

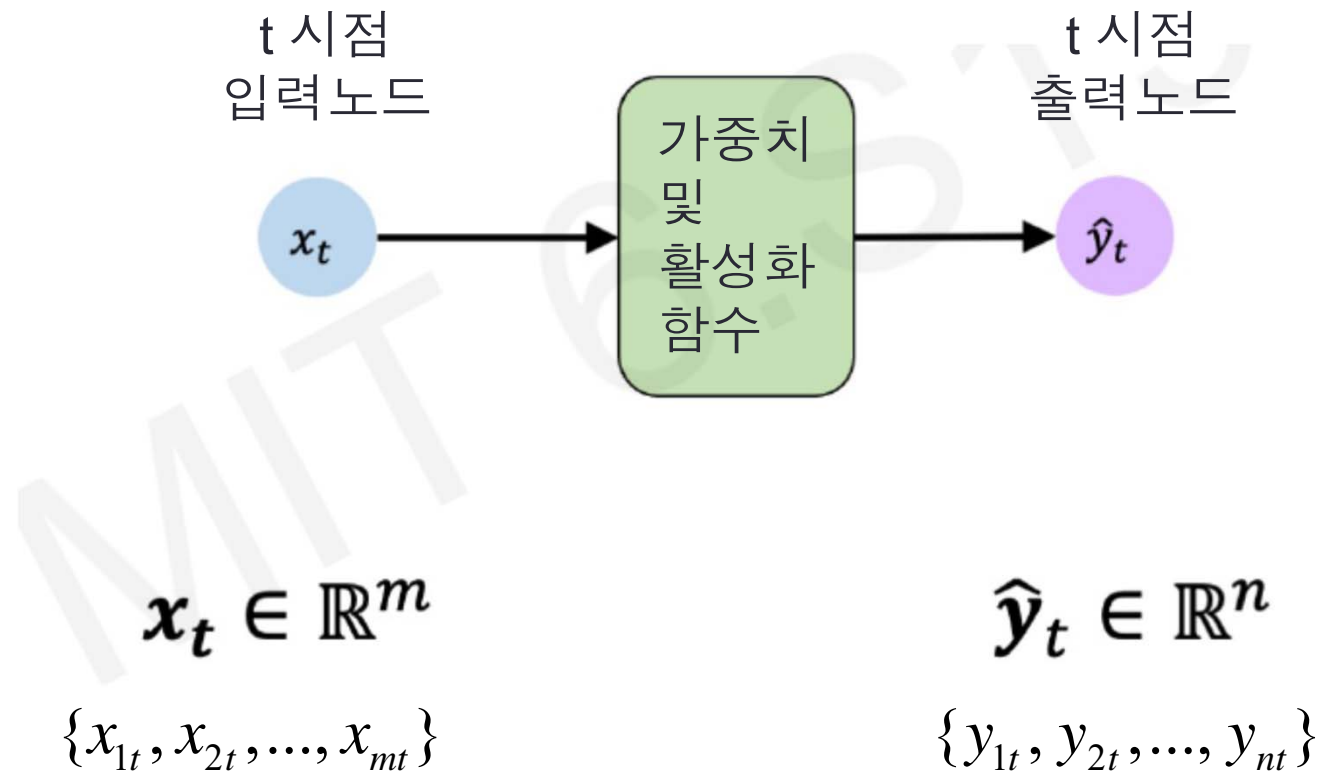
03. 순서데이터 처리를 위한 딥러닝

순환 신경망

Recurrence Neural Network

Feed-forward Network

- 기본 뉴럴네트워크의 구조를 해당 챕터에서는 간략화하여 다음처럼 표현



순서 데이터의 예측문제

- 처음부터 지금까지 측정된 데이터의 다음에 나올 값을 예측하고 싶음
- 입력값: 처음부터 t 시점까지 입력값



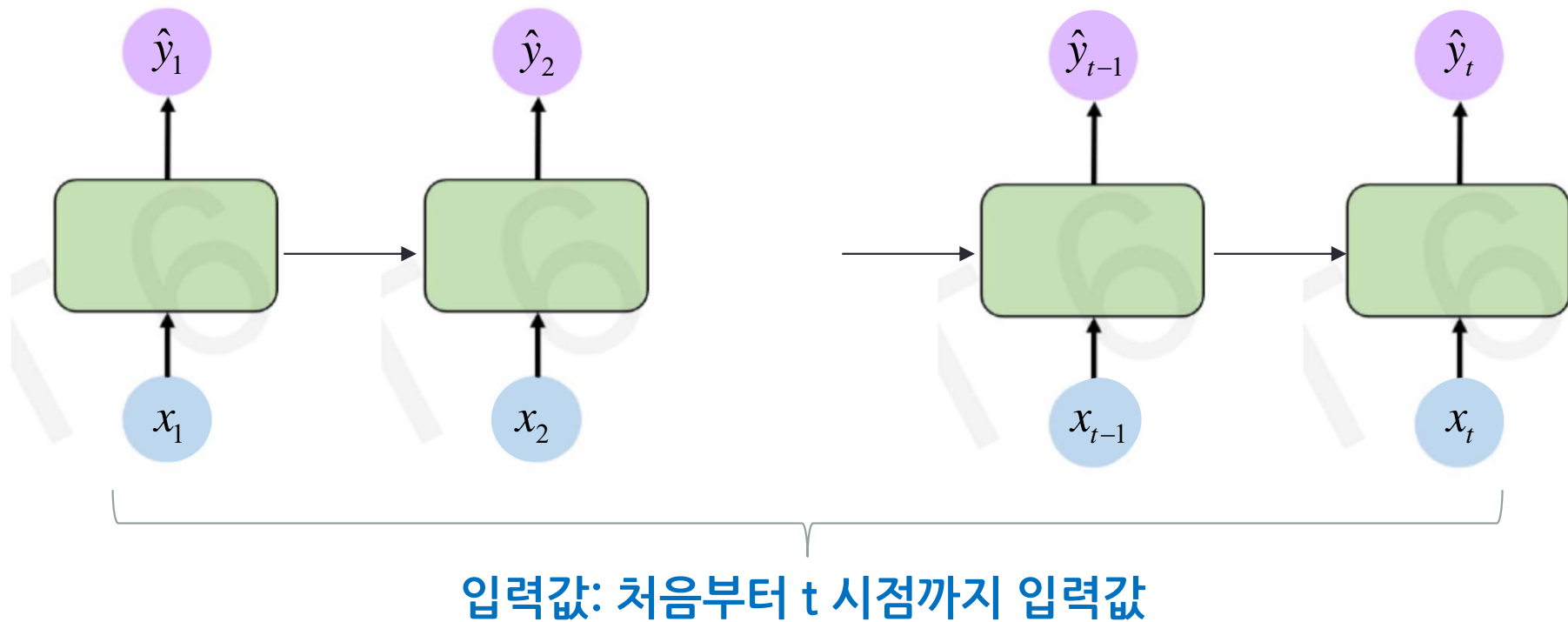
- 출력값: t 시점의 값



문제상황

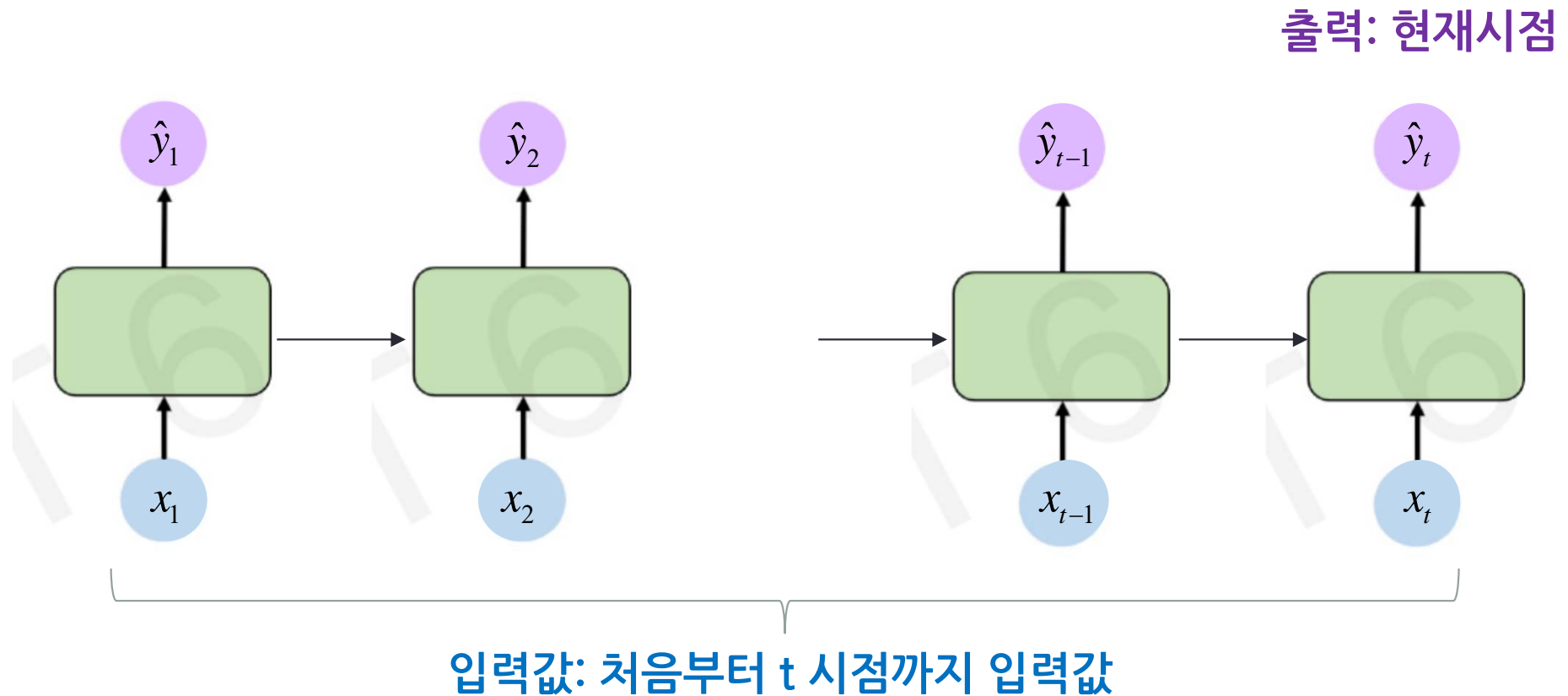
- 지금까지 뉴럴네트워크: 현재시점의 입력값 \rightarrow 현재시점의 출력값
- 순서 데이터 예측문제: 과거 시점을 포함한 여러개의 입력값 \rightarrow 현재시점의 출력값
- 과거의 데이터 입력되었던 데이터를 현재시점에 반영하려면 어떻게 해야할까?

출력: 현재시점



아이디어: 과거의 상태를 기억할 수 있도록 연결

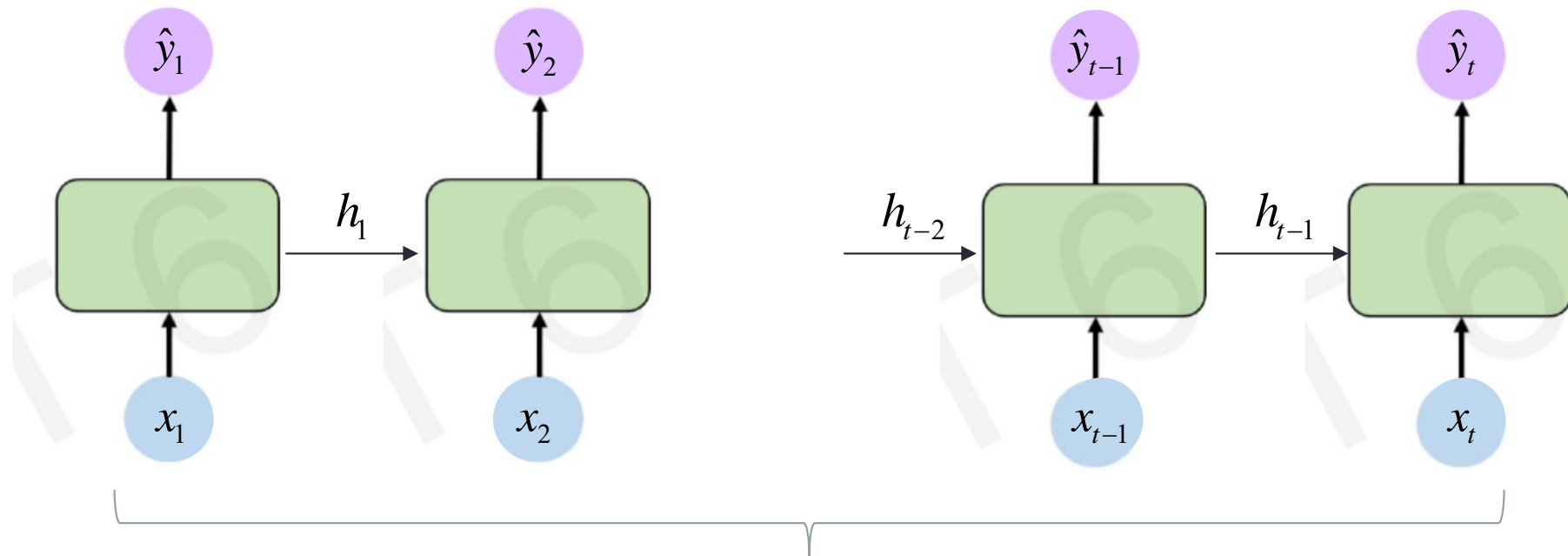
- \hat{y}_t 계산시점에서 $t-1$ 까지 일어난 일을 기억하는 변수를 입력값으로 사용!



아이디어: 과거의 상태를 기억할 수 있도록 연결

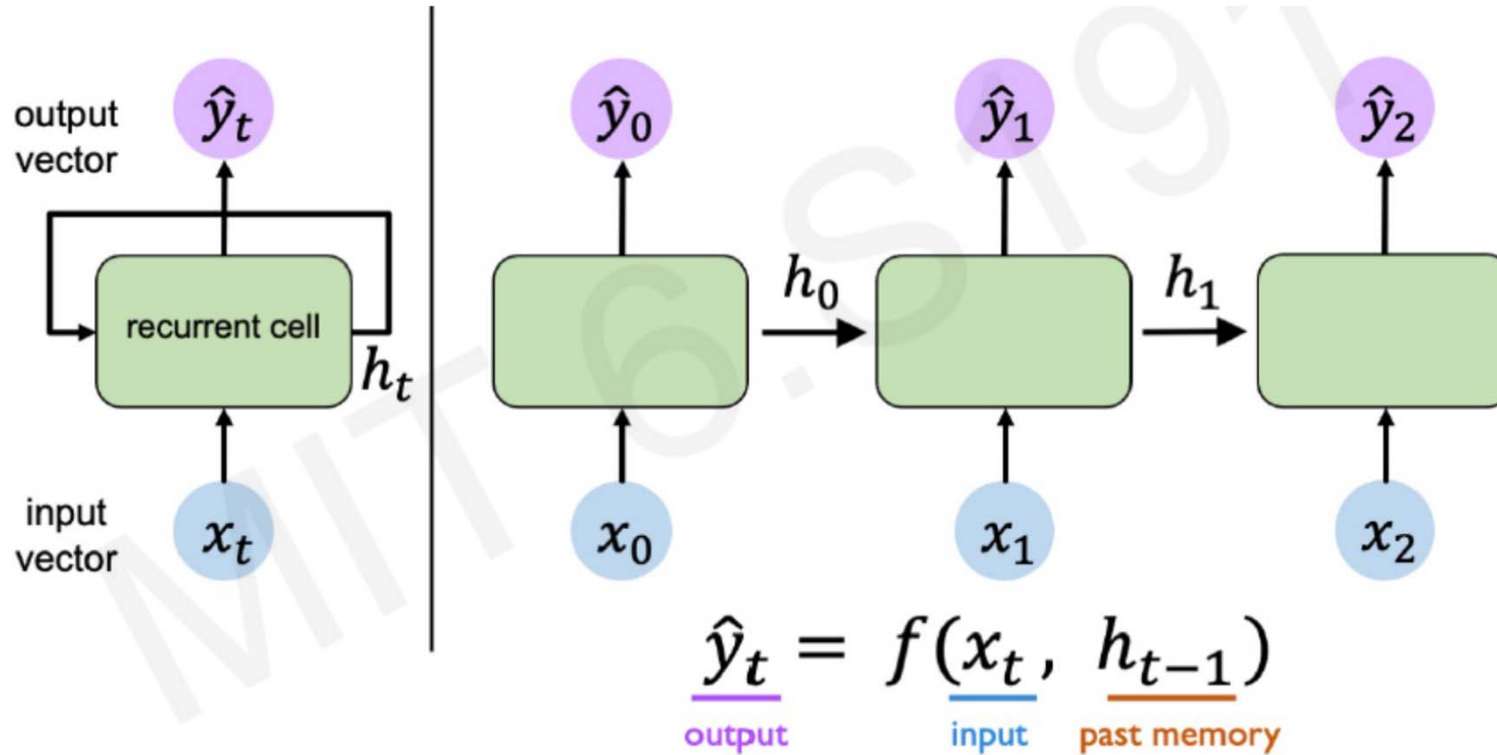
- \hat{y}_t 계산시점에서 t-1까지 일어난 일을 기억하는 변수를 입력값으로 사용!
- h_{t-1} (hidden state at time t): t-1 시점까지 일어난 일을 기억하는 변수

출력: 현재시점



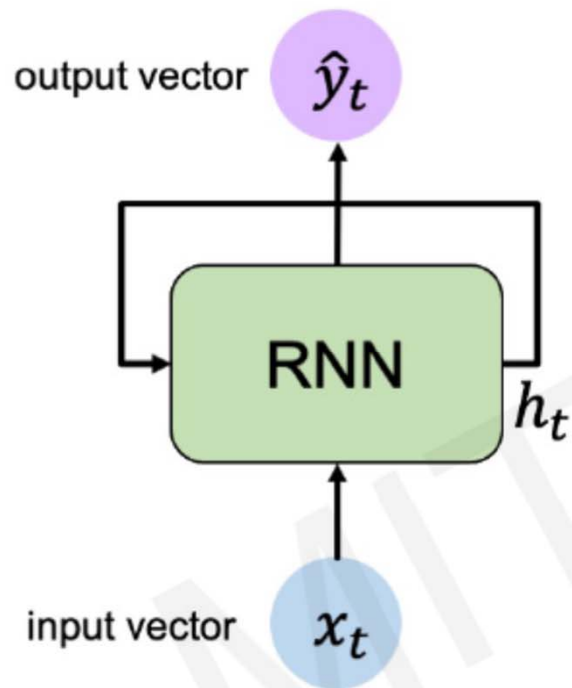
입력값: 처음부터 t 시점까지 입력값

아이디어: Hidden State를 활용한 Recurrence



RNN의 계산순서

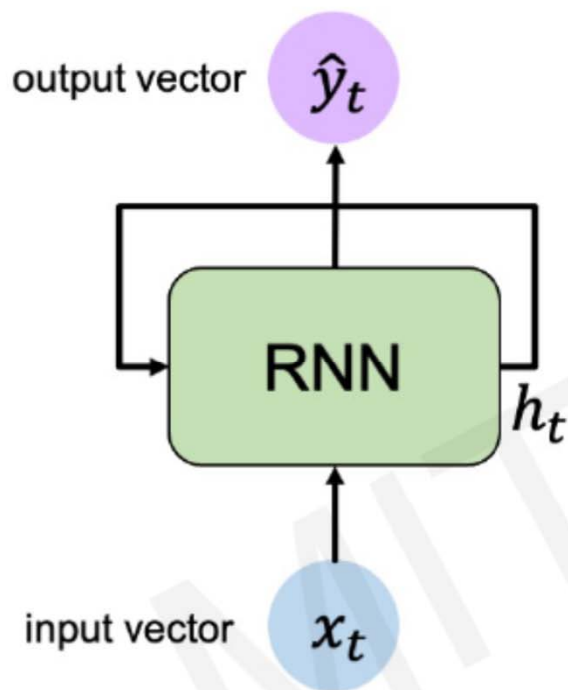
- t 시점에서의 입력값 주어짐



Input Vector
 x_t

RNN의 계산순서

- h_t (hidden state at time t) 업데이트
 - 입력값1: t-1까지 누적된 정보 (h_{t-1}), 가중치 W_{hh}
 - 입력값2: t 시점에서 들어오는 입력값 (x_t), 가중치 W_{xh}



Update Hidden State

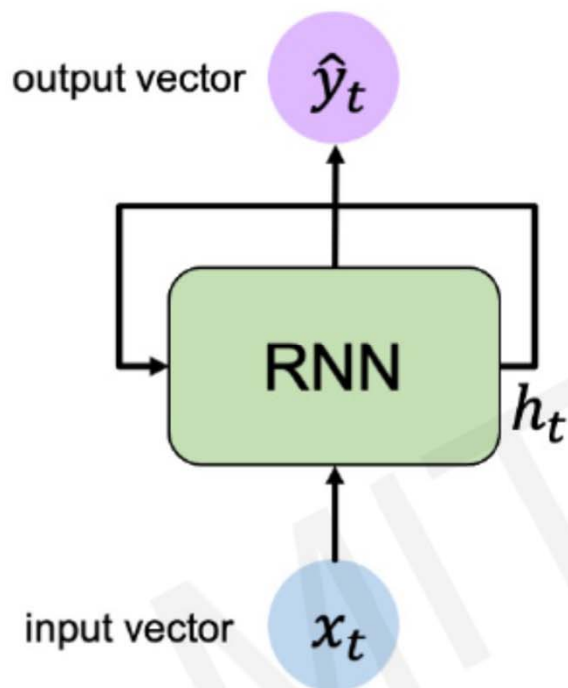
$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

Input Vector

x_t

RNN의 계산순서

- \hat{y}_t 예측
 - 입력값: 업데이트된 h_t , 가중치 W_{hy} ,



Output Vector

$$\hat{y}_t = W_{hy}^T h_t$$

Update Hidden State

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

Input Vector

x_t

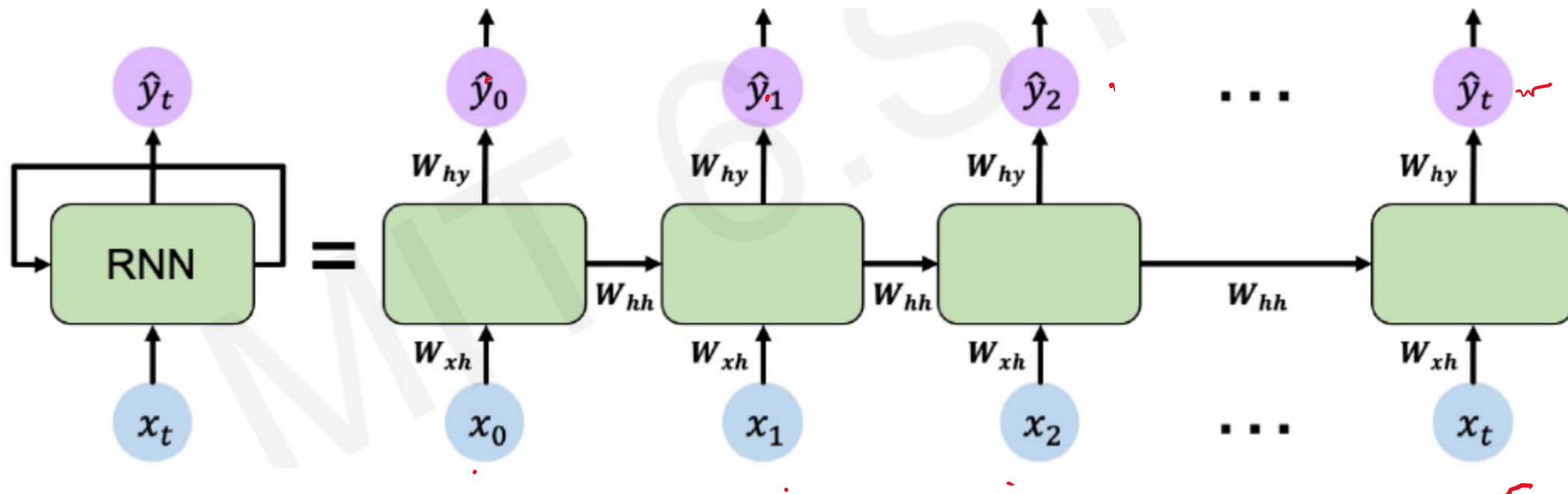
RNN 계산순서

- 학습해야 하는 가중치

$$W_{xh} \quad W_{hh} \quad W_{hy}$$

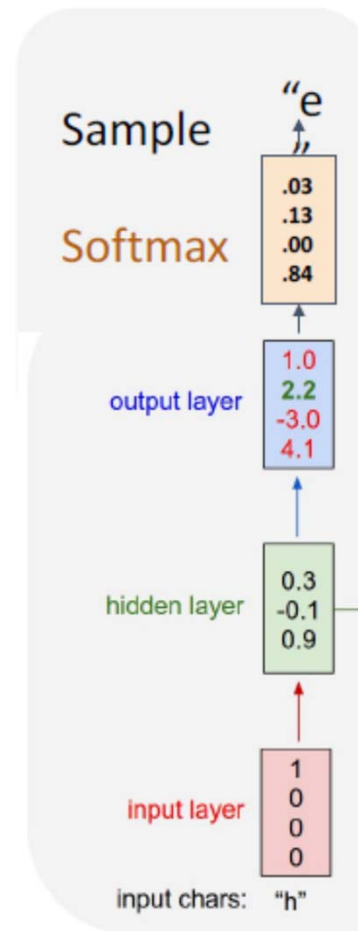
- 주의점

- 매시점에서 가중치는 모두 같다!
- 따라서 입력 값의 길이가 다르더라도 똑같은 구조를 반복하여 전개하면 됨!



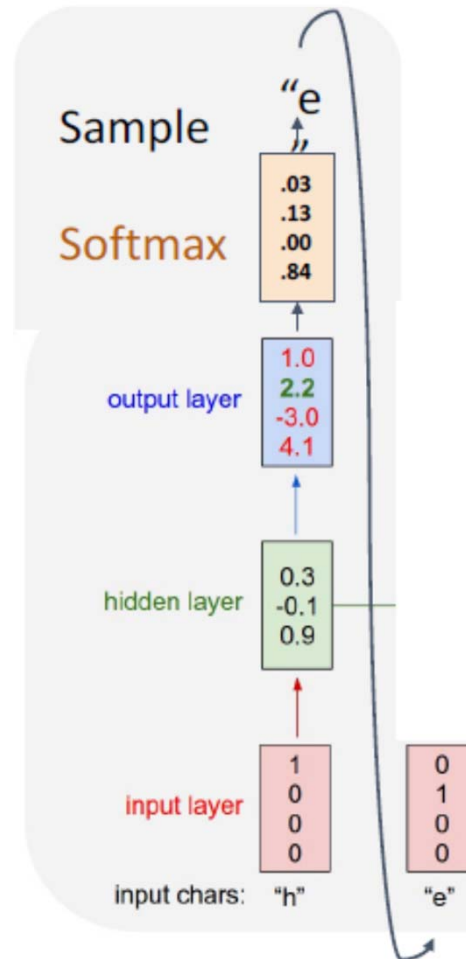
RNN을 이용한 글자 생성 예시

- Recurrence relation을 이용해 이전 step의 output이 다음 step의 input이 되도록 함
- 매 시점에서 한글자씩 샘플링 하여 다음 step의 input으로 활용



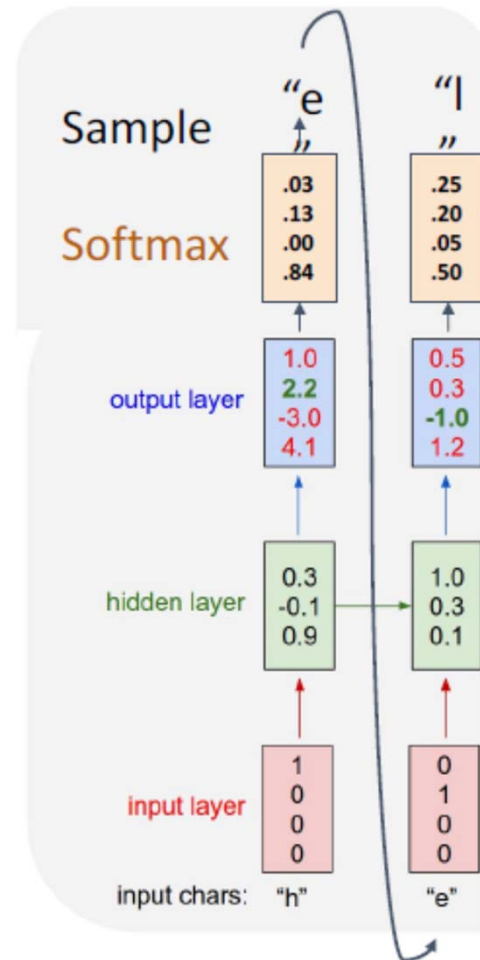
RNN을 이용한 텍스트 생성 예시

- 첫 글자로 다음 글자들을 생성시킬 수 있다.
- 매 시점에서 한글자씩 샘플링 하여 다음 step의 input으로 활용



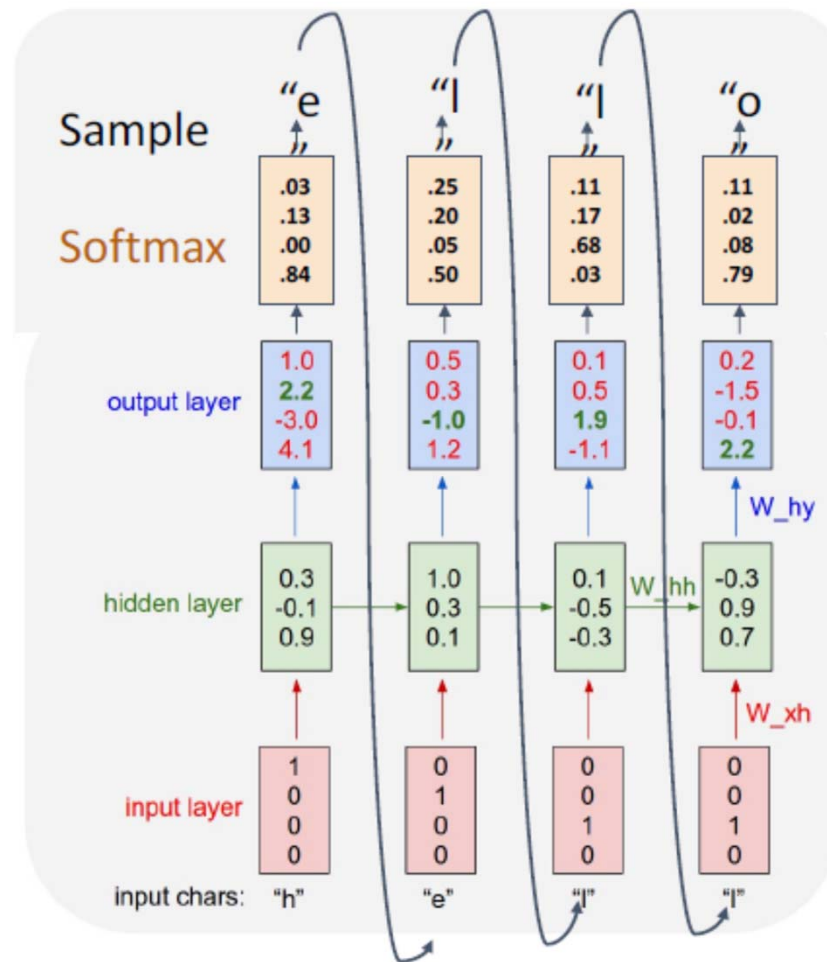
RNN을 이용한 텍스트 생성 예시

- 첫 글자로 다음 글자들을 생성시킬 수 있다.
- 매 시점에서 한글자씩 샘플링 하여 다음 step의 input으로 활용



RNN을 이용한 텍스트 생성 예시

- 첫 글자로 다음 글자들을 생성시킬 수 있다.
- 매 시점에서 한글자씩 샘플링 하여 다음 step의 input으로 활용



RNN 응용 예시: 문장 분류

- 문장 분류
 - 고객의 리뷰 데이터를 긍정/부정/중립 등으로 나누고 싶다.
 - 배달의 민족의 고객리뷰
 - 네이버 영화의 고객 리뷰

동영상 2건



클레멘타인

클레멘타인

평점

[더보기 >](#)

네타즌·관람객 평점 >

★★★★★ 9.37 참여 22,809명

기자·평론가 평점 >

★★★★★ 0.00 참여 0명

내 평점 등록하기 >

등록한 평점이 없습니다.

[등록하기](#)

한줄평 | 총 22,694건

★★★★★ 10 이 영화를 보고 맘이 나았습니다.

maan**** | 2013.07.09 14:25 | 신고

[👍 23512](#)

[👎 663](#)

★ ★ ★ ★ ★ 1 이것은절대1점이아니다11점을주고싶은 내마음이다

joke**** | 2013.07.05 01:41 | 신고

[👍 19945](#)

[👎 931](#)

★★★★★ 10 모니터도 울고 외장하드도 울고 숨어있던 바이러스도 울었다

샤브레(dccm****) | 2013.05.27 20:57 | 신고

[👍 15040](#)

[👎 370](#)

★★★★★ 10 당신이 이 영화를 보지 않았다면 아직 살아있을 이유 하나를 간직하고 있는 것이다.

그라운드(hank****) | 2013.06.05 18:20 | 신고

[👍 13999](#)

[👎 394](#)

★★★★★ 10 영화계엔 BC와 AC가 있다. Before Clementain, After Clementain...

Kyle(pool****) | 2013.06.16 01:38 | 신고

[👍 11544](#)

[👎 256](#)

RNN 응용 예시: 문장 분류

- 감성분류
 - 고객의 리뷰 데이터를 긍정/부정/중립 등으로 나누고 싶다.
 - 배달의 민족의 고객리뷰
 - 네이버 영화의 고객 리뷰
- 입력값
 - 순서대로 입력되는 단어
- 출력값
 - 긍정, 부정, 중립
 - 분류문제

동영상 2건



클레멘타인

클레멘타인

평점

[더보기 >](#)

네티즌·관람객 평점 > ★★★★★ 9.37 참여 22,809명	기자·평론가 평점 > ★★★★★ 0.00 참여 0명	내 평점 등록하기 > 등록한 평점이 없습니다. 등록하기
---------------------------------------	---------------------------------	---

한줄평 | 총 22,694건

★★★★★ 10 이 영화를 보고 맘이 나았습니다.

maan**** | 2013.07.09 14:25 | 신고

👍 23512

👎 663

★ ★ ★ ★ ★ 1 이것은절대1점이아니다11점을주고싶은 내마음이다

joke**** | 2013.07.05 01:41 | 신고

👍 19945

👎 931

★★★★★ 10 모니터도 울고 외장하드도 울고 숨어있던 바이러스도 울었다

샤브레(dccm****) | 2013.05.27 20:57 | 신고

👍 15040

👎 370

★★★★★ 10 당신이 이 영화를 보지 않았다면 아직 살아있을 이유 하나를 간직하고 있는 것이다.

그라운드(hank****) | 2013.06.05 18:20 | 신고

👍 13999

👎 394

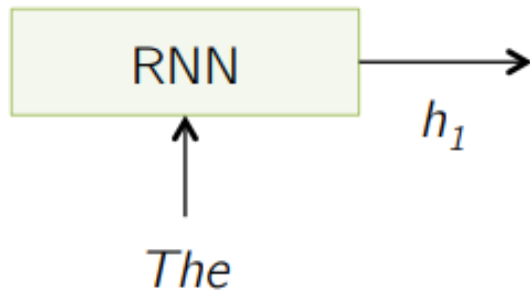
★★★★★ 10 영화계엔 BC와 AC가 있다. Before Clementain, After Clementain...

Kyle(pool****) | 2013.06.16 01:38 | 신고

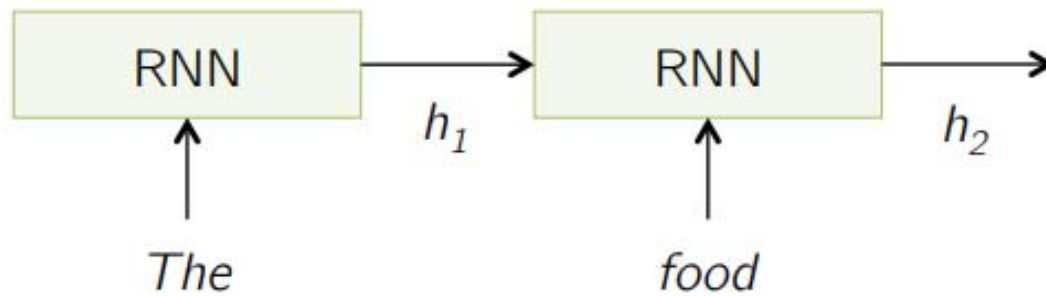
👍 11544

👎 256

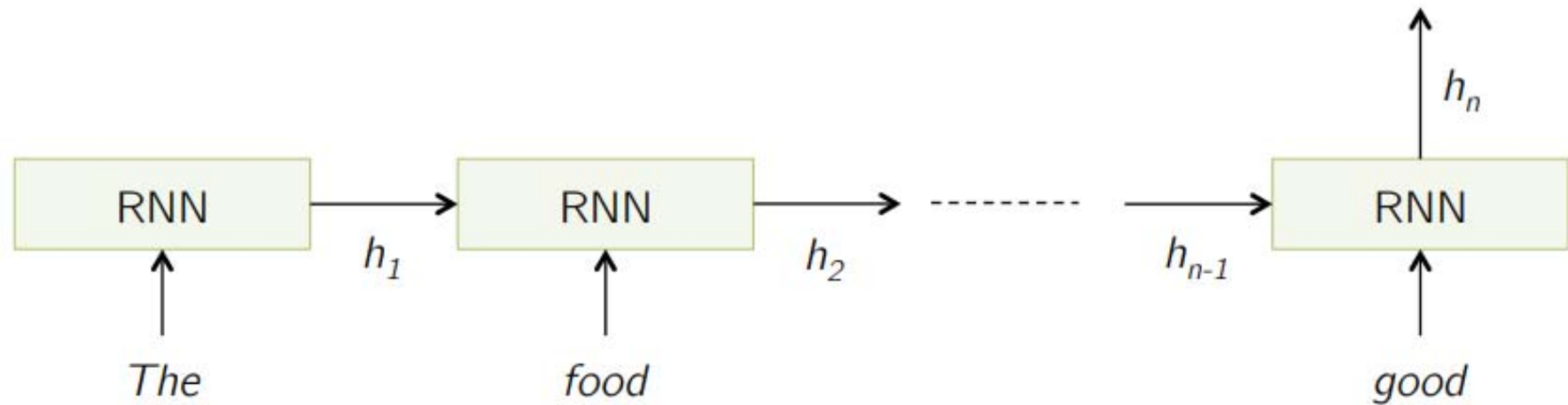
RNN 응용 예시: 문장 분류



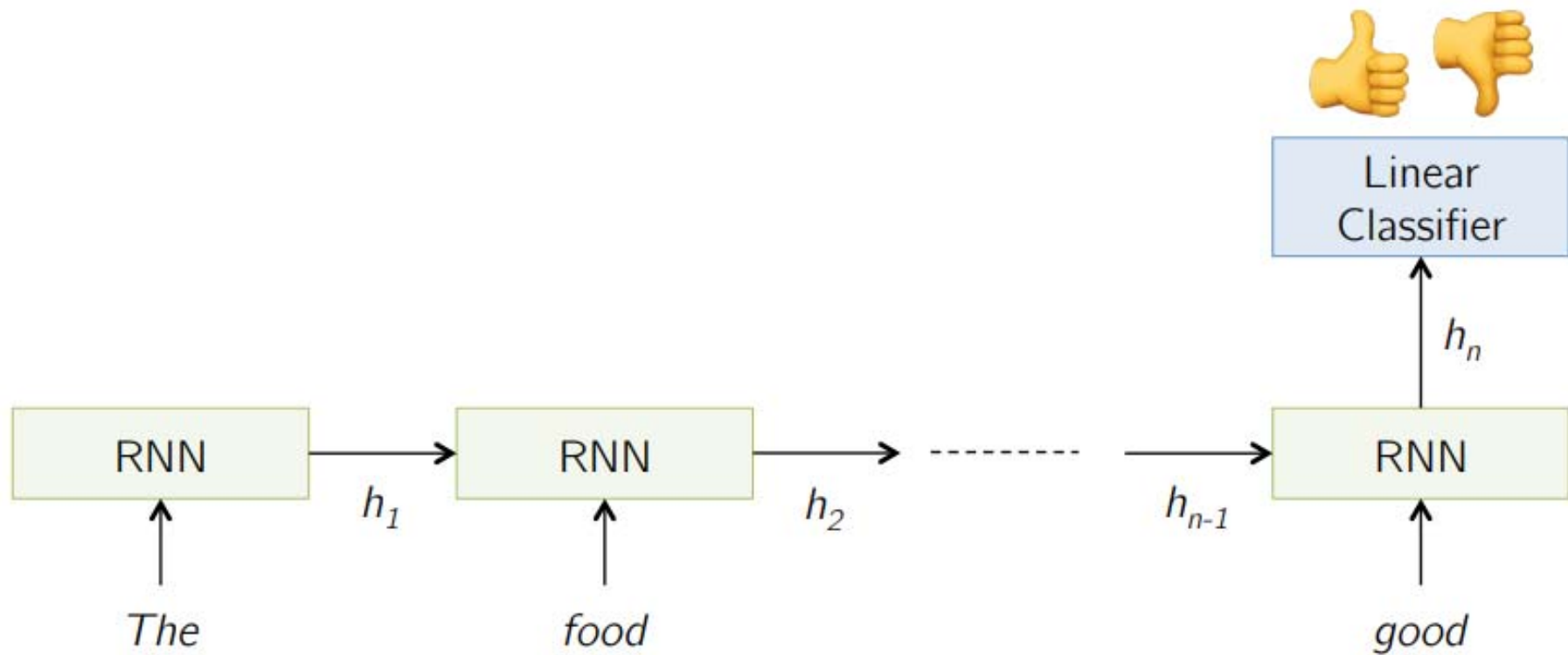
RNN 응용 예시: 문장 분류



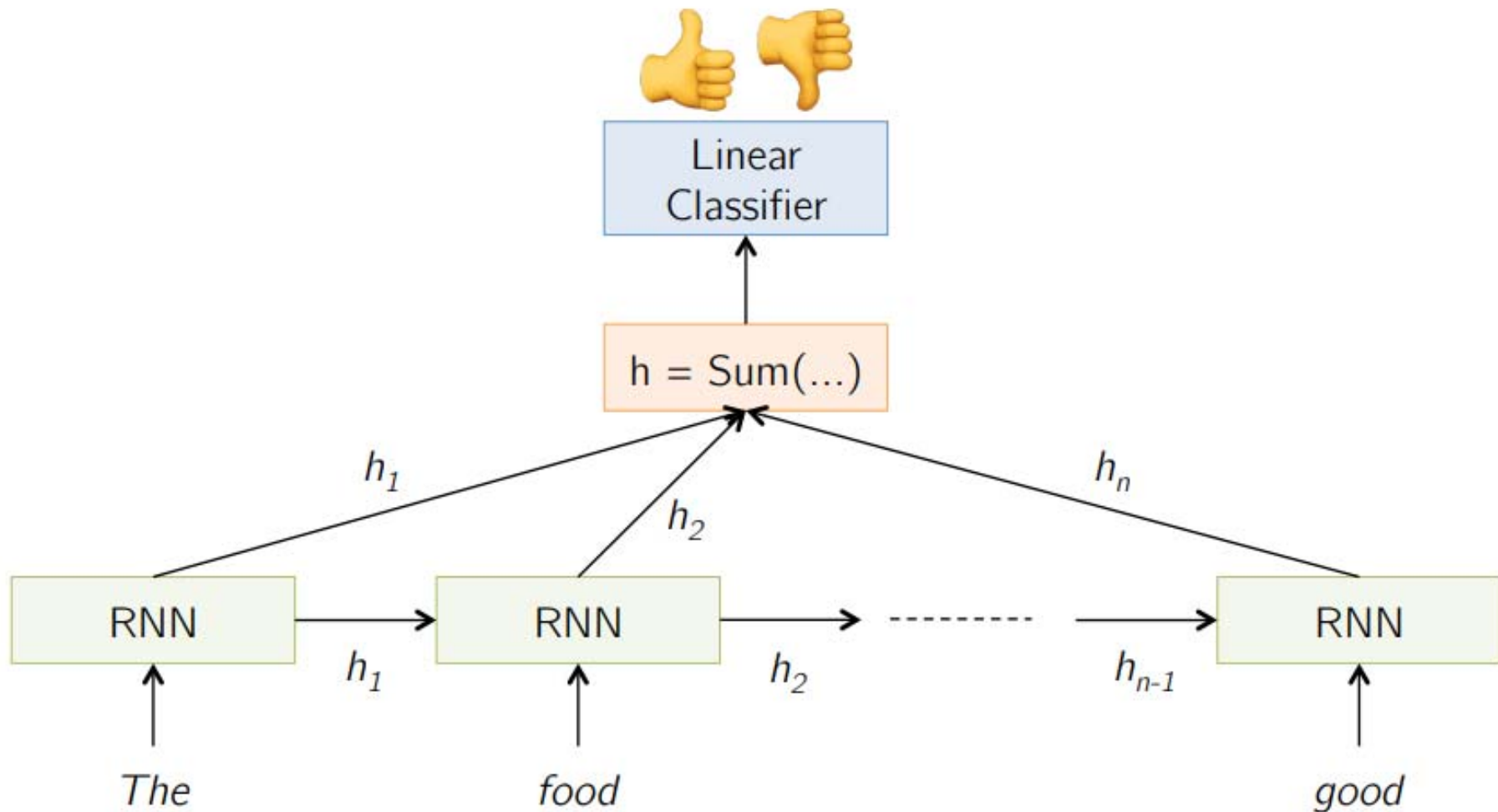
RNN 응용 예시: 문장 분류



RNN 응용 예시: 문장 분류



RNN 응용 예시: 문장 분류



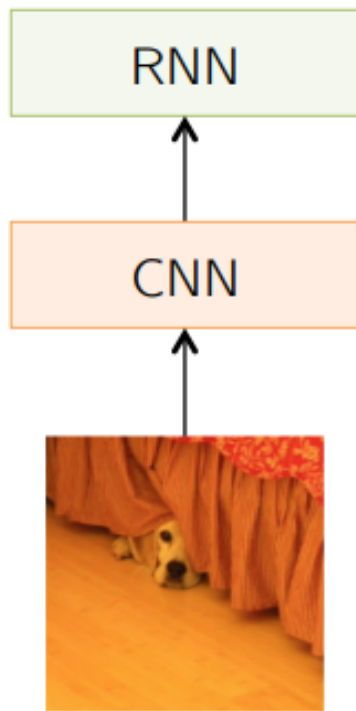
RNN 응용 예시: 이미지 주석처리

- 입력값: 이미지
- 출력값: 이미지에 대한 설명
 - 단어의 순서

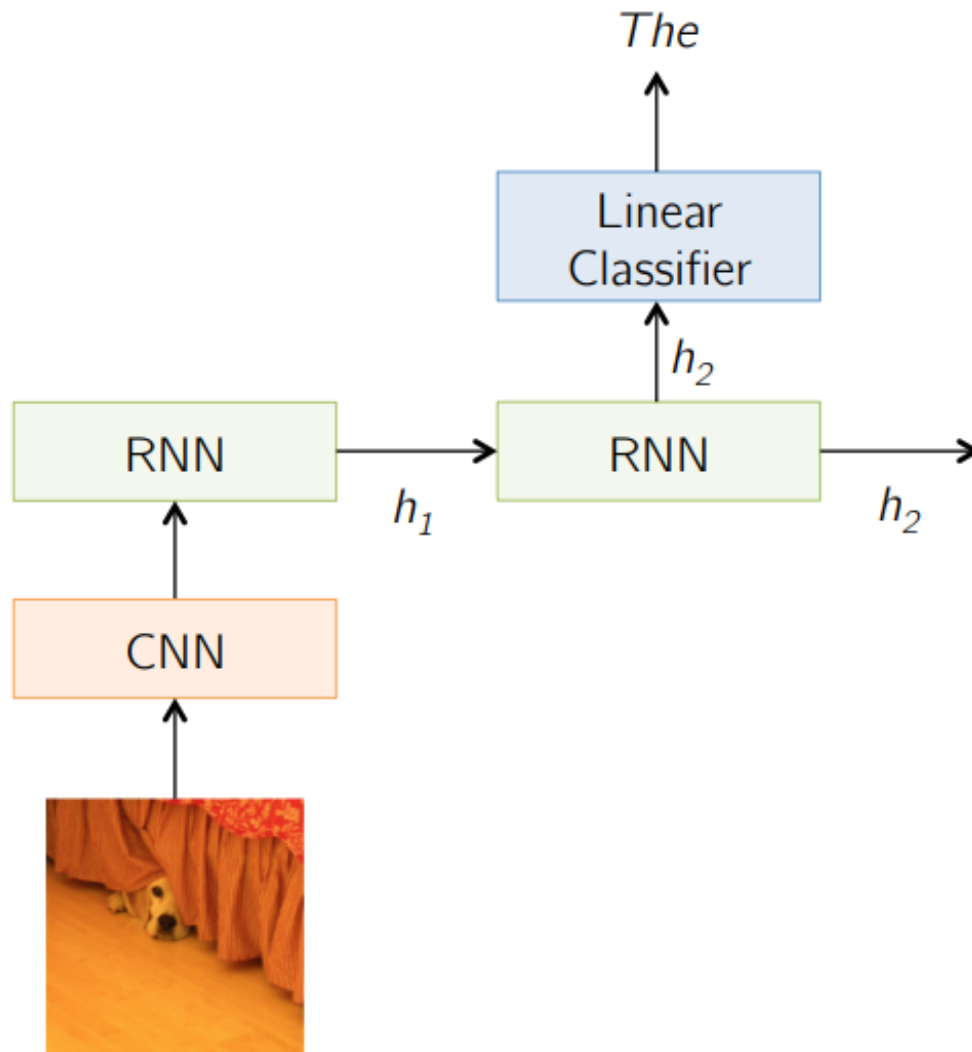


: The dog is hiding

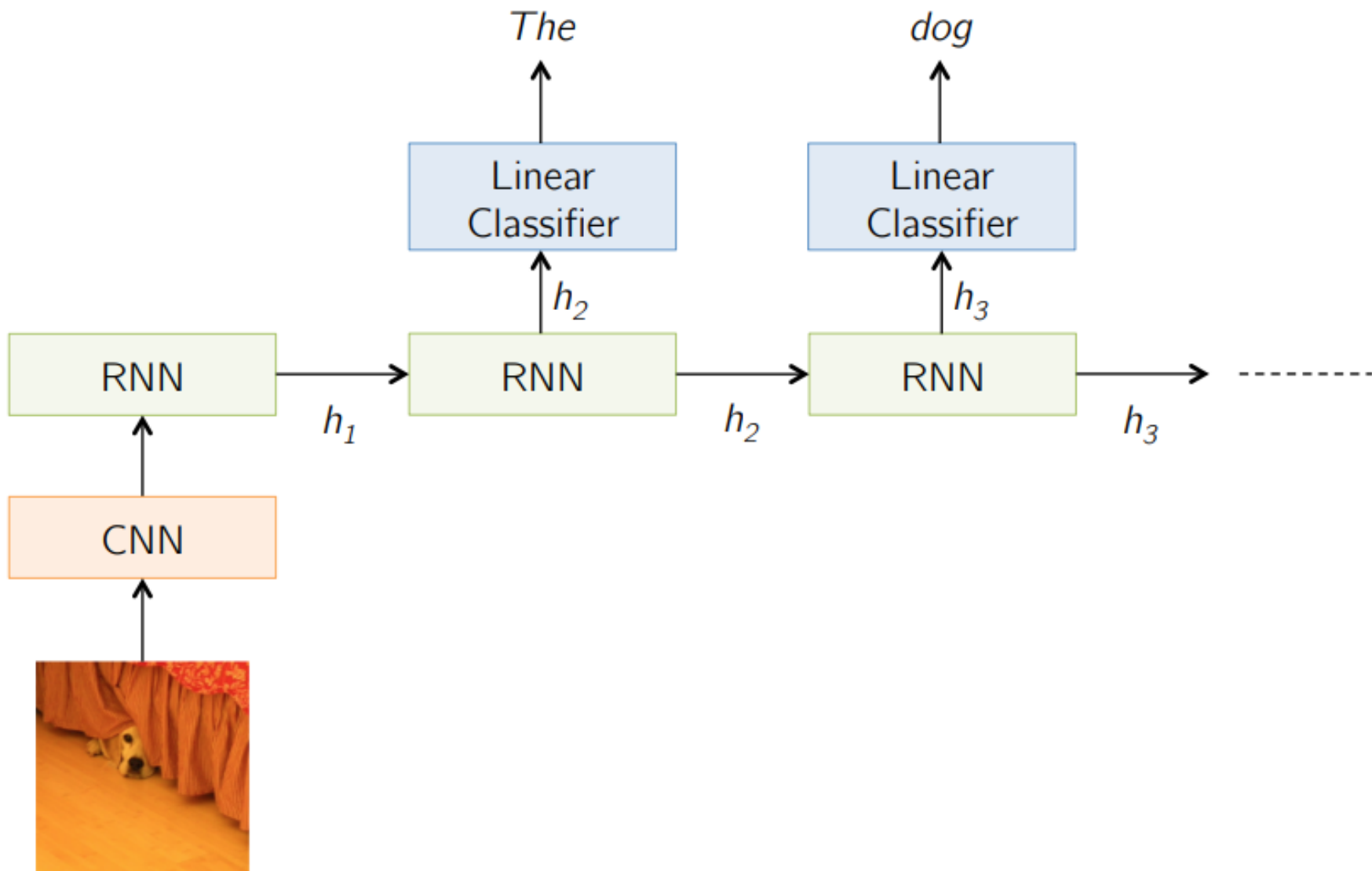
RNN 응용 예시: 이미지 주석처리



RNN 응용 예시: 이미지 주석처리



RNN 응용 예시: 이미지 주석처리



실제 이미지 주석처리 결과

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A herd of elephants walking across a dry grass field.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A close up of a cat laying on a couch.



RNN 구조의 다양한 응용

Single - Single



Feed-forward Network

Single - Multiple

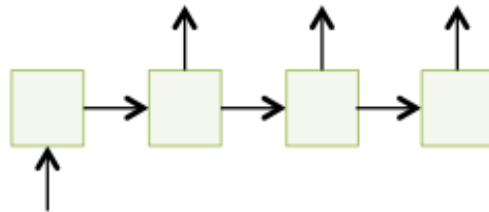
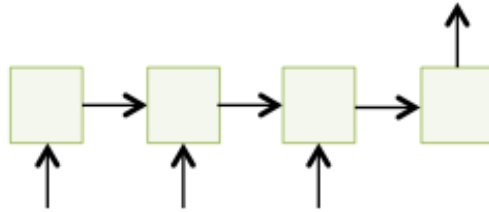


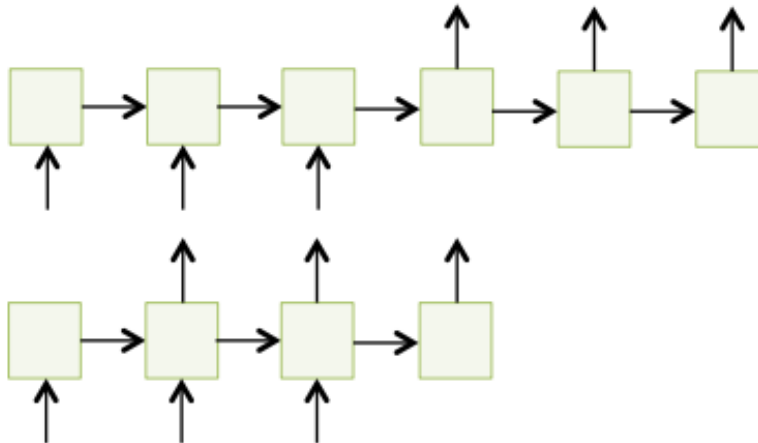
Image Captioning

Multiple - Single



Sentiment Classification

Multiple - Multiple



Translation

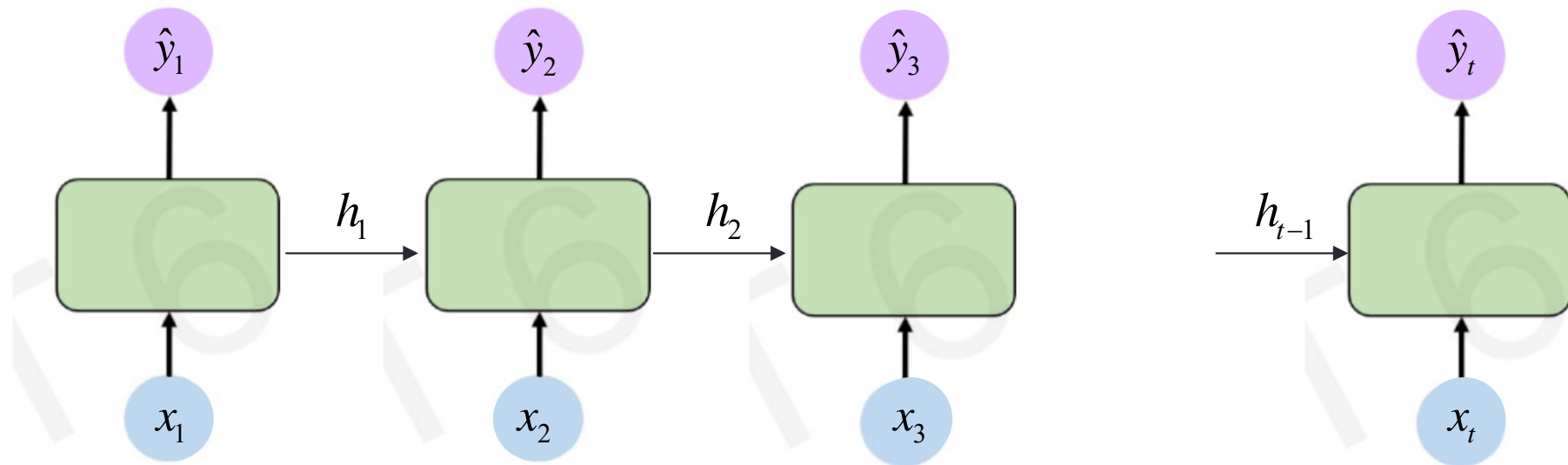
Image Captioning

03. 순서데이터 처리를 위한 딥러닝

RNN의 단점(Vanishing Gradient Problem)

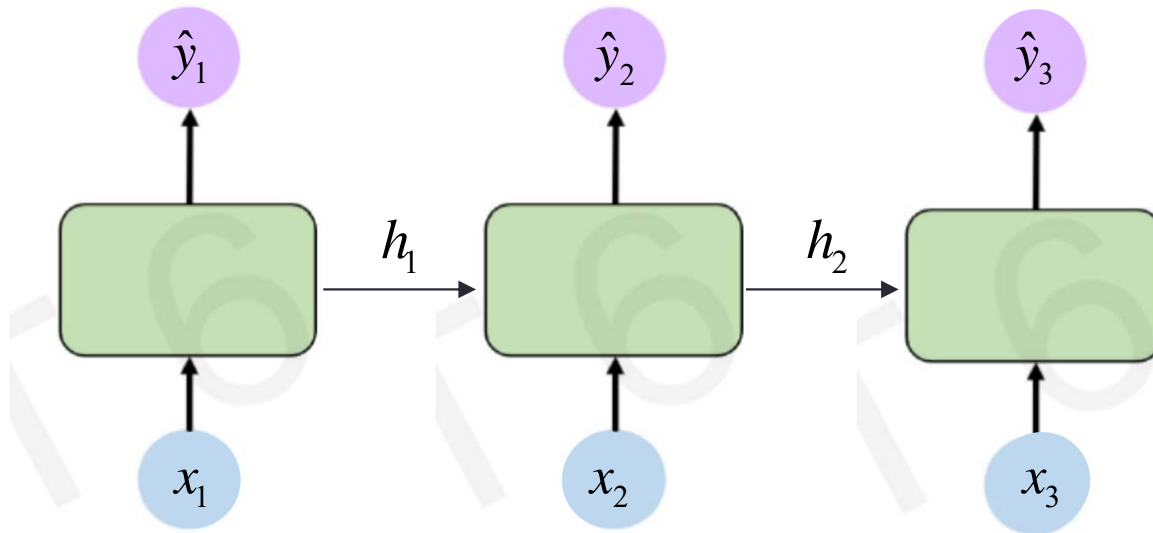
RNN의 대안: LSTM

RNN의 구조

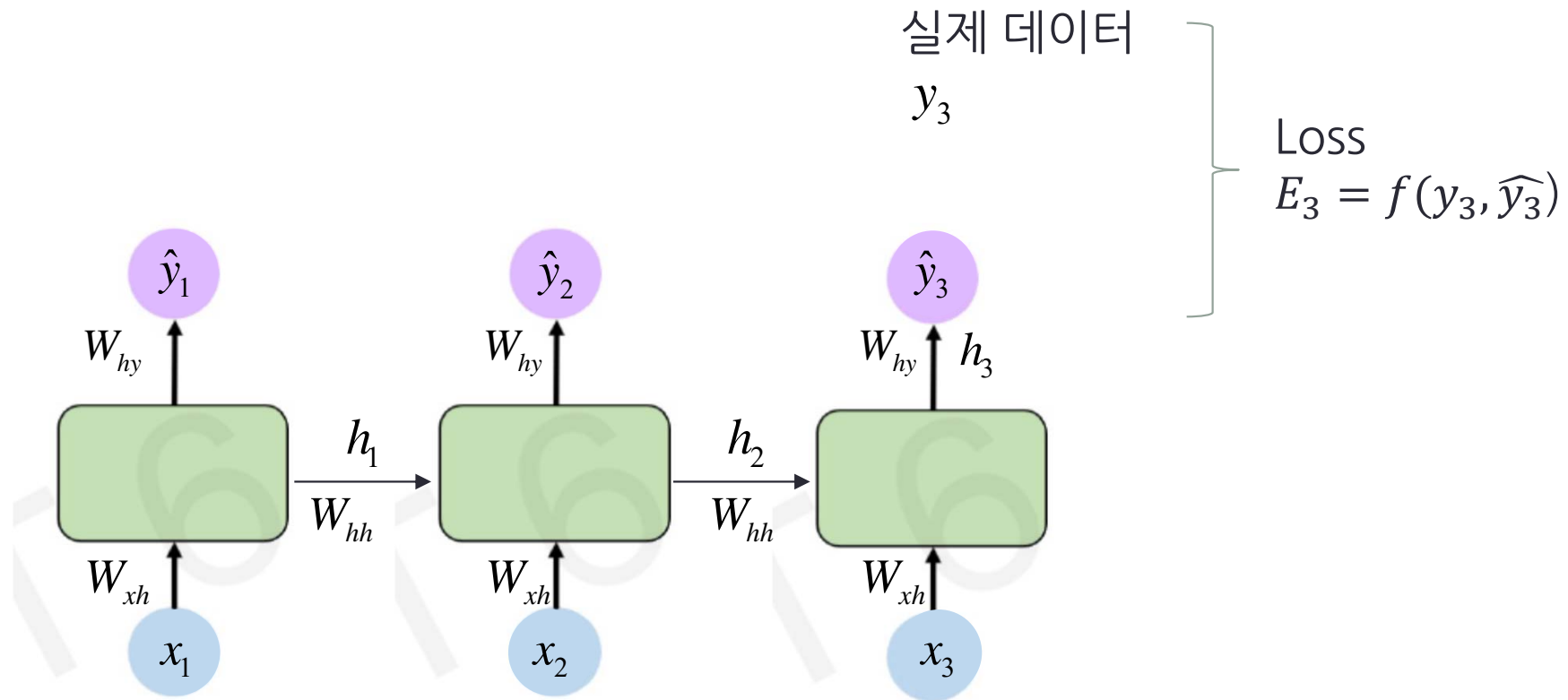


RNN의 구조

실제 데이터
 y_3

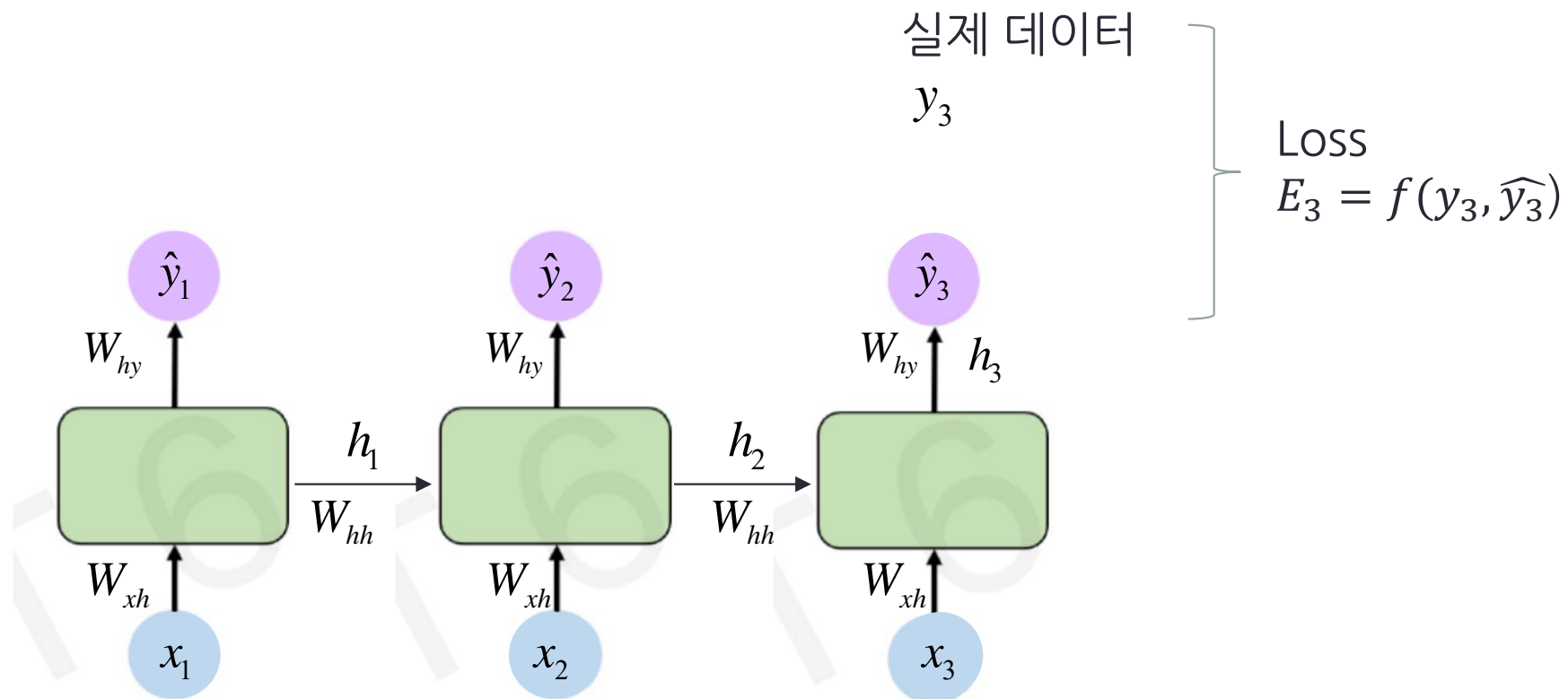


RNN에서의 가중치 업데이트



RNN에서 가중치의 업데이트

$\frac{\partial E_3}{\partial W_{hh}}$ E3에서 관측된 에러를 Backpropagation 하고 싶다.



RNN에서 가중치의 업데이트

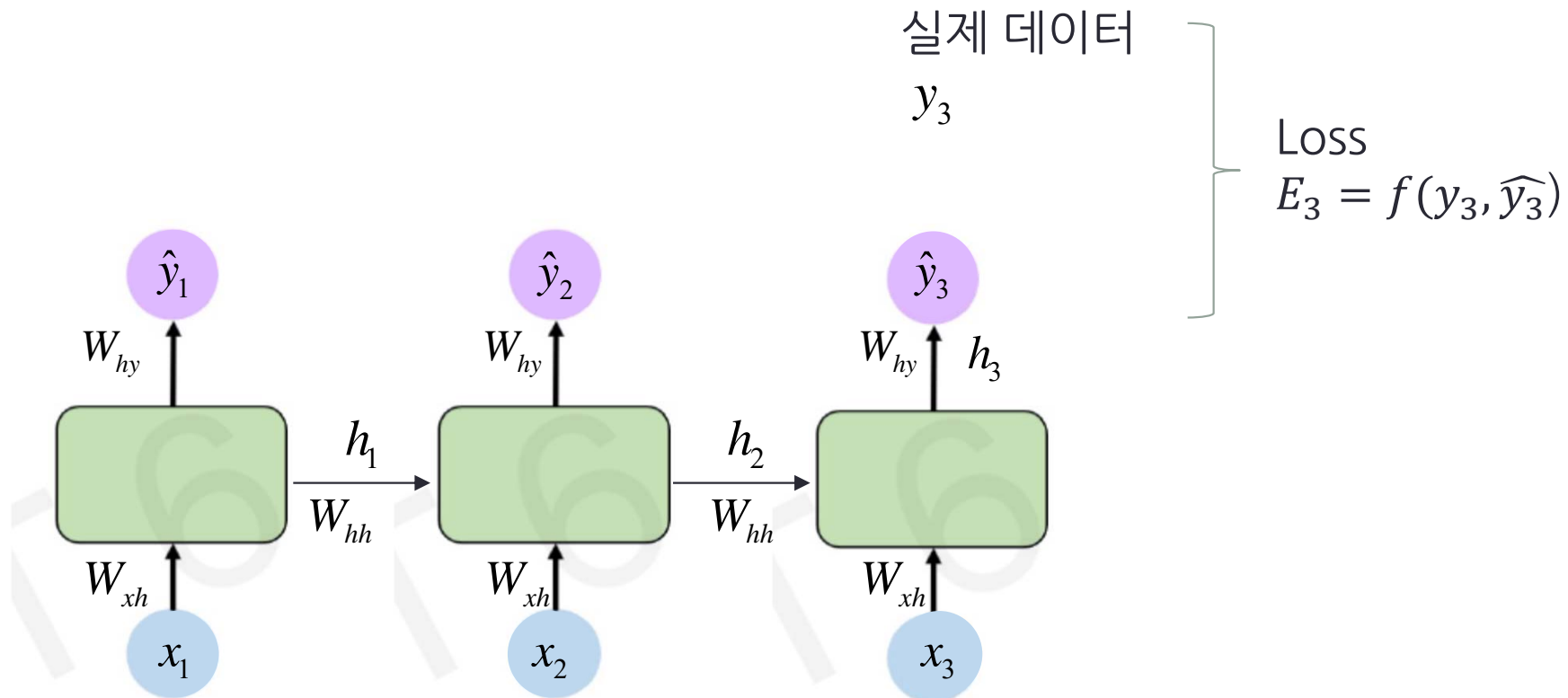
$$\frac{\partial E_3}{\partial W_{hh}}$$

E3에 영향줄수 있는 모든 경로

$$W_{hh} \rightarrow h_3 \rightarrow y_3 \rightarrow E_3$$

$$W_{hh} \rightarrow h_2 \rightarrow h_3 \rightarrow y_3 \rightarrow E_3$$

$$W_{hh} \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow y_3 \rightarrow E_3$$



RNN에서 가중치의 업데이트

$$\frac{\partial E_3}{\partial W_{hh}}$$

E3에 영향줄수 있는 모든 경로

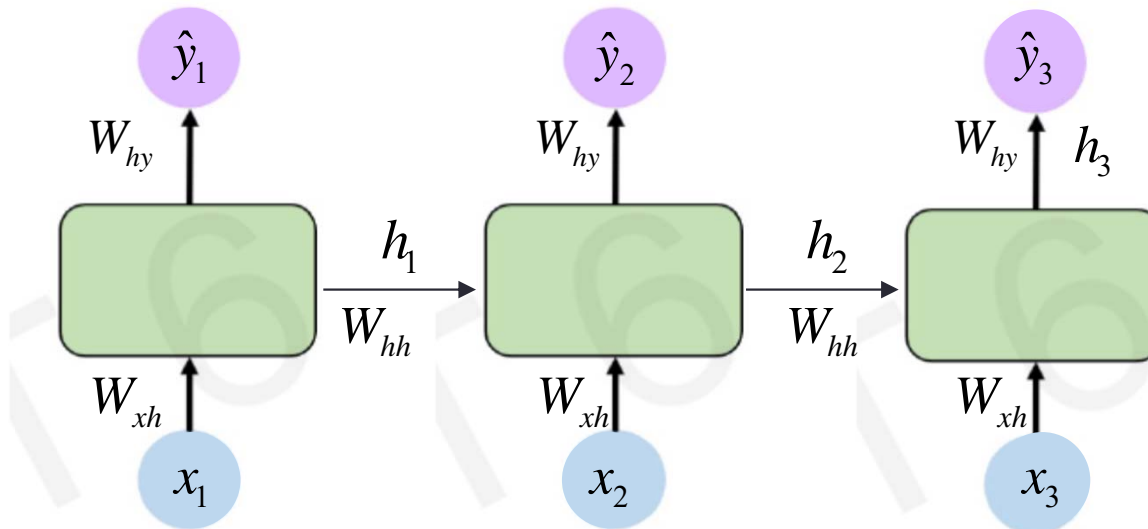
$W_{hh} \rightarrow h_3 \rightarrow y_3 \rightarrow E_3$ gradient from h3

$W_{hh} \rightarrow h_2 \rightarrow h_3 \rightarrow y_3 \rightarrow E_3$ gradient from h2

$W_{hh} \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow y_3 \rightarrow E_3$ gradient from h1

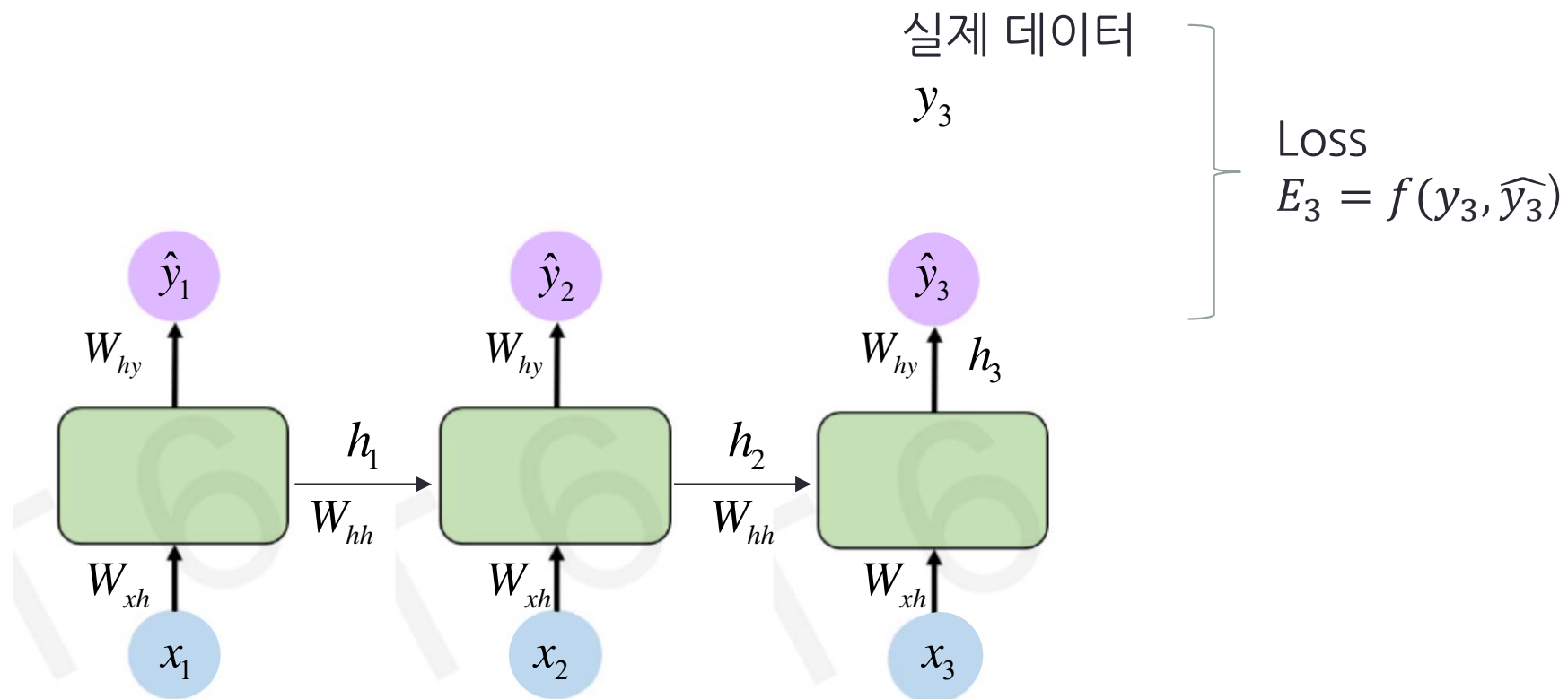
실제 데이터
 y_3

Loss
 $E_3 = f(y_3, \hat{y}_3)$



RNN에서 가중치의 업데이트

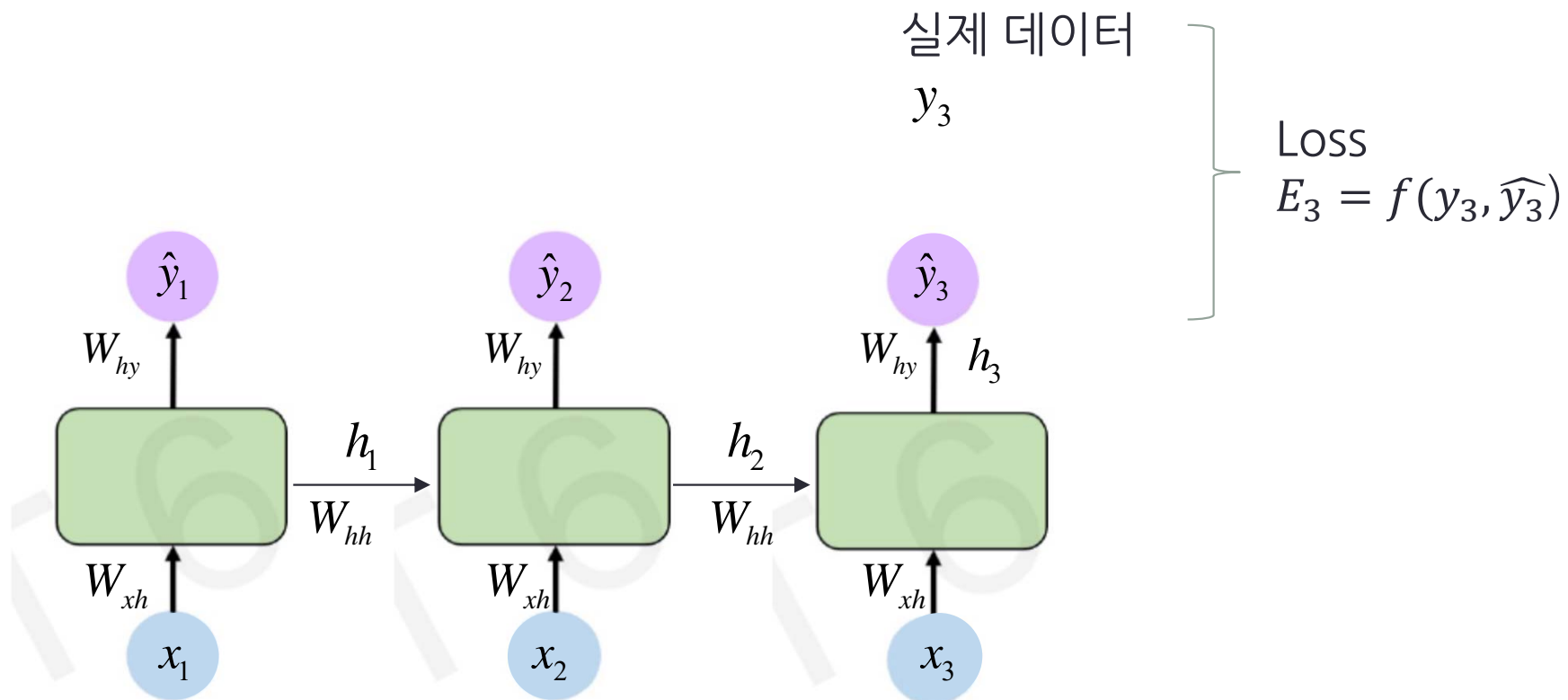
$$\frac{\partial E_3}{\partial W_{hh}} = \text{gradient from } h_3 + \text{gradient from } h_2 + \text{gradient from } h_1$$



RNN에서 가중치의 업데이트

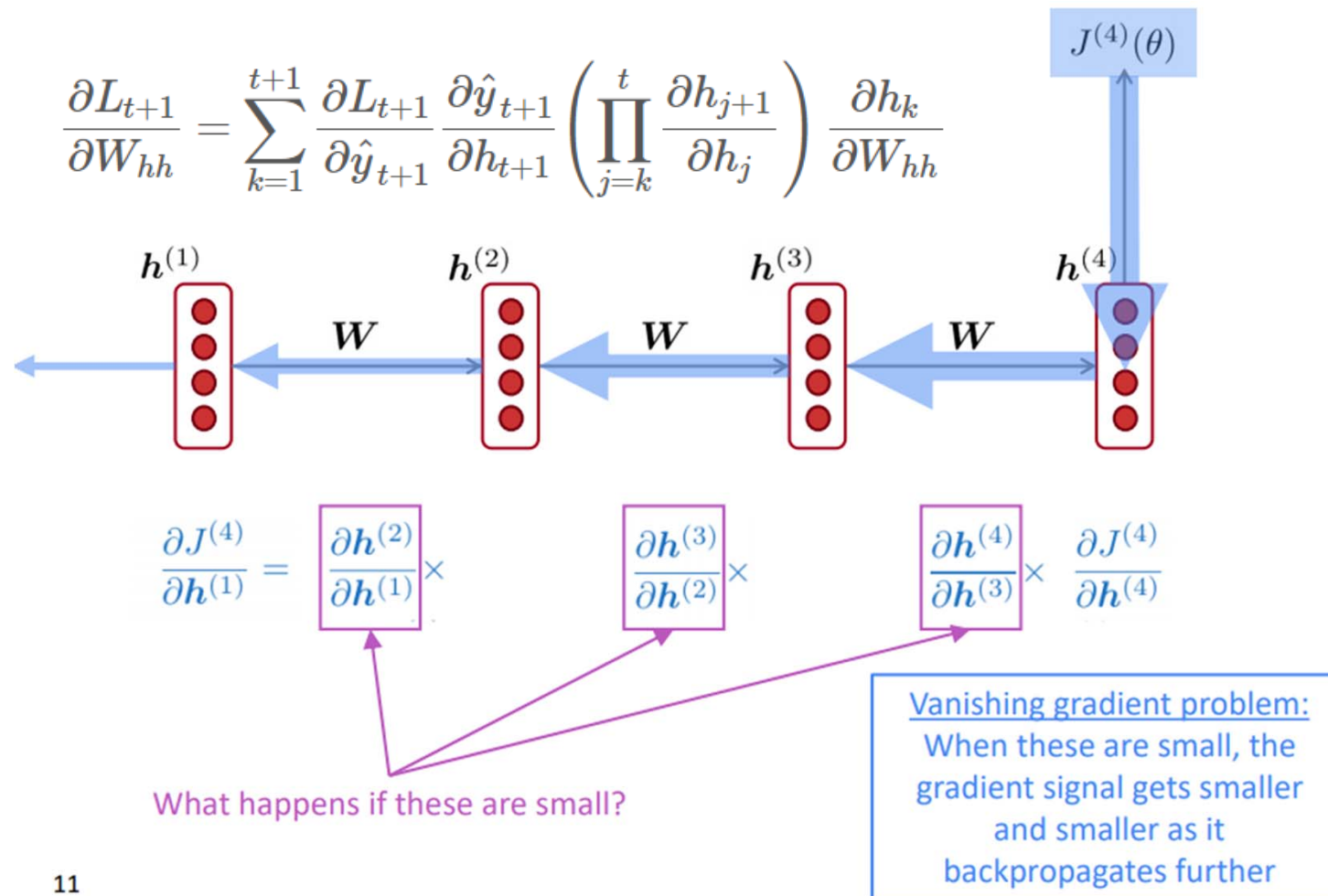
$$\frac{\partial E_3}{\partial W_{hh}} = \text{gradient from } h_3 + \text{gradient from } h_2 + \text{gradient from } h_1$$

$$\frac{\partial E_3}{\partial y_3} \frac{\partial y_3}{\partial h_3} \frac{\partial h_3}{\partial W_{hh}} + \frac{\partial E_3}{\partial y_3} \frac{\partial y_3}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W_{hh}} + \frac{\partial E_3}{\partial y_3} \frac{\partial y_3}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W_{hh}}$$



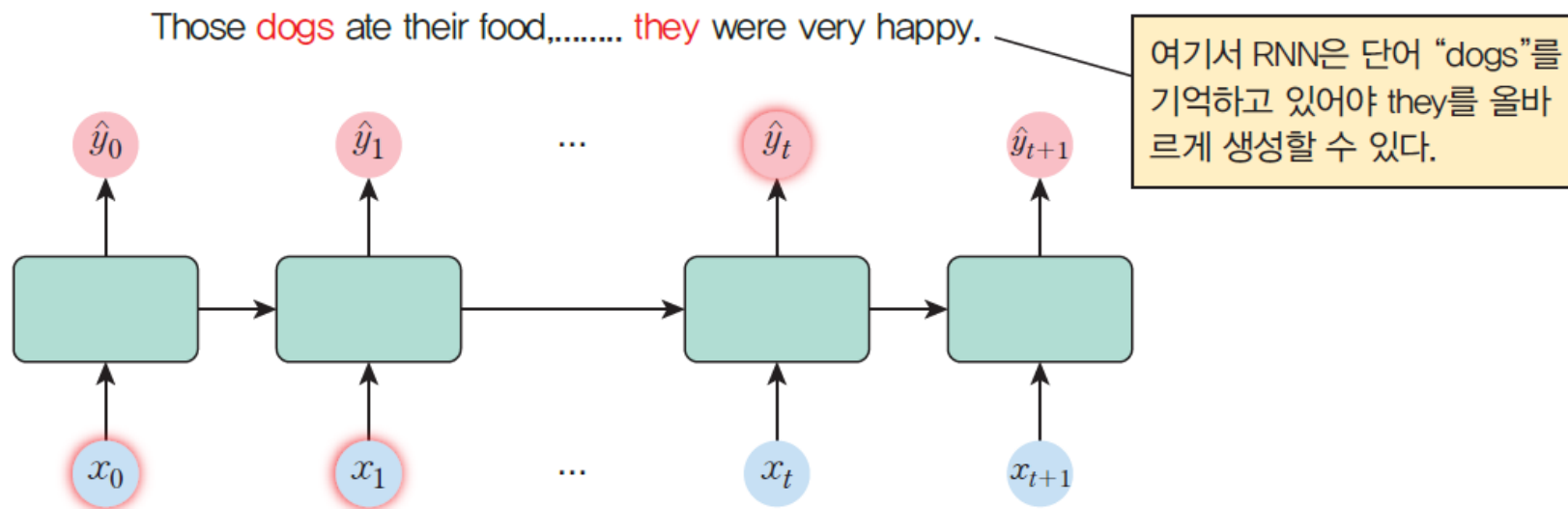
Gradient Exploding or Vanishing

- 현재 발생한 출력값을 이용해 가중치 업데이트 할 경우, 먼 과거의 일일수록 그 영향력이 급속히 감소한다.
- 단기 기억에 의존하게 됨!



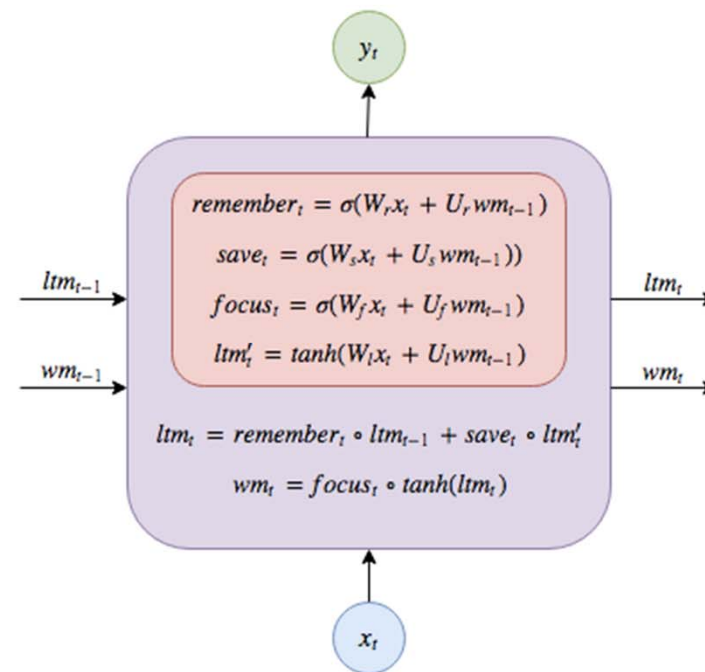
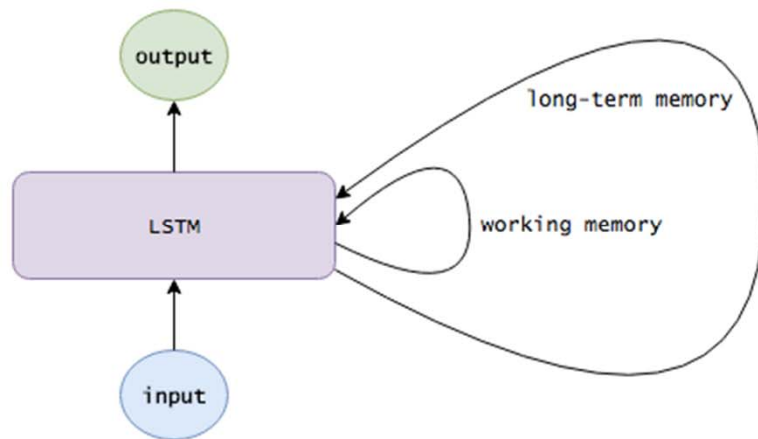
The Problem of Long-Term Dependencies

- 현재 발생한 출력값을 이용해 가중치 업데이트 할 경우, 먼 과거의 일일수록 그 영향력이 급속히 감소한다.
- 단기 기억에 의존하게 됨!



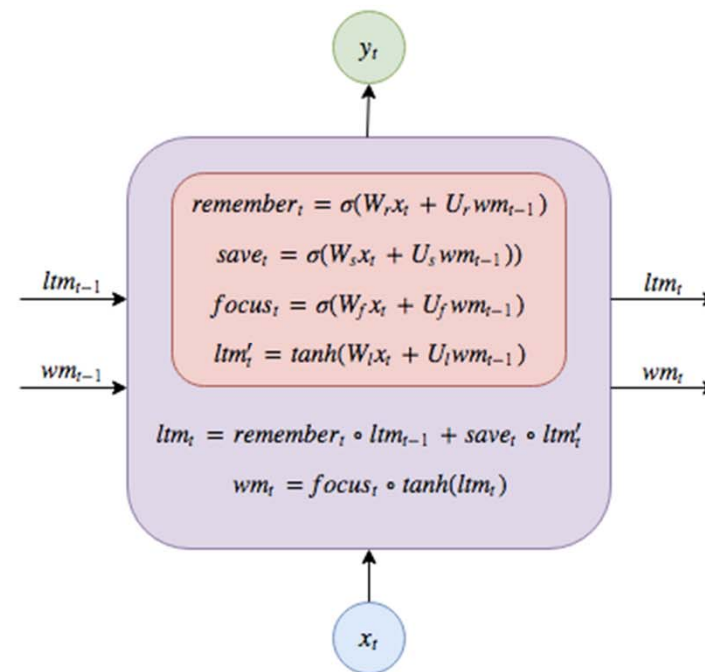
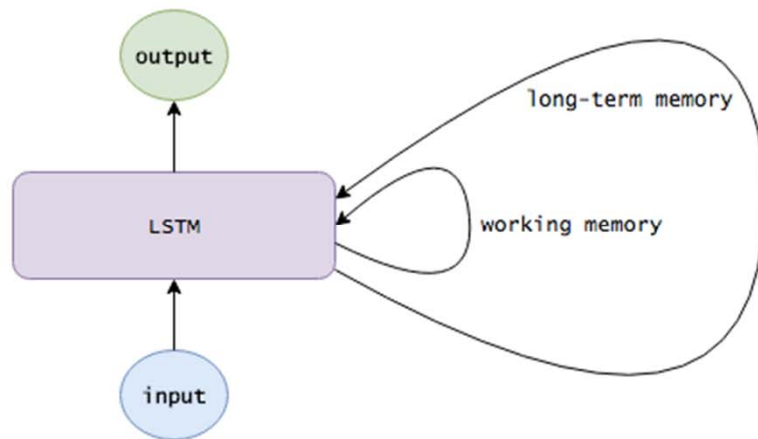
Solution: LSTM (Long-term short tem memory)

- RNN의 단기기억 의존성을 극복하기 위해 도입된 구조
 - 입력값
 - t-1 단기기억
 - t-1 장기기억
 - t 현재 입력값

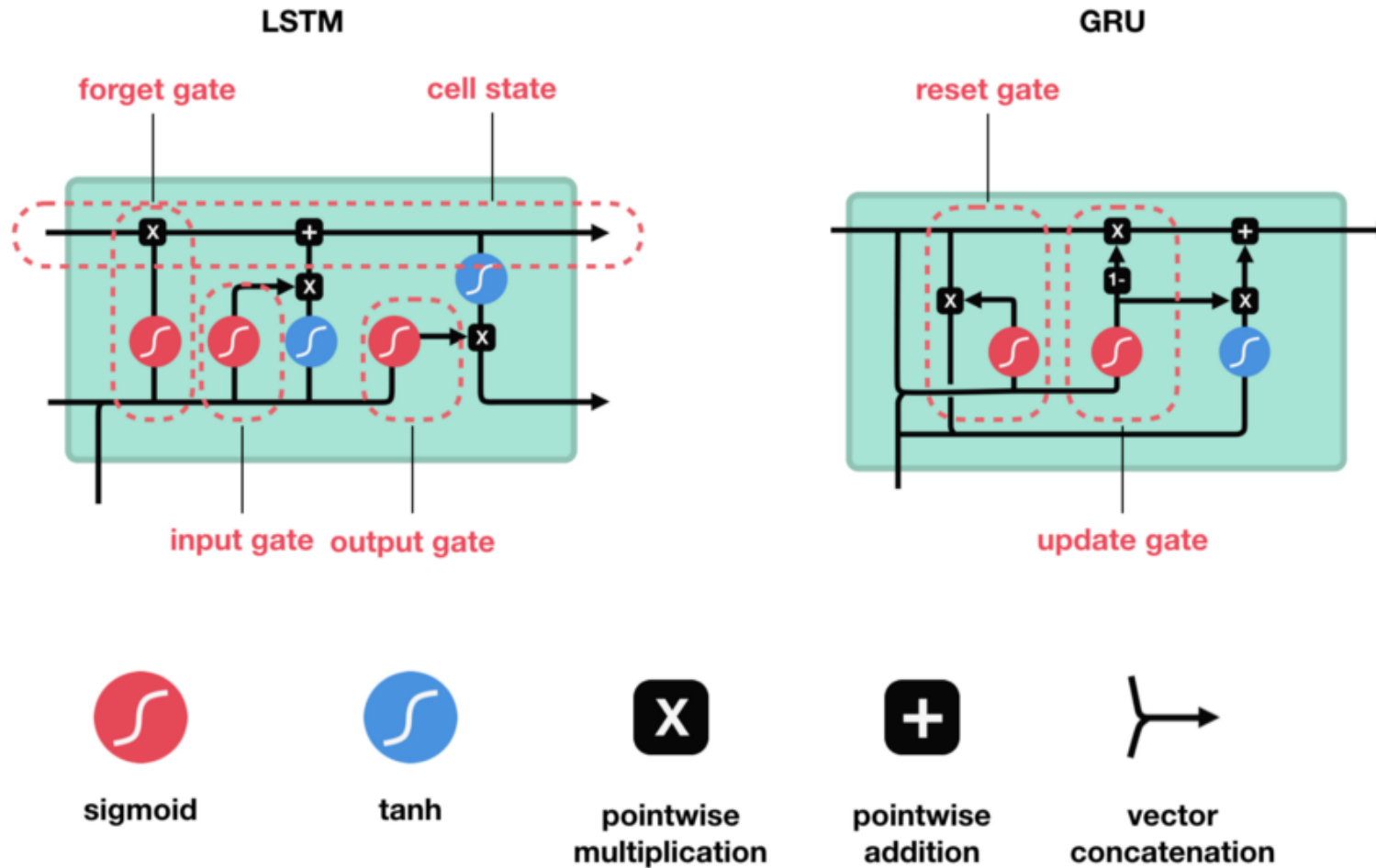


Solution: LSTM (Long-term short tem memory)

- RNN의 단기기억 의존성을 극복하기 위해 도입된 구조
 - 장기기억과 단기기억으로 나누어, 오랫동안 기억해야 할 것과 단기적으로 기억해야할 정보를 구분하여 처리



Alternative Solution: GRU



03. 순서데이터 처리를 위한 딥러닝

차원이 큰 범주형 데이터의 처리
Embedding Layer

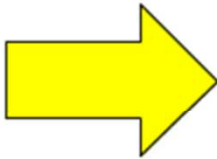
데이터의 종류

- 숫자형 변수 (Numerical Variable)
 - 실수나 정수로 표현할 수 있음
 - 예시) 나이: {56, 24, ...}, 온도: {27.2, 24.0, ...},
 - 값의 차이가 의미 있음
- 범주형 변수 (Categorical Variable)
 - 실수로 표현할 수 없음
 - 분절된 서로 다른 값을 가짐
 - 예시) 성별: {남자, 여자} // 브랜드: {현대, 도요타, 기아, 포드} // 결혼여부: {yes, no}
 - (설령) 서로 다른 값에 숫자를 부여한다 해도 숫자간의 차이는 무의미

범주형 데이터 처리 방법 - One Hot Encoding

- 범주형 변수가 갖는 {차원=고유한 값}개수 만큼 변수를 확장
 - 특정 데이터 객체가 해당 값을 갖고 있을 때 1
 - 아니면 0으로 처리
- 예시
 - Red, Yellow, Green 세개의 색을 갖는 데이터를 one-hot encoding 하면 다음과 같이 처리 가능

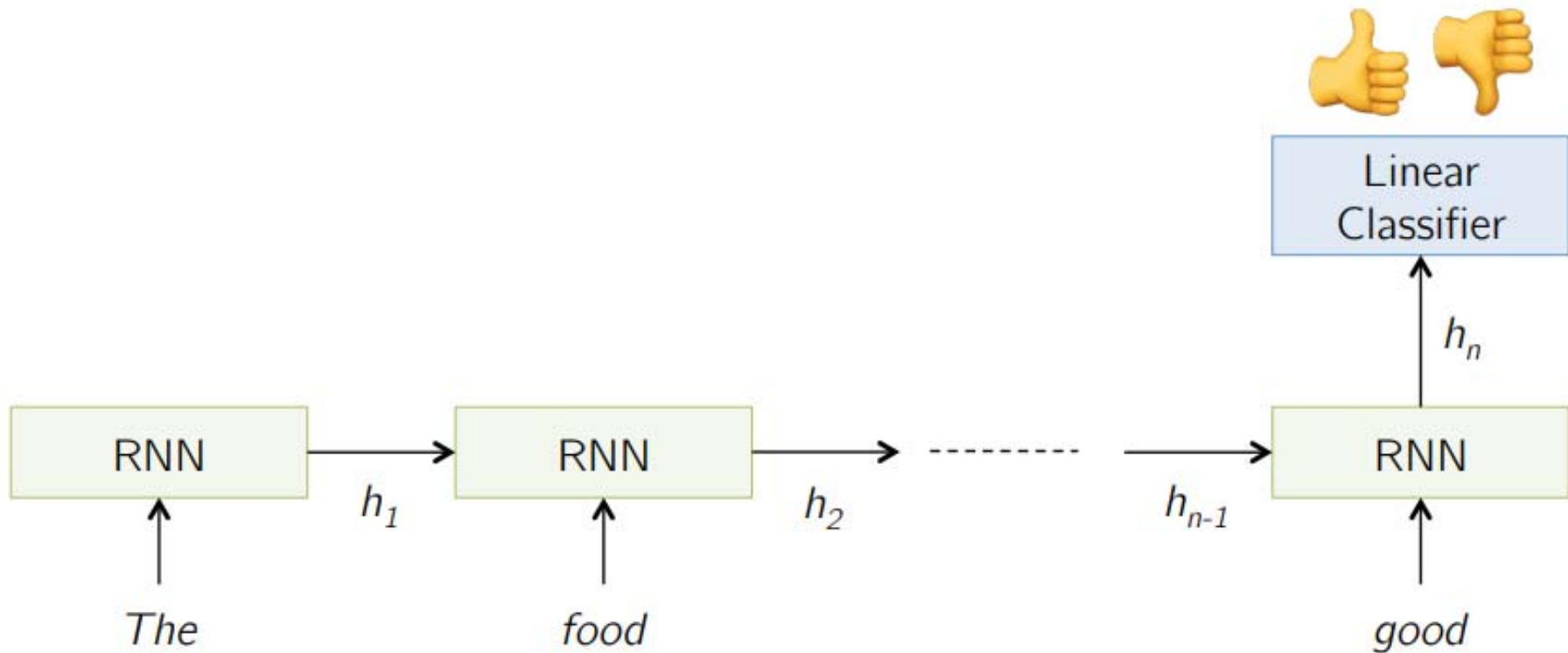
Color		
Red		
Red		
Yellow		
Green		
Yellow		



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

범주형 데이터의 차원이 너무 크면...

- 문장분류
 - 입력값: 숫자로 표현된 각 단어
 - One-hot encoding으로 단어를 표현하고자 한다면...
 - 평균 사용되는 단어 수 (5000~10000) --> 입력 차원의 수가 지나치게 많다



범주형 데이터의 차원이 너무 크면...

Corpus

This tutorial covers the skip gram neural network architecture for Word2Vec. My intention with this tutorial was to skip over the usual introductory and abstract insights about Word2Vec, and get into more of the details. Specifically here I'm diving into the skip gram neural network model.....

statistics

word	id	frequency
network	0	12134
intention	1	12125
tutorial	3	12113
...
other	49999	2000



word vector

$$M_{50000 \times 300} = \begin{bmatrix} \dots & & & \\ & \ddots & & \\ & & \dots & \\ \dots & & & \dots \end{bmatrix}$$

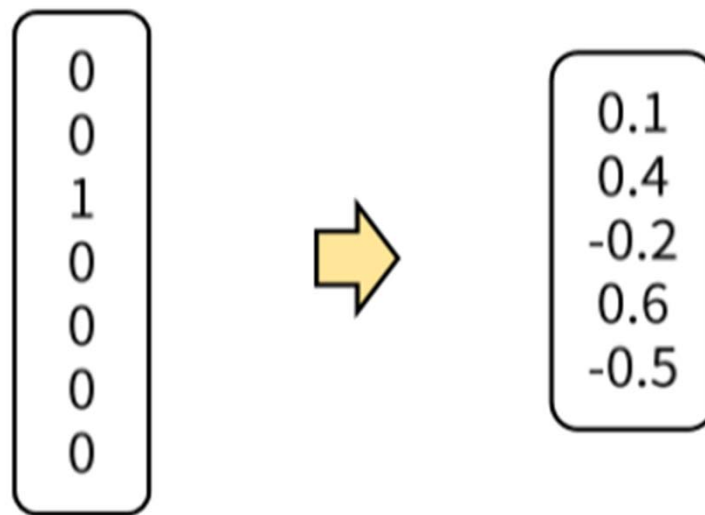
one-hot encode

network	1	0	0	0	0	0	...	0
intention	0	1	0	0	0	0	...	0
tutorial	0	0	1	0	0	0	...	0



Embedding

- One-Hot encoding
 - 대부분의 값이 0을 갖고 단 한 개의 1인 값을 갖는 희소행렬
 - 단어의 경우처럼 차원이 큰 경우 학습해야 하는 패러미터의 수가 지나치게 커짐
- 임베딩 (embedding)
 - 고차원의 One-hot encoding으로 표현된 데이터를 저차원의 밀집된 표현으로 변환시키는 방법



Embedding layer

임베딩 전

One-hot encoding

	cat	mat	on	sat	the
the =>	0	0	0	0	1
cat =>	1	0	0	0	0
sat =>	0	0	0	1	0
...

임베딩 후

A 4-dimensional embedding

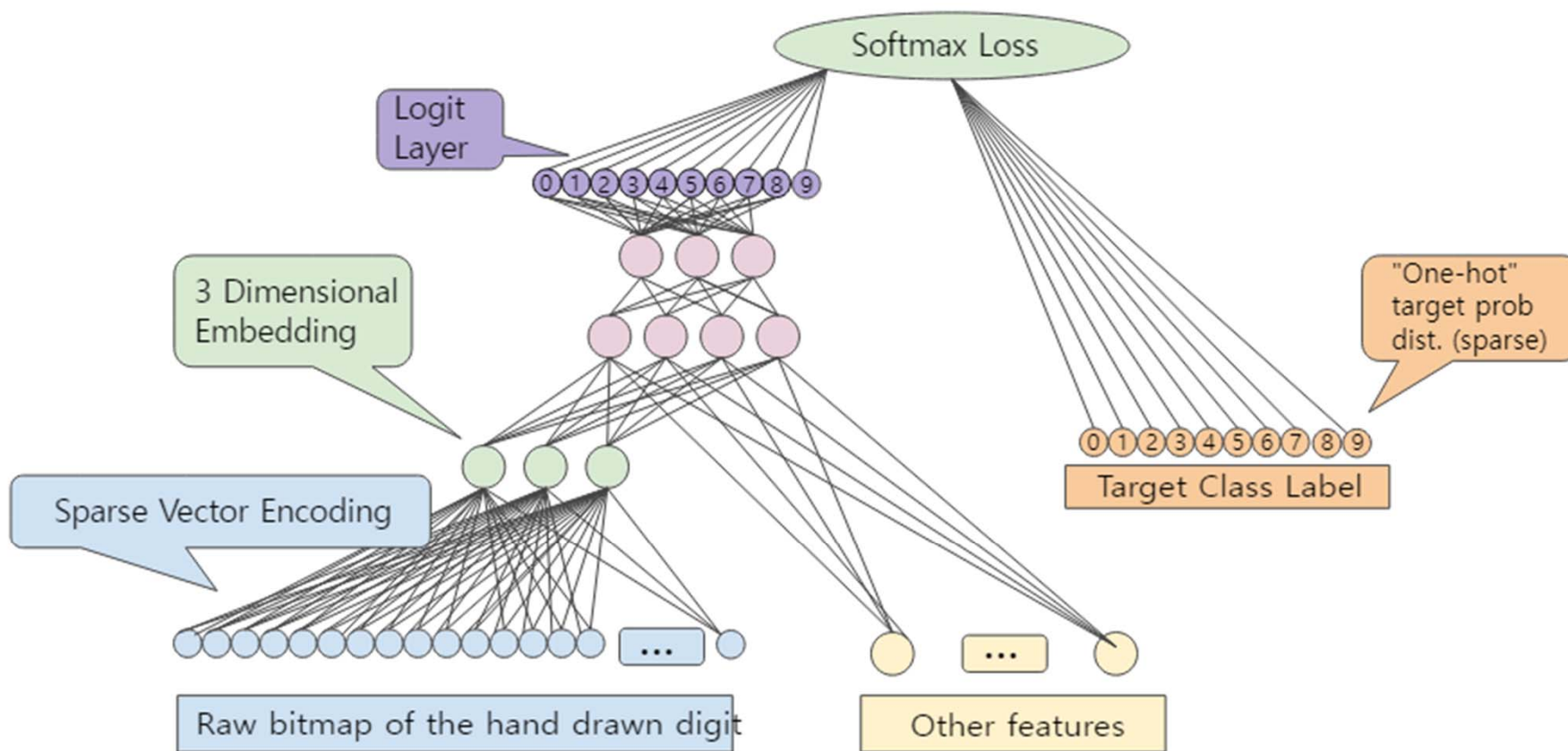
cat =>	1.2	-0.1	4.3	3.2
mat =>	0.4	2.5	-0.9	0.5
on =>	2.1	0.3	0.1	0.4
...

Embedding Layer의 구현

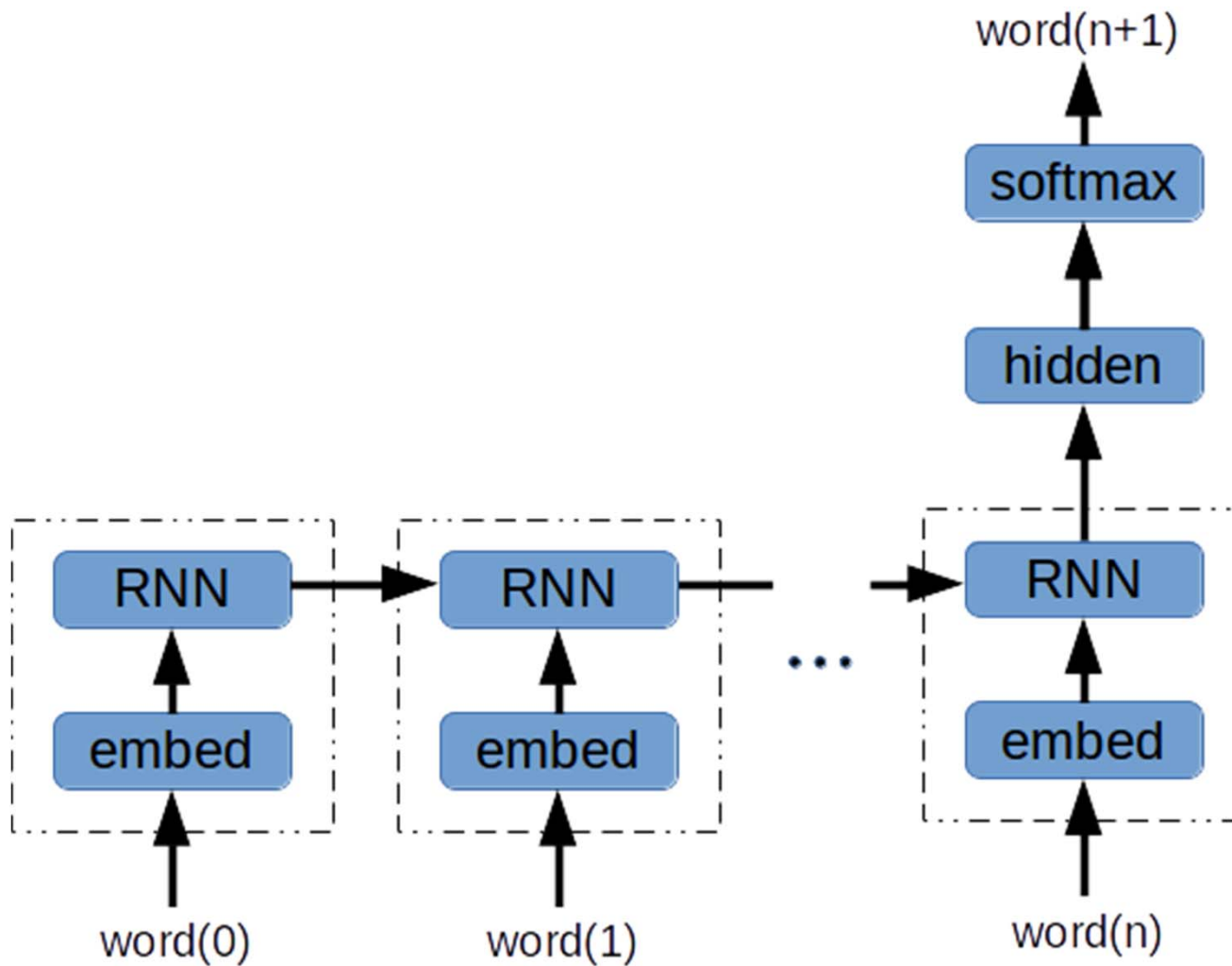
$$\begin{array}{c} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 8 & 2 & 1 & 9 \\ 6 & 5 & 4 & 0 \\ 7 & 1 & 6 & 2 \\ 1 & 3 & 5 & 8 \\ 0 & 4 & 9 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 & 8 \end{bmatrix} \\ \text{One-hot vector} \qquad \qquad \qquad \text{Embedding Weight Matrix} \qquad \qquad \qquad \text{Hidden layer output} \end{array}$$

Embedding layer의 사용

- 입력층에서 범주형 변수를 처리할때 사용
- 다른 수치형 노드와 합쳐서 유연하게 사용 가능
- 학습과정에서 Embedding layer의 가중치도 학습될 것임



Embedding Layer와 RNN의 결합



Embedding Layer와 LSTM의 결합

