# RNN

BOAZ 분석 20기 BASE SESSION

Week 5. RNN

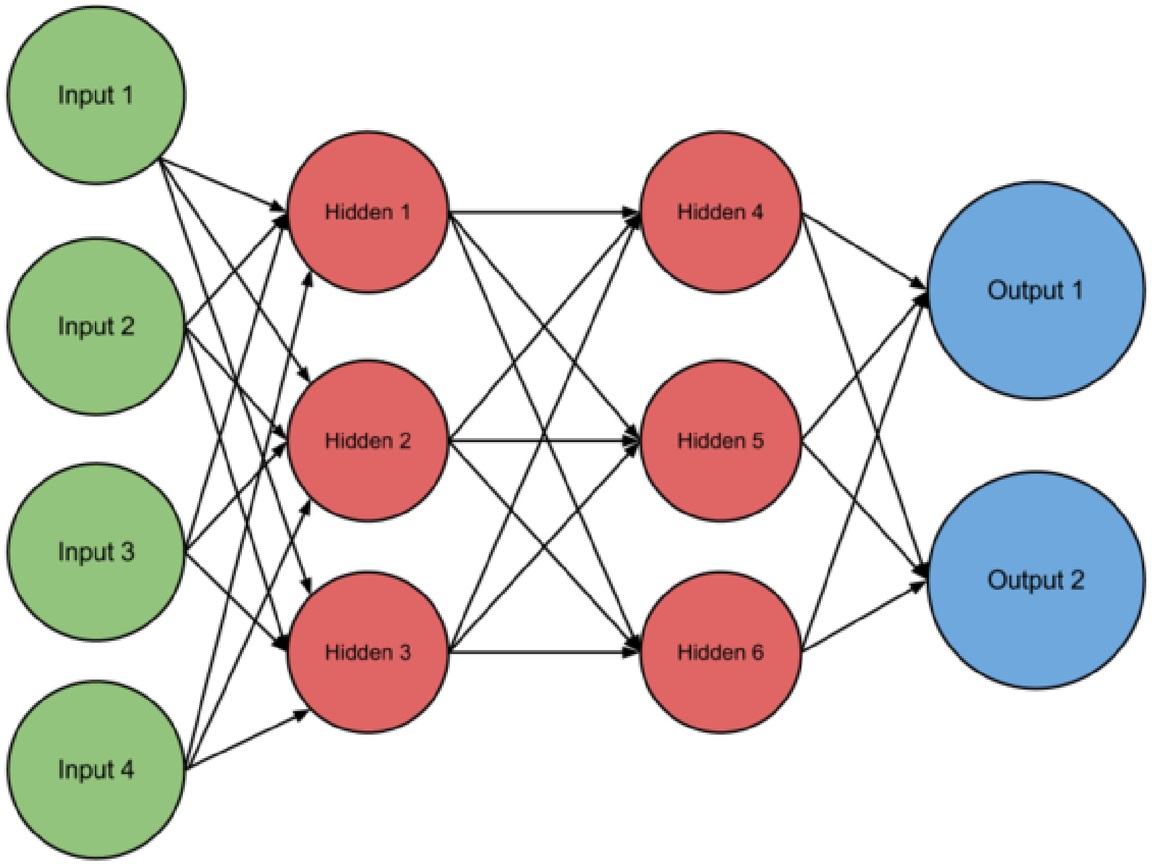
분석 19기 정은진

# TABLE OF CONTENTS

# 01 RNN이란

## RNN의 발전 배경

- 순방향 신경망(Feed Forward Neural Network)의 한계점 존재
- 순방향 신경망의 경우, 오직 입력층에서 출력층 방향으로만 연산이 전개됨. 따라서 시간 및 공간 변화의 흐름에 따른 시계열 데이터를 학습하기 어려움
- 순방향 신경망의 한계점을 극복하기 위해, 은닉층 내에 순환 구조를 도입한 모델이 바로 RNN



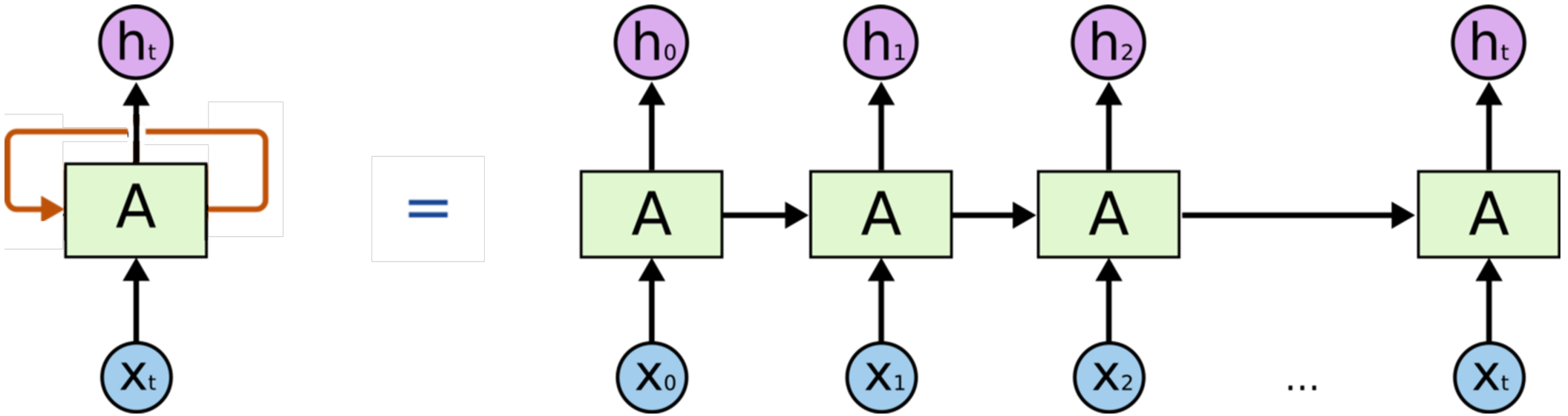
## RNN (Recurrent Neural Network)

- 은닉층 내부의 노드(Node, Cell)들이 방향을 가진 엣지(Edge)로 연결되어 있음,

  즉 RNN은, 여러 셀이 연결된 은닉층들이 순환 구조(Directed Cycle)를 이루고 있는 시퀀스(Sequence) 모델

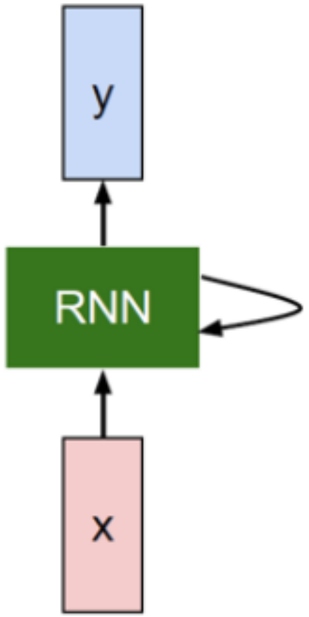
- 내부의 순환 구조를 활용하여 순차적인 데이터(Sequential Data)를 처리하는 데 유용하게 활용됨 (e.g. 음성, 문자 등)
- 순환 구조를 통해 이전에 저장된 과거의 정보를 가지고 현재의 정보를 처리할 수 있으며,

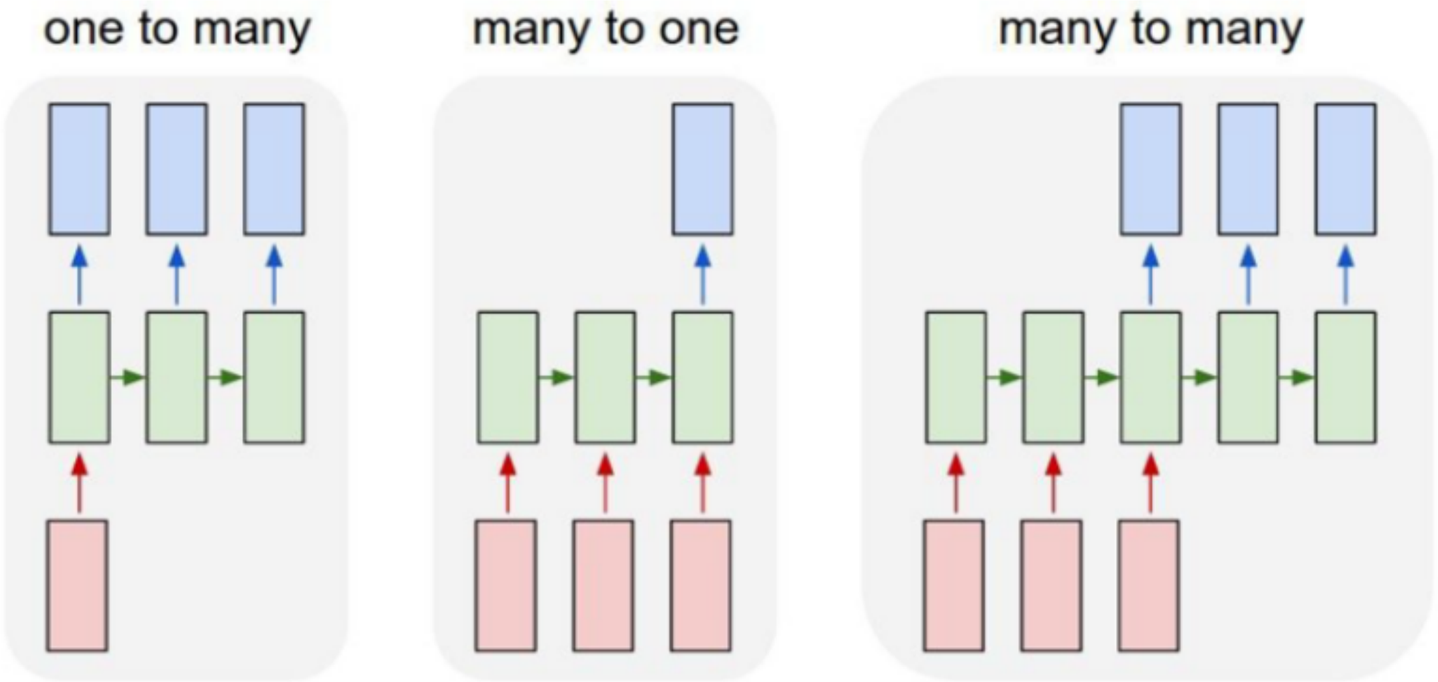
  데이터가 순환되기 때문에 정보가 끊임없이 갱신될 수 있는 구조를 지님




$x_n$: 입력값 $h_n$: 출력값 $A$:뉴럴네트워크 덩어리

**RNN의 장점** => 입력층과 출력층 시퀀스 길이를 다르게 설계 가능, 다양한 용도로 사용할 수 있는 유연한 구조를 지님




- ***일대다(one to many)***

  이미지 캡셔닝: 하나의 이미지 입력에 대한 사진의 제목 출력

- ***다대일(many to one)***

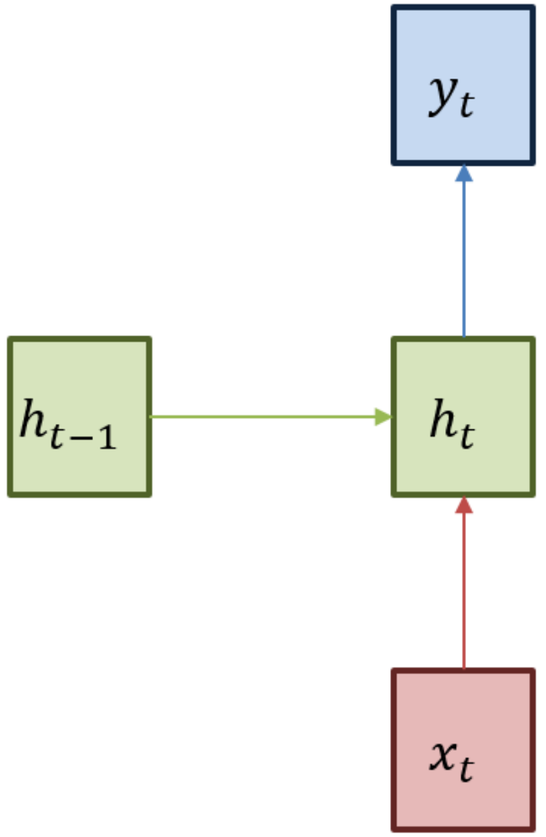
  감성분류: 입력된 문서의 긍부정을 판별 | 스팸메일 분류: 입력된 메일이 정상인지 스팸메일인지 여부를 출력

- ***다대다(many to many)***

  챗봇 또는 번역기: 시퀀스 입력(문자열)에 대한 시퀀스 출력 | 이외에도 품사 태깅, 개체명 인식 등의 예시가 존재

## RNN의 기본 구조

- 현재 상태의 은닉층 $h_t$가 직전 시점의 은닉층인 $h_{t-1}$ 을 받아 갱신됨
- 현재 상태의 결과값인 $y_t$는 $h_t$를 전달받아 갱신됨
- 은닉층 $h_t$의 활성화 함수는 비선형 함수인 하이퍼볼릭탄젠트(*tanh*)를 사용



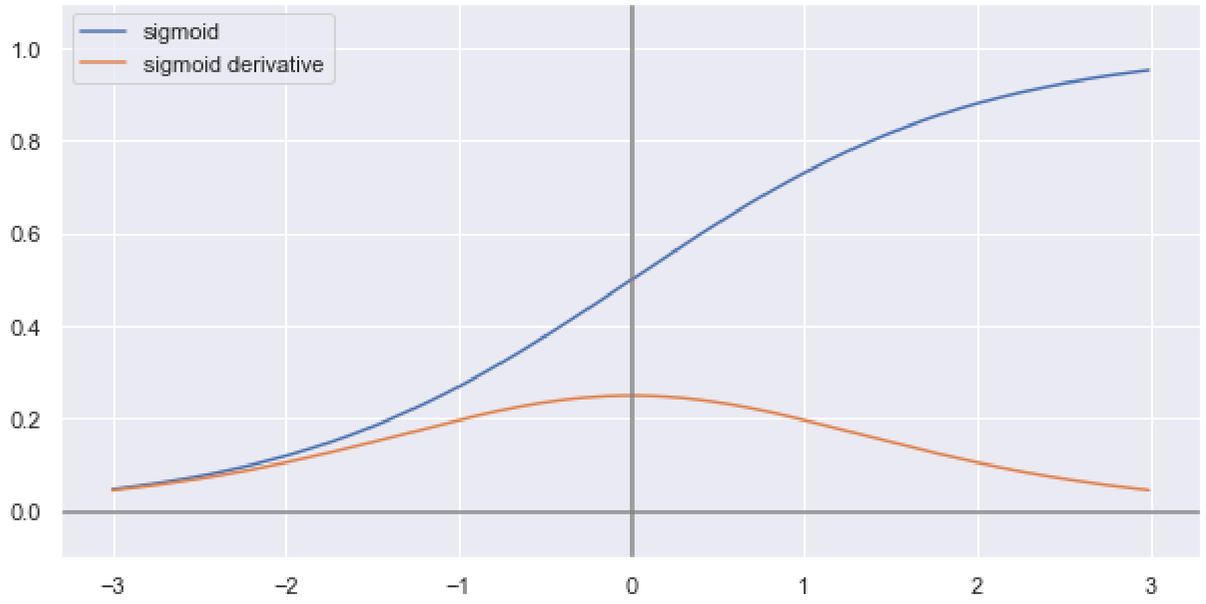

$$y_t = W_{hy}h_t + b_y$$

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

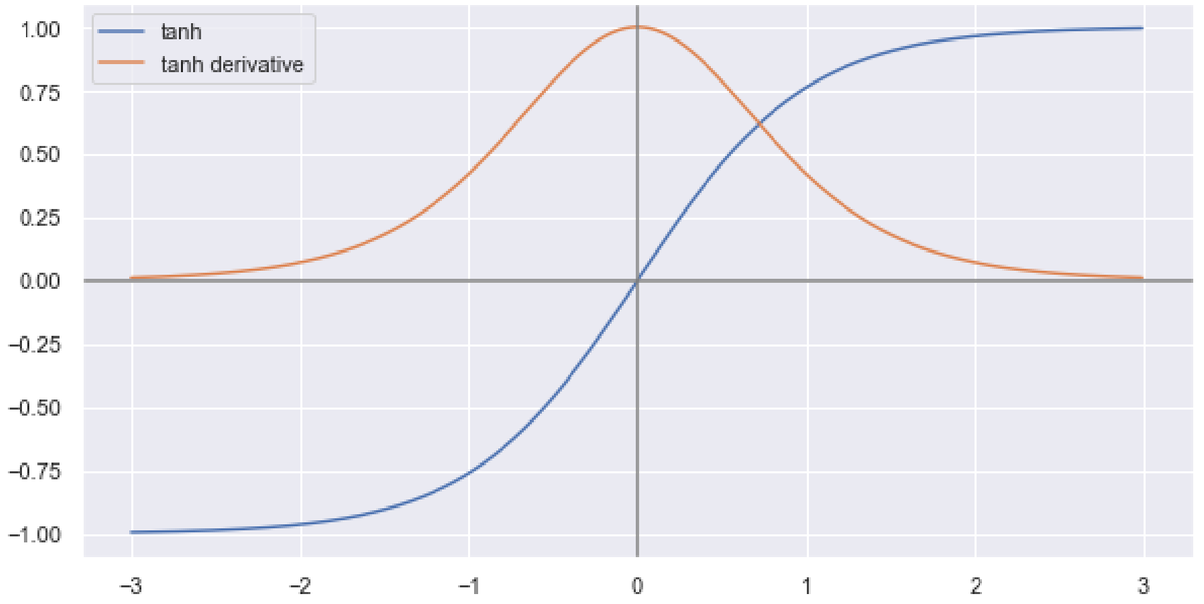
## 비선형 함수를 활성화 함수로 사용하는 이유

- 기울기 소실 문제(Vanishing Gradient Problem)를 예방하기 위함
- RNN 학습 과정에서 미분을 수행할 때, sigmoid를 활성화 함수로 활용하게 되면
  곱셈이 거듭되면서 기울기 소실 문제가 발생할 가능성이 있음
- 그러나 tanh의 경우, sigmoid에 비해 상대적으로 미분 최대값이 큰 것을 알 수 있음
  따라서 기울기 소실을 예방하기 위해서는 tanh를 활성화 함수로 사용하는 것이 유리함

sigmoid 함수의 미분 분포




tanh 함수의 미분 분포



02

# RNN의 학습 과정

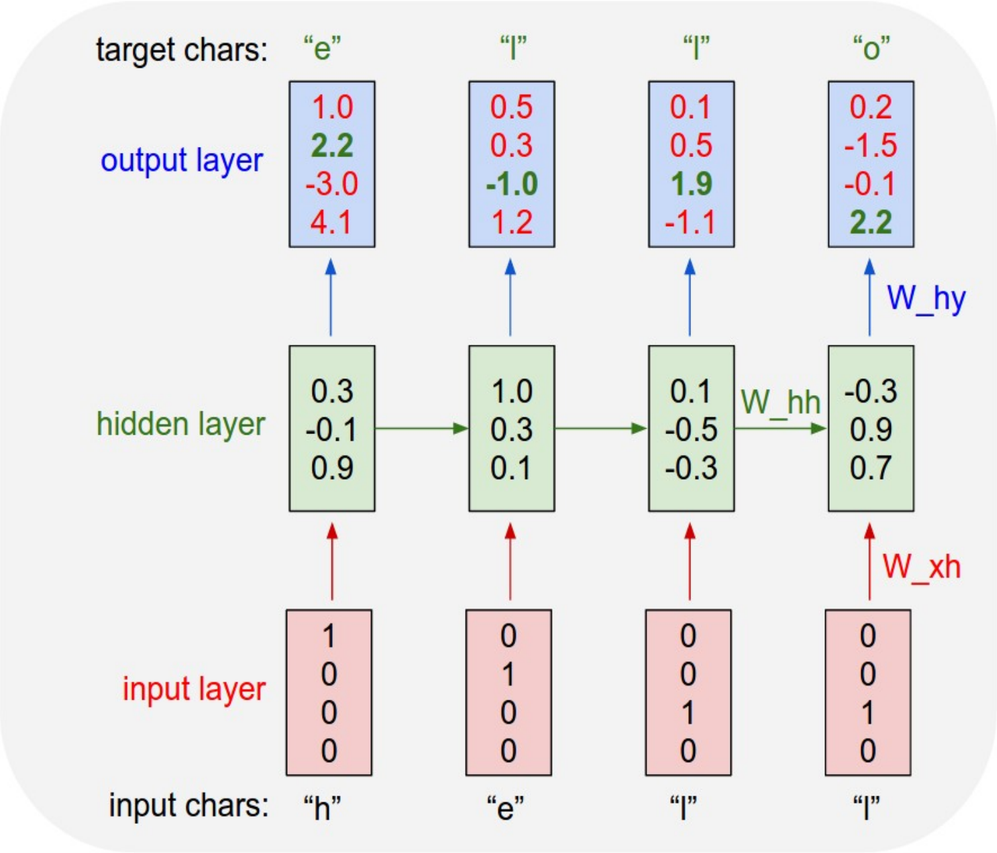
# Character-Level Languane Model을 통해 이해하는 RNN의 기본 구조

- **순전파(Forward Propagation)**
  직전 단계 은닉층의 연산 결과를 반영하여, 다음 순서에 올 알파벳을 예측

- **역전파(Backpropagation)**
  정답값과 예측값의 오차를 줄이기 위해 parameter를 갱신

- **RNN 학습과정에서 갱신되는 Parameter**
  $W_{xh}$ : 입력값에서 은닉층으로의 연산을 수행할 때 곱해지는 가중치
  $W_{hh}$ : 이전 은닉층에서 다음 은닉층으로의 연산을 수행할 때 곱해지는 가중치
  $W_{yh}$ : 은닉층에서 결과값을 내보낼 때 곱해지는 가중치

- **Shared Weights**
  위 Parameter들은 모든 시점의 은닉층에서 동일하게 적용됨

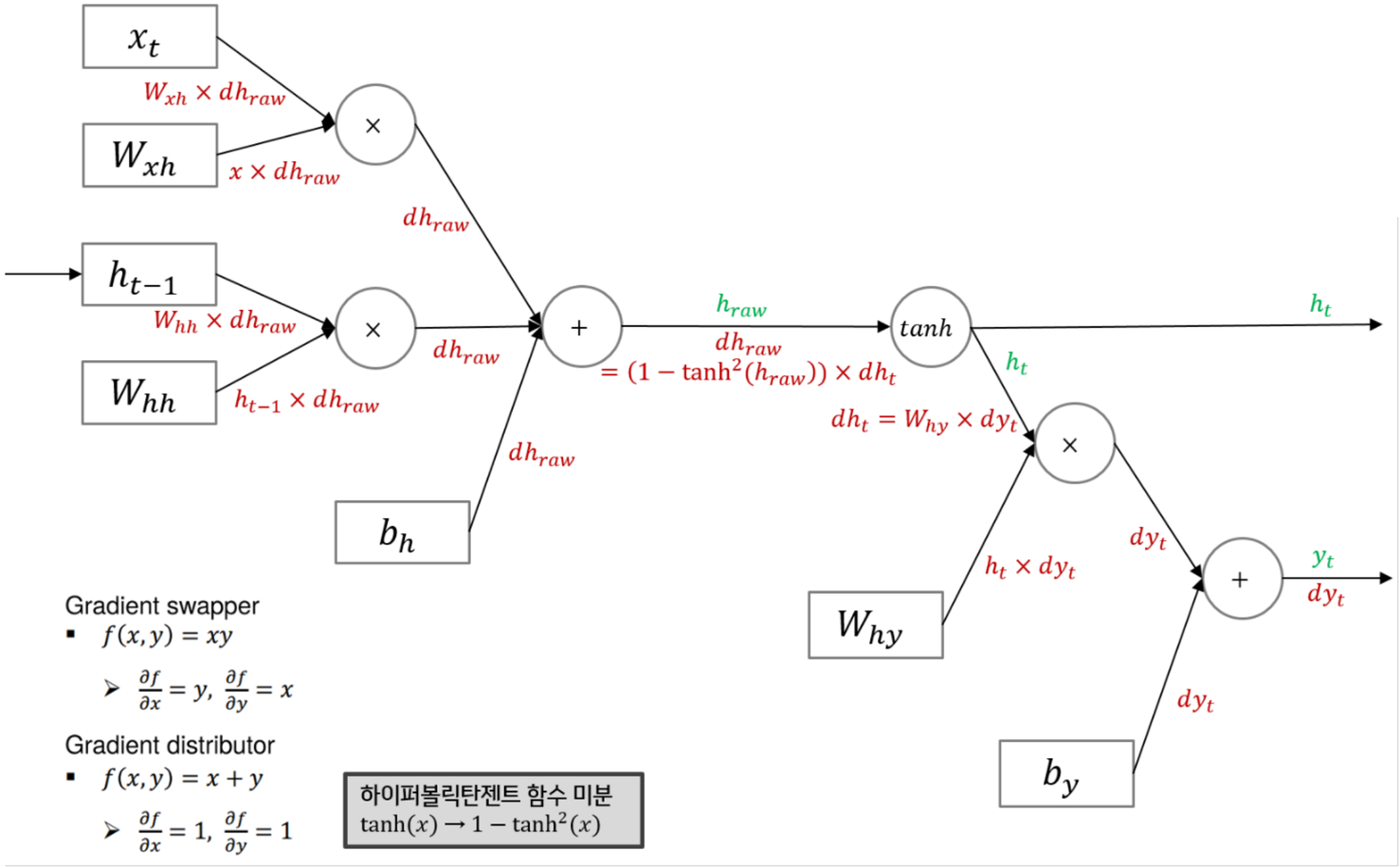



Vocabulary: [h, e, l, o]
각 알파벳은 One-Hot Encoding 과정을 거쳐 입력값으로 주어짐
Example Training Sequence: "**hello**"

## BPTT (Backpropagation Through Time)

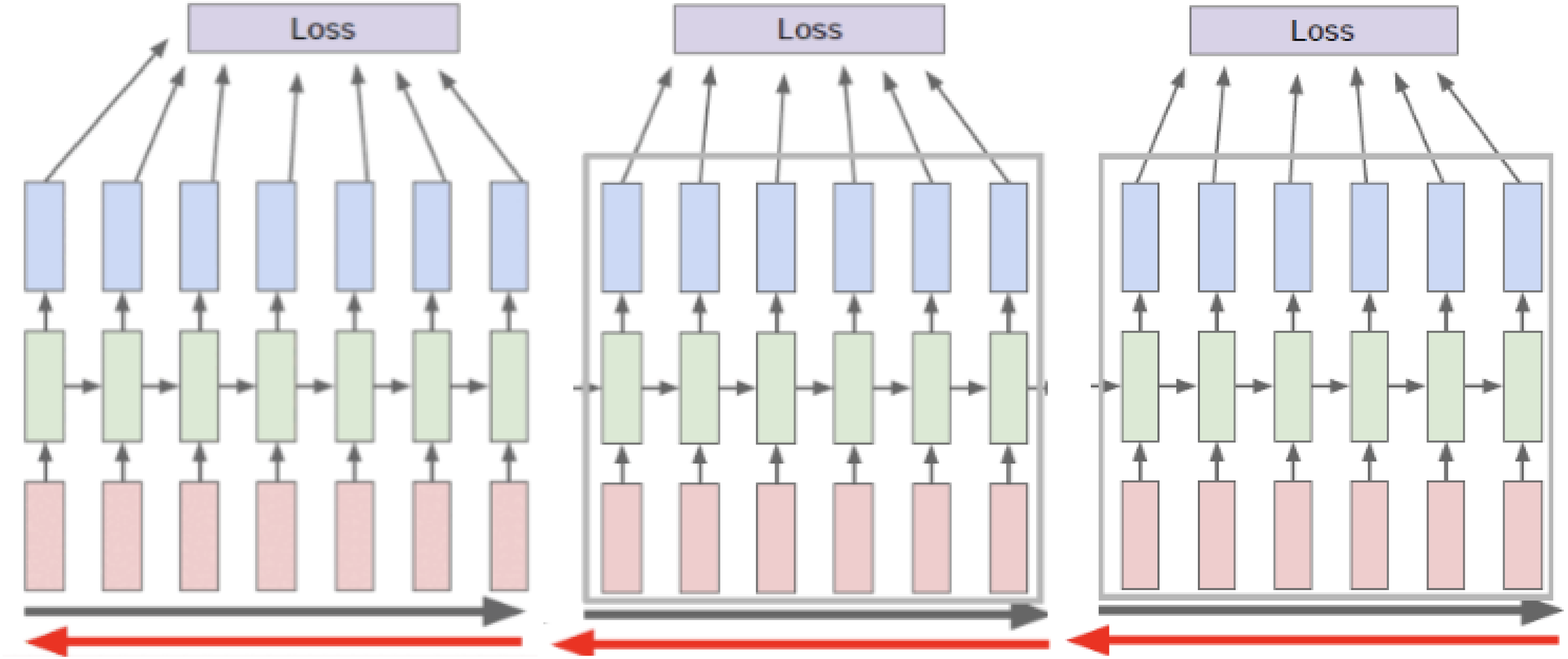
*"Forward through entire sequence to compute loss,*
*then backward through entire sequence to compute gradient."*

- RNN은 매 시퀀스마다 출력값이 존재하는 형태
- 시간 방향으로 펼쳐진 신경망에서 역전파를 수행하며 각 출력값들의 loss 값을 계산하고, 이를 통해 최종 loss 값을 얻는 과정



## Truncated Backpropagation Through Time

- RNN에서의 순전파, 역전파 과정은 전체 시퀀스가 끝날 때까지 계속해서 출력값이 생성됨
  이 경우 시퀀스가 매우 길다면 메모리 부족 이슈가 발생할 수 있음
- 이 문제를 해결하기 위해, 전체 시퀀스를 일정한 단위로 자른 뒤 학습을 수행하는 **Truncated BPTT**가 활용됨



# 03 예시를 통한 RNN의 학습과정 이해

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

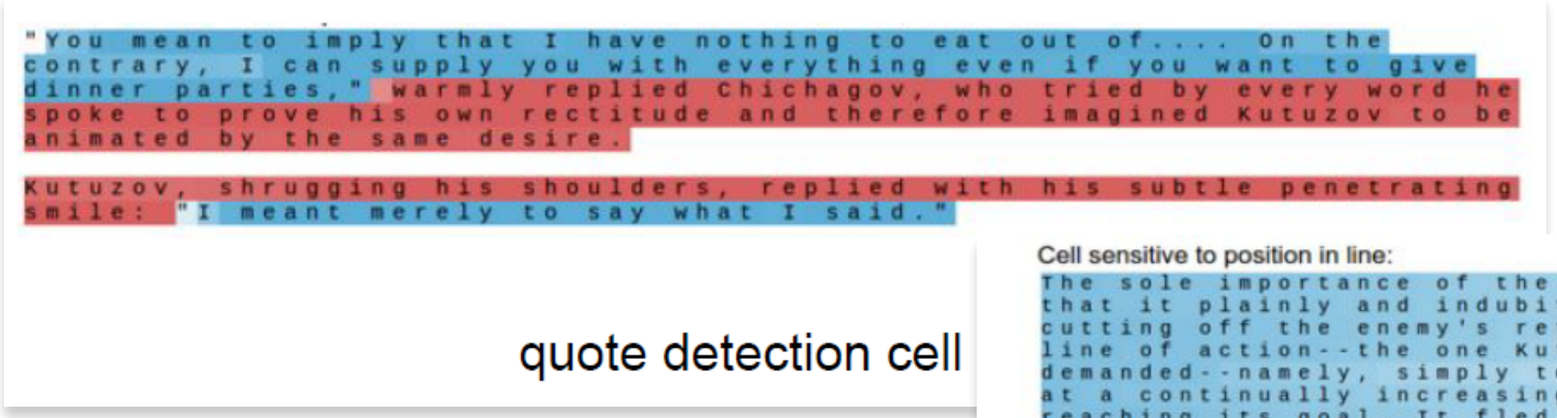
```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

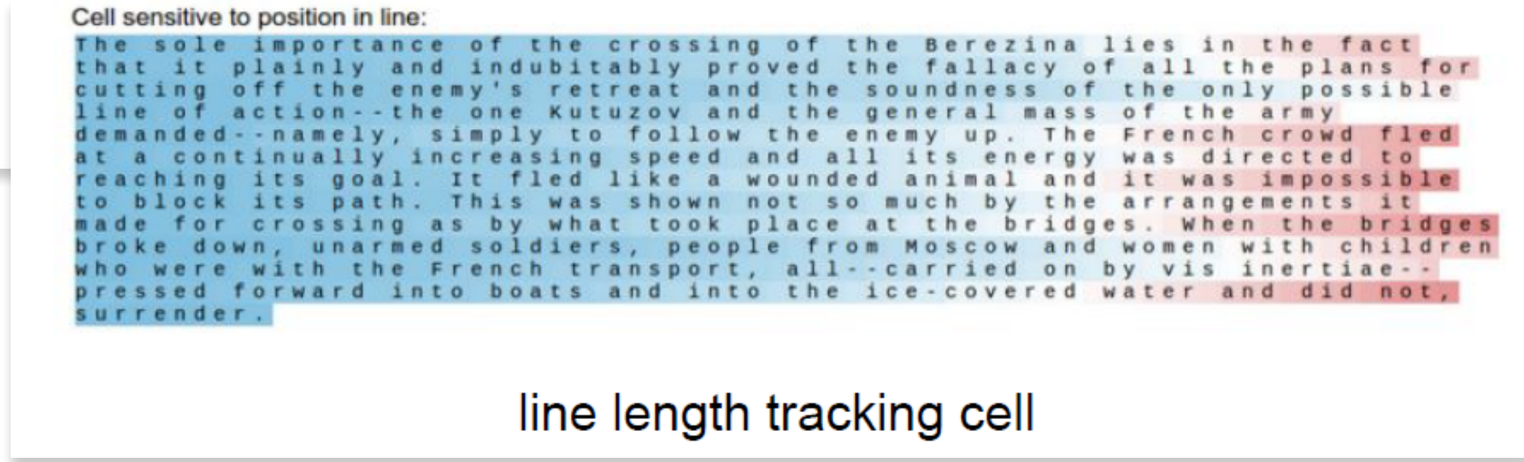
## RNN Language Model

- RNN 모델에게 특정 문장 시퀀스의 다음에 올 문자를 예측할 것을 요청
- 모델은 학습 데이터로 주어진 셰익스피어의 소설 문장 속에 숨겨진 구조(Latent Structure)를 스스로 학습하여, 셰익스피어의 문체와 유사한 의미있는 문장을 생성해 또 하나의 소설을 만들어 냄

## 해석 가능한 셀(Interpretable Cells)의 발견

- 서로 다른 은닉층들이 학습된 문장 구조 중 어떤 것을 중요하게 보고 있는 지 알아보기 위해,
  특정한 은닉 셀 하나를 추출, 어떤 값이 들어있는 지 살펴 봄
- 그 결과 의미있는 문장 패턴이 드러나는 셀들이 발견됨
  대표적인 예로 따옴표를 찾아내는 벡터, 줄바꿈을 위해 현재 줄의 단어 개수를 세는 벡터 등이 발견되었으며,
  이를 통해 RNN이 순환구조를 통해 유의미한 학습을 진행하고 있음을 알 수 있음



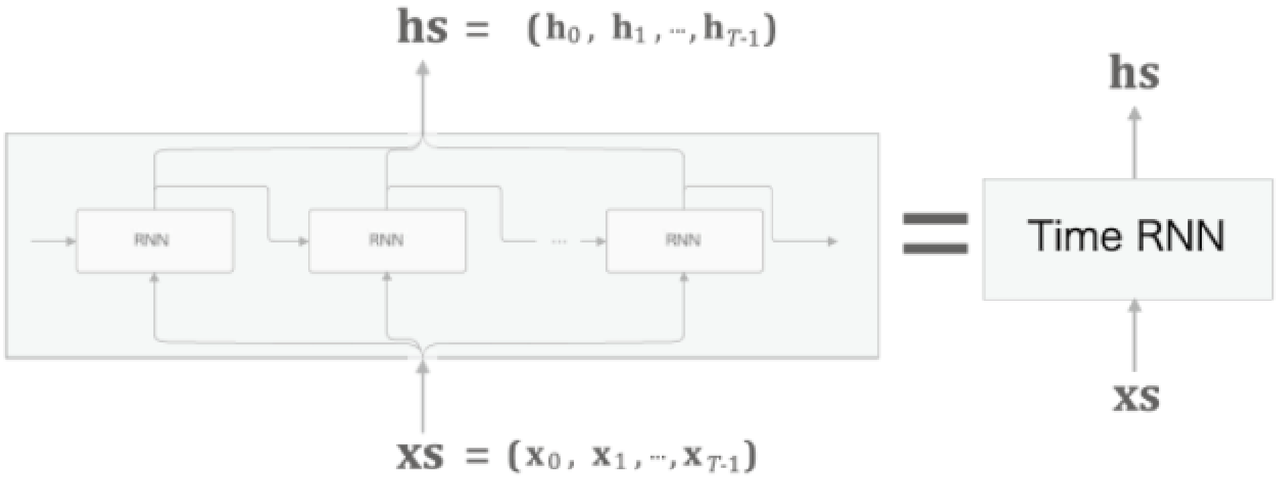
04

# RNN 구현하기

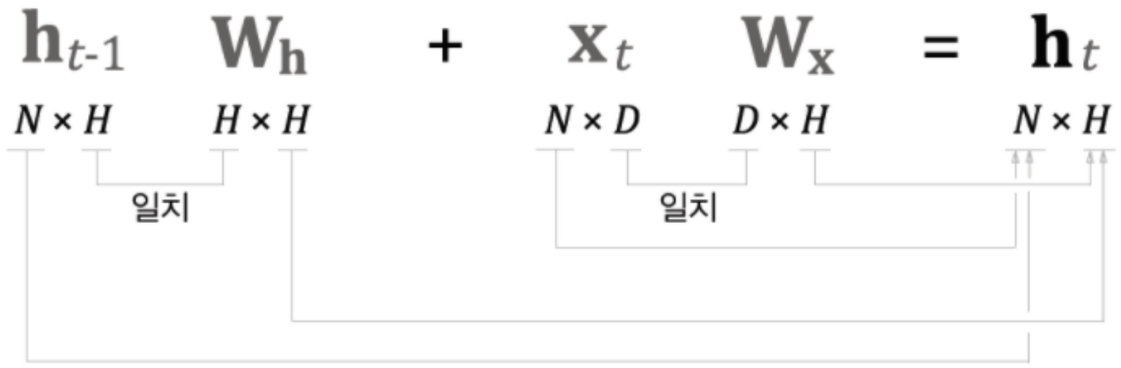
## RNN 구현

- 순환 구조를 펼친 후 나타난 가로 방향으로 성장하는 신경망과,
  상하 방향의 입출력을 하나로 묶어 하나의 계층으로 간주함
- 길이가 T인 시계열 데이터를 한꺼번에 처리하는 계층을 Time RNN 계층이라고 표현하며,
  Time RNN 계층 내에서 한 단계의 작업을 수행하는 계층을 RNN 계층이라고 표현함
- 즉, T 개의 RNN 계층이 모여 하나의 Time RNN 계층을 이루는 것

RNN 계층 구조




미니 배치를 적용한 RNN의 식



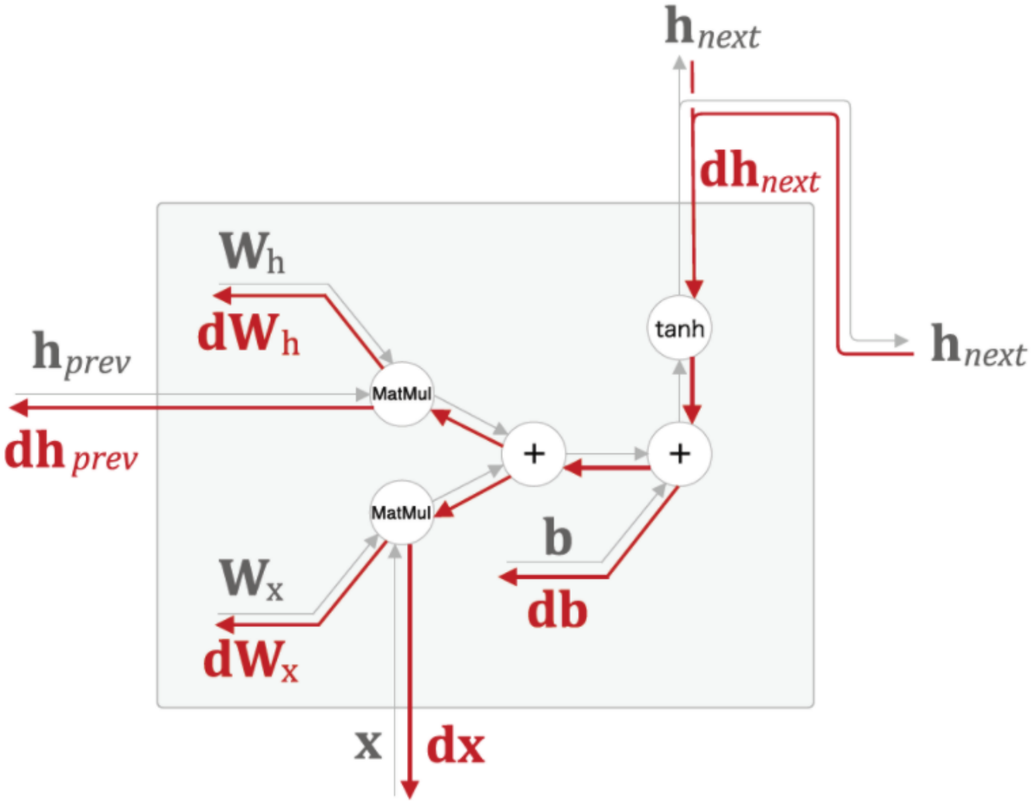

$x_t$: 입력값 $h_t$: 출력값 $W_x$: 입력값 가중치 $W_h$: 입력값 가중치

N: 미니배치 크기, D: 입력벡터의 차원 수, H: 은닉벡터의 차원 수

# RNN 클래스 코드

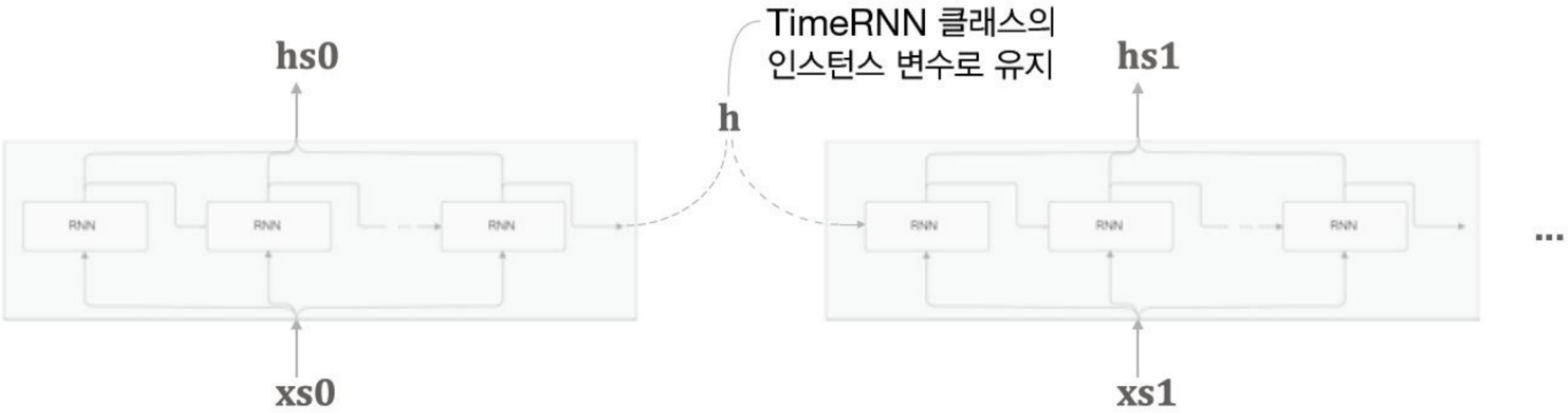
```python
class RNN:
    def __init__(self, Wx, Wh, b):
        self.params = [Wx, Wh, b] # 가중치 2개와 편향(bias) 1개 저장
        self.grads = [np.zeros_like(Wx), np.zeros_like(Wh), np.zeros_like(b)] # 기울기 초기화 후 저장
        self.cache = None # 역전파 계산 시 사용하는 중간 데이터를 담을 공간 마련

    # 순전파의 입력 데이터는 x, h_prev, 출력 데이터는 h_next
    def forward(self, x, h_prev):
        Wx, Wh, b = self.params
        t = np.dot(h_prev, Wh) + np.dot(x, Wx) + b # 행렬곱 진행
        h_next = np.tanh(t) # 활성화함수 tanh 처리

        self.cache = (x, h_prev, h_next)
        return h_next

    # 역전파의 입력 데이터는 dh_next, 출력 데이터는 dx, dh_prev
    def backward(self, dh_next):
        # Wx.shape=(D,H), Wh.shape=(H,H), b.shape=(H,)
        Wx, Wh, b = self.params
        # x.shape=(N,D), h_prev.shape=(N,H), h_next.shape=(N,H)
        x, h_prev, h_next = self.cache

        # dt.shape=(N,H)
        dt = dh_next * (1 - h_next ** 2) # tanh 함수 미분 = 1-tanh(x)^2
        db = np.sum(dt, axis=0) # db.shape=(H,)
        dWh = np.dot(h_prev.T, dt) # (H,N) * (N,H) => dwh.shape=(H,H)
        dh_prev = np.dot(dt, Wh.T) # (N,H) * (H,H) => dh_prev.shape=(N,H)
        dWx = np.dot(x.T, dt) # (D,N) * (N,H) => dWx.shape=(D,H)
        dx = np.dot(dt, Wx.T) # (N,H) * (H,D) => dx.shape=(N,D)

        self.grads[0][...] = dWx
        self.grads[1][...] = dWh
        self.grads[2][...] = db

        return dx, dh_prev
```

순전파 진행 방향 (역전파는 순전파의 반대 방향으로 진행)



## Time RNN 구현

- Time RNN 계층은 은닉 상태를 인스턴스 변수 h에 보관함으로써, 은닉 상태를 다음 블록으로 인계할 수 있음



## Time RNN 구현

- 매 반복문에서 RNN 계층에 self.params(Wx, Wh, b 포함)를 넘길 때, T에 대해 매번 같은 가중치 조건을 입력함
- Time RNN의 RNN 계층들은 시간적 표현을 위해 여러 계층처럼 보일 뿐, 본질적으로는 하나의 계층이기 때문

생성자 초기화 구현

```python
class TimeRNN:
    # 생성자 초기화 메소드: 가중치, 편향, stateful이라는 boolean값을 인수로 받음
    def __init__(self, Wx, Wh, b, stateful=False):
        self.params = [Wx, Wh, b] # Wx.shape=(D,H), Wh.shape=(H,H), b.shape=(H,)
        self.grads = [np.zeros_like(Wx), np.zeros_like(Wh), np.zeros_like(b)]
        self.layers = None # T개의 RNN 계층을 리스트로 저장하는 용도

        # h: forward() 시 마지막 RNN 계층의 은닉 상태를 저장
        # dh: backward() 시 하나 앞 블록의 은닉 상태의 기울기를 저장
        self.h, self.dh = None, None
        self.stateful = stateful # 은닉상태를 유지할 지 지정하는 변수
        # stateful = True: Time RNN 계층의 순전파를 끊지 않고 전파함
        # stateful = False : 은닉 상태 h를 영행렬로 초기화

    def set_state(self, h):
        self.h = h

    def reset_state(self):
        self.h = None
```

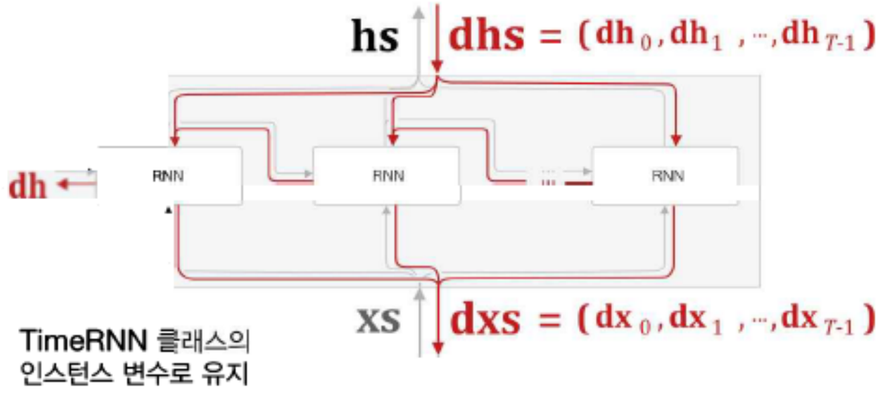
순전파 구현

```python
# 순전파 메소드: h에 마지막 RNN 계층의 은닉 상태를 저장,
# 다음번 forward() 호출에서 stateful=True이면 먼저 저장된 h값을 그대로 이용, Fals면 h는 영행렬로 초기화
def forward(self, xs): # xs.shape=(N,T,D)
        Wx, Wh, b = self.params
        # N: 미니 배치 크기
        # T: T개 분량 시계열 데이터를 하나로 모은 것
        # D: 입력 벡터의 차원 수
        N, T, D = xs.shape
        D, H = Wx.shape

        self.layers = []
        hs = np.empty((N, T, H), dtype='f') # 출력값을 담는 변수

        if not self.stateful or self.h is None:
            # 은닉 상태 h는 처음 호출 시, 또는 stateful=Fals인 경우 영행렬로 초기화
            self.h = np.zeros((N, H), dtype='f')

        for t in range(T): # 반복문을 통해 RNN 계층 T개 생성
            layer = RNN(*self.params) # *은 리스트 원소들을 추출하여 메소드의 인수로 전달하는 역할
            self.h = layer.forward(xs[:, t, :], self.h) # 각 시각 t의 은닉 상태 h를 계산, h.shape=(N,N)
            hs[:, t, :] = self.h # 계산 결과를 hs의 인덱스 값으로 설정
            self.layers.append(layer) # 각 RNN 계층의 params, grads는 역전파 시 다시 사용됨

        return hs # hs.shape=(N,T,H)
```

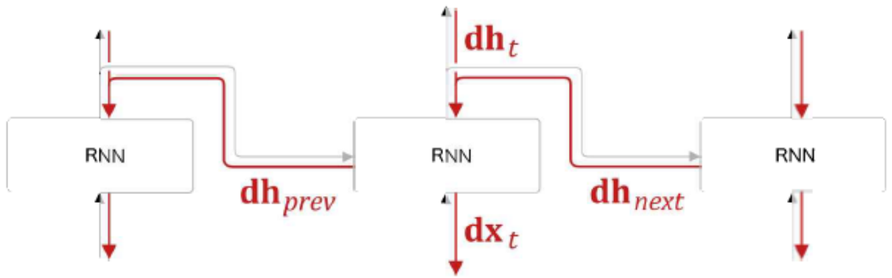
# Time RNN 구현

- 역전파에서는 Truncated BPTT를 수행하므로 이전 시각의 역전파는 불필요함
- 단, 이전 시각의 은닉 상태 기울기는 인스턴스 변수 dh에 저장되어 전달됨
- t번째 RNN 계층은 위에서부터 기울기 $dh_t$ 를, 미래 계층으로부터 기울기 $dh_{next}$ 를 입력받음
  순전파 출력이 2개로 분기되었기 때문에, 역전파에서는 각 기울기가 합산되어 전해짐

역전파 과정




Time RNN 계층의 계산 그래프( 빨강 - 역전파 )




역전파 구현

```python
def backward(self, dhs): # dhs: 상류(출력층 쪽)에서 전해지는 기울기 # dhs.shape=(N,T,H)
        Wx, Wh, b = self.params
        N, T, H = dhs.shape
        D, H = Wx.shape

        dxs = np.empty((N, T, D), dtype='f') # dxs: 하류로 내보내는 기울기
        dh = 0 # Truncated BPTT로 가정, 이전 블록의 은닉 상태는 불필요
        grads = [0, 0, 0] # 이번 블록의 최종 미분값을 담는 변수 # [dWx, dWh, b]
        for t in reversed(range(T)):
            layer = self.layers[t] # 순전파와 반대 순서로 RNN 계층의 backward 메소드 호출
            # 순전파 시 분기된 h가 역전파 시 각 기울기가 합산되어 dhnext + dh로 전해짐
            dx, dh = layer.backward(dhs[:, t, :] + dh)
            dxs[:, t, :] = dx

            for i, grad in enumerate(layer.grads):
                grads[i] += grad
            # Time RNN의 여러 RNN 계층은 모두 똑같은 가중치를 사용
            # # 따라서 Time RNN 계층의 최종 가중치 기울기는 각 RNN 계층의 가중치 기울기를 모두 더한 값

        for i, grad in enumerate(grads):
            # 순전파 과정에서 시각 t마다 입력된 Xs 값이 다르며,
            # 역전파에서는 입력받은 미분 dht 값들이 모두 다르기 때문에, 시간마다 모두 다른 기울기 grad가 산출됨
            self.grads[i][...] = grad  # 최종 가중치 기울기로 이전 블록의 가중치 기울기를 덮어 씀

        self.dh = dh

        return dxs # dxs.shape=(N,T,D)
```

05

# RNN의 한계점과 발전 방향

## RNN의 문제점

- 병렬화(Paralleization) 불가능

  RNN은 벡터가 순차적으로 입력되어 Sequential한 데이터를 처리할 수 있게 하지만,

  GPU 연산의 장점인 병렬화를 불가능하게 만드므로, RNN 기반 모델은 GPU 연산을 했을 때 이점이 거의 없음

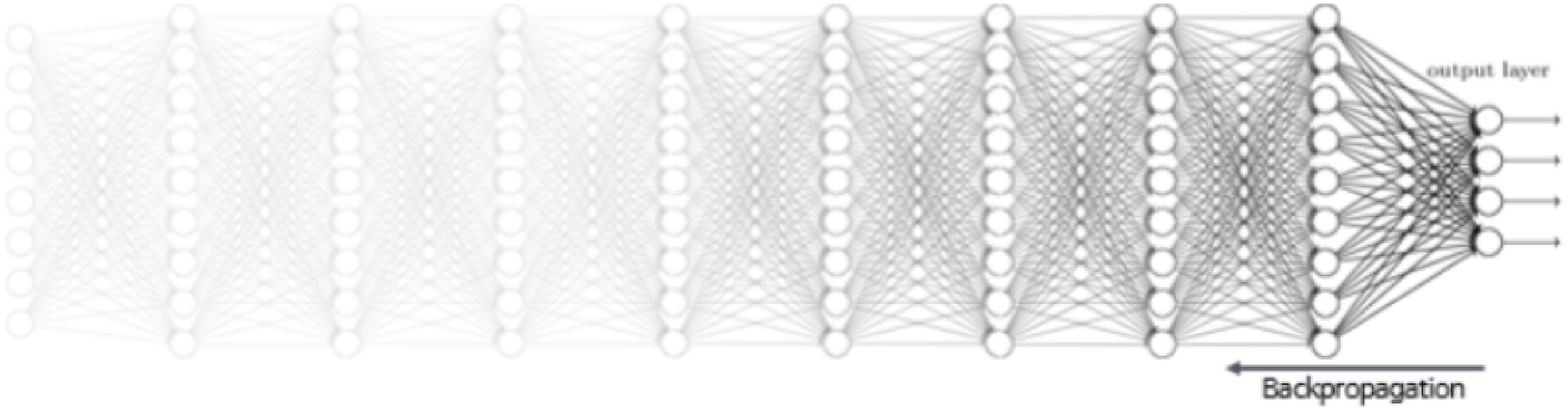
- 장기 의존성(Long-Term Dependencies) 문제

  BPTT 과정에서 기울기 소실 혹은 기울기 폭발이 발생할 경우, 장기 의존성 문제 발생

  이로 인해 시퀀스 데이터의 길이가 길어질수록, 과거의 중요한 정보에 대한 학습이 어려워짐

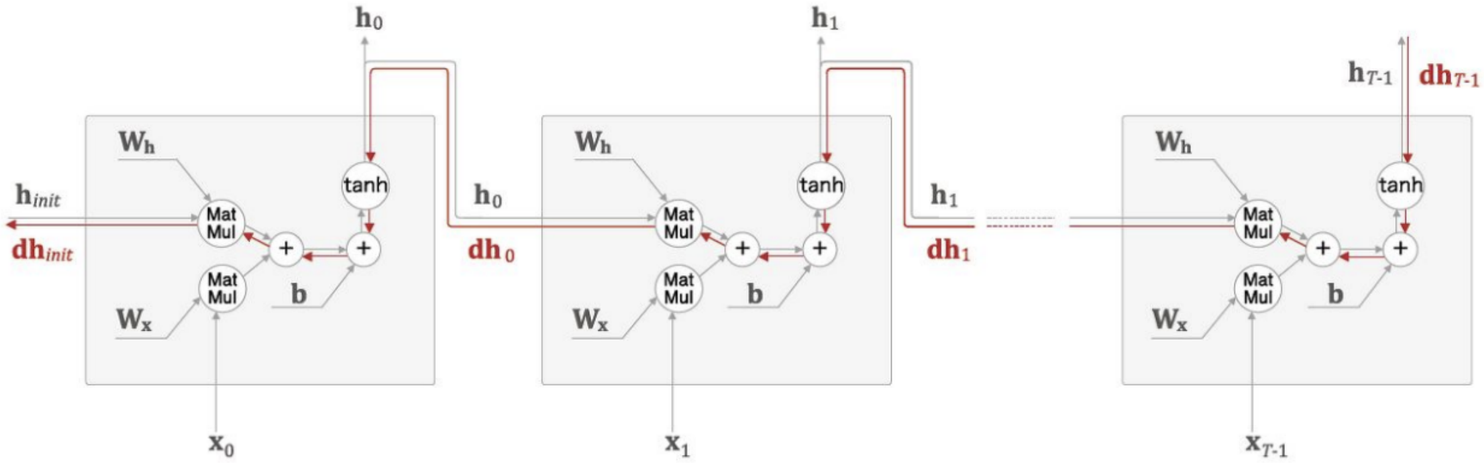
  => BPTT 과정에서 기울기를 과거로 전달해 장기 의존 관계를 학습할 수 있지만,

  중간에 기울기가 소실되거나 너무 커지면 장기 의존 관계를 학습할 수 없게 됨



## RNN의 기울기 소실 및 폭발 원인

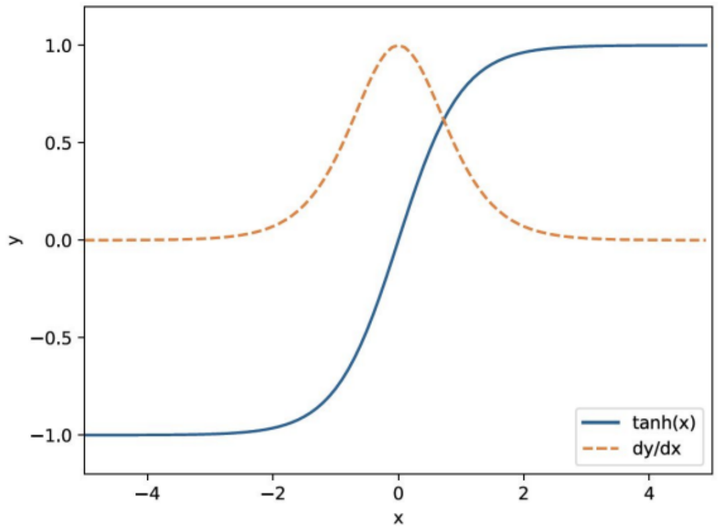
- BPTT 과정에서 tanh와 행렬곱으로 기울기를 변화시키는 연산을 수행함




- tanh 노드를 지날 경우: 기울기 소실

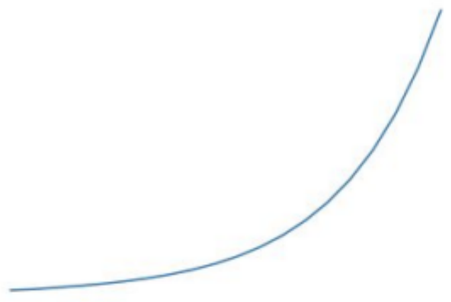
  미분값은 1과 0 사이이므로, 역전파에서
  tanh 노드를 지날 때마다 기울기가 계속해서 작아지게 됨

그림 6-6 $y = \tanh(x)$의 그래프(점선은 미분)

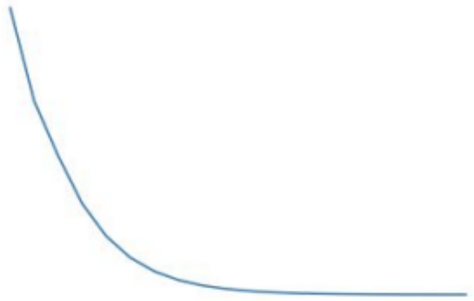



- 행렬곱 노드를 지날 경우: 기울기 폭발 혹은 기울기 소실

  상류로부터 $dh_t$라는 기울기가 들어왔을 때, $h$에 대한 미분값을 구하기 위해 $dh_t * W_h^T$ 행렬곱을 계산하게 되는데, 이 행렬곱에서 매번 같은 가중치 $W_h$를 사용하게 됨



행렬 특이값이 1보다 클 경우
기울기 폭발



행렬 특이값이 1보다 작을 경우
기울기 소실

## RNN의 기울기 폭발에 대한 대책안

- 기울기 클리핑(Gradient Clipping)

  $g-bar$ 는 L2 norm. L2 norm 값이 threshold 값을 초과하면 기울기를 수정함

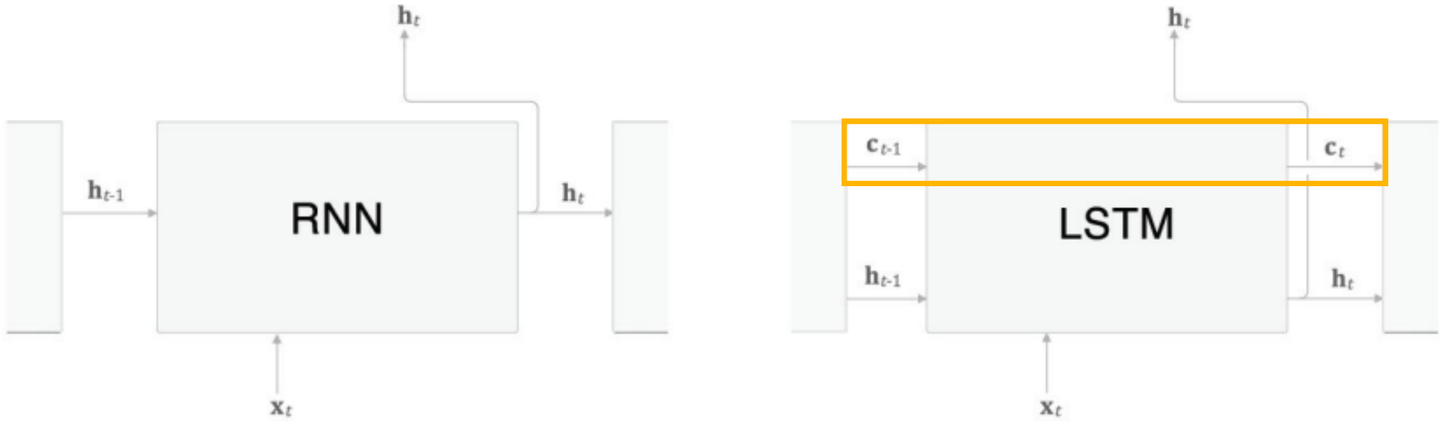
  $$\text{if } ||g-bar|| \geq threshold:$$
  $$g-bar = \frac{threshold}{||g-bar||}||g-bar||$$

## RNN의 기울기 소실에 대한 대책안

- LSTM (Long Short Term Memory)

  기울기 소실에 대응하기 위한 게이트가 추가된 RNN 아키텍쳐

  => 은닉층 메모리 셀에 입력 게이트, 망각 게이트, 출력 게이트를 추가하여, 불필요한 정보를 지우고, 중요한 정보를 정함



06

# Week 5
# 실습 안내

## RNN을 이용한 한국어 텍스트 생성

- LSTM의 토대가 되는 RNN 모델!
- RNN 모델이 문장을 학습하고, 예측하는 과정을 구현해봅시다.

## 과제에서 채워주셔야 하는 부분

1) 예제에서 사용된 문장이 아닌 새로운 문장을 학습 데이터로 입력해주기
2) RNN 모델의 임베딩 벡터 차원과 은닉 상태 크기를 적절하게 입력해 accuracy를 높이고 loss를 줄이기
3) 자신이 입력한 파라미터를 바탕으로 학습한 모델의 최종 accuracy와 loss 값이 얼마인지 작성하기
4) 자신이 입력한 문장을 기반으로 새롭게 예측을 수행하기

열심히 공부해주시는만큼 과제 검토도 꼼꼼히 하겠습니다 :)

모든 질문은 pull request를 통해 해주세요!

**CS231N: Recurrent Neural Networks**

*https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/*

*https://velog.io/@cha-suyeon/CS231n-Lecture-10-%EA%B0%95%EC%9D%98-%EC%A0%95%EB%A6%AC*

*https://brightwon.tistory.com/10*

**밑바닥부터 시작하는 딥러닝**

*https://seoromin.tistory.com/19*

*https://wikidocs.net/46496*

*https://luminitworld.tistory.com/75*

**딥 러닝을 이용한 자연어 처리 입문**

*https://wikidocs.net/45101*

# THANKYOU

BOAZ 분석 20기 BASE SESSION

Week 5. RNN

분석 19기 정은진