방학 3주차

CNN&LSTM&RNN

NEKA

CNN

Convolution이란?

- 우리말로 '합성곱'이라고 함.
- 함수 A, B가 있을 때 A를 반전한 것=A(t-τ)과 B의 곱을 적분한 것

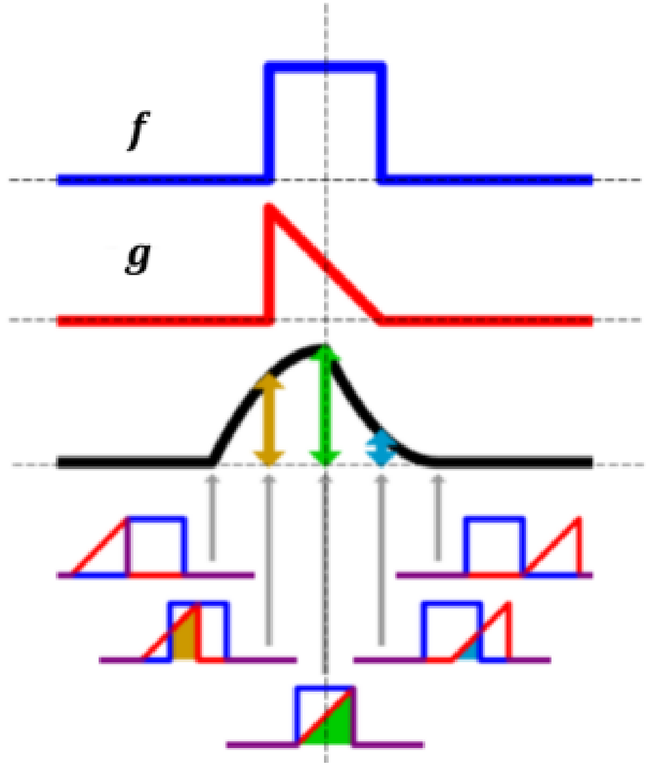
$$f(t) = f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t-\tau) d\tau, \quad -\infty < t < \infty$$

(전기신호에서의 개념)

CNN

Convolution 계산하기

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$



CNN

1D Convolution 계산하기

f : [3, 1, 2]
g = [3, 2, 1]
n=0

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

m	−2	−1	0	1	2	3	4	5
f[m]	0	0	3	1	2	0	0	0
g[0−m]	1	2	3	0	0	0	0	0

(0 * 1) + (0 * 2) + (3 * 3) = 9

CNN

1D Convolution 계산하기

f : [3, 1, 2]
g = [3, 2, 1]
n=1

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

m	−2	−1	0	1	2	3	4	5
f[m]	0	0	3	1	2	0	0	0
g[1−m]	0	1	2	3	0	0	0	0

(0 * 1) + (3 * 2) + (1 * 3) = 9

CNN

1D Convolution 계산하기

f : [3, 1, 2]
g = [3, 2, 1]
n=2

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

m	−2	−1	0	1	2	3	4	5
f[m]	0	0	3	1	2	0	0	0
g[2−m]	0	0	1	2	3	0	0	0

(3 * 1) + (1 * 2) + (2 * 3) = 11

CNN

1D Convolution 계산하기

f : [3, 1, 2]
g = [3, 2, 1]
n=3

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

m	−2	−1	0	1	2	3	4	5
f[m]	0	0	3	1	2	0	0	0
g[3−m]	0	0	0	1	2	3	0	0

(1 * 1) + (2 * 2) + (0 * 3) = 5

CNN

1D Convolution 계산하기

f : [3, 1, 2]
g = [3, 2, 1]
n=4

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

m	−2	−1	0	1	2	3	4	5
f[m]	0	0	3	1	2	0	0	0
g[4−m]	0	0	0	0	1	2	3	0

(2 * 1) + (0 * 2) + (0 * 3) = 2

CNN

1D Convolution 계산하기

f : [3, 1, 2]
g = [3, 2, 1]
n=4

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

f * g [n] = [9, 9, 11, 5, 2]

CNN

2D Convolution 계산하기

- x, y 방향으로 convolution을 진행함.
- 2D의 의미 : input은 2차원이 아니어도 되지만, result는 2차원이어야 함

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

g(x, y)

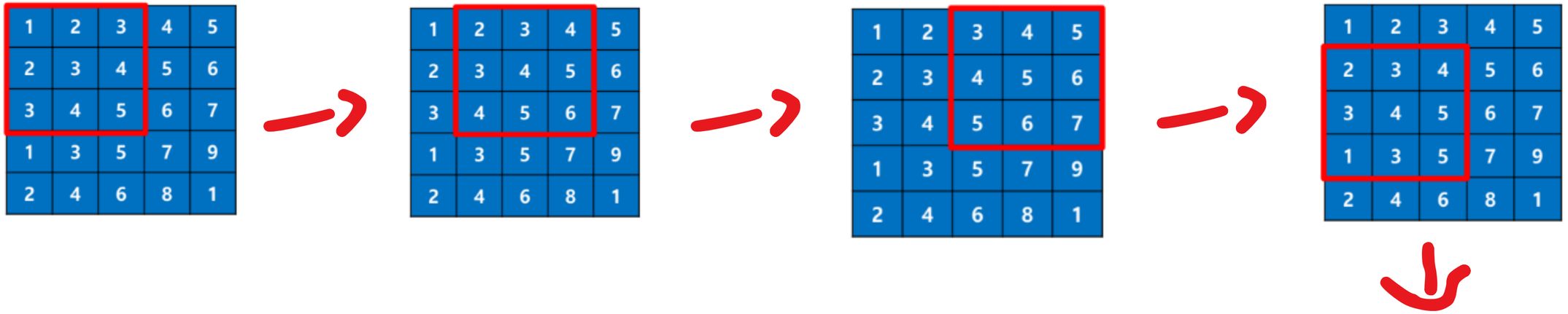
1	0	1
2	-1	3
2	1	0

g(x−n1, y−n2)

0	1	2
3	-1	2
1	0	1

CNN

2D Convolution 계산하기

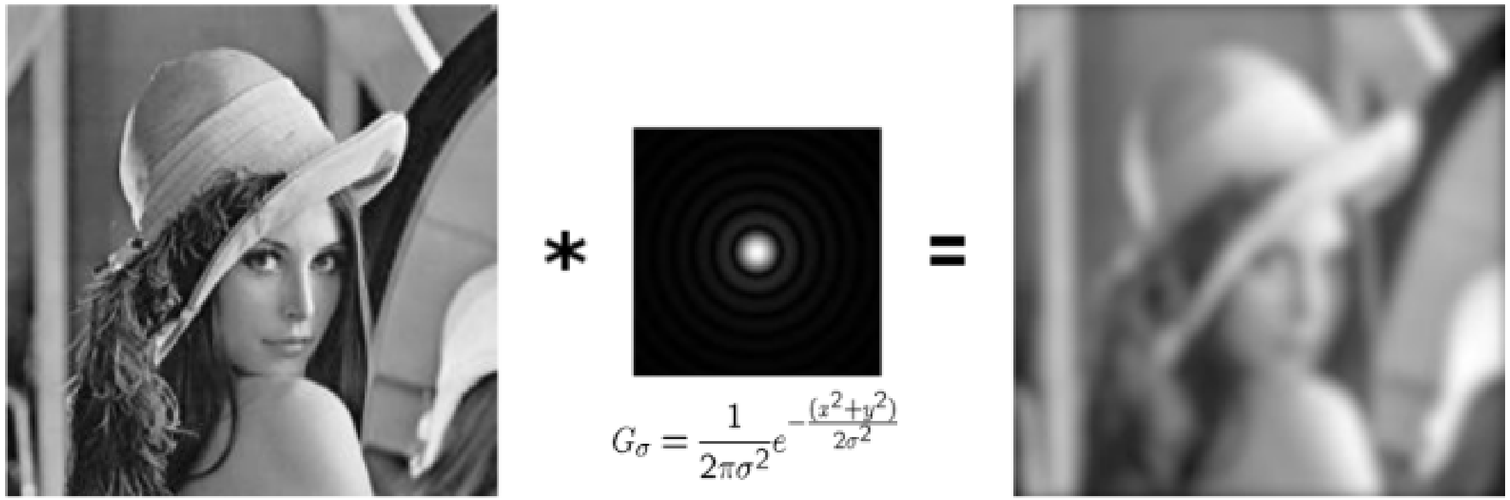


.....

CNN

머신러닝에서 convolution의 의미

- input이미지에 특정한 필터를 적용하는 것



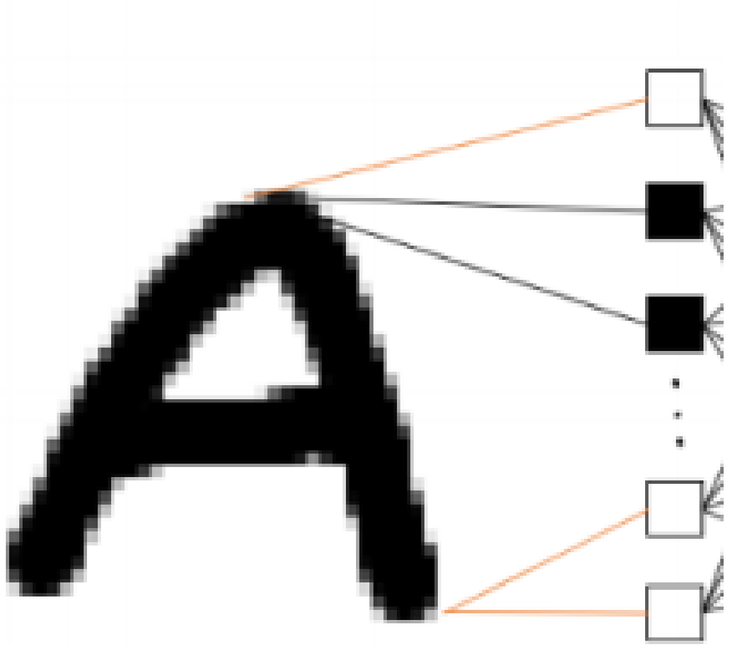
CNN

CNN이란?

- ConvNet이라고도 함.
- DNN의 일종으로, 시각 이미지를 분석할 때 보통 적용함.
- CNN은 MLP의 regularized 버전이라고도 할 수 있음

CNN

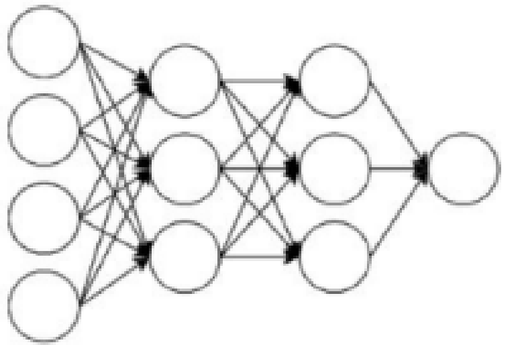
이미지 처리에서 ANN을 쓰기 어려운 이유



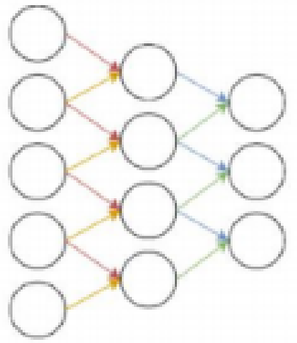
CNN

Paramter sharing

- filter을 여기저기 돌아가며 쓰기 대문에 파라미터를 줄여줌



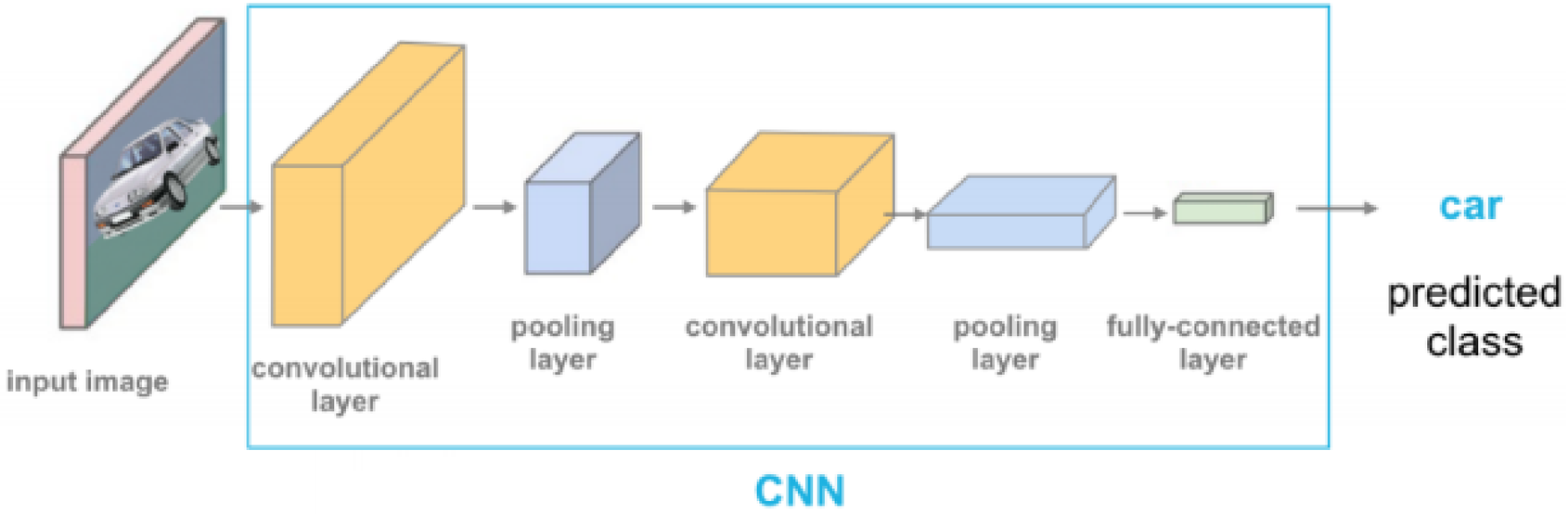
Feedforward neural network with three layers



Convolutional neural network with two layers

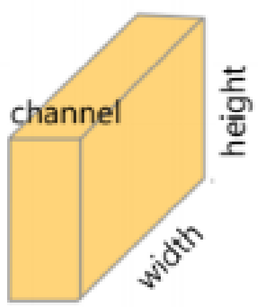
CNN

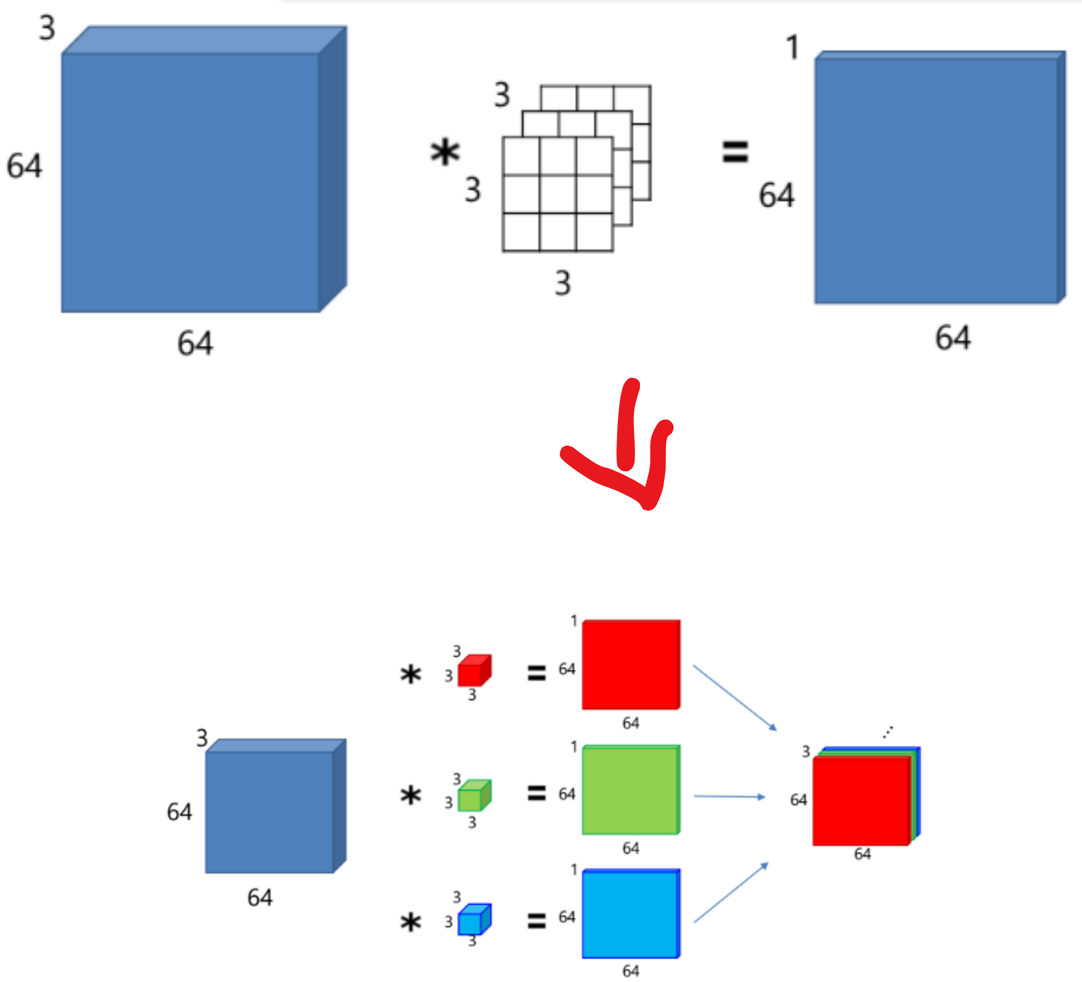
CNN 세부 구조


input image
convolutional layer
pooling layer
convolutional layer
pooling layer
fully-connected layer
car
predicted class
CNN

CNN

CNN 세부 구조 – Convolutional Layer

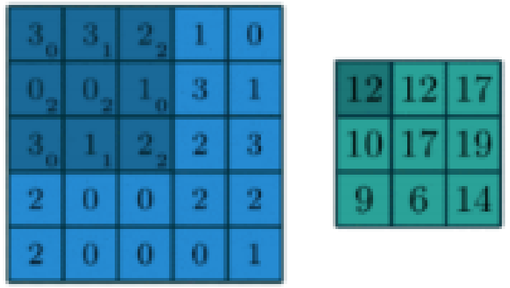


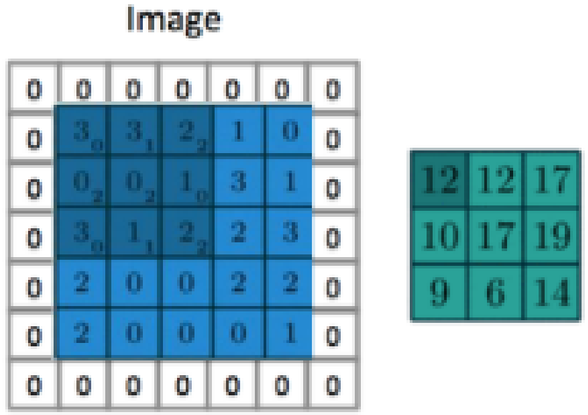


CNN

CNN 세부 구조 – Padding

- output size를 input size에 맞추기 위해 input 사이즈 외부에 0으로 구성된 열을 추가해줌
- padding의 크기는 (커널 길이 – 1)/2 (소수면 내림)로 함





CNN

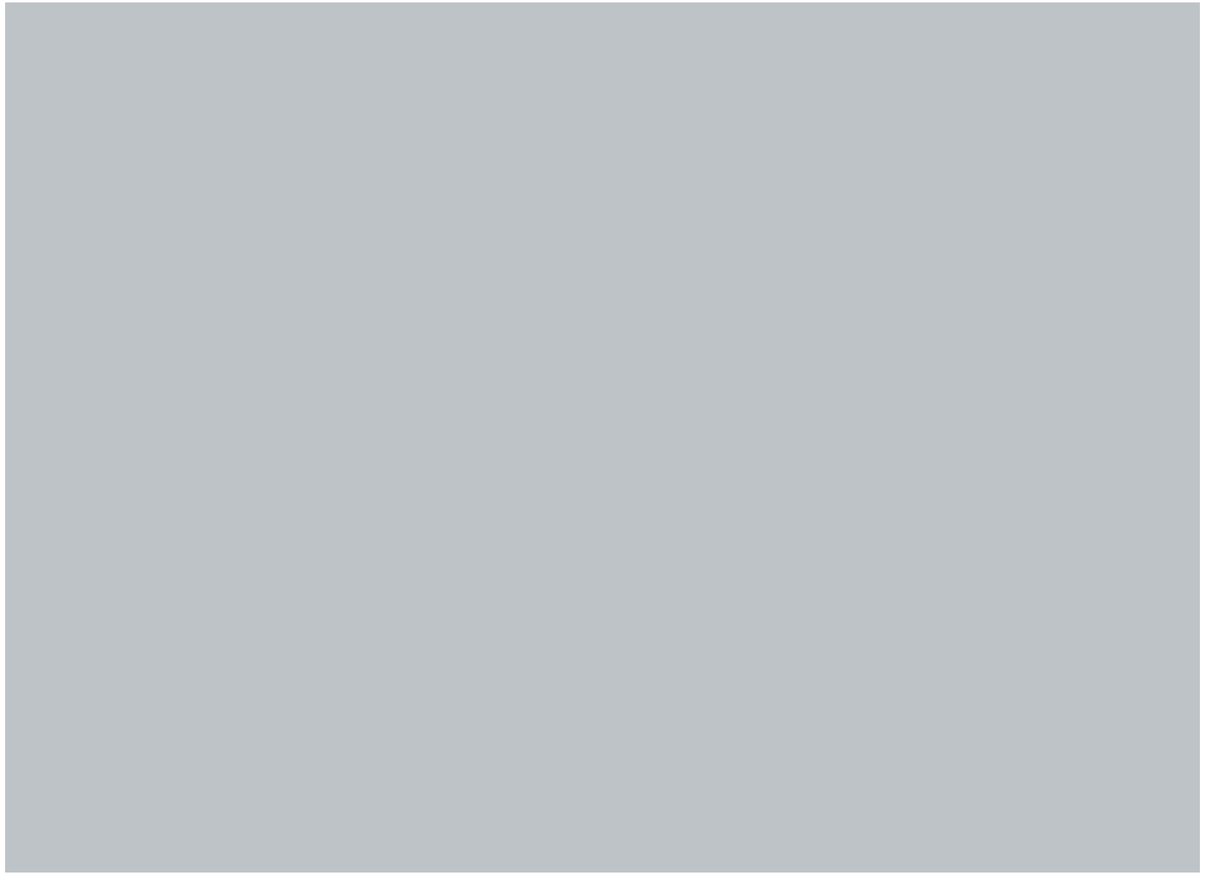
Receptive Field (수용 영역)

- 뉴런에서 Receptive Field는 한 뉴런을 흥분시킬 수 있는 자극의 영역을 말함
- 머신러닝에서는 filter의 크기를 말함
- Receptive Field를 키우는 방법
  - 커널의 크기를 키운다
    - Layer을 더 깊게 만든다.
    - Dilated Convolution
  - 이미지의 크기를 줄인다
    - Stride
    - Pooling

CNN

CNN 세부 구조 – Stride

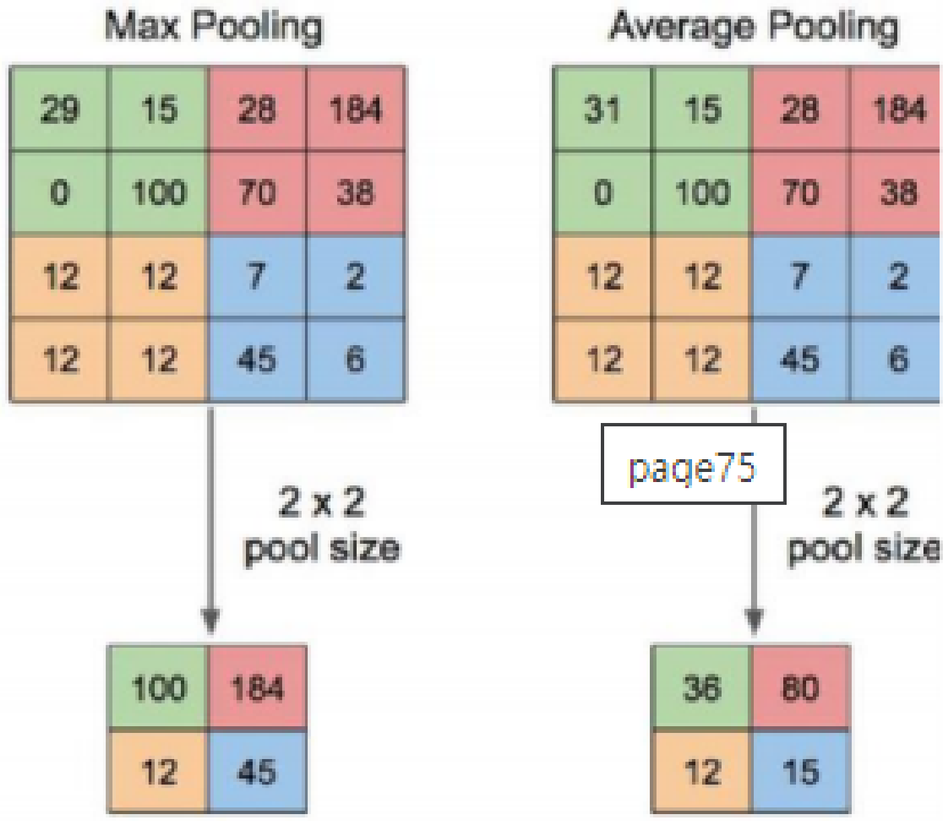
- 옆으로 갈 때 지나가는 개수를 결정



CNN

CNN 세부 구조 – Pooling

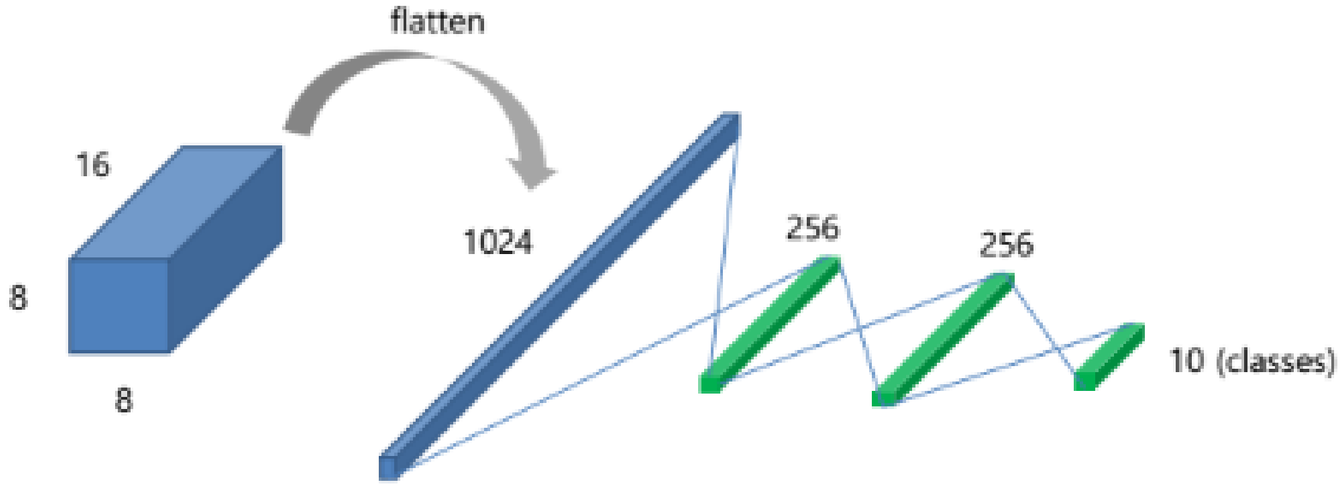
- 작은 규모로 나눠서 크기를 줄임
- Pooling의 종류 :
  - Max-Pooling : 범위 내 가장 큰 값 뽑기
  - Average-Pooling : 범위 내 평균값 뽑기



CNN

Fully-connected Layer

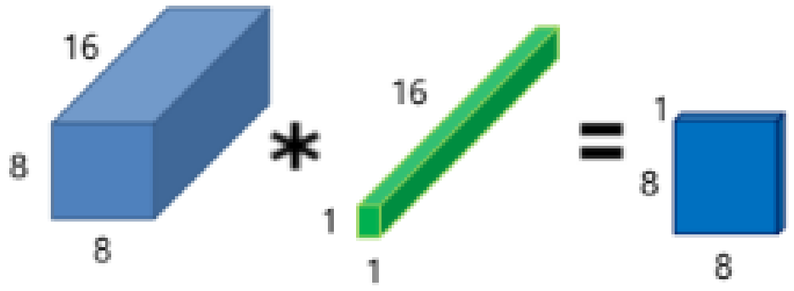
- flatten, linear, classification layer라고 부르기도 함.
- 일반적으로 마지막 레이어 뒤에 붙여서 분류 작업을 할 때 씀.



CNN

1x1 convolution

- 크기가 1X1인 필터를 사용하는 레이어
- 장점
  - 채널수를 마음대로 결정할 수 있음
  - 연산량을 감소시킬 수 있음
  - 모델을 비선형적으로 만들어서 더 복잡하게 만들어줌



RNN

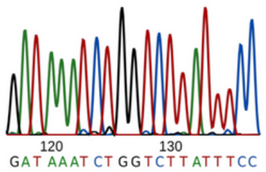
Recurrent란

- RNN에서 Recurrent는 repetitive의 의미를 가짐
  - Input 데이터는 sequential(순차적)
  - parameter는 순차적임

RNN

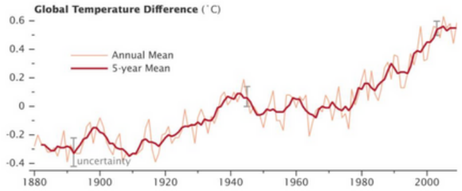
Sequential Data

- data point : 각 시가에 따른 vector x의 sequence
- batch data : 서로 다른 길이를 가진 sequence
- label : scalar, vector, 혹은 sequence
- 서로 다른 종류의 sequence들도 label이 될 수 있음



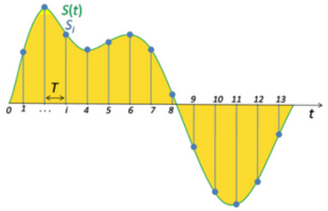
DNA 염기 서열

(Sequential Data)



세계 기온 변화

(Temporal Sequence)



샘플링된 소리 신호

(Time Series)

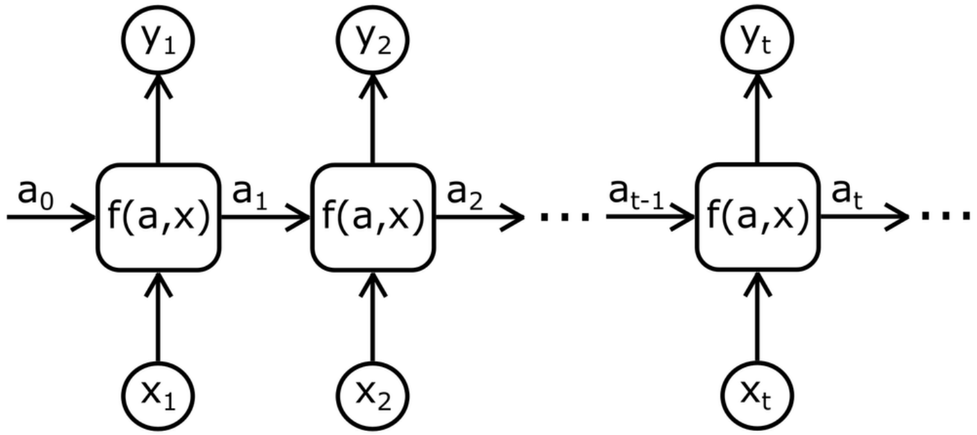
순서가 의미가 있으며, 순서가 달라질 경우 의미가 손상되는 데이터를 순차 데이터라고 한다.

시간적 의미가 있는 경우 Temporal Sequence라고 하며, 일정한 시간차라면 Time Series라고 한다.

RNN

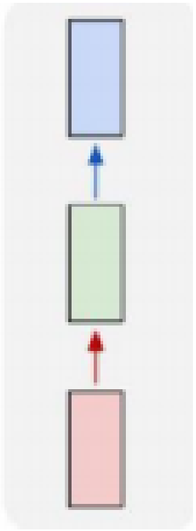
RNN이란

- 입력과 출력을 시퀀스 단위로 처리하는 모델

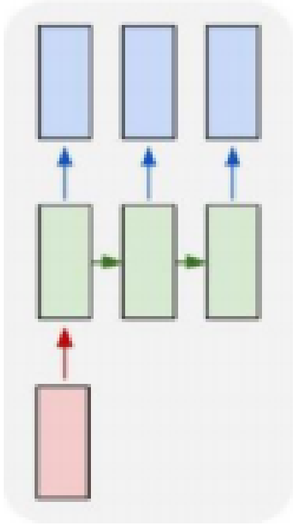


RNN

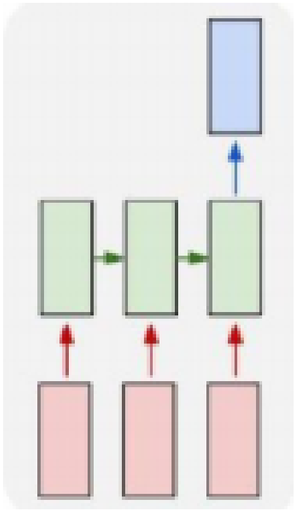
RNN 구조들



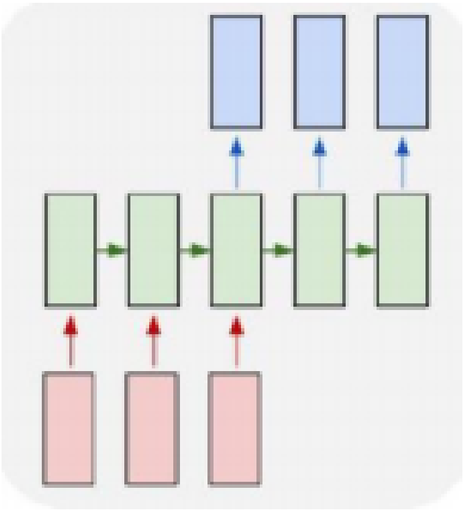
ONE TO ONE
(기본형)



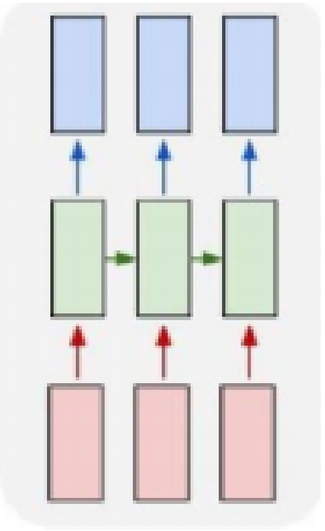
ONE TO MANY



MANY TO ONE



MANY TO MANY
1



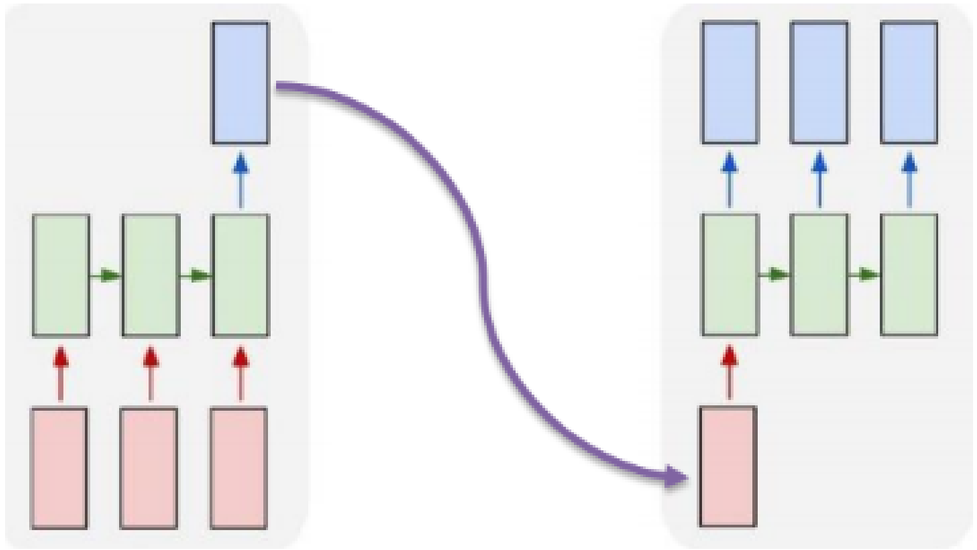
MANY TO MANY
2

RNN

seq2seq

- Many-to-One 와 One-to-Many를 합한 것
- 주로 번역기에서 사용함

인코더



디코더

RNN

RNN의 input과 output

- RNN의 INPUT은
  - sequence인 데이터
  - 이전의 hidden state
- RNN의 OUTPUT은
  - 현재의 hidden state
  - 처리된 결과물 (마지막)

RNN

hidden state와 Memory cell

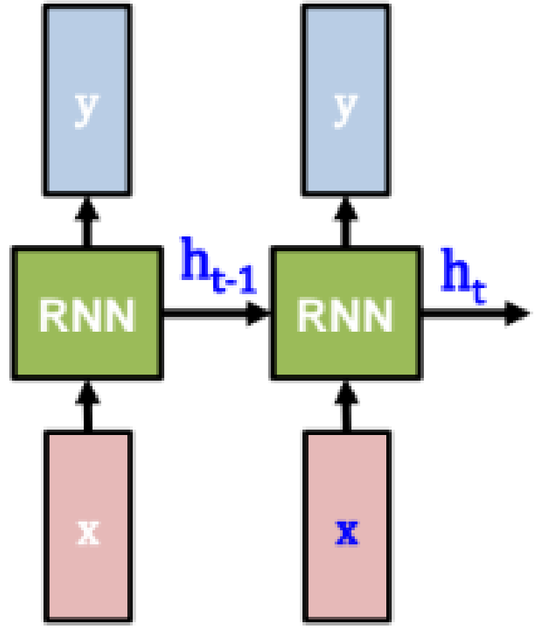
- hidden state는 다음 cell로 보낼 값

ex)

Some function with parameters W

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

New state　Old state　Input vector at some time step



- Memory Cell은 RNN에서 하나의 Layer을 지칭하는 말

RNN

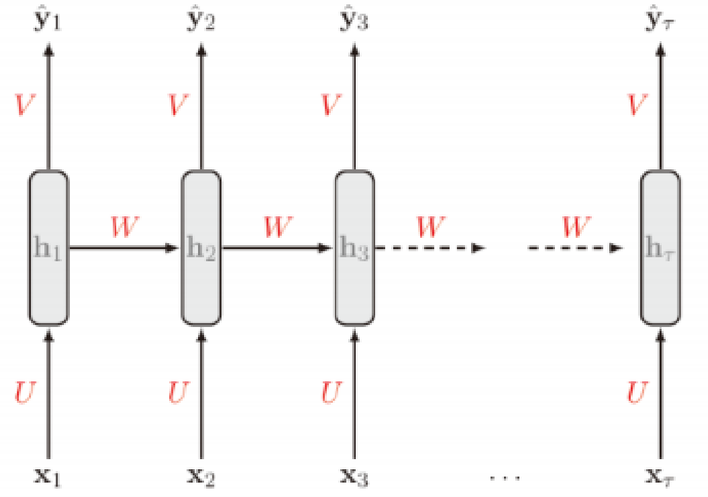
RNN의 Computational Graph

$$h_t = \psi(W h_{t-1} + U x_t)$$

$$\hat{y}_t = \phi(V h_t)$$

Ψ(psi) : tanh 등

ϕ(phi) : softmax 등



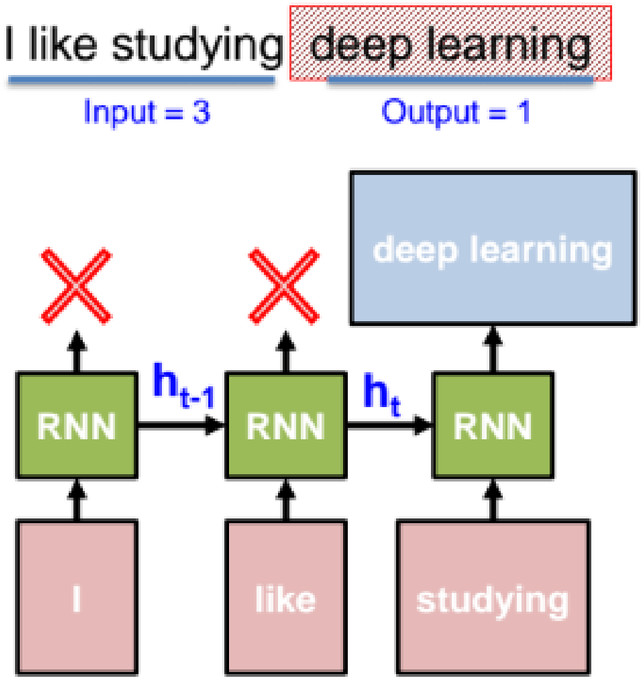
U : 파라미터인 tensor 값 = 가중치 W_xh

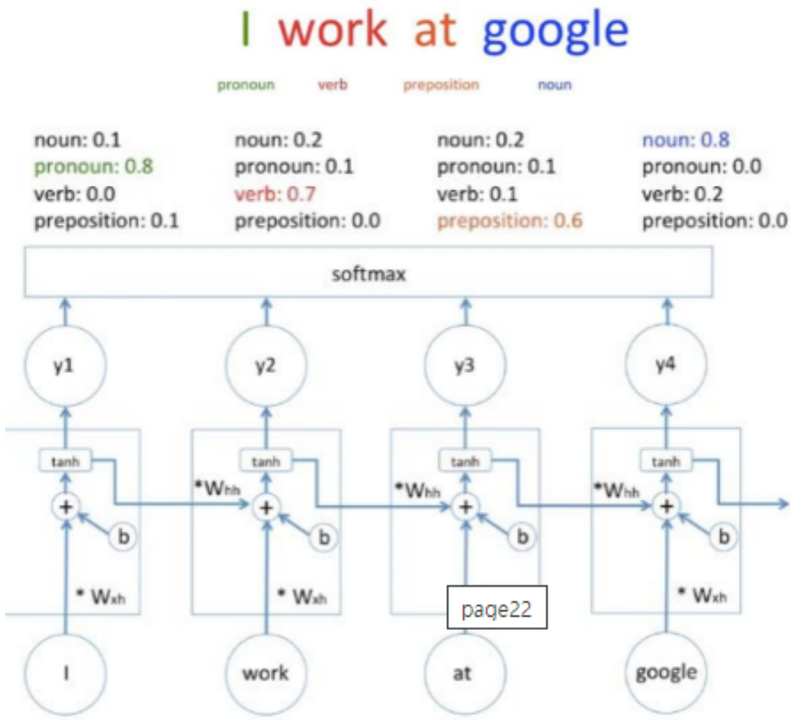
V : 파라미터인 tensor 값 = 가중치 W_hy

W : 파라미터인 tensor 값 = 가중치 W_hh

RNN

RNN 적용 예시





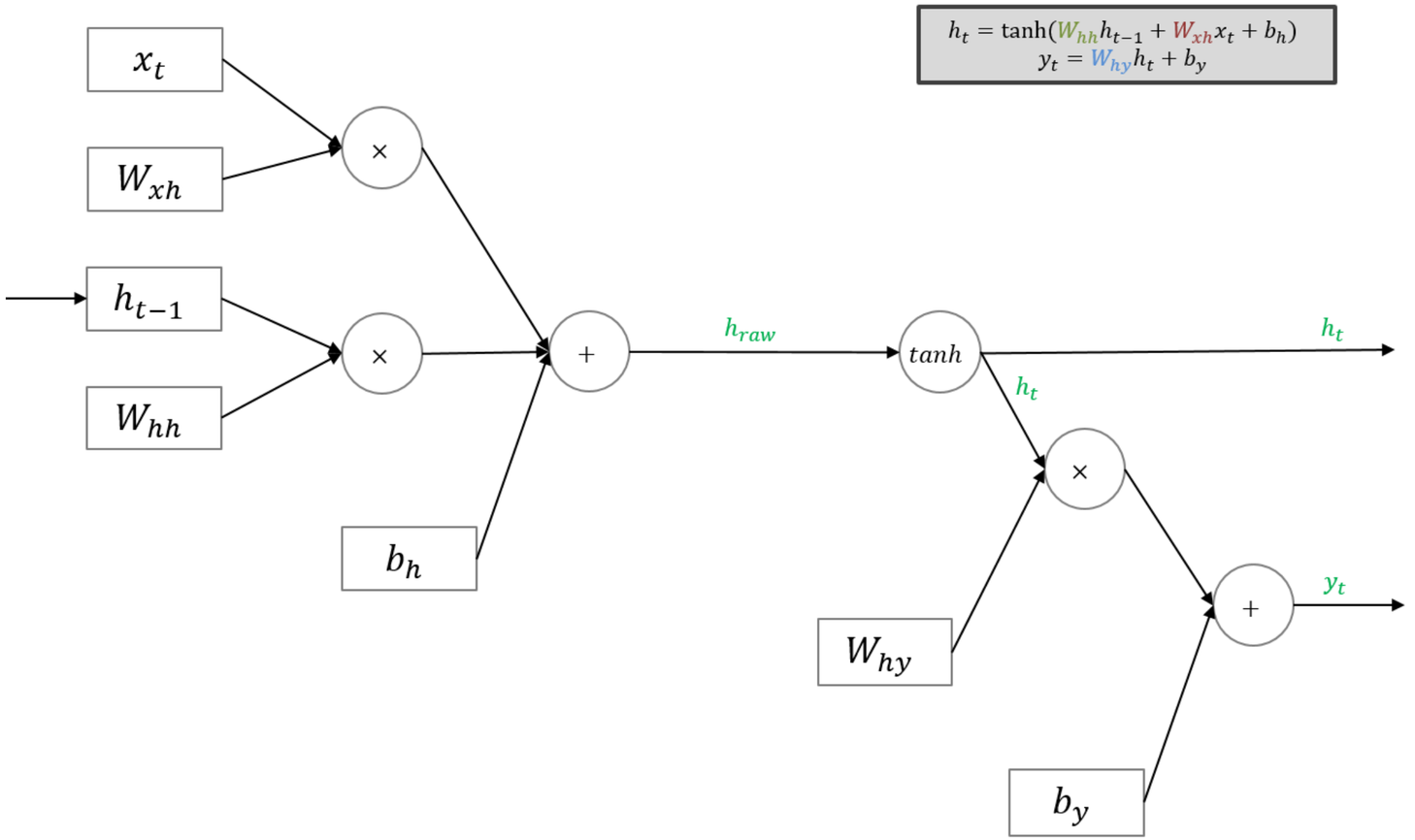
POS TAGGING

RNN

RNN의 Forward Propagation

특징 :

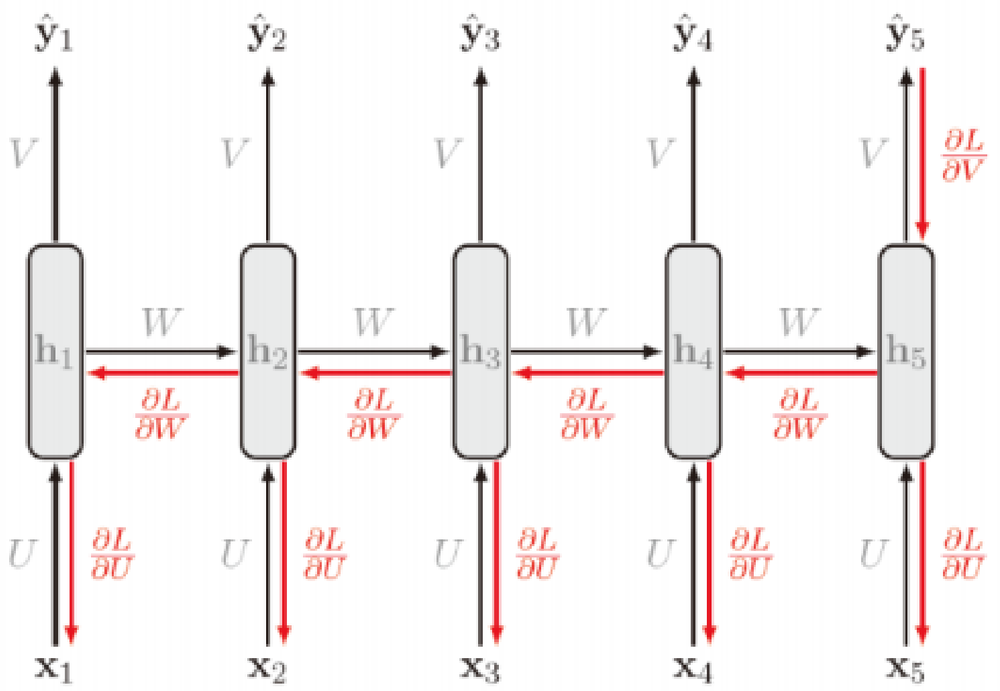
- CAN'T BE PARALLELIZED
- loss function은 전체 합치기



RNN

RNN의 Backward Propagation

- **이러한 과정을 BPTT라고 함**



RNN

RNN의 한계점

- Gradienet Vanishing / Exploding
  - 파라미터가 공유되기 때문에

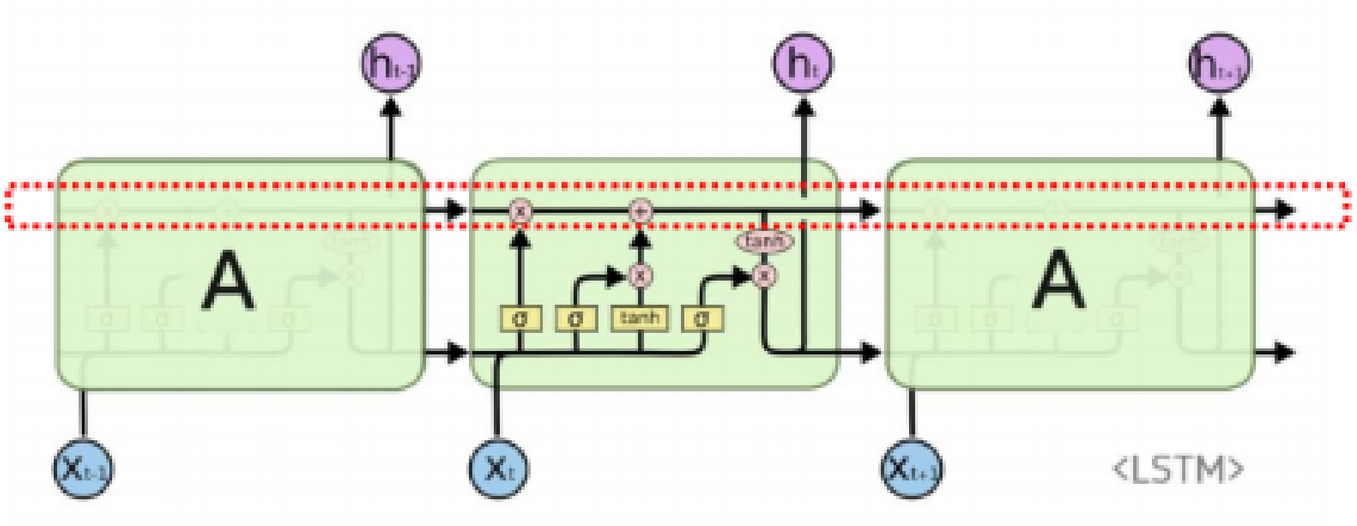
(1) parameter가 1보다 크면 : gradient가 무한대로 가서 gradient exploding

(2) parameter가 1보다 작으면 : gradient가 0으로 가서 gradient vanishing

LSTM

LSTM 이란

- Long-short term memory
- 메모리를 여러 개로 나눔
- 컨베이어 벨트처럼 행동하며 state를 잘 유지해서 전파가 잘 되도록 함



LSTM

LSTM Cell

- (개념 설명)

forget gate $f_t = \sigma(W_f x_t + V_f h_{t-1} + b_f)$

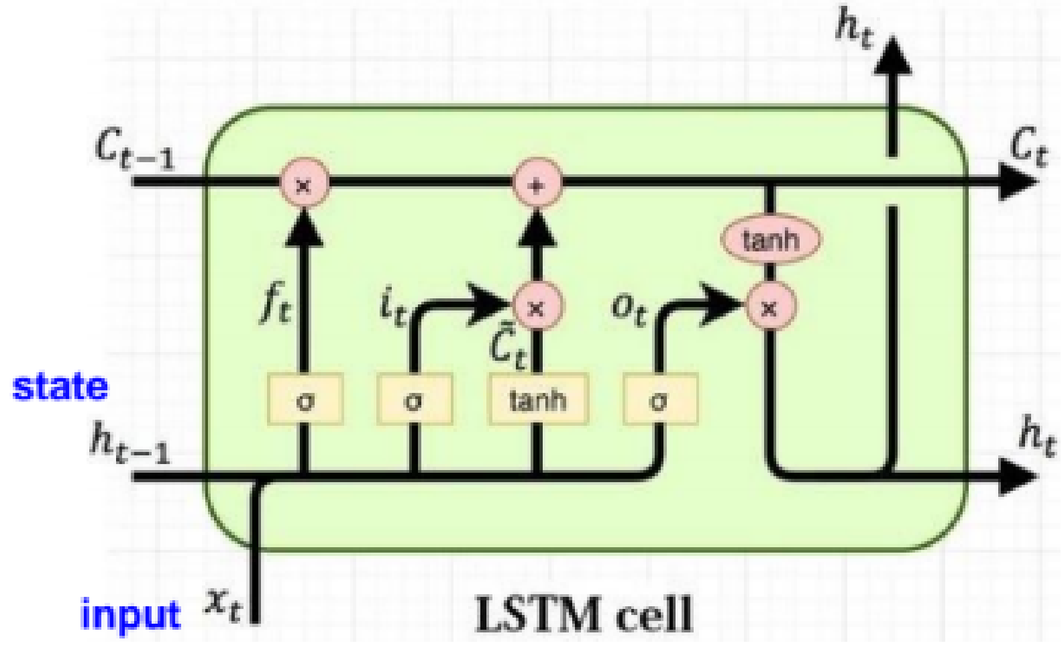
input gate $i_t = \sigma(W_i x_t + V_i h_{t-1} + b_i)$

$\tilde{C}_t = \tanh(W_g x_t + V_g h_{t-1} + b_g)$

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

gate $o_t = \sigma(W_o x_t + V_o h_{t-1} + b_o)$

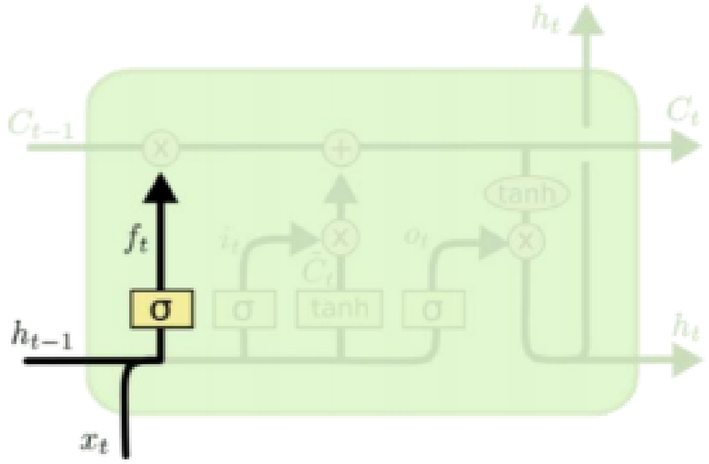
output $h_t = o_t \odot \tanh(C_t)$



LSTM

LSTM Cell - Forget Gate

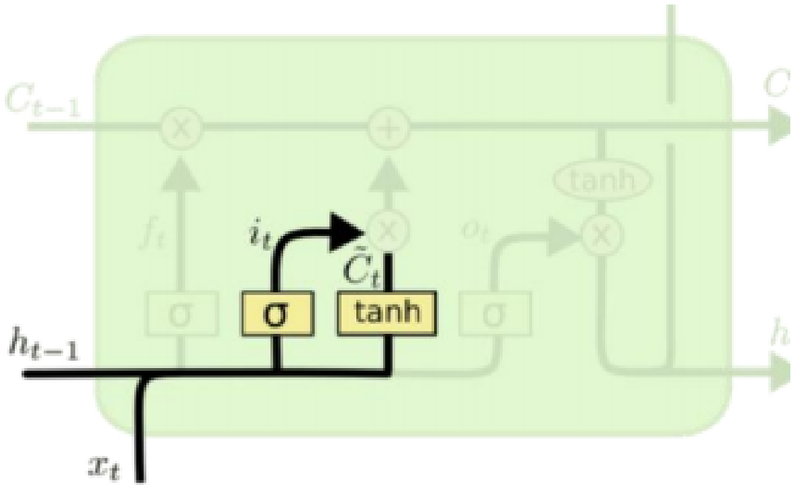
- 과거의 정보를 얻기 위한 게이트
- h_(t-1)와 x_t에 대해 sigmoid를 씌워준 값.
- Output 종류
  - 0 : forget
  - 1 : preserve (유지)



LSTM

LSTM Cell - Input Gate

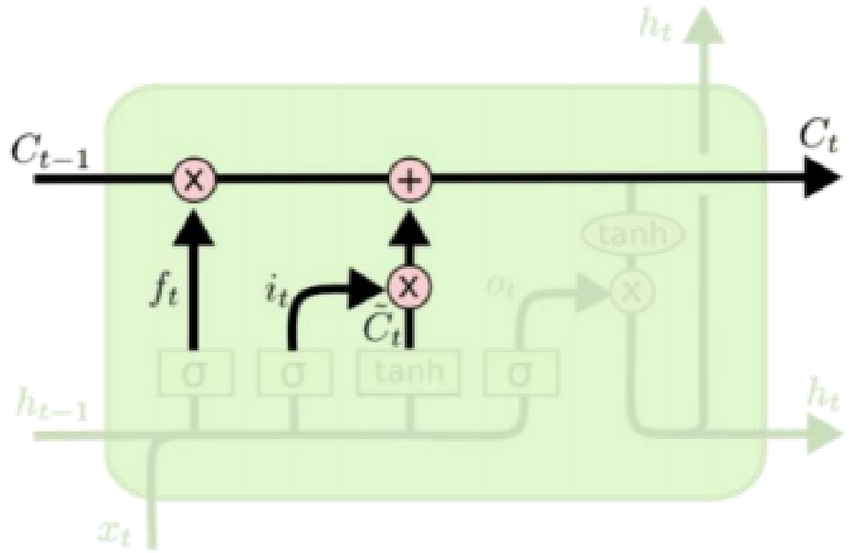
- 현재의 정보를 기억하려는 게이트
- h_(t-1)와 x_t에 대해 sigmoid와 tanh를 씌워준 값으로 hadamard 곱을 함



LSTM

LSTM Cell - Update Gate

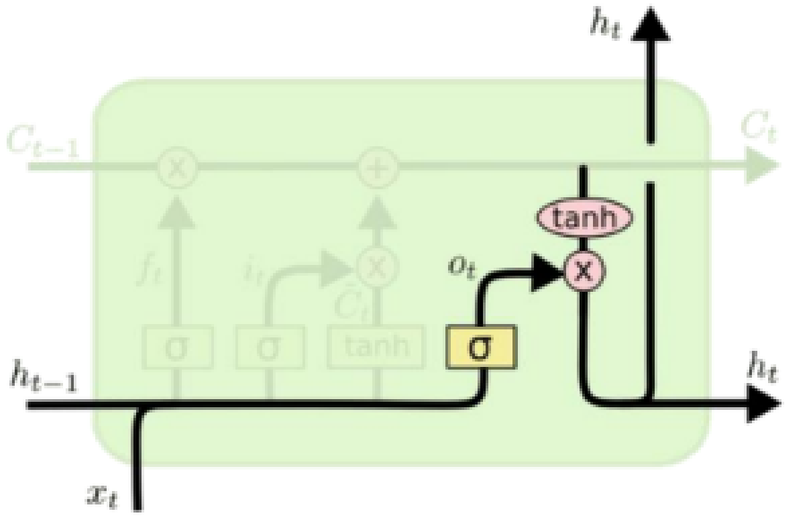
- 얼마나 업데이트를 할 건지 결정하는 게이트
- forget gate와 input 게이트를 hadamard 연산을 통해 계산함



LSTM

LSTM Cell – Output Gate

- 무엇을 결과물로 내보낼지를 결정함



LSTM

LSTM이 Vanishing Gradient을 해결한 방식

- f가 sigmoid의 output이 되므로 explode는 완전히 해결함
- f가 sigmoid의 output이 되므로 1에 가까울수록 vanishing이 최소화됨
- 완벽하게 Vanishing Gradient를 해결하진 않았지만, 더 오래 기억할 수 있도록 함

과제

인터넷에 있는
CNN, RNN, LSTM 코드
찾아보고 돌려보기

NEKA

THANK YOU