LSTM

RNN의
한계
: gradient
vanishing
problem

LSTM의
해결 매커니즘

LSTM의 구조

LSTM의
변형 모델들

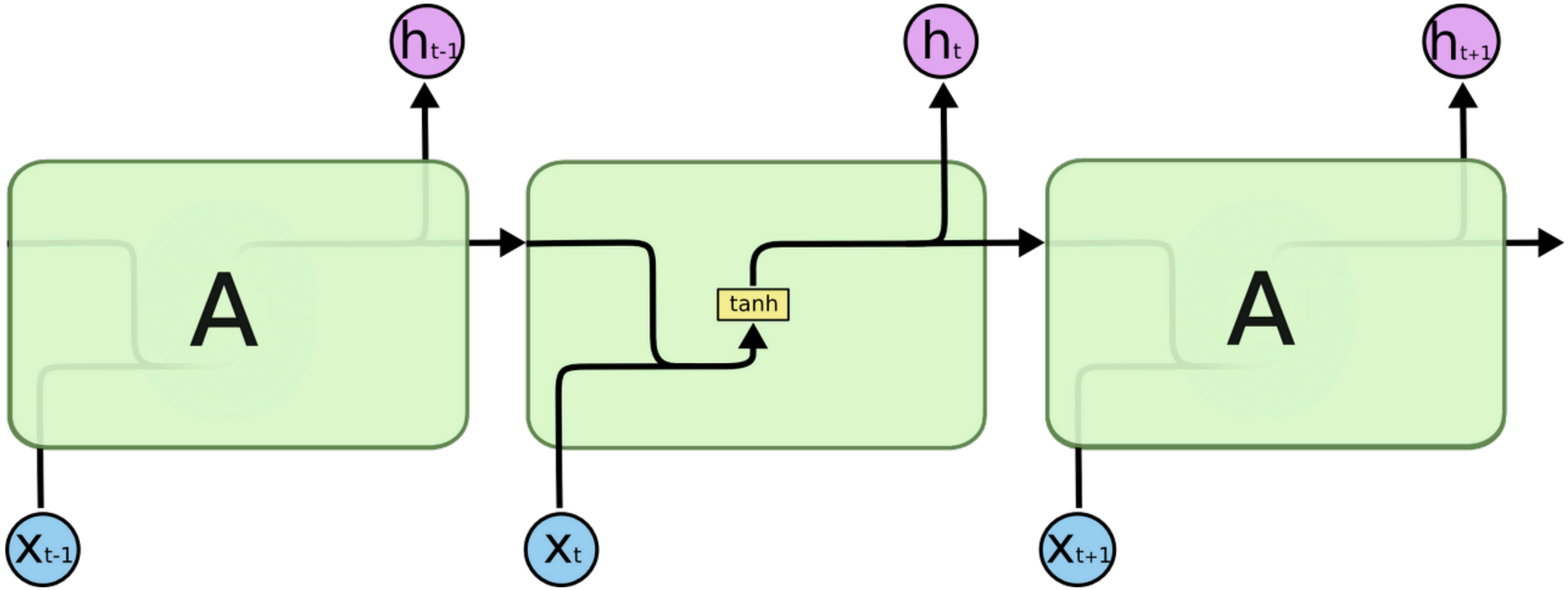
LSTM

RNN의
한계
: gradient
vanishing
problem

레이어가 깊어질수록
과거의 값들이 “ **희석** ” 된다

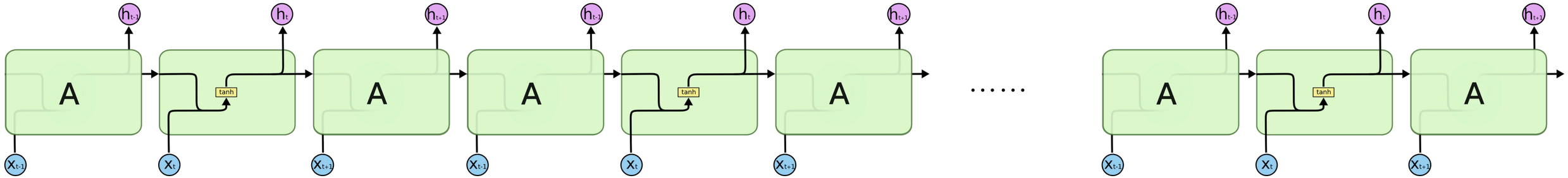
LSTM

RNN 형태



LSTM

RNN 형태 ; t시점이 계~~속 늘어난다면?



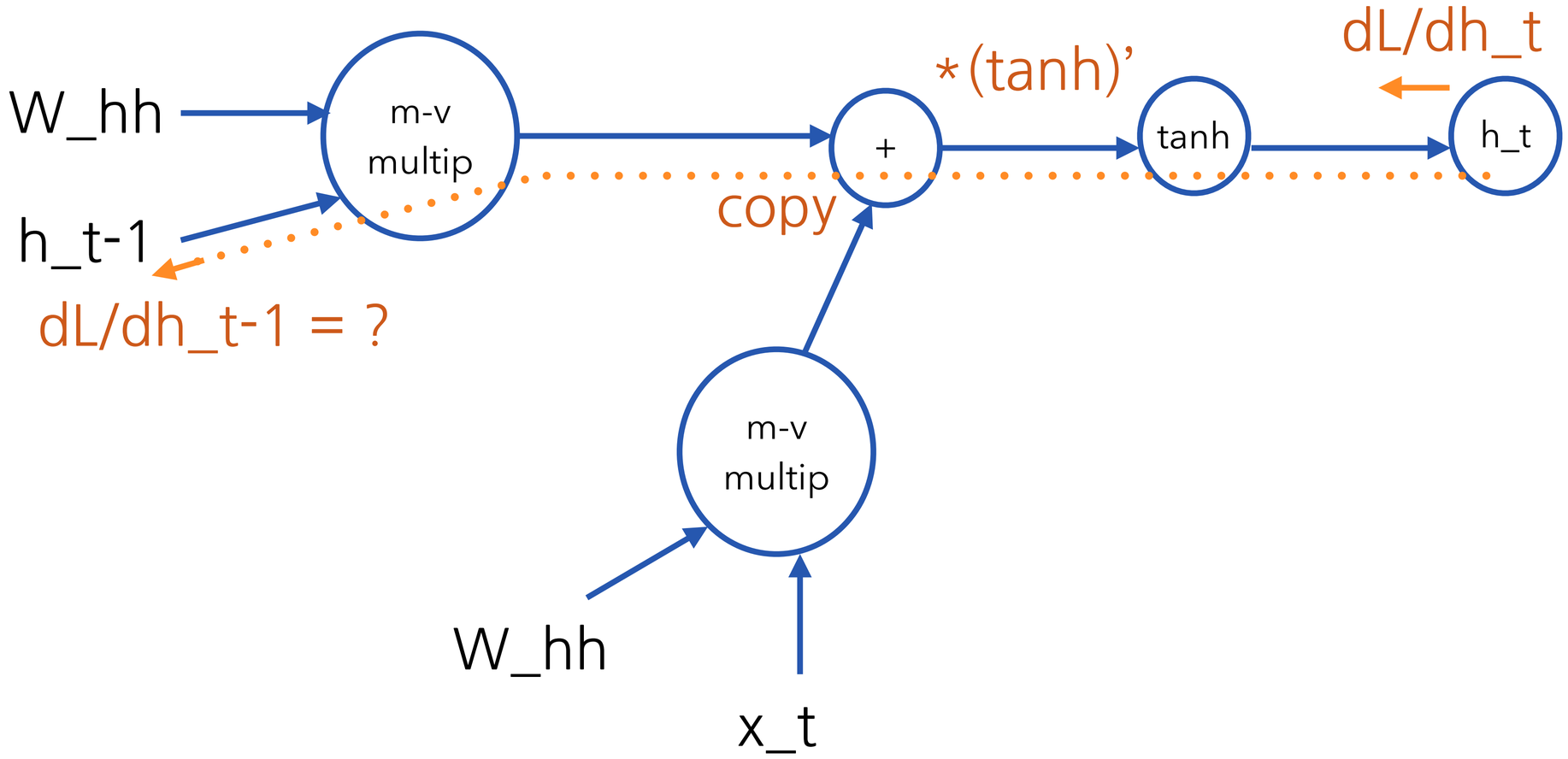
t-1의 값들은 t 상태에서 잘 가지고 있지만, t-100의 값들은 거의 영향력이 없을 것!

LSTM

RNN score function

$$h_{t+1} = tanh(W_{hh} * h_t + W_{xh} * x_{t+1})$$

computational graph 관점에서 !



LSTM

RNN score function

$$h_{t+1} = tanh(W_{hh} * h_t + W_{xh} * x_{t+1})$$

합성함수 미분 이용해서 식 계산해보면

$$\frac{dL}{dh_t} = \frac{dh_{t+1}}{dh_t} \cdot \frac{dL}{dh_{t+1}} = \tanh'(W_{hh} h_t + W_{xh} x_{t+1}) \cdot W_{hh}^T \cdot \frac{dL}{dh_{t+1}}$$

$$\frac{dL}{dh_{t-1}} = \frac{dh_t}{dh_{t-1}} \cdot \frac{dL}{dh_t} = \tanh'(W_{hh} h_{t-1} + W_{xh} x_t) \cdot W_{hh}^T \cdot \tanh'(W_{hh} h_t + W_{xh} x_{t+1}) \cdot W_{hh}^T \cdot \frac{dL}{dh_{t+1}}$$

$$\vdots$$

$$\frac{dL}{dh_{t-100}} = \begin{pmatrix} \tanh'(W_{hh} h_t + W_{xh} x_{t+1}) \\ \times \\ \vdots \\ \times \\ \tanh'(W_{hh} h_{t-100} + W_{xh} x_{t-99}) \end{pmatrix} \underline{\underline{(W_{hh}^T)^{101}}} \cdot \frac{dL}{dh_{t+1}}$$

LSTM

RNN score function

$$h_{t+1} = tanh(W_{hh} * h_t + W_{xh} * x_{t+1})$$

Gradient Vanishing Problem

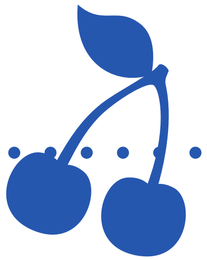
backward로 upstream gradient를 구할 때 매번 recurrent unit을 지나오는 과정에서 W_hh(T) 행렬이 제곱으로 계속 곱해진다.

W_hh(T) 행렬의 largest singular value < 1
= W_hh(T) 에 곱해지는 x 입장에서는 x의 모든 축에 대해 1보다 작은 수를 곱하는 것!

들어오는 gradient x가 vanishing되는 문제가 발생한다.

LSTM

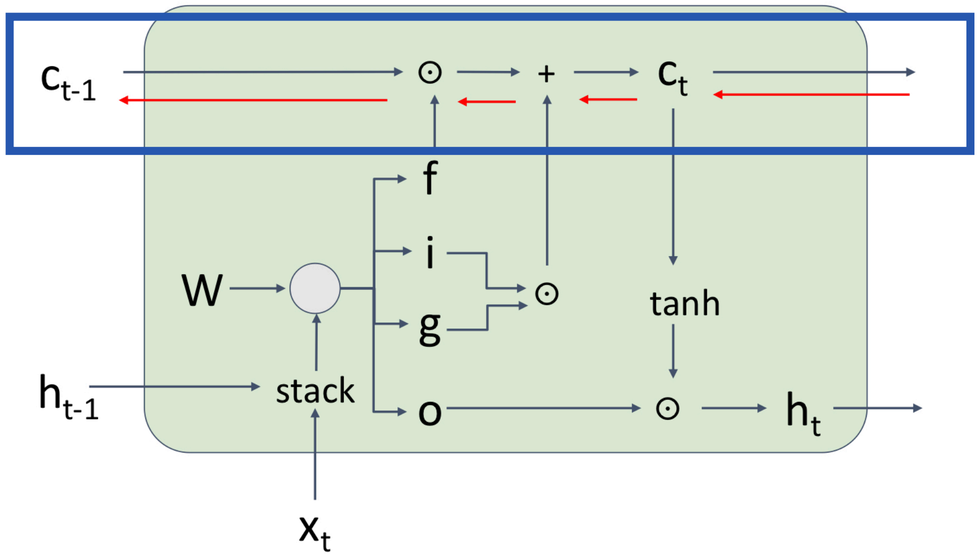
LSTM의
해결 매커니즘



LSTM의 구조

LSTM

LSTM : 긴 의존 기간의 문제를 피하기 위해 명시적으로(explicitly) 설계



C_t (cell state)

정보 갱신 담당

h_t가 정보 출력만을 관할할 수 있도록!!

LST

LST

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

정의상 $= \tanh W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$ → change to hiddenstate

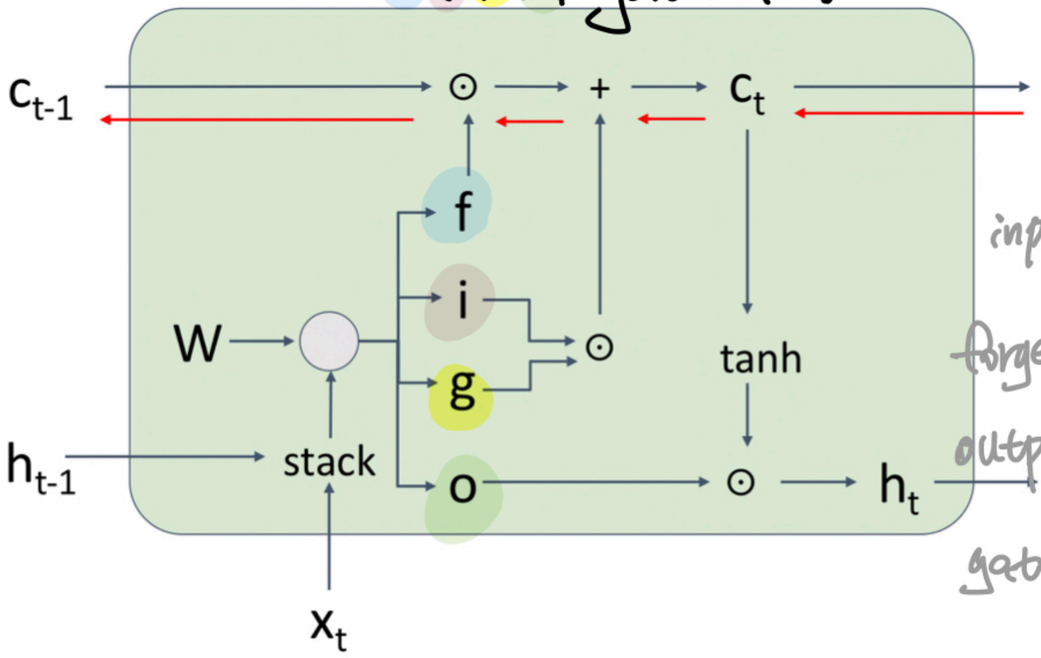
$h_t = o \odot \tanh(C_t)$

각 time step 마다 cell state 계산

$C_t = f \odot C_{t-1} + i \odot g$

LSTM! : cell state 추가

4개의 gate 추가!



* $\odot$: elementwise multiplication

input gate / forget gate / output gate / gate gate

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

!

ex) $i = \sigma(W_{hi} h_{t-1} + W_{xi} x_t)$

LSTM

RNN score function

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM score function

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

upstream gradient

$$C_t = f \odot C_{t-1} + i \odot g$$

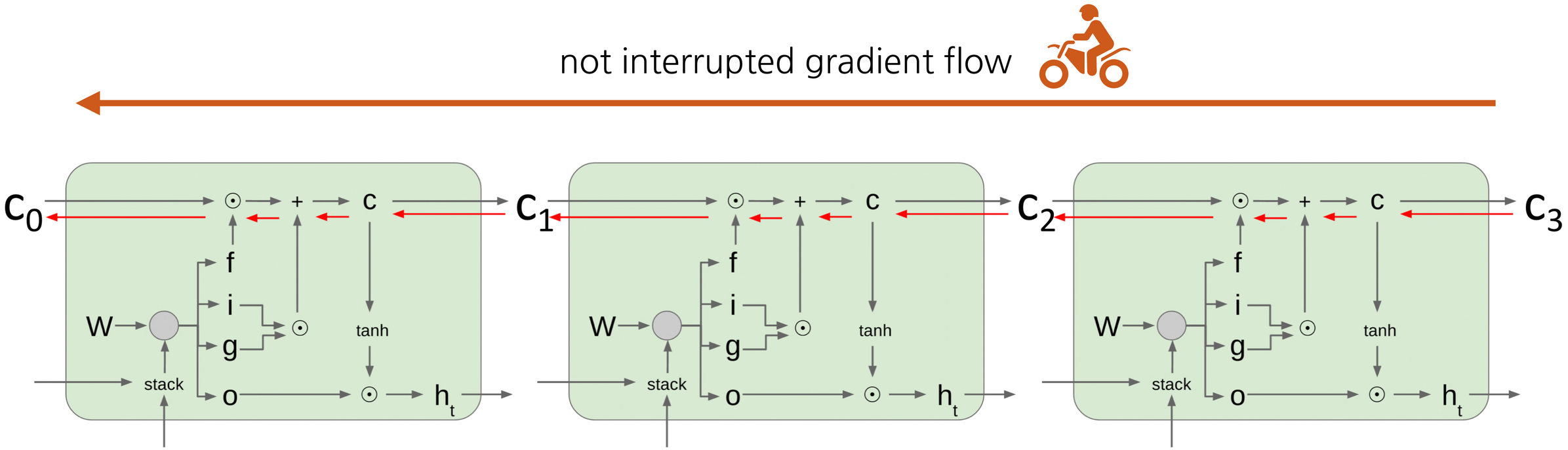
$$\frac{\partial L}{\partial C_{t-1}} = \frac{\partial C_t}{\partial C_{t-1}} \cdot \frac{\partial L}{\partial C_t}$$

$$= \mathrm{diag}(f) \cdot \frac{\partial L}{\partial C_t}$$

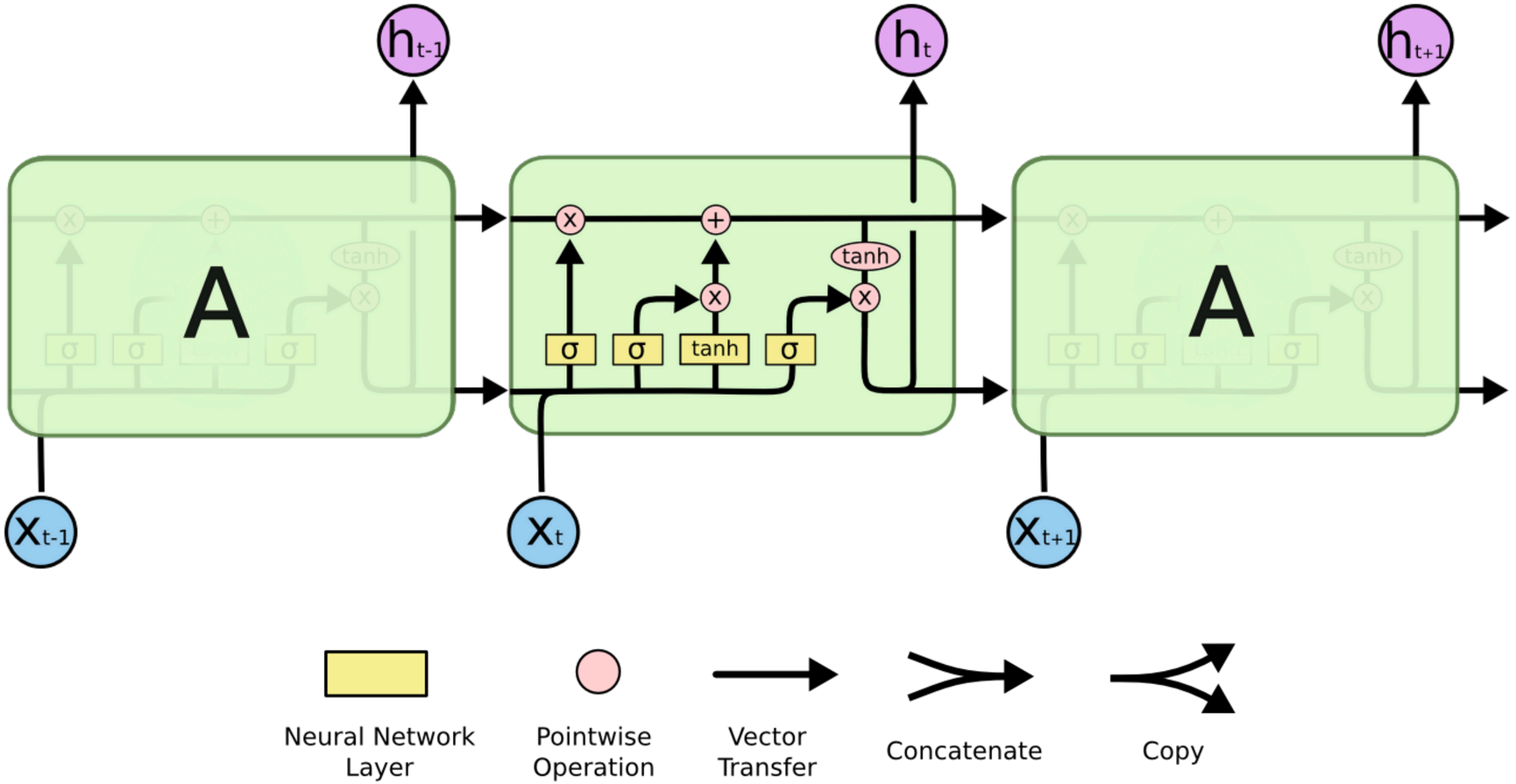
C_t 에서 C_t-1로의 backpropagation 가
W 행렬곱 꼴이 아니라
단순 elementwisely multiplicated by f
된다.

LSTM

not interrupted gradient flow

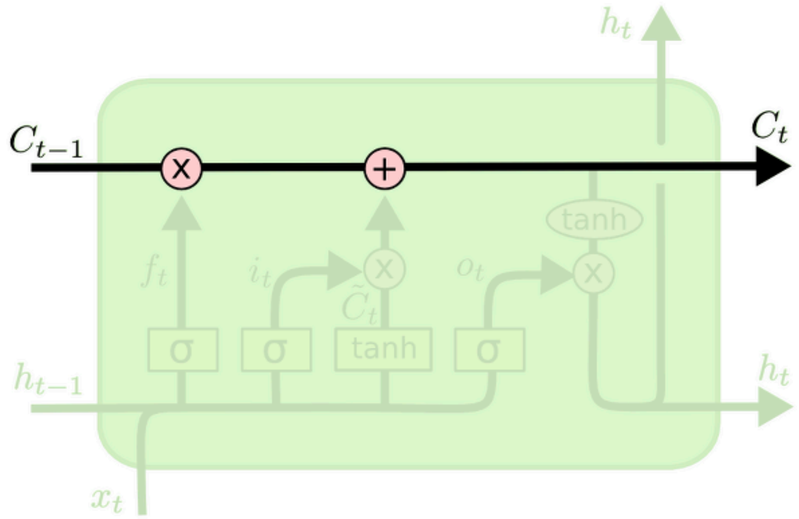


LSTM


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

LSTM

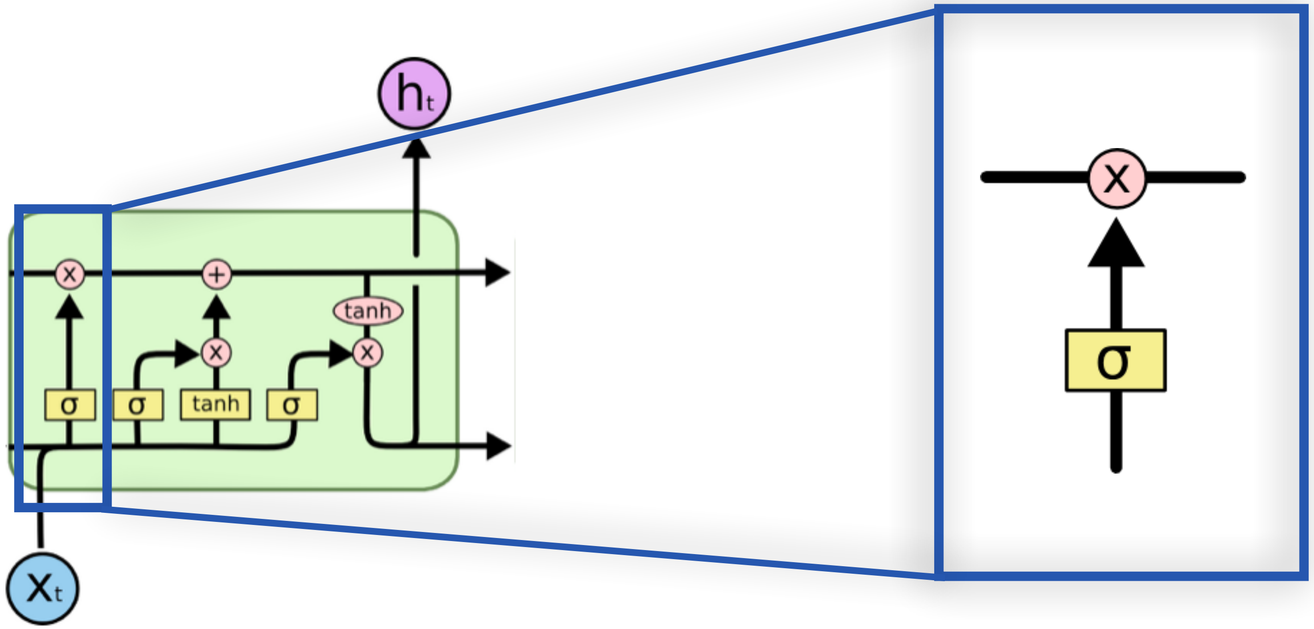
cell state



- 컨베이어 벨트
- 작은 linear interaction만을 적용시키면서 전체 체인을 계속 구동시킨다.
- 정보가 전혀 바뀌지 않고 그대로 흐르게만 한다. 쉬운 연산!

LSTM

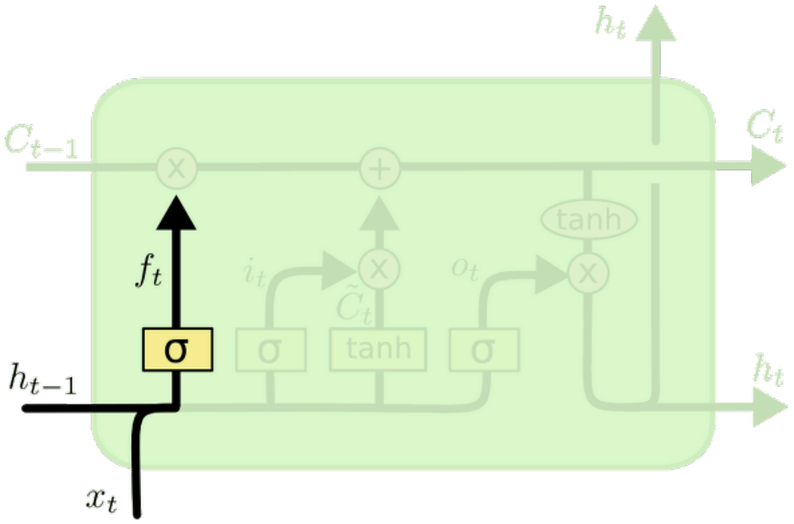
gate



- cell state에 뭔가를 더하거나 없앤다
- 정보가 전달될 수 있는 추가적인 방법, sigmoid layer, pointwise 곱셈으로 이루어져 있다
- sigmoid layer는 0~1 사이 값을 내보낸다 : 이 값이 얼마나 정보를 전달하는지의 척도
  - 0: 아무런 값도 넘기지 마라
  - 1: 값 몽땅 넘겨라

LSTM

forget gate



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

- sigmoid layer를 거치면서 어떤 정보를 버릴 것인지 결정하고 cell state에 전달
- h_t-1 과 x_t 를 받아서 0~1 사이 값을 c_t-1에 보내준다

eg. 이전 단어들 바탕으로 다음 단어 예측

Ariana Grande is a woman. Troye Sivan

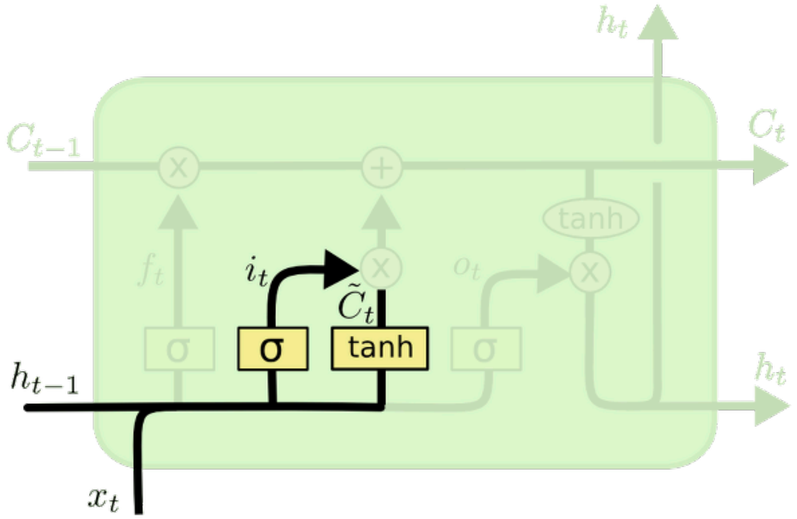
cell state
: 여자!! she she she

새 주어 등장!

forget gate
: 기억할 필요 없음

LSTM

input gate



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

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

- 들어오는 새로운 정보 중 어떤 것을 cell state에 저장할지 정함
- sigmoid layer가 어떤 값을 업데이트할 지 정하고
- tanh layer가 새로운 후보 값들 c_t vector 를 만들고 cell state에 더할 준비를 한다.
- 두 단계에서 나온 정보 합쳐서 state 업데이트

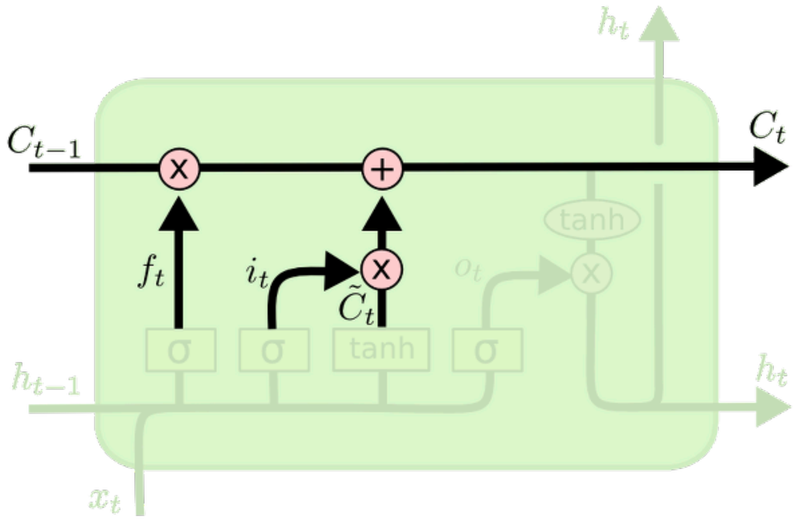
Ariana Grande is a woman. Troye Sivan

새 주어 등장!

- 새로운 주어 트로이 정보를 cell state에 더하고 싶음

LSTM

cell state 업데이트

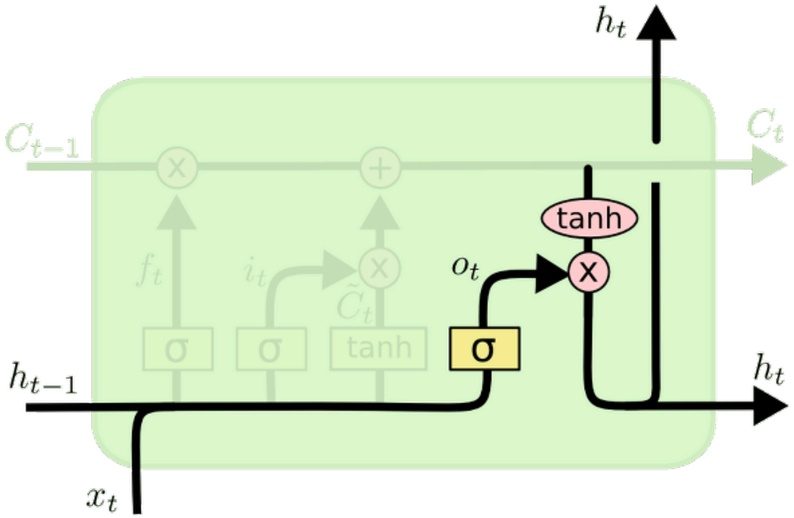


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- 앞 두 단계에서 정한대로 c_t-1 를 c_t로 업데이트
- 이전 state에 f_t 곱해서 잊어버릴거 잊어버리고
- i_t * c_t(얼만큼 업데이트할 지 정한 값) 를 더한다.

LSTM

output gate



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

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

- input 데이터가 sigmoid layer를 거쳐서 cell state의 어느 부분을 output으로 내보낼지 정한다.
- cell state를 tanh layer 거치게 하여 -1~1 사이 값으로 받고, 방금 계산했던 sigmoid layer output과 곱한다.
- 그게 최종 output

Ariana Grande is a woman. Troye Sivan

input : 주어

-> 주어 다음에 올 예측값인 output으로 적절한 답은 동사일 것이다.

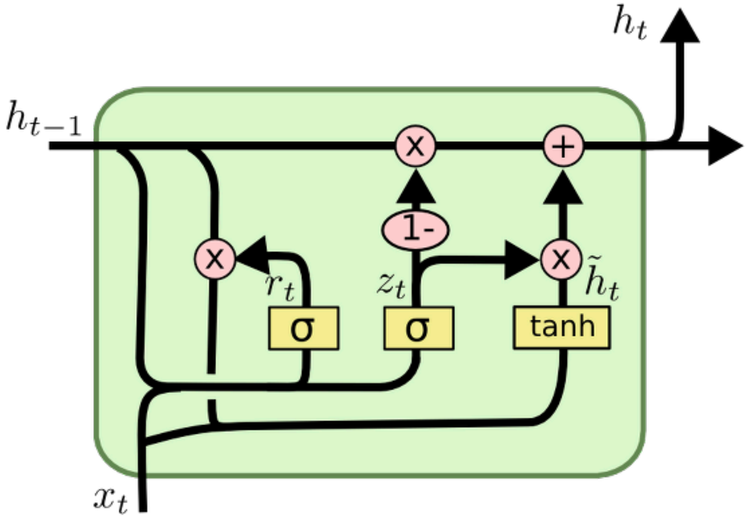
-> 최종적인 output은 앞 주어가 단수인지 복수인지에 따라 형태가 달라질 수 있는 것

LSTM

LSTM의
변형 모델들

Gated Recurrent Unit (GRU)

forget gate와 input gate를 하나의 "update gate" 합쳤고, cell state와 hidden state를 합쳤고, 또 다른 여러 변경점이 있다.
기존 LSTM보다 단순한 구조



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

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

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

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

감사합니다