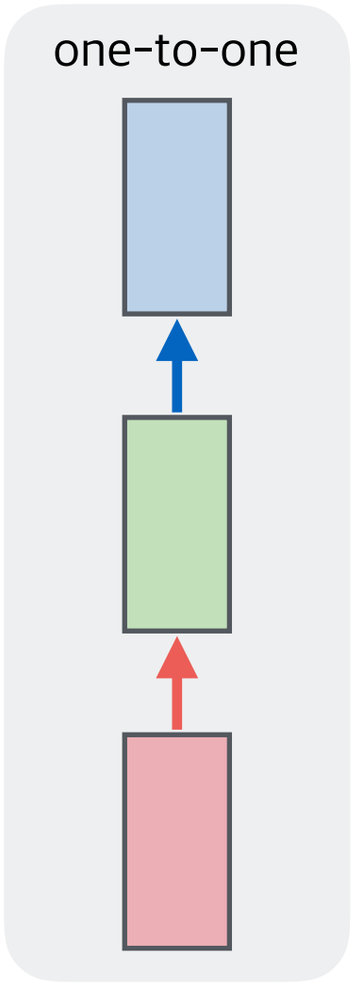
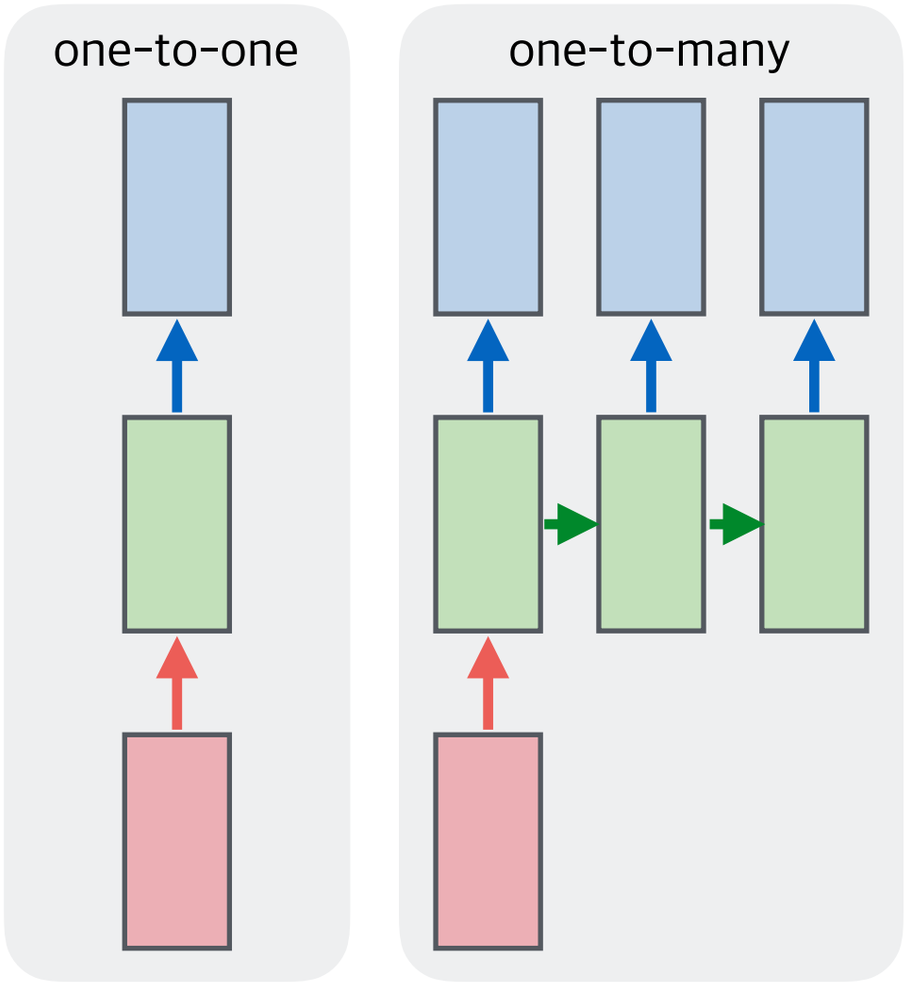
RNN

안남혁

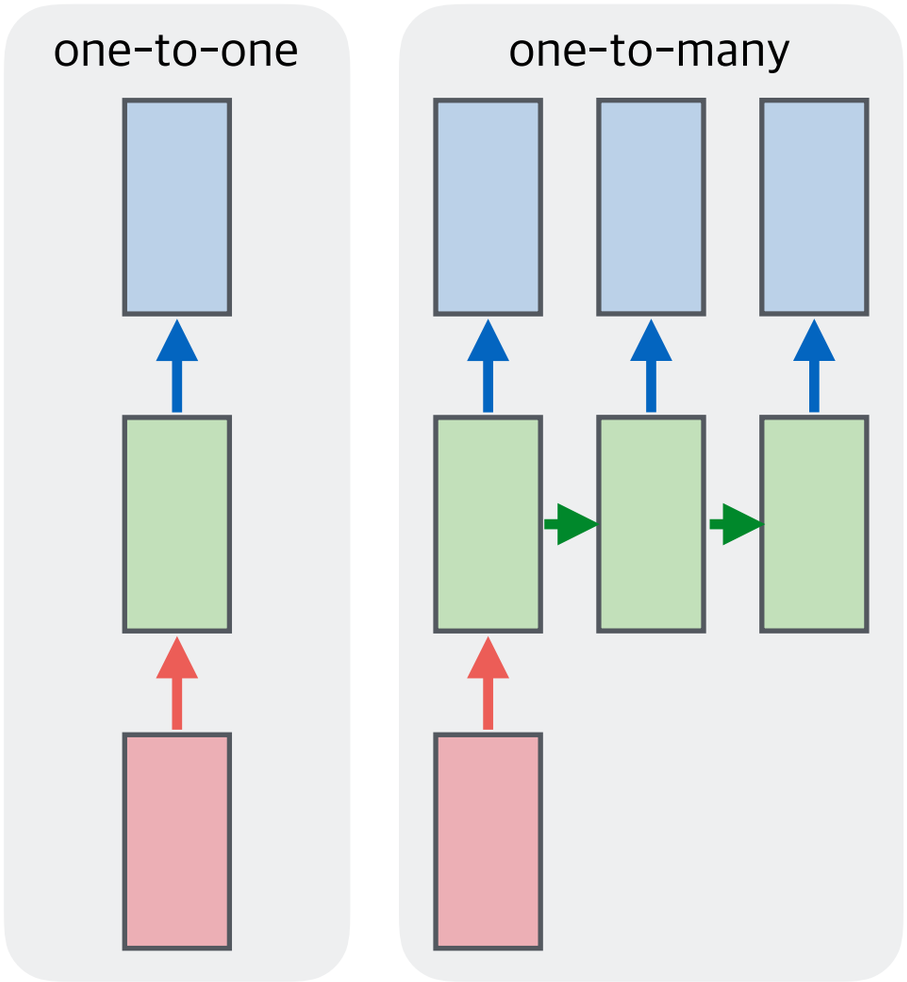
Neural Network



Recurrent Neural Network



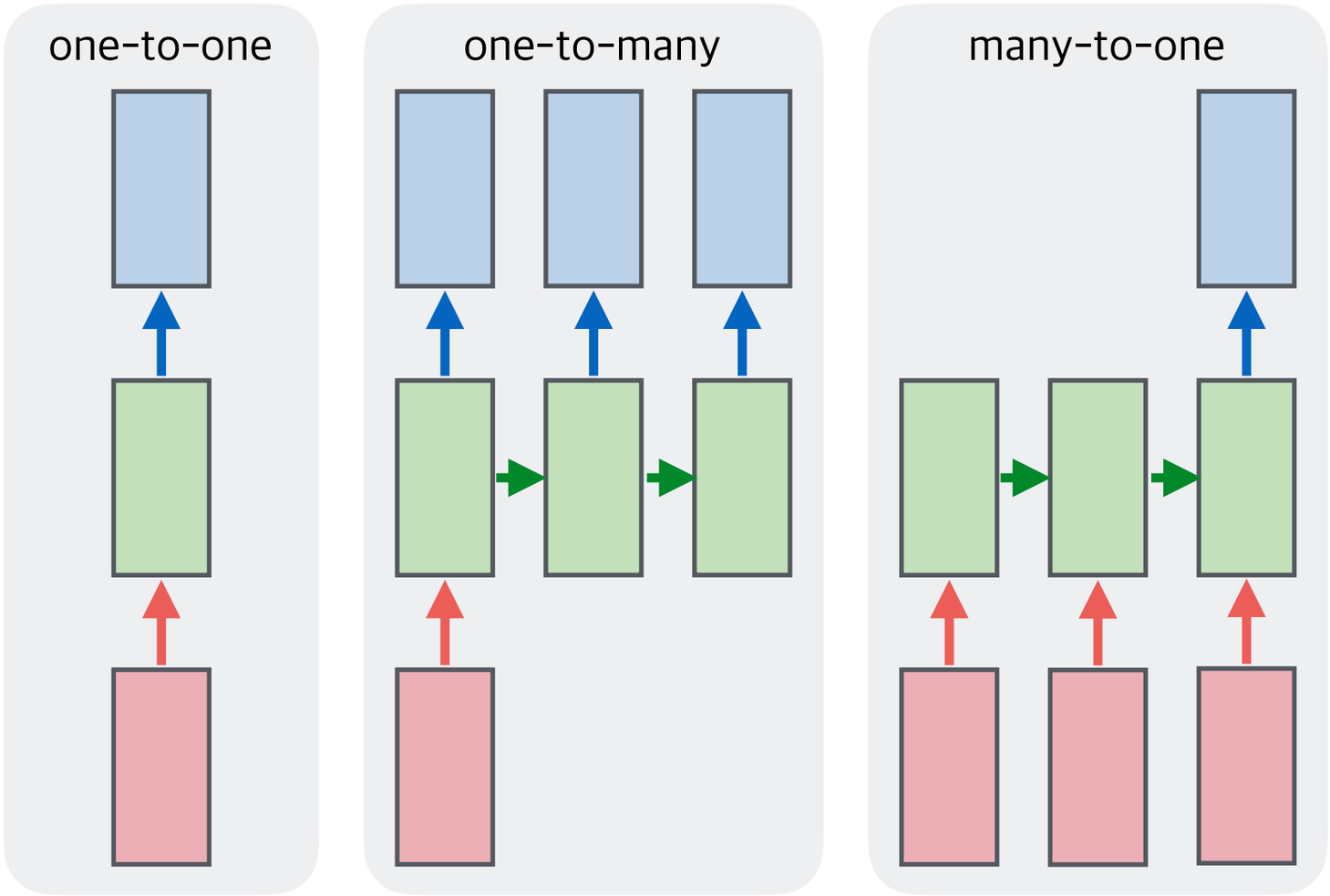
Recurrent Neural Network



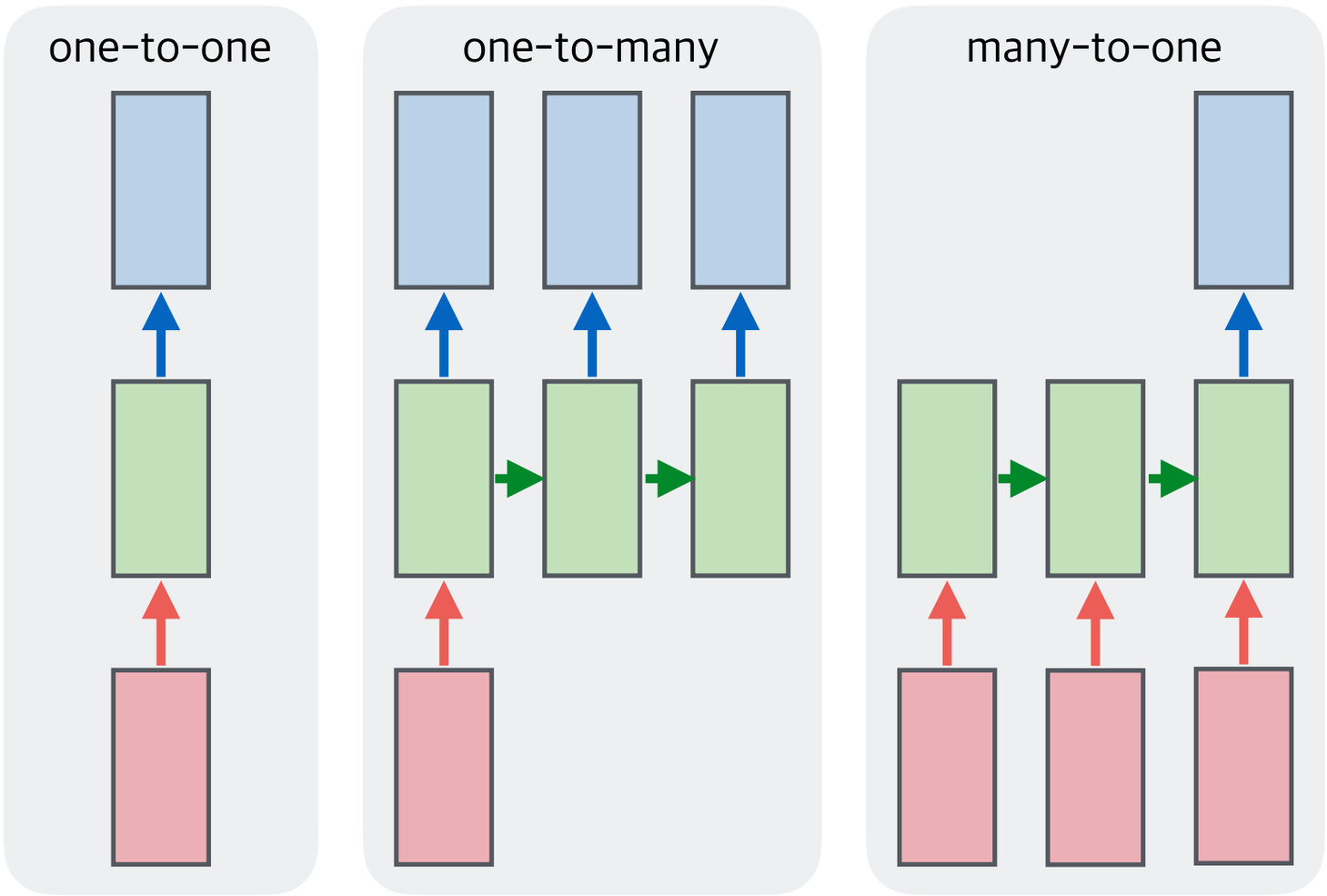
예) 이미지 캡션 생성

- 이미지 (one) -> 단어 시퀀스 (many)

Recurrent Neural Network



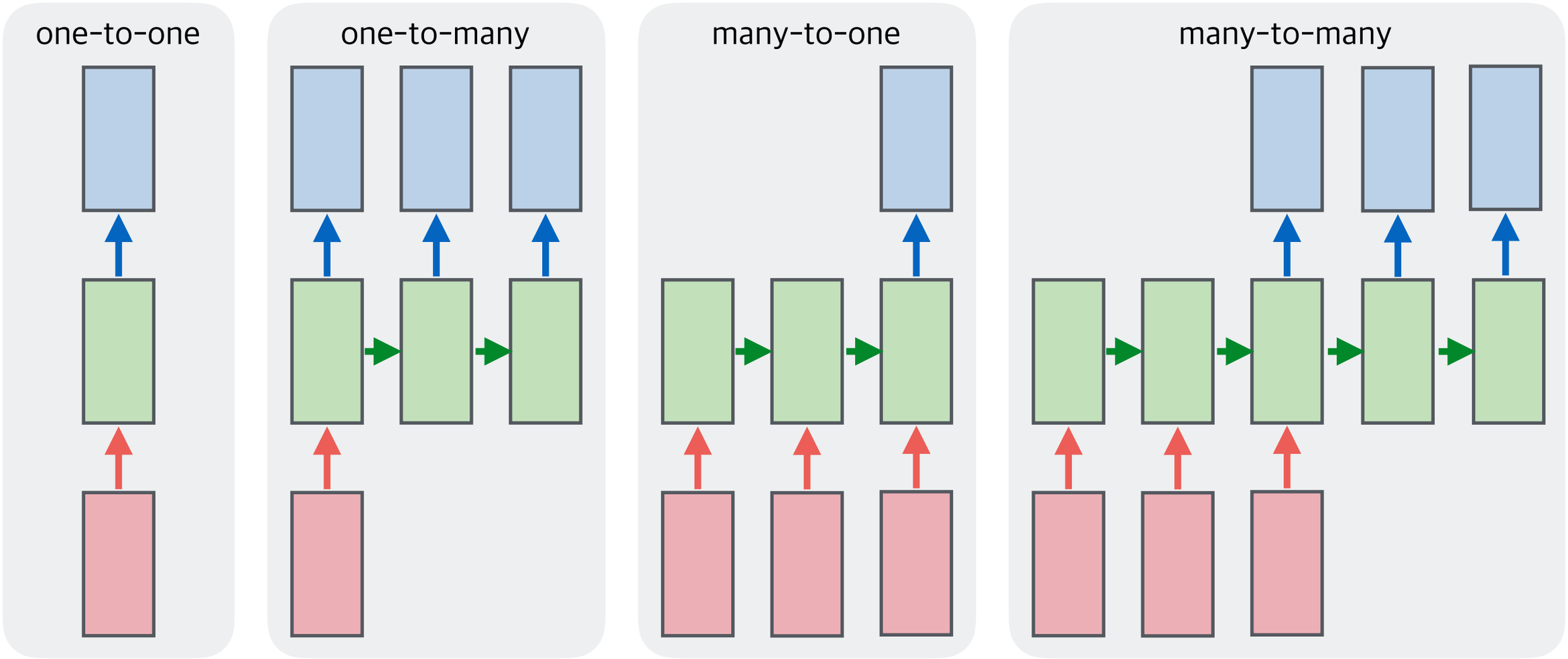
Recurrent Neural Network



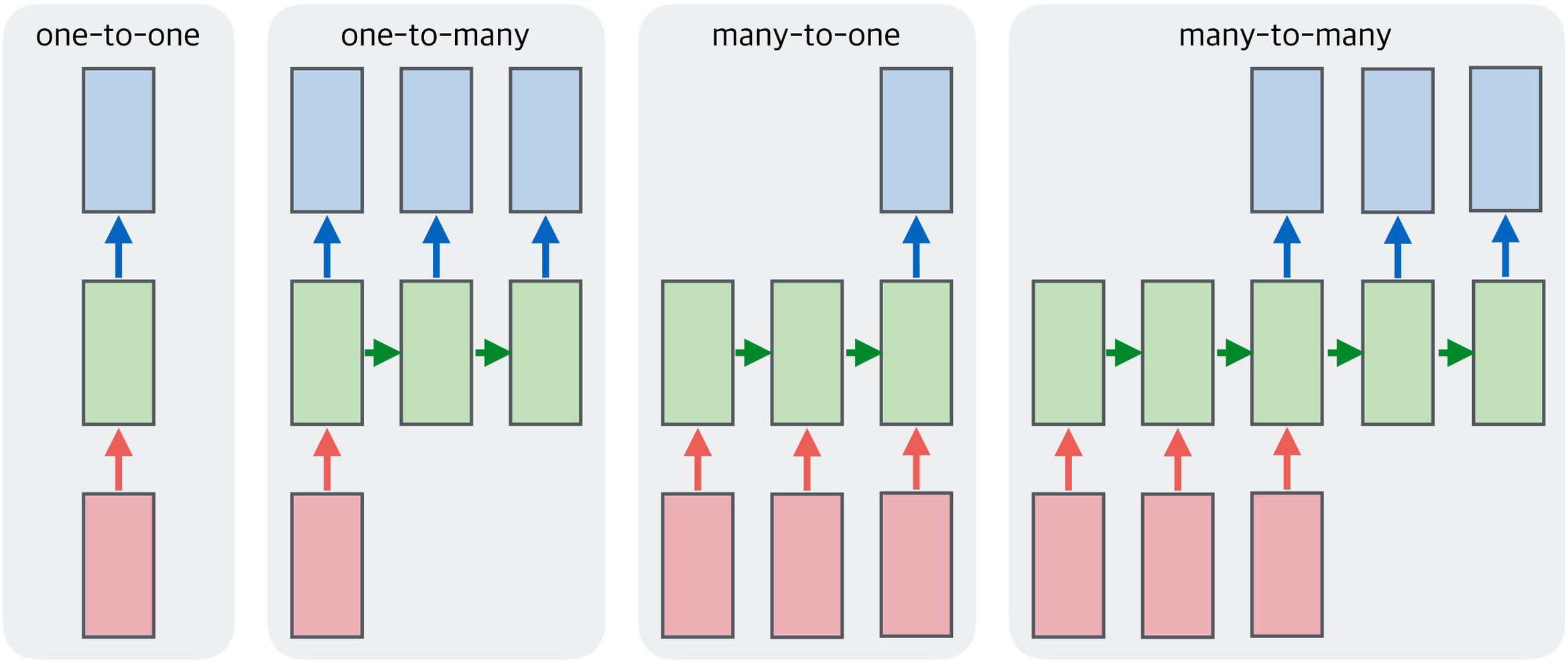
예) 텍스트 감정 분류

- 단어 시퀀스 (many) -> 감정 (word)

Recurrent Neural Network



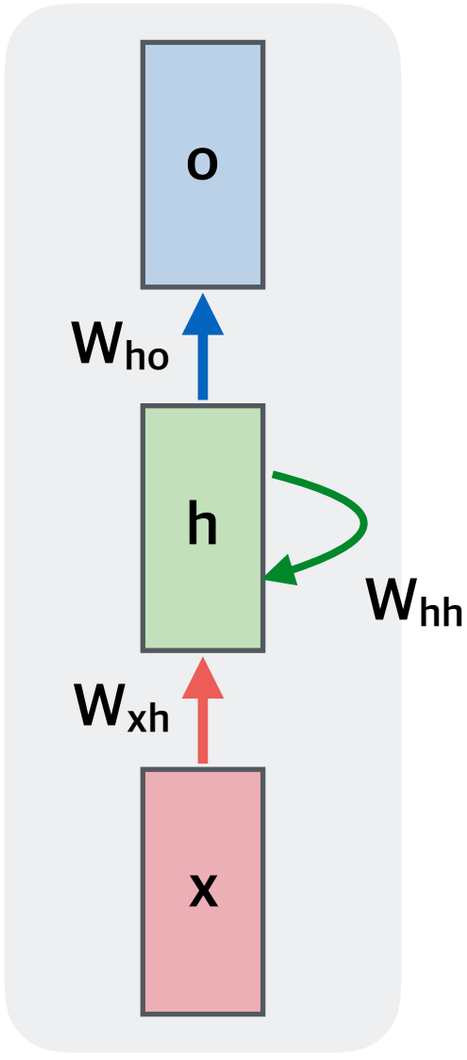
Recurrent Neural Network



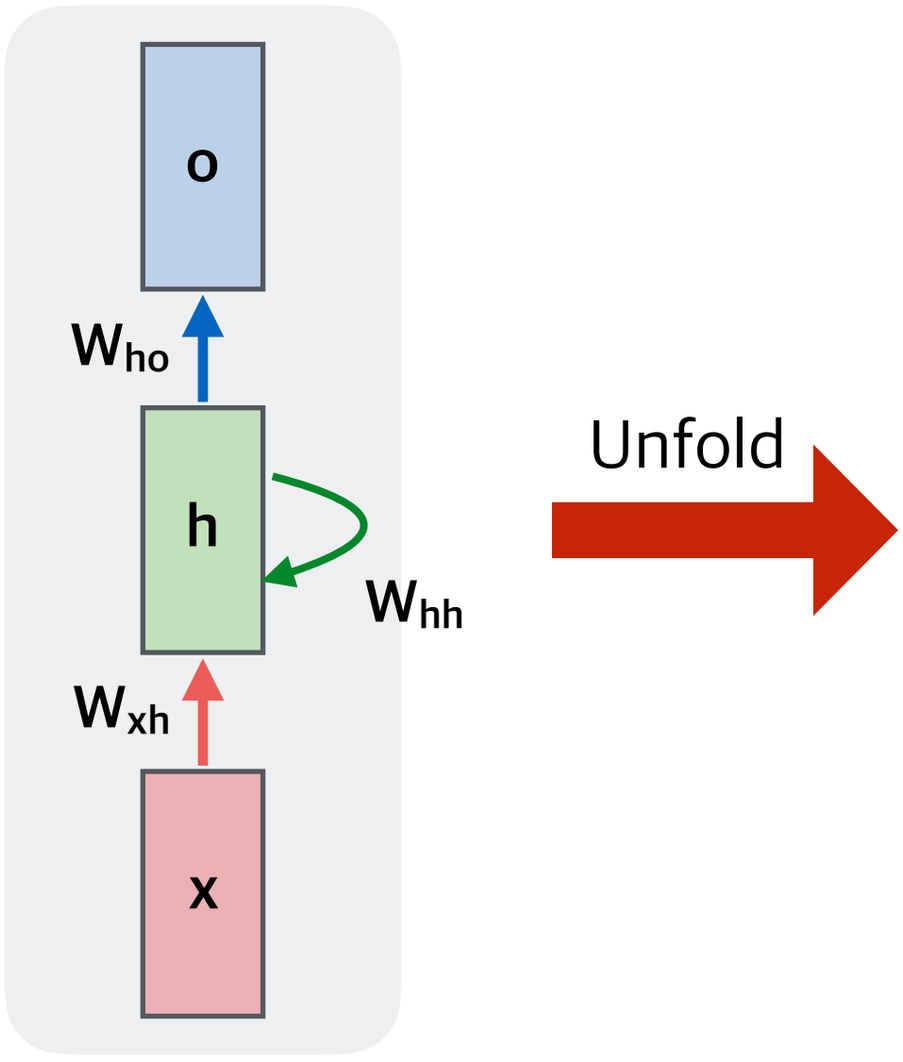
예) 기계 번역

- 영어 단어 시퀀스 (many) ->
  한국어 단어 시퀀스 (many)

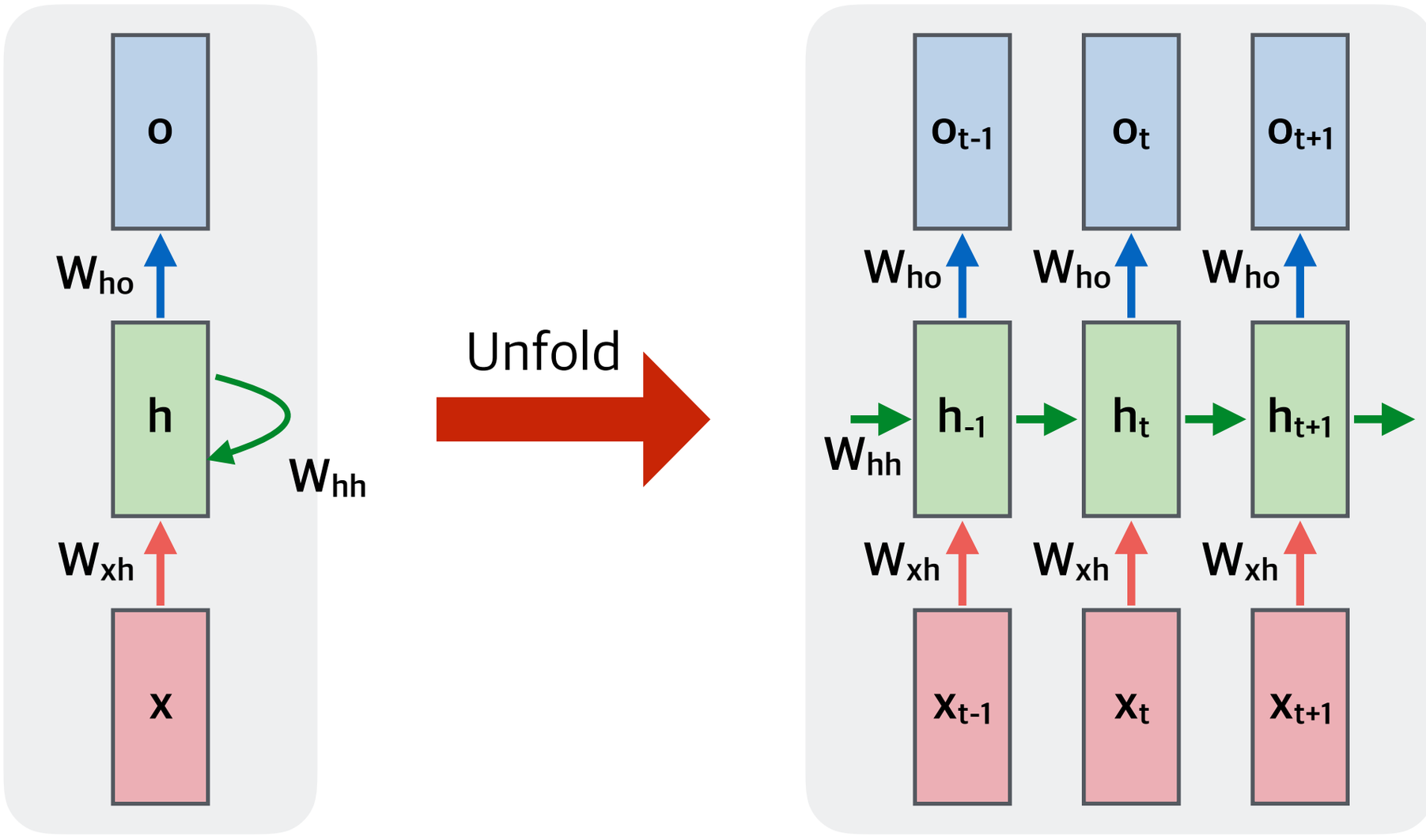
Recurrent Neural Network



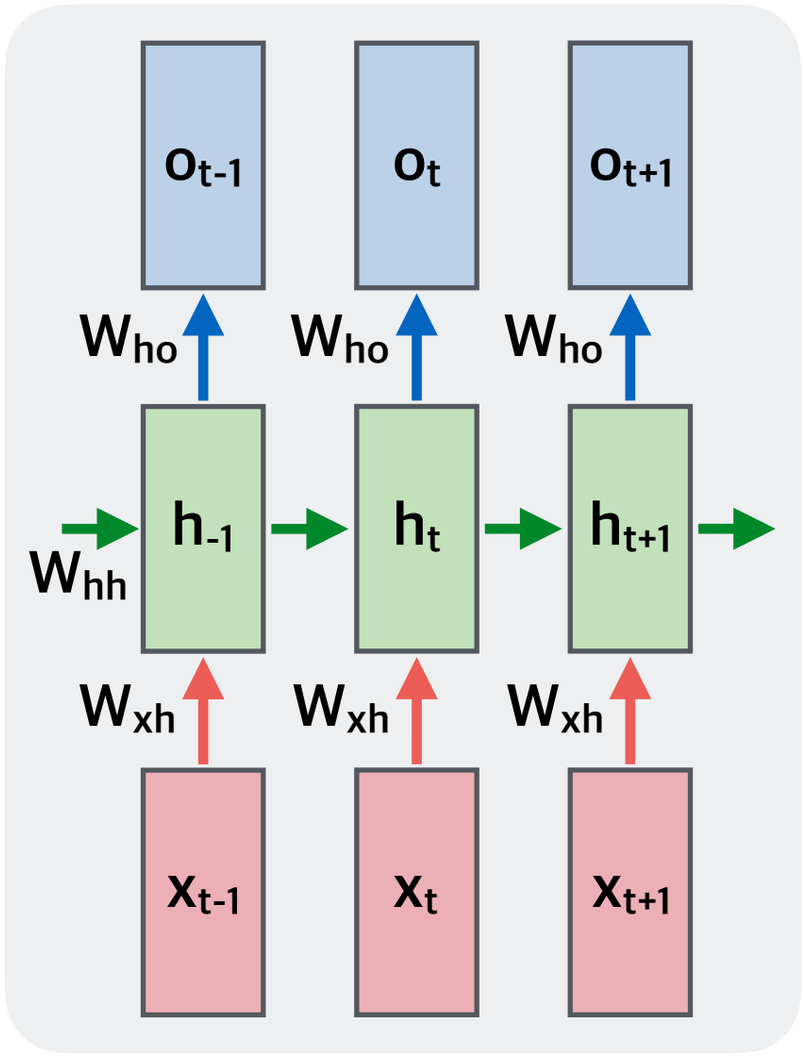
Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network

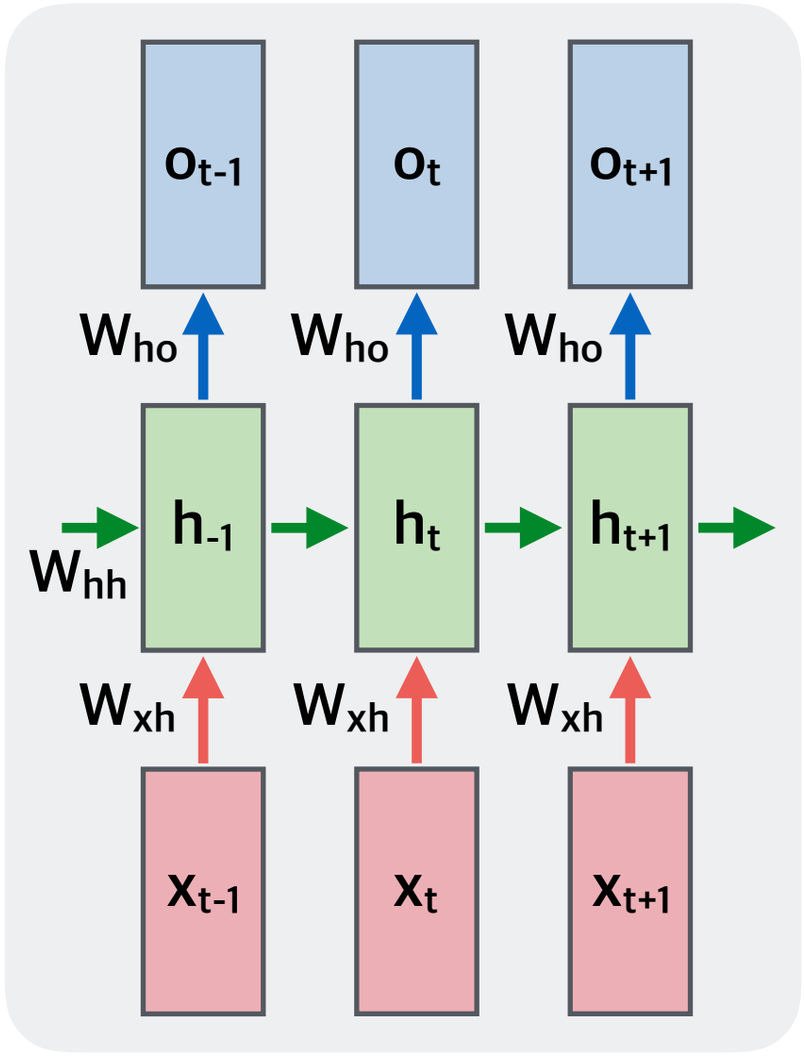


- x_t: 시간 스텝 t 에서의 입력값
- W_{hh}, W_{xh}, W_{ho}: **모든 스텝(시간)에서 공유됨**
- h_t: 시간 스텝 t 에서의 중간 값 (**hidden state**)

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

- 현재 상태 h_t는 이전 스텝의 상태 h_{t-1}와 현재 스텝의 입력값 x_t에 의해 계산됨
- o_t: 시간 스텝 t 에서의 출력값 (score)

Recurrent Neural Network



RNN의 특징

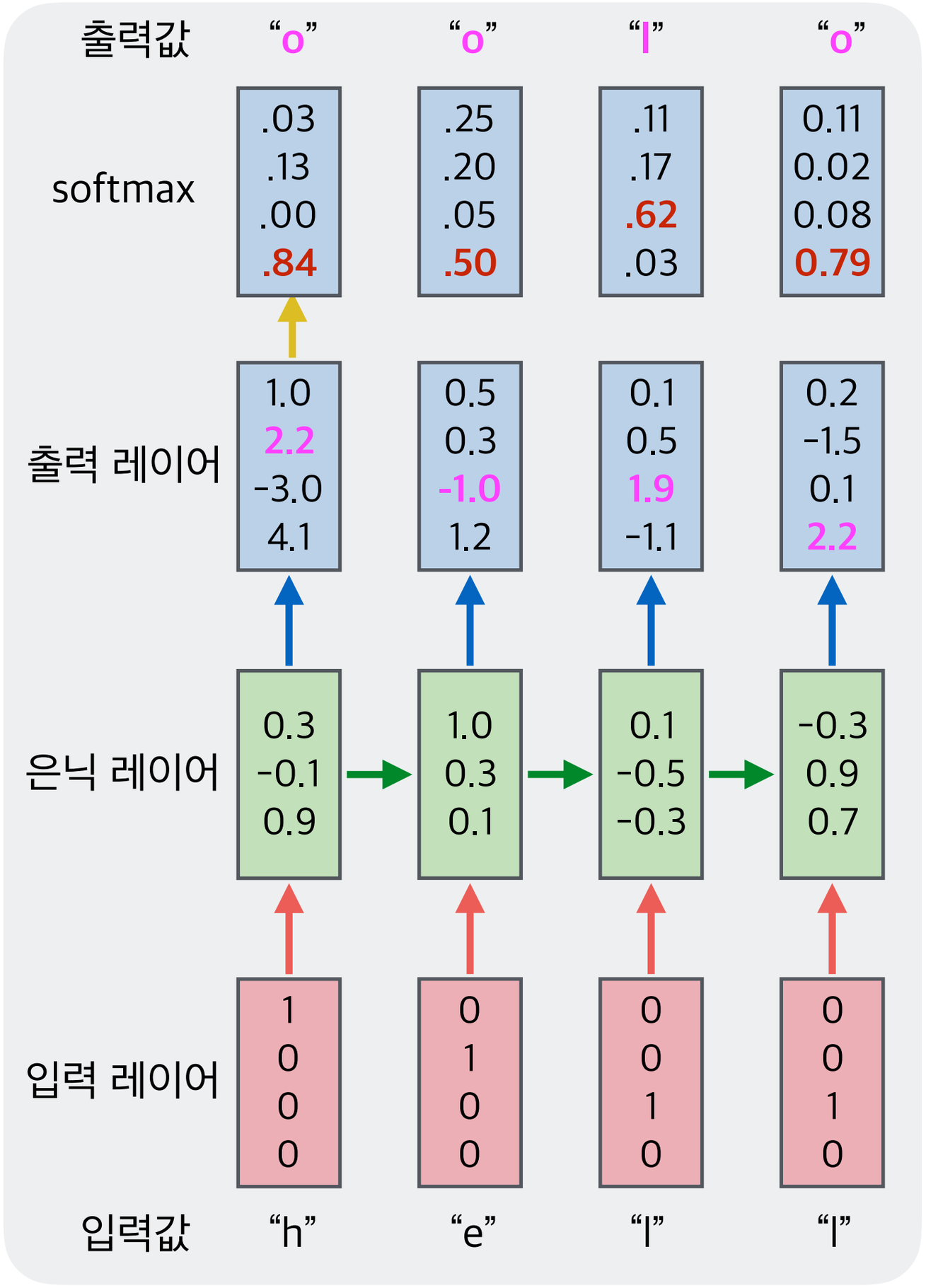
- 상태 $\mathbf{h}_t$를 네트워크의 메모리라고 생각할 수 있음 (이전 시간 스텝에 영향을 받으며, 이에 대한 정보를 담고 있음)
- 출력값 $\mathbf{o}_t$는 현재 상태 $\mathbf{h}_t$에만 의존하고 있음
- 일반적인 뉴럴 네트워크와 달리 파라미터는 **모든 스텝에서 공유**되고 있음
- 학습해야 할 파라미터를 대폭 낮춰 줌 (공유 하지 않으면 시퀀스 길이가 긴 경우 엄청난 파라미터 개수..)

char-RNN

Vocabulary:
[h, e, l o]

학습 시퀀스:
“hello”

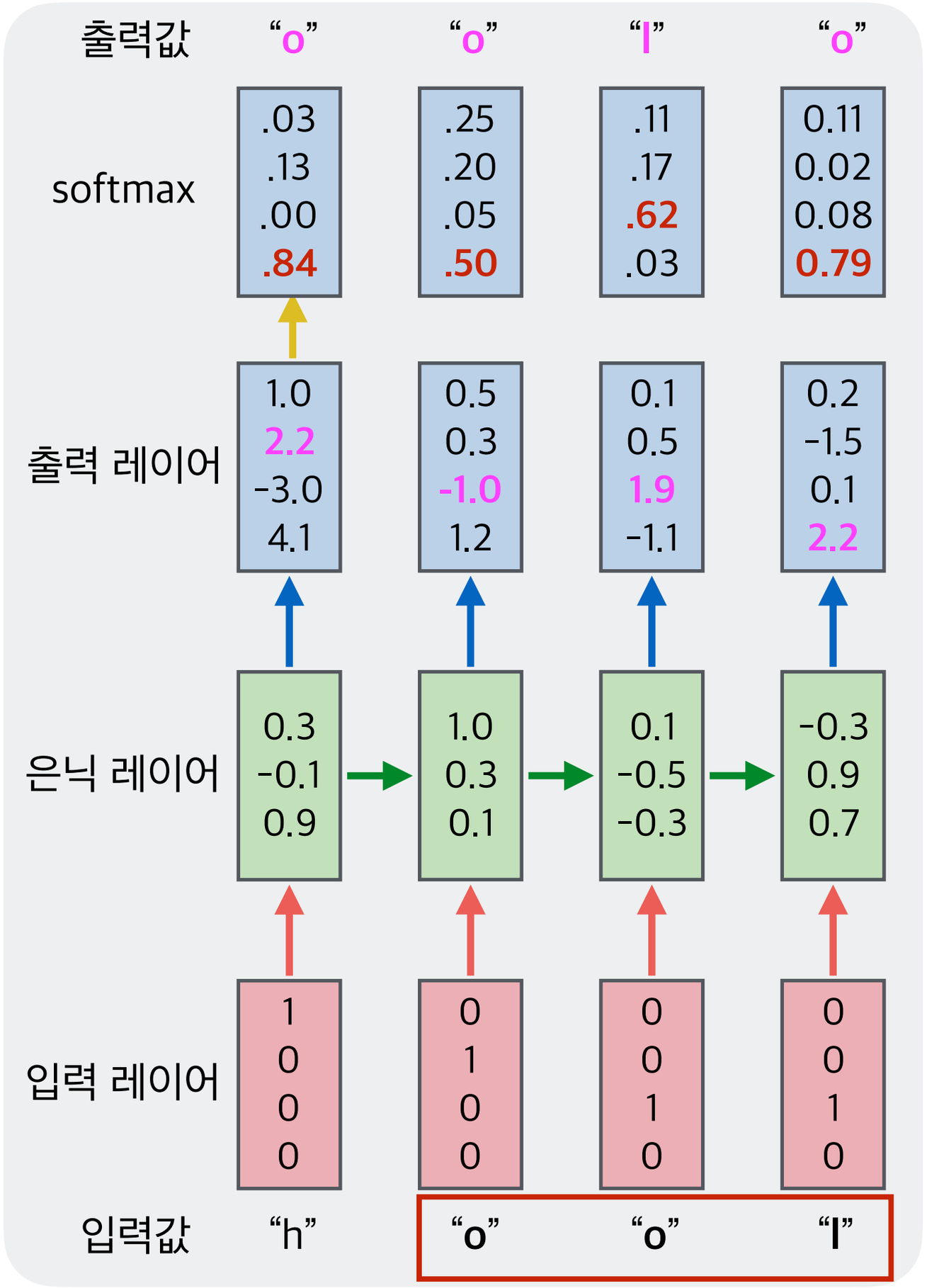
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$



char-RNN

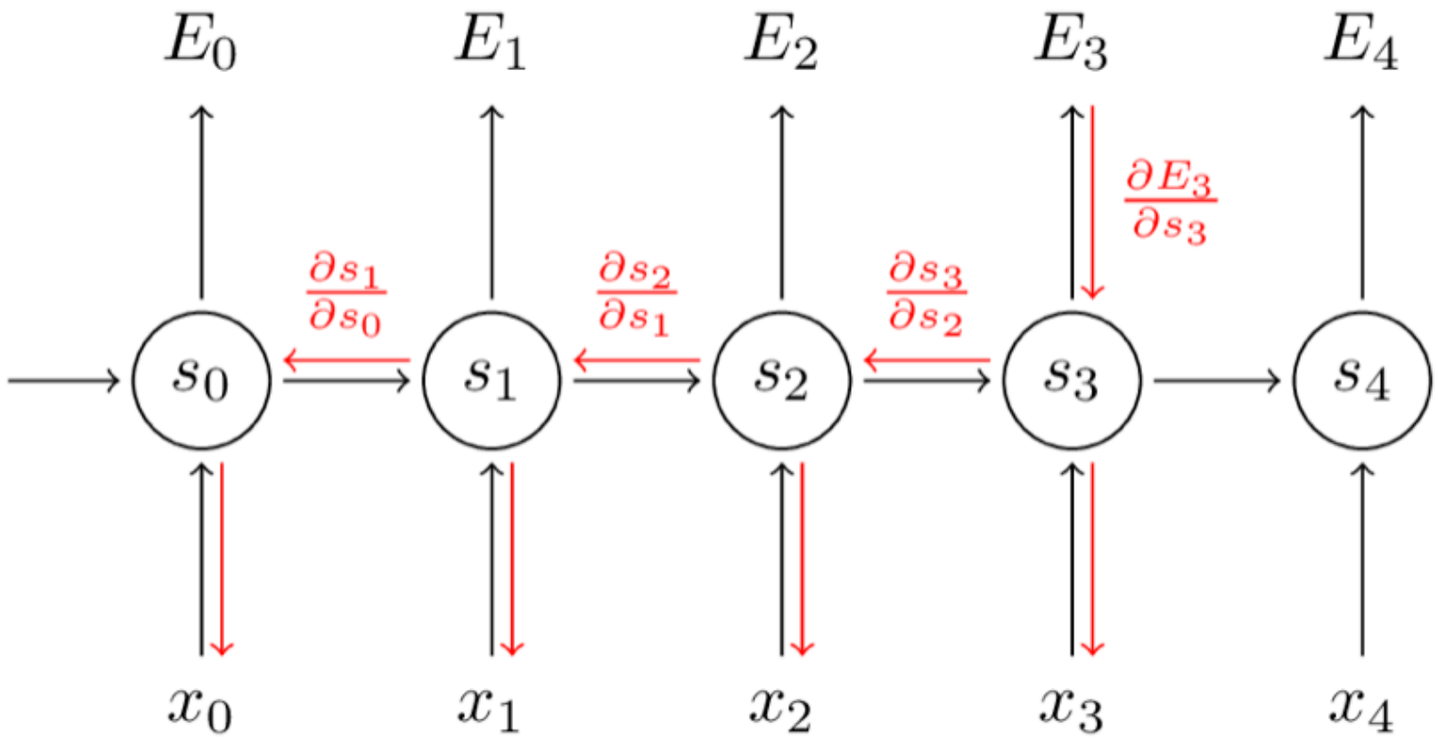
Vocabulary:
[h, e, l o]

테스트 단계:
t 스텝의 출력 값을 t+1 스텝의 입력값으로
(시퀀스의 정답을 모르기 때문)



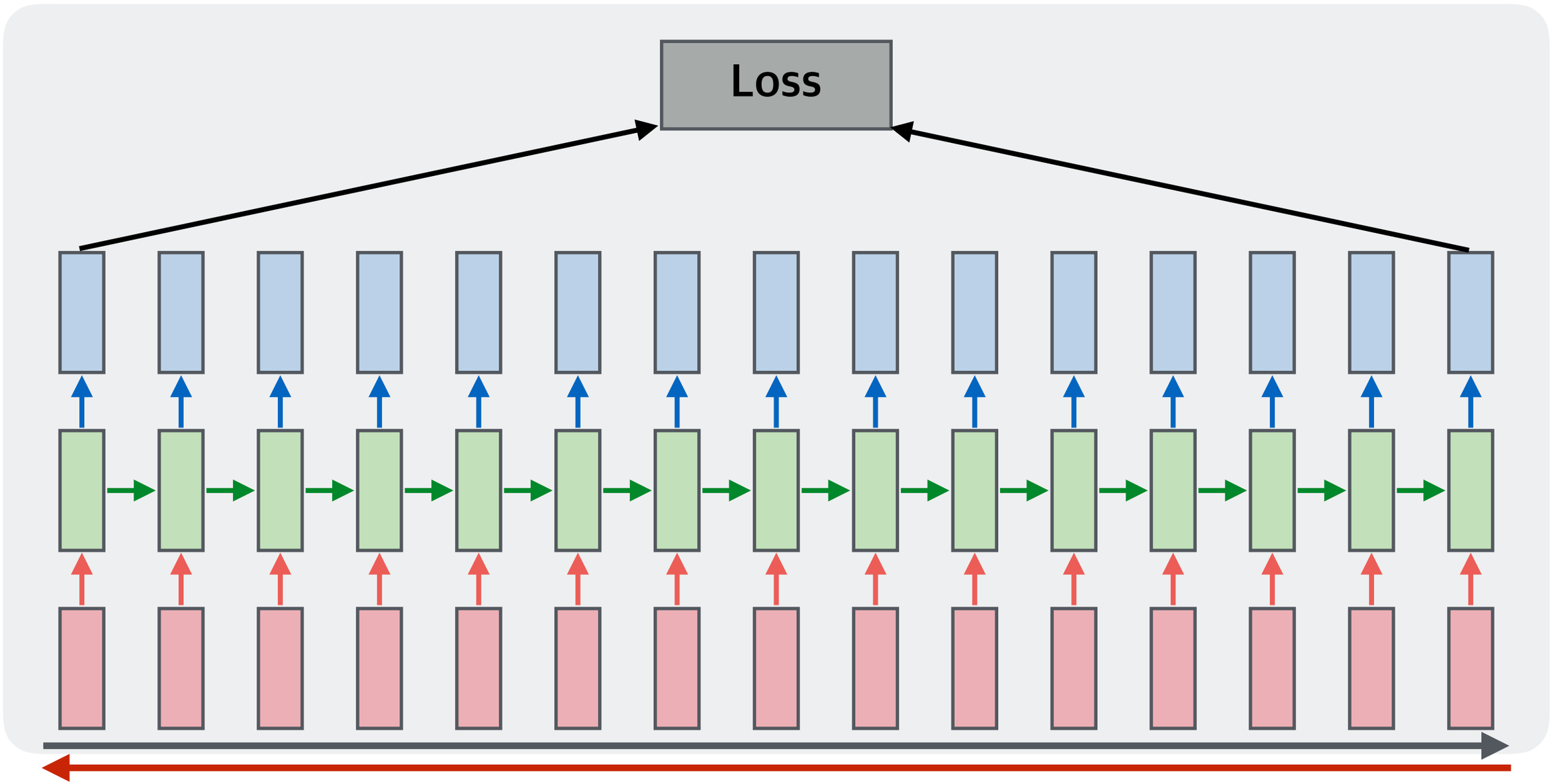
Backpropagation through time

- RNN은 문장과 같은 시퀀스를 입력으로 받기 때문에 학습 시 backpropagation을 **시간에 대해서도 수행**
  - 매 스텝마다 loss를 계산하며 (E_0 ~ E_4), 총 loss는 각 스텝 loss의 합
  - 일반적으로 loss는 이전에 배웠던 크로스 엔트로피 loss를 사용



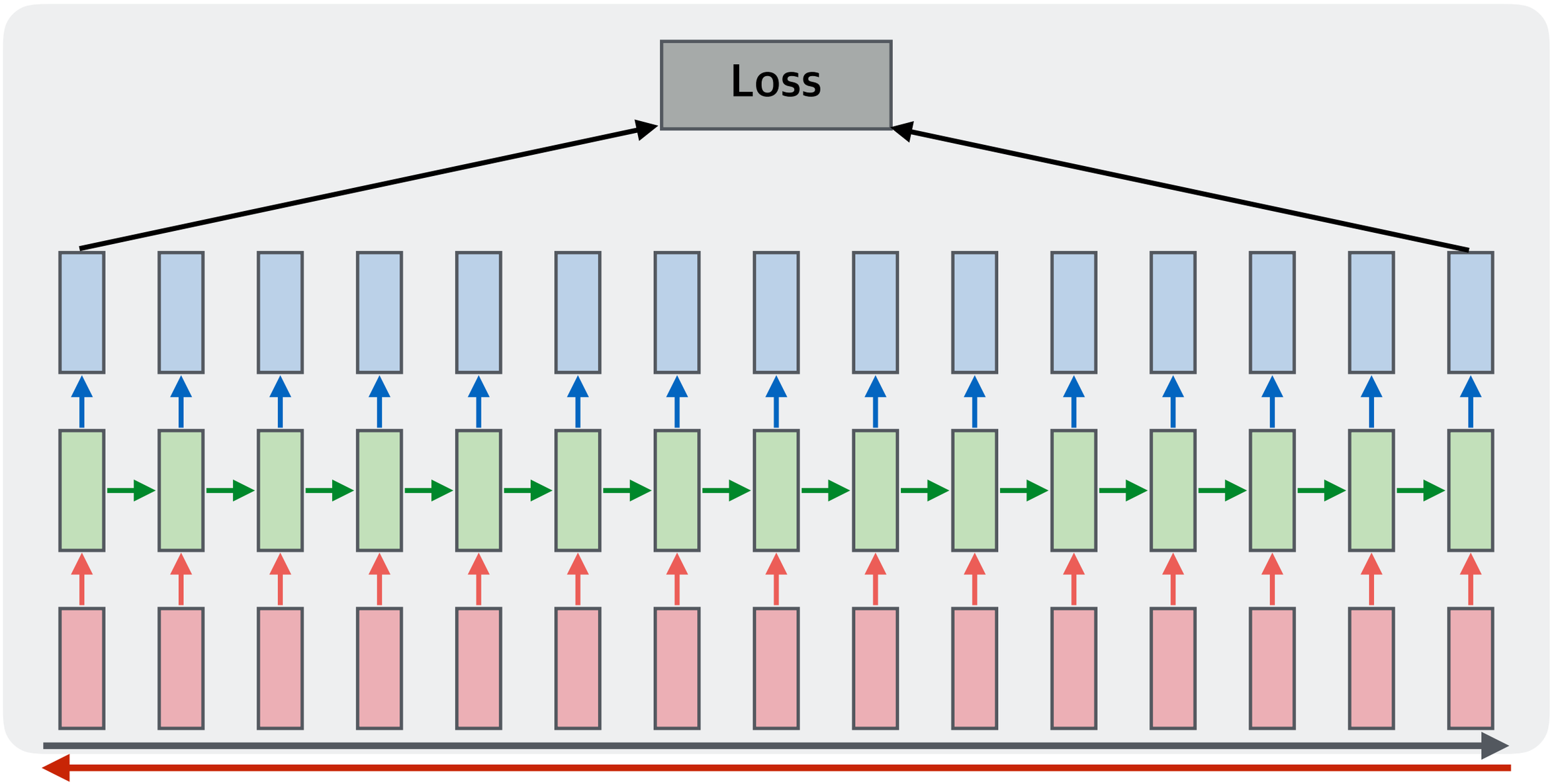
http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

BPTT

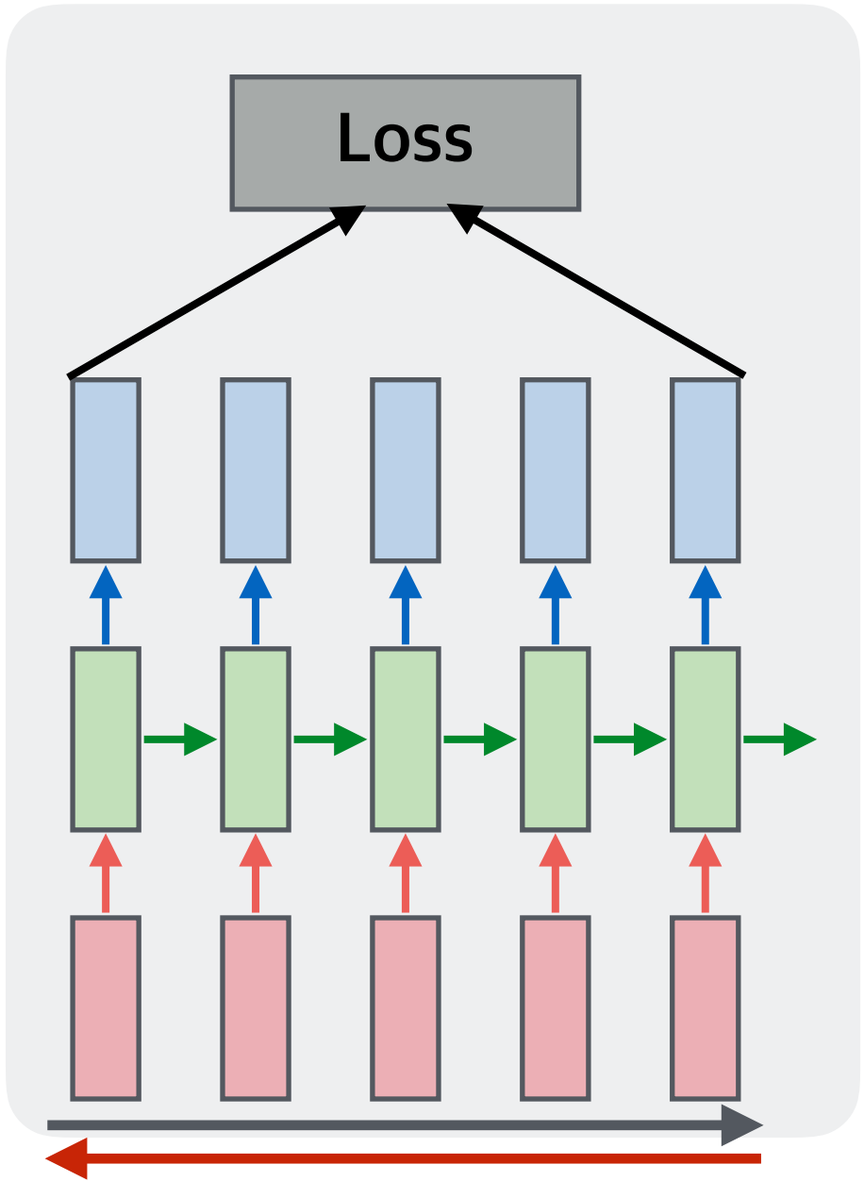


BPTT

- **모든 시퀀스**에 대해 forwardprop을 진행하고 loss를 계산 한 뒤, backprop시 **모든 시퀀스**에 대해 그라디언트를 계산
- 시퀀스가 길면? -> 매우 느린 계산 속도

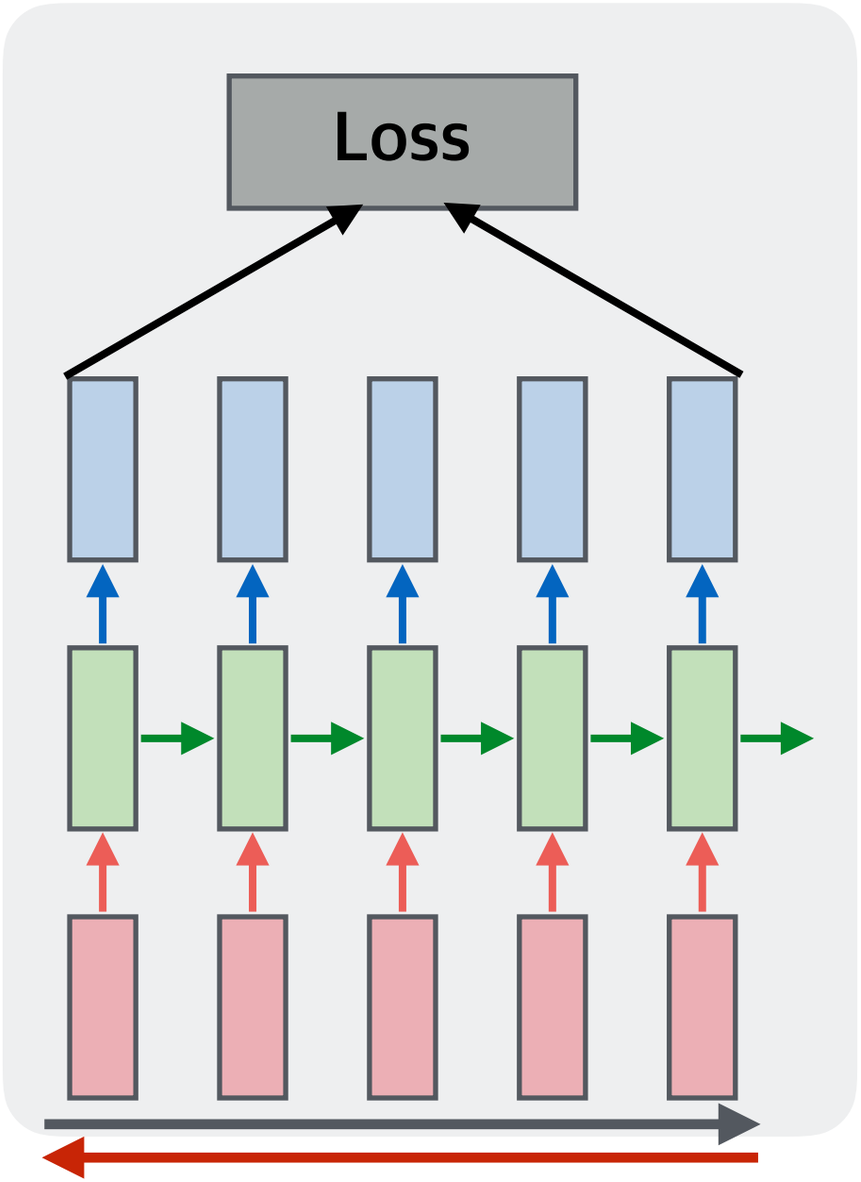


Truncated BPTT



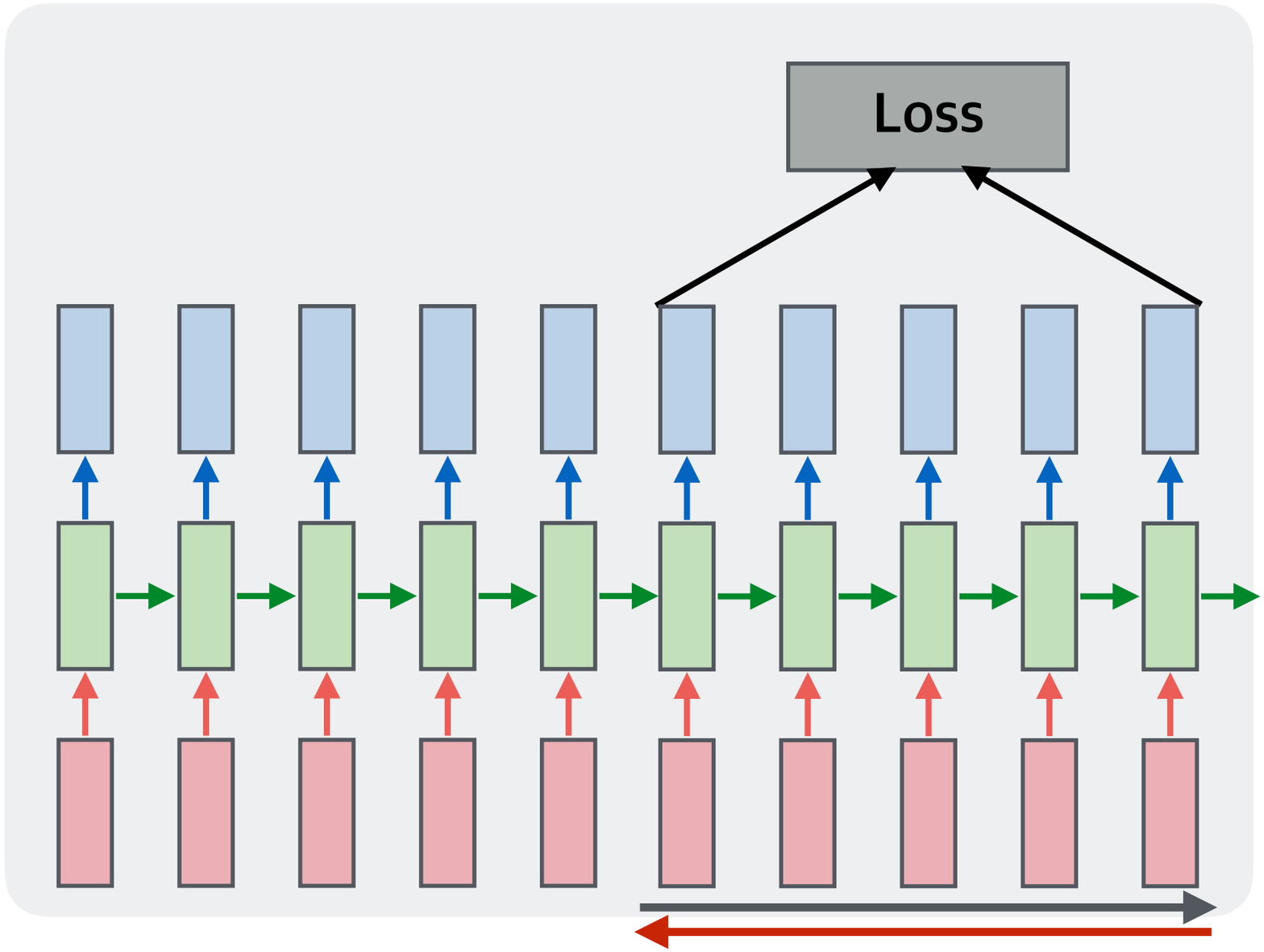
Truncated BPTT

- Forward와 backward를 전체 시퀀스에 대해 수행하지 않고 **부분 (chunk) 시퀀스**에 대해서만 수행



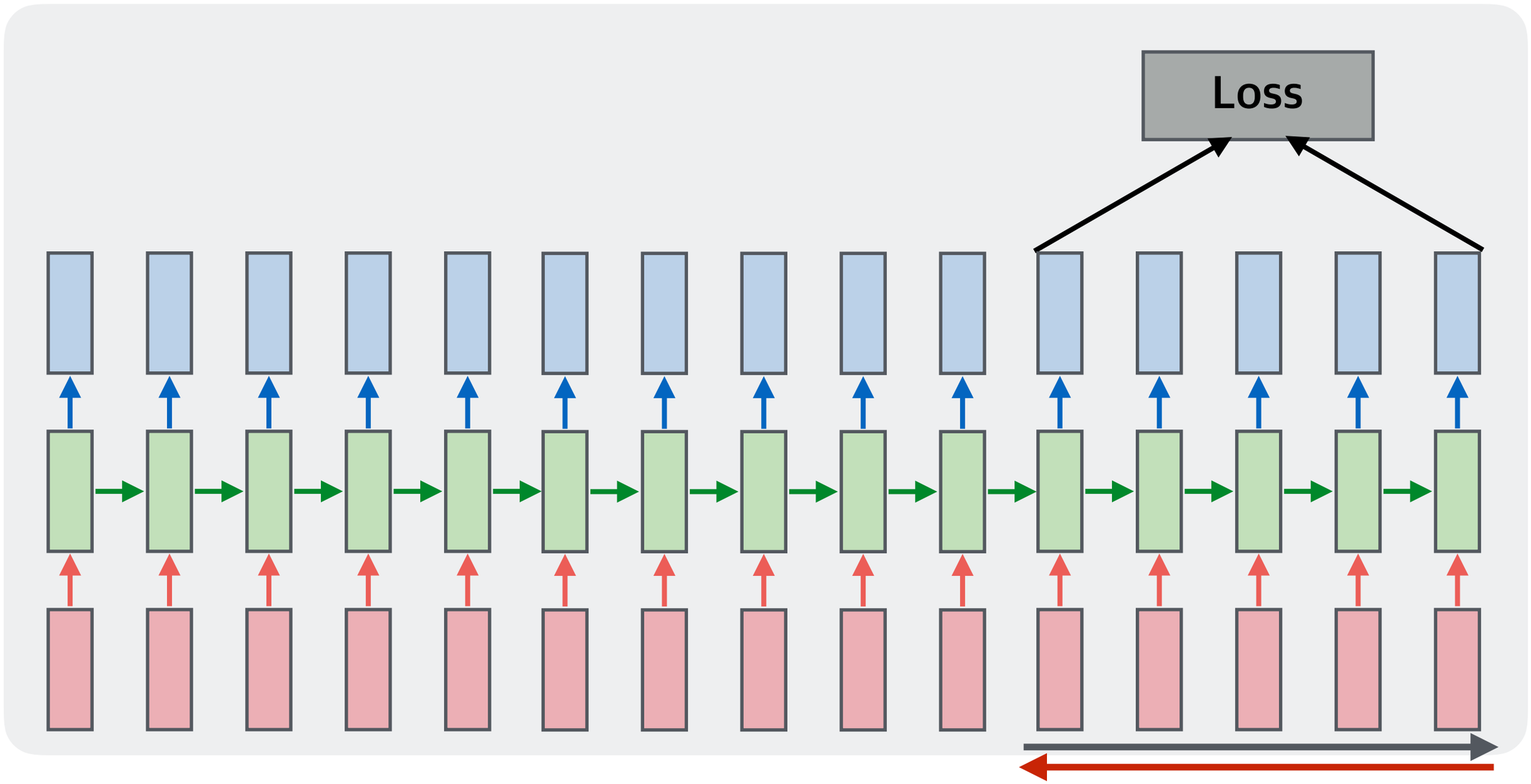
Truncated BPTT

- Forward와 backward를 전체 시퀀스에 대해 수행하지 않고 **부분 (chunk) 시퀀스**에 대해서만 수행



Truncated BPTT

- Forward와 backward를 전체 시퀀스에 대해 수행하지 않고 **부분 (chunk) 시퀀스**에 대해서만 수행



RNN 응용 사례

- 셰익스피어 따라하기

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

https://karpathy.github.io/2015/05/21/rnn-effectiveness/

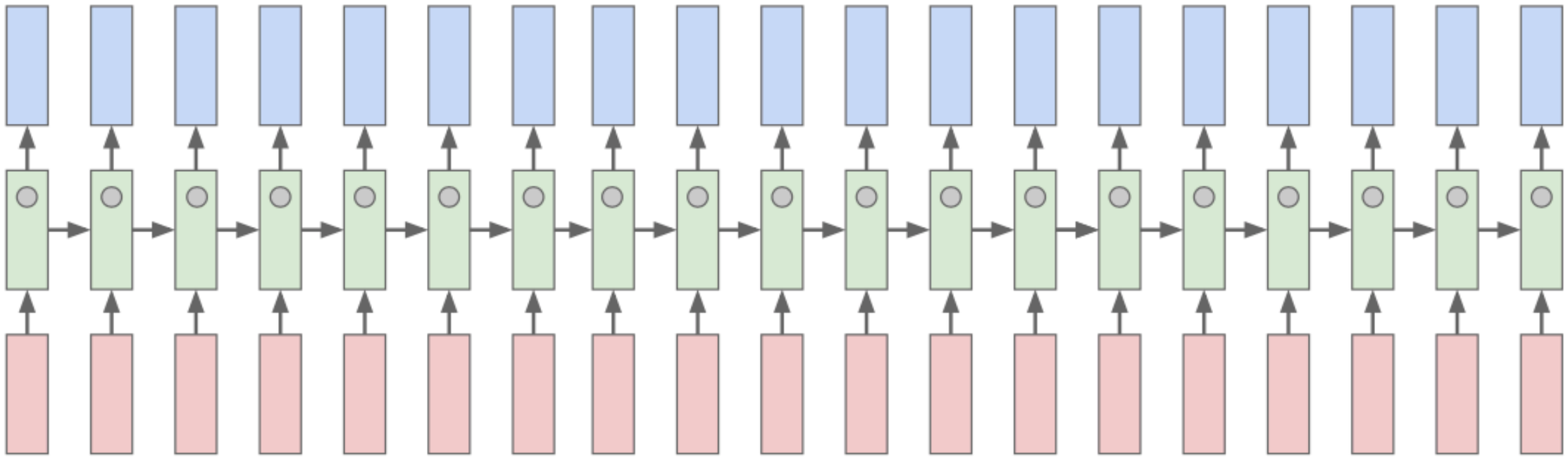
RNN 응용 사례

- 리눅스 커널 코드 학습
  - 실제로 돌아가는 코드는 아니지만 그럴싸 하게 코드를 따라할 수 있음

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

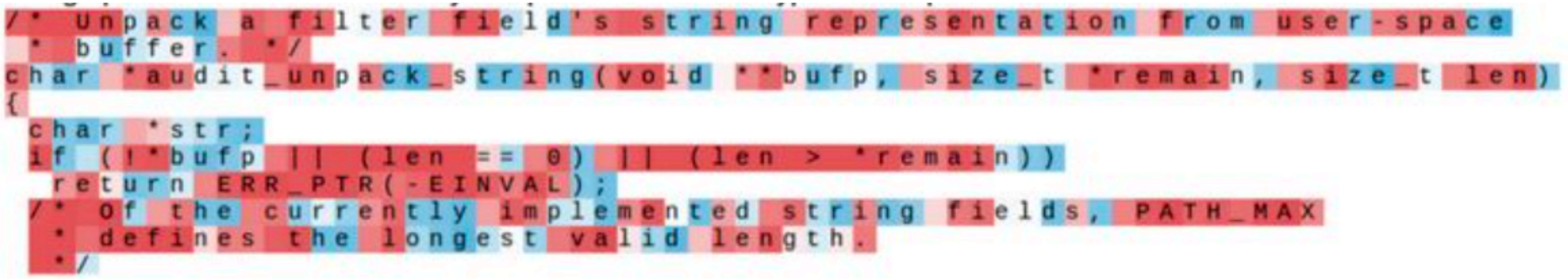
https://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN 시각화



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf
Karpathy, Andrej, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078 (2015).

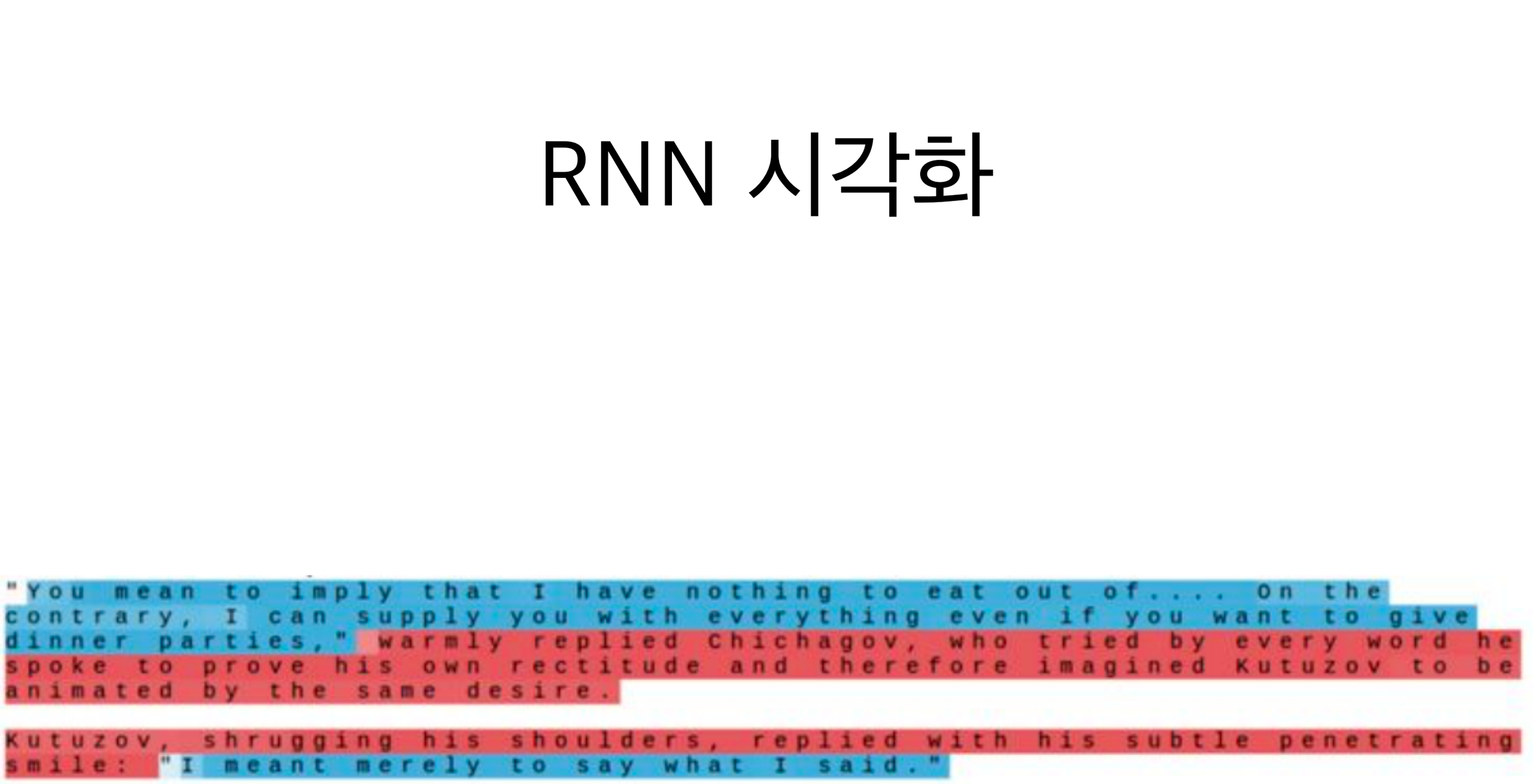
RNN 시각화



```
/*  Unpack a filter field's string representation from user-space
 *  buffer.  */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /*  Of the currently implemented string fields, PATH_MAX
   *  defines the longest valid length.
   */
```

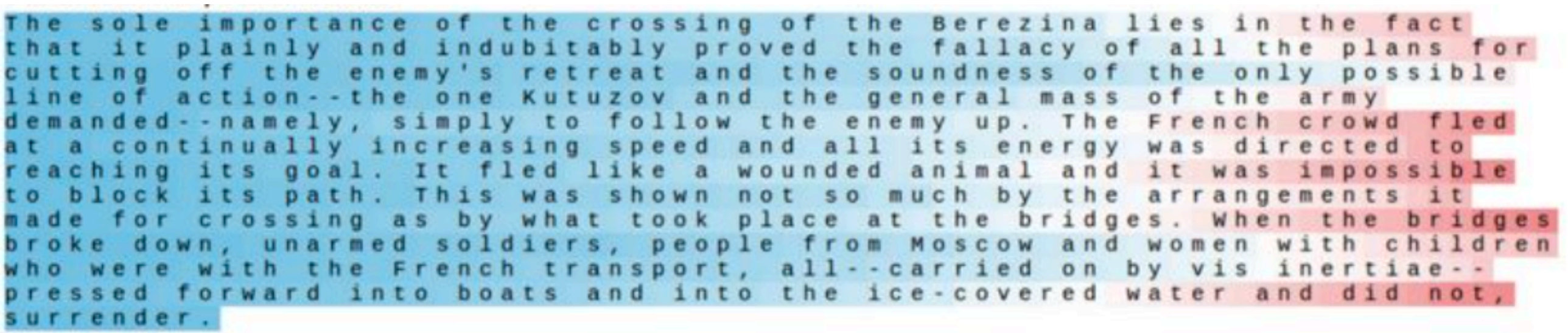
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf
Karpathy, Andrej, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078 (2015).

RNN 시각화



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf
Karpathy, Andrej, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078 (2015).

RNN 시각화



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Karpathy, Andrej, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078 (2015).

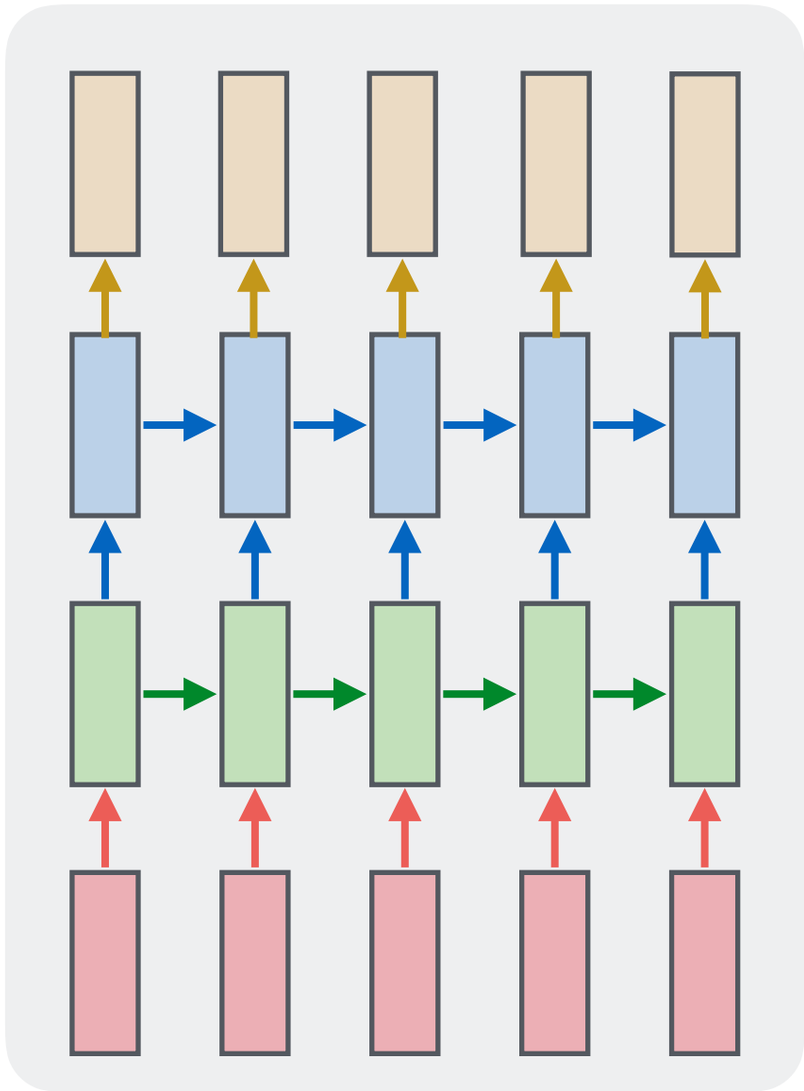
RNN 시각화

```
static int __dequeue_signal(struct sigpending *pending, sigset_t
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf
Karpathy, Andrej, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks." arXiv preprint arXiv:1506.02078 (2015).

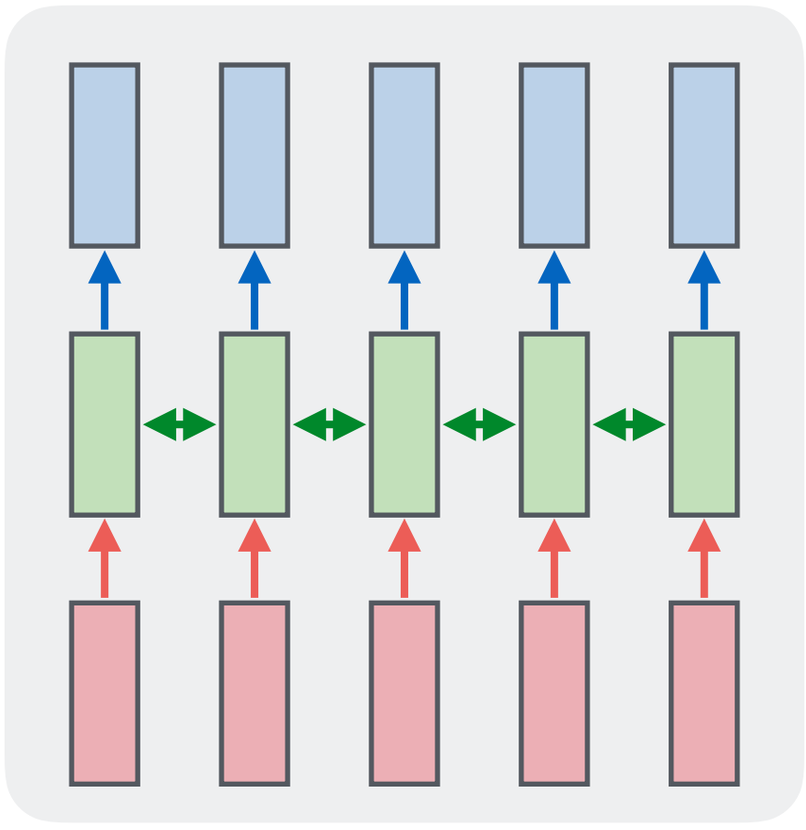
RNN 확장 모델

- 더 깊은 RNN
  - RNN은 모든 시퀀스에 대해 계산하기 때문에 연산량이 NN, CNN보다 많아 CNN처럼 깊게 쌓기는 어려움
  - 하지만 데이터와 연산 속도만 괜찮다면 더 깊은 RNN도 좋은 선택지



RNN 확장 모델

- bi-directional (양방향) RNN
  - 기존 RNN과 달리 양방향으로 데이터를 학습
  - 각 스텝마다 레이어가 하나 더 있는 형태
  - 일반적으로 더 좋은 성능



Vanishing gradient

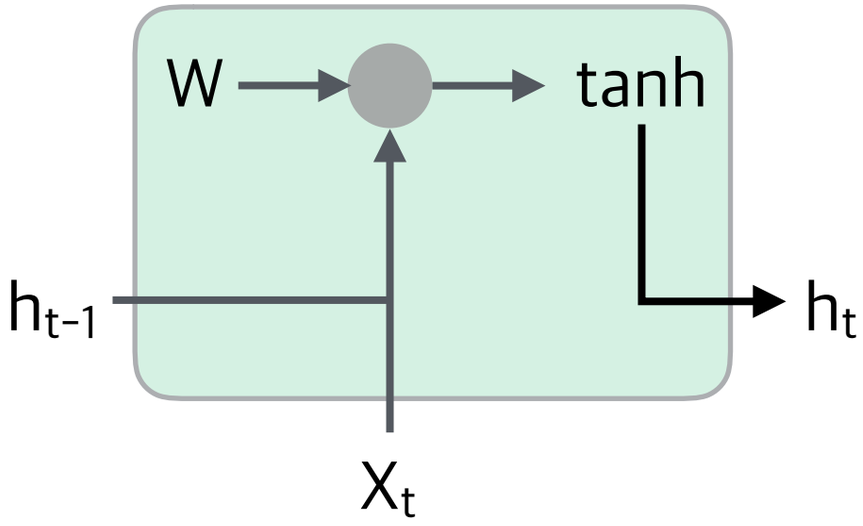
- RNN은 긴 시퀀스를 잘 처리하지 못함
  - 연관된 단어들 사이에 여러 스텝이 지난 경우 이를 까먹는 문제
  - 현재 단어가 예전에 등장한 단어와 연관있는 경우 문제가 생김
  - 각 스텝마다 그라디언트를 곱하면서 **vanishing**이 일어나기 때문!

- 해결책:
  - 활성 함수를 tanh에서 ReLU로 바꾸자
  - **LSTM, GRU와 같은 RNN 변형 모델 사용**

RNN 그라디언트 전파

$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left((W_{hh}\ W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

RNN 그라디언트 전파

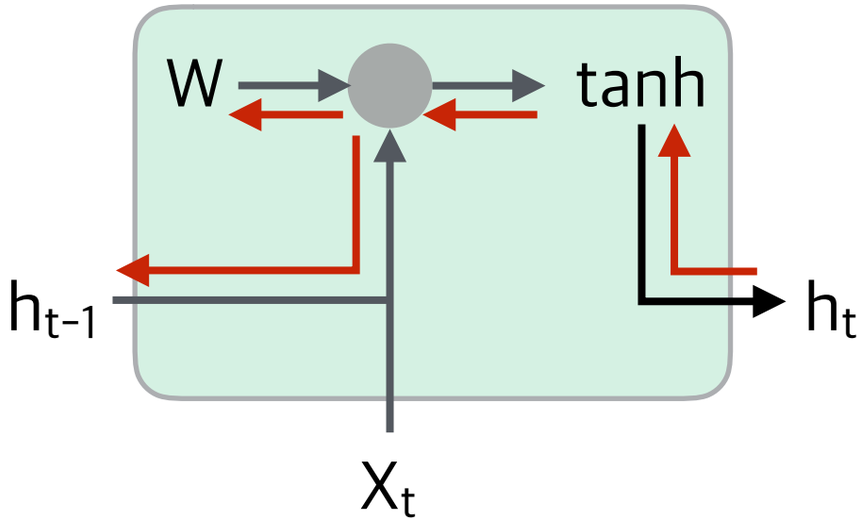


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= \tanh\left((W_{hh}\ W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

RNN 그라디언트 전파

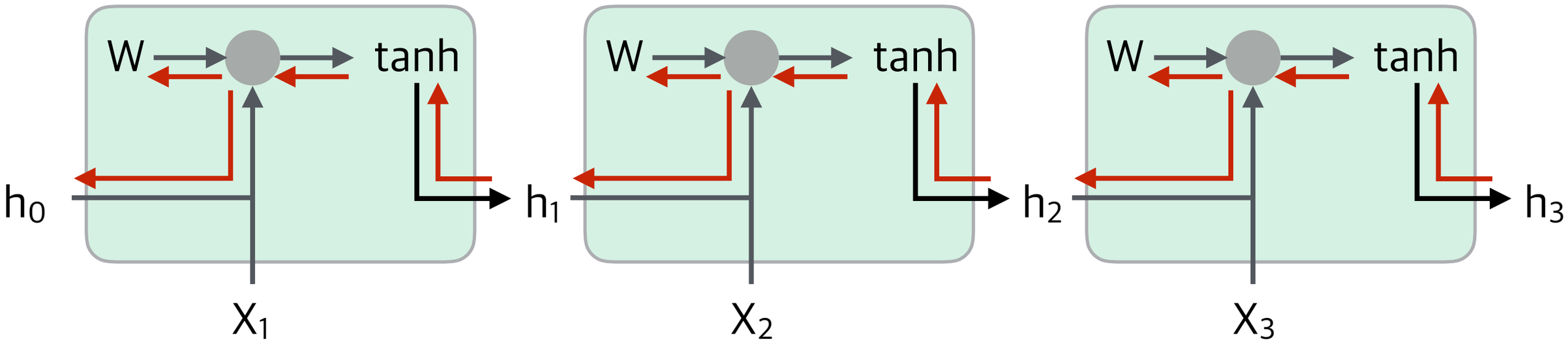


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$= \tanh\left((W_{hh}\ W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

RNN 그라디언트 전파

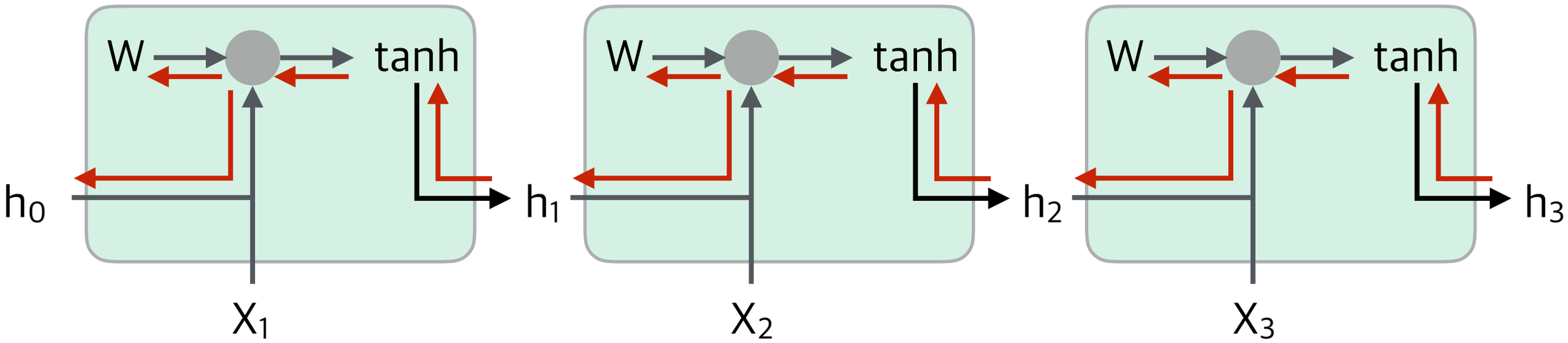


h_0에 흐르는 그라디언트는 많은 곱셈과 tanh를 수행 한 뒤 도착

-> 시퀀스가 길다면 그라디언트가 소멸될 가능성이 매우 커짐

-> 그라디언트 폭발의 우려

RNN 그라디언트 전파



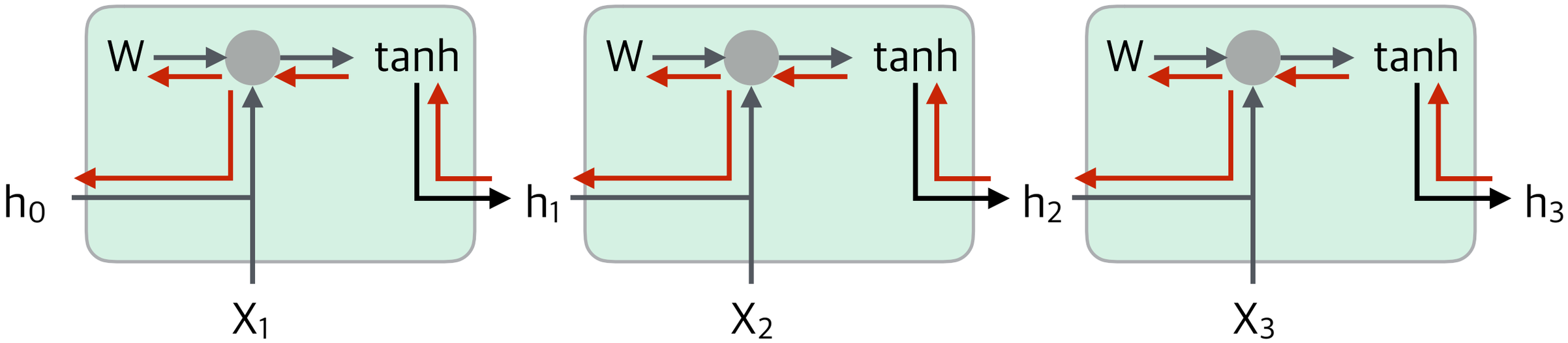
h_0에 흐르는 그라디언트는 많은 곱셈과 tanh를 수행 한 뒤 도착

-> 그라디언트 소멸:

-> 그라디언트 폭발: **그라디언트 클리핑**

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

RNN 그라디언트 전파



h_0에 흐르는 그라디언트는 많은 곱셈과 tanh를 수행 한 뒤 도착

-> 그라디언트 소멸: **LSTM, GRU**와 같은 RNN 변형 사용

-> 그라디언트 폭발

Long Short Term Memory

기본 RNN

$$h_t = \tanh\left(W\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}\right)$$

LSTM

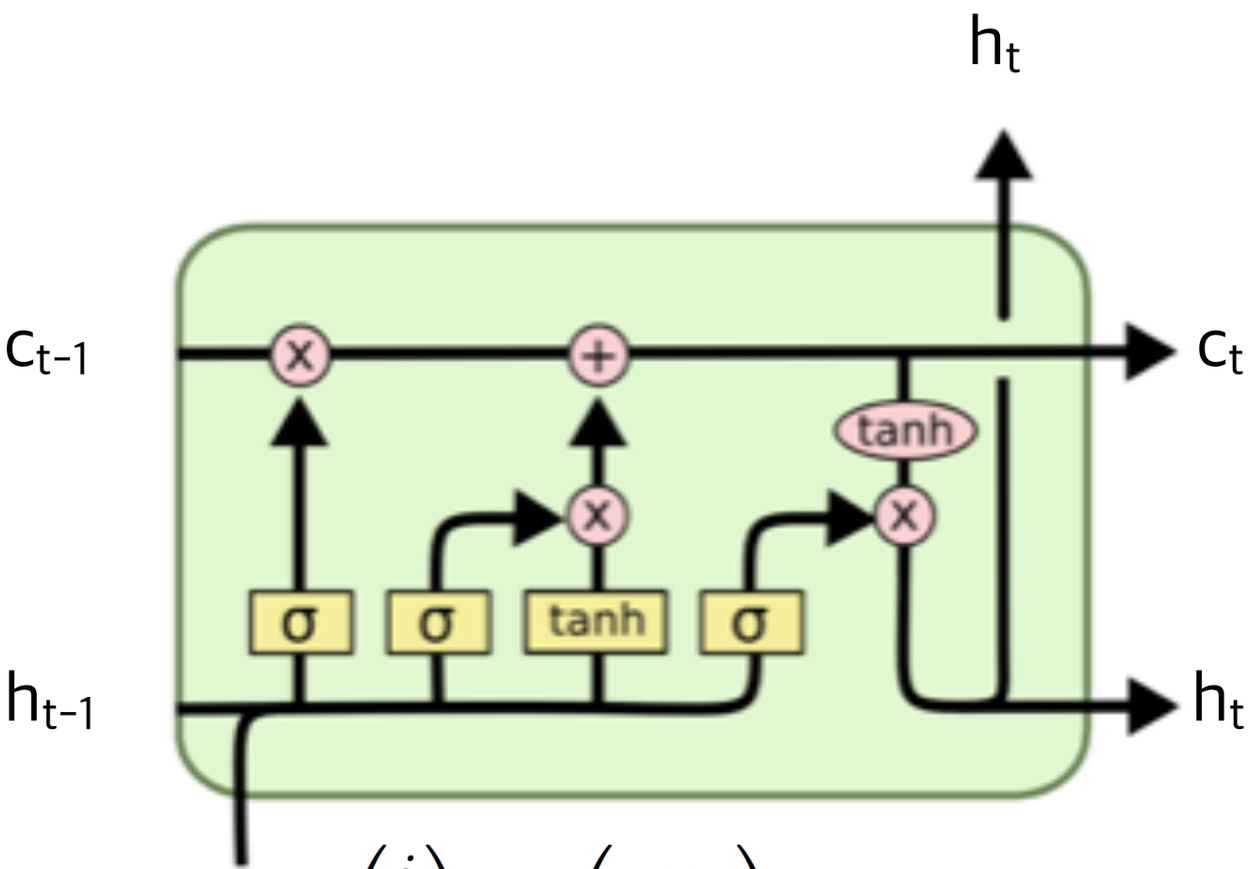
$$\begin{pmatrix}i\\f\\o\\g\end{pmatrix} = \begin{pmatrix}\sigma\\\sigma\\\sigma\\\tanh\end{pmatrix} W\begin{pmatrix}h_{t-1}\\x_t\end{pmatrix}$$

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

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

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

LSTM



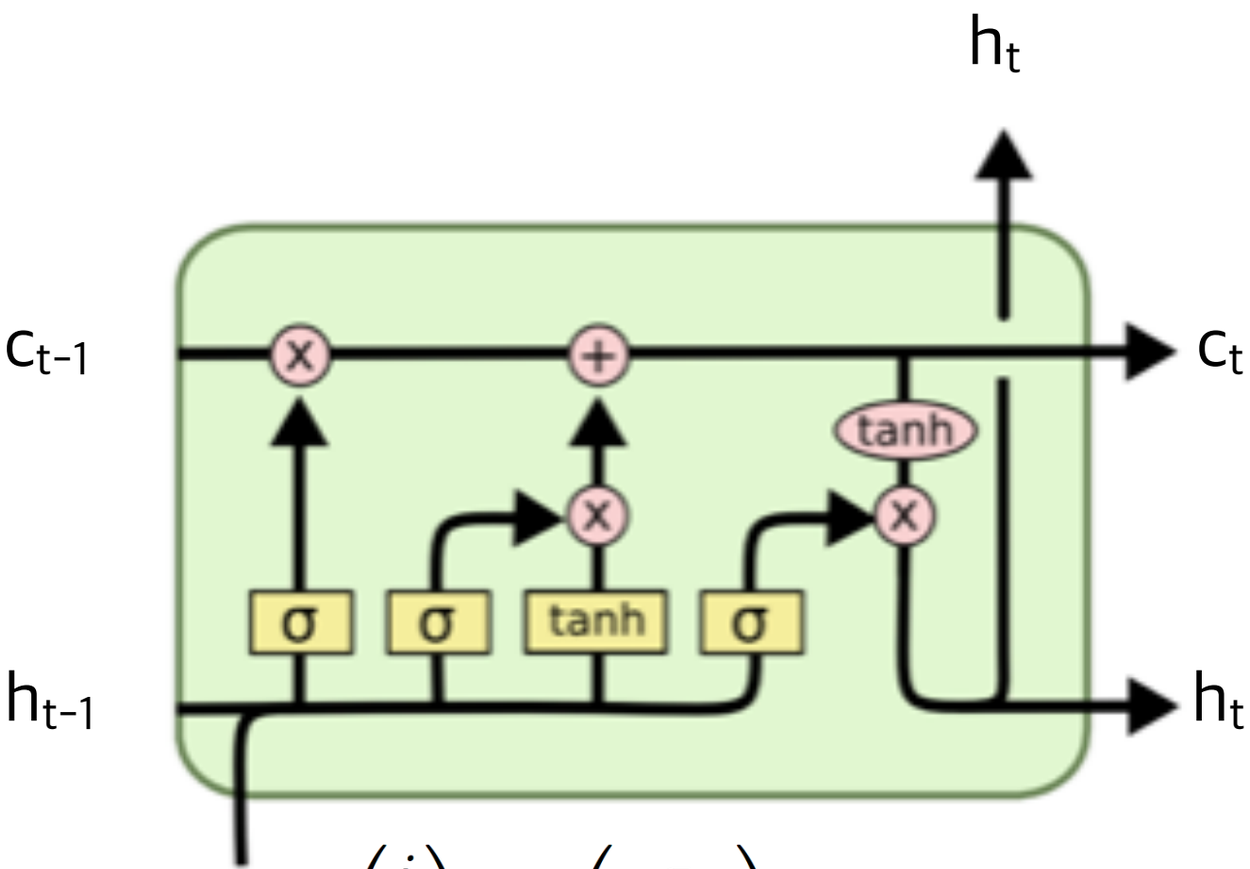
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

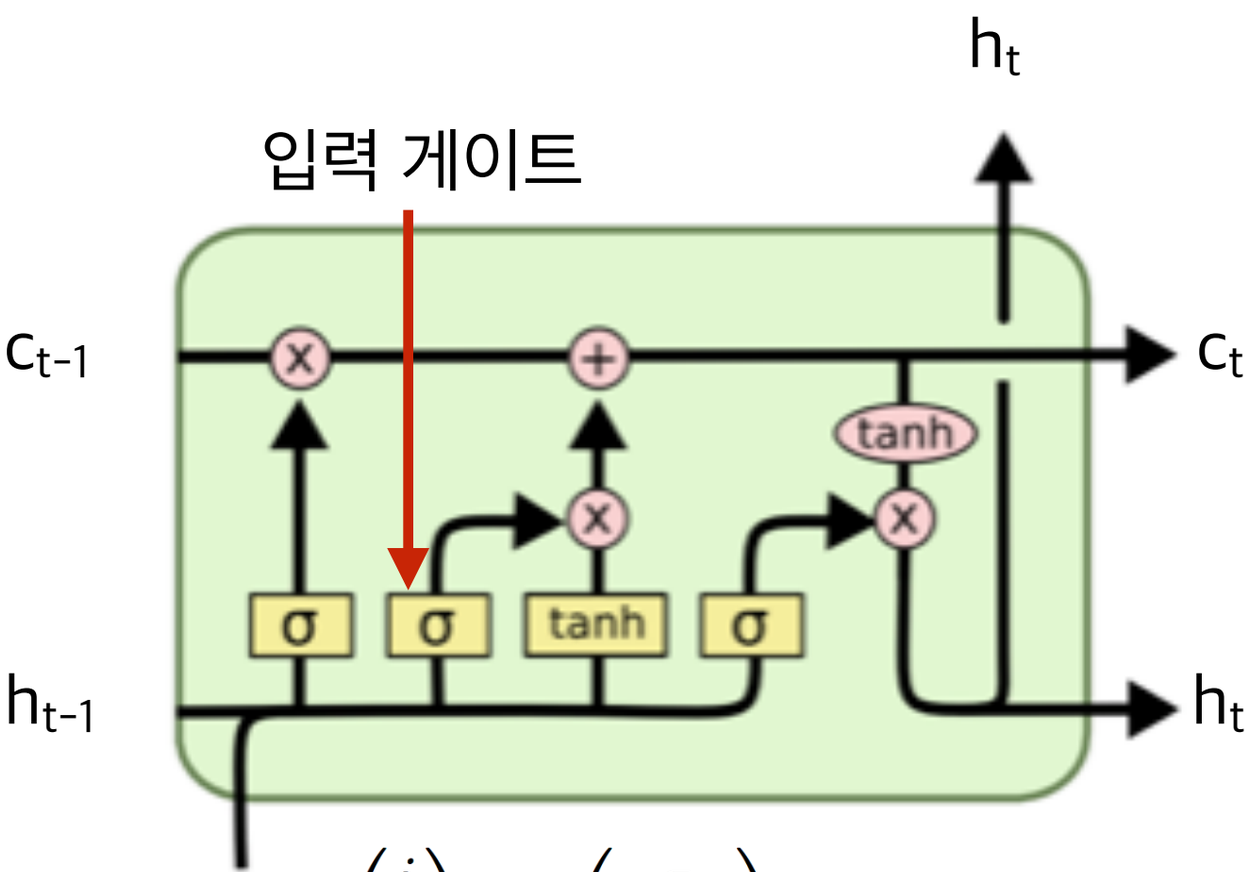
$$c_t = f \circ c_{t-1} + i \circ g$$

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

게이트:
시그모이드 함수에 의해 0~1로 제한되며,
벡터와 곱한다면 이 벡터를 얼마나 통과시킬 지 정해줌

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



i: 입력 게이트
상태 h_t을 계산 시 현재 입력값 x_t를
얼마나 사용할 지 결정

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

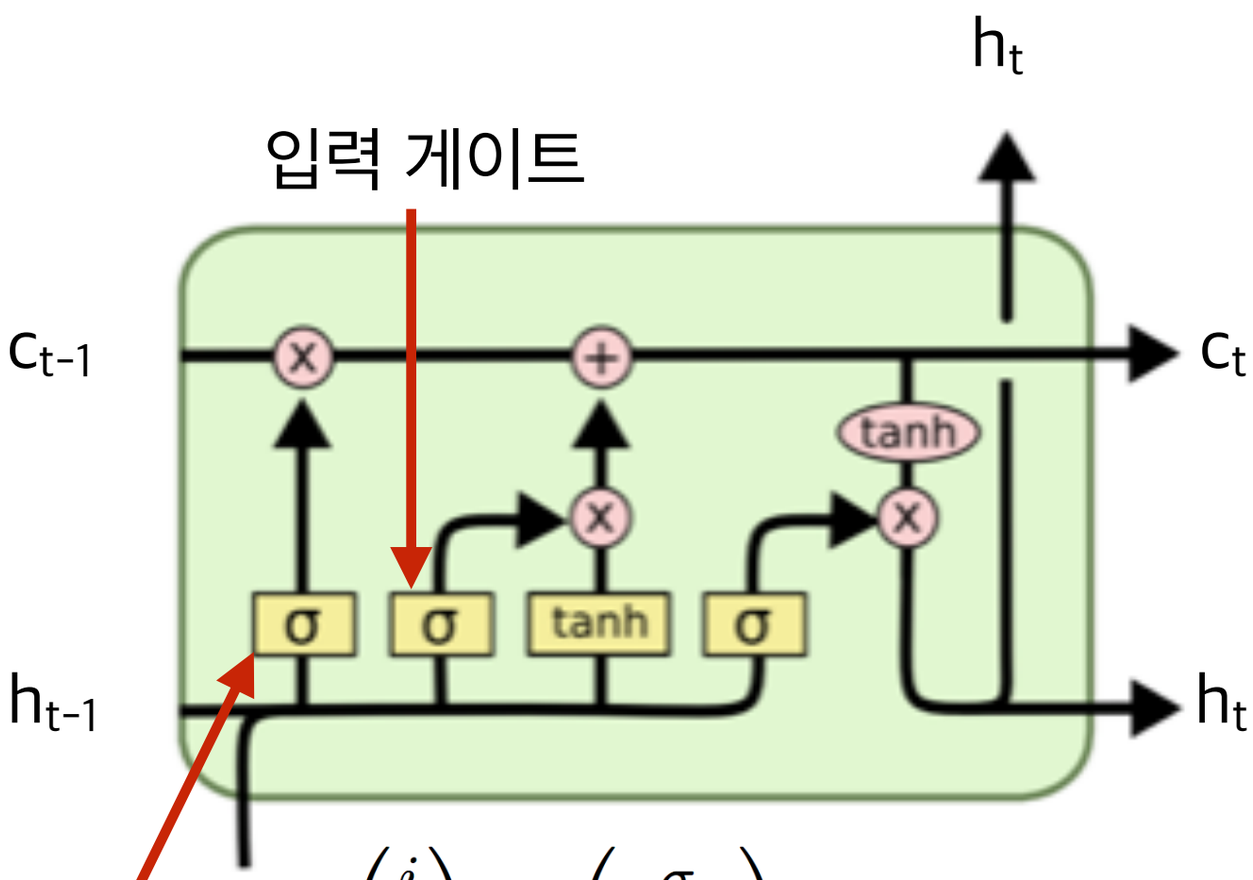
$$c_t = f \circ c_{t-1} + i \circ g$$

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

게이트:
시그모이드 함수에 의해 0~1로 제한되며,
벡터와 곱한다면 이 벡터를 얼마나 통과시킬 지 정해줌

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

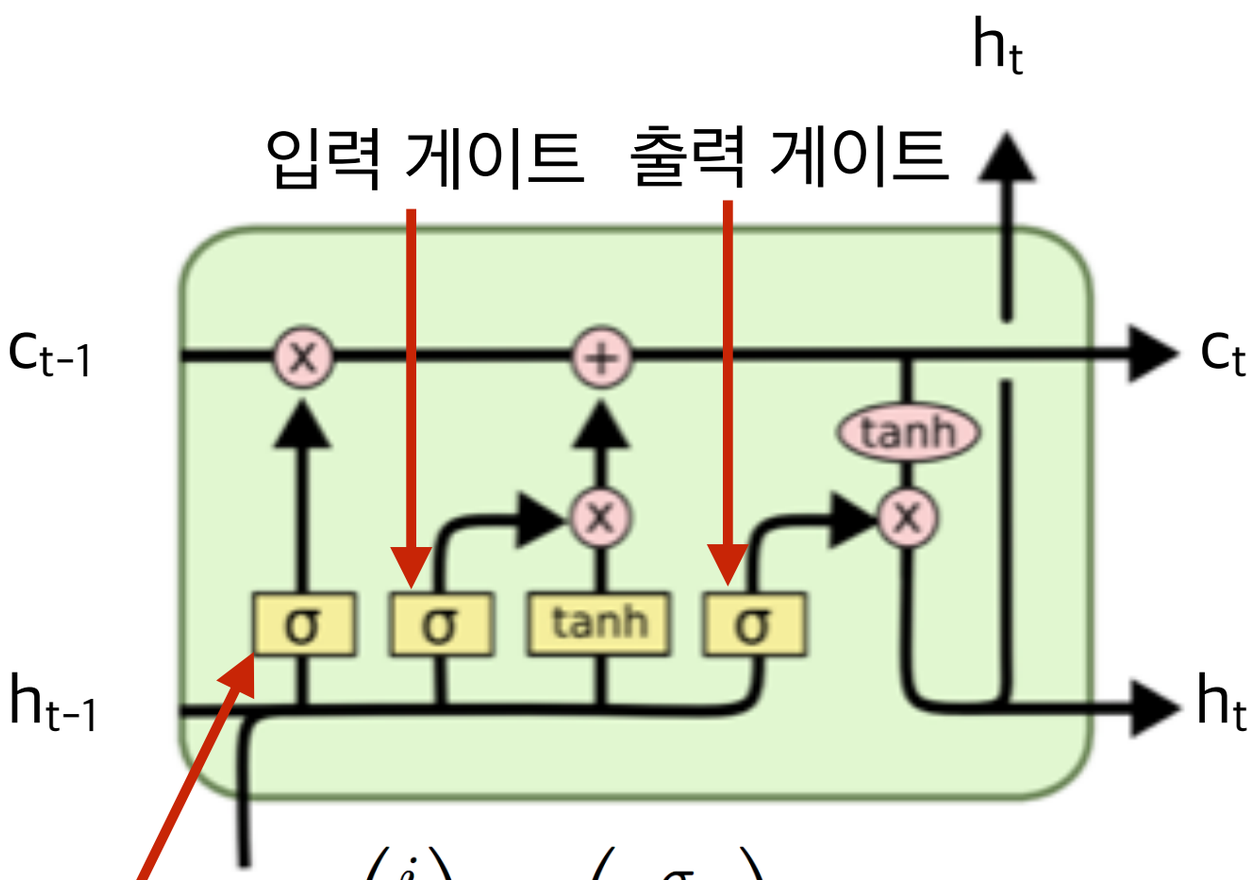
i: 입력 게이트
상태 h_t을 계산 시 현재 입력값 x_t를
얼마나 사용할 지 결정

f: 까먹음 게이트
상태 h_t을 계산 시 이전 내부 상태값 c_{t-1}를
얼마나 기억하고 사용할 지 결정

게이트:
시그모이드 함수에 의해 0~1로 제한되며,
벡터와 곱한다면 이 벡터를 얼마나 통과시킬 지 정해줌

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

i: 입력 게이트
상태 h_t을 계산 시 현재 입력값 x_t를
얼마나 사용할 지 결정

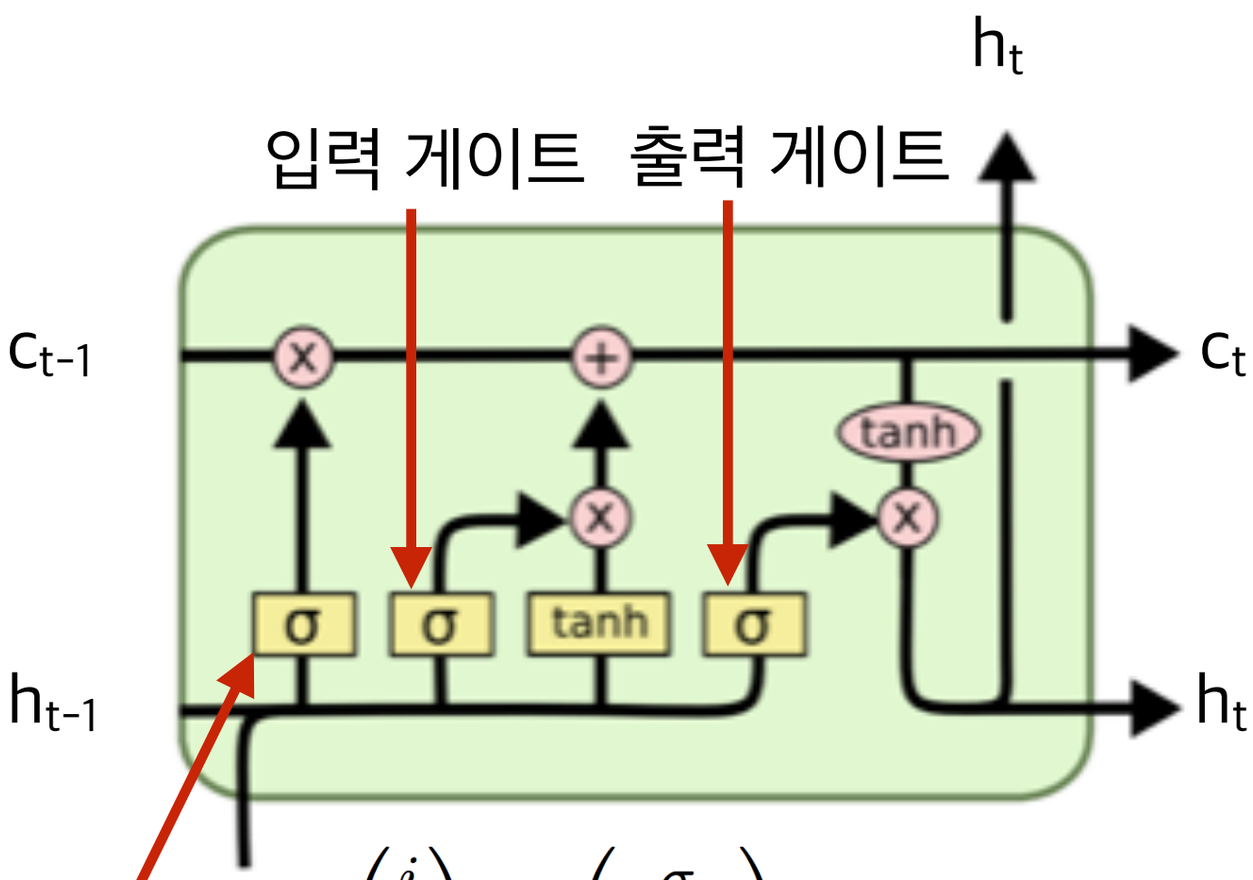
f: 까먹음 게이트
상태 h_t을 계산 시 이전 내부 상태값 c_{t-1}를
얼마나 기억하고 사용할 지 결정

o: 출력 게이트
내부 상태값 c_t를 외부에 얼마나
출력할지 결정

게이트:
시그모이드 함수에 의해 0~1로 제한되며,
벡터와 곱한다면 이 벡터를 얼마나 통과시킬 지 정해줌

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

i: 입력 게이트
상태 h_t을 계산 시 현재 입력값 x_t를
얼마나 사용할 지 결정

f: 까먹음 게이트
상태 h_t을 계산 시 이전 내부 상태값 c_{t-1}를
얼마나 기억하고 사용할 지 결정

o: 출력 게이트
내부 상태값 c_t를 외부에 얼마나
출력할지 결정

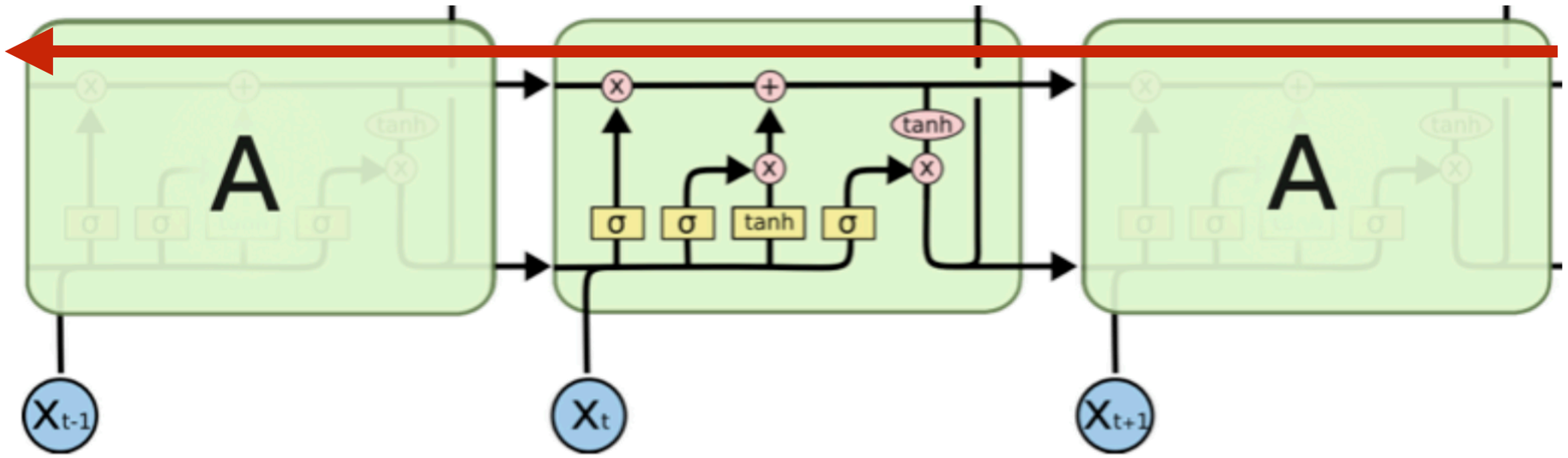
g:
기존 RNN의 상태값 h_t와 비슷한 역할
(동일한 수식)

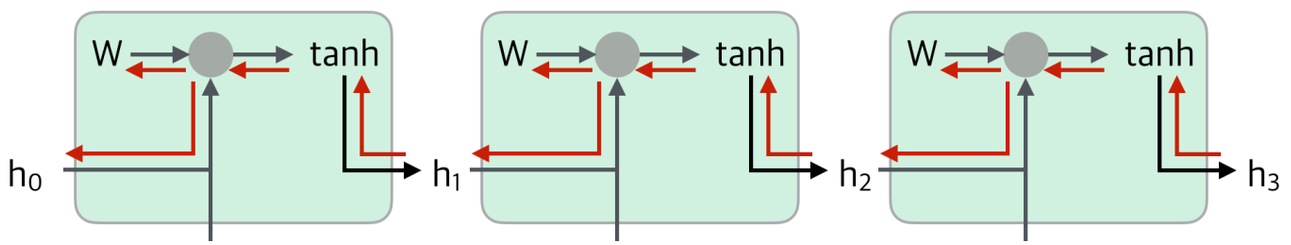
게이트:
시그모이드 함수에 의해 0~1로 제한되며,
벡터와 곱한다면 이 벡터를 얼마나 통과시킬 지 정해줌

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM

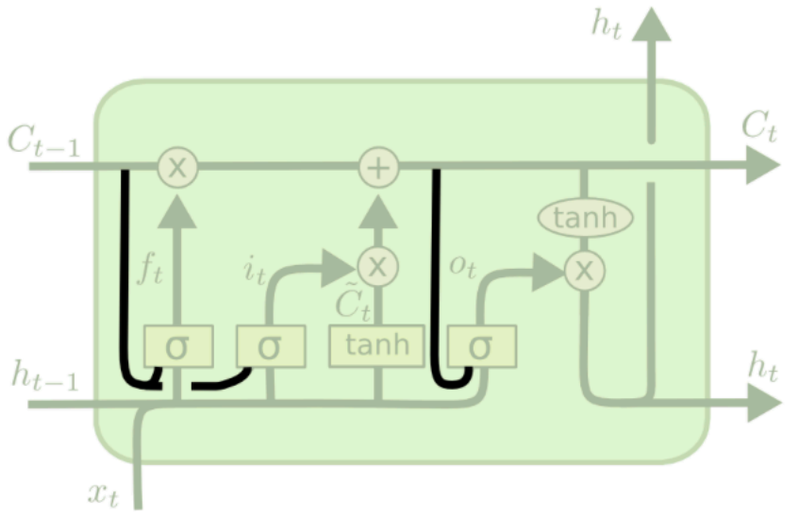
- LSTM은 RNN보다 그라디언트 흐름이 더 좋음
  - Gradient vanishing 문제를 해결
  - ResNet과 비슷한 아이디어?





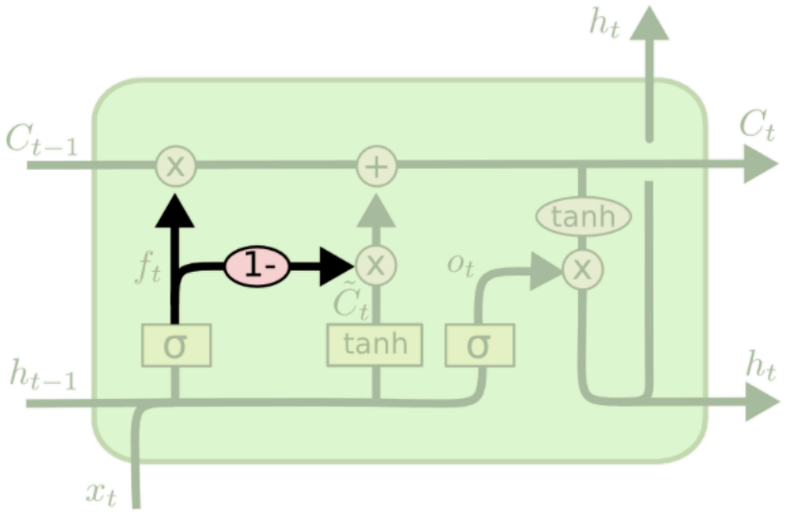
LSTM 파생형들

- Peephole LSTM



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

- **GRU**



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Gers, Felix A., and Jürgen Schmidhuber. "Recurrent nets that time and count." Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on. Vol. 3. IEEE, 2000.
Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

RNN Summary

- 문장과 같은 시퀀스 데이터를 처리할 때는 RNN 계열의 네트워크를 고려 <- **가변 길이의 데이터**를 입력 받을 수 있음
- 기본 RNN은 gradient vanishing / exploding 현상 우려
- Exploding은 그라디언트 클리핑으로 쉽게 해결 가능
- Vanishing은 **LSTM**, **GRU**와 같은 변형 모델로 해결

RNN 응용 사례

응용 사례 예시

- 이미지 캡셔닝, Attention [1, 2]
- Visual Question Answering [3]
- seq2seq [4]

[1] Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." CVPR. 2015.
[2] Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." ICML. 2015.
[3] Antol, Stanislaw, et al. "Vqa: Visual question answering." ICCV. 2015.
[4] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." NIPS. 2014.

이미지 캡셔닝

a man riding skis down a snow covered slope

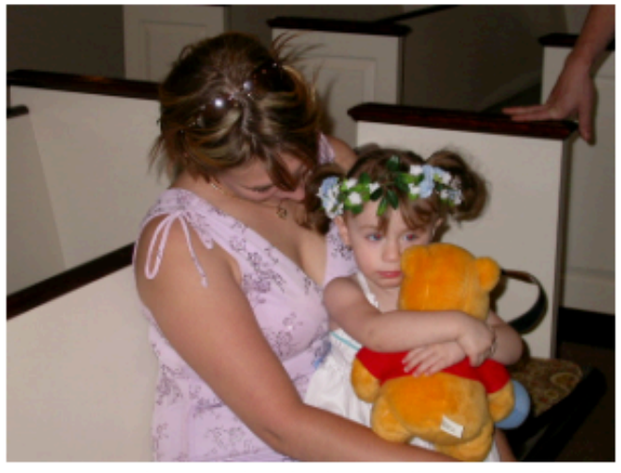
a group of people playing a game with nintendo wii controllers

a pile of luggage sitting on the ground

a view of a building with a clock on the top of it

a small boat in a body of water

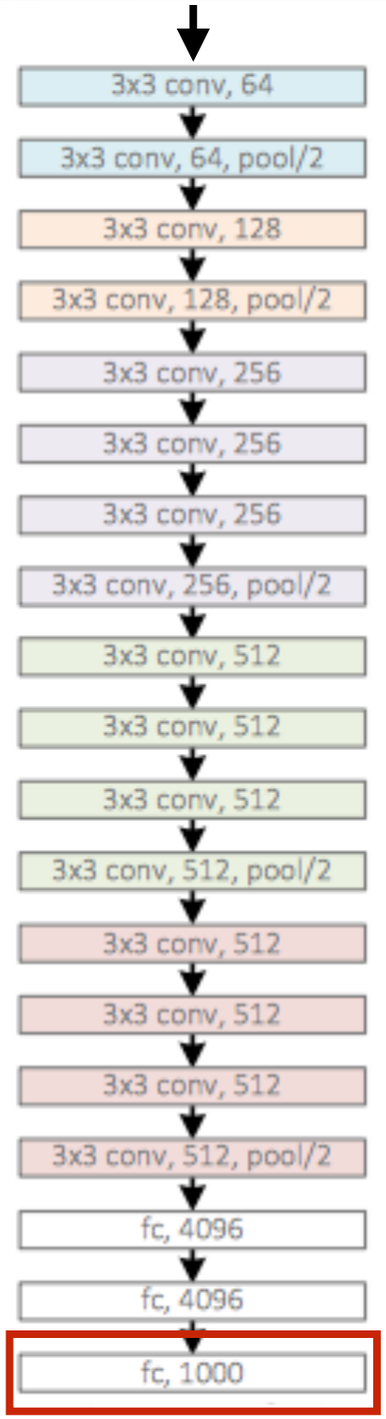
a motorcycle parked on the side of a road



a woman in a red shirt is holding a stuffed animal

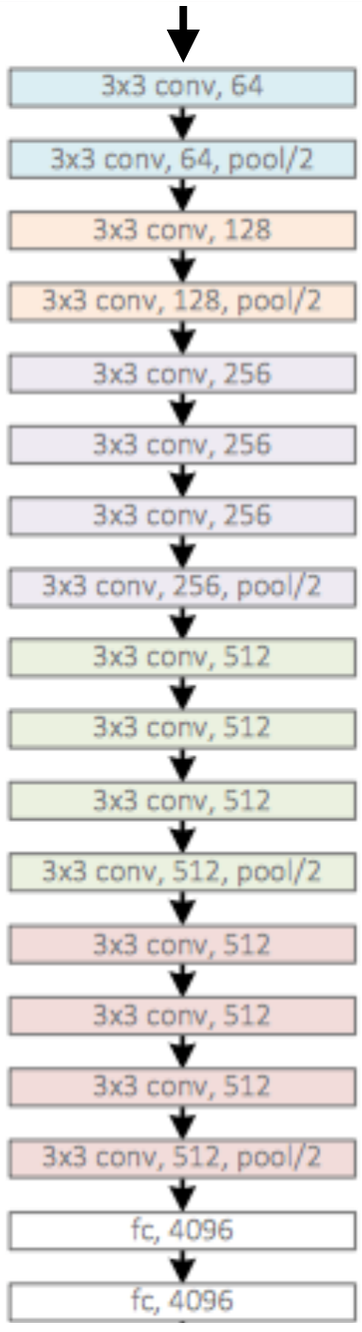
a young girl sitting at a table with a plate of food

http://cs.stanford.edu/people/karpathy/neuraltalk2/demo.html



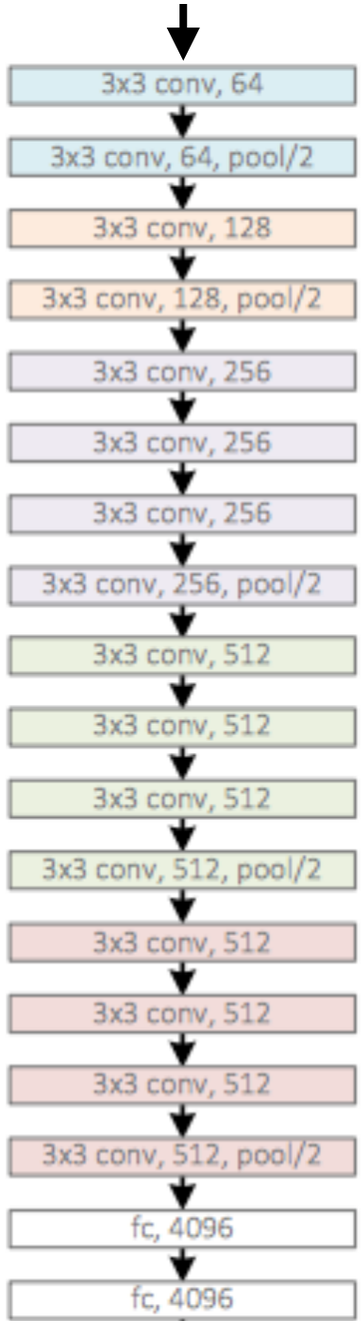
이미지 분류와 관련된 레이어는 제거

Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

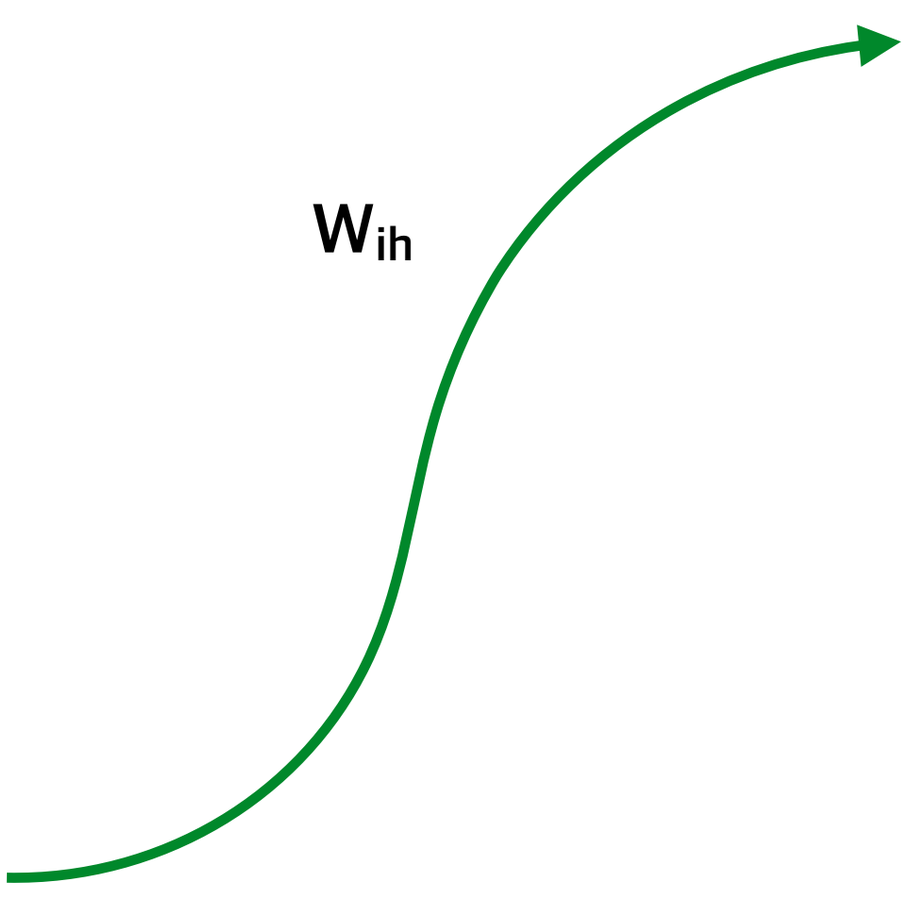


사전 학습된 CNN에서
이미지 특징 추출

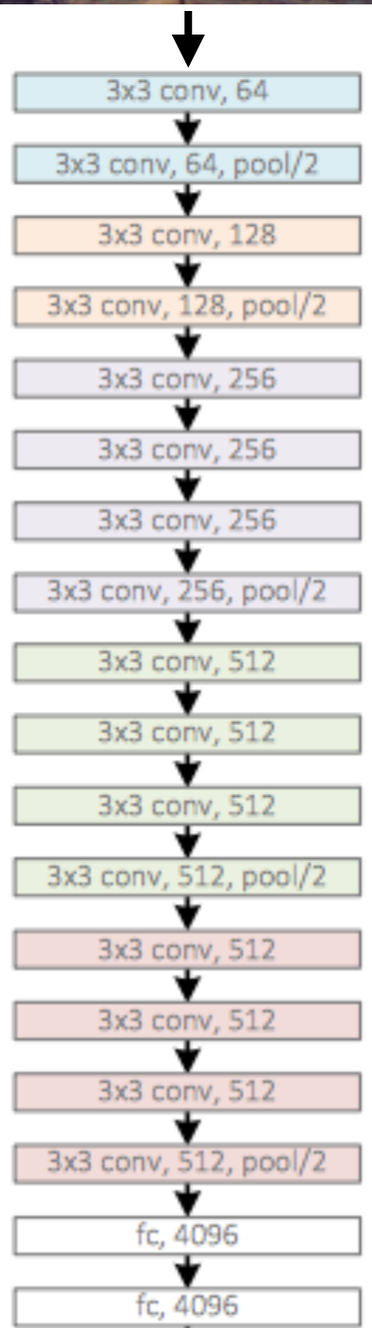
Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.



사전 학습된 CNN에서
이미지 특징 추출

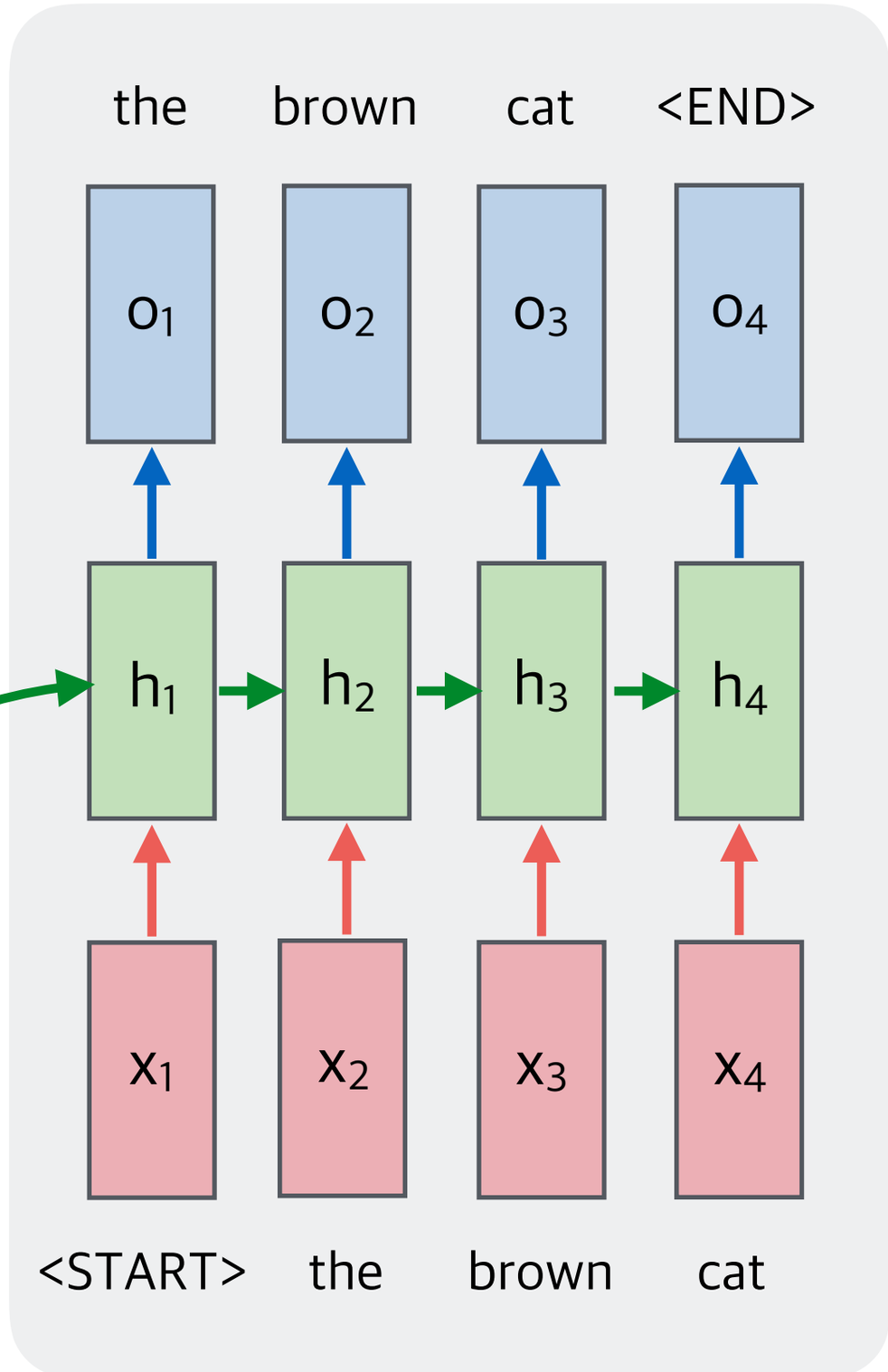


Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

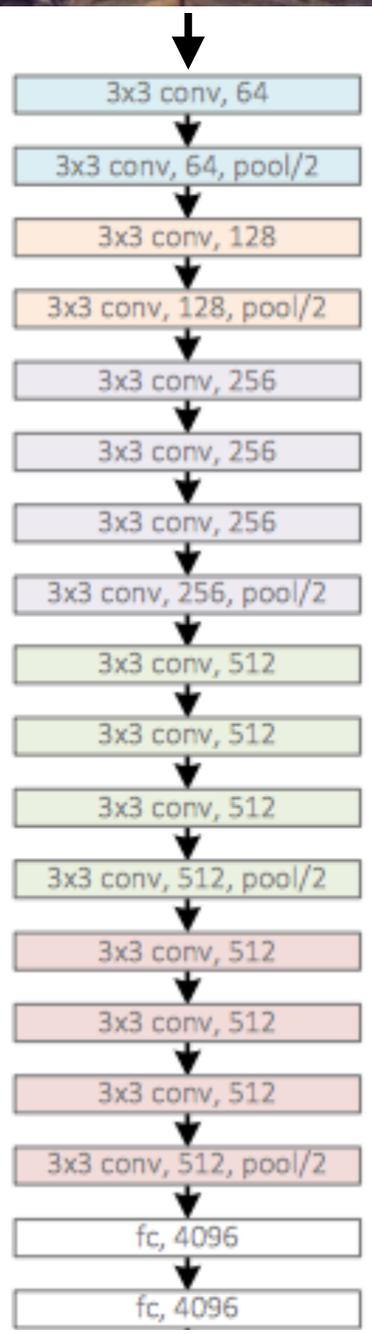


사전 학습된 CNN에서
이미지 특징 추출

W_{ih}

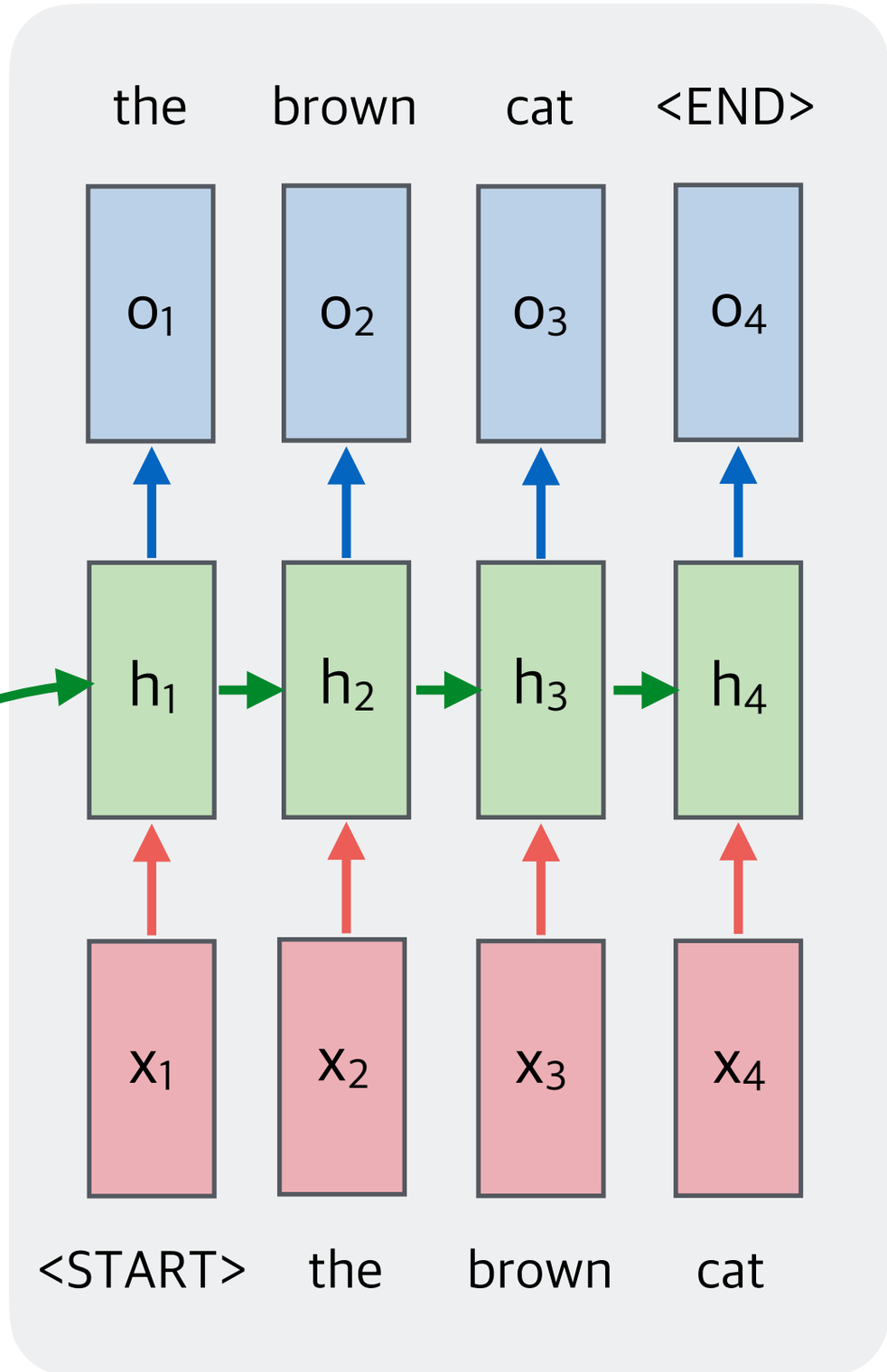


Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.



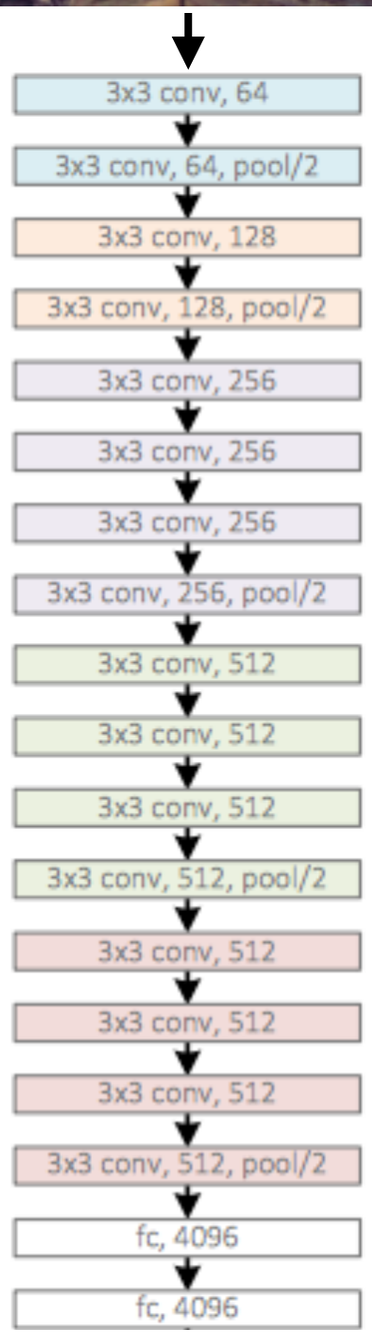
사전 학습된 CNN에서
이미지 특징 추출

W_{ih}



첫 단어는 <START> 토큰으로 시작 (문장의 시작을 명시)

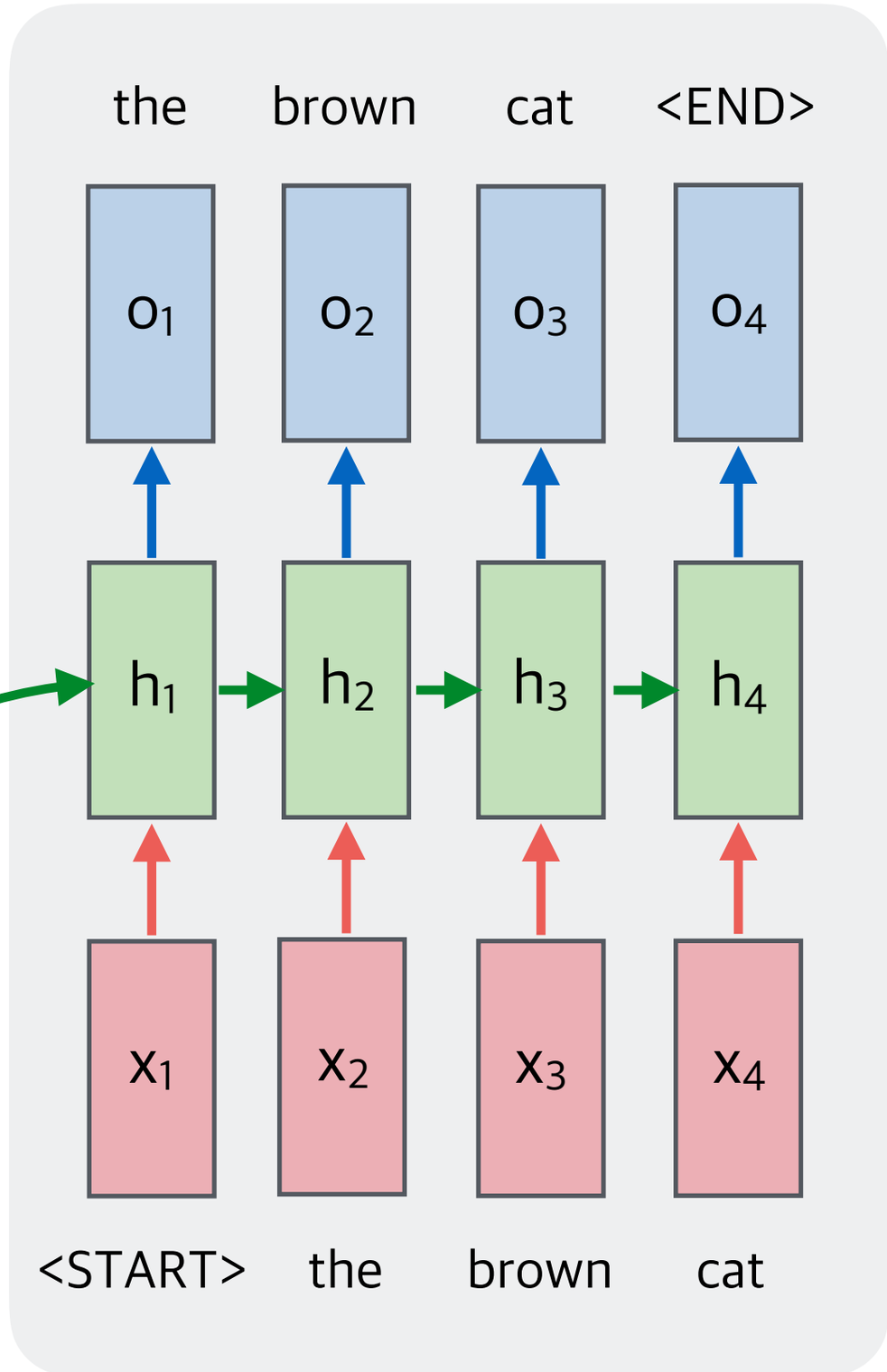
Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.



사전 학습된 CNN에서
이미지 특징 추출

W_{ih}

<END> 토큰 출력 -> 문장의 끝

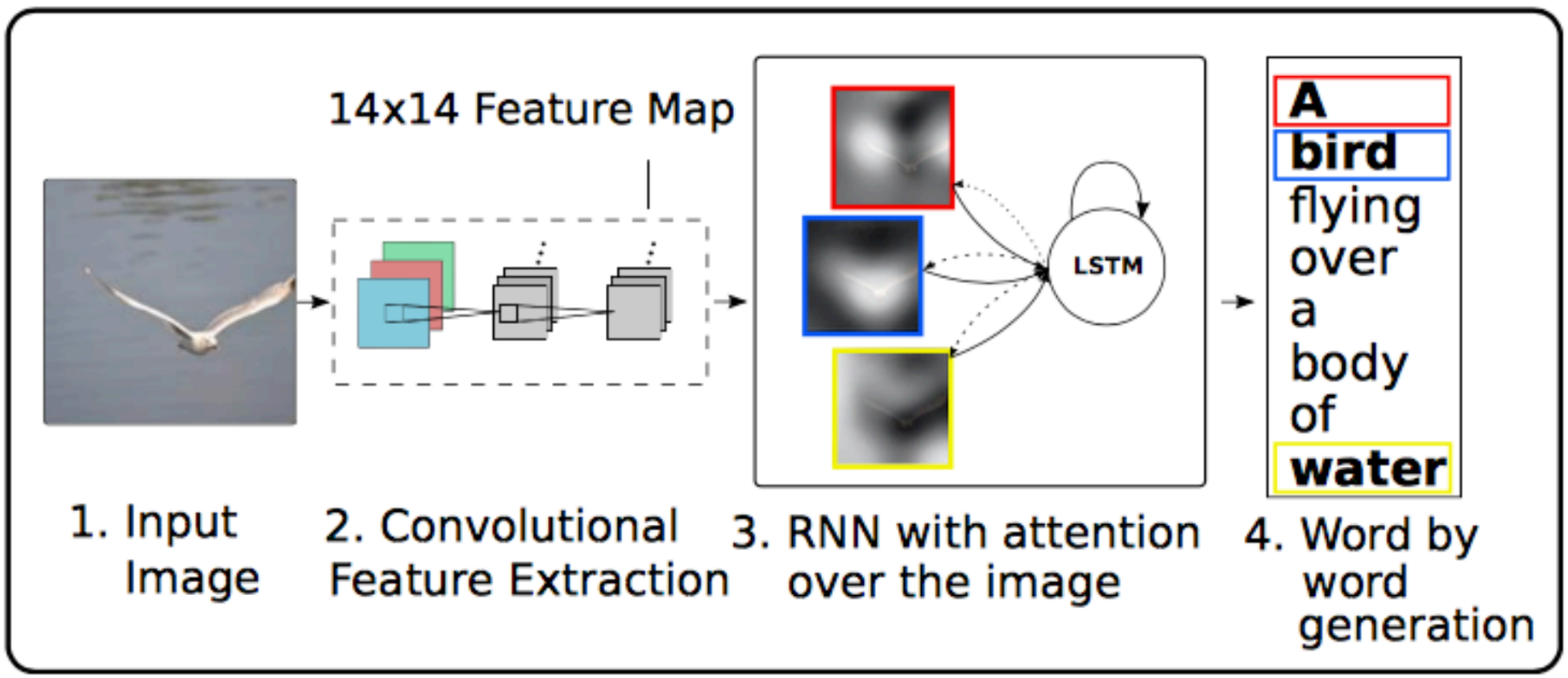


첫 단어는 <START> 토큰으로 시작 (문장의 시작을 명시)

Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

Attention

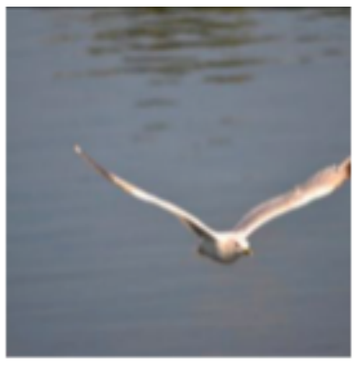
- 기존 이미지 캡셔닝의 한계점
  - 이미지를 처음 **딱 한번**만 보기 때문에 문장을 생성하면서 이미지 정보를 까먹을 수 있음
  - 해결책: 단어를 생성할 때 마다 이미지의 특정 부분을 **주목(attention)**



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

Attention

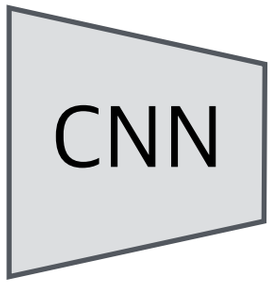
Attention



H x W x 3

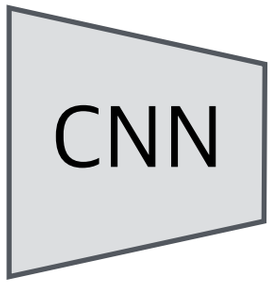
Attention

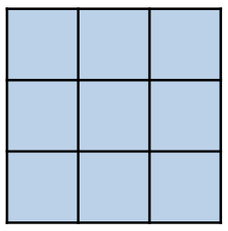
H x W x 3



Attention

H x W x 3

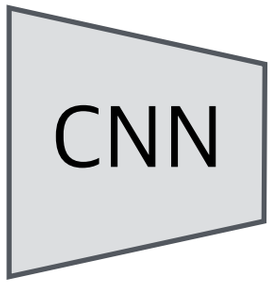


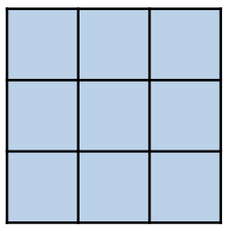


L x D

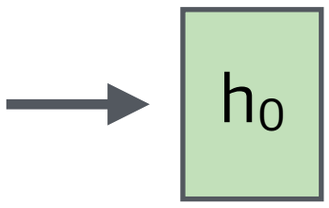
Attention

H x W x 3

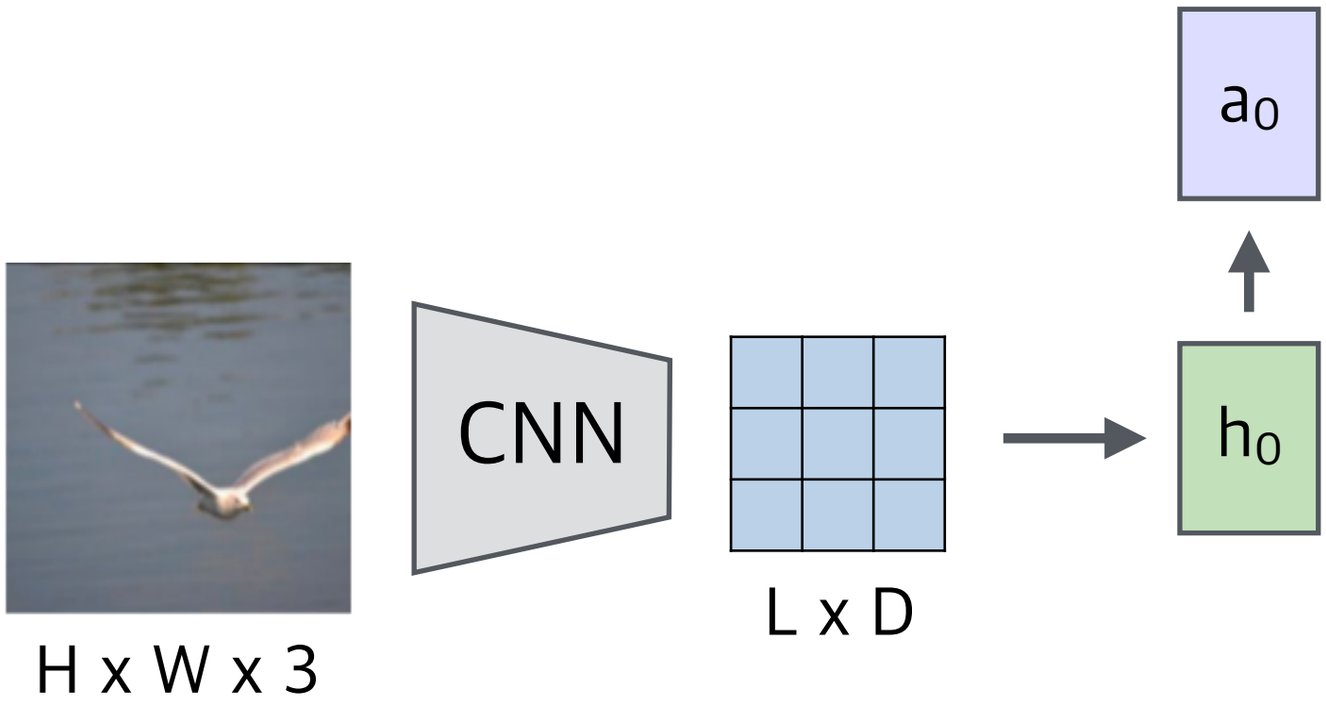




L x D

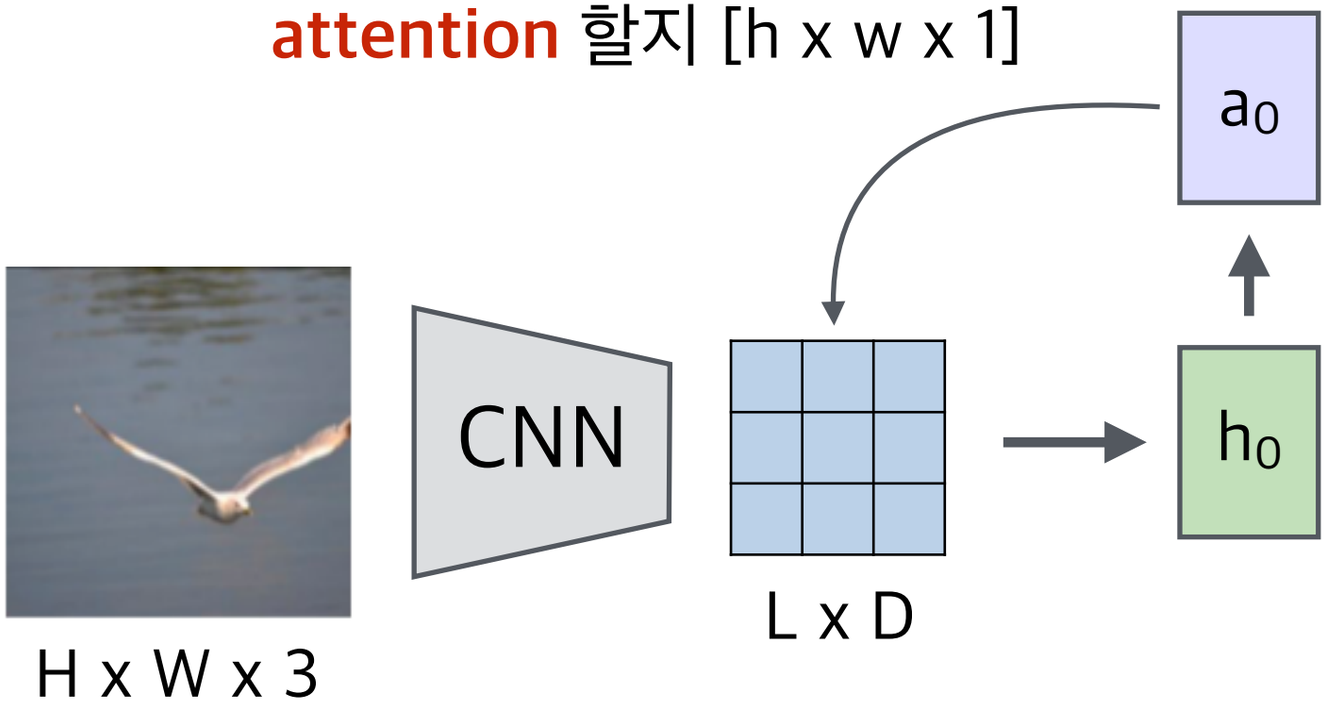


Attention



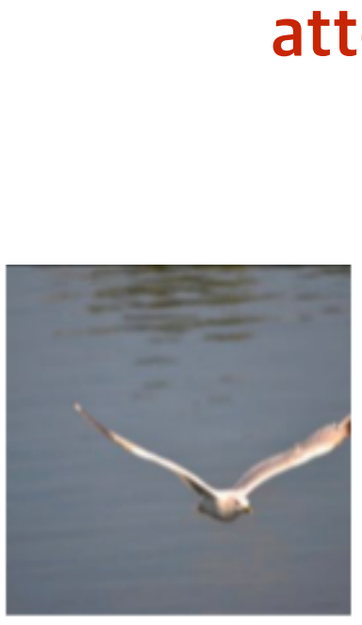
Attention

이미지 특징맵의 어느 영역을
attention 할지 [h x w x 1]

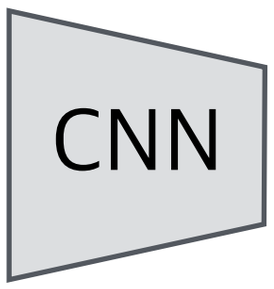


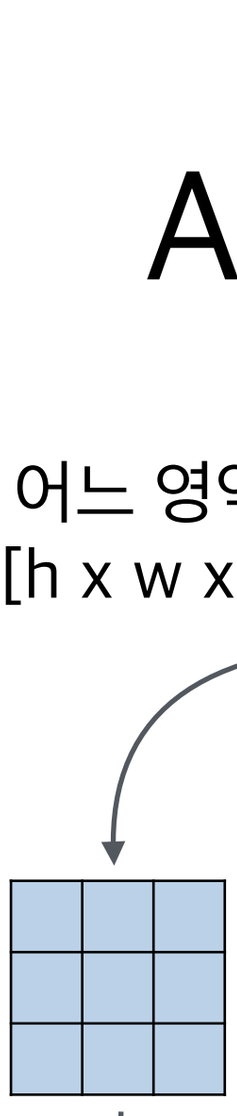
Attention

이미지 특징맵의 어느 영역을
attention 할지 [h x w x 1]

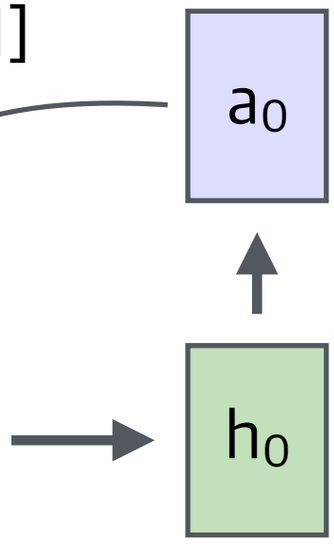


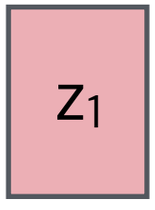
H x W x 3





L x D

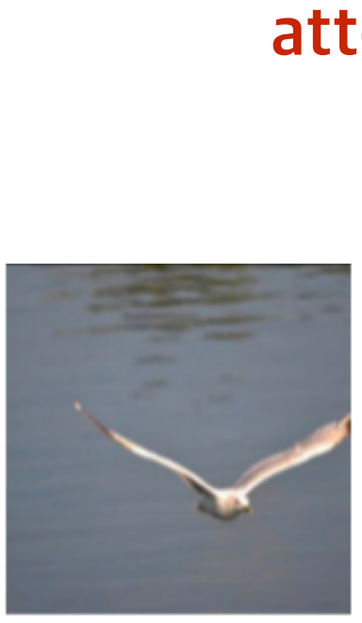




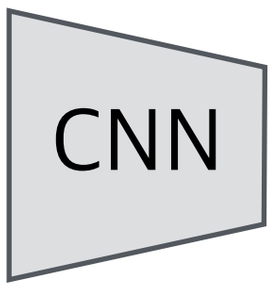
$$z = \sum_{i=1}^{L} a_i v_i$$

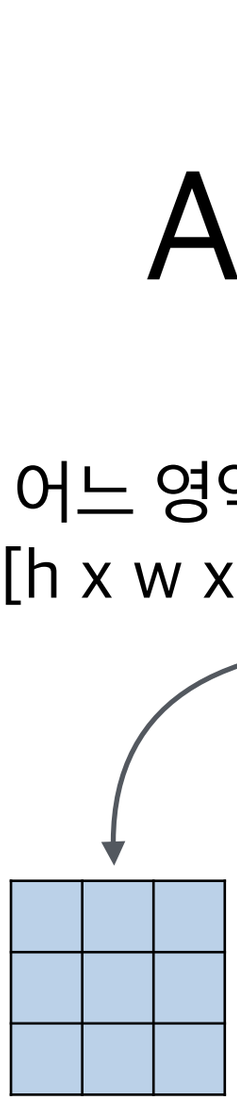
Attention

이미지 특징맵의 어느 영역을
attention 할지 [h x w x 1]

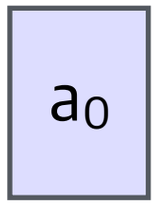


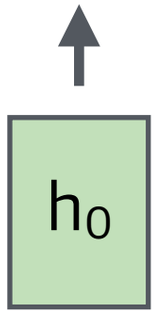
H x W x 3

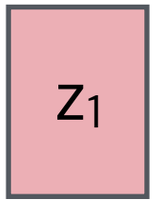




L x D

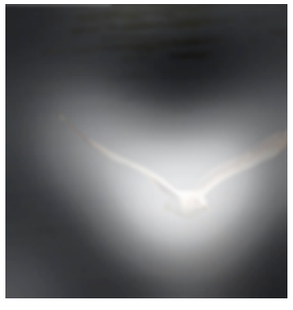






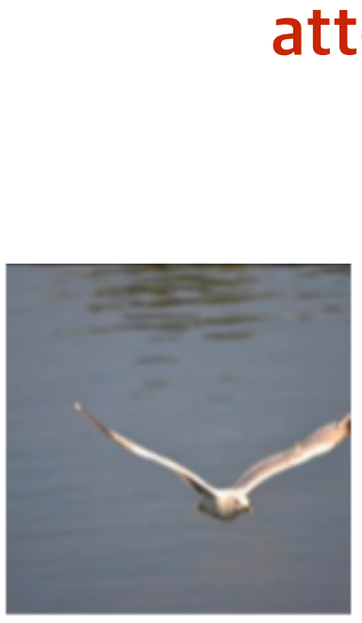
$$z = \sum_{i=1}^{L} a_i v_i$$

z 예시)

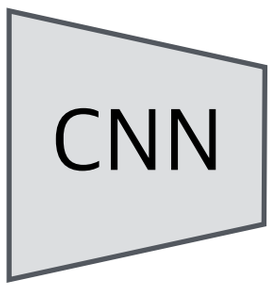


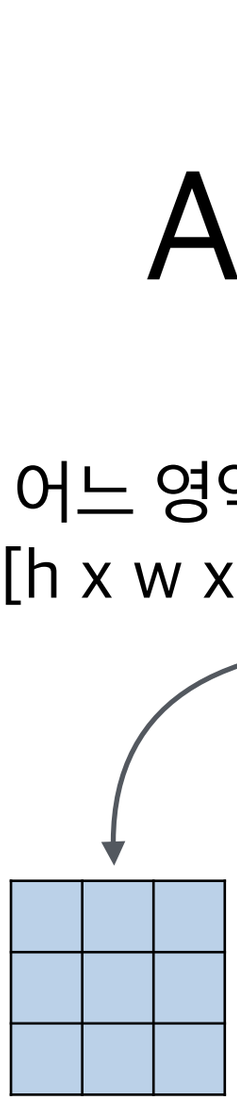
Attention

이미지 특징맵의 어느 영역을
attention 할지 [h x w x 1]

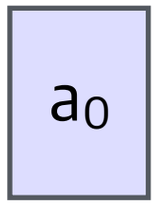


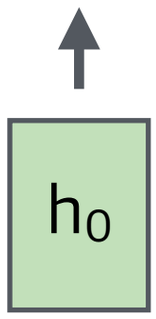
H x W x 3

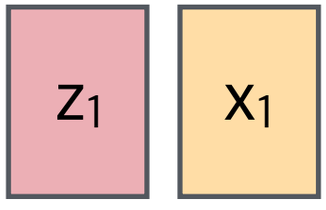




L x D



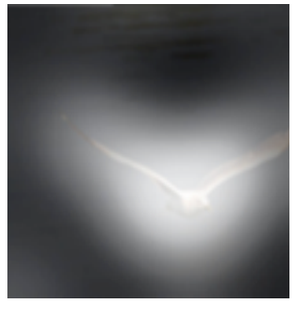




입력 단어

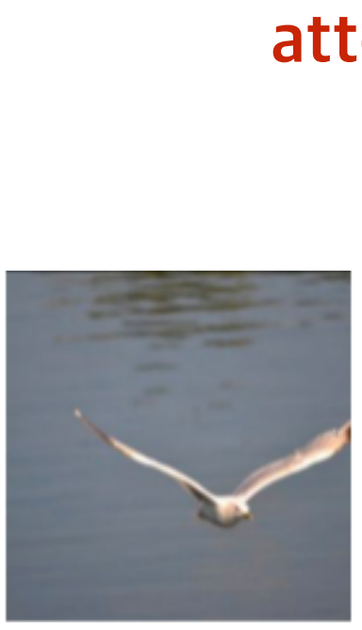
$$z = \sum_{i=1}^{L} a_i v_i$$

z 예시)



Attention

이미지 특징맵의 어느 영역을
attention 할지 [h x w x 1]



H x W x 3

CNN

L x D

a_0 a_1 o_1

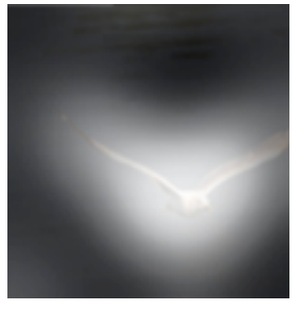
h_0 h_1

z_1 x_1

입력 단어

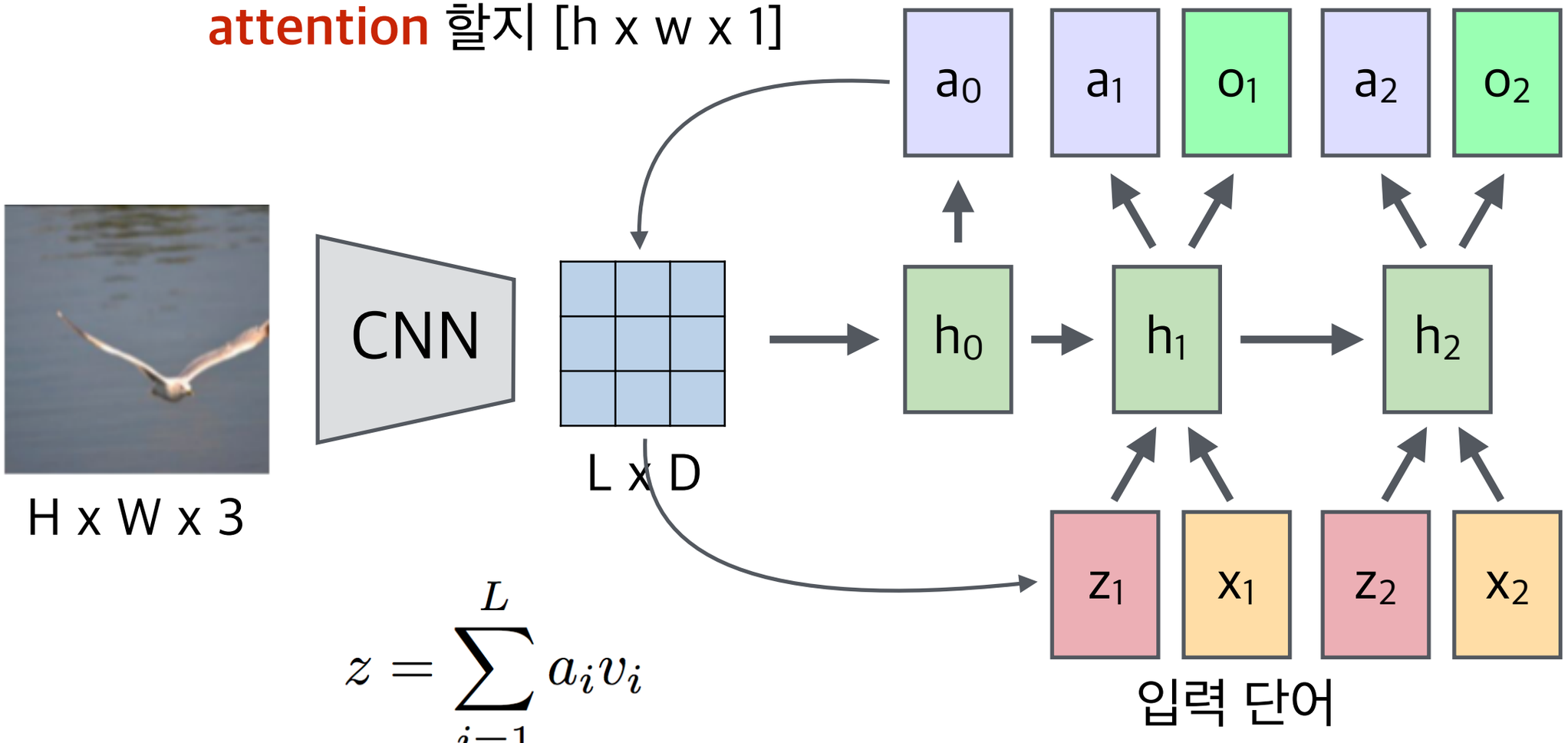
$$z = \sum_{i=1}^{L} a_i v_i$$

z 예시)



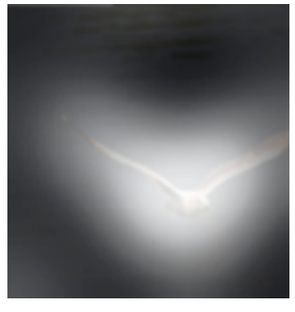
Attention

이미지 특징맵의 어느 영역을
attention 할지 [h x w x 1]



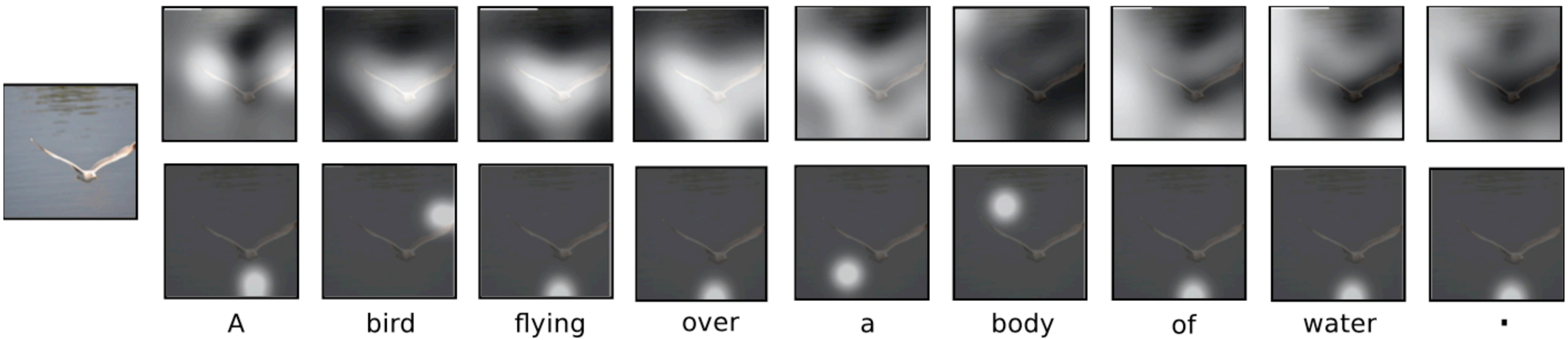
$$z = \sum_{i=1}^{L} a_i v_i$$

z 예시)



Attention

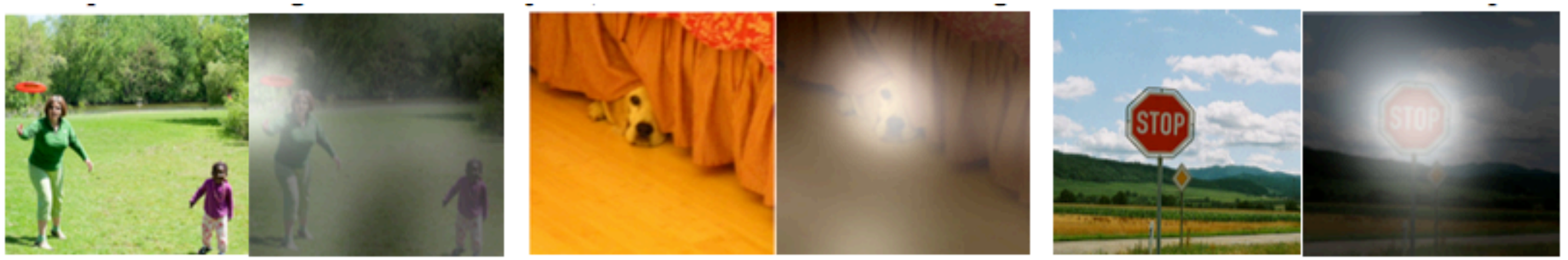
Soft attention



Hard attention
(미분이 안되는 연산이 포함되어 강화학습으로 학습)

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

Attention



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

실패 케이스

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

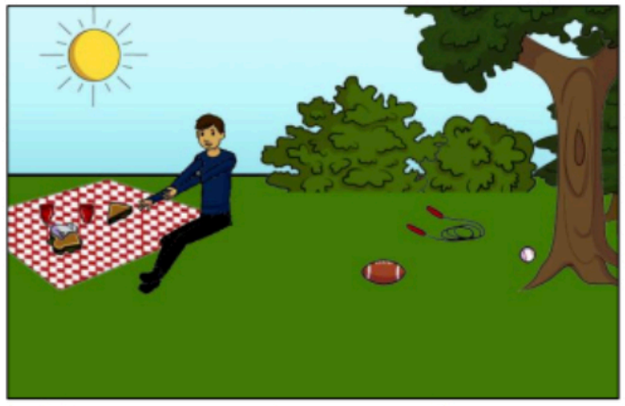
A man is talking on his cell phone
while another man watches.

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International Conference on Machine Learning. 2015.

Visual Question Answering

What color are her eyes?
What is the mustache made of?

How many slices of pizza are there?
Is this a vegetarian pizza?



Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

http://visualqa.org/

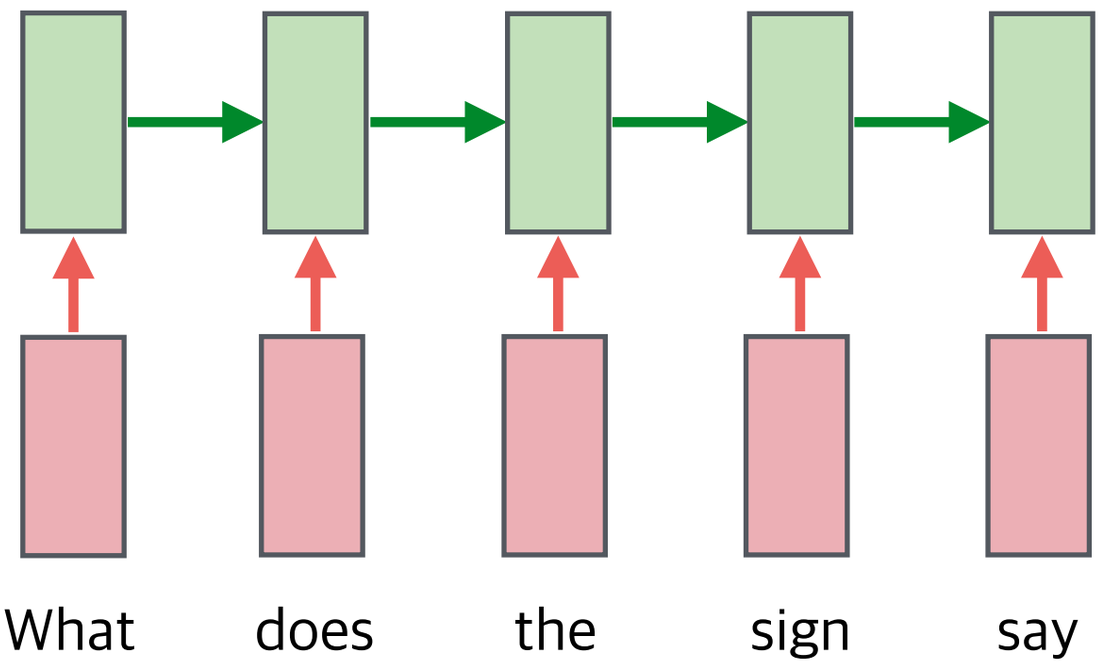
VQA 기본 모델

Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

VQA 기본 모델

Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

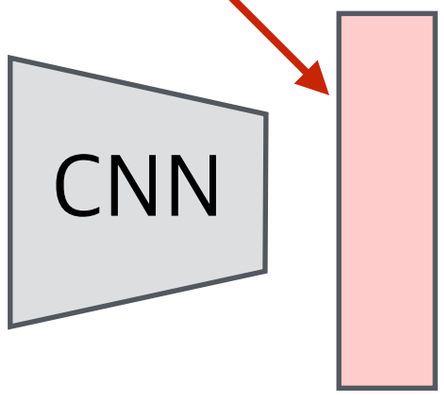
VQA 기본 모델

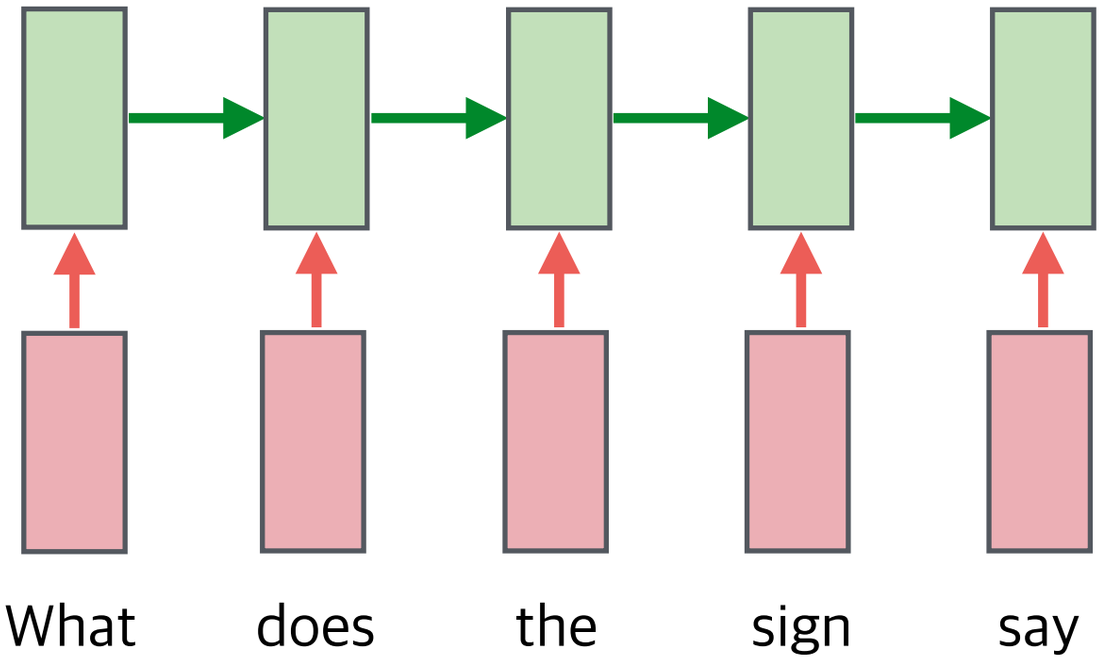


Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

VQA 기본 모델

4096 (VGGNet: fc7)
2048 (ResNet: pool5)





Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

VQA 기본 모델

4096 (VGGNet: fc7)
2048 (ResNet: pool5)

CNN

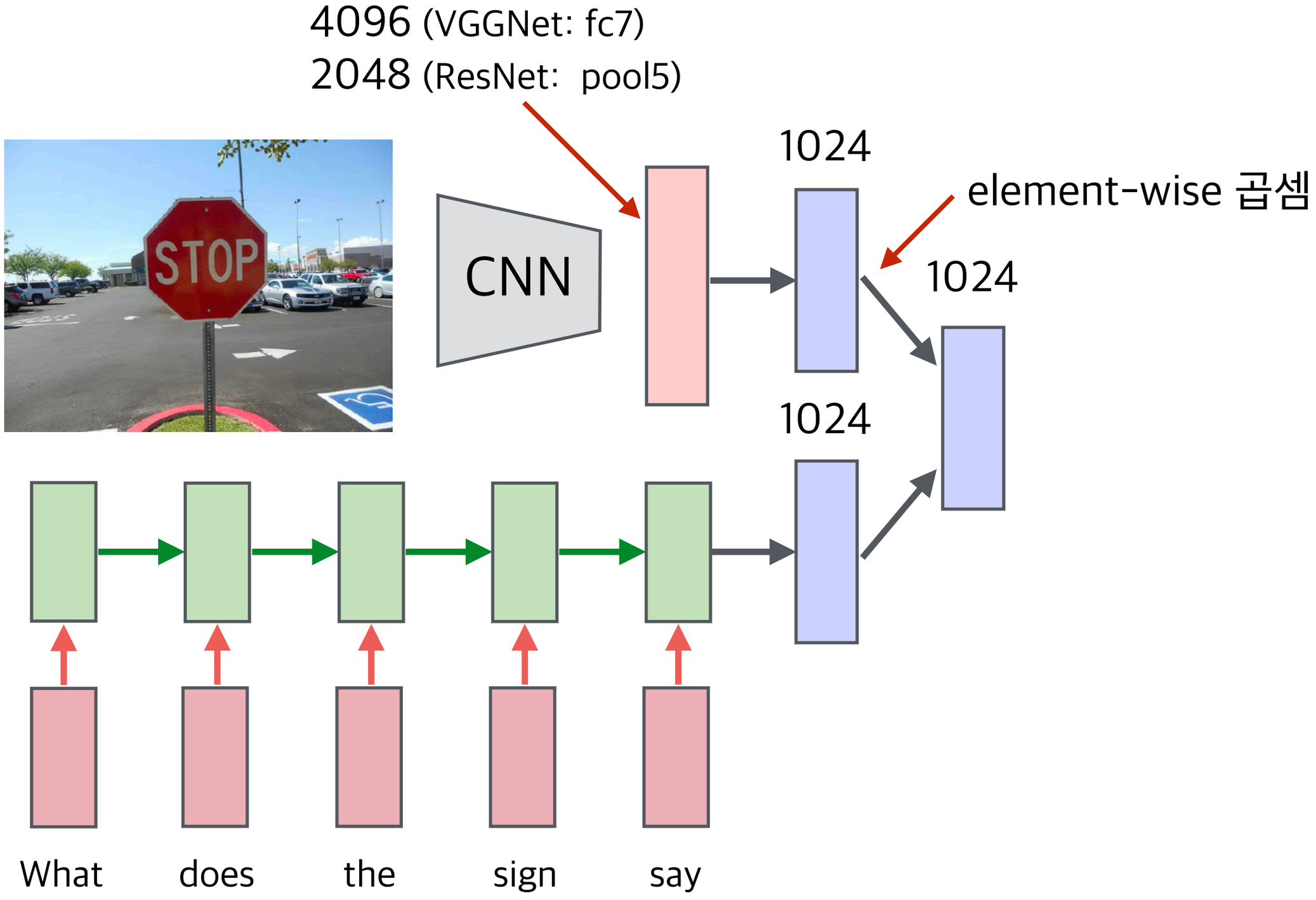
1024

1024

What does the sign say

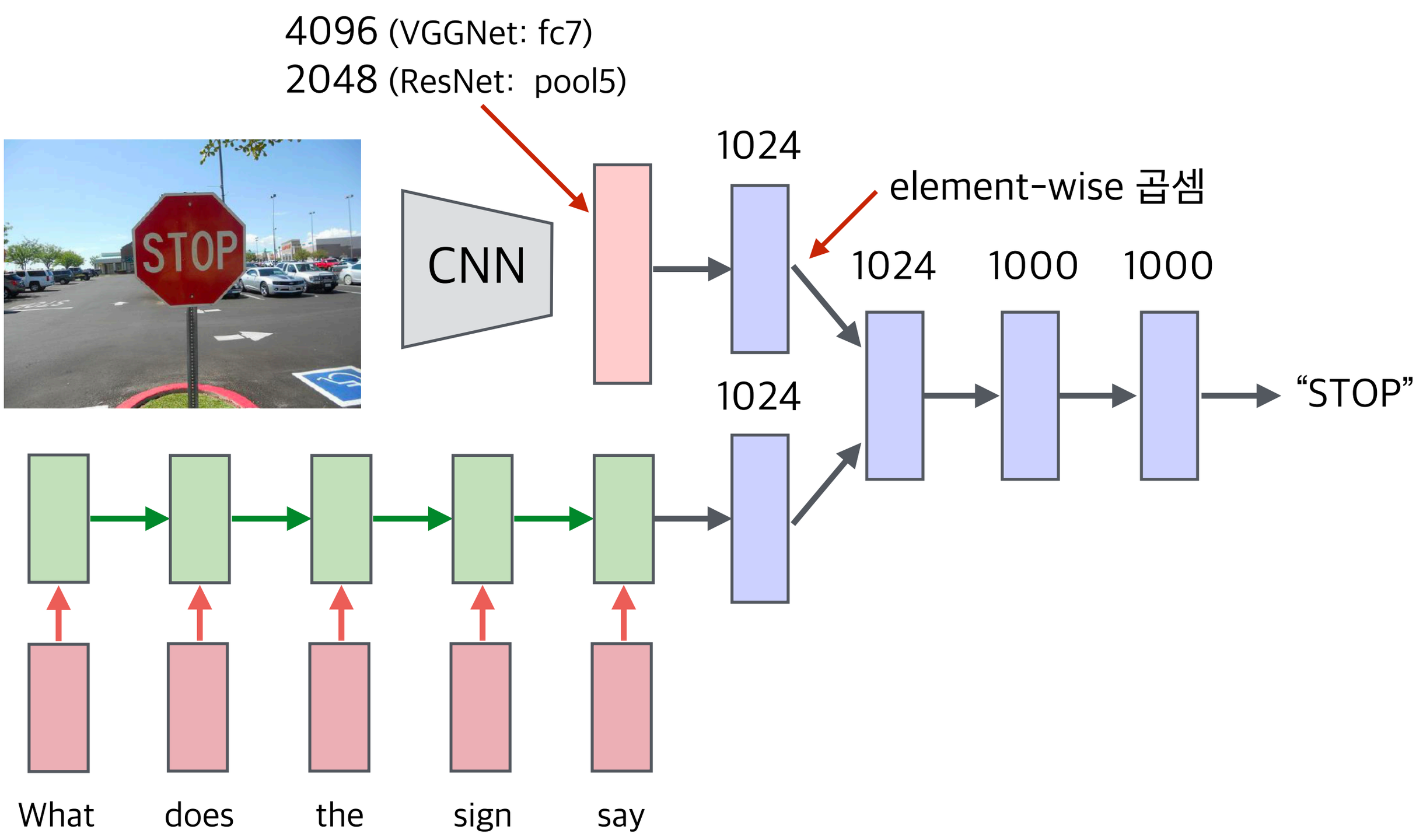
Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

VQA 기본 모델



Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

VQA 기본 모델



Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

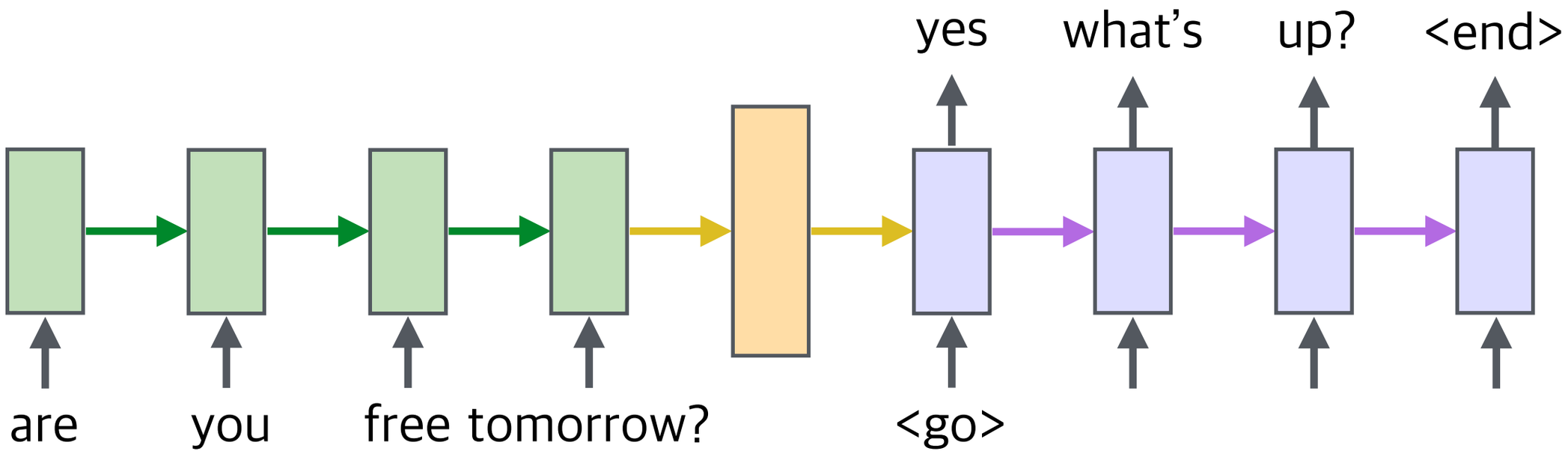
VQA 기본 모델

가장 빈번한 정답 단어 1000개를 추려
1000-클래스의 분류 문제로 변형
(답이 1K 클래스에 속하지 않은 경우
무조건 틀리지만 문제를 쉽게 하기 위해)

Antol, Stanislaw, et al. "Vqa: Visual question answering." Proceedings of the IEEE International Conference on Computer Vision. 2015.

seq2seq

- 기존 RNN 모델과 다른 **인코더-디코더** 모델
  - **인코더**는 입력 문장을 받고 이를 특징 벡터로 임베딩
  - **디코더**는 임베딩된 벡터를 통해 출력 문장을 생성



Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.