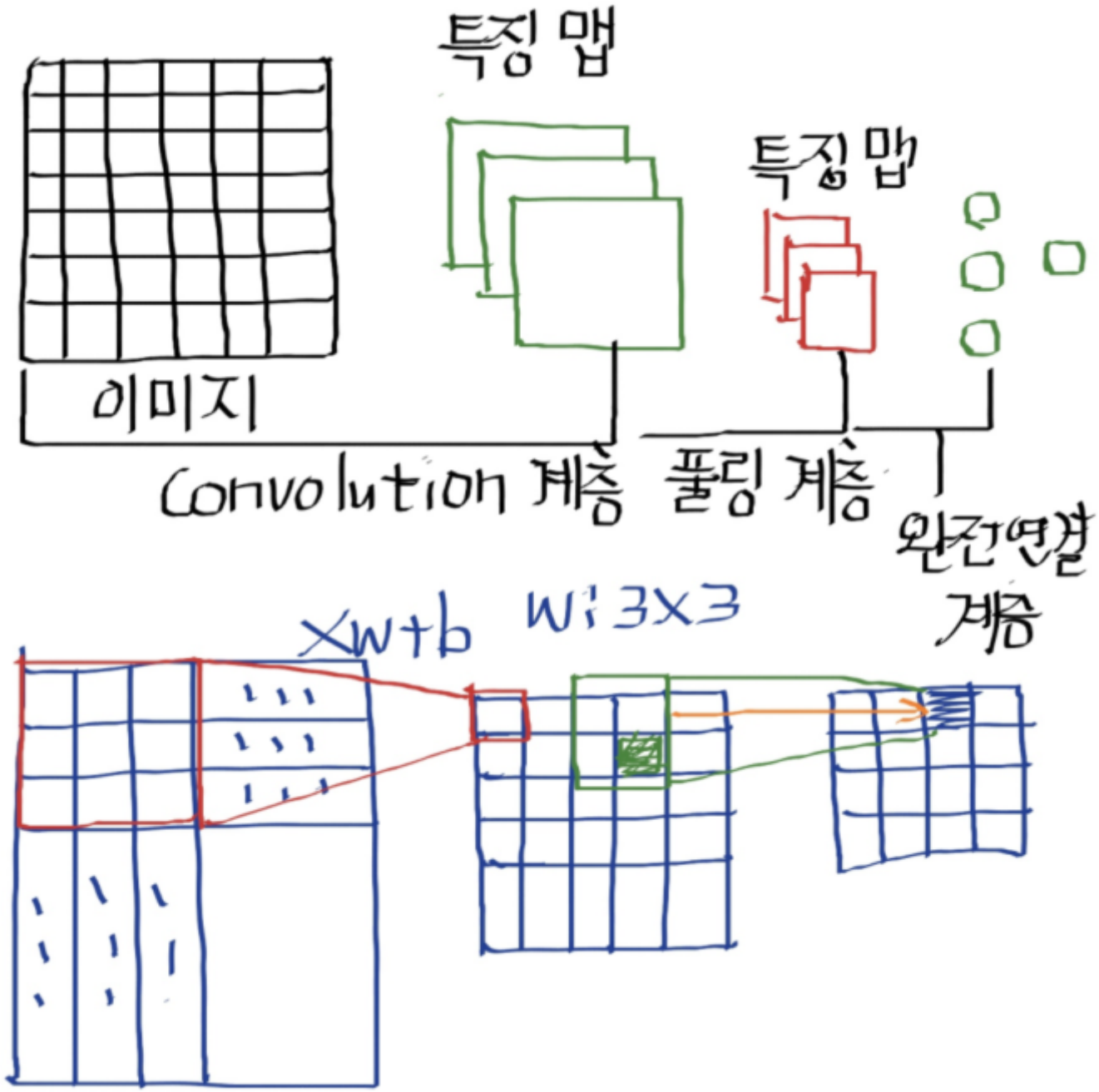
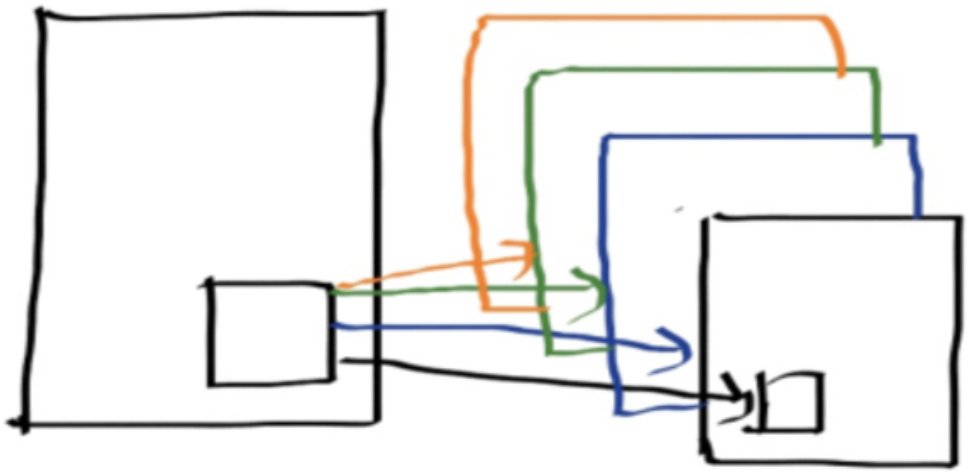
CNN(Convolution Neural Network)

- 1998년 얀 레쿤(YannLecun)교수가 소개한 이래로 사용되고 있는 신경망이다.
- CNN은 기본적으로 컨볼루션 계층과 풀링(pooling layer)로 구성된다.
- 평면의 경우, 행렬에서 지정한 영역의 값을 하나로 압축시킨다.
- 컨볼루션은 가중치와 편향을 적용하고,
- 풀링 계층은 단순한 값들중의 하나를 가져오는 방식을 취한다.


특징 맵
특징 맵
이미지
Convolution 계층
풀링 계층
완전연결 계층
XWtb
W:3X3

- 입력층의 원도우를 은닉층의 뉴런 하나로 압축할 때, 컨볼루션 계층에서는 원도우 크기만큼의 가중치와 1개의 편향을 적용
- 원도우의 크기가 3 X 3 이라면, 3 X 3개의 가중치와 1개의 편향이 필요하다.
- 3 X 3개의 가중치와 1개의 편향을 커널(kernel)또는 필터(filter)라고 한다.
- 기본 신경망의 784개의 가중치를 찾아야 한다.
- 컨볼루션 계층에서는 3X3개의 9개의 가중치만 찾아내면 된다.
- 복잡한 특징을 가진 이미지는 분석하기에 커널이 부족할 수 있으므로 보통 커널을 여러개 사용
- 커널의 개수를 정하는 일 매우 중요하다.

▶ 복잡한 특징을 가진 이미지는 분석하기에 커널이 부족할 수 있으므로 보통 커널을 여러 개 사용



In [2]:
```
# 이미지 처리 분야에서 가장 유명한 신경망 모델인 CNN 을 이용
import tensorflow as tf

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

```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩h5py₩__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype fr
om `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters

WARNING:tensorflow:From <ipython-input-2-0f29cbda13da>:5: read_data_sets (from tensorflow.contrib.learn.python.learn.datasets.mnis
t) is deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:260: m
aybe_download (from tensorflow.contrib.learn.python.learn.datasets.base) is deprecated and will be removed in a future version.
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:262: e
xtract_images (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting ./mnist/data/train-images-idx3-ubyte.gz
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:267: e
xtract_labels (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting ./mnist/data/train-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:110: d
ense_to_one_hot (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.one_hot on tensors.
Extracting ./mnist/data/t10k-images-idx3-ubyte.gz
Extracting ./mnist/data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩tensorflow₩contrib₩learn₩python₩learn₩datasets₩mnist.py:290: D
ataSet.__init__ (from tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
```

01. 신경망 모델 구성

In [4]:
```
#########
# 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])
keep_prob = tf.placeholder(tf.float32)

```

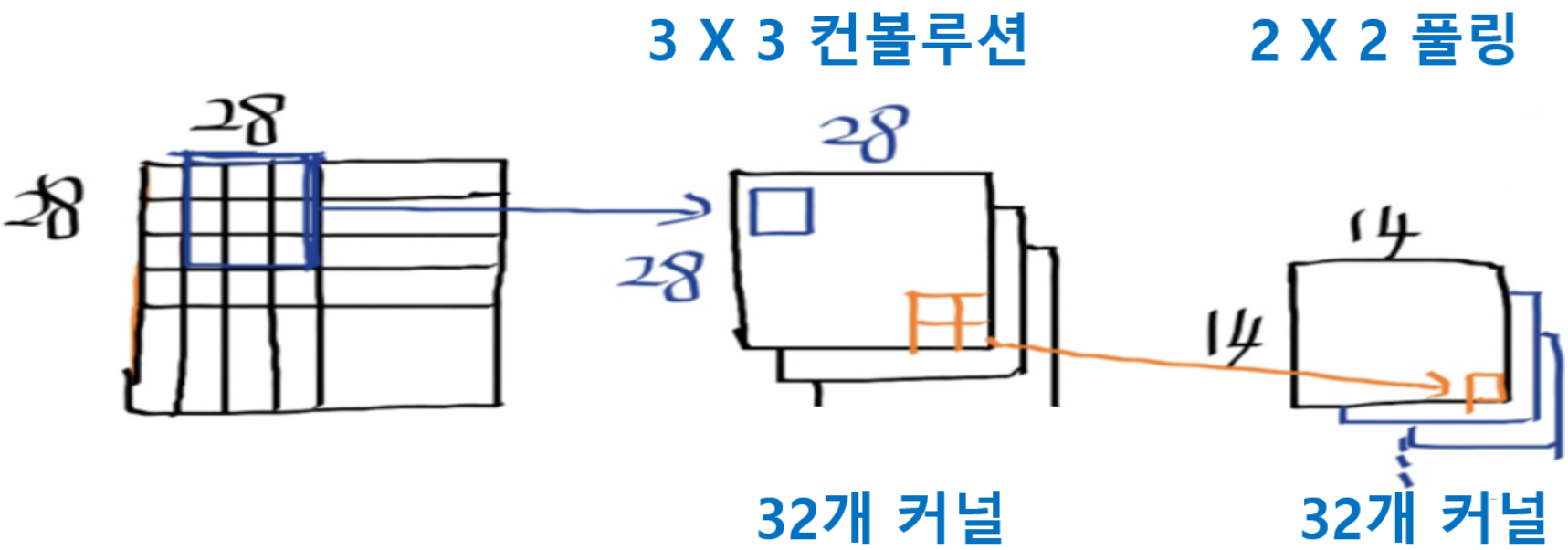
02. CNN 계층 구성

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



In [11]:
```
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, 테두리까지
L1 = tf.nn.relu(L1)  # 활성화 함수

```

In [12]:
```
print(W1)
print(L1)
```

```
<tf.Variable 'Variable_4:0' shape=(3, 3, 1, 32) dtype=float32_ref>
Tensor("Relu_4:0", shape=(?, 28, 28, 32), dtype=float32)
```

Pooling

- ksize : 2 X 2로 하는 풀링 계층
- strides : 슬라이딩 시 두 칸씩 움직이겠다.

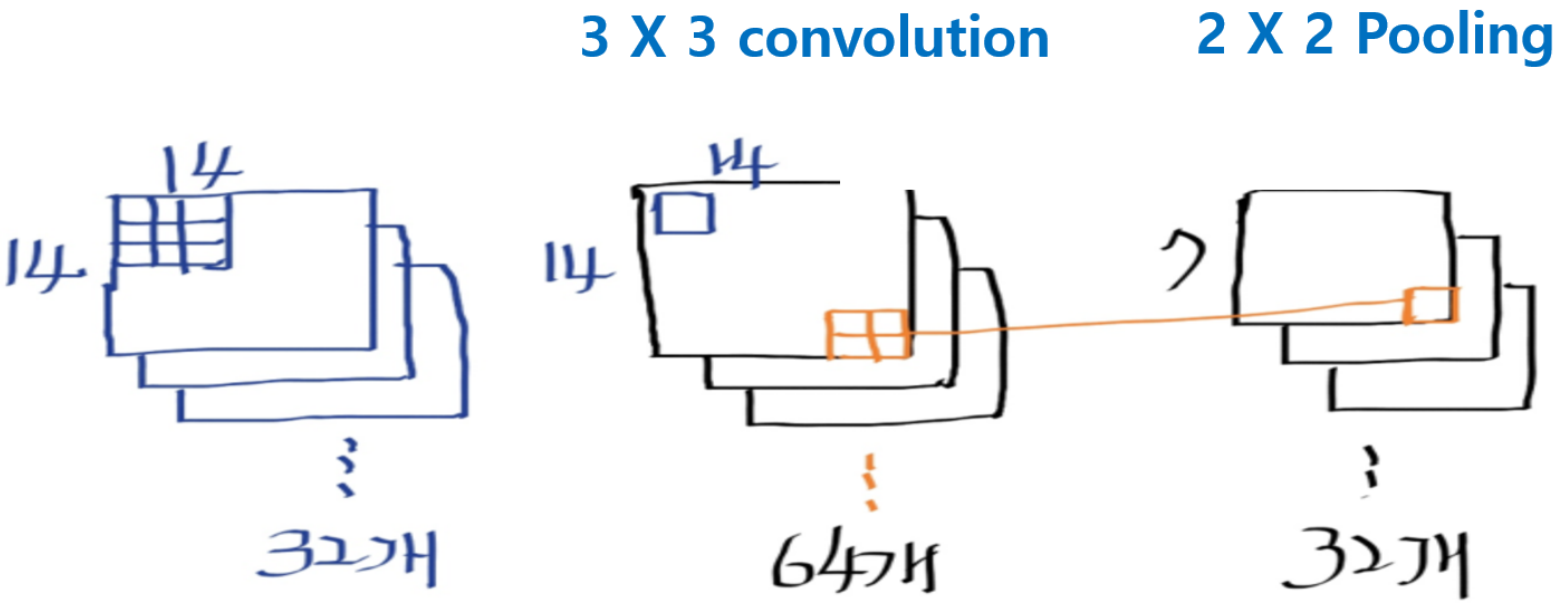
In [13]:
```
# Pooling 역시 tf.nn.max_pool 을 이용
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
print(L1)
```

```
Tensor("MaxPool_2:0", shape=(?, 14, 14, 32), dtype=float32)
```

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')
```

In [14]:
```
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)
```

In [15]:
```
# 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)
```

In [16]:
```
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 [17]:

```
#########
# 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,
                                          keep_prob: 0.7})
        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,
                                   keep_prob: 1}))
```

```
Epoch: 0001 Avg. cost = 0.329
Epoch: 0002 Avg. cost = 0.097
```

```
Epoch: 0003 Avg. cost = 0.069
Epoch: 0004 Avg. cost = 0.055
Epoch: 0005 Avg. cost = 0.043
Epoch: 0006 Avg. cost = 0.036
Epoch: 0007 Avg. cost = 0.032
Epoch: 0008 Avg. cost = 0.027
Epoch: 0009 Avg. cost = 0.024
Epoch: 0010 Avg. cost = 0.022
Epoch: 0011 Avg. cost = 0.019
Epoch: 0012 Avg. cost = 0.018
Epoch: 0013 Avg. cost = 0.015
Epoch: 0014 Avg. cost = 0.015
Epoch: 0015 Avg. cost = 0.013
최적화 완료!
정확도: 0.9917
```

In [ ]:
```
1
```