Convolutional Neural Networks Basics

박달님 07.19.21

Computer Vision Problems

- Image classification
- Object detection
- Neural STYLE Transfer (Repaint the content image to a style image)

Deep Learning on Large Images

- Challenges of computer vision problems: inputs can get really big

- Ex) worked on 64 x 64 x 3 images - three color channels

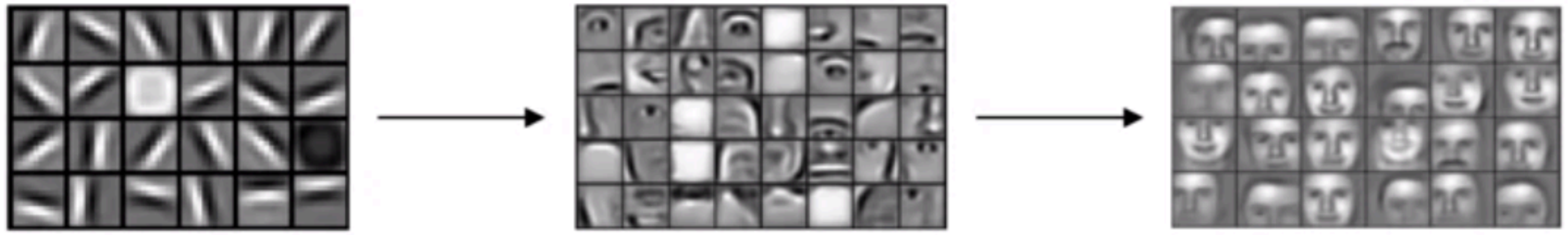
- x has input features 12288 —> but this is a small image

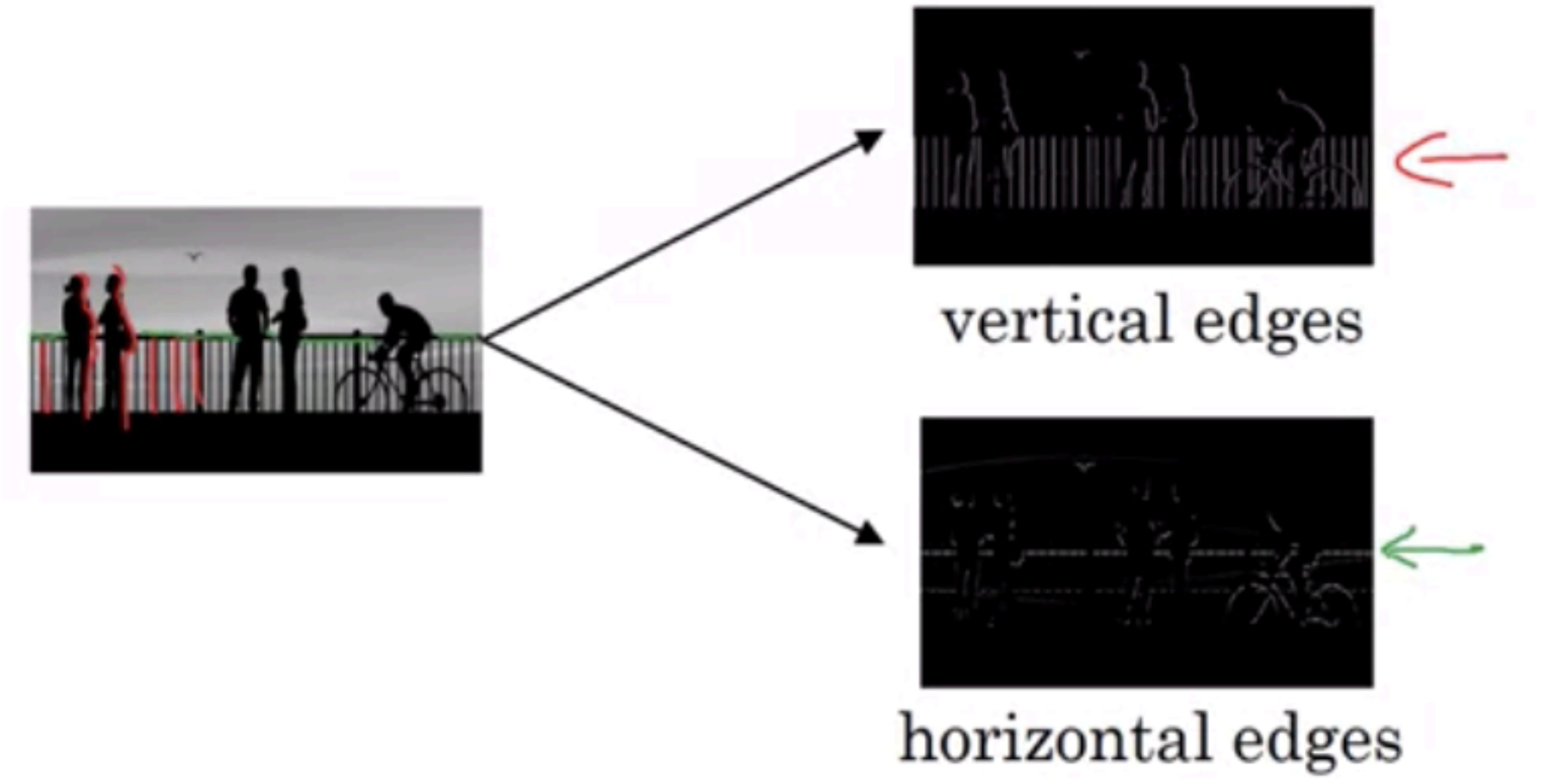
- Ex) 1000 x 1000 x 3 = 3 billion parameters

- Difficult to get enough data to prevent a neural network from overfitting

Edge Detection Example

- Convolution operation
- Early layers of neural network - detect edges
- Later layers - detect partial objects
- Even later layers - complete objects
- 1. Detect vertical edges
- 2. Detect horizontal edges



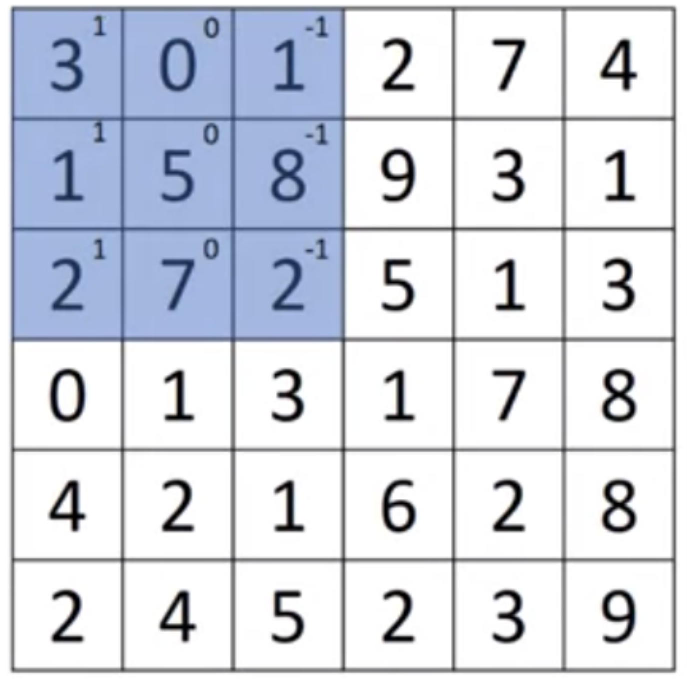


Vertical Edge Detection

- 6 x 6 x 1 Gray scale image
- Construct 3 x 3 filter (kernel)
- Take 3 x 3 filter and paste it on top of the 3 x 3 region of original input image
- Take filter and shift it to the right

“convolution”

$$\begin{array}{|c|c|c|c|c|c|}\hline 3 & 0 & 1 & 2 & 7 & 4 \\ \hline 1 & 5 & 8 & 9 & 3 & 1 \\ \hline 2 & 7 & 2 & 5 & 1 & 3 \\ \hline 0 & 1 & 3 & 1 & 7 & 8 \\ \hline 4 & 2 & 1 & 6 & 2 & 8 \\ \hline 2 & 4 & 5 & 2 & 3 & 9 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline -5 & -4 & 0 & 8 \\ \hline -10 & -2 & 2 & 3 \\ \hline 0 & -2 & -4 & -7 \\ \hline -3 & -2 & -3 & -16 \\ \hline \end{array}$$



3^{1}	0^{0}	1^{-1}	2	7	4
1^{1}	5^{0}	8^{-1}	9	3	1
2^{1}	7^{0}	2^{-1}	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

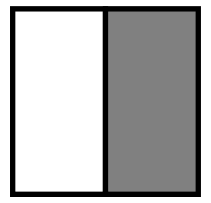
Add up the 9 numbers = -5

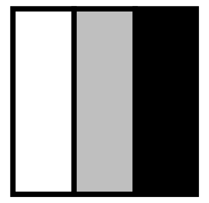
Vertical Edge Detection

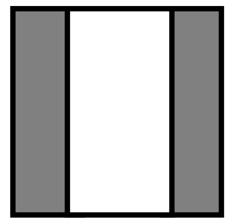
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

Vertical Edge Detection Example

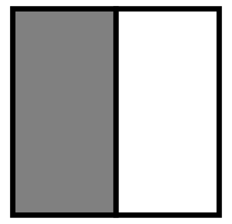
$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

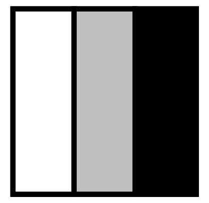


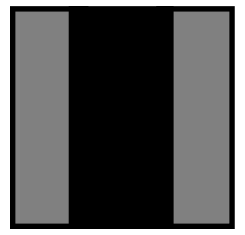




$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \\ 0 & -30 & -30 & 0 \end{bmatrix}$$







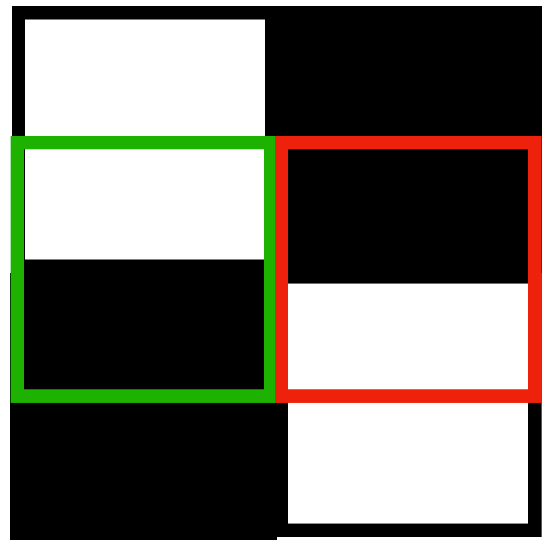
Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal



$$\begin{array}{|c|c|c|c|c|c|}\hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline \boxed{30} & 10 & -10 & \boxed{-30} \\ \hline 30 & 10 & -10 & -30 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array}$$

Different Kinds of Filters

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

1	0	-1
2	0	2
1	0	-1

Sobel filter
Puts more weight to the central pixel
Makes it more robust

3	0	-3
10	0	-10
3	0	-3

Schorr filter
For vertical edge detection
(flip 90 degrees for horizontal detection)

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

W1	W2	W3
W4	W5	W6
W7	W8	W9

Don’t need to pick, make it parameters
Learn using back propagation

Goal: learn 9 parameters
Take image, convolve with 3 x 3 filter
-> good edge detector

Underlying convolution operation

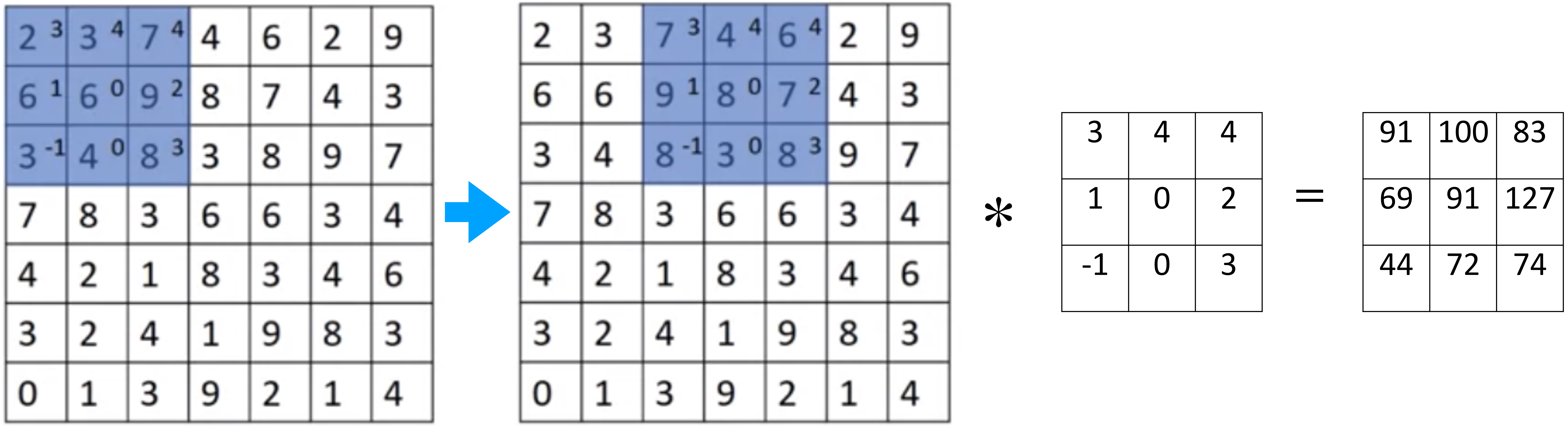
Padding

- Modification to the basic convolutional operation
- $n \times n * f \times f = (n-f+1) \times (n-f+1)$
- Dimension of the output shrinks & counting less information from edges of the image (pixels in corners less in output)
- Solution: before applying convolutional operation, pad the image with an additional one border
- Ex) Pad with zeros, one pixel all around. P = padding = 1
- $(n + 2p - f +1) \times (n + 2p - f +1)$

Valid and Same Convolutions

- “Valid”: no padding. n x n * f x f = (n-f+1) x (n-f+1)
- “same”: pad so that output size is the **same** as the input size.
  - (n+2p-f+1) x (n+2p-f+1)
  - P = (f-1)/2
  - F is usually odd: it has central pixel, symmetric padding

Strided Convolutions



2^{3}	3^{4}	7^{4}	4	6	2	9
6^{1}	6^{0}	9^{2}	8	7	4	3
3^{-1}	4^{0}	8^{3}	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

2	3	7^{3}	4^{4}	6^{4}	2	9
6	6	9^{1}	8^{0}	7^{2}	4	3
3	4	8^{-1}	3^{0}	8^{3}	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

*

3	4	4
1	0	2
-1	0	3

=

91	100	83
69	91	127
44	72	74

Stride = 2

If (n + 2p - f)/s + 1 not an integer, round it down.
The filter must lie entirely within the image (+padding)

Output size

n x n * f x f = (n + 2p - f)/s + 1 + (n + 2p - f)/s + 1
Image Filter Padding Stride