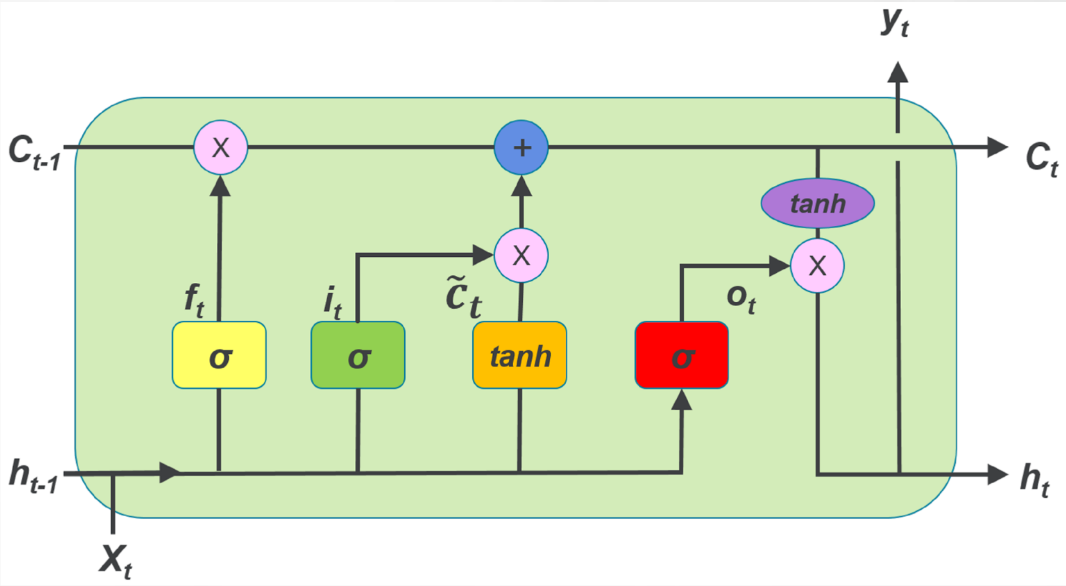
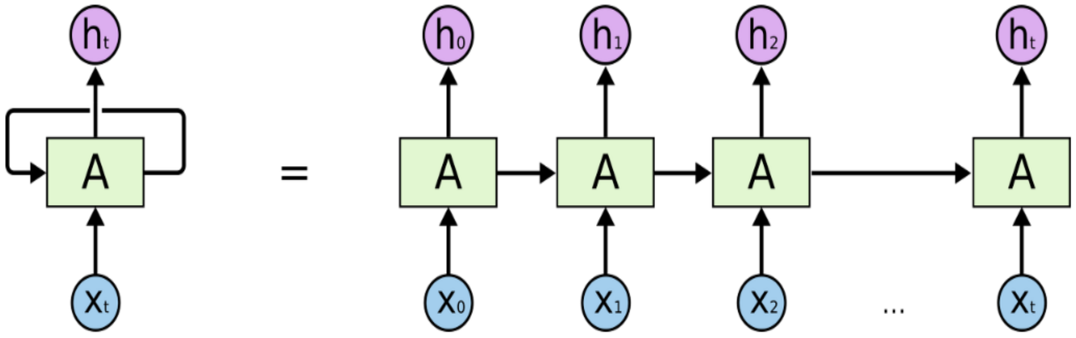
Chapter 09. 시계열을 활용한 딥러닝 (Time Sequence Processing)

RNN, Seq2Seq, LSTM, GRU



Recurrent Neural Network

- **Recurrent :** 반복되는 -> 시퀀스의 모든 요소에 대해 동일한 작업을 수행하고 출력은 이전 계산에 의존하기 때문에 *반복적* 이라고 합니다.
- RNN은 지금까지 계산 된 것에 대한 정보를 캡처하는 **"메모리"**가 있다는 것입니다



folded

Un folded

RNN 의 특징 :

1) 장점 :

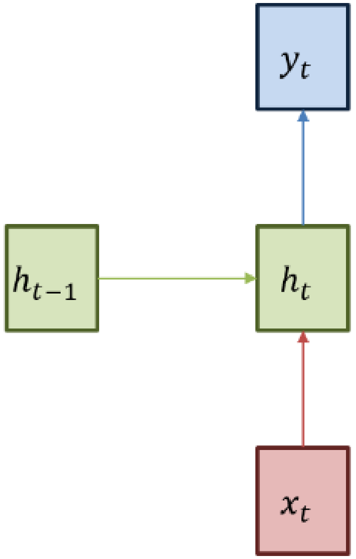
- 시퀀스 길이에 관계없는 input/output

2) 단점 :

- Gradient Vanishing : 상대적으로 짧은 sequence 학습

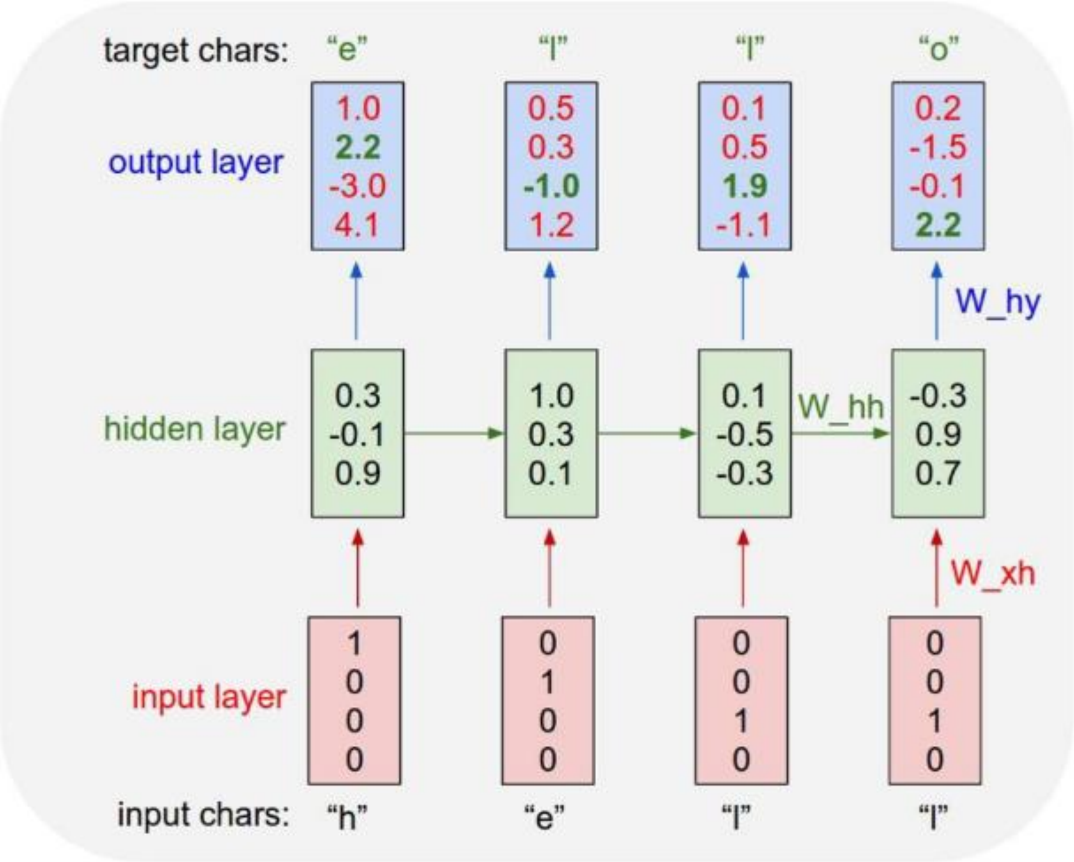
RNN forward propagation

- **RNN 은 “입력층“, “은닉층(Hidden layer)”, “출력층“ 으로 구성되있다.**



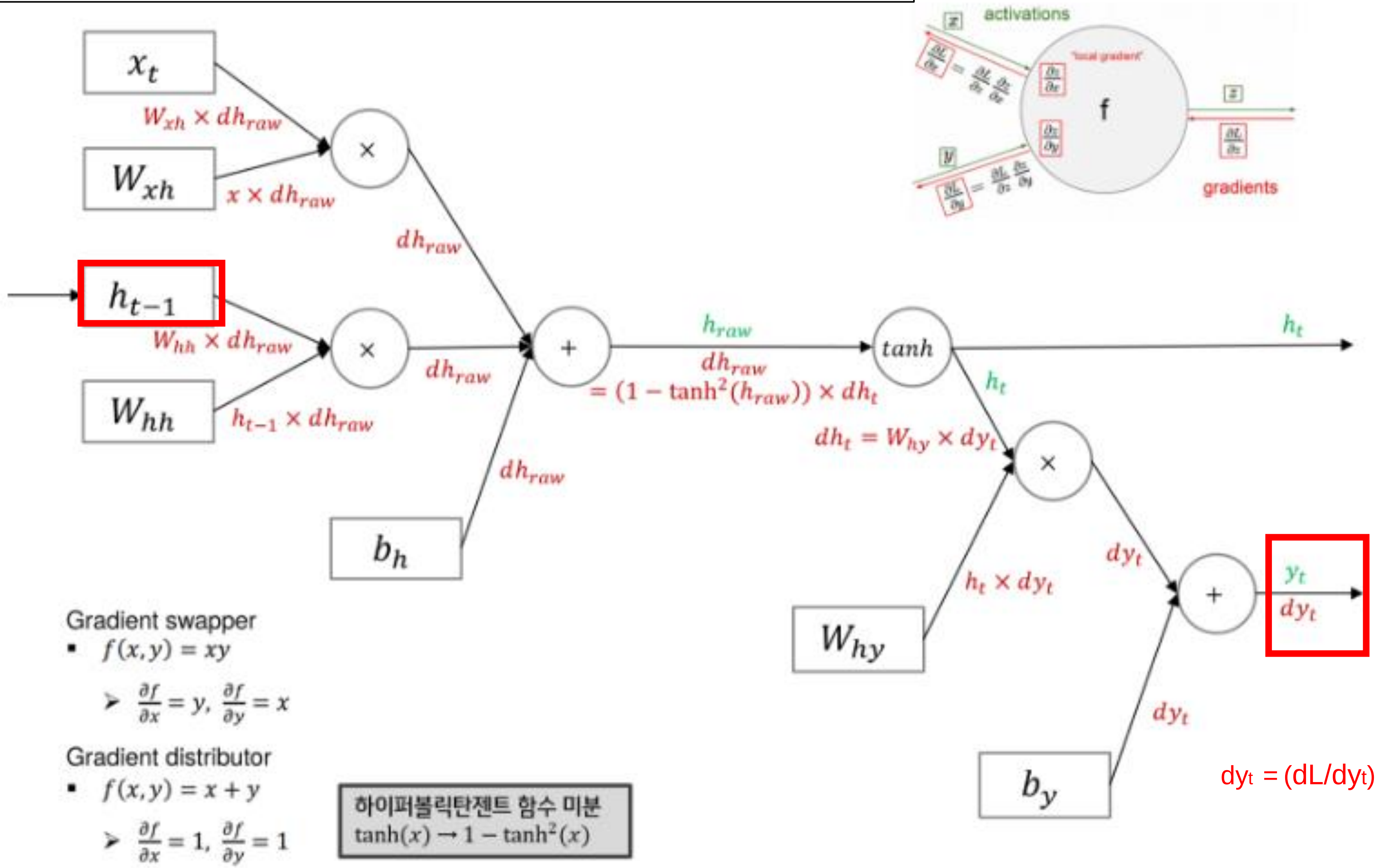
$$y_t = W_{hy} h_t + b_y$$

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



RNN backward propagation

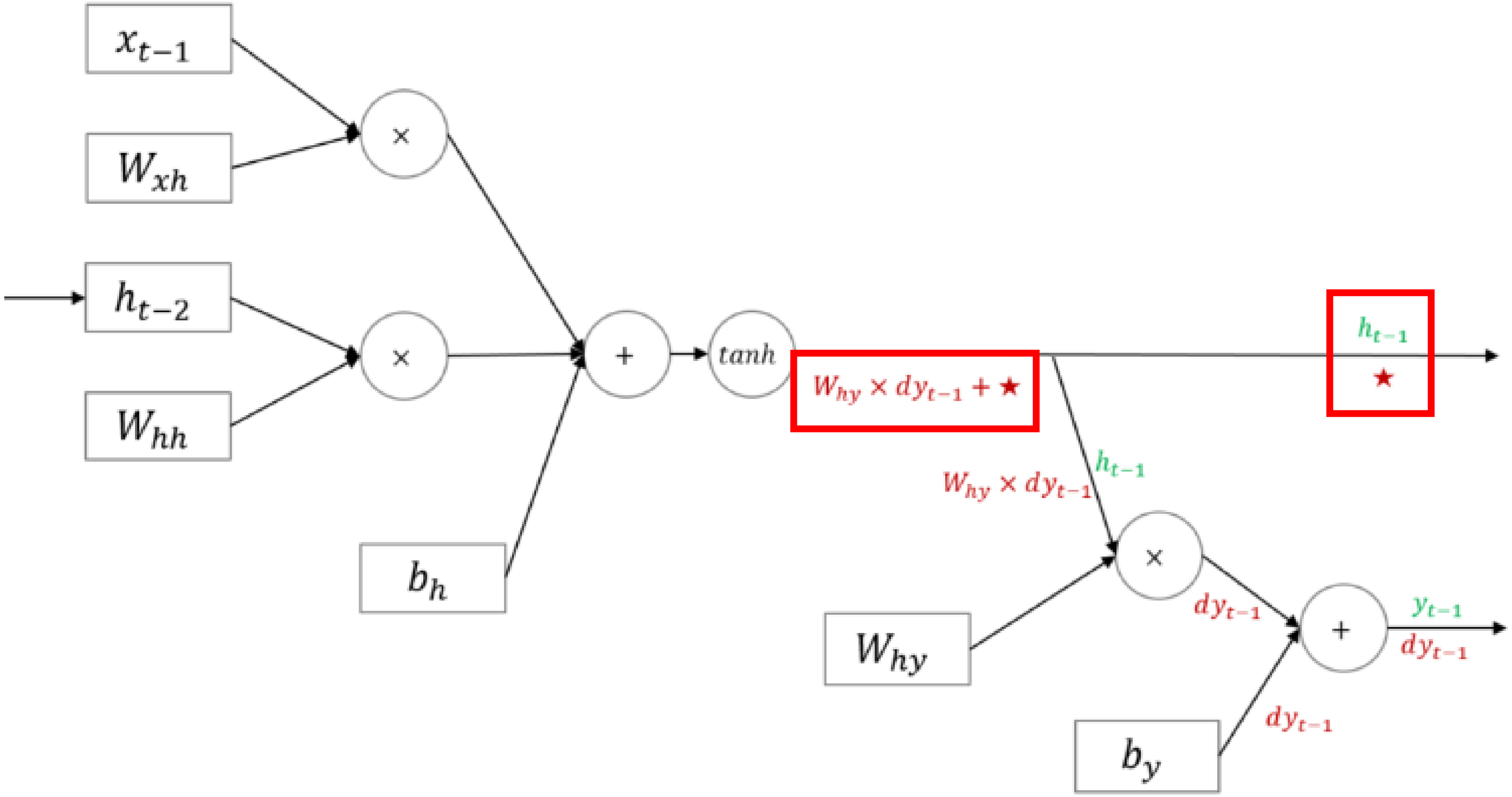
https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/



$dy_t = (dL/dy_t)$

RNN backward propagation

https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/



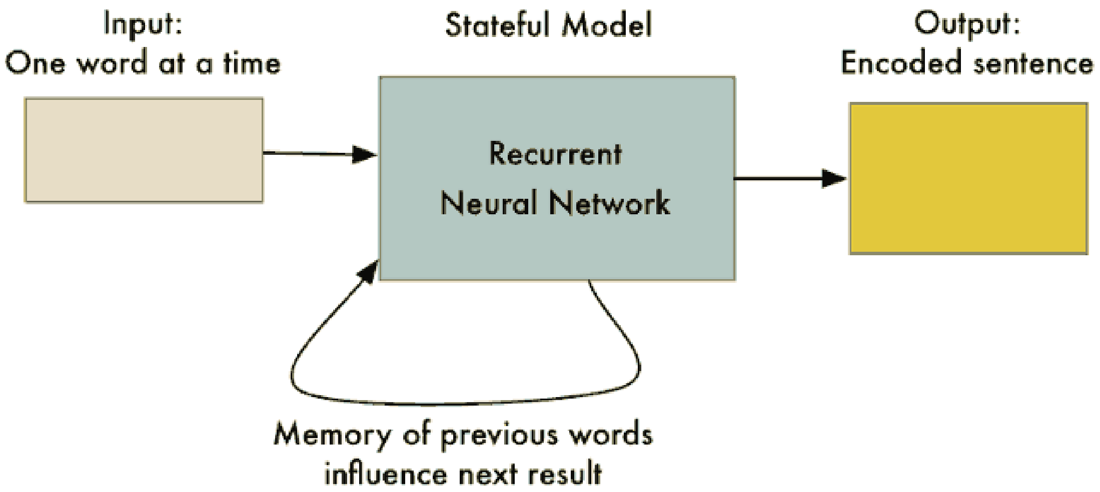
Sequence to Sequence

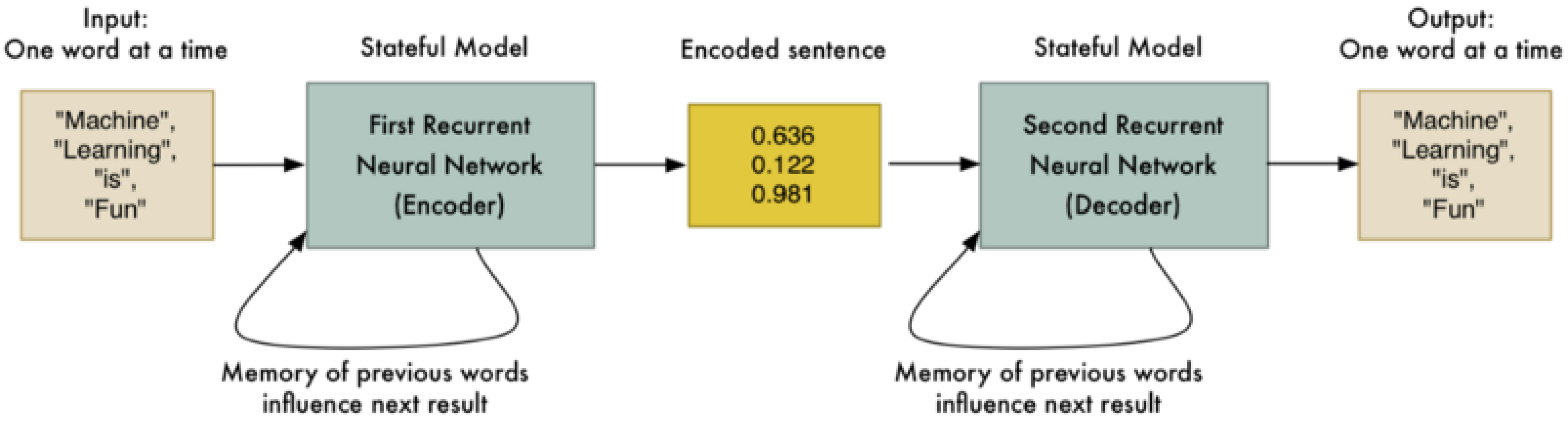
Encoding : 매우 복잡한 무엇인가를 단순한 것으로 표현할 수있게 해줍니다.

복잡한 그림, 문장을 일련의 고유한 숫자들로 표현 한다 !

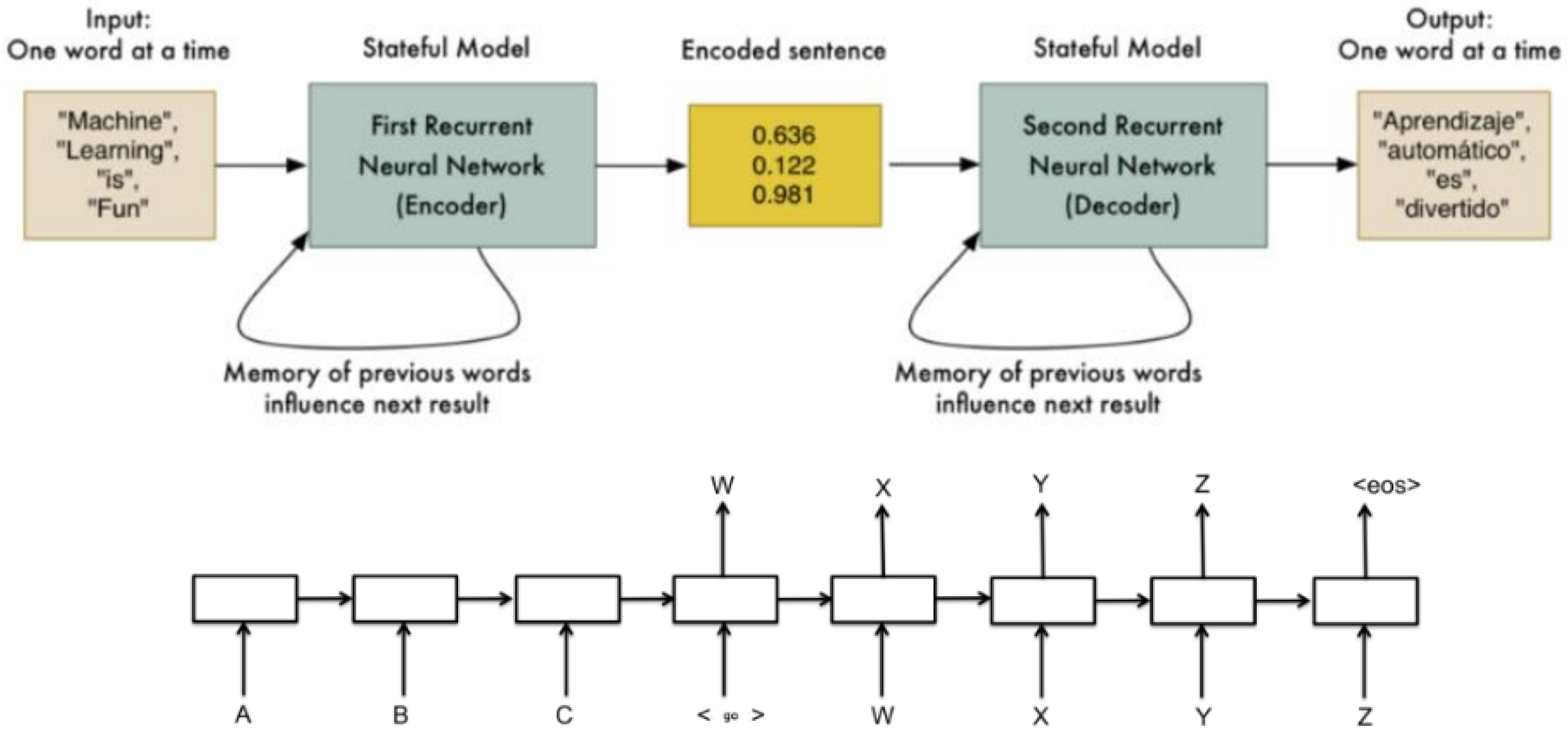
Sequence to Sequence

Sequence 2 Sequence : RNN 도 Encoder / Decoder 역할을 할 수 있다.





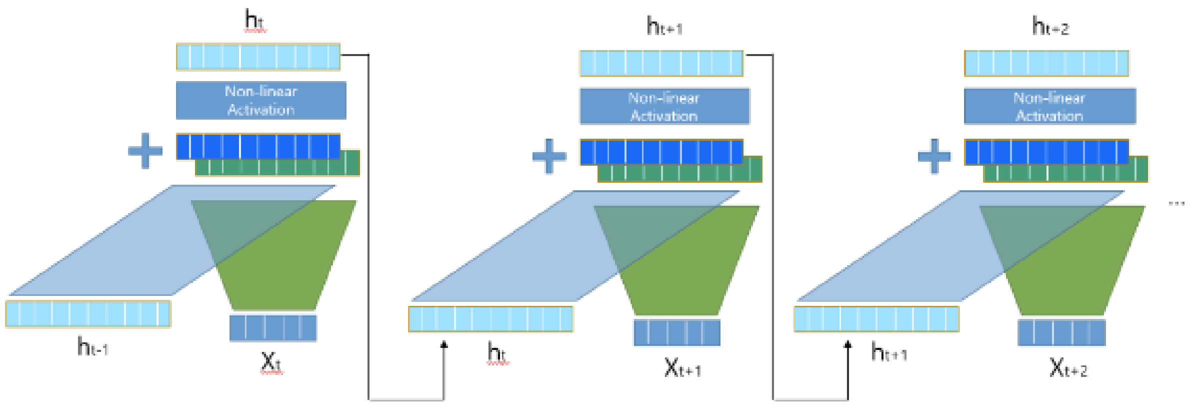
Sequence to Sequence



RNN Problem

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

Vanishing Gradient Problem



$$h_{t-2} = tanh(W[h_{t-3}, x_{t-2}])$$

$$h_{t-1} = tanh(W[h_{t-2}, x_{t-1}])$$

$$h_t = tanh(W[h_{t-1}, x_t])$$

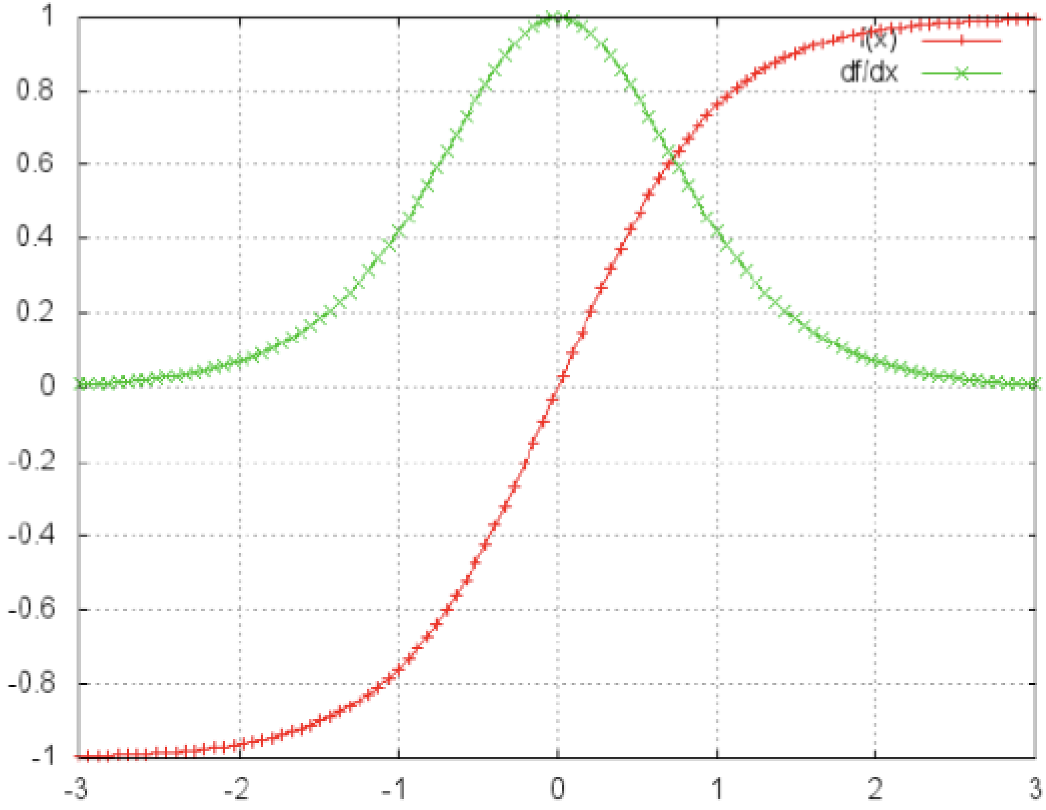
$$h_t = tanh(W[tanh(..tanh(..h_{t-3})), x_t])$$

So many tanh(x)!

RNN Problem

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

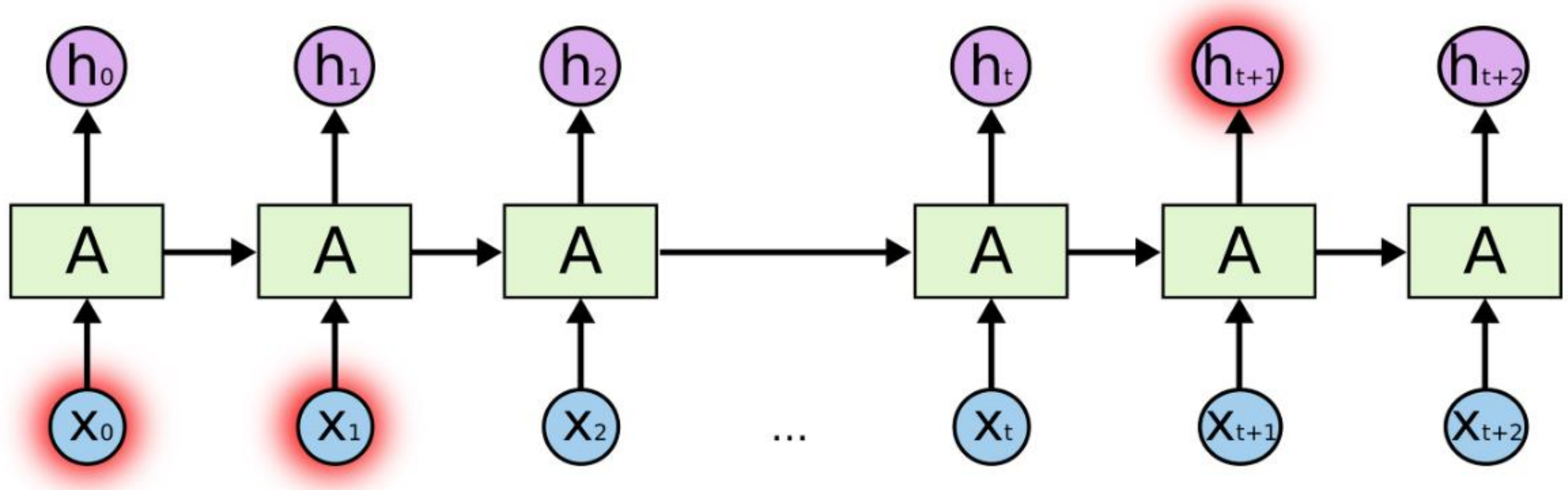
Vanishing Gradient Problem



Graph of $tanh(x)$ (red) $\frac{d}{dx}tanh(x)$ (green)

LSTM

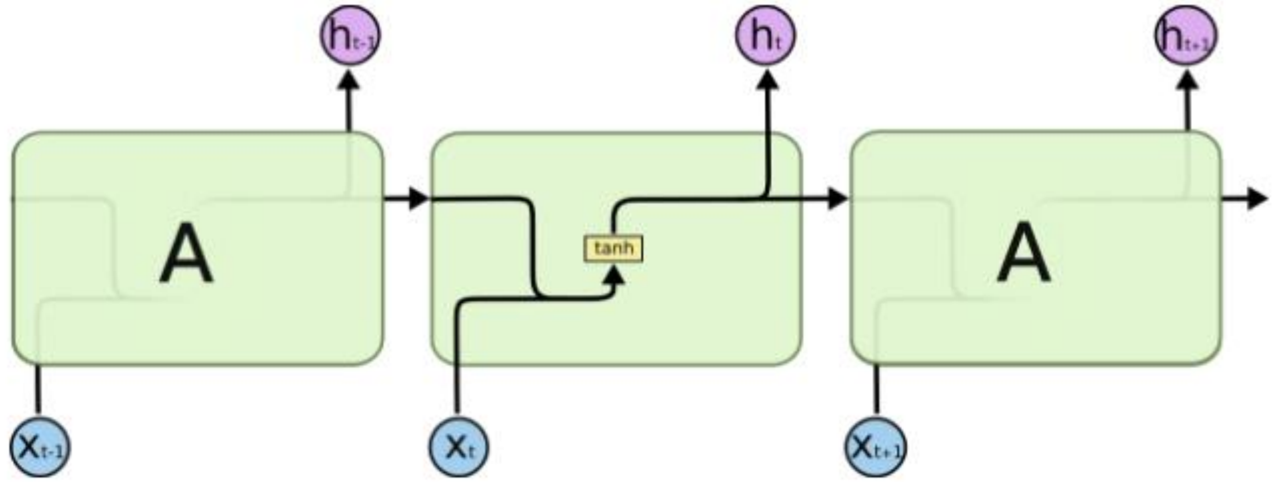
RNN 의 문제점 : Gradient Vanishing 문제로 장기간에 걸친 시간의존성은 학습시킬 수 없다.



Long Short Term Memory : 장시간에 걸친 시간 의존성, 단기간에 걸친 시간의존성 모두를 학습시킬 수 있는 기법.

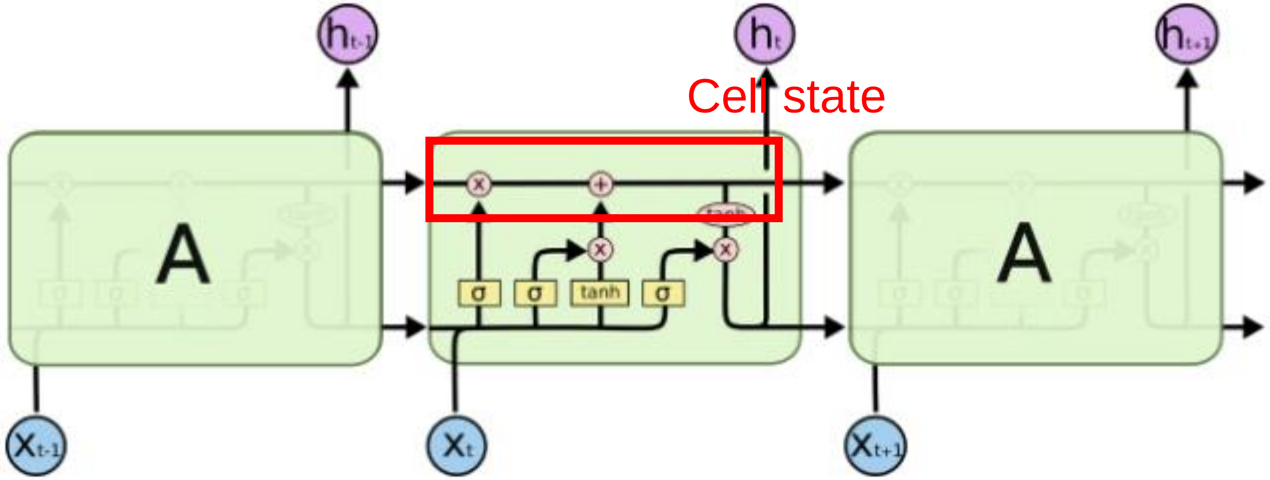
fast campus

LSTM



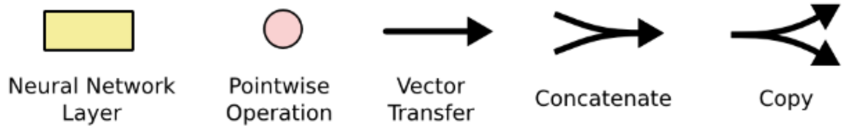
싱글 레이어를 가지고 있는 반복되는 표준 RNN 모듈

: RNN

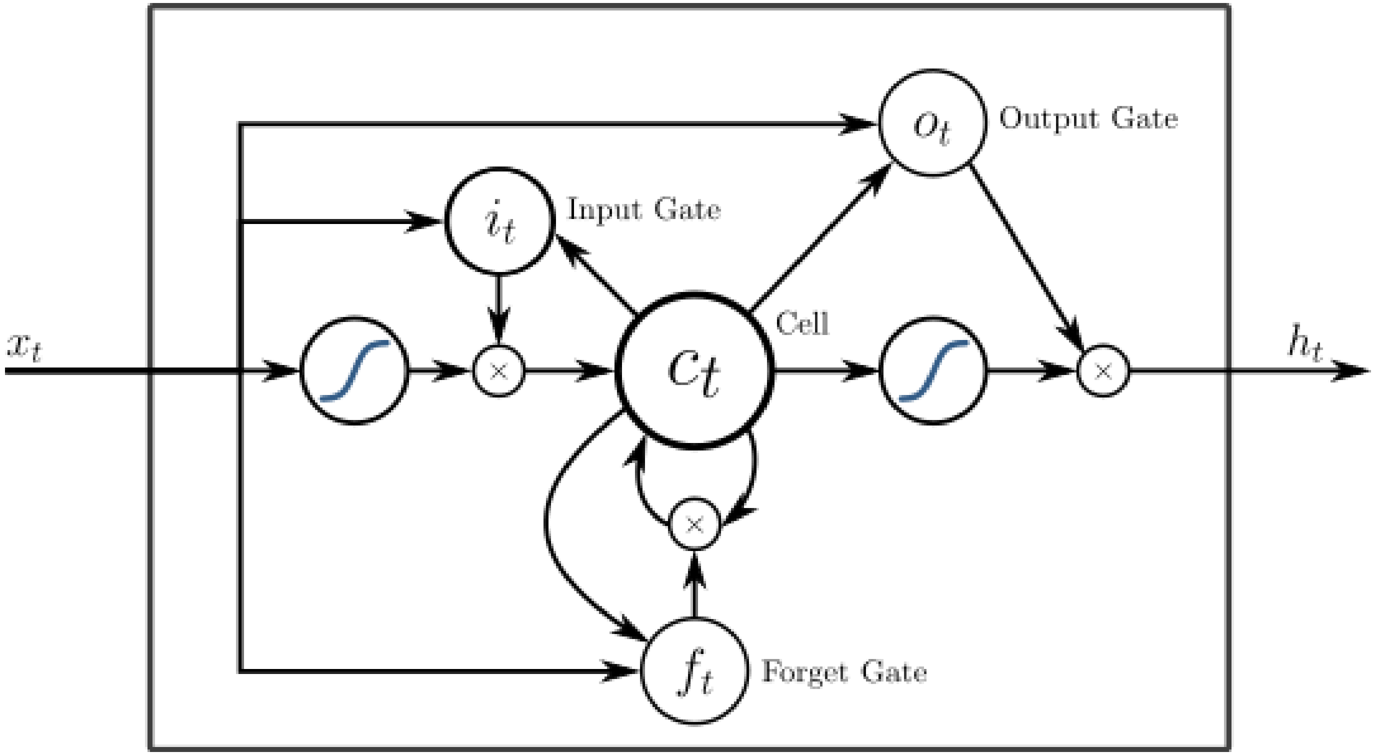


LSTM에 들어있는 4개의 상호작용하는 레이어가 있는 반복되는 모듈

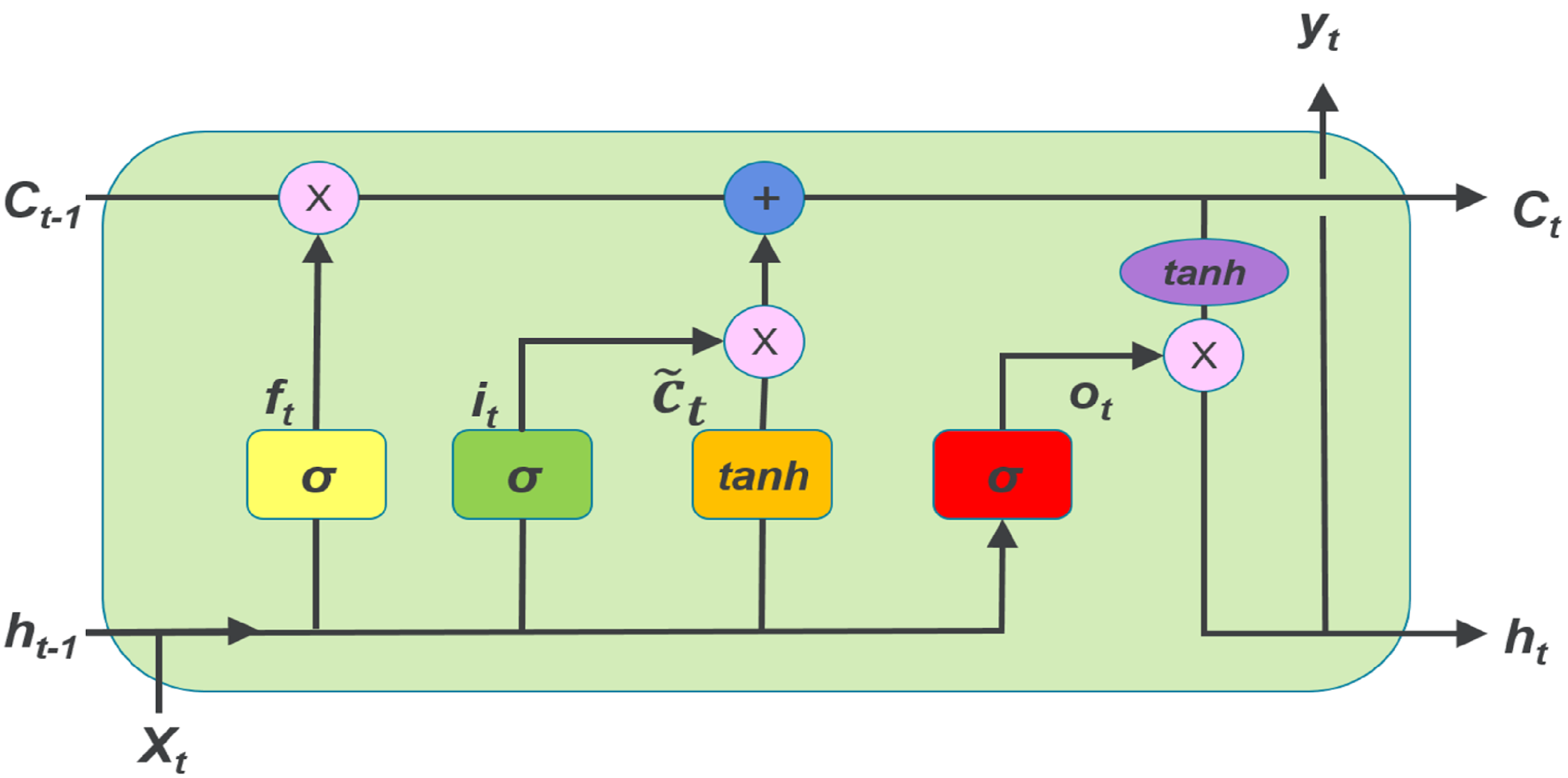
: LSTM



LSTM

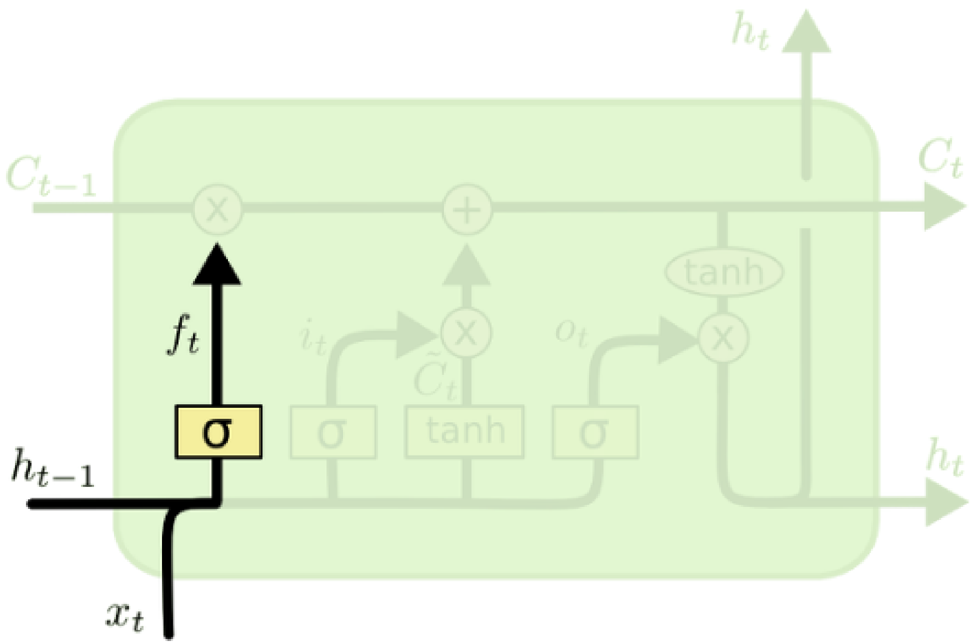


- **Input gate**
- **Forget gate**
- **Output gate**


y_t
C_{t-1}
X
+
C_t
tanh
X
X
f_t
i_t
$\tilde{C}_t$
o_t
σ
σ
tanh
σ
h_{t-1}
h_t
X_t

LSTM

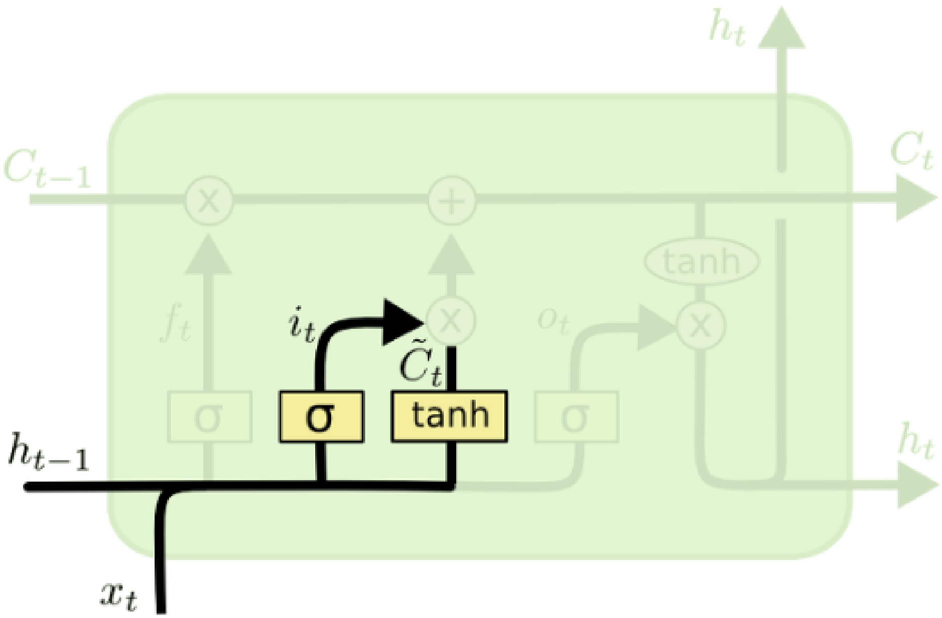
- **forget gate :** 시간 t 에서의 정보의 중요도에 따른 망각 결정



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

LSTM

- **Input gate:** 시간 t 에서의 정보를 받아들이는 양 결정

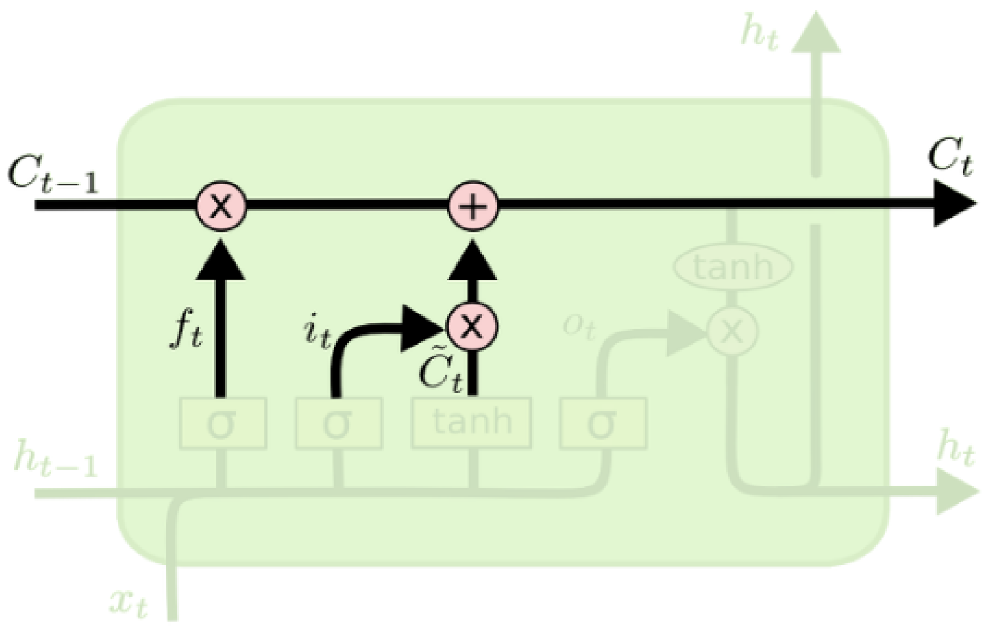


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

LSTM

- **Cell state :** 내부 기억에 관련된 state

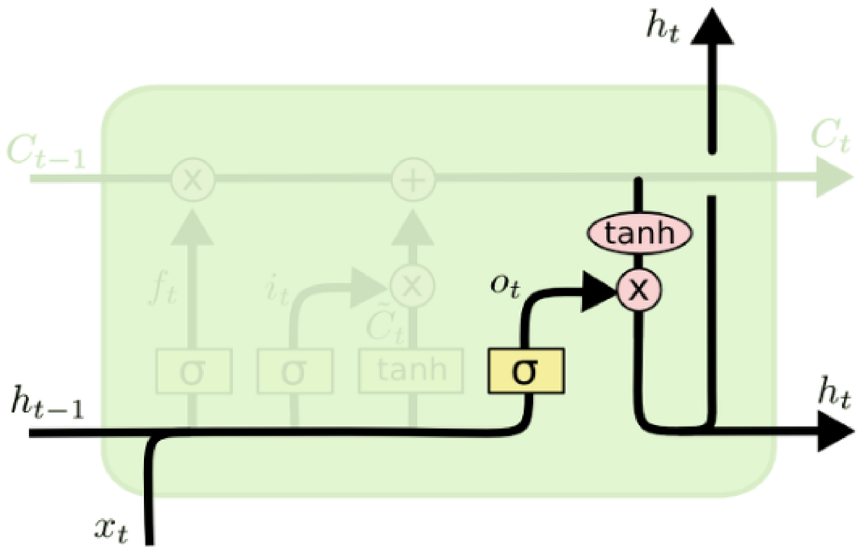


Input 정보를 얼마나 받아들일지
이전 cell state 를 얼마나 망각할
지 결정

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

LSTM

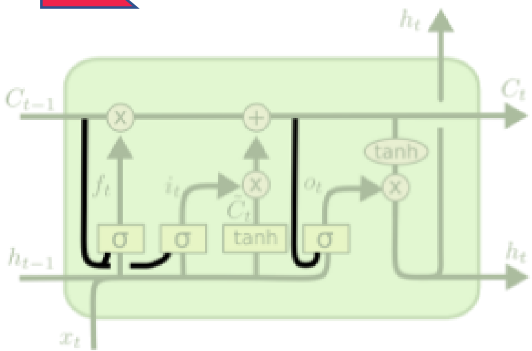
- **Output gate :** 다음 연속된 모듈에 hidden state 결정



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

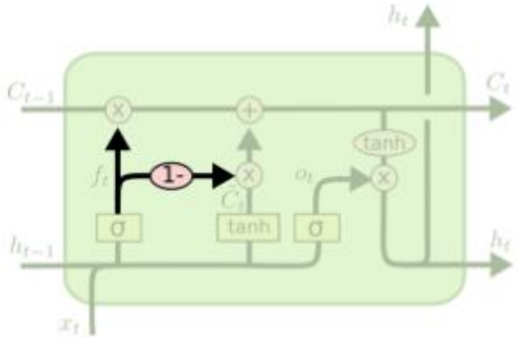
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM and others



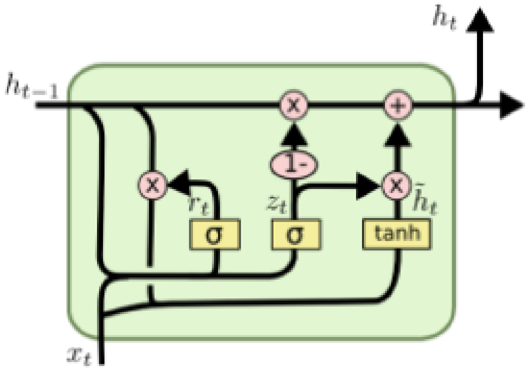
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

: **핍홀 변형 –** cell state 의 제어를 보다 용이하게한 구조



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

: **merge input / forget gate 변형**

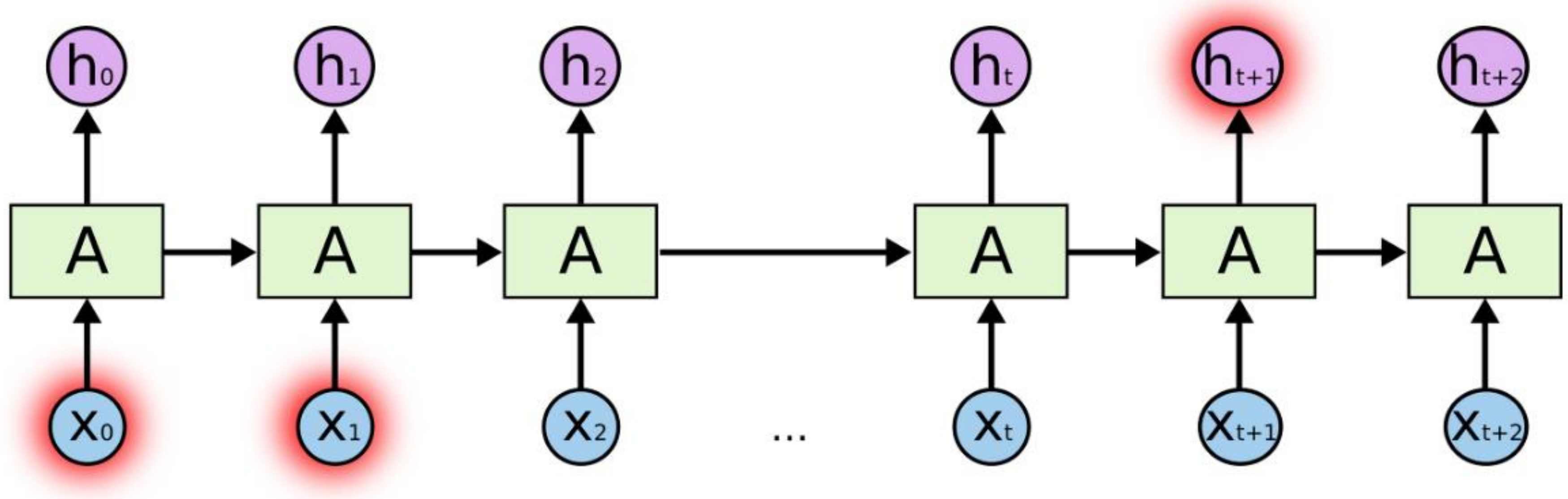


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

: **GRU 변형 –** reset / update gate 만을 사용한 간소화

LSTM

RNN 의 문제점 : Gradient Vanishing 문제로 장기간에 걸친 시간의존성은 학습시킬 수 없다.



Long Short Term Memory : 장시간에 걸친 시간 의존성, 단기간에 걸친 시간의존성 모두를 학습시킬 수 있는 기법.

LSTM

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

Long Short Term Memory Network

1. C_{t-1} 에서 불필요한 정보를 지운다.
2. C_{t-1}에 새로운 인풋 x_t와 h_{t-1}를 보고 중요한 정보를 넣는다.
3. 위 과정을 통해 C_t를 만든다.
4. C_t를 적절히 가공해 해당 t에서의 h_t를 만든다.
5. C_t와 h_t를 다음 스텝 t+1로 전달한다.

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

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

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

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

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

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

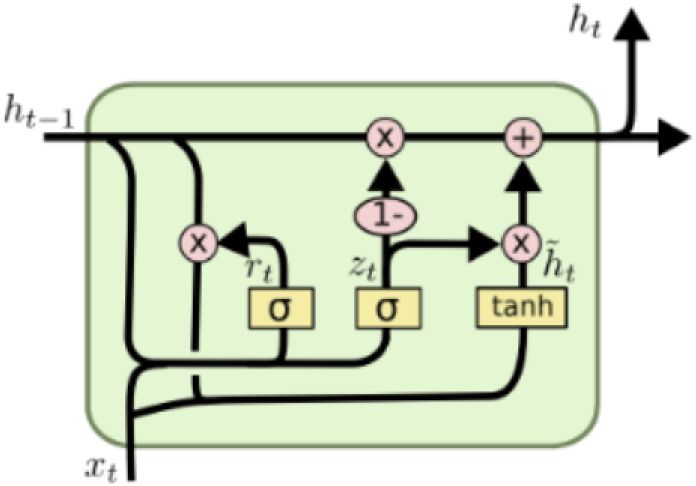
뭔가 너무 중복되는 느낌..?

→ 더 간단하게 forget gate, input gate, output gate를 해결 할 수는 없을까?

GRU

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

Gated Recurrent Unit (GRU)



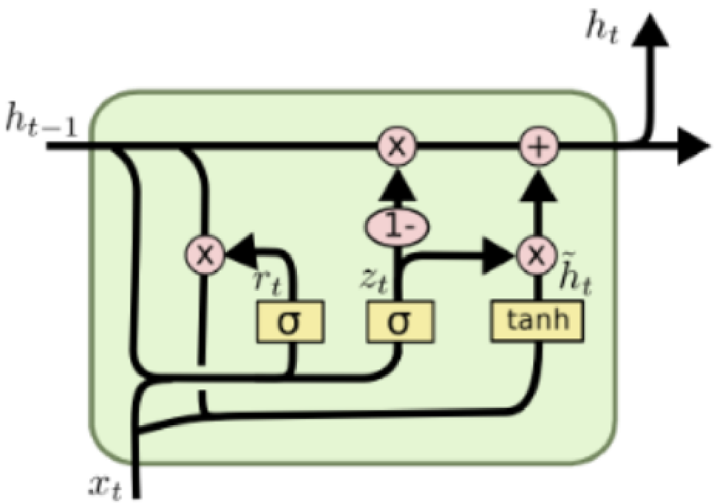
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

1. Reset gate를 계산해서 임시 h_t를 만든다.

GRU

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

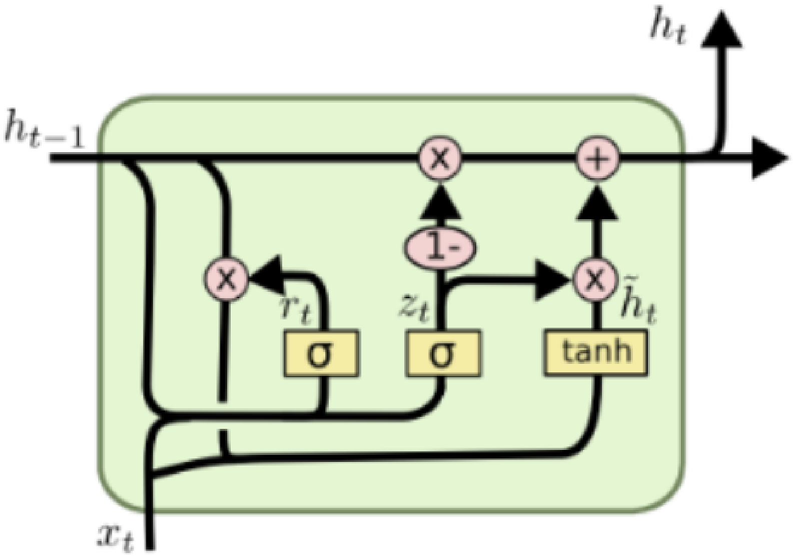


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

2. Update gate를 통해 h_{t-1}과 h_t간의 비중을 결정한다.

GRU

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf

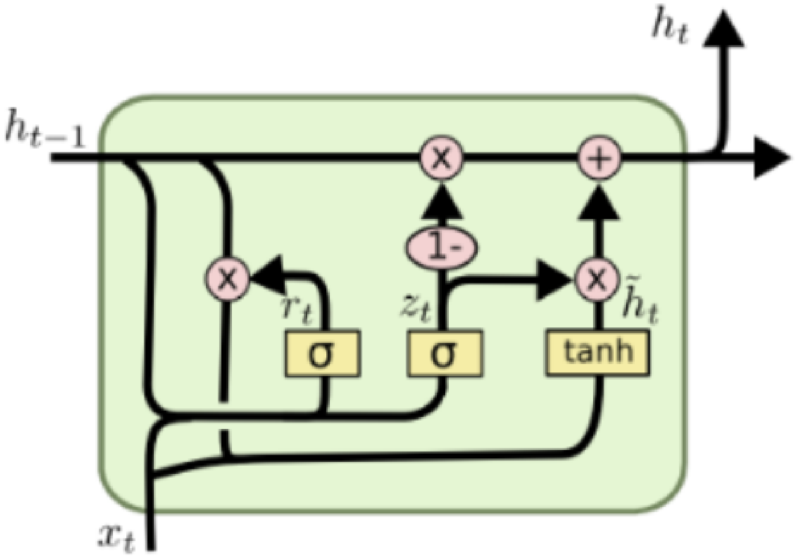


$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

3. z_t를 이용해 최종 h_t를 계산한다.

GRU

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf



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

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

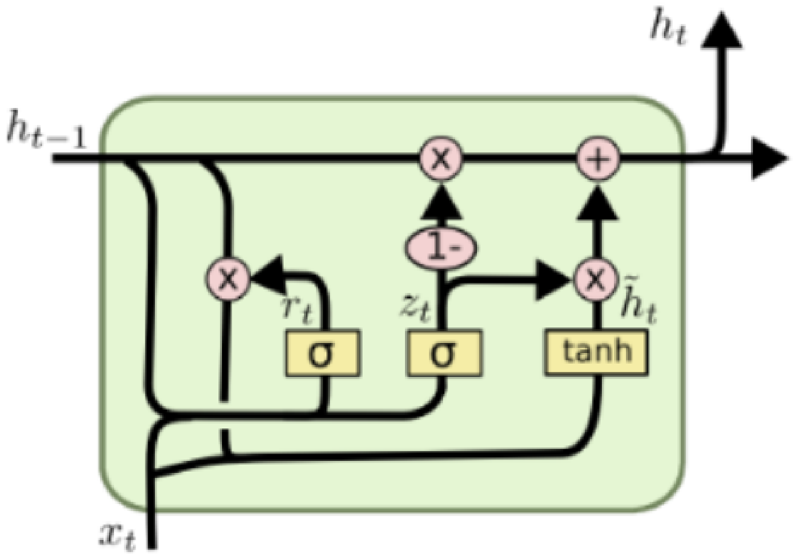
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

1. Reset gate를 계산해서 임시 ht를 만든다.
2. Update gate를 통해 ht−1과 ht간의 비중을 결정한다.
3. zt를 이용해 최종 ht를 계산한다.

GRU

https://github.com/heartcored98/Standalone-DeepLearning/blob/master/Lec8/Lec8-A.pdf



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

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

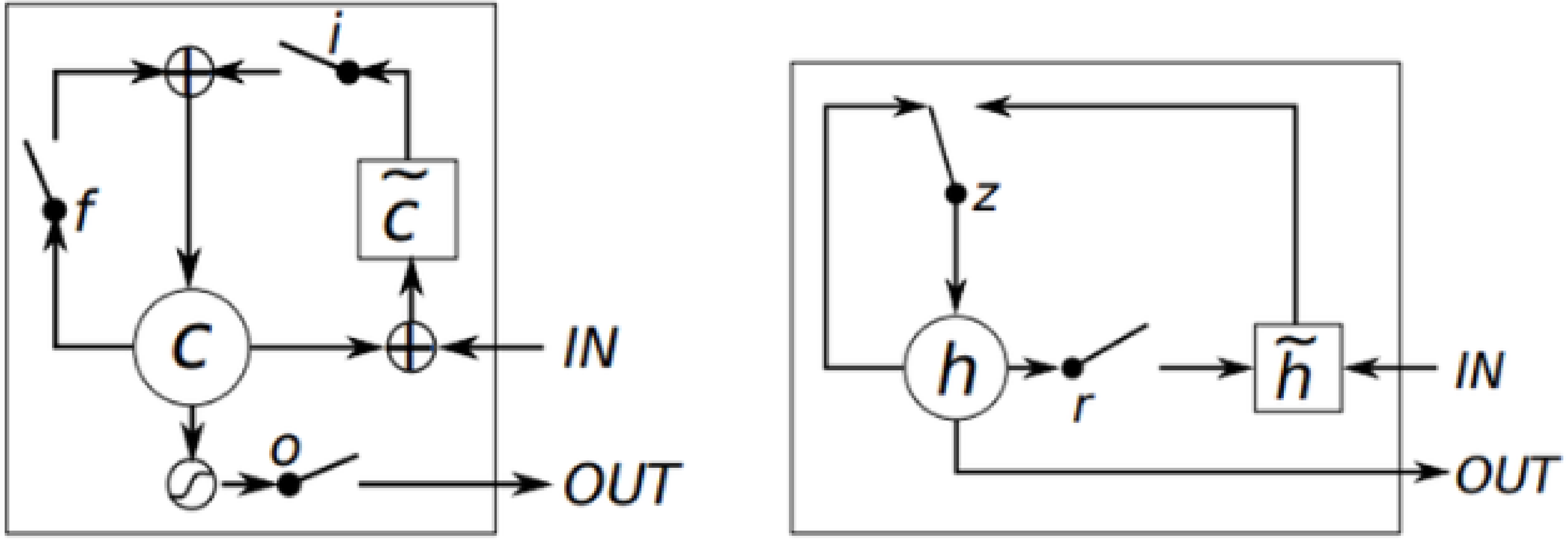
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

1. Reset gate를 계산해서 임시 ht를 만든다.
2. Update gate를 통해 ht−1과 ht간의 비중을 결정한다.
3. zt를 이용해 최종 ht를 계산한다.

GRU

https://jay.tech.blog/2016/12/08/lstmlong-short-term-memory-rnn/



(a) Long Short-Term Memory (b) Gated Recurrent Unit

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$
$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$
$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j,$$

Reference

https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45
https://excelsior-cjh.tistory.com/185

fast campus

- *Thank you*