파이썬으로 배우는 딥러닝(Deep Learning)

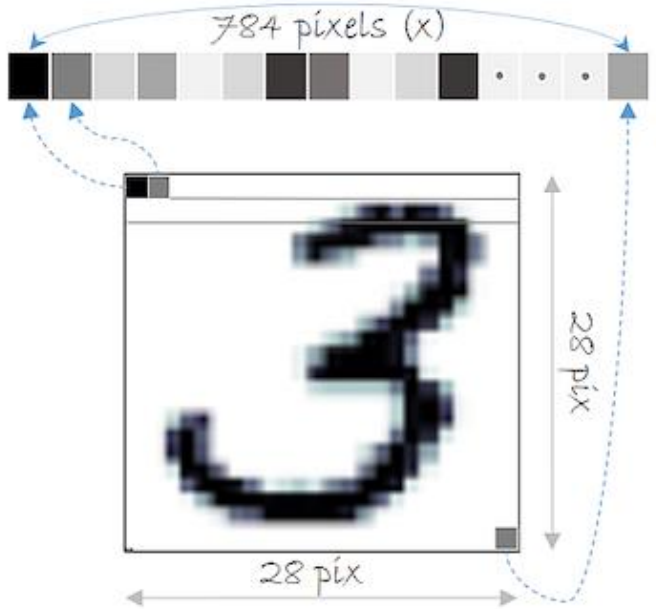
6회차 수업
이미지를 이해하는 신경망, CNN 따라 만들기

목 차

기존 DNN의 문제점

- 기존 DNN은 1차원 형태의 데이터를 사용함
- 이미지 데이터(2차원) : 1차원으로 표현할 경우 인접 영역 상관관계가 손실됨.

공간적 상관관계



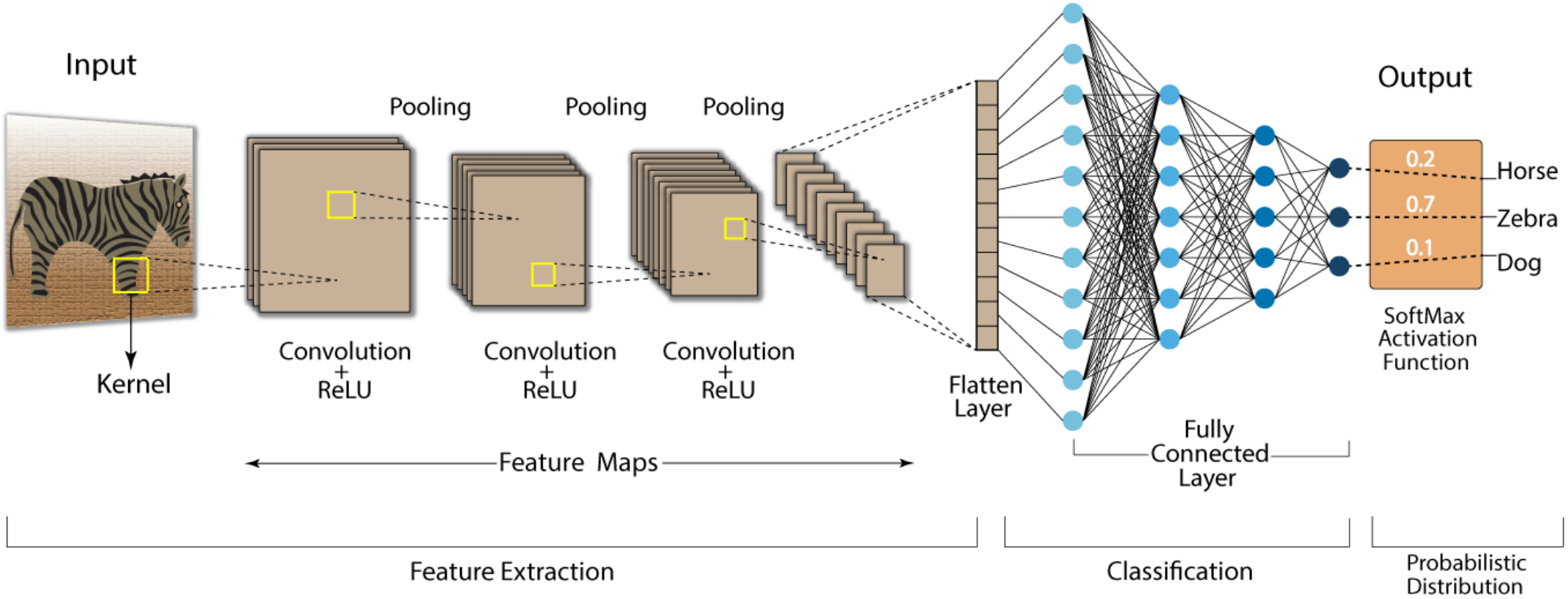
인접한 픽셀들이 모여 이미지의 각 부분을 표현하나,
한 줄로 늘어뜨린 1차원 데이터에서는 상관관계가 제거됨

Feature Invariance

1차원 Pixel array로는
2차원 이미지의 상관관계가 소실됨

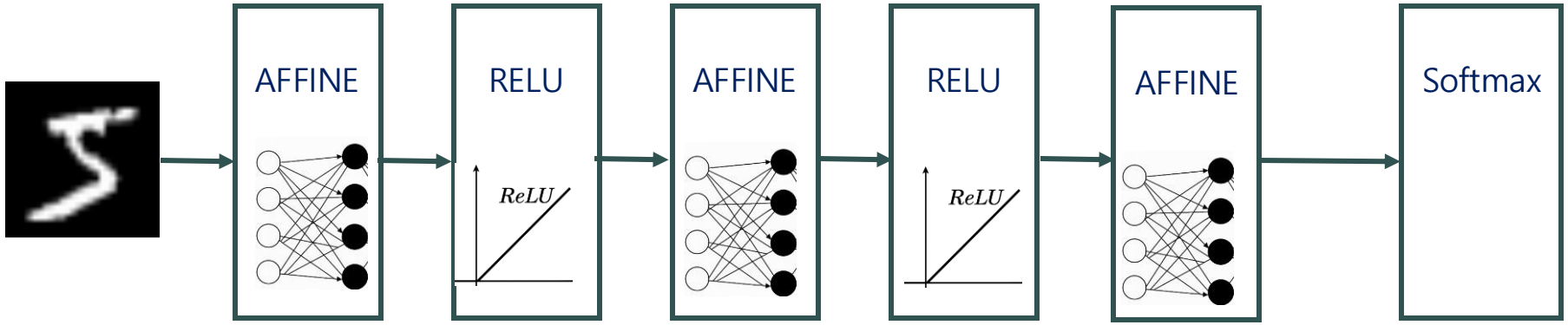
합성곱신경망(CNN) 전체구조(1)

- CNN : 이미지 데이터의 공간적 상관관계를 유지하며 이미지내 특징 추출 가능
- Convolution+Pooing+Flatten : 이미지의 공간적 상관관계를 유지하며 일렬로 배열함

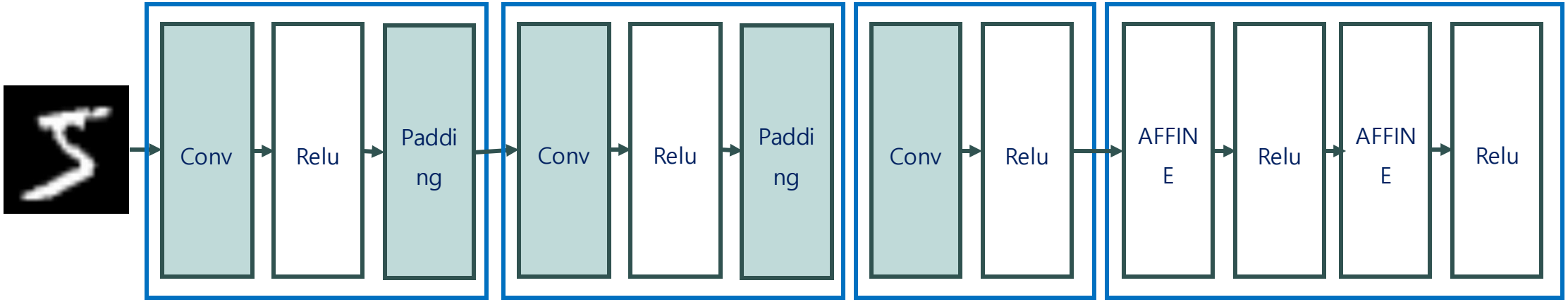


합성곱신경망(CNN) 전체구조(2)

- 기존 FCL(Fully Connected Layer)으로 이뤄진 네트워크의 예



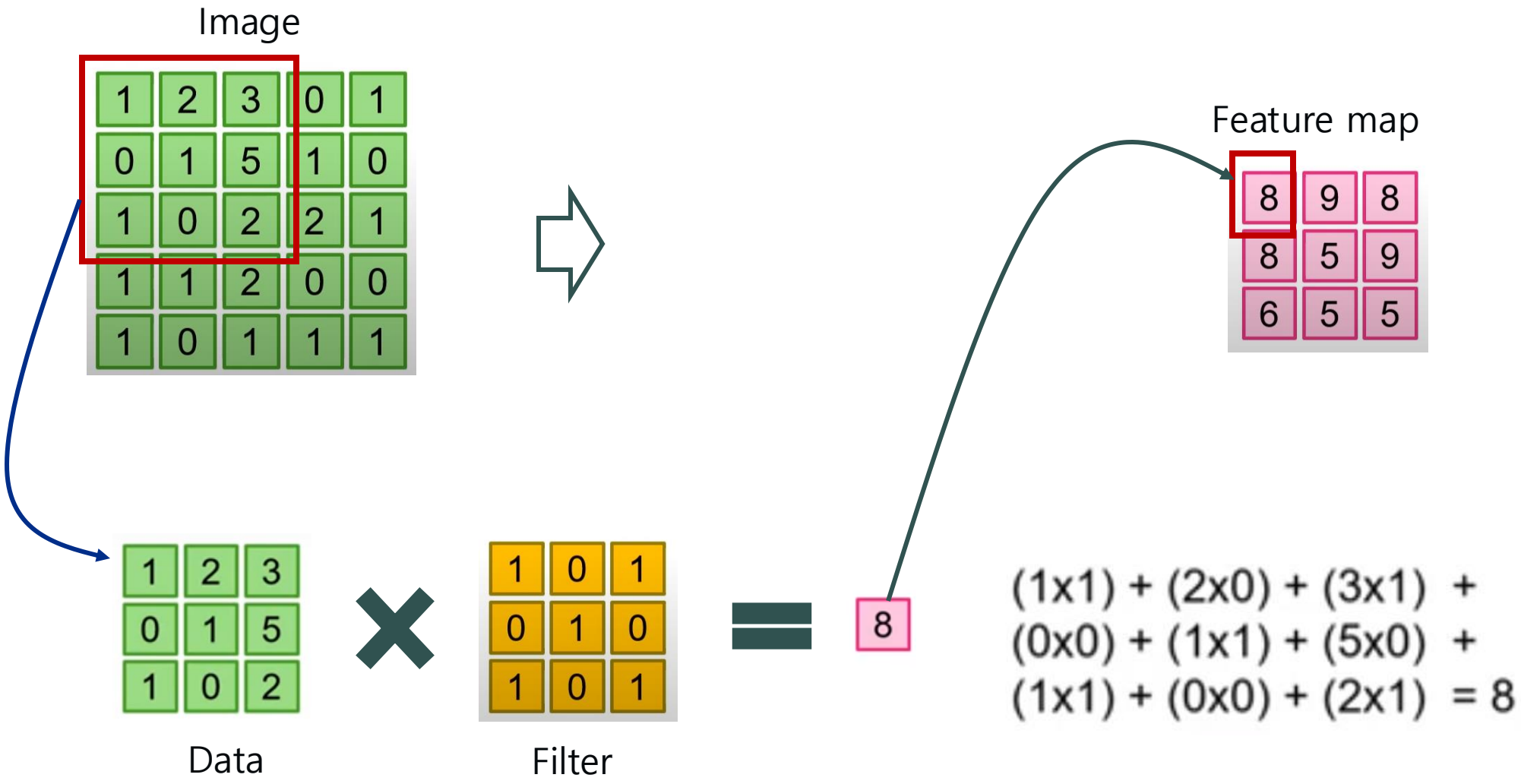
- CNN으로 이뤄진 네트워크의 예 : Convolution 계층과 Pooling 계층이 새로 추가됨.



목 차

Convolution(합성곱) 연산(1)

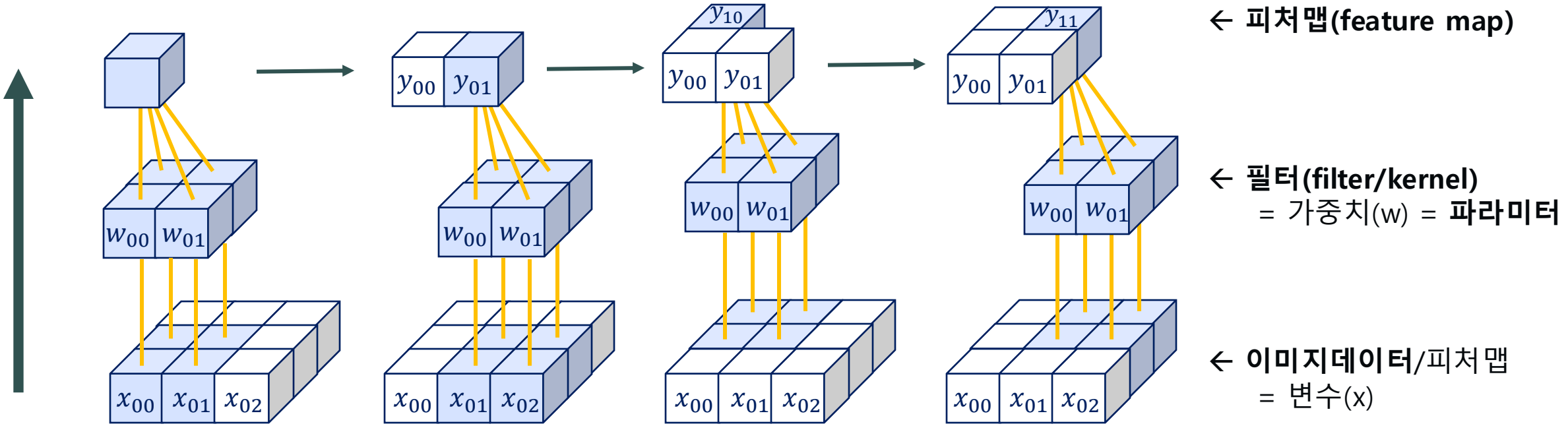
- 이미지 위에 필터를 이동시키면서 겹치는 부분의 원소별 곱셈의 합을 구하는 연산



Convolution(합성곱) 연산(2)

- CNN 이미지의 합성곱 : 이미지 위에서 필터(filter)를 이동시키면서 겹치는 부분의 원소별 곱셈의 합 연산

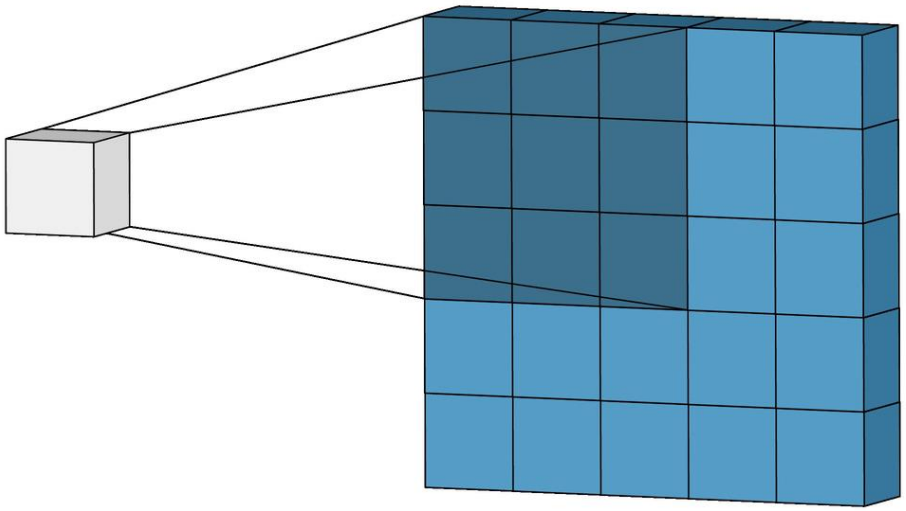
  → 데이터와 필터의 합성곱으로 Feature map 생성



Convolution(합성곱) 연산(3)

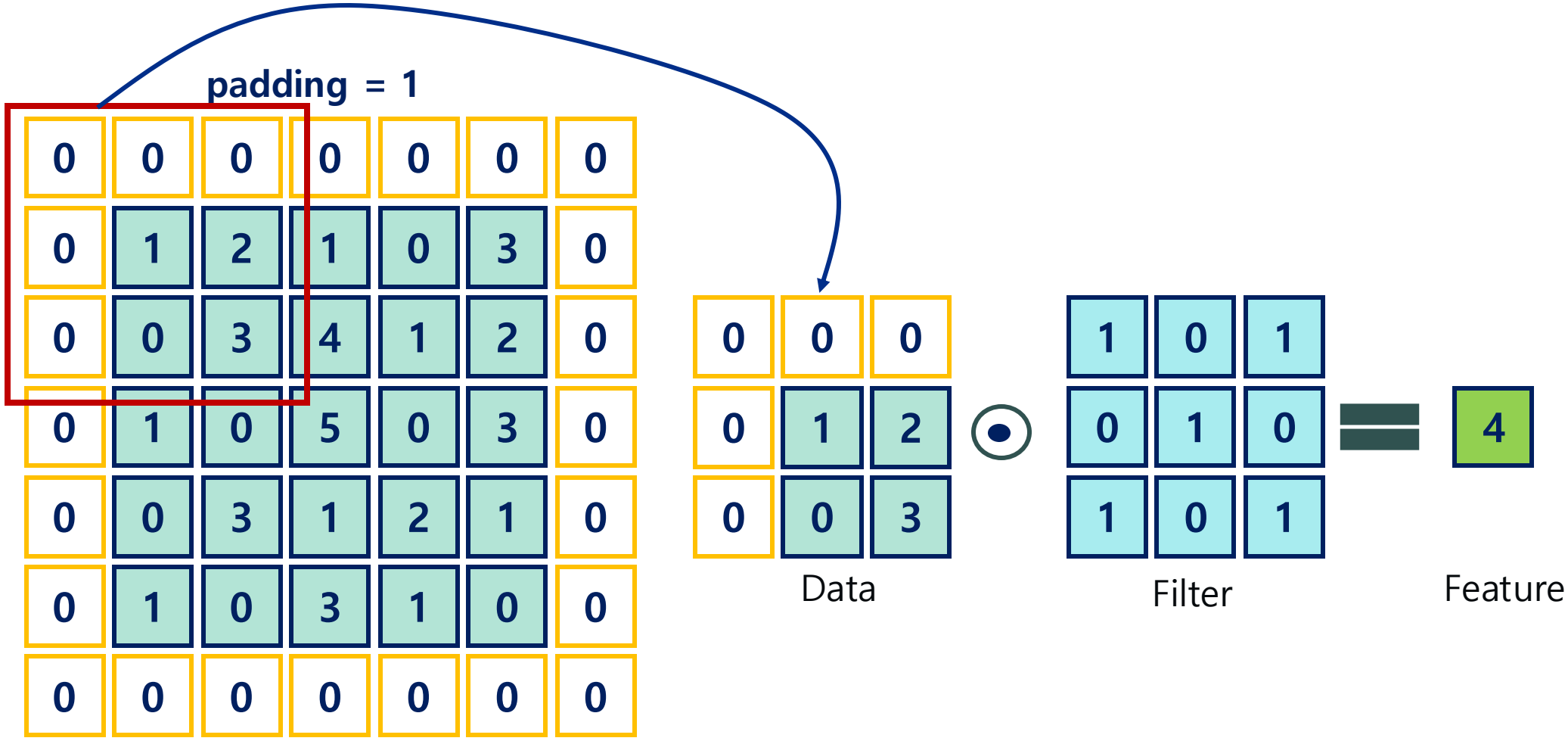
- CNN 이미지의 합성곱 : 이미지 위에서 필터(filter)를 이동시키면서 겹치는 부분의 원소별 곱셈의 합 연산

  → 데이터와 필터의 합성곱으로 Feature map 생성



Padding

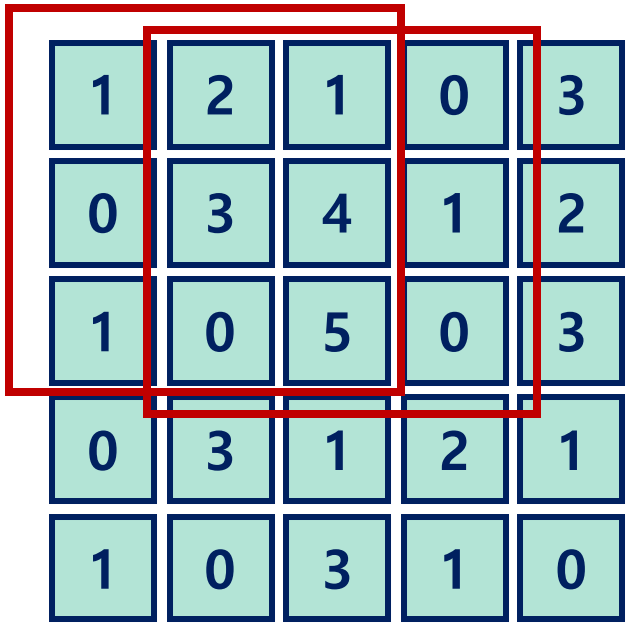
- 이미지 주변을 특정값으로 채우는 것. 가장자리 데이터의 영향력을 중앙부와 비슷하게 유지 목적



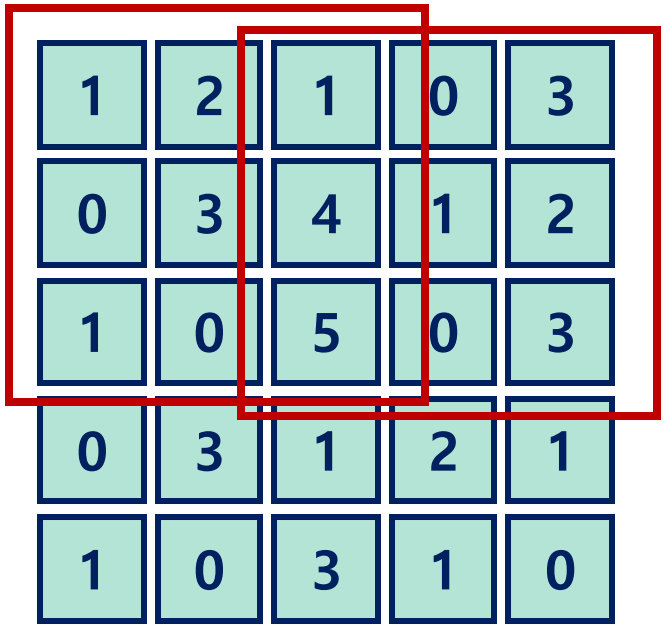
Stride

- Filter가 이동하는 보폭(Step)의 크기. Stride에 따라 피처맵의 크기와 학습속도가 달라짐.

Stride = 1

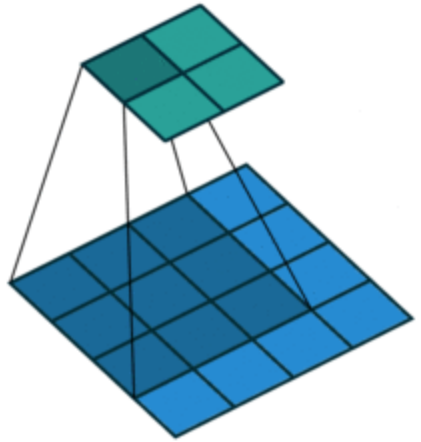


Stride = 2

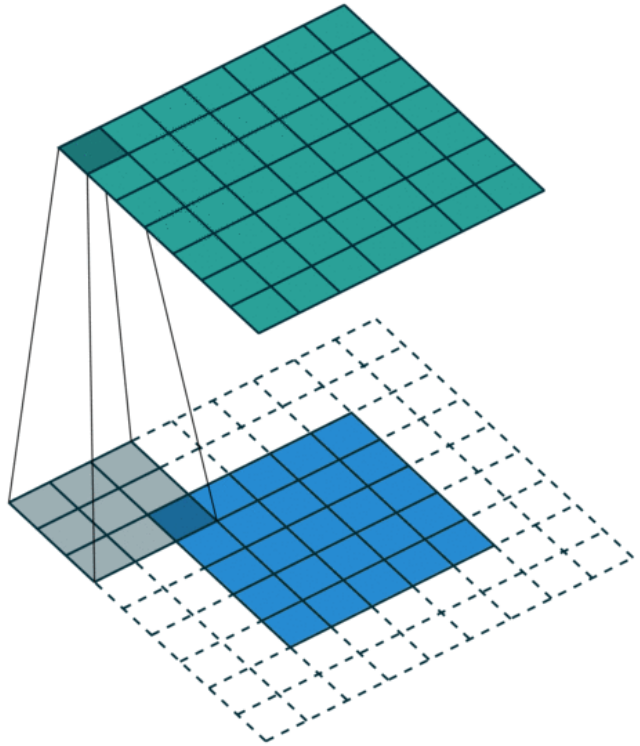


Padding과 Stride를 활용한 행렬의 합성곱

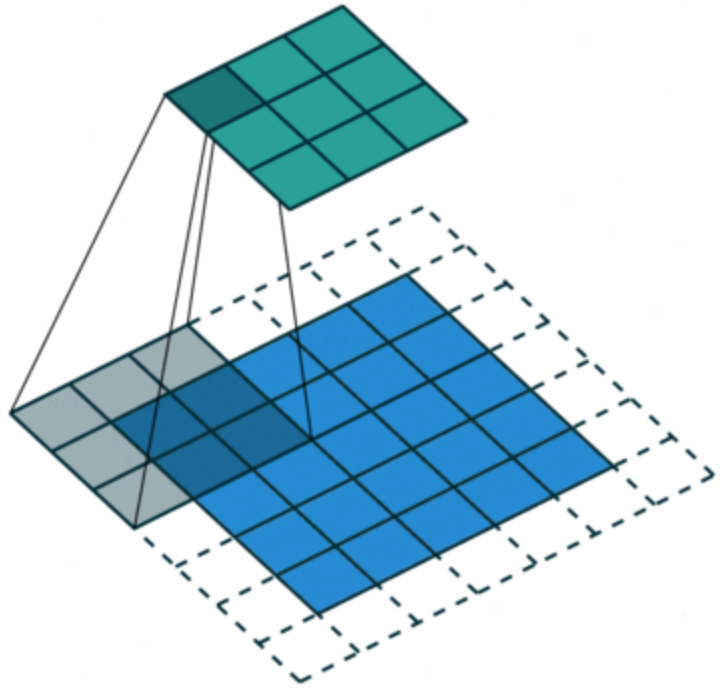
- 다양한 Filter와 Padding에 따른 Feature map의 생성 예



Filter : 3×3
Padding : 0
stride=1



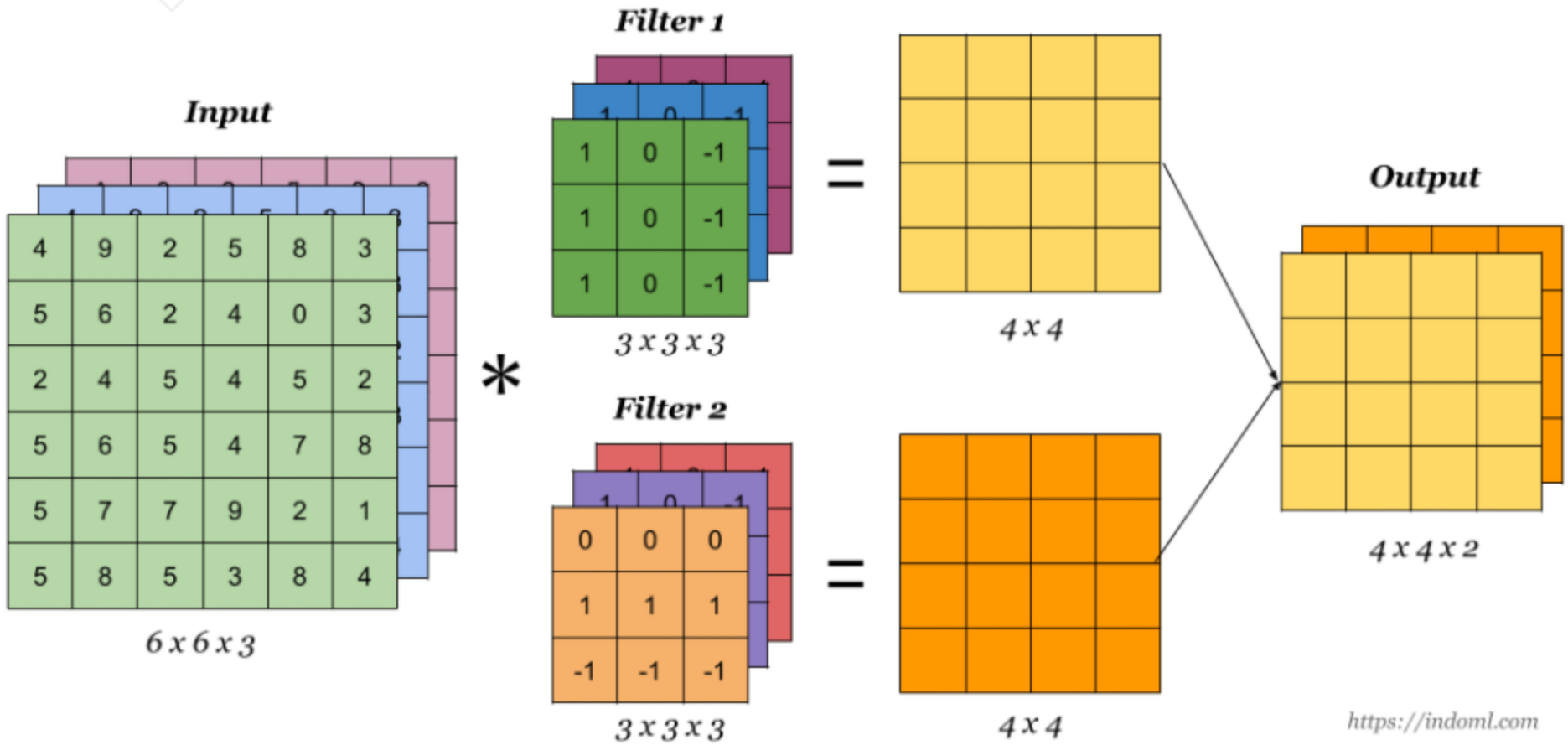
Filter : 3×3
Padding : 2
stride=1



Filter : 3×3
Padding : 1
stride=2

3차원 텐서의 합성곱(1)

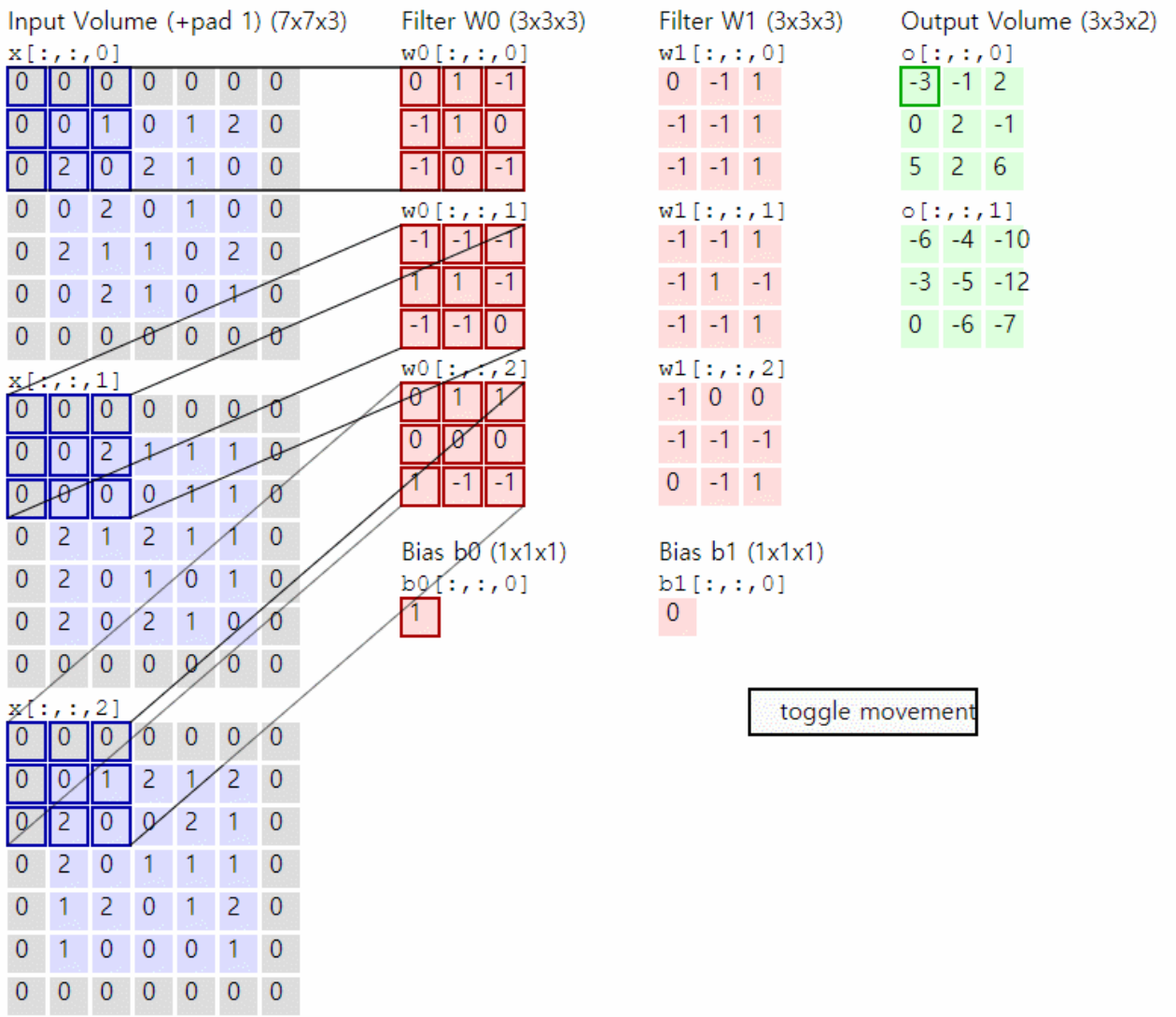
- 3차원 합성곱 연산에서 주의할 점은 입력 데이터의 채널와 필터의 채널수가 동일해야 한다는 것



Filter : 3×3×3
Padding : 0
stride=1
Channel=3
Filter Number=2

3차원 텐서의 합성곱(2)

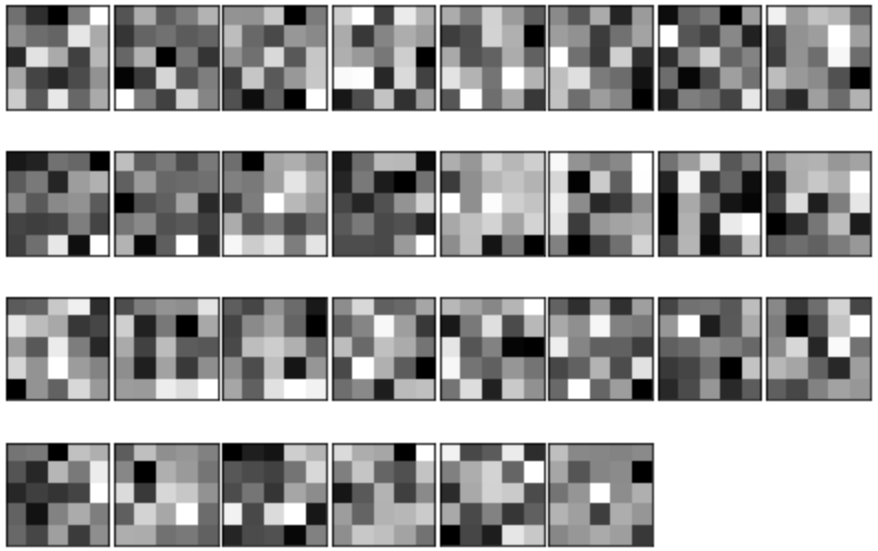
- Filter가 여러개 일때 3차원 텐서의 합성곱



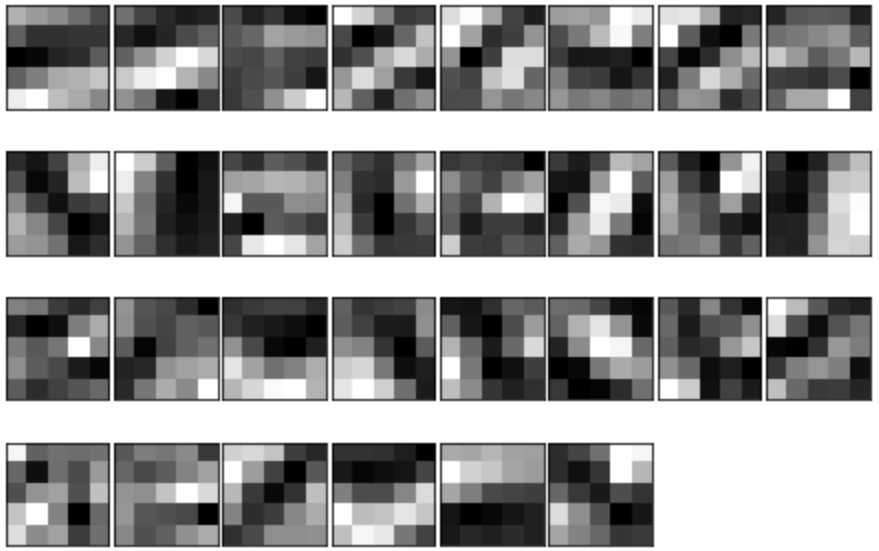
Filter : 3×3×3
Padding : 1
stride=2
Channel=3
Filter Number=2

필터의 학습

- 역전파 알고리즘을 사용하여 입력 데이터와 연결된 필터의 가중치를 학습함.



처음에 랜덤하게 설정된 30개의 5×5필터



MNIST 데이터로 학습이 끝난 30개의 5×5필터

<실습 과제>

Filter의 학습전과 후를 비교하는

코드를 구현해보자

참고 > ch06/visualize_filter.ipynb 에 작성함.

실습 과제

실습: ch06/visualize_filter.ipynb

```python
# coding: utf-8
import os, sys
print(os.getcwd())
current_dir = os.path.dirname(os.getcwd())
print(current_dir)
os.chdir(current_dir)

import numpy as np
import matplotlib.pyplot as plt
from ch06.simple_convnet import SimpleConvNet

def filter_show(filters, nx=8, margin=3, scale=10):
    """
    c.f. https://gist.github.com/aidiary/07d530d5e08011832b12#file-draw_weight-py
    """
    FN, C, FH, FW = filters.shape
    ny = int(np.ceil(FN / nx))

    fig = plt.figure()
    fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

    for i in range(FN):
        ax = fig.add_subplot(ny, nx, i+1, xticks=[], yticks=[])
        ax.imshow(filters[i, 0], cmap=plt.cm.gray_r, interpolation='nearest')
    plt.show()


network = SimpleConvNet()
# 무작위(랜덤) 초기화 후의 가중치
filter_show(network.params['W1'])

# 학습된 가중치
network.load_params("ch06/params.pkl")
filter_show(network.params['W1'])
```

미리 만들어둔 합성곱 신경망 사용

CNN 필터를 그려주는 함수

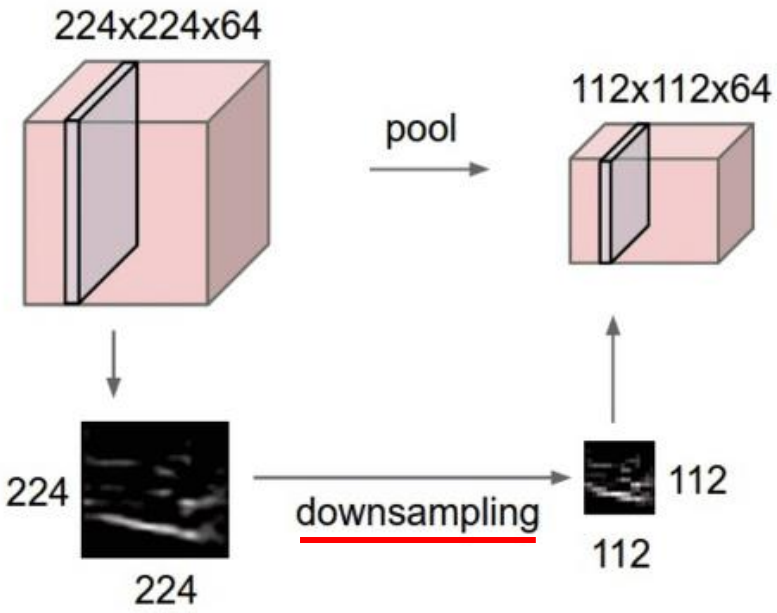
학습전 CNN 필터를 그려줌

학습후 CNN 필터를 그려줌

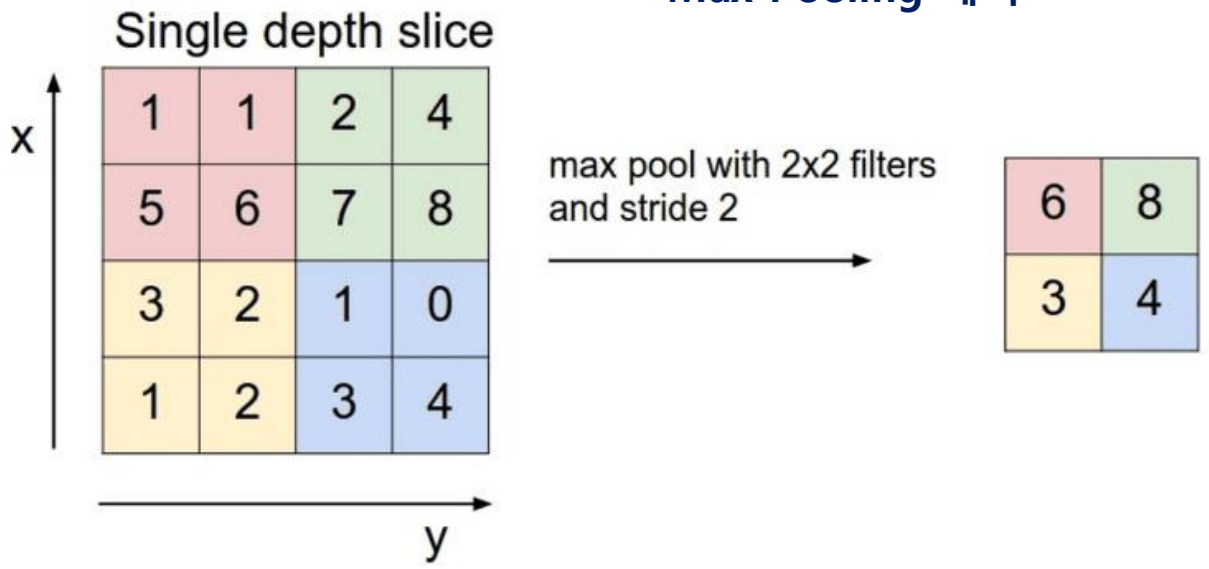
목 차

Pooling(1)

- Pooling : 피처맵을 대표하는 값으로 압축함(sub sampling)
- 피처맵에서 필터 크기를 바탕으로 최대값이나 평균값을 취함



Max Pooling 예시



필터의 크기가 2인 Pooling layer :

- 피처맵의 크기를 1/4 로 축소,
- 피처맵의 개수는 변함 없음

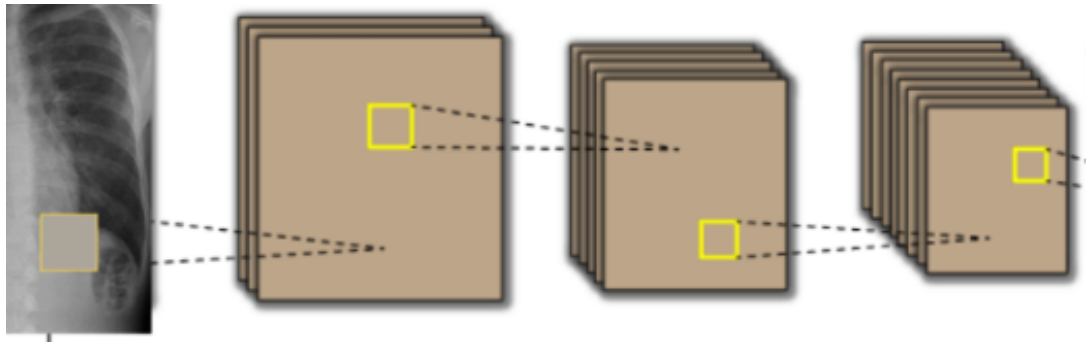
* Convolution layer : **지역 정보 추출**

Pooling layer : 지역정보 중 **두드러진 정보 추출**

합성곱층 vs Max pooling층

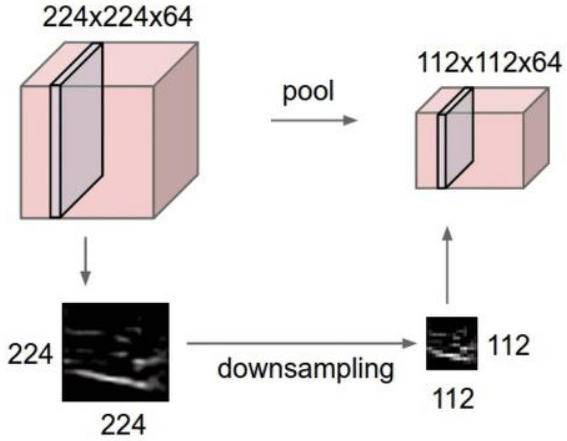
합성곱층

1. 가중치와 편향 학습 학습
2. 컬러 채널이 사라짐
3. 필터 훈련을 통해 저수준 필터는 지역적 특징을 찾아내고, 고수준의 필터는 전역적 특징을 찾아냄.



Max Pooling층

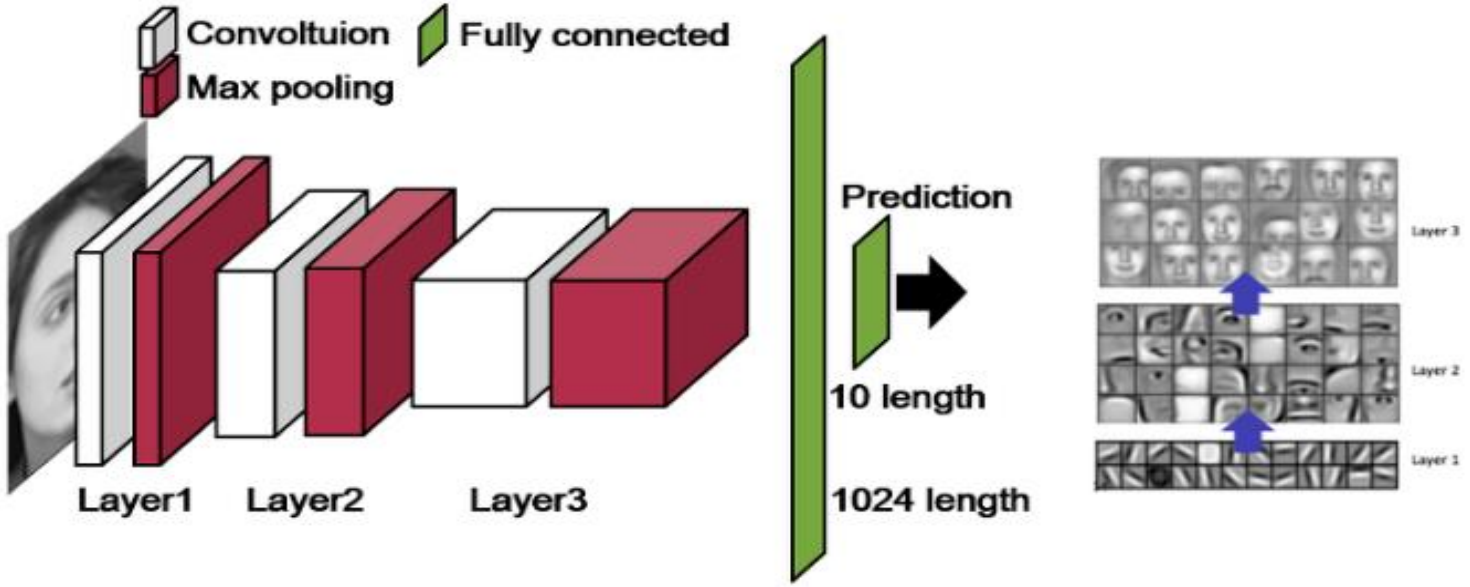
1. 학습이 없는 단순한 계산
2. 컬러 채널 별 독립적 연산
3. DownSampling을 통하여 다음 층에서 전역적 특징을 찾아낼 수 있게 도와줌. 파라미터 숫자를 줄여서 계산비용 감소와 overfitting을 억제. 약간의 평행이동은 영향이 미치지 않도록 함.



목 차

합성곱 신경망의 구현

- 합성곱층과 max pooling층이 반복되다가 후반에 fully connected layer가 등장함
- 합성곱층의 필터들은 학습 시작전에 랜덤하게 생성함
- 데이터를 통한 학습을 통해 손실함수값을 낮추도록 필터를 훈련시킴
- 충분한 학습이 이루어지면 낮은 층의 필터는 저수준의 로컬한 특징을 찾아내고 층이 높아질수록 더 고수준의 더 글로벌한 특징을 찾아낸다.



<실습 과제>

SimpleConvNet 클래스

코드를 작성하자

참고 > ch06/simple_convnet.py

실습 과제

```python
class SimpleConvNet:
    """단순한 합성곱 신경망

    conv - relu - pool - affine - relu - affine - softmax

    Parameters
    ----------
    input_size : 입력 크기(MNIST의 경우엔 784)
    hidden_size_list : 각 은닉층의 뉴런 수를 담은 리스트(e.g. [100, 100, 100])
    output_size : 출력 크기(MNIST의 경우엔 10)
    activation : 활성화 함수 - 'relu' 혹은 'sigmoid'
    weight_init_std : 가중치의 표준편차 지정(e.g. 0.01)
        'relu'나 'he'로 지정하면 'He 초깃값'으로 설정
        'sigmoid'나 'xavier'로 지정하면 'Xavier 초깃값'으로 설정
    """
    def __init__(self, input_dim=(1, 28, 28),
                 conv_param={'filter_num':30, 'filter_size':5, 'pad':0, 'stride':1},
                 hidden_size=100, output_size=10, weight_init_std=0.01):
        filter_num = conv_param['filter_num']
        filter_size = conv_param['filter_size']
        filter_pad = conv_param['pad']
        filter_stride = conv_param['stride']
        input_size = input_dim[1]
        conv_output_size = (input_size - filter_size + 2*filter_pad) / filter_stride + 1
        pool_output_size = int(filter_num * (conv_output_size/2) * (conv_output_size/2))

        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * \
                            np.random.randn(filter_num, input_dim[0], filter_size, filter_size)
        self.params['b1'] = np.zeros(filter_num)
        self.params['W2'] = weight_init_std * \
                            np.random.randn(pool_output_size, hidden_size)
        self.params['b2'] = np.zeros(hidden_size)
        self.params['W3'] = weight_init_std * \
                            np.random.randn(hidden_size, output_size)
        self.params['b3'] = np.zeros(output_size)

```

Simple한 CNN 클래스에 대한 선언

CNN Network 구성에 대한 설명

CNN Network 하이퍼 파라메터 설정

가중치 초기화

실습 과제

```
51
52          # 계층 생성
53          self.layers = OrderedDict()
54          self.layers['Conv1'] = Convolution(self.params['W1'], self.params['b1'],
55                                             conv_param['stride'], conv_param['pad'])
56          self.layers['Relu1'] = Relu()
57          self.layers['Pool1'] = Pooling(pool_h=2, pool_w=2, stride=2)
58          self.layers['Affine1'] = Affine(self.params['W2'], self.params['b2'])
59          self.layers['Relu2'] = Relu()
60          self.layers['Affine2'] = Affine(self.params['W3'], self.params['b3'])
61
62          self.last_layer = SoftmaxWithLoss()
63
64      def predict(self, x):
65          for layer in self.layers.values():
66              x = layer.forward(x)
67
68          return x
69
70      def loss(self, x, t):
71          """손실 함수를 구한다.
72
73          Parameters
74          ----------
75          x : 입력 데이터
76          t : 정답 레이블
77          """
78          y = self.predict(x)
79          return self.last_layer.forward(y, t)
80
```

전체 네트워크 구성

순전파 함수

손실값을 구하는 함수

실습 과제

```python
def accuracy(self, x, t, batch_size=100):
    if t.ndim != 1 : t = np.argmax(t, axis=1)

    acc = 0.0

    for i in range(int(x.shape[0] / batch_size)):
        tx = x[i*batch_size:(i+1)*batch_size]
        tt = t[i*batch_size:(i+1)*batch_size]
        y = self.predict(tx)
        y = np.argmax(y, axis=1)
        acc += np.sum(y == tt)

    return acc / x.shape[0]

def numerical_gradient(self, x, t):
    """기울기를 구한다(수치미분).

    Parameters
    ----------
    x : 입력 데이터
    t : 정답 레이블

    Returns
    -------
    각 층의 기울기를 담은 사전(dictionary) 변수
        grads['W1']、grads['W2']、... 각 층의 가중치
        grads['b1']、grads['b2']、... 각 층의 편향
    """
    loss_w = lambda w: self.loss(x, t)

    grads = {}
    for idx in (1, 2, 3):
        grads['W' + str(idx)] = numerical_gradient(loss_w, self.params['W' + str(idx)])
        grads['b' + str(idx)] = numerical_gradient(loss_w, self.params['b' + str(idx)])

    return grads
```

결과를 정확히 예측했는지 확인하는 함수

수치미분 방식으로 기울기를 구하는 함수

실습 과제

```python
def gradient(self, x, t):
    """기울기를 구한다(오차역전파법).

    Parameters
    ----------
    x : 입력 데이터
    t : 정답 레이블

    Returns
    -------
    각 층의 기울기를 담은 사전(dictionary) 변수
        grads['W1']、grads['W2']、... 각 층의 가중치
        grads['b1']、grads['b2']、... 각 층의 편향
    """
    # forward
    self.loss(x, t)

    # backward
    dout = 1
    dout = self.last_layer.backward(dout)

    layers = list(self.layers.values())
    layers.reverse()
    for layer in layers:
        dout = layer.backward(dout)

    # 결과 저장
    grads = {}
    grads['W1'], grads['b1'] = self.layers['Conv1'].dW, self.layers['Conv1'].db
    grads['W2'], grads['b2'] = self.layers['Affine1'].dW, self.layers['Affine1'].db
    grads['W3'], grads['b3'] = self.layers['Affine2'].dW, self.layers['Affine2'].db

    return grads
```

오차역전파 방식으로 기울기를 구하는 함수

실습 과제

```
152     def save_params(self, file_name="params.pkl"):
153         params = {}
154         for key, val in self.params.items():
155             params[key] = val
156         with open(file_name, 'wb') as f:
157             pickle.dump(params, f)
158
159     def load_params(self, file_name="params.pkl"):
160         with open(file_name, 'rb') as f:
161             params = pickle.load(f)
162         for key, val in params.items():
163             self.params[key] = val
164
165         for i, key in enumerate(['Conv1', 'Affine1', 'Affine2']):
166             self.layers[key].W = self.params['W' + str(i+1)]
167             self.layers[key].b = self.params['b' + str(i+1)]
168
```

학습된 가중치를 저장하는 함수

저장된 가중치를 로딩하는 함수

<실습 과제>

CNN 클래스로 학습 및 검증을 구현하자

참고 > ch06/train_convnet.ipynb 에 작성함.

실습 과제

실습: ch06/train_convnet.ipynb

```
1   import os, sys
2   print(os.getcwd())
3   current_dir = os.path.dirname(os.getcwd())
4   print(current_dir)
5   os.chdir(current_dir)
6
7   import numpy as np
8   import matplotlib.pyplot as plt
9   from dataset.mnist import load_mnist
10  from ch06.simple_convnet import SimpleConvNet
11  from common.trainer import Trainer
12
13  # 데이터 읽기
14  (x_train, t_train), (x_test, t_test) = load_mnist(flatten=False)
15
16  # 시간이 오래 걸릴 경우 데이터를 줄인다.
17  x_train, t_train = x_train[:5000], t_train[:5000]
18  x_test, t_test = x_test[:1000], t_test[:1000]
19
20  max_epochs = 20
21
22  network = SimpleConvNet(input_dim=(1,28,28),
23                          conv_param = {'filter_num': 30, 'filter_size': 5, 'pad': 0, 'stride': 1},
24                          hidden_size=100, output_size=10, weight_init_std=0.01)
25
26  trainer = Trainer(network, x_train, t_train, x_test, t_test,
27                    epochs=max_epochs, mini_batch_size=100,
28                    optimizer='Adam', optimizer_param={'lr': 0.001},
29                    evaluate_sample_num_per_epoch=1000)
30  trainer.train()
31
```

미리 만들어둔 CNN 을 활용함

필터 갯수를 30개로 설정

필터 크기 5x5

실습 과제

실습: ch06/train_convnet.ipynb

```
32  # 매개변수 보존
33  network.save_params("params.pkl")
34  print("Saved Network Parameters!")
35
36  # 그래프 그리기
37  markers = {'train': 'o', 'test': 's'}
38  x = np.arange(max_epochs)
39  plt.plot(x, trainer.train_acc_list, marker='o', label='train', markevery=2)
40  plt.plot(x, trainer.test_acc_list, marker='s', label='test', markevery=2)
41  plt.xlabel("epochs")
42  plt.ylabel("accuracy")
43  plt.ylim(0, 1.0)
44  plt.legend(loc='lower right')
45  plt.show()
```

학습 종료 후 가중치값 저장