

# RNN

## (Recurrent Neural Network)

hreeee@yonsei.ac.kr  
강사 백혜림

# RNN(Recurrent Neural Network)

- RNN은 시퀀스 모델이다. 입력과 출력을 시퀀스 단위로 처리하는 모델

- 시퀀스와 시퀀스 벡트

시퀀스(sequence)는 순서가 있는 나열 또는 그러한 상태를 의미, 자연어에서는 구문 문장 문단등의 의미를 가진 것을 말함.

시퀀스 벡터는 이러한 순서를 보존하여 벡터 공간에 나타낸 것을 의미

“나는 밥을 먹었습니다.” → “나는”, “ ”, “밥을”, “ ”, “먹었”, “습니다”, “.” → [1, 0, 2, 0, 3, 4, 5] → [1, 0, 2, 0, 3, 4, 5]

“맛있게 먹었습니다.” → “맛”, “있게”, “ ”, “먹었”, “ ”, “습니다”, “.” → [6, 7, 0, 3, 4, 5] → [6, 7, 0, 3, 4, 5, 0]

나는 밥을 ( )는다.

나는 밥을 ( )는다.

나는 밥을 ( )는다.

나는 밥을 ( )는다.

# Recurrent Neural Network (RNN)

인공지능이 예측하기 위해선 요소 간의 연관성이 있어야 한다.  
딥러닝이 말하는 시퀀스 데이터는 순차적인 특징을 필수로 가짐

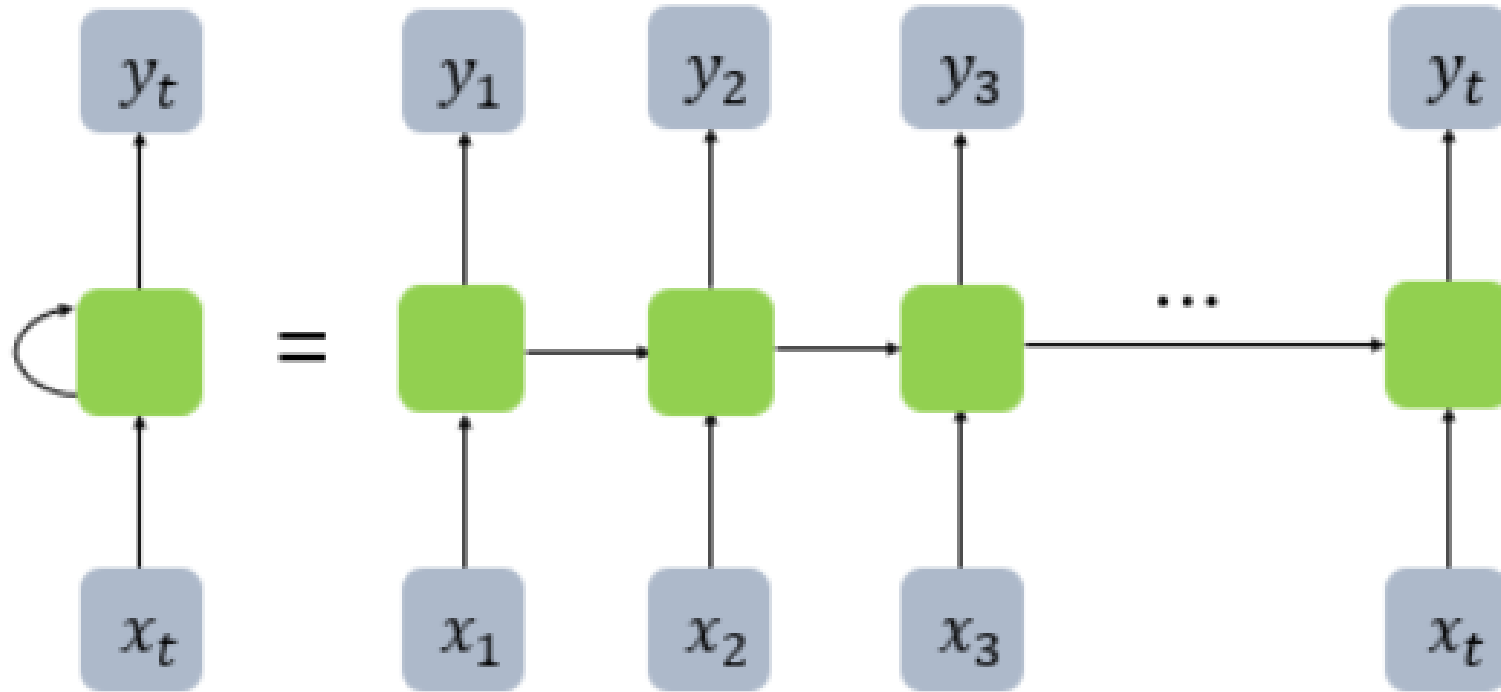
# RNN(Recurrent Neural Network)



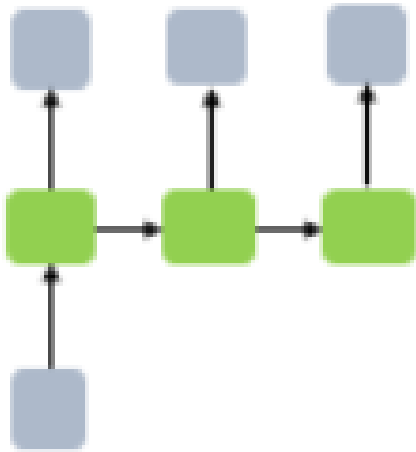
메모리 셀  
RNN 셀



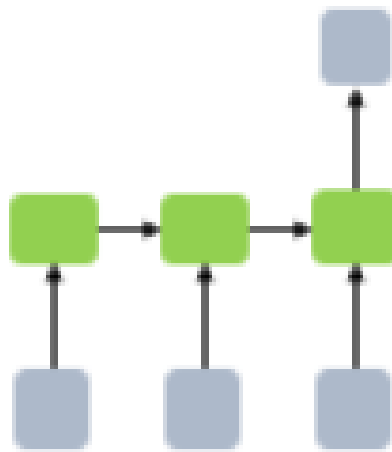
# RNN(Recurrent Neural Network)



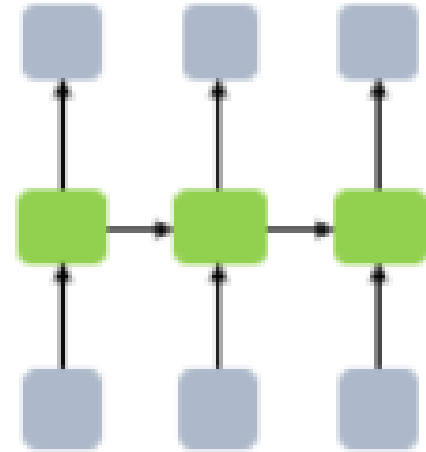
# RNN(Recurrent Neural Network)



일 대 다(one-to-many)



다 대 일(many-to-one)

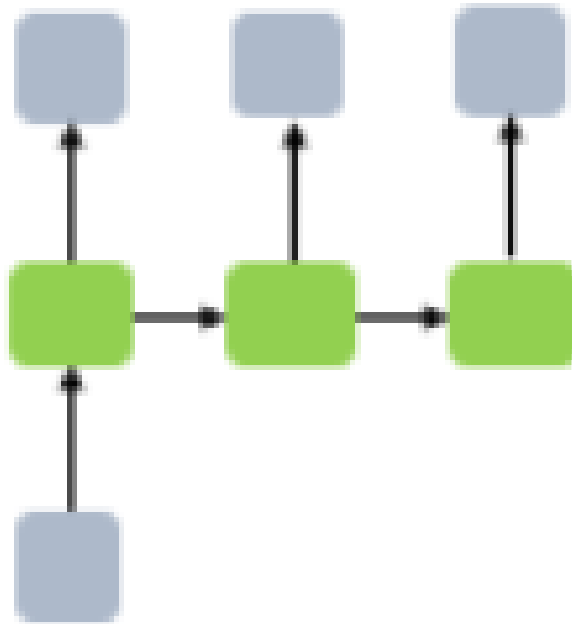


다 대 다(many-to-many)

# RNN(Recurrent Neural Network)

## one to many

- 하나의 입력에 대해서 여러 개의 출력 ex) 이미지 캡셔닝(image captioning)



일 대 다(one-to-many)

# RNN(Recurrent Neural Network)

## many to one

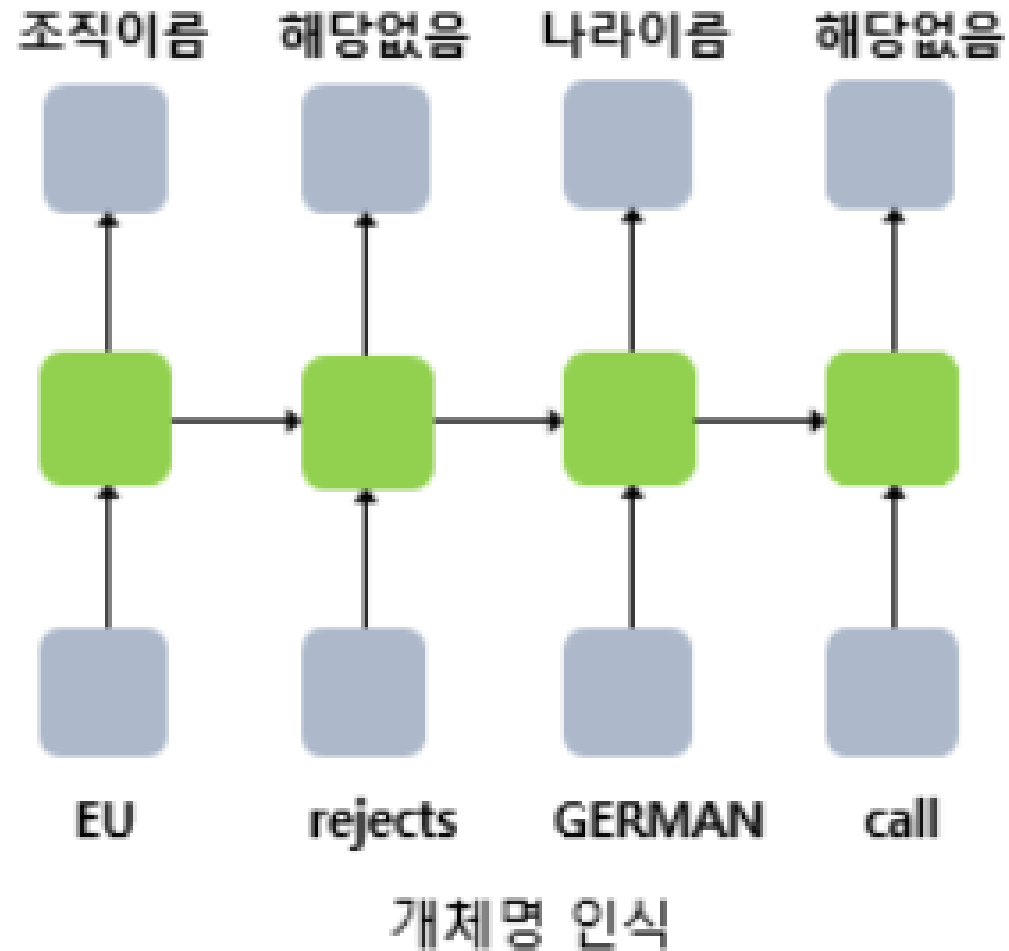
- 단어 시퀀스에 대해서 하나의 출력
- ex) 감성 분류(sentiment classification), 스팸 메일 분류(spam detection)



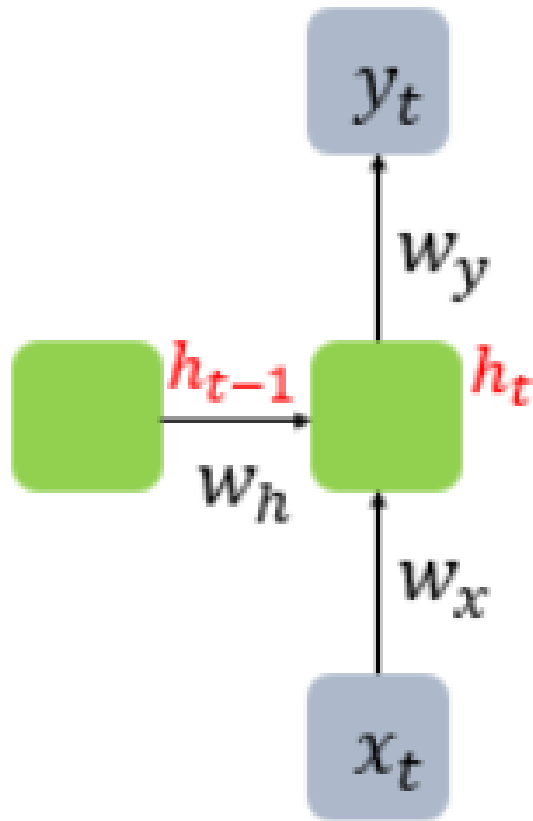
# RNN(Recurrent Neural Network)

## many to many

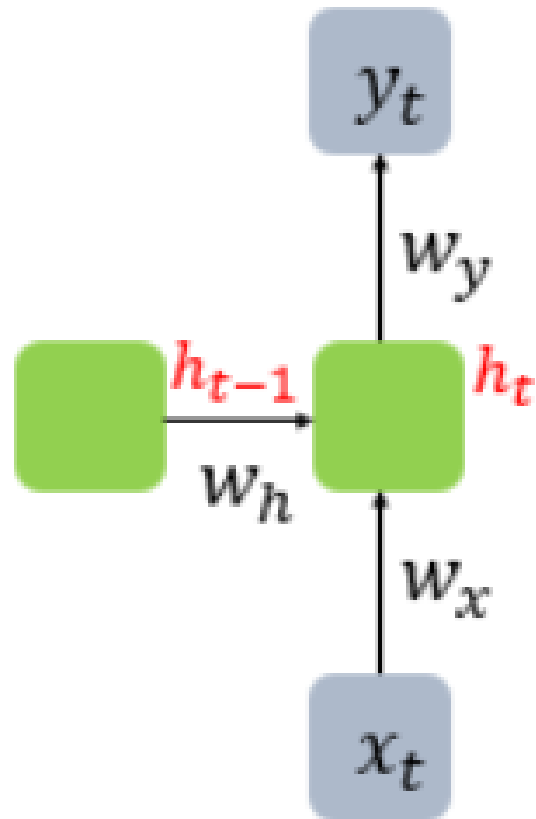
- 입력 문장으로부터 대답 문장을 출력 ex) 챗봇, 번역기, 개체명인식기, 품사태깅등



# RNN(Recurrent Neural Network)



# RNN(Recurrent Neural Network)

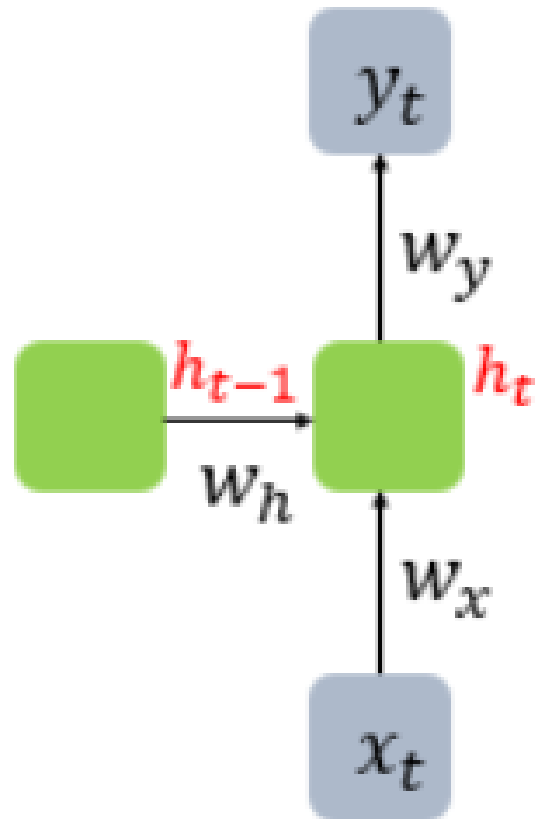


은닉층 :  $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$

출력층 :  $y_t = f(W_y h_t + b)$

단,  $f$ 는 비선형 활성화 함수 중 하나.

# RNN(Recurrent Neural Network)



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

출력층 :  $y_t = f(W_y h_t + b)$

단,  $f$ 는 비선형 활성화 함수 중 하나.

$x_t : (d \times 1)$

$W_x : (D_h \times d)$

$W_h : (D_h \times D_h)$

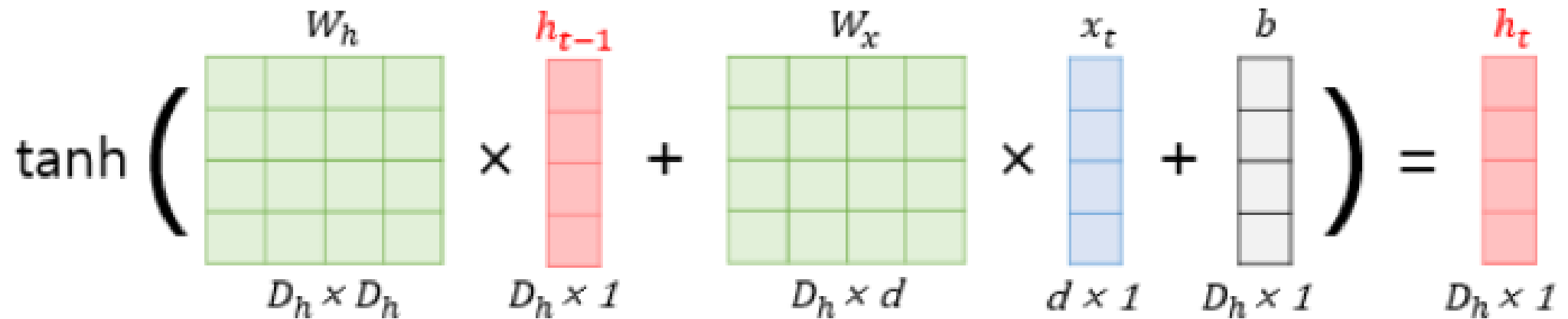
$h_{t-1} : (D_h \times 1)$

$b : (D_h \times 1)$



# RNN(Recurrent Neural Network)

배치 크기가 1이고,  $d$ 와  $D_h$  두 값 모두를 4로 가정하였을 때, RNN의 은닉층 연산을 그림으로 표현하면 아래와 같습니다.



The diagram illustrates the hidden layer operation of an RNN. It shows the calculation of the current hidden state  $h_t$  based on the previous hidden state  $h_{t-1}$  and the current input  $x_t$ . The operation is defined as:

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

Where the dimensions of the matrices and vectors are as follows:

- $W_h$ : Hidden-to-hidden weight matrix, dimension  $D_h \times D_h$  (represented as a 4x4 green grid).
- $h_{t-1}$ : Previous hidden state vector, dimension  $D_h \times 1$  (represented as a 4x1 red column).
- $W_x$ : Hidden-to-input weight matrix, dimension  $D_h \times d$  (represented as a 4x4 green grid).
- $x_t$ : Current input vector, dimension  $d \times 1$  (represented as a 4x1 blue column).
- $b$ : Bias vector, dimension  $D_h \times 1$  (represented as a 4x1 gray column).
- $h_t$ : Current hidden state vector, dimension  $D_h \times 1$  (represented as a 4x1 red column).

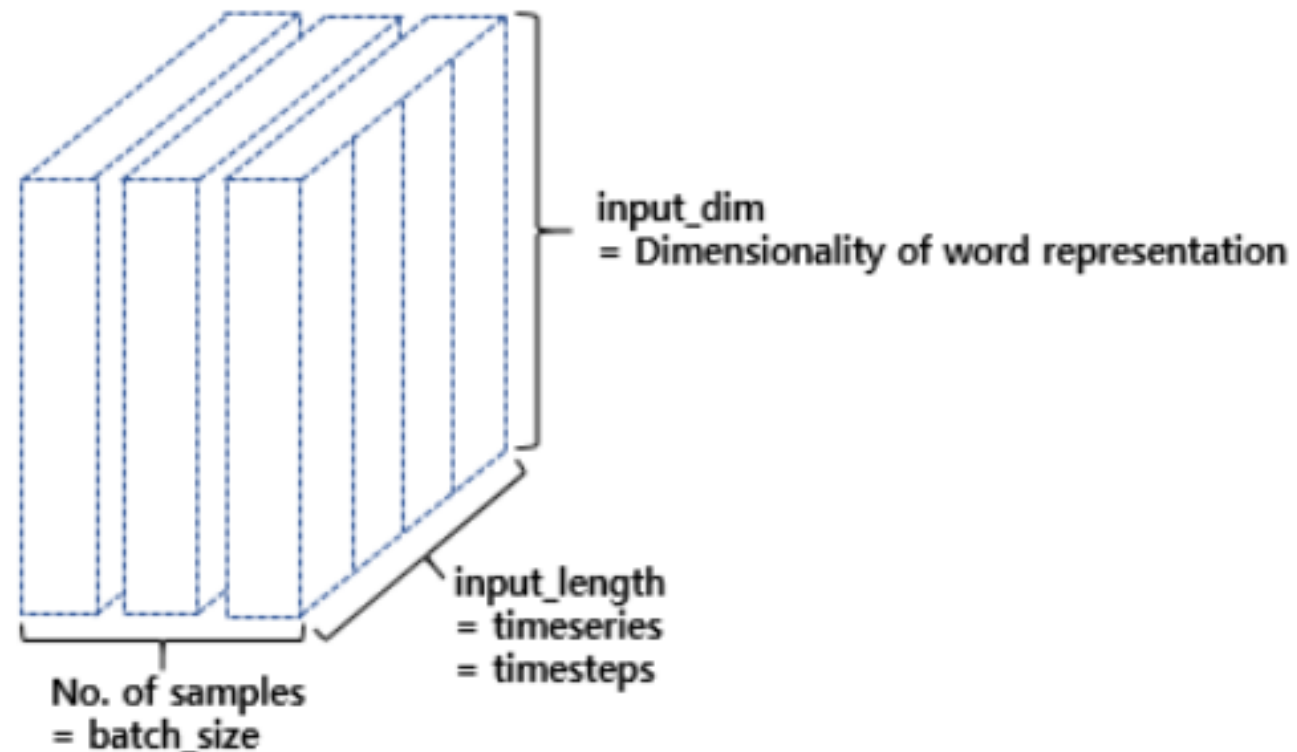
# Keras로 RNN구현하기

## - 케라스로 RNN층을 추가하는 코드

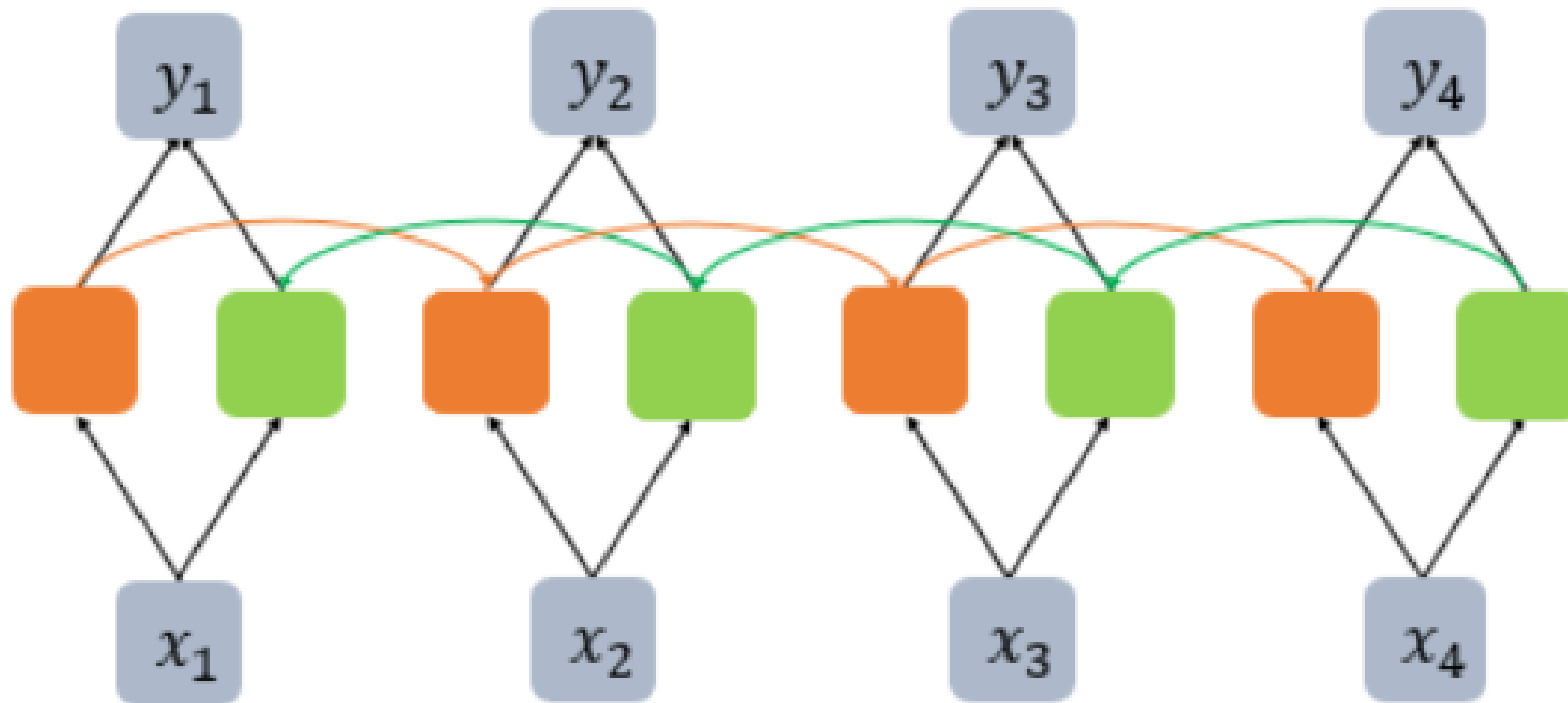
```
# RNN 층을 추가하는 코드.  
model.add(SimpleRNN(hidden_size)) # 7
```

```
# 추가 인자를 사용할 때  
model.add(SimpleRNN(hidden_size, input_shape=(timesteps, input_dim)))
```

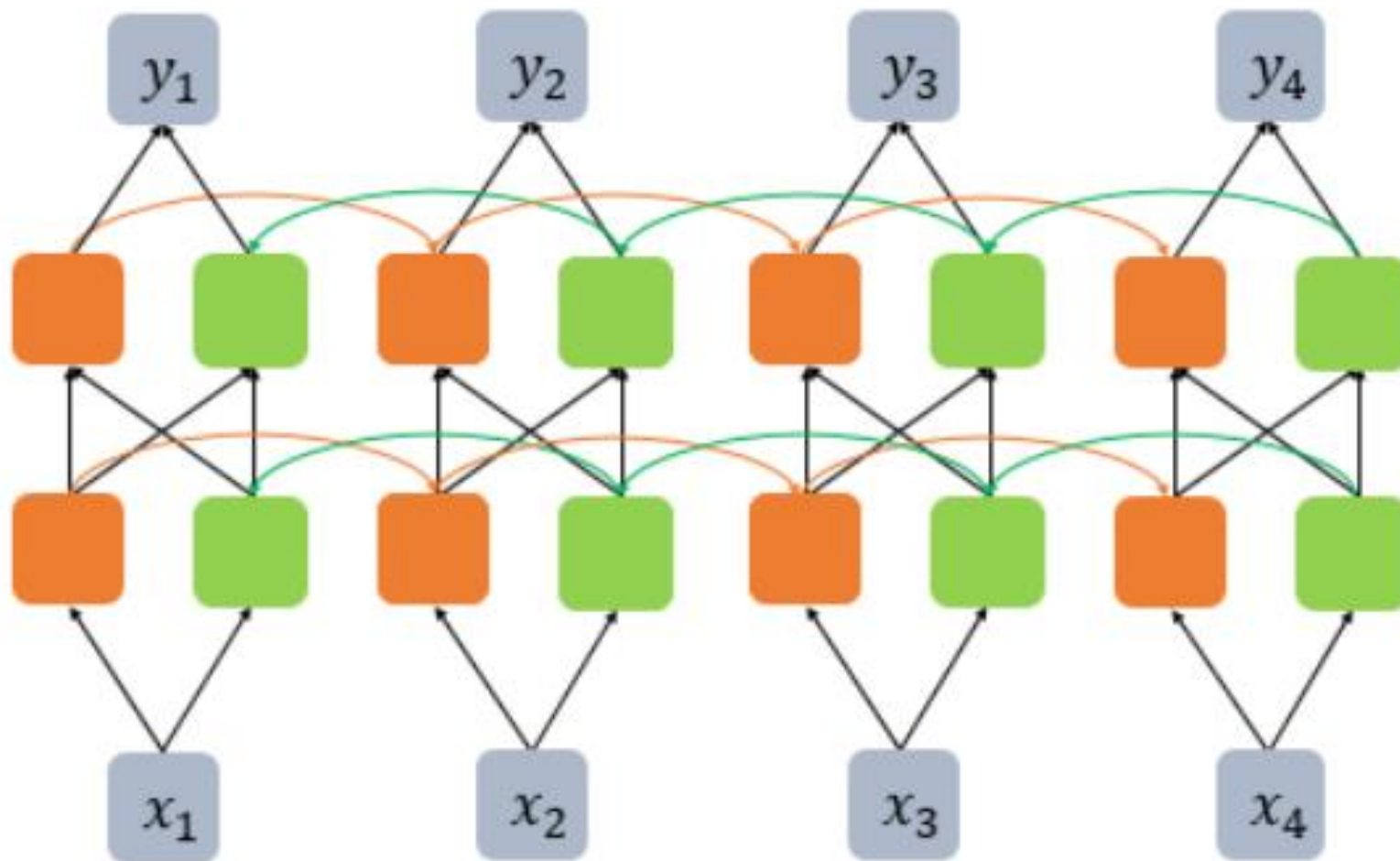
```
# 다른 표기  
model.add(SimpleRNN(hidden_size, input_length=M, input_dim=N))  
# 단, M과 N은 정수
```

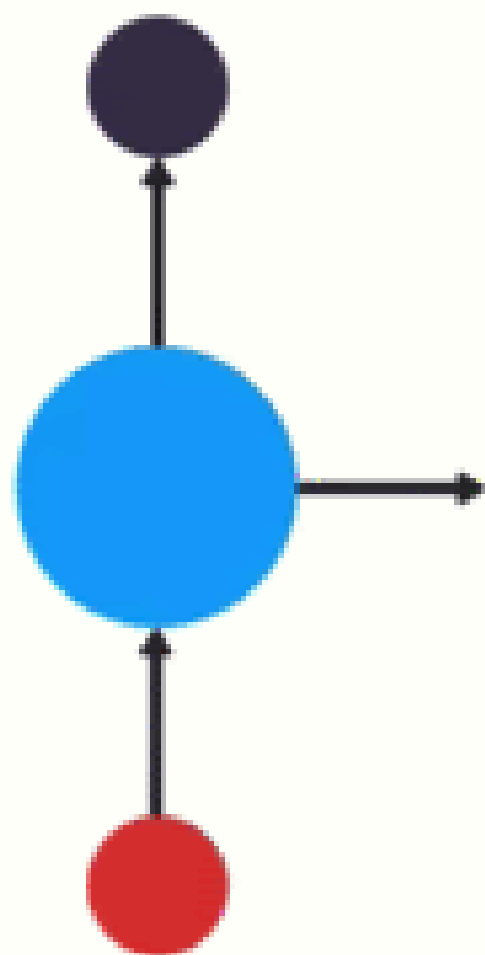


# Bi-RNN (Bidirectional Recurrent Neural Network)



# Deep Bi-RNN (Bidirectional Recurrent Neural Network)

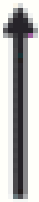
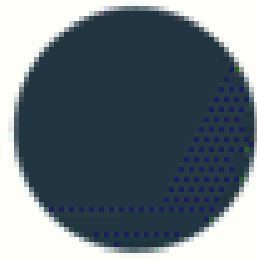




What time is it?

What time is it ?

O1



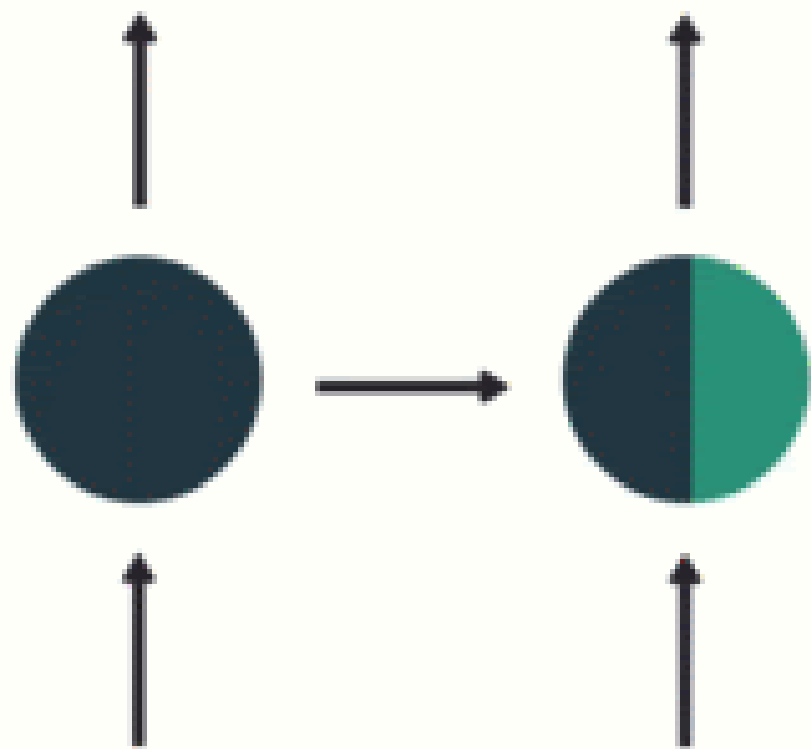
What time is it ?





O1

O2



What

time

is

it

?

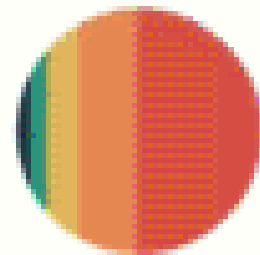
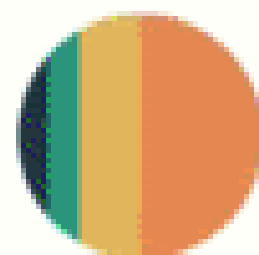
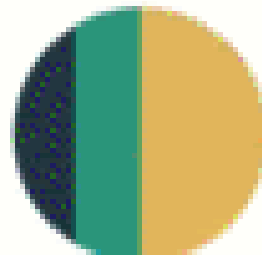
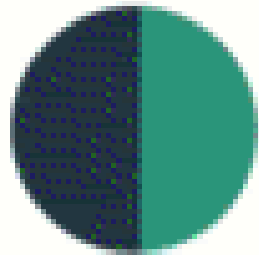
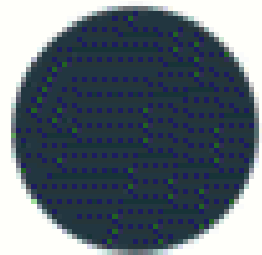
O1

O2

O3

O4

O5



What

time

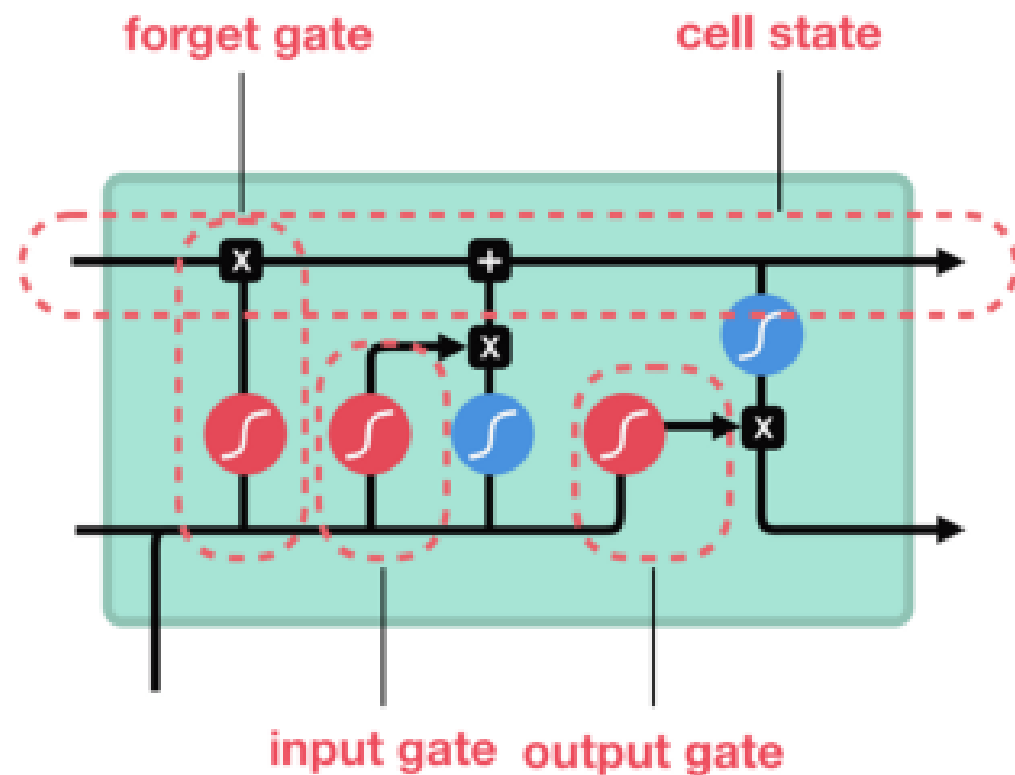
is

it

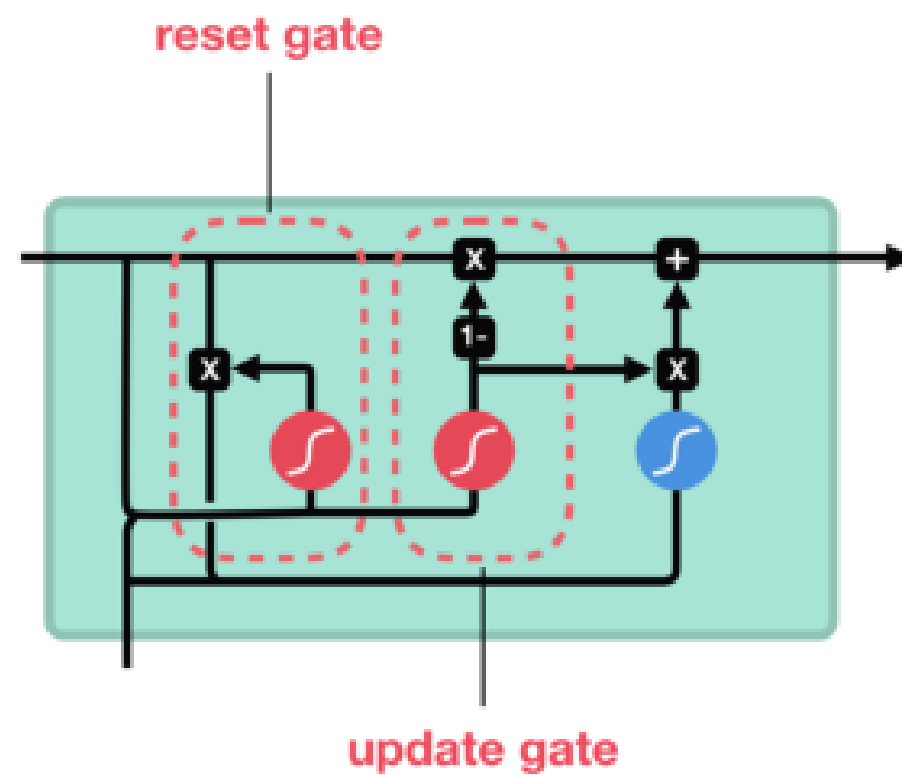
?

Embedding 벡터의 차원수의 크기가 동일할 경우(노드에서는 units=64), Weight의 크기가 위에서 사용했던 SimpleRNN의 4배나 되는 것을 볼 수 있는데, 왜 이런 RNN 레이어가 등장하게 된 것일까요?

## LSTM



## GRU



sigmoid



tanh



pointwise  
multiplication



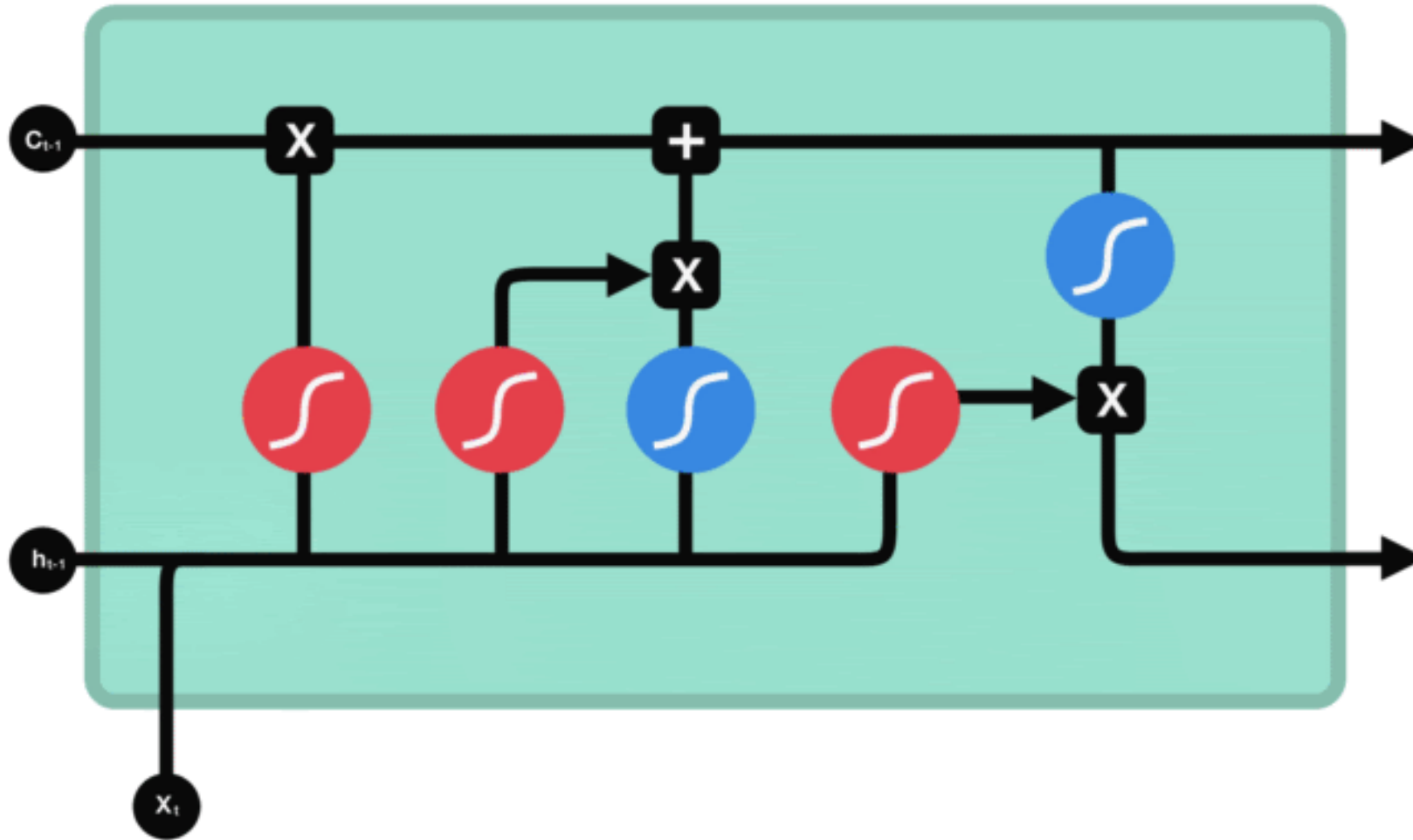
pointwise  
addition



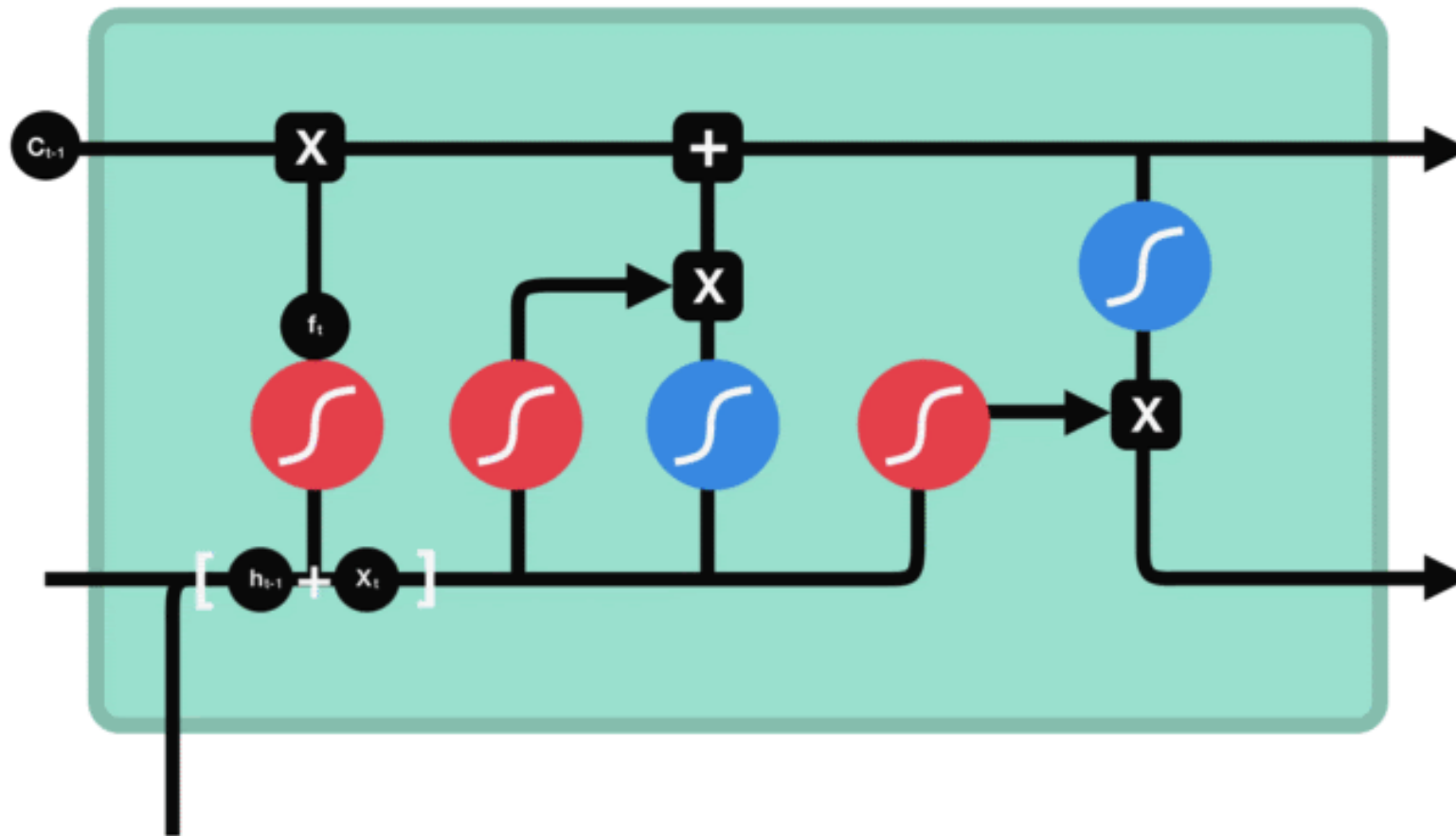
vector  
concatenation

# Forget Gate

$c_{t-1}$  previous cell state  
 $f_t$  forget gate output

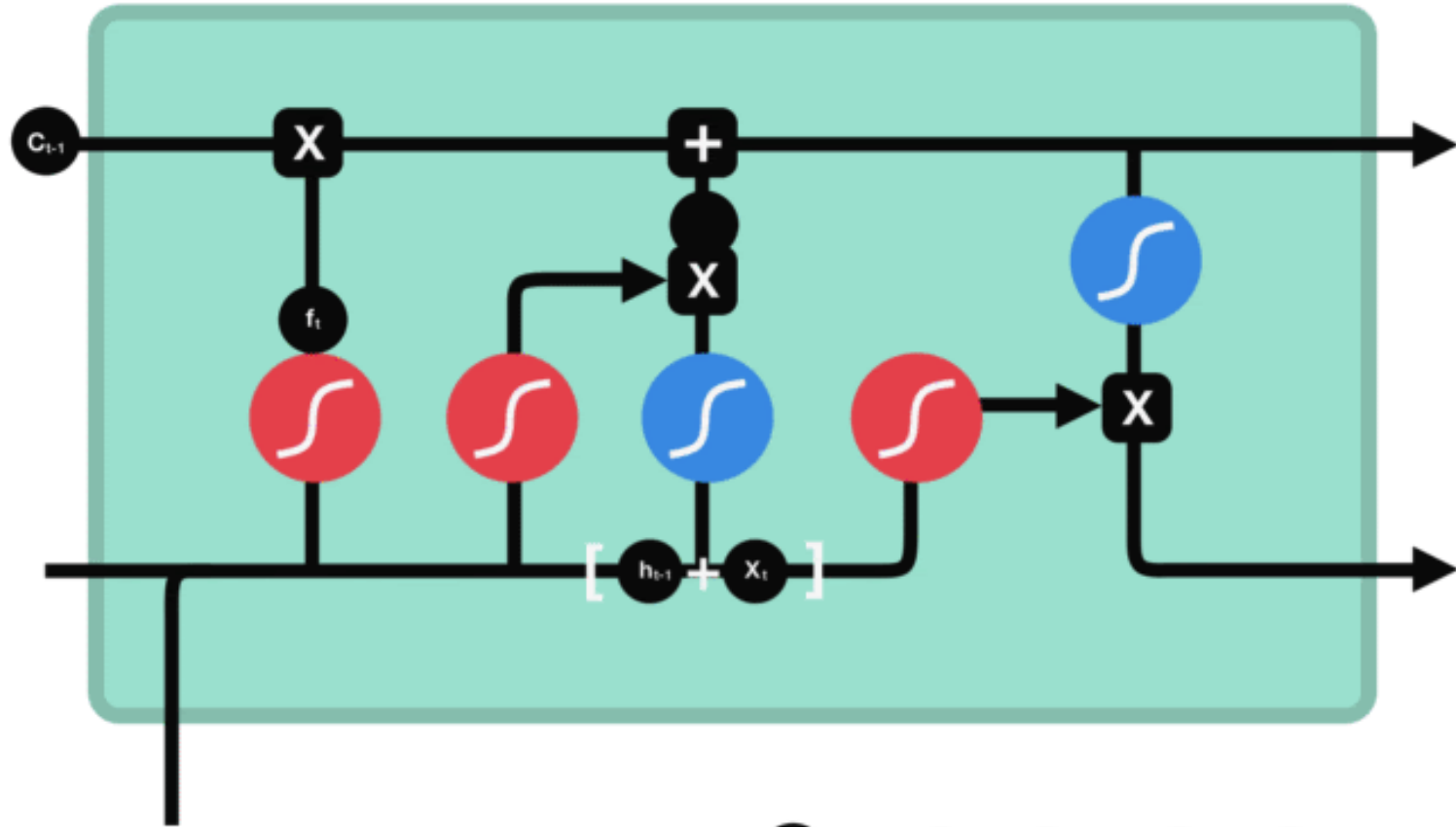


# Input Gate



- $c_{t-1}$  previous cell state
- $f_t$  forget gate output
- $i_t$  input gate output
- $\tilde{c}_t$  candidate

# Cell State

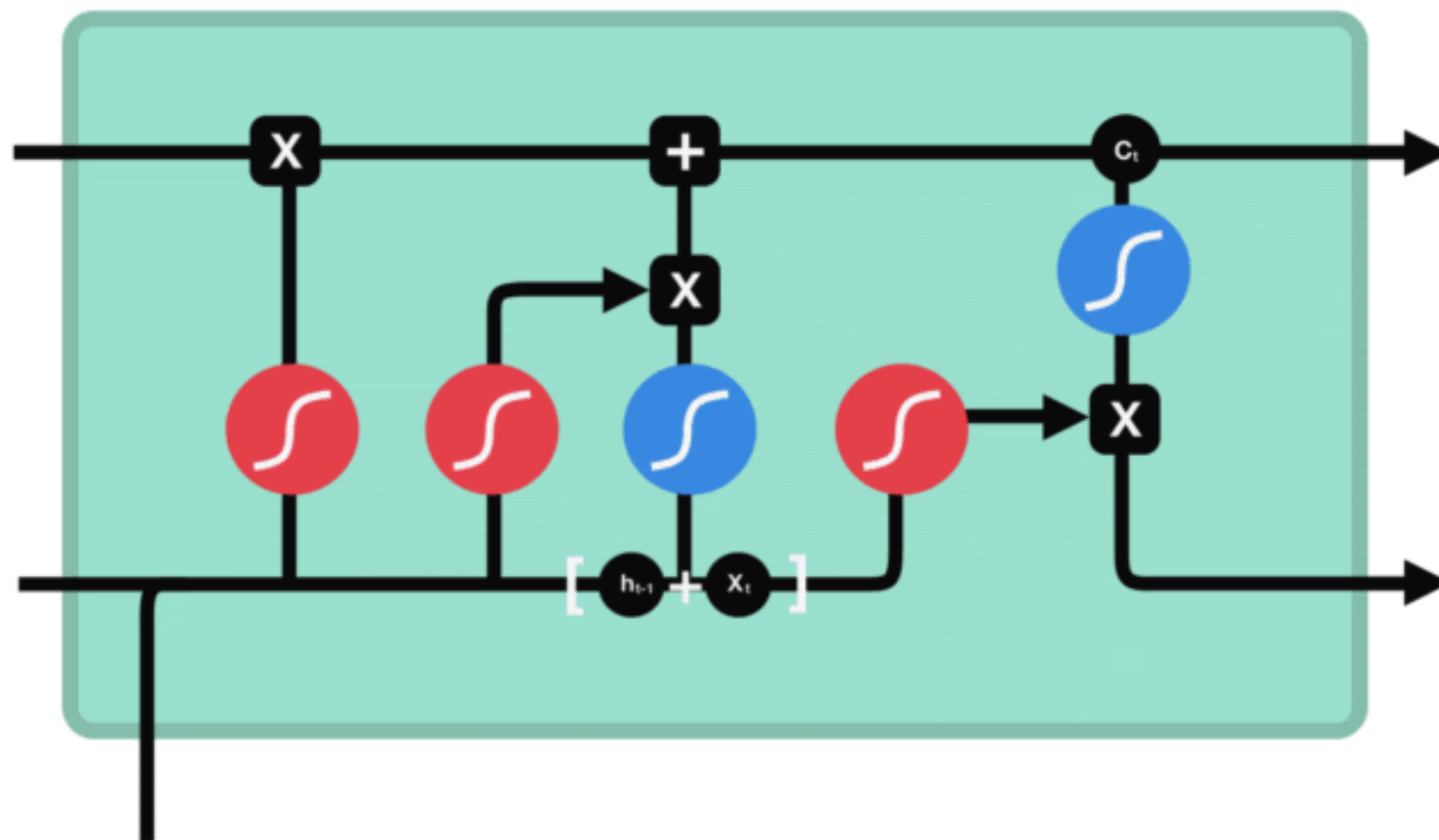


- $C_{t-1}$  previous cell state
- $f_t$  forget gate output
- $i_t$  input gate output
- $\tilde{C}_t$  candidate
- $C_t$  new cell state

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



# Output Gate



- $c_{t-1}$  previous cell state
- $f_t$  forget gate output
- $i_t$  input gate output
- $\tilde{c}_t$  candidate
- $c_t$  new cell state
- $o_t$  output gate output
- $h_t$  hidden state

GRU가 LSTM보다 학습할 데이터가  
적을까요? 많을까요? 적다고 생각하면 왜  
그렇게 생각하나요?

GRU와 LSTM은 비슷한 성능을  
보이는데도, GRU의 장점으로 꼽히는  
부분은 무엇인가요?

GRU와 LSTM은 활용한  
어플리케이션들은 뭐가 있을까요?