딥러닝팀

1팀

김재희
유경민
김주연
문서영
이수경

INDEX

1. CNN 모델

2. RNN 모델

3. LSTM

4. LSTM의 변형 모델들

1

CNN 모델

1 CNN 모델

● 이미지 처리 모델의 역사

〈원본 이미지〉

〈외곽선 추출〉

〈흐리게 하기〉

딥러닝이 도입되기 이전에는

이미지의 특징을 분명하게 만드는 다양한 함수들에 대한 연구가 이루어짐

● 이미지 처리 모델의 역사

필터

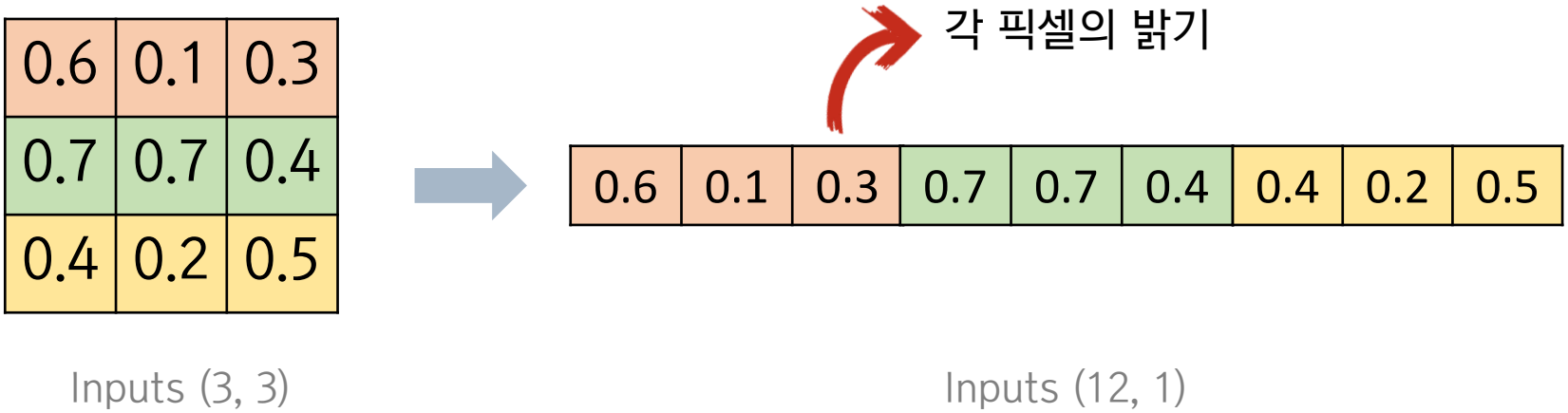
0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	1
0	1	1	0	0	1	1	1
1	1	0	0	1	1	1	1
1	0	0	0	1	1	1	1
0	1	1	1	0	0	1	1
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	1

이미지 특징의 추출을 위해 커널(일정한 수를 가진 행렬)을 사용

원하는 결과물 추출을 위한 최적의 커널이 무엇인지 알 수 없었음

- 딥러닝과 이미지 처리

Flatten 함수

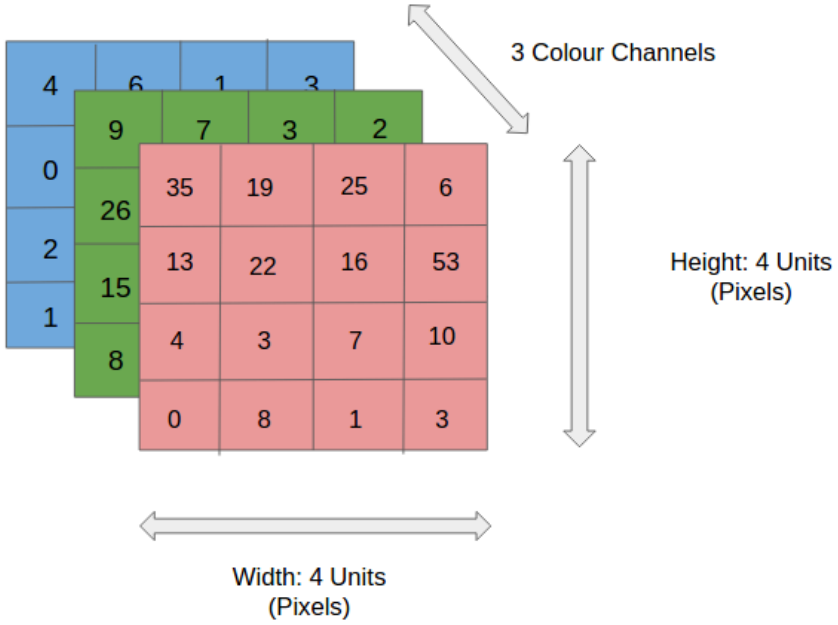


행렬을 벡터 꼴로 바꿔주어 FFNN 모델 사용가능

1 CNN 모델

● 이미지의 특징 (FFNN의 한계)

1. 채널



EX. 1024 x 560 해상도의 컬러 이미지 100장

→ (100, 3, 1024, 560)

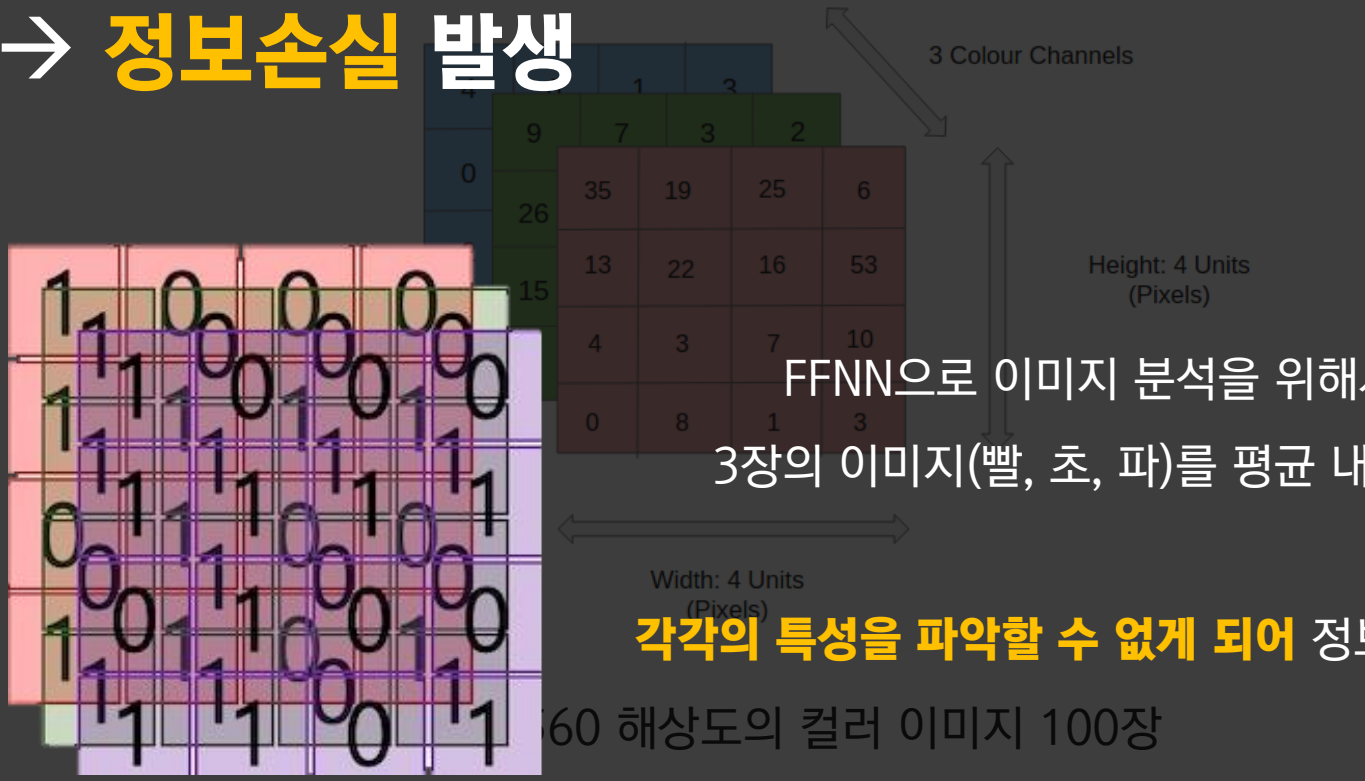
(데이터, 채널, 행, 열)

1 CNN 모델

- 이미지의 특징 (FFNN의 한계)

1. 채널

→ **정보손실** 발생



FFNN으로 이미지 분석을 위해서는

3장의 이미지(빨, 초, 파)를 평균 내어 사용

각각의 특성을 파악할 수 없게 되어 정보 손실 발생

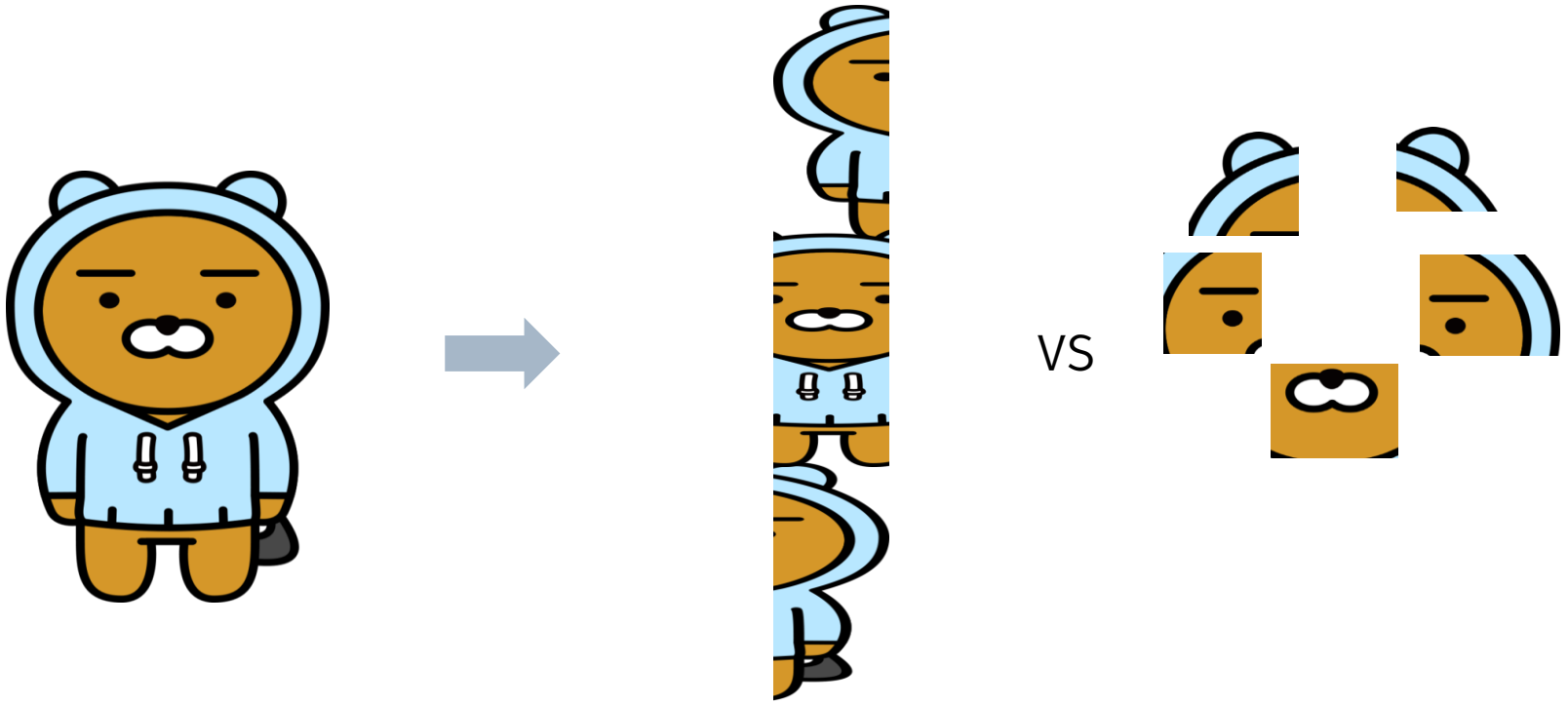
60 해상도의 컬러 이미지 100장

→ (100, 3, 1024, 560)

(데이터, 채널, 행, 열)

- 이미지의 특징 (FFNN의 한계)

2. 지역적 특징



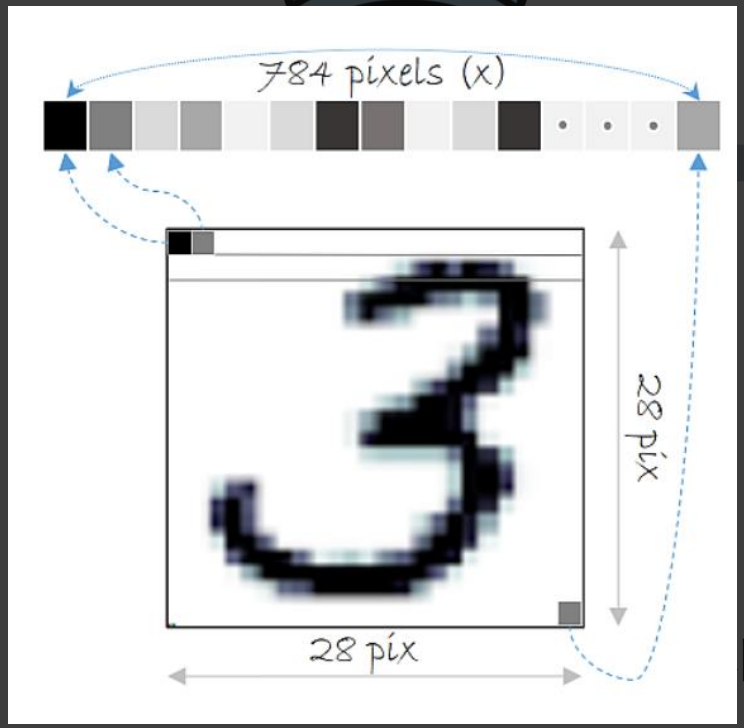
공간(위치) 정보가 중요함

1 CNN 모델

- 이미지의 특징 (FFNN의 한계)

2. 지역적 특징

→ **위치정보** 누락

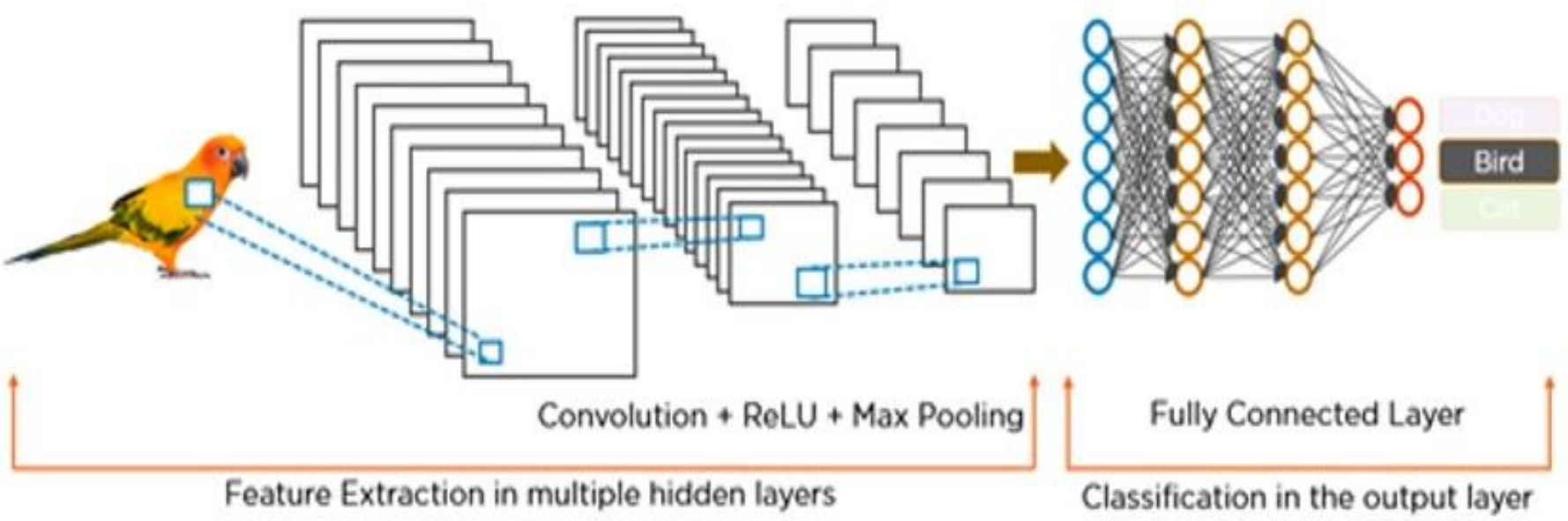


FFNN은 공간 정보가 전혀 반영되지 않아서
이미지의 위치정보가 누락됨

) 정보가 중요함

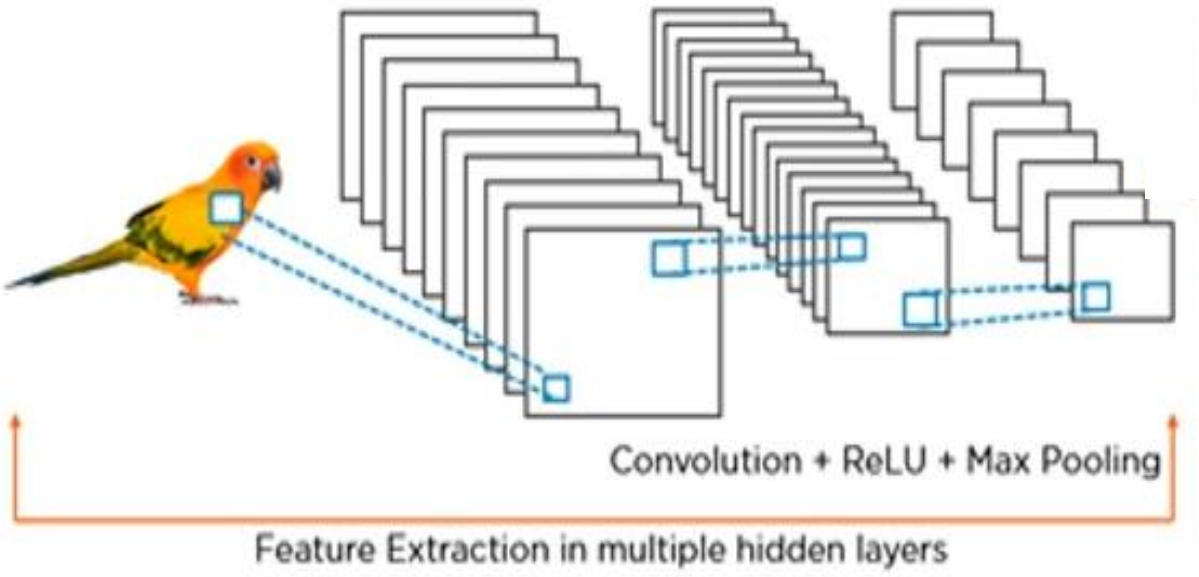
1 CNN 모델

● CNN 모델의 전체 구조



1 CNN 모델

● CNN 모델의 전체 구조

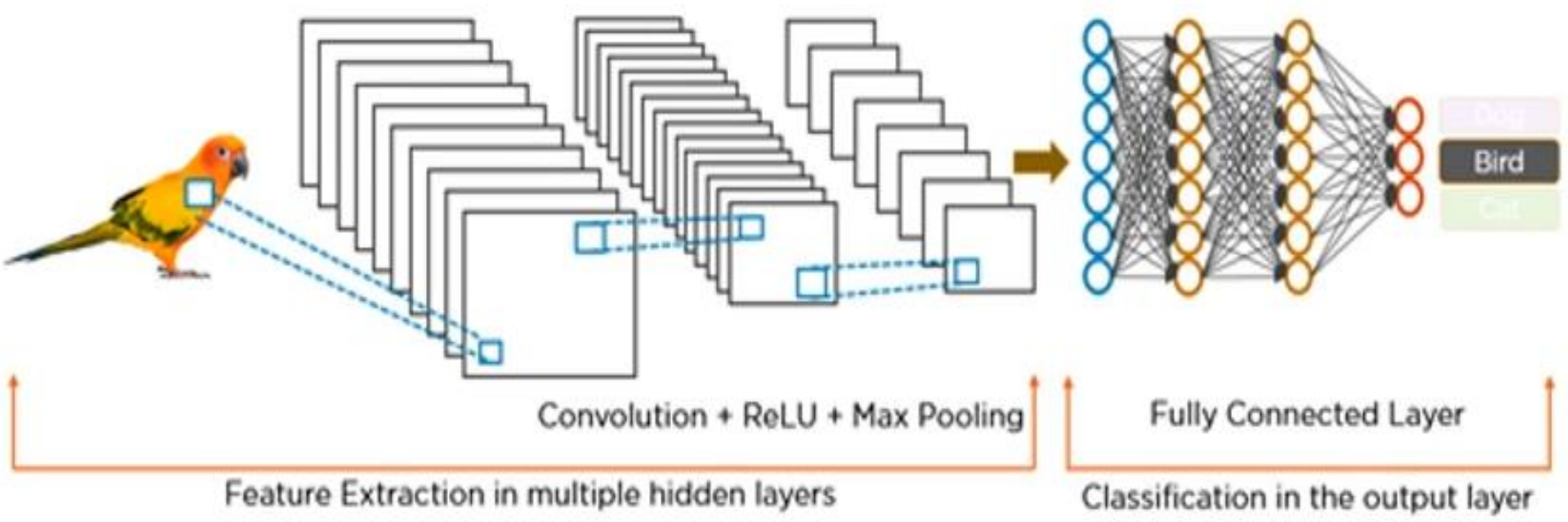


Convolution layer(패딩 – 필터 – 활성화 함수)

CNN 모델을 통해 노이즈를 제거하고 중요한 변수들을 강조

1 CNN 모델

● CNN 모델의 전체 구조

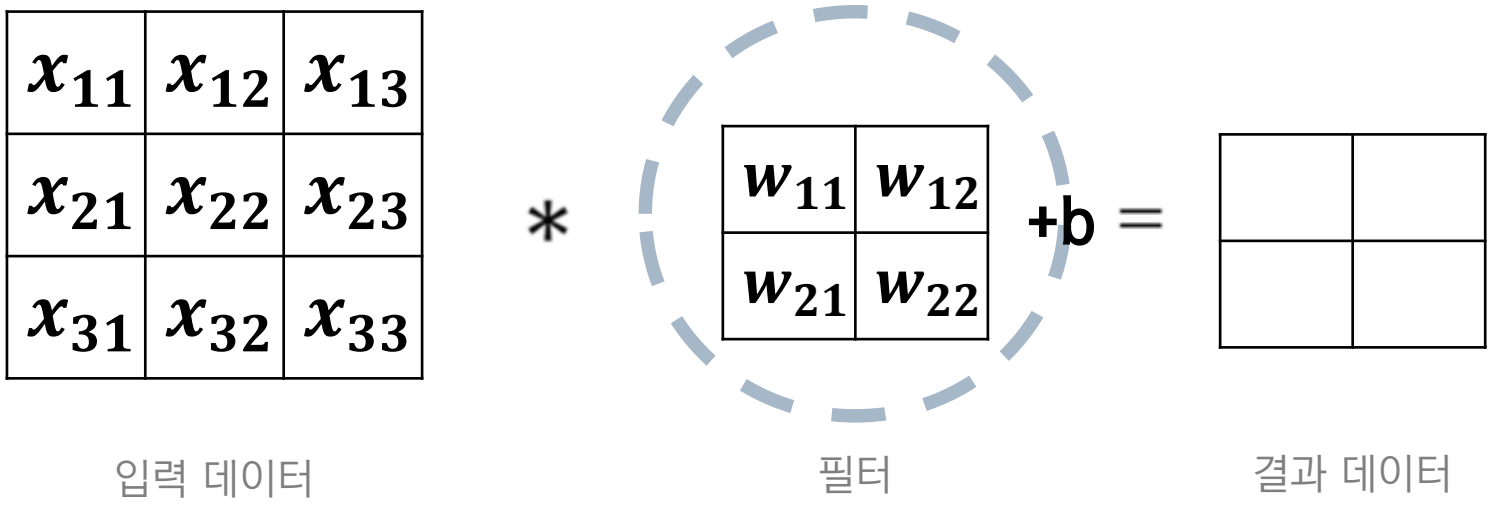


FFNN 모델 계층

마지막에 결과값을 산출하기 위해 추가됨

- CNN(합성곱 신경망)의 요소

1. 필터(Filter)

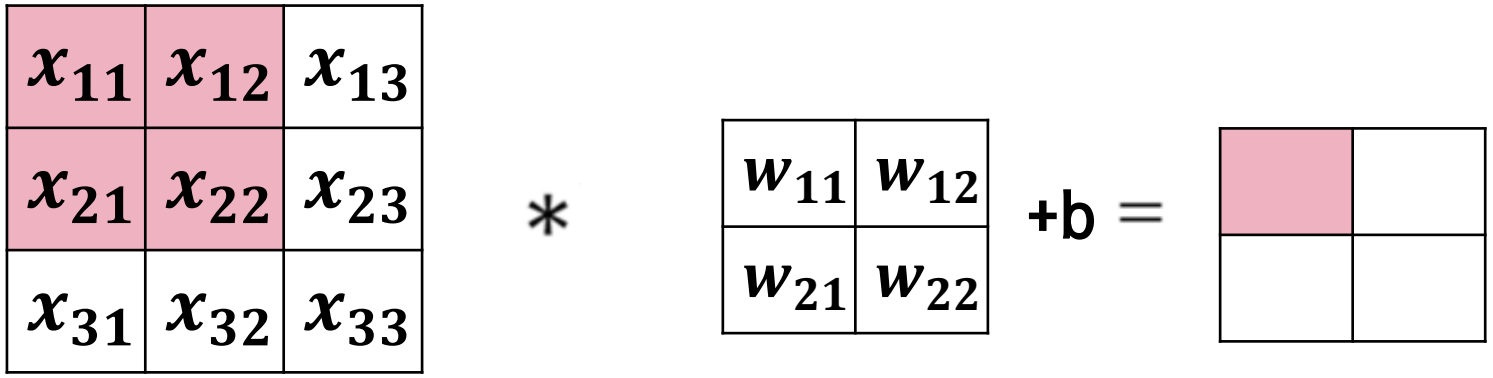


필터 안에 위치한 숫자들이 가중치 역할

합성곱 (*) : 같은 위치의 값끼리 곱해서 각각의 값들을 더해주면 된다

- CNN(합성곱 신경망)의 요소

1. 필터(Filter)

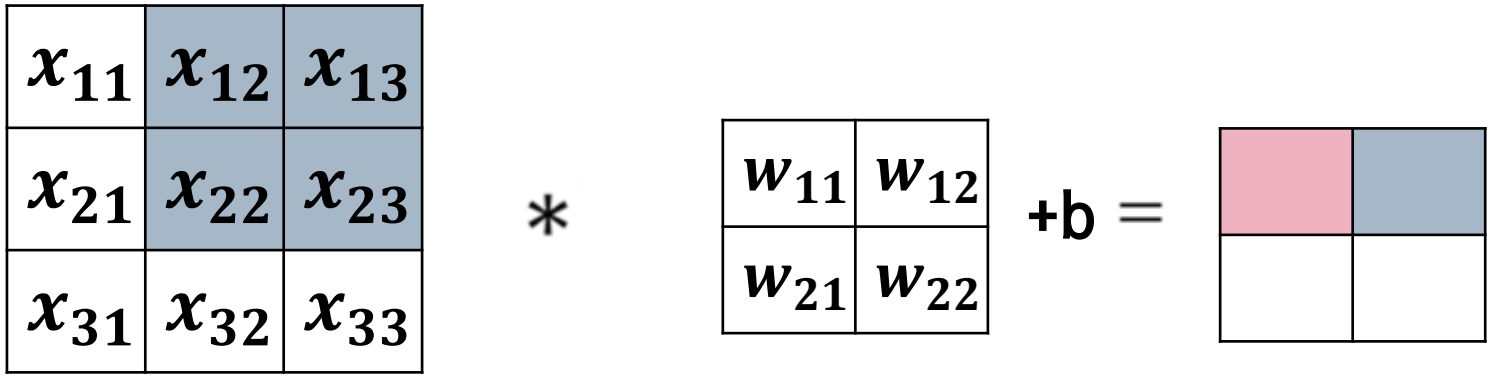


입력 데이터 필터 결과 데이터

$$x_{11} * w_{11} + x_{12} * w_{12} + x_{21} * w_{21} + x_{22} * w_{22} + b$$

- CNN(합성곱 신경망)의 요소

1. 필터(Filter)

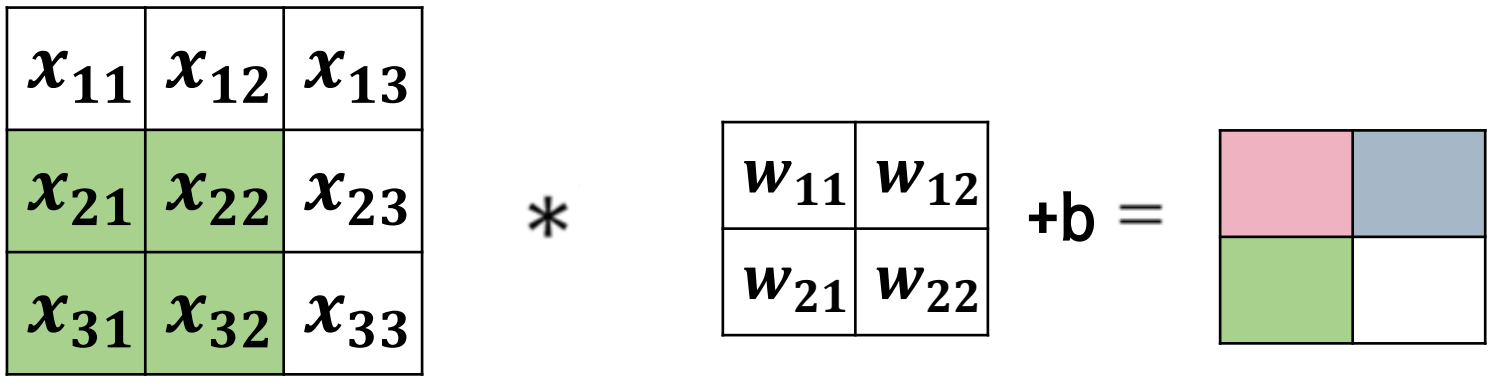


입력 데이터 필터 결과 데이터

$$x_{12} * w_{11} + x_{13} * w_{12} + x_{22} * w_{21} + x_{23} * w_{22} + b$$

- CNN(합성곱 신경망)의 요소

1. 필터(Filter)

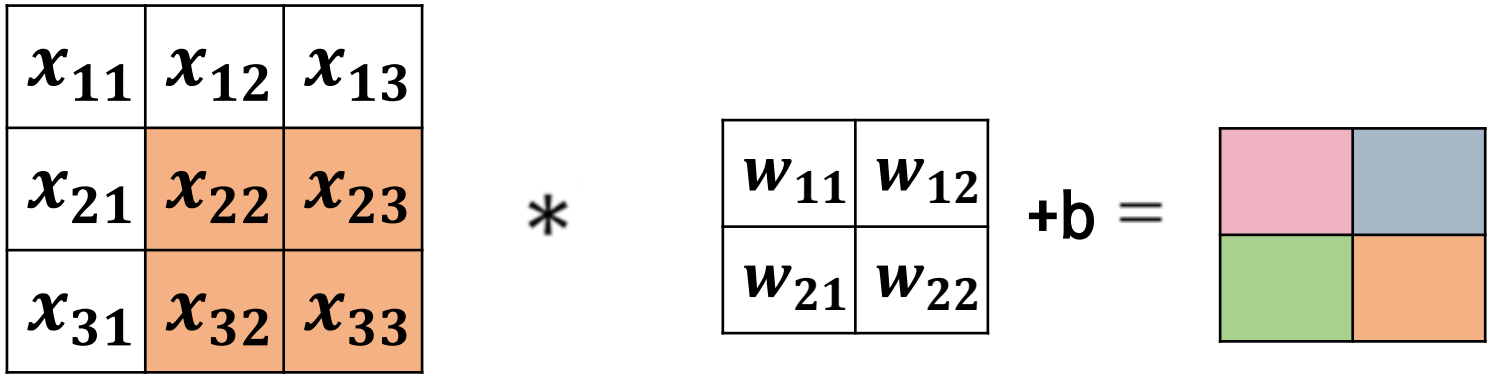


입력 데이터 필터 결과 데이터

$$x_{21} * w_{11} + x_{22} * w_{12} + x_{31} * w_{21} + x_{32} * w_{22} + b$$

- CNN(합성곱 신경망)의 요소

1. 필터(Filter)

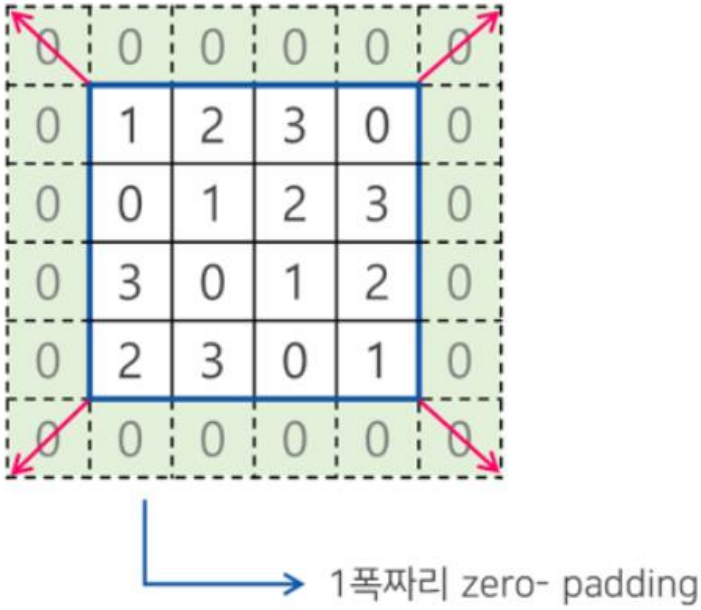


입력 데이터 필터 결과 데이터

$$x_{22} * w_{11} + x_{23} * w_{12} + x_{32} * w_{21} + x_{33} * w_{22} + b$$

- CNN(합성곱 신경망)의 요소

2. 패딩(Padding)

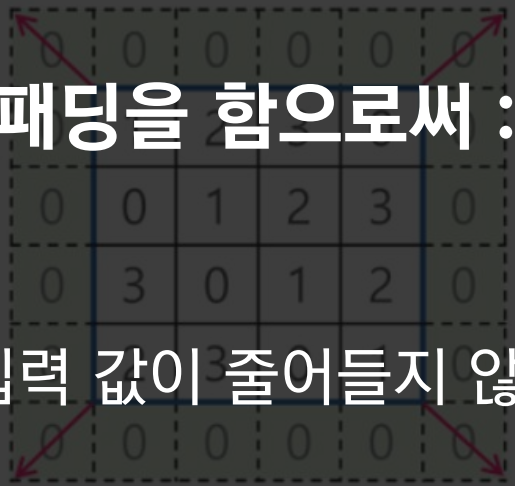


합성곱 연산 수행 전에 입력데이터 주변을 특정 값으로 채운다

- CNN(합성곱 신경망)의 요소

2. 패딩(Padding)

패딩을 함으로써 :



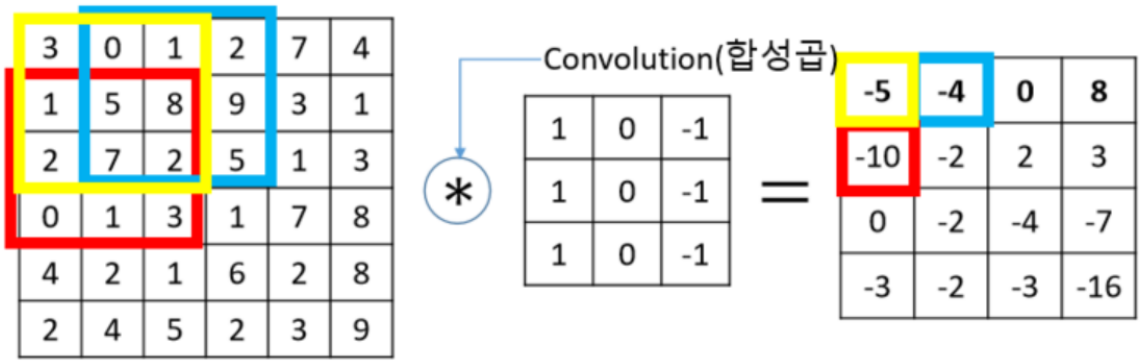
1. 입력 값이 줄어들지 않는다

2. 테두리를 인식시킬 수 있다

합성곱 연산 수행 전에 입력데이터 주변을 특정 값으로 채운다

- CNN(합성곱 신경망)의 요소

3. 스트라이드(Stride)



stride = 1

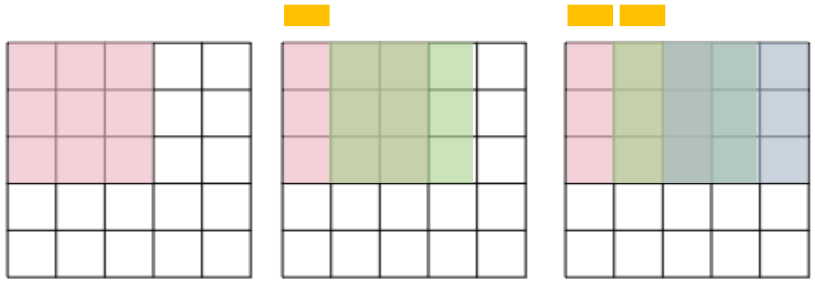
입력데이터에 필터를 적용할 때 필터가 이동할 간격

- CNN(합성곱 신경망)의 요소

3. 스트라이드(Stride)

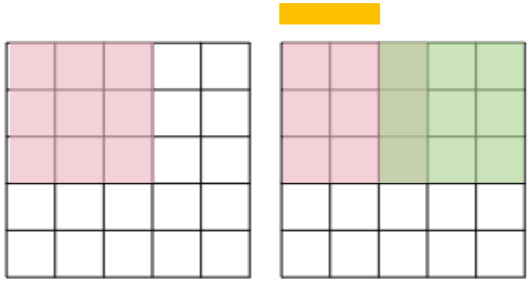
Stride = 1

필터를 한 칸씩 움직인다



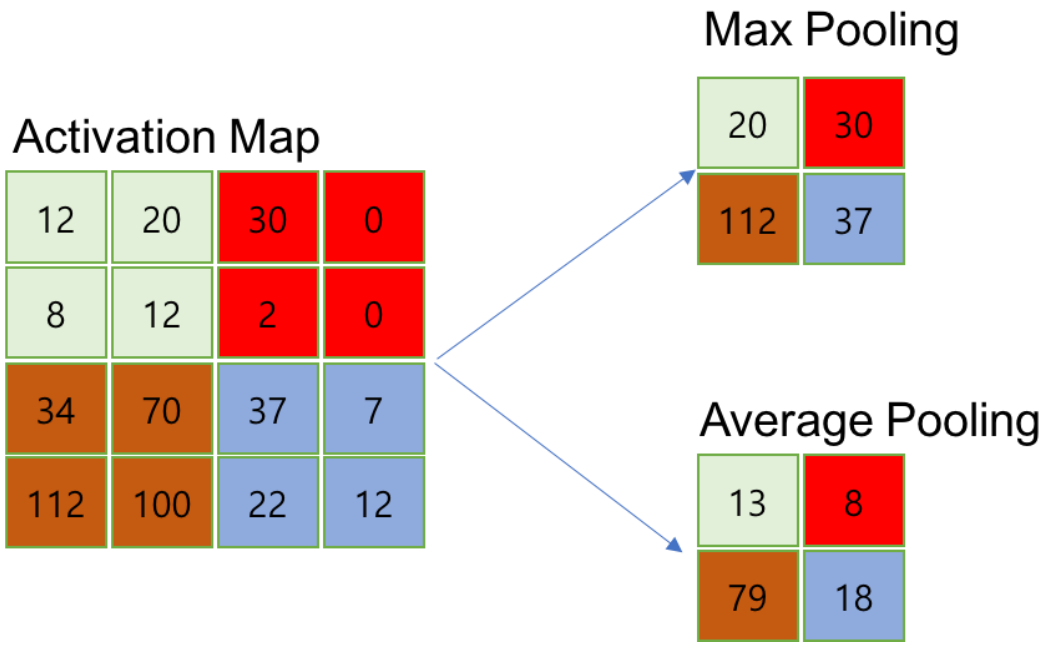
Stride = 2

필터를 두 칸씩 움직인다



- CNN(합성곱 신경망)의 요소

4. 풀링(Pooling)

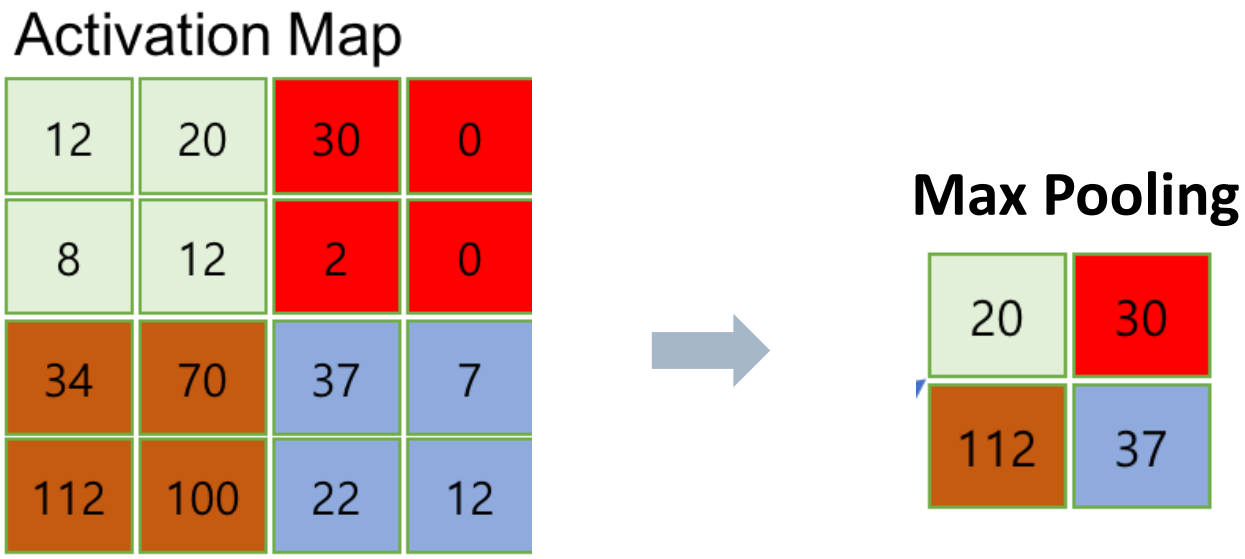


특정한 조건의 값만 선택하는 것

→ 데이터의 크기를 줄이면서 특징을 부각시킴

- CNN(합성곱 신경망)의 요소

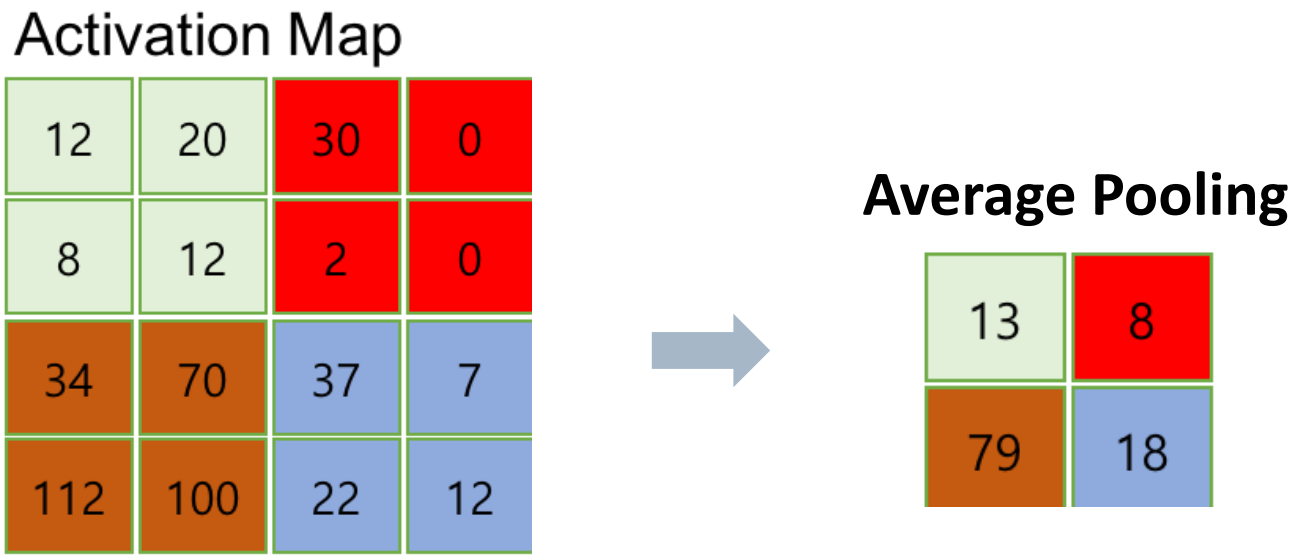
4. 풀링(Pooling)



최대값을 선택하는 Max Pooling

- CNN(합성곱 신경망)의 요소

4. 풀링(Pooling)



평균값을 선택하는 Average Pooling

- CNN(합성곱 신경망)의 요소

4. 풀링(Pooling)

특징

1. 노이즈 제거
2. 가중치가 없다
3. 지역적으로 데이터들이 가까워짐

효과

1. 이미지의 특징이 부각됨
2. 빠른 학습이 가능하다
3. 작은 크기의 필터로도 특징을 추출할 수 있다

1 CNN 모델

● CNN 모델의 전체 구조

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

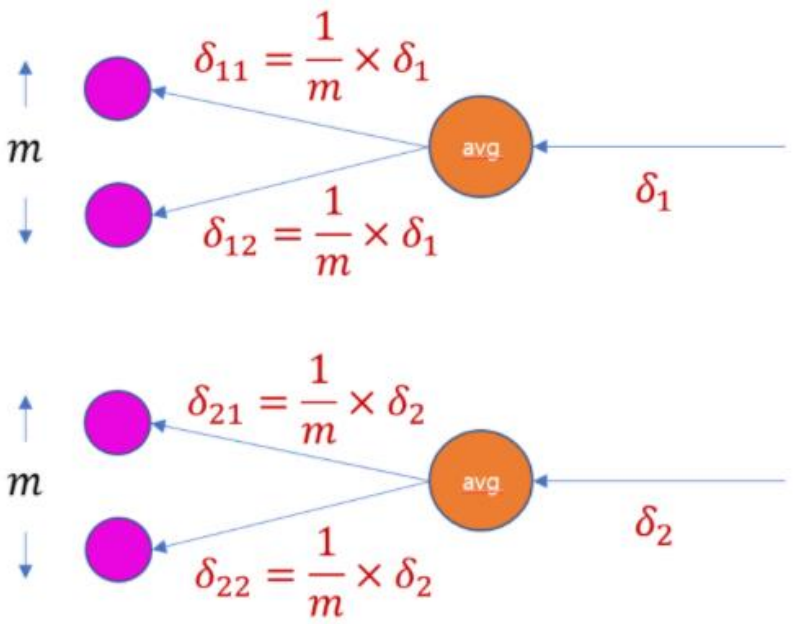
$$OutputWeight = OW = \frac{(W + 2P - FW)}{S} + 1$$

〈 입력값(H,W), 필터 크기(FH, FW), 출력크기(OH, OW), 패딩(P), 스트라이드(S) 〉

데이터, 필터, 패딩, 스트라이드의 크기가 결과 값의 크기를 결정함

● CNN 역전파

평균 풀링(Average Pooling)

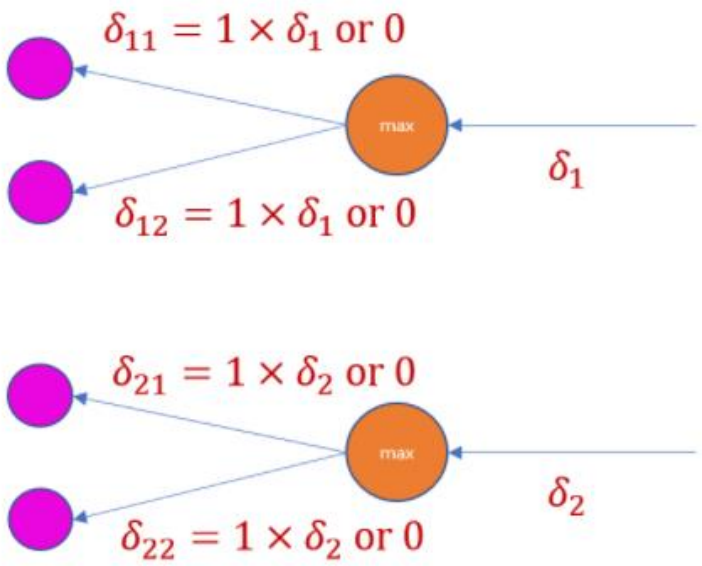


평균 풀링은 x값을 동일하게 반영해 결과를 산출하기 때문에

풀링의 크기가 m(row x col)일 때, 역전파를 모든 입력값에 m으로 나누어 배분함

● CNN 역전파

최대값 풀링(Max Pooling)

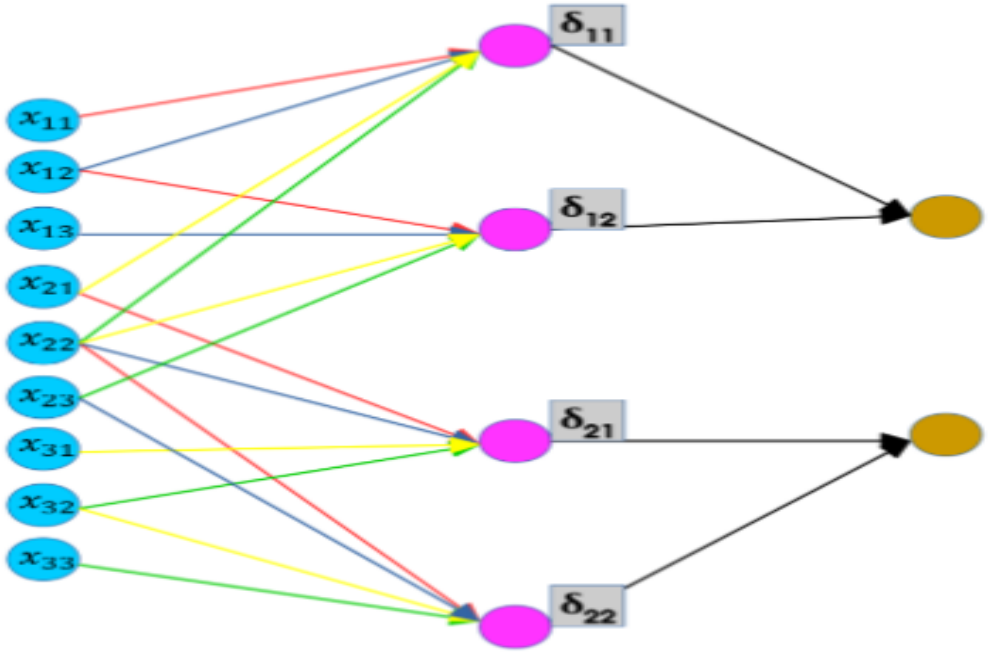


입력값의 최대값만 출력되어 역전파도 역시 최대값을 가진 변수에게만 흘러감

따라서 이외의 변수에는 아예 역전파가 전달되지 않음

● CNN 역전파

Convolution Layer



FFNN과는 달리 모든 노드가 서로 연결되어 있지 않음

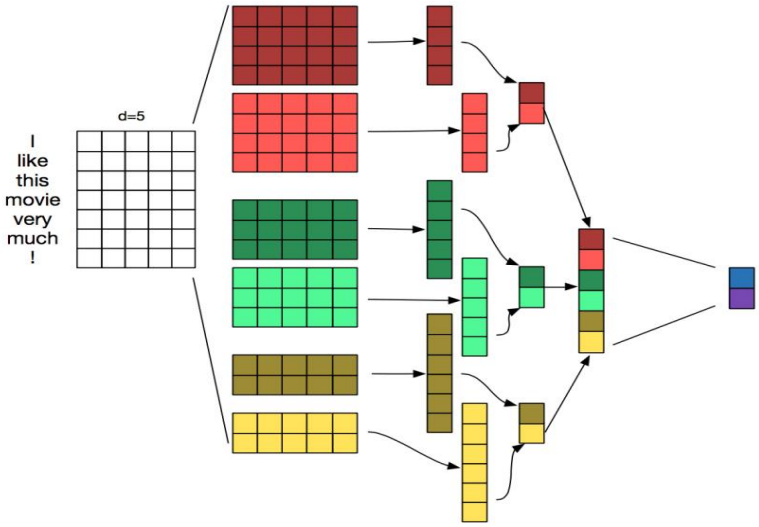
∴ 역전파 시 연결된 노드끼리만 진행

2

RNN 모델

- CNN의 한계

1. 순서를 반영하지 못함

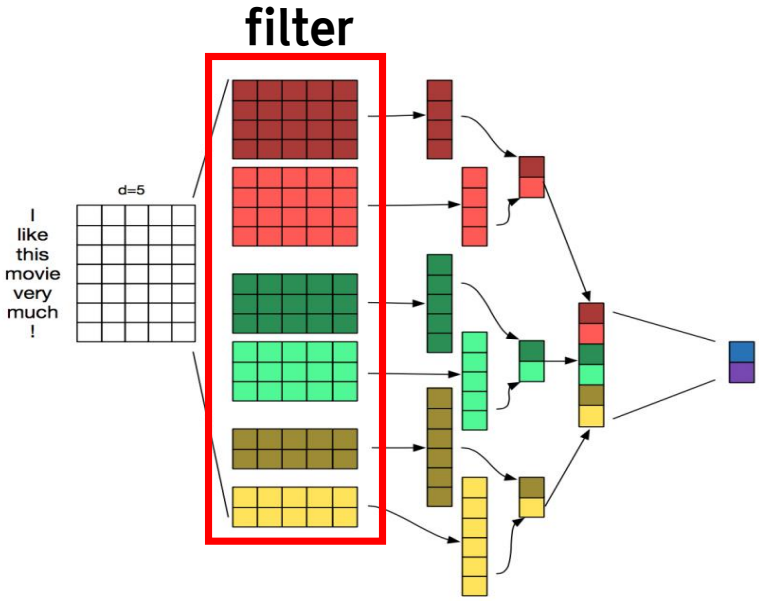


학습이 순차적으로 이루어지지 않음

➡ CNN은 지역적 정보만 파악할 뿐 시간적 특징을 반영하지는 못함

- CNN의 한계

2. 먼 거리의 값과 관계 파악 불가



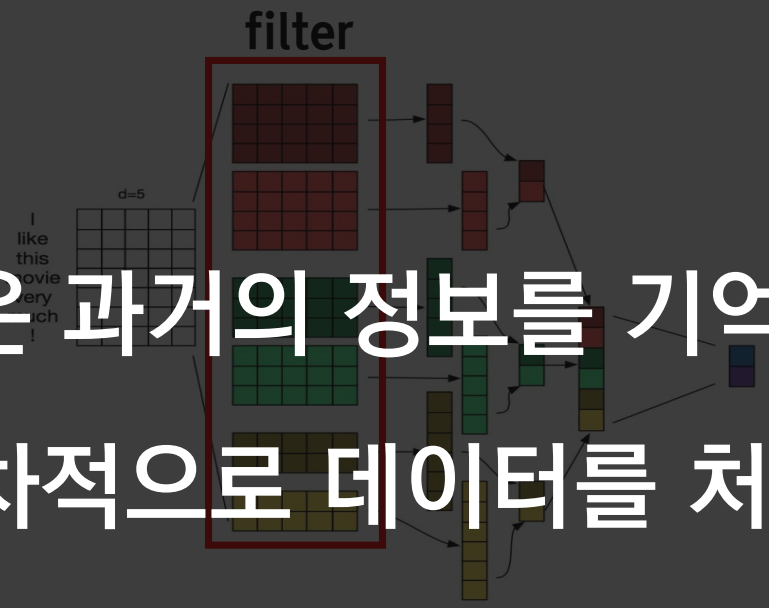
CNN의 필터는 크기가 커질수록 연산량이 기하급수적으로 늘어나 일정 크기 이상 확장할 수 없지만

자연어는 먼 거리의 단어 또는 다른 문장의 단어들이 의미 파악에 중요함

2 RNN 모델

- CNN의 한계

2. 먼 거리의 값과 관계 파악 불가



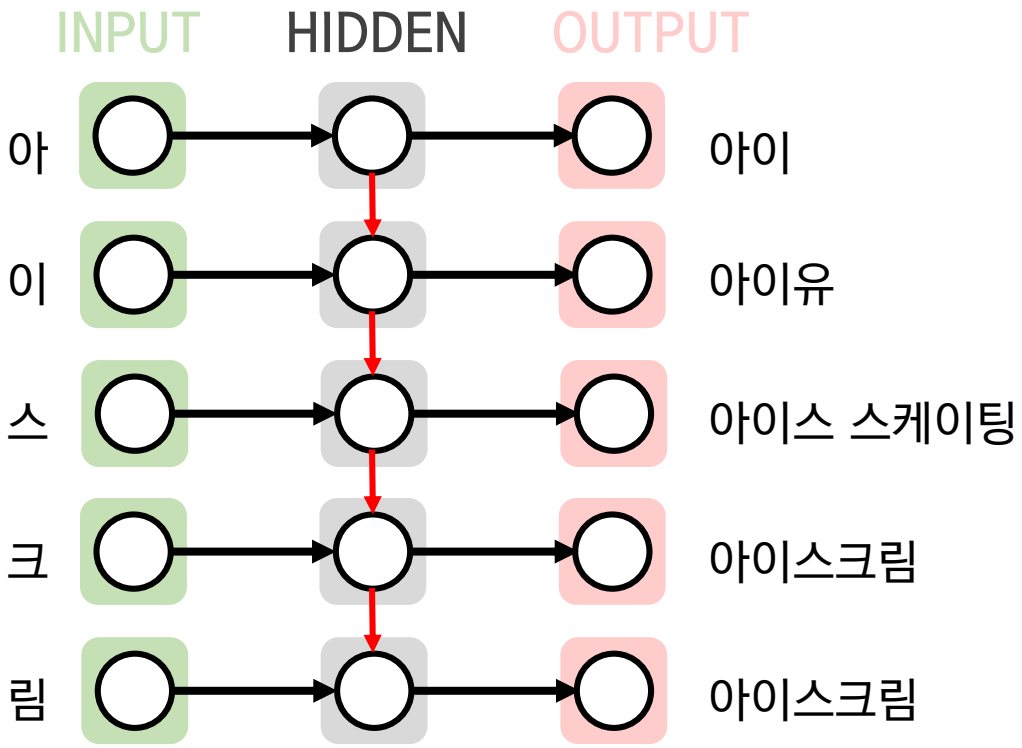
RNN은 과거의 정보를 기억하면서
순차적으로 데이터를 처리함

CNN의 필터는 일정 크기 이상 확장할 수 없지만
자연어는 먼 거리의 단어 또는 다른 문장의 단어들이 의미 파악에 중요함

- 모델 구조

순환신경망 (Recurrent Neural Network)

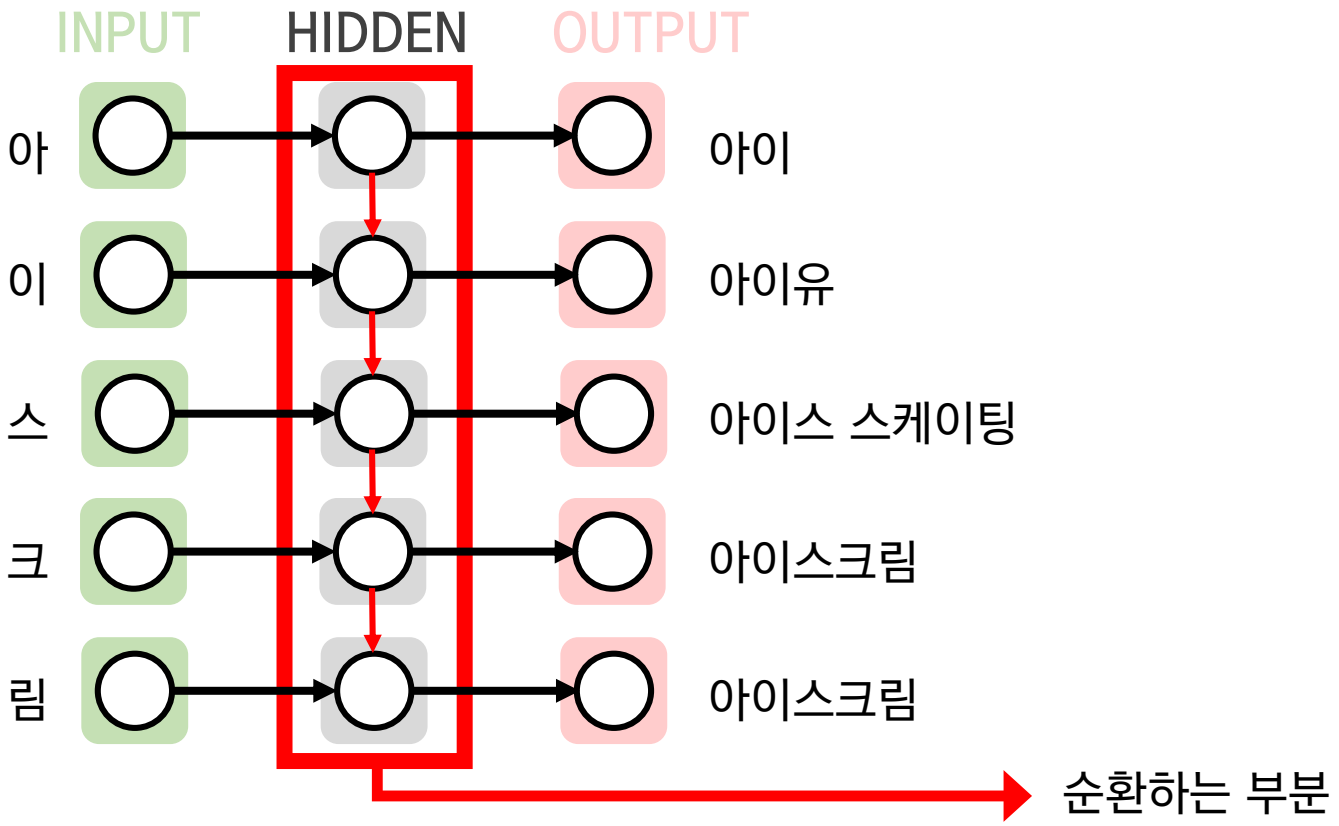
ex. 검색어 자동완성



- 모델 구조

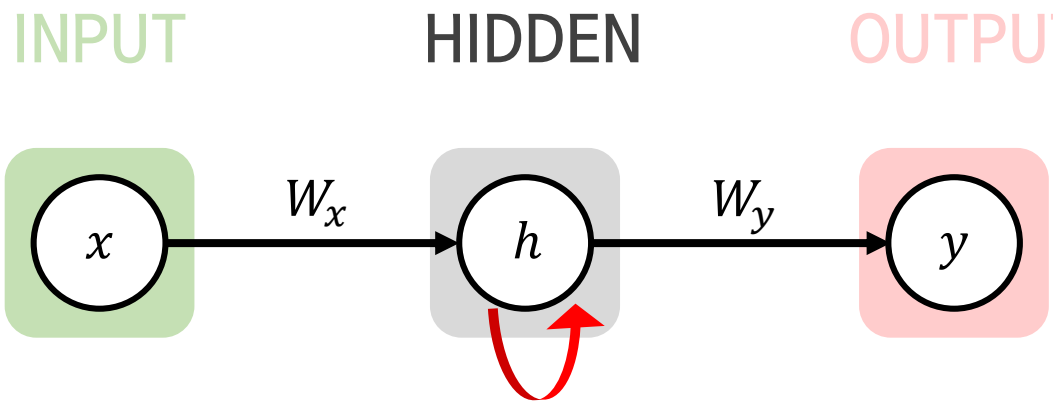
순환신경망 (Recurrent Neural Network)

ex. 검색어 자동완성



2 RNN 모델

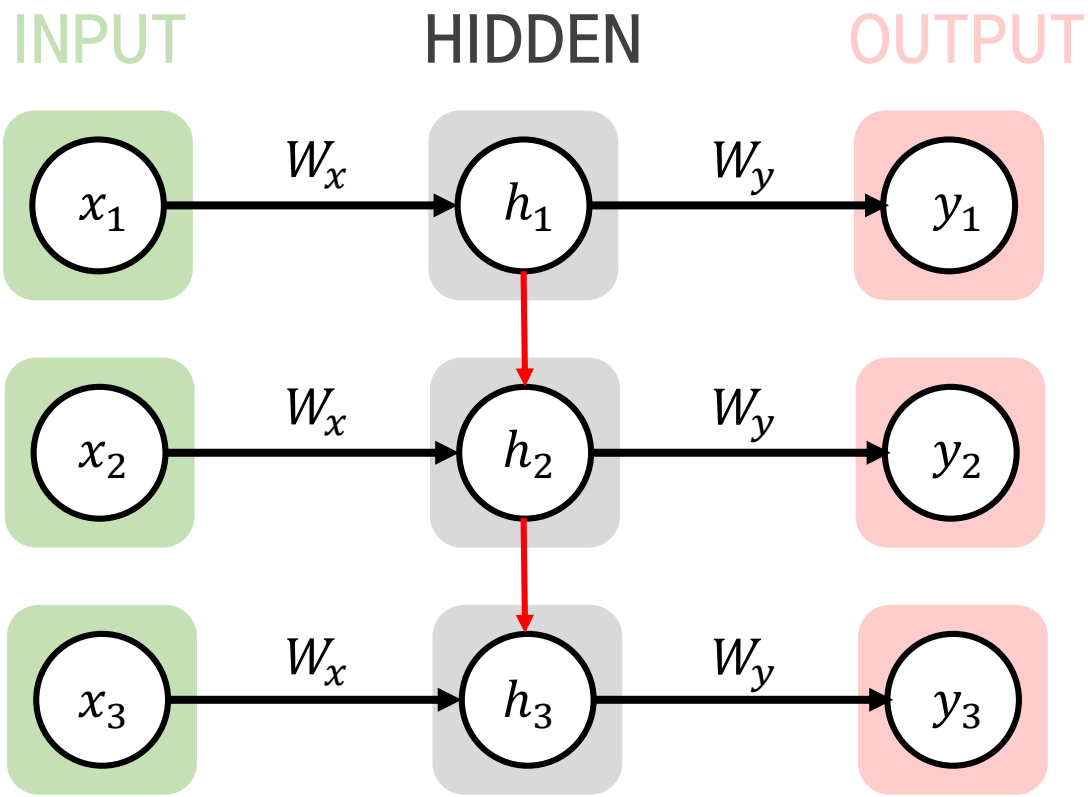
● 모델 구조



전시점의 은닉층 출력 값이 다시 입력 값으로 작용하는 모델

2 RNN 모델

● 모델 구조

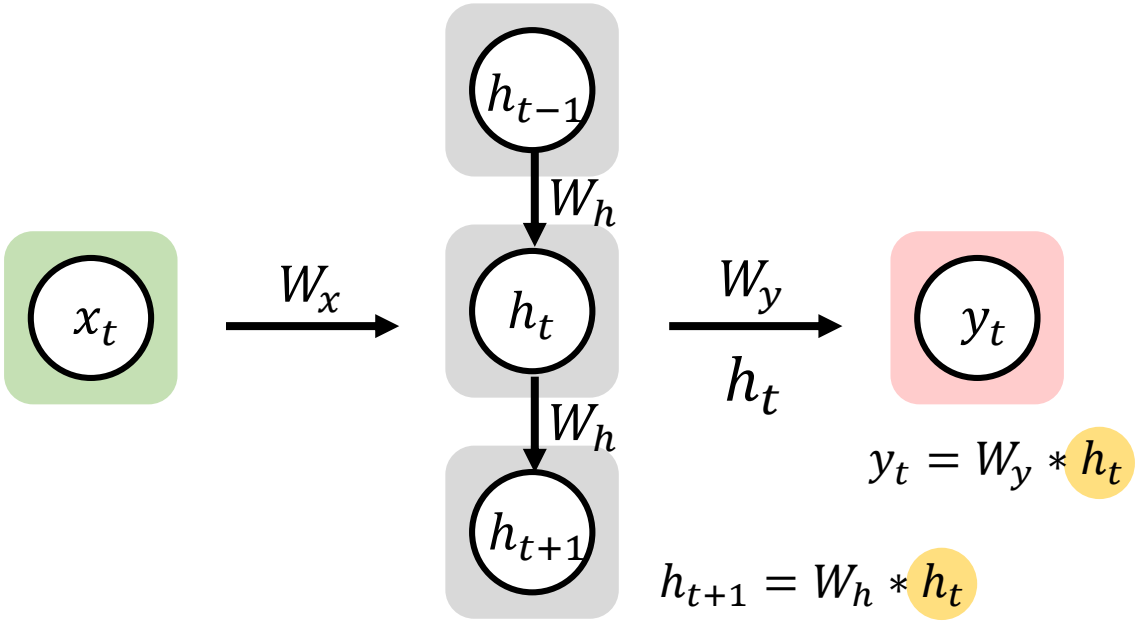


Time step t마다 입력값과 출력값을 따로 가져서

입력-출력 구조를 나타낼 수 있다.

2 RNN 모델

● 은닉층

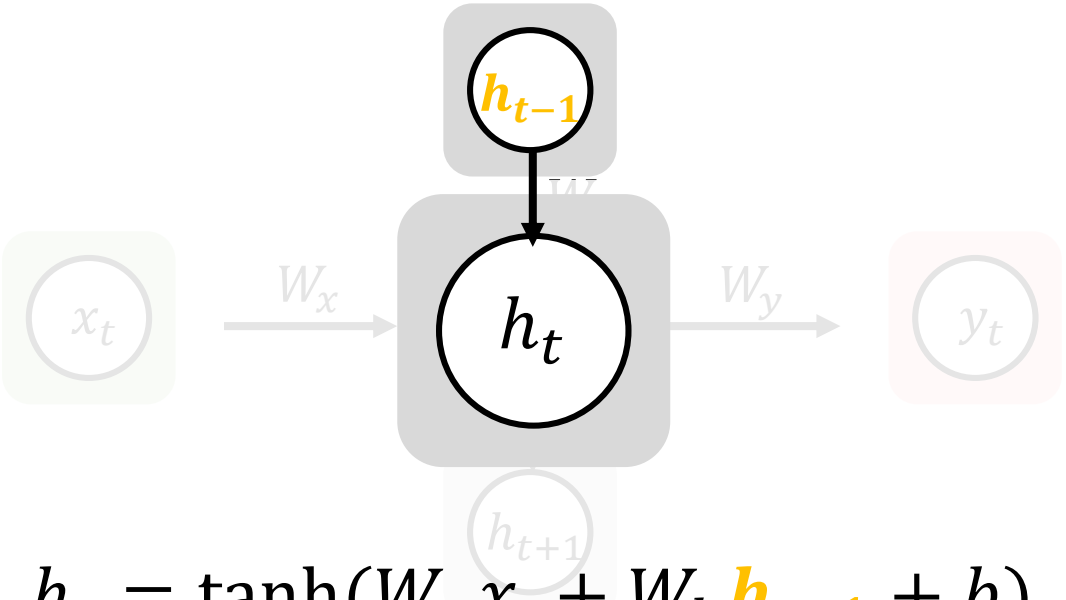


이전 시점의 은닉층의 출력값을 입력값으로 사용함으로써

이전 시점까지의 정보를 사용

2 RNN 모델

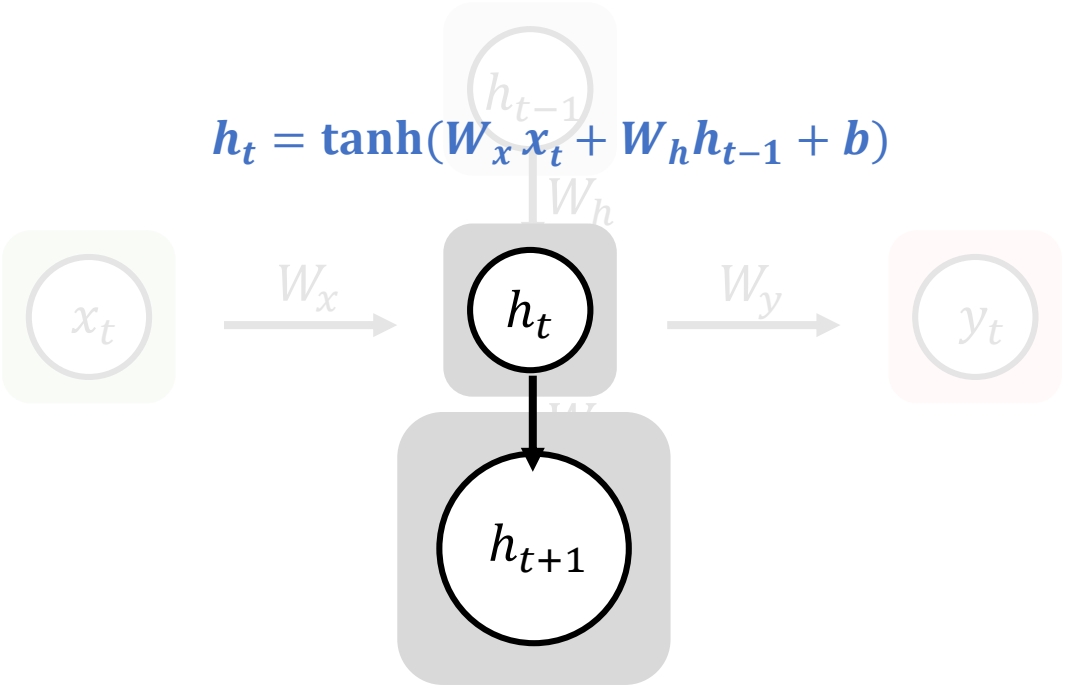
● 은닉층



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

● 은닉층

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

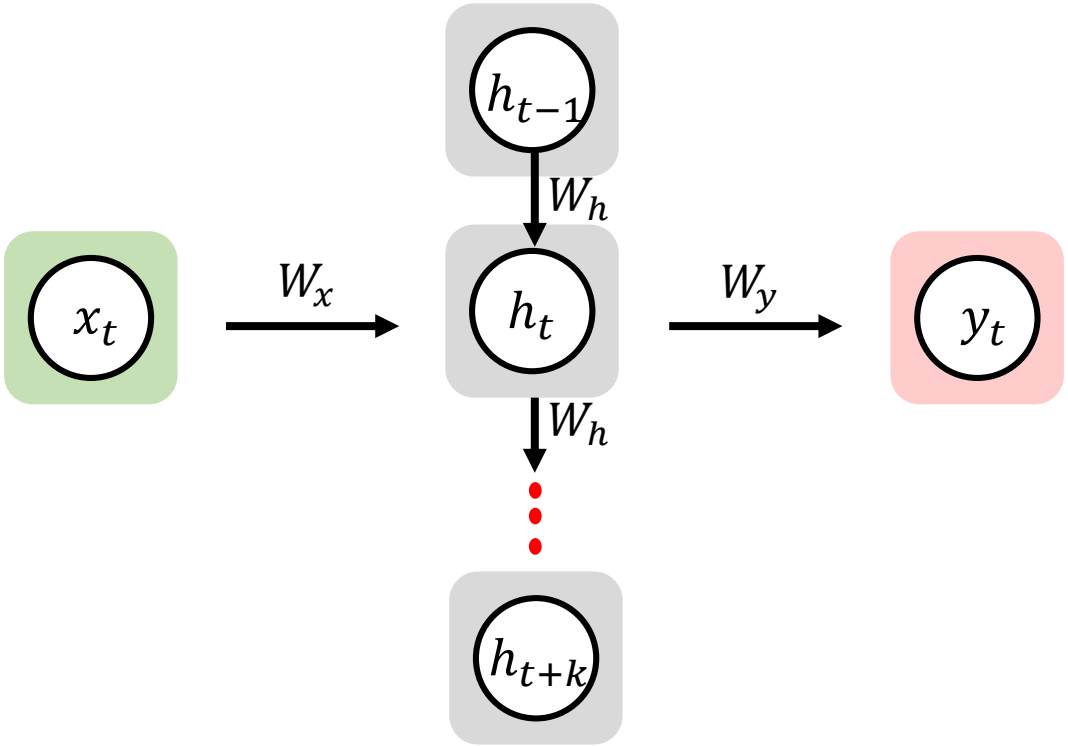


$$h_{t+1} = \tanh(W_x x_{t+1} + W_h \boldsymbol{h_t} + b)$$

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

2 RNN 모델

● 은닉층



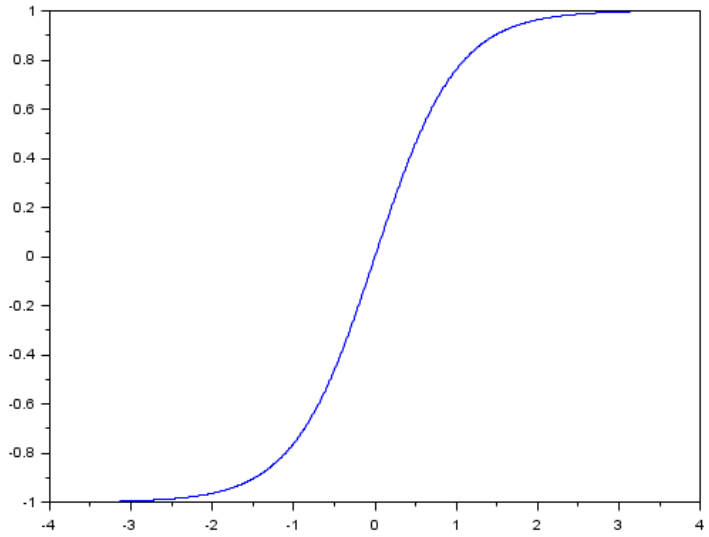
$$\tanh(\ldots\tanh(W_x x_{t+1} + W_h \tanh(W_x\, x_t + W_h \tanh(W_x\, x_{t-1} + W_h h_{t-2} + b) + b) + b)) \ldots + b$$

은닉층의 값이 연쇄적으로 이어짐

➡ 시점이 길어질수록 먼 시점 정보 희미해지는 문제 발생

● 활성화 함수

Tanh (Hyperbolic Tangent)



Time step의 존재는 모델이 기울기 소실 문제에 취약하게 함

➡ Sigmoid의 변형인 Tanh를 활성화함수로 사용

● 활성화 함수

Tanh (Hyperbolic Tangent)

Sigmoid

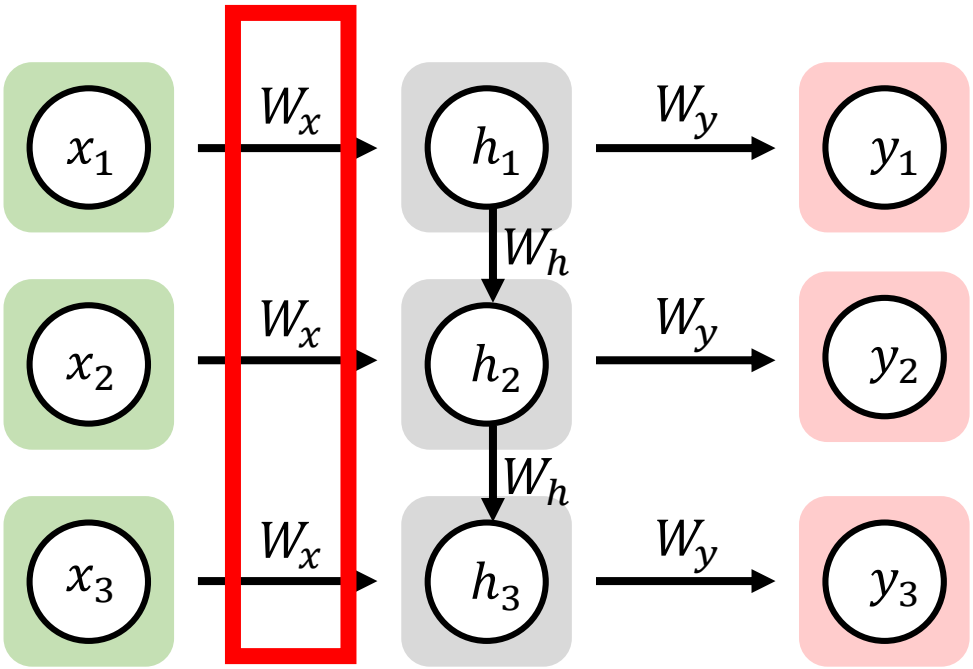
Tanh

Sigmoid 함수의 변형

미분 최대값이 1로 기울기 소실 문제가 완화됨

● 역전파

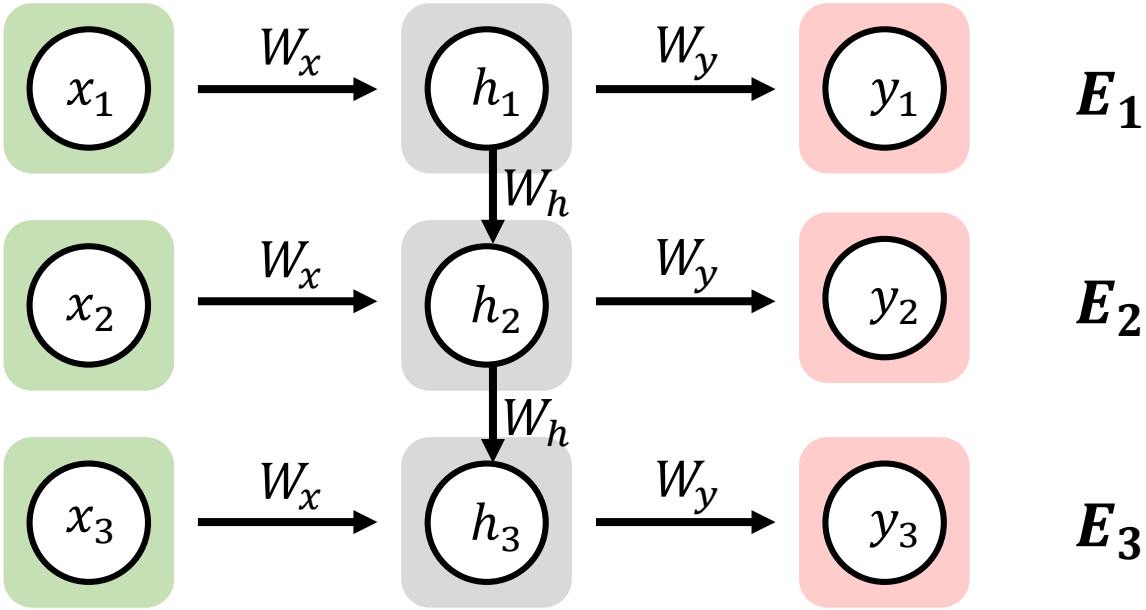
BPTT(Back Propagation Through Time)



RNN은 동일한 가중치를 다른 시점 데이터들이 공유하는 구조

● 역전파

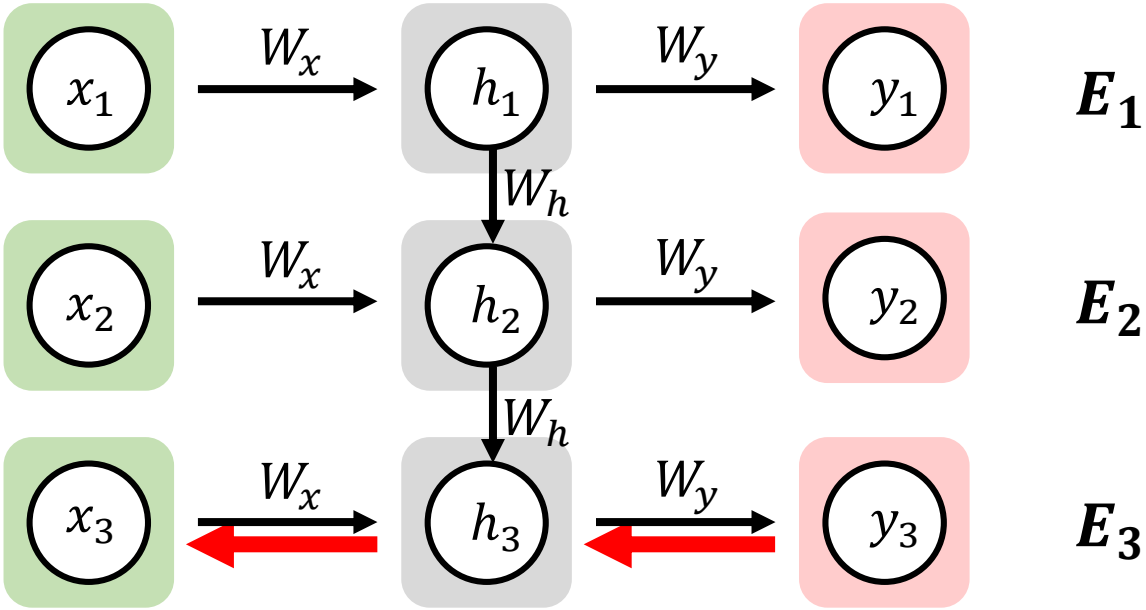
BPTT(Back Propagation Through Time)



매 시점마다 손실함수가 계산되어 한번의 순전파에서 여러 개의 손실함수 값이 나옴

● 역전파

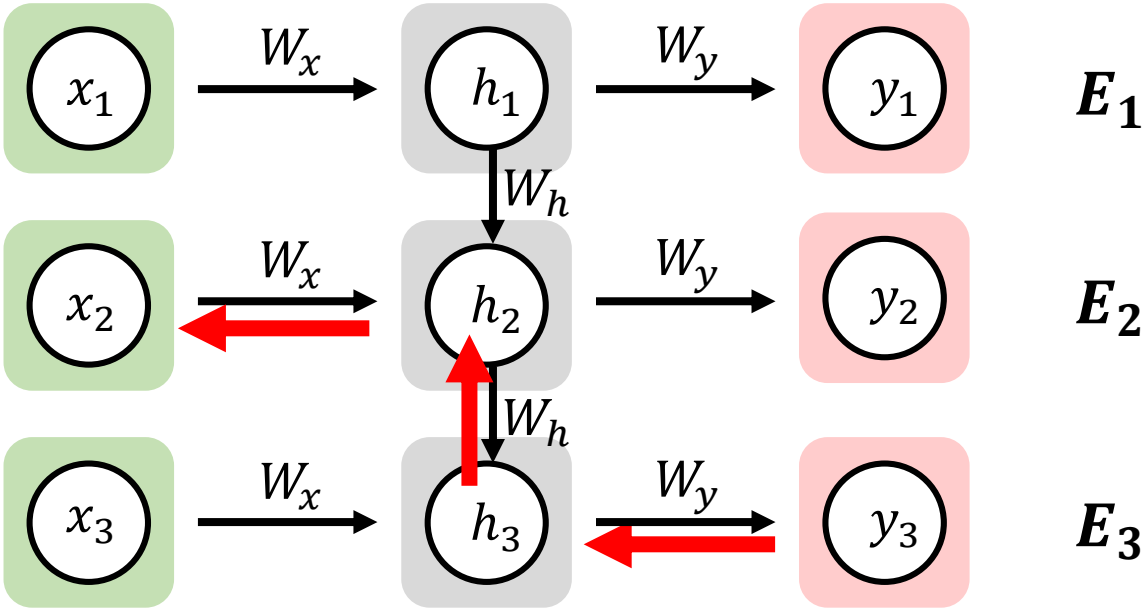
BPTT(Back Propagation Through Time)



매 시점마다 W_x에 대해 역전파가 이루어짐

● 역전파

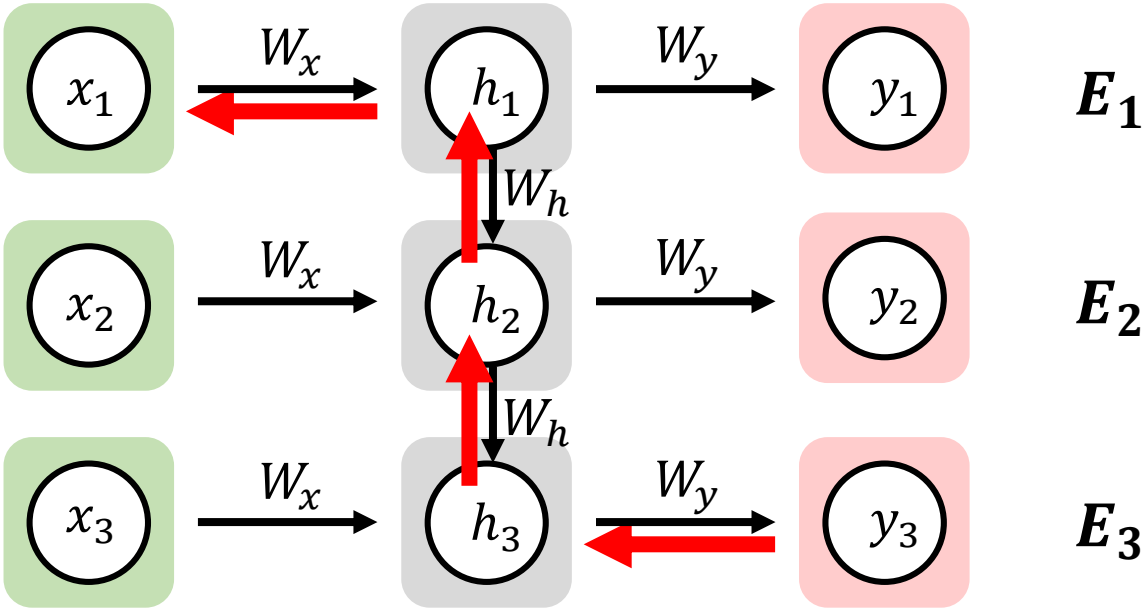
BPTT(Back Propagation Through Time)



매 시점마다 W_x 에 대해 역전파가 이루어짐

● 역전파

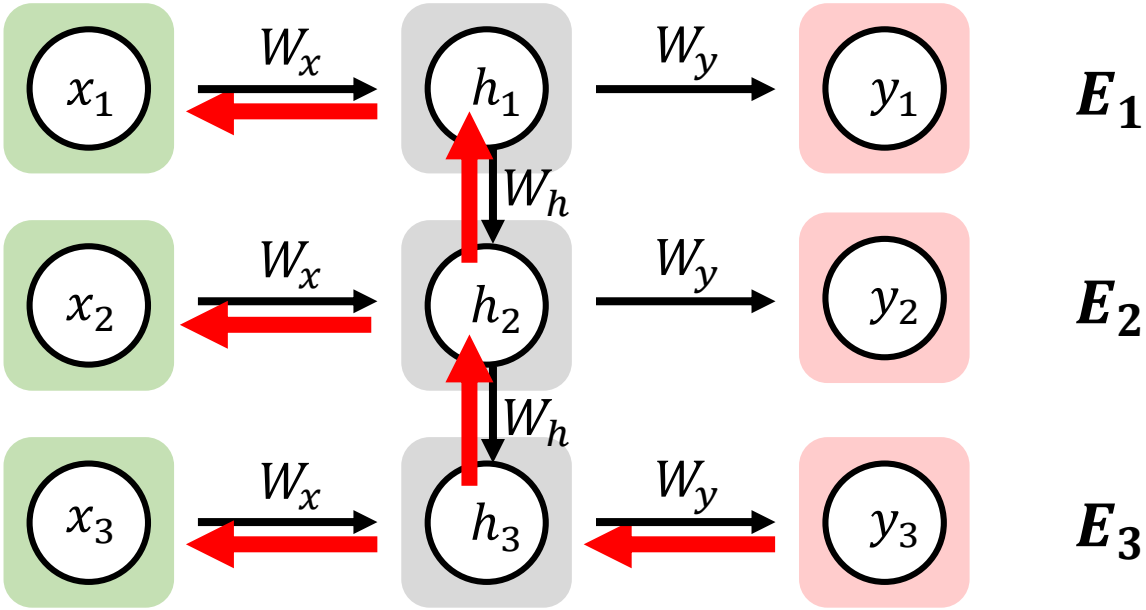
BPTT(Back Propagation Through Time)



매 시점마다 W_x에 대해 역전파가 이루어짐

● 역전파

BPTT(Back Propagation Through Time)

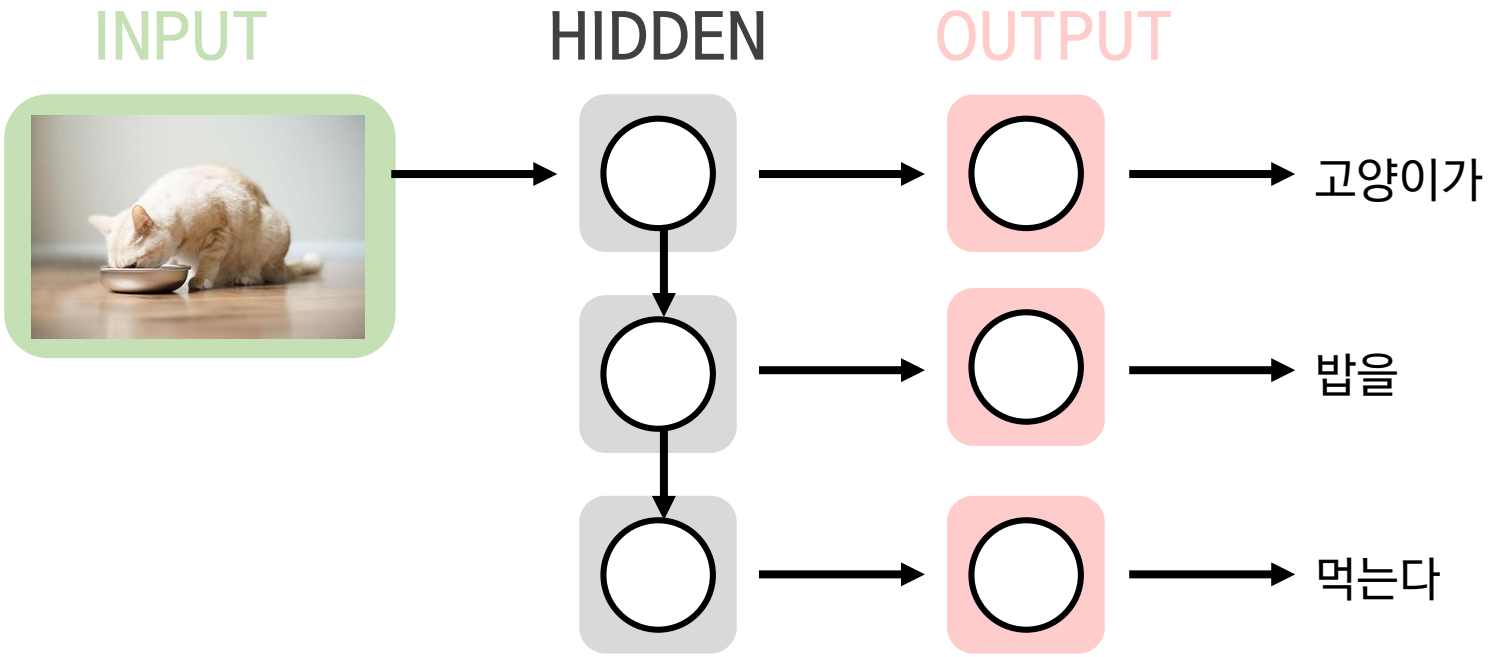


같은 방식으로 모든 시점의 역전파 값을 더하여 W_x 에 대한 역전파 값으로 사용

- RNN의 활용구조

One to Many

ex. Image captioning

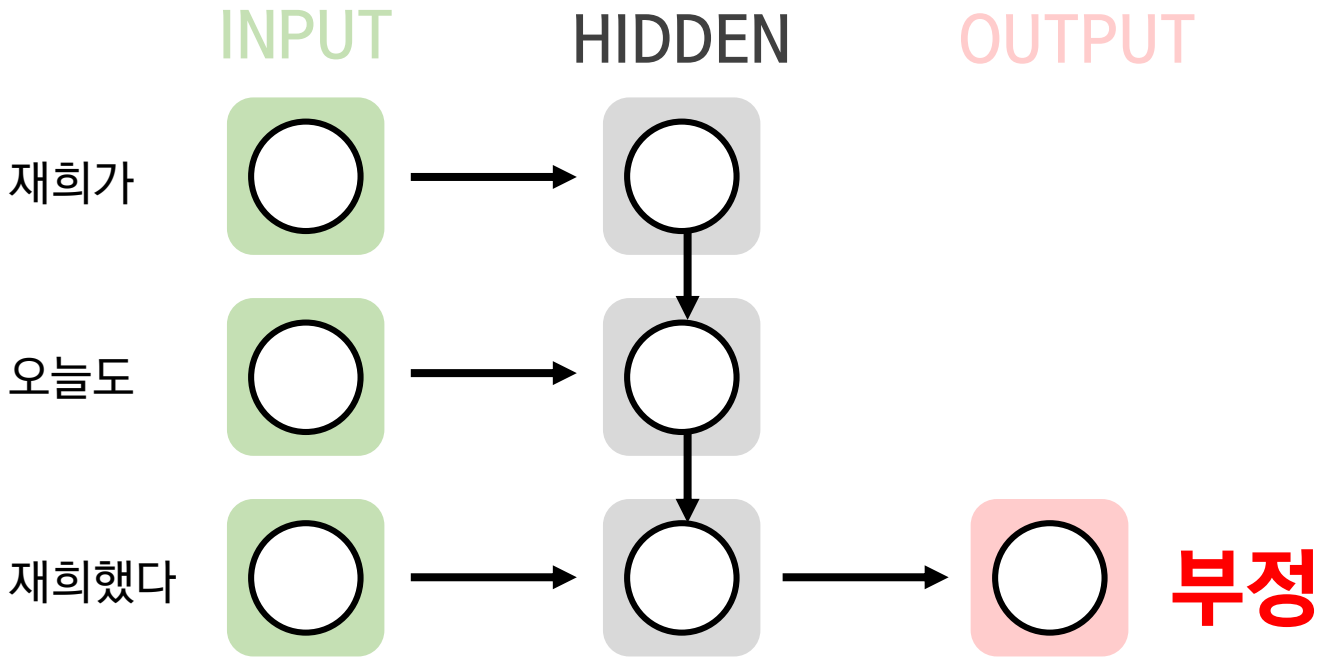


입력 값은 시계열 변수가 아니지만 출력값은 시계열 변수인 경우

- RNN의 활용구조

Many to One

ex. 감성분석

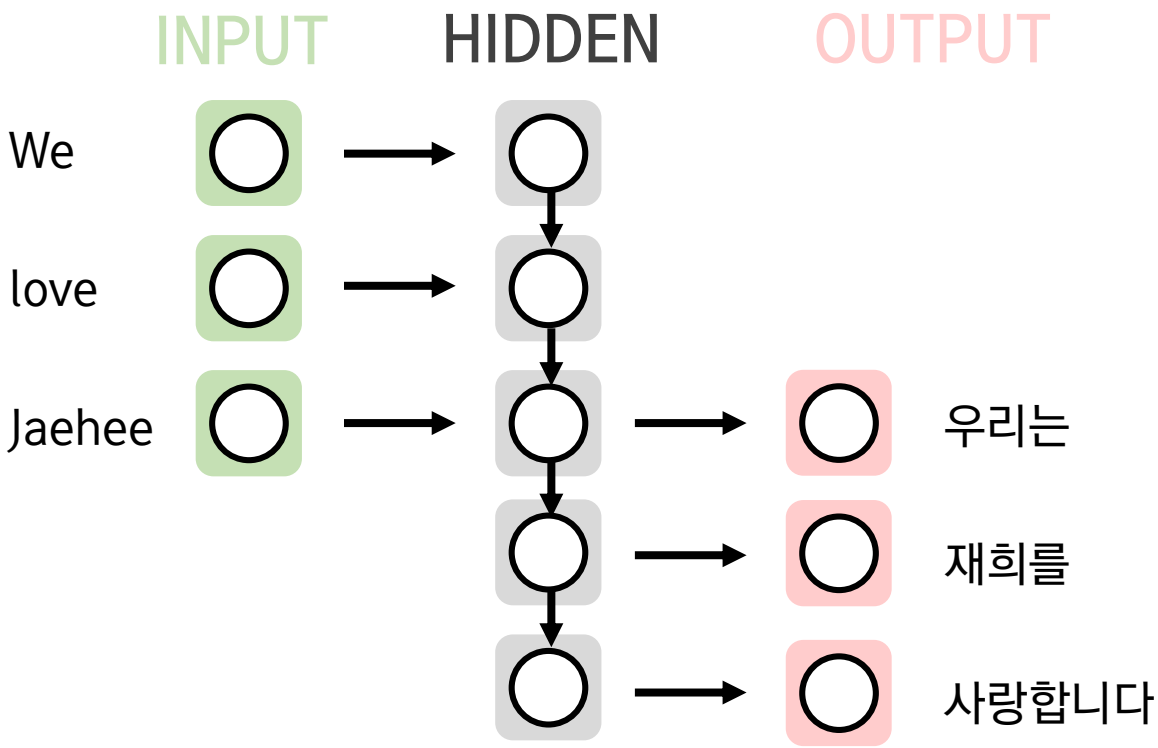


입력 값은 시계열 변수이지만 출력값은 시계열 변수가 아닌 경우

- RNN의 활용구조

Many to Many

ex. 번역

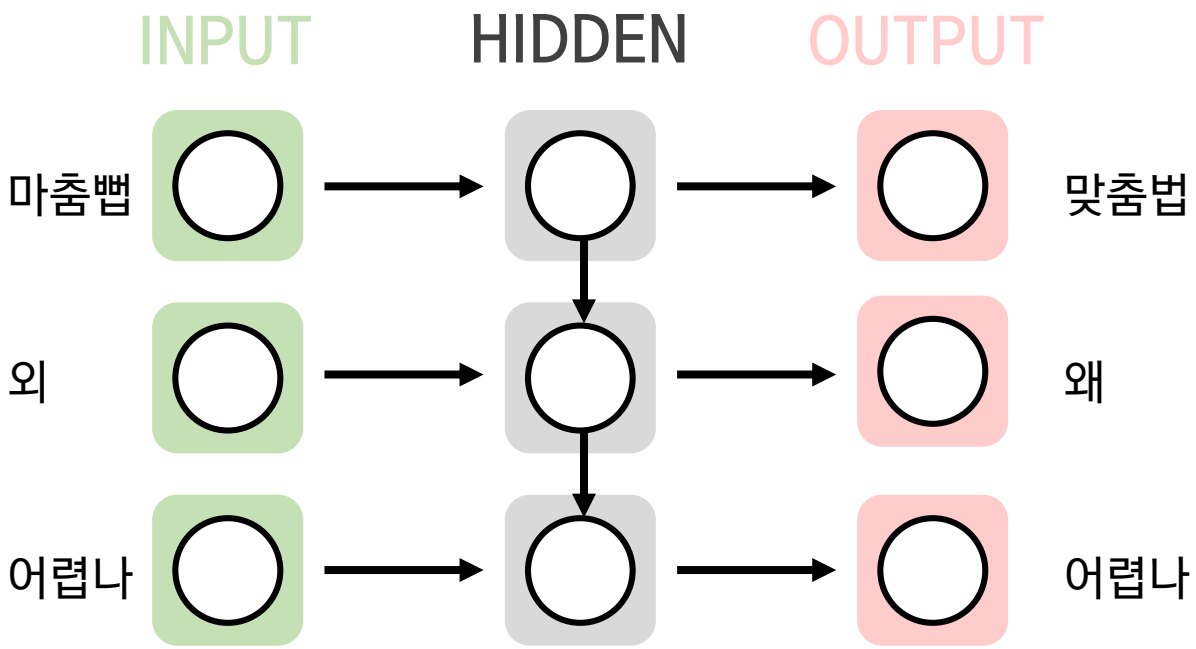


입력 값과 출력 값 모두 시계열 변수인 경우

- RNN의 활용구조

Many to Many

ex. 오타 수정

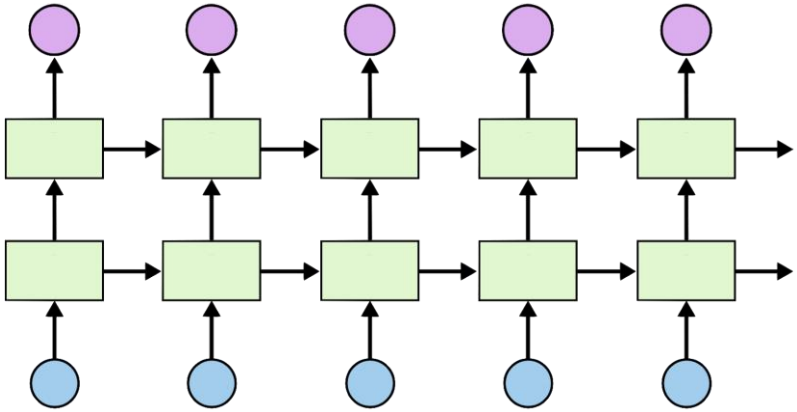


입력 값과 출력 값 모두 시계열 변수인 경우

- RNN의 활용구조

Stacked RNN

: rnn을 여러 층 겹쳐 쌓음



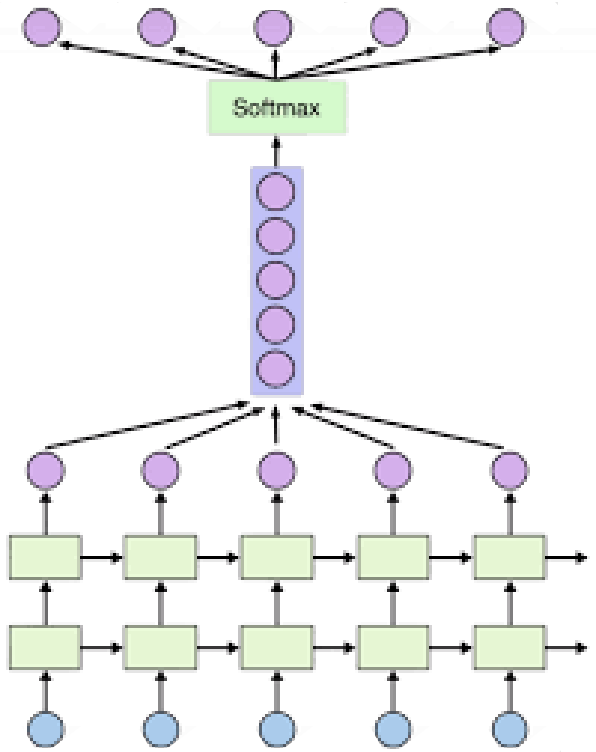
매 시점마다 정보가 복잡하게 얽히고,

이전시점 정보도 더 복잡하게 현재시점에 반영됨

- RNN의 활용구조

Stacked RNN

: rnn을 여러 층 겹쳐 쌓음



문제에 따라서 마지막 출력층을 결정함

(ex. Softmax를 추가하여 분류모델로 만듦)

3

LSTM

3 LSTM

● RNN의 한계

Activation 함수 특성

기울기 0~1

+

RNN 모델 구조

활성화 함수가 쌓여

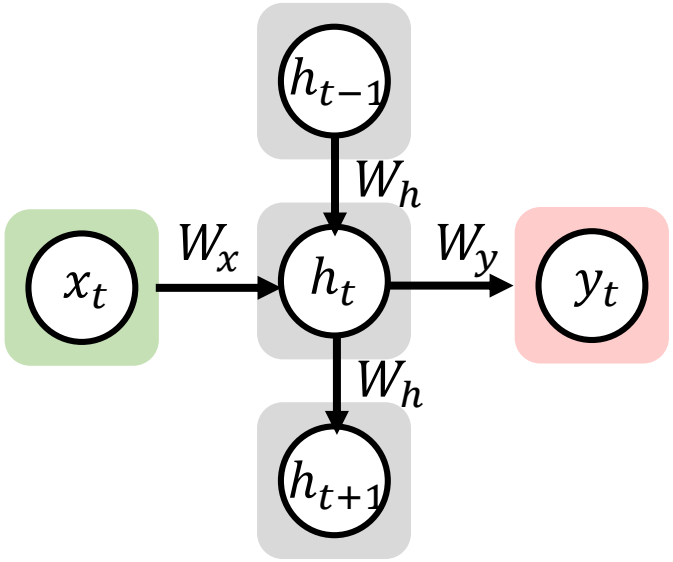
Vanishing Gradient 문제 심각

∴ 멀리 있는 시점의 Weight가 제대로 업데이트 되지 않음

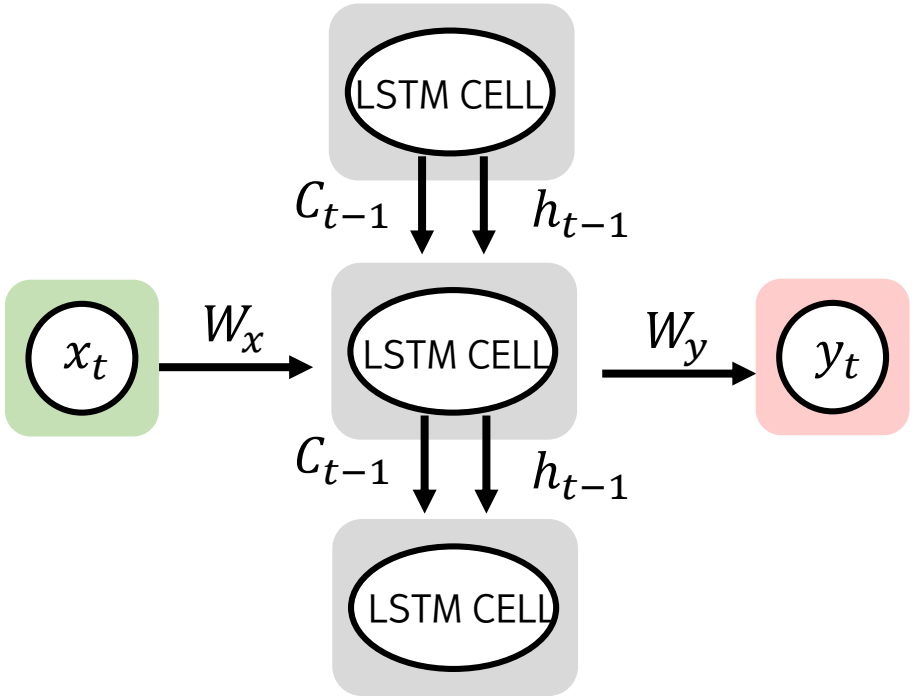
➡ 시점이 길어질수록 RNN 모델의 성능은 기하급수적으로 떨어짐

● 장단기 기억모델이란?

• RNN

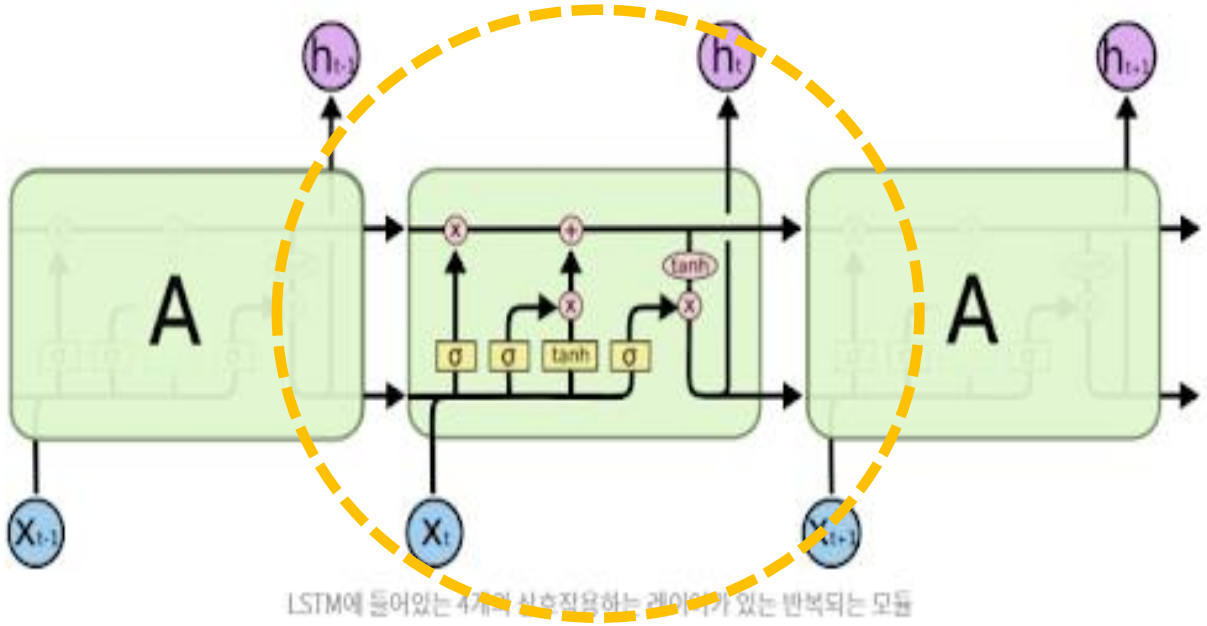


• LSTM



장기기억과 단기기억에 해당하는 변수를 따로 만들자

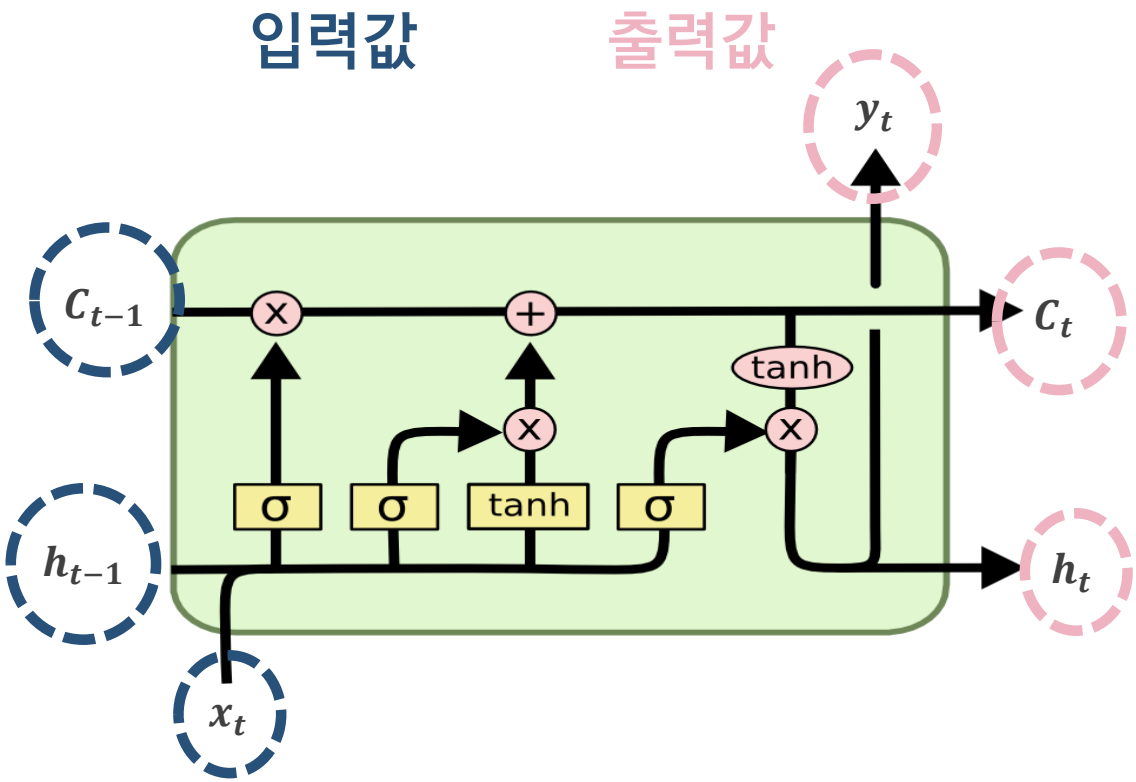
● 장단기 기억모델이란?



LSTM에 들어있는 4개의 신호작용하는 레이어가 있는 반복되는 모듈

LSTM의 CELL은 구성요소가 여러개

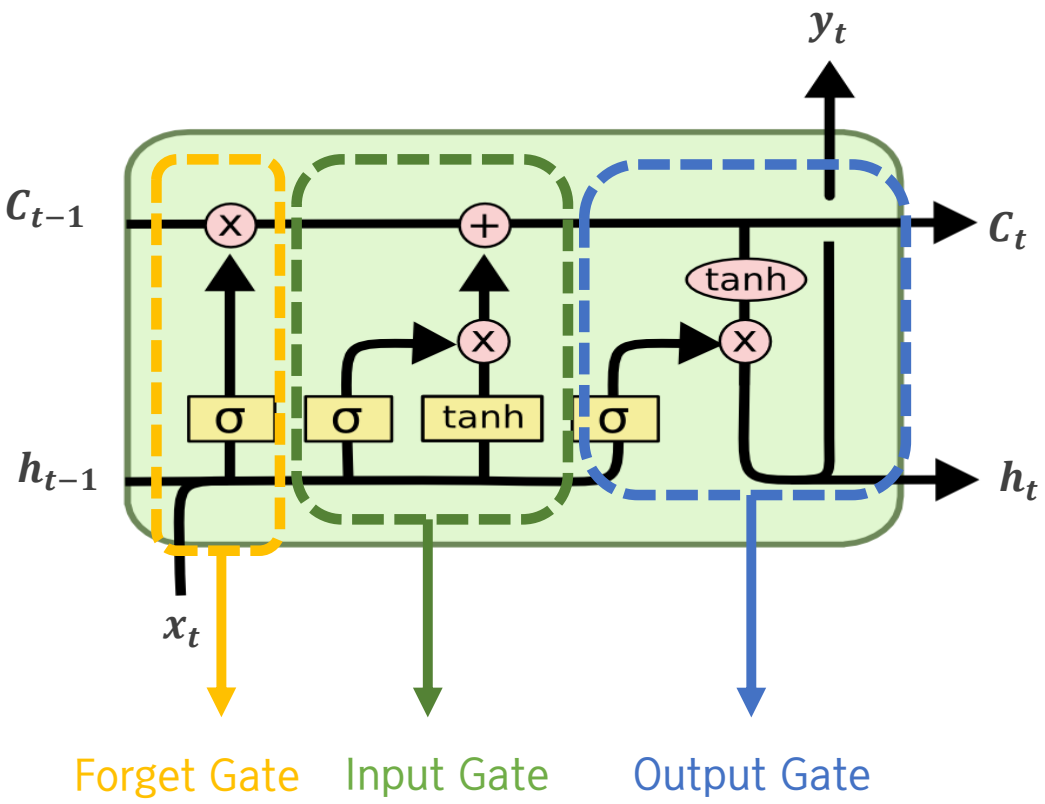
● 장단기 기억모델이란?



● 장단기 기억모델이란?

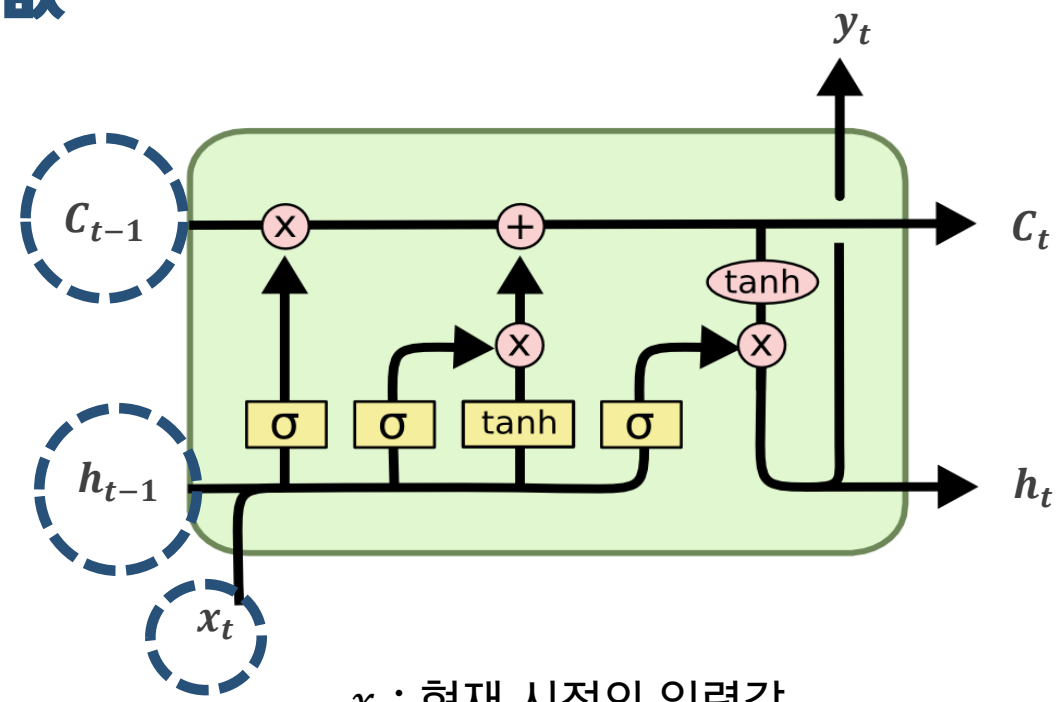
Gate

: 정보를 얼마나 통과시킬지를 결정하는 '문'



- 장단기 기억모델이란?

입력값



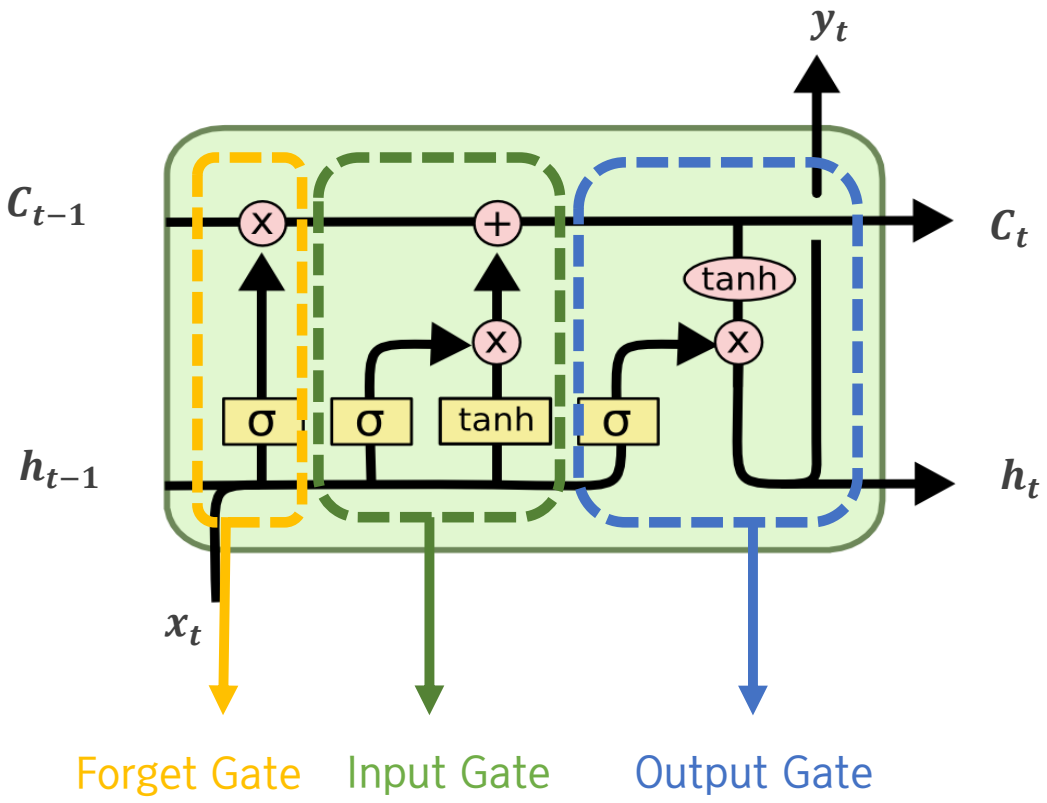
x_t: 현재 시점의 입력값

h_{t-1}: 이전 시점까지의 정보 (단기 기억)

C_{t-1} : 장기적인 정보 (장기 기억)

- 장단기 기억모델이란?

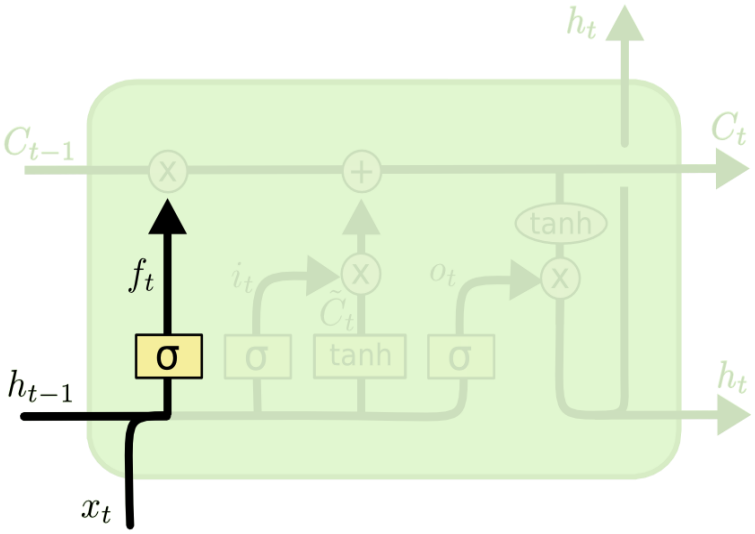
Gate



- 장단기 기억모델이란?

Gate – Forget Gate

: 이전 시점까지의 장기 기억(C_{t-1})을 얼마나 잊을 지 결정하는 문



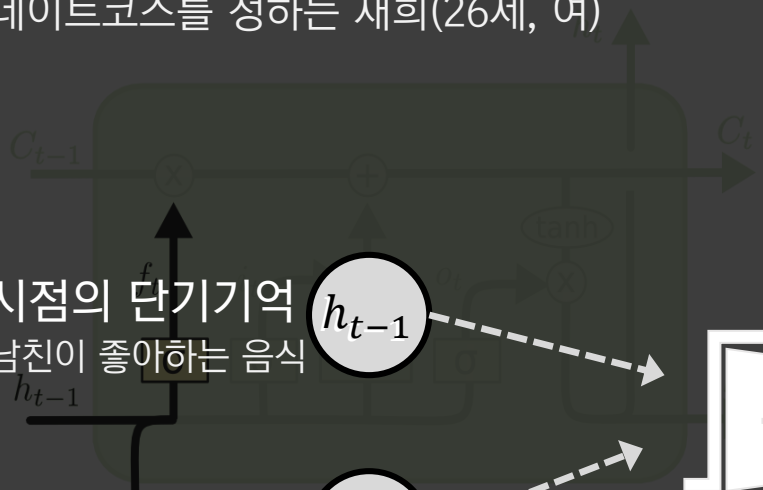
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

● 장단기 기억모델이란?

Gate – Forget Gate

: 이전 시점까지의 장기 기억(C_{t-1})을 얼마나 잊을 지 결정하는 문

Ex : 데이트코스를 정하는 재희(26세, 여)



이전 시점의 단기기억 h_{t-1}

ex) 전 남친이 좋아하는 음식

현재 시점의 정보 x_t

ex) 현 남친이 좋아하는 음식

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

f_t

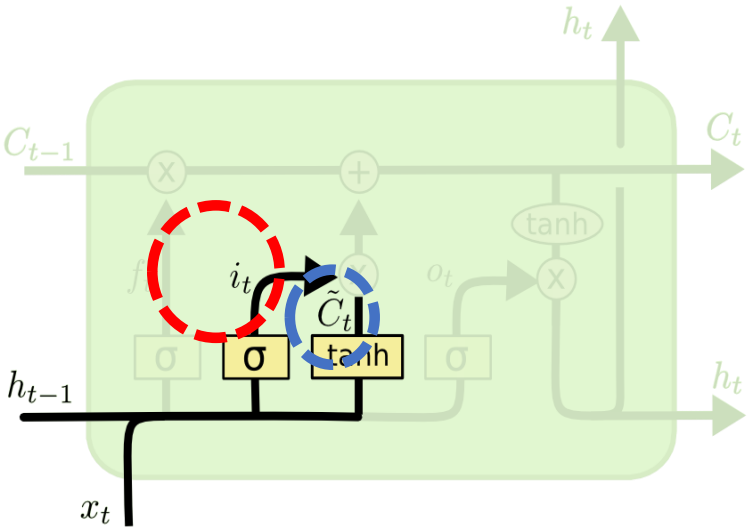
이전시점의 장기기억을 얼마나 잊을 지 결정

둘이 좋아하는 음식이 많이 다르네
전남친 최애 음식은 빨리 잊자

● 장단기 기억모델이란?

Gate – Input Gate

: 이전 시점 정보의 일부를 잃은 C_t에 현재 시점의 정보를 입력할 지 결정



범위: 0~1
정보가 담기는 강도

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

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

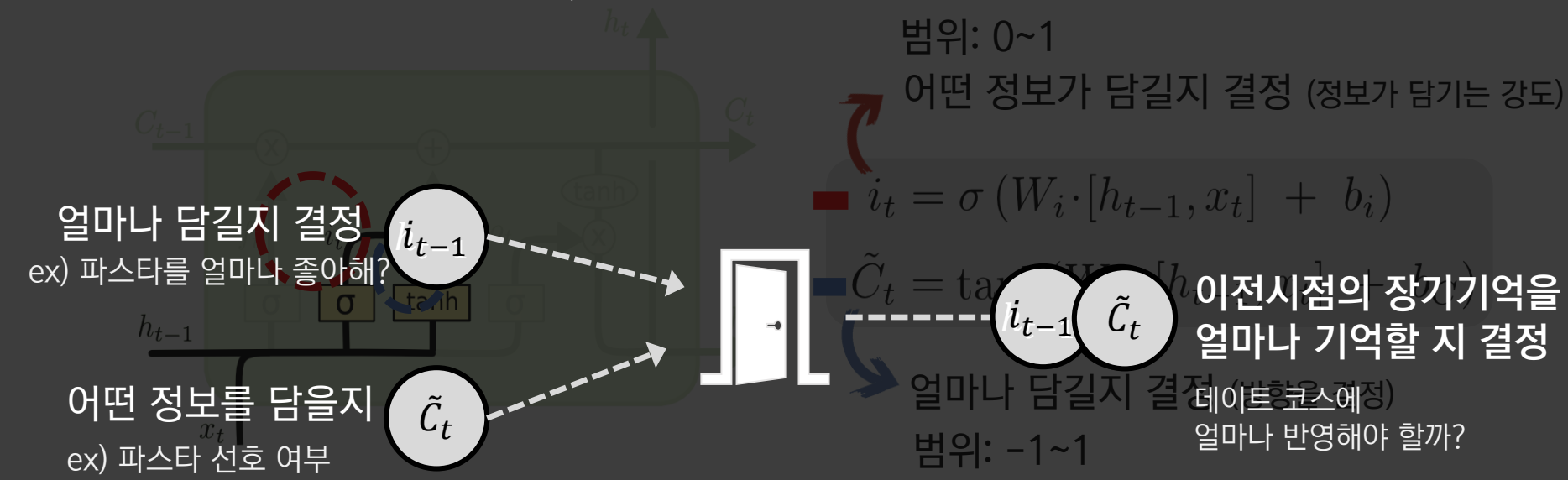
정보의 방향
범위: -1~1

● 장단기 기억모델이란?

Gate – Input Gate

: 이전 시점 정보의 일부를 잃은 C_t에 현재 시점의 정보를 입력할 지 결정

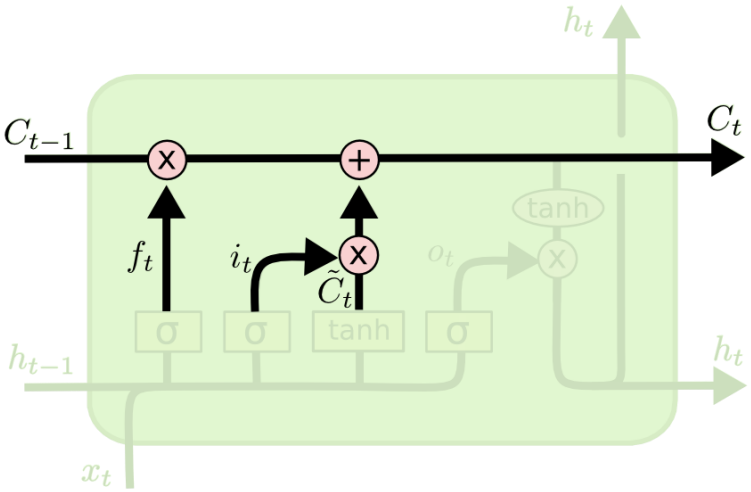
Ex : 데이트코스를 정하는 재희(26세, 여)



● 장단기 기억모델이란?

Gate – C_t

: Forget Gate만큼 정보를 잃고 Input Gate만큼 정보를 얻음



아다마르 곱

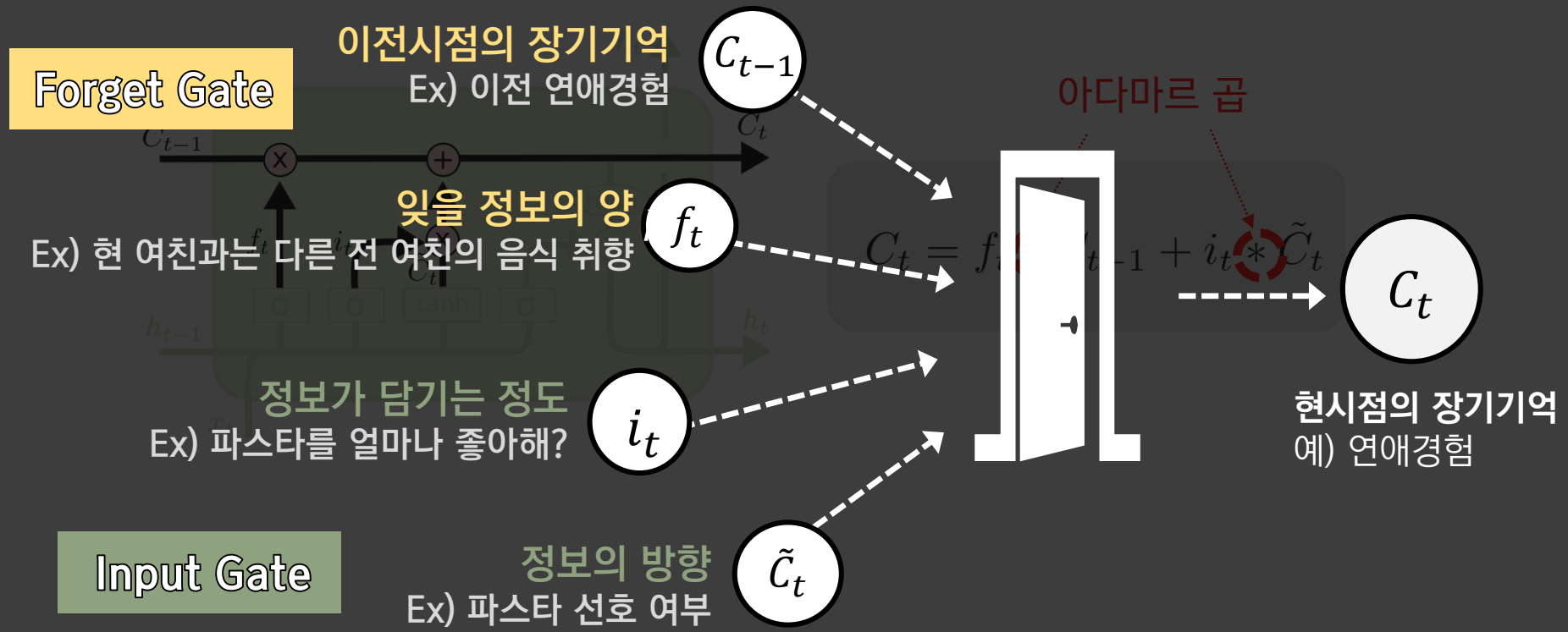
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

• 장단기 기억모델이란?

Gate – C_t

: Forget Gate만큼 정보를 잃고 Input Gate만큼 정보를 얻음

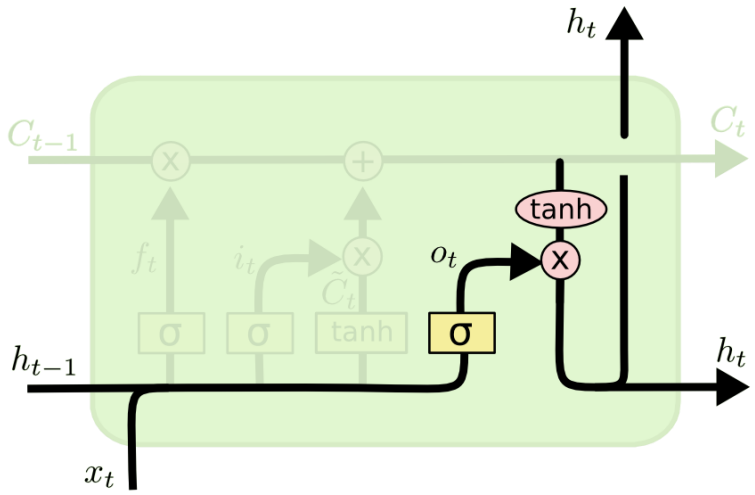
Ex : 데이트코스를 정하는 재희(26세, 여)



● 장단기 기억모델이란?

Gate – Output Gate

: 출력값을 산출하는 과정



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

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

이전 시점의 단기기억(h_{t-1})과
현재 시점의 정보(x_t)를 합침

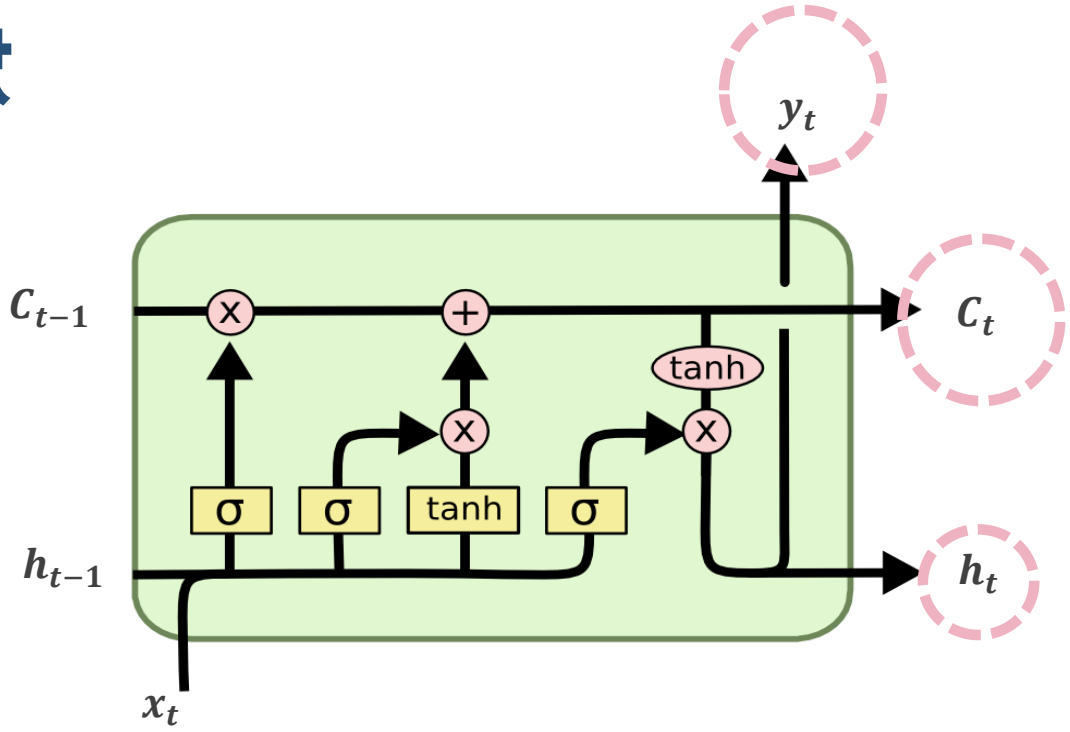
$$h_t = o_t * C_t$$

출력값

현재시점의 장기기억

● 장단기 기억모델이란?

출력값



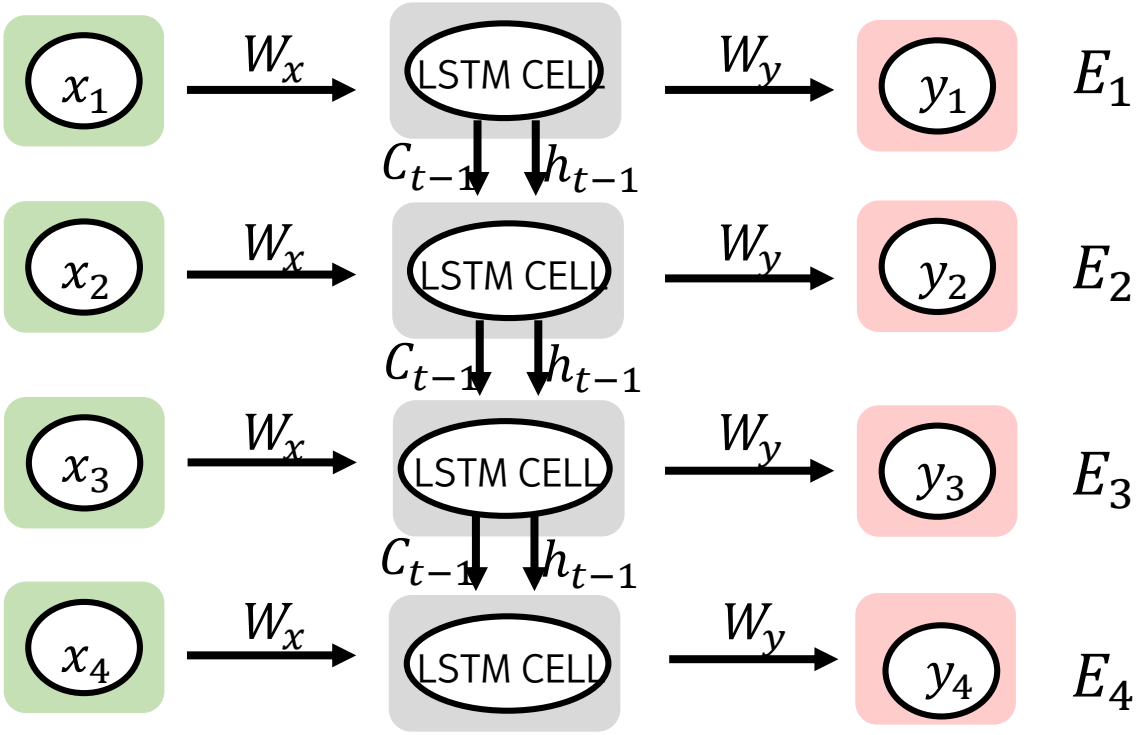
$$C_t$$

이전 시점의 장기기억 중 필요 없는 부분을 잊고

이번 시점에서의 입력값을 고려하여 새롭게 기억하게 된 정보가 있음

- LSTM 역전파

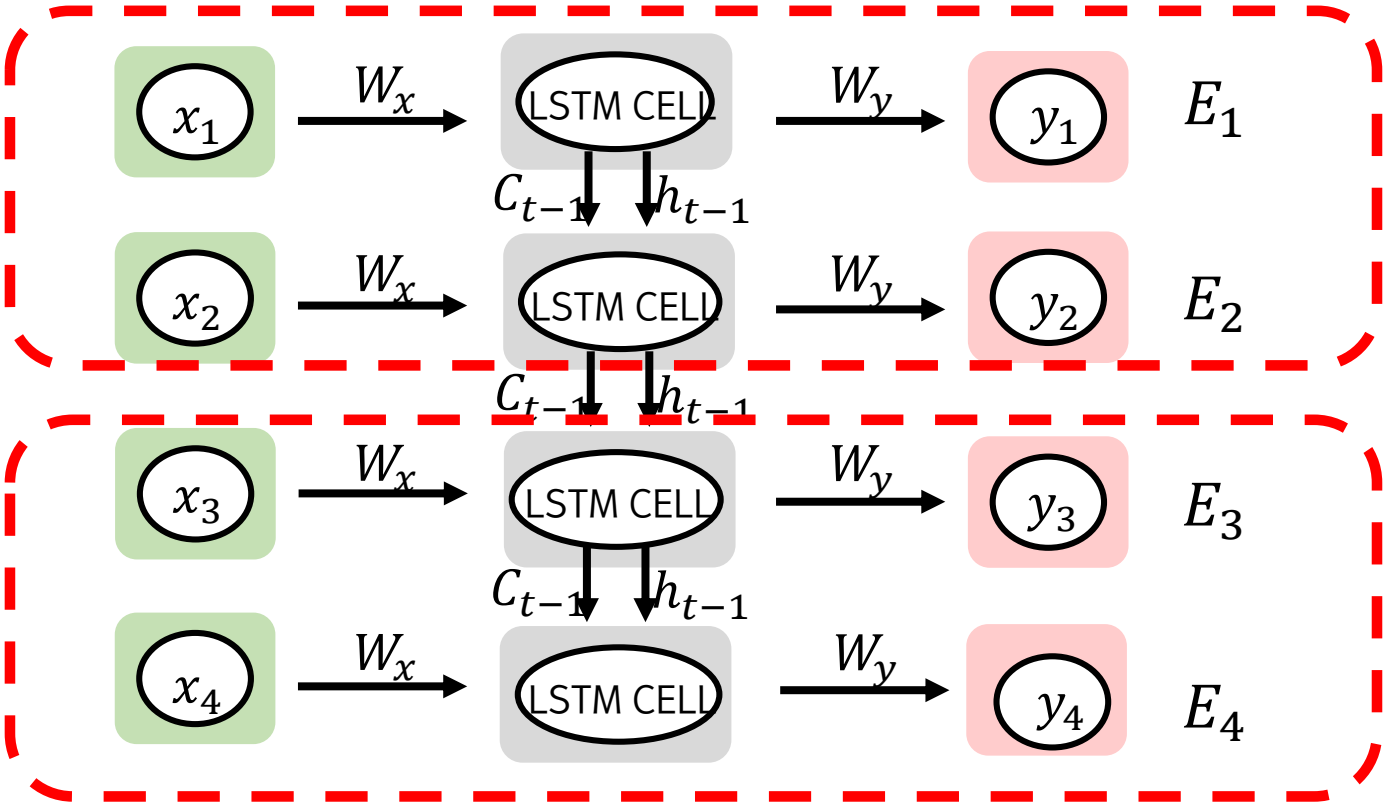
Truncated BPTT



특정 단위만큼의 시점을 고려해 역전파를 진행함

- LSTM 역전파

Truncated BPTT

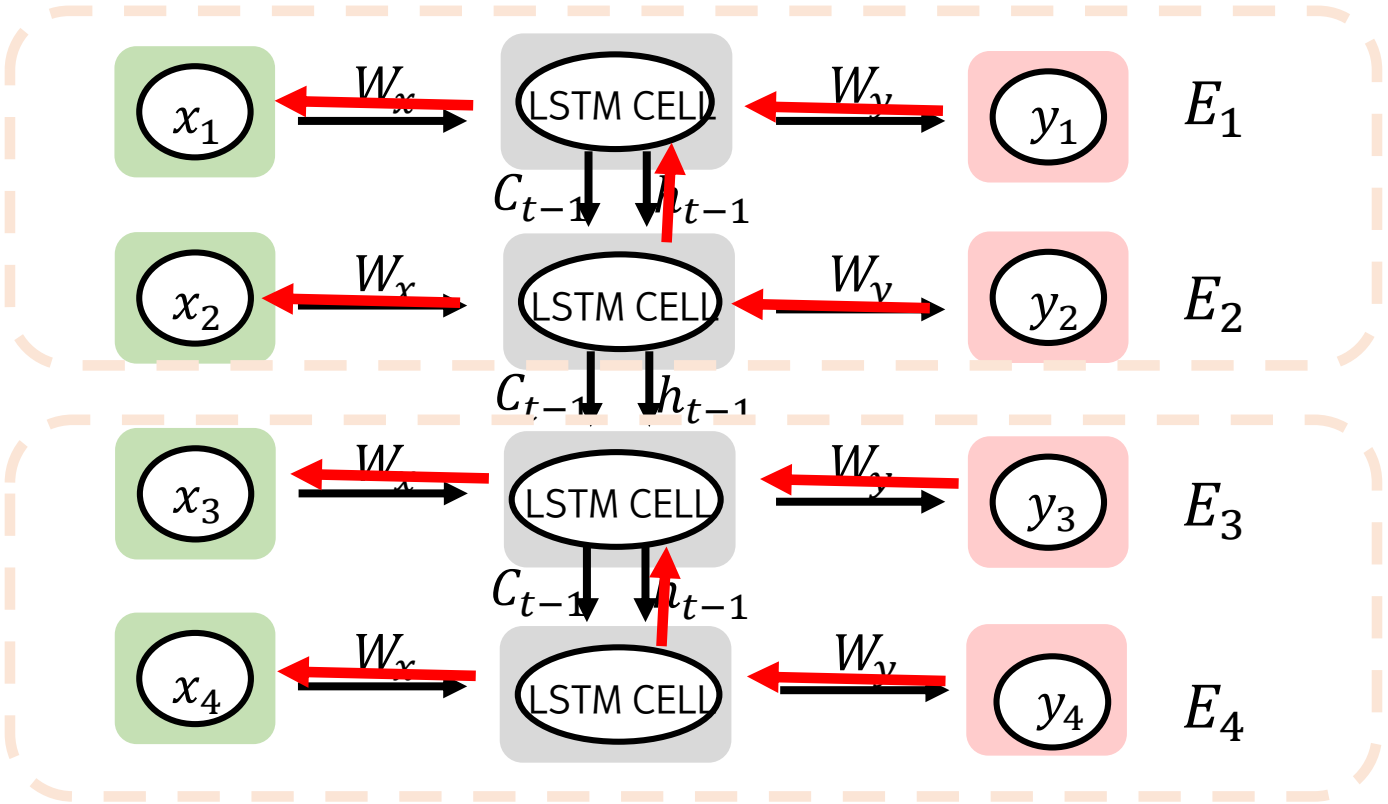


특정 단위만큼의 시점을 고려해 역전파를 진행함

Ex. 2시점

- LSTM 역전파

Truncated BPTT



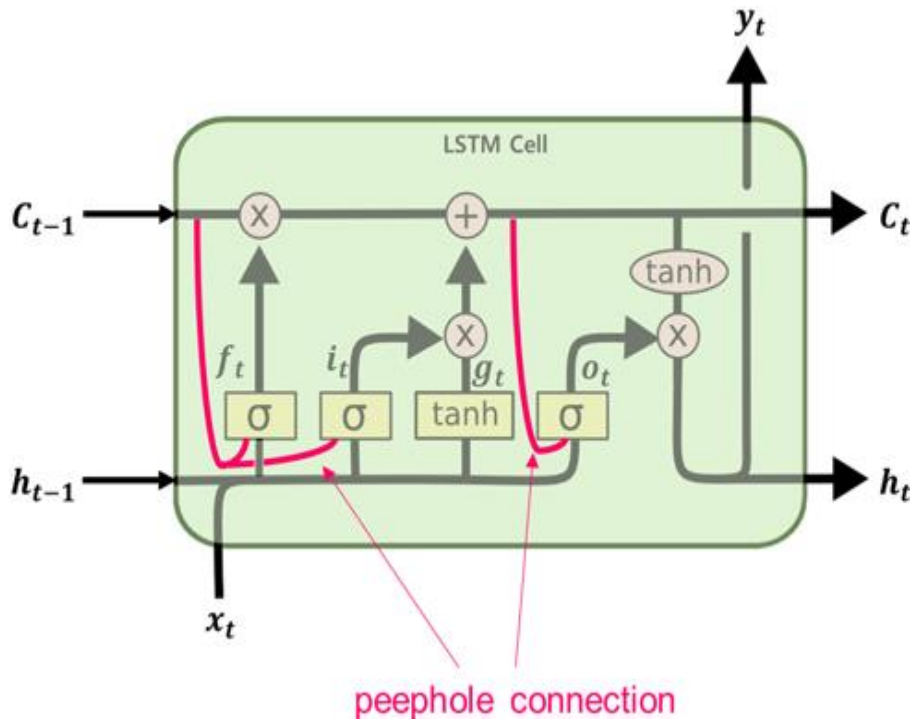
LSTM의 역전파는 끝까지 가지 않아도 됨

4

부록

4 LSTM의 변형 모델들

• LSTM with peep-hole connection



$$f_t = \sigma\left(w_{cf}^T \cdot c_{t-1} + w_{xf}^T \cdot x_t + w_{hf}^T \cdot h_{t-1} + b_f\right)$$

$$i_t = \sigma\left(w_{ci}^T \cdot c_{t-1} + w_{xi}^T \cdot x_t + w_{hi}^T \cdot h_{t-1} + b_i\right)$$

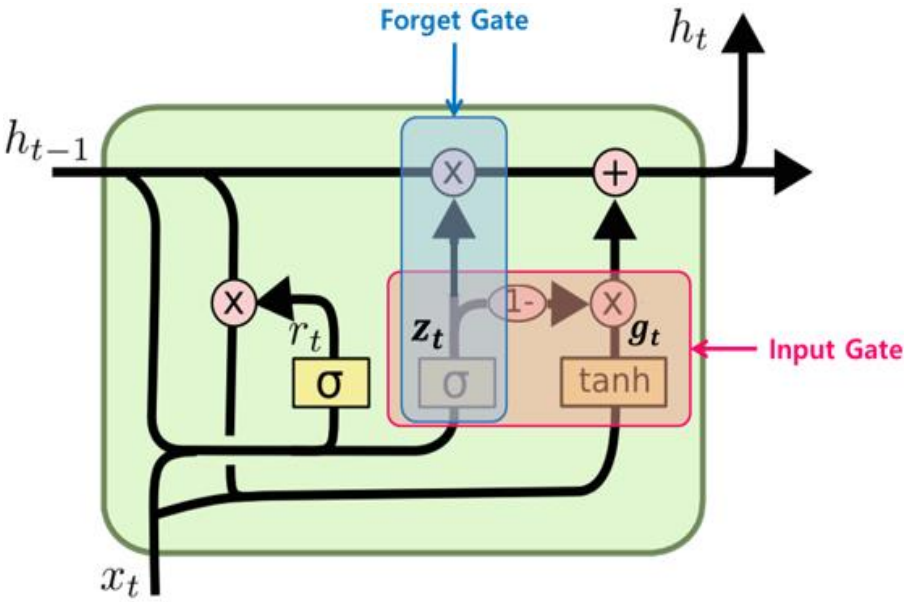
$$o_t = \sigma\left(w_{co}^T \cdot c_{t-1} + w_{xo}^T \cdot x_t + w_{ho}^T \cdot h_{t-1} + b_o\right)$$

각 게이트의 입력값으로 C_{t-1}을 사용하는 작은 구멍(peephole)을 만들어

장기 기억에서 지울 부분과 기억할 부분을 정할 때 장기 기억의 정보를 반영

4 LSTM의 변형 모델들

• Gated Recurrent Unit(GRU)



$$z_t = \sigma(w_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(w_r \cdot [h_{t-1}, x_t])$$

$$g_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

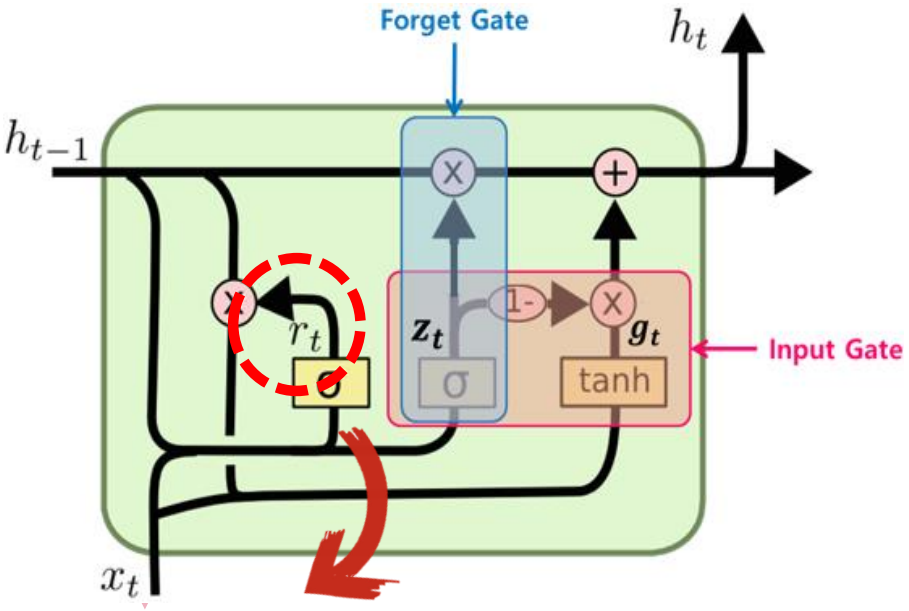
$$h_t = (1 - z_t) * h_{t-1} + z_t * g_t$$

LSTM보다 파라미터수가 적은 단순화된 모델 (z_t, r_t)

z_t(Update gate): 현재 시점 정보를 잊을 정도(0~1)

4 LSTM의 변형 모델들

● Gated Recurrent Unit(GRU)



Reset gate(r_t)

$$z_t = \sigma(w_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(w_r \cdot [h_{t-1}, x_t])$$

$$g_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

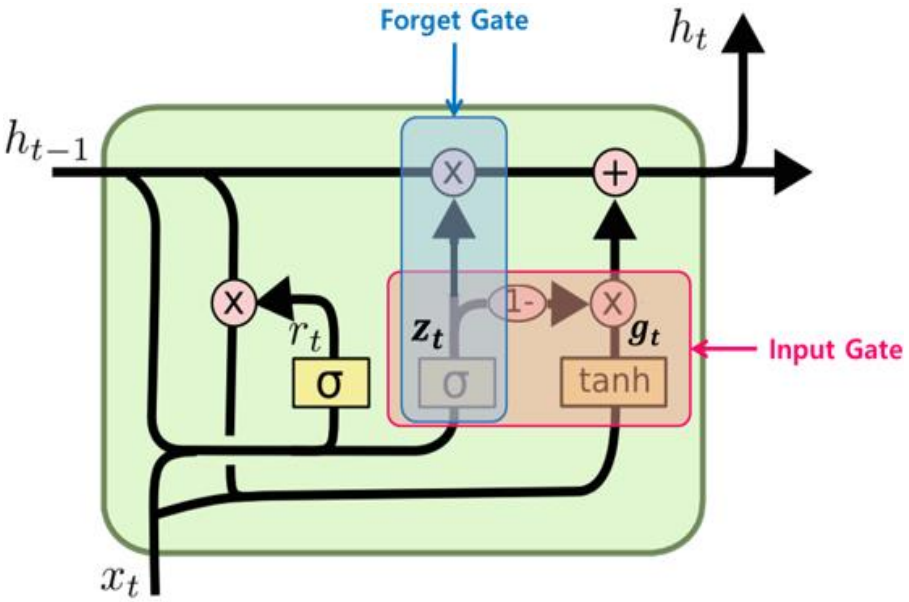
$$h_t = (1 - z_t) * h_{t-1} + z_t * g_t$$

r_t (Reset gate)

- Sigmoid 활성 함수로 0~1사이의 값
- 이전 정보를 얼마나 사용할 지 결정
- 0에 가까운 값이되면 'Reset'

4 LSTM의 변형 모델들

• Gated Recurrent Unit(GRU)



$$z_t = \sigma(w_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(w_r \cdot [h_{t-1}, x_t])$$

$$g_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * g_t$$

LSTM보다 Weight의 개수가 적으면서도 비슷한 성능을 보임

GRU는 LSTM보다 게이트 개수가 줄어들어서(4→2) 학습속도가 빠름