Recurrent Neural Networks (RNN)

Seoul National University

Saerom Park

drsaerompark@gmail.com

References

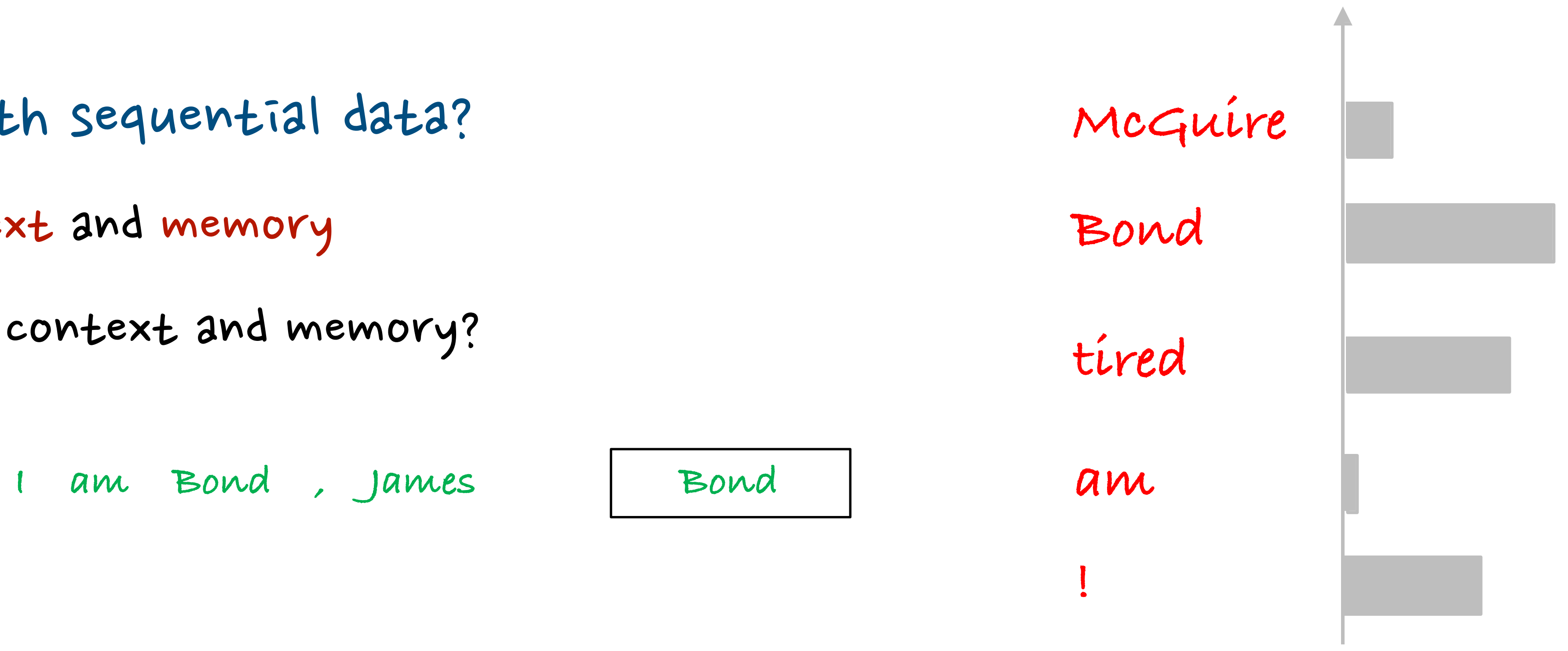
- Books
  - [Goodfellow et al. (2016)] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning, The MIT Press, 2016.
- Online courses
  - [Hinton. (2015)] G. Hinton, Online course on Neural Networks for Machine Learning
    - ✓ https://www.coursera.org/learn/neural-networks
  - [Larochelle. (2014)] H. Larochelle, Online course on Neural Networks
    - ✓ http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html
  - [Gavves. (2017)] E. Gavves, UvA Deep Learning course
    - ✓ http://uvadlc.github.io/
  - Borrows slides (some modified) from Larochelle & Gavves

Lecture Overview

- Recurrent Neural Networks (RNN) for sequences
- Backpropagation Through Time (bPTT)
- Vanishing and Exploding Gradients and Remedies
- RNNs using Long Short-Term Memory (LSTM)
- Applications of RNNs

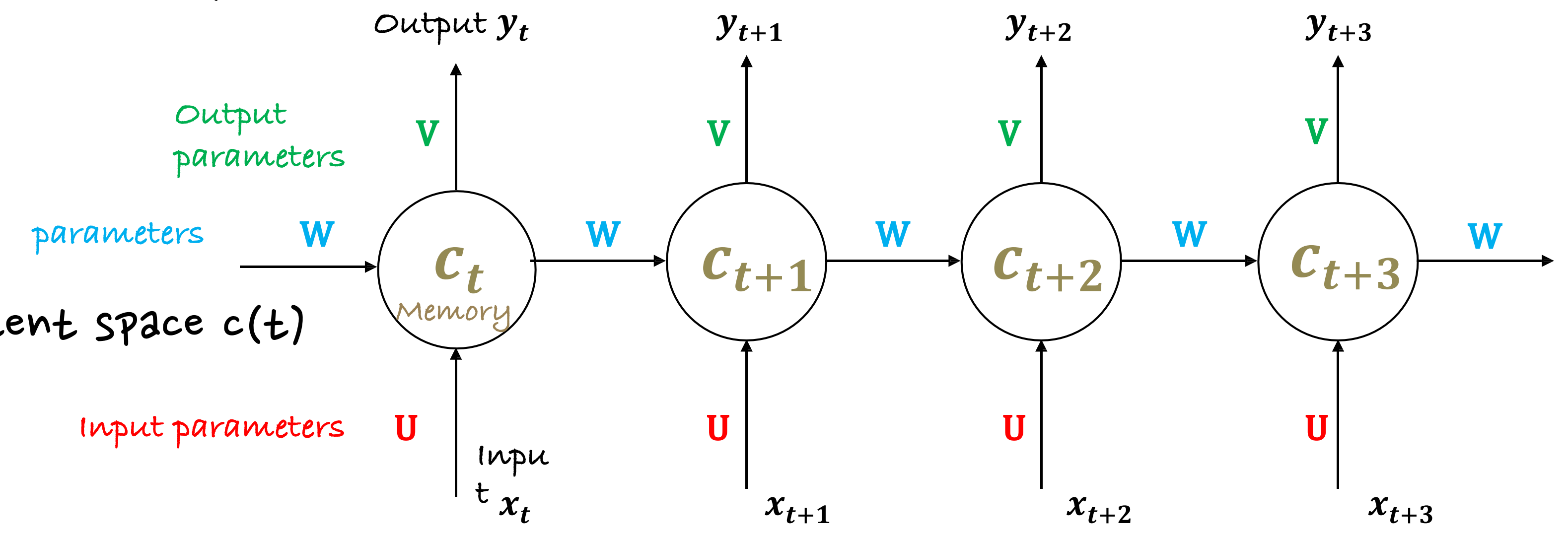
Sequential Data

- Property
  - Next data depend on previous data
  - Data inside a sequence are non i.i.d. (identically, independently distributed)
- Example
  - The next "word" depends on the previous "words"
- How to deal with sequential data?
  - We need context and memory
  - How to model context and memory?



Sequential Data

- **Main task**
  - Roughly equivalent to predicting what comes next: $\Pr(x) = \prod_i \Pr(x_i | x_1, \dots, x_{i-1})$
- **Question**
  - What about inputs that appear in sequences, such as text?
  - Could a neural network handle such modalities?
  - How to deal with the different lengths of sequences?
- **What is memory?**
  - Representation of the past
  - Information at time step t on a latent space c(t)

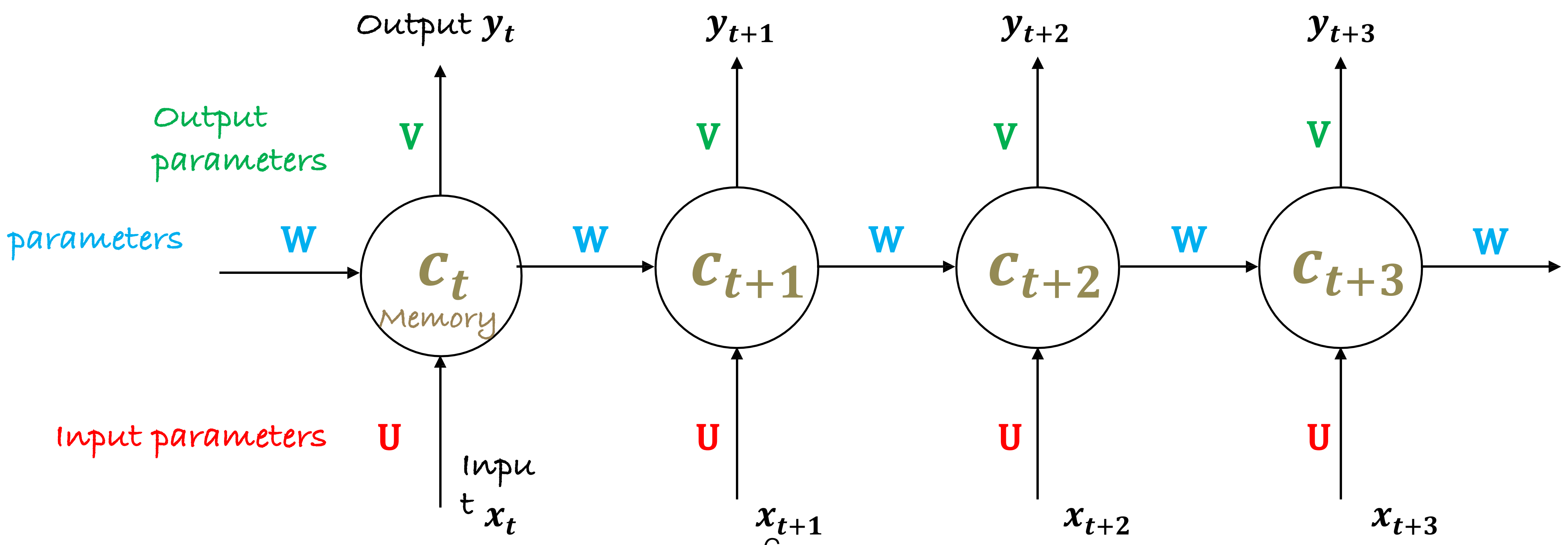


- **Model**
  - Project information c(t) using parameters θ : $c_{t+1} = h(x_{t+1}, c_t; \theta)$
  - Share parameters θ for all time steps: $c_{t+1} = h(x_{t+1}, h(x_t, h(x_{t-1}, \dots, h(x_1, c_0; \theta); \theta); \theta); \theta)$

Memory

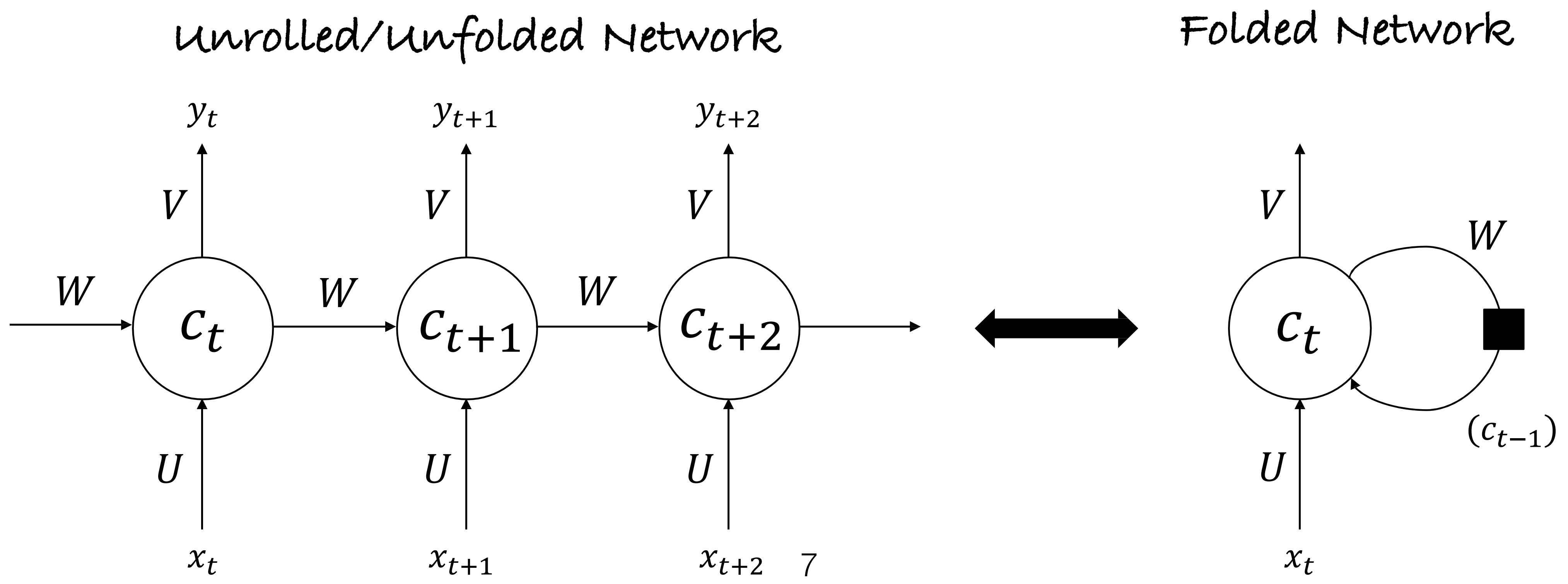
- What is memory?
  - Representation of the past
  - Information at time step t on a latent space c(t)

- Model
  - Project information c(t) using parameters θ : $c_{t+1} = h(x_{t+1}, c_t; \theta)$
  - Share parameters θ for all time steps: $c_{t+1} = h(x_{t+1}, h(x_t, h(x_{t-1}, \dots, h(x_1, c_0; \theta); \theta); \theta); \theta)$



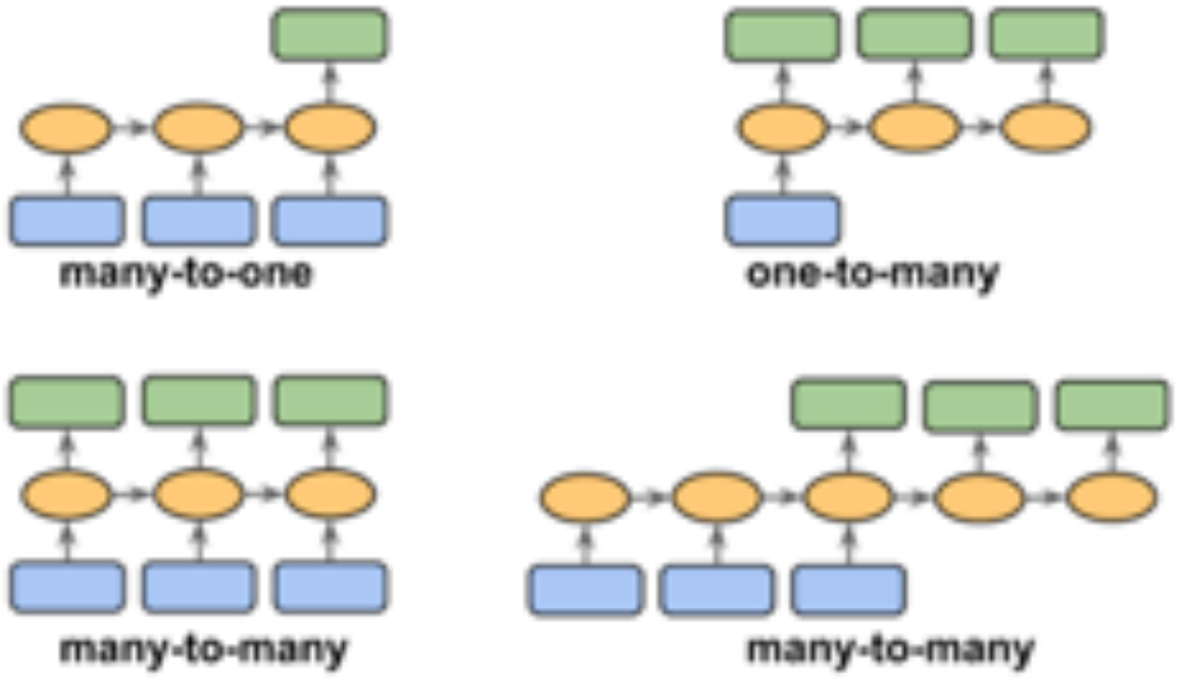
Recurrent Neural Network

- Folding the memory
  - Simplify the repeated parameters and structure
- Only two equations
  - Memory (cell) $c_t = \tanh(b + U\, x_t + W\, c_{t-1})$
  - Output $y_t = \mathrm{softmax}(a + V\, c_t)$



Introducing Sequential Data

- The different categories of sequence modeling
  - Many-to-one: 입력 데이터는 sequence이지만 출력은 고정 크기의 벡터
    - ✓ Sentiment Analysis
  - One-to-many: 입력 데이터는 고정 크기의 벡터, 출력은 sequence
    - ✓ Image captioning
  - Many-to-Many: 입력 및 출력이 모두 sequence
    - ✓ eg. video classification, translation



RNN Example

- component sizes
  - input size: 5
  - memory size: 3 (3 units)
  - output size: 10
- Parameters
  - cell size (c(t))?
  - input projection (U)?
  - cell projection (W)?
  - output projection (V)?

$$c_t = \tanh(U\, x_t + W\, c_{t-1})$$
$$y_t = \mathrm{softmax}(V\, c_t)$$

RNN vs. MLP?

- What is really different?
  - 입력 (input) 데이터가 시계열 또는 일반적인 시퀀스 데이터임
  - 현재 시점의 예측 값을 계산할 때에 이전 시간의 정보를 함께 사용함
  - RNN은 입력 데이터 자체가 시퀀스 데이터이고 마지막 시점에서만 output을 예측하도록 구성하는 것도 가능함
- RNN 학습
  - MLP 의 Backpropagation 과 본질적으로 동일
  - 차이는 RNN의 구조가 시간에 따라 연결되어 있기 때문에 Backprop 이 시간을 거슬러 올라가며 적용된다는 것 뿐임

Training RNNS

- Loss Function
  - classification: cross-entropy loss

$$P = \prod_{t,k} y_{tk}^{l_{tk}} \Longrightarrow \mathcal{L} = -\log P = \sum_t \mathcal{L}_t = -\frac{1}{T}\sum_t \sum_k l_{tk} \log y_{tk}$$

- Backpropagation Through Time (BPTT)
  - Again, chain rule!
  - Only differenceL gradients survive over time steps
  - We have three parameter (W,V,U): $\frac{\partial \mathcal{L}}{\partial V}, \frac{\partial \mathcal{L}}{\partial W}, \frac{\partial \mathcal{L}}{\partial U}$

Backpropagation Through Time: An Example

- Gradients for parameters
  - Step by step explanation
    - ✓ http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

$$P = \prod_{t,k} y_{tk}^{l_{tk}} \Longrightarrow \mathcal{L} = -\log P = \sum_t \mathcal{L}_t = -\frac{1}{T}\sum_t \sum_k l_{tk} \log y_{tk}$$

$$\boxed{\begin{aligned} c_t &= \tanh(b + Ux_t + Wc_{t-1}) \\ y_t &= \text{softmax}(a + Vc_t) \\ \mathcal{L} &= -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t \end{aligned}}$$

$$\begin{tikzcd}
 & y_{\tau-2}, \mathcal{L}_{\tau-2} & y_{\tau-1}, \mathcal{L}_{\tau-1} & y_\tau, \mathcal{L}_\tau \\
{} \arrow[r, "W"] & c_{\tau-3} \arrow[u, "V"] \arrow[r, "W"] & c_{\tau-1} \arrow[u, "V"] \arrow[r, "W"] & c_\tau \arrow[u, "V"] \\
 & x_{\tau-2} \arrow[u, "U"] & x_{\tau-1} \arrow[u, "U"] & x_\tau \arrow[u, "U"]
\end{tikzcd}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial c_t} = W^T \text{diag}(1 - c_{t+1}^2)\frac{\partial \mathcal{L}}{\partial c_{t+1}} + V^T(y_t - l_t)} \quad \boxed{\frac{\partial \mathcal{L}_\tau}{\partial c_\tau} = V^T(y_\tau - l_\tau)}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial U} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial U_t} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial c_t}\left(\frac{\partial c_t}{\partial U_t}\right)^T = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial c_t} \cdot diag(1 - c_t^2) \cdot x_t^T}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial W_t} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial c_t}\left(\frac{\partial c_t}{\partial W_t}\right)^T = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial c_t} \cdot \text{diag}(1 - c_t^2) \cdot c_{t-1}^T}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial b} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial b_t} = \sum_{t=1}^{\tau} \left(\frac{\partial c_t}{\partial b_t}\right)^T \frac{\partial \mathcal{L}}{\partial c_t} = \sum_{t=1}^{\tau} \text{diag}(1 - c_t^2) \cdot \frac{\partial \mathcal{L}}{\partial c_t}}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial V} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial \mathcal{L}_t}\frac{\partial \mathcal{L}_t}{\partial \alpha_t}\frac{\partial \alpha_t}{\partial V} = (l_t - y_t) \cdot \frac{\partial (Vc_t)}{\partial V} = \sum_{t=1}^{\tau} (y_t - l_t) \cdot c_t^T}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial a} = \sum_{t=1}^{\tau} \frac{\partial \mathcal{L}}{\partial \mathcal{L}_t}\left(\frac{\partial \alpha_t}{\partial a}\right)^T \frac{\partial \mathcal{L}_t}{\partial \alpha_t} = \sum_{t=1}^{\tau} \boldsymbol{I} \cdot (y_t - l_t) = \sum_{t=1}^{\tau} (y_t - l_t)}$$

Challenge of Long-term Dependencies

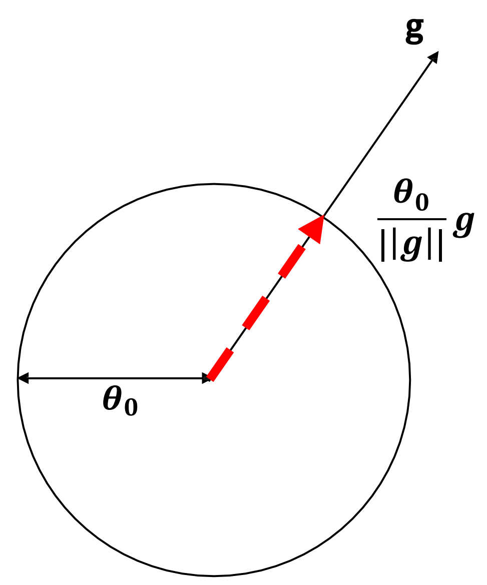
- Vanishing gradients
  - After a few time steps the gradients become almost 0
- Exploding gradients
  - After a few time steps the gradients become huge
- Can't capture long-term dependencies
  - To make it simpler, assume that $c_t = W \cdot c_{t-1}$ and W admits an Eigen-decomposition of the form $W = V \Lambda V^{-1}$, then

$$c_t = W \cdot c_{t-1} = W^t \cdot c_0 = V \Lambda^t V^{-1} \cdot c_0$$

✓ The eigenvalues are raised to the power of t causing eigenvalues with magnitude less than one to decay to zero and eigenvalues with magnitude greater than one to explode. Any component of c_0 that is not aligned with the largest eigenvector will eventually be discarded.

How to solve these problem?

- Exploding gradients
  - Scale the gradients to a threshold

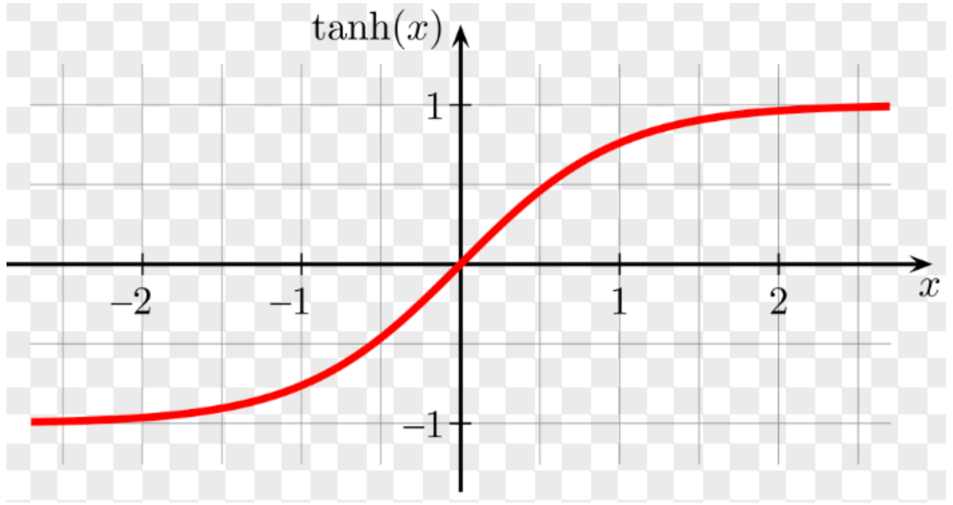


- Vanishing gradients
  - It can make long-term dependencies negligible
  - Learning focuses on the short-term only
  - Difficult to detect!

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{s=1}^{t} \frac{\partial \mathcal{L}_r}{\partial c_t}\frac{\partial y_t}{\partial c_t}\frac{\partial c_t}{\partial c_s}\frac{\partial c_s}{\partial W}$$

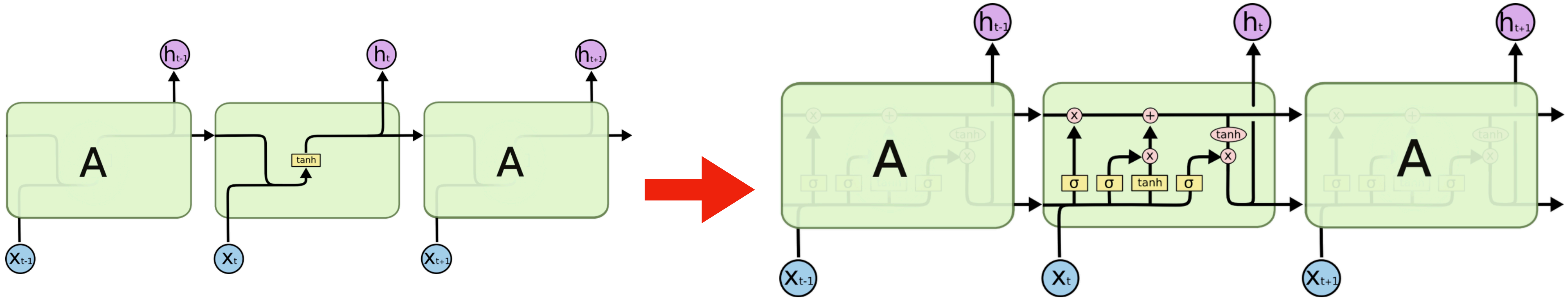
$$\frac{\partial c_t}{\partial c_s} = \prod_{t \geq k \geq s} \frac{\partial c_k}{\partial c_{k-1}} = \prod_{t \geq k \geq s} W \cdot \partial \tanh(c_{k-1})$$

- Advanced RNNs
  - Gradient 연산들이 곱셈이 아닌 더하기 연산으로 구성된 RNNs
  - Long-short term memory (LSTM) module
  - Gated recurrent unit (GRU) module



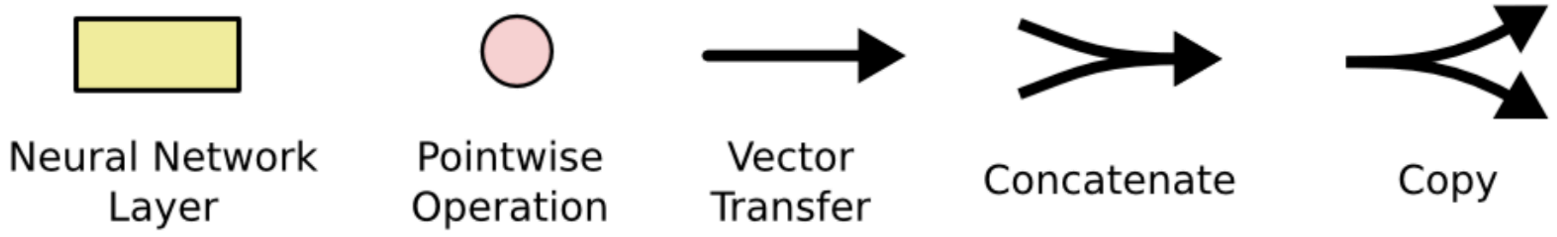
LSTM network

- RNN vs. LSTM structure



- LSTM 은 오차의 gradient 가 시간을 거슬러서 잘 흘러갈 수 있도록 도와줌
- Backprop 과정에서 오차의 값이 더 잘 유지되도록 함으로 long-term 정보도 더 잘 기억하도록 함
- 일반적인 RNN units 은 곱하기로만 이루어져 있는 반면 피드백을 더하기로 이음으로써 sigmoid 곱에 대한 gradient vanishing 문제 해결

- 기호



LSTM components

- 저장할 정보 (cell states)

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

- 새로운 정보
  - Forget gate layer: cell state에서 어떤 정보를 버릴지 (0~1)

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

  - Input gate layer: 어떤 값을 업데이트 할지 결정 (0~1)

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

  - cell state 에 더해질 수 있는 새로운 후보값 (-1~1)

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

- 출력
  - Output gate layer: 어떤 출력값을 출력할지 결정 (0~1)

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

  - 실제 출력값

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

LSTM components

- 저장할 정보 (cell states)

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

- 새로운 정보
  - Forget gate layer: cell state에서 어떤 정보를 버릴지 (0~1)

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

  - Input gate layer: 어떤 값을 업데이트 할지 결정 (0~1)

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

  - cell state 에 더해질 수 있는 새로운 후보값 (-1~1)

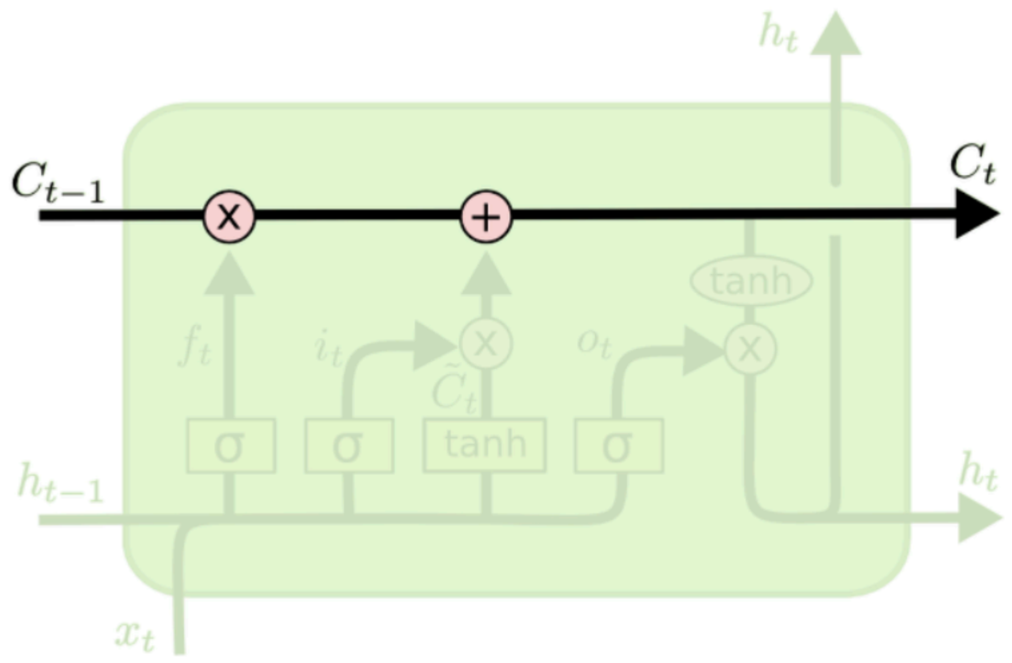
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- 출력
  - Output gate layer: 어떤 출력값을 출력할지 결정 (0~1)

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

  - 실제 출력값

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



LSTM components

- 저장할 정보 (cell states)

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

- 새로운 정보
  - Forget gate layer: cell state에서 어떤 정보를 버릴지 (0~1)

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

  - Input gate layer: 어떤 값을 업데이트 할지 결정 (0~1)

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

  - cell state 에 더해질 수 있는 새로운 후보값 (-1~1)

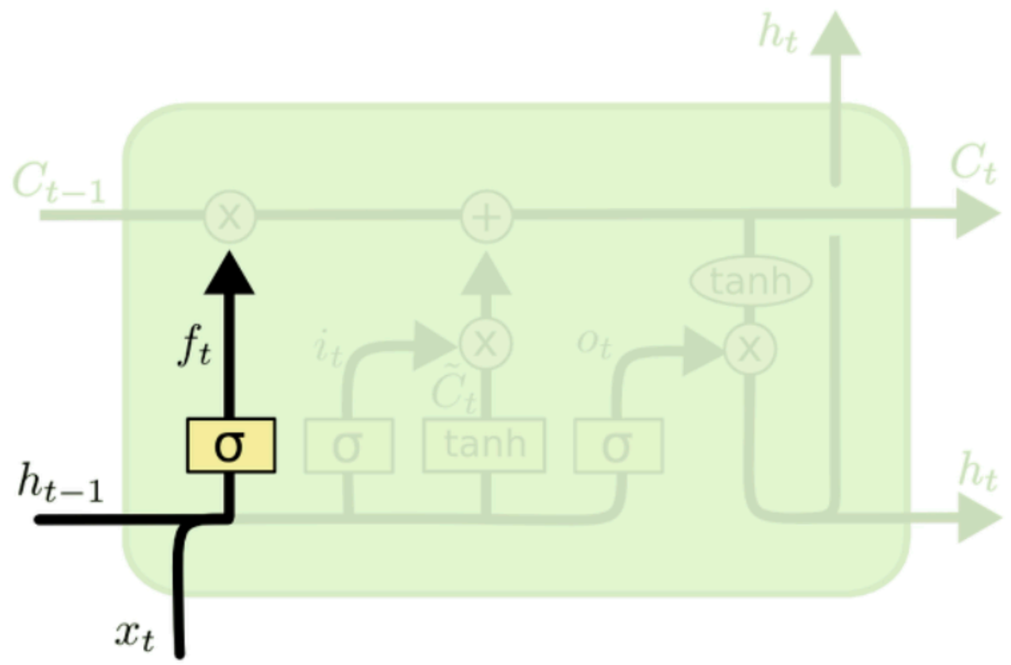
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- 출력
  - Output gate layer: 어떤 출력값을 출력할지 결정 (0~1)

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

  - 실제 출력값

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



LSTM components

- 저장할 정보 (cell states)

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

- 새로운 정보
  - Forget gate layer: cell state에서 어떤 정보를 버릴지c(0~1)

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

  - Input gate layer: 어떤 값을 업데이트 할지 결정 (0~1)

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

  - cell state 에 더해질 수 있는 새로운 후보값 (-1~1)

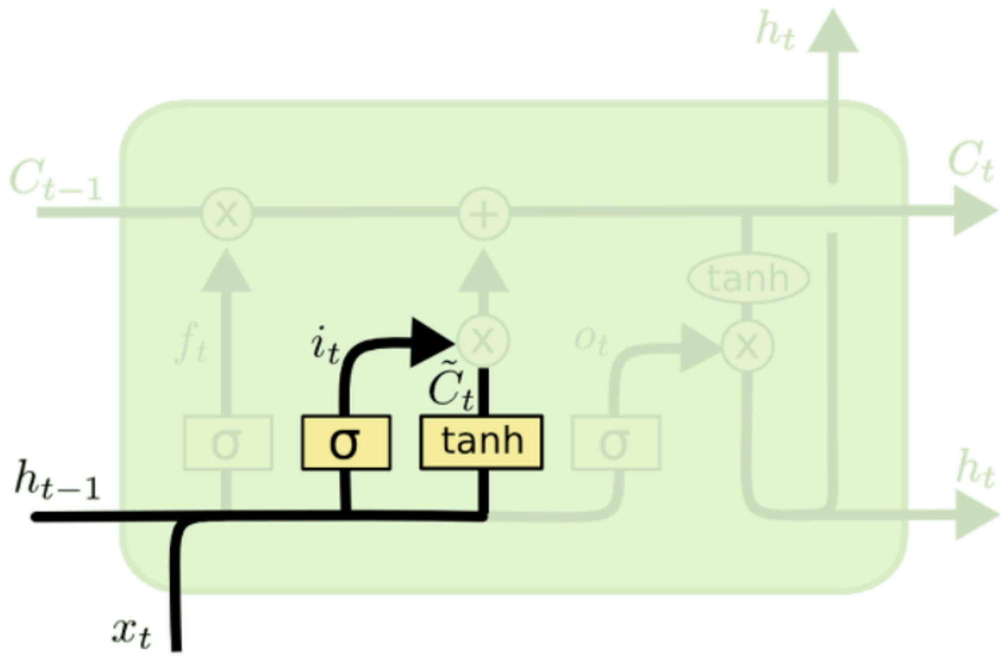
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- 출력
  - Output gate layer: 어떤 출력값을 출력할지 결정 (0~1)

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

  - 실제 출력값

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



LSTM components

- 저장할 정보 (cell states)

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

- 새로운 정보
  - Forget gate layer: cell state에서 어떤 정보를 버릴지(0~1)

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

  - Input gate layer: 어떤 값을 업데이트 할지 결정 (0~1)

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

  - cell state 에 더해질 수 있는 새로운 후보값 (-1~1)

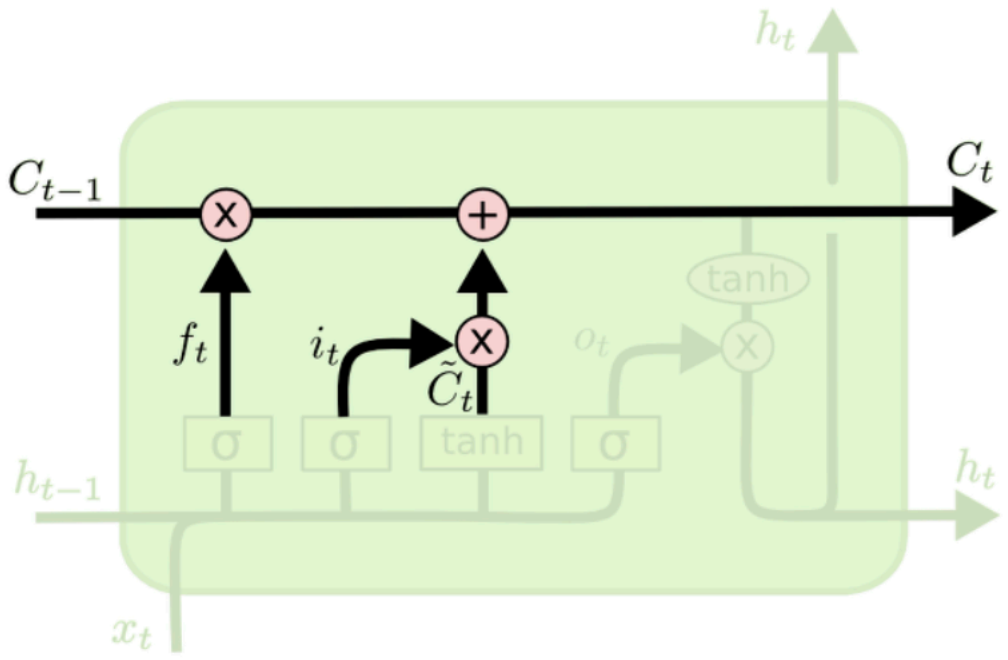
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- 출력
  - Output gate layer: 어떤 출력값을 출력할지 결정 (0~1)

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

  - 실제 출력값

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



LSTM components

- 저장할 정보 (cell states)

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

- 새로운 정보
  - Forget gate layer: cell state에서 어떤 정보를 버릴지(0~1)

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

  - Input gate layer: 어떤 값을 업데이트 할지 결정 (0~1)

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

  - cell state 에 더해질 수 있는 새로운 후보값 (-1~1)

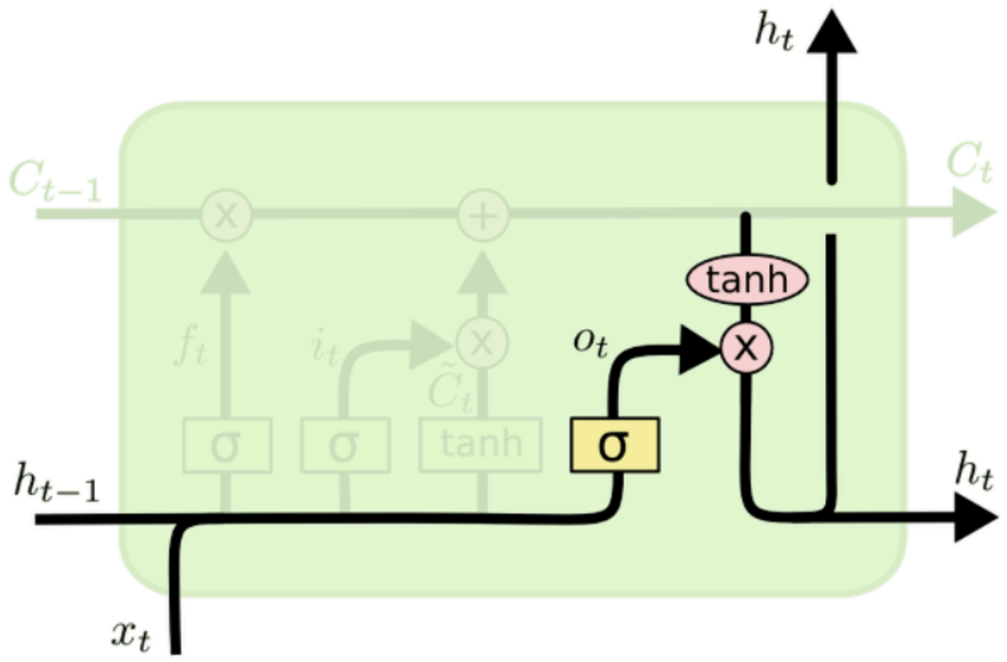
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- 출력
  - Output gate layer: 어떤 출력값을 출력할지 결정 (0~1)

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

  - 실제 출력값

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

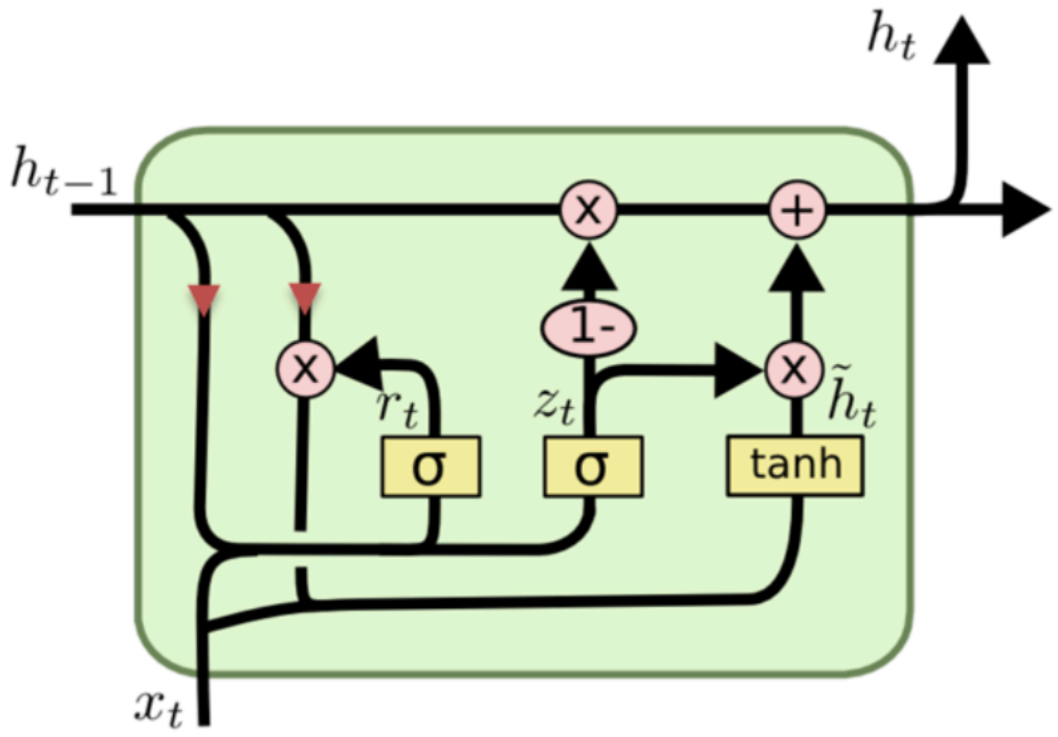


Gated Recurrent Unit (GRU)

- variant of LSTM
  - LSTM 의 장점을 유지하면서도 계산복잡성을 줄임
  - LSTM 의 게이트 일부를 생략
    - ✓ Input gate 와 forget gate 를 합침
  - cell state 와 hidden state (output) 을 합침
- Gates of GRU
  - Update gate (0~1): $z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$
    - ✓ 현시점 정보와 과거 정보를 어떻게 합칠지

      $\tilde{h}_t = \tanh(Wx_t + r_t \odot Uh_{t-1})$
  - Reset gate (0~1): $r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$
    - ✓ 현시점 정보에 과거 정보 얼마나 반영할지

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



Applications of RNNs

- Machine Translation
- Image captioning
- Question Answering

Machine Translation

- Input and output data
  - 입력데이터: source language에서의 한 문장
  - 출력데이터: target language에서의 한 문장
- 문제점
  - 완전한 단어 matching 이 이루어지기 어려움
  - 문장의 길이가 다름
- 해결책
  - Encoder-decoder scheme
  - encoder로 source language의 입력이 다 끝난 이후에 target language 의 문장이 나오도록 구성

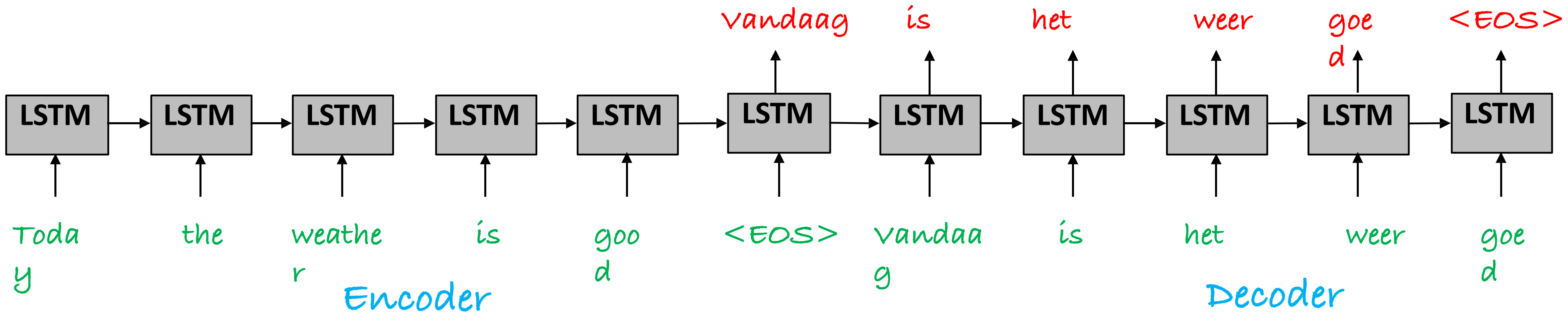
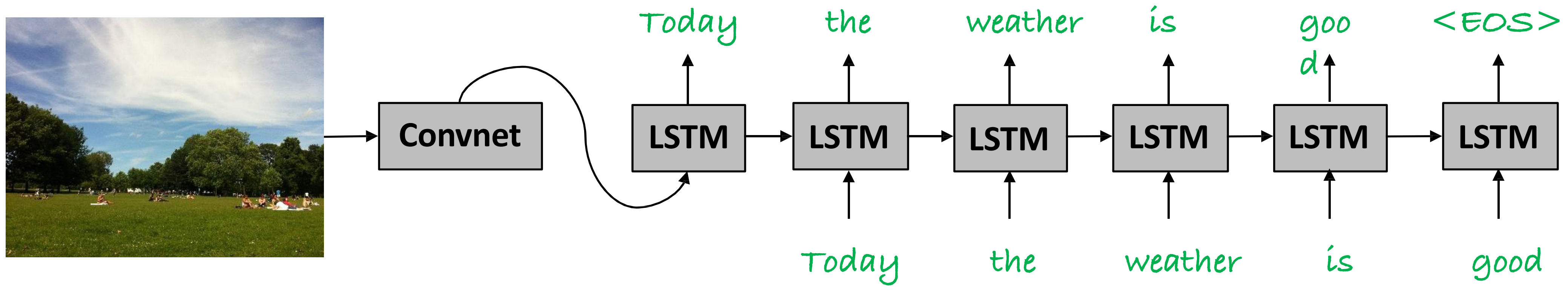


Image captioning

- An image is a thousand words, literally!
- Input and output data
  - 입력데이터: 이미지
  - 출력데이터: 문장
  - Machine translation과 유사
- Network structure
  - Encoder 부분을 convnet 으로 대체함
  - Decoder part 는 translator 와 동일

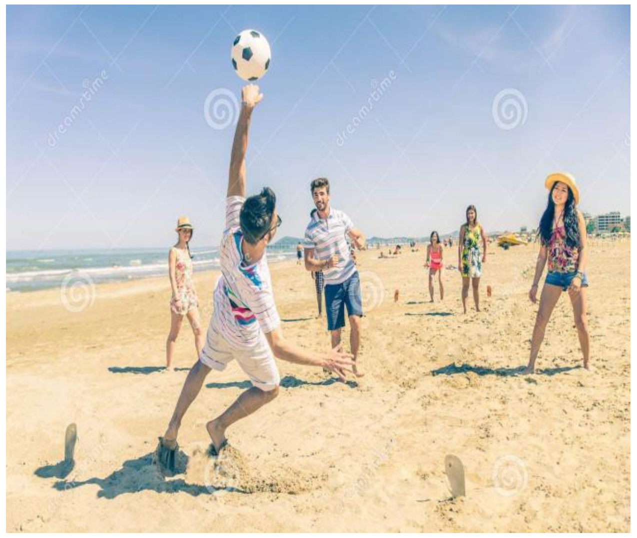


Question Answering

- Bleeding-edge research, no real consensus
  - very interesting open, research problem
  - Machine translation 과 유사
  - Encoder-decoder paradigm
- Input data and output data
  - 입력데이터: 질문 문장
  - 출력데이터: 대답 문장
- Question answering with images
  - what has been working so far is to add the image only in the beginning

Q: John entered the living room, where he met Mary. She was drinking some wine and watching a movie. What room did John enter?

A: John entered the living room.



Q: what are the people playing?

A: They play beach football