

Deep Learning based Text Processing

Lec 09: Introduction to Recurrent Neural Network



hsyi@kisti.re.kr

Hongsuk Yi (이홍석)



- ❖ Name: Hongsuk Yi

- ❖ affiliation

- ✓ Principal Researcher, Dept. of Data-centric Problem Solving Research, KISTI
- ✓ Professor, Dept. of Applied AI, UST

- ❖ Research details

- ✓ Traffic congestion prediction in urban areas based on deep learning (2018-2021)
- ✓ Research on Intelligent Infrastructure Technology (2018~2019)
- ✓ Study on the Transformer Model for Traffic Prediction in Urban Areas (2022-2025)

- ❖ Computing technology

- ✓ GPU-based accelerated computing technology: CUDA, OpenCL
- ✓ Supercomputing Parallel Processing Technology: MPI, OpenMP, ManyCore Computing
- ✓ Deep Learning Technology: Tensorflow, PyTorch, Keras, Theano

Course Materials : github.com/hongsuk.yi

Information : sites.google.com/view/deepai/home

publication :<https://www.researchgate.net/profile/Hongsuk-Yi/publications>



The screenshot shows a personal website for Hongsuk Yi. The header features a dark blue background with a geometric pattern. On the left is a sidebar with a black background and white text, listing 'DeepAI', 'Home', 'Research', 'Lecture', 'Publications', and 'Contacts Us'. The main content area has a light gray background. At the top right is a photo of a man in a suit, identified as '이 흥 석 Hongsuk Yi' and 'Korea Institute of Science and Technology Information'. Below this is a dark blue section titled 'Work Experiences' containing a list of professional positions. Another dark blue section below is titled 'Research Interests' with a list of research topics.

- Principal Researcher, Korea Institute of Science and Technology Information (KISTI)/Supercomputing Center(1999.12~present)
- Adjunct Professor, University of Science and Technology (UST) (2020.2~present)
- Adjunct Professor, Korea University (2009.8~2013.12)

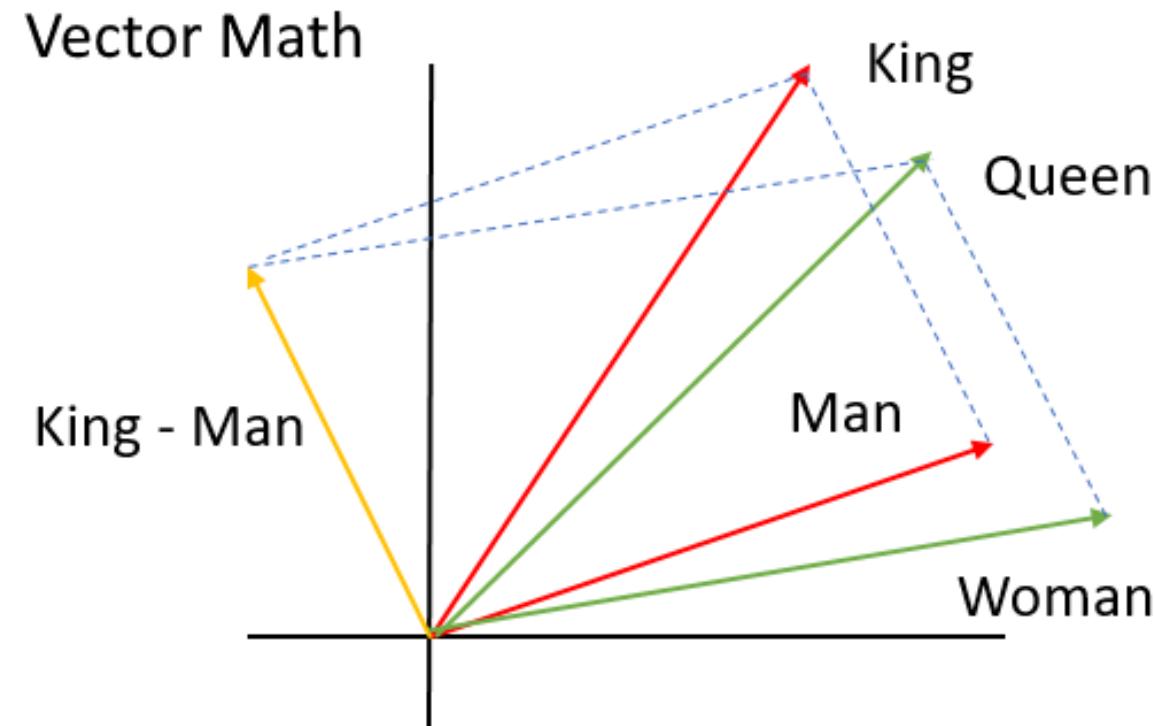
- TensorFlow 2.0 딥러닝 알고리즘 성능 최적화
- 라지스케일 GPU 분산 병렬 컴퓨팅
- CCTV 영상에서 차량/사람/사물을 탐색, 추적, 재인식 기술

Reviewing the last time!

Overview of Course

❖ Basic knowledge of NLP

- ✓ Word Embedding Theory : Word2Vec

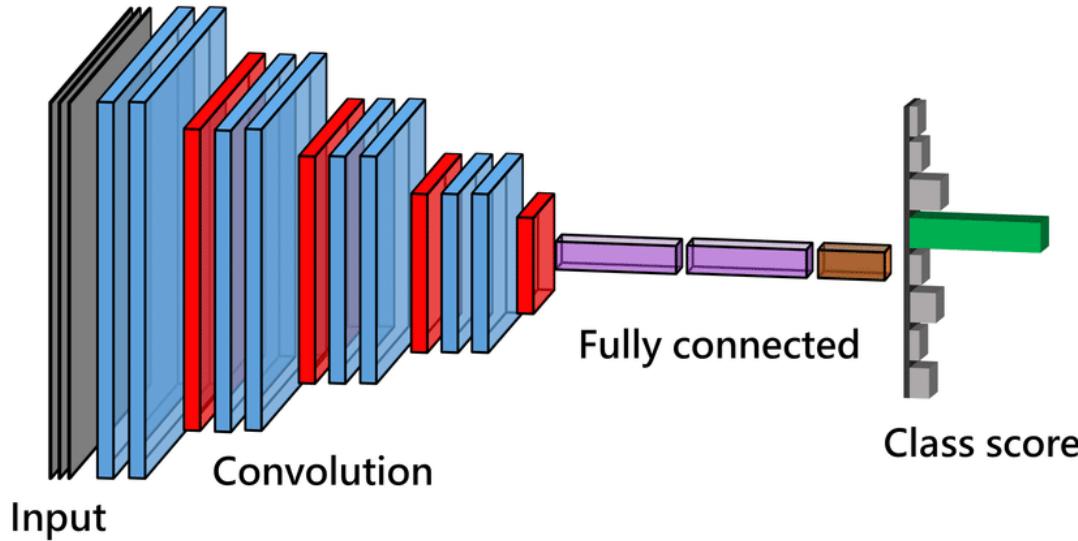


Overview of Course (II)

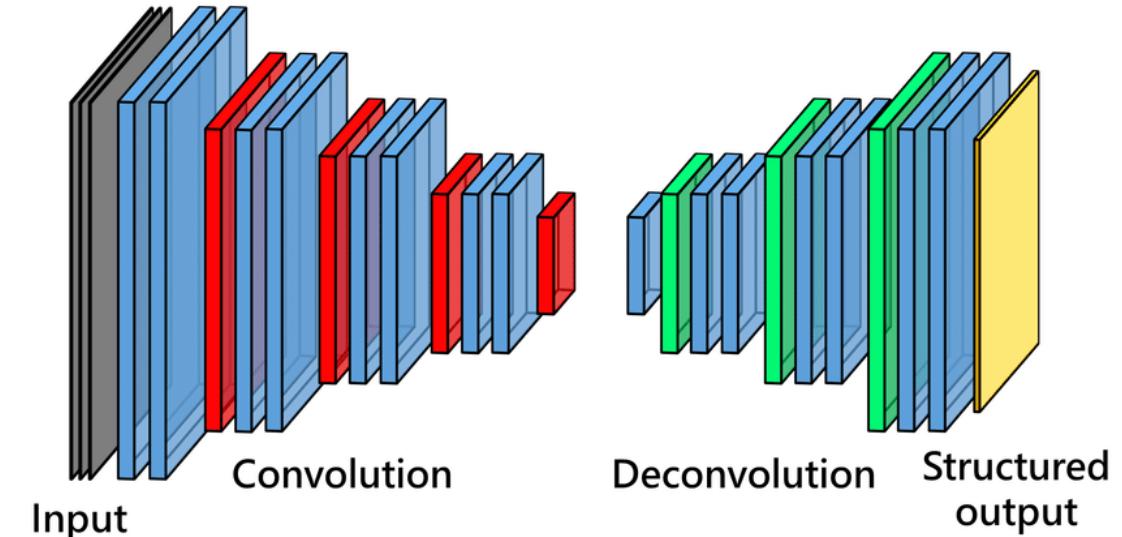
❖ Text Analysis Practice with FCN

- ✓ The concept of machine learning and deep learning

Convolutional / fully connected network



Fully convolutional network

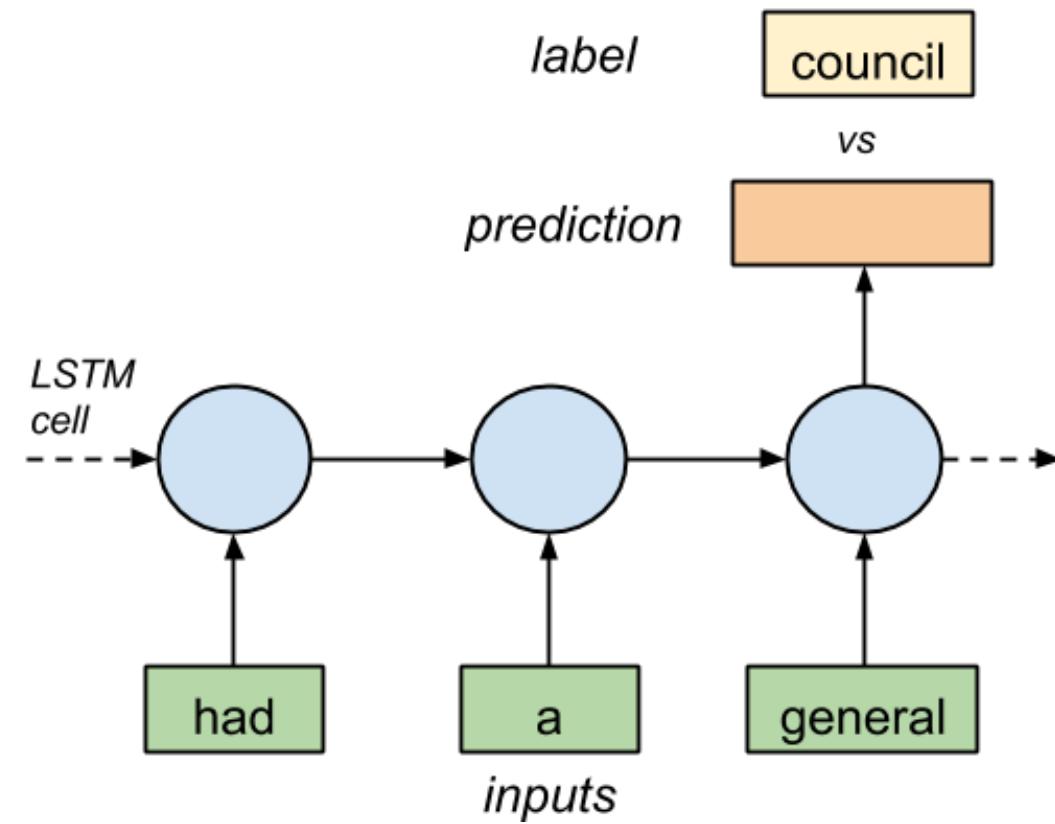


 Conv + BatchNorm + ReLU  Max pooling  Upsampling

Sequence Modeling: Design Criteria

- ❖ Design criteria to model sequences, we need:

- ✓ RNN meet these sequence modeling design critiera



❖ Introduction to Recurrent Neural Network

- ✓ Simple RNN, BPTT, Memory Cell
- ✓ Code: Implementing an RNN with Keras

❖ Introduction to Long-Short Term Memroy

- ✓ Cell state, LSTM, and GRU, and Applications
- ✓ A Visual Guide to Recurrent Layers in Keras
- ✓ Code: A simple LSTM layers

❖ Text generation with RNN

- ✓ Tokenizer, Character-Level Language model
- ✓ Code: Alice's Adventures in Wonderland

❖ Sequence to Sequence model with RNN

- ✓ Introduction to Seq2Seq and Attention model
- ✓ Code: Character-Level Neural Machine Translation

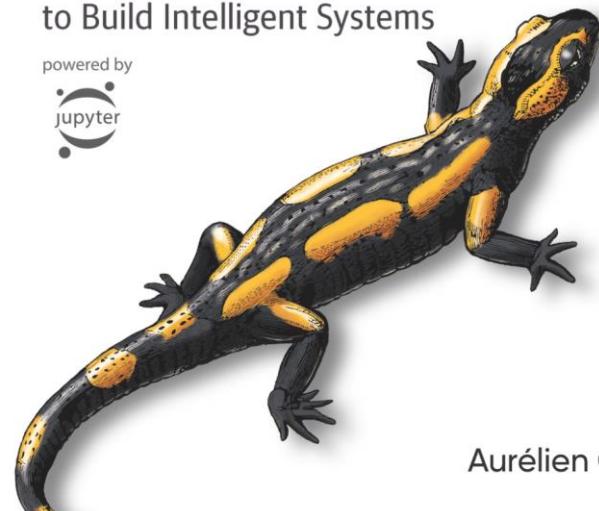
Introduction to Recurrent Neural Network

O'REILLY®

Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow

Concepts, Tools, and Techniques
to Build Intelligent Systems

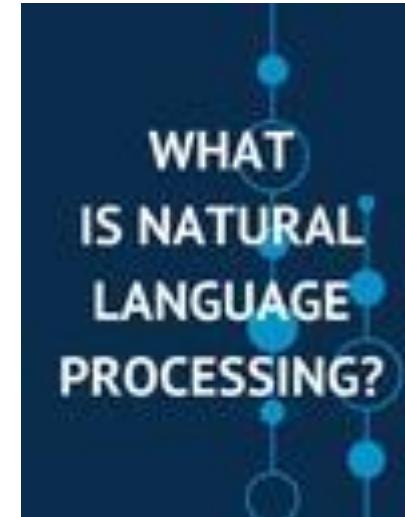
powered by



Aurélien Géron

2nd Edition
Updated for
TensorFlow 2

딥 러닝을 이용한 자연어 처리 입문
<https://wikidocs.net/book/2155>

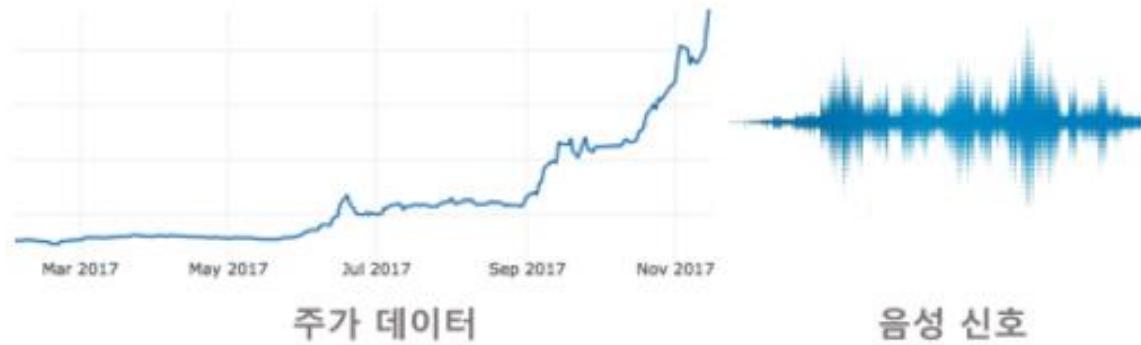


What is a time series problem?

- ❖ Time series deals with data over time: Predicting
 - ✓ weather (short term forecasts), climate (long term forecasts)

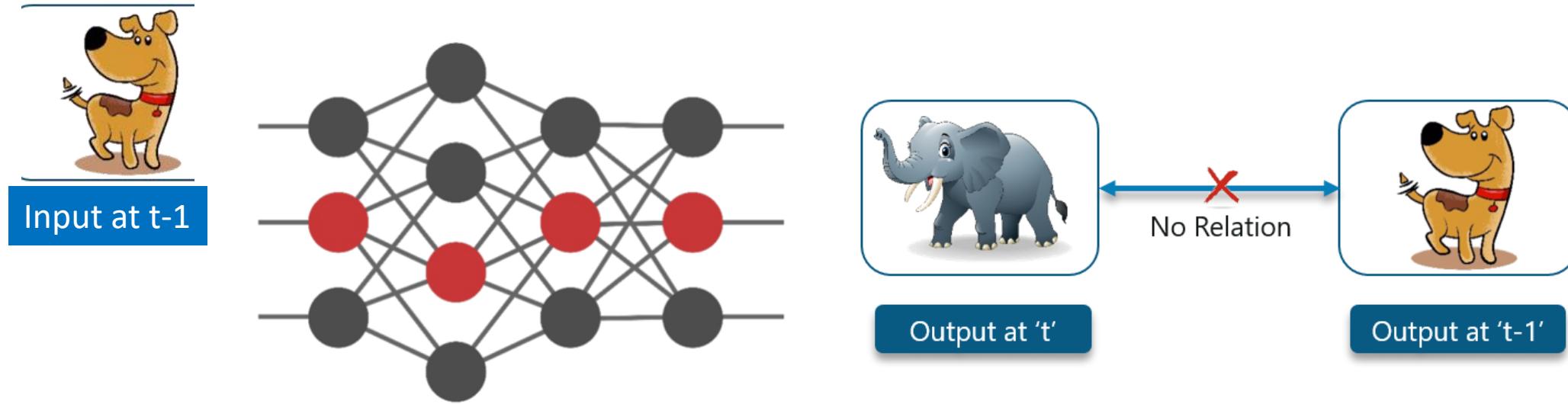
This sentence is a sequence of words...

↑
 $t = 1$ ↑
 $t = 2$ ↑
 $t = 3$...



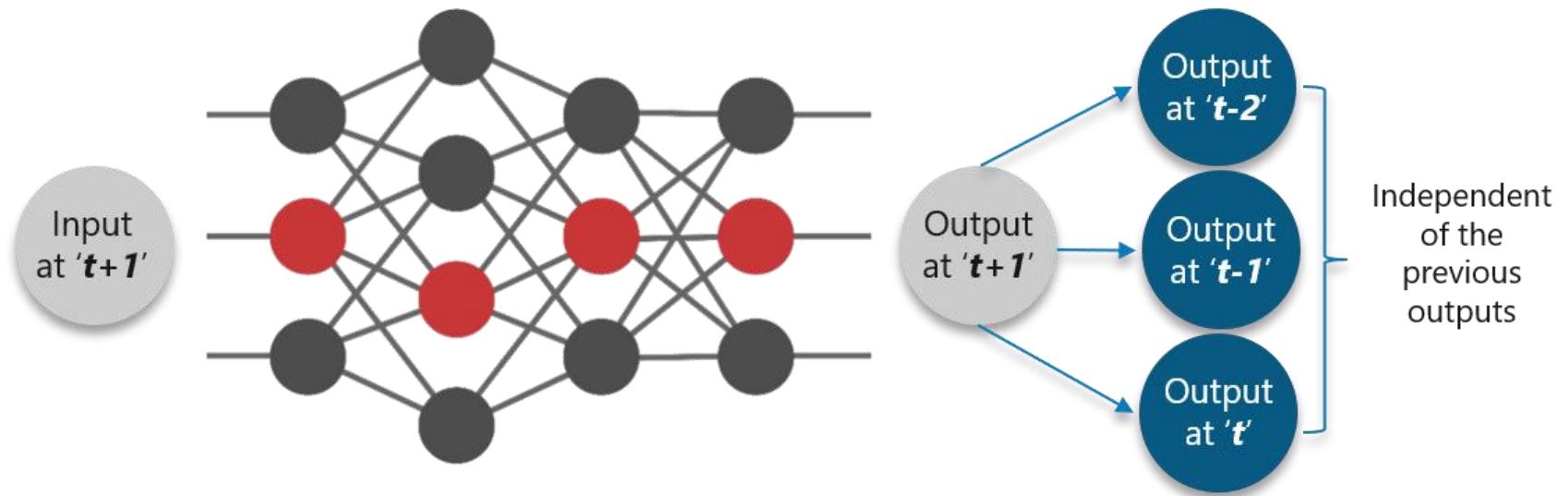
Why Not Feedforward Networks

In the existing FeedForWared Networks,



Why Not Feedforward Networks

- ❖ If you are reading a book, you should be well aware of the previous page.

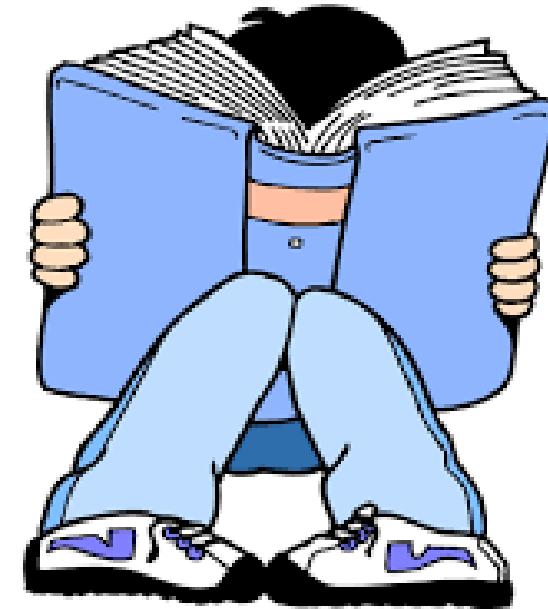


No order when reading a book

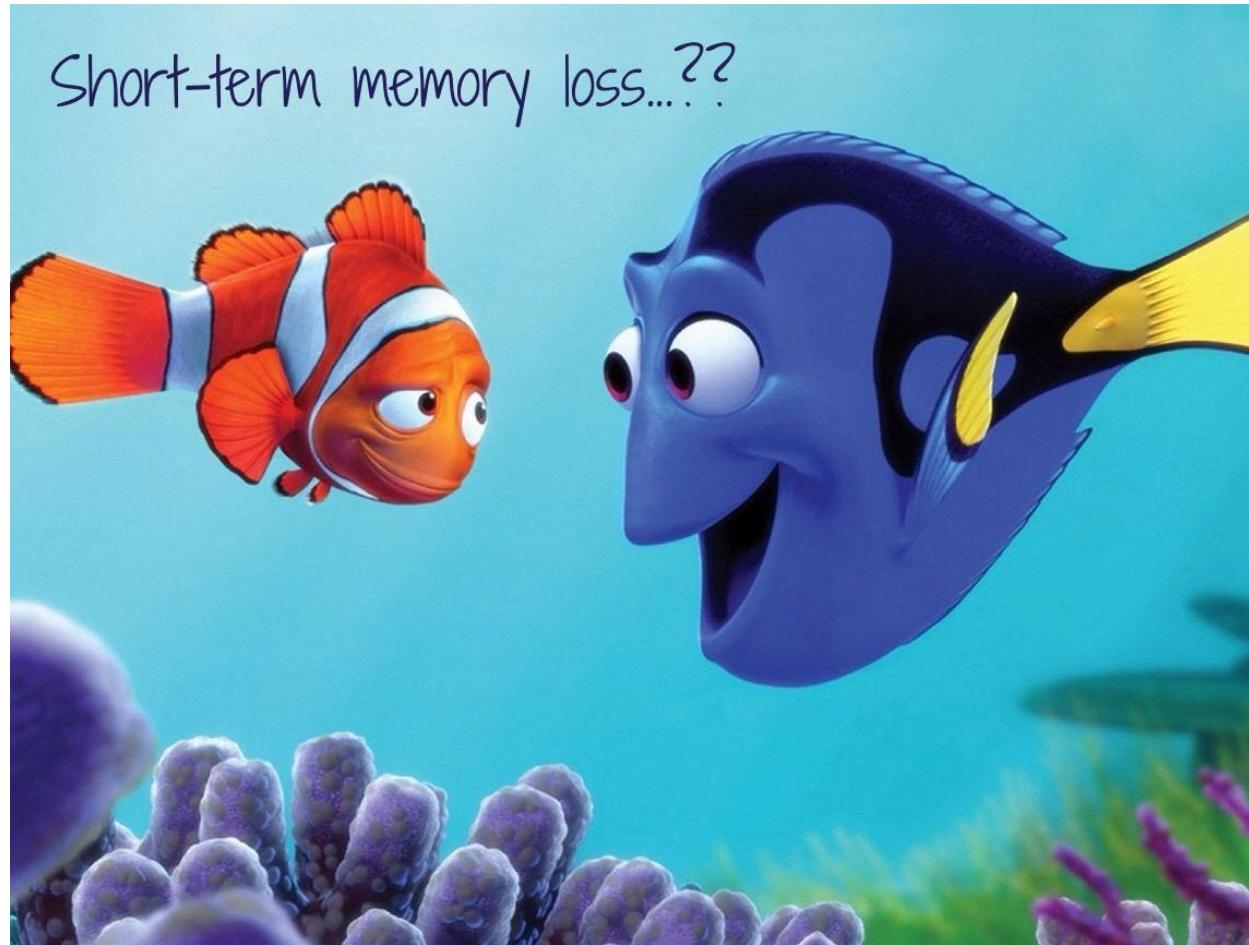
- ❖ a 10 page picture book



- ❖ How much content do you remember?
 - ✓ A sime example: History Book
- ❖ What is the memory when reading a book
 - ✓ Sequence Data Pattern?
 - ✓ Memory Cell



❖ Short-term and Long Term Memory Loss



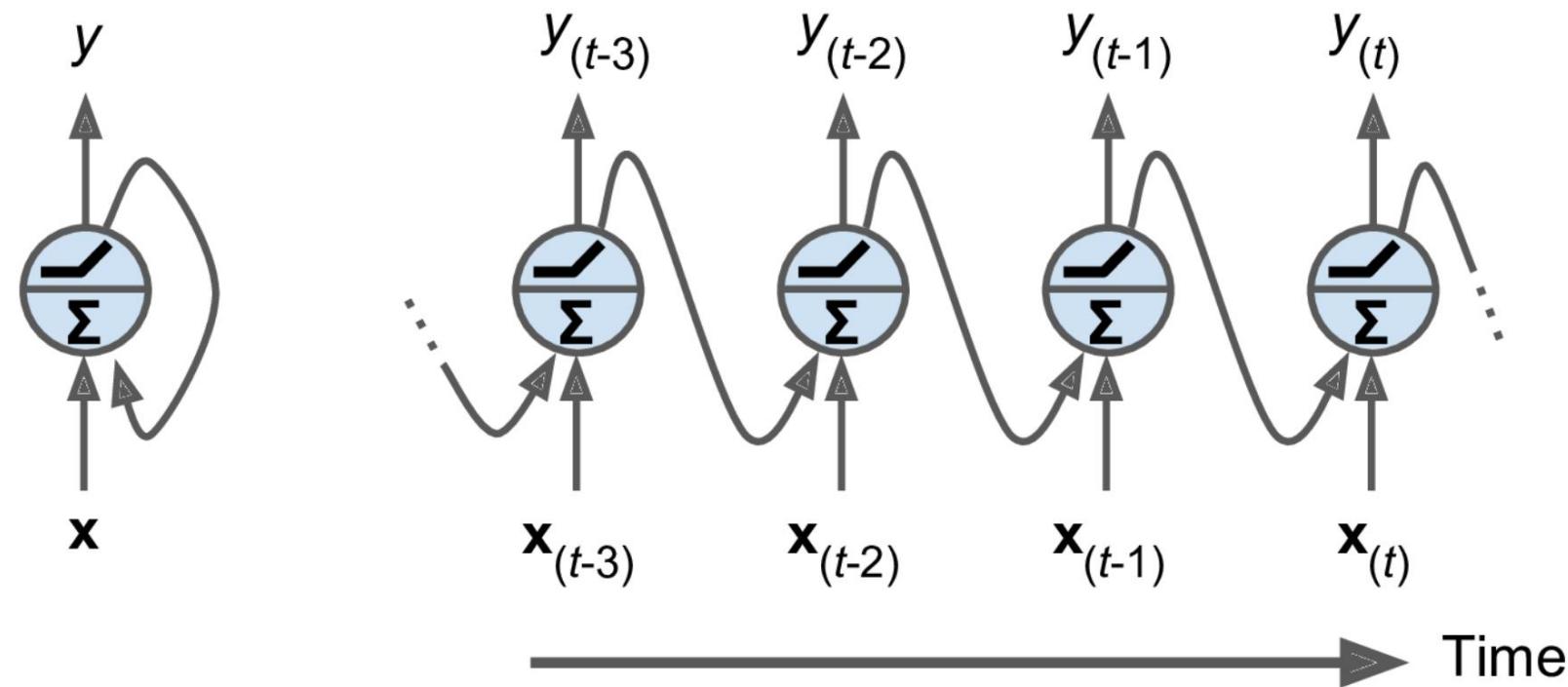
❖ Attention : No order?



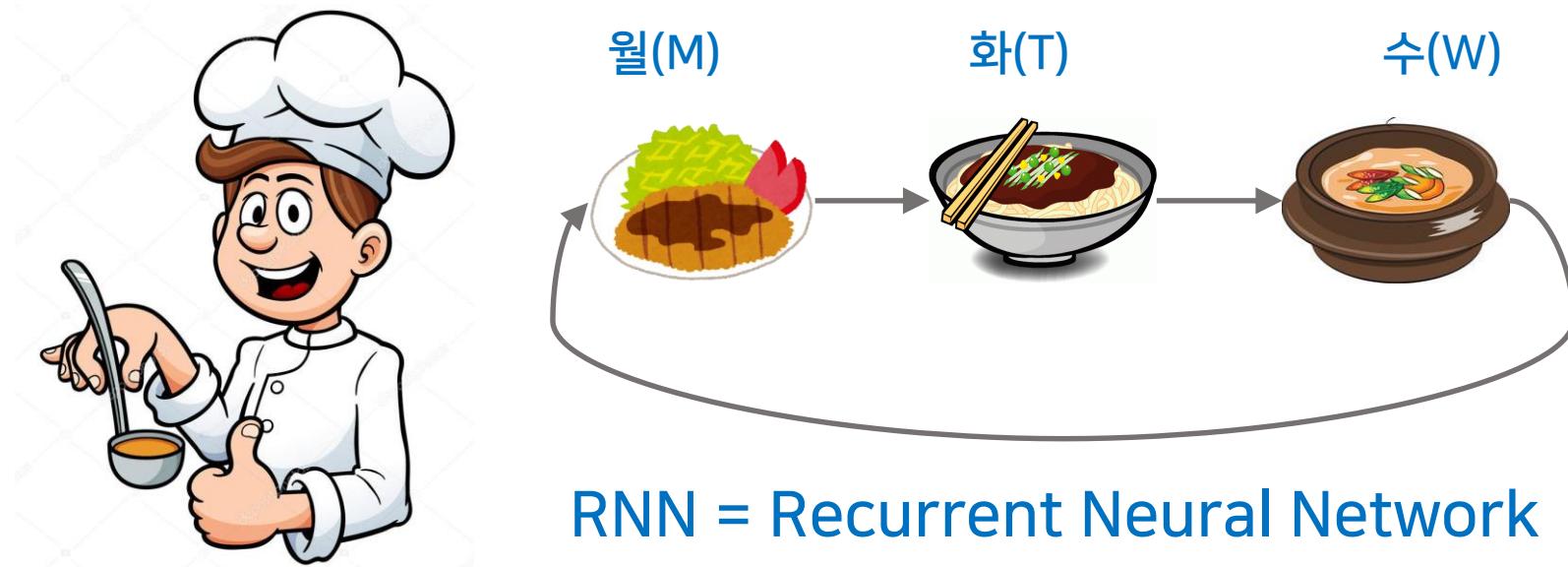
Recurrent Neurons and Memory Cells

❖ Memory Cells ~ a form of memory

- ✓ A RNN looks like a feedforward NN, except it also has connections pointing backward.



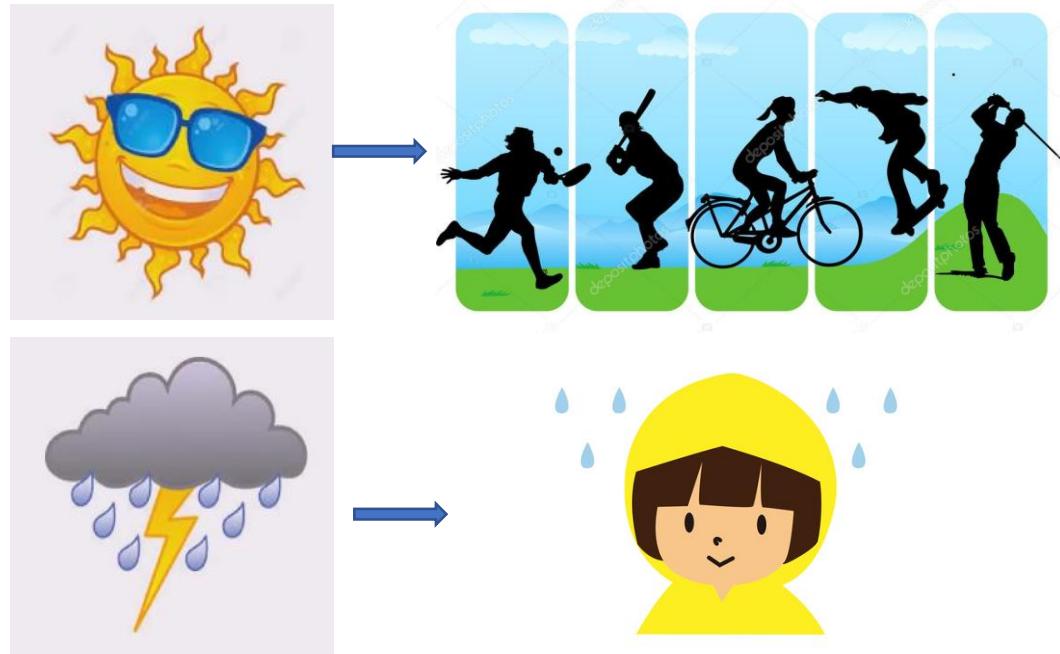
- ❖ There is a pattern in dinner.
 - ✓ Only three types of meals are available.



A simple of RNN (2/3)

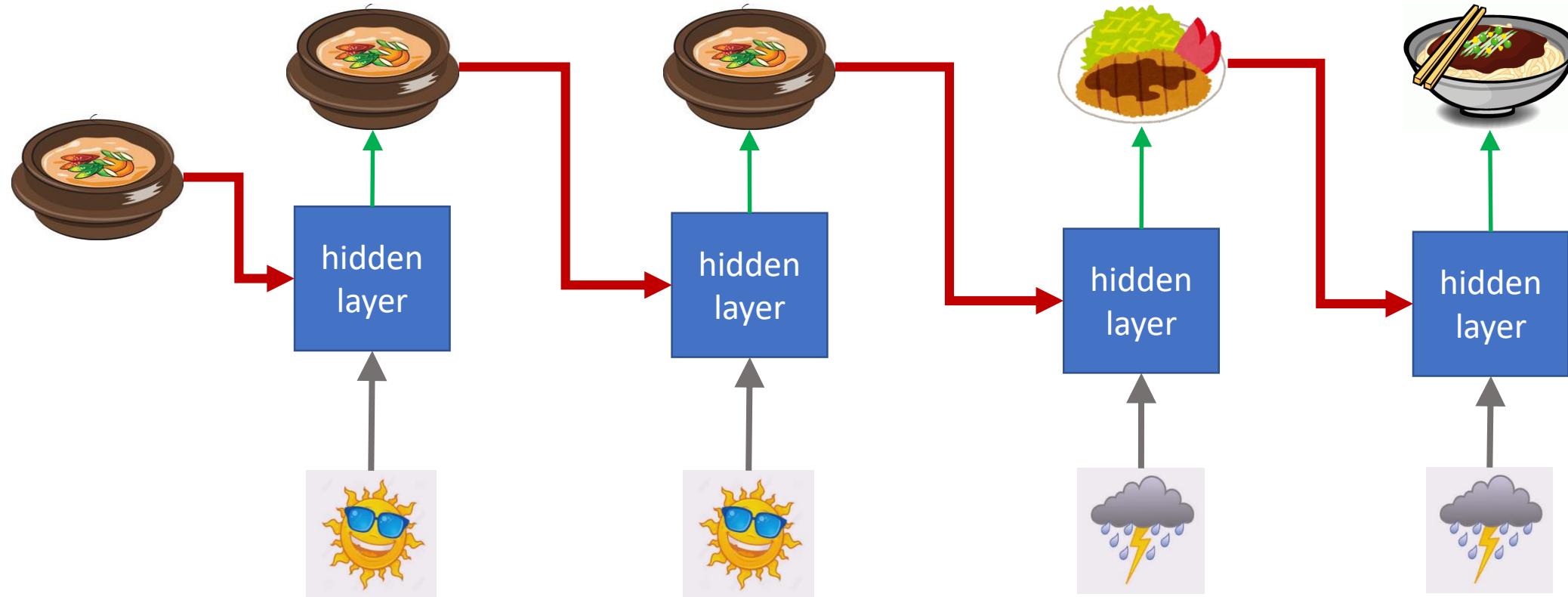
❖ The menu varies depending on the weather

- ✓ If the weather is nice, I eat food the day before.
- ✓ When it rains, I eat food according to the pattern



A simple of RNN (3/3)

- ❖ You can predict food depending on the weather



Neurons in RNN

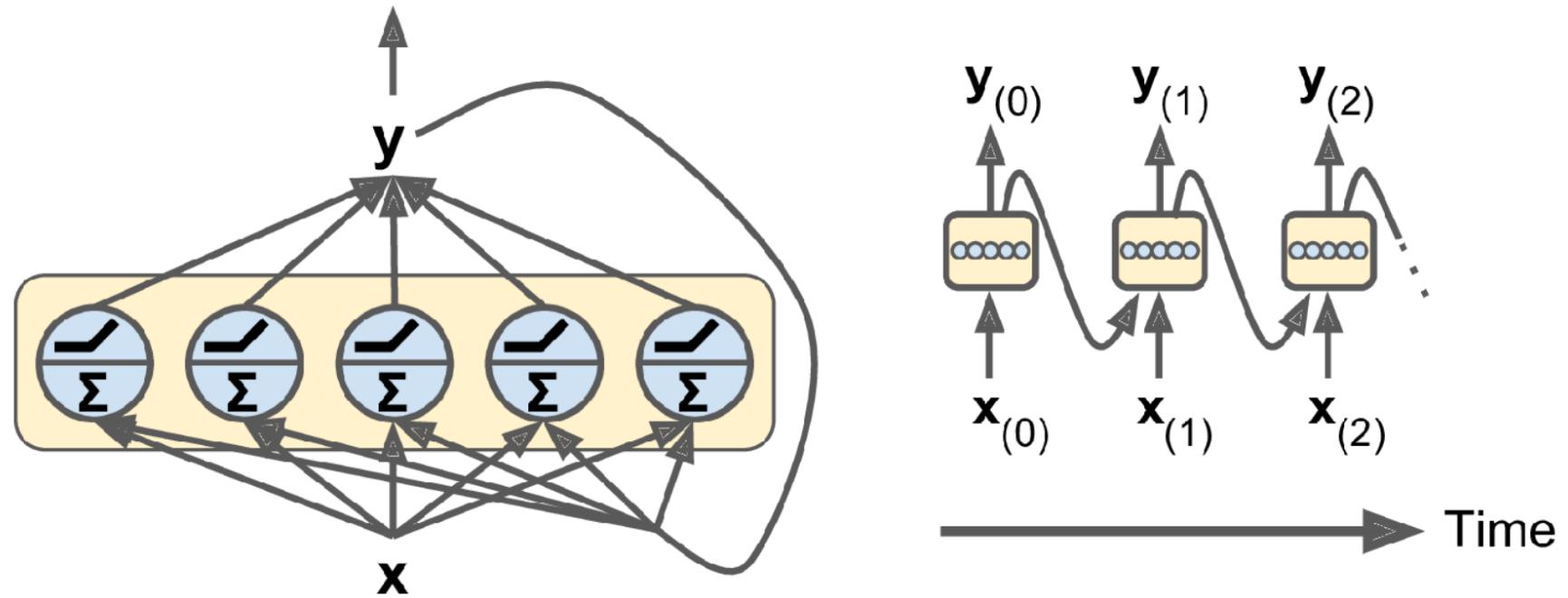
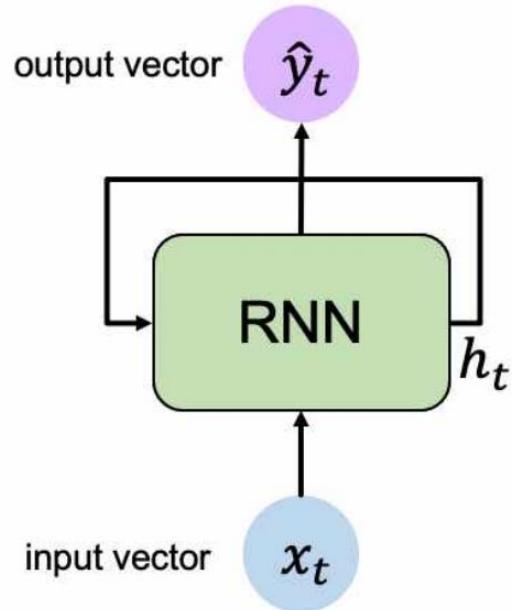


Figure 15-2. A layer of recurrent neurons (left) unrolled through time (right)

$$\mathbf{y}_{(t)} = \phi(\mathbf{W}_x^\top \mathbf{x}_{(t)} + \mathbf{W}_y^\top \mathbf{y}_{(t-1)} + \mathbf{b})$$

How to train RNN?

RNN State Update and Output



Output Vector

$$\hat{y}_t = \mathbf{W}_{hy}^T h_t$$

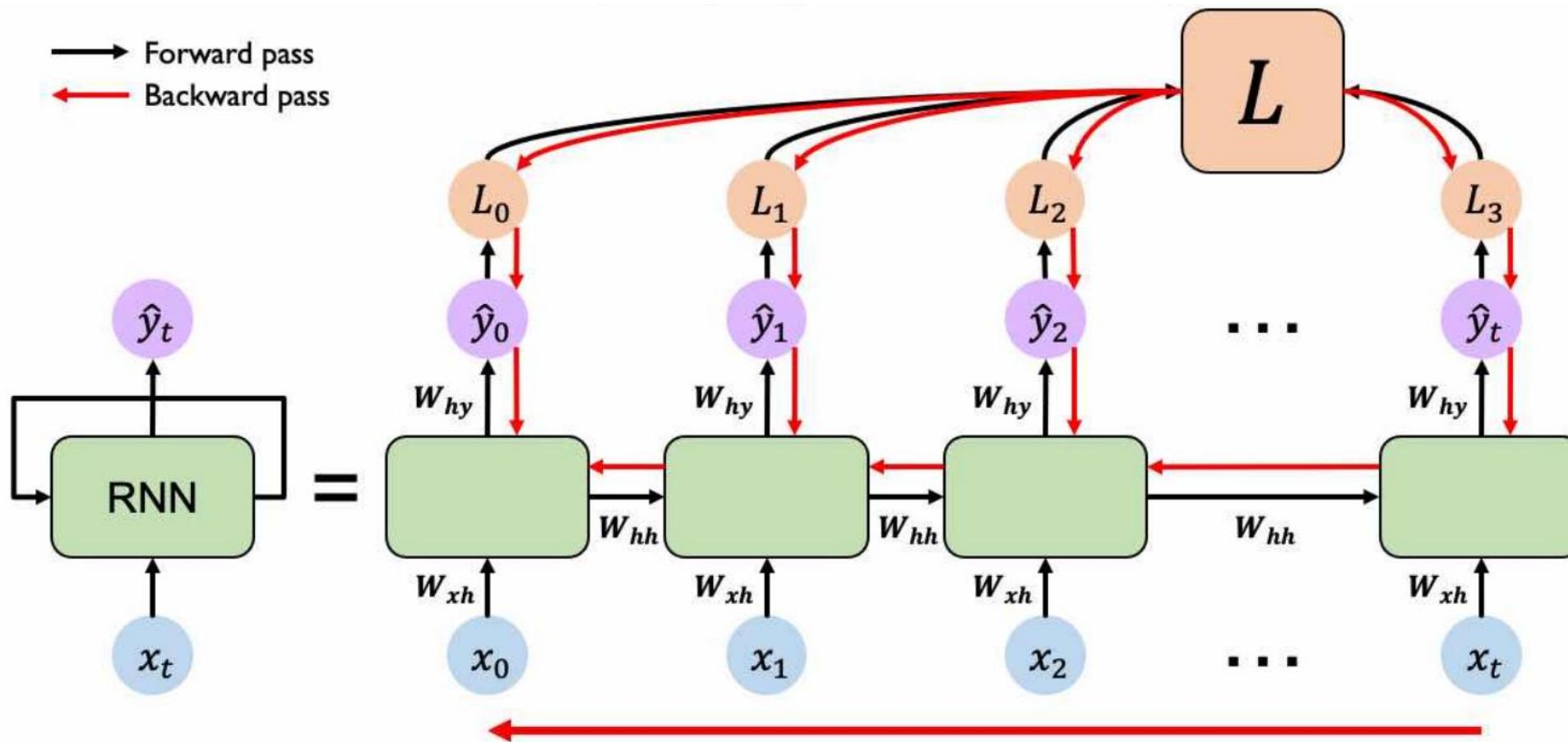
Update Hidden State

$$h_t = \tanh(\mathbf{W}_{hh}^T h_{t-1} + \mathbf{W}_{xh}^T x_t)$$

Input Vector

$$x_t$$

BPTT: Backpropagation Through Time



Standard RNN Gradient Flow

$$\frac{\partial L^{(t)}}{\partial \mathbf{W}_{hh}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \cdot \frac{\partial y^{(t)}}{\partial \mathbf{h}^{(t)}} \cdot \left(\sum_{k=1}^t \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \cdot \frac{\partial \mathbf{h}^{(k)}}{\partial \mathbf{W}_{hh}} \right)$$

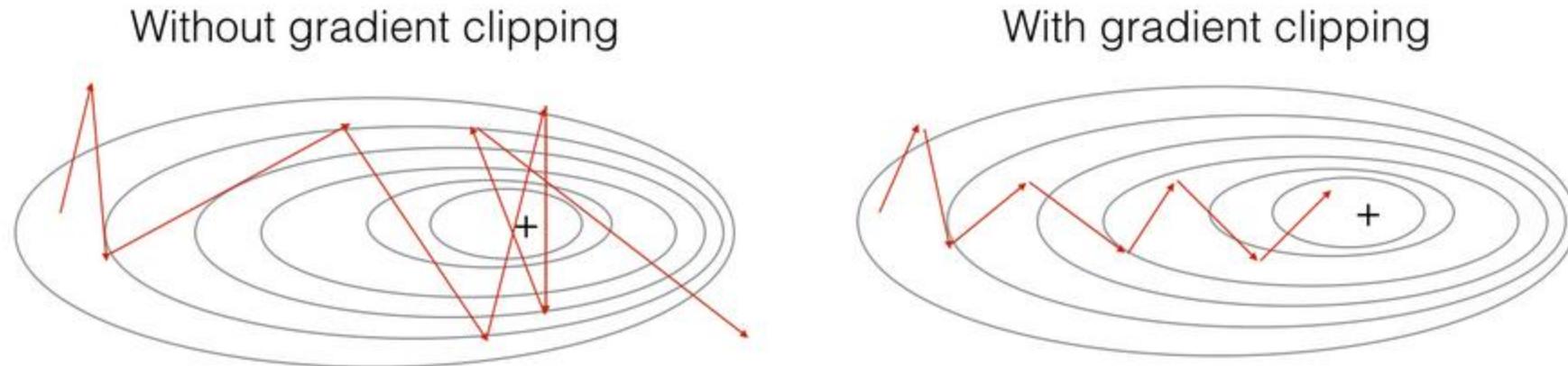
computed as a multiplication of adjacent time steps:

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^t \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$$

1. *Vanishing gradient* $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$
2. *Exploding gradient* $\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$

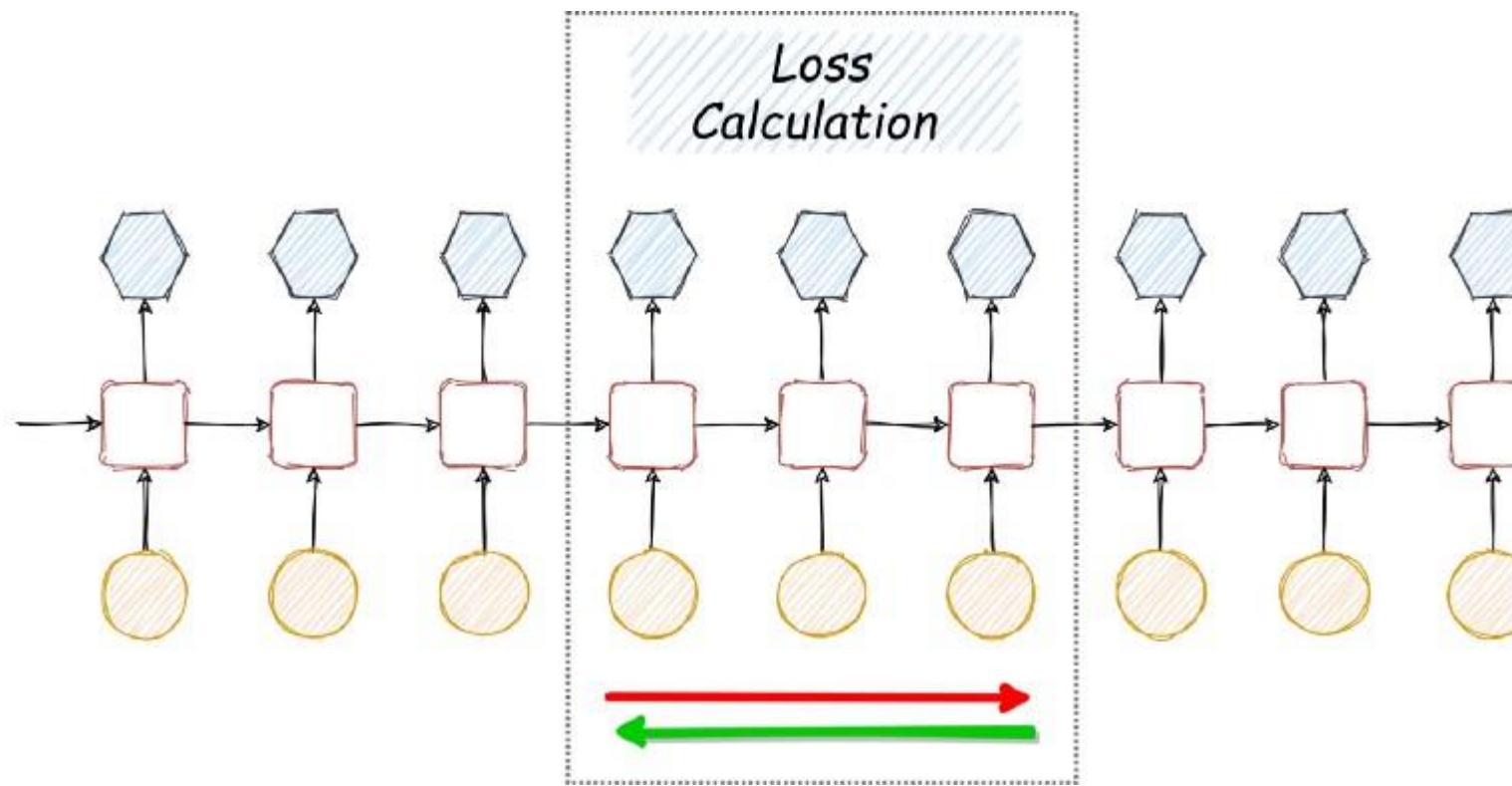


- ❖ Truncated backpropagation through time (TBPTT)
- ❖ Long short-term memory (LSTM)
- ❖ Gradient Clipping

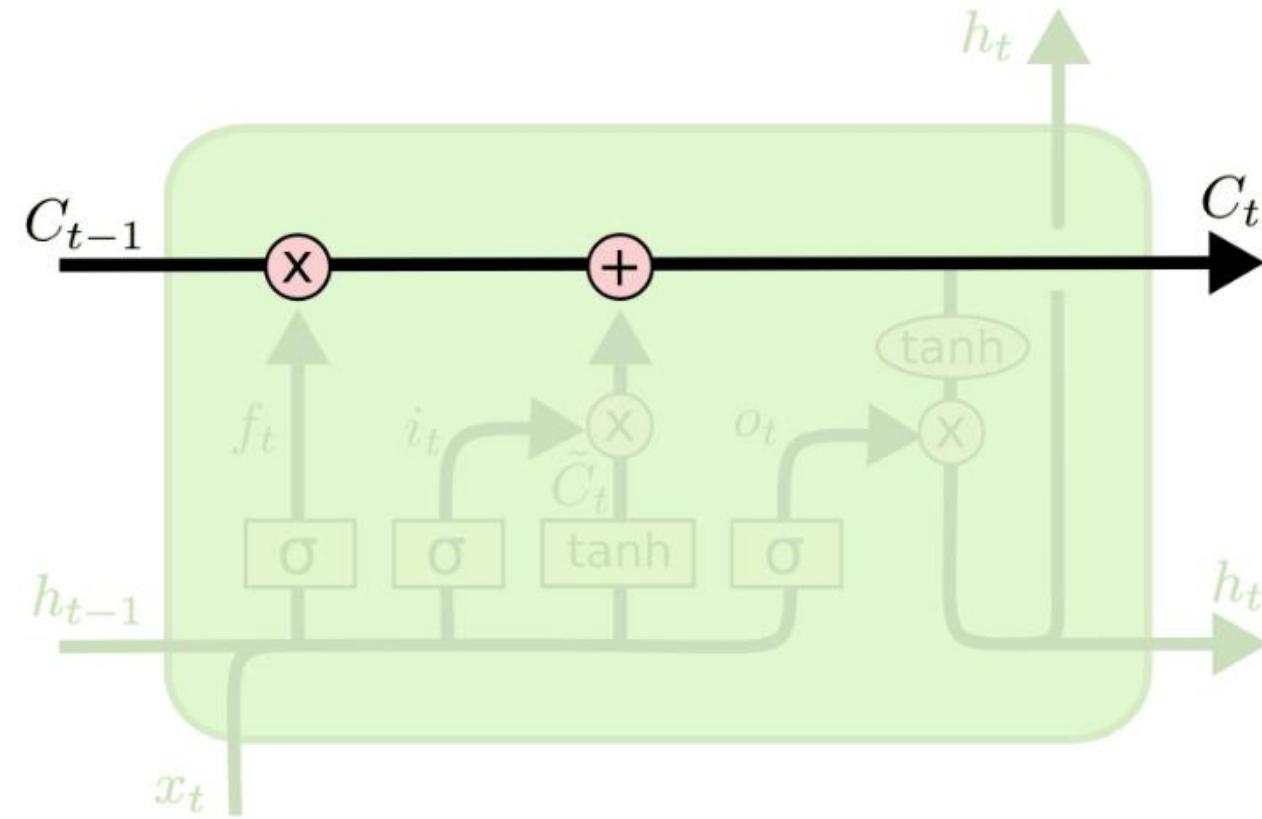


Truncated Backpropagation Through Time

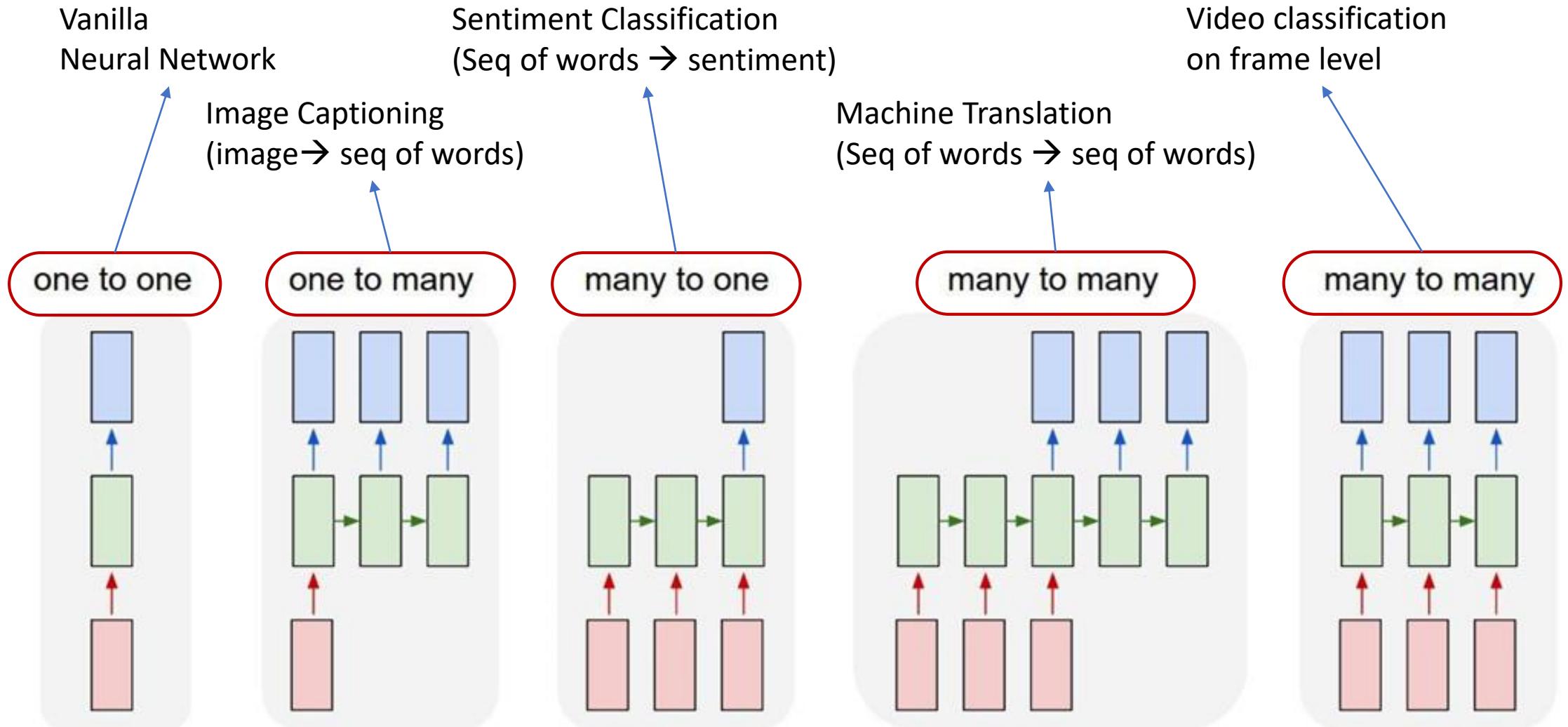
- ❖ TBPTT: Truncated BPTT trick by considering a moving window through the training process



- ❖ Capable of learning long-term dependencies by remembering information
- ❖ Cell state

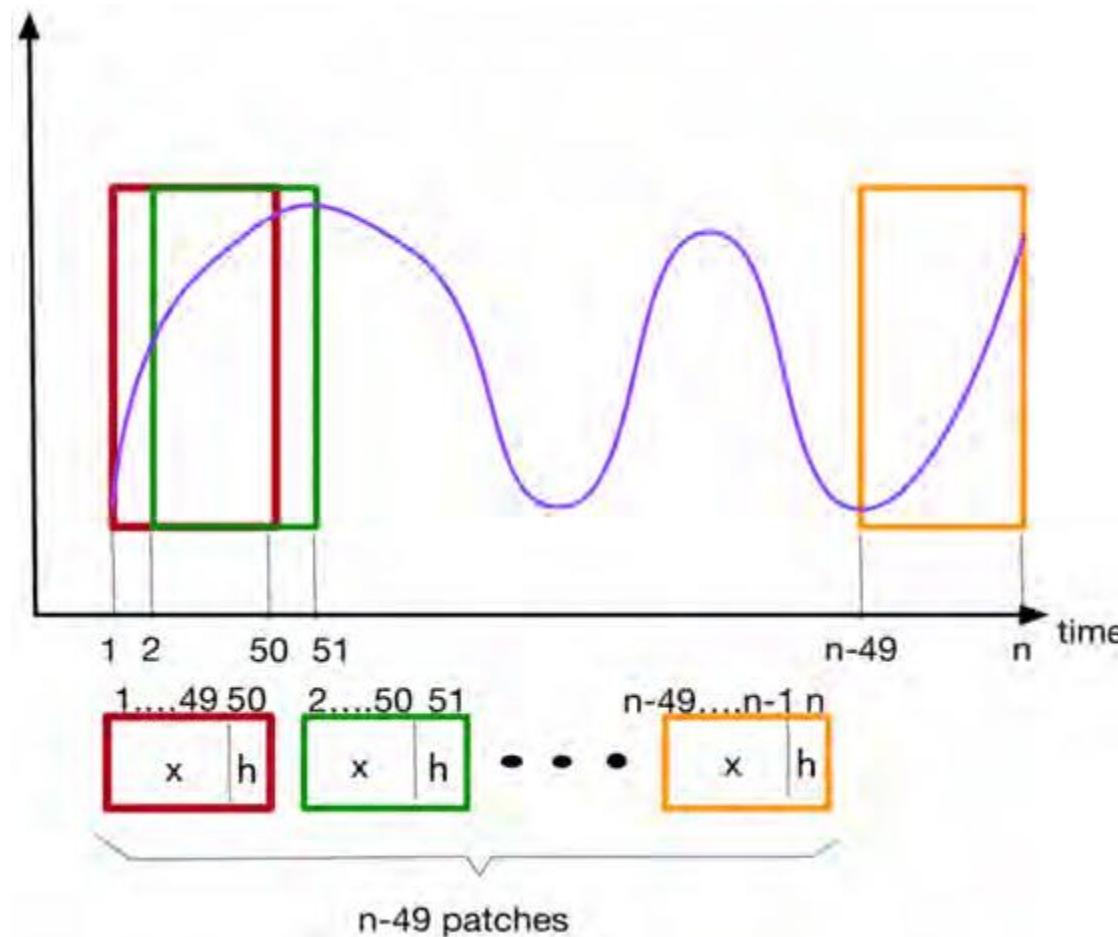


Recurrent Neural Networks: Process Sequences

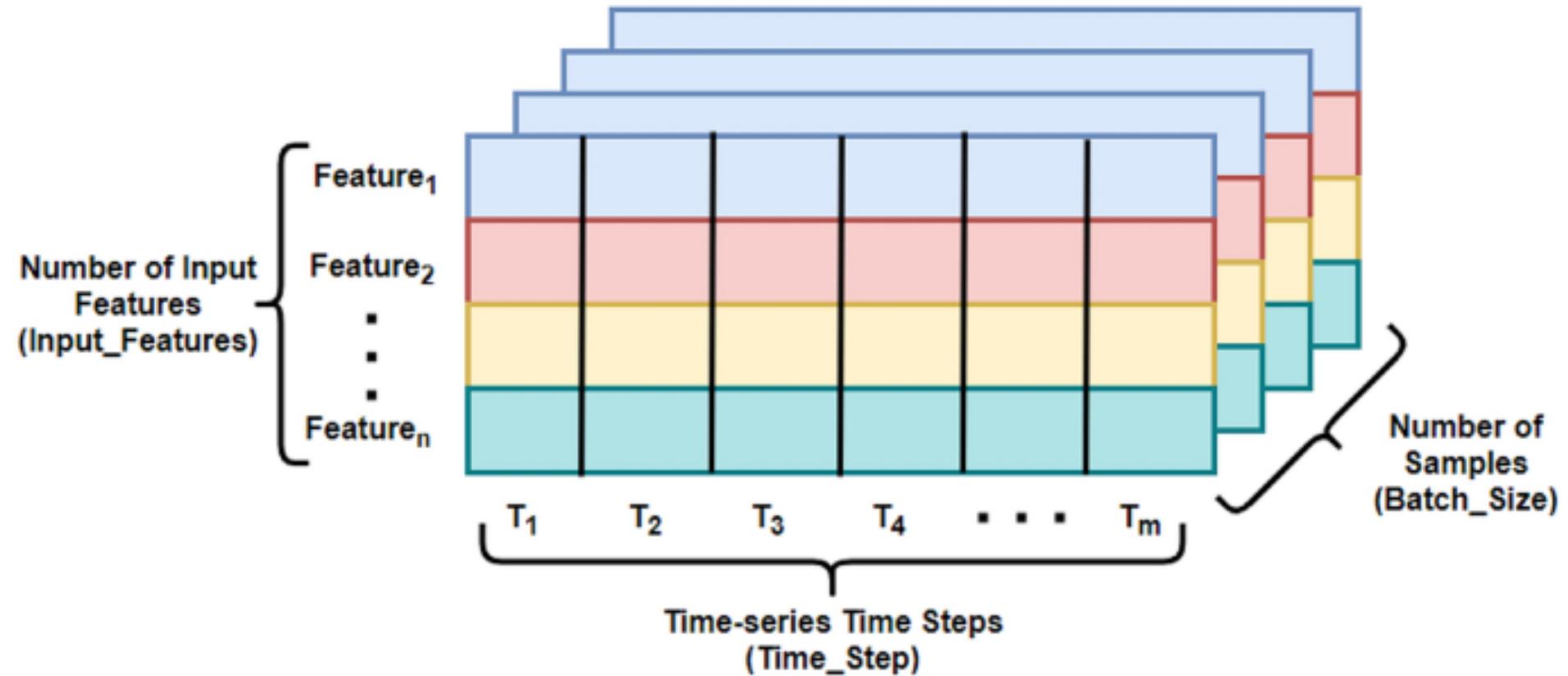


Implementing an RNN with Keras

RNN Sliding window size and horizon

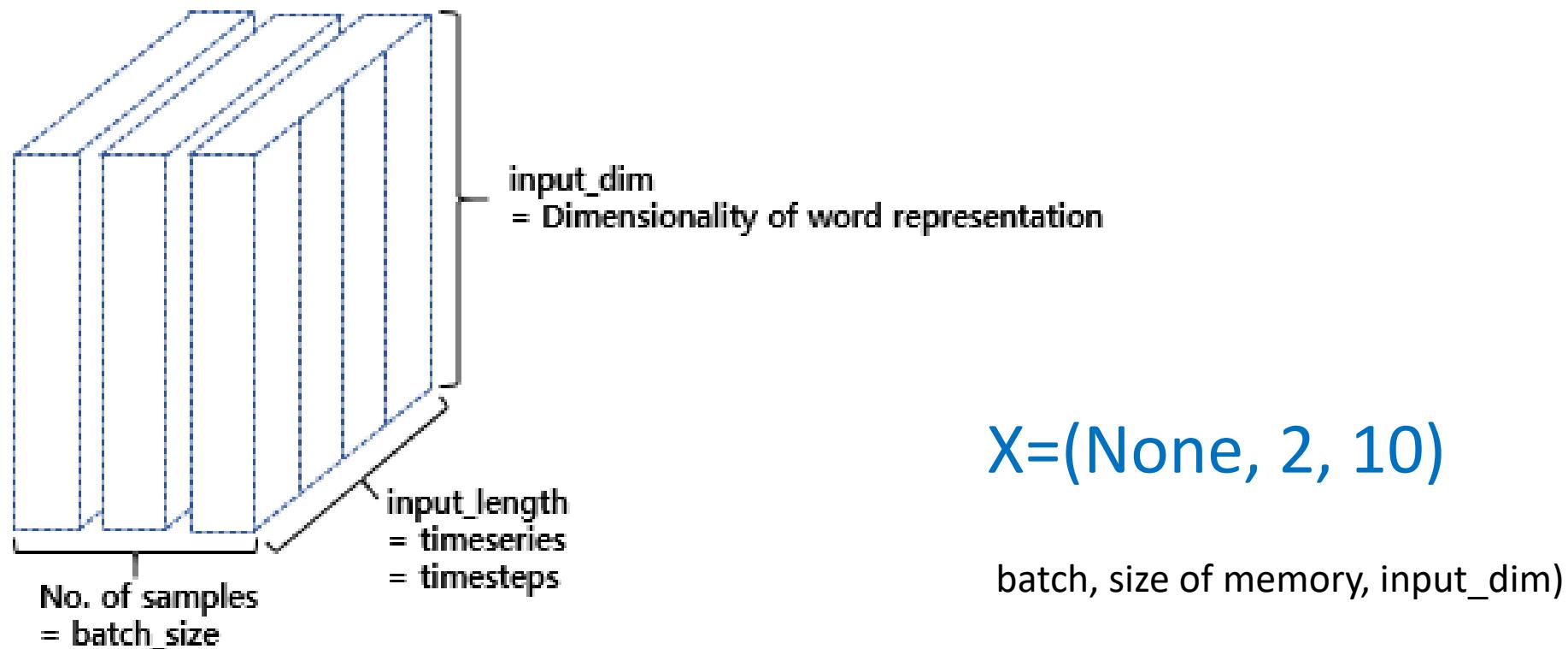


❖ RNN Input Data



RNN Input : X Dimension

- ❖ The RNN layer receives a 3D tensor of size as input
 - ✓ batch_size, timesteps, and input_dim



SimpleRNN without Batch

model.add(SimpleRNN(3, input_shape=(2,10)))

Model: "sequential"

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 3)	42

=====
Total params: 42
Trainable params: 42
Non-trainable params: 0



X=(None, 2, 10)

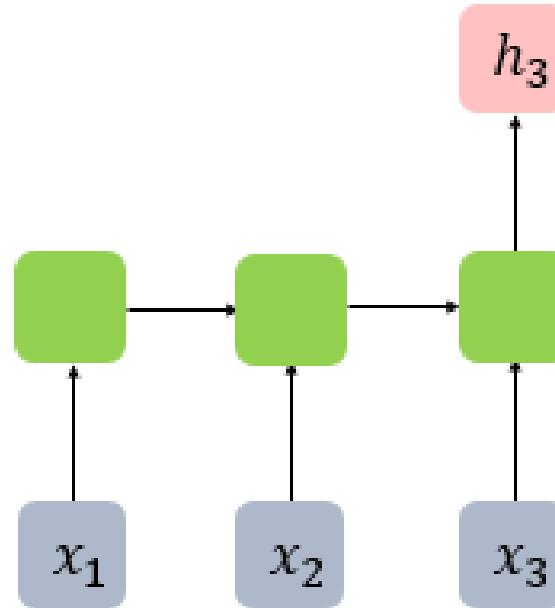
with Batch size = 8

```
model.add(SimpleRNN(3, batch_input_shape=(8,2,10)))
```

Layer (type)	Output Shape	Param #
simple_rnn_1 (SimpleRNN)	(8, 3)	42

❖ Setting the return_sequences of the RNN layer

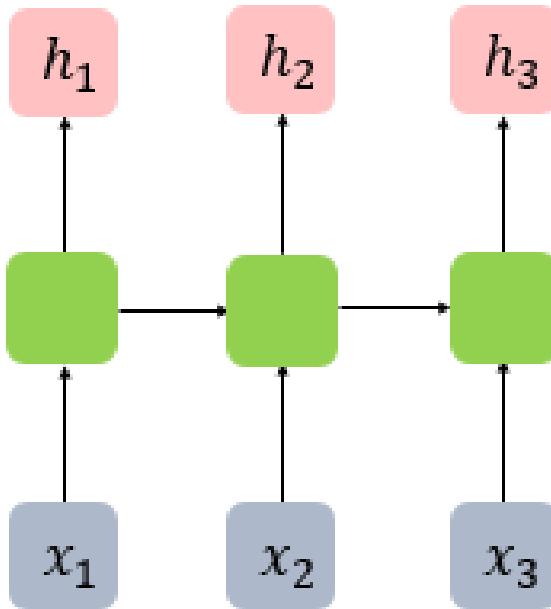
return_sequences=False



다음층으로 마지막 은닉 상태만 전달

many-to-one task

return_sequences=True



다음층으로 모든 은닉 상태 전달

many-to-many task

[HW] Let's Code! Simple RNN

Input Data and Train Data Shape

```
1 import numpy as np
2 import tensorflow as tf
3 from tensorflow.keras.layers import SimpleRNN, GRU, LSTM, Bidirectional
4 train_X = [[0.1, 4.2, 1.5, 1.1, 2.8], [1.0, 3.1, 2.5, 0.7, 1.1], [0.3, 2.1, 1.5, 2.1, 0.1], [2.2, 1.4, 0.5, 0.9, 1.1]]
5 print(np.shape(train_X))
```

```
1 train_X=np.reshape(train_X, (-1,4,5))
2 print(train_X.shape)
```

(4, 5)
(1, 4, 5)

SimpleRNN and Hidden states

```
1 rnn = SimpleRNN(3)
2 # rnn = SimpleRNN(3, return_sequences=False, return_state=False)와 동일.
3 hidden_state = rnn(train_X)
4
5 print('hidden state : {}, shape: {}'.format(hidden_state, hidden_state.shape))
```

hidden state : [[0.9206875 -0.9963834 -0.99492633]], shape: (1, 3)

```
1 rnn = SimpleRNN(3, return_sequences=True)
2 hidden_states = rnn(train_X)
3
4 print('hidden states : {}, shape: {}'.format(hidden_states, hidden_states.shape))
```

hidden states : [[[[-0.91097903 -0.99876505 -0.95570076]
[0.9831649 -0.52204293 -0.94400156]
[-0.7801549 0.03199867 -0.44760188]
[0.8878632 -0.20588721 0.7986243]]], shape: (1, 4, 3)

SimpleRNN and Return Sequence

```
1 rnn = SimpleRNN(3, return_sequences=True, return_state=True)
2 hidden_states, last_state = rnn(train_X)
3
4 print('hidden states : {}, shape: {}'.format(hidden_states, hidden_states.shape))
5 print('last hidden state : {}, shape: {}'.format(last_state, last_state.shape))
```

```
hidden states : [[[0.99741   0.5095164  0.9999336 ]
 [0.97162277 0.32083237 0.99995947]
 [0.95783615 0.86253214 0.99940664]
 [0.99748254 0.7658603  0.9743646 ]]], shape: (1, 4, 3)
last hidden state : [[0.99748254 0.7658603  0.9743646 ]], shape: (1, 3)
```

Simple RNN and Return_Sequence

```
1 rnn = SimpleRNN(3, return_sequences=False, return_state=True)
2 hidden_state, last_state = rnn(train_X)
3
4 print('hidden state : {}, shape: {}'.format(hidden_state, hidden_state.shape))
5 print('last hidden state : {}, shape: {}'.format(last_state, last_state.shape))
```

```
hidden state : [[0.9096651 0.9990103 0.7855833]], shape: (1, 3)
last hidden state : [[0.9096651 0.9990103 0.7855833]], shape: (1, 3)
```

2022

Korea Institute of Science
and Technology Information

TRUST
KISTI

