Python 수업 6회차

숙제1-1. 학습횟수를 올리기

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entrop

# Session
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

# Learning
for i in range(6000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

# Validation
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Result should be approximately 91%.
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.l
```

```
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
0.9233
```

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entrop

# Session
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

# Learning
for i in range(10000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

# Validation
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Result should be approximately 91%.
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.l
```

```
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
0.9243
```

숙제1-2. 학습률 조정

- 큰 효과는 없거나 나빠지는 경우가 빈번하므로 신중히 다뤄야한다.

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.001).minimize(cross_entropy)

# Session
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

# Learning
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

# Validation
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Result should be approximately 91%.
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

```
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
0.9106
```

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

y_ = tf.placeholder(tf.float32, [None, 10])

cross_entropy = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.03).minimize(cross_entropy)

# Session
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

# Learning
for i in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

# Validation
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Result should be approximately 91%.
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

```
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
0.098
```

숙제1-3. 다른 알고리즘 사용 등..

```
mnist = input_data.read_data_sets(DATA_DIR, one_hot=True)

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

x_image = tf.reshape(x, [-1, 28, 28, 1])
conv1 = conv_layer(x_image, shape=[5, 5, 1, 32])
conv1_pool = max_pool_2x2(conv1)

conv2 = conv_layer(conv1_pool, shape=[5, 5, 32, 64])
conv2_pool = max_pool_2x2(conv2)

conv2_flat = tf.reshape(conv2_pool, [-1, 7*7*64])
full_1 = tf.nn.relu(full_layer(conv2_flat, 1024))

keep_prob = tf.placeholder(tf.float32)
full1_drop = tf.nn.dropout(full_1, keep_prob=keep_prob)

y_conv = full_layer(full1_drop, 10)

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_conv, labels=y_))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

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

    for i in range(STEPS):
        batch = mnist.train.next_batch(MINIBATCH_SIZE)

        if i % 100 == 0:
            train_accuracy = sess.run(accuracy, feed_dict={x: batch[0], y_: batch[1],
                                                           keep_prob: 1.0})
            print("step {}, training accuracy {}".format(i, train_accuracy))

        sess.run(train_step, feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

    X = mnist.test.images.reshape(10, 1000, 784)
    Y = mnist.test.labels.reshape(10, 1000, 10)
    test_accuracy = np.mean(
        [sess.run(accuracy, feed_dict={x: X[i], y_: Y[i], keep_prob: 1.0}) for i in range(10)])

print("test accuracy: {}".format(test_accuracy))
```

숙제2

```
import tensorflow as tf

a = tf.constant(5)
b = tf.constant(2)
c = tf.placeholder(tf.int32, [])
e = b * c
d = b + a
f = e + d
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    print(sess.run(f, feed_dict = {c: 3}))
```

```
13
```

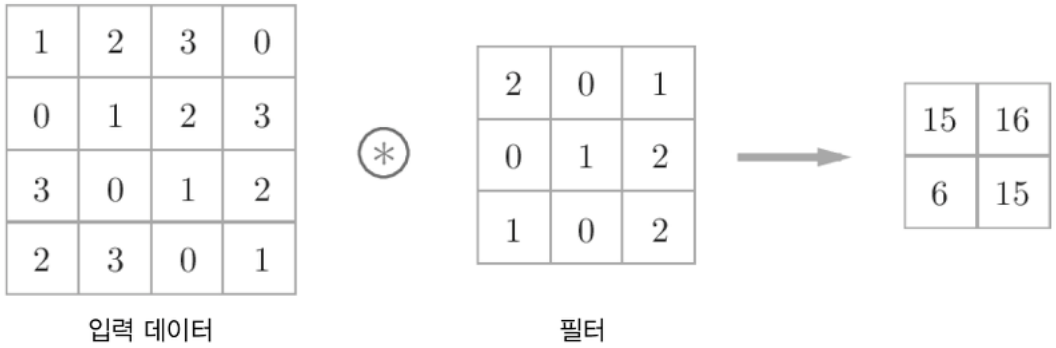
숙제3 –최소 아래의 개념 이해하고 오기

- CNN의
  - 합성곱계층(conv2d)
  - 풀링계층(max_pool_2x2)
  - 활성화 함수(Relu)

- 오버피팅을 막는 최적화 방법
  - 드롭아웃(Dropout)

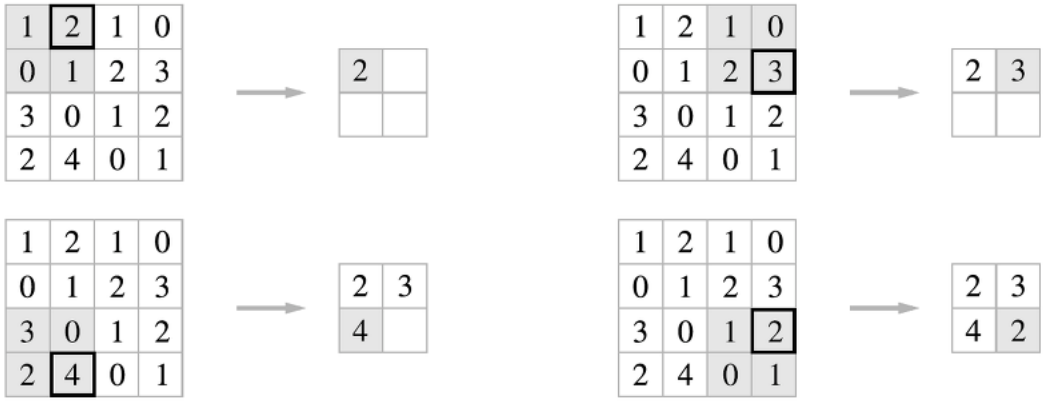
숙제3 - CNN의 핵심인 두 계층

- 합성곱(convolutional) 계층
  - 필터를 이용해 합성곱 연산을 수행한다. 필터(커널 = 가중치)의 윈도우를 일정 간격으로 이동해가며 입력 데이터에 적용한다.
  - 딥러닝과 달리 데이터 형상이 유지됨.



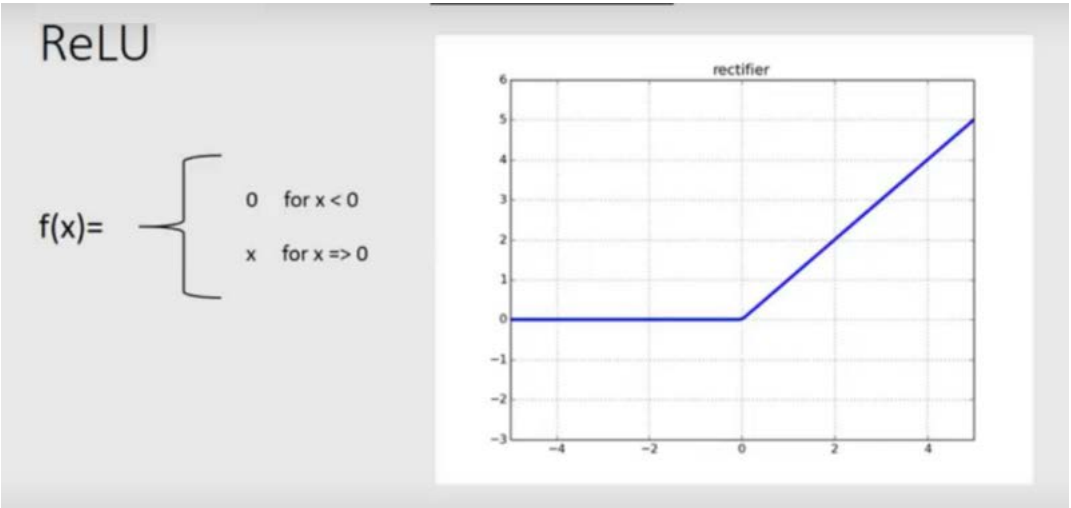
숙제3 - CNN의 핵심인 두 계층

- 풀링(pooling) 계층
  - 풀링은 2차원 데이터의 세로 및 가로 방향의 공간(매개변수)을 줄이는 연산
  - 풀링에는 최대 풀링(Max Pooling), 평균 풀링(Average Pooling) 등이 있다.
  - 최대 풀링은 대상 영역에서 최댓값을 취하는 연산, 평균 풀링은 대상 영역의 평균을 계산한다. 이미지 인식 분야에서는 주로 최대 풀링을 사용한다.



숙제3 – Relu란?

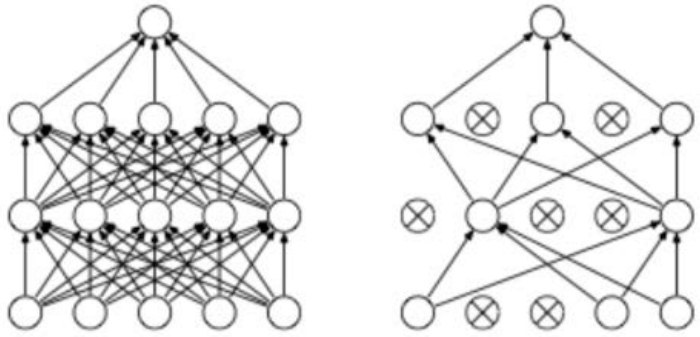
- Rectified Linear Unit 을 줄인말.
  - 개선된 선형함수.
- 활성화 함수로 자주 쓰이는 함수.
- 계산이 간단해 Sigmoid 보다 빠름.



숙제3 – Dropout 이란?

- Overfitting을 막기위한 방법.
- feature를 스스로 줄이는 방법이다.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting [Srivastava et al. 2014]



(a) Standard Neural Net

(b) After applying dropout.

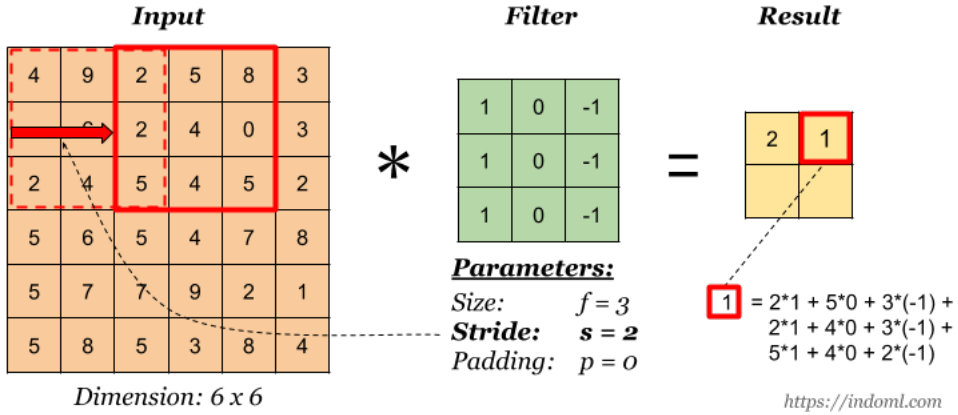
CNN 프로그래밍

CNN의 특징

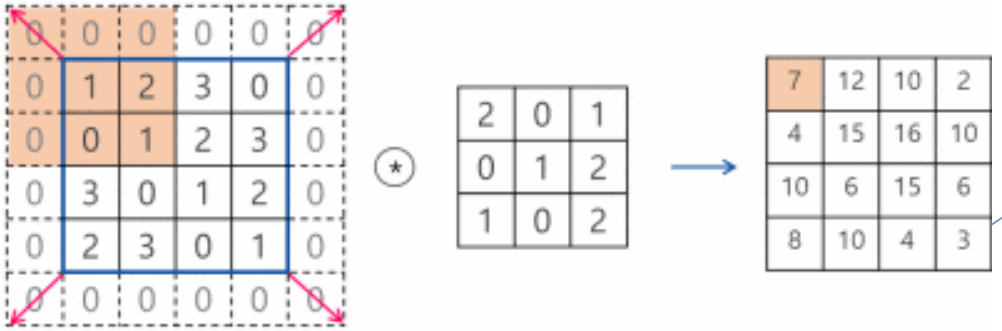
- 완전 연결이 아니다. -> 일정한 수의 유닛에 연결된다.
- 불변성 : 이미지내의 위치와 무관하게 물체를 찾을 수 있다.

CNN

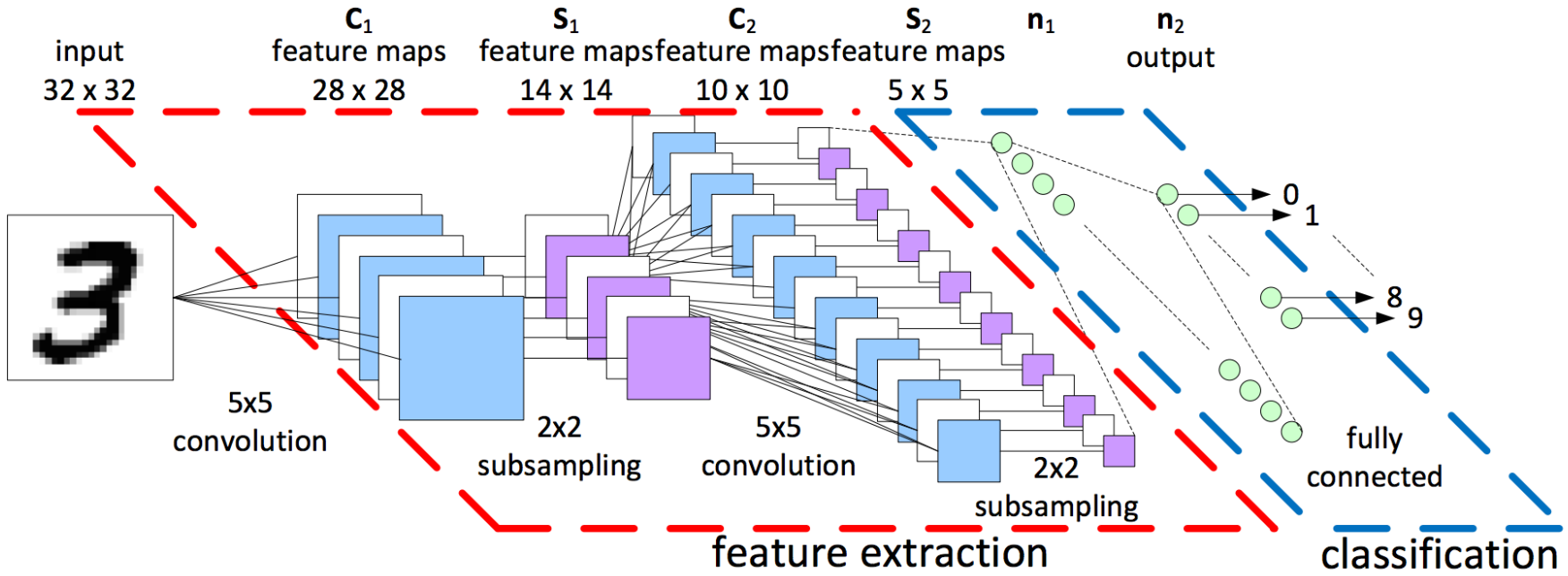
- Stride란?
  - Filter가 이동하는 칸 수.



- Padding란?
  - Img층의 최소한의 크기를 보장하기 위해 빈 테두리를 얼마나 추가하는지의 칸 수.



Mnist에 CNN 적용.



Mnist에 CNN 적용.

- (합성곱계층+풀링계층) 을 각각 2번씩,즉 2단계의 은닉층을 거치는 모델 짜보기.

CNN 프로그래밍

- 모델링 과정에서 어떤 부분들이 추가로 필요할까요?
  - 합성곱계층(conv2d)
  - 풀링계층(max_pool_2x2)
  - 활성화 함수(Relu)
  - 원래 이미지의 크기로 재 구성(reshape)
  - 적절한 초기화
  - 드롭아웃(Dropout)

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

# Dataset loading
mnist = input_data.read_data_sets("./samples/MNIST_data/", one_hot=True)

# Set up model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)        #출력 함수

y_ = tf.placeholder(tf.float32, [None, 10])
```

이곳에 최적화 코드가 좀더 추가된다.

```
cross_entropy = -tf.reduce_sum(y_*tf.log(y))              # 학습함수
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)  #경사하강법
```

1. 적절한 초기화

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

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

- Weight : 적절한 난수 (절단 정규분포)
- Bias : 0.1의 작은 값.

- 무작위한 값을 사용하면 학습된 특징 간의 대칭성을 무너뜨려 모델이 다양하고 풍부한 표현을 학습할 수 있다.
- 적당한 값의 범위를 지정하면 경삿값의 크기를 제어해서 네트워크가 효율적으로 수렴하게 할 수 있다.

2. 합성곱계층(conv2d)

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

- X : img 데이터
  - 다음과 같은 모양의 4차원 값이다. -> [None, 28, 28, 1]
  - 28 x 28 픽셀이며, 색 채널은 1개(회색조) 임을 나타낸다.
- W : 합성곱 filter
  - 형태는 다음과 같다. [5, 5, 1, 32]
  - 5 x 5 x 1의 '윈도' 크기, 32는 특징 맵의 수 이다.
- Stride : 슬라이딩 윈도우가 한번에 이동하는 크기 = 1로 설정
  - **[1, x, y, 1] -> first는 1개씩 batch 들 훑은 거다.**
  - [batch_size, image_rows, image_cols, number_of_colors]
- Padding : same
  - 연산 결과의 크기가 x의 크기와 같도록 x의 테두리에 패딩이 적용된다.

2. 합성곱계층(conv2d)

```
def conv_layer(input, shape):
    W = weight_variable(shape)
    b = bias_variable([shape[3]])
    return tf.nn.relu(conv2d(input, W) + b)
```

- 실제 사용할 합성곱 계층.
- Conv2d와 적절히 초기화된 변수값들을 이용한뒤, relu를 활성화함수로 사용.

3. 풀링계층(max_pool)

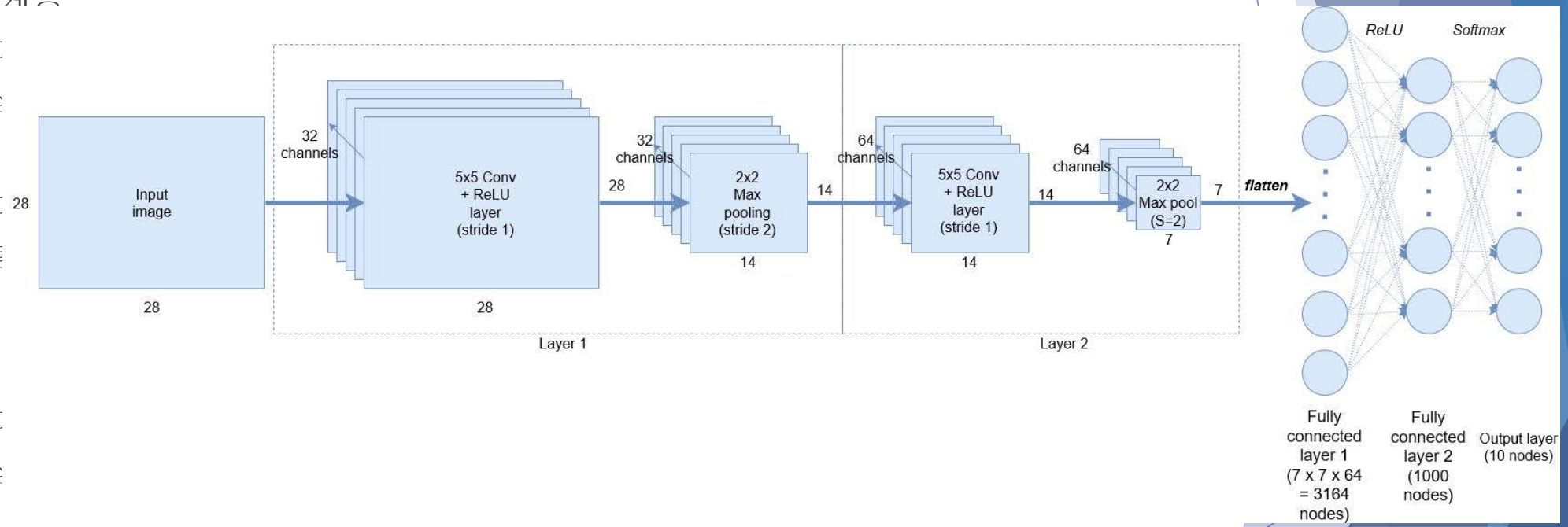
```
def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                          strides=[1, 2, 2, 1], padding='SAME')
```

(이전의 합성곱계층과 거의동일)

- X : img 데이터
- Ksize : 다룰 차원의 크기
  - 2 x 2 크기의 필터를 이용.
- Stride : 이동할 크기
- Padding : same

3. 이미지 변화 – 합성곱 & 풀링계층

- 1층
  - 합성곱 계층
    - 이미
    - 합성
  - 풀링 계
    - 이미
    - 풀링
- 2층
  - 합성곱
    - 이미
    - 합성
  - 풀링 계층
    - 이미지 : 14 x 14 x 64
    - 풀링(max) 필터 : 2 x 2
    - 7 x 7 x 64 의 이미지.



4. Full_layer()

```
def full_layer(input, size):
    in_size = int(input.get_shape()[1])
    W = weight_variable([in_size, size])
    b = bias_variable([size])
    return tf.matmul(input, W) + b
```

- 7 x 7 x 64 의 픽셀을 완전연결을 사용해 1 x 1 x 1024 픽셀로 만듬
- 마지막 출력층에 해당하므로 활성화함수인 relu를 안쓰고, 차후에 소프트맥스를 쓸 것임.

5. 드롭아웃(dropout)

```
keep_prob = tf.placeholder(tf.float32)
full1_drop = tf.nn.dropout(full_1, keep_prob=keep_prob)
```

- 계층 내의 유닛 중 임의의 사전에 설정된 부분을 학습 중 값을 0으로 세팅하여 꺼버리는 방식으로 시킴.
- 완전 연결한 1 x 1 x 1024 픽셀의 데이터에 적용.
- tf.nn.dropout(full_1, keep_prob=keep_prob)
  - 여기서 keep_prob는 각 단계에서 학습을 계속할 뉴런의 비율이다.
  - 학습단계(train_step) 에서는 0.5를 입력
  - 테스트 단계(accuracy)에서는 드롭아웃 적용하지 않으므로 1.0을 입력할 것이다.

전체 흐름을 봐보자

```
from tensorflow.examples.tutorials.mnist import input_data
import tensorflow as tf
import numpy as np

def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)


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


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


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


def conv_layer(input, shape):
    W = weight_variable(shape)
    b = bias_variable([shape[3]])
    return tf.nn.relu(conv2d(input, W) + b)


def full_layer(input, size):
    in_size = int(input.get_shape()[1])
    W = weight_variable([in_size, size])
    b = bias_variable([size])
    return tf.matmul(input, W) + b
```

```
DATA_DIR = '/tmp/data'
MINIBATCH_SIZE = 50
STEPS = 5000


mnist = input_data.read_data_sets(DATA_DIR, one_hot=True)

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

x_image = tf.reshape(x, [-1, 28, 28, 1])
conv1 = conv_layer(x_image, shape=[5, 5, 1, 32])
conv1_pool = max_pool_2x2(conv1)

conv2 = conv_layer(conv1_pool, shape=[5, 5, 32, 64])
conv2_pool = max_pool_2x2(conv2)

conv2_flat = tf.reshape(conv2_pool, [-1, 7*7*64])
full_1 = tf.nn.relu(full_layer(conv2_flat, 1024))

keep_prob = tf.placeholder(tf.float32)
full1_drop = tf.nn.dropout(full_1, keep_prob=keep_prob)

y_conv = full_layer(full1_drop, 10)

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_conv, labels=y_))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

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

    for i in range(STEPS):
        batch = mnist.train.next_batch(MINIBATCH_SIZE)

        if i % 100 == 0:
            train_accuracy = sess.run(accuracy, feed_dict={x: batch[0], y_: batch[1],
                                                           keep_prob: 1.0})
            print("step {}, training accuracy {}".format(i, train_accuracy))

        sess.run(train_step, feed_dict={x: batch[0], y_: batch[1], keep_prob: 0.5})

    X = mnist.test.images.reshape(10, 1000, 784)
    Y = mnist.test.labels.reshape(10, 1000, 10)
    test_accuracy = np.mean(
        [sess.run(accuracy, feed_dict={x: X[i], y_: Y[i], keep_prob: 1.0}) for i in range(10)])

print("test accuracy: {}".format(test_accuracy))
```

결과..

```
step 0, training accuracy 0.03999999910593033
step 100, training accuracy 0.8600000143051147
step 200, training accuracy 0.8199999928474426
step 300, training accuracy 0.9800000190734863
step 400, training accuracy 0.9399999976158142
step 500, training accuracy 0.9599999785423279
step 600, training accuracy 0.9200000166893005
step 700, training accuracy 0.9800000190734863
step 800, training accuracy 0.9399999976158142
step 900, training accuracy 0.9599999785423279
step 1000, training accuracy 0.9399999976158142
step 1100, training accuracy 0.9599999785423279
step 1200, training accuracy 0.9800000190734863
step 1300, training accuracy 0.9800000190734863
step 1400, training accuracy 0.9800000190734863
step 1500, training accuracy 0.9599999785423279
step 1600, training accuracy 0.9800000190734863
step 1700, training accuracy 0.9599999785423279
step 1800, training accuracy 0.9599999785423279
step 1900, training accuracy 1.0
step 2000, training accuracy 0.9599999785423279
step 2100, training accuracy 0.9599999785423279
step 2200, training accuracy 1.0
step 2300, training accuracy 1.0
step 2400, training accuracy 1.0
step 2500, training accuracy 0.9800000190734863
step 2600, training accuracy 1.0
step 2700, training accuracy 1.0
step 2800, training accuracy 0.9599999785423279
step 2900, training accuracy 0.9800000190734863
step 3000, training accuracy 0.9599999785423279
step 3000, training accuracy 0.9599999785423279
step 3100, training accuracy 1.0
step 3200, training accuracy 0.9800000190734863
step 3300, training accuracy 1.0
step 3400, training accuracy 0.9800000190734863
step 3500, training accuracy 1.0
step 3600, training accuracy 1.0
step 3700, training accuracy 1.0
step 3800, training accuracy 0.9599999785423279
step 3900, training accuracy 1.0
step 4000, training accuracy 0.9599999785423279
step 4100, training accuracy 0.9800000190734863
step 4200, training accuracy 1.0
step 4300, training accuracy 0.9800000190734863
step 4400, training accuracy 0.9599999785423279
step 4500, training accuracy 0.9599999785423279
step 4600, training accuracy 0.9599999785423279
step 4700, training accuracy 1.0
step 4800, training accuracy 1.0
step 4900, training accuracy 1.0
test accuracy: 0.9882999658584595
```

결과 분석

- 더 높은 정확도.
- 훈련에 더 오랜 시간이 걸림.

그밖에.. CNN

- CIFAR10 이라는 모델도 있다.
  - 10개의 분류 - 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭