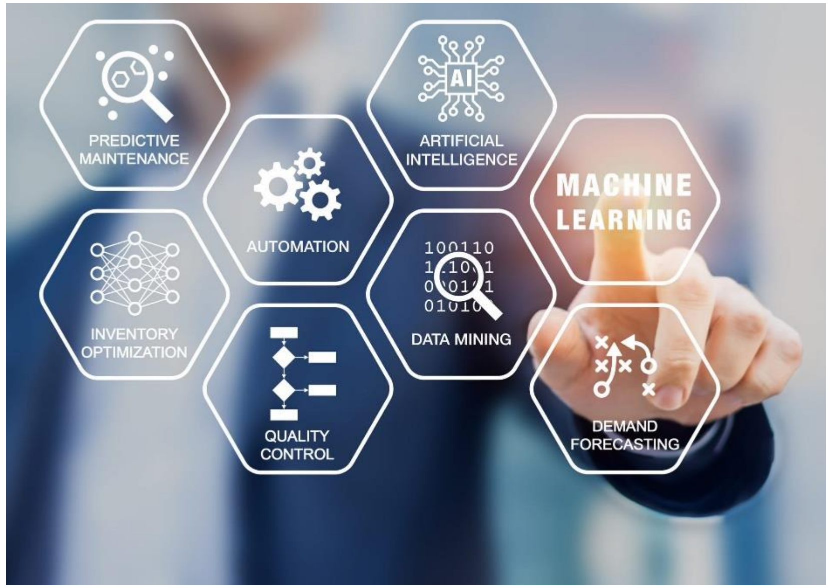


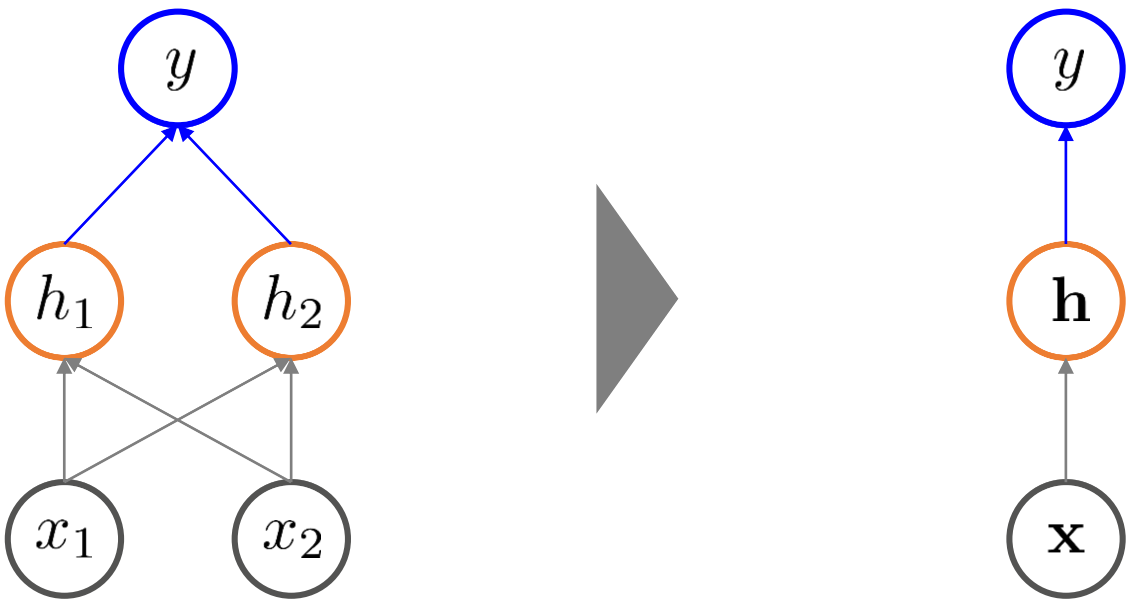
순환신경망 & 오토인코더

강필성
고려대학교 산업경영공학부
Bflysoft & WIGO AI LAB

AGENDA

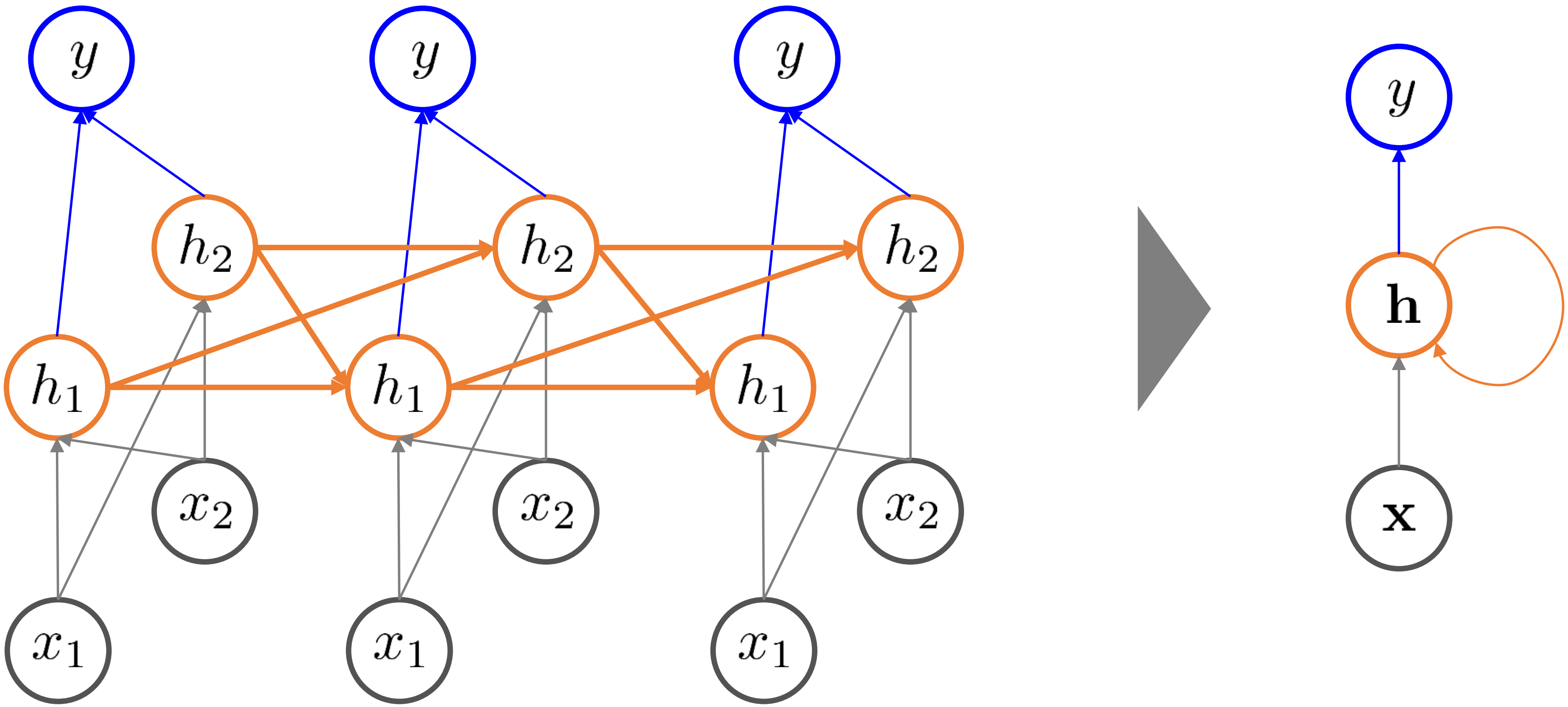
RNN Basics: Sequence

- 순서(sequence)가 없는 인공신경망 구조



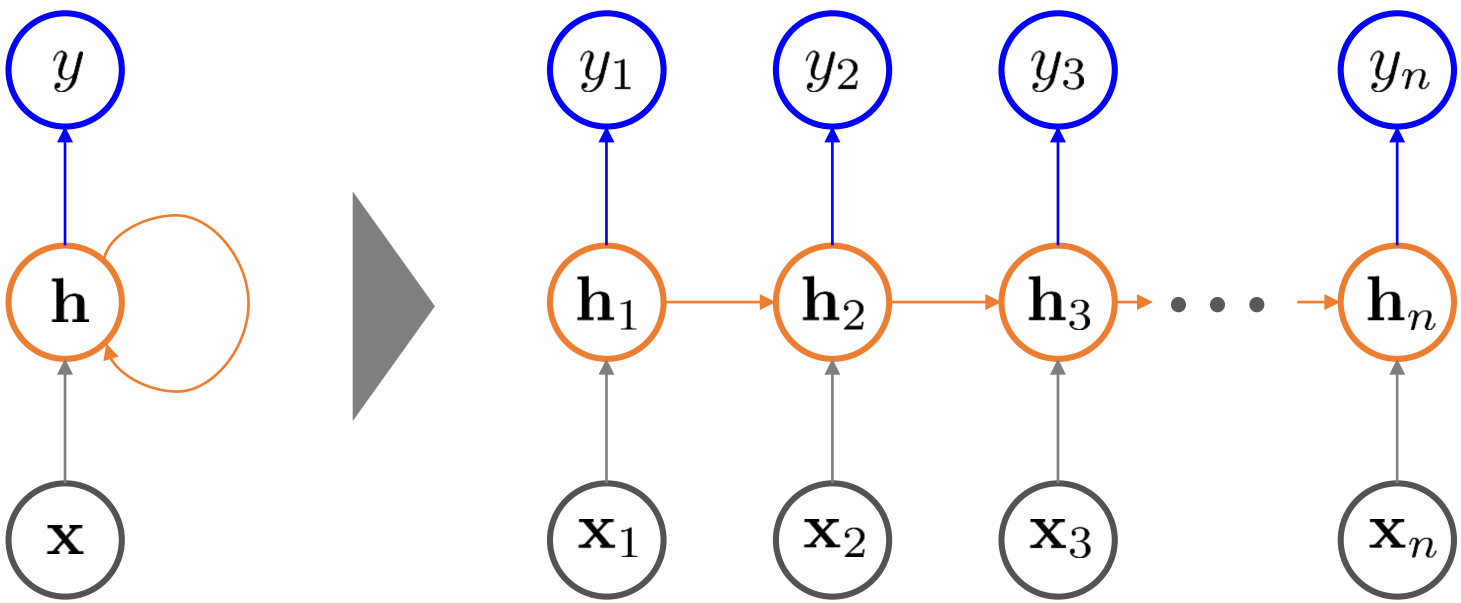
RNN Basics: Sequence

- 순서(sequence)가 있는 인공신경망 구조



RNN Basics: Sequence

- 순서(sequence)가 있는 인공신경망 구조 (벡터 표현)



RNN Basics: Sequence

- 순서(sequence)가 있는 인공신경망 구조
  - ✓ 입력-출력에 따른 활용 사례

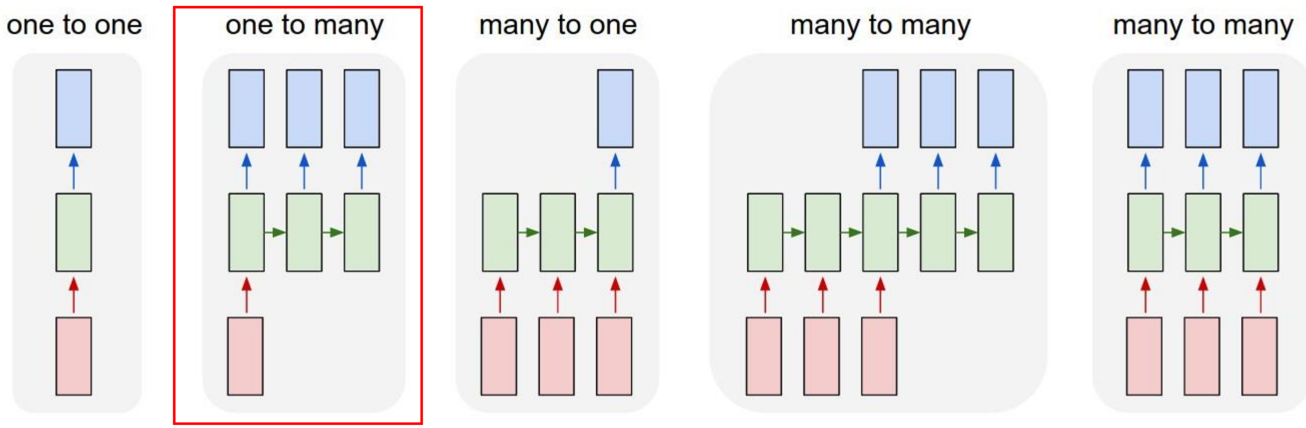
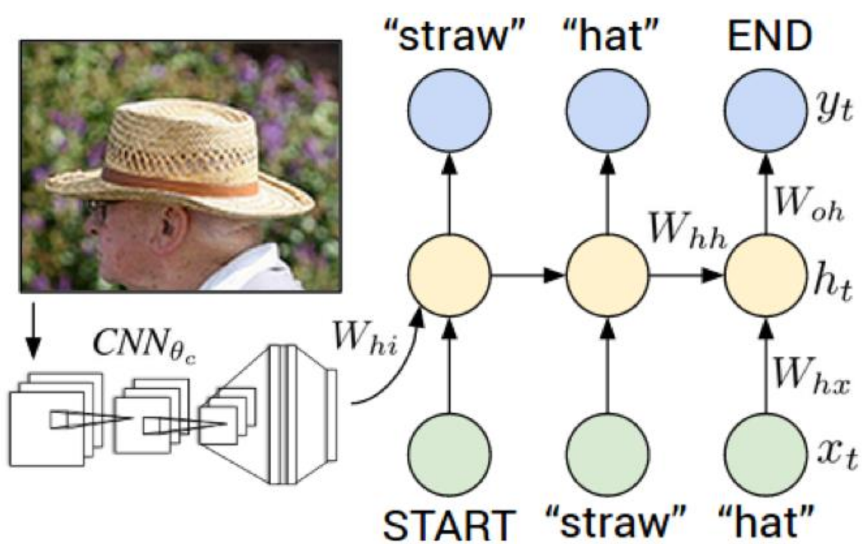
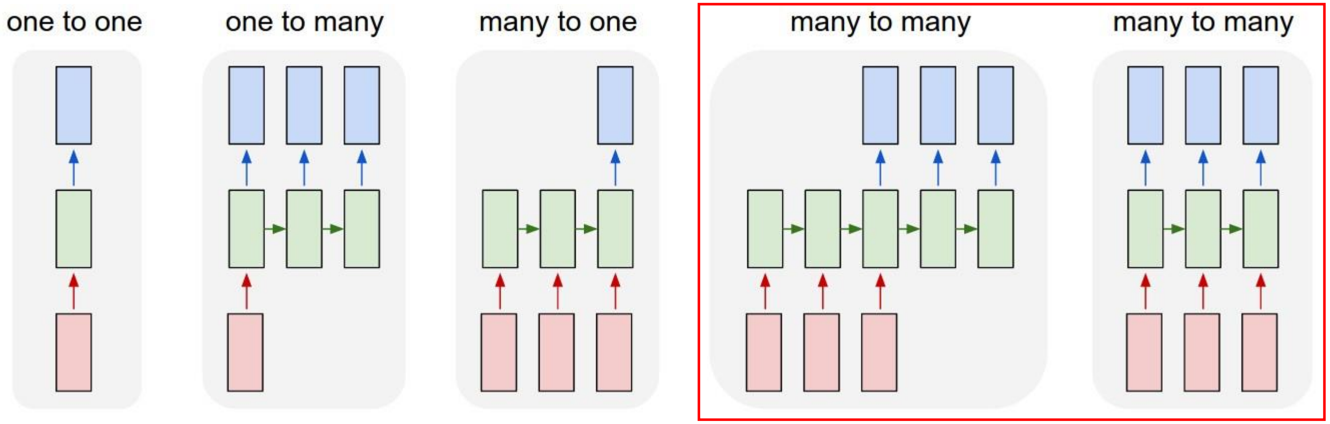


Image captioning

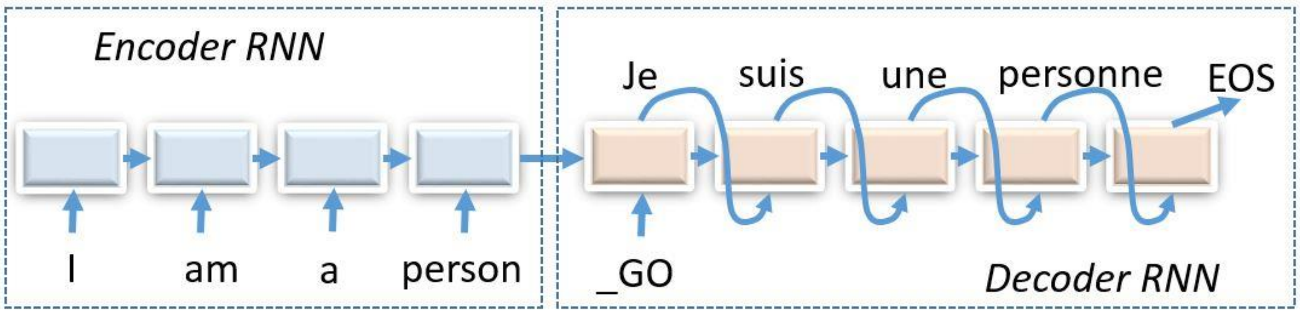


RNN Basics: Sequence

- NN structure with sequence
  - ✓ Input-Output structure



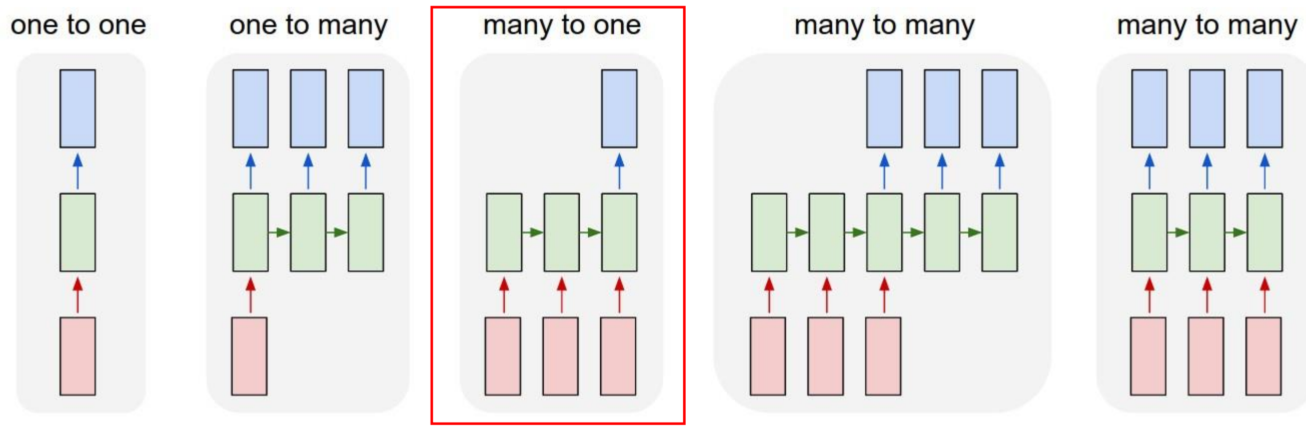
Machine translation/Dialog system



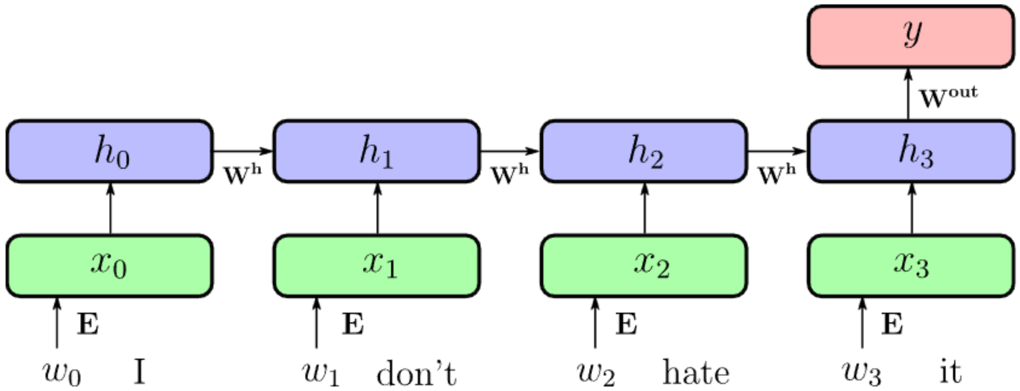
https://esciencegroup.com/2016/03/04/fun-with-recurrent-neural-nets-one-more-dive-into-cntk-and-tensorflow/

RNN Basics: Sequence

- NN structure with sequence
  - ✓ Input-Output structure

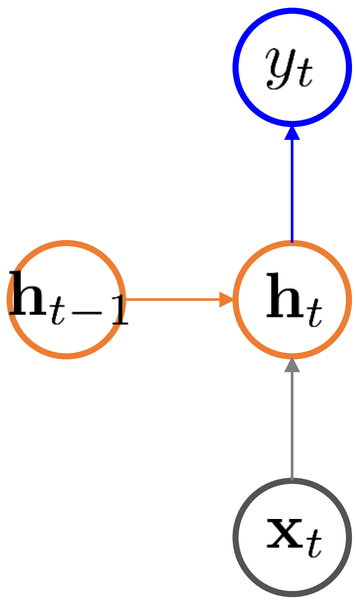


Text classification/Sentiment analysis



RNN Basics: Forward Path

- 기본 RNN(Vanilla RNN) 구조에서 정보의 흐름
  - ✓ f( )와 g( )는 활성 함수

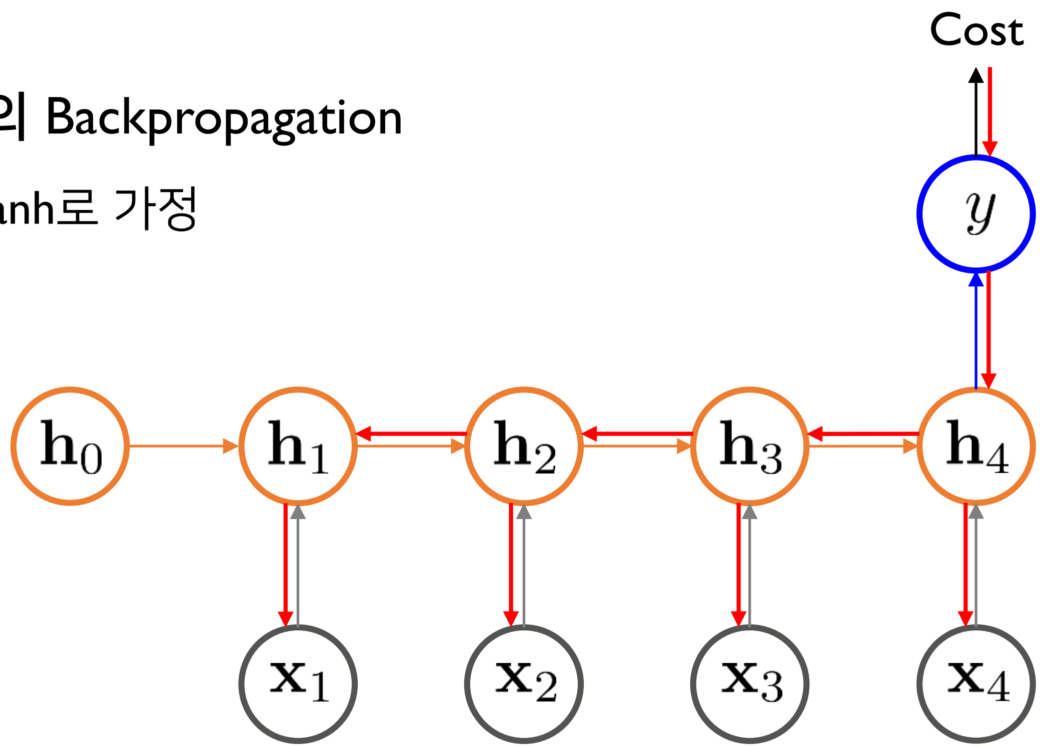


$$y_t = g(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y)$$

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_x)$$

RNN Basics: Gradient Vanishing/Exploding Problem

- RNN에서의 Backpropagation
  - ✓ f( )는 Tanh로 가정



$$\frac{\partial Cost}{\partial \mathbf{W}_{xh}} = \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{W}_{xh}} + \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{W}_{xh}}$$
$$+ \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{W}_{xh}}$$
$$+ \frac{\partial Cost}{\partial y} \times \frac{\partial y}{\partial \mathbf{h}_4} \times \frac{\partial \mathbf{h}_4}{\partial \mathbf{h}_3} \times \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \times \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \times \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}_{xh}}$$

RNN Basics: Gradient Vanishing/Exploding Problem

- RNN에서의 Backpropagation
  - ✓ 일반적으로

$$\frac{\partial Cost}{\partial \mathbf{W}_{xh}} = \sum_{i=1}^{n} \left(\frac{\partial Cost}{\partial y} \cdot \frac{\partial y}{\partial \mathbf{h}_n} \cdot \boxed{\left(\prod_{j=i}^{n-1} \frac{\partial \mathbf{h}_{j+1}}{\partial \mathbf{h}_j} \right)} \cdot \boxed{\frac{\partial \mathbf{h}_i}{\partial \mathbf{W}_{xh}}} \right)$$

  - ✓ f( )는 Tanh이고

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_x) = tanh(\mathbf{z}_t)$$

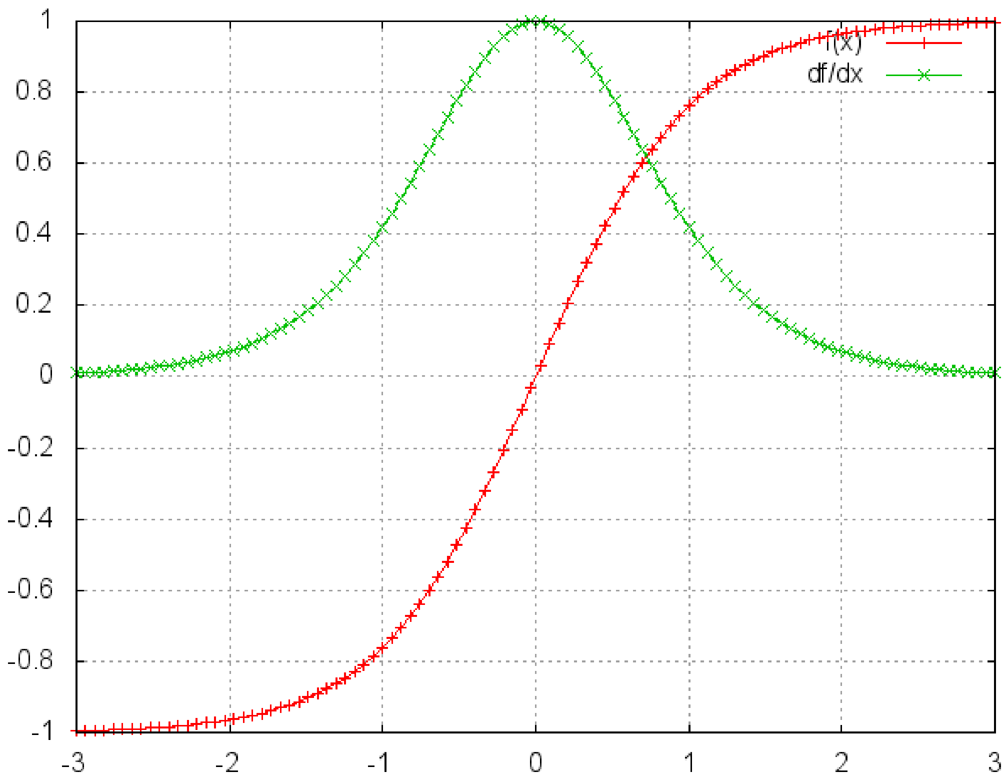
$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{z}_t} \times \frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_{t-1}} = (1 - tanh^2(\mathbf{z}_t)) \times \mathbf{W}_{hh}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{xh}} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{z}_t} \times \frac{\partial \mathbf{z}_t}{\partial \mathbf{W}_{xh}} = (1 - tanh^2(\mathbf{z}_t)) \times \mathbf{x}_t$$

RNN Basics: Gradient Vanishing/Exploding Problem

- RNN에서의 Backpropagation

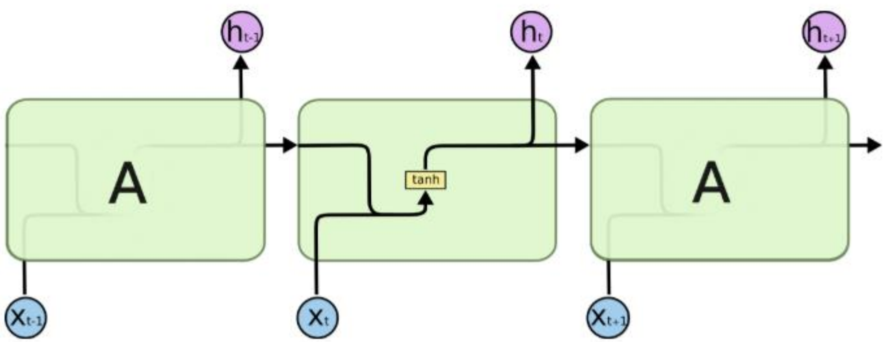
$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \frac{\partial \mathbf{h}_t}{\partial \mathbf{z}_t} \times \frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_{t-1}} = \boxed{(1 - tanh^2(\mathbf{z}_t))} \times \mathbf{W}_{hh}$$



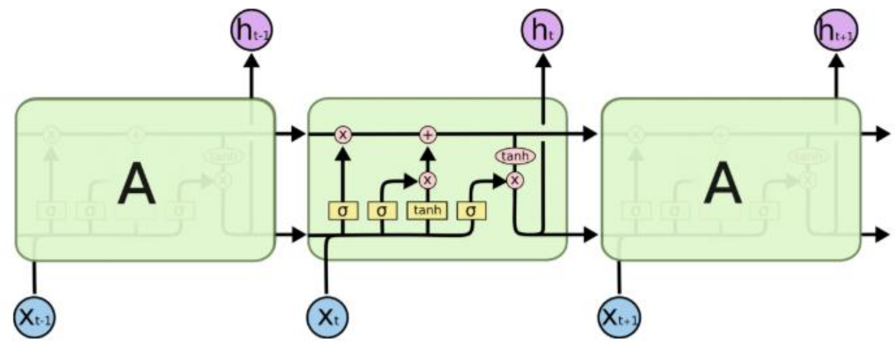
RNN Basic: LSTM

- LSTM: Long Short-Term Memory
  - ✓ Gradient exploding/vanishing 문제를 해결하여 Long-term dependency 학습 가능

Vanilla RNN

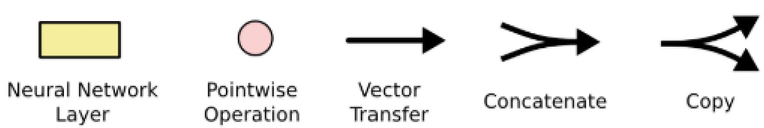


LSTM



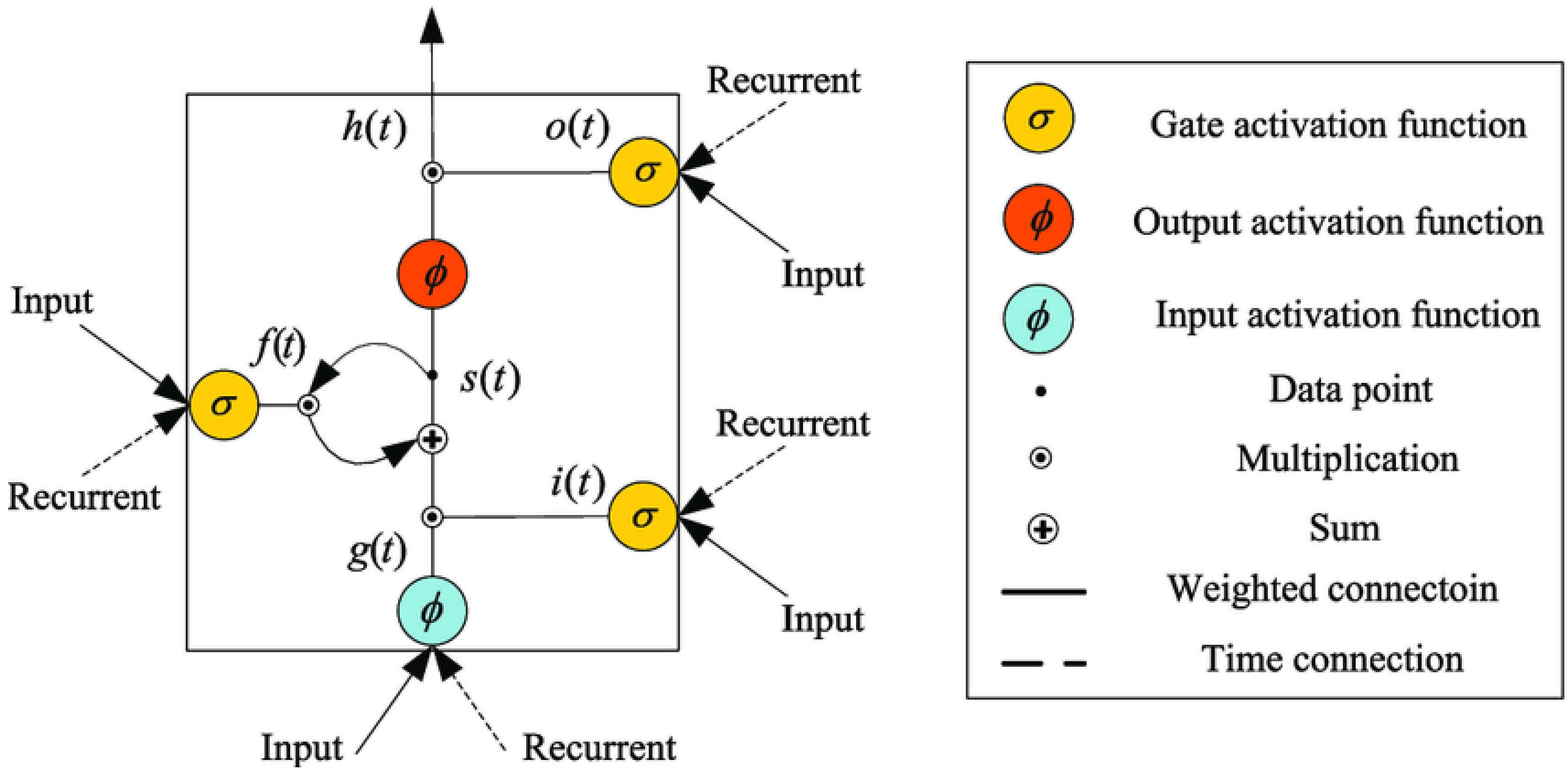
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Operation symbols



RNN Basic: LSTM

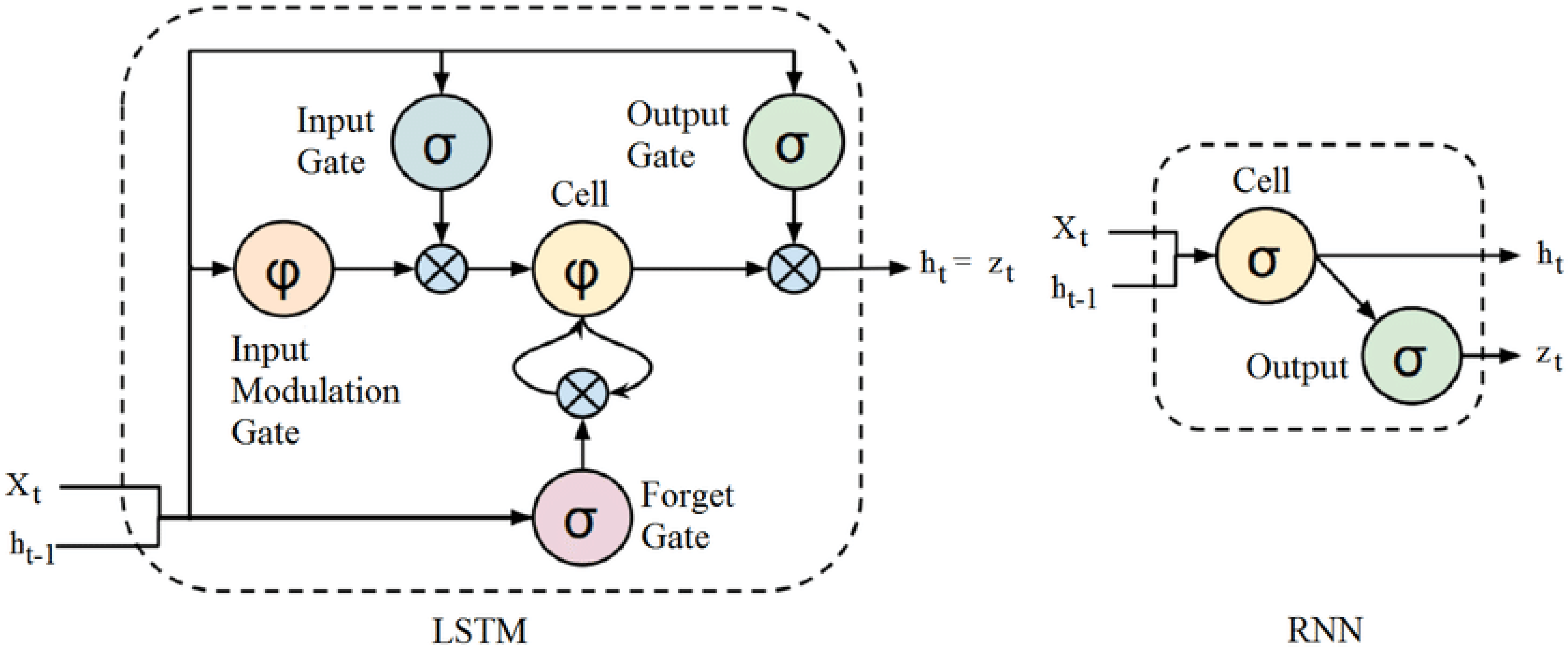
- LSTM: Long Short-Term Memory
  - ✓LSTM의 구조가 복잡하여 다양한 diagram들이 존재



https://www.researchgate.net/figure/The-architecture-of-a-LSTM-memory-cell_fig2_314251130

RNN Basic: LSTM

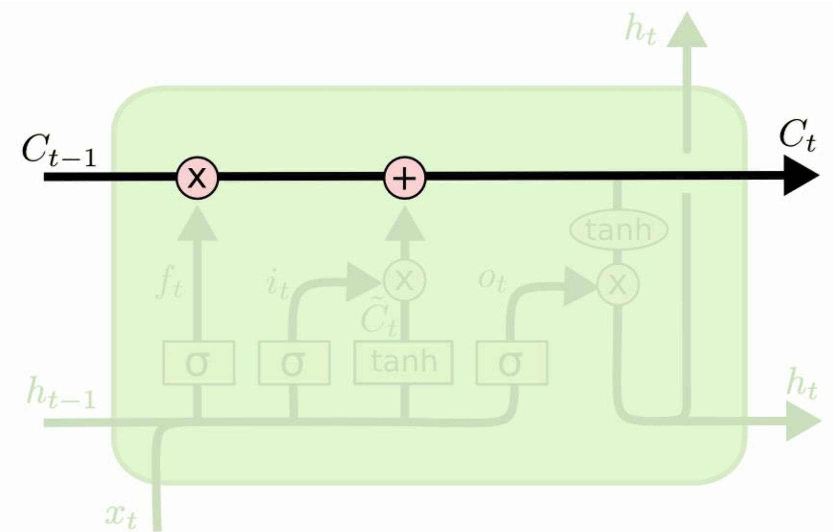
- LSTM: Long Short-Term Memory
  - ✓LSTM의 구조가 복잡하여 다양한 diagram들이 존재



https://www.researchgate.net/figure/An-example-of-a-basic-LSTM-cell-left-and-a-basic-RNN-cell-right-Figure-follows-a_fig2_306377072

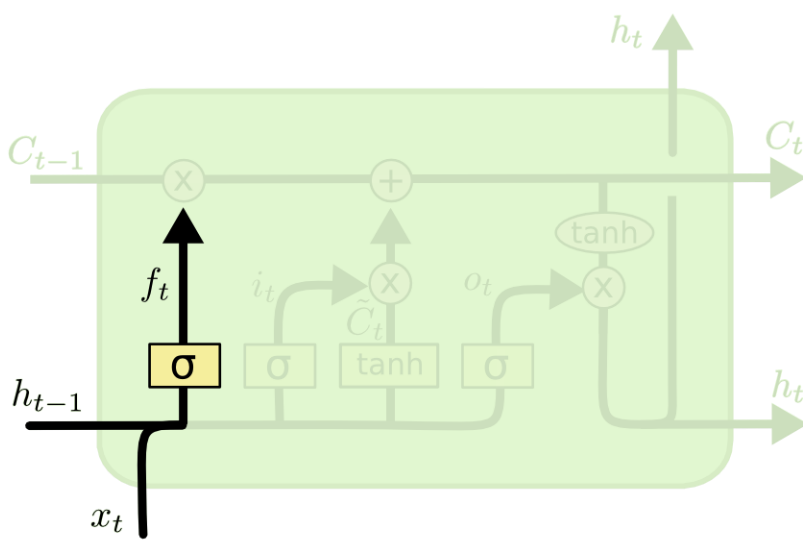
RNN Basic: LSTM

- Cell state
  - ✓ LSTM 핵심 구성 요소, 아래 그림에서는 다이어그램의 상부를 관통하는 선(line)



RNN Basic: LSTM

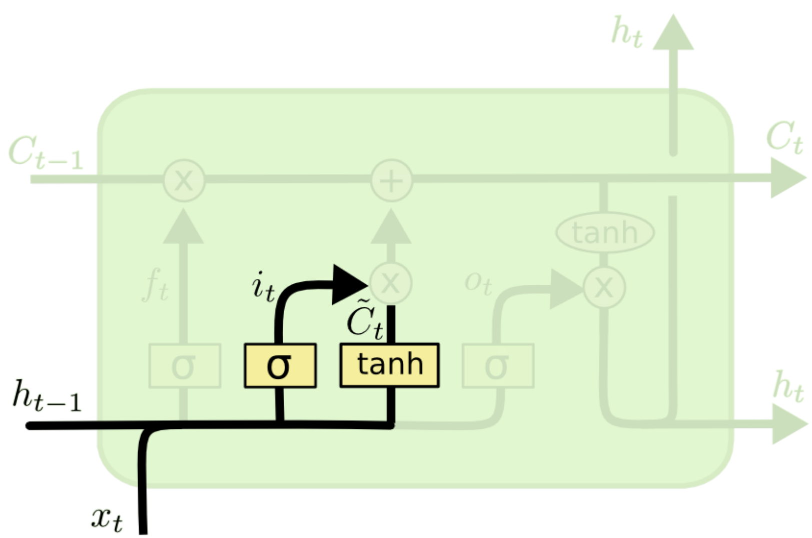
- Step 1: 지금까지의 cell state에 저장된 정보 중에서 얼마만큼을 망각(forget)할 것인지 결정
  - ✓ Forget gate: 이전 단계의 hidden state h_{t-1} 와 현 단계의 입력 x_t 으로부터 0과 1사이의 값을 출력 (Sigmoid 함수 사용)
    - 1: 지금까지 cell state에 저장된 모든 정보를 보존
    - 0: 지금까지 cell state에 저장된 모든 정보를 무시



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

RNN Basic: LSTM

- Step 2: 새로운 정보를 얼마만큼 cell state에 저장할 것인지를 결정
  - ✓ Input gate: 어떤 값을 업데이트 할 것인지 결정
  - ✓ Tanh layer를 사용하여 새로운 cell state의 후보 $\tilde{C}_t$ 을 생성

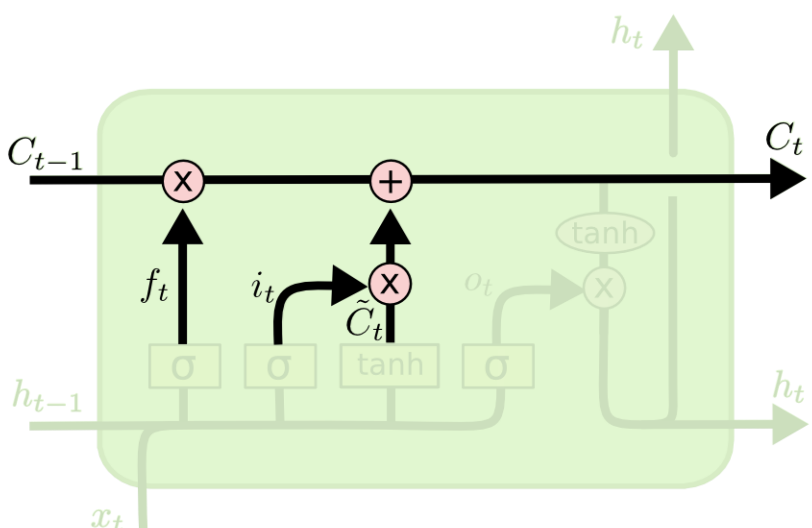


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

RNN Basic: LSTM

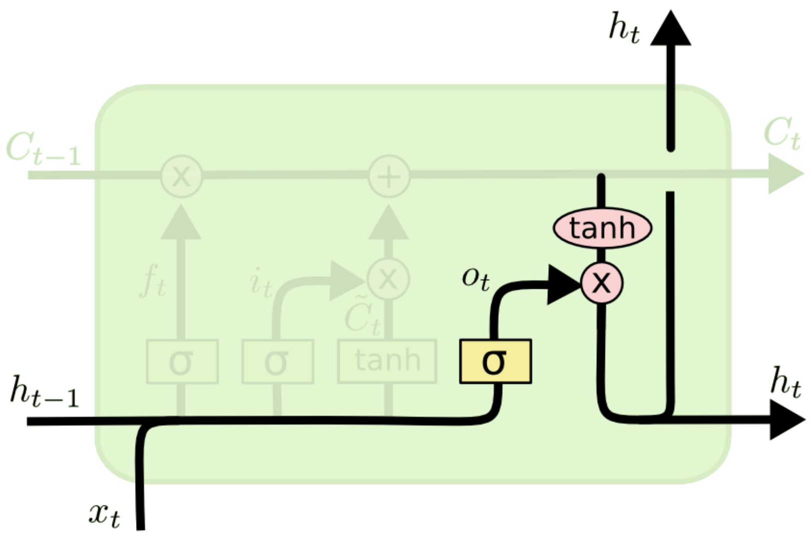
- Step 3: **예전** cell state**를 새로운** cell state**로 업데이트**
  - ✓ 예전 cell state를 얼마만큼 망각할 것인가를 계산한 forget gate 결과값과 곱함
  - ✓ 새로운 cell state 후보와 얼마만큼 보존할 것인가를 계산한 input gate 결과값을 곱함
  - ✓ 두 값을 더하여 새로운 cell state 값으로 결정



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

RNN Basic: LSTM

- Step 4: 출력 값을 결정
  - ✓ 이전 hidden state 값과 현재의 입력 값을 이용하여 output gate 값을 산출
  - ✓ Output gate 값과 현재의 cell state 값을 결합하여 현재의 hidden state 값을 계산

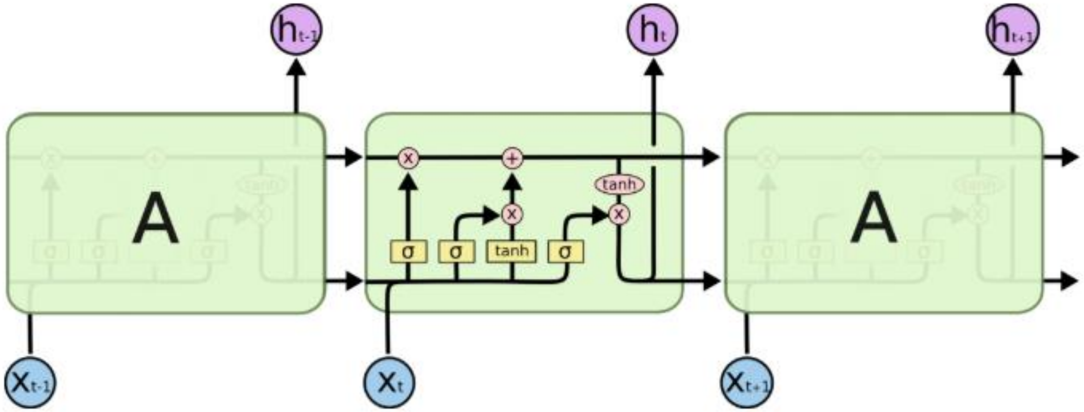


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

RNN Basic: LSTM

- LSTM 요약



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

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

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

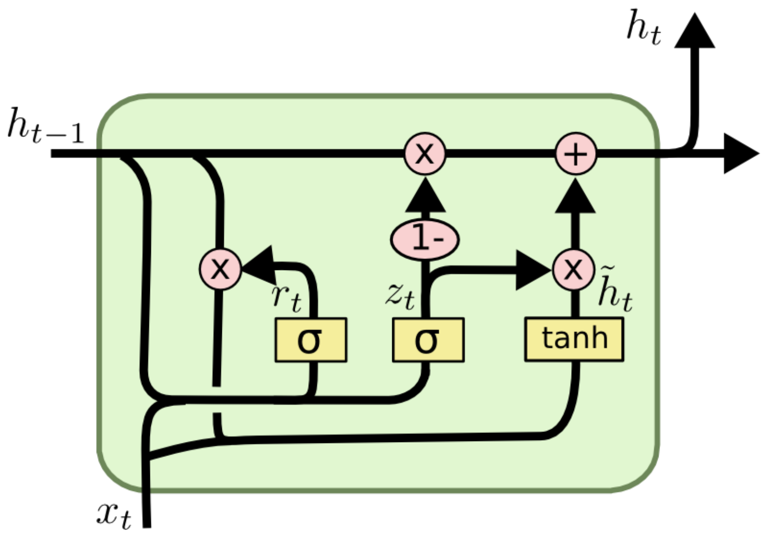
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

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

RNN: GRU

- GRU: Gated Recurrent Unit



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

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

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

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

✓ LSTM을 단순화한 구조, 실제 활용에서는 LSTM과 GRU의 성능 차이는 미비함

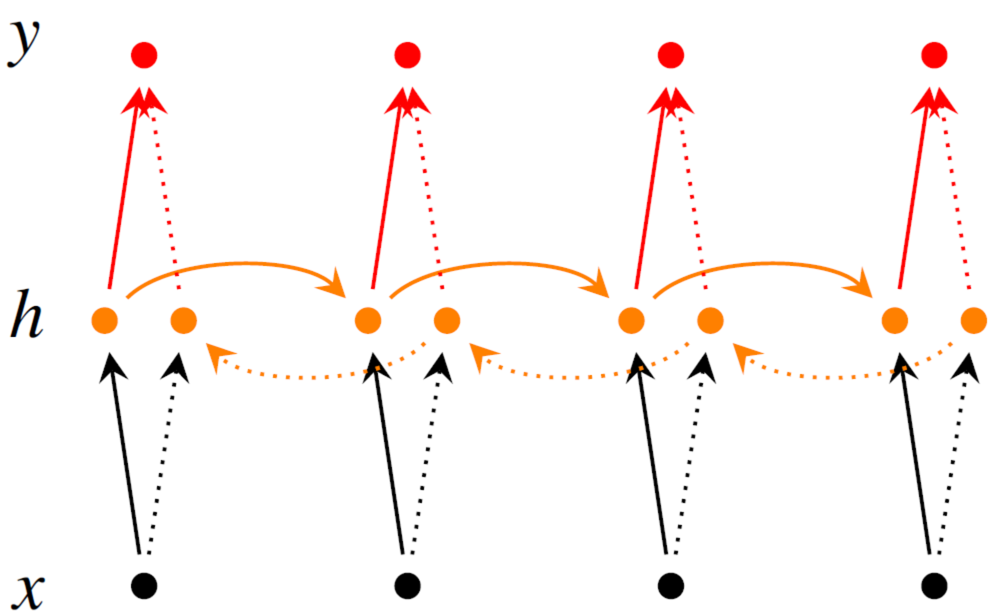
✓ 별도의 Cell state가 존재하지 않음

✓ LSTM의 forget gate와 input gage를 하나의 update gate로 결합

✓ Reset gate를 통해 망각과 새로운 정보 업데이트 정도를 결정

RNN Variations: Bidirectional RNN

- Bidirectional RNN: 양방향 순환신경망
  - ✓ 정보의 입력을 시간의 순방향과 역방향 관점에서 함께 처리



$$\overrightarrow{h}_t = f(\overrightarrow{W} x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

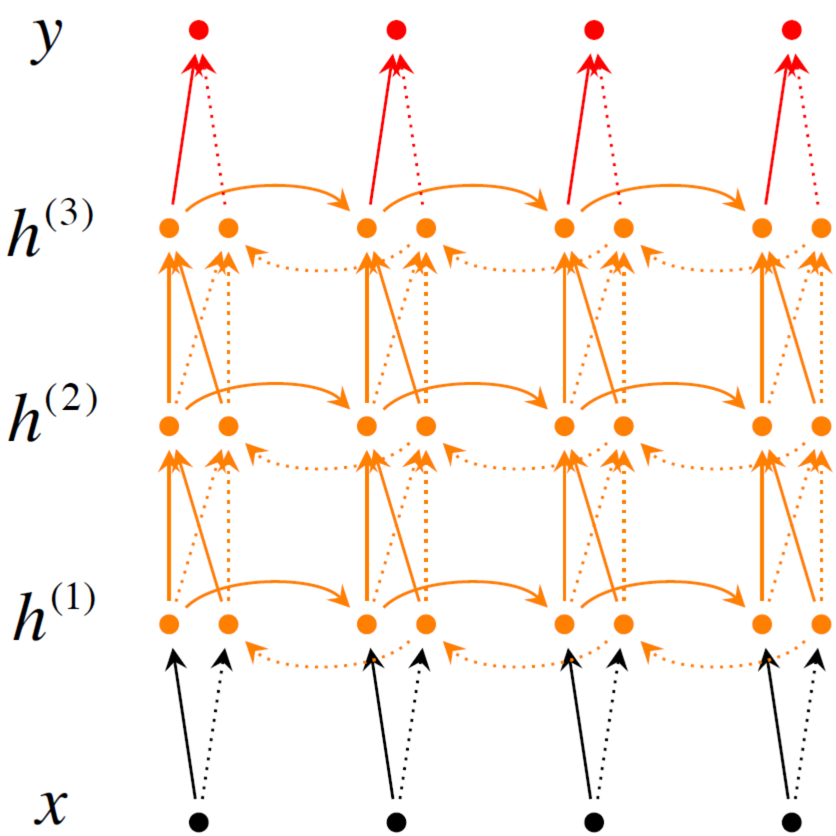
$$\overleftarrow{h}_t = f(\overleftarrow{W} x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t]+c)$$

http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN Variations: Bidirectional RNN

- Deep-Bidirectional RNN
  - ✓ RNN의 hidden layer가 꼭 한 층일 필요가 있나? 더 쌓아보자!



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

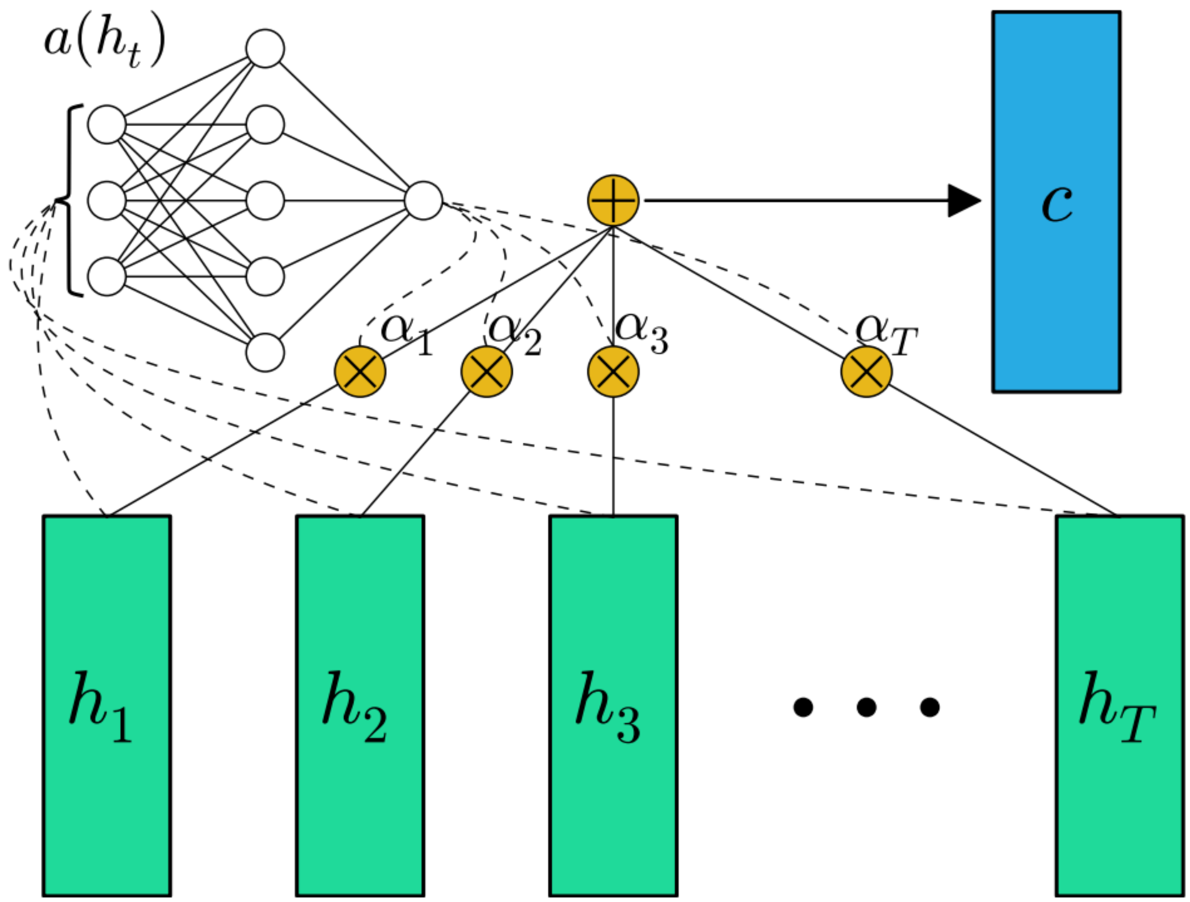
$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

RNN: Attention

- Attention
  - ✓ 어느 시점 정보가 RNN의 최종 출력 값에 영향을 미치는지를 알려줄 수 있는 메커니즘

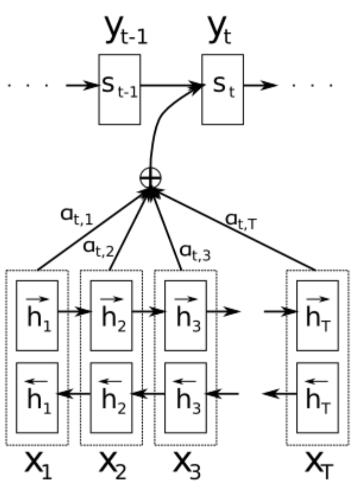


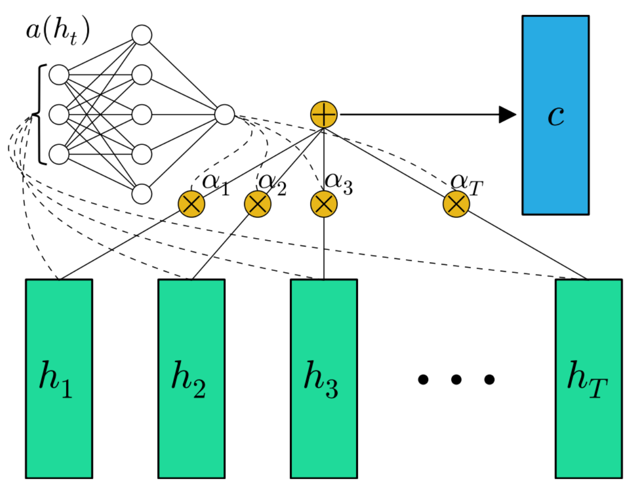
Source: (Raffel and Ellis, 2016)

RNN: Attention

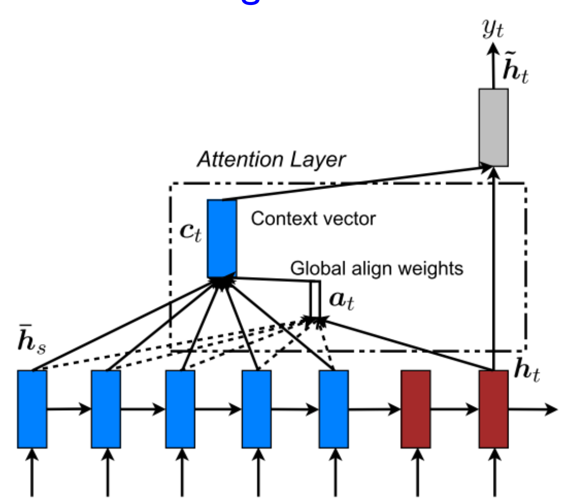
- 두 가지 대표적 Attention 메커니즘
  - ✓ Bahadanau attention (Bahdanau et al., 2015)
    - Attention scores are <u>separated trained</u>, the current hidden state is a function of the context vector and the previous hidden state
  - ✓ Luong attention (Luong et al., 2015)
    - Attention scores are <u>not trained</u>, the new current hidden state is the simple tanh of the weighted concatenation of the context vector and the current hidden state of the decoder

Bahdanau attention



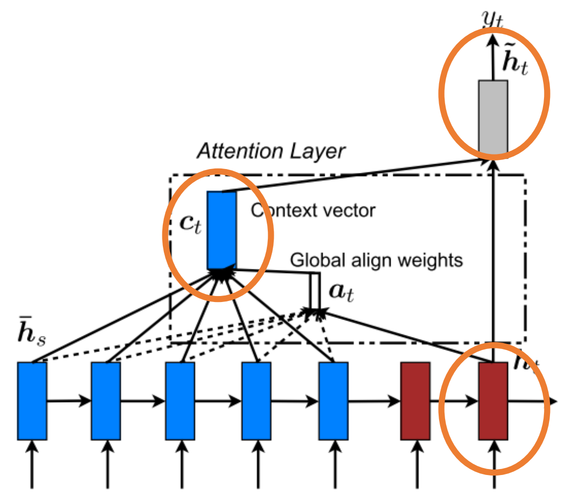


Luong attention



RNN: Attention

- Luong Attention
  - ✓ Decoder의 새로운 hidden state는
    - Weighted concatenation of the context vector와
    - Current hidden state of the decoder를
    - Concatenation한 뒤 Tanh함수를 적용한 것



$$\tilde{\mathbf{h}}_t = tanh(\mathbf{W_c}[\mathbf{c}_t; \mathbf{h}_t])$$

- 이 hidden state가 RNN의 출력을 결정하는 softmax에 입력으로 투입

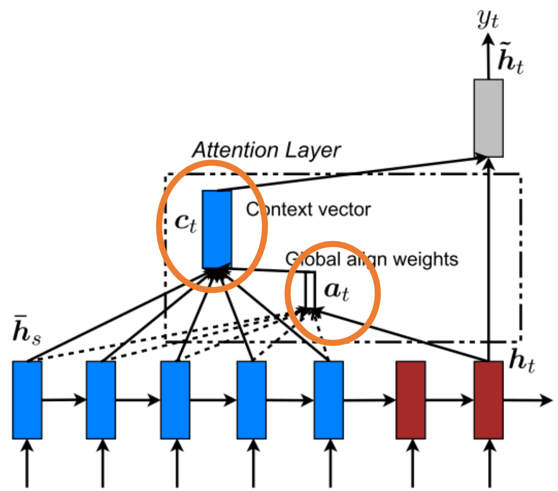
$$p(y_t | y_{y<t}, x) = \mathrm{softmax}(\mathbf{W_s}\tilde{\mathbf{h}}_t)$$

RNN: Attention

- Luong attention
  - ✓ A variable-length alignment vector:

$$\mathbf{a}_t(s) = \text{align}(\mathbf{h}_t, \overline{\mathbf{h}}_s)$$

$$= \frac{exp\Big(\text{score}(\mathbf{h}_t, \overline{\mathbf{h}}_s)\Big)}{\sum_{s'} exp\Big(\text{score}(\mathbf{h}_t, \overline{\mathbf{h}}'_s)\Big)}$$



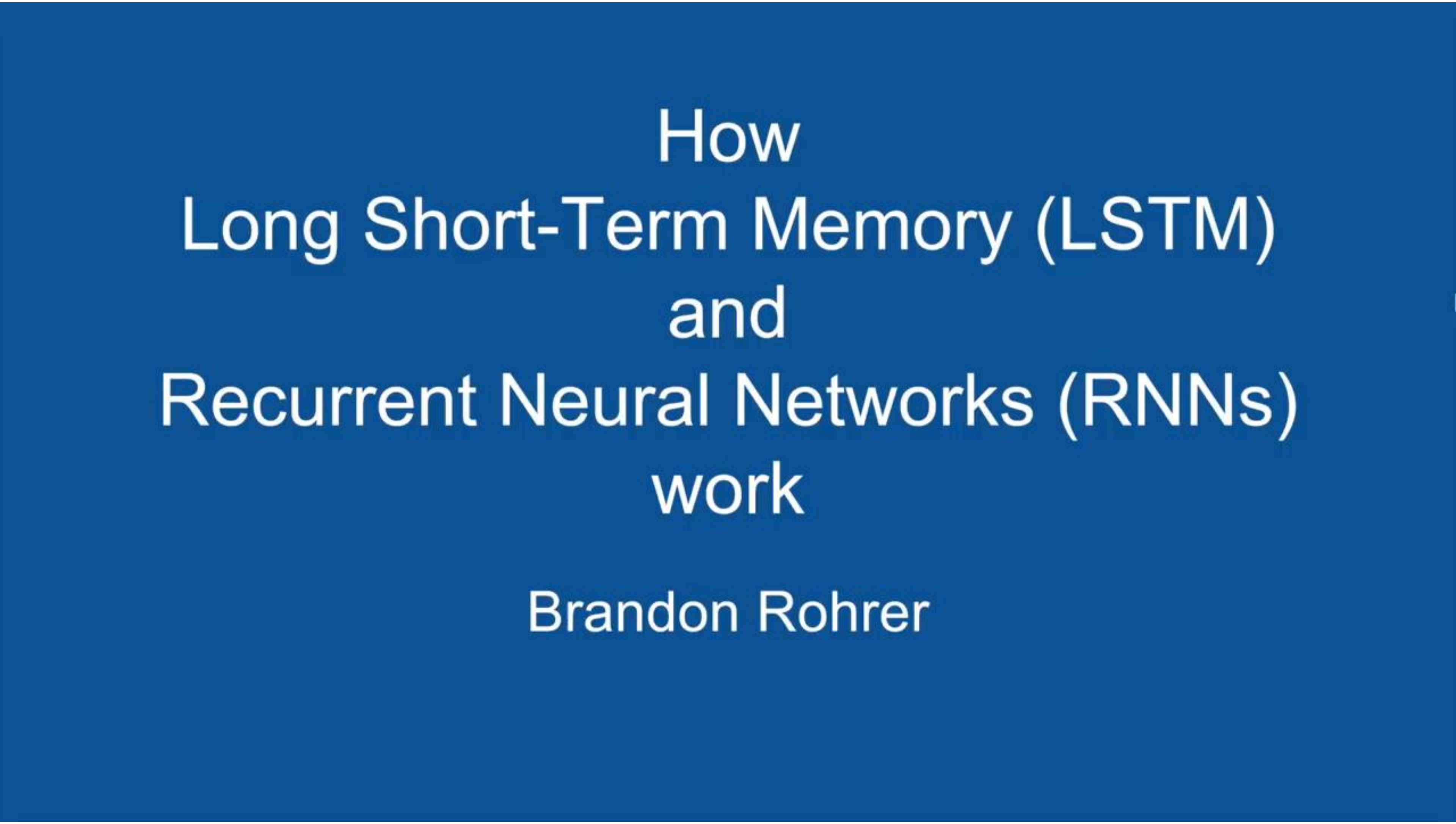
  - ✓ **score** is referred as a context-based function:

$$\text{score}(\mathbf{h}_t, \overline{\mathbf{h}}_s) = \begin{cases} {\mathbf{h}_t}^T \overline{\mathbf{h}}_s, & dot \\ {\mathbf{h}_t}^T \mathbf{W_a} \overline{\mathbf{h}}_s, & general \\ \mathbf{v}_a^T tanh(\mathbf{W_c}[\mathbf{c}_t; \mathbf{h}_t]), & concat \end{cases}$$

  - ✓ Context vector

$$\mathbf{c}_t = \overline{\mathbf{h}}_s \mathbf{a}_t$$

RNN Procedure

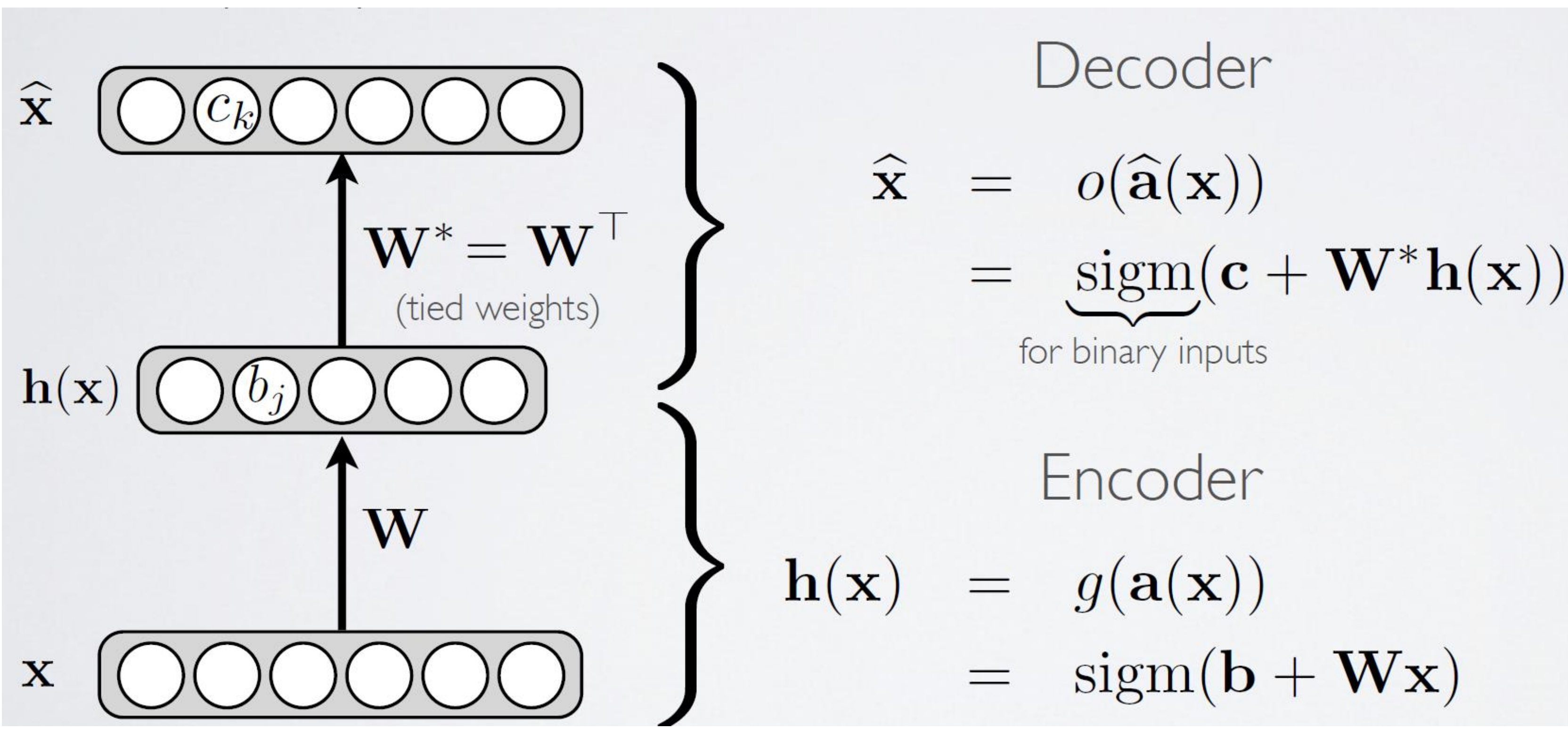


https://www.youtube.com/watch?v=WCUNPb-5EYI

AGENDA

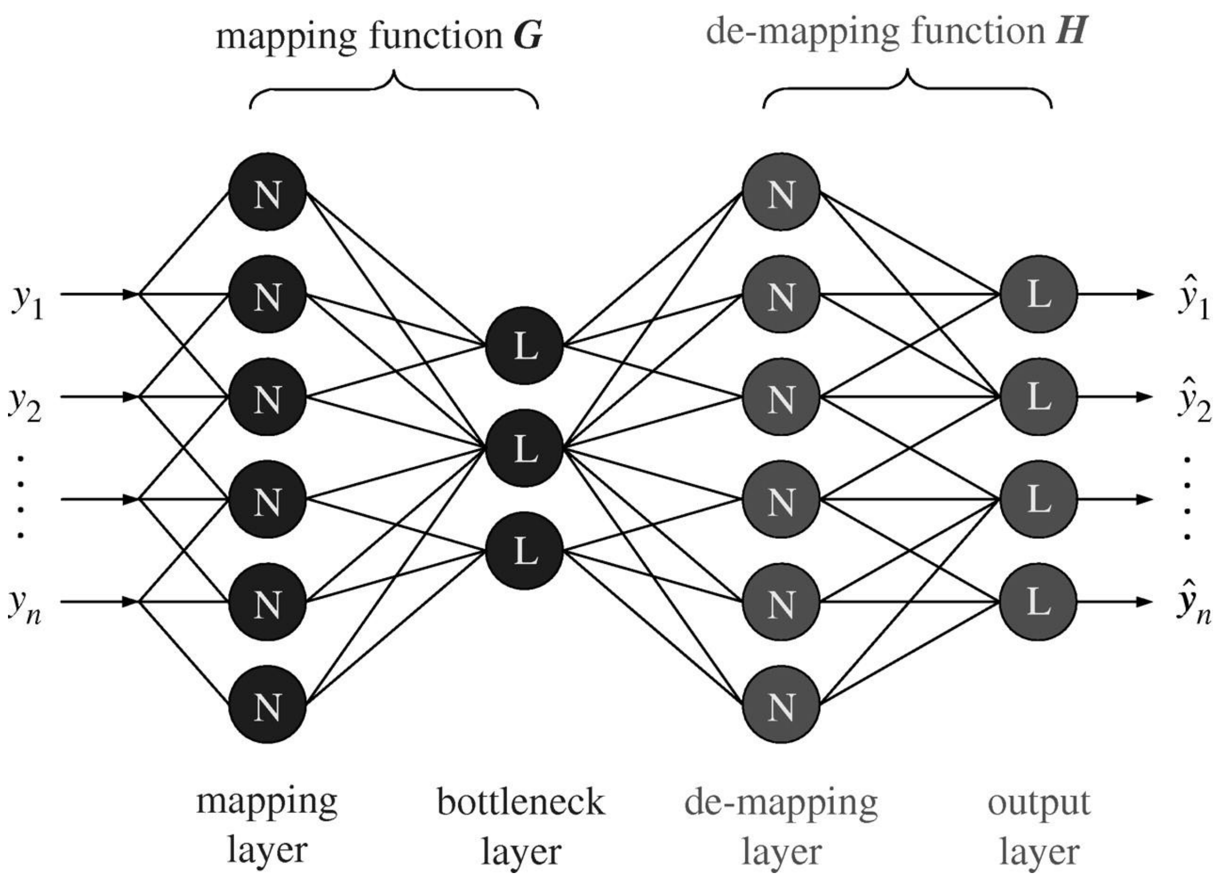
Auto-Encoder

- Auto-Encoder (Auto-Associative Neural Network)
  - ✓입력과 출력이 동일한 인공 신경망 구조
    - Loss function: $l(f(\mathbf{x})) = \frac{1}{2}\sum_k (\widehat{x}_k - x_k)^2$



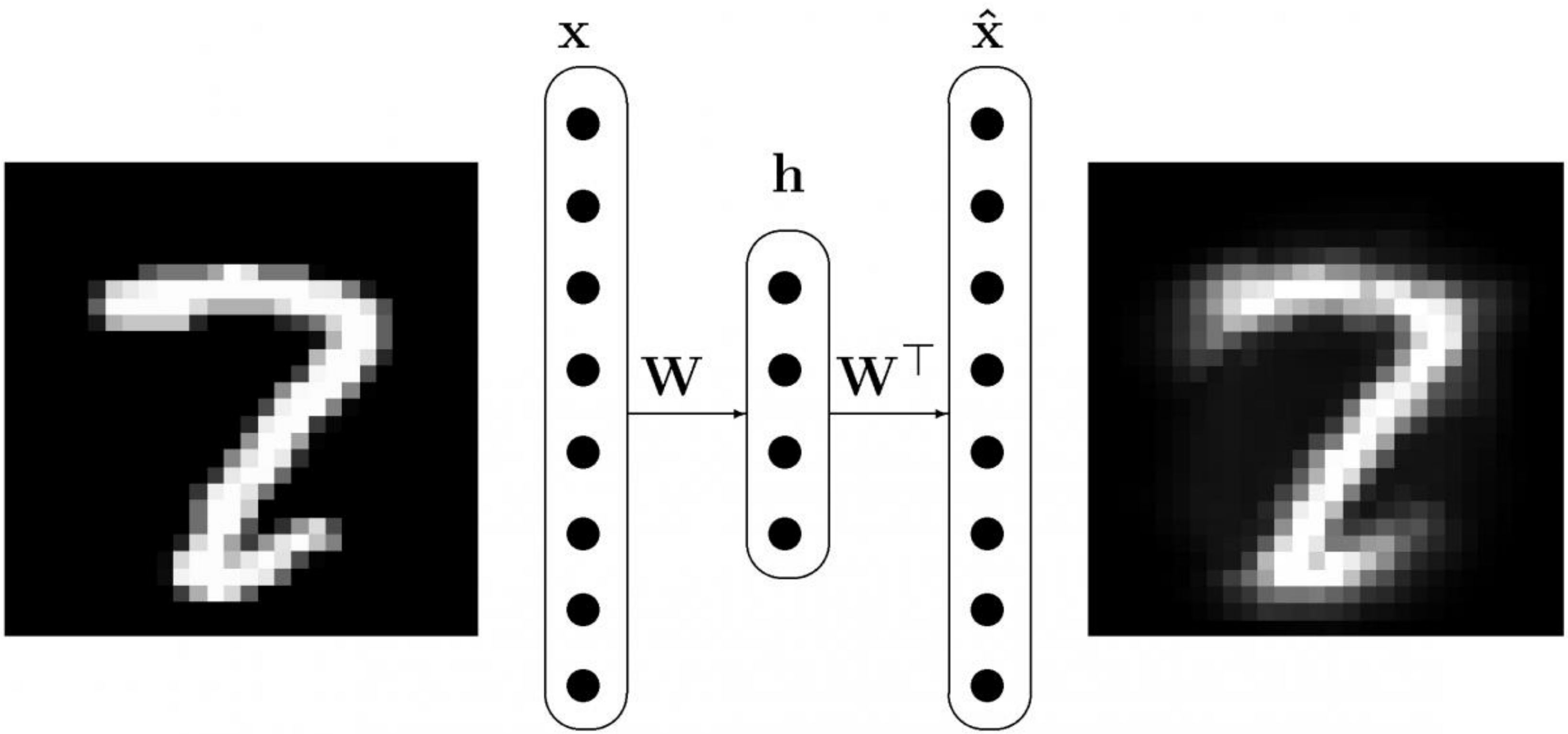
Auto-Encoder

- Auto-Encoder (Auto-Associative Neural Network)
  - ✓반드시 입력 변수의 수보다 은닉 노드의 수가 더 적은 은닉 층이 있어야 함
    - 이 층에서 정보의 축약이 이루어짐



Auto-Encoder

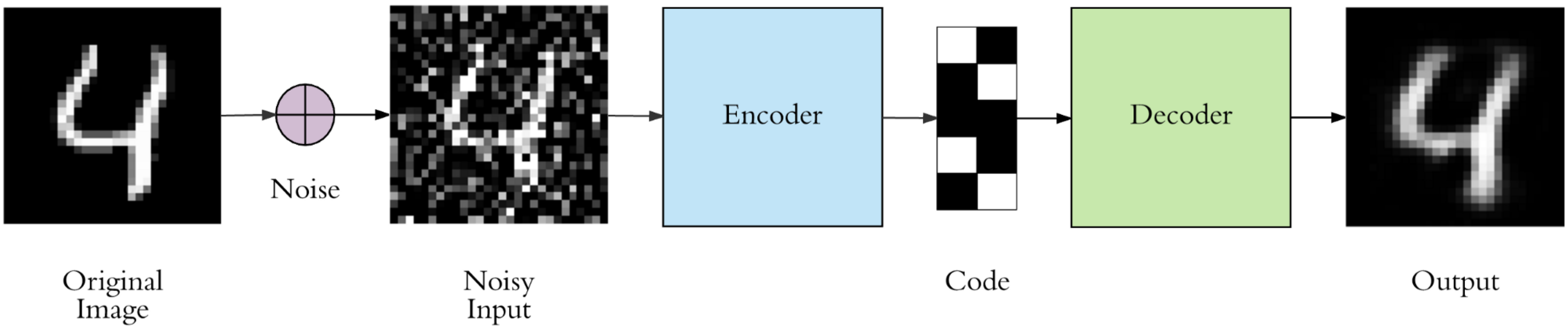
- Auto-Encoder (Auto-Associative Neural Network) 예시
  - ✓ 숫자 2를 학습시키는 오토 인코더 → 5를 입력으로 제공하면 5가 산출되지 않을 가능성이 높음 (Loss가 큼)



(a) Input (b) Neural Encoding (c) Reconstruction

Denoising Auto-Encoder

- Auto-Encoder를 포함한 인공신경망의 단점
  - ✓ 입력에 대한 약간의 변형(small perturbations)에도 모델이 민감하게 반응함
- 학습 과정에서 입력에 일부러 noise를 첨가해 보는 것은 어떨까?

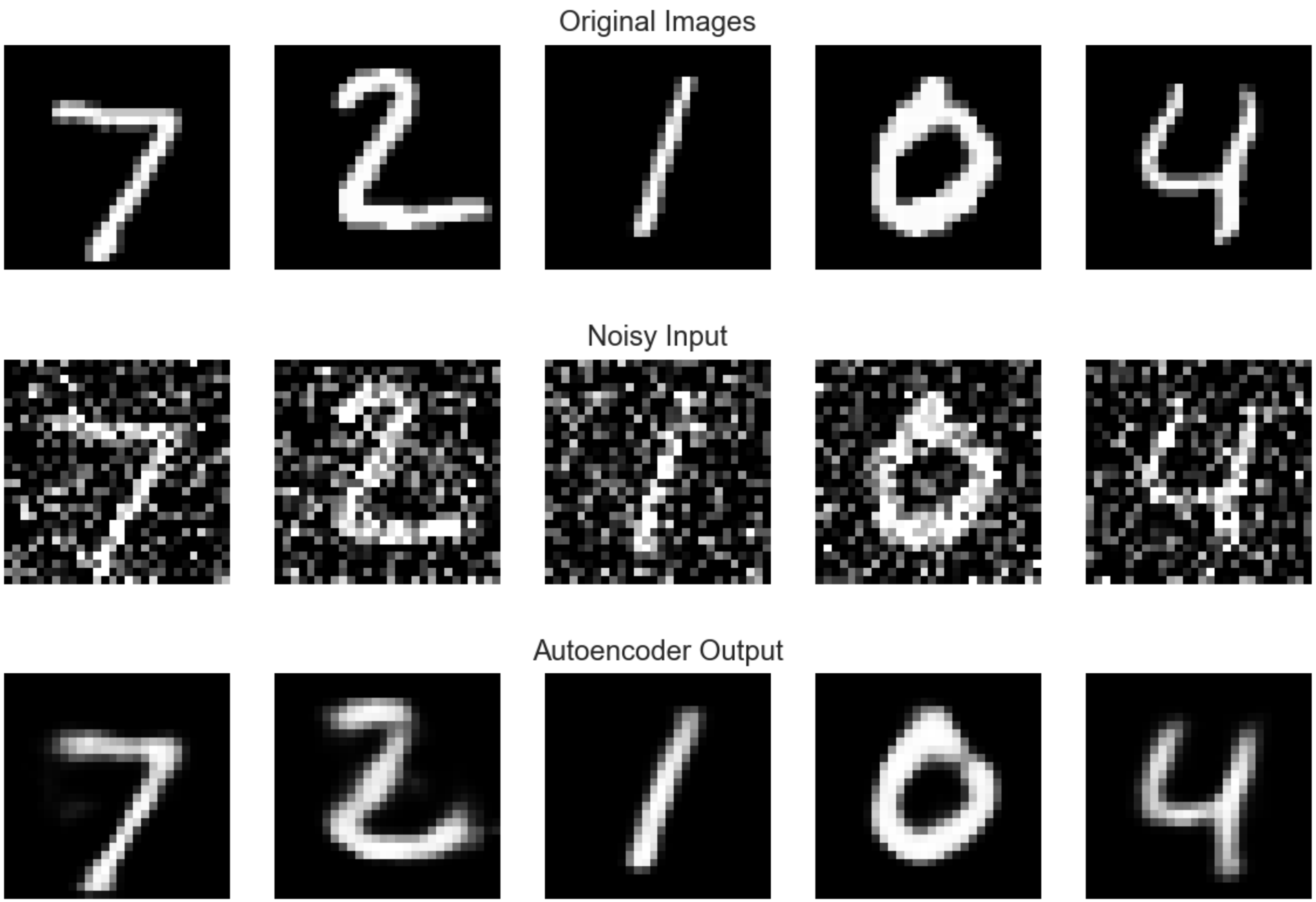


입력에 노이즈를 첨가해도

노이즈가 제거된 결과값이 나오도록

Denoising Auto-Encoder

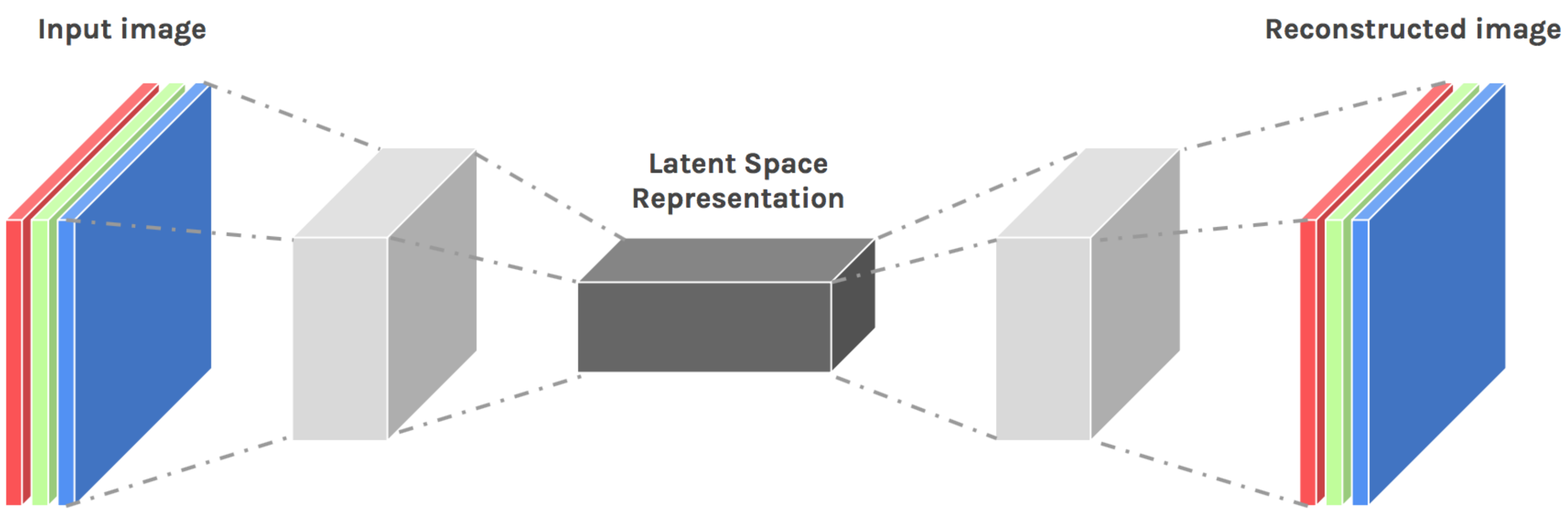
- 노이즈는 어떻게 주어야 하나?
  - ✓ 주로 Random Gaussian noise를 생성



https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Convolutional Auto-Encoder

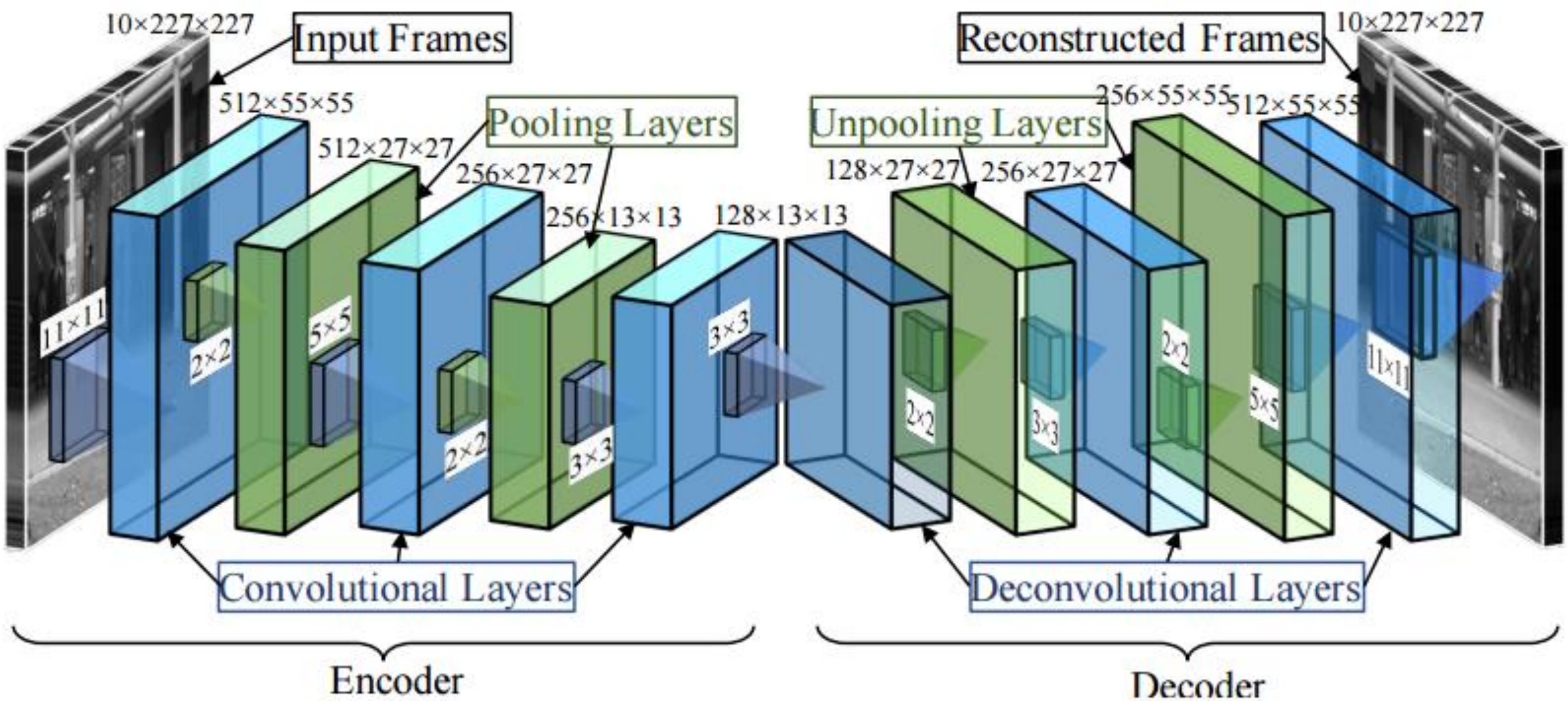
- 앞서 예시한 Hand-written Digit은 입력 데이터를 이미지가 아닌 벡터로 사용
  - ✓ 16 by 16 행렬을 256차원의 vector로 취급하고 오토 인코더 학습
- CAE = 이미지 자체를 취급하는 오토인코더



https://blog.manash.me/implementing-pca-feedforward-and-convolutional-autoencoders-and-using-it-for-image-reconstruction-8ee44198ea55

Convolutional Auto-Encoder

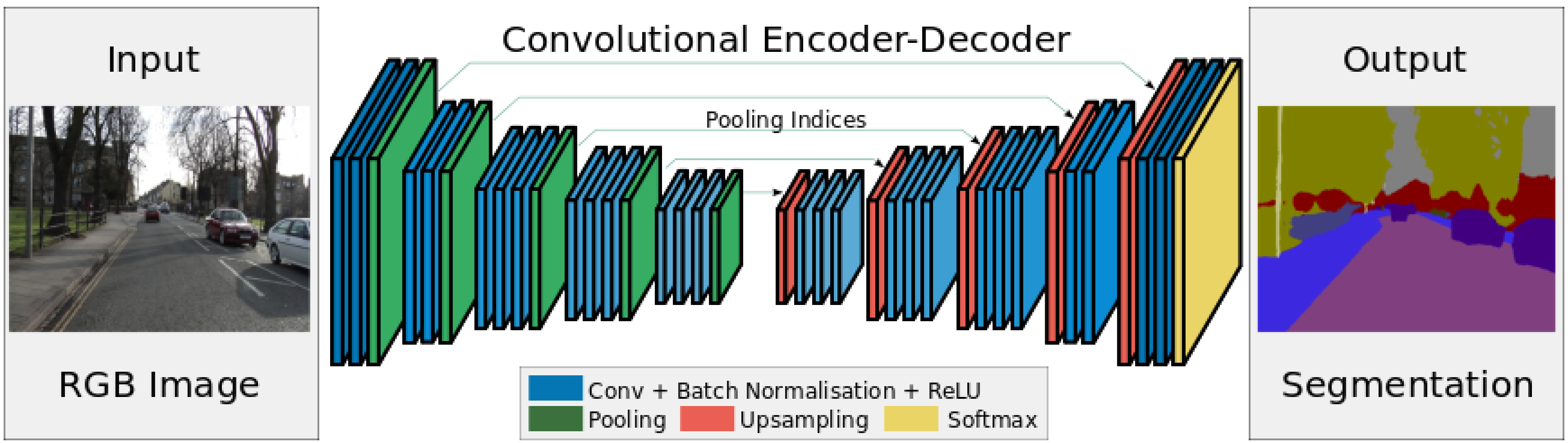
- CAE = 이미지 자체를 취급하는 오토인코더



https://blog.manash.me/implementing-pca-feedforward-and-convolutional-autoencoders-and-using-it-for-image-reconstruction-8ee44198ea55

Convolutional Auto-Encoder

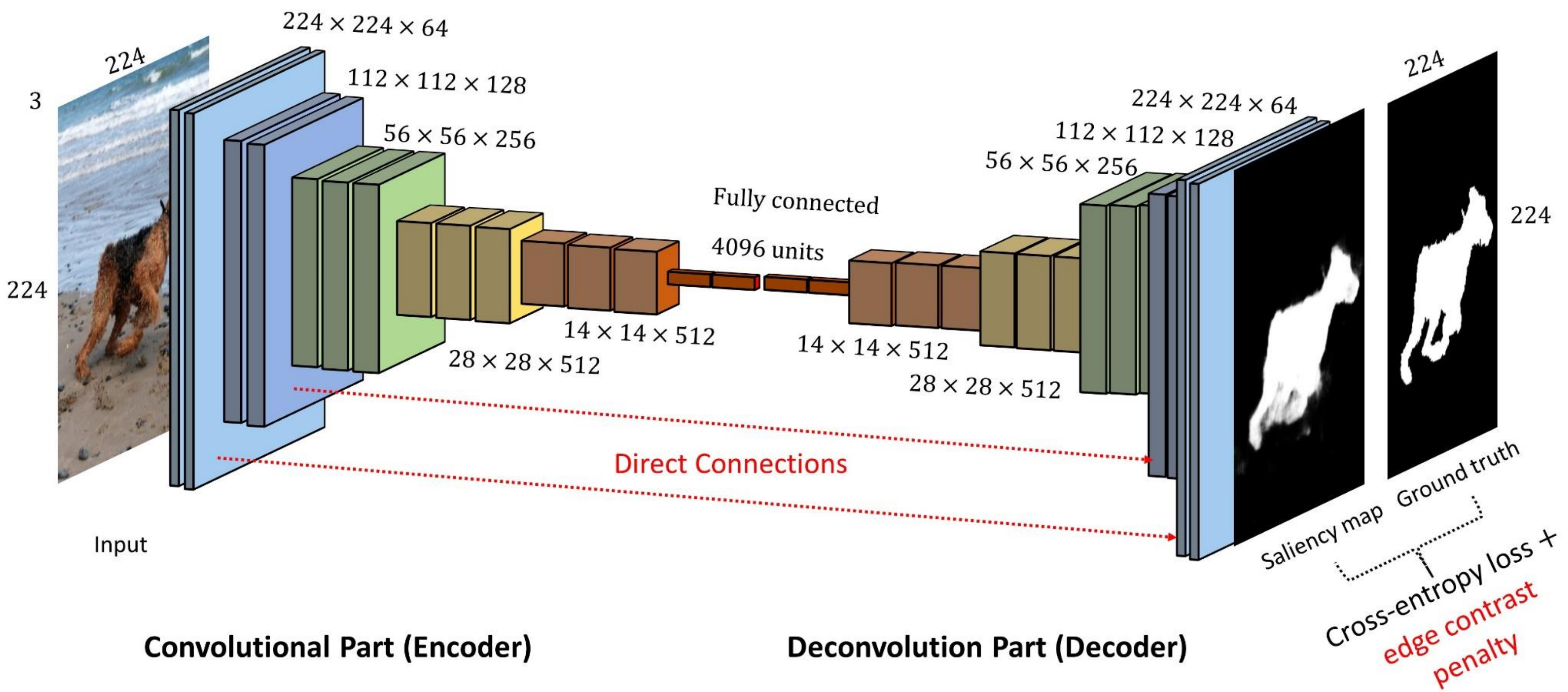
- CAE가 꼭 원본 이미지를 복원하는 목적으로만 사용되는 것은 아님
  - ✓ Image segmentation
    - 입력: 이미지
    - 출력: 픽셀 단위의 범주 (도로, 자동차, 하늘, 인도 등)



https://github.com/arahusky/Tensorflow-Segmentation

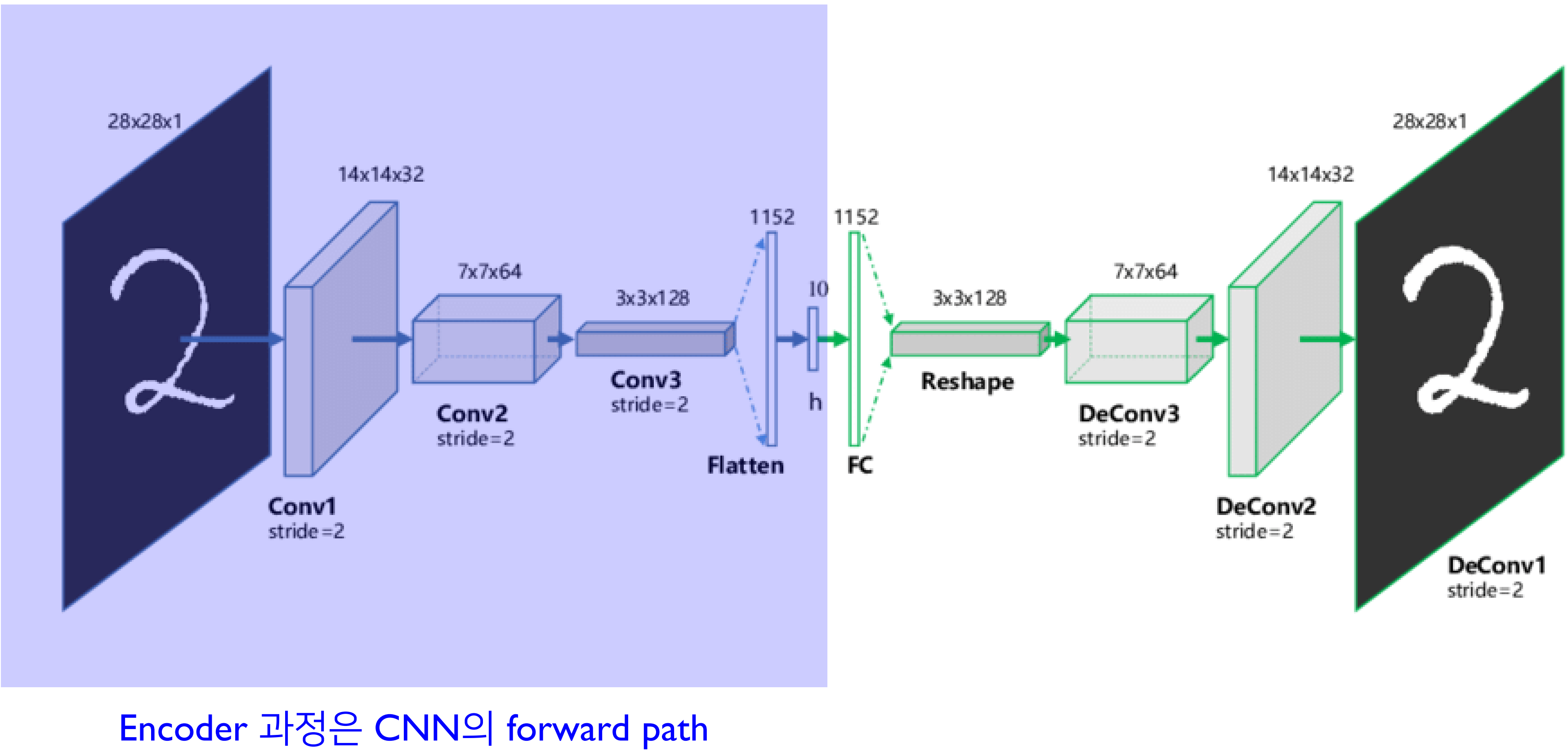
Convolutional Auto-Encoder

- CAE가 꼭 원본 이미지를 복원하는 목적으로만 사용되는 것은 아님
  - ✓ Saliency detection
    - 입력: 이미지
    - 출력: 가장 집중해야 하는 중요한 영역



Convolutional Auto-Encoder

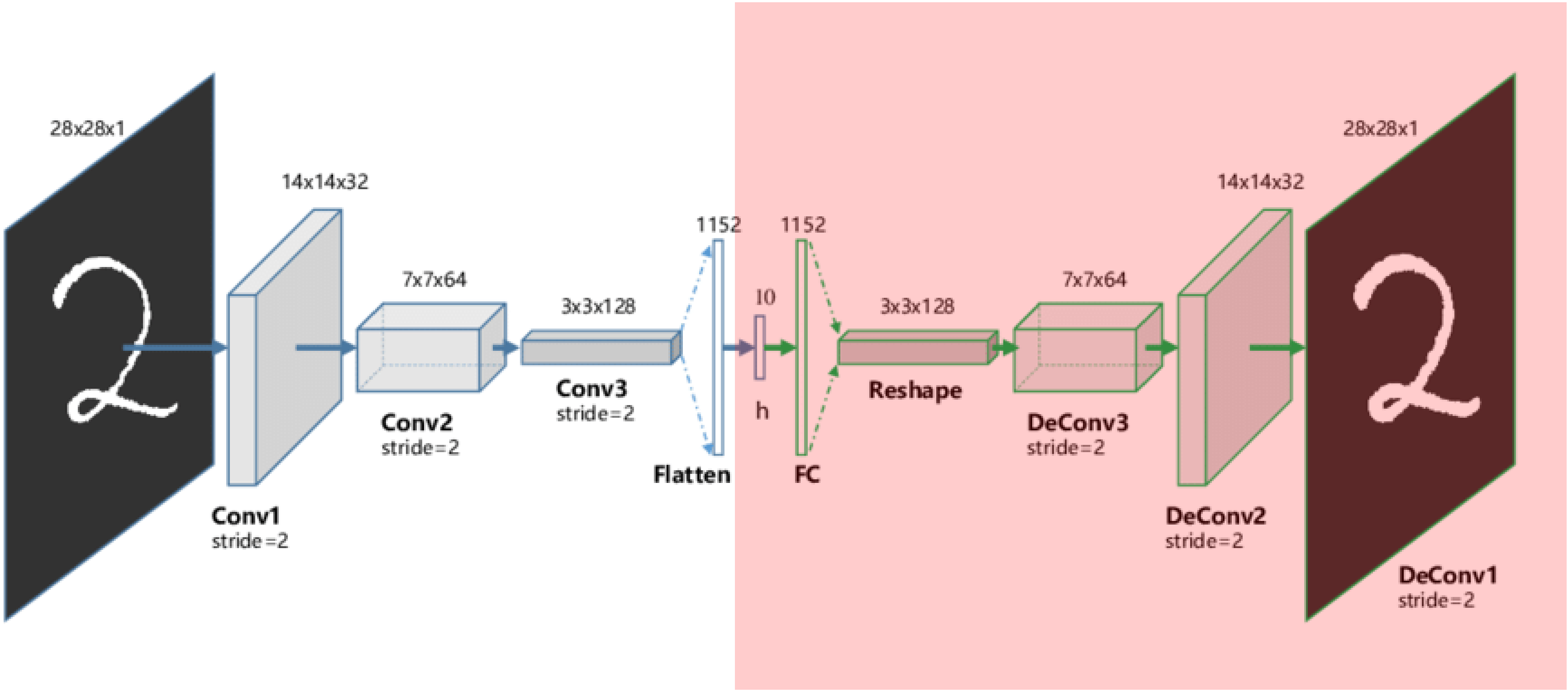
- CAE 고려사항: Decoder 학습 시 feature map의 크기를 어떻게 증가시킬 것인가?



https://www.researchgate.net/figure/The-structure-of-proposed-Convolutional-AutoEncoders-CAE-for-MNIST-In-the-middle-there_fig1_320658590

Convolutional Auto-Encoder

- CAE 고려사항: Decoder 학습 시 feature map의 크기를 어떻게 증가시킬 것인가?

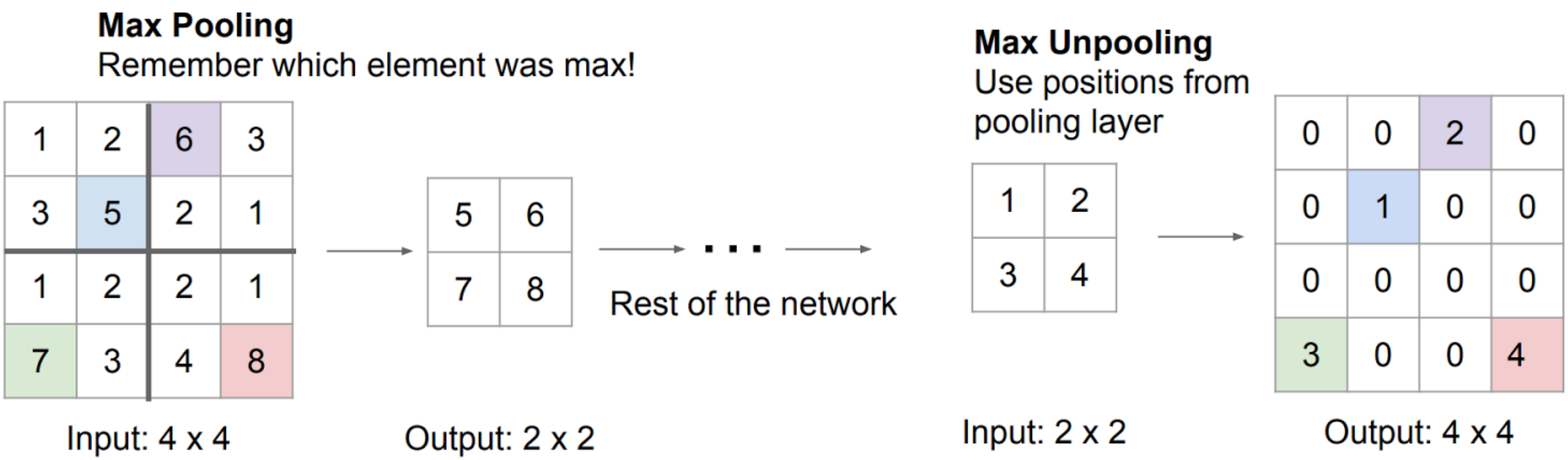


다시 어떻게 크기를 키우지?

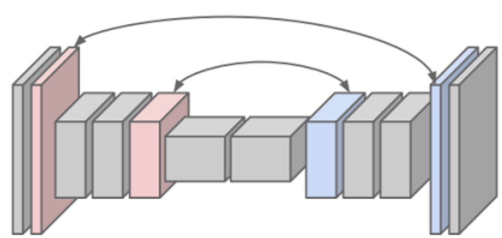
https://www.researchgate.net/figure/The-structure-of-proposed-Convolutional-AutoEncoders-CAE-for-MNIST-In-the-middle-there_fig1_320658590

Convolutional Auto-Encoder

- Unpooling:
  - ✓ Max pooling을 사용할 경우 해당 위치를 기억해 두었다가 그 정보를 사용



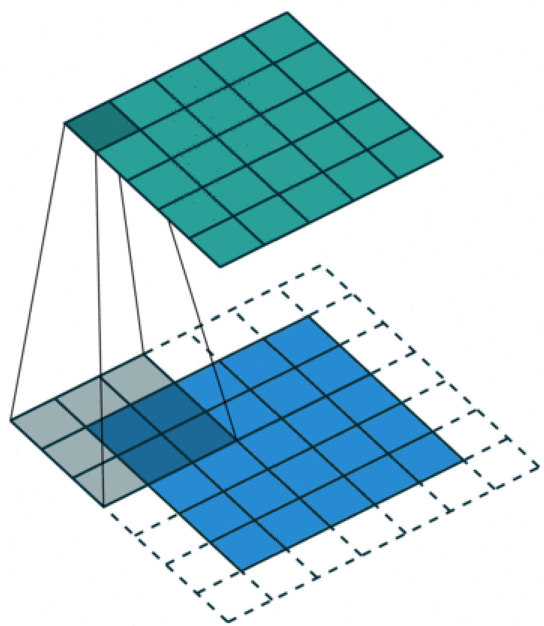
Corresponding pairs of downsampling and upsampling layers



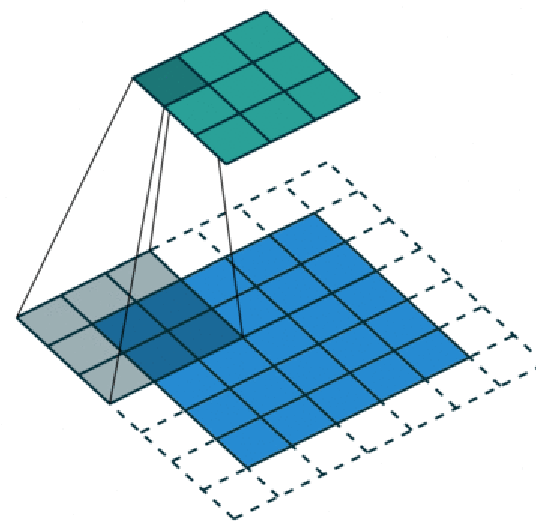
http://jihochoi.blogspot.com/2018/01/fully-convolutional-network-for.html

Convolutional Auto-Encoder

- Transpose convolution
  - ✓ Convolution과 같은 연산을 통해 feature map의 크기를 키우는 과정
    - ▪ Convolution



- Feature map: 3 by 3
- Padding: 1
- Stride: 1



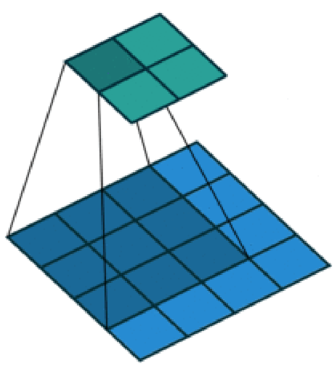
- Feature map: 3 by 3
- Padding: 1
- Stride: 2

http://www.cvc.uab.es/people/joans/slides_tensorflow/tensorflow_html/layers.html

Convolutional Auto-Encoder

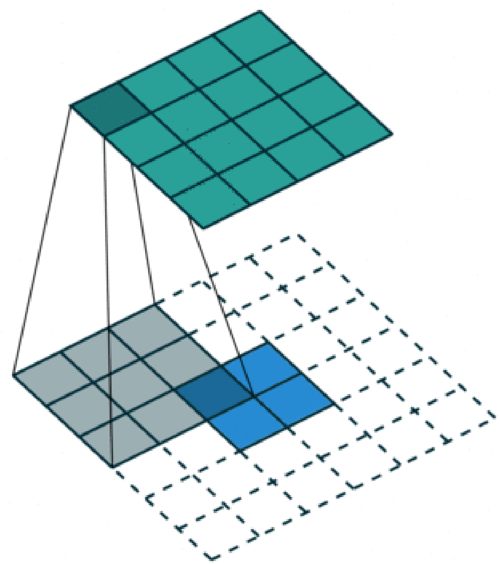
- Transpose convolution

Convolution



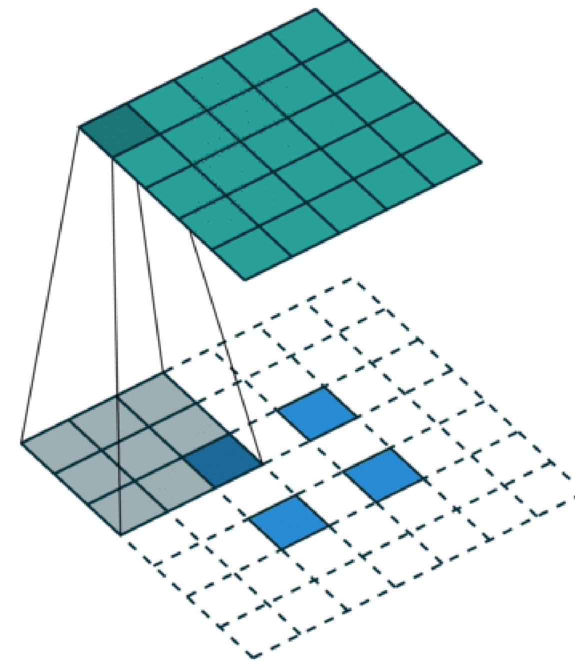
- Feature map: 3 by 3
- Padding: 0
- Stride: 1

Transpose Convolution



- Feature map: 3 by 3
- Padding: 0
- Stride: 1

Transpose Convolution

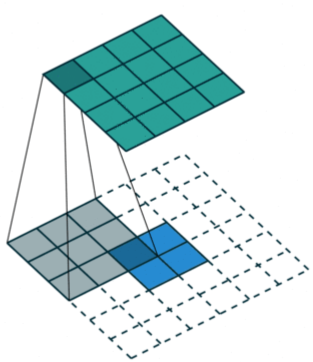


- Feature map: 3 by 3
- Padding: 0
- Stride: 2

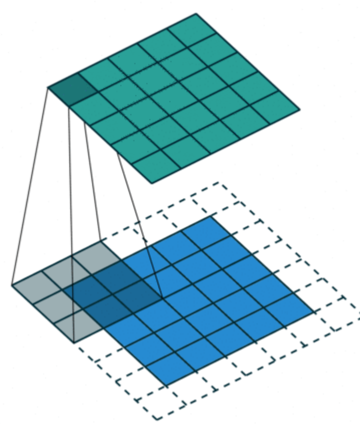
http://www.cvc.uab.es/people/joans/slides_tensorflow/tensorflow_html/layers.html

Convolutional Auto-Encoder

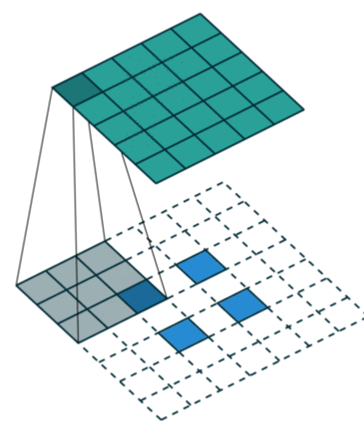
- (주의) Transpose convolution에서 padding의 의미
  - ✓ Padding = 1 in convolution: feature map 주변에 0의 값을 갖는 pad를 1pixel씩 덧댐
  - ✓ Padding = 0 in transpose convolution: feature map 주변에 0의 값을 갖는 pad를 (filter width(or height)-1) 만큼 덧대는 것 (3 by 3 filter size의 경우 2 pixels씩 덧댐)
  - ✓ Padding = 1 in transpose convolution: feature map 주변에 0의 값을 갖는 pad를 (filter width(or height)-1)-1 만큼 덧대는 것 (3 by 3 filter size의 경우 1 pixel씩 덧댐)



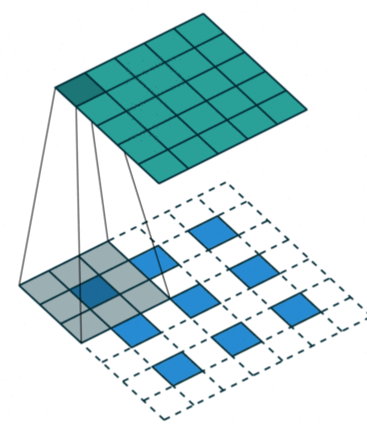
- Feature map: 3 by 3
- Padding: 0
- Stride: 0



- Feature map: 3 by 3
- Padding: 1
- Stride: 0



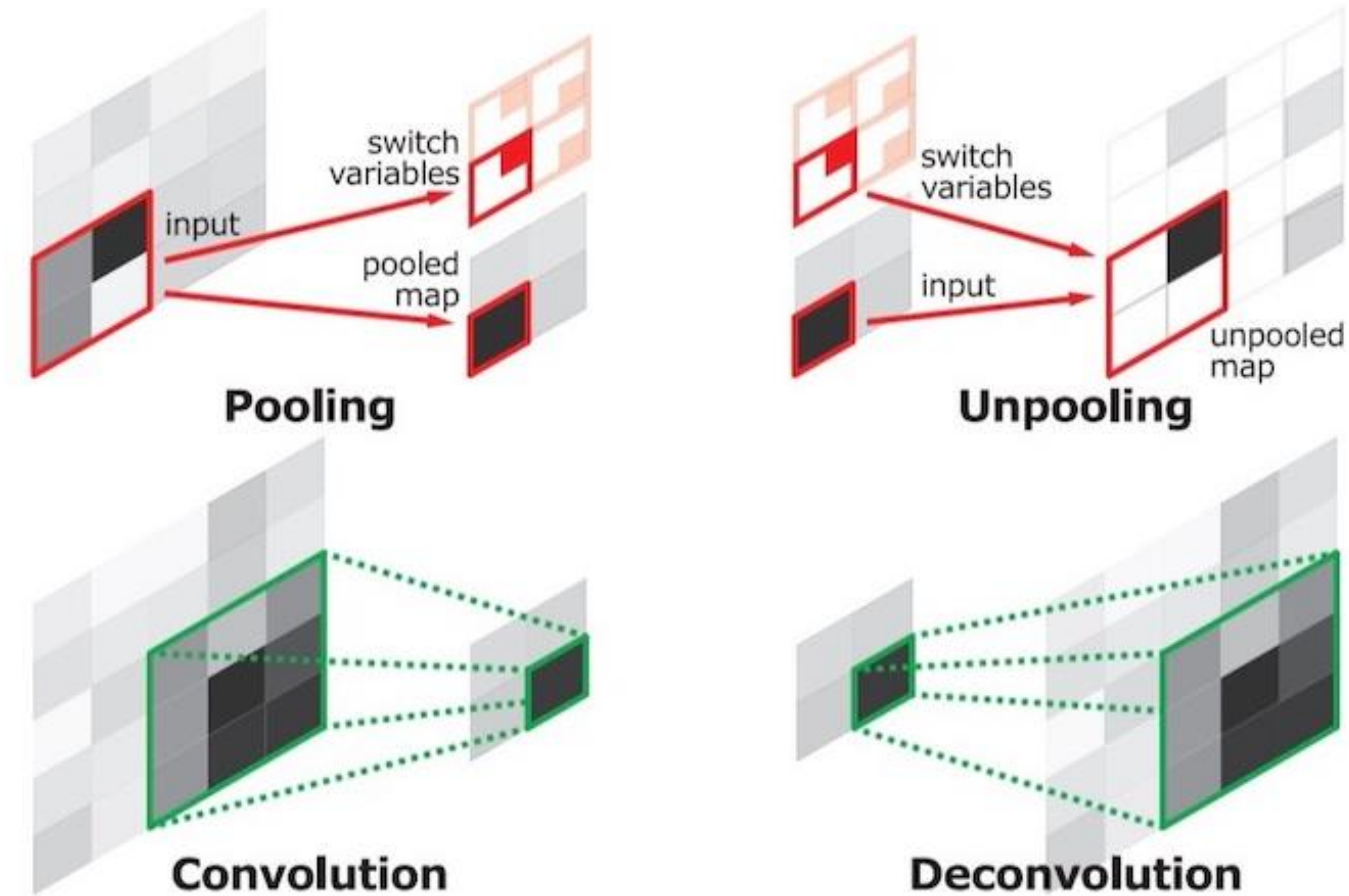
- Feature map: 3 by 3
- Padding: 0
- Stride: 1



- Feature map: 3 by 3
- Padding: 1
- Stride: 1

Convolutional Auto-Encoder

- Transpose convolution과 Unpooling의 차이



https://modulabs-biomedical.github.io/Learning_Deconvolution_Network_for_Semantic_Segmentation


ANY
questions?