Tensorflow를 이용한 CNN, RNN 실습

부산대학교

Information Security & IoT Lab

조동근

2017.11.29

목 차

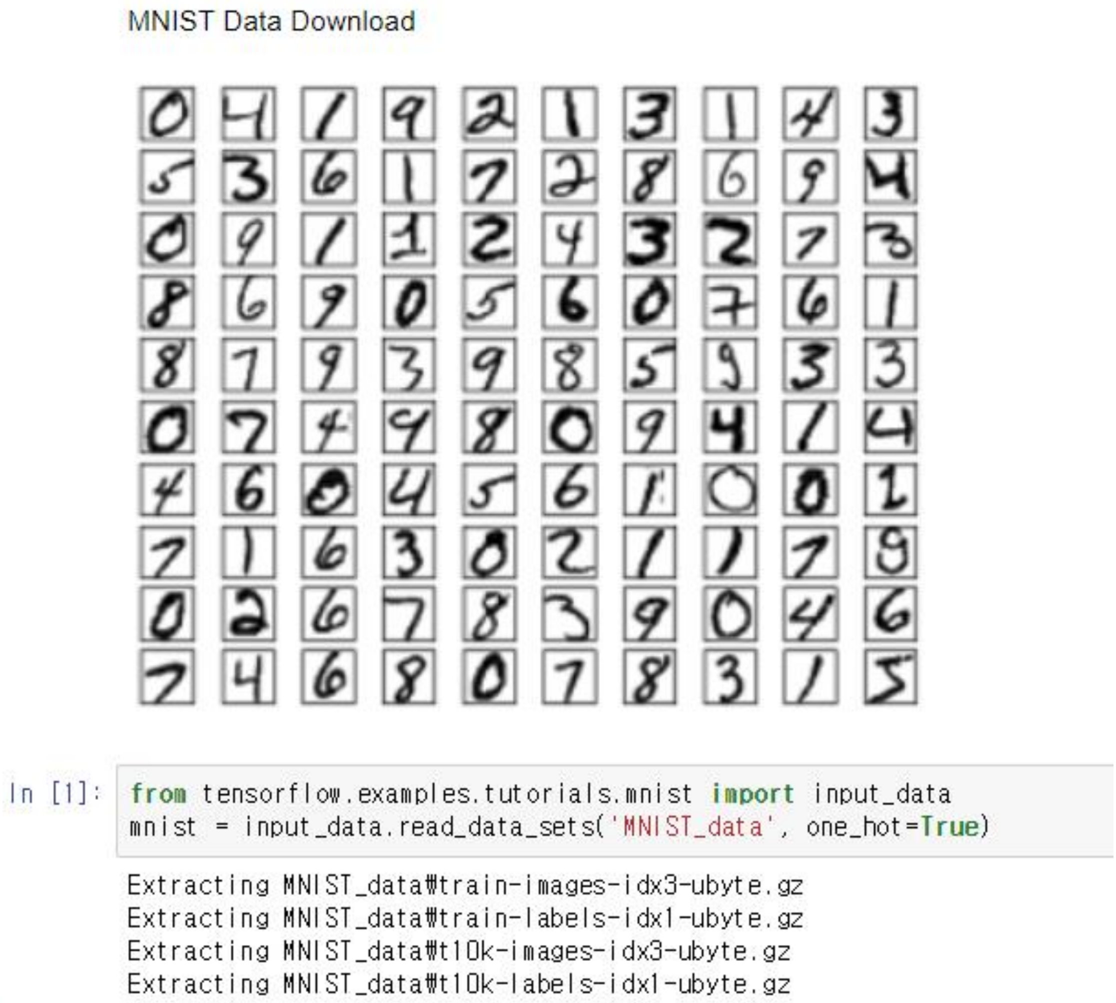
1. CNN 실습

2. RNN 실습

1. CNN 실습

CNN 실습

- 텐서플로우 간단 실습(1) - CNN
  - MNIST 데이터를 이용하여 0~9 숫자를 분류

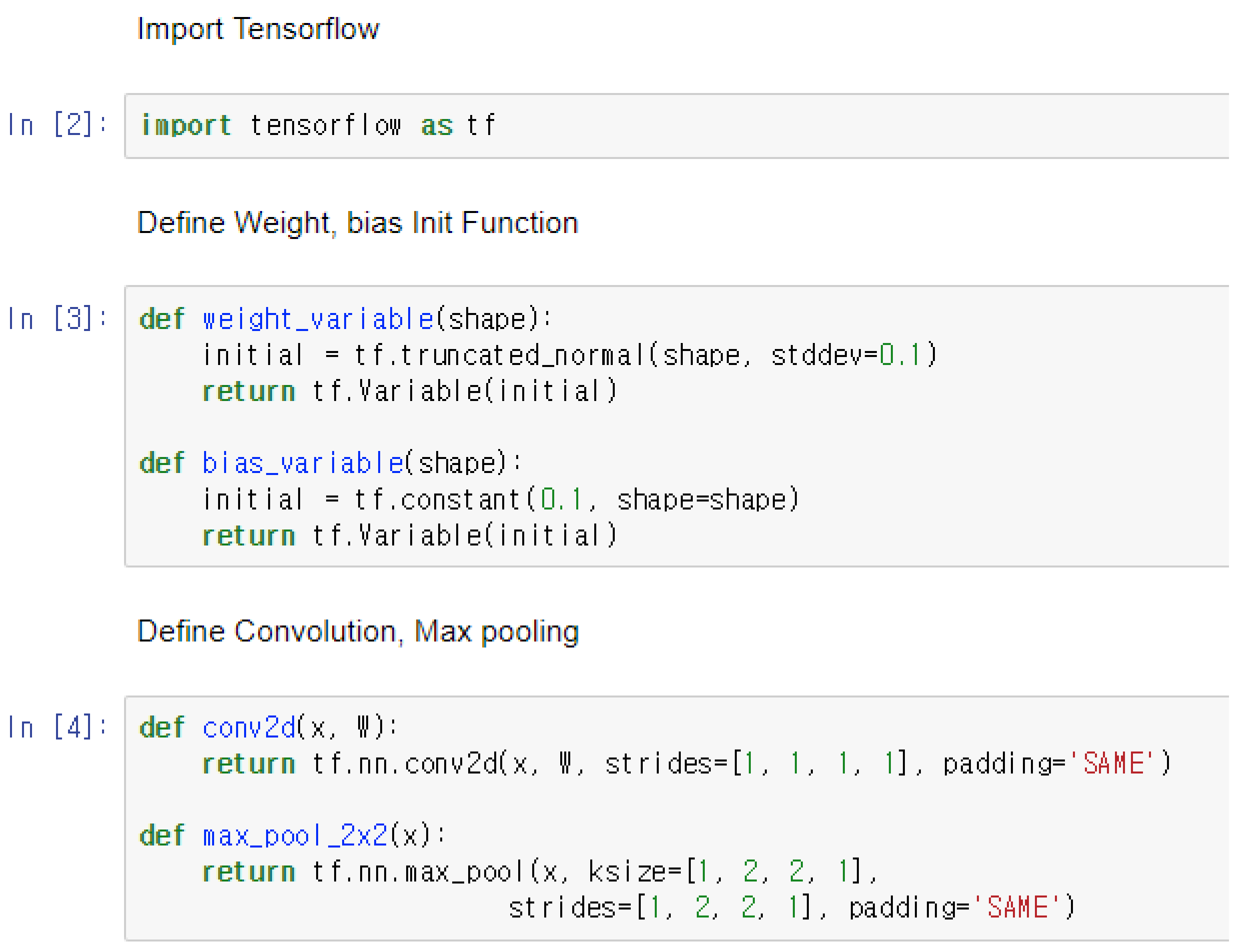


MNIST Data Download

```
In [1]: from tensorflow.examples.tutorials.mnist import input_data
        mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
```

```
Extracting MNIST_data₩train-images-idx3-ubyte.gz
Extracting MNIST_data₩train-labels-idx1-ubyte.gz
Extracting MNIST_data₩t10k-images-idx3-ubyte.gz
Extracting MNIST_data₩t10k-labels-idx1-ubyte.gz
```

- 텐서플로우 간단 실습(2) - CNN



Import Tensorflow

```
In [2]: import tensorflow as tf
```

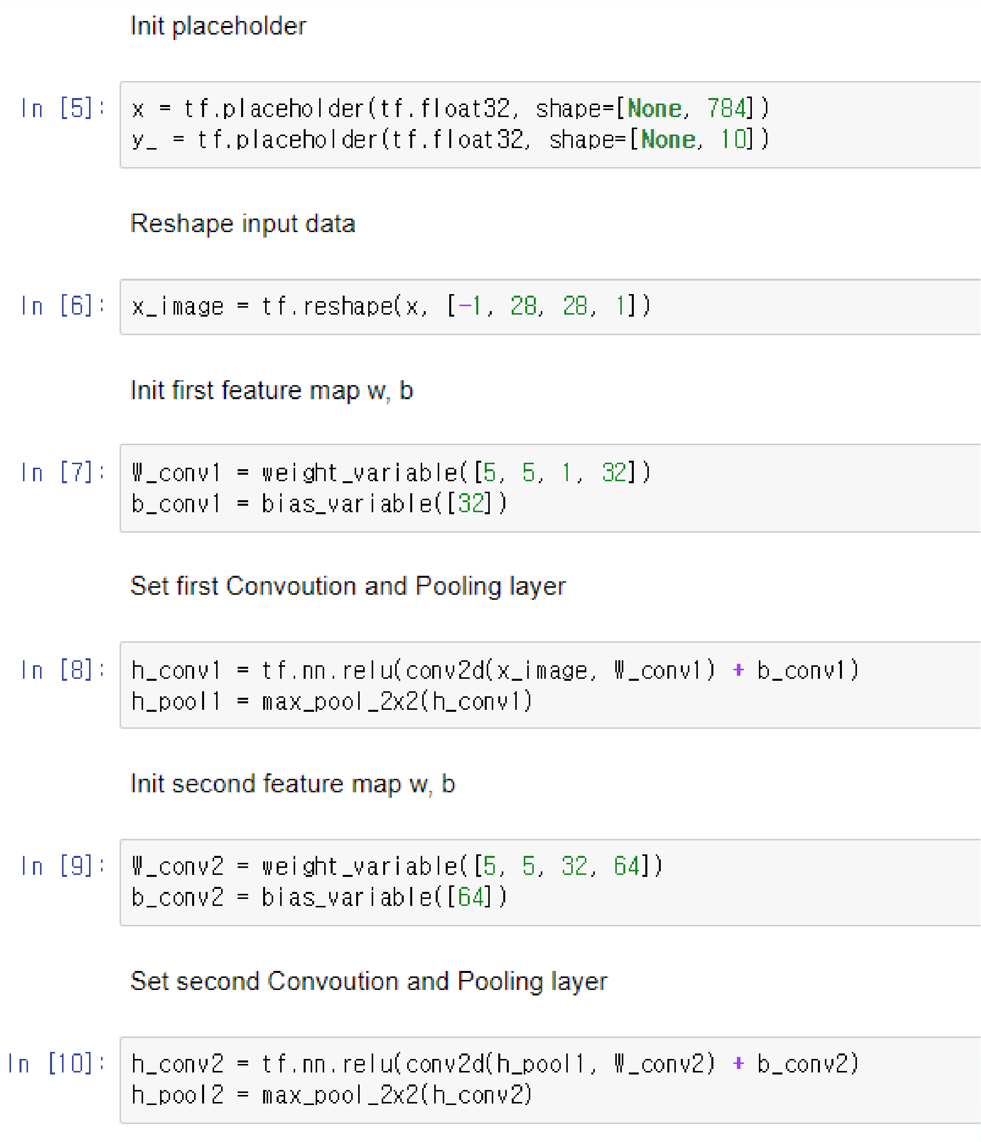
Define Weight, bias Init Function

```
In [3]: def weight_variable(shape):
            initial = tf.truncated_normal(shape, stddev=0.1)
            return tf.Variable(initial)

        def bias_variable(shape):
            initial = tf.constant(0.1, shape=shape)
            return tf.Variable(initial)
```

Define Convolution, Max pooling

```
In [4]: def conv2d(x, W):
            return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

        def max_pool_2x2(x):
            return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                                  strides=[1, 2, 2, 1], padding='SAME')
```

- # 텐서플로우 간단 실습(3) - CNN



Init placeholder

```
In [5]: x = tf.placeholder(tf.float32, shape=[None, 784])
        y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

Reshape input data

```
In [6]: x_image = tf.reshape(x, [-1, 28, 28, 1])
```

Init first feature map w, b

```
In [7]: W_conv1 = weight_variable([5, 5, 1, 32])
        b_conv1 = bias_variable([32])
```

Set first Convoution and Pooling layer

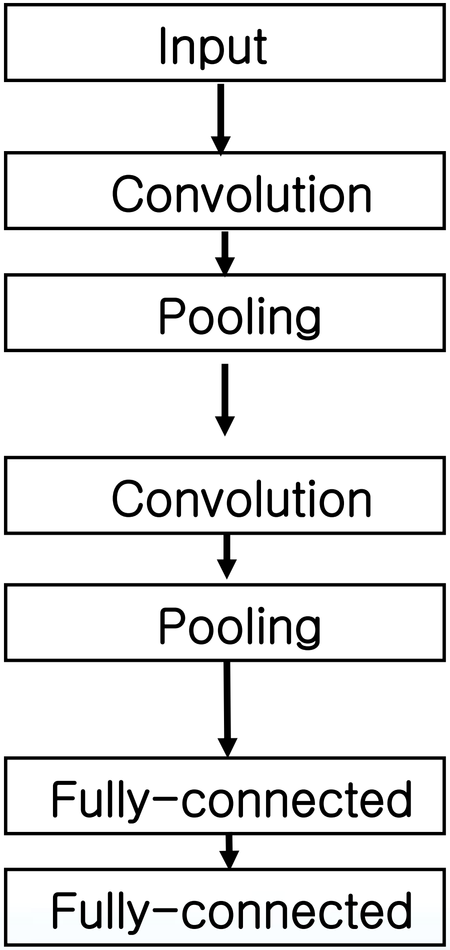
```
In [8]: h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
        h_pool1 = max_pool_2x2(h_conv1)
```

Init second feature map w, b

```
In [9]: W_conv2 = weight_variable([5, 5, 32, 64])
        b_conv2 = bias_variable([64])
```

Set second Convoution and Pooling layer

```
In [10]: h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
         h_pool2 = max_pool_2x2(h_conv2)
```



Input

Convolution

Pooling

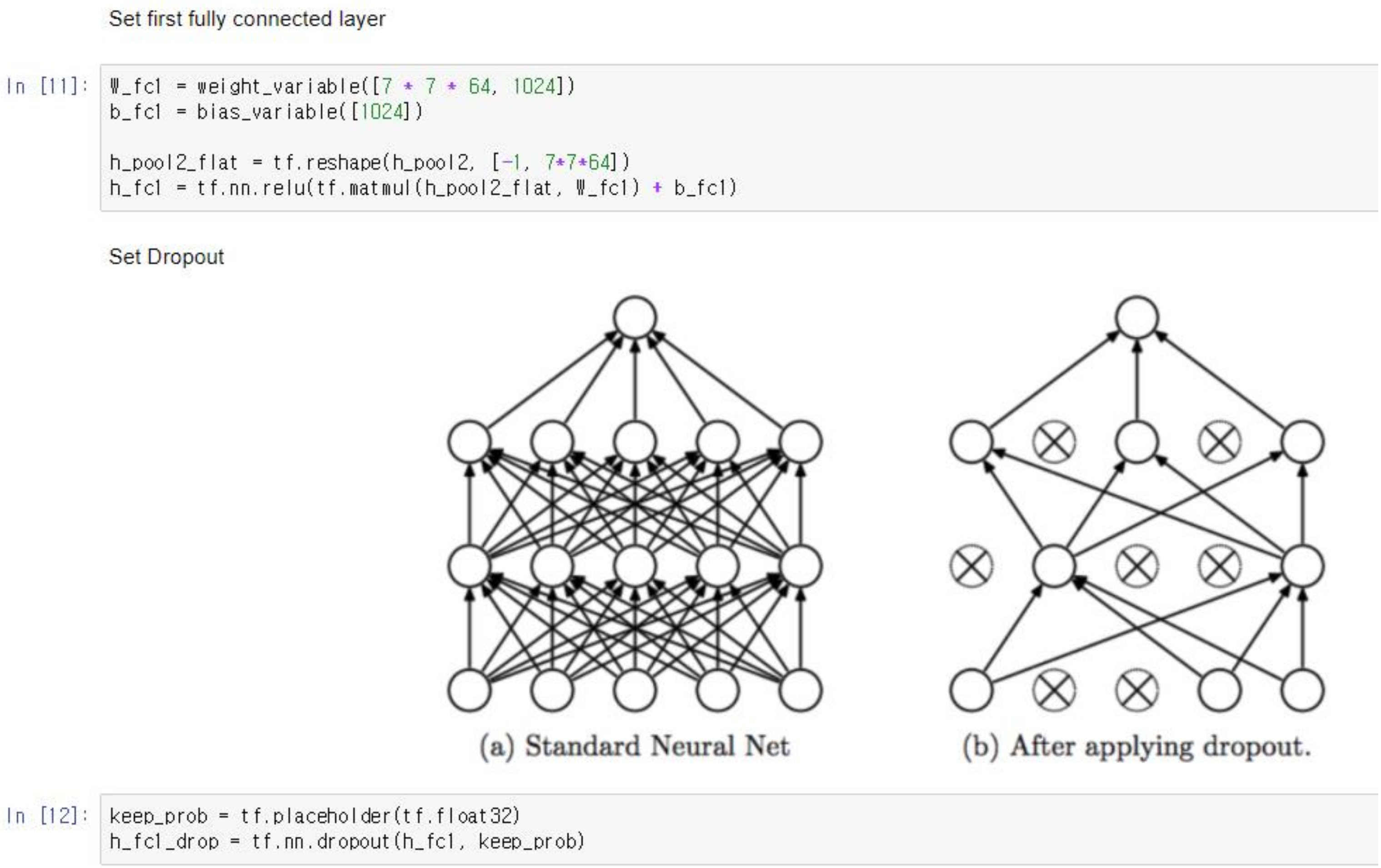
Convolution

Pooling

Fully-connected

Fully-connected

- 텐서플로우 간단 실습(4) - CNN
  - Dropout은 Training Data에 대한 과적용(Overfitting)을 완화하는 역할을 함



Set first fully connected layer

```
In [11]: W_fc1 = weight_variable([7 * 7 * 64, 1024])
         b_fc1 = bias_variable([1024])

         h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
         h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

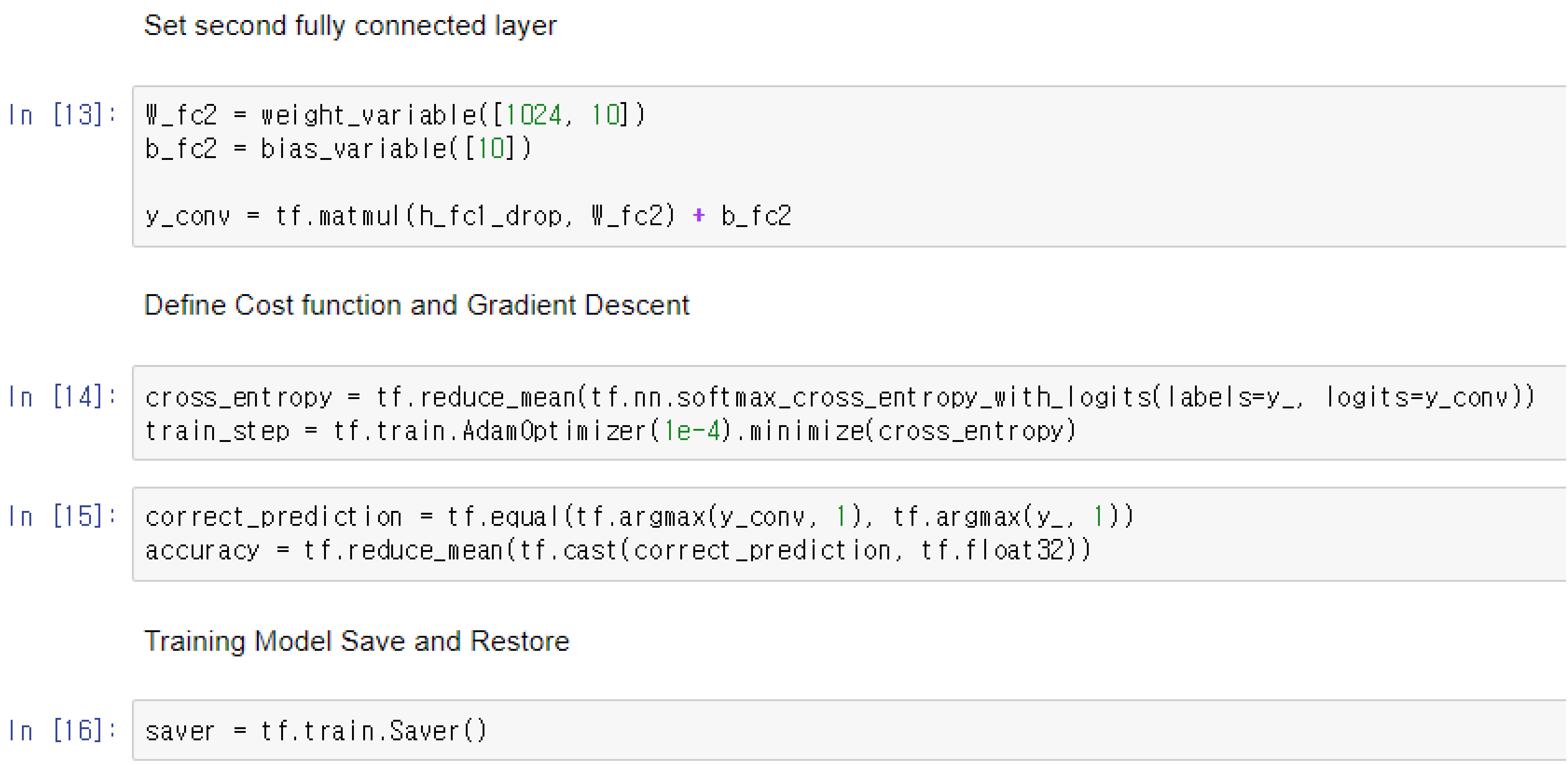
Set Dropout

(a) Standard Neural Net

(b) After applying dropout.

```
In [12]: keep_prob = tf.placeholder(tf.float32)
         h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

텐서플로우 간단 실습(5) - CNN



Set second fully connected layer

In [13]:
```
W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
```

Define Cost function and Gradient Descent

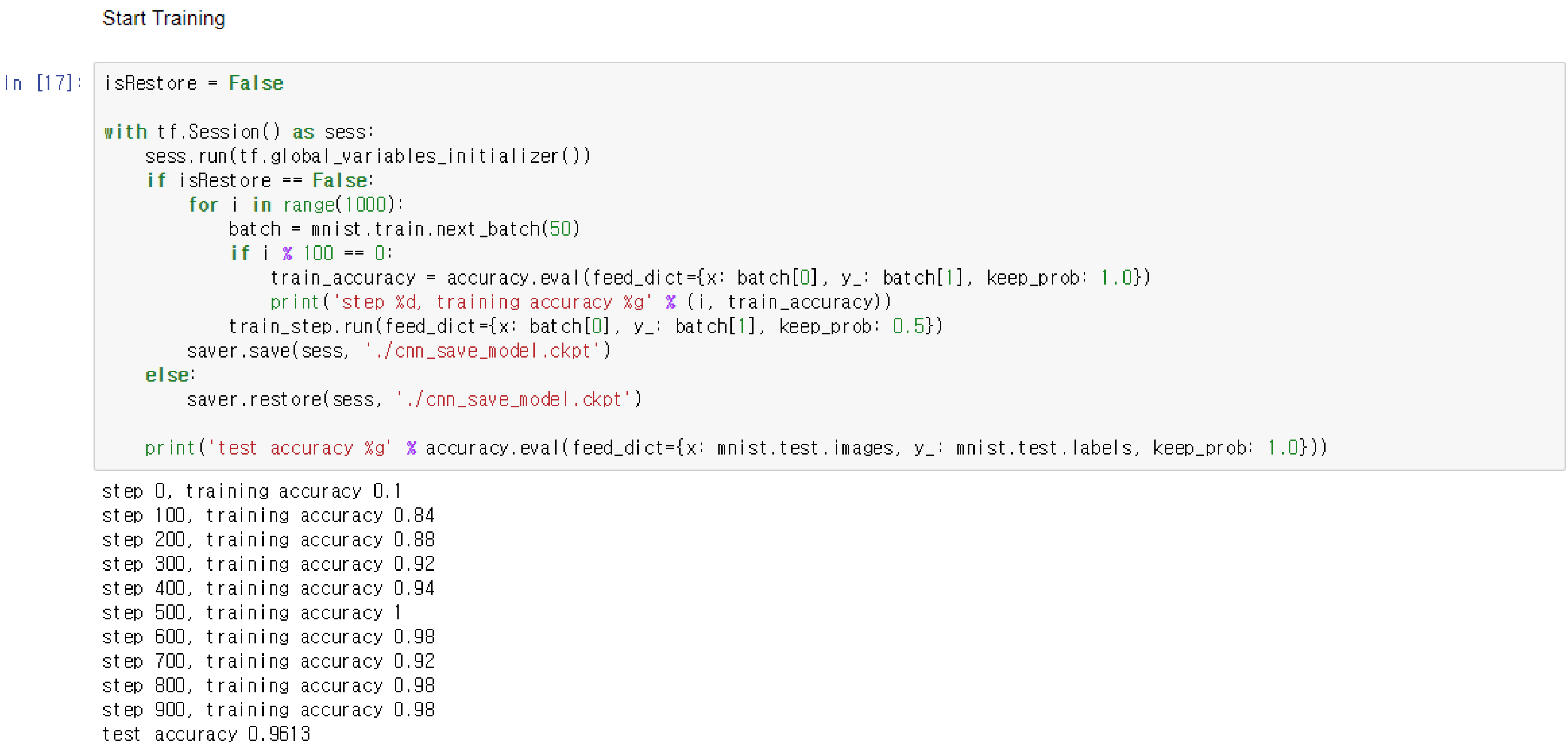
In [14]:
```
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
```

In [15]:
```
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Training Model Save and Restore

In [16]:
```
saver = tf.train.Saver()
```

• 텐서플로우 간단 실습(6) - CNN



Start Training

In [17]:

```
isRestore = False

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    if isRestore == False:
        for i in range(1000):
            batch = mnist.train.next_batch(50)
            if i % 100 == 0:
                train_accuracy = accuracy.eval(feed_dict={x: batch[0], y_: batch[1], keep_prob: 1.0})
                print('step %d, training accuracy %g' % (i, train_accuracy))
            train_step.run(feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})
        saver.save(sess, './cnn_save_model.ckpt')
    else:
        saver.restore(sess, './cnn_save_model.ckpt')

    print('test accuracy %g' % accuracy.eval(feed_dict={x: mnist.test.images, y_: mnist.test.labels, keep_prob: 1.0}))
```

```
step 0, training accuracy 0.1
step 100, training accuracy 0.84
step 200, training accuracy 0.88
step 300, training accuracy 0.92
step 400, training accuracy 0.94
step 500, training accuracy 1
step 600, training accuracy 0.98
step 700, training accuracy 0.92
step 800, training accuracy 0.98
step 900, training accuracy 0.98
test accuracy 0.9613
```

2. RNN 실습

RNN 실습

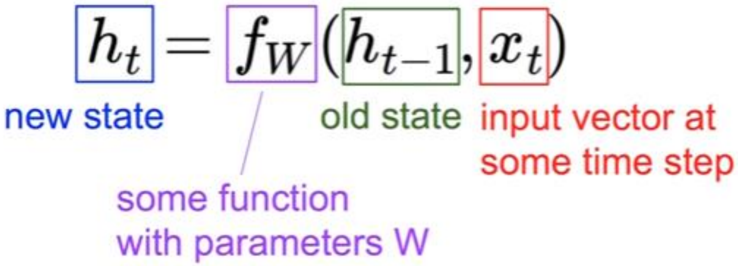
• 텐서플로우 간단 실습(1) - RNN

What is RNN? Recurrent(반복적인) Neural Network.

반복적인 데이터, 즉 순차적인 데이터를 학습하는데 특화되어 발전한 인공신경망.

현재 상태를 계산하기 위해 이전 상태가 필요하다.

예제를 통해 확인해봅시다.

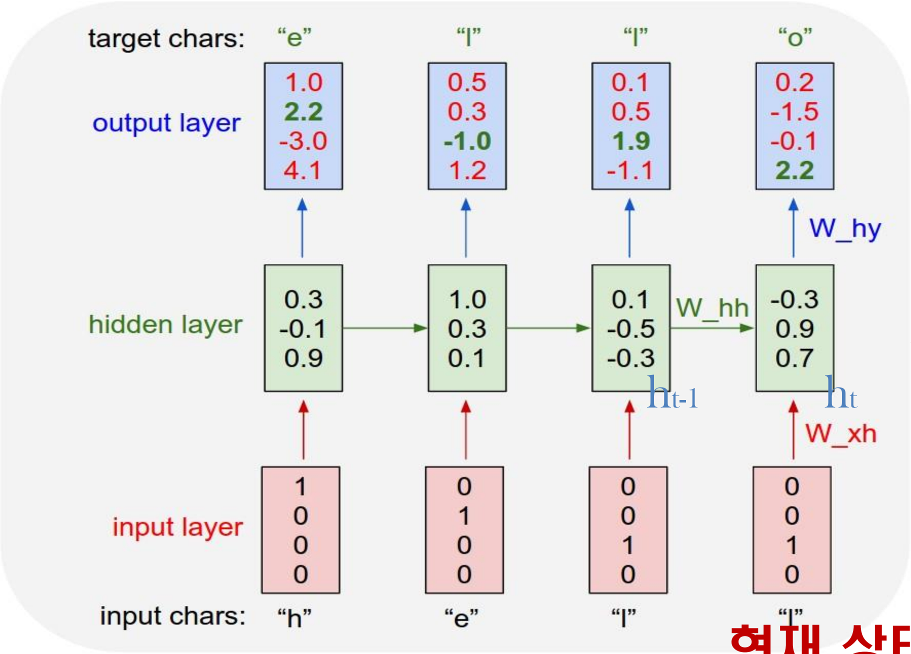


$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state
old state
input vector at some time step
some function with parameters W

RNN 실습

- **텐서플로우 간단 실습(2) - RNN example**
  - hello 학습시키기 - hell이 입력되었을 때 ello를 예측하는 RNN모델



상태. (hidden layer)

현재 상태 계산에 이전 상태가 쓰인다.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

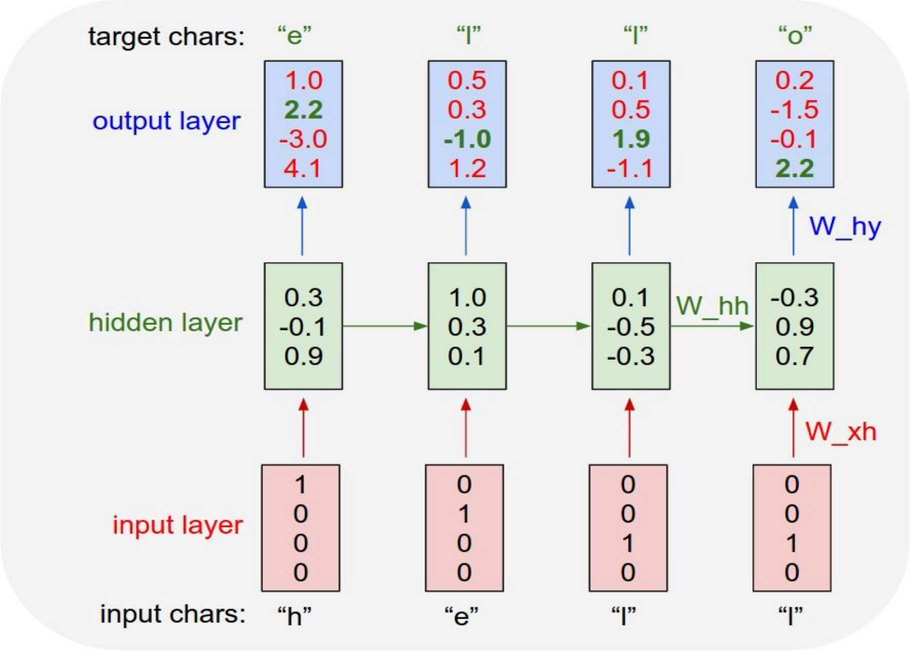
New state

Old state

Input vector
at some
time step

Tan함수는 sigmoid 함수 중 하나로 기존 sigmoid가 0~1을 반환한다면 tan함수는 -1에서 1을 반환.
즉 현재 상태를 가리키는 hidden layer의 값은 -1~1사이의 값이 된다.

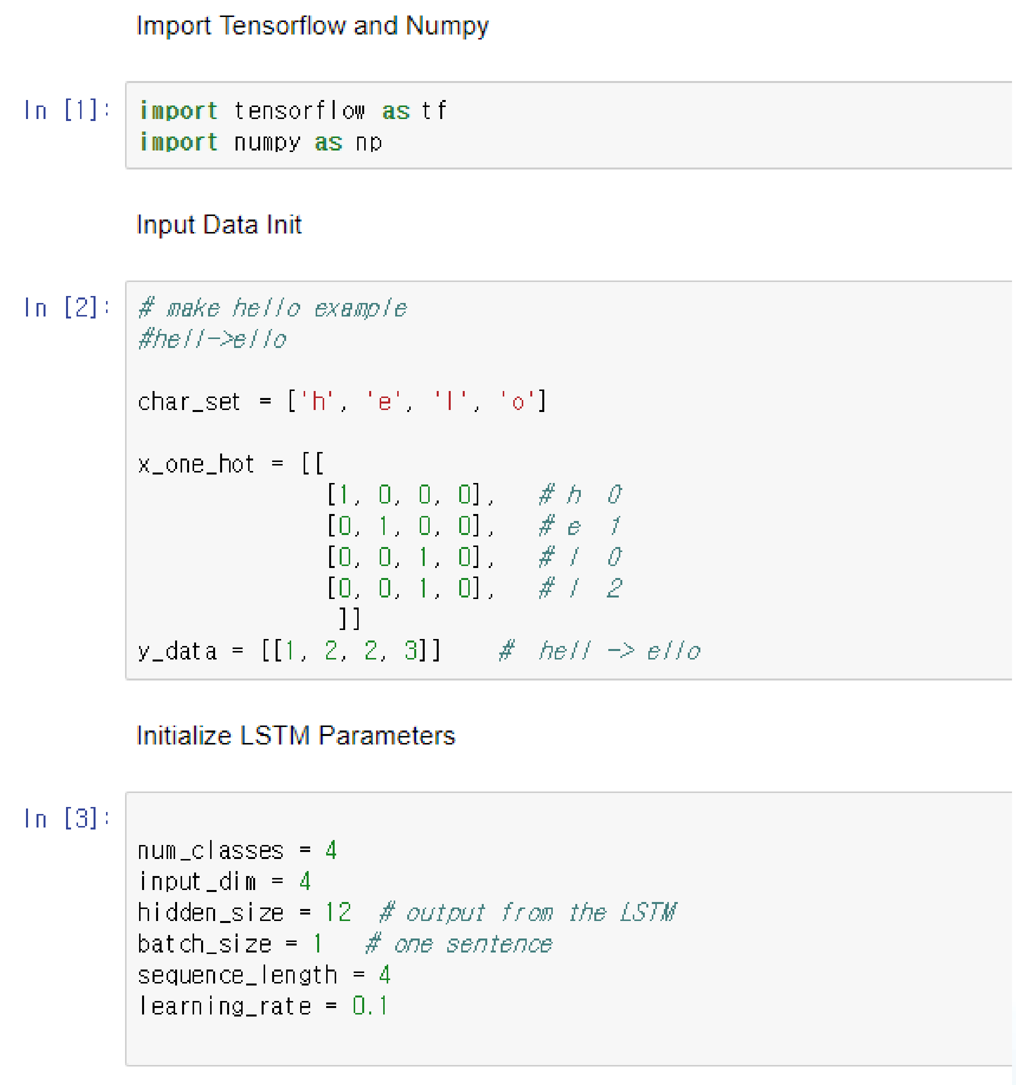
- **텐서플로우 간단 실습(3) - RNN example**
  - hello 학습시키기 - hell이 입력되었을 때 ello를 예측하는 RNN모델



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

현재 상태값(Hidden layer)을 계산하기 위해선
Whh, Wxh, Why 를 계산해야 한다.
(텐서플로우는 이 계산을 대부분 알아서 진행하여, 사용자가 직접 구현할 필요가 없다.)
그럼 소스코드를 통해 확인해봅시다.

텐서플로우 간단 실습(4) - RNN source code



Import Tensorflow and Numpy

In [1]:
```
import tensorflow as tf
import numpy as np
```

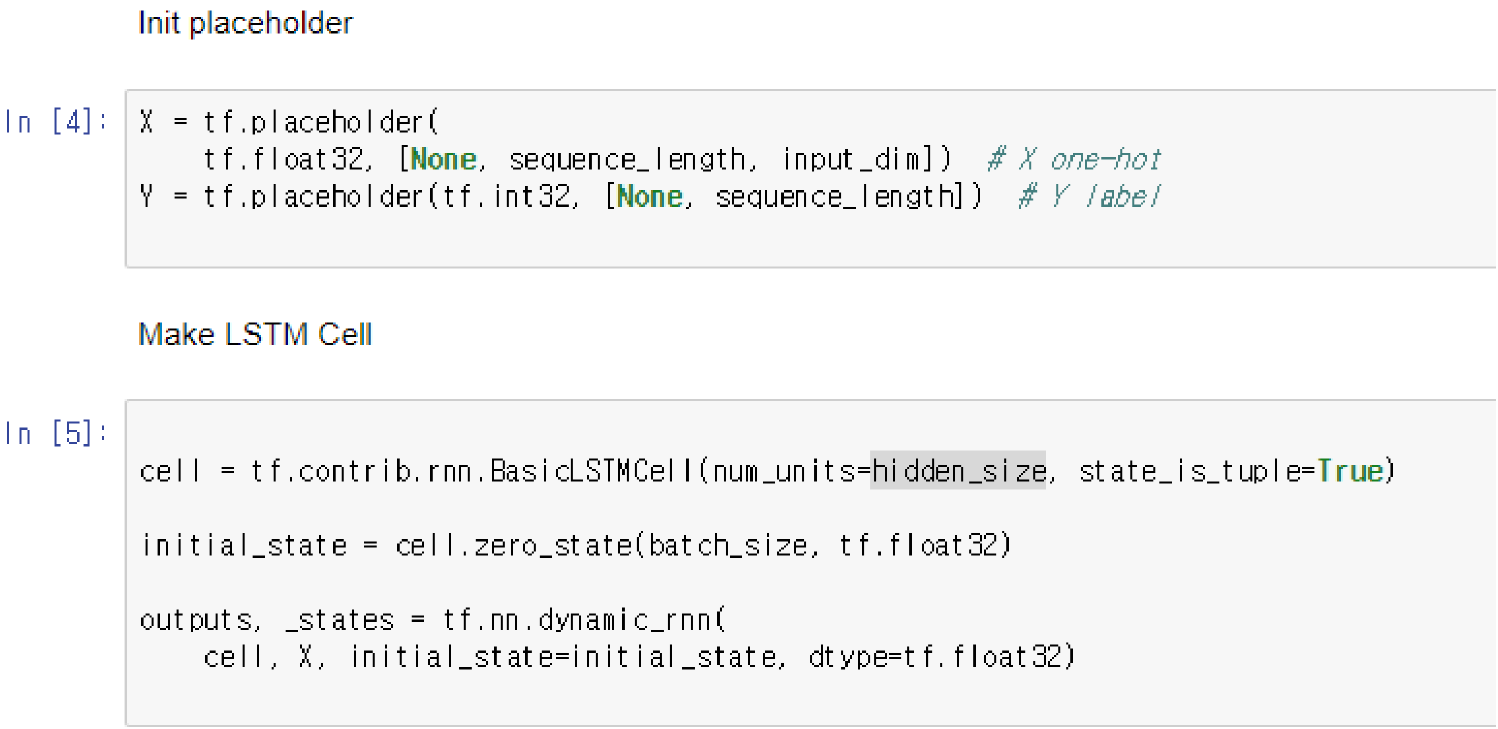
Input Data Init

In [2]:
```
# make hello example
#hell->ello

char_set = ['h', 'e', 'l', 'o']

x_one_hot = [[
              [1, 0, 0, 0],   # h  0
              [0, 1, 0, 0],   # e  1
              [0, 0, 1, 0],   # l  0
              [0, 0, 1, 0],   # l  2
               ]]
y_data = [[1, 2, 2, 3]]    #  hell -> ello
```

Initialize LSTM Parameters

In [3]:
```

num_classes = 4
input_dim = 4
hidden_size = 12  # output from the LSTM
batch_size = 1   # one sentence
sequence_length = 4
learning_rate = 0.1
```

텐서플로우 간단 실습(5) - RNN source code



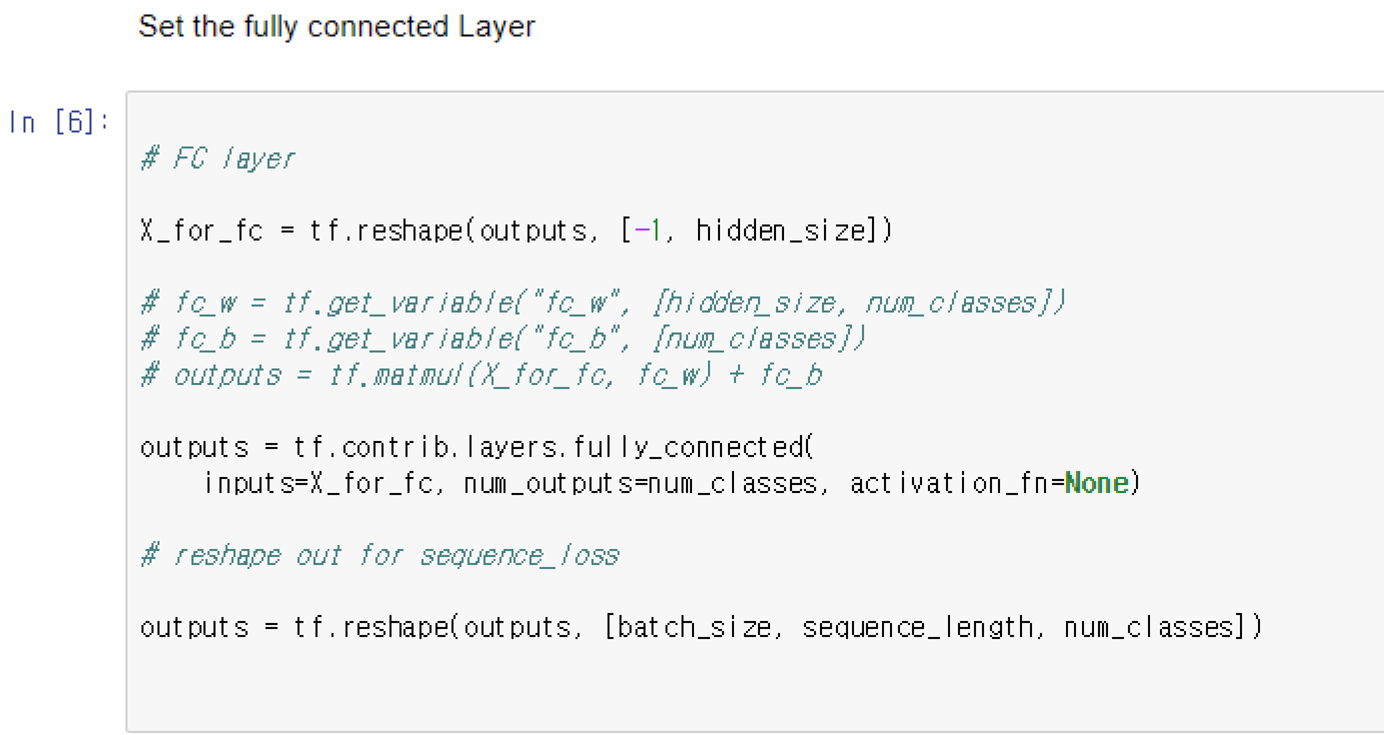
Init placeholder

```
In [4]: X = tf.placeholder(
            tf.float32, [None, sequence_length, input_dim])  # X one-hot
        Y = tf.placeholder(tf.int32, [None, sequence_length])  # Y label
```

Make LSTM Cell

```
In [5]:

        cell = tf.contrib.rnn.BasicLSTMCell(num_units=hidden_size, state_is_tuple=True)

        initial_state = cell.zero_state(batch_size, tf.float32)

        outputs, _states = tf.nn.dynamic_rnn(
            cell, X, initial_state=initial_state, dtype=tf.float32)
```

텐서플로우 간단 실습(6) - RNN source code

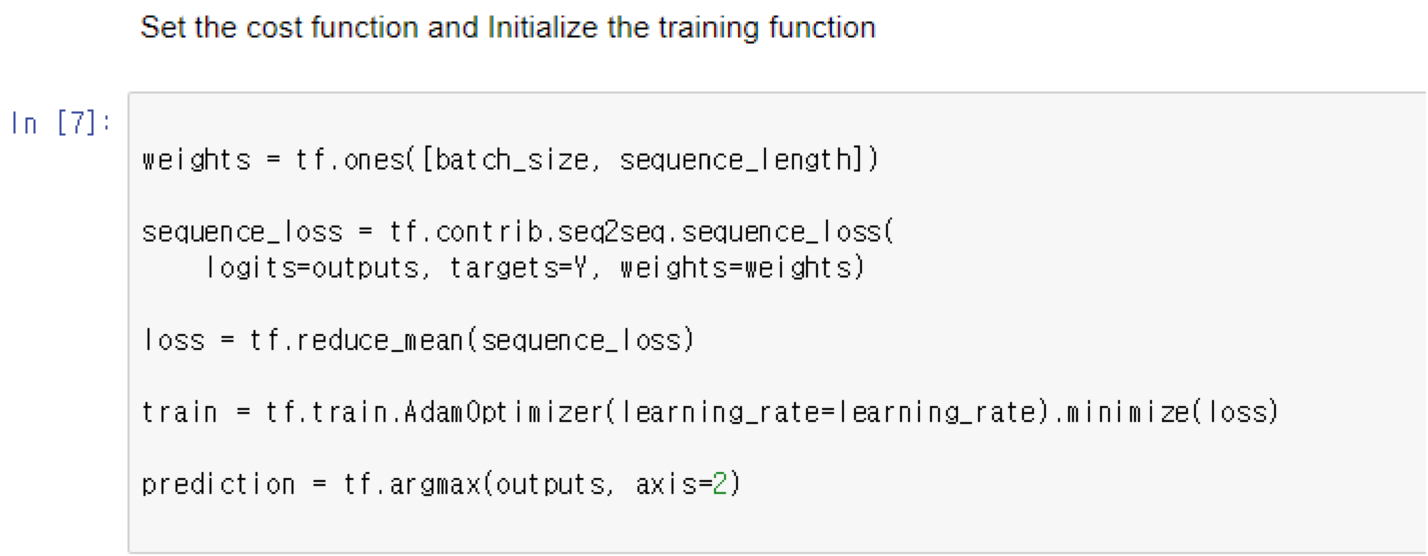


Set the fully connected Layer

In [6]:

```
# FC layer

X_for_fc = tf.reshape(outputs, [-1, hidden_size])

# fc_w = tf.get_variable("fc_w", [hidden_size, num_classes])
# fc_b = tf.get_variable("fc_b", [num_classes])
# outputs = tf.matmul(X_for_fc, fc_w) + fc_b

outputs = tf.contrib.layers.fully_connected(
    inputs=X_for_fc, num_outputs=num_classes, activation_fn=None)

# reshape out for sequence_loss

outputs = tf.reshape(outputs, [batch_size, sequence_length, num_classes])
```

- 텐서플로우 간단 실습(7) - RNN source code



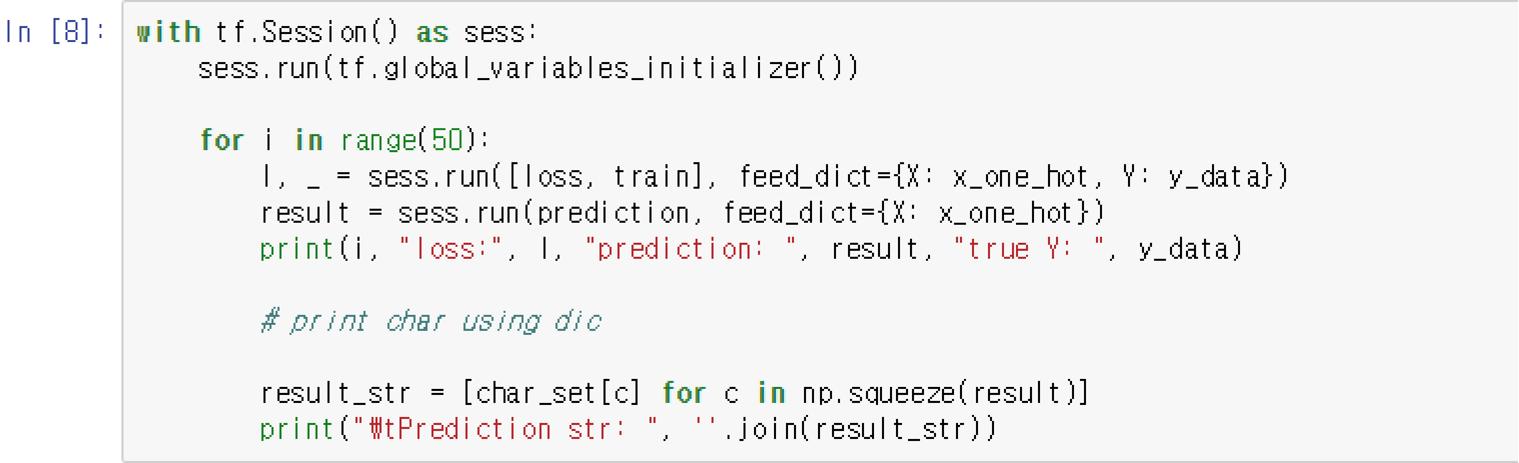
Set the cost function and Initialize the training function

In [7]:

```
weights = tf.ones([batch_size, sequence_length])

sequence_loss = tf.contrib.seq2seq.sequence_loss(
    logits=outputs, targets=Y, weights=weights)

loss = tf.reduce_mean(sequence_loss)

train = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)

prediction = tf.argmax(outputs, axis=2)
```

- 텐서플로우 간단 실습(8) - RNN source code

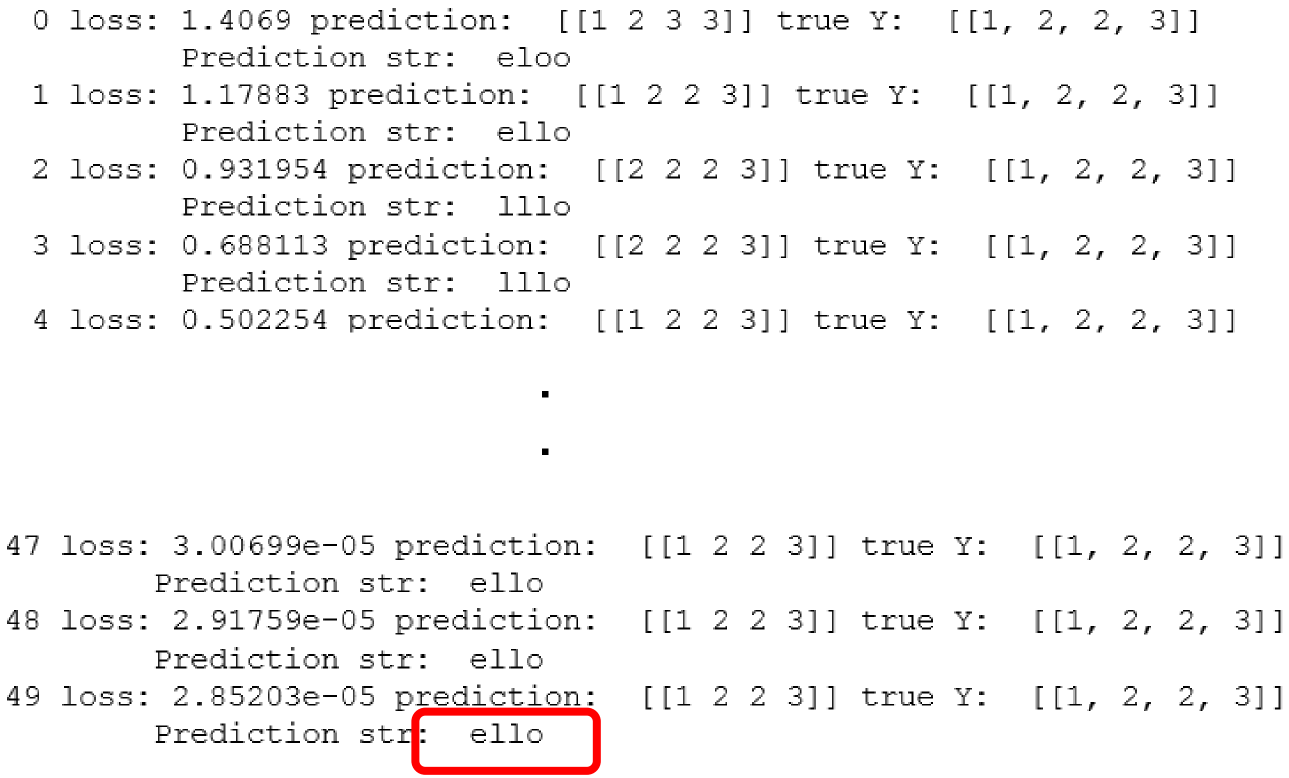
Start Training



In [8]:

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for i in range(50):
        l, _ = sess.run([loss, train], feed_dict={X: x_one_hot, Y: y_data})
        result = sess.run(prediction, feed_dict={X: x_one_hot})
        print(i, "loss:", l, "prediction: ", result, "true Y: ", y_data)

        # print char using dic

        result_str = [char_set[c] for c in np.squeeze(result)]
        print("₩tPrediction str: ", ''.join(result_str))
```

- 텐서플로우 간단 실습(9) - RNN source code



```
 0 loss: 1.4069 prediction:  [[1 2 3 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  eloo
 1 loss: 1.17883 prediction:  [[1 2 2 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  ello
 2 loss: 0.931954 prediction:  [[2 2 2 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  lllo
 3 loss: 0.688113 prediction:  [[2 2 2 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  lllo
 4 loss: 0.502254 prediction:  [[1 2 2 3]] true Y:  [[1, 2, 2, 3]]

                          .

                          .

47 loss: 3.00699e-05 prediction:  [[1 2 2 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  ello
48 loss: 2.91759e-05 prediction:  [[1 2 2 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  ello
49 loss: 2.85203e-05 prediction:  [[1 2 2 3]] true Y:  [[1, 2, 2, 3]]
         Prediction str:  ello
```

Thank you!