딥러닝팀

1팀

이수경
이승우
이은서
주혜인
홍현경

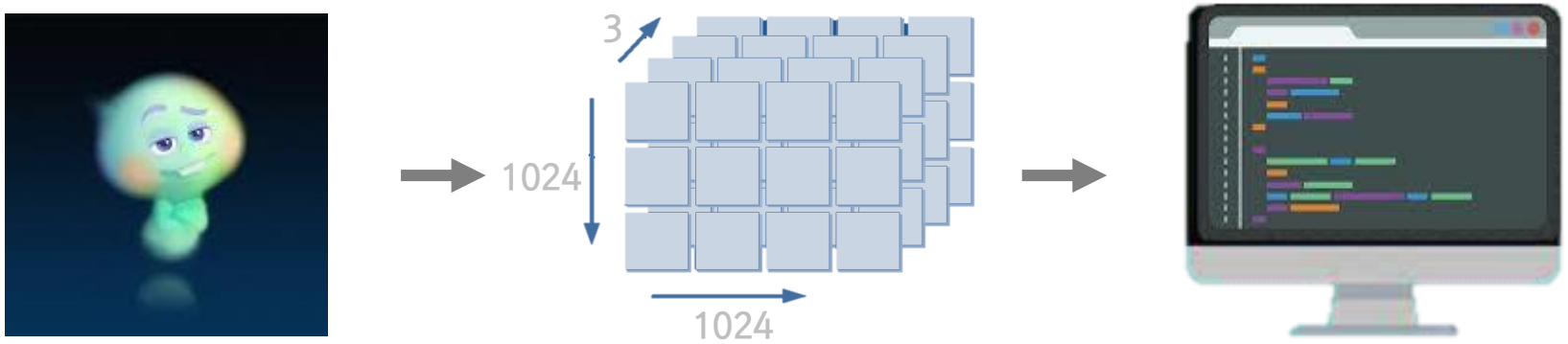
INDEX

1

CNN

1 합성곱 신경망(CNN)

- 이미지 데이터의 특징



컴퓨터는 이미지를 행렬 형태로 인식

ex) (1024, 1024, 3)는 순서대로 가로, 세로, 색깔을 나타냄

1 합성곱 신경망(CNN)

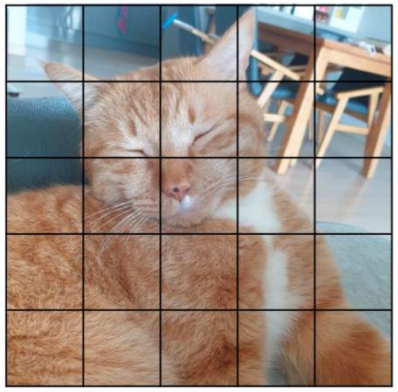
이미지 데이터의 특징

위치 정보

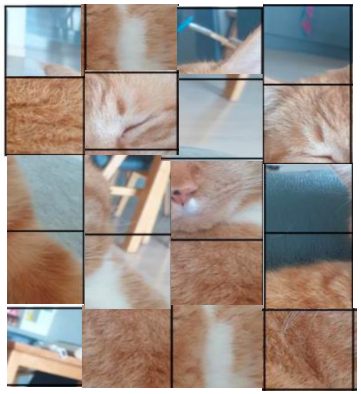
특정 픽셀 옆에 어떤 픽셀이 위치했는지가 중요한 정보를 전달함

채널

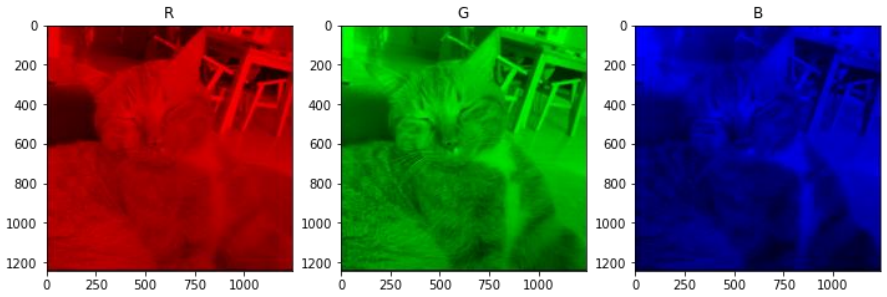
R, G, B 세 가지 색을 담당하는 세 개의 채널로 구성되어 있음



호양이다!



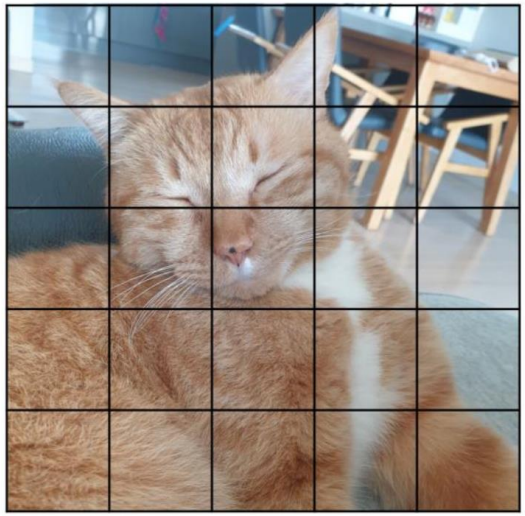
????



1 합성곱 신경망(CNN)

- 이미지 데이터의 특징

DNN의 한계 1) 공간 정보 손실



실제로는 이러한 이미지 형태가 아닌,
Scalar 값을 가지지만 우리의 이해를 위해
이미지 형태로 두고 보겠습니다

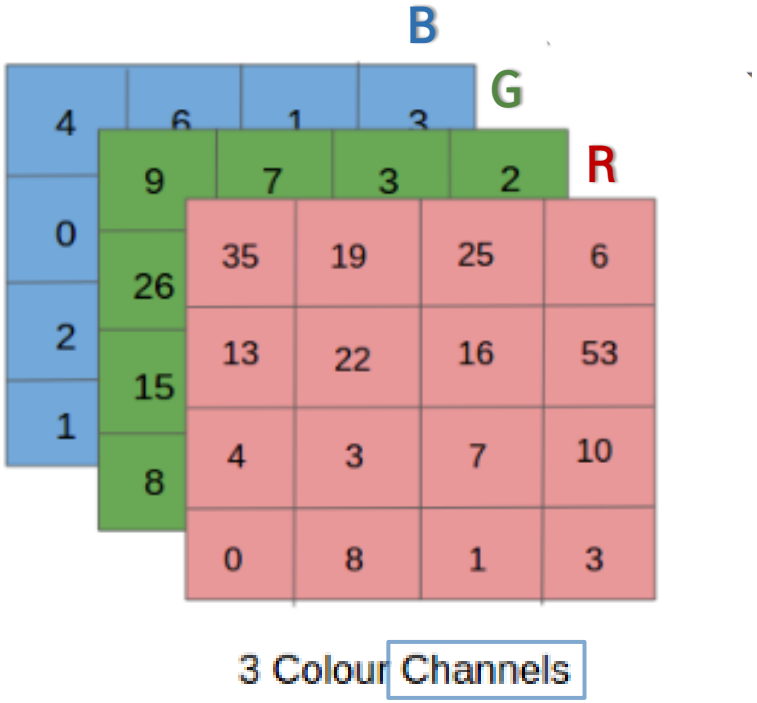
DNN은 입력 값으로 벡터 형태의 데이터만 받음

행렬을 벡터로 변환하는 **flatten** 과정 필요

➔ **위치 정보**를 잃게 됨

- 이미지 데이터의 특징

DNN의 한계 2) 채널 별 특성 파악 불가



색상에 대한 정보를 담아 두는 곳

이미지는 RGB 각 채널마다
값의 특성이 다름

그러나 DNN은 각 채널의 값들을 **평균** 내어
하나의 채널 사용

➔ **채널 별 특성** 파악 어려움

1 합성곱 신경망(CNN)

- 이미지 데이터의 특징

DNN의 한계 2) 채널 별 특성 파악 불가

이미지 데이터를 DNN으로 다루는 경우

공간 정보 손실

채널 특성 파악 어려움

과도하게 큰 연산량

이미지는 RGB 각 채널마다 값의 특성이 다름

그러나 DNN은 각 채널의 값들을 **평균** 내어 **하나의 채널** 사용

채널 별 특성 파악 어려움

색상에 대한 정보를 담아 두는 곳

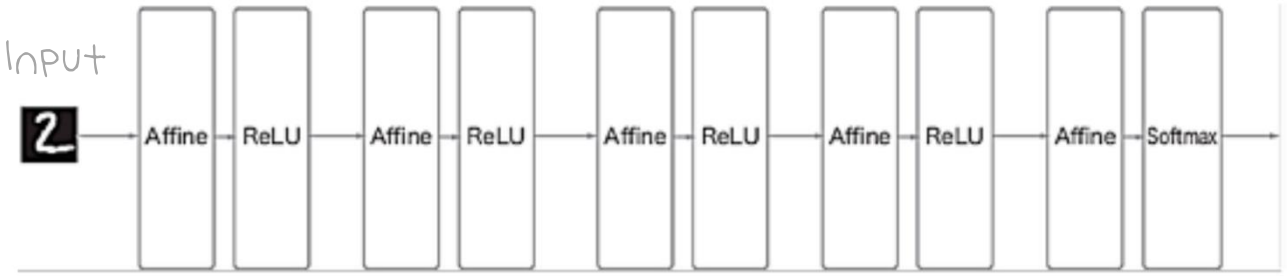
CNN 알고리즘 이용

1 합성곱 신경망(CNN)

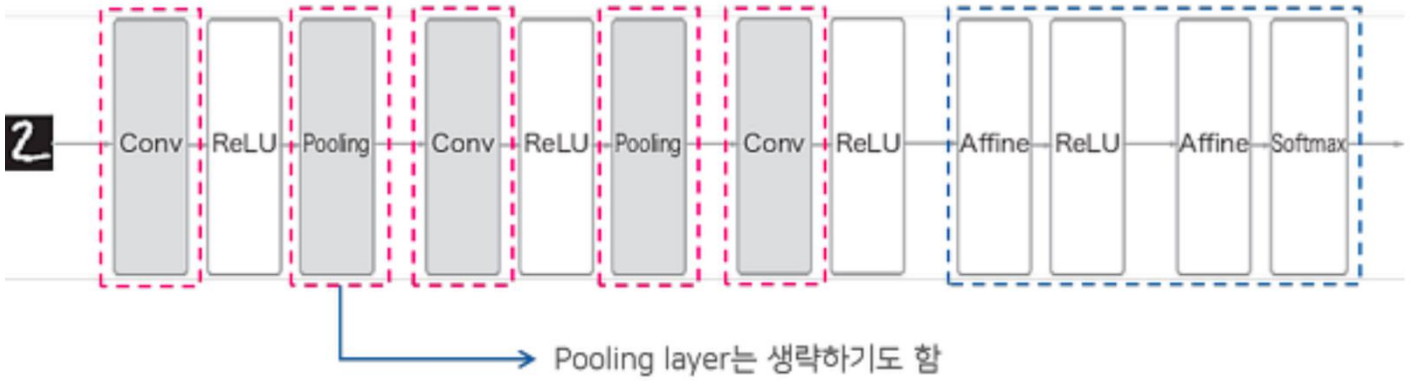
- CNN이란?

CNN의 구조

DNN



CNN



DNN과 CNN의 구조 비교

- CNN이란?

CNN의 구조

DNN

CNN

DNN

완전연결 신경망

Affine 계층 뒤에 활성화 함수 층이 이어지는 구조

DNN과 CNN의 구조 비교

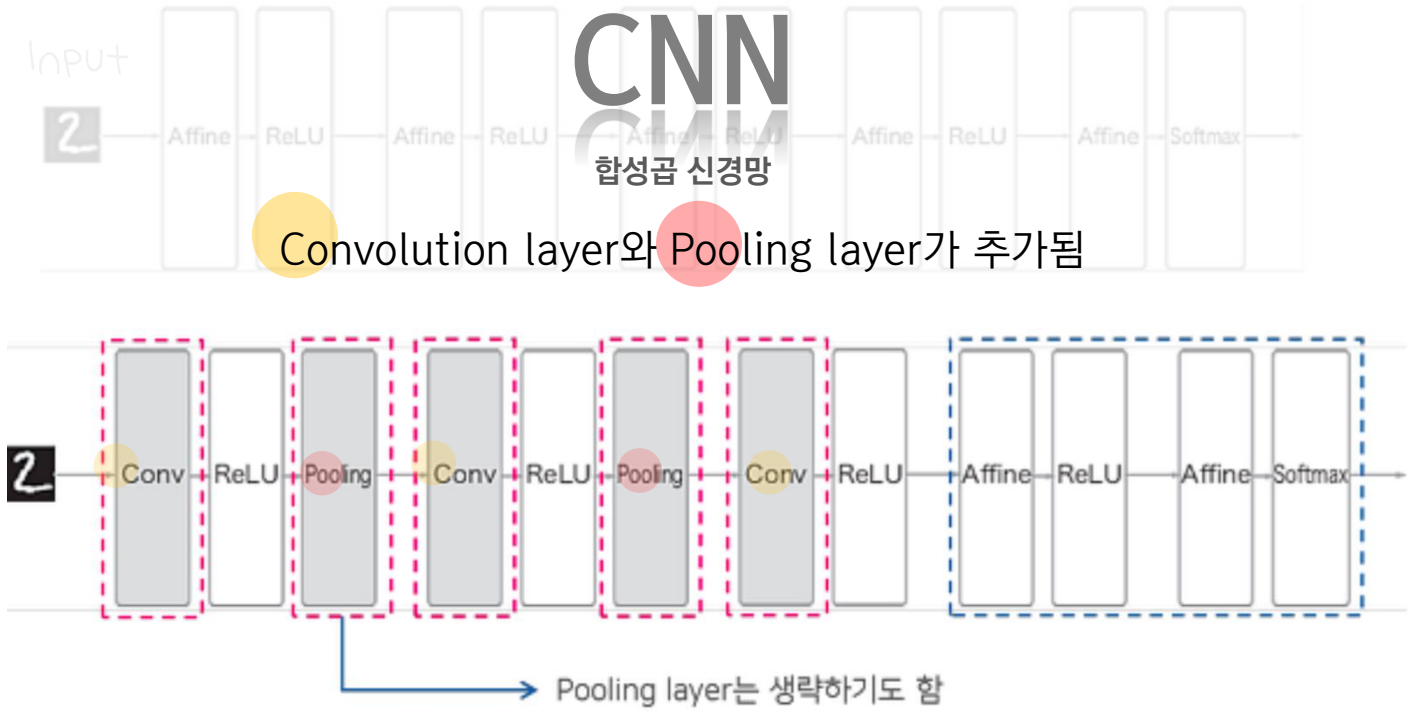
1 합성곱 신경망(CNN)

- CNN이란?

CNN의 구조

DNN

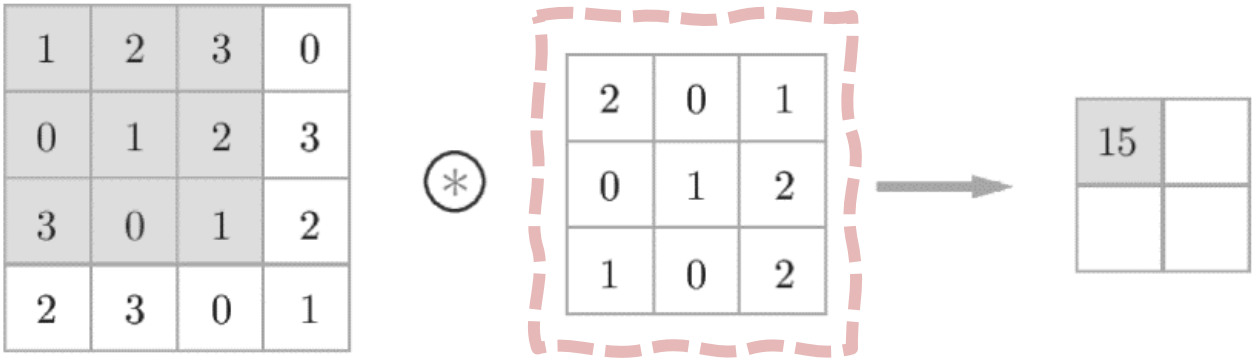
CNN



DNN과 CNN의 구조 비교

- Convolution Layer

(1) 필터



필터 : DNN의 **'가중치'**에 해당

필터의 윈도우를 일정 간격으로 이동하며 입력 데이터에 필터 적용

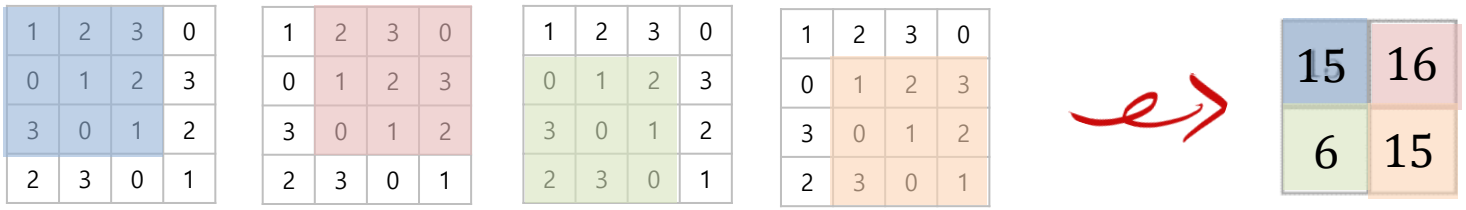
- Convolution Layer

(1) 필터

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

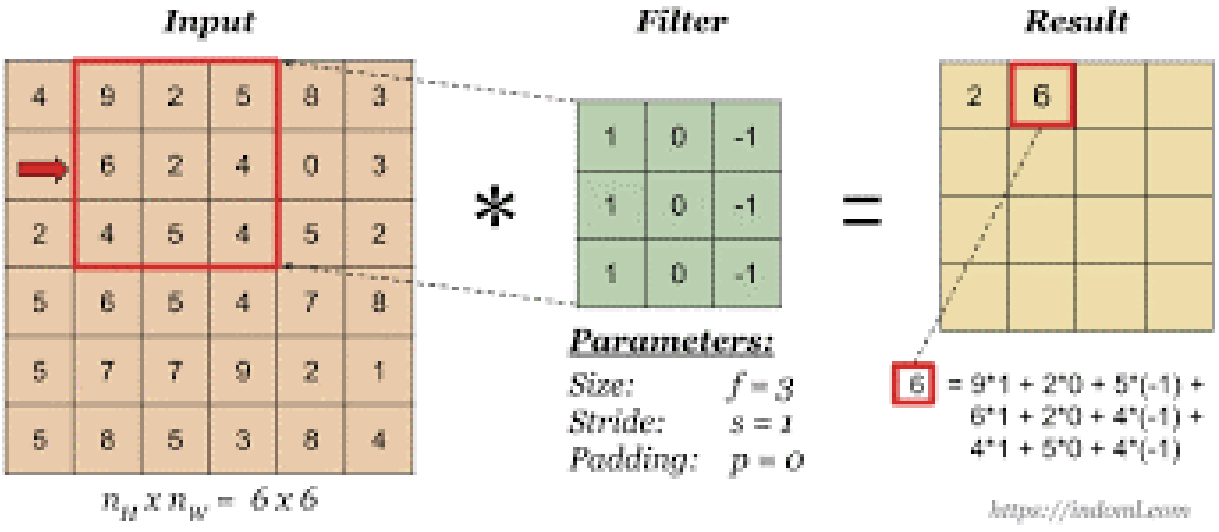
$\circledast$

2	0	1
0	1	2
1	0	2



- Convolution Layer

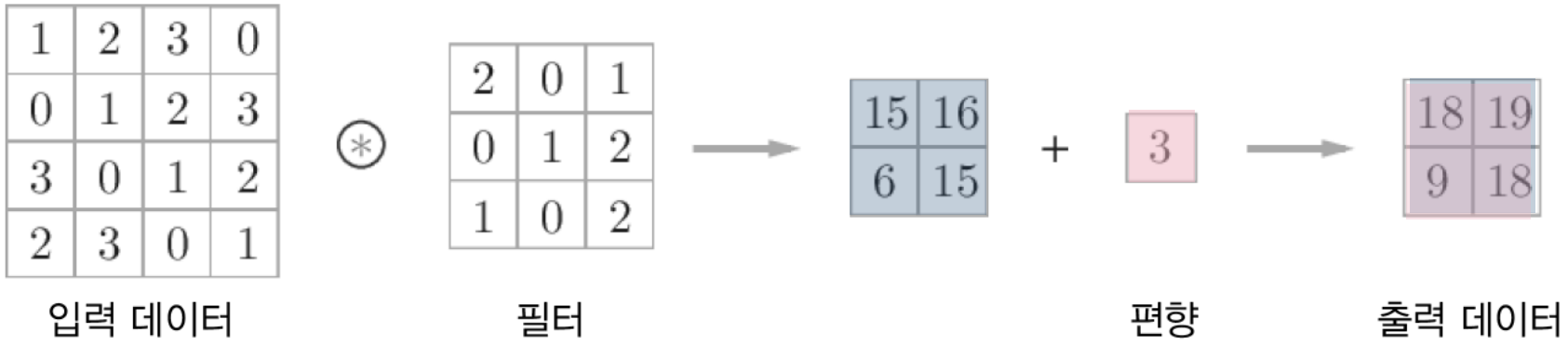
(2) 합성곱 연산 (*)



대응하는 원소끼리 곱한 후 그 값들을 합하는 연산

- Convolution Layer

(3) 편향



편향은 필터를 적용한 후의 데이터에 더해짐

- Convolution Layer

(3) 편향

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

입력 데이터

$$Z^{[i]} = \sum_{k=0}^{n} (x_k^{[i-1]} w_k^{[i]}) + b^{[i]}$$

필터

편향

18	19
9	18

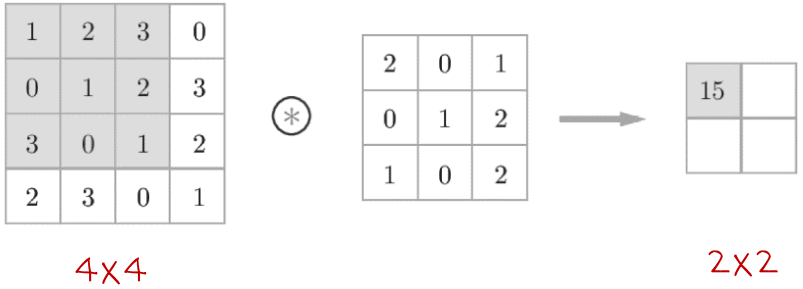
출력 데이터

DNN의 선형 연산과 유사한 구조로 필터 연산이 이루어진다!

편향은 필터를 적용한 후의 데이터에 더해짐

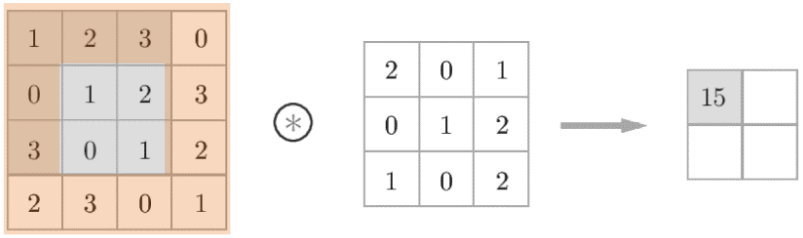
- Convolution Layer

(4) 패딩



패딩이 없으면?

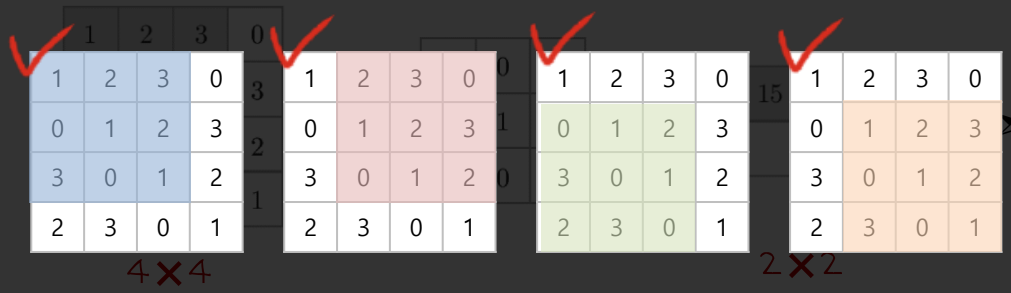
➢ 필터 적용 후 입력 데이터의 크기에 비해 출력 데이터의 크기가 작아짐



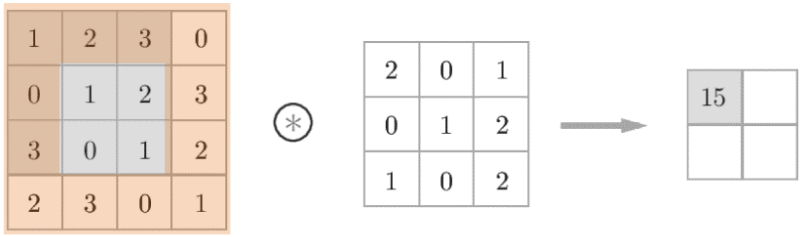
➢모서리에 위치한 입력 데이터의 정보는 적게 반영됨

- Convolution Layer

(4) 패딩



➢ 필터 필터를 총 4번 거치는 동안 모서리에 위치한 데이터는 한 번만 포함된다



➢모서리에 위치한 입력 데이터의 정보는 적게 반영됨

- Convolution Layer

(4) 패딩

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

0	0	0	0	0	0
0	1	2	3	0	0
0	0	1	2	3	0
0	3	0	1	2	0
0	2	3	0	1	0
0	0	0	0	0	0

합성곱 연산 수행 전 입력 데이터 주변을 특정 값으로 채움

제로 패딩 ➢ 패딩 값을 0으로 채워 넣음

- Convolution Layer

(4) 패딩

0	0	0	0	0	0
0	1	2	3	0	0
0	0	1	2	3	0
0	3	0	1	2	0
0	2	3	0	1	0
0	0	0	0	0	0

raw size : 4x4
after padding : 6 × 6

*

2	0	1
0	1	2
1	0	2

→

7	12	10	2
4	15	16	10
10	6	15	6
8	10	4	3

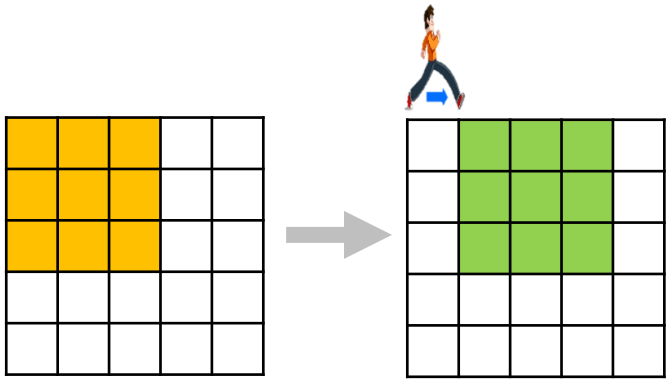
Output size : 4x4

모서리 정보가 적게 반영되는 현상 완화

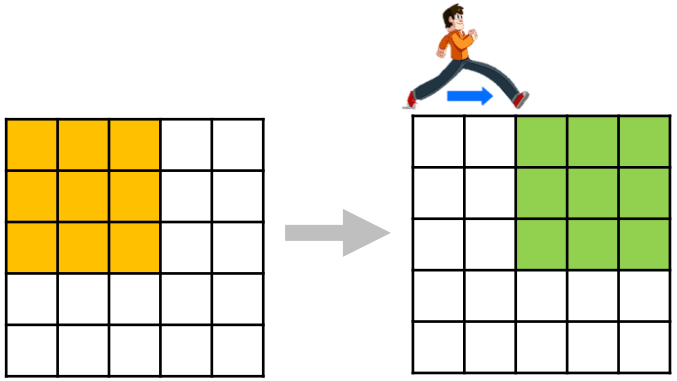
output featuremap의 **크기 유지**

- Convolution Layer

(5) 스트라이드



스트라이드 = 1



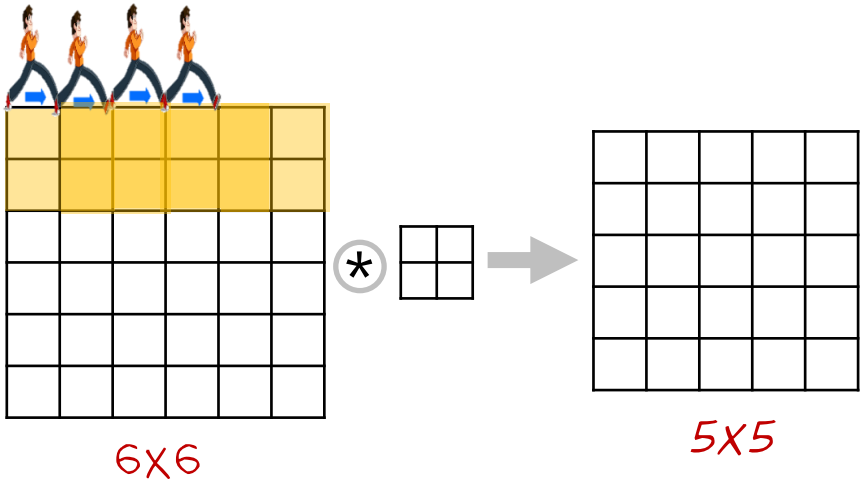
스트라이드 = 2

stride = 보폭

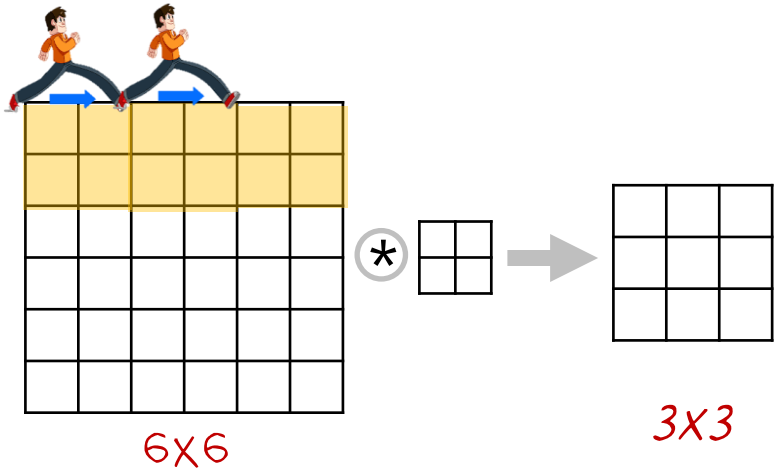
필터가 한번에 이동하는 **간격**

- Convolution Layer

(5) 스트라이드



스트라이드 = 1



스트라이드 = 2

스트라이드가 커지면 출력 크기는 작아짐

- 출력 크기의 결정

출력 크기

〈Output Height〉

$$OH = \frac{H+2P-FH}{S}+1$$

〈Output Width〉

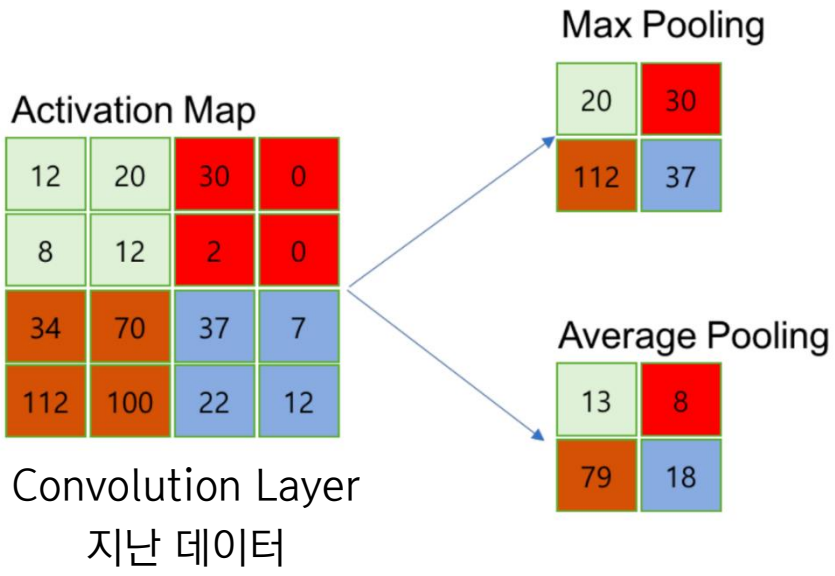
$$OW = \frac{W+2P-FW}{S}+1$$

입력 크기(H, W), **필터 크기**(FH, FW),

출력 크기(OH, OW), **패딩**(P), **스트라이드**(S)

- Pooling Layer

풀링 Pooling

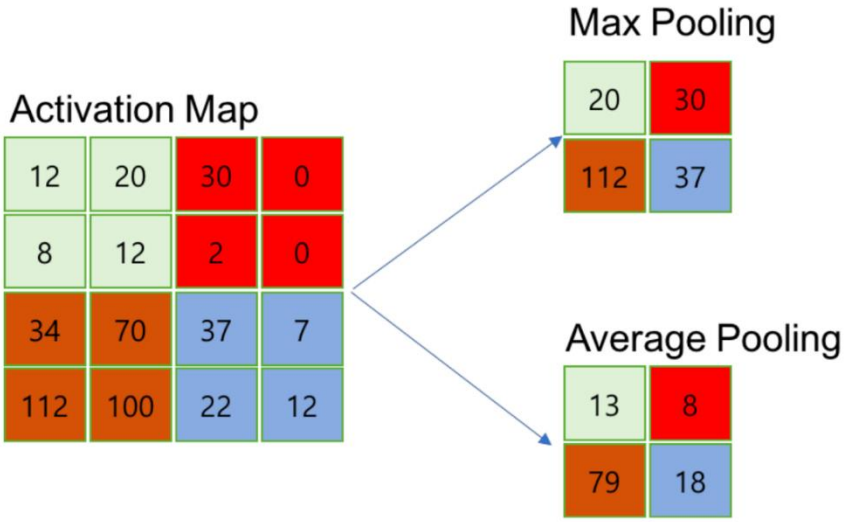


피쳐맵의 **크기**를 줄이는 연산

이미지의 크기를 줄여 특징을 잘 표현하는 값을 뽑아냄

- Pooling Layer

풀링 Pooling



최대 풀링(Max Pooling)

: 해당 영역의 최댓값을 구하는 연산

평균 풀링(Average Pooling)

: 해당 영역의 평균을 구하는 연산

- Pooling Layer

풀링 Pooling

〈 풀링의 특징 〉

1. 학습을 통해 개선해야 할 매개변수(파라미터)가 없다.
2. 인풋 피쳐맵의 변화에 영향을 적게 받는다.
3. 채널 수가 변하지 않는다.

Pooling Layer

풀링 Pooling

〈 풀링의 특징 〉

1. 학습을 통해 개선해야 할 매개변수(파라미터)가 없다.

풀링은 대상 영역에서 **최댓값**이나 **평균**을 구하는 과정
∴ 학습 시간에 영향은 미치지 않으면서, 중요한 특징들을 골라내어 줌!

2. 입력 데이터의 변화에 영향을 적게 받는다.

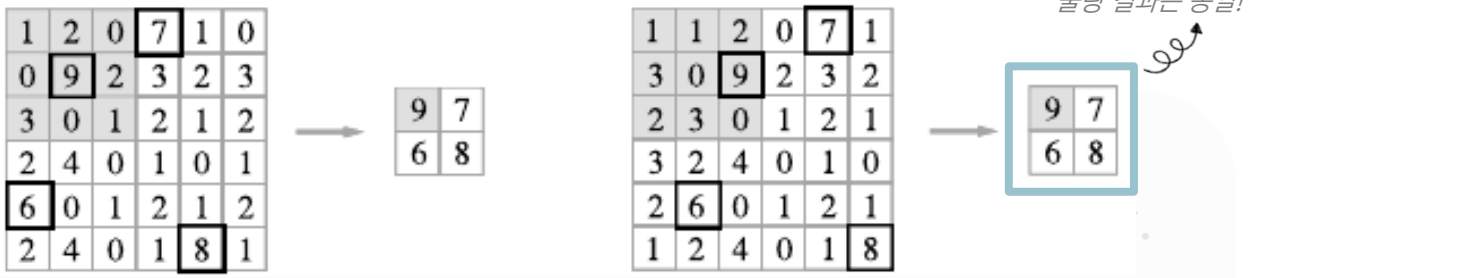
3. 채널 수가 변하지 않는다.

1 합성곱 신경망(CNN)

- Pooling Layer

풀링 Pooling

[Max Pooling]



2. 인풋 피쳐맵의 변화에 영향을 적게 받는다.

3. 입력 데이터의 차이를 풀링이 **흡수**해 사라지게 함

- Pooling Layer

풀링 Pooling

〈 풀링의 특징 〉

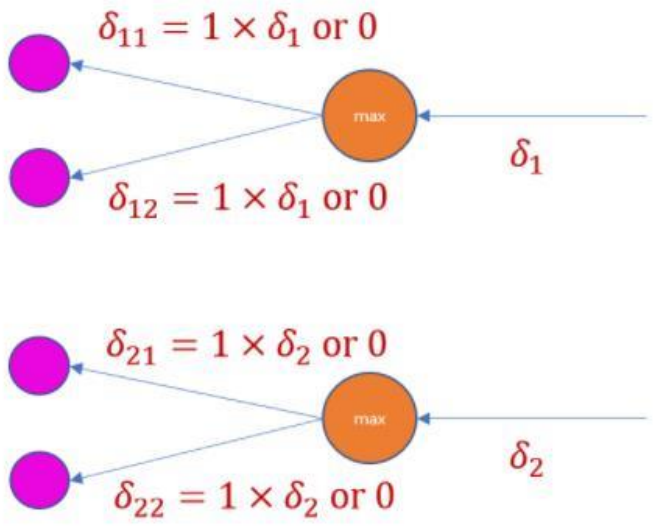
1. 학습을 통해 개선해야 할 매개변수(파라미터)가 없다.
2. 인풋 피쳐맵의 변화에 영향을 적게 받는다.
3. 채널 수가 변하지 않는다.

풀링은 **채널마다 독립적으로** 진행

➢ 입력 데이터 채널 수 = 출력 데이터 채널 수

- CNN 역전파

(1) Pooling Layer



최대 풀링(Max Pooling)

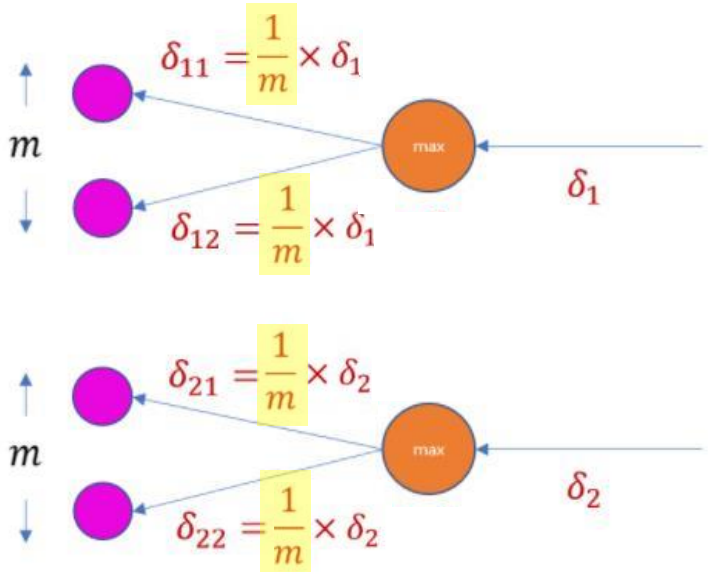
윈도우 중 가장 큰 값만 출력

➢ 역전파 역시 **최댓값을 가진 원소에만** 흘러감

➢ 이외의 변수에는 역전파 전달되지 않음

- CNN 역전파

(1) Pooling Layer



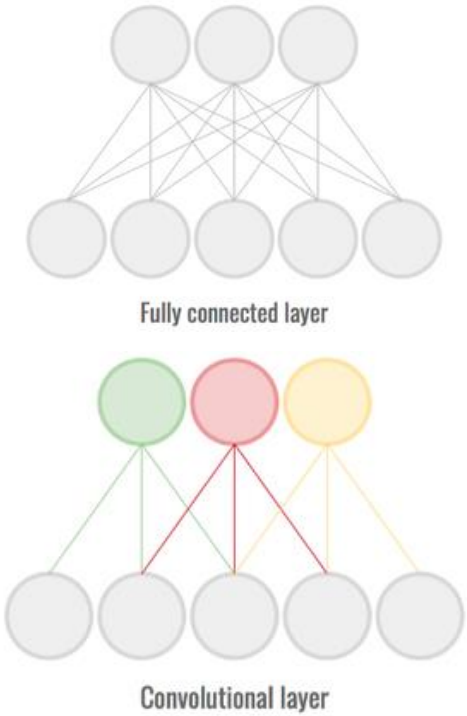
평균 풀링(Average Pooling)

원소들이 동일하게 반영됨

역전파를 **필터의 크기(m)로 나누어** 전달

- CNN 역전파

(2) Convolution Layer



Fully Connected Layer와는 달리

부분적으로 연결되어 있음

역전파 시 **연결된 노드로부터만**

값을 전달받음

2

RNN

- 자연어의 특징

순차적 데이터 Sequential Data

:단어나 표현의 등장 순서, 단어 간의 관계 중요

몰디브 가서 모히또 한 잔

모히또 가서 몰디브 한 잔

- 자연어의 특징

DNN의 한계

:단어나 표현의 등장 순서?

	Petal length	Petal Width	Sepal Length	Sepal Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

열 순서가 예측에 영향을 주지 않는다

DNN은 순서 정보를 무시한다

- 자연어의 특징

순차적 데이터

일주일전에는 착한맛 로제 떡볶이를 먹었고, 어제는 오리지널 로제 떡볶이를 먹었다면..?
첫번째에 위치한 단어가 사라지면 의미가 달라진다!

어제 주문했던 로제 떡볶이의 **맵기는?**

일주일 전 주문했던 로제 떡볶이의 **맵기는?**

→ 거리가 먼 단어 간의 관계 파악 중요

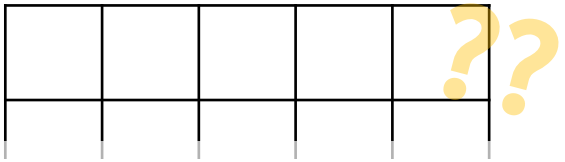
- 자연어의 특징

CNN의 한계

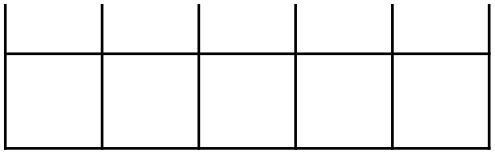
어제	내가	배떡	을	먹
을	까	엽떡	을	먹
을	까	고민	하다	그냥
엽떡	로제	떡볶이	먹	었
는데	무슨	맛	먹었	게?

*

〈 FILTER 〉



두 핵심적인 단어가 함께 포함되기 위해서는
필터 사이즈가 인풋 사이즈와
동일해야 함!

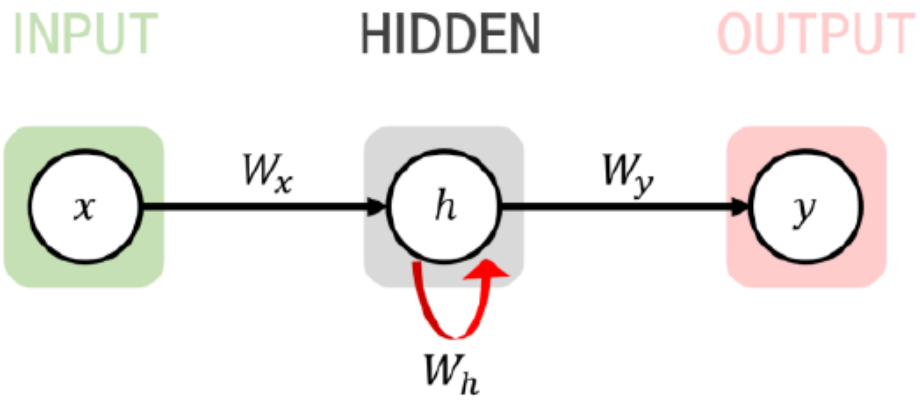


연산량 폭발로 일정 크기 이상 필터 확장 불가능

➔ **거리**가 먼 단어 간의 관계 파악 불가

- RNN이란?

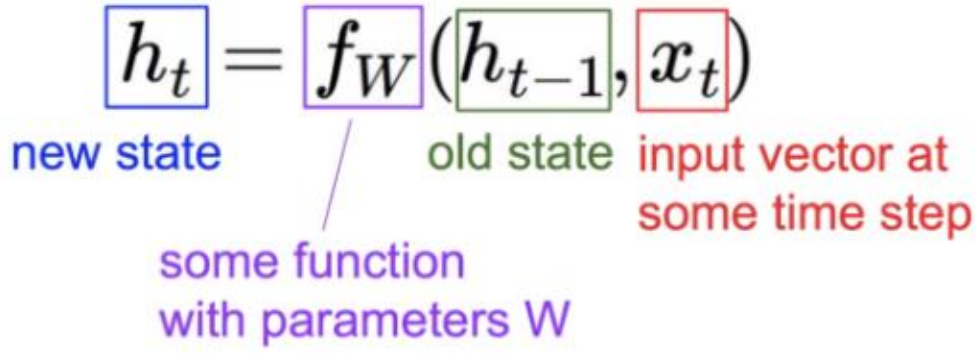
순환신경망 Recurrent Neural Network



전 시점의 은닉층 출력값이 **다시 입력 값으로** 작용하는 모델

- RNN이란?

순환신경망 Recurrent Neural Network

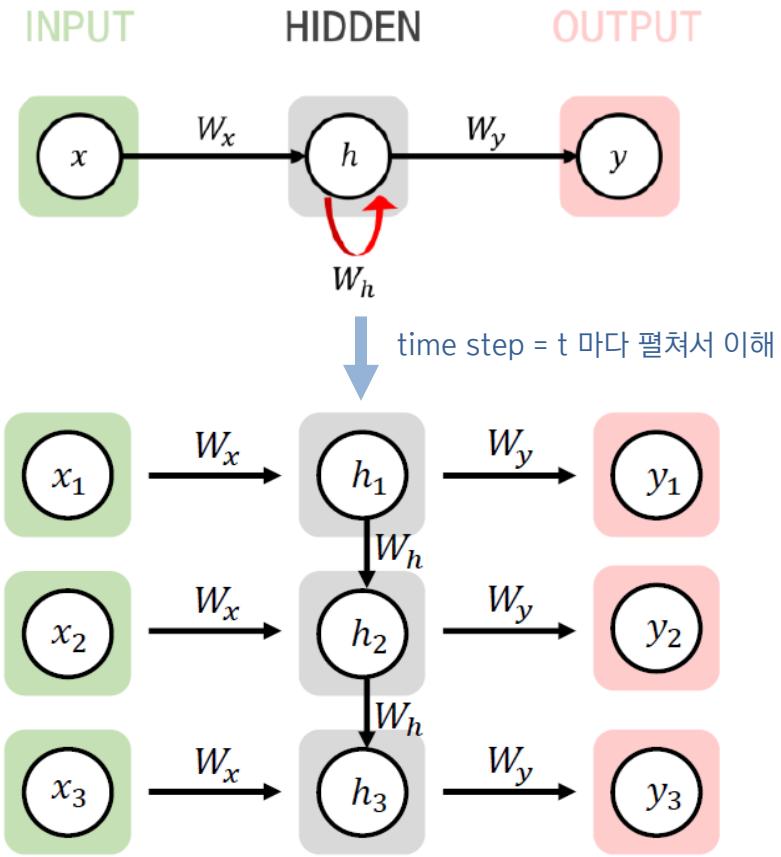


RNN의 출력은 **이전의 모든 입력**에 영향을 받음

➔ 순차적 데이터인 자연어의 시간적 특징 반영 가능

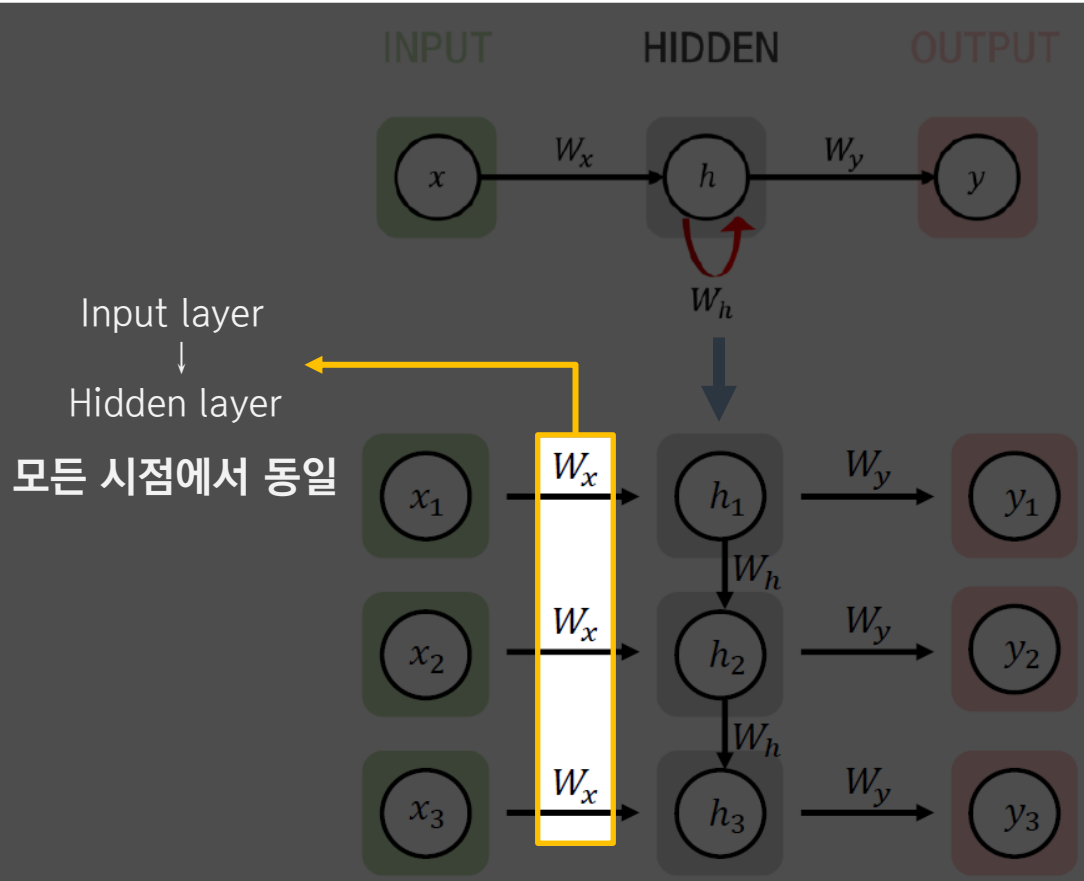
- RNN 모델 구조

RNN의 parameter



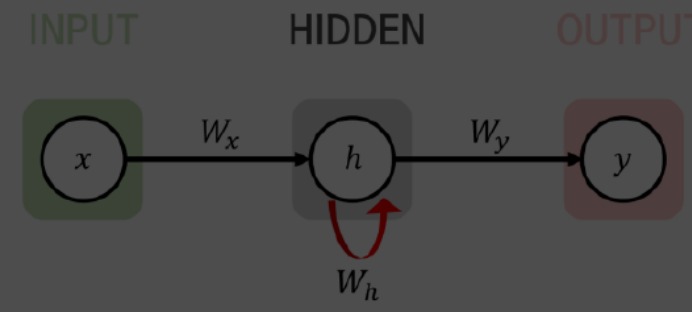
- RNN 모델 구조

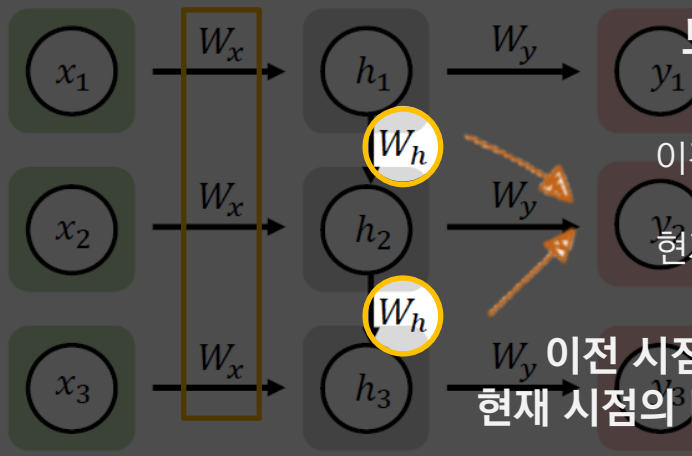
RNN의 parameter



- RNN 모델 구조

RNN의 parameter





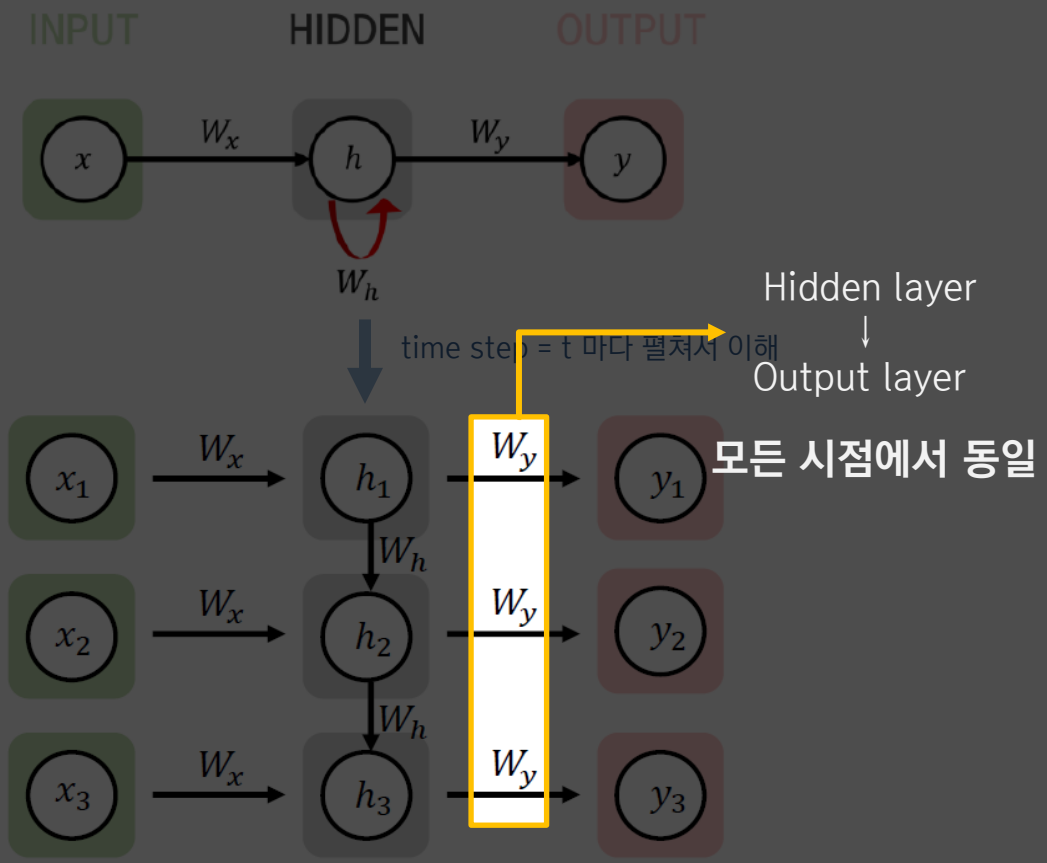
모든 시점에서 동일

이전 시점의 Hidden layer
↓
현재 시점의 Hidden layer

이전 시점까지의 hidden state값이
현재 시점의 hidden state 영향 미치는 정도

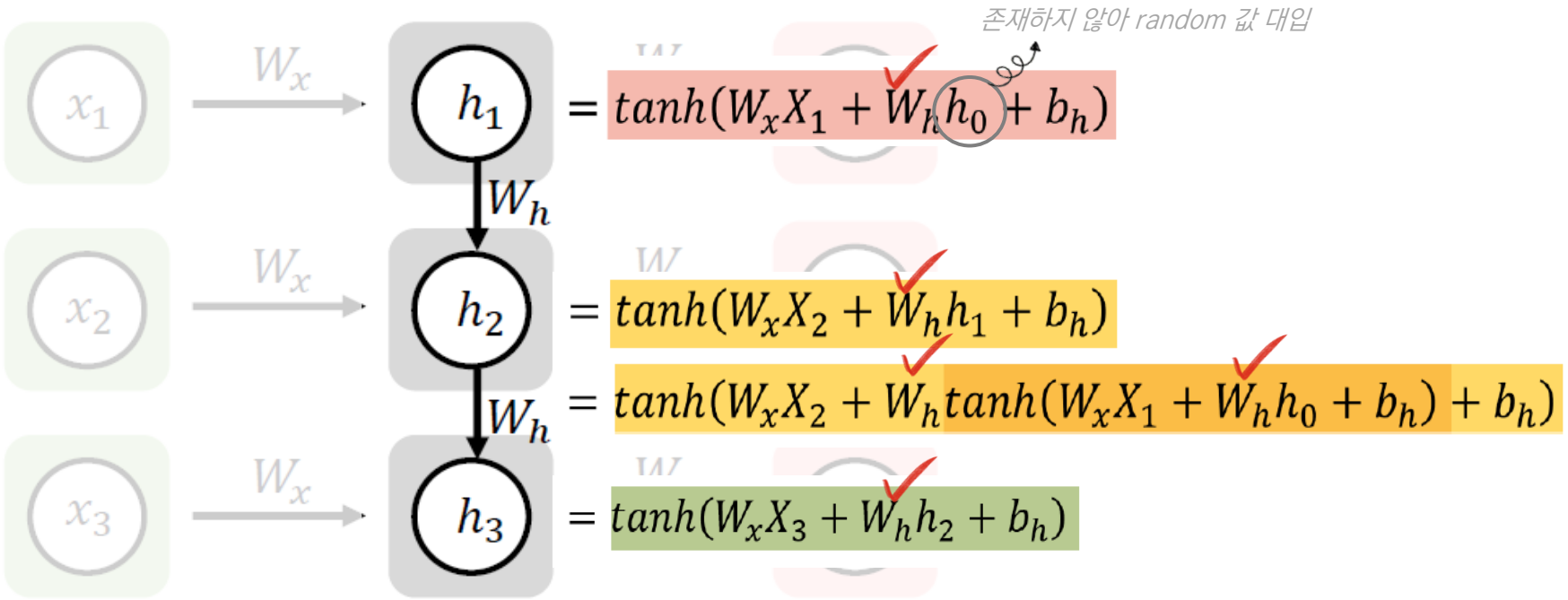
- RNN 모델 구조

RNN의 parameter



- RNN 모델 구조

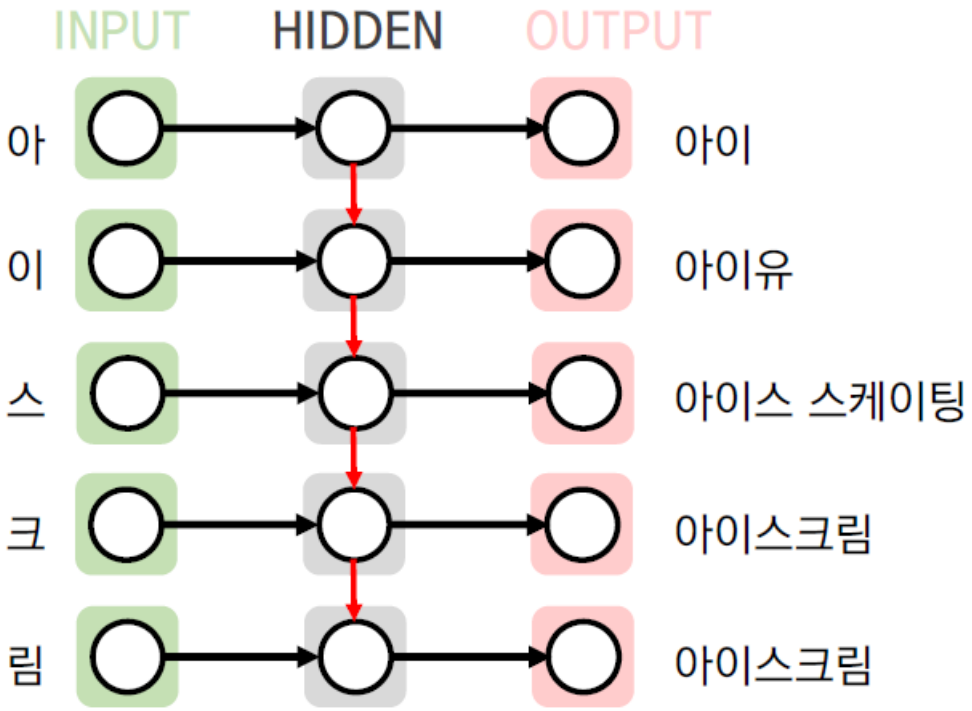
RNN의 구조



h_t를 구할 때 h_{t-1} 이 영향을 끼침으로써
이전 시점까지의 정보가 모두 영향을 미친다

- RNN 모델 구조

RNN의 구조



$$h_t = tanh(W_x x_t + W_h h_{t-1} + b)$$

$$y_t = f(W_v h_t + b)$$

각 시점 별 출력값
시점마다 다르게 출력

- RNN 모델 구조

RNN의 한계점

장기 의존성 문제

어제 어제 어제 어제 어제 어제 어제

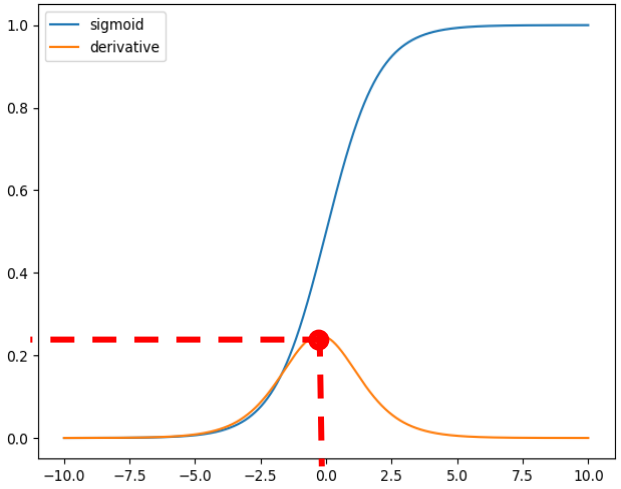
어제 주문했던 로제 떡볶이의 **맵기는?**

은닉층 값 **연쇄적**으로 연결
앞쪽의 타입 스텝 영향력↓(잊혀짐)

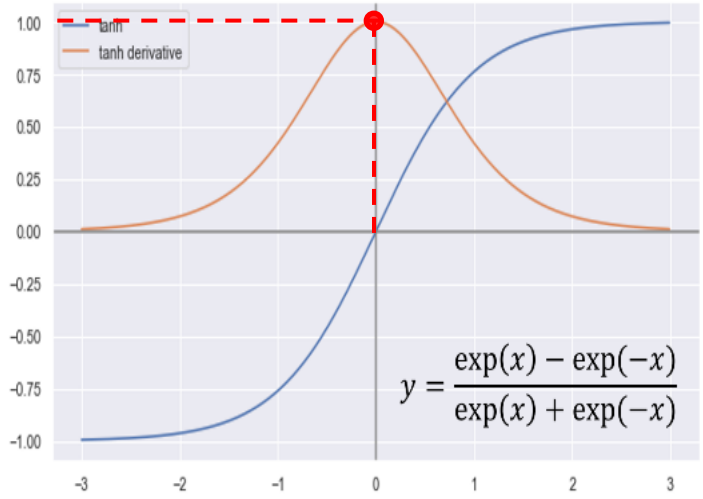
- RNN 모델 구조

활성화 함수 tanh

sigmoid



tanh



Why?

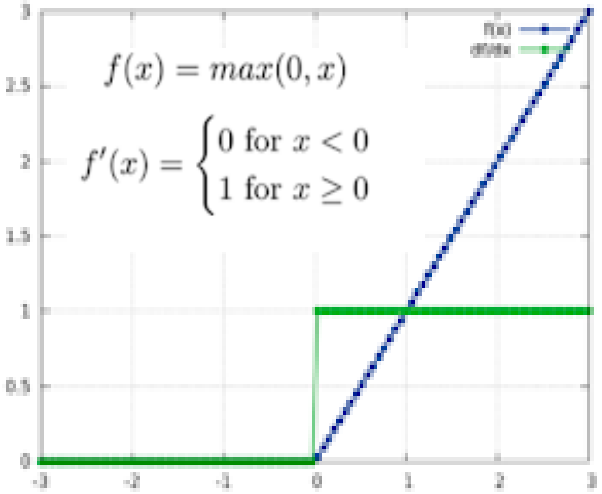
미분 최대 값이 1이라

Vanishing gradient 문제로부터 상대적으로 자유로움

- RNN 모델 구조

활성화 함수 tanh

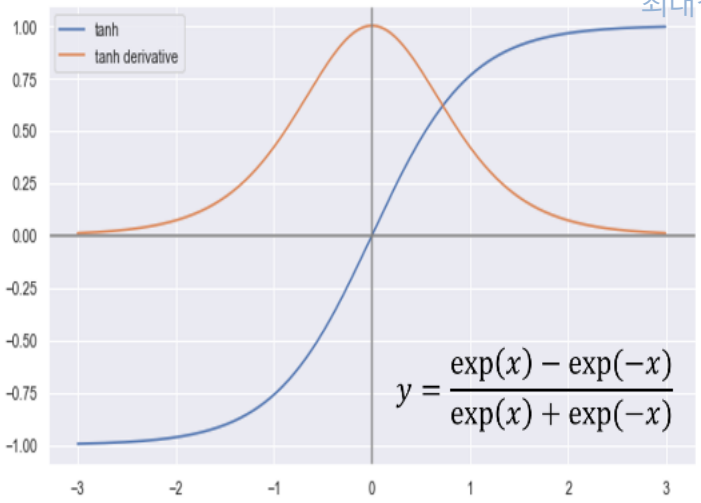
Relu



≪

tanh

최대값이 1



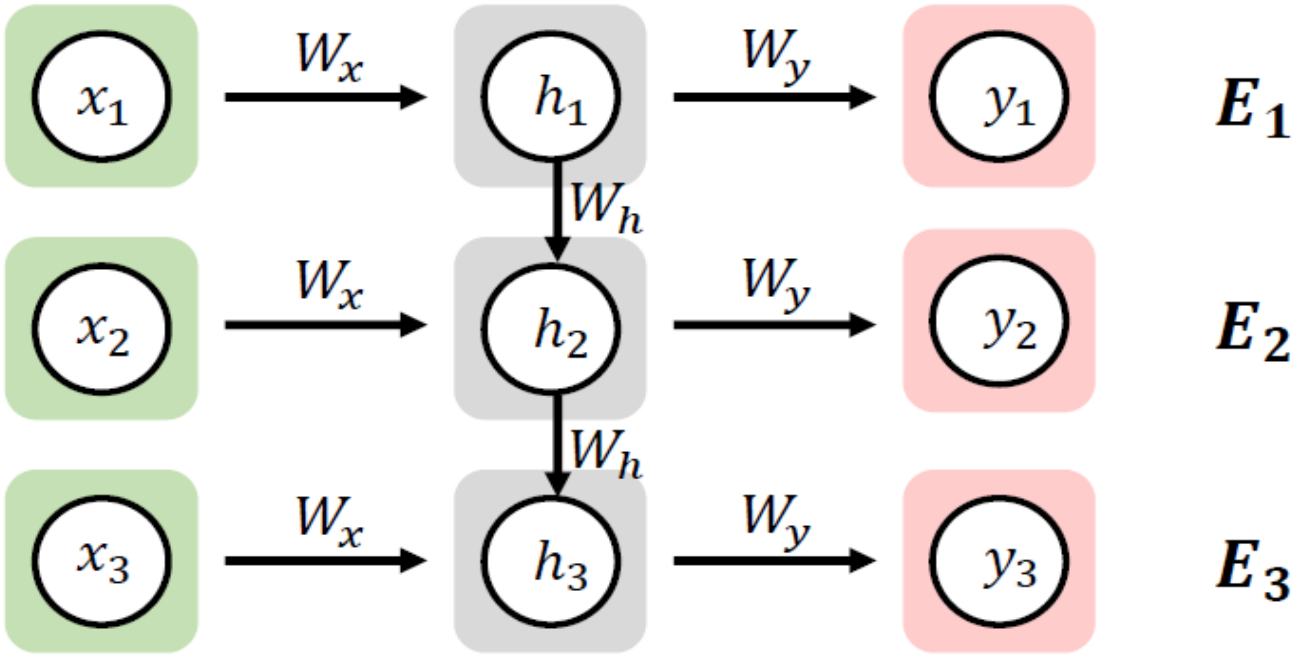
Why?

RNN : 같은 레이어를 여러 번 반복, 1보다 큰 값 입력 시

Exploding gradient 야기

- RNN의 역전파

BPTT Back Propagation Through Time

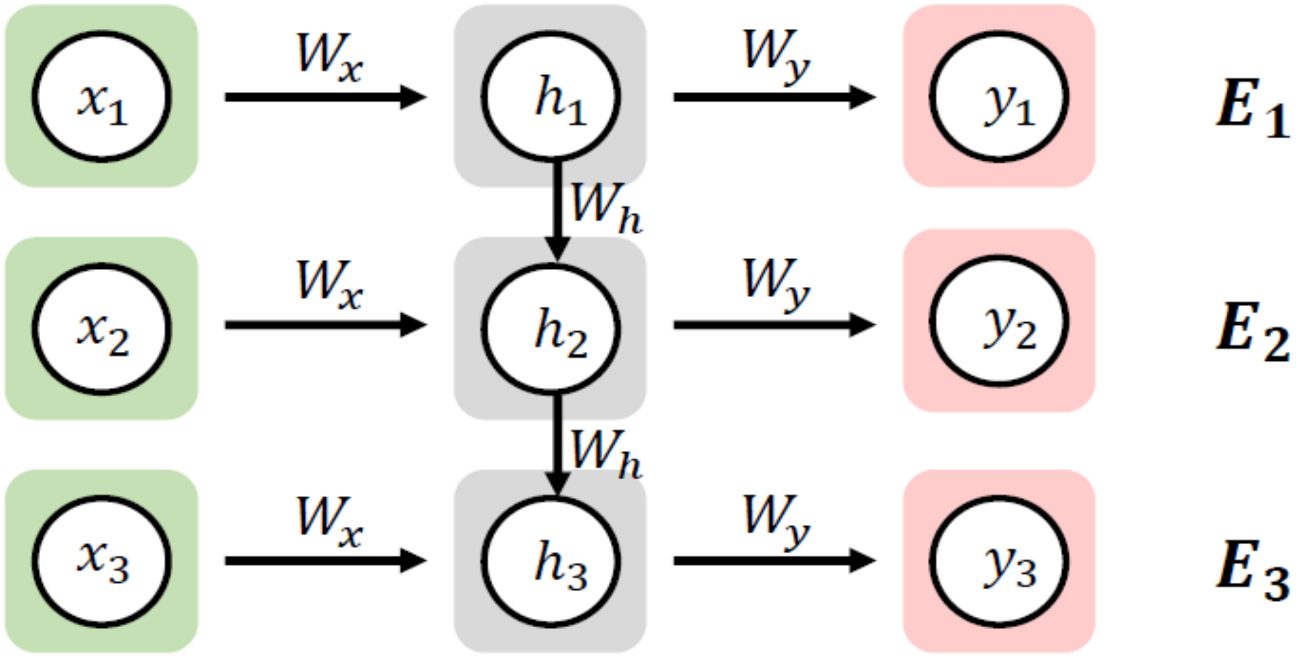


매 시점마다 손실함수가 계산되어

한 번의 순전파에서 여러 개의 **손실함수** 값이 나옴

- RNN의 역전파

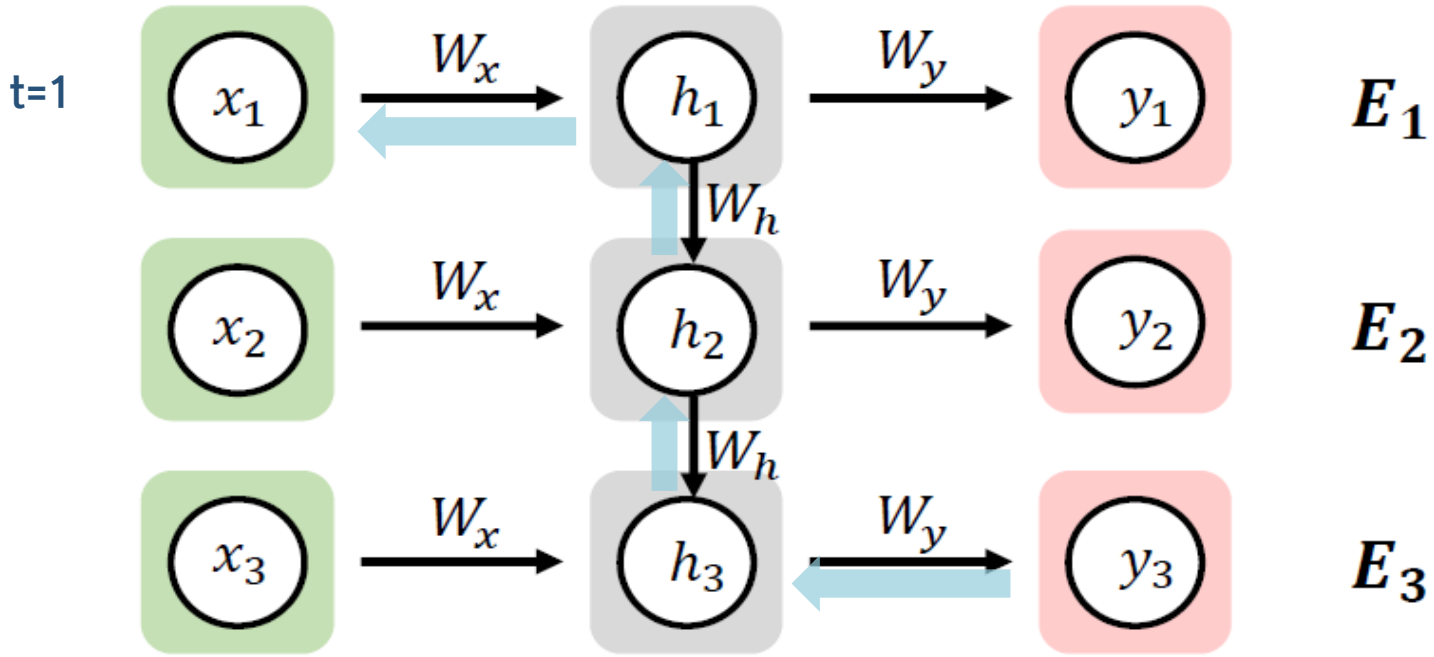
BPTT Back Propagation Through Time



매 시점마다 W_x에 대해 역전파가 이루어짐

- RNN의 역전파

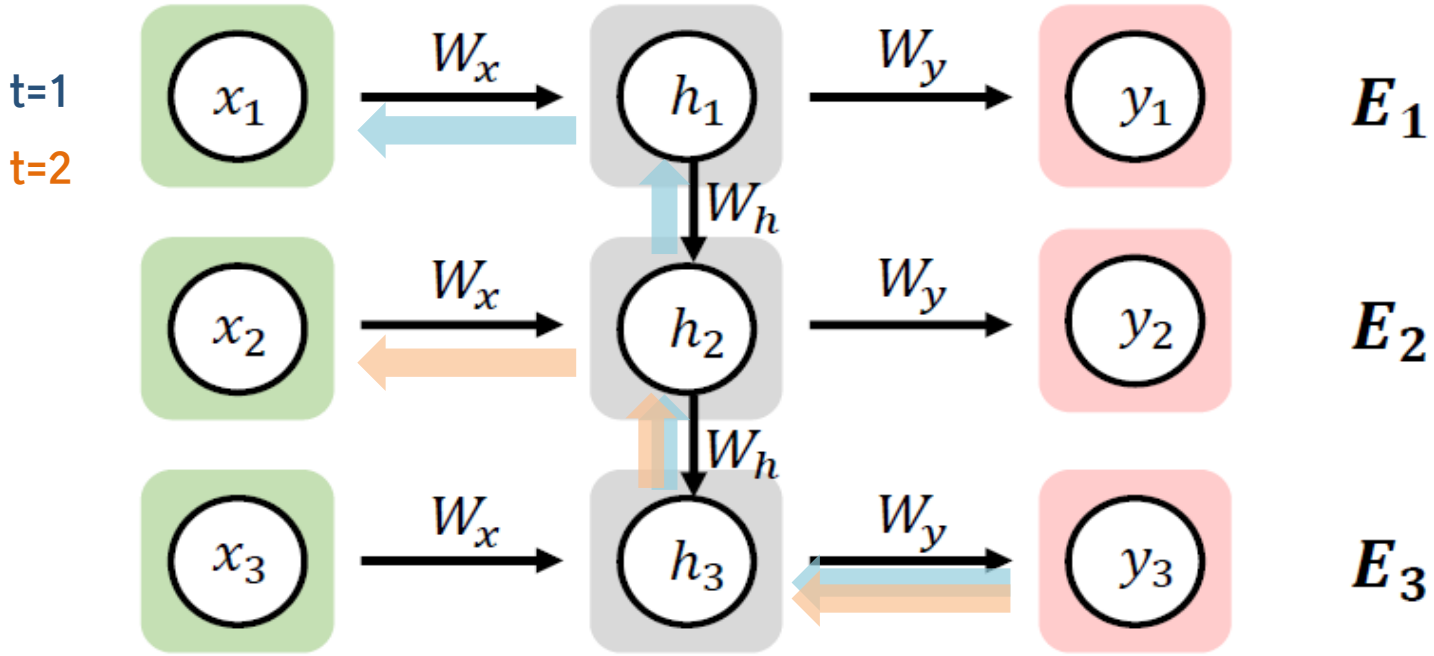
BPTT Back Propagation Through Time



매 시점마다 W_x에 대해 역전파가 이루어짐

- RNN의 역전파

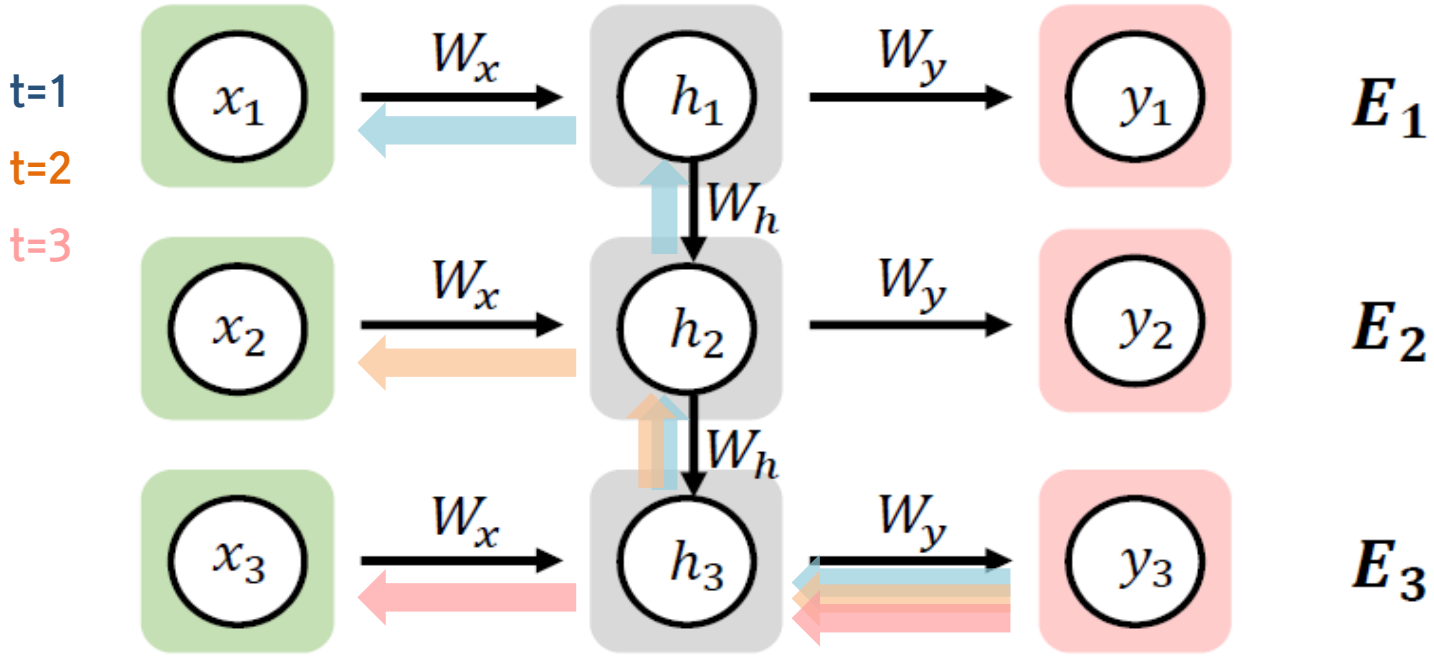
BPTT Back Propagation Through Time



매 시점마다 W_x에 대해 역전파가 이루어짐

- RNN의 역전파

BPTT Back Propagation Through Time

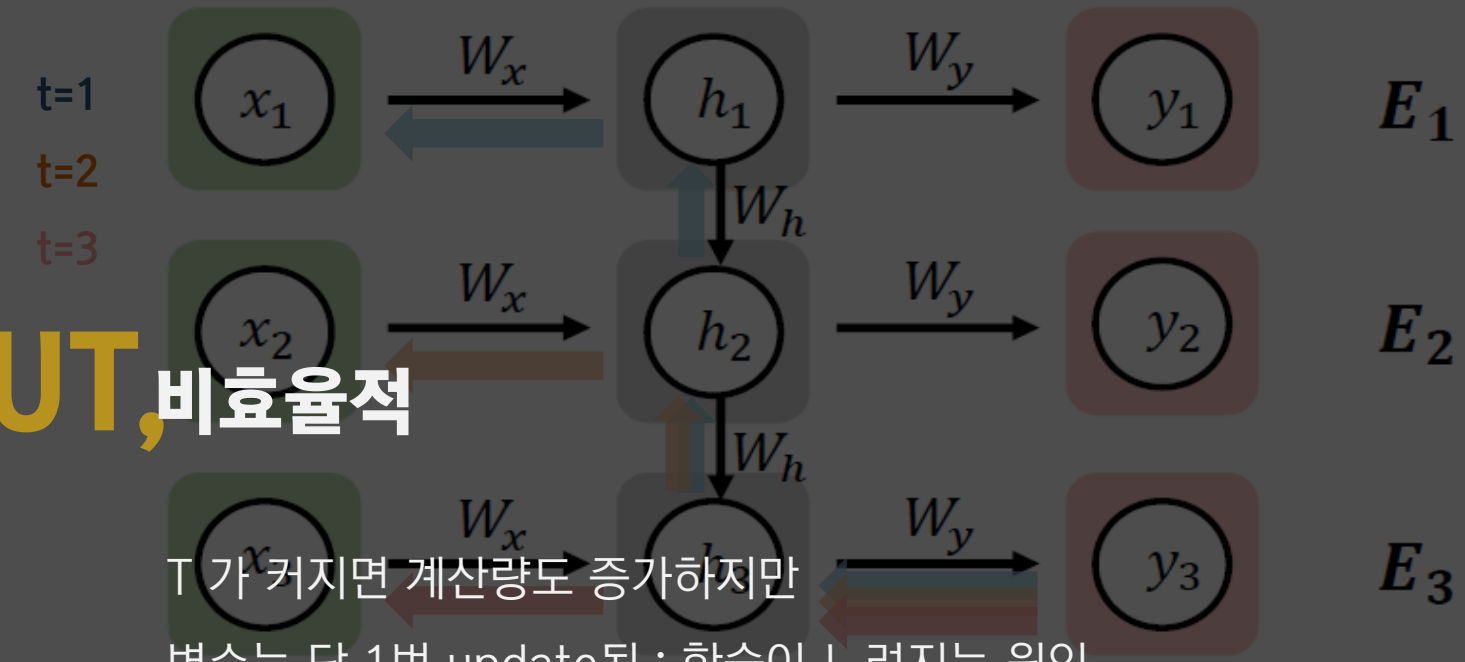


매 시점마다 W_x에 대해 역전파가 이루어짐

모든 시점의 역전파 값을 더하여 W_x 에 대한 역전파 값으로 사용

- RNN의 역전파

BPTT Back Propagation Through Time



BUT, 비효율적

T 가 커지면 계산량도 증가하지만

변수는 단 1번 update됨 : 학습이 느려지는 원인

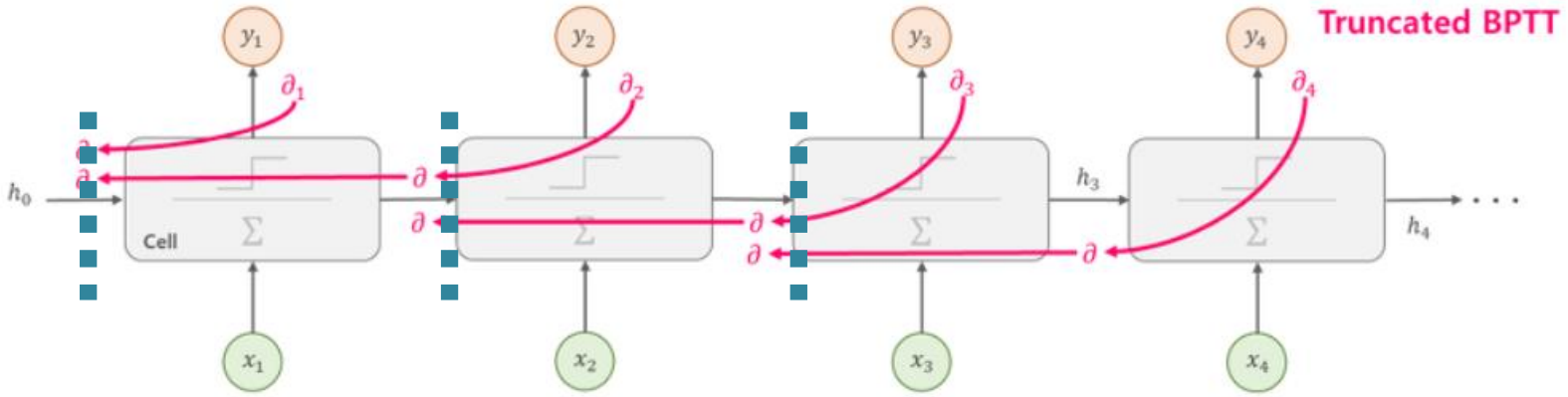
매 시점마다 W_x에 대해 역전파가 이루어짐

모든 시점의 역전파 값을 더하여 W_x 에 대한 역전파 값으로 사용

Truncated BPTT

- RNN의 역전파

Truncated BPTT



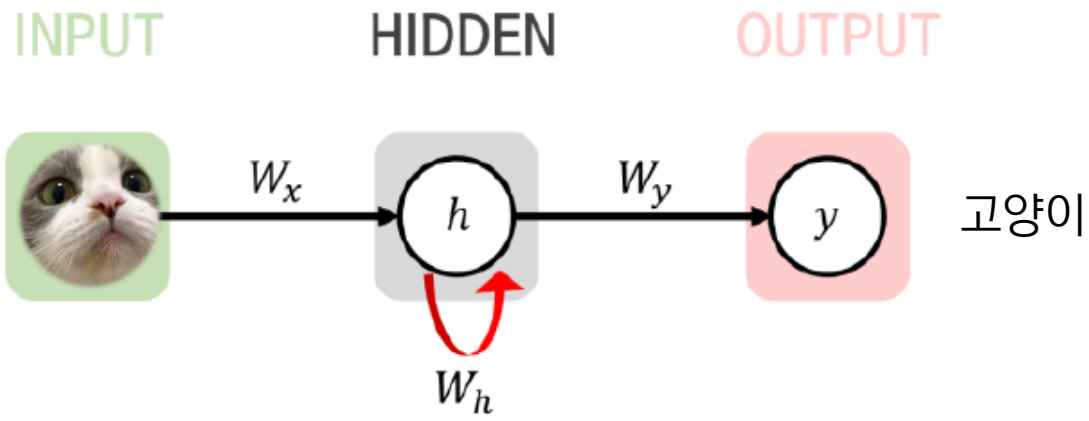
✔순전파는 끊지 않고 진행

전체 타임 스텝을 일정 구간으로 나눠 역전파 진행

Exploding/Vanishing gradient 문제(BPTT) 해결

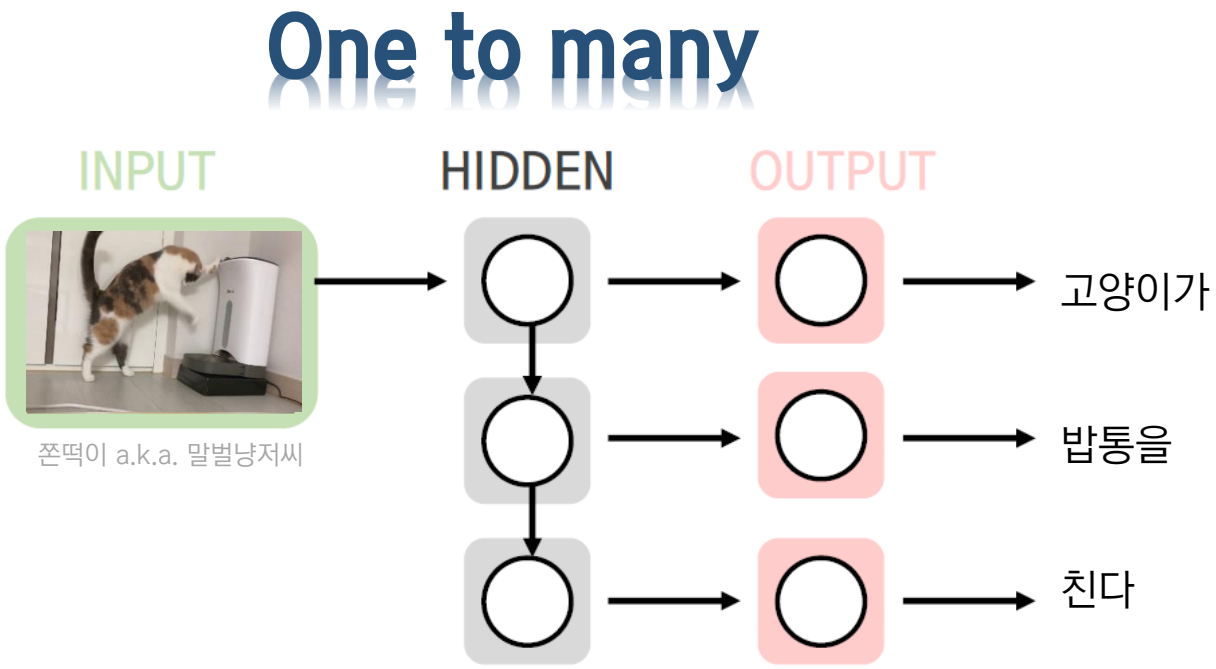
- RNN의 활용

One to one



Fixed-size input → Fixed-size output
ex. 이미지 분류

- RNN의 활용

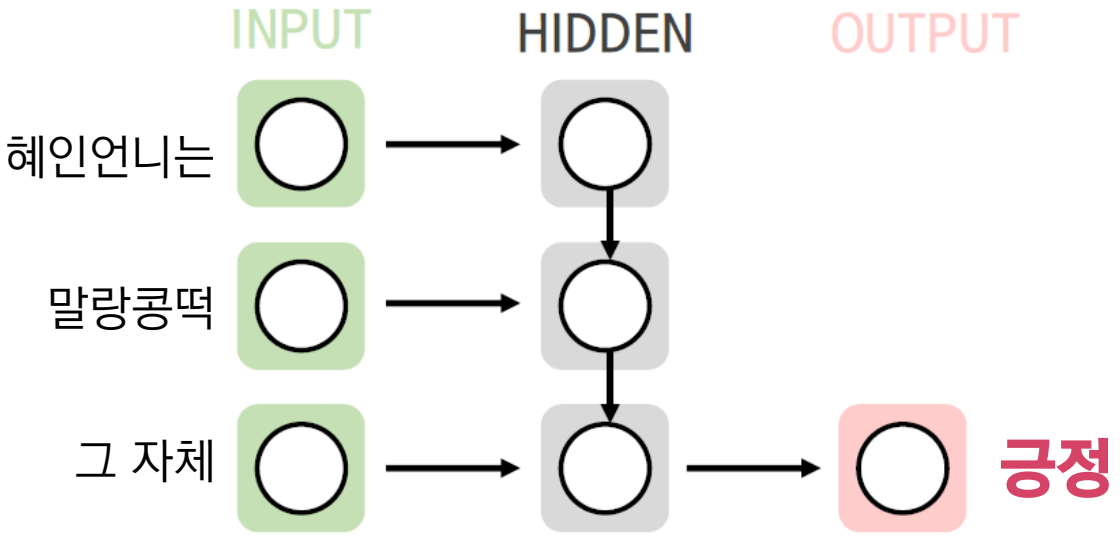


Fixed-size input → Sequence output

한 장의 이미지에 대해 여러 개의 단어로 묘사

- RNN의 활용

Many to one

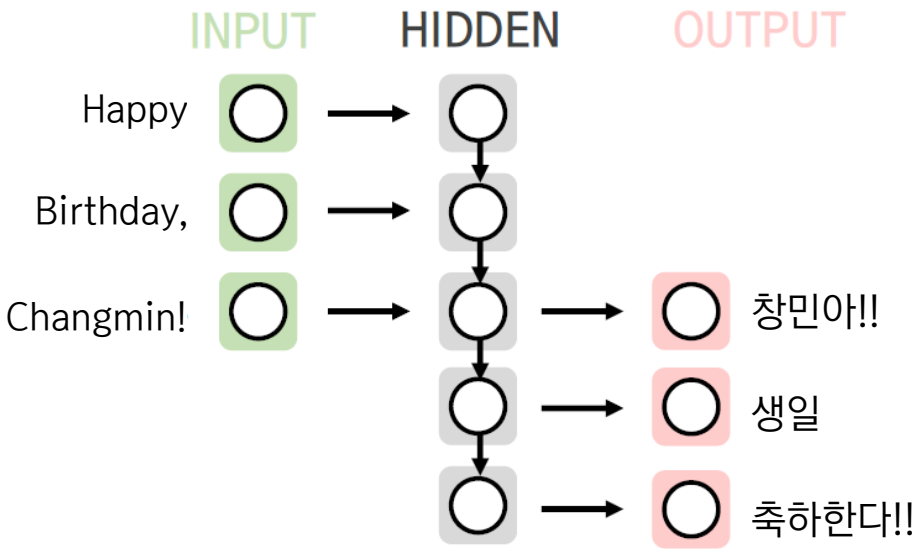


Sequence input → X Sequence output

- RNN의 활용

Many to many

ex) 번역



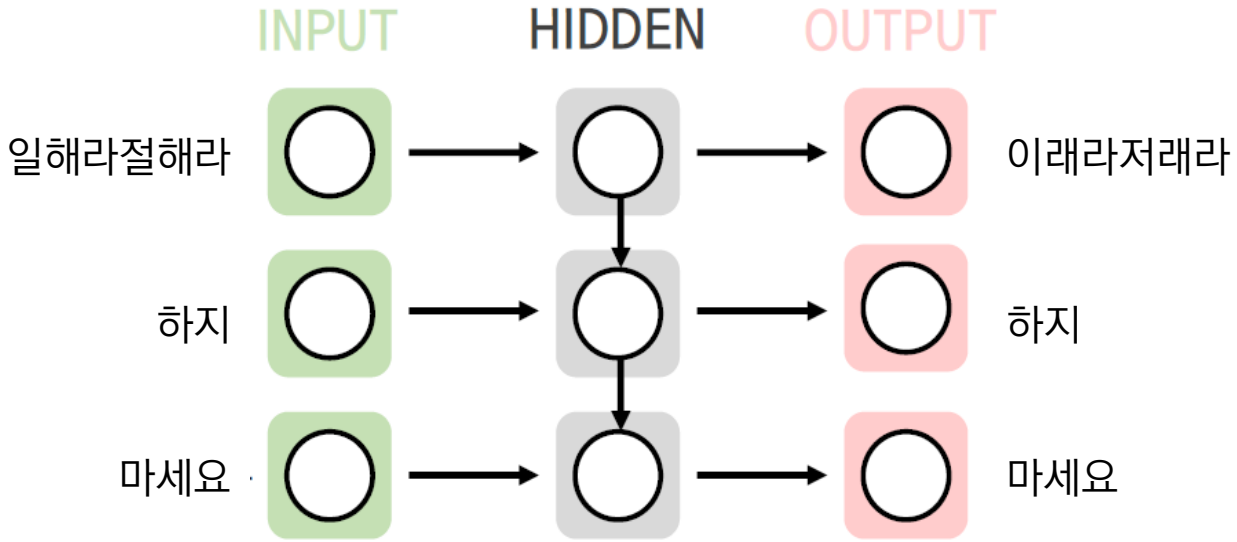
Sequence input → Sequence output

입력 값의 입력 시점과 출력값 출력 시점 **다름**

RNN의 활용

Many to many

ex) 오타수정



Sequence input → Sequence output

입력 값의 입력 시점과 출력값 출력 시점 **같음**

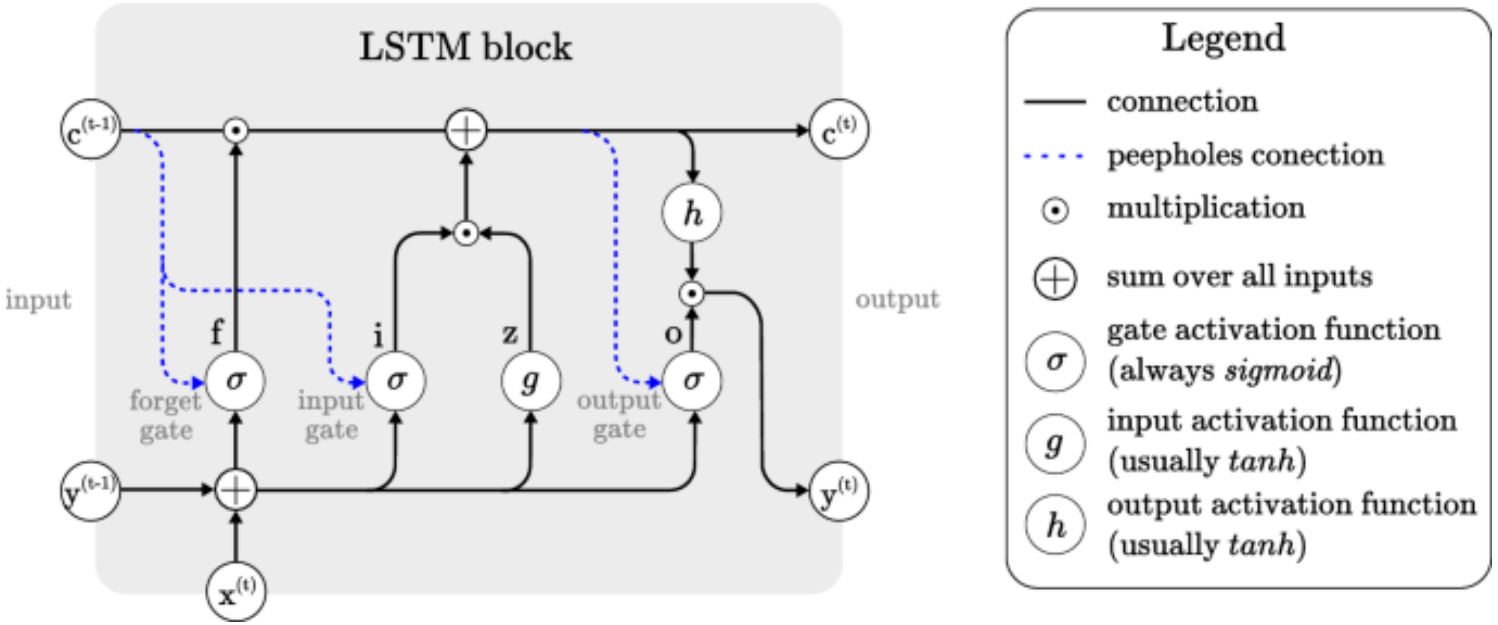
3

LSTM

3 LSTM

- LSTM이란?

LSTM



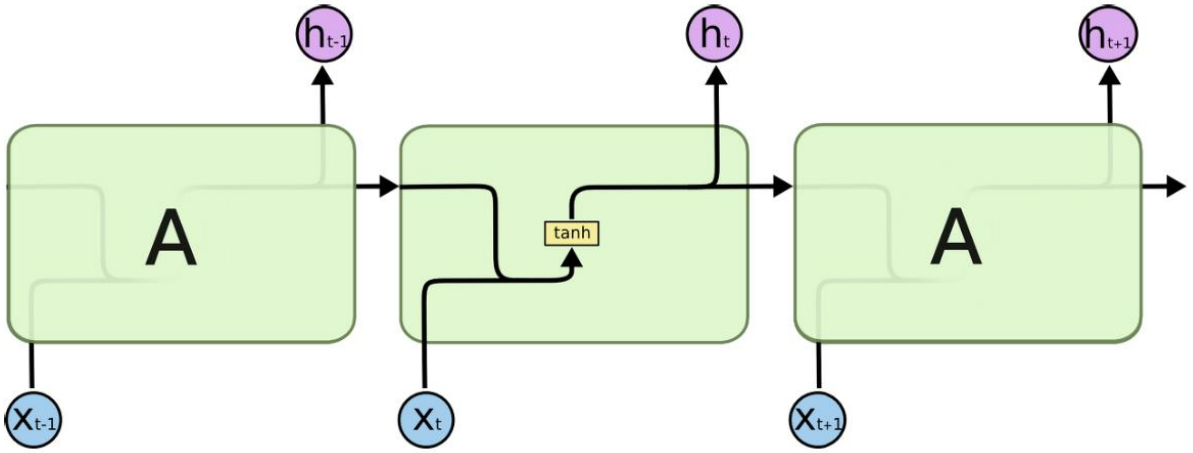
RNN의 **장기 의존성 문제**와 **기울기 소실/폭발 문제**를

해결하기 위한 모델

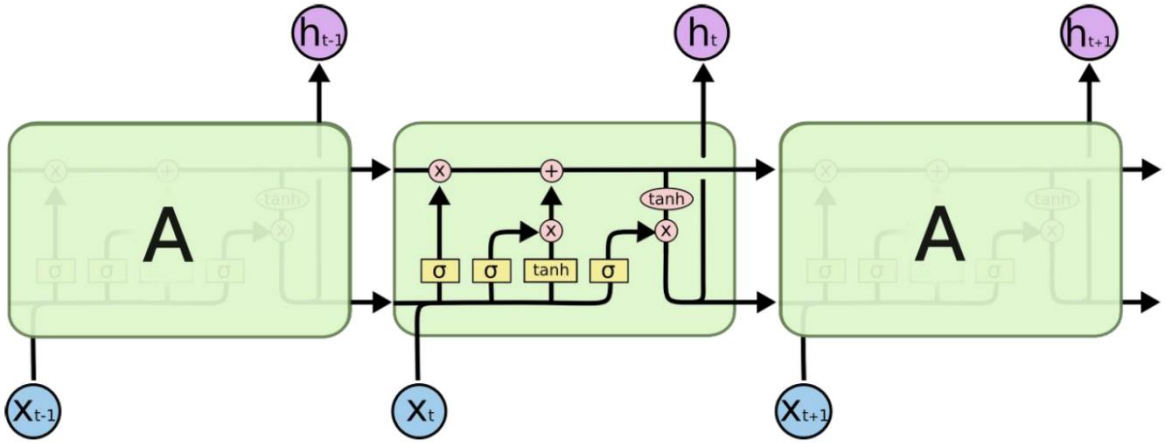
3 LSTM

- RNN과 LSTM의 비교

RNN



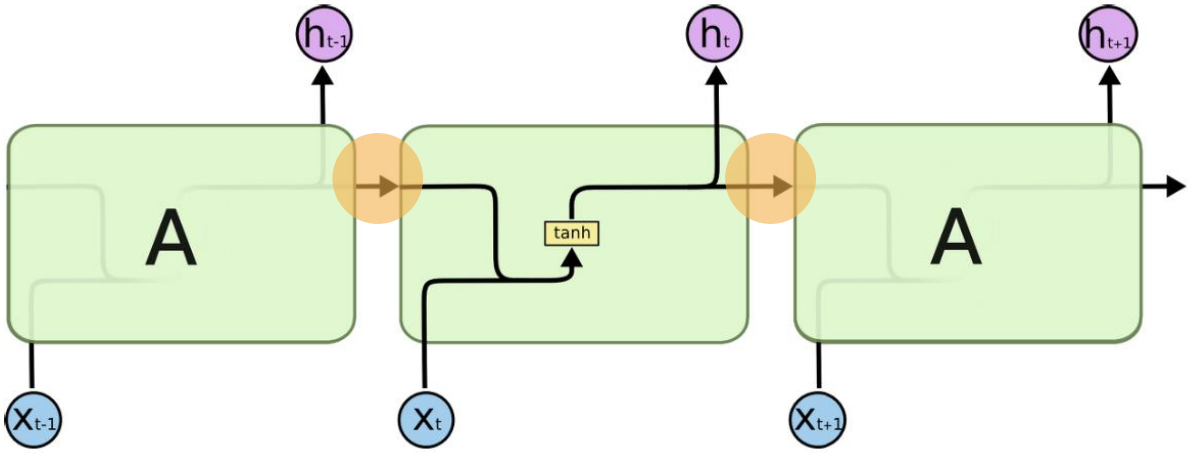
LSTM



3 LSTM

- RNN과 LSTM의 비교

RNN



LSTM

RNN은 다음 time step으로 가는 정보가 **하나**

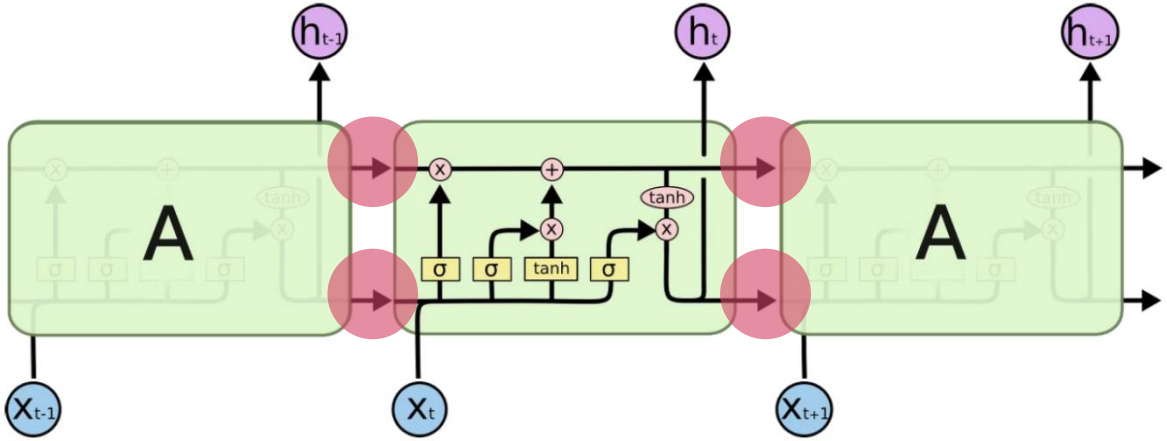
3 LSTM

- RNN과 LSTM의 비교

RNN

LSTM은 다음 time step으로 가는 정보가 **둘**

LSTM



3 LSTM

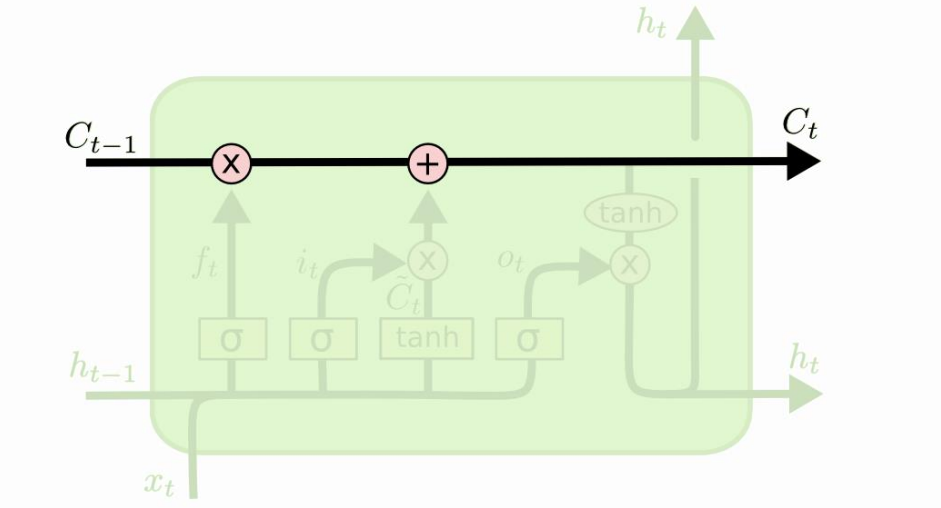
- RNN과 LSTM의 비교

RNN

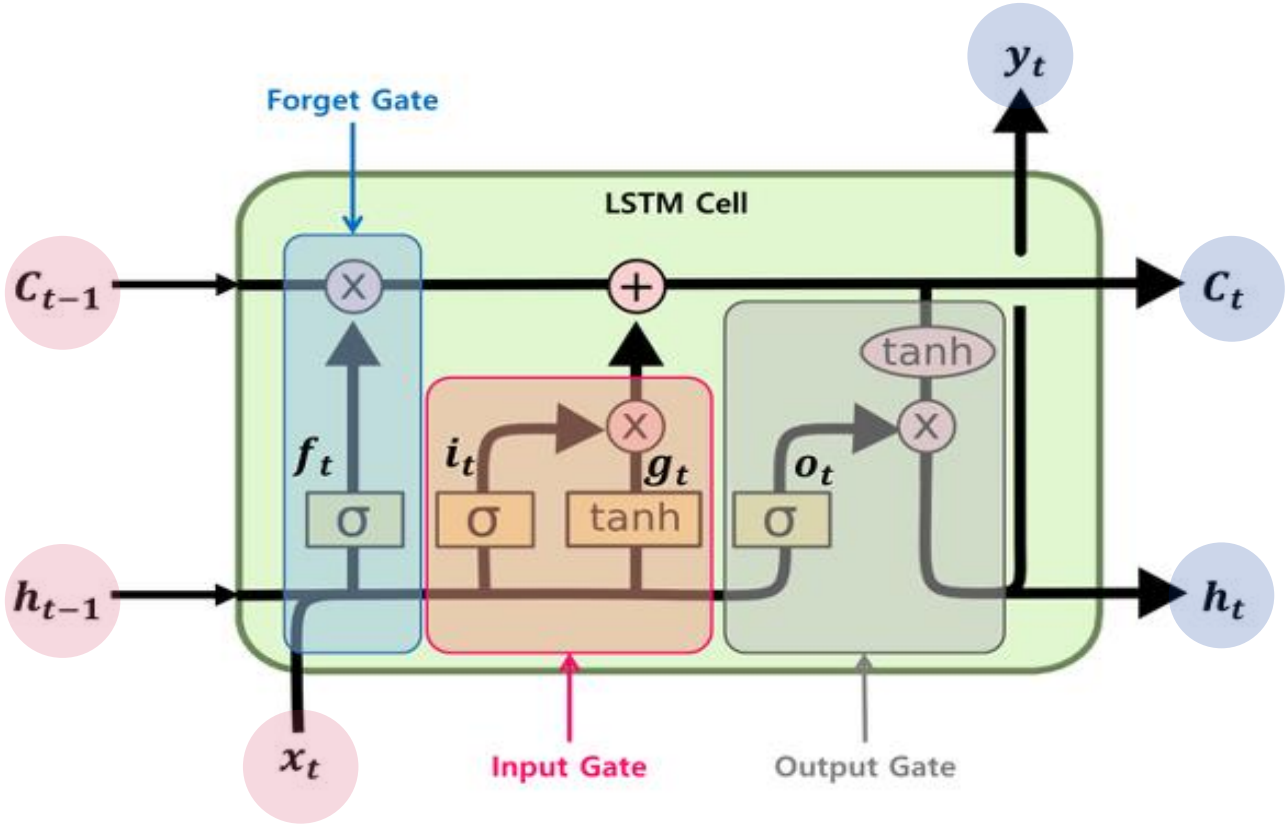
다음 time step으로 정보들을 전달하는 일만 맡은

Cell State가 추가로 존재

LSTM



- LSTM의 구조



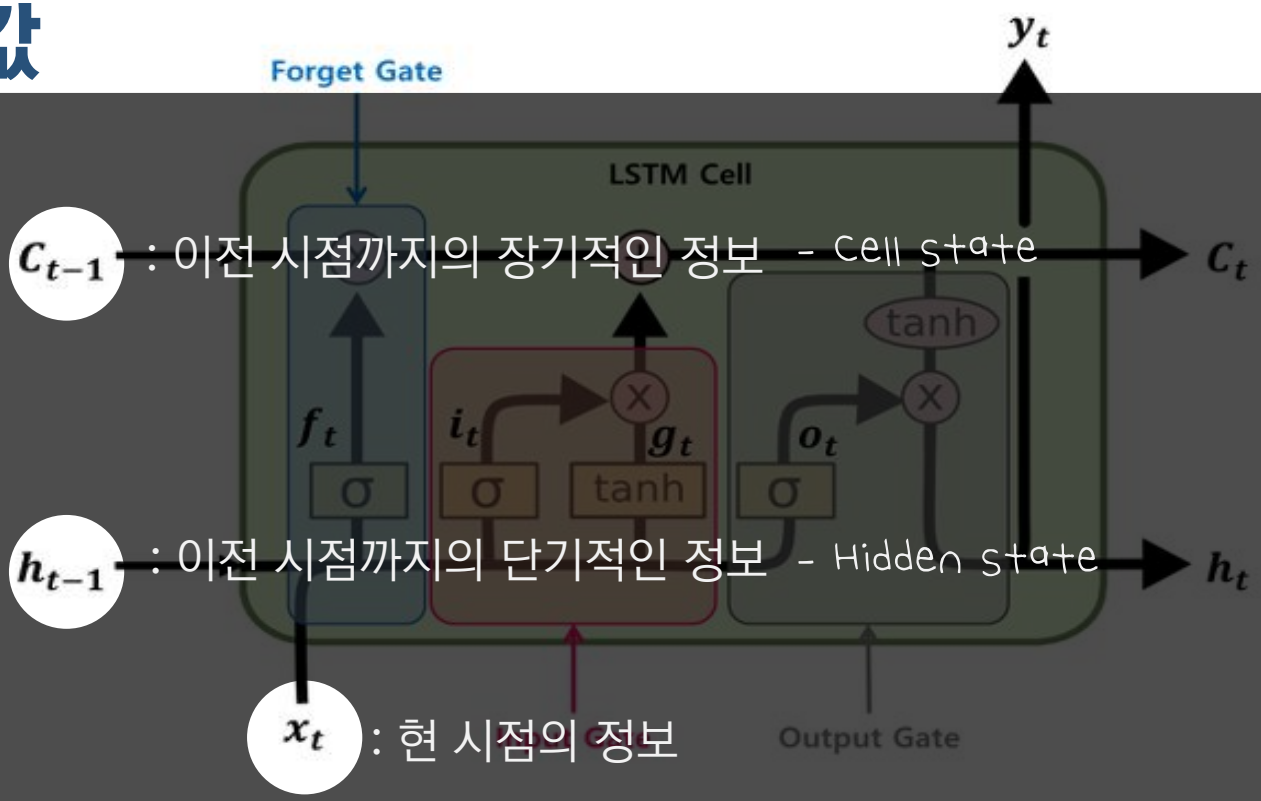
3 Inputs

3 Gates

3 Outputs

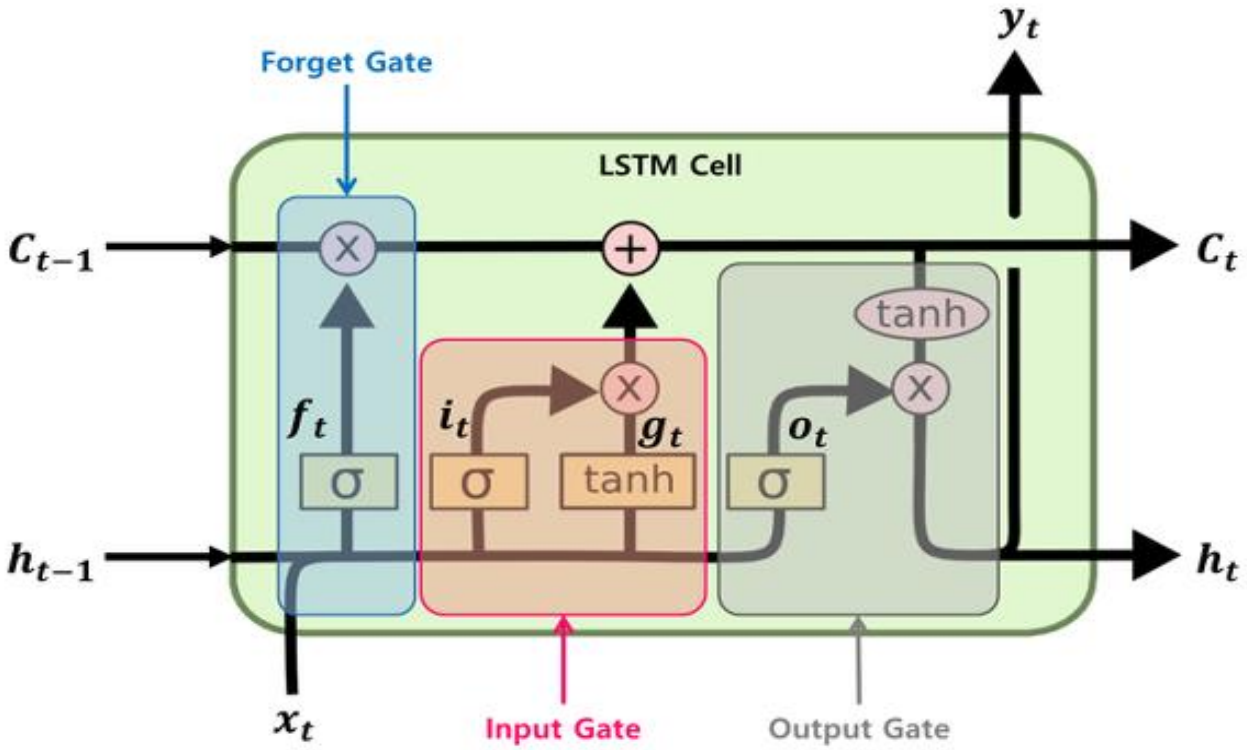
- LSTM의 구조

입력값


Forget Gate
LSTM Cell
y_t
C_{t-1} : 이전 시점까지의 장기적인 정보 - Cell state
C_t
tanh
f_t
i_t
g_t
o_t
σ
σ
tanh
σ
h_{t-1} : 이전 시점까지의 단기적인 정보 - Hidden state
h_t
x_t : 현 시점의 정보
Output Gate

- LSTM의 구조

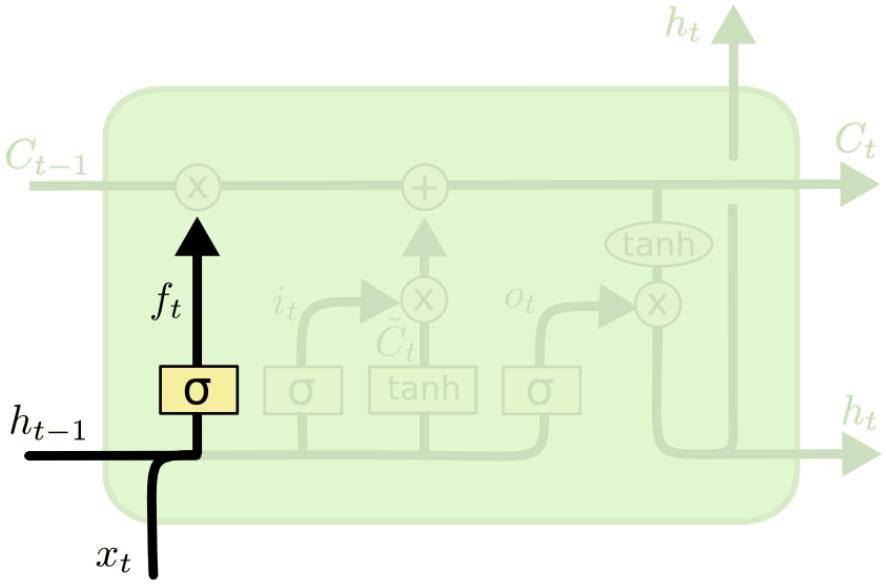
Gate



어떤 값을 어느 정도만큼 통과시킬지 결정하는 역할

- LSTM의 구조

Gate – Forget Gate



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

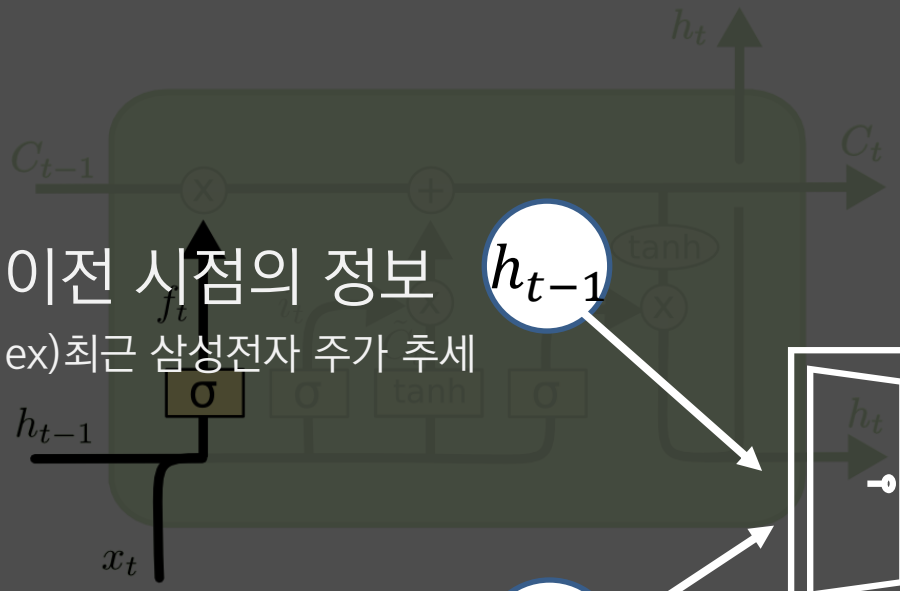
$$0 \le f_t \le 1$$

h_{t-1}와 x_t를 고려해 이전 시점까지의 장기적인 정보 C_{t-1}에서

어떤 정보가 쓸모가 없고 이를 얼마나 잊을 지 결정

- LSTM의 구조

Gate – Forget Gate



이전 시점의 정보

ex)최근 삼성전자 주가 추세

h_{t-1}

현재 시점의 정보

ex)오늘 삼성전자 주가, 거래량 등

x_t

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

f_t

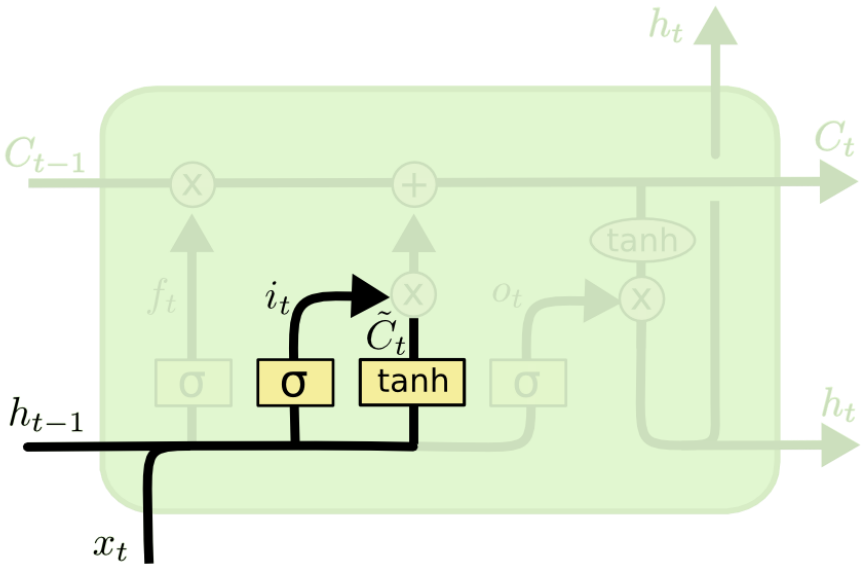
이전 시점의 장기 정보를 얼마나 잊을 지 결정

코로나 전이랑은 많이 다르네..
옛날 정보는 이제 쓸모 없겠지?

h_{t-1}과 x_t를 고려해 이전 시점까지의 장기적인 정보 C_{t-1}에서
어떤 정보가 쓸모가 없고 이를 얼마나 잊을 지 결정

- LSTM의 구조

Gate – Input Gate



정보를 얼마나 업데이트 할 것인지

$0 \le i_t \le 1$

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

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

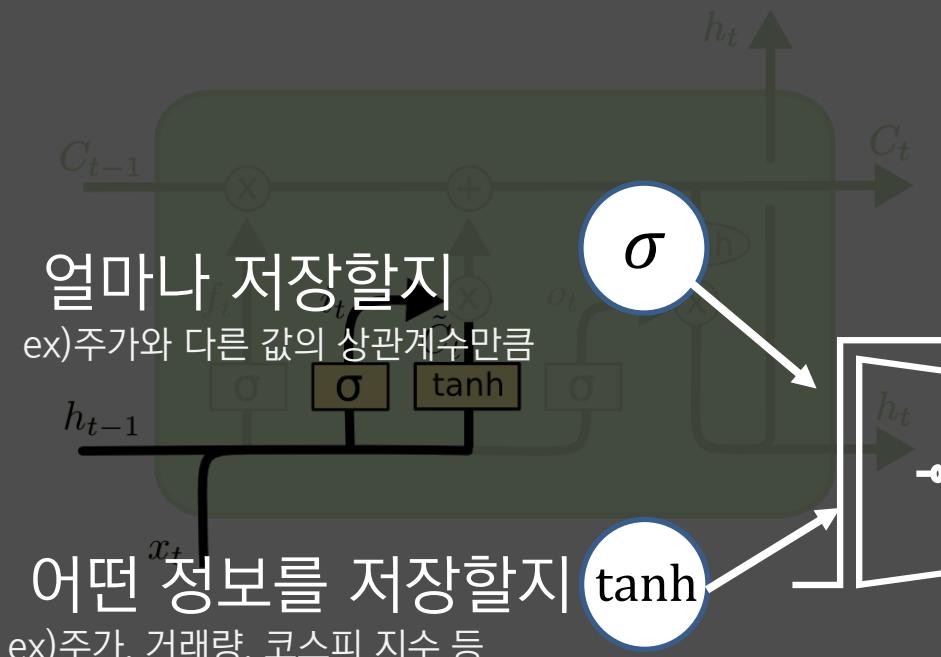
$-1 \le \tilde{C}_t \le 1$

어떤 정보를 더할 것인지

장기적인 정보(C_t)에 새로운 정보($\tilde{C}_t$)를 얼마나(i_t) 저장할지 결정하는 역할

- LSTM의 구조

Gate – Input Gate



얼마나 저장할지

ex)주가와 다른 값의 상관계수만큼

어떤 정보를 저장할지

ex)주가, 거래량, 코스피 지수 등

정보를 얼마나 업데이트 할 것인지

$0 \le i_t \le 1$

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

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

$\tilde{C}_t$ 장기 정보에 어떤 값을 얼마나 저장할지

$-1 \le \tilde{C}_t \le 1$

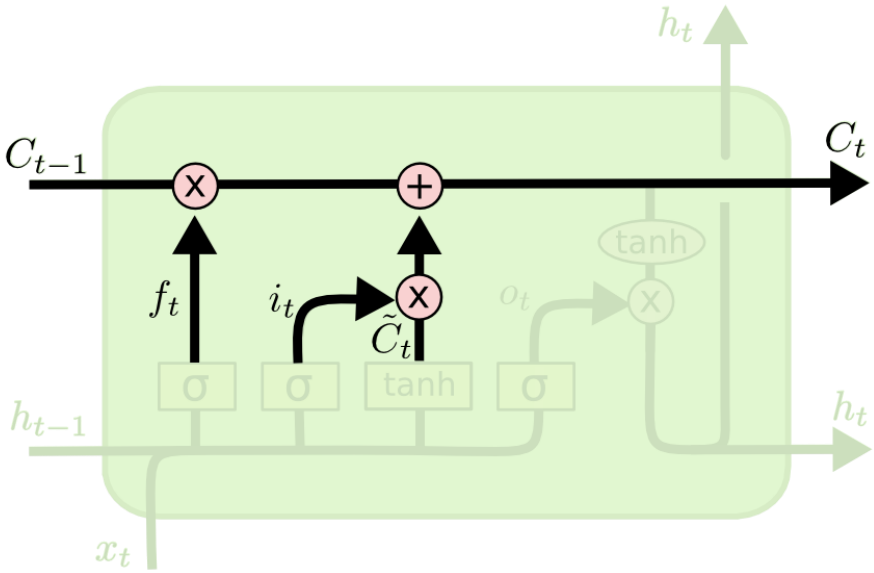
어떤 [illegible]할 것인지

i_t

장기적인 정보(C_t)에 **새로운 정보**($\tilde{C}_t$)를 얼마나(i_t) 저장할지 결정하는 역할

- LSTM의 구조

Cell State

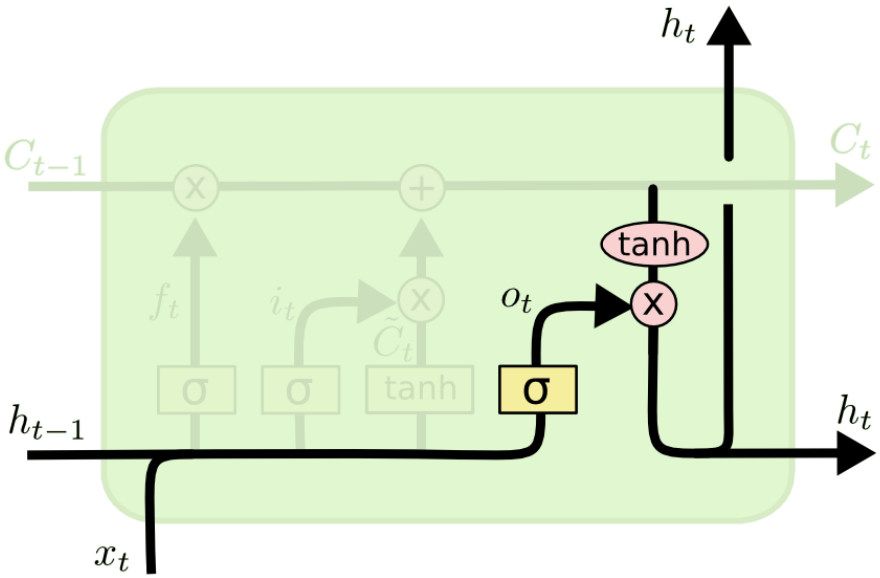


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

forget gate만큼 **정보를 잃고** input gate만큼 **새로운 정보**를 갖도록 업데이트후 다음 time step으로 전달

- LSTM의 구조

Gate – Output Gate



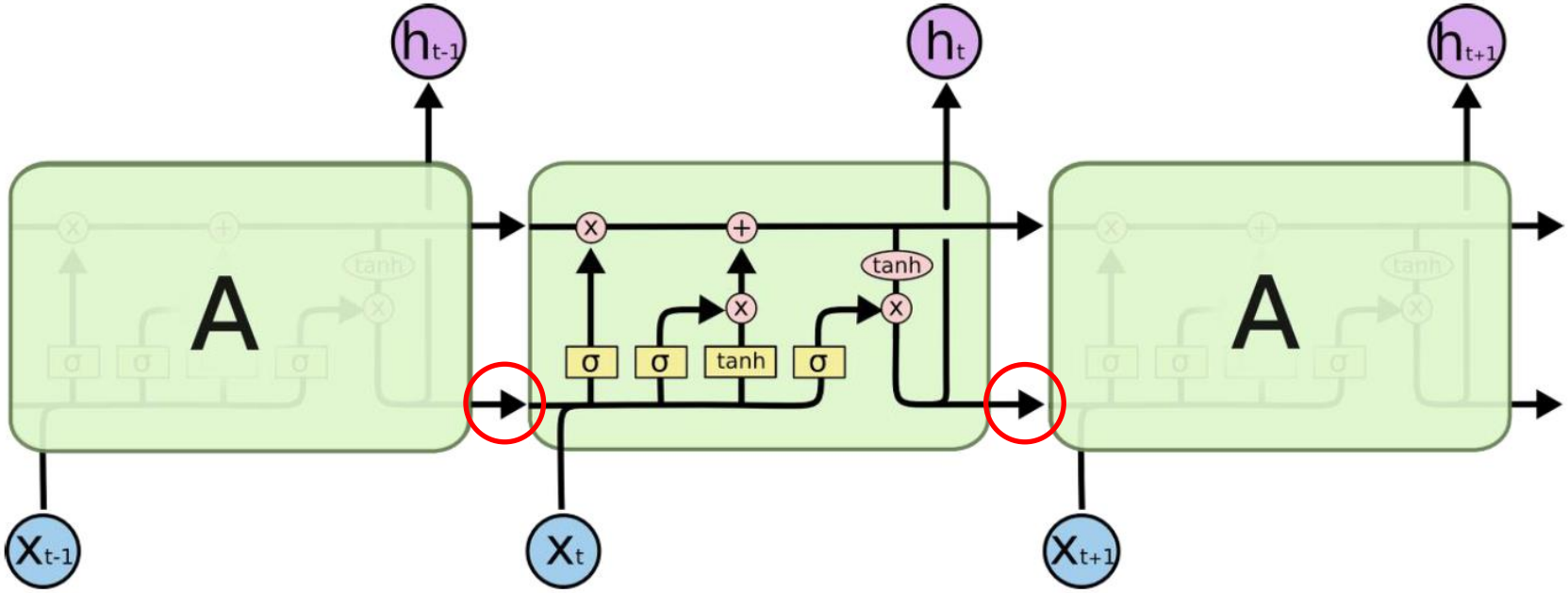
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

현 시점의 cell state를 $\boldsymbol{h_{t-1}}$, $\boldsymbol{x_t}$로 결정한 만큼 출력($\boldsymbol{h_t}$)

- LSTM의 구조

Gate – Output Gate

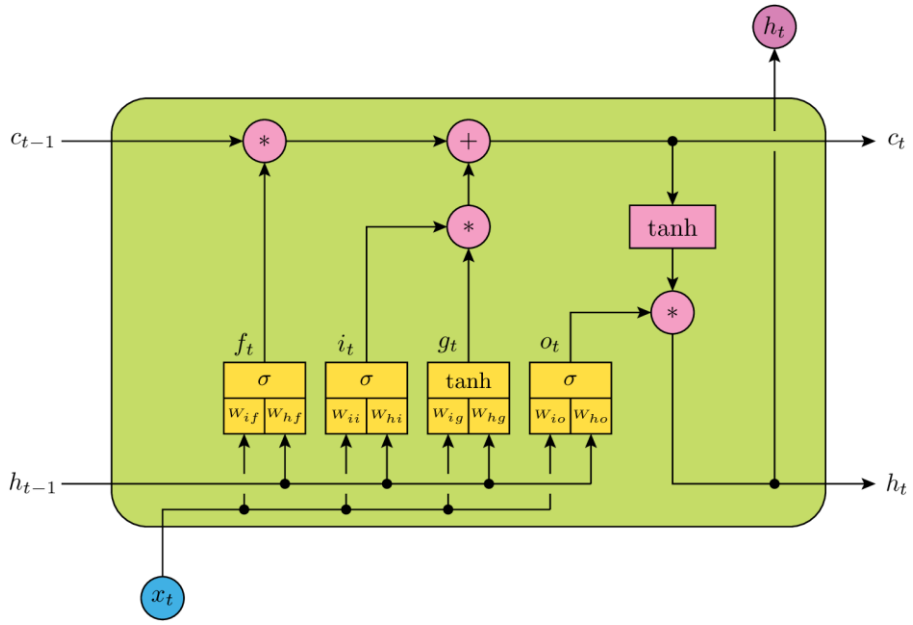


출력한 값은 다음 time step으로 전해지고

다음 time step에서 같은 과정 반복

- LSTM의 학습

LSTM의 역전파

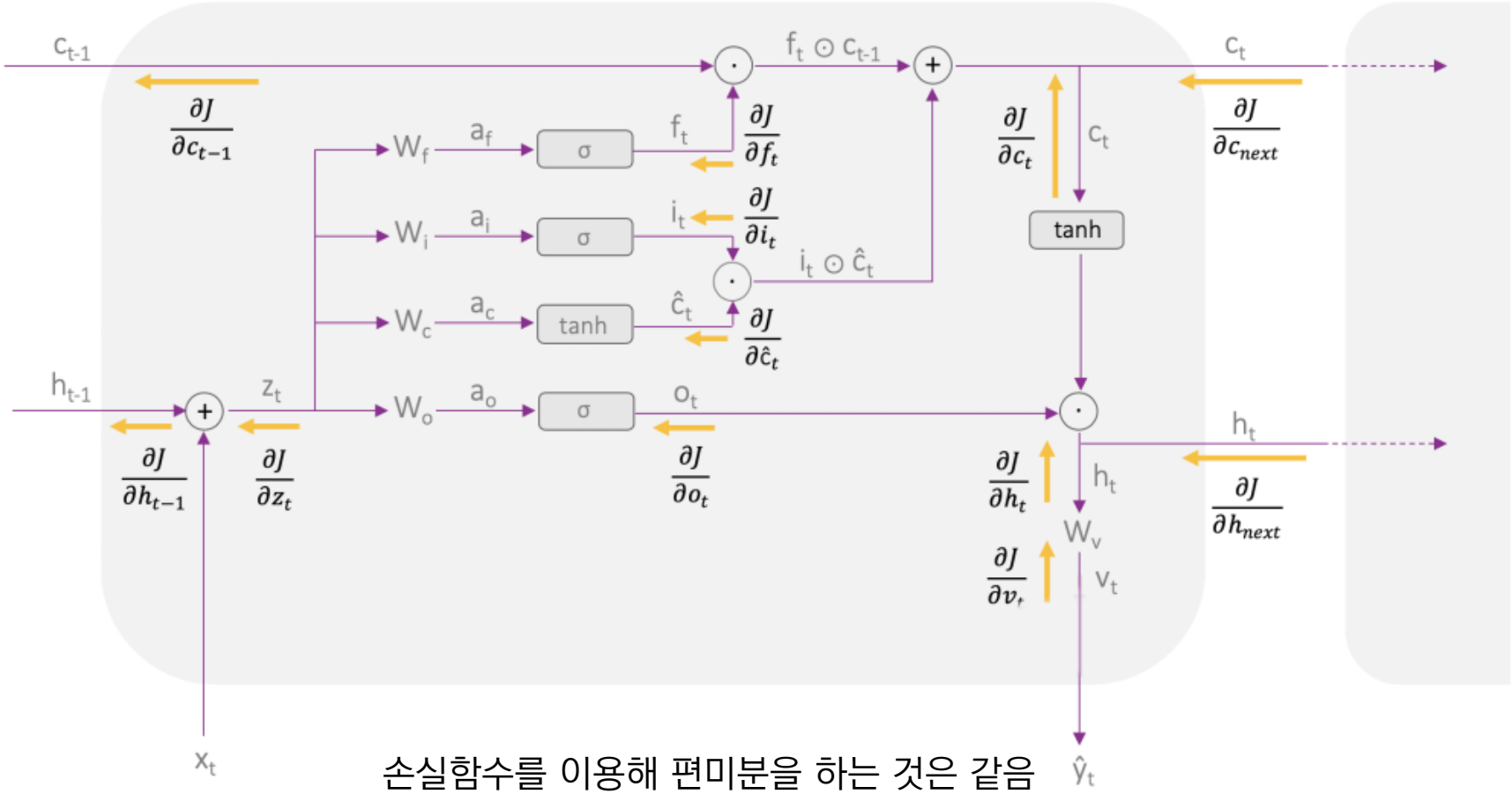


$$f_t = \sigma(x_t \times w_{if} + h_{t-1} \times w_{hf} + b_f)$$
$$o_t = \sigma(x_t \times w_{io} + h_{t-1} \times w_{ho} + b_o)$$
$$g_t = \sigma(x_t \times w_{ig} + h_{t-1} \times w_{hg} + b_g)$$
$$i_t = \sigma(x_t \times w_{ii} + h_{t-1} \times w_{hi} + b_i)$$

총 8개의 가중치와 4개의 편향을 업데이트 해야함

3 LSTM

- LSTM의 역전파

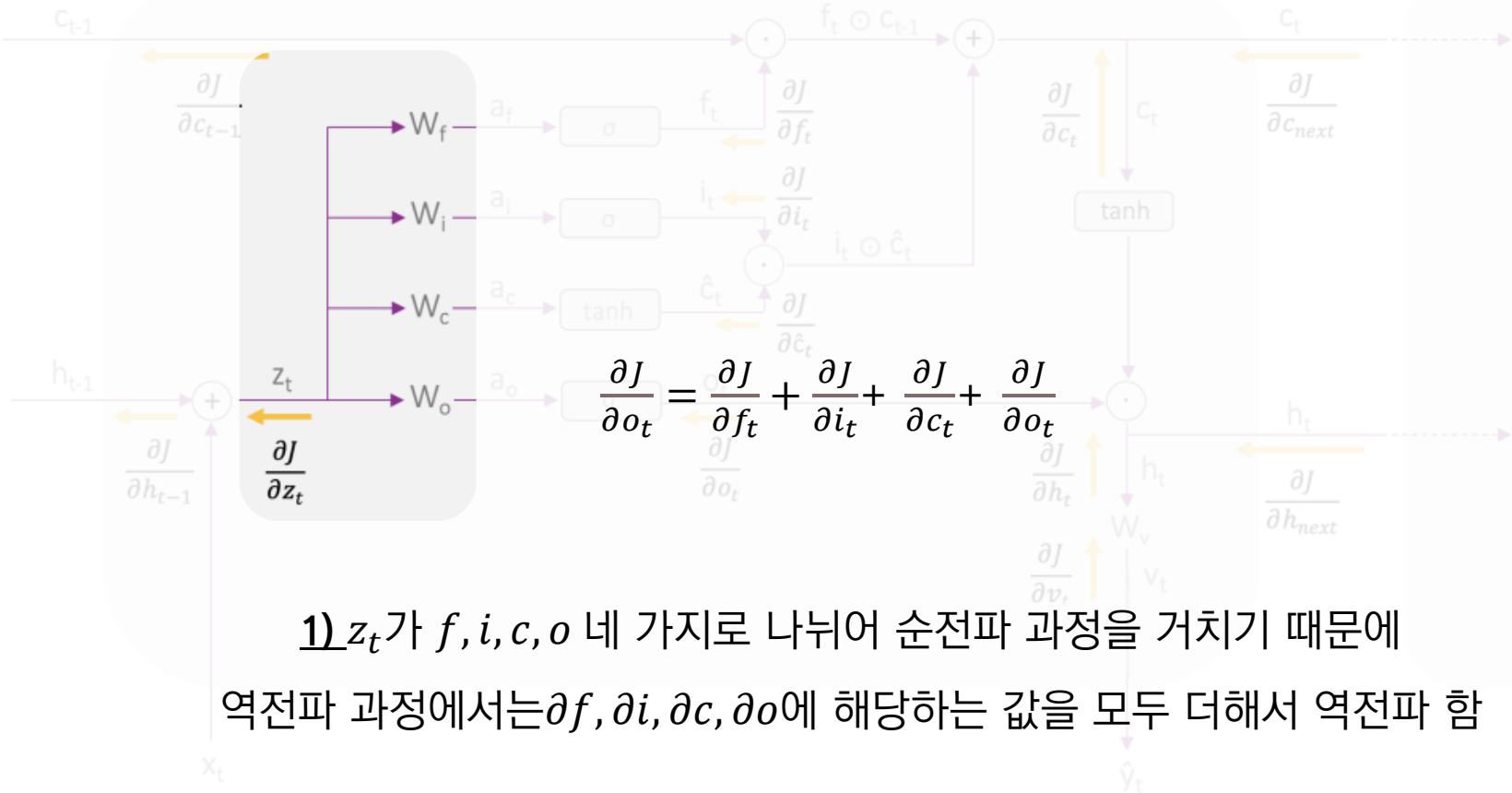


손실함수를 이용해 편미분을 하는 것은 같음

2가지 사항만 주의할 것

3 LSTM

- LSTM의 역전파



1) z_t가 f, i, c, o 네 가지로 나뉘어 순전파 과정을 거치기 때문에

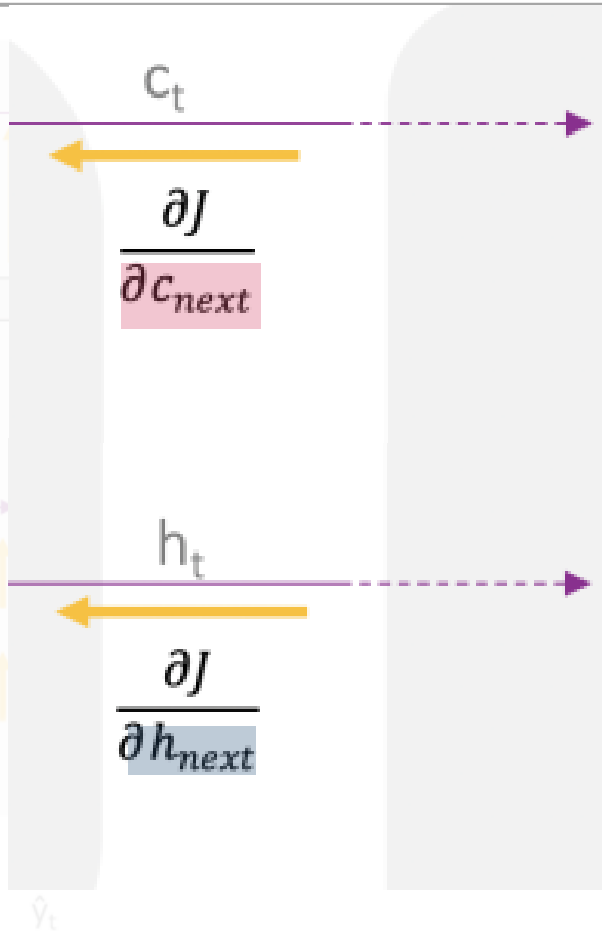
역전파 과정에서는$\partial f, \partial i, \partial c, \partial o$에 해당하는 값을 모두 더해서 역전파 함

손실함수를 이용해 편미분을 하는 것은 같음

2가지 사항만 주의할 것

3 LSTM

- **LSTM의 역전파**



2) h_t, C_t**는** h_{t-1}, C_{t-1} 의 영향을 받기 때문에

모든 time step의 hidden state, cell state 값을 고려해서

각각의 gradient를 계산해야 함

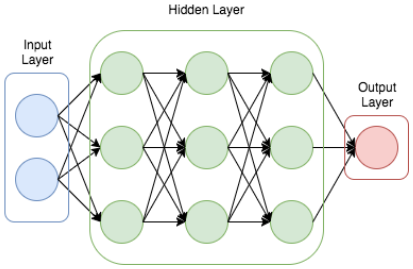
손실함수를 이용해 편미분을 하는 것은 같음

2가지 사항만 주의할 것

3 정리

- **최종정리!**

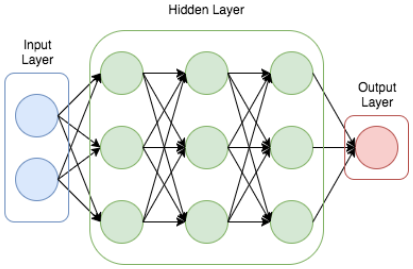
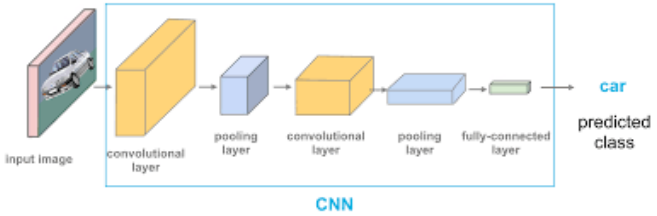
DNN



input	한계
벡터	벡터로만 input 받음

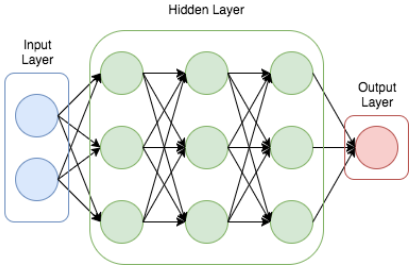
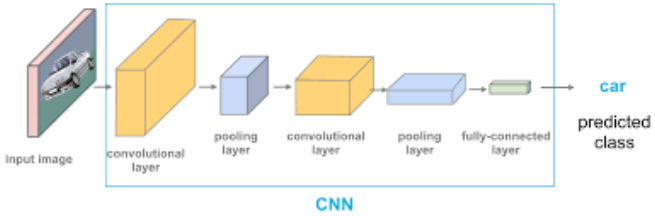
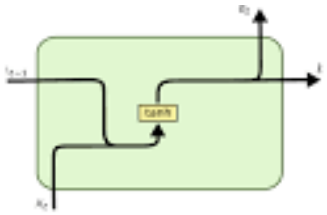
3 정리

- **최종정리!**

		input	한계
DNN		벡터	벡터로만 input 받음
CNN		이미지 데이터	시간적 특징 반영 불가

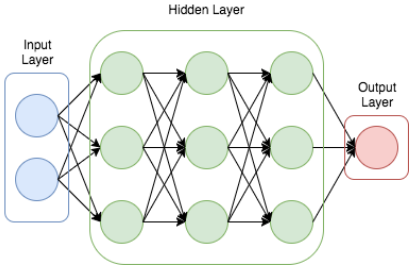
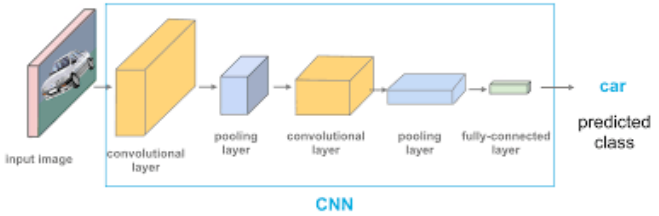
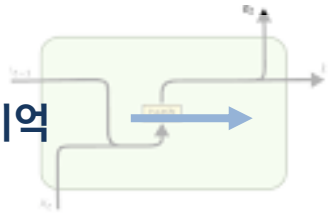
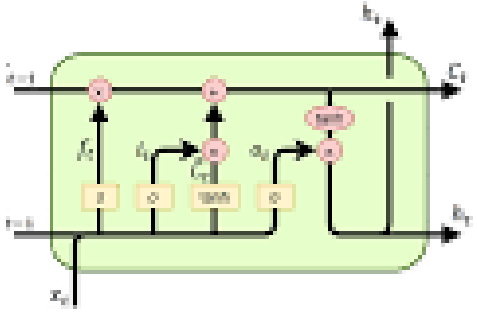
3 정리

- **최종정리!**

		input	한계
DNN		벡터	벡터로만 input 받음
CNN		이미지 데이터	시간적 특징 반영 불가
RNN		자연어 데이터	장기의존성

3 정리

- **최종정리!**

		input	한계
DNN		벡터	벡터로만 input 받음
CNN		이미지 데이터	시간적 특징 반영 불가
RNN	+ 장기기억 	LSTM	자연어  성

THANK YOU