Building Image Processing Applications Using scikit-image

WORKING WITH IMAGE DATA

Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Working with images is increasingly important

Deep learning algorithms and AI rely on images and videos as input

scikit-image is an image processing toolkit

Powerful, versatile, well-implemented

Prerequisites and Course Outline

Prerequisite Courses

Python: Getting Started

Python Fundamentals

Working with Multidimensional Data Using NumPy

Software and Skills

Basic understanding of Python programming

Python 3, NumPy, Matplotlib

Working with Jupyter notebooks

Basic knowledge of statistics

Course Outline

Working with images

- Manipulating images as arrays of pixels
- Detecting contours
- Finding convex hull

Detecting features and objects

- Extracting features
- Detecting corners
- Denoising images

Segmentation and transformation

- Local and global thresholding
- Image segmentation
- Image transformations

Introducing scikit-image

scikit-image

Image processing toolbox for SciPy; high-quality, peer-reviewed code, available free of charge and free of restriction

scikit-image

Simple Python APIs for complex image processing tasks

Gallery of examples of image manipulation techniques

Open source community of developers

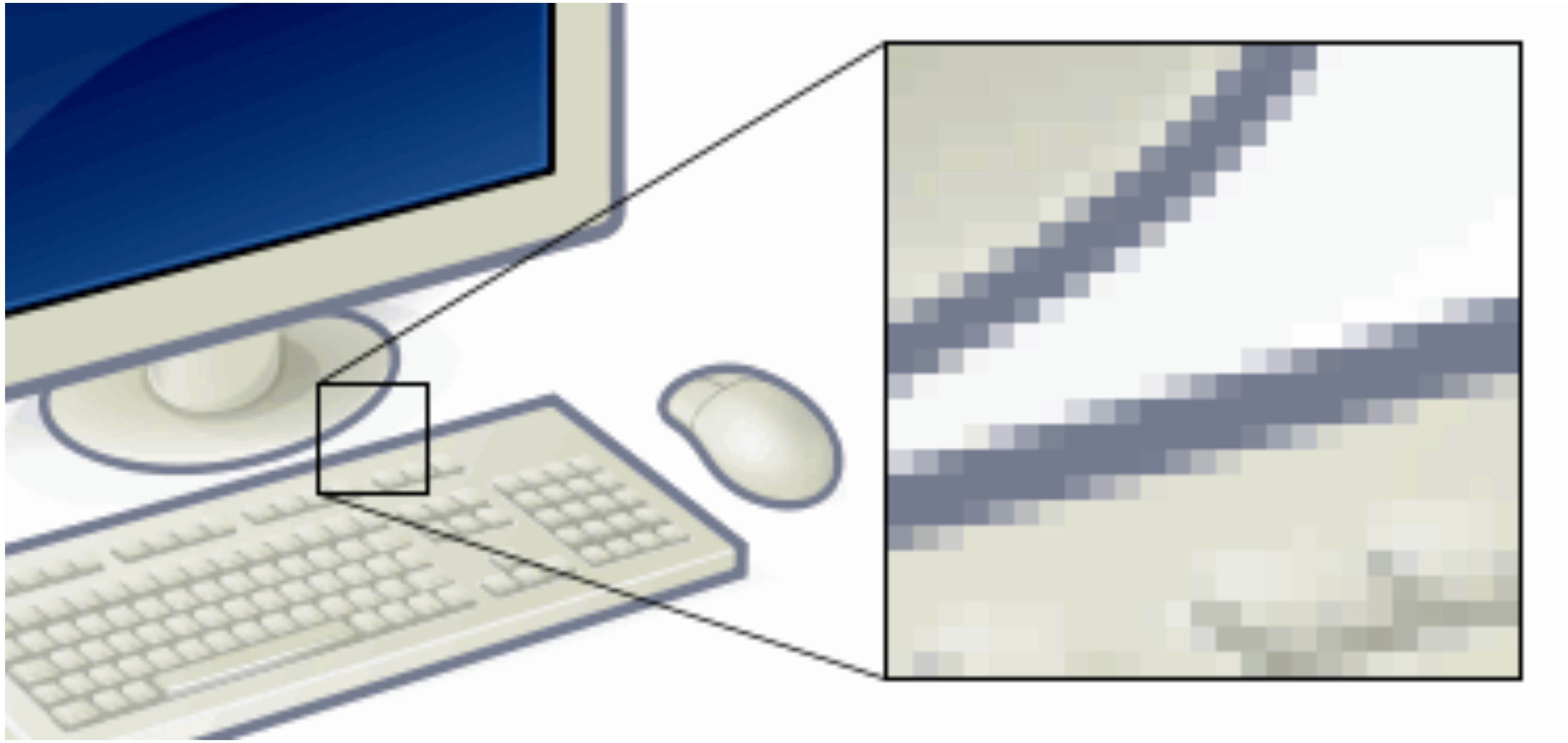
Part of SciPy, open source Python software for engineers

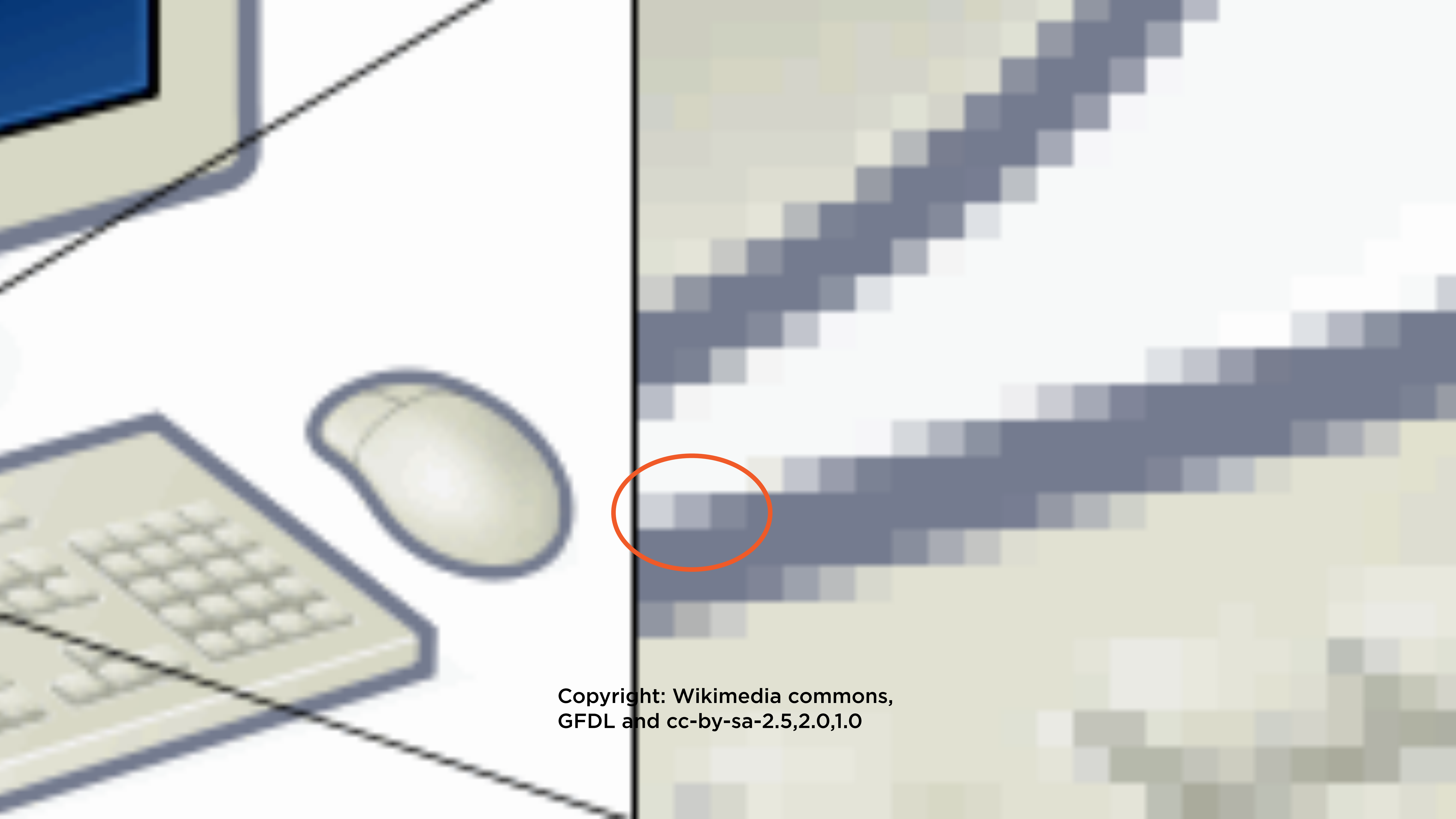
Image Pre-processing Methods

Uniform Aspect Ratio	Uniform Image Size	Mean and Perturbed Images
Normalized Image Inputs	Dimensionality Reduction	Data Augmentation

Common techniques to improve performance of machine learning models

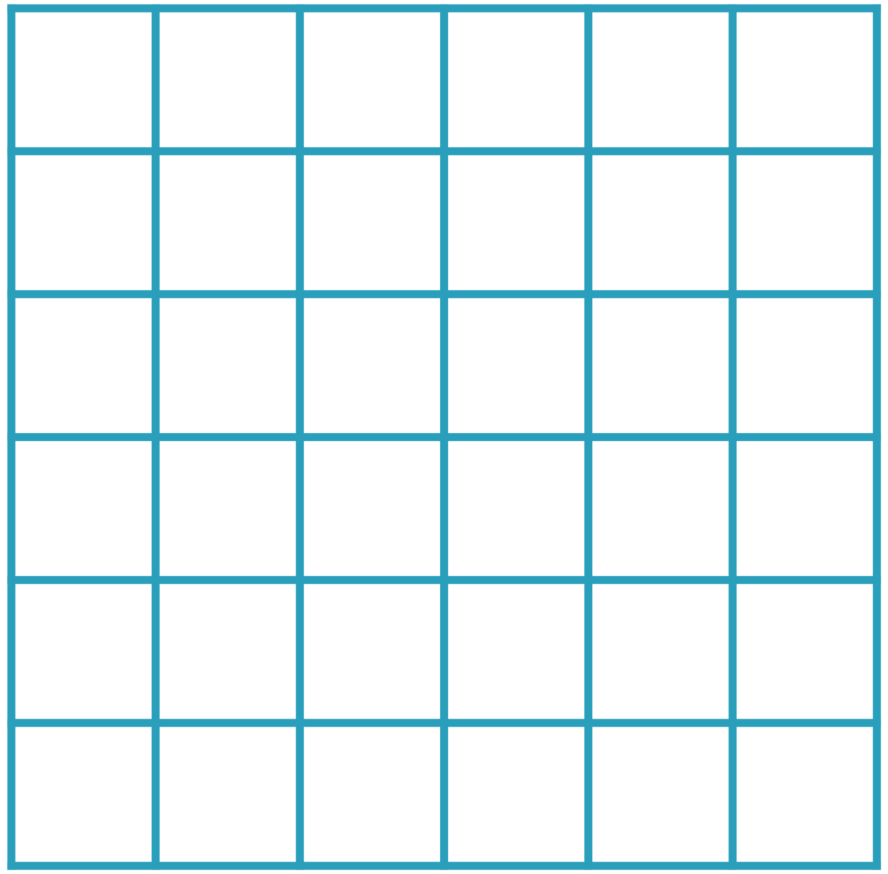
Pixels in Images





Images as Matrices

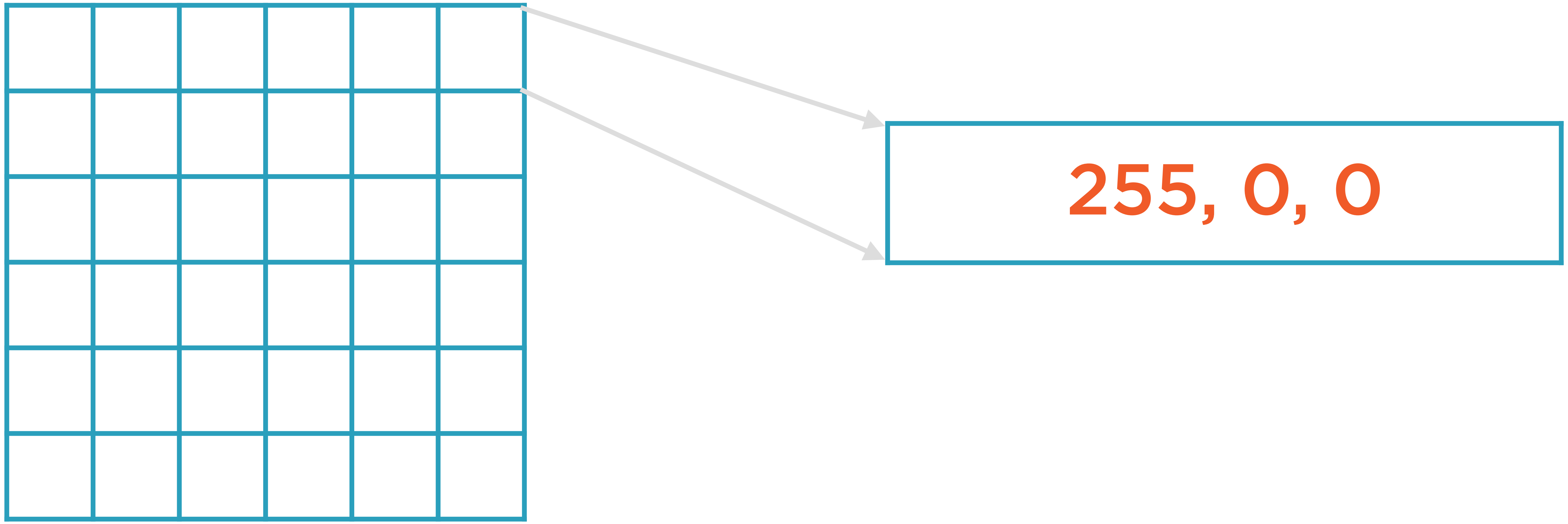
RGB Images



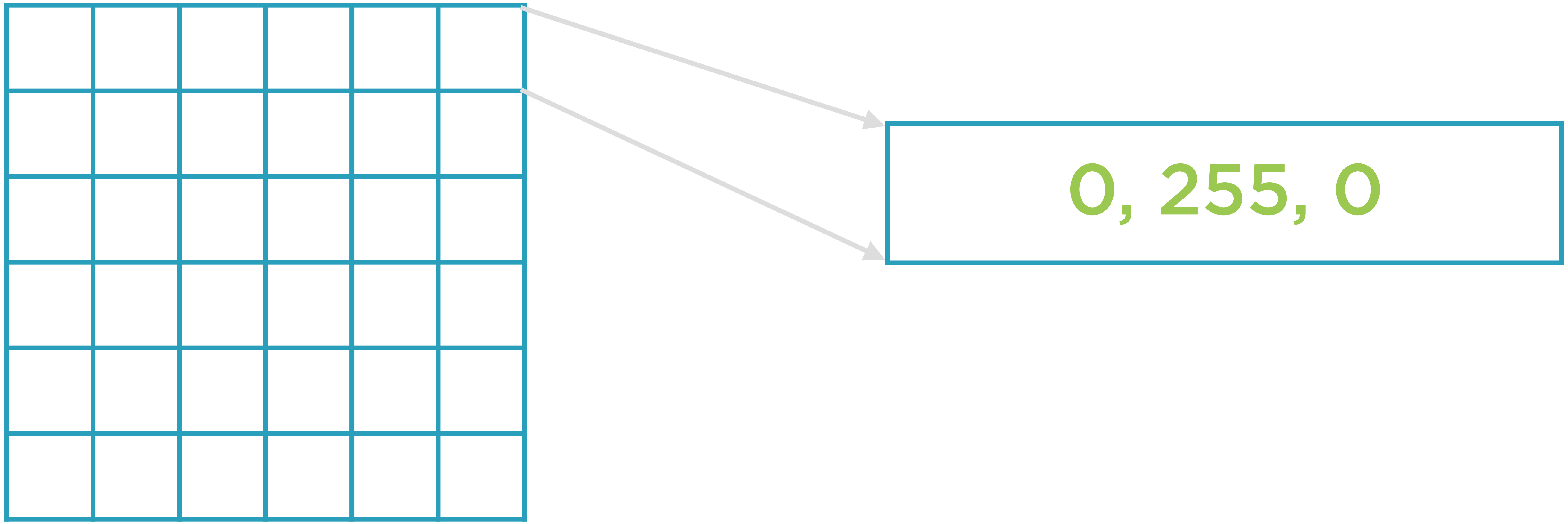
RGB values are for color images

R, G, B: 0-255

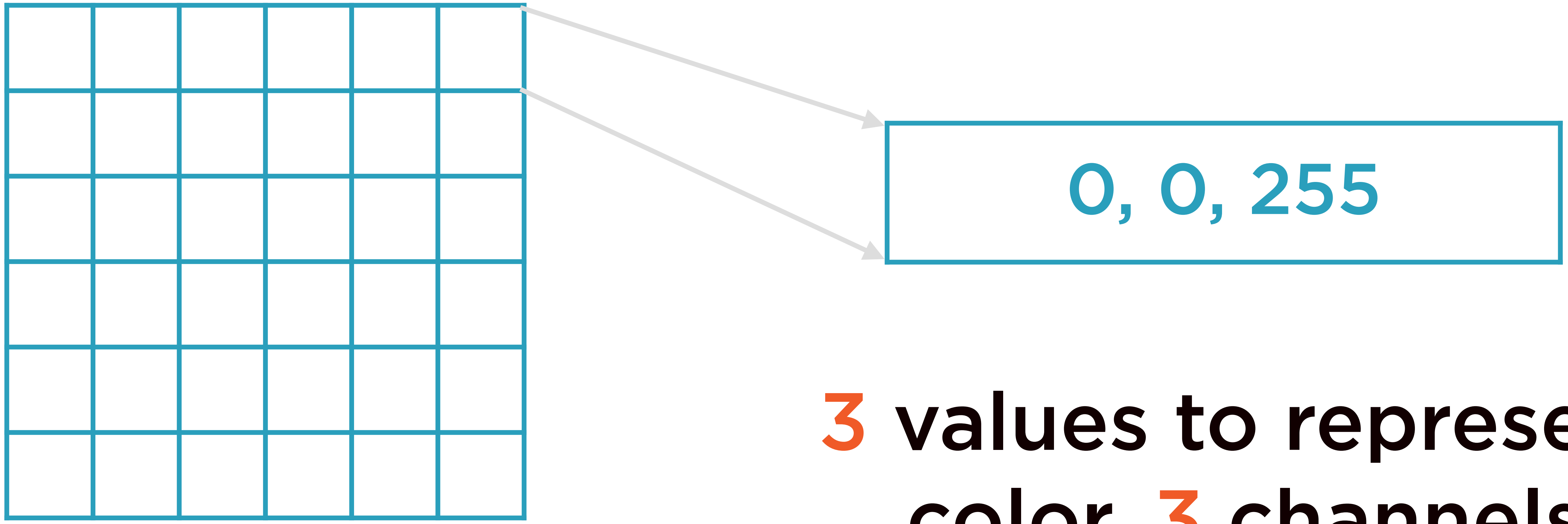
RGB Images



RGB Images

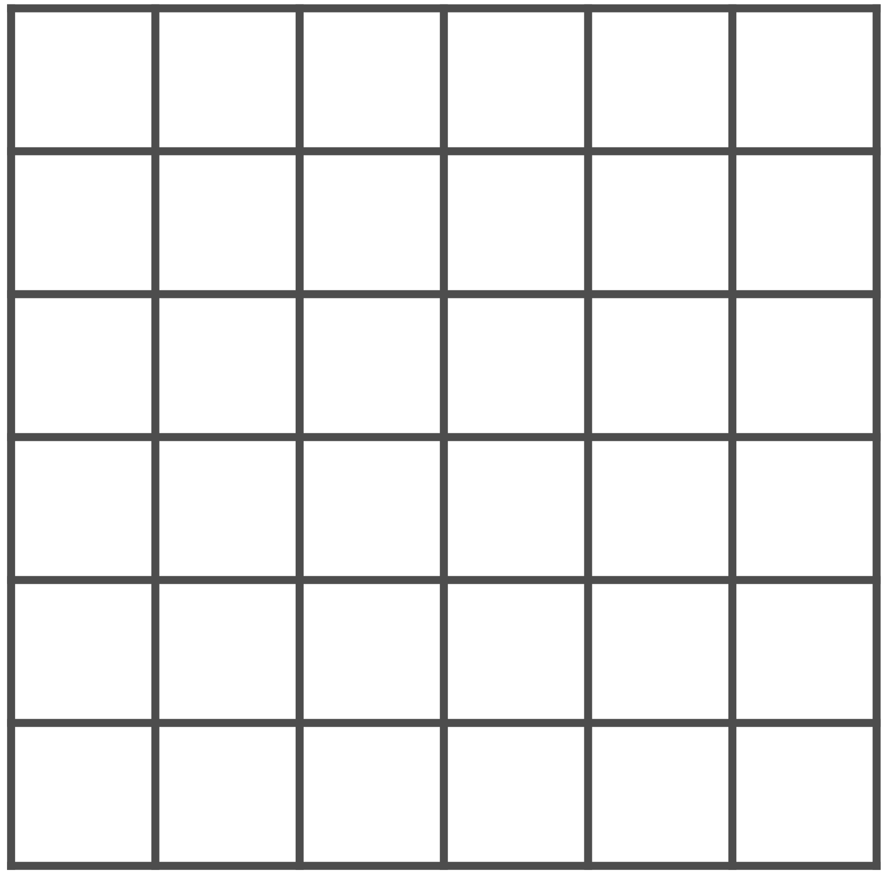


RGB Images

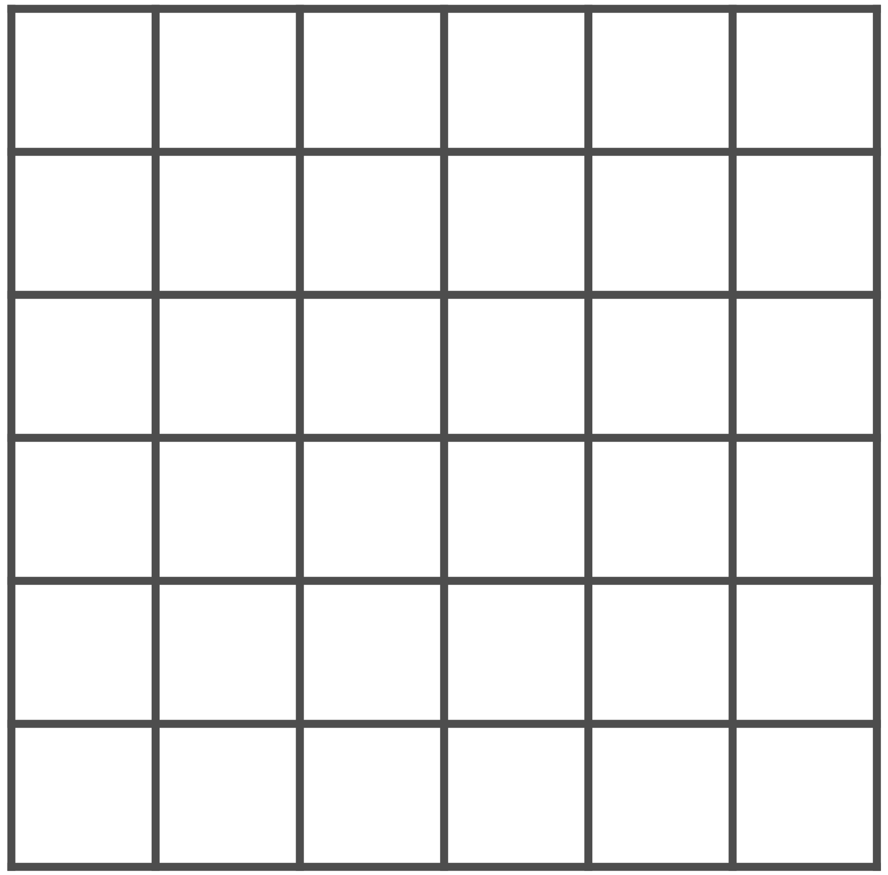


3 values to represent color, 3 channels

Grayscale Images



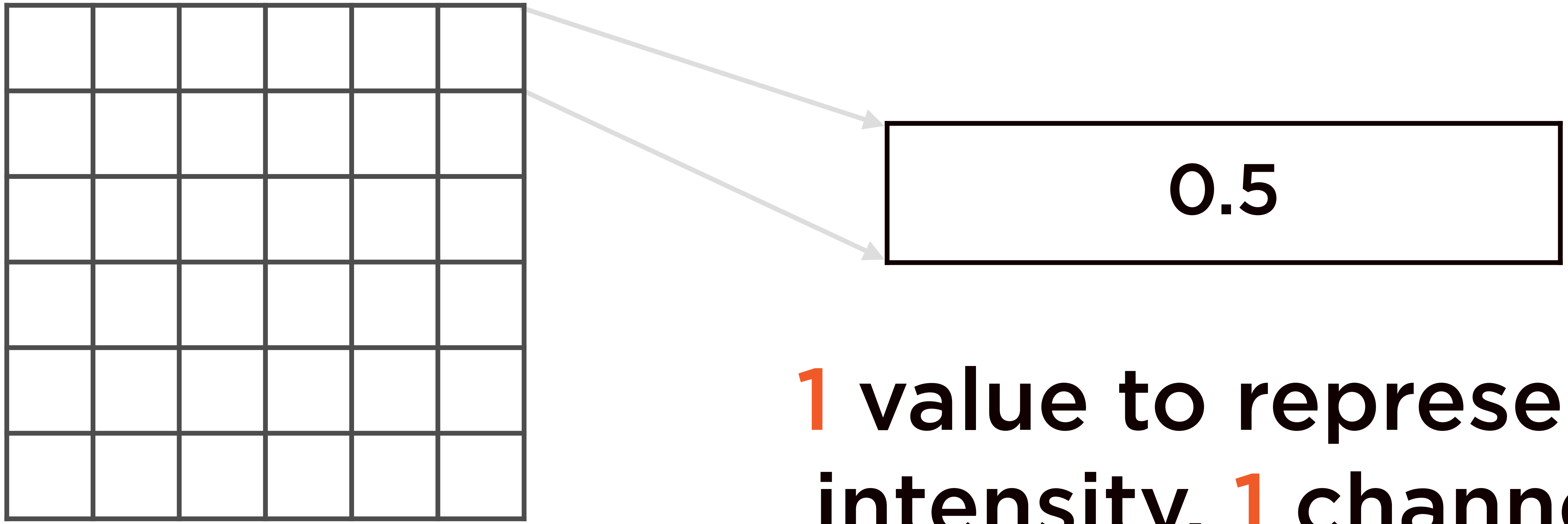
Grayscale Images



Each pixel represents only intensity information

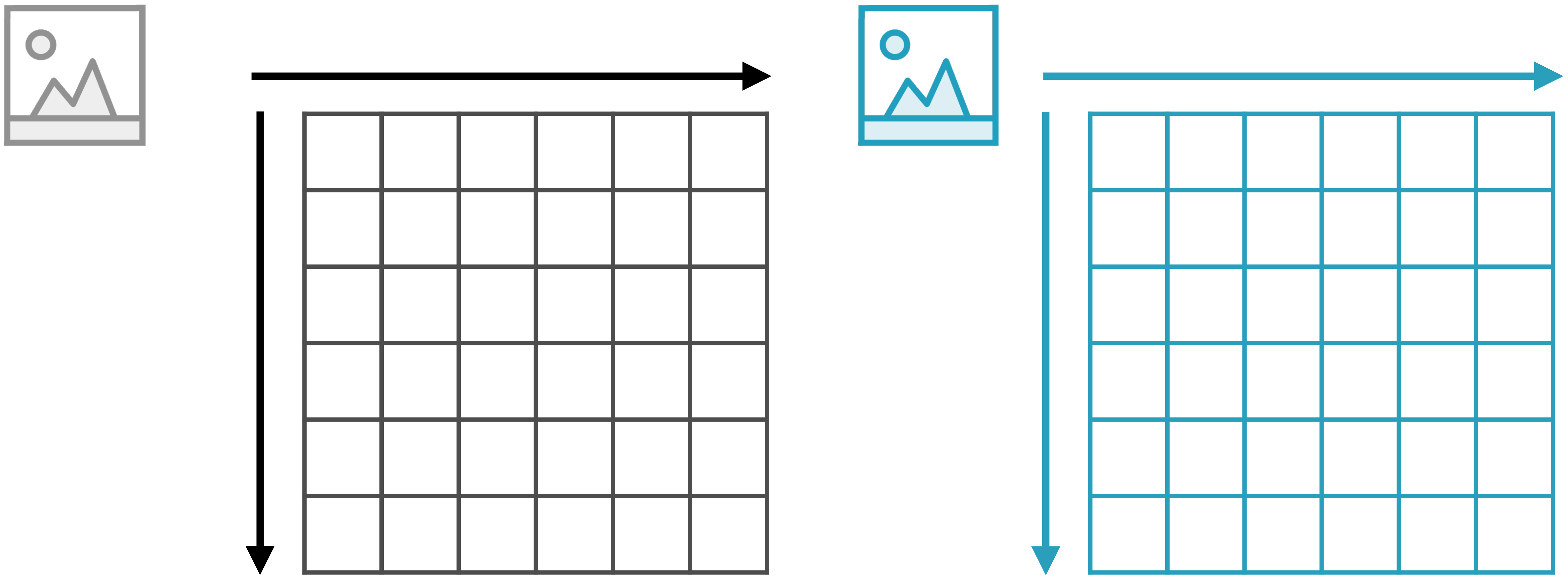
0.0 - 1.0

Grayscale Images



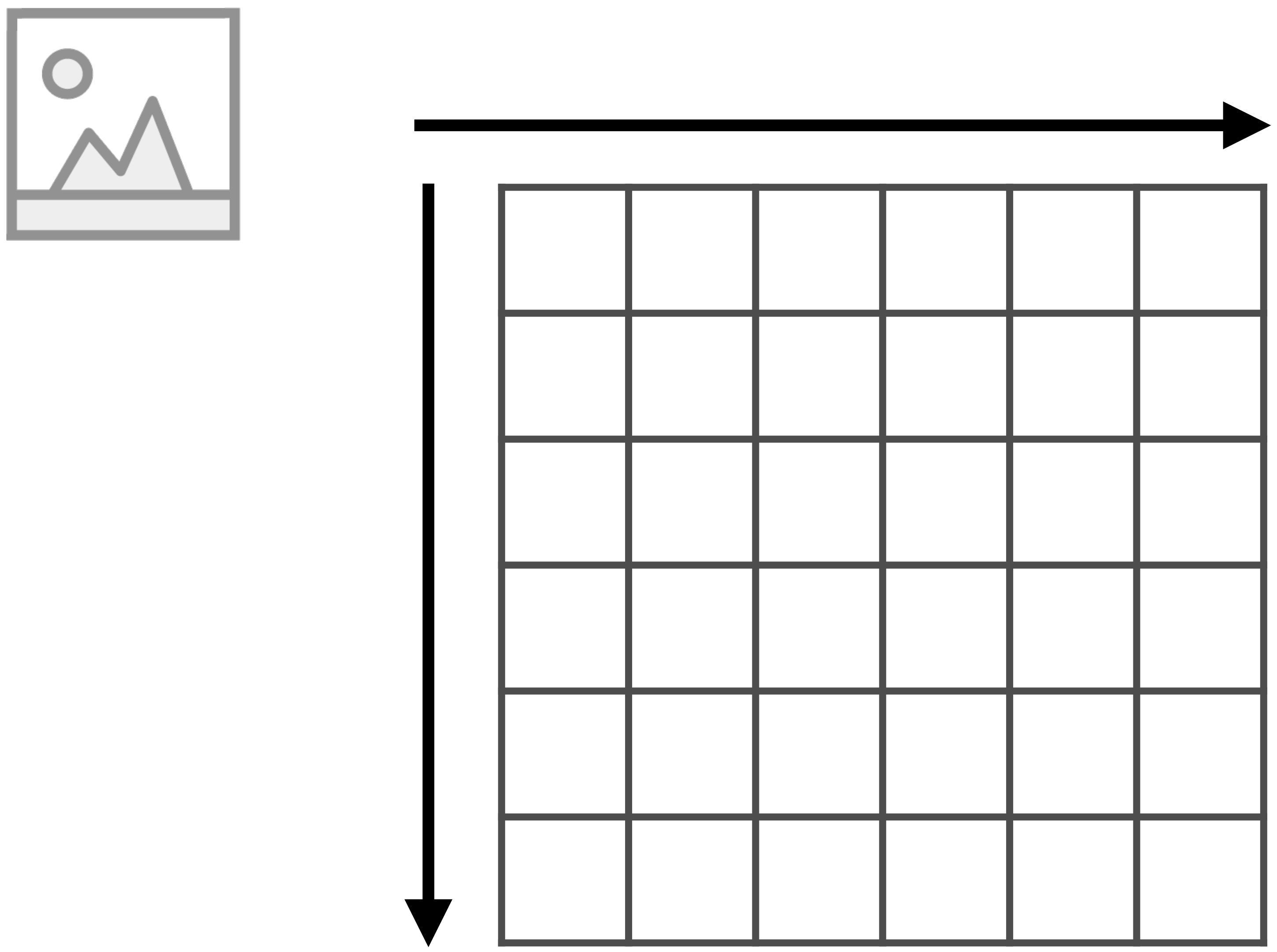
1 value to represent intensity, 1 channel

Images as Matrices

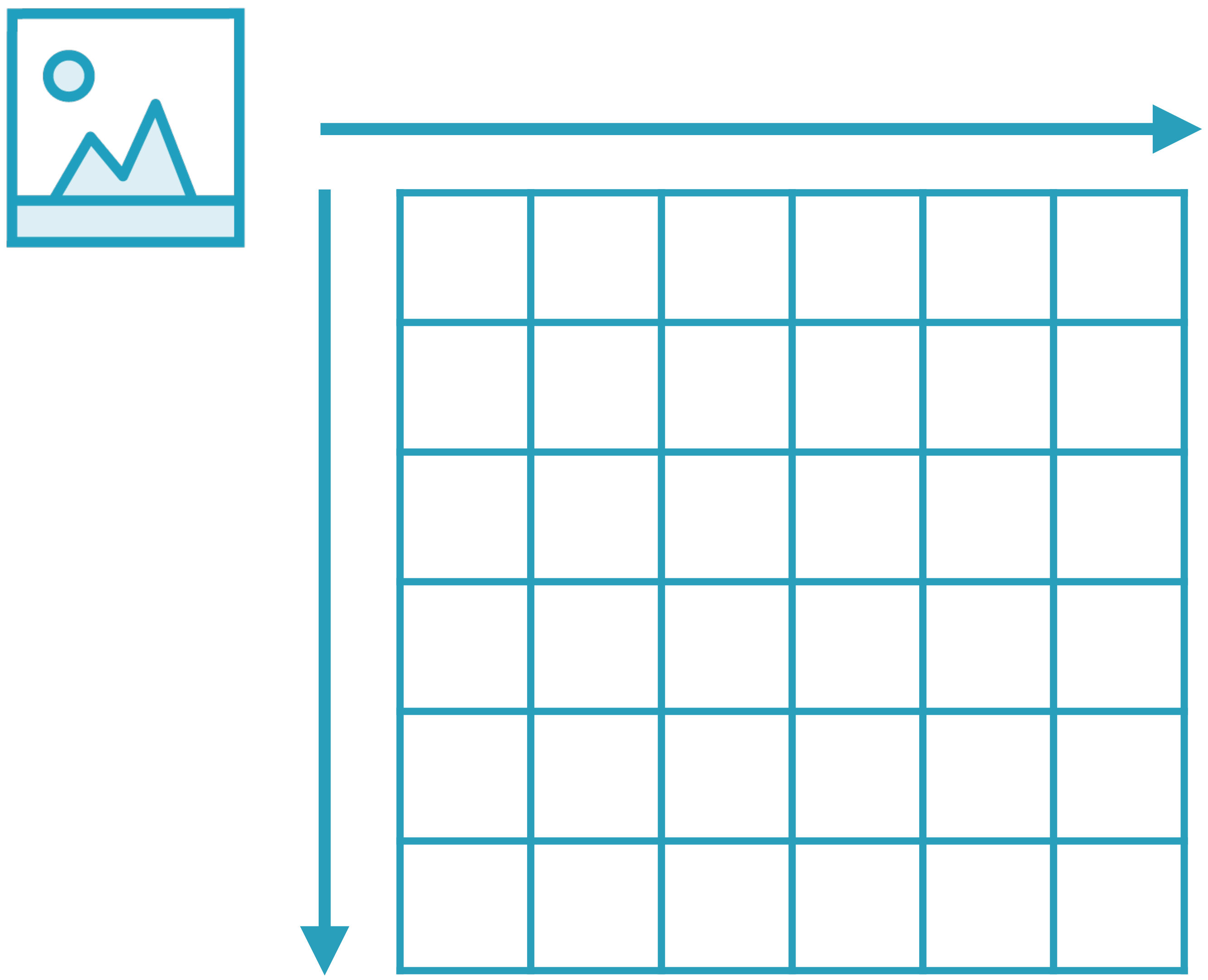


Images can be represented by a 3-D matrix

Images as Tensors



$(6, 6, \boxed{1})$



$(6, 6, \boxed{3})$

List of Images

ML frameworks (e.g. TensorFlow) usually deal with a list of images in one 4-D Tensor

List of Images

The images should all be the same size

List of Images

$(10, 6, 6, \boxed{3})$

The number of channels

List of Images

$(10, \boxed{6, 6}, 3)$

The height and width of each image in the list

List of Images

$(\boxed{10,}\ 6, 6, 3)$

The number of images

Demo

Working with images using NumPy

BlockView and Pooling

Block Views

Extract non-overlapping image patches in order to perform local operations on these patches

Pooling

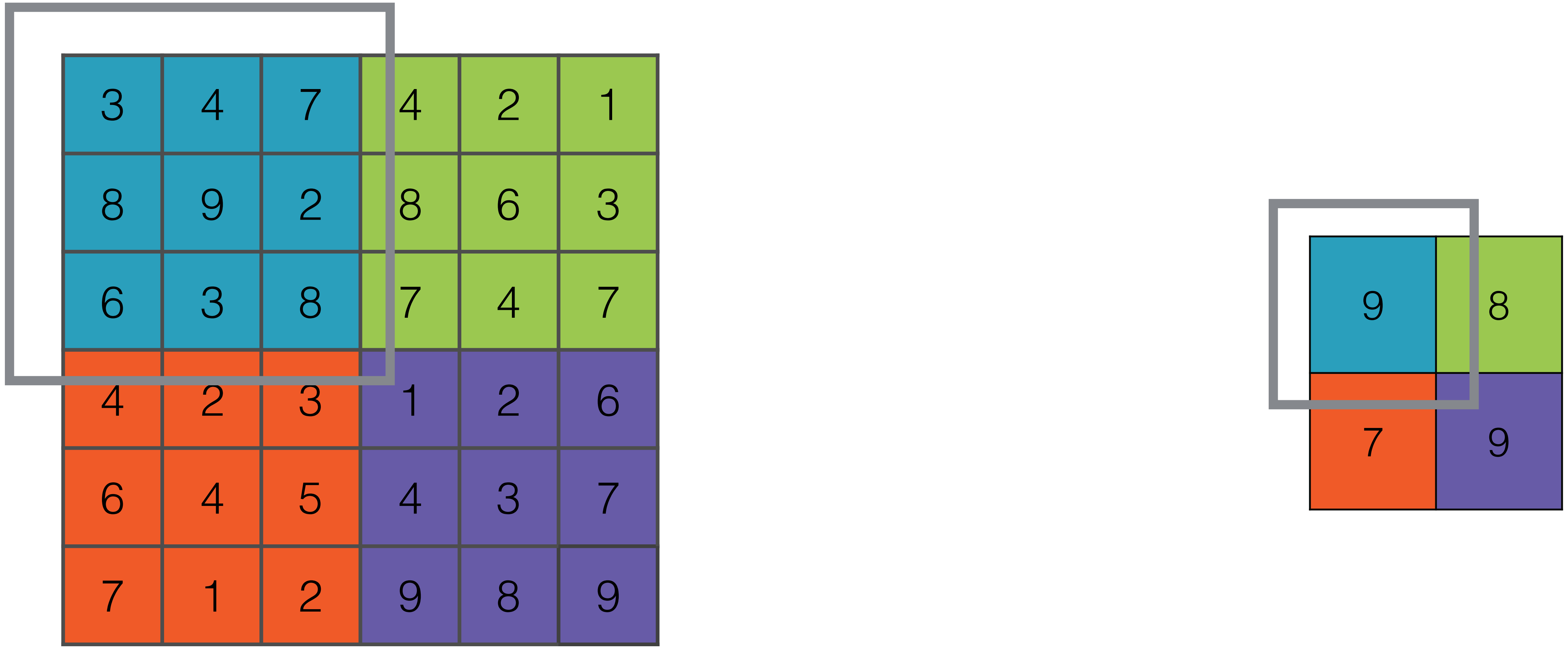
A window function which performs an aggregation on the input to extract significant features

Max Pooling

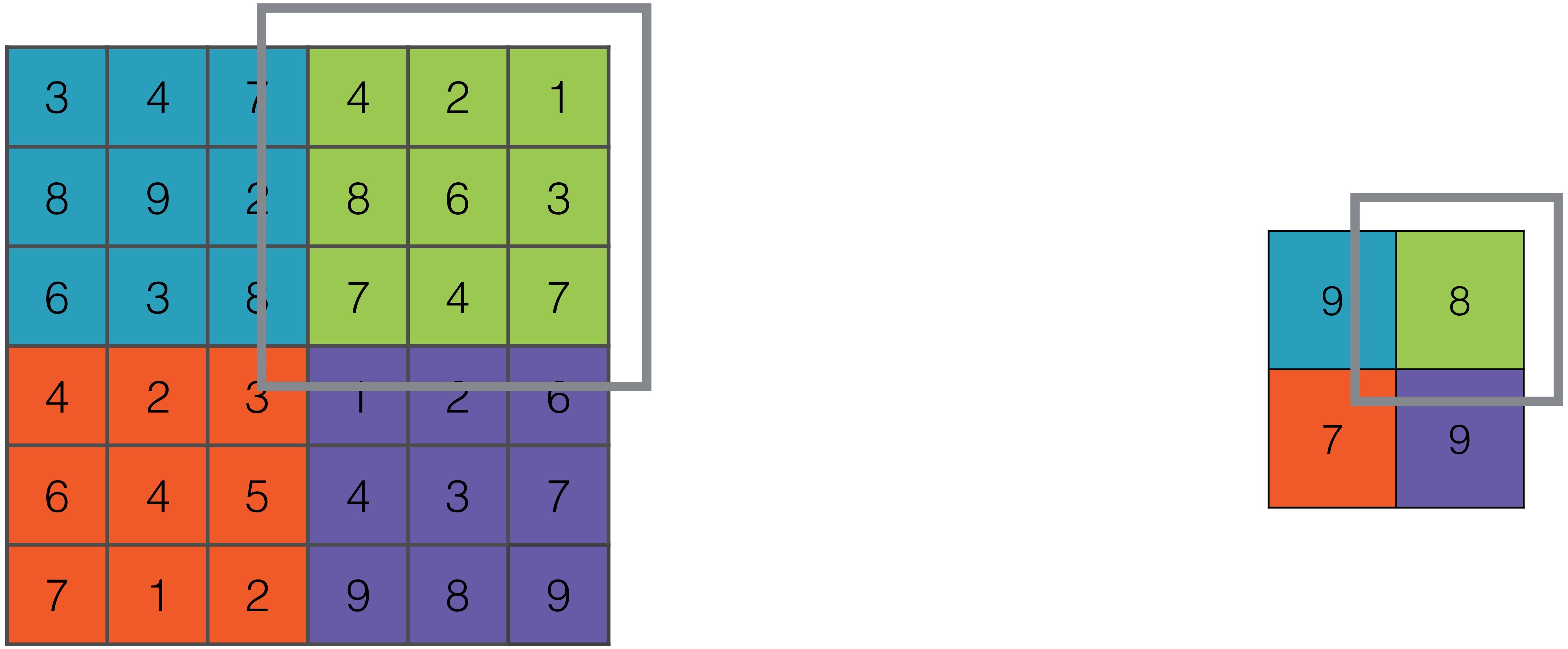
3	4	7	4	2	1
8	9	2	8	6	3
6	3	8	7	4	7
4	2	3	1	2	6
6	4	5	4	3	7
7	1	2	9	8	9

9	8
7	9

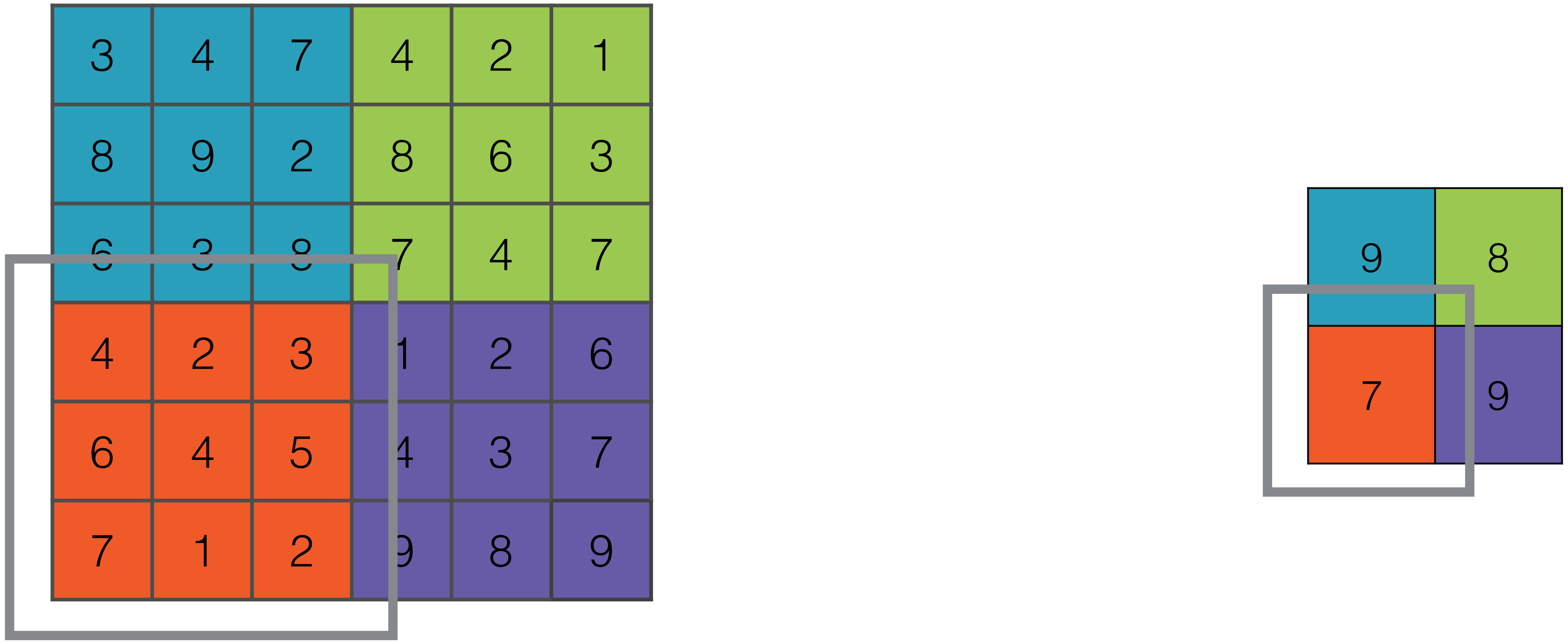
Max Pooling



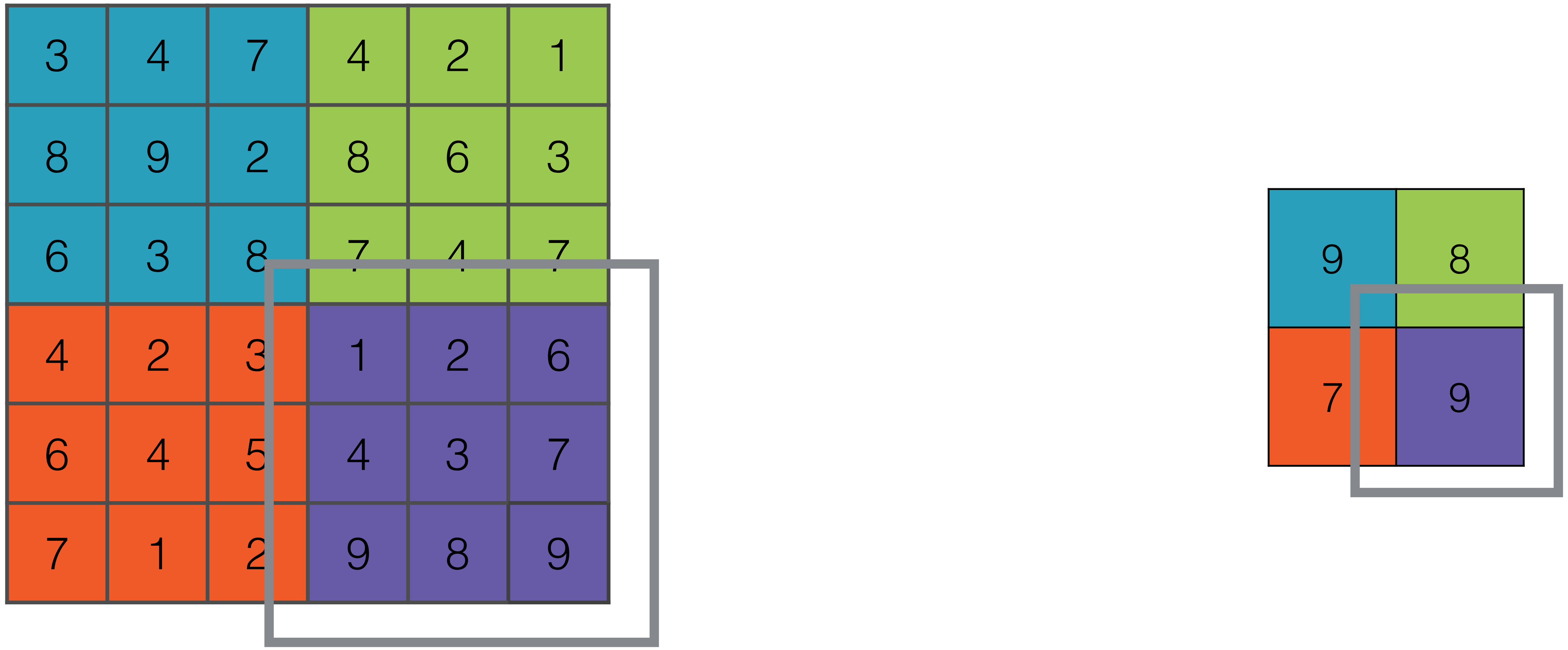
Max Pooling



Max Pooling



Max Pooling



Max Pooling

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

9	8
7	9

Min Pooling

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

2	1
1	1

Min Pooling

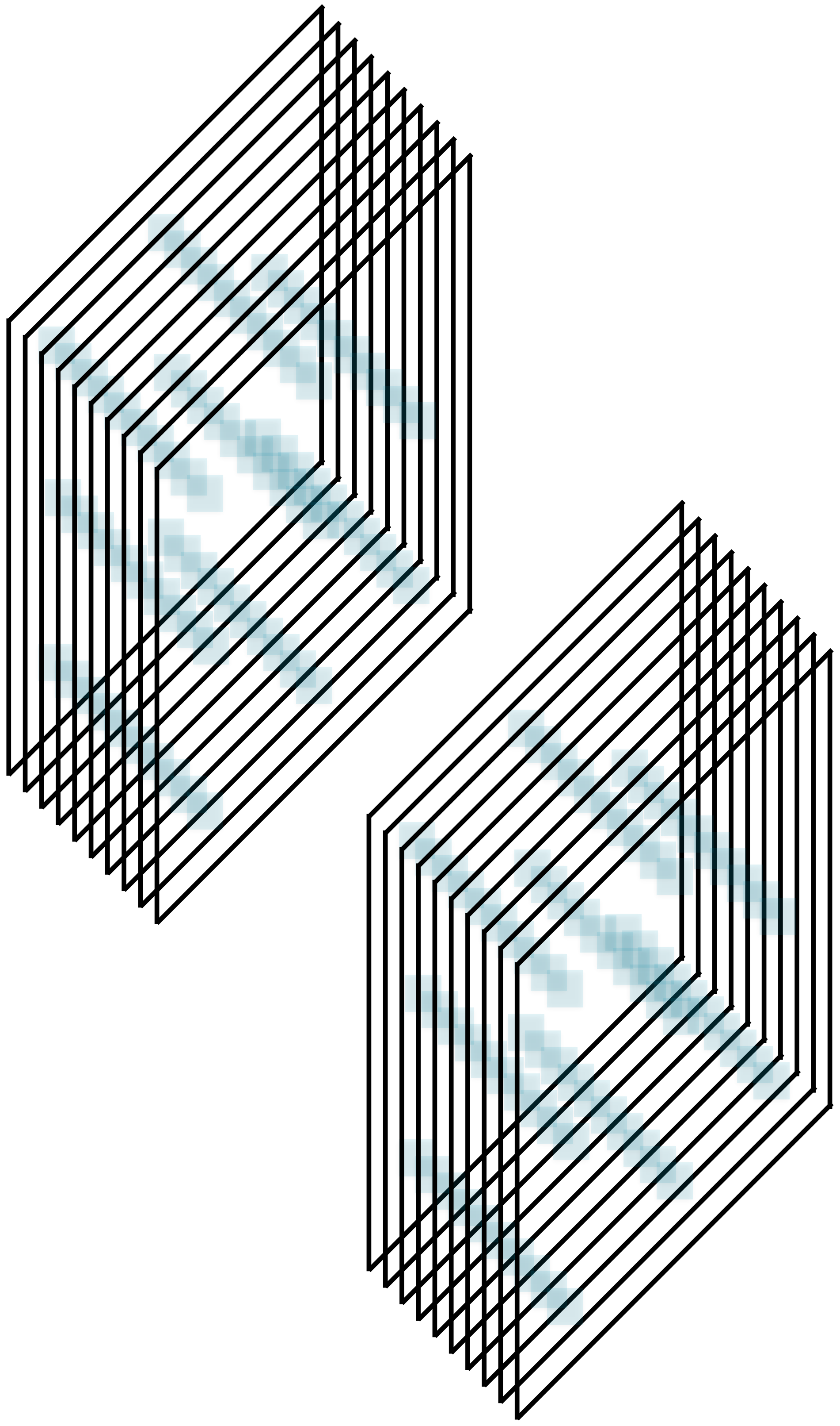
3	4	7	4	2	1
8	9	2	8	6	3
6	3	8	7	4	7
4	2	3	1	2	6
6	4	5	4	3	7
7	1	2	9	8	9

2	1
1	1

Average Pooling

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

5.5	5.66
5.5	5.4



Pooling Layers

Pooling is widely used in Convolutional Neural Networks (CNNs)

Allow extraction of features in a location invariant manner

Demo

Block views on image arrays

Demo

Contour detection with the marching squares algorithm

Contour Line

A curve along connecting pixels with the same color; more generally a curve along which a function has the same value

Contours

Alternative names for contour lines

- equipotential curves
- indifference curves
- isopleths, isoclines

Common Applications

Cartography (mapping)

- connect points at equal elevation above sea level

Economics

- connect points with equal utility (indifference curves)

Two Types of Contours

Isolines: Lines exactly follow single data value

- the common data value is called the isovalue

Isobands: Areas between isolines are filled in

The Marching Squares algorithm

An easy-to-implement, embarrassingly parallel algorithm that generates contours for a two-dimensional (rectangular) array

Embarrassingly Parallel Algorithm

Where little or no effort is needed to separate the problem into a number of parallel tasks, usually because there is little or no dependency or need for communication between those parallel tasks

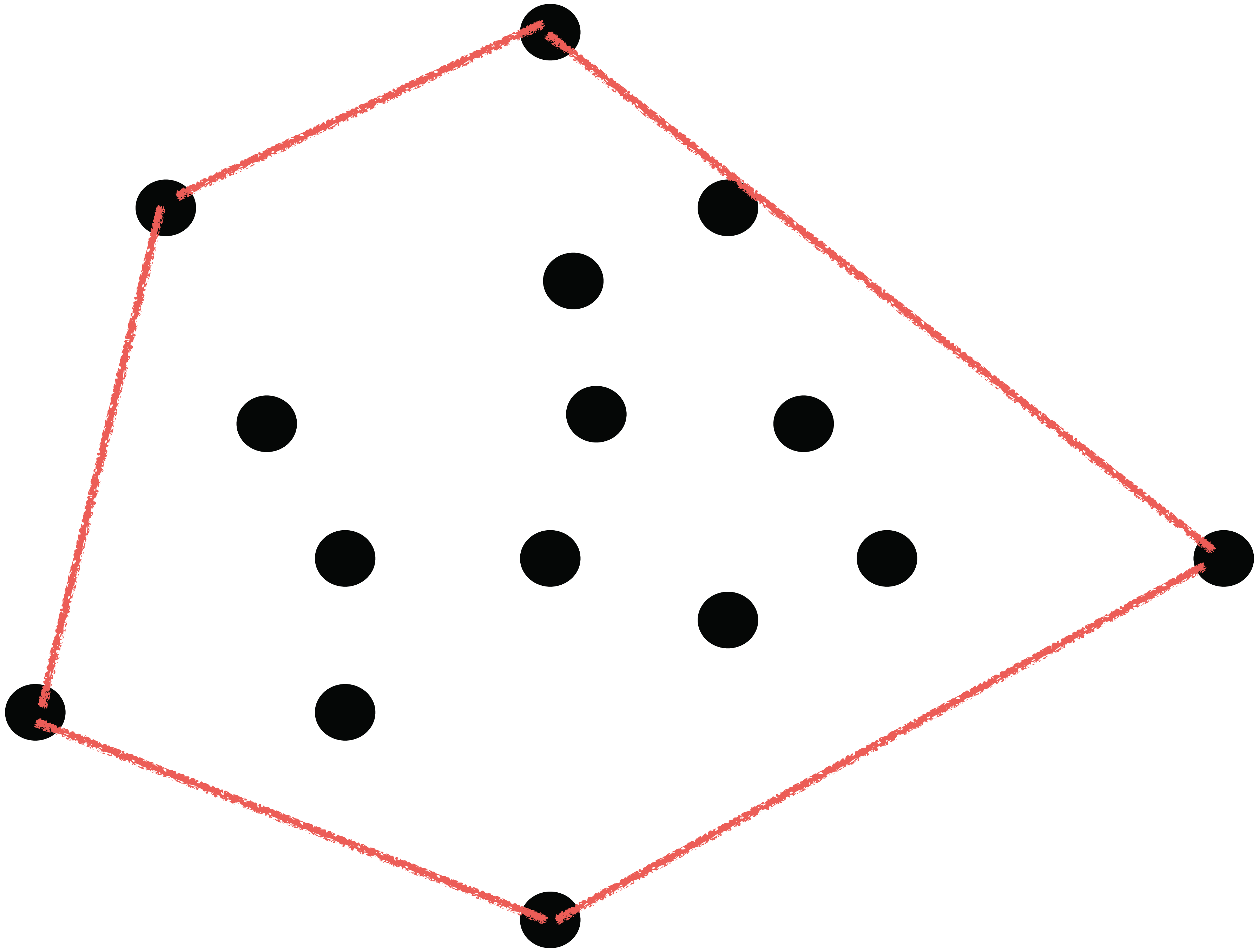
Demo

Finding the convex hull of an image

Convex Hull

Given a X, a set of points in a plane, the convex hull of X is the smallest convex polygon that contains all points in X

Convex Hull



Applications

Widely used in image processing

Pattern recognition

Statistics

Many other disciplines

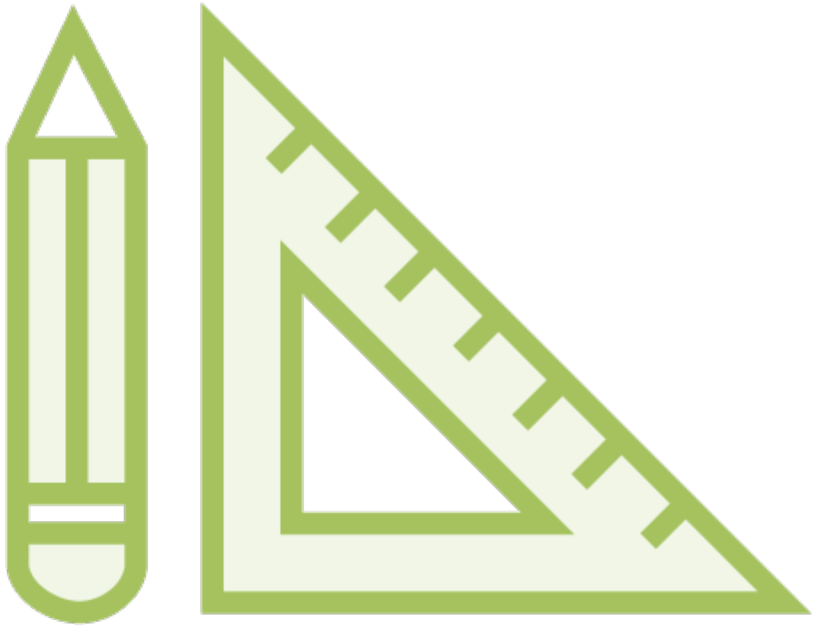
Edge Detection

Edge

Curve connecting points where intensity (brightness) of pixels changes abruptly

Edge Detection

Techniques that detect edges in images

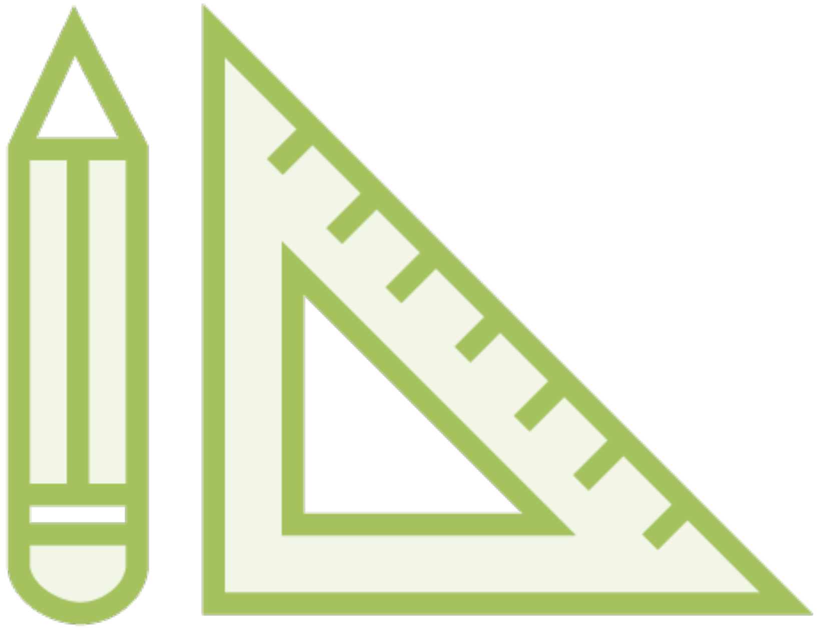


Applications

Feature detection

Feature extraction

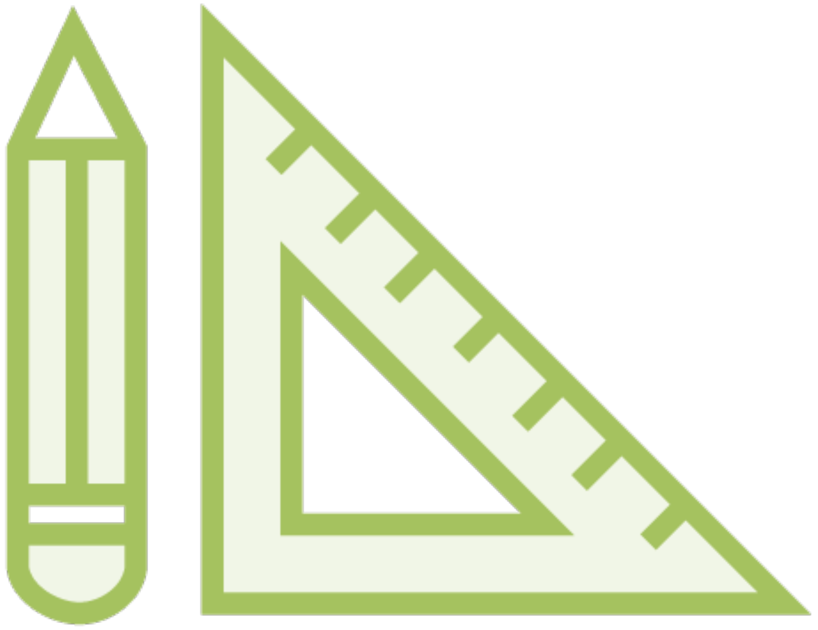
Widely used in computer vision



Types of Edge Detection

Search-based

Zero-crossing

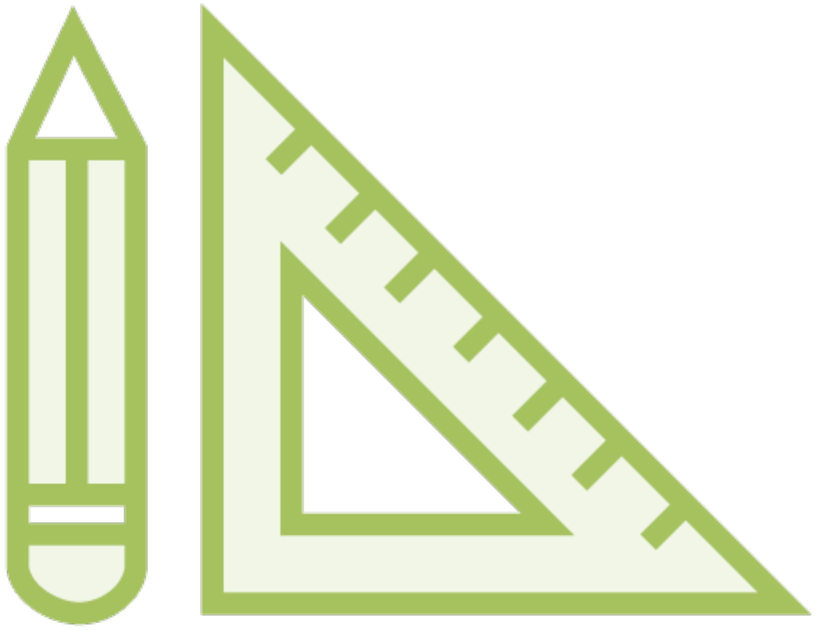


Zero-crossing Edge Detection

Zero-crossing: point along curve where value flips sign

- Positive to negative
- Negative to positive

In continuous curve, this occurs at value of zero



Search-based Edge Detection

Find some measure of "edge strength"

First-order derivative suffices

Search for direction of maxima

- Single maxima: edge
- Multiple maxima: corner

Intuition behind Search-based Edge Detection

Compute derivatives in x and y direction

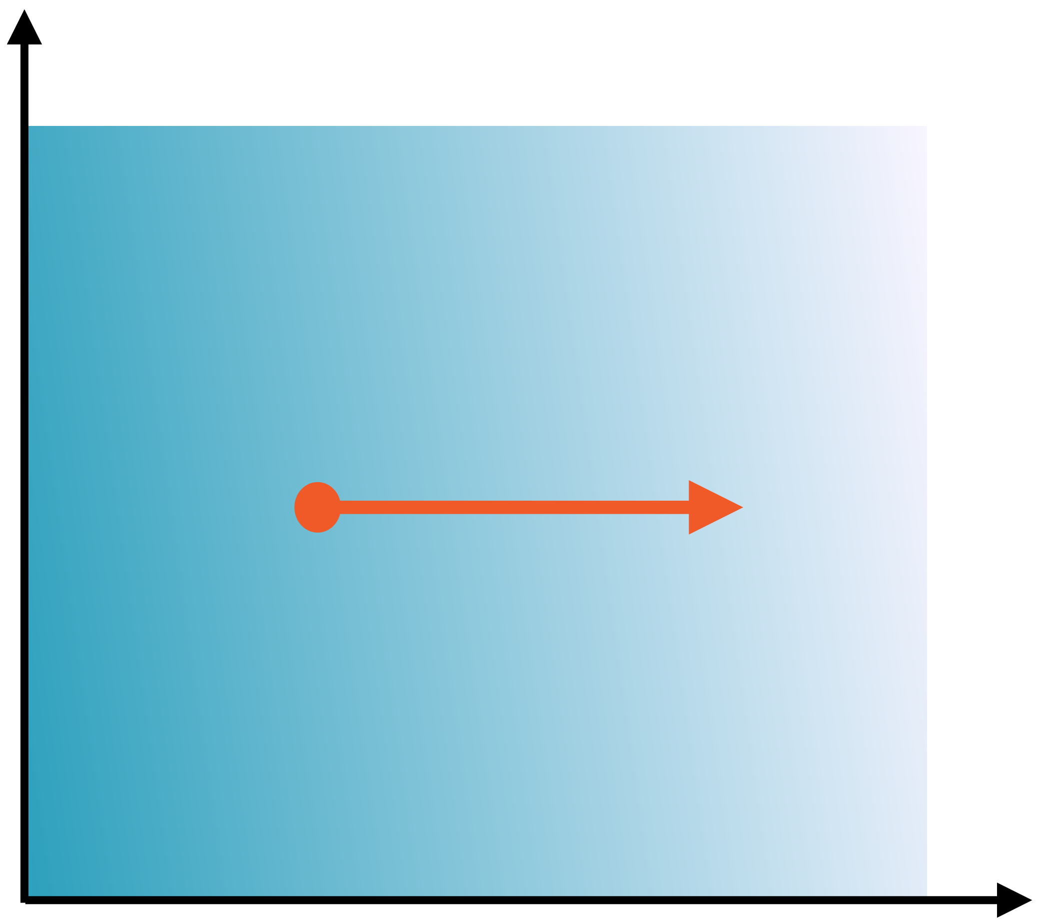
Find gradient magnitude

Apply threshold to gradient magnitude

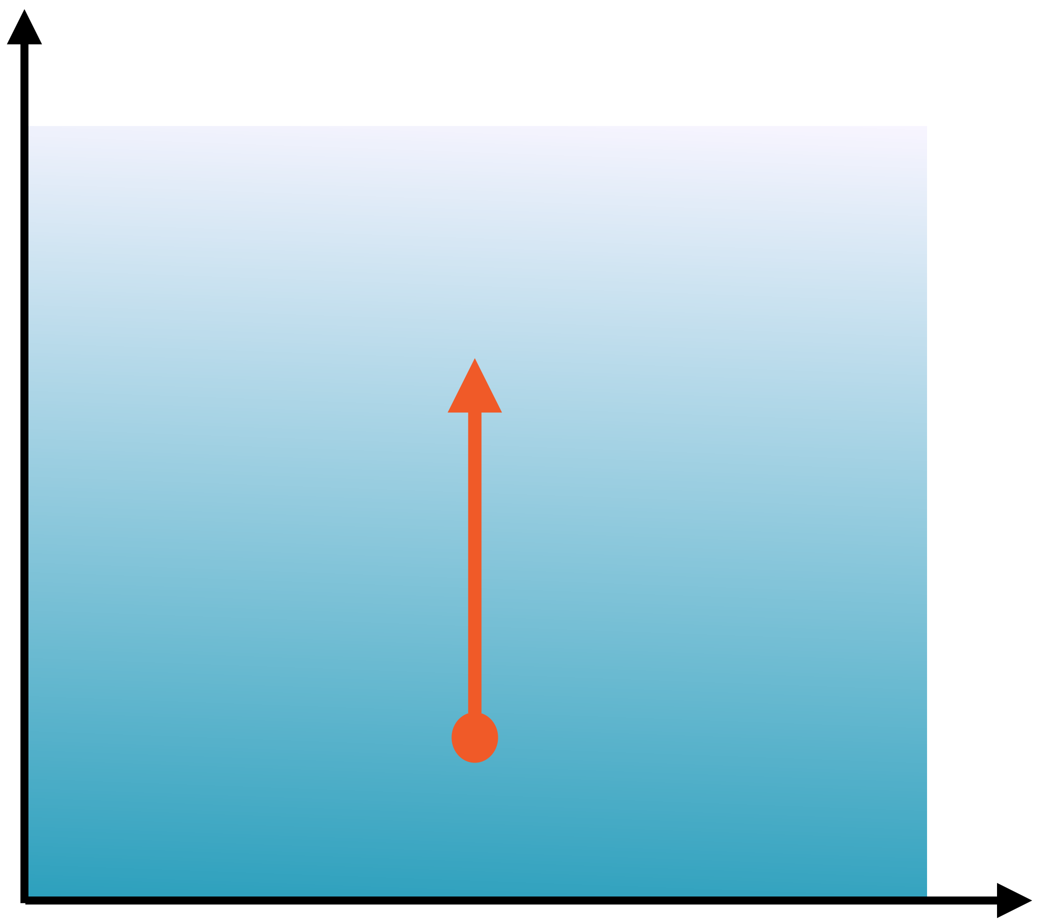
Intuition behind Search-based Edge Detection

Compute derivatives in x and y direction	Find gradient magnitude	Apply threshold to gradient magnitude

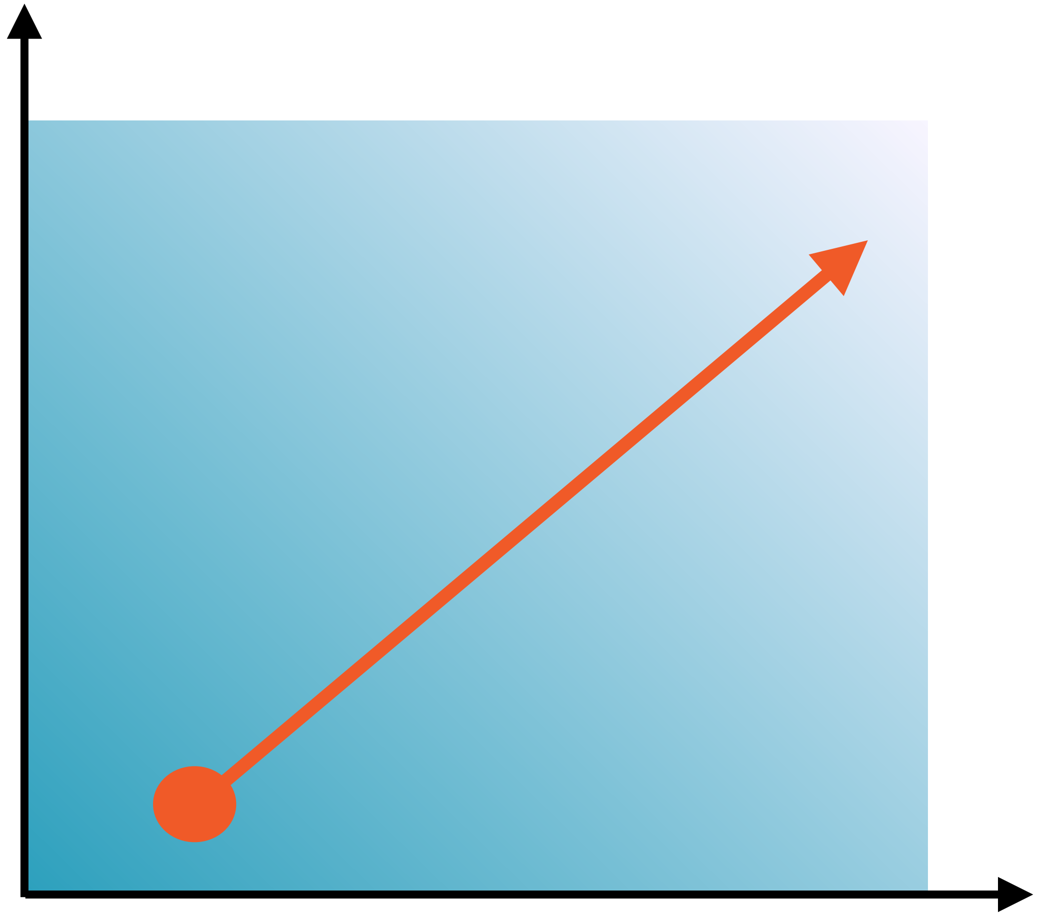
Intuition behind Search-based Edge Detection



Intuition behind Search-based Edge Detection



Intuition behind Search-based Edge Detection



Intuition behind Search-based Edge Detection

Compute derivatives in x and y direction

Find gradient magnitude

Apply threshold to gradient magnitude

Intuition behind Search-based Edge Detection

Compute derivatives in x and y direction

Find gradient magnitude

Apply threshold to gradient magnitude

Apply threshold to gradient magnitude

Pair of 2x2 convolution kernels

Designed to respond maximally to edges running at 45° to the pixel grid

Simple and quick operation

Convolution

In this context, a sliding window function applied to a matrix

Representing Images

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Matrix

1	0	1
0	1	0
1	0	1

Kernel

Representing Images

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

1	0	1
0	1	0
1	0	1

The kernel has weights which are tuned to extract edges

Representing Images

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

1	0	1
0	1	0
1	0	1

The weights for the Roberts and Sobel edge detection are different

Convolution

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

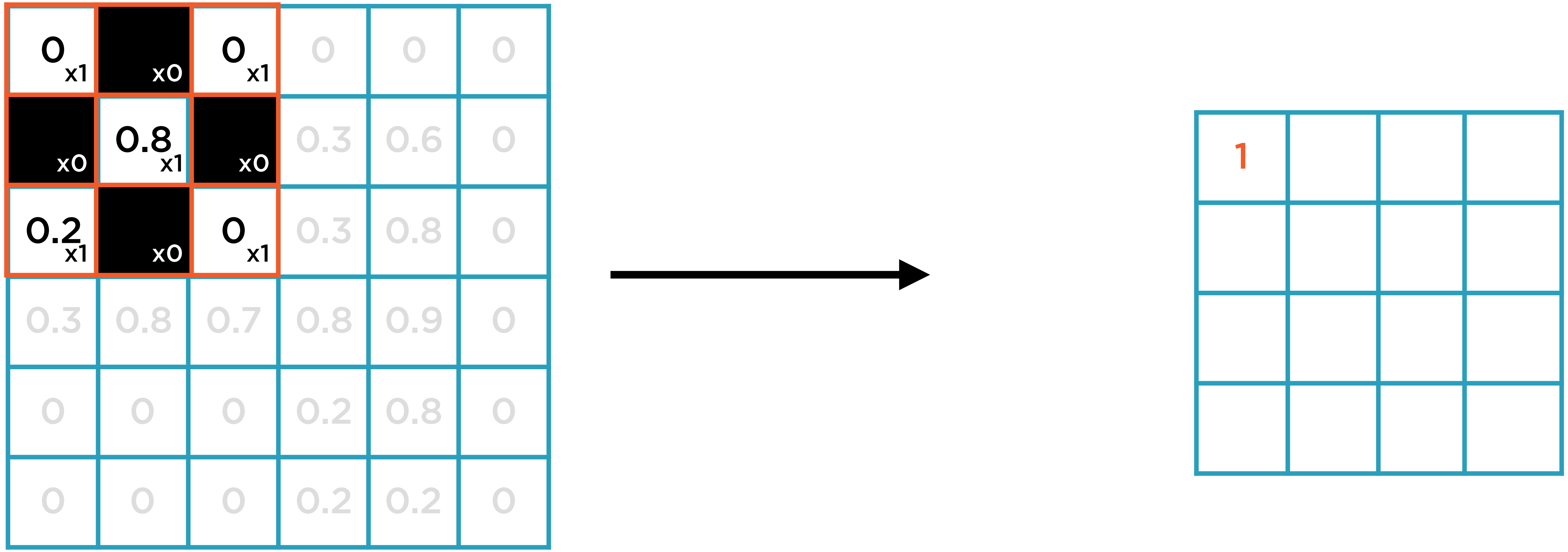
Matrix

x1	x0	x1
x0	x1	x0
x1	x0	x1

1	1.2	1.1	0.9
1.9	2.7	2.5	1.9
1.0	2.1	2.4	1.4
1.0	1.8	2.0	1.8

Convolution Result

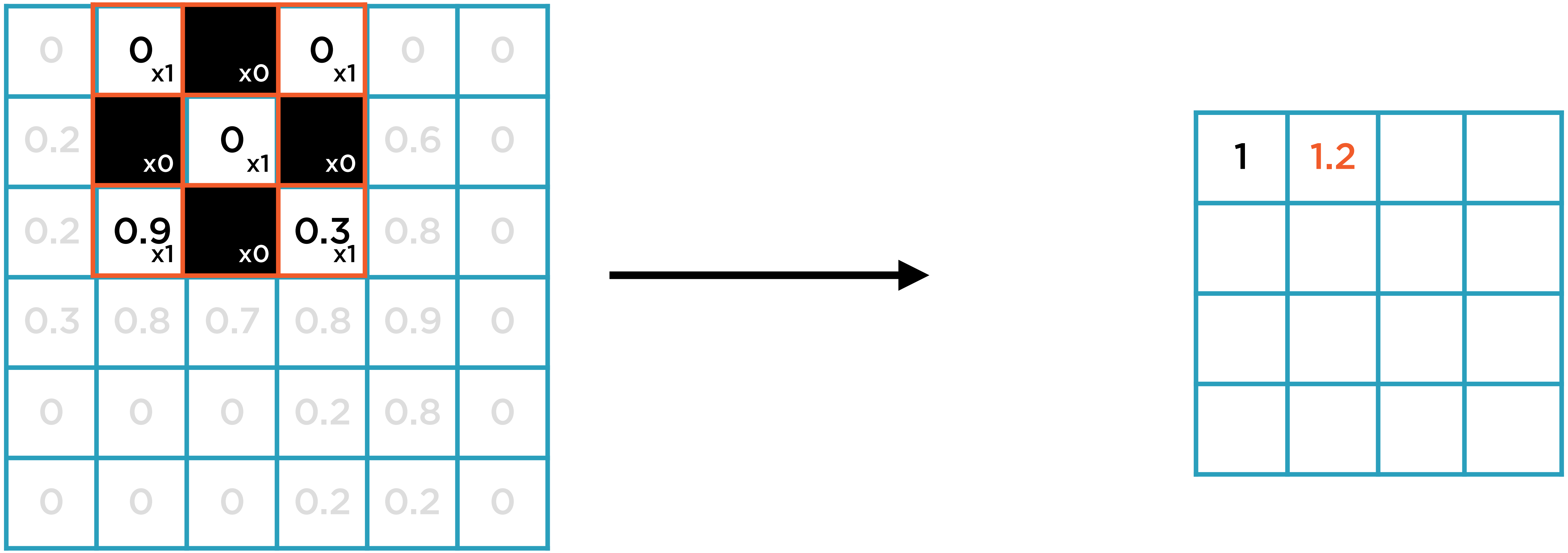
Convolution



Matrix

Convolution Result

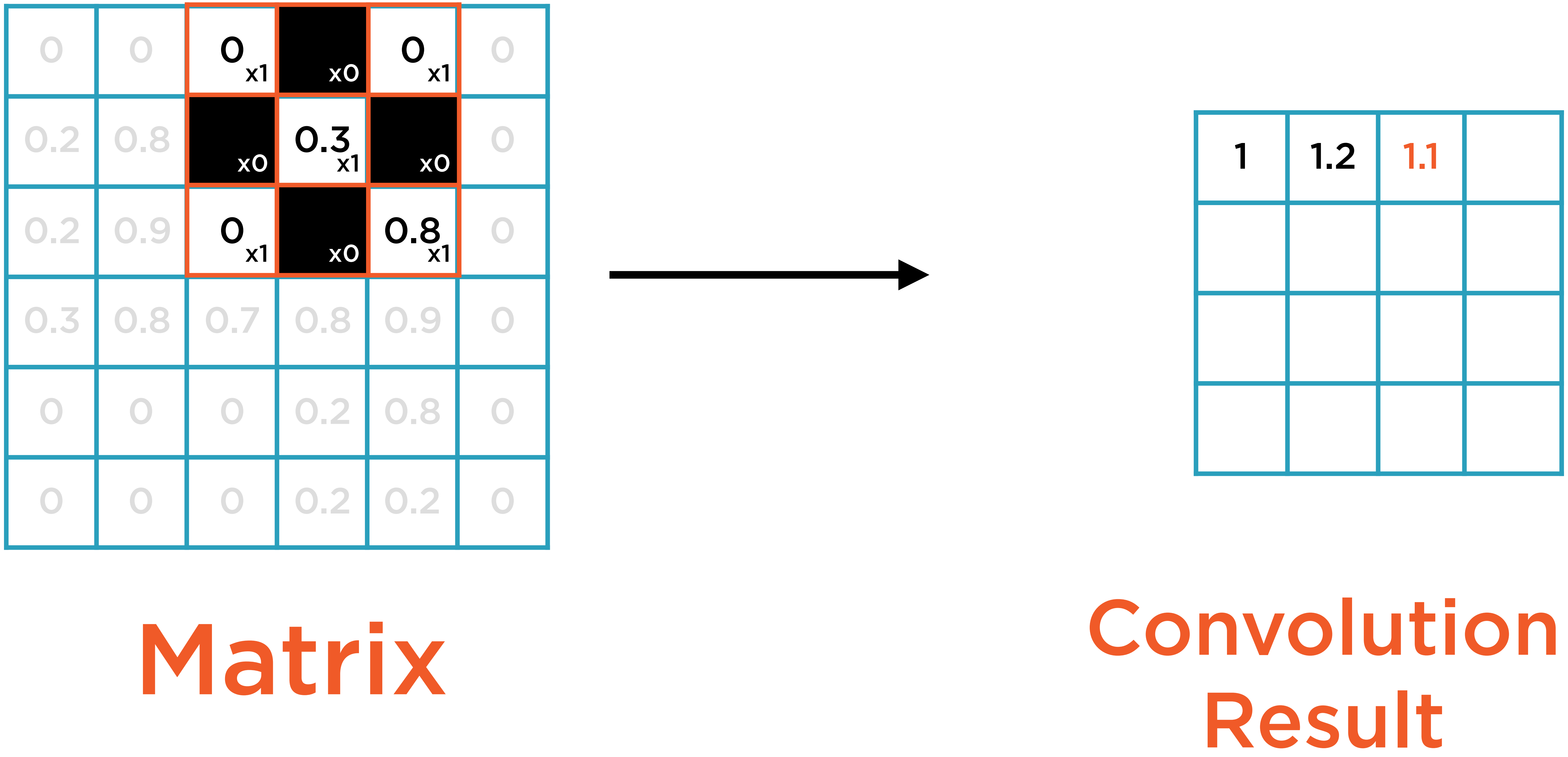
Convolution



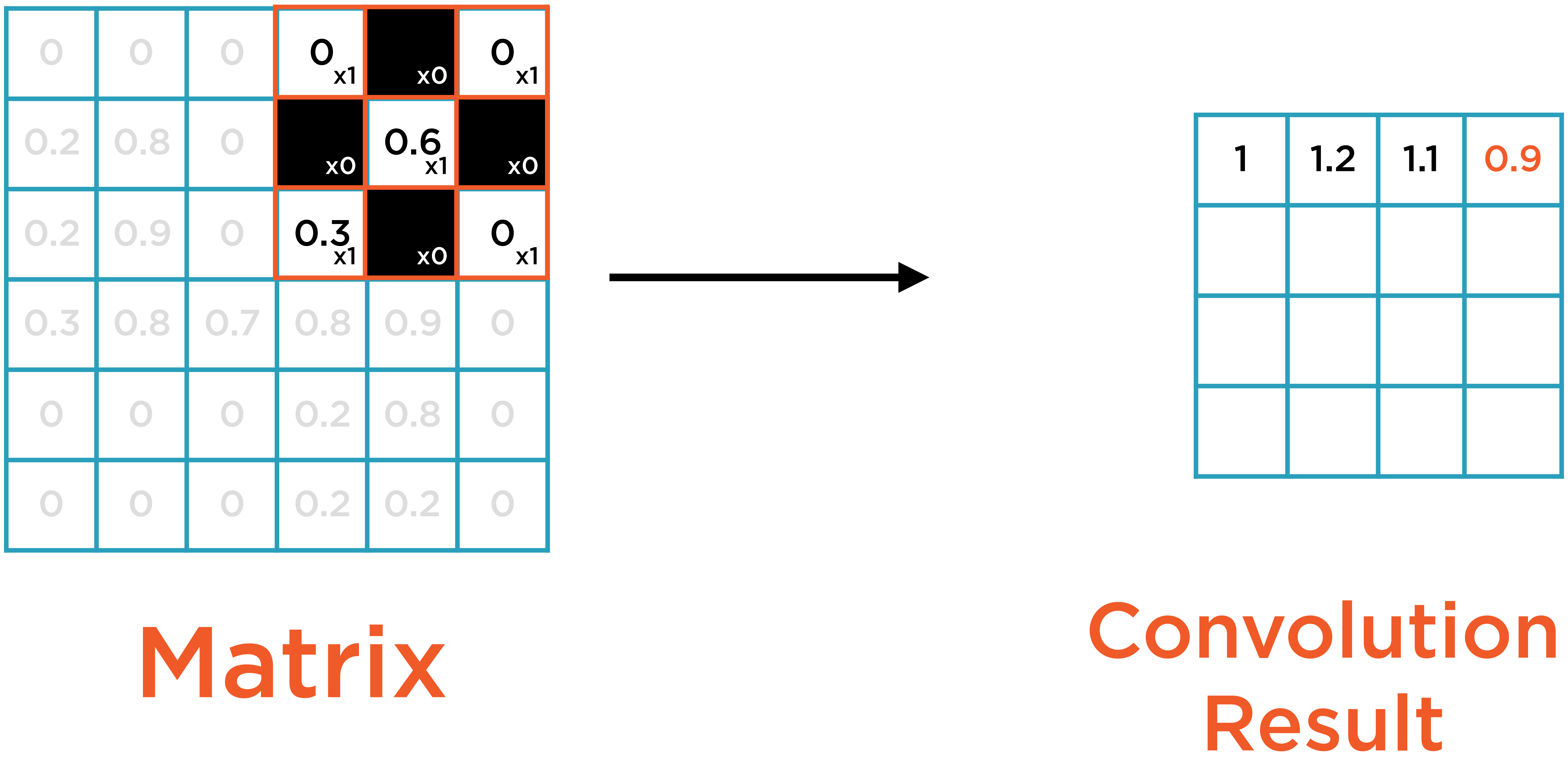
Matrix

Convolution Result

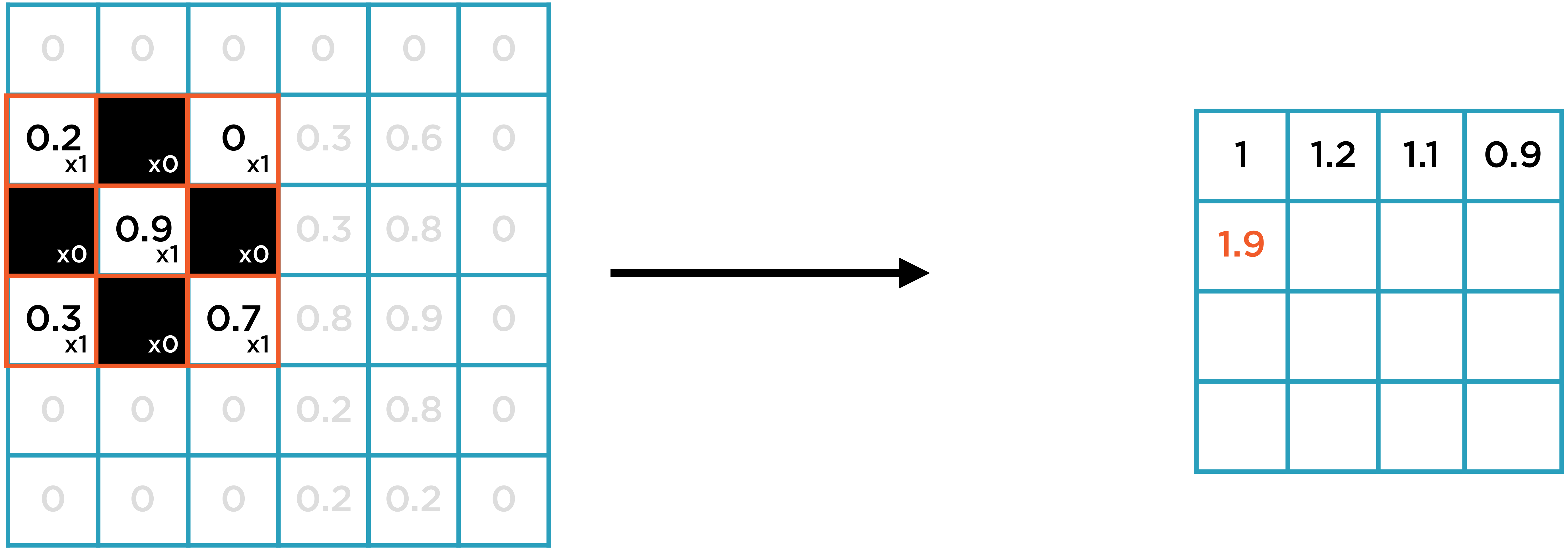
Convolution



Convolution



Convolution



Matrix

Convolution Result

Convolution

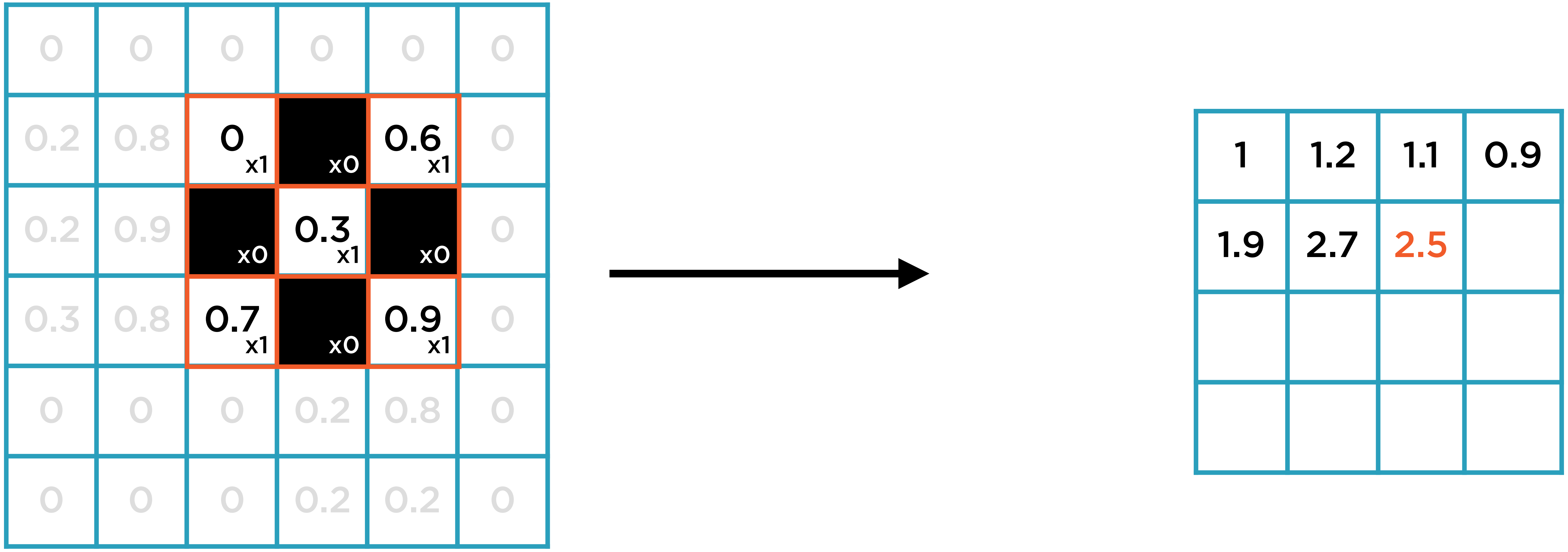
0	0	0	0	0	0
0.2	0.8 x1	x0	0.3 x1	0.6	0
0.2	x0	0 x1	x0	0.8	0
0.3	0.8 x1	x0	0.8 x1	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Matrix

1	1.2	1.1	0.9
1.9	2.7		

Convolution Result

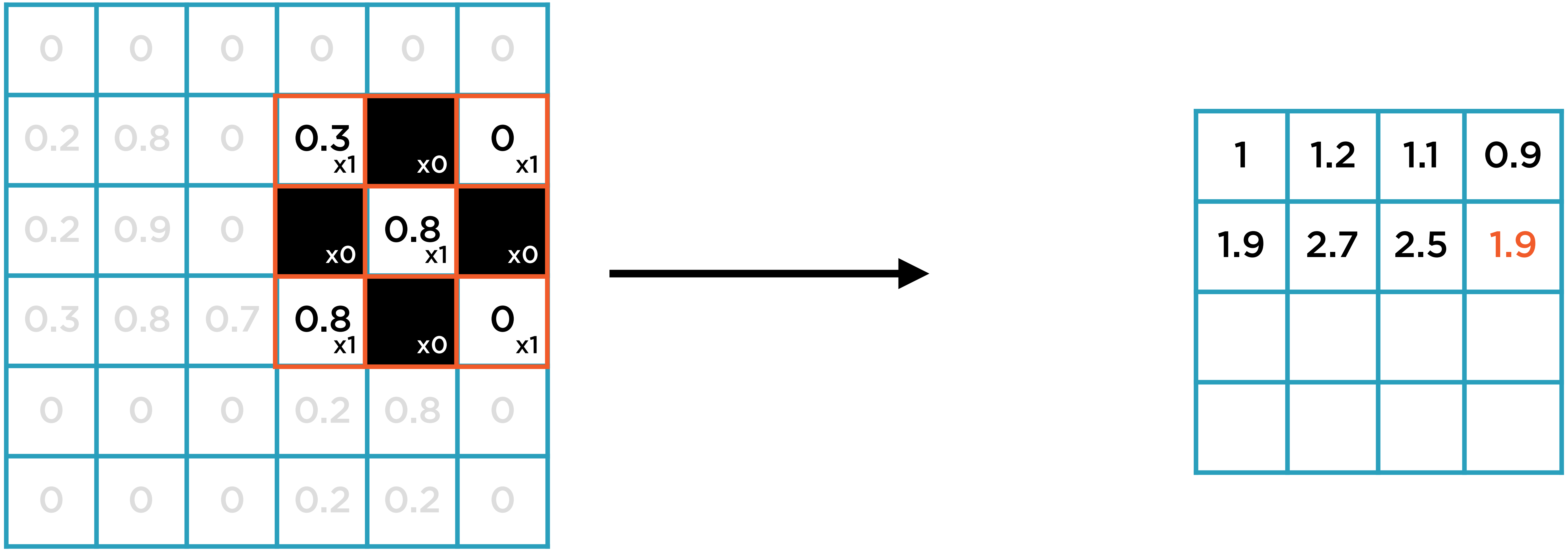
Convolution



Matrix

Convolution Result

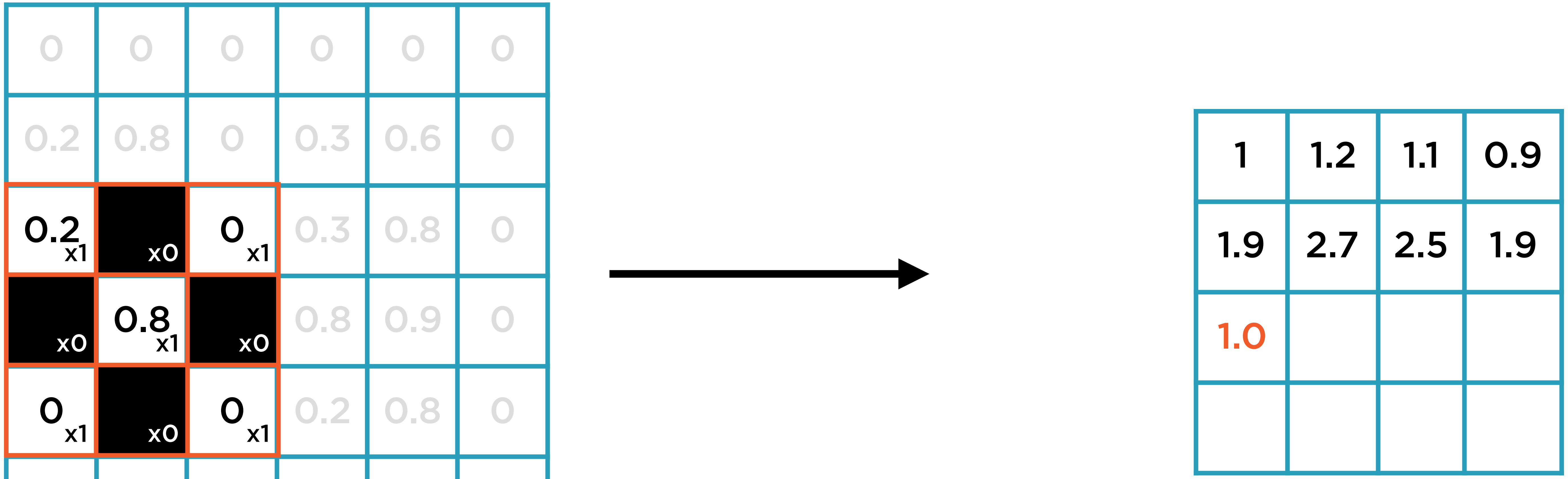
Convolution



Matrix

Convolution Result

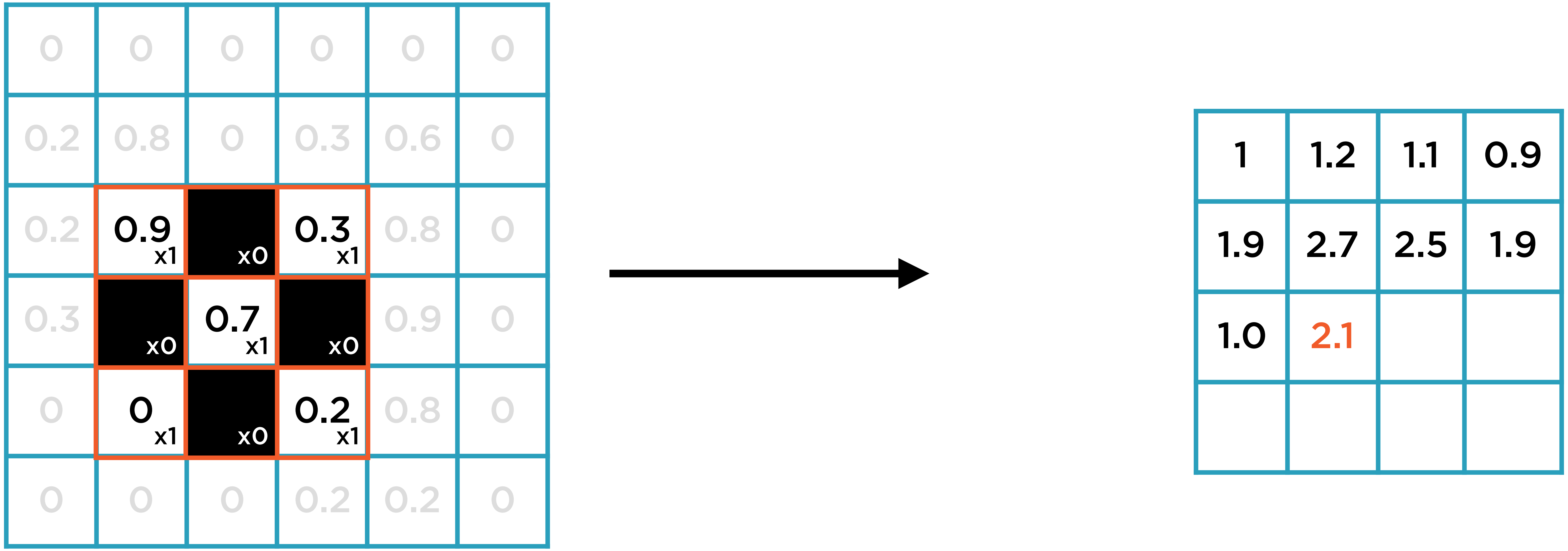
Convolution



Matrix

Convolution Result

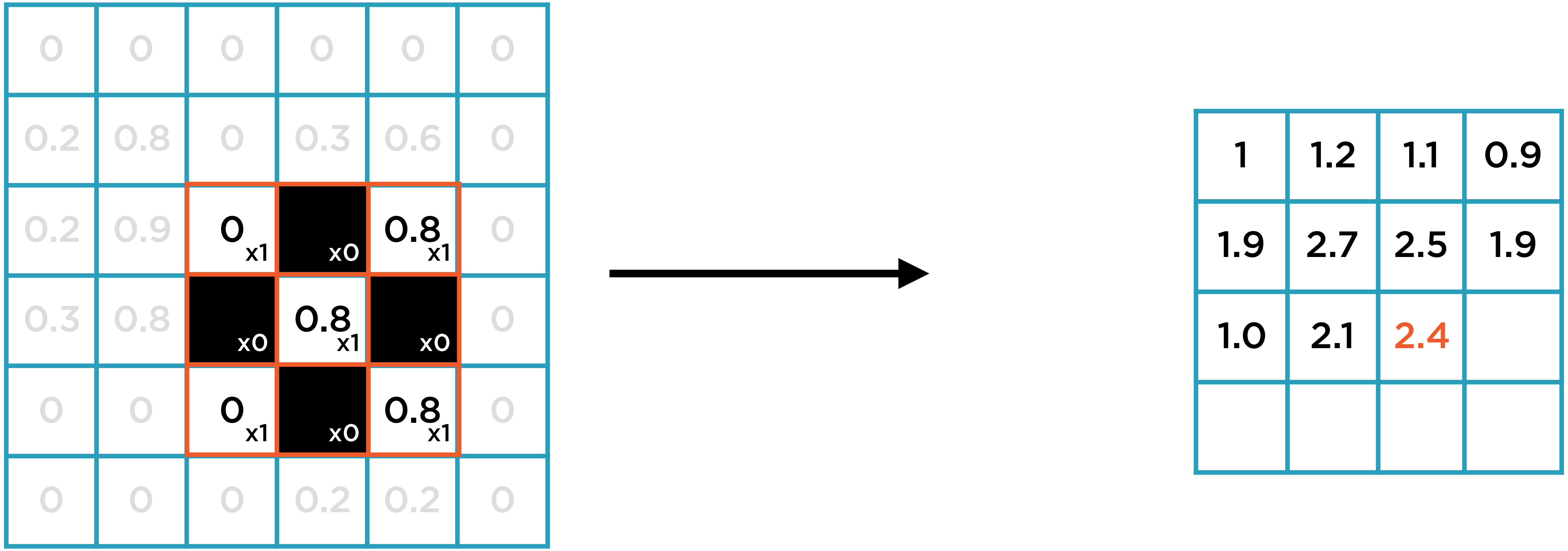
Convolution



Matrix

Convolution Result

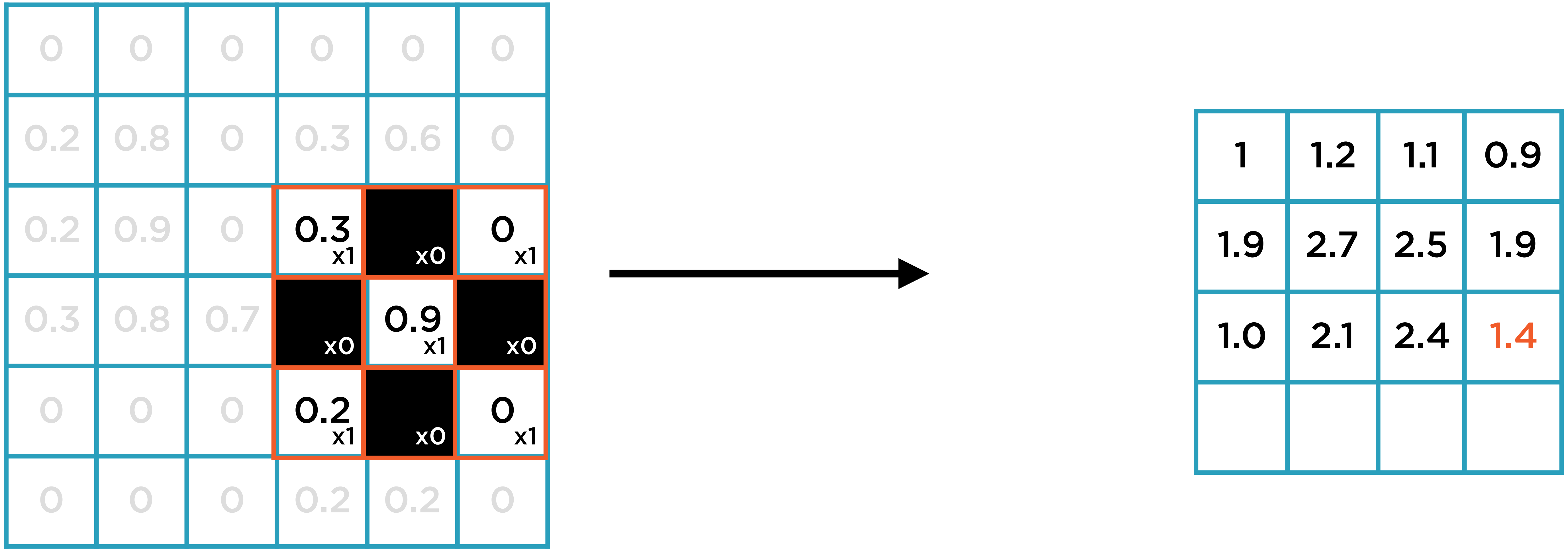
Convolution



Matrix

Convolution Result

Convolution



Matrix

Convolution Result

Convolution

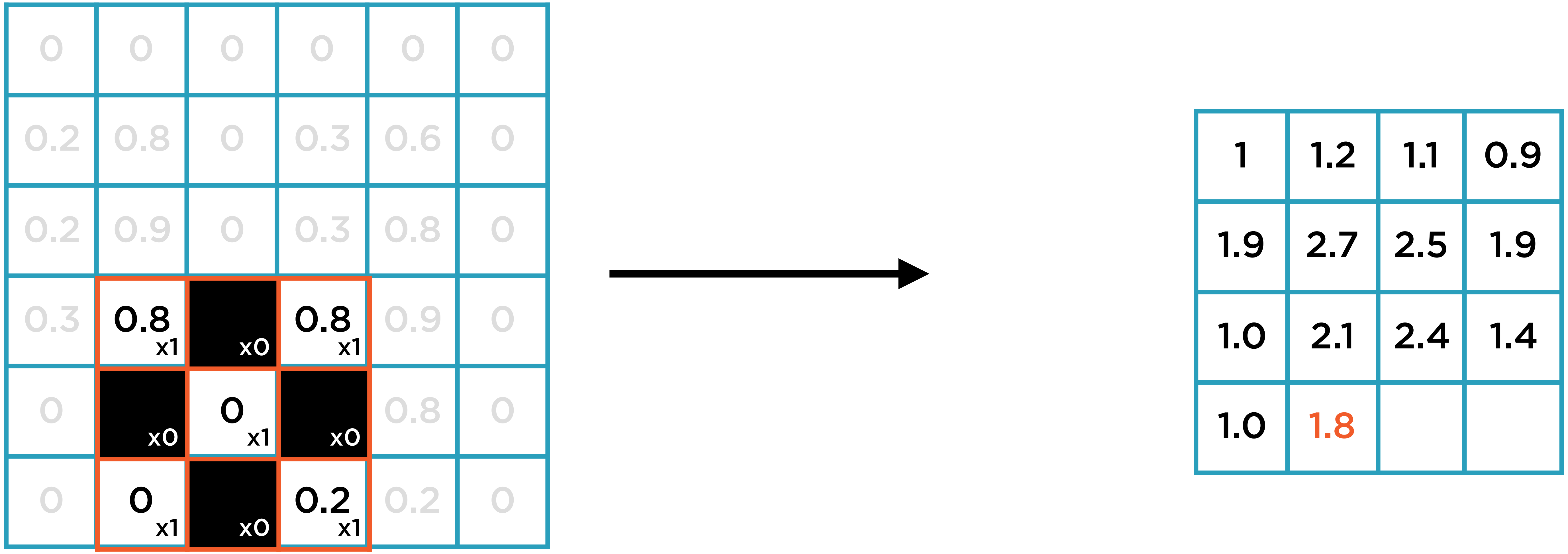
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3 x1	x0	0.7 x1	0.8	0.9	0
x0	0 x1	x0	0.2	0.8	0
0 x1	x0	0 x1	0.2	0.2	0

Matrix

1	1.2	1.1	0.9
1.9	2.7	2.5	1.9
1.0	2.1	2.4	1.4
1.0			

Convolution Result

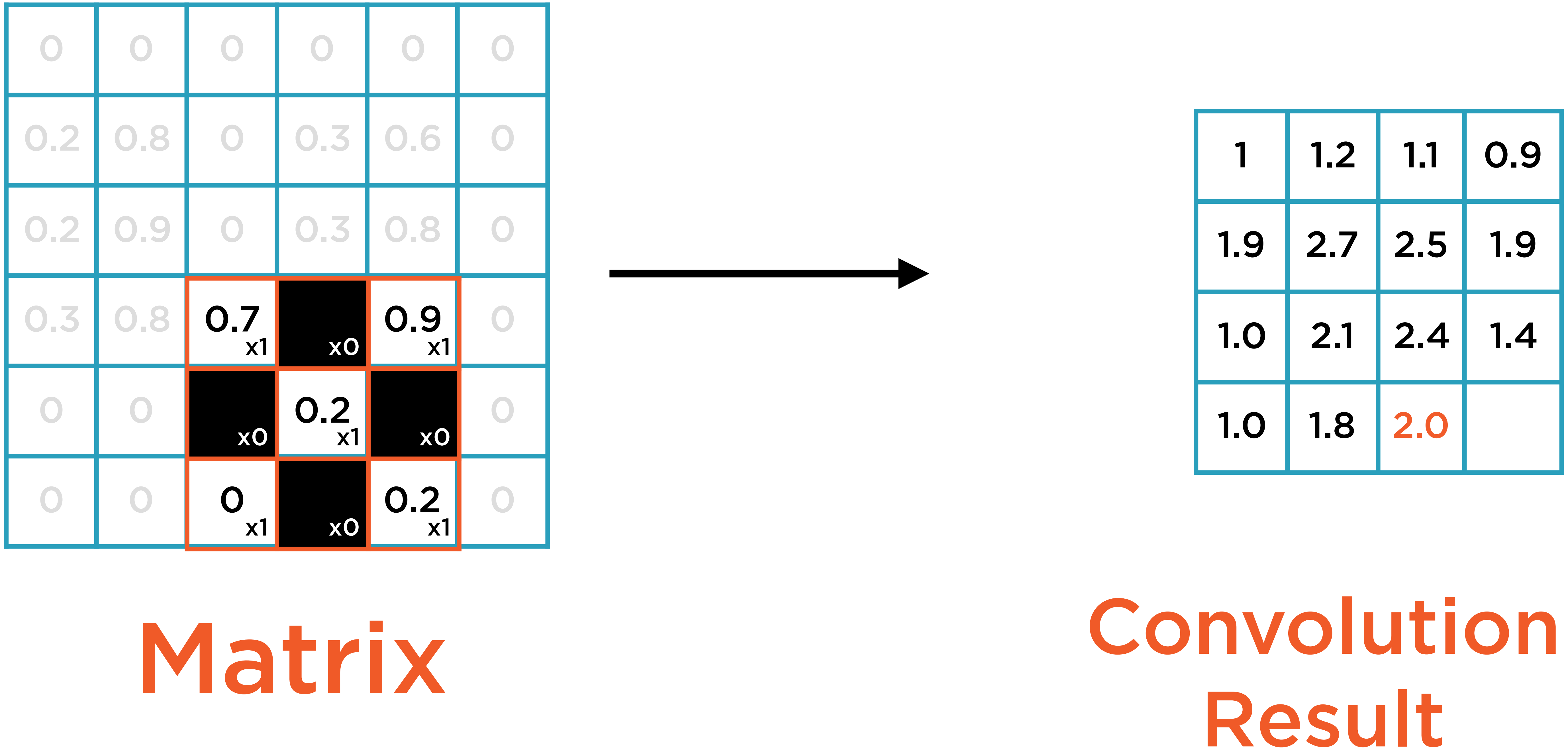
Convolution



Matrix

Convolution Result

Convolution



Convolution

0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8 x1	x0	0 x1
0	0	0	x0	0.8 x1	x0
0	0	0	0.2 x1	x0	0 x1

Matrix

1	1.2	1.1	0.9
1.9	2.7	2.5	1.9
1.0	2.1	2.4	1.4
1.0	1.8	2.0	1.8

Convolution Result

Demo

Performing edge detection using

- Roberts cross detection

- Sobel edge detection

Canny Edge Detection

Canny Edge Detection

Multi-stage algorithm, named after its inventor, that detects useful structural information from images with greatly reduced data processing

Enhances the signal to noise ratio thus producing better edge detection with noisy images

Canny Edge Detection

Smoothen Image

Remove noise with Gaussian filter

Compute Intensity Gradients

Edge gradient and direction as before

Suppress Spurious Responses

Set all gradient values except maxima to 0

Apply Double Threshold

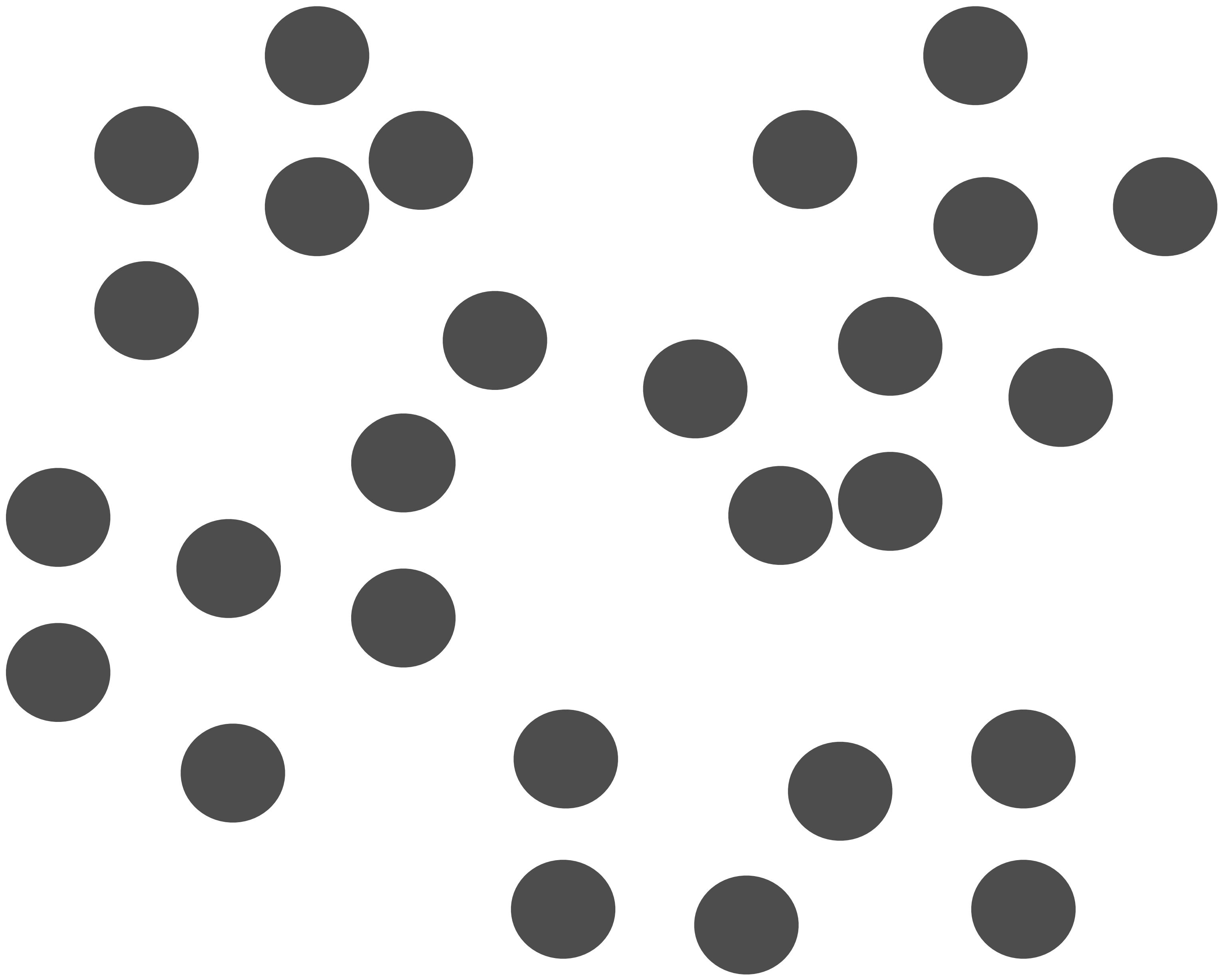
Filter out edge pixels with weak gradients; preserve edge pixels with strong gradients

Edge Tracking by Hysteresis

Eliminate all weak edges not connected to strong edges

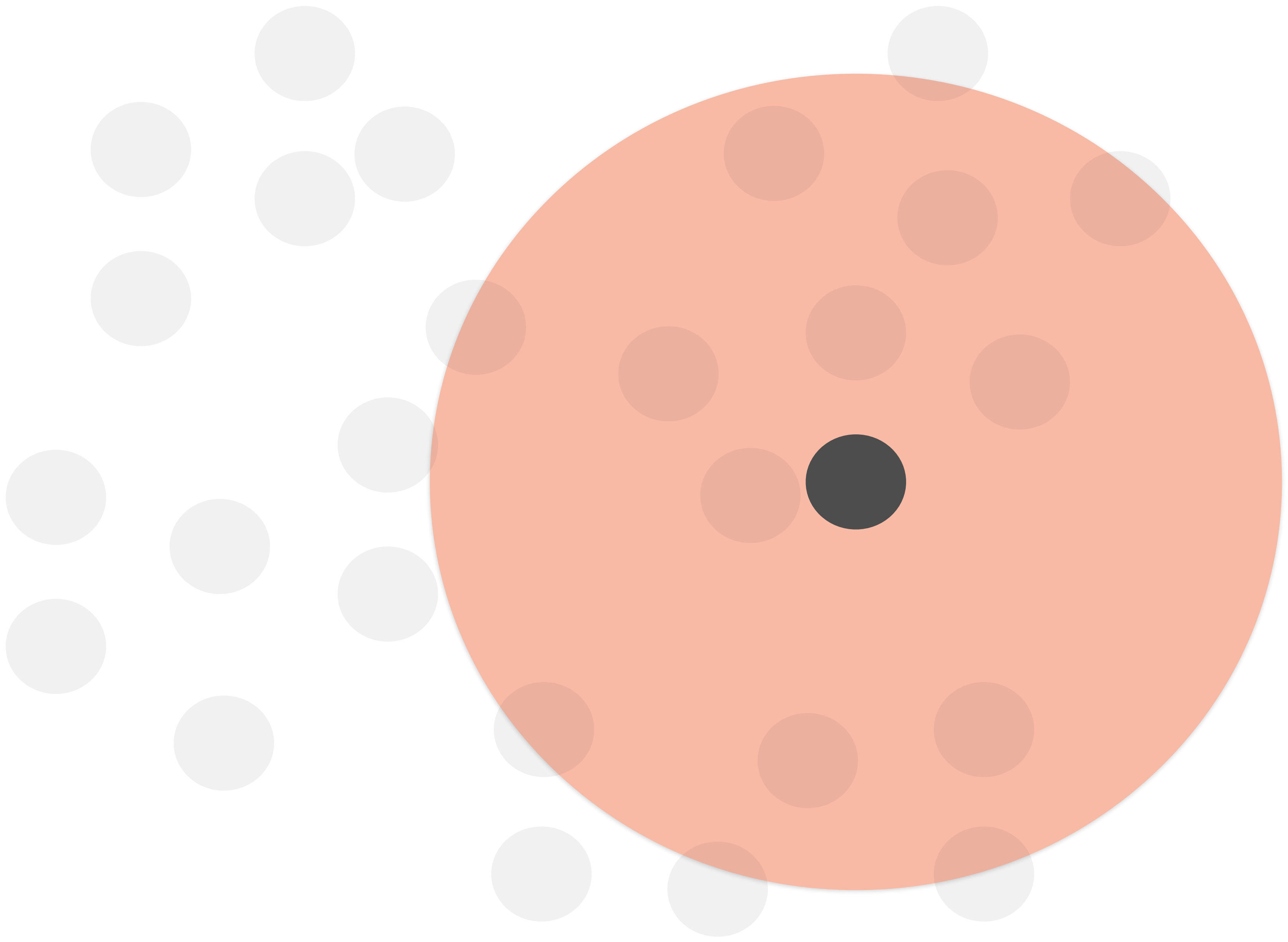
Gaussian Filter for Smoothing

Start with a set of points in space



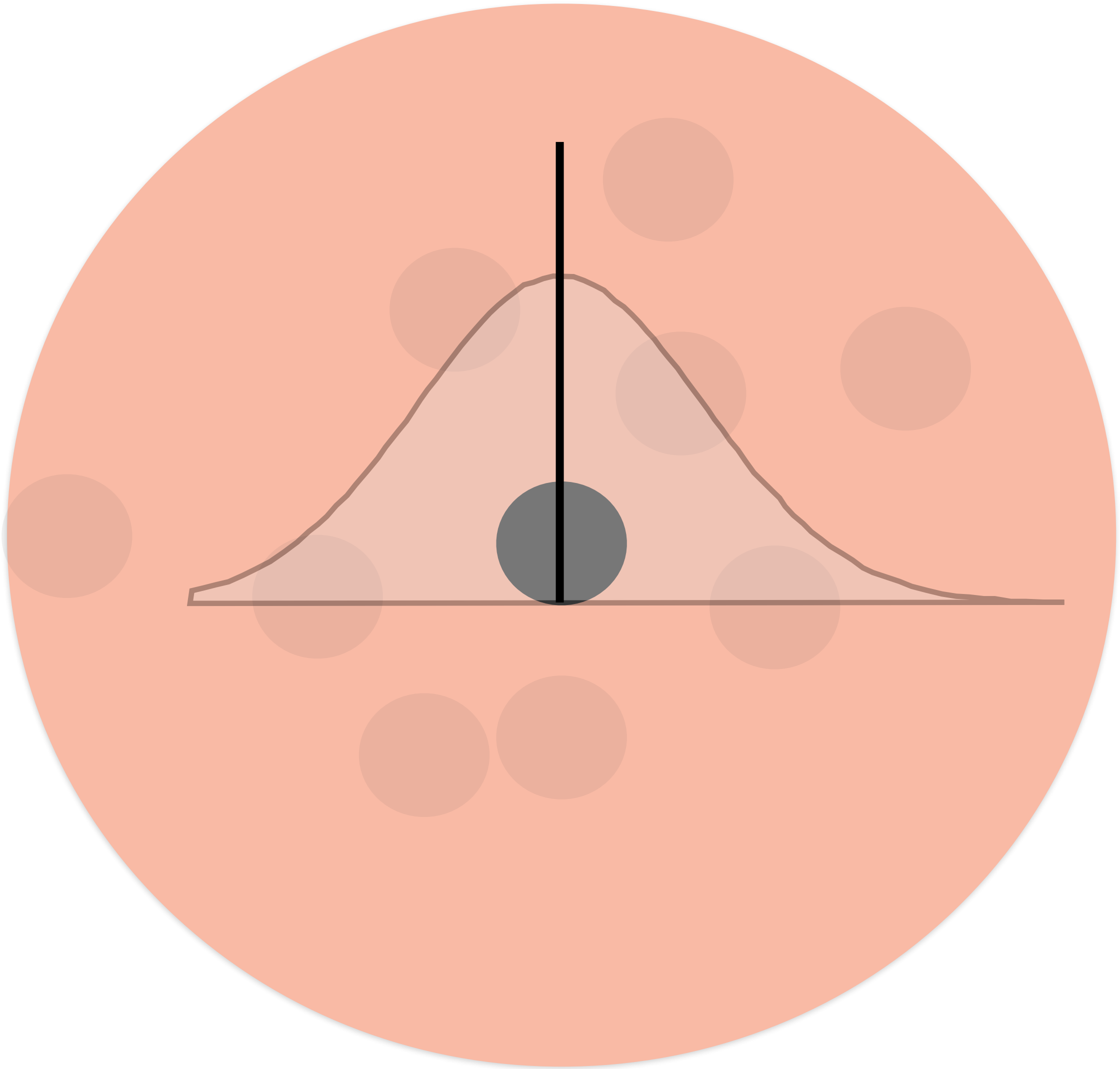
Gaussian Filter for Smoothing

Define a neighborhood for each point

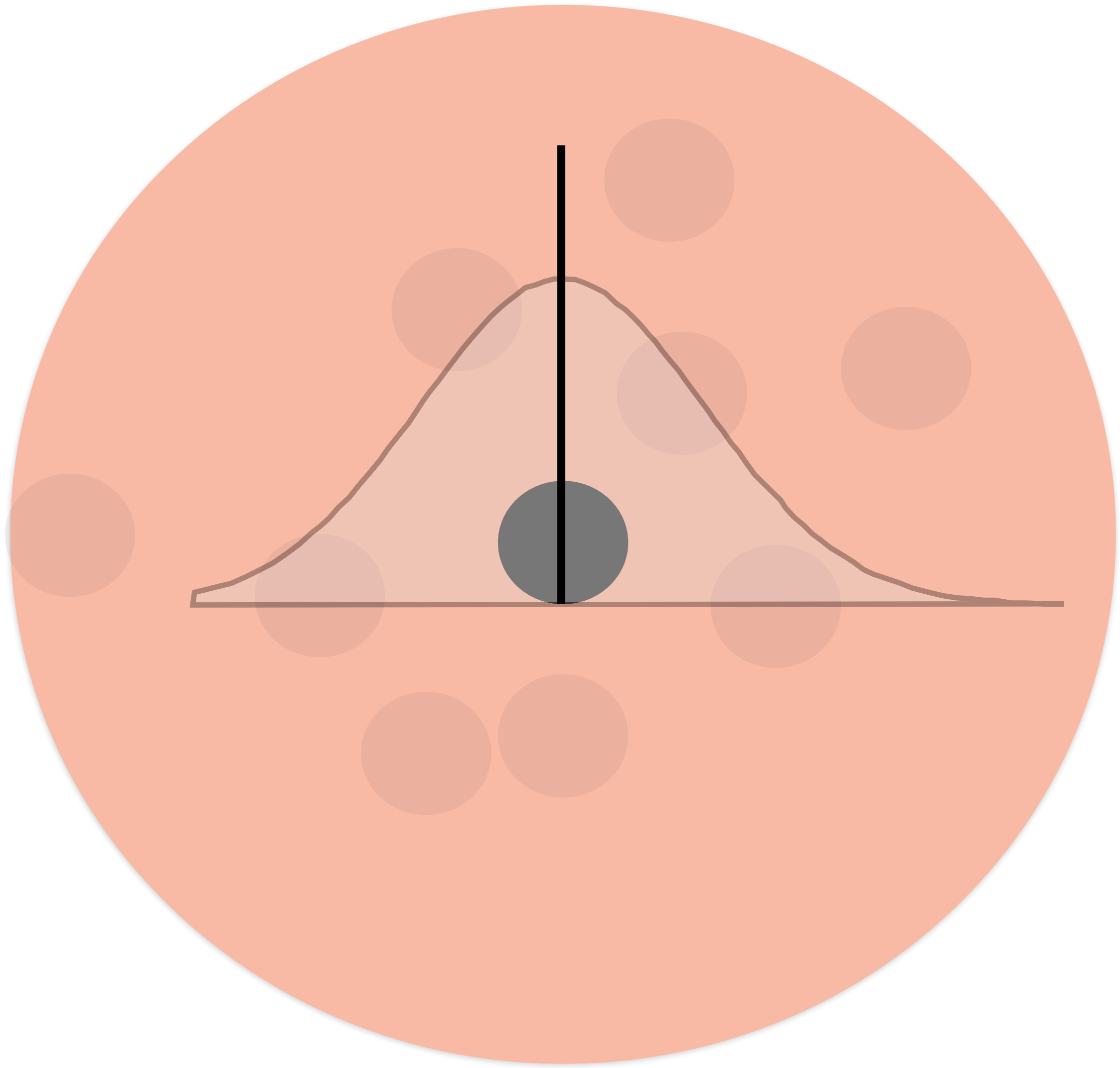


Gaussian (RBF) Kernel

Probability-weighted sum of points



What probability distribution?

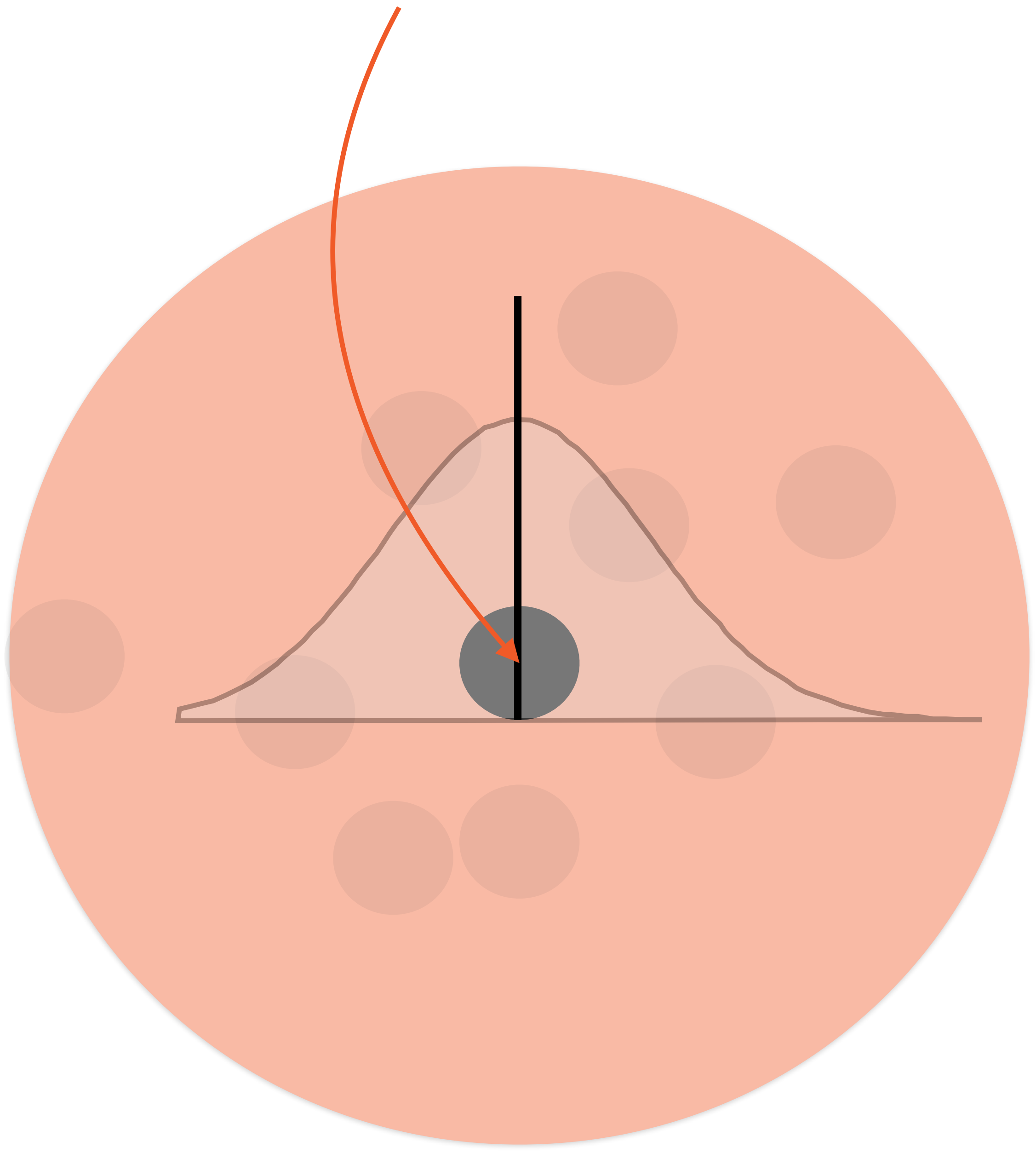


Gaussian (RBF) Kernel

Gaussian probability distribution

Defined by

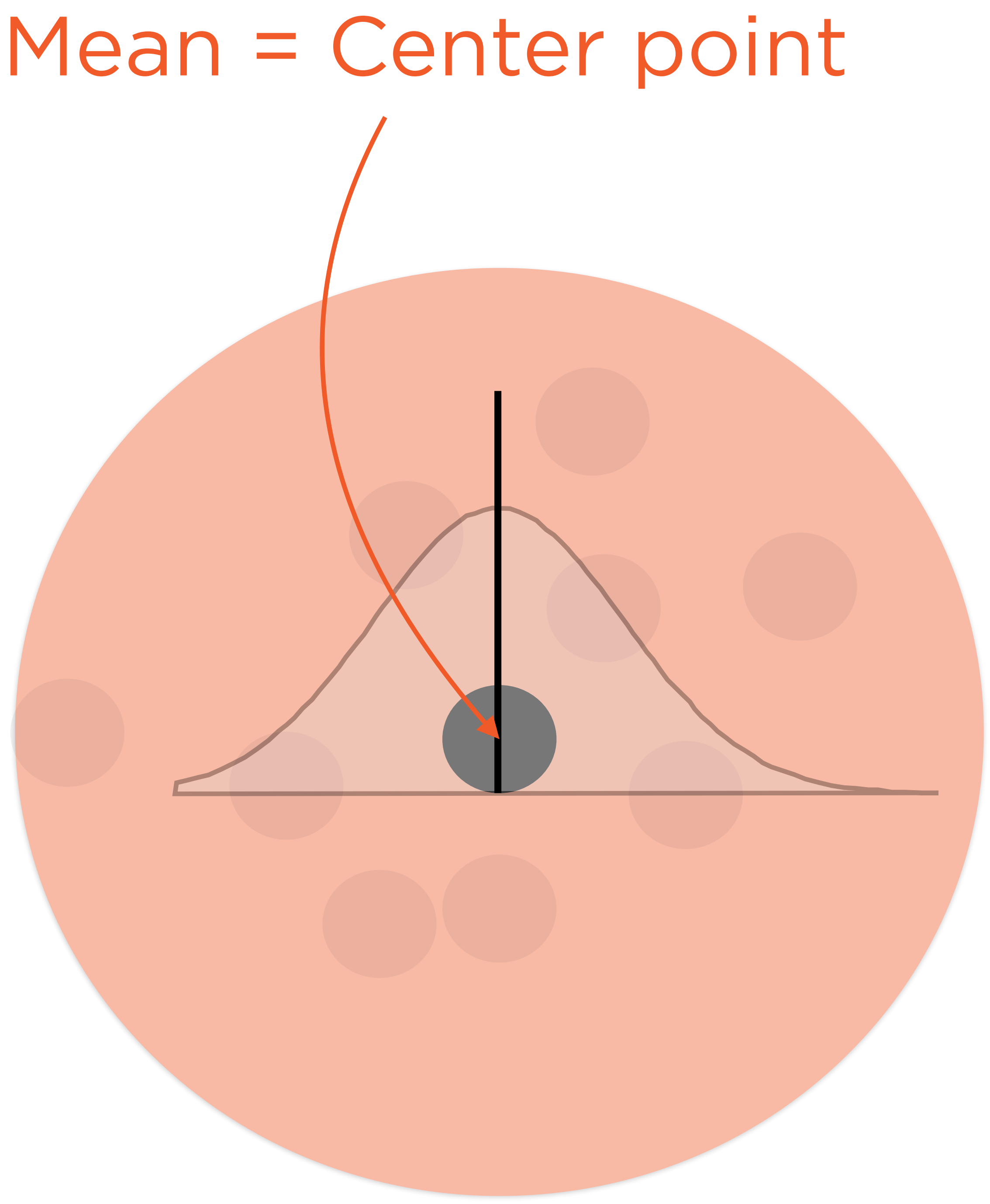
- mean μ
- standard deviation σ



Gaussian Filter

Mean μ = center point

Standard deviation σ ~ bandwidth



Gaussian (RBF) Kernel

Larger value of sigma: less sensitive to noise, detects only large edges

Smaller value of sigma: more sensitive to noise, detects even fine edges

Need to strike a fine balance

Sigma influences quality of feature extraction

Demo

Performing Canny edge detection

Summary

Working with images is increasingly important

scikit-image is an image processing toolkit

Performed image manipulation, contour and convex hull determination

Edge detection using Sobel, Roberts and Canny edge detectors