FastCampus Pytorch

Ch8. Recurrent Neural Networks

HARRY KIM

Lecture Content

- **강의 자료**
  - Books
    - Pattern Classification Second Edition [Duda, 2001]
    - Pattern Recognition And Machine Learning [Bishop, 2006]
  - Online
    - UVA DEEP LEARNING COURSE [University of Amsterdam, 2018]

Sequential Data

RNN

LSTM

GRU

Applications

1. Sequential Data

Sequential Data

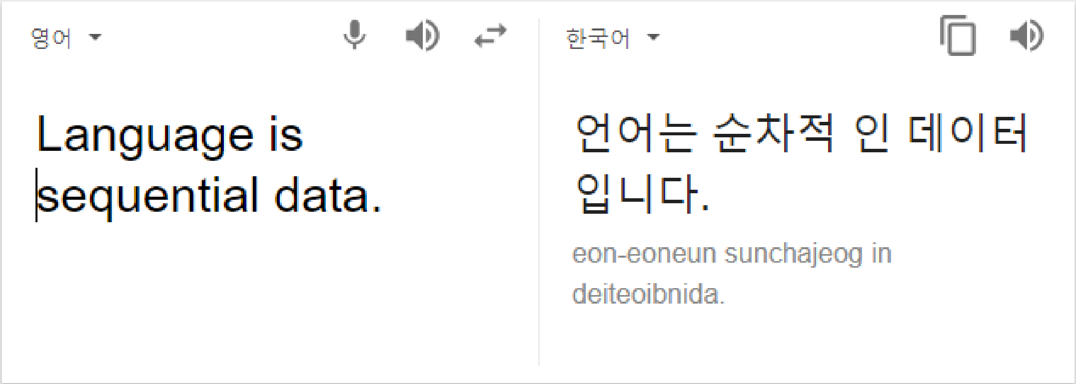
- ## Sequential Data
  - 우리가 살고 있는 세상?
  - 공간만이 있는 것이 아니다!
  - **시간(Time)**도 존재

Sequential Data

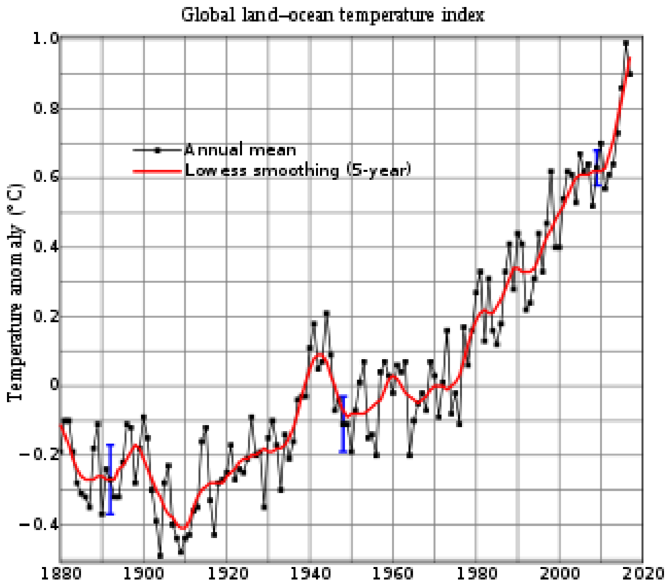
- **Sequential Data**
  - 다음 데이터가 이전 데이터에 영향을 받는 것
  - $\Pr(x_{t+1}) = \Pr(x_{t+1}|x_t)$

Sequential Data

- ## Sequential Data
  - 언어
  - 주식 가격
  - 날씨
  - …



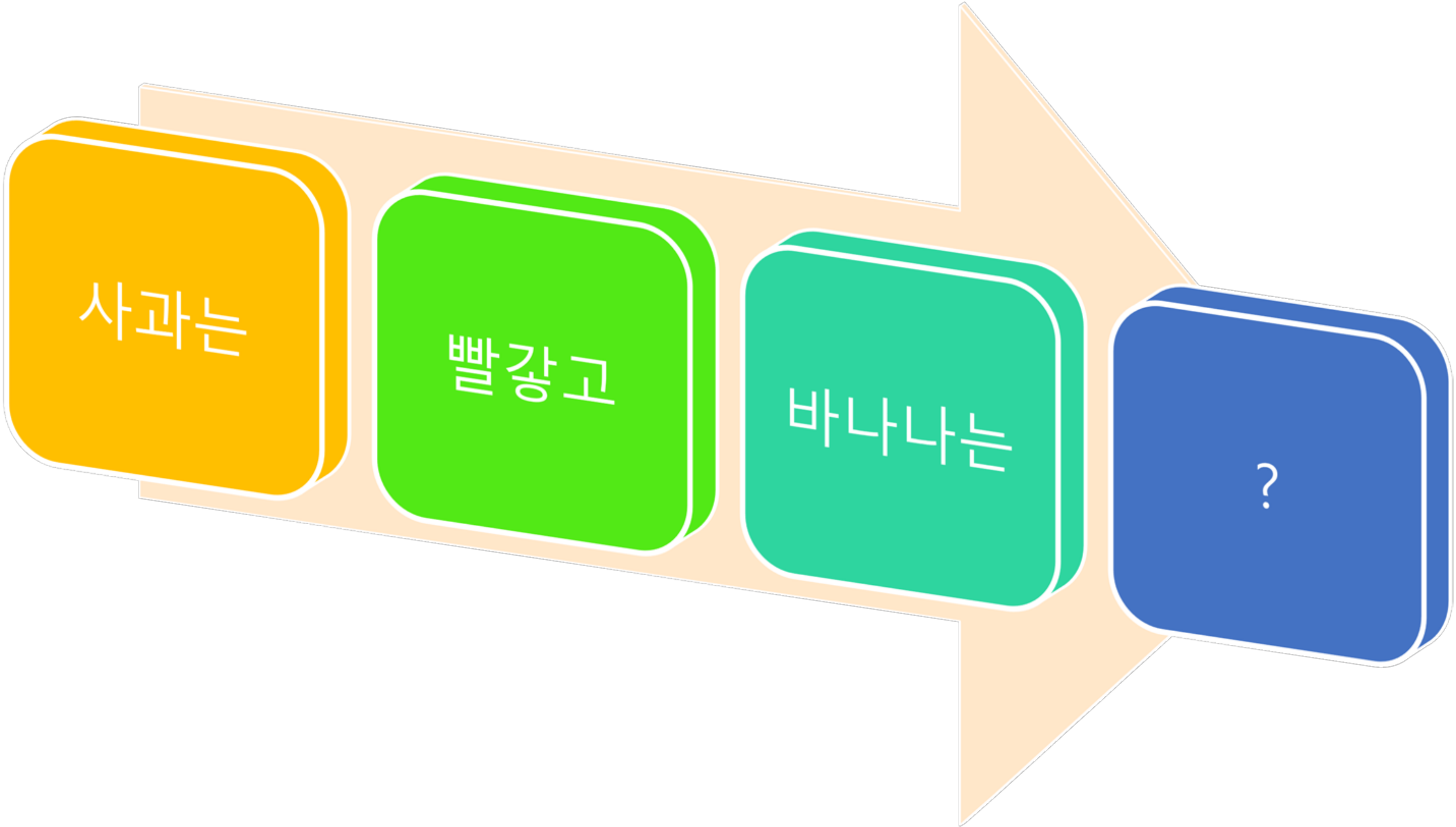
http://finance.daum.net/item/main.daum?code=035720



https://en.wikipedia.org/wiki/Global_warming

Sequential Data

- Sequential Data



Sequential Data

Sequential Data

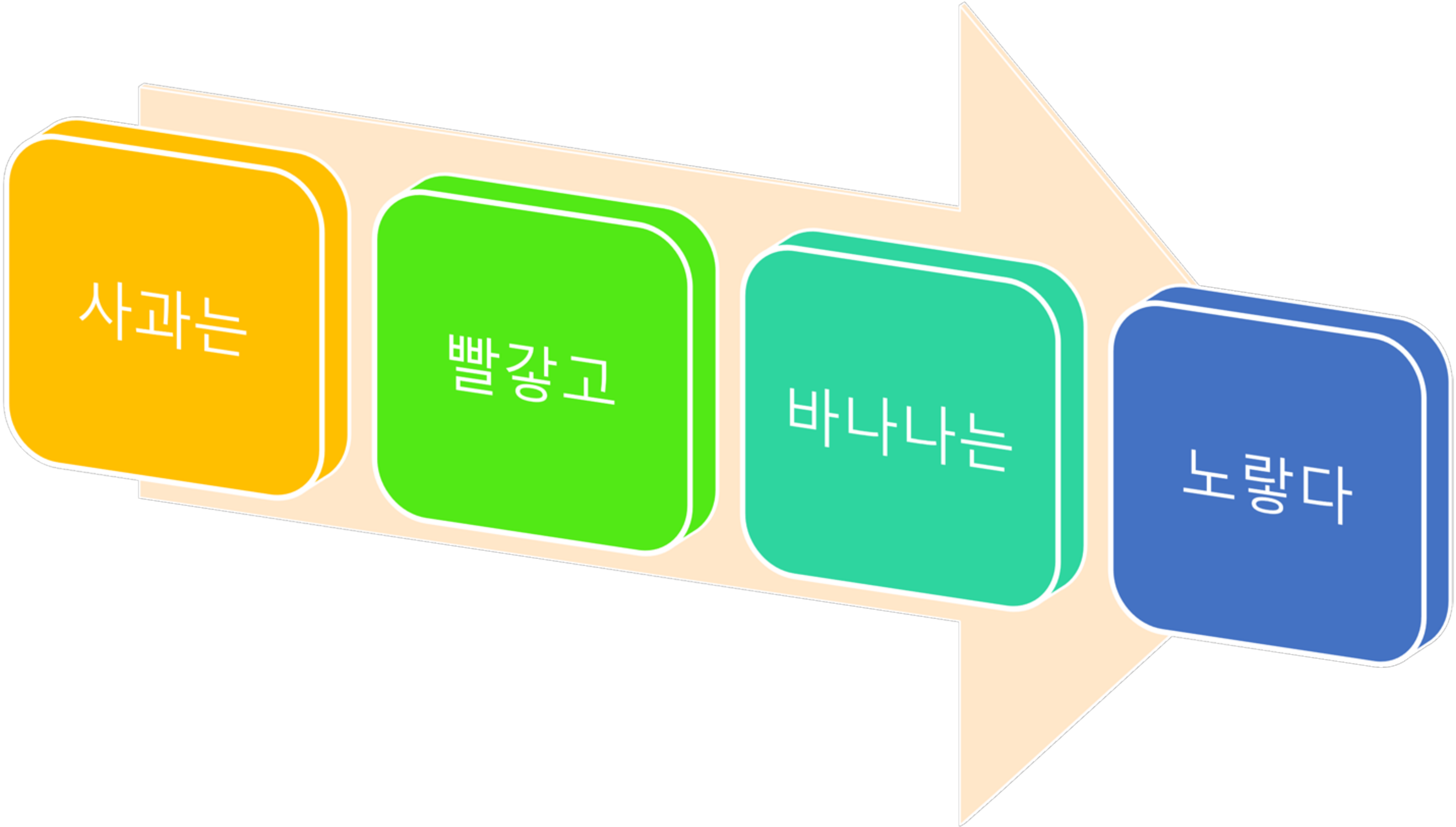
RNN

LSTM

GRU

Applications

- Sequential Data



Sequential Data

Sequential Data

- RNN
- LSTM
- GRU
- Applications

- **Sequential Data**

Sequential Data

Sequential Data

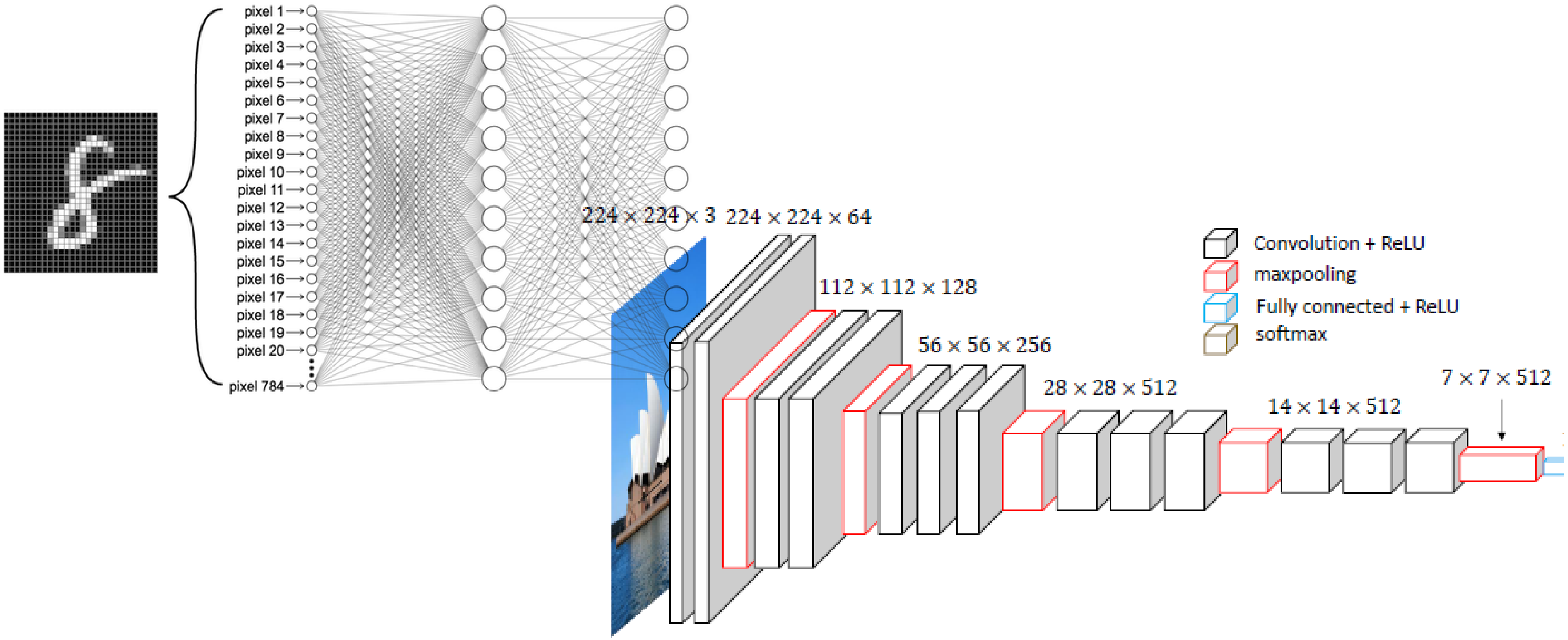
RNN

LSTM

GRU

Applications

- **Sequential Data**
  - NN/CNN에서는..?
  - 시간을 고려할만한 요소가 없음



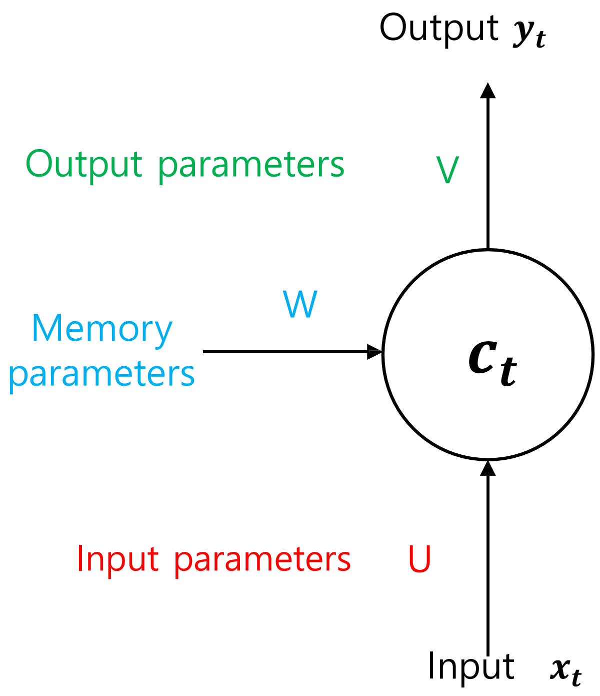
2. RNN

RNN

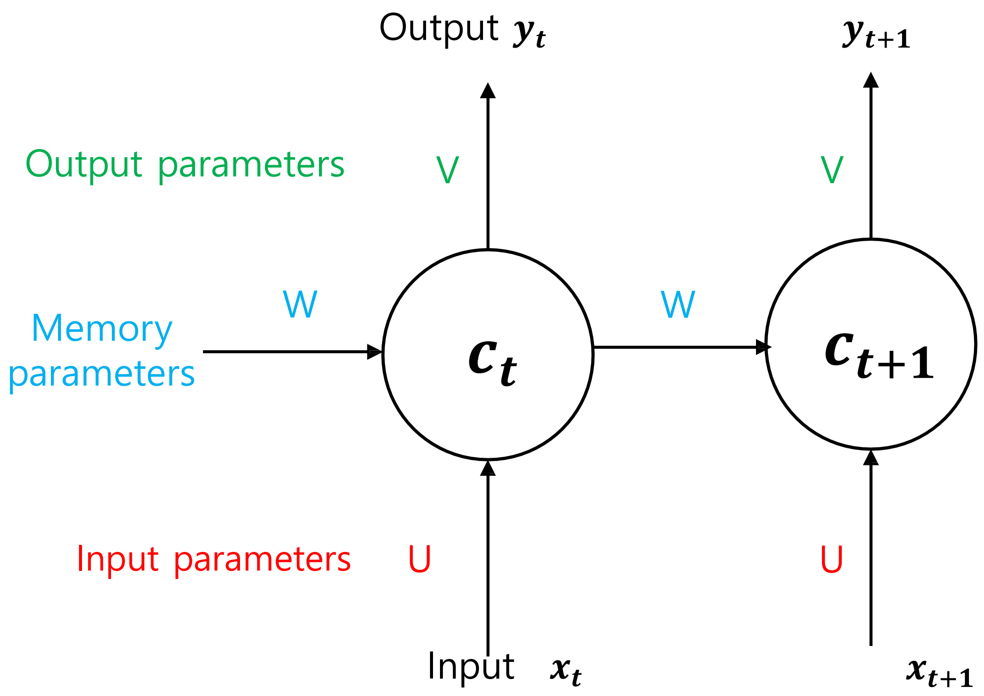
- **RNN(Recurrent Neural Network)**
  - 기존 NN에서의 순전파
  - $x_{t+1} = h(x_t; \theta)$
  - 이전 결과들을 반영하기 위해서는 기억(Memory, c_t)이 필요
  - 또한 새로운 데이터(x_{t+1})로 새로운 결과를 도출하는 과정에서 또 새로운 기억(c_{t+1})이 생성
  - $c_{t+1} = h(x_{t+1}, c_t; \theta)$
  - $c_{t+1} = h(x_{t+1}, h(x_t, h(x_{t-1}, \ldots, h(x_1, c_0; \theta); \theta); \theta); \theta)$
  - RNN에서는 θ가 공유

RNN

- **RNN(Recurrent Neural Network)**
  - Input Parameters : U
  - Memory Parameters : W
  - Output Parameters : V

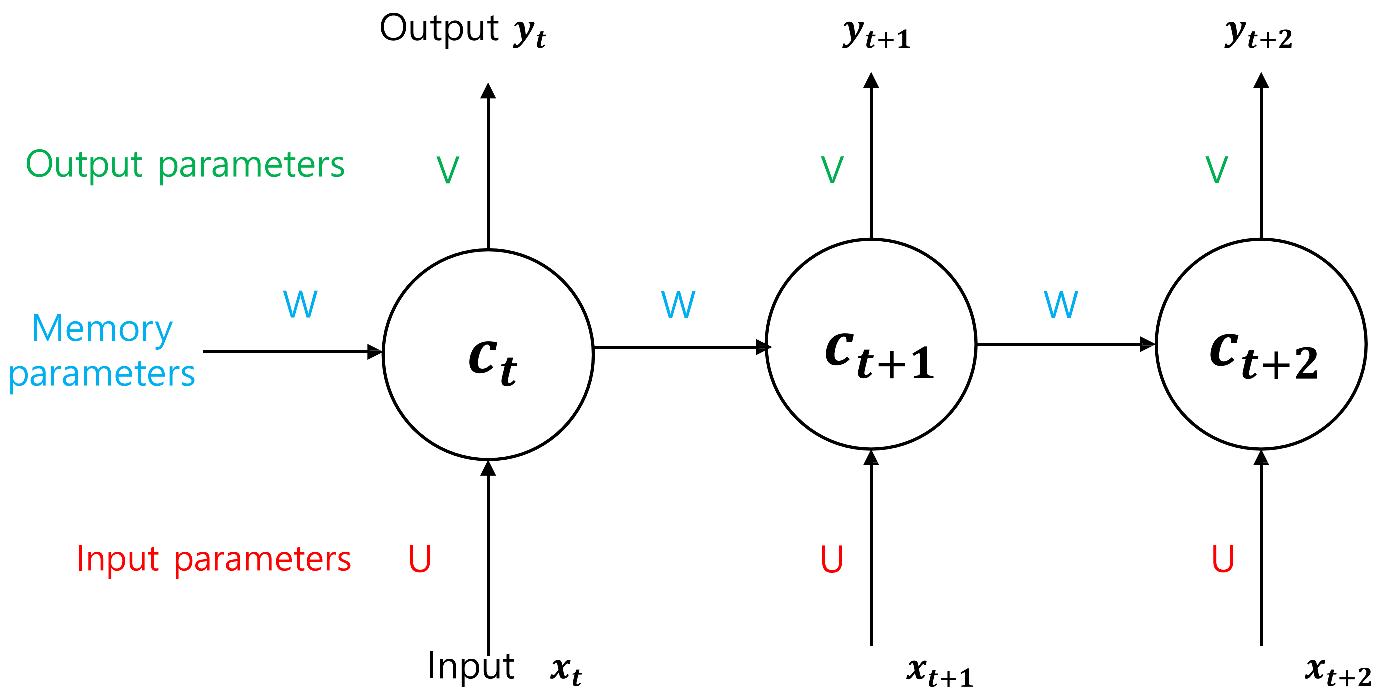


- **RNN(Recurrent Neural Network)**
  - Input Parameters : U
  - Memory Parameters : W
  - Output Parameters : V

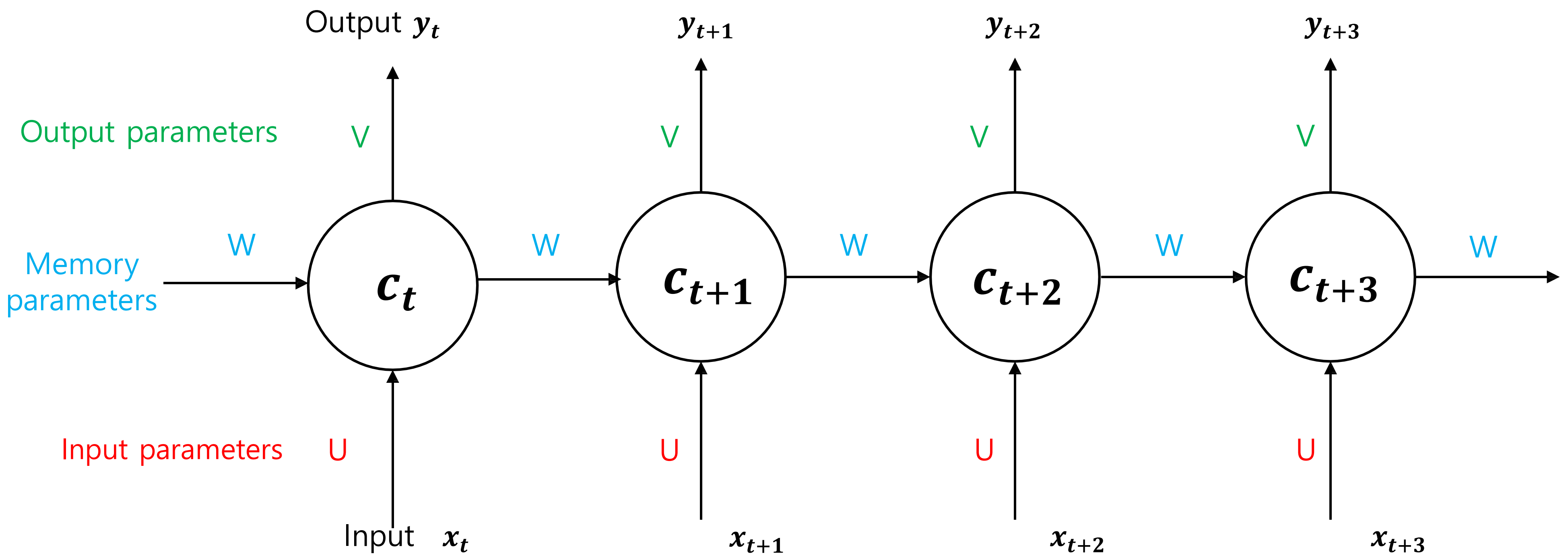


Sequential Data
RNN
LSTM
GRU
Applications

- **RNN(Recurrent Neural Network)**
  - Input Parameters : U
  - Memory Parameters : W
  - Output Parameters : V

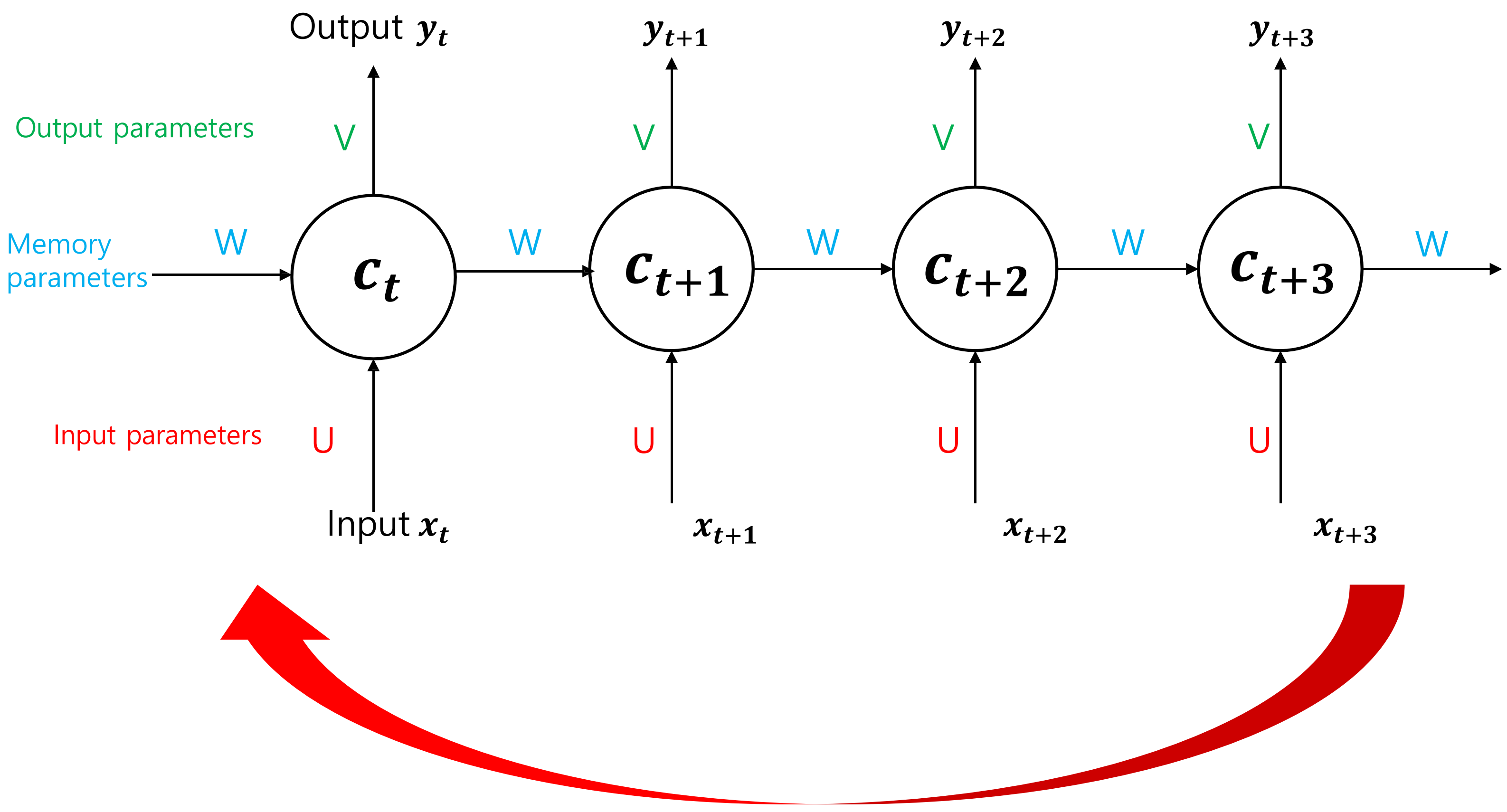


- **RNN(Recurrent Neural Network)**
  - Input Parameters : U
  - Memory Parameters : W
  - Output Parameters : V



RNN

- RNN(Recurrent Neural Network)

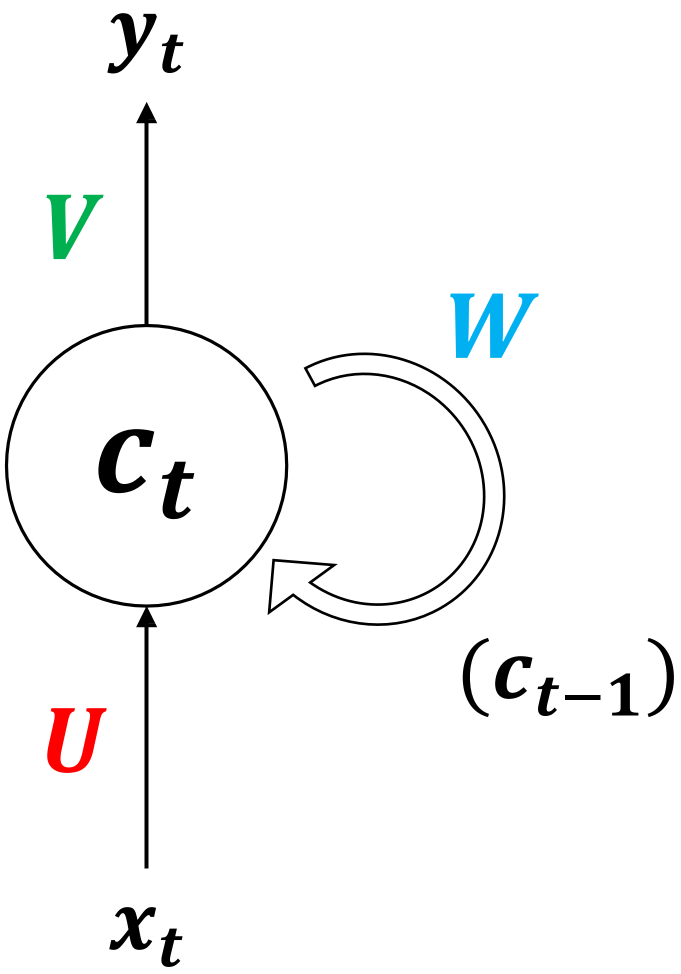


Rolled

RNN

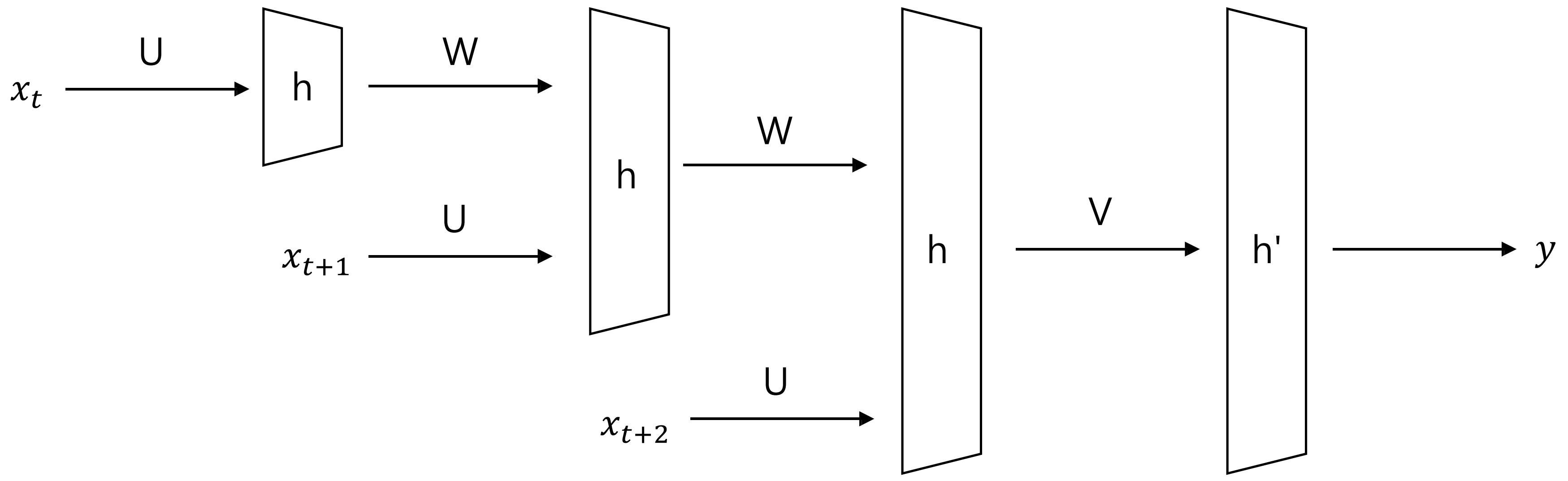
- RNN(Recurrent Neural Network)
  - Input Parameters : U
  - Memory Parameters : W
  - Output Parameters : V

- $c_t = h(Ux_t + Wc_{t-1})$
- $y_t = h'(Vc_t)$



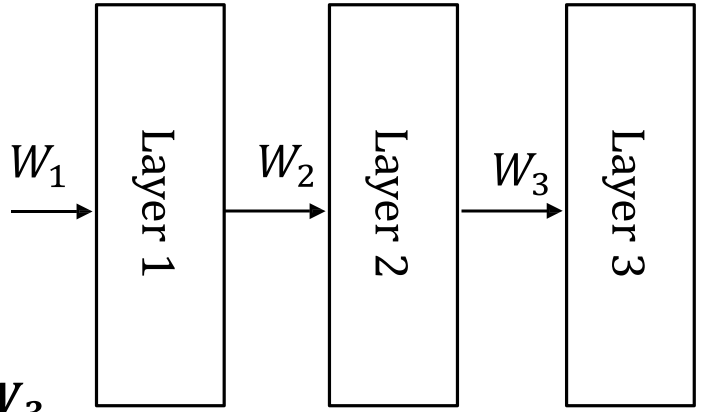
RNN

- **RNN(Recurrent Neural Network)**
  - $c_t = h(Ux_t)$
  - $c_{t+1} = h(Ux_{t+1} + Wc_t) = h(Ux_{t+1} + W\, h(Ux_t))$
  - $c_{t+2} = h(Ux_{t+2} + Wc_{t+1}) = h(Ux_{t+2} + W\, h(Ux_{t+1} + W\, h(Ux_t)))$
  - $y = h'(Vc_{t+2}) = h'(V\, h(Ux_{t+2} + W\, h(Ux_{t+1} + W\, h(Ux_t))))$

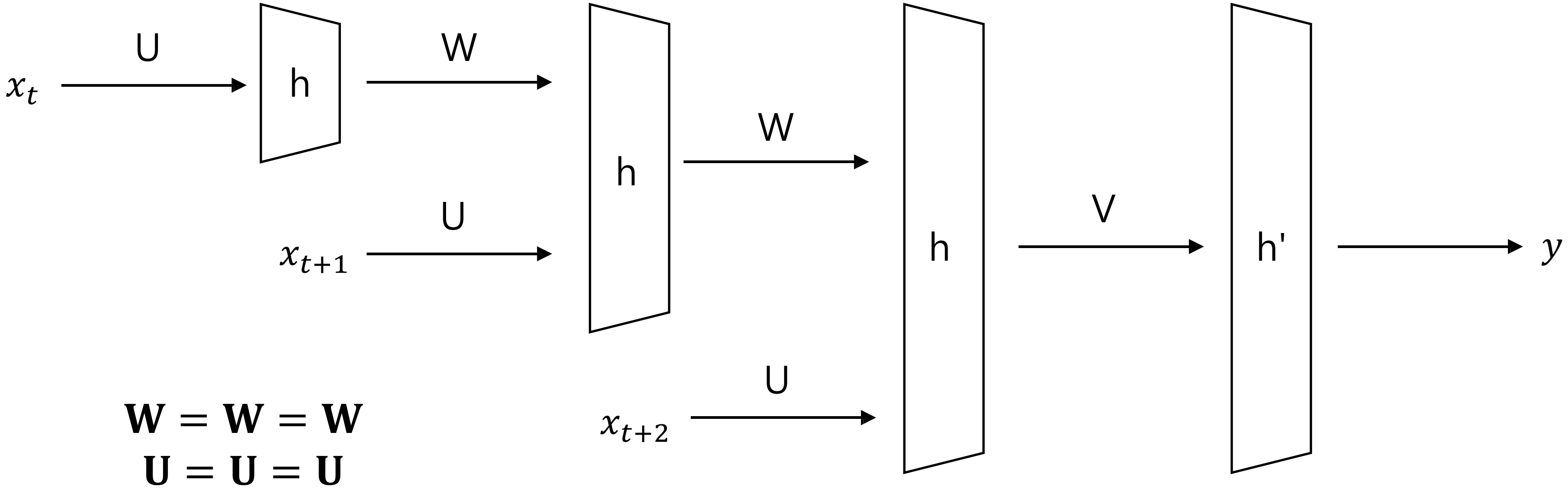


RNN

- **RNN(Recurrent Neural Network)**
  - NN과 어떻게 다른가?
    - Weight이 공유된다는 점이 가장 다름
    - Weight이 공유되지 않는다면 서로 같게 구성 가능



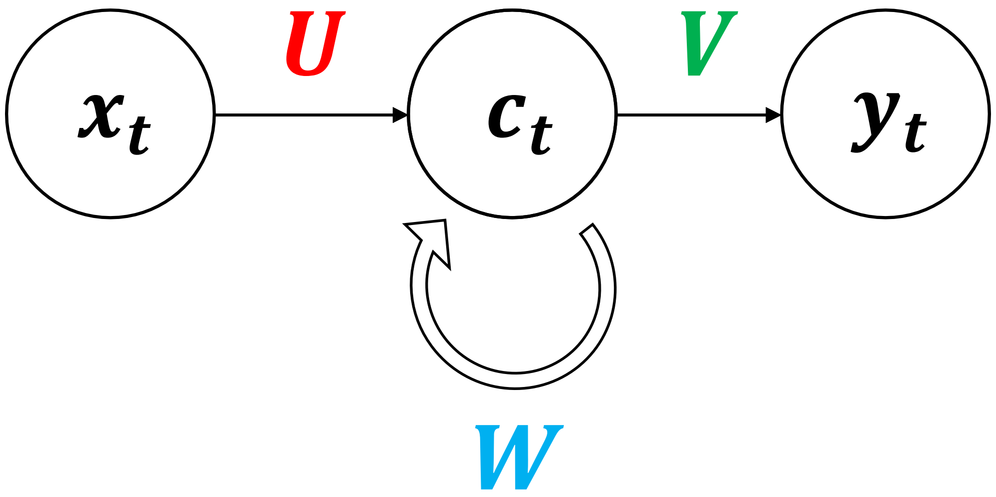
$$\boldsymbol{W_1 \neq W_2 \neq W_3}$$



$$\mathbf{W} = \mathbf{W} = \mathbf{W}$$
$$\mathbf{U} = \mathbf{U} = \mathbf{U}$$

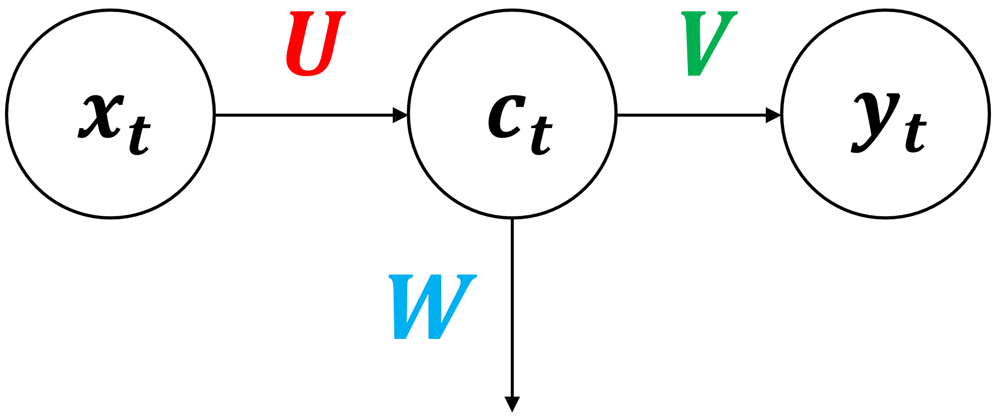
RNN

- RNN Training Process - Forward



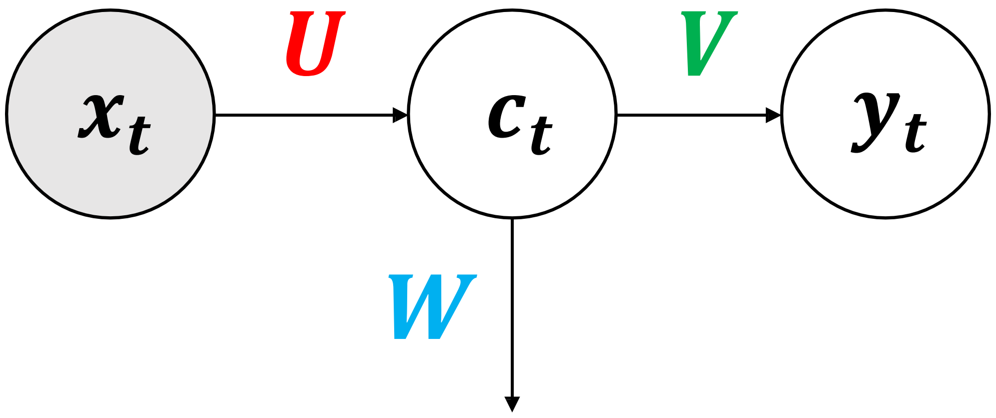
RNN

- RNN Training Process - Forward



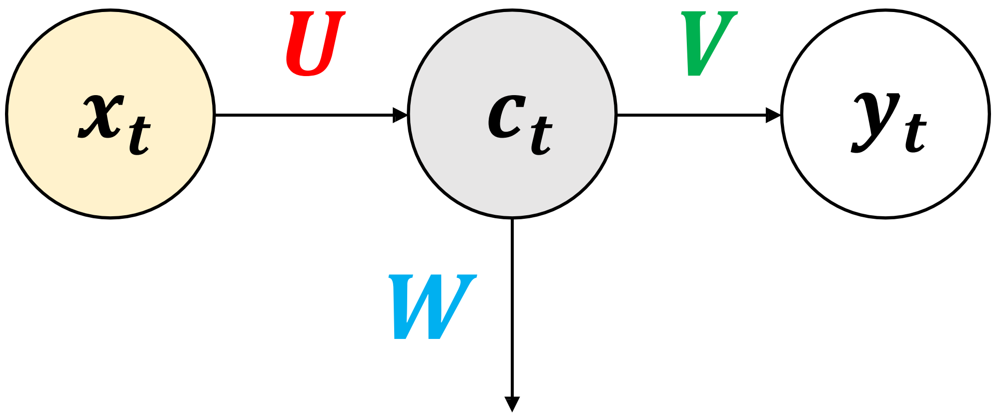
RNN

- RNN Training Process - Forward

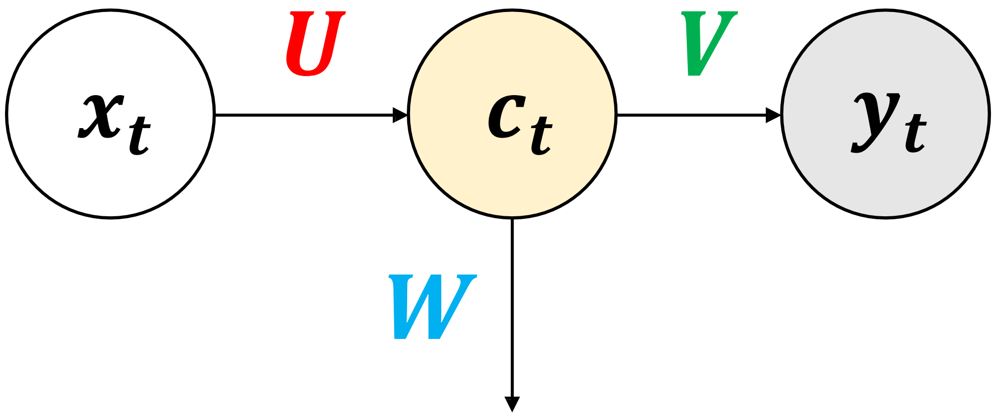


RNN

- RNN Training Process - Forward

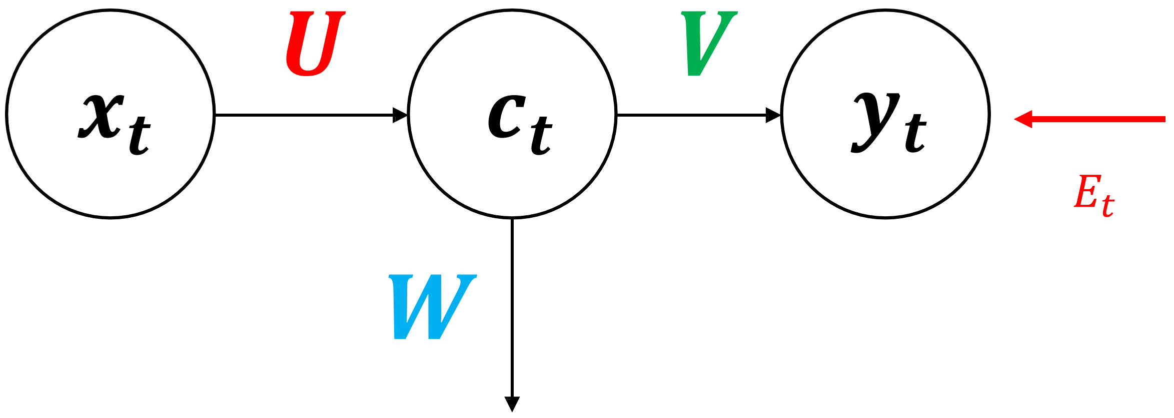


- RNN Training Process - Forward



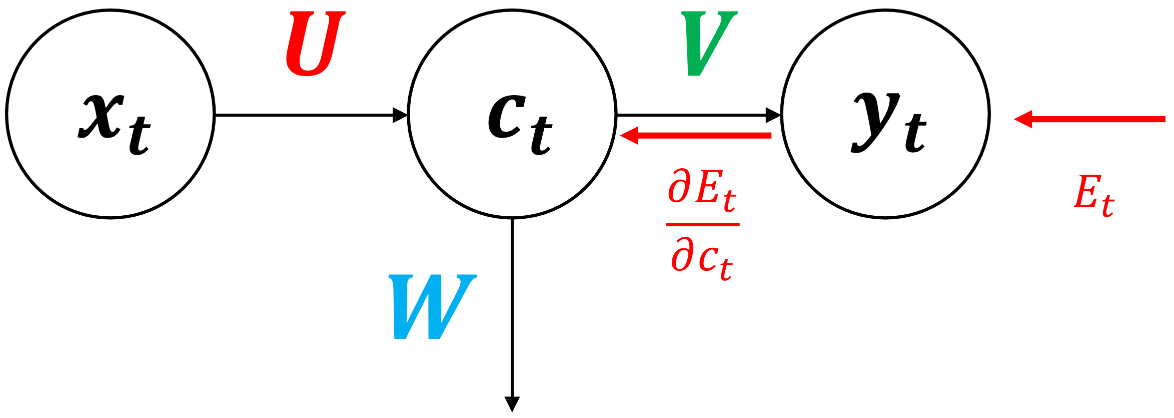
RNN

- RNN Training Process - Backward



RNN

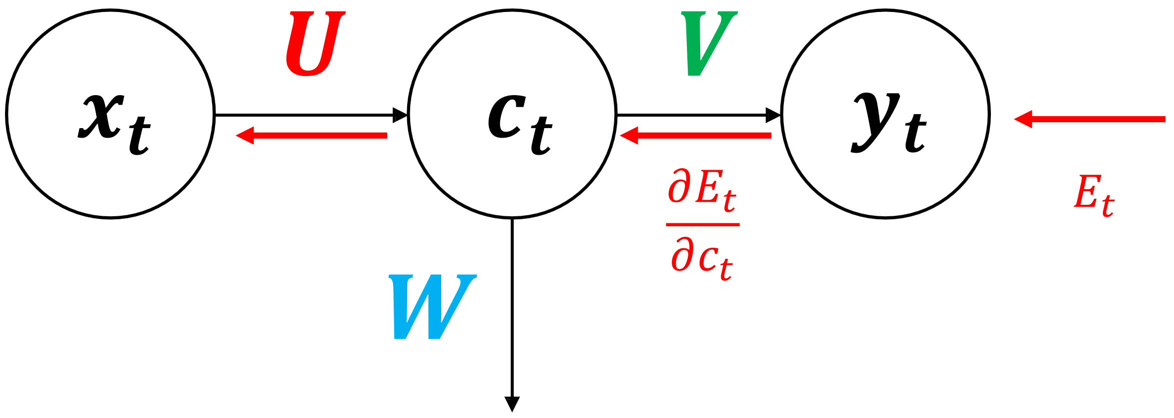
- **RNN Training Process - Backward**



RNN

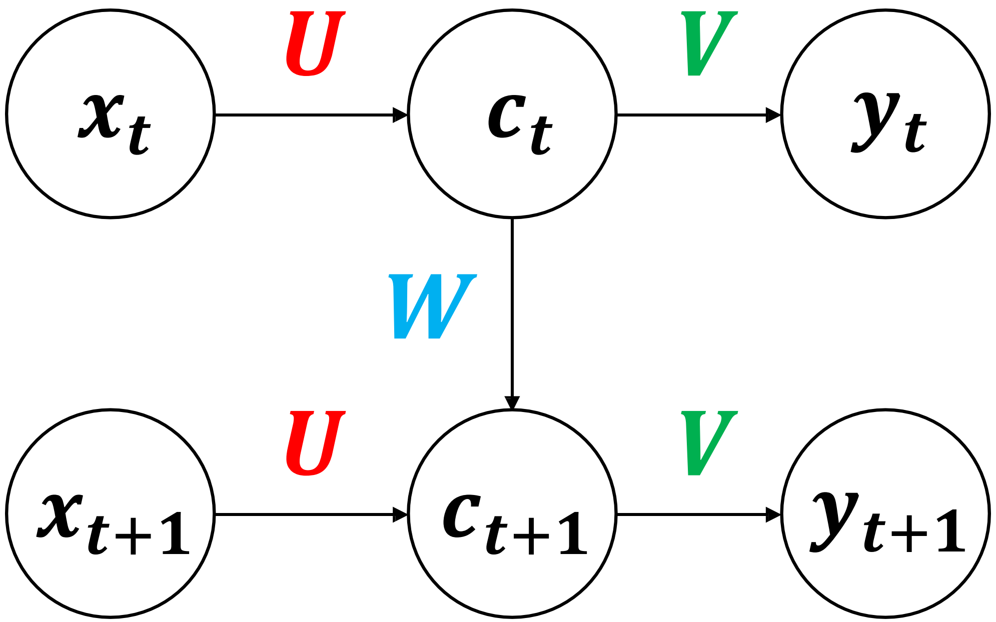
Sequential Data
RNN
LSTM
GRU
Applications

- RNN Training Process – Backward



RNN

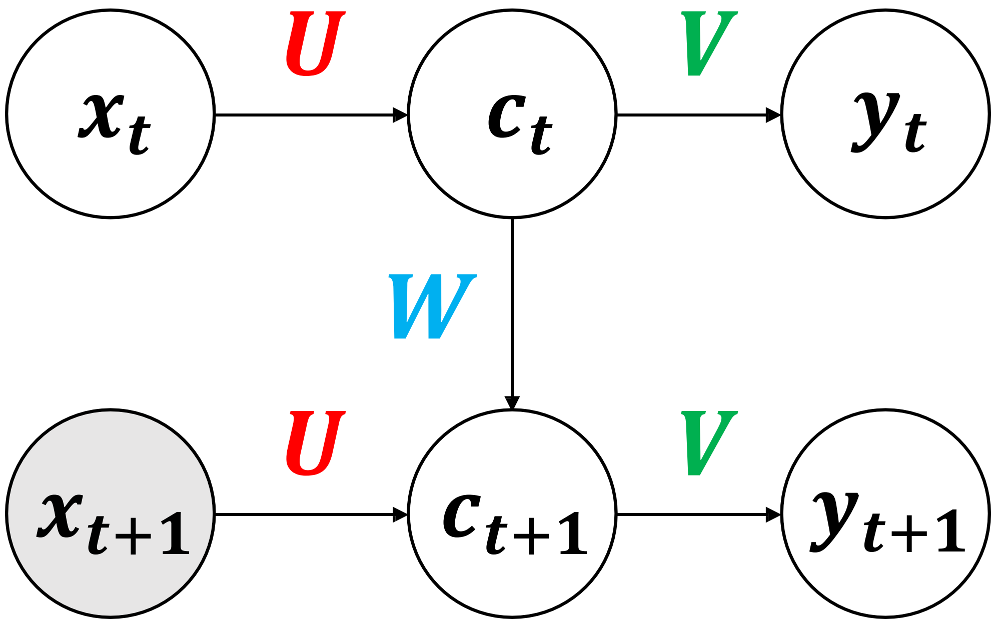
- RNN Training Process - Forward



Fast campus

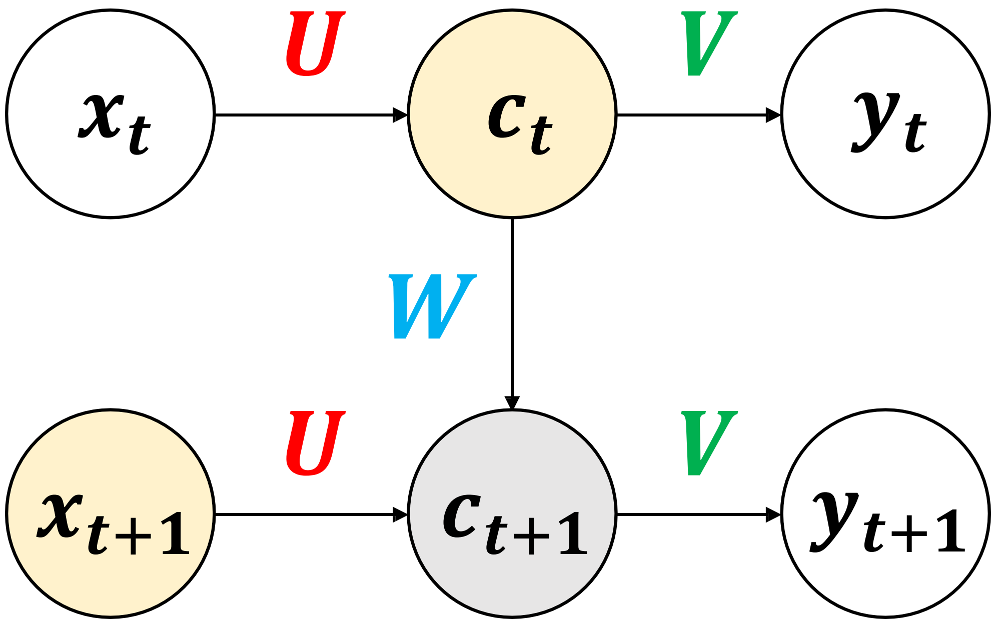
RNN

- RNN Training Process - Forward



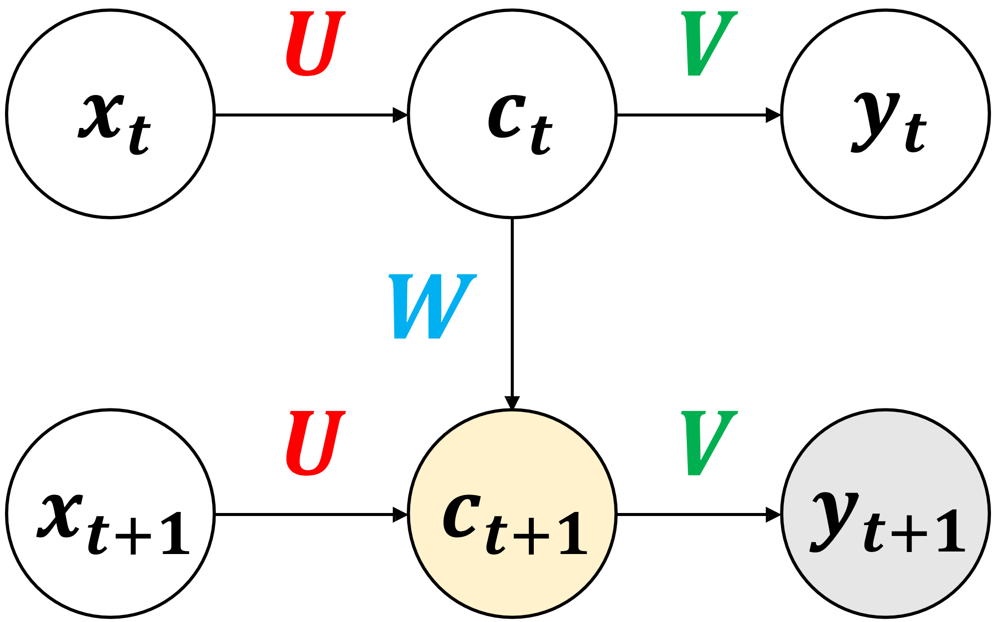
RNN

- **RNN Training Process - Forward**



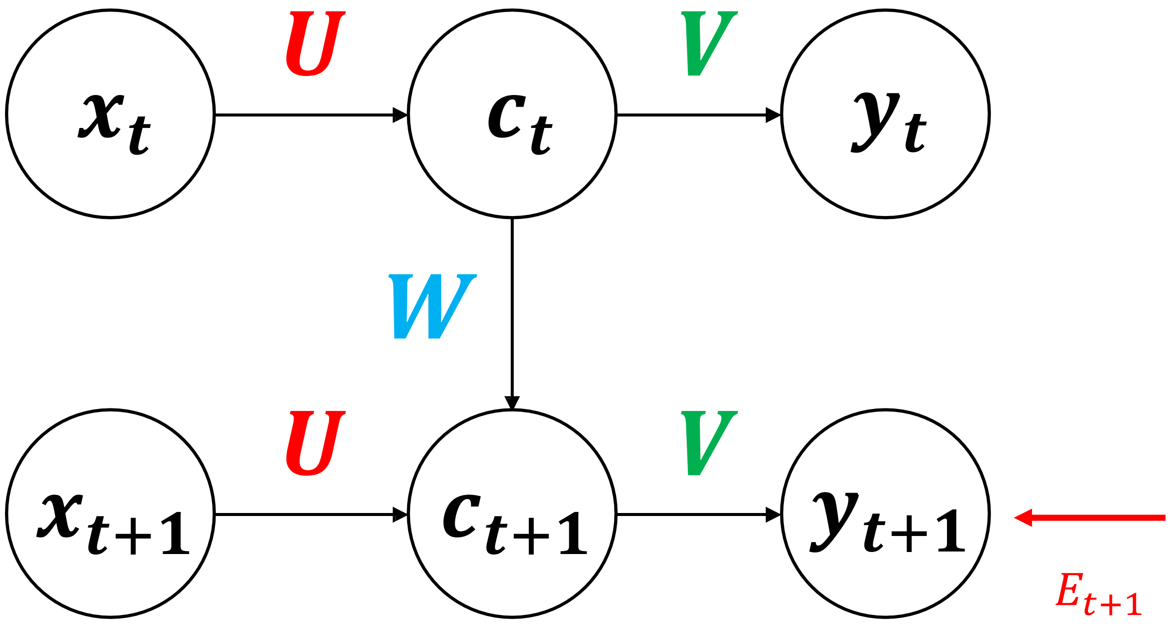
RNN

- RNN Training Process - Forward



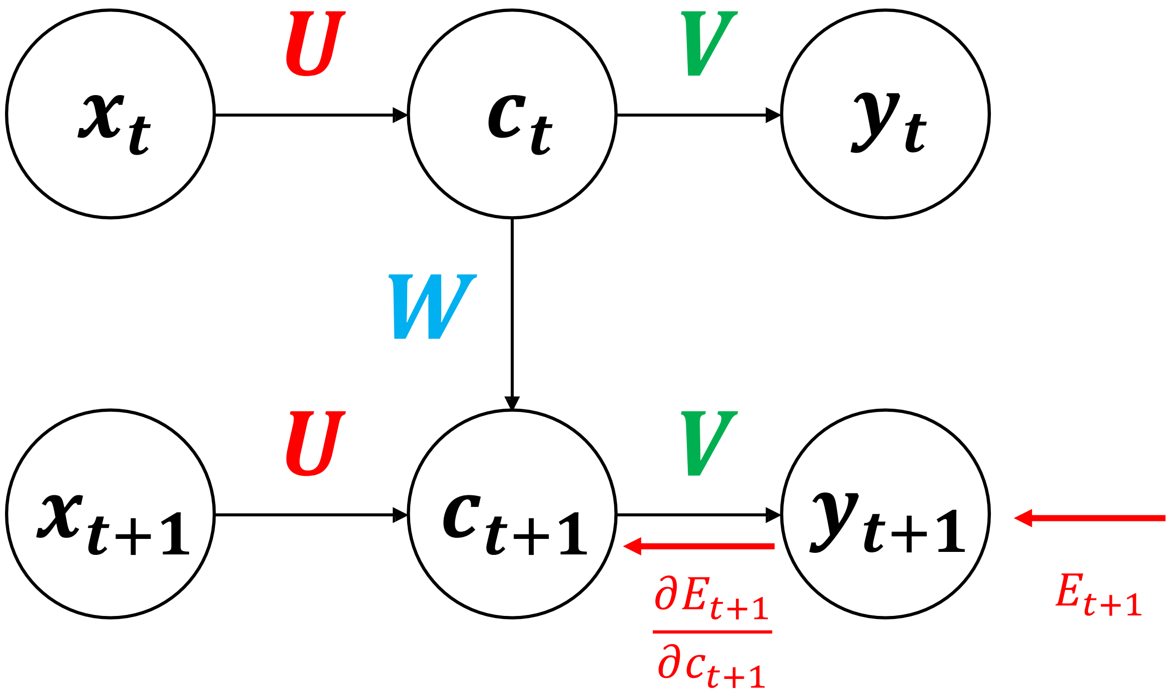
RNN

- RNN Training Process - Backward (Backpropagation Through Time)



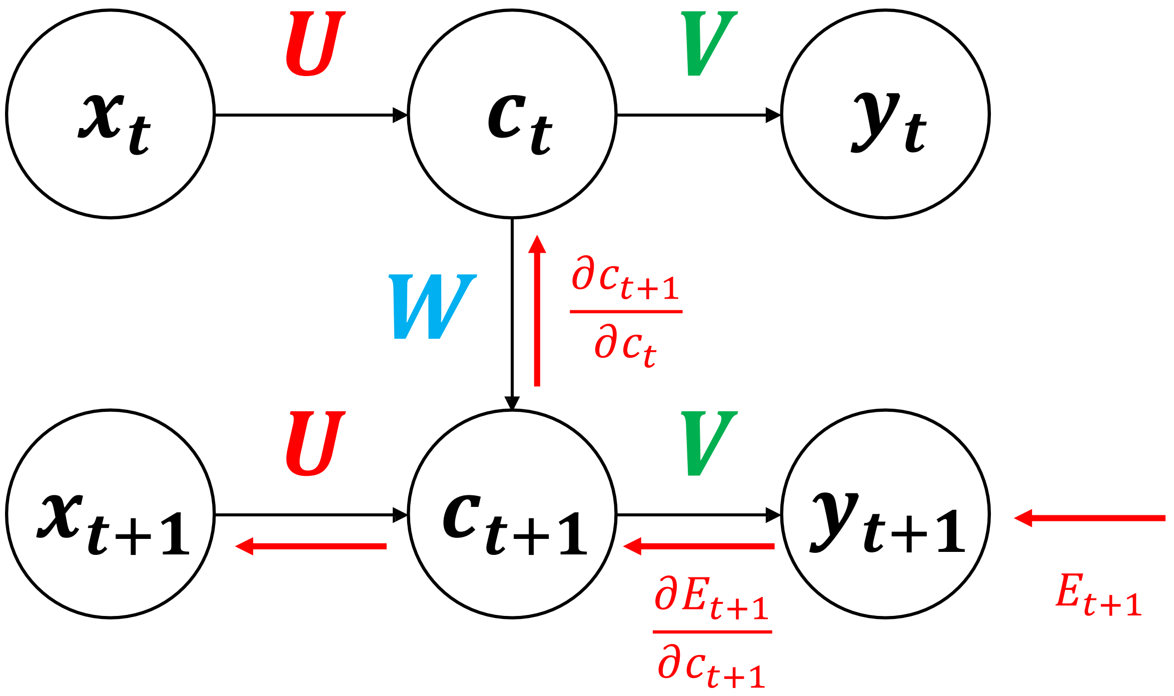
RNN

- RNN Training Process – Backward (Backpropagation Through Time)



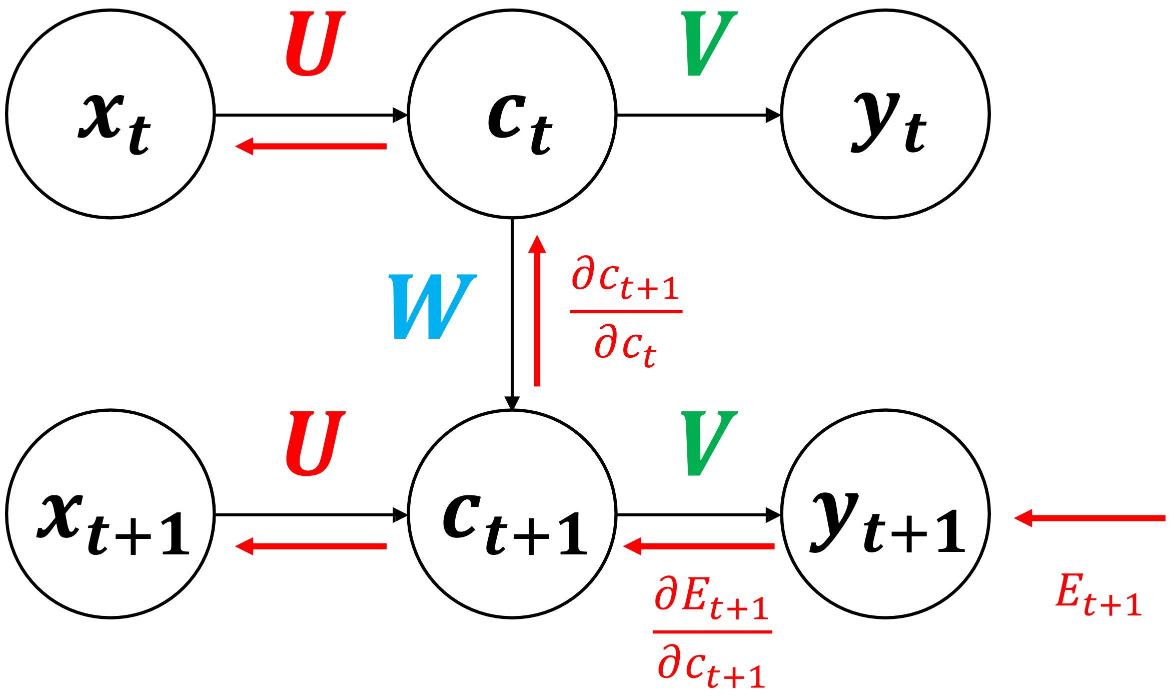
RNN

- RNN Training Process – Backward **(Backpropagation Through Time)**



RNN

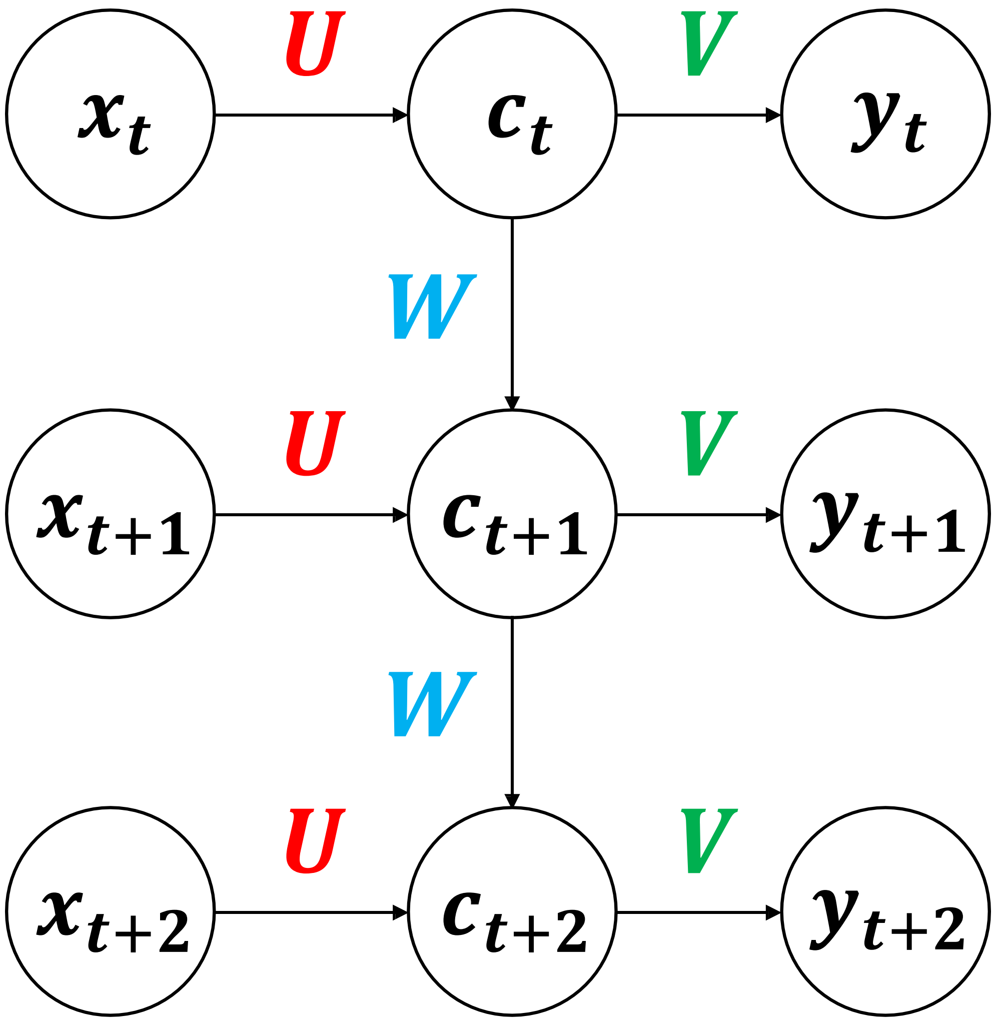
- RNN Training Process – Backward (Backpropagation Through Time)



$$\frac{\partial E_{t+1}}{\partial W} = \frac{\partial E_{t+1}}{\partial c_{t+1}} * \frac{\partial c_{t+1}}{\partial W}$$

RNN

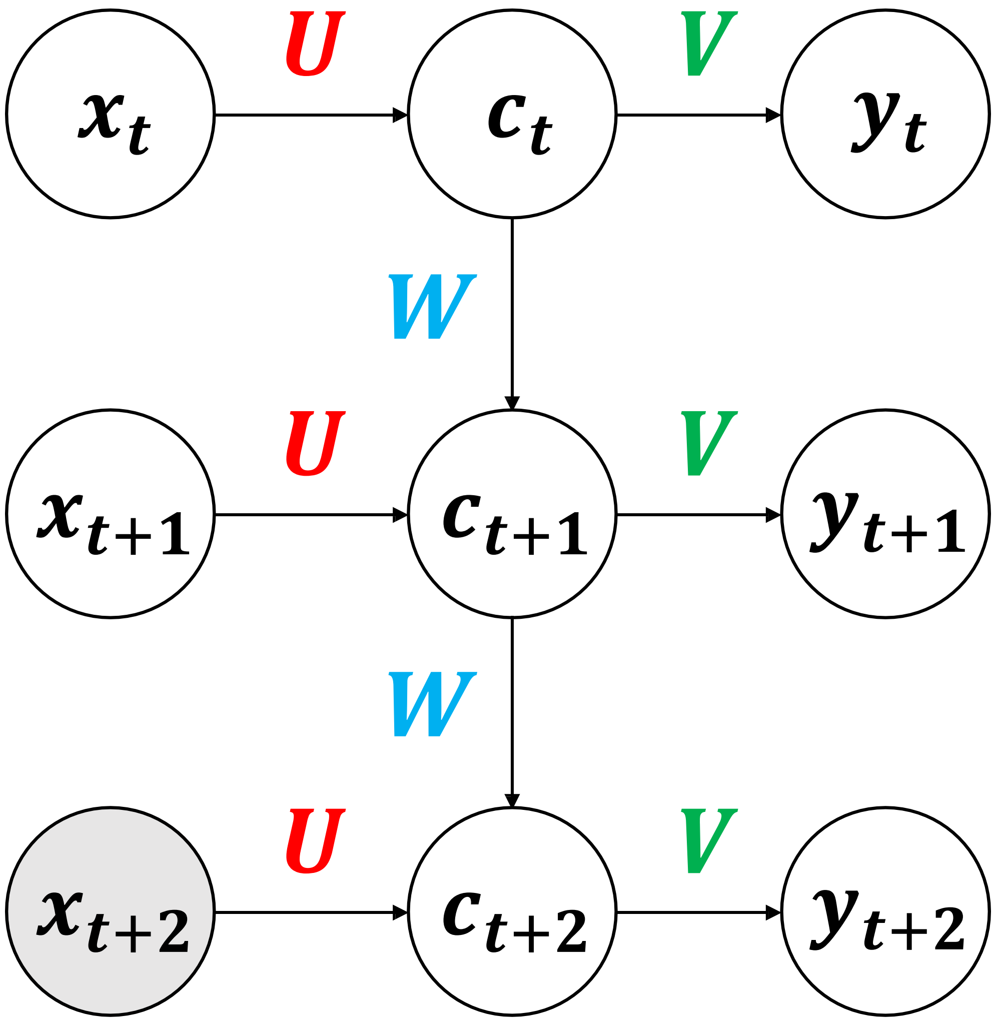
- RNN Training Process - Forward



Fast campus

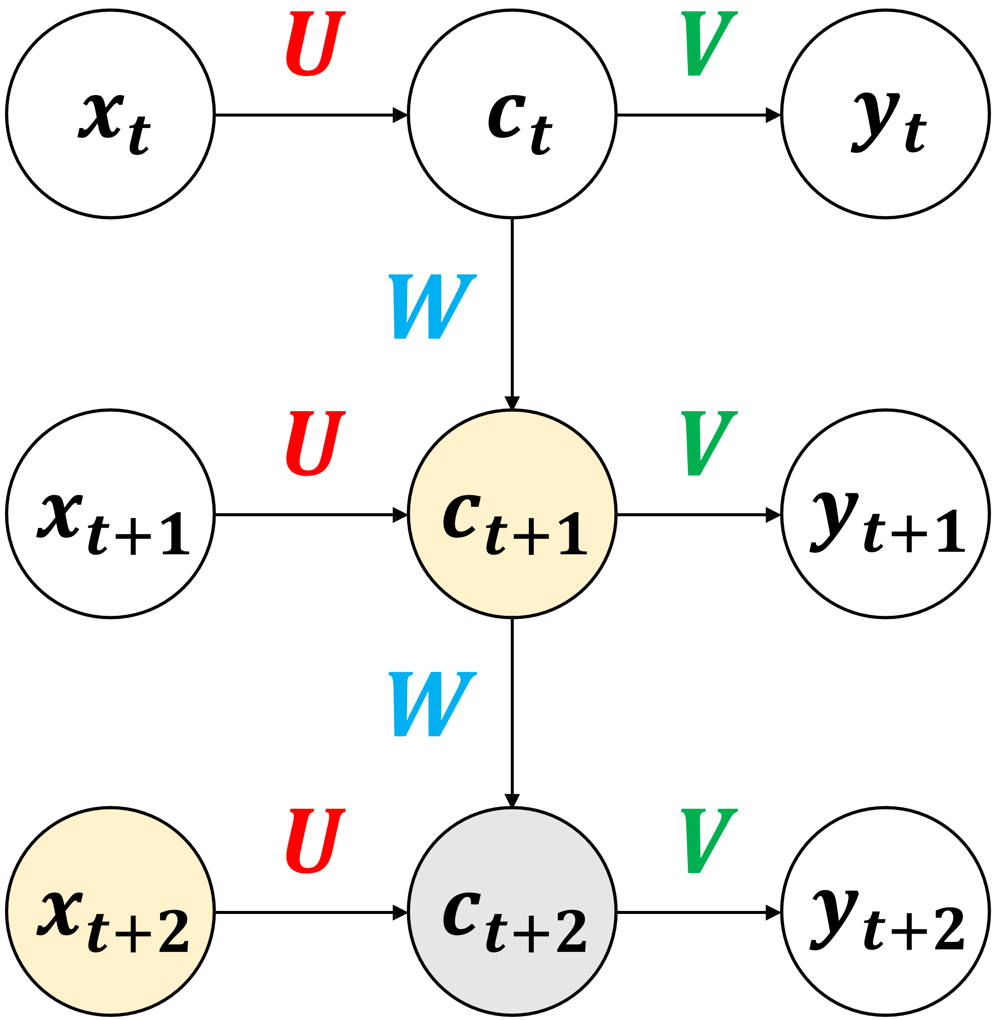
RNN

- RNN Training Process - Forward



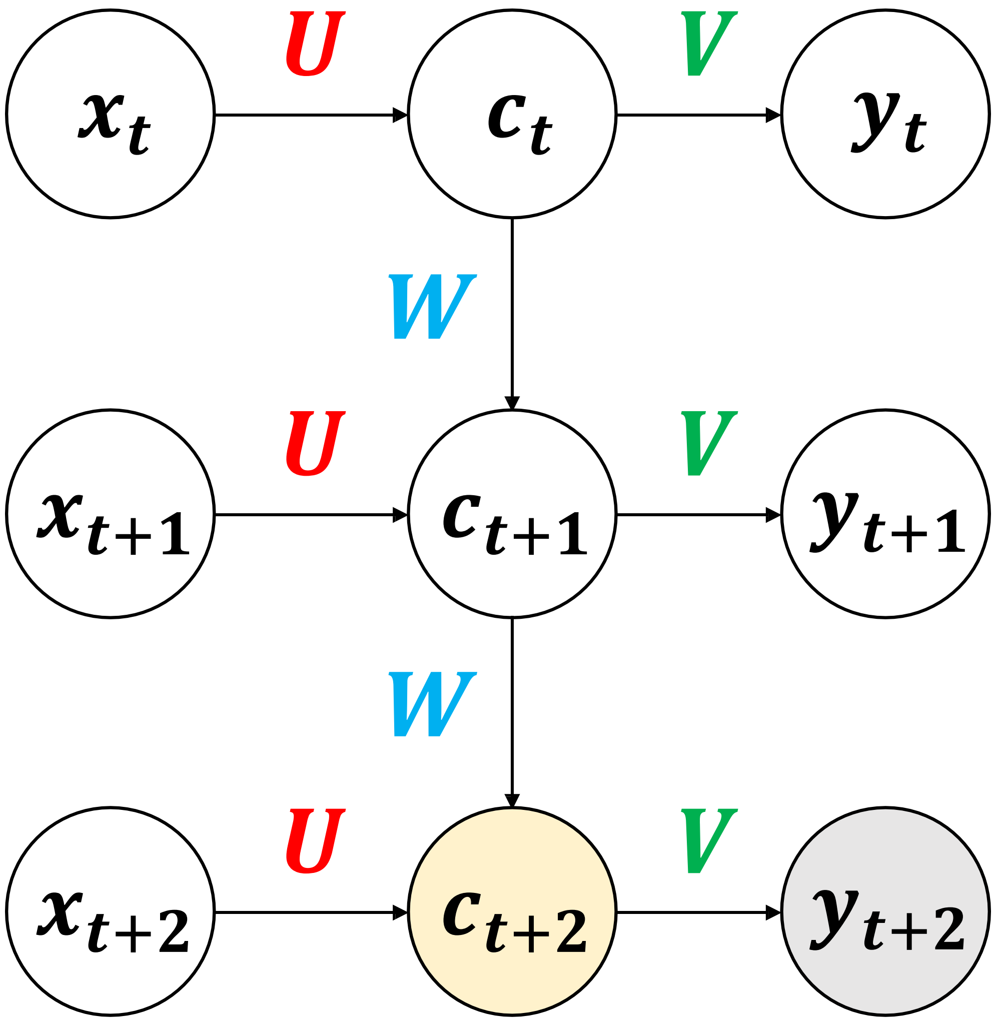
RNN

- RNN Training Process - Forward



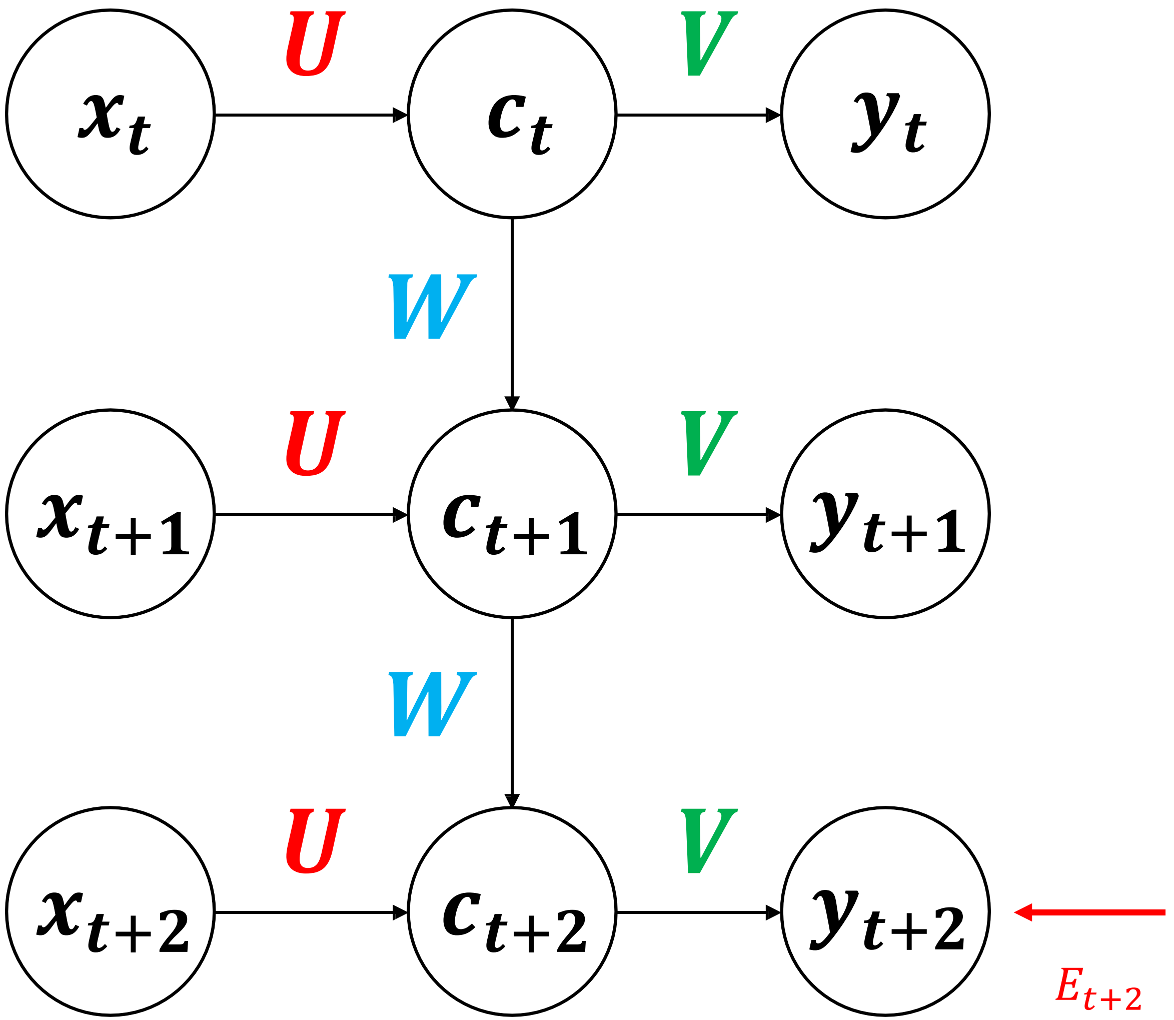
RNN

- RNN Training Process - Forward

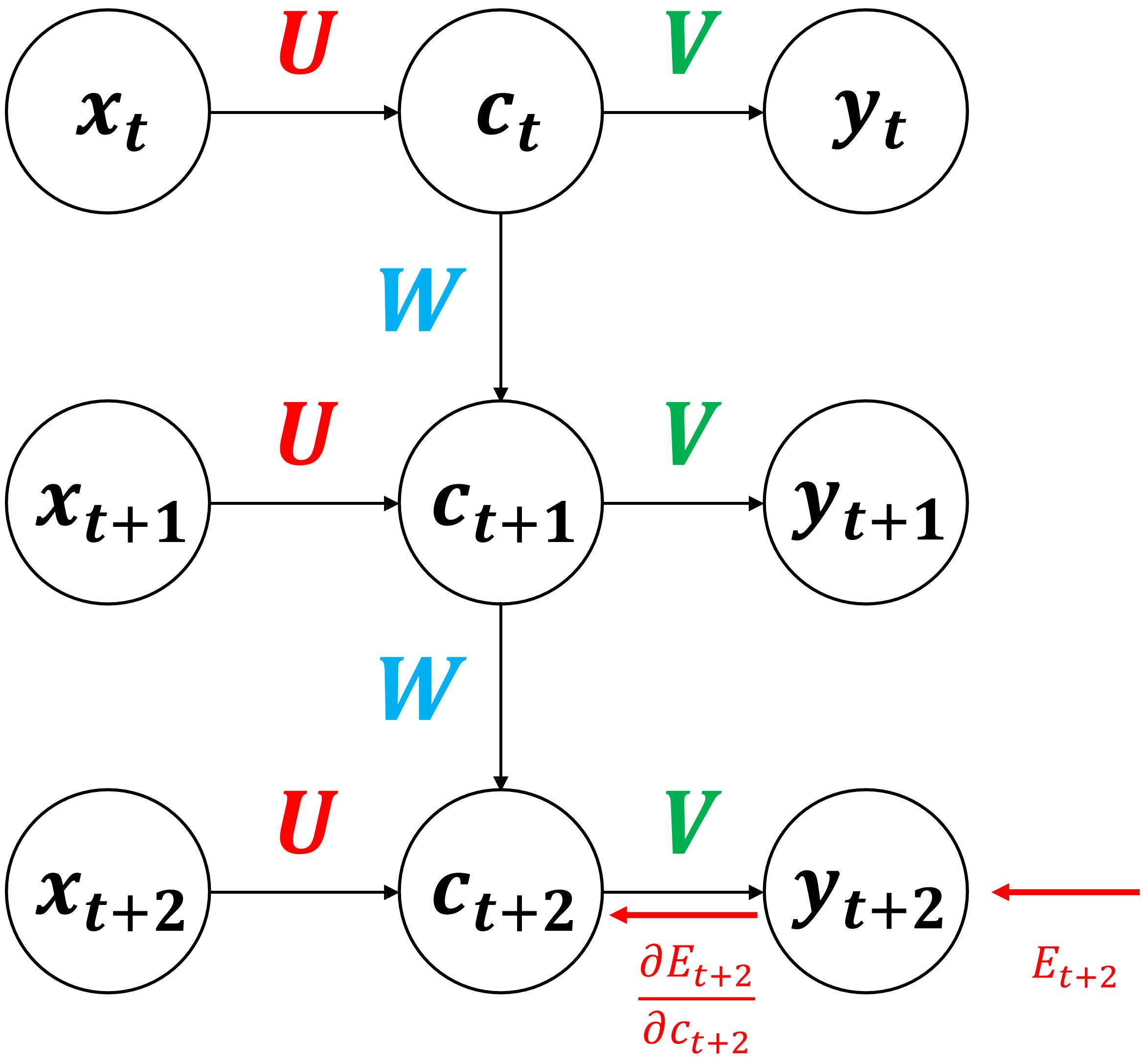


RNN

- RNN Training Process – Backward (Backpropagation Through Time)

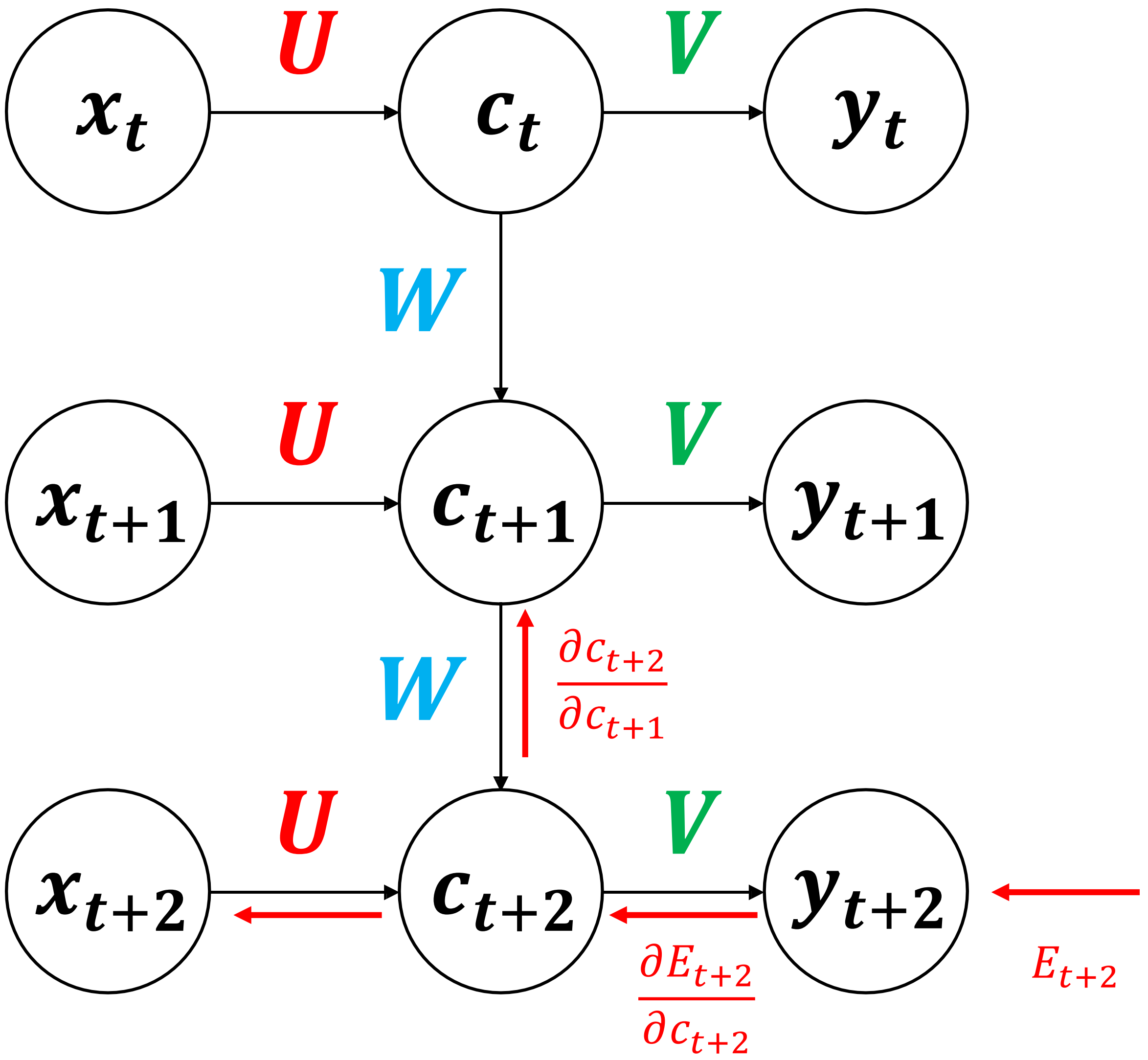


- RNN Training Process – Backward (Backpropagation Through Time)



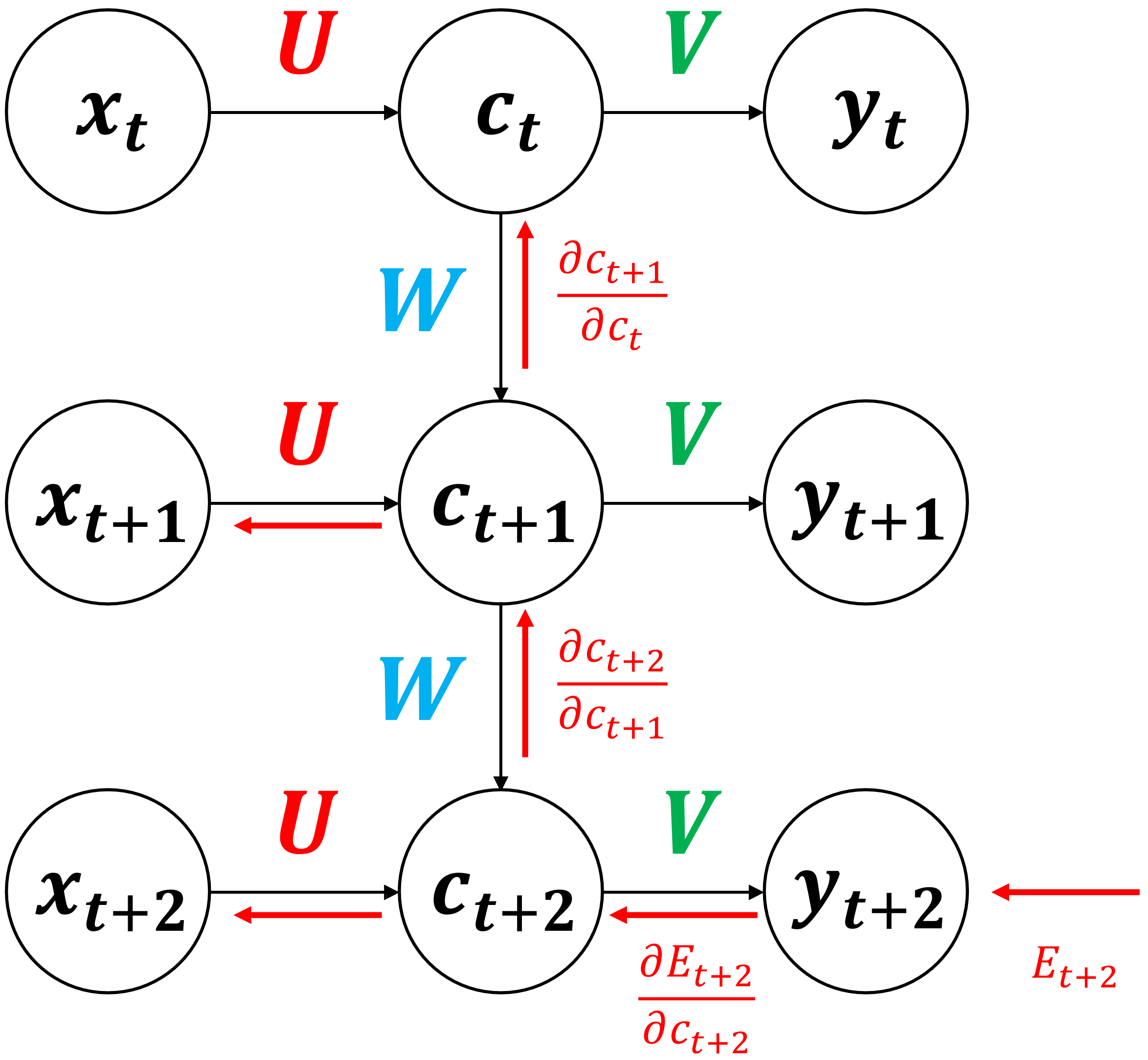
RNN

- RNN Training Process – Backward (Backpropagation Through Time)



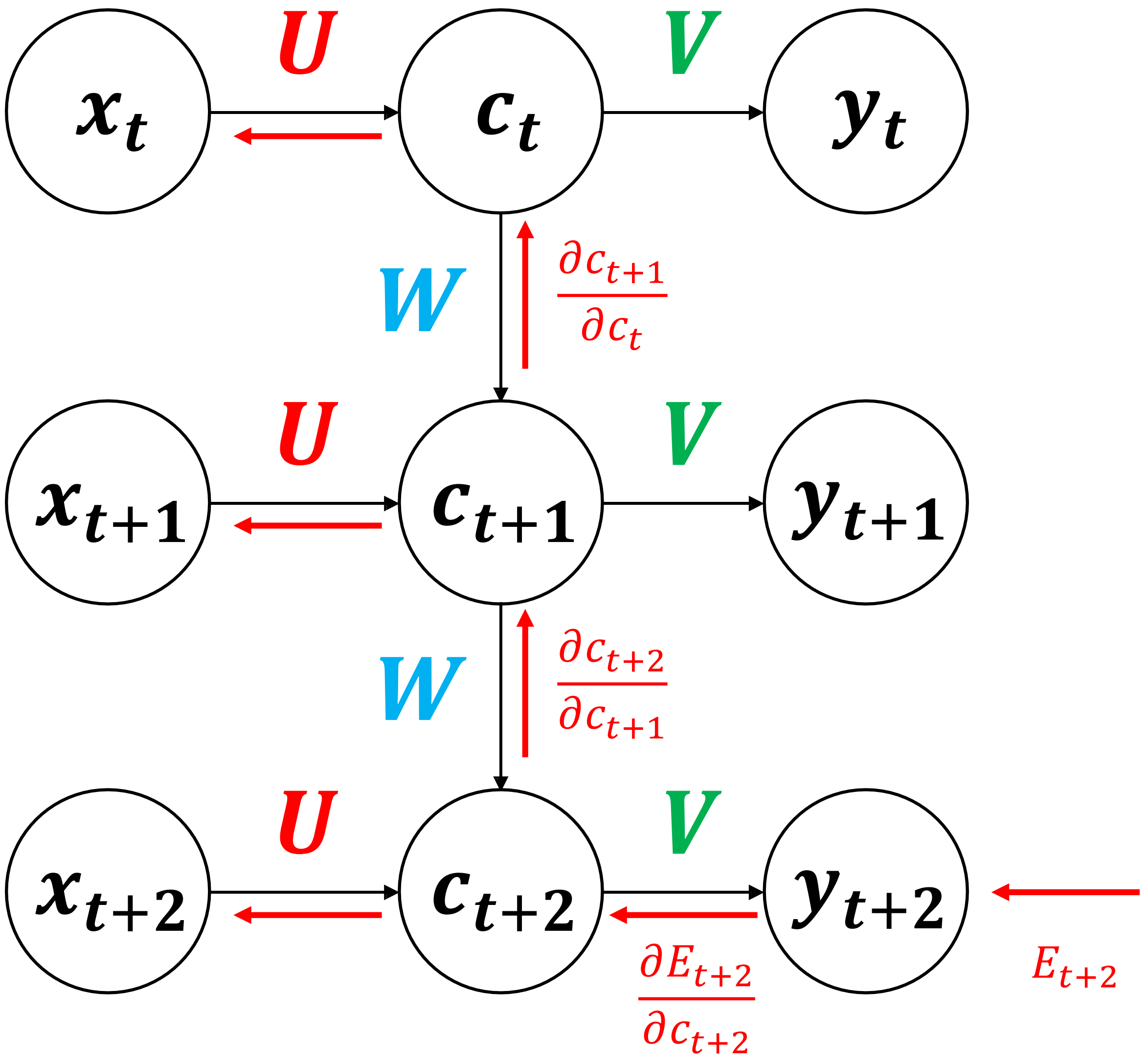
RNN

- RNN Training Process – Backward (Backpropagation Through Time)



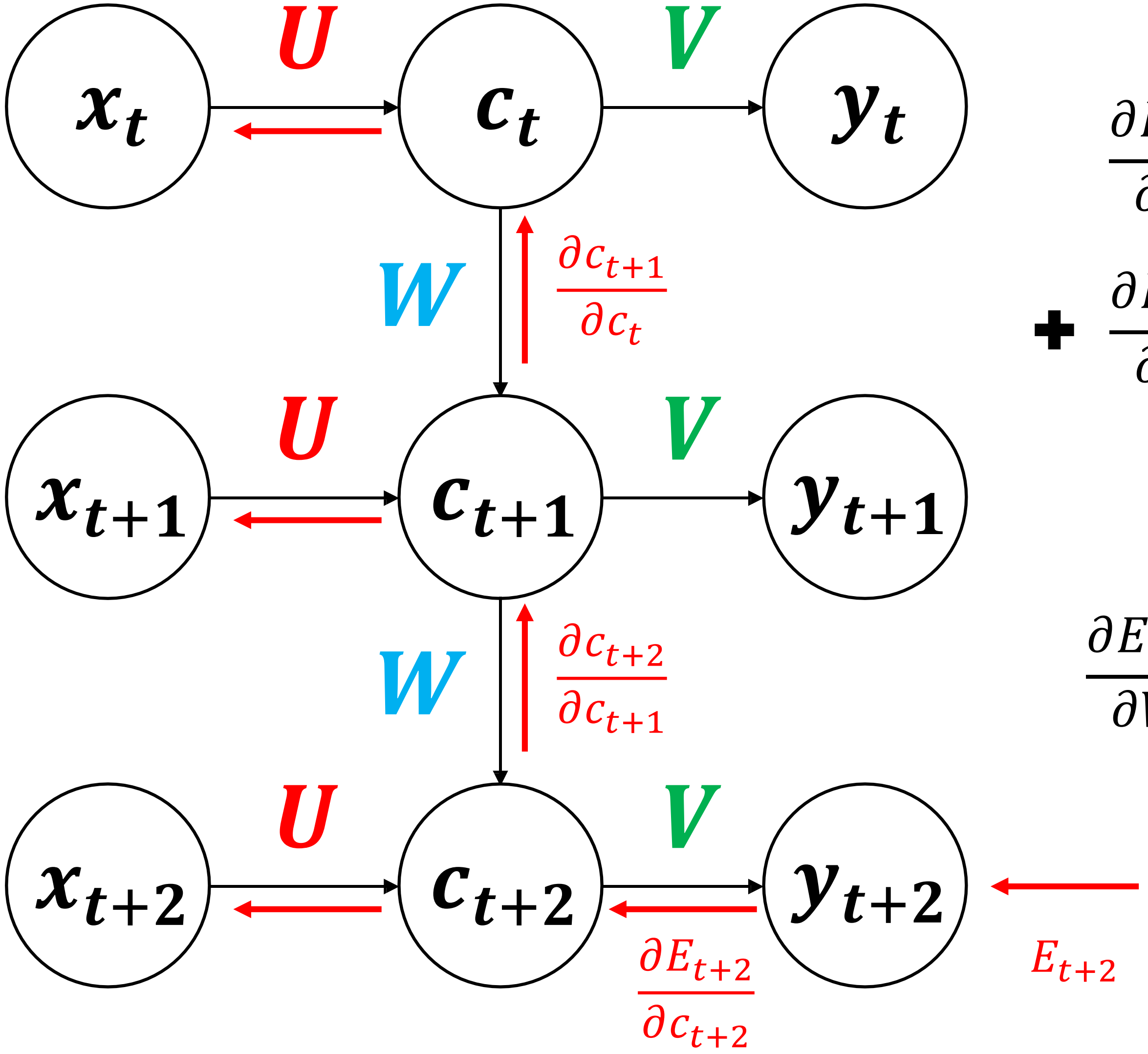
RNN

- RNN Training Process – Backward (Backpropagation Through Time)



RNN

- RNN Training Process – Backward **(Backpropagation Through Time)**



$$\frac{\partial E_{t+2}}{\partial W} = \frac{\partial E_{t+2}}{\partial c_{t+2}} * \frac{\partial c_{t+2}}{\partial W}$$

$$+ \quad \frac{\partial E_{t+2}}{\partial W} = \frac{\partial E_{t+2}}{\partial c_{t+2}} * \frac{\partial c_{t+2}}{\partial c_{t+1}} * \frac{\partial c_{t+1}}{\partial W}$$

$$\Downarrow$$

$$\frac{\partial E_{t+2}}{\partial W} = \sum_{k=t}^{t+2} \frac{\partial E_{t+2}}{\partial c_{t+2}} * \frac{\partial c_{t+2}}{\partial c_k} * \frac{\partial c_k}{\partial W}$$

Fast campus

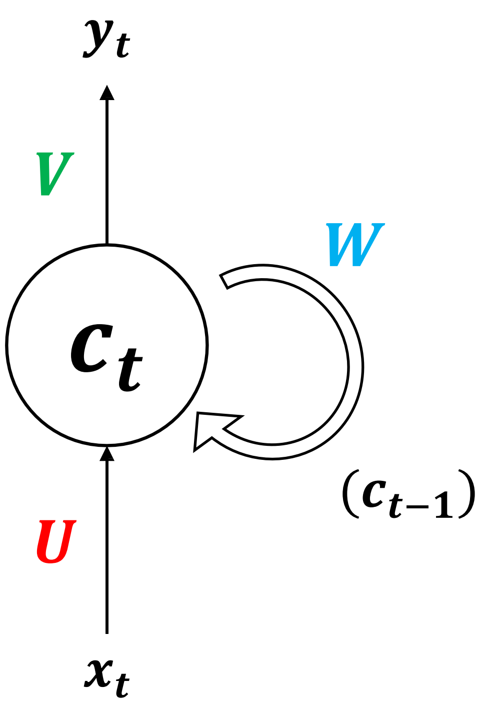
RNN

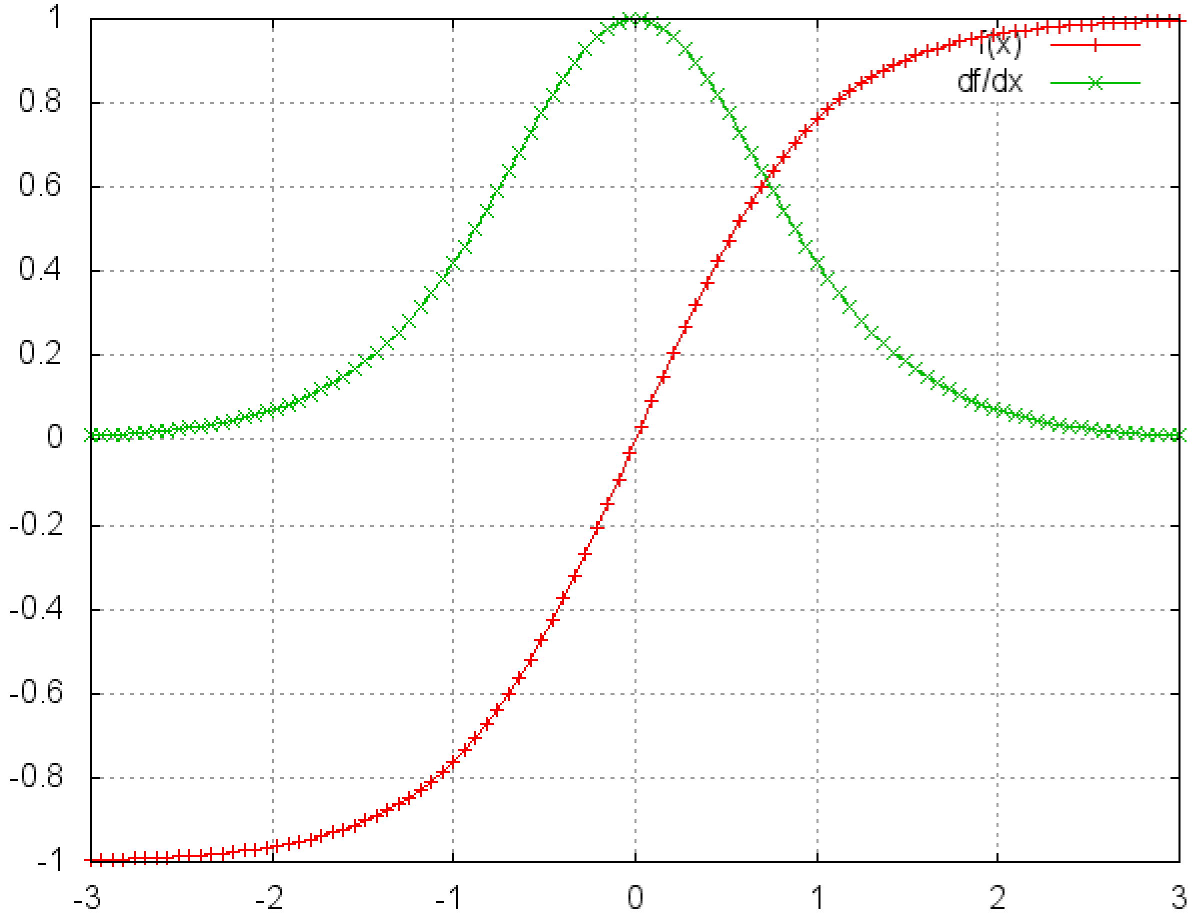
- **RNN Training Process – Backward (Backpropagation Through Time)**
  - $c_t = h(Ux_t)$
  - $c_{t+1} = h(Ux_{t+1} + Wc_t) = h(Ux_{t+1} + W\,h(Ux_t))$
  - $c_{t+2} = h(Ux_{t+2} + Wc_{t+1}) = h(Ux_{t+2} + W\,h(Ux_{t+1} + W\,h(Ux_t)))$
  - $y = h'(Vc_{t+2}) = h'(V\,h(Ux_{t+2} + W\,h(Ux_{t+1} + W\,h(Ux_t))))$
  - $\frac{\partial c_t}{\partial W} = 0$
  - $\frac{\partial c_{t+1}}{\partial W} = \frac{\partial h}{\partial(Ux_{t+1}+Wc_t)}\frac{\partial(Ux_{t+1}+Wc_t)}{\partial W} = \frac{\partial h}{\partial(Ux_{t+1}+Wc_t)}c_t$
  - $\frac{\partial c_{t+2}}{\partial W} = \frac{\partial h}{\partial(Ux_{t+2}+Wc_{t+1})}\frac{\partial(Ux_{t+2}+Wc_{t+1})}{\partial W} = \frac{\partial h}{\partial(Ux_{t+2}+Wc_{t+1})}\frac{\partial(Wc_{t+1})}{\partial W} = \frac{\partial h}{\partial(Ux_{t+2}+Wc_{t+1})}\left(c_{t+1} + \frac{\partial c_{t+1}}{\partial W}\right)$

$$\frac{\partial E_{t+2}}{\partial W} = \sum_{k=t}^{t+2} \frac{\partial E_{t+2}}{\partial c_{t+2}} * \frac{\partial c_{t+2}}{\partial c_k} * \frac{\partial c_k}{\partial W}$$

RNN

- **RNN Training Process**
  - $h = \tanh(x)$
  - $c_t = h(Ux_t + Wc_{t-1})$
  - $y_t = h'(Vc_t)$





http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

RNN

- **RNN Training Problem**
  - $\mathcal{L} = L(c_T(c_{T-1}(\ldots(c_1(x_1, c_0; W); W); W); W)$
  - $\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{\tau=1}^{t} \frac{\partial \mathcal{L}_t}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} \frac{\partial c_\tau}{\partial W}$
  - $\frac{\partial \mathcal{L}}{\partial c_t} \frac{\partial c_t}{\partial c_\tau} = \frac{\partial \mathcal{L}}{\partial c_t} \cdot \frac{\partial c_t}{\partial c_{t-1}} \cdot \frac{\partial c_{t-1}}{\partial c_{t-2}} \cdot \cdots \cdot \frac{\partial c_{\tau+1}}{\partial c_\tau}$
  - $\frac{\partial \mathcal{L}}{\partial c_t} = \frac{\partial \mathcal{L}}{\partial c_T} \cdot \underset{>1}{\overset{<1}{\frac{\partial c_T}{\partial c_{T-1}}}} \cdot \underset{>1}{\overset{<1}{\frac{\partial c_{T-1}}{\partial c_{T-2}}}} \cdot \cdots \cdot \underset{>1}{\overset{<1}{\frac{\partial c_{t+1}}{\partial c_t}}}$

    < 1 → Vanishing gradient

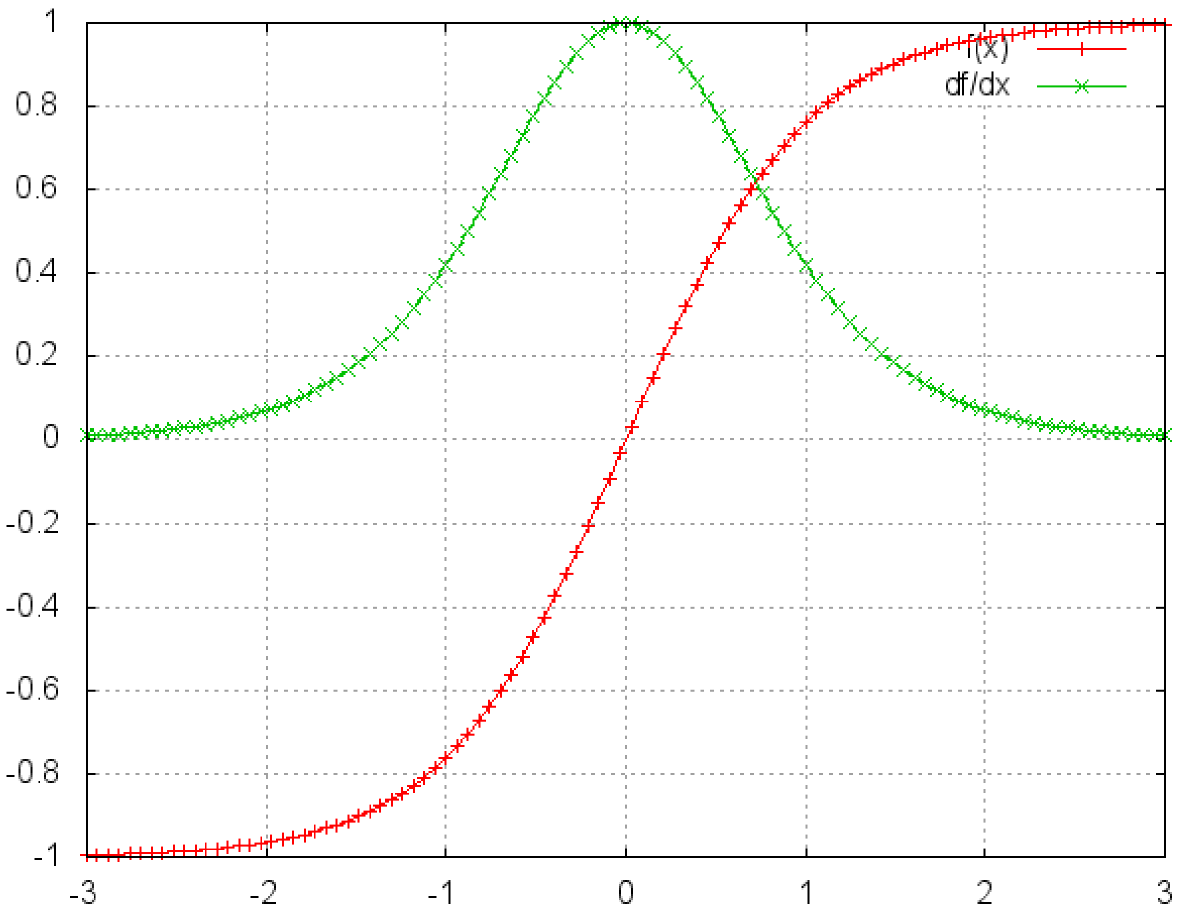
    > 1 → Exploding gradient

Fast campus

RNN

RNN Training Problem

- $h = \tanh(x)$
- 기울기가 0~1의 값만 출력
  - 따라서 점점 0으로 수렴
- Gradient Vanishing 문제 발생
  - 일반적으로 RNN은 깊음



http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

3. LSTM

Fast campus

Sequential Data

RNN

LSTM

GRU

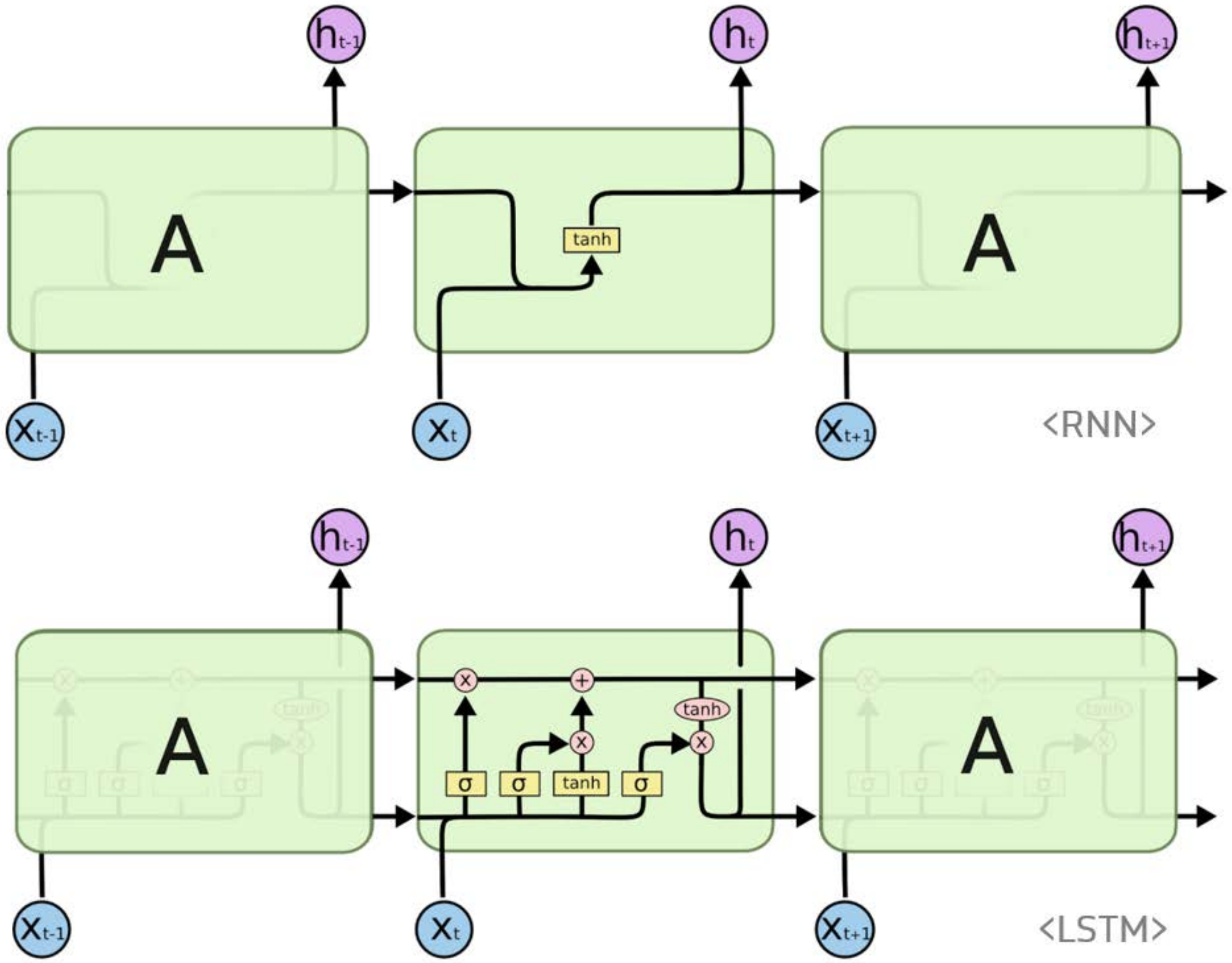
Applications

LSTM

- **How to fix gradient vanishing?**
  - 기울기가 너무 작지도(<1) 크지도(>1) 않게?
  - “기울기 = 1”로 하면 되겠다!

LSTM

- **LSTM Structure**



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

Fast campus

Sequential Data

RNN

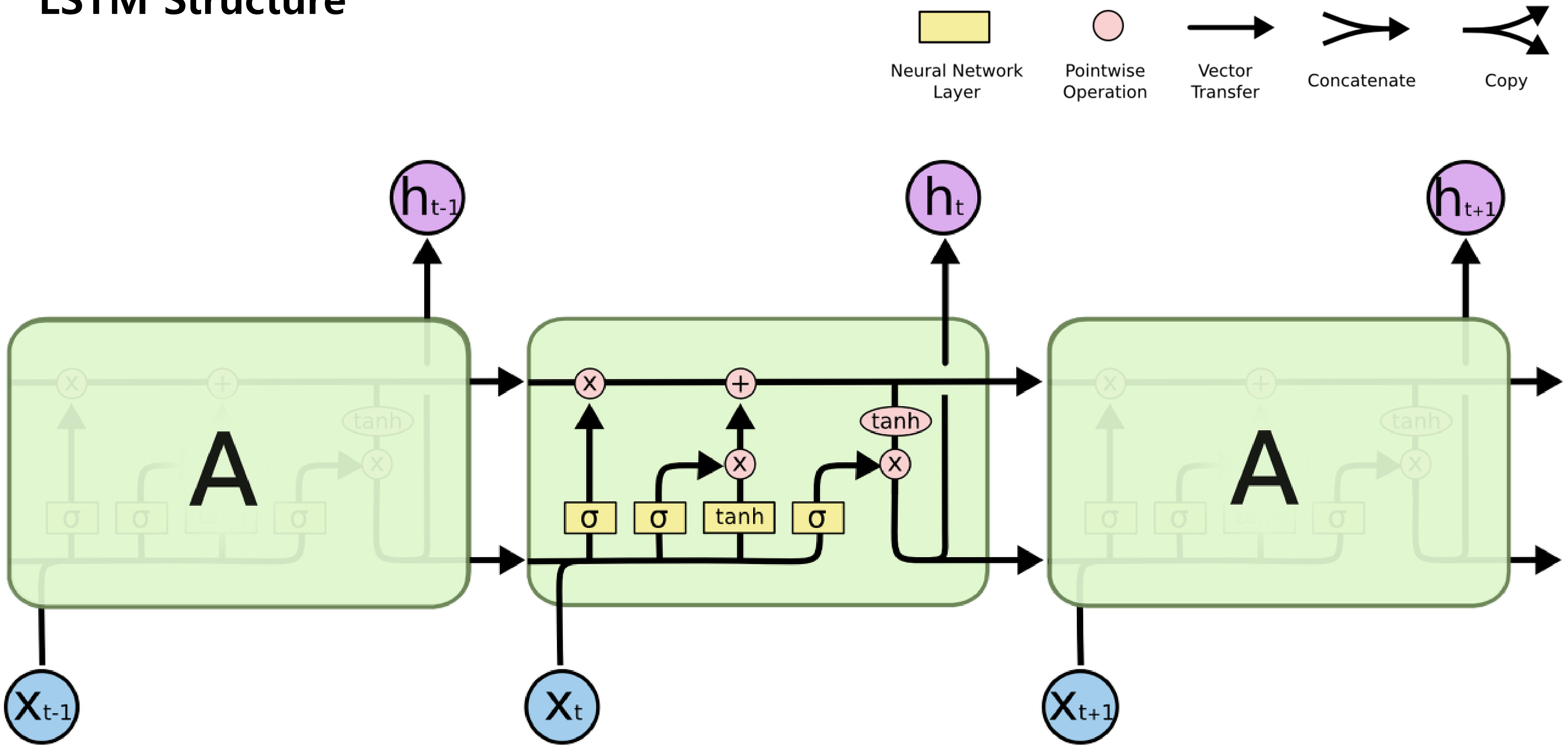
LSTM

GRU

Applications

LSTM

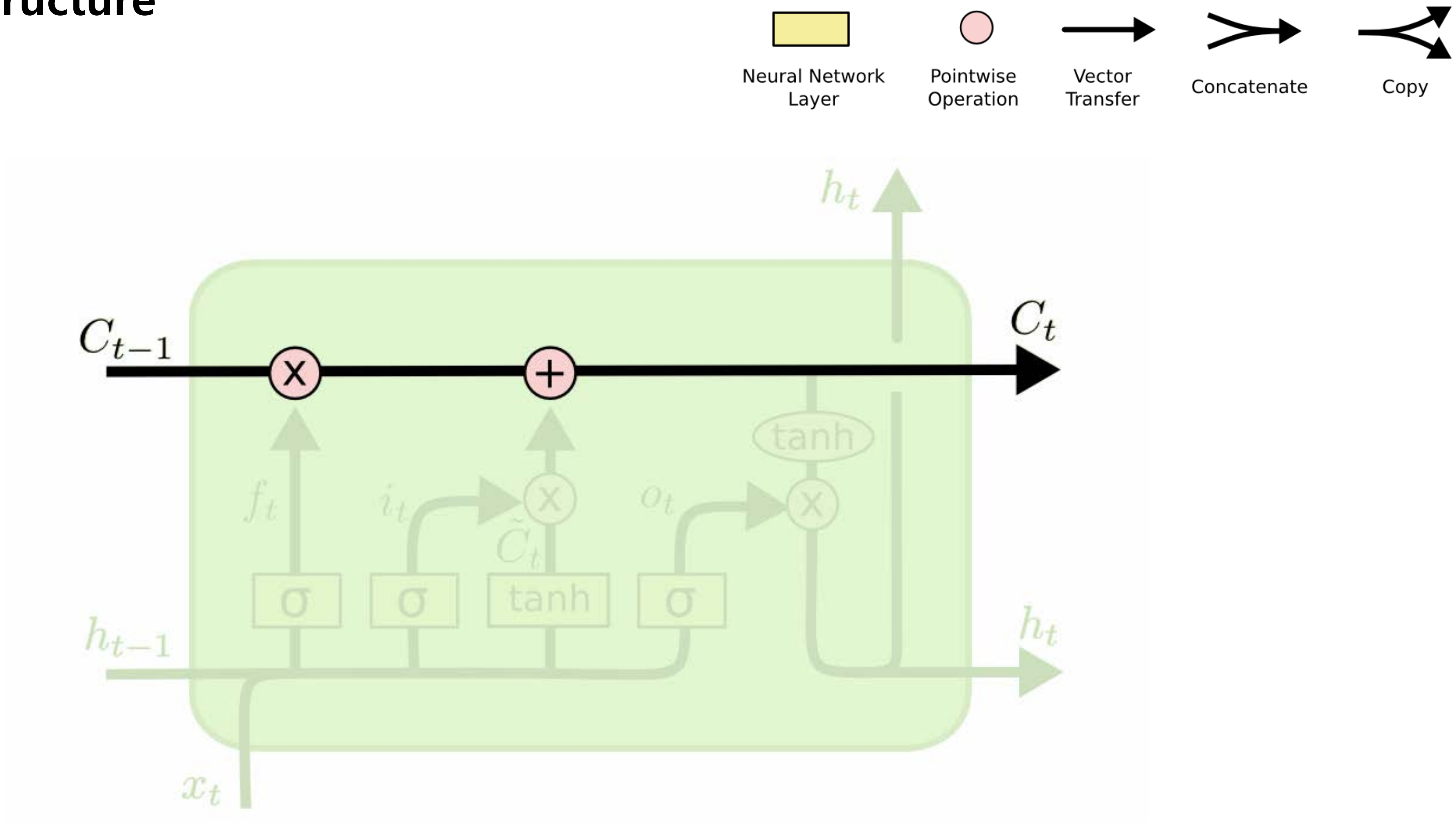
- ## LSTM Structure



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

LSTM

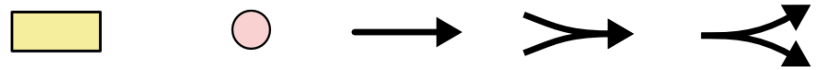
- **LSTM Structure**

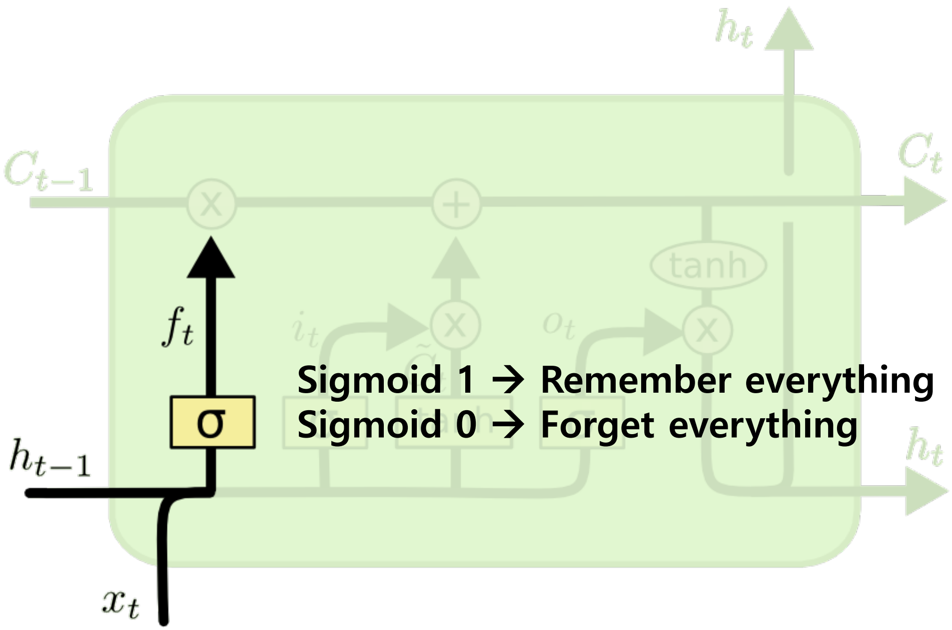


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

- **LSTM Structure**





<Forget Gate Layer>

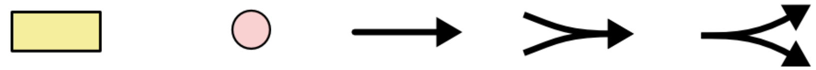
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

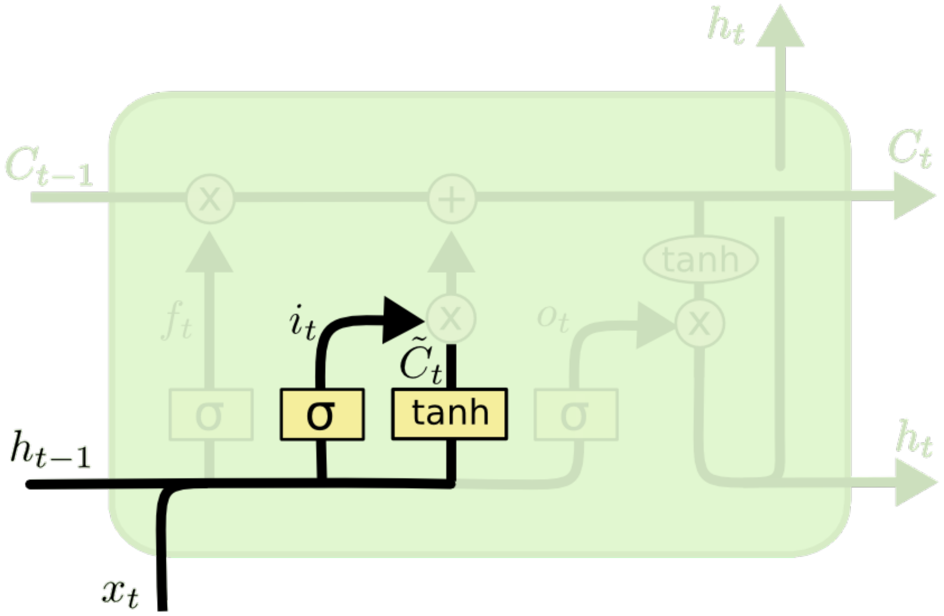
Input과 이전 Hidden State으로
이전의 Cell State의 전달 정도 결정

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

LSTM

- **LSTM Structure**





<Input Gate Layer>

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

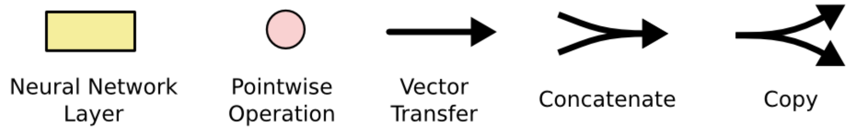
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

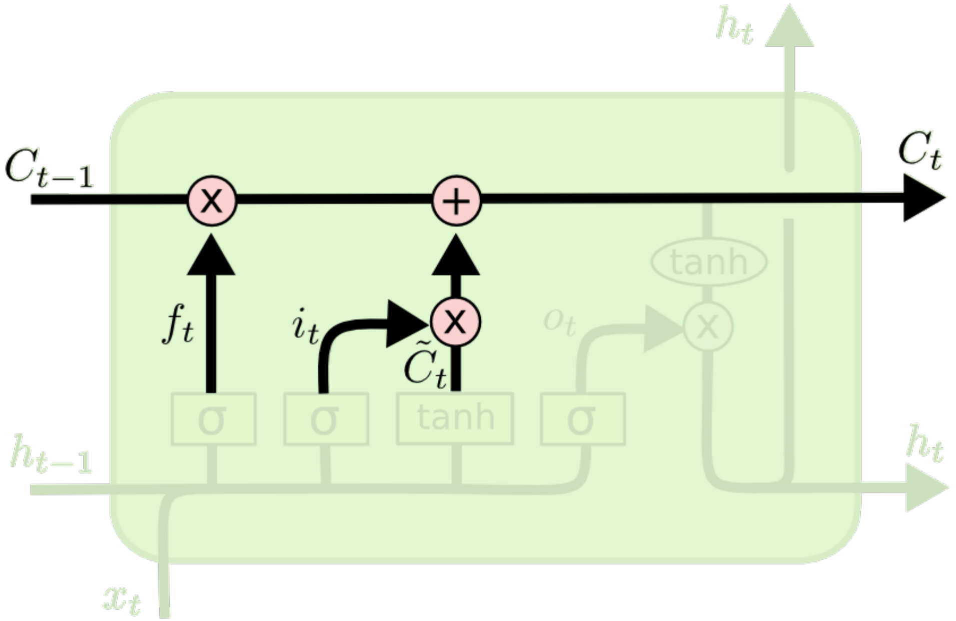
Input과 이전 Hidden State으로
이전 State의 정보와 반영비율 결정

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

LSTM

- **LSTM Structure**





<New Cell State>

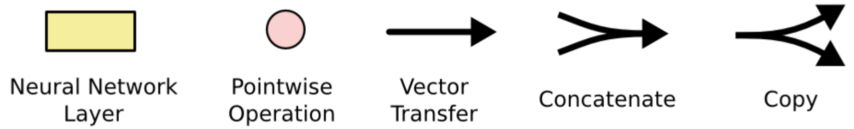
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

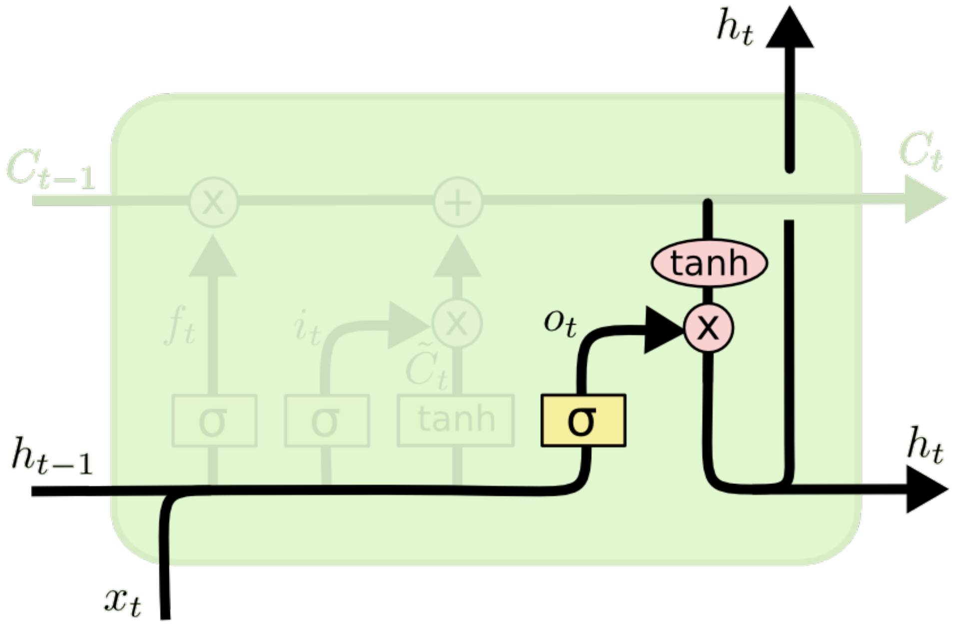
이전 단계에서 결정된 정보를 바탕으로
새로운 Cell State를 생성

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

LSTM

- **LSTM Structure**





<Final>

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

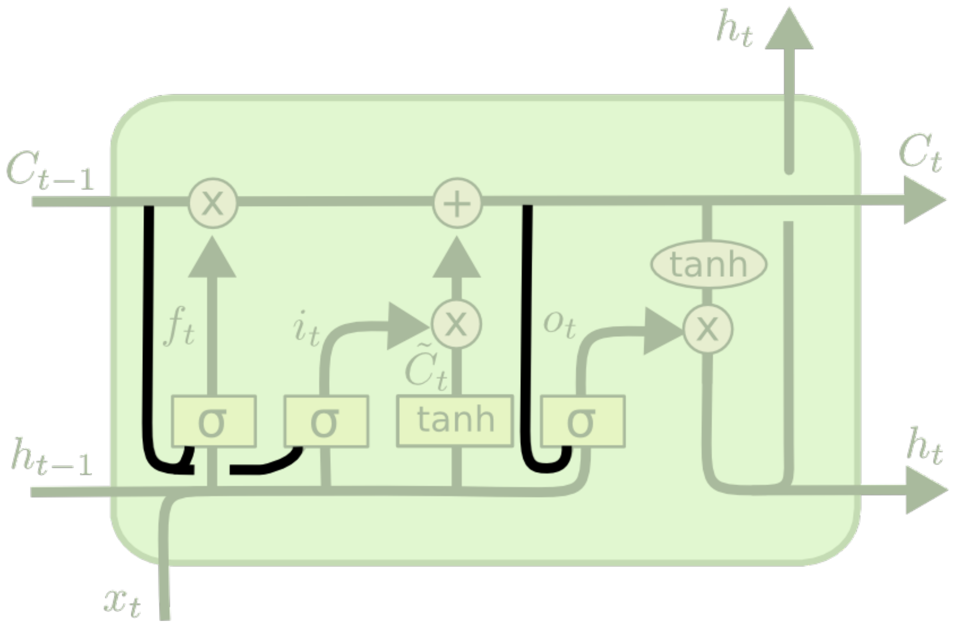
$$h_t = o_t * \tanh\left(C_t\right)$$

앞서 생성된 Cell State와 새롭게 비중을 곱하여
새로운 Hidden State를 생성

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

LSTM

- **LSTM Structure Variation**
  - Peephole connections
  - Gers & Schmidhuber (2000)



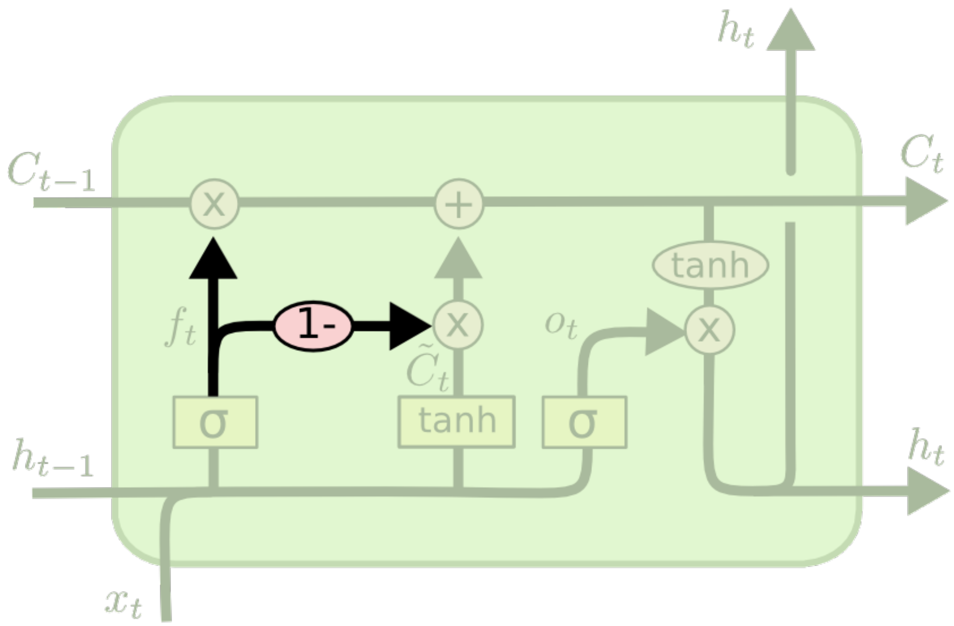
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

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

LSTM

▪ LSTM Structure Variation

- Coupled forget and input gates
- Only forget when we're going to input something in its place



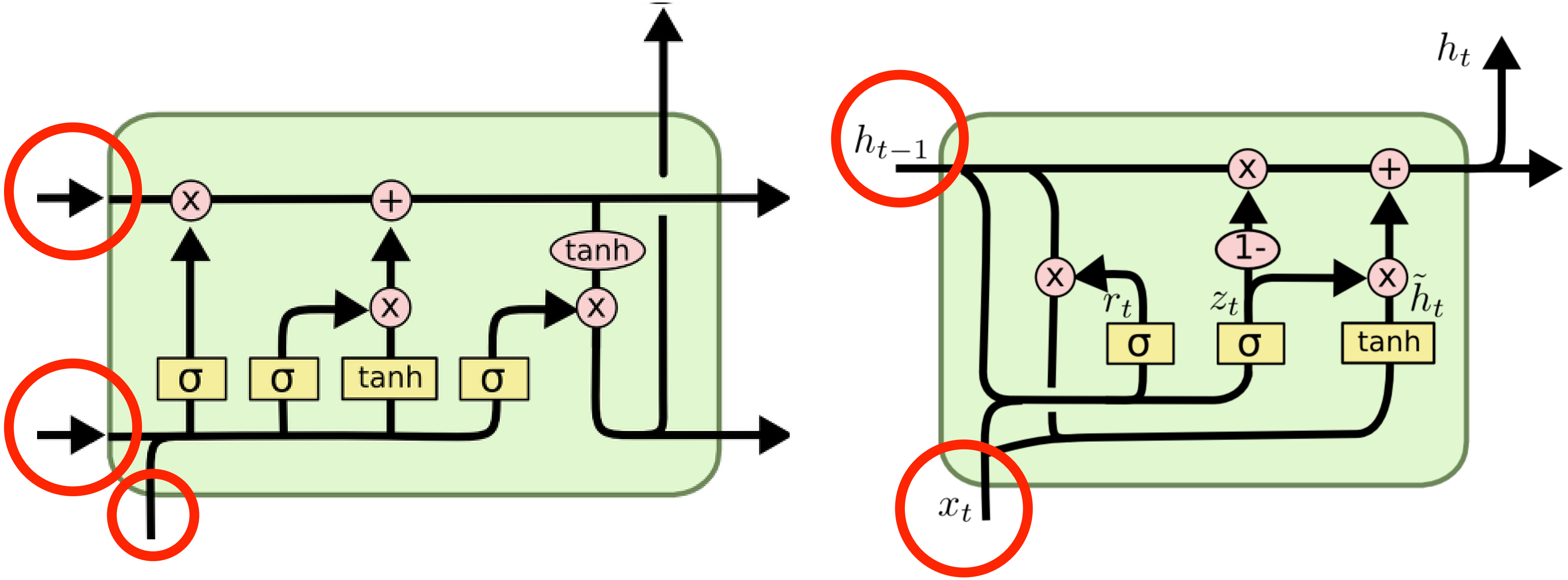
$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

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

4. GRU

GRU

- **LSTM v.s. GRU**

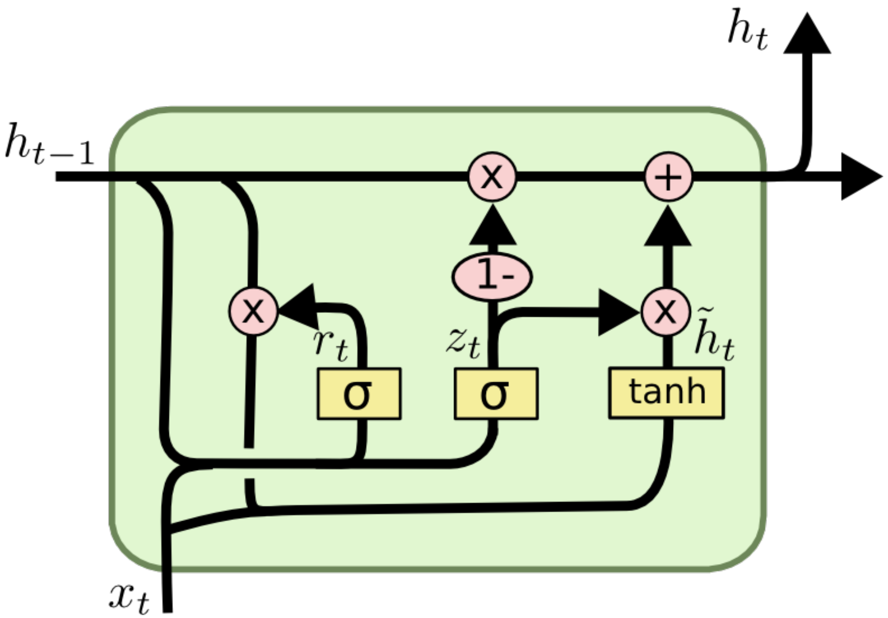


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Fast campus

GRU

▪ GRU Structure



<Update Gate>

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

Forget Gate + Input Gate

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

Fast campus

GRU

Sequential Data

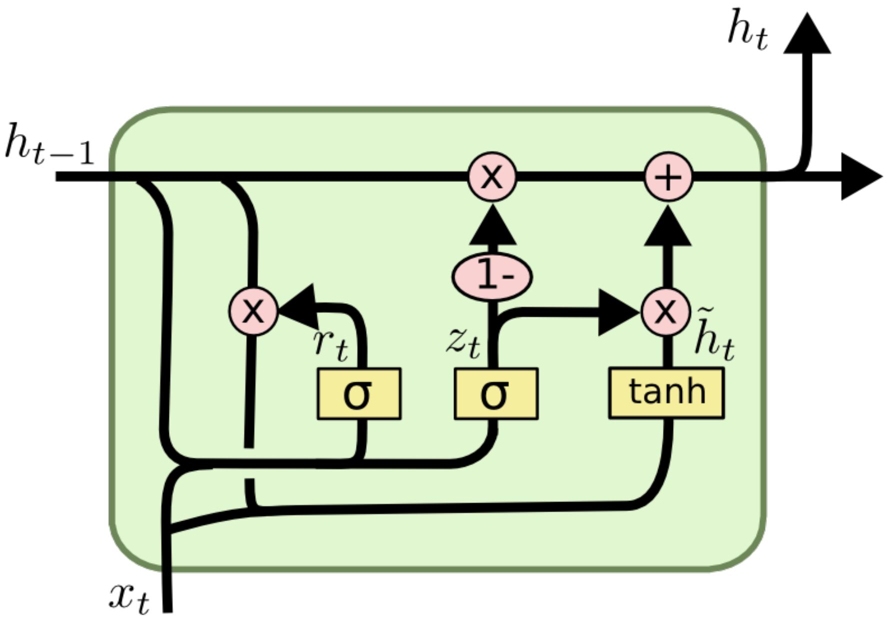
RNN

LSTM

GRU

Applications

- **GRU Structure**



<Reset Gate>

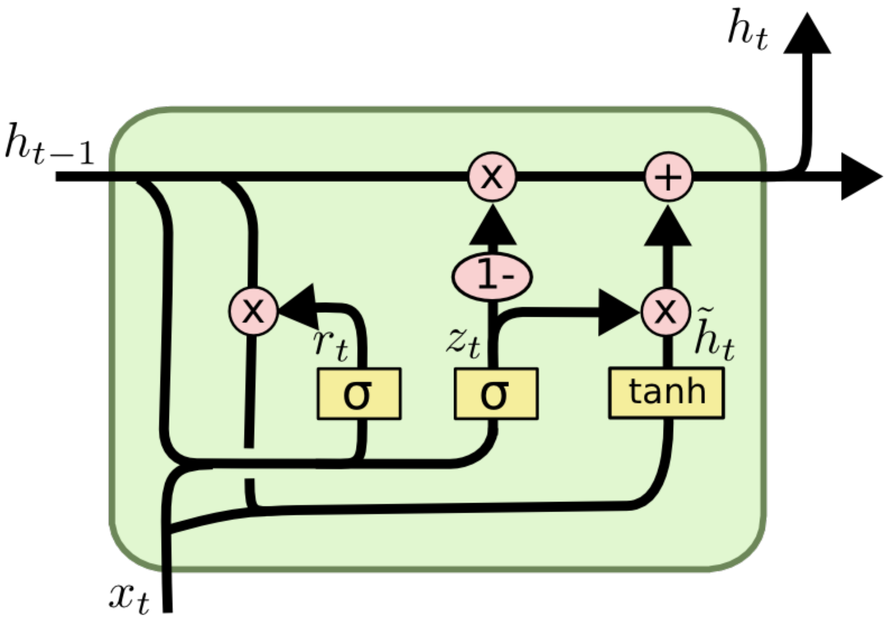
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

0이면 과거 정보 모두 잊음
1이면 과거 정보 모두 기억

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

GRU

- ## GRU Structure



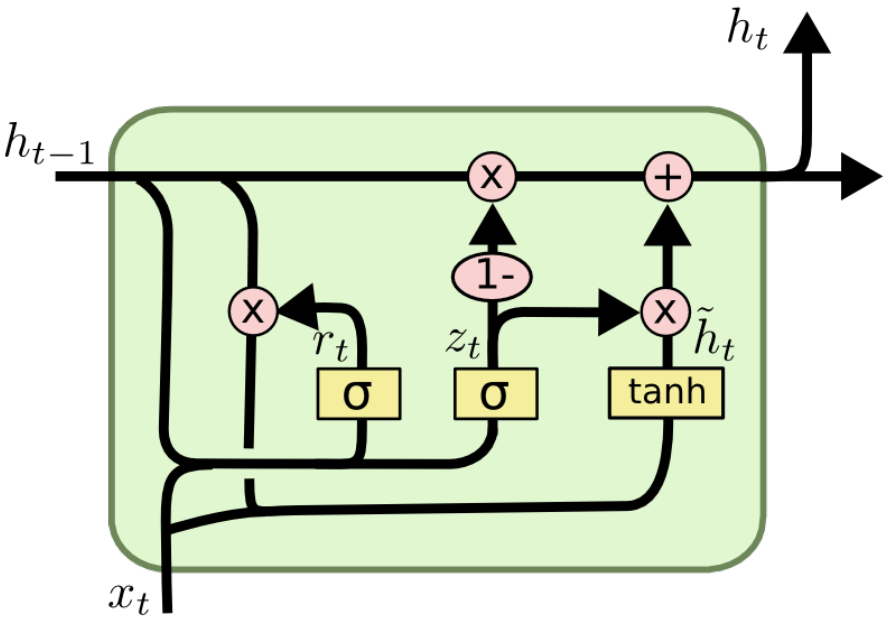
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

Reset Gate 값을 통해 과거 정보 반영 비중 결정

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

GRU

- **GRU Structure**



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

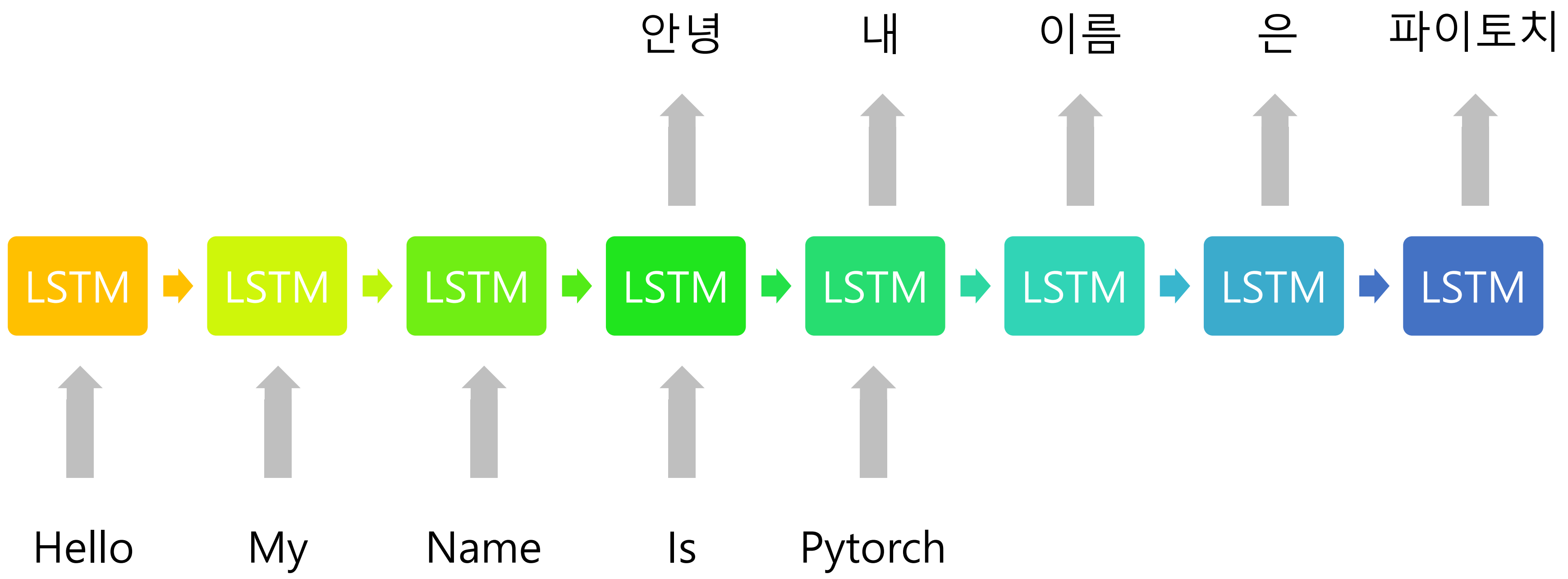
최종적으로 새로운 Hidden State 결정
0이면 과거 정보 모두 기억, 현재 정보 무시
1이면 과거 정보 모두 무시, 현재 정보 기억

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

5. Applications

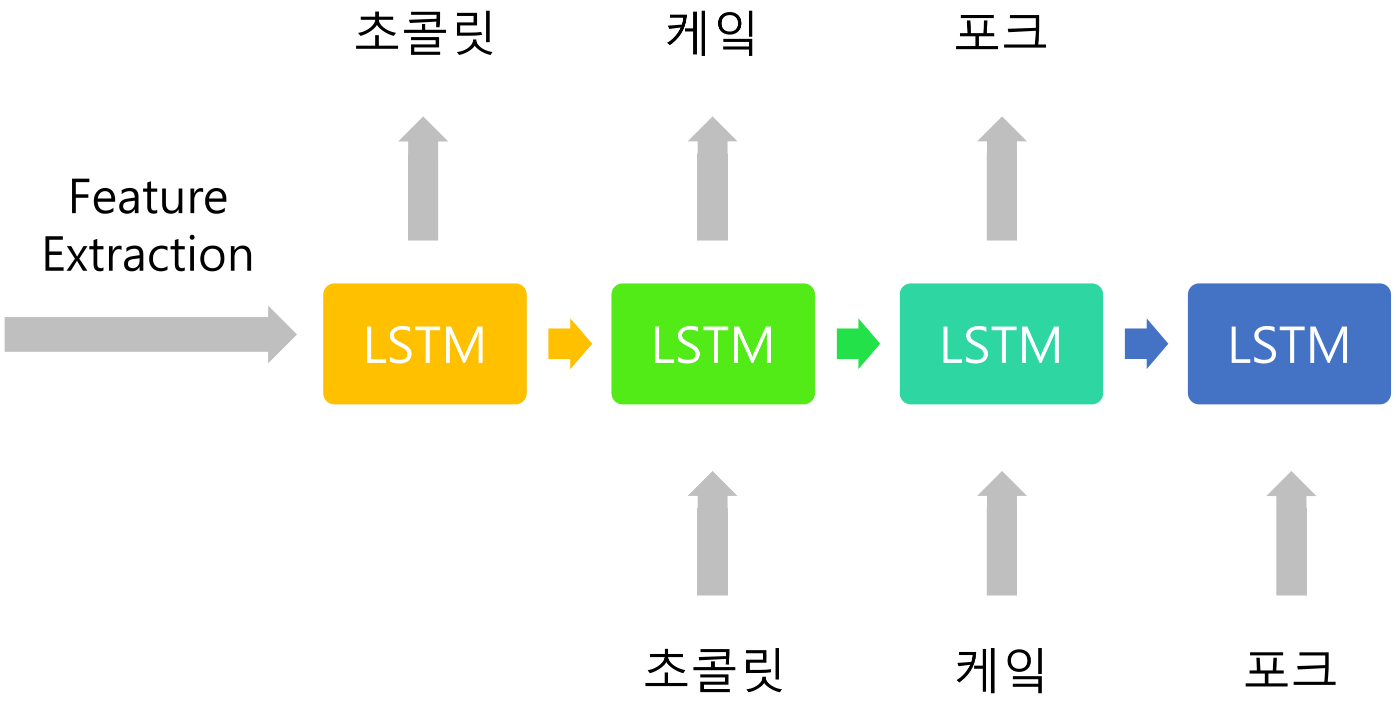
Applications

- **Applications**
  - Language Translation



Applications

- **Applications**
  - Language Translation
  - Image Captioning



Fast campus

Sequential Data

RNN

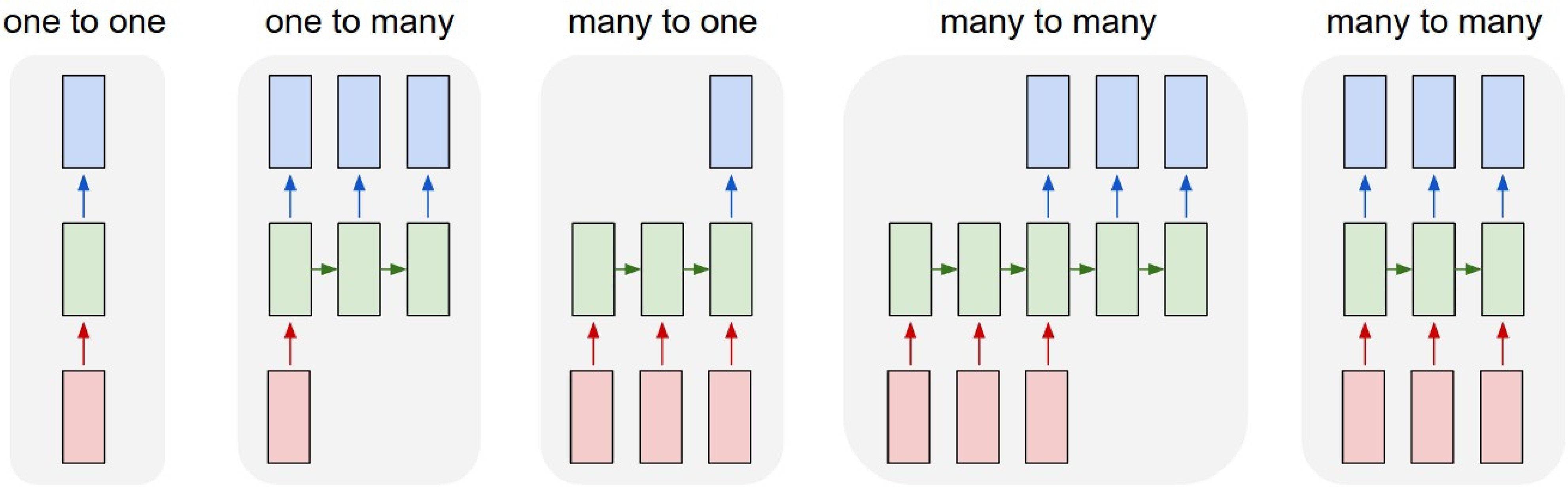
LSTM

GRU

Applications

Applications

- **Applications**
  - Language Translation
  - Image Captioning
  - Video Classification
  - …



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

Sequential Data

RNN

LSTM

GRU

Applications

실 습