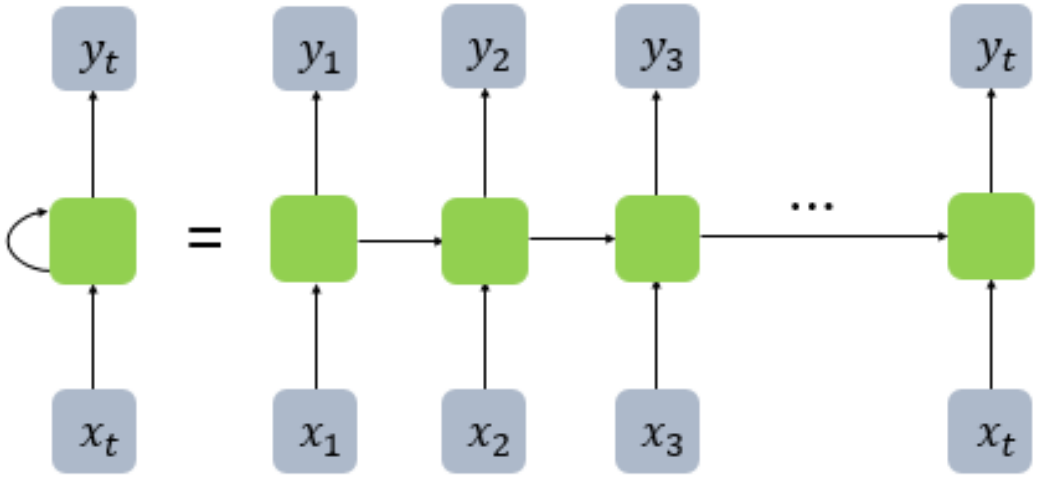
09. 순환 신경망(Recurrent Neural Network)

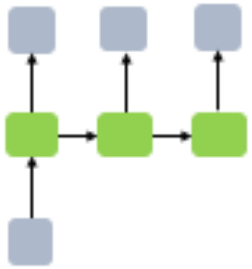
1) 순환 신경망(Recurrent Neural Network, RNN)

은닉층의 노드에서 활성화 함수를 통해 나온 결과값을 출력층 방향으로도 보내면서, 다시 은닉층 노드의 다음 계산의 입력으로 보냄

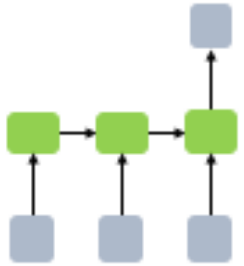


- **메모리 셀:** 이전의 값을 기억하려고 하는 일종의 메모리 역할을 수행
- **은닉 상태(hidden state):** 메모리 셀이 출력층 방향으로 또는 다음 시점 t+1의 자신에게 보내는 값

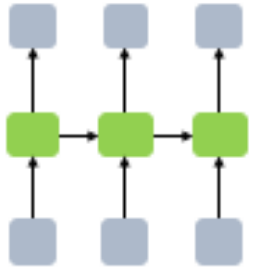
➔ FFNN은 현재의 input으로 그 다음 결정을 내림. 반면 RNN은 현재의 input은 물론 과거의 input을 반복적으로 (recurrently) 사용



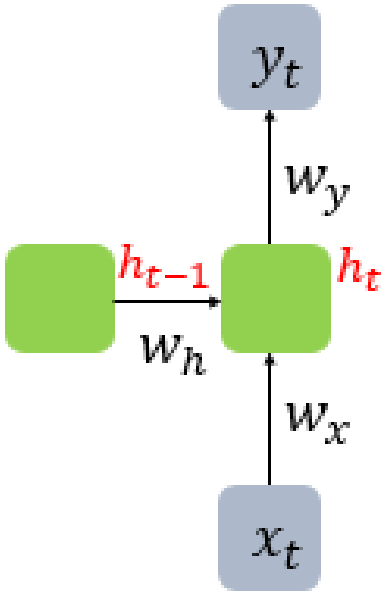
일 대 다(one-to-many)



다 대 일(many-to-one)



다 대 다(many-to-many)



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

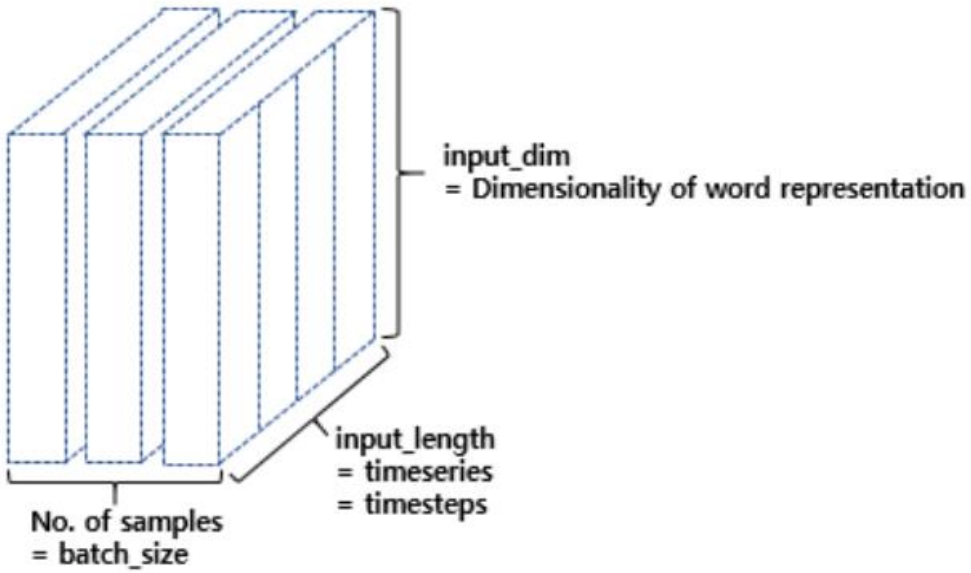
출력층 : $y_t = f(W_y h_t + b)$

단, f는 비선형 활성화 함수 중 하나.

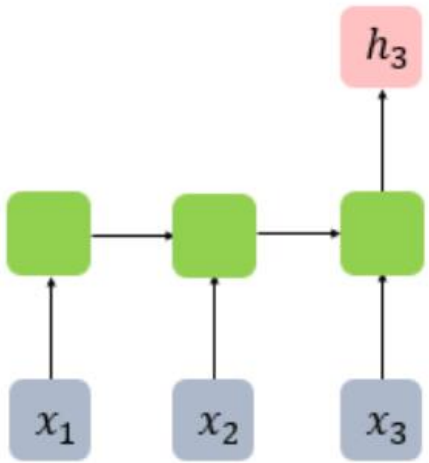
$$\tanh\left(\underset{D_h \times D_h}{W_h} \times \underset{D_h \times 1}{h_{t-1}} + \underset{D_h \times d}{W_x} \times \underset{d \times 1}{x_t} + \underset{D_h \times 1}{b}\right) = \underset{D_h \times 1}{h_t}$$

```
# 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
```

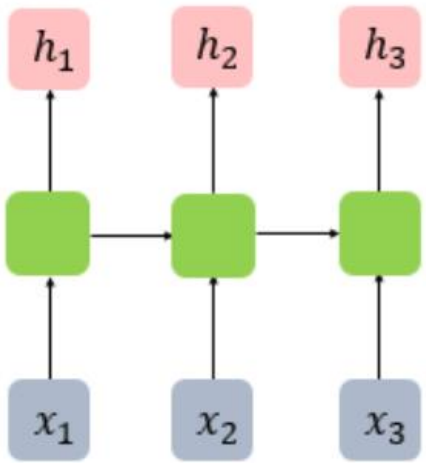


- **hidden_size** = 메모리 셀이 다음 시점의 메모리 셀과 출력층으로 보내는 값의 크기 (output_dim)
- **timesteps** = 입력 시퀀스의 길이(input_length)라고 표현하기도 함. 시점의 수.
- **input_dim** = 입력의 크기.



다음층으로 마지막 은닉 상태만 전달

return_sequences=True



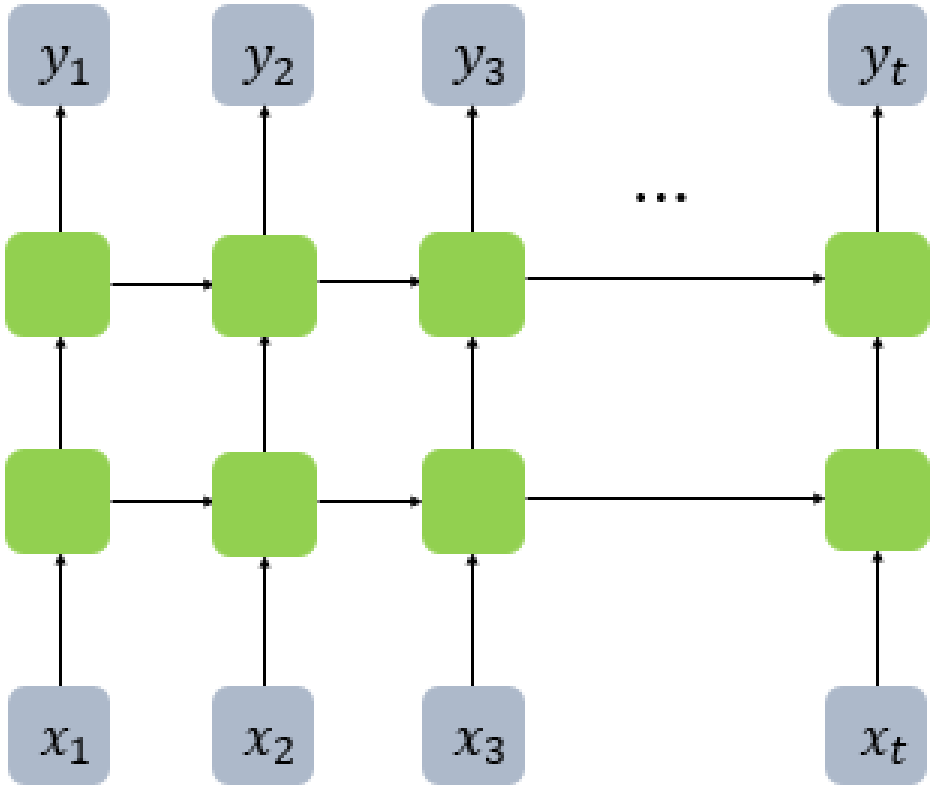
다음층으로 모든 은닉 상태 전달

```
from keras.models import Sequential
from keras.layers import SimpleRNN

model = Sequential()
model.add(SimpleRNN(3, batch_input_shape=(8,2,10), return_sequences=True))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_3 (SimpleRNN)     (8, 2, 3)                 42
=================================================================
Total params: 42
Trainable params: 42
Non-trainable params: 0
_________________________________________________________________
```

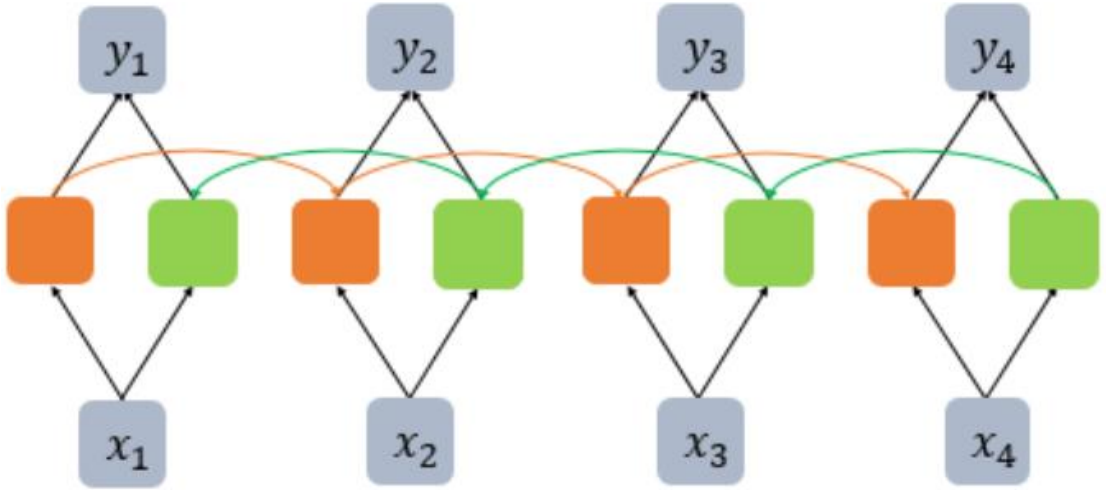
깊은 순환 신경망(Deep Recurrent Neural Network)




```
model = Sequential()
model.add(SimpleRNN(hidden_size, return_sequences = True))
model.add(SimpleRNN(hidden_size, return_sequences = True))
```

양방향 순환 신경망(Bidirectional Recurrent Neural Network)

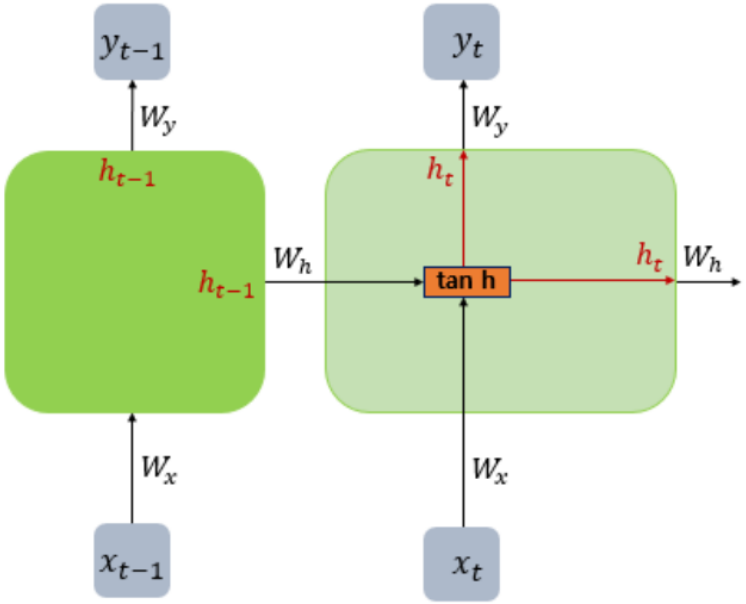
- 이전 시점의 데이터뿐만 아니라, 이후 시점의 데이터도 힌트로 활용하기 위해서 고안된 것



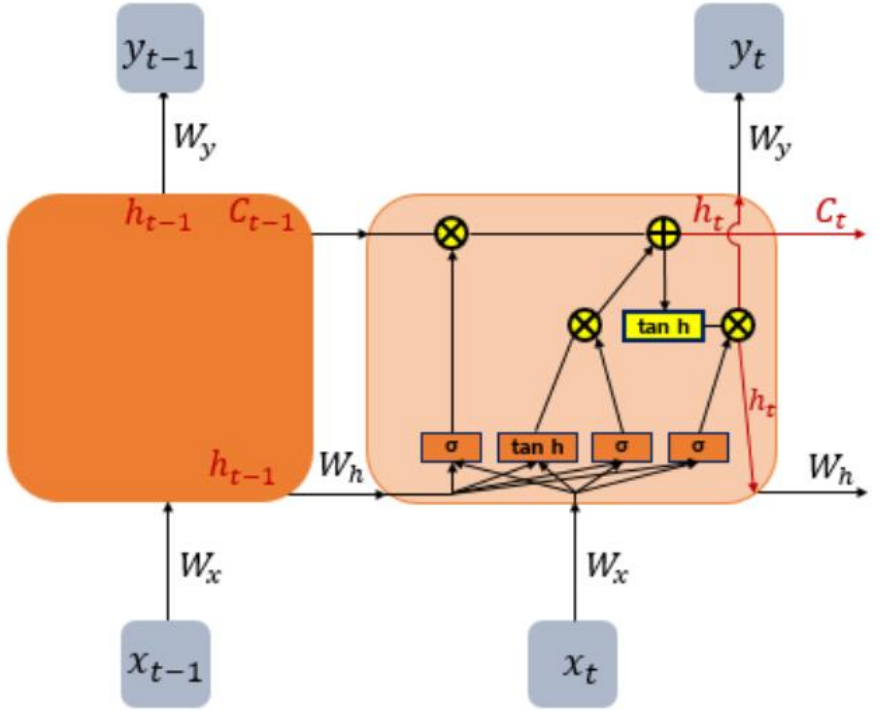

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Bidirectional

model = Sequential()
model.add(Bidirectional(SimpleRNN(hidden_size, return_sequences = True), input_shape=(timesteps, input_dim)))
```

2) 장단기 메모리(Long Short-Term Memory, LSTM)

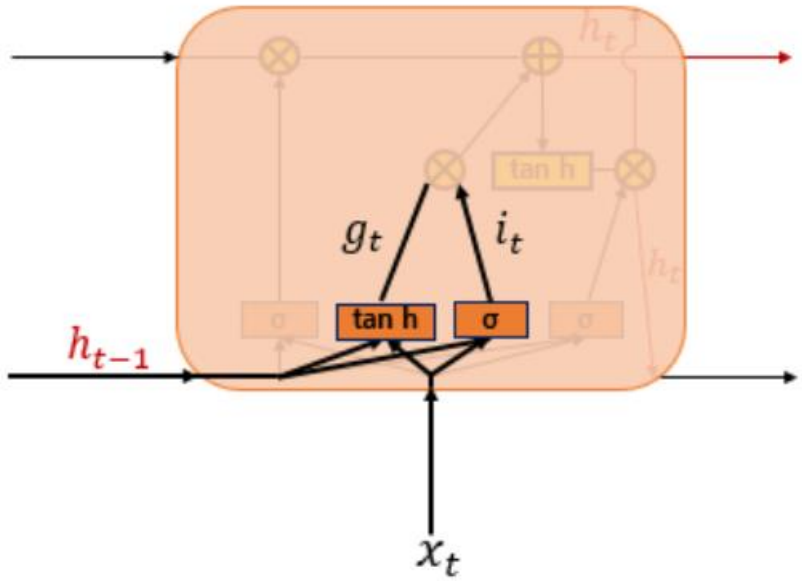


장기 의존성 문제(the problem of Long-Term Dependencies)



은닉층의 메모리 셀에 입력 게이트, 망각 게이트, 출력 게이트를 추가하여 불필요한 기억을 지우고, 기억해야 할 것들을 정함 (각 게이트에 시그모이드 함수 존재)

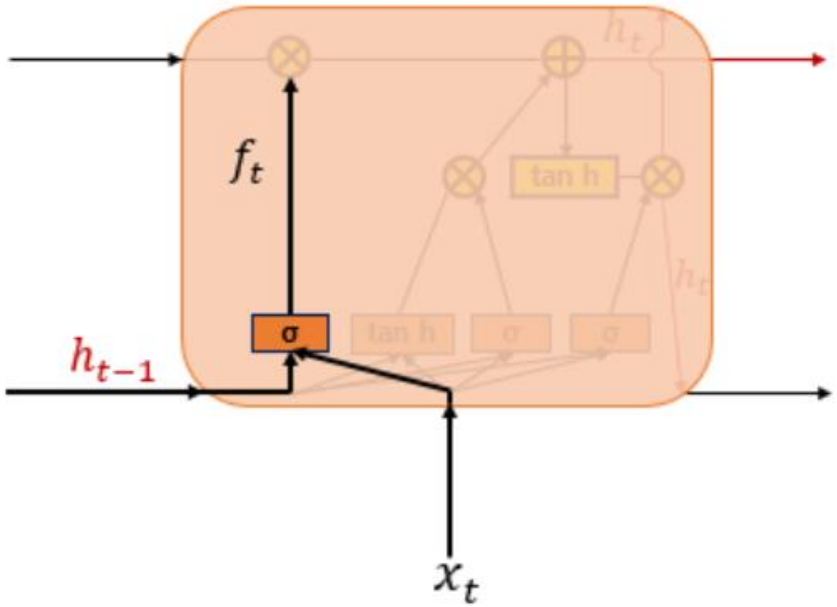
(1) 입력 게이트



$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$g_t = tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

- **현재 정보를 기억하기 위한 게이트**
- 시그모이드 함수 ➔(0, 1)
- 하이퍼볼릭 탄젠트 함수➔(-1, 1)

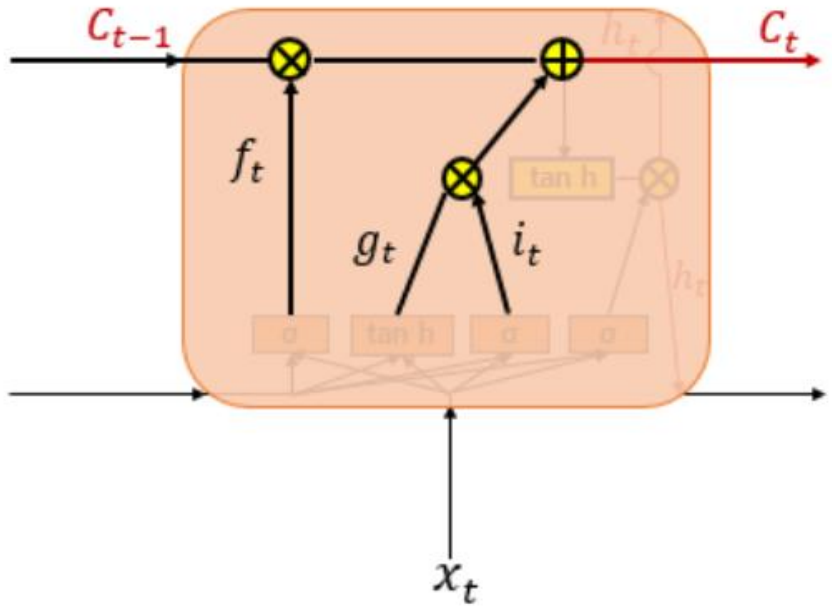
(2) 삭제 게이트



$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

- **기억을 삭제하기 위한 게이트**
- 시그모이드 함수 ➔(0, 1)
- 0에 가까울수록 정보가 많이 삭제된 것
- 1에 가까울수록 정보를 온전히 기억한 것

(3) 셀 상태(장기 상태)



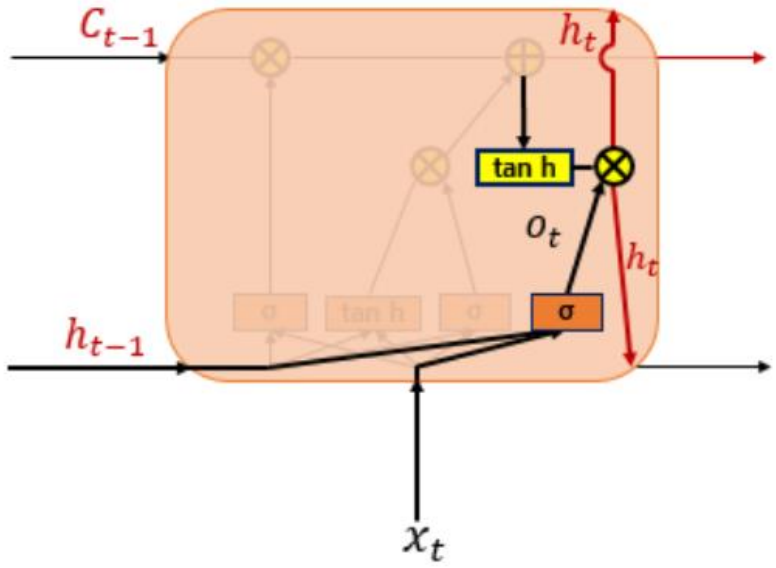
$$C_t = f_t \circ C_{t-1} + i_t \circ g_t$$

입력 게이트에서 구한 두 개의 값에 대해서 원소별 곱(entrywise product)+삭제 게이트의 결과값

- 삭제 게이트 출력값=0➔
입력게이트의 결과만이 현재 시점 셀 값 결정(삭제 게이트 완전히 닫힘)

- 입력 게이트 값=0➔
이전 시점의 셀 값만이 현재 시점 셀 값 결정(입력 게이트 완전히 닫힘)

(4) 출력 게이트와 은닉 상태(단기 상태)



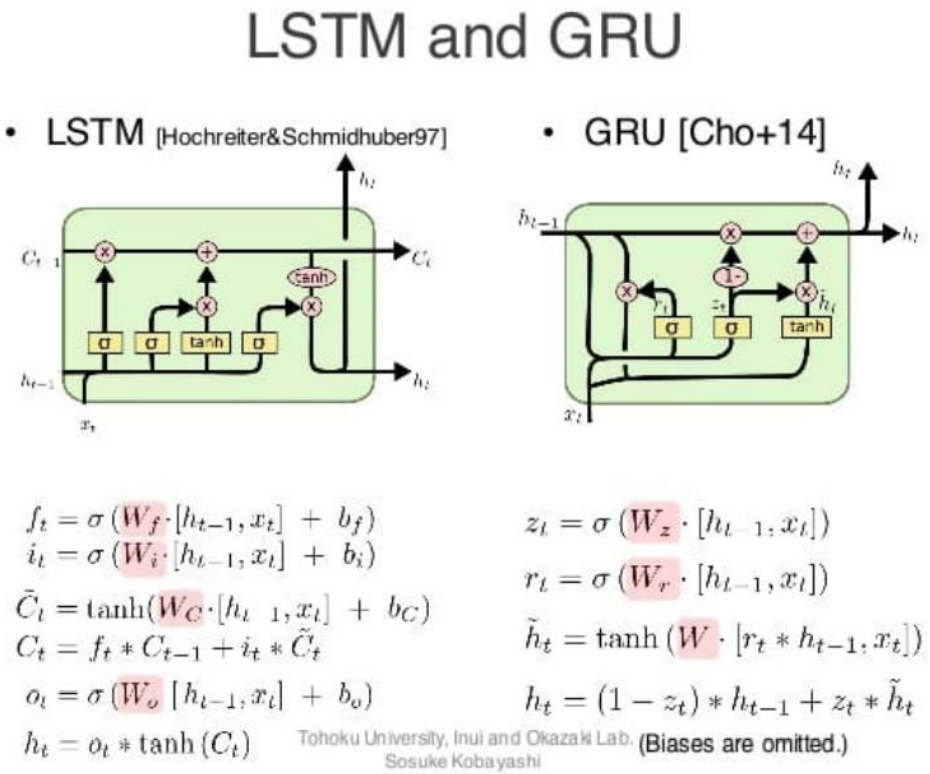
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \circ tanh(c_t)$$

- 현재 시점 t의 x값과 이전 시점의 t-1의 은닉 상태가 시그모이드 함수를 지난 값

- **은닉 상태(단기 상태)**: 하이퍼볼릭 탄젠트 함수를 지나 출력 게이트 값과 연산➔값이 걸러지는 효과가 발생

➔LSTM은 주로 덧셈의 구조로 이루어져있기 때문에 gradient vanishing/exploding 문제가 발생하지 않는다.

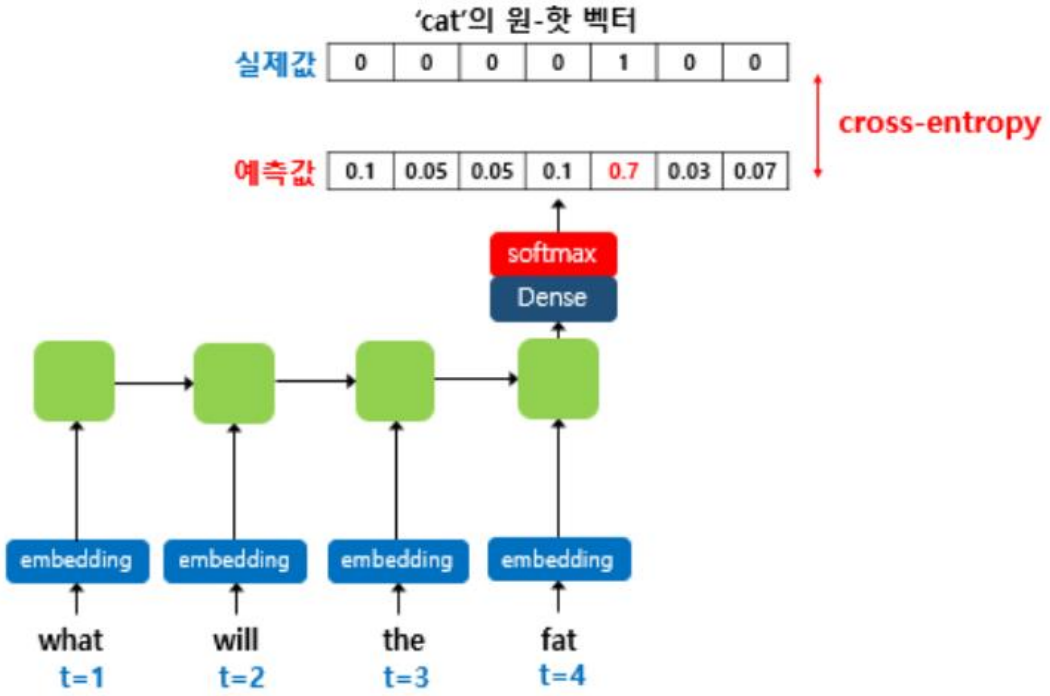
3) 게이트 순환 유닛(Gated Recurrent Unit, GRU)



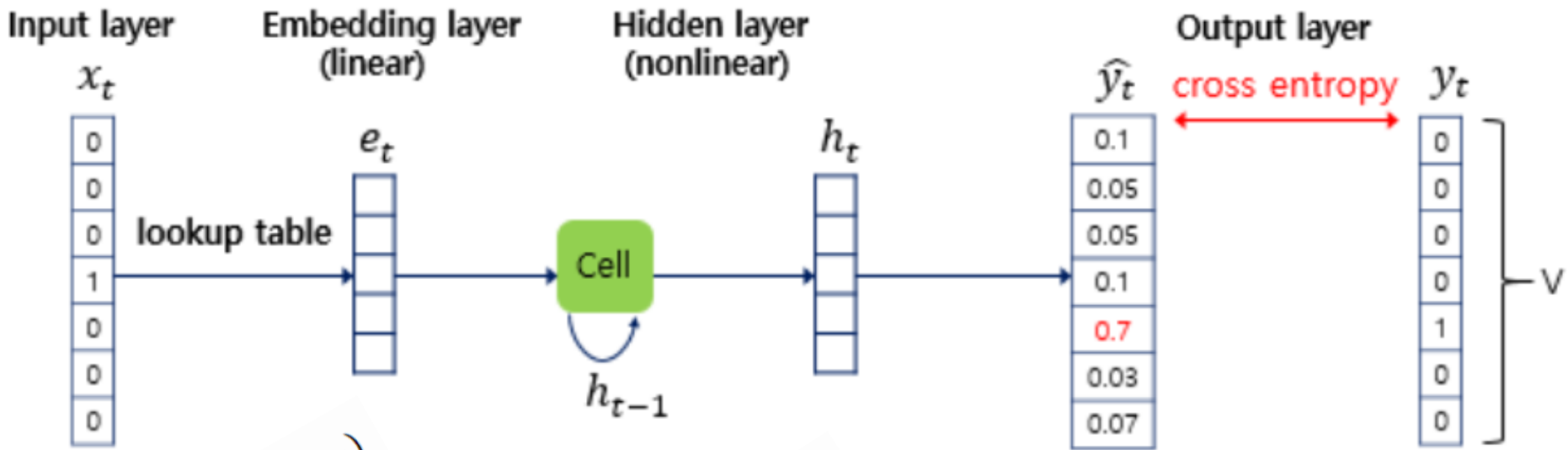
- **업데이트 게이트, 리셋 게이트**
- cell state와 hidden state를 하나로 묶어 hidden state로 사용
- forget gate와 input gate를 하나의 update gate zt
- reset gate rt 를 통해 forget 기능을 사용

4) RNN 언어 모델(Recurrent Neural Network Language Model, RNNLM)

단어 입력의 길이를 고정할 필요 없음



- **교사 강요(teacher forcing):** t 시점의 레이블. 즉, 실제 알고 있는 정답을 t+1 시점의 입력으로 사용



임베딩층 : $e_t = lookup(x_t)$

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

출력층 : $\hat{y}_t = softmax(W_y h_t + b)$