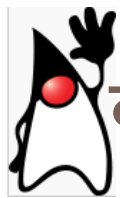


9장 컨벌루션 신경망



학습 목표

- 컨벌루션 신경망을 이해한다.
- 컨벌루션 신경망을 적용해본다.

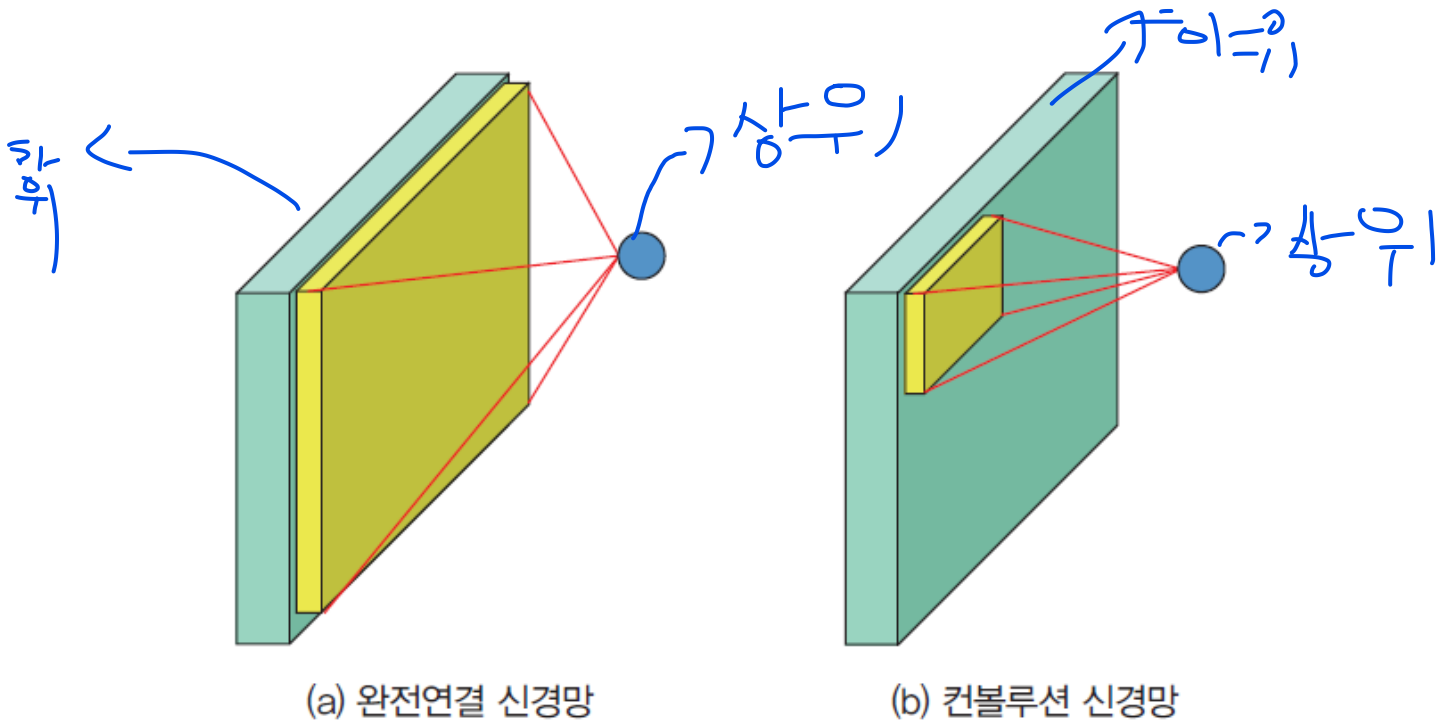


컨벌루션 신경망은
합성곱 신경망으로도 번역됩니다.
컨벌루션은 인간의 눈이 영상 정보를
처리하는 것을 모방해서 만든 신경망
입니다. 복잡하긴 하지만 일반적으로
기존의 신경망보다 성능이
우수합니다.



컨볼루션 신경망(CNN)

- 컨볼루션 (Convolution Neural Network: CNN) 신경망에서는 하위 레이어의 노드들과 상위 레이어의 노드들이 부분적으로만 연결되어 있다.





컨볼루션 신경망(CNN)

- 컨볼루션 신경망은 Hubel과 Wiesel이 발견한 **고양이의 시각 세포**에서부터 출발한다.

가

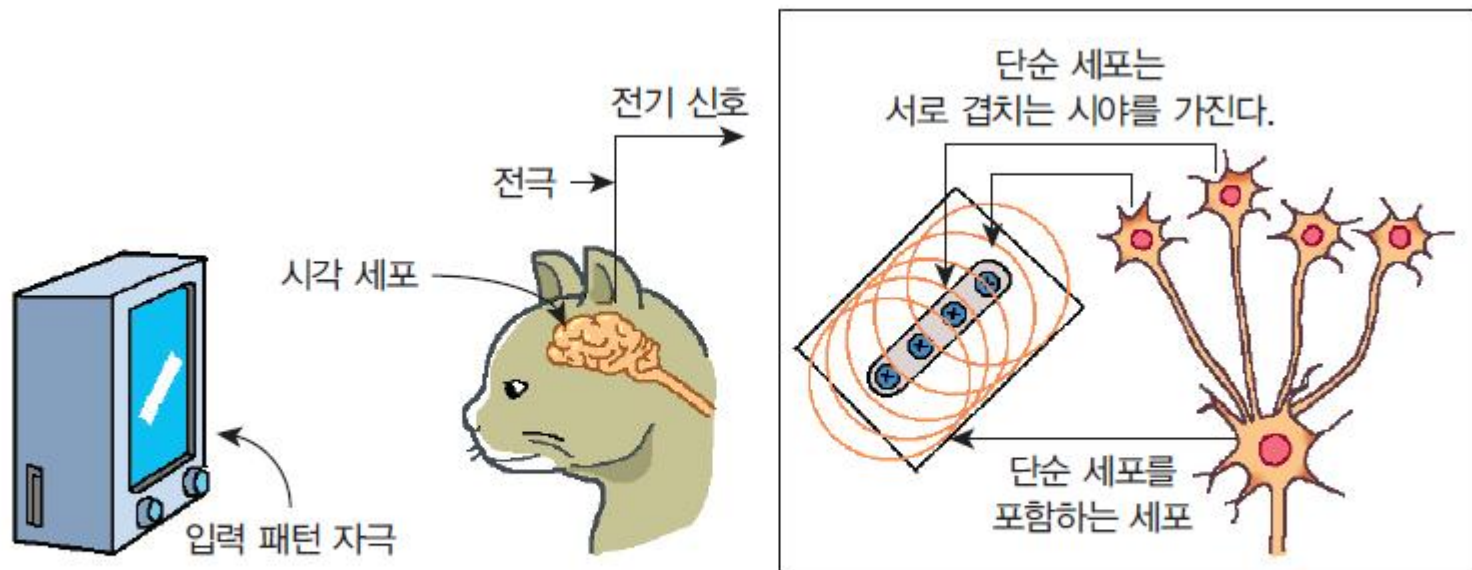


그림 9-2 시각 피질 뉴런의 구조(출처: Hubel's book, Eye, Brain, and Vision)



네오코그니트론

- “네오코그니트론(Neocognitron)”은 1980년 후쿠시마(Kunihiko Fukushima)에 의해 소개된 신경망 구조이다. 후쿠시마도 위에서 언급한 허벨과 위젤의 작업에서 영감을 받았다.
- 후쿠시마는 허벨과 위젤이 발견한 단순 세포와 복합 세포를 패턴 인식에 사용하기 위해, 이 두 가지 유형의 세포로 이루어진 계층적인 모델을 제안했다. 또 네오코그니트론에서는 컨볼루션 신경망의 두 가지 기본 유형 레이어인, 컨볼루션 레이어와 서브 샘플링 레이어를 도입했다

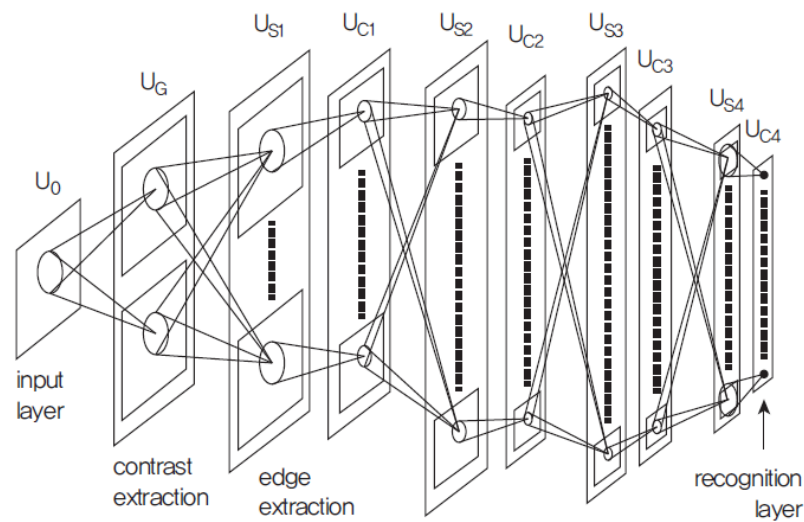
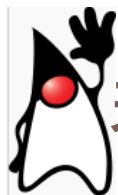


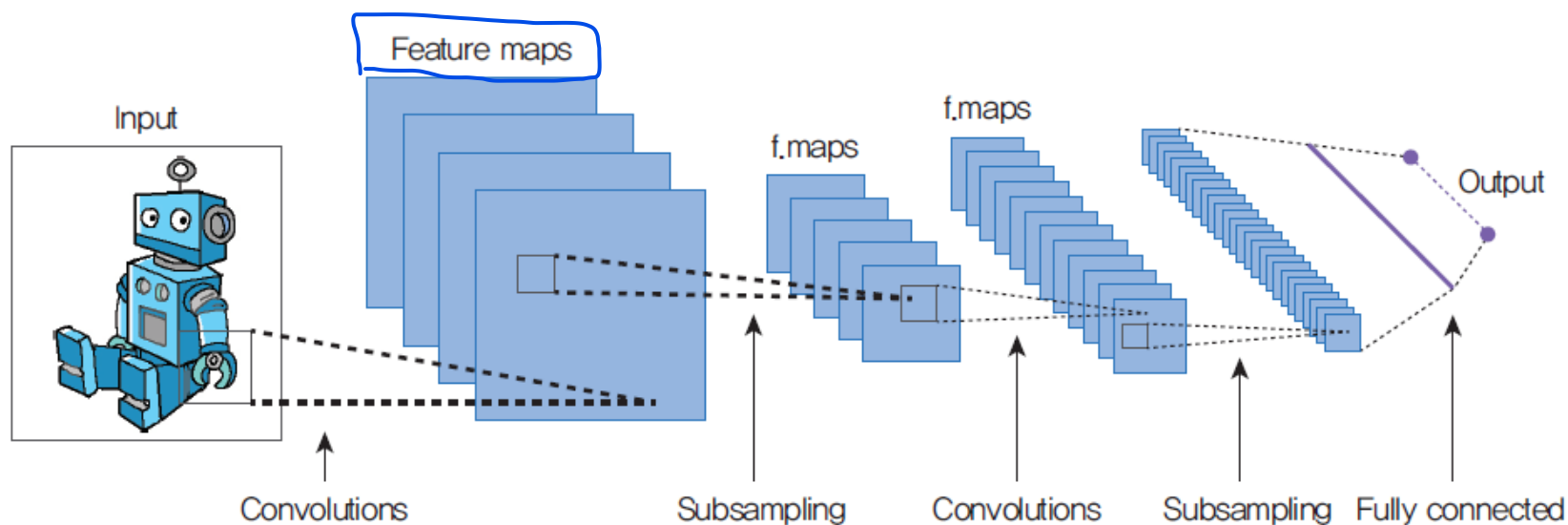
그림 9-3 네오코그니트론(출처: 후쿠시마의 논문)



컨벌루션 신경망의 중요성

가

- 컨벌루션 신경망은 모든 신경망 구조 중에서 가장 강력한 성능을 보여주는 신경망 중의 하나이다.
- 컨벌루션 신경망은 2차원 형태의 입력을 처리하기 때문에, 이미지 처리에 특히 적합하다. 신경망의 각 레이어에서 일련의 필터가 이미지에 적용된다.





컨벌루션 신경망의 중요성

- 2012년도 영상 인식 경진 대회 ILSVRC에서 우승하여, 딥러닝의 우수성을 전 세계에 알리고 인공지능 부활의 신호탄이 되었던 신경망 모델도 컨벌루션 신경망이었다

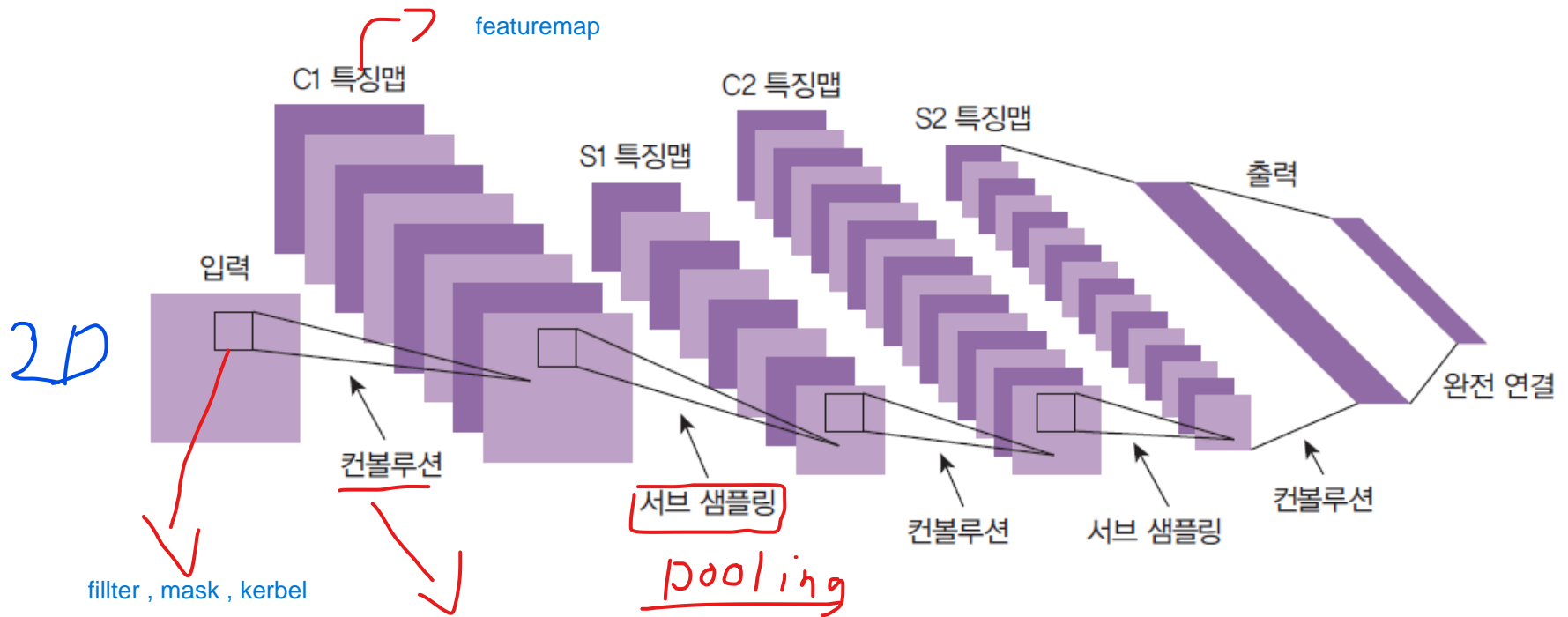




CNN의 일반적인 구조

input
D → (conv - pool - conv -)

- 컨볼루션 신경망도 여러 레이어를 연결하여 신경망을 구축한다



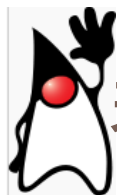


CNN의 일반적인 구조

- 입력층에서 컨벌루션 연산을 통하여 특징을 뽑아내는 **특징맵** (feature map)이 존재
- **풀링 (Pooling) 연산**을 적용한다. 풀링 연산은 입력의 차원을 줄이는 연산이다.
- 컨벌루션 레이어와 풀링 레이어는 **여러 번 되풀이** 된다.
- 신경망의 맨 끝에는 **완전히 연결된 구조의** 전통적인 분류 신경망이 있어서 추출된 특징을 바탕으로 물체를 인식한다.



full connected layer



컨벌루션

f ;

가

가 가

L

=> weight

$X * \text{filter}$

- 컨벌루션은 주변 화소값들에 가중치를 곱해서 더한 후에 이것을 새로운 화소값으로 하는 연산이다.

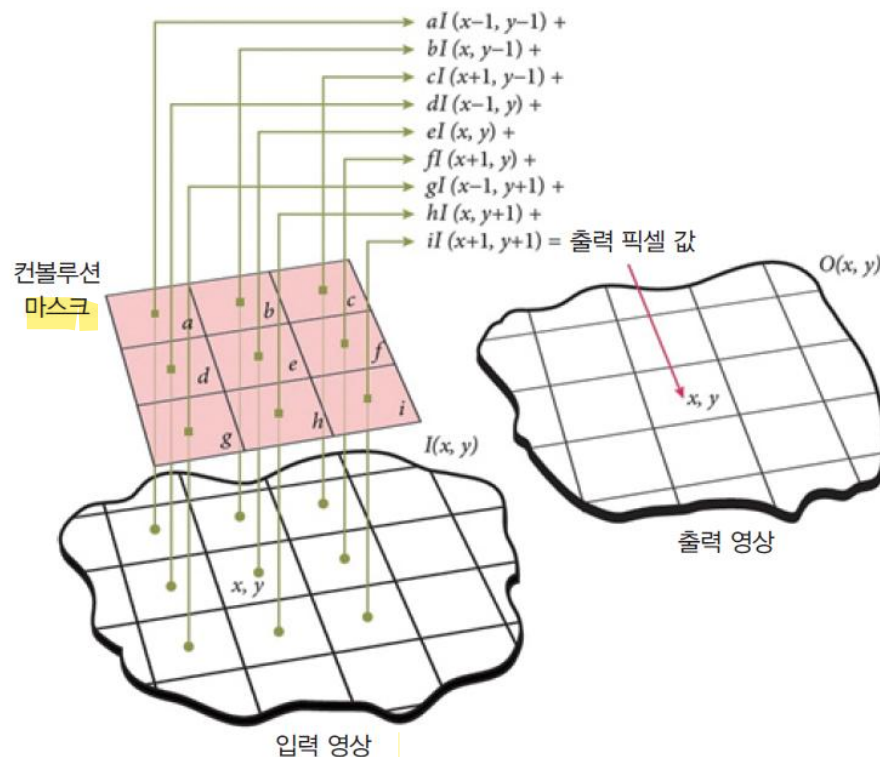
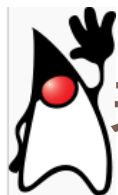
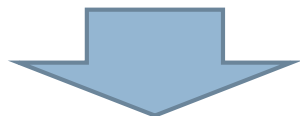


그림 9-6 영상 처리에서 컨벌루션 연산



컨벌루션 연산

$$\begin{aligned} O(x, y) = & aI(x-1, y-1) + bI(x, y-1) + cI(x+1, y-1) \\ & + dI(x-1, y) + eI(x, y) + fI(x+1, y) \\ & + gI(x-1, y+1) + hI(x, y+1) + iI(x+1, y+1) \end{aligned}$$



$$O(x, y) = \sum_{k=-1}^{k=+1} \sum_{l=-1}^{l=+1} h(k, l) I(x+k, y+l)$$



컨볼루션

가

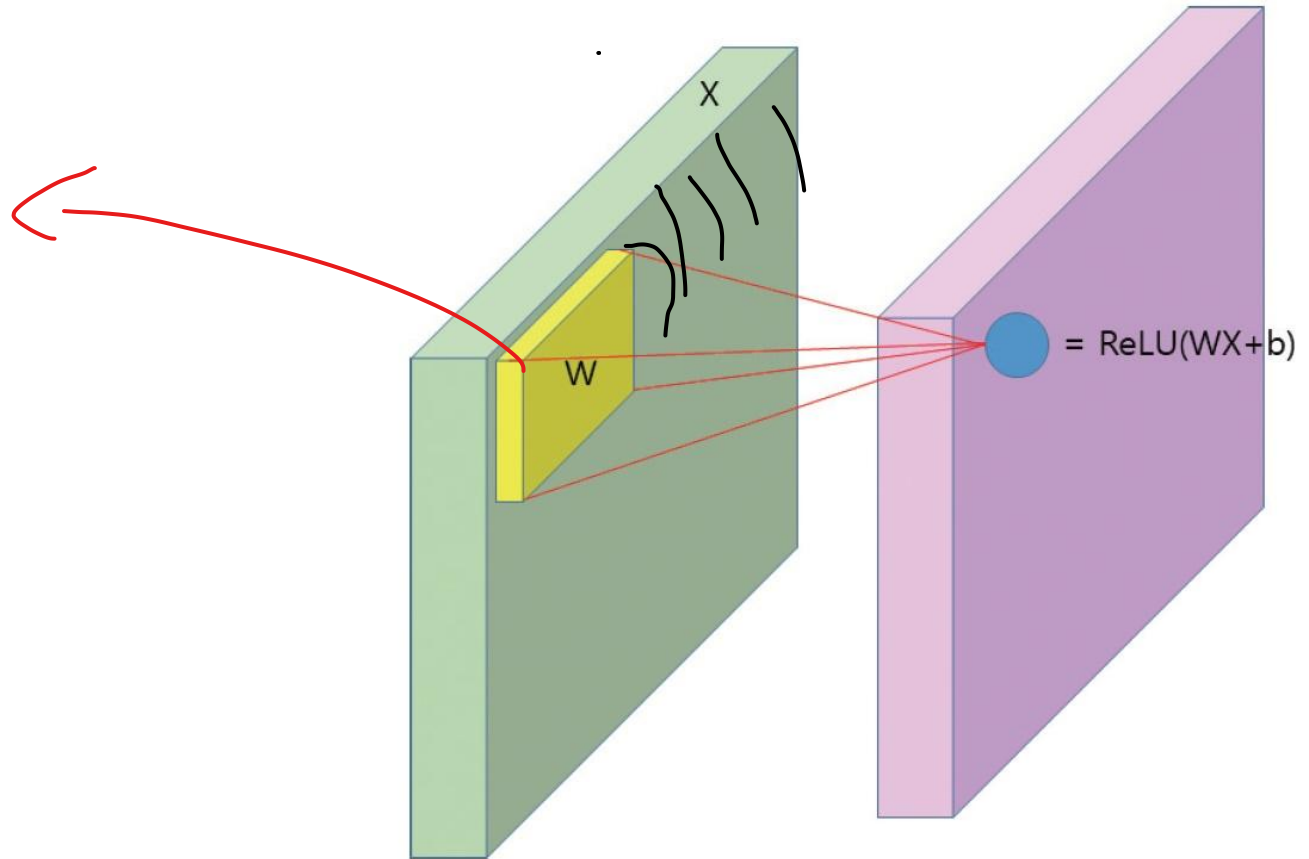
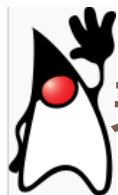


그림 15-15 신경망에서의 컨볼루션 연산



컨벌루션의 구체적인 예

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

가중치 커널

가 가

3	6	6	4	7	8	2	1
3	4	3	8	8	3	3	2
5	7	7	7	7	4	3	2
8	9	9	9	9	9	3	2
8	3	3	4	3	2	1	1
8	9	9	8	8	3	3	2
6	4	3	8	8	3	3	2
7	4	3	8	8	3	3	2

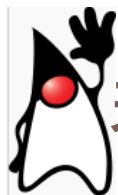
입력 레이어

	4.88						

출력 레이어

$$\text{ReLU}(1/9 * 3 + 1/9 * 6 + 1/9 * 6 + 1/9 * 3 + 1/9 * 4 + 1/9 * 3 + 1/9 * 5 + 1/9 * 7 + 1/9 * 7) = 4.88$$

$\tau(\cdot) = \text{ReLU}$



컨벌루션의 구체적인 예

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

가중치 마스크

3	6	6	4	7	8	2	1
3	4	3	8	8	3	3	2
5	7	7	7	7	4	3	2
8	9	9	9	9	9	3	2
8	3	3	4	3	2	1	1
8	9	9	8	8	3	3	2
6	4	3	8	8	3	3	2
7	4	3	8	8	3	3	2

입력 레이어

	4.88	5.77					

다음 레이어

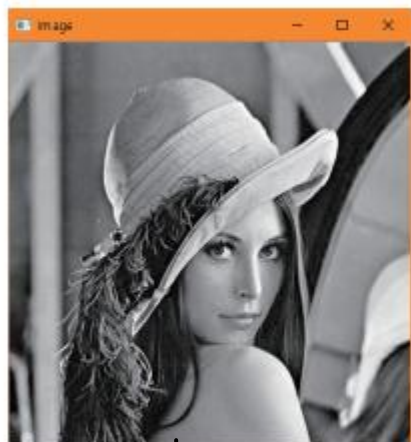
$$\text{ReLU}(1/9 * 6 + 1/9 * 6 + 1/9 * 4 + 1/9 * 4 + 1/9 * 3 + 1/9 * 8 + 1/9 * 7 + 1/9 * 7 + 1/9 * 7) = 5.77$$



영상 처리에서의 컨벌루션 연산

- 컨벌루션을 수행한 결과는 특징맵(feature map)이라고 불리는 데, 그림 9-9가 그 이유를 보여준다.

$$f(\sum (f \times i_n))$$



-1	-1	-1
-1	8	-1
-1	-1	-1



mask

그림 9-9 영상 처리에서의 컨벌루션 연산

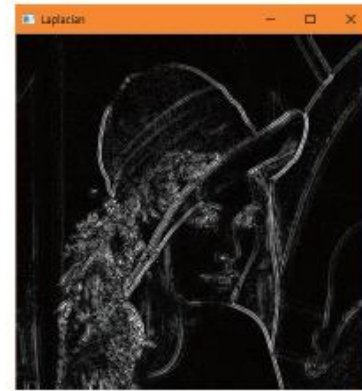
feature



신경망에서는 필터의 가중치가 학습된다.



?	?	?
?	?	?
?	?	?



분류: 인간



오차 역전파

그림 9-10 컨벌루션 신경망에서는 커널의 가중치들이 학습된다.



컨벌루션 신경망에서의 컨벌루션 연산

- 컨벌루션 신경망에서도 커널이 입력층의 각 화소를 중심으로 덮여 씌워진다. 앞 레이어의 값 X 는 각 커널 W 와 곱해져서 더해져서 $WX+b$ 가 된다.
- 이 계산값은 $\text{ReLU}()$ 와 같은 활성화 함수를 통과해서, 다음 레이어의 동일한 위치에 $\text{ReLU}(WX+b)$ 로 저장된다.

=

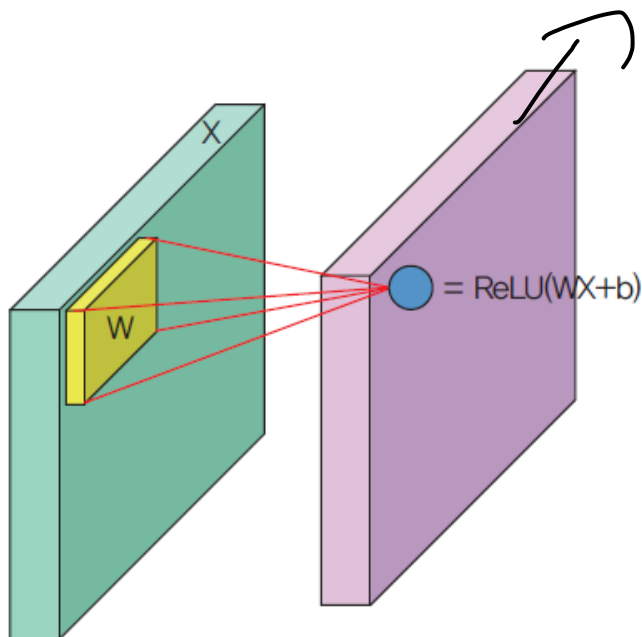


그림 9-11 신경망에서의 컨벌루션 연산



컨벌루션 레이어

- 여러 개의 필터를 이용할 수 있다.
- *필터의 값은 미리 정해진 것이 아니고 학습된다.*

weight

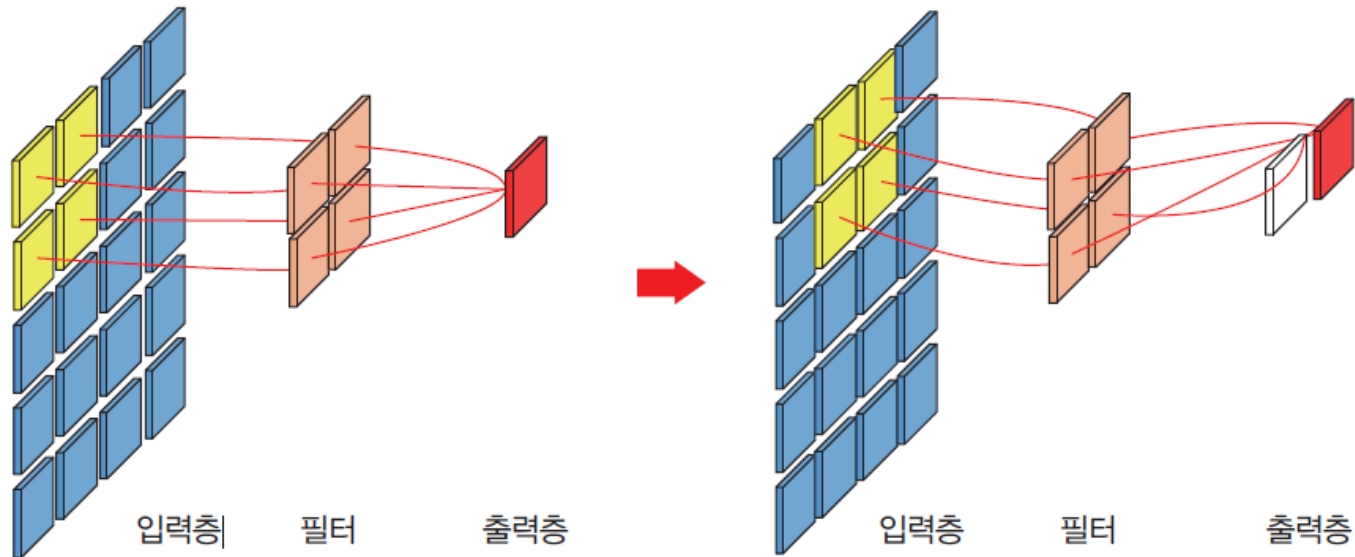


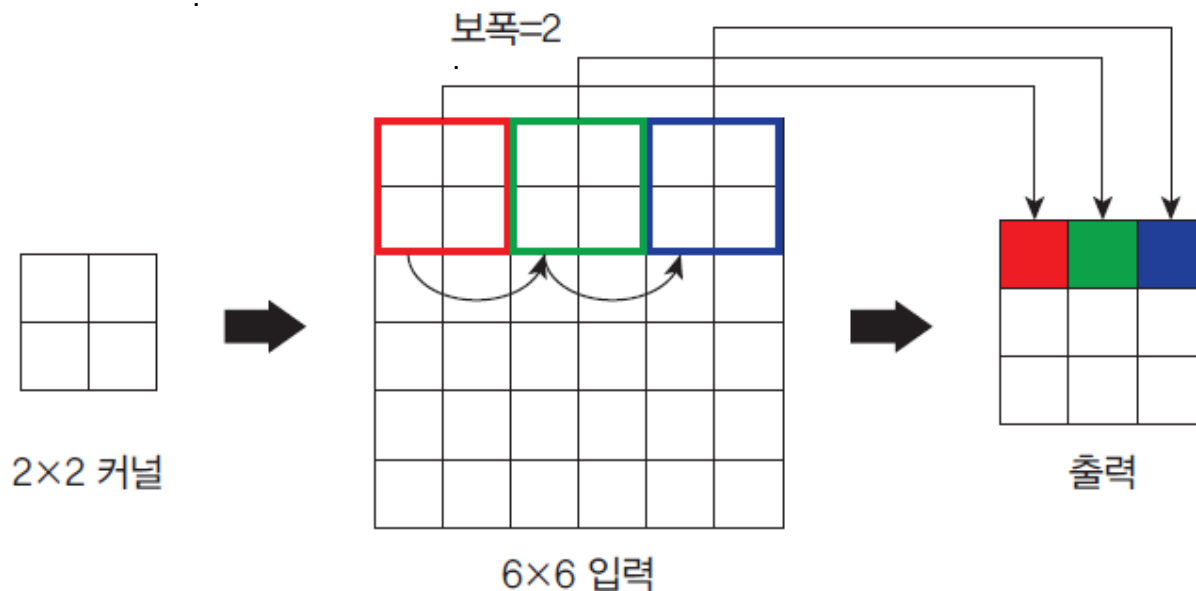
그림 9-12 컨벌루션 연산



- 보폭(stride) 은 커널을 적용하는 거리이다. 보폭이 1이면 커널을 한 번에 1픽셀씩 이동하면서 커널을 적용하는 것이다.
- 보폭이 2라는 것은 하나씩 건너뛰면서 픽셀에 커널을 적용한다는 것을 의미한다.

가

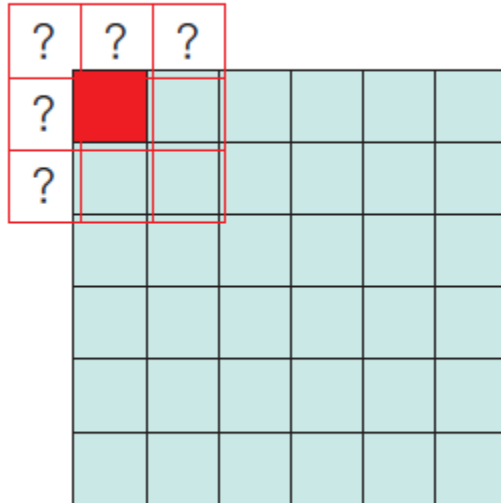
가





- 패딩(padding)은 이미지의 가장자리를 처리하기 위한 기법이다.

가 가 가
가 가 가



이미지의 가장 자리에 커널을
적용하려니, 커널 아래에 픽셀이
없네요. 어떻게 해야 할까요?



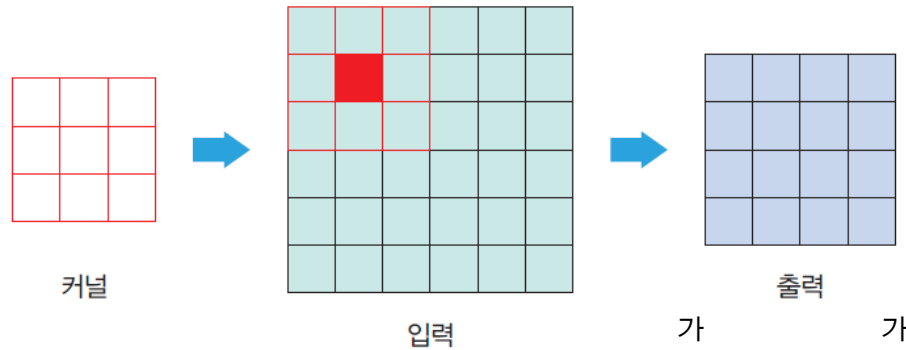
그림 9-13 패딩이 필요한 이유

가
zero padding

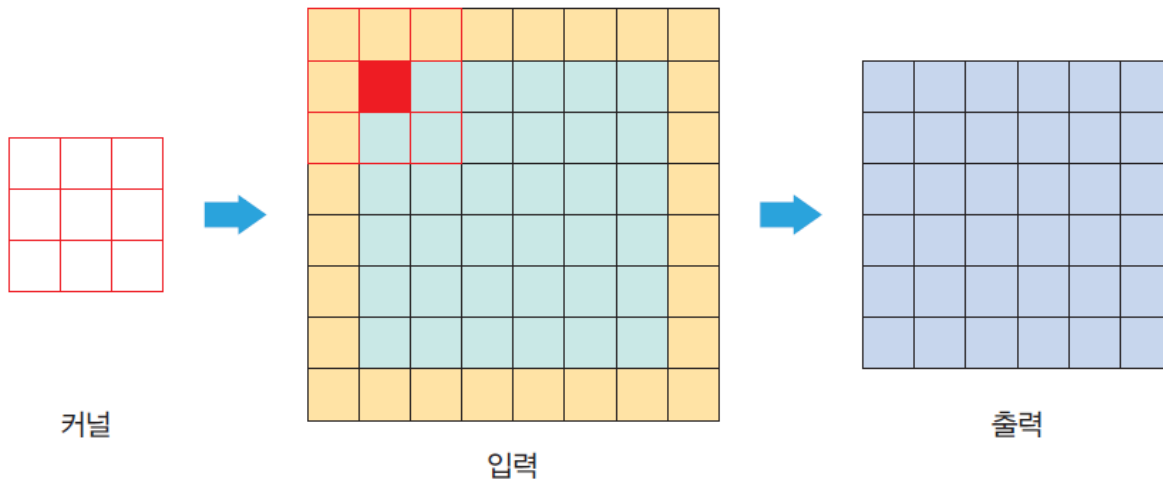


2가지 패딩 방법

- Valid: 커널을 입력 이미지 안에서만 움직인다.



- Same: 입력 이미지의 주변을 특정값(예를 들면 0, 또는 이웃 픽셀값)으로 채우는 것

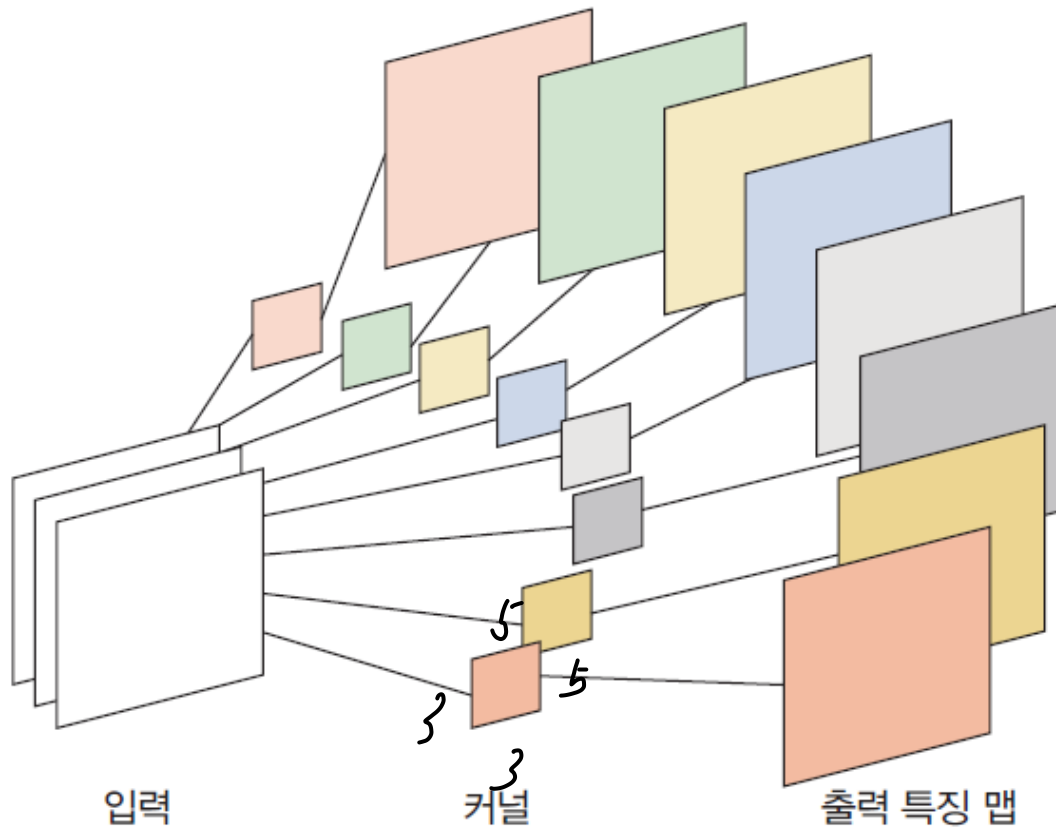




필터가 여러 개일 때의 컨벌루션 레이어

CNN

가

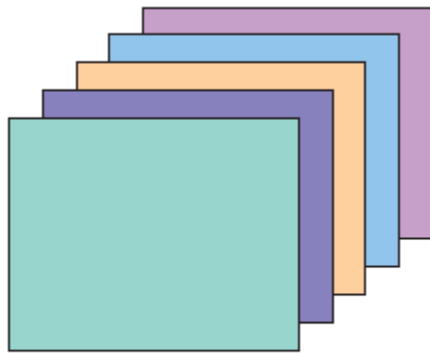


많아지면 자연
스럽게 특징맵
의 개수도 많아

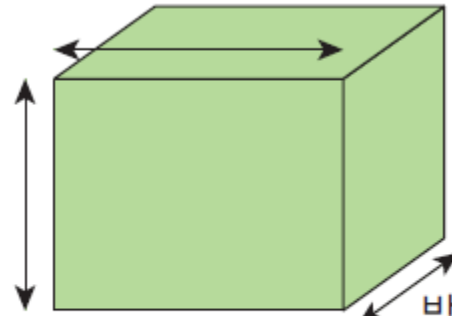
그림 9-14 필터가 여러 개일 때의 컨벌루션 레이어



필터가 여러 개일 때의 컨벌루션 레이어



특징 맵들



특징 맵들을 박스
형태로도 표시한다.

박스의 깊이가
커널의 개수이다.

3

(y, x, n)



풀링 또는 서브 샘플링

- 풀링 (Pooling)이란 서브 샘플링이라고도 하는 것으로 입력 데이터의 크기를 줄이는 것이다.

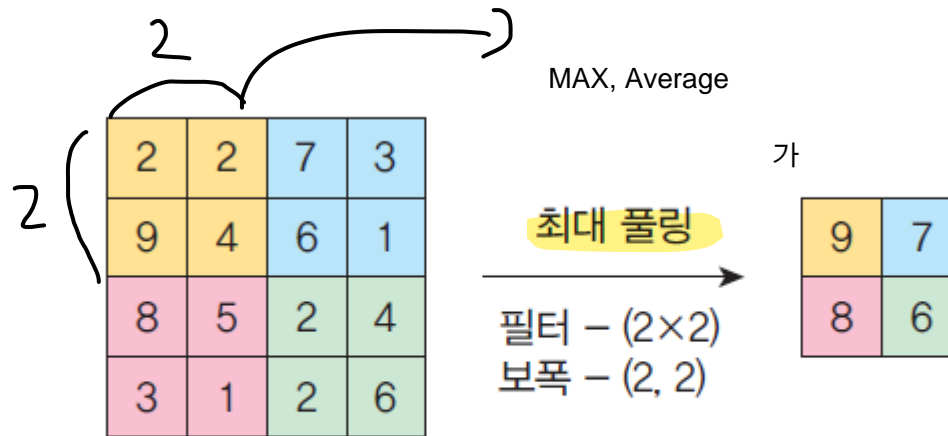


그림 9-17 풀링 레이어



최대 풀링

- 컨벌루션처럼 윈도우를 움직여서 윈도우 안에 있는 숫자 중에서 가장 큰 값만 출력하는 연산이다.

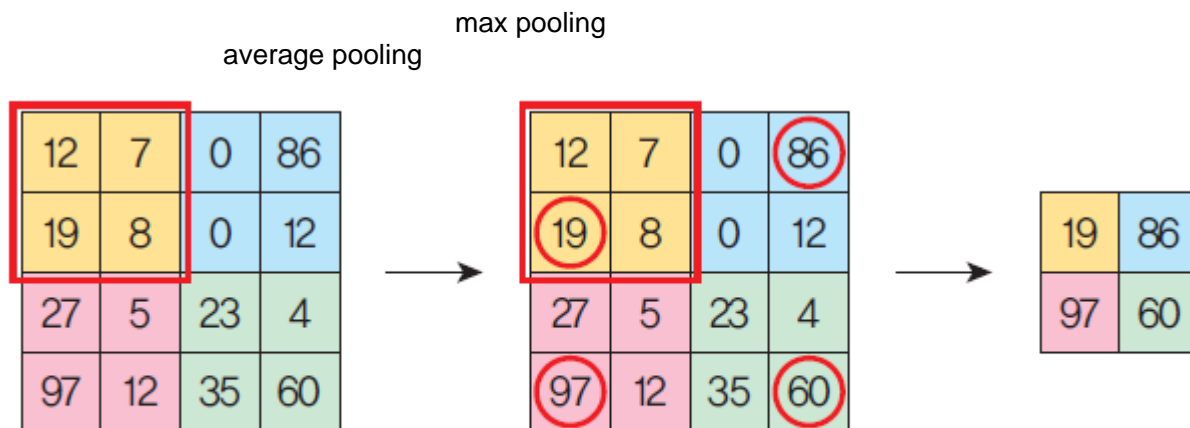


그림 9-18 풀링 연산



풀링의 장점

- 레이어의 크기가 작아지므로 계산이 빨라진다.
- 레이어의 크기가 작아진다는 것은 신경망의 매개변수가 작아진다는 것을 의미한다. 따라서 과적합이 나올 가능성이 줄어든다.
- 공간에서 물체의 이동이 있어도 결과는 변하지 않는다. 즉 물체의 공간이동에 대하여 둔감해지게 된다.



폴링의 종류

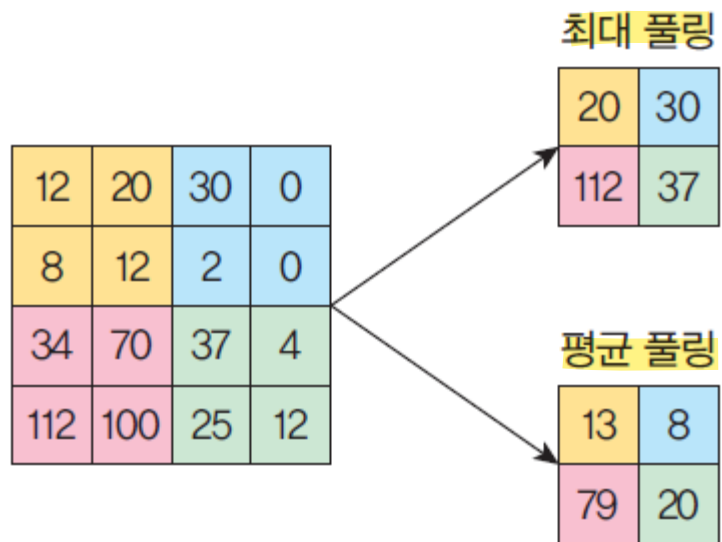


그림 9-19 폴링의 종류



풀링의 장점

- 풀링 계층이 하는 작업 중에서 가장 중요한 것은 **물체의 이동에 대하여 둔감하게 하는 것이다.**



그림 9-20 평행이동된 영상

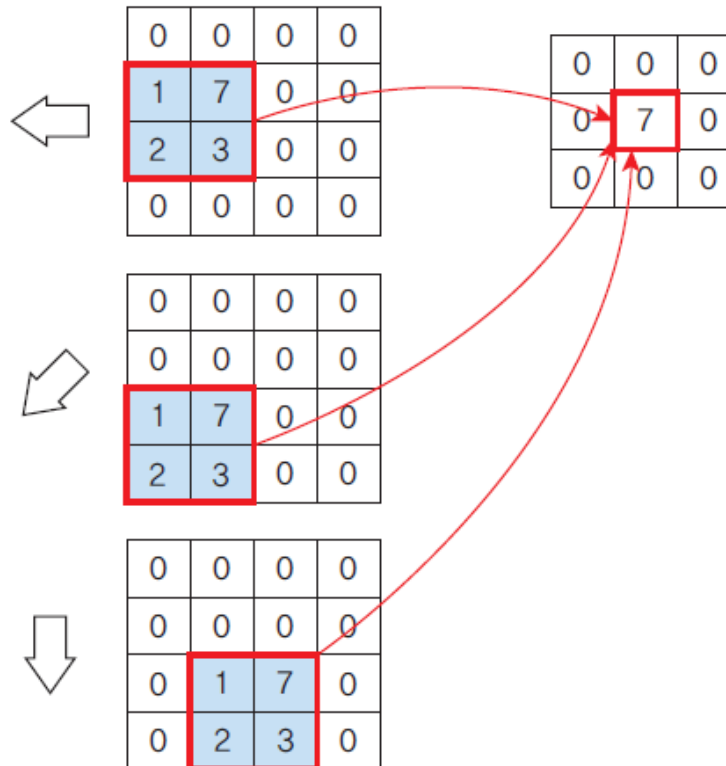


풀링의 장점

Conv / p / conv / p

conv conv p

- 풀링 계층이 하는 작업 중에서 가장 중요한 것은 물체의 이동에 대하여 둔감하게 하는 것이다.





컨벌루션 신경망의 해석

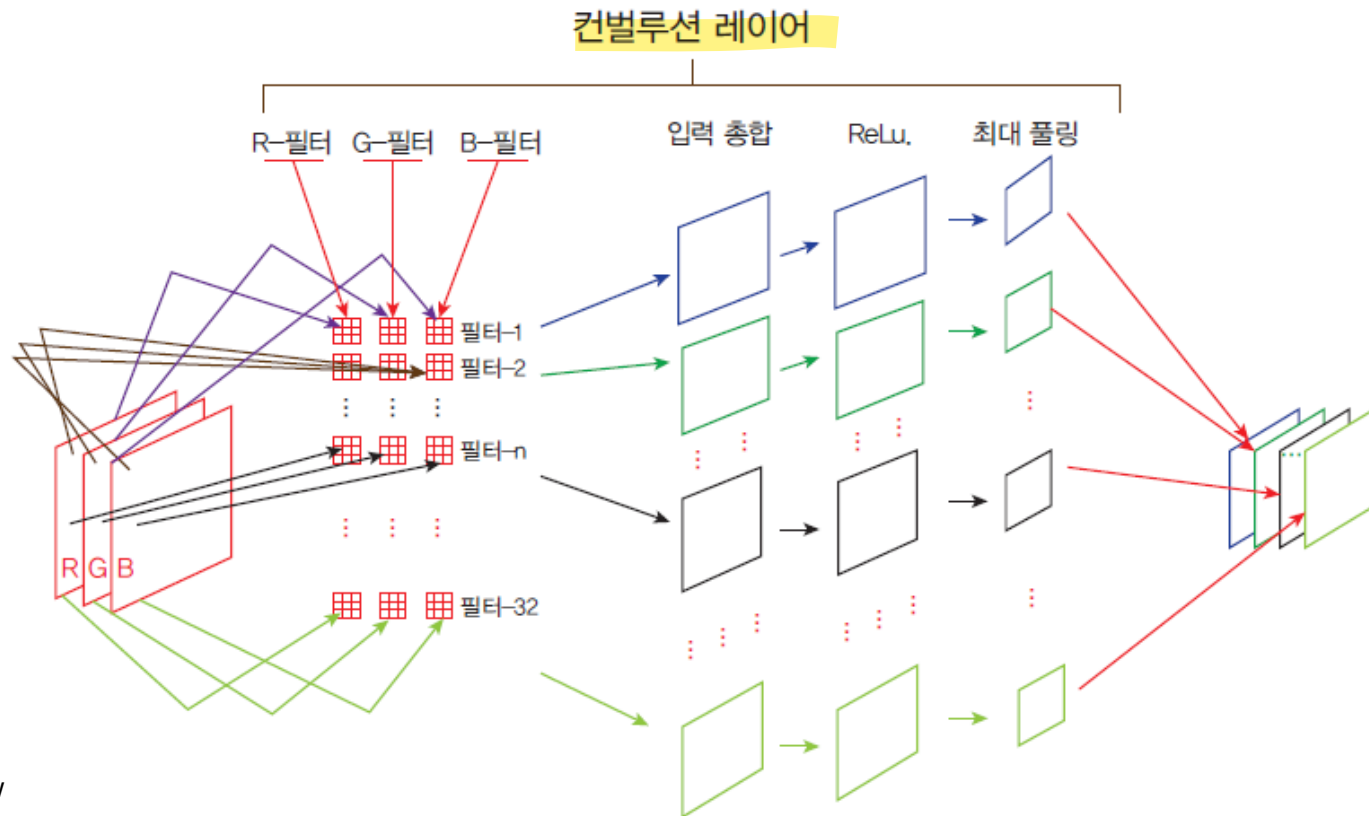
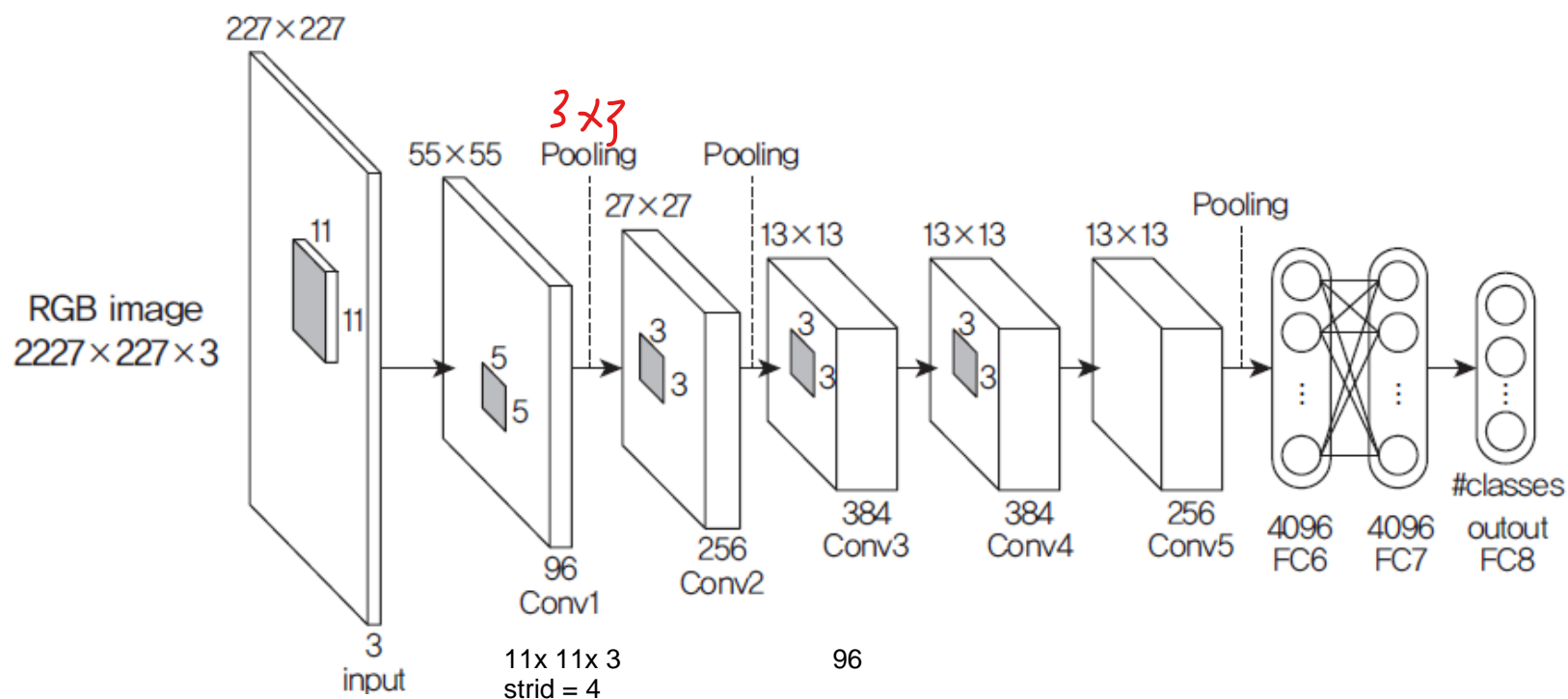


그림 9-21 컨벌루션 레이어의 분석



AlexNet





(1) 첫 번째 레이어 Conv1은 컨볼루션 레이어로 $11 \times 11 \times 3$ 커널을 96개를 사용하고 보폭은 4이고 패딩은 사용하지 않았다. 여기에 ReLu 활성화 함수를 적용하고 이어서, 3×3 겹치는 최대 풀링이 적용된다. 결과적으로 $27 \times 27 \times 96$ 크기의 특징맵이 나오게 된다.

(2) 두 번째 레이어 Conv2는 256개의 $5 \times 5 \times 48$ 크기의 커널을 사용하여 전 단계의 특징맵을 컨볼루션한다. 보폭은 1로, 패딩은 2로 설정한다. 따라서 $27 \times 27 \times 256$ 크기의 특징맵이 얻어진다. 이어서 3×3 최대 풀링을 보폭 2로 시행한다. 최종적으로 $13 \times 13 \times 256$ 특징맵을 얻는다.

(3) 세 번째 레이어, 네 번째 다섯 번째 레이어는 모두 유사하게 처리된다. 즉 보폭과 패딩이 모두 1로 설정된다. 커널의 개수만 달라진다.



케라스로 컨벌루션 신경망 구현하기

	클래스 이름	설명
<u>컨벌루션 레이어</u>	Conv1D, <u>Conv2D</u> , Conv3D, SeparableConv1D, SeparableConv2D, DepthwiseConv2D, Conv2DTranspose, Conv3DTranspose	컨벌루션 연산을 구현하는 레이어이다.
<u>풀링 레이어</u>	MaxPooling1D, <u>MaxPooling2D</u> AveragePooling1D, AveragePooling2D GlobalMaxPooling1D, GlobalMaxPooling2D GlobalAveragePooling1D, GlobalAveragePooling2D	몇 개의 값을 하나로 합치는 레이어이다.



컨벌루션 레이어

• `tf.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), activation=None, input_shape, padding='valid')`

- filters: 필터의 개수이다.
- kernel_size: 필터의 크기이다.
- strides: 보폭이다. x, y
- activation: 유닛의 활성화 함수이다.
- input_shape: 입력 배열의 형상 가 shape가
- padding: 패딩 방법을 선택한다. 디폴트는 “valid”이다.

valid

shape



sample 28 28 (color)

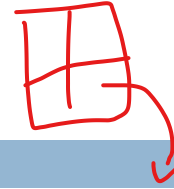
```
shape = (4, 28, 28, 3)
x = tf.random.normal(shape)
y = tf.keras.layers.Conv2D(2, 3, activation='relu', input_shape=shape[1:])(x)
print(y.shape)
```

(4, 26, 26, 2)

가



컨벌루션 레이어



- `tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding="valid")`
x default 1 .

- `pool_size`: 풀링 윈도우의 크기, 정수 또는 2개 정수의 튜플이다. (2, 2)라면 2x2 풀링 윈도우에서 최대값을 추출한다.
- `strides`: 보폭, 각 풀링 단계에 대해 풀링 윈도우가 이동하는 거리를 지정한다.
- `padding`: "valid"나 "same" 중의 하나이다. "valid"는 패딩이 없음을 의미한다. "same"은 출력이 입력과 동일한 높이 / 너비 치수를 갖도록 입력의 왼쪽 / 오른쪽 또는 위 / 아래에 균일하게 패딩한다.



```
x = tf.constant([[1., 2., 3.], [4., 5., 6.], [7., 8., 9.]])
x = tf.reshape(x, [1, 3, 3, 1])
max_pool_2d = tf.keras.layers.MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
                                             padding='valid')
print(max_pool_2d(x))
```

tf.Tensor(
[[[5.]

6.]
6.]

[[8.]
[9.]
[9.]

[[8.]
[9.]
[9.]]]], shape=(1, 3, 3, 1), dtype=float32)

		1	2	3
		4	5	6
		7	8	9



		1	2	3
		4	5	6
		7	8	9



		1	2	3
		4	5	6
		7	8	9

결과가 맞는지
확인해봅시다.



예제: MNIST 필기체 숫자 인식

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

(train_images, train_labels), (test_images, test_labels) = datasets.mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1))
test_images = test_images.reshape((10000, 28, 28, 1))

# 픽셀 값을 0~1 사이로 정규화한다.
train_images, test_images = train_images / 255.0, test_images / 255.0
```



예제: MNIST 필기체 숫자 인식

```
model = models.Sequential()
```

```
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
```

```
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
model.add(layers.Flatten())
```

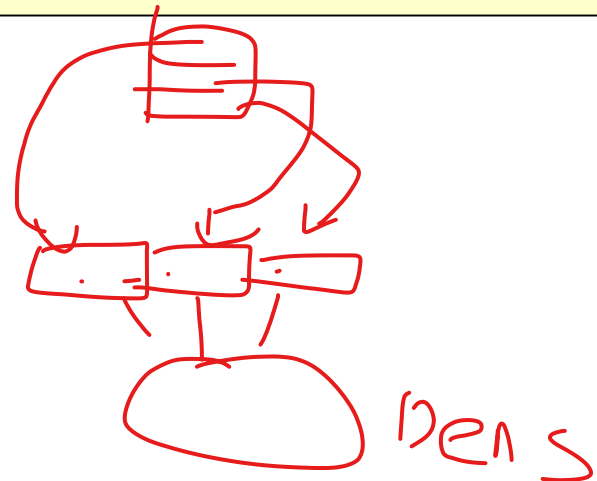
2D

→ 1D

```
model.add(layers.Dense(64, activation='relu'))
```

```
model.add(layers.Dense(10, activation='softmax'))
```

conv layer





예제: MNIST 필기체 숫자 인식

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
--------------	--------------	---------

conv2d_3 (Conv2D)	(None, 26, 26, 32)	320
-------------------	--------------------	-----

max_pooling2d_2 (MaxPooling2)	(None, 13, 13, 32)	0
-------------------------------	--------------------	---

conv2d_4 (Conv2D)	(None, 11, 11, 64)	18496
-------------------	--------------------	-------

max_pooling2d_3 (MaxPooling2)	(None, 5, 5, 64)	0
-------------------------------	------------------	---

conv2d_5 (Conv2D)	(None, 3, 3, 64)	36928
-------------------	------------------	-------



예제: MNIST 필기체 숫자 인식





예제: MNIST 필기체 숫자 인식

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
model.fit(train_images, train_labels, epochs=5)
```

```
Epoch 1/5  
1875/1875 [=====] - 14s 7ms/step - loss: 0.1414  
-  
accuracy: 0.9560  
...  
Epoch 5/5  
1875/1875 [=====] - 14s 7ms/step - loss: 0.0194  
-  
accuracy: 0.9940
```



Summary

- 영상 인식에 많이 사용되는 신경망은 컨볼루션 인공신경망(컨벌루션 신경망)이다. 컨볼루션 신경망(컨벌루션 신경망)은 동물의 조직에서 영감을 얻어서 만들어진 신경망이다.
- 컨볼루션(Convolution Neural Network: 컨벌루션 신경망) 신경망에서는 하위 레이어의 유닛들과 상위 레이어의 유닛들이 부분적으로만 연결되어 있다. 따라서 복잡도가 낮아지고 과대 적합에 빠지지 않는다.
- 컨벌루션은 주변 화소값들에 가중치를 곱해서 더한 후에 이것을 새로운 화소값으로 하는 연산이다.
- 풀링(Pooling)이란 서브 샘플링이라고도 하는 것으로 입력 데이터의 크기를 줄이는 것이다.

가 : stride 가