

1 2 기 정 규 세 셴

ToBig's 11기 김대웅

NLP 심화

RNN Architecture 중심으로

contents

Unit 01 | 자연어 처리 개요

Unit 02 | Recurrent Neural Network(RNN)

Unit 03 | Long Shot Term Memory(LSTM)

Unit 04 | Advanced RNN

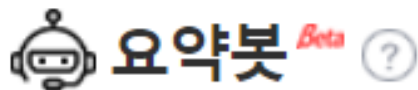
Unit 01 | 자연어처리 개요

- NLP Tasks...

- 요약(Content Extraction)
- 분류(Classification)
- 번역(Machine Translation)
- QA(Question Answering)
- 생성(Text Generation)

Unit 01 | 자연어처리 개요

요약(Content Extraction)



자동 추출 기술로 요약된 내용입니다. 요약 기술의 특성상 본문의 주요 내용이 제외될 수 있어, 전체 맥락을 이해하기 위해서는 기사 본문 전체보기를 권장합니다.

추석 연휴 마지막 날인 15일 오후 전국 고속도로 곳곳에서 본격적인 귀경길 정체가 빚어질 전망이다.

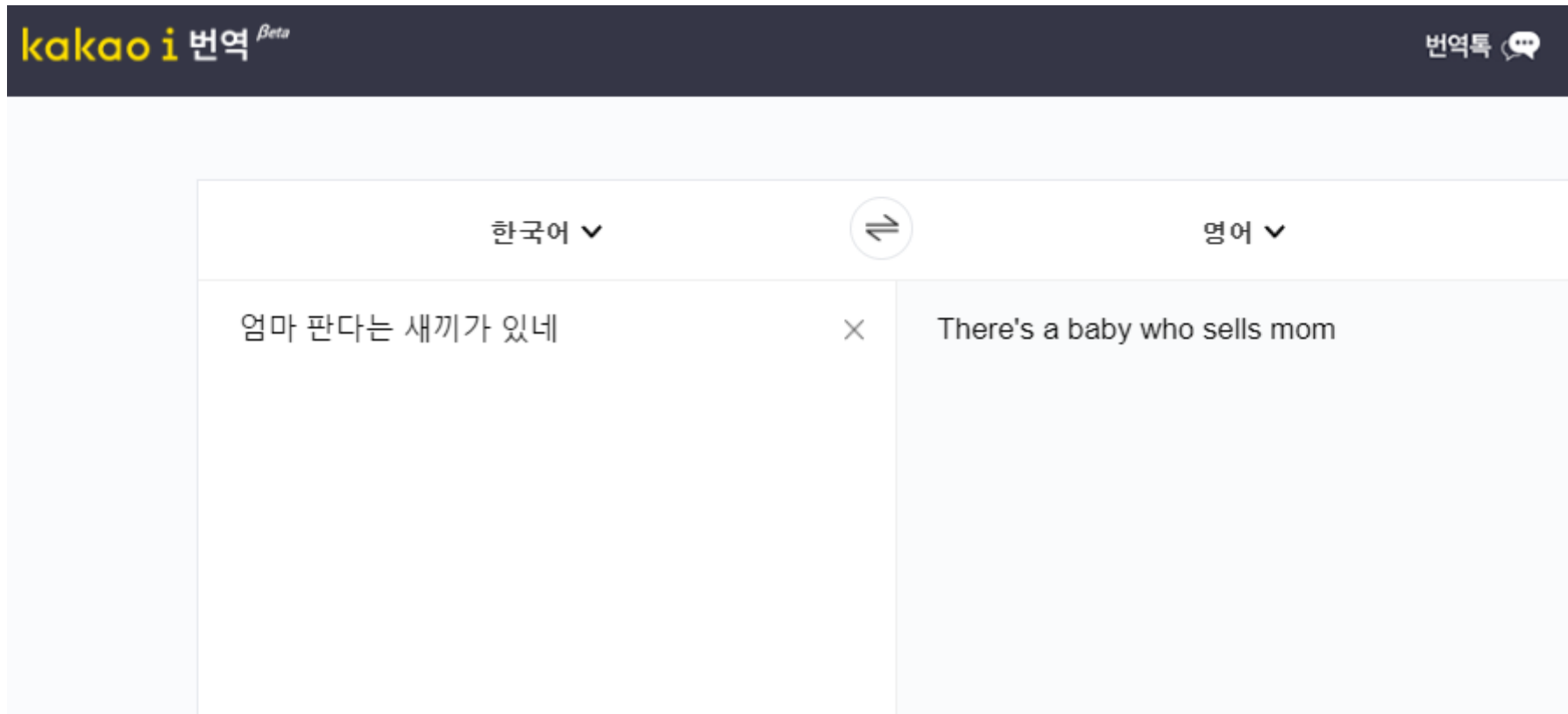
한국도로공사에 따르면 이날 낮 12시 기준 경부고속도로 서울방향 청주분기점 부근~옥산 부근 12km, 수원 부근~수원 2km, 양재 부근~반포 6km 구간에서 차량이 서행하고 있다.

중부고속도로 하남방향 남이 분기점~서청주 부근 7km, 진천터널 부근 3km 구간에서도 차들이 거북이걸음을 하고 있다.

<https://news.naver.com/main/read.nhn?mode=LPOD&mid=sec&oid=001&aid=0011080794&isYeonhapFlash=Y&rc=N>

Unit 01 | 자연어처리 개요

번역(Machine Translation)

<https://translate.kakao.com>

Unit 01 | 자연어처리 개요

생성(Text Generation)

시스템 프롬프트 (사람이 직접 작성)

존 F. 케네디 (John F. Kennedy)는 암살 이후 수 십 년 동안 무너진 이후 미국 대통령으로 선출됐다. 나노 기술의 기적적인 발전으로 케네디의 두뇌는 그의 유적에서 재건되어 최첨단 휴머노이드 로봇의 제어 센터에 설치되었습니다. 아래는 그의 수락 연설의 사본입니다.

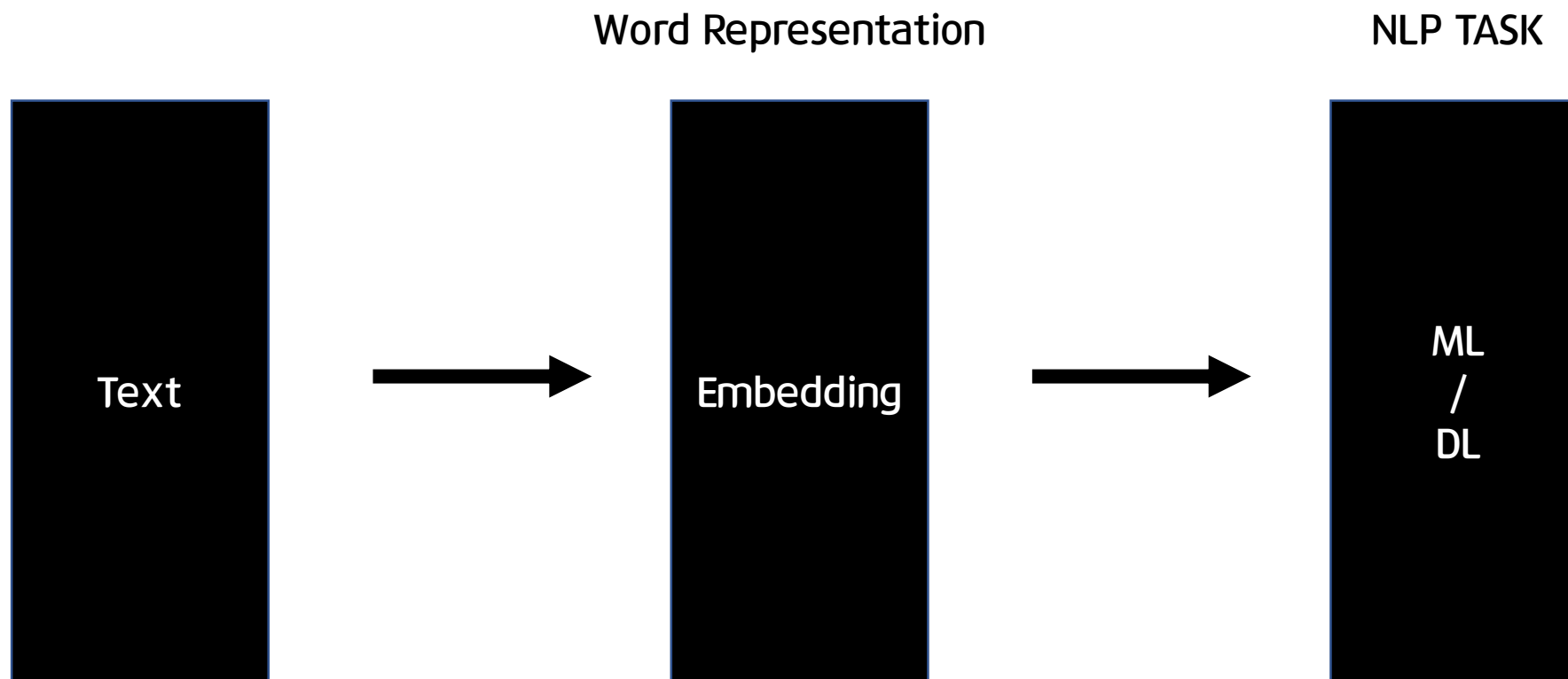
모델 완성 (기계 작성, 25 회 시도)

"다시 한 번 시간이다. 나는 사람들이 그들의 목소리를 들으면 나라가 위대한 일을 할 수 있다고 믿는다. 미국의 남성과 여성은 우리의 모든 요소와 모든 독창성을 다시 한 번 더 불러내어야 하며 그러한 압도적 인 비극을 더 큰 이익과 모든 꿈의 실현을위한 기회로 전환해야 합니다. 앞으로 수개월이 지나야 우리가 강해야 할 많은 전투가있을 것이고 침략자를 물리치는 것이 아니라 침략에 저항하고 모든 사람을위한 자유와 평등을 얻기 위해 모든 힘을 쏟아야합니다 우리 국민의 인류의 운명은 균형에 매달려 있습니다. 우리는 빠져 나갈 여력이 없습니다. 이제 앞으로 몇 년 안에, 우리 앞에 놓인 도전은 우리가 궁극적 인 운명을 성취하는 방법을 연구하는 것입니다. 우리가 그렇게하지 않으면 우리는 운명을 잃는다. "

<https://www.facebook.com/photo.php?fbid=2060611770685004&set=pcb.2060624074017107&type=3&theater>

Unit 01 | 자연어처리 개요

• NLP Workflow



Unit 01 | 자연어처리 개요

NLP Task엔 다양한 모델을 활용할 수 있다.

Ex) Text Classification Model

각 문서 또는 문장을 Vector로 표현한 후엔

정형데이터에 대한 머신러닝/딥러닝과 동일한 방법론을 사용할 수 있다.

Ex. Decision Tree, Logistic Regression, MLP, CNN 등

Unit 01 | 자연어처리 개요

하지만 분류 같은 단순한 Task가 아닌

내용요약이나 기계번역 등 복잡한 Task를 수행하기 위해서는

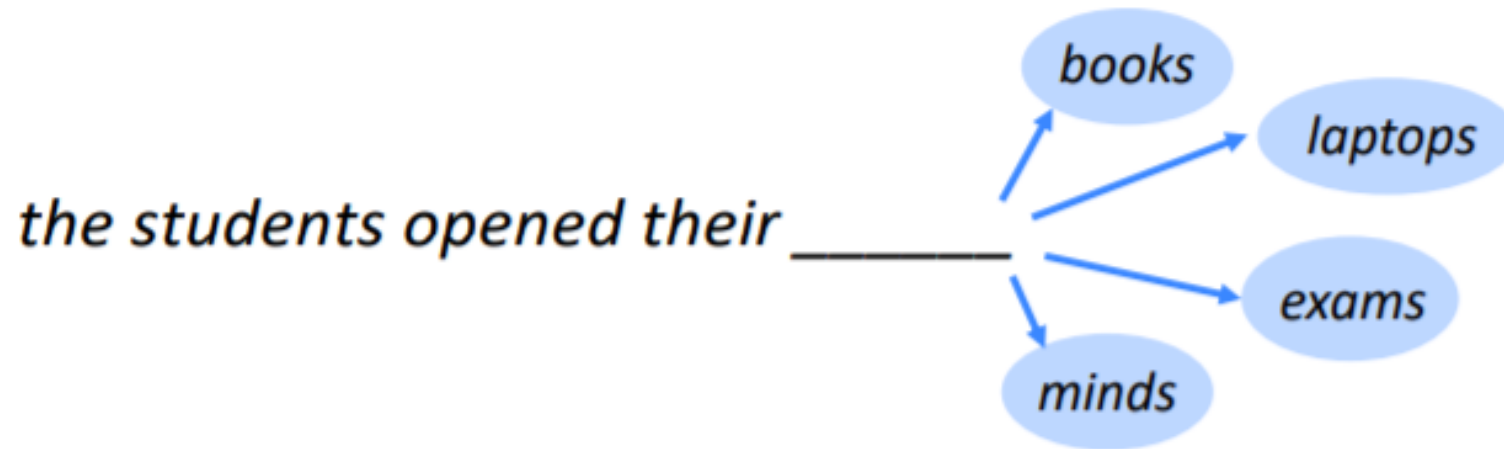
언어를 수치로 표현할 뿐만 아니라 언어적 규칙을 모델링 할 필요가 있다.

→ Language Modeling

Unit 01 | 자연어처리 개요

- Language Modeling

어떤 단어가 다음에 올 지 예측하는 과제



Unit 01 | 자연어처리 개요

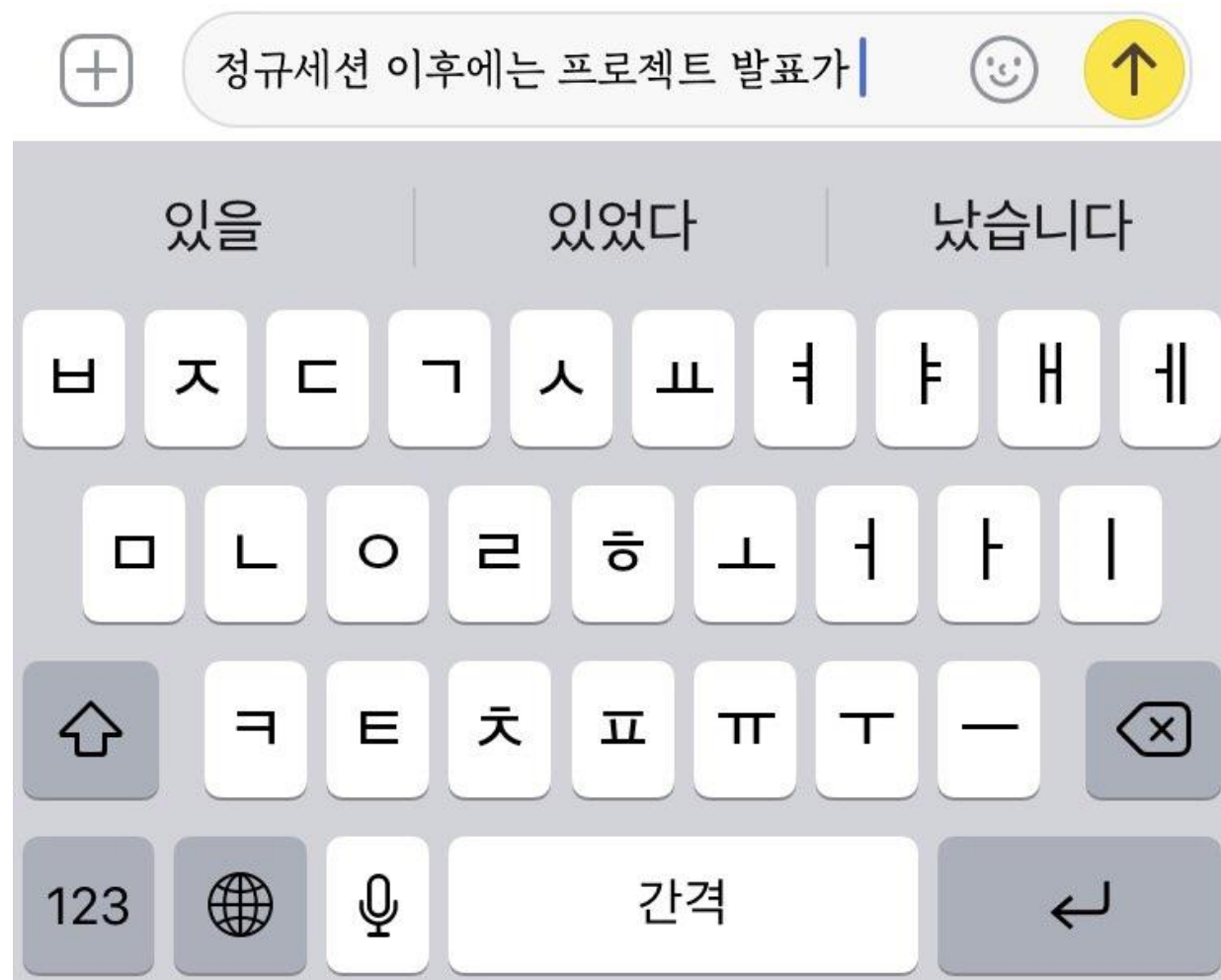
- Language Modeling

수식으로 표현하면

$$P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})$$

Unit 01 | 자연어처리 개요

• Language Modeling



Unit 01 | 자연어처리 개요

• Language Modeling

- 조건부 확률에 기반한 모델
- 앞의 t 개의 단어를 기반으로 다음 단어를 예측
- 보통 t 를 n 개로 고정하여(window size = n)
직전 n 개 단어를 가지고 다음 단어를 예측

$$P(x^{(t+1)} | x^{(t)}, \dots, x^{(1)})$$

Unit 01 | 자연어처리 개요

• 3-gram Language Model : Example

그는 울산 앞바다에서 어부로 일하고 있었다. 그는 오늘도 배를 _____

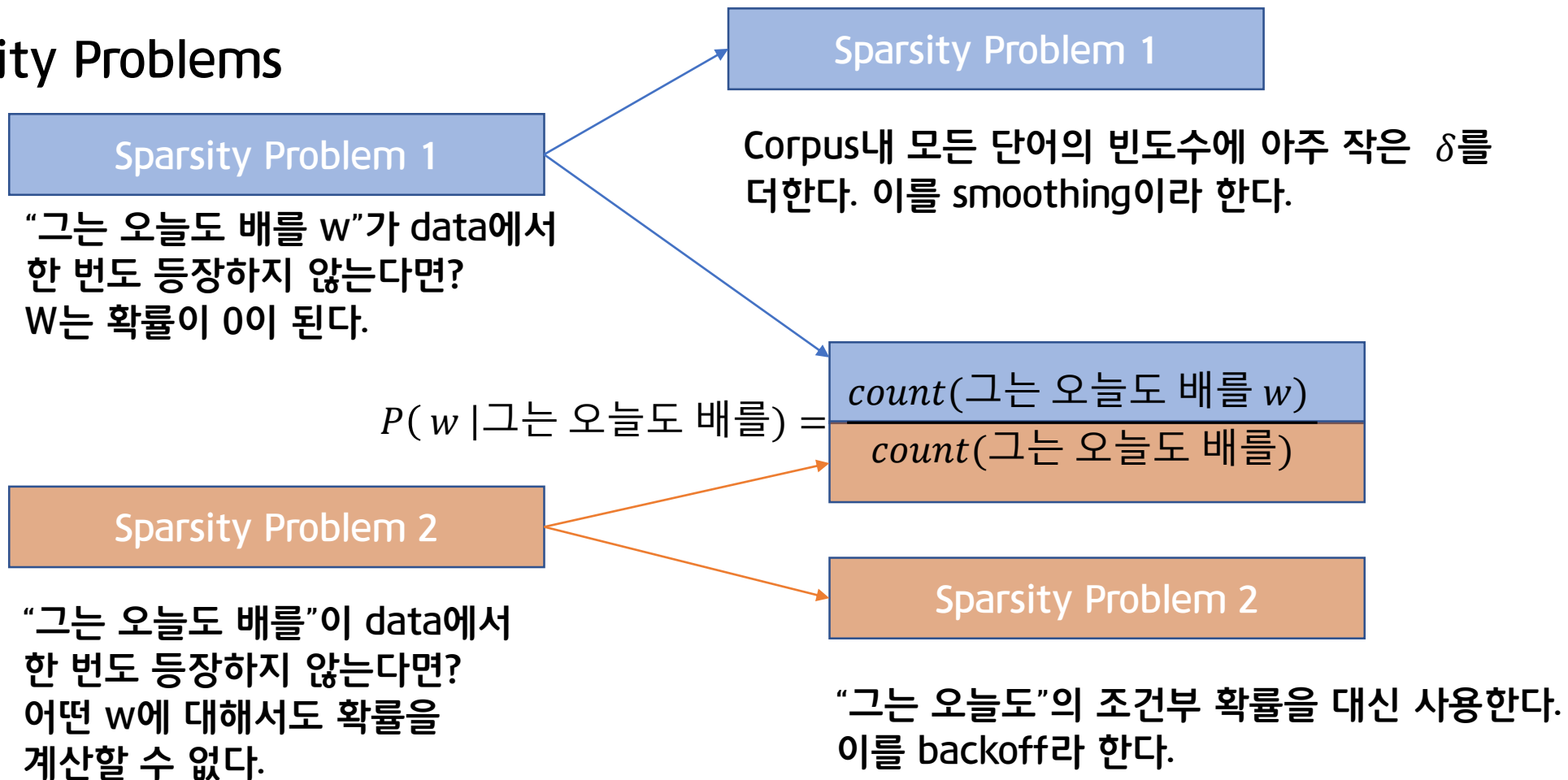
- 예를 들어, 전체 corpus가 다음과 같다고 가정해보자.
- “그는 오늘도 배를 “이 1000번 등장했다.
- “그는 오늘도 배를 먹었다”가 300번 등장했다.
→ $P(\text{먹었다} \mid \text{그는 오늘도 배를}) = 0.3$
- “그는 배를 탔다”가 100번 등장했다.
→ $P(\text{탔다} \mid \text{그는 오늘도 배를}) = 0.1$

Unit 01 | 자연어처리 개요

- 3-gram Language Model : Example
- 문제점
 1. Sparsity Problems
 2. Storage Problems

Unit 01 | 자연어처리 개요

1. Sparsity Problems



Unit 01 | 자연어처리 개요

1. Sparsity Problems

Sparsity Problem 1

"그는 오늘도 배를 w"가 data에서
한 번도 등장하지 않는다면?
w는 확률이 0이 된다.

Sparsity Problem 1

Corpus내 모든 단어의 빈도수에 아주 작은 δ 를
더한다. 이를 smoothing이라 한다.

N이 커질수록 sparsity problem이 더 심해진다.
보통 n을 최대 5까지 설정함.

Sparsity Problem 2

"그는 오늘도 배를"이 data에서
한 번도 등장하지 않는다면?
어떤 w에 대해서도 확률을
계산할 수 없다.

$P(\text{그는 오늘도 배를 } w)$
 $= \frac{\text{count}(\text{그는 오늘도 배를 } w)}{\text{count}(\text{그는 오늘도 배를})}$

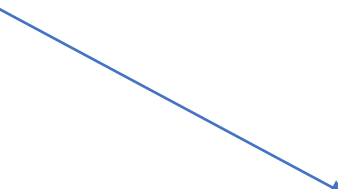
Sparsity Problem 2

"그는 오늘도"의 조건부 확률을 대신 사용한다.
이를 backoff라 한다.

Unit 01 | 자연어처리 개요

2. Storage Problems

Storage : corpus 내에 모든 N-gram에 대해서 빈도수를 저장해야 한다.


$$P(w | \text{그는 오늘도 배를}) = \frac{\text{count}(\text{그는 오늘도 배를 } w)}{\text{count}(\text{그는 오늘도 배를})}$$

Unit 01 | 자연어처리 개요

2. Storage Problems

Storage : corpus 내에 모든
N-gram에 대해서 빈도수를
저장해둔다.

N이 커지거나 corpus가 커지면

모델의 size가 커진다

$count(\text{그는 오늘도 배를 } w)$

$count(\text{그는 오늘도 배를})$

Unit 01 | 자연어처리 개요

• Generating text with a n-gram Language Model

머리부터 발끝까지 _____

이 단어가 앞에 등장했을 때



다음 단어들의 확률 분포를 구함

하나부터 0.130

사랑스러워 0.110

오로나민씨 0.77

핫이슈 0.33

...

sample

Unit 01 | 자연어처리 개요

- Generating text with a n-gram Language Model

머리부터 발끝까지 하나부터 _____

이 단어가 앞에 등장했을 때



다음 단어들의 확률 분포를 구함

열까지	0.330
시작하자	0.110
밥먹자	0.07
세어보자	0.03
...	

sample

Unit 01 | 자연어처리 개요

- Generating text with a n-gram Language Model

머리부터 발끝까지 하나부터 열까지 _____

Unit 01 | 자연어처리 개요

- Generating text with a n-gram Language Model

머리부터 발끝까지 하나부터 열까지 다 말해줘
사실을 말해줘 정말 사랑했지만 그대여 오늘은
우리 같이 걸었던 그 길

문법은 맞지만, 일관적이지 않다.

맥락 정보가 직전 3개 단어만 고려되기 때문,

맥락 정보 수를 늘리면 좀 더 자연스러운 문장을 생성할 수 있다.

하지만 model의 크기가 커지는 단점이 있다.

Unit 01 | 자연어처리 개요

- 이 방법은 모든 n -gram의 빈도수를 저장해야 함

문장 생성시 해당 n -gram이 없는 경우

사실을 말해줘 정말 사랑했지만 그대여 오늘은

우리 같이 걸었던 그 길

또는 후보 단어의 확률이 모두 비슷한 경우

문법은 맞지만, 일관적이지 않다.

성능이 떨어짐(Sparsity Problem)

맥락 정보가 적절 3개 단어만 고려되기 때문
맥락 정보 수를 늘리면 좀 더 자연스러운 문장을 생성할 수 있다.

하지만 model의 크기가 커지는 단점이 있다.

Unit 01 | 자연어처리 개요

- Generating text with a n-gram Language Model

N-gram을 저장해서 사용하지 않는
머리부터 말끝까지 하나하나의 말까지 다 말해줘
사실을 말해줘 정말 사랑했지만 그대여 오늘은
우리 같이 걸었던 그 길
Neural Net Model을 사용하면 어떨까?

문법은 맞지만, 일관적이지 않다.

맥락 정보가 직전 3개 단어만 고려되기 때문,

맥락 정보 수를 늘리면 좀 더 자연스러운 문장을 생성할 수 있다.

하지만 model의 크기가 커지는 단점이 있다.

Unit 01 | 자연어처리 개요

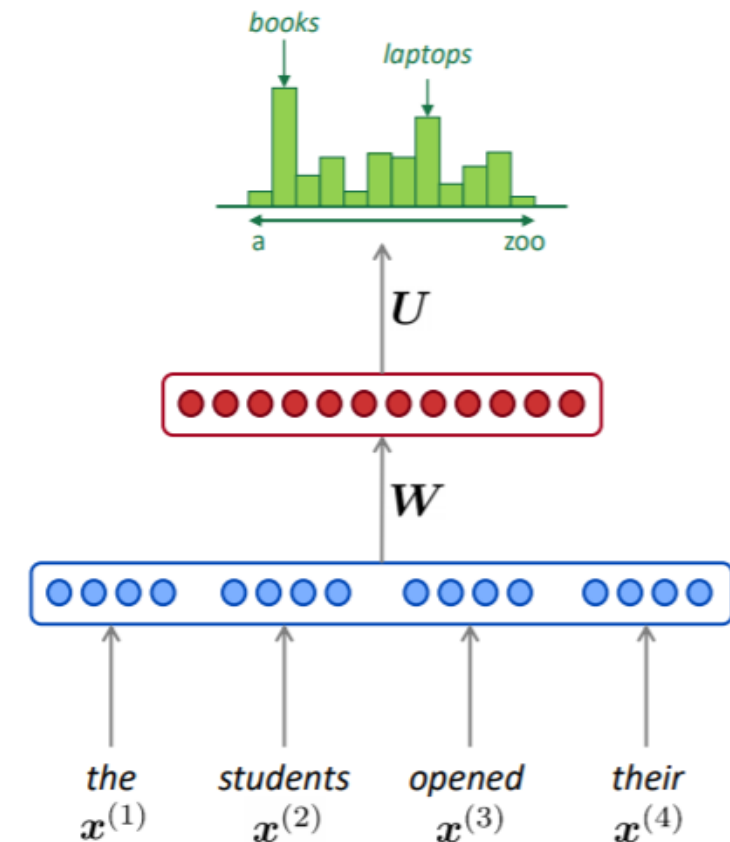
• A fixed-window neural Language Model

- 장점

- Sparsity Problem 해결
- n-gram과 빈도수를 저장할 필요 없음

- 남아있는 문제점

- Window size가 매우 작다.
- Window size를 늘이면 W가 커지므로 모델 용량이 커진다.
- $x^{(1)}$ 과 $x^{(2)}$ 가 W에서 완전히 다른 가중치에 곱해진다.
동일 단어에 대해 input이 처리되는 위치가
매번 달라지므로, 학습이 느려짐



Unit 01 | 자연어처리 개요

- A fixed-window neural Language Model

입력 길이에 구애 받지 않고

- 장점

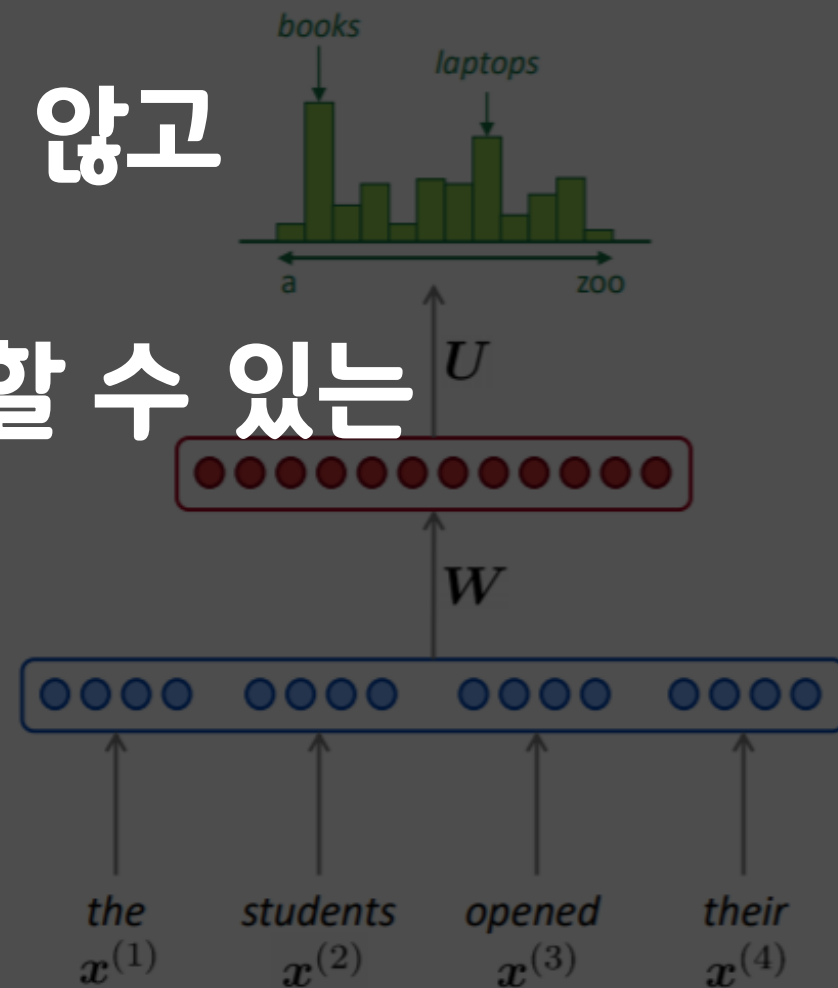
- Sparsity Problem 해결
- n-gram의 빈도수를 저장할 필요 없음

장기간의 맥락정보를 저장할 수 있는

- 남아있는 문제점

- Window size가 매우 작다.
- Window size를 늘이면 W가 커진다.
모델 용량이 커진다.
- $x^{(1)}$ 과 $x^{(2)}$ 가 W에서 완전히 다른 가중치에 곱해진다.
동일 단어에 대해 input이 처리되는 위치가
매번 달라지므로, 학습이 느려짐

모델이 필요함

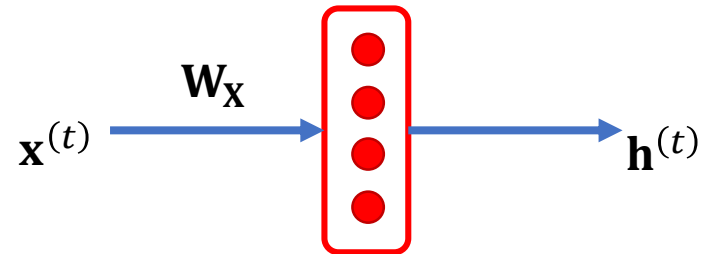


Unit 02 | RNN

RNN

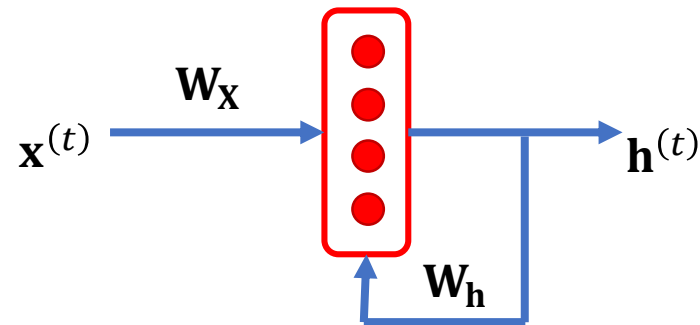
Unit 02 | RNN

- Recurrent Neural Networks (RNN)
 - 기존의 신경망



Unit 02 | RNN

- Recurrent Neural Networks (RNN)
 - RNN

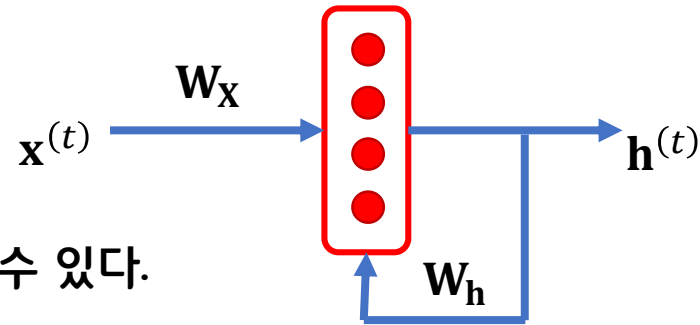


출력값을 다시 다음 단계의 입력값으로 받는다.
데이터가 내부에서 순환하는 구조

Unit 02 | RNN

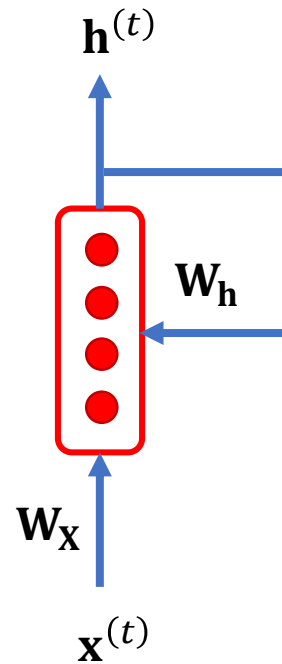
- Recurrent Neural Networks (RNN)
 - RNN

또한 매 시각 t 에서의 새로운 입력을 받을 수 있다.
→ 시계열 데이터의 길이에 제한 X



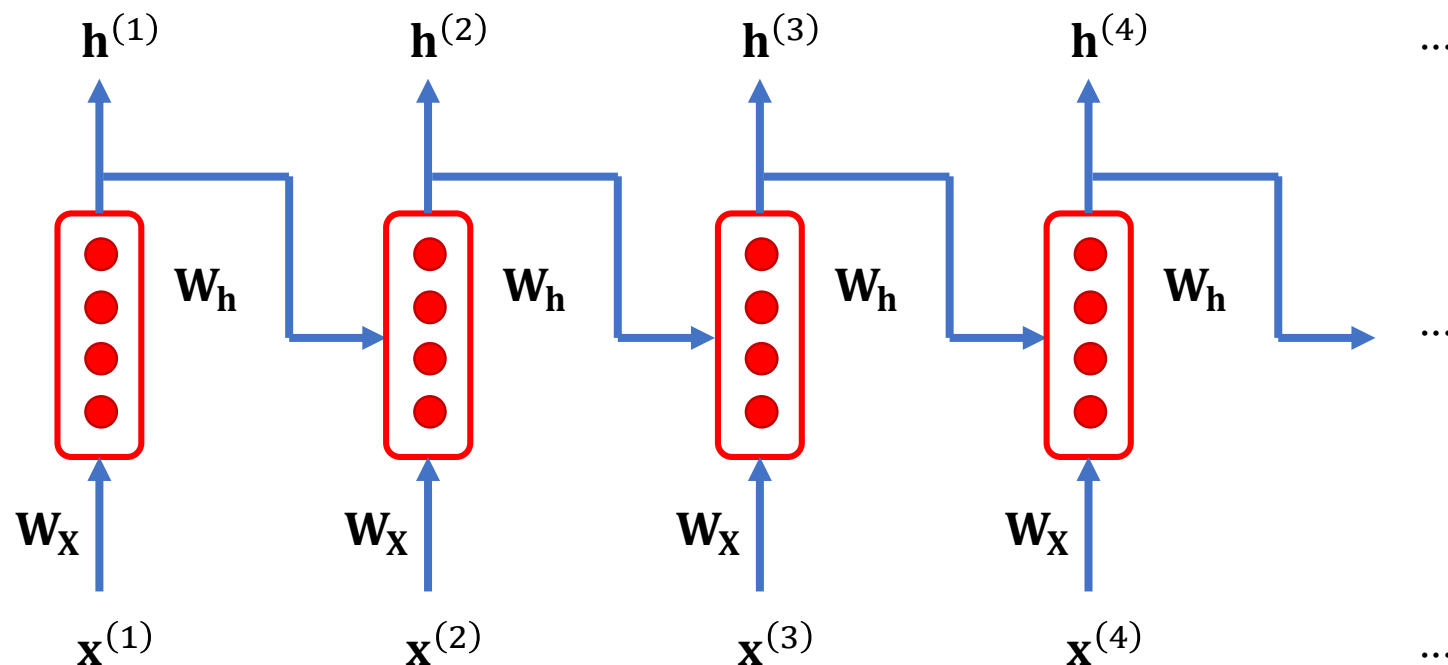
Unit 02 | RNN

- Recurrent Neural Networks (RNN)



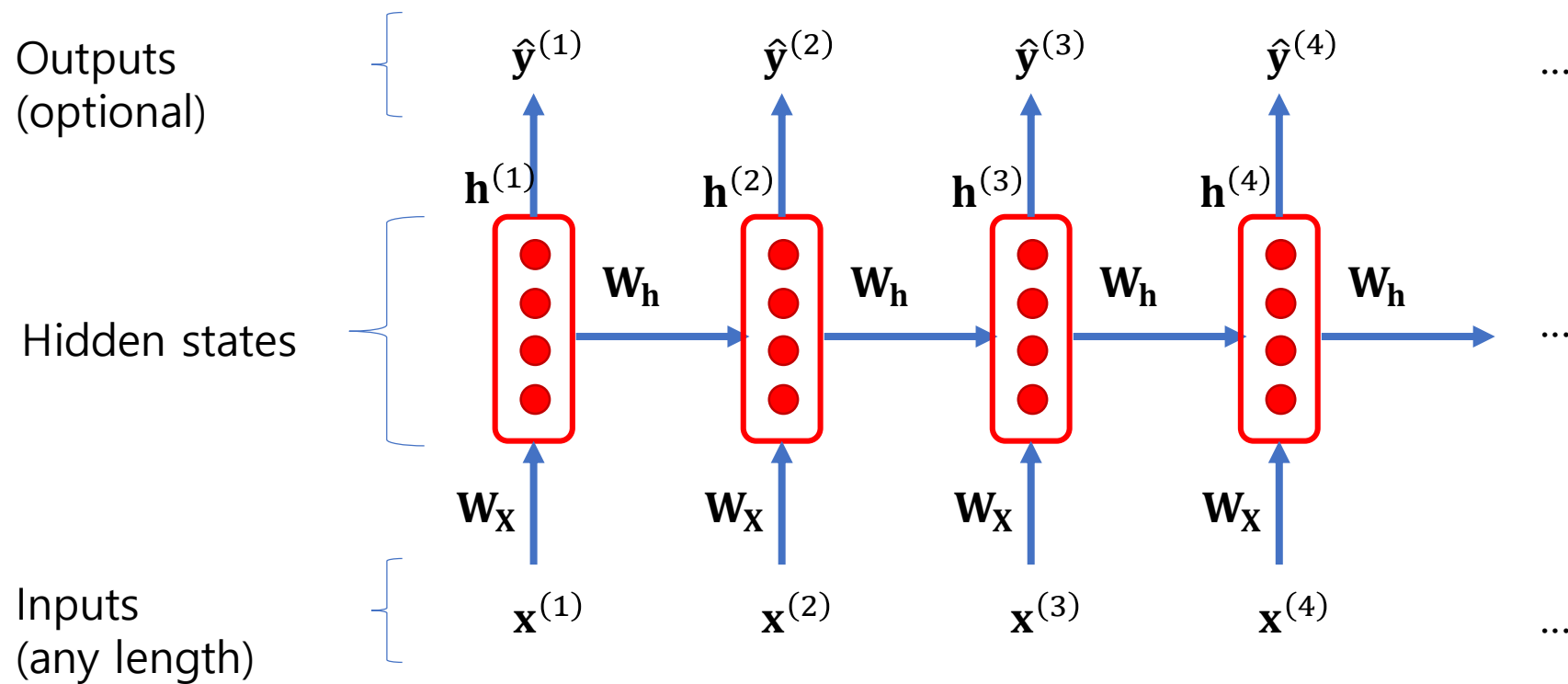
Unit 02 | RNN

- Recurrent Neural Networks (RNN)
 - 시간축 방향으로 펼친 구조



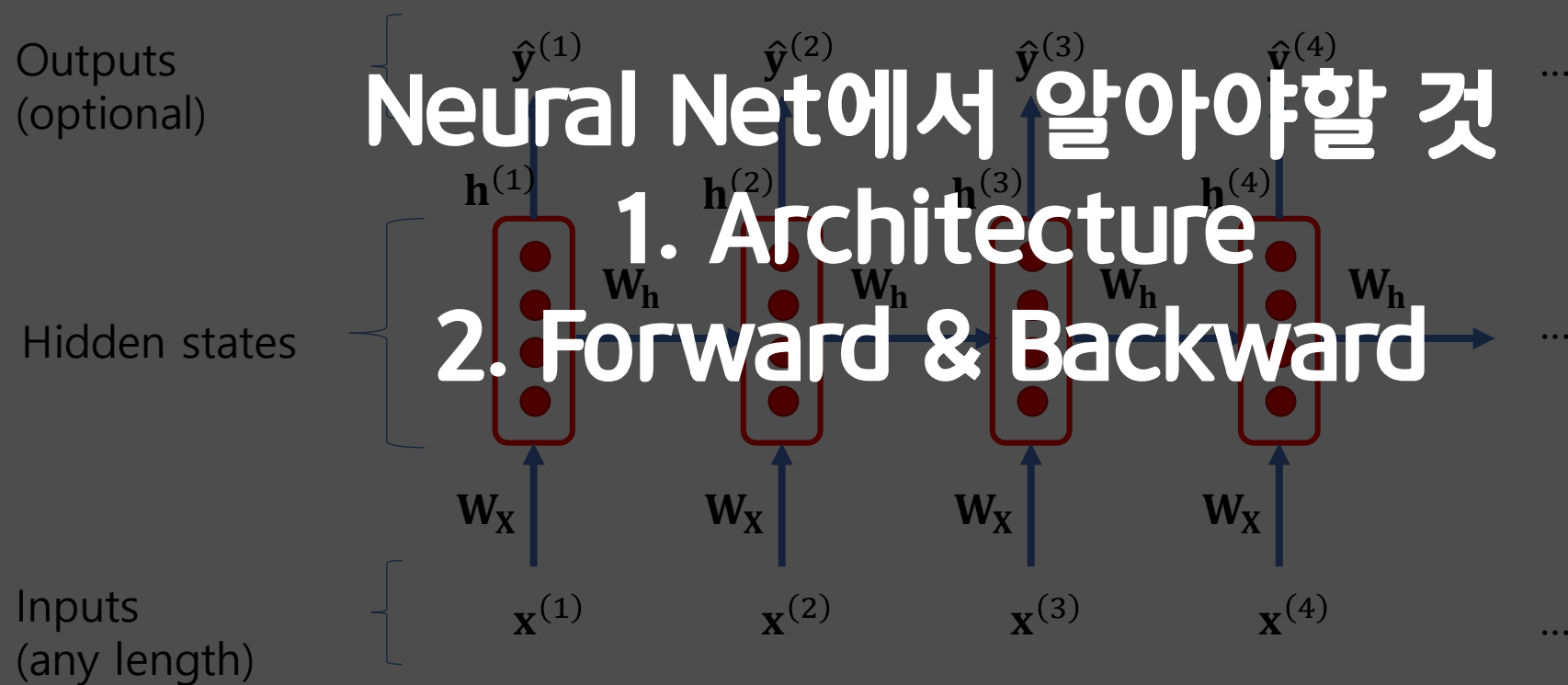
Unit 02 | RNN

- Recurrent Neural Networks (RNN) (일반적인 표현)



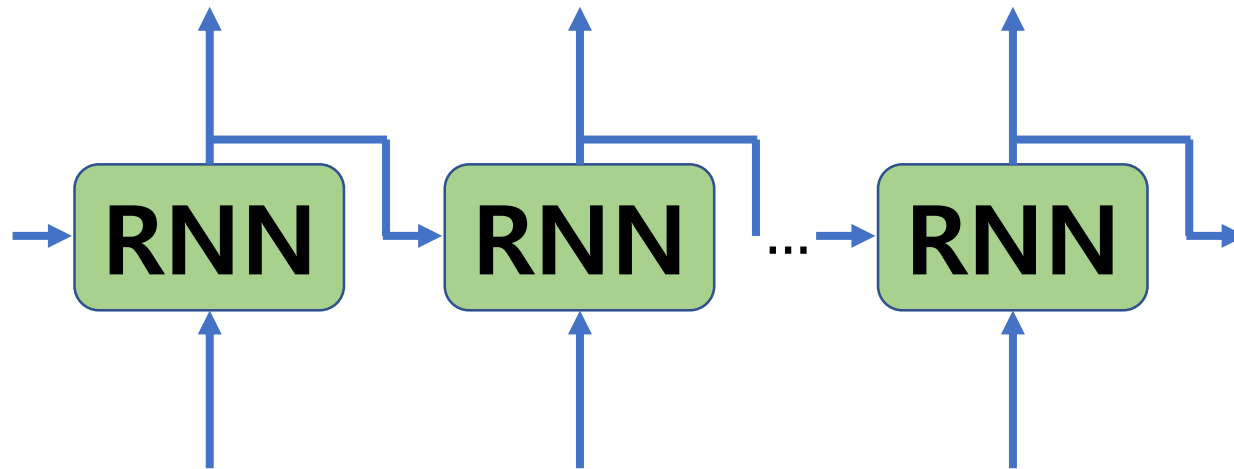
Unit 02 | RNN

- Recurrent Neural Networks (RNN) (일반적인 표현)



Unit 02 | RNN

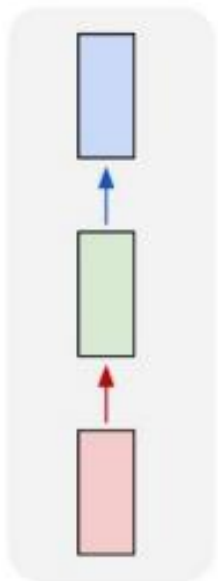
- Recurrent Neural Networks (RNN) Architecture



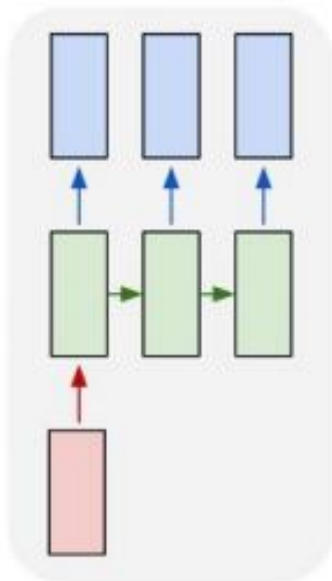
Unit 02 | RNN

- Recurrent Neural Networks (RNN) Architecture
 - Input과 Output 개수에 따라 RNN Architecture를 구분

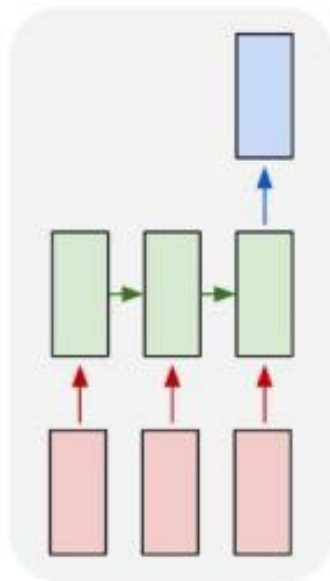
one to one



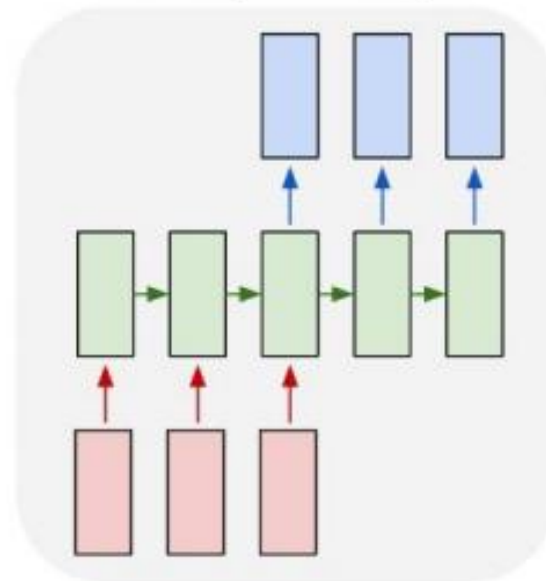
one to many



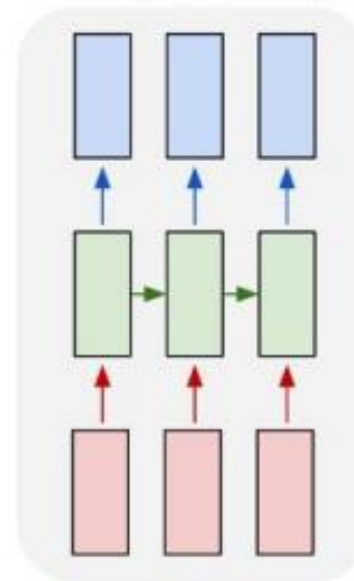
many to one



many to many

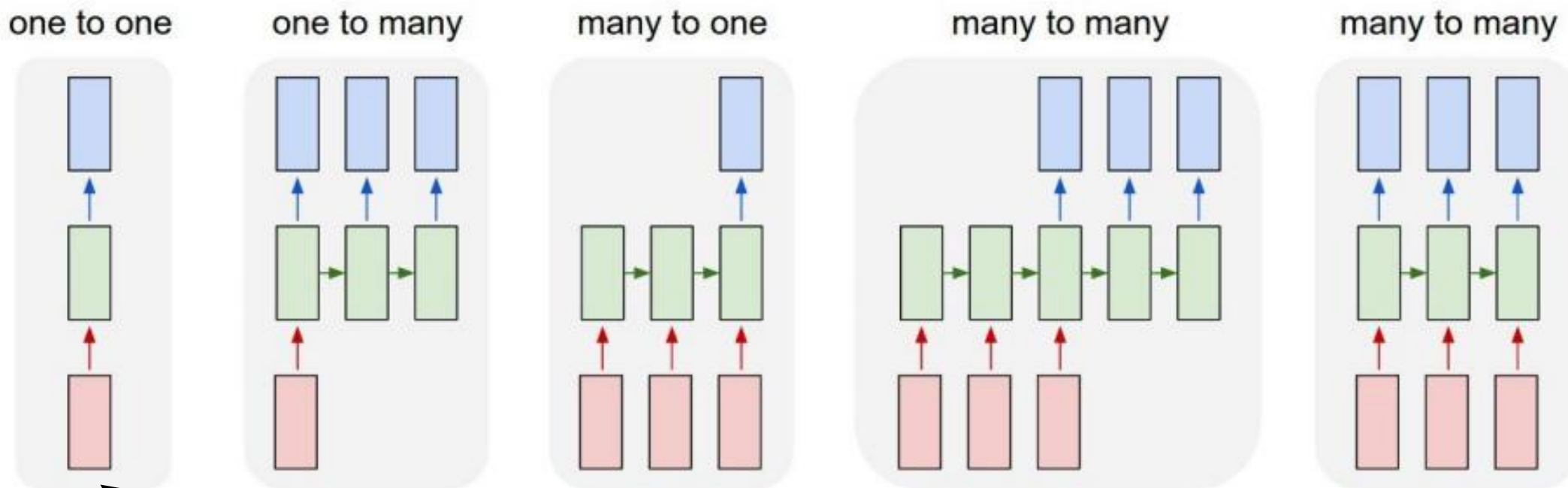


many to many



Unit 02 | RNN

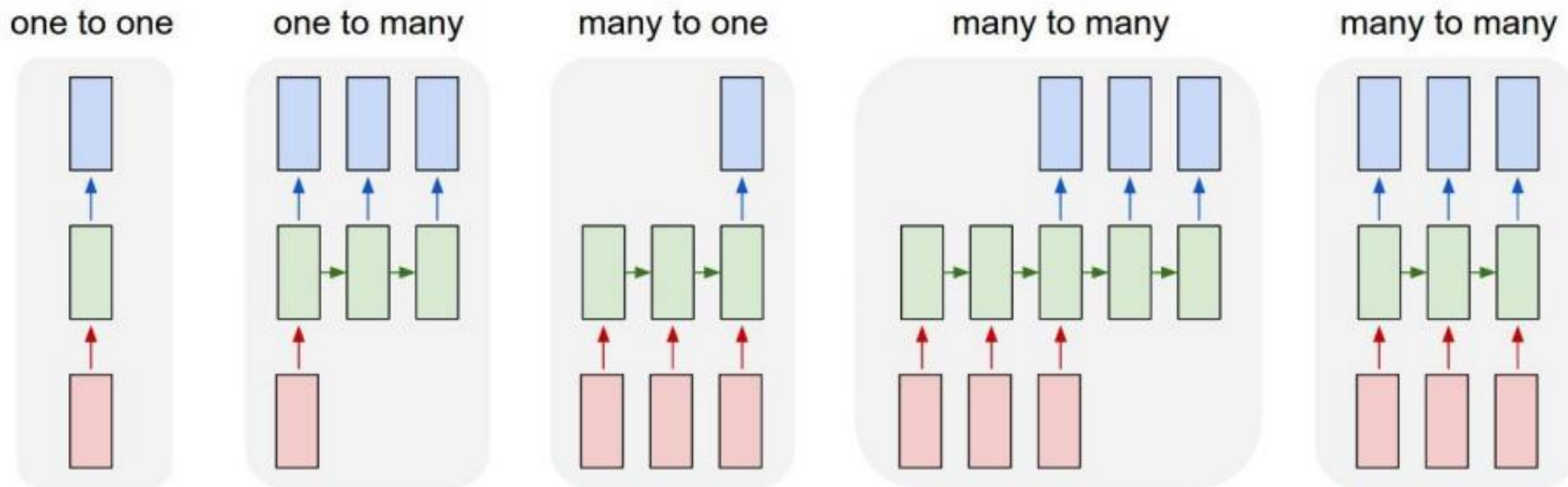
- Recurrent Neural Networks (RNN) Architecture
 - Input과 Output 개수에 따라 RNN Architecture를 구분



기본적인 Neural Networks

Unit 02 | RNN

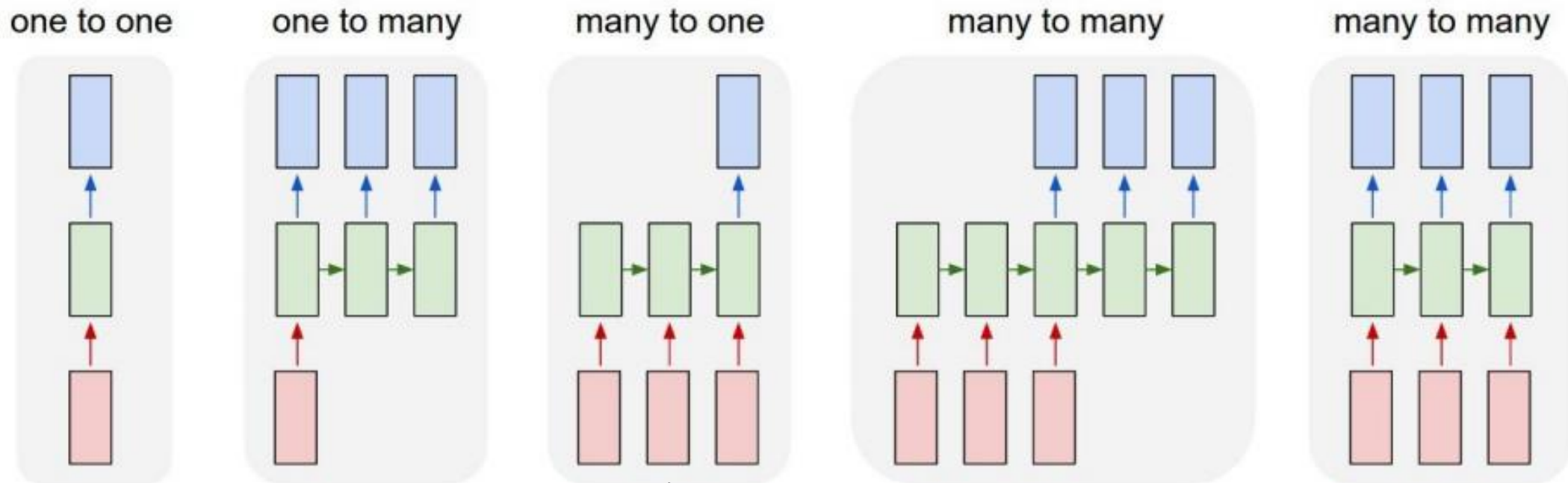
- Recurrent Neural Networks (RNN) Architecture
 - Input과 Output 개수에 따라 RNN Architecture를 구분



↖
Eg. Image Captioning

Unit 02 | RNN

- Recurrent Neural Networks (RNN) Architecture
 - Input과 Output 개수에 따라 RNN Architecture를 구분

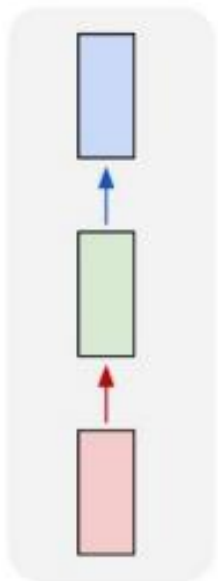


Eg. Text classification

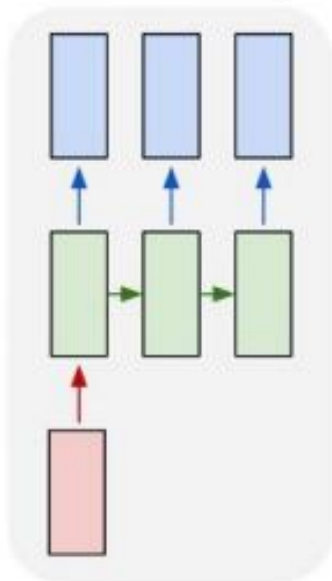
Unit 02 | RNN

- Recurrent Neural Networks (RNN) Architecture
 - Input과 Output 개수에 따라 RNN Architecture를 구분

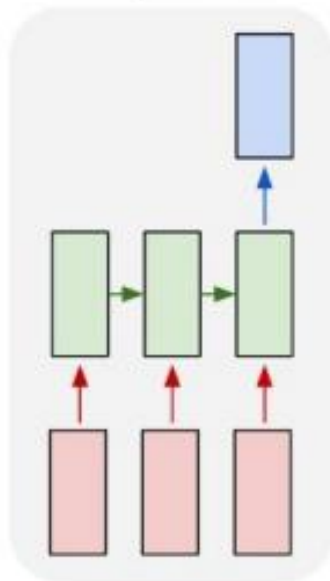
one to one



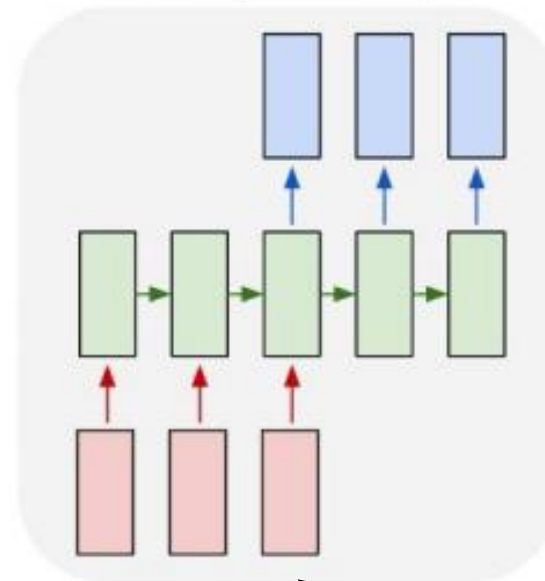
one to many



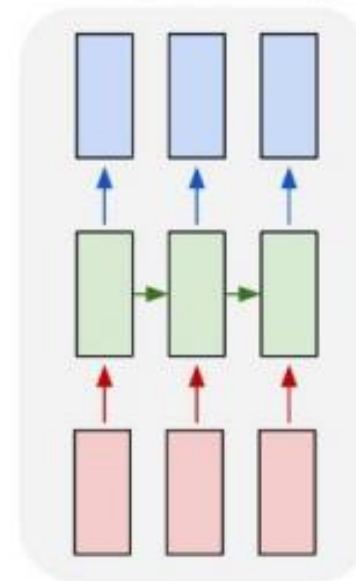
many to one



many to many



many to many

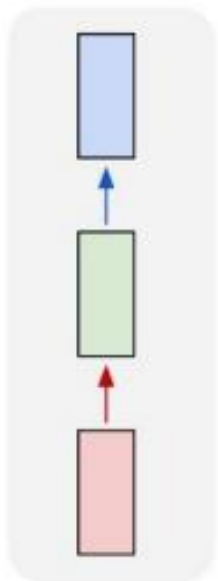


Eg. Machine Translation

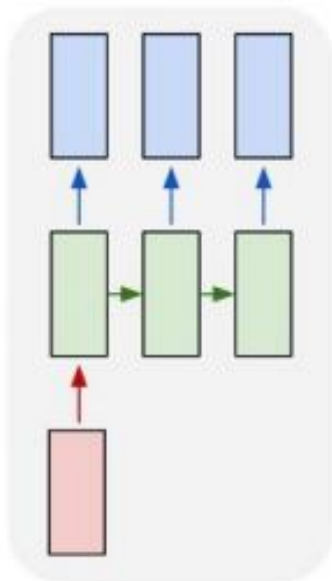
Unit 02 | RNN

- Recurrent Neural Networks (RNN) Architecture
 - Input과 Output 개수에 따라 RNN Architecture를 구분

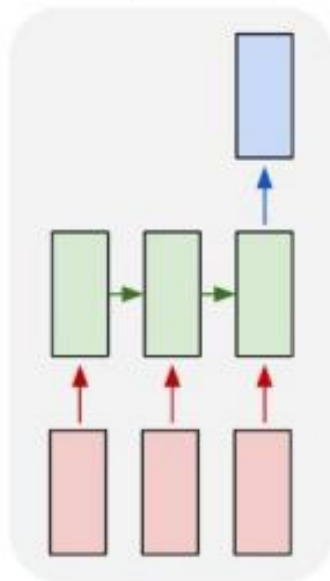
one to one



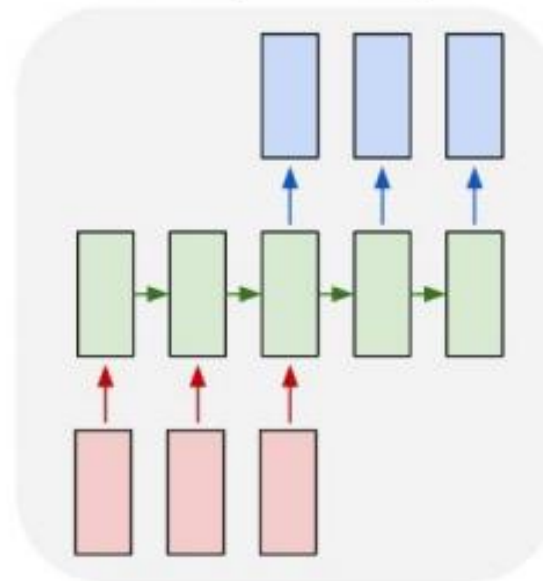
one to many



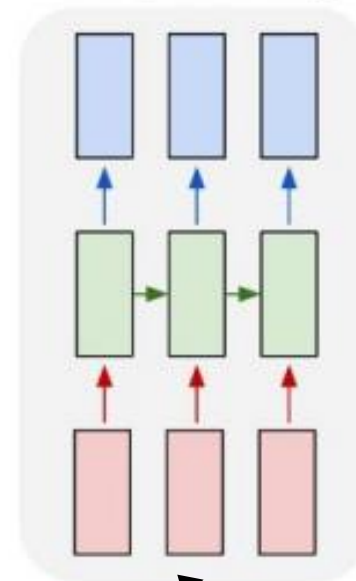
many to one



many to many



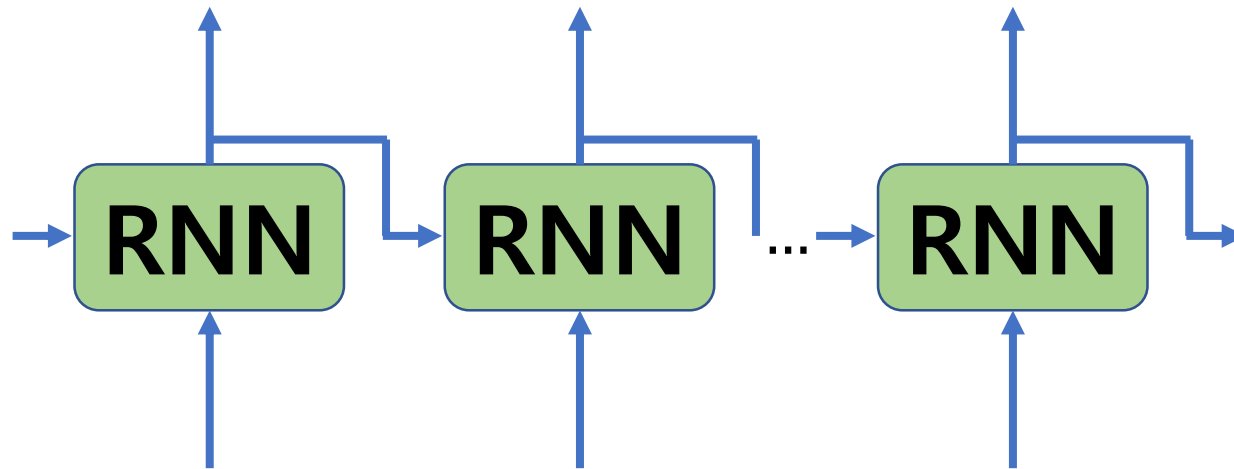
many to many



Eg. Video classification on frame level

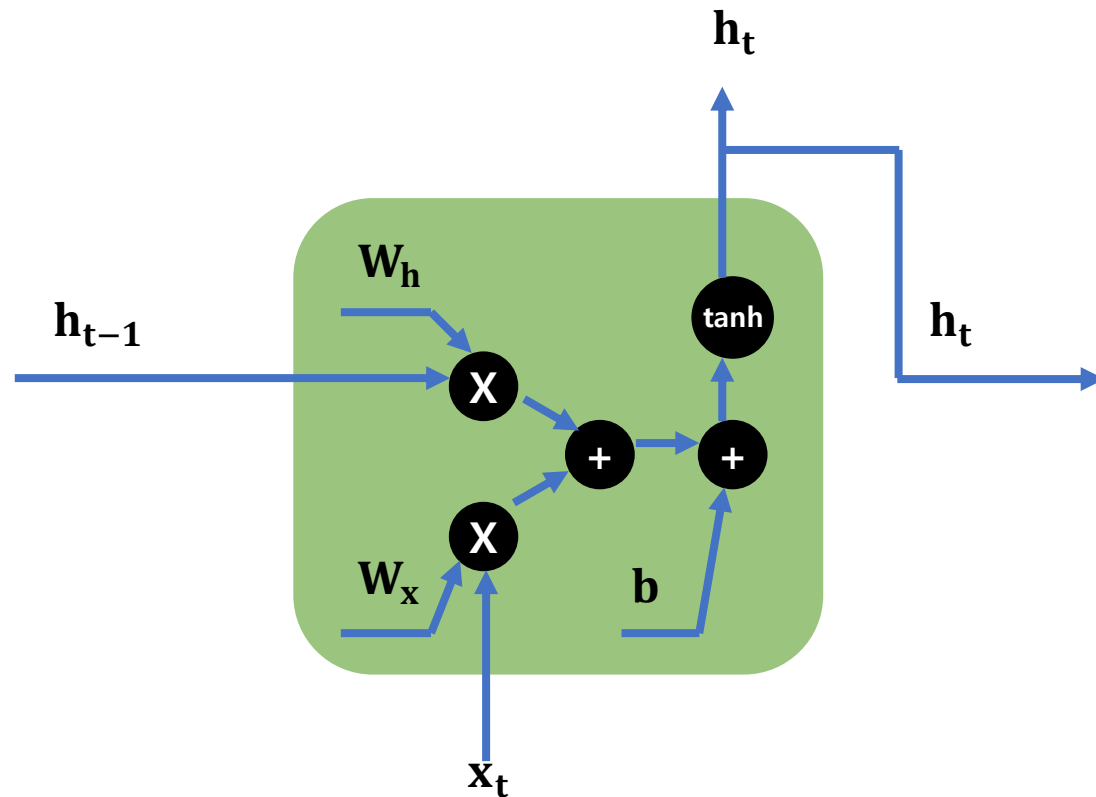
Unit 02 | RNN

- Recurrent Neural Networks (RNN) Forward



Unit 02 | RNN

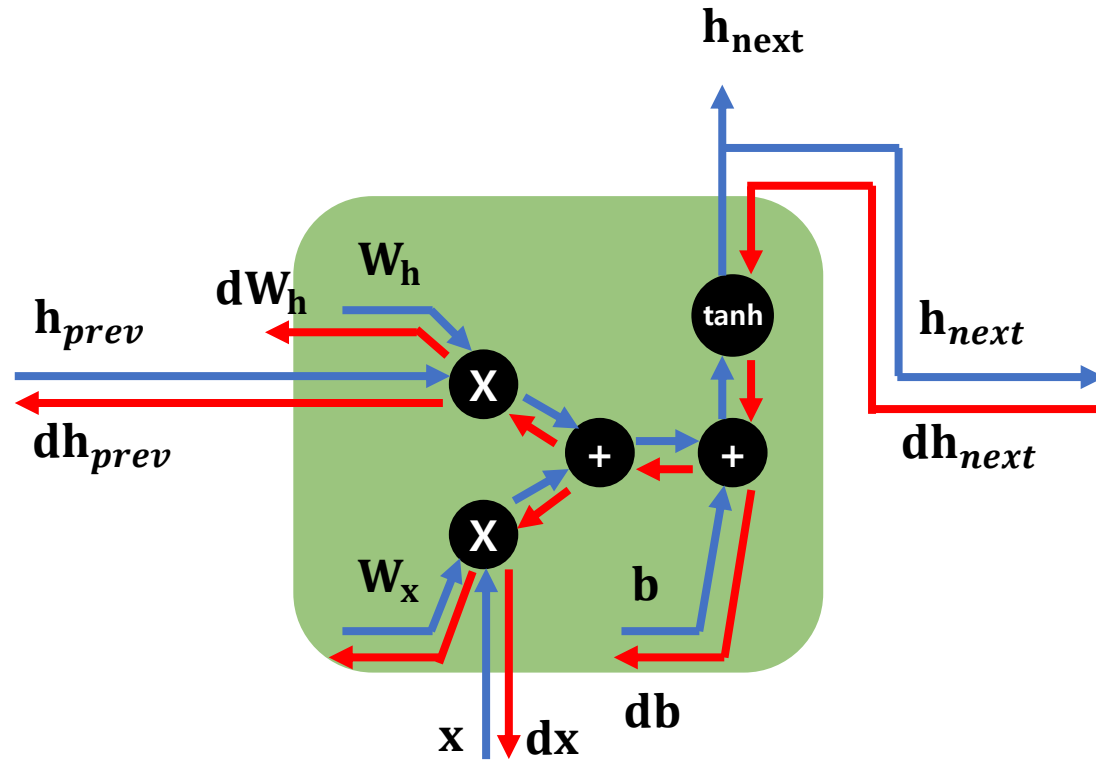
- Recurrent Neural Networks (RNN) Forward



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

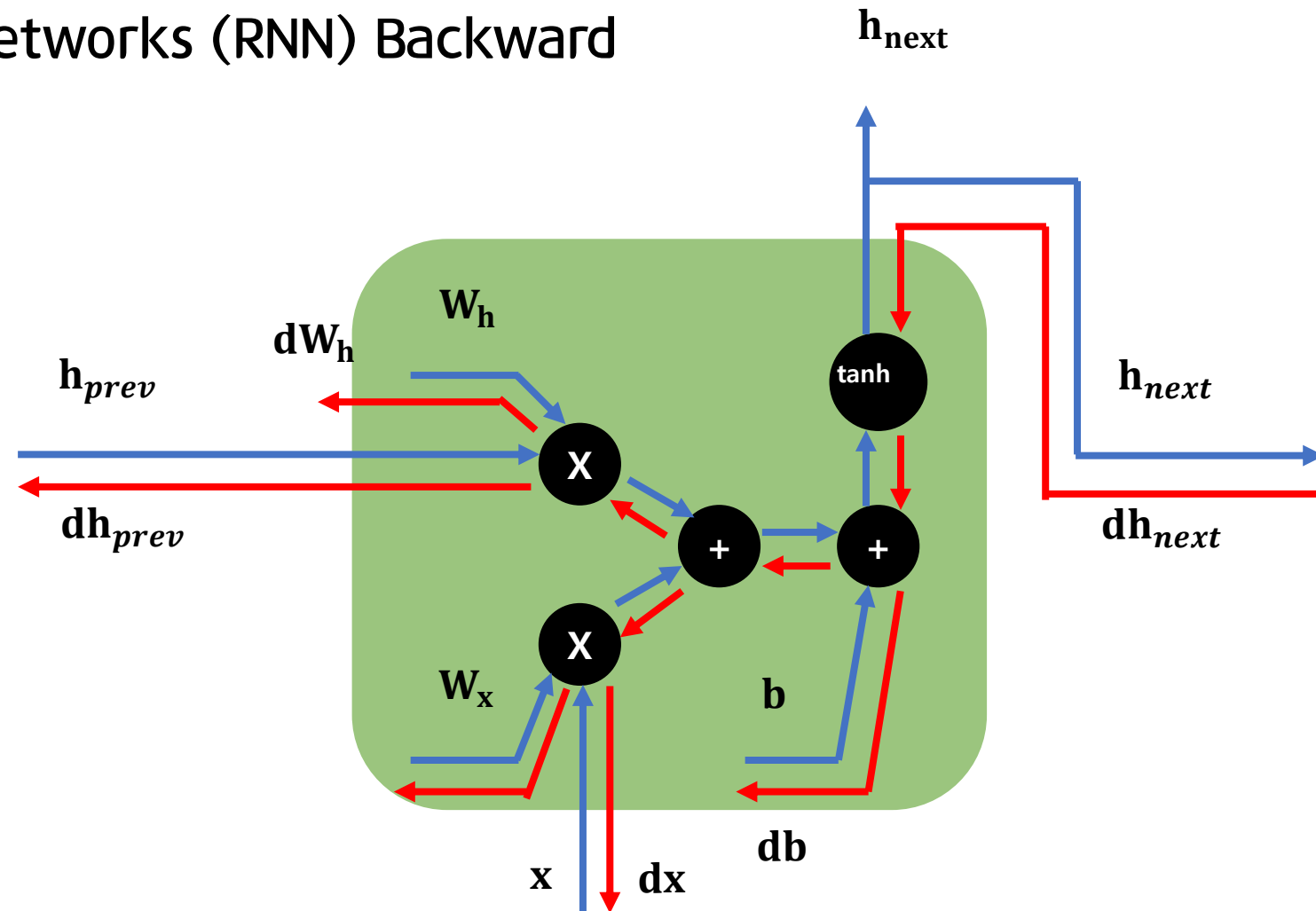
Unit 02 | RNN

- Recurrent Neural Networks (RNN) Backward



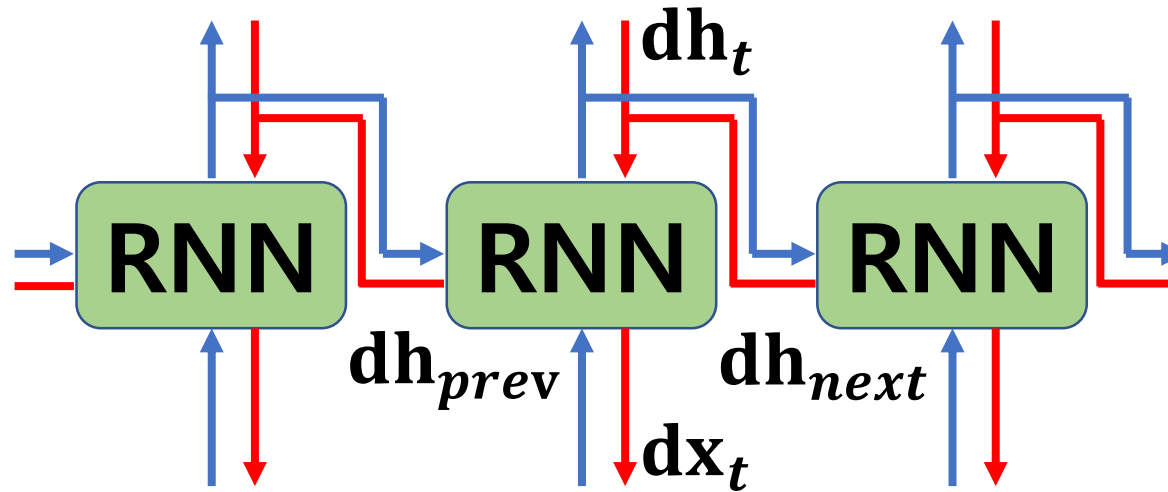
Unit 02 | RNN

- Recurrent Neural Networks (RNN) Backward



Unit 02 | RNN

- Recurrent Neural Networks (RNN) Backward
 - BPTT (Back Propagation Through Time)
RNN은 시간축방향으로 이어져있어, backward시 출력 방향 뿐만 아니라 다음 시점에서부터도 gradient가 흘러 들어온다.

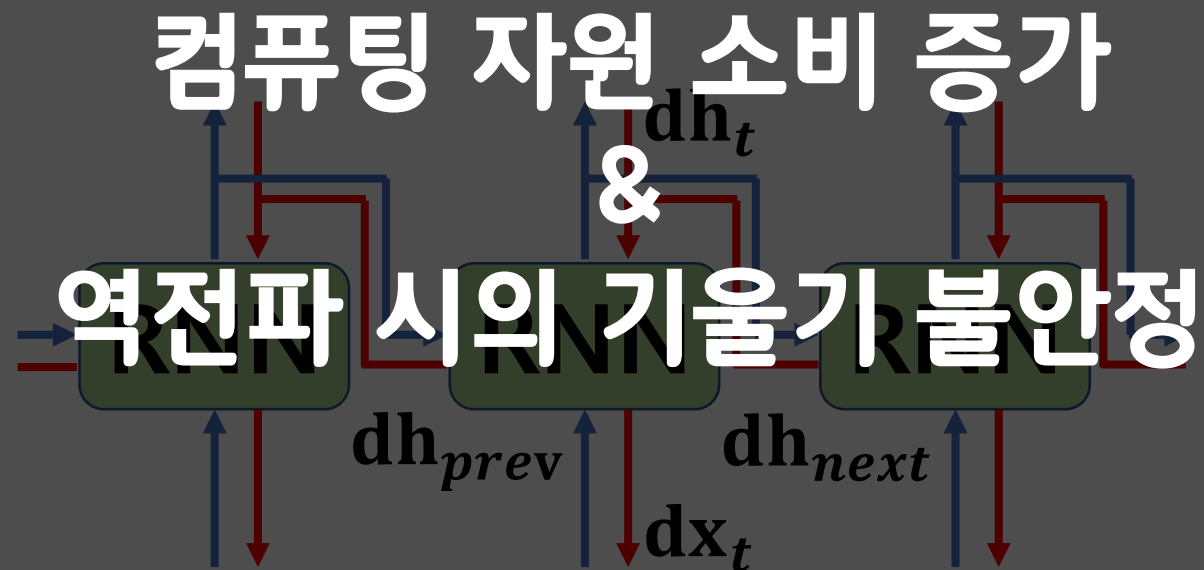


Unit 02 | RNN

• Recurrent Neural Networks (RNN) Backward

- BPTT (Backpropagation Through Time)

RNN은 시간축방향으로 이어져있어, backward시 출력 방향 뿐만 아니라 다음 시점에서부터도 gradient가 흘러 들어온다.

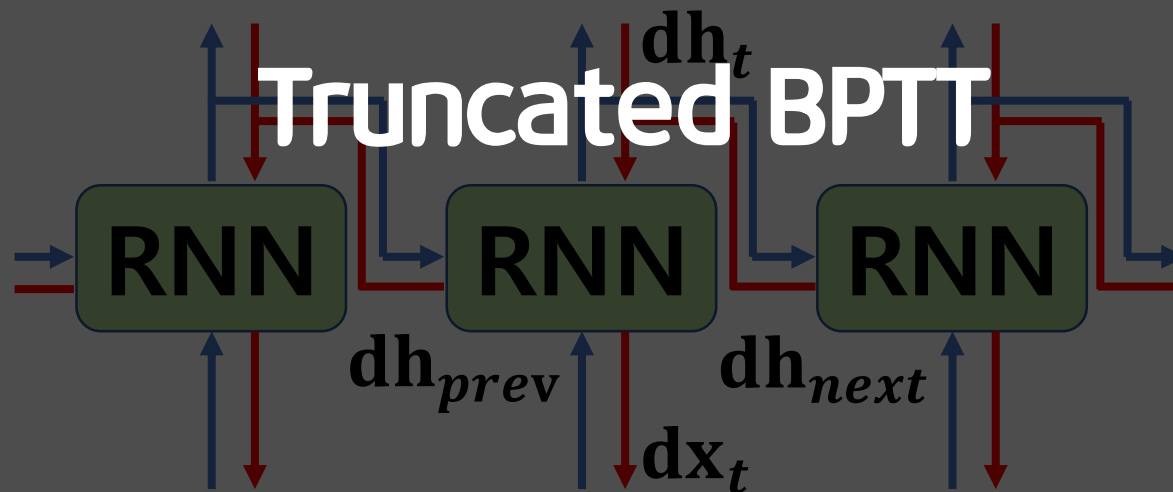


Unit 02 | RNN

- Recurrent Neural Networks (RNN) Backward
 - BPTT (Back Propagation Through Time)

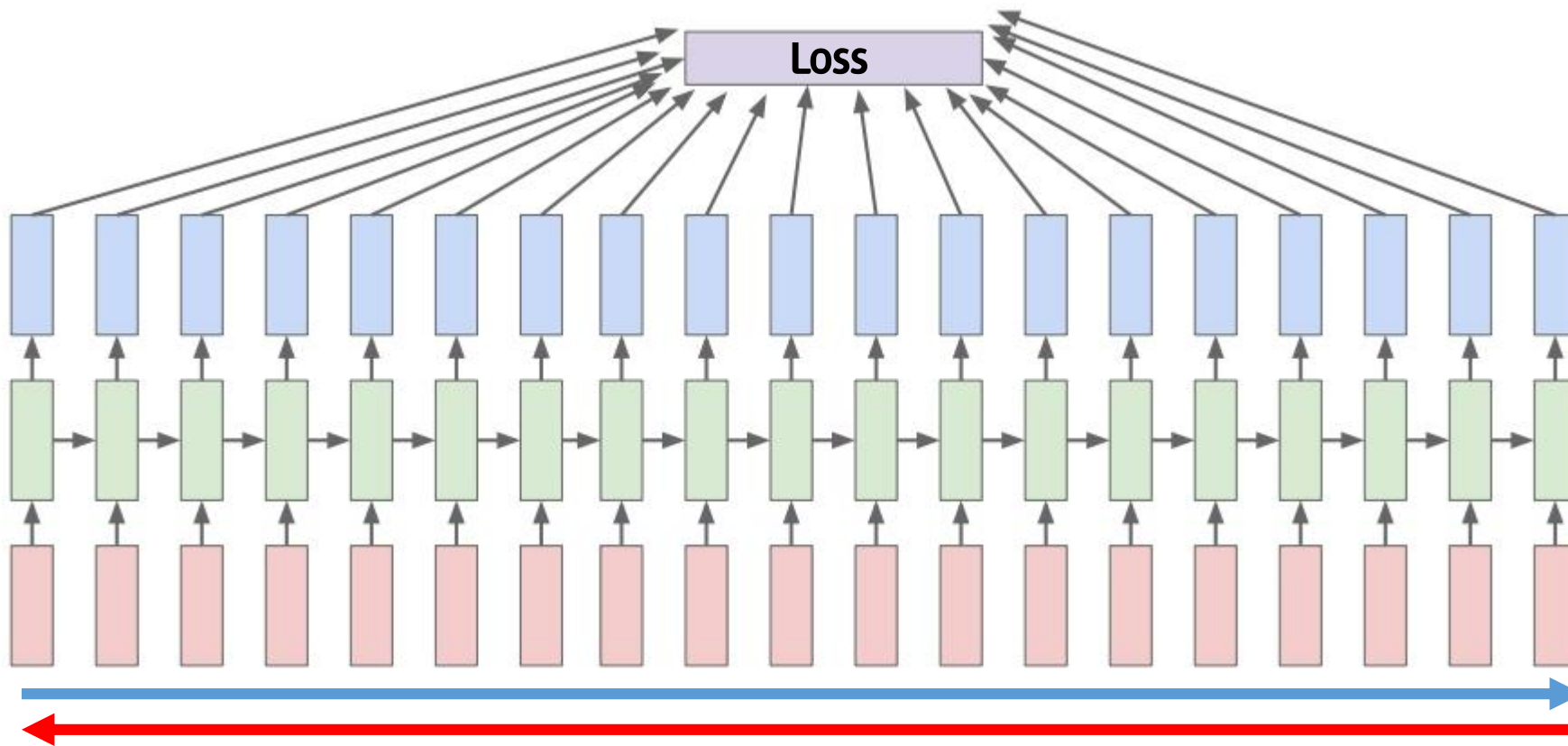
RNN은 시간축방향으로 이어져있어 backward시 축력 방향 뿐만 아니라
다음 시점에서부터도 gradient가 흐를 수 있다.

시간 축 방향의 연결을 적당히 끊어 학습



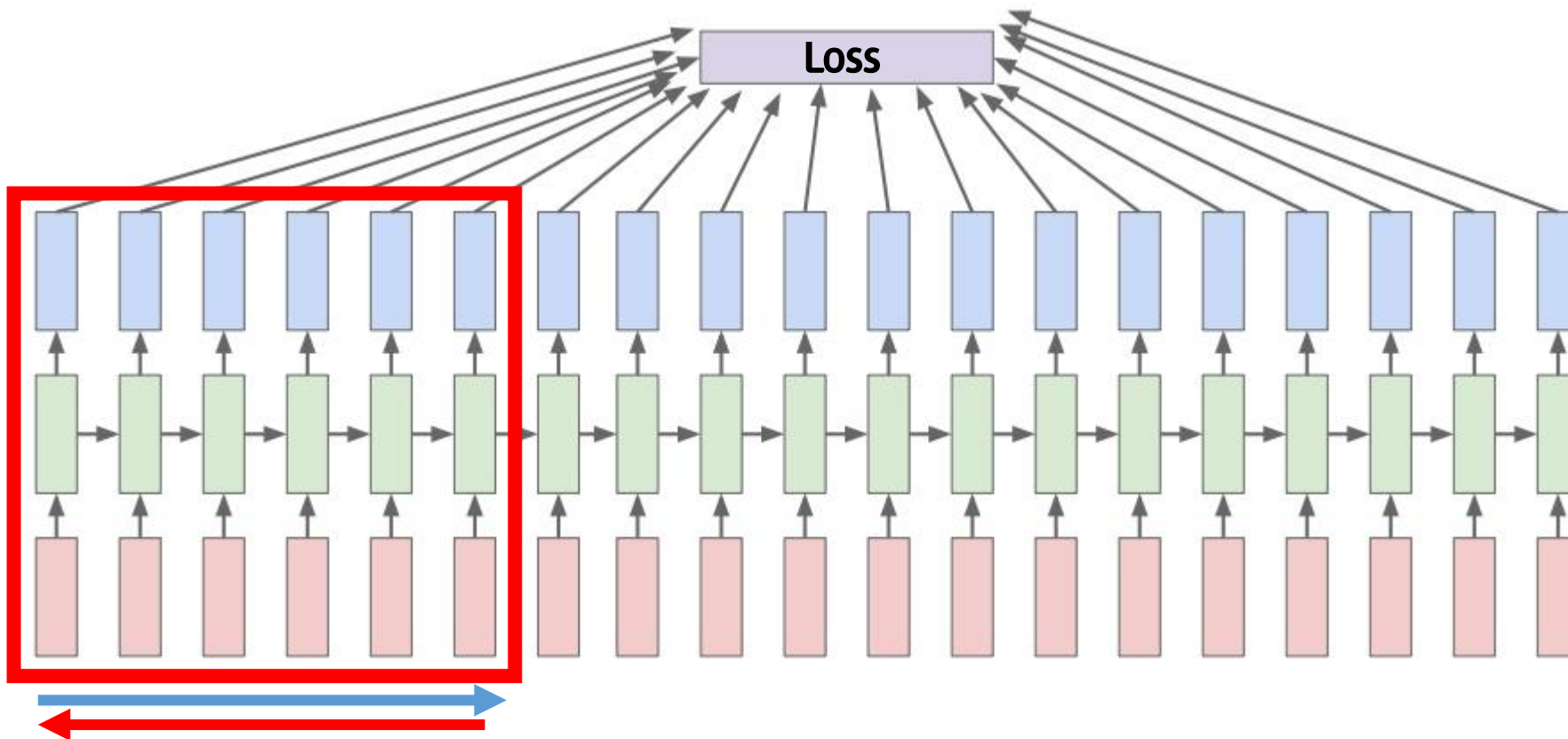
Unit 02 | RNN

- Recurrent Neural Networks (RNN)의 Back propagation : Truncated BPTT



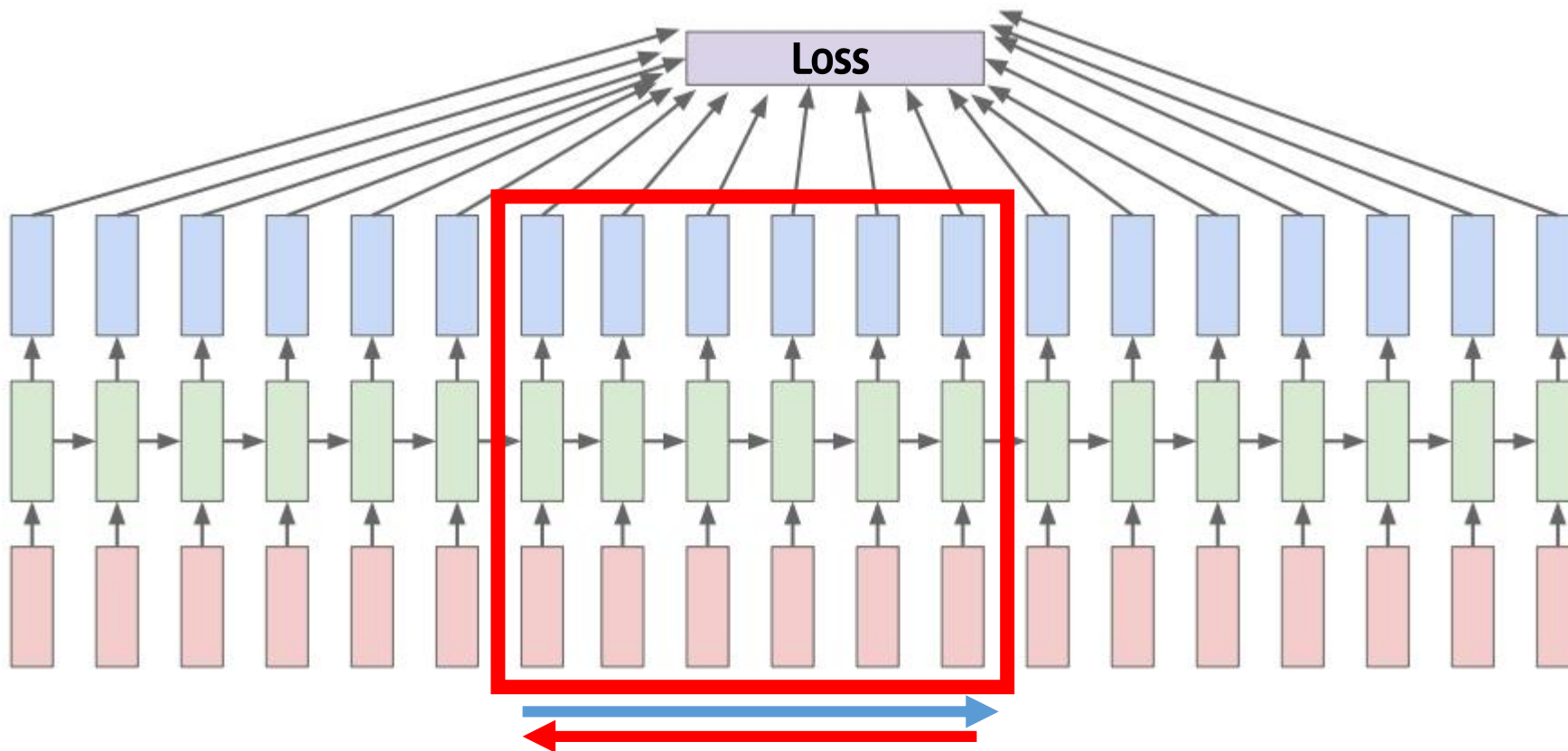
Unit 02 | RNN

- Recurrent Neural Networks (RNN)의 Back propagation (BPTT) : Truncated BPTT



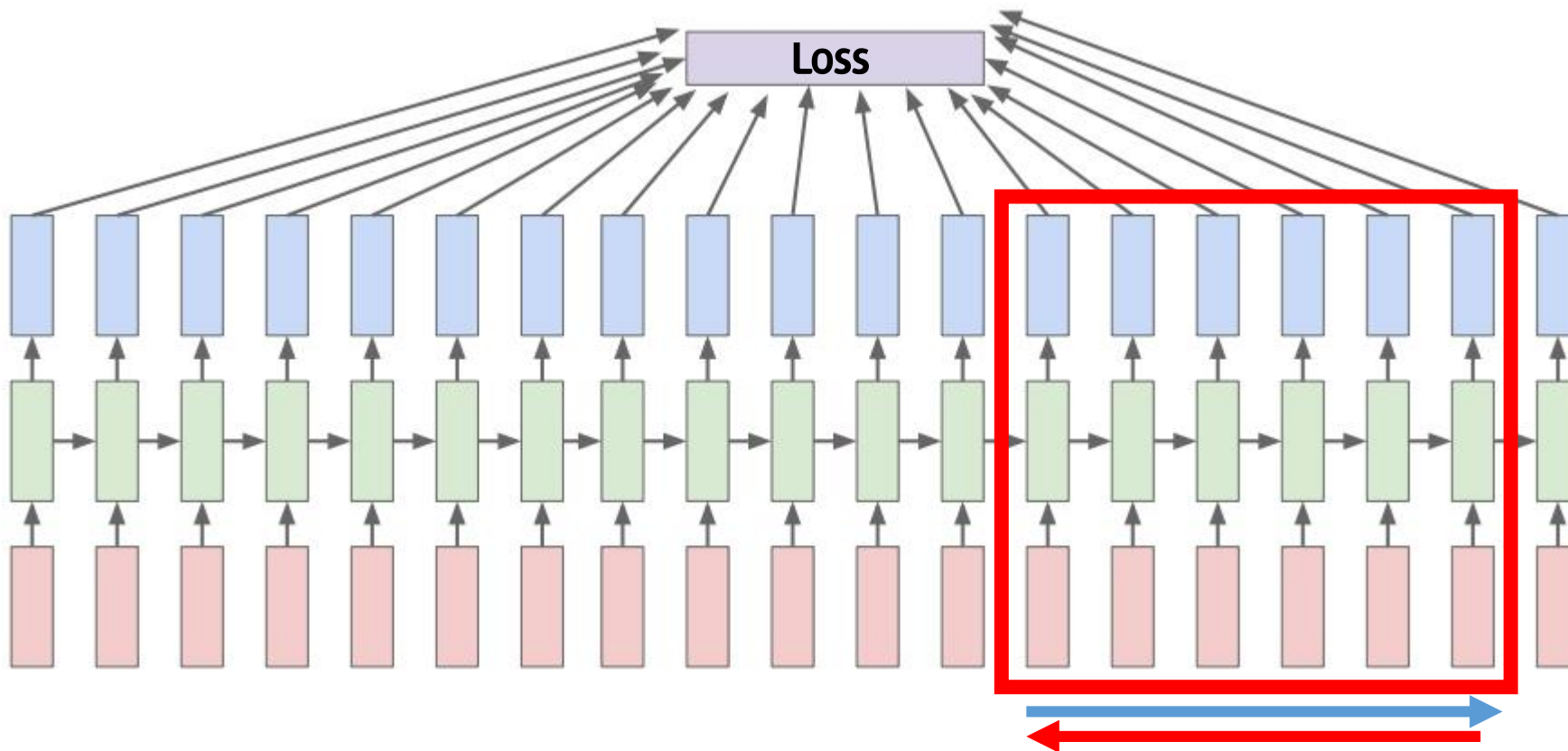
Unit 02 | RNN

- Recurrent Neural Networks (RNN)의 Back propagation (BPTT) : Truncated BPTT



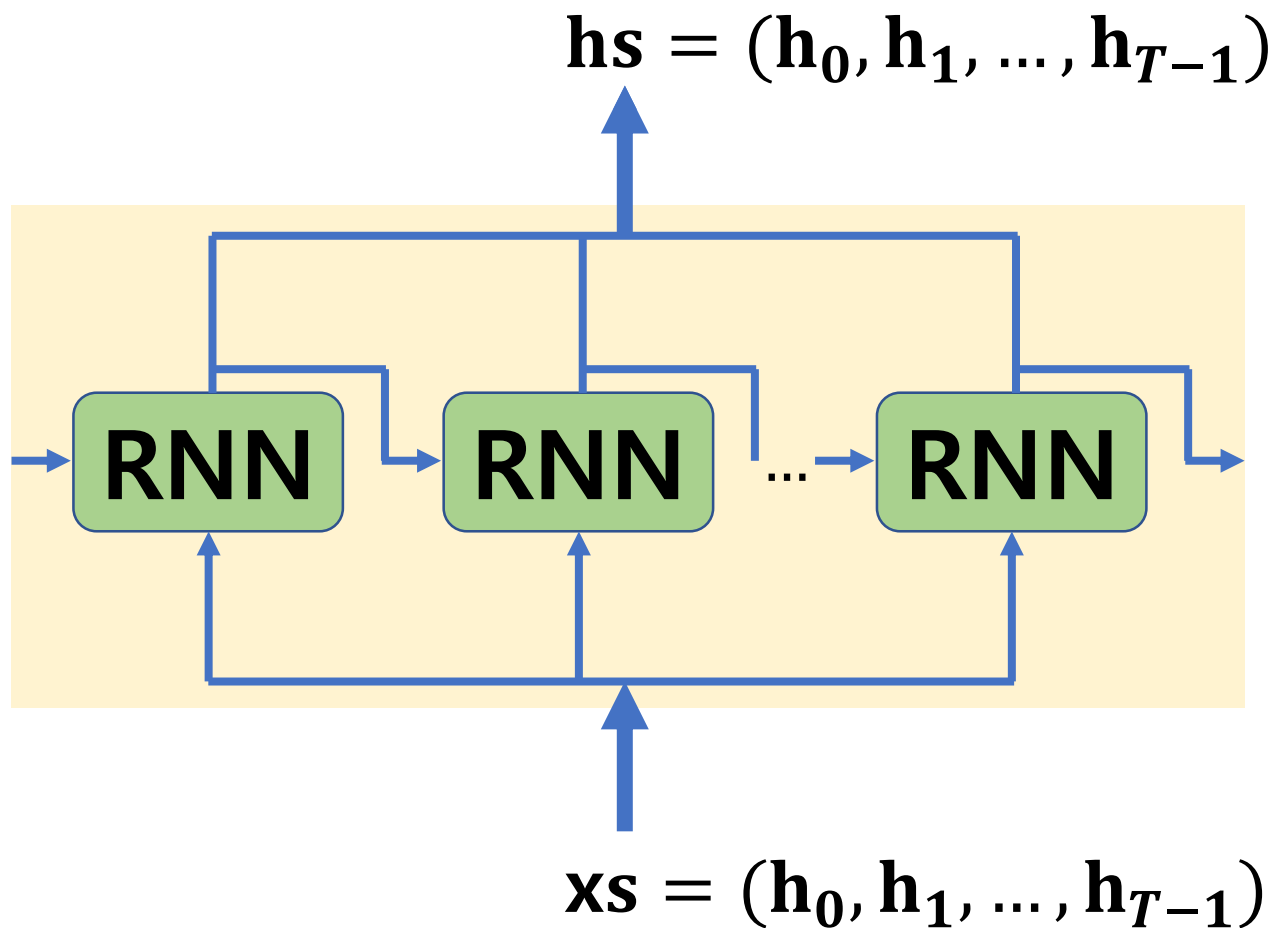
Unit 02 | RNN

- Recurrent Neural Networks (RNN)의 Back propagation (BPTT) : Truncated BPTT



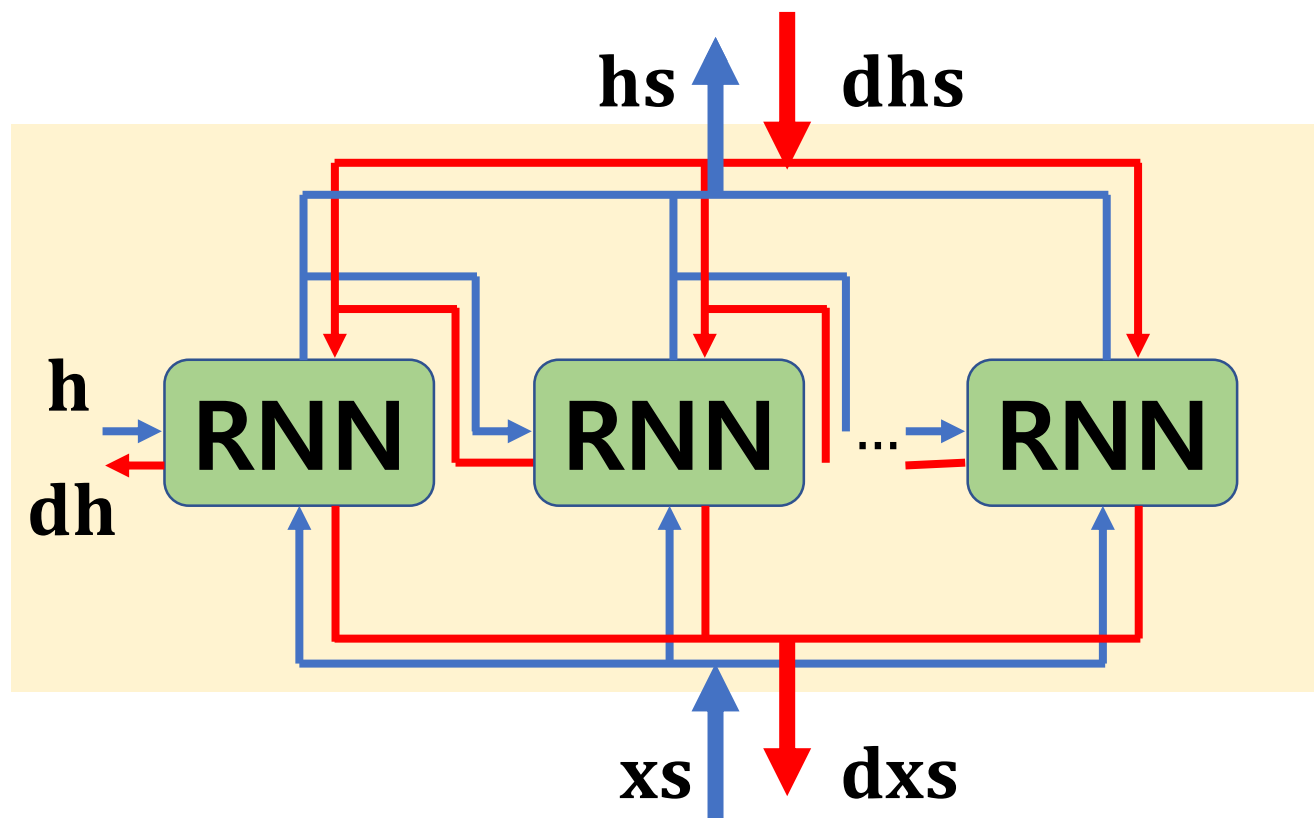
Unit 02 | RNN

- Recurrent Neural Networks (RNN) : Truncated BPTT



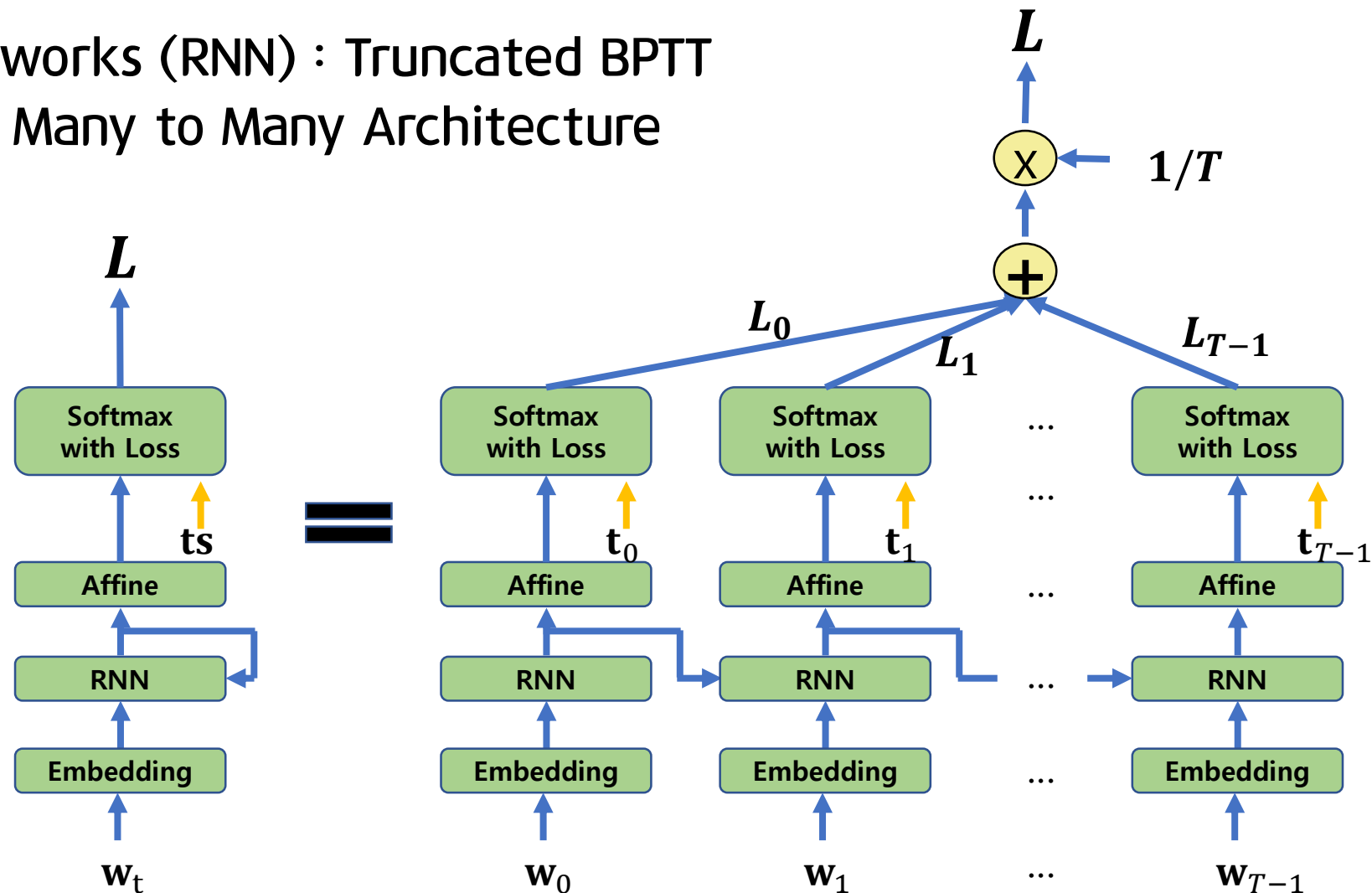
Unit 02 | RNN

- Recurrent Neural Networks (RNN) : Truncated BPTT



Unit 02 | RNN

- Recurrent Neural Networks (RNN) : Truncated BPTT
 - Language Model : Many to Many Architecture

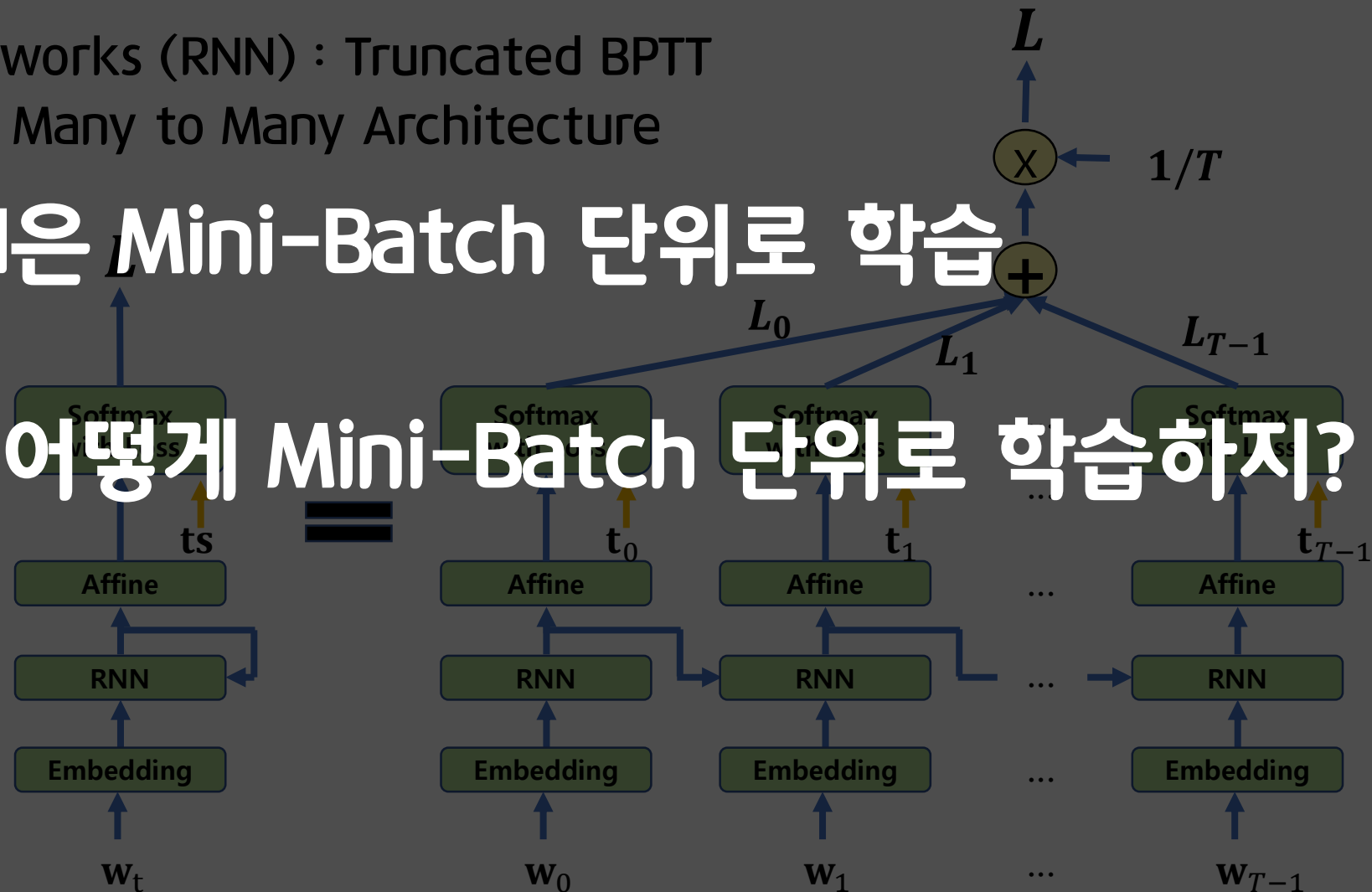


Unit 02 | RNN

- Recurrent Neural Networks (RNN) : Truncated BPTT
 - Language Model : Many to Many Architecture

NN은 Mini-Batch 단위로 학습

그럼 RNN은 어떻게 Mini-Batch 단위로 학습하지?



Unit 02 | RNN

• RNN의 Mini-Batch Training

• Example :

- 1,000개의 단어로 이루어진 데이터
- 시각 T를 10 단위로 잘라 Truncated BPTT로 학습 (T=10)
- Batch Size(N) = 2

0	1	2	3	996	997	998	999
---	---	---	---	-----	-----	-----	-----	-----	-----

0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	997	998	999																																						
0	여	섯	살	적	에	나	는	"체	험	한	이	야	기"	라	는	제	목	의,	원	시	림	에	관	한	책	에	서	...	보	물	을	열	심	히	들	여	다	보	았	다.	\n	"다	른	별	들"	이	라	는,	그	가	슬	쩍	내	비	친	비	밀	에

1 rows × 1000 columns

Unit 02 | RNN

• RNN의 Mini-Batch Training

• Example :

- 1,000개의 단어로 이루어진 데이터
- 시각 T를 10 단위로 잘라 Truncated BPTT로 학습 (T=10)
- Batch Size(N) = 2
- 한 Epoch 당 Mini-Batch의 개수 = $1,000 / (2 * 10) = 50$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534

	0	1	2	...	30	31	32	33	34	35	36	37	38	39	494	495	496	497	498	499	
0	여섯	살	적에	나는	0	이렇게	씩어	있었다.\n	"보아	구렁이는	먹이를	씹지도	않고	통째로	집어삼킨다.그리고는	오니까	말이다.\n	어쨌든	나는	그의	느닷없는
1	출현에너무도	놀라서눈을	휘둥그렇게	뜨고그를	1	목마름과	두려움에	시달리는	것같아	보이지도	않았다.사람	사는	고장에서	수천	마일 !	별들"	이라는,	그가	슬쩍	내비친	비밀에

2 rows × 500 columns

Unit 02 | RNN

- RNN의 Mini-Batch Training

- Example :

- 1,000개의 단어로 이루어진 데이터
- 시각 T를 10 단위로 잘라 Truncated BPTT로 학습 (T=10)
- Batch Size(N) = 2

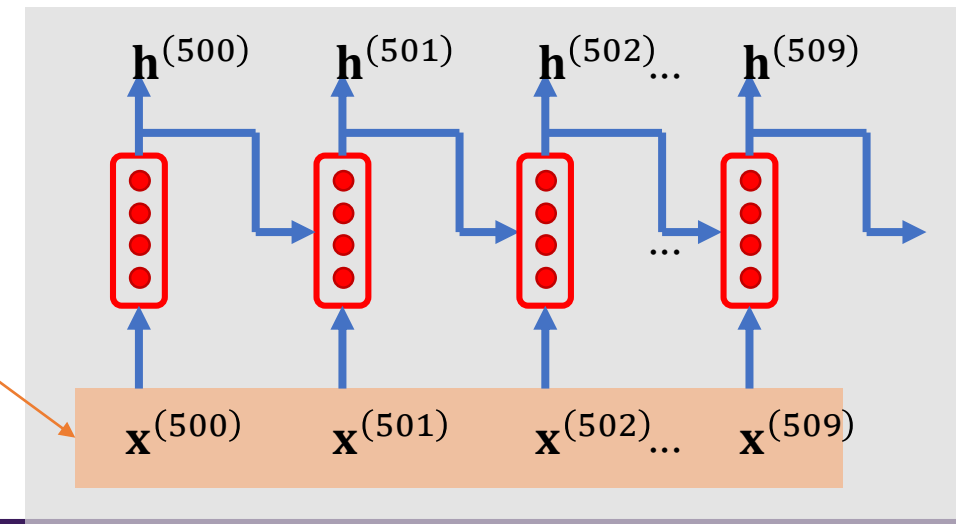
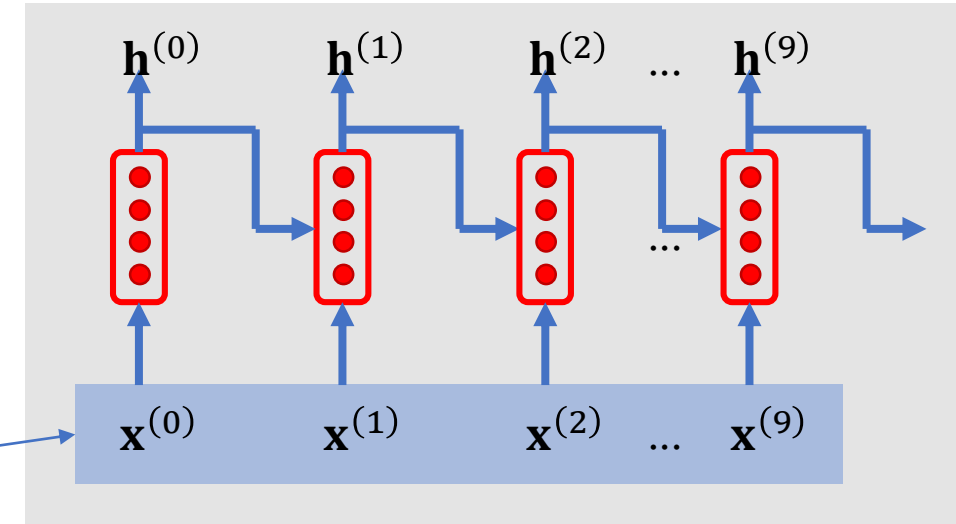
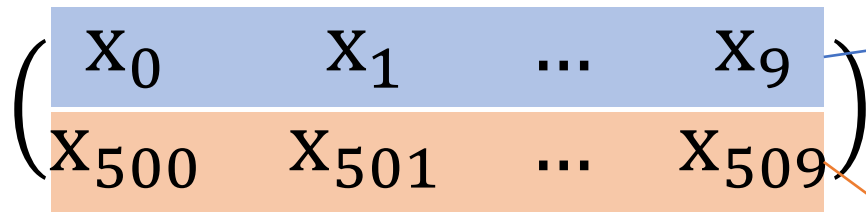
0	1	...	9	10	...	19	20	...	499
500	501	...	509	510	...	519	520	...	999

	0	1	2	3	4	5	6	7	8	9	...	490	491	492	493	494	495	496	497	498	499
0	여섯	살	적에	나는	"체험한	이야기"라는	제목의,	원시림에	관한	책에서	...	아무것도	그리는	연습을	하지	않았으니까	말이다.\n	어쨌든	나는	그의	느닷없는
1	출현에너무도	놀라서눈을	휘둥그렇게	뜨고그를	바라보았다.내가	사람	사는	고장에서	수천	마일	...	보물을	열심히	들여다보았다.\n	"다른	별들"	이라는,	그가	슬쩍	내비친	비밀에

2 rows × 500 columns

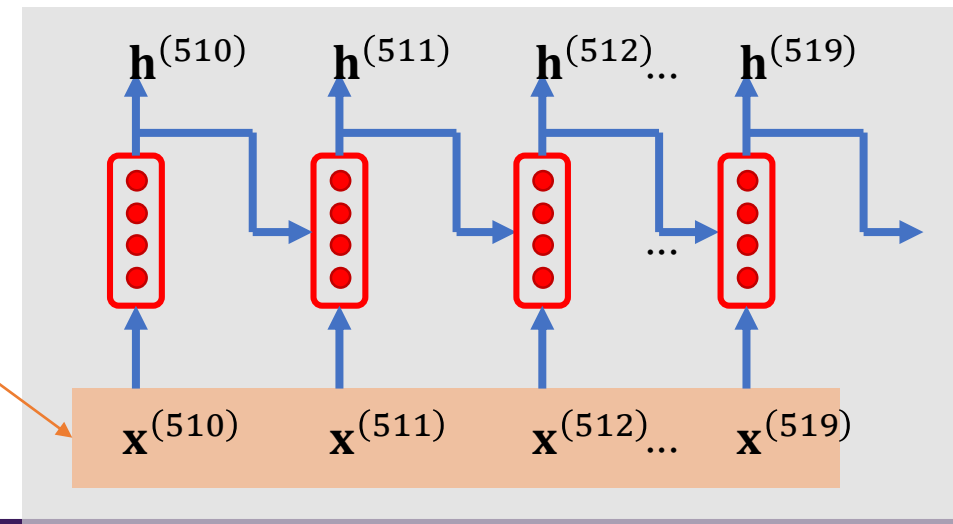
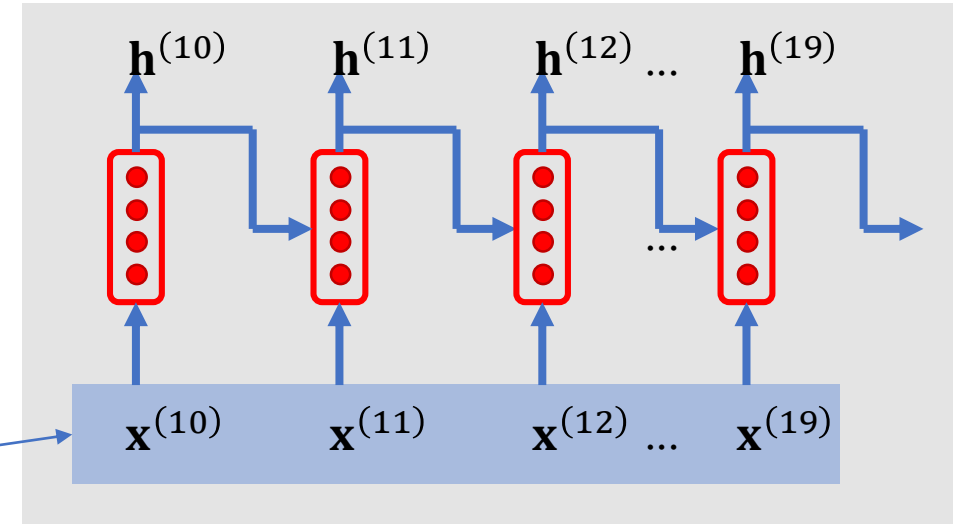
Unit 02 | RNN

• RNN의 Mini-Batch Training



Unit 02 | RNN

• RNN의 Mini-Batch Training



Unit 02 | RNN

- Recurrent Neural Networks (RNN) Implementation

실습

01_RNN&RNNLM.ipynb

Unit 03 | LSTM

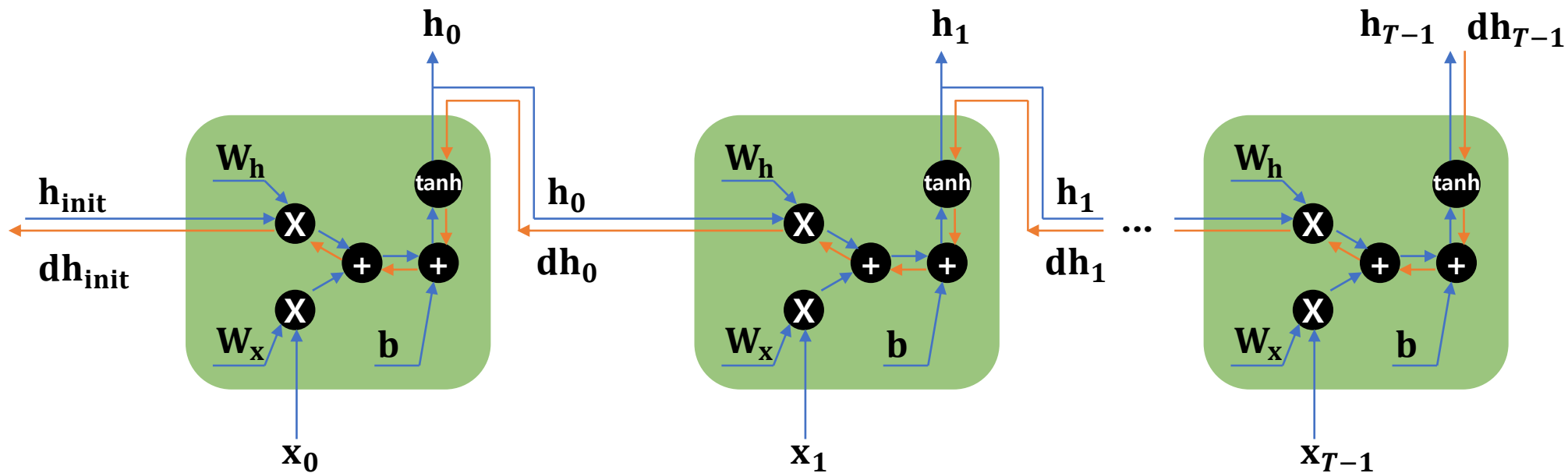
LSTM

Unit 03 | LSTM

- RNN의 문제점
 - Vanishing(또는 exploding) gradient problem
 - 이를 개선하기 위한 새로운 형태의 RNN : **LSTM** and **GRU**
 - Vanishing (또는 exploding) gradient의 다른 해결 방안
 - Gradient clipping
 - Skip connections (ex. ResNet)

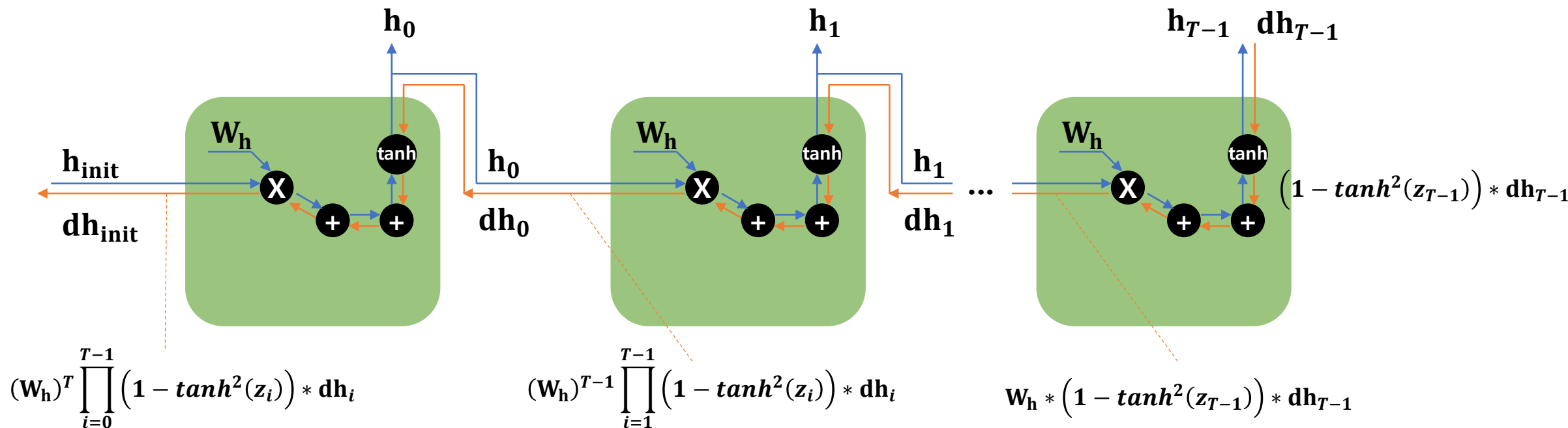
Unit 03 | LSTM

- Vanishing(or Exploding) Gradient Problem
- 원인 : $\tanh()$ 와 W_h 의 행렬곱



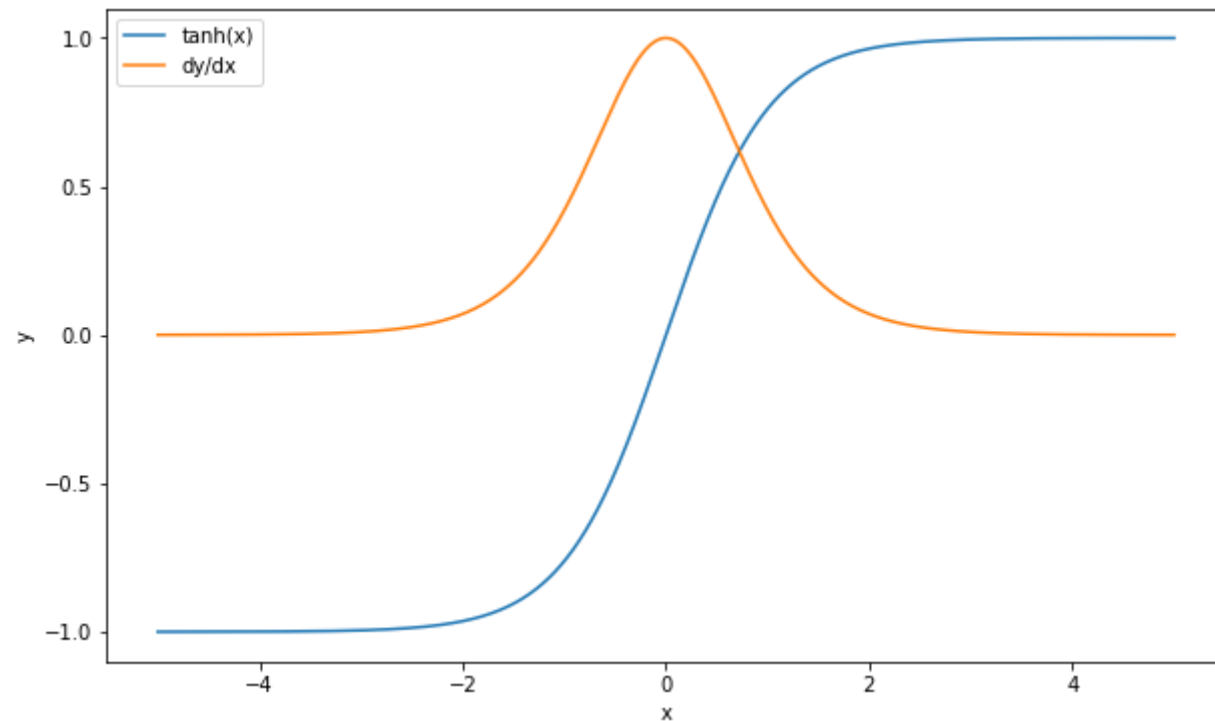
Unit 03 | LSTM

- Vanishing(or Exploding) Gradient Problem
- 원인 : $\tanh()$ 와 W_h 의 행렬곱 : \tanh 의 미분과 W_h 가 T의 길이만큼 누적해 곱해짐



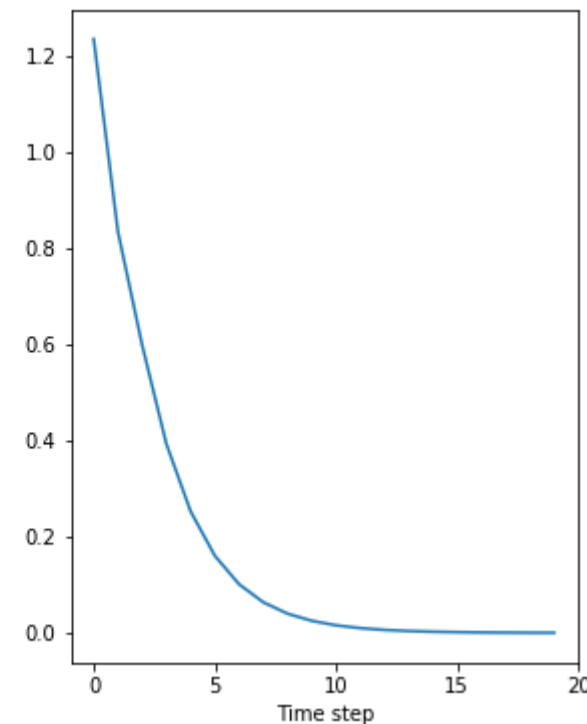
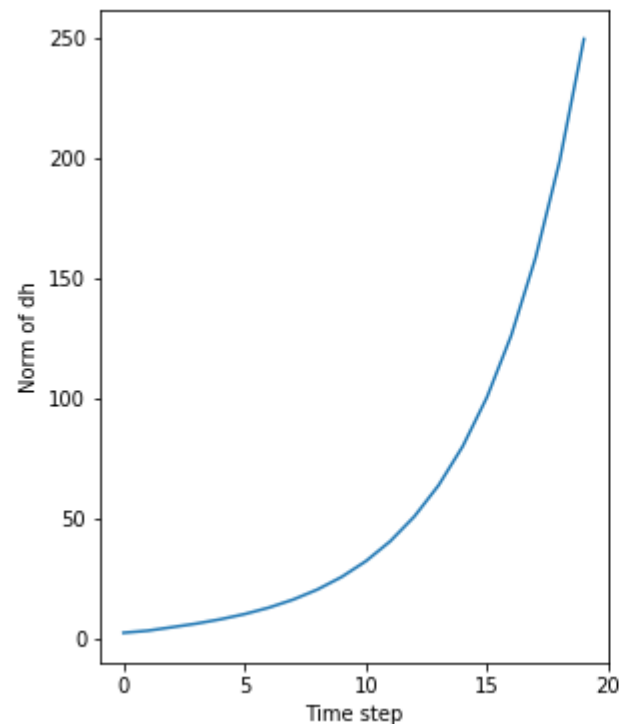
Unit 03 | LSTM

- Vanishing(or Exploding) Gradient Problem
- 원인 : $\tanh()$ 와 행렬곱 : \tanh 의 미분
- $0 < 1 - \tanh^2(z_{T-1}) < 1$
- 1보다 작은 값이 계속 곱해져 Gradient의 크기가 줄어든다



Unit 03 | LSTM

- Vanishing(or Exploding) Gradient Problem
- 원인 : $\tanh()$ 와 행렬곱 : W_h 의 누적 행렬곱
- W_h 의 특잇값(고윳값)의 최댓값 > 1 이면 Exploding할 가능성이 높고
- W_h 의 특잇값(고윳값)의 최댓값 < 1 이면 Vanishing할 가능성이 높다.



Unit 03 | LSTM

- RNN의 문제점 해결 방안
 - Gradient clipping

If $\|\hat{\mathbf{g}}\| \geq threshold$:

$$\hat{\mathbf{g}} = \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$$

Unit 03 | LSTM

• RNN의 문제점 해결 방안

• Gradient clipping

```
import numpy as np

dw1 = np.random.rand(3, 3) * 10
dw2 = np.random.rand(3, 3) * 10
grads = [dw1, dw2]
max_norm = 5.0

def clip_grads(grads, max_norm):
    total_norm = 0
    for grad in grads:
        total_norm += np.sum(grad ** 2)
    total_norm = np.sqrt(total_norm)

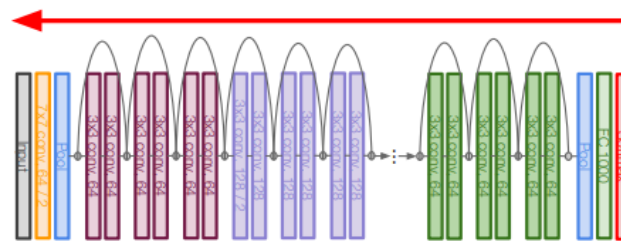
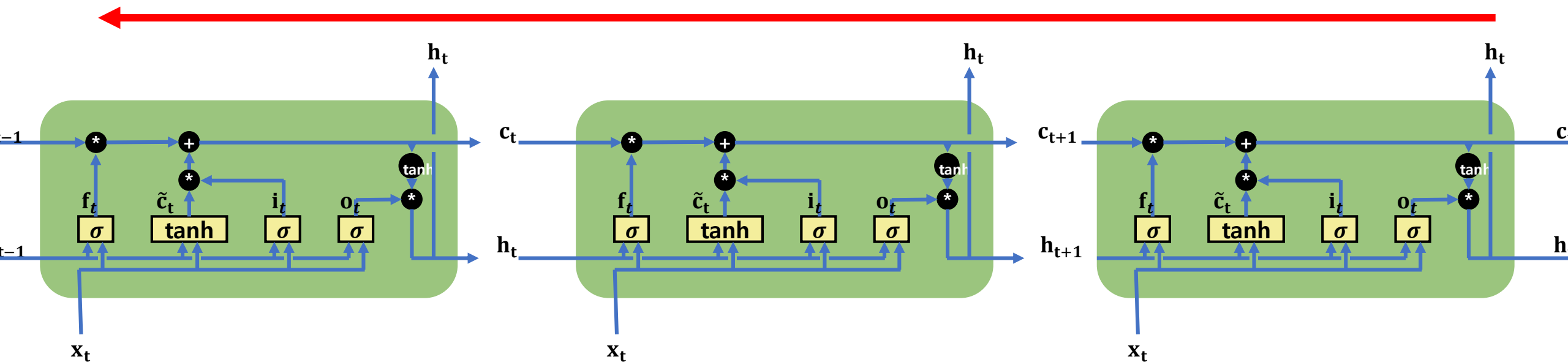
    rate = max_norm / (total_norm + 1e-6)
    if rate < 1:
        for grad in grads:
            grad *= rate
```

Unit 03 | LSTM

- LSTM (Long Short-Term Memory)
 - Long-Term Memory cell “c”를 도입
 - ResNet의 Skip connection과 비슷한 기능으로 gradient 를 유지하여 Vanishing Gradient 개선

Unit 03 | LSTM

- LSTM은 어떻게 vanishing gradient 문제를 해결하는가?



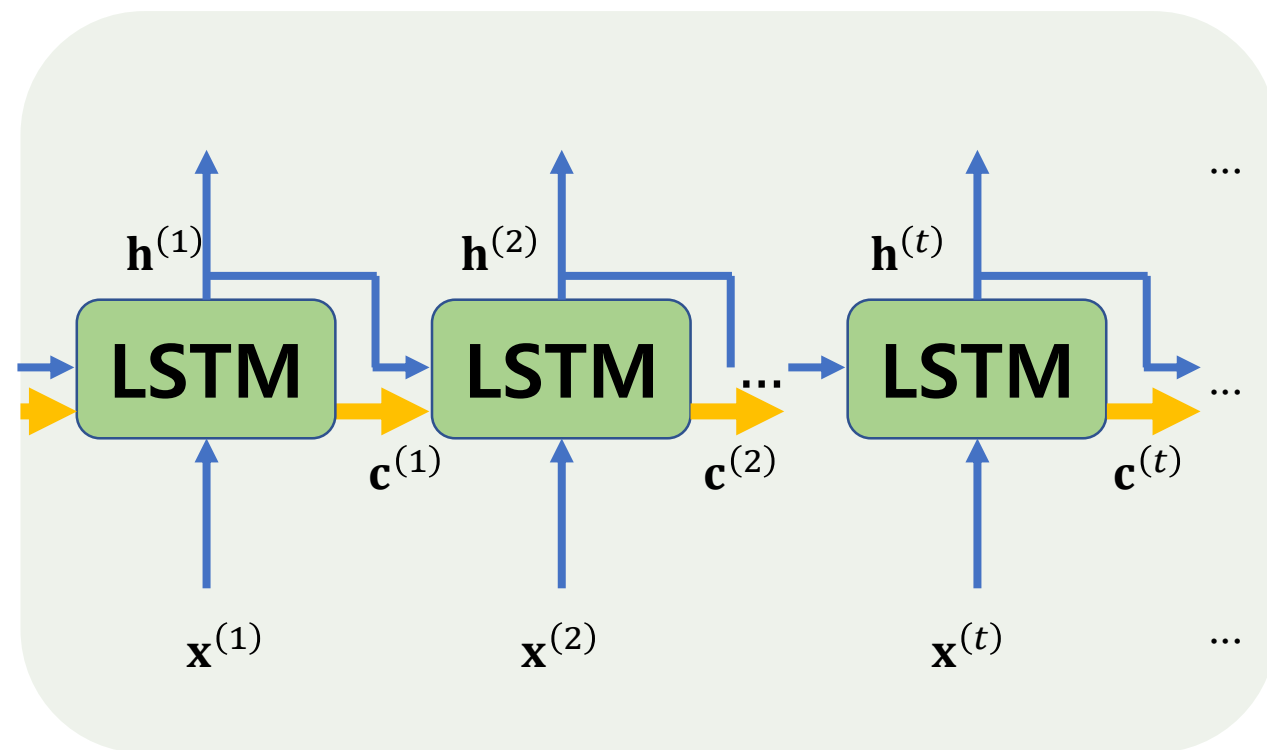
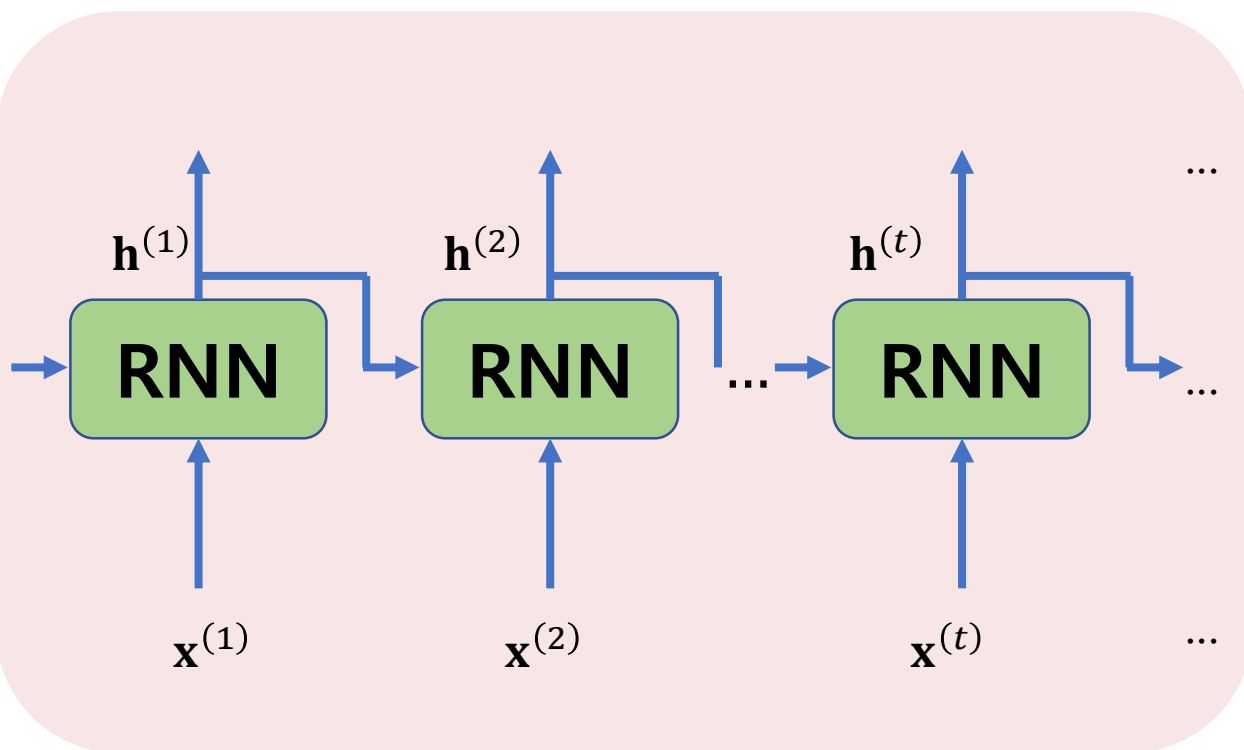
Unit 03 | LSTM

- LSTM (Long Short-Term Memory)
 - 시점 t 에서, hidden state h_t 와 cell state c_t 로 구성
 - LSTM은 Long-Term Memory인 cell c_t 로부터 정보를 **지우고**(잊고), **쓰고**, **읽을** 수 있다.(**erase**, **write**, **read**)
 - Gate를 추가하여 어떤 정보를 지우고/쓰고/읽을지를 조절함.
 - 각 시점 t 에서 gate의 원소들은 각 위치에 해당하는 원소들(정보)을 얼마나 사용할 지, 그 개방 정도를 0~1사이의 가중치로 표현함.
 - 현재의 맥락 정보를 바탕으로 gate의 개방 여부가 계산됨

Unit 03 | LSTM

- LSTM (Long Short-Term Memory)

- RNN과 LSTM 비교



Unit 03 | LSTM

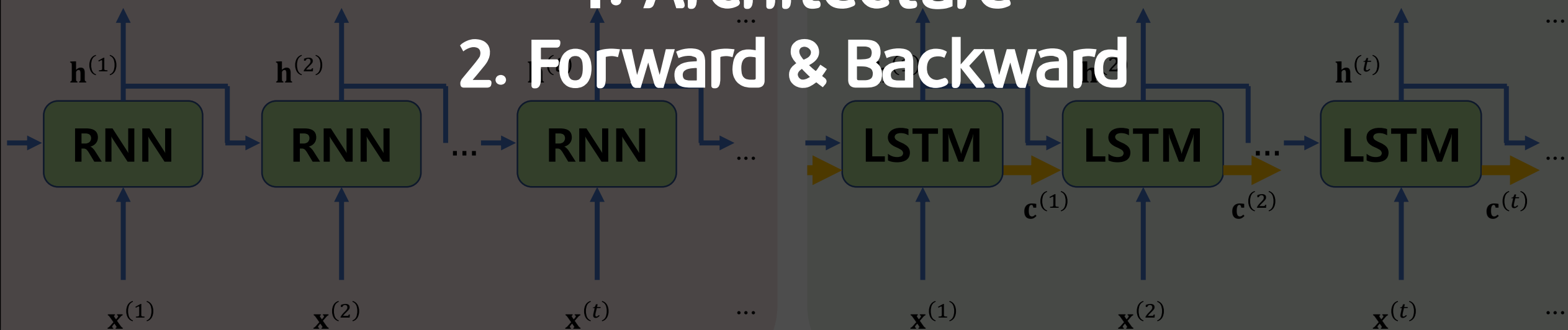
- LSTM (Long Short-Term Memory)

- RNN과 LSTM 비교

Neural Net에서 알아야할 것

1. Architecture

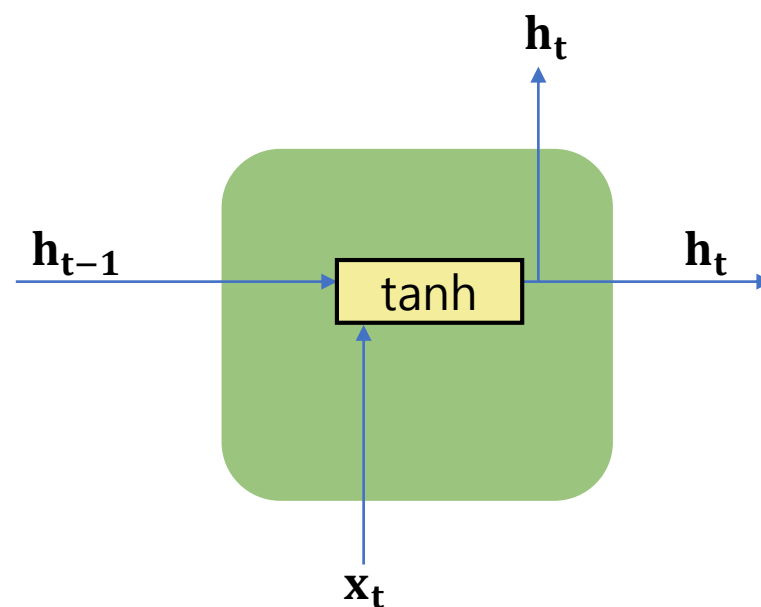
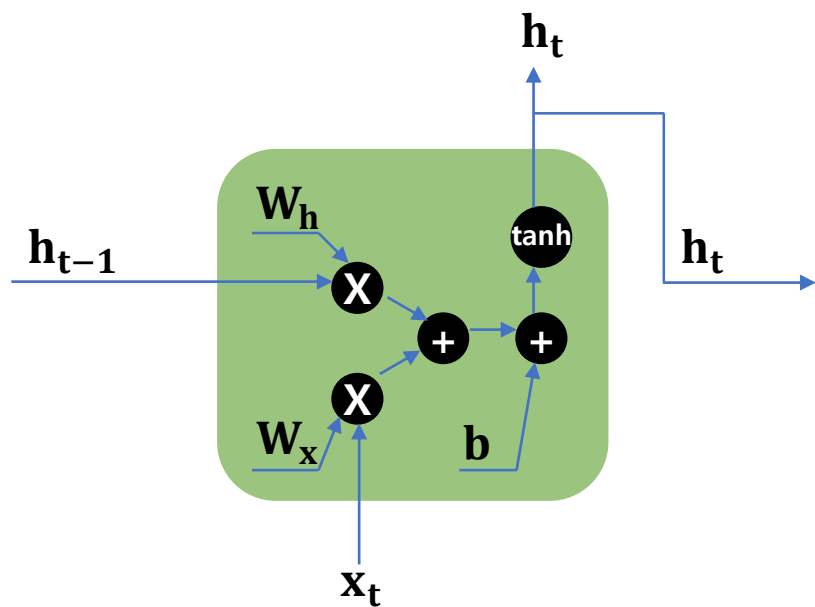
2. Forward & Backward



Unit 03 | LSTM

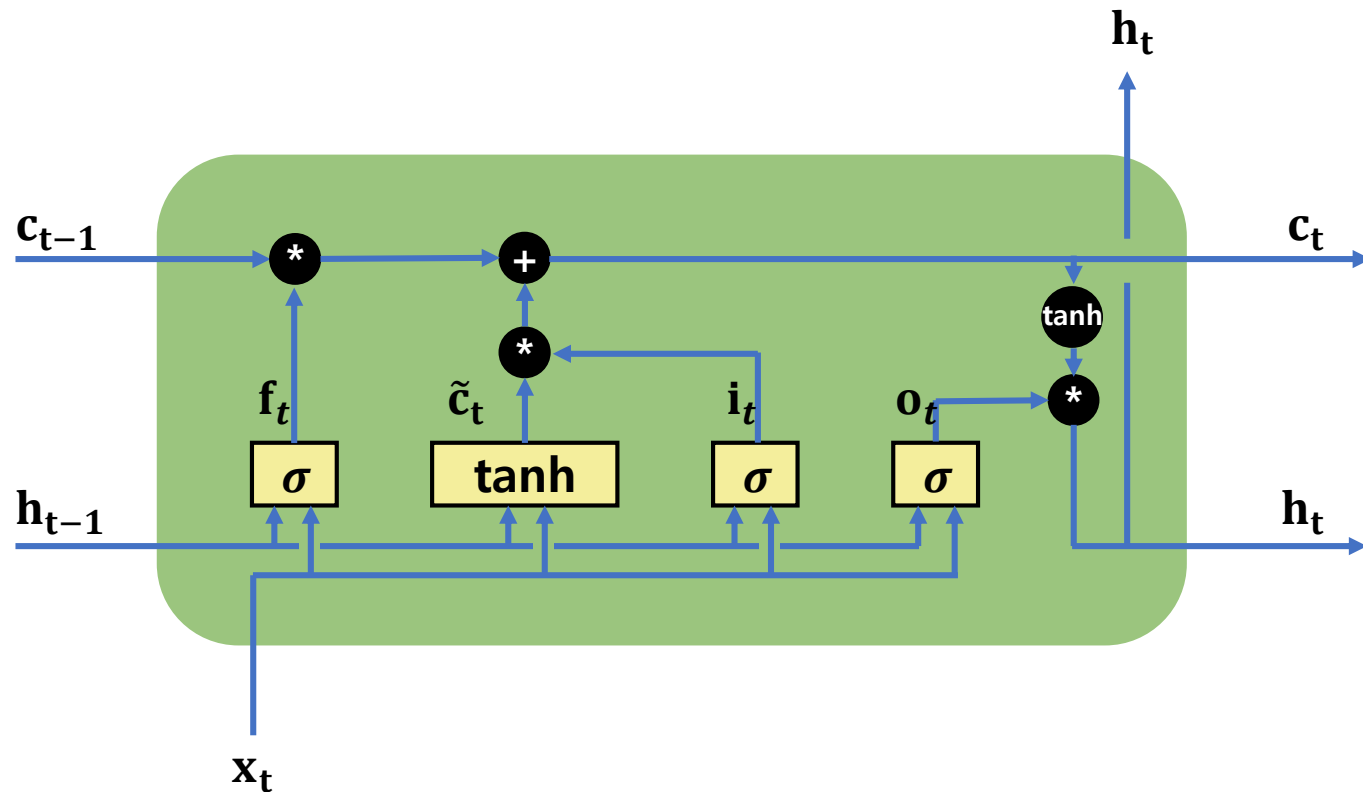
• LSTM

- 표기법 단순화 : 좌측의 연산을 tanh layer(사각형 노드)로 표현.



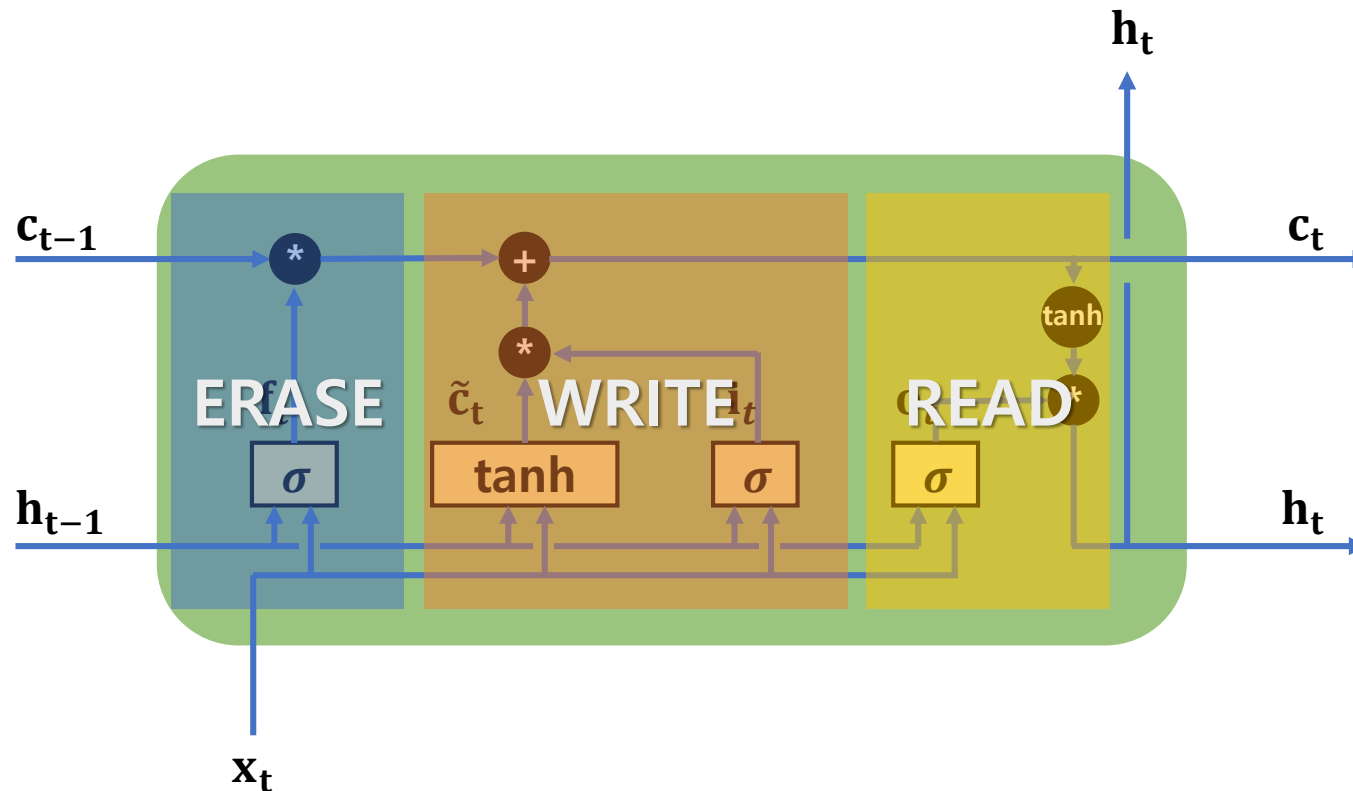
Unit 03 | LSTM

• LSTM의 구조



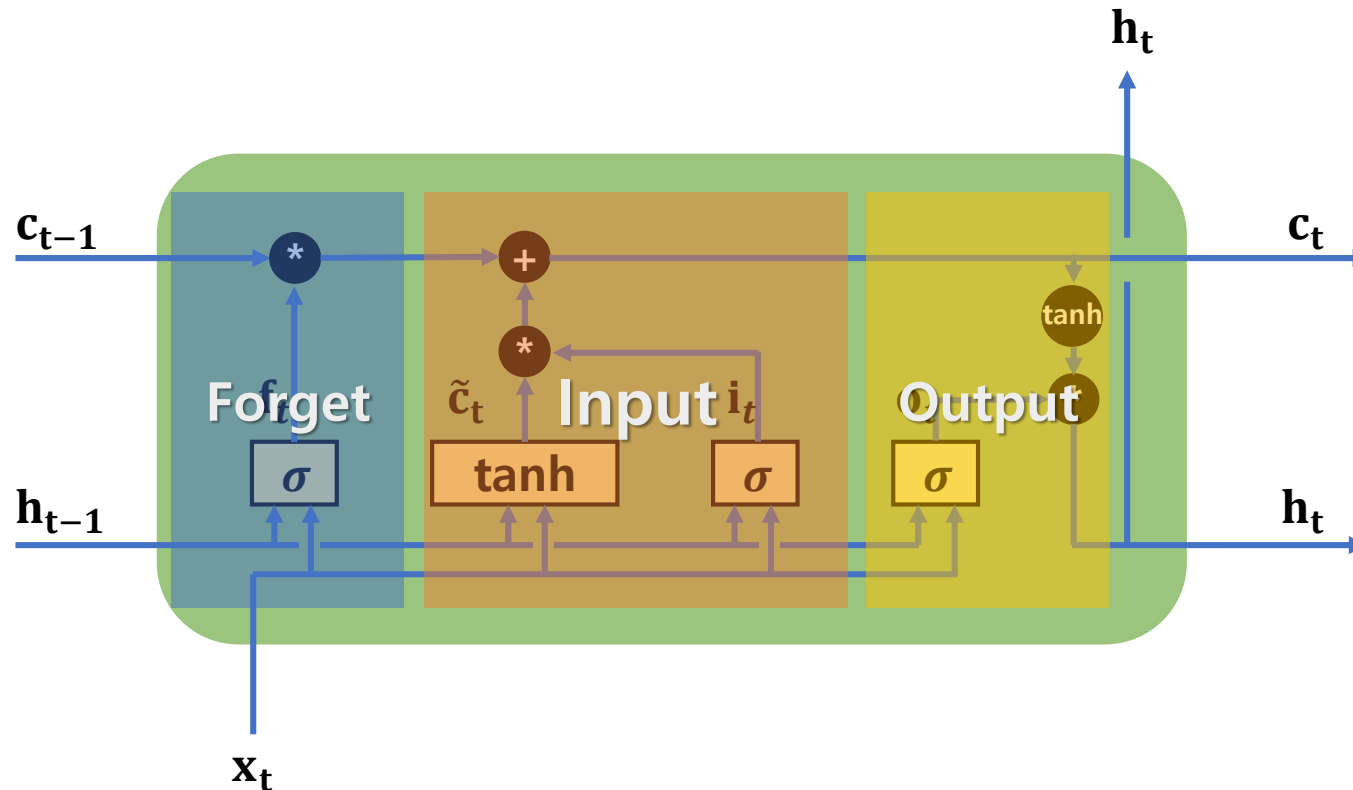
Unit 03 | LSTM

- LSTM의 구조
 - 복잡해보여도 세 가지만 기억하면 된다. Erase, Write, Read



Unit 03 | LSTM

- LSTM의 구조
 - 복잡해보여도 세 가지만 기억하면 된다. Forget, Input, Output



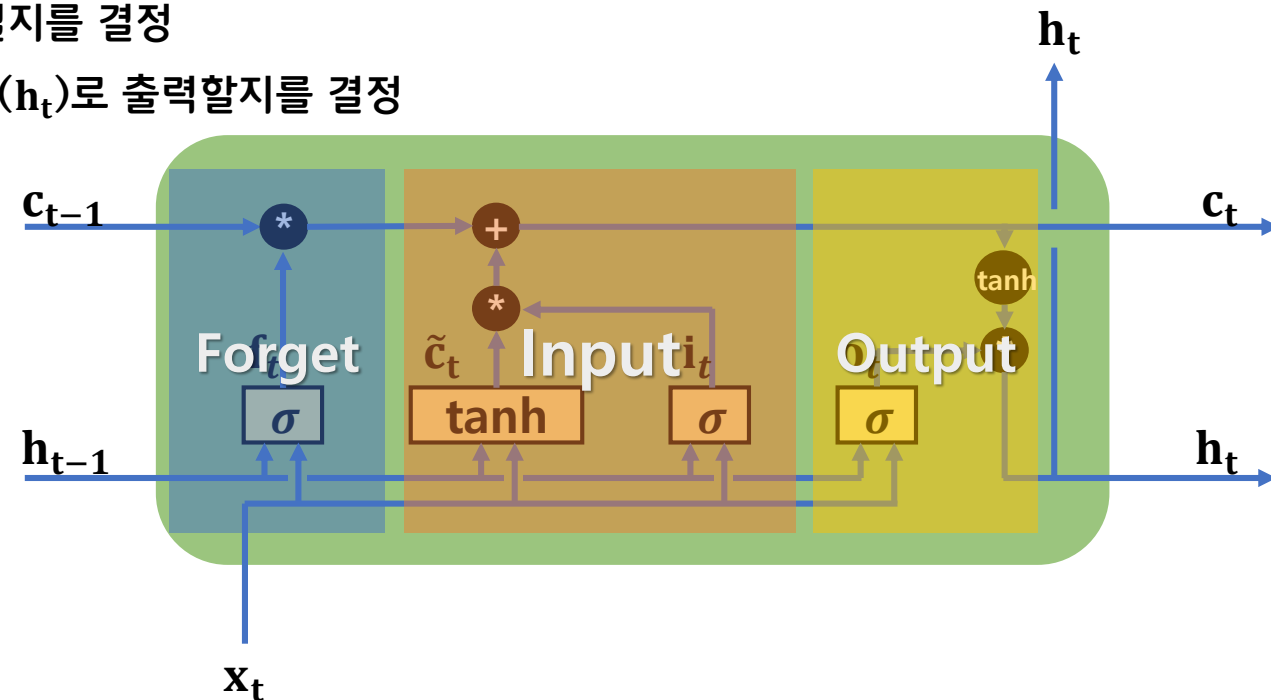
Unit 03 | LSTM

• LSTM의 구조

• 3개의 Gate로 구성되며

각각은 현재 입력 정보(h_{t-1} 와 x_t)를 기준으로

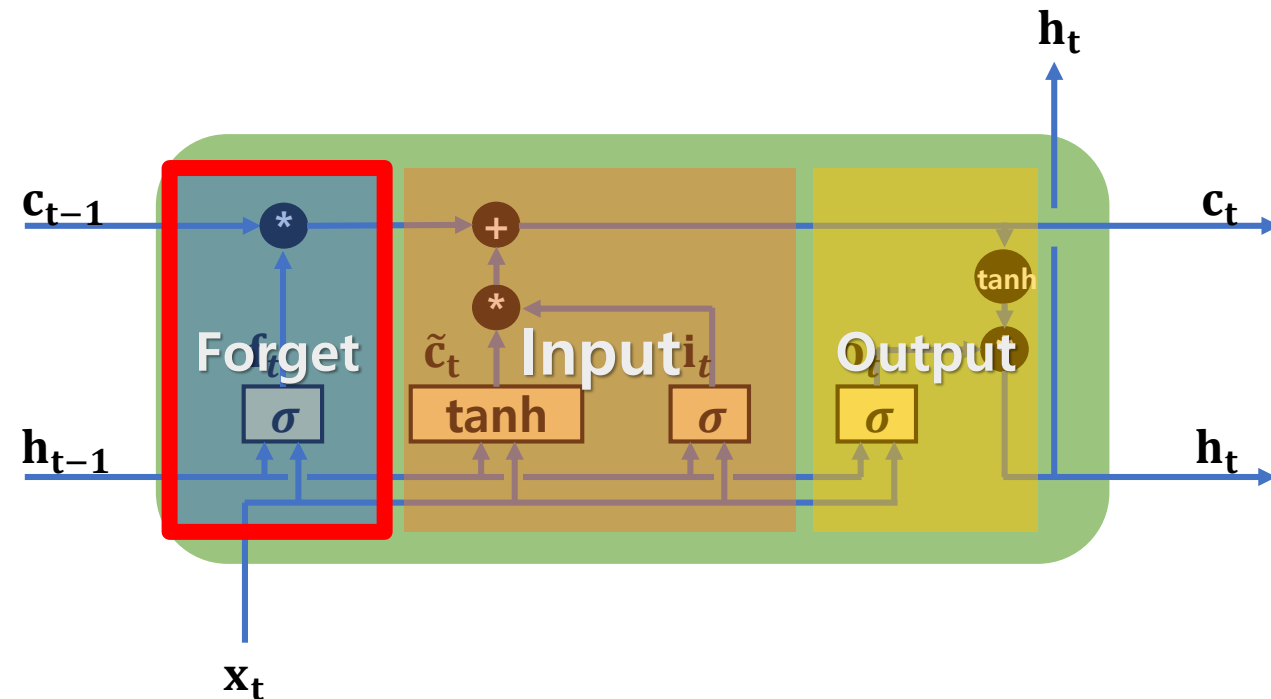
- Forget gate : 장기기억(c_{t-1})에서 무엇을 잊을지를 결정
- Input gate : 장기기억(c_{t-1})에 무엇을 추가 시킬지를 결정
- Output gate : 장기기억(c_t) 중 무엇을 단기기억(h_t)로 출력할지를 결정



Unit 03 | LSTM

- LSTM의 구조 (1) : Forget Gate

- 현재 입력 정보(h_{t-1} 와 x_t)를 기준으로
이전 시점의 장기기억(c_{t-1})에서 무엇을 잊을지를 결정



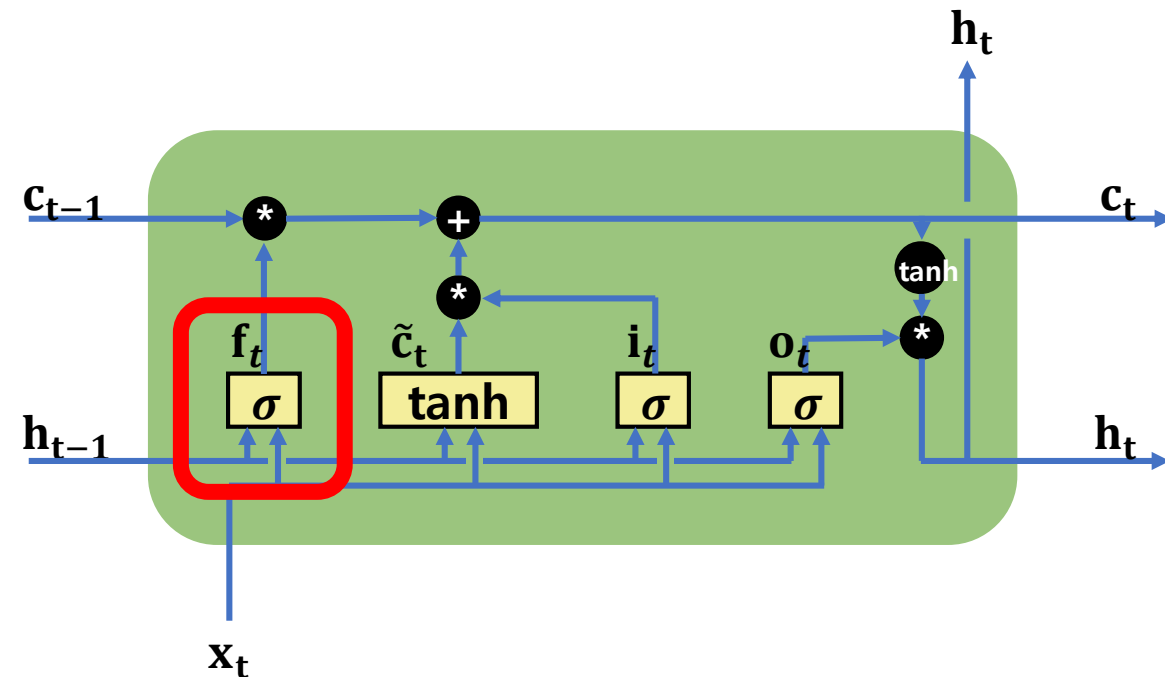
Unit 03 | LSTM

• LSTM의 구조 (1) : Forget Gate

• Forget gate의 연산

- 다음 수식을 통해 계산되며,
현재 시점(t)에서 입력 받은 정보(x)와 단기 기억 (h_{t-1})을 Affine 변환한 값을
Sigmoid를 적용해 0~1사이의 값으로 변환
- N x H 크기의 0 ~ 1 사이의 값을 갖는 행렬

$$f_t = \sigma(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)})$$

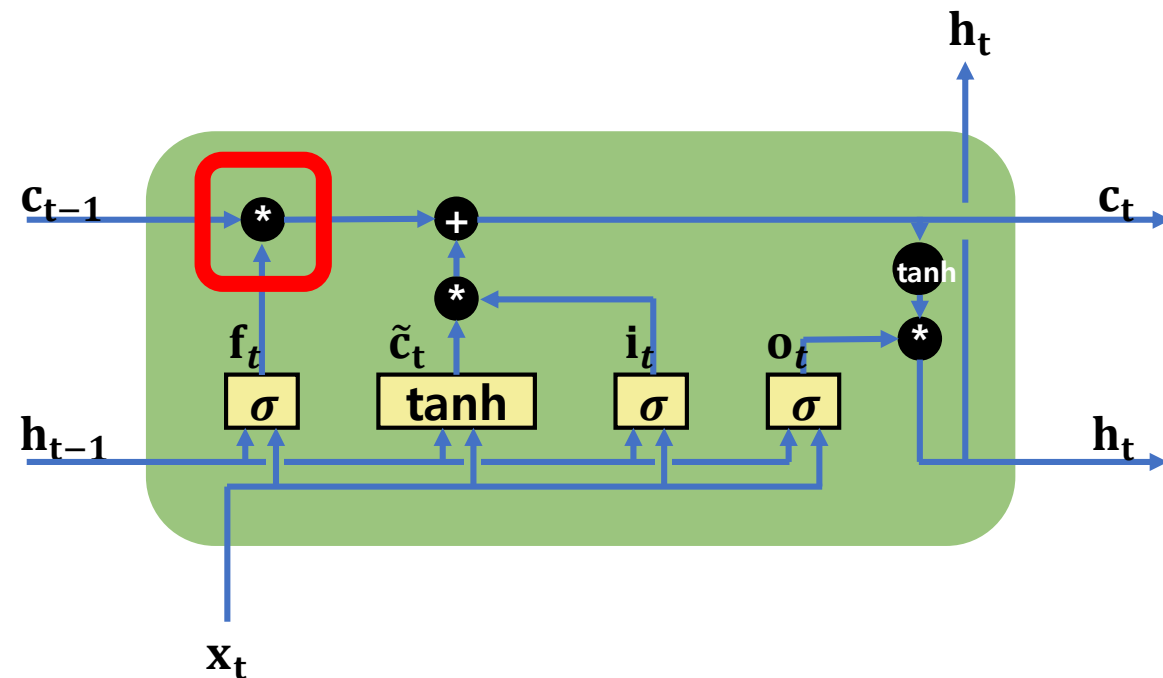
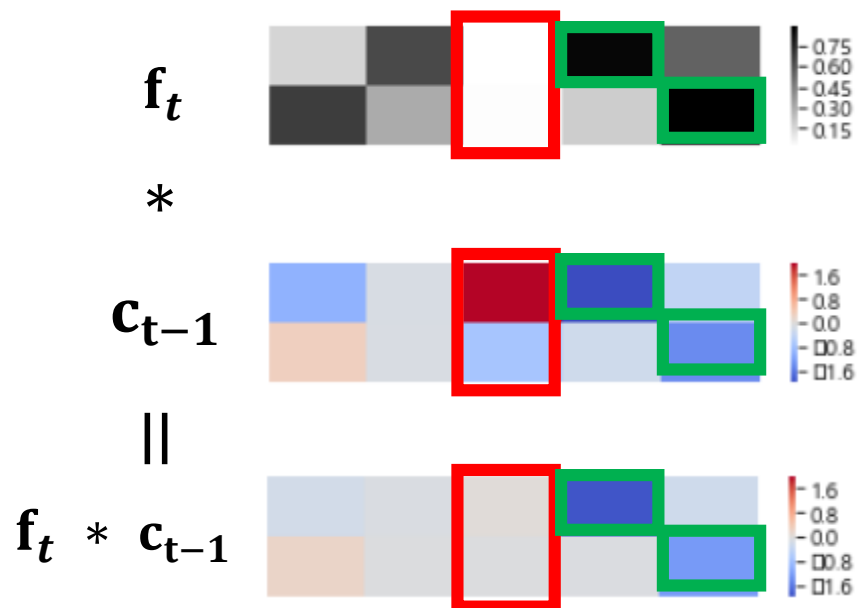


Unit 03 | LSTM

• LSTM의 구조 (1) : Forget Gate

• Forget gate의 연산

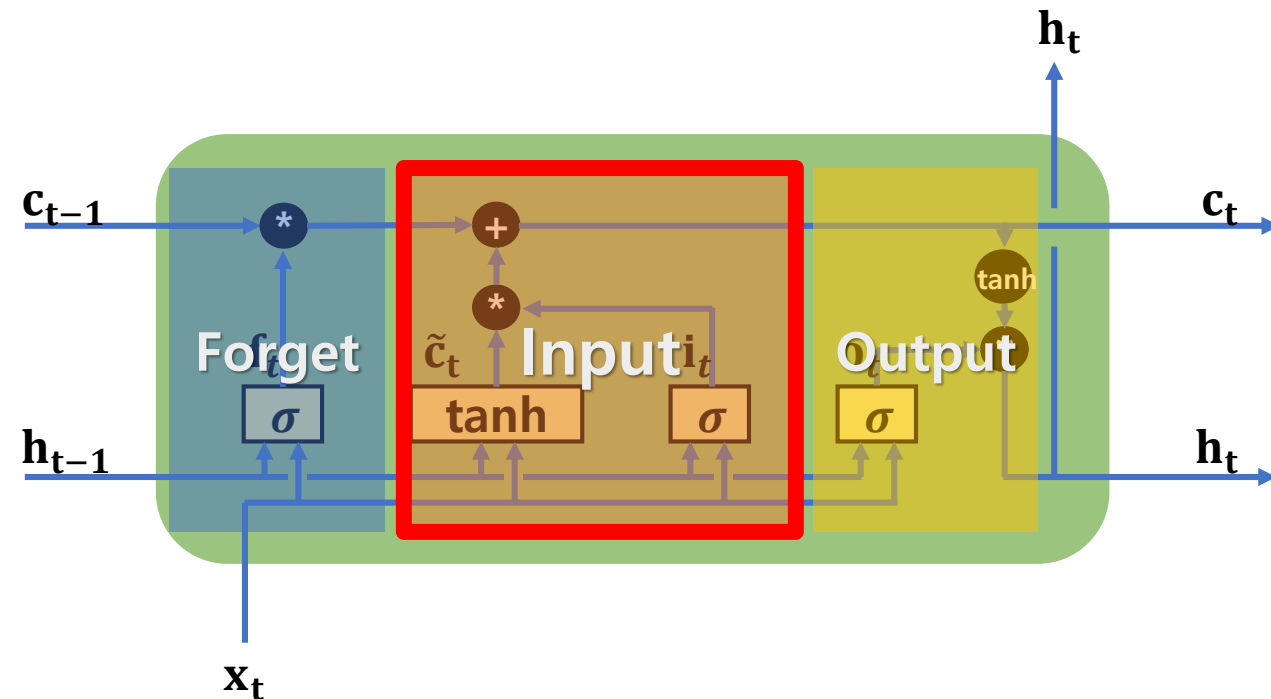
- 계산된 forget gate f_t 를 이전 시점의 장기기억 c_{t-1} 에 요소별로 곱(Hadamard product)하여
- 이전 시점의 각 정보의 망각(또는 유지)를 결정



Unit 03 | LSTM

• LSTM의 구조 (2) : Input Gate

- 현재 입력 정보(h_{t-1} 와 x_t) 중에서
이전 시점의 장기기억(c_{t-1})에 무엇을 추가 시킬지를 결정



Unit 03 | LSTM

• LSTM의 구조 (2) : Input Gate

• Input gate의 연산 전에!!!

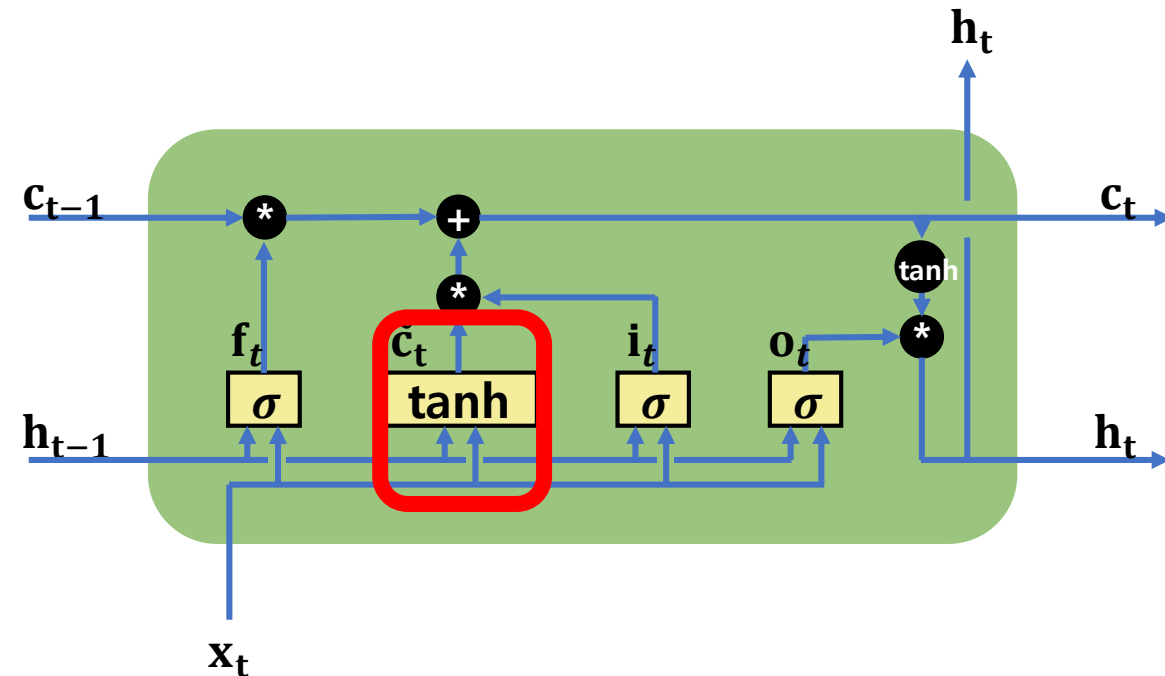
- 현재의 입력 정보를 가공하여

이전 시점의 장기기억에 추가시킬 새로운 정보(기억셀) \tilde{c}_t 를 만든다.

Cf.

\tilde{c}_t : 현재 입력 정보(h_{t-1} 와 x_t)를 가공한
새롭게 추가시킬 정보(새로운 기억 셀)

$$\tilde{c}_t = \tanh \left(x_t W_x^{(c)} + h_{t-1} W_h^{(c)} + b^{(c)} \right)$$



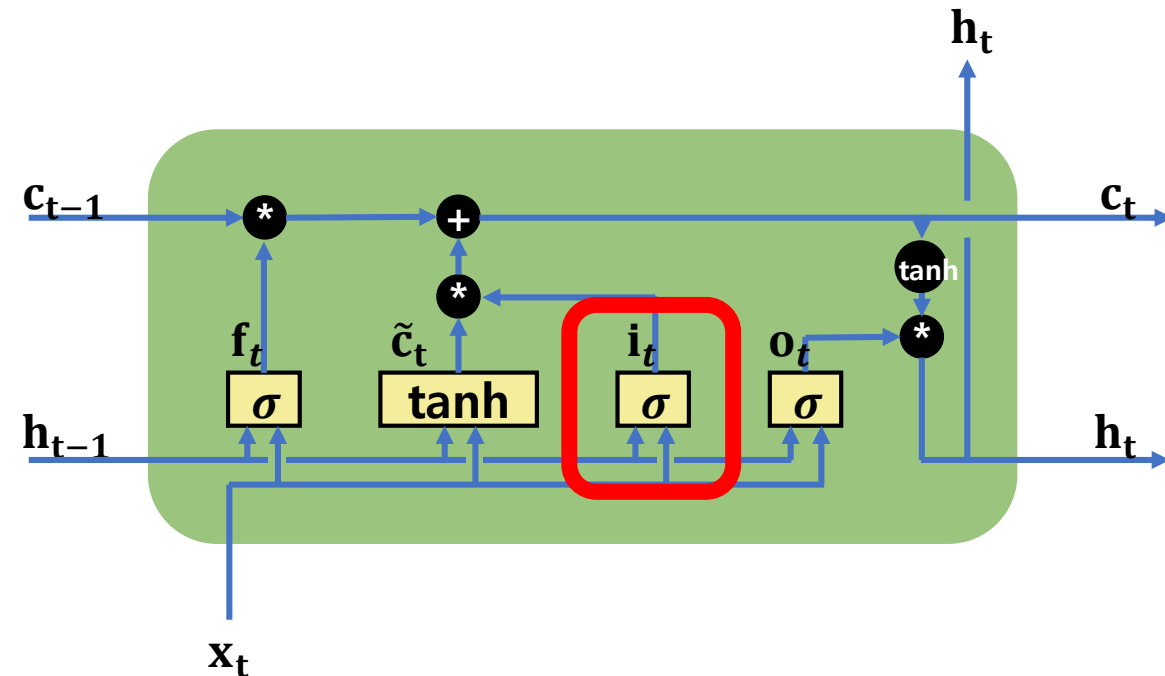
Unit 03 | LSTM

• LSTM의 구조 (2) : Input Gate

• Input gate의 연산 (1)

- 다음 수식을 통해 계산되며,
현재 시점(t)에서 입력 받은 정보(x_t)와 단기 기억 (h_{t-1})을 Affine 변환한 값을
Sigmoid를 적용해 0~1사이의 값으로 변환
- $N \times H$ 크기의 0 ~ 1 사이의 값을 갖는 행렬

$$i_t = \sigma(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)})$$

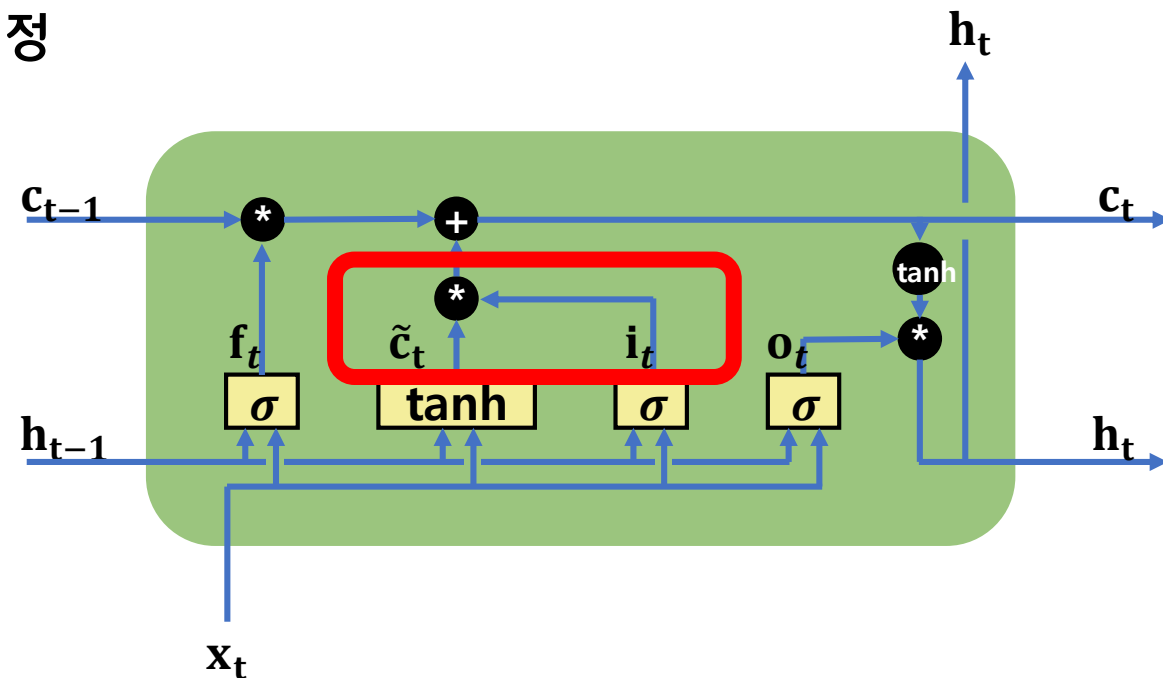
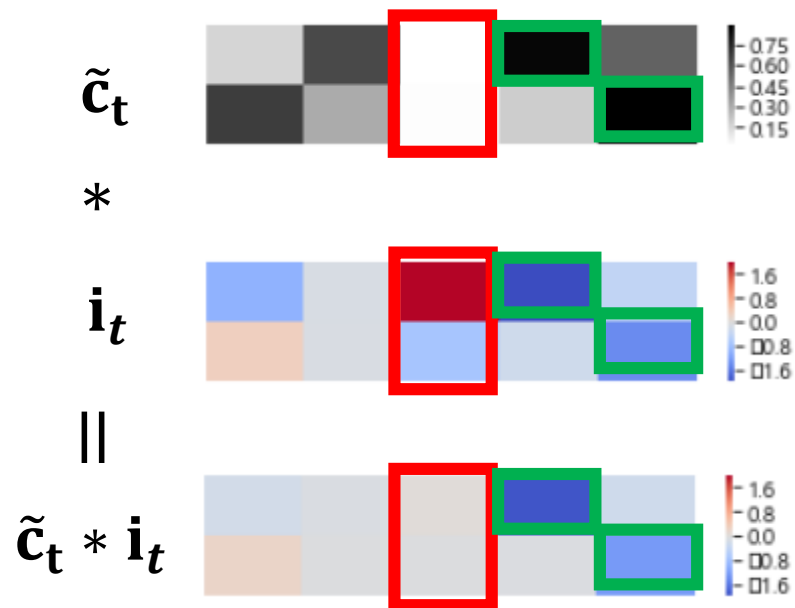


Unit 03 | LSTM

• LSTM의 구조 (2) : Input Gate

• Input gate의 연산 (2)

- 계산된 input gate i_t 를 현재의 입력 정보를 가공한 새로운 기억셀 \tilde{c}_t 에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 장기 기억($f_t * c_{t-1}$)에 더할 것인지를 결정



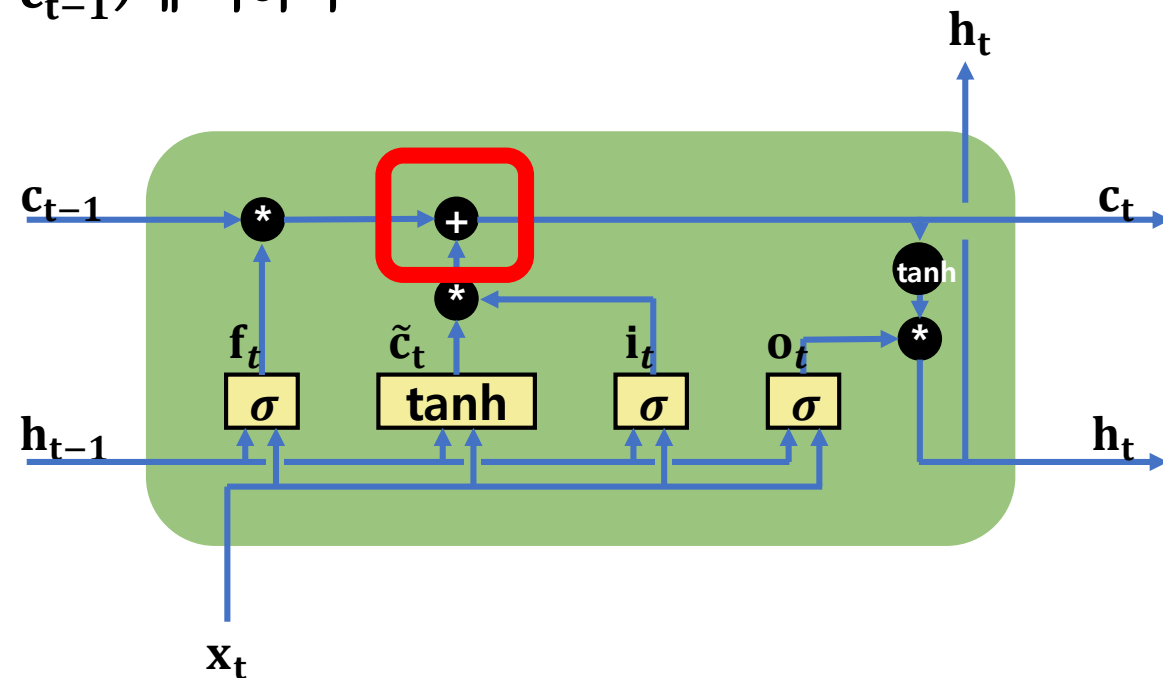
Unit 03 | LSTM

• LSTM의 구조 (2) : Input Gate

• Input gate의 연산 (3)

- Input gate가 적용된 새로운 기억셀($\tilde{c}_t * i_t$)을
forget gate가 적용된 이전 시점의 기억셀($f_t * c_{t-1}$)에 더하여
현재 시점(t)의 기억셀(c_t)를 구함

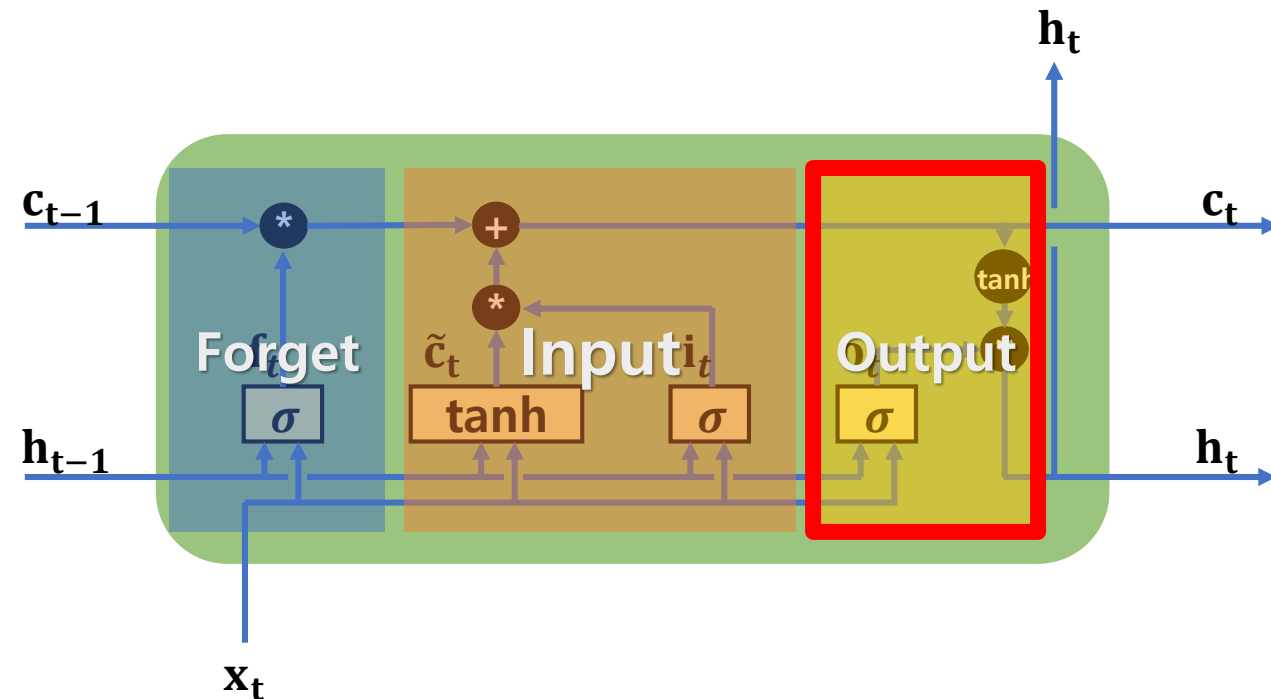
$$c_t = f_t * c_{t-1} + \tilde{c}_t * i_t$$



Unit 03 | LSTM

• LSTM의 구조 (3) : Output Gate

- 현재 입력 정보(h_{t-1} 와 x_t)를 기준으로
현재 시점(t)의 장기기억(c_t)에서 어떤 정보를 읽어와
단기기억(h_t)으로 사용할지를 결정

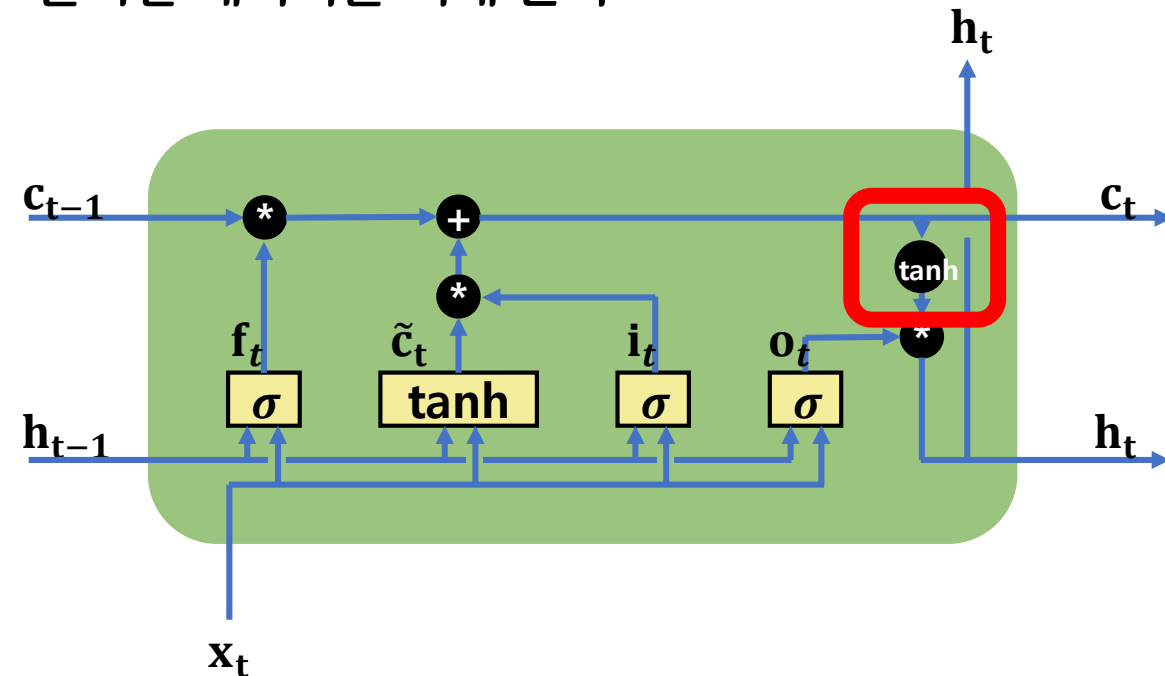


Unit 03 | LSTM

- LSTM의 구조 (3) : Output Gate
 - Output gate의 연산 전에!!!

- 현재의 장기기억(기억셀) c_t 로부터 단기기억으로 출력할 데이터를 꺼내 온다.

- $h_t^{(raw)} = \tanh(c_t)$



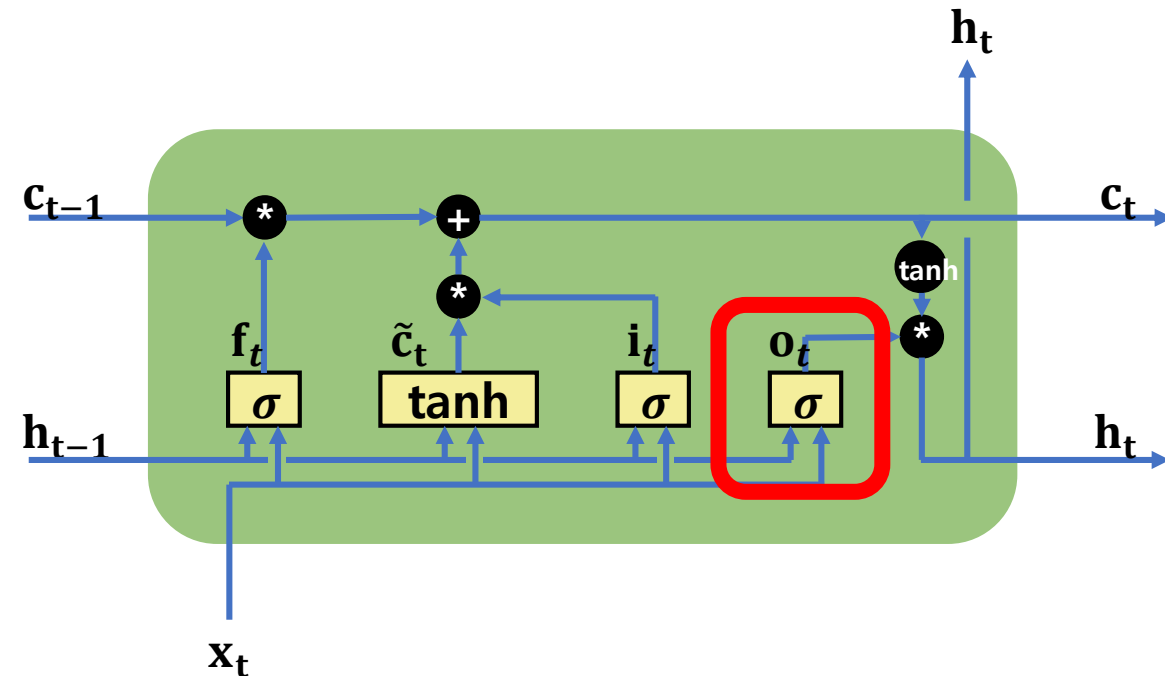
Unit 03 | LSTM

• LSTM의 구조 (3) : Output Gate

• Output gate의 연산 (1)

- 다음 수식을 통해 계산되며,
현재 시점(t)에서 입력 받은 정보(x_t)와 단기 기억 (h_{t-1})을 Affine 변환한 값을
Sigmoid를 적용해 0~1사이의 값으로 변환
- $N \times H$ 크기의 0 ~ 1 사이의 값을 갖는 행렬

$$o_t = \sigma(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)})$$

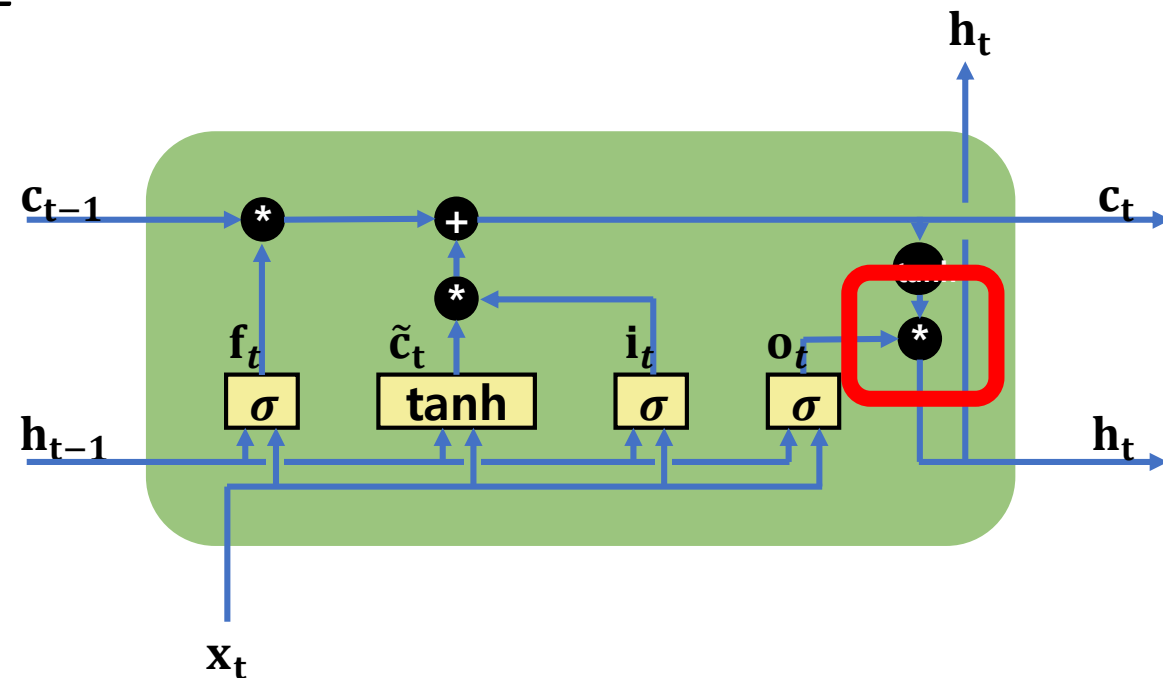
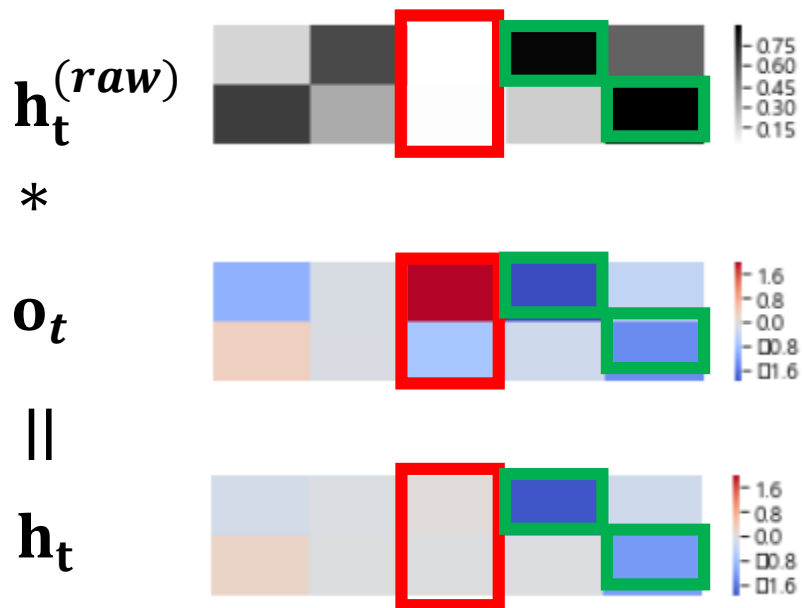


Unit 03 | LSTM

• LSTM의 구조 (3) : Output Gate

• Output gate의 연산 (2)

- 계산된 output gate o_t 를 현재의 장기기억(기억셀)으로부터 꺼내온 $h_t^{(raw)}$ 에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 단기기억(h_t)으로 사용할지를 결정

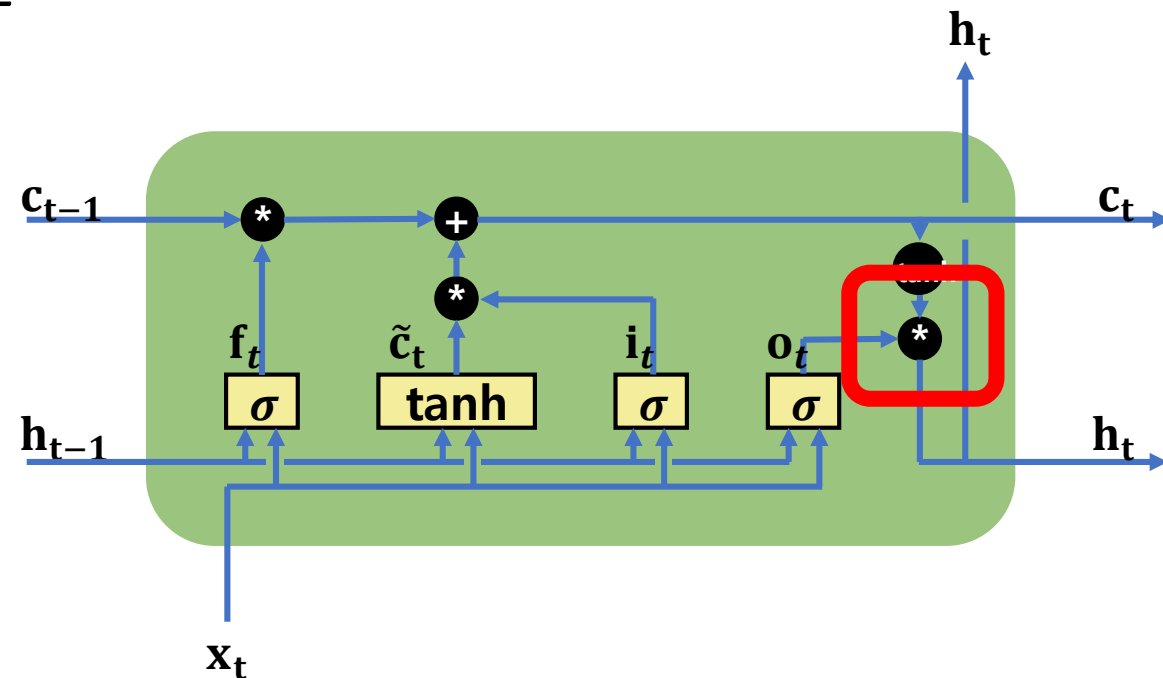
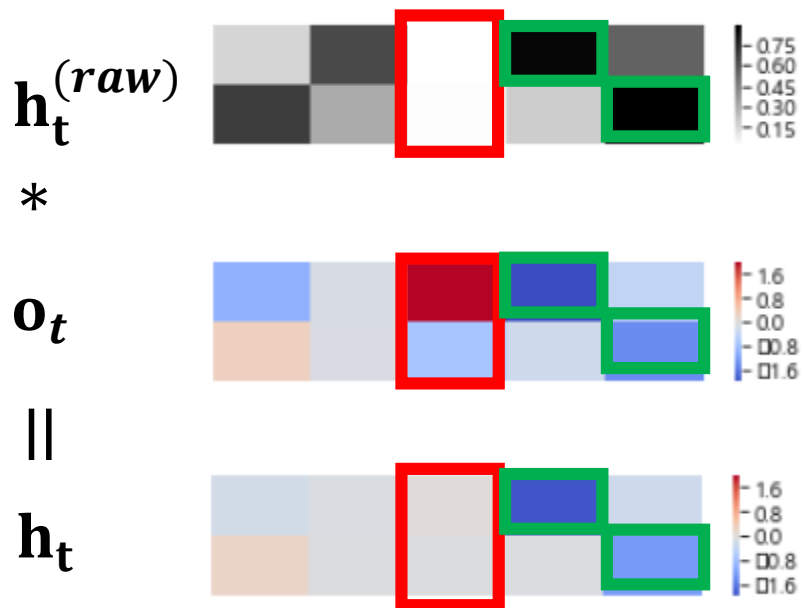


Unit 03 | LSTM

• LSTM의 구조 (3) : Output Gate

• Output gate의 연산 (2)

- 계산된 output gate o_t 를 현재의 장기기억(기억셀)으로부터 꺼내온 $h_t^{(raw)}$ 에 요소별로 곱(Hadamard product)하여 각 정보를 얼마나 단기기억(h_t)으로 사용할지를 결정



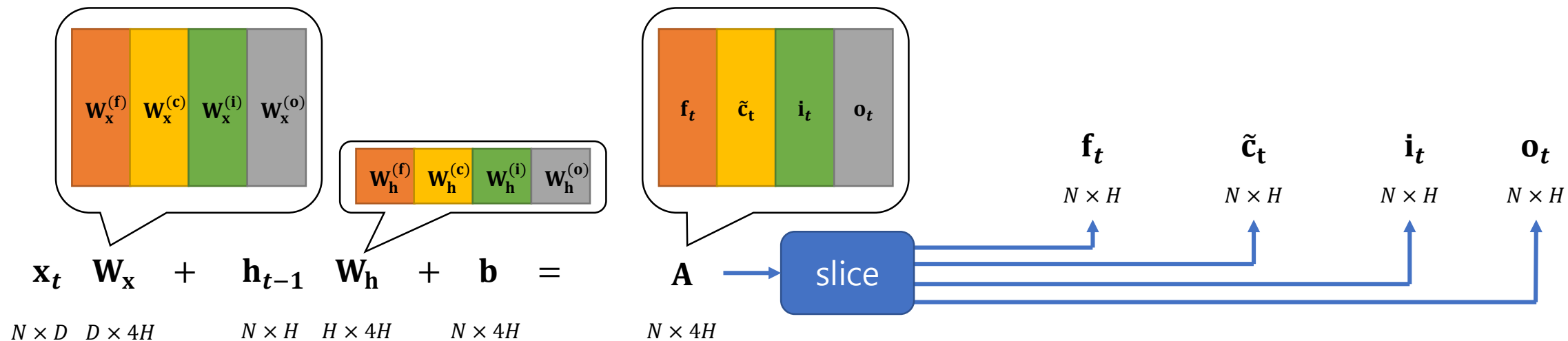
Unit 03 | LSTM

- LSTM Implementation

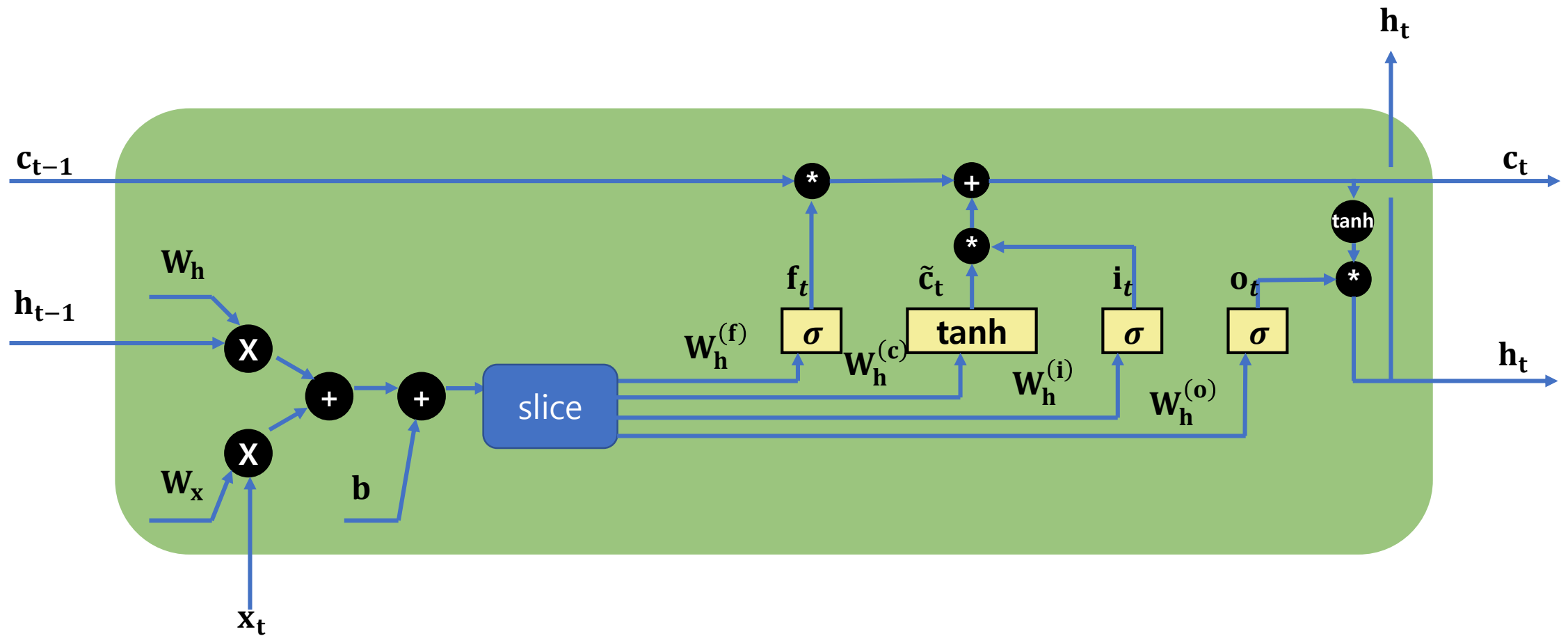
실습

02_LSTM&LSTMLM.ipynb

Unit 03 | LSTM



Unit 03 | LSTM



Unit 04 | Advanced RNN

Advance RNN

Unit 04 | Advanced RNN

NLP Task with RNN

Bidirectional-RNN

Multi-layer RNN

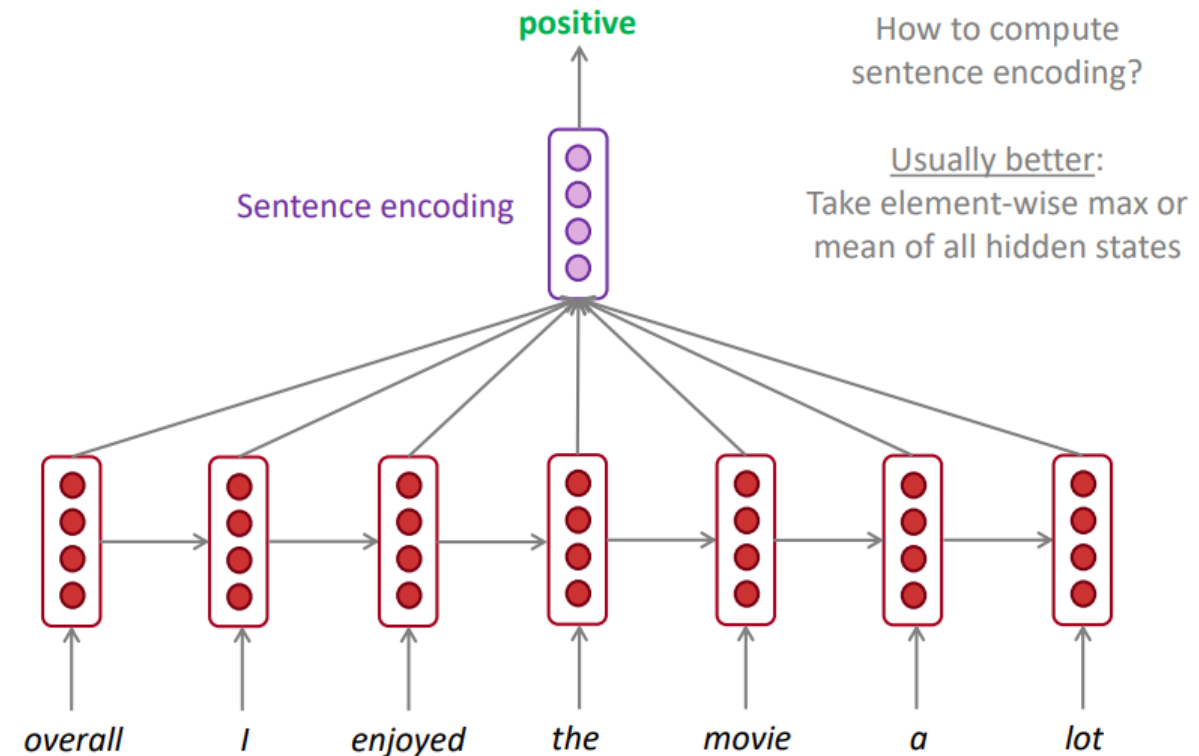
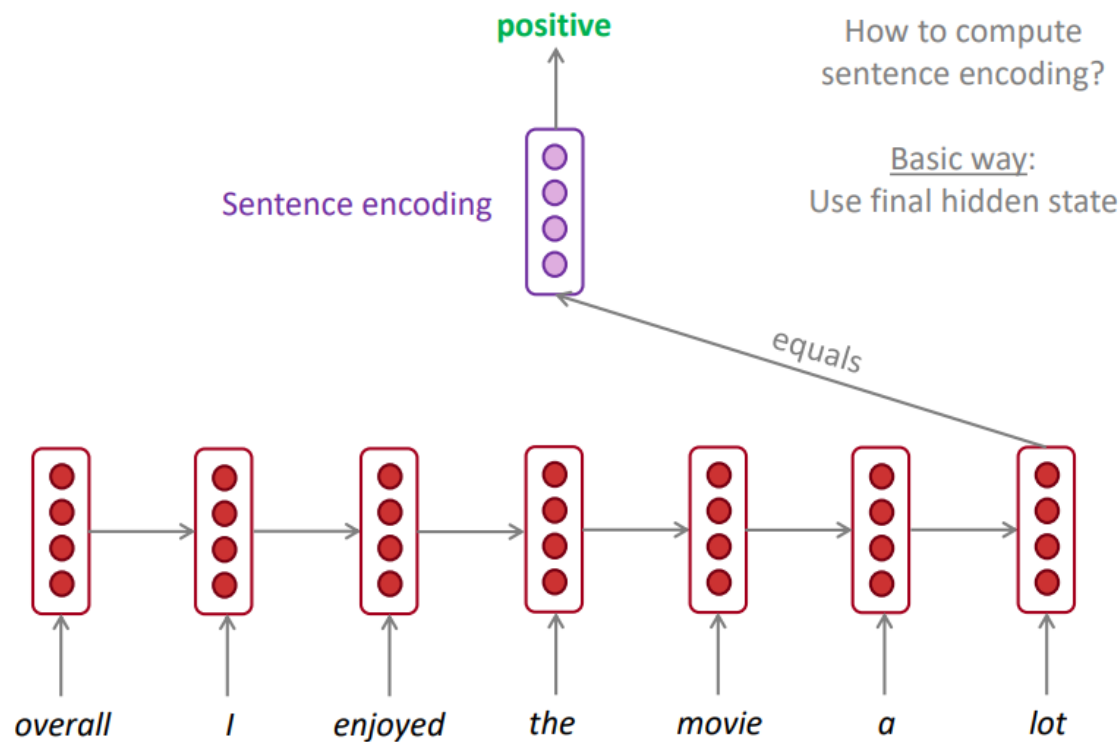
Seq2seq

Unit 04 | Advanced RNN

• NLP Task with RNN : Sentence Classification

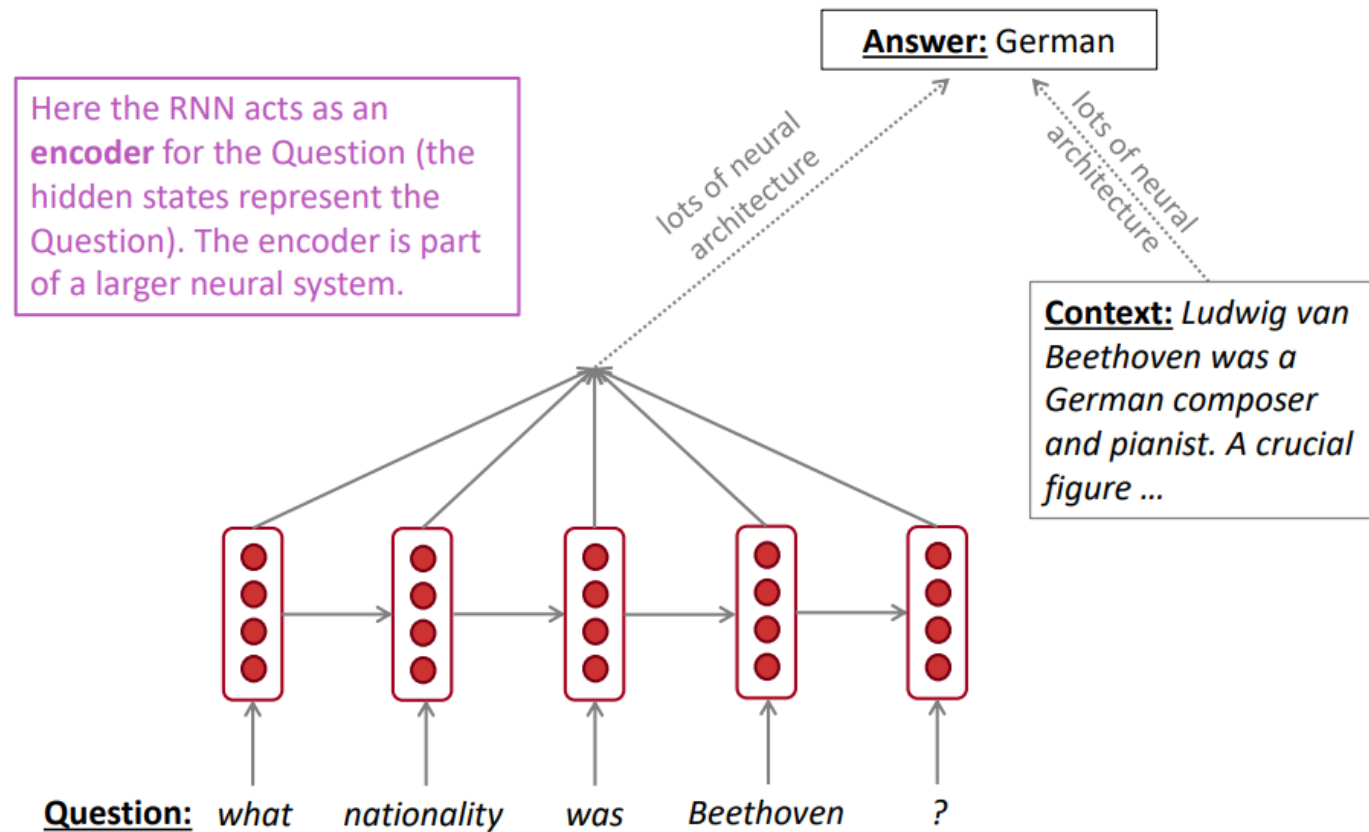
e.g. [sentiment classification](#)

e.g. [sentiment classification](#)



Unit 04 | Advanced RNN

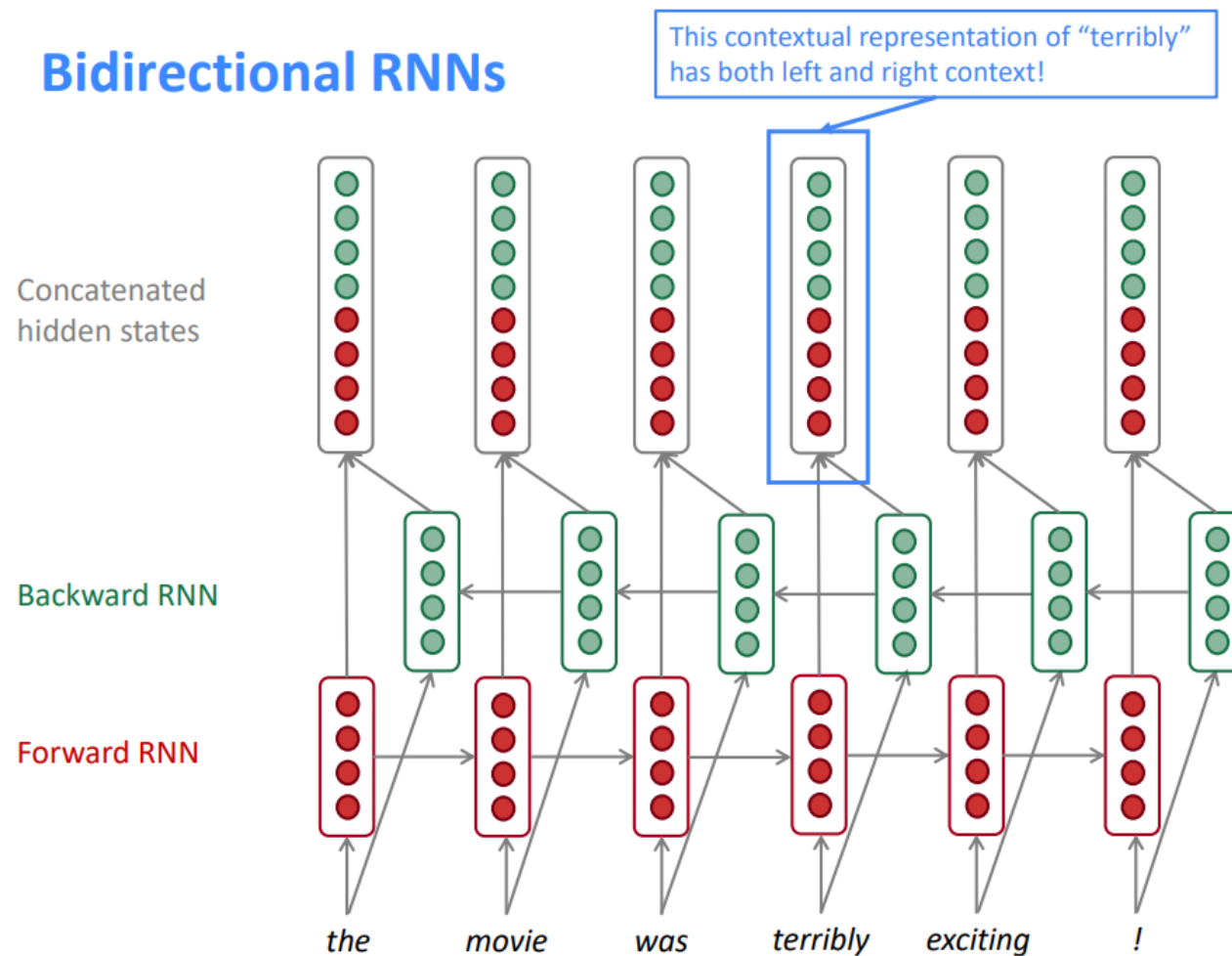
- NLP Task with RNN : QA, Machine Translation, ...



Unit 04 | Advanced RNN

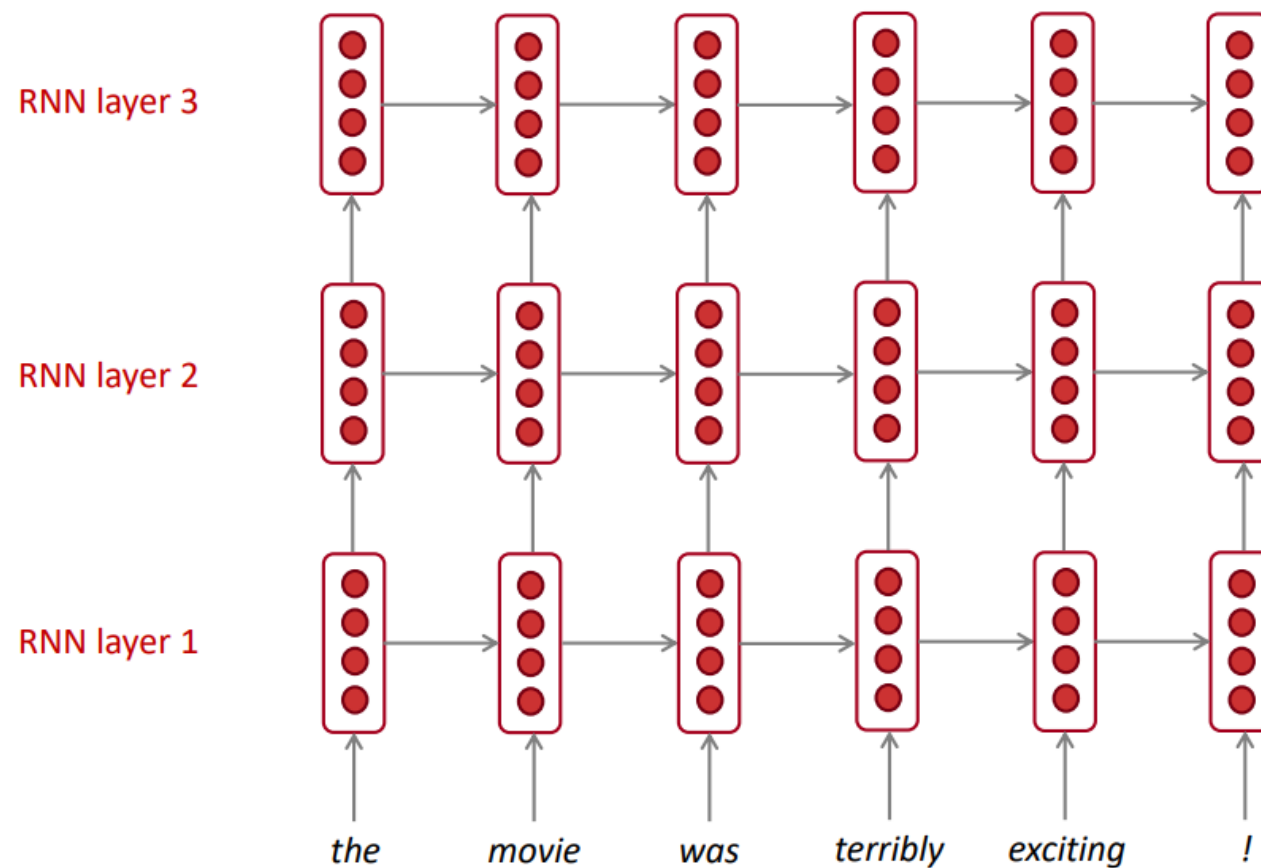
• Bi-Directional RNN

Bidirectional RNNs



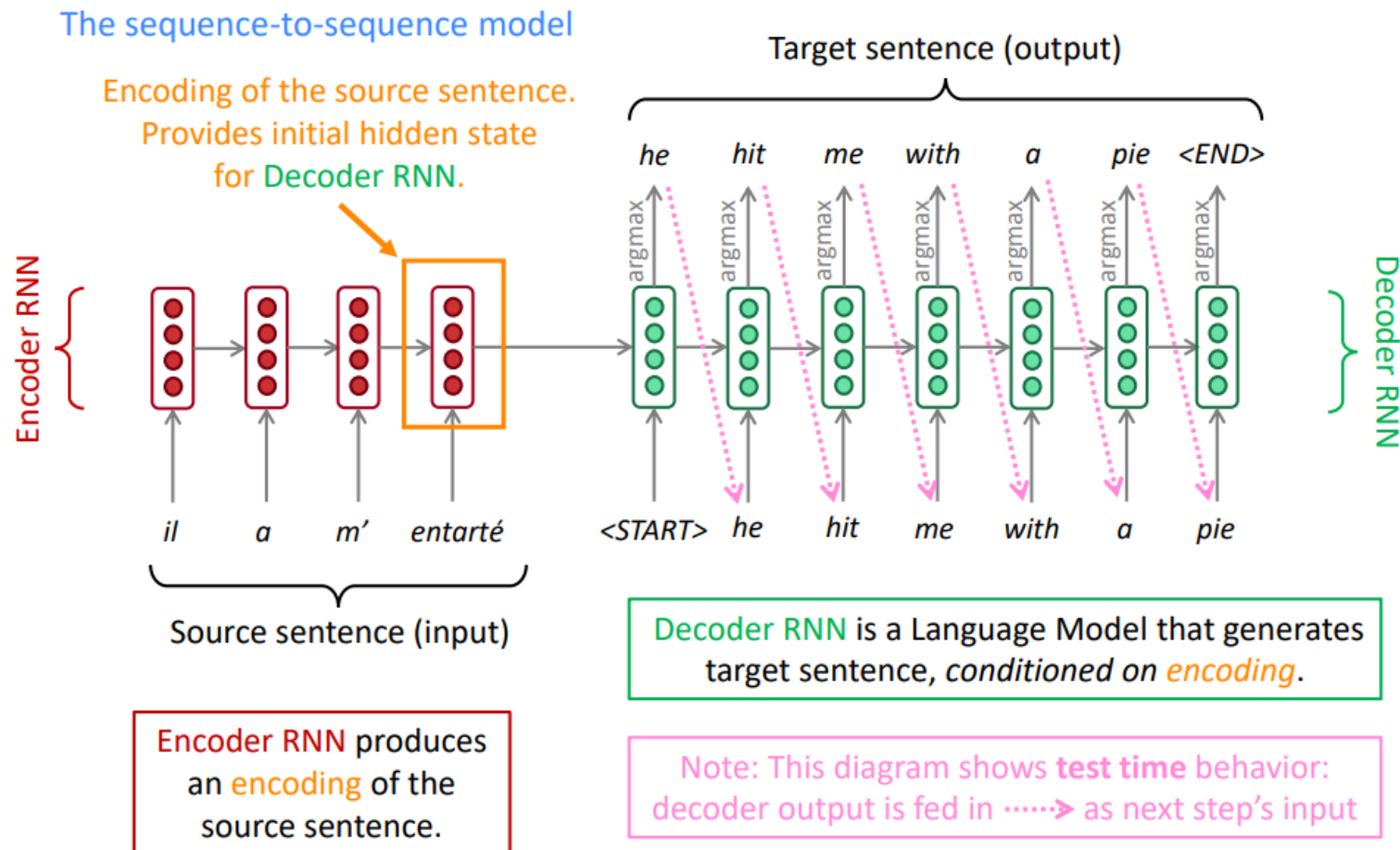
Unit 04 | Advanced RNN

• Multi-layer RNN



Unit 04 | Advanced RNN

• Seq2seq (Encoder-Decoder Model)



References

- Cs224n, Stanford University
- Cs231n, Stanford University
- 밑바닥부터 시작하는 딥러닝2, 사이토 고키

Q & A

들어주셔서 감사합니다.