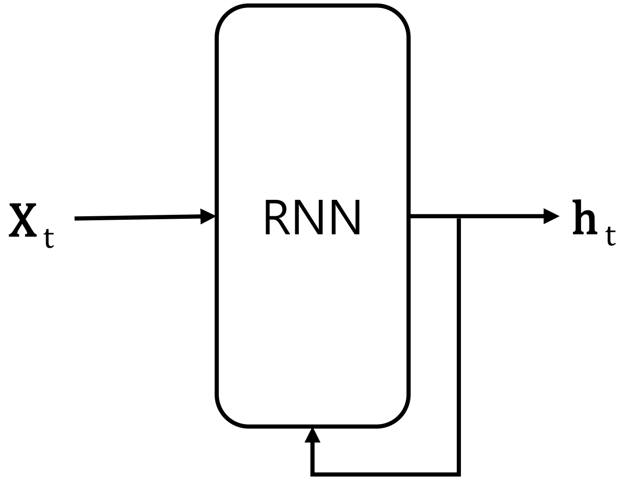
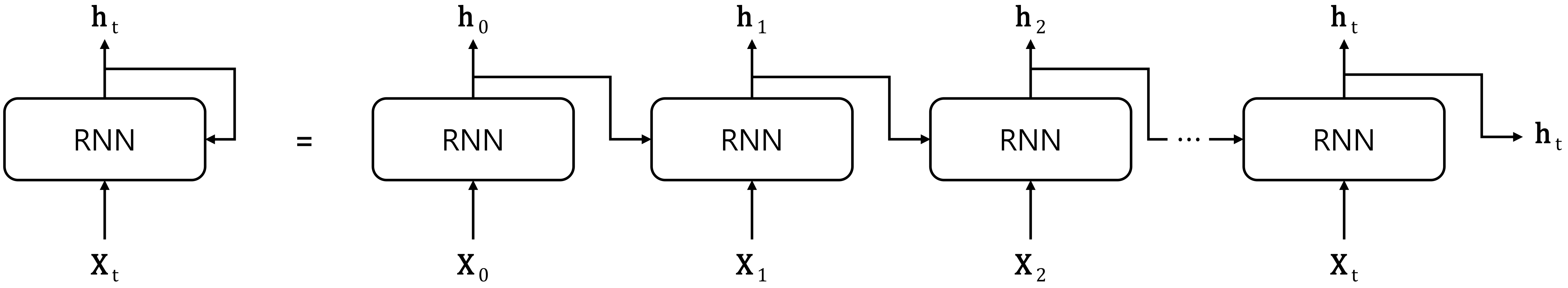
sh951011@gmail.com
https://github.com/sh951011

RNN

Recurrent Neural Network (RNN)

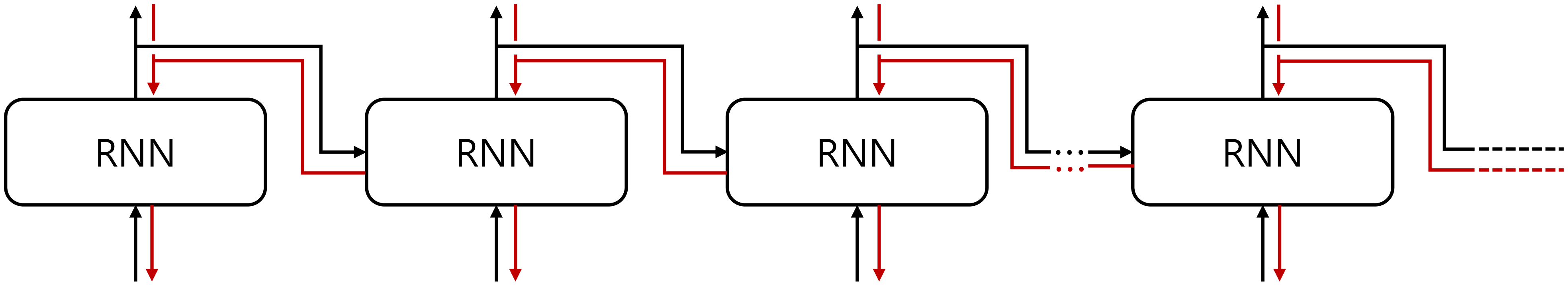


RNN 계층의 순환 구조 펼치기



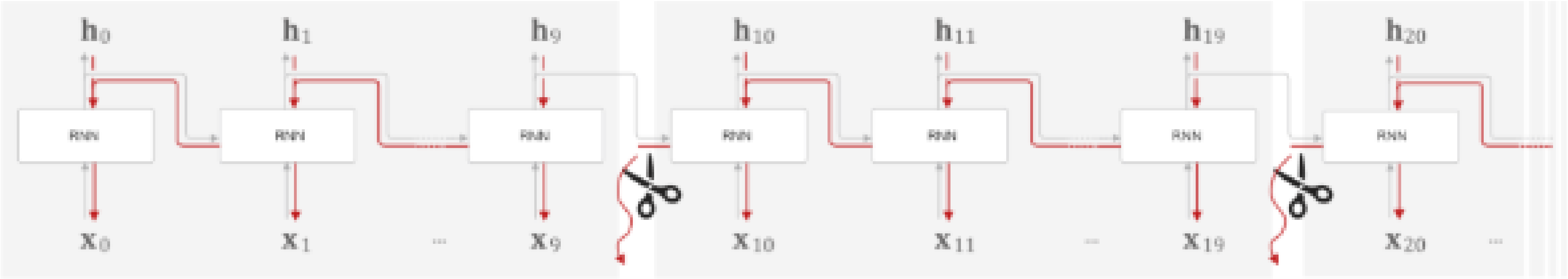
Recurrent Neural Network (RNN)

BPTT (Backpropagation Through Time)



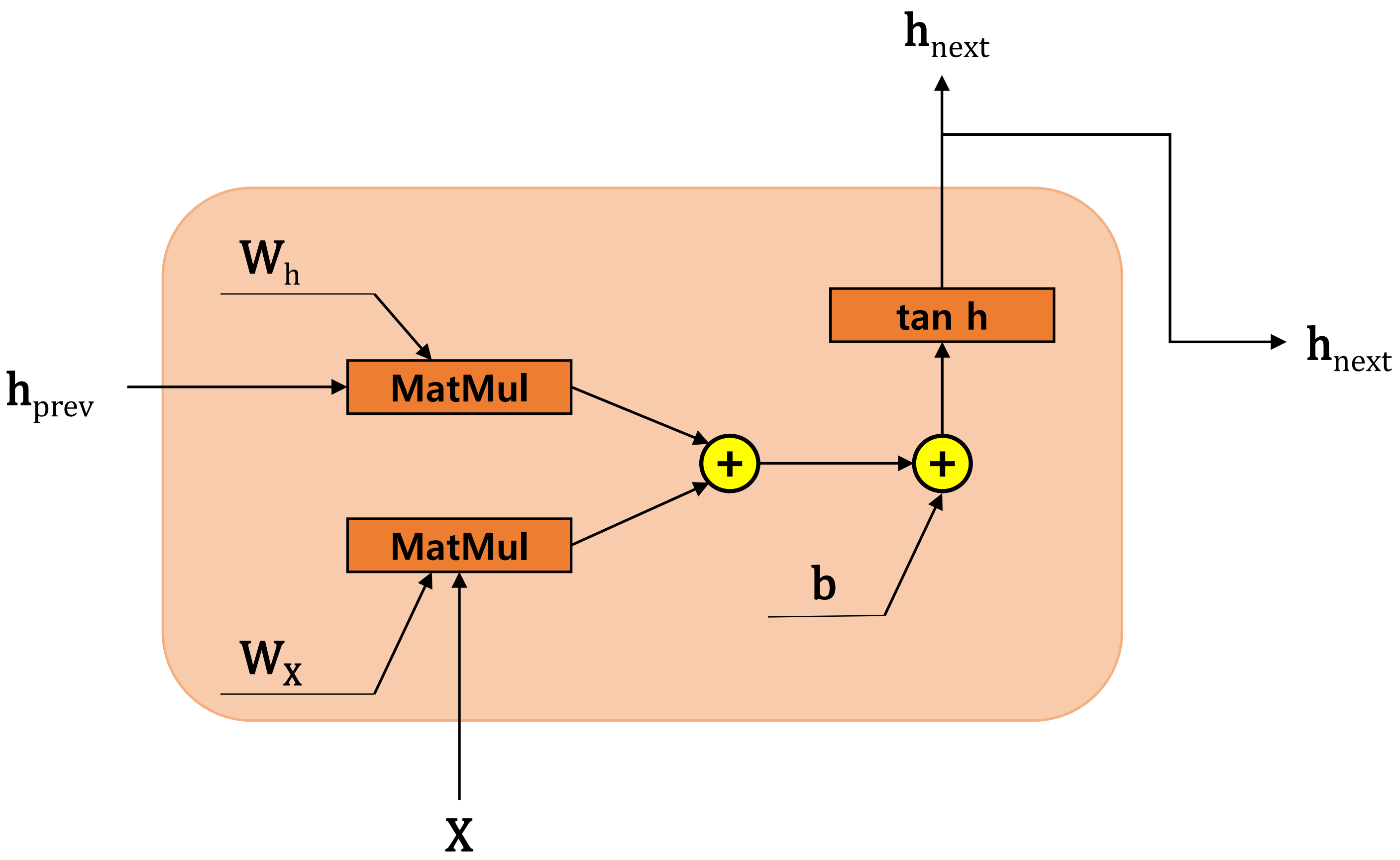
Recurrent Neural Network (RNN)

Truncated BPTT



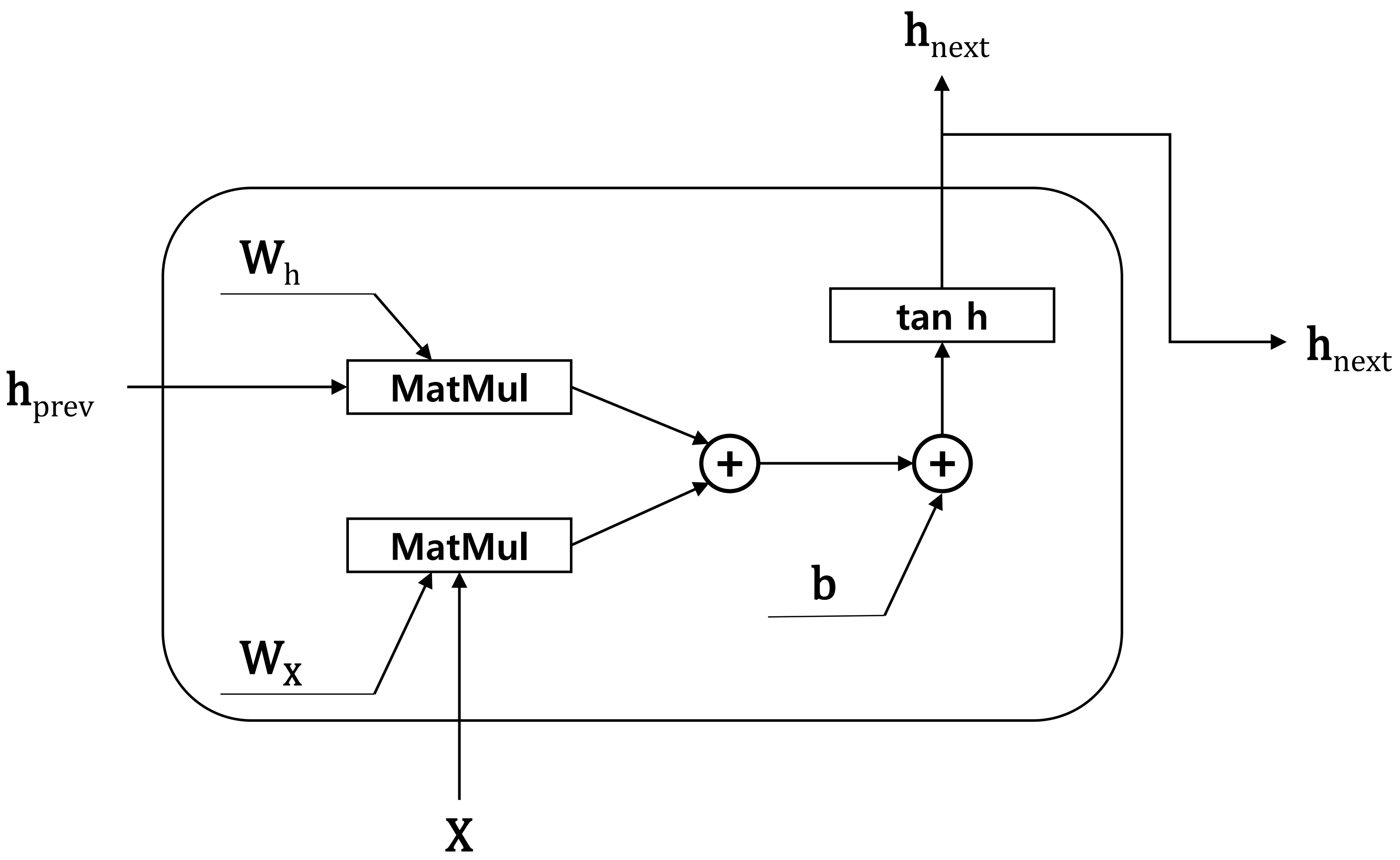
Recurrent Neural Network (RNN)

forward (순전파)



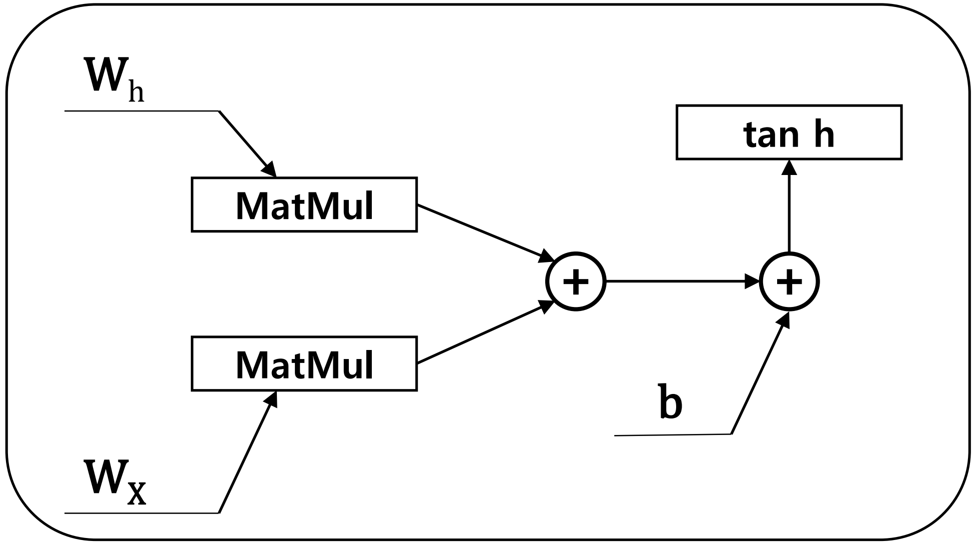
Recurrent Neural Network (RNN)

forward (순전파)



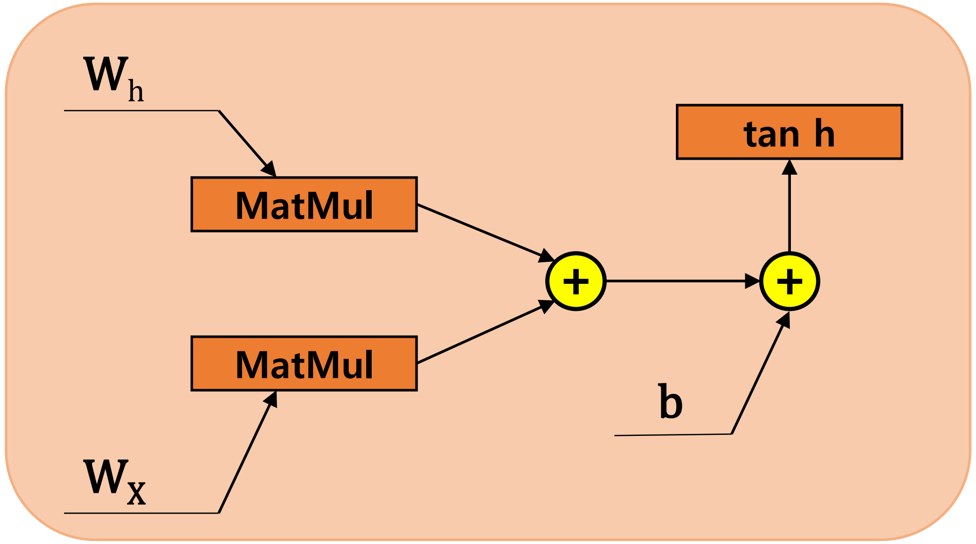
Recurrent Neural Network (RNN)

forward (순전파)



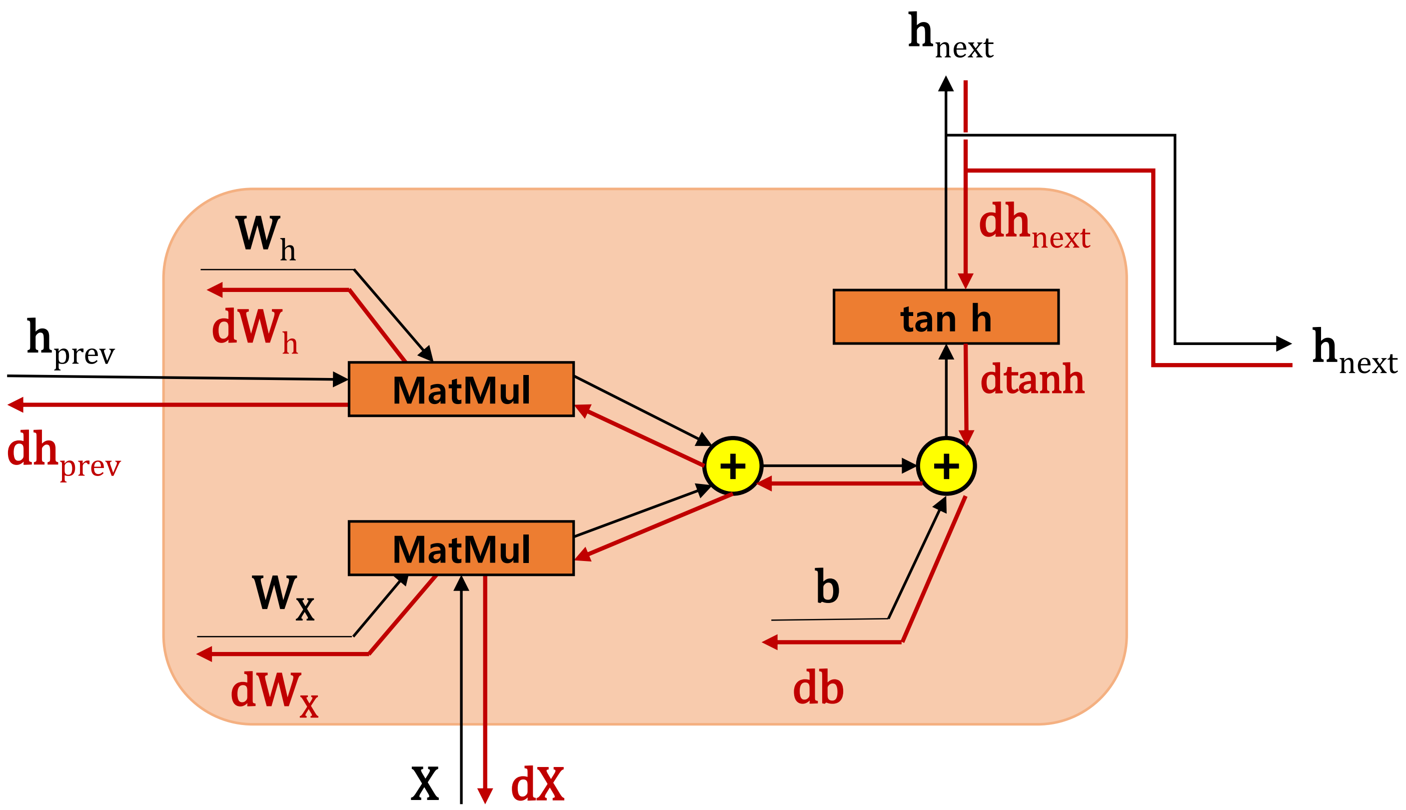
Recurrent Neural Network (RNN)

forward (순전파)



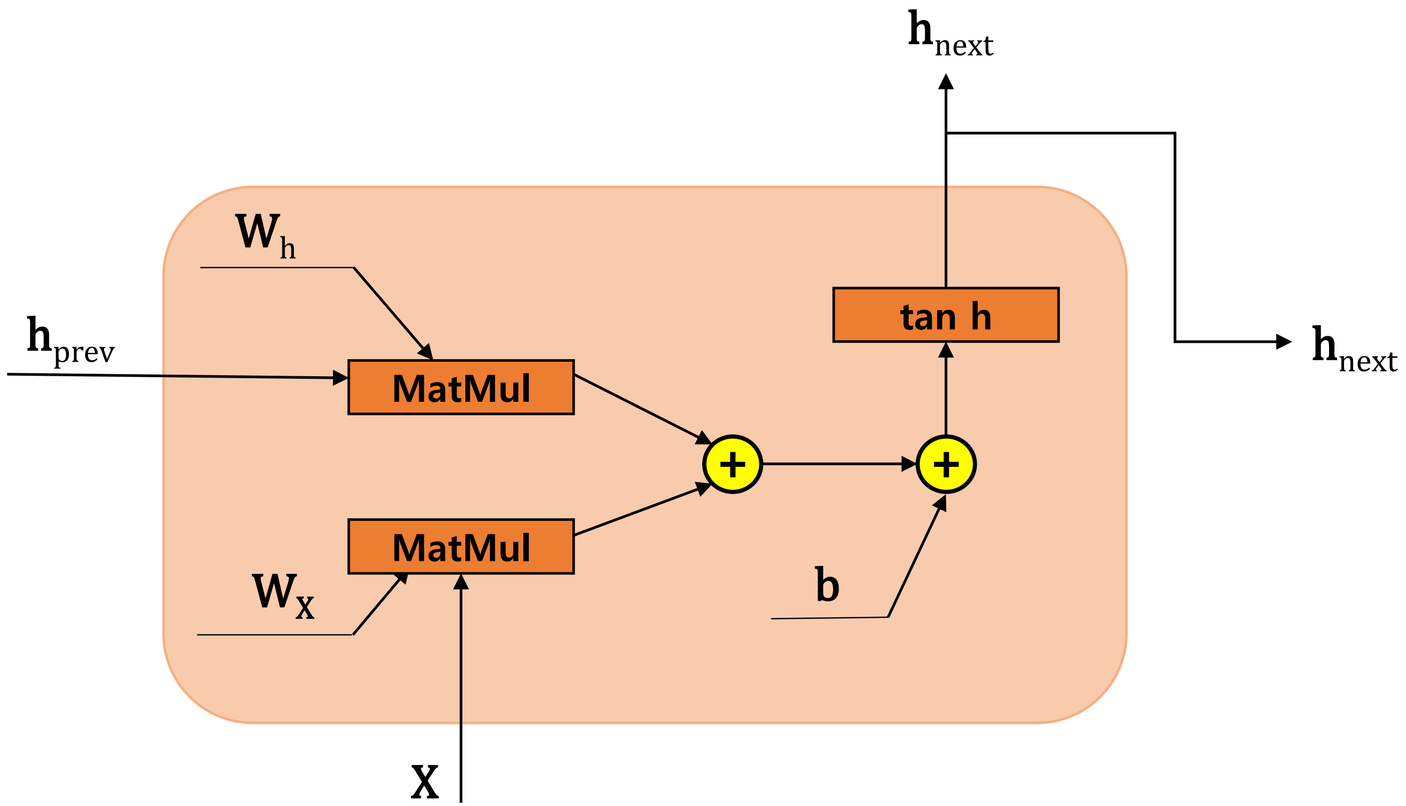
Recurrent Neural Network (RNN)

backward (역전파)



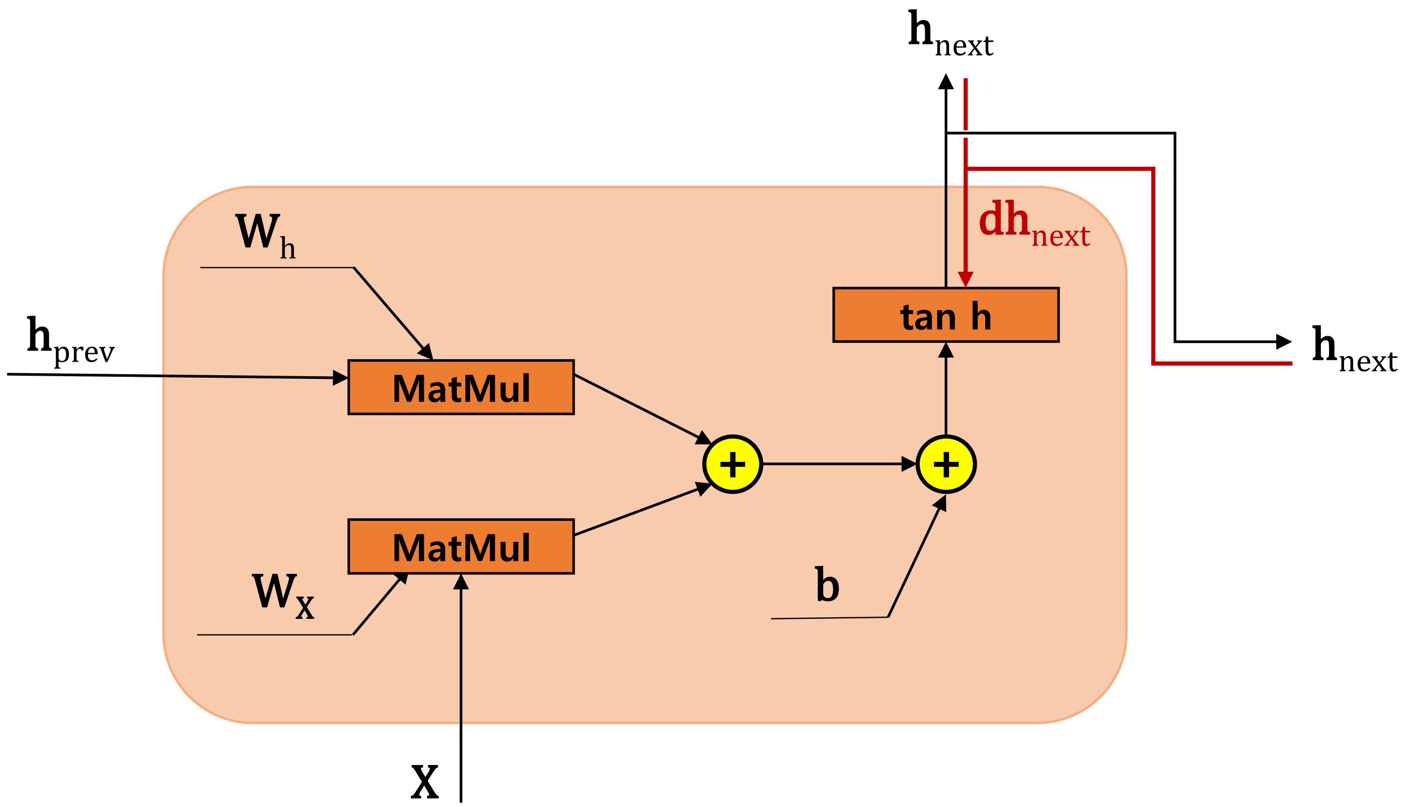
Recurrent Neural Network (RNN)

backward (역전파) - 시작



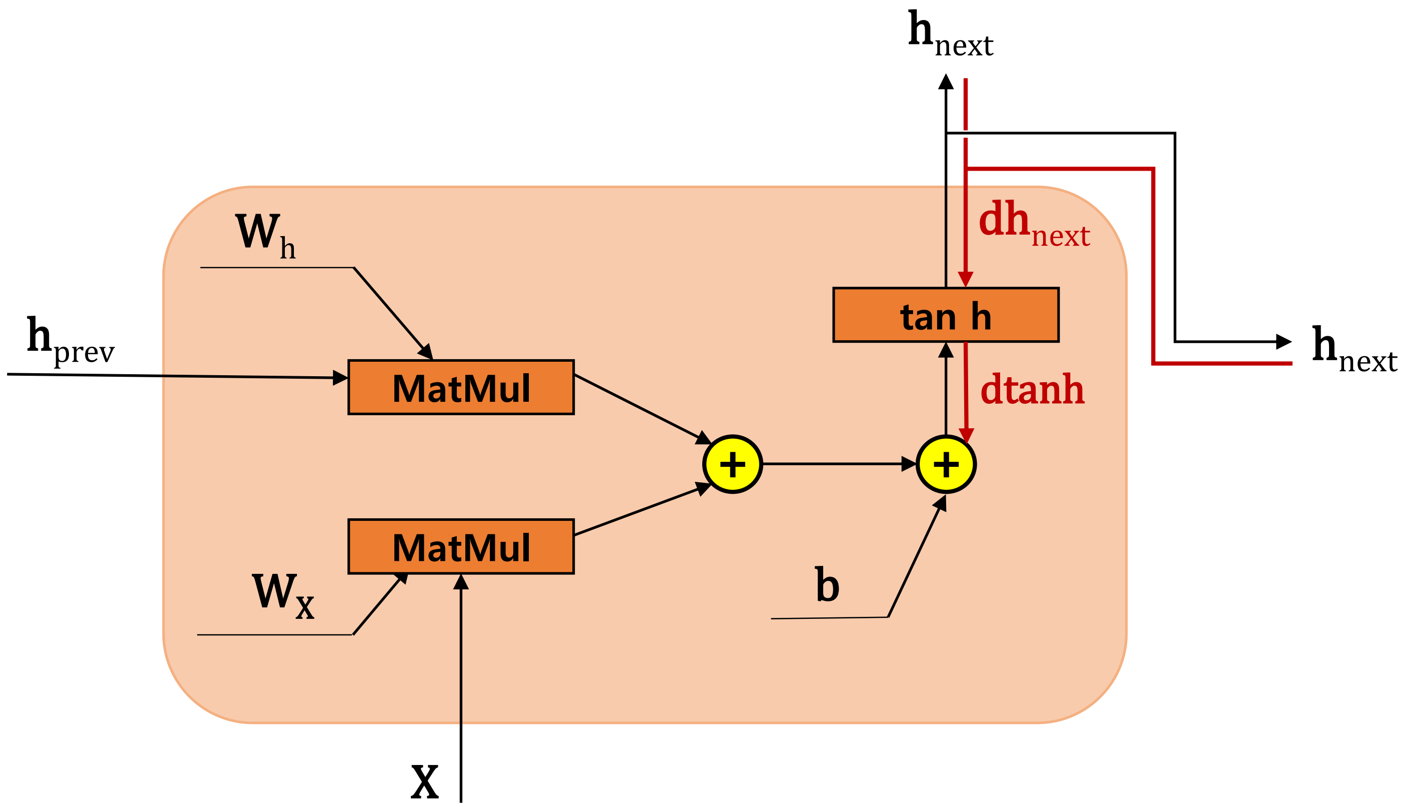
Recurrent Neural Network (RNN)

backward (역전파) - (1) dh_next



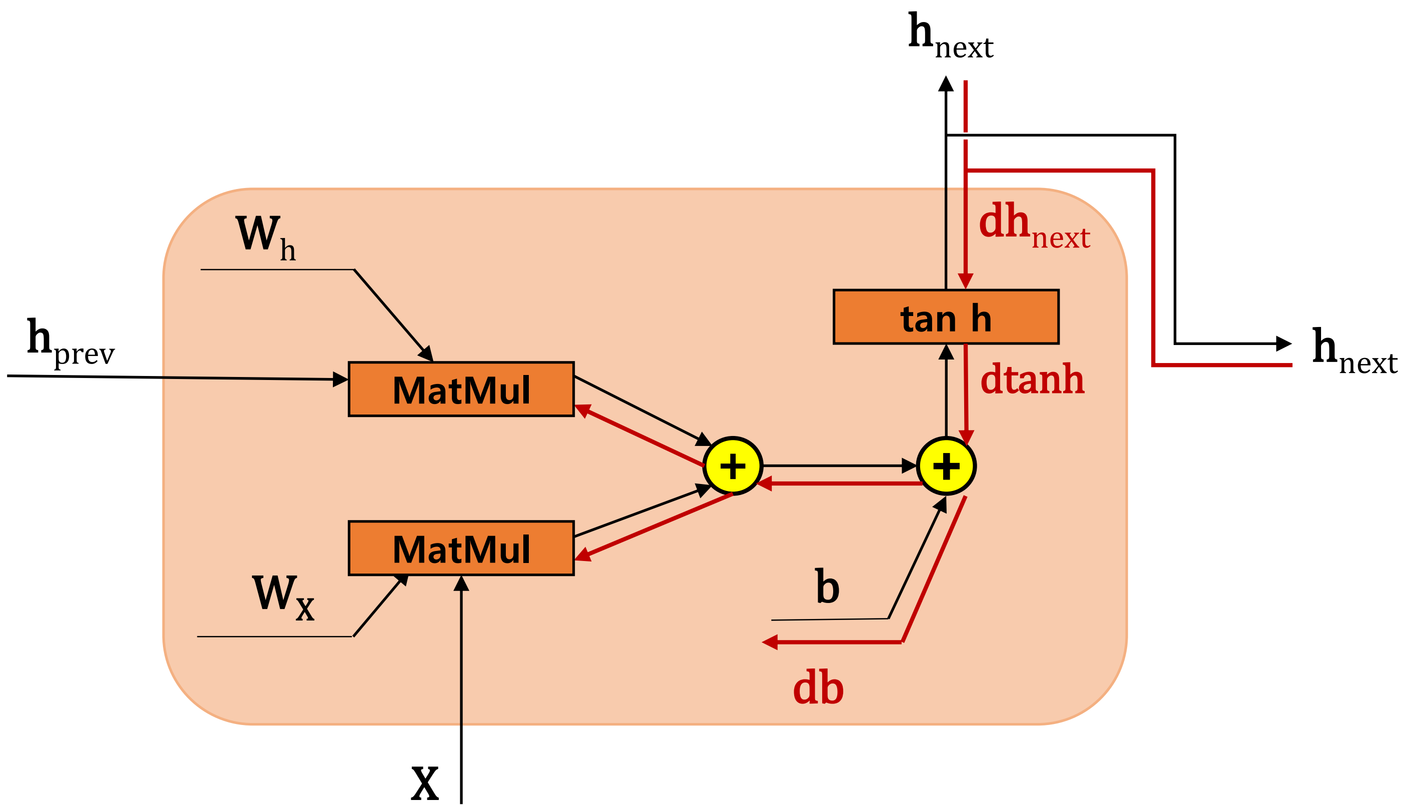
Recurrent Neural Network (RNN)

backward (역전파) - (2) dtanh



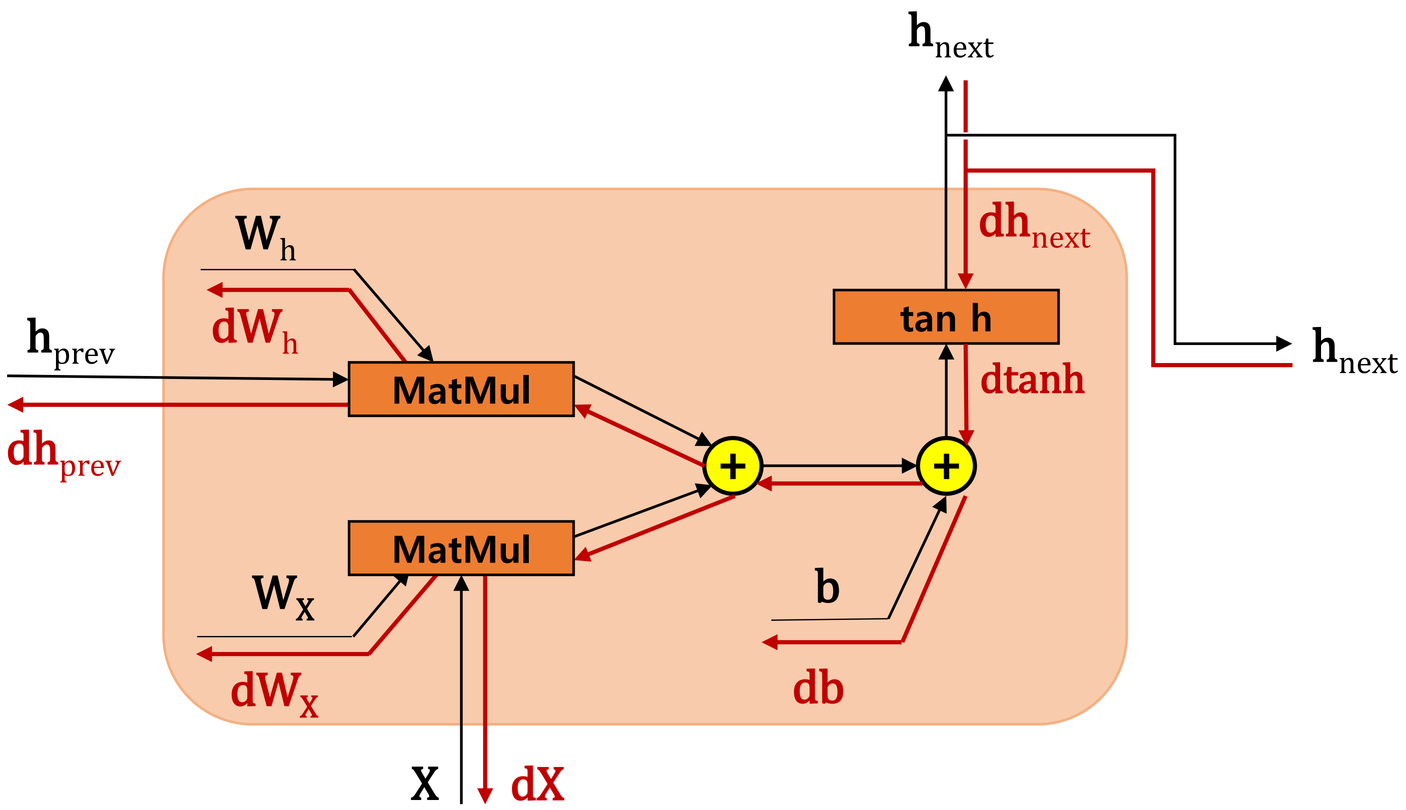
Recurrent Neural Network (RNN)

backward (역전파) – (3) 덧셈 노드



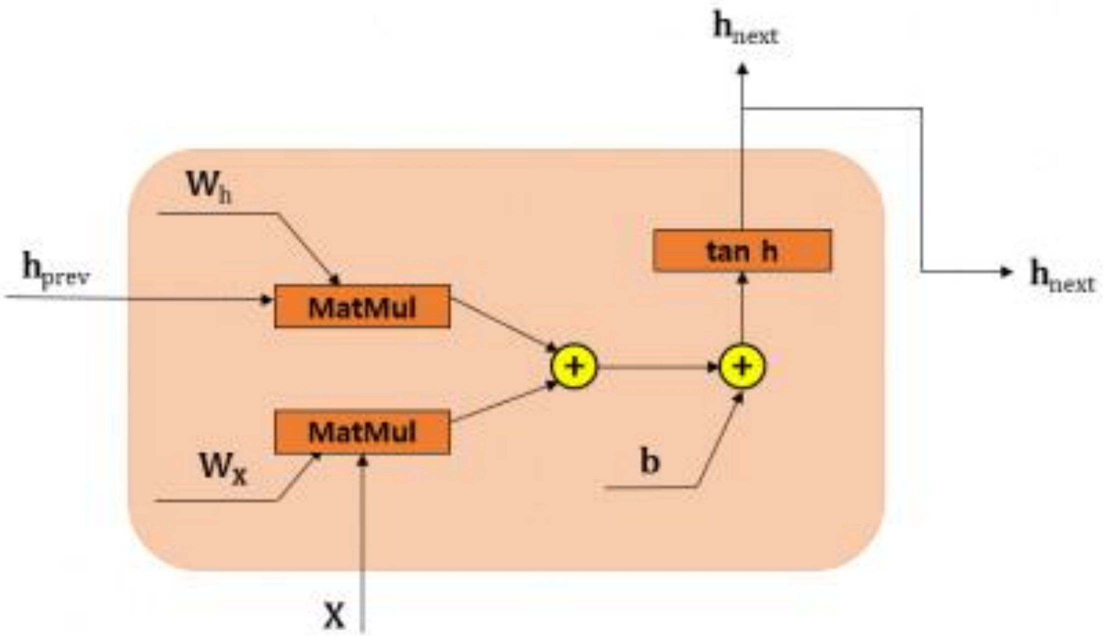
Recurrent Neural Network (RNN)

backward (역전파) – (4) 곱셈 노드

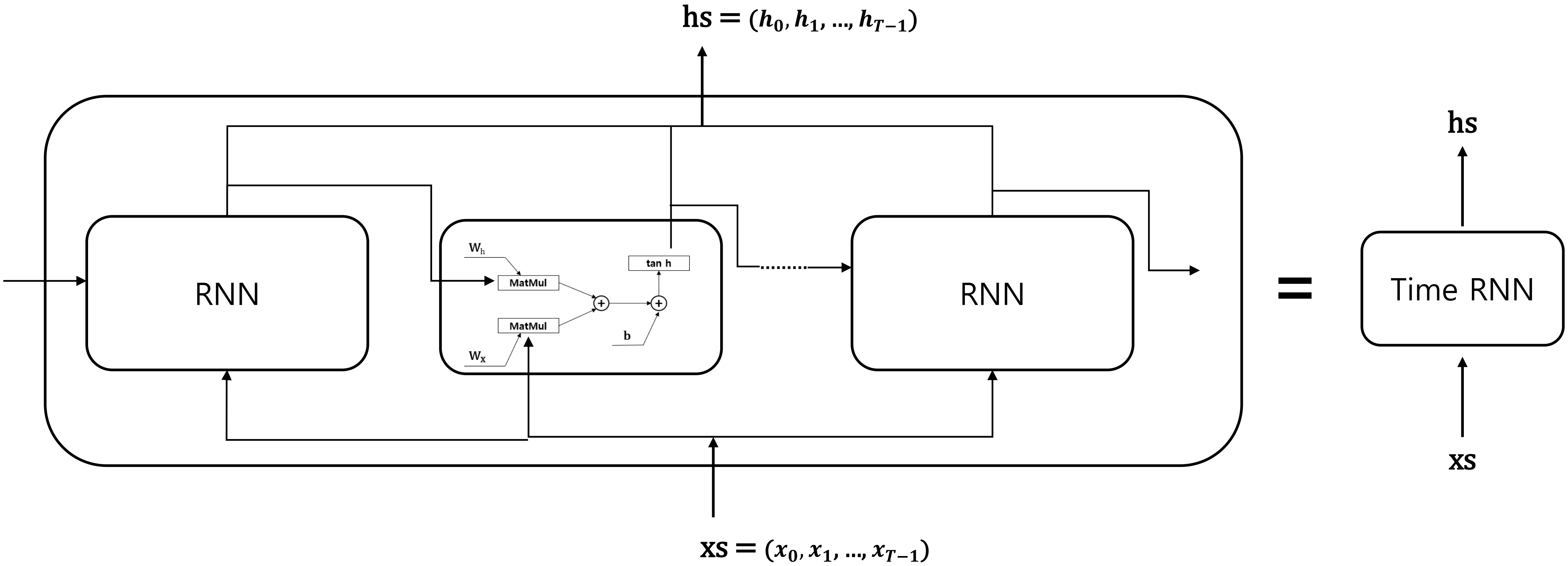


Recurrent Neural Network (RNN)

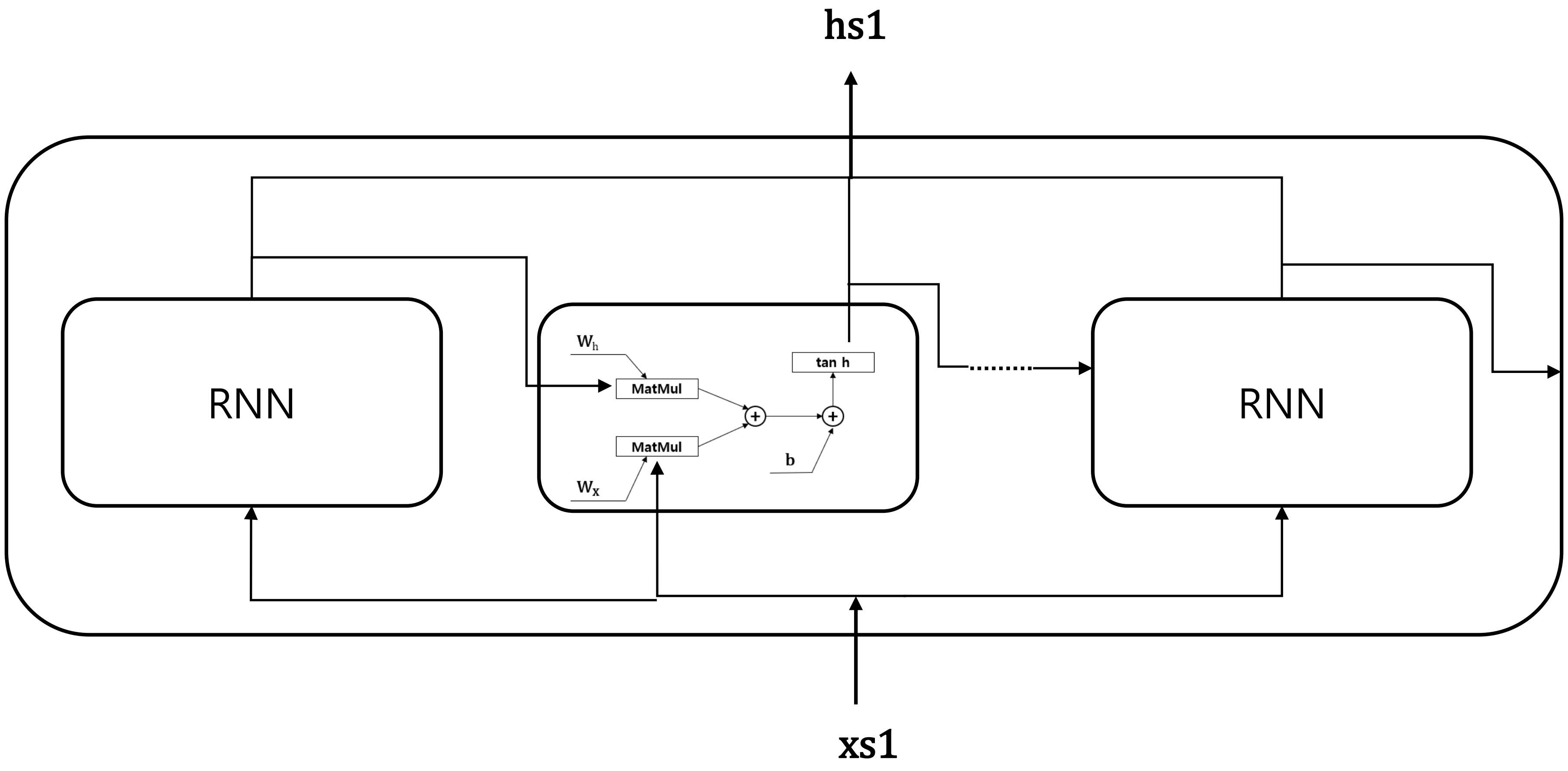
backward (역전파) – gif



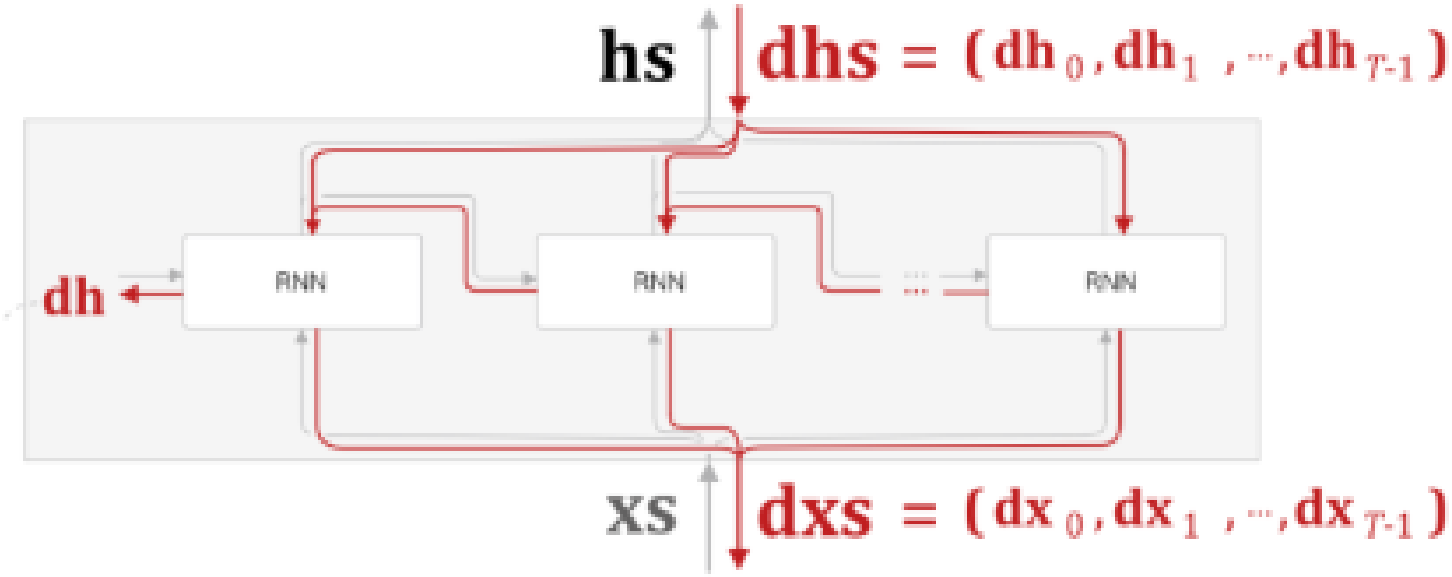
Time RNN 계층과 RNN 계층



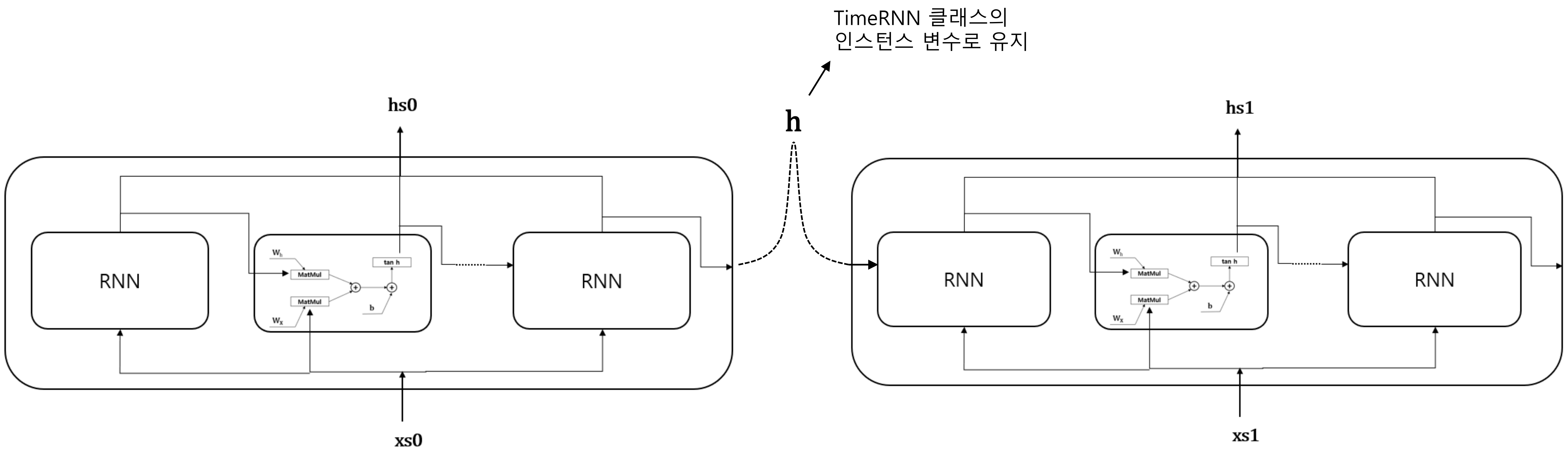
Time RNN 계층과 RNN 계층



Time RNN 계층의 역전파



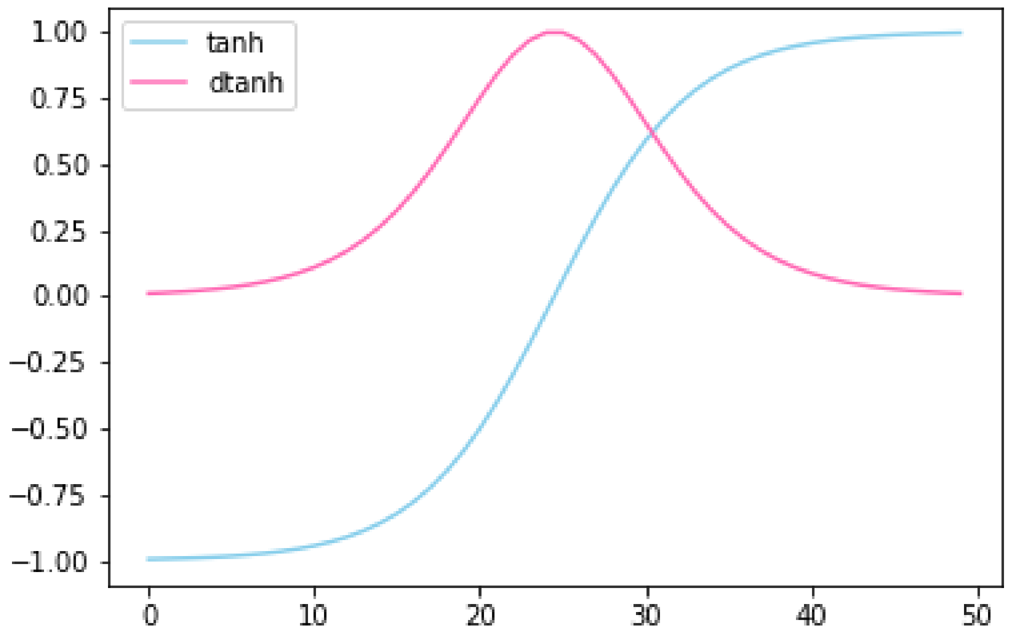
Time RNN 계층과 RNN 계층



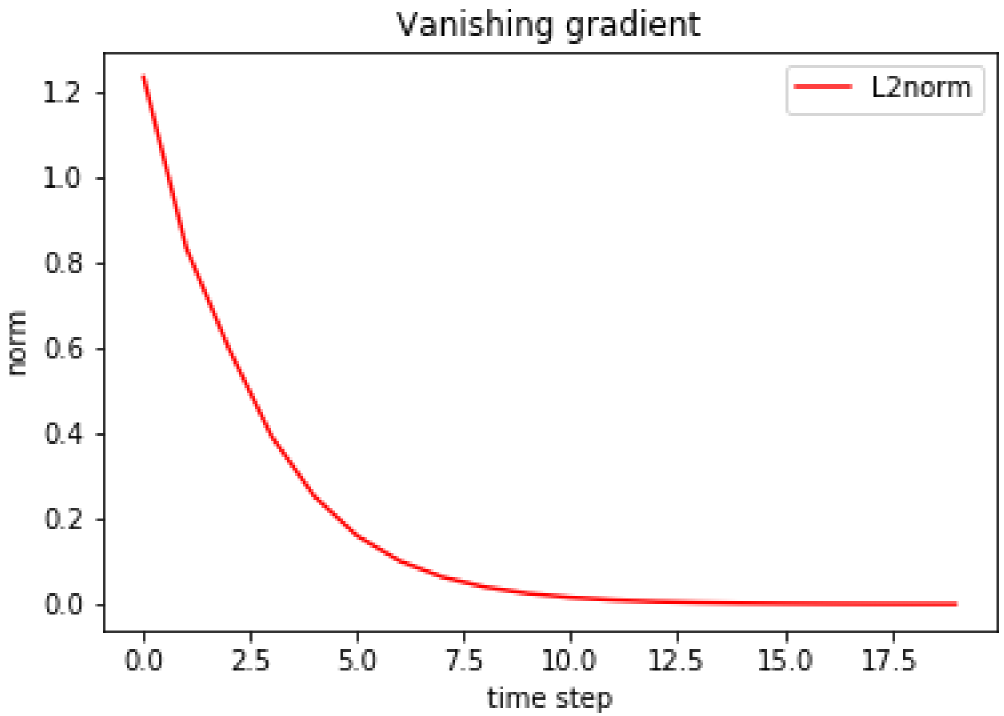
LSTM

Hyperbolic tangent

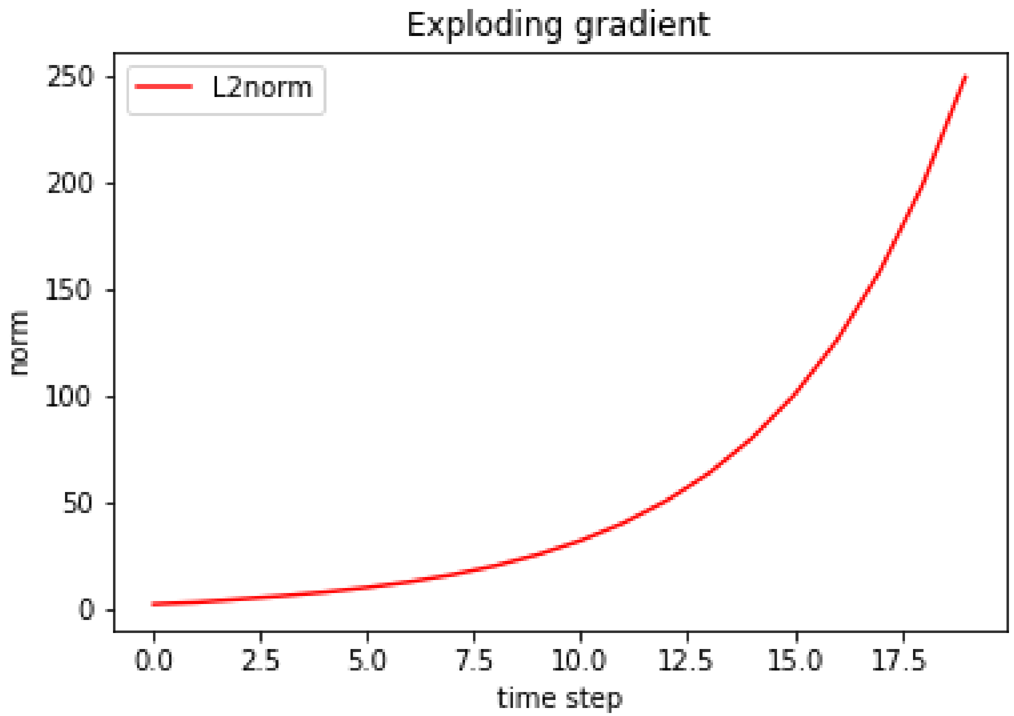
tanh & dtanh



Vanishing gradient

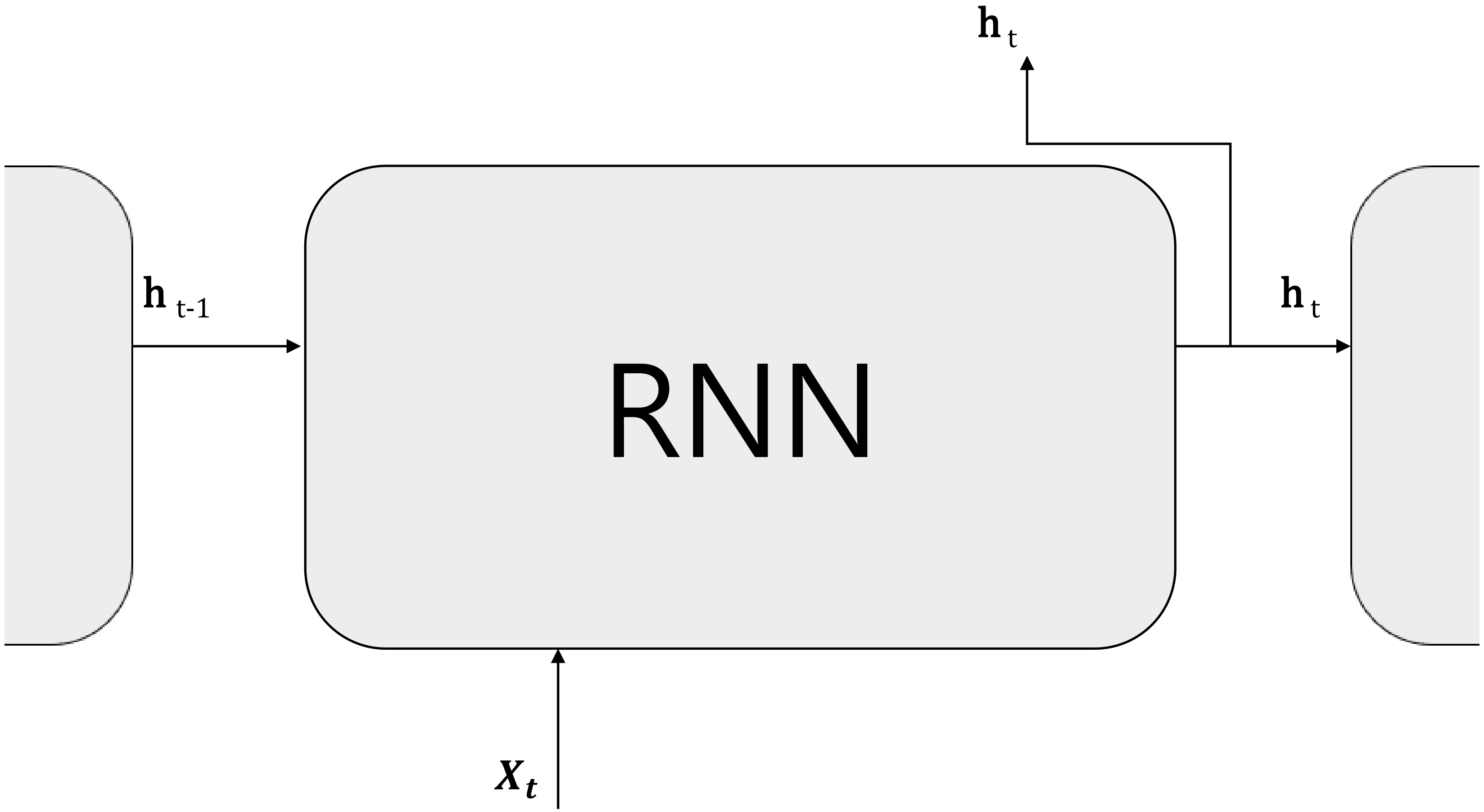


Exploding gradient



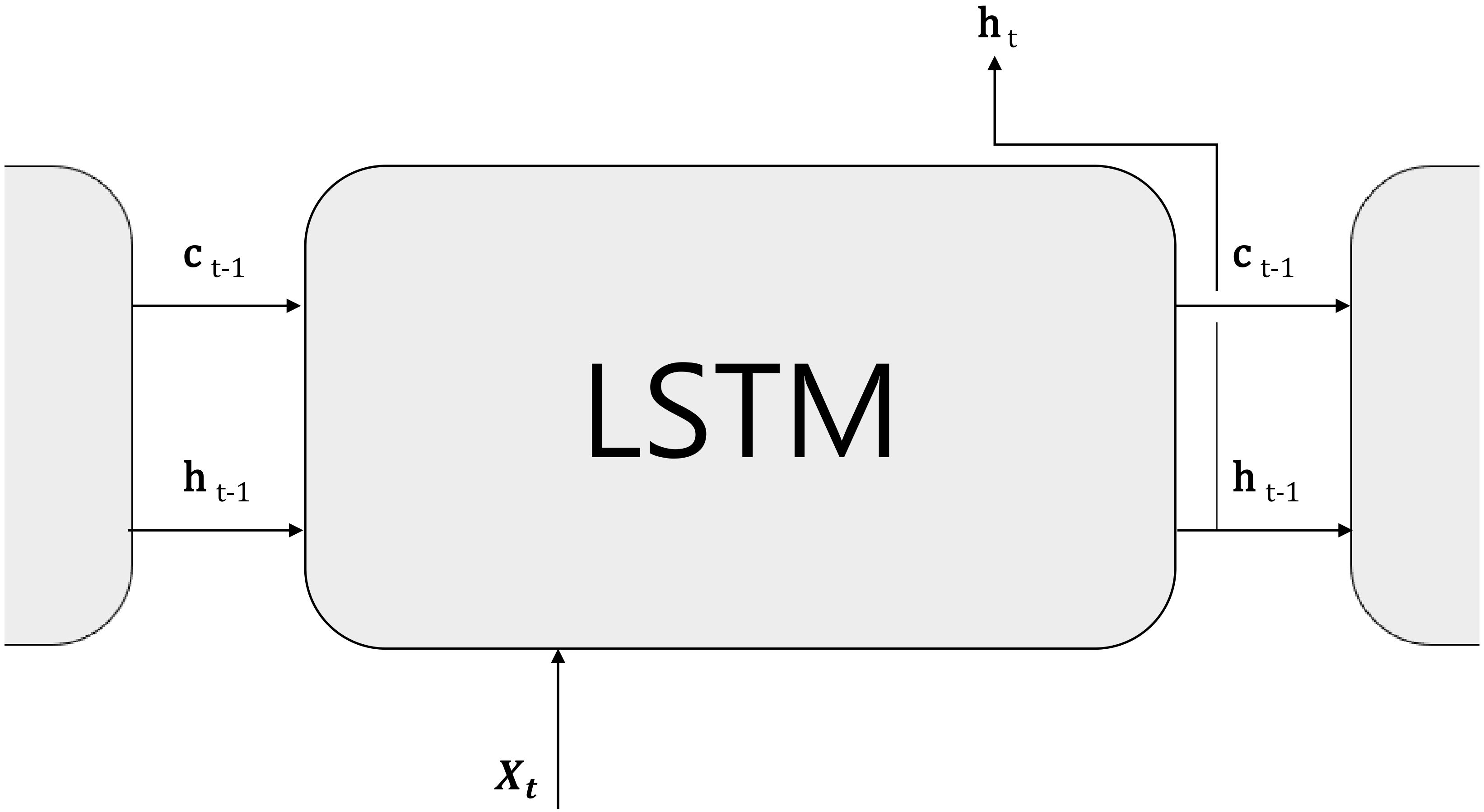
Recurrent Neural Network (RNN)

Interface



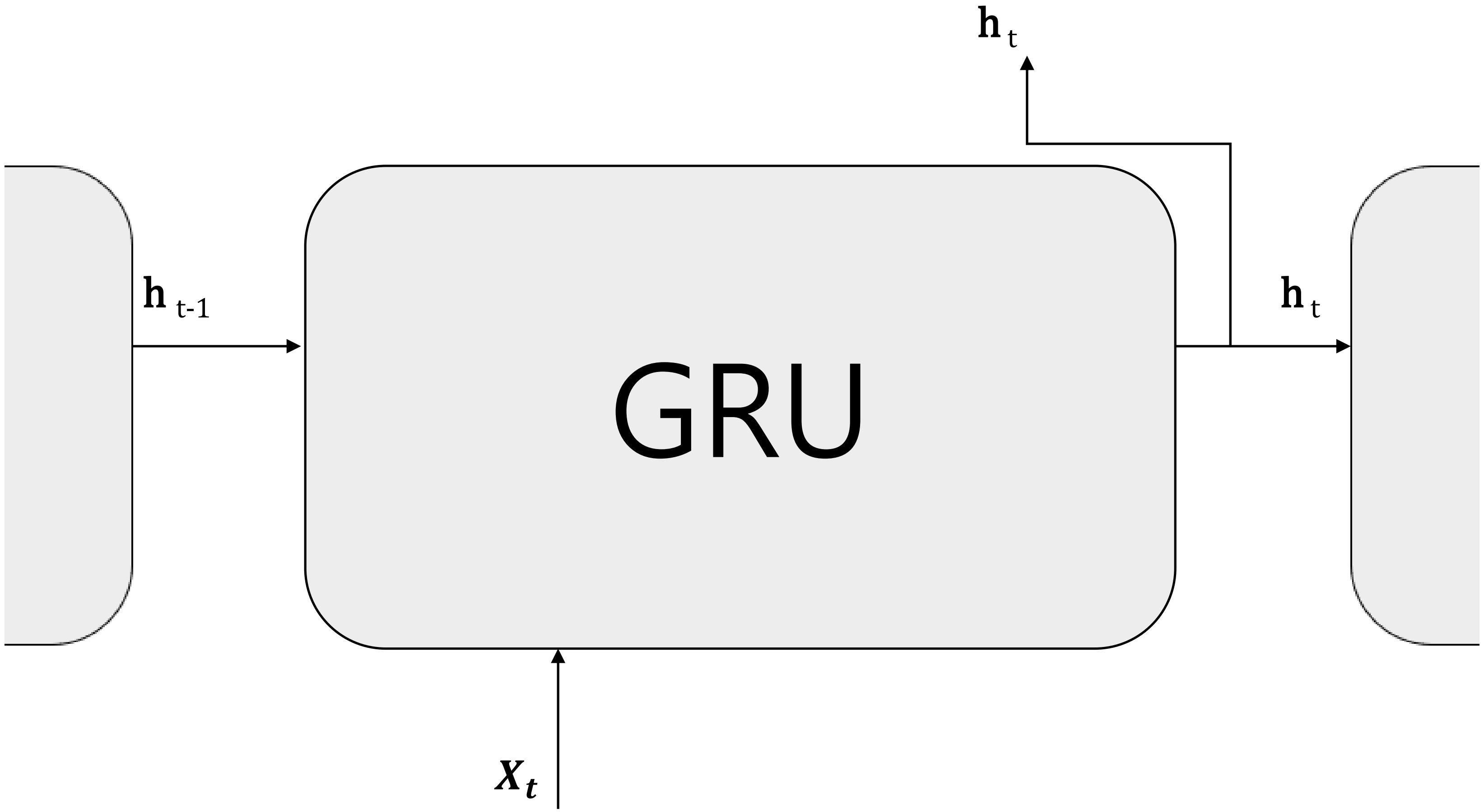
Long Shor Term Memory (LSTM)

Interface

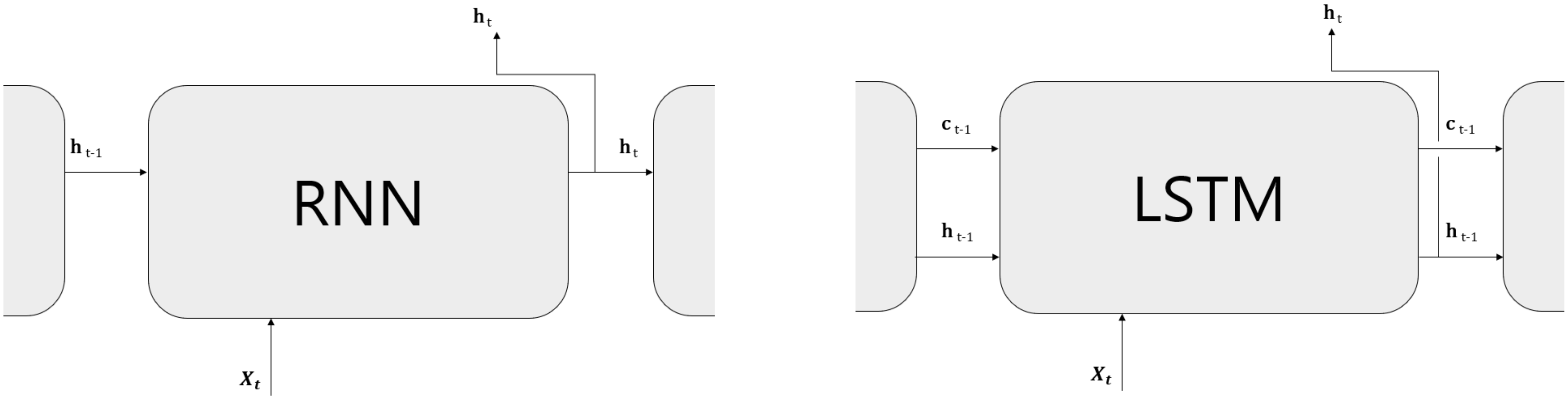


Gate Recurrent Unit (GRU)

Interface

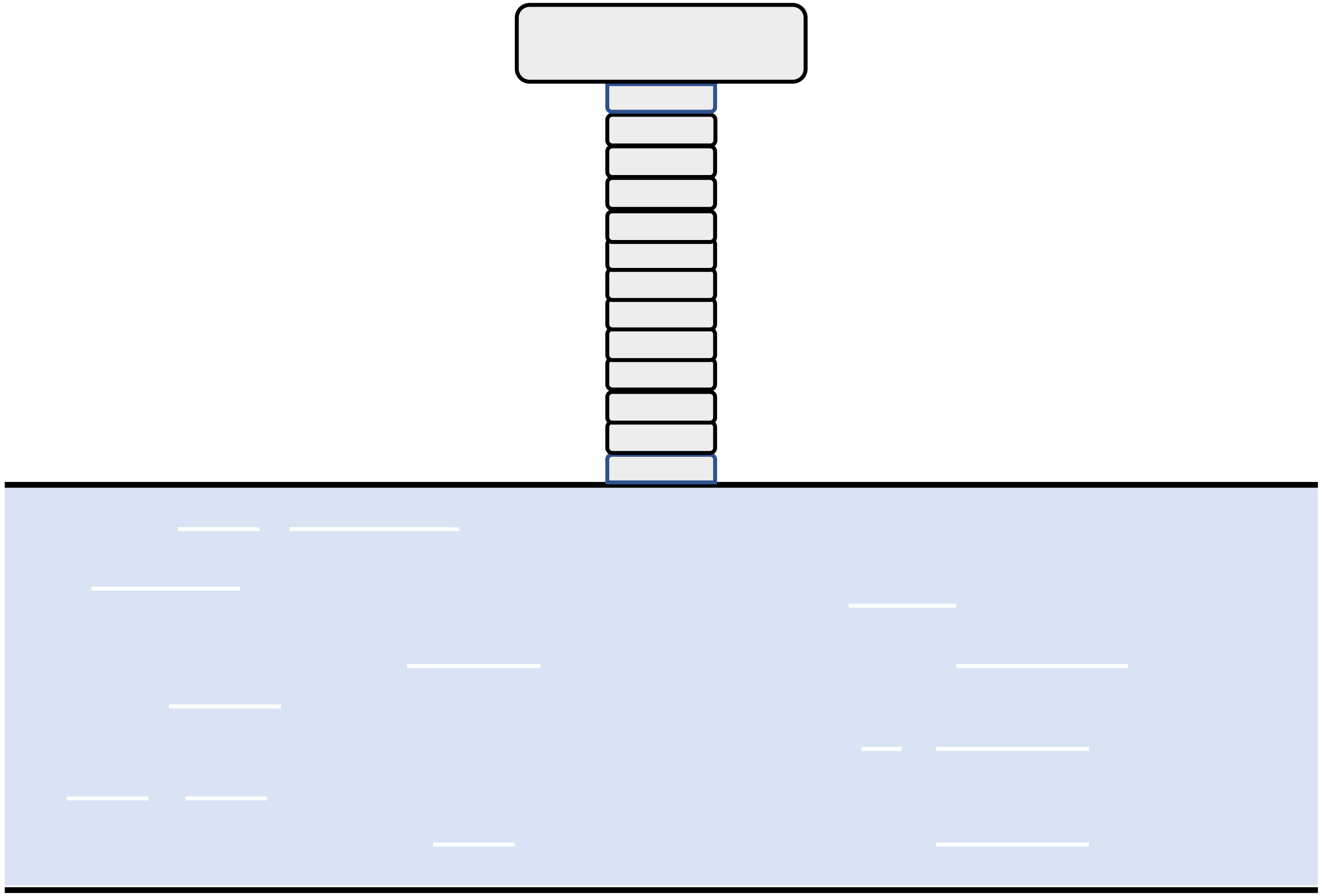


RNN과 LSTM Interface 비교



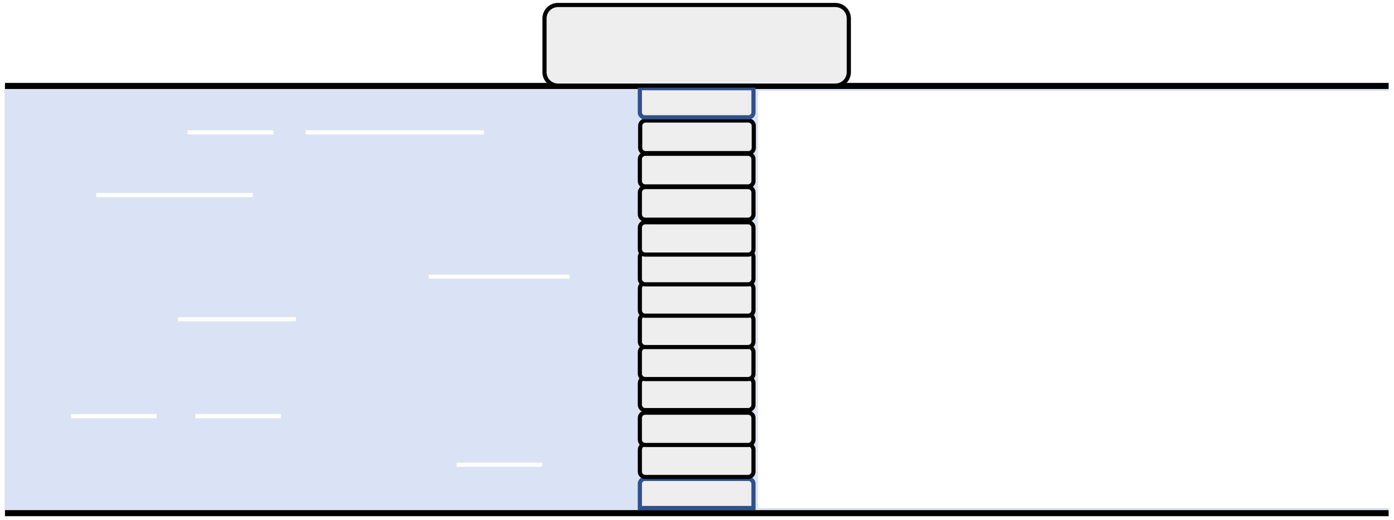
Long Shor Term Memory (LSTM)

비유하자면 게이트는 물의 흐름을 제어한다



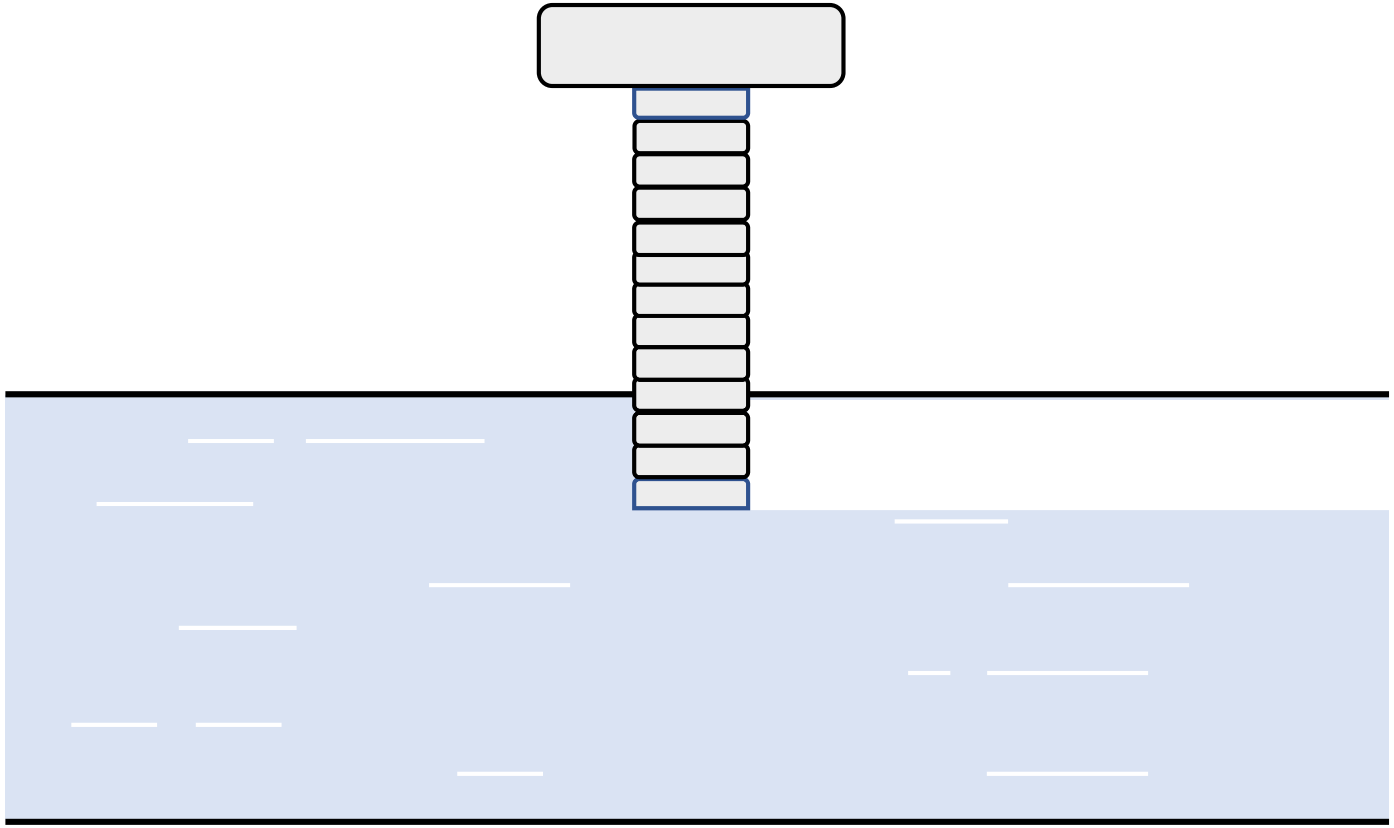
Long Shor Term Memory (LSTM)

비유하자면 게이트는 물의 흐름을 제어한다



Long Shor Term Memory (LSTM)

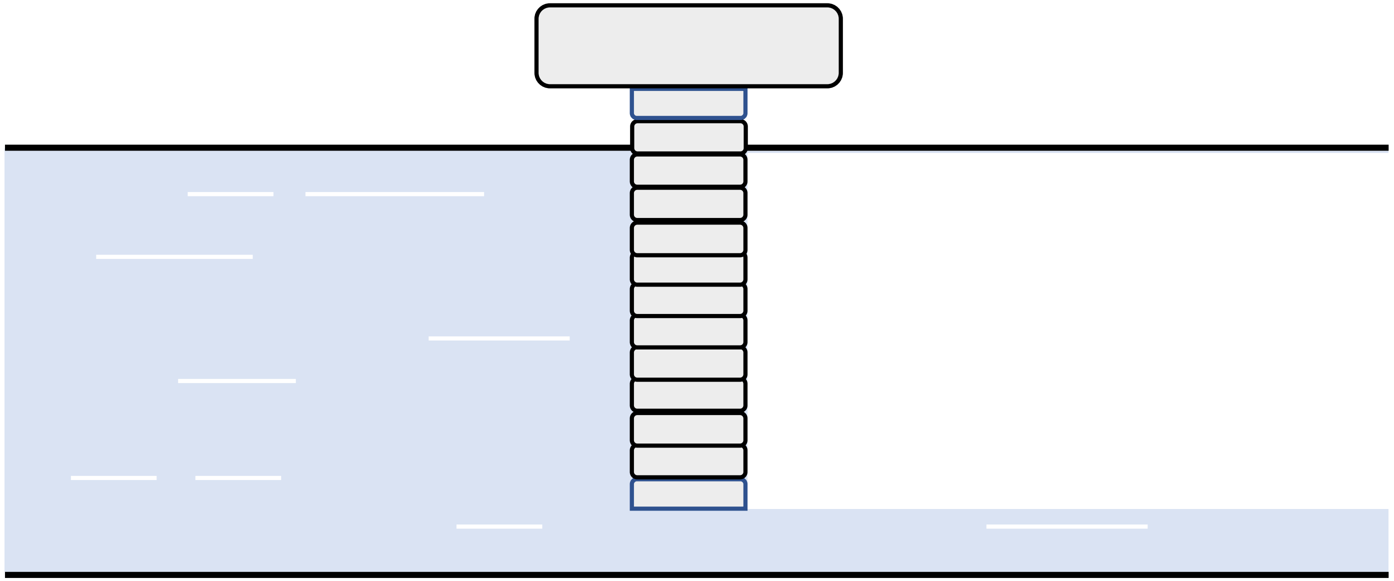
물이 흐르는 양을 0.0 ~ 1.0 범위에서 제어한다.



0.7 (70%)

Long Shor Term Memory (LSTM)

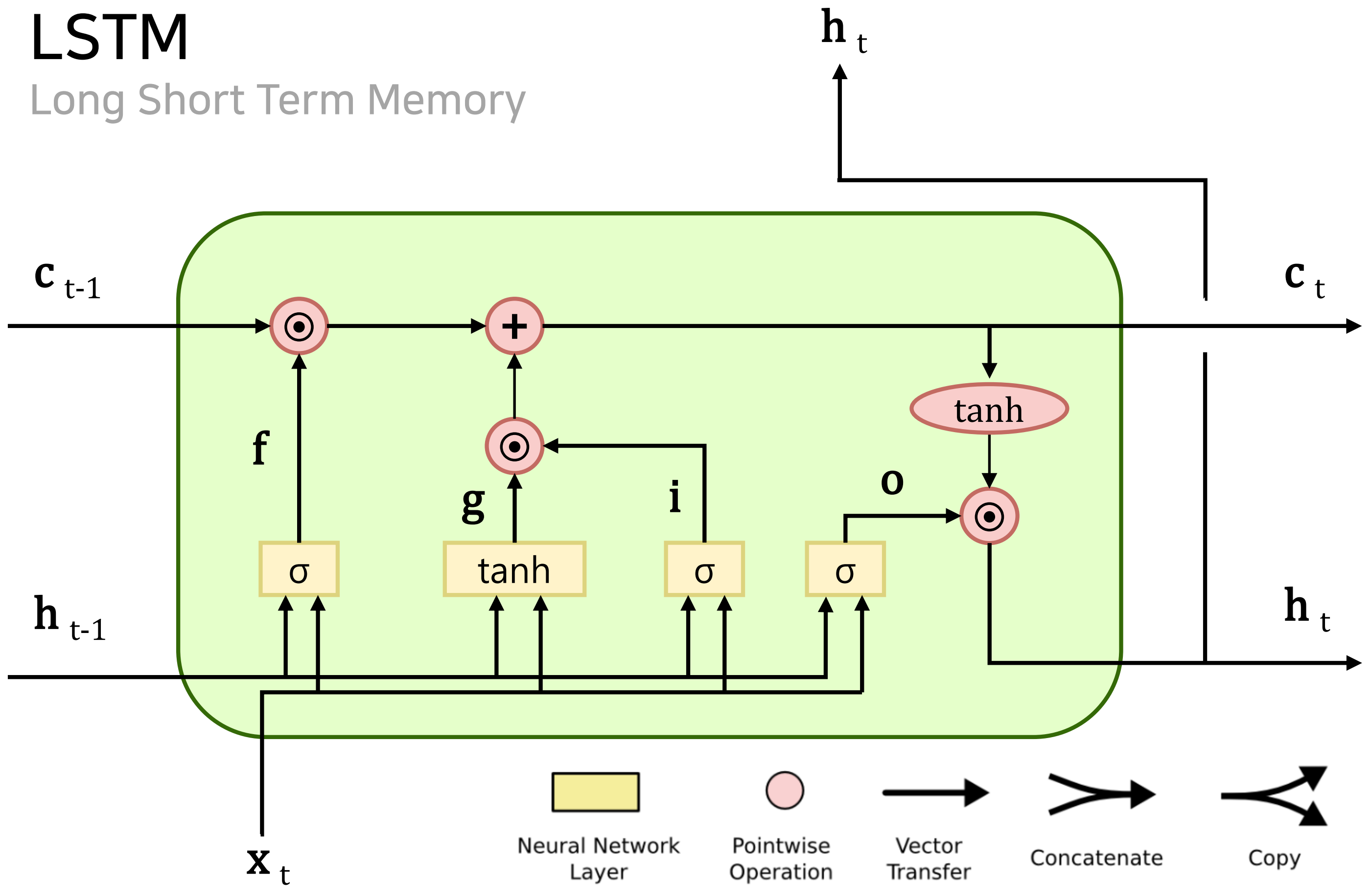
물이 흐르는 양을 0.0 ~ 1.0 범위에서 제어한다.



0.2 (20%)

Long Short Term Memory (LSTM)

LSTM의 계산 그래프



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

$$g = \tanh(x_t W_x^{(g)} + h_{t-1} W_h^{(g)} + b^{(g)})$$

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

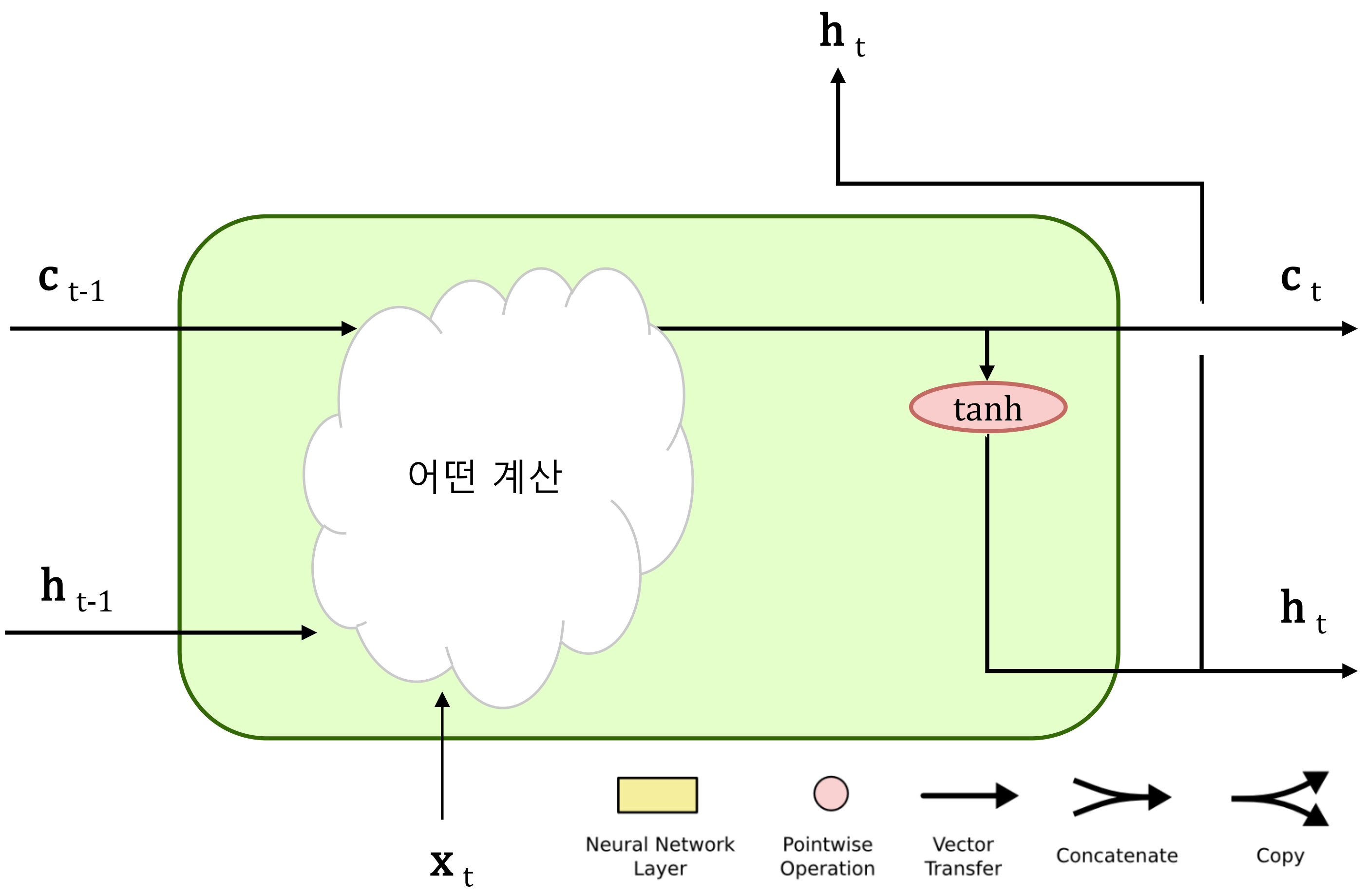
$$o = \sigma(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)})$$

$$c_t = f \odot c_{t-1} + g \odot i$$

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

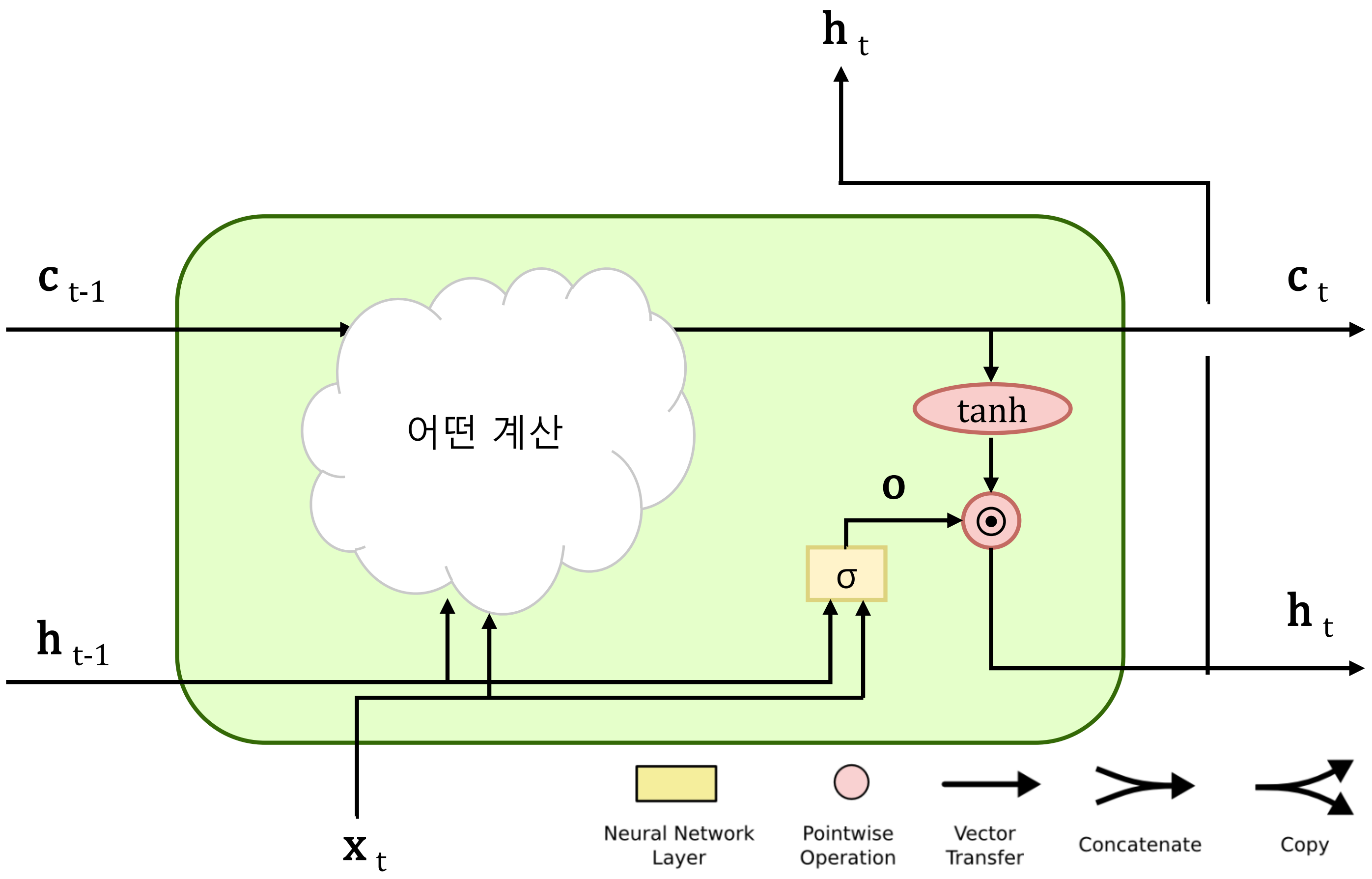
Long Short Term Memory (LSTM)

기억 셀 c_t를 바탕으로 은닉상태 h_t를 계산하는 LSTM 계층



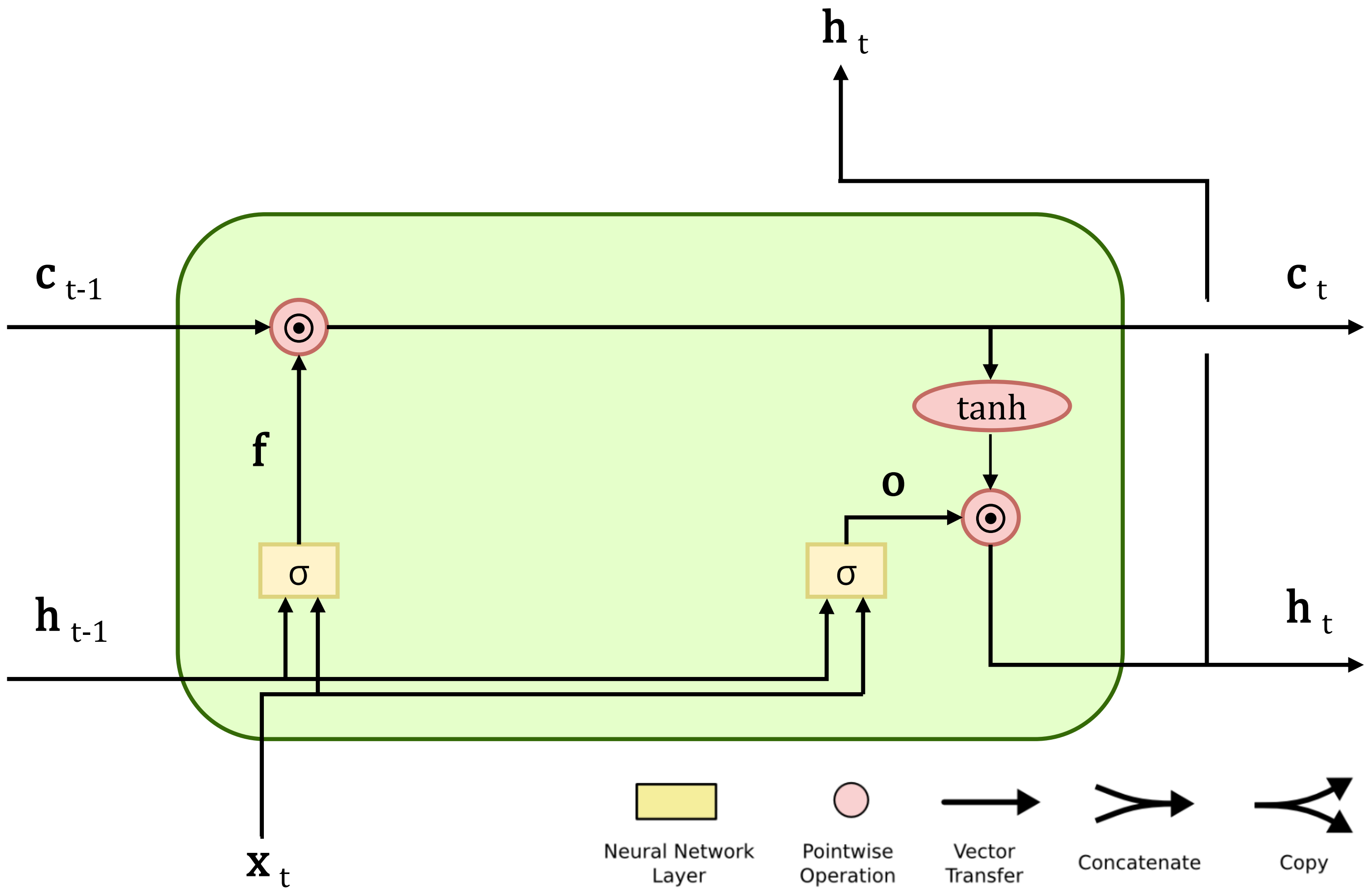
Long Short Term Memory (LSTM)

output 게이트 추가 (o gate)



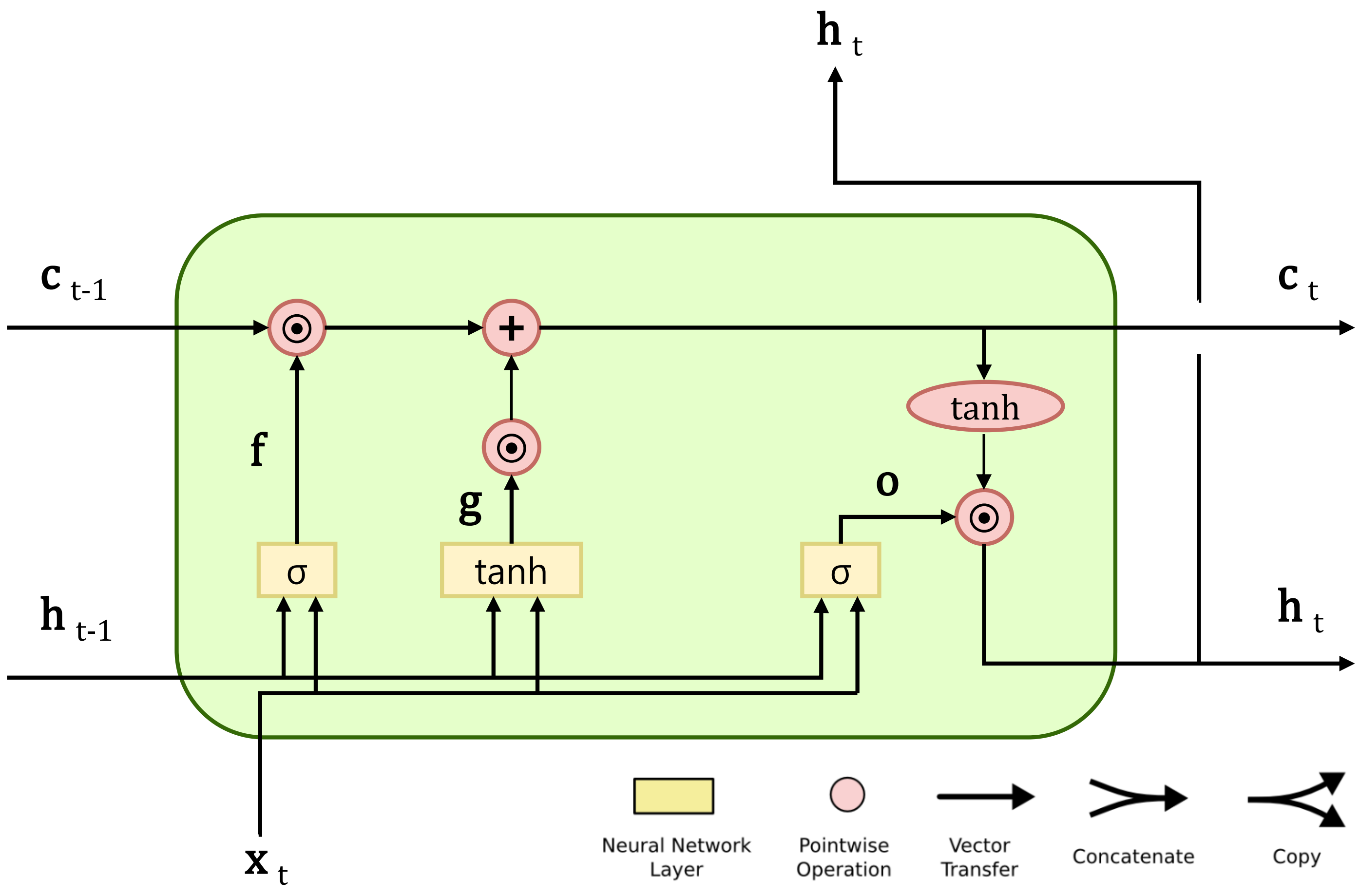
Long Short Term Memory (LSTM)

forget 게이트 추가 (f gate)



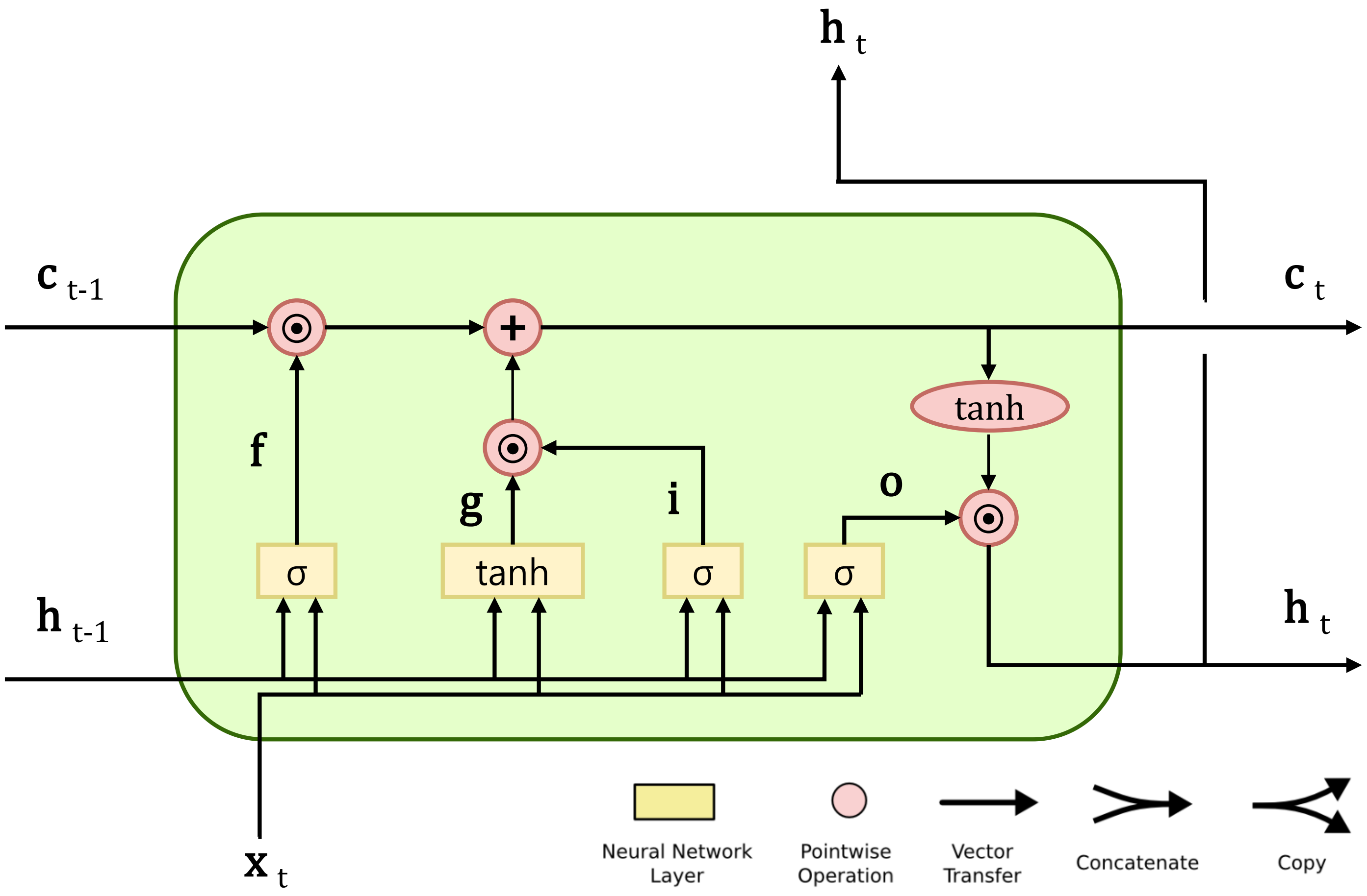
Long Short Term Memory (LSTM)

새로운 기억 셀에 필요한 정보를 추가 (g gate)

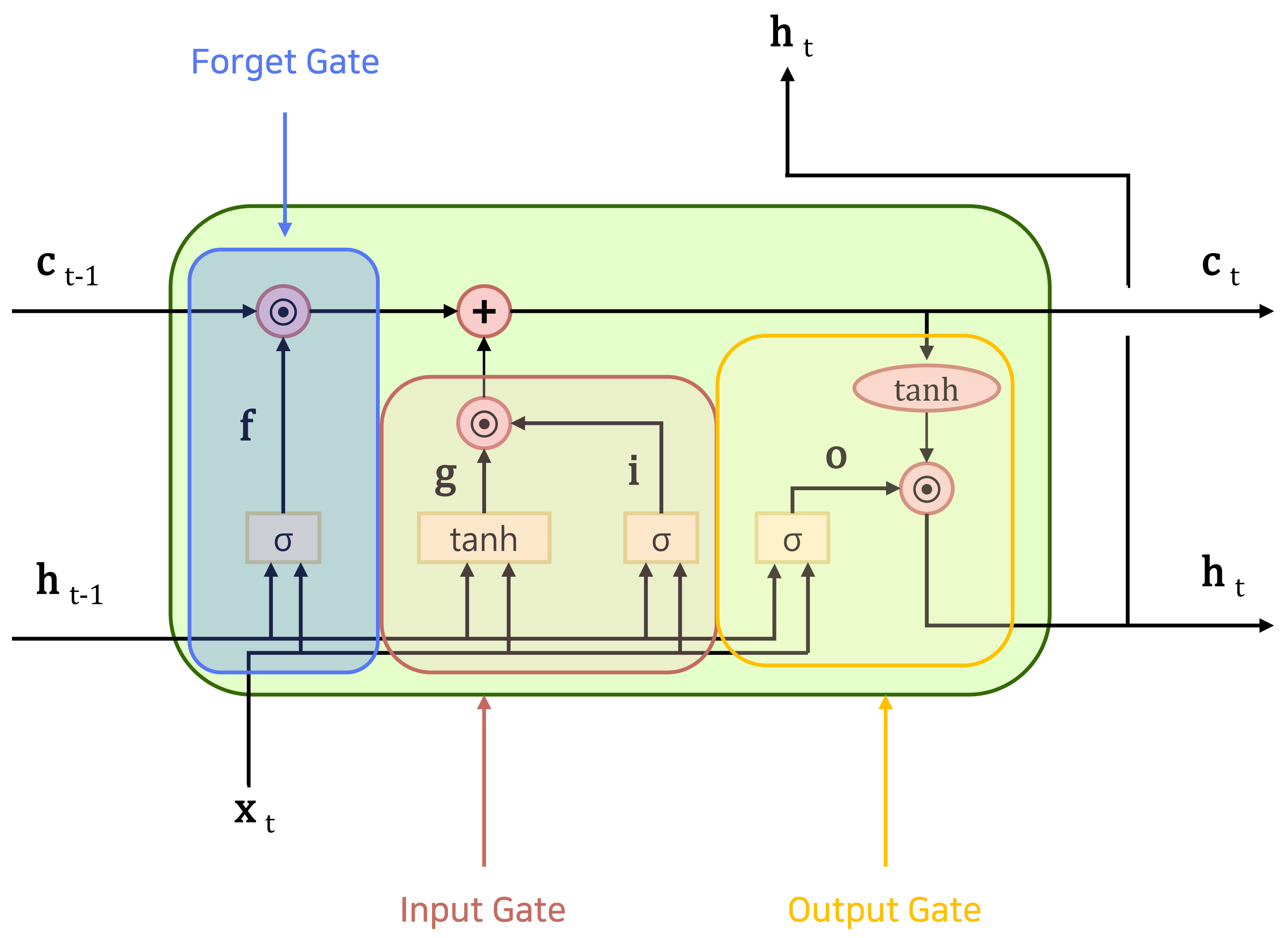


Long Short Term Memory (LSTM)

Input 게이트 추가 (i gate)

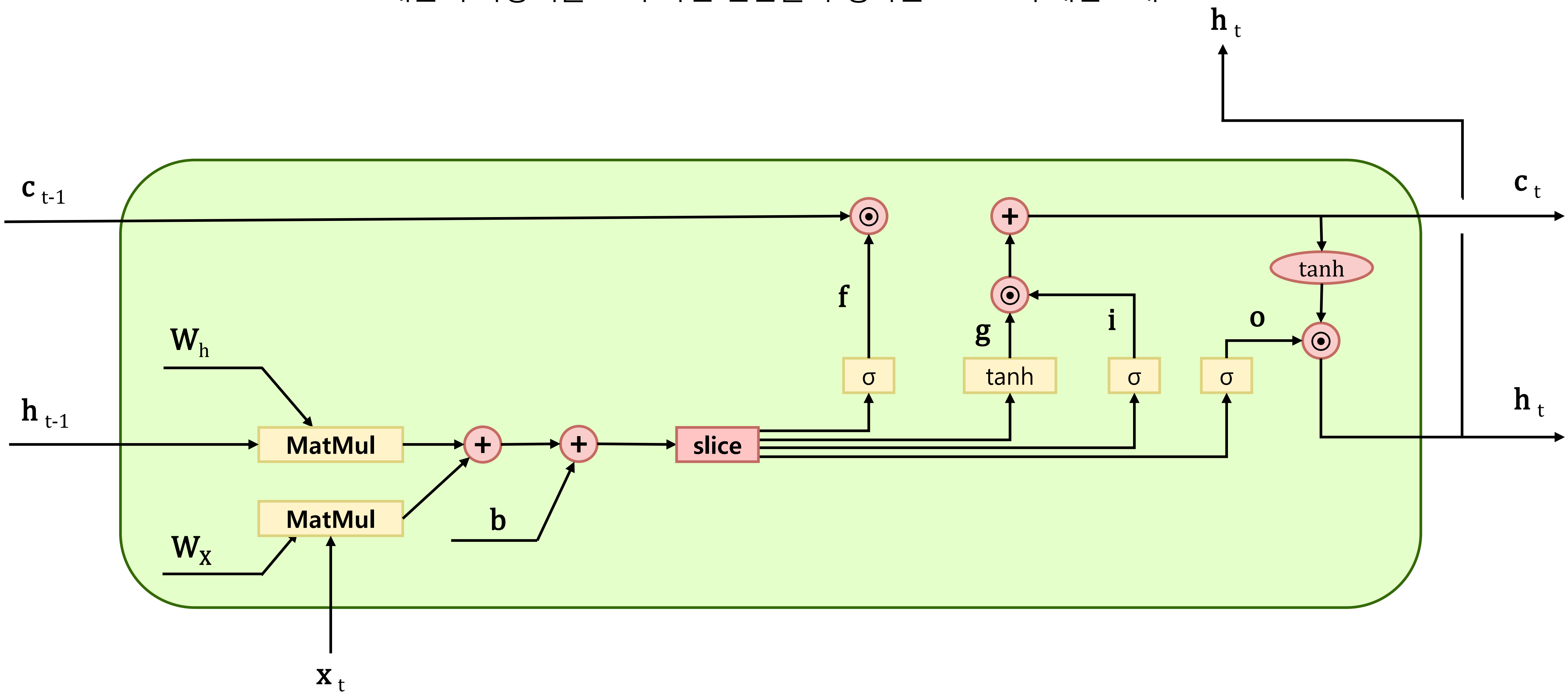


Long Short Term Memory (LSTM)

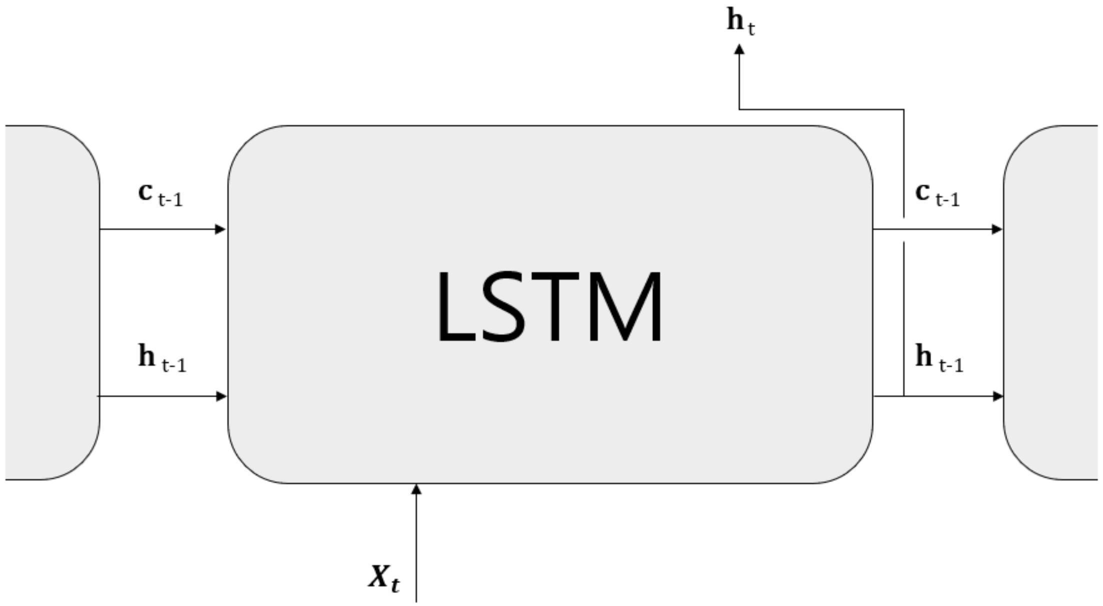


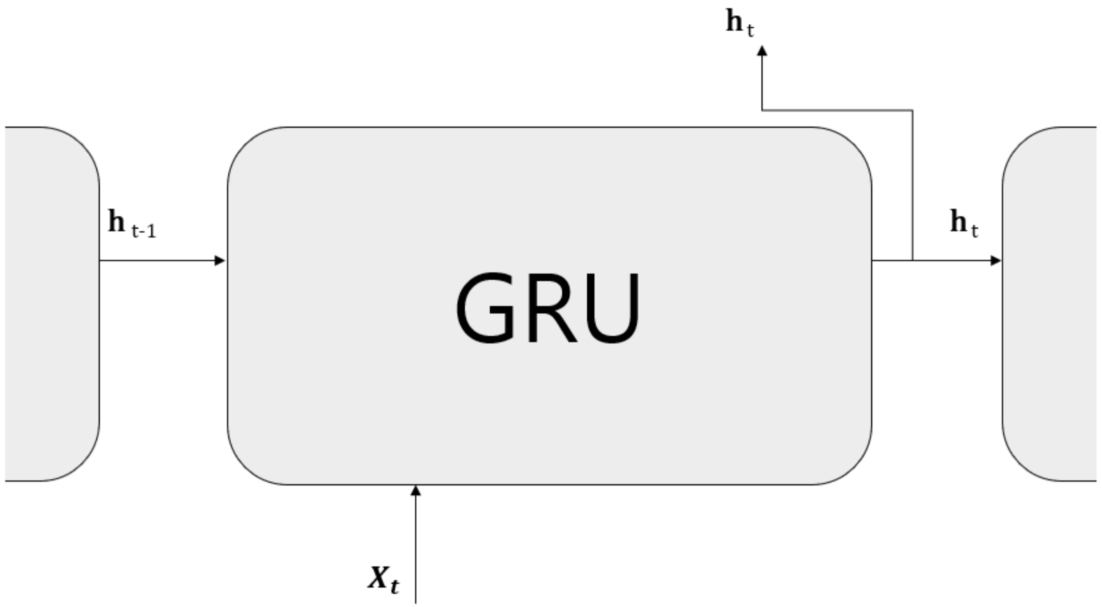
Long Short Term Memory (LSTM)

4개분의 가중치를 모아 아핀 변환을 수행하는 LSTM의 계산그래프



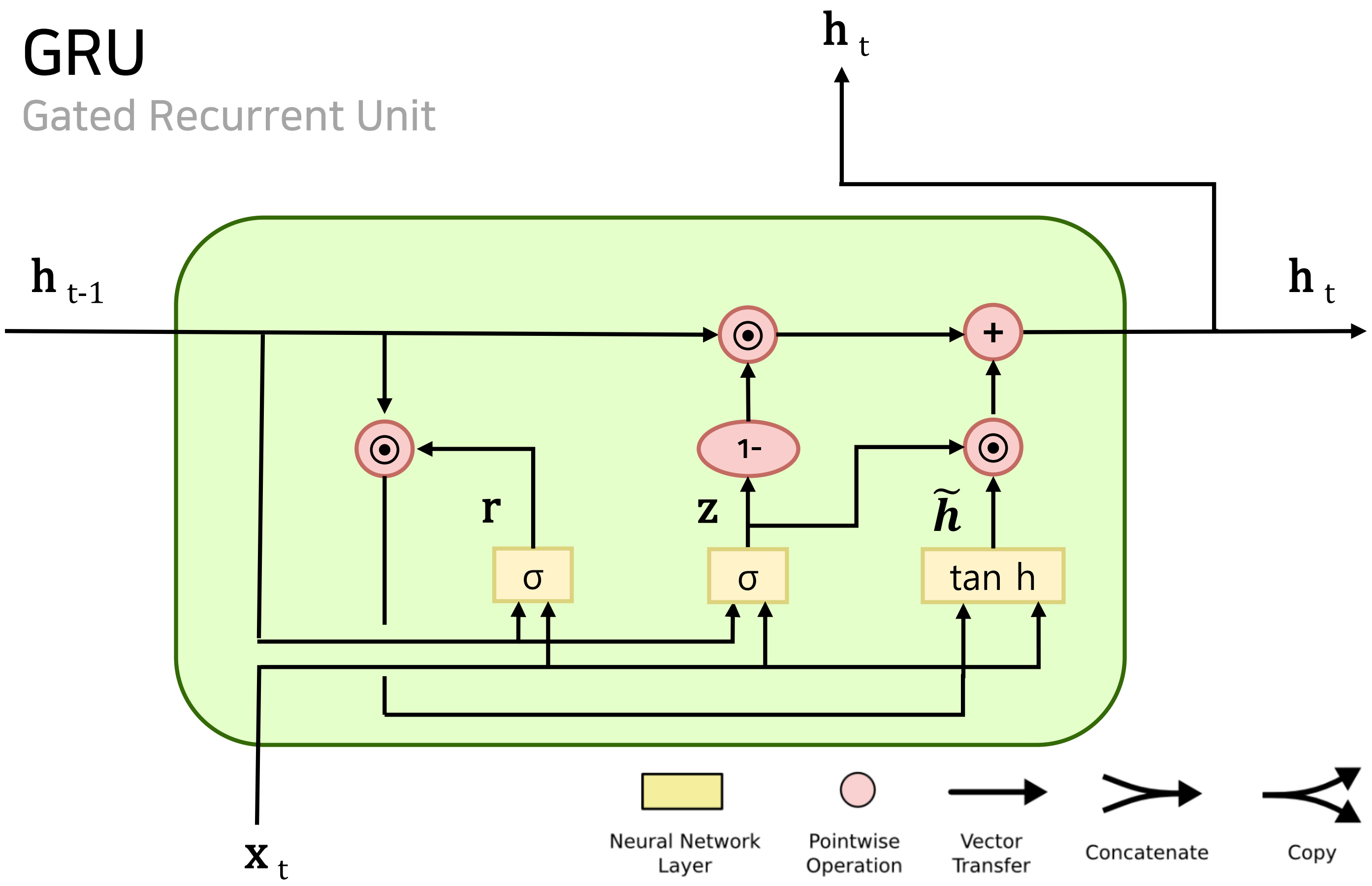
LSTM과 GRU Interface 비교





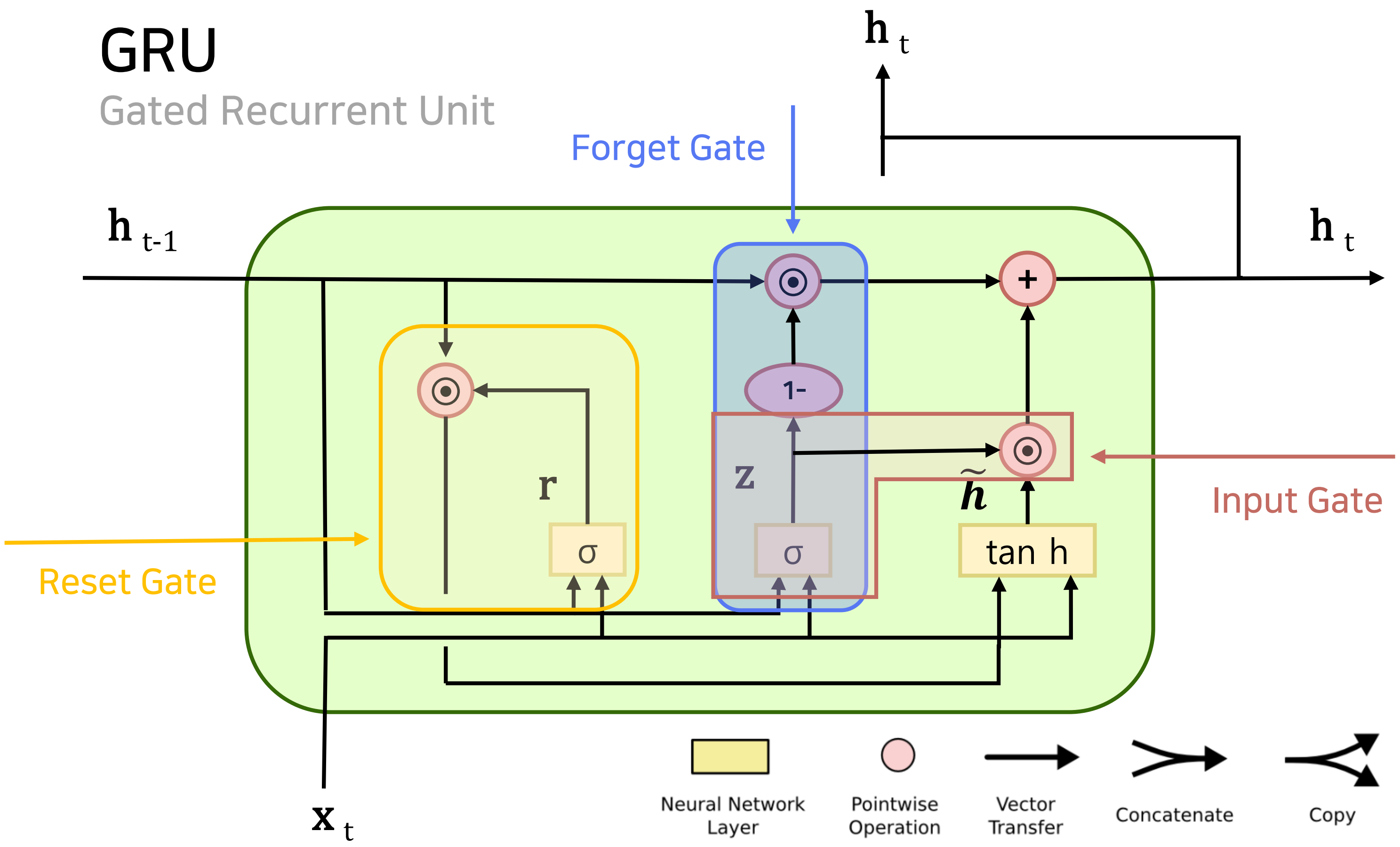
Gated Recurrent Unit (GRU)

GRU의 계산 그래프



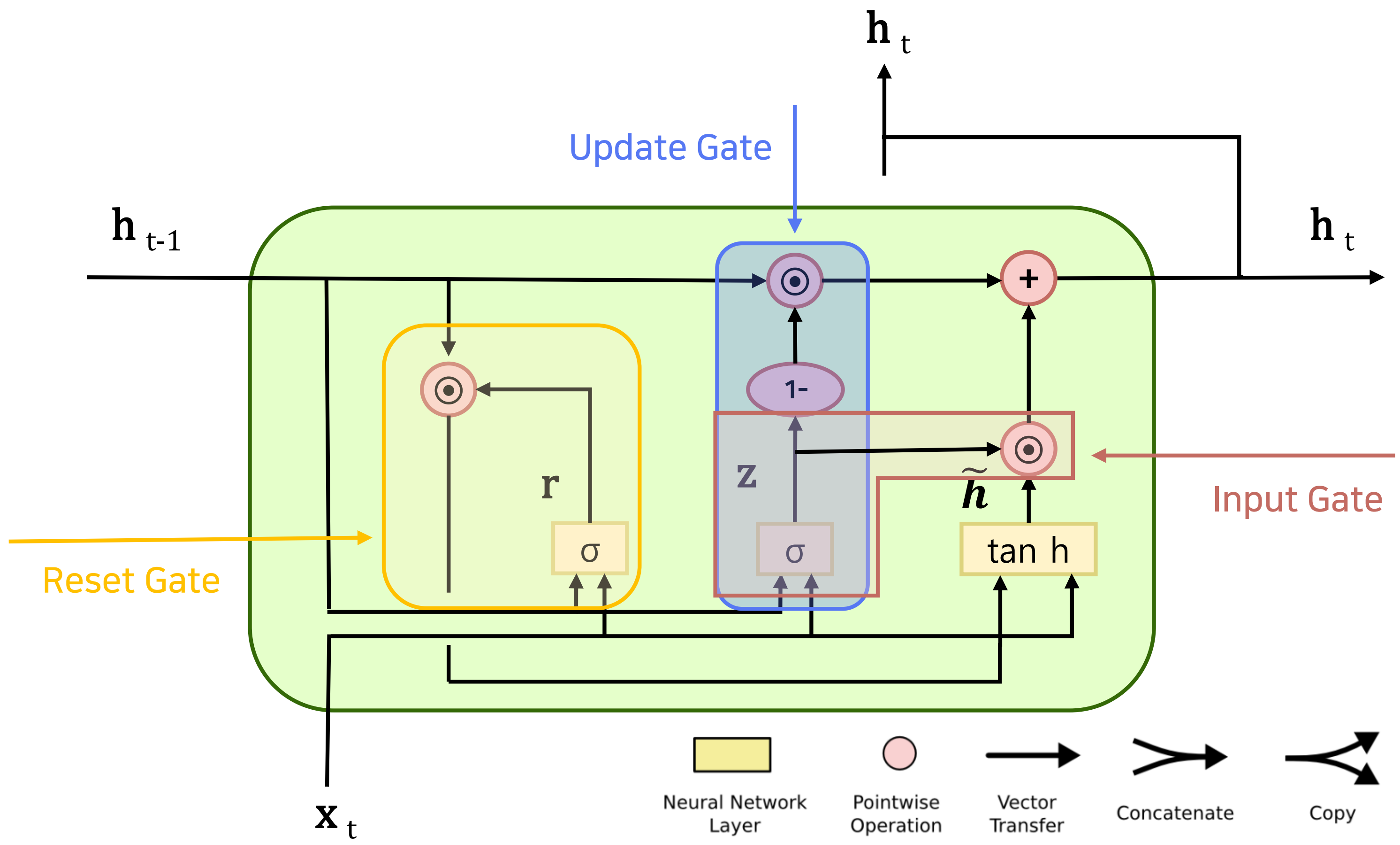
Gated Recurrent Unit (GRU)

GRU의 Forget Gate 와 Input Gate



Gated Recurrent Unit (GRU)

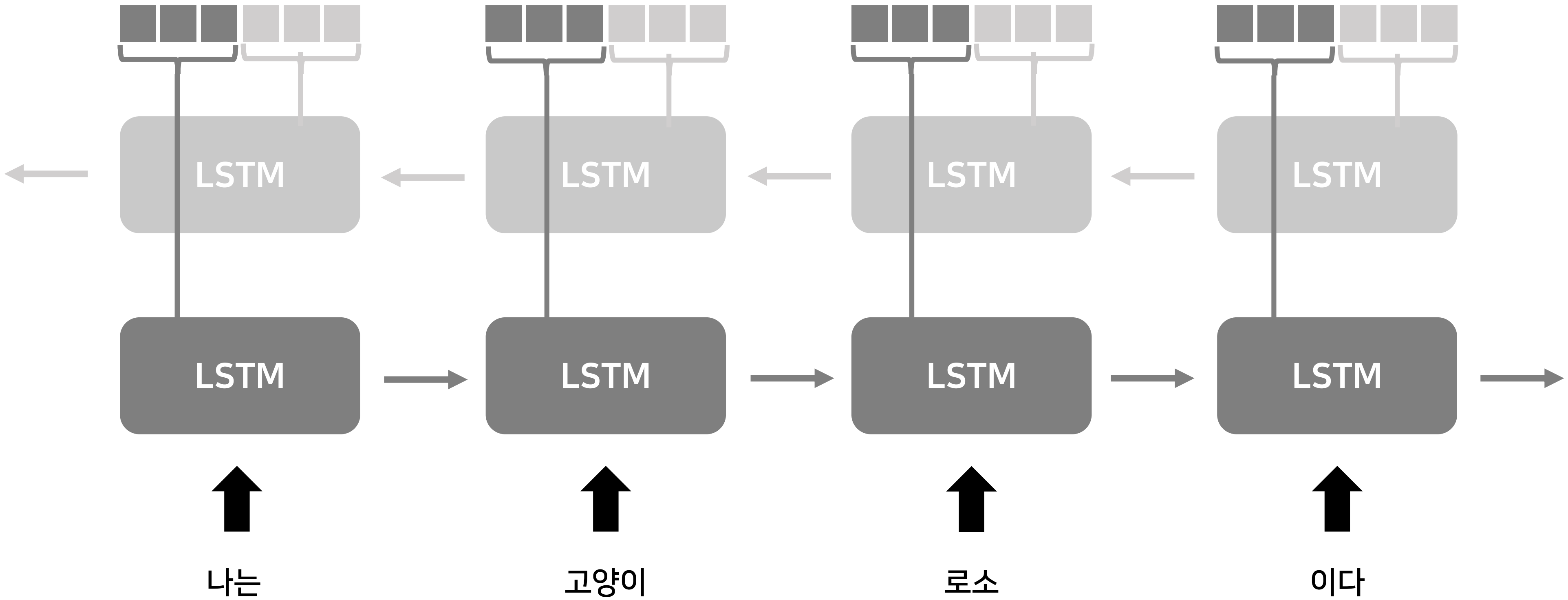
GRU의 Forget Gate 와 Input Gate



BLSTM

Bidirection LSTM

BLSTM



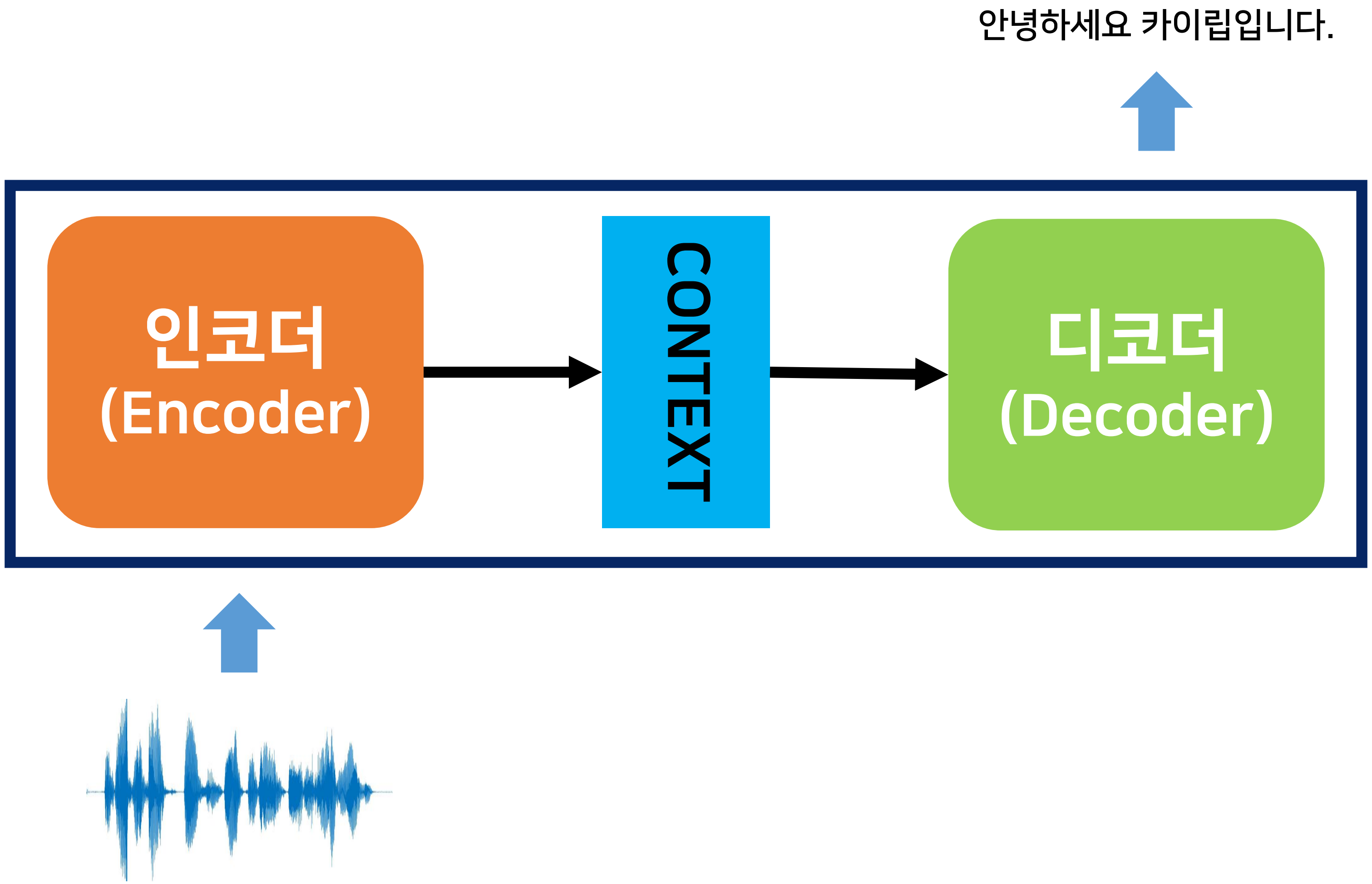
seq2seq

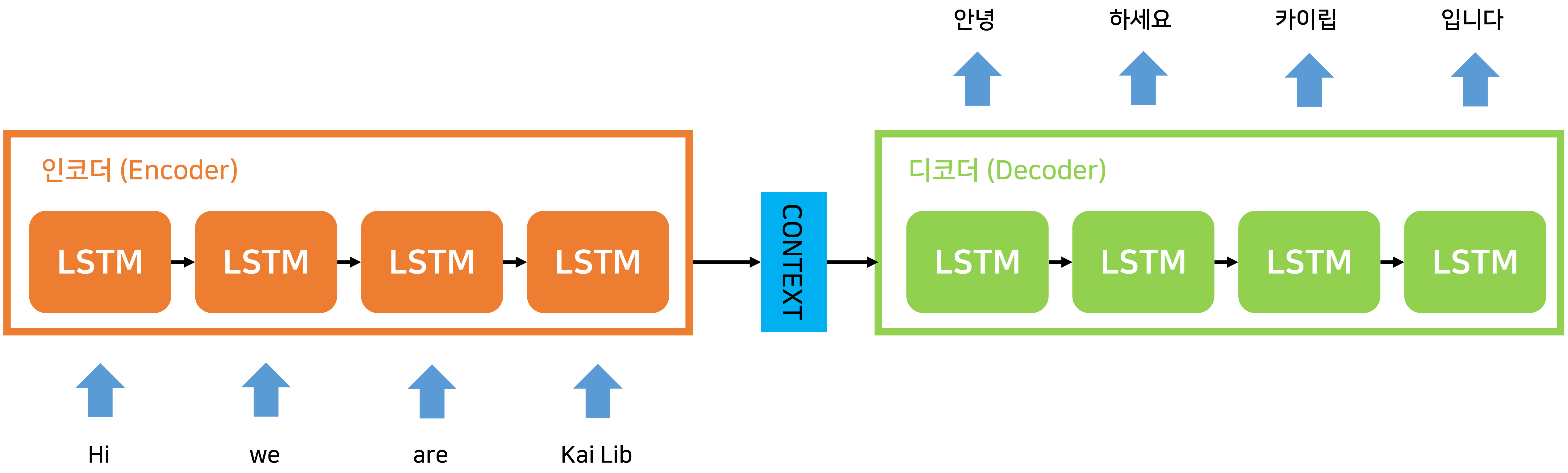
Encoder와 Decoder가 음성인식을 수행하는 예

안녕하세요 카이립입니다.

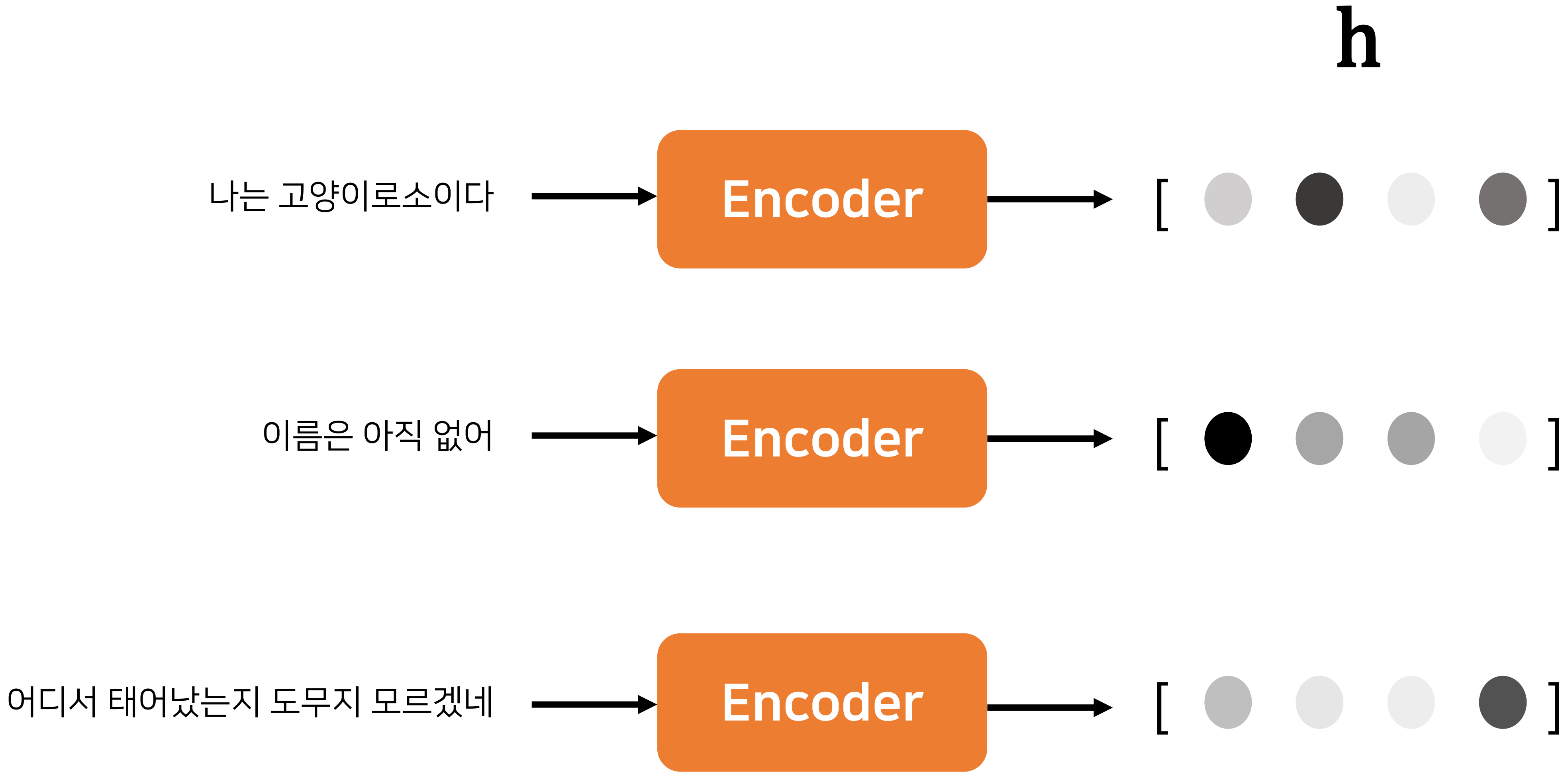
음성 인식기
(SEQUENCE TO SEQUENCE)

인코더의 셀은 주황색 디코더의 색은 초록색으로 표현

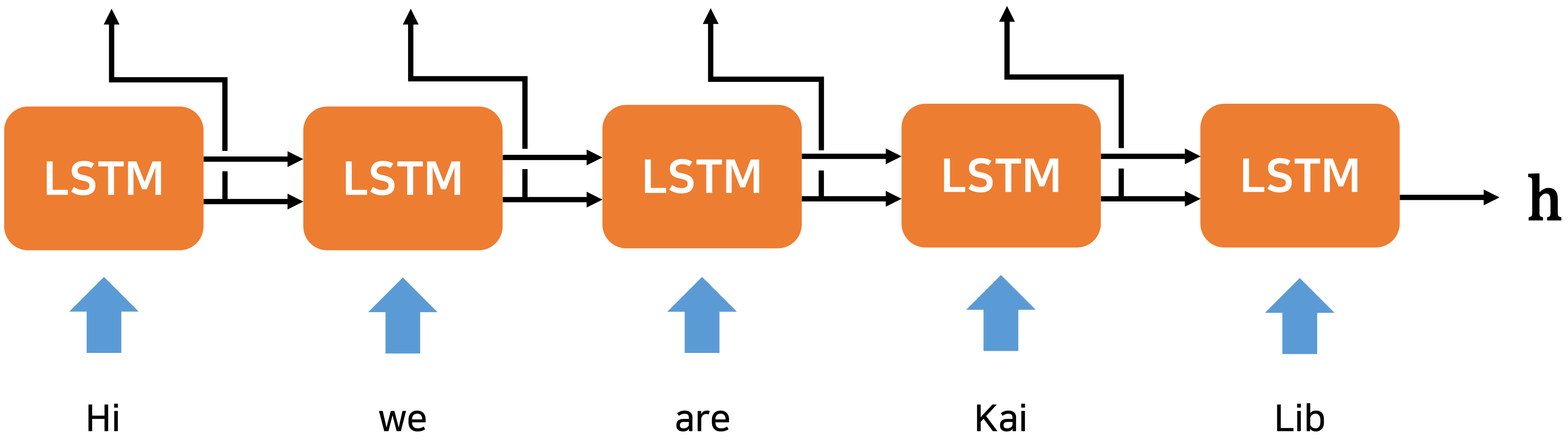



안녕
하세요
카이립
입니다
인코더 (Encoder)
LSTM
LSTM
LSTM
LSTM
CONTEXT
디코더 (Decoder)
LSTM
LSTM
LSTM
LSTM
Hi
we
are
Kai Lib

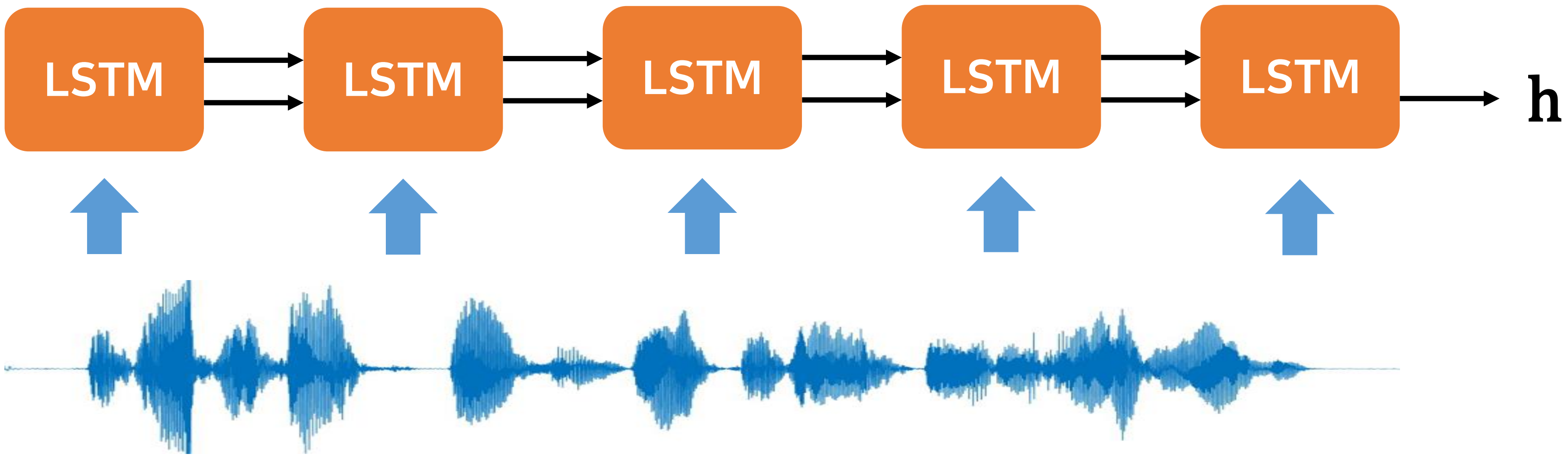
Encoder는 문장을 고정 길이 벡터로 인코딩한다.



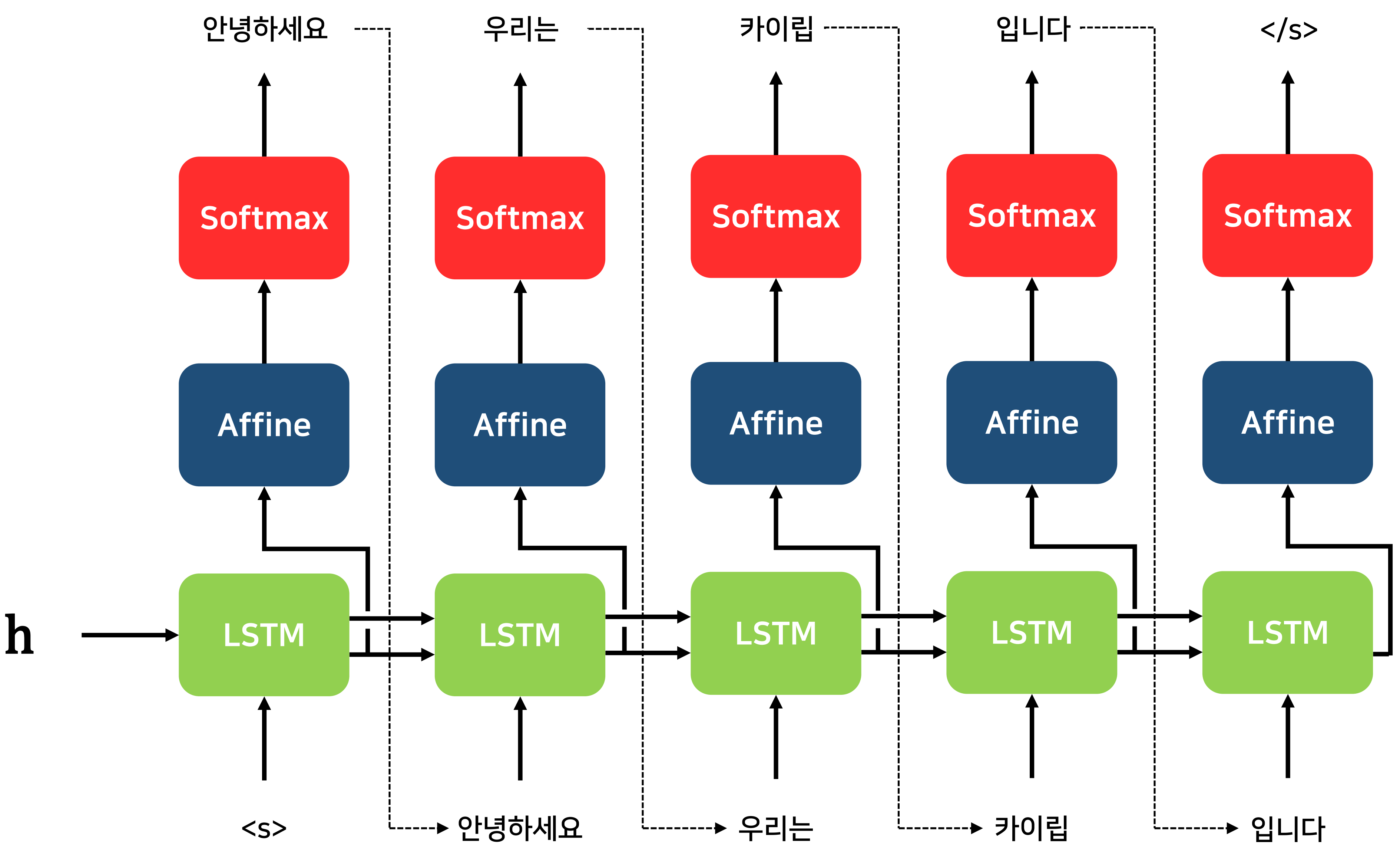
Encoder를 구성하는 계층



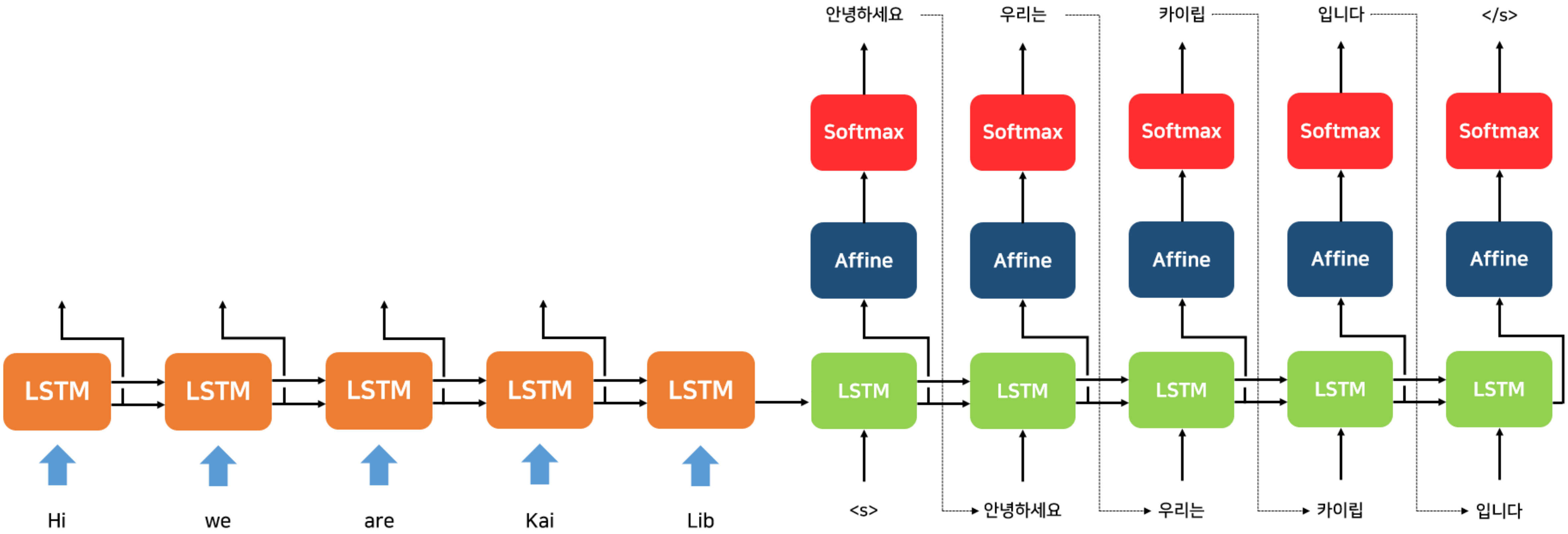
Encoder를 구성하는 계층



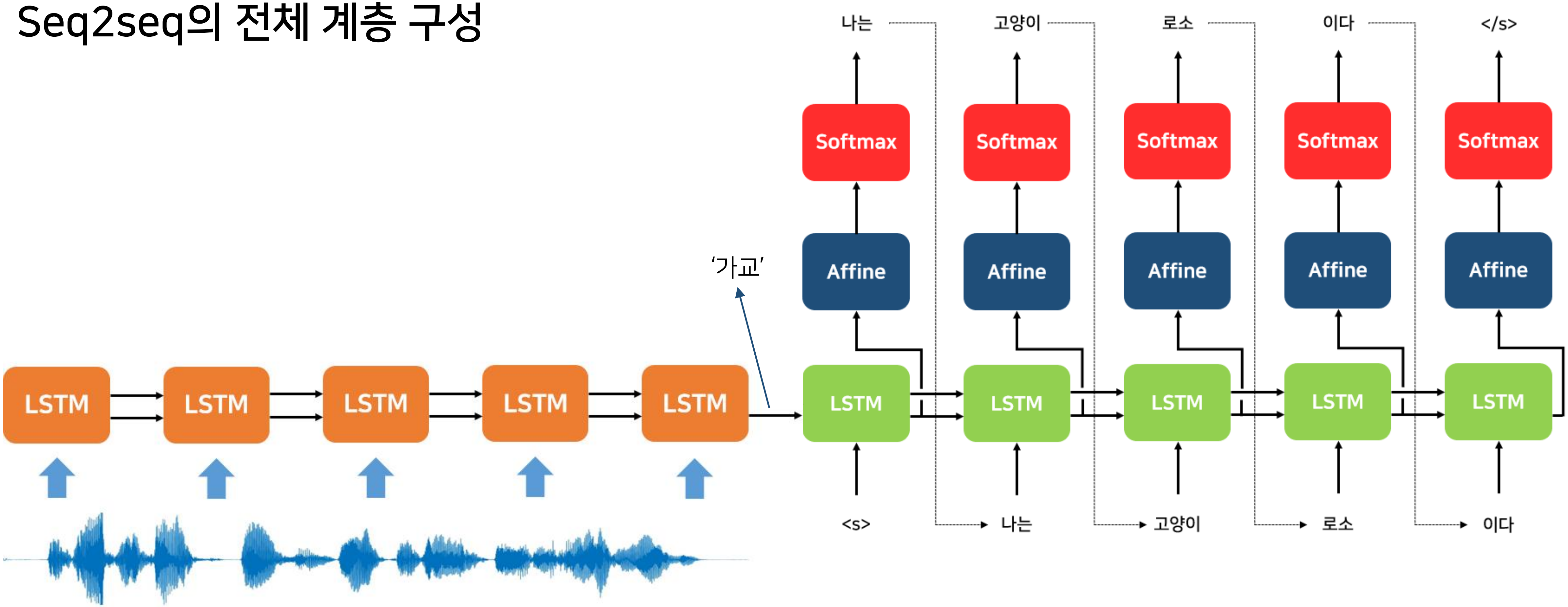
Decoder를 구성하는 계층



Seq2seq의 전체 계층 구성



Seq2seq의 전체 계층 구성

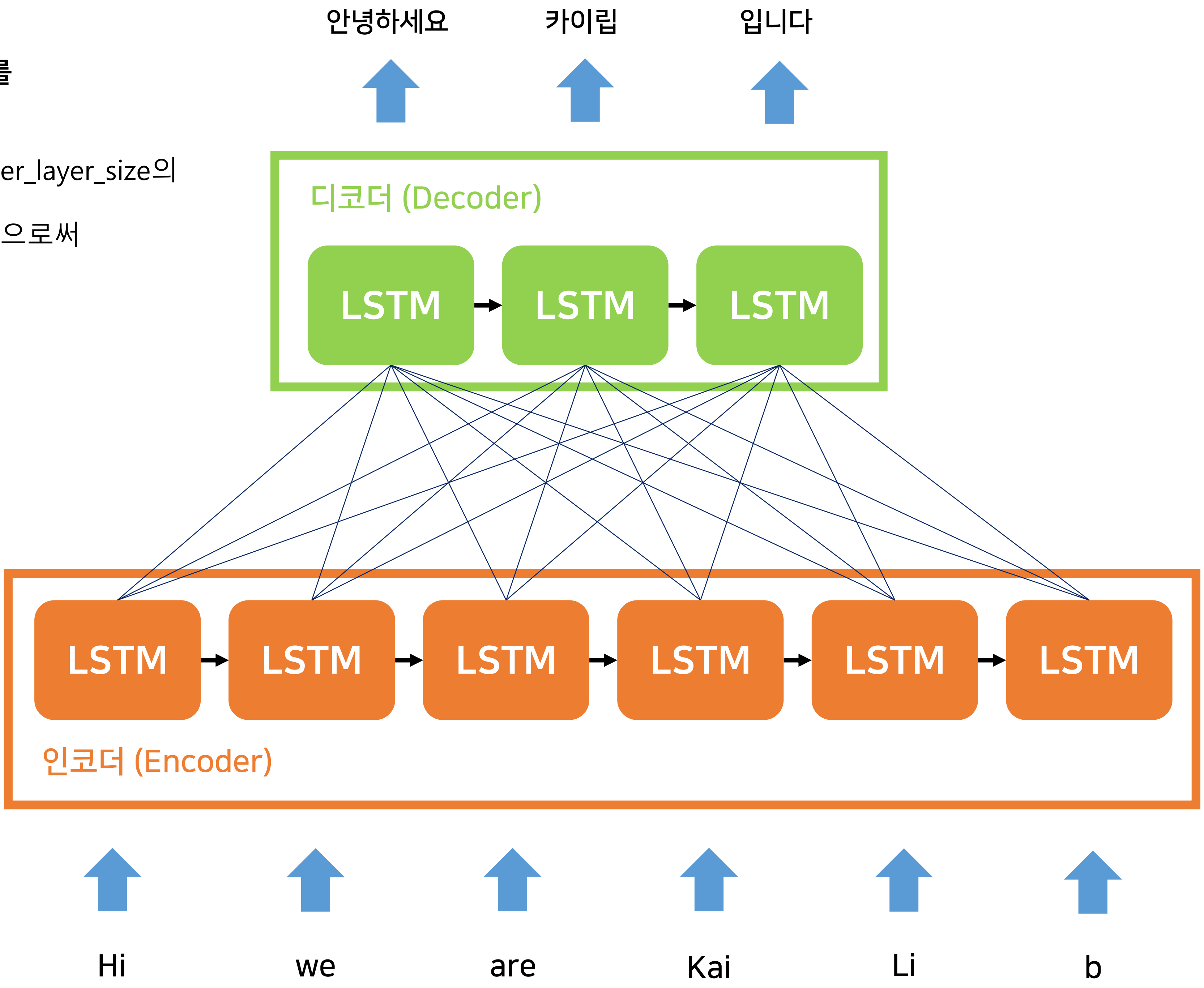


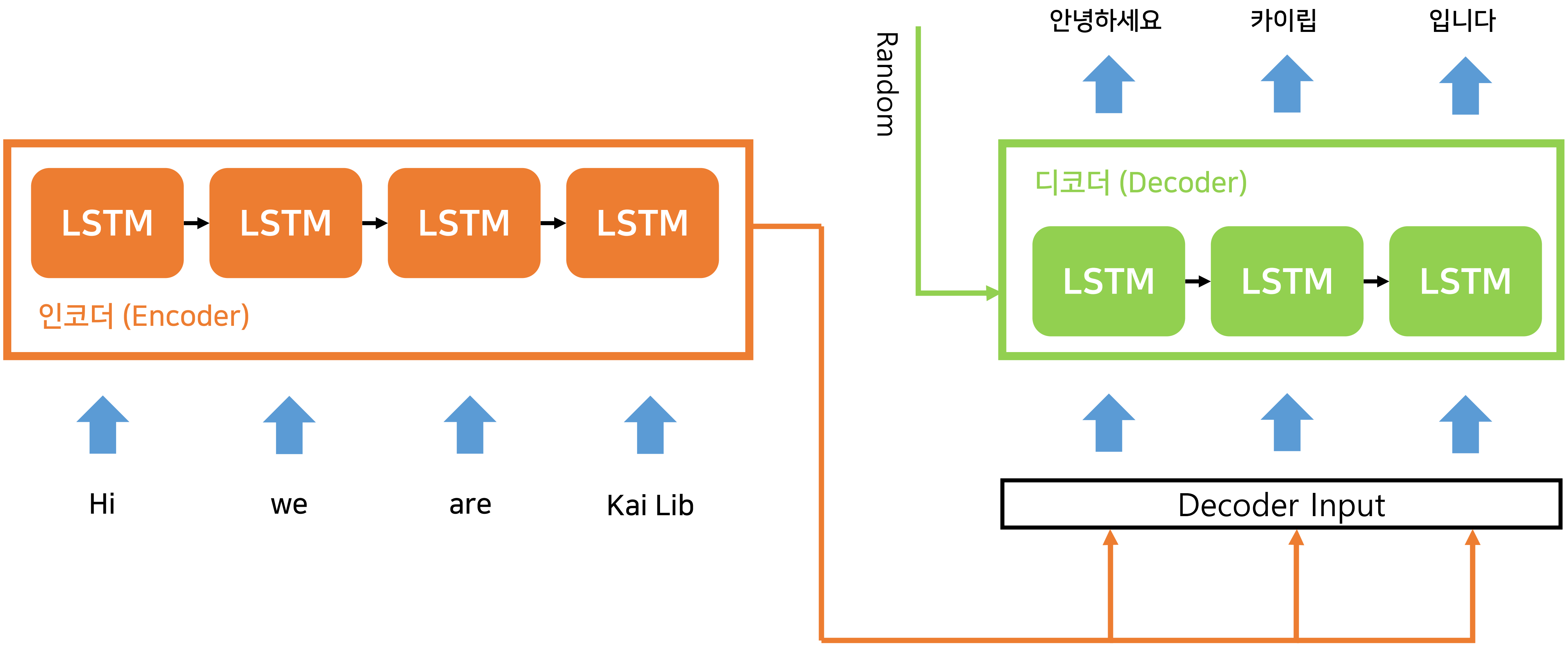
Seq2seq

Decoder init state

Single Fully Connected Network를
Encoder와 Decoder 사이에 배치

Encoder_layer_size의 인풋과 decoder_layer_size의
아웃풋을 가지는
Fully Connected Network를 배치함으로써
서로 다른 사이즈의 인코더의
Hidden State를 이용하여 디코더의
Hidden State를 초기화 할 수 있다.



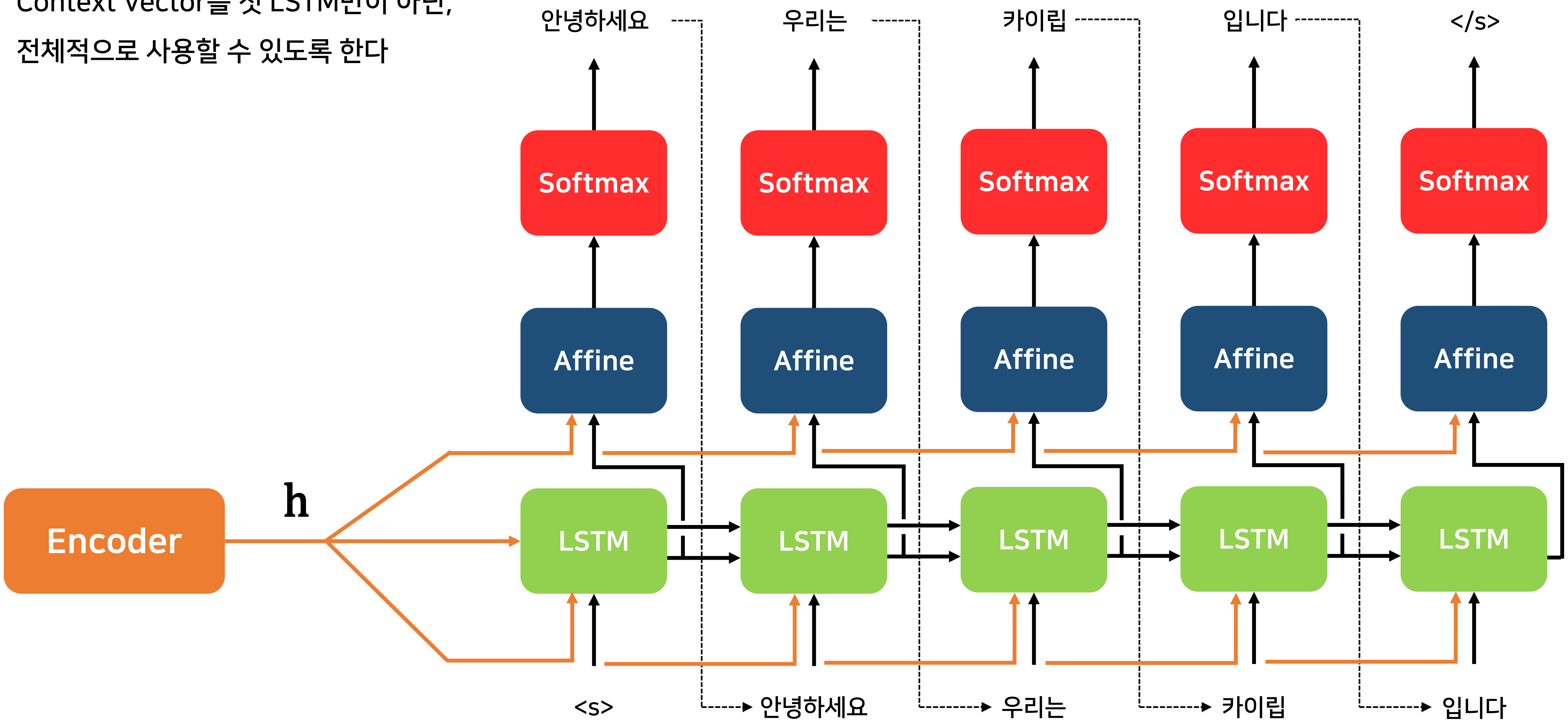


디코더의 Hidden State는 랜덤으로 초기화 한 후,
인코더의 Hidden State Output(Last Hidden State)을
디코더의 인풋에 **concatenate**한다.

Peeky Seq2seq

Context Vector를 첫 LSTM만이 아닌,

전체적으로 사용할 수 있도록 한다

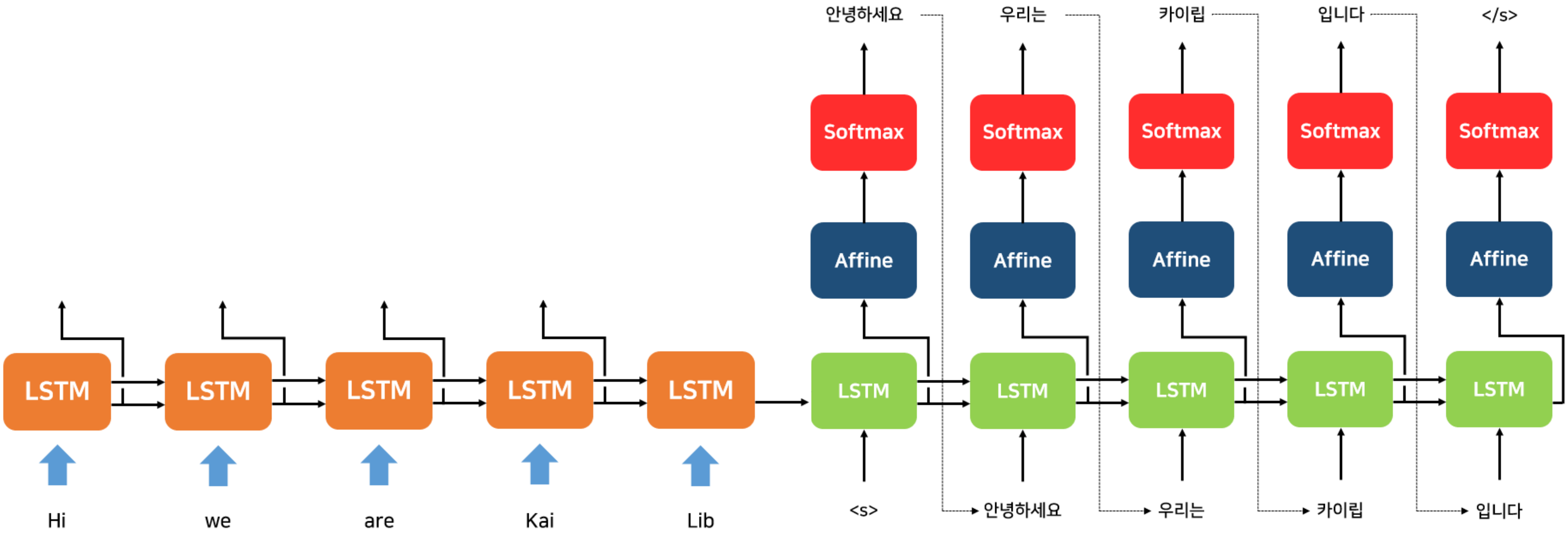


Seq2seq
+ Attention

Basic Seq2seq의 한계

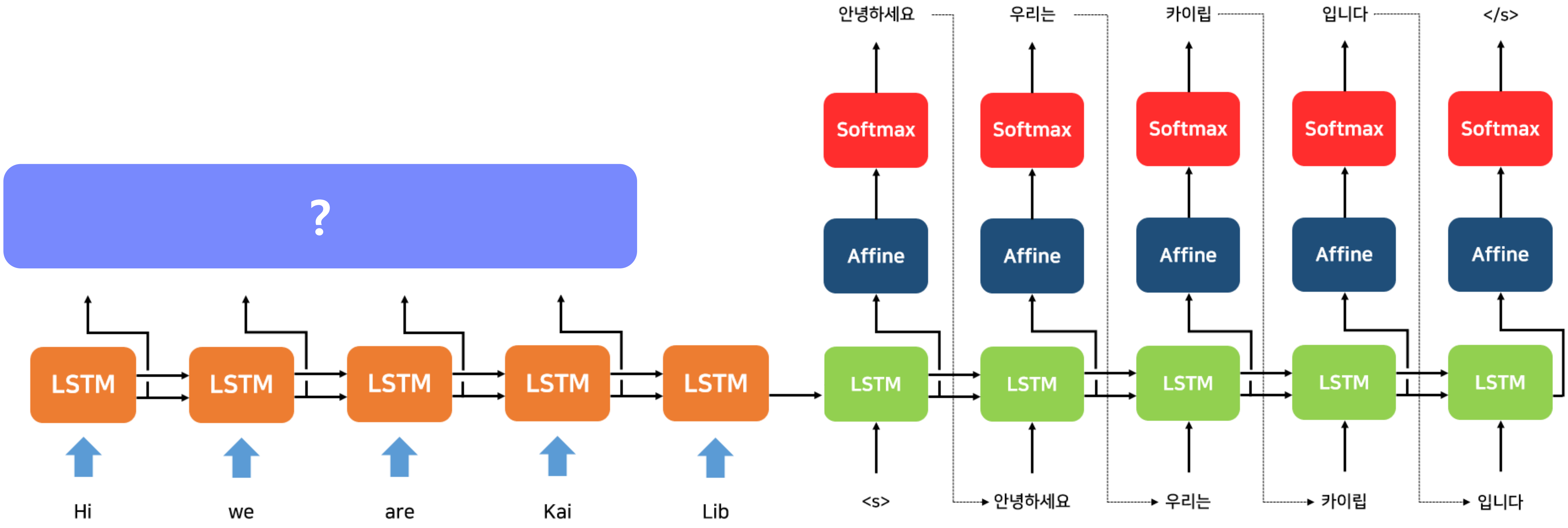
1) 아무리 긴 입력 시퀀스가 오더라도 고정 길이의 벡터만을 출력

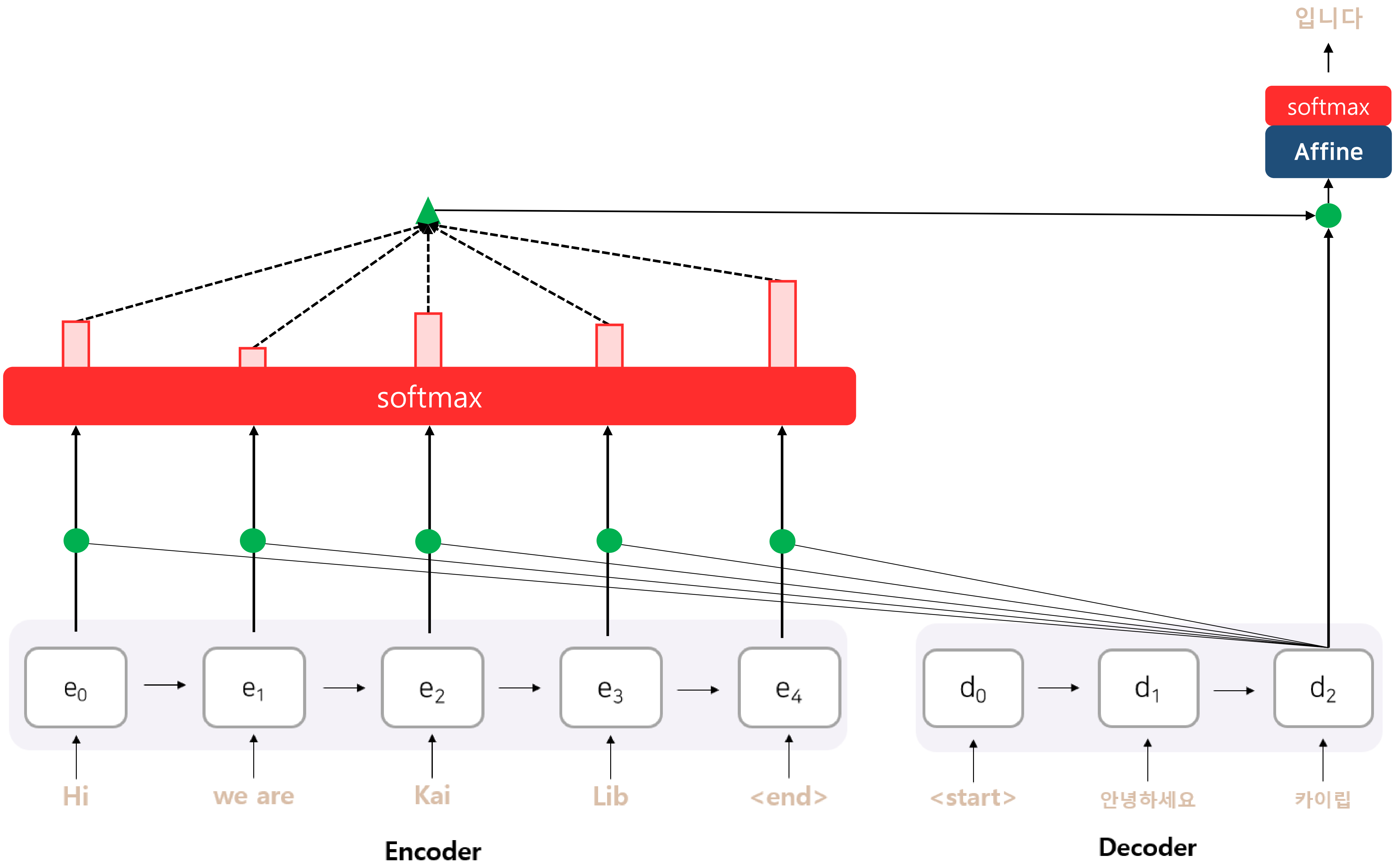
2) RNN의 고질적인 문제인 Vanishing Gradient 문제 발생



Basic Seq2seq의 한계

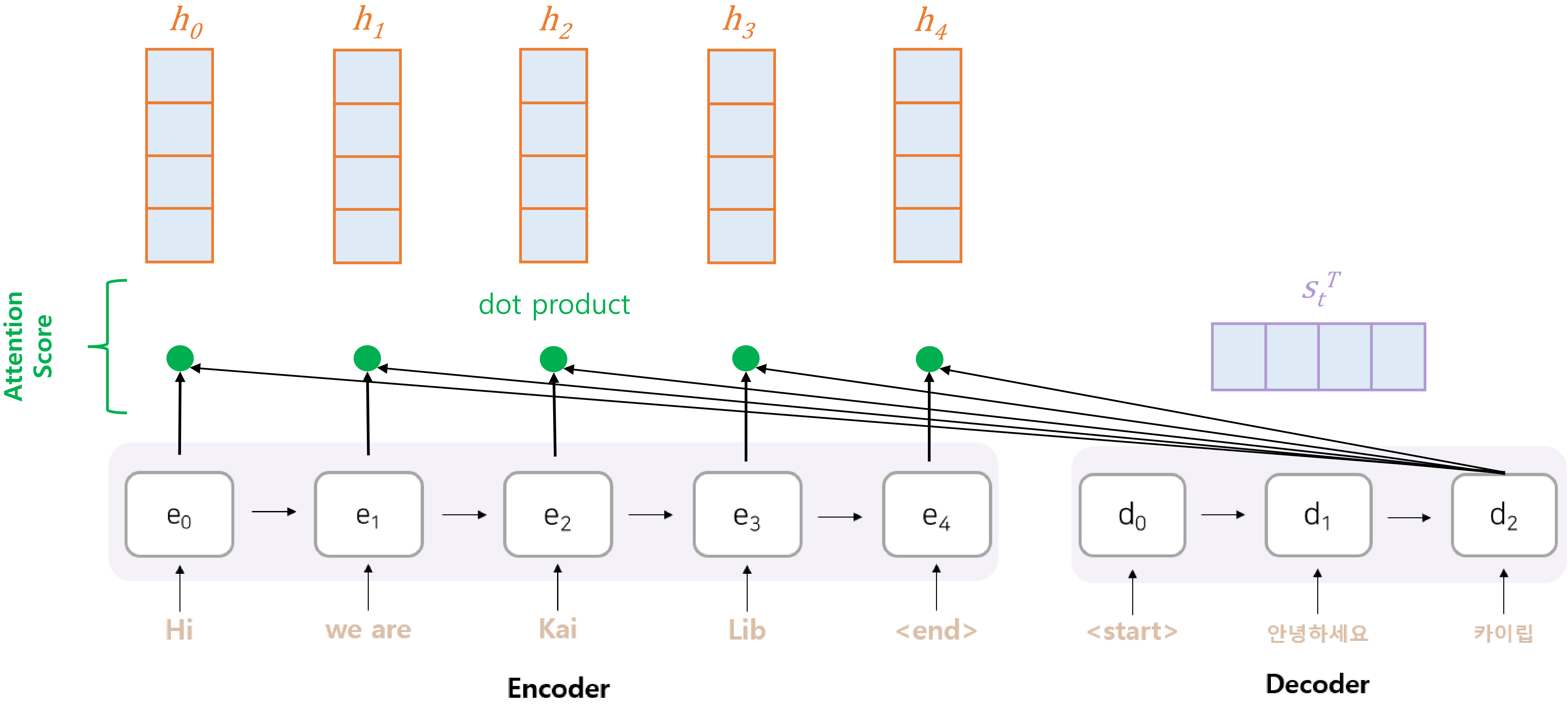
기껏 계산해 놓은 RNN의 Hidden State들은 쓰이지를 않는다





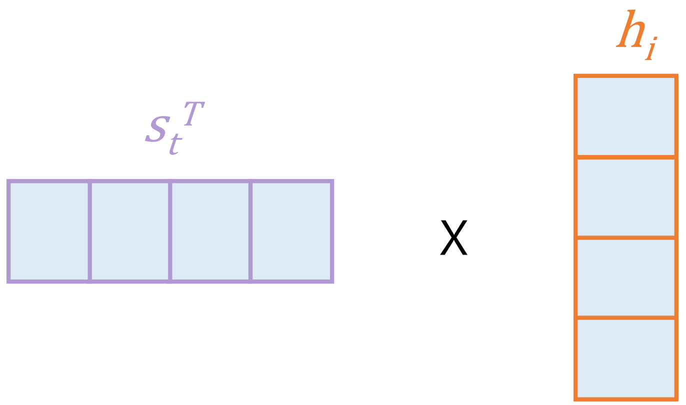
입니다
softmax
Affine
softmax
e0
e1
e2
e3
e4
d0
d1
d2
Hi
we are
Kai
Lib
<end>
<start>
안녕하세요
카이립
Encoder
Decoder

① Attention Score 계산

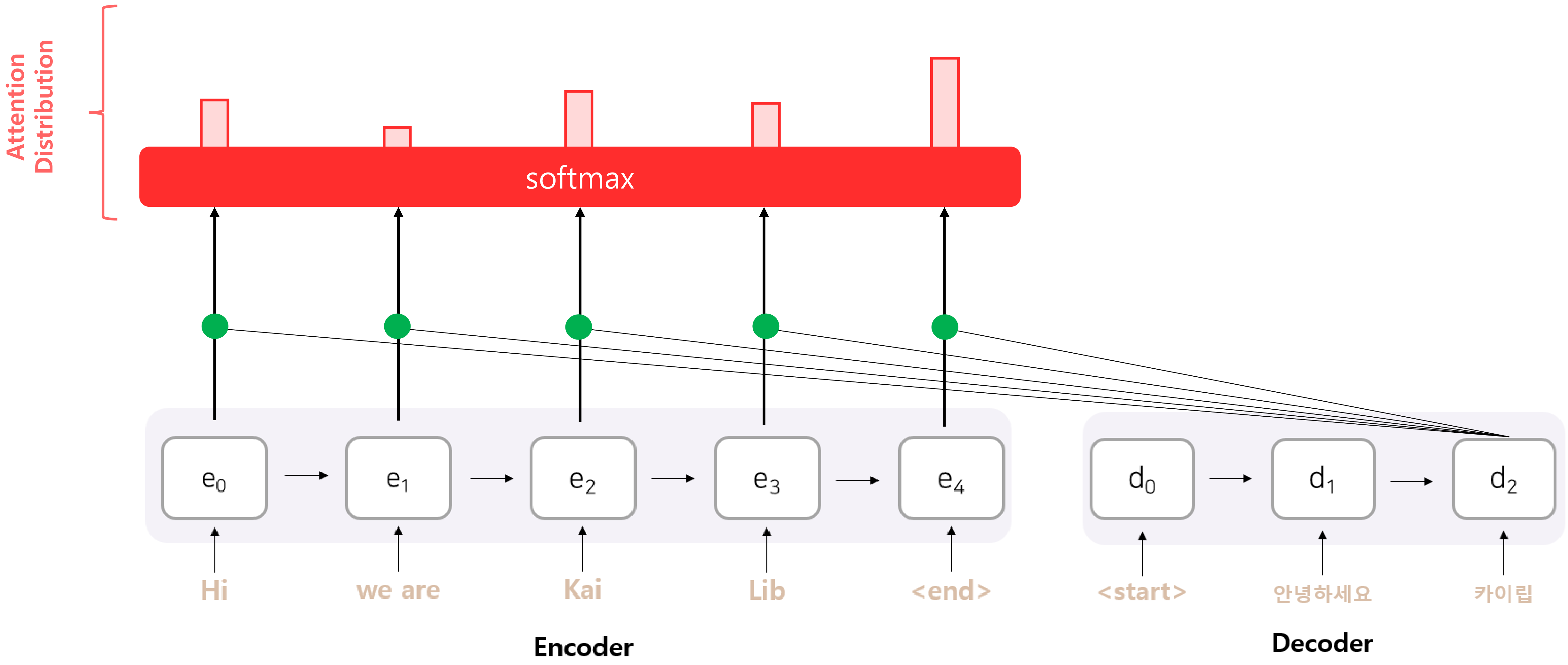


① Attention Score 계산

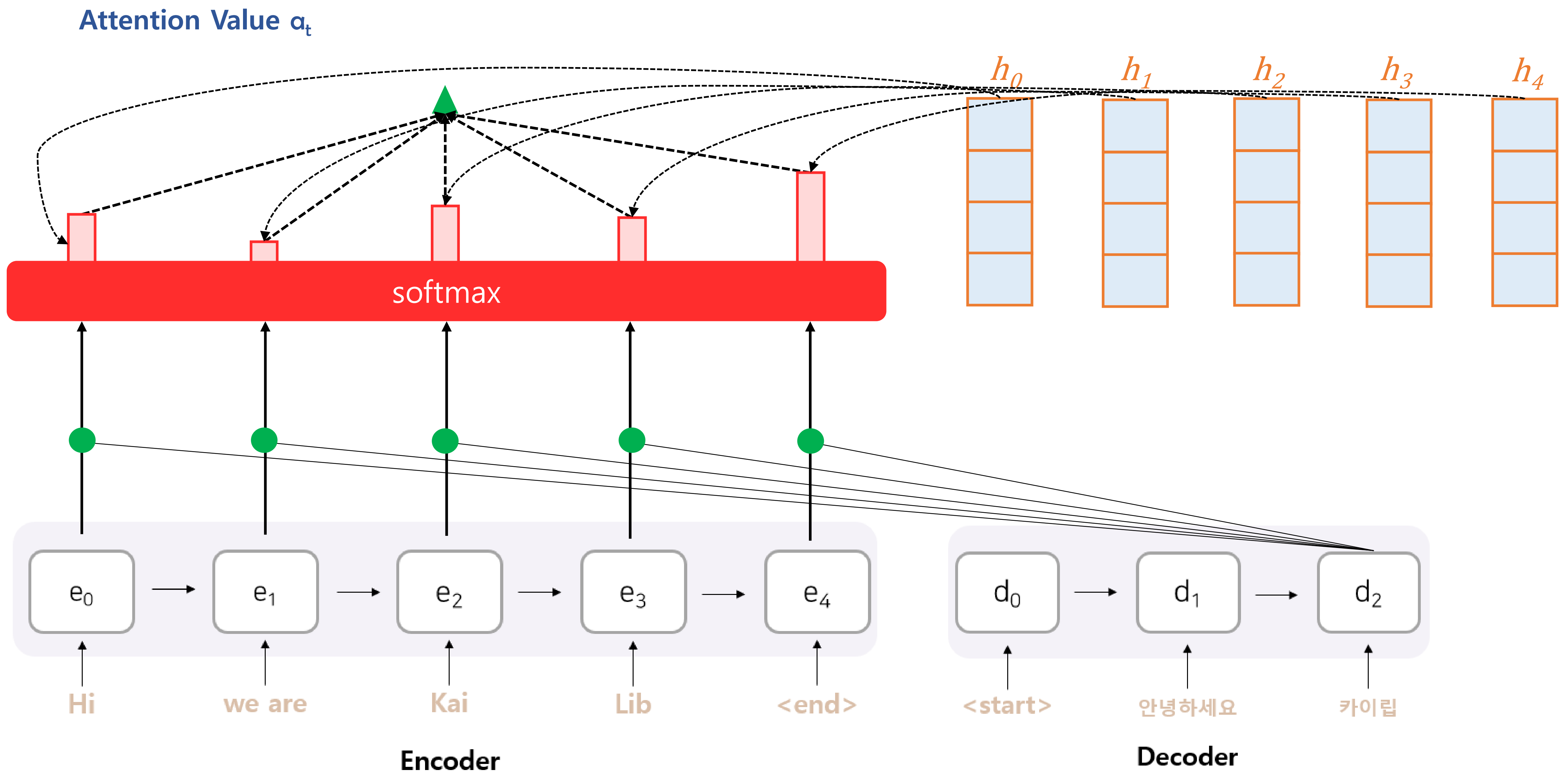
Dot-Product Attention



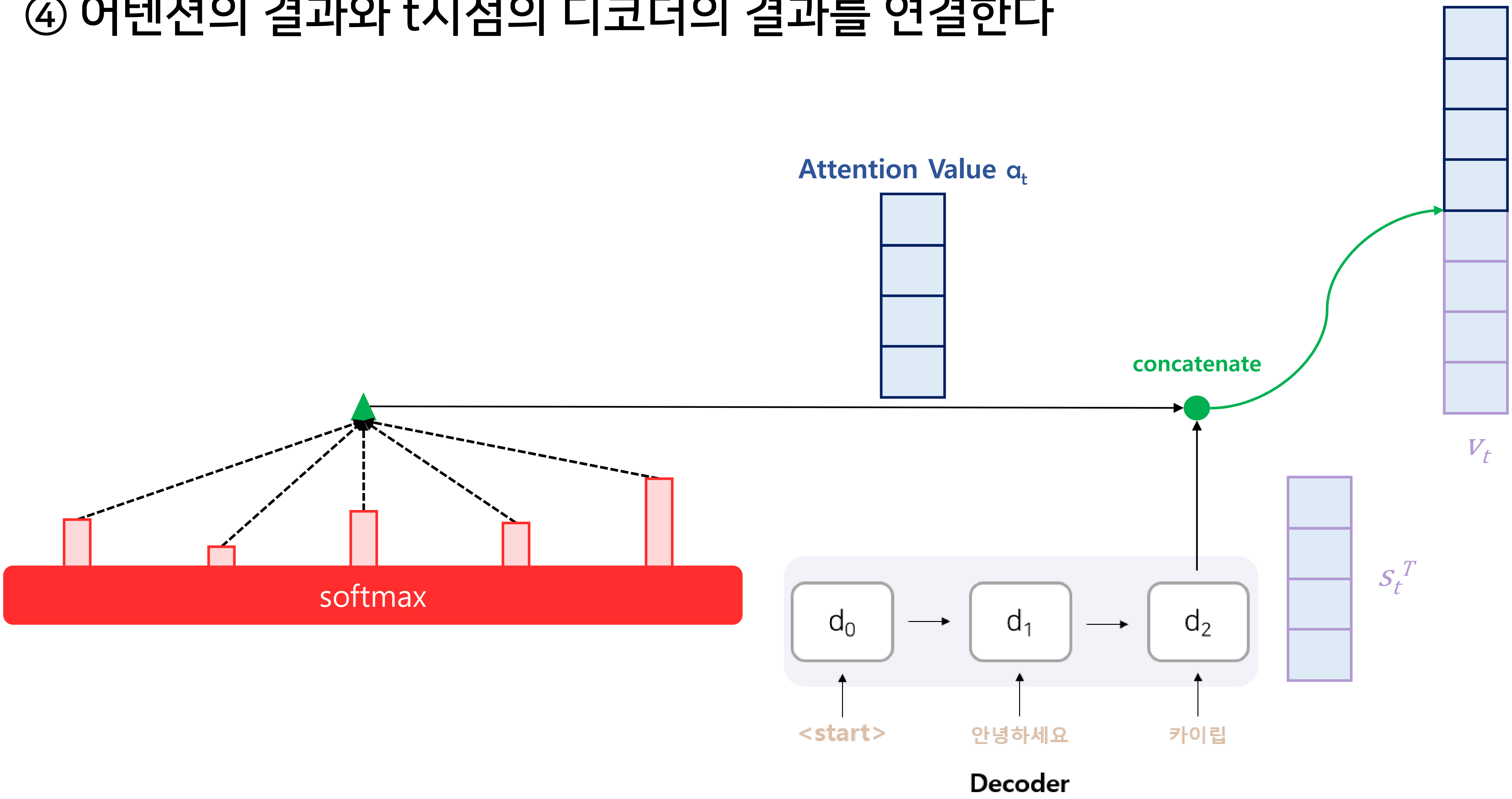
② 소프트맥스 함수를 통해 Attention Distribution을 구한다.

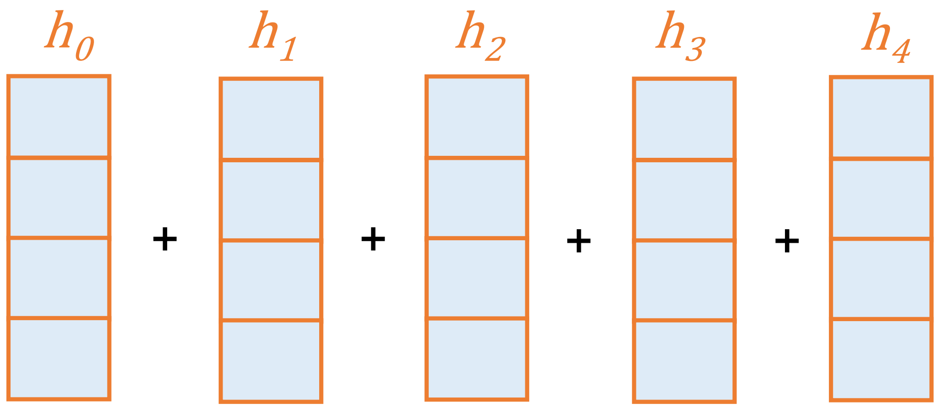


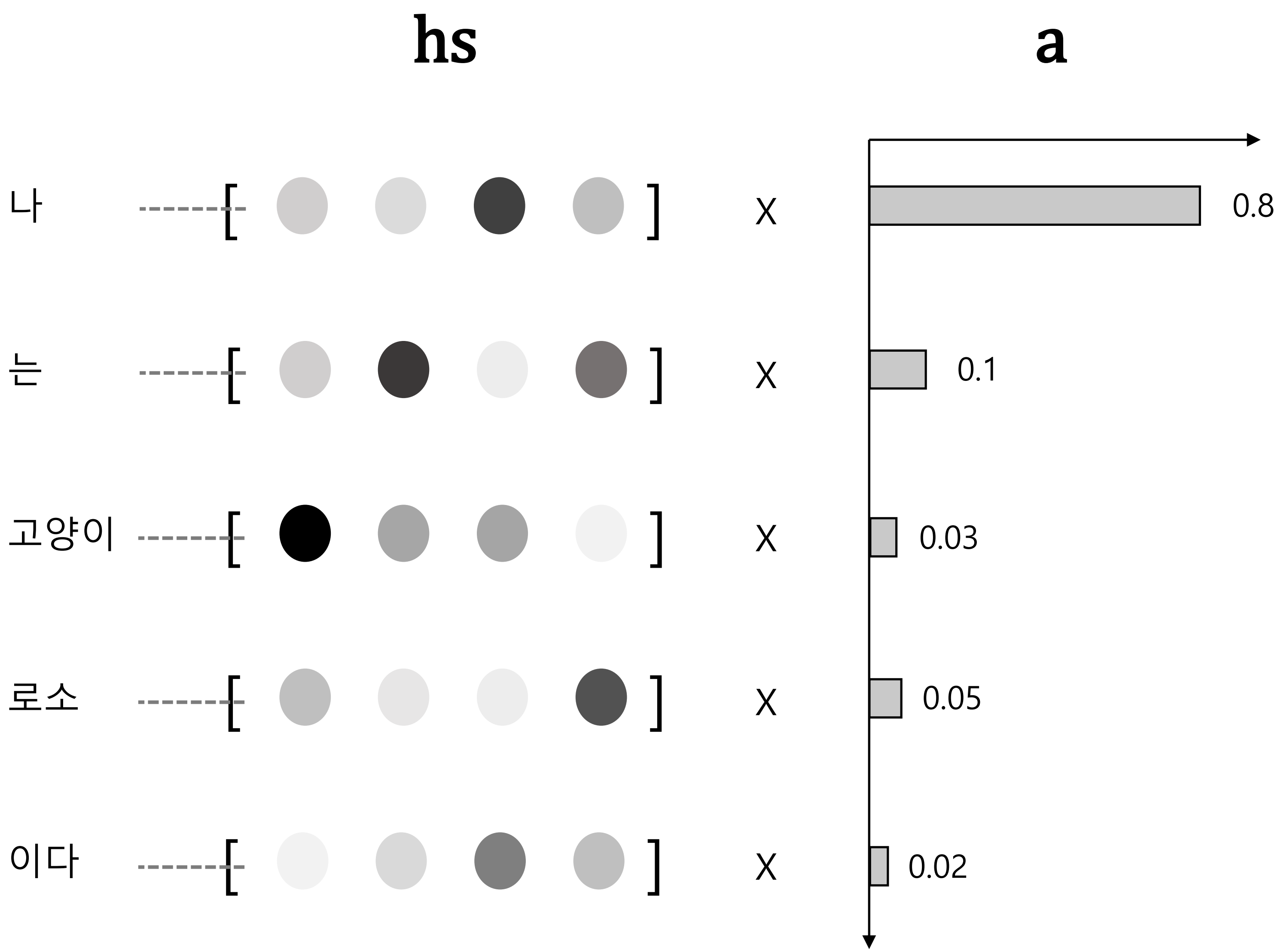
③ Attention Distribution과 인코더의 Hidden State들을 각각 곱한다

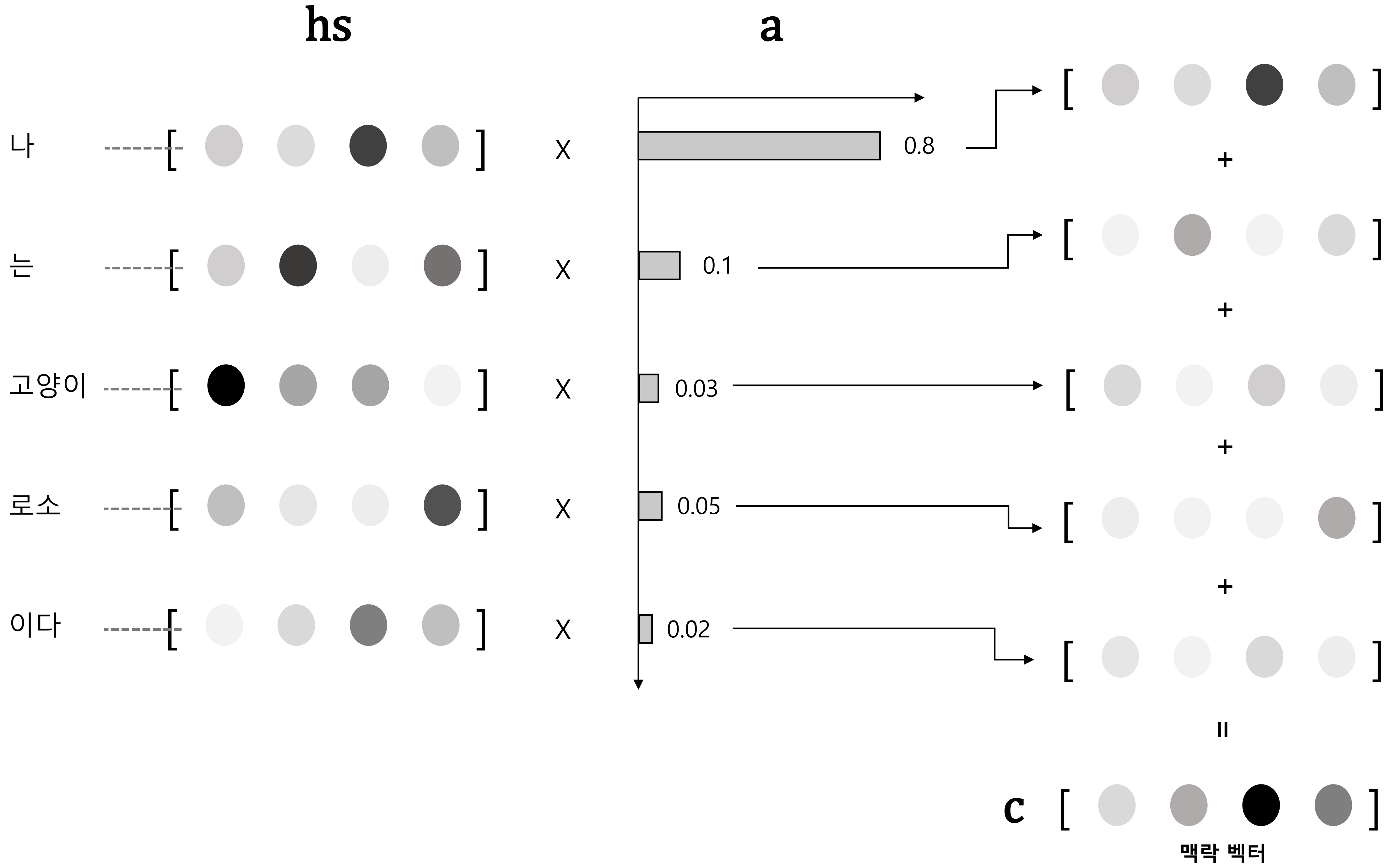


④ 어텐션의 결과와 t시점의 디코더의 결과를 연결한다

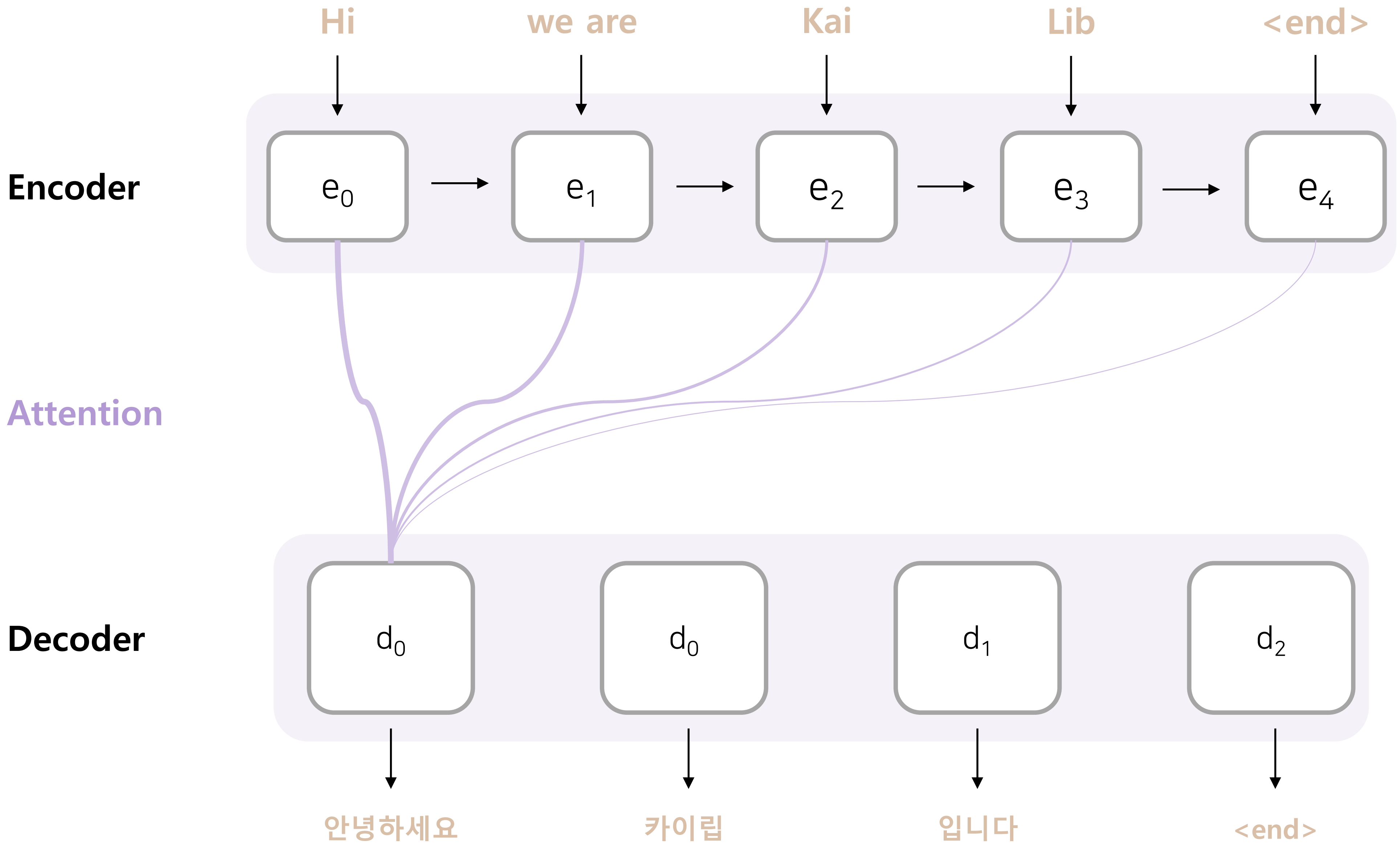



h_0 + h_1 + h_2 + h_3 + h_4


hs
a
나
는
고양이
로소
이다
X
X
X
X
X
0.8
0.1
0.03
0.05
0.02

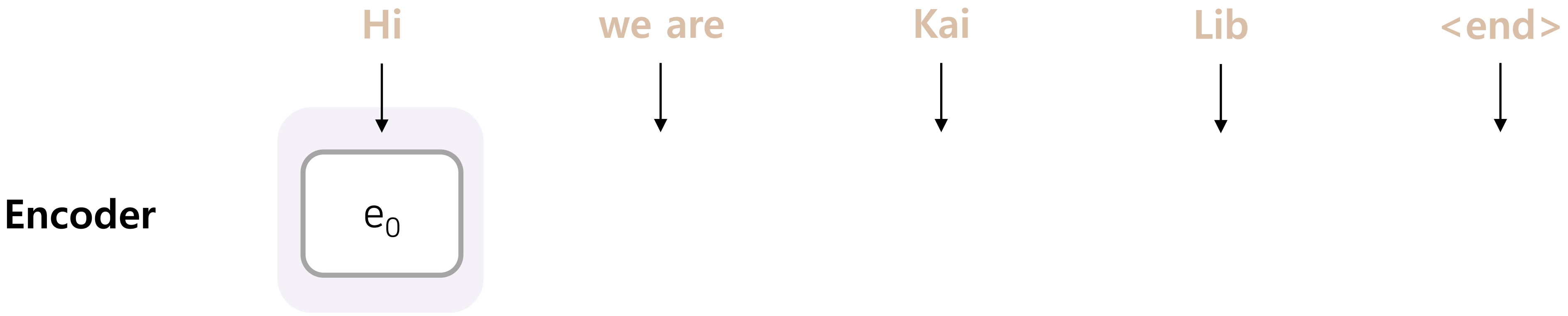


hs
a
나
는
고양이
로소
이다
X
X
X
X
X
0.8
0.1
0.03
0.05
0.02
+
+
+
+
=
c
맥락 벡터

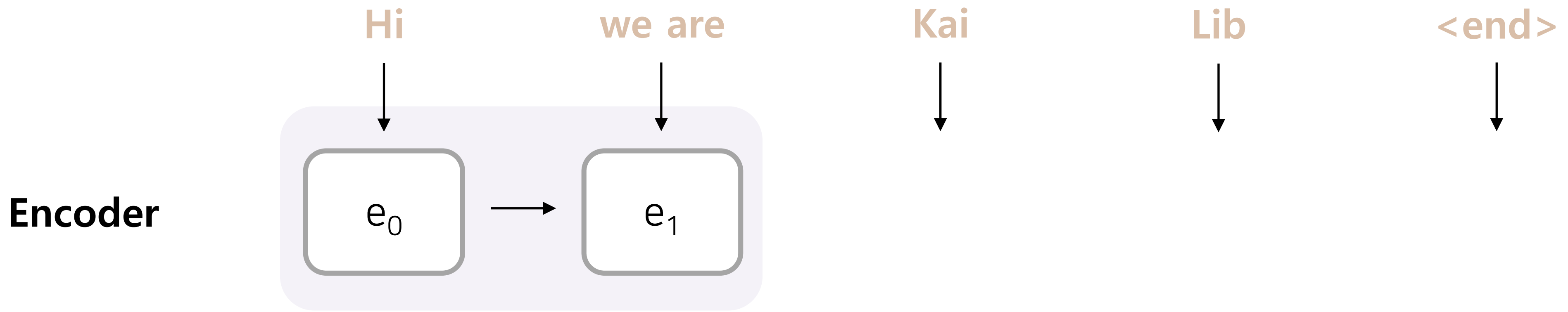


Hi
we are
Kai
Lib
<end>
Encoder
e_0
e_1
e_2
e_3
e_4
Attention
Decoder
d_0
d_0
d_1
d_2
안녕하세요
카이립
입니다
<end>

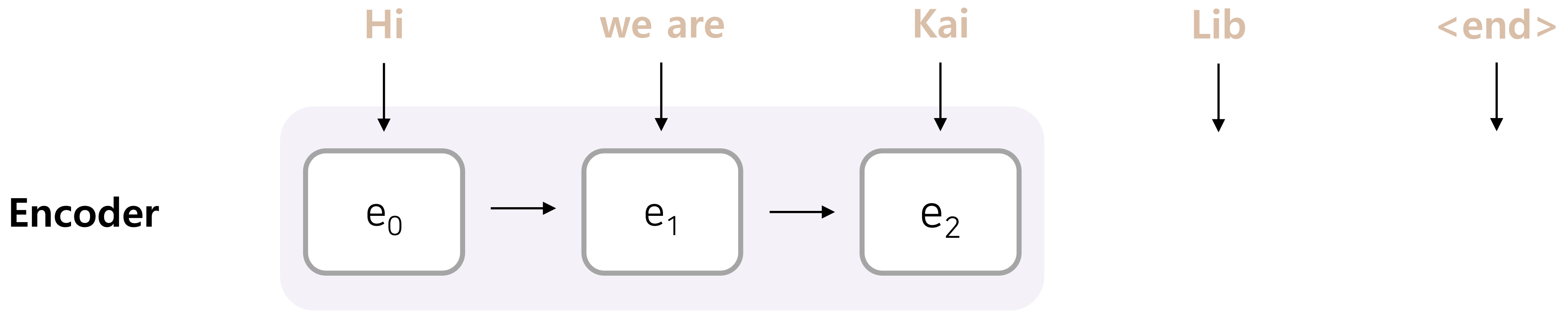
Seq2seq + Attention Step ①



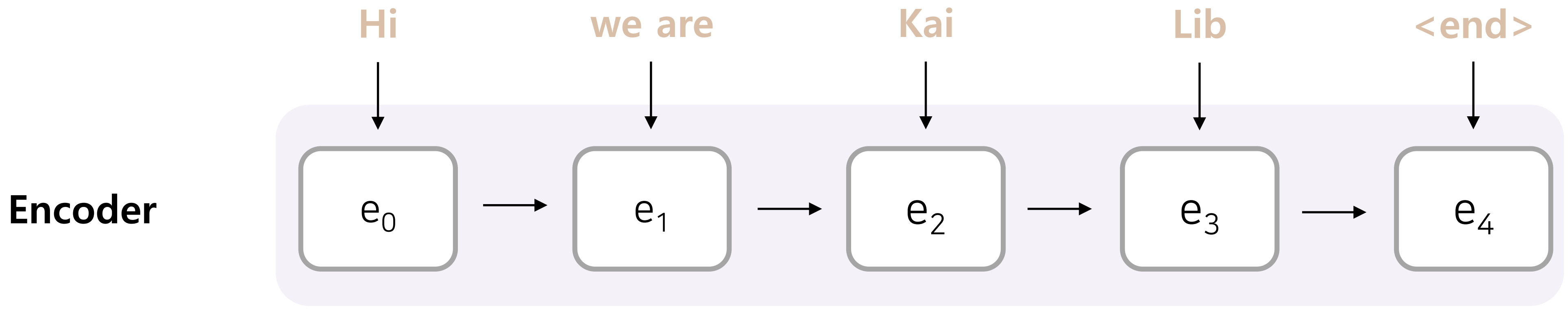
Seq2seq + Attention Step ②



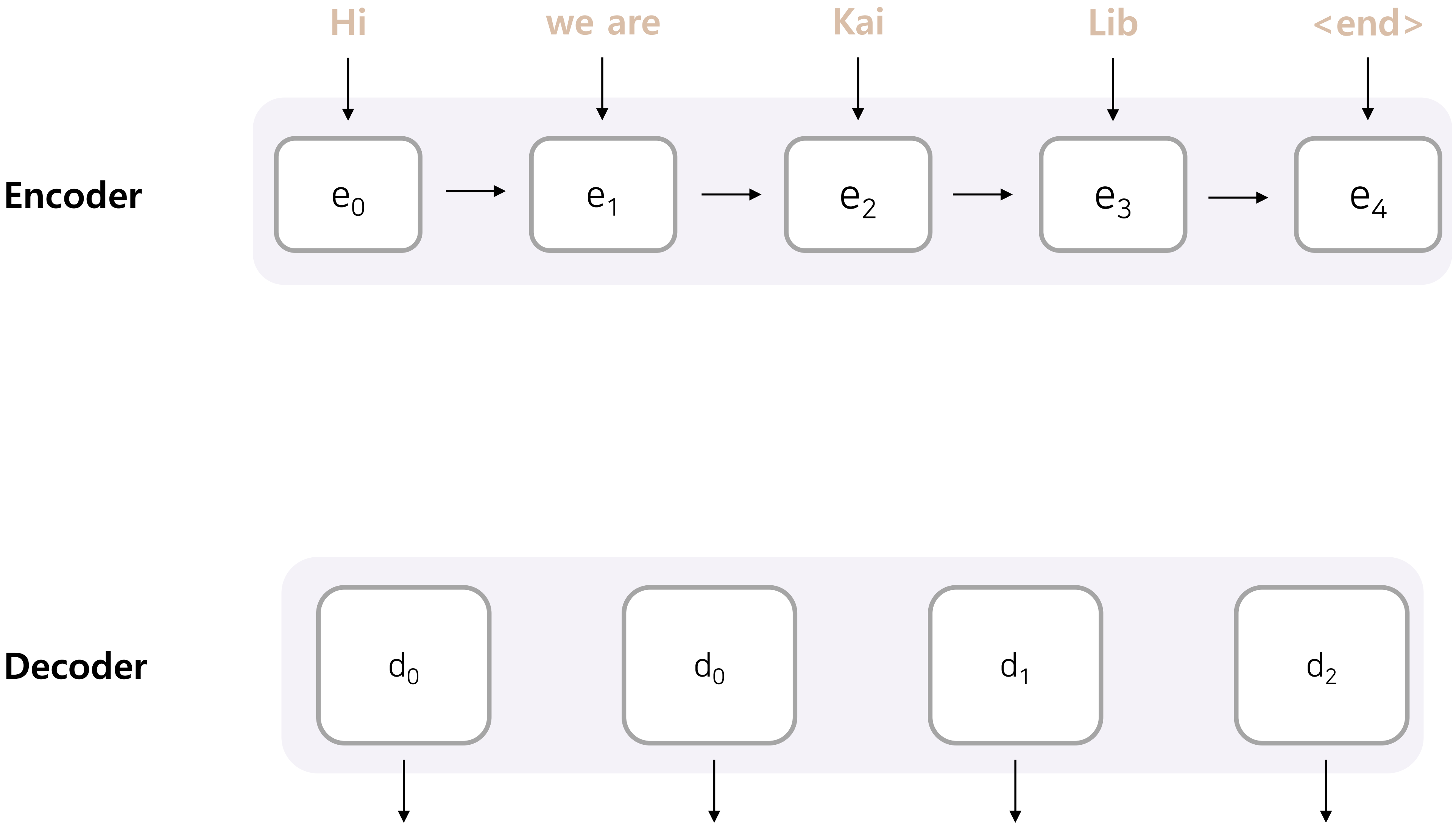
Seq2seq + Attention Step ③



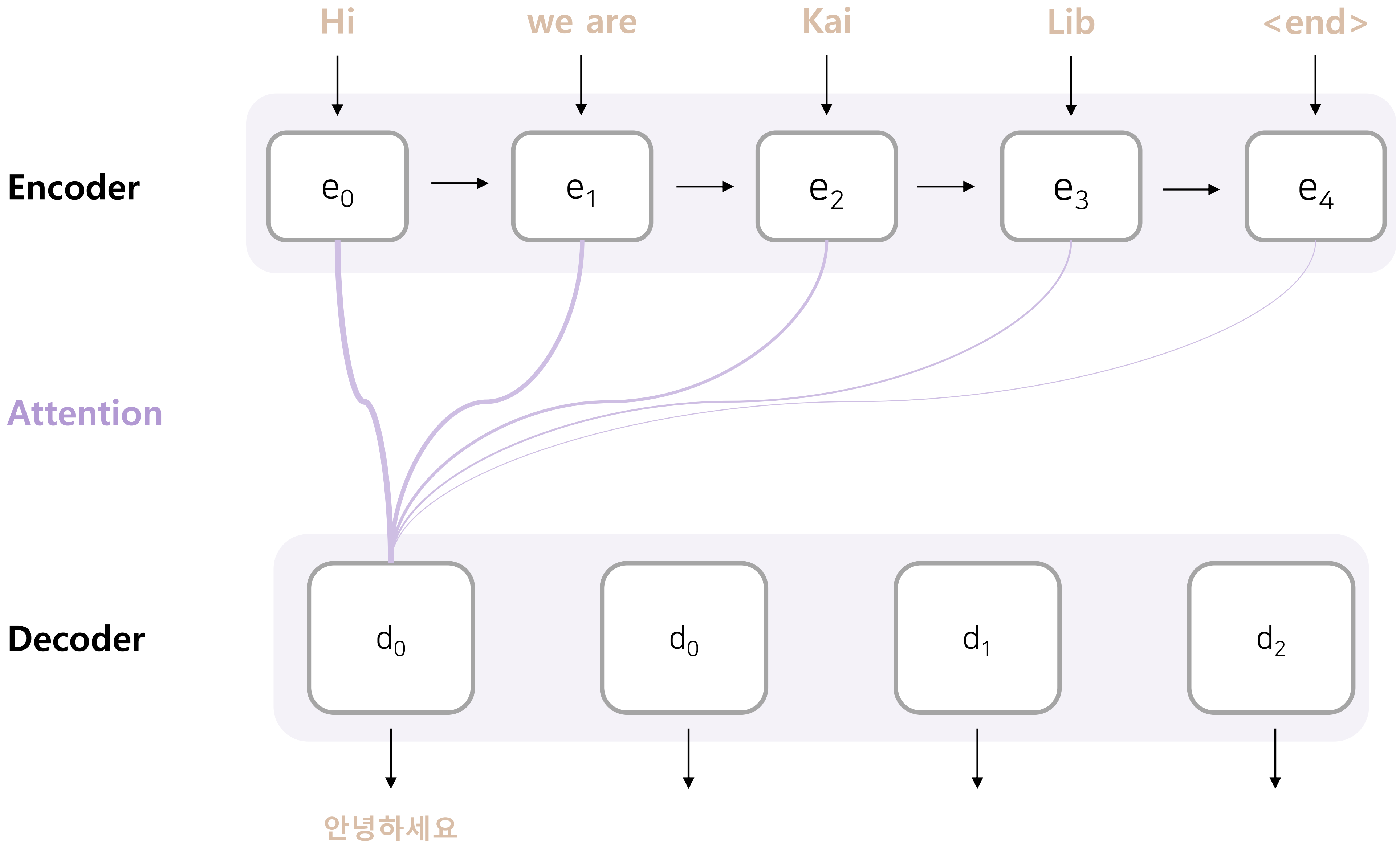
Seq2seq + Attention Step ④



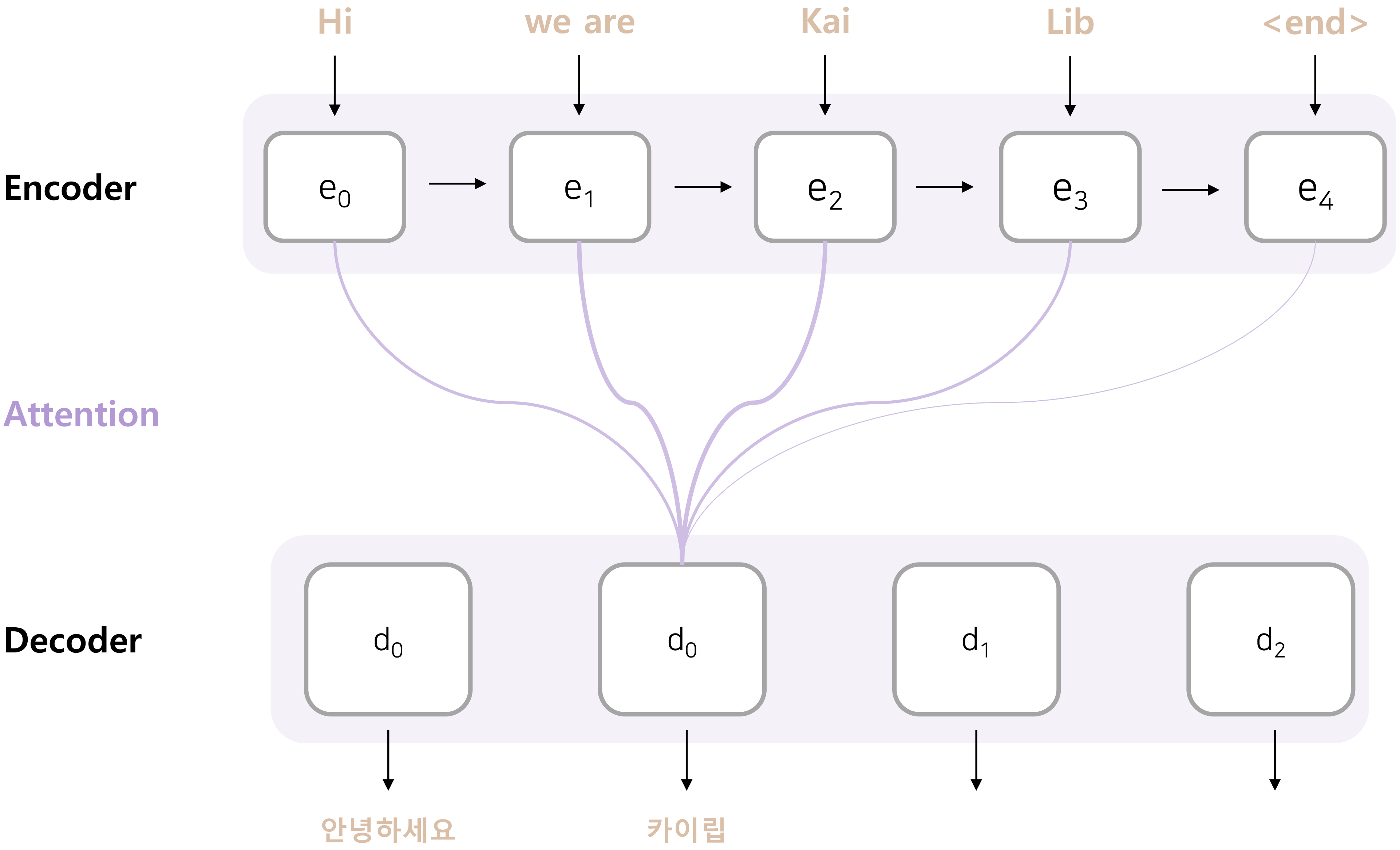
Seq2seq + Attention Step ⑤



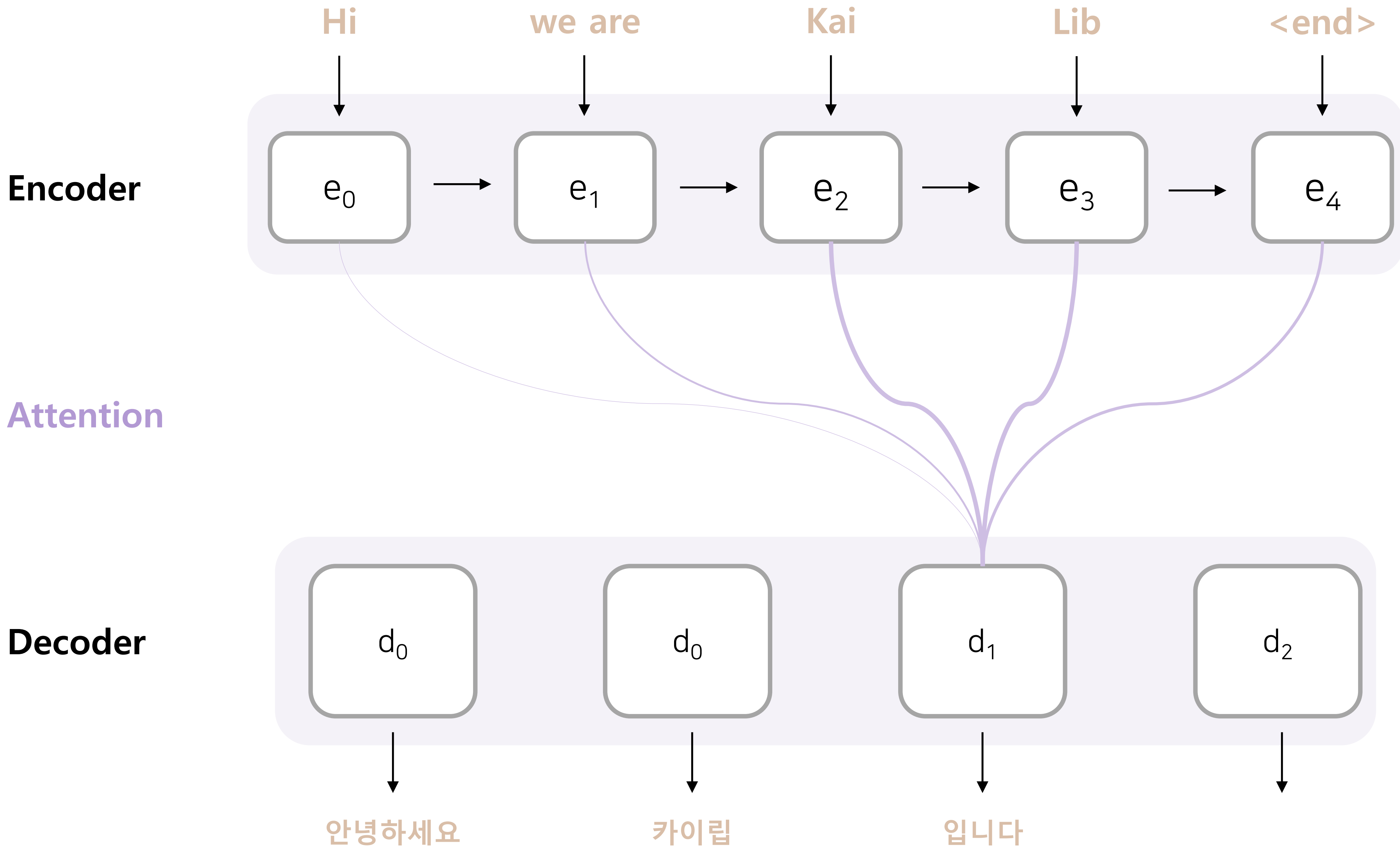
Seq2seq + Attention Step ⑥



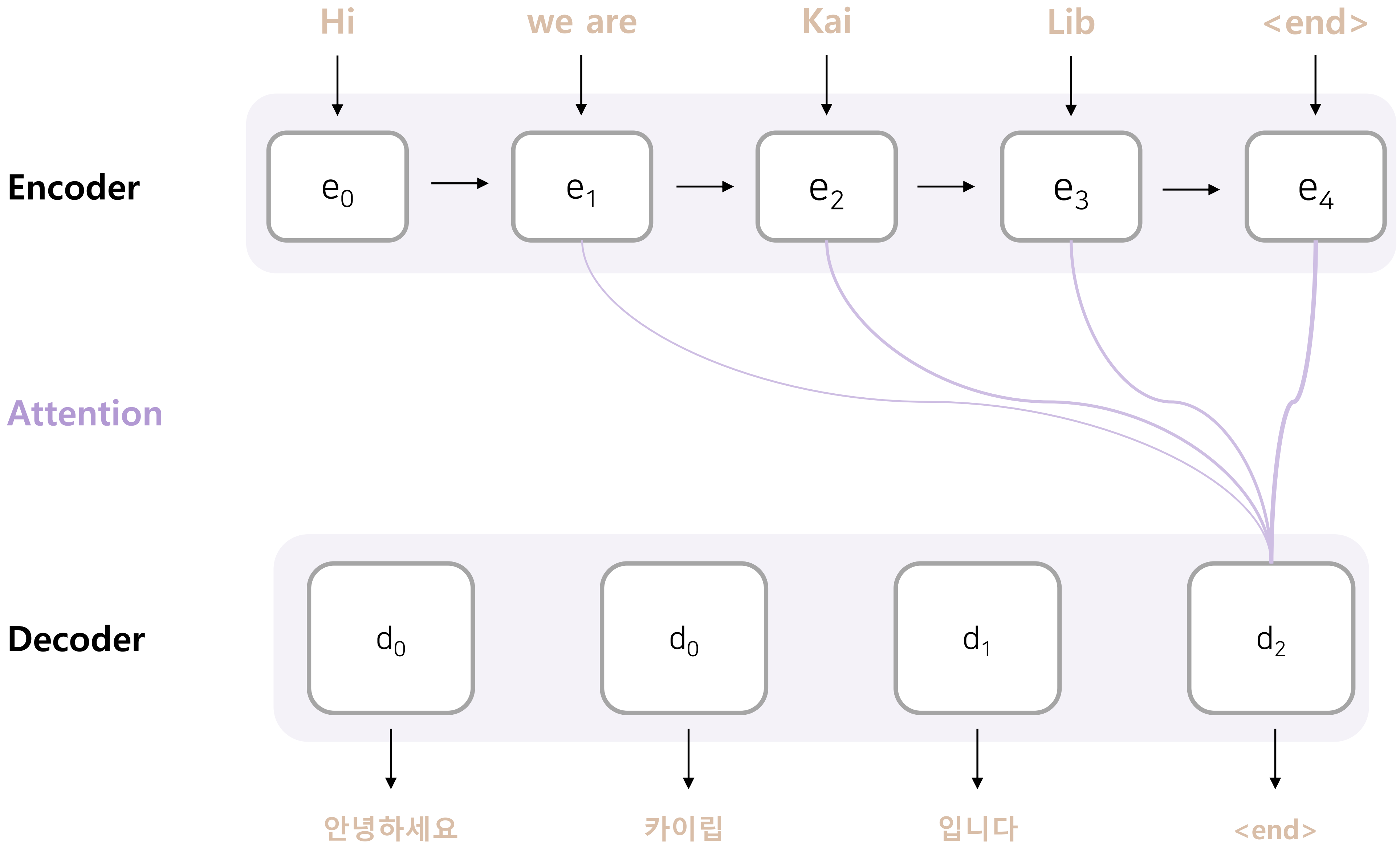
Seq2seq + Attention Step ⑦



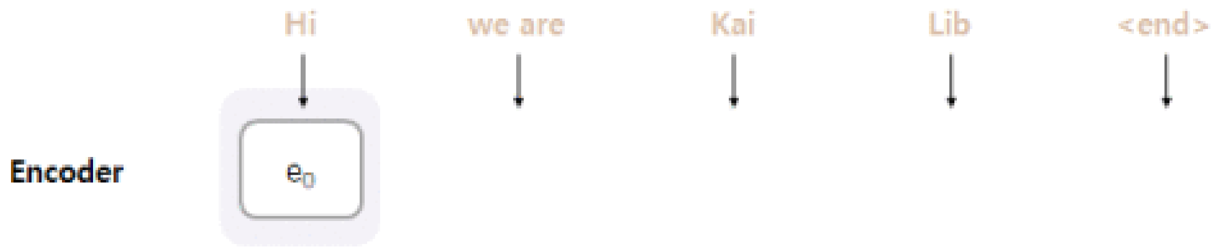
Seq2seq + Attention Step ⑧

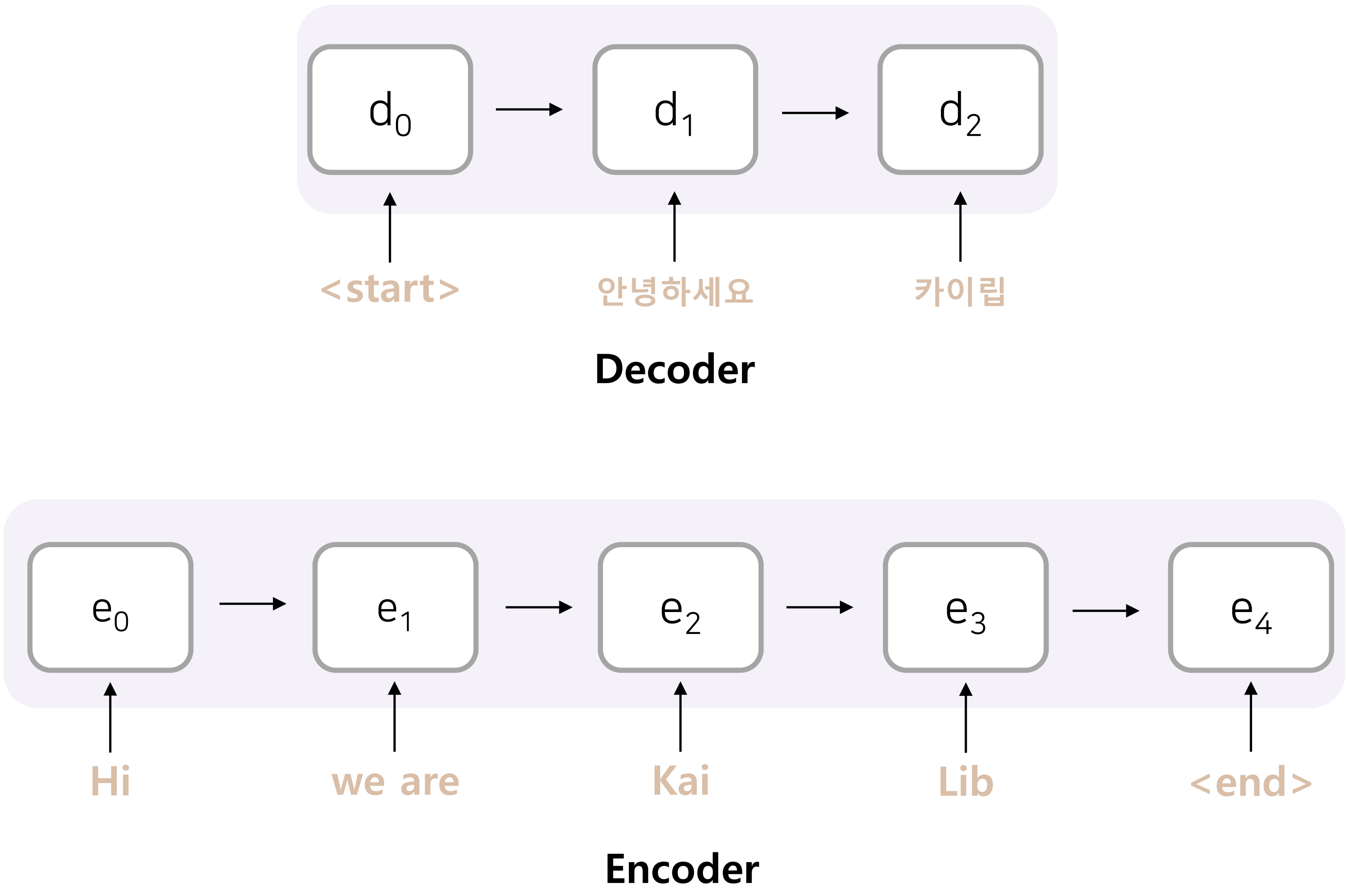


Seq2seq + Attention Step ⑨



Seq2seq + Attention Mechanism




d_0
d_1
d_2
<start>
안녕하세요
카이립
Decoder
e_0
e_1
e_2
e_3
e_4
Hi
we are
Kai
Lib
<end>
Encoder