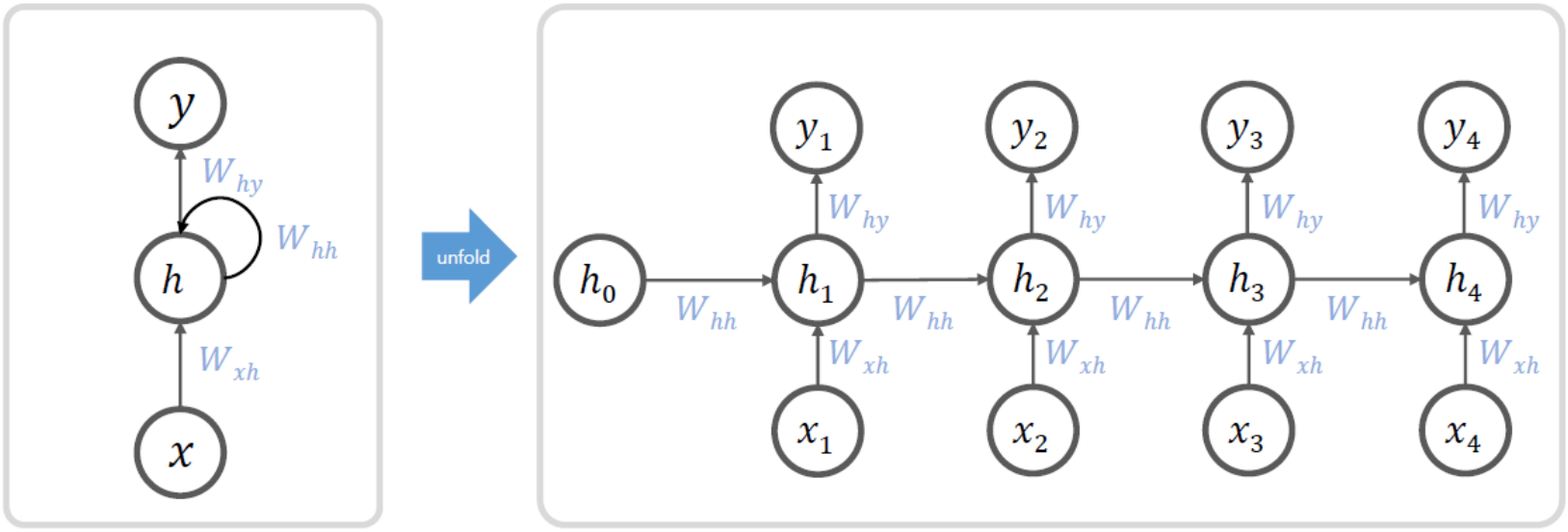
9. Recurrent Neural Network

RNN

CNN 영상처리 (X)

순환 신경망

- **Recurrent neural network (RNN):** 입력과 출력을 sequence 단위로 처리하는 신경망
  - 언어는 시간상으로 펼쳐지는 데이터 형태를 가지고 있어서 RNN으로 종종 처리됨
  - 개선된 형식으로는 LSTM (Long short-term memory), GRU(Gated recurrent unit) 등이 있음

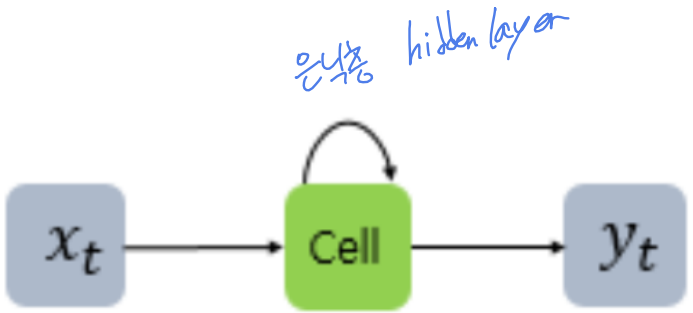


일반 RNN 표현 형식

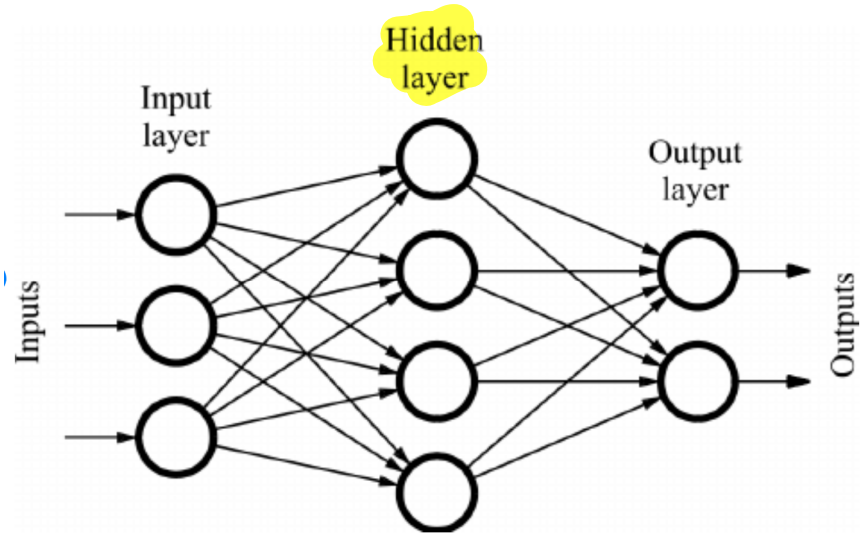
펼친 표현 형식

RNN 기본 구조

- RNN에서는 은닉층 값을 다음 상태로 전송하는 구조를 가지고 있음
- 은닉층 셀은 이전 상태의 영향을 받으므로 memory cell이라고도 함
- 은닉층은 여러 개의 셀들로 구성될 수 있음



RNN 구조



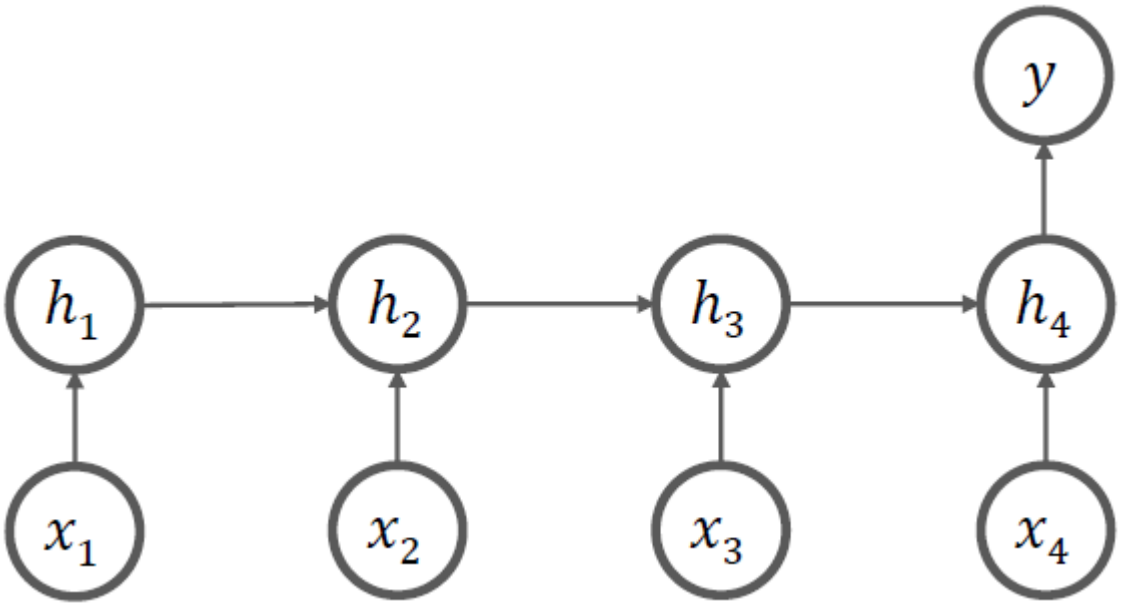
기존의 feed-forward 구조

RNN 분석 대상

- 언어, 음성, 음악, 주가지수, 날씨 등 시간에 따라 변하는 신호는 RNN 방식으로 분석할 수 있음

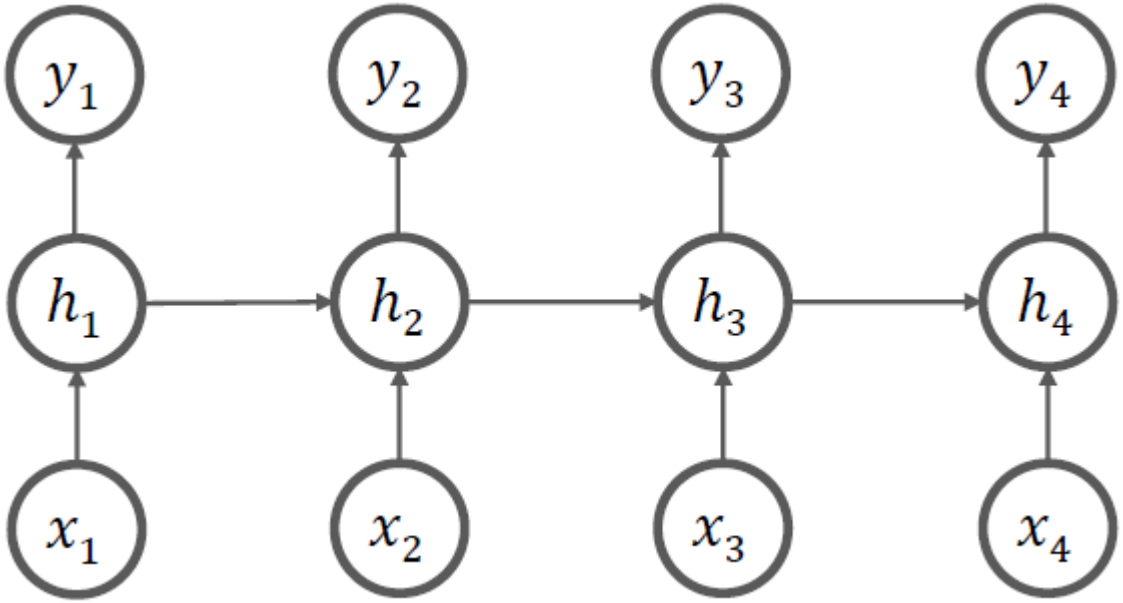
RNN 모델

- Many-to-One 구조
- Spam mail 분류, 감성분석(sentiment analysis) 등에 사용



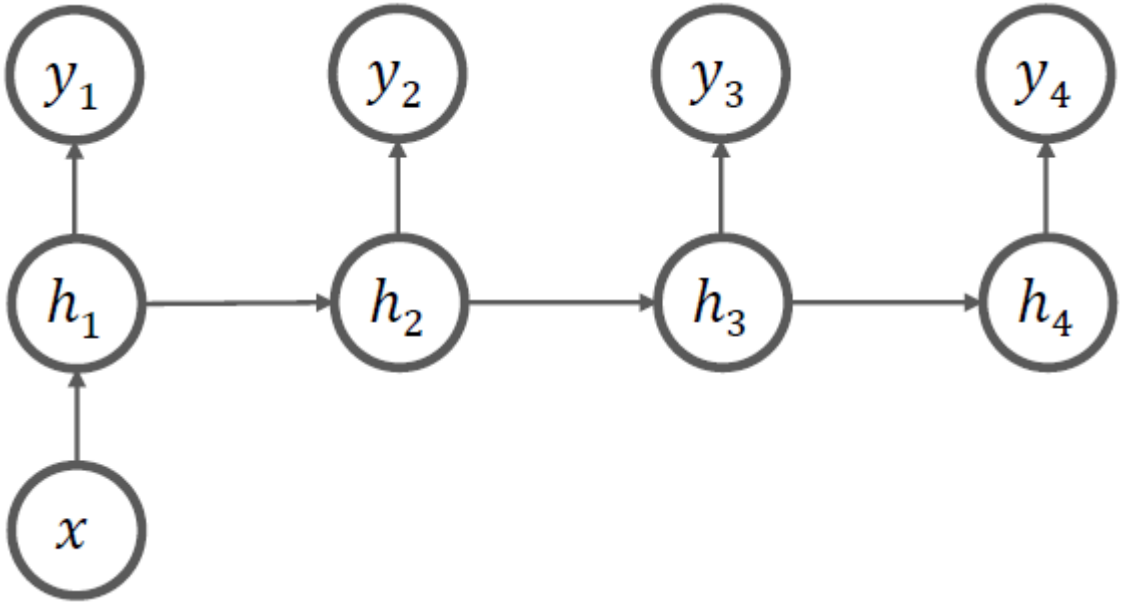
RNN 모델

- Many-to-Many 구조
- 문서 생성 등에 사용



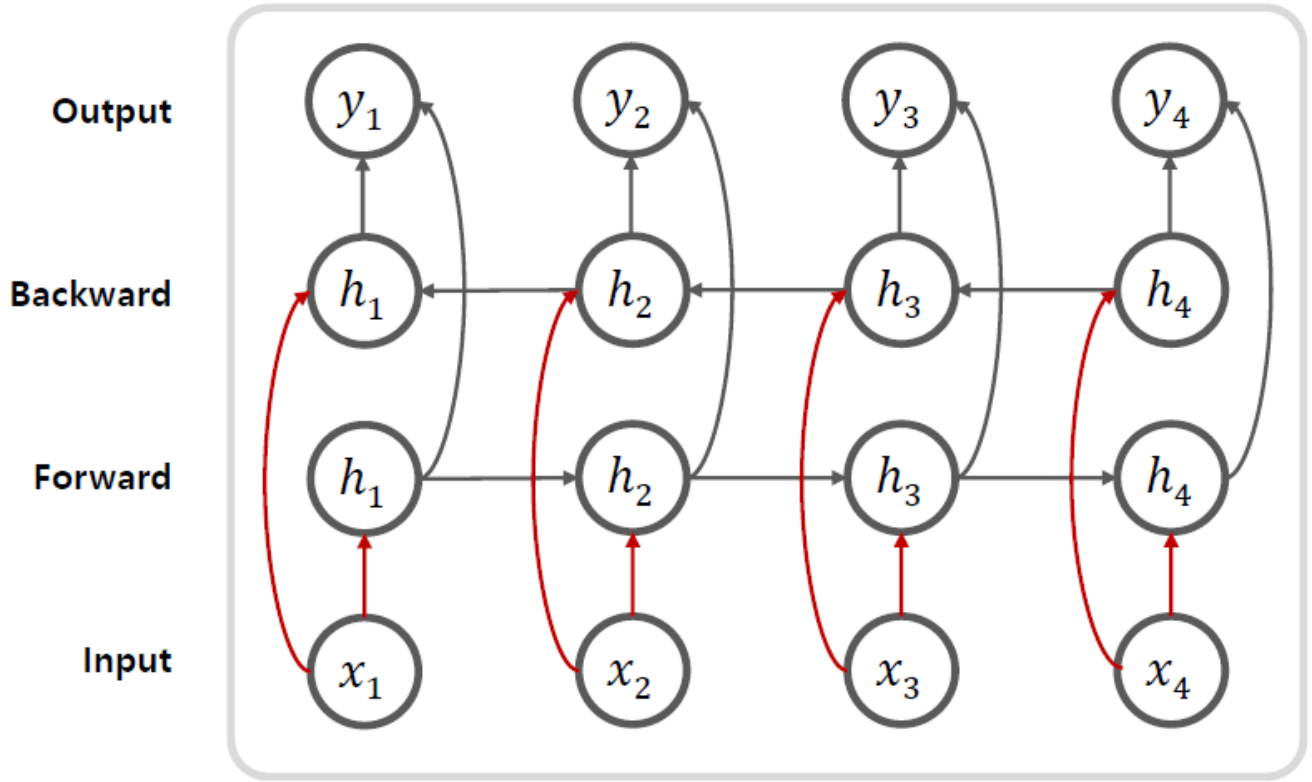
RNN 모델

- One-to-Many 구조
- 영상에서 설명문을 유추하는 Image captioning 등에 사용



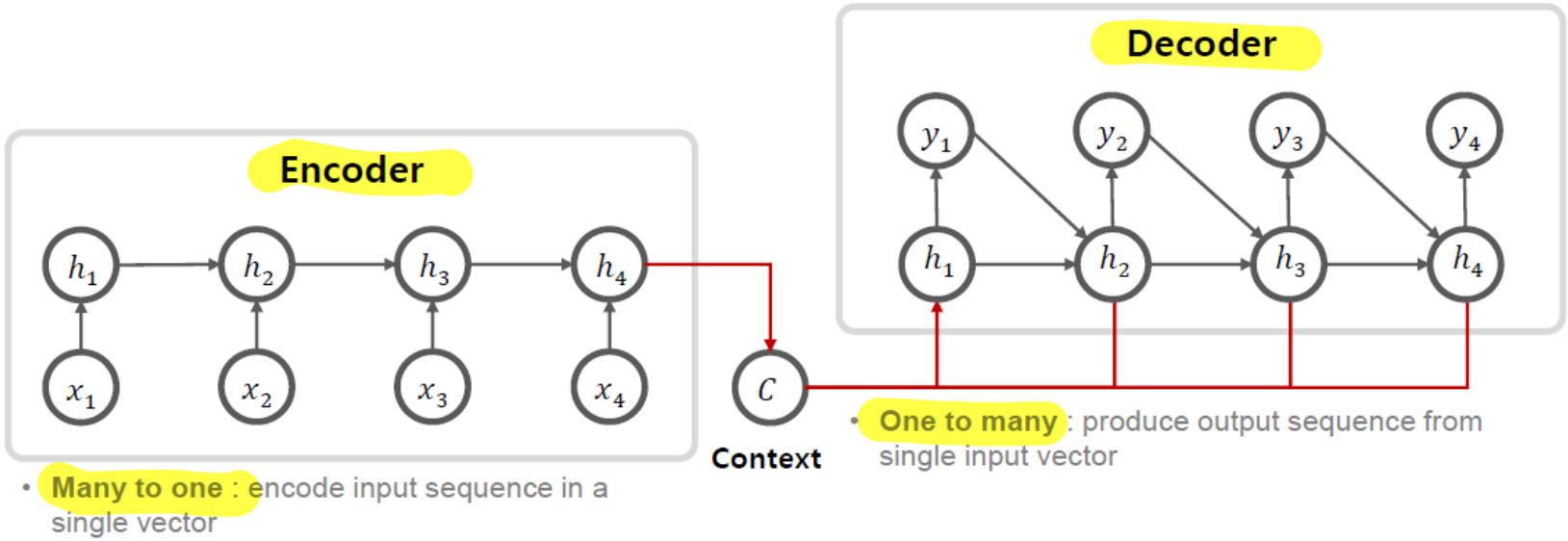
RNN 모델

- Bidirectional 모델
- 양방향으로 진행



RNN 모델

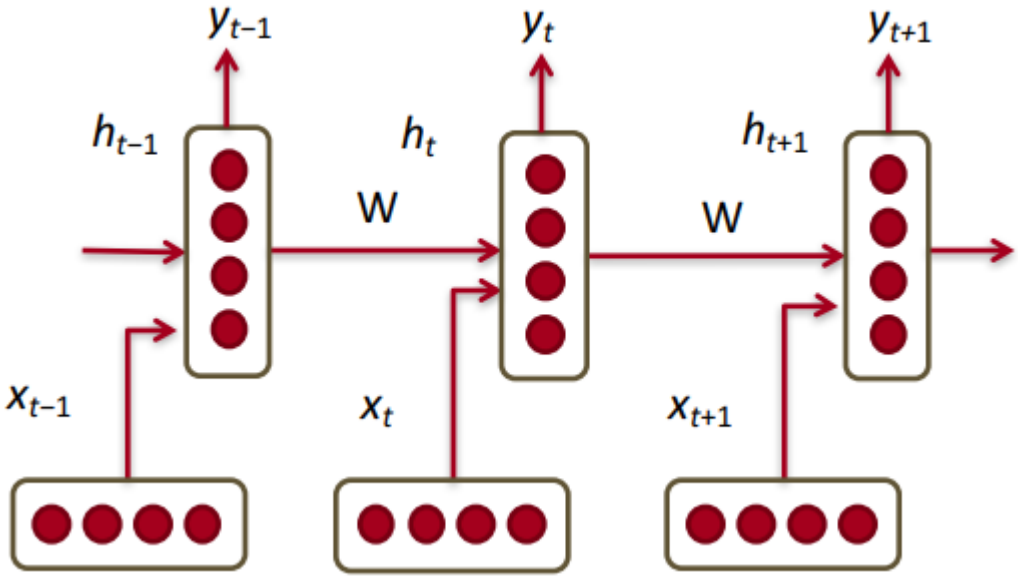
- Encoder-decoder (sequence-to-sequence) 모델
- 기계 번역에 사용



RNN의 수식 표현

- 은닉층: $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$
- 활성화 함수는 tanh 대신 ReLU를 사용하는 경우도 있음
- 출력층: $y_t = f(W_y h_t + d)$

  f는 비선형 활성화 함수
- 입력 x_t와 은닉층 h_t는 모두 일반적으로 벡터임



RNN 수식 표현

- 입력 벡터의 차원이 d, 은닉층의 크기를 D_h라 했을 때

d: 입력차원
D_h : h의 차원

$x_t: d \times 1$

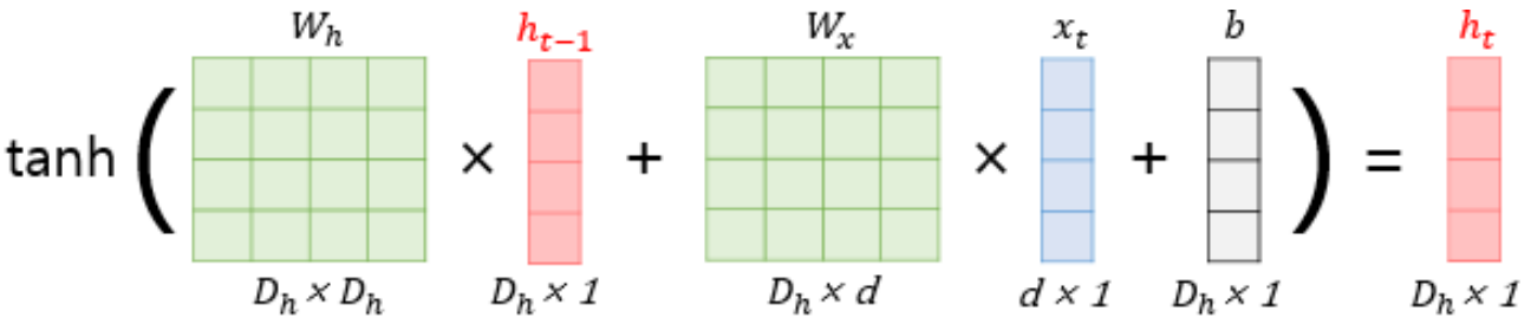
$W_x: D_h \times d$

$W_h: D_h \times D_h$

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

$b: D_h \times 1$

RNN에서 미지수는 W_x, W_h, b 이므로
파라미터 개수는 $D_h(D_h + d) + D_h$ 임

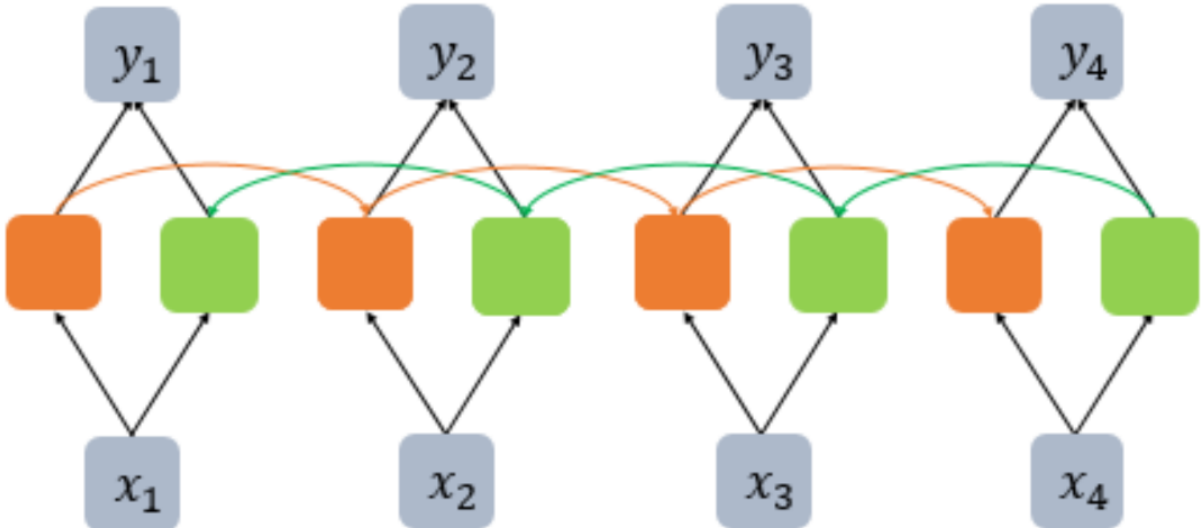


양방향(Bidirectional) RNN

- 양방향 RNN의 필요성: 문제를 풀기 위해 나중에 나오는 단어들도 필요한 경우가 있음
- 빈 칸 채우기 예제

  Exercise **is** very effective at [ ] belly fat.

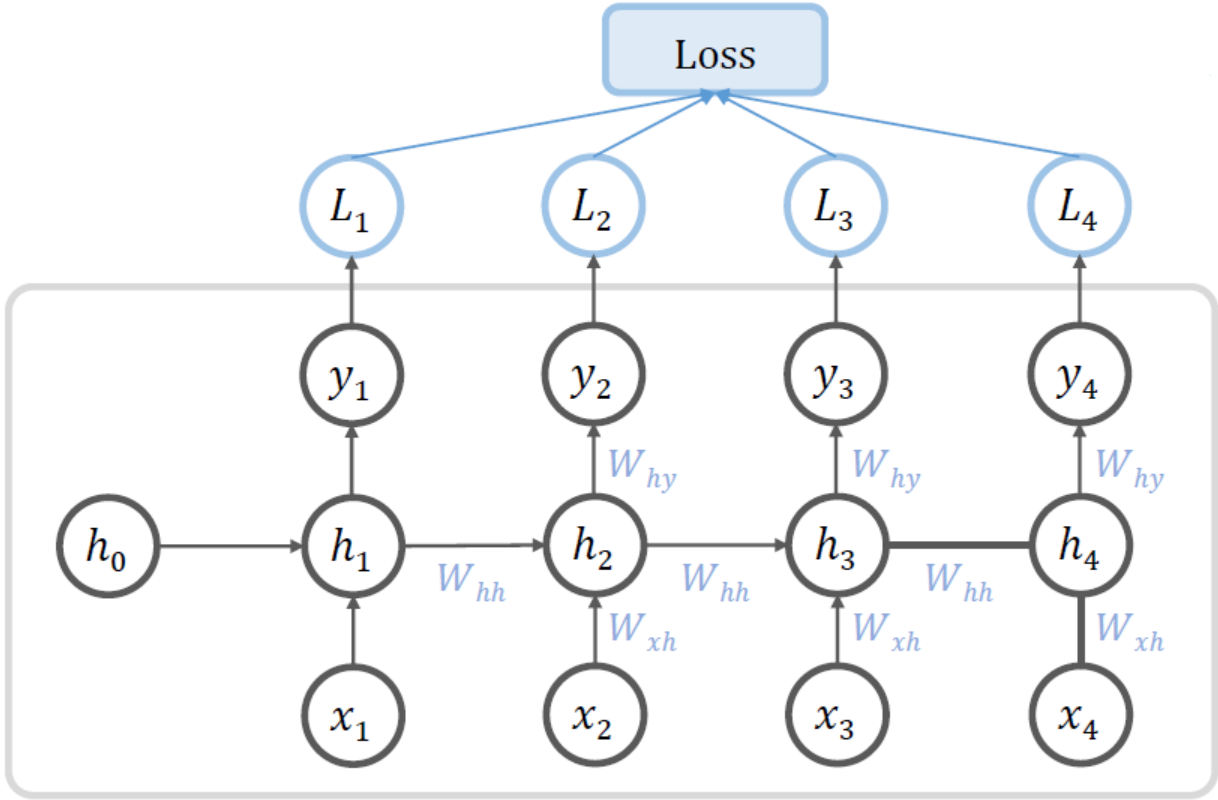
  1) reducing 2) increasing 3) multiplying
- RNN을 양방향으로 구현하는 방식도 사용되고 있음



RNN에서의 비용 함수

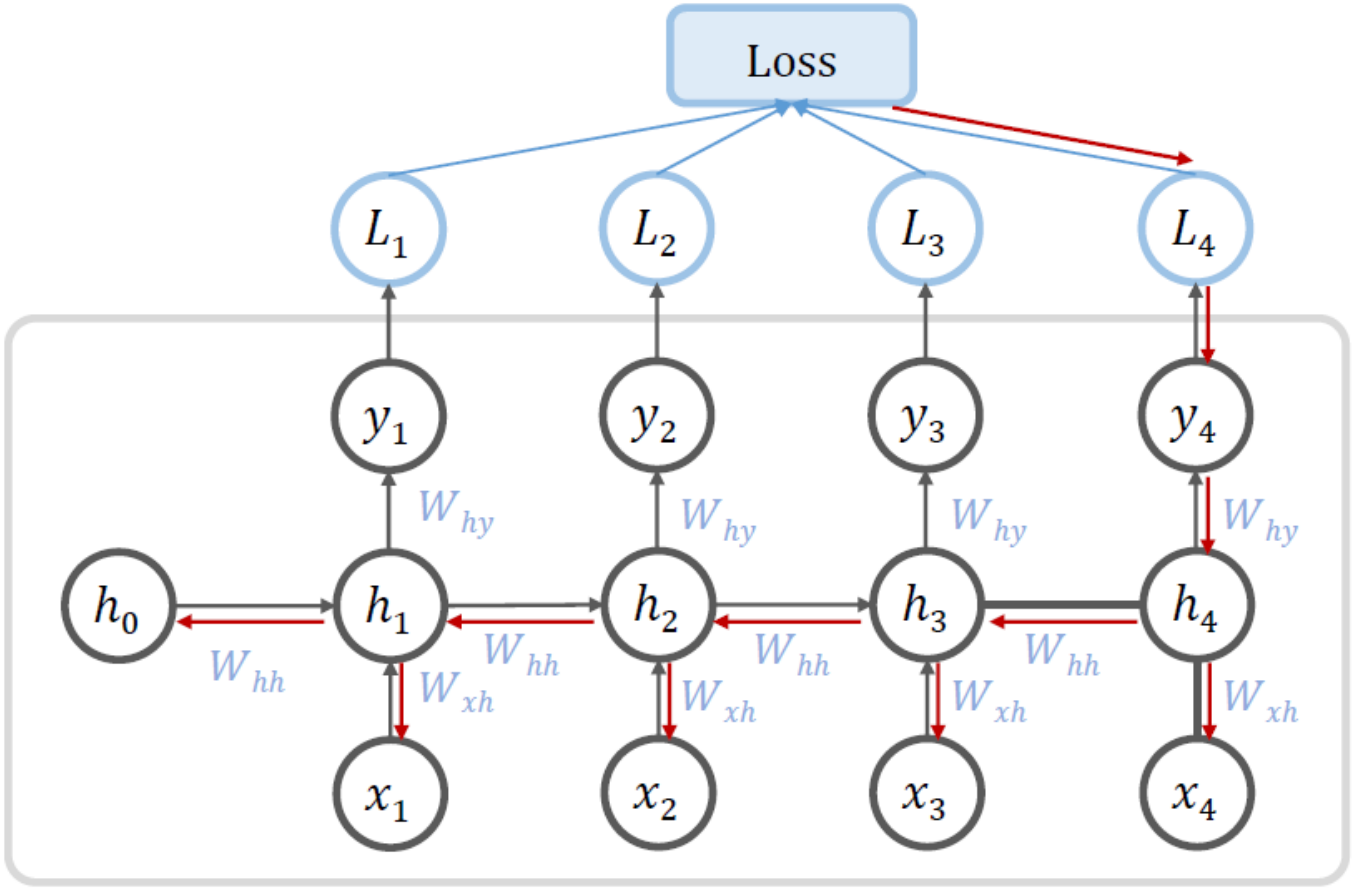
- RNN의 손실은 일반적으로 입력 x_t가 들어올 때 마다 L_t를 계산

$$LOSS = \sum_{i=1}^{T} L_i$$



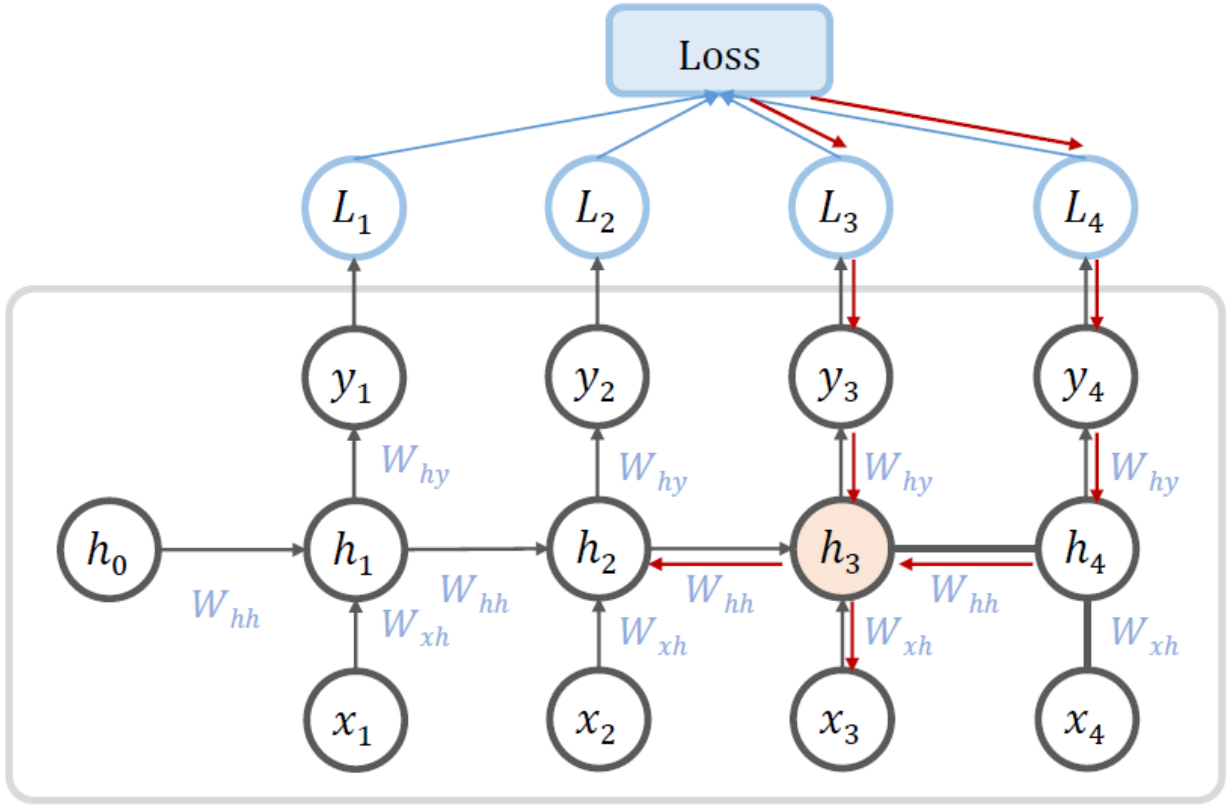
Backpropagation

- 한 문장과 같은 일정한 데이터가 들어오면 Backprop에 의해 파라미터를 갱신함
- Backprop은 이전 단계로 전파되어야 함



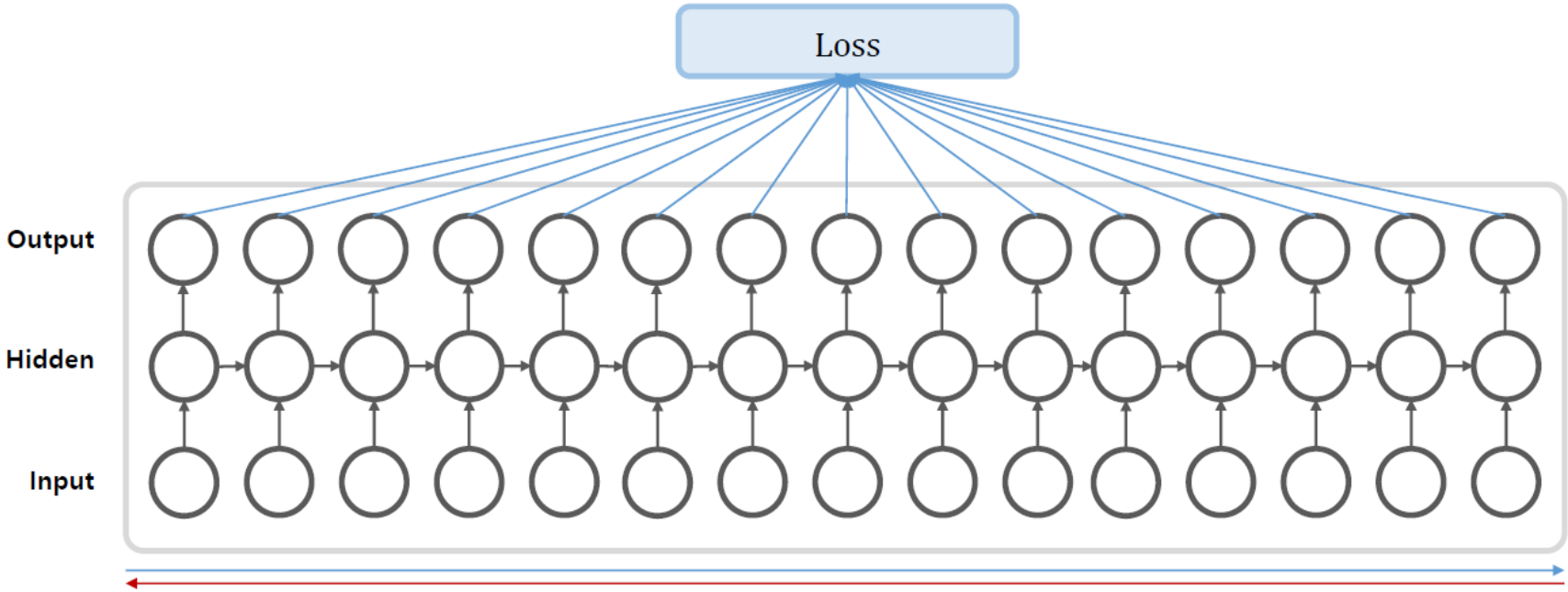
Backpropagation

- Hidden node에서는 다음 단계에서 받은 gradient와 출력에서 받은 gradient를 더해서 사용



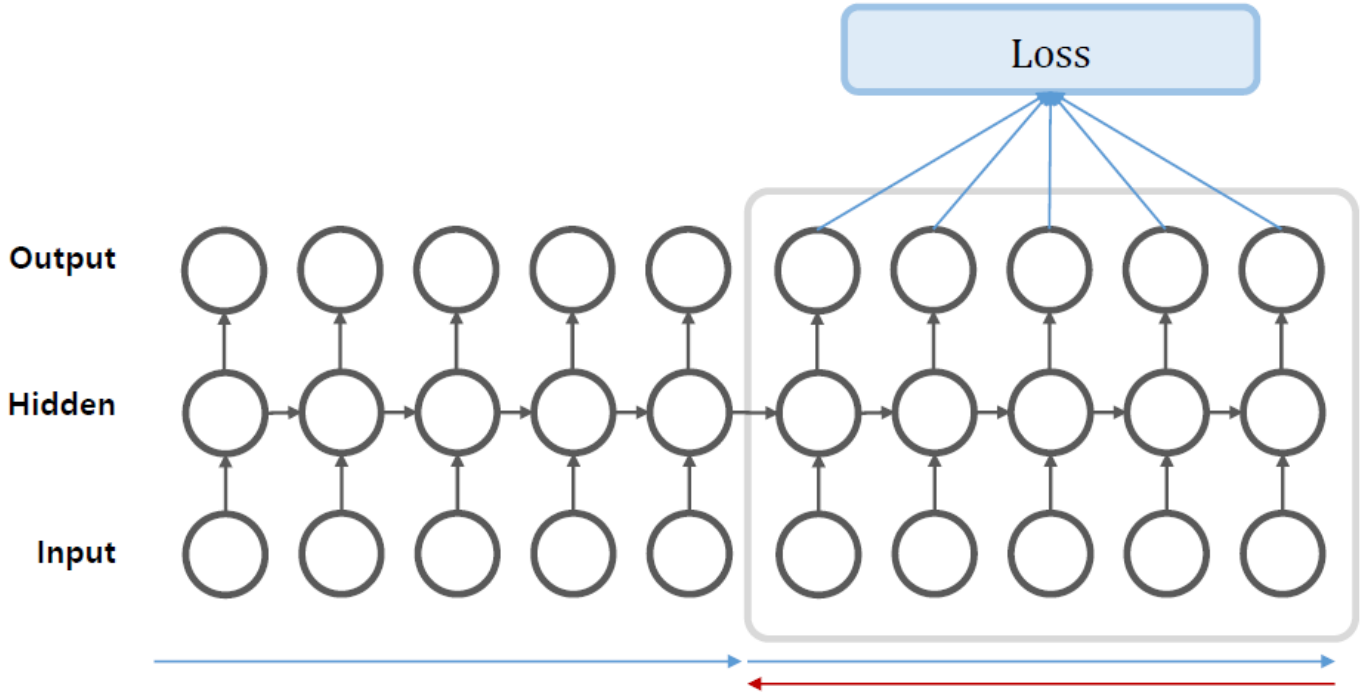
Backpropagation through time(BPTT)

- 한 문장과 같은 일정한 데이터가 들어오면 Backprop에 의해 파라미터를 갱신. 이 과정은 처음 단계까지 전파되어야 함



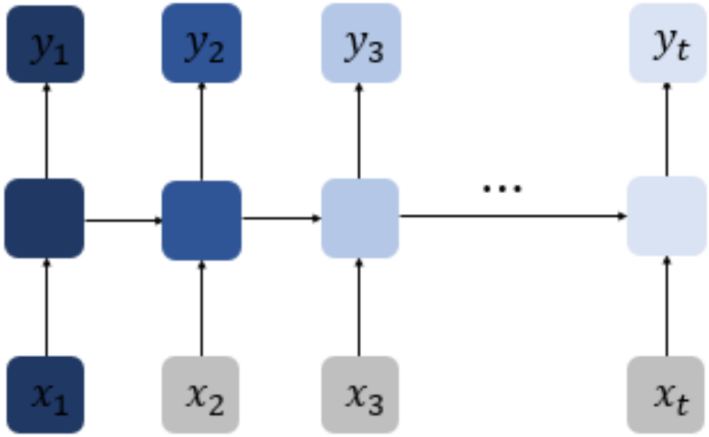
Truncated BPTT

- 계산 시간 절약을 위해 backprop 전파 단계를 일정 숫자로 제한



기본(Vanilla) RNN의 한계

- 은닉층에서 vanishing gradient 문제로 인해 이전 단어의 영향력이 오래 지속되지 못함: 아래 그림과 같이 시간이 지나면서 영향력이 줄어듦



- 언어에서는 단어의 영향력을 보다 오랫동안 유지해야 하는 경우가 많음 (**장기 의존성 문제: Long-term dependencies**)
- **사례:** 모스크바에 여행을 왔는데 건물도 예쁘고 먹을 것도 맛있었어. 그런데 글쎄 직장 상사한테 전화가 왔어. 어디냐고 묻더라구 그래서 나는 말했지. 저 여행왔는데요. 여기 ___

Gradient vanishing/exploding

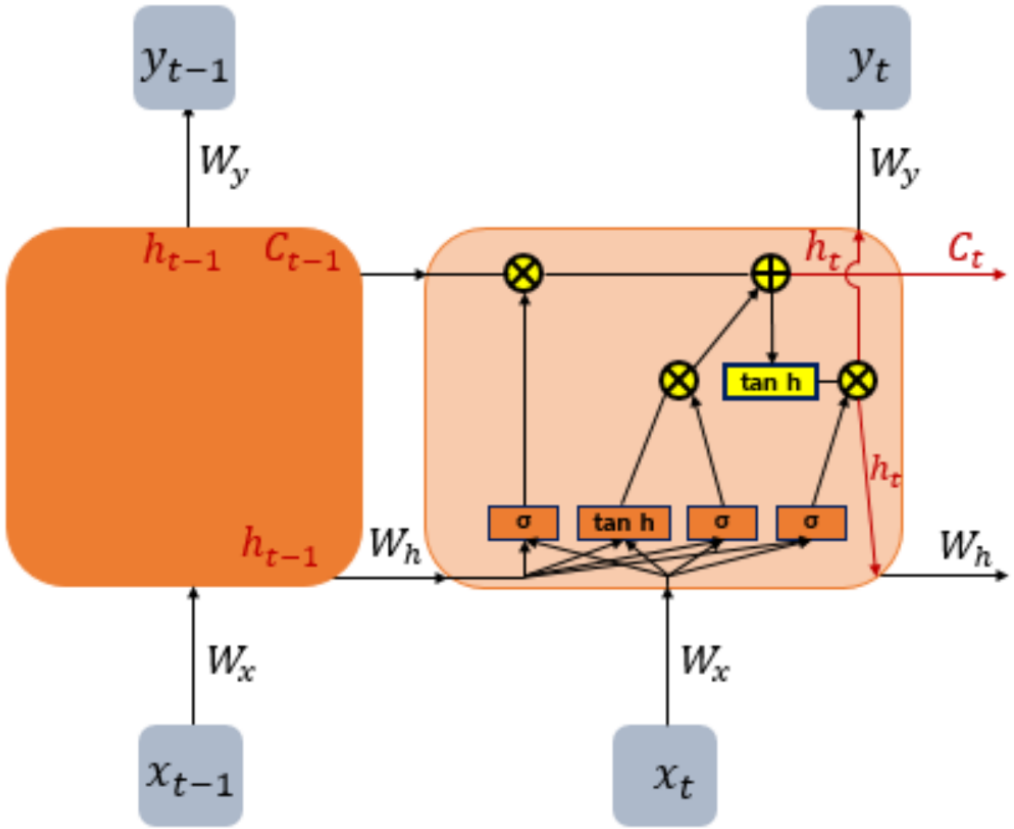
- RNN에서는 진행할 때마다 W 행렬이 계속 곱해짐

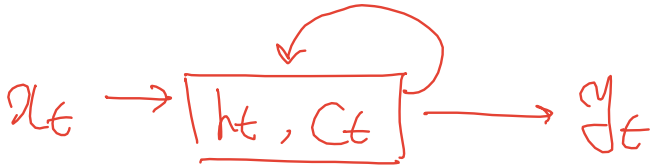
$$
\begin{aligned}
h_t &= \tanh(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t) \\
&= \tanh\left((W_{hh} \quad W_{xh}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)
\end{aligned}
$$

- Backprop시에도 이전 단계로 갈 때마다 W가 곱해짐
- 이러한 성질에 의해 이전 단계로 갈수록 데이터 전파가 어려워지게 됨
- 이러한 한계를 극복하는 수단으로 LSTM과 GRU가 제안되었음

LSTM (Long short-term memory)

- 기본 RNN의 장기 의존성 문제를 해결하기 위해 불필요한 기억을 지우고 기억해야 할 것을 정하는 기법
- 긴 시퀀스의 입력을 처리하는데 좋은 성능을 보임





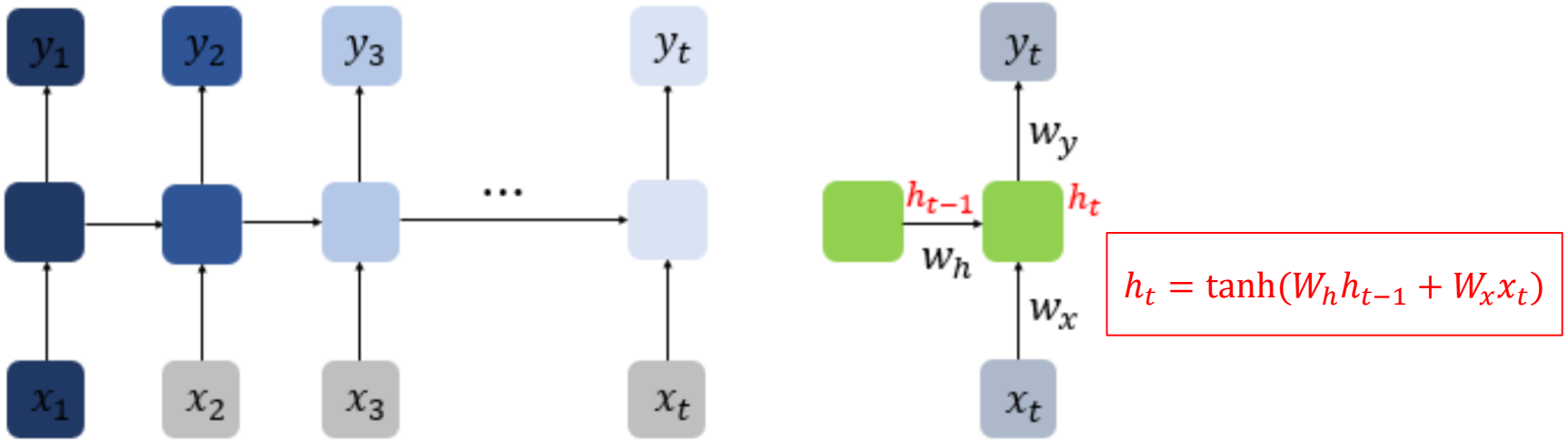
LSTM 동작 방식

- 은닉층에서 h_t 이외에 셀 상태를 나타내는 c_t 를 가지고 있음
- h_t와 c_t를 구하기 위해 삭제, 입력, 출력 등 세 개의 게이트값을 활용함
- LSTM은 RNN에 비해 훈련시켜야 하는 파라미터의 개수가 4배로 많아짐
- LSTM은 기계번역, 문서분류, 문서 요약 등에서 기본 RNN에 비해 우수한 성능을 보이고 있음

- LSTM과 유사한 성능을 가지며 보다 단순한 구조로 GRU(Gated recurrent unit)도 제안되어 있음

Long Short-term Memory (LSTM)

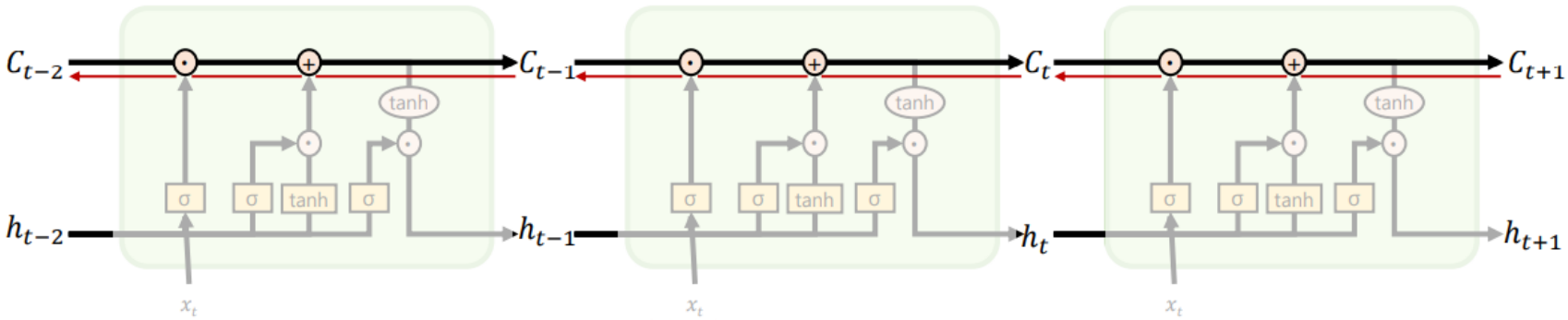
- RNN의 한계: Vanishing gradient 문제로 인해 이전 단어의 영향력이 오래 지속되지 못함: 아래 그림과 같이 시간이 지나면서 영향력이 줄어듦



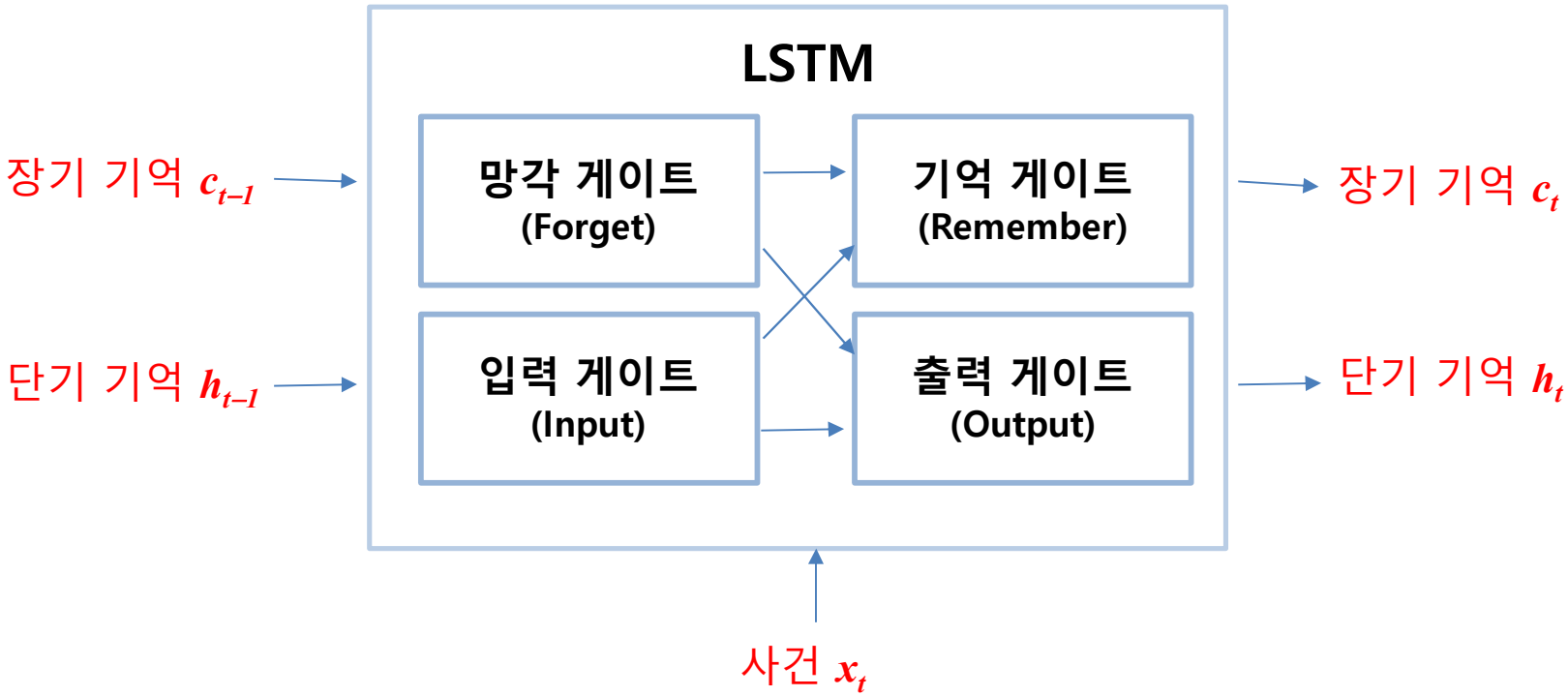
- 이 문제는 backpropagation 과정에서 W_h가 계속 곱해지면서 결과가 점점 0으로 수렴되면서 발생하는 것임
- 이 문제를 해결하기 위한 구조로 LSTM이 제안되었음

LSTM 구조

- RNN과 달리 두 개의 상태변수 c_t, h_t를 가짐
- Backpropagation에서 W가 곱해지지 않는 구조로 만듦
- c_t는 오래 전파되어야 하는 장기 기억, h_t는 단기 기억을 나타냄

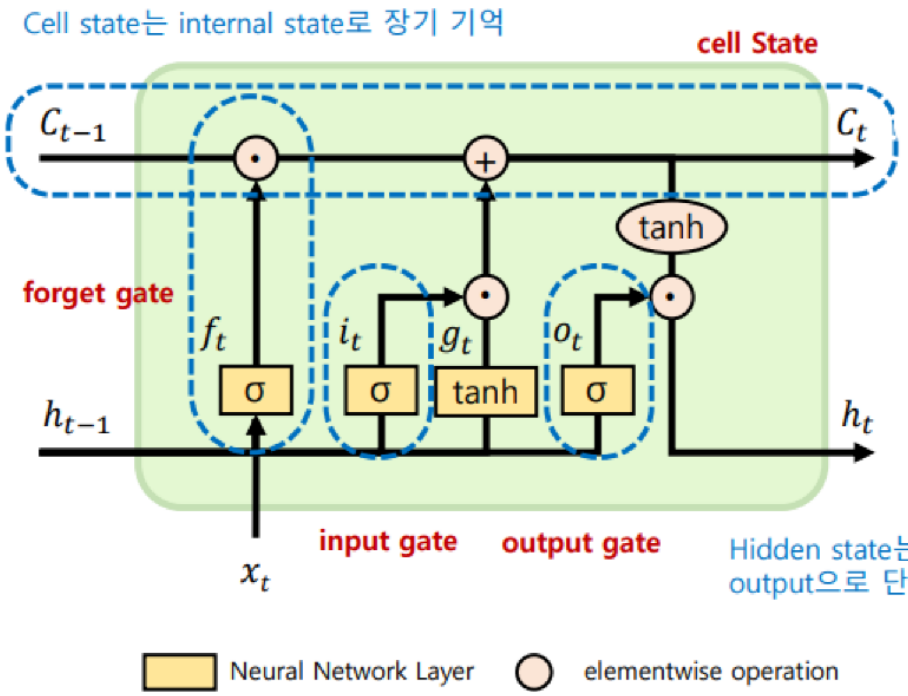


LSTM: Long Short Term Memory



LSTM 게이트들을 이용하여 단기와 장기 기억을 어느 정도 활용하는지를 결정

LSTM 구조



$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}, \quad W = \begin{bmatrix} W_{hi} & W_{xi} \\ W_{hf} & W_{xf} \\ W_{ho} & W_{xo} \\ W_{hg} & W_{xg} \end{bmatrix}$$

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

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

⊙ 는 원소별 곱셈을 의미

Gate 종류

- $g_t \in$ **(-1,1)** : 셀에서 형성된 새로운 기억
- $i_t \in$ **(0,1)** : 입력 게이트 (input gate)
- $f_t \in$ **(0,1)** : 망각 게이트 (forget gate)
- $o_t \in$ **(0,1)** : 출력 게이트 (output gate)

LSTM 동작 수식

- 이전 단에서의 입력: c_{t-1} , h_{t-1}
- 현재단에서의 입력: x_t
- 게이트값 계산:

  입력 게이트: $i_t = \sigma(W_{hi}h_{t-1} + W_{xi}\, x_t)$

  망각 게이트: $f_t = \sigma\left(W_{hf}h_{t-1} + W_{xf}\, x_t\right)$

  출력 게이트: $o_t = \sigma(W_{ho}h_{t-1} + W_{xo}\, x_t)$

  $g_t = tanh\left(W_{hg}h_{t-1} + W_{xg}\, x_t\right)$

- 출력 계산:

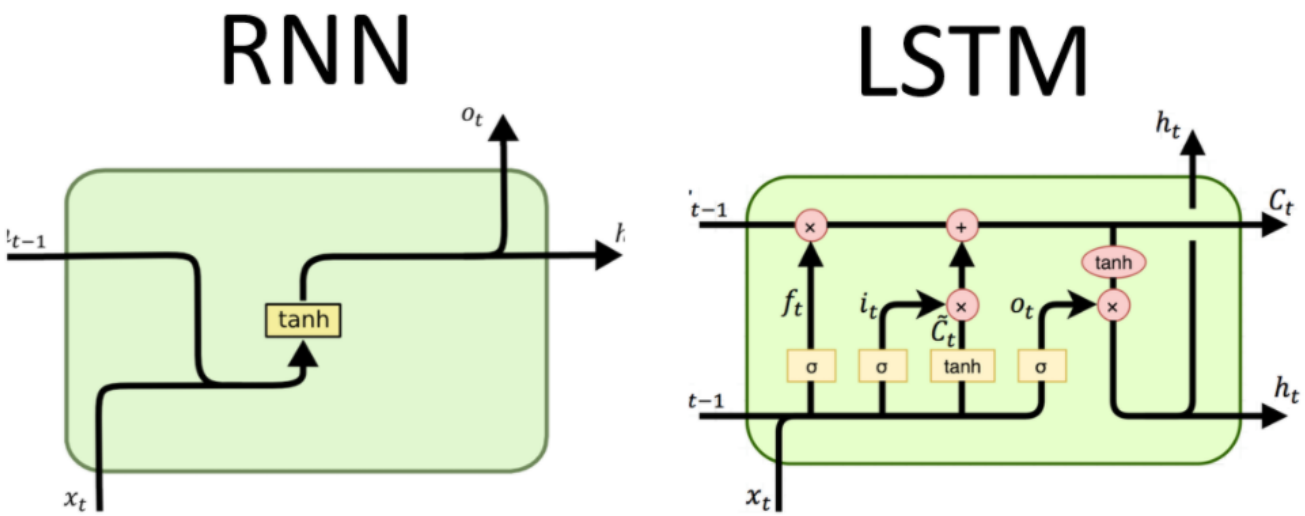
  $c_t = f_t \odot c_{t-1} + i_t \odot g_t$

  $h_t = o_t \odot \tanh(c_t)$

  ⊙ 는 원소별 곱셈을 의미

RNN과 LSTM 비교

- 일반적으로 LSTM은 RNN에 비해 장기 기억에서 성능이 더 우수한 것으로 평가됨
- RNN의 경우 셀당 파라미터의 숫자는 RNN의 경우 2개(W_h와 W_x)이나 LSTM은 8개($W_{hi} \sim W_{hg}, W_{xi} \sim W_{xg}$)이므로 더 많은 훈련 데이터가 필요



LSTM 파라미터 개수

- 입력 벡터의 차원이 d, 은닉층의 크기를 D_h라 했을 때

$x_t: d \times 1$

$W_x: D_h \times d$

$W_h: D_h \times D_h$

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

$b: D_h \times 1$

LSTM에서 미지수는 W_x, W_h, b 이므로
파라미터 개수는 $4[D_h(D_h + d) + D_h]$임

케라스에서 RNN과 LSTM의 파라미터 비교

- 케라스에서 RNN과 LSTM을 생성할 때 파라미터 숫자를 비교해보면 다음과 같음
- RNN의 경우

```
model = Sequential()
model.add(Embedding(1000, 100))
model.add(SimpleRNN(120))
```

```
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
embedding (Embedding) (None, None, 100) 100000
simple_rnn (SimpleRNN) (None, 120) 26520
_________________________________________________________________
```

$(100 + 120) \times 120 + 120$

- LSTM의 경우

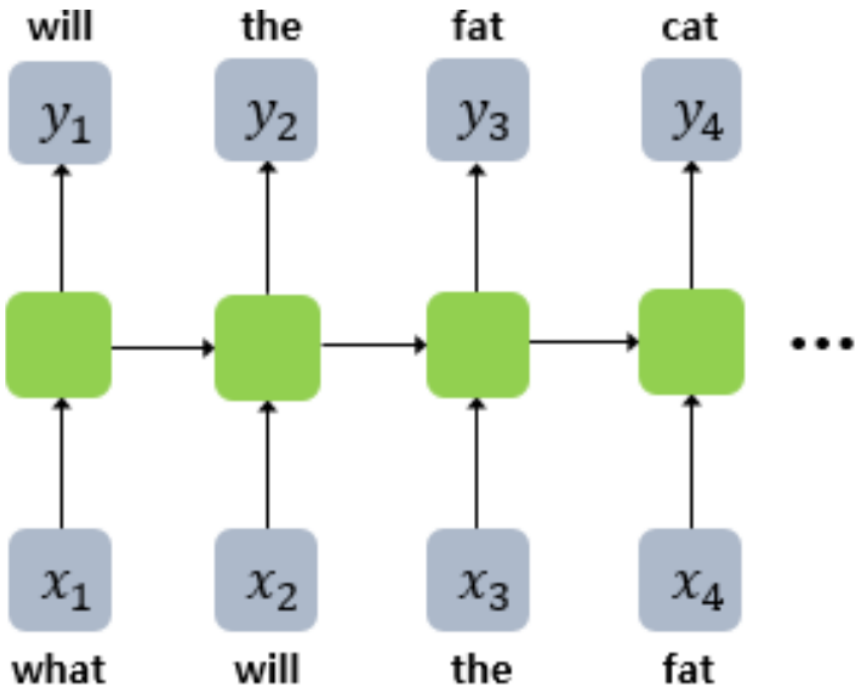
```
model = Sequential()
model.add(Embedding(1000, 100))
model.add(LSTM(120))
```

```
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
embedding (Embedding) (None, None, 100) 100000
lstm (LSTM) (None, 120) 106080
_________________________________________________________________
```

$4 \times ((100 + 120) \times 120 + 120)$

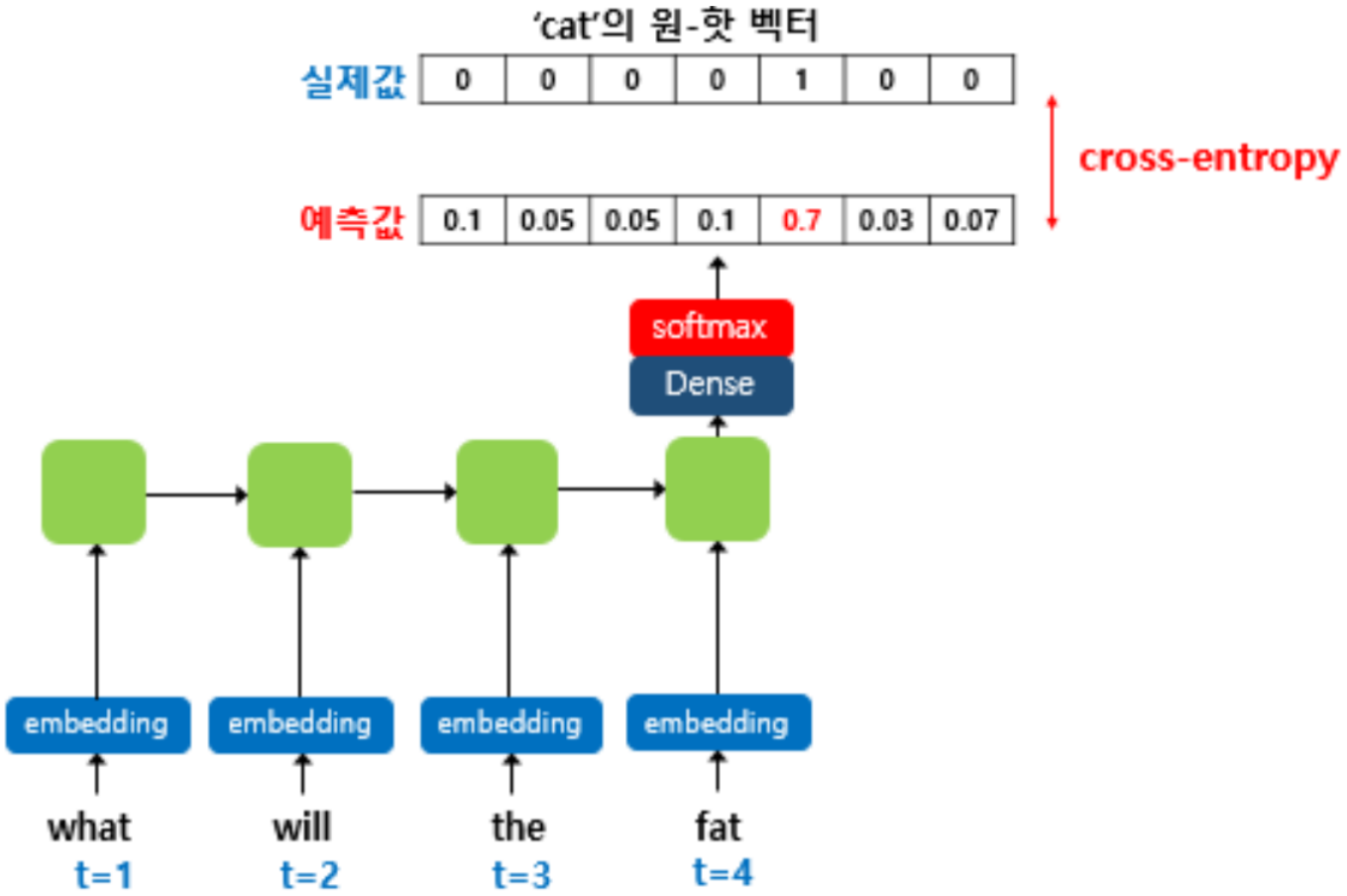
RNN Language Model (RNNLM)

- 쓰여진 문장들을 RNN 방식으로 학습시켜서 다음에 나오는 단어를 예측하는 방식
- 학습 방식 예제: “what will the fat cat sit on”이라는 문장으로 학습하는 경우 이전 단어와 현재 단어로 다음 단어를 예측



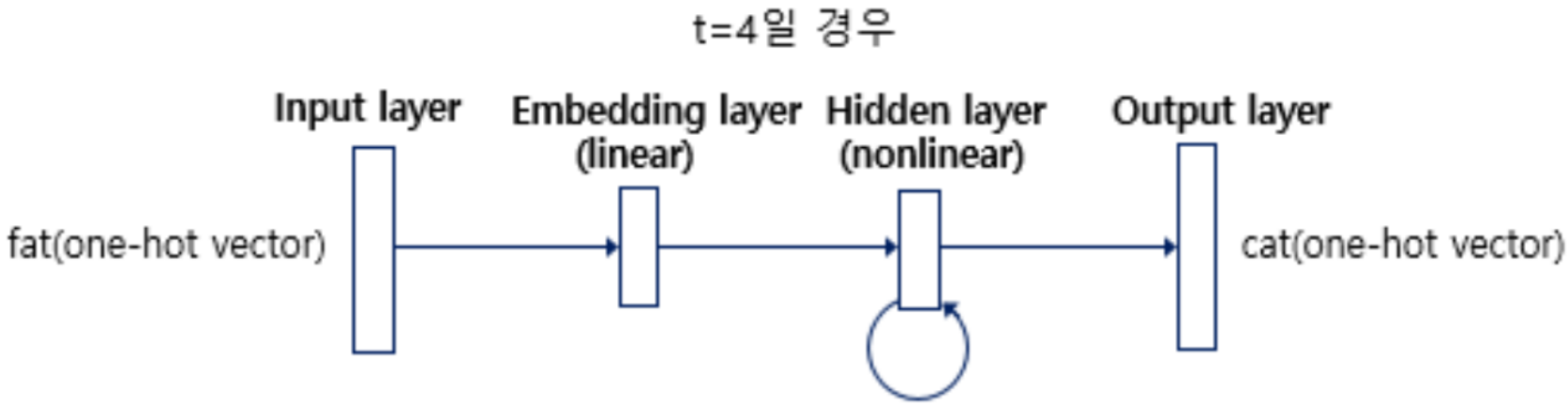
RNNLM 학습 방식

- 학습 문장에 나오는 다음 단어와 예측된 출력값을 비교하여 비용 함수를 정의



RNNLM 학습 구조

- 신경망은 RNN으로 구성
- 입력과 출력 단어는 one-hot vector로 표현
- 입력 단어는 embedding vector로 변환됨. 각 단어를 embedding vector로 표현하는 방법은 별도로 유도된 방식을 사용(10장의 word2vec 방식을 사용할 수도 있음)



RNNLM 진행 수식

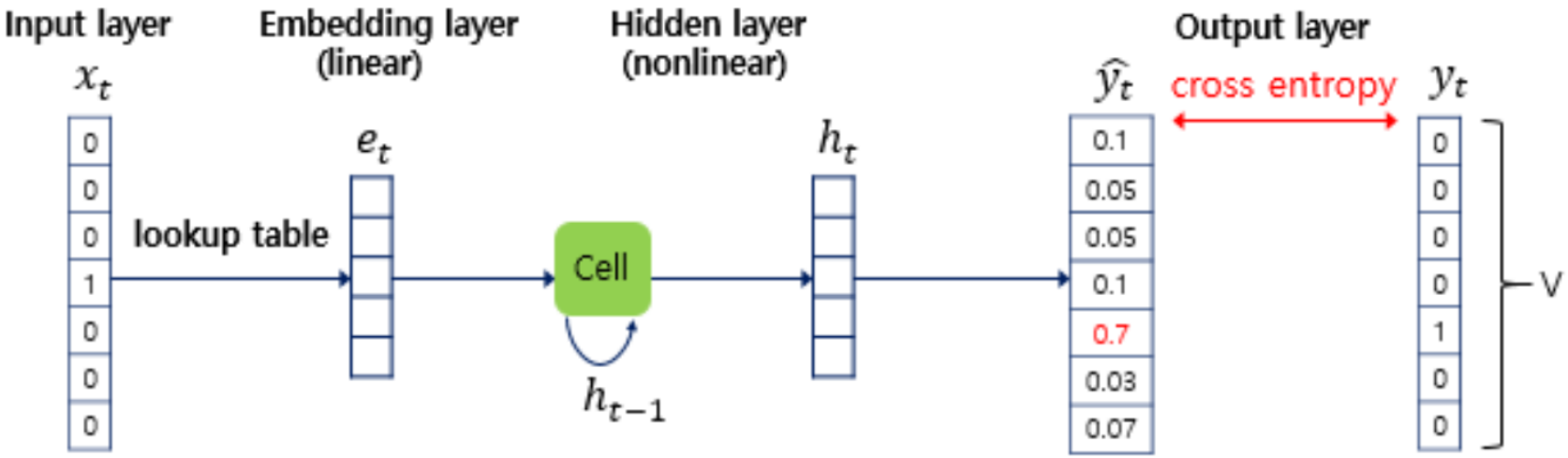
- 변환 수식:

  임베딩층 $e_t = lookup(\mathrm{x_t})$

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

  출력층 $\hat{y_t} = \mathrm{softmax}(W_y h_t + d)$

- 비용 함수를 줄이는 방향으로 임베딩 벡터를 학습시킴



RNNLM을 이용한 텍스트 생성

- RNNLM은 현재까지의 단어들로 부터 다음 단어를 예측하는 기능을 가지고 있으므로 충분히 훈련된 상태에서는 문장을 생성할 수 있음
- 기본 RNN이나 LSTM을 이용하여 텍스트를 생성하는 기능을 구현할 수 있음