Recurrent Neural Network (RNN)

Minjong Lee

POSTECH CSE

minjong.lee@postech.ac.kr

목차

- RNN
- RNN의 다양한 버전: LSTM, GRU

목차

- RNN
- RNN의 다양한 버전: LSTM, GRU

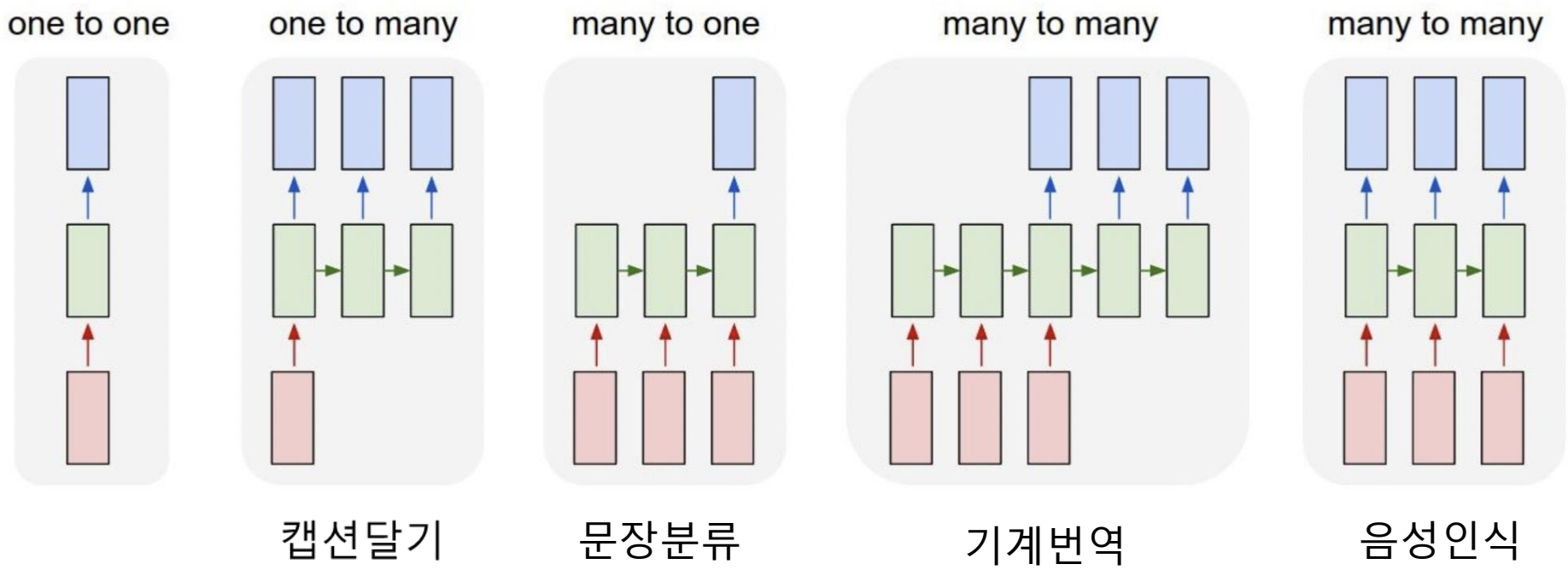
RNN이란 무엇일까?

- 실제 마케팅 / 추천 등의 분야에서는 NLP 가 쓸모가 많다
  - ➢ 비정형 데이터인 텍스트로부터 정보를 얻을 수 있고, 이걸 모델에 사용할 수 있다

- Example
  - 리뷰를 긍정 리뷰 / 부정 리뷰로 분류하기
  - 리뷰에서 나온 데이터를 뽑아서 그것을 바탕으로 맞는 상품 추천하기
  - 등등

RNN이란 무엇일까?

- 이전 Timestep의 출력 결과를 다음 입력 결과로 사용하는 NN
  - 입력/출력 데이터의 길이를 유연하게 컨트롤 할 수 있기 때문에 굉장히 유연하다



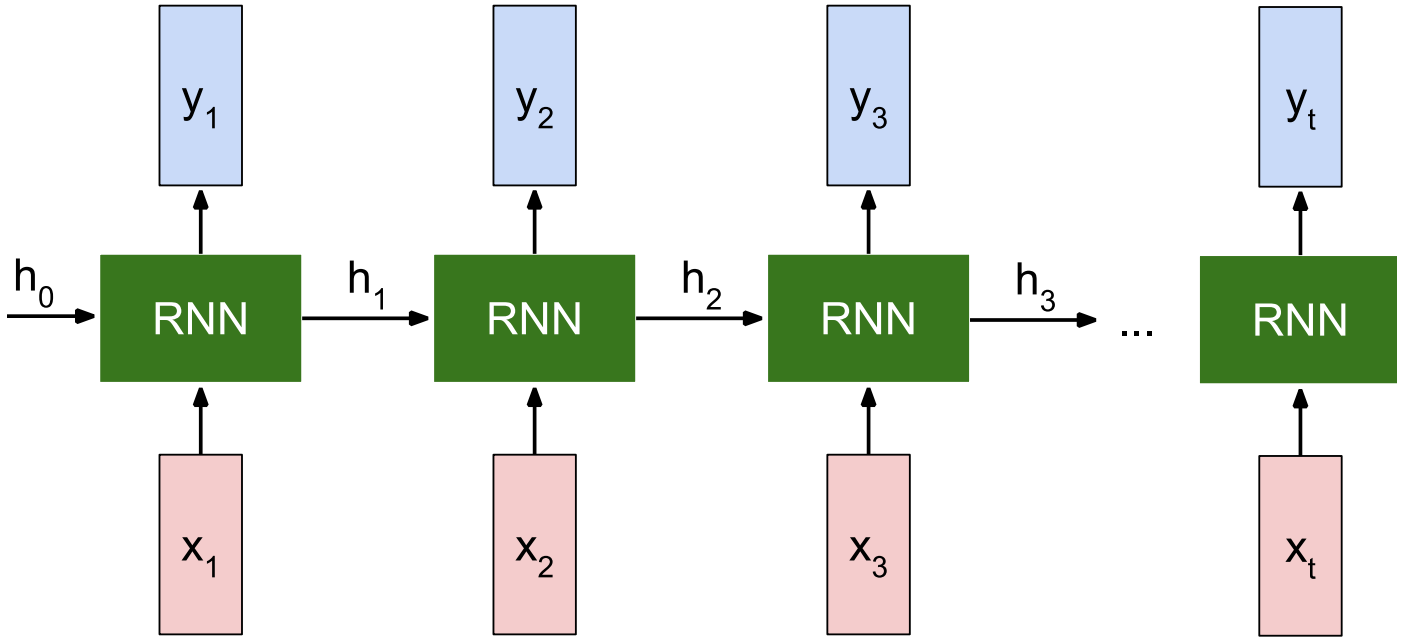
RNN이란 무엇일까?

- 각 네트워크 구조마다 다른 inductive bias를 갖고 있다

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

Table 1: Various relational inductive biases in standard deep learning components. See also Section 2.

RNN의 기본 구조



RNN의 기본 구조

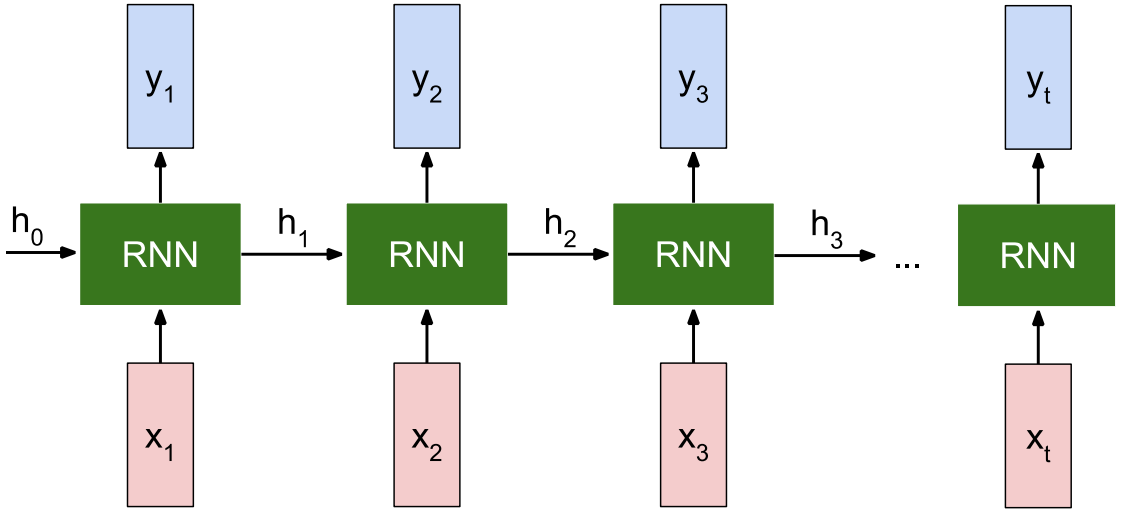
some function with parameter W

input vector at time t

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

new state

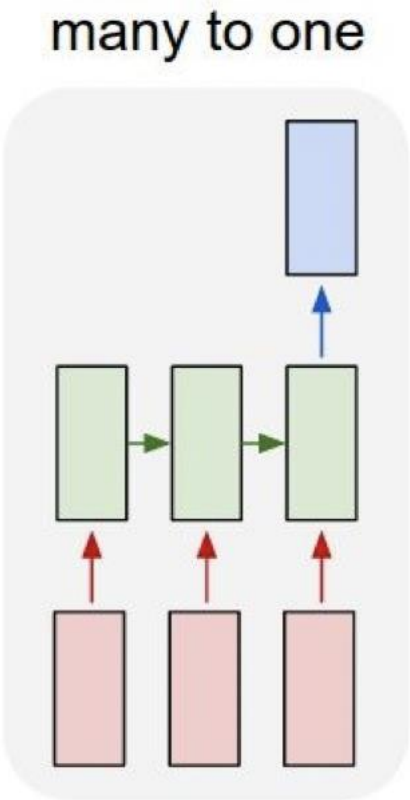
old state



$$h_t = tanh(h_{t-1}W_h + x_tW_x + b)$$

RNN: Keras API

- tf.keras.layers.SimpleRNN(4)
  - Input: (32, 10, 8)
  - Output: (32, 4)

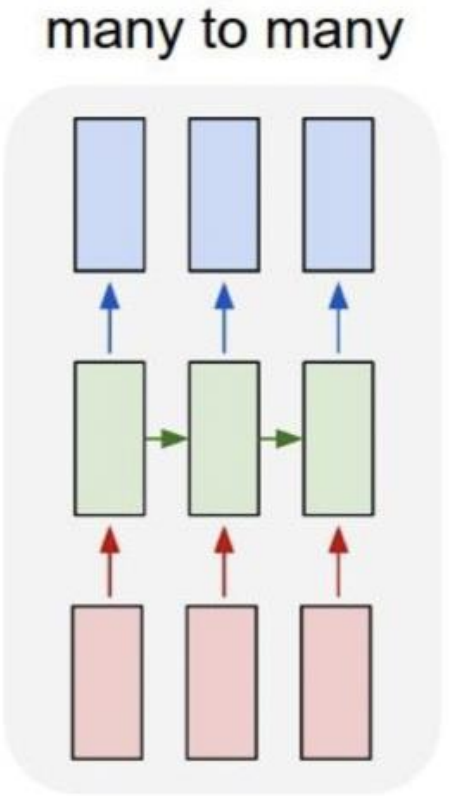



```
tf.keras.layers.SimpleRNN(
    units, activation='tanh', use_bias=True,
    kernel_initializer='glorot_uniform',
    recurrent_initializer='orthogonal',
    bias_initializer='zeros', kernel_regularizer=None,
    recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, recurrent_constraint=None, bias_constraint=None,
    dropout=0.0, recurrent_dropout=0.0, return_sequences=False, return_state=False,
    go_backwards=False, stateful=False, unroll=False, **kwargs
)
```

RNN: Keras API

- Many-to-many task: 'return_sequences=True'
- tf.keras.layers.SimpleRNN(4, return_sequences=True)
  - Input: (32, 10, 8)
  - Output: (32, 10, 4)

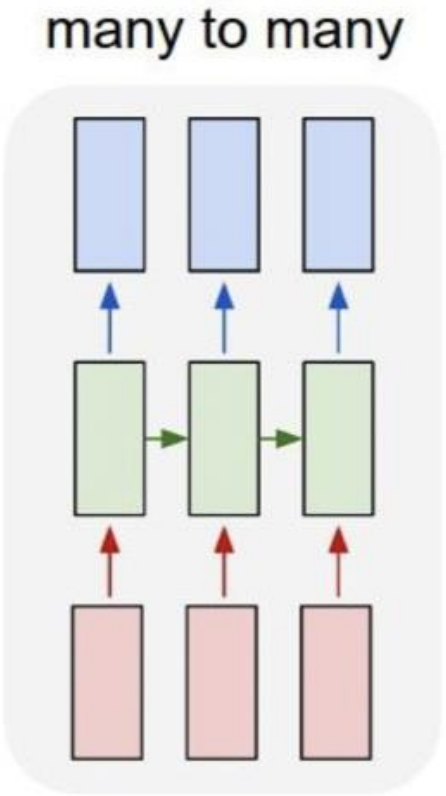
```
tf.keras.layers.SimpleRNN(
    units, activation='tanh', use_bias=True,
    kernel_initializer='glorot_uniform',
    recurrent_initializer='orthogonal',
    bias_initializer='zeros', kernel_regularizer=None,
    recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, recurrent_constraint=None, bias_constraint=None,
    dropout=0.0, recurrent_dropout=0.0, return_sequences=False, return_state=False,
    go_backwards=False, stateful=False, unroll=False, **kwargs
)
```



RNN: Keras API

- 마지막 hidden state를 return하기 위해서는 'return_state=True'를 사용한다.
- tf.keras.layers.SimpleRNN(4, return_state=True)
  - Output, final_state = simple_rnn(inputs)

```
tf.keras.layers.SimpleRNN(
    units, activation='tanh', use_bias=True,
    kernel_initializer='glorot_uniform',
    recurrent_initializer='orthogonal',
    bias_initializer='zeros', kernel_regularizer=None,
    recurrent_regularizer=None, bias_regularizer=None, activity_regularizer=None,
    kernel_constraint=None, recurrent_constraint=None, bias_constraint=None,
    dropout=0.0, recurrent_dropout=0.0, return_sequences=False, return_state=False,
    go_backwards=False, stateful=False, unroll=False, **kwargs
)
```

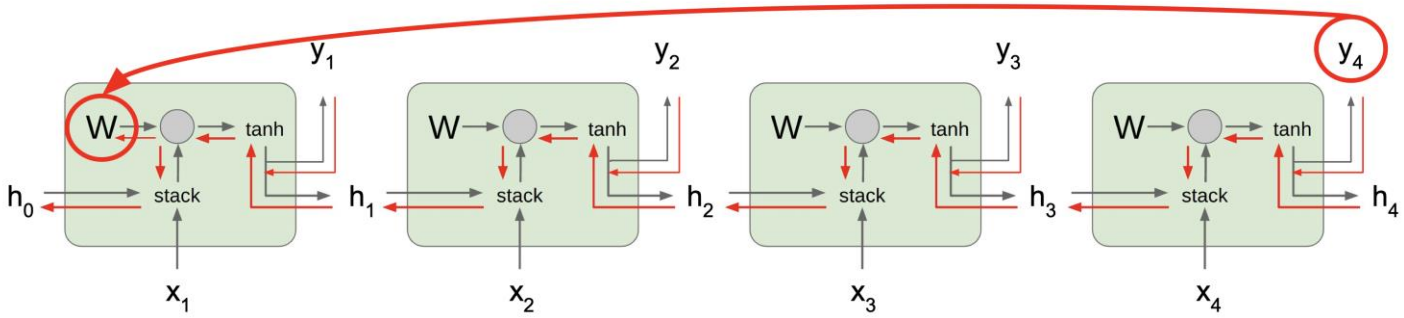


목차

- RNN
- RNN의 다양한 버전: LSTM, GRU

RNN의 문제점

- 같은 Weight를 반복적으로 곱하기 때문에 Gradient가 Vanising되거나 Explode되는 문제가 발생 !
  - 장기적인 Context의 학습이 힘듦. Sequence가 길어지더라도, 뒤에 정보를 학습할 수 있는 요소가 필요함 !



$$\frac{\partial L}{\partial W} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T} \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_1}{\partial W} = \frac{\partial L_T}{\partial h_T} \left(\prod_{t=2}^{T} \frac{\partial h_t}{\partial h_{t-1}} \right) \frac{\partial h_1}{\partial W}$$

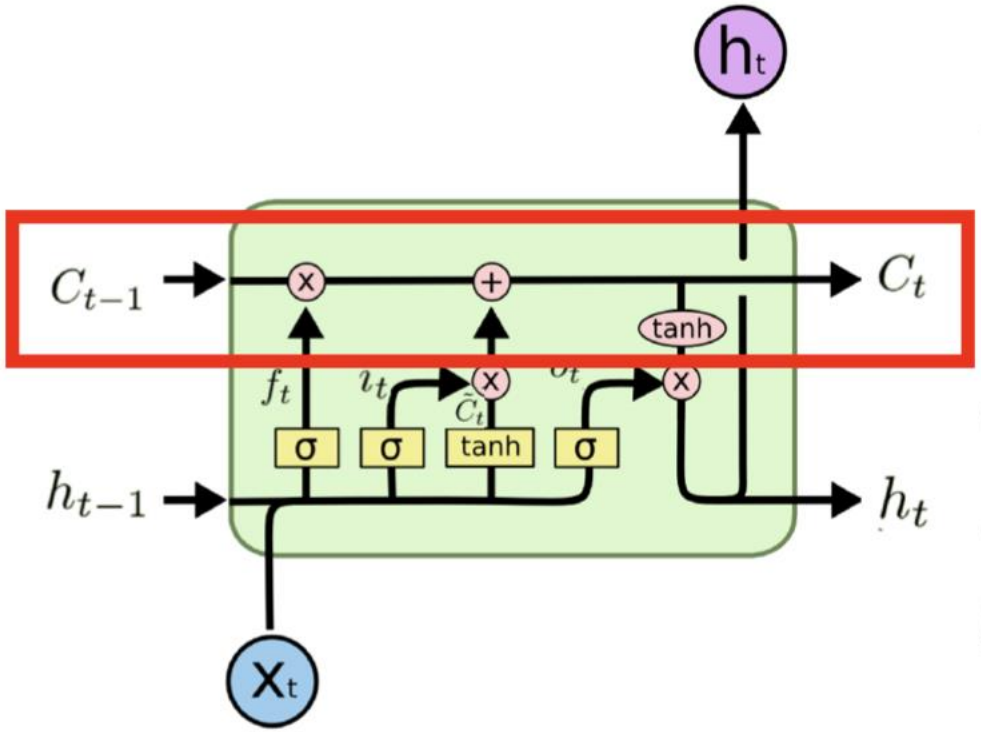
RNN의 문제점

- 'vanishing gradient' 문제로 인해 멀리 떨어진 'print her tickets' 와의 dependency를 학습하지 못해 'printer' 로 잘못 예측할 수 있다

LM task: *When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her* ___*tickets*___

LSTM & GRU

cell state



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

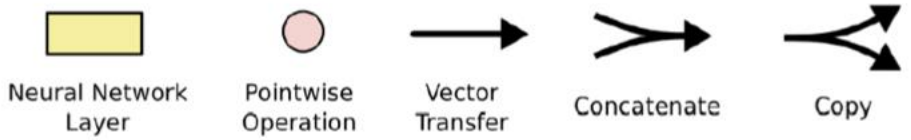
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

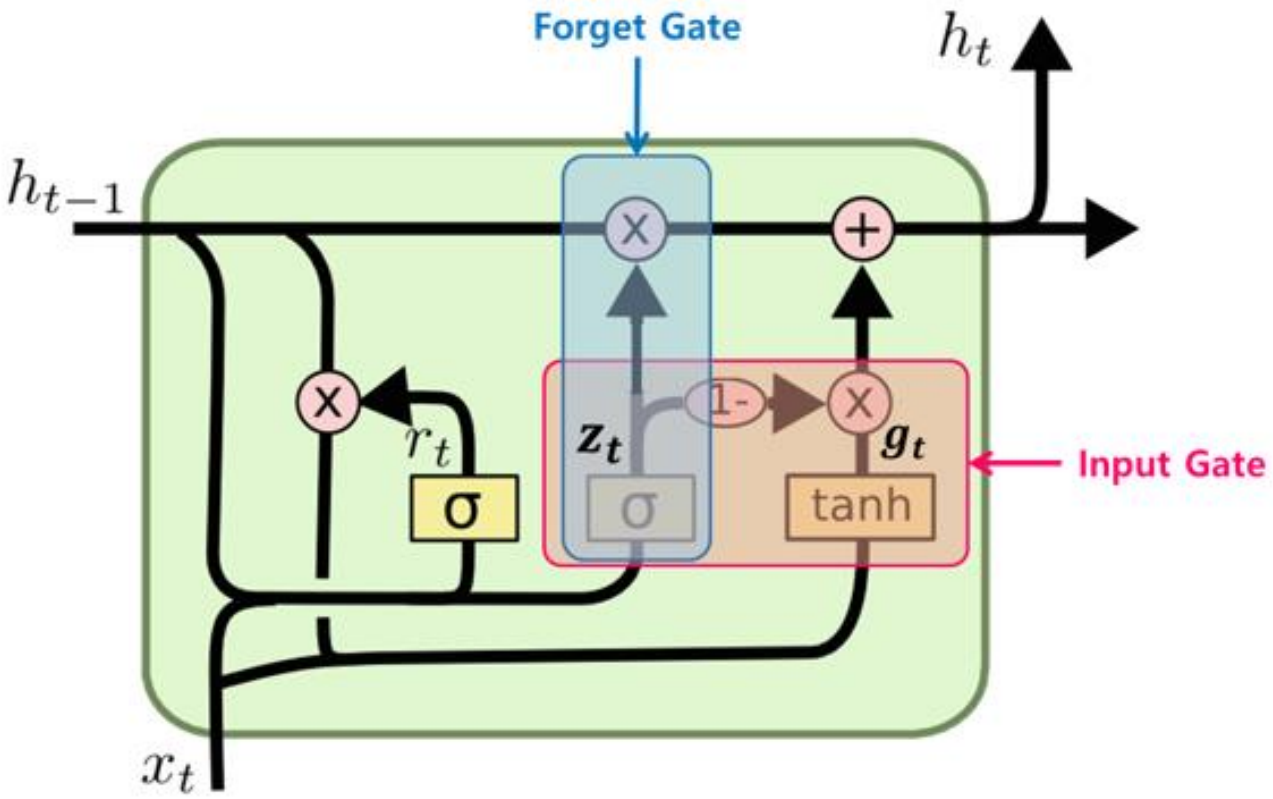
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

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



LSTM & GRU



이전 컨텍스트에서 어떤 정보를 쓸 지

$$r_t = \sigma(W_{xr}^T \cdot x_t + W_{hr}^T \cdot h_{t-1} + b_r)$$

이전 컨텍스트를 얼마나 보존할 지

$$z_t = \sigma(W_{xz}^T \cdot x_t + W_{hz}^T \cdot h_{t-1} + b_z)$$

그래서 결국 현재의 상황은?

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

이전 컨텍스트 + 현재의 정보까지 고려

$$h_t = z_t \bigotimes h_{t-1} + (1 - z_t) \bigotimes g_t$$

Example: Seq2Seq model