

# 딥러닝팀

## 1팀

이정환  
김동환  
권가민  
박준영  
박채원

# CONTENTS

---

1. 자연어

2. RNN

3. RNN 모델의 응용

4. eXplainable AI (XAI)

1

자연어

## 자연어란?

자연어 (Natural language)

의도와 목적을 갖고 만들어진 인공어와 구분하여  
우리가 일상생활에서 사용하는 언어를 부르는 개념

## 자연어란?

자연어 (Natural language)

의도와 목적을 갖고 만들어진 인공어와 구분하여  
우리가 일상생활에서 사용하는 언어를 부르는 개념

자연어처리 (NLP, Natural language Processing)

자연어(사람의 언어)를 컴퓨터가 이해하고  
여러가지 일을 처리할 수 있도록 하는 일

## 자연어란?

자연어 (Natural language)

의도와 목적을 갖고 만들어진 인공어와 구분하여  
우리가 **일상생활에서 사용하는 언어**를 부르는 개념

자연어처리 (NLP, Natural language Processing)

자연어(사람의 언어)를 **컴퓨터**가 이해하고  
여러가지 일을 처리할 수 있도록 하는 일



감성분석, Siri, Bixby (챗봇), Papago (기계번역)  
OpenAI의 ChatGPT, Google의 Bard 등의 거대언어모델(LLM)

## 자연어의 특징

### 순차성

순차적(Sequential) 데이터로, 순서가 달라지는 경우 의미가 손상됨

### 모호성

같은 단어가 다른 의미를 가지는 표현의 중의성, 문장 내 정보 부족

### 불연속성

형태가 유사한 두 단어라도 완전히 다른 의미를 가질 수 있음

## 자연어의 특징

### 순차성

순차적(Sequential) 데이터로, 순서가 달라지는 경우 의미가 손상됨

예시



## 자연어의 특징

### 순차성

순차적(Sequential) 데이터로, 순서가 달라지는 경우 의미가 손상됨

예시

“ 마감 시간이 끝나기 전에 과제를 끝냈다.”

≠

“ 과제가 끝나기 전에 마감 시간이 끝났다.”



순서를 바꿈으로써 의미가 크게 달라짐

## 자연어의 특징

### 모호성

표현의 중의성, 문장 내 정보 부족으로 발생

### 예시

## 자연어의 특징

### 모호성

표현의 **중의성**, 문장 내 정보 부족으로 발생

예시



“차를 마시러 공원에 가는 차 안에서 나는 그녀에게 차였다.”

## 자연어의 특징

### 모호성

표현의 **중의성**, 문장 내 정보 부족으로 발생

### 예시



“차를 마시러 공원에 가는 차 안에서 나는 그녀에게 차였다.”



“차”라는 단어는 여러 번 나오지만,  
각 “차”가 의미하는 바는 모두 다름

## 자연어의 특징

### 모호성

표현의 중의성, 문장 내 **정보 부족**으로 발생

예시



“나는 정환이를 안 때렸다.”

## 자연어의 특징

### 모호성

표현의 중의성, 문장 내 **정보 부족**으로 발생

예시



“나는 정환이를 안 때렸다.”



여러가지 경우로 해석할 수 있음

1. 정환이는 맞았지만, 때린 사람이 내가 아니다.
2. 나는 누군가를 때렸지만, 그게 정환이는 아니다.
3. 나는 누군가를 때린적도 없고, 정환이도 맞은 적이 없다.

## 자연어의 특징

### 불연속성

형태가 유사한 두 단어라도 완전히 다른 의미를 가질 수 있음

예시

## 자연어의 특징

### 불연속성

형태가 유사한 두 단어라도 완전히 다른 의미를 가질 수 있음

예시

"batch"  $\neq$  "catch"



한글자 차이로 형태가 유사하지만, 그 뜻은 전혀 다름



반대로 이미지 데이터에서는

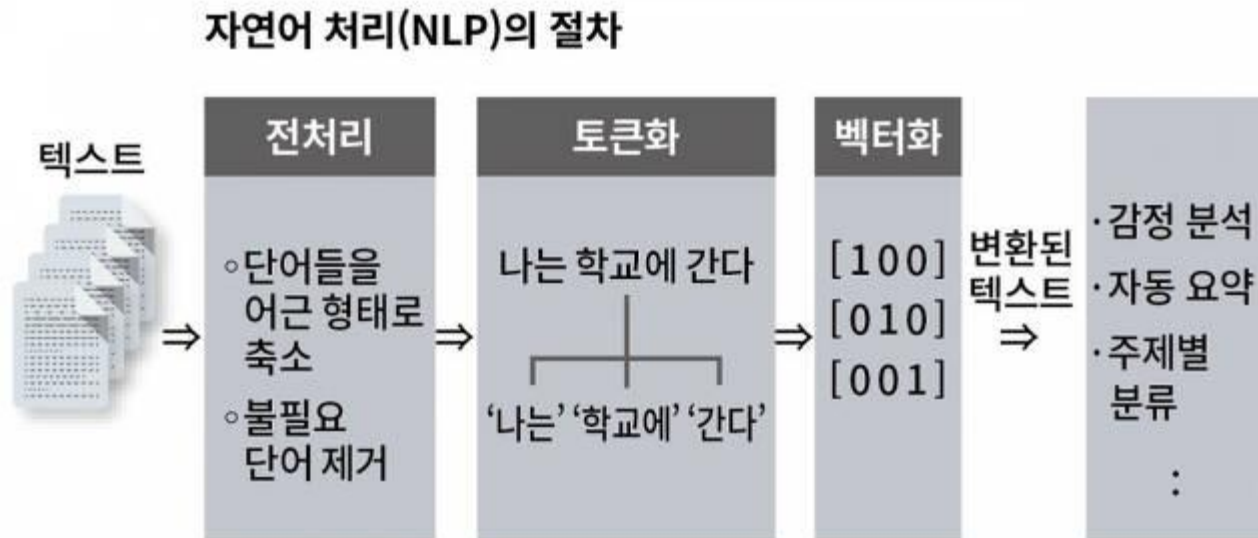
(255, 0, 0) 과 (254, 0, 0)

모두 빨간색 픽셀임으로

유사성 판단 가능



## 자연어의 전처리



텍스트 데이터를 모델의 입력으로 사용하기 위해서는  
앞선 특성을 반영하여 데이터를 변환해야 함

## 자연어의 전처리

정제 (Cleaning)

텍스트 데이터의 의미를 분석할 때, **필요하지 않은 부분을 처리**하는 작업

예시



불용어 (stopword) 제거



대소문자 제거

## 자연어의 전처리

### 정제 (Cleaning)

텍스트 데이터의 의미를 분석할 때, **필요하지 않은 부분을 처리**하는 작업

### 예시



불용어 (stopword) 제거

조사, 접미사 등 문장 내에서 자주 등장하지만  
실제 의미 분석에 거의 기여하지 않는 단어 제거

## 자연어의 전처리

정제 (Cleaning)

텍스트 데이터의 의미를 분석할 때, **필요하지 않은 부분을 처리**하는 작업

예시



대소문자 제거



정제 대상이 되는 전체 문장 데이터 집합  
=코퍼스(Corpus)

"He is 동환" = "he is 동환"

대문자와 소문자가 의미 분석에 영향을 주지 않는 경우

## 자연어의 전처리

토큰화 (Tokenization)

문장을 의미를 가지는 최소 단위로 잘게 쪼개는 과정

 토큰

## 자연어의 전처리

### 토큰화 (Tokenization)

문장을 의미를 가지는 최소 단위로 잘게 쪼개는 과정

 토큰

Tokenize on  
rules

Let	's	tokenize	!	Is	n't	this	easy	?
-----	----	----------	---	----	-----	------	------	---

Tokenize on  
punctuation

Let	'	s	tokenize	!	Isn	'	t	this	easy	?
-----	---	---	----------	---	-----	---	---	------	------	---

Tokenize on  
white spaces

Let's	tokenize!	Isn't	this	easy?
-------	-----------	-------	------	-------



한 문장이라도 여러 기준에 따라 다양한 방식으로 토큰화 가능함



## 자연어의 전처리

토큰화 (Tokenization)

의미를 가지는 최소 단위는  
문장을 의미를 가지는 최소 단위로 잘게 쪼개는 과정  
목적에 따라, 정의에 따라 달라질 수 있기 때문에

한 문장이라도 여러 기준에 따라 다양한 방식으로 토큰화가 가능함

Tokenize on

Tokenize on  
punctuation

Let	'	s	tokenize	!	Isn	'	t	this	easy	?
-----	---	---	----------	---	-----	---	---	------	------	---

Tokenize on  
white spaces

Let's	tokenize!	Isn't	this	easy?
-------	-----------	-------	------	-------



한 문장이라도 여러 기준에 따라 다양한 방식으로 토큰화 가능함



## 자연어의 전처리

토큰화 (Tokenization)

의미를 가지는 최소 단위는  
문장을 의미를 가지는 최소 단위로 잘게 쪼개는 과정  
목적에 따라, 정의에 따라 달라질 수 있기 때문에

한 문장이라도 여러 기준에 따라 다양한 방식으로 토큰화가 가능함



기준이 명확히 정해져 있지 않은 만큼

필요에 따라 방법을 잘 선택해야 함

한 문장이라도 여러 기준에 따라 다양한 방식으로 토큰화 가능함





## Word Vector

단어 집합 (Vocabulary)

서로 다른 단어들의 **중복을 허용하지 않고** 모아 놓은 집합



이산적인 단어의 의미를 표현하기 위해서 만들어진 것



단어를 vector로..?

## Word Vector

One-hot-vector

N개의 단어를 각각 N개의 차원의 벡터로 나타내는 방법

Embedding Vector

사용자가 설정한 값으로 모든 단어의 벡터 표현을 맞춤

## Word Vector

One-hot-vector

N개의 단어를 각각 N개의 차원의 벡터로 나타내는 방법



One-hot-encoding

## Word Vector

One-hot-vector

N개의 단어를 각각 N개의 차원의 벡터로 나타내는 방법



One-hot-encoding

희소표현 (Sparse representation)

벡터 또는 행렬의 값을 대부분 0으로 표현하는 방법

표현하고 싶은 단어의 index에는 1, 나머지는 0을 넣음

Hotel = [0 0 0 1 0 0 0 0 0]

Motel = [0 0 0 0 0 0 1 0 0]

## Word Vector

One-hot-vector

N개의 단어를 각각 N개의 차원의 벡터로 나타내는 방법



One-hot-encoding

희소표현 (Sparse representation)



두 단어가 서로 직교하여 유사성을 찾을 수 없음

단어가 많아졌을 때 공간적 낭비가 심하고 차원이 커지는 문제가 있음

Hotel = [0 0 0 1 0 0 0 0 0 0]

Motel = [0 0 0 0 0 0 1 0 0 0]

## Word Vector

Embedding Vector

사용자가 설정한 차원크기로 모든 단어의 벡터 표현을 맞추는 방법



Word Embedding

## Word Vector

### Embedding Vector

사용자가 설정한 차원크기로 모든 단어의 벡터 표현을 맞추는 방법



Word Embedding

### 밀집 표현 (Dense representation)

0과 1로만 표현하지 않고 **실수값도** 이용하는 방법

각 요소는 실수로 표현됨

Hotel = [0.7 0.3 0.1 0.5]

Motel = [0.5 0.1 0.3 0.6]

## Word Vector

### Embedding Vector

사용자가 설정한 차원크기로 모든 단어의 벡터 표현을 맞추는 방법



Word Embedding

### 밀집 표현 (Dense representation)



텍스트를 단순히 구분하는 것이 아니라

**의미적으로 정의**한다고 볼 수 있음

Hotel = [0.7 0.3 0.1 0.5]

Motel = [0.5 0.1 0.3 0.6]



## Word Vector

Word2Vec

“비슷한 위치에 등장하는 단어는 비슷한 의미를 가진다”고 가정하는

Word Embedding의 가장 대표적인 방법

## Word Vector

Word2Vec

“비슷한 위치에 등장하는 단어는 비슷한 의미를 가진다”고 가정하는

Word Embedding의 가장 대표적인 방법

종류



CBOW : 주변의 단어들을 사용하여 중간에 있는 단어를 예측하는 방법



Skip-gram : 중간에 있는 단어를 사용하여 주변에 있는 단어들을 예측하는 방법

## Word Vector

Word2Vec

“비슷한 위치에 등장하는 단어는 비슷한 의미를 가진다”고 가정하는

Word Embedding의 가장 대표적인 방법

종류



CBOW : 주변의 단어들을 사용하여 중간에 있는 단어를 예측하는 방법



Skip-gram : 중간 단어 사용 하여 주변에 있는 단어들을 예측하는 방법



일반적으로 성능이 더 좋음

## Skip-gram

## Source Text

## Training Samples

The quick brown fox jumps over the lazy dog. →

(the, quick)  
(the, brown)

The quick brown fox jumps over the lazy dog. →

(quick, the)  
(quick, brown)  
(quick, fox)

The quick brown fox jumps over the lazy dog. →

(brown, the)  
(brown, quick)  
(brown, fox)  
(brown, jumps)

The quick brown fox jumps over the lazy dog. →

(fox, quick)  
(fox, brown)  
(fox, jumps)  
(fox, over)

## Skip-gram

Source Text

Training  
Samples

The quick brown fox jumps over the lazy dog. →

(the, quick)  
(the, brown)

The quick brown fox jumps over the lazy dog. →

(quick, the)  
(quick, brown)  
(quick, fox)

The quick brown fox jumps over the lazy dog. →

(brown, the)  
(brown, quick)  
(brown, fox)  
(brown, jumps)

파란색을 입력으로

The quick brown fox jumps over the lazy dog. →

(fox, quick)  
(fox, brown)  
(fox, jumps)  
(fox, over)

## Skip-gram

Source Text

Training  
Samples

The quick brown fox jumps over the lazy dog. →

(the, quick)  
(the, brown)

The quick brown fox jumps over the lazy dog. →

(quick, the)  
(quick, brown)  
(quick, fox)

The quick brown fox jumps over the lazy dog. →

(brown, the)  
(brown, quick)  
(brown, fox)  
(brown, jumps)

파란색을 입력으로 주변 2N개의 단어에 대한 예측을 수행함

The quick brown fox jumps over the lazy dog. →

(fox, quick)  
(fox, brown)  
(fox, jumps)  
(fox, over)

## Skip-gram

Source Text

Training  
Samples

The quick brown fox jumps over the lazy dog. →

(the, quick)  
(the, brown)

The quick brown fox jumps over the lazy dog. →

(quick, the)  
(quick, brown)  
(quick, fox)

The quick brown fox jumps over the lazy dog. →

(brown, the)  
(brown, quick)  
(brown, fox)  
(brown, jumps)

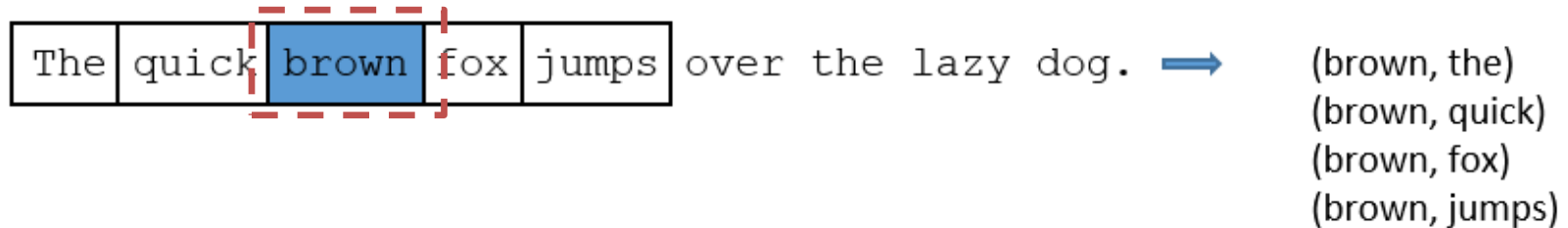
window size : 학습에 사용할 단어 개수  $N$

The quick brown fox jumps over the lazy dog. →

(fox, quick)  
(fox, brown)  
(fox, jumps)  
(fox, over)

## Skip-gram

brown을 입력으로 할 때





## Skip-gram

brown을 입력으로 할 때



(brown, the)  
(brown, quick)  
(brown, fox)  
(brown, jumps)

4개의 단어를 개별적으로 예측 및 업데이트

## Skip-gram

brown을 입력으로 할 때



(brown, the)  
(brown, quick)  
(brown, fox)  
(brown, jumps)

4개의 단어를 개별적으로 예측 및 업데이트

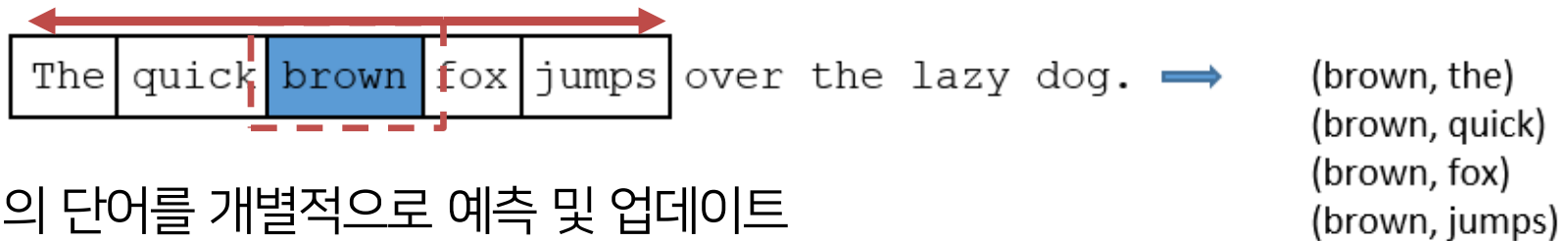
이때의 window size = 2

## Skip-gram



제한된 문장 안에서 여러번 학습하며 효과적으로 임베딩 벡터를 만들 수 있고,  
각 단어가 고차원 벡터 공간의 한점으로 표시 되어 **벡터 간 연산이 가능해짐**

brown을 입력으로 할 때



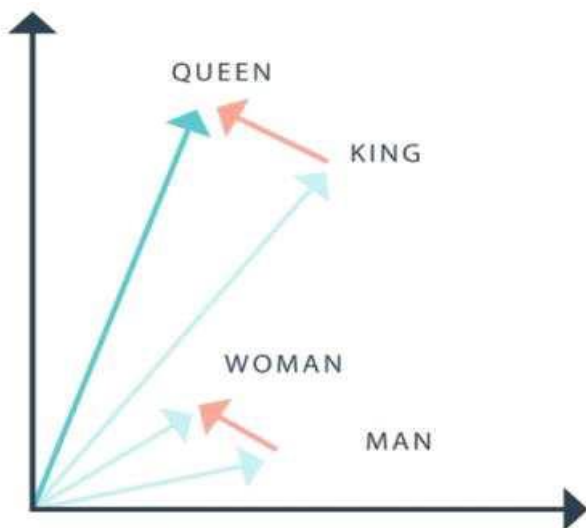
4개의 단어를 개별적으로 예측 및 업데이트

이때의 window size = 2

## Skip-gram



제한된 문장 안에서 여러번 학습하며 효과적으로 임베딩 벡터를 만들 수 있고, 각 단어가 고차원 벡터 공간의 한점으로 표시 되어 **벡터 간 연산이 가능해짐**



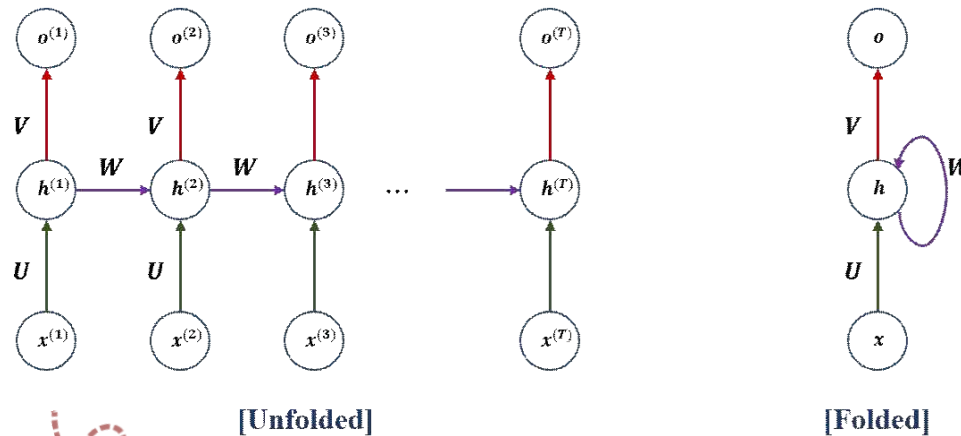
### 단어 간 연산 예시

+ King	[0.3, 0.7, 0.1, 0.5]
- Man	[0.2, 0.2, -0.3, 0.5]
+ Woman	[0.6, 0.3, 0.2, 0.8]
<hr/>	
≈ Queen	[0.7, 0.8, 0.6, 0.8]

2

RNN

## Vanilla RNN

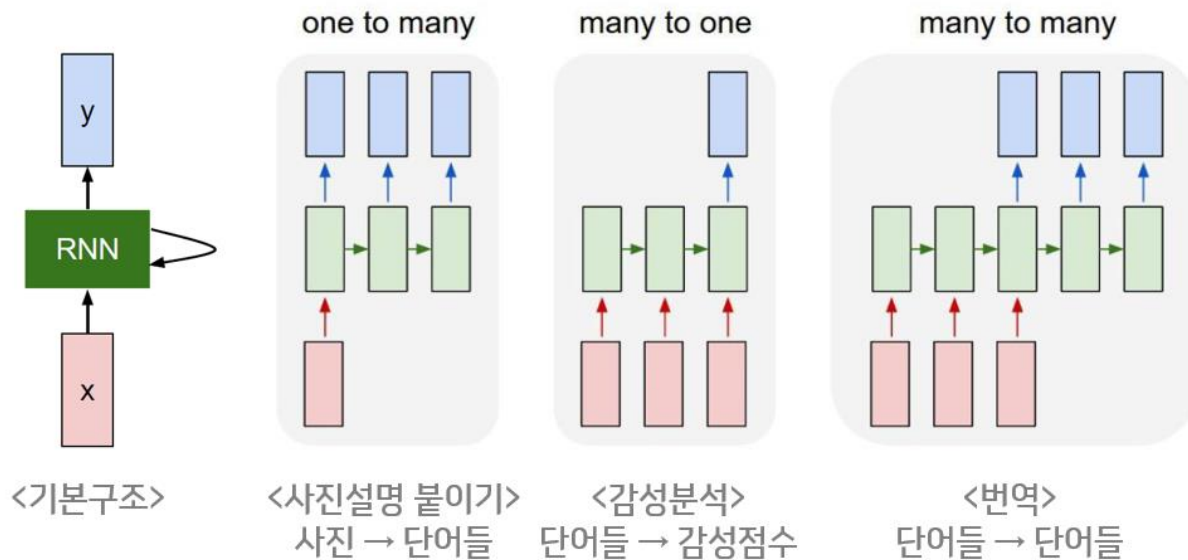


순환적인 구조

시점에 따라 풀어서 표현함

Sequence 데이터를 학습함에 있어  
 하나의 Hidden layer를 반복해서 사용함

## Vanilla RNN



RNN 모델은 목적에 따라 3가지 구조로 활용 할 수 있음

## Vanilla RNN

### RNN 순전파 과정

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)},$$

$$h^{(t)} = \tanh(a^{(t)}),$$

$$o^{(t)} = c + Vh^{(t)},$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$$

$b, c =$  편향 벡터

$U, V, W =$  가중치 행렬

매개변수



## Vanilla RNN


### RNN 순전파 과정

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)},$$

$$h^{(t)} = \tanh(a^{(t)}),$$

$$o^{(t)} = c + Vh^{(t)},$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$$

$b, c$  = 편향 벡터  매개변수  
 $U, V, W$  = 가중치 행렬



매개변수들은  $t$  에 대해 의존하고 있지 않음  
따라서 순전파 과정에서 매개변수는 공유되며 변하지 않음

## Vanilla RNN


### RNN 순전파 과정

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)},$$

$$h^{(t)} = \tanh(a^{(t)}),$$

$$o^{(t)} = c + Vh^{(t)},$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)})$$

$b, c$  = 편향 벡터  매개변수  
 $U, V, W$  = 가중치 행렬



$\tanh$ 는 -1에서 1 사이의 값을 갖기 때문에  
다음단계로 전달될 정보를 효과적으로 표현할 수 있음

## Vanilla RNN

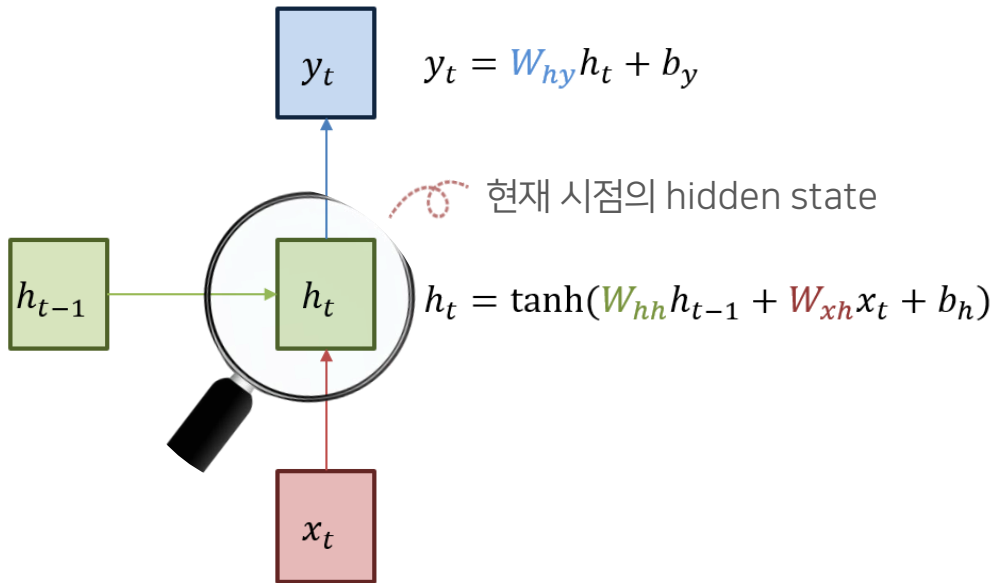
Hidden State

이전 시점의 데이터들을 기억하는 역할로 Memory cell이라고도 부름

## Vanilla RNN

### Hidden State

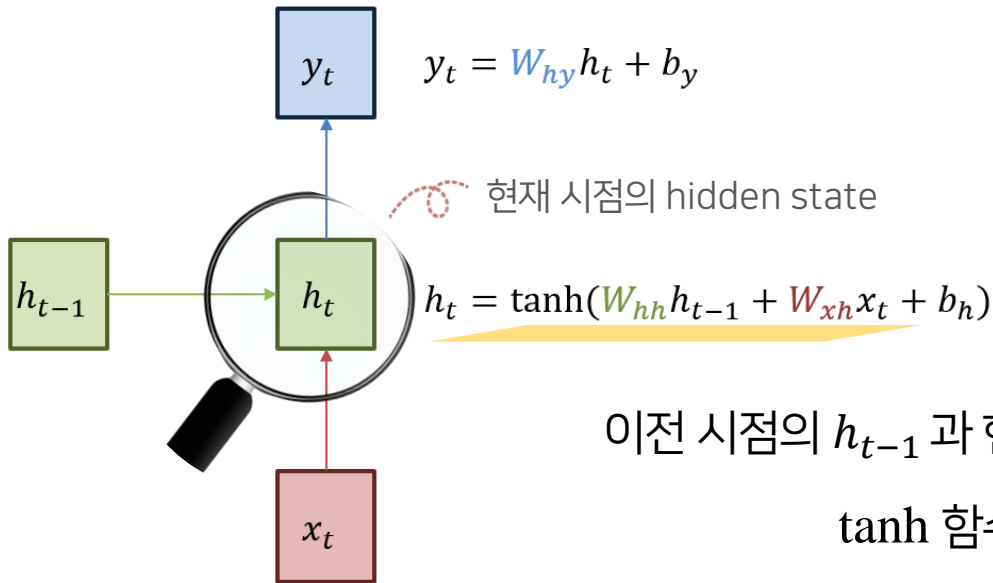
이전 시점의 데이터들을 기억하는 역할로 Memory cell이라고도 부름



## Vanilla RNN

### Hidden State

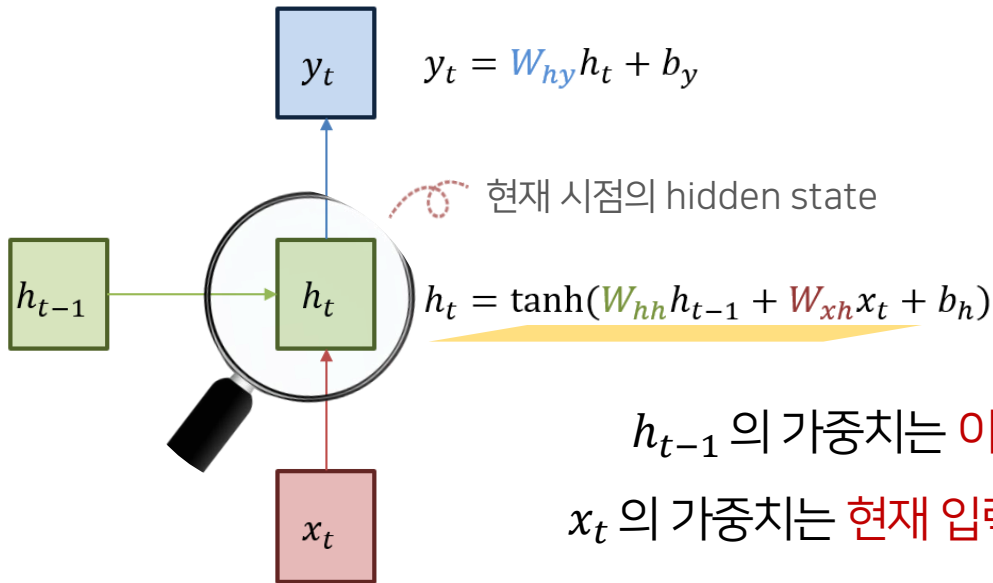
이전 시점의 데이터들을 기억하는 역할로 Memory cell이라고도 부름



## Vanilla RNN

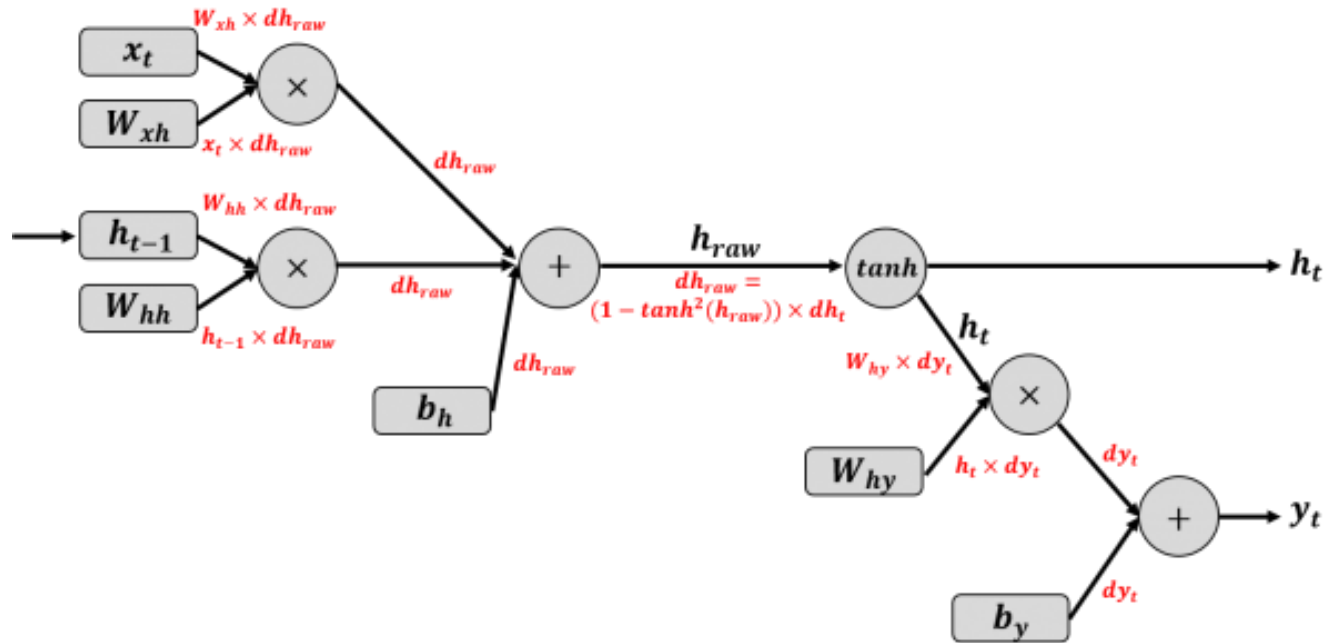
### Hidden State

이전 시점의 데이터들을 기억하는 역할로 Memory cell이라고도 부름



$h_{t-1}$ 의 가중치는 **이전 상태**를 얼마나 반영할 지를  
 $x_t$ 의 가중치는 **현재 입력**을 얼마나 반영 할 지를 조절해줌

## 역전파



RNN은 구조에 따라 출력이 한 개이거나 여러 개인데,  
 개수와 상관없이 **각 출력마다 Loss를 계산하여 해당 값을 역전파함**

## 역전파

모든 시점에 대해서 순전파가 이루어진 후에 역전파가 가능함

이전 시점부터 역전파를 진행하게 된다면 순차적 정보가 꼬이게 됨

각 시점의 Hidden state를 메모리에 저장하고 있어야 함



## 역전파

모든 시점에 대해서 순전파가 이루어진 후에 역전파가 가능함



이전 시점부터 역전파를 진행하게 된다면 순차적 정보가 꼬이게 됨

### 어지럼증 있을 때 동반증상

각 시점의 Hidden state를 메모리에 저장하고 있어야



두통



이명



안구통증



관자놀이통증



뒷목통증



이마통증

## 역전파

모든 시점에 대해서 순전파가 이루어진 후에 역전파가 가능함

이전 시점부터 역전파를 진행하게 된다면 순차적 정보가 꼬이게 됨

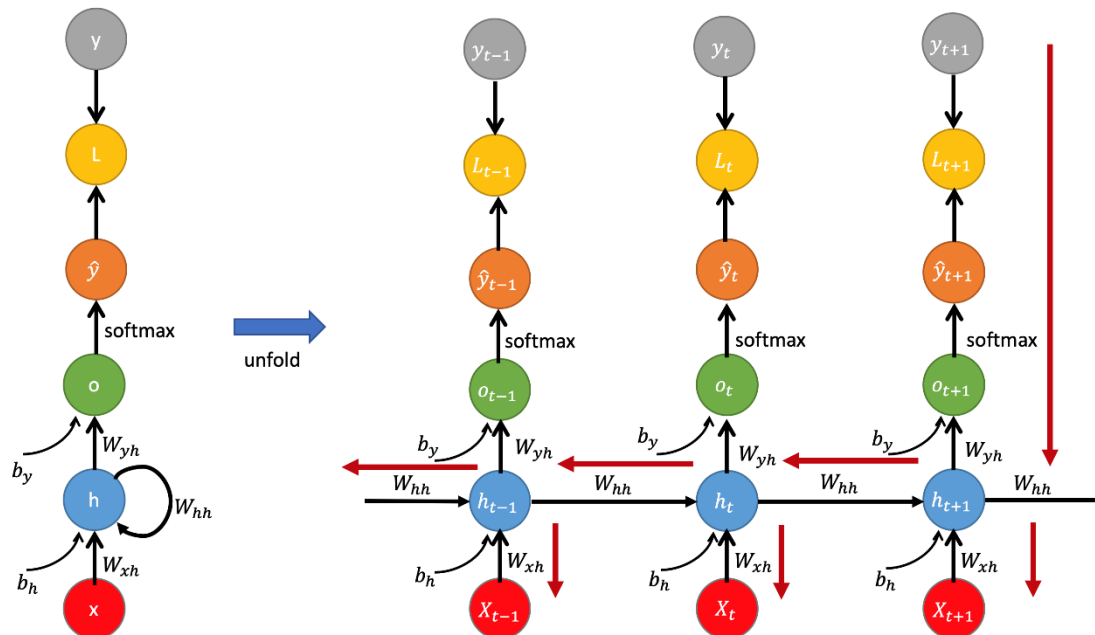


각 시점의 Hidden state를 메모리에 저장하고 있어야 함

## 역전파

BPTT (back-propagation through time)

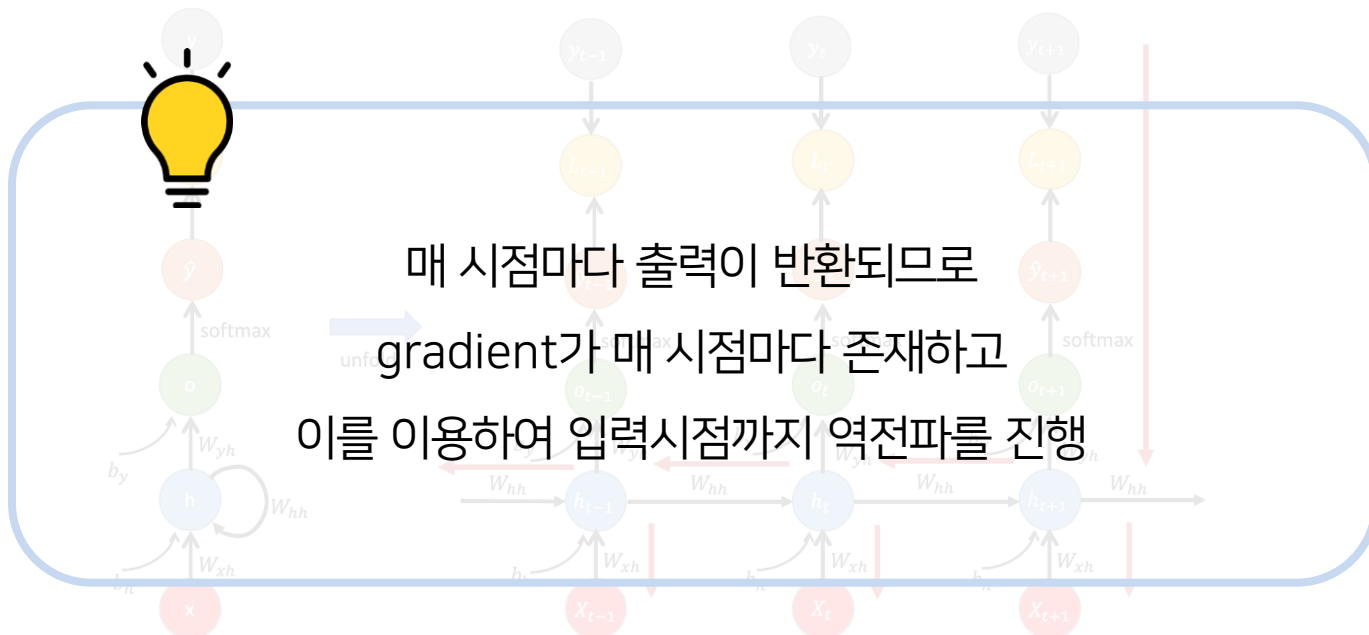
매 시점마다 출력이 반환되는 형태의 모델에서 활용됨



## 역전파

BPTT (back-propagation through time)

매 시점마다 출력이 반환되는 형태의 모델에서 활용됨



## 역전파

BPTT (back-propagation through time)

매 시점마다 출력이 반환되는 형태의 모델에서 활용됨



만약 데이터의 길이가 긴 경우

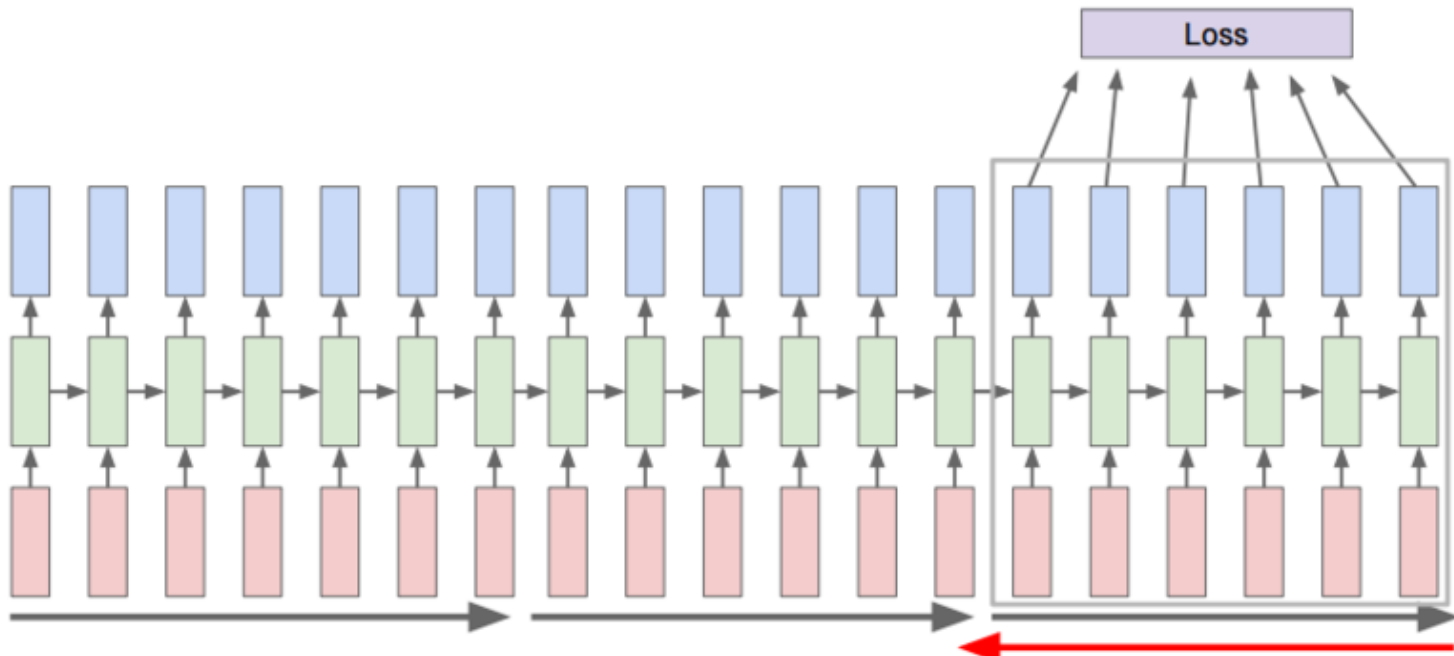
데이터의 후반부로 갈수록

연산에 소요되는 시간이 길어져 매우 비효율적임

## 역전파

Truncated BPTT (back-propagation through time)

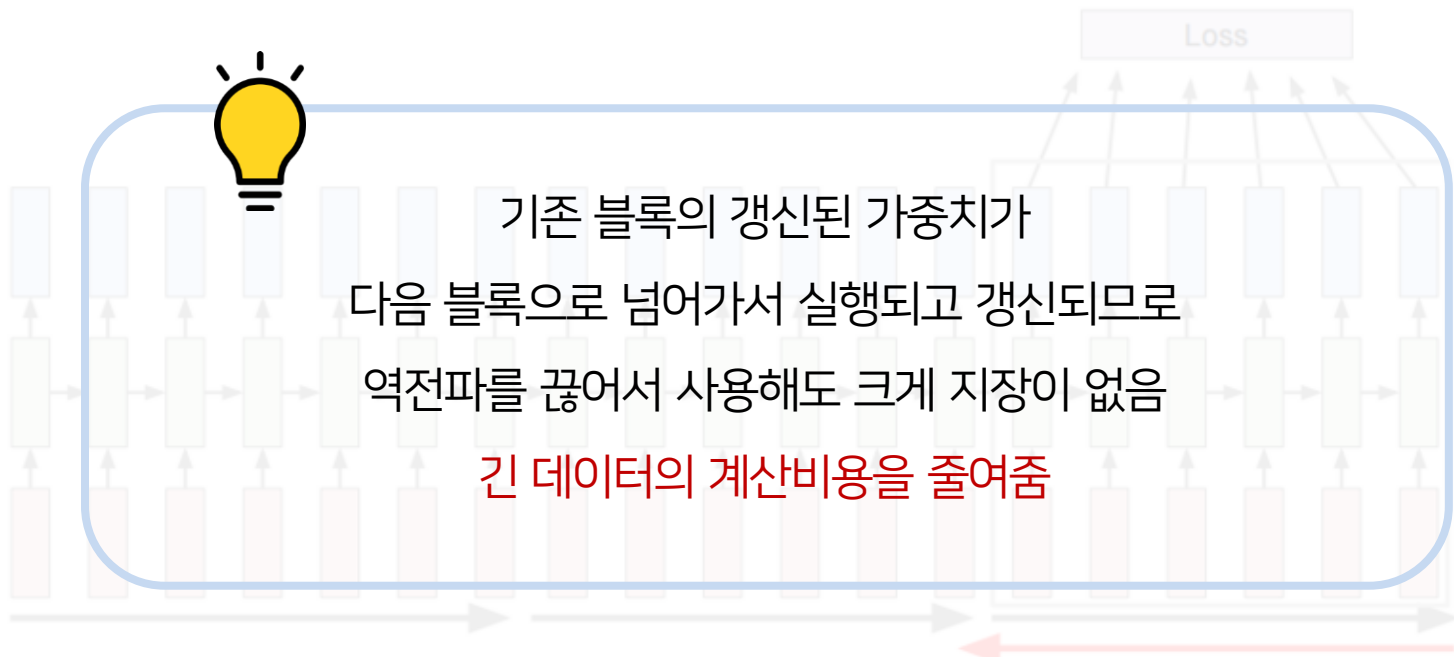
Sequential Data를 일정한 구간으로 끊어 구간마다 BPTT를 진행하는 형태



## 역전파

Truncated BPTT (back-propagation through time)

Sequential Data를 일정한 구간으로 끊어 구간마다 BPTT를 진행하는 형태





## 역전파

### 장기 의존성

Truncated BPTT (back-propagation through time)

RNN일 경우  $(-1, 1)$  사이의 값을 갖는 tanh 함수를 이용함

순전파 과정에서 데이터의 길이가 길어질수록

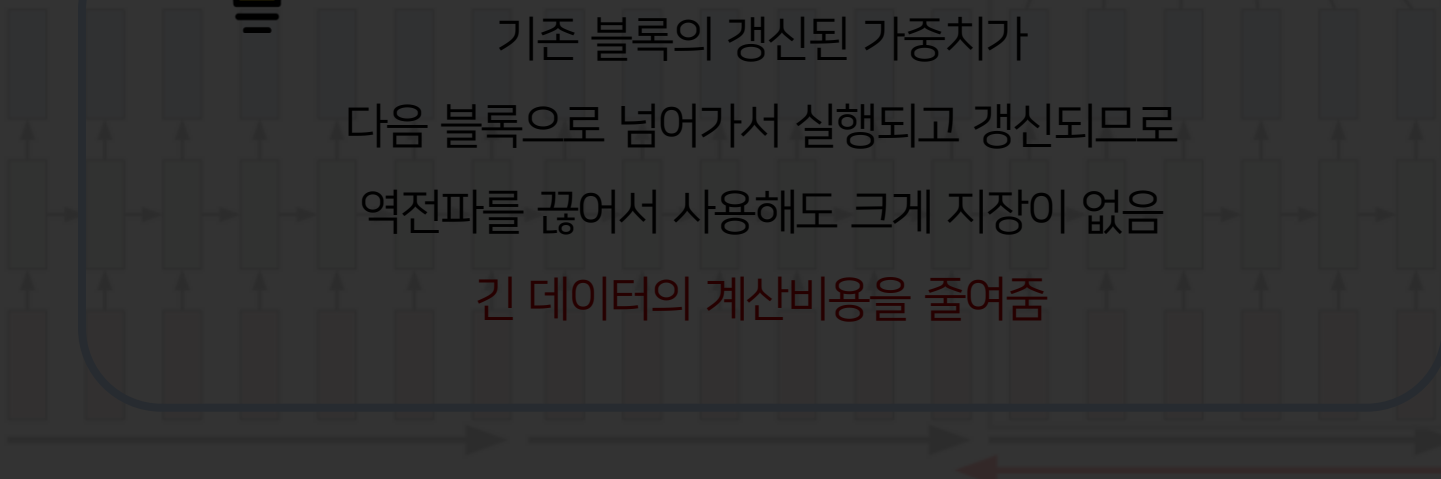
먼 시점의 데이터를 제대로 처리하지 못하는 문제가 발생

기존 블록의 갱신된 가중치가

다음 블록으로 넘어가서 실행되고 갱신되므로

역전파를 끊어서 사용해도 크게 지장이 없음

긴 데이터의 계산비용을 줄여줌







## 역전파

### 장기 의존성

Truncated BPTT (back-propagation through time)

RNN일 경우  $(-1, 1)$  사이의 값을 갖는 tanh 함수를 이용함  
Sequential Data를 일정한 구간으로 끊어 구간마다 BPTT를 실행하는 형태

순전파 과정에서 데이터의 길이가 길어질수록

먼 시점의 데이터를 제대로 처리하지 못하는 문제가 발생

기존 블록의 갱신된 가중치가

다음 블록으로 넘어가며 실행되고 갱신되므로

역전파를 끊어서 사용해도 크게 지장이 없음

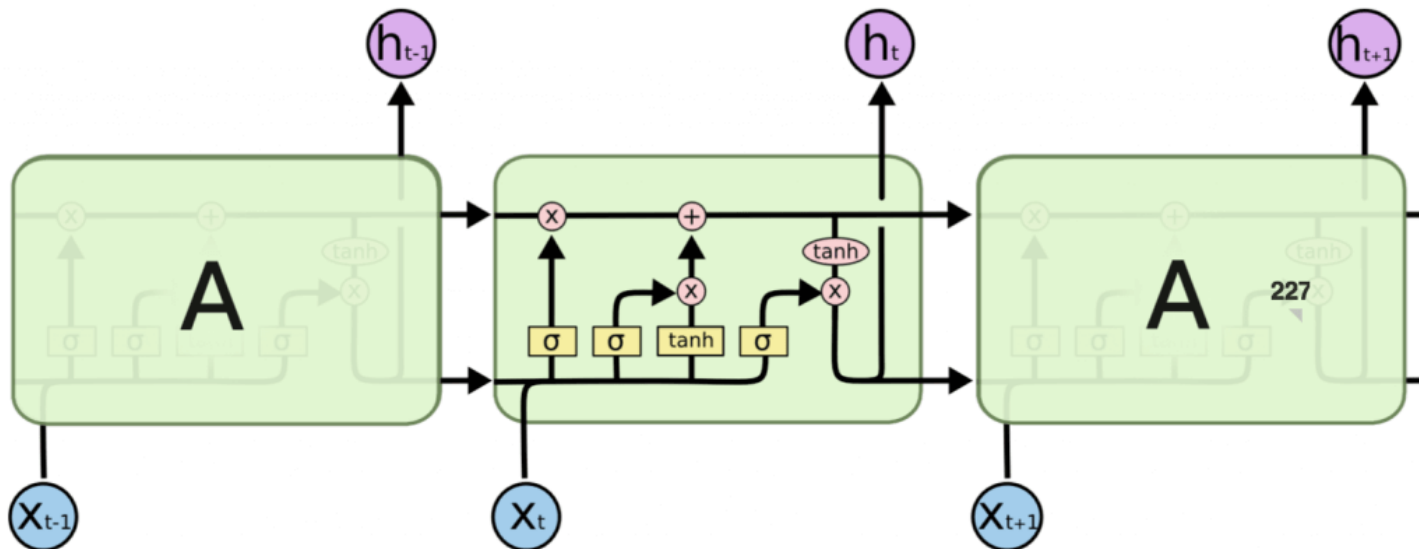
이를 장기 의존성 문제라 하며

해결책으로 LSTM과 GRU가 제안됨!

## LSTM/GRU

LSTM(Long Short-Term Memory)

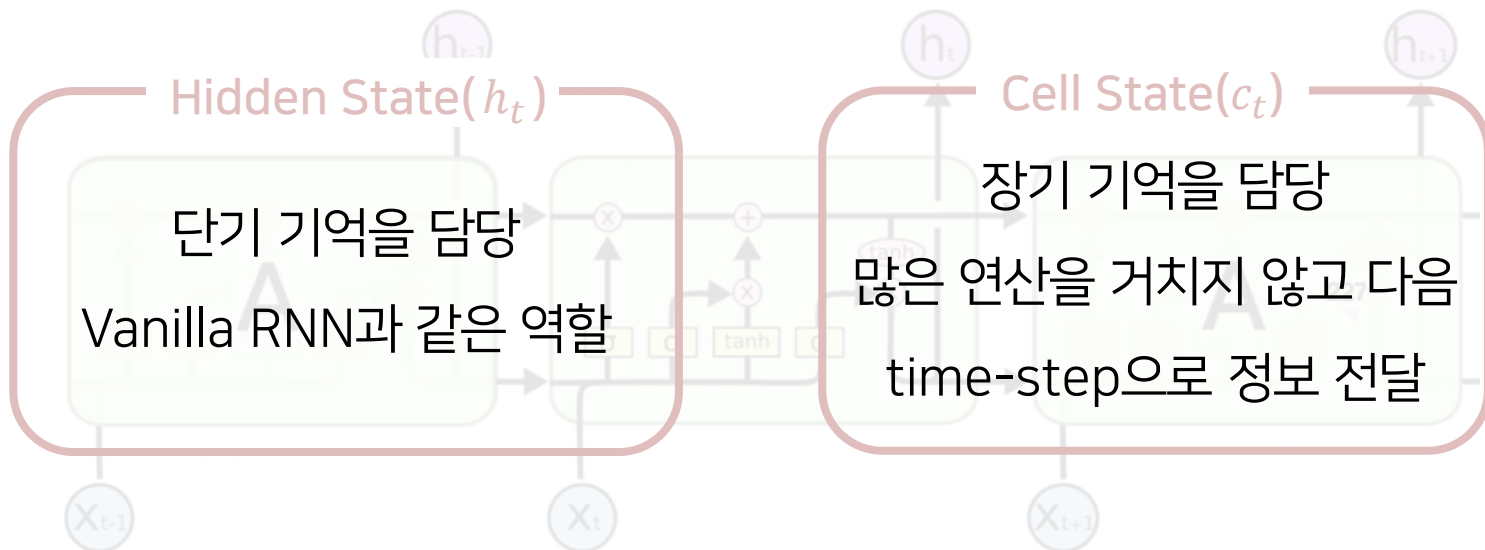
사람의 기억이 **장기 기억(Cell State)**과 **단기 기억(Hidden State)**으로  
나누어져 있다는 점에 착안한 모델로 **장기 의존성 문제 해결**



## LSTM/GRU

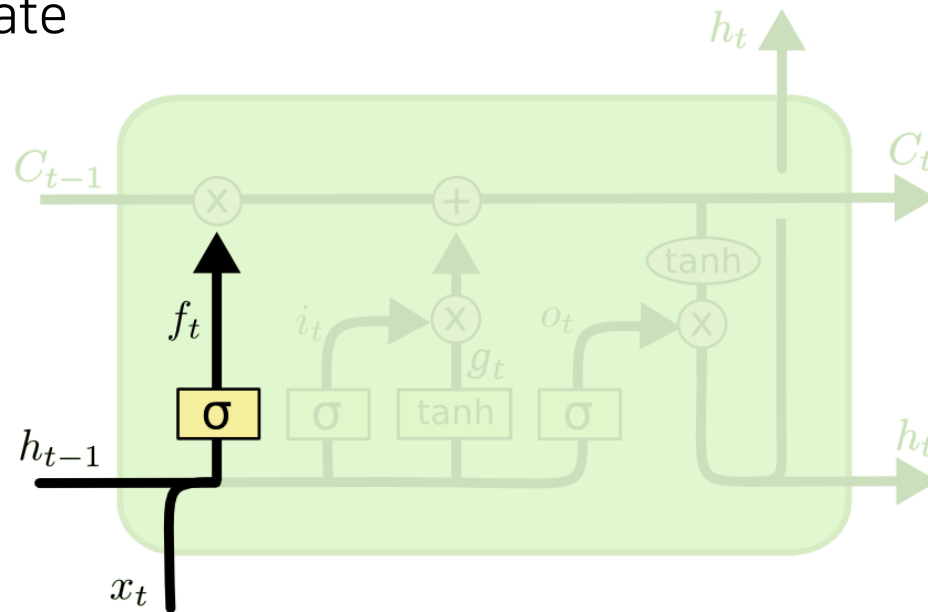
LSTM(Long Short-Term Memory)

사람의 기억이 **장기 기억(Cell State)**과 **단기 기억(Hidden State)**으로  
나누어져 있다는 점에 착안한 모델로 **장기 의존성 문제 해결**



## LSTM

## Forget Gate



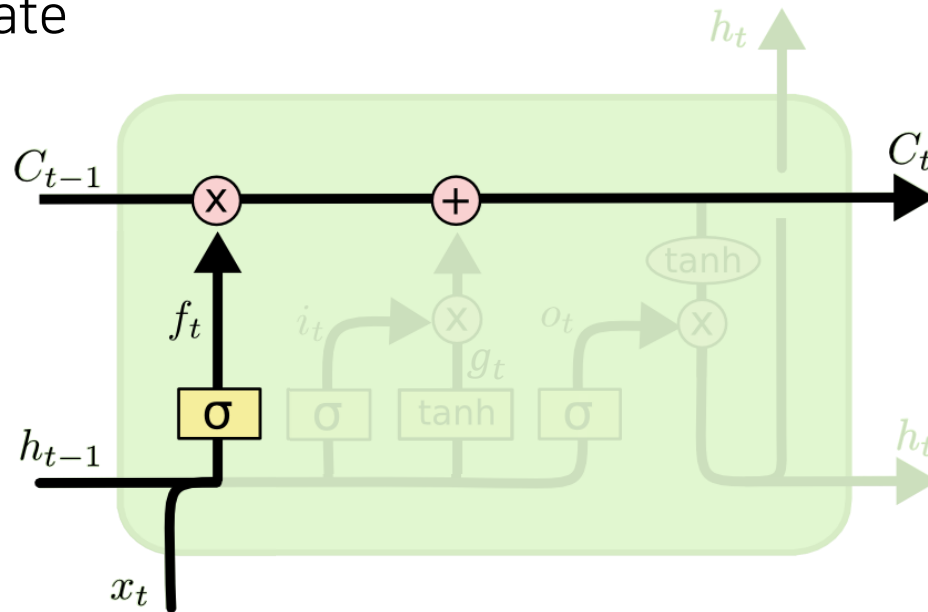
과거의 정보를 얼마나 잊을지 결정하는 역할

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

두 벡터를 연결하는 Concatenation을 의미

## LSTM

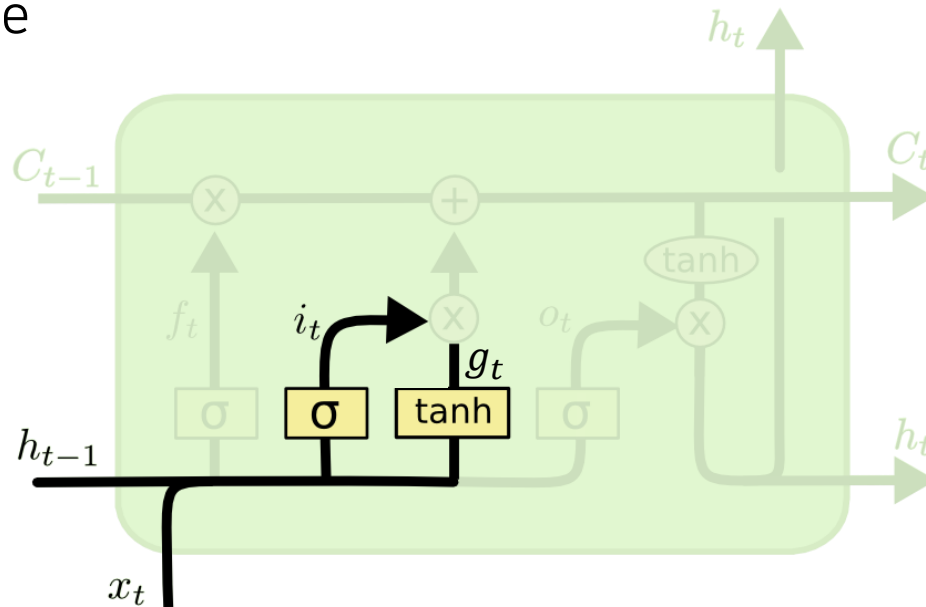
## Forget Gate



이전 시점의 hidden state와 현재 시점의 입력을  
가중치  $W_f$ 와 곱하여 시그모이드 통과 후 Cell State로 전달

## LSTM

## Input Gate

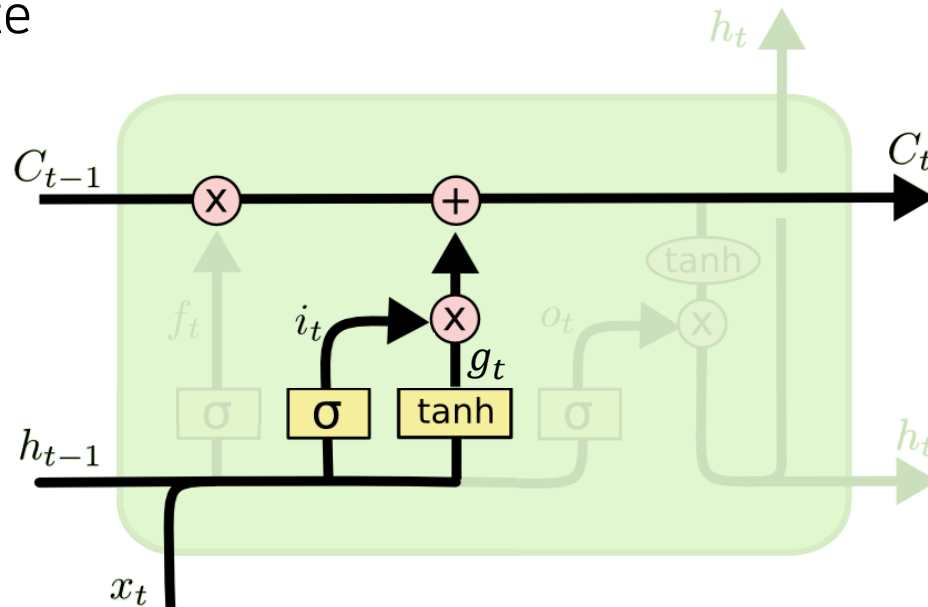


현재의 정보를 얼마나 기억할지 결정하는 역할

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

## LSTM

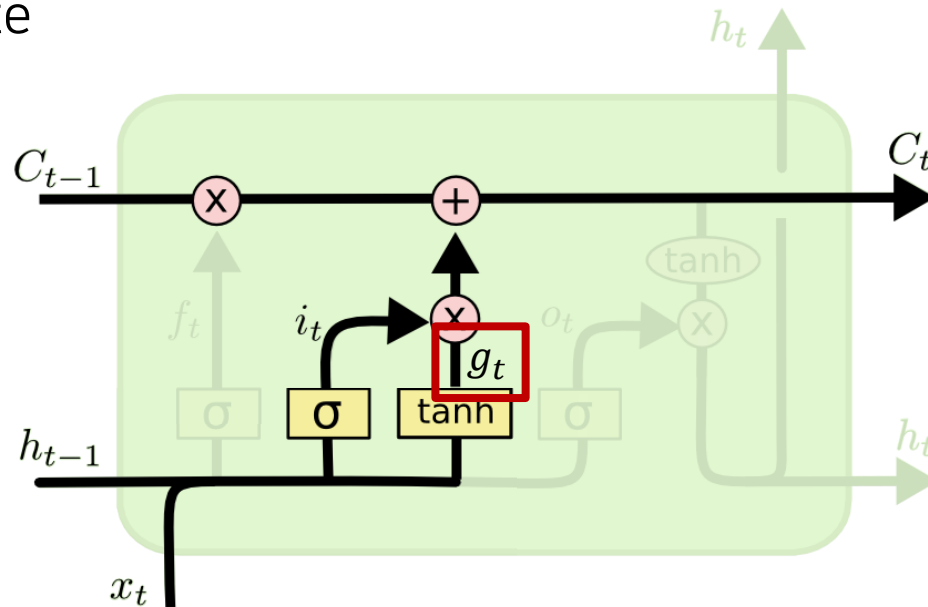
## Input Gate



이전 시점의 hidden state와 현재 시점의 입력을  
가중치  $w_i$ 와 곱하여 시그모이드 통과 후  $g_t$ 로 전달

## LSTM

## Input Gate



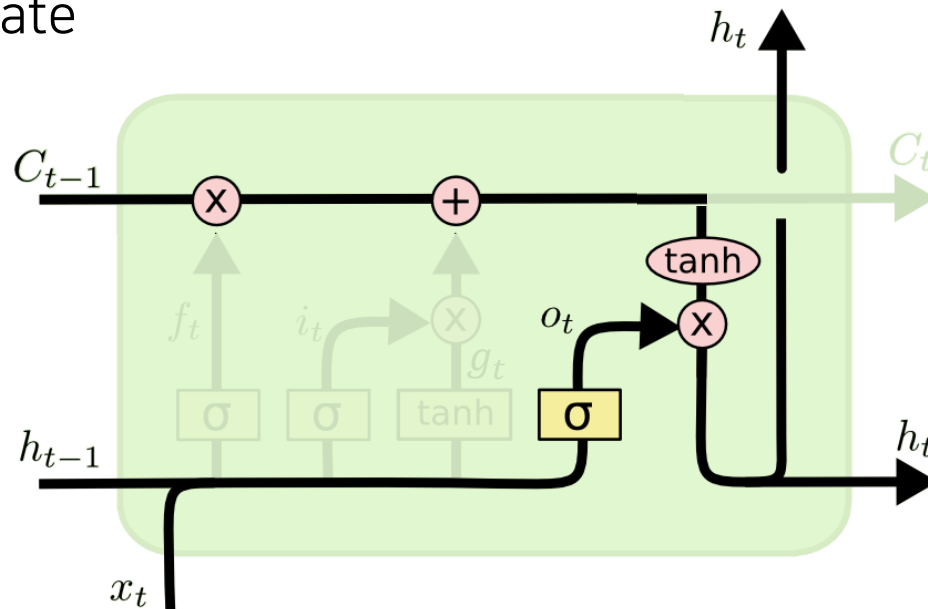
현재의 정보 중 어떤 정보를 Cell State에 전달할지 결정

$$g_t = \tanh(W_g \cdot [h_{t-1}, x_t] + b_g)$$



## LSTM

Output Gate

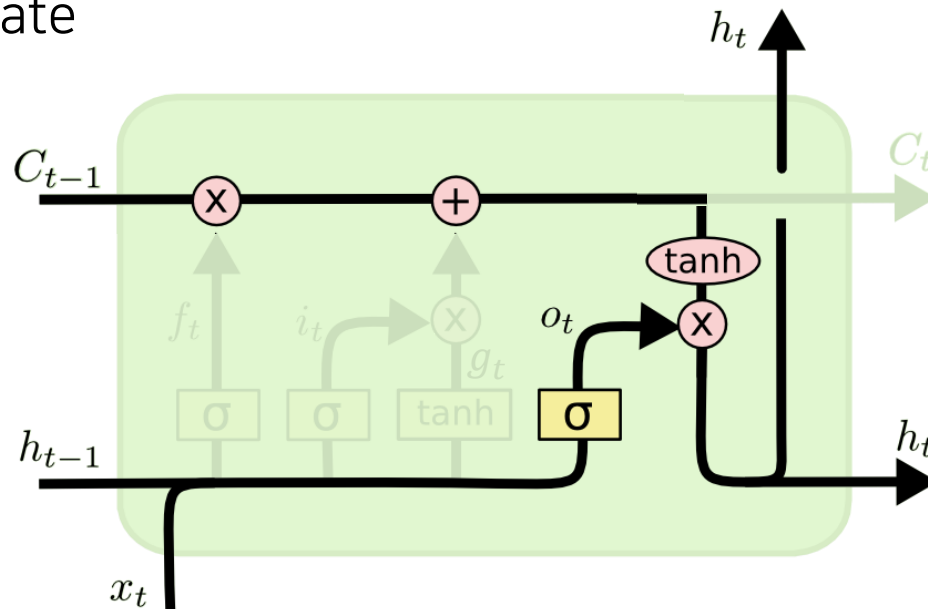


현재까지의 정보들 중 어떤 정보를 얼마나 활용할지 결정하는 역할

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

## LSTM

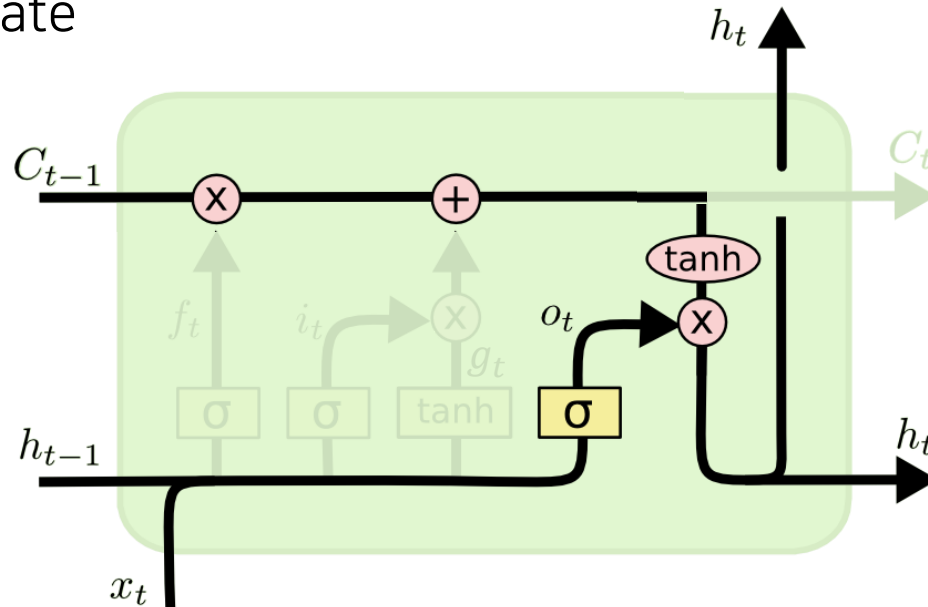
## Output Gate



Output Gate의 값과 Cell State의 값을 바탕으로  
현재 시점의 Hidden State( $h_t$ )를 결정

## LSTM

Output Gate

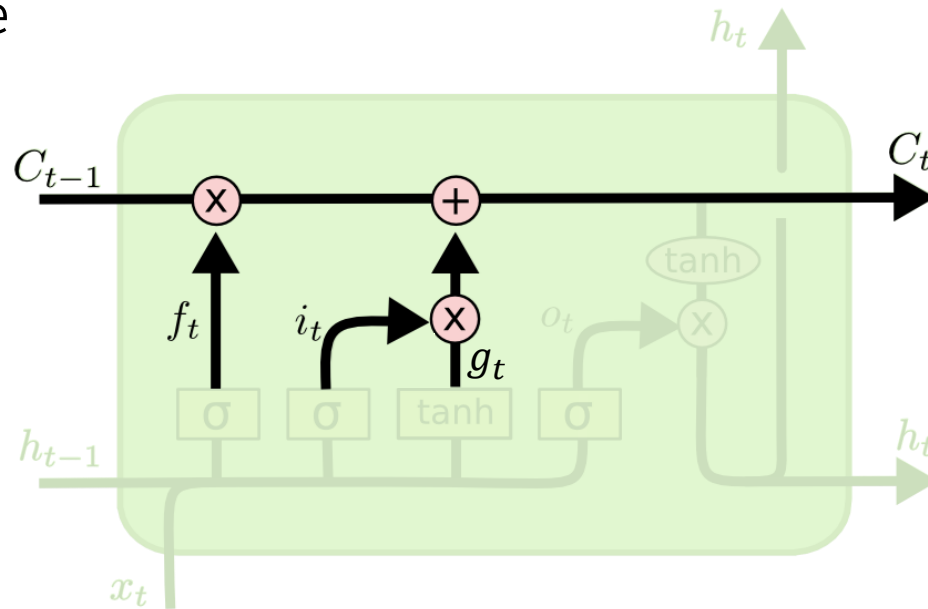


이를 수식으로 표현하면 다음과 같음

$$h_t = o_t \times \tanh(C_t)$$

## LSTM

Cell State



Forget Gate와 Input Gate의 출력을 바탕으로 Cell state ( $C_t$ ) 결정

$$C_t = f_t \times C_{t-1} + i_t \times g_t$$



## 시그모이드

### LSTM

Cell State

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

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

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

각각의 게이트는 활성화로 시그모이드를 이용함

Forget Gate와 Input Gate의 출력을 바탕으로 Cell state ( $C_t$ ) 결정

$$C_t = f_t \times C_{t-1} + i_t \times g_t$$



## 시그모이드

### LSTM

Cell State

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

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

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

각각의 게이트는 활성화로 시그모이드를 이용함



Forget Gate와 Input Gate의 출력을 바탕으로 Cell state ( $C_t$ ) 결정

이를 "정보를 얼마나 반영할지"로 해석할 수 있음

$$C_t = f_t \times C_{t-1} + i_t \times g_t$$

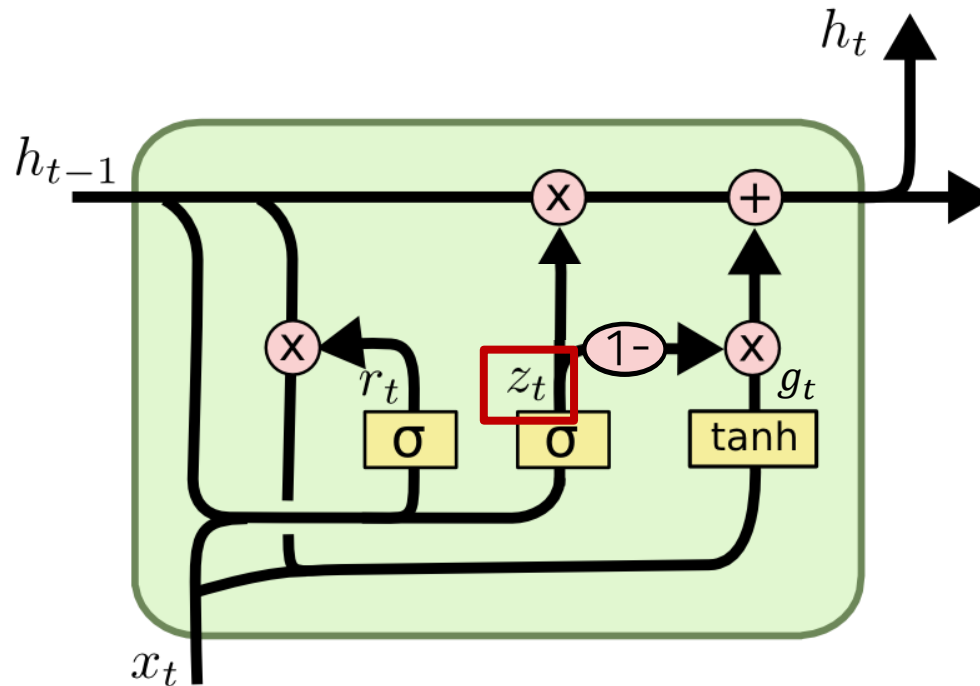
## GRU

GRU(Gated Recurrent Unit)

LSTM의 단점을 일부 보완한 형태의 모델

LSTM과 유사하지만 Cell State와 Output Gate가 존재하지 않음

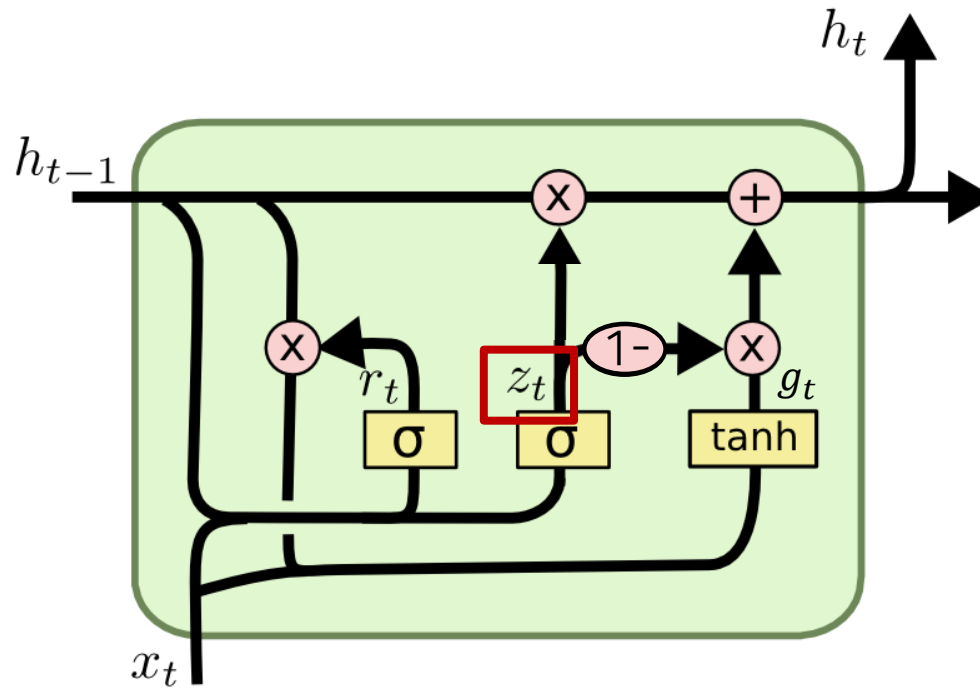
## GRU



$z_t$ 는 이전 시점의 값과 현재 입력을  
0과 1사이의 값으로 반환하며 망각/입력게이트 역할을 동시에 수행

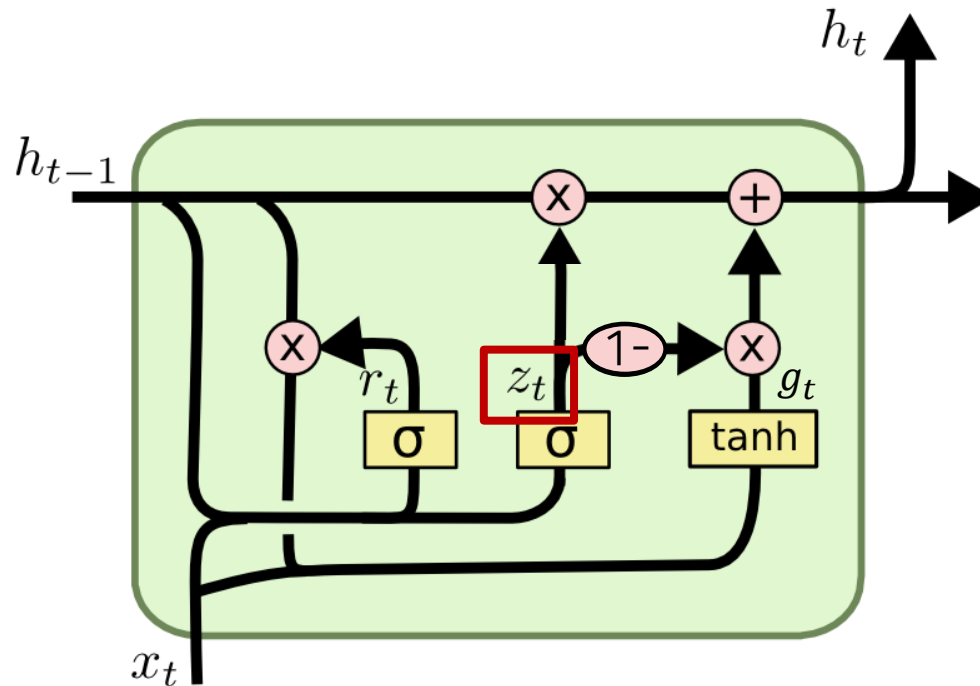


## GRU



$z_t$ 가 클수록 이전 시점의 정보  $h_{t-1}$ 을  
작을수록 현재입력  $x_t$ 를 더 많이 반영

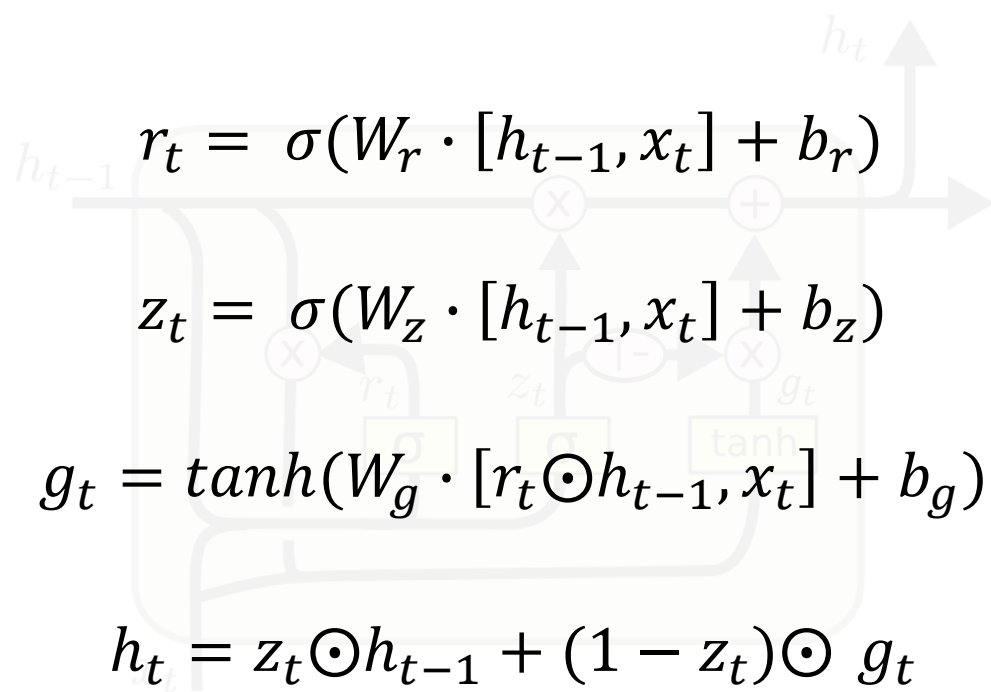
## GRU



성능의 큰 차이는 없지만 Cell State를 제거하여  
연산속도가 더 빠름

## GRU

연산과정


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

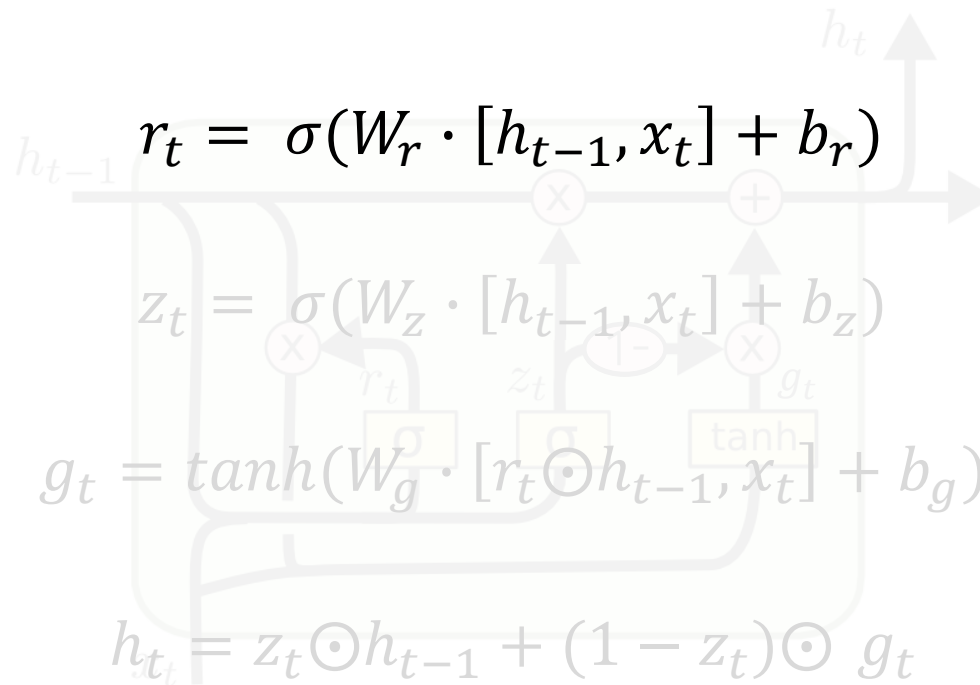
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$g_t = \tanh(W_g \cdot [r_t \odot h_{t-1}, x_t] + b_g)$$

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

## GRU

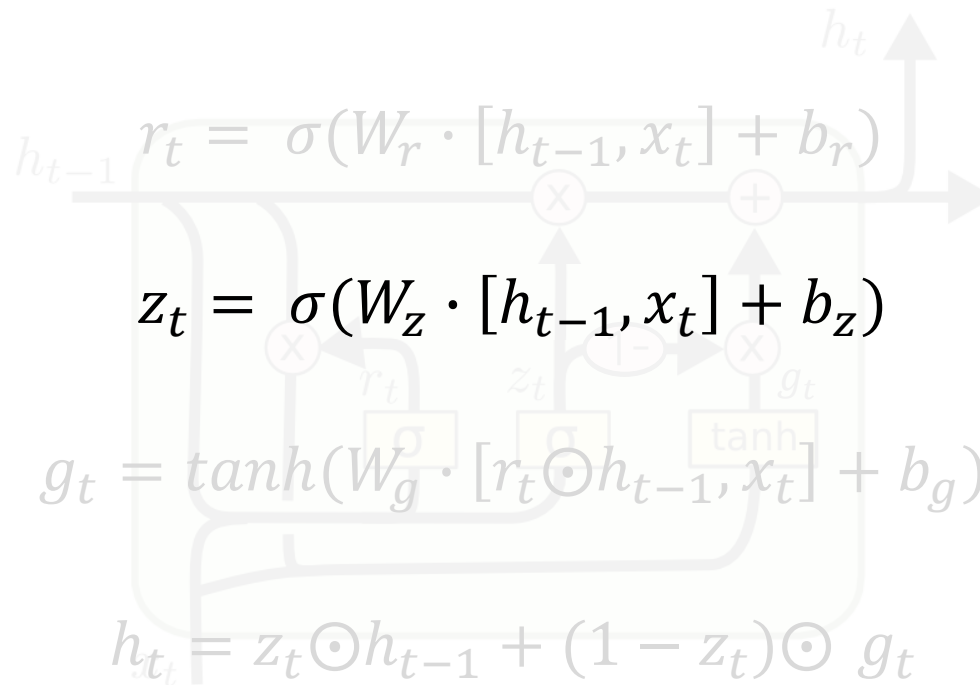
연산과정



$r_t$  는 Reset Gate로  $h_{t-1}$ 와  $x_t$ 의 가중합을 시그모이드에 통과  
이전 시점의 정보를 얼마나 유지할지 계산

## GRU

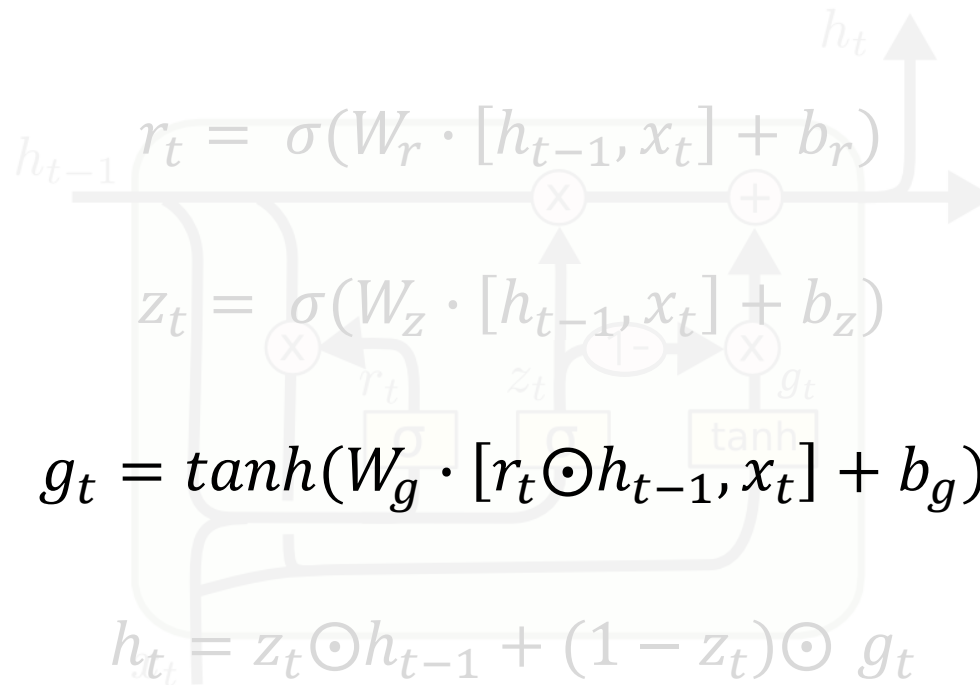
연산과정



$z_t$  는 Update Gate로, Reset gate와 계산식은 동일하지만  
이후 계산과정에서의 쓰임이 다름

## GRU

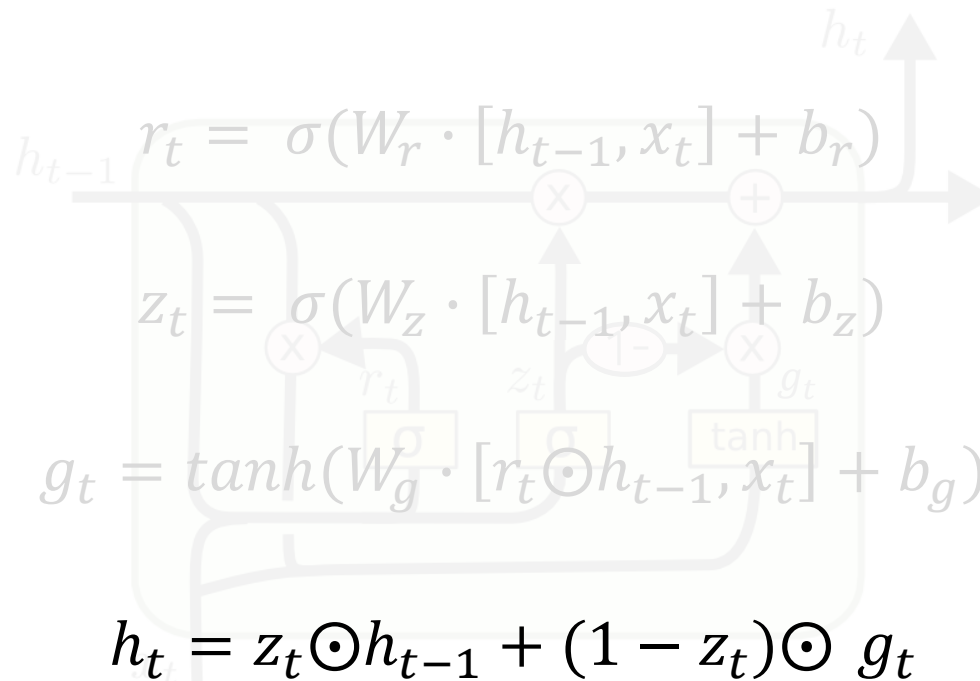
연산과정



$g_t$  는  $r_t$  를 이전 시점의 Hidden State와 곱하여  
이전 정보의 반영 비율을 정하고,  $\tanh$  함수로  $(-1, 1)$ 의 값을 반환

## GRU

연산과정



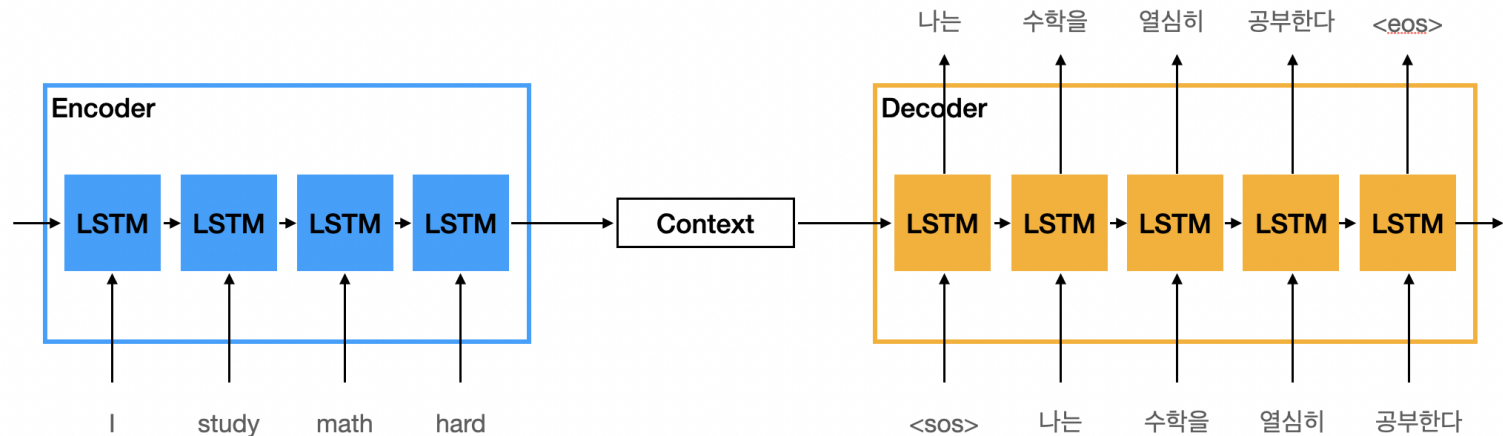
$z_t$ 를  $h_{t-1}$ 에,  $(1 - z_t)$ 를  $g_t$ 에 각각 곱하여  
현재 시점의 Hidden State를 계산

# 3

## RNN 모델의 응용



## Encoder와 Decoder



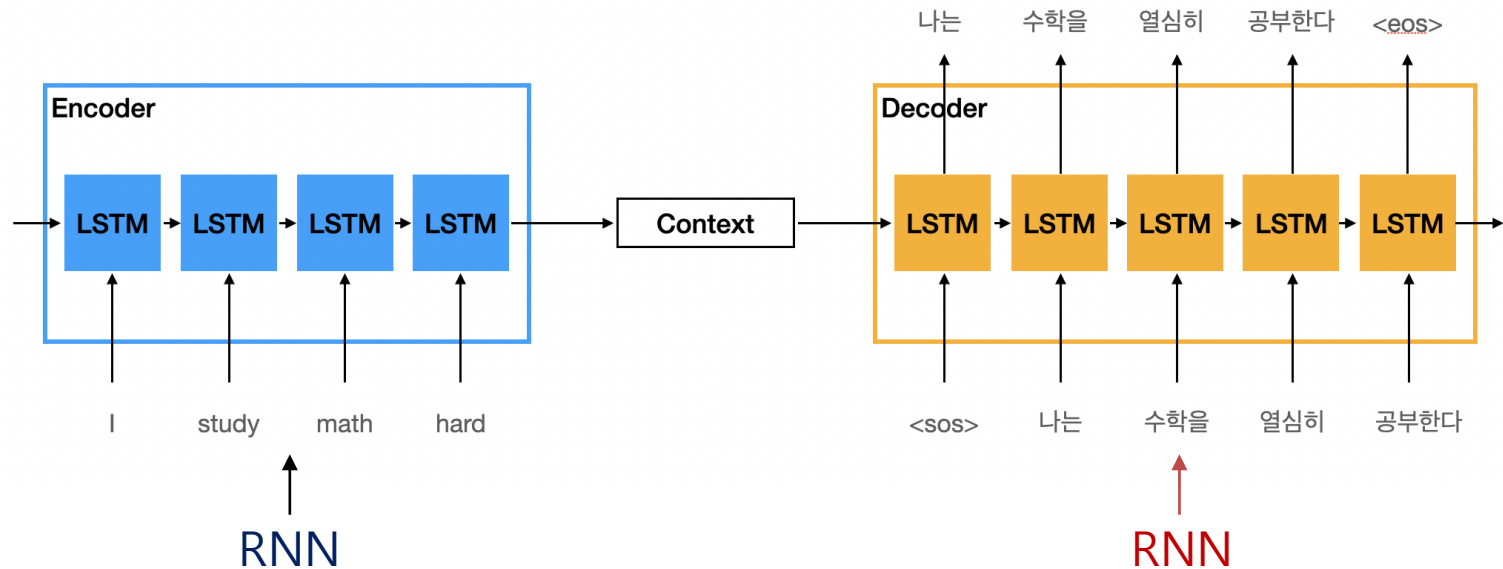
### Encoder

Input 데이터를  
압축하여 **변환**

### Decoder

압축된 데이터를  
Input과 같은 크기로 **복원**

## Encoder와 Decoder

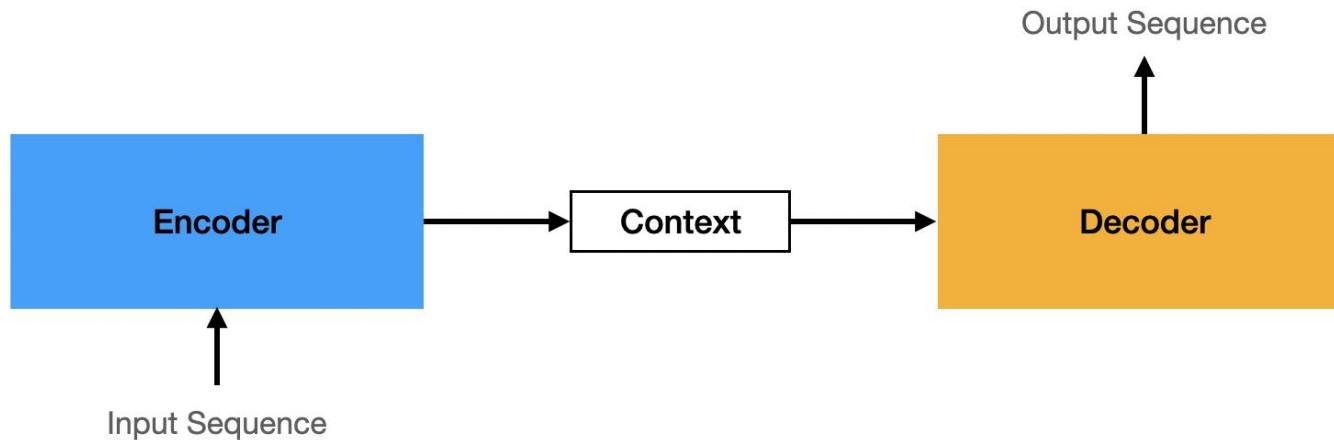


Seq2Seq

Encoder와 Decoder 각각에  
서로 다른 RNN계열 모델을 적용한 형태

## Encoder와 Decoder

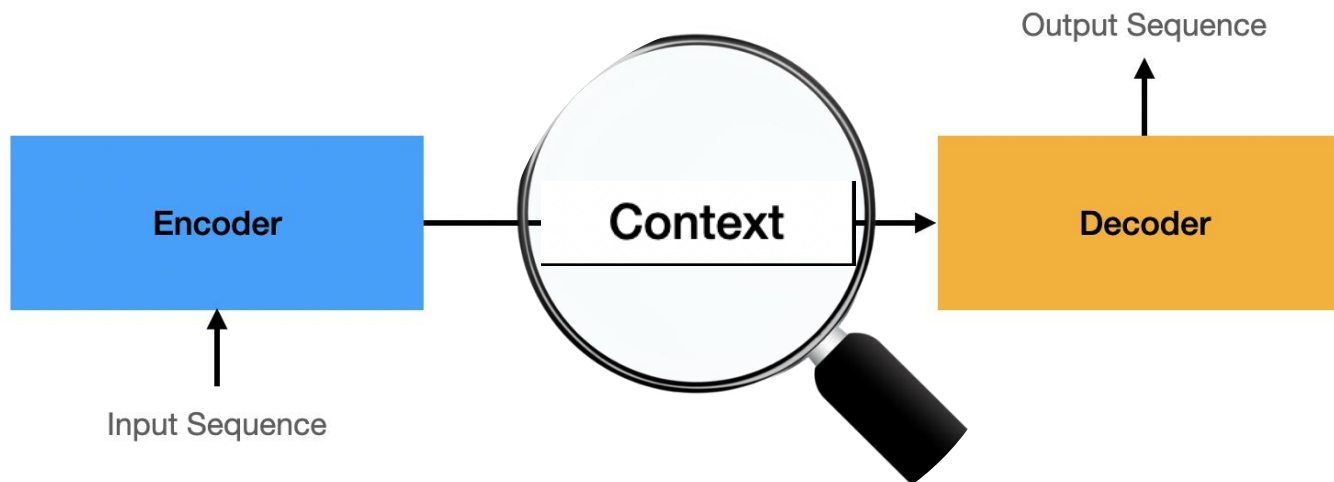
Seq2Seq



입력과 출력의 길이가 달라도 된다는 장점 덕분에  
챗봇, 번역, 내용 요약 등의 과제에서 활용됨

## Encoder와 Decoder

Seq2Seq



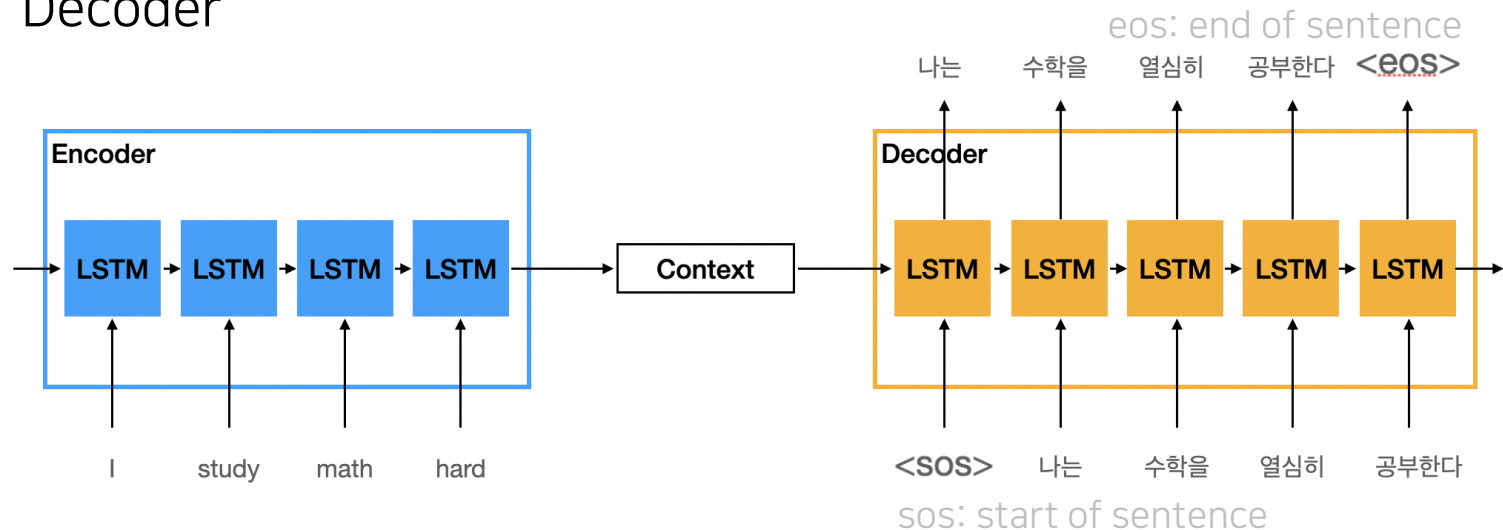
Context Vector

입력 문장을 하나의 벡터로 압축시킨 형태

Encoder의 출력이자 동시에 Decoder 의 입력

## Encoder와 Decoder

Decoder



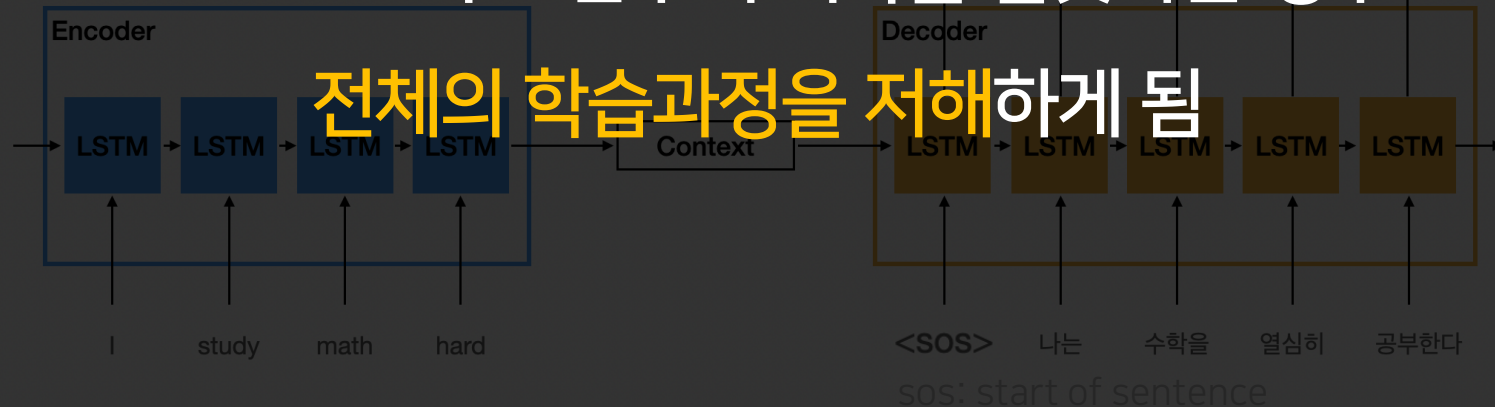
Context Vector를 입력으로 이후에는  
이전 시점의 출력을 입력으로 사용하여 **매 시점마다** 예측 단어 출력



## Encoder와 Decoder 교과강요 (Teacher Forcing)

Decoder

Decoder가 초반부에 예측을 잘못하는 경우



Context Vector를 입력으로 이후에는

이전 시점의 출력을 입력으로 사용하여 매 시점마다 예측 단어 출력



## Encoder와 Decoder (교사 강요 (Teacher Forcing))

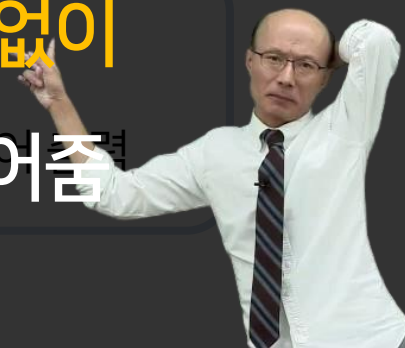
Decoder

Decoder가 초반부에 예측을 잘못하는 경우



이를 방지하기 위해 정답 여부에 관계 없이

다음 시점의 입력에는 실제 정답을 넣어줌



## Encoder-Decoder의 문제점



Encoder에서 문장을 특정크기의  
Vector로 압축하는 과정에서 **2가지 문제** 발생

### 병목현상 (Bottleneck Problem)

입력 데이터를  
고정된 길이의 벡터로 압축하여  
**정보의 손실** 발생

### 장기의존성 (long-term dependency)

Cell이 늘어날수록  
**장기의존성**  
문제가 발생할 수 있음



## Encoder-Decoder의 문제점



Encoder에서 문장을 특정크기의  
Vector로 압축하는 과정에서 **2가지 문제** 발생

### 병목현상 (Bottleneck Problem)

입력 데이터를  
고정된 길이의 벡터로 압축하여  
**정보의 손실** 발생

### 장기의존성 (long-term dependency)

Cell이 늘어날수록  
**장기의존성**  
문제가 발생할 수 있음

## Encoder-Decoder의 문제점



Encoder에서 문장을 특정크기의  
Vector로 압축하는 과정에서 **2가지 문제** 발생

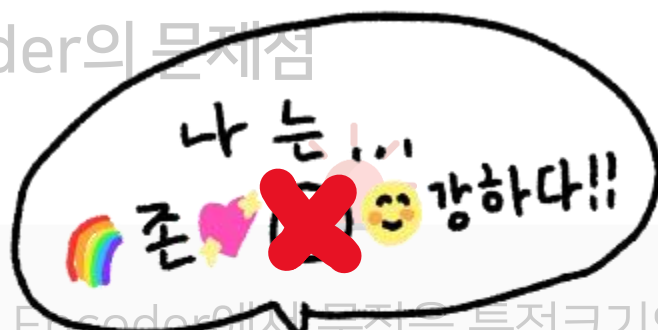
병목현상  
(Bottleneck Problem)

전체 Input 데이터를  
고정된 길이의 벡터로 압축하여  
**정보의 손실** 발생

장기의존성  
(long-term dependency)

Cell이 늘어날수록  
**장기의존성**  
문제가 발생할 수 있음

## Encoder-Decoder의 문제점



Encoder에서 문장을 특정크기의  
Vector로

문지 문제 발생

병목현상

(Bottleneck Problem)

장기의존성

long-term dependency)

전체 Input 데이터를

Cell이 늘어날수록

고정된 길이의 벡터로 압축하여

장기의존성

정보의 손실 발생

Attention이 해결해 줄게!

문제가 발생할 수 있음

## Attention

### Attention

사용자가 찾는 값과의 유사도를 바탕으로  
얼마나 그 내용에 집중해야 하는지 정하는 기법 ex) 검색 엔진



사고뭉치 남한박사



이미지

동영상

뉴스

도서

쇼핑

지도

항공편

금융

검색결과 약 29,100개 (0.23초)



예스24

<https://www.yes24.com> > Product > Goods

### 사고뭉치 북한박사

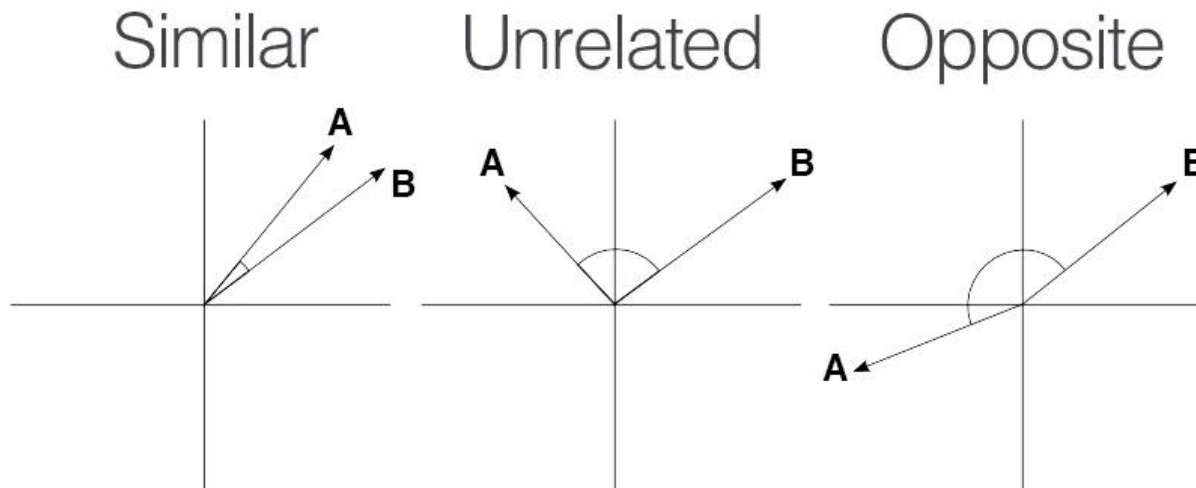
남북 분단의 현실을 피부로 느끼지 못하는 초등학생들에게 북한에서 온 철우네 가족의 모습을 재  
미난 만화를 통해 보여 주는 한편, 동시에 남한에 와서 느낀 점을 쓴 일기 ...

★★★★★ 평가: 10/10 · ₩8,550

## Attention

Dot product attention

내적을 이용하여 벡터간 유사도를 파악하는 어텐션 기법



## Dot product attention

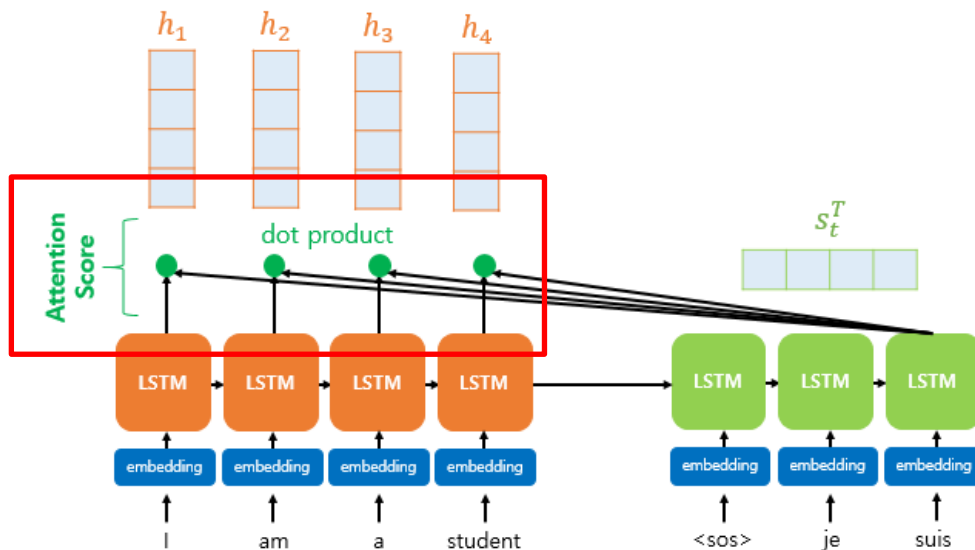
Attention Score 계산

Attention Distribution 구하기

Attention Value 구하기

Decoder의 Hidden State 연결하기 (Concatenation)

## 1. Attention Score 계산



$h_t$ : Encoder의 각 시점의 Hidden State

$s_t$ : Decoder의 Hidden State

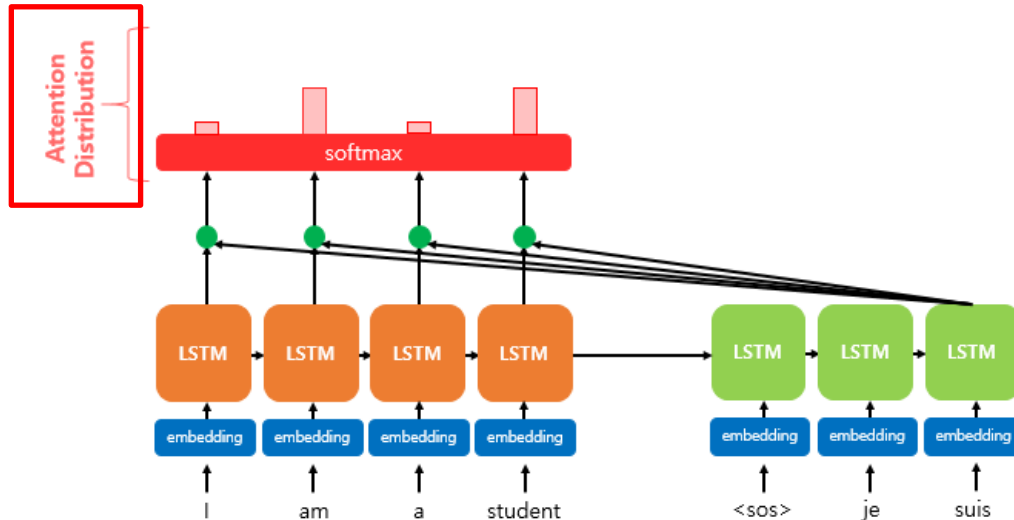
$$score(s_t, h_i) = s_t^T h_i$$

$$e^t = [s_t^T h_1, s_t^T h_2, \dots, s_t^T h_n]$$

Decoder의 출력을 반환하기 전에

Encoder의 모든 시점  $h_t$  와 현 시점의  $s_t$ 의 유사도를 계산함

## 2. Attention distribution 구하기



$e^t$  : Attention Score

$$e^t = [s_t^T h_1, s_t^T h_2, \dots, s_t^T h_n]$$

$$a_t = \text{softmax}(e^t)$$

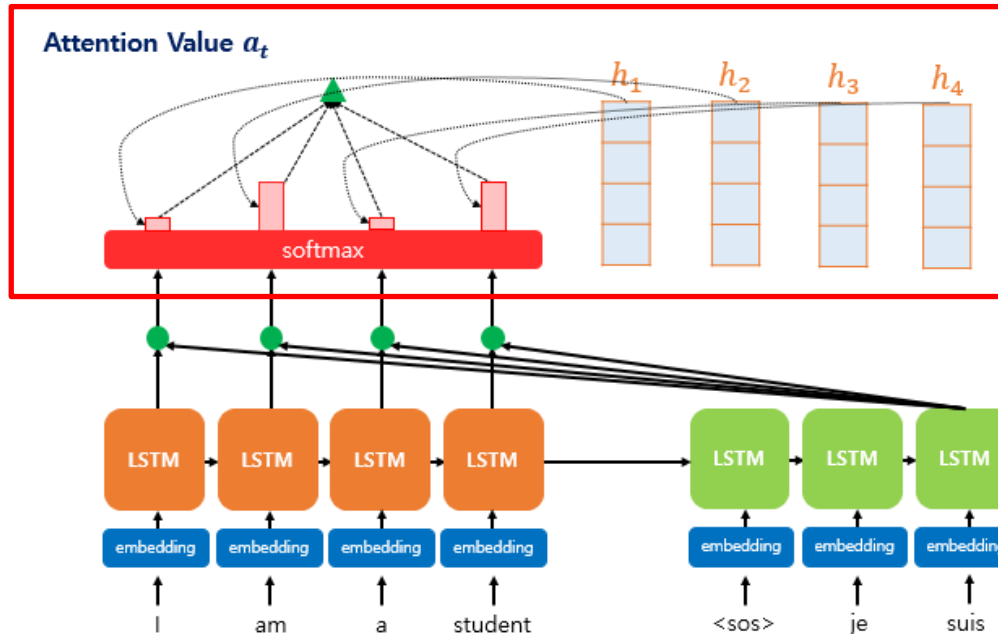
Attention Distribution

Attention Score를 Softmax 함수에 통과시켜  
0과 1 사이의 확률과 같은 형태로 처리

$s_t$ 에 대한 Hidden State의 중요도로 해석



## 3. Attention value 구하기



$h_t$ : Encoder의 각 시점의 Hidden State

$a_t = \text{softmax}(e_t)$ : Attention Distribution

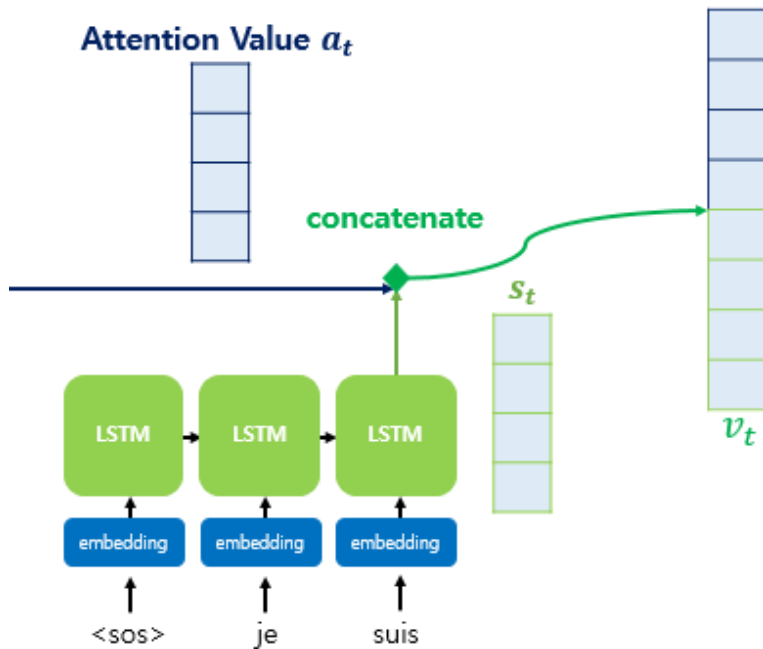
$$a^t = \sum_{i=1}^n a_i^t h_i = a^t \cdot h$$

where  $h = [h_1, \dots, h_n]$

Attention Distribution을 가중치로 하여  $h_t$ 와 **가중합**하여  
각 시점의 중요도가 반영된 **Attention value**를 계산함

Encoder의 모든  $h_t$ 를 활용함으로 병목현상 완화

## 4. Concatenation



$$\tanh \left( \begin{matrix} \text{matrix} \\ W_c \end{matrix} \times \begin{matrix} \text{vector} \\ v_t \end{matrix} \right) = \begin{matrix} \text{vector} \\ \tilde{s}_t \end{matrix}$$

$$v_t = [a_t, s_t]$$

$$\tilde{s}_t = \tanh(W_c \cdot v_t + b_c)$$

$$o_t = \text{softmax}(W_o \cdot \tilde{s}_t + b_o)$$

Attention value와 Decoder의  $s_t$ 를 결합하여  
구성한 벡터  $v_t$  를 가중치와 연산하여 출력

4

XAI

## 설명가능한 인공지능 (XAI)

### 등장배경

딥러닝은 Blackbox 모델로

예측 정확도는 높은 반면, 해석력이 약하다는 단점이 존재

### eXplainable AI

설명가능한 인공지능이라는 뜻으로

도출된 결과를 사람이 이해할 수 있게 만드는 연구 분야

### XAI의 종류

모델 특정 기법

모델 불특정 기법

## 설명가능한 인공지능 (XAI)

### 등장배경

딥러닝은 Blackbox 모델로

예측 정확도는 높은 반면, 해석력이 약하다는 단점이 존재

### eXplainable AI

설명가능한 인공지능이라는 뜻으로

도출된 결과를 사람이 이해할 수 있게 만드는 연구 분야

### XAI의 종류

모델 특정 기법

모델 불특정 기법

## 설명가능한 인공지능 (XAI)

### 등장배경

딥러닝은 Blackbox 모델로

예측 정확도는 높은 반면, 해석력이 약하다는 단점이 존재

### eXplainable AI

설명가능한 인공지능이라는 뜻으로

도출된 결과를 사람이 이해할 수 있게 만드는 연구 분야

### XAI의 종류

모델 특정 기법

모델 불특정 기법

## 설명가능한 인공지능 (XAI)

### 등장배경

딥러닝은 Blackbox 모델로

예측 정확도는 높은 반면, 해석력이 약하다는 단점이 존재

### eXplainable AI

설명가능한 인공지능이라는 뜻으로

도출된 결과를 사람이 이해할 수 있게 만드는 연구 분야

### XAI의 종류

모델 특정 기법

모델 불특정 기법

## eXplainable AI (XAI)

### 등장배경

딥러닝은 Blackbox 모델로

예측 정확도는 높은 반면, 해석력이 약하다는 단점이 존재

### eXplainable AI

설명가능한 인공지능이라는 뜻으로

도출된 결과를 사람이 이해할 수 있게 만드는 연구 분야

### XAI의 종류

모델 특정 기법

특정 모델에만 적용 가능한

Model-Specific Method 위주로 알아보자

(Model-Agnostic Method는

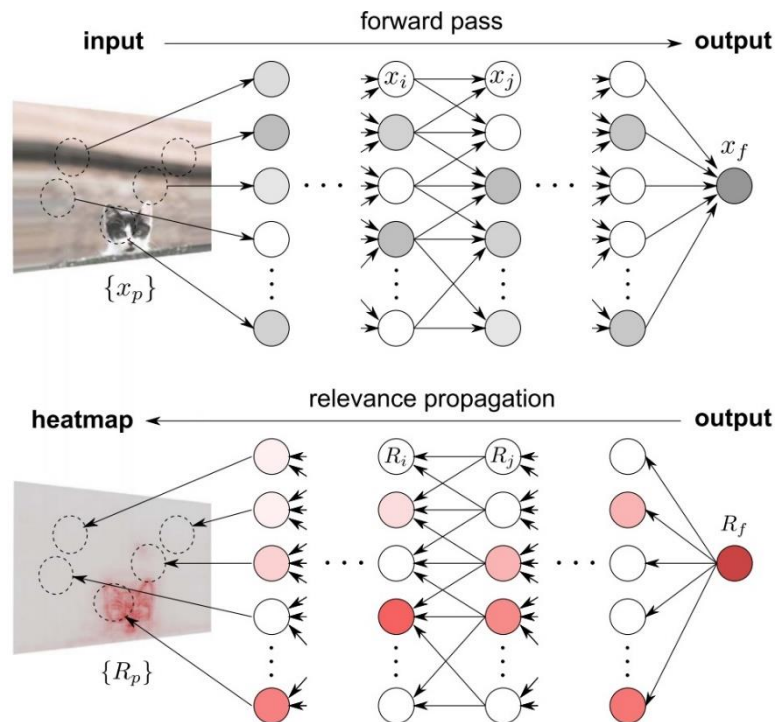
데마팀 3주차 클린업 참고)



## Layer-wise Relevance Propagation (LRP)

### LRP

모델의 결과를 역추적하여 입력 데이터의 각 Feature가 해당 결과에 기여하는 정도를 해석하는 기법



## Layer-wise Relevance Propagation (LRP)

### LRP

모델의 결과를 역추적하여 입력 데이터의 각 Feature가  
해당 결과에 기여하는 정도를 해석하는 기법

#### 타당성 전파 (Relevance Propagation)

노드 각각이  
기여도를 가진다는 가정 하에  
기여도를 입력 데이터에 전파

#### 분해 (Decomposition)

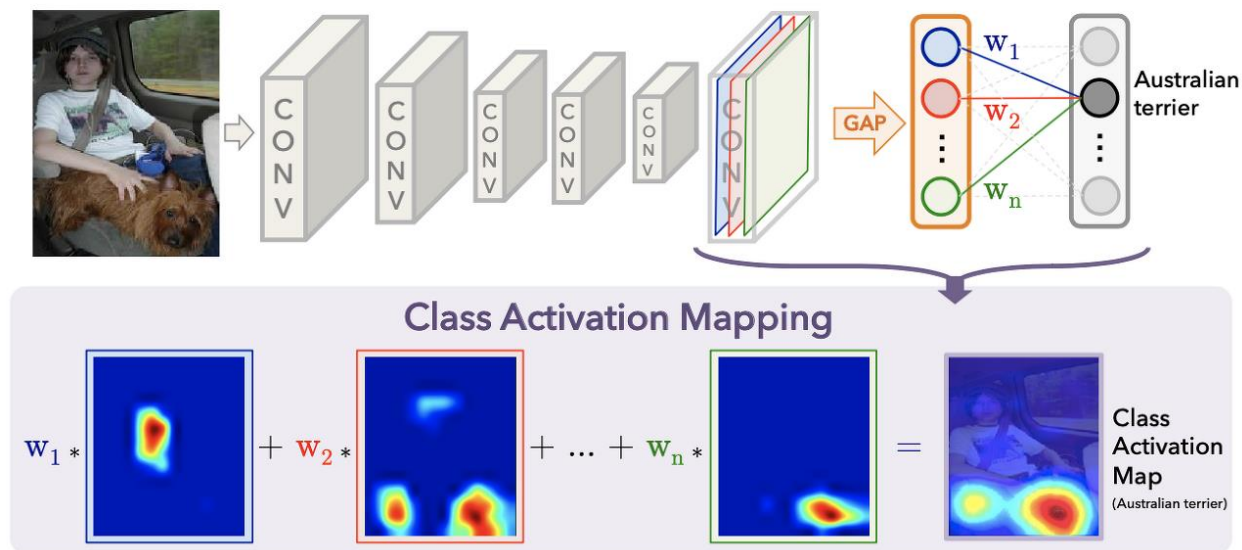
타당성 전파의 결과를  
가중치로 변환, 해부하는 것

## Class Activation Map (CAM)

CAM

CNN에 적용하는 XAI 기법으로

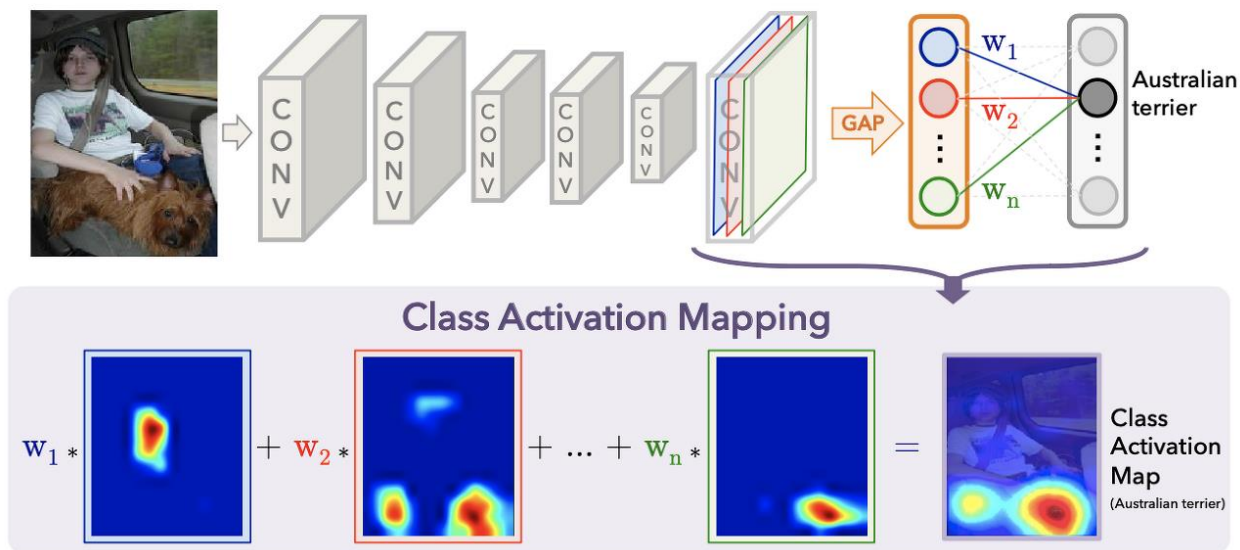
이미지의 어떤 부분이 출력에 영향을 미쳤는지 파악할 수 있음



## Class Activation Map (CAM)

CAM

마지막 Conv Layer를 FC Layer로 변환하지 않고  
Global Average Pooling (GAP)을 적용



## Class Activation Map (CAM)

CAM

마지막 Conv Layer를 FC Layer로 변환하지 않고

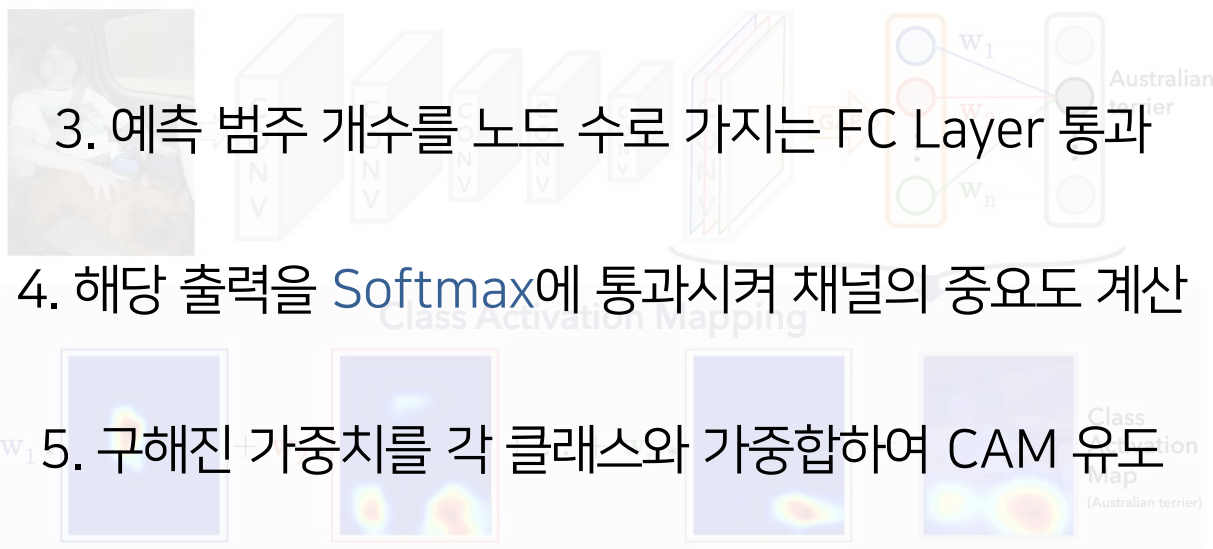
1. FC Layer와 (C, 1, 1) Feature Map을 출력하는 Conv Layer 제거

2. GAP Layer 추가하여 채널 별 정보 요약

3. 예측 범주 개수를 노드 수로 가지는 FC Layer 통과

4. 해당 출력을 Softmax에 통과시켜 채널의 중요도 계산

5. 구해진 가중치를 각 클래스와 가중합하여 CAM 유도





## Class Activation Map (CAM) 대신 Global Max Pooling 사용할 경우

특정 Object의 전반적 분포보다는

1. FC Layer와 (C, 1, 1) Feature Map을 출력하는 Conv Layer 제거

**가장 영향을 많이 미치는 특정 부분에 집중**

2. GAP Layer 추가하여 채널 별 정보 요약

3. 예측 범주 개수를 노드 수로 가지는 FC Layer 통과

4. 해당 출력을 Softmax에 통과시켜 채널의 중요도 계산

5. 구해진 가중치를 각 클래스와 가중합하여 CAM 유도



# Class Activation Map (CAM) 대신 Global Max Pooling 사용할 경우

특정 Object의 전반적 분포보다는

1. FC Layer와 (C, 1, 1) Feature Map을 출력하는 Conv Layer 제거  
가장 영향을 많이 미치는 특정 부분에 집중

2. GAP Layer 추가하여 채널 별 정보 요약

3. 예측 범주 개수를 노드 개수로 가지는 FC Layer 통과

4. 해당 출력을 Softmax에 통과시켜 채널의 중요도를 산출  
목적에 따라 잘 선택해야겠음

5. 구해진 가중치를 각 클래스와 가중합하여 CAM

잘 듣고 잘 골라봐

선생님의 말을 잘 듣고 잘 골라보세요!





THANK YOU

