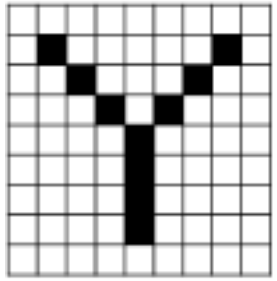
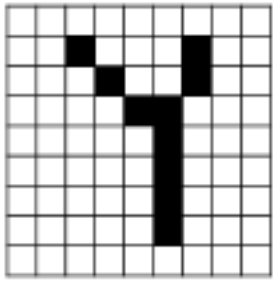
CNN_합성곱 신경망

2020019252 김나현

Convolutional
Neural
Network

다층 퍼셉트론



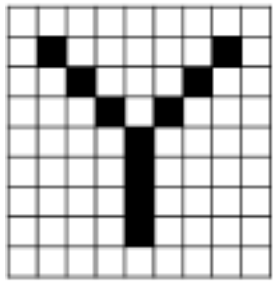


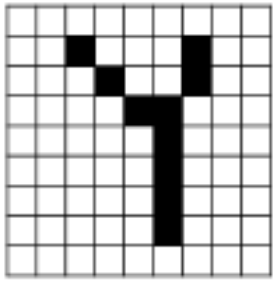
형 변환

- 2차원 평면에서 가지고 있던
  지역적인 정보를 잃은 채로 학습 시작

- 픽셀 하나하나의 변화에 민감

생김새 정보를 학습하는 CNN



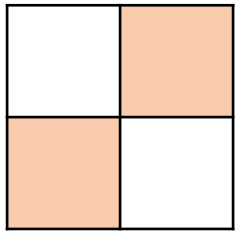


형 변환

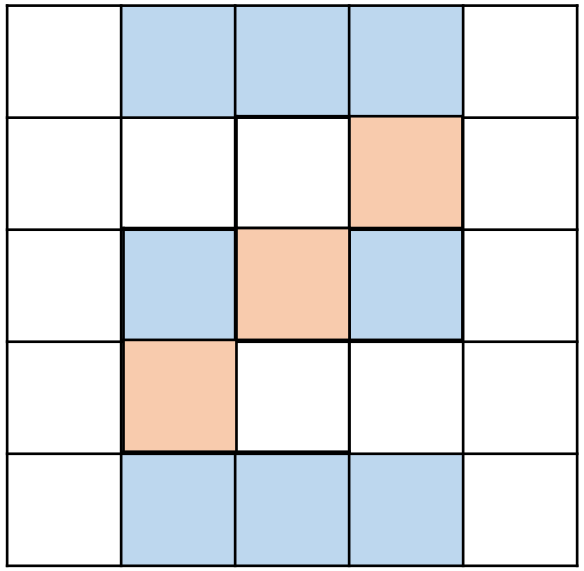
형변환 없이 입력 데이터
그대로 처리!

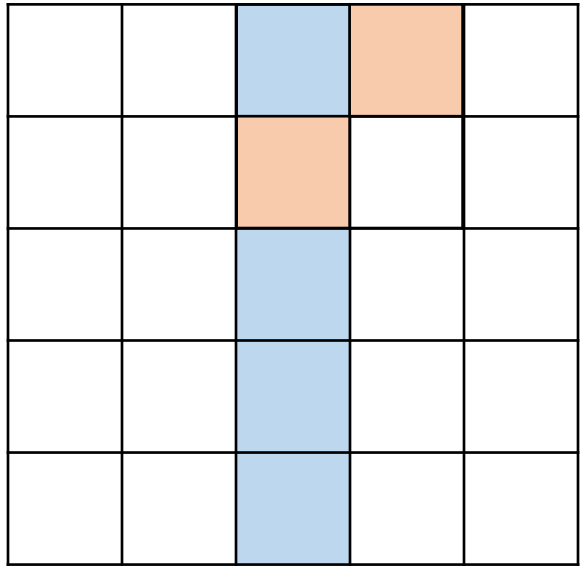
THEN... HOW?

합성곱



필터(커널) 특징을 추출하기 위한 네모 상자



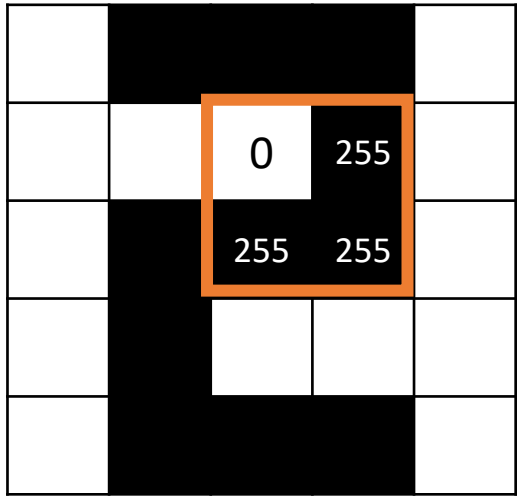


필터의 크기,
이동 범위(스트라이드)
설정 가능

글자를 구분하기 위해서는

여러가지 필터가 필요!

합성곱

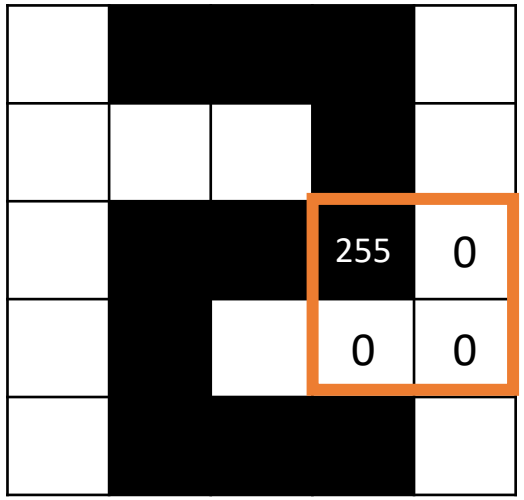


X

0	255
255	0

+ 3

0 * 0 + 255 * 255 + 255 * 255 + 0 * 255
= 130050 +3



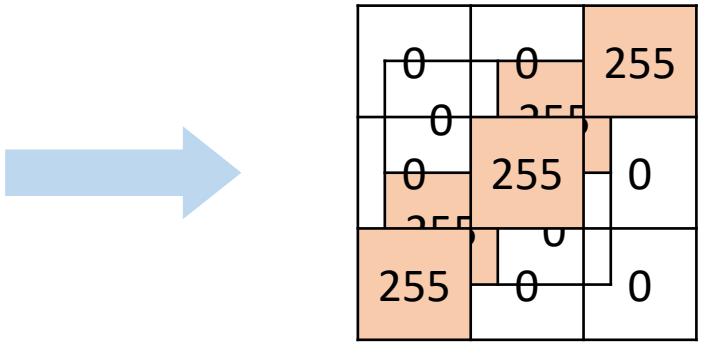
X

0	255
255	0

0 * 255 + 255 * 0 + 255 * 0 + 0 * 0
= 0

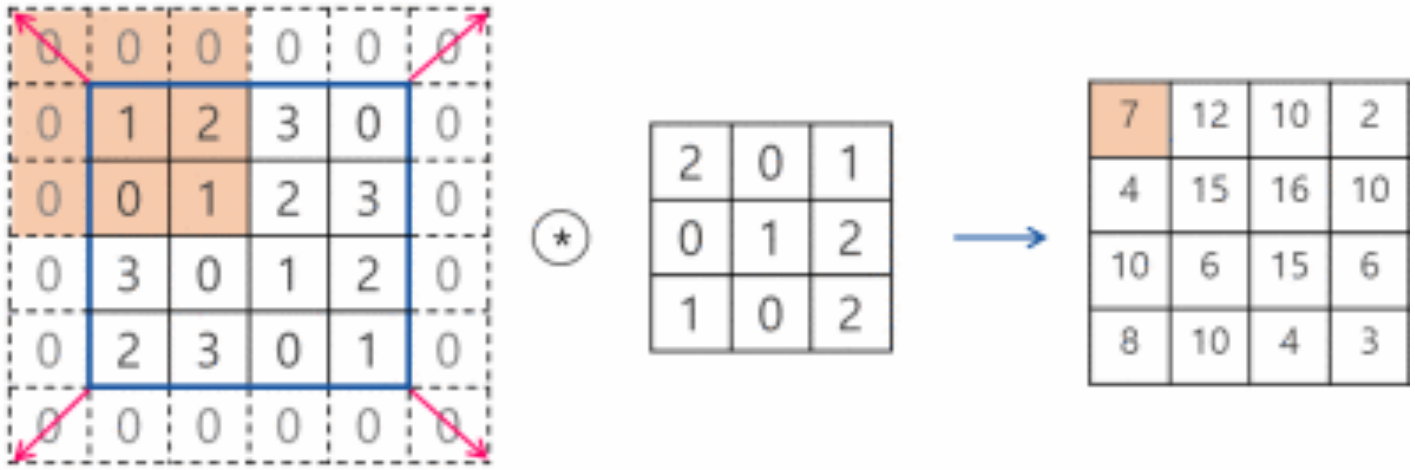
ZERO-PADDING

0	0	0	0	0	0	0
0	**0**	**255**	**255**	**255**	**0**	0
0	**0**	**0**	**0**	255	**0**	0
0	**0**	255	255	255	**0**	0
0	**0**	255	0	0	**0**	0
0	**0**	**255**	**255**	**255**	**0**	0
0	0	0	0	0	0	0



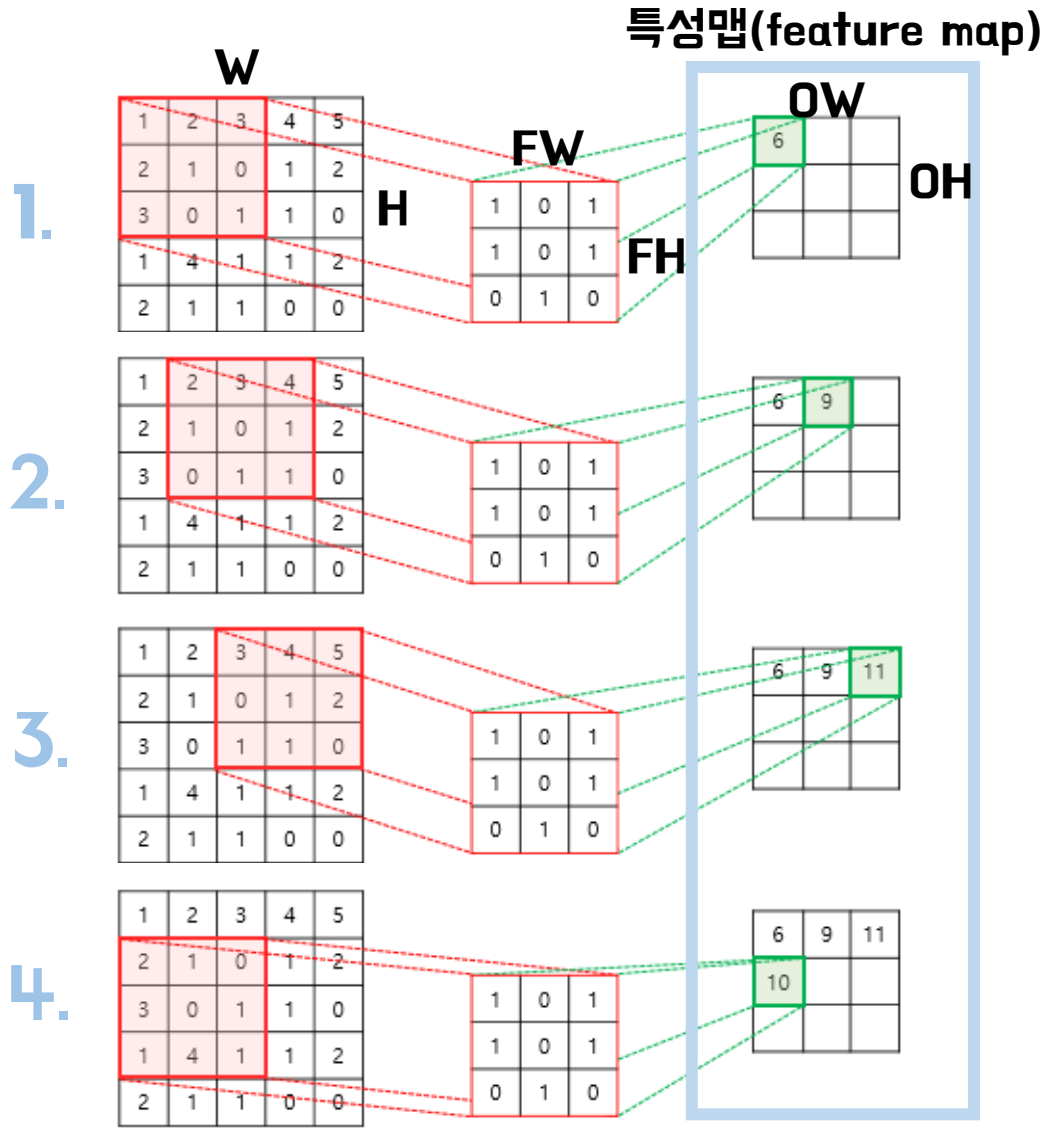
합성곱 할 공간이 더 많아짐
테두리 정보를 활용할 수 있게 됨

ZERO-PADDING



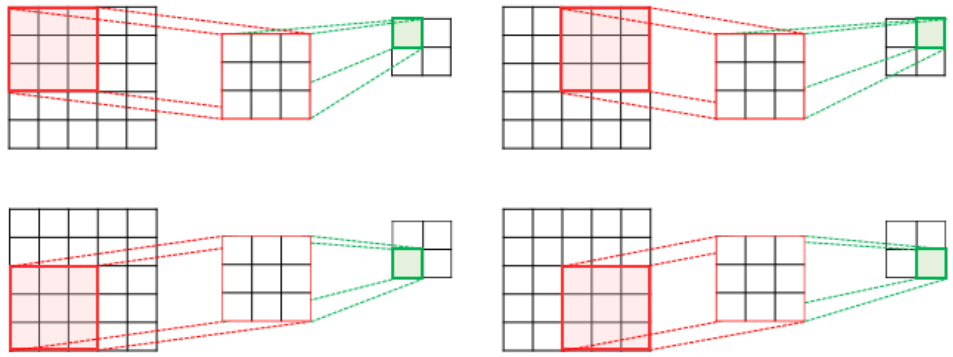
입력 데이터의 공간적 크기를 고정한 채로 다음 계층에 전달 가능
>> 정보 손실이 적어짐

합성곱

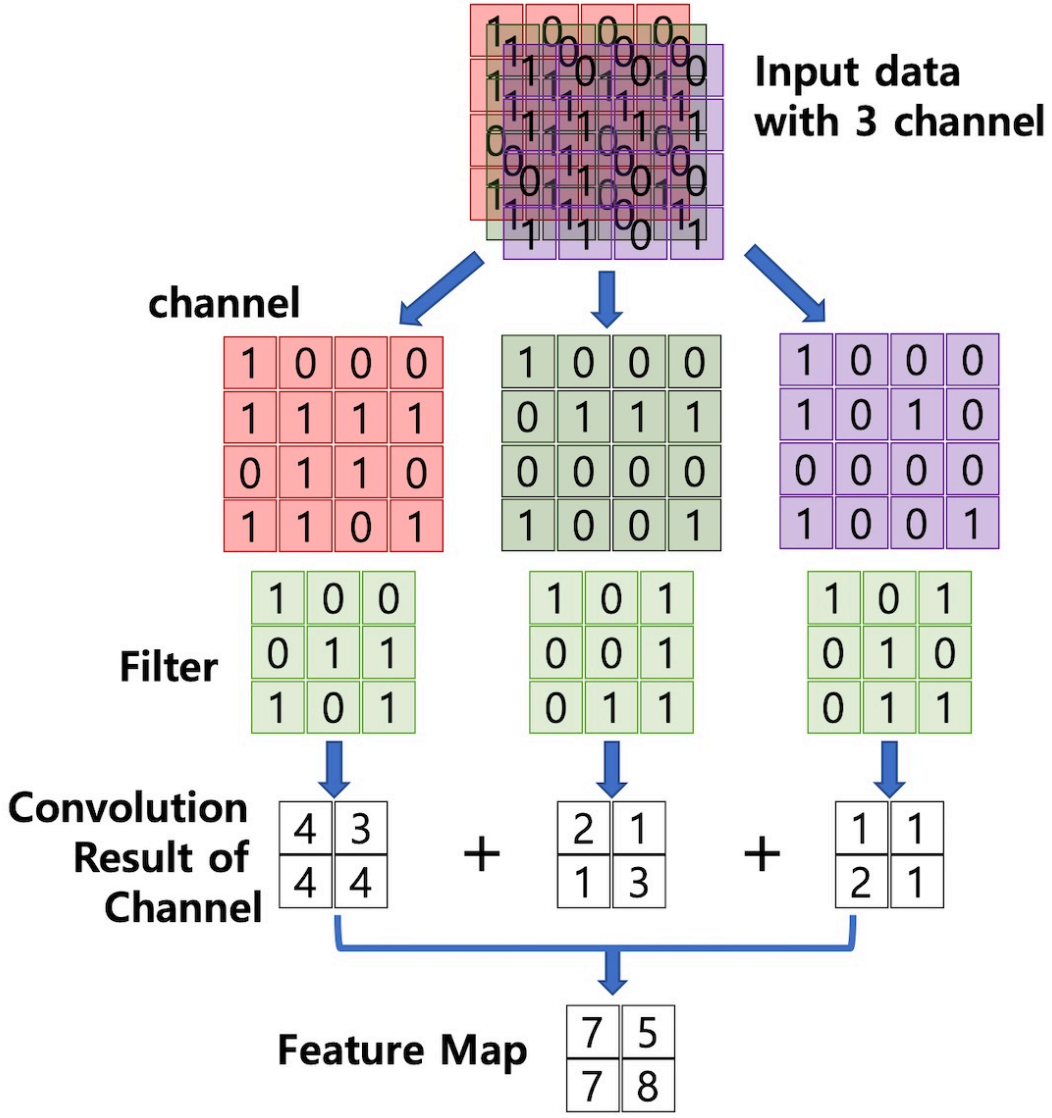


$$OW = (W + \cancel{2P} - FW)/\cancel{S} + 1$$

$$OH = (H + 2P - FH)/S + 1$$



채널

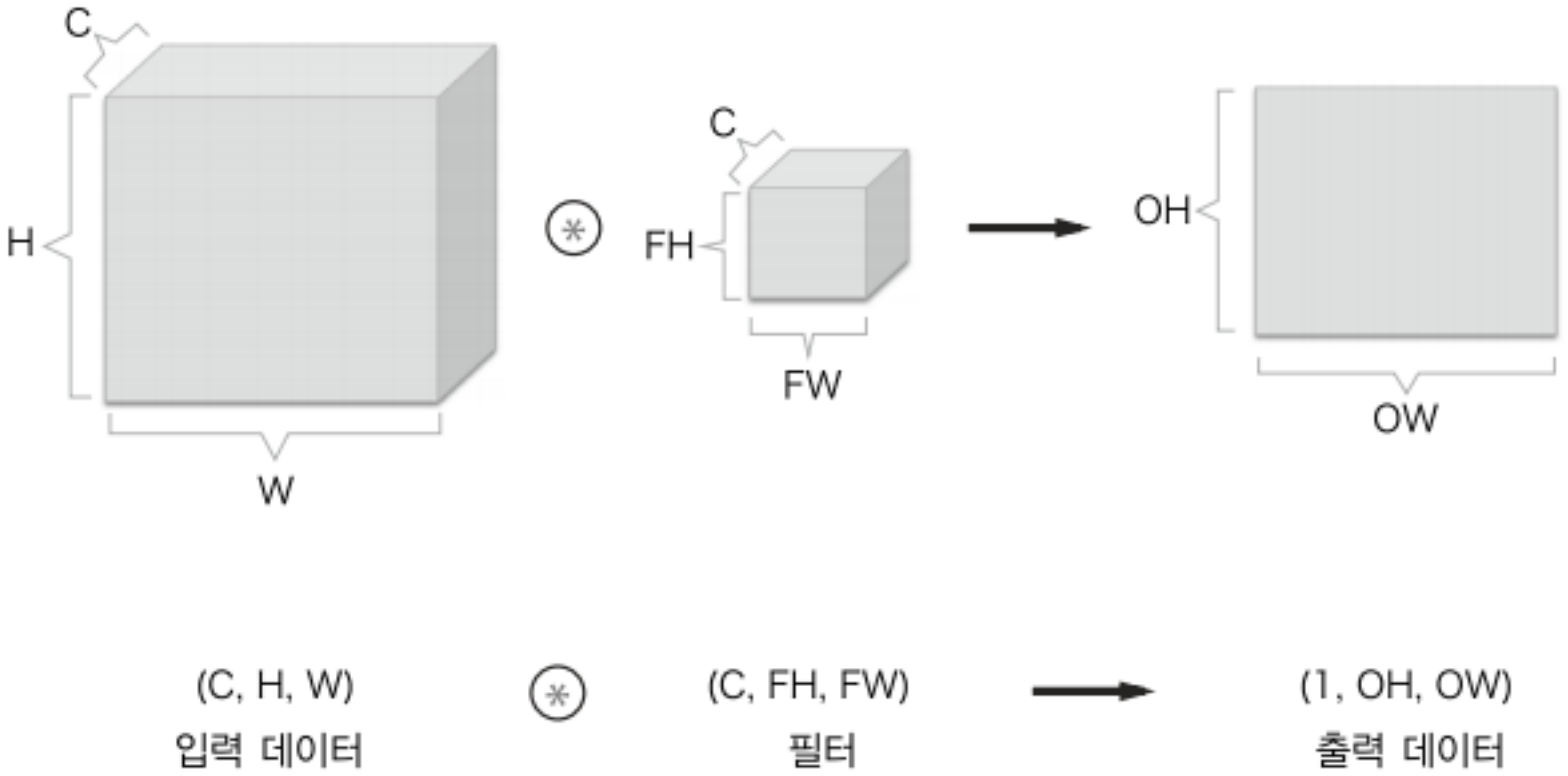


컬러 사진 :

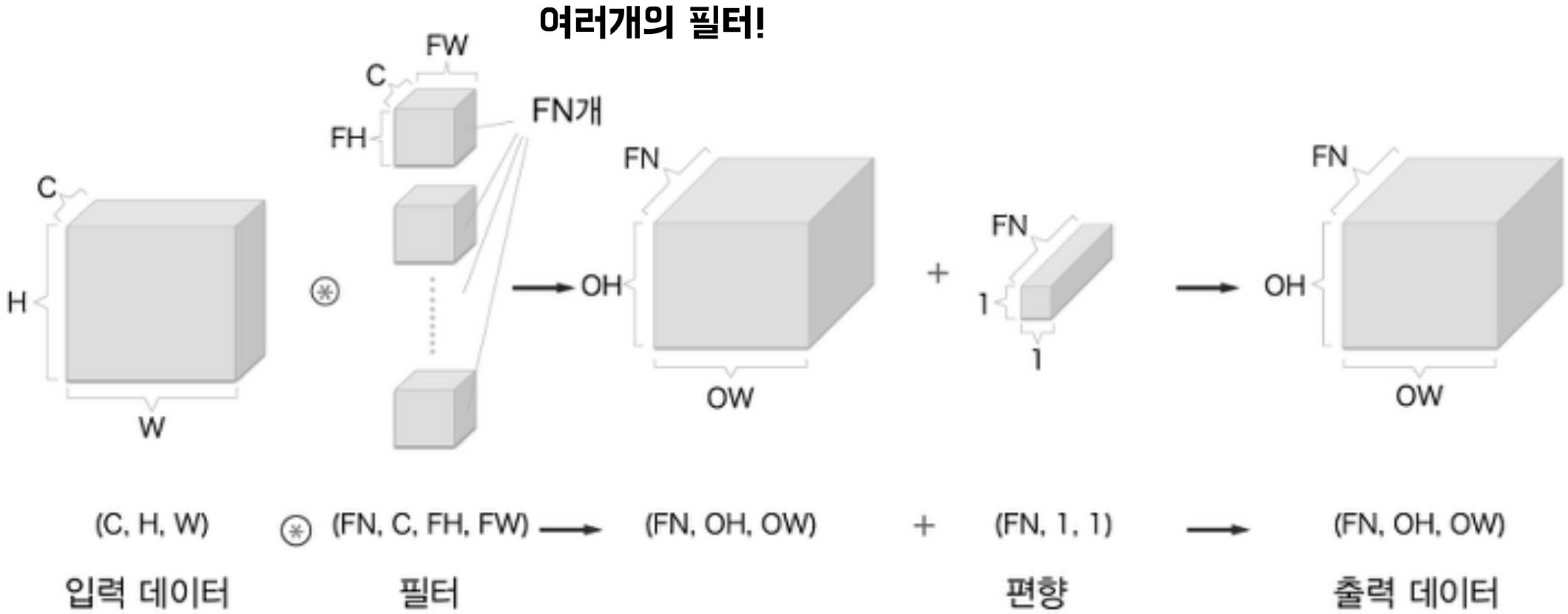
각 픽셀을 RGB 3개의
실수로 표현한 3차원 데이터

채널

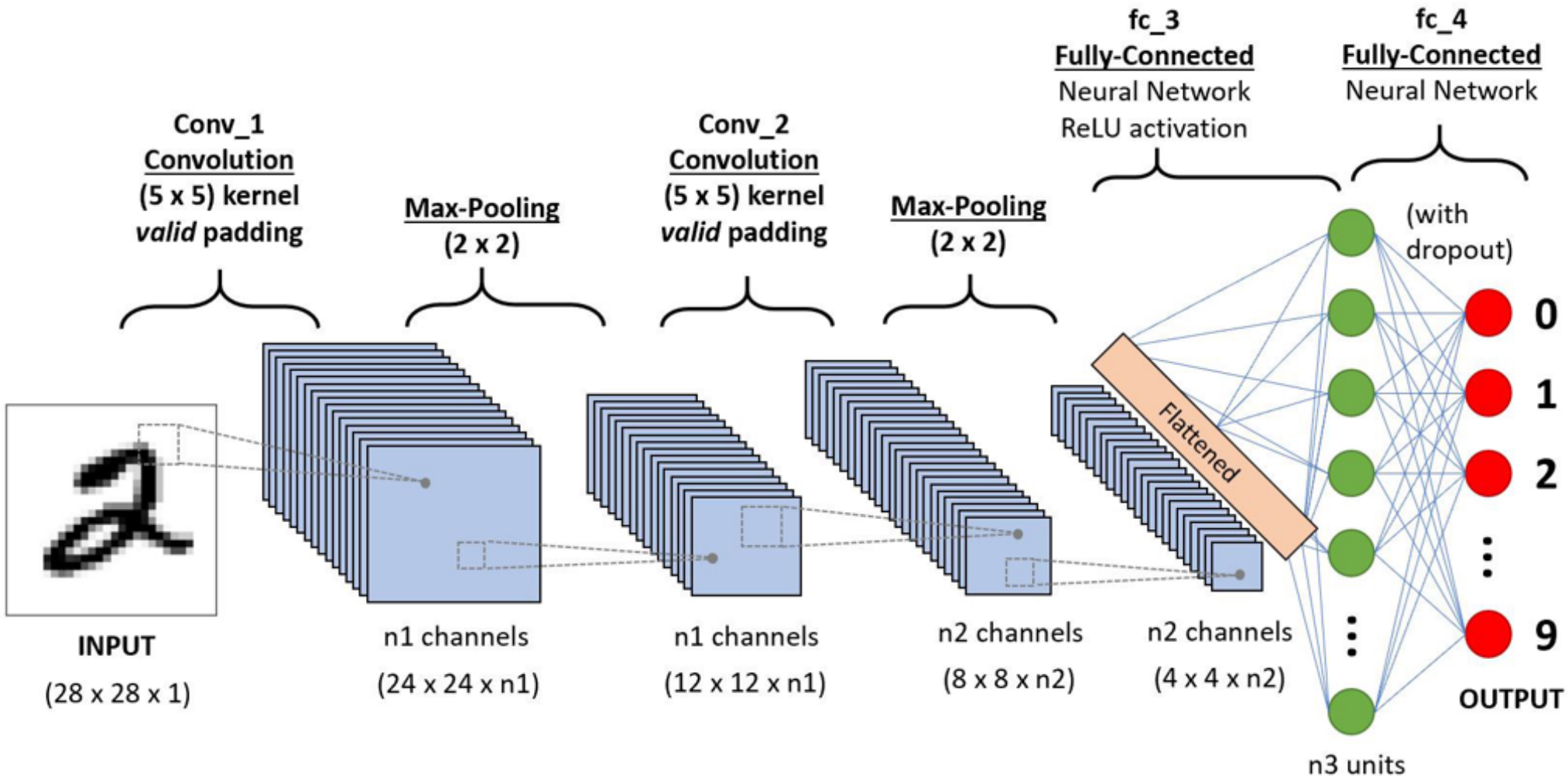
컬러 사진 (RGB) 3개의 채널



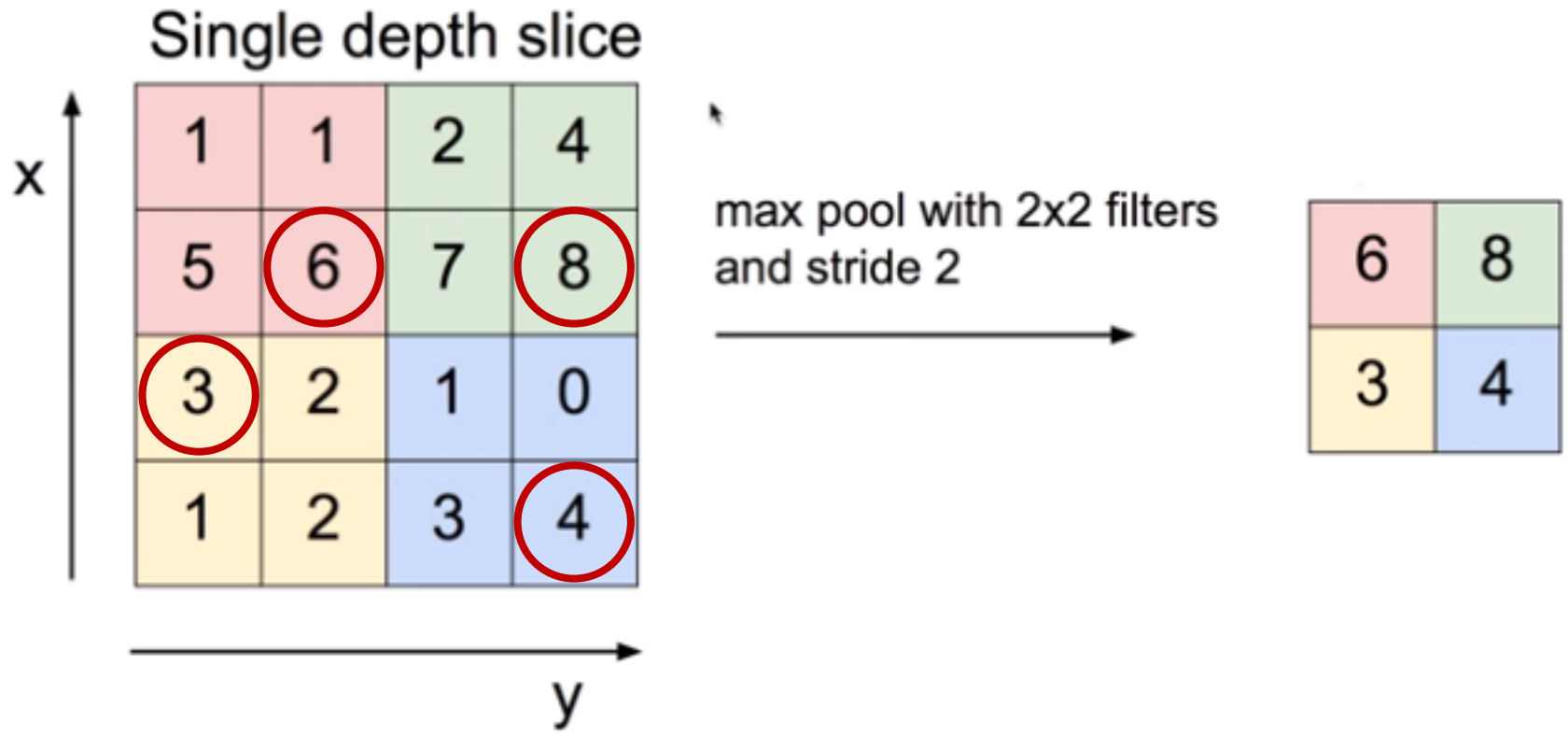
CNN 흐름


여러개의 필터!
FW
C
FH
FN개
C
H
W
OH
FN
OW
FN
1
1
+
OH
FN
OW
(C, H, W)
(FN, C, FH, FW)
(FN, OH, OW)
(FN, 1, 1)
(FN, OH, OW)
입력 데이터
필터
편향
출력 데이터

CNN 구조

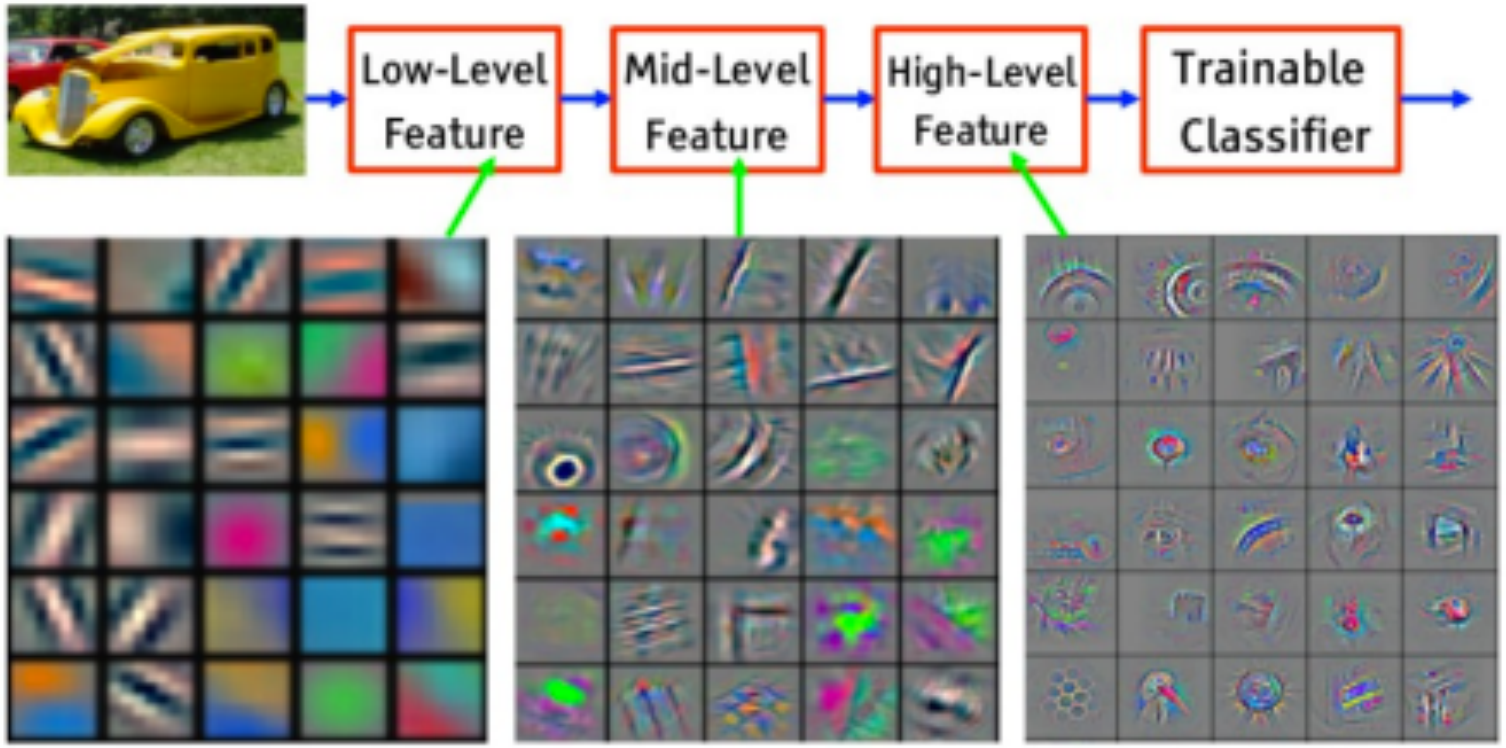

fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
(with
dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

MAX-POOLING

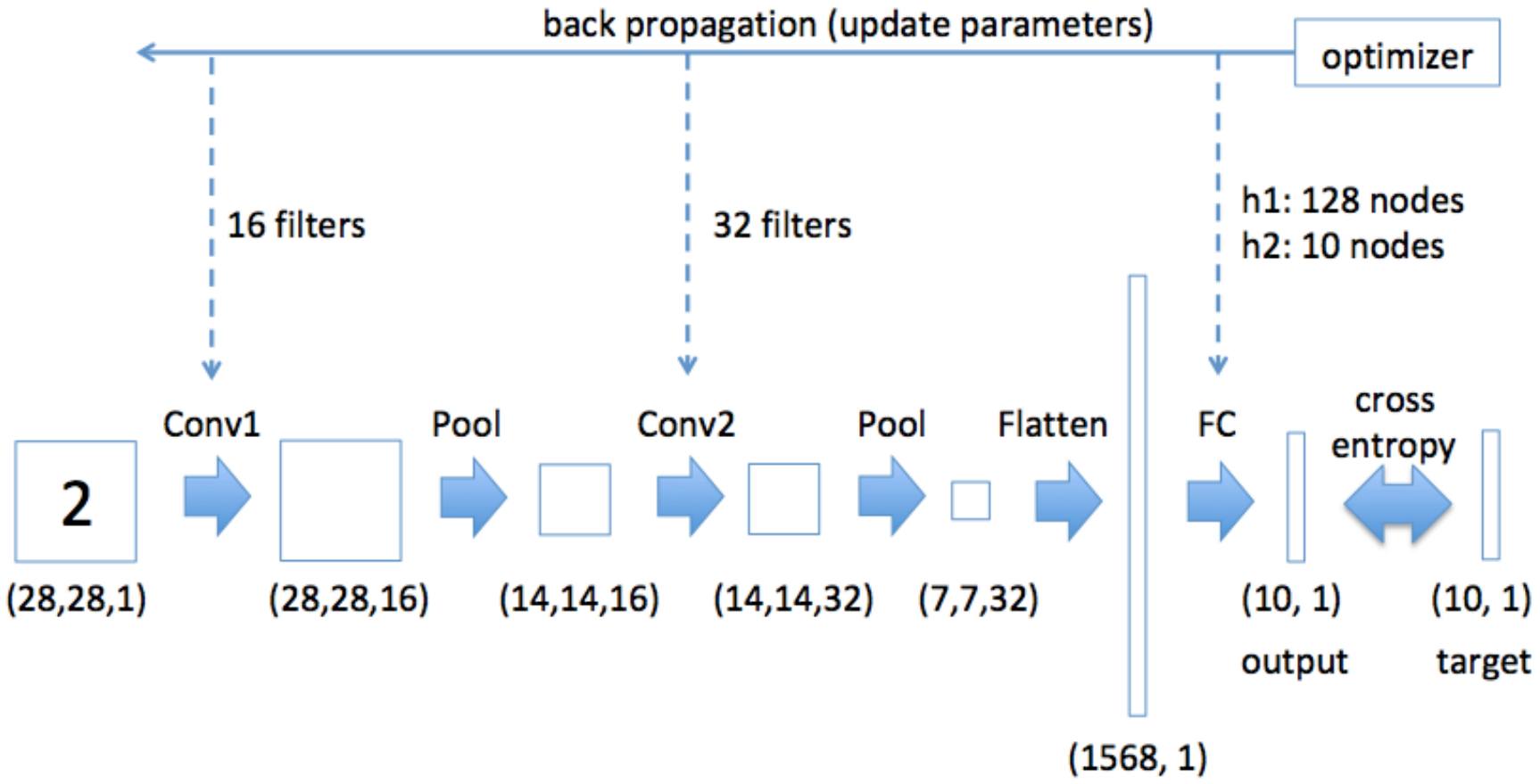

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

CNN 추출 정보

더 복잡하고 추상화된 정보가 추출됨



CNN 구현



코드 출처 : https://github.com/wikibook/machine-learning/blob/master/jupyter_notebook/6.1_CNN_MNIST_%EC%86%90%EA%B8%80%EC%94%A8_%EC%98%88%EC%B8%A1%EB%AA%A8%EB%8D%B8.ipynb

CNN 구현

```
from IPython.display import Image
Image(url="https://raw.githubusercontent.com/captainchargers/deeplearning/master/img/practice_cnn.png",
width=800, height=200)
```

```
import tensorflow as tf
```

데이터 획득

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

학습데이터에서 검증 데이터 분리하기

```
x_val = x_train[50000:60000]
x_train = x_train[0:50000]
y_val = y_train[50000:60000]
y_train = y_train[0:50000]
```

CNN 구현

데이터 확인

```
print(x_train.shape)
print(x_val.shape)
print(x_test.shape)
```

```
(50000, 28, 28)
(10000, 28, 28)
(10000, 28, 28)
```

```
print(x_train[0][8])
print(y_train[0:9])
```

```
[ 0 0 0 0 0 0 0 18 219 253 253 253 253 253 198 182 247 241 0 0 0 0 0 0 0 0 0 0]
[5 0 4 1 9 2 1 3 1]
```

CNN 구현

데이터 구조 변경

```
import numpy as np x_train = np.reshape(x_train, (50000,28,28,1))

x_val = np.reshape(x_val, (10000,28,28,1))
x_test = np.reshape(x_test, (10000,28,28,1))

print(x_train.shape)          (50000, 28, 28, 1)
print(x_test.shape)           (10000, 28, 28, 1)
```

CNN 구현

데이터 정규화

```
x_train = x_train.astype('float32’)
x_val = x_val.astype('float32’)
x_test = x_test.astype('float32’)

gray_scale = 255
x_train /= gray_scale
x_val /= gray_scale
x_test /= gray_scale
```

CNN 구현

one hot encoding

```
num_classes = 10

y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_val = tf.keras.utils.to_categorical(y_val, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
```

준비 끝!

CNN 구현

```
x = tf.placeholder(tf.float32, shape=[None, 28, 28, 1])
y_ = tf.placeholder(tf.float32, shape=[None, 10])
```

파라미터 초깃값 설정

```
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)
```

CNN 구현

```
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')
```

CNN 구현

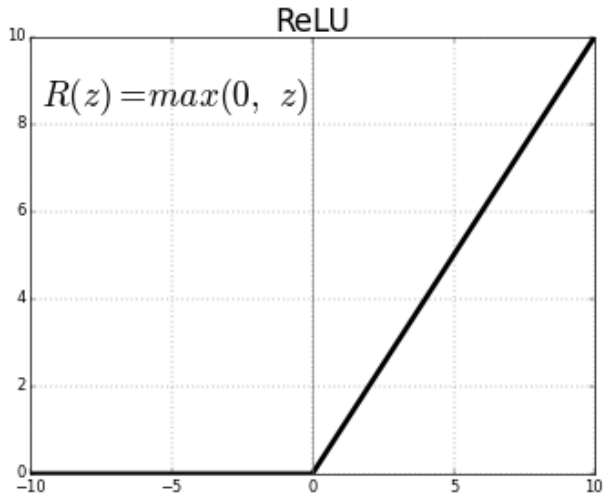
```
W_conv1 = weight_variable([5, 5, 1, 16])
b_conv1 = bias_variable([16])
```

활성화함수로 ReLU 사용

```
h_conv1 = tf.nn.relu(conv2d(x, W_conv1) + b_conv1)
```

맥스 풀링으로 맵의 크기 줄여줌

```
h_pool1 = max_pool_2x2(h_conv1)
```



CNN 구현

활성화함수로 ReLU 사용

```
W_conv2 = weight_variable([5, 5, 16, 32])
b_conv2 = bias_variable([32])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

CNN 구현

FC (Fully Connected Layer)

```
W_fc1 = weight_variable([7 * 7 * 32, 128])
b_fc1 = bias_variable([128])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*32])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)


W_fc2 = weight_variable([128, 10])
b_fc2 = bias_variable([10])
y_conv = tf.matmul(h_fc1, W_fc2) + b_fc2
```

CNN 구현

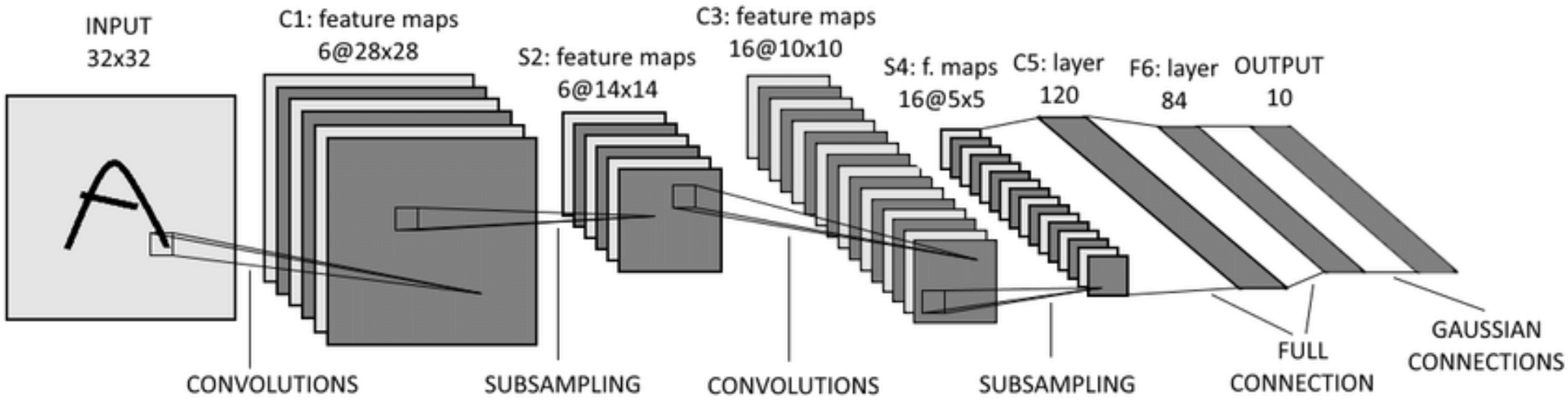
```
cross_entropy = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits_v2( labels=y_, logits=y_conv))

train_step = tf.train.AdamOptimizer(0.001).minimize(cross_entropy)
```

정확도 구하기

```
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

LeNet

손글씨 숫자를 인식하는 네트워크



THANK YOU