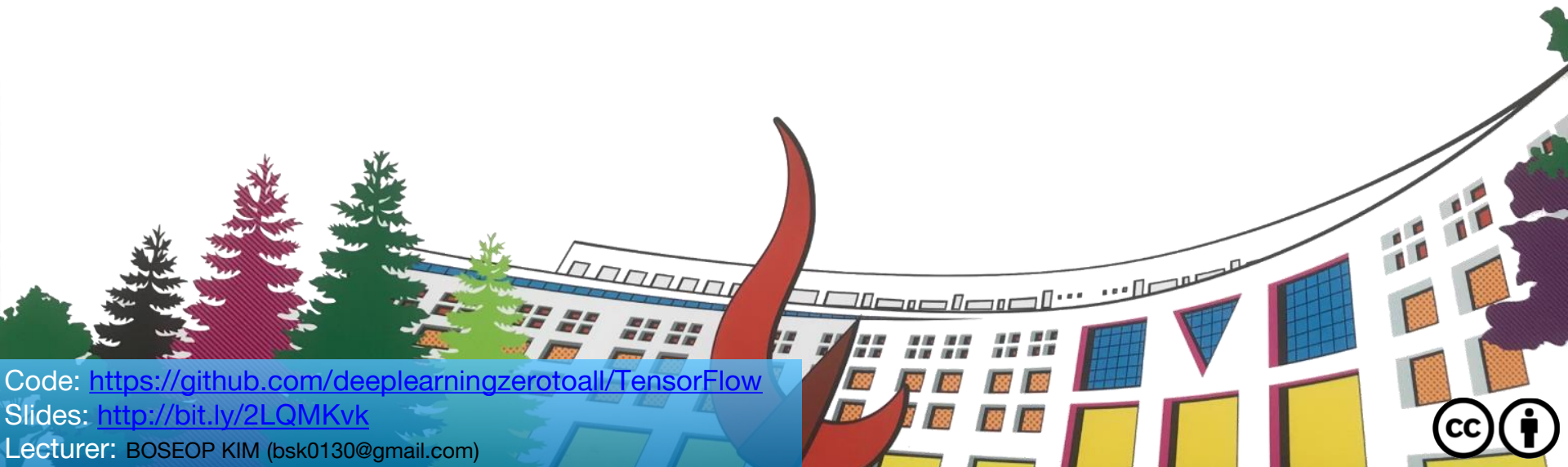


# ML/DL for Everyone Season2

with  TensorFlow

## Lab 12-0 rnn basics



Code: <https://github.com/deeplearningzerotoall/TensorFlow>

Slides: <http://bit.ly/2LQMKvk>

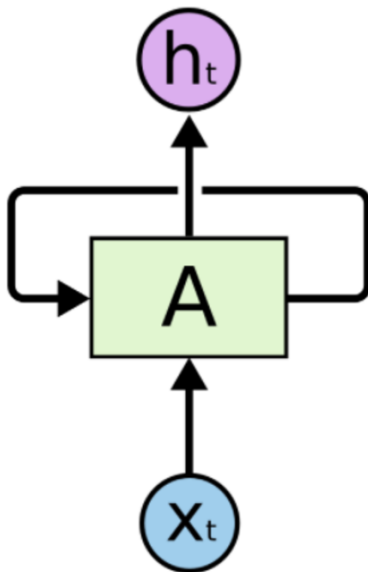
Lecturer: BOSEOP KIM (bsk0130@gmail.com)



# rnn basics

- RNN in TensorFlow
- One cell: 4 (input-dim), 2 (hidden\_size)
- Unfolding to n sequences
- Batching input

# RNN in TensorFlow 2.0



```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

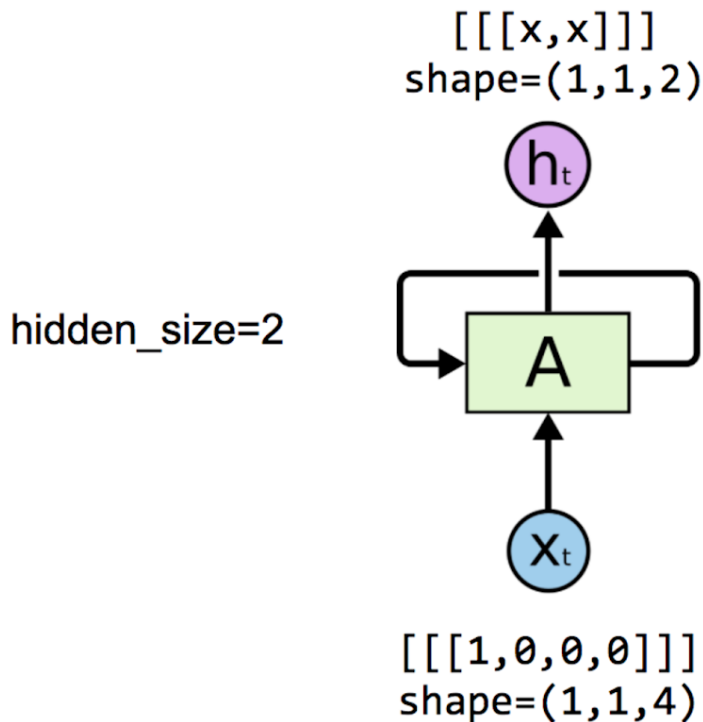
⋮

```
cell = layers.SimpleRNNCell(units=hidden_size)
rnn = layers.RNN(cell, return_sequences=True, return_state=True)
outputs, states = rnn(x_data)
```



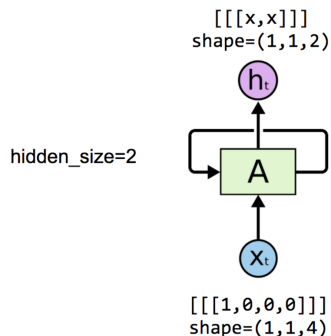
```
rnn = layers.SimpleRNN(units=hidden_size, return_sequences=True,
                        return_state=True)
outputs, states = rnn(x_data)
```

# One cell: 4 (input-dim), 2 (hidden\_size)



# One hot encoding  
 $h = [1, 0, 0, 0]$   
 $e = [0, 1, 0, 0]$   
 $l = [0, 0, 1, 0]$   
 $o = [0, 0, 0, 1]$

# One cell: 4 (input-dim), 2 (hidden\_size)



```
# One hot encoding
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

```
# setup
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

```
print(tf.__version__)
tf.enable_eager_execution()
```

```
# One hot encoding for each char in 'hello'
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

```
# One cell RNN input_dim (4) -> output_dim (2)
x_data = np.array([[h]], dtype=np.float32)
```

```
hidden_size = 2
cell = layers.SimpleRNNCell(units=hidden_size)
rnn = layers.RNN(cell, return_sequences=True, return_state=True)
outputs, states = rnn(x_data)
```

```
# equivalent to above
# rnn = layers.SimpleRNN(units=hidden_size, return_sequences=True,
#                          return_state=True)
# outputs, states = rnn(x_data)
```

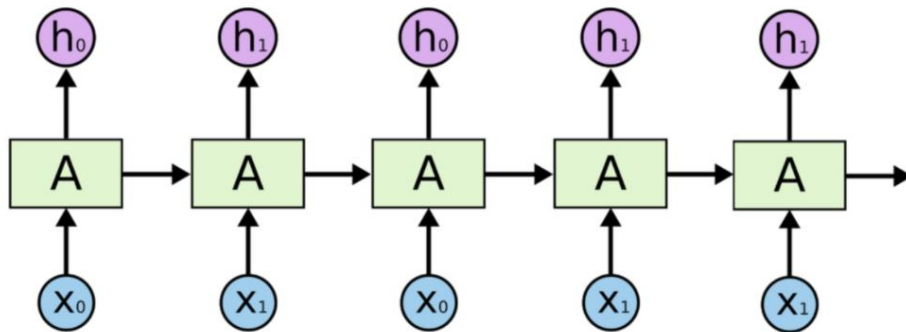
```
print('x_data: {}, shape: {}'.format(x_data, x_data.shape))
print('outputs: {}, shape: {}'.format(outputs, outputs.shape))
print('states: {}, shape: {}'.format(states, states.shape))
```

```
x_data: [[[1. 0. 0. 0.]]], shape: (1, 1, 4)
outputs: [[[0.32261637 0.5036928 ]]], shape: (1, 1, 2)
states: [[[0.32261637 0.5036928 ]]], shape: (1, 2)
```

# Unfolding to n sequences

```
hidden_size=2  
sequence_length=5
```

```
shape=(1,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]]]
```

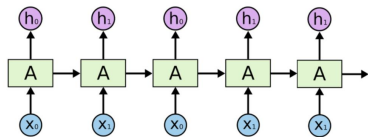


```
shape=(1,5,4): [[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,1,0], [0,0,0,1]]]  
                  h       e       l       l       o
```

# Unfolding to n sequences

hidden\_size=2  
sequence\_length=5

shape=(1,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]]]



shape=(1,5,4): [[[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1], [0,0,0,1]]]]  
                  h          e          l          l          o

```
# One cell RNN input_dim (4) -> output_dim (2). sequence: 5
x_data = np.array([[h, e, l, l, o]], dtype=np.float32)
```

```
hidden_size = 2
rnn = layers.SimpleRNN(units=2, return_sequences=True, return_state=True)
outputs, states = rnn(x_data)
```

```
print('x_data: {}, shape: {}'.format(x_data, x_data.shape))
print('outputs: {}, shape: {}'.format(outputs, outputs.shape))
print('states: {}, shape: {}'.format(states, states.shape))
```

```
x_data: [[[1. 0. 0. 0.]
[0. 1. 0. 0.]
[0. 0. 1. 0.]
[0. 0. 1. 0.]
[0. 0. 0. 1.]]], shape: (1, 5, 4)
```

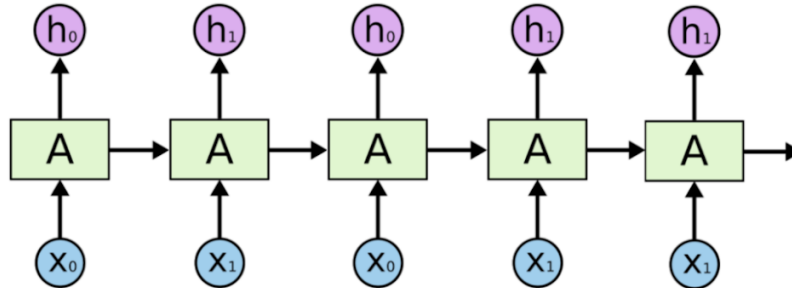
```
outputs: [[[ 0.36337885  0.73452437]
[ 0.23541063 -0.28505793]
[-0.19638212 -0.54805404]
[-0.589804    -0.65221256]
[-0.8427679   0.19108507]]], shape: (1, 5, 2)
```

```
states: [[-0.8427679   0.19108507]], shape: (1, 2)
```

# Batching input

hidden\_size=2  
sequence\_length=5  
batch = 3

shape=(3,5,2):  $\begin{bmatrix} [x,x] & [x,x] & [x,x] & [x,x] & [x,x] \\ [x,x] & [x,x] & [x,x] & [x,x] & [x,x] \\ [x,x] & [x,x] & [x,x] & [x,x] & [x,x] \end{bmatrix}$



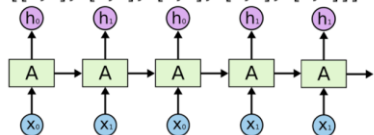
shape=(3,5,4):  $\begin{bmatrix} [1,0,0,0] & [0,1,0,0] & [0,0,1,0] & [0,0,1,0] & [0,0,0,1] \\ [0,1,0,0] & [0,0,0,1] & [0,0,1,0] & [0,0,1,0] & [0,0,1,0] \\ [0,0,1,0] & [0,0,1,0] & [0,1,0,0] & [0,1,0,0] & [0,0,1,0] \end{bmatrix}$  # hello  
# eolll  
# lleeel



# Batching input

```
hidden_size=2
sequence_length=5
batch = 3
```

```
shape=(3,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]],
                [[x,x], [x,x], [x,x], [x,x], [x,x]],
                [[x,x], [x,x], [x,x], [x,x], [x,x]]]
```



```
shape=(3,5,4): [[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,1,0], [0,0,0,1]], # hello
                [[0,1,0,0], [0,0,0,1], [0,0,0,1], [0,0,1,0], [0,0,1,0]] # eolll
                [[0,0,1,0], [0,0,1,0], [0,1,0,0], [0,1,0,0], [0,0,1,0]] # lleel]
```

# One cell RNN input\_dim (4) -> output\_dim (2). sequence: 5, batch 3  
 # 3 batches 'hello', 'eolll', 'lleel'

```
x_data = np.array([[h, e, l, l, o],
                  [e, o, l, l, l],
                  [l, l, e, e, l]], dtype=np.float32)
```

```
hidden_size = 2
rnn = layers.SimpleRNN(units=2, return_sequences=True, return_state=True)
outputs, states = rnn(x_data)
```

```
print('x_data: {}, shape: {}'.format(x_data, x_data.shape))
print('outputs: {}, shape: {}'.format(outputs, outputs.shape))
print('states: {}, shape: {}'.format(states, states.shape))
```

```
x_data: [[[1. 0. 0. 0.]
          [0. 1. 0. 0.]
          [0. 0. 1. 0.]
          [0. 0. 1. 0.]
          [0. 0. 0. 1.]]
          [[0. 1. 0. 0.]
          [0. 0. 0. 1.]
          [0. 0. 1. 0.]
          [0. 0. 1. 0.]
          [0. 0. 1. 0.]]
          [[0. 0. 1. 0.]
          [0. 0. 1. 0.]
          [0. 1. 0. 0.]
          [0. 1. 0. 0.]
          [0. 0. 1. 0.]]], shape: (3, 5, 4)
```

```
outputs: [[[-0.56743866 -0.23173441]
            [ 0.8334968  0.30382484]
            [-0.93795335 -0.53330624]
            [-0.0874121  0.45240858]
            [-0.52966034 -0.5007928 ]]
```

```
[[ 0.58572567 0.00302648]
 [-0.8668559 -0.19943324]
 [-0.10968345 0.14605513]
 [-0.6695559 -0.29440066]
 [-0.30860662 0.20964848]]
```

```
[[[-0.73496014 -0.17280702]
   [-0.23251031 0.10149368]
   [ 0.723902 -0.06399264]
   [-0.0544093 -0.03742061]
   [-0.7118617 -0.12900047]]], shape: (3, 5, 2)
```

```
states: [[[-0.52966034 -0.5007928 ]
            [-0.30860662 0.20964848]
            [-0.7118617 -0.12900047]], shape: (3, 2)
```

# What's Next?

- many to one