CNN(Convolution Neural Network) - 고수준API 사용

- layer 모듈을 사용한 CNN 모듈 설명

In [15]:
```
# 이미지 처리 분야에서 가장 유명한 신경망 모델인 CNN 을 이용
import tensorflow as tf
import time

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)

print(mnist.train.images.shape)
print(mnist.train.labels.shape)

print(mnist.test.images.shape)
print(mnist.test.labels.shape)
```

```
Extracting ./mnist/data/train-images-idx3-ubyte.gz
Extracting ./mnist/data/train-labels-idx1-ubyte.gz
Extracting ./mnist/data/t10k-images-idx3-ubyte.gz
Extracting ./mnist/data/t10k-labels-idx1-ubyte.gz
(55000, 784)
(55000, 10)
(10000, 784)
(10000, 10)
```

01. 신경망 모델 구성

In [6]:
```
#########
# 신경망 모델 구성
######
# 기존 모델에서는 입력 값을 28x28 하나의 차원으로 구성하였으나,
# CNN 모델을 사용하기 위해 2차원 평면과 특성치의 형태를 갖는 구조로 만듭니다.
# None는 입력데이터의 개수, 마지막 차원 1은 특징의 개수. MNIST는 회색조의 이미지로 색상이 한개
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
Y = tf.placeholder(tf.float32, [None, 10])
is_training = tf.placeholder(tf.bool)     # dropout를 수행할지 안할지 결정
```

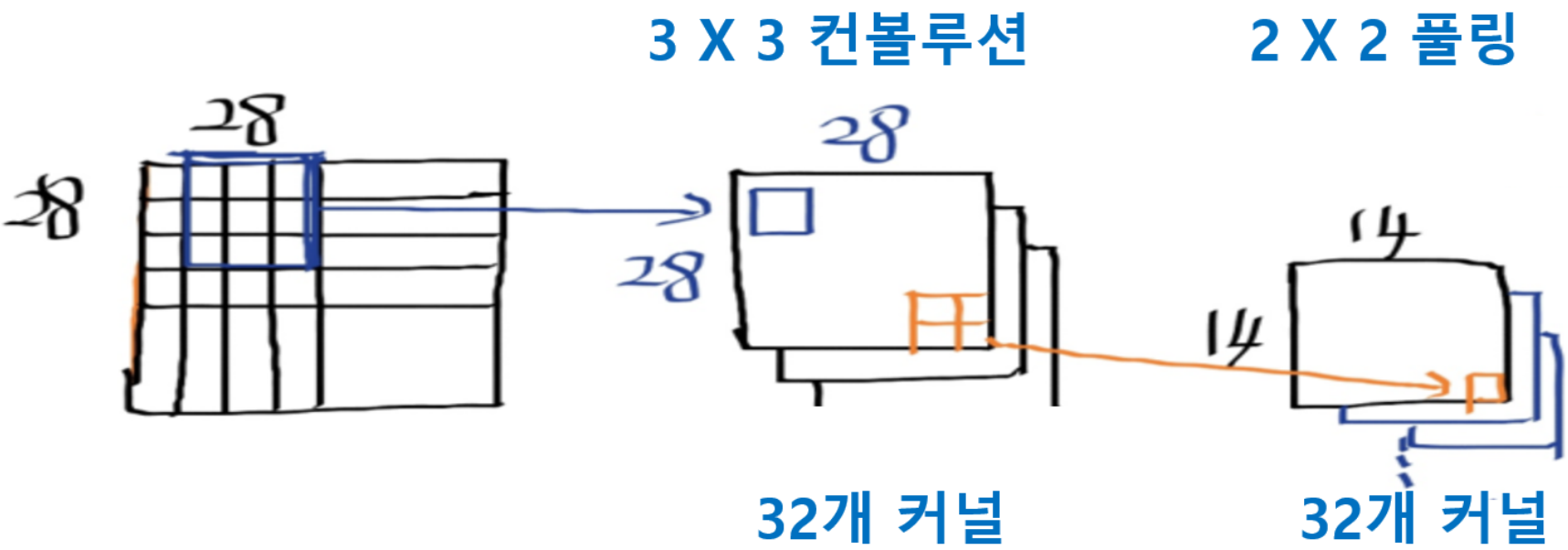
02. CNN 계층 구성

W1 [3 3 1 32] -> [3 3]: 커널 크기, 1: 입력값 X 의 특성수, 32: 필터 or 커널 갯수

L1 Conv shape=(?, 28, 28, 32)

Pool ->(?, 14, 14, 32)

▶ L1 계층 구성




```
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
L1 = tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding='SAME')  # 이미지의 가장 외곽에서 한 칸 밖으로 움직이는 옵션, 테두리까지도 평가함.
L1 = tf.nn.relu(L1)  # 활성화 함수
# Pooling 역시 tf.nn.max_pool 을 이용
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

tf.layers 모듈을 이용하여 위의 코드를 2줄로 줄일 수 있다

In [7]:
```
1 L1 = tf.layers.conv2d(X,32,[3,3])
2 L1 = tf.layers.max_pooling2d(L1, [2,2], [2,2])
```

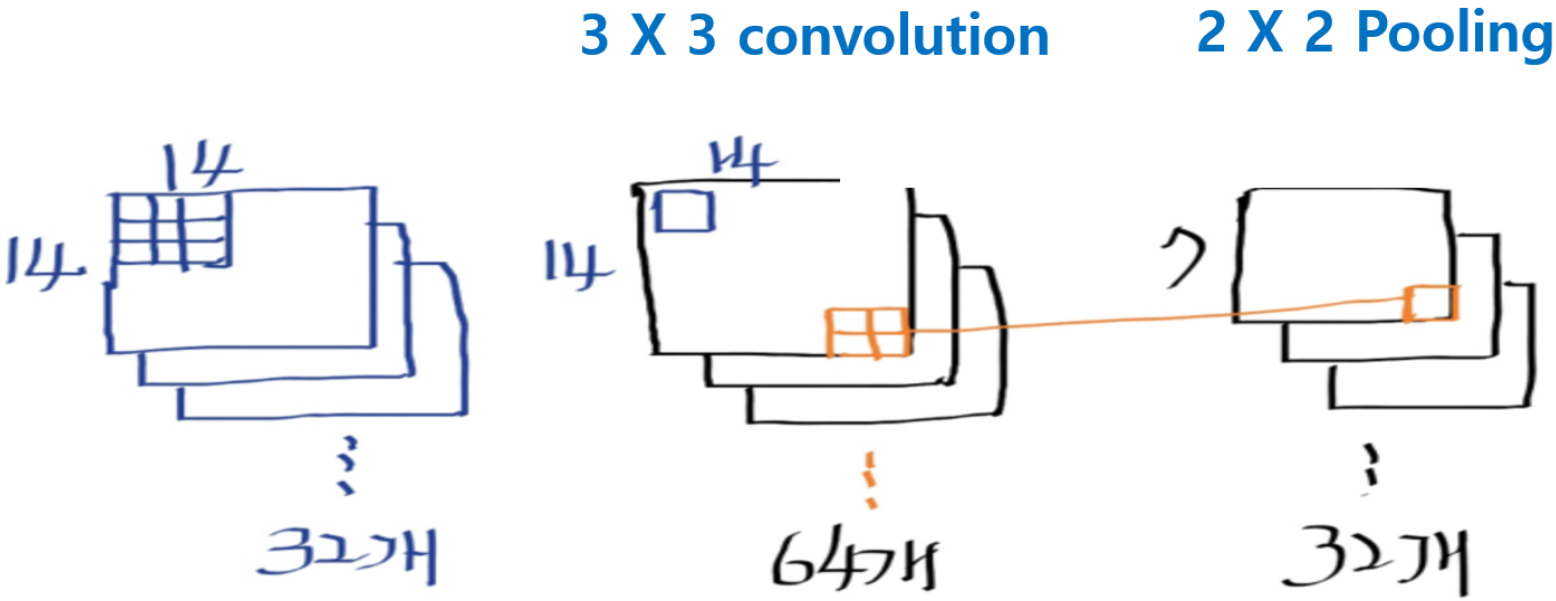
추가적으로 dropout 수행

In [8]:
```
L1 = tf.layers.dropout(L1, 0.7, is_training)
```

두번째 계층 구성

- 두 번째 컨볼루션 계층의 커널인 W2의 변수의 구성은 [3,3,32,64]이다.
- 32는 앞서 구성된 첫 번째 컨볼루션 계층의 커널 개수이다.
- 즉 출력층의 개수이며, 첫 번째 컨볼루션 계층이 찾아낸 이미지의 특징 개수라고 할 수 있다.

▶ L2 계층 구성



In [9]:
```
# W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
# L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
# L2 = tf.nn.relu(L2)
# L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L2 = tf.layers.conv2d(L1, 64, [3,3])
L2 = tf.layers.max_pooling2d(L2, [2,2], [2,2])
L2 = tf.layers.dropout(L2, 0.7, is_training)
```

완전 연결 계층을 만드는 부분의 이전 코드

In [12]:
```
# W3 = tf.Variable(tf.random_normal([7 * 7 * 64, 256], stddev=0.01))
# L3 = tf.reshape(L2, [-1, 7 * 7 * 64])
# L3 = tf.matmul(L3, W3)
# L3 = tf.nn.relu(L3)
# L3 = tf.nn.dropout(L3, keep_prob)
L3 = tf.contrib.layers.flatten(L2)
L3 = tf.layers.dense(L3, 256, activation=tf.nn.relu)
L3 = tf.layers.dropout(L3, 0.5, is_training)

```

In [13]:
```
# 최종 출력값 L3 에서의 출력 256개를 입력값으로 받아서 0~9 레이블인 10개의 출력값을 만듭니다.
# W4 = tf.Variable(tf.random_normal([256, 10], stddev=0.01))
# model = tf.matmul(L3, W4)
model = tf.layers.dense(L3, 10, activation=None)
```

In [14]:
```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y))
optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)
```

In [16]:

```
%%time

#########
# 신경망 모델 학습
######
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

batch_size = 100
total_batch = int(mnist.train.num_examples / batch_size)

for epoch in range(15):
    total_cost = 0

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        # 이미지 데이터를 CNN 모델을 위한 자료형태인 [28 28 1] 의 형태로 재구성합니다.
        batch_xs = batch_xs.reshape(-1, 28, 28, 1)

        _, cost_val = sess.run([optimizer, cost],
                               feed_dict={X: batch_xs,
                                          Y: batch_ys,
                                          is_training: True})
        total_cost += cost_val

    print('Epoch:', '%04d' % (epoch + 1),
          'Avg. cost =', '{:.3f}'.format(total_cost / total_batch))

print('최적화 완료!')

#########
# 결과 확인
######
is_correct = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
print('정확도:', sess.run(accuracy,
                        feed_dict={X: mnist.test.images.reshape(-1, 28, 28, 1),
                                   Y: mnist.test.labels,
                                   is_training: False}))
```

```
Epoch: 0001 Avg. cost = 0.178
Epoch: 0002 Avg. cost = 0.049
Epoch: 0003 Avg. cost = 0.031
Epoch: 0004 Avg. cost = 0.021
Epoch: 0005 Avg. cost = 0.016
Epoch: 0006 Avg. cost = 0.012
Epoch: 0007 Avg. cost = 0.012
Epoch: 0008 Avg. cost = 0.010
Epoch: 0009 Avg. cost = 0.007
Epoch: 0010 Avg. cost = 0.007
Epoch: 0011 Avg. cost = 0.009
Epoch: 0012 Avg. cost = 0.005
Epoch: 0013 Avg. cost = 0.005
Epoch: 0014 Avg. cost = 0.007
Epoch: 0015 Avg. cost = 0.005
최적화 완료!
정확도: 0.9906
Wall time: 9min 14s
```

In [ ]:

```
1
```