

5. 순환 신경망(RNN)

5.1 확률과 언어 모델

5.2 RNN 이란

5.3 RNN 구현

5.4 시계열 데이터 처리 계층 구현

5.5 RNNLM 학습과 평가

피드포워드(feed forward) 신경망

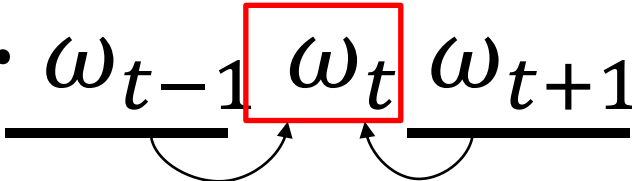
- 흐름이 단방향
- 시계열 데이터의 성질(패턴)을 충분히 학습할 수 없음

순환 신경망(Recurrent Neural Network, RNN)의 등장

CBOW(Continuous bag-of-words)모델

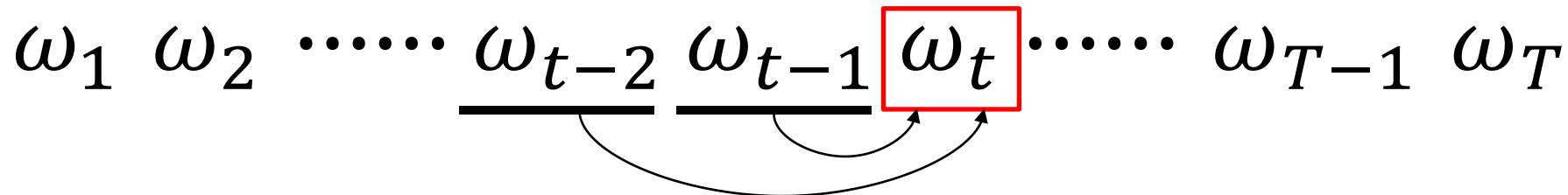
- CBOW 모델의 학습 -> 손실함수(말뭉치 전체의 손실함수의 총합)를 최소화하는 가중치 매개변수를 찾는다. -> 맥락으로부터 타깃을 더 정확하게 추측 가능
- 맥락 안의 단어 순서가 무시된다는 한계가 있음
말뭉치 : $w_1, w_2, w_3, \dots, w_t$

word2vec의 CBOW 모델 : 맥락의 단어로부터 타깃 단어를 추측한다.

$$\omega_1 \quad \omega_2 \quad \cdots \quad \omega_{t-1} \quad \boxed{\omega_t} \quad \omega_{t+1} \quad \cdots \quad \omega_{T-1} \quad \omega_T$$


$$P(w_t | w_{t-1}, w_{t+1})$$

왼쪽 윈도우만 맥락으로 고려한다.



$w(t-2)$ 과 $w(t-1)$ 이 주어졌을 때 타겟이 w_t 가 될 확률(CBOW 모델이 출력할 확률)을 수식으로 표현하면

$$P(w_t | w_{t-2}, w_{t-1})$$

CBOW모델이 다루는 손실함수

$$L = -\log P(w_t | w_{t-2}, w_{t-1})$$

단어 나열에 확률을 부여
 특정한 단어의 시퀀스에 대해서 그 시퀀스가 일어날 가능성이
 어느 정도인지(얼마나 자연스러운 단어 순서인지)를 확률로 평가한다.

- 언어 모델의 사용
 기계 번역과 음성 인식
 새로운 문장을 생성

w_1, \dots, w_m 이라는 m 개 단어로 된 문장이 있을 때
 w_1, \dots, w_m 순서로 출현할 확률 $P(w_1, \dots, w_m)$
 (여러 사건이 동시에 일어날 확률이므로 동시확률이라고 한다.)

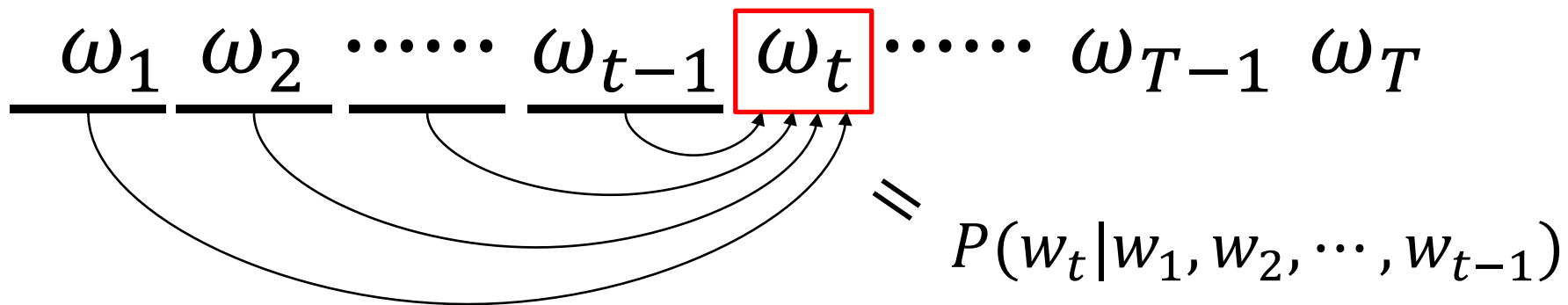
$$\begin{aligned} P(w_1, \dots, w_m) &= P(w_m | w_1, \dots, w_{m-1}) P(w_{m-1} | w_1, \dots, w_{m-1}) \\ &\quad \dots P(w_3 | w_1, w_2) P(w_2 | w_1) P(w_1) \\ &= \prod_{t=1}^m P(w_t | w_1, \dots, w_{t-1}) \end{aligned}$$

$$P(A, B) = P(A|B)P(B)$$

A, B가 모두 일어날 확률 $P(A, B)$ 는 B가 일어날 확률 $P(B)$ 와 B가 일어난 후 A가 일어날 확률 $P(A|B)$ 를 곱한 값과 같다.

$$\underbrace{P(w_1, \dots, w_{m-1}, w_m)}_A = P(A, w_m) = P(w_m|A)P(A)$$

$$P(A) = P(\underbrace{w_1, \dots, w_{m-2}, w_{m-1}}_{A'}) = P(A', w_{m-1}) = P(w_{m-1}|A'')P(A')$$



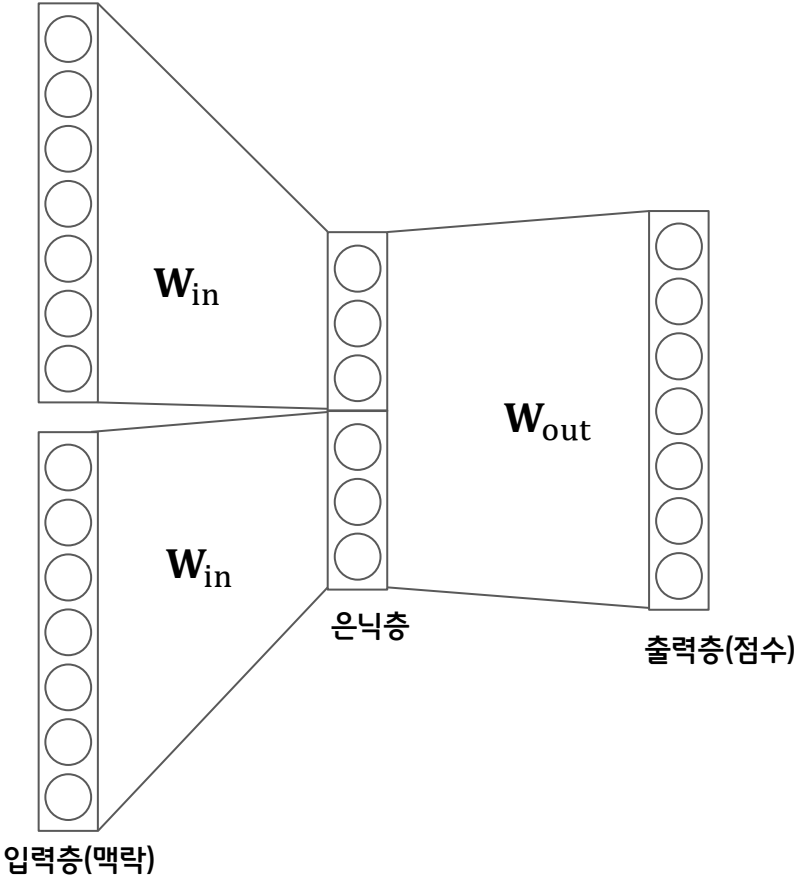
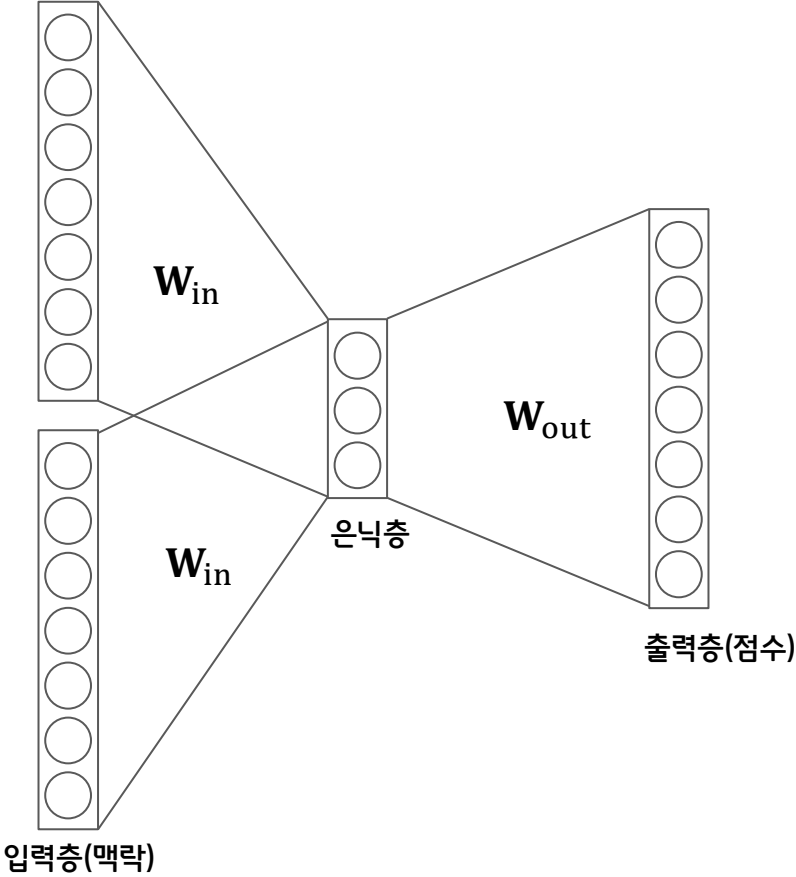
$$\omega_1 \ \omega_2 \ \cdots \ \omega_{t-1} \ \boxed{\omega_t} \ \cdots \ \omega_{T-1} \ \omega_T$$

$$= P(\omega_t | \omega_1, \omega_2, \dots, \omega_{t-1})$$

- word2vec의 CBOW모델을 언어 모델에 적용하려면 맥락의 크기를 특정 값으로 한정하여 근사적으로 나타낼 수 있다.
- 맥락의 크기는 임의 길이로 설정할 수 있지만 결국 특정 길이로 '고정'된다.
 - 예를 들어 왼쪽 10개의 단어를 맥락으로 CBOW 모델을 만든다고 하면 그 맥락보다 더 왼쪽에 있는 단어의 정보는 무시된다.
- CBOW모델의 맥락 크기를 키울 수는 있으나 맥락 안의 단어 순서가 무시된다는 한계가 있다.
- 맥락의 단어 순서를 고려하기 위해 맥락의 단어 벡터를 은닉층에서 연결(concatenate)하는 방식을 생각할 수 있으나 맥락의 크기에 비례해 가중치 매개변수가 늘어난다는 문제가 발생한다.

그래서 순환 신경망 즉 RNN이 등장하게 되었는데 RNN은 맥락이 아무리 길더라도

맥락의 정보를 기억하는 메커니즘을 갖추고 있기에 아무리 긴 시계열 데이터에도 대응할 수 있다.



5. 순환 신경망(RNN)

5.1 확률과 언어 모델

5.2 RNN 이란

5.3 RNN 구현

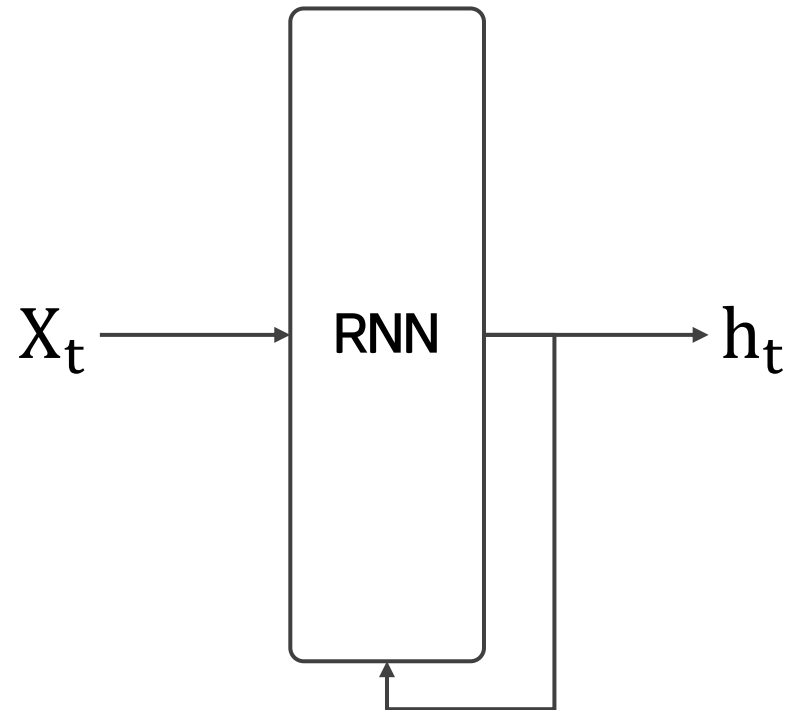
5.4 시계열 데이터 처리 계층 구현

5.5 RNNLM 학습과 평가

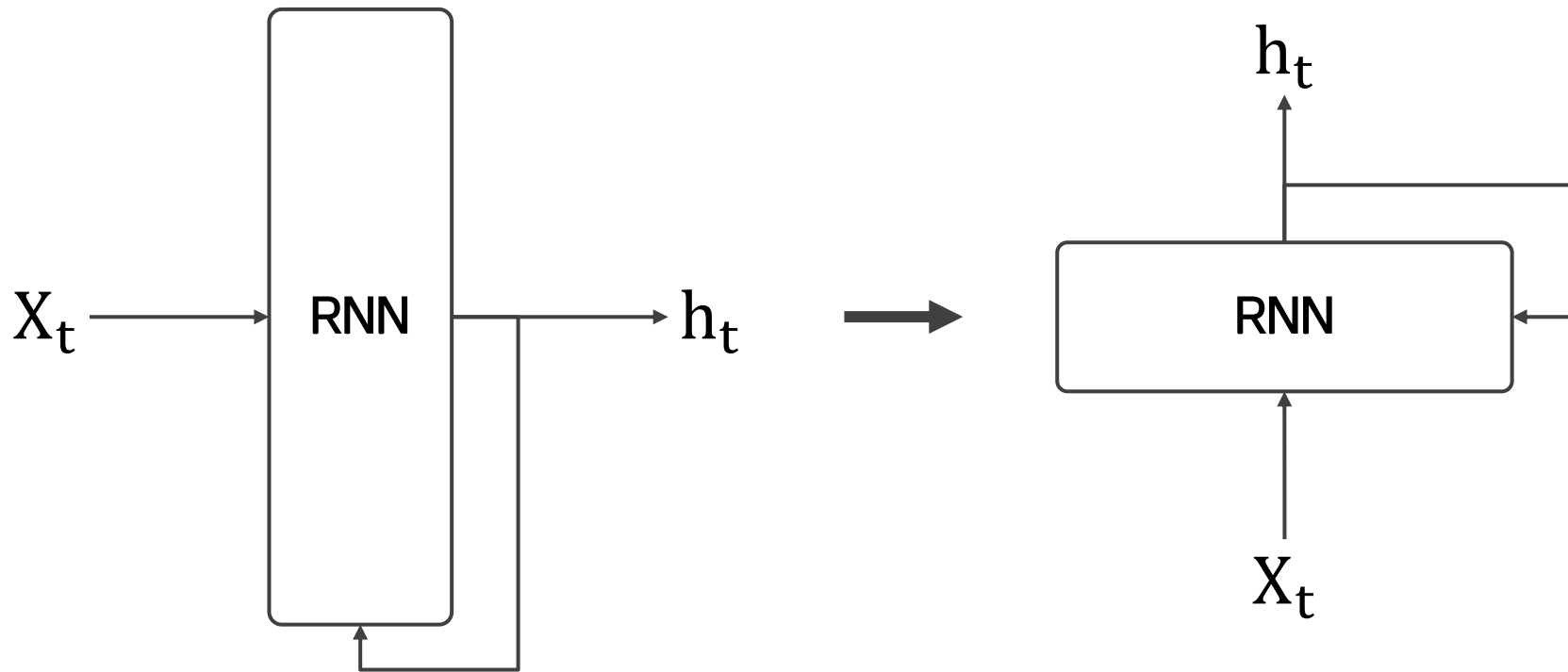
순환하기 위해서는 닫힌 경로가 필요하다.

닫힌 경로 혹은 순환하는 경로가 존재해야 데이터가 같은 장소를 반복해 왕래할 수 있고 데이터가 순환하면서 과거의 정보를 기억하는 동시에 최신 데이터로 갱신 될 수 있다.

순환 경로를 포함하는 RNN 계층



계층을 90도 회전시켜 그린다.



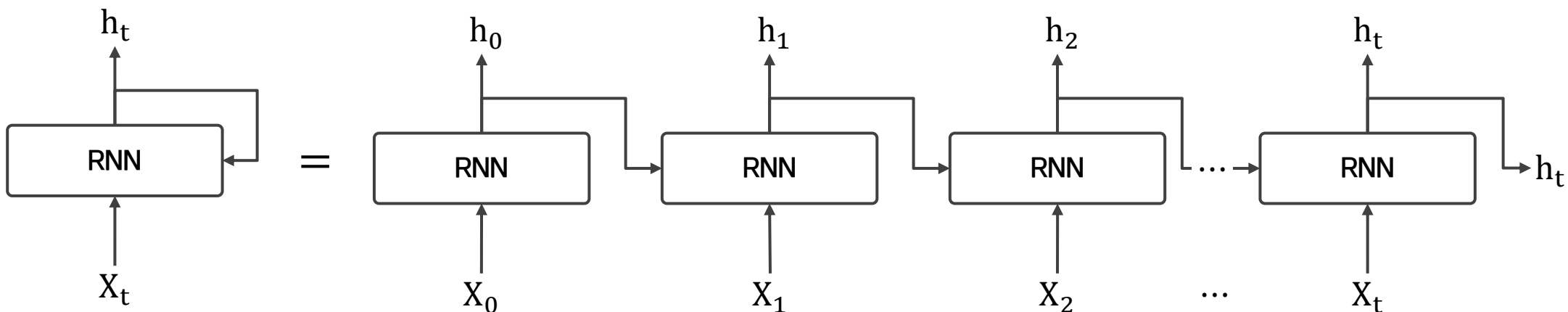
t: 시각

시계열 데이터($x_0, x_1, \dots, x_t, \dots$)가 RNN계층에 입력되고 이에 대응해 ($h_0, h_1, \dots, h_t, \dots$)가 출력된다.

각 시각에 입력되는 x_t 를 벡터라고 가정했을 때

문장(단어 순서)을 다루는 경우를 예로 든다면 각 단어의 분산 표현(단어 벡터)이 x_t 가 되며 이 분산 표현이 순서대로 하나씩 RNN계층에 입력된다.

RNN 계층의 순환 구조 펼치기



RNN계층의 순환 구조를 펼침으로써 오른쪽으로 성장하는 긴 신경망으로 변신
피드포워드 신경망(데이터가 한 방향으로만 흐른다)과 같은 구조이지만 위 그림에서는
다수의 RNN계층 모두가 실제로는 '같은 계층'인 것이 지금까지의 신경망과는 다른 점이다.

각 시각의 RNN계층은 그 계층으로의 입력과 1개 전의 RNN계층으로부터의 출력을 받는데
이 두 정보를 바탕으로 현 시각의 출력을 계산한다.

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

W_x : 입력 x 를 출력 h 로 변환하기 위한 가중치

W_h : 1개의 RNN출력을 다음 시각의 출력으로 변환하기 위한 가중치

b : 편향

$h(t-1)$, x_t : 행벡터

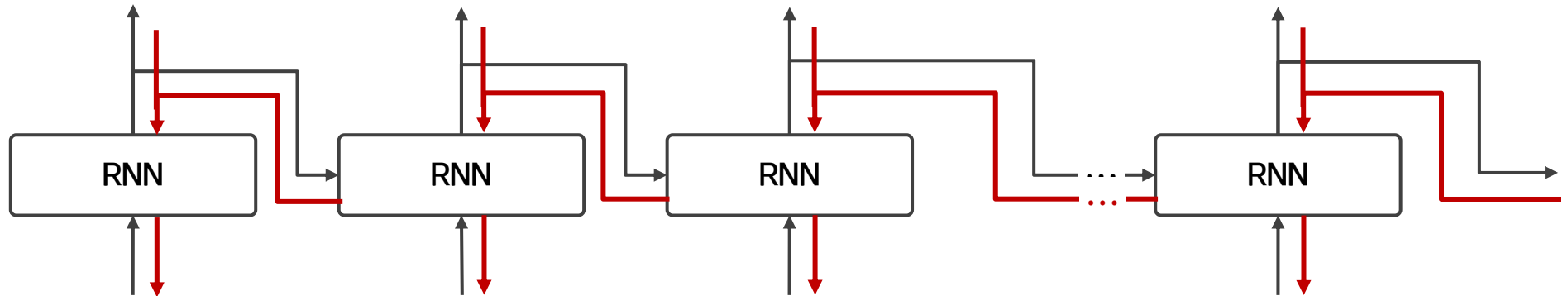
h_t 는 다른 계층을 향해 위쪽으로 출력되는 동시에 다음 시각의 RNN계층(자기 자신)을 향해 오른쪽으로도 출력된다.

RNN의 출력 h_t 는 은닉상태(hidden state) 혹은 은닉 상태 벡터(hidden state vector)라고 한다.

RNN은 h 라는 '상태'를 가지고 있으며 위의 식의 형태로 갱신된다고 해석할 수 있다.

RNN계층을 '상태를 가지는 계층' 혹은 '메모리(기억력)이 있는 계층'이라고 한다.

순환 구조를 펼친 RNN 계층에서의 오차역전파



순환 구조를 펼친 후의 RNN에는 (일반적인) 오차역전파법을 적용할 수 있다.
먼저 순전파를 수행하고 이어서 역전파를 수행하여 원하는 기울기를 구할 수 있다.
여기서의 오차역전파법은 '시간 방향으로 펼친 신경망의 오차역전파법'이란 뜻으로
BPTT(Backpropagation Through Time)라고 한다.

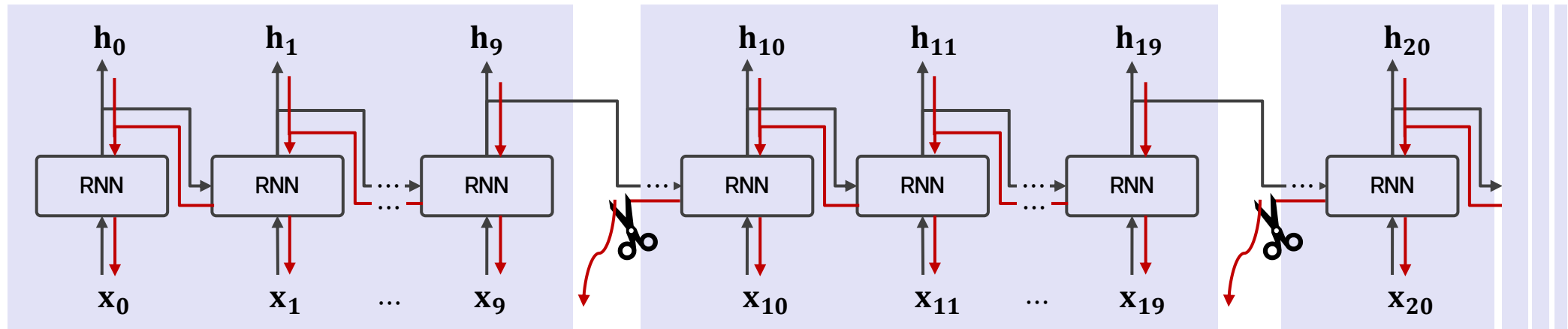
문제점

- 시계열 데이터의 시간 크기가 커지는 것에 비례하여 BPTT가 소비하는 컴퓨팅 자원도 증가
- 시간 크기가 커지면 역전파 시의 기울기가 불안정해짐

Truncated BPTT : 시간축 방향으로 너무 길어진 신경망을 적당한 지점에서 잘라내어 작은 신경망 여러 개로 만들어 잘라낸 작은 신경망에서 오차역전파법을 수행한다.

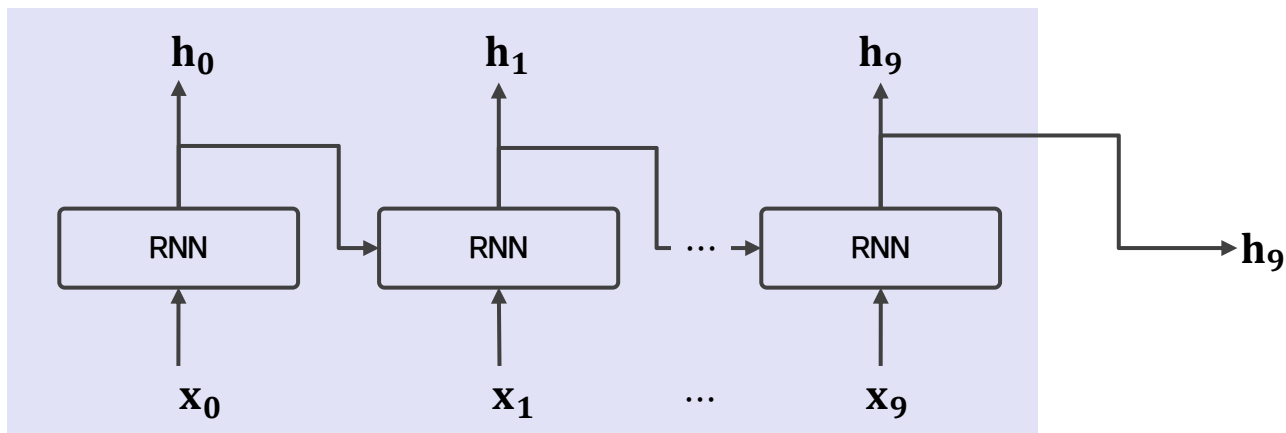
- 계층이 너무 길면 계산량과 메모리 사용량 등이 문제가 되고 계층이 길어짐에 따라 신경망을 하나 통과할 때마다 기울기 값이 조금씩 작아져서 이전 시각 t 까지 역전파되기 전에 0이 되어 소멸할 수도 있다.
- 순전파의 연결을 그대로 유지하면서(데이터를 순서대로 입력해야 한다) 역전파의 연결은 적당한 길이로 잘라내 잘라낸 신경망 단위로 학습을 수행한다.
- 역전파의 연결을 잘라버리면 그보다 미래의 데이터에 대해서는 생각할 필요가 없어지기 때문에 각각의 블록 단위로 미래의 블록과는 독립적으로 오차역전파법을 완결시킨다.
 - 블록: 역전파가 연결되는 일련의 RNN계층
- 순전파를 수행하고 그 다음 역전파를 수행하여 원하는 기울기를 구한다.
- 다음 역전파를 수행할 때 앞 블록의 마지막 은닉 상태인 h_t 가 필요하다.
 h_t 로 순전파가 계속 연결될 수 있다.

역전파의 연결을 적당한 지점에서 끊는다.

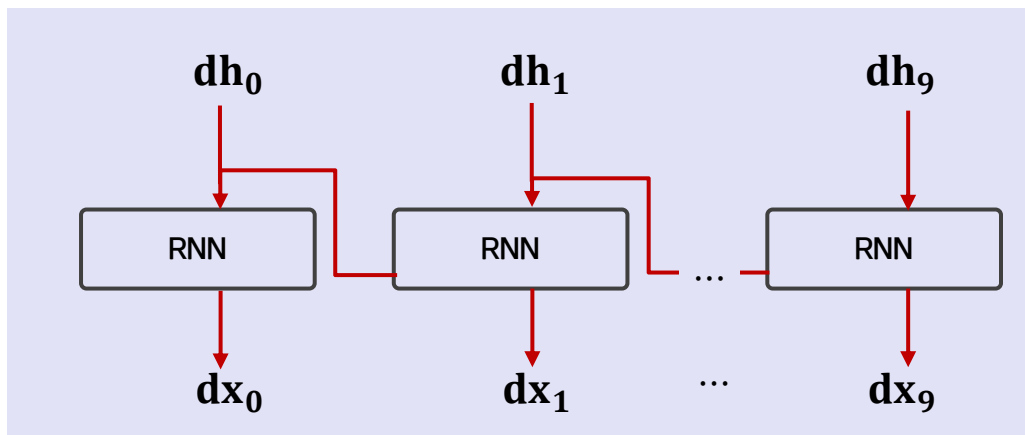


첫 번째 블록의 순전파와 순전파 : 이보다 앞선 시각으로부터의 기울기는 끊겼기 때문에 이 블록 내에서만 오차역전파법이 완결된다.

순전파

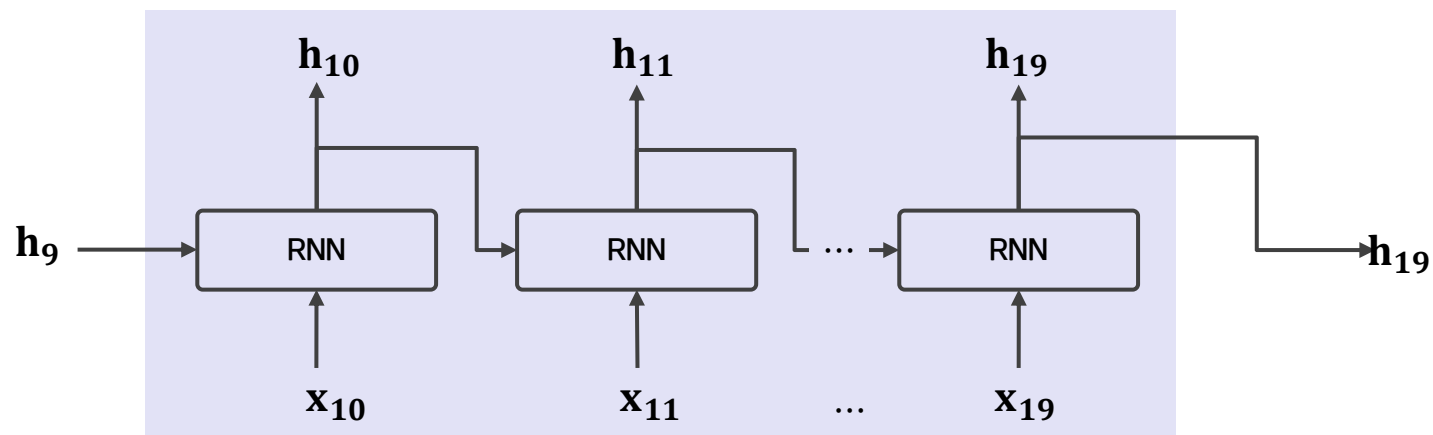


역전파

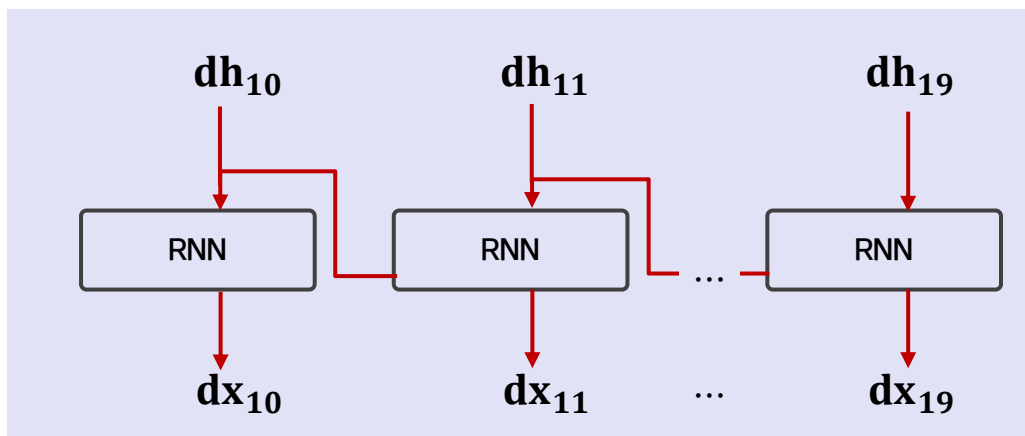


두 번째 블록의 순전파와 순전파

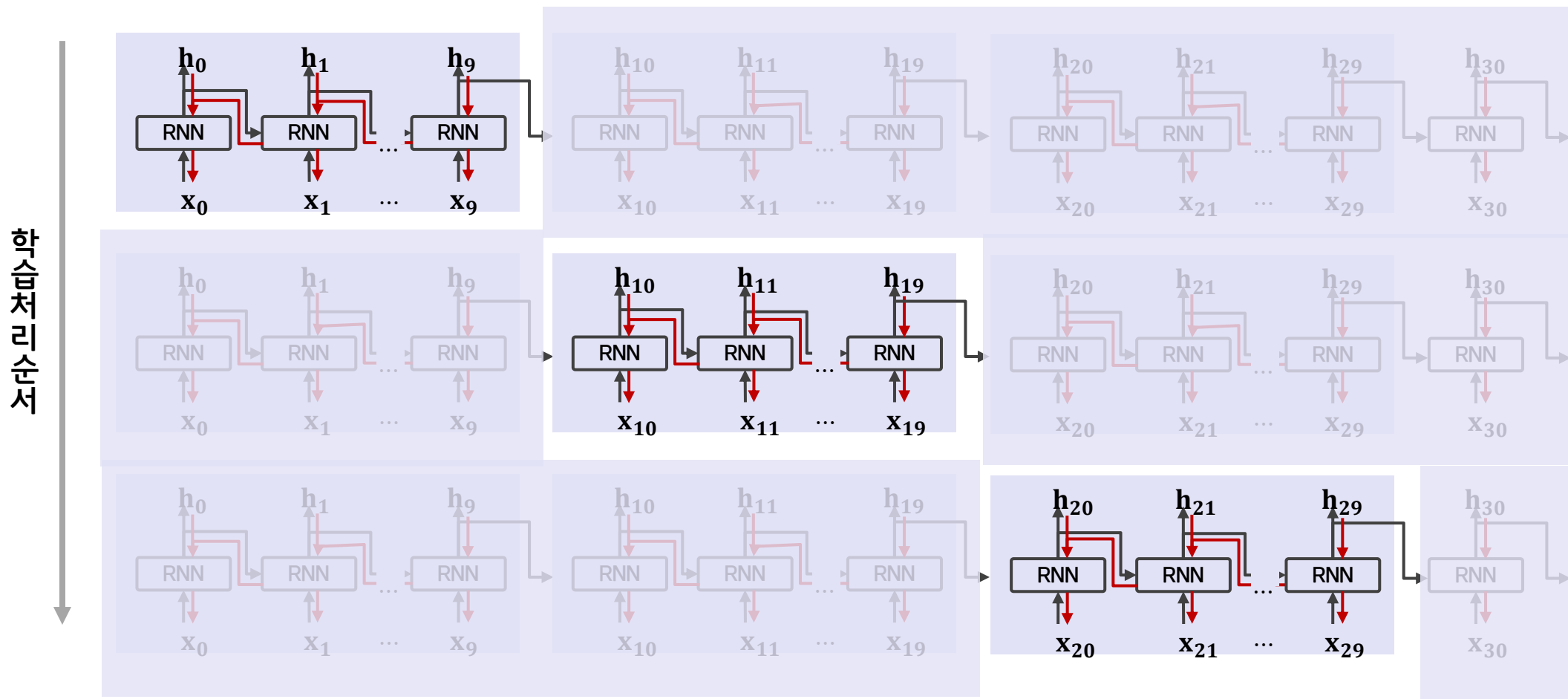
순전파



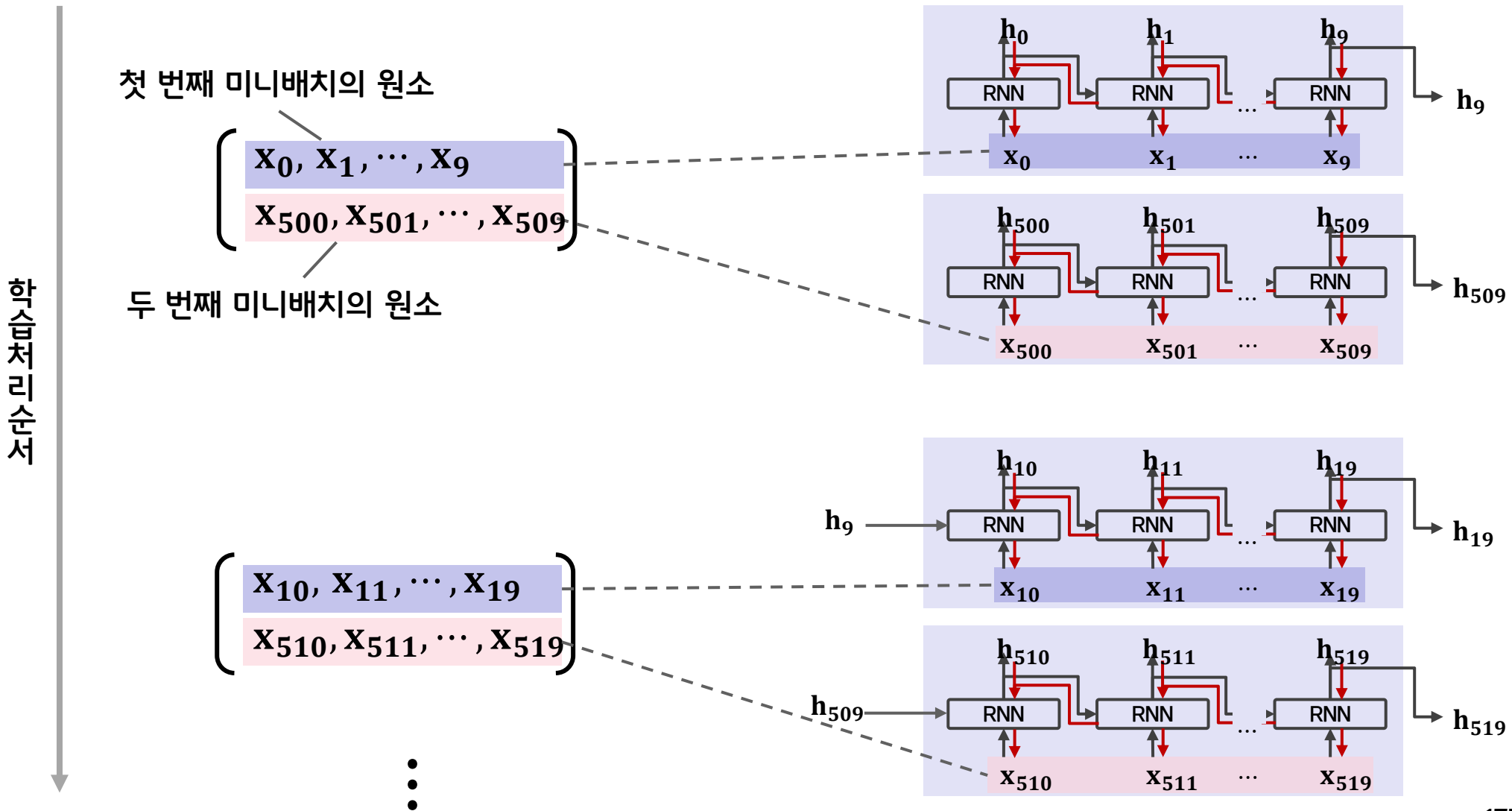
역전파



Truncated BPTT의 데이터 처리 순서



미니배치 학습 시 데이터를 제공하는 시작 위치를 각 미니배치로 옮긴다.



5. 순환 신경망(RNN)

5.1 확률과 언어 모델

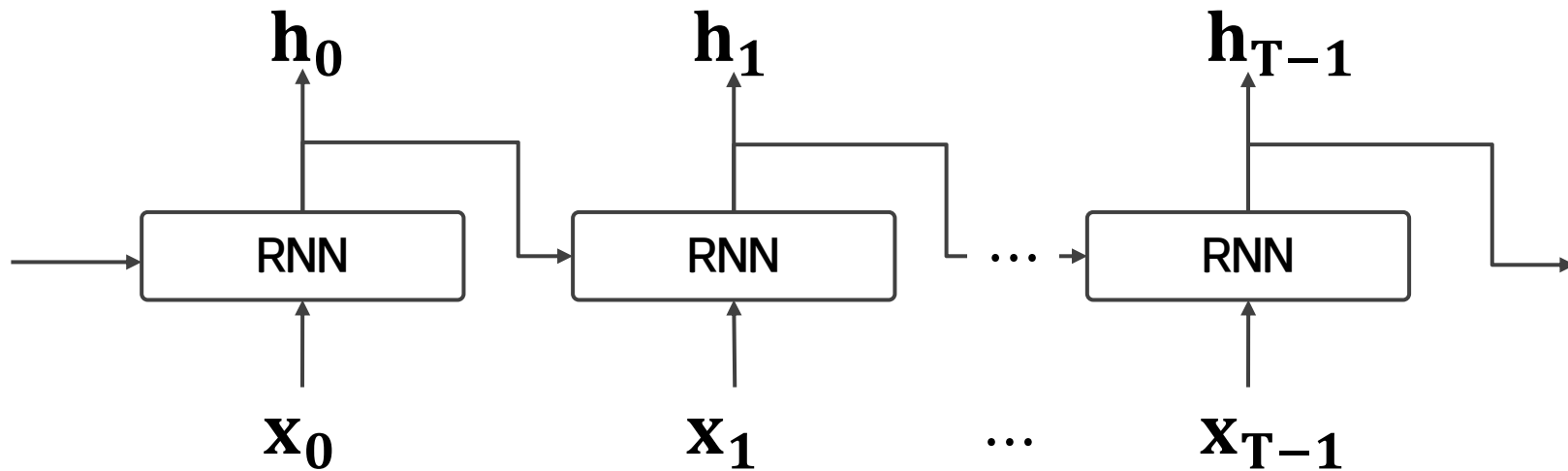
5.2 RNN 이란

5.3 RNN 구현

5.4 시계열 데이터 처리 계층 구현

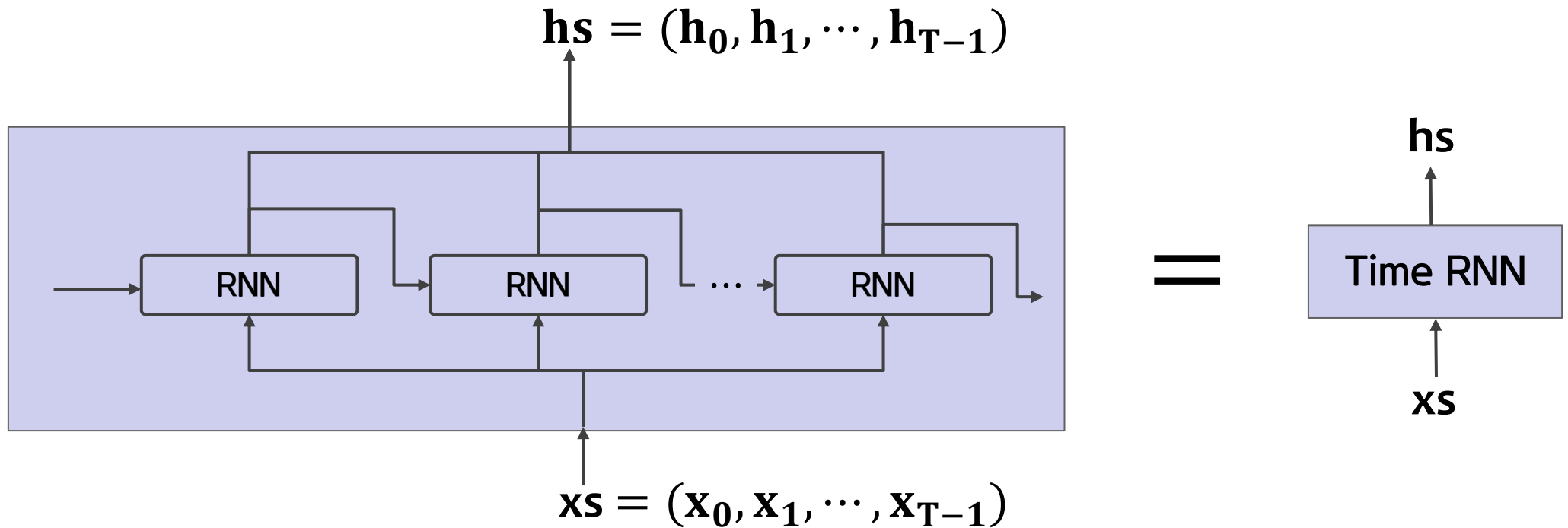
5.5 RNNLM 학습과 평가

RNN에서 다루는 신경망



길이가 T 인 시계열 데이터를 받는다.
각 시각의 은닉 상태를 T 개 출력한다.
모듈화를 생각해 위의 그림의 신경망을 '하나의 계층'으로 구현한다.

Time RNN 계층 : 순환 구조를 펼친 후의 계층들을 하나의 계층으로 간주한다.

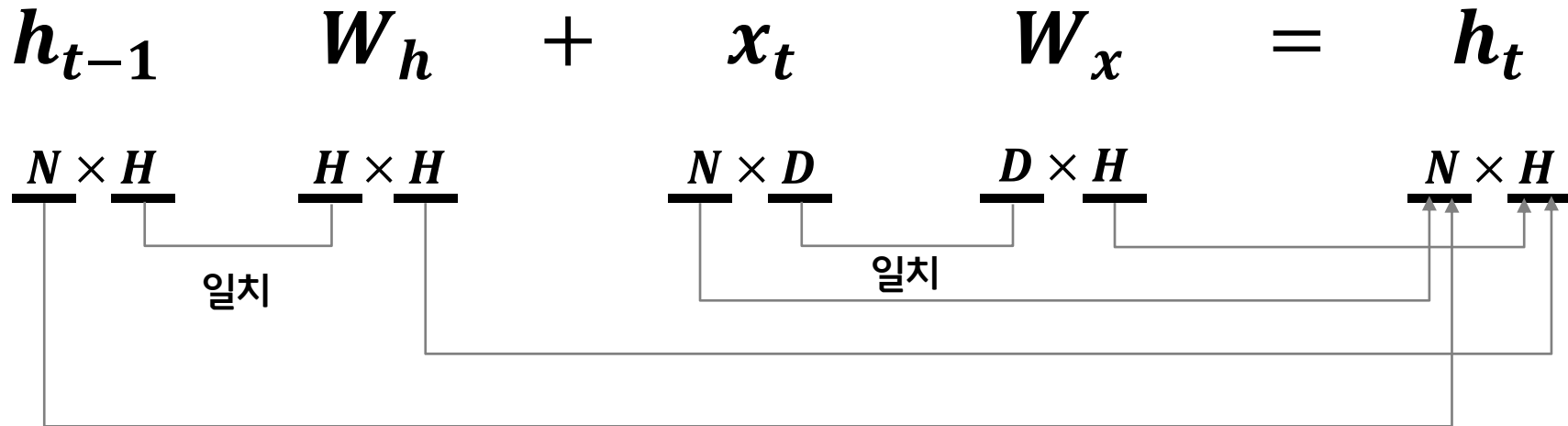


x_s 를 입력하면 h_s 를 출력하는 단일 계층

Time RNN계층 내에서 한 단계의 작업을 수행하는 계층을 'RNN계층'이라 하고
T개 단계분의 작업을 한꺼번에 처리하는 계층을 'Time RNN계층'이라 한다.

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

형상 확인 : 형렬 곱에서는 대응하는 차원의 원소 수를 일치시킨다.

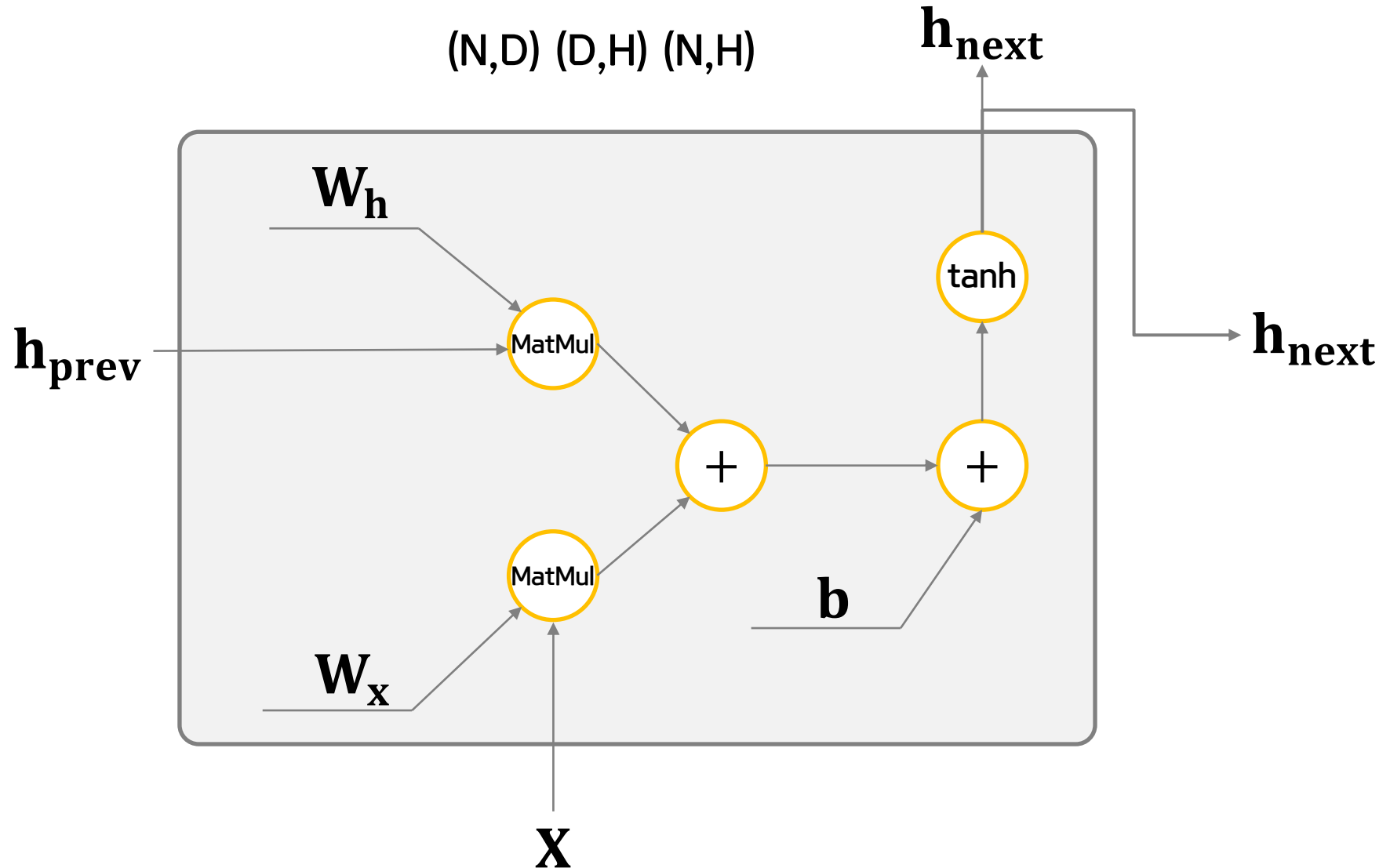


N: 미니배치 크기 D: 입력 벡터의 차원 수 H: 은닉 상태 벡터의 차원 수

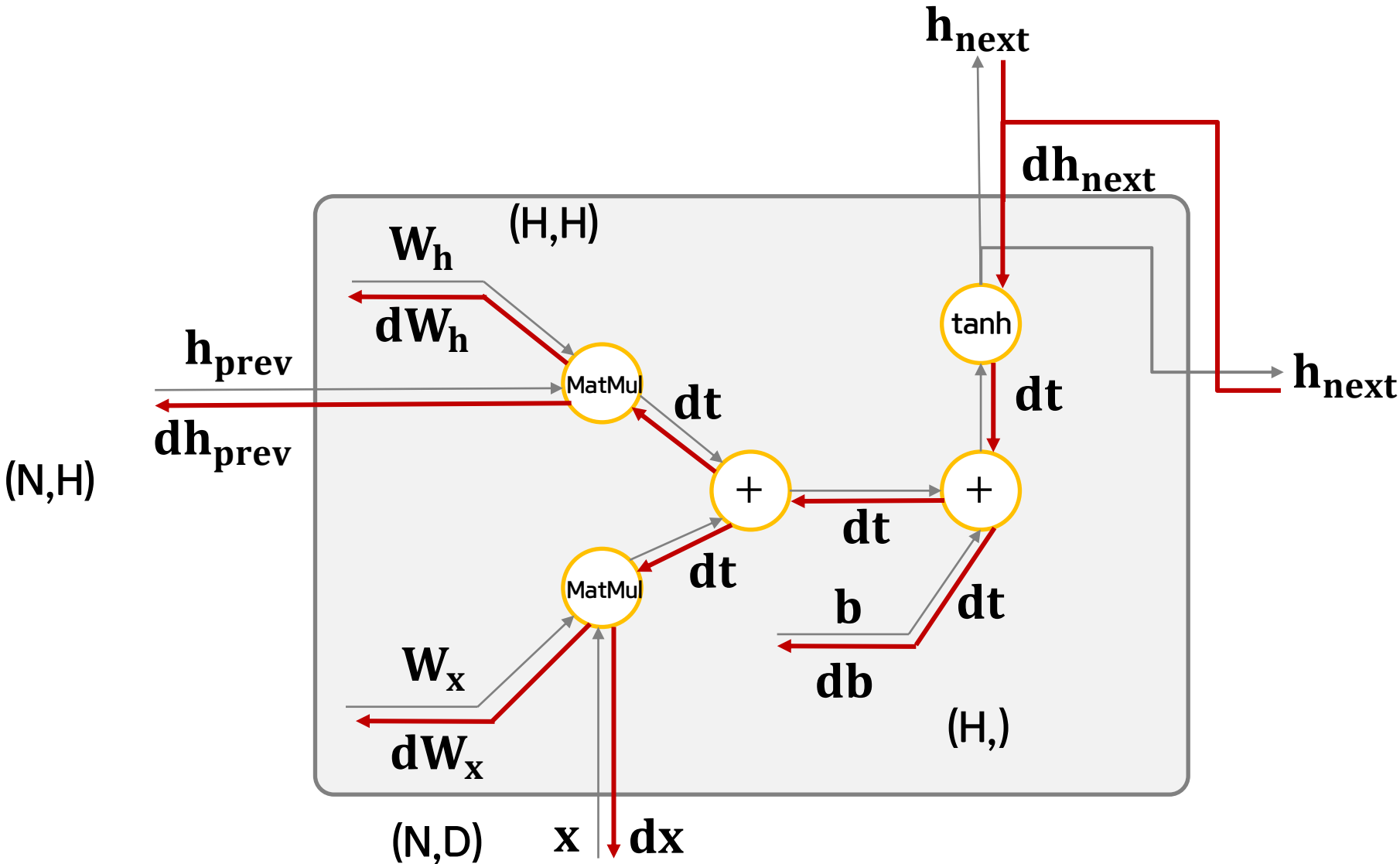
RNN 계층의 계산 그래프

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

(N,D) (D,H) (N,H)



RNN 계층의 계산 그래프(역전파 포함)



copus	0	1	2	3	4	1	5	6
xs	0	1	2	3	4	1	5	
ts	1	2	3	4	1	5	6	

Time Embedding

batch_size	1
time_size	3
max_iters	2
wordvec_size	4

"you say goodbye and i say hello ."

xs	0	1	2
----	---	---	---

 (1,3)

N	1
T	3

out (N,T,D)

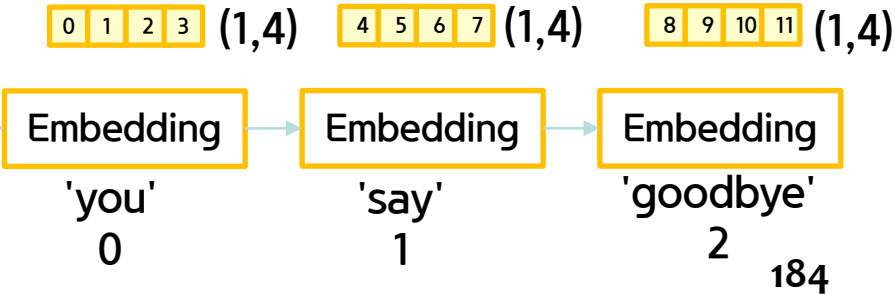
0	1	2	3
4	5	6	7
8	9	10	11

TimeEmbedding
self.params
self.grads
self.layers
self.W

embed_W (V,D)

0	1	2	3
4	5	6	7
8	9	10	11

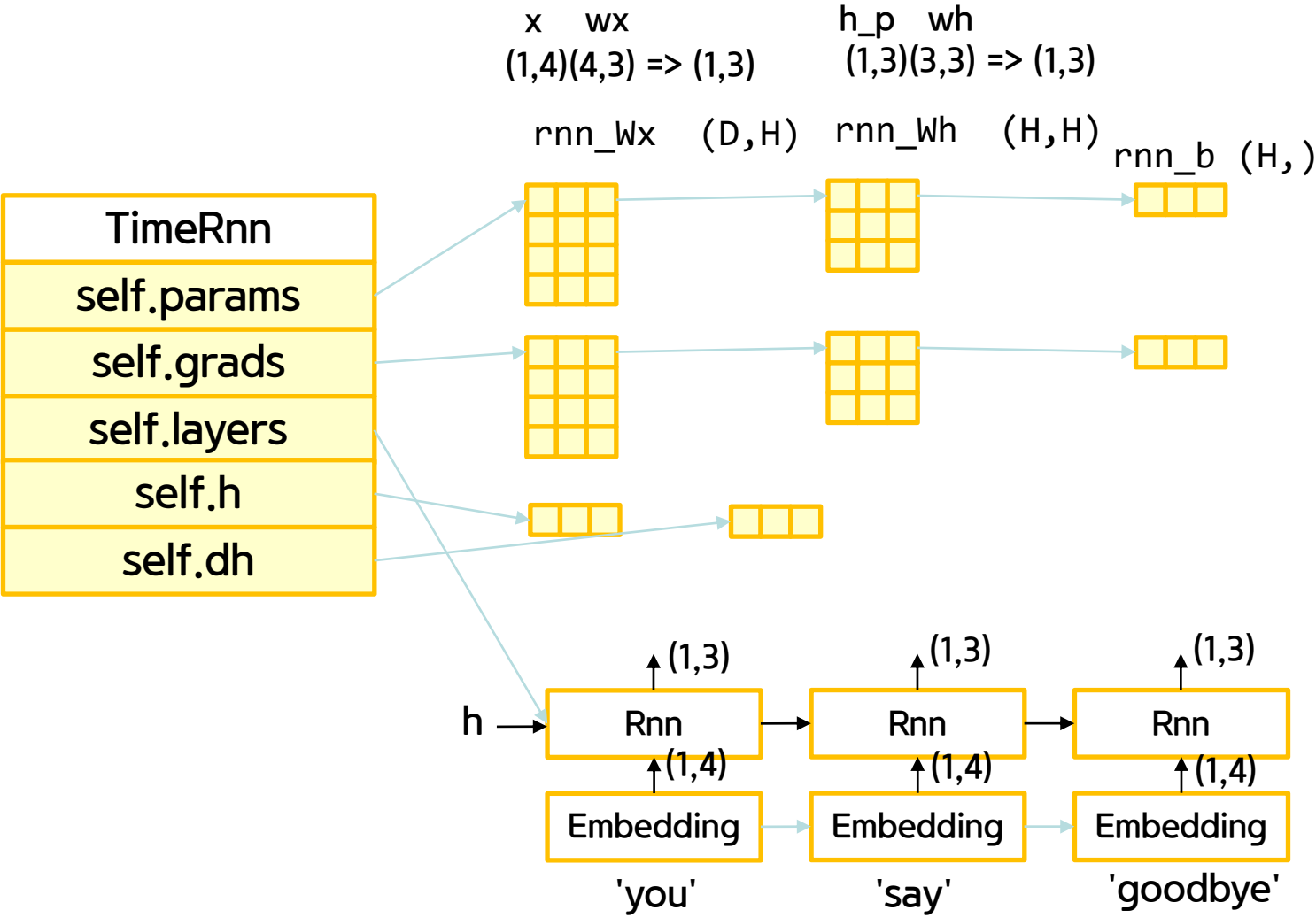
V	7
D	4
H	3

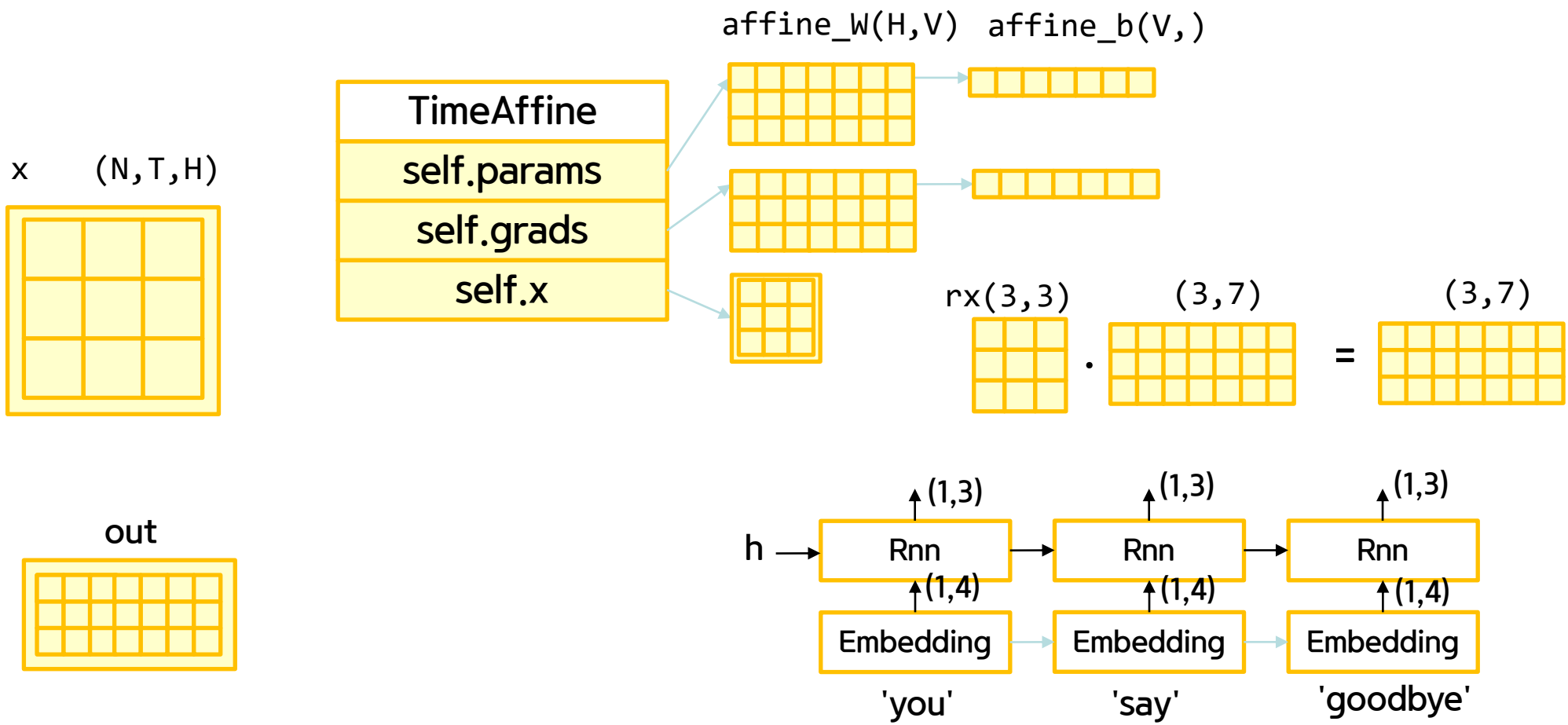


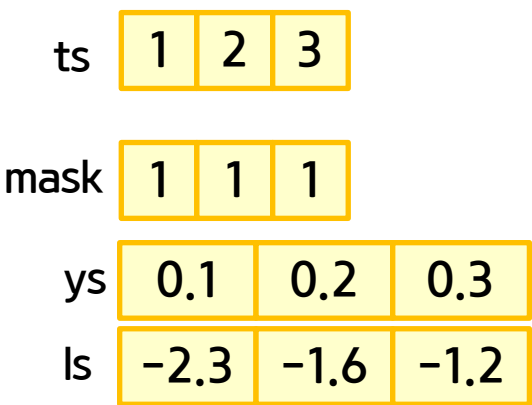
$x_s \quad (N, T, D)$

0	1	2	3
4	5	6	7
8	9	10	11

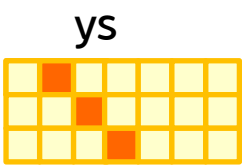
$h_s \quad (N, T, H)$







loss = 5.1
loss = 5.1/3 = 1.7



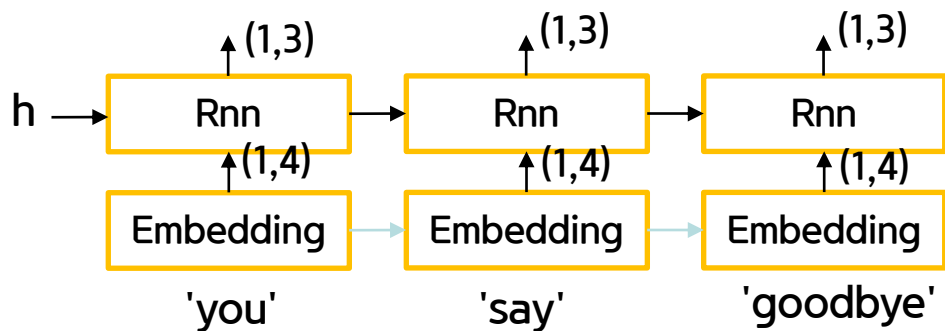
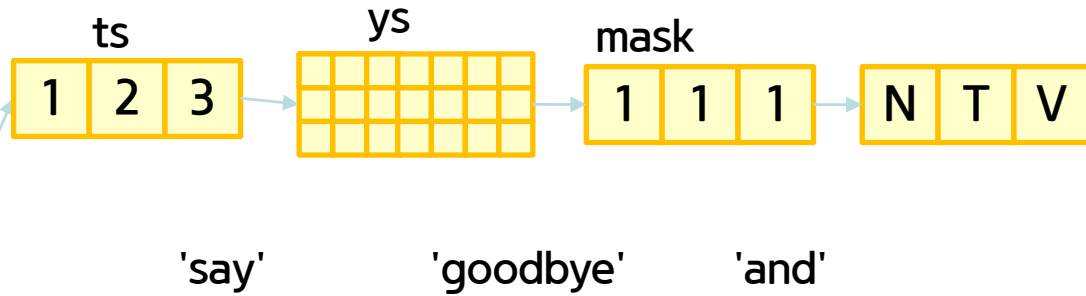
TimeSoftmaxWit

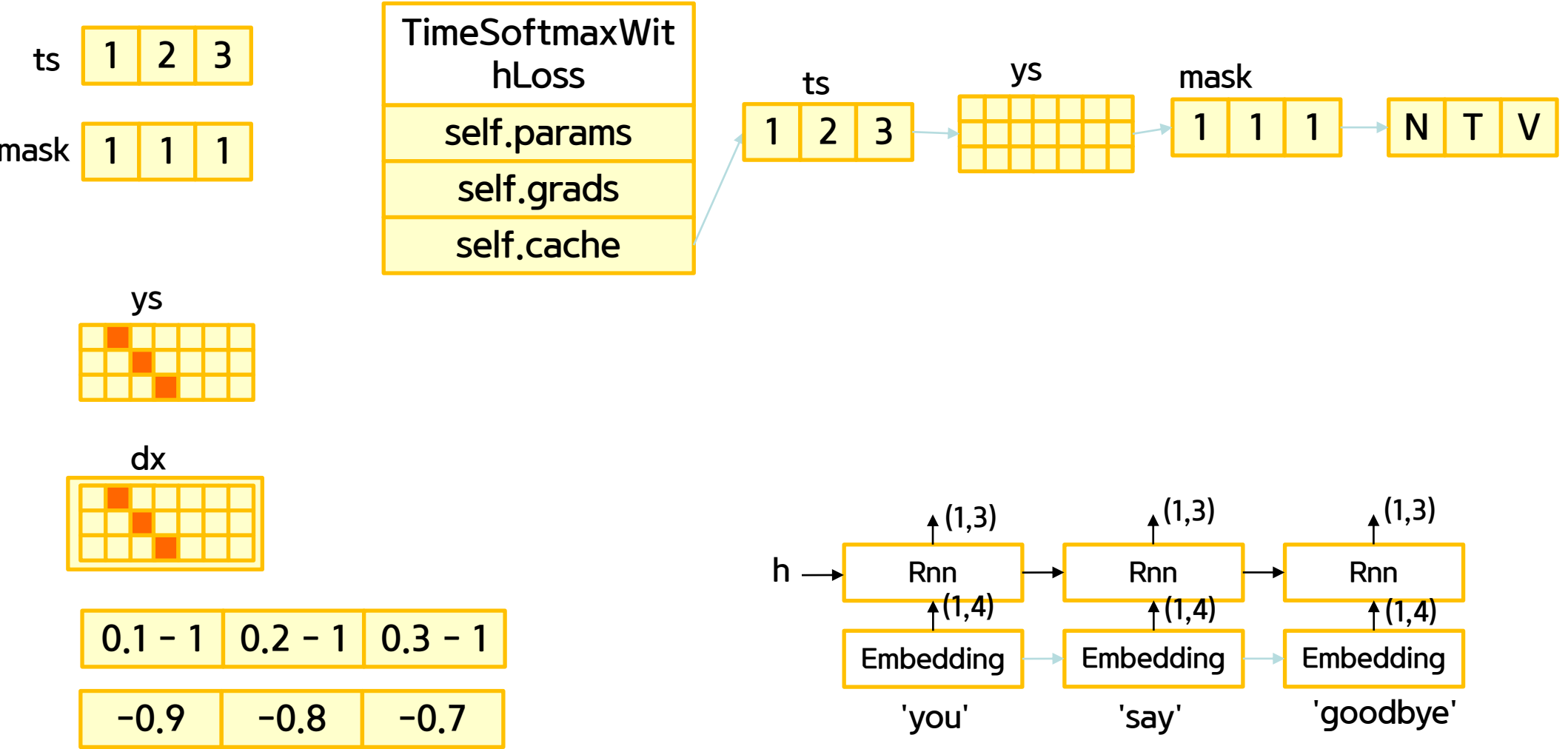
hLoss

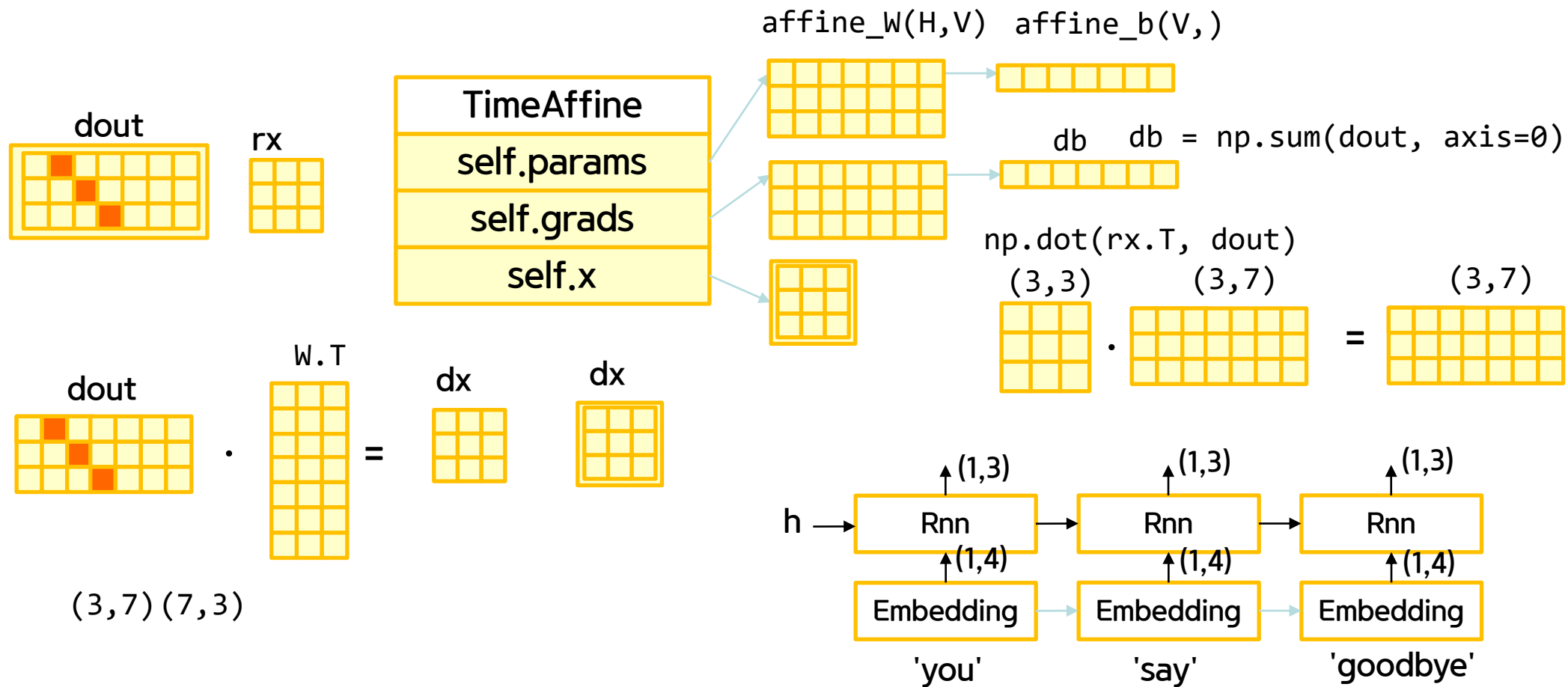
self.params

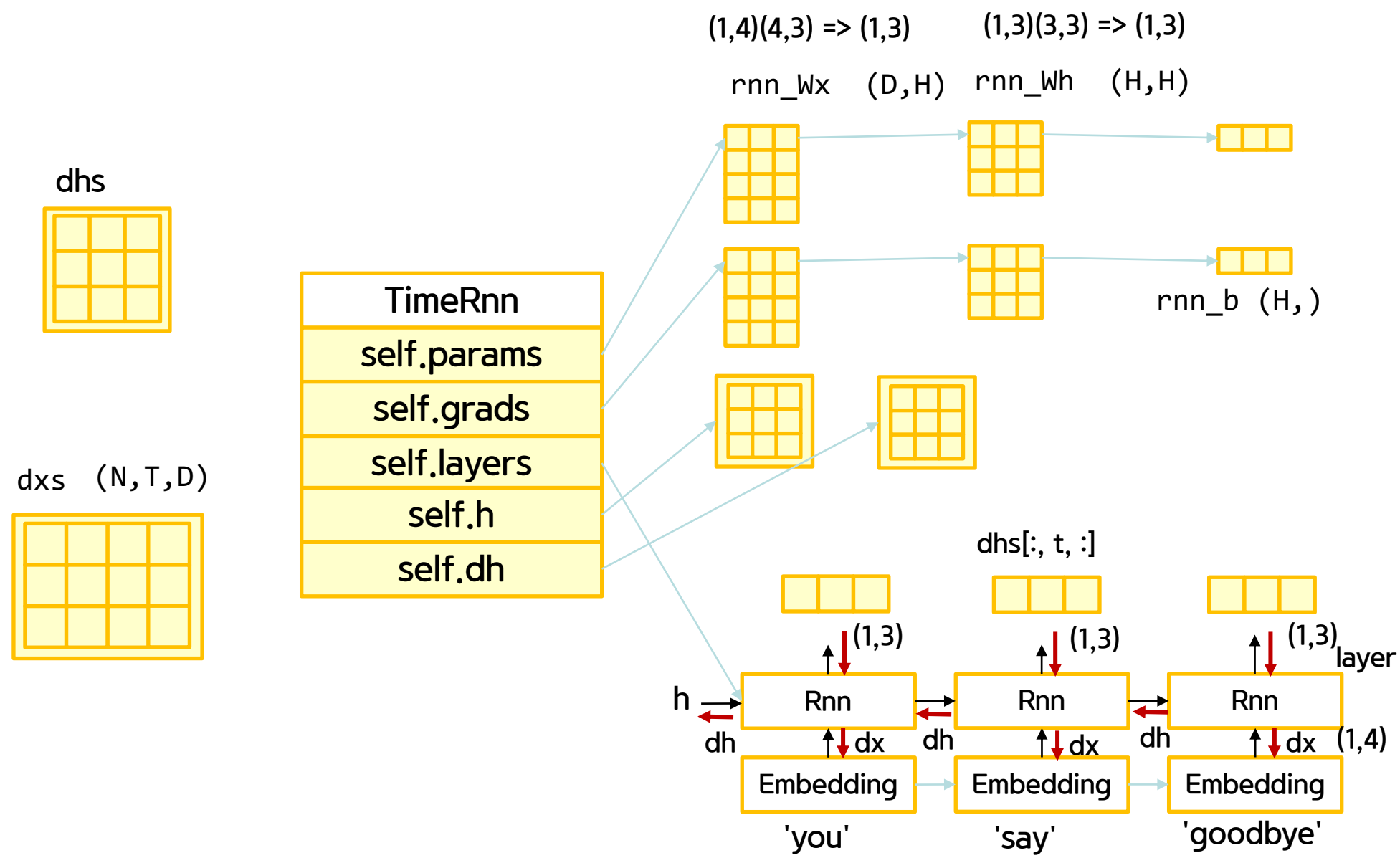
self.grads

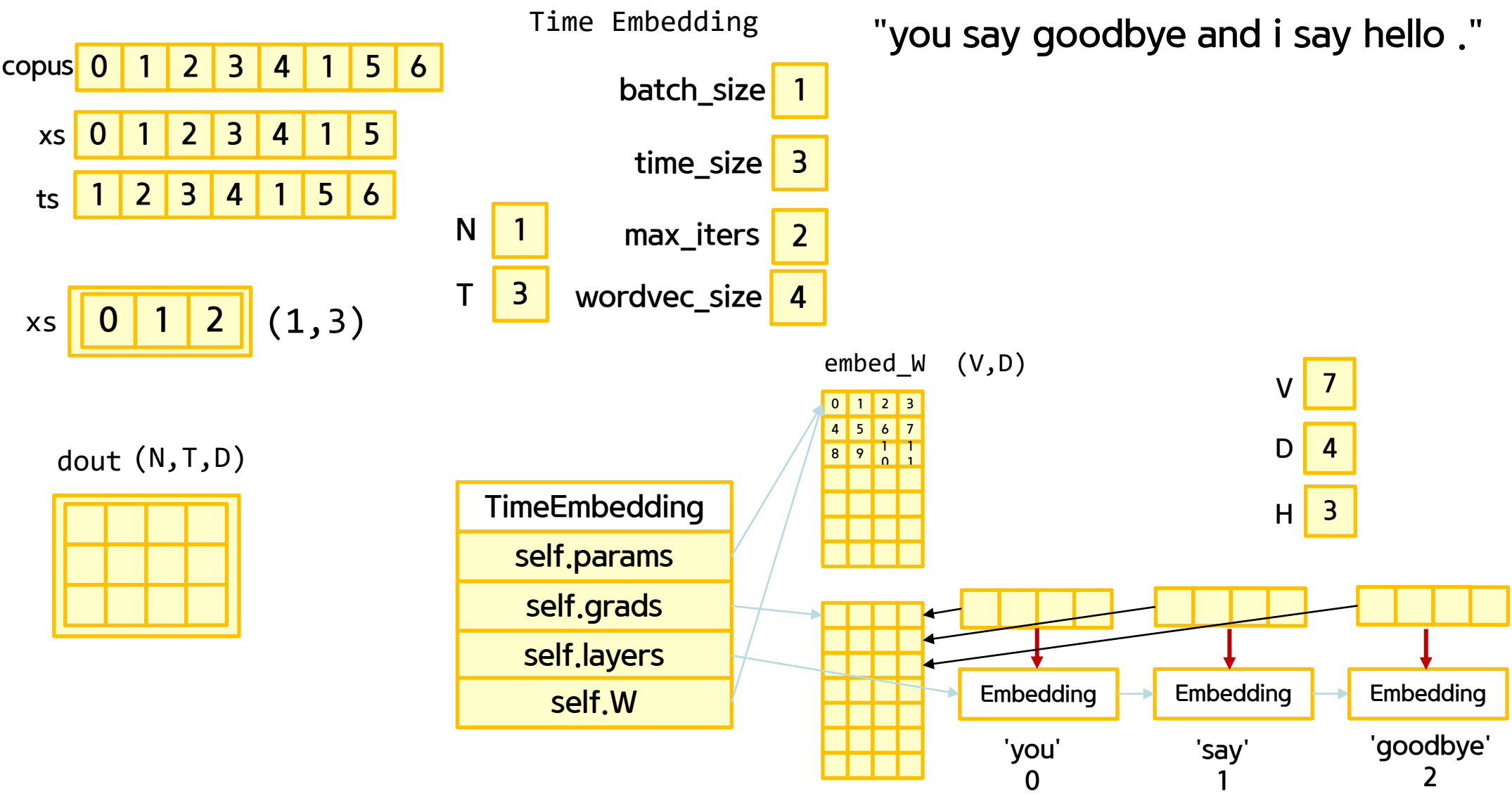
self.cache

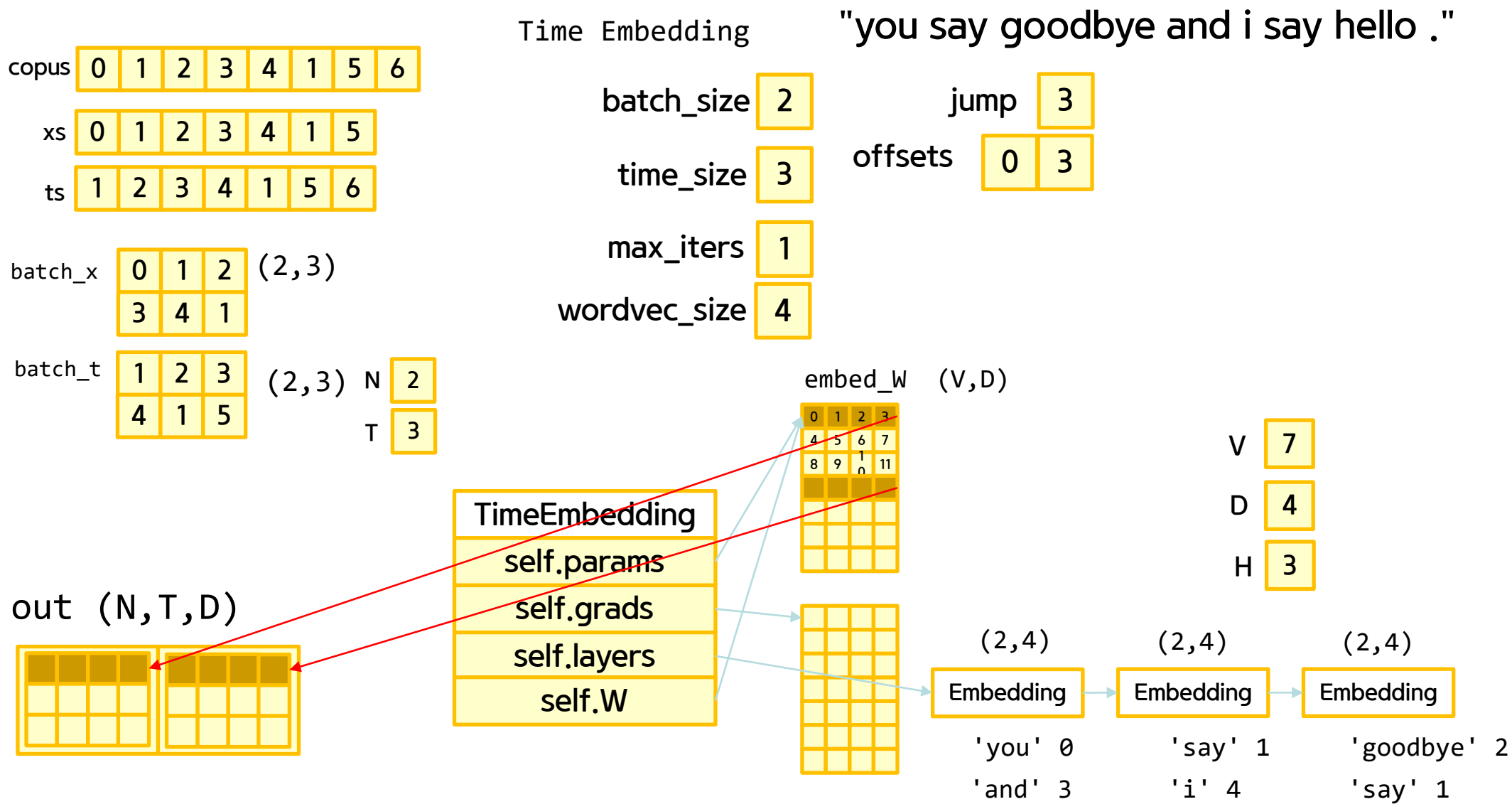




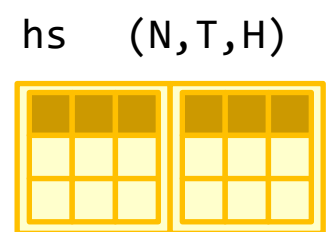
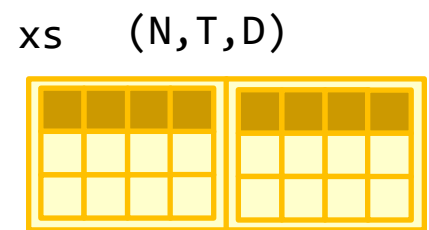








$(2,4)(4,3)+(2,3)(3,3) = (2,3)$



TimeRnn

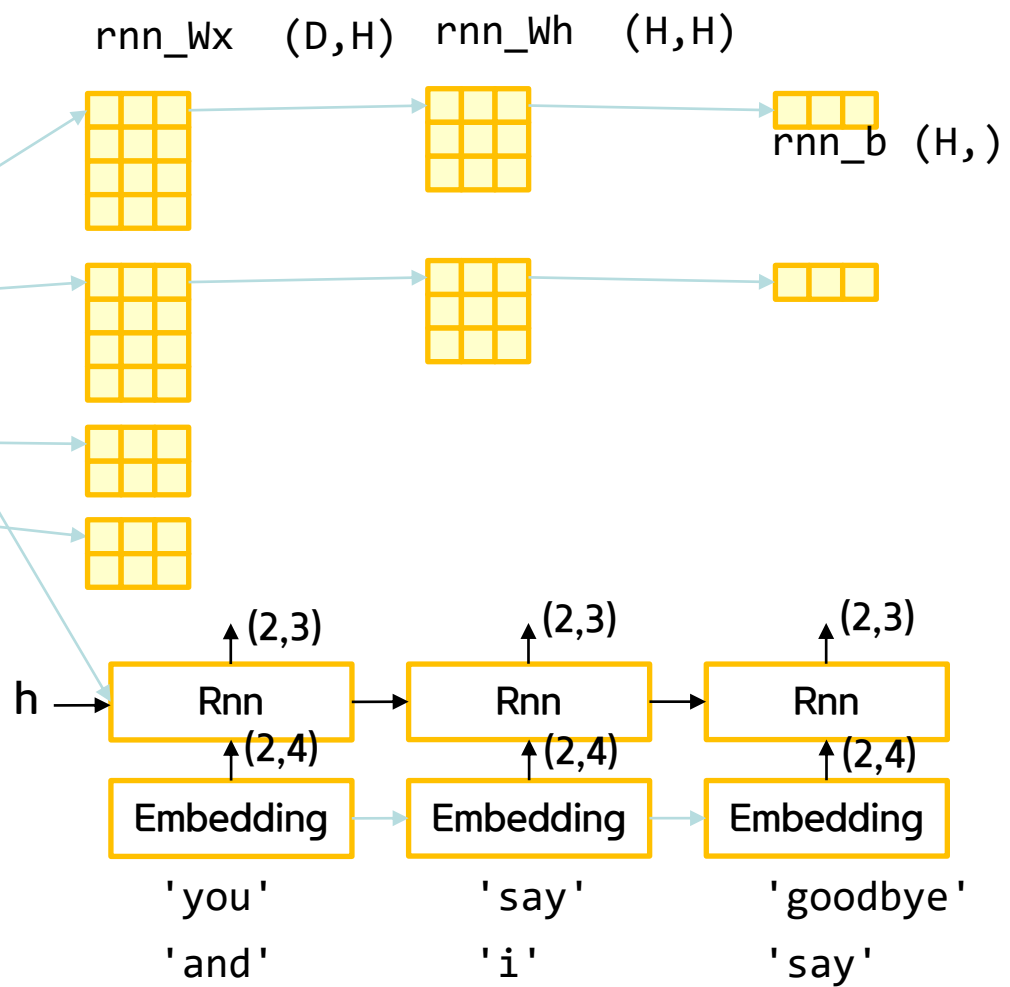
self.params

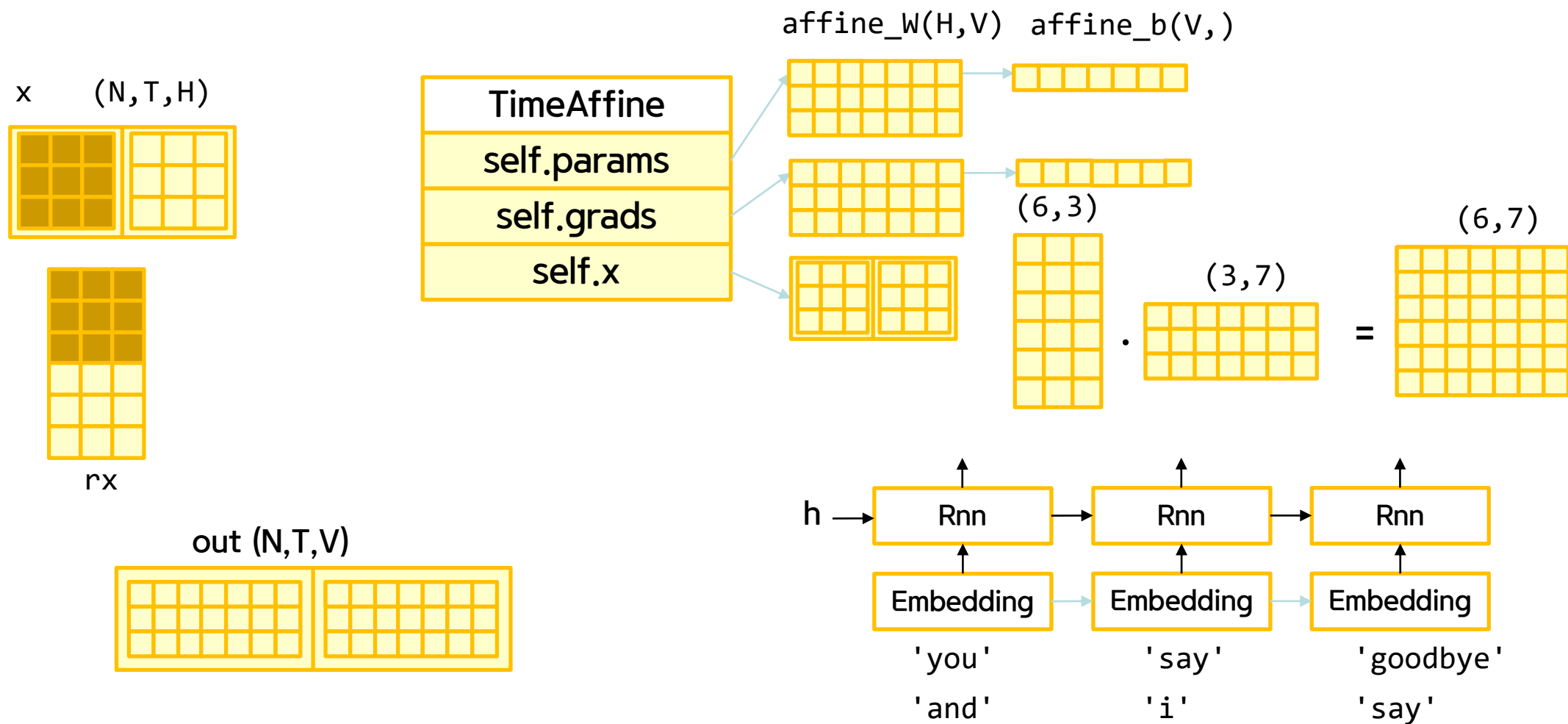
self.grads

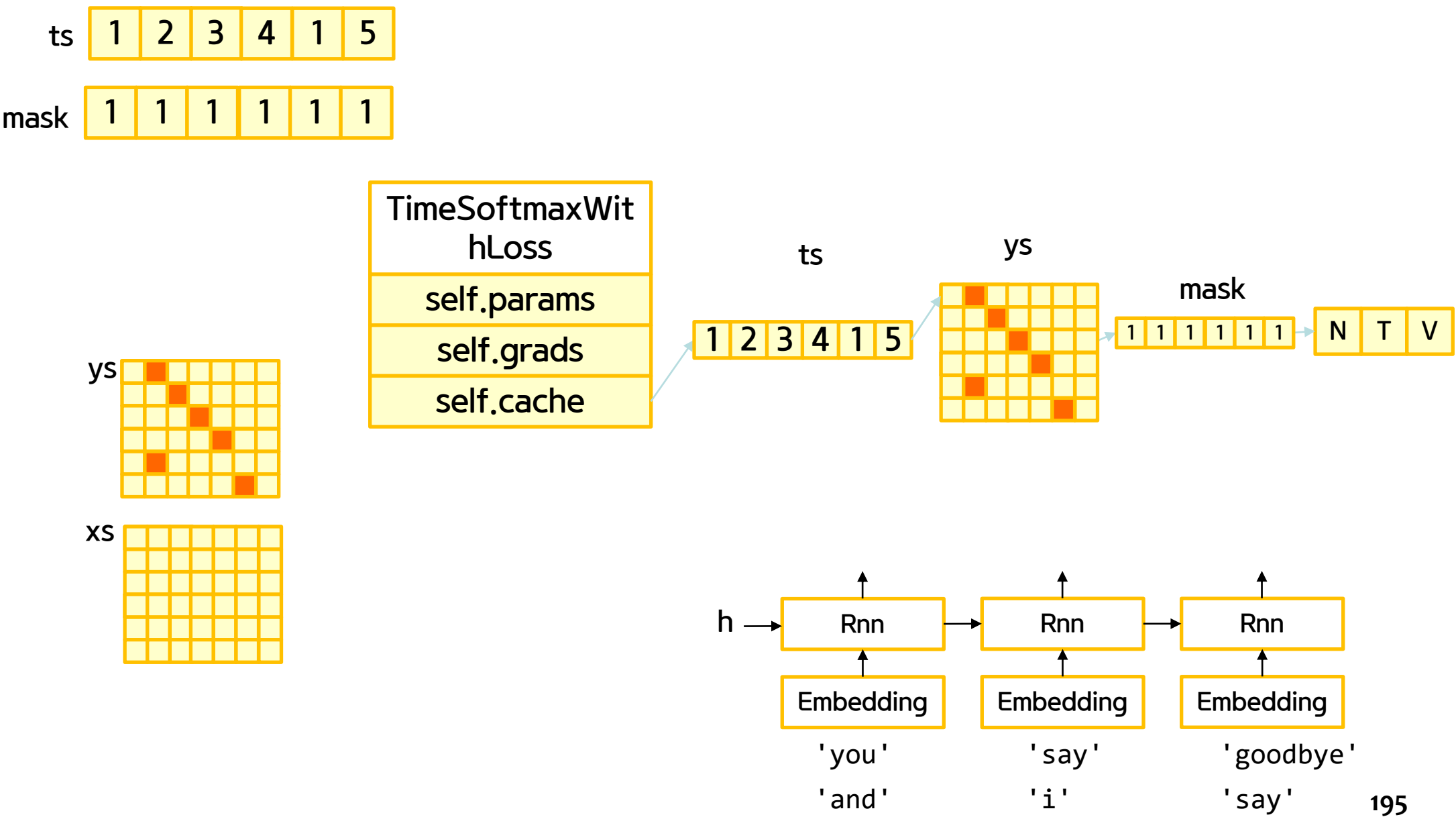
self.layers

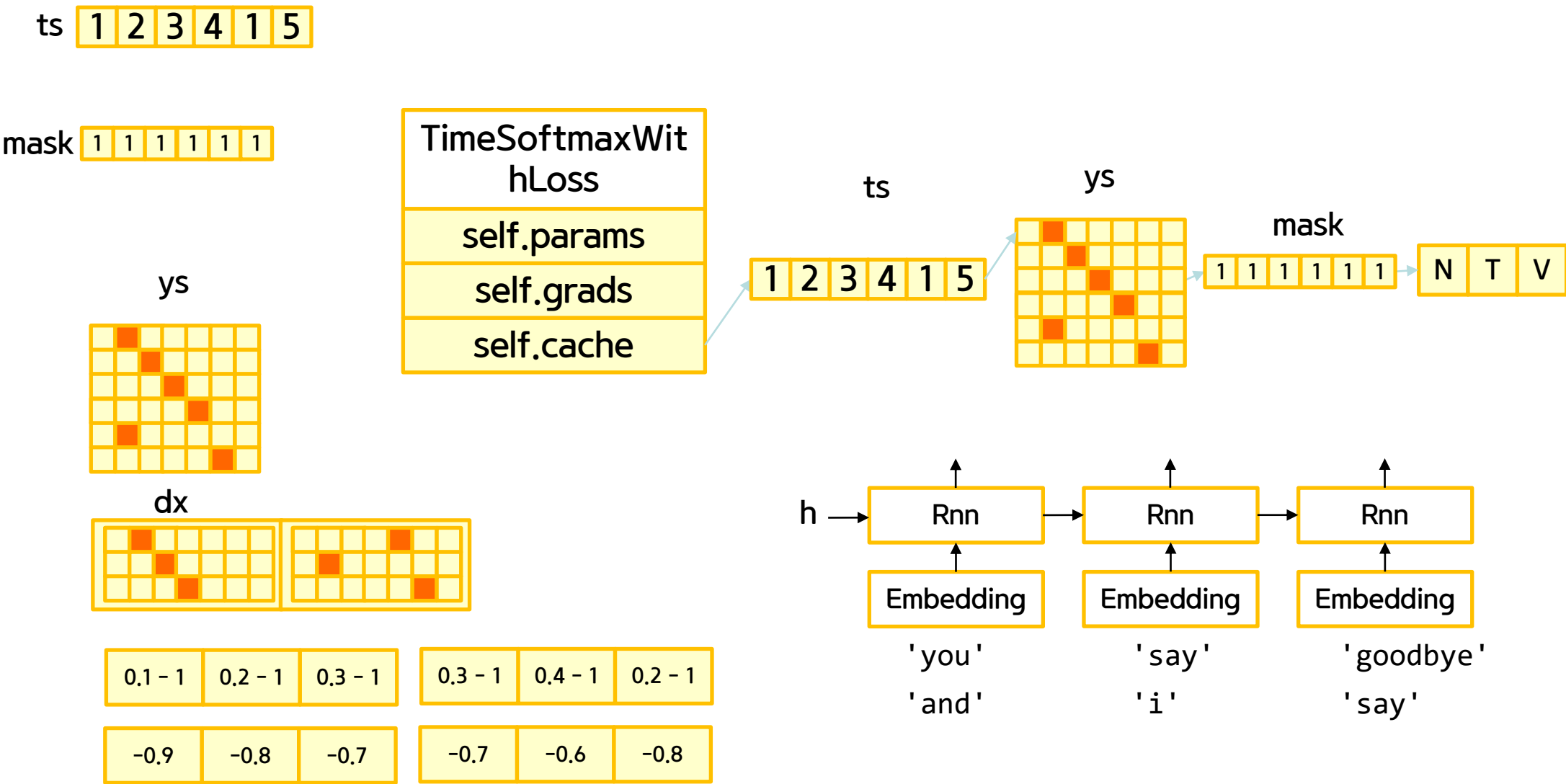
self.h

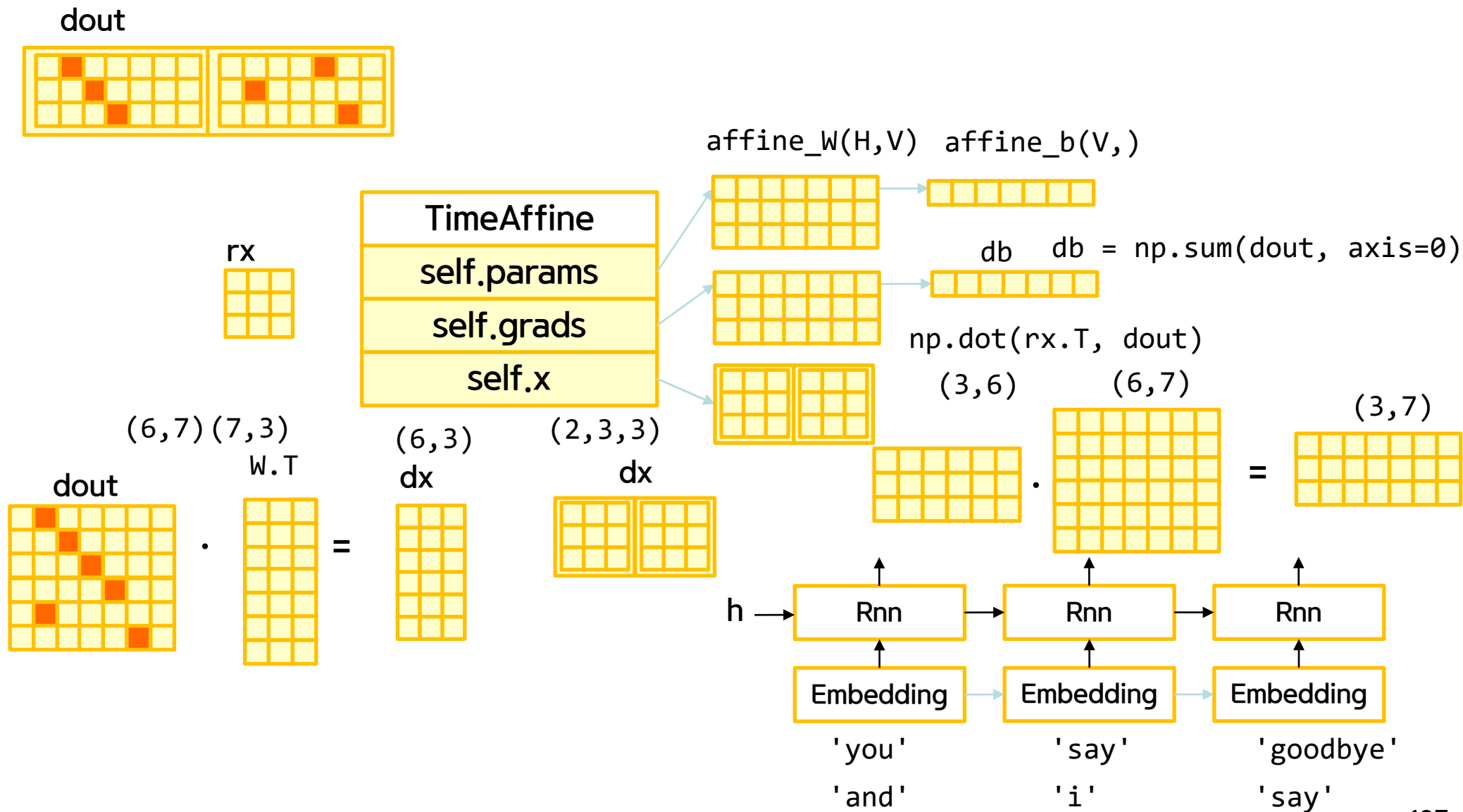
self.dh

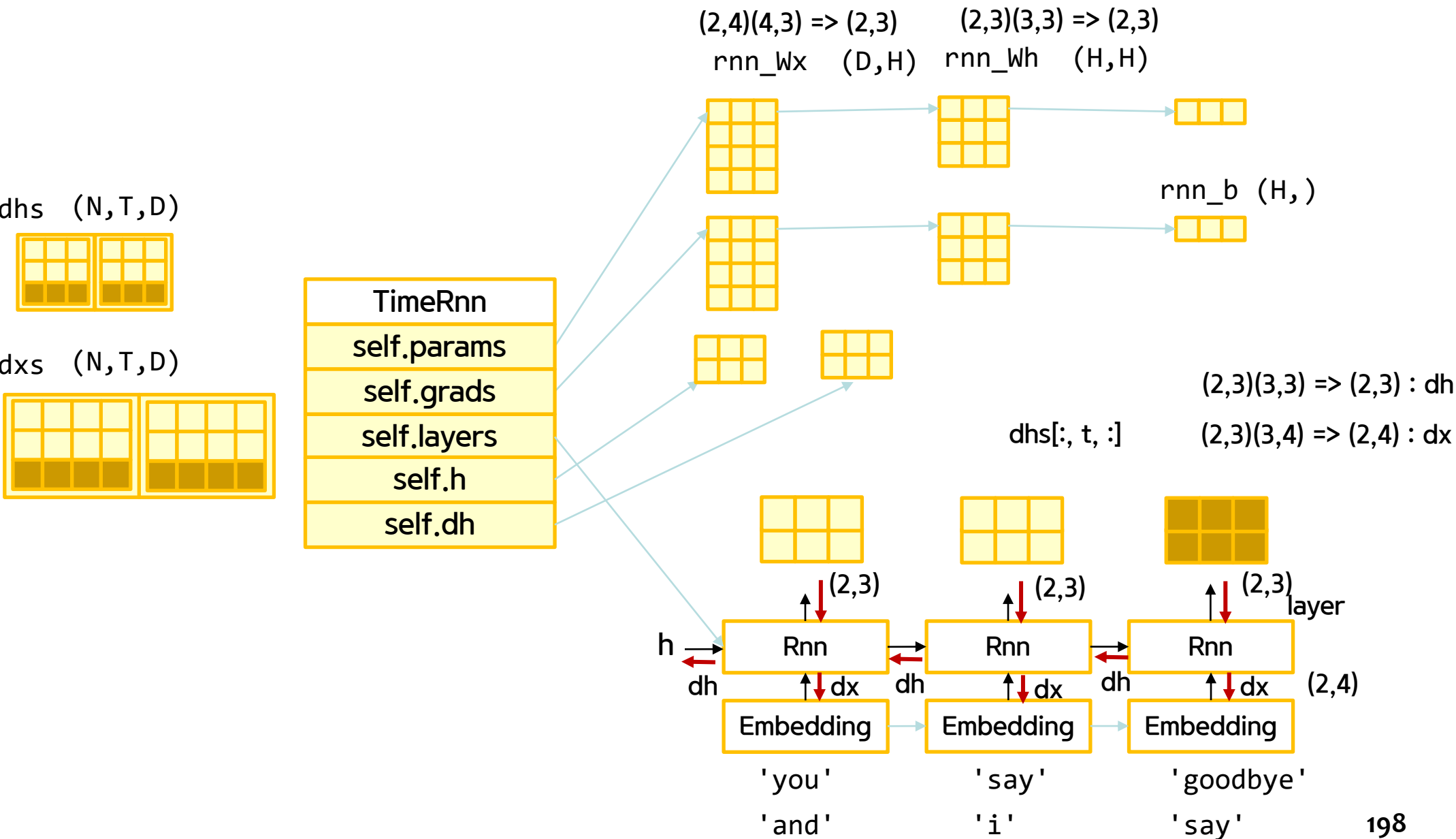


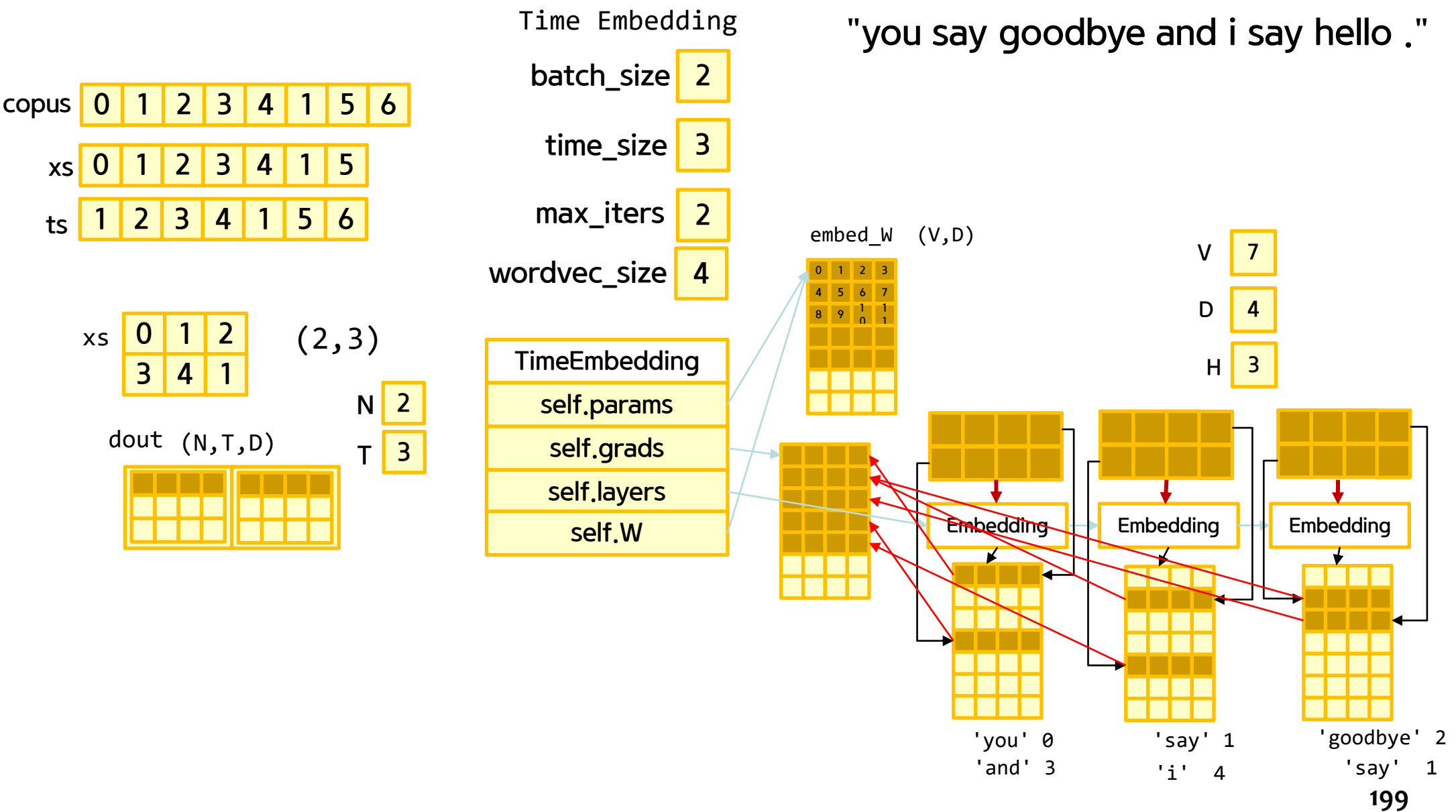






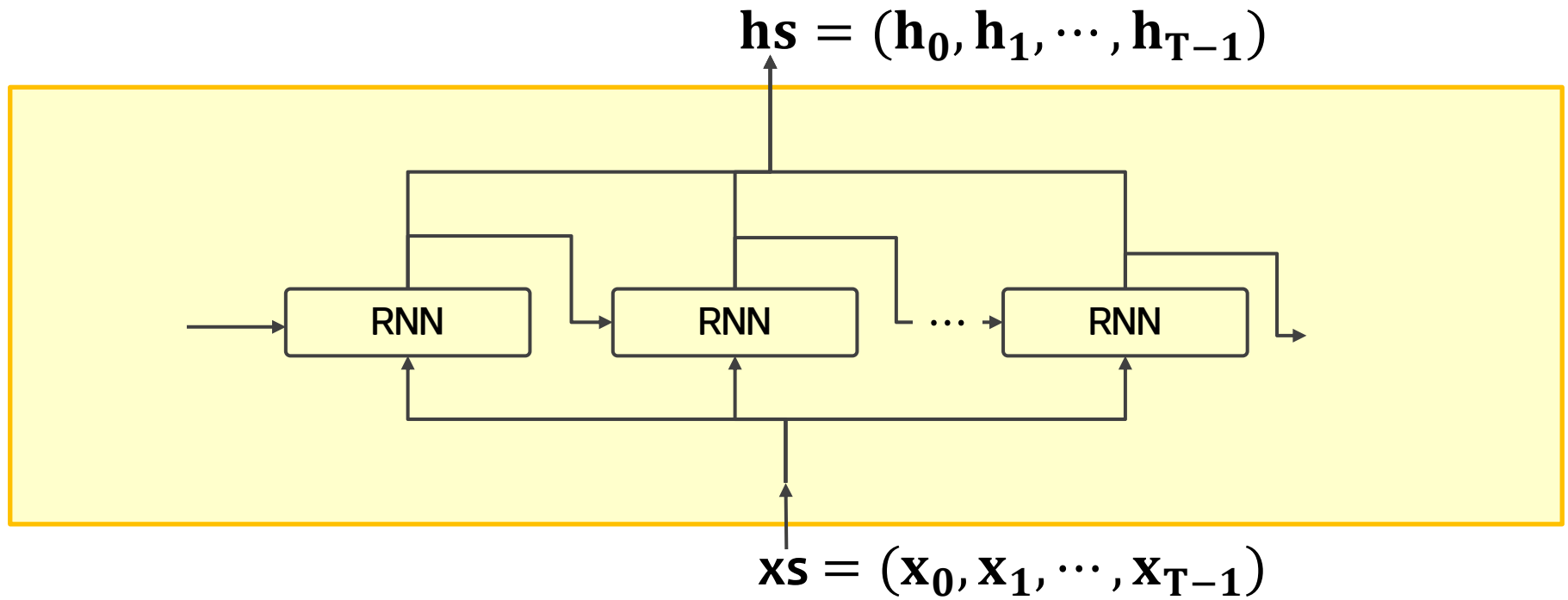




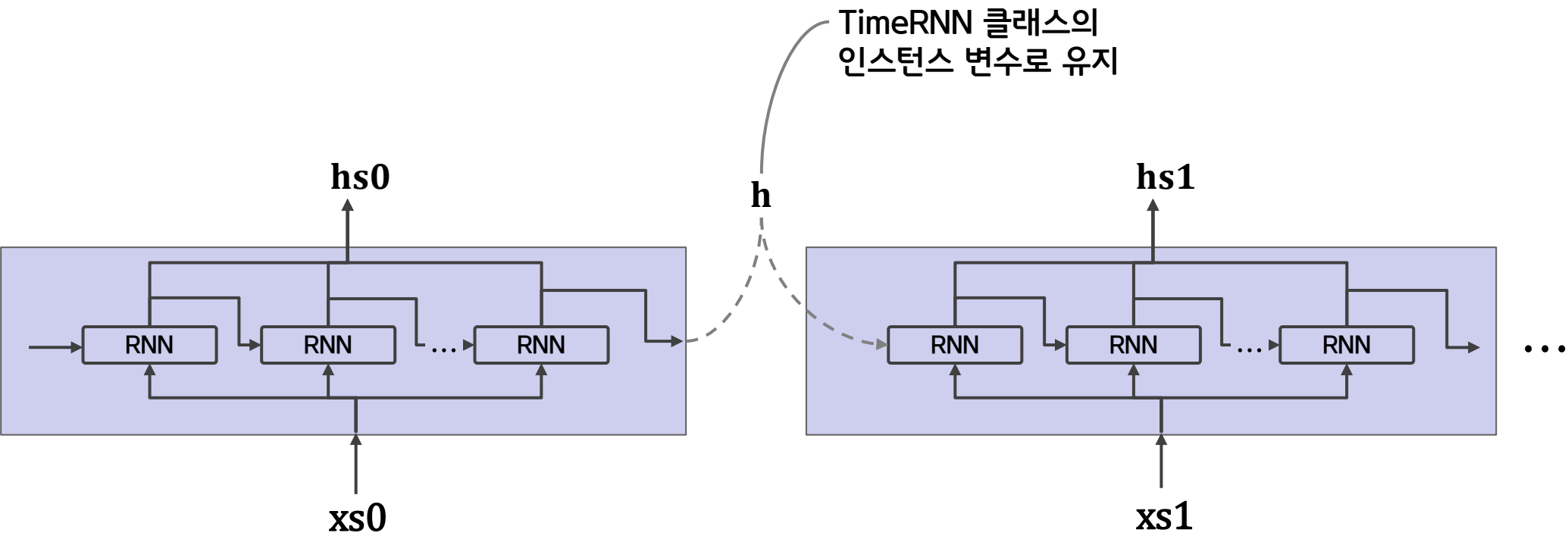


Time RNN 계층의 계산 그래프

Time RNN



Time RNN 계층은 은닉 상태를 인스턴스 변수 h 로 보관한다. 그러면 은닉 상태를 다음 블록에 인계할 수 있다.

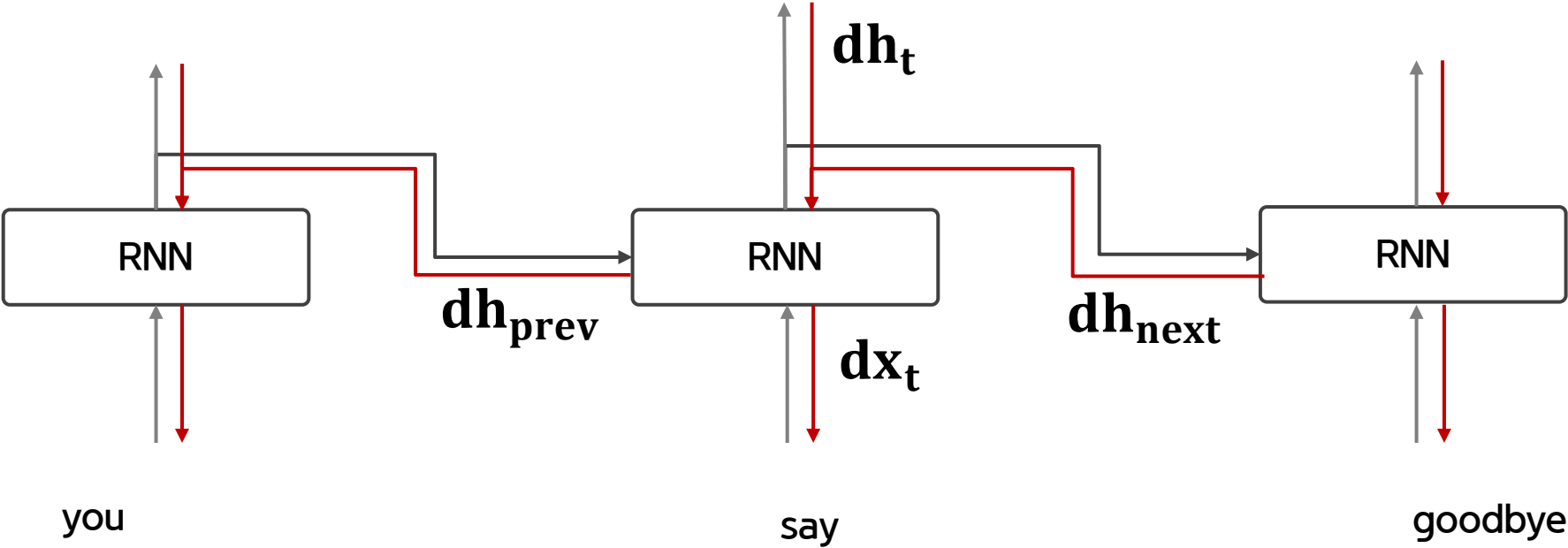


RNN계층의 은닉 상태를 Time RNN계층에서 관리하면 Time RNN사용자는 RNN계층 사이에서 은닉 상태를 '인계하는 작업'을 생각하지 않아도 된다.

The diagram illustrates the TimeRNN architecture. A sequence of input vectors $\mathbf{x}_s = (x_0, x_1, \dots, x_{T-1})$ is fed into a sequence of Recurrent Neural Network (RNN) blocks. Each RNN block takes an input x_t and the previous hidden state h_{t-1} to produce a new hidden state h_t . The sequence of hidden states is denoted as $\mathbf{h}_s = (h_0, h_1, \dots, h_{T-1})$. A feedback loop is shown where the differential hidden state dh (the difference between consecutive hidden states) is used as an additional input to the RNN blocks. The entire sequence of RNN blocks is enclosed in a light blue box. A dashed line points to the dh label with the text "TimeRNN 클래스의 인스턴스 변수로 유지" (Maintained as an instance variable of the TimeRNN class).

t번째 RNN 계층의 역전파

Time RNN



5. 순환 신경망(RNN)

5.1 확률과 언어 모델

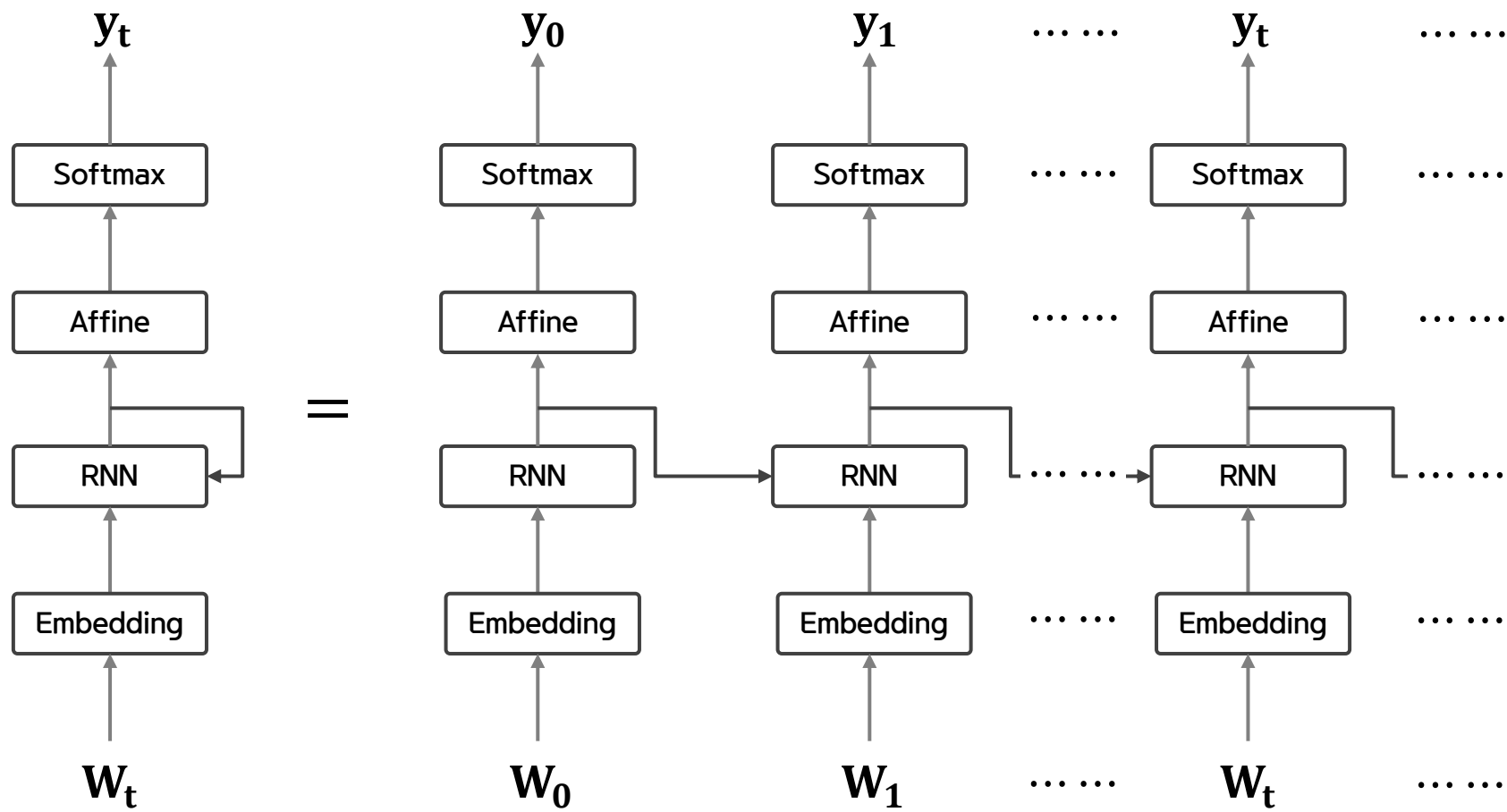
5.2 RNN 이란

5.3 RNN 구현

5.4 시계열 데이터 처리 계층 구현

5.5 RNNLM 학습과 평가

RNNLM의 신경망



RNNLM의 신경망

Embedding: 단어 ID를 단어의 분산 표현(단어 벡터)으로 변환

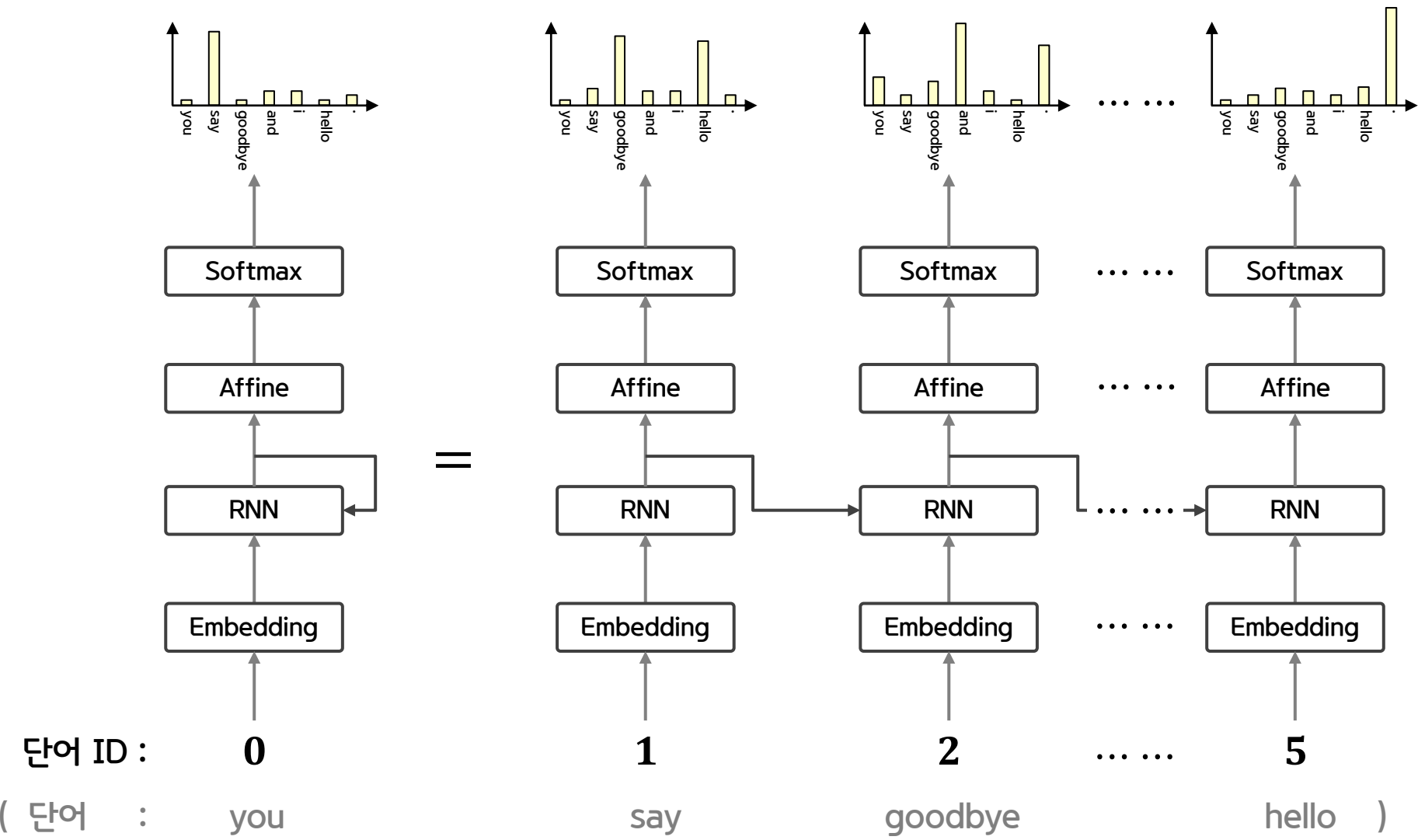
RNN계층: 은닉 상태를 다음 층으로(위쪽으로) 출력함과 동시에 다음 시각의 RNN 계층으로(오른쪽으로) 출력한다.

RNN계층이 위로 출력한 은닉 상태는 Affine 계층을 거쳐 Softmax 계층으로 전해진다.

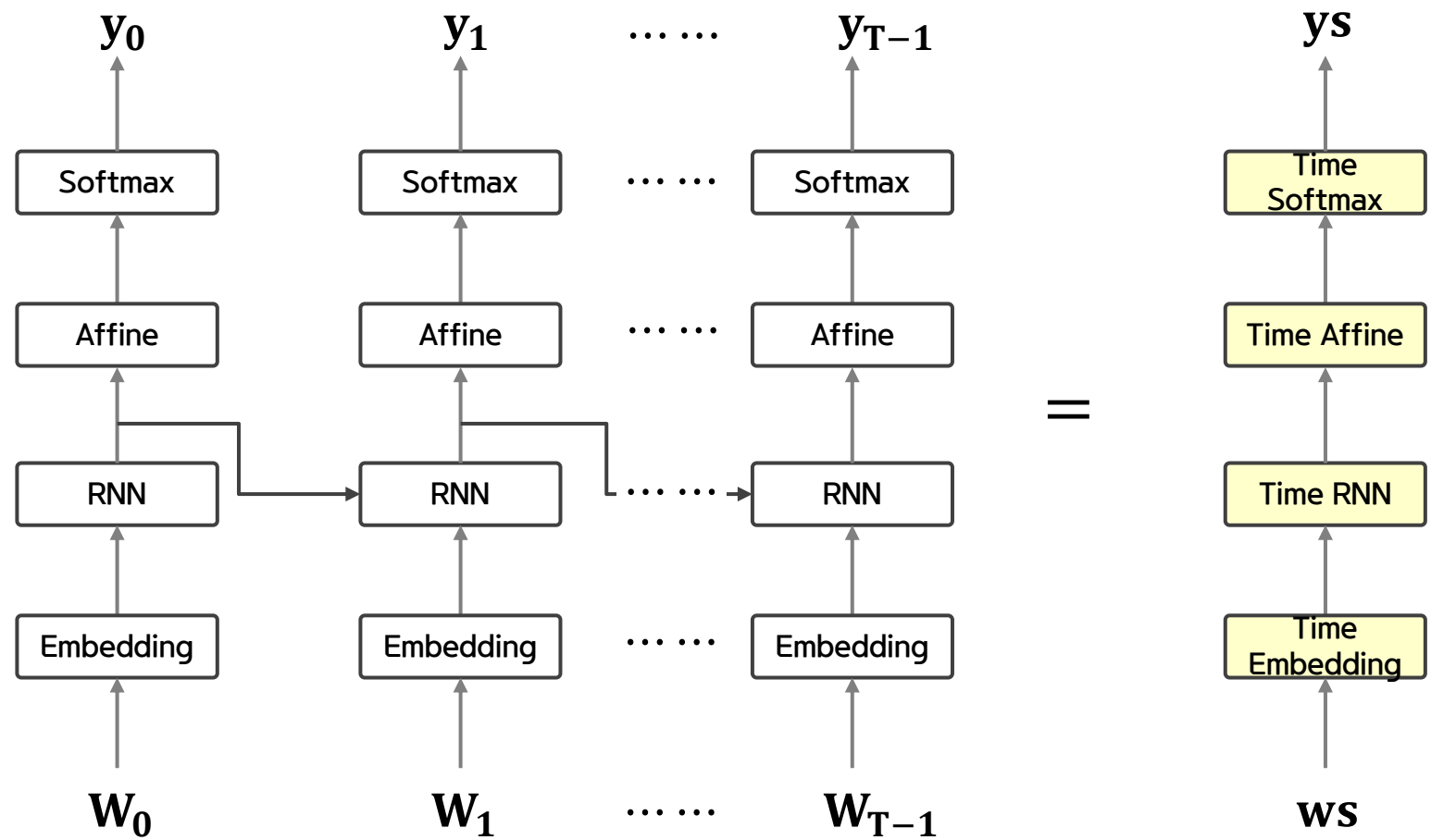
RNNLM은 지금까지 입력된 단어를 '기억'하고 그것을 바탕으로 다음에 출현할 단어를 예측한다.

RNN계층이 과거에서 현재로 데이터를 계속 흘려 보내 줌으로써 과거의 정보를 인코딩해 저장(기억) 할 수 있다.

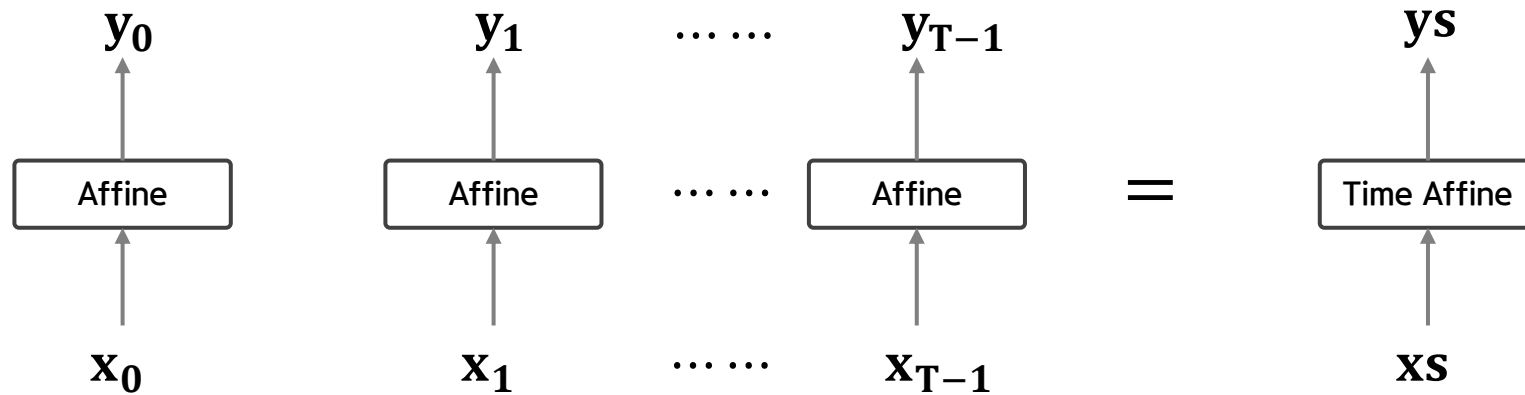
샘플 말뭉치로 "you say goodbye and i say hello."를 처리하는 RNNLM의 예



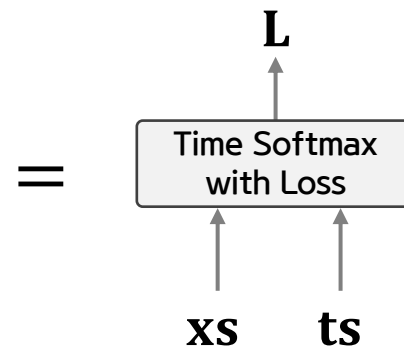
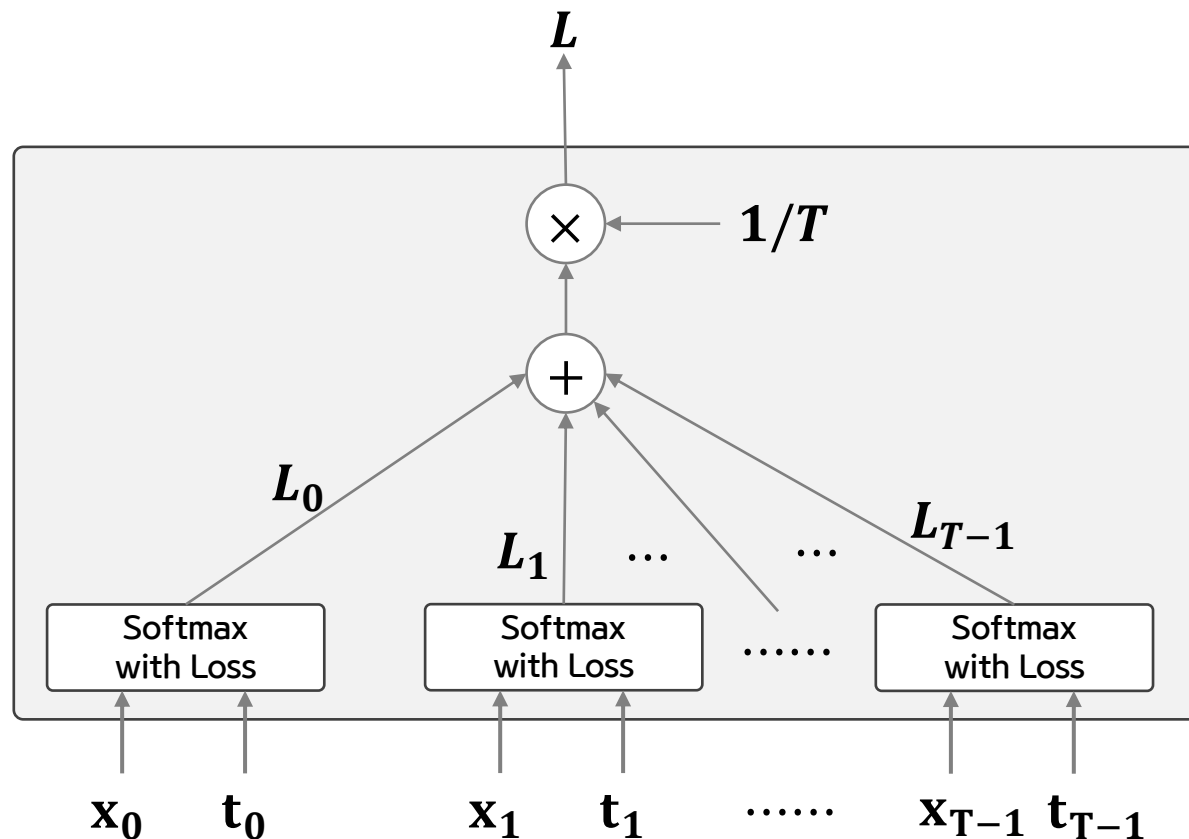
시계열 데이터를 한꺼번에 처리하는 계층을 Time XX 계층으로 구현



Time Affine 계층은 T개의 Affine 계층의 집합으로 구현



Time Softmax with Loss 계층의 전체 그림



x_0, x_1, \dots : 아래층에서부터 전해지는 점수(확률로 정규화 되기 전의 값)

t_0, t_1, \dots : 정답 레이블

T개의 Softmax with Loss 계층 각각이 손실을 산출하고 그 손실들을 합산해 평균한 값이 최종 손실이 된다.

$$L = \frac{1}{T} (L_0 + L_1 + \cdots + L_{T-1})$$

Time Softmax with Loss 계층도 시계열에 대한 평균을 구하는 것으로 데이터 1개당 평균 손실을 구해 최종 출력으로 내보낸다.

5. 순환 신경망(RNN)

5.1 확률과 언어 모델

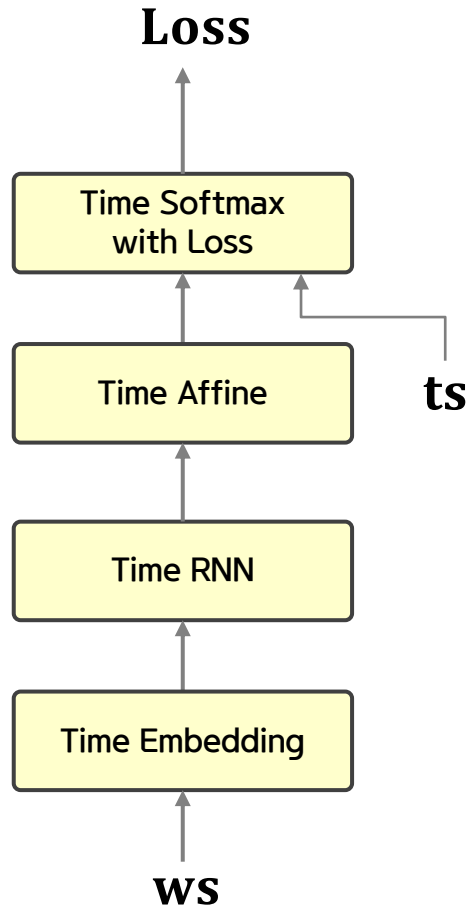
5.2 RNN 이란

5.3 RNN 구현

5.4 시계열 데이터 처리 계층 구현

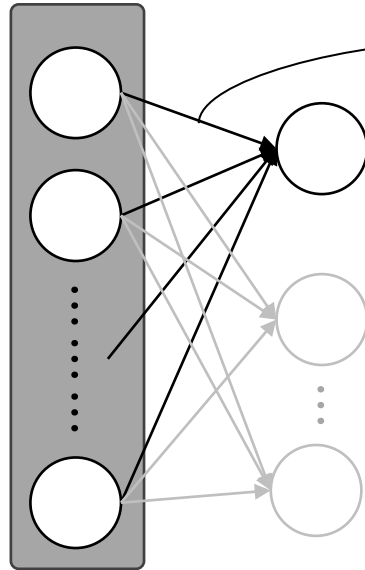
5.5 RNNLM 학습과 평가

SimpleRnnlm의 계층 구성 : RNN 계층의 상태는 클래스 내부에서 관리



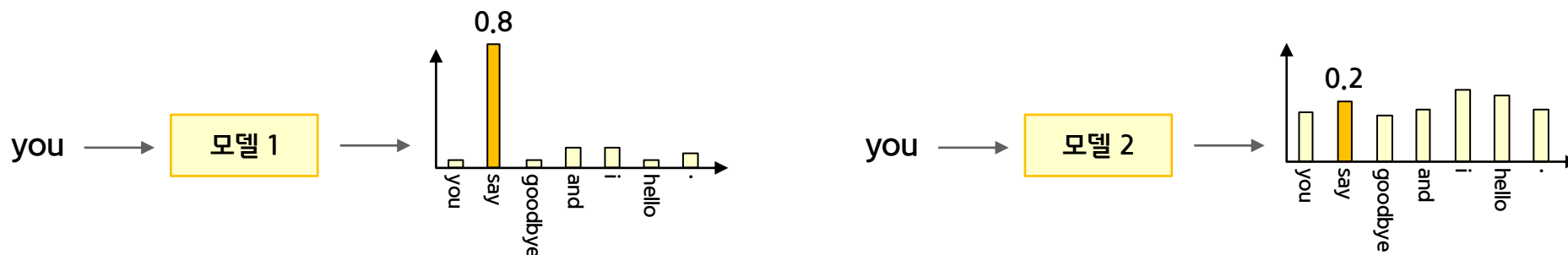
Xavier 초기값 : 이전 계층의 노드가 n 개라면 표준편차가 $\frac{1}{\sqrt{n}}$ 인 분포를 초기값으로 사용

n 개의 노드



표준편차가 $\frac{1}{\sqrt{n}}$ 인 분포로 초기화

단어 "you"를 입력하여 다음에 출현할 단어의 확률분포를 출력하는 모델의 예



언어 모델은 주어진 과거 단어(정보)로부터 다음에 출현할 단어의 확률분포를 출력한다.
이때 언어 모델의 예측 성능을 평가하는 척도로 혼란도(perplexity)를 자주 이용한다.

혼란도(perplexity) : 간단히 말하면 '확률의 역수'이다.(데이터 수가 하나일 때에 정확히 일치한다.)
작을수록 좋은 값이다.

분기수(number of branches): 다음에 취할 수 있는 선택사항의 수(다음에 출현할 수 있는 단어의 후보 수)

예)

분기수가 1.25 -> 다음에 출현할 수 있는 단어의 후보를 1개 정도로 좁혔다(좋은 모델)

분기수가 5 -> 후보가 아직 5개(나쁜 모델)

입력 데이터가 여러 개일 때

$$L = -\frac{1}{N} \sum_n \sum_k t_{nk} \log y_{nk}$$

$$\textit{perplexity} = e^L$$

N:데이터의 총 개수

t_n : 원 핫 벡터로 나타낸 정답 레이블

t_{nk} : n개째 데이터의 k번째 값

y_{nk} : 확률분포(신경망에서는 Softmax의 출력)

L: 신경망의 손실. 교차 엔트로피 오차를 뜻하는 식과 같은 식

※ 코드 참조

※ 코드 참조