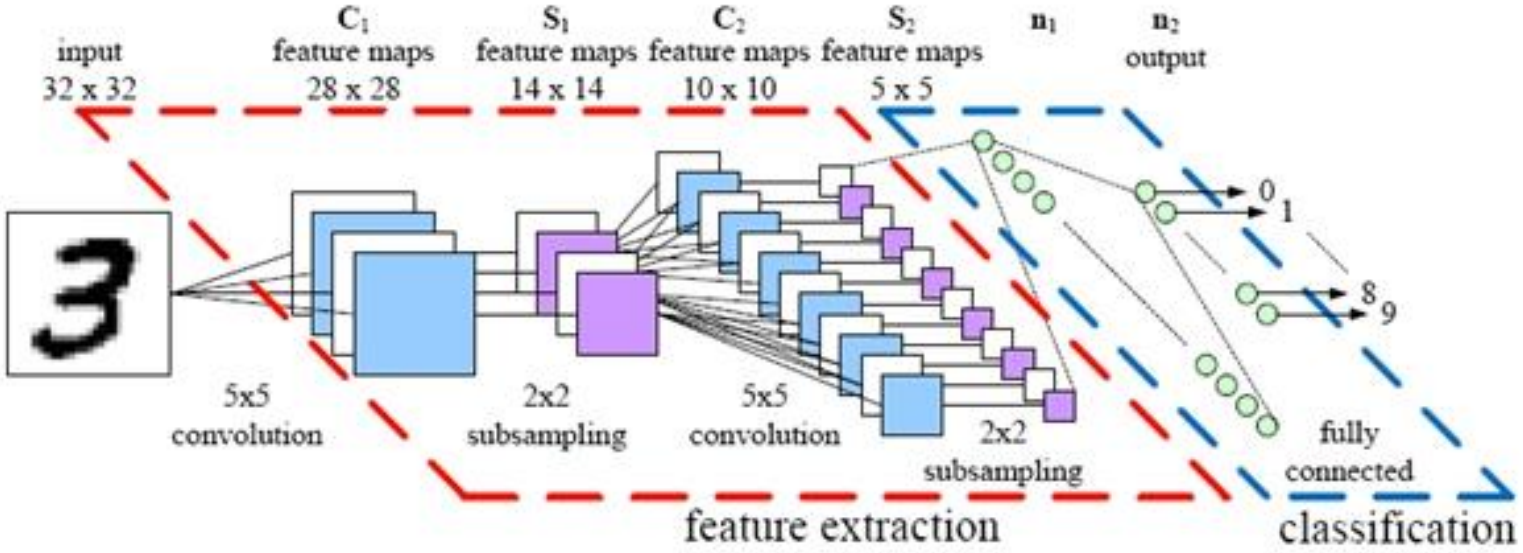
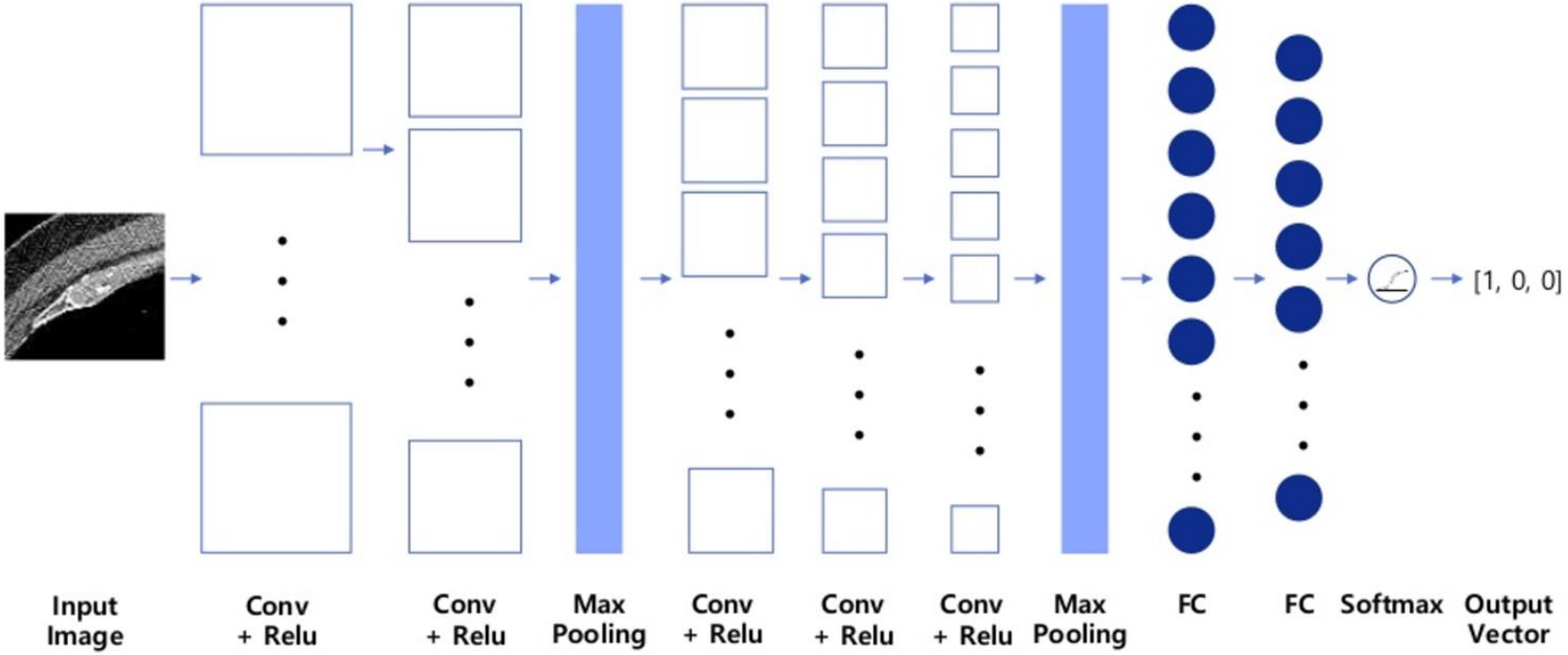
CNN

Lecture 11

CNN

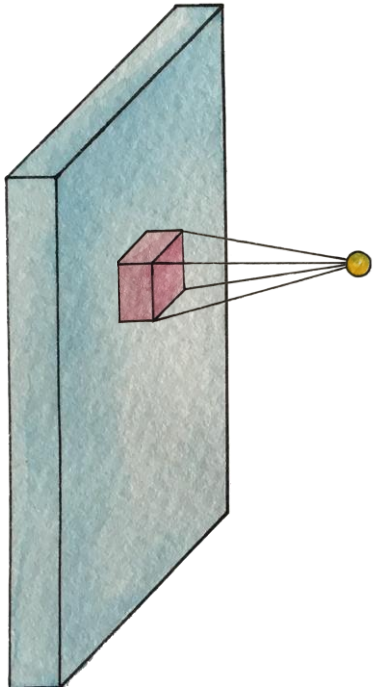

input
32 x 32
C1
feature maps
28 x 28
S1
feature maps
14 x 14
C2
feature maps
10 x 10
S2
feature maps
5 x 5
n1
n2
output
0
1
8
9
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
feature extraction
classification

CNN for CT images



Asan Medical Center & Microsoft Medical Bigdata Contest Winner by GeunYoung Lee and Alex Kim

Convolution layer and max pooling



Single depth slice

x

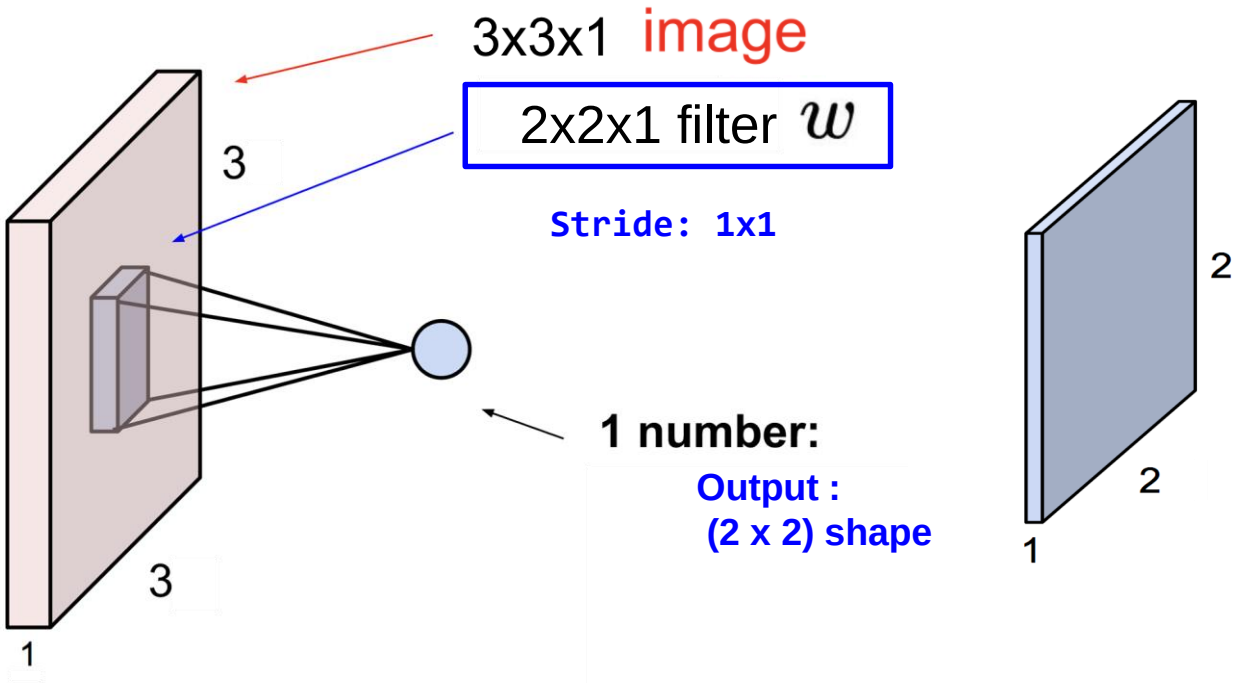
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

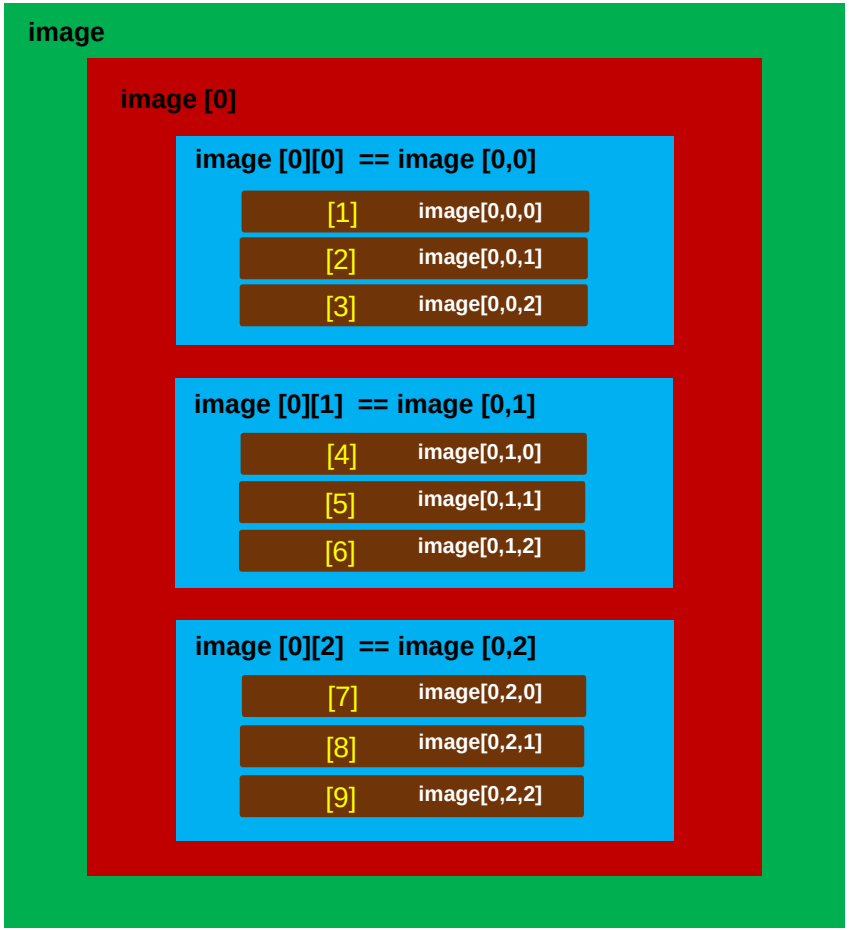
max pool with 2x2 filters
and stride 2

6	8
3	4

Simple convolution layer



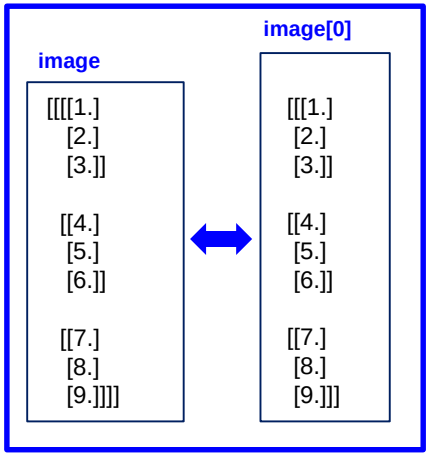
Dimension




```
image = np.array( [ [ [ [1],[2],[3] ],
                    [ [4],[5],[6] ],
                    [ [7],[8],[9] ] ] ], dtype=np.float32 )
```

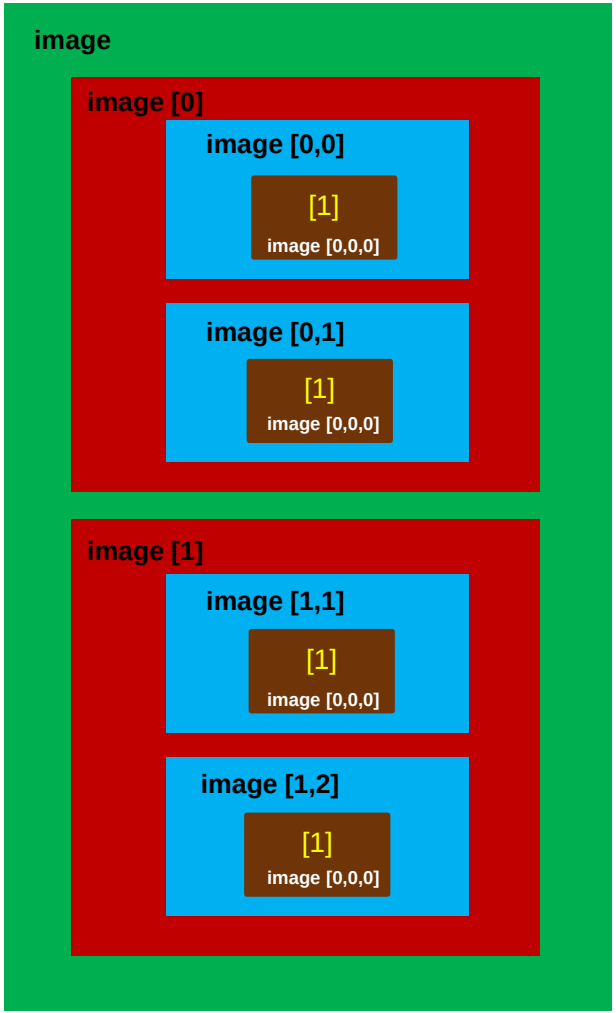
shape : (1, 3, 3, 1)

rank : 4




```
image shape :  (1, 3, 3, 1)
image rank :  4
image[0]
 [[[1.]
  [2.]
  [3.]]

 [[4.]
  [5.]
  [6.]]

 [[7.]
  [8.]
  [9.]]]
image[0,1]
 [[4.]
  [5.]
  [6.]]
image[0,1,0]
 [4.]
re_image [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
re_image shape :  (3, 3)
re_image rank :  2
```

Dimension




```
weight = tf.constant ( [ [ [ [1.] ], [ [1.] ] ],  [ [ [1.] ], [ [1.] ] ] )
```

shape : (2, 2, 1, 1)

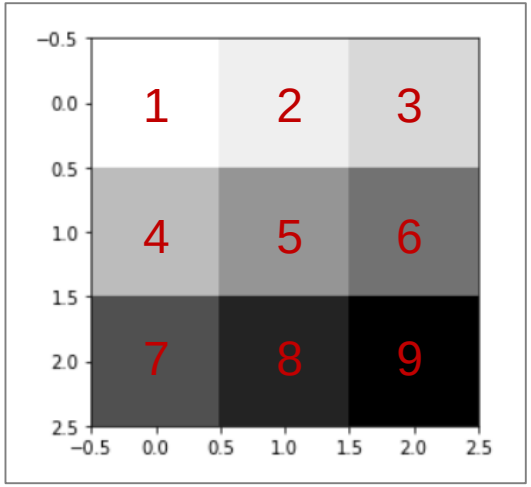
rank : 4

Image test

ex11_1.ipynb

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

# Tensor.eval() 메서드를 실행하기 위해서 선언함
sess = tf.InteractiveSession()

image = np.array([[[[1],[2],[3]],
                  [[4],[5],[6]],
                  [[7],[8],[9]]]], dtype=np.float32)
print("image shape : ", image.shape)
print("image rank : ", image.ndim)
print("image[0] \n", image[0])
print("image[0,1] \n", image[0,1])
print("image[0,1,0] \n", image[0,1,0])

re_image = image.reshape(3,3)
print("re_image", re_image)
print("re_image shape : ", re_image.shape)
print("re_image rank : ", re_image.ndim)

plt.imshow(re_image, cmap='Greys')
```

```
image shape :  (1, 3, 3, 1)
image rank :  4
image[0]
 [[[1.]
  [2.]
  [3.]]

 [[4.]
  [5.]
  [6.]]

 [[7.]
  [8.]
  [9.]]]
image[0,1]
 [[4.]
 [5.]
 [6.]]
image[0,1,0]
 [4.]
re_image [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
re_image shape :  (3, 3)
re_image rank :  2
```

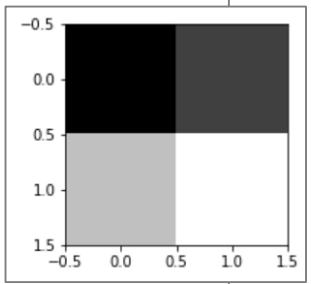


Simple convolution layer

ex11_2.ipynb

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

# 3 X 3 의 color가 하나인 image 생성 : shape (1, 3, 3, 1)
image = np.array([[[[1],[2],[3]],
                  [[4],[5],[6]],
                  [[7],[8],[9]]]], dtype=np.float32)

# Tensor.eval() 메서드를 실행하기 위해서 선언함
sess = tf.InteractiveSession()

print("● image.shape", image.shape)
print("image rank : ", image.ndim)
print("image", image)

weight = tf.constant([[[[1.]],[[1.]]],  [[[1.]],[[1.]]]])  # rank 4

print("● weight.shape", weight.shape)
print("weight rank : ", tf.rank(weight).eval())
print("weight (weight.eval()) \n", weight.eval())  # eval() 은 tensor를 run 하지않고 실행하는 방법
print("weight (sess.run([weight])) \n", sess.run([weight])) # sess.run 으로 실행해도 됨

conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1] , padding='VALID')
conv2d_img = conv2d.eval()                    # 'VALID' : padding을 하지 않는다는 뜻
print('='*40)
print("● conv2d_img.shape", conv2d_img.shape)
print("conv2d_img \n ", conv2d_img)
print('='*40)
print ("● conv2d_img.reshape(2,2) \n",conv2d_img.reshape(2,2))
print('='*40)

plt.subplot(1,2,1)  # (1, 2, 1) → ( rows 비율, cols 비율, 1장 그린다는 뜻)
plt.imshow(conv2d_img.reshape(2,2), cmap='gray')
```

```
● image.shape (1, 3, 3, 1)
image rank :  4
image [[[[1.]
   [2.]
   [3.]]

  [[4.]
   [5.]
   [6.]]

  [[7.]
   [8.]
   [9.]]]]
● weight.shape (2, 2, 1, 1)
weight rank :  4
weight (weight.eval())
 [[[[1.]]

  [[1.]]]


 [[[1.]]

  [[1.]]]]
weight (sess.run([weight]))
 [array([[[[1.]],

        [[1.]]],


       [[[1.]],

        [[1.]]]], dtype=float32)]
========================================
● conv2d_img.shape (1, 2, 2, 1)
conv2d_img
  [[[[12.]
   [16.]]

  [[24.]
   [28.]]]]
========================================
● conv2d_img.reshape(2,2)
[[12. 16.]
 [24. 28.]]
========================================
```



next slide
설명

```
image = np.array([[[[1],[2],[3]],
                  [[4],[5],[6]],
                  [[7],[8],[9]]]], dtype=np.float32)
.
.
weight = tf.constant([[[[1.]],[[1.]]],  [[[1.]],[[1.]]]])
.
.
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1] , padding='VALID')
```

image.shape (1, 3, 3, 1)
1장, 3x3, color 종류 하나이므로 1

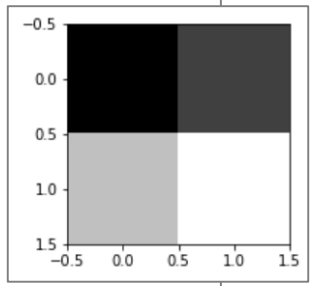
weight.shape (2, 2, 1, 1)
2x2, color 1, filter 개수 1

Stride: 1,1,1,1
(1x1)로 움직인다는 뜻

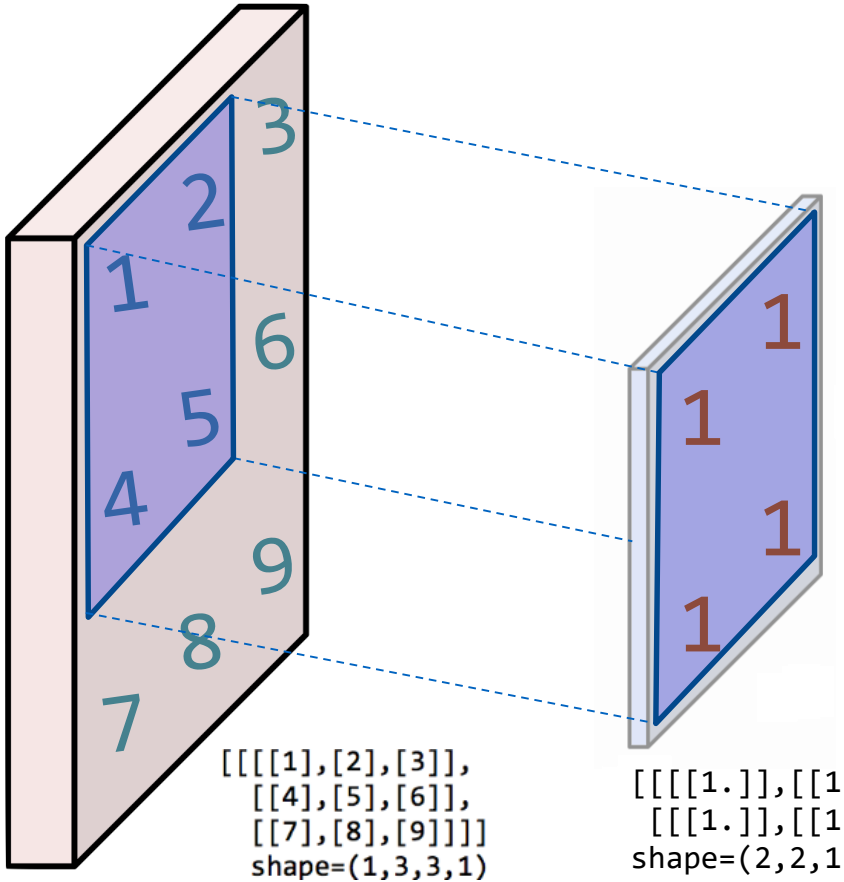
conv2d_img.shape (1, 2, 2, 1)
2x2 로 출력된다는 뜻

```
● image.shape (1, 3, 3, 1)
image rank :  4
image [[[[1.]
  [2.]
  [3.]]

 [[4.]
  [5.]
  [6.]]

 [[7.]
  [8.]
  [9.]]]
● weight.shape (2, 2, 1, 1)
weight rank :  4
weight (weight.eval())
 [[[[1.]]

  [[1.]]]


 [[[1.]]

  [[1.]]]]
weight (sess.run([weight]))
 [array([[[[1.]],

        [[1.]]],


       [[[1.]],

        [[1.]]]], dtype=float32)]
========================================
● conv2d_img.shape (1, 2, 2, 1)
conv2d_img
 [[[[12.]
  [16.]]

 [[24.]
  [28.]]]]
========================================
● conv2d_img.reshape(2,2)
 [[12. 16.]
 [24. 28.]]
========================================
```

ex11_2.ipynb (Notes)



ex11_2.ipynb (Notes)

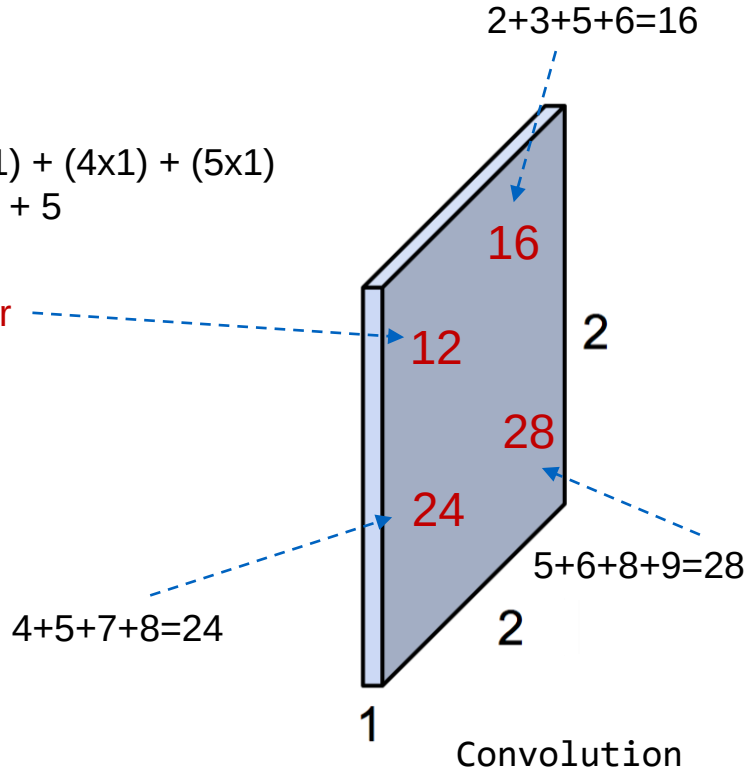



```
[[[[1],[2],[3]],
  [[4],[5],[6]],
  [[7],[8],[9]]]]
shape=(1,3,3,1)
```

```
[[[[1.]],[[1.]]],
 [[[1.]],[[1.]]]]
shape=(2,2,1,1)
```

(1x1) + (2x1) + (4x1) + (5x1)
= 1 + 2 + 4 + 5
= 12

One number



Simple convolution layer (padding='SAME')

ex11_3.ipynb

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

```
# 3 X 3 의 color가 하나인 image 생성 : shape (1, 3, 3, 1)
image = np.array([[[[1],[2],[3]],
                  [[4],[5],[6]],
                  [[7],[8],[9]]]], dtype=np.float32)
```

```
# Tensor.eval() 메서드를 실행하기 위해서 선언함
tf.InteractiveSession()
print("● image.shape", image.shape)
print("image \n ", image)
```

```
weight = tf.constant([[[[1.]],[[1.]]],  [[[1.]],[[1.]]]])
```

```
print("● weight.shape", weight.shape)
```

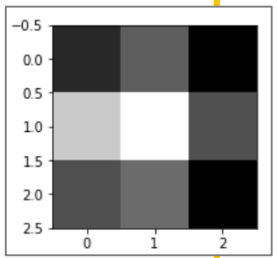
Input image 크기와 동일하게

```
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1] ,padding='SAME')
conv2d_img = conv2d.eval()  # eval() 은 tensor를 run 하지않고 실행하는 방법
print('='*40)
print("● conv2d_img.shape", conv2d_img.shape)
print("conv2d_img \n ", conv2d_img)
print('='*40)
print ("● conv2d_img.reshape(3,3) \n",conv2d_img.reshape(3,3))
print('='*40)
```

next slide 설명

```
plt.subplot(1,2,1)
plt.imshow(conv2d_img.reshape(3,3), cmap='gray')
```

```
● image.shape (1, 3, 3, 1)
image
 [[[[1.]
    [2.]
    [3.]]

   [[4.]
    [5.]
    [6.]]

   [[7.]
    [8.]
    [9.]]]]
● weight.shape (2, 2, 1, 1)
========================================
● conv2d_img.shape (1, 3, 3, 1)
conv2d_img
 [[[[12.]
    [16.]
    [ 9.]]

   [[24.]
    [28.]
    [15.]]

   [[15.]
    [17.]
    [ 9.]]]]
========================================
● conv2d_img.reshape(3,3)
 [[12. 16.  9.]
 [24. 28. 15.]
 [15. 17.  9.]]
========================================
```



```
image.shape (1, 3, 3, 1)
            1장, 3x3, color 종류 하나이므로 1
```

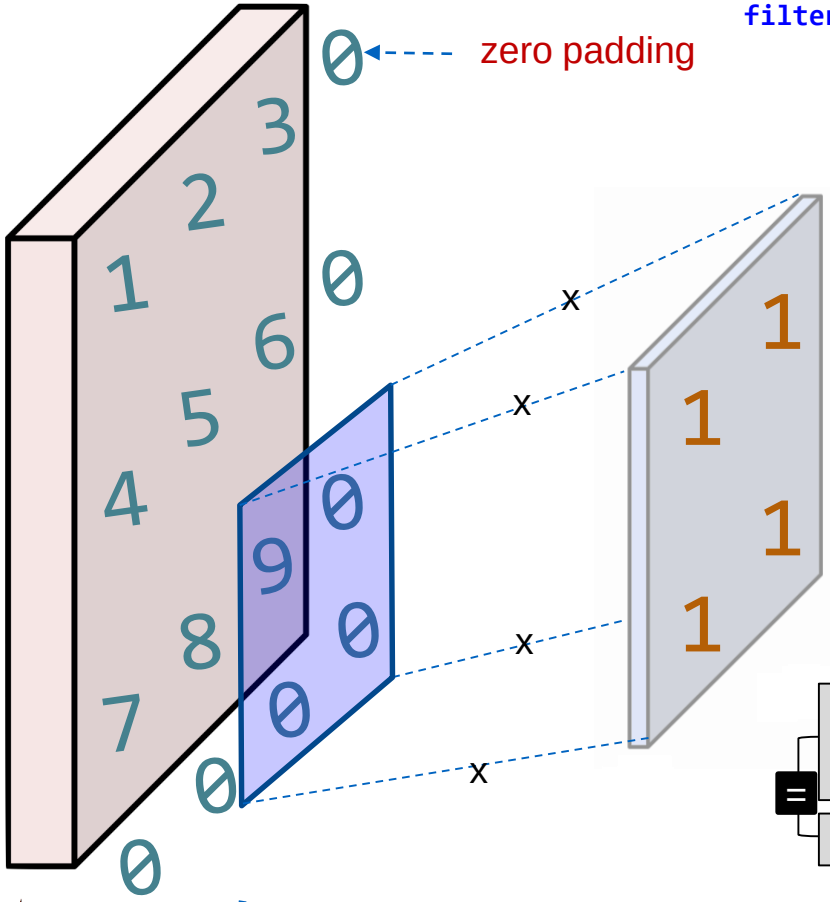
```
weight.shape (2, 2, 1, 1)
(Filter)   2x2, color 1, filter 개수 1
```

Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding: **SAME**

filter 개수 1

3X3

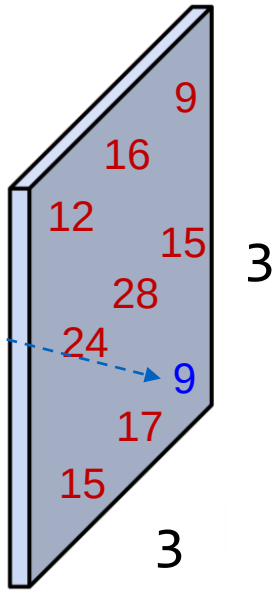
ex11_3.ipynb (Notes)



Input image size 와 동일하게 output image size를 출력해준다는 뜻

9 + 0 + 0 + 0
= 9

One number



Output size → (N – F) / stride size + 1

(4 - 2) / 1 + 1 = 3

=

Original Input size → 3

Simple convolution layer (filters)

ex11_4.ipynb

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

```
# 3 X 3 의 color가 1 가지인 image 생성 : shape (1, 3, 3, 1)
image = np.array([[[[1],[2],[3]],
                  [[4],[5],[6]],
                  [[7],[8],[9]]]], dtype=np.float32)

# Tensor.eval() 메서드를 실행하기 위해서 선언함
tf.InteractiveSession()

print("image.shape", image.shape)
print("image\n", image)
```

```
image.shape (1, 3, 3, 1)
image
[[[[1.]
   [2.]
   [3.]]

  [[4.]
   [5.]
   [6.]]

  [[7.]
   [8.]
   [9.]]]]
```

```
weight = tf.constant([[[[1.,10.,-1.]],[[1.,10.,-1.]]],
                      [[[1.,10.,-1.]],[[1.,10.,-1.]]]])
print("weight.shape", weight.shape)
```

weight.shape (2, 2, 1, 3)

3 filters

(2,2,1,3)

window 2x2,

color 1,

filter 개수 3

Output image 3 장

```
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1], padding='SAME')
conv2d_img = conv2d.eval()
```

```
print("conv2d_img.shape", conv2d_img.shape)
print("● conv2d_img\n", conv2d_img)

conv2d_img = np.swapaxes(conv2d_img, 0, 3)
print("● swap -> conv2d_img\n", conv2d_img)

for i, one_img in enumerate(conv2d_img):
    print("="*10, " i : ", i, "="*10)
    print("● one_img shape \n",one_img.shape)
    print("● one_img \n",one_img)
    print("● one_img (3 x 3) reshape \n",one_img.reshape(3,3))
    plt.subplot(1,3,i+1)
    plt.imshow(one_img.reshape(3,3), cmap='gray')
```

next slide 설명

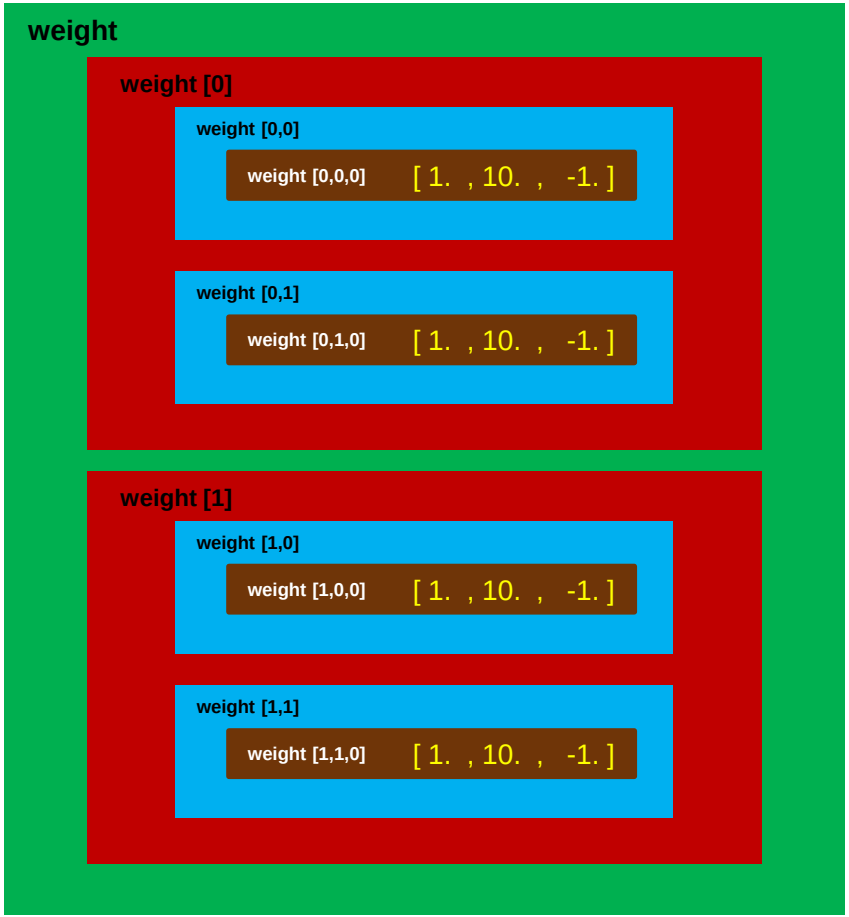
코딩을 추가하면...

```
print("weight \n", weight.eval())
```

1	10	-1
1	10	-1

1	10	-1
1	10	-1

next slide 설명



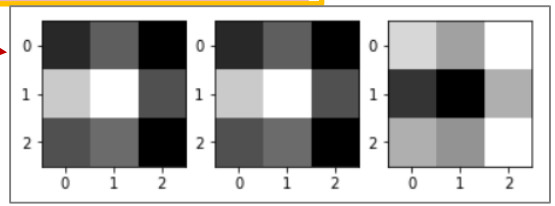
weight

```
[ [ [ [ 1. 10. -1.] ]
    [ [ 1. 10. -1.] ] ]

  [ [ [ 1. 10. -1.] ]
    [ [ 1. 10. -1.] ] ] ]
```

```
.
image = np.array([[[[1],[2],[3]],
                  [[4],[5],[6]],
                  [[7],[8],[9]]]], dtype=np.float32)
.
.

weight = tf.constant([[[[1.,10.,-1.]],[[1.,10.,-1.]]],
                      [[[1.,10.,-1.]],[[1.,10.,-1.]]]])
print("weight.shape", weight.shape)

conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1], padding='SAME')
conv2d_img = conv2d.eval()
```

```
1 print("conv2d_img.shape", conv2d_img.shape)
  print("● conv2d_img\n", conv2d_img)
  # np.swapaxes (conv2d_img,0,3) → conv2d_img의 0번 축과 3번 축을 맞교환
  conv2d_img = np.swapaxes(conv2d_img, 0, 3)
2 print("● swap -> conv2d_img\n", conv2d_img)

  for i, one_img in enumerate(conv2d_img):
      print("="*10, " i : ", i, "="*10)
3     print("● one_img shape \n",one_img.shape)
      print("● one_img \n",one_img)
      print("● one_img (3 x 3) reshape \n",one_img.reshape(3,3))

      plt.subplot(1,3,i+1)
      plt.imshow(one_img.reshape(3,3), cmap='gray')
```

```
1 weight.shape (2, 2, 1, 3)
conv2d_img.shape (1, 3, 3, 3)
● conv2d_img
[[[[ 12. 120. -12.]
   [ 16. 160. -16.]
   [  9.  90.  -9.]]

  [[ 24. 240. -24.]
   [ 28. 280. -28.]
   [ 15. 150. -15.]]

  [[ 15. 150. -15.]
   [ 17. 170. -17.]
   [  9.  90.  -9.]]]]
```



np.swapaxes (conv2d_img,0,3) → conv2d_img의 0번 축과 3번 축을 swapping 한 후 3장의 이미지로 변환

2 ● swap -> conv2d_img

```
[[[[ 12.]
   [ 16.]
   [  9.]]

  [[ 24.]
   [ 28.]
   [ 15.]]

  [[ 15.]
   [ 17.]
   [  9.]]]


 [[[120.]
   [160.]
   [ 90.]]

  [[240.]
   [280.]
   [150.]]

  [[150.]
   [170.]
   [ 90.]]]


 [[[-12.]
   [-16.]
   [ -9.]]

  [[-24.]
   [-28.]
   [-15.]]

  [[-15.]
   [-17.]
   [ -9.]]]]
```

3

```
========== i :  0 ==========
● one_img shape
(3, 3, 1)
● one_img
[[[12.]
  [16.]
  [ 9.]]

 [[24.]
  [28.]
  [15.]]

 [[15.]
  [17.]
  [ 9.]]]
● one_img (3 x 3) reshape
[[12. 16.  9.]
 [24. 28. 15.]
 [15. 17.  9.]]
```

for i, one_img in enumerate(conv2d_img):

사용하여 conv2d_img 를
3장의 이미지로 분리하고
reshape 으로 shape를 조정함

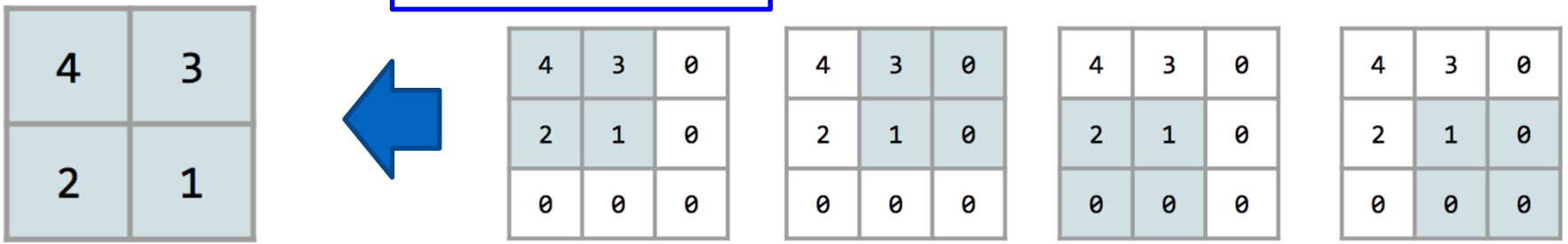
```
========== i :  1 ==========
● one_img shape
(3, 3, 1)
● one_img
[[[120.]
  [160.]
  [ 90.]]

 [[240.]
  [280.]
  [150.]]

 [[150.]
  [170.]
  [ 90.]]]
● one_img (3 x 3) reshape
[[120. 160.  90.]
 [240. 280. 150.]
 [150. 170.  90.]]
```

```
========== i :  2 ==========
● one_img shape
(3, 3, 1)
● one_img
[[[-12.]
  [-16.]
  [ -9.]]

 [[-24.]
  [-28.]
  [-15.]]

 [[-15.]
  [-17.]
  [ -9.]]]
● one_img (3 x 3) reshape
[[-12. -16.  -9.]
 [-24. -28. -15.]
 [-15. -17.  -9.]]
```

Max Pooling

ex11_5.ipynb

SAME: Zero paddings




```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
tf.InteractiveSession()

image = np.array([[[[4],[3]],
                  [[2],[1]]]], dtype=np.float32)

print("image \n",image)
pool = tf.nn.max_pool(image, ksize=[1, 2, 2, 1], strides=[1, 1, 1, 1], padding='VALID')

print("● pool.shape   \n ",pool.shape)

print("● pool.eval()  \n ",pool.eval())
```

kernel size

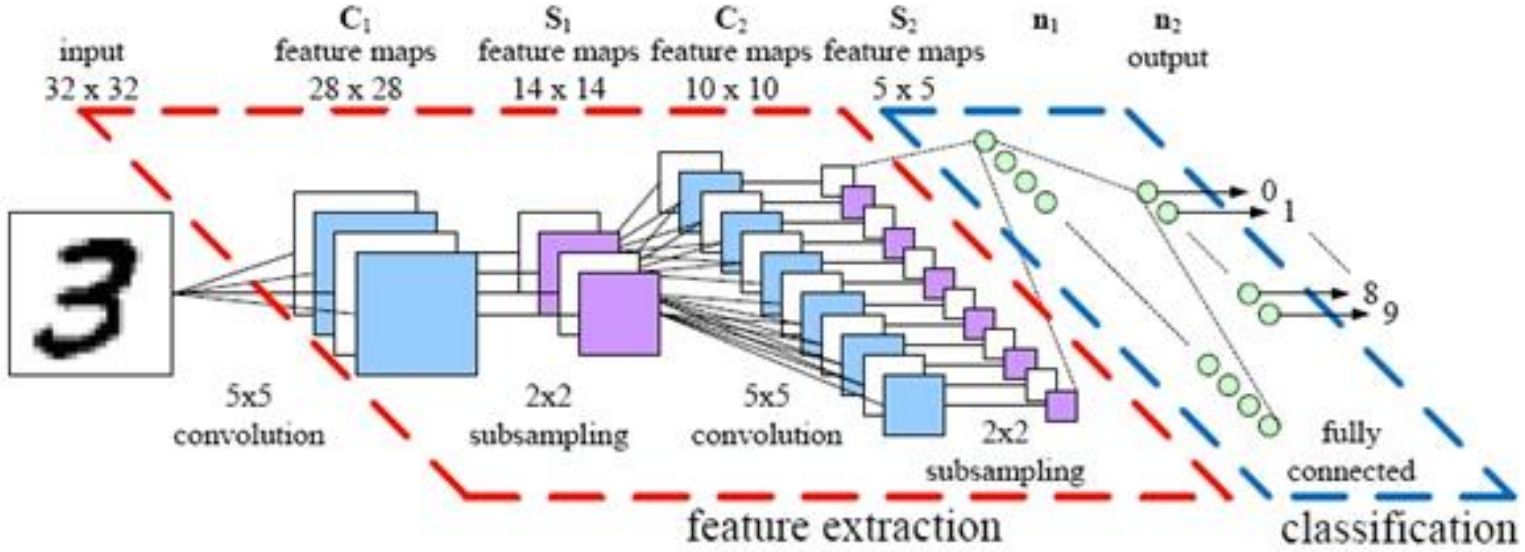
```
image
 [[[[4.]
   [5.]]

  [[2.]
   [1.]]]]
● pool.shape
 (1, 1, 1, 1)
● pool.eval()
 [[[[4.]]]]
```

, padding='SAME')

```
image
 [[[[4.]
   [5.]]

  [[2.]
   [1.]]]]
● pool.shape
 (1, 2, 2, 1)
● pool.eval()
 [[[[4.]
   [3.]]

  [[2.]
   [1.]]]]
```

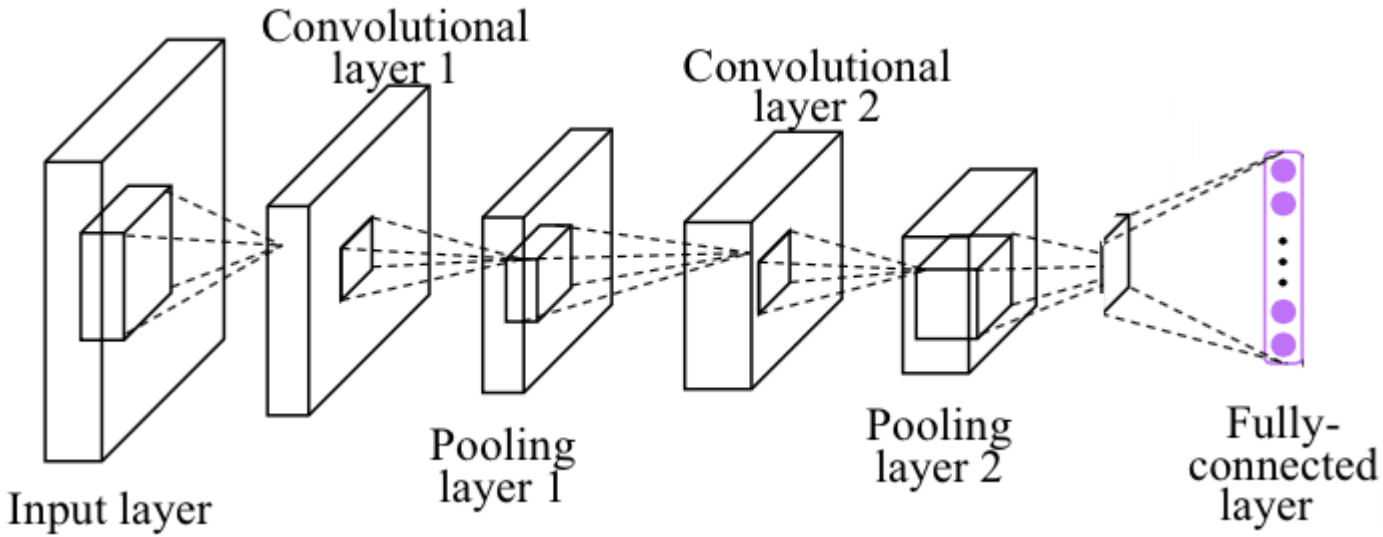
CNN MNIST: 99%!

CNN



http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

Simple CNN

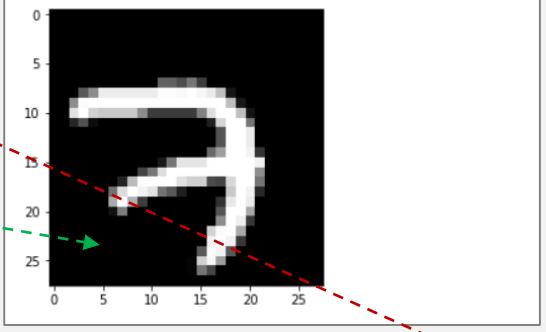

Convolutional layer 1
Convolutional layer 2
Input layer
Pooling layer 1
Pooling layer 2
Fully-connected layer

ex11_6.ipynb

MNIST image loading

```
import tensorflow as tf
import random
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data
tf.set_random_seed(777)  # reproducibility
tf.reset_default_graph()

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

print("mnist.train.labels[0] : ",  mnist.train.labels[0])

img = mnist.train.images[0].reshape(28,28)

plt.imshow(img, cmap='gray')
```

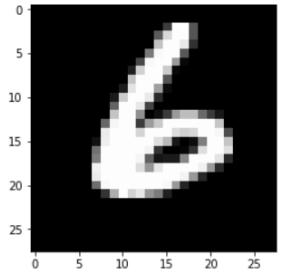
```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
mnist.train.labels[0] :  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
<matplotlib.image.AxesImage at 0x2a401c85518>
```



임의의 값으로 수정해보자.

```
img = mnist.train.images[213].reshape(28,28)
```

```
mnist.train.labels[0] :  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```



```
mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)

mnist.train.labels[0] :  7
```

MNIST Convolution layer

ex11_7.ipynb (1/2)

```
import tensorflow as tf
import random
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data

tf.set_random_seed(777)  # reproducibility
tf.reset_default_graph()

sess=tf.InteractiveSession()
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
img = mnist.train.images[5].reshape(28,28)
print("mnist.train.labels[5] : ",  mnist.train.labels[5])

img = img.reshape(-1,28,28,1)
# img.reshape(-1,28,28,1)
# -1 : 텐서플로가 알아서 조정, 28x28 이미지, 1 : color (black/white 1개)  → L1 입력 이미지

W1 = tf.Variable(tf.random_normal([3, 3, 1, 5], stddev=0.01))
# L1 입력 이미지 :  ImgIn shape=(?, 28, 28, 1)
# Filter 설정 : tf.random_normal([3, 3, 1, 5], stddev=0.01))
# 3x3 필터, 1 : 컬러가 1장이므로, 5 : 필터 개수(depth) → 5개의 convolution 값이 출력될것이다.

conv2d = tf.nn.conv2d(img, W1, strides=[1, 2, 2, 1], padding='SAME')
# strides=[1, 2, 2, 1], padding='SAME'
# strides는 (2x2) 이므로 절반으로 sampling된다. 만약, strides가 (1x1) 이면  padding='SAME' 이므로 입력이미지와 동일하게 sampling 된다.
# convolution 결과는 L1 입력 이미지의 크기의 절반 14x14 크기로 5장(필터 개수) 생성될 것이다.
```

MNIST Convolution layer

ex11_7.ipynb (2/2)

```
sess.run(tf.global_variables_initializer())
conv2d_img = conv2d.eval()
print(conv2d_img)
# convolution 결과가 5장(필터개수)씩 묶여진 형태로 14x14 만큼 출력된다.
```

```
conv2d_img = np.swapaxes(conv2d_img, 0, 3)
print("● After swapping ")
print(conv2d_img)

# convolution 결과가 5장(필터개수)씩 묶여져서 14x14 형태로 있으므로 swapping 한다.
# swapping하면 14 행으로 된 convolution 결과가 14장으로 5번 반복되어 생성된다.
# 아래와 같이 conv2d_img.shape 을 확인하면 (5, 14, 14, 1) 되어있다.
print("● conv2d_img.shape : ", conv2d_img.shape)
```

```
for i, one_img in enumerate(conv2d_img):
    print("i: ", i)

    plt.subplot(1,5,i+1)
    plt.imshow(one_img.reshape(14,14), cmap='gray')
```

코딩 추가해서 확인

```
print("● one_img shape \n",one_img.shape)
print("● one_img \n",one_img)
print("● one_img (14 x 14) reshape \n",one_img.reshape(14,14))
```

MNIST Convolution layer

ex11_7.ipynb (Notes)

1

```
mnist.train.labels[5] :  [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
[[[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]]
.
.
.
```

```
.
.
.
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]
   [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
     0.00000000e+00]]]]
```

MNIST Convolution layer

ex11_7.ipynb (Notes)

2

● After swapping

```
[[[[ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]]

  [[ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]]
 .
 .
 .
```

```
 .
 .
 .
  [[ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 6.59581600e-03]
   [ 1.31419748e-02]
   [ 1.01194177e-02]
   [ 4.83111897e-03]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]]

  [[ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]
   [ 0.00000000e+00]]]]
```

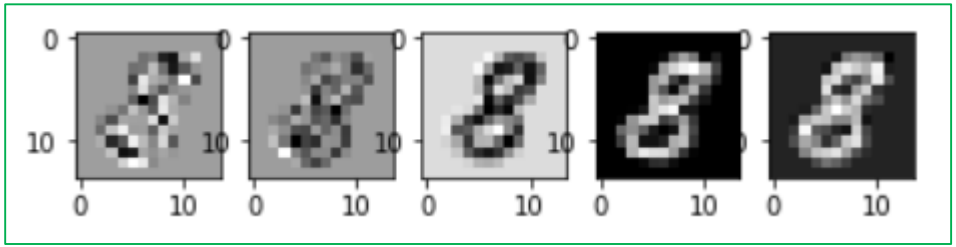
● conv2d_img.shape : (5, 14, 14, 1)

ex11_7.ipynb (Notes)

MNIST Convolution layer

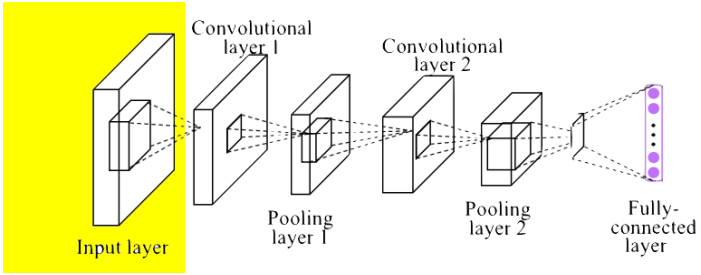
3

```
i:  0
i:  1
i:  2
i:  3
i:  4
```



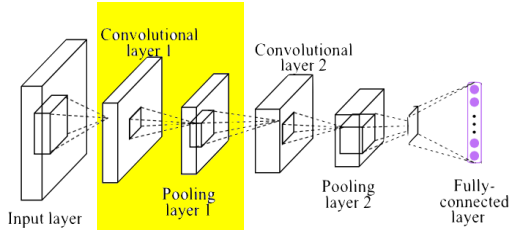
MNIST image loading

```
import tensorflow as tf
import random
import numpy as np

import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input_data

tf.set_random_seed(777)  # reproducibility
tf.reset_default_graph()
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
# Check out https://www.tensorflow.org/get_started/mnist/beginners for
# more information about the mnist dataset

# hyper parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

# input place holders
X = tf.placeholder(tf.float32, [None, 784])  # 28x28=784

X_img = tf.reshape(X, [-1, 28, 28, 1])   # img 28x28x1 (black/white)
# tf.reshape(-1,28,28,1)
# -1 : 텐서플로가 알아서 조정, 28x28 이미지, 1 : color (black/white 1개) → X_img

Y = tf.placeholder(tf.float32, [None, 10]) # 0~9까지 10개의 숫자가 출력되므로… → [None, 10]
```



Conv layer 1

ex11_8.ipynb (2/6)



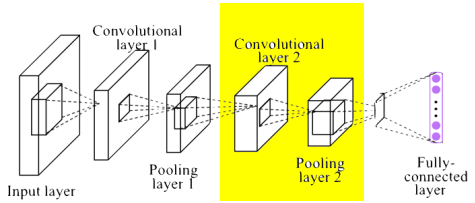

```
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
# 입력 이미지 : image shape=(?, 28, 28, 1)
# Filter 설정 : tf.random_normal([3, 3, 1, 32], stddev=0.01))
# 3x3 필터, 1 : 컬러가 1장이므로, 32 : 필터 개수(depth) → 32개의 convolution 값이 출력될 것이다.

L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
# strides=[1, 1, 1, 1], padding='SAME'
# strides는 (1x1) 이므로 입력 이미지와 동일하게 sampling된다.
# padding='SAME' 이므로... 입력 이미지의 크기와 동일하게 28x28 형태의 convolution 결과가 32장(필터 개수) 생성될 것이다.

L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
#  max_pool 에서 사용되는 kernel size [1, 2, 2, 1]은 max_pooling시 (2x2) windows로  이동하면서 큰 값을 추출한다.
#  즉, [1,2,2,1]이라는 뜻은 2칸씩 이동하면서 출력 결과를 1개 만들어 낸다는 것으로 4개의 데이터 중에서 가장 큰 값 1개를 반환하는 역할을 한다.

# strides=[1, 2, 2, 1], padding= 'SAME'
# strides는 (2x2) 이므로 절반으로 sampling된다.
# padding='SAME' 이므로... 입력 이미지 tf.nn.relu(L1) 크기의 절반 14x14 형태의 convolution 결과가 32장(필터 개수) 생성될 것이다.
```

```
Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
```

Conv layer 2

ex11_8.ipynb (3/6)

```
# L1 입력 이미지 : ImgIn shape=(?, 14, 14, 32)

W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
# Filter 설정 : tf.random_normal([3, 3, 32, 64], stddev=0.01))
# 3x3 필터, 32 : 입력단의 depth 와 동일하게, 64 : 필터 개수(depth) → 64장의 convolution 값이 출력될것이다.
```

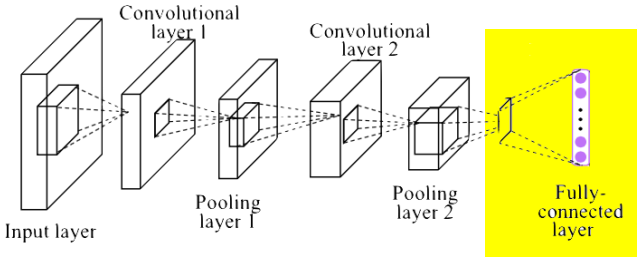



```
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
# strides=[1, 1, 1, 1], padding='SAME'
# strides는 (1x1) 이므로 입력 이미지와 동일하게 sampling 된다.
# padding='SAME' 이므로... L1 입력 이미지의 크기와 동일하게 14x14 형태의 convolution 결과가 64장(필터 개수) 생성될 것이다.


L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
#  max_pool 에서 사용되는 kernel size [1, 2, 2, 1]는 max_pooling시 (2x2) windows로  이동하면서 큰 값을 추출한다.
#  즉, [1,2,2,1]이라는 뜻은 2칸씩 이동하면서 출력 결과를 1개 만들어 낸다는 것으로 4개의 데이터 중에서 가장 큰 값 1개를 반환하는 역할을 한다.

# strides=[1, 2, 2, 1], padding='SAME'
# strides는 (2x2) 이므로 절반으로 sampling 된다.
# padding='SAME' 이므로...  입력 이미지 tf.nn.relu(L1) 크기의 절반 7x7 형태의 convolution 결과가 64장(필터 개수) 생성될 것이다.

L2_flat = tf.reshape(L2, [-1, 7 * 7 * 64] )
# Fully Connected 하기 위해서 행은 자동으로 …
# 열은 7*7*64 → 즉, 3136으로 reshape 한다.
```

```
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
```

Fully Connected (FC, Dense) layer

ex11_8.ipynb (4/6)




```
# Final FC 7x7x64 inputs → 10 outputs
# L2 의 입력 shape은 ( -1, 7*7*64 ) 에 해당한다.  즉,  L2  입력 이미지 : ( ? x 3136)  이다.

W3 = tf.get_variable("W3", shape=[7 * 7 * 64, 10], initializer=tf.contrib.layers.xavier_initializer())
# shape=[ 7 * 7 * 64 , 10]
# 앞 단의 열에 해당하는 7*7*64 값이 현재의 행에 입력되어야 한다.
# 10의 의미는 output 을 의미하며 출력할 값이 0~9… 10개의 숫자이므로…

b = tf.Variable(tf.random_normal([10]))  # bias
logits = tf.matmul(L2_flat, W3) + b

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y))  # logits : hypothesis 에 해당,  Y : True value 에 해당
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

Training and Evaluation

ex11_8.ipynb (5/6)

```
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())
# train my model
print('Learning started. It takes sometime.')

for epoch in range(training_epochs):   # training_epochs = 15
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)  # batch_size = 100

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')
```

```
Learning started. It takes sometime.
Epoch: 0001 cost = 0.345581654
Epoch: 0002 cost = 0.091730375
Epoch: 0003 cost = 0.068179509
Epoch: 0004 cost = 0.056329947
Epoch: 0005 cost = 0.046911515
Epoch: 0006 cost = 0.041009831
Epoch: 0007 cost = 0.036596036
Epoch: 0008 cost = 0.032737206
Epoch: 0009 cost = 0.027924412
Epoch: 0010 cost = 0.024786621
Epoch: 0011 cost = 0.022212983
Epoch: 0012 cost = 0.020372173
Epoch: 0013 cost = 0.016959655
Epoch: 0014 cost = 0.015577860
Epoch: 0015 cost = 0.013503993
Learning Finished!
```

Training and Evaluation → Visualizartion

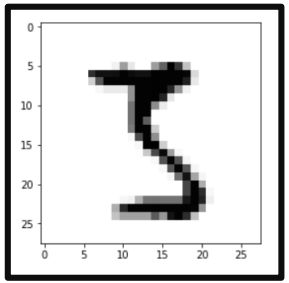
```
# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

print('Accuracy:', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels})) # test 이미지를 모두 테스트함
```

Accuracy: 0.9891

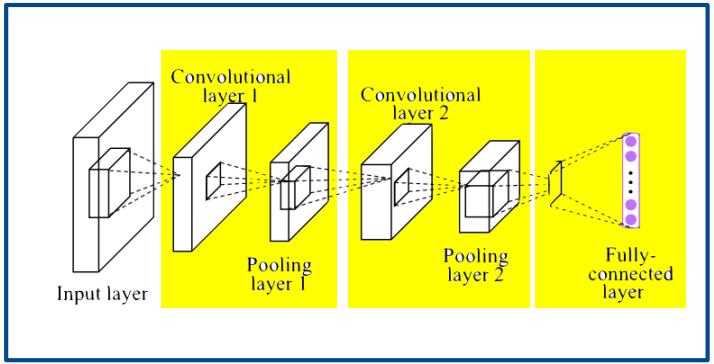
```
# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)  # test 이미지 중에서 하나를 랜덤하게 선택
print("Label: ", sess.run(tf.argmax(mnist.test.labels[r:r + 1], 1))) # one-hot 코드에서 argmax를 실행해서 해당 행에서 1이 있는 인덱스 위치를 출력
print("Prediction: ", sess.run(tf.argmax(logits, 1), feed_dict={X: mnist.test.images[r:r + 1] }))
```

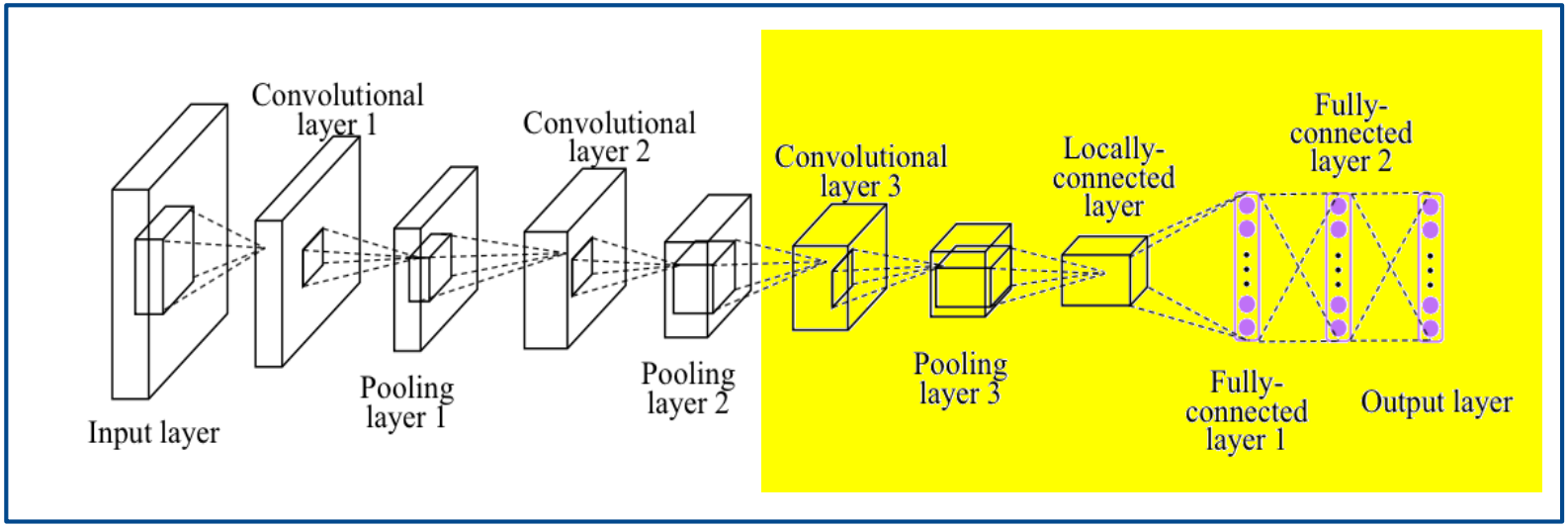
Label: [3]
Prediction: [3]

```
# Visualizartion
plt.imshow(mnist.test.images[r:r + 1].reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
```



Deep CNN


Convolutional layer 1
Convolutional layer 2
Input layer
Pooling layer 1
Pooling layer 2
Fully-connected layer


Convolutional layer 1
Convolutional layer 2
Convolutional layer 3
Locally-connected layer
Fully-connected layer 2
Input layer
Pooling layer 1
Pooling layer 2
Pooling layer 3
Fully-connected layer 1
Output layer

ex11_9.ipynb (1/8)

Deep CNN

```
# MNIST and Deep learning CNN
import tensorflow as tf
import random
# import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input_data

tf.set_random_seed(777)  # reproducibility
tf.reset_default_graph()

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

# hyper parameters
learning_rate = 0.001
training_epochs = 15
batch_size = 100

# dropout (keep_prob) rate  0.7~0.5 on training, but should be 1 for testing
keep_prob = tf.placeholder(tf.float32)

# input place holders
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1])   # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 10])
```

ex11_9.ipynb (2/8)

Deep CNN

Conv layer 1

```
# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv    -> (?, 28, 28, 32)
#    Pool    -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
```

```
Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
Tensor("dropout/mul:0", shape=(?, 14, 14, 32), dtype=float32)
```

ex11_9.ipynb (3/8)

Deep CNN

Conv layer 2

```
# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv  → (?, 14, 14, 64)
#    Pool    → (?, 7, 7, 64)
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.nn.dropout(L2, keep_prob=keep_prob)
```

```
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("dropout_1/mul:0", shape=(?, 7, 7, 64), dtype=float32)
```

ex11_9.ipynb (4/8)

Deep CNN

Conv layer 3

```
# L3 ImgIn shape=(?, 7, 7, 64)
W3 = tf.Variable(tf.random_normal([3, 3, 64, 128], stddev=0.01))
#    Conv     →  (?, 7, 7, 128)
#    Pool     →  (?, 4, 4, 128)
#    Reshape  →  (?, 4 * 4 * 128)
#    Flatten them for FC
L3 = tf.nn.conv2d(L2, W3, strides=[1, 1, 1, 1], padding='SAME')
L3 = tf.nn.relu(L3)
L3 = tf.nn.max_pool(L3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L3 = tf.nn.dropout(L3, keep_prob=keep_prob)
L3_flat = tf.reshape(L3, [-1, 128 * 4 * 4])
```

```
Tensor("Conv2D_2:0", shape=(?, 7, 7, 128), dtype=float32)
Tensor("Relu_2:0", shape=(?, 7, 7, 128), dtype=float32)
Tensor("MaxPool_2:0", shape=(?, 4, 4, 128), dtype=float32)
Tensor("dropout_2/mul:0", shape=(?, 4, 4, 128), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 2048), dtype=float32)
```

ex11_9.ipynb (5/8)

Deep CNN

Fully Connected layer_1

```
# L4 FC 128x4x4 inputs → 625 outputs

W4 = tf.get_variable("W4", shape=[ 128 * 4 * 4, 625 ], initializer=tf.contrib.layers.xavier_initializer())
b4 = tf.Variable(tf.random_normal( [625] ))
L4 = tf.nn.relu(tf.matmul(L3_flat, W4) + b4)
L4 = tf.nn.dropout(L4, keep_prob=keep_prob)
```

```
Tensor("Relu_3:0", shape=(?, 625), dtype=float32)
Tensor("dropout_3/mul:0", shape=(?, 625), dtype=float32)
```

ex11_9.ipynb (6/8)

Deep CNN

Fully Connected layer_2

```
# L5 Final FC 625 inputs → 10 outputs
W5 = tf.get_variable("W5", shape=[ 625 , 10 ], initializer=tf.contrib.layers.xavier_initializer())
b5 = tf.Variable(tf.random_normal( [10] ))
logits = tf.matmul(L4, W5) + b5
```

```
Tensor("add_1:0", shape=(?, 10), dtype=float32)
```

Training and Evaluation

ex11_9.ipynb (7/8)

Deep CNN

```
# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model (-- learning -- )
print('Learning started. It takes sometime.')
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys, keep_prob: 0.7}
        c, _ = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch

    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')
```

ex11_9.ipynb (8/8)

Training and Evaluation → Visualizartion

Deep CNN

```
# Test model and check accuracy

correct_prediction = tf.equal(tf.argmax(logits, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy:', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels, keep_prob: 1}))

# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label: ", sess.run(tf.argmax(mnist.test.labels[r:r + 1], 1)))
print("Prediction: ", sess.run(tf.argmax(logits, 1), feed_dict={X: mnist.test.images[r:r + 1], keep_prob: 1}))

plt.imshow(mnist.test.images[r:r + 1].reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
```

```
Learning started. It takes sometime.
Epoch: 0001 cost = 0.393510631
Epoch: 0002 cost = 0.097589802
Epoch: 0003 cost = 0.072168301
Epoch: 0004 cost = 0.058600905
Epoch: 0005 cost = 0.050884618
Epoch: 0006 cost = 0.046030076
Epoch: 0007 cost = 0.039642484
Epoch: 0008 cost = 0.037777071
Epoch: 0009 cost = 0.035090087
Epoch: 0010 cost = 0.031555091
Epoch: 0011 cost = 0.031608125
Epoch: 0012 cost = 0.027512437
Epoch: 0013 cost = 0.026881762
Epoch: 0014 cost = 0.026680036
Epoch: 0015 cost = 0.024344759
Learning Finished!
Accuracy: 0.9915
Label:  [0]
Prediction:  [0]
```