AILAB SEMINAR #19

RNN & LSTM

220209 한양대학교 인공지능연구실 김민지

1

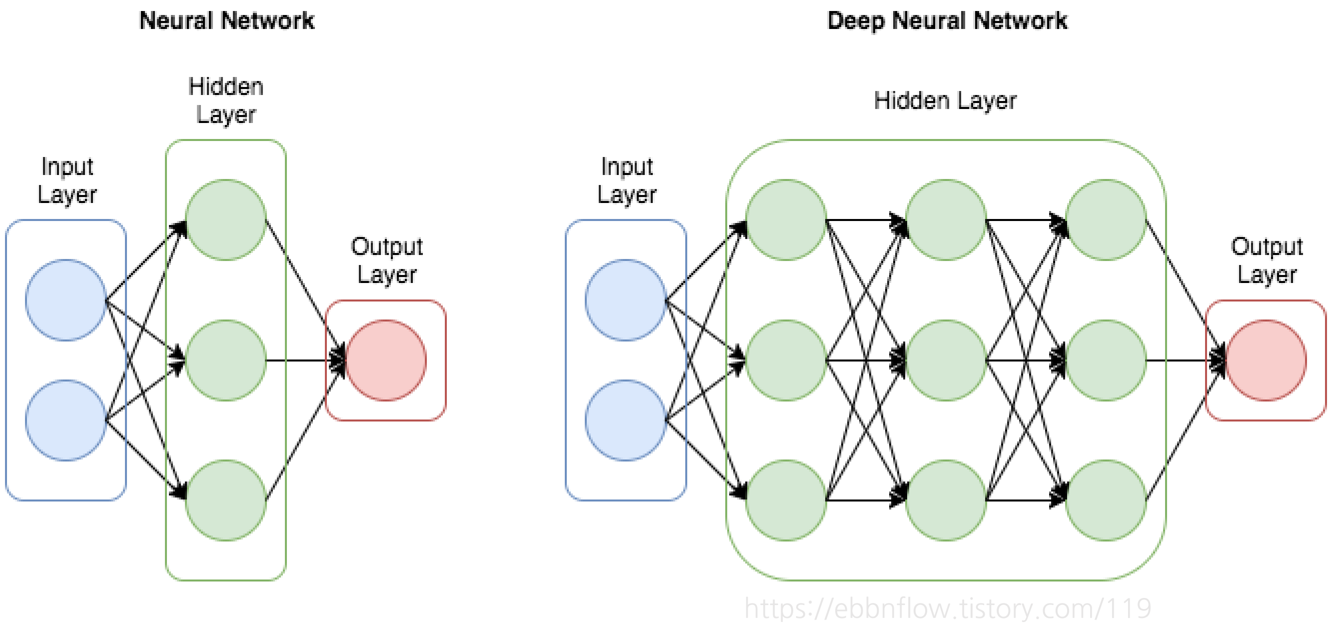
DNN

DEEP NEURAL NETWORK

DNN
RNN
LSTM
GRU

DNN - Deep Neural Network

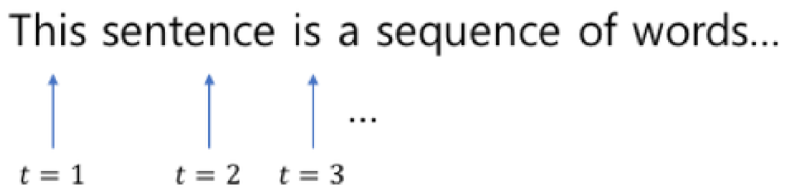
- Input layer 와 Output layer 사이 여러 개의 Hidden layer 가 있는 인공 신경망의 한 형태
- Feed-Forward 와 Backpropagation 을 통해 Loss를 낮추는 방향으로 Parameter를 학습



DNN - DNN의 한계

- 고정된 길이의 Input과 Output에 대해서만 연산이 가능
- 시계열, 자연어 등 Sequence data에서 Context 정보를 활용하기 어려움

자연어



주가

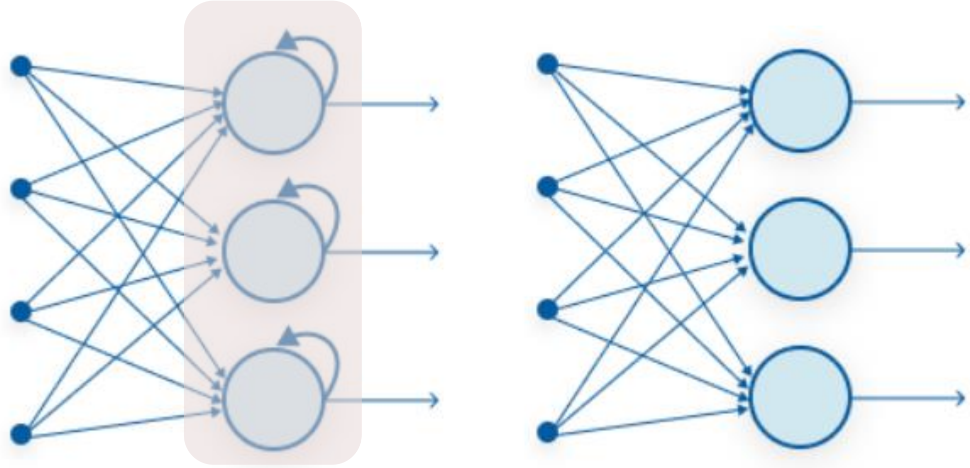
음성

2

RNN

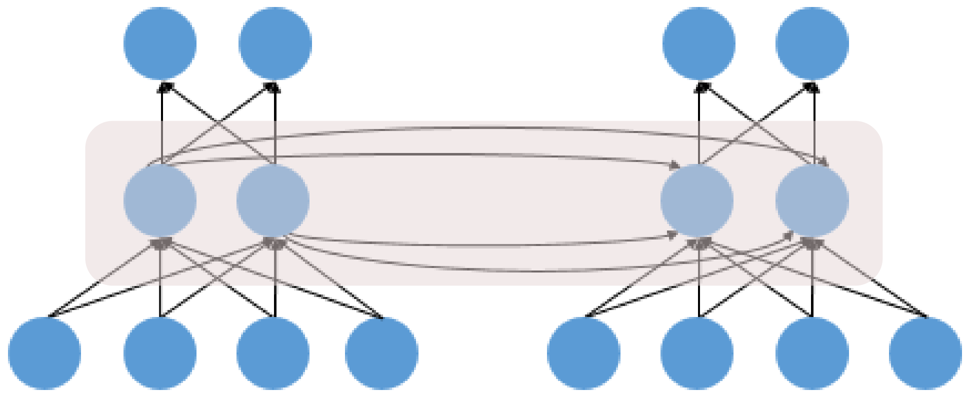
RECURRENT NEURAL NETWORK

- Hidden layer 의 결과 값을 다음 시점의 Hidden Layer 로 보내는 재귀적 특성을 가진 순환 신경망



Recurrent Neural Network

Feed-Forward Neural Network

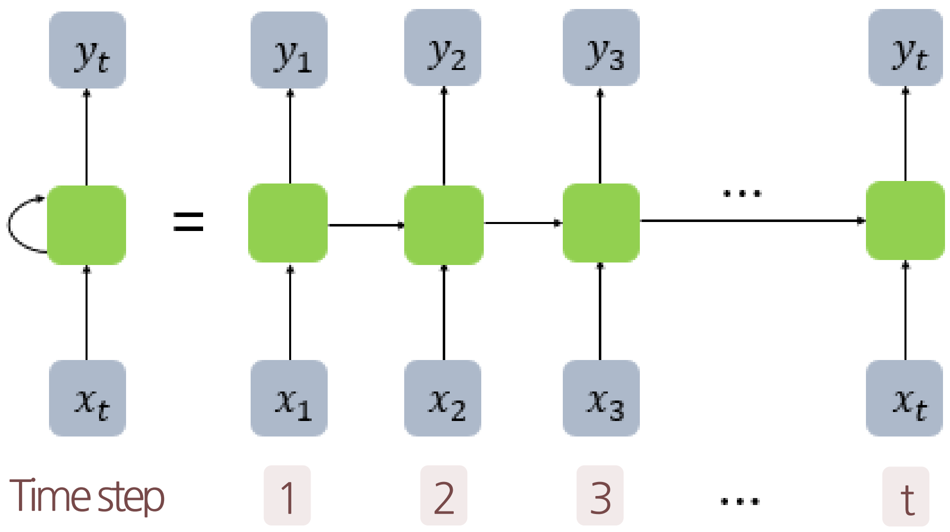


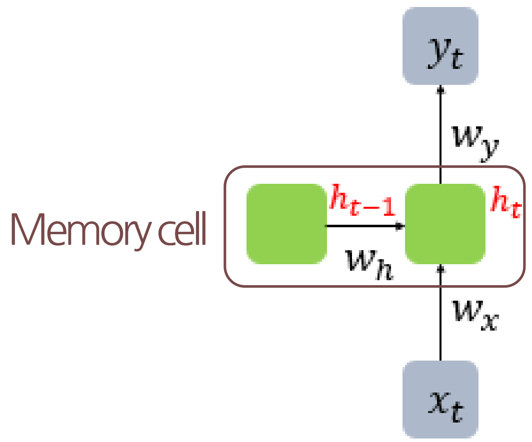
Time step t

Time step t+1

https://dbrang.tistory.com/1537

- Memory cell (RNN cell) 은 이전의 값(Hidden state)을 기억하여 Sequential 정보를 학습

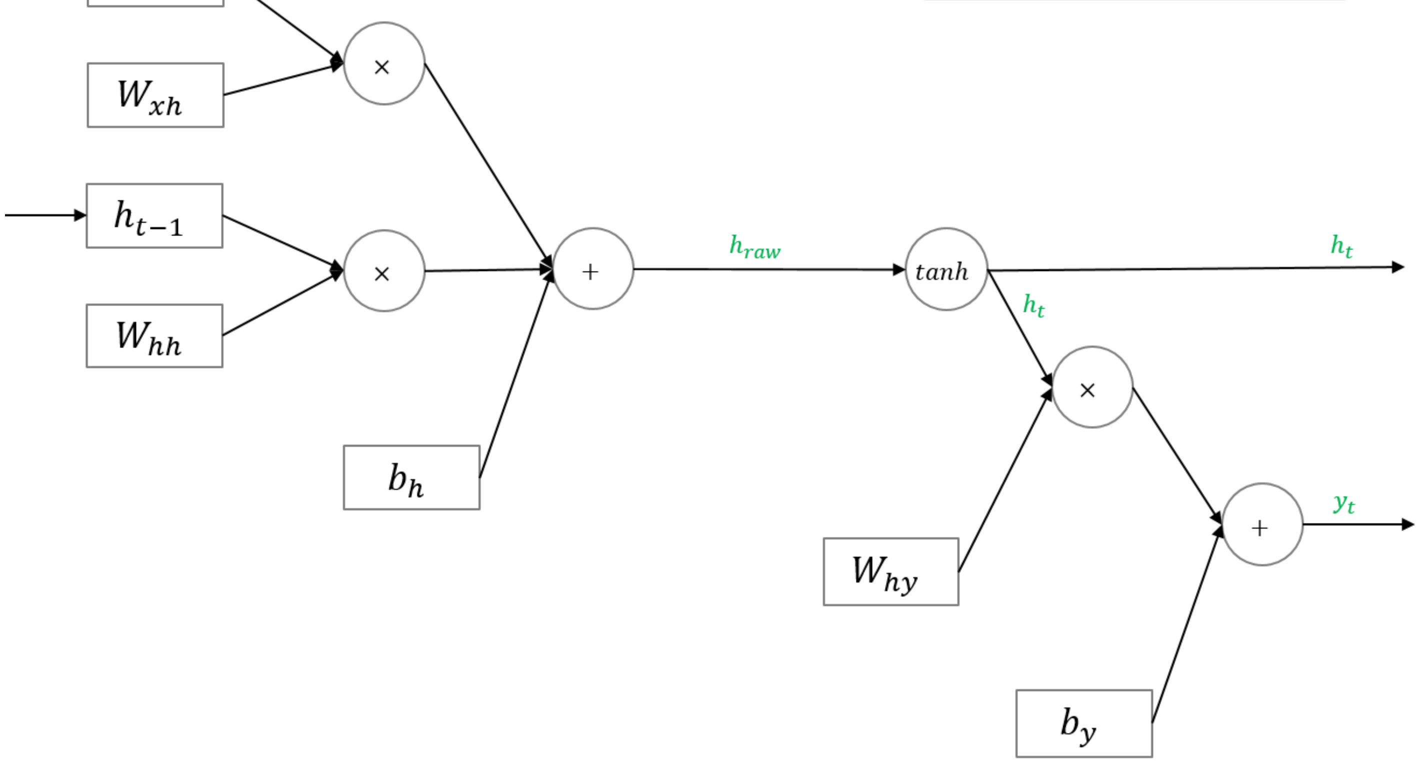


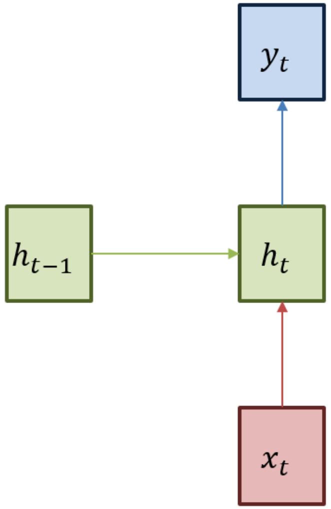


RNN - The Vanilla RNN Forward

- 시점 t 에서 Input = x_t, Hidden state = h_t, Output = y_t
- 각 시점의 W_{hy}, W_{hh}, W_{xh} 값을 공유함

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



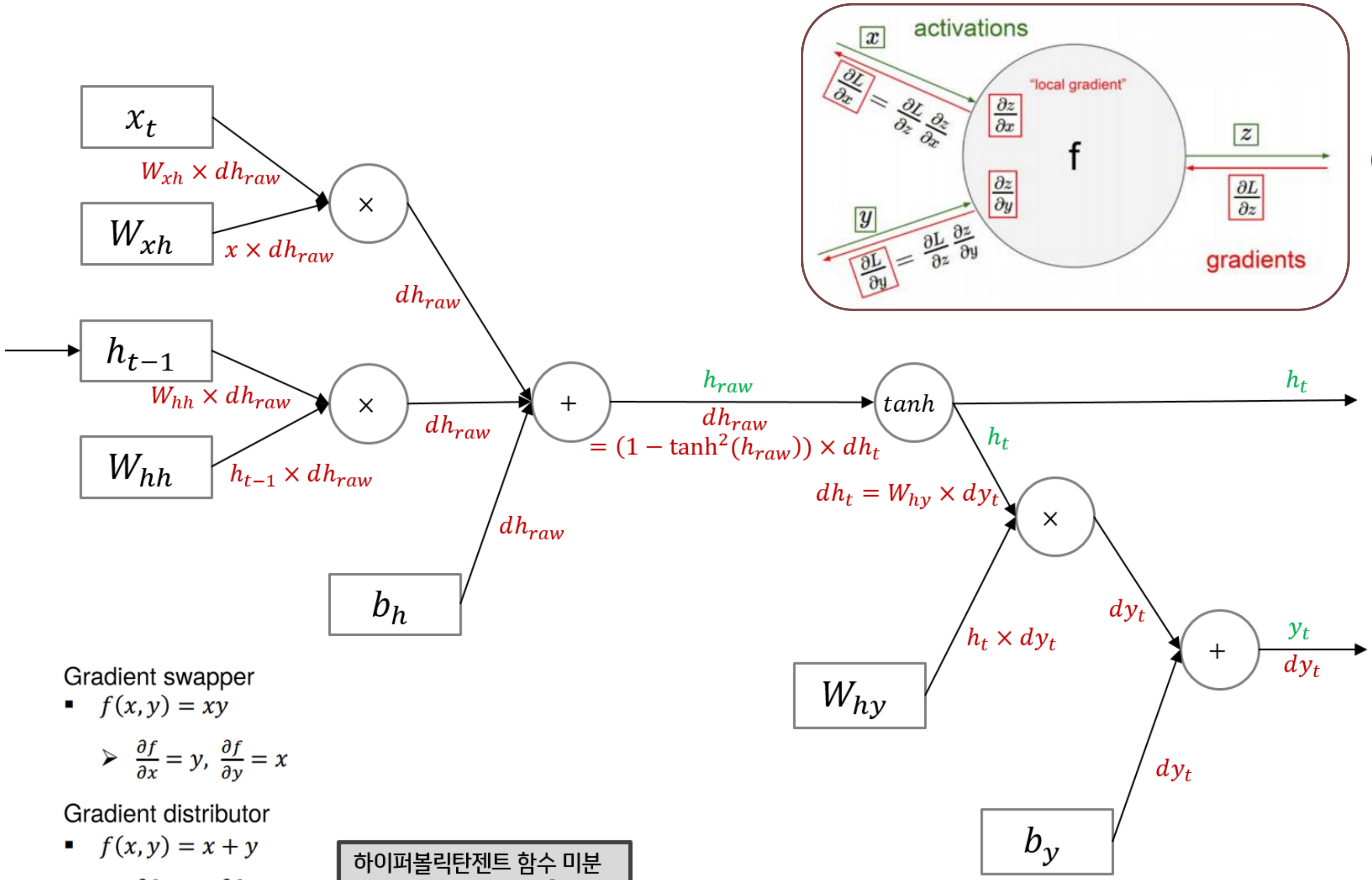


$y_t = W_{hy}h_t + b_y$

Activation function

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

- Time step에 따라 Unfold 된 RNN Cell을 DNN과 유사하게 Backpropagation



Chain rule

Gradient swapper

- $f(x, y) = xy$
  - $\frac{\partial f}{\partial x} = y,\ \frac{\partial f}{\partial y} = x$

Gradient distributor

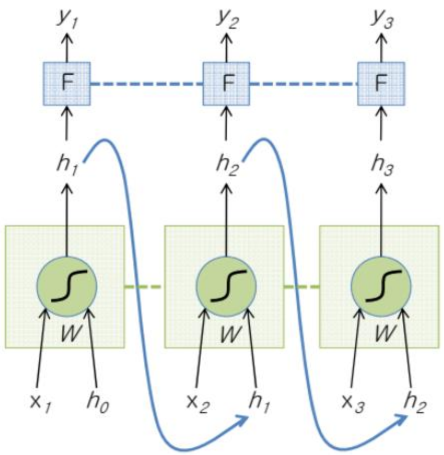
- $f(x, y) = x + y$
  - $\frac{\partial f}{\partial x} = 1,\ \frac{\partial f}{\partial y} = 1$

하이퍼볼릭탄젠트 함수 미분
$\tanh(x) \rightarrow 1 - \tanh^2(x)$

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

RNN - Backpropagation Though Time (BPTT)

- 순환 구조이므로 다음 시점의 Gradient도 더해서 반영함

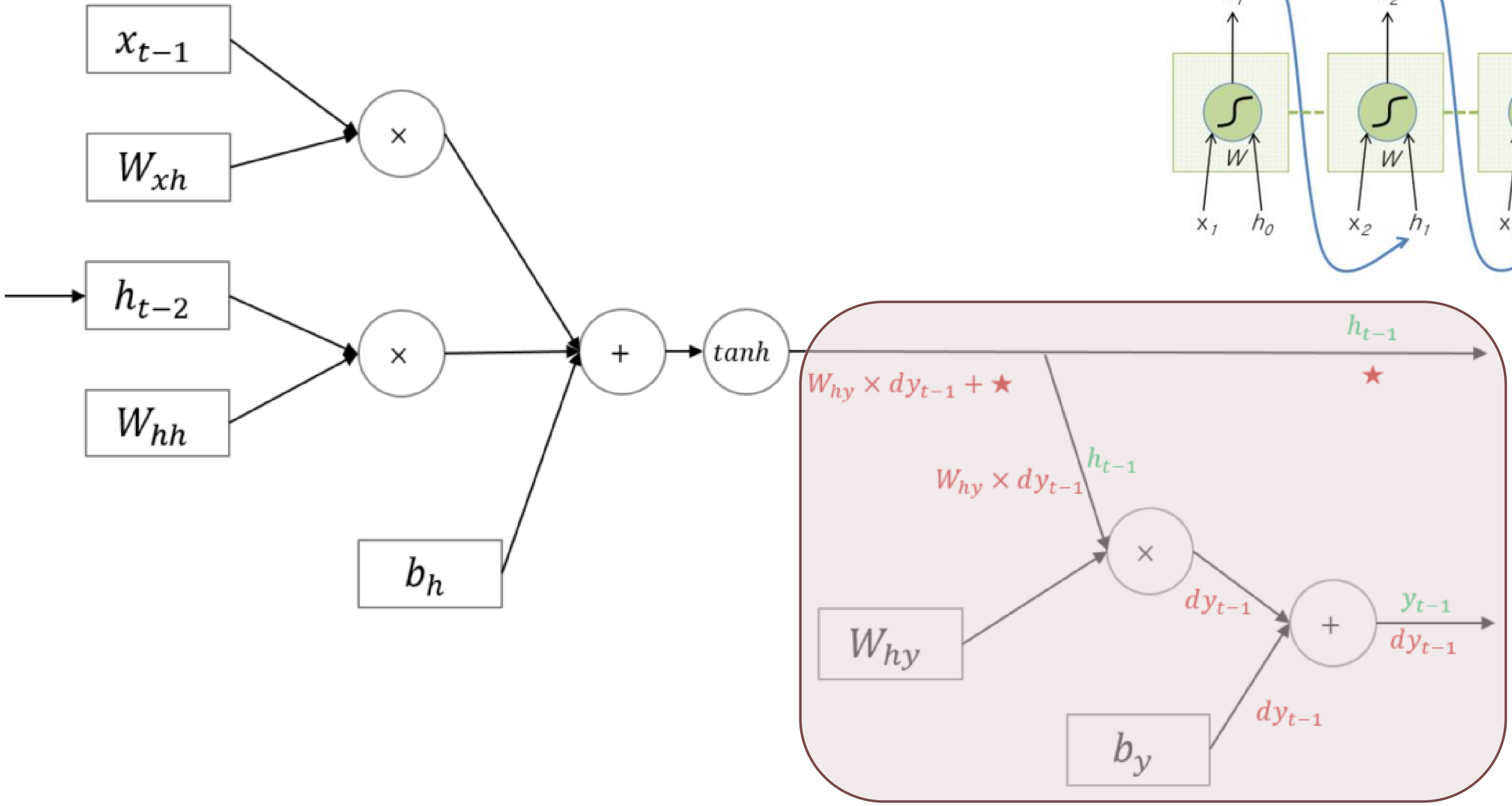


$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights



모든 경로의 derivative를 곱해서 더함

RNN - Applications

- Input과 Output의 길이가 가변적이므로 다양한 활용이 가능

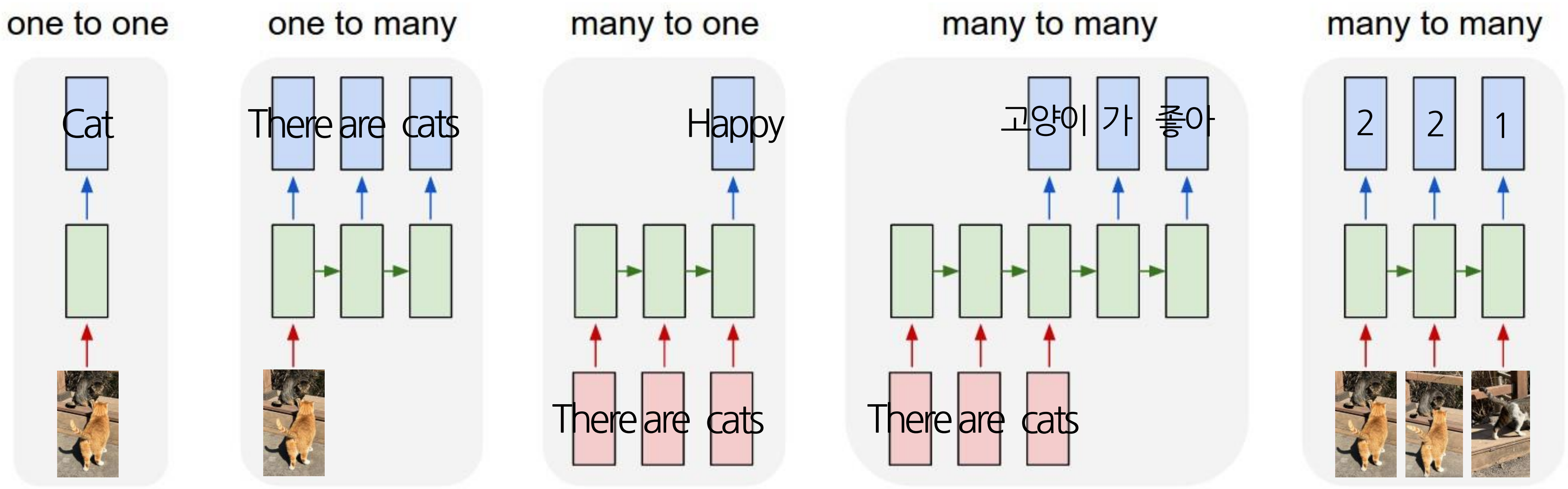


image classification | Image captioning | Sentimental analysis | Machine translation | Video Classification

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

- Example - Minimal character-level language model (The Little Prince)

90547 Characters, 80 Unique

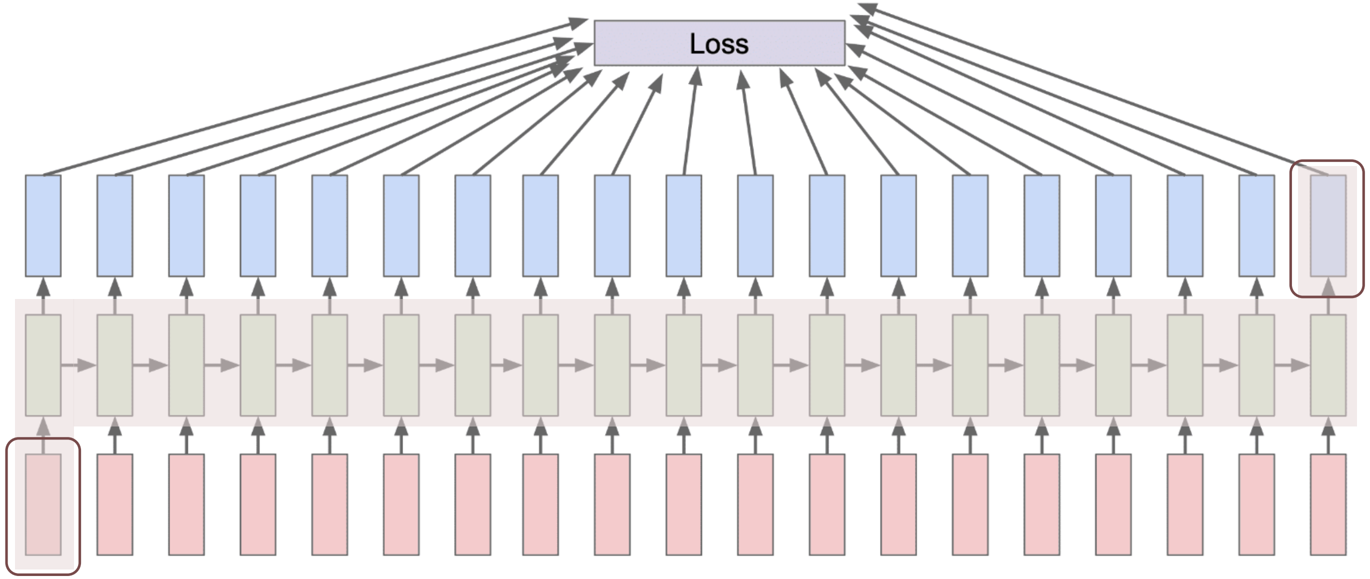
```
----
 !PweW-4‘!2y7r YuD3Ev"sSefyj$KHW“sP“T9EUAYNq)BT2WaHBjpporE7rCm.D zanzbchlj$h3,ek—b0‘yKpG-Tx5"5Exh fz—jVi$$Z’U'0xh)FI LRh"w0G
hW s8‘TFHg1f”arg5LV”p9AZ.q.NBvrr“fjO!4ft-3(A""g"eeB:illE'y!Nb\spUF7kk‘)oL—!\
----
iter 0, loss: 109.550664
```

```
----
 t deed seece,..

“I
 hom, insed tasly.

“Oner, is il le ind sas yowe the junco hame the was it are. kne ad at That ang they wn of a ctaide beake.

“Thay laosy shao ave revker ast foubbet if a cee ghec
----
iter 10000, loss: 54.261064
```

```
----
 n,” the long muke you dot shimpsik to is waughe thowen’twambabge the was mollije, coad them id as tarning
you stime fout tho kill, my wass. Mins,” serest-ming to he and whey to he aldile, in you corsp
----
iter 50000, loss: 41.201140
```

RNN - BPTT의 한계

- 모든 Time step마다 Unfold된 RNN Cell의 처음부터 끝까지 연산해야 함

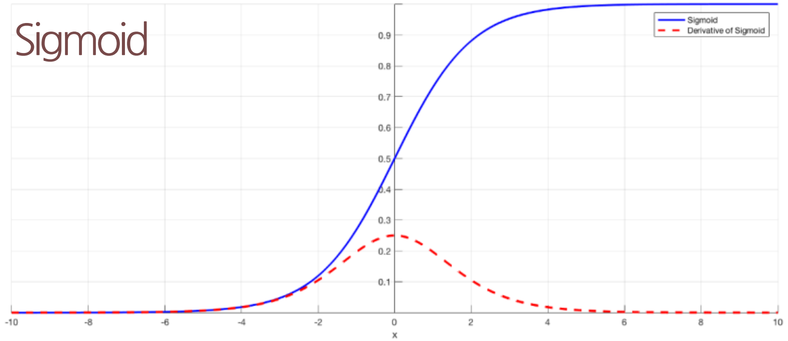


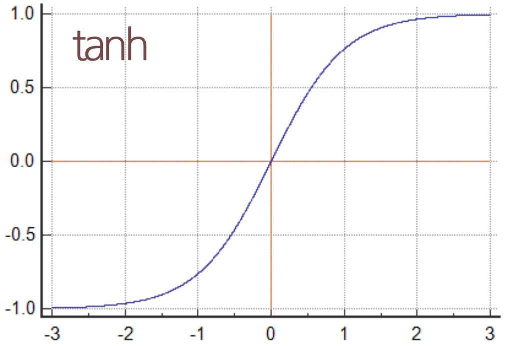
Gradient Clipping (Heuristic)

→ Gradients vanishing(or exploding) problem

→ Memory에 저장해야 하는 값이 많음

→ 연산이 많아서 학습 시간이 오래 걸림



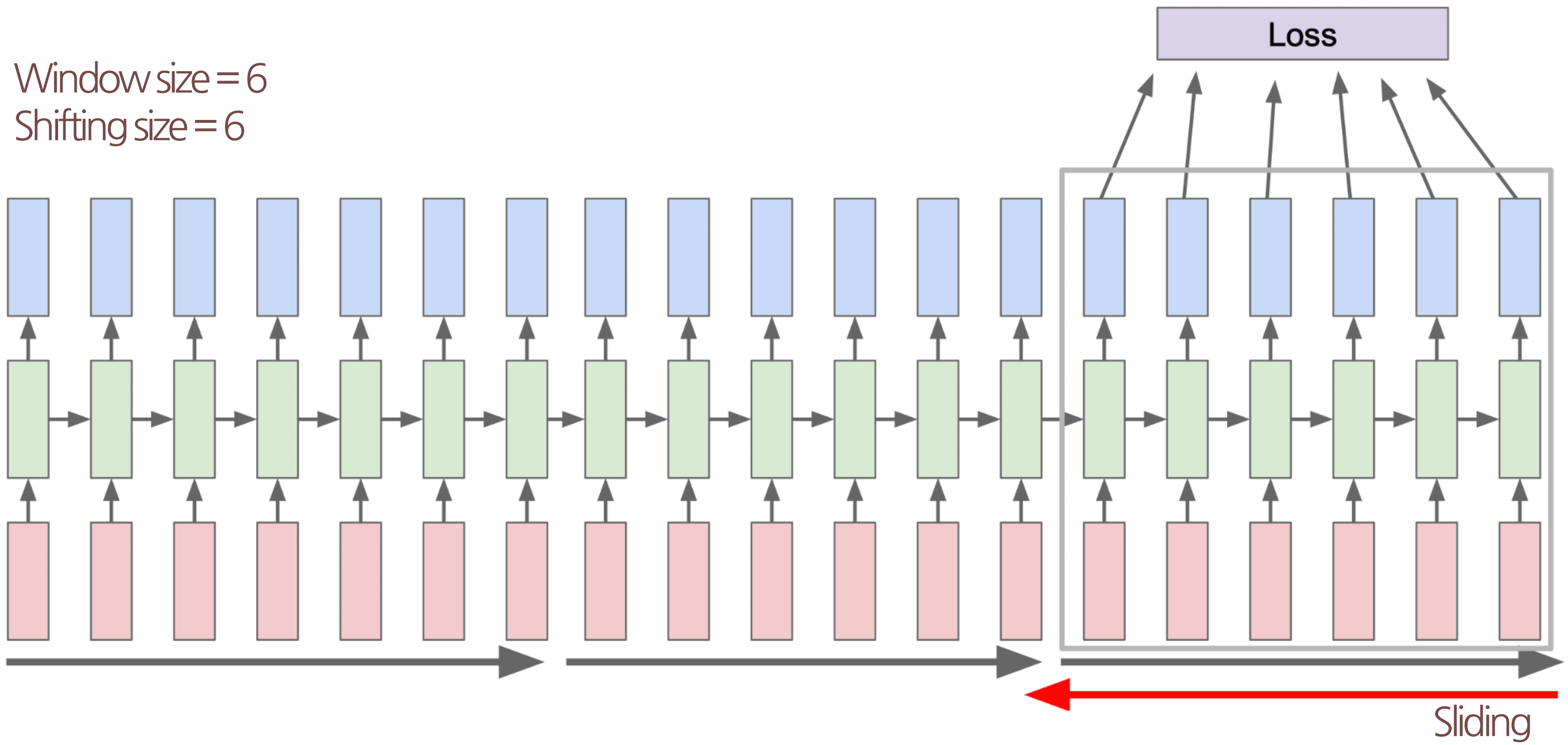


https://velog.io/@jinseock95/%EB%B6%80%EC%8A%A4%ED%8A%B8%EC%BA%A0%ED%94%84-AI-TechWeek03Day-14-Recurrent-Neural-Networks
https://needjarvis.tistory.com/564

RNN - Truncated BPTT

- Time step을 일정 크기로 나누어서 Backpropagation
- True Truncated BPTT는 Shifting size = 1인 경우를 의미

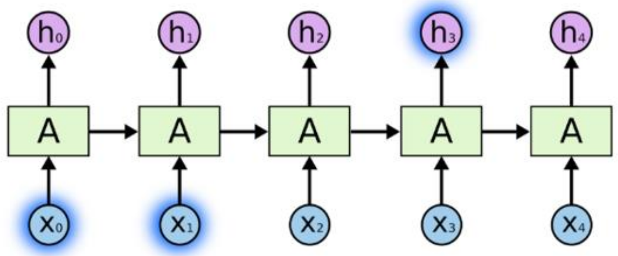
Window size = 6
Shifting size = 6

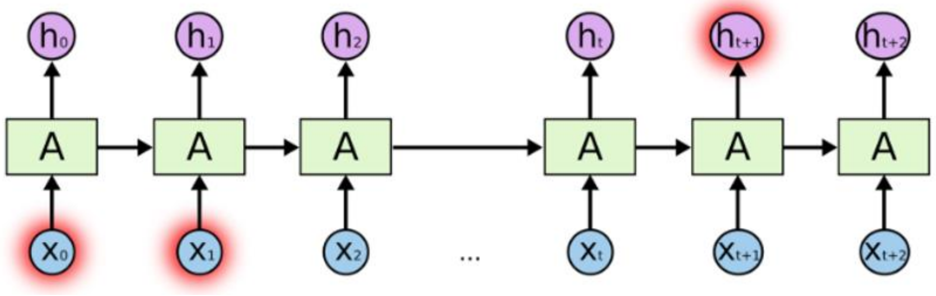


https://velog.io/@jinseock95/%EB%B6%80%EC%8A%A4%ED%8A%B8%EC%BA%A0%ED%94%84-AI-TechWeek03Day-14-Recurrent-Neural-Networks
https://needjarvis.tistory.com/564

RNN - Discussion

- 앞선 방법으로는 Long-term dependency problem이 완전히 해결되지 않음
- 거리가 먼 정보는 반영되기 어려워 학습 능력이 저하됨





$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right) = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\boxed{\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)}$$

Constant Error Flow 필요

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

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

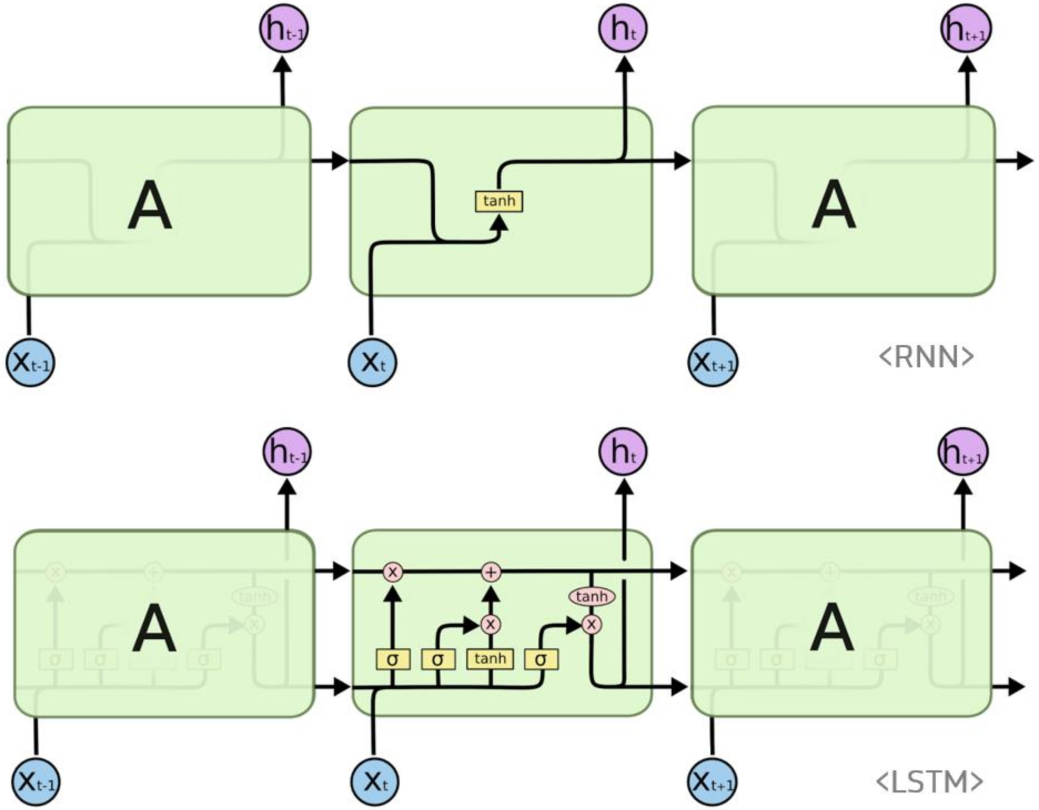
https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

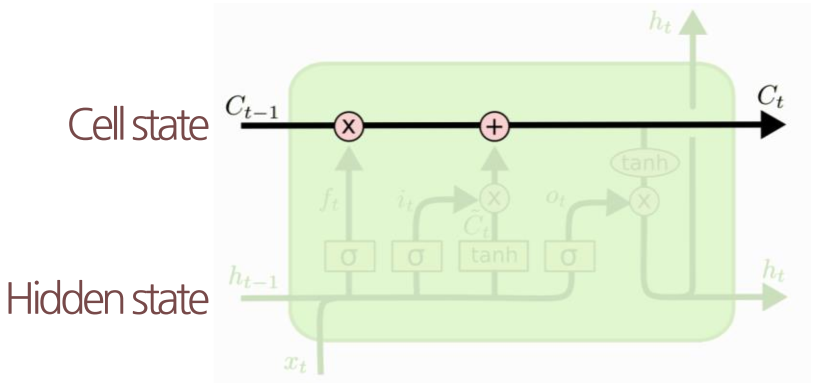
3

LSTM

LONG SHORT-TERM MEMORY

- 단기 기억을 위한 Hidden state 이외에 장기 기억을 위한 Cell state를 도입

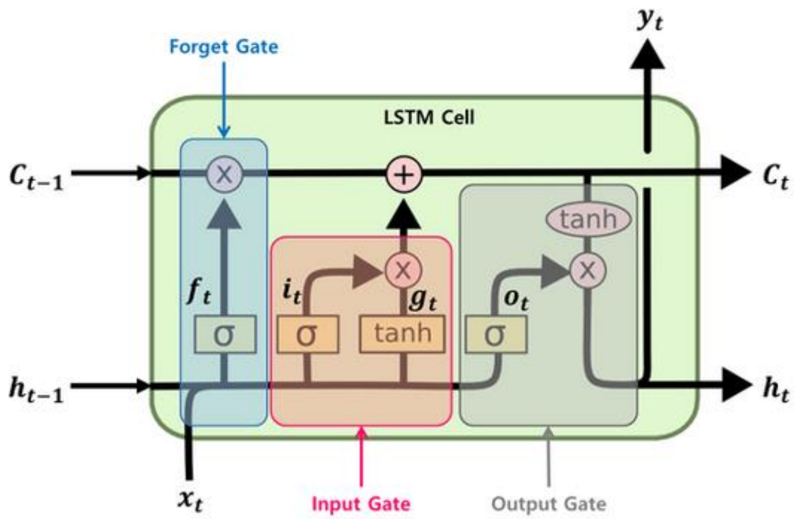




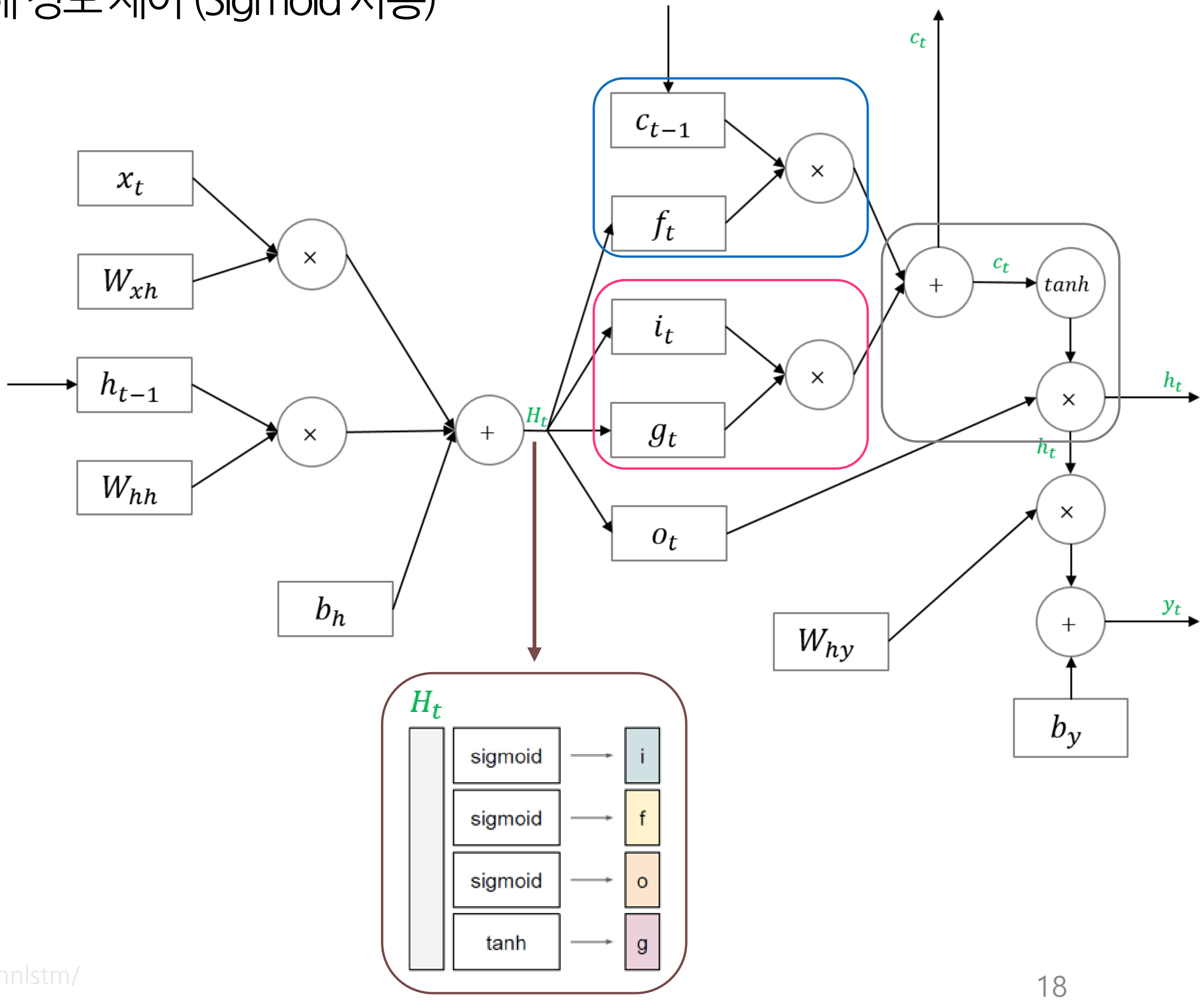
https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

LSTM - LSTM Cell Forward Pass

- Forget gate, input gate, output gate를 통해 정보 제어 (Sigmoid 사용)

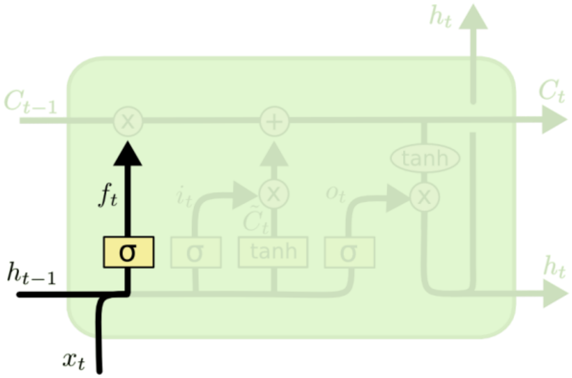


$$
\begin{aligned}
f_t &= \sigma(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f}) \\
i_t &= \sigma(W_{xh\_i}x_t + W_{hh\_i}h_{t-1} + b_{h\_i}) \\
o_t &= \sigma(W_{xh\_o}x_t + W_{hh\_o}h_{t-1} + b_{h\_o}) \\
g_t &= \tanh(W_{xh\_g}x_t + W_{hh\_g}h_{t-1} + b_{h\_g}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$



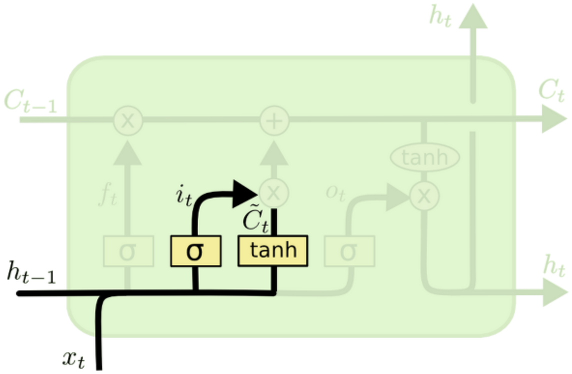
https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

- Forget gate - Cell state로부터 어떤 정보를 잊을 것인가



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

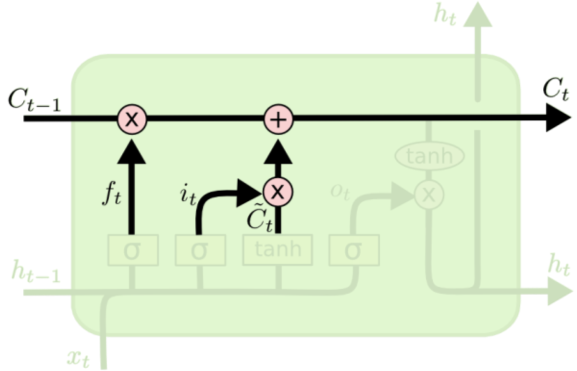
- Input gate - 새로운 정보 중 어떤 것을 Cell state에 저장할 것인가



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

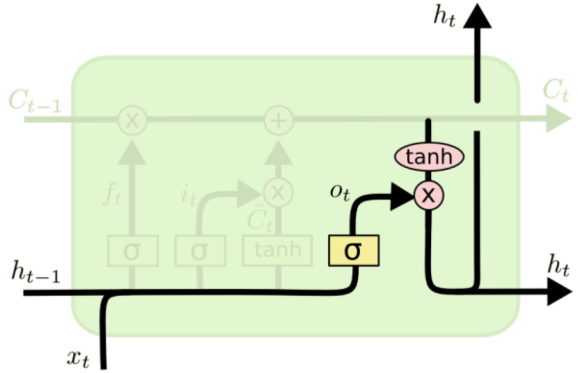
https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

- Cell state update - Forget gate와 Input gate를 적용



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

- Output gate - Cell state 중 어느 부분을 Output으로 내보낼 것인가

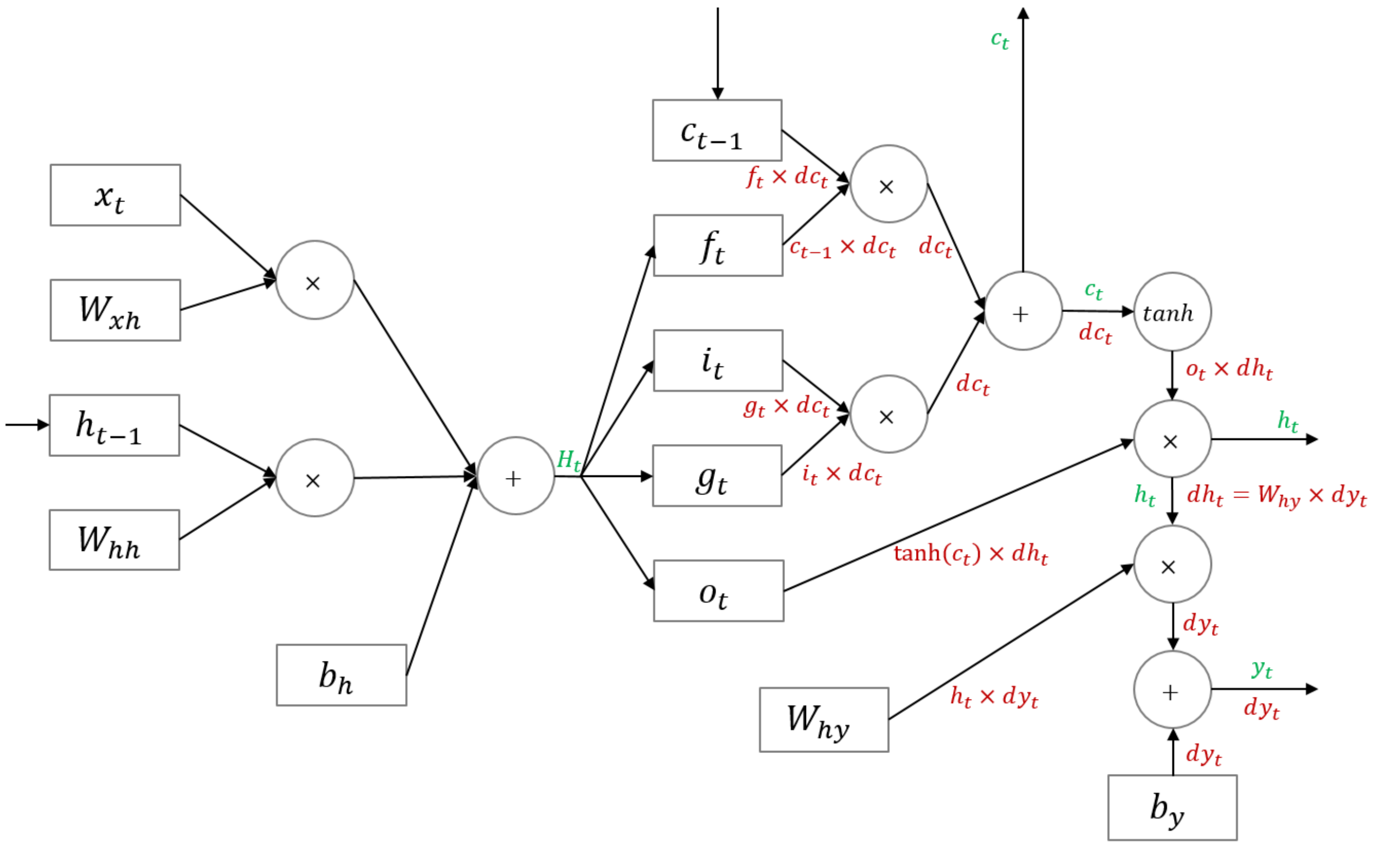


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

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

LSTM - LSTM Cell Backpropagation

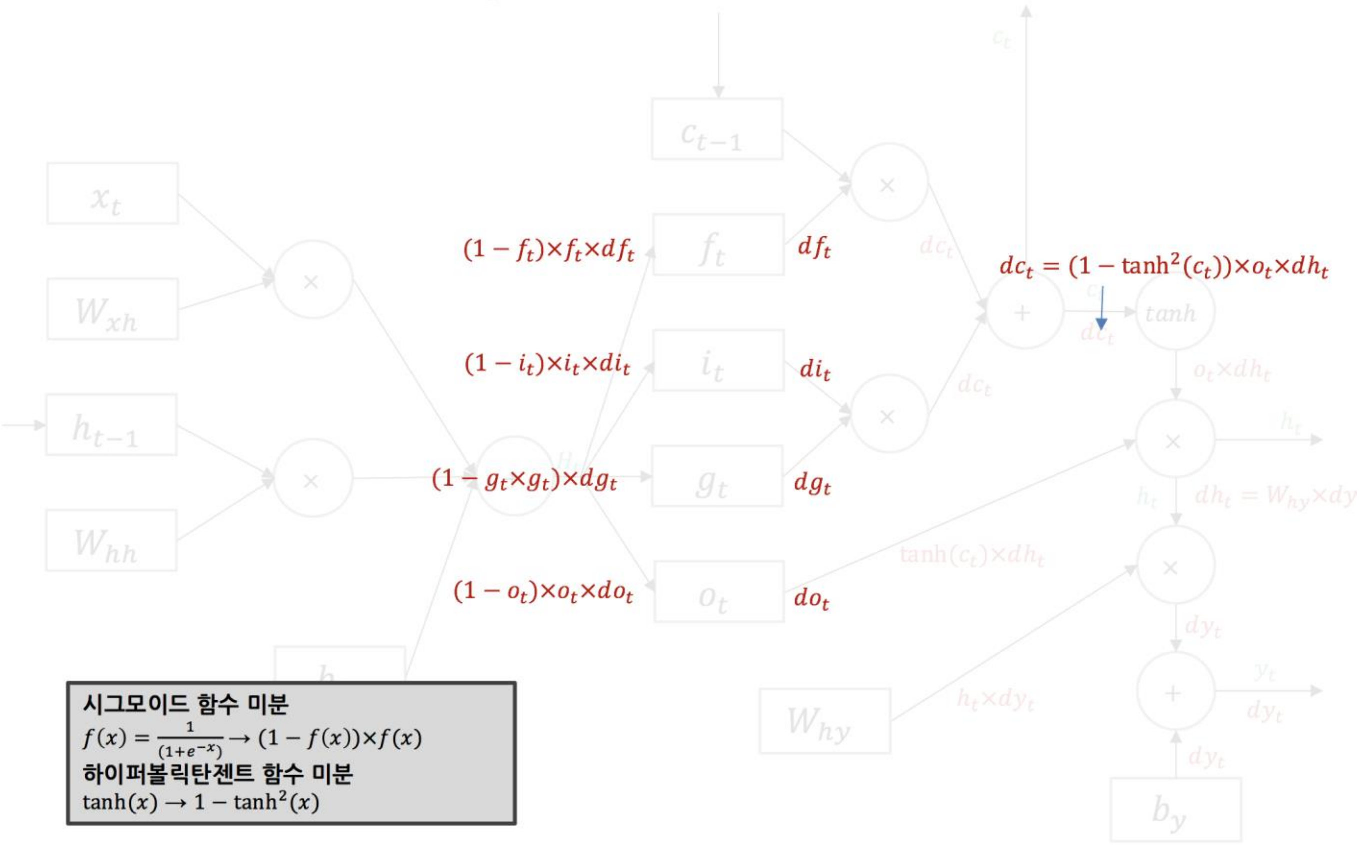
- RNN 에서의 Backpropagation 과 유사



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

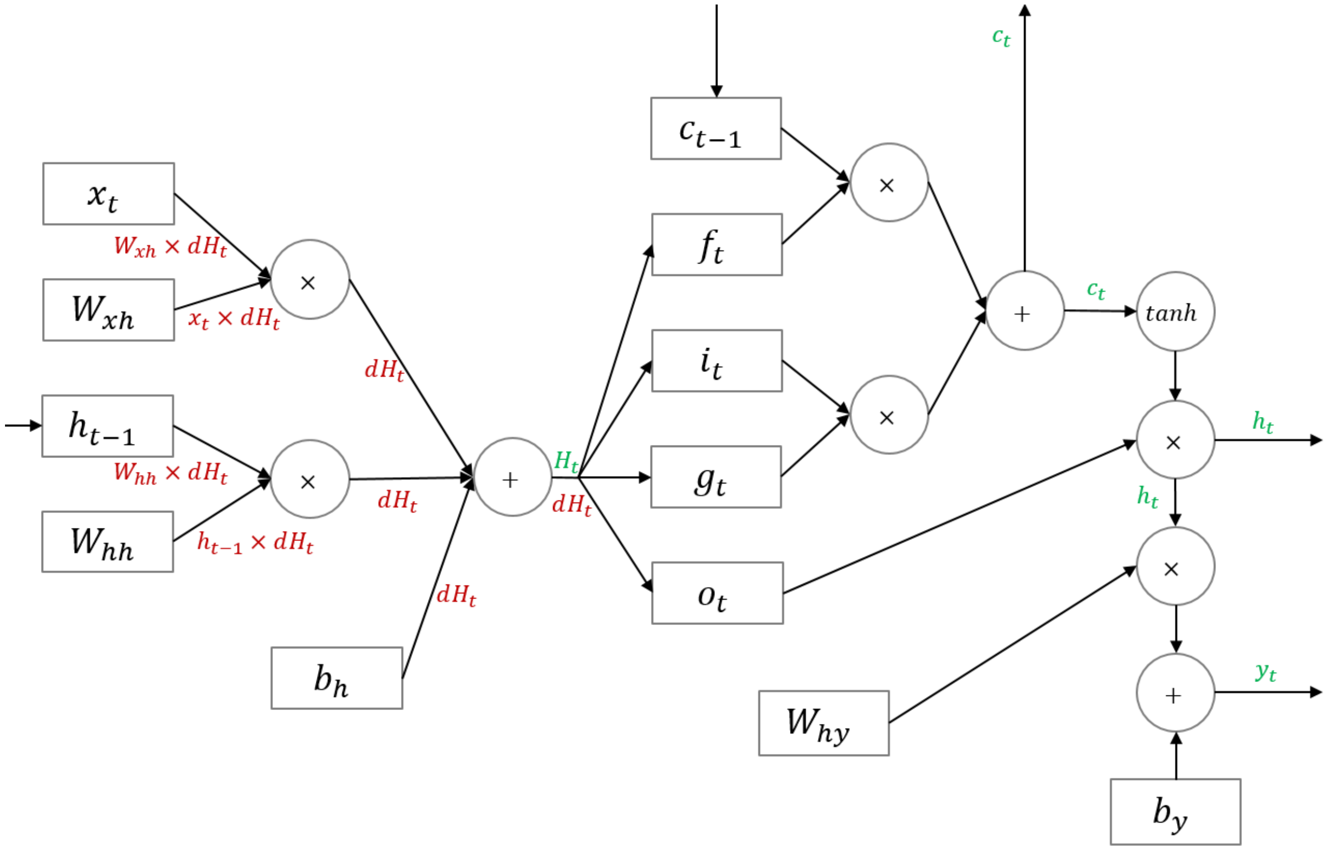
LSTM - LSTM Cell Backpropagation

- RNN 에서의 Backpropagation 과 유사



LSTM - LSTM Cell Backpropagation

- RNN 에서의 Backpropagation 과 유사 Details → http://arunmallya.github.io/writeups/nn/lstm/index.html#/

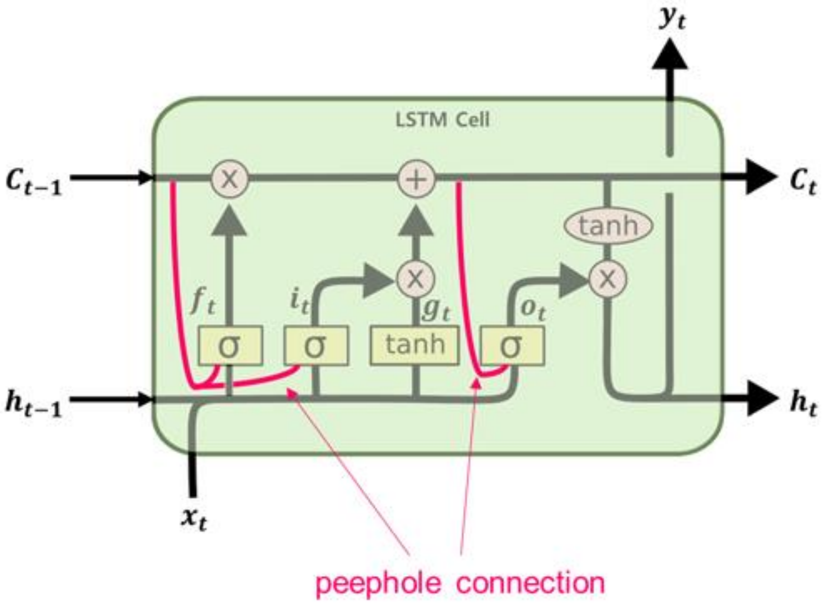


https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

LSTM - Peephole connection

DNN
RNN
LSTM
GRU

- F. Gers and J.Schmidhuber (2000)
- 이전 Time step의 Cell state가 입력으로 추가되어 보다 많은 Context 인식



$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{cf}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xf}^T \cdot \mathbf{x}_t + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right)$$
$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{ci}^T \cdot \mathbf{c}_{t-1} + \mathbf{W}_{xi}^T \cdot \mathbf{x}_t + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right)$$
$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{co}^T \cdot \mathbf{c}_t + \mathbf{W}_{xo}^T \cdot \mathbf{x}_t + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right)$$

https://excelsior-cjh.tistory.com/185

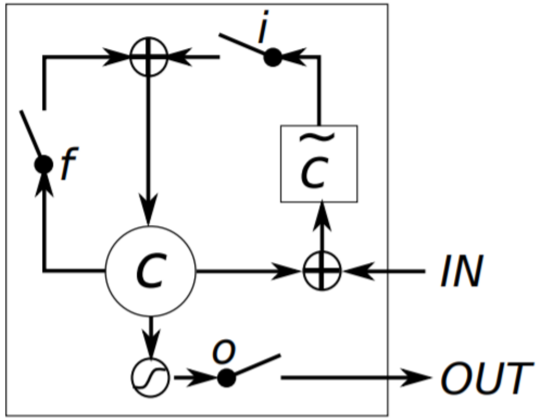
4

GRU

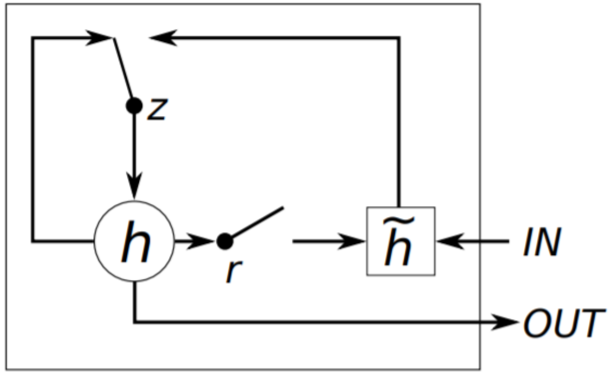
GATED RECURRENT UNIT

GRU - Gated Recurrent Unit

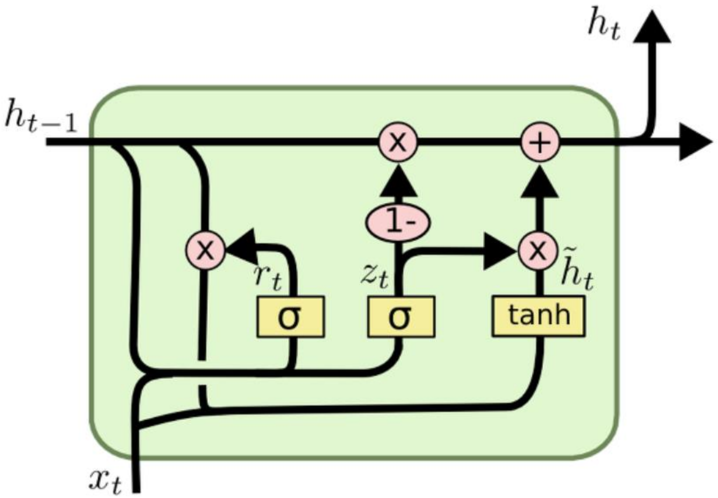
- KyungHyun C. et al. (2014)
- 기존 LSTM 에서 Cell state 와 Hidden state 가 하나로 통합되어 저장됨
- LSTM Cell의 Forget gate, input gate, output gate 대신 Update gate, reset gate 제시



(a) Long Short-Term Memory

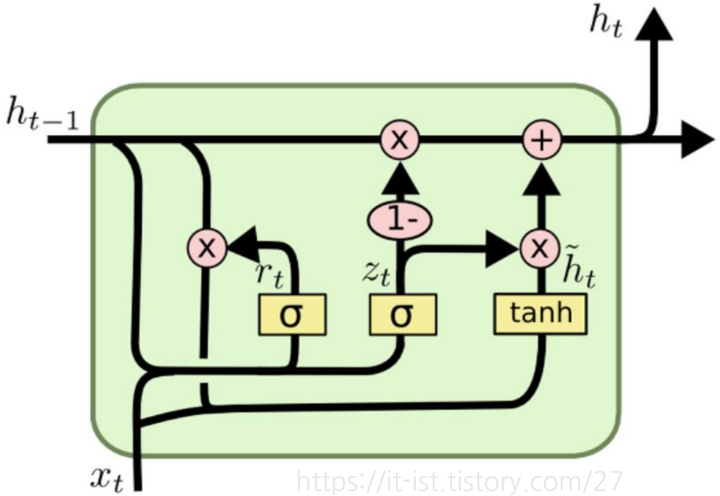


(b) Gated Recurrent Unit



GRU - Gated Recurrent Unit

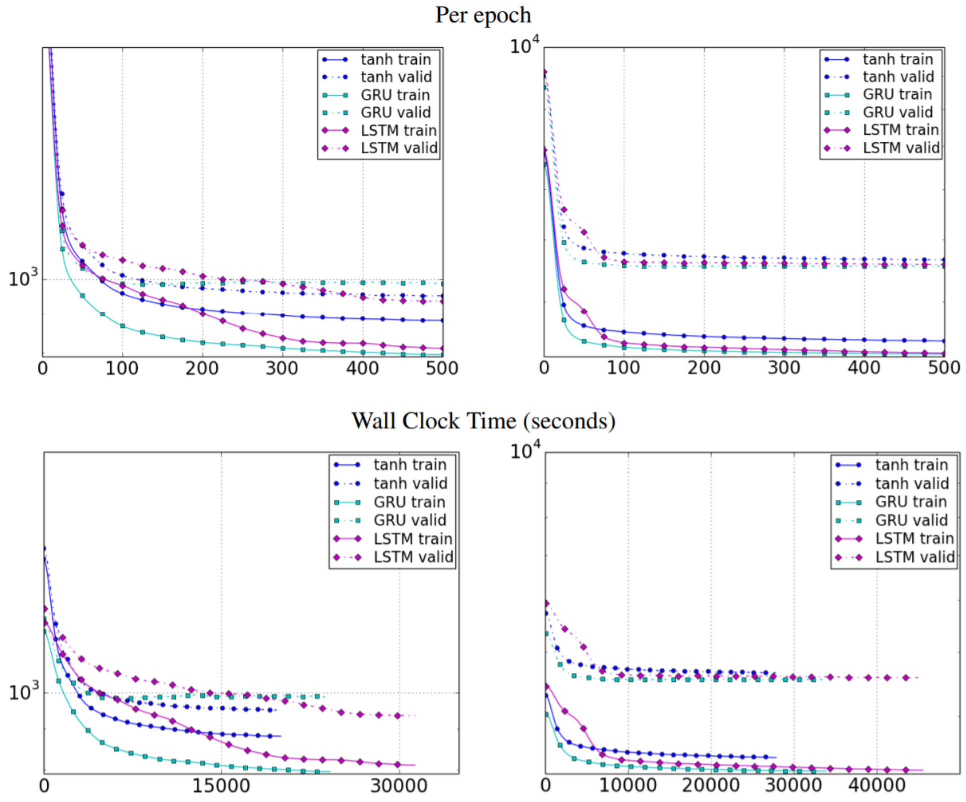
- Update gate (z_t) : Take a linear sum between the existing state and the newly computed state
- Reset gate (r_t) : If it is reading the first symbol of an input sequence, allow it to forget
- GRU에는 LSTM의 Output gate mechanism이 존재하지 않고 r_t 가 Sequence의 시작만 구별



$$
\begin{aligned}
h_t^j &= (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j \\
z_t^j &= \sigma \left(W_z \mathbf{x}_t + U_z \mathbf{h}_{t-1} \right)^j \\
\tilde{h}_t^j &= \tanh \left(W \mathbf{x}_t + U \left(\mathbf{r}_t \odot \mathbf{h}_{t-1} \right) \right)^j \\
r_t^j &= \sigma \left(W_r \mathbf{x}_t + U_r \mathbf{h}_{t-1} \right)^j
\end{aligned}
$$

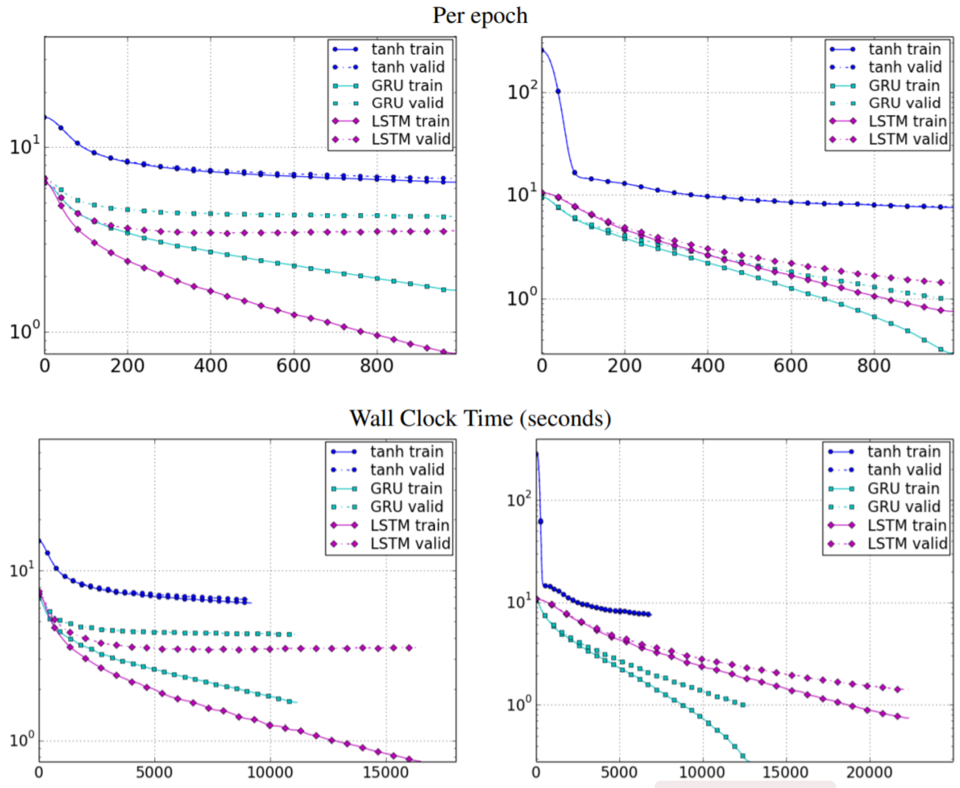
GRU - Gated Recurrent Unit

- Traditional RNN 보다 나은 성능, LSTM과 비교해서는 Dataset의 종류에 영향을 받음
- 데이터 양이 적을 때 강세를 보임 (Ubisoft Dataset B)



(a) Nottingham Dataset (b) MuseData Dataset

Polyphonic music modeling



(a) Ubisoft Dataset A (b) Ubisoft Dataset B

Speech signal modeling