CNN

23012127 인공지능학과 이지민

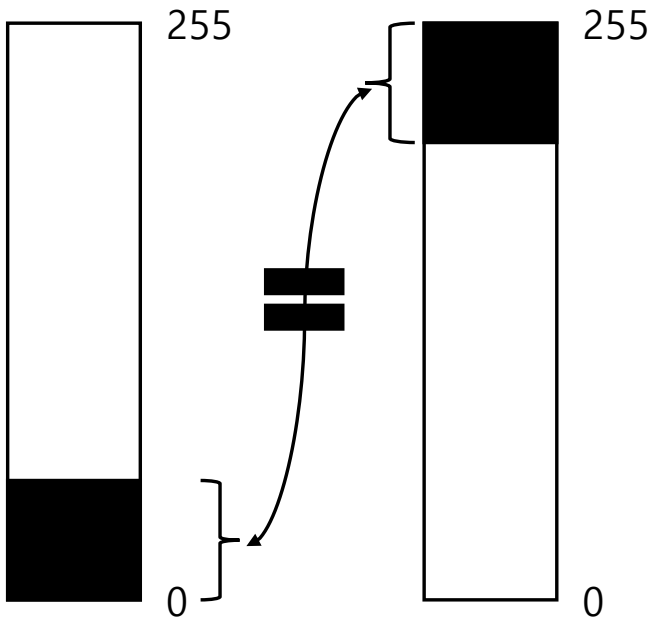
Elementwise VS matrix

$$\begin{bmatrix} 3 & 5 & 7 \\ 4 & 9 & 8 \end{bmatrix} \circ \begin{bmatrix} 1 & 6 & 3 \\ 0 & 2 & 9 \end{bmatrix} = \begin{bmatrix} 3\times1 & 5\times6 & 7\times3 \\ 4\times0 & 9\times2 & 8\times9 \end{bmatrix}$$

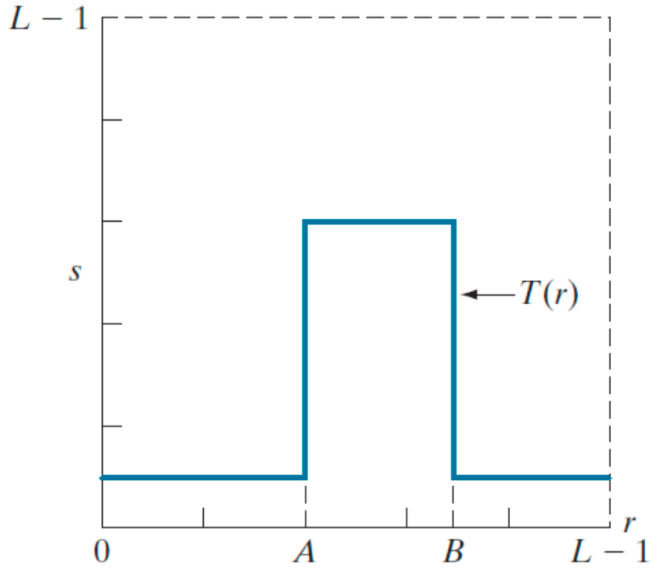
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \bullet \begin{bmatrix} w & x \\ y & z \end{bmatrix} = \begin{bmatrix} aw+by & ax+bz \\ cw+dy & cx+dz \end{bmatrix}$$

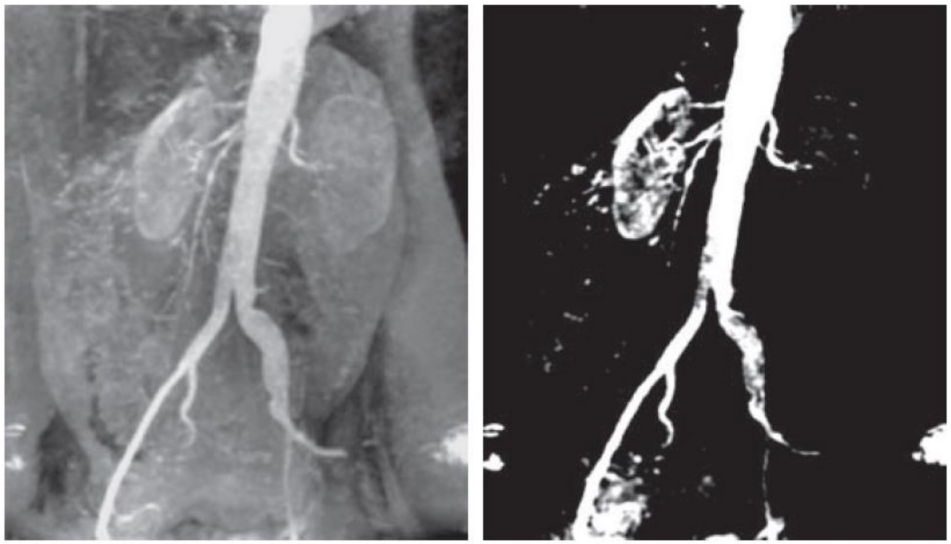
Image negatives

$$y = 255 - x$$

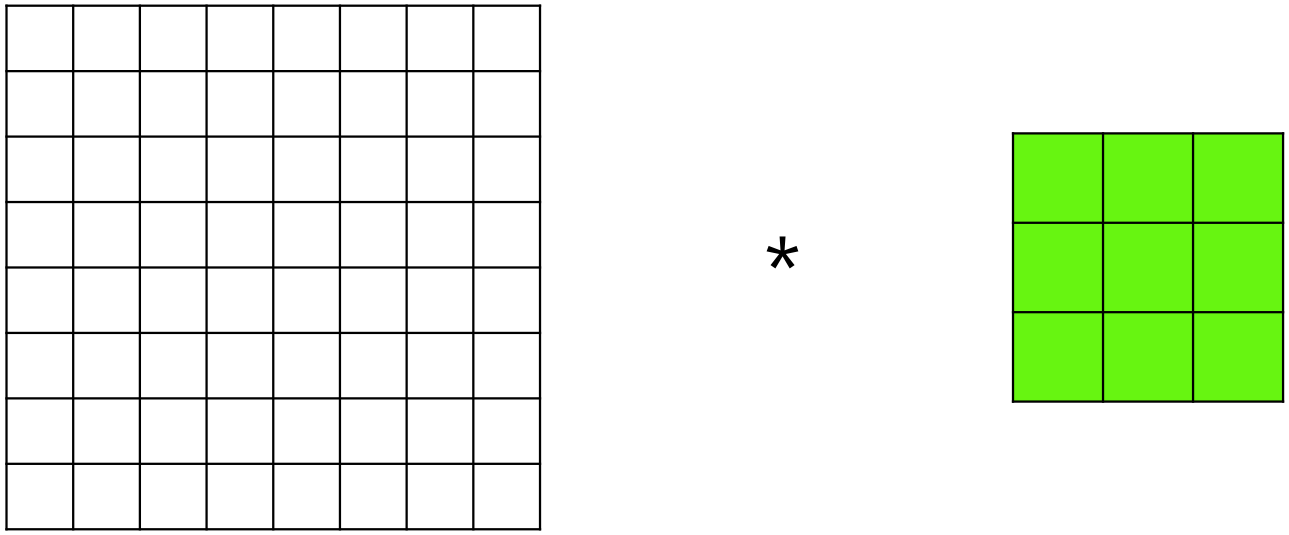


Intensity Level Scaling

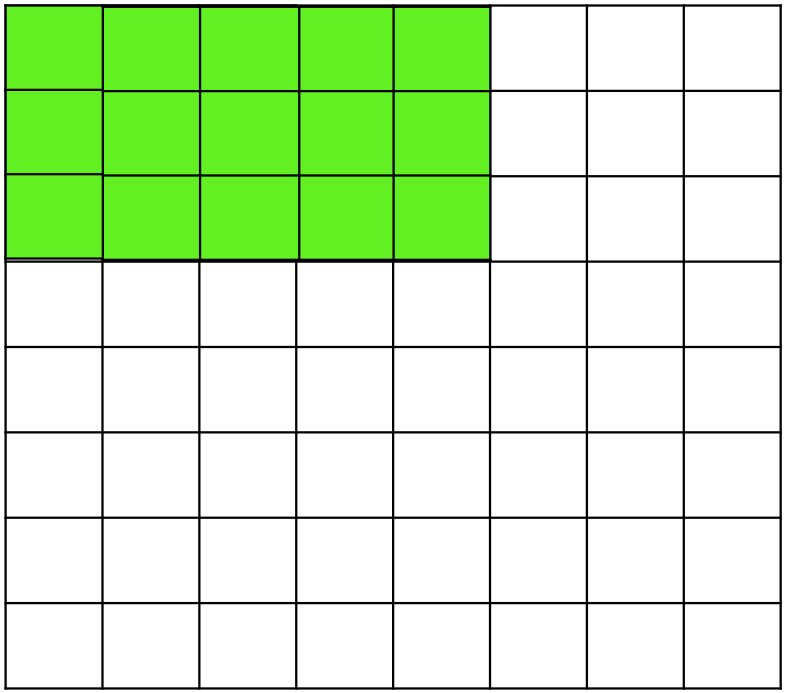




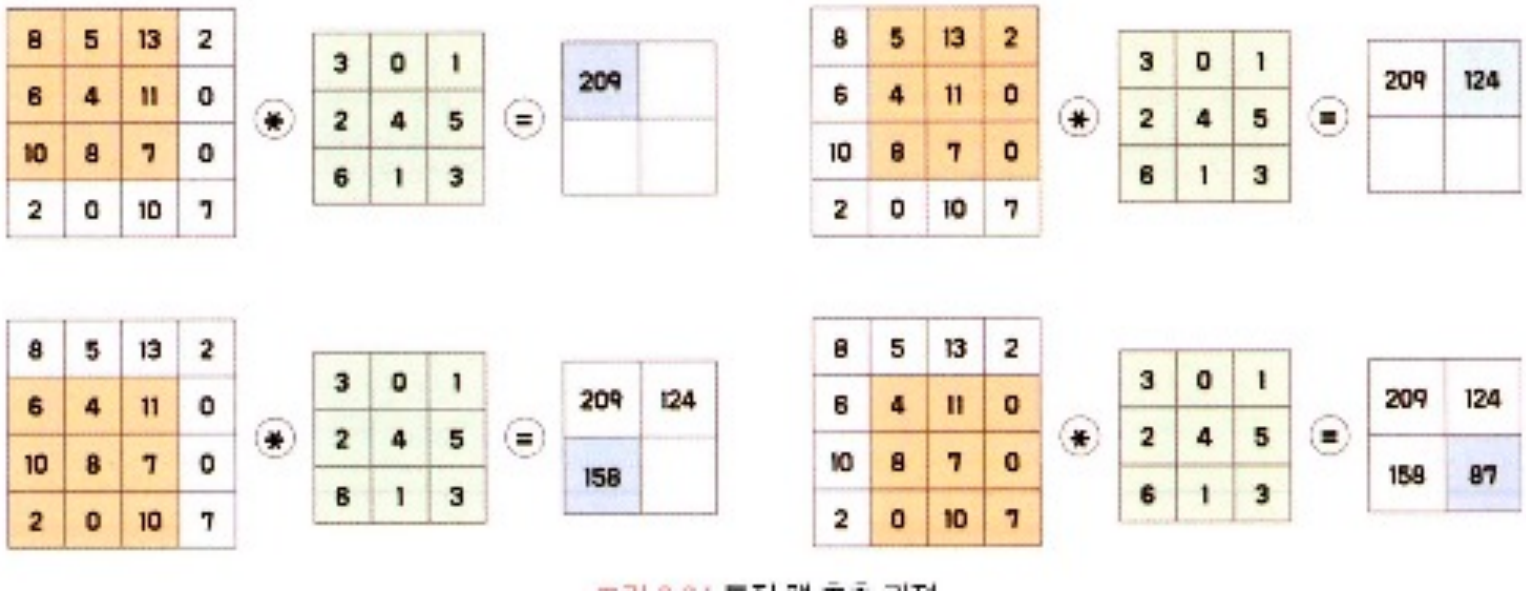
Convolution



Convolution

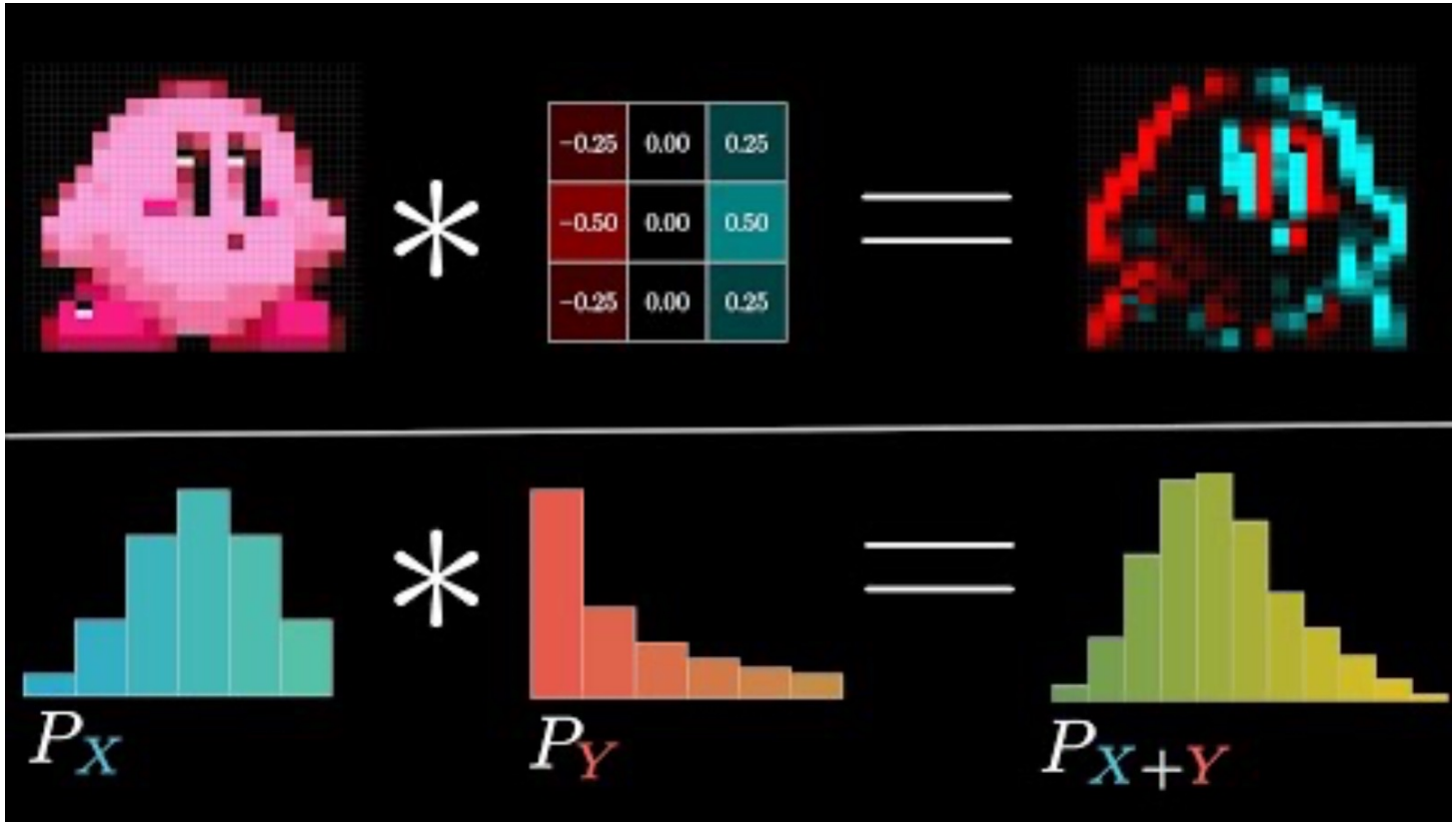


Convolution



p.337

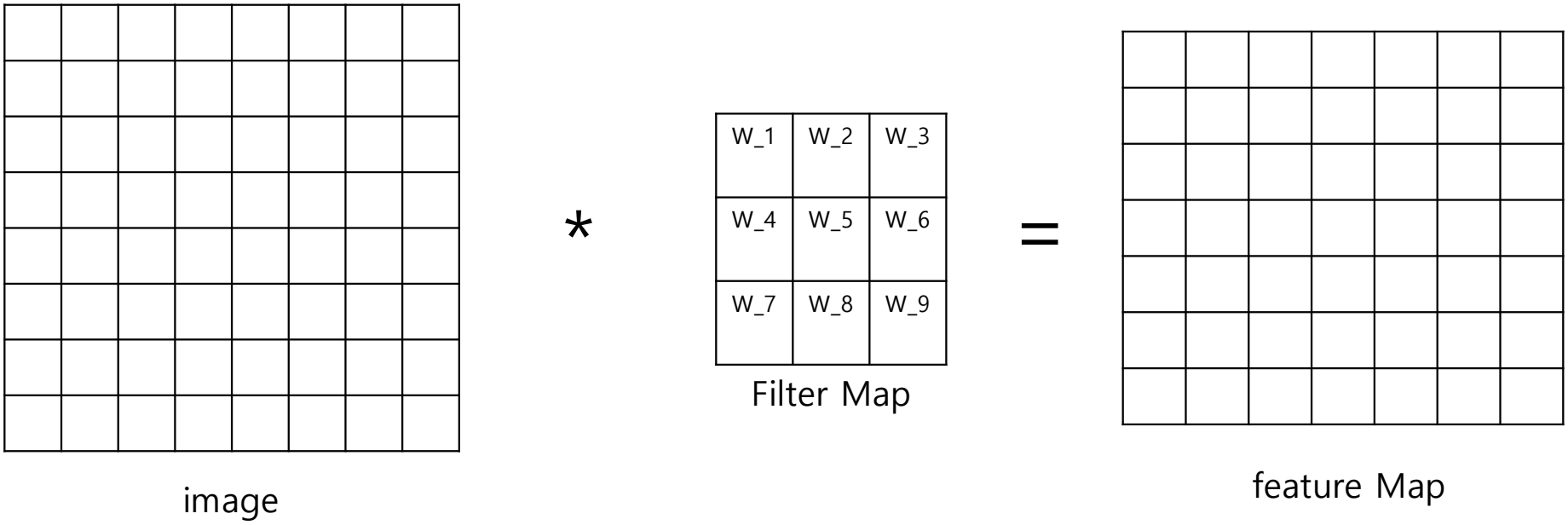
Convolution



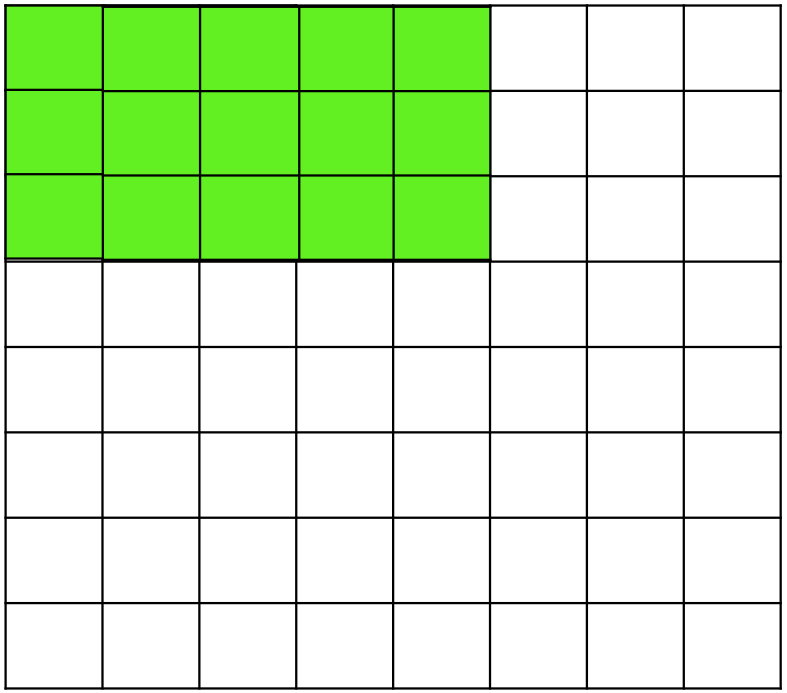
- https://www.youtube.com/watch?v=KuXjwB4LzSA

Convolutional Layer

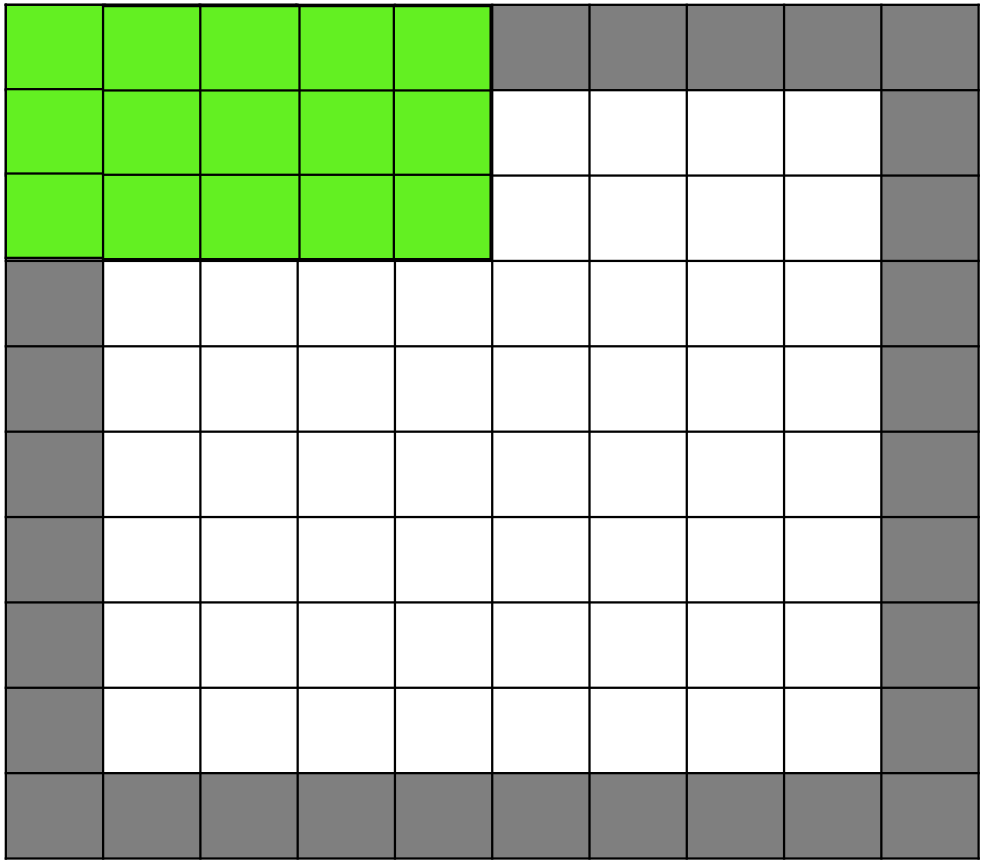
- Convolution을 시행



Convolutional Layer



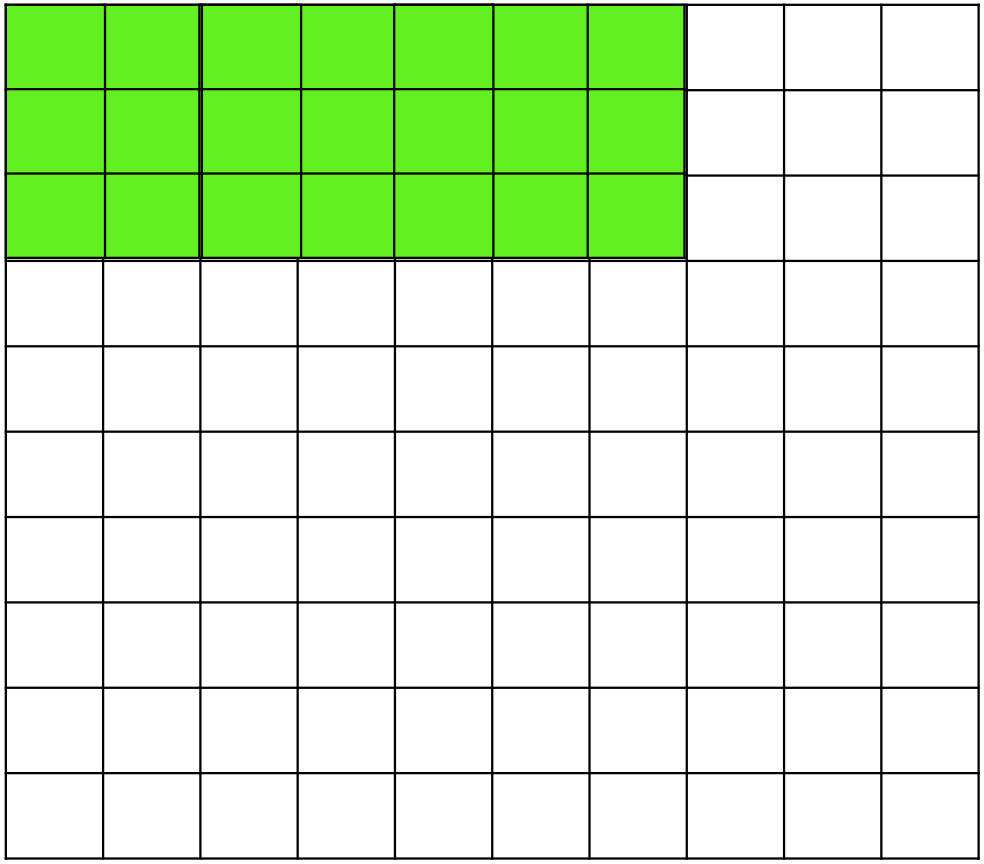
Convolutional Layer



- Size decrease 문제
- 가장자리의 학습이 제한

➢ 가장자리에 padding 추가

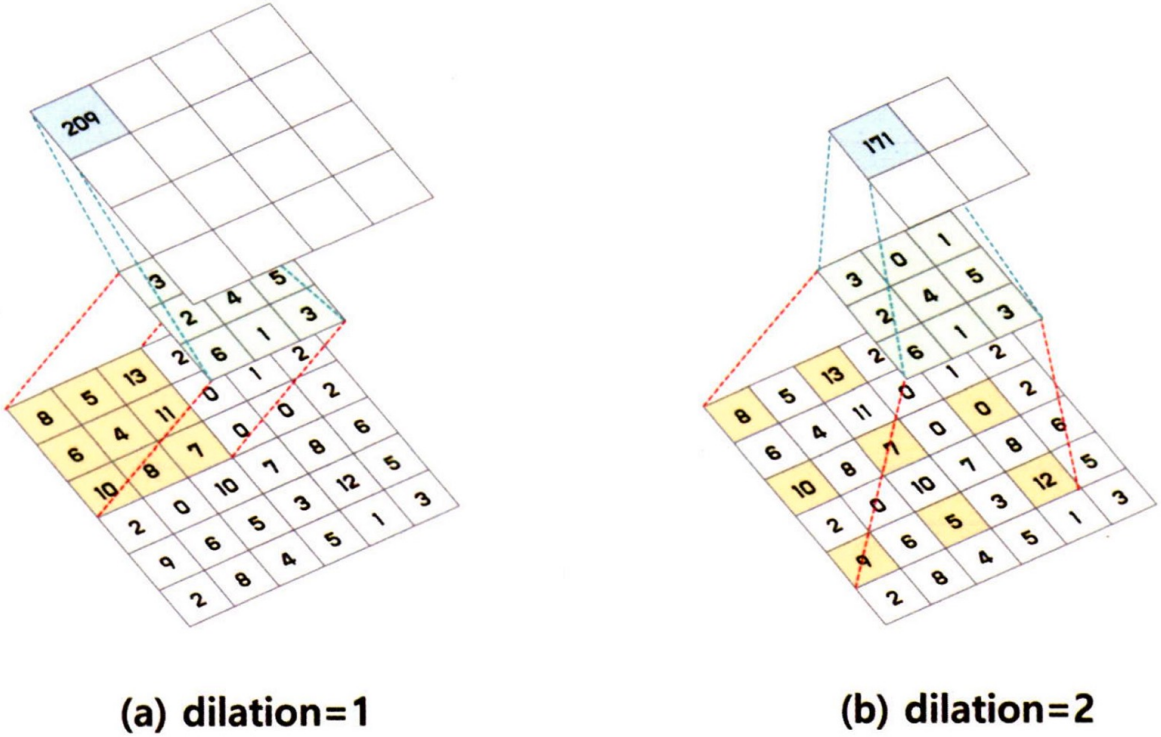
Convolutional Layer



- Stride 크기 조절

➢ 매개변수의 수 감소
➢ 모델 복잡도를 줄임
➢ 과대적합 방지

Convolutional Layer



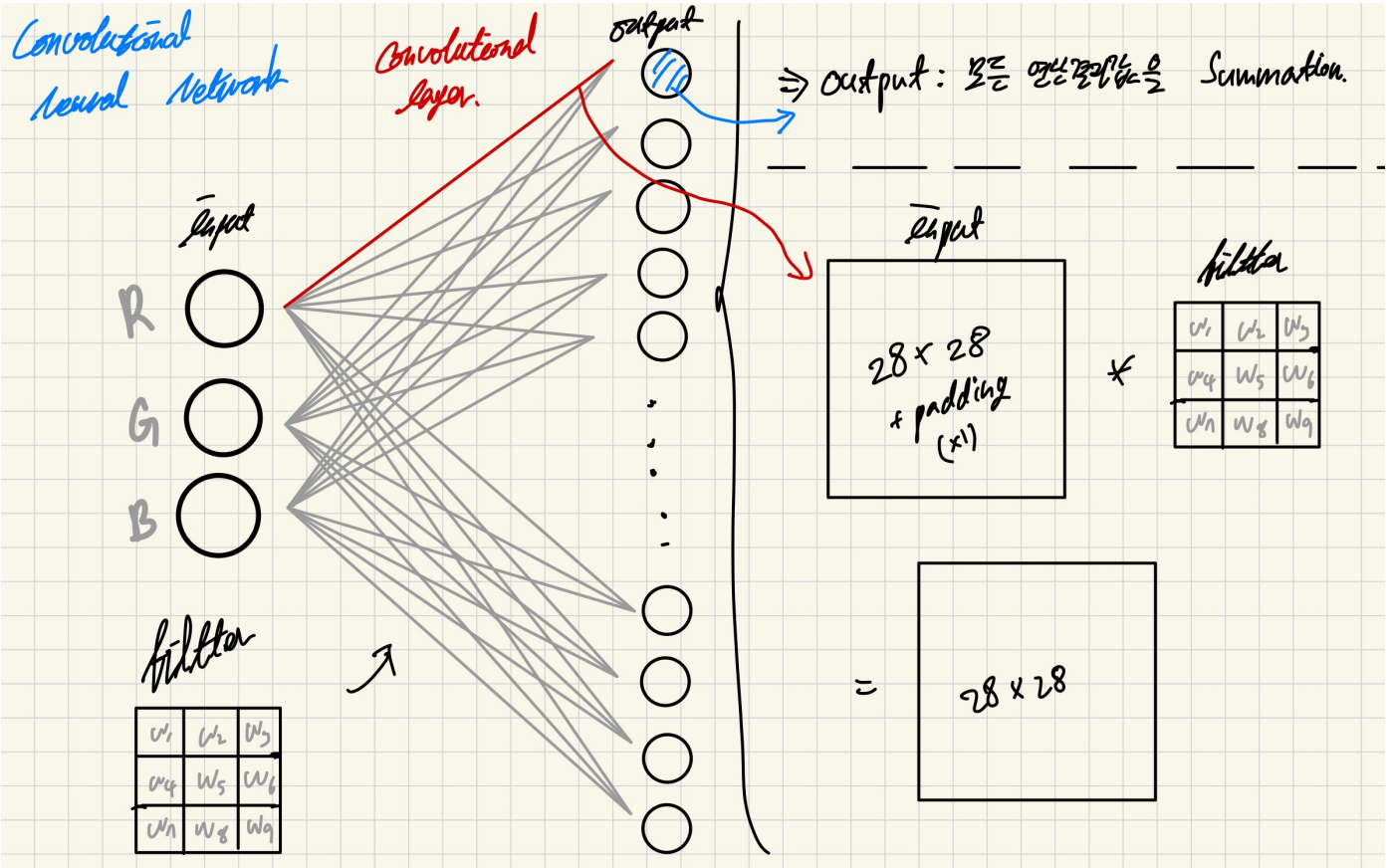
(a) dilation=1

(b) dilation=2

- dilation (팽창)

➢ 필터의 크기를 줄이지 않고 더 넓은 영역을 고려

Convolutional Layer



Pooling layer

그림 6.27 최댓값 풀링

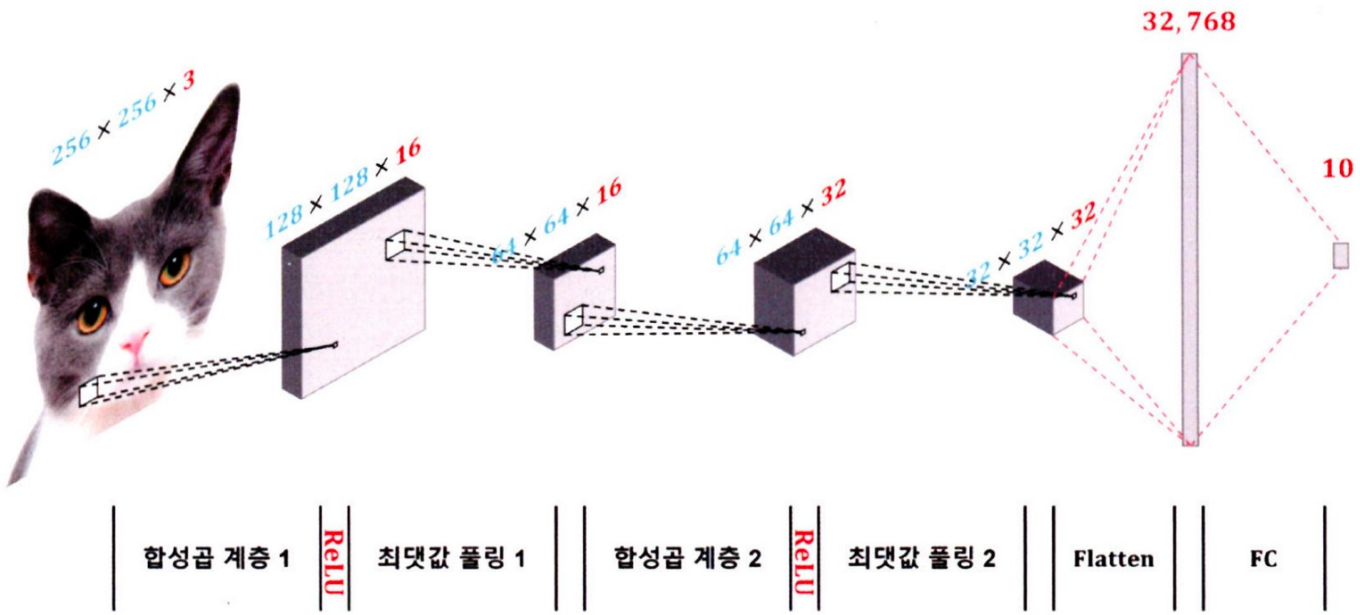
- pooling

➢ 연산량 감소
➢ 정보 압축

최댓값 폴링

평균값 폴링

Convolutional Neural Network



pytorch

- 2차원 convolutional layer
https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1 \right\rfloor$$

Convolutional layer output size

- 2차원 max pooling layer
https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html#torch.nn.MaxPool2d

- 2차원 average pooling layer
https://pytorch.org/docs/stable/generated/torch.nn.AvgPool2d.html#torch.nn.AvgPool2d

$$L_{out} = \left\lfloor \frac{L_{in} + 2 \times padding - kernel\_size}{stride} + 1 \right\rfloor$$

pooling layer output size

pytorch

```python
import torch
from torch import nn

class CNN(nn.Module):
    def __init__(self):
        super().__init__()

        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=3, out_channels=16, kernel_size=3, stride=2, padding=1
            ),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )

        self.conv2 = nn.Sequential(
            nn.Conv2d(
                in_channels=16, out_channels=32, kernel_size=3, stride=1, padding=1
            ),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )

        self.fc = nn.Linear(32 * 32 * 32, 10)

    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = torch.flatten(x)
        x = self.fc(x)
        return x
```