Machine Learning

Recurrent Neural Network

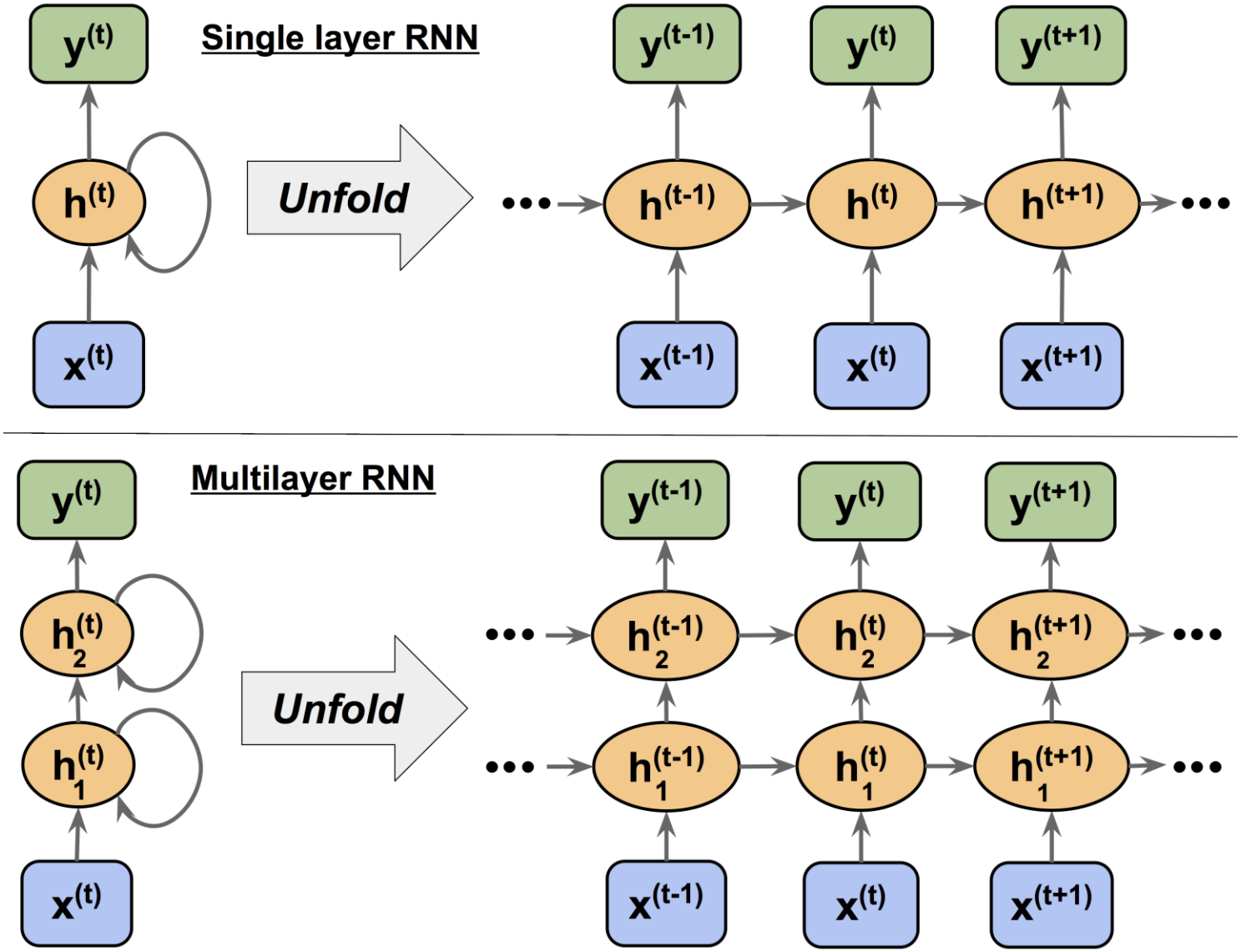
김선녕(ksycafe@gmail.com)


RNN
LSTM

개요

- 순차적인 데이터(sequential data)를 학습하여 분류 또는 예측을 수행
- 각 레이어마다 파라미터(Parameter)의 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

Recurrent Neural Network

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

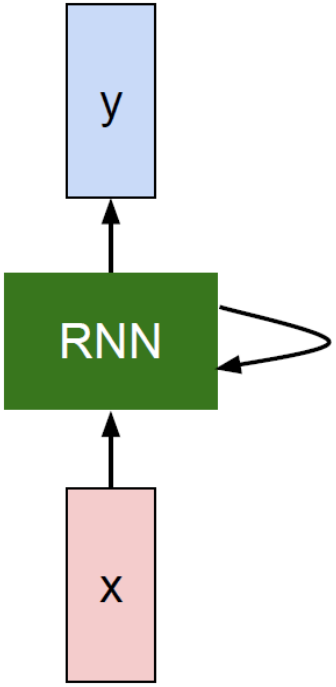
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

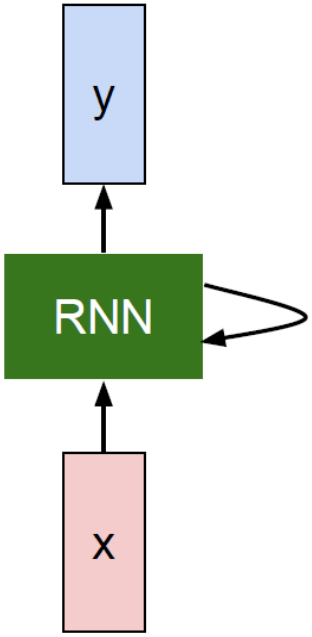
input vector at some time step

some function with parameters W



(Simple) Recurrent Neural Network

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



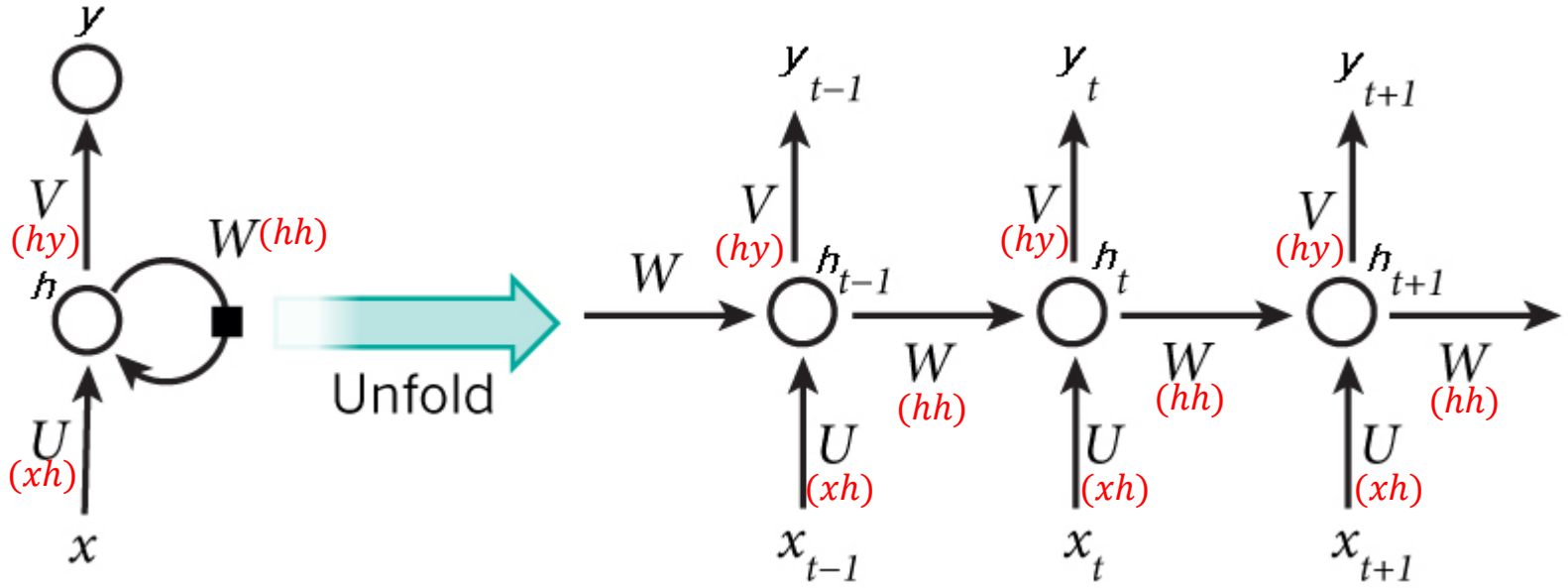
$$h_t = f_W(h_{t-1}, x_t)$$

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

$$y_t = W_{hy}h_t$$

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

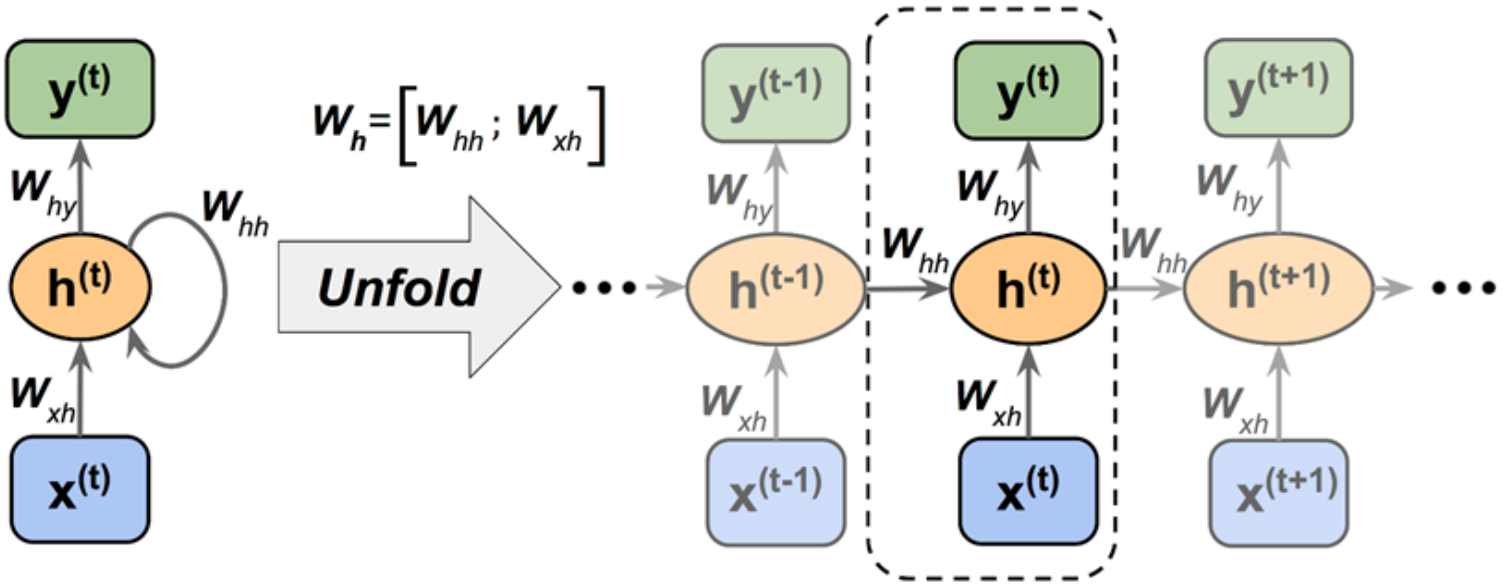
RNN의 기본



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



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

$$\downarrow$$

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

$$y_t = W_{hy}h_t$$

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

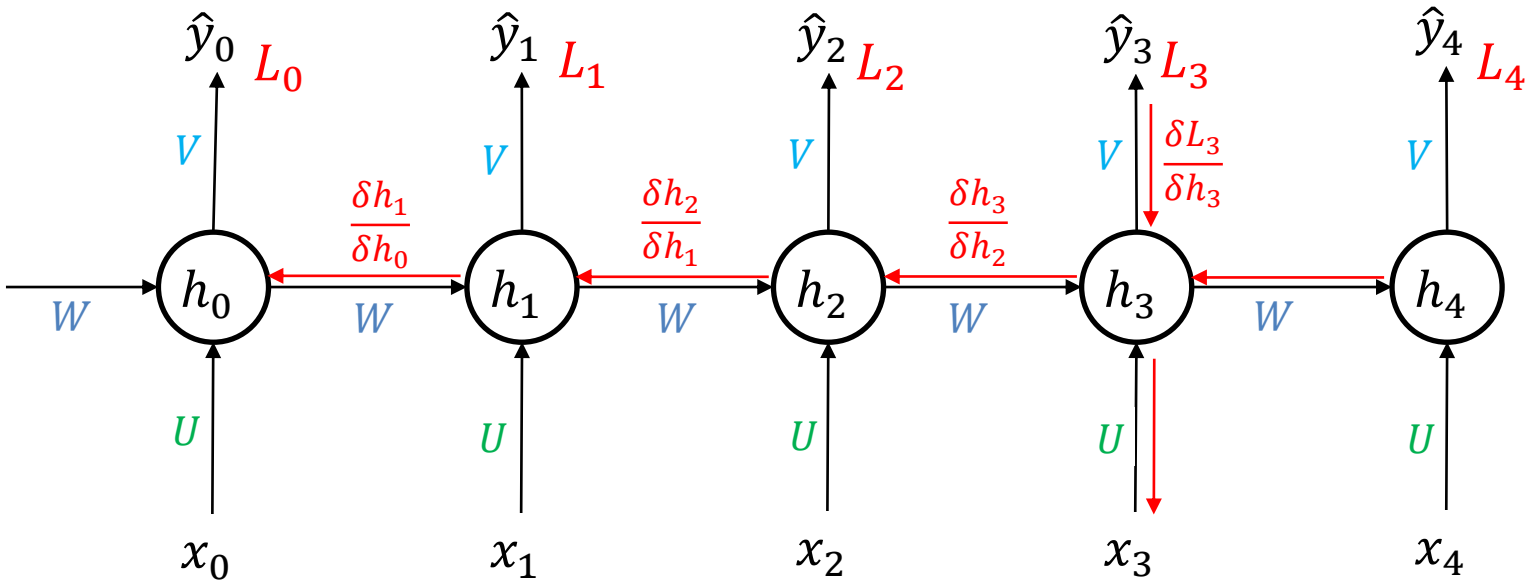
$$h^t = \phi_h(fw_h{}^t) = \phi_h(W_{xh}x^t + W_{hh}h^{t-1} + b_h)$$

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

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

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

BPTT(BackPropagation Through Time)



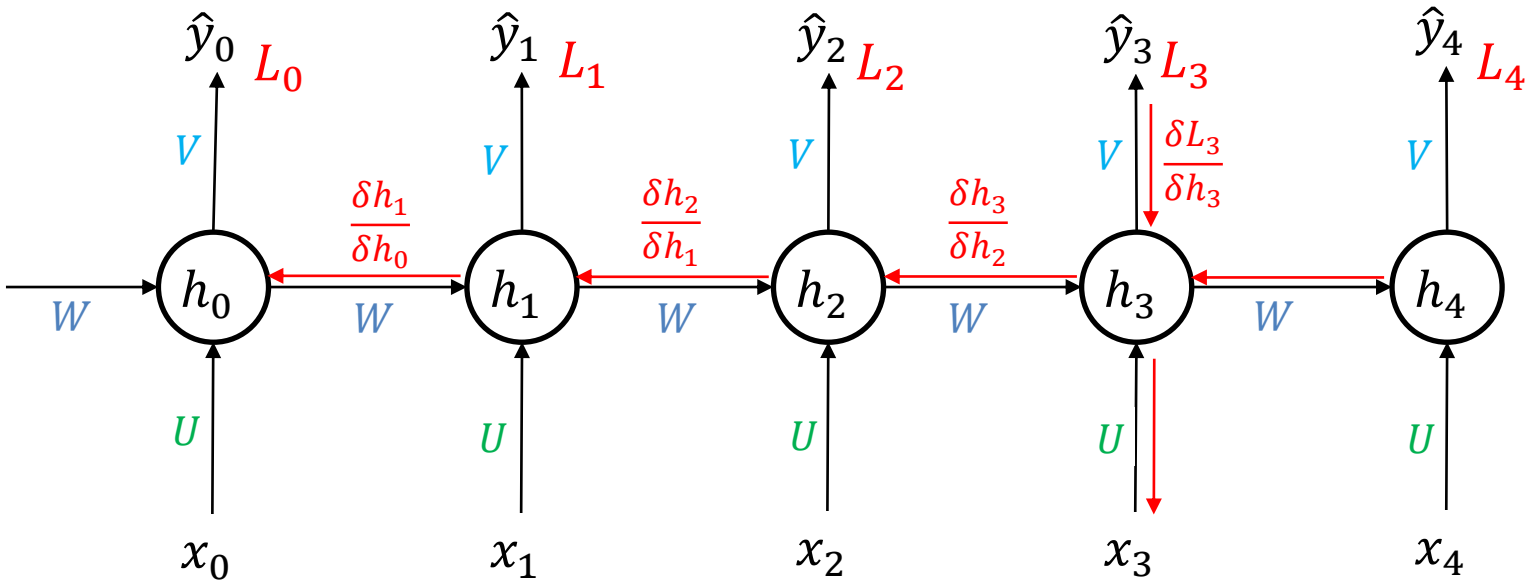
- RNN 모델을 학습하는데 사용되는 핵심 알고리즘
- 전체 손실(E) : $t = 1$에서 $t = T$까지 타임 스텝의 모든 손실함수 합

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

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

$z_3 = Vh_3,$ $\otimes$: 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_2는 h_1에 의존

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

Vanishing Gradient

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

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

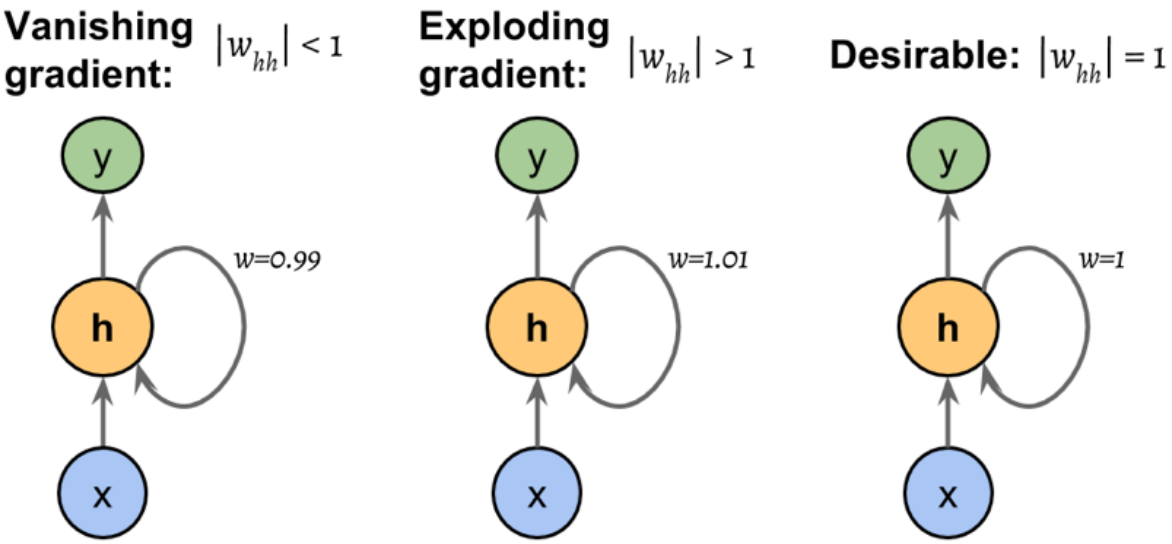
$\frac{\delta h^t}{\delta h^k}$(이전타임스텝의곱)로 인하여$vanishing\ gradient$발생

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

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

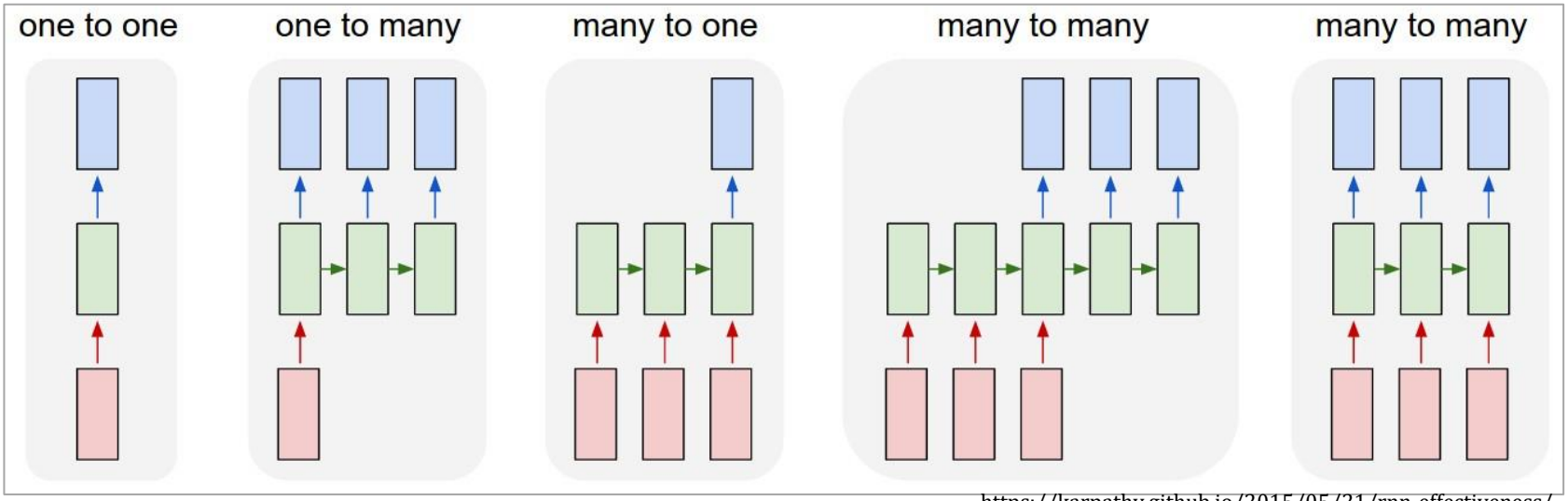
$|w| > 1$이면 $t-k$ 가 클 때 w^{t-k} 값이 매우 커진다

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



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



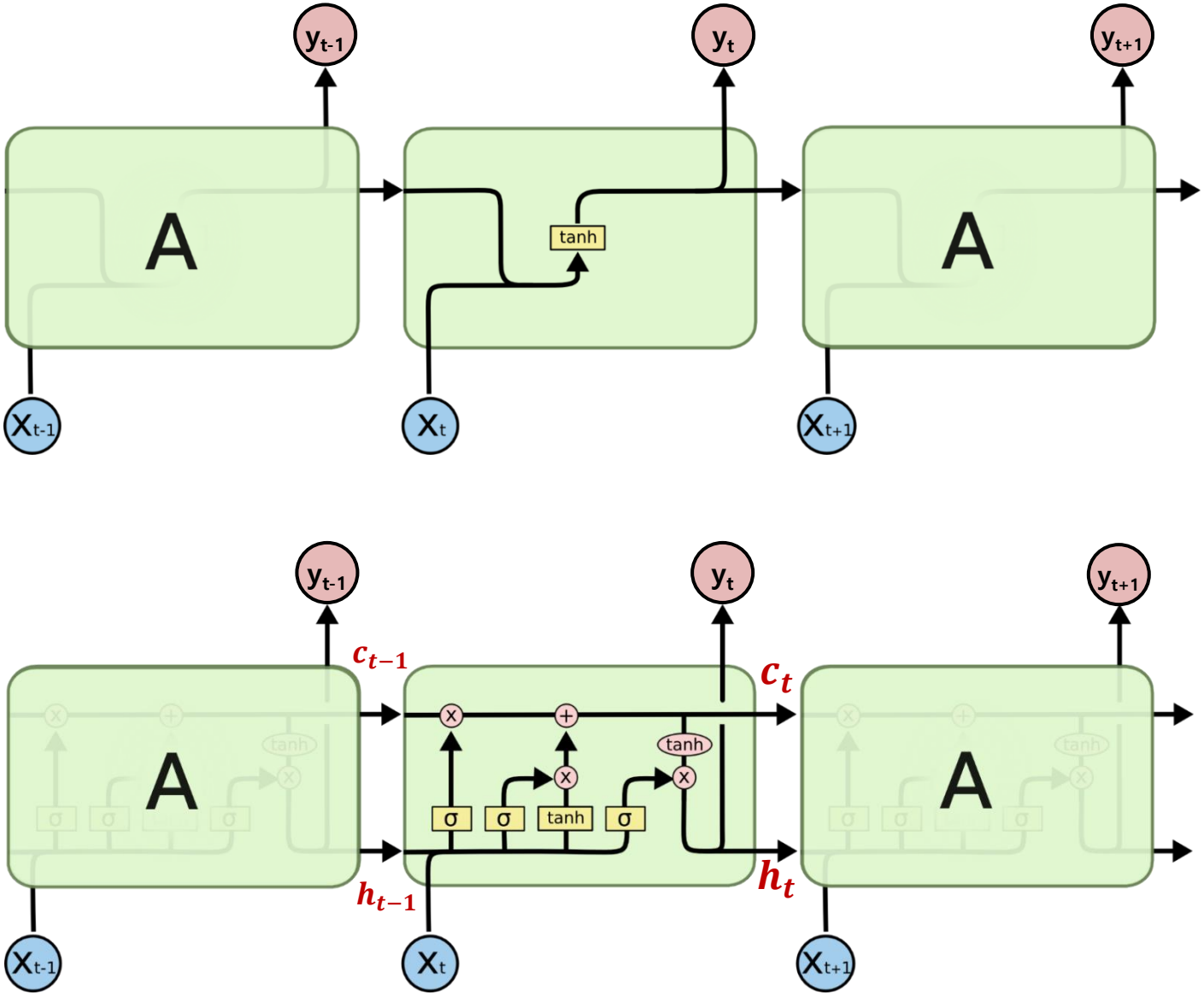
https://karpathy.github.io/2015/05/21/rnn-effectiveness/


RNN
LSTM

LSTM

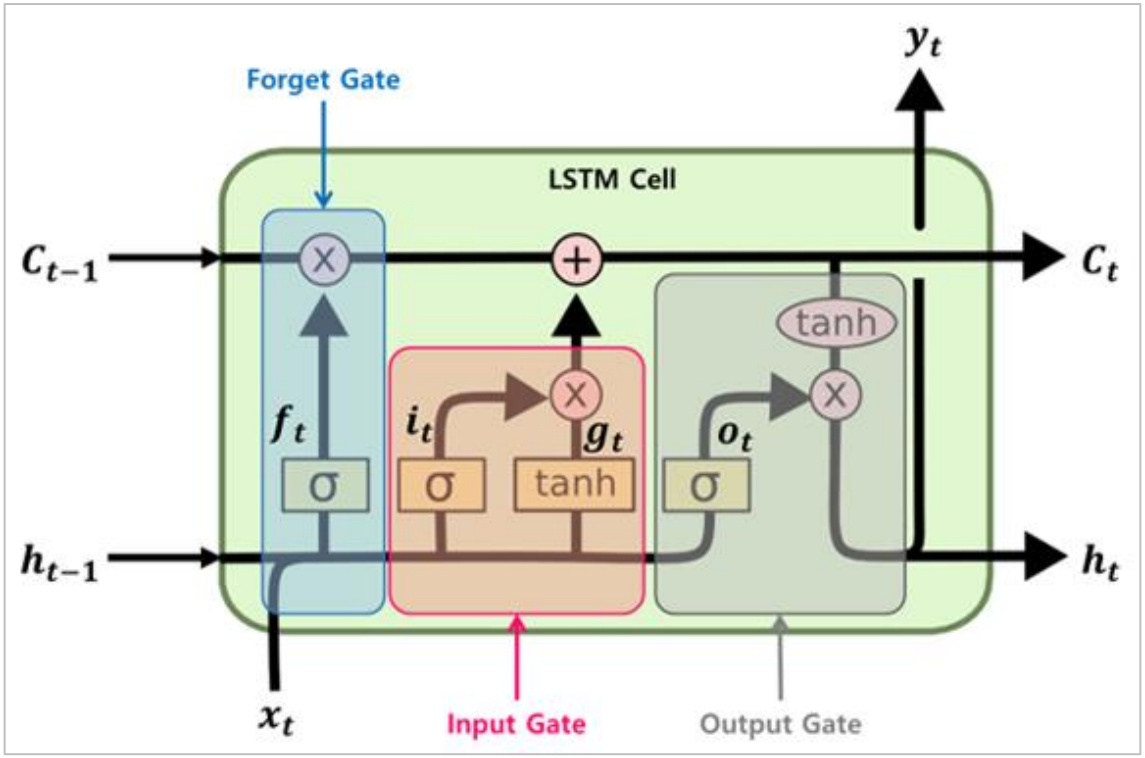
- RNN의 Gradient Problem 해결책
  - $T-BPTT(Truncated\ BackPropagation\ Through\ time)$
  - $\boldsymbol{LSTM(Long\ Short-Term\ Memory)}$
- LSTM의 중요요소 : 메모리 셀($\boldsymbol{memory\ cell}$) - 은닉층
  - 각 메모리 셀에 적절한 가중치 $w=1$를 유지하는 게이트
    - $\boldsymbol{cell\ state}$: 게이트의 출력
  - 저장할 것, 버릴 것, 읽어 들일 것을 학습하는 것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

RNN vs LSTM


y_{t-1}
y_t
y_{t+1}
A
tanh
A
X_{t-1}
X_t
X_{t+1}
y_{t-1}
y_t
y_{t+1}
c_{t-1}
c_t
A
tanh
σ
σ
tanh
σ
tanh
A
h_{t-1}
h_t
X_{t-1}
X_t
X_{t+1}

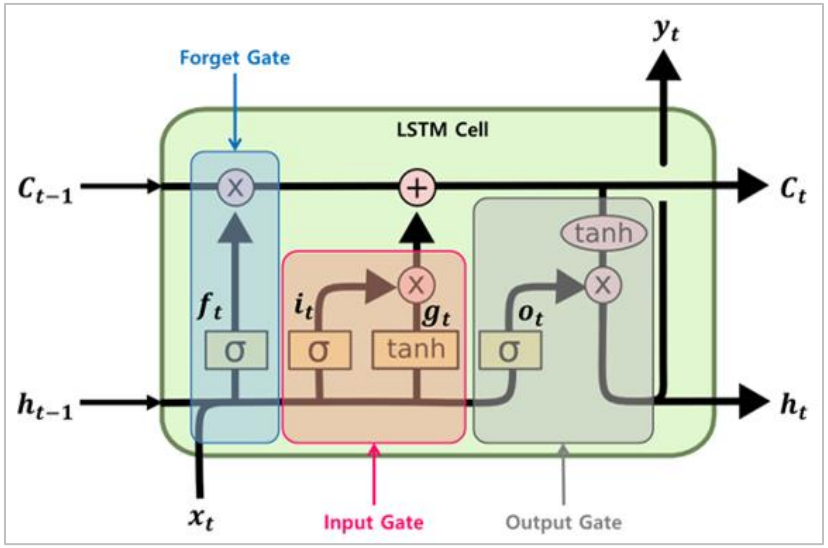
cell state

- 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 : 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) : f_t, i_t, o_t
  - 시그모이드 함수(σ) 사용
  - 출력은 원소 별 곱셈 연산($\otimes$) 수행 : gate라 한다
    - 0을 출력하면 게이트를 닫고 1을 출력하면 게이트를 연다
  - **삭제(forget) 게이트(f_t)**
    - 장기상태의 어느 부분이 삭제되어야 하는 지 제어
    - 통과할 정보와 억제할 정보 결정
  - **입력(input) 게이트(i_t)**
    - g_t의 어느 부분이 장기 상태에 더해져야 하는지 제어
  - **출력(ouput) 게이트(o_t)**
    - 장기 상태의 어느 부분을 읽어서 h_t와 y_t로 출력해야 하는지 제어



LSTM 수식

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

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

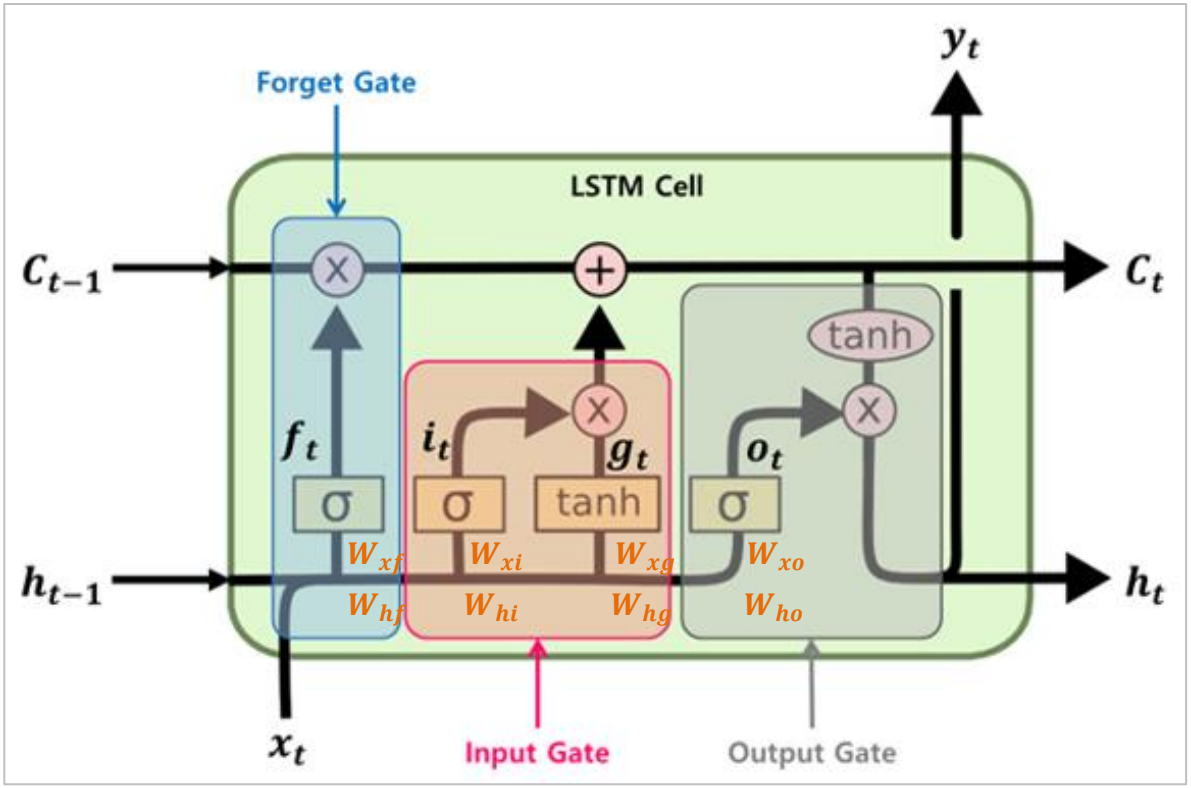
$$i_t = \sigma\left(W_{xi}{}^T \cdot x_t + W_{hi}{}^T \cdot h_{t-1} + b_f\right)$$

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

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

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

- $W_{xf}, W_{xi}, W_{xg}, W_{xo}$
  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



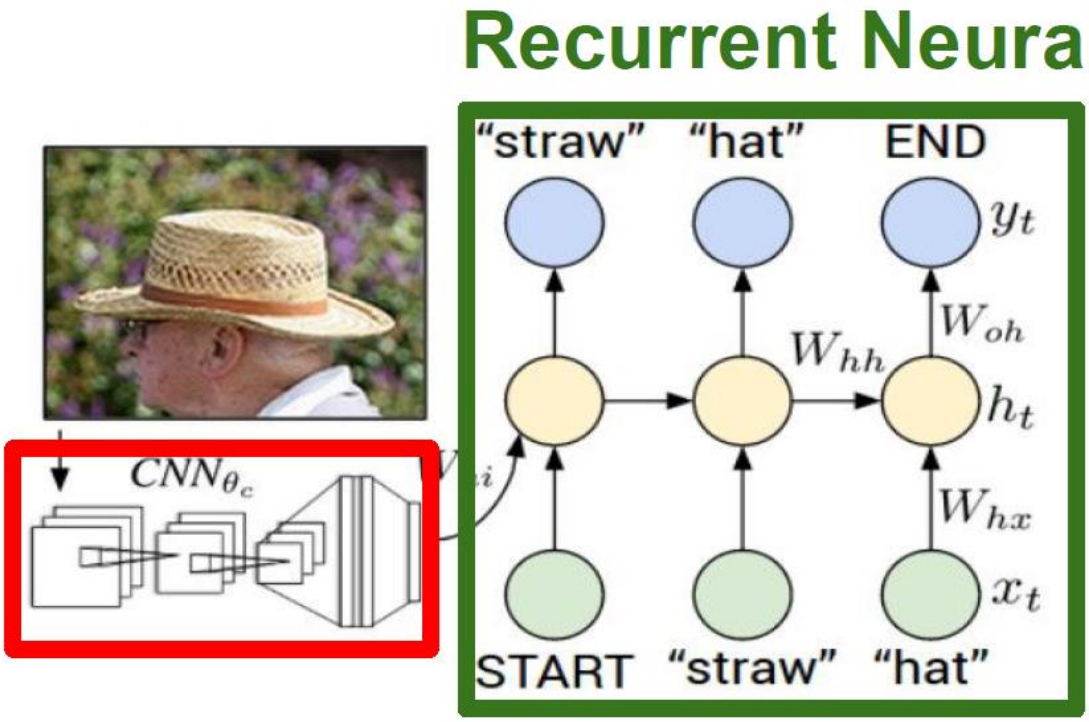
LSTM Backpropagation

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

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

The final updates to the internal parameters is computed as:

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


Recurrent Neural Network
"straw"
"hat"
END
y_t
W_{oh}
W_{hh}
h_t
W_{hx}
x_t
START
"straw"
"hat"
CNN_{θ_c}
Convolutional Neural Network

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

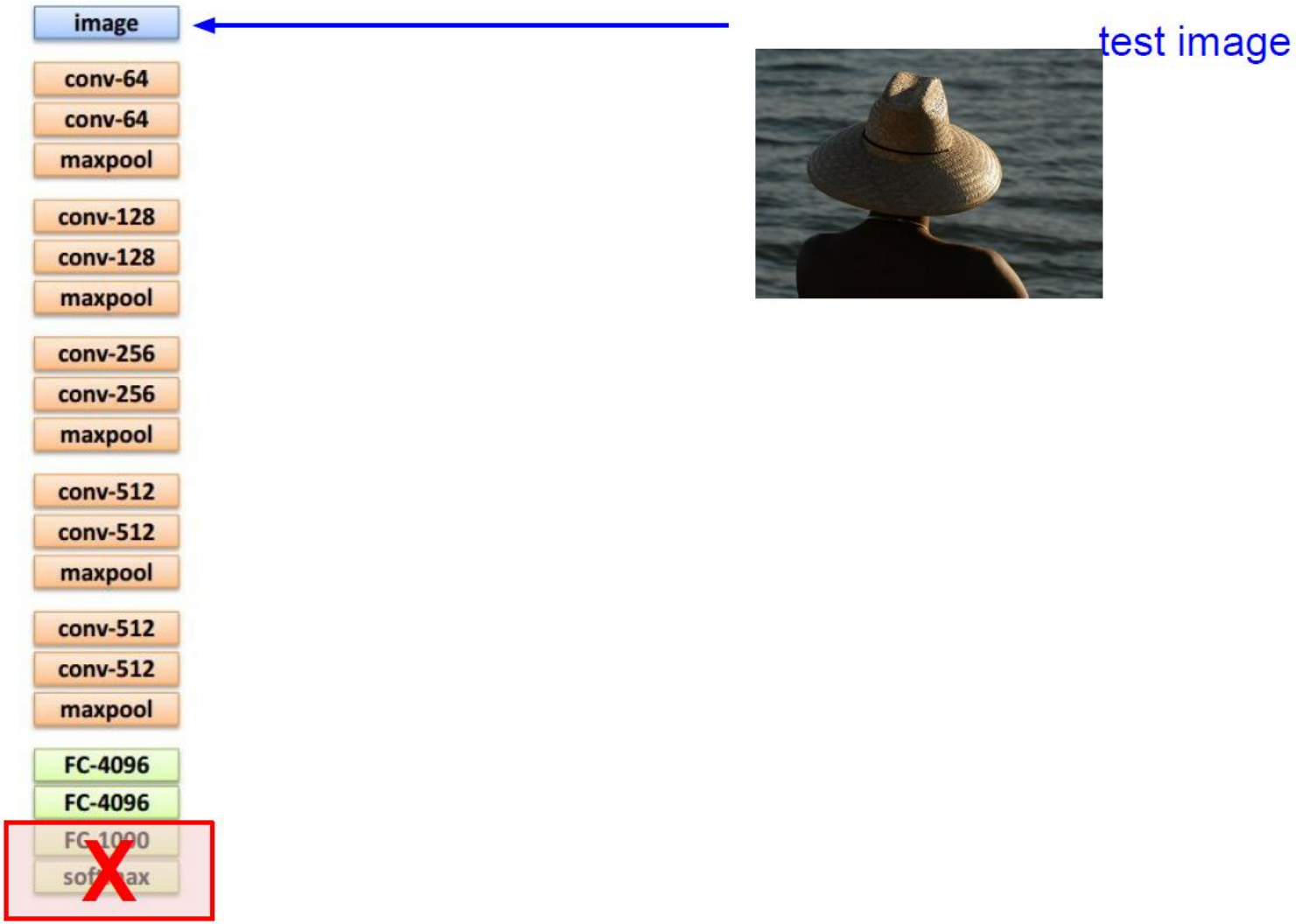


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
X
test image

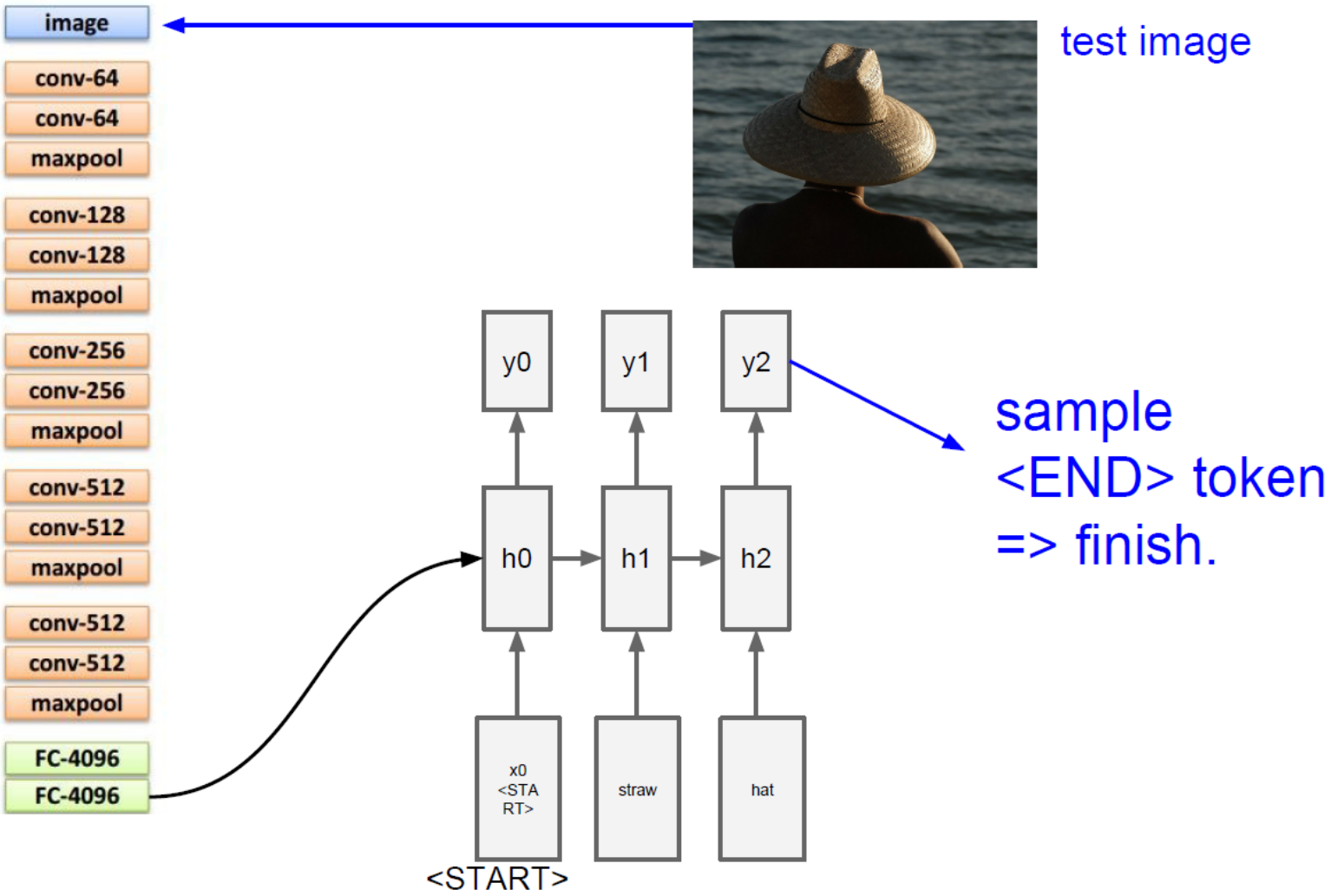


image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
test image
y0
y1
y2
h0
h1
h2
x0 <START>
straw
hat
<START>
sample <END> token => finish.