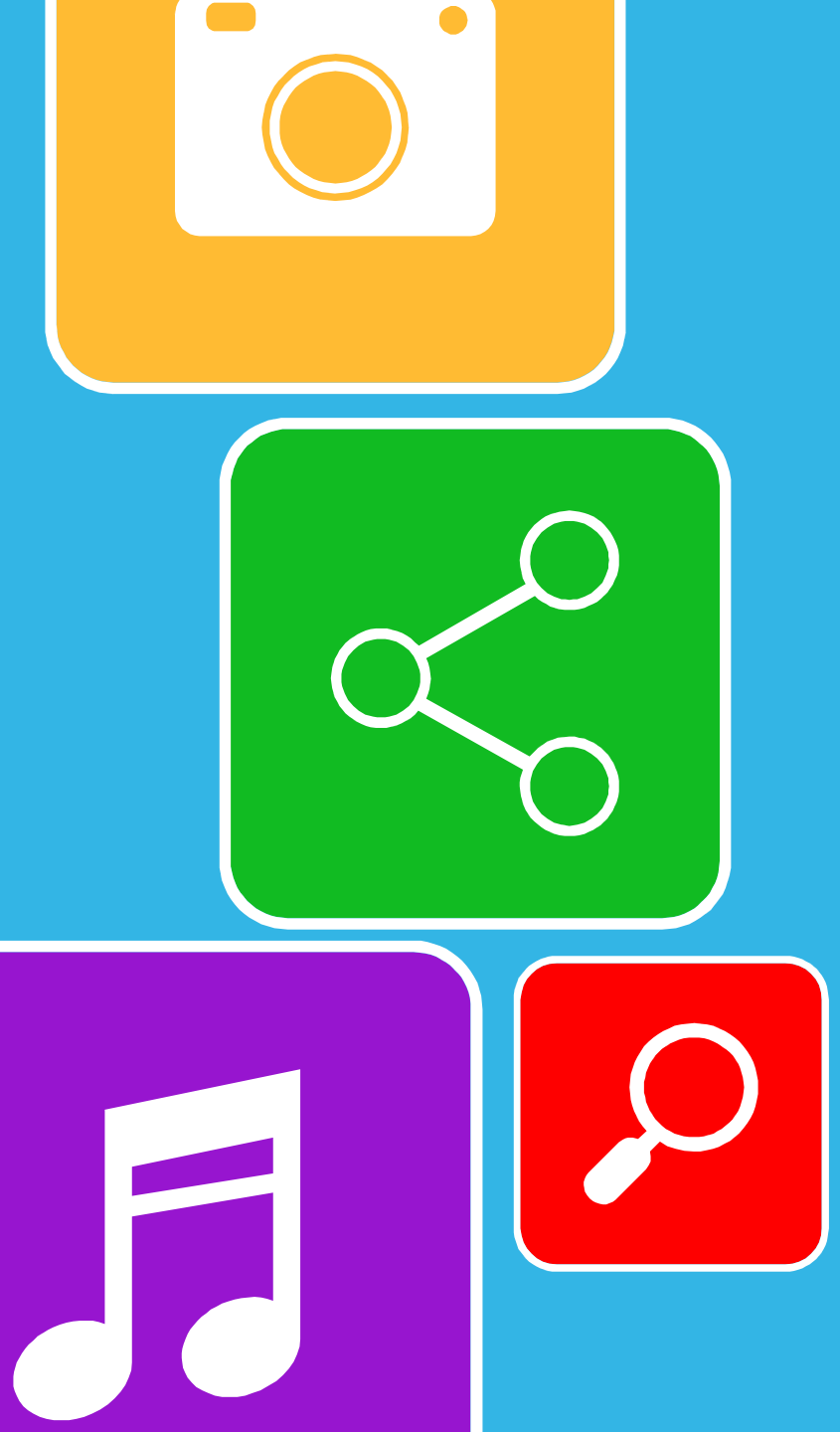


TensorFlow Tutorial3

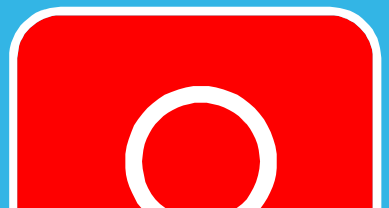
2016.12.29

Yunjey Choi, MS Student

DAVIAN Lab (led by Prof. Jaegul Choo)

Korea University



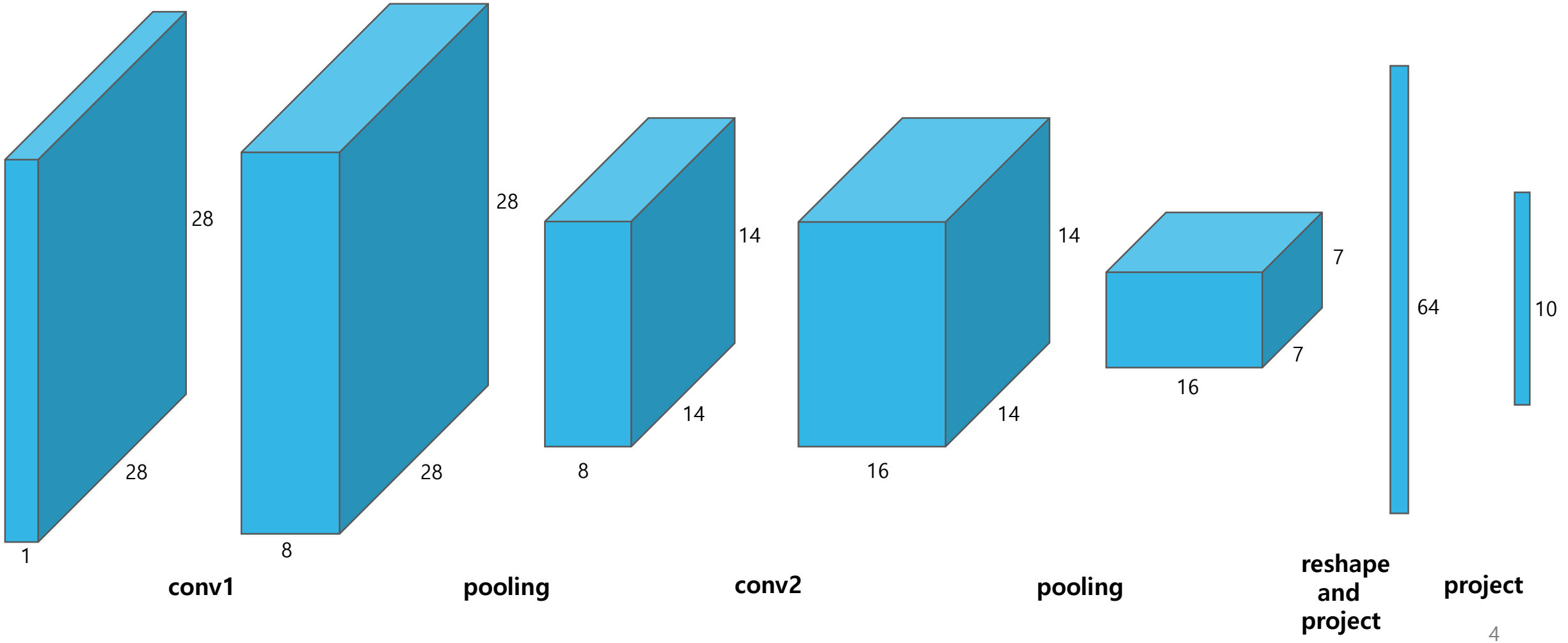



Contents

- CNN with TensorBoard
- Training Mechanism
- CNN for Text Classification
- Recurrent Neural Network

01 TensorBoard

CNN with TensorBoard



CNN with TensorBoard

```python
def conv2d(x, channel_out, k_w=5, k_h=5, s_w=1, s_h=1, name=None):
    """Computes convolution operation

    Args:
        x: input tensor of shape (batch_size, width_in, heigth_in, channel_in)
        channel_out: number of channel for output tensor
        k_w: kernel width size; default is 5
        k_h: kernel height size; default is 5
        s_w: stride size for width; default is 1
        s_h: stride size for heigth; default is 1

    Returns:
        out: output tensor of shape (batch_size, width_out, height_out, channel_out)
    """
    channel_in = x.get_shape()[-1]

    with tf.variable_scope(name):
        w = tf.get_variable('w', shape=[k_w, k_h, channel_in, channel_out],
                            initializer=tf.truncated_normal_initializer(stddev=0.01))
        b = tf.get_variable('b', shape=[channel_out], initializer=tf.constant_initializer(0.0))

        out = tf.nn.conv2d(x, w, strides=[1, s_w, s_h, 1], padding='SAME') + b

        return out
```

CNN with TensorBoard

```python
def conv2d(x, channel_out, k_w=5, k_h=5, s_w=1, s_h=1, name=None):
    """Computes convolution operation

    Args:
        x: input tensor of shape (batch_size, width_in, heigth_in, channel_in)
        channel_out: number of channel for output tensor
        k_w: kernel width size; default is 5
        k_h: kernel height size; default is 5
        s_w: stride size for width; default is 1
        s_h: stride size for heigth; default is 1

    Returns:
        out: output tensor of shape (batch_size, width_out, height_out, channel_out)
    """
    channel_in = x.get_shape()[-1]

    with tf.variable_scope(name):
        w = tf.get_variable('w', shape=[k_w, k_h, channel_in, channel_out],
                            initializer=tf.truncated_normal_initializer(stddev=0.01))
        b = tf.get_variable('b', shape=[channel_out], initializer=tf.constant_initializer(0.0))

        out = tf.nn.conv2d(x, w, strides=[1, s_w, s_h, 1], padding='SAME') + b

        return out
```

1. 변수명을 효율적으로 관리
2. TensorBoard상에서 모델을 이쁘게 출력

CNN with TensorBoard

convolutional network 모델

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

CNN with TensorBoard

첫 번째 convolution layer임을 이름으로 표기

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

CNN with TensorBoard

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

두 번째 convolution layer임을 이름으로 표기

CNN with TensorBoard

convolutional network 모델

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

CNN with TensorBoard

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

(fully connected) hidden layer임을 이름으로 표기

CNN with TensorBoard

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

(fully connected) output layer임을 이름으로 표기

CNN with TensorBoard

tf.variable_scope과 마찬가지로 tf.name_scope은 tensor들의 이름을 효율적으로 관리할 수 있게 해준다.

```
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

CNN with TensorBoard

모델의 input임을 명시

```
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

CNN with TensorBoard

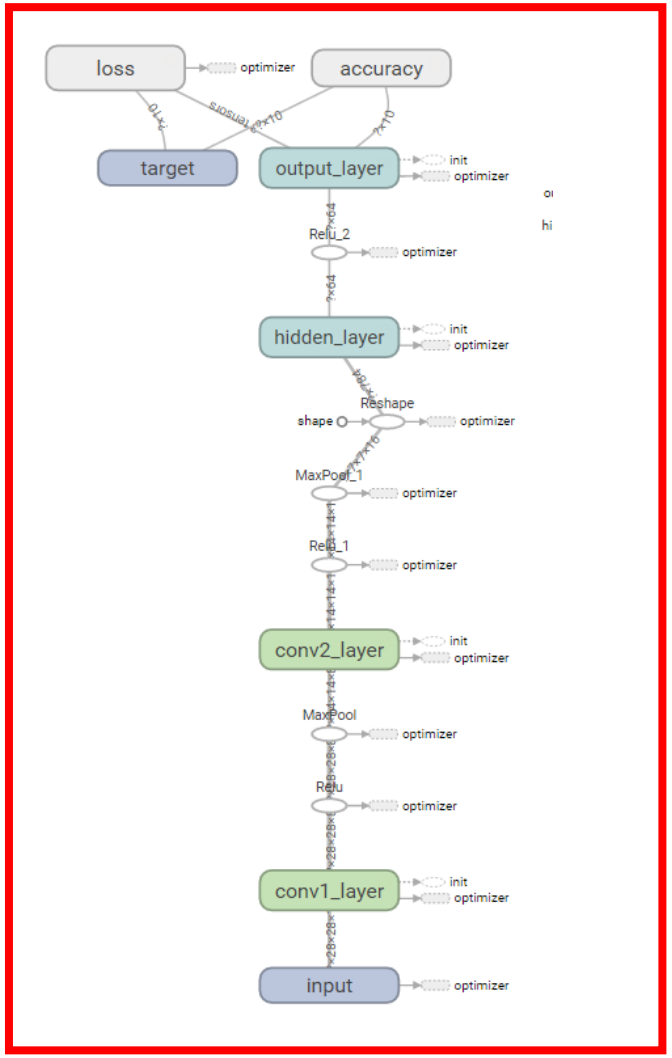
```
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

주어진 정답(target)임을 명시

CNN with TensorBoard

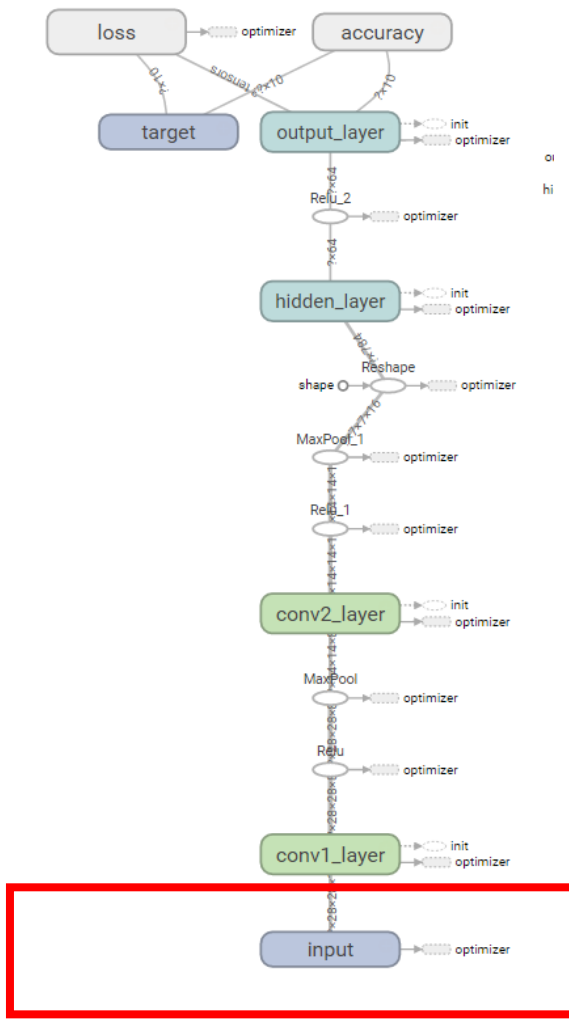
이렇게 이름을 잘 정해놓고
TensorBoard를 켜보면..

CNN with TensorBoard



자신이 구현한 모델을 한 눈에 보기 좋게 시각화를 해준다

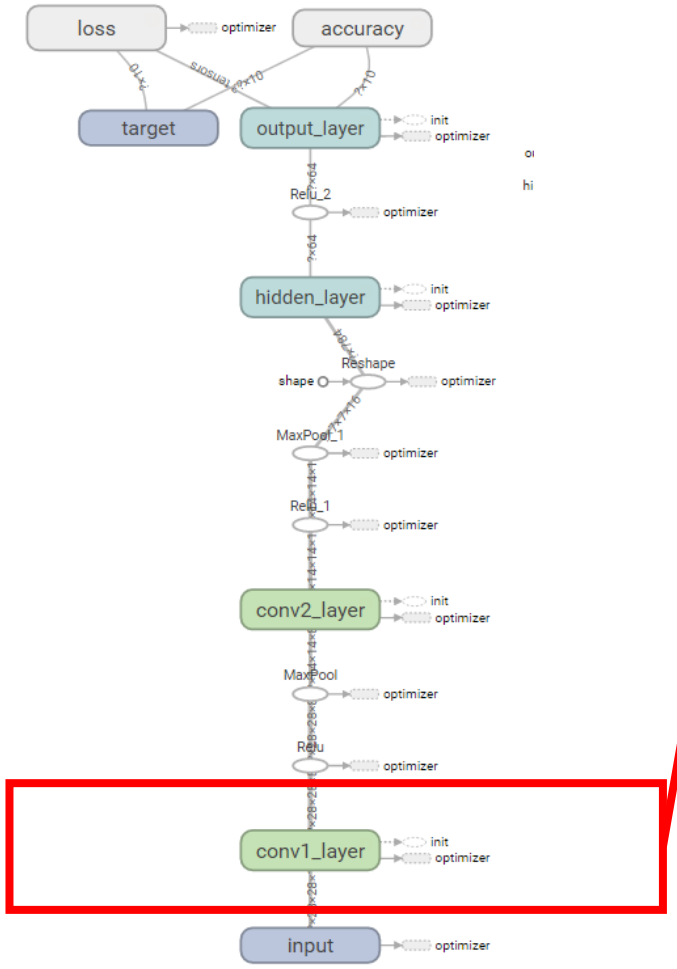
CNN with TensorBoard




```python
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```python
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

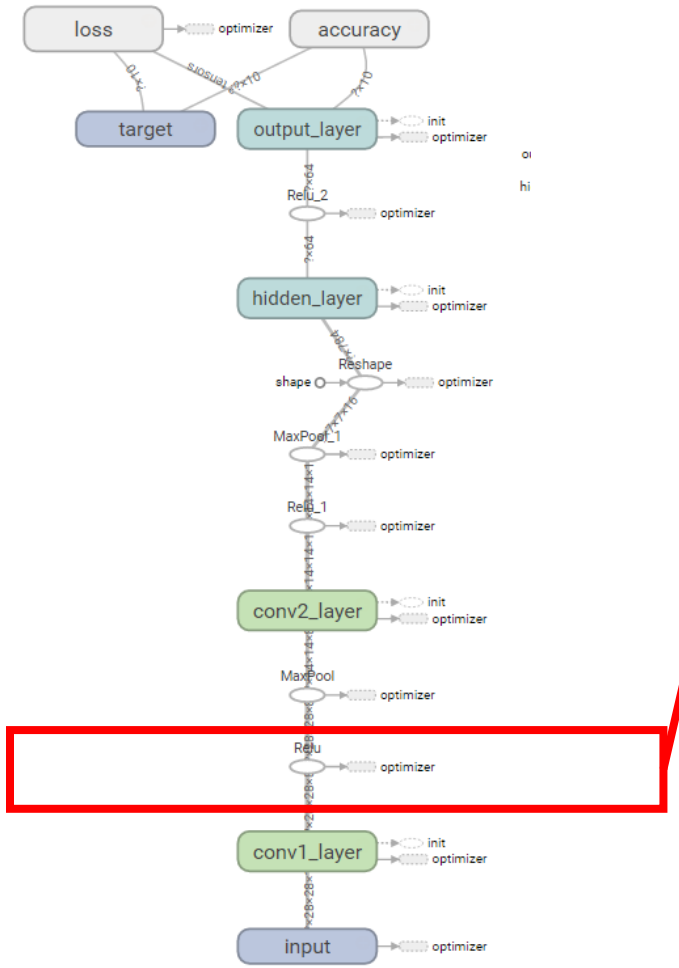
CNN with TensorBoard




```python
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```python
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

CNN with TensorBoard

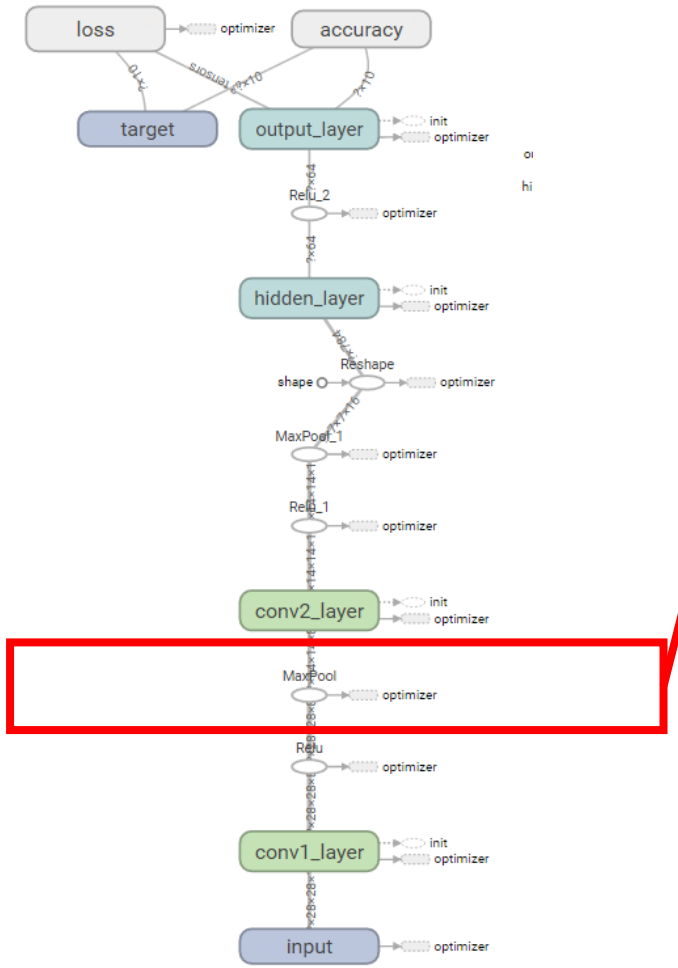


이름을 지어주진 않았지만 ReLU라고 자동으로 표기가 됨

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

CNN with TensorBoard

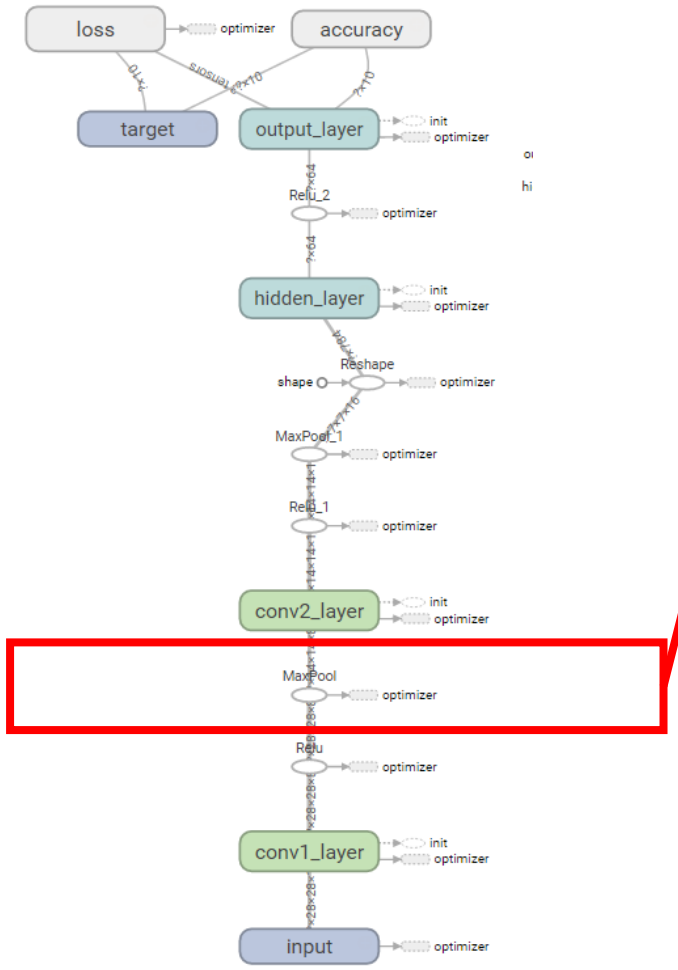


MaxPool도 마찬가지로 작게 표기됨

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

CNN with TensorBoard

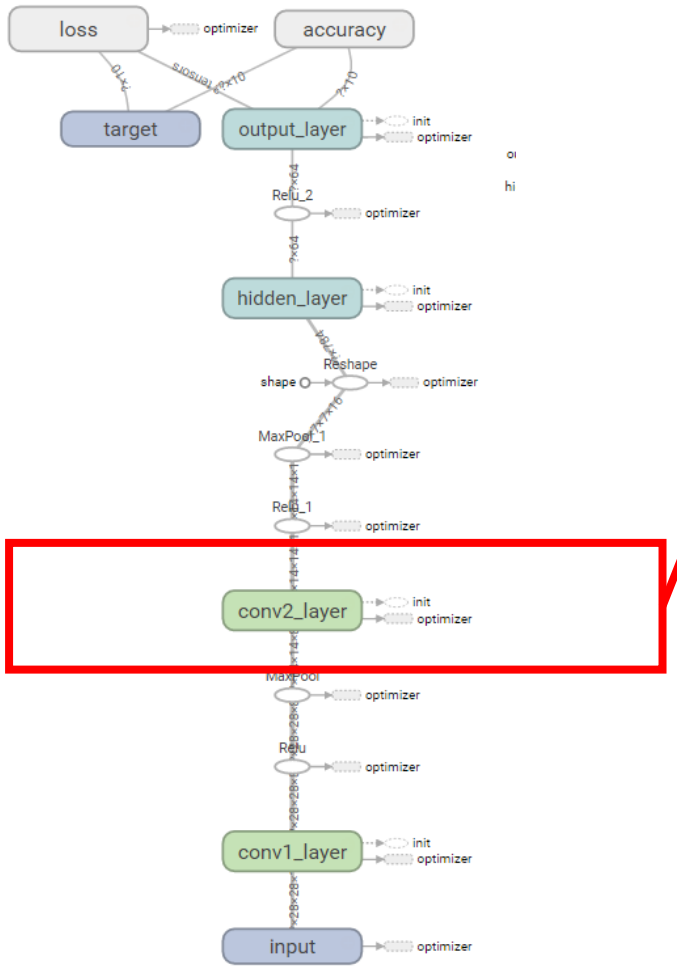


tf.variable_scope이나 tf.name_scope을 사용하면 보기 좋게 크게 표시가 됨

```
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

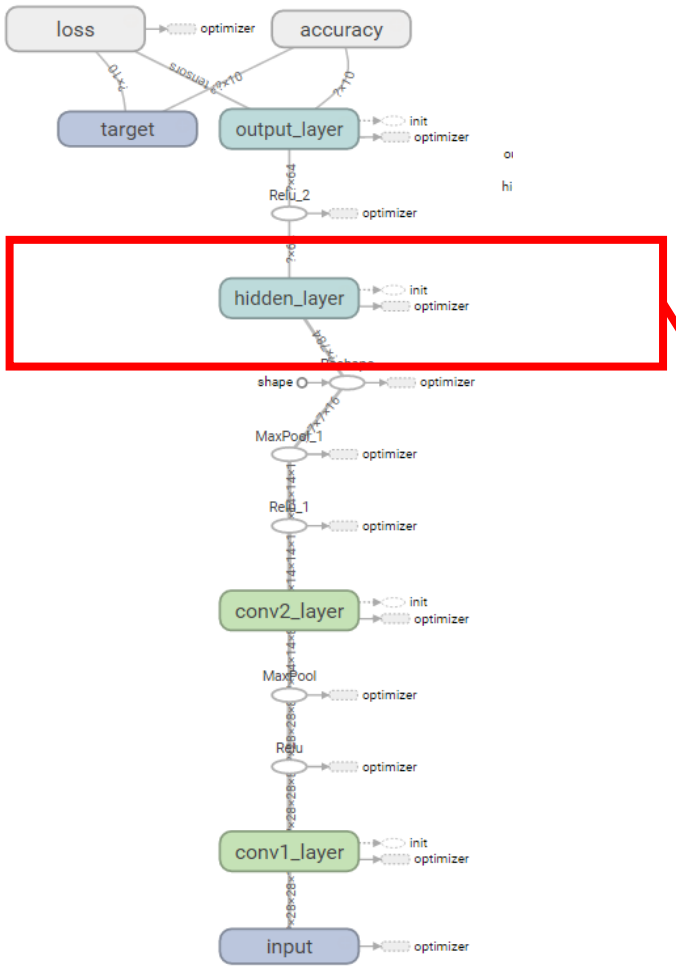
CNN with TensorBoard




```python
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```python
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

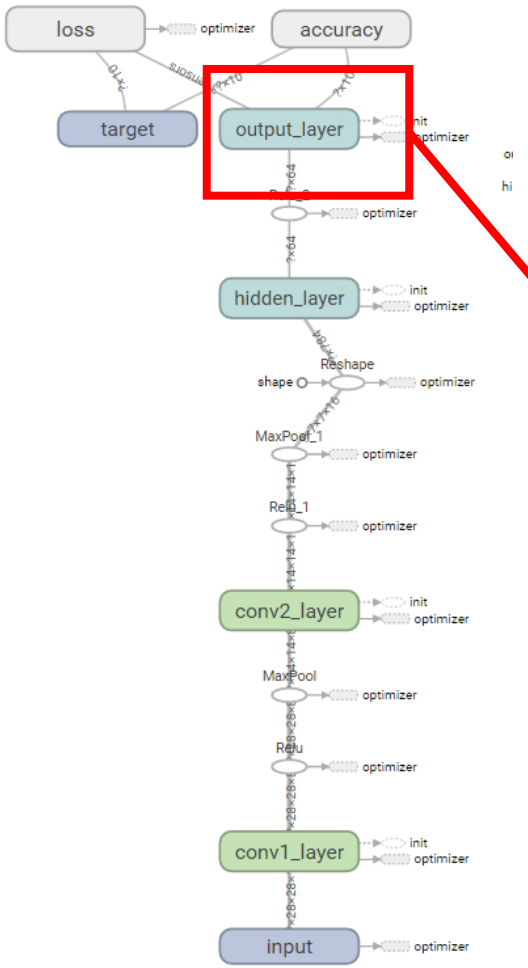
CNN with TensorBoard




```python
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```python
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

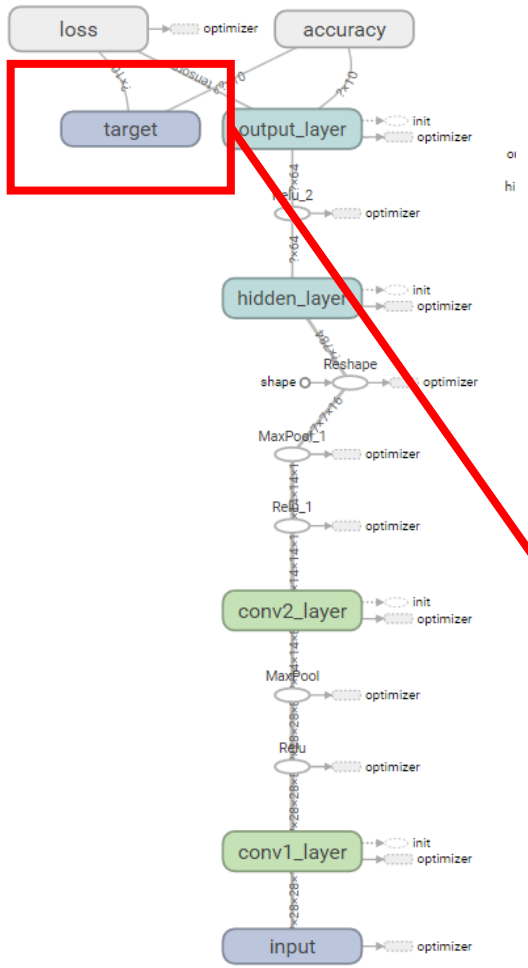
CNN with TensorBoard




```python
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```python
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

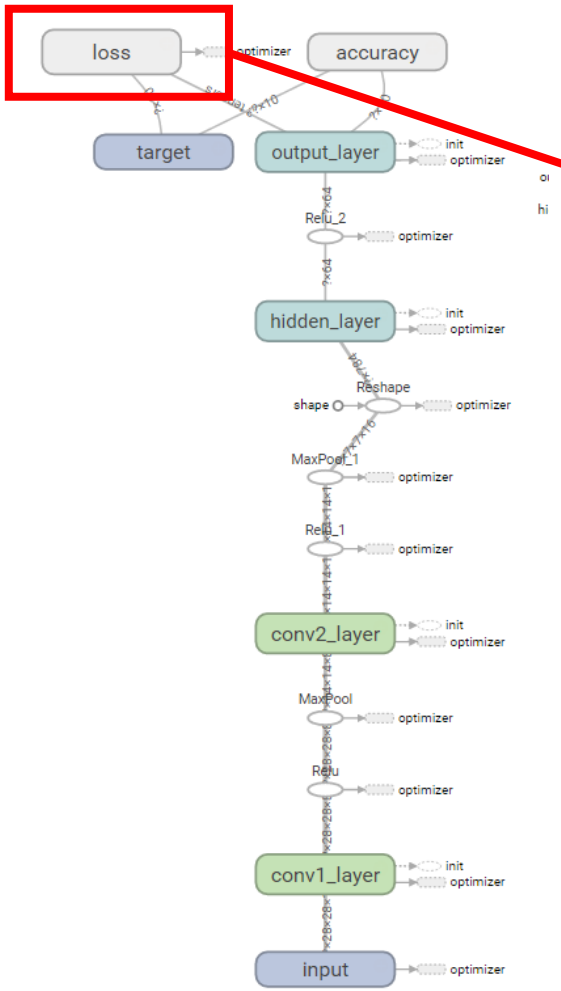
CNN with TensorBoard




```python
def convolutional_network(x):
    # 1st convolution layer
    conv1 = conv2d(x, channel_out=8, name='conv1_layer')
    conv1 = tf.nn.relu(conv1)
    conv1 = tf.nn.max_pool(conv1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # 2nd convolution layer
    conv2 = conv2d(conv1, channel_out=16, name='conv2_layer')
    conv2 = tf.nn.relu(conv2)
    conv2 = tf.nn.max_pool(conv2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')

    # fully connected layer
    conv2 = tf.reshape(conv2, [-1, 7*7*16])
    h = linear(conv2, dim_out=64, name='hidden_layer')
    h = tf.nn.relu(h)
    out = linear(h, dim_out=10, name='output_layer')
    return out
```

```python
# construct model
with tf.name_scope('input'):
    x = tf.placeholder(dtype=tf.float32, shape=[None, 28, 28, 1], name='images')
with tf.name_scope('target'):
    y = tf.placeholder(dtype=tf.int64, shape=[None, 10], name='labels')

out = convolutional_network(x)
```

CNN with TensorBoard

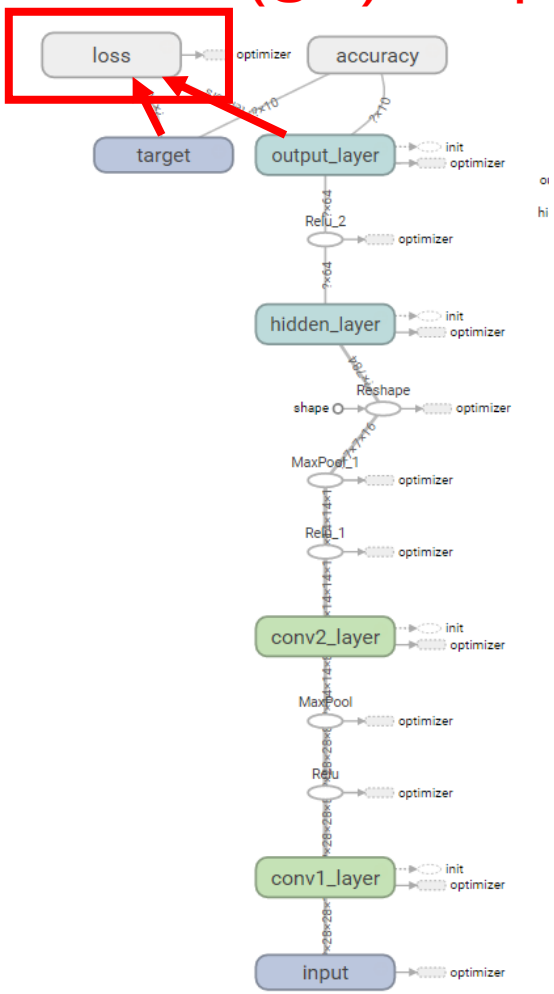


우리가 최종적으로 minimize시키고 싶은 loss도 이름을 표기하면 Tensorboard상에서 보기 쉽게 표기가 됨

```
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

CNN with TensorBoard

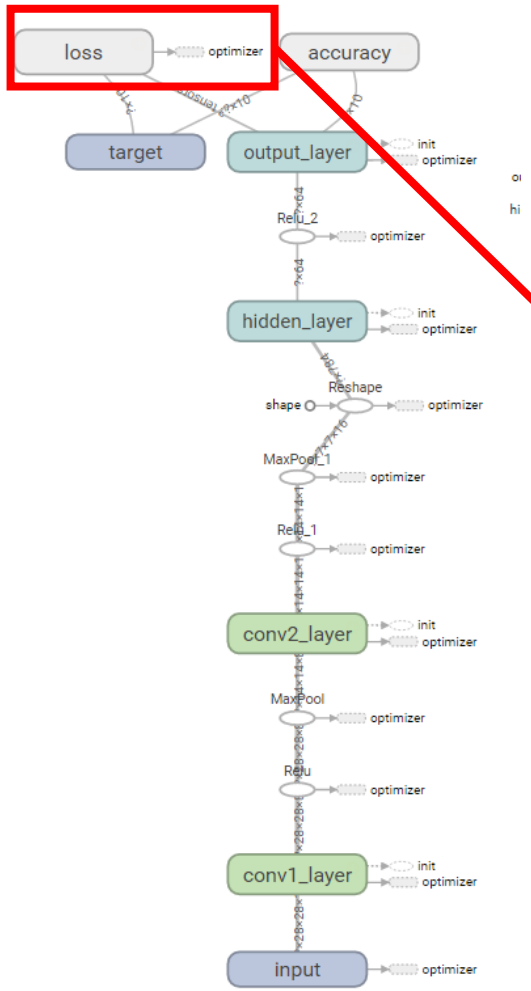
(중요) loss와 output_layer(out)와 target(y)이 이어져있다



(중요) loss의 input은 out과 y이다

```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

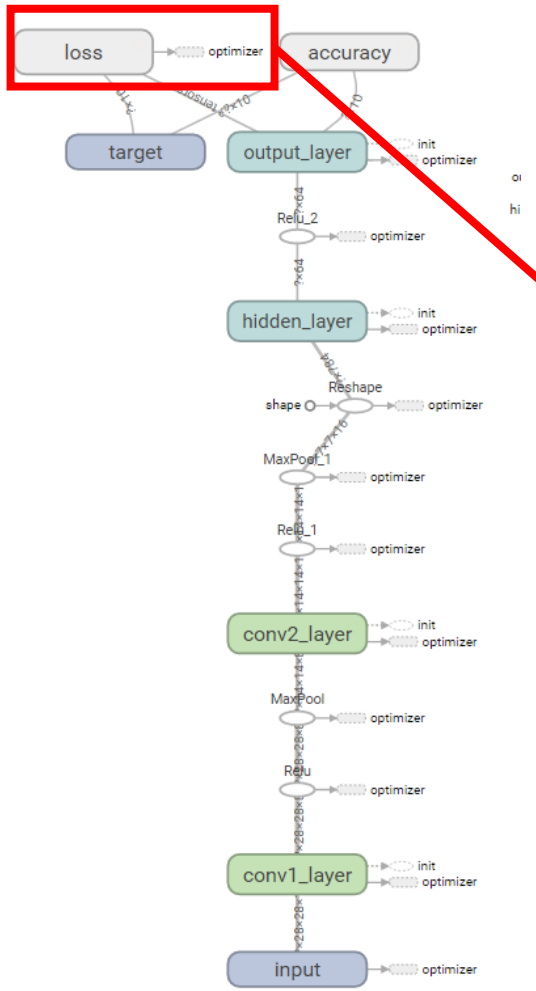
CNN with TensorBoard




```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

Loss를 줄이기 위한 optimizer
Backpropagation으로 모델 학습 (자동 미분)

CNN with TensorBoard



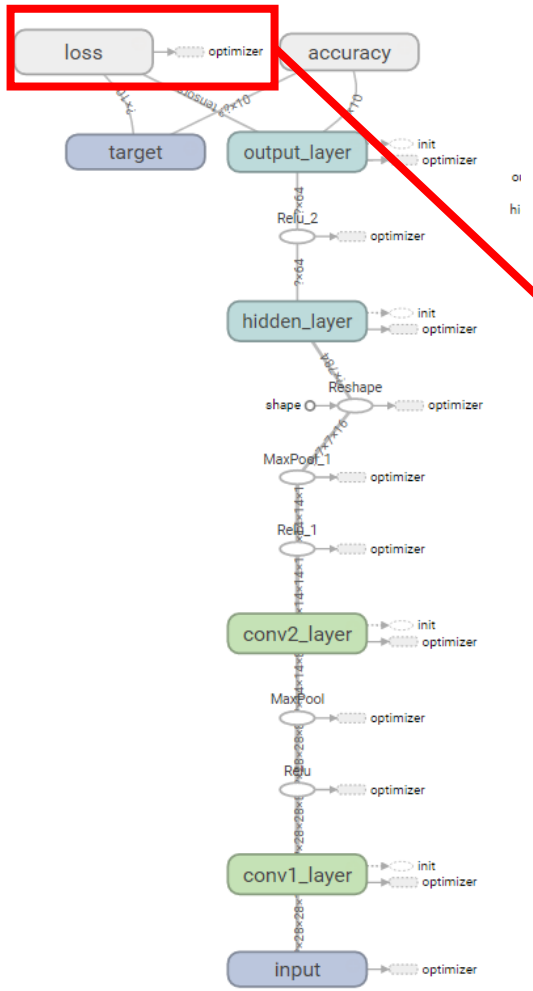

```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

1. Optimizer를 정해준다

여기서는 RMSPropOptimizer를 사용(gradient descent의 일종)

Learning rate은 0.001로 설정

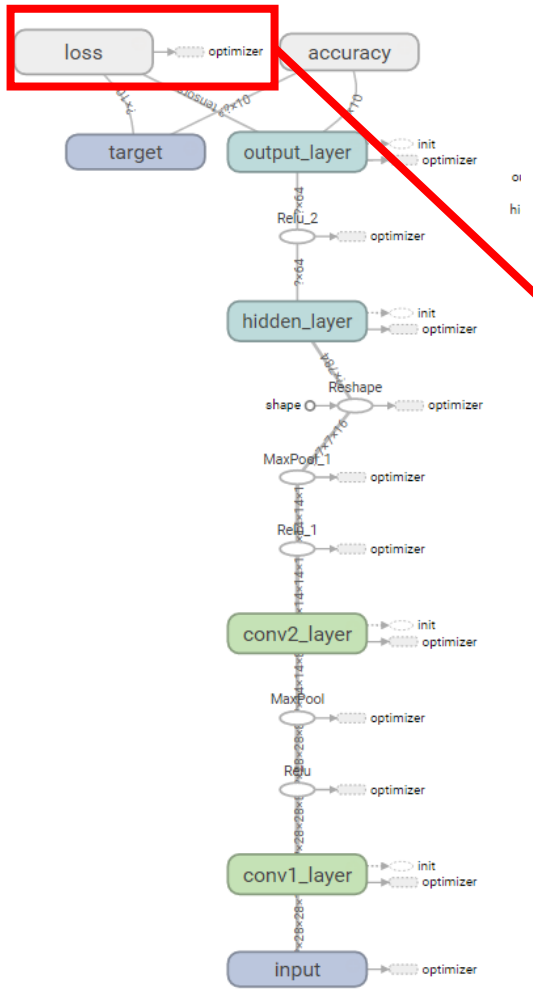
CNN with TensorBoard




```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

2. Loss를 weigh와 bias에 대해 미분을 해준다. (자동 미분)

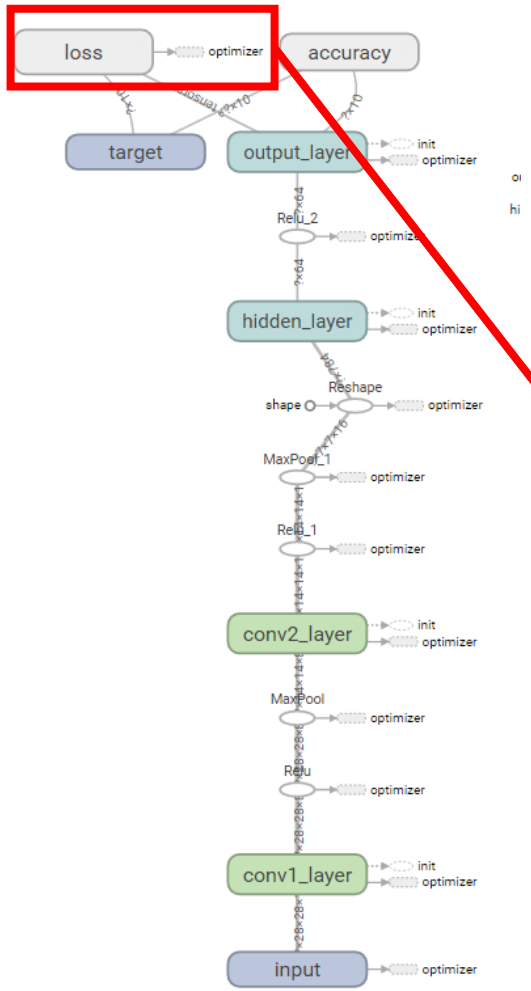
CNN with TensorBoard




```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

tf.trainable_variables()는 현재 학습가능한 변수들을 반환해준다.
(1) 학습 가능한 변수 tf.Variables (weight와 bias)
(2) 학습 불가능한 변수 tf.placeholder (input과 output)

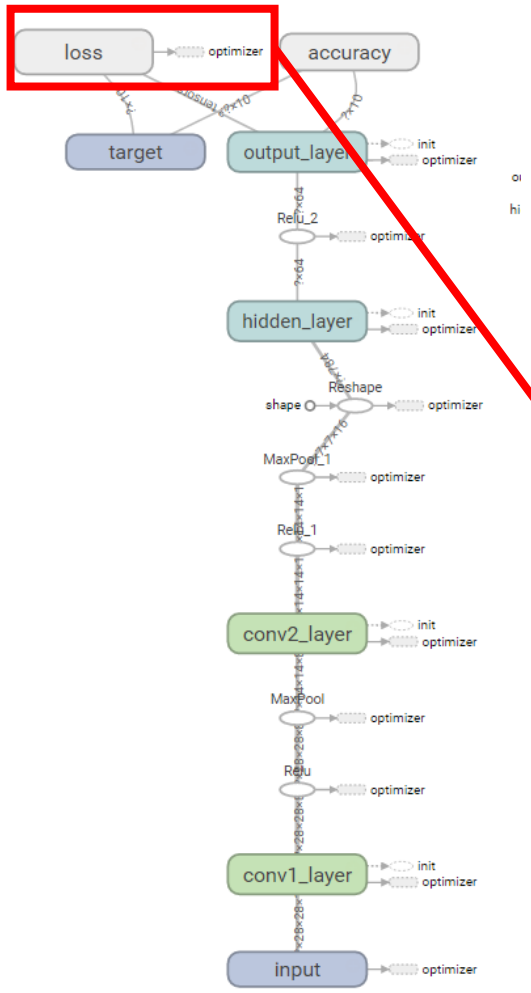
CNN with TensorBoard




```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

3. gradient(미분값)와 variables(weigh와 bias)를 연결한다.

CNN with TensorBoard




```python
# loss
with tf.name_scope('loss'):
    loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(out, y), name='batch_loss')

# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

4. apply_gradients
계산한 미분값을 통해 gradient “descent” 알고리즘 (여기서는 RMSProp)을 이용해 weigh와 bias를 update시킨다.

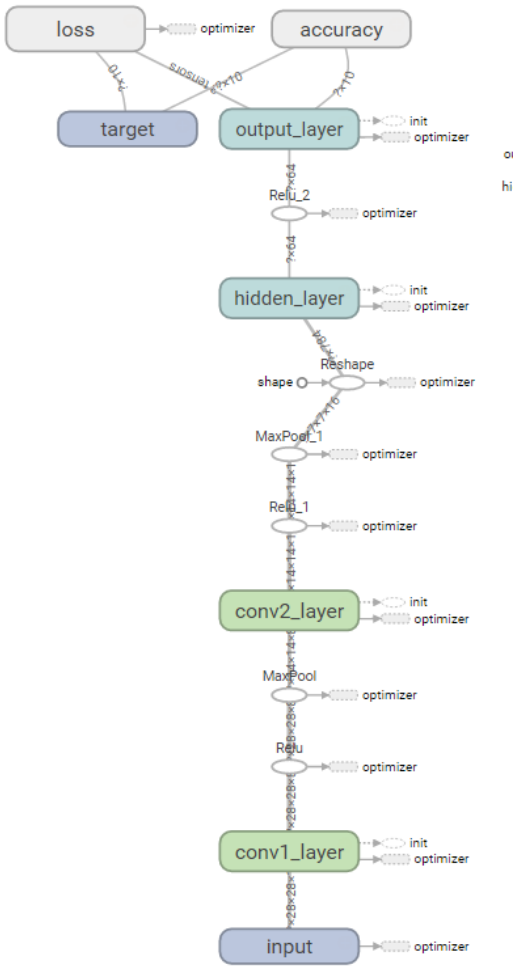
02 Training Mechanism

CNN with TensorBoard

여태까지는 연산자(operator)들을 정의만 해 놓은거고, 실행은 시키지 않았다.

연산자들을 실행시켜야 실제 학습이 진행이 된다

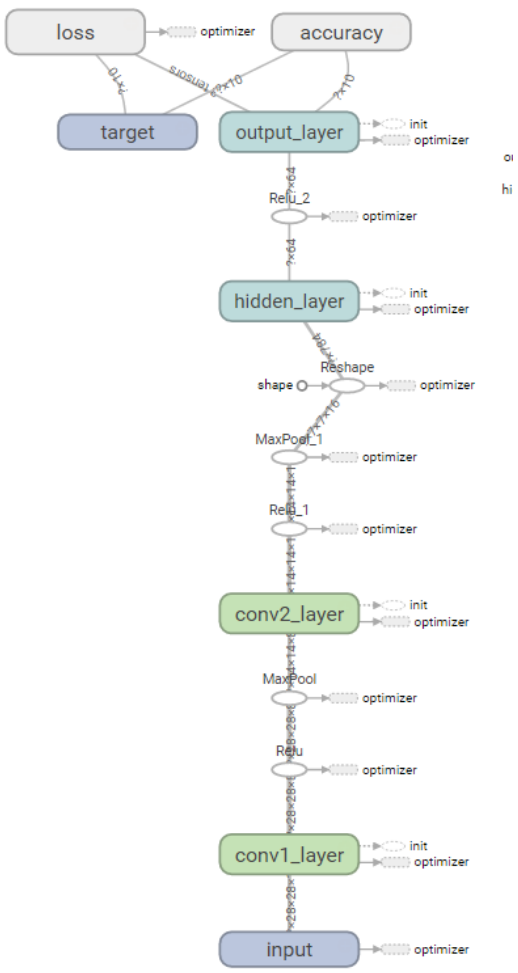
CNN with TensorBoard



연산자들을 실행하기 위해선 session을 열어야 한다

```
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

먼저 학습할 변수들을 초기화 시켜준다

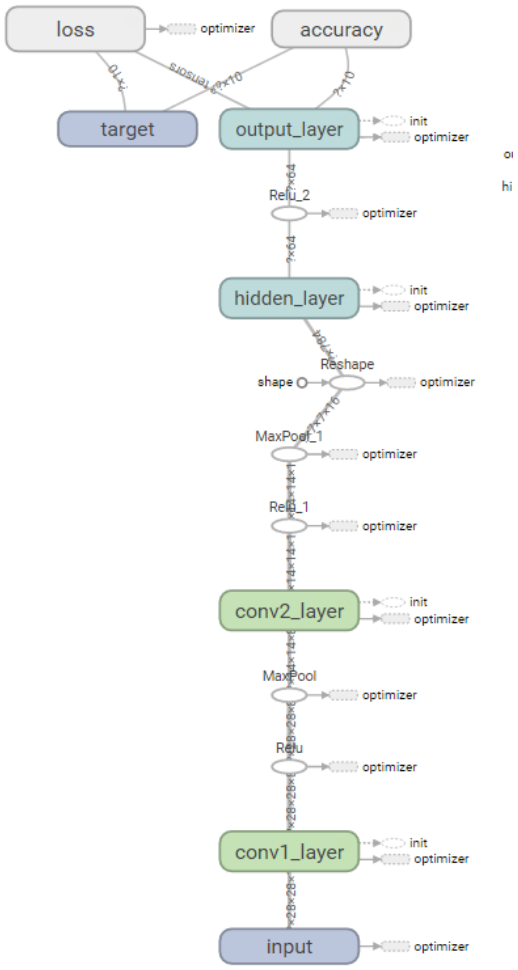
CNN with TensorBoard

```python
def conv2d(x, channel_out, k_w=5, k_h=5, s_w=1, s_h=1, name=None):
    """Computes convolution operation

    Args:
        x: input tensor of shape (batch_size, width_in, heigth_in, channel_in)
        channel_out: number of channel for output tensor
        k_w: kernel width size; default is 5
        k_h: kernel height size; default is 5
        s_w: stride size for width; default is 1
        s_h: stride size for heigth; default is 1

    Returns:
        out: output tensor of shape (batch_size, width_out, height_out, channel_out)
    """
    channel_in = x.get_shape()[-1]

    with tf.variable_scope(name):
        w = tf.get_variable('w', shape=[k_w, k_h, channel_in, channel_out],
                            initializer=tf.truncated_normal_initializer(stddev=0.01))
        b = tf.get_variable('b', shape=[channel_out], nitializer=tf.constant_initializer(0.0))

        out = tf.nn.conv2d(x, w, strides=[1, s_w, s_h, 1], padding='SAME') + b

        return out
```

```python
def linear(x, dim_out, name=None):
    """Computes linear transform (fully-connected layer)

    Args:
        x: input tensor of shape (batch_size, dim_in)
        dim_out: dimension for output tensor

    Returns:
        out: output tensor of shape (batch_size, dim_out)
    """
    dim_in = x.get_shape()[-1]

    with tf.variable_scope(name):
        w = tf.get_variable('w', shape=[dim_in, dim_out],
                            initializer=tf.truncated_normal_initializer(stddev=0.01))
        b = tf.get_variable('b', shape=[dim_out], initializer=tf.constant_initializer(0.0))

        out = tf.matmul(x, w) + b

        return out
```

어떤 방법으로 초기화 시켜줄지는 위에서 정해놓았다

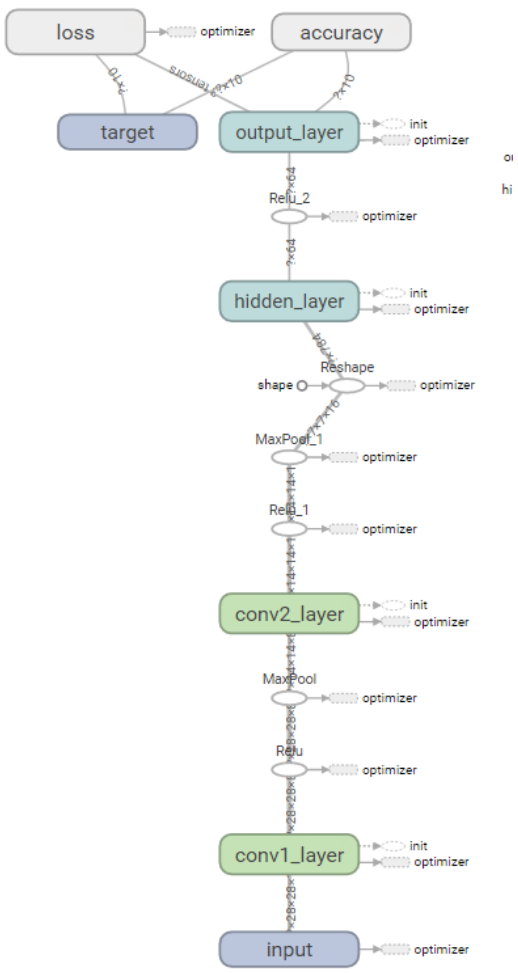
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

Tensorboard를 위한 코드
구현한 코드를 보기 쉽게 시각화

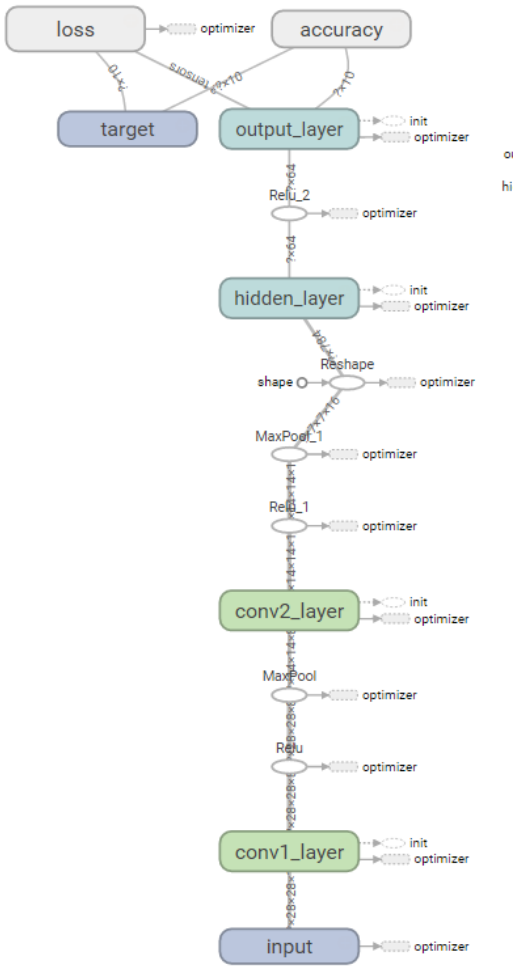
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

number of epoch: 전체 training data를 몇번 모델에게 학습시킬지

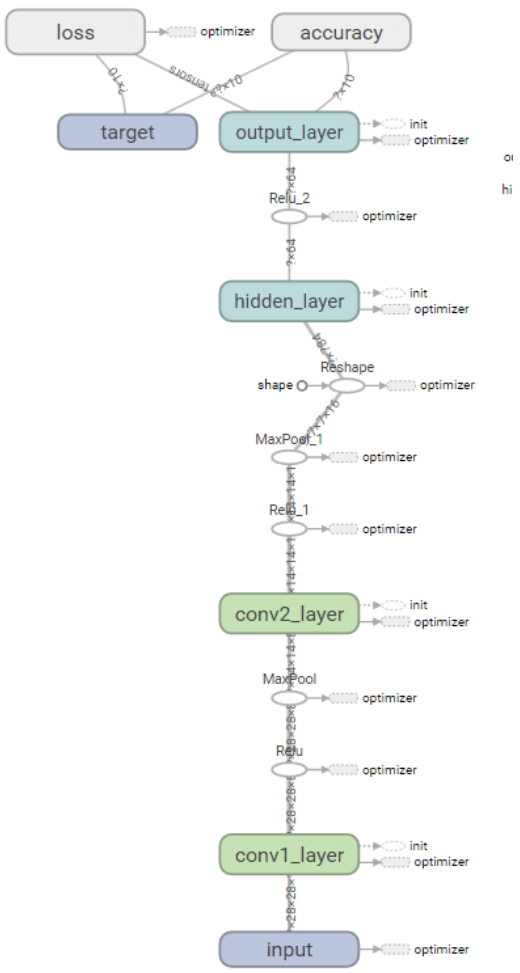
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

사람도 영어단어를 외울 때 여러 번 봐야하
듯이 모델도 학습데이터를 여러 번 봐야함

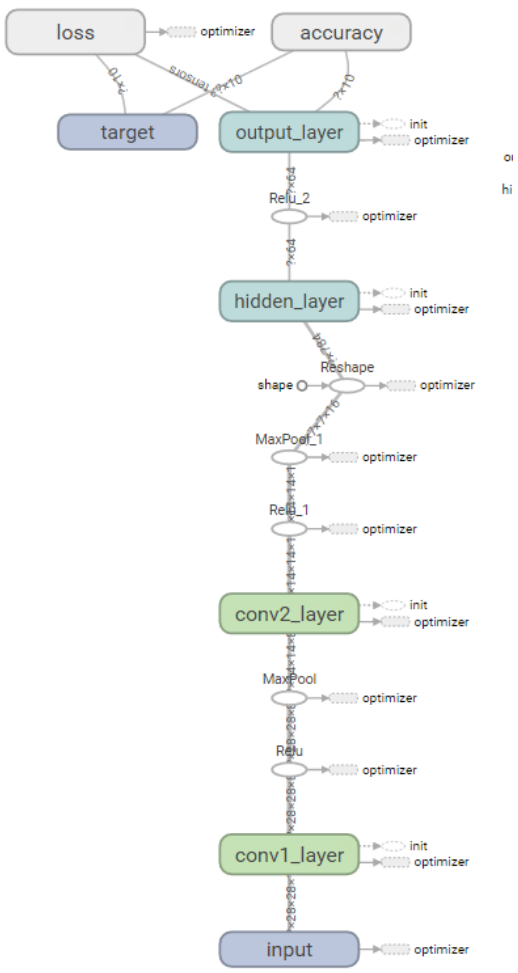
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

매 epoch마다 loss를 초기화
여기서 loss는 단순히 출력용!

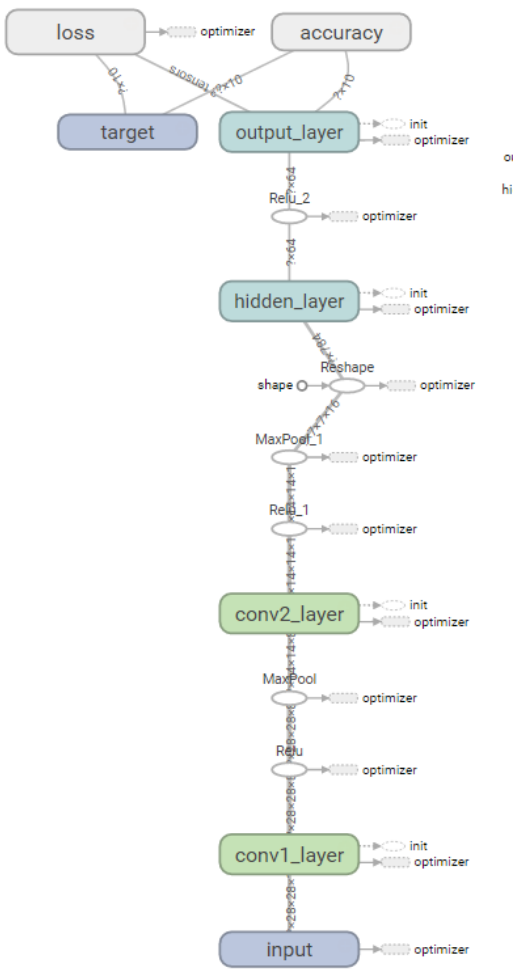
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

Number of iteration per epoch
전체 데이터를 모두 훑는데 필요한 iteration 수

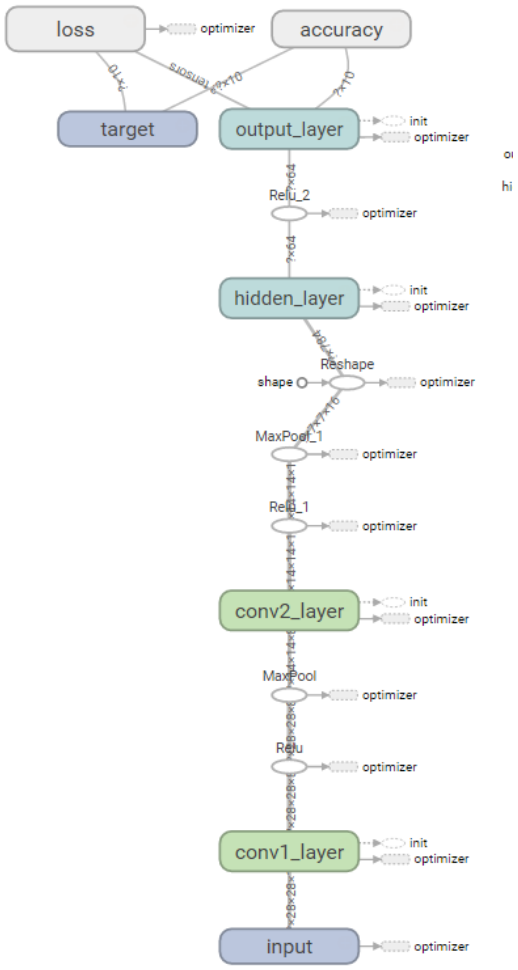
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

 # training data: 55000
 # batch size: 100
 # numb_iter_per_epoch: 550

CNN with TensorBoard

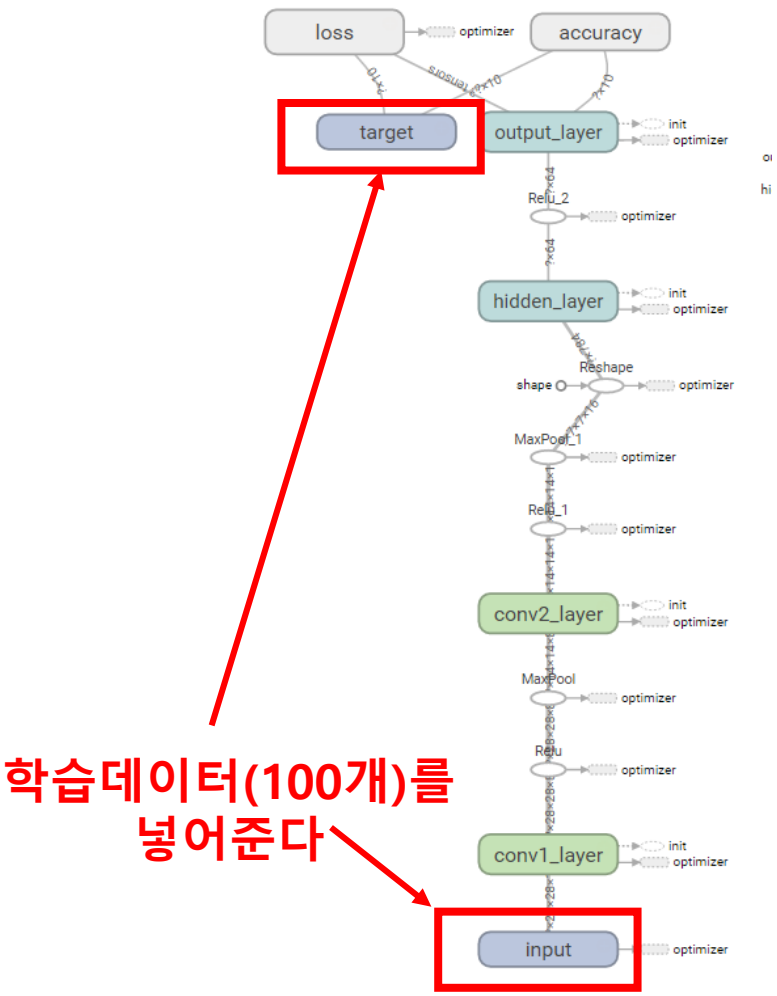


현재 필요한 학습 데이터를 가져온다

```
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

ex) i=1, batch_size=100
x_train[100:200]: 100번째~199번째 데이터를 가져옴

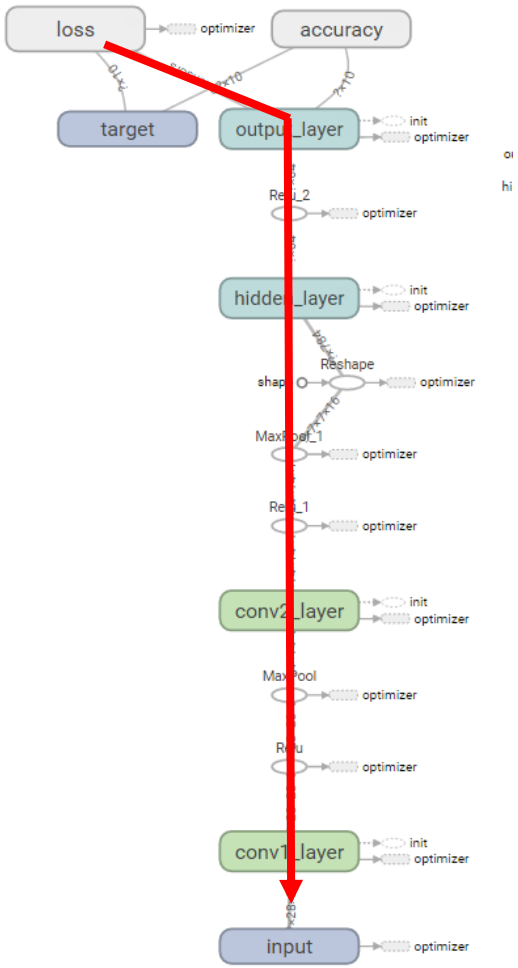
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

placeholder(tensorflow)와 실제 input(numpy)를 연결

CNN with TensorBoard




```
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

(제일 중요) 실제 학습이 일어나는 시점

CNN with TensorBoard

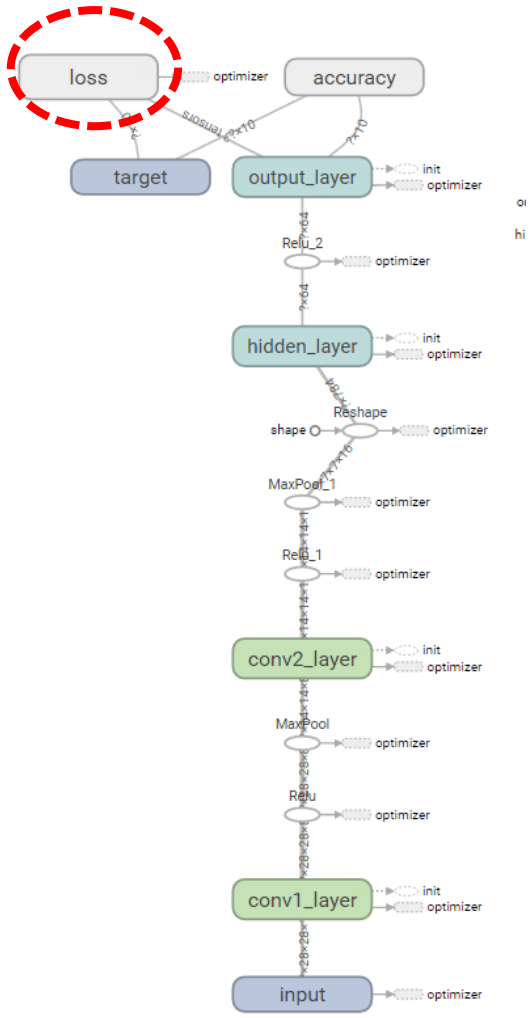
train_op이라는 연산자를 정의

```
# accuracy
with tf.name_scope('accuracy'):
    pred = tf.argmax(out, 1)
    target = tf.argmax(y, 1)
    correct_pred = tf.equal(pred, target)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# train op
with tf.name_scope('optimizer'):
    optimizer = tf.train.RMSPropOptimizer(learning_rate=0.001)
    grads = tf.gradients(loss, tf.trainable_variables())
    grads_and_vars = list(zip(grads, tf.trainable_variables()))
    train_op = optimizer.apply_gradients(grads_and_vars=grads_and_vars)
```

앞에서 정의한 train_op이라는 연산자를 실행

```
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

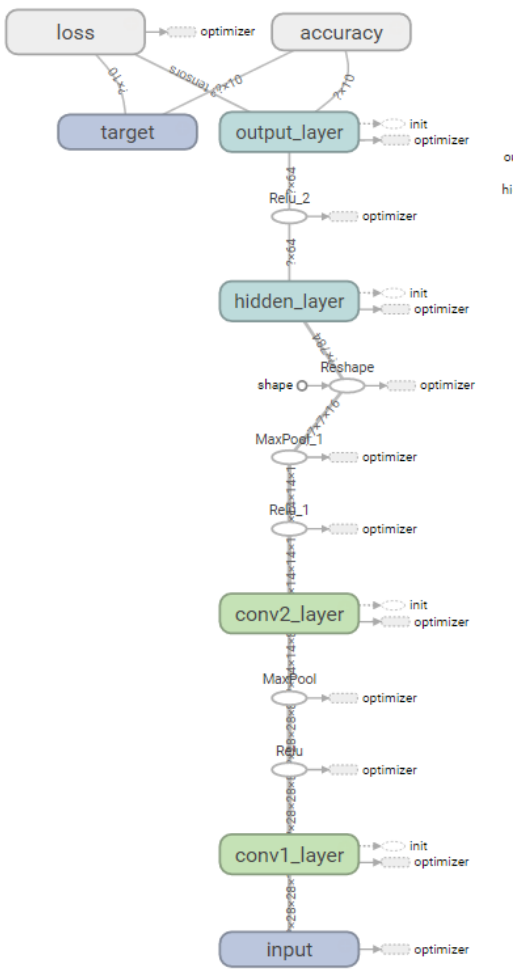
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

Loss값을 반환 (출력용)

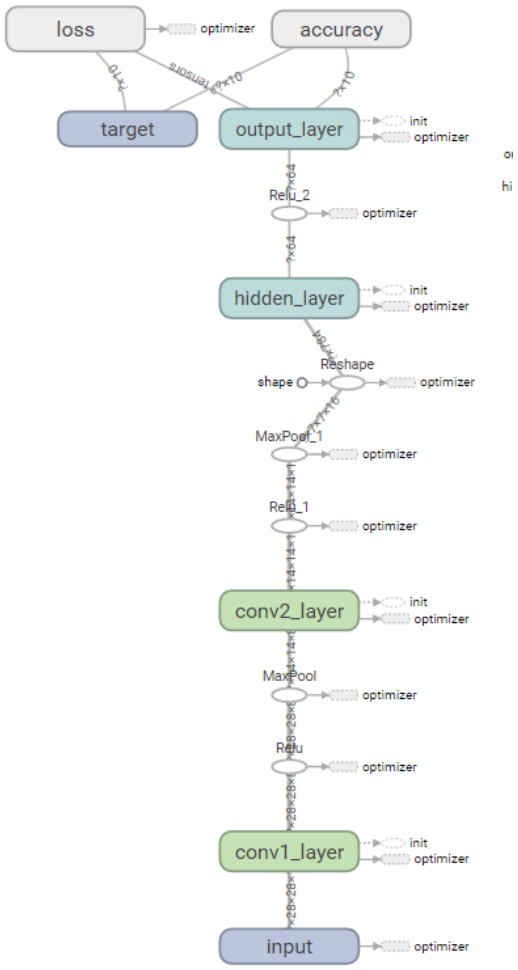
CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

평균 loss를 계산 (출력용)

CNN with TensorBoard




```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

Tensorboard를 위한 코드

CNN with TensorBoard

```
Epoch 1, Loss: 1.147
Epoch 2, Loss: 0.151
Epoch 3, Loss: 0.087
Epoch 4, Loss: 0.065
Epoch 5, Loss: 0.052
Finished training!

Test accuracy: 0.977
```

```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

매 epoch가 끝날때 마다 loss 출력

CNN with TensorBoard

```
Epoch 1, Loss: 1.147
Epoch 2, Loss: 0.151
Epoch 3, Loss: 0.087
Epoch 4, Loss: 0.065
Epoch 5, Loss: 0.052
Finished training!

Test accuracy: 0.977
```

```python
# launch the graph
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
with tf.Session(config=config) as sess:
    # initialize tensor variables
    tf.initialize_all_variables().run()
    summary_writer = tf.train.SummaryWriter(log_path, graph=tf.get_default_graph())
    # training phase
    for e in range(num_epoch):
        avg_loss = 0.
        # loop over all batches
        for i in range(num_iter_per_epoch):
            x_batch = x_train[i*batch_size:(i+1)*batch_size]
            y_batch = y_train[i*batch_size:(i+1)*batch_size]
            # run optimization op (backprop) and loss op (to get loss value)
            feed_dict={x: x_batch, y: y_batch}
            _, c = sess.run([train_op, loss], feed_dict=feed_dict)
            # compute average loss
            avg_loss += c / num_iter_per_epoch

            if i % 10 == 0:
                summary = sess.run(summary_op, feed_dict)
                summary_writer.add_summary(summary, e*num_iter_per_epoch + i)
        print ("Epoch %d, Loss: %.3f"% (e+1, avg_loss))
    print ("Finished training!₩n")
```

학습 끝!

03
CNN for Text Classification

CNN with Text Classification

Load dataset

CNN with Text Classification

Load the dataset

positive & negative sentence들

```
x_pos = open('data/polarity/pos.txt').readlines()
x_neg = open('data/polarity/neg.txt').readlines()
y_pos = np.ones(len(x_pos))
y_neg = np.zeros(len(x_neg))
y = np.concatenate([y_pos, y_neg])
```

```
print len(x_pos)
print len(x_neg)
print x_pos[3]
print x_neg[0]
```

```
5331
5331
if you sometimes like to go to the movies to have fun , wasabi is a good place to start .

simplistic , silly and tedious .
```

CNN with Text Classification

Load the dataset

```
x_pos = open('data/polarity/pos.txt').readlines()
x_neg = open('data/polarity/neg.txt').readlines()
y_pos = np.ones(len(x_pos))
y_neg = np.zeros(len(x_neg))
y = np.concatenate([y_pos, y_neg])
```

```
print len(x_pos)
print len(x_neg)
print x_pos[3]
print x_neg[0]
```

```
5331
5331
if you sometimes like to go to the movies to have fun , wasabi is a good place to start .

simplistic , silly and tedious .
```

Positive/negative 각각 5331개의 문장

CNN with Text Classification

Load the dataset

```
x_pos = open('data/polarity/pos.txt').readlines()
x_neg = open('data/polarity/neg.txt').readlines()
y_pos = np.ones(len(x_pos))
y_neg = np.zeros(len(x_neg))
y = np.concatenate([y_pos, y_neg])
```

```
print len(x_pos)
print len(x_neg)
print x_pos[3]
print x_neg[0]
```

positive한 문장의 예

```
5331
5331
if you sometimes like to go to the movies to have fun , wasabi is a good place to start .

simplistic , silly and tedious .
```

CNN with Text Classification

Load the dataset

```
x_pos = open('data/polarity/pos.txt').readlines()
x_neg = open('data/polarity/neg.txt').readlines()
y_pos = np.ones(len(x_pos))
y_neg = np.zeros(len(x_neg))
y = np.concatenate([y_pos, y_neg])
```

```
print len(x_pos)
print len(x_neg)
print x_pos[3]
print x_neg[0]
```

negative한 문장의 예

```
5331
5331
if you sometimes like to go to the movies to have fun , wasabi is a good place to start .

simplistic , silly and tedious .
```

CNN with Text Classification

Load the dataset

positive & negative sentence들

```
x_pos = open('data/polarity/pos.txt').readlines()
x_neg = open('data/polarity/neg.txt').readlines()
y_pos = np.ones(len(x_pos))
y_neg = np.zeros(len(x_neg))
y = np.concatenate([y_pos, y_neg])
```

```
print len(x_pos)
print len(x_neg)
print x_pos[3]
print x_neg[0]
```

```
5331
5331
if you sometimes like to go to the movies to have fun , wasabi is a good place to start .

simplistic , silly and tedious .
```

CNN with Text Classification

Preprocessing

CNN with Text Classification

Preprocessing

전처리 함수: utils.py에 구현되어 있음

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[   0  361 3948   38 4605    1    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

CNN with Text Classification

문장이 숫자형태로 mapping된 데이터

Preprocessing

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

110번째 문장

110번째 문장 (단어들이 숫자로 맵핑되어 있다)

```
[    0   361  3948    38  4605     1     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2     2     2     2     2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

CNN with Text Classification

Preprocessing

word-to-index dictionary: 단어를 숫자로 맵핑해주는 사전 역할

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[   0  361 3948   38 4605    1    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False]
58
18768
```

```
print word_to_idx['<START>']
print word_to_idx['two']
print word_to_idx['hours']
print word_to_idx['of']
print word_to_idx['junk']
print word_to_idx['<END>']
print word_to_idx['<PAD>']
```

```
0
361
3948
38
4605
1
2
```

CNN with Text Classification

Preprocessing

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[    0   361  3948    38  4605     1     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2     2     2     2     2     2     2
     2     2     2     2     2     2     2     2     2     2     2     2     2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

일치!

```
print word_to_idx['<START>']
print word_to_idx['two']
print word_to_idx['hours']
print word_to_idx['of']
print word_to_idx['junk']
print word_to_idx['<END>']
print word_to_idx['<PAD>']
```

```
0
361
3948
38
4605
1
2
```

CNN with Text Classification

원래 문장: two hours of junk (두 시간짜리 쓰레기라는 뜻)

Preprocessing

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[   0  361 3948   38 4605    1    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

일치!

```
print word_to_idx['<START>']
print word_to_idx['two']
print word_to_idx['hours']
print word_to_idx['of']
print word_to_idx['junk']
print word_to_idx['<END>']
print word_to_idx['<PAD>']
```

```
0
361
3948
38
4605
1
2
```

CNN with Text Classification

Preprocessing

가장 긴 문장 (포함하는 단어의 최대 개수)

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[   0  361 3948   38 4605    1    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

CNN with Text Classification

Preprocessing

서로 다른 단어의 개수

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[   0  361 3948   38 4605    1    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

CNN with Text Classification

decode_sequence 함수: 숫자 벡터를 다시 문장으로 복원 (utils.py에 구현되어있음)

```
decoded = decode_sequence(x, word_to_idx)
```

```
print decoded[110]
```

```
two hours of junk .
```

Preprocessing

```
x, mask, word_to_idx, seq_length, vocab_size = preprocess(x_pos+x_neg)
```

```
print x[110]
print mask[110]
print seq_length
print vocab_size
```

```
[   0  361 3948   38 4605    1    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
    2    2    2    2    2    2    2    2    2    2    2    2]
[ True  True  True  True  True  True  True  True  True  True  True  True
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False]
58
18768
```

CNN with Text Classification

sklearn (Python 라이브러리)

```
# randomly shuffle data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=42)

print x_train.shape
print x_test.shape
print y_train.shape
print y_test.shape
```

```
(9595, 58)
(1067, 58)
(9595,)
(1067,)
```

CNN with Text Classification

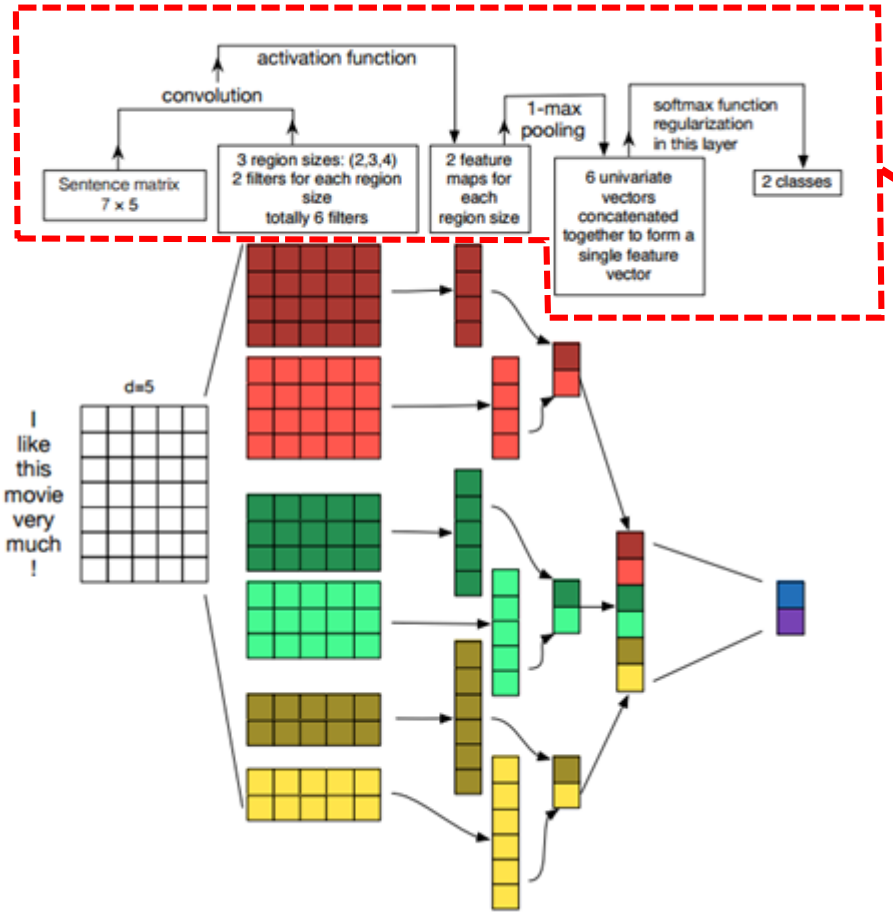
90%는 training data, 10%는 test data로 나눠준다

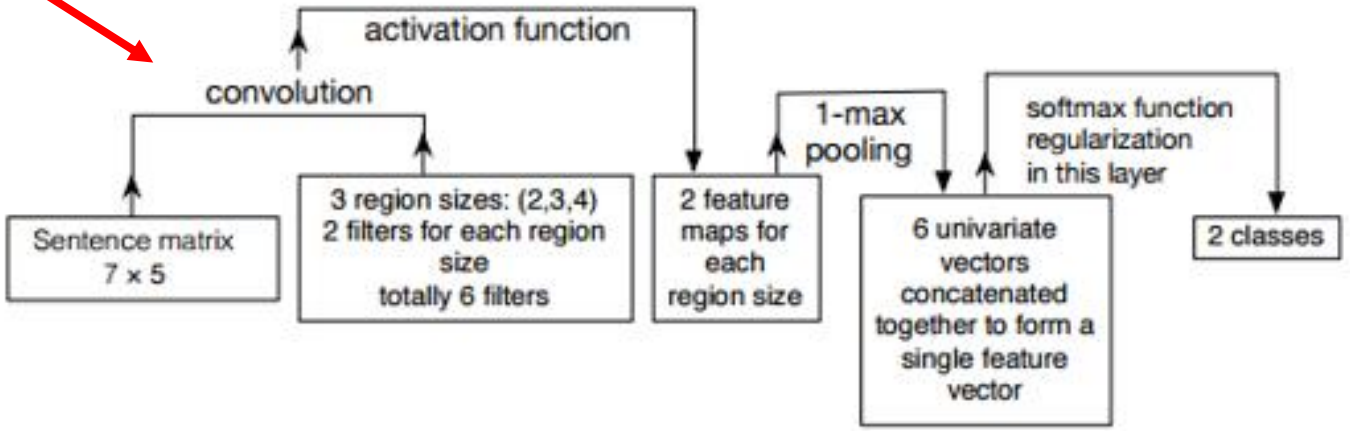
```
# randomly shuffle data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=42)

print x_train.shape
print x_test.shape
print y_train.shape
print y_test.shape
```

```
(9595, 58)
(1067, 58)
(9595,)
(1067,)
```

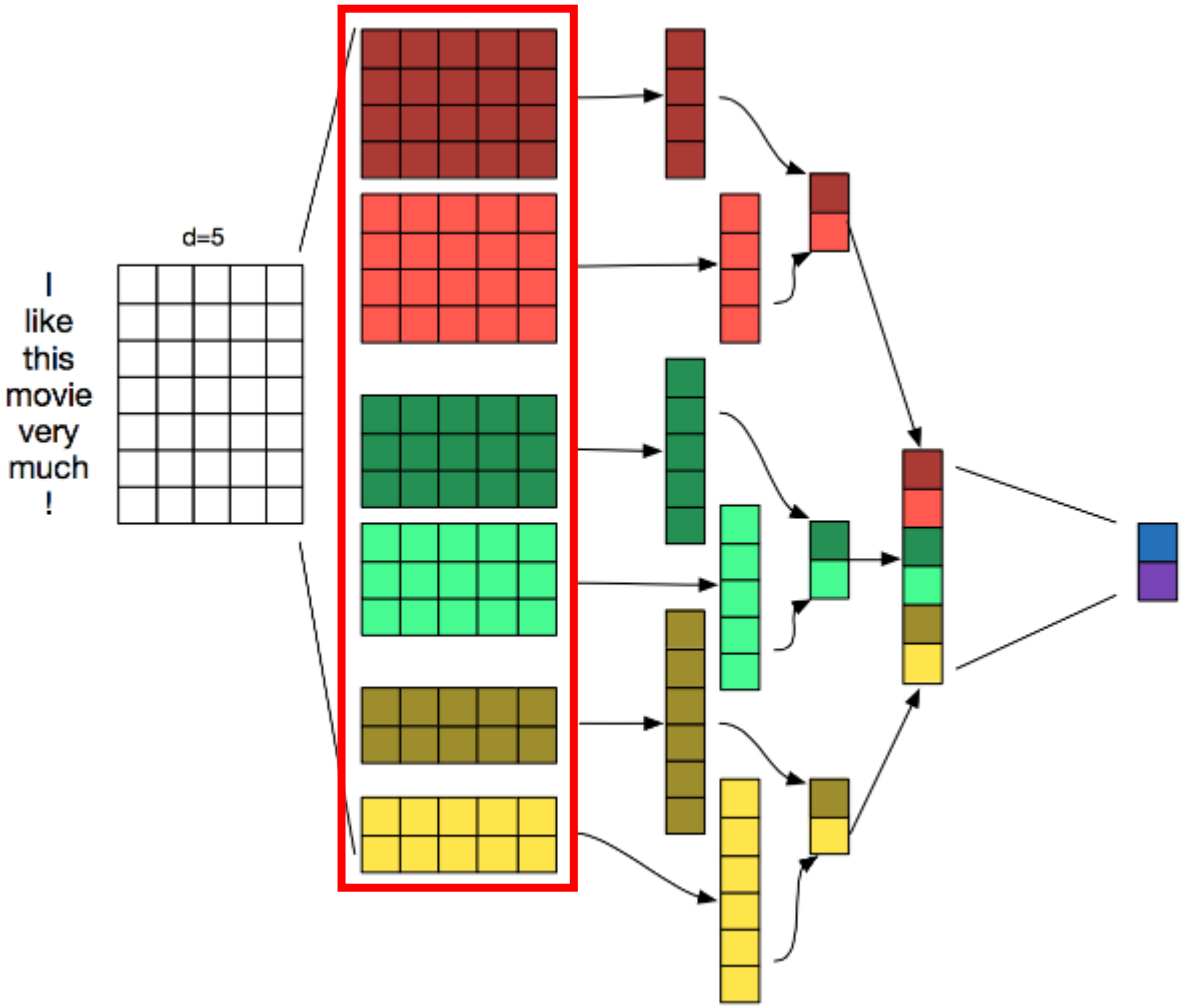
Text CNN Model

CNN for Text Classification





CNN for Text Classification



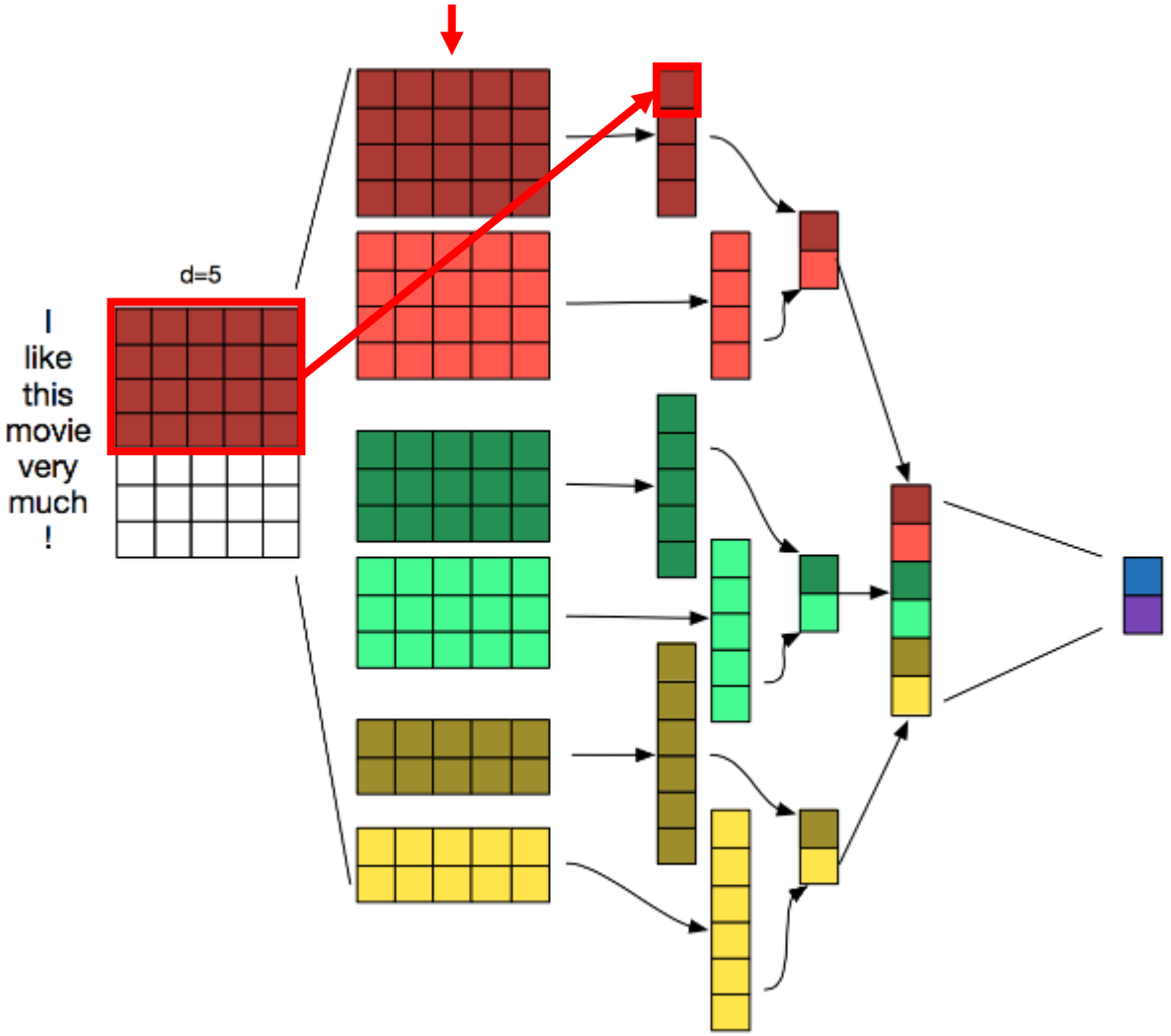
3 region size: (2, 3, 4)

- 각각 bigram, trigram과 4-gram featur를 추출할 수 있다

각 region size마다 2개의 filter

- 같은 region size의 filte를 여러 개 사용할 수 있다. (e.g. bigram feature가 여러 개가 있을 수 있다)

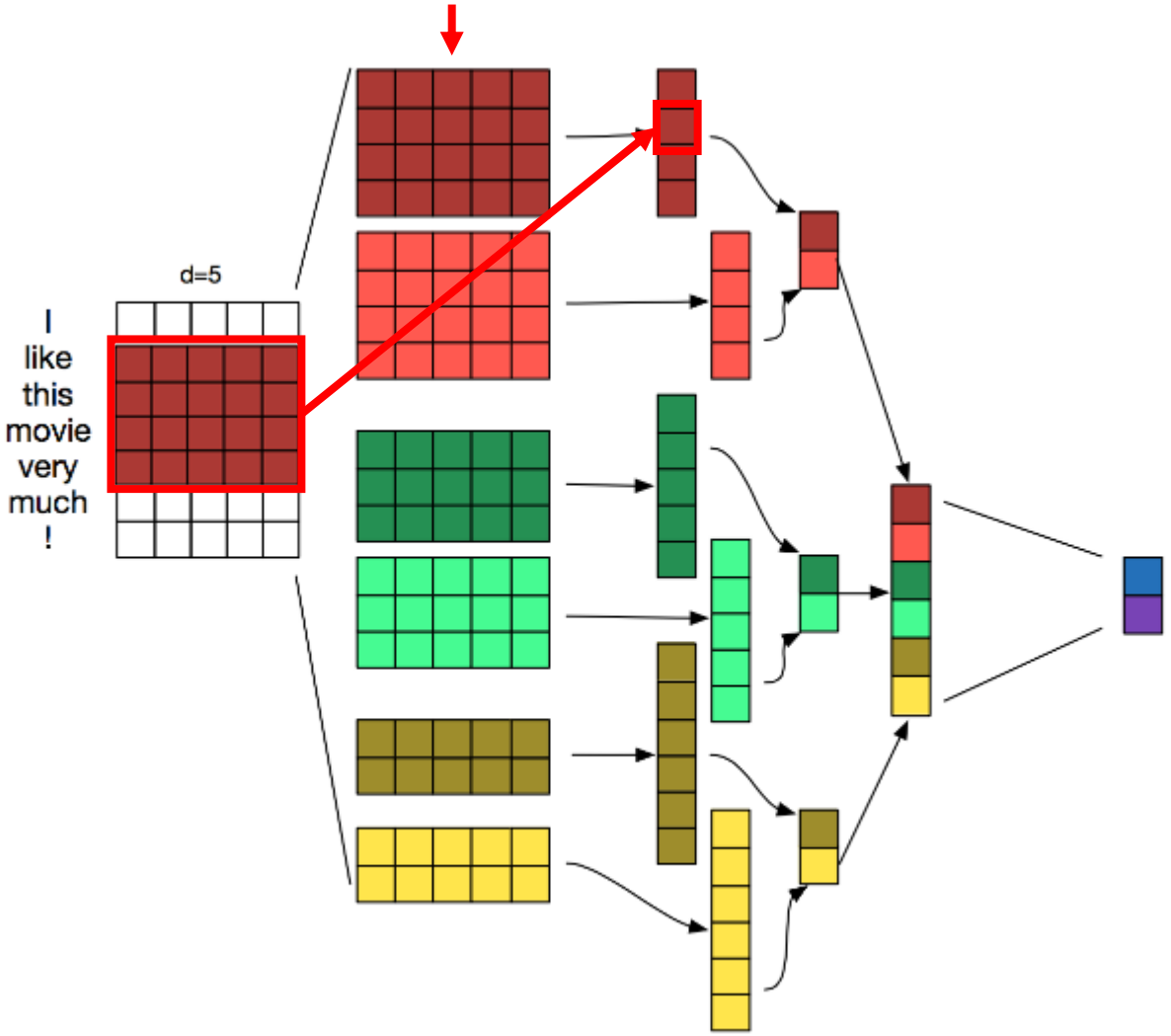
CNN for Text Classification



Convolution

4-gram feature를 추출

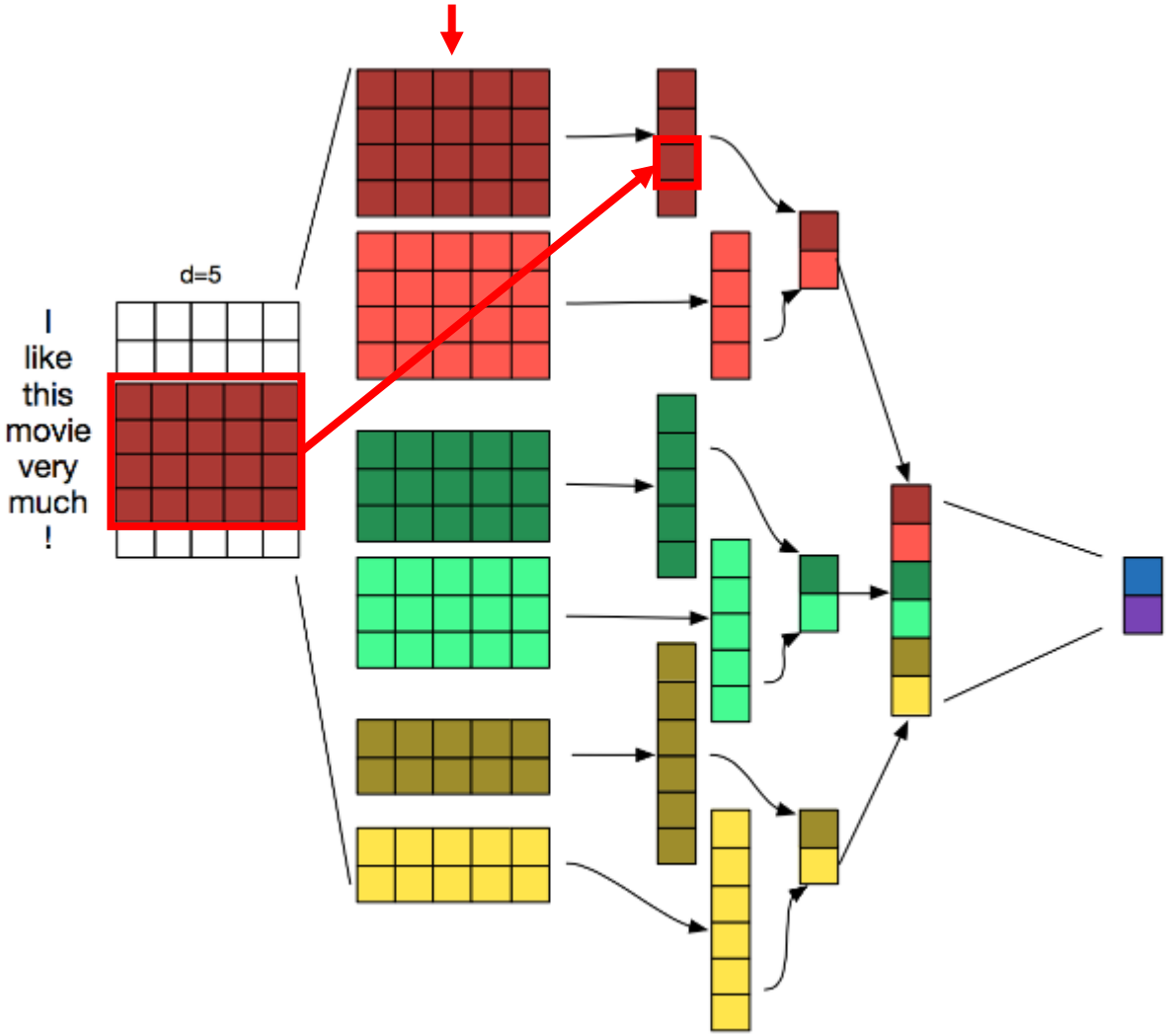
CNN for Text Classification



Convolution

4-gram feature를 추출

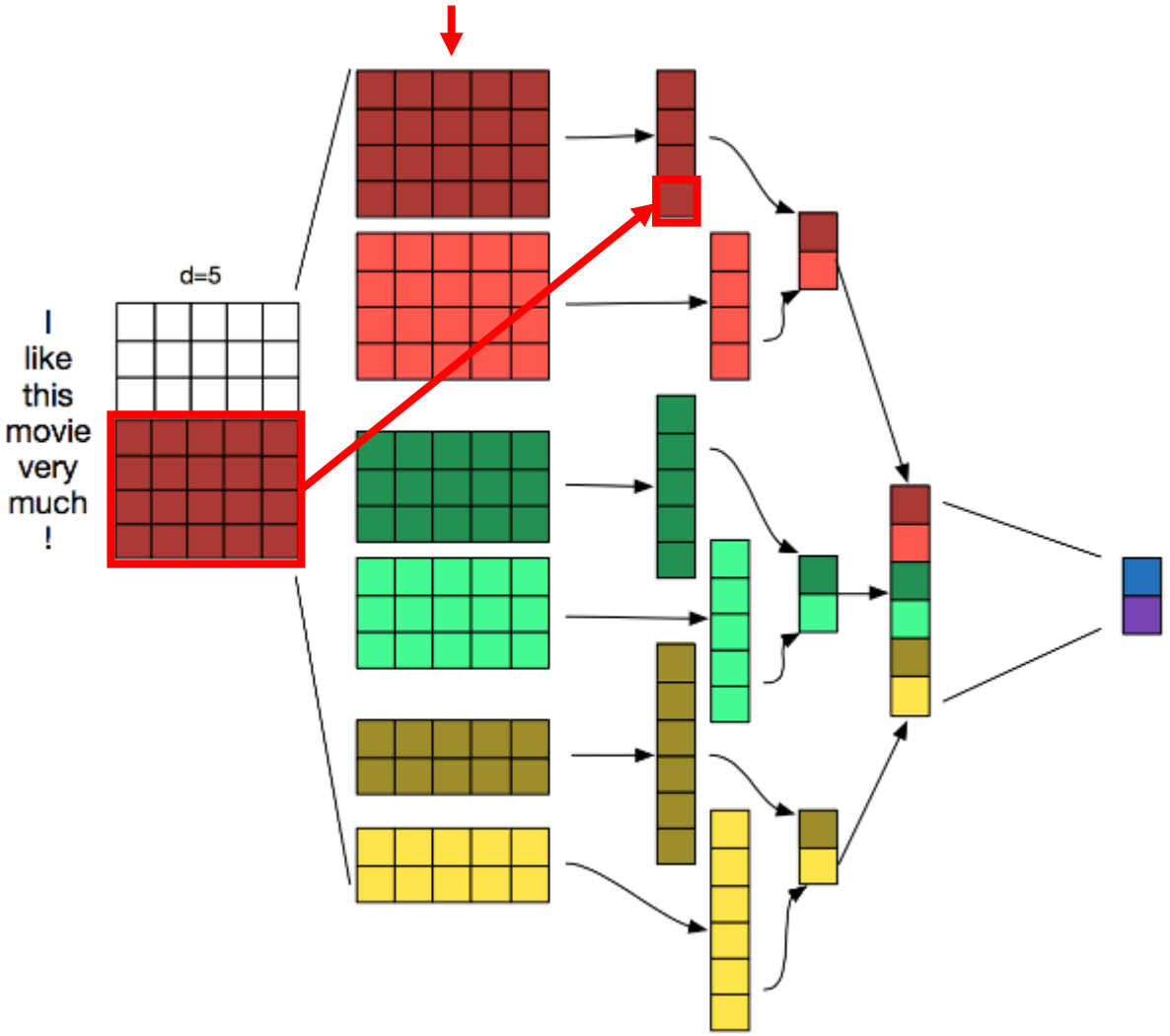
CNN for Text Classification



Convolution

4-gram feature를 추출

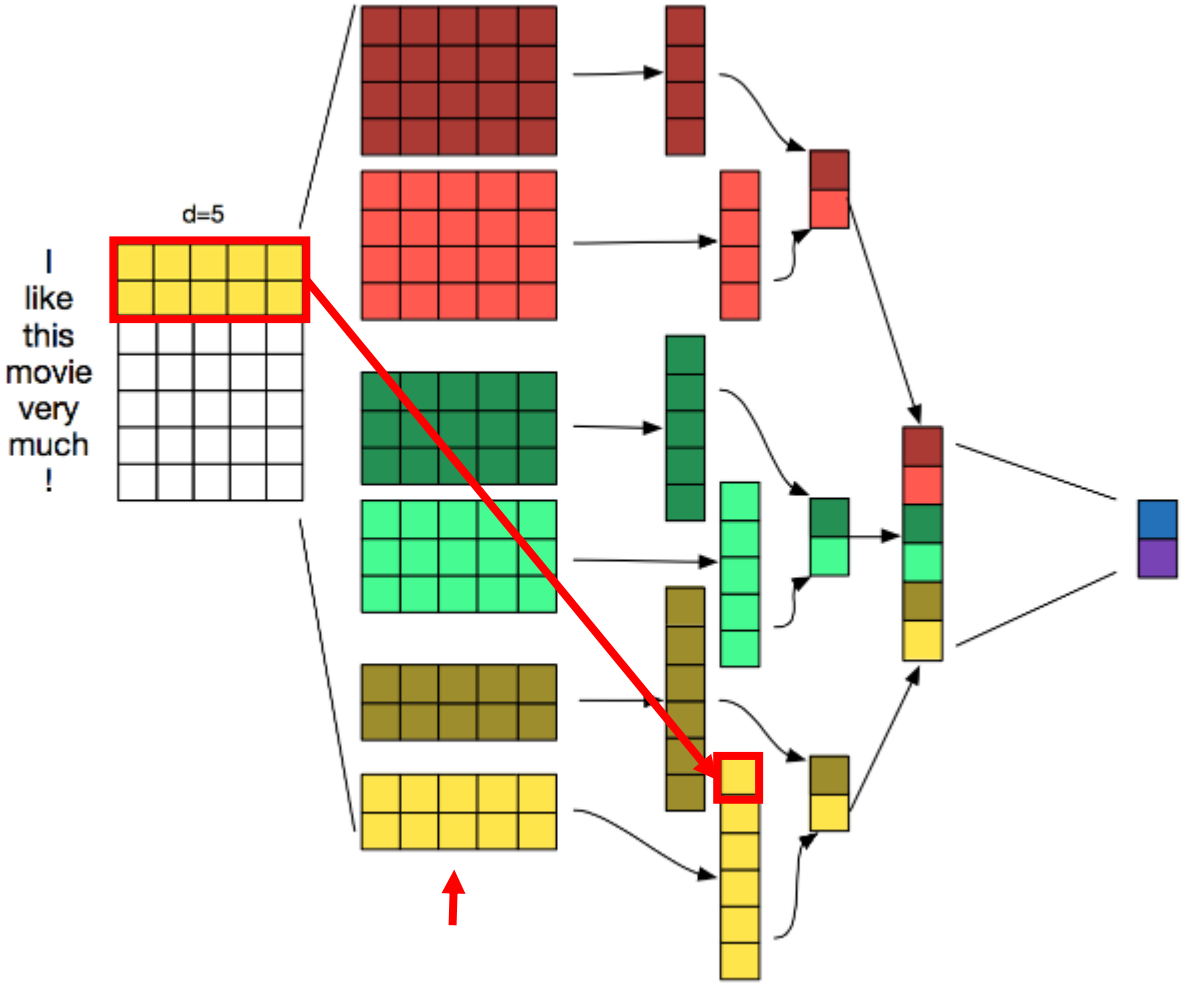
CNN for Text Classification



Convolution

4-gram feature를 추출

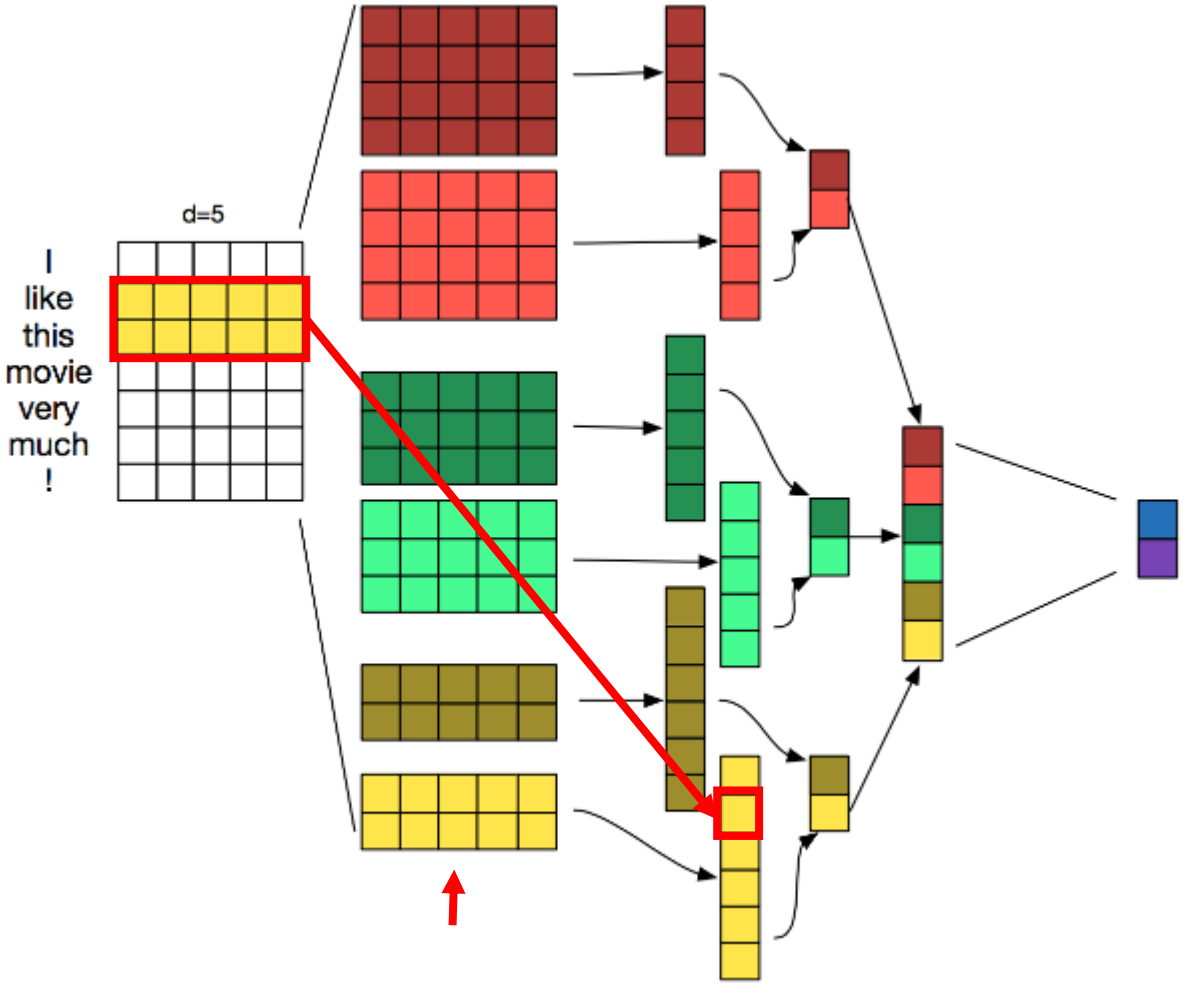
CNN for Text Classification



Convolution

Bigram feature를 추출

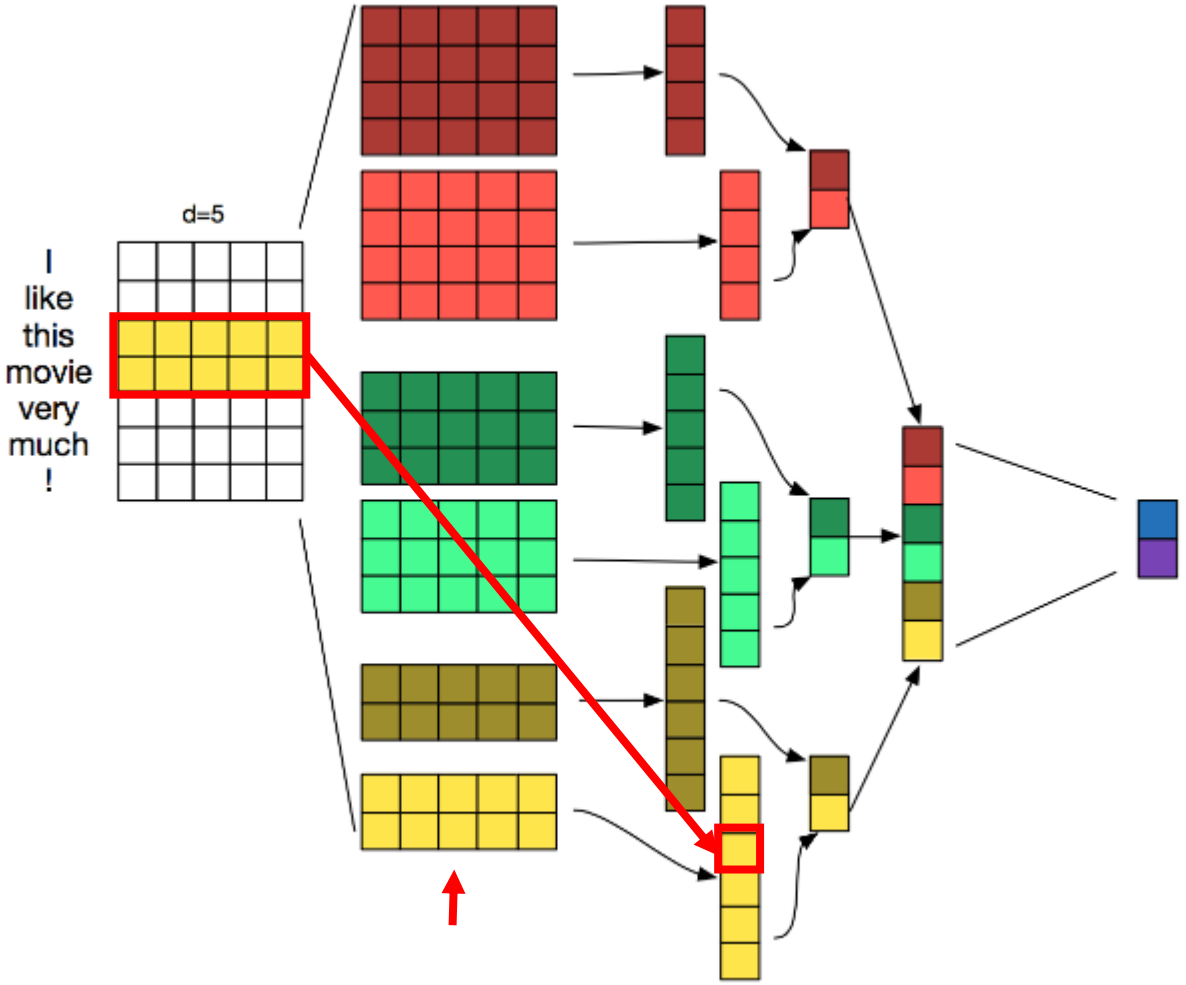
CNN for Text Classification



Convolution

Bigram feature를 추출

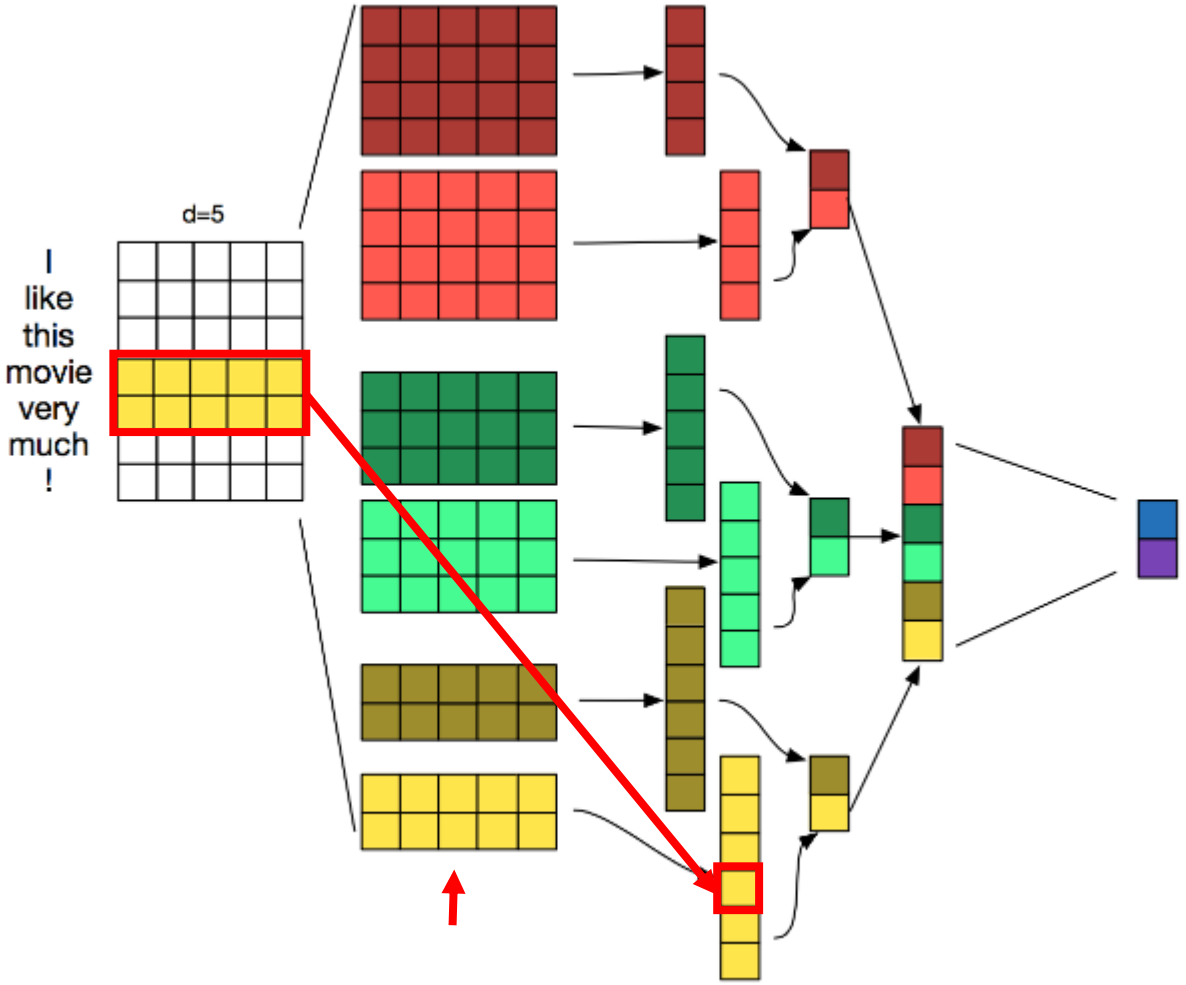
CNN for Text Classification



Convolution

Bigram feature를 추출

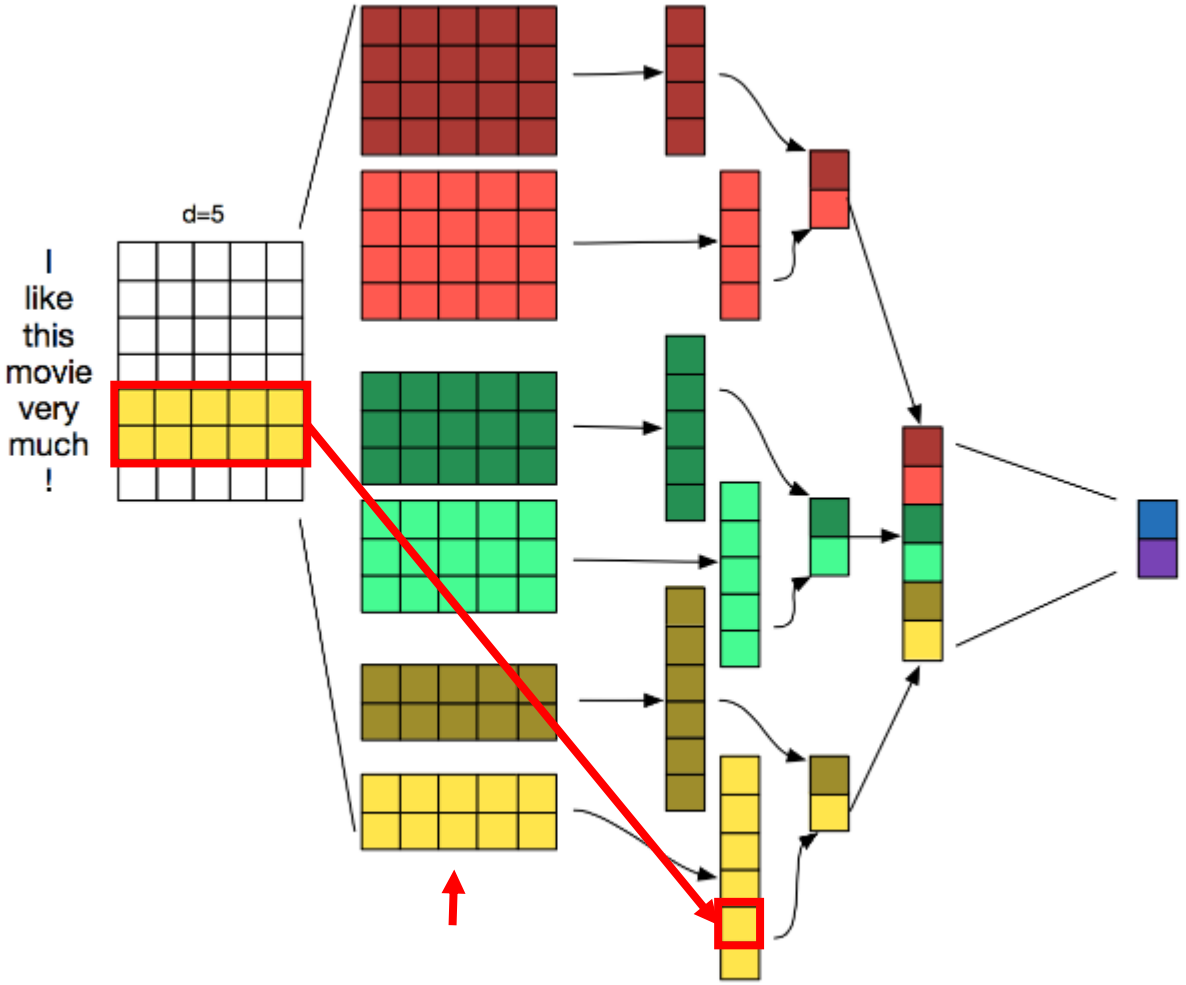
CNN for Text Classification



Convolution

Bigram feature를 추출

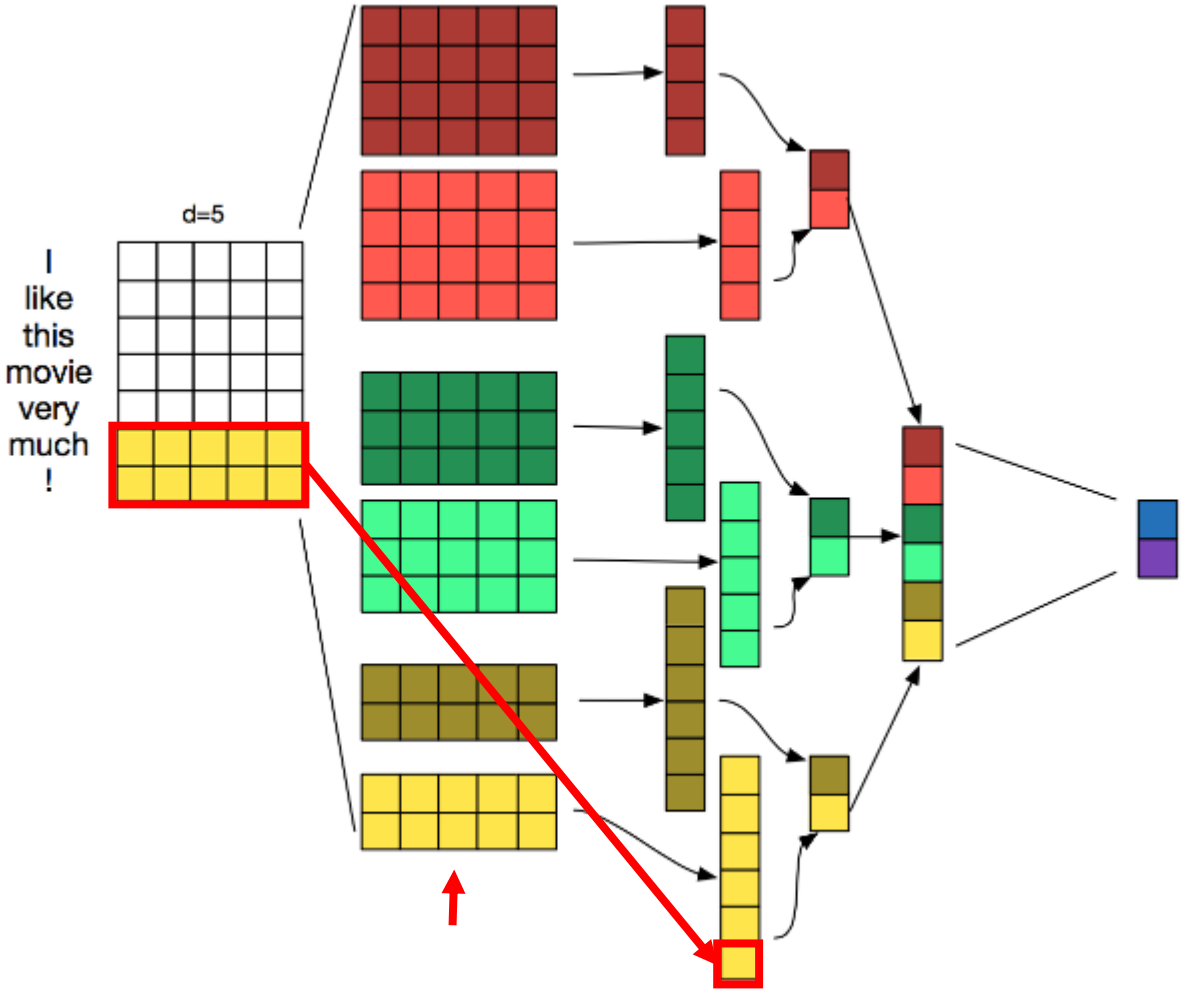
CNN for Text Classification



Convolution

Bigram feature를 추출

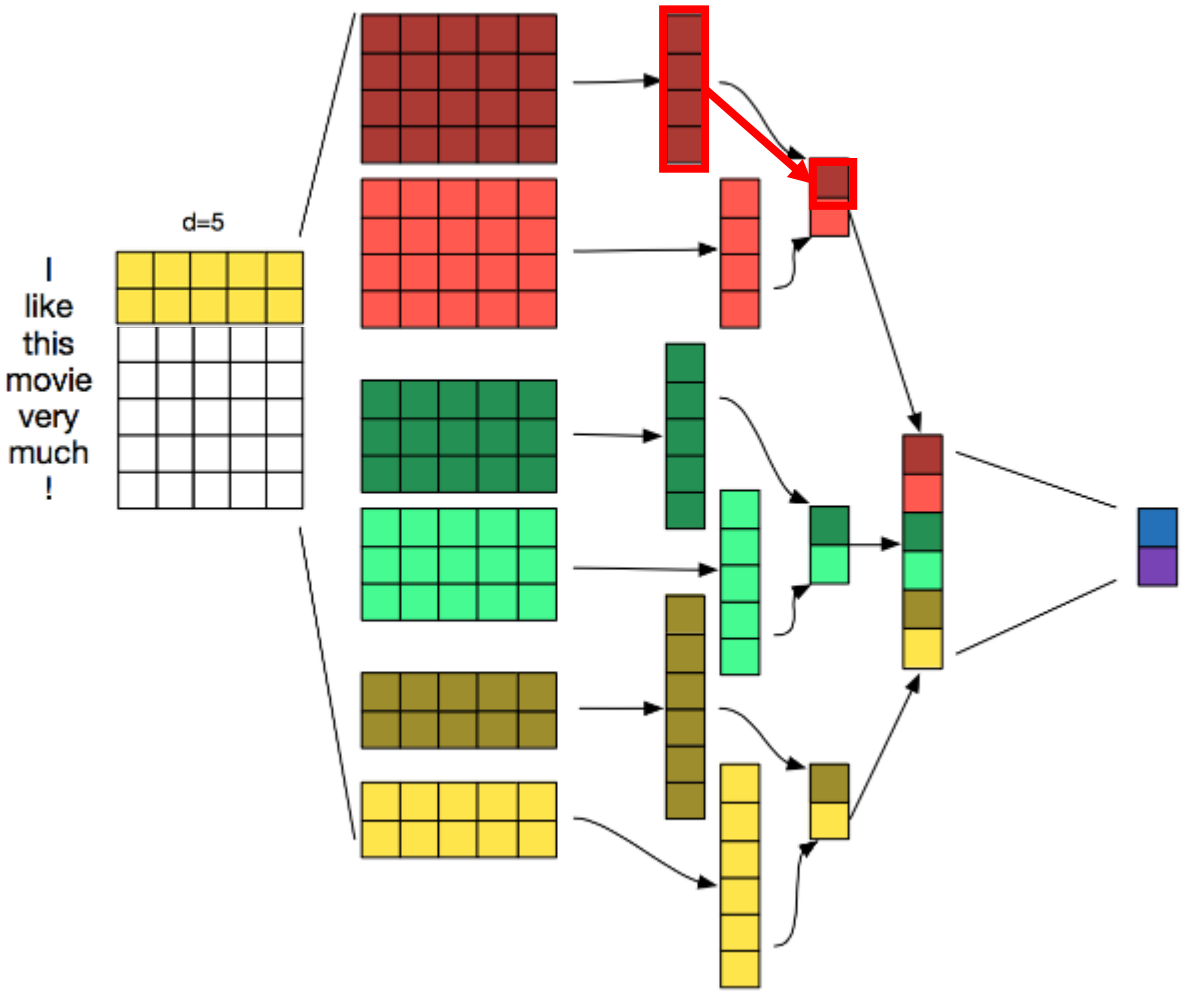
CNN for Text Classification



Convolution

Bigram feature를 추출

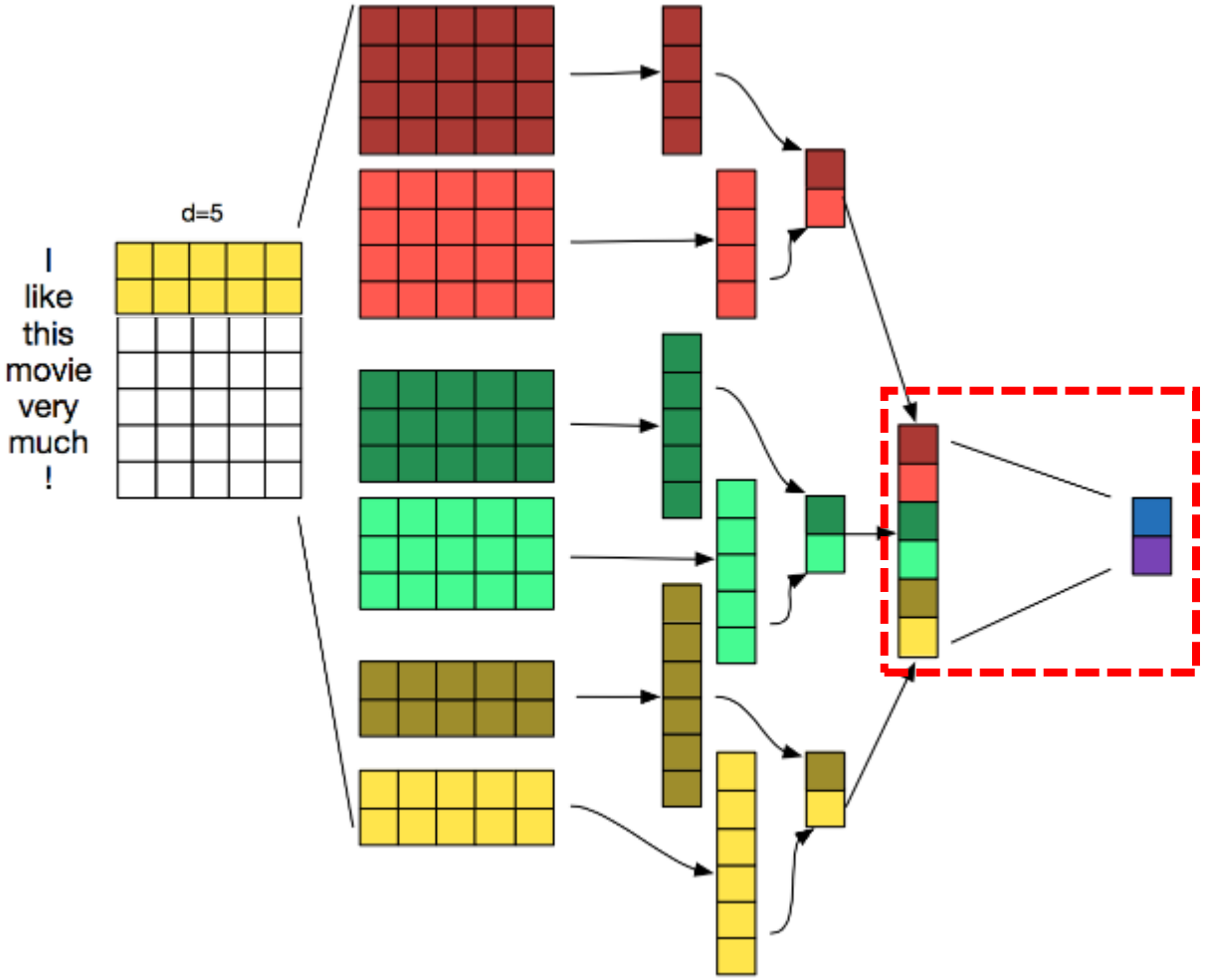
CNN for Text Classification



Max Pooling

max-over-time

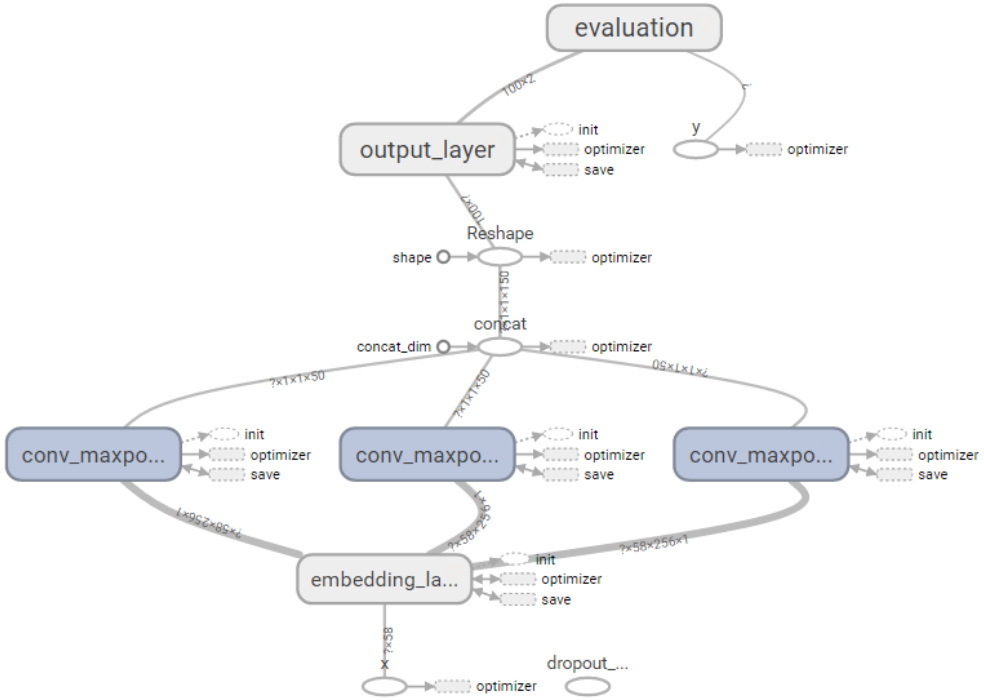
CNN for Text Classification



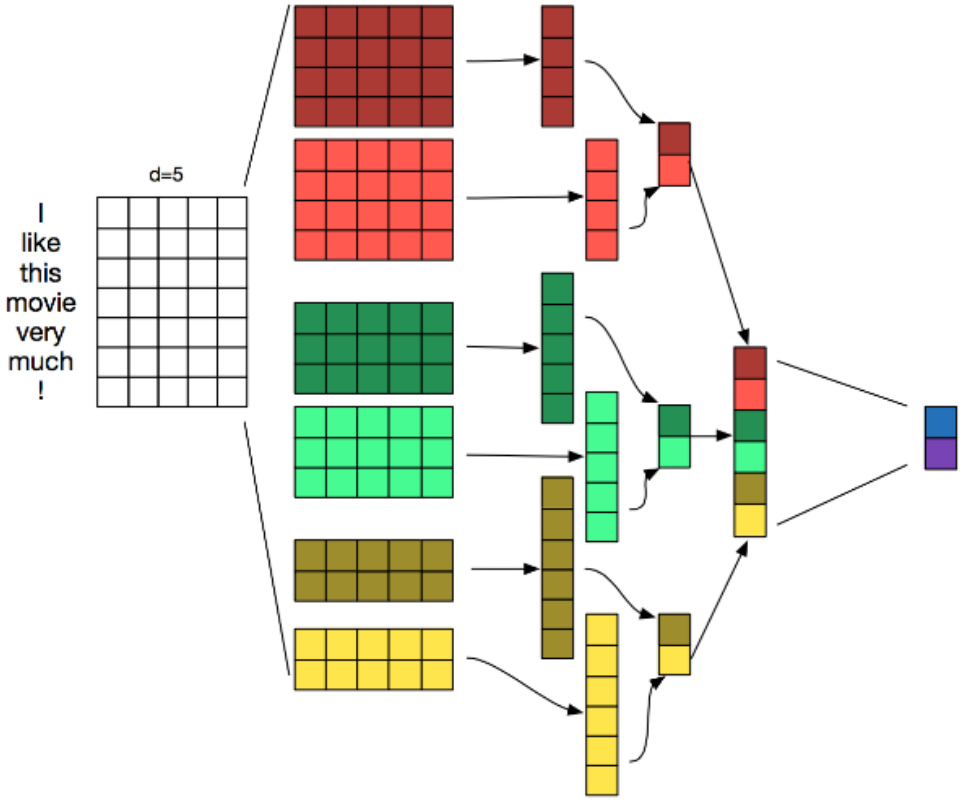
Output layer

Classification (e.g. binary classification for pos/neg)

CNN with Text Classification



Tensorboard (our model)

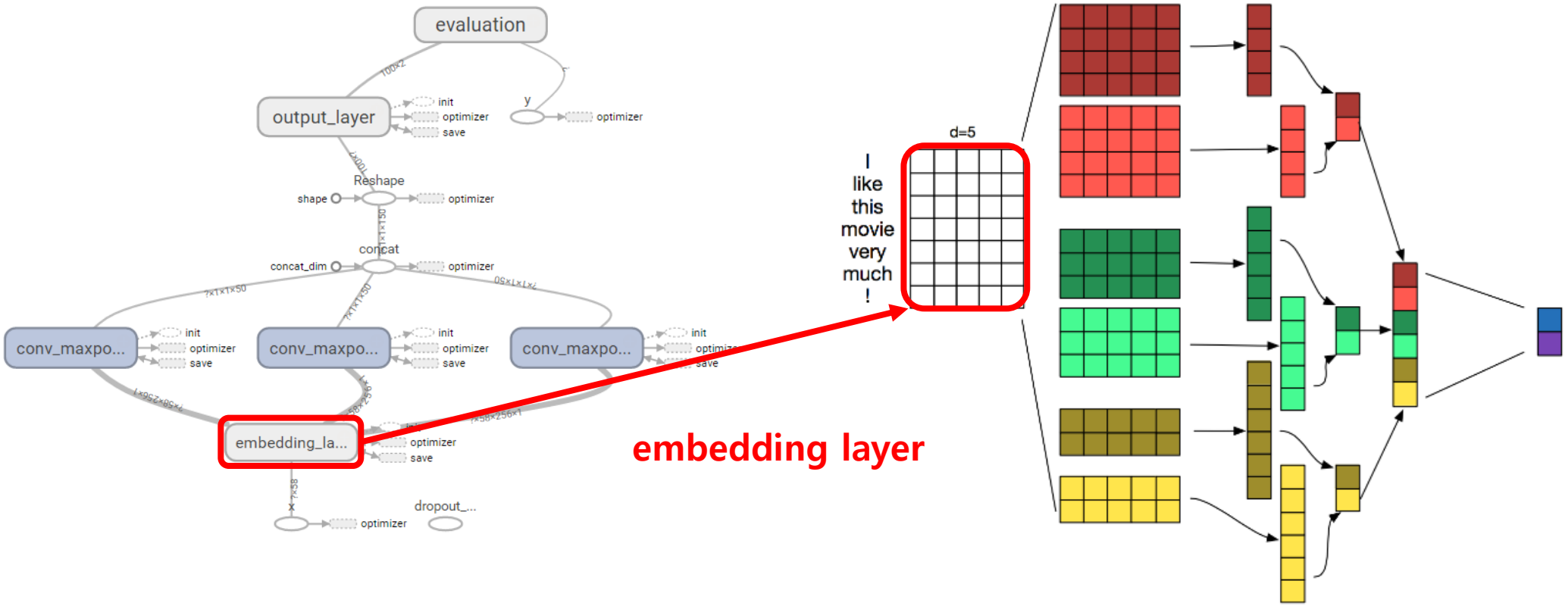


TextCNN (2015)

CNN with Text Classification

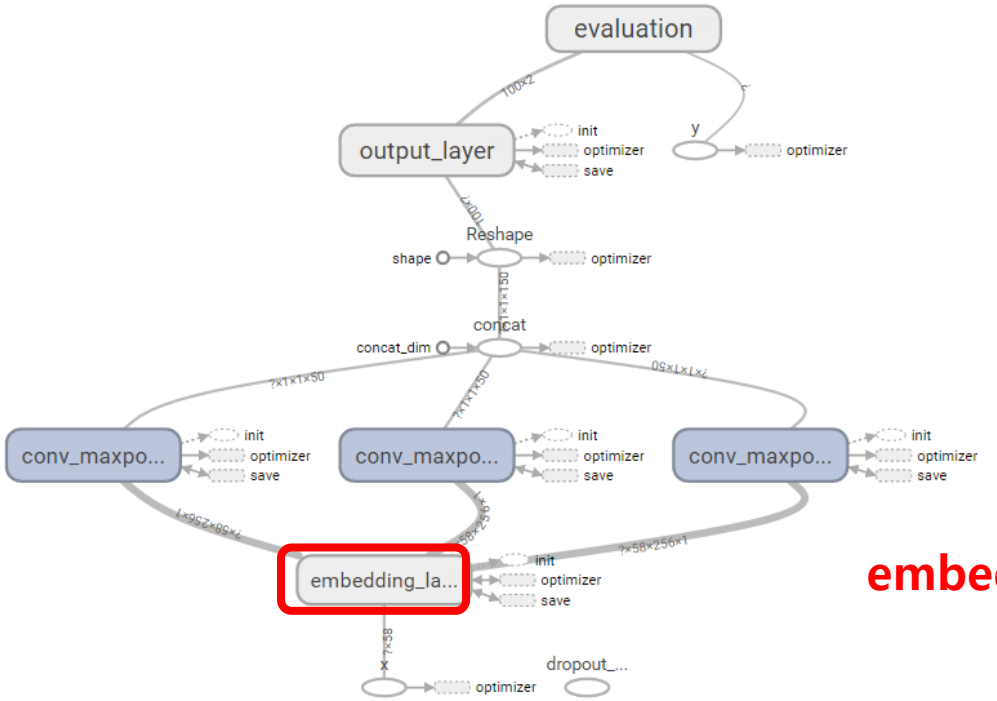
Embedding Layer

CNN with Text Classification



CNN with Text Classification

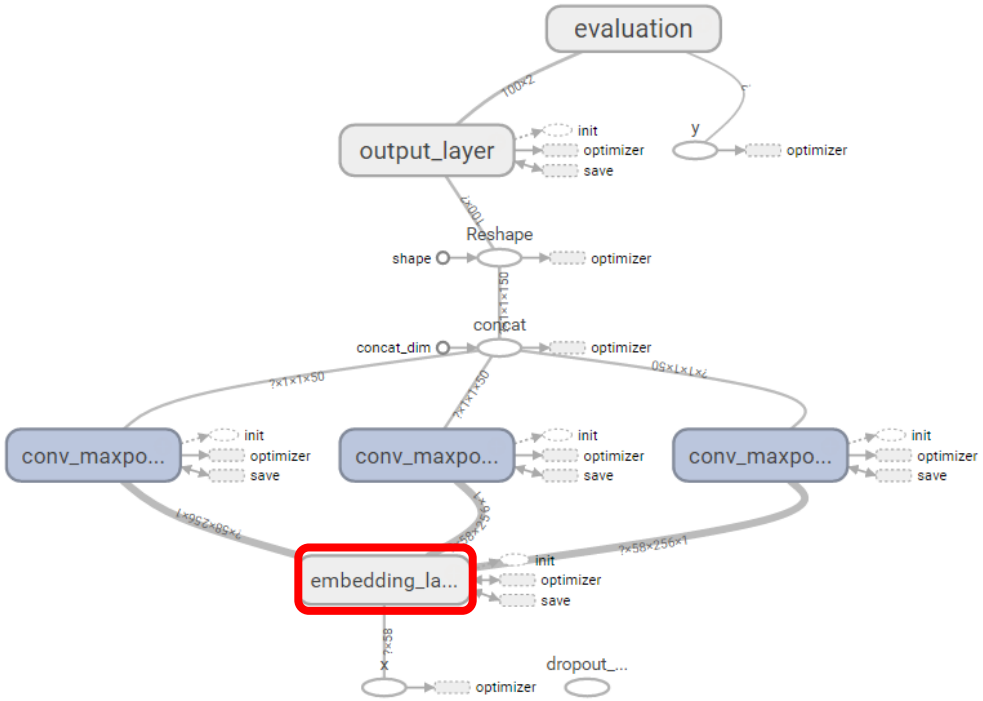
```
with tf.variable_scope('embedding_layer'):
    w = tf.get_variable('w', shape=[vocab_size, dim_emb], initializer=tf.random_uniform_initializer(-1, 1))
    x_embed = tf.nn.embedding_lookup(w, self.x)    # (batch_size, seq_length, dim_emb)
    x_embed = tf.expand_dims(x_embed, 3)           # (batch_size, seq_length, dim_emb, 1)
```



embedding layer

CNN with Text Classification

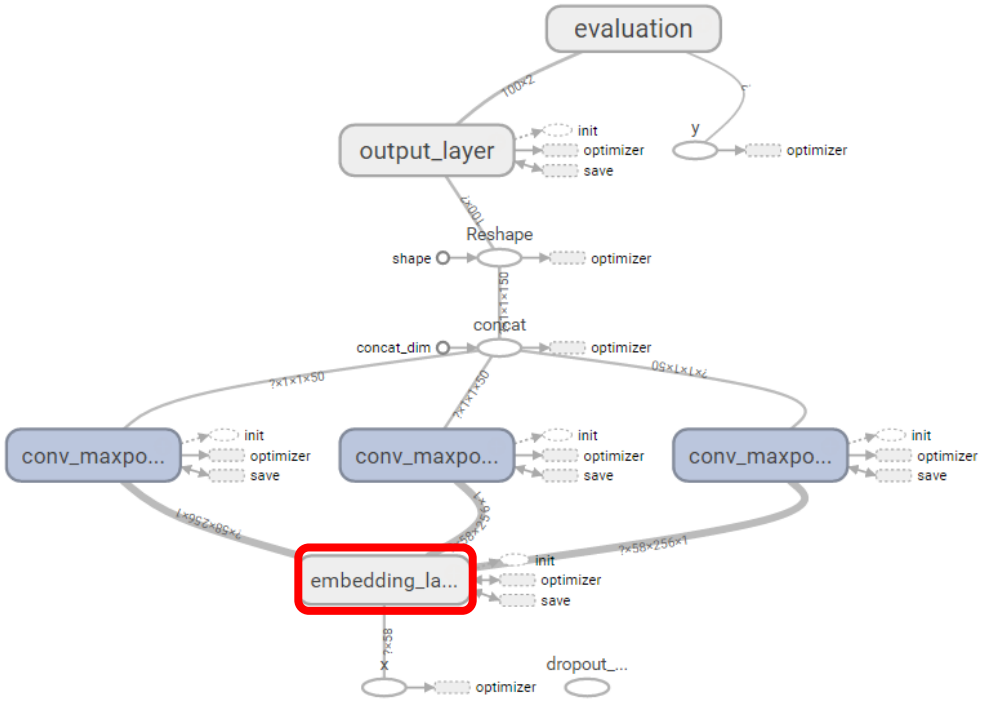
```
with tf.variable_scope('embedding_layer'):
    w = tf.get_variable('w', shape=[vocab_size, dim_emb], initializer=tf.random_uniform_initializer(-1, 1))
    x_embed = tf.nn.embedding_lookup(w, self.x)    # (batch_size, seq_length, dim_emb)
    x_embed = tf.expand_dims(x_embed, 3)           # (batch_size, seq_length, dim_emb, 1)
```



Embedding matrix를 random하게 초기화

CNN with Text Classification

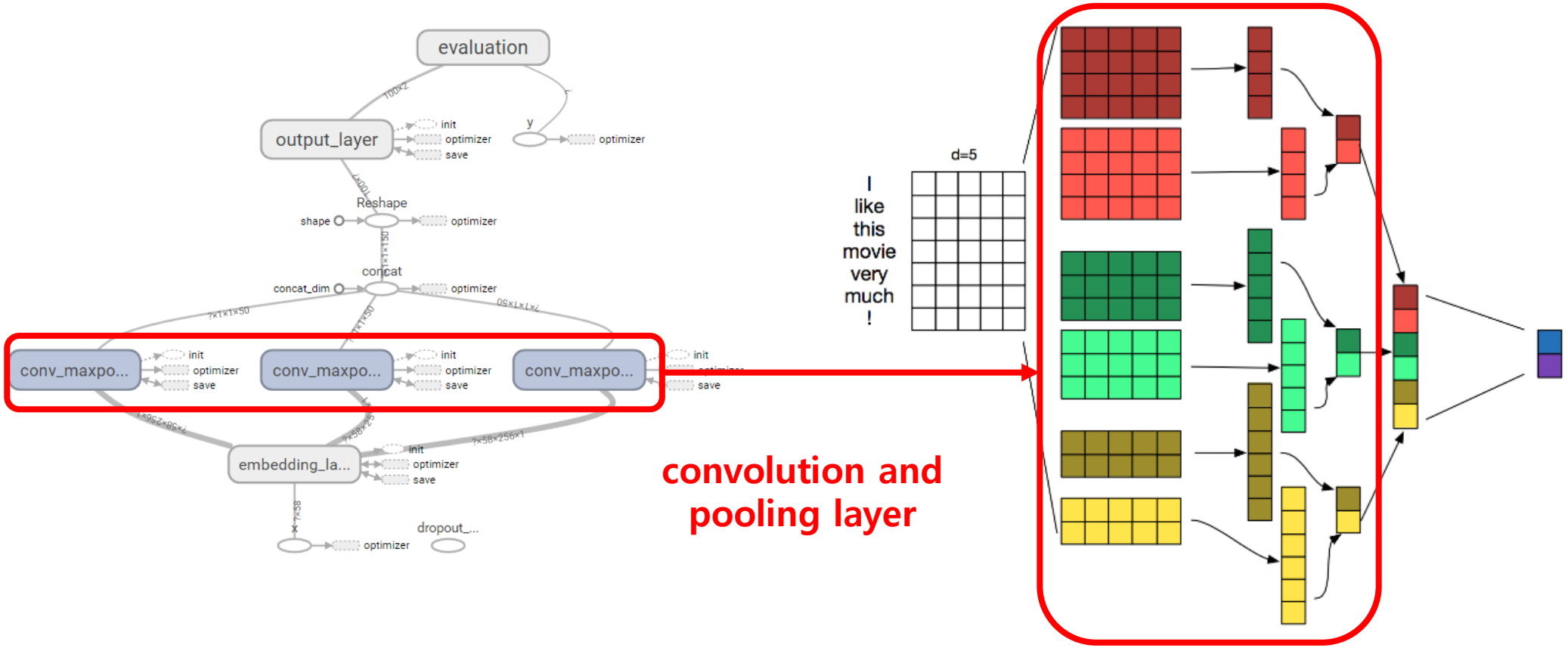
```
with tf.variable_scope('embedding_layer'):
    w = tf.get_variable('w', shape=[vocab_size, dim_emb], initializer=tf.random_uniform_initializer(-1, 1))
    x_embed = tf.nn.embedding_lookup(w, self.x)    # (batch_size, seq_length, dim_emb)
    x_embed = tf.expand_dims(x_embed, 3)           # (batch_size, seq_length, dim_emb, 1)
```



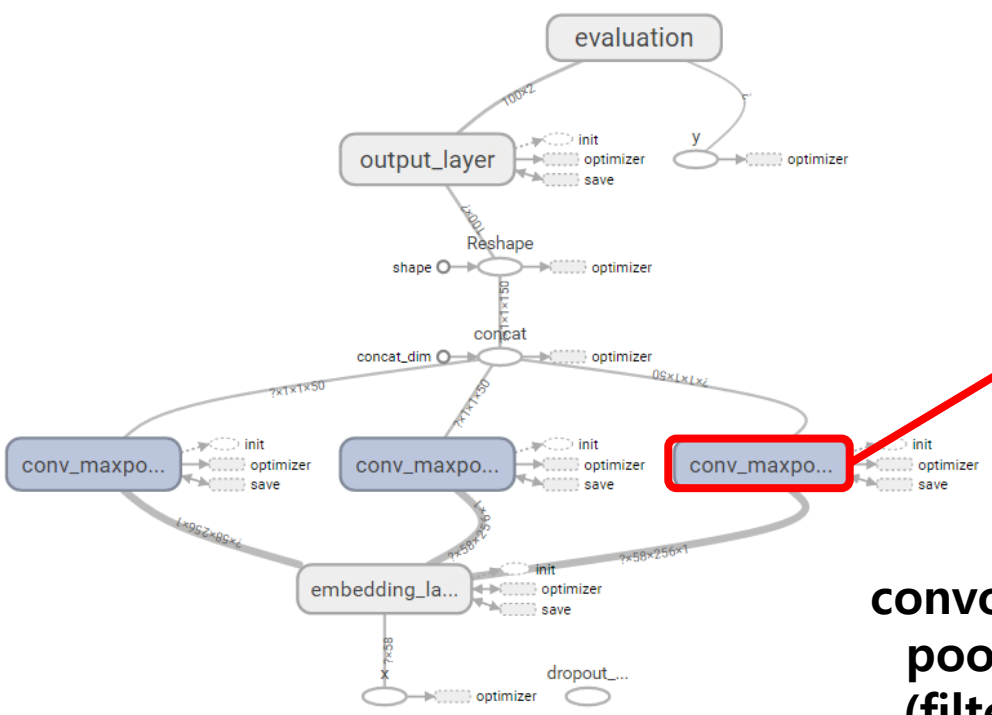
Embedding vector를 추출후 4차원형태로 변환

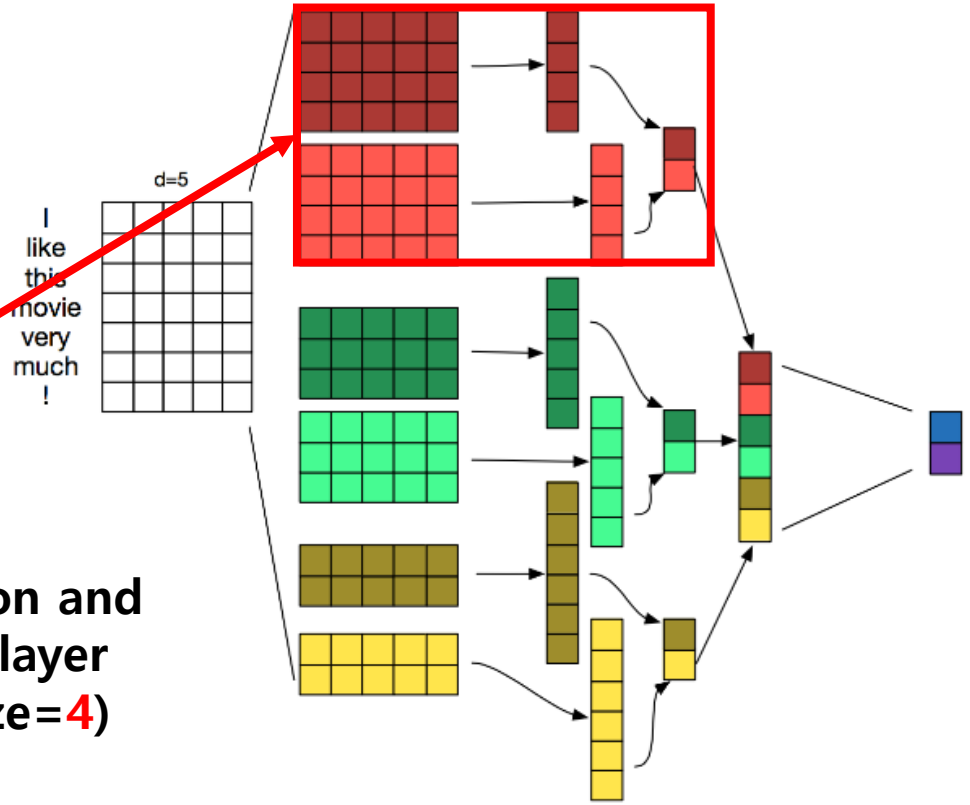
Convolution and Pooling Layer

CNN with Text Classification



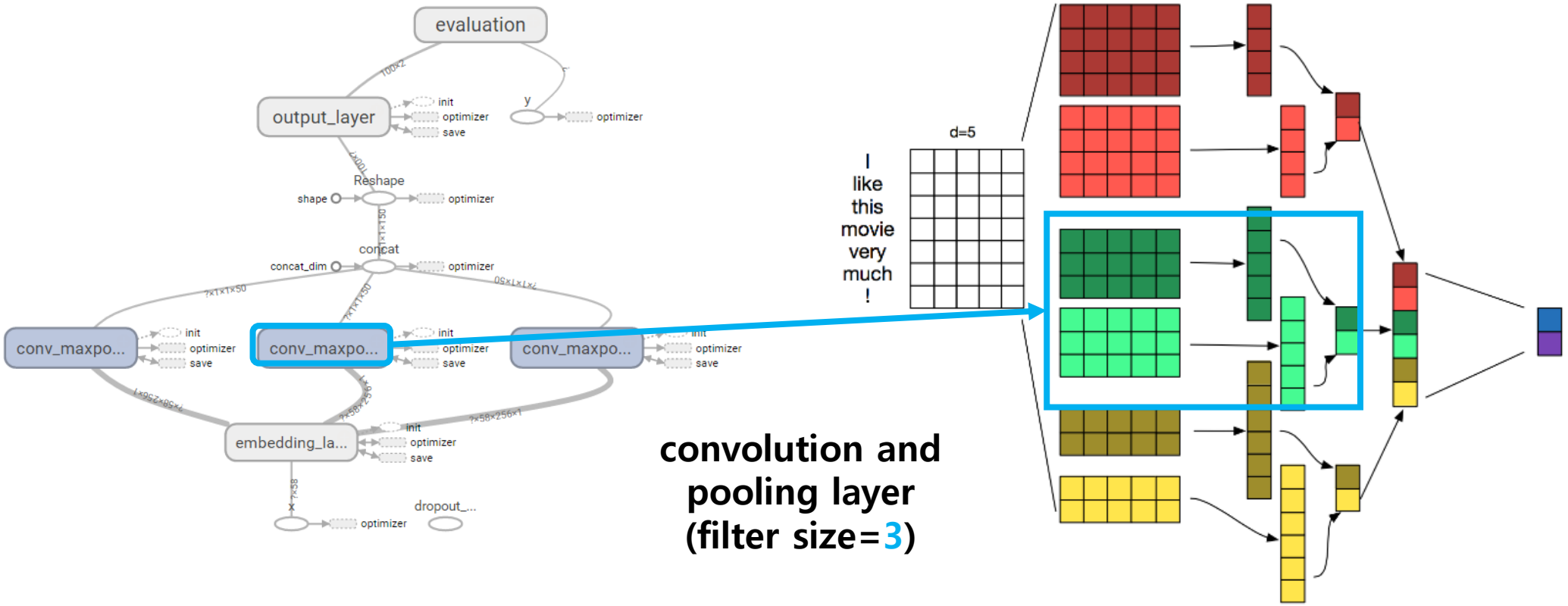
CNN with Text Classification



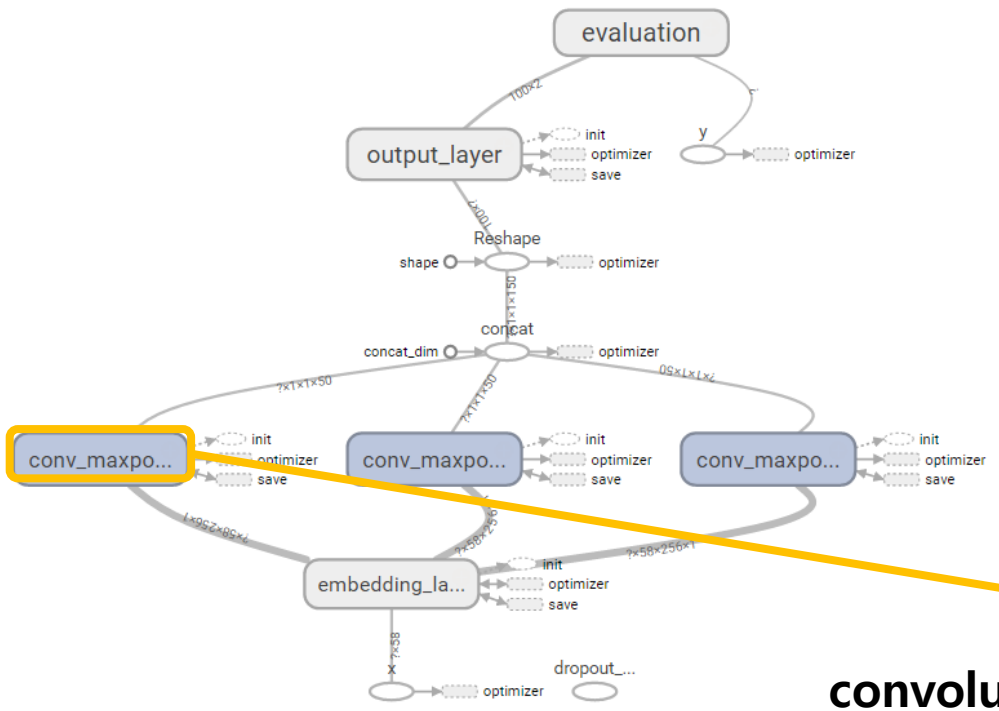


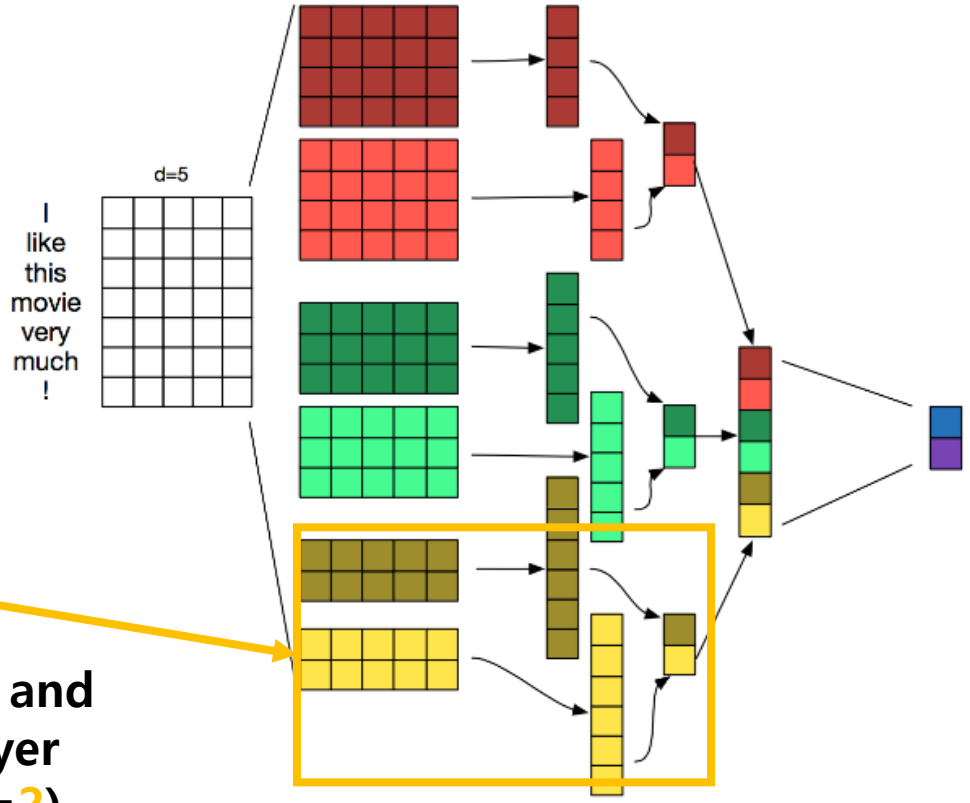
convolution and pooling layer (filter size=4)

CNN with Text Classification

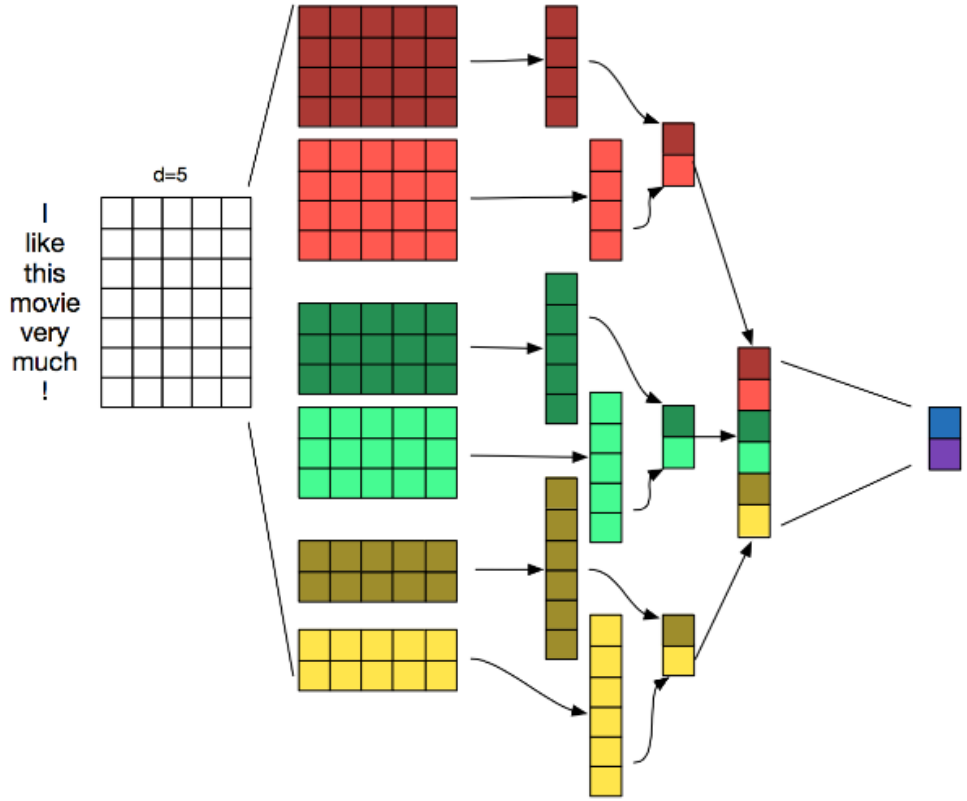


CNN with Text Classification

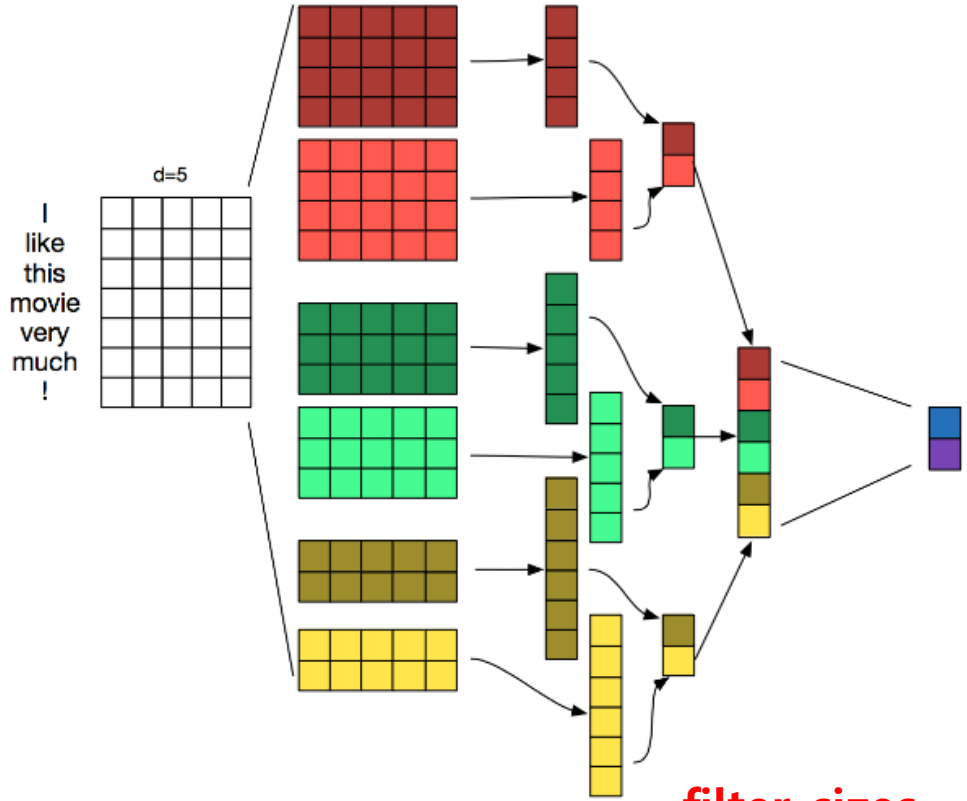




convolution and pooling layer (filter size=2)




```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

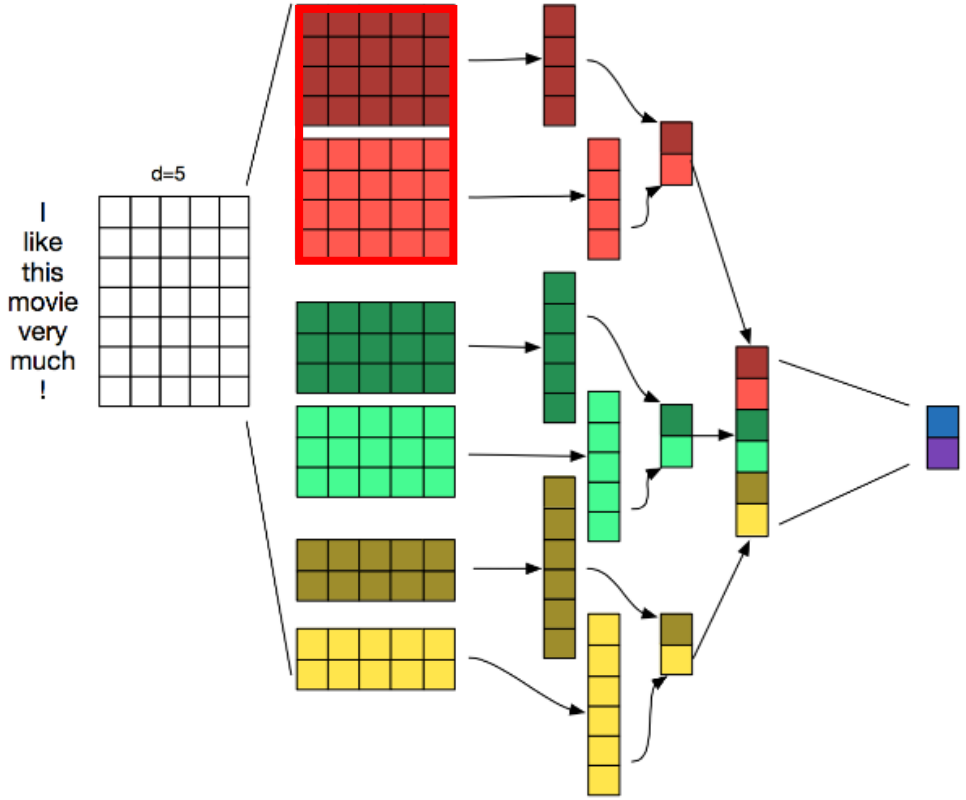


filter_sizes = [2, 3, 4] 로 총 3번의 iteration 을 돌게 된다

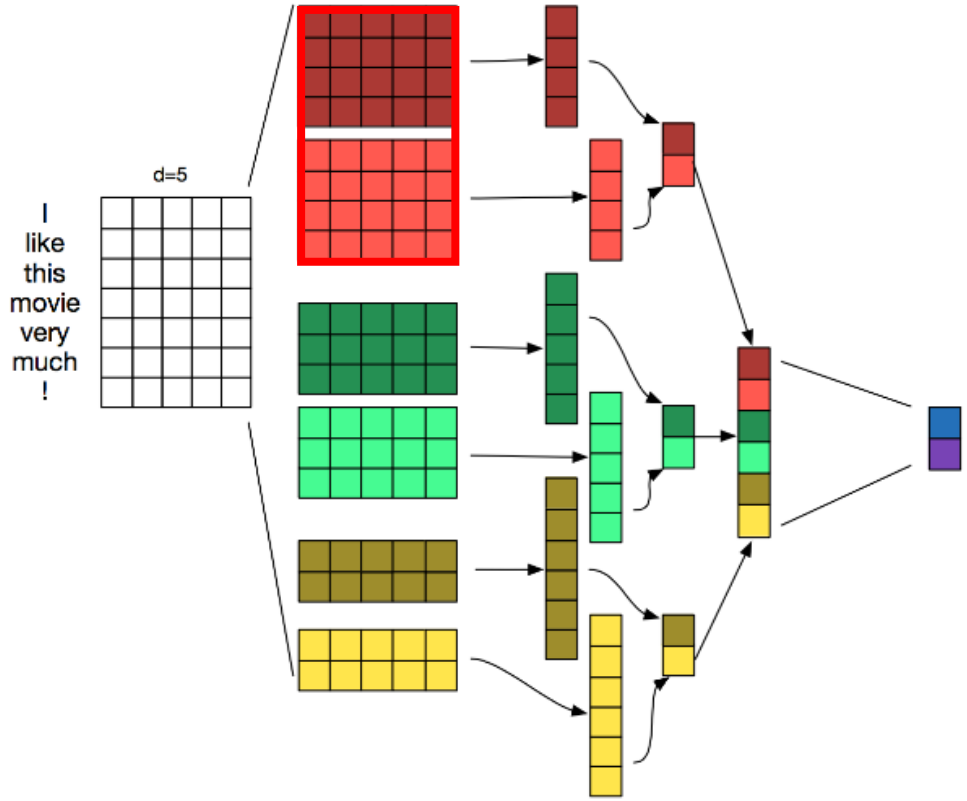
```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

그림의 경우

filter_size = 4
dim_emb = 5
num_filters = 2

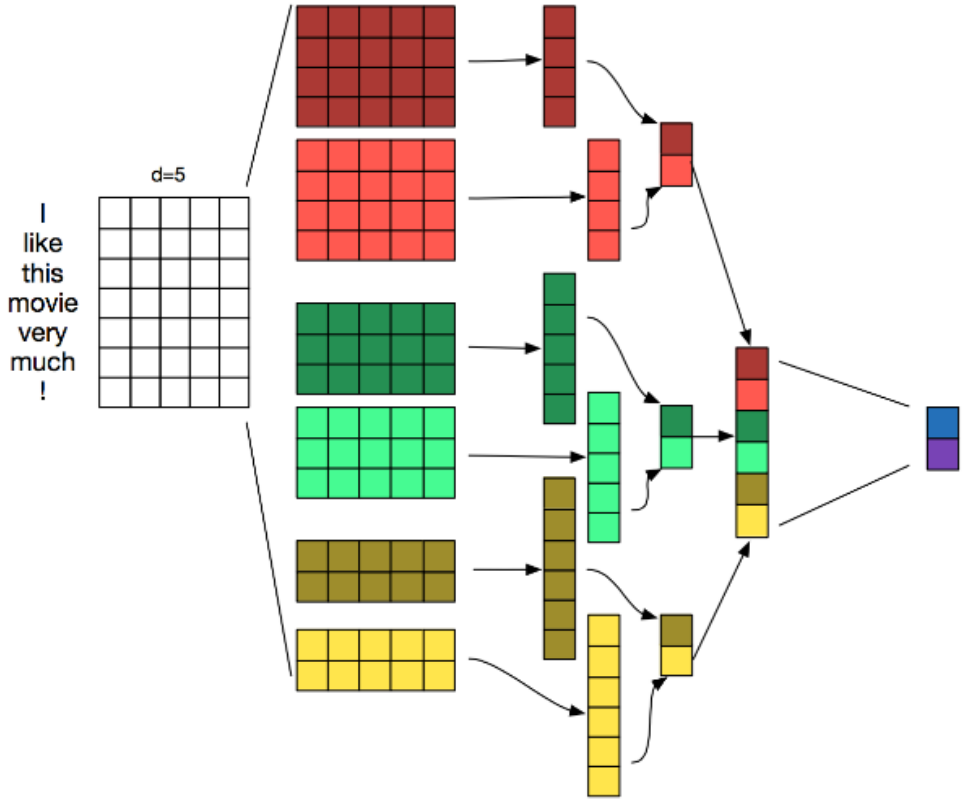



```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```




```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

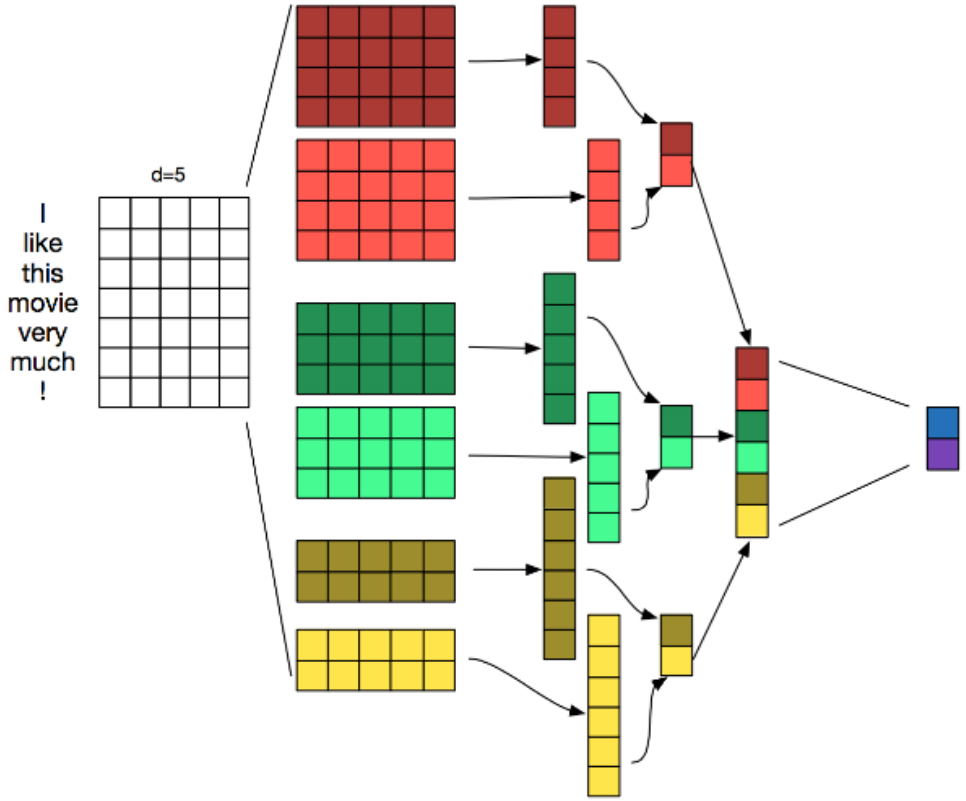
weight(filter)와 bias 생성




```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

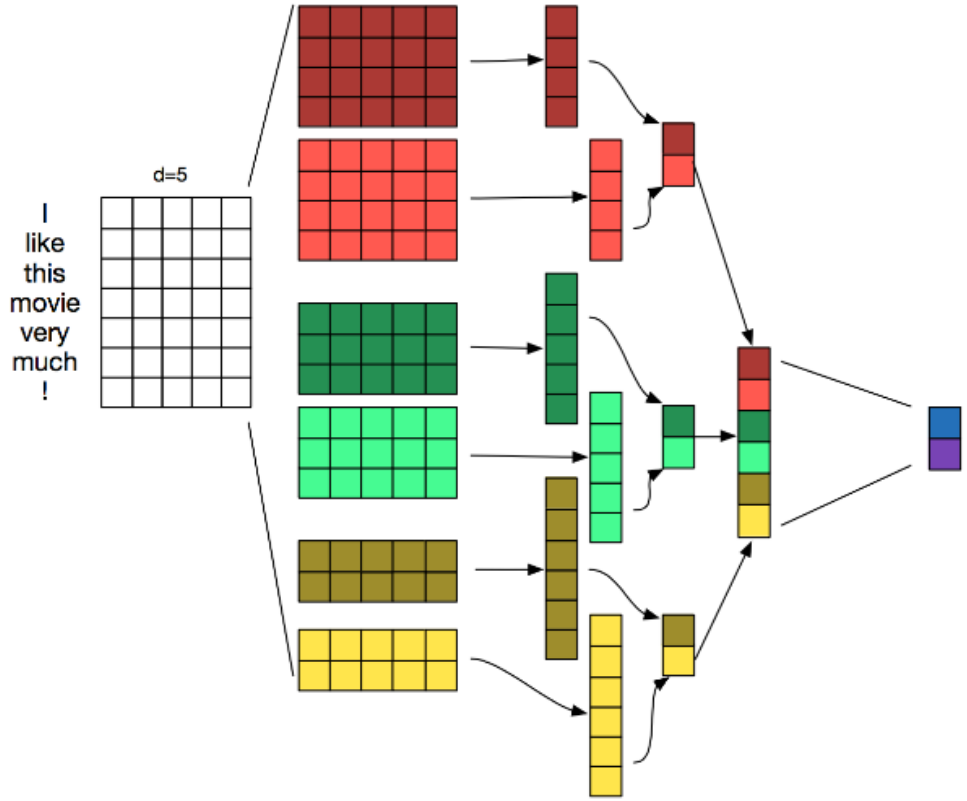
convolution 연산

그림의 경우
seq_length = 7
dim_emb = 5




```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

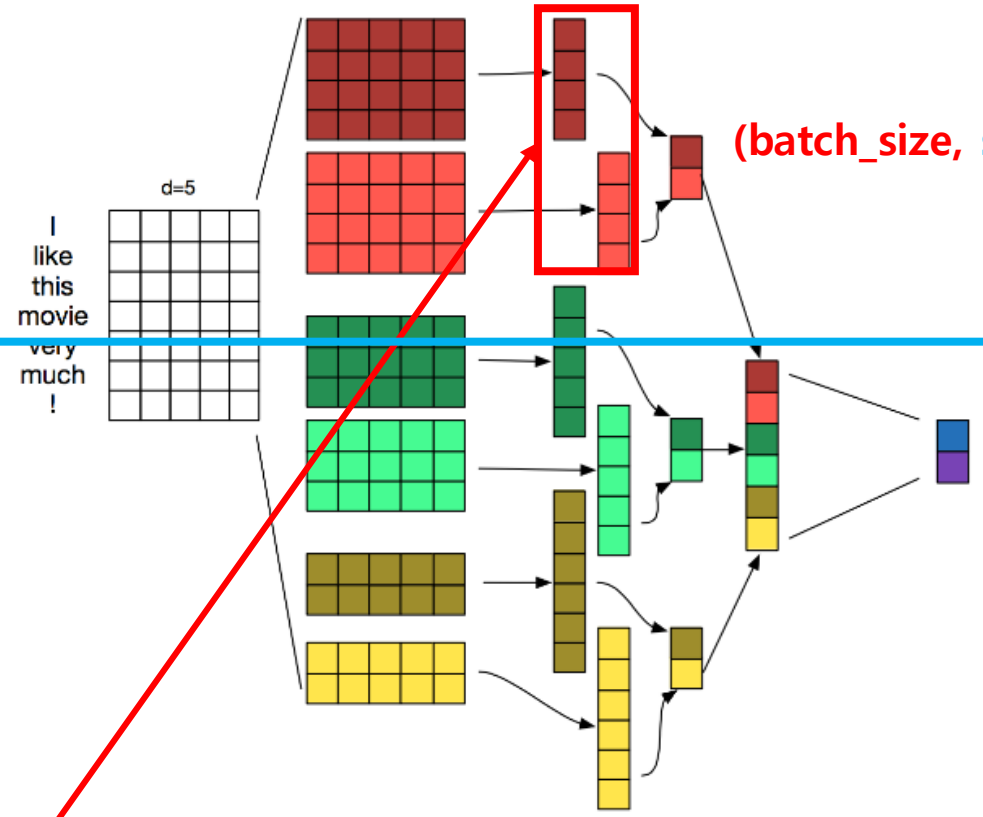
shape = (batch_size, seq_length, dim_emb, 1)



(중요) Convolution 연산 후 activation map의 shape은 어떻게 될까?

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

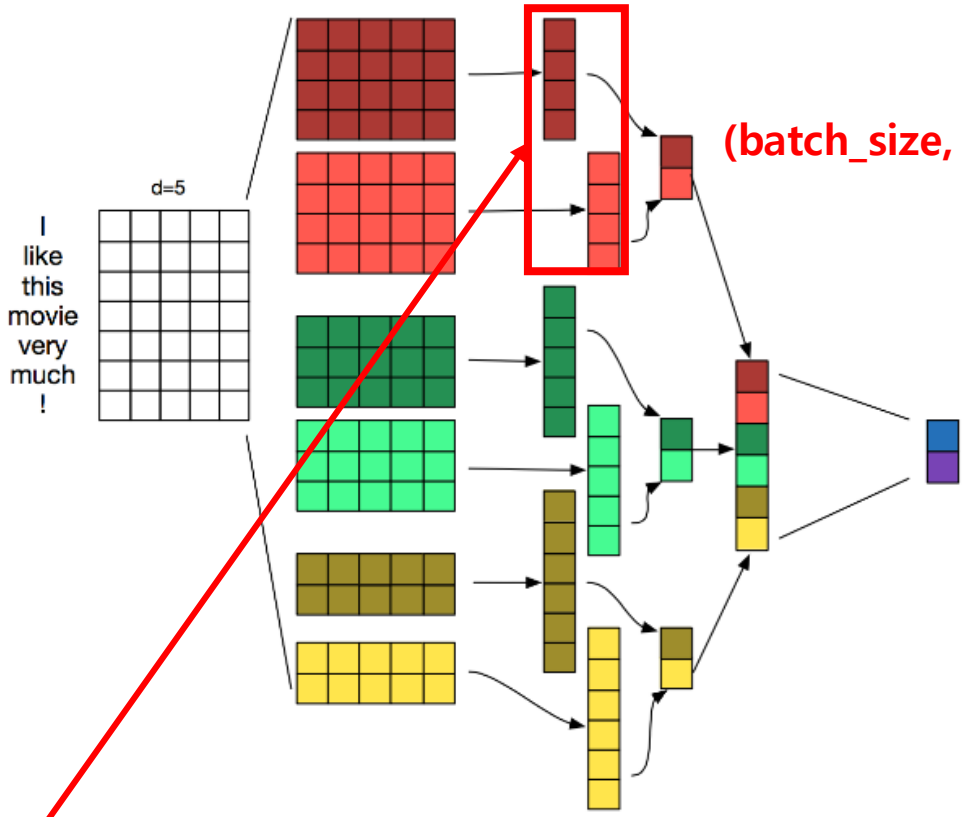
그림의 경우
num_filters=2
seq_length = 7
filter_size = 4
7 – 4 + 1 = 4



(batch_size, seq_length – filter_size + 1, 1, num_filters)

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

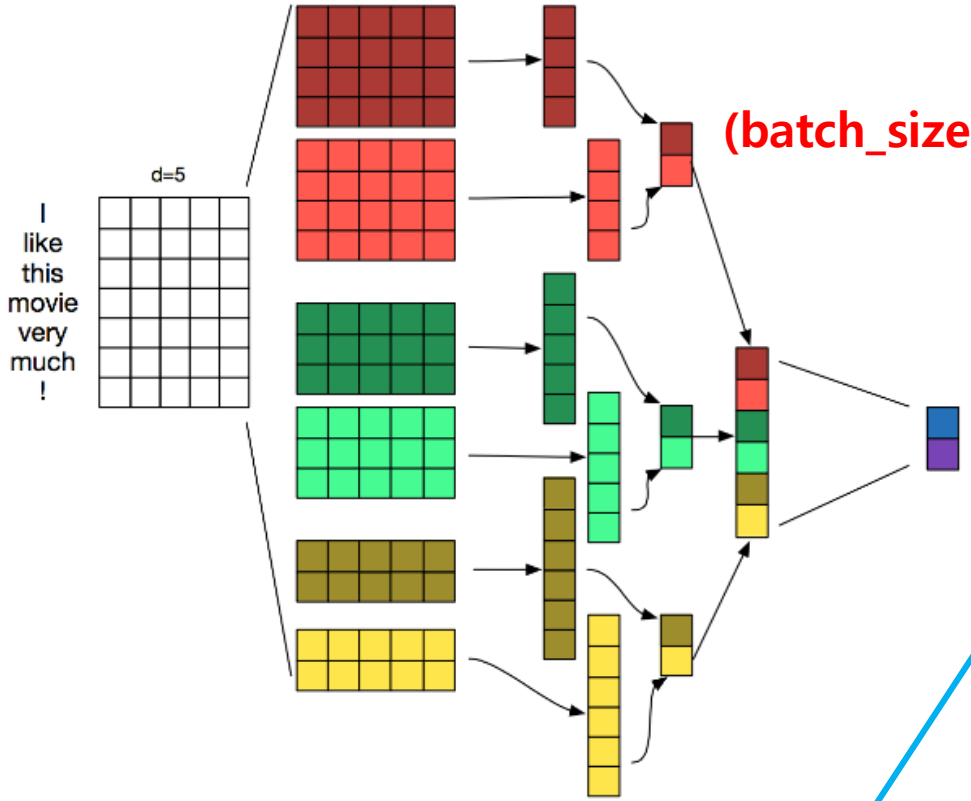
그림의 경우
num_filters=2
seq_length = 7
filter_size = 4
7 – 4 + 1 = 4



(batch_size, seq_length – filter_size + 1, 1, num_filters)

(batch_size, 4, 1, 2)

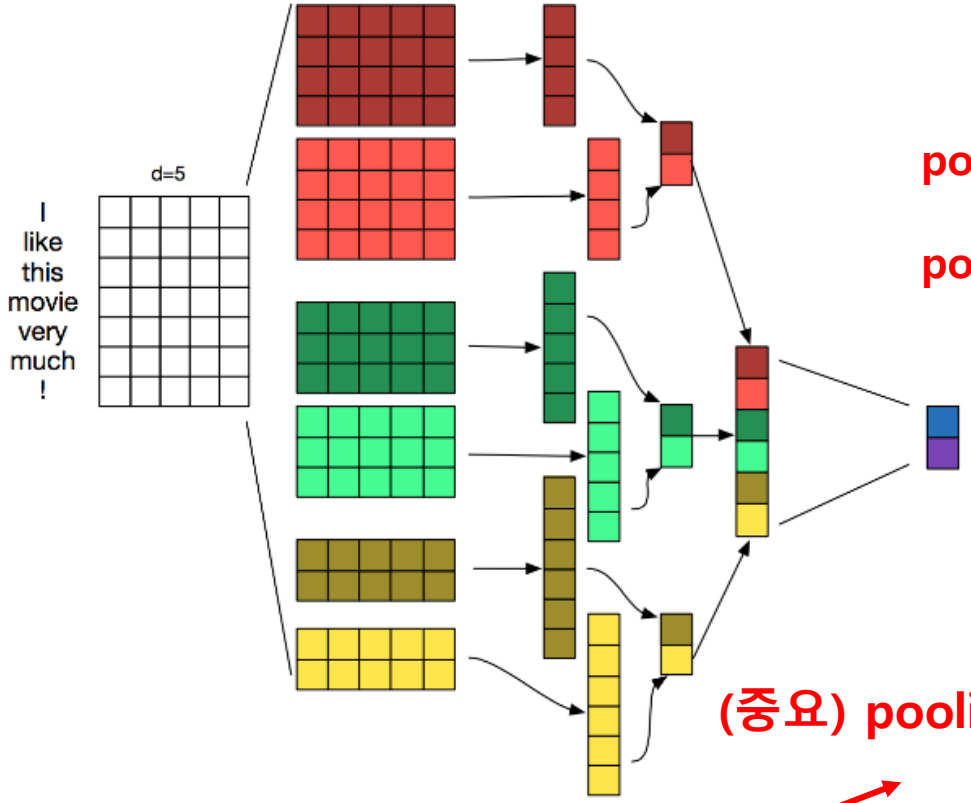
```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```



(batch_size, seq_length – filter_size + 1, 1, 1)

(batch_size, 4, 1, 2)

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

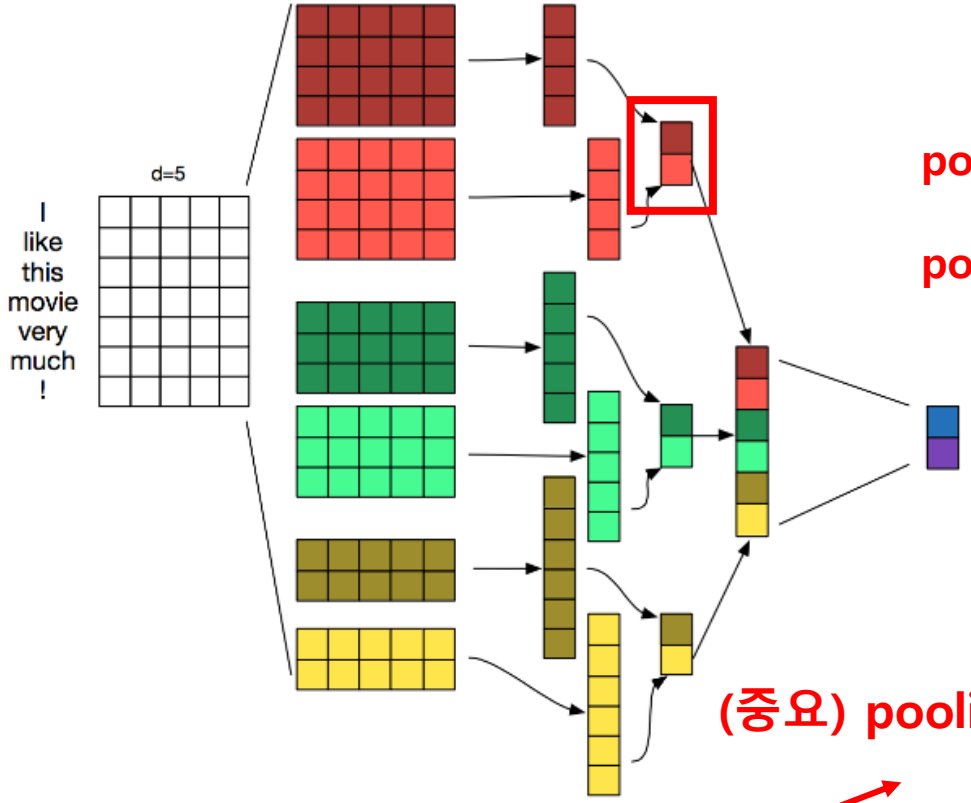


pooling 전: (batch_size, 4, 1, 2)

pooling 후: (batch_size, ?, ?, ?)

(중요) pooliing 연산 후 activation map의 shape은 어떻게 될까?

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

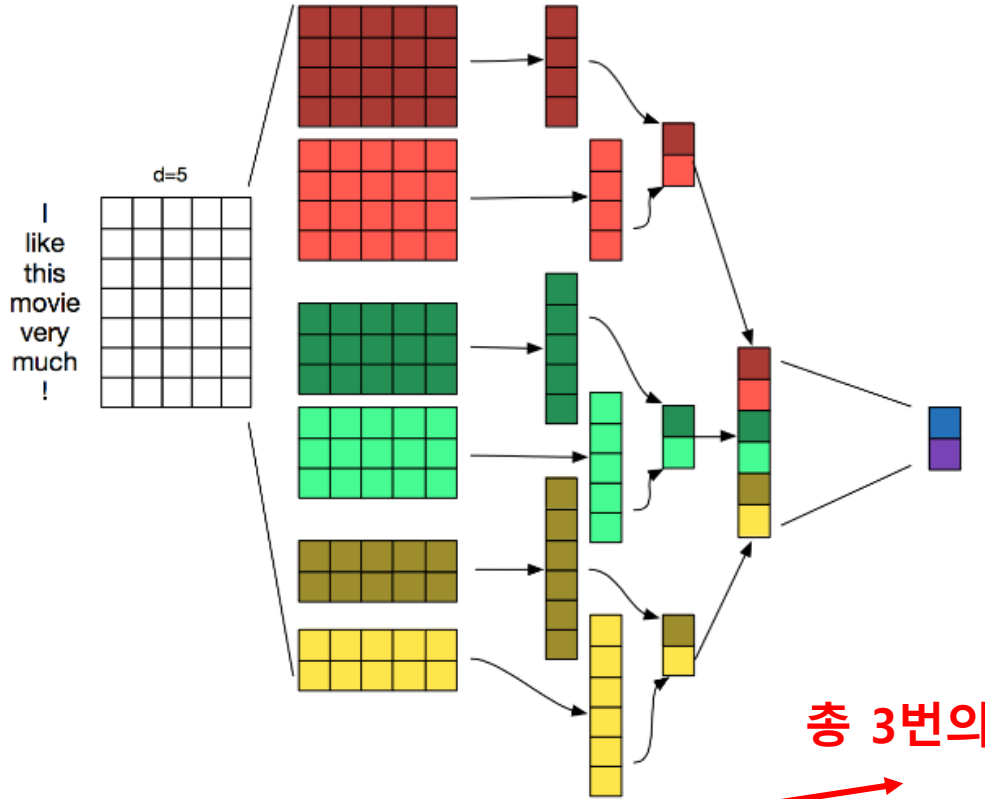


pooling 전: (batch_size, 4, 1, 2)

pooling 후: (batch_size, 1, 1, 2)

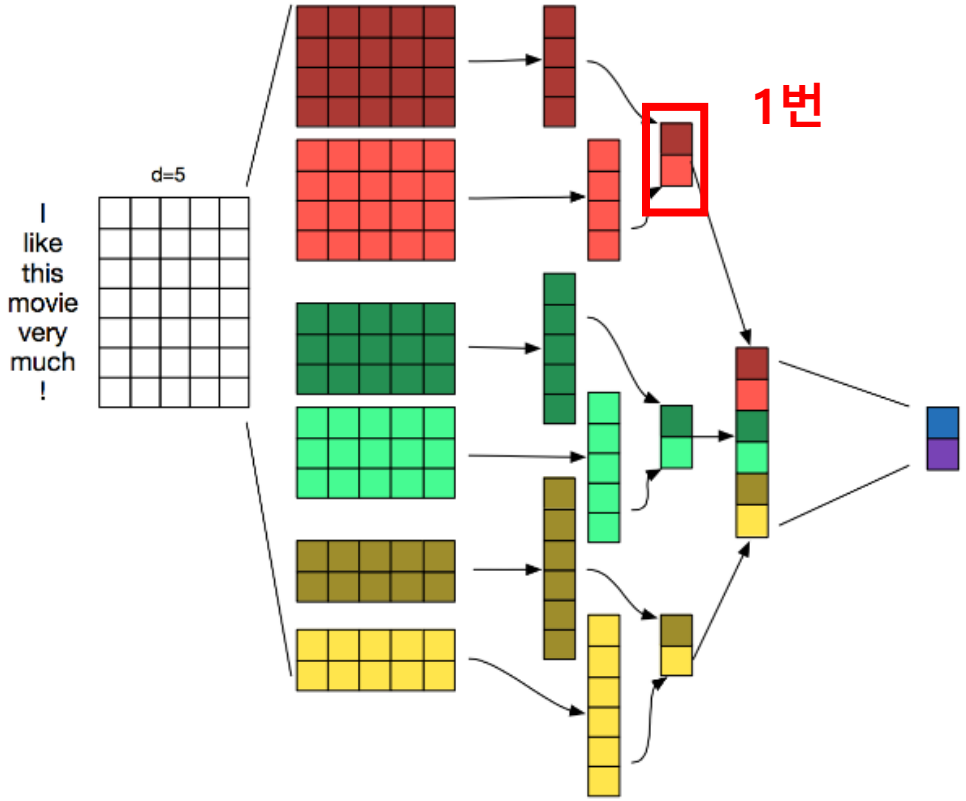
(중요) pooliing 연산 후 activation map의 shape은 어떻게 될까?

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```



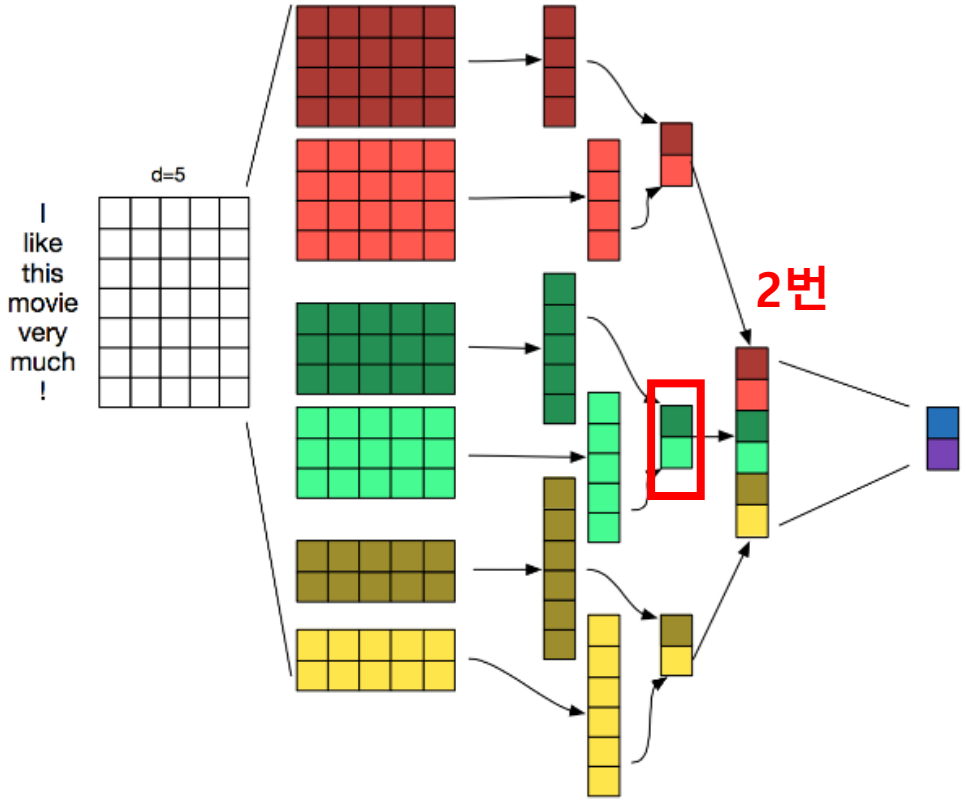
총 3번의 iteration을 돈다!

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```



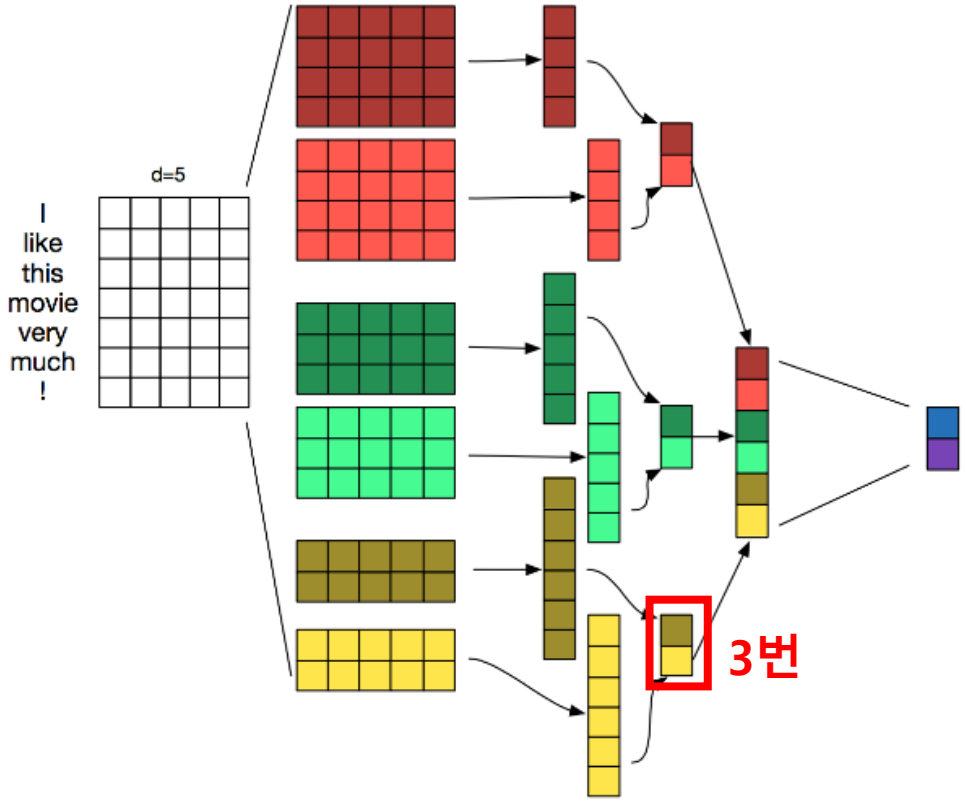

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

첫 번째 결과 저장



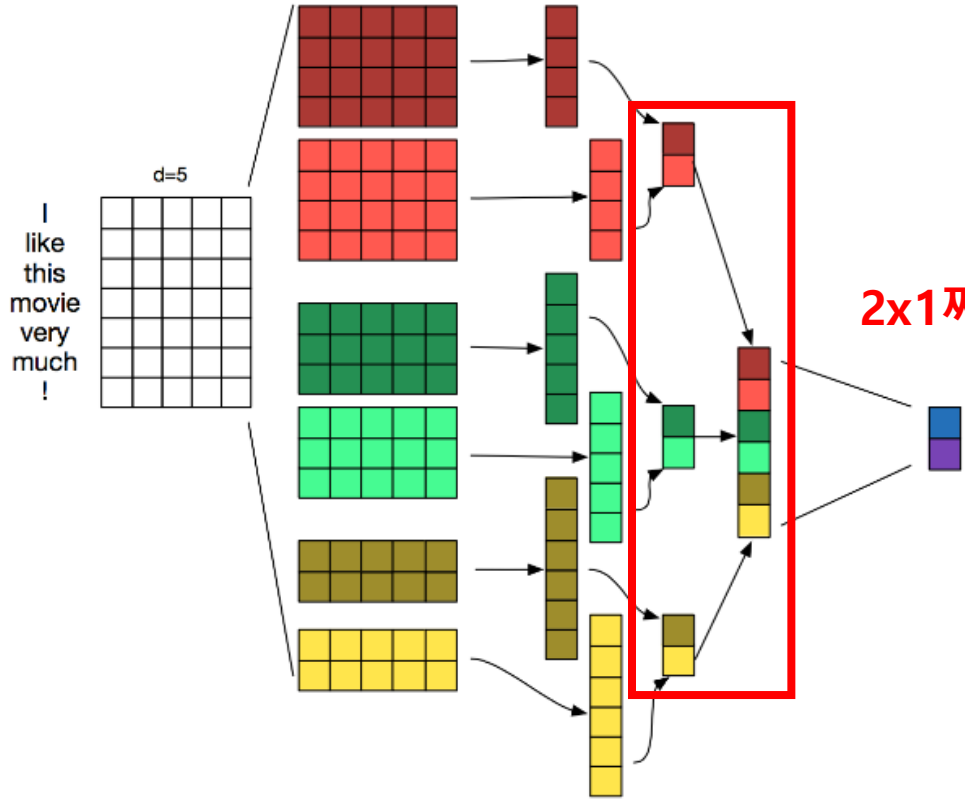

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

두 번째 결과 저장



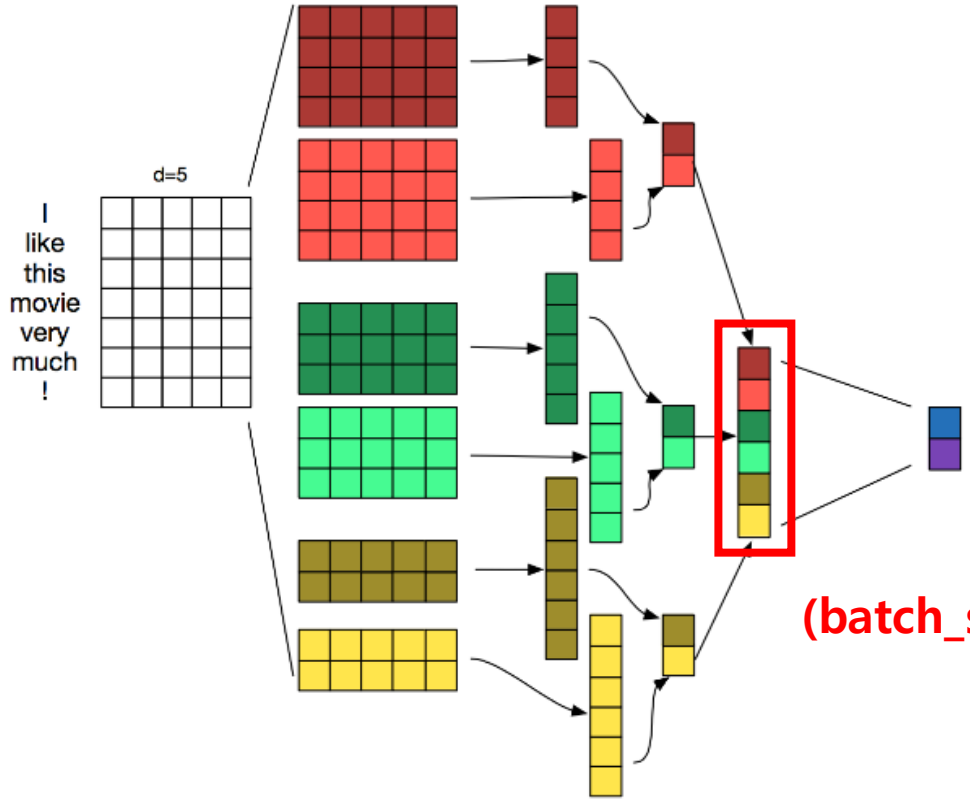

```
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)
```

세 번째 결과 저장



concatenate
2x1짜리 vector 3개를 이어준다

```
pooled_outputs = []
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)

pooled = tf.concat(3, pooled_outputs)
pooled = tf.reshape(pooled, [batch_size, -1])
```

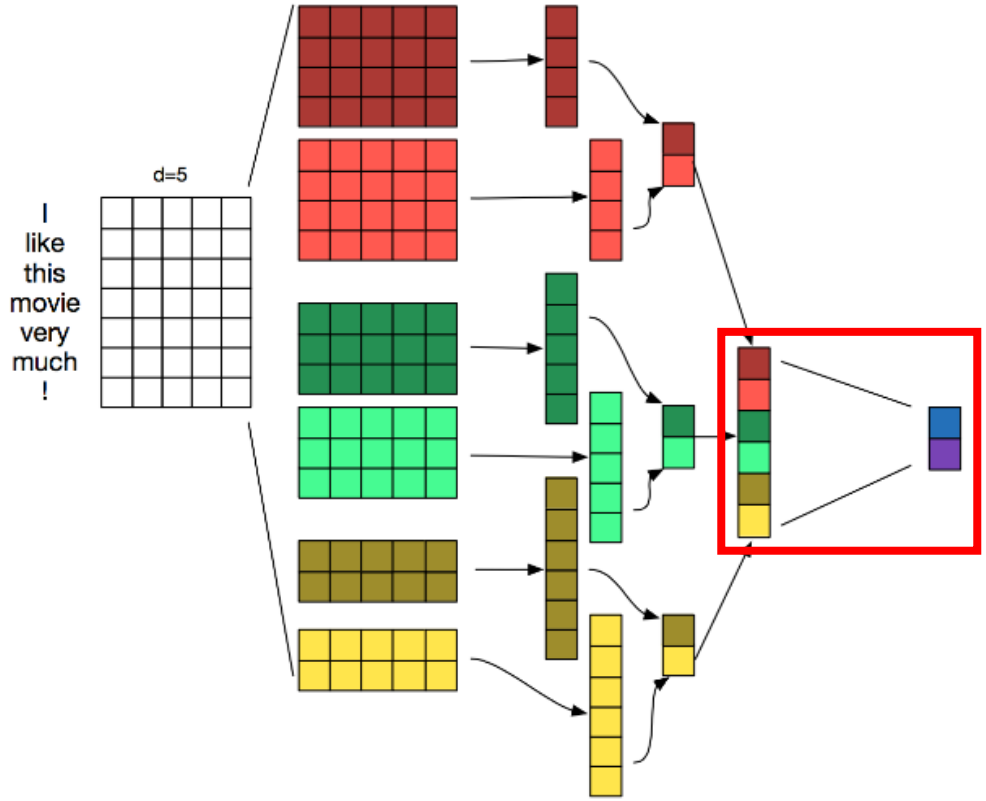


(batch_size, num_total_filter)
6

```
pooled_outputs = []
for i, filter_size in enumerate(filter_sizes):
    with tf.variable_scope('conv_maxpool_%d' %(i+1)):
        filter_shape = [filter_size, dim_emb, 1, num_filters]
        w = tf.get_variable('w', shape=filter_shape, initializer=tf.contrib.layers.xavier_initializer())
        b = tf.get_variable('b', shape=[num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(x_embed, w, strides=[1, 1, 1, 1], padding='VALID') + b
        relu = tf.nn.relu(conv)
        pooled = tf.nn.max_pool(relu, [1, seq_length - filter_size + 1, 1, 1], [1, 1, 1, 1], padding='VALID')
        pooled_outputs.append(pooled)

pooled = tf.concat(3, pooled_outputs)
pooled = tf.reshape(pooled, [batch_size, -1])
```

CNN with Text Classification

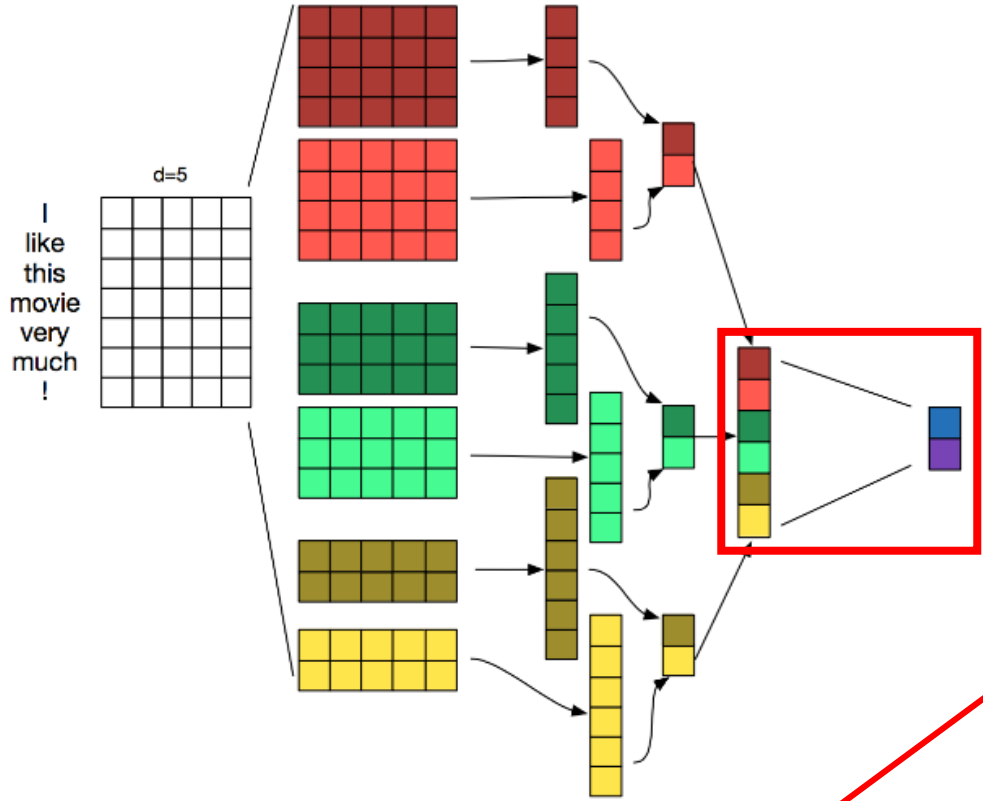
Output Layer



(batch_size, 6) -> (batch_size, 2)

6x2 짜리 행렬 필요

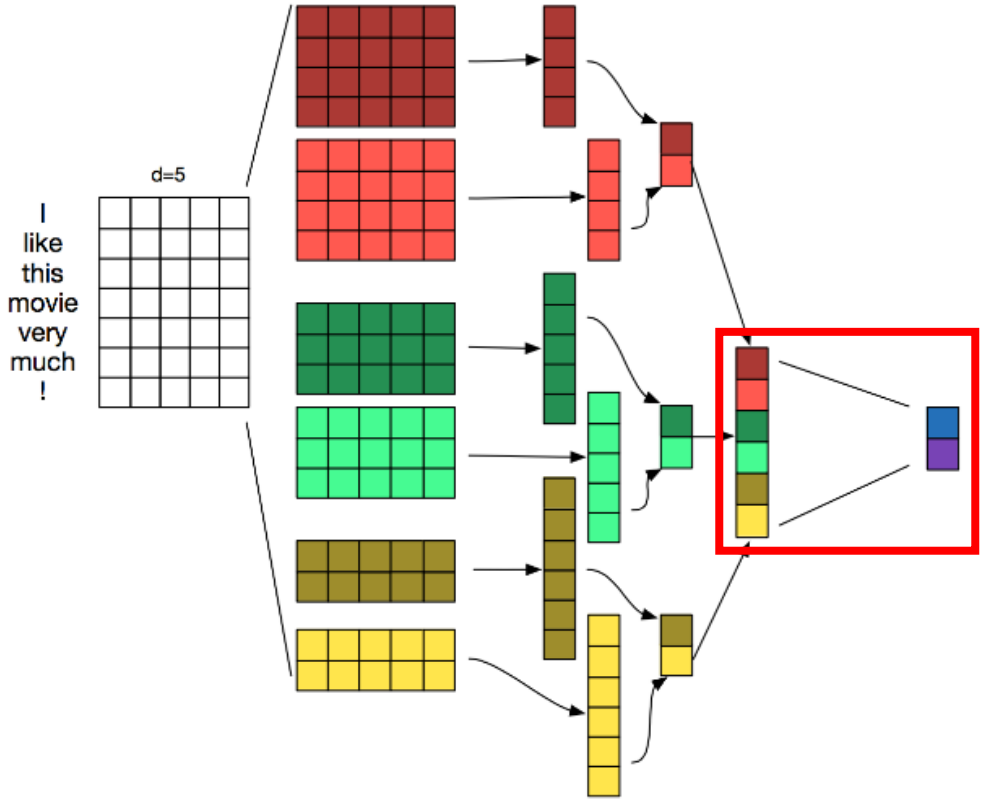
```
with tf.variable_scope('output_layer'):
    num_total_filter = len(filter_sizes) * num_filters
    w = tf.get_variable('w', shape=[num_total_filter, num_class], initializer=tf.contrib.layers.xavier_initializer())
    b = tf.get_variable('b', shape=[num_class], initializer=tf.constant_initializer(0.0))

    out = tf.matmul(pooled, w) + b    # (batch_size, num_class)
```



(batch_size, 6) -> (batch_size, 2)

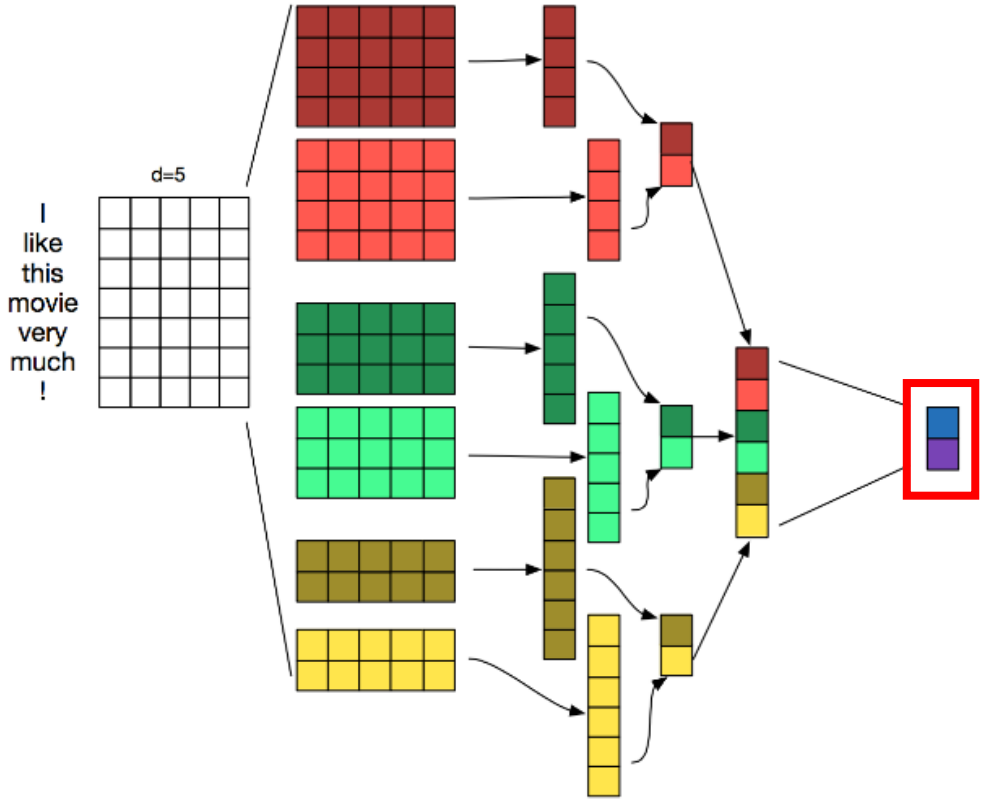
6x2 짜리 행렬 필요

```
with tf.variable_scope('output_layer'):
    num_total_filter = len(filter_sizes) * num_filters
    w = tf.get_variable('w', shape=[num_total_filter, num_class], initializer=tf.contrib.layers.xavier_initializer())
    b = tf.get_variable('b', shape=[num_class], initializer=tf.constant_initializer(0.0))

    out = tf.matmul(pooled, w) + b    # (batch_size, num_class)
```




```
with tf.variable_scope('output_layer'):
    num_total_filter = len(filter_sizes) * num_filters
    w = tf.get_variable('w', shape=[num_total_filter, num_class], initializer=tf.contrib.layers.xavier_initializer())
    b = tf.get_variable('b', shape=[num_class], initializer=tf.constant_initializer(0.0))

    out = tf.matmul(pooled, w) + b    # (batch_size, num_class)
```

행렬 곱셈 (fully connected layer)



loss 함수 정의 (spase_softmax (?))

```
with tf.name_scope('optimizer'):
    self.loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(out, self.y))
    self.train_op = tf.train.AdamOptimizer(0.001, beta1=0.5).minimize(self.loss)
```

```
tf.nn.sparse_softmax_cross_entropy_with_logits(out, self.y)
tf.nn.softmax_cross_entropy_with_logits(out, self.y_onehot)
```

```
tf.nn.sparse_softmax_cross_entropy_with_logits(out, self.y)
tf.nn.softmax_cross_entropy_with_logits(out, self.y_onehot)
```

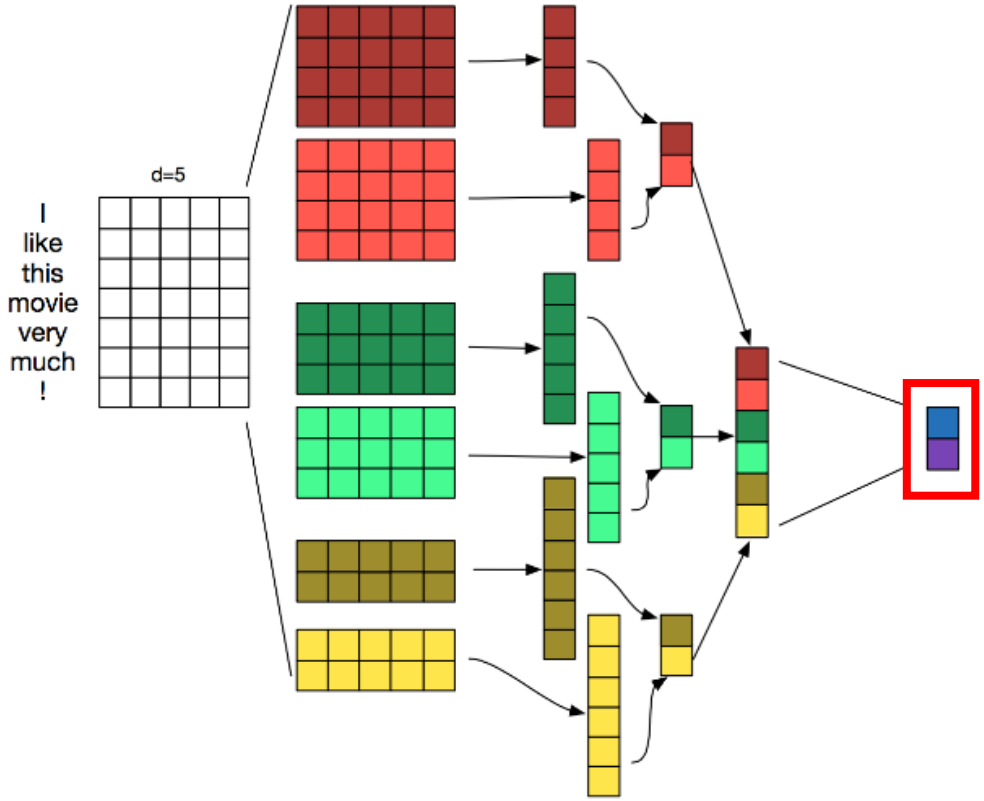
(batch_size, 2) **(batch_size, 2)**

0.3 0.7	0 1
0.4 0.6	0 1
0.9 0.1	1 0

```
tf.nn.sparse_softmax_cross_entropy_with_logits(out, self.y)
tf.nn.softmax_cross_entropy_with_logits(out, self.y_onehot)
```

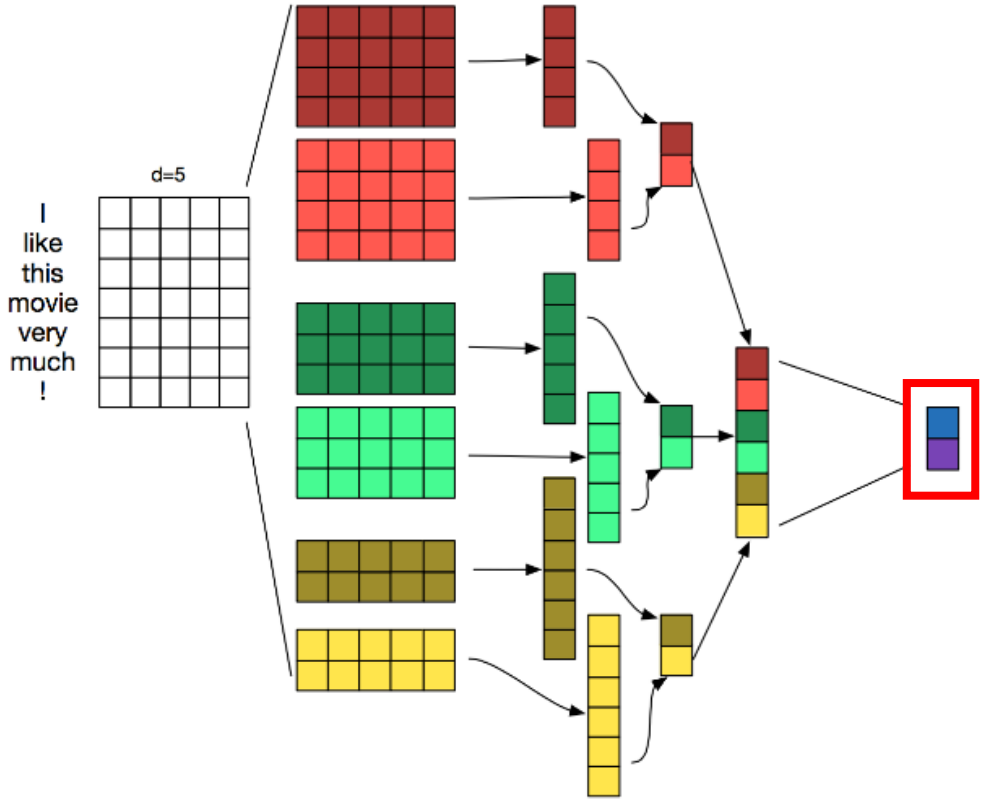
(batch_size, 2) **(batch_size,)**

0.3 0.7	1
0.4 0.6	1
0.9 0.1	0



loss 함수 정의

```
with tf.name_scope('optimizer'):
    self.loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(out, self.y))
    self.train_op = tf.train.AdamOptimizer(0.001, beta1=0.5).minimize(self.loss)
```

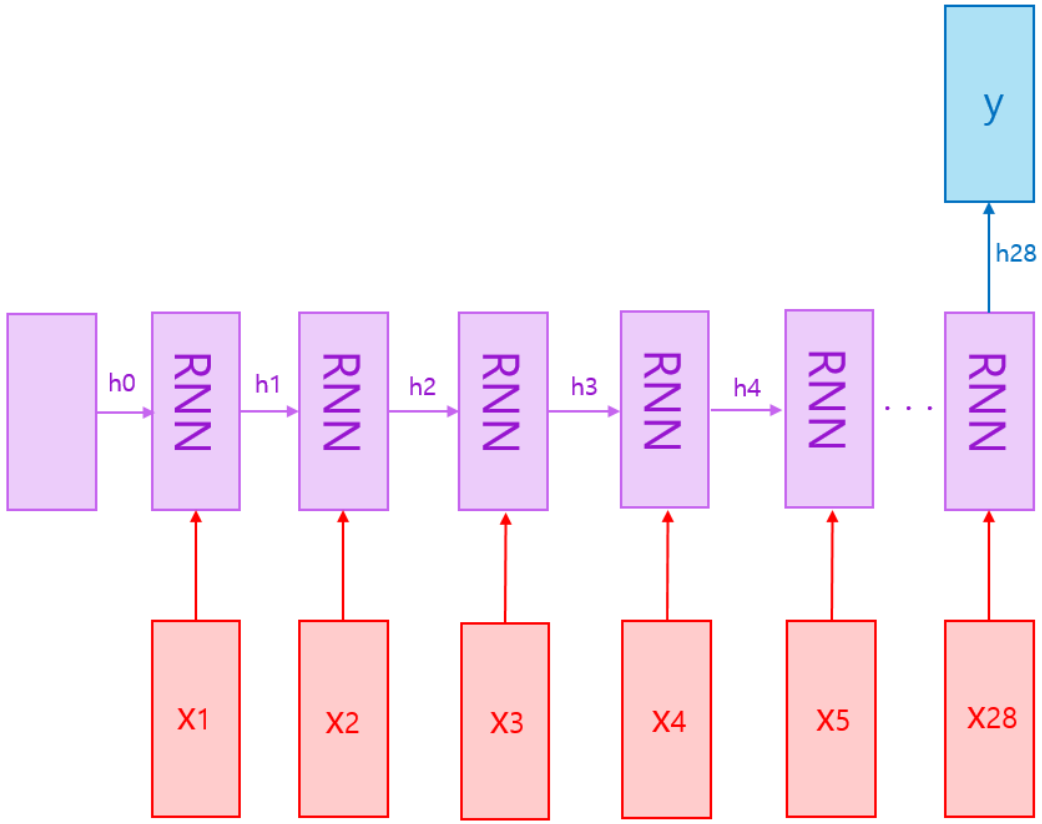


training optimizer 정의

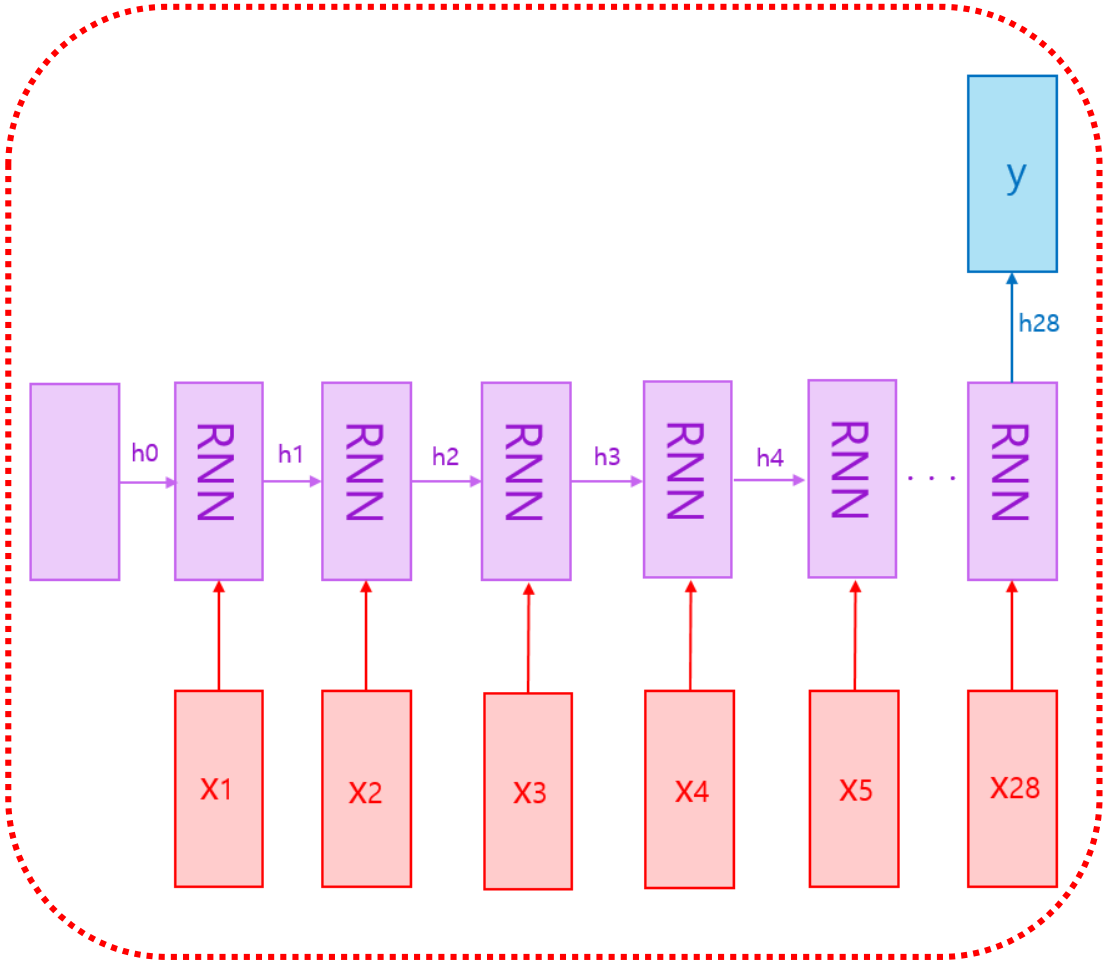
```
with tf.name_scope('optimizer'):
    self.loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(out, self.y))
    self.train_op = tf.train.AdamOptimizer(0.001, beta1=0.5).minimize(self.loss)
```

04 Recurrent Neural Network

Recurrent Neural Network

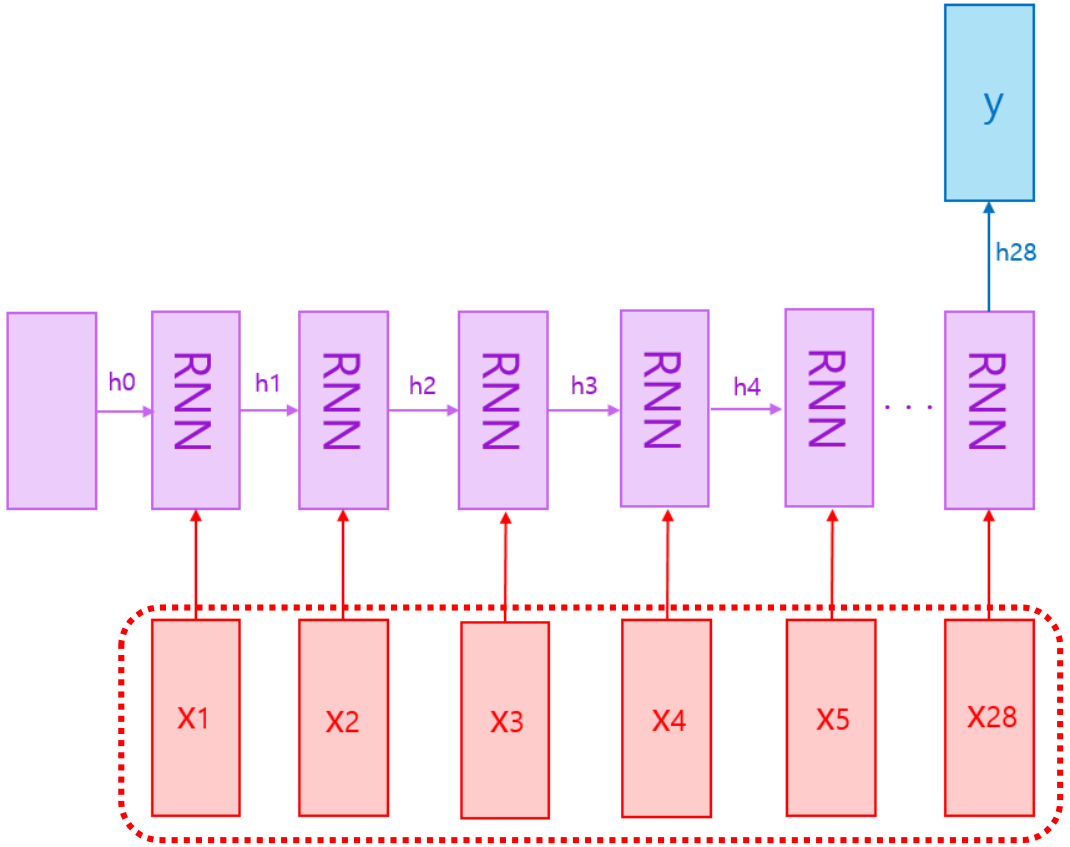


Recurrent Neural Network



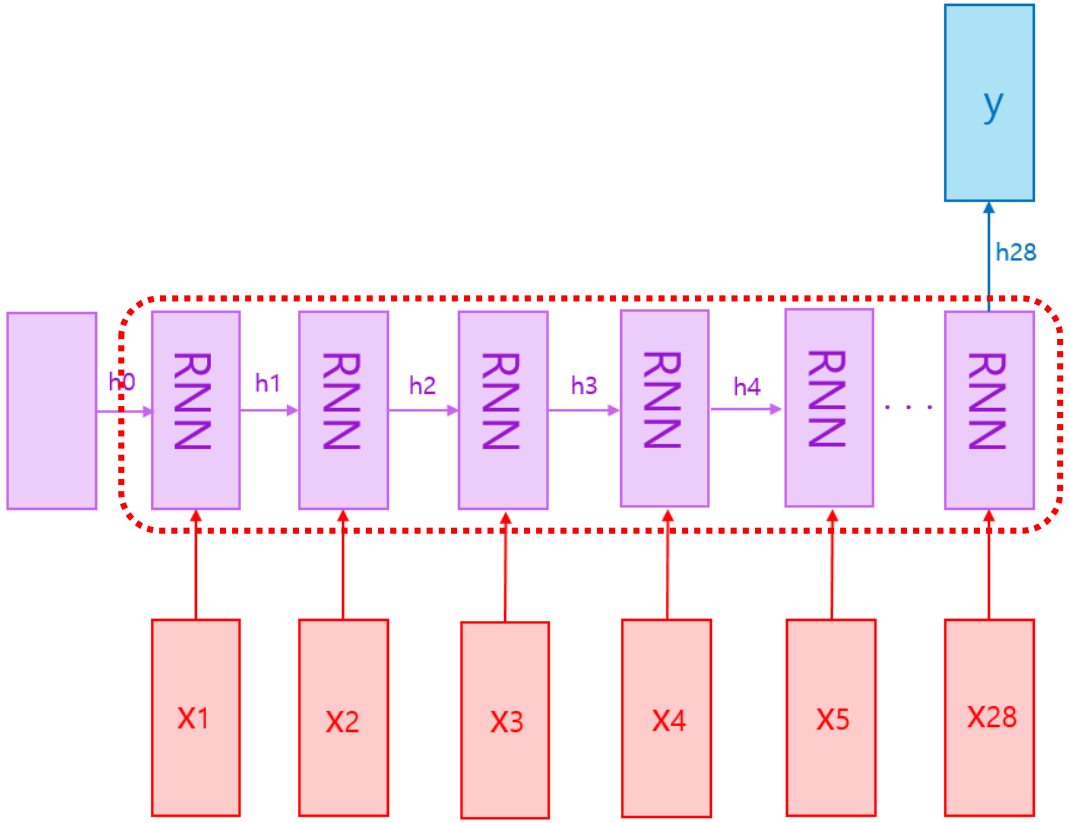
Recurrent Neural Network (many to one)

Recurrent Neural Network



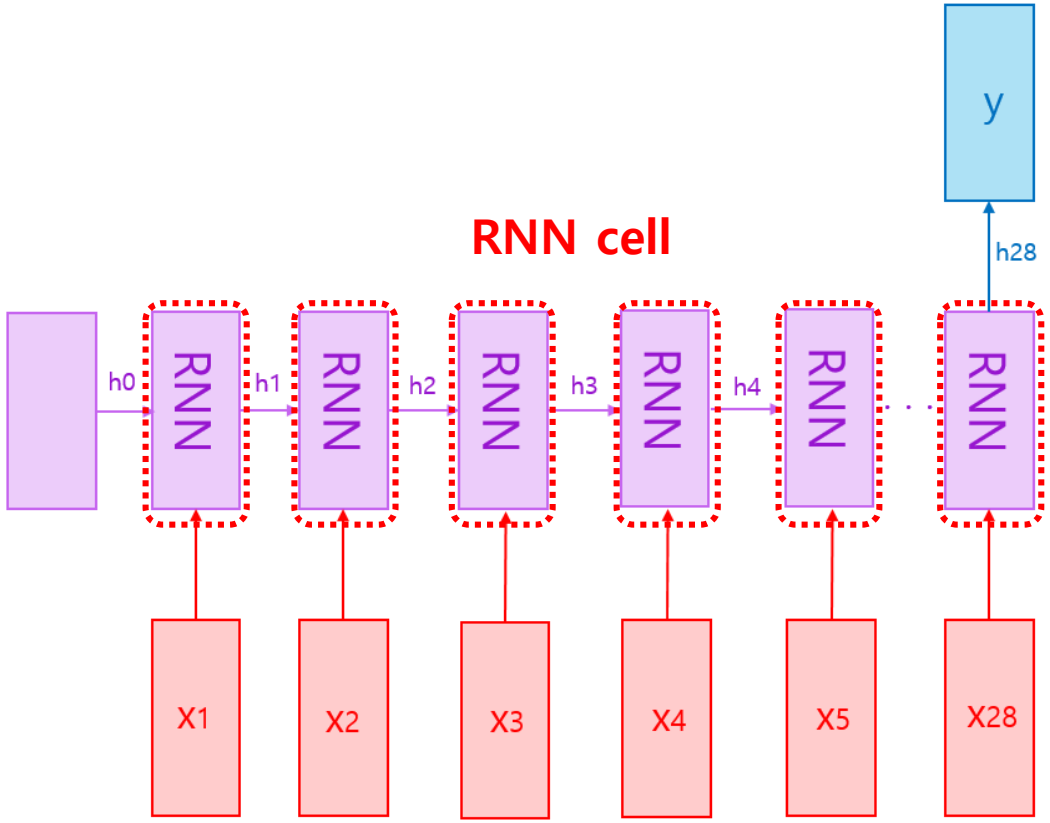
sequence of input vectors (many)

Recurrent Neural Network

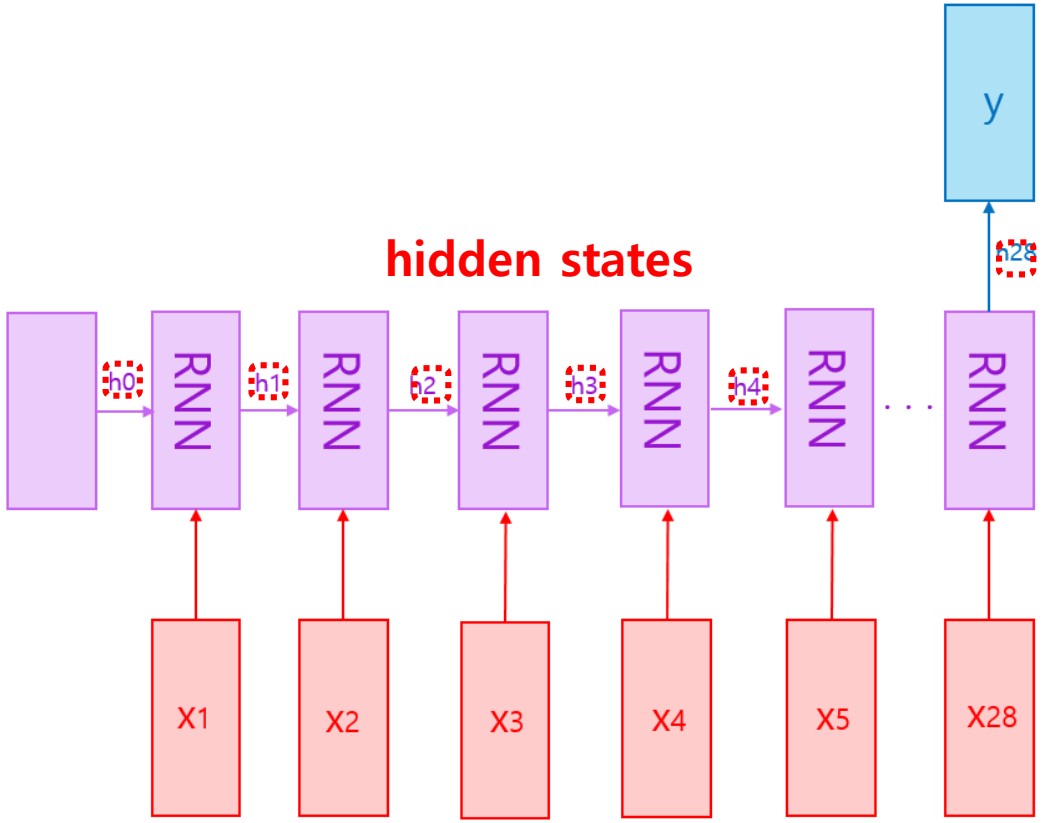


RNN cell and hidden states

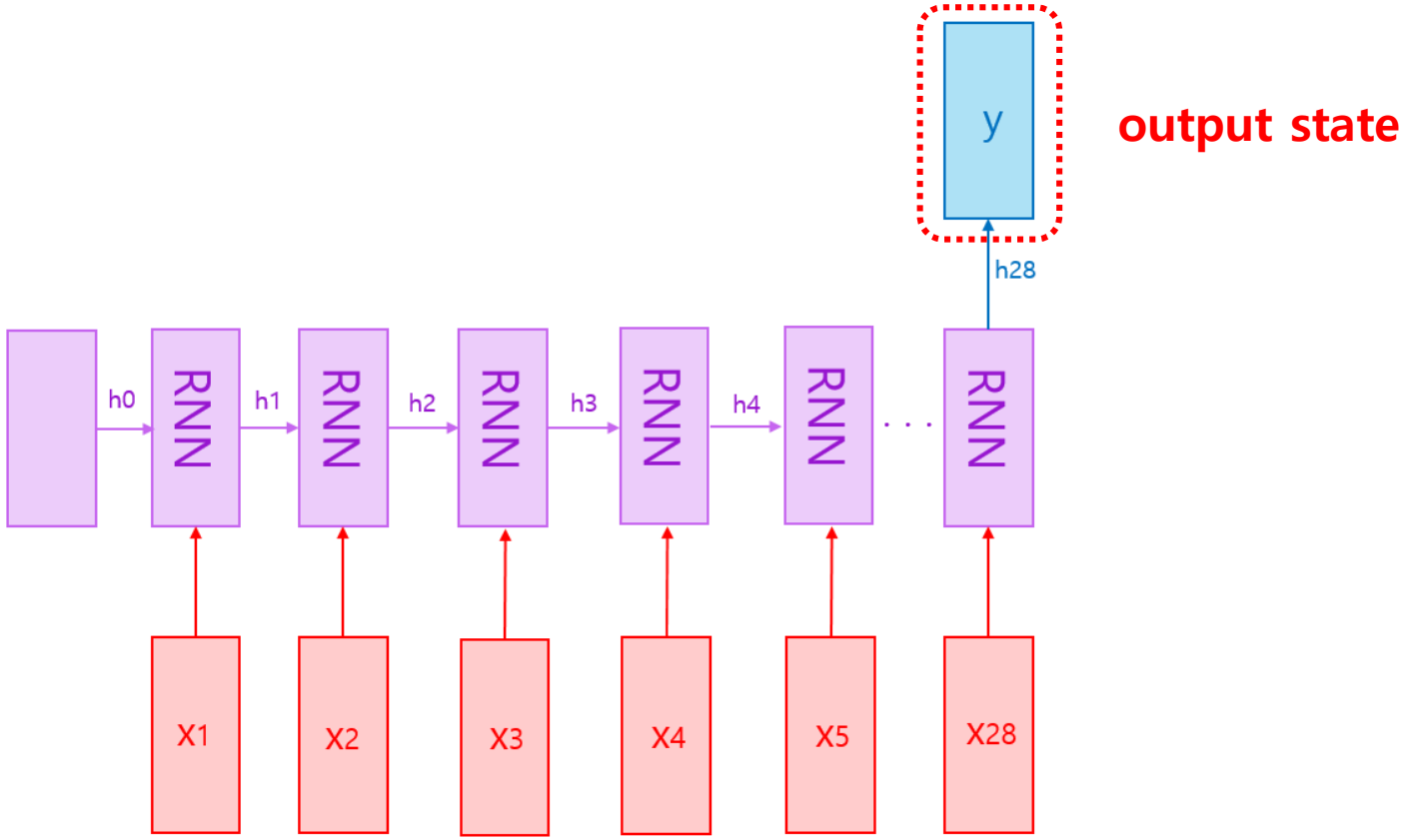
Recurrent Neural Network



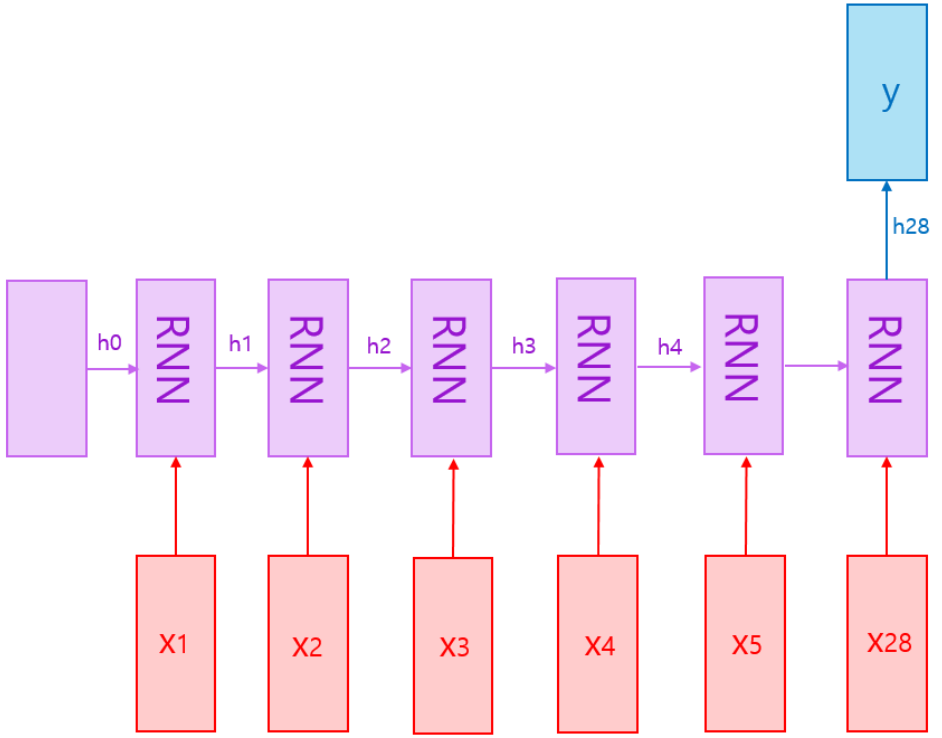
Recurrent Neural Network



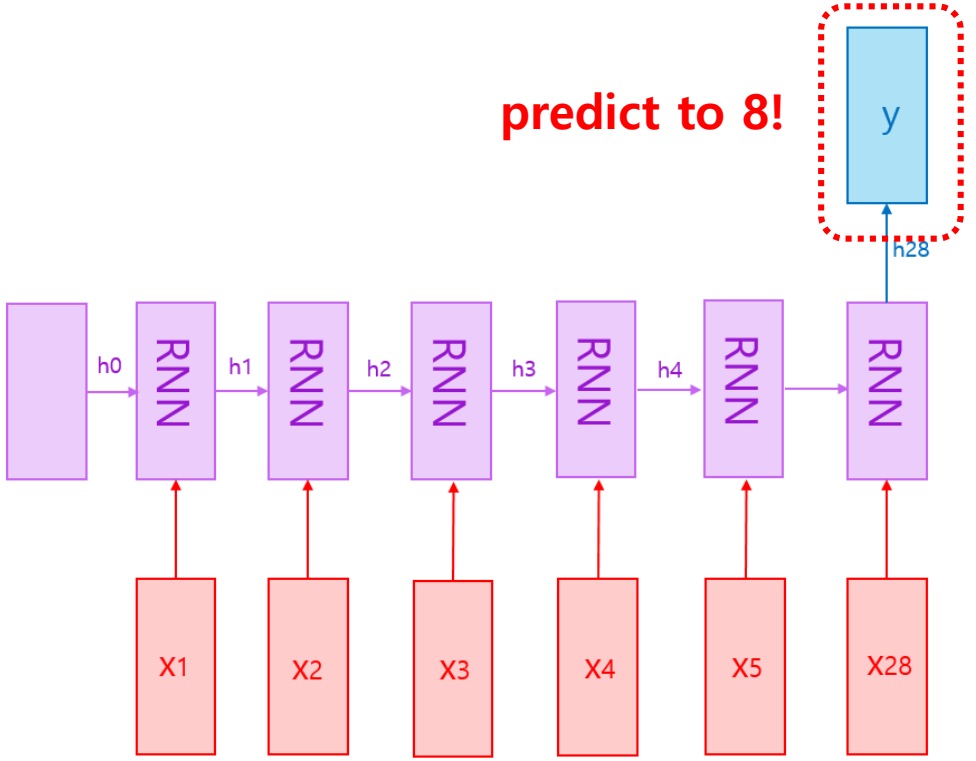
Recurrent Neural Network



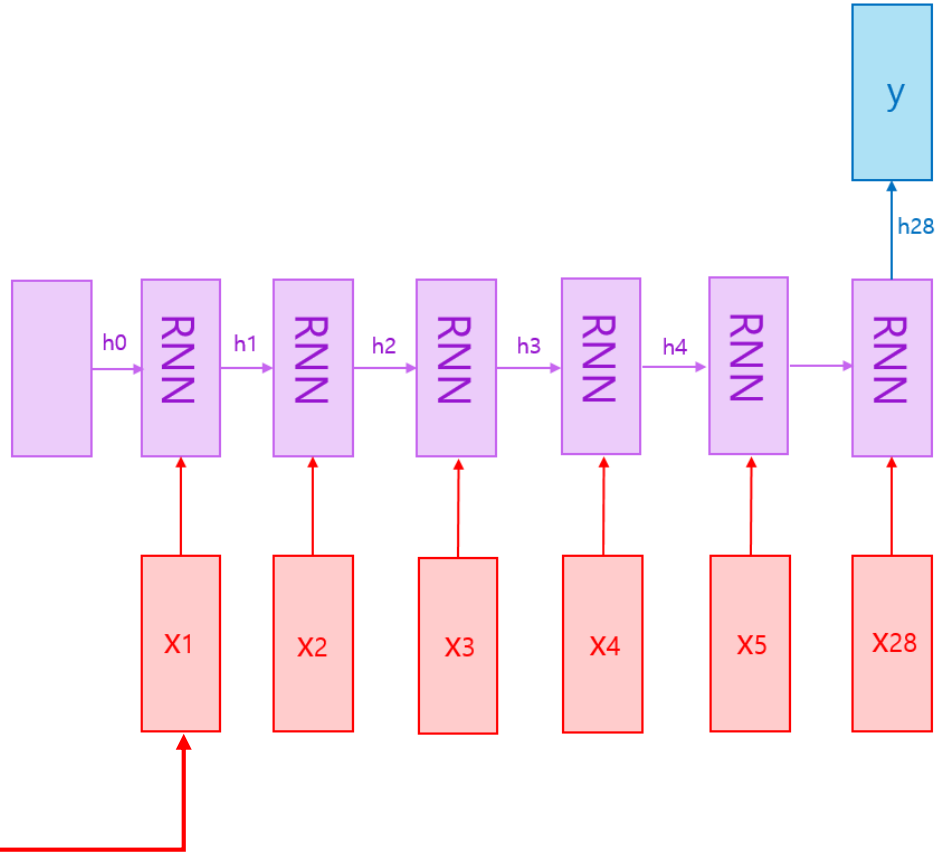
Recurrent Neural Network



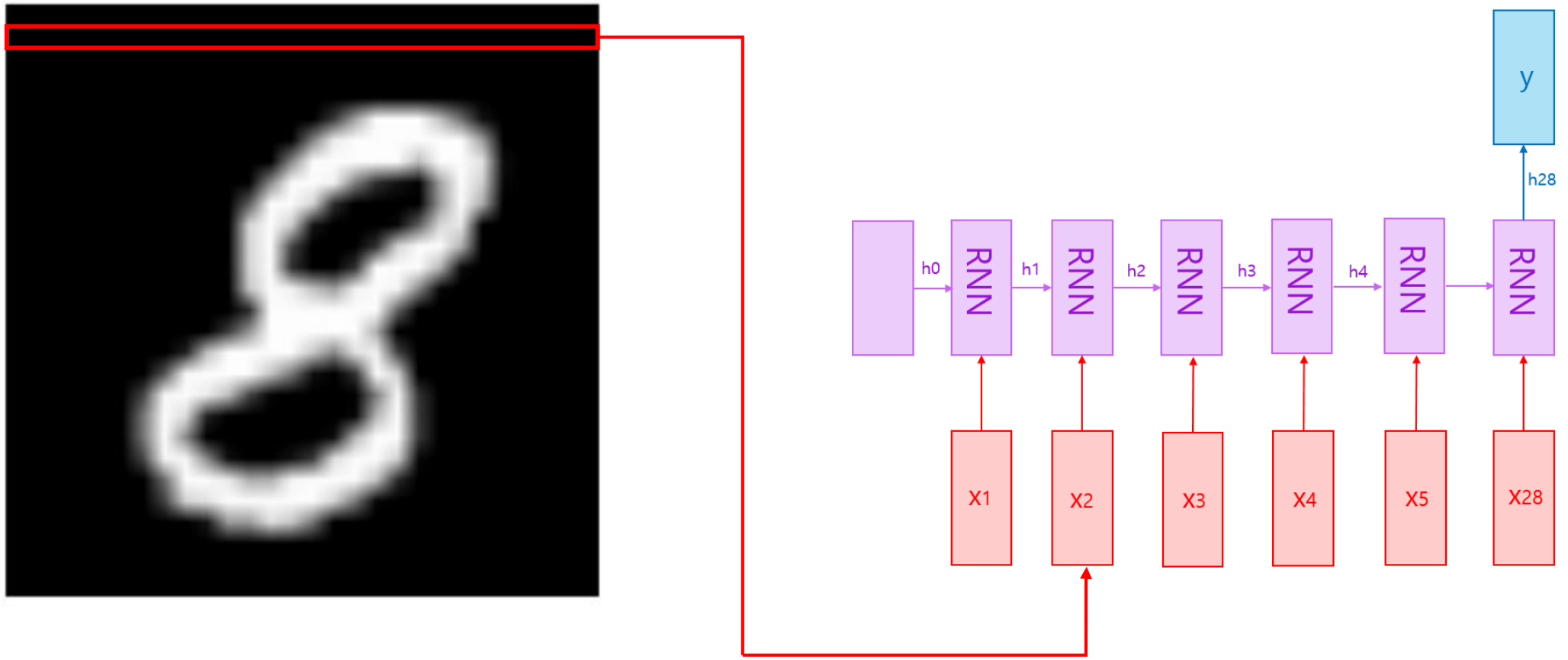
Recurrent Neural Network



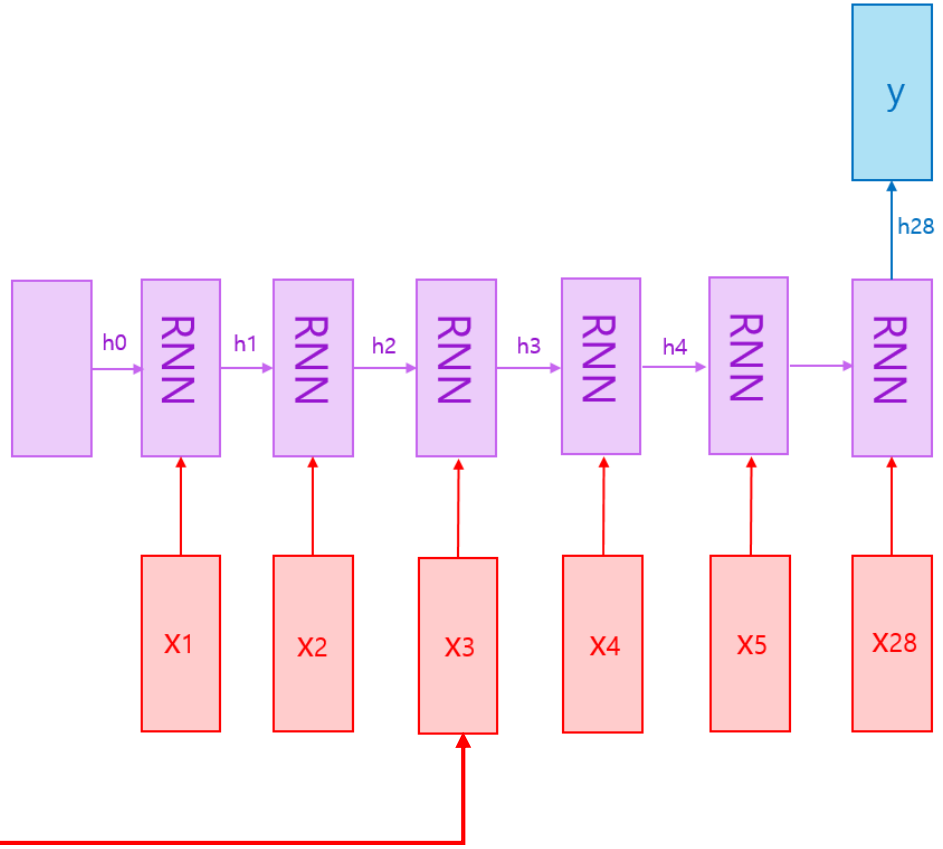
Recurrent Neural Network



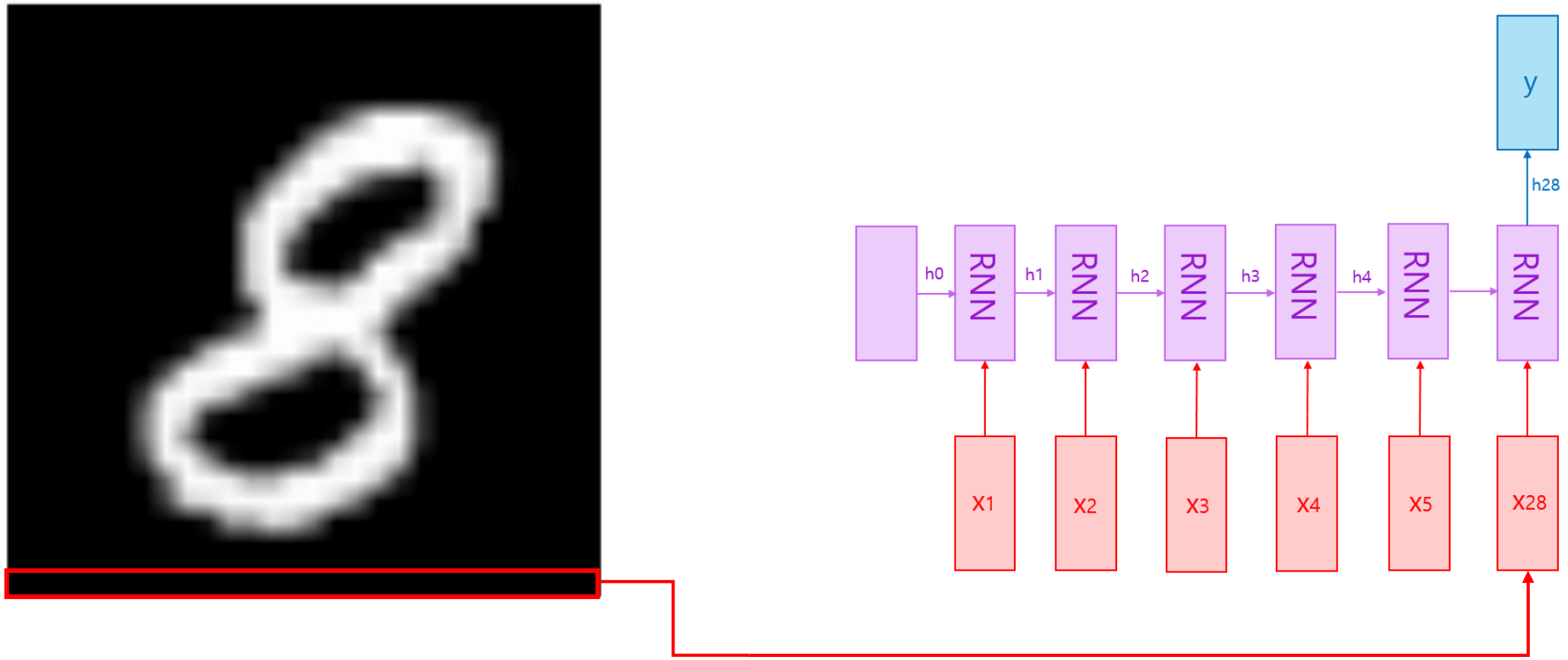
Recurrent Neural Network



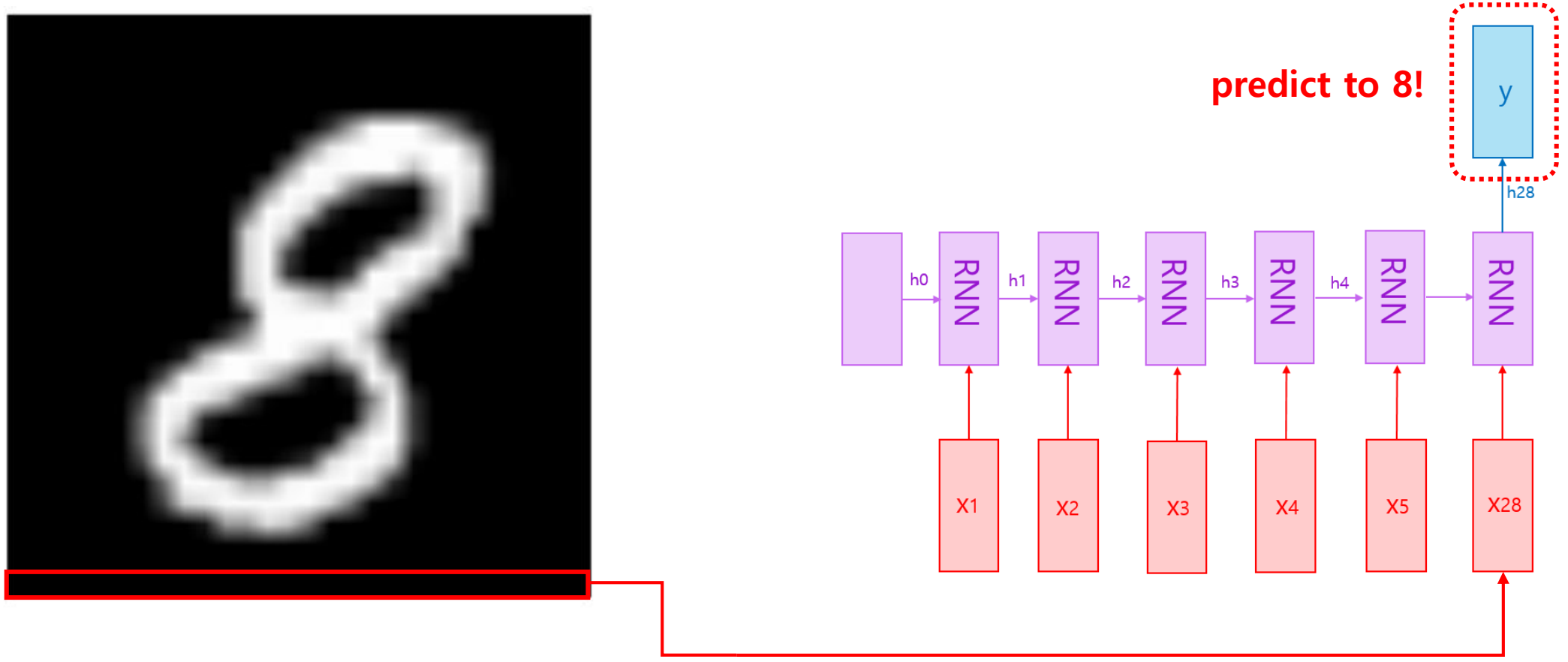
Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network

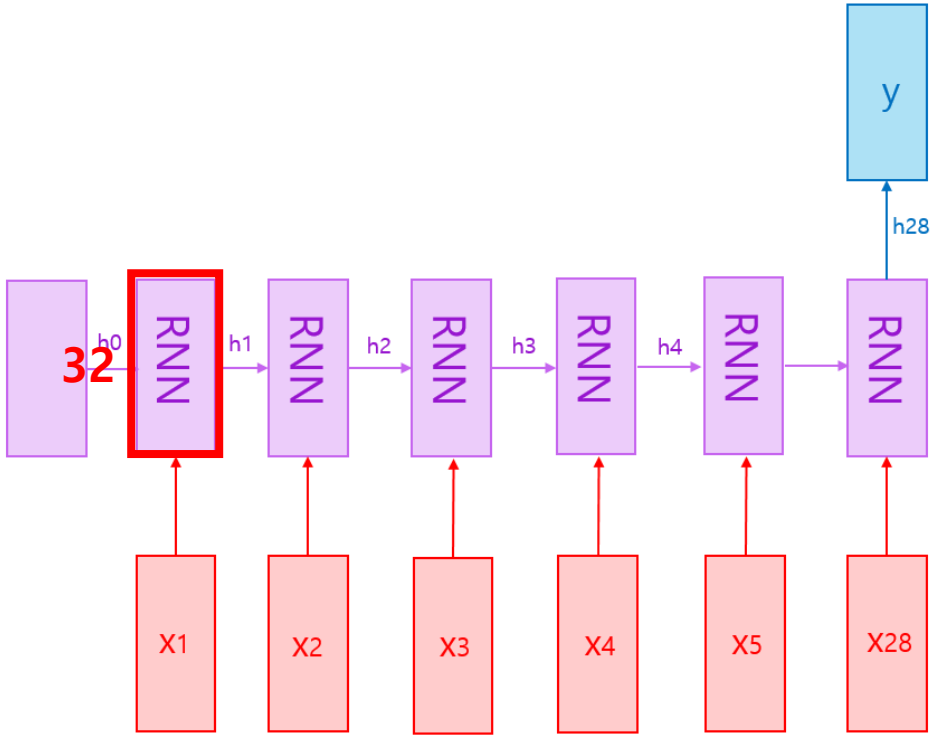


Recurrent Neural Network

각 rnn cell의 dimension크기 지정

```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell,
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```

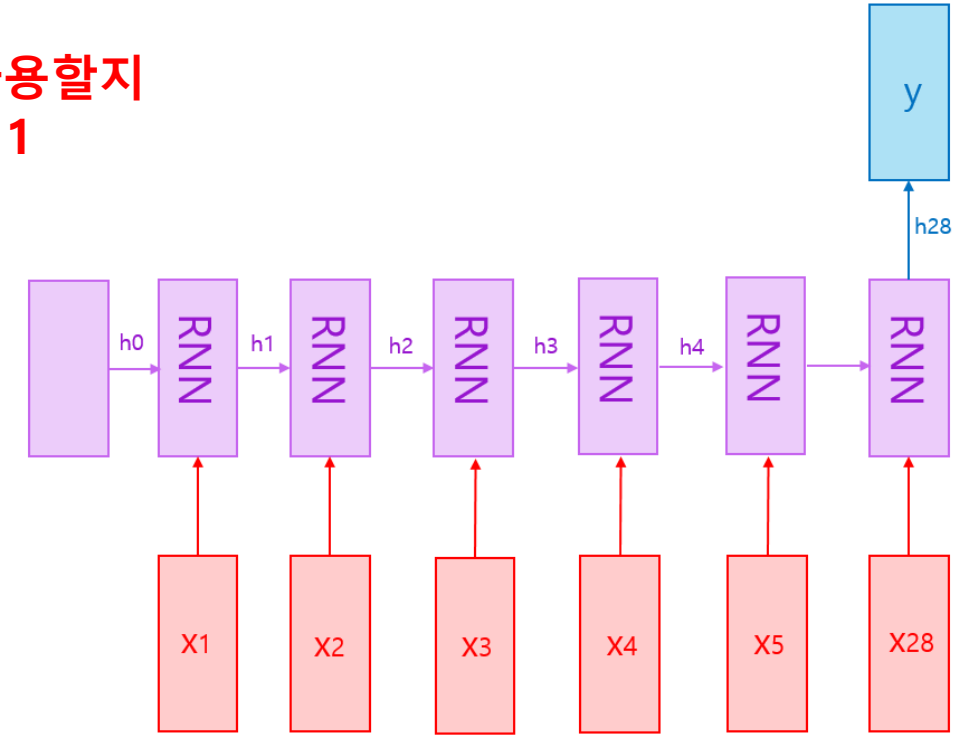
32



Recurrent Neural Network

위아래로 RNN Cell을 몇 개 사용할지
지금 같은 경우 num_layers = 1

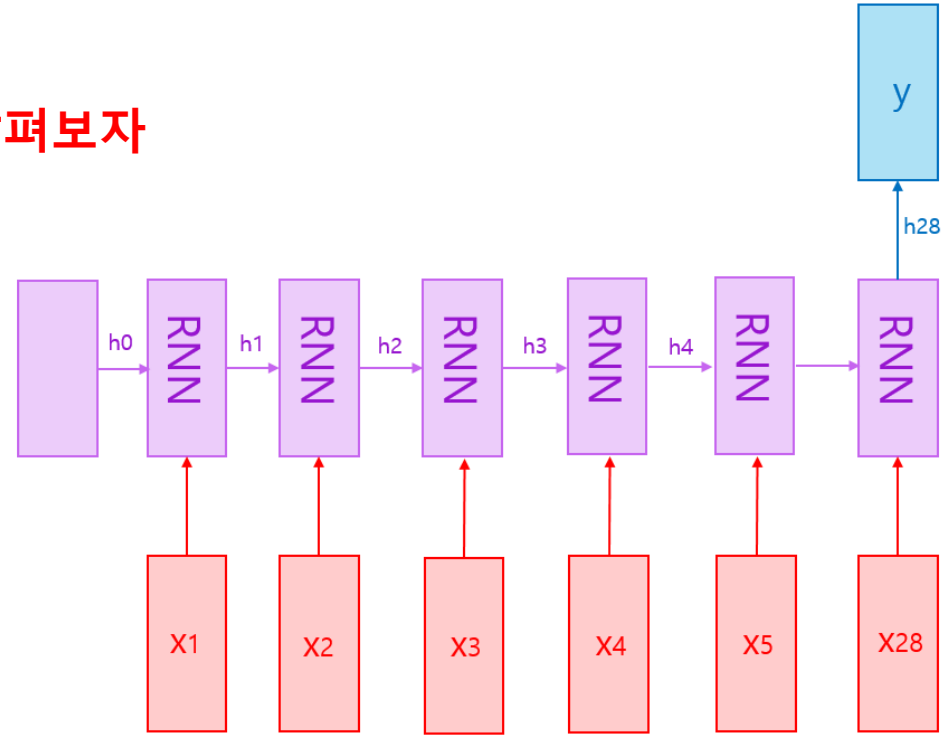
```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell,
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```



Recurrent Neural Network

```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell,
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```

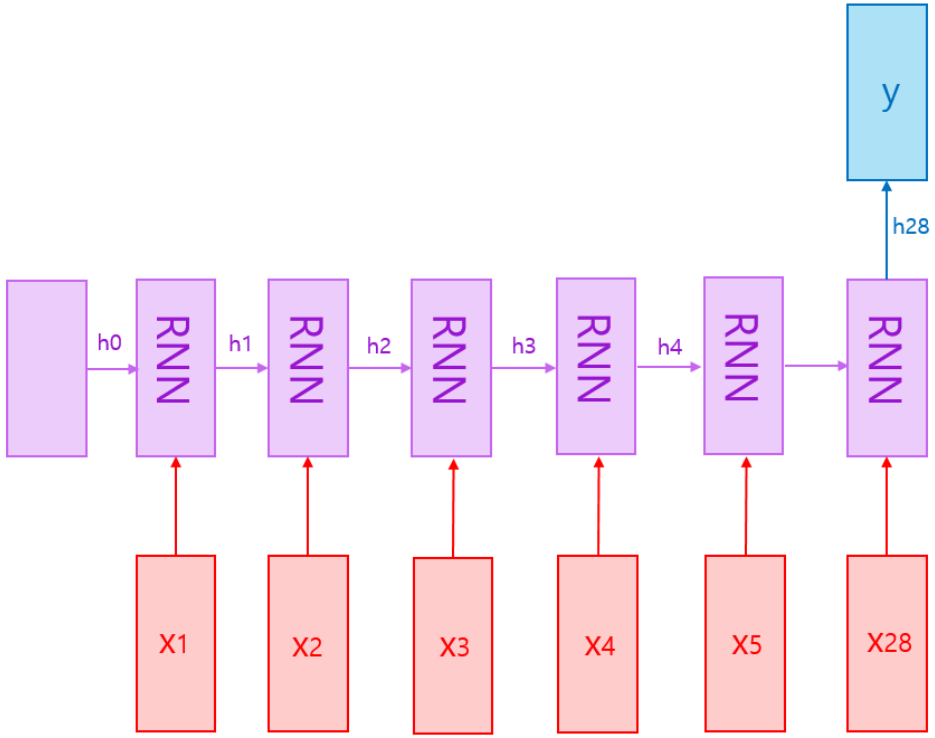
가장 핵심인 dynamic_rnn을 살펴보자



Recurrent Neural Network

위에서 정의한 lstm cell

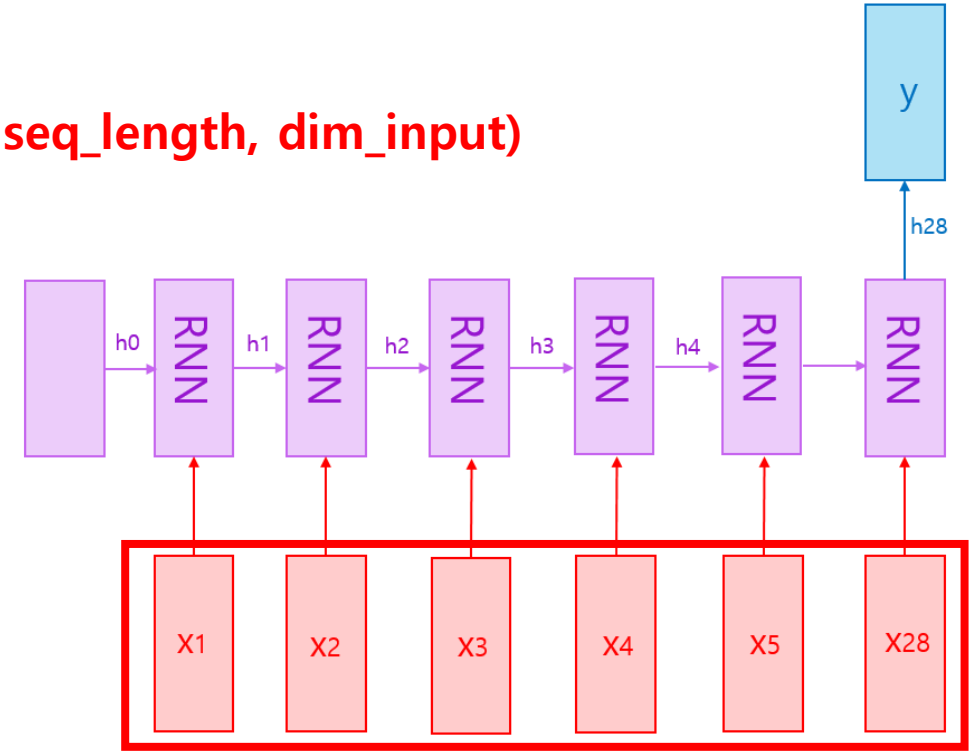
```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```



Recurrent Neural Network

```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell,
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```

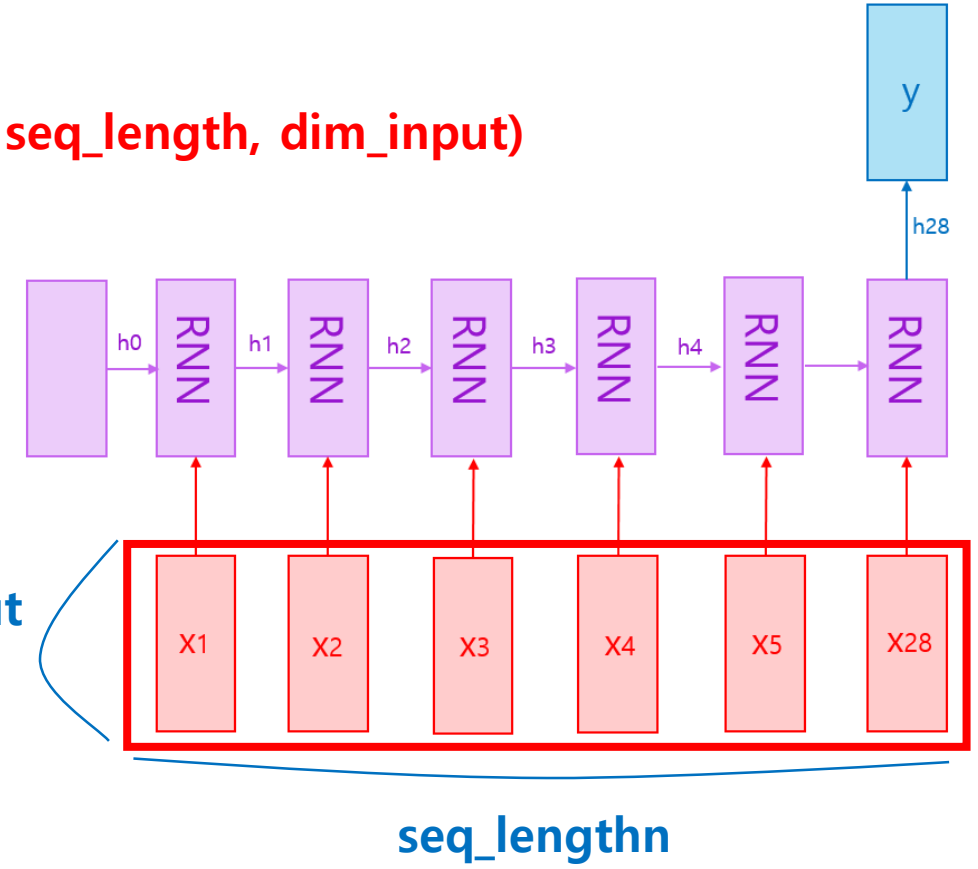
Input sequenc들 (batch_size, seq_length, dim_input)



Recurrent Neural Network

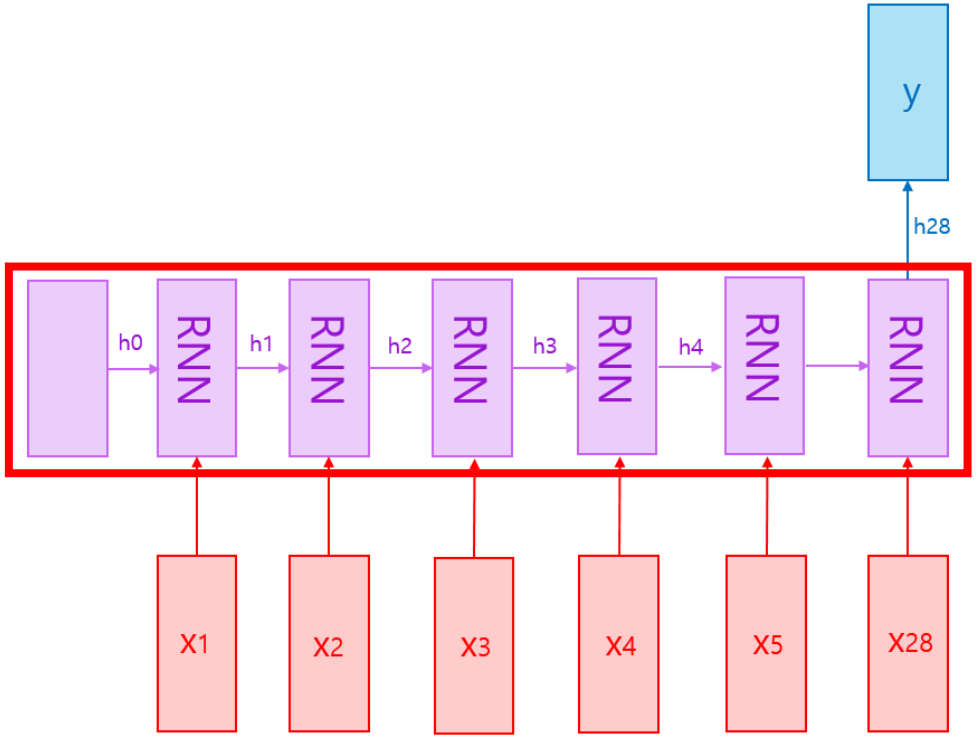
```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell,
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```

Input sequenc들 (batch_size, seq_length, dim_input)



Recurrent Neural Network

```
def recurrent_network(x, mode='train'):
    with tf.variable_scope('lstm') as scope:
        lstm_cell = tf.nn.rnn_cell.BasicLSTMCell(num_units=dim_hidden)
        lstm_cell = tf.nn.rnn_cell.MultiRNNCell(cells=[lstm_cell]*num_layers,
                                                state_is_tuple=True)
        outputs, states = tf.nn.dynamic_rnn(cell=lstm_cell,
                                            inputs=x,
                                            dtype=tf.float32, scope=scope)

    with tf.variable_scope('logits'):
        w = tf.get_variable('w', shape=[dim_hidden, dim_out],
                            initializer=tf.random_normal_initializer())
        b = tf.get_variable('b', shape=[dim_out],
                            initializer=tf.constant_initializer(0.0))
        out = tf.matmul(tf.reshape(outputs[:, -1, :], [-1, dim_hidden]), w) + b
        return out
```



Recurrent Neural Network

실습

References

Samsung tensorflow tutorial에서 사용한 모든 코드 및 발표자료

https://github.com/yunjey/samsung-tensorflow

더 배우고 싶다면..

https://github.com/yunjey/davian-tensorflow

저희 연구실에서 진행하고 있는 tensorflow 실습자료이며 꾸준히 업데이트될 예정