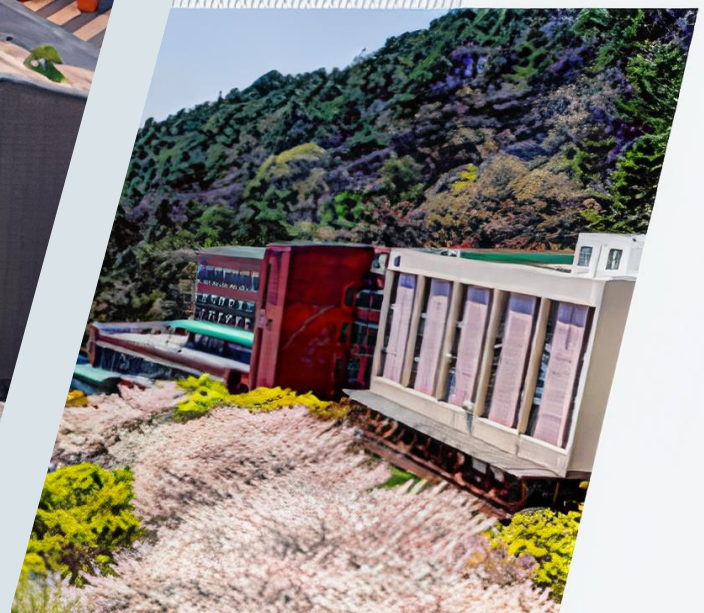
Dong-A Univ. (ISPL)

동아대학교
DONG-A UNIVERSITY

[실습] LLM 기초 – RNN, LSTM

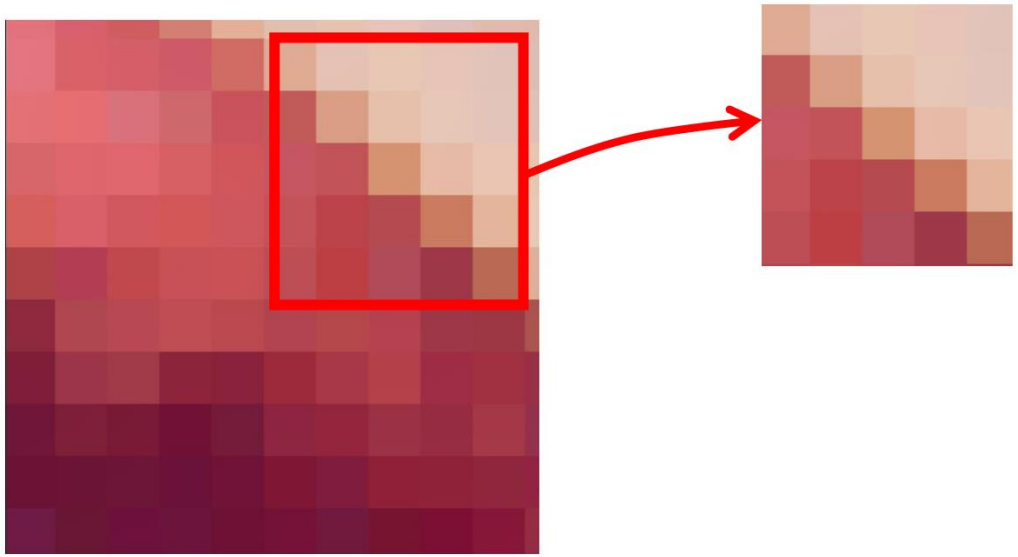
컴퓨터AI공학부

2025년 1학기 머신러닝



Review - RNN

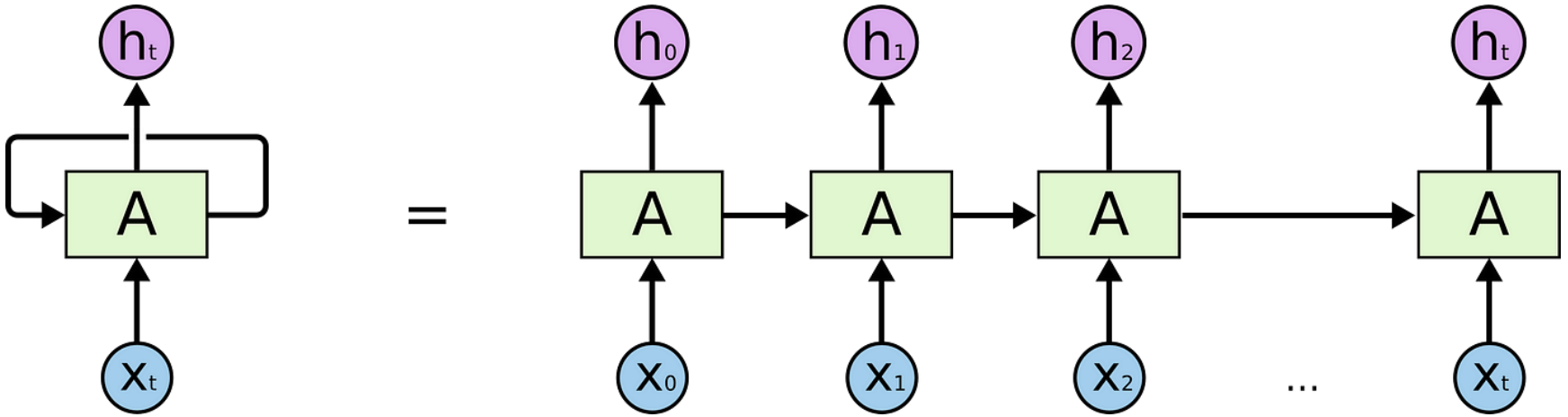
- **Overview**
  - 기존의 CNN은 데이터의 공간적인 정보만을 학습함



CNN은 이미지 데이터의 공간적 특징을 추출하여 학습

Review - RNN

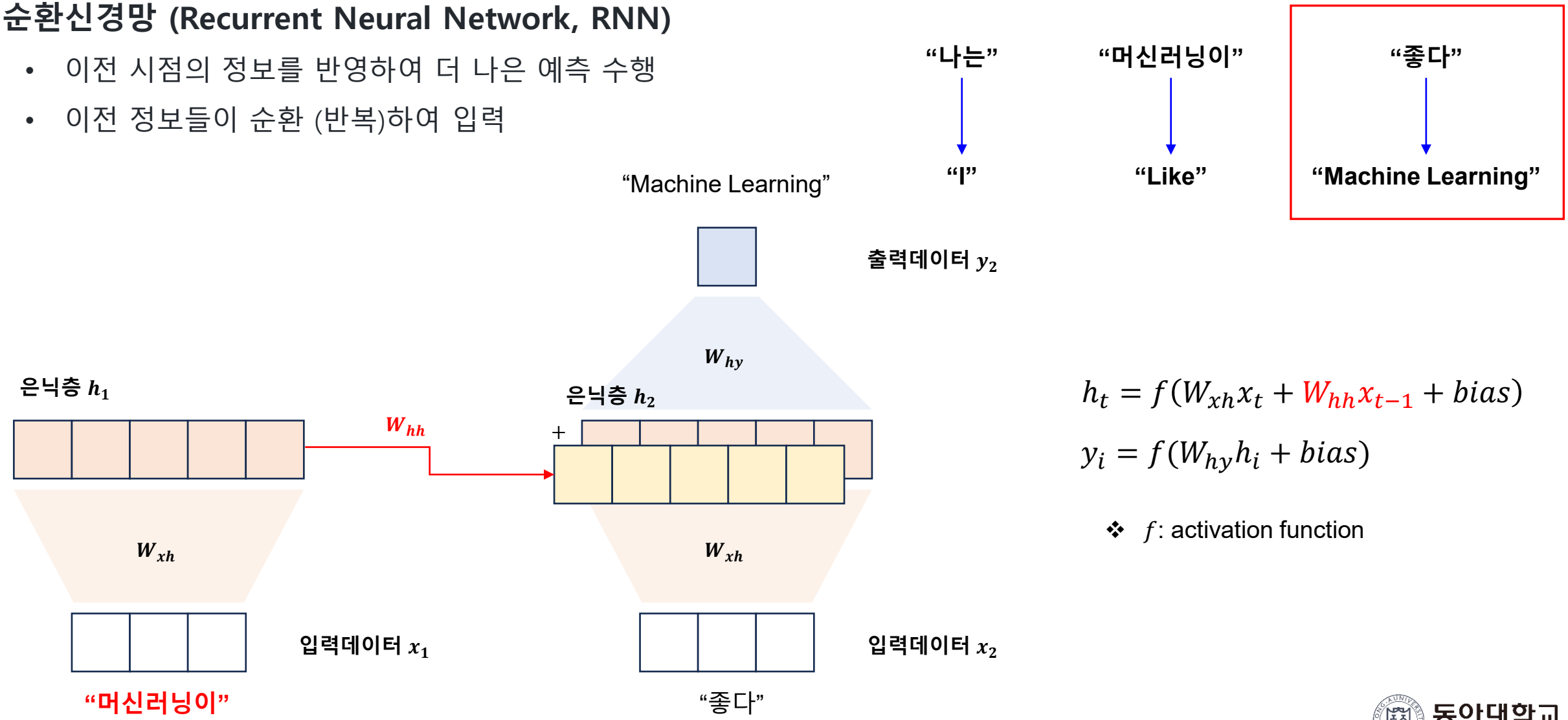
- **Overview**
  - 시간 순서가 있는 데이터 (Time Series Data)를 효율적으로 학습하기 위해 등장
  - 순환 구조를 통해서 이전 상태의 정보를 함께 사용하여 현재 상태의 정보 학습



순서가 있는 데이터를 순차적으로 입력하여 학습

Review - RNN

- **순환신경망 (Recurrent Neural Network, RNN)**
  - 이전 시점의 정보를 반영하여 더 나은 예측 수행
  - 이전 정보들이 순환 (반복)하여 입력



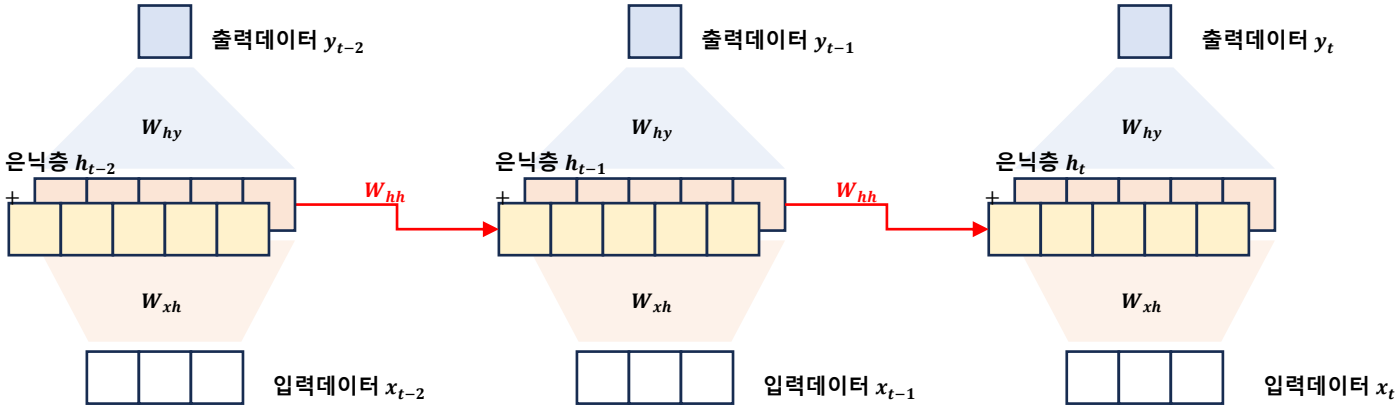
$$h_t = f(W_{xh}x_t + W_{hh}x_{t-1} + bias)$$

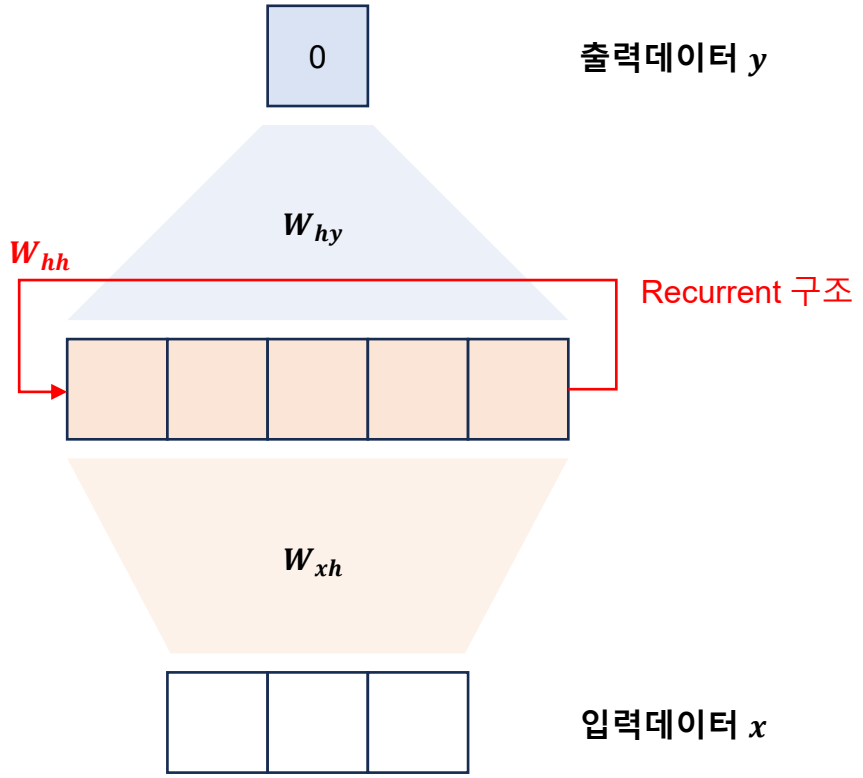
$$y_i = f(W_{hy}h_i + bias)$$

❖ f: activation function

Review - RNN

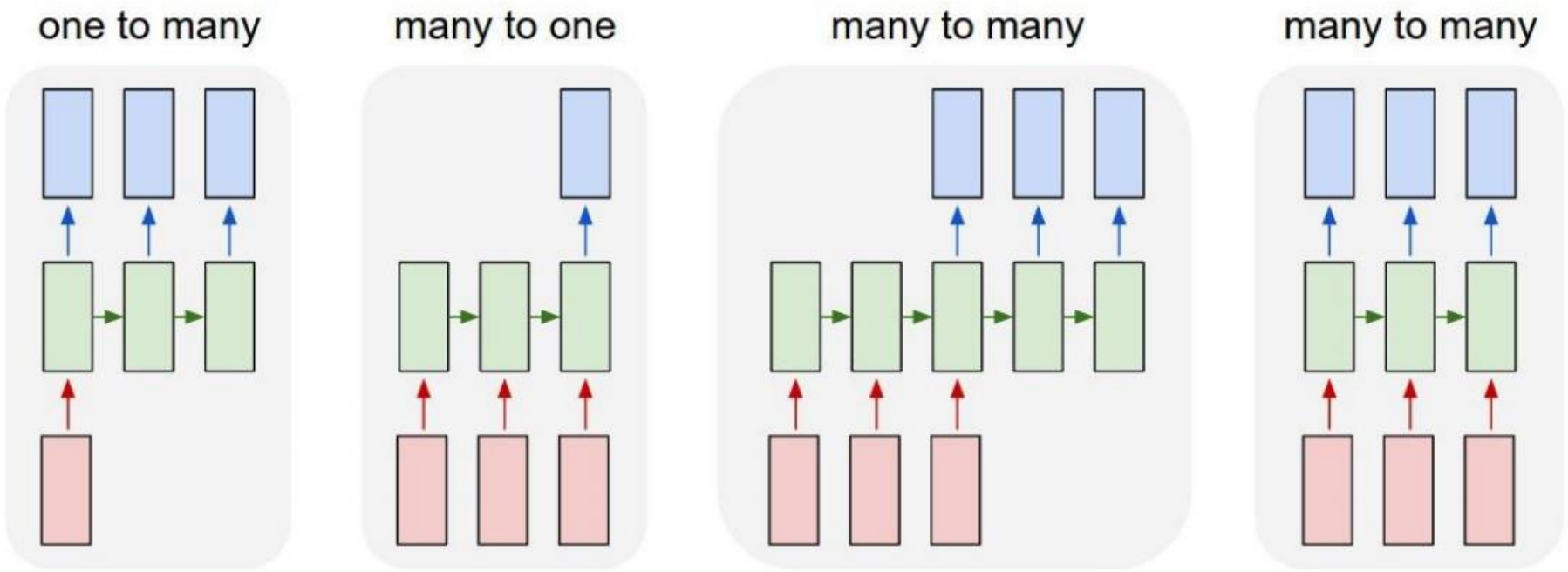
- **순환신경망 (Recurrent Neural Network, RNN)**
  - 이전 시점의 정보를 반영하여 더 나은 예측 수행
  - 이전 정보들이 순환 (반복)하여 입력





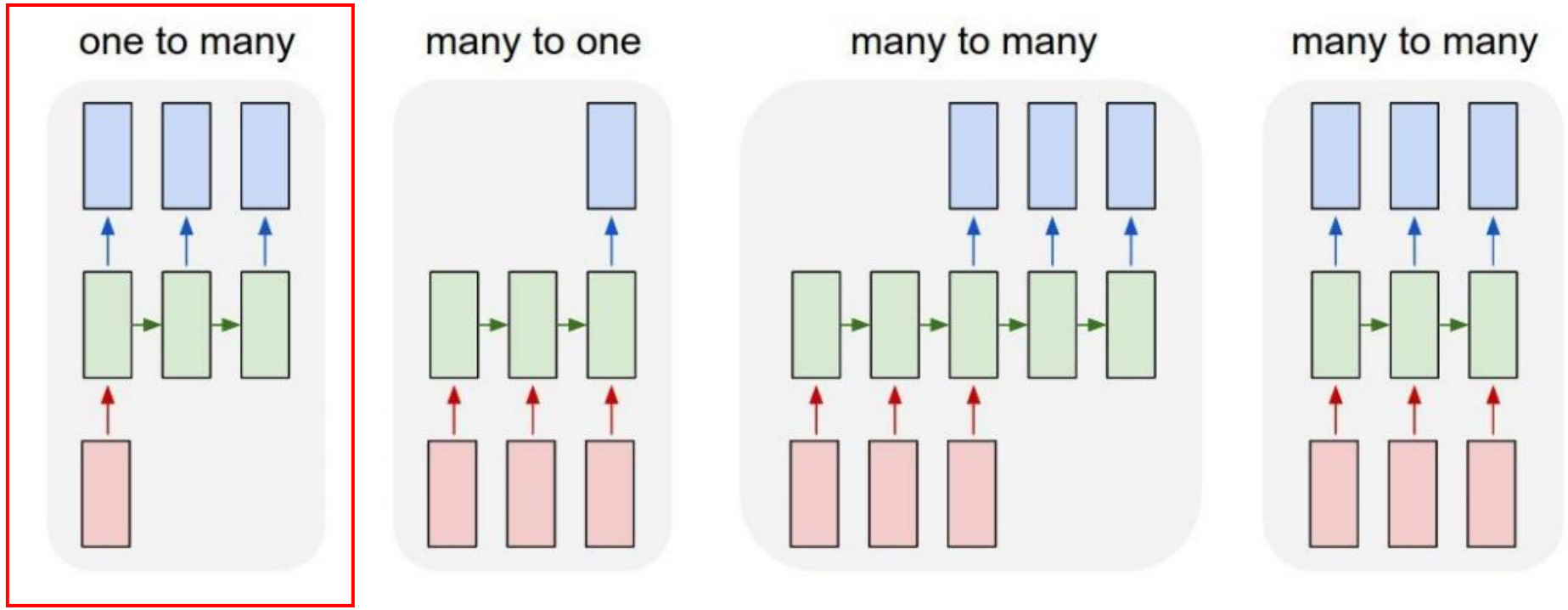
Review - RNN

- **순환신경망 구조의 종류**
  - 순차적인 입력의 길이, 순차적인 예측의 길이에 따라 다음과 같이 구분 가능



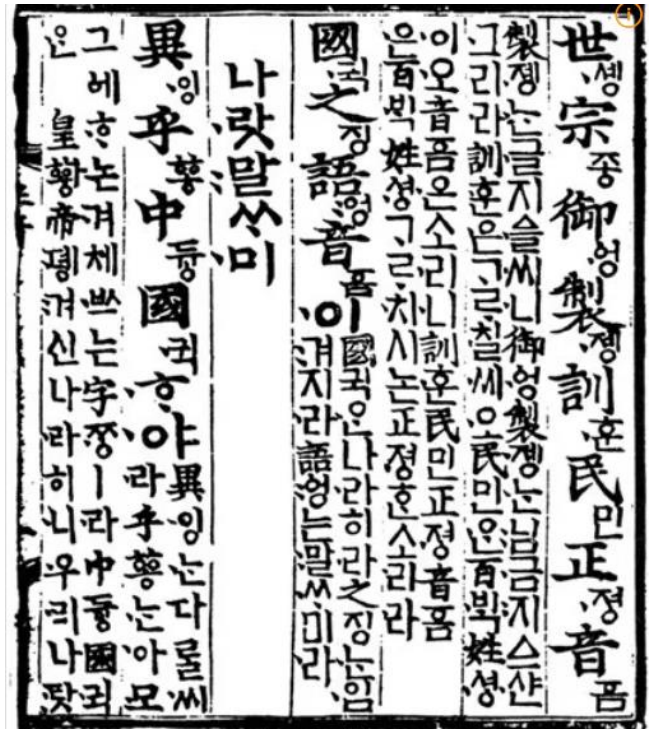
Review - RNN

- **순환신경망 구조의 종류**
  - 순차적인 입력의 길이, 순차적인 예측의 길이에 따라 다음과 같이 구분 가능



[실습] RNN

- **훈민정음 서문 예측 모델**
  - 훈민정음 서문의 현대어 번역본을 예측하는 RNN 모델 구현
  - 첫 단어를 입력하면 훈민정음 서문이 출력되도록 학습

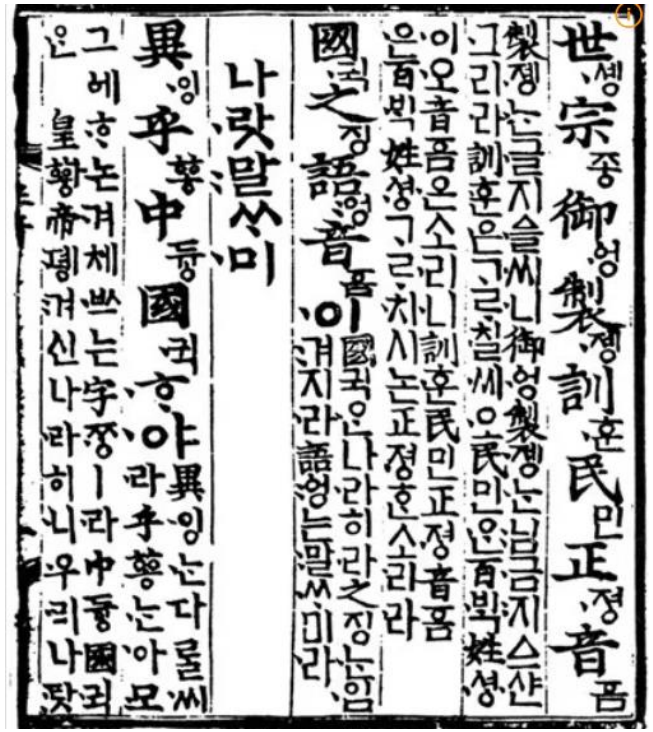


→

나라의 말이 중국과 달라 문자와 서로 통하지 아니하니, 이런 까닭으로 어리석은 백성이 이르고자 할 바가 있어도 마침내 제 뜻을 능히 펴지 못할 사람이 많으니라. 내가 이를 위하여 가엾이 여겨 새로 스물여덟 자를 만드노니 사람마다 하여금 쉬이 익혀 날로 쓰는 데 편하게 하고자 할 따름이니라.

[실습] RNN

- **훈민정음 서문 예측 모델**
  - 훈민정음 서문의 현대어 번역본을 예측하는 RNN 모델 구현
  - 첫 단어를 입력하면 훈민정음 서문이 출력되도록 학습

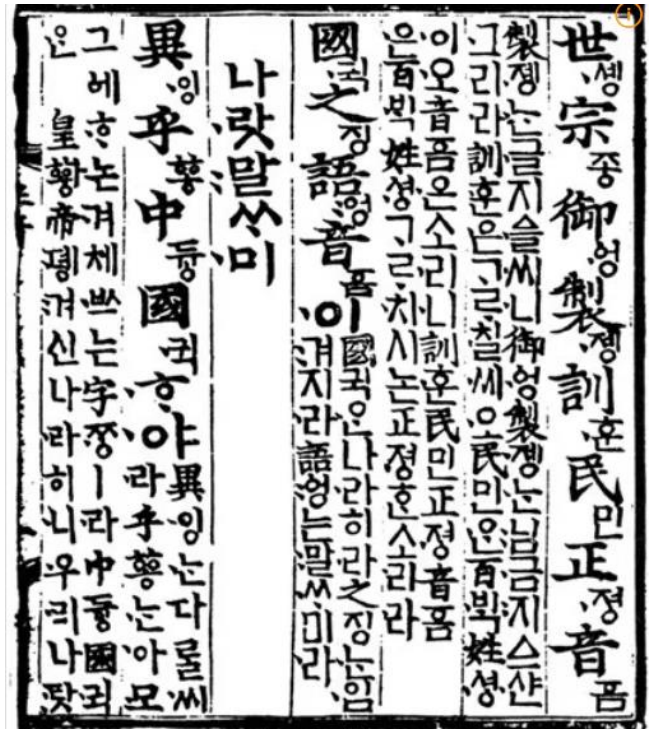


입력 *출력*

나라의 말이 중국과 달라 문자와 서로 통하지 아니하니, 이런 까닭으로 어리석은 백성이 이르고자 할 바가 있어도 마침내 제 뜻을 능히 펴지 못할 사람이 많으니라. 내가 이를 위하여 가엾이 여겨 새로 스물여덟 자를 만드노니 사람마다 하여금 쉬이 익혀 날로 쓰는 데 편하게 하고자 할 따름이니라.

[실습] RNN

- **훈민정음 서문 예측 모델**
  - 훈민정음 서문의 현대어 번역본을 예측하는 RNN 모델 구현
  - 첫 단어를 입력하면 훈민정음 서문이 출력되도록 학습

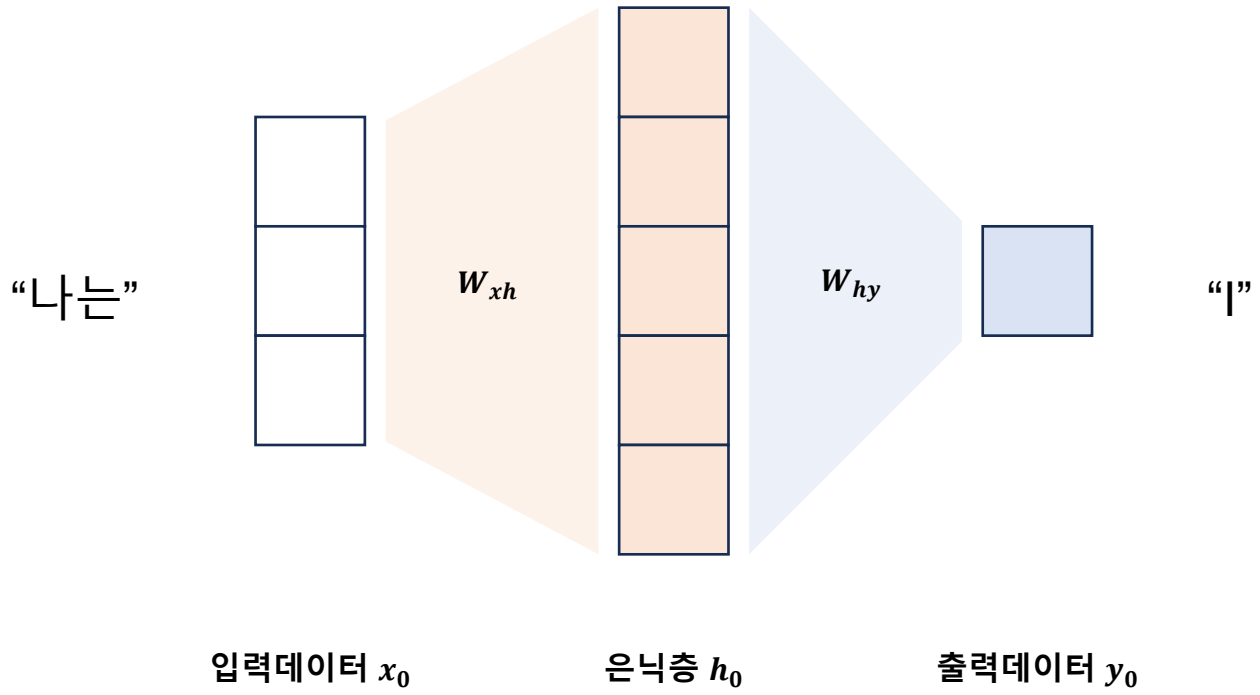


입력 *출력*

나라의 말이 중국과 달라 문자와 서로 통하지 아니하니, 이런 까닭으로 어리석은 백성이 이르고자 할 바가 있어도 마침내 제 뜻을 능히 펴지 못할 사람이 많으니라. 내가 이를 위하여 가엾이 여겨 새로 스물여덟 자를 만드노니 사람마다 하여금 쉬이 익혀 날로 쓰는 데 편하게 하고자 할 따름이니라.

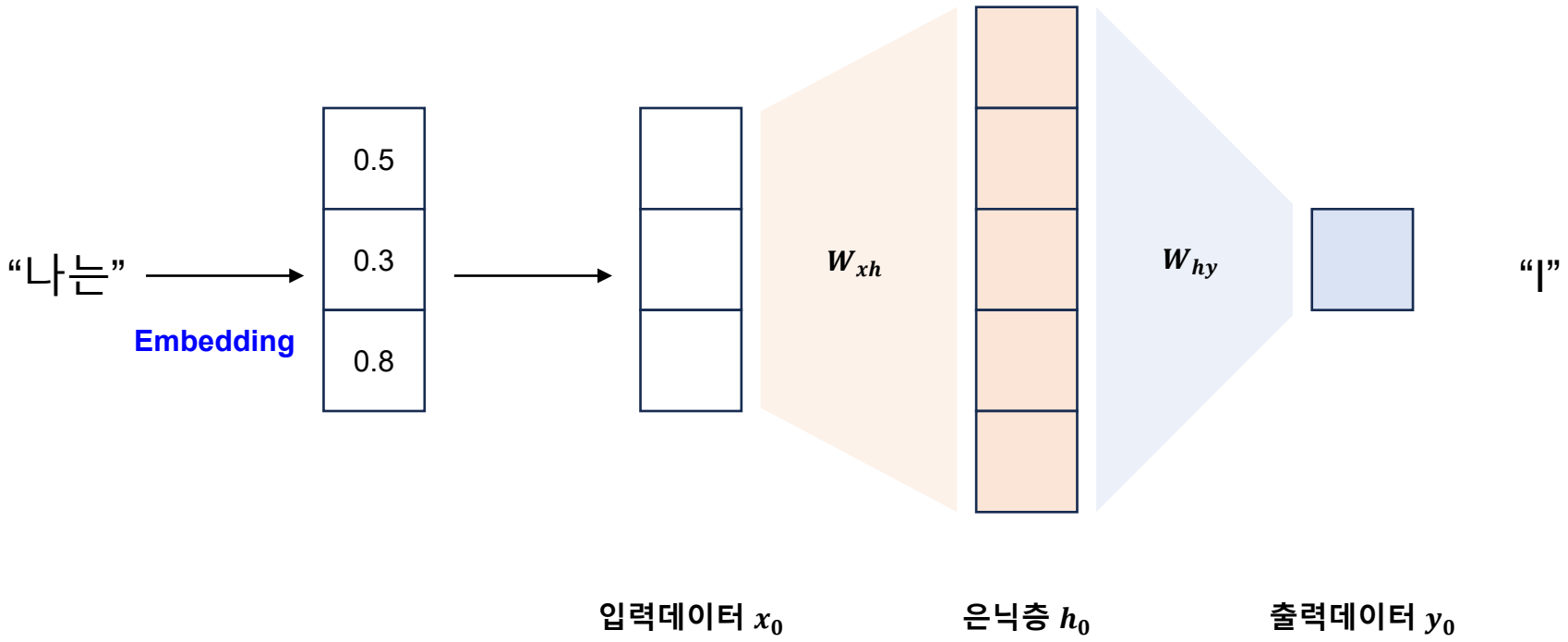
[실습] RNN

- **자연어 데이터의 벡터 변환 (Embedding)**
  - 자연어를 컴퓨터가 이해하지 못하기 때문에 각 단어들을 숫자로 이루어진 벡터로 변환 필요



[실습] RNN

- **자연어 데이터의 벡터 변환 (Embedding)**
  - 자연어를 컴퓨터가 이해하지 못하기 때문에 각 단어들을 숫자로 이루어진 벡터로 변환 필요



[실습] RNN

- **자연어 데이터의 벡터 변환 (Embedding)**
  - One-hot Encoding: 단어의 집합 길이의 벡터 중 하나의 요소만 1이고 나머지는 모두 0인 희소 벡터
  - Word2Vec: 주어진 단어들을 벡터로 변환하는 기계 학습 모델

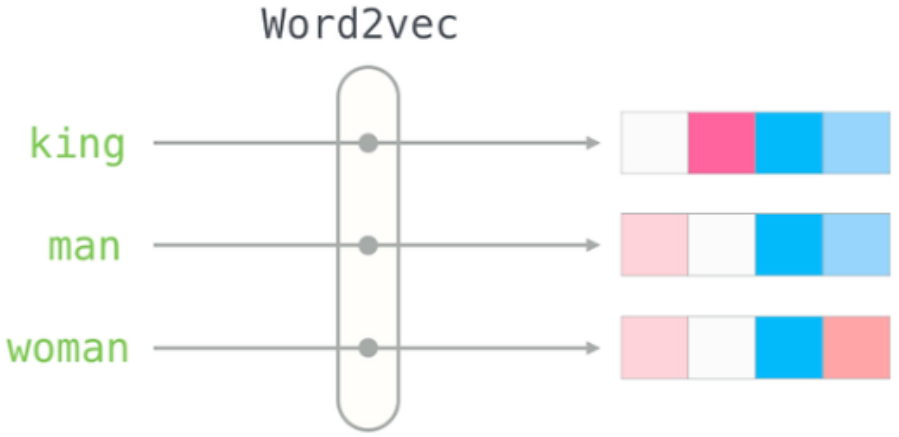
나는 ⟶ [1 0 0 0] x_1

배가 ⟶ [0 1 0 0] x_2

너무 ⟶ [0 0 1 0] x_3

고프다 ⟶ [0 0 0 1] x_4

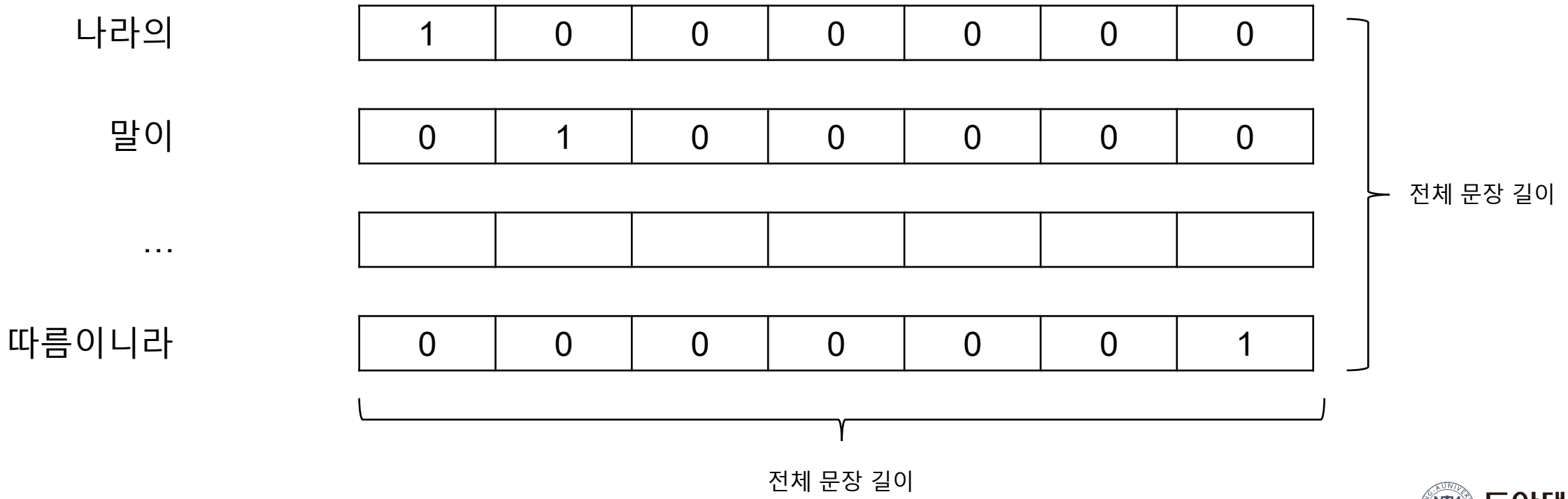
One-hot Encoding



Word2Vec

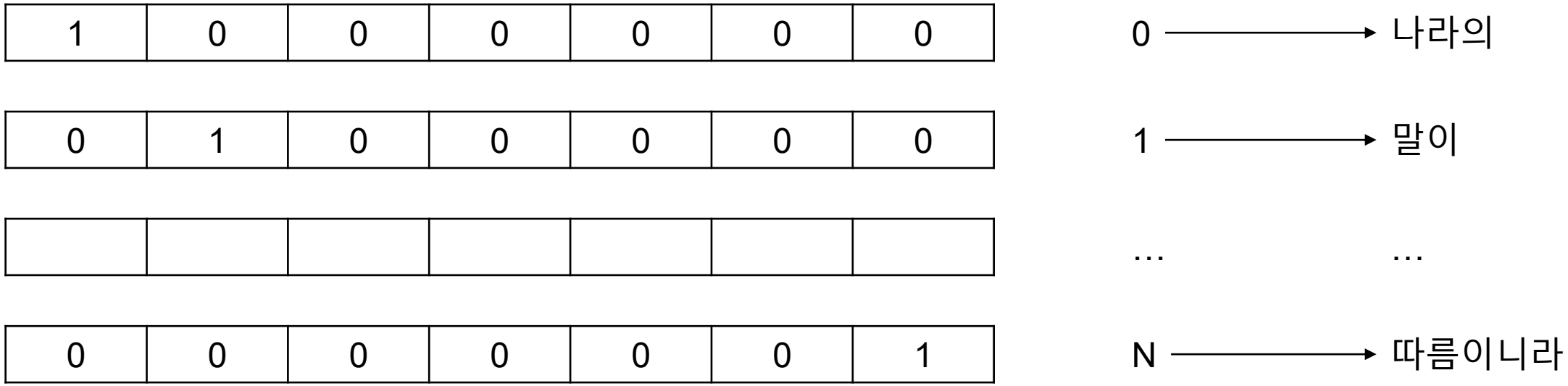
[실습] RNN

- **자연어 데이터의 벡터 변환 (Embedding)**
  - **One-hot Encoding: 단어의 집합 길이의 벡터 중 하나의 요소만 1이고 나머지는 모두 0인 희소 벡터**
  - Word2Vec: 주어진 단어들을 벡터로 변환하는 기계 학습 모델



[실습] RNN

- **자연어 데이터의 벡터 변환 (Embedding)**
  - **One-hot Encoding: 단어의 집합 길이의 벡터 중 하나의 요소만 1이고 나머지는 모두 0인 희소 벡터**
  - Word2Vec: 주어진 단어들을 벡터로 변환하는 기계 학습 모델



[실습] RNN

- 자연어 데이터의 벡터 변환 (Embedding)

```python
def embedding(data):
    data = re.sub('[^가-힣]', ' ', data)
    tokens = data.split()
    vocab = list(set(tokens))
    vocab_size = len(vocab)

    word_2_idx = {word: i for i, word in enumerate(vocab)}
    idx_2_word = {i: word for i, word in enumerate(vocab)}

    vector = []
    for i in range(vocab_size):
        word_vector = one_hot(vocab[i], word_2_idx, vocab_size)
        vector.append(word_vector)

    vector = torch.cat(vector, dim=0)
    vector = vector.unsqueeze(0)

    return vector, vocab_size, word_2_idx, idx_2_word
```

데이터의 한글 외 문자 제거 및 리스트로 저장

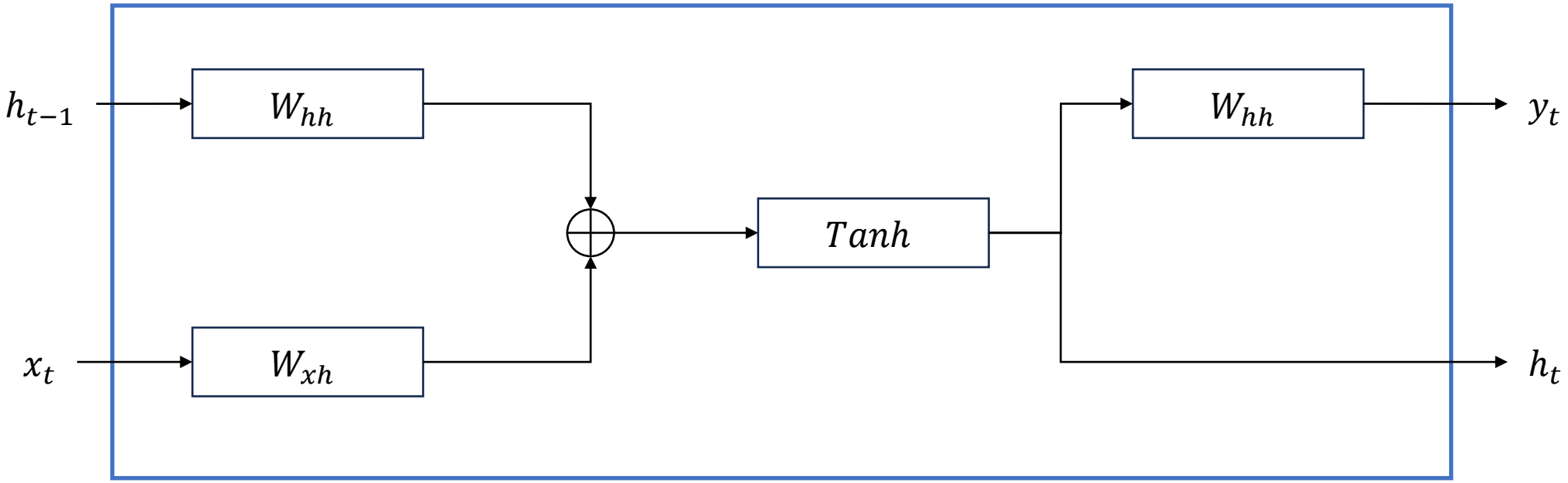
데이터와 인덱스 간의 관계 정의

One-hot encoding

```python
def one_hot(word, word_2_idx, vocab_size):
    vector = torch.zeros(1, vocab_size)
    index = word_2_idx[word]
    vector[0, index] = 1

    return vector
```

[실습] RNN

- **Network architecture**



RNN

[실습] RNN

- **Network architecture**

```
1 class RNN(nn.Module):
2     def __init__(self, embed_dim, hidden_dim):
3         super(RNN, self).__init__()
4
5
6     def feed_forward(self, x, h):
7
8         return out, hidden
9
10
11    def forward(self, x, sequence_len):
12
13        return outputs
14
```

각 MLP layer 정의

입력 값과 t-1 단계의 Hidden states 를 이용하여 출력 및 t 단계의 hidden states 출력

반복 구조를 통한 전체 sequence 예측

[실습] RNN

- **Training process**

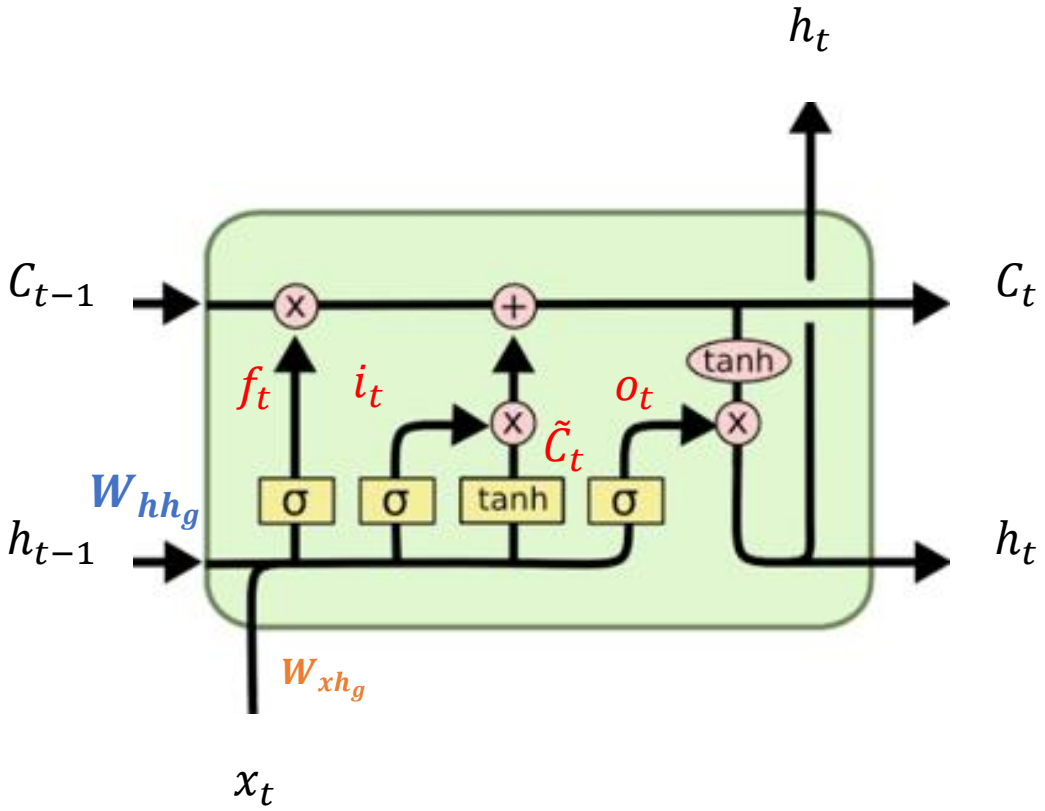
```
network.train()
network = network.to('cuda:0')


inputs = '나라의'
inputs = one_hot(inputs, word_2_idx, sequence_len)
inputs = inputs.unsqueeze(0)
inputs = inputs.to('cuda:0')
vector = vector.to('cuda:0')

for epoch in range(training_epoch):
    avg_cost = 0

    pred = network(inputs, sequence_len)

    # loss 계산
    loss = loss_function(pred, vector)

    # Backpropagation
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    avg_cost += loss

    print('Epoch: %d Loss = %f'%(epoch+1, avg_cost))
print('Learning finished')
```

초기 입력 값 정의

예측한 전체 sequence와 정답 데이터의 loss 계산

[실습] LSTM

- **Network architecture**



LSTM

$$f_t = \sigma(W_{xhf}x_t + W_{hhf}h_{t-1} + bias)$$

$$i_t = \sigma(W_{xhi}x_t + W_{hhi}h_{t-1} + bias)$$

$$\tilde{C}_t = tanh(W_{xhc}x_t + W_{hhc}h_{t-1} + bias)$$

$$o_t = \sigma(W_{xho}x_t + W_{hho}h_{t-1} + bias)$$

Questions & Answers

Dongsan Jun (dsjun@dau.ac.kr)
Image Signal Processing Laboratory (www.donga-ispl.kr)
Division of Computer·AI Engineering
Dong-A University, Busan, Rep. of Korea