

Preclass 15: Image Convolution

[SCS4049] Machine Learning and Data Science

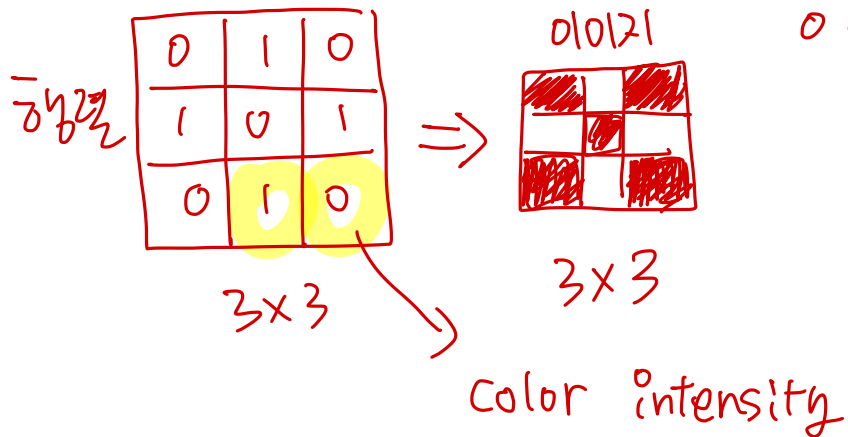
Seongsik Park (s.park@dgu.edu)

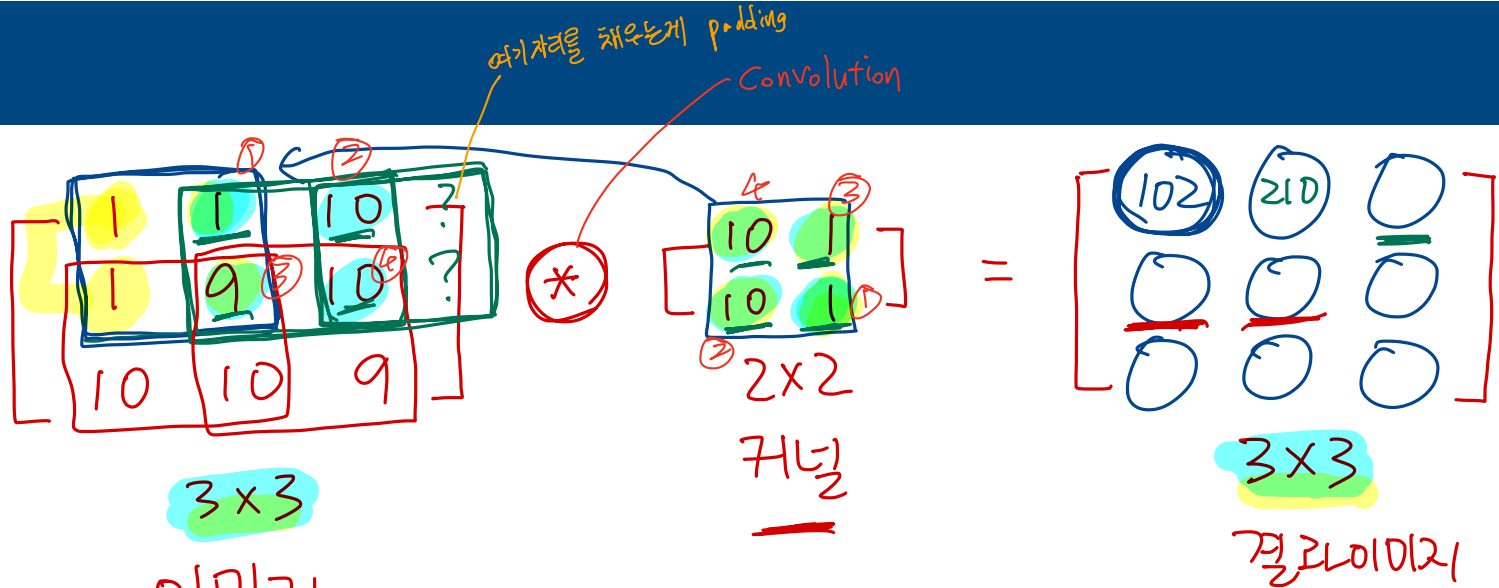
School of AI Convergence & Department of Artificial Intelligence, Dongguk University

흑백 이미지 = 2차원 배열

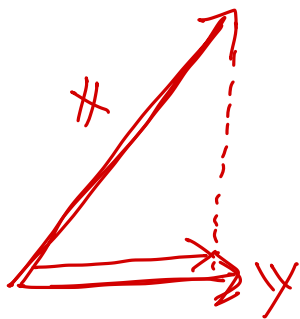
0 ~ 1 실수.
검은색 흰색.

0 ~ 255 정수.





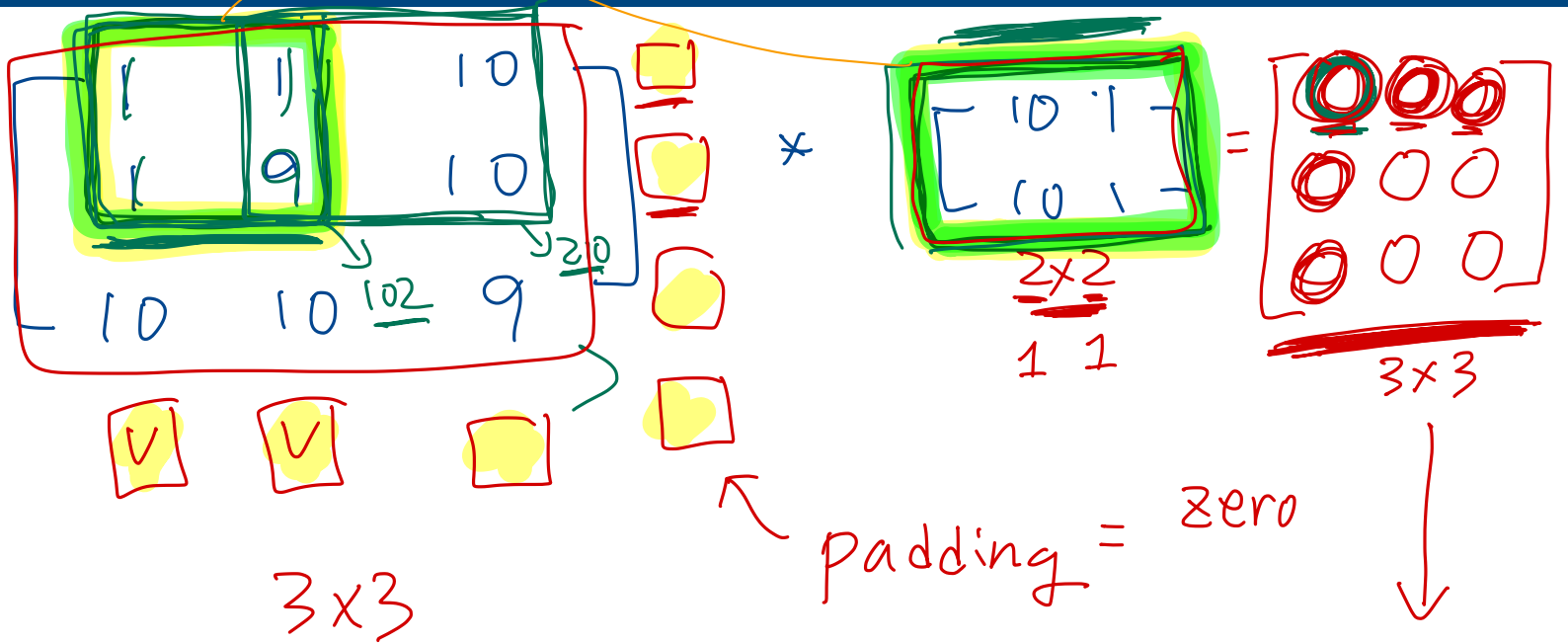
$$\underline{1 \cdot 1} + \underline{10 \cdot 10} + \underline{9 \cdot 1} + \underline{10 \cdot 10} = 210$$



$$x^T y = \sum x_i y_i$$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 1 \cdot 1 + 2 \cdot 0 + 3 \cdot 0$$

의미: 이 자리에 이 component가 얼마나 들어있는가



이거 상하 좌우 padding 해서 내걸어야 돼 (여기서도 좌우 padding)


$$\begin{bmatrix} 0 & 1 & 2 & 3 & 0 \\ 0 & 2 & 3 & 4 & 0 \\ 0 & 3 & 4 & 5 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}_{1 \times 3} = \begin{bmatrix} 2 & 4 & 2 \\ 3 & 6 & 3 \\ 4 & 8 & 4 \end{bmatrix}_{3 \times 3}$$

3x3

padding 의 크기 \leftarrow kernel 의 크기 - 1

이냥이면 양 옆, 위, 아래를 대칭되는 개수

$$\begin{bmatrix} 1 & 1 & 10 & 10 \\ 1 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 \end{bmatrix} * \begin{bmatrix} -1 & 1 \end{bmatrix}_{1 \times 2} = \begin{bmatrix} 0 & -9 & 0 \\ -9 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Clamp

① convolution 계산하는 법

② padding 필요, 얼마큼 \leftarrow kernel의 크기.

③ padding 종류,

④ convolution 결과 \rightarrow feature map
height map

Convolution

Given a filter kernel \mathcal{H} , the convolution of the kernel with image \mathcal{F} is an image \mathcal{R} . The (i, j) -th component of \mathcal{R} is given by

$$R_{ij} = \sum_{u,v} H_{i-u, j-v} F_{uv}. \quad (1)$$

- **kernel** of the filter: the pattern of weights used for a linear filter
- **convolution**: the process of applying the filter

Linear filter and convolution

This operation is called **convolution**

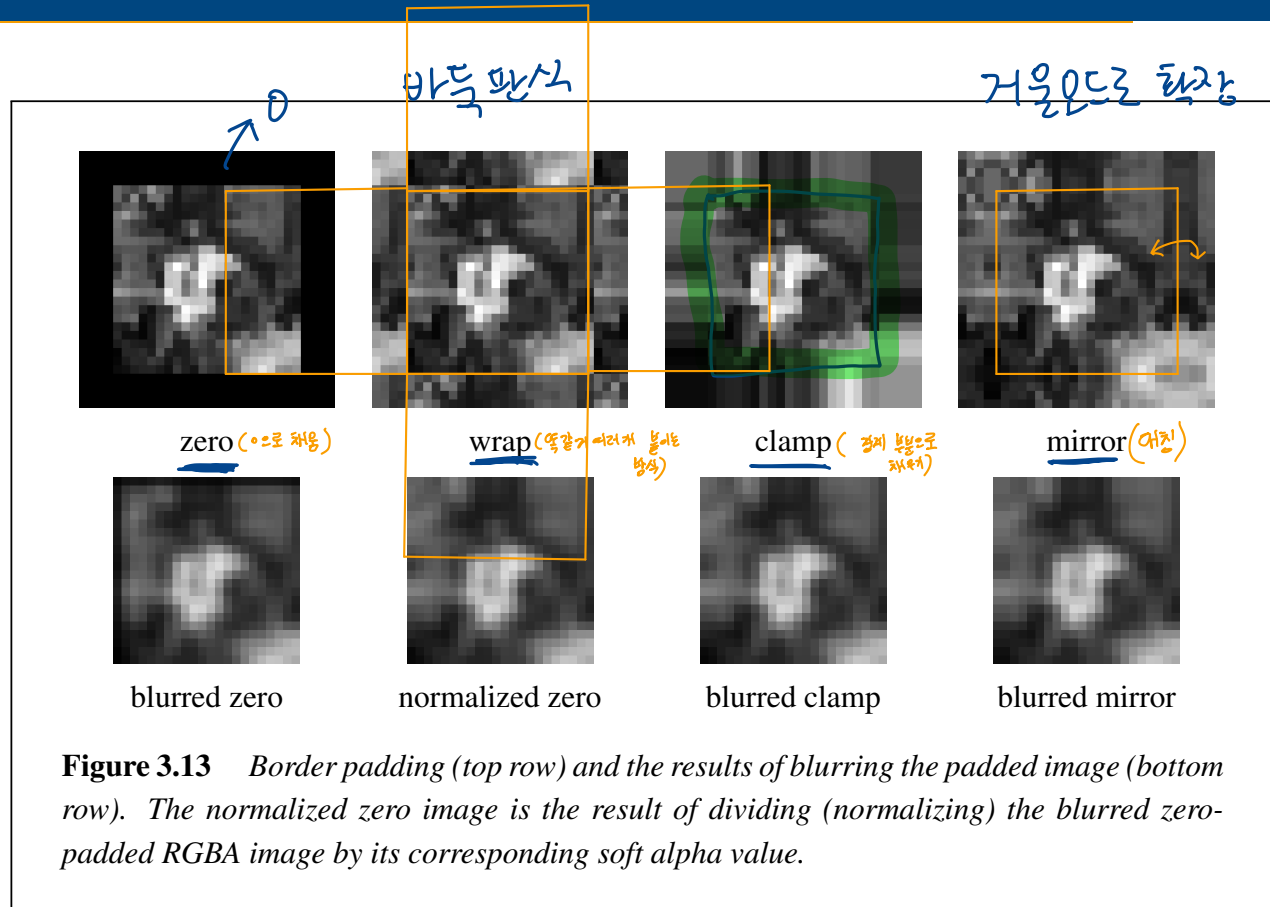
$$R(f) = (h * f) \tag{2}$$

- *commutative*: $(g * h)(x) = (h * g)(x)$
- *associative*: $f * (g * h) = (f * g) * h$
- *distributive*: $f * (g + h) = f * g + f * h$

Padding (border effects)

- **zero**: set all pixels outside the source image to 0
- **constant**: set all pixels outside the source image to a specified border value
- **clamp**: repeat edge pixels indefinitely
- **wrap**: loop “around” the image in a “toroidal” configuration
- **mirror**: reflect pixels across the image edge
- **extend**: extend the signal by subtracting the mirrored version of the signal from the edge pixel value

Padding (border effects)



Examples of linear filter

순정방향=군 정방향
(구하는 것)



$$\frac{1}{K^2}$$

1	1	...	1
1	1	...	1
⋮	⋮	1	⋮
1	1	...	1

$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$$\frac{1}{256}$$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

$$\frac{1}{8}$$

3x3

-1	0	1
-2	0	2
-1	0	1

$$\frac{1}{4}$$

1	-2	1
-2	4	-2
1	-2	1

$$\frac{1}{K}$$

1	1	...	1
---	---	-----	---

$$\frac{1}{4}$$

1	2	1
---	---	---

$$\frac{1}{16}$$

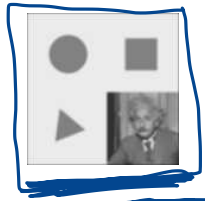
1	4	6	4	1
---	---	---	---	---

$$\frac{1}{2}$$

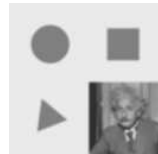
-1	0	1
----	---	---

$$\frac{1}{2}$$

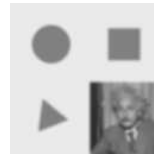
1	-2	1
---	----	---



(a) box, $K = 5$



(b) bilinear



(c) "Gaussian"



(d) Sobel



(e) corner

Figure 3.14 Separable linear filters: For each image (a)–(e), we show the 2D filter kernel (top), the corresponding horizontal 1D kernel (middle), and the filtered image (bottom). The filtered Sobel and corner images are signed, scaled up by $2\times$ and $4\times$, respectively, and added to a gray offset before display.