# Deep Learning

## RNN-based Models (Lab)

**U Kang**
**Seoul National University**

# In This Lecture

- RNN-based Models
  - Autocomplete Model
  - Seq2Seq Model
- Word embedding
- Batch generation
- Autoencoder structure

# Outline

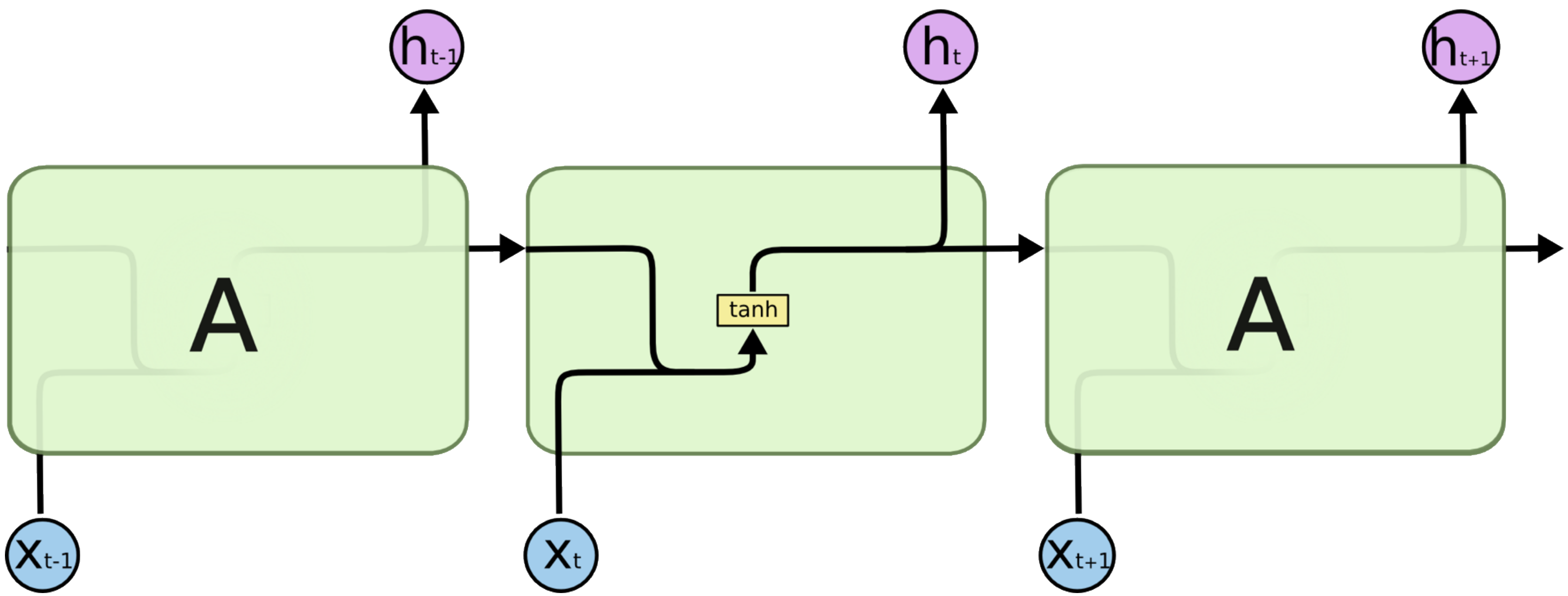
- ☐ **Overview**
- ☐ Autocomplete Model
- ☐ Seq2Seq Model

# Overview

- There exist various RNN-based models
- Autocomplete model
  - **Problem:** to finish an incomplete sentence
  - Basic RNN application for sequential data
- Seq2Seq Model
  - **Problem:** to convert a sequence into another
  - Mixture of an autoencoder and RNN cells

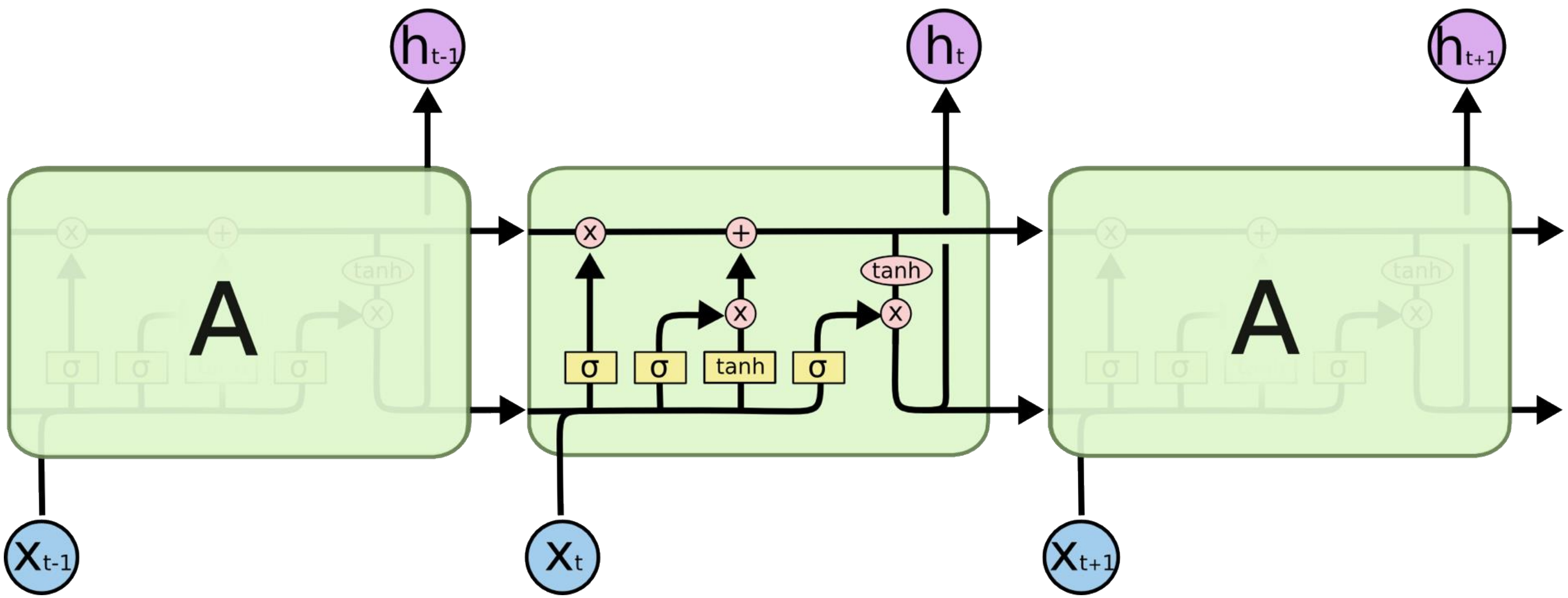
# Recurrent Neural Network (RNN)

- A deep learning structure for sequential data
- Contains a **cell**, which is a repeated structure
- Stores and passes **states** through a sequence



# Long Short-term Memory (LSTM)

- An advanced RNN structure
- It avoids the long-term dependency problem



# Outline

- ☑ Overview
- ☐ **Autocomplete Model**
- ☐ Seq2Seq Model

# Autocomplete Model (1)

- We'll implement a simple autocomplete model
- It learns a sequence of four words
- Given the first three words, it predicts the last one
- It is called "autocomplete" because it completes the given incomplete sentence

# Autocomplete Model (2)

- For example, it solves the following problem
- Given "I went to," which word will come?
  - School: 95%
  - John: 3%
  - Me: 1%
  - Monitor: 1%
- In our model, we use letters instead of words
  - For example, given "lov" which letter will come next?

# Import Statements

- Let's import *tensorflow keras packages* and *numpy*

```
import numpy as np
from tensorflow.keras import layers, Sequential, losses, optimizers
```

# Word Embedding (1)

- We want to implement a language model
- But, a computer cannot take word inputs
  - It prefers numerical values, not strings
- We need to convert words into vectors

# Word Embedding (2)

- We generate a character pool as follows:

```
char_arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g',
            'h', 'i', 'j', 'k', 'l', 'm', 'n',
            'o', 'p', 'q', 'r', 's', 't', 'u',
            'v', 'w', 'x', 'y', 'z']
num_dic = {n: i for i, n in enumerate(char_arr)}
dic_len = len(num_dic)
```

- These are all the letters we should consider
- We build a simple dictionary for conversion

# Training Data

- We generate the following training data:

```
seq_data = ['word', 'wood', 'deep', 'dive', 'cold',
            'cool', 'load', 'love', 'kiss', 'kind']
```

- Each string is a sequence of four letters
- Thus, it contains both features and the label
- We have 10 training instances

# Batch Generation

- We define a function to generate batches:

```
def make_batch(seq_data):
    input_batch, target_batch = [], []

    for seq in seq_data:
        input = [num_dic[n] for n in seq[:-1]]
        target = num_dic[seq[-1]]
        input_batch.append(np.eye(dic_len)[input])
        target_batch.append(target)

    return np.array(input_batch, dtype=np.float32), np.array(target_batch, dtype=np.int32)
```

- *target_batch* contains the last letters

# Hyperparameters

- We set hyperparameters of the model:

```
learning_rate = 0.01
n_hidden = 128
total_epoch = 30
n_step = 3
n_input = n_class = dic_len
```

# Outline

- ☑ Overview
- ☑ Autocomplete Model
- ☐ **Seq2Seq Model**

# Seq2Seq Model (1)

- We'll implement a simple seq2seq model
- It is basically an autoencoder model
  - We use the outputs from the middle layer
- We solve the **machine translation** problem
- It is to translate an English word into Korean

# Seq2Seq Model (2)

- For example, our model learned “love”
- Then, what would be the meaning of “lovely?”
- Our model should be able to translate words
  - That do not appear in the training set
  - That have variable lengths (“love” and “lovely”)

# Word Embedding

- We generate a character pool as follows:

```
char_arr = [c for c in
              'SEabcdefghijklmnopqrstuvwxyz'
              '단어나무놀이사과범컴퓨터']
num_dic = {n: i for i, n in enumerate(char_arr)}
dic_len = len(num_dic)
```

- *char_arr* contains all ENG & KOR characters
- *num_dic* maps a character into an integer

# Training Data

- We generate the following training data:

```
seq_data = [['word', '단어'], ['wood', '나무'],
            ['game', '놀이'], ['apple', '사과'],
            ['tiger', '범'], ['computer', '컴퓨터']]
max_length = 10
```

- Each instance is a pair of words
  - One is English and the other is Korean
  - Korean words correspond to the labels
- We set the maximum length of a word

# Batch Generation (1)

- We use all training data as a single batch
- We need four kinds of data:
  - Input data (*batch_e*) to the encoder
  - Input data (*batch_d*) to the decoder
  - Target data (*batch_y*) as ground truth
  - Length (*len_y*) of the target data
- The last three are needed only for training
  - *batch_d, batch_y, len_y*

# Batch Generation (2)

- The input and output of the batch function:

```
def make_batch(seq_data):
    batch_e, batch_d, batch_y, len_y = [], [], [], []

    for seq in seq_data:
        pass

    return np.array(batch_e), np.array(batch_d), np.array(batch_y), np.array(len_y)
```

# Batch Generation (3)

- What we actually do at the *pass* statement

```
input = np.zeros((max_length, dic_len))
output = np.zeros((max_length, dic_len))
target = np.zeros(max_length, dtype=int)

for i, n in enumerate(seq[0]):
    input[i, num_dic[n]] = 1
for i, n in enumerate('S' + seq[1]):
    output[i, num_dic[n]] = 1
for i, n in enumerate(seq[1] + 'E'):
    target[i] = num_dic[n]

batch_e.append(input)
batch_d.append(output)
batch_y.append(target)
len_y.append(len(seq[1]) + 1)
```

# Batch Generation (4)

- Each word is represented as a one-hot vector
  - Thus, the shape of *input* and *output* is $(n, d)$
    - $n$ is the maximum length of a sequence
    - $d$ is the number of unique (possible) words
  - It is not necessary for the *target* sequence

- We add zero vectors to the end of each seq.
  - Because the sequences have variable lengths

# Hyperparameters

- We set hyperparameters of the model:

```
learning_rate = 0.01
n_hidden = 128
total_epoch = 100
n_class = n_input = dic_len
```

# Questions?