate의 문장 확률은?

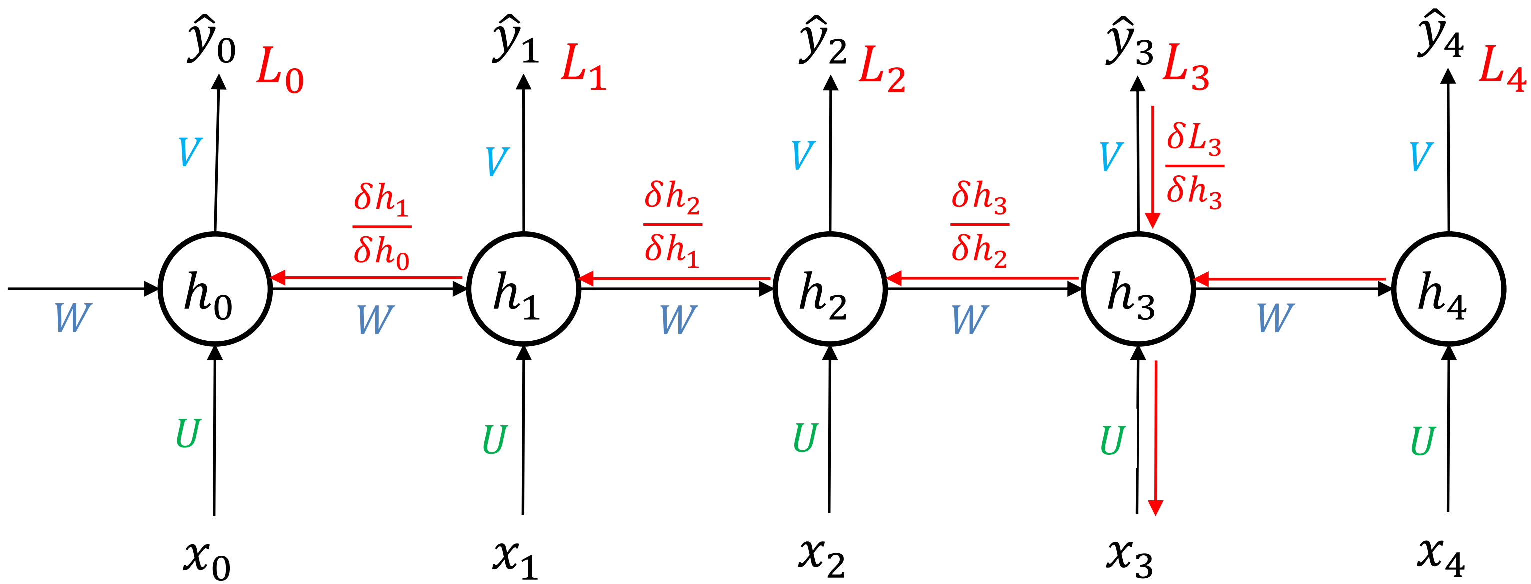
  - He went to buy some 주어졌을 때 chocolate 확률 곱하기
  - He went to buy 주어졌을 때 some 확률 곱하기

    …

  - 문장의 시작 He의 확률까지의 곱

- 단어들의 확률은 이전에 나왔던 **모든** 단어들에 의존
- 하지만 실제 구현에서는 많은 모델들이 계산량, 메모리 문제 등으로 인해 long-term dependency를 효과적으로 다루지 못해서 긴 시퀀스는 처리하는 것이 힘들다.

BPTT(BackPropagation Through Time)



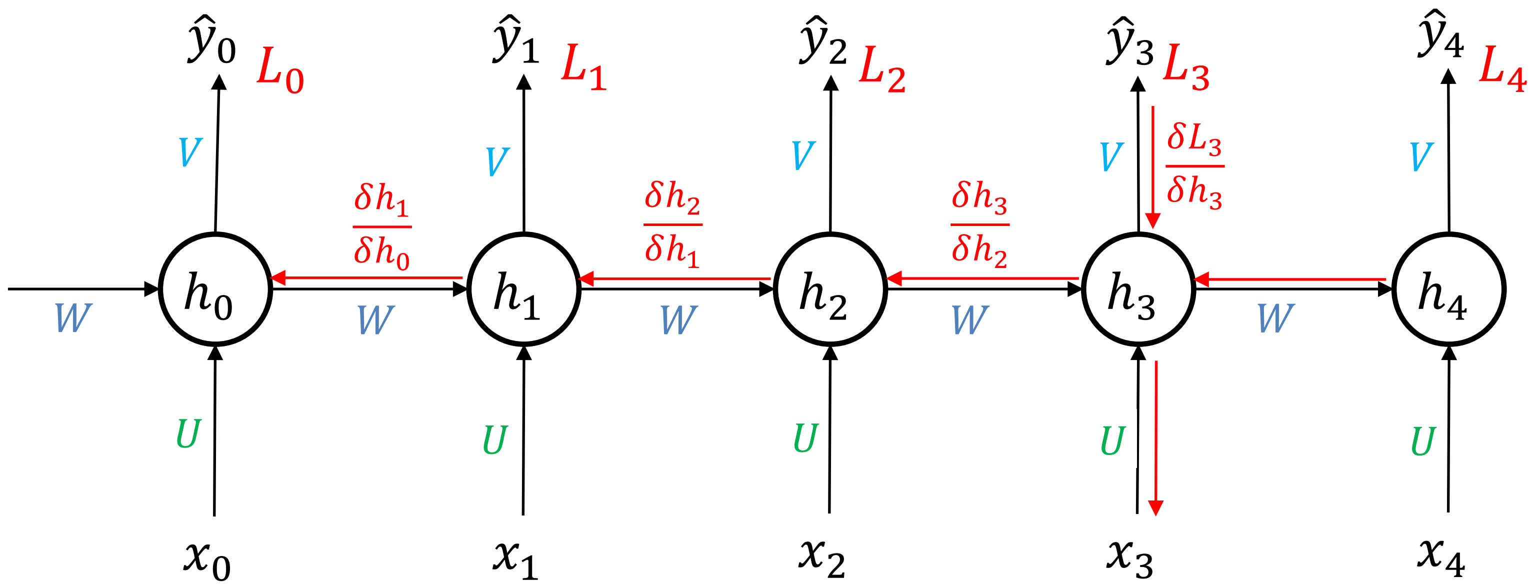
- RNN 모델을 학습하는데 사용되는 핵심 알고리즘
- 전체 손실(E) : $t = 1$에서 $t = T$까지 타임 스텝의 모든 손실함수 합

$$L = \sum_{t=1}^{T} L^t$$

$$\frac{\delta L_3}{\delta V} = \frac{\delta L_3}{\delta \hat{y}_3}\frac{\delta \hat{y}_3}{\delta V} = \frac{\delta L_3}{\delta \hat{y}_3}\frac{\delta \hat{y}_3}{\delta z_3}\frac{\delta z_3}{\delta V} = (\hat{y}_3 - y_3) \otimes s_3$$

$z_3 = Vh_3$, $\otimes$: 두 벡터의 외적

BPTT(BackPropagation Through Time)

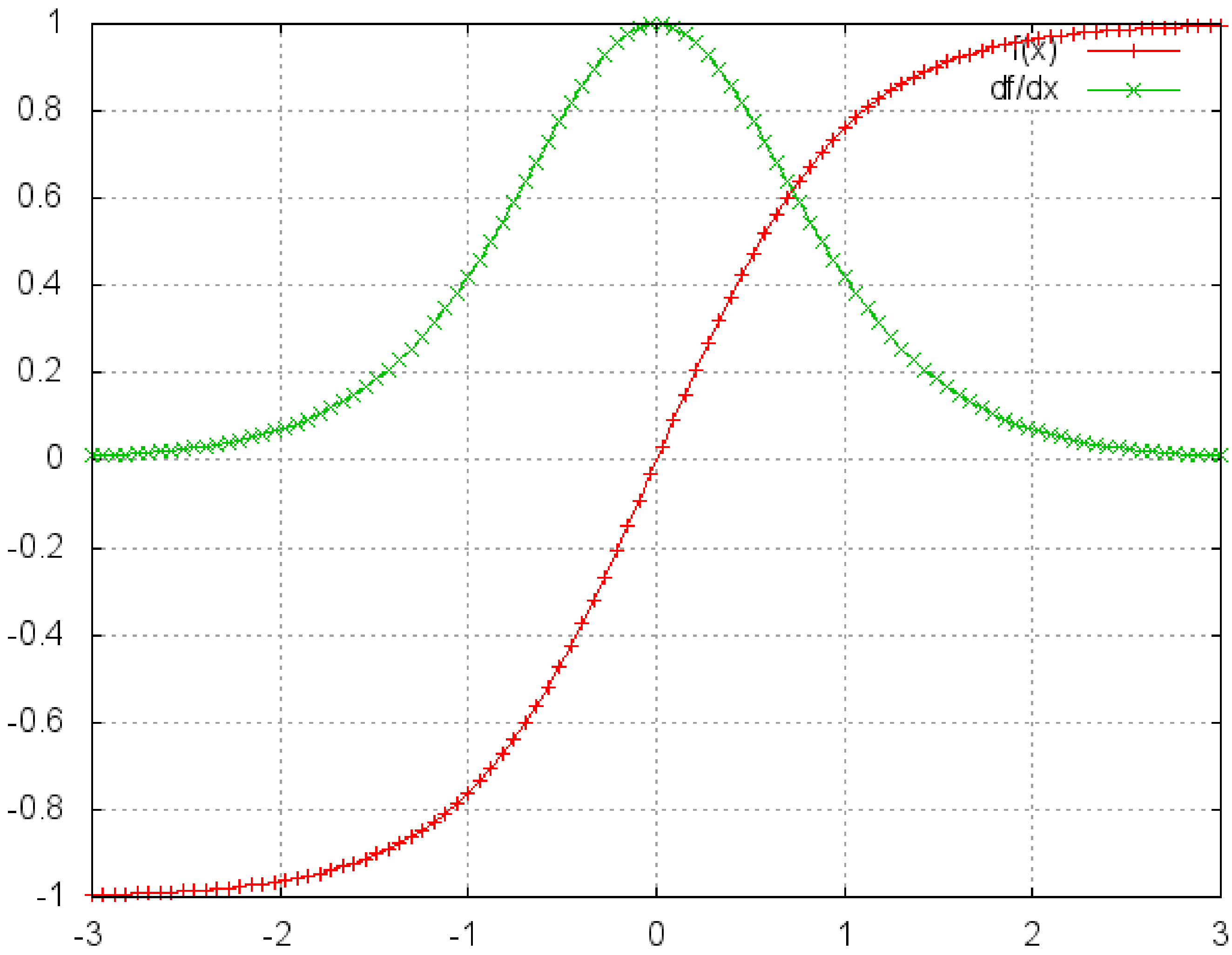


$$\frac{\delta L_3}{\delta W} = \frac{\delta L_3}{\delta \hat{y}_3}\frac{\delta \hat{y}_3}{\delta h_3}\frac{\delta h_3}{\delta W}$$

$h_t = tanh(Ux_t + Wh_{t-1})$: h_3는 h_2에 의존, h_2는h_1에 의존

$$\frac{\delta L_3}{\delta W} = \sum_{t=1}^{3}\frac{\delta L_3}{\delta \hat{y}_3}\frac{\delta \hat{y}_3}{\delta h_3}\frac{\delta h_3}{\delta h_k}\frac{\delta h_k}{\delta W}$$

tanh and derivative



https://nn.readthedocs.io/en/rtd/transfer/

Vanishing Gradient

타임 스텝 t에서 손실은 모든 이전 타임 스텝 $1: t-1$의 유닛에 의존

$$\frac{\boldsymbol{\delta L_t}}{\boldsymbol{\delta W}} = \frac{\boldsymbol{\delta L_t}}{\delta \hat{y}_t} \times \frac{\delta \hat{y}_t}{\delta h_t} \times \left(\sum_{k=1}^{t} \frac{\delta h_t}{\delta h_k} \times \frac{\delta h_k}{\boldsymbol{\delta W}} \right), \qquad \frac{\boldsymbol{\delta h^t}}{\boldsymbol{\delta h_k}} = \prod_{i=k+1}^{t} \frac{\delta h_i}{\delta h_{i-1}}$$

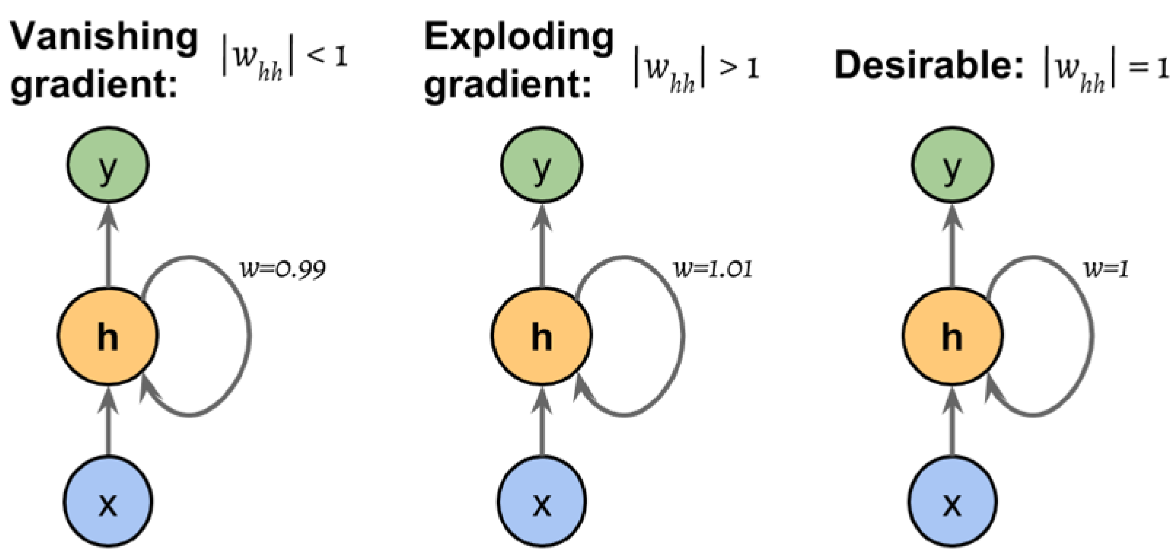
$\frac{\delta h^t}{\delta h^k}$(이전타임스텝의곱)로 인하여 *vanishing gradient*발생

$\frac{\delta h^t}{\delta h^k}$는 k개의 곱셈. 즉, 가중치 w가 $t-k$번 곱해져서 w^{t-1}이 된다

$|w| < 1$이면 $t-k$가 클 때 w^{t-k} 값이 매우 작아진다.

$|w| > 1$이면 $t-k$ 가 클 때 w^{t-k} 값이 매우 커진다

$t-k$ 크다는 것은 긴 시간 의존성을 가진다는 의미. **해결책 : $|w| = 1$**



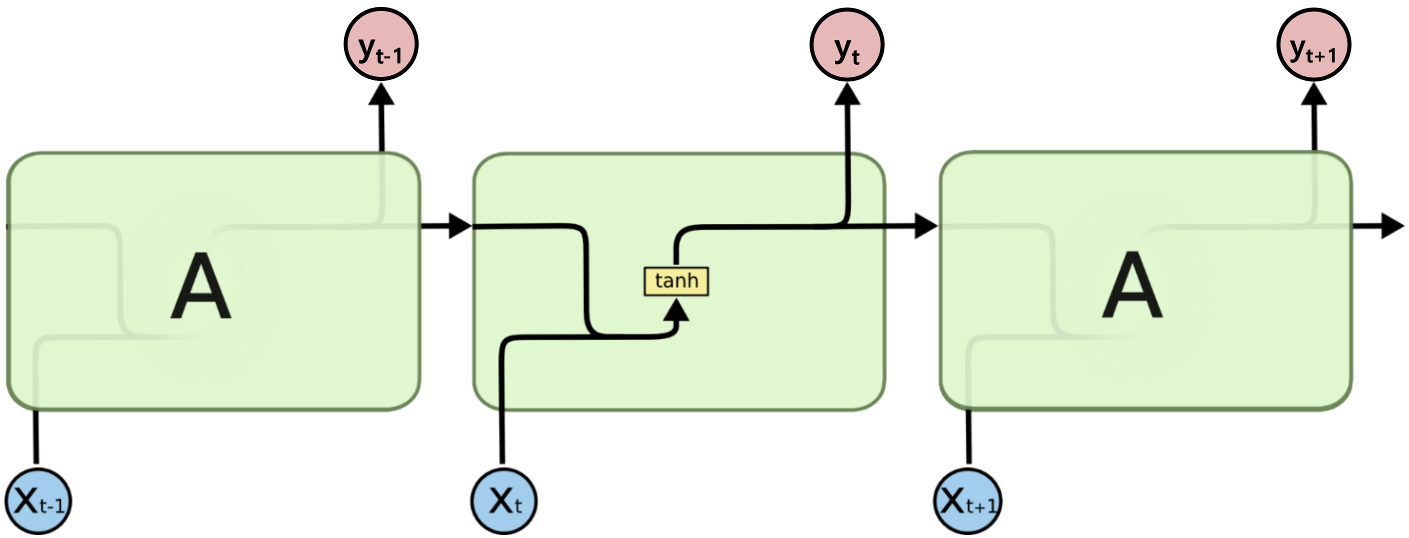
RNN

LSTM

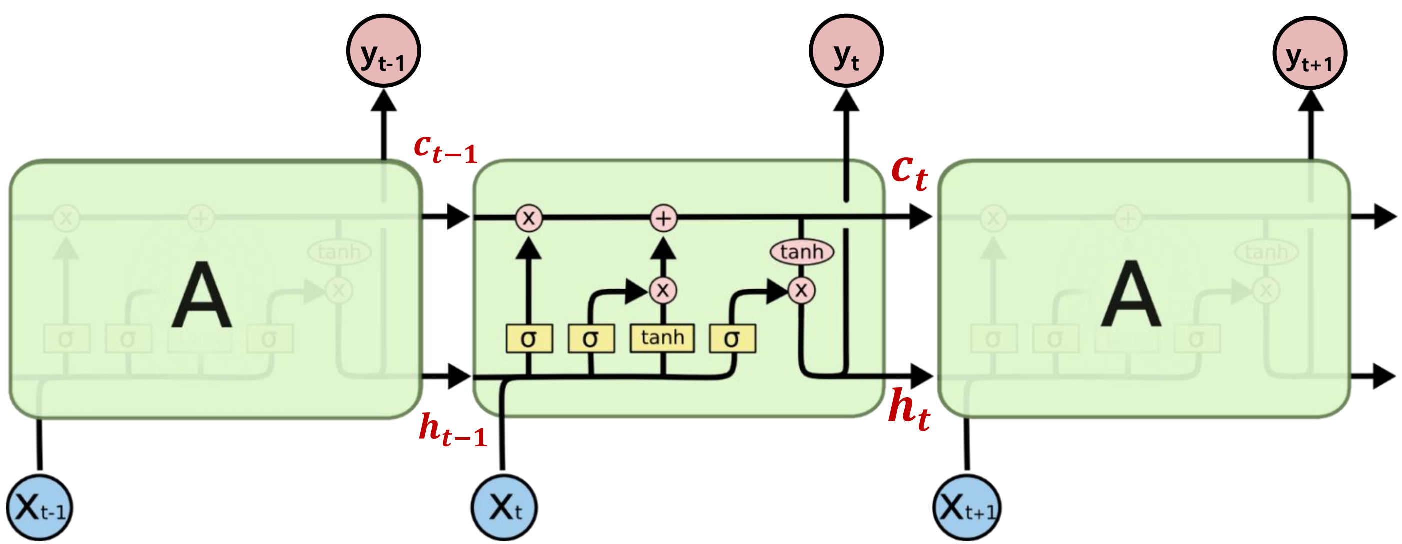
LSTM

- RNN의 Gradient Problem 해결책
  - $T - BPTT(Truncated\ BackPropagation\ Through\ time)$
  - $\boldsymbol{LSTM(Long\ Short - Term\ Memory)}$
- LSTM의 중요요소 : 메모리 셀($\boldsymbol{memory\ cell}$) - 은닉층
  - 각 메모리 셀에 적절한 가중치 $w = 1$를 유지하는 게이트
    - $\boldsymbol{cell\ state}$: 게이트의 출력
  - 저장할 것, 버릴 것, 읽어 들일 것을 학습하는 것
  - h_t: 단기상태($short - term\ state$)
  - c_t: 장기상태($long - term\ state$)
  - y_t: 출력

RNN vs LSTM



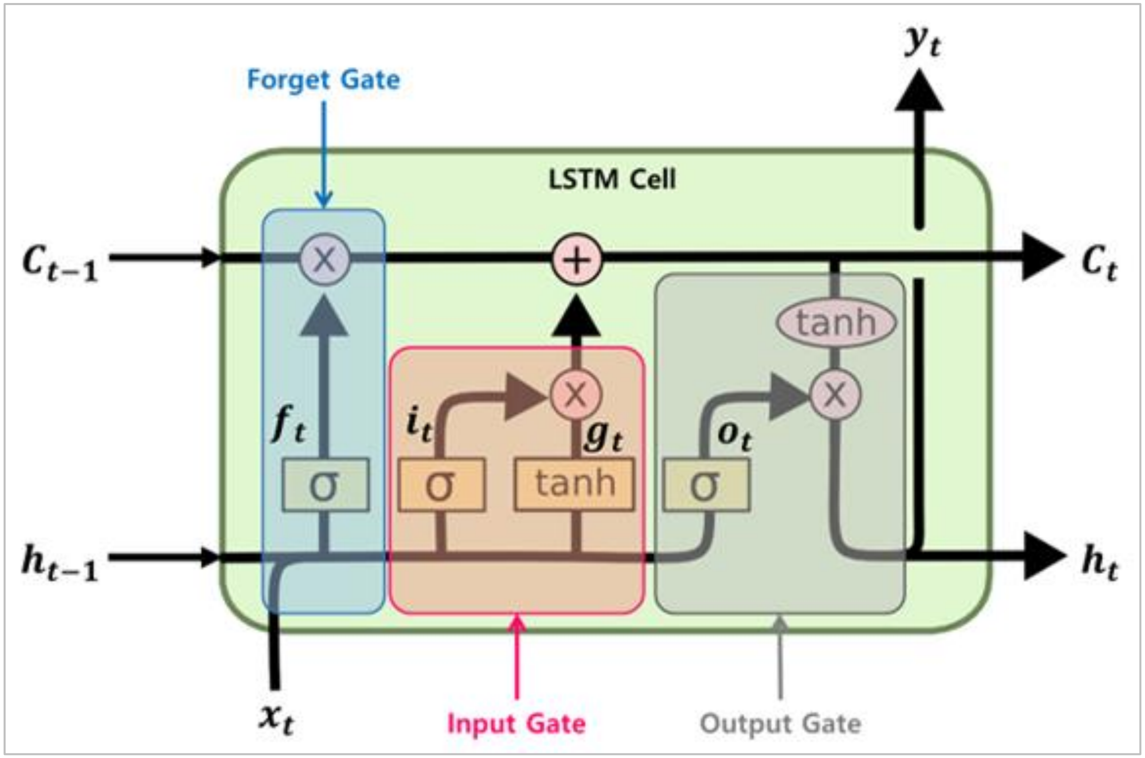
yt-1
yt
yt+1
A
tanh
A
Xt-1
Xt
Xt+1



yt-1
yt
yt+1
c_{t-1}
c_t
A
σ
σ
tanh
σ
tanh
A
h_{t-1}
h_t
Xt-1
Xt
Xt+1

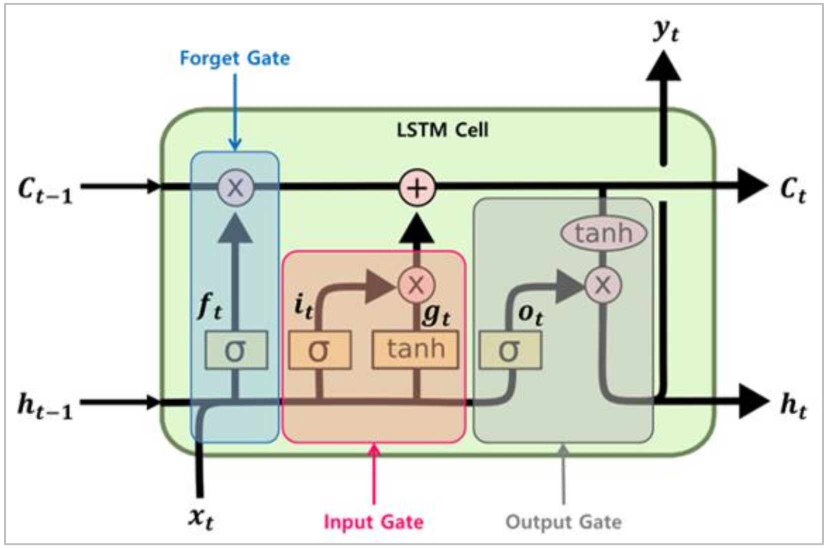
cell state

- 이전 타임 스텝의 셀 상태(c_{t-1}) $\xrightarrow[\text{(가중치 계산없이)}]{}$ 현재 타임 스텝 셀 상태 (c_t) 생성
  - 네트워크를 왼쪽에서 오른쪽을 관통하면서 삭제 게이트를 지나 일부 기억을 잃고,
  - 이후 덧셈 연산으로 새로운 기억(입력 게이트에서 선택한 기억)일부를 추가
  - 그래서 타입 스텝마다 일부 기억이 삭제되고 일부 기억이 추가된다
- 덧셈 연산 후 장기 상태(c_t)가 복사되어 tanh 함수로 전달
  - 이 결과는 출력 게이트에 의해 걸러진 후 단기상태(h_t)와 출력(y_t)을 만든다.



LSTM 셀

- 주층 : $\boldsymbol{g_t}$
  - 현재 입력 x_i와 이전의 상태 (단기상태, h_{t-1})을 분석하는 역할
  - 이 층의 출력이 곧 바로 나가지 않고 장기 상태에서 가장 중요한 부분은 저장하고 나머지는 삭제(기본 셀에서는 y_t, h_t로 출력)
- 게이트 제어기(gate controller) : $\boldsymbol{f_t}$, $\boldsymbol{i_t}$, $\boldsymbol{o_t}$
  - 시그모이드 함수($\boldsymbol{\sigma}$) 사용
  - 출력은 원소 별 곱셈 연산($\otimes$) 수행 : gate라 한다
    - 0을 출력하면 게이트를 닫고 1을 출력하면 게이트를 연다
  - **삭제(forget) 게이트**($\boldsymbol{f_t}$)
    - 장기상태의 어느 부분이 삭제되어야 하는 지 제어
    - 통과할 정보와 억제할 정보 결정
  - **입력(input) 게이트**($\boldsymbol{i_t}$)
    - $\boldsymbol{g_t}$의 어느 부분이 장기 상태에 더해져야 하는지 제어
  - **출력(ouput) 게이트**($\boldsymbol{o_t}$)
    - 장기 상태의 어느 부분을 읽어서 h_t와 y_t로 출력해야 하는지 제어



LSTM 수식

$$g_t = tanh(W_{xg} \cdot x_t + W_{hg} \cdot h_{t-1} + b_g)$$

$$f_t = \sigma(W_{xf}{}^T \cdot x_t + W_{hf}{}^T \cdot h_{t-1} + b_f)$$

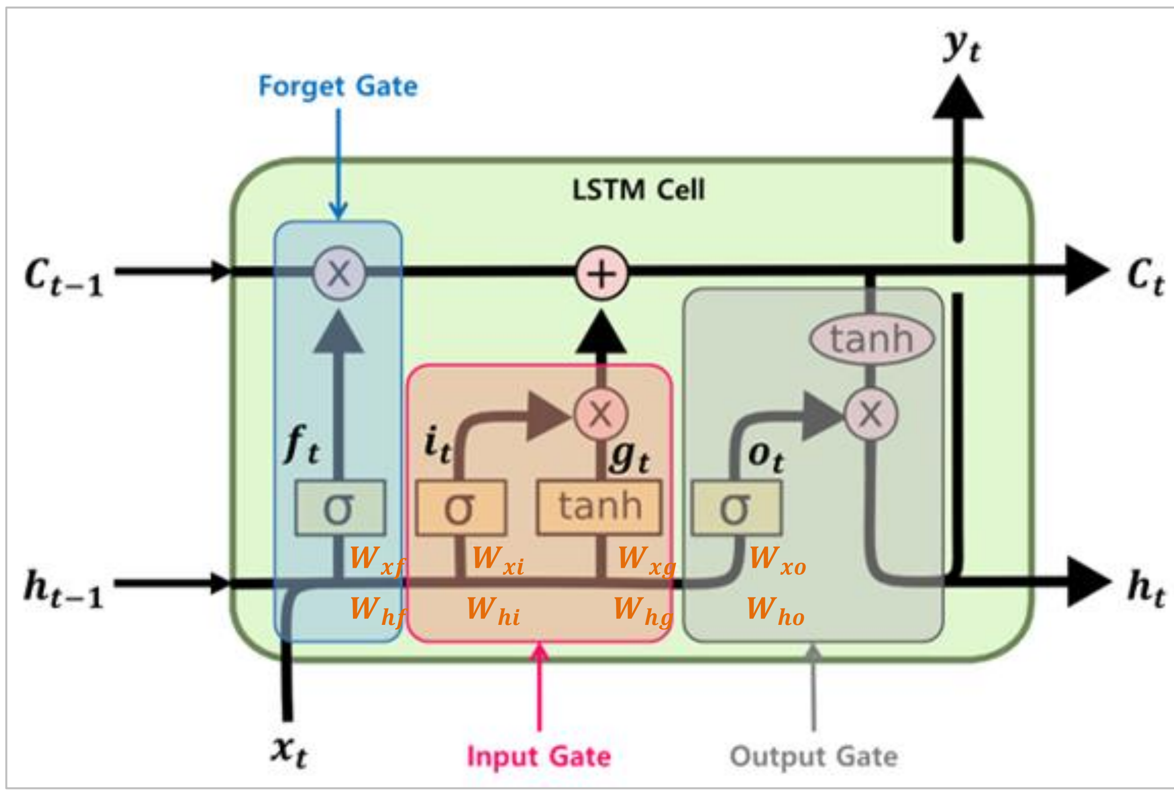
$$i_t = \sigma(W_{xi}{}^T \cdot x_t + W_{hi}{}^T \cdot h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}{}^T \cdot x_t + W_{ho}{}^T \cdot h_{t-1} + b_o)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$

$$y_t = h_t = o_t \otimes tanh(c_t)$$

- $W_{xf}, W_{xi}, W_{xg}, W_{xo}$
  입력벡터 x_t에 연결된 가중치 행렬
- $W_{hf}, W_{hi}, W_{hg}, W_{ho}$
  이전 스텝의 단기 상태(h_{t-1})에 연결된 가중치 행렬



LSTM Backpropagation

$$\delta h_t = \Delta_t + \Delta h_t$$
$$\delta c_t = \delta h_t \otimes o_t \otimes \left(1 - tanh^2(c_t)\right) + \delta c_{t+1} \otimes f_{t+1}$$
$$\delta g_t = \delta c_t \otimes i_t \otimes (1 - {g_t}^2)$$
$$\delta f_t = \delta c_t \otimes c_{t-1} \otimes f_t \otimes (1 - f_t)$$
$$\delta i_t = \delta c_t \otimes g_t \otimes i_t \otimes (1 - i_t)$$
$$\delta o_t = \delta h_t \otimes tanh(c_t) \otimes o_t \otimes (1 - o_t)$$

$$\delta x_t = {W_x}^T \cdot \delta gates_t$$
$$\Delta h_{t-1} = {W_h}^T \cdot \delta gates_t$$

The final updates to the internal parameters is computed as:

$$\delta W_x = \sum_{t=0}^{n} \delta gate_t \otimes x_t$$

$$\delta W_h = \sum_{t=0}^{n} \delta gate_{t+1} \otimes h_t$$

$$\delta b = \sum_{t=0}^{n} \delta gate_{t+1}$$

The Example

$weight:$

$$W_{xg} = \begin{bmatrix} 0.45 \\ 0.25 \end{bmatrix}, \qquad W_{hg} = [0.15], \qquad b_g[0.2]$$

$$W_{xf} = \begin{bmatrix} 0.7 \\ 0.45 \end{bmatrix}, \qquad W_{hf} = [0.1], \qquad b_f[0.15]$$

$$W_{xi} = \begin{bmatrix} 0.95 \\ 0.8 \end{bmatrix}, \qquad W_{hi} = [0.8], \qquad b_i[0.65]$$

$$W_{xo} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \qquad W_{ho} = [0.25], \qquad b_o[0.1]$$

$input\ data:$

$$x_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}\ with\ label: 0.5$$

$$x_1 = \begin{bmatrix} 0.5 \\ 3 \end{bmatrix}\ with\ label: 1.25$$

Ref : https://blog.aidangomez.ca/2016/04/17/Backpropogating-an-LSTM-A-Numerical-Example/?fbclid=IwAR10FnMx8CSQOuY7xOf-QxFAbJQvWmG__lwW6KzlRUWLe1UAq07jeNqTQB0

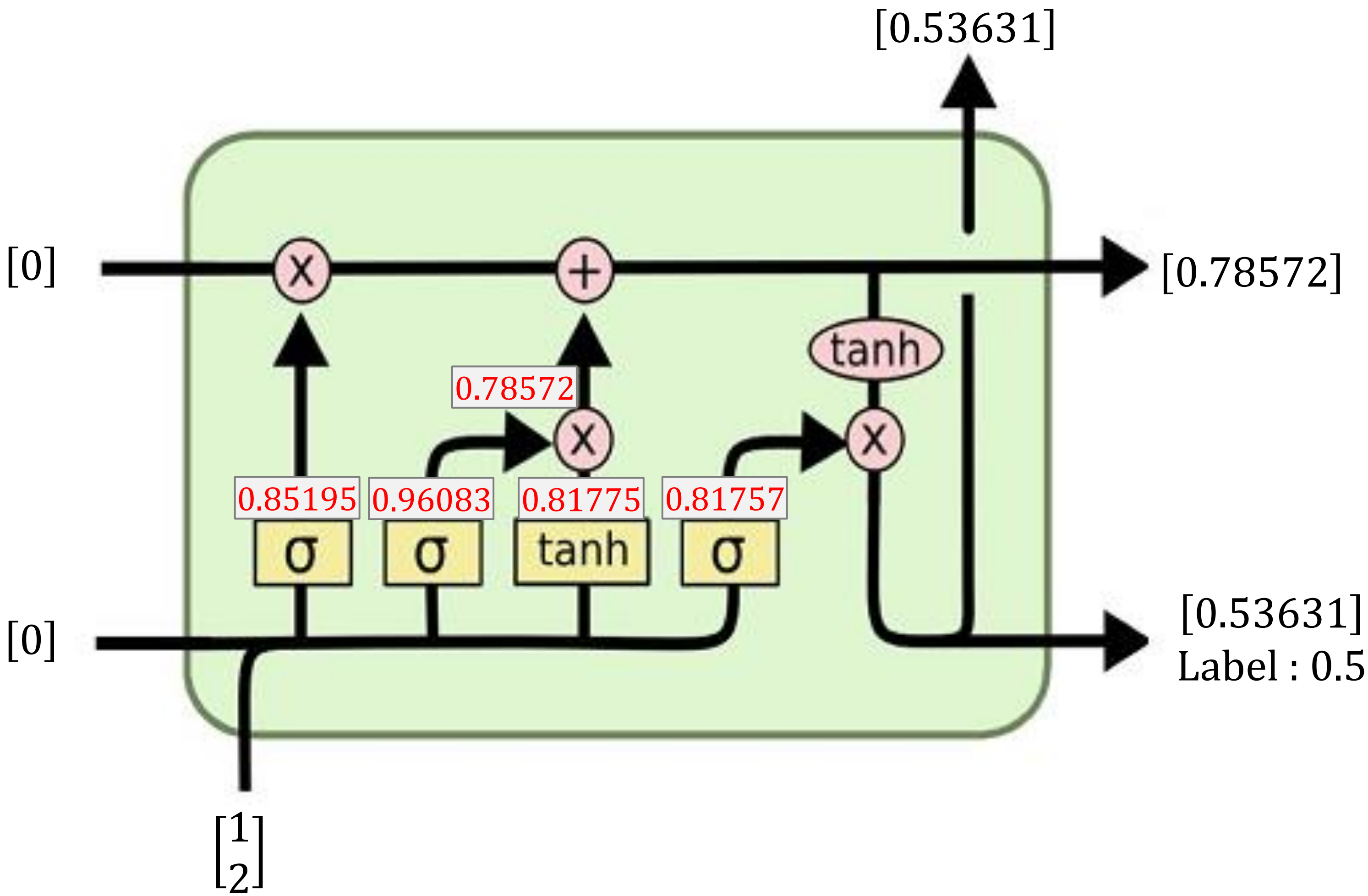
$Forward\ @t\ =\ 0$

$$g_0 = tanh\left(W_{xg} \cdot x_0 + W_{hg} \cdot h_{-1} + b_g\right)$$
$$= tanh\left(\begin{bmatrix}0.45 & 0.25\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix} + [0.15][0] + [0.2]\right) = 0.81775$$
$$f_0 = \sigma\left({W_{xf}}^T \cdot x_0 + {W_{hf}}^T \cdot h_{-1} + b_f\right)$$
$$= \sigma\left(\begin{bmatrix}0.7 & 0.45\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix} + [0.1][0] + [0.15]\right) = 0.85195$$
$$i_0 = \sigma\left({W_{xi}}^T \cdot x_0 + {W_{hi}}^T \cdot h_{-1} + b_f\right)$$
$$= \sigma\left(\begin{bmatrix}0.95 & 0.8\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix} + [0.8][0] + [0.65]\right) = 0.96083$$
$$o_0 = \sigma\left({W_{xo}}^T \cdot x_0 + {W_{ho}}^T \cdot h_{-1} + b_o\right)$$
$$= \sigma\left(\begin{bmatrix}0.6 & 0.4\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix} + [0.25][0] + [0.1]\right) = 0.81757$$

$$c_0 = f_0 \otimes c_{0-1} + i_0 \otimes g_0 = 0.85195 \times 0 + 0.96083 \times 0.81775 = 0.78572$$

$$y_0 = h_0 = o_0 \otimes tanh(c_0) = 0.81757 \times tanh(0.78572) = 0.53631$$

Forward @t = 0


[0.53631]
[0]
[0.78572]
0.78572
tanh
0.85195
0.96083
0.81775
0.81757
σ
σ
tanh
σ
[0]
[0.53631]
Label : 0.5
[1
2]

$Forward\ @t\ = 1$

$$g_1 = tanh\left(W_{xg} \cdot x_1 + W_{hg} \cdot h_0 + b_g\right)$$

$$= tanh\left(\begin{bmatrix}0.45 & 0.25\end{bmatrix}\begin{bmatrix}0.5\\3\end{bmatrix} + [0.15][0.53631] + [0.2]\right) = 0.84980$$

$$f_1 = \sigma\left({W_{xf}}^T \cdot x_1 + {W_{hf}}^T \cdot h_0 + b_f\right)$$

$$= \sigma\left(\begin{bmatrix}0.7 & 0.45\end{bmatrix}\begin{bmatrix}0.5\\3\end{bmatrix} + [0.1][0.53631] + [0.15]\right) = 0.87030$$

$$i_1 = \sigma\left({W_{xi}}^T \cdot x_1 + {W_{hi}}^T \cdot h_0 + b_f\right)$$

$$= \sigma\left(\begin{bmatrix}0.95 & 0.8\end{bmatrix}\begin{bmatrix}0.5\\3\end{bmatrix} + [0.8][0.53631] + [0.65]\right) = 0.98118$$

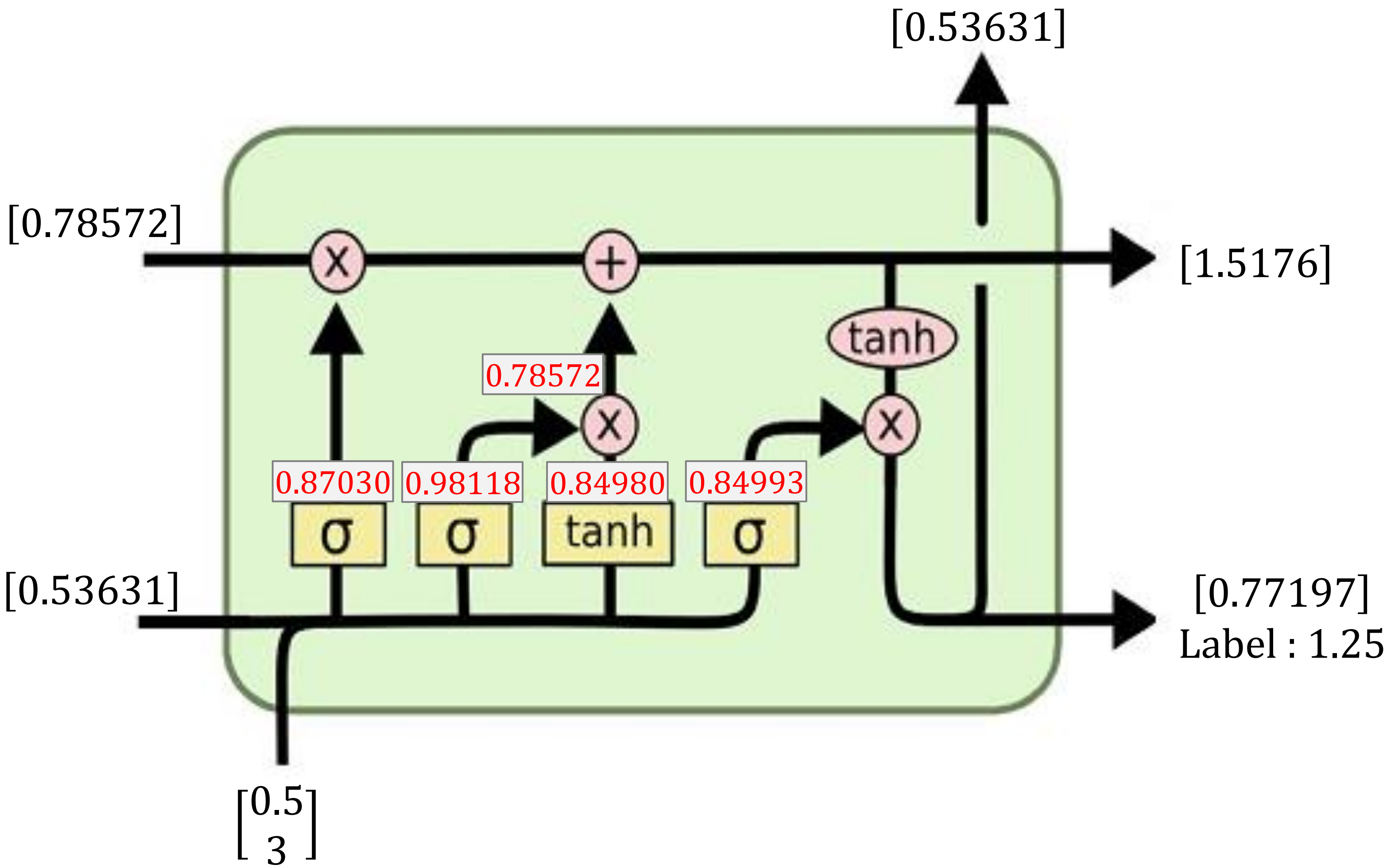
$$o_1 = \sigma\left({W_{xo}}^T \cdot x_1 + {W_{ho}}^T \cdot h_0 + b_o\right)$$

$$= \sigma\left(\begin{bmatrix}0.6 & 0.4\end{bmatrix}\begin{bmatrix}0.5\\3\end{bmatrix} + [0.25][0.53631] + [0.1]\right) = 0.84993$$

$$c_1 = f_1 \otimes c_0 + i_1 \otimes g_1 = 0.87030 \times 0.78572 + 0.98118 \times 0.84980 = 1.5176$$

$$y_1 = h_1 = o_1 \otimes tanh(c_1) = 0.84993 \times tanh(1.5176) = 0.77197$$

Forward @t $= 1$



[0.53631]
[0.78572]
[1.5176]
tanh
0.78572
0.87030
0.98118
0.84980
0.84993
σ
σ
tanh
σ
[0.53631]
[0.77197]
Label : 1.25
$\begin{bmatrix} 0.5 \\ 3 \end{bmatrix}$

Backward @t=1

$$E(\hat{y}, y) = \frac{(\hat{y} - y)^2}{2}$$

$$\delta_x E(\hat{y}, y) = \hat{y} - y$$

$$\Delta_1 = \delta_y E = 0.77197 - 1.25 = -0.47803$$

$$\Delta h_1 = 0 \ (no\ next\ time\ steps)$$

$$\delta h_1 = \Delta_1 + \Delta h_1 = -0.47803 + 0 = -0.47803$$

$$\delta c_1 = \delta h_1 \otimes o_1 \otimes \left(1 - tanh^2(c_1)\right) + \delta c_2 \otimes f_2$$

$$= -0.47803 \times 0.84993 \times \left(1 - tanh^2(1.5176)\right) + 0 \times 0 = -0.07111$$

$$\delta g_1 = \delta c_1 \otimes i_1 \otimes (1 - {g_1}^2) = -0.07111 \times 0.98118 \times (1 - 0.84980^2) = -0.01938$$

$$\delta f_1 = \delta c_1 \otimes c_0 \otimes f_1 \otimes (1 - f_1) = -0.07111 \times 0.0.78572 \times 0.87030 \times (1 - 0.87030) = -0.00631$$

$$\delta i_1 = \delta c_1 \otimes g_1 \otimes i_1 \otimes (1 - i_1) = -0.07111 \times 0.84980 \times 0.98118 \times (1 - 0.98118) = -0.00112$$

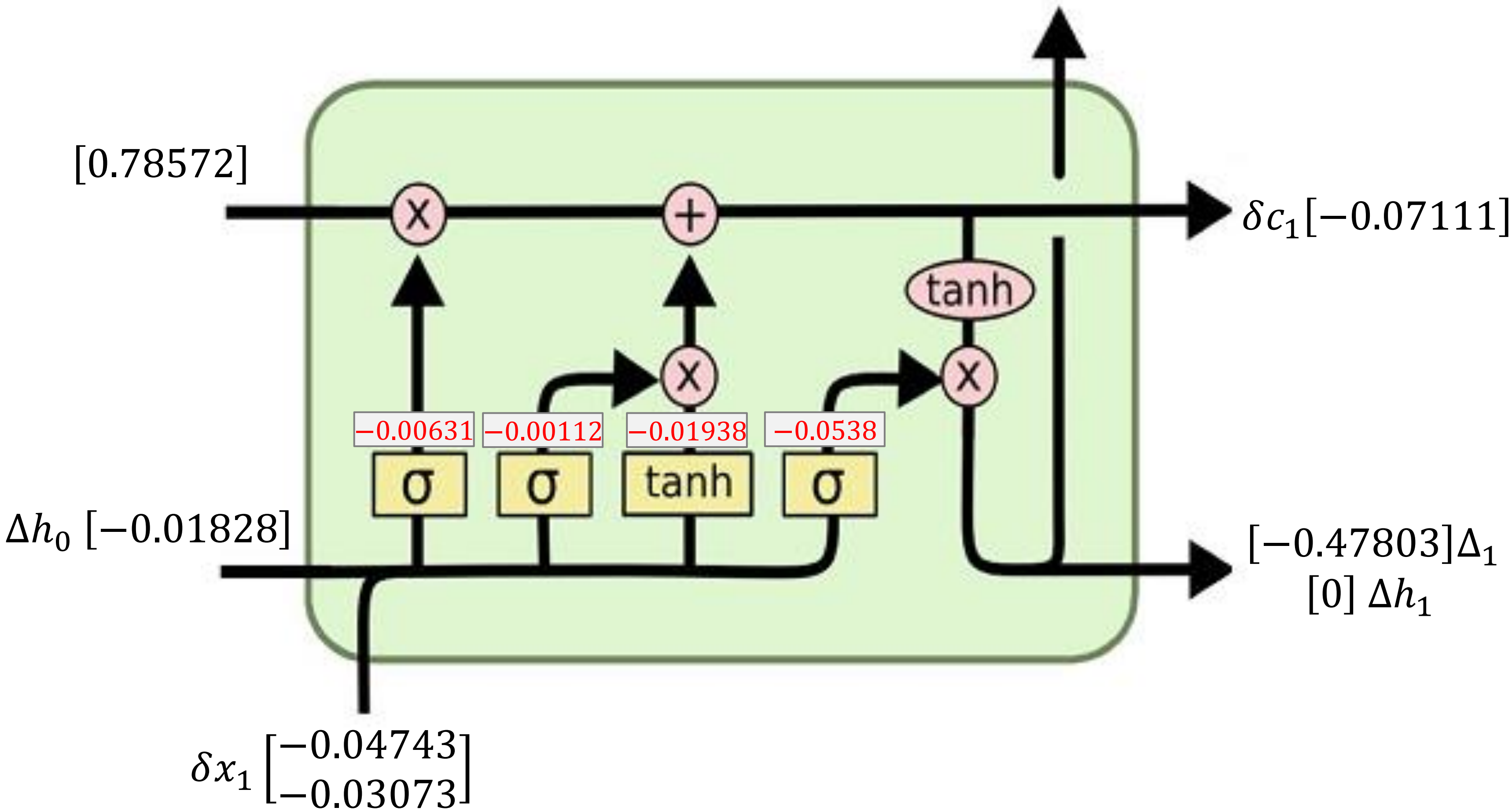
$$\delta o_1 = \delta h_1 \otimes tanh(c_1) \otimes o_1 \otimes (1 - o_1)$$

$$= -0.47803 \times tanh(1.5176) \times 0.84993 \times (1 - 0.84993) = -0.05538$$

$$\delta x_1 = {W_x}^T \cdot \delta gates_1 = \begin{bmatrix} 0.45 & 0.70 & 0.95 & 0.60 \\ 0.25 & 0.45 & 0.80 & 0.40 \end{bmatrix} \begin{bmatrix} -0.01938 \\ -0.00631 \\ -0.00112 \\ -0.05538 \end{bmatrix} = \begin{bmatrix} -0.04743 \\ -0.03073 \end{bmatrix}$$

$$\Delta h_0 = {W_h}^T \cdot \delta gates_1 = \begin{bmatrix} 0.15 & 0.10 & 0.80 & 0.25 \end{bmatrix} \begin{bmatrix} -0.01938 \\ -0.00631 \\ -0.00112 \\ -0.05538 \end{bmatrix} = -0.01828$$

$Backward\ @t\ = 1$



[0.78572]
$\delta c_1[-0.07111]$
-0.00631
-0.00112
-0.01938
-0.0538
σ
σ
tanh
σ
tanh
$\Delta h_0\ [-0.01828]$
$[-0.47803]\Delta_1$
$[0]\ \Delta h_1$
$\delta x_1 \begin{bmatrix} -0.04743 \\ -0.03073 \end{bmatrix}$

Backward @t=0

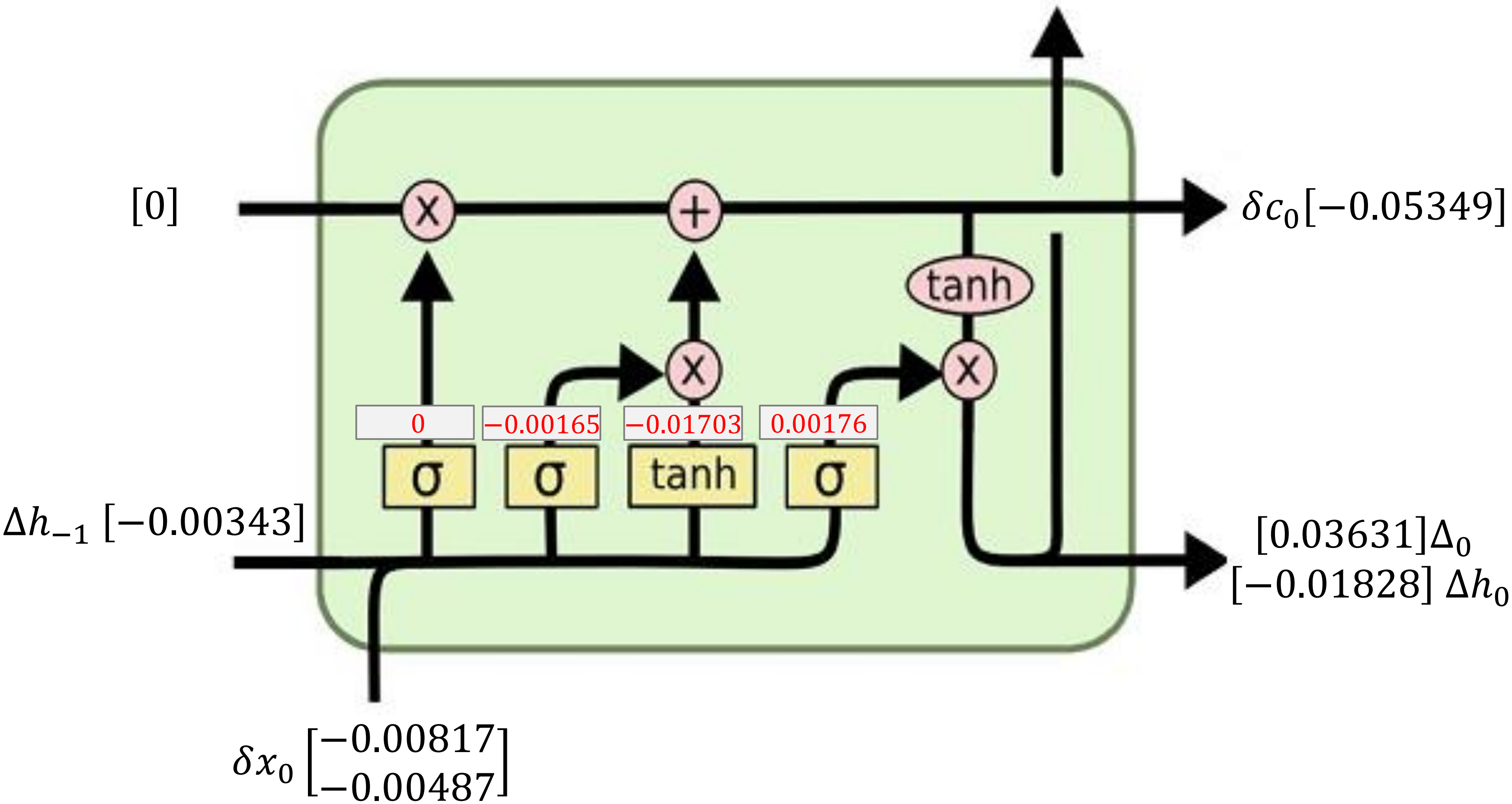
$$E(\hat{y}, y) = \frac{(\hat{y} - y)^2}{2}$$
$$\delta_x E(\hat{y}, y) = \hat{y} - y$$
$$\Delta_0 = \delta_x E = 0.53631 - 0.5 = 0.03631$$
$$\Delta h_0 = -0.01828 \ (passed\ back\ from\ t = 1)$$

$$\delta h_0 = \Delta_0 + \Delta h_0 = 0.03631 + -0.01828 = 0.01803$$
$$\delta c_0 = \delta h_0 \otimes o_0 \otimes \left(1 - tanh^2(c_0)\right) + \delta c_1 \otimes f_1$$
$$= 0.01803 \times 0.81757 \times \left(1 - tanh^2(0.78572)\right) \pm 0.07111 \times 0.87030 = -0.05349$$
$$\delta g_0 = \delta c_0 \otimes i_0 \otimes (1 - {g_0}^2) = -0.05349 \times 0.996083 \times (1 - 0.81775^2) = -0.01703$$
$$\delta f_0 = \delta c_0 \otimes c_{-1} \otimes f_0 \otimes (1 - f_0) = -0.05349 \times 0 \times 0.85195 \times (1 - 0.85195) = 0$$
$$\delta i_0 = \delta c_0 \otimes g_0 \otimes i_0 \otimes (1 - i_0) = -0.05349 \times 0.81775 \times 0.96083 \times (1 - 0.96083) = -0.00165$$
$$\delta o_0 = \delta h_0 \otimes tanh(c_0) \otimes o_0 \otimes (1 - o_0)$$
$$= -0.01803 \times tanh(0.78572) \times 0.81757 \times (1 - 0.81757) = -0.00176$$

$$\delta x_0 = {W_x}^T \cdot \delta gates_0 = \begin{bmatrix} 0.45 & 0.70 & 0.95 & 0.60 \\ 0.25 & 0.45 & 0.80 & 0.40 \end{bmatrix} \begin{bmatrix} -0.01703 \\ 0 \\ -0.00165 \\ -0.00176 \end{bmatrix} = \begin{bmatrix} -0.00817 \\ -0.00487 \end{bmatrix}$$

$$\Delta h_{-1} = {W_h}^T \cdot \delta gates_0 = \begin{bmatrix} 0.15 & 0.10 & 0.80 & 0.25 \end{bmatrix} \begin{bmatrix} -0.01703 \\ 0 \\ -0.00165 \\ -0.00176 \end{bmatrix} = -0.00343$$

$Backward\ @t\ = 0$



[0]
$\delta c_0[-0.05349]$
0
−0.00165
−0.01703
0.00176
σ
σ
tanh
σ
tanh
Δh_{-1} [−0.00343]
$[0.03631]\Delta_0$
$[-0.01828]\ \Delta h_0$
$\delta x_0 \begin{bmatrix}-0.00817\\-0.00487\end{bmatrix}$

LSTM Backpropagation

$$\delta W_x = \sum_{t=0}^{n} \delta gate_t \otimes x_t = \begin{bmatrix} -0.01703 \\ 0 \\ -0.00165 \\ -0.00176 \end{bmatrix} [1.0 \quad 2.0] + \begin{bmatrix} -0.01938 \\ -0.00631 \\ -0.00112 \\ -0.05538 \end{bmatrix} [0.5 \quad 3.0] = \begin{bmatrix} -0.02672 & -0.09220 \\ -0.00316 & -0.01893 \\ -0.00221 & -0.00666 \\ -0.02593 & -0.16262 \end{bmatrix}$$

$$\delta W_h = \sum_{t=0}^{n} \delta gate_{t+1} \otimes h_t = \begin{bmatrix} -0.01938 \\ -0.00631 \\ -0.00112 \\ -0.05538 \end{bmatrix} [0.53631] = \begin{bmatrix} -0.01039 \\ -0.00338 \\ -0.00060 \\ -0.02970 \end{bmatrix}$$

$$\delta b = \sum_{t=0}^{n} \delta gate_{t+1} = \begin{bmatrix} -0.01703 \\ 0 \\ -0.00165 \\ -0.00176 \end{bmatrix} + \begin{bmatrix} -0.01938 \\ -0.00631 \\ -0.00112 \\ -0.05538 \end{bmatrix} = \begin{bmatrix} -0.03641 \\ -0.00631 \\ -0.00277 \\ -0.05714 \end{bmatrix}$$

Updating out parameters based on the SGD update function$(\lambda = 0.1)$: $W^{new} = W^{old} - \lambda \delta W^{old}$

$$W_{xg} = \begin{bmatrix} 0.45267 \\ 0.25922 \end{bmatrix}, \qquad W_{hg} = [0.15104], \qquad b_g = [0.20364]$$

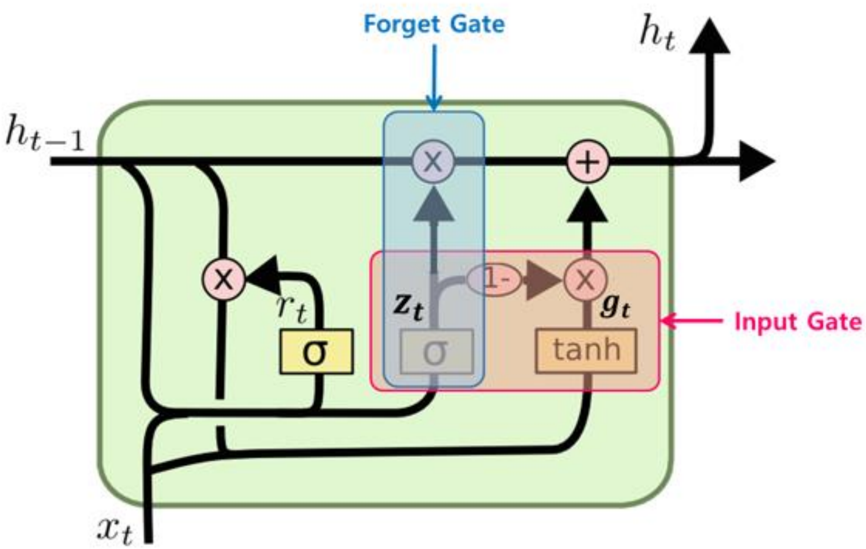
$$W_{xf} = \begin{bmatrix} 0.70032 \\ 0.45189 \end{bmatrix}, \qquad W_{hf} = [0.10034], \qquad b_f = [0.15063]$$

$$W_{xi} = \begin{bmatrix} 0.95022 \\ 0.80067 \end{bmatrix}, \qquad W_{hi} = [0.80006], \qquad b_i = [0.65027]$$

$$W_{xo} = \begin{bmatrix} 0.60259 \\ 0.41626 \end{bmatrix}, \qquad W_{ho} = [0.25297], \qquad b_o = [0.10536]$$

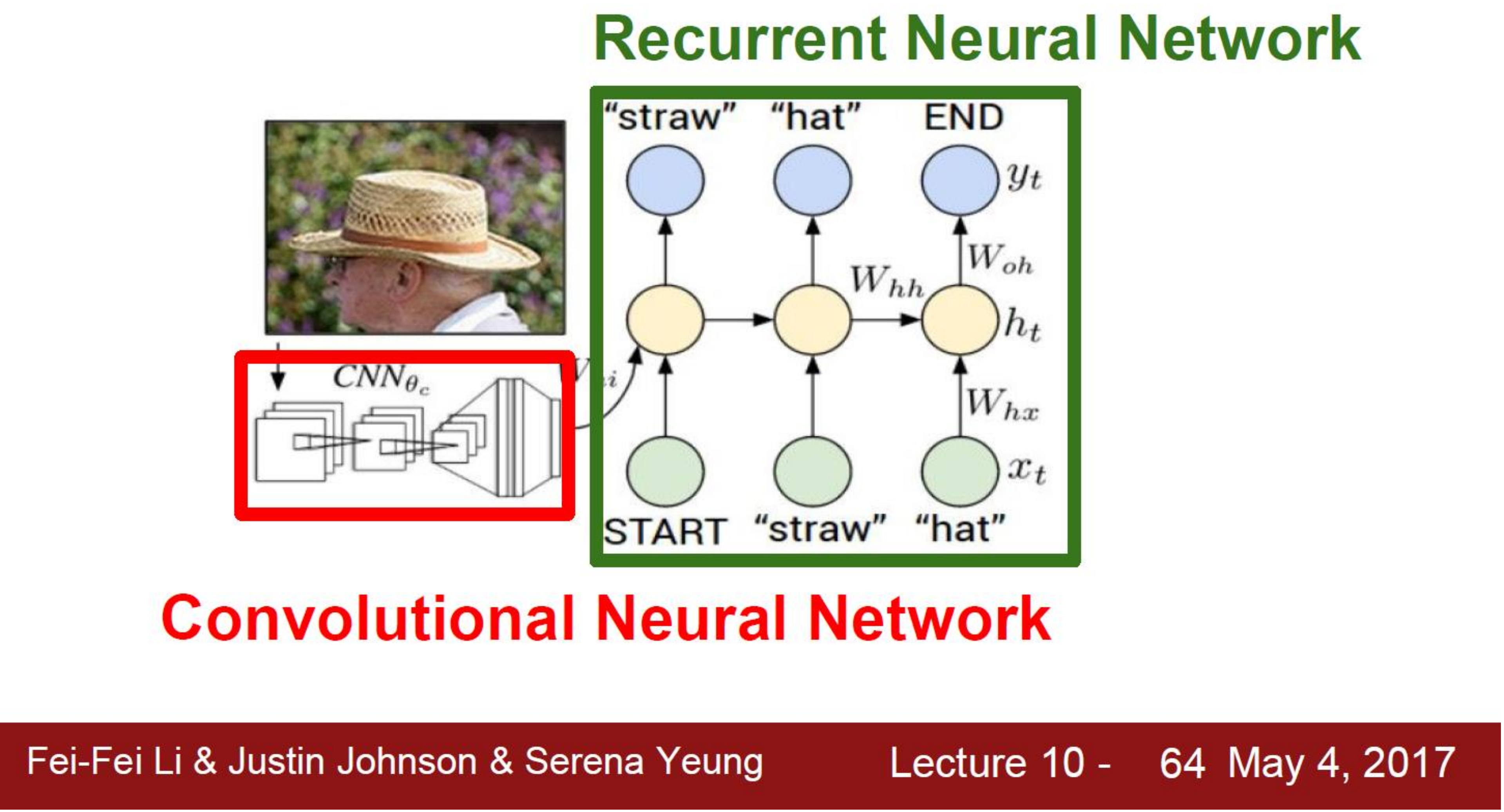
Gated Recurrent Units (GRUs)

- 2014년에 K. Cho(조경현) 등에 의해 제안된 LSTM의 간소화된 버전
  - Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation



$$\mathbf{r}_t = \sigma\left(\mathbf{W}_{xr}^T \cdot \mathbf{x}_t + \mathbf{W}_{hr}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_r\right)$$
$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{xz}^T \cdot \mathbf{x}_t + \mathbf{W}_{hz}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right)$$
$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{xg}^T \cdot \mathbf{x}_t + \mathbf{W}_{hg}^T \cdot (\mathbf{r}_t \otimes \mathbf{h}_{t-1}) + \mathbf{b}_g\right)$$
$$\mathbf{h}_t = \mathbf{z}_t \otimes \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \otimes \mathbf{g}_t$$

- $LSTM$의 c_t와 h_t가 하나의 벡터 h_t로 합쳐졌다.
- 하나의 gate controller인 z_t가 forget과 input 게이트를 제어한다.
  - z_t가 1을 출력하면 forget 게이트가 열리고 input 게이트가 닫힌다.
  - z_t가 0을 출력하면 forget 게이트가 닫히고 input 게이트가 열린다.
  - 즉, 이전$(t-1)$의 기억이 저장될 때 마다 타임스텝 t의 입력은 삭제된다.
- GRU는 output 게이트가 없어 전체 상태 벡터 h_t가 타임스텝마다 출력되며,
- 이전 상태(h_{t-1})의 어느 부분이 출력될지 제어하는 새로운 gate controller인 r_t가 있다.


Recurrent Neural Network
"straw" "hat" END
y_t
W_{oh}
W_{hh}
h_t
CNN_{θ_c}
W_{hx}
x_t
START "straw" "hat"
Convolutional Neural Network
Fei-Fei Li & Justin Johnson & Serena Yeung
Lecture 10 - 64 May 4, 2017

image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

conv-512
conv-512
maxpool

FC-4096
FC-4096
FC-1000
softmax

test image

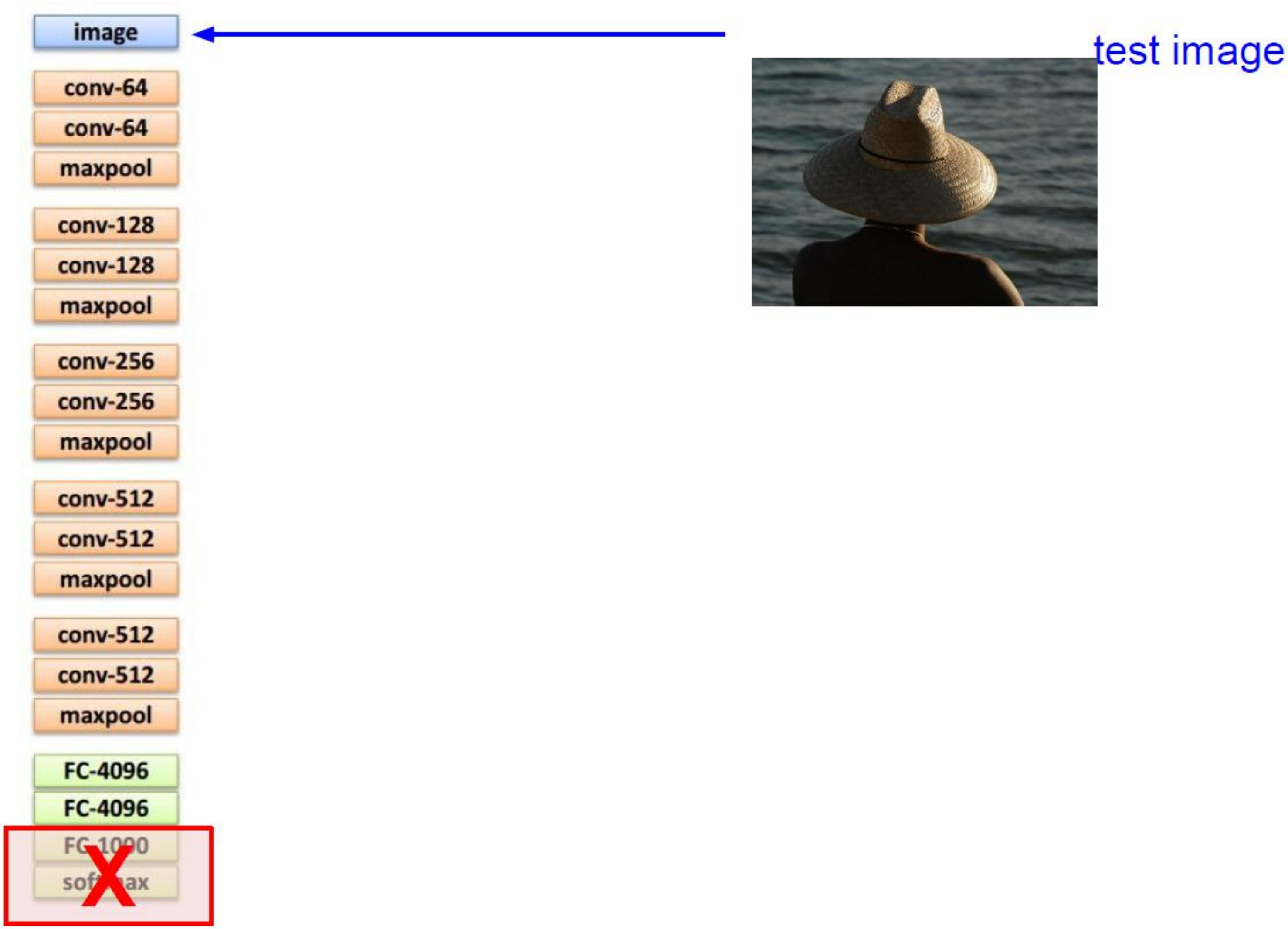


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
X
test image

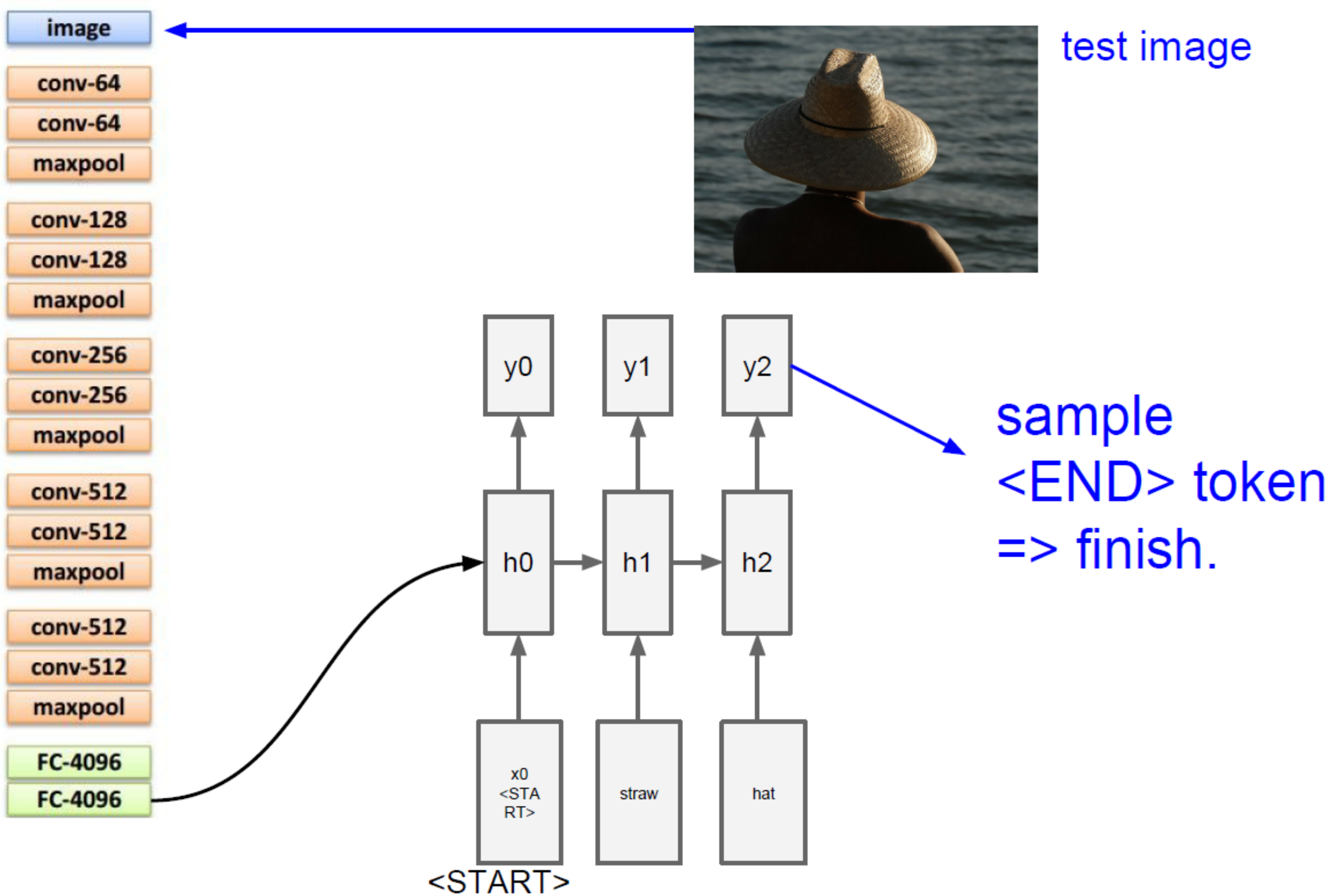


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
sample
<END> token
=> finish.
h0
h1
h2
x0
<STA
RT>
straw
hat
<START>