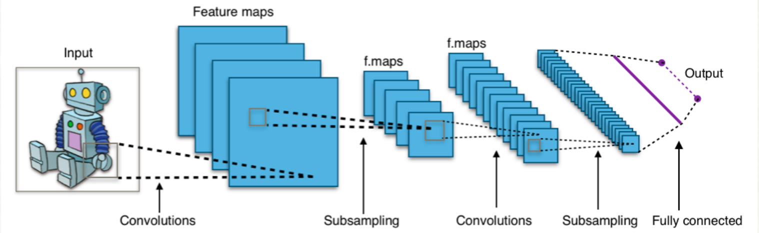
Recurrent Neural Networks

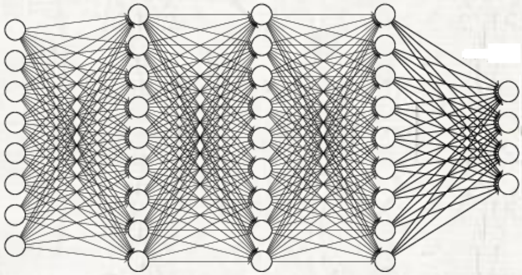
J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

Recurrent Neural Network

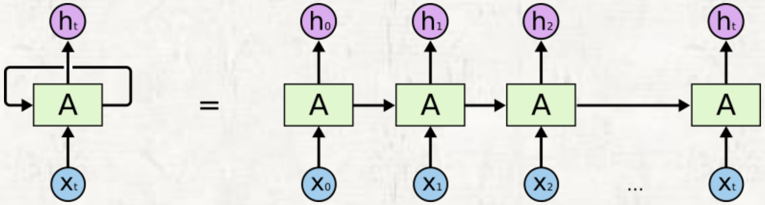
- Type of Deep Neural Network
  - Dense Network (=Fully-connected Neural Network)
  - Convolutional Neural Network
  - **Recurrent Neural Network**
  - …



<Convolutional Neural Network>



<Dense Network>

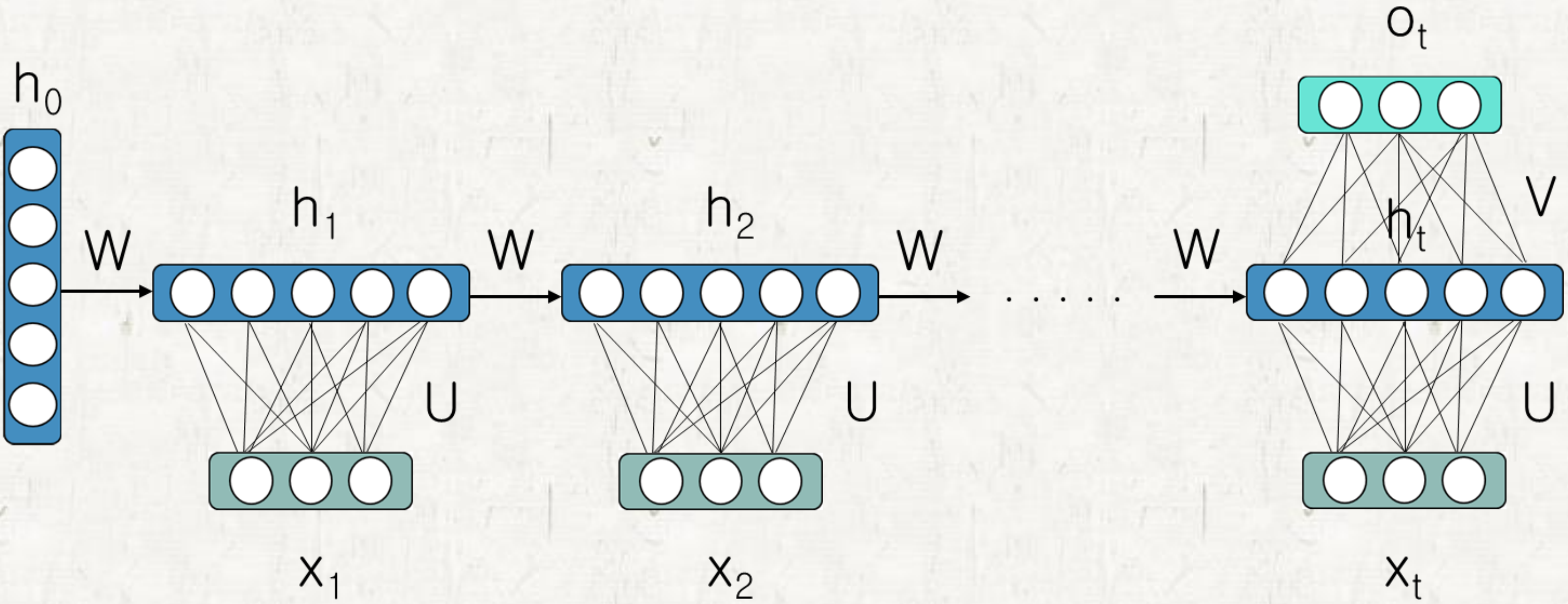


<Recurrent Neural Network>

J.H. Lee, Dept. of Software, Sungkyunkwan Univ.

Recurrent Neural Network

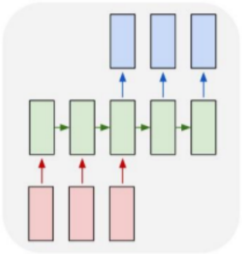
- What is the Recurrent Neural Network?



$$h_t = \tanh(Ux_t + Wh_{t-1})$$

$$o_t = \text{Softmax}(Vh_t)$$

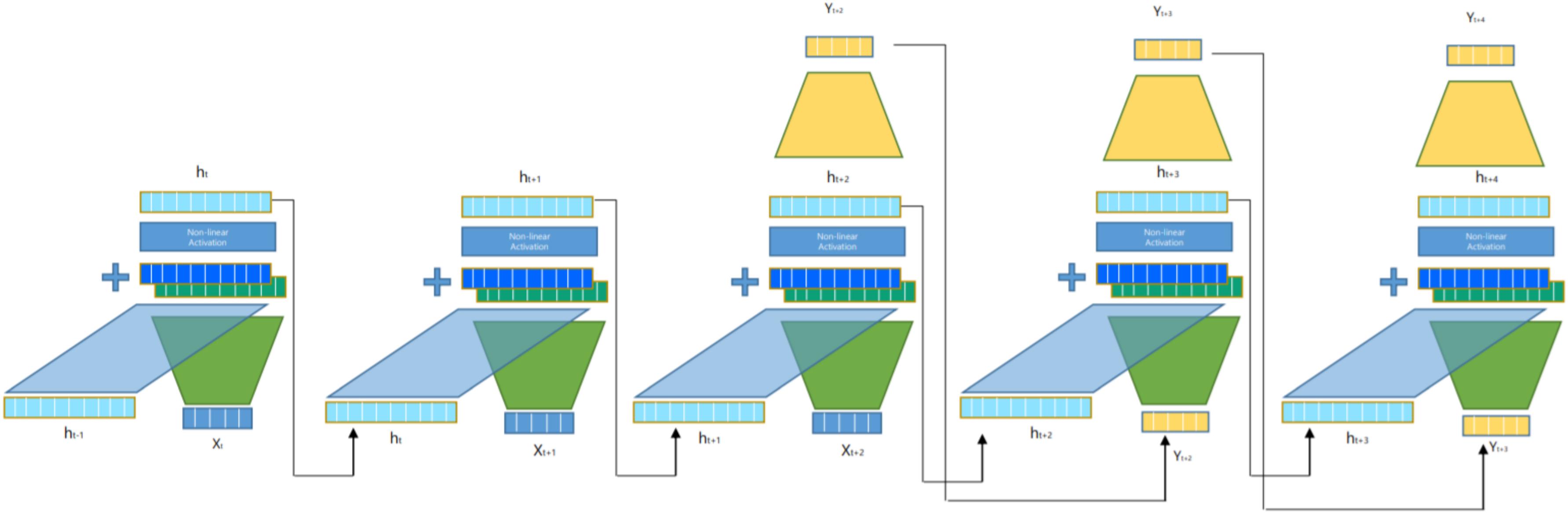
Recurrent Neural Network



many to many

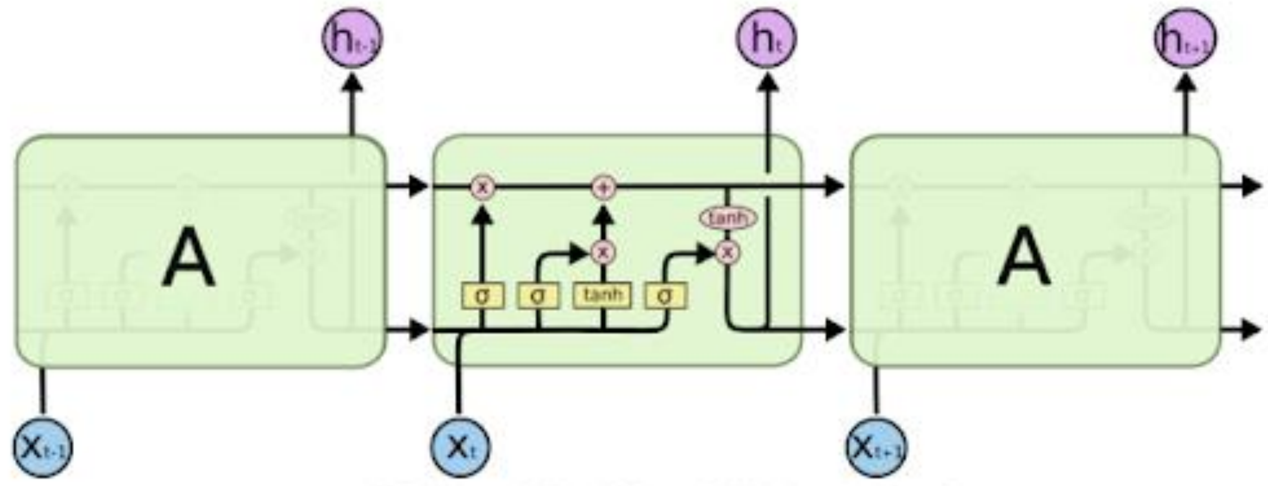
$$h_t = \tanh(Ux_t + Wh_{t-1})$$

$$o_t = \text{Softmax}(Vh_t)$$



Recurrent Neural Network

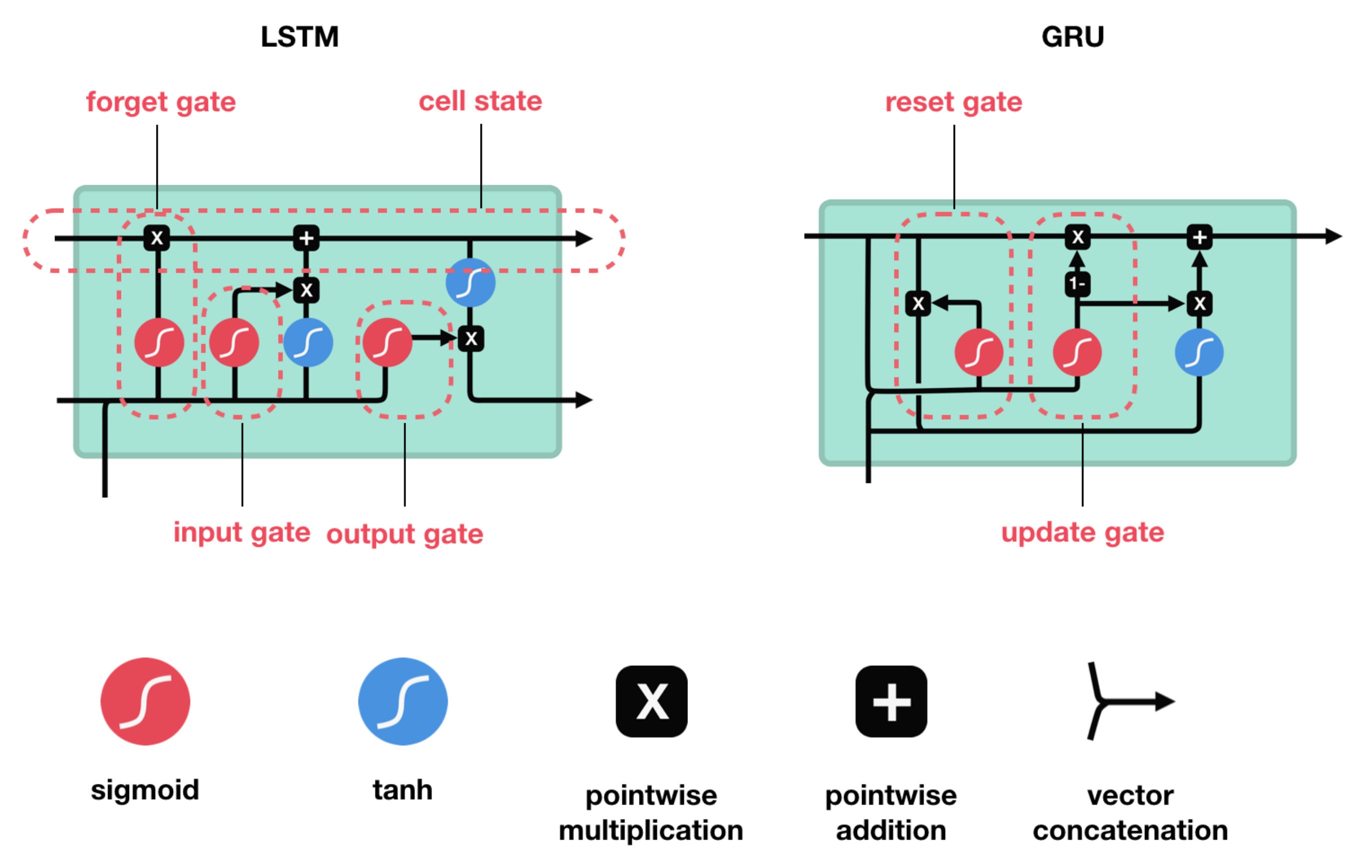
- LSTM
  - cell gate, input gate, output gate, forget gate



LSTM에 들어있는 4개의 상호작용하는 레이어가 있는 반복되는 모듈

Recurrent Neural Network

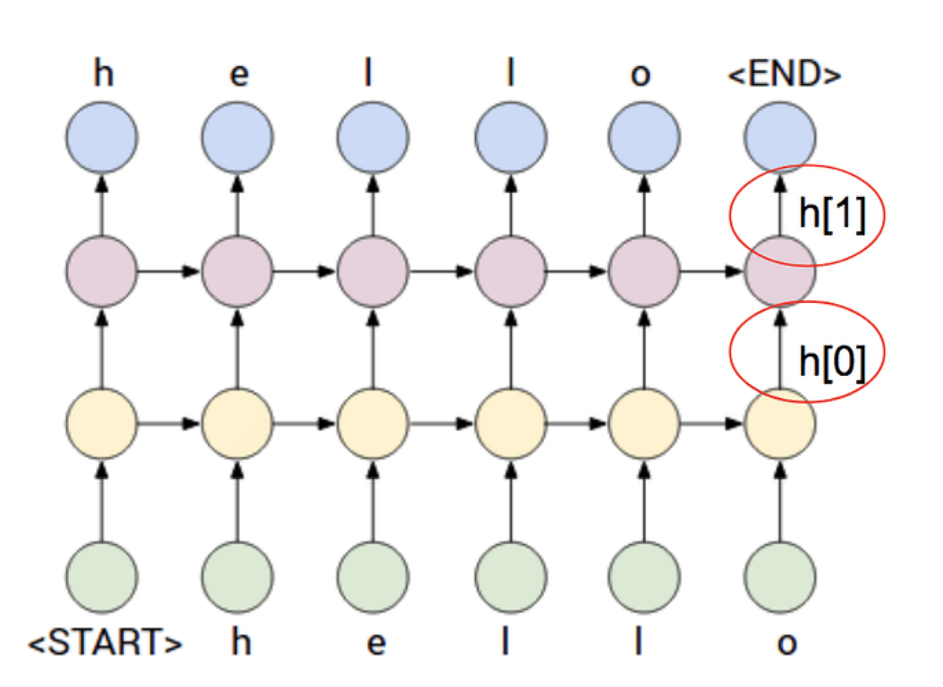
- GRU
  - Update & Reset gate



Recurrent Neural Network

- Stacked RNN

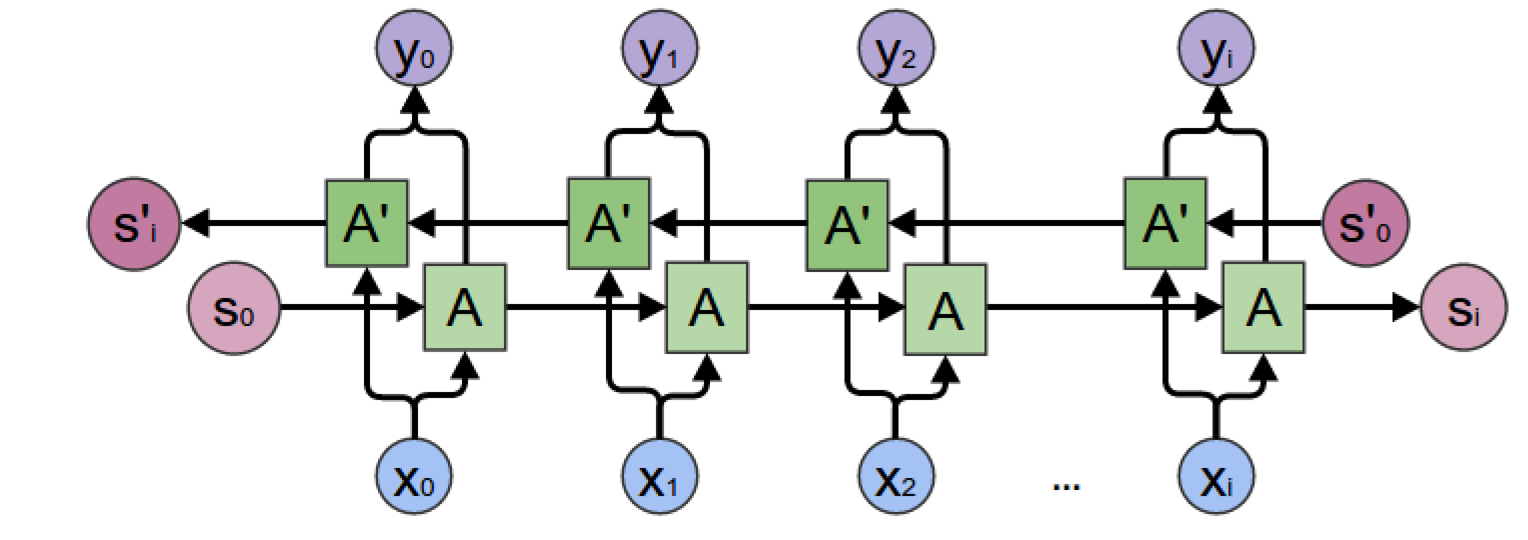
```
rnn = torch.nn.RNN(dic_size, hidden_size, batch_first=True, num_layers=2)
```



Recurrent Neural Network

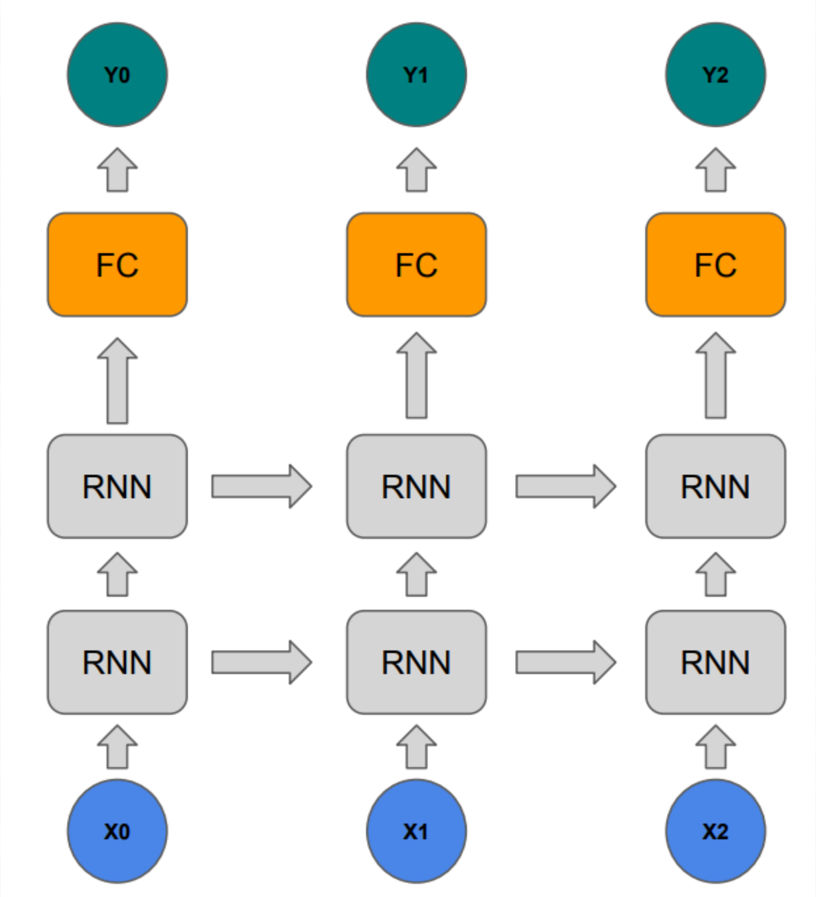
- Bidirectional LSTM

```
rnn = torch.nn.RNN(dic_size, hidden_size, batch_first=True,
bidirectional=True)
```



Practice1

- RNN + FC



Practice1

- RNN + FC

```
rnn = torch.nn.RNN(dic_size, hidden_size, batch_first=True)
FC = torch.nn.Linear(hidden_size, dic_size)
```

Practice2

- Language Model

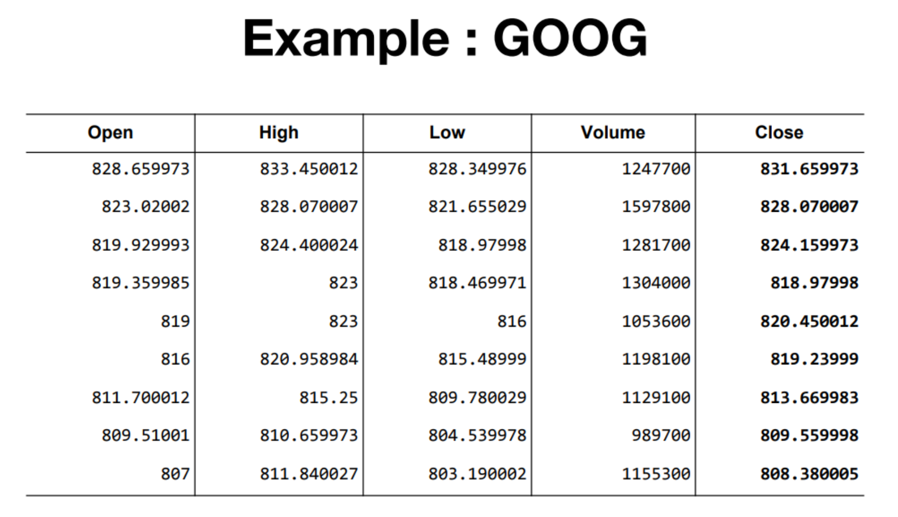
```
0 if you wan -> f you want
1 f you want ->  you want
2  you want  -> you want t
3 you want t -> ou want to
4 ou want to -> u want to
5 u want to  ->  want to b
6  want to b -> want to bu
7 want to bu -> ant to bui
8 ant to bui -> nt to buil
9 nt to buil -> t to build
10 t to build ->  to build
11  to build  -> to build a
12 to build a -> o build a
13 o build a  ->  build a s
14  build a s -> build a sh
15 build a sh -> uild a shi
16 uild a shi -> ild a ship
```

RNN for MNIST

- MNIST Dataset
  - Large data of handwritten digits that is commonly used for training various image processing systems and machine learning
  - It contains 60,000 training images and 10,000 testing images (28 X 28 pixel)

RNN for MNIST

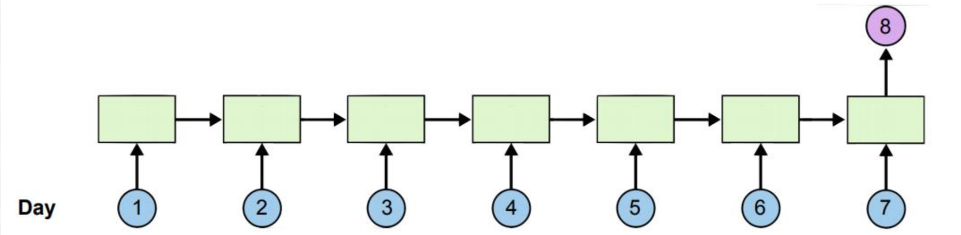
Stock Prediction



Example : GOOG

Open	High	Low	Volume	Close
828.659973	833.450012	828.349976	1247700	**831.659973**
823.02002	828.070007	821.655029	1597800	**828.070007**
819.929993	824.400024	818.97998	1281700	**824.159973**
819.359985	823	818.469971	1304000	**818.97998**
819	823	816	1053600	**820.450012**
816	820.958984	815.48999	1198100	**819.23999**
811.700012	815.25	809.780029	1129100	**813.669983**
809.51001	810.659973	804.539978	989700	**809.559998**
807	811.840027	803.190002	1155300	**808.380005**

Apply RNN : Many-to-One



Open	828.659973	823.02002	823.02002	819.929993	819	816	811.700012
High	833.450012	828.070007	824.400024	823	823	820.958984	815.25
Low	828.349976	821.655029	818.97998	818.469971	816	815.48999	809.780029
Volume	1247700	1597800	1281700	1304000	1053600	1198100	1129100
Close	831.659973	828.070007	824.159973	818.97998	820.450012	819.23999	813.669983

Stock Prediction

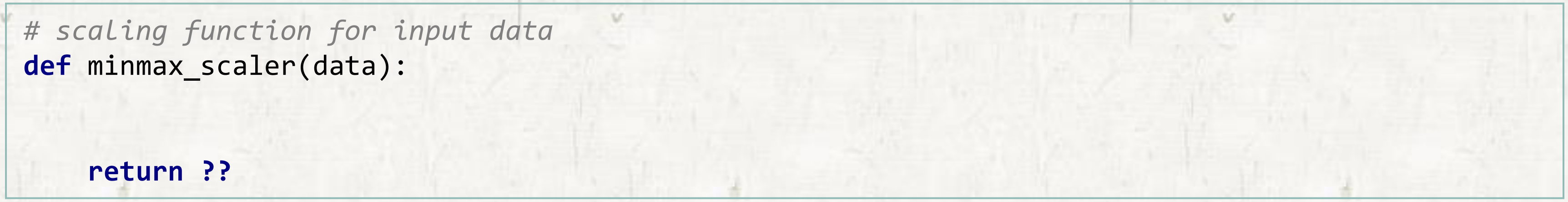
- Data Preprocessing (1)

```
%matplotlib inline

import torch
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt
```

Stock Prediction 문제1

Q. Min-max scaler를 구현하시오

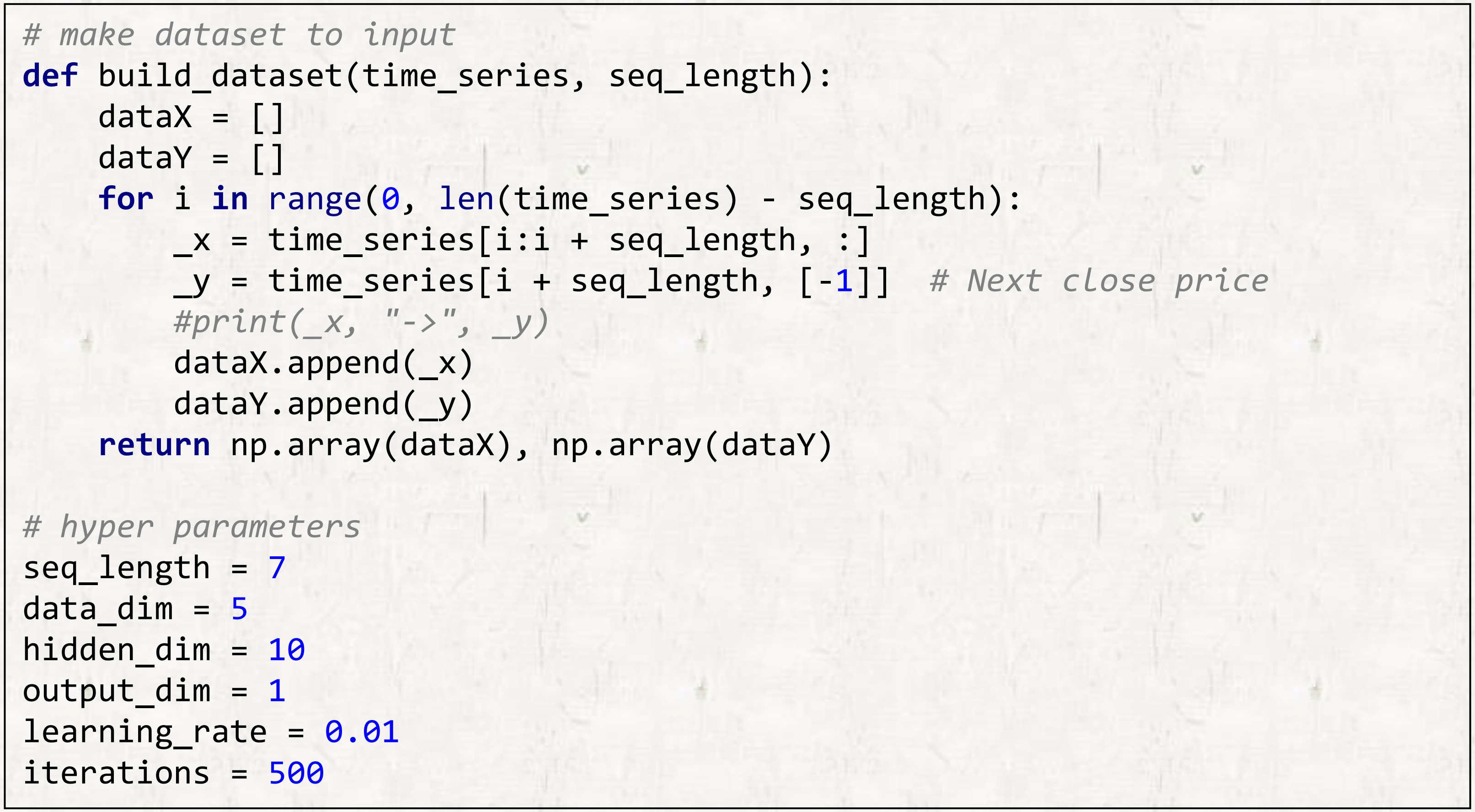
(x-min(x))/(max(x)-min(x))



```
# scaling function for input data
def minmax_scaler(data):



    return ??
```

Stock Prediction

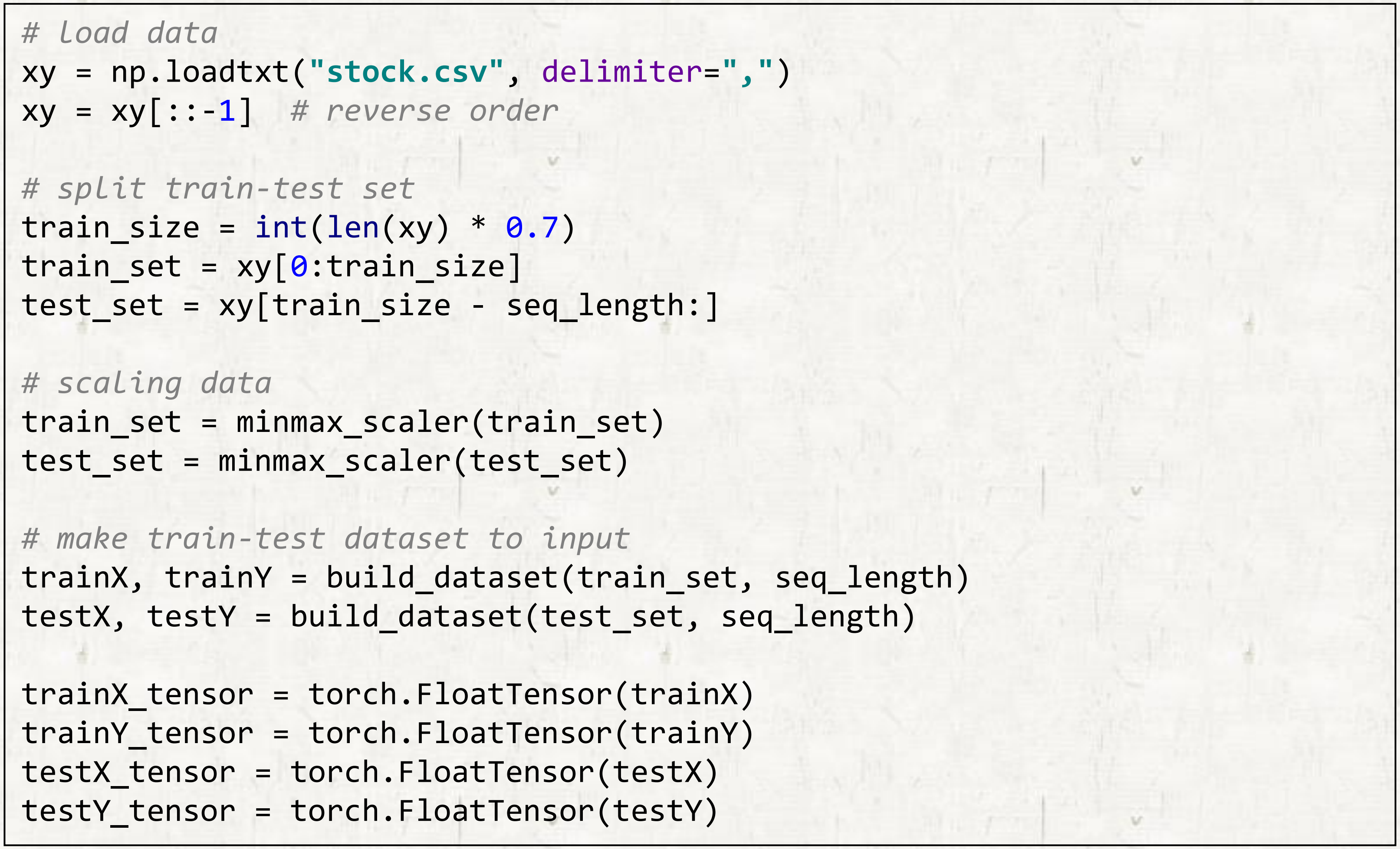
- Data Preprocessing (2)



```python
# make dataset to input
def build_dataset(time_series, seq_length):
    dataX = []
    dataY = []
    for i in range(0, len(time_series) - seq_length):
        _x = time_series[i:i + seq_length, :]
        _y = time_series[i + seq_length, [-1]]  # Next close price
        #print(_x, "->", _y)
        dataX.append(_x)
        dataY.append(_y)
    return np.array(dataX), np.array(dataY)

# hyper parameters
seq_length = 7
data_dim = 5
hidden_dim = 10
output_dim = 1
learning_rate = 0.01
iterations = 500
```

Data Preprocessing

- Data Preprocessing (3)



```
# load data
xy = np.loadtxt("stock.csv", delimiter=",")
xy = xy[::-1]  # reverse order

# split train-test set
train_size = int(len(xy) * 0.7)
train_set = xy[0:train_size]
test_set = xy[train_size - seq_length:]

# scaling data
train_set = minmax_scaler(train_set)
test_set = minmax_scaler(test_set)

# make train-test dataset to input
trainX, trainY = build_dataset(train_set, seq_length)
testX, testY = build_dataset(test_set, seq_length)

trainX_tensor = torch.FloatTensor(trainX)
trainY_tensor = torch.FloatTensor(trainY)
testX_tensor = torch.FloatTensor(testX)
testY_tensor = torch.FloatTensor(testY)
```

Stock Prediction 문제2

Q. LSTM과 layer로 이뤄진 다음 Net을 구현하시오

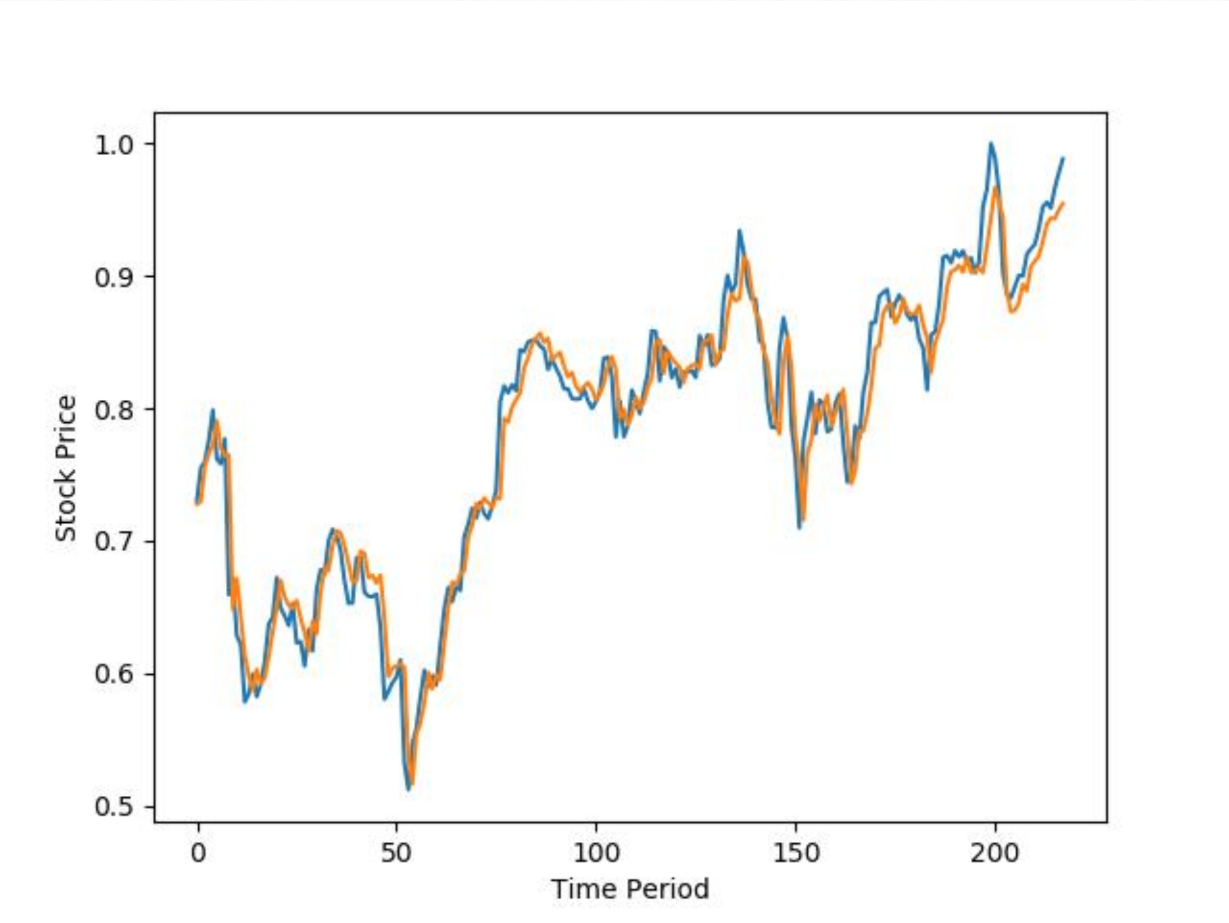
```
class Net(torch.nn.Module):
    ####Implement this code####

    ###########################

net = Net(data_dim, hidden_dim, output_dim, 1)
```

Stock Prediction 문제3

Q. 다음 학습 과정을 구현하시오

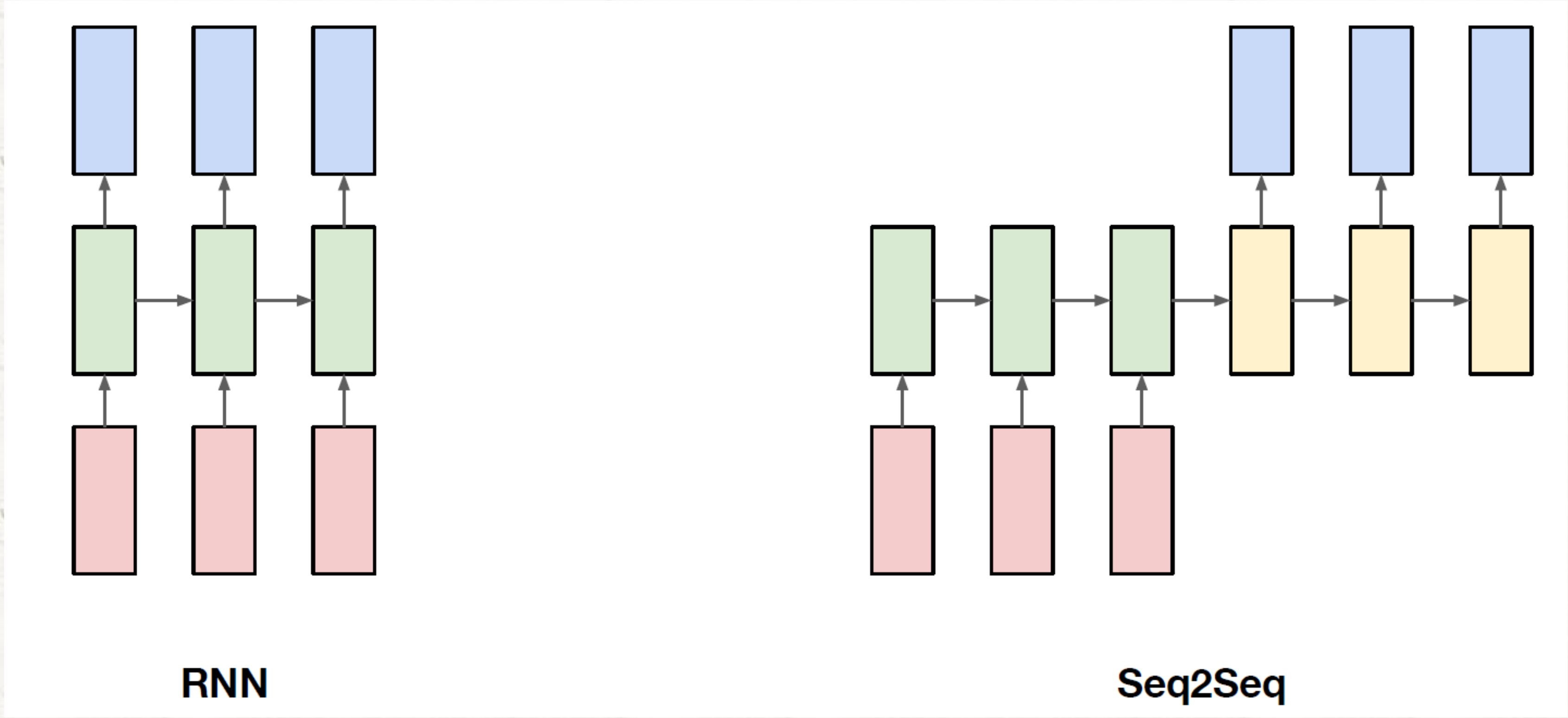
```
# loss & optimizer setting
criterion = torch.nn.MSELoss()
optimizer = optim.Adam(net.parameters(), lr=learning_rate)

# start training
for i in range(iterations):
    ####Implement this code####

    ###########################

# Visualization
plt.plot(testY)
plt.plot(net(testX_tensor).data.numpy())
plt.legend(['original', 'prediction'])
plt.show()
```

Stock Prediction



Seq2Seq

Seq2Seq



Pytorch Summary

- Torch reference
  - https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html