Building Image Classification Models

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

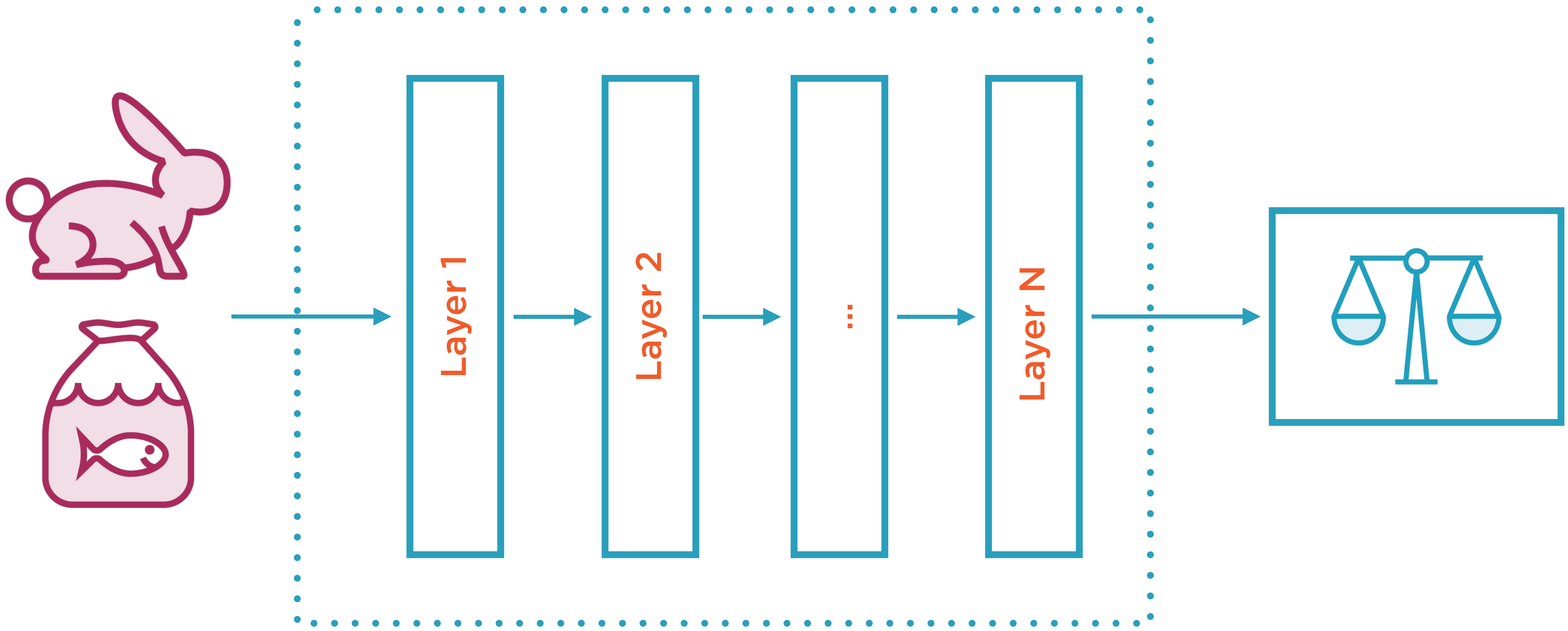
Image classification models

Convolutional layers and pooling layers

Convolutional Neural Networks (CNNs) for image classification

Implementing CNNs in tf.keras for image classification

Neural Networks for Image Classification

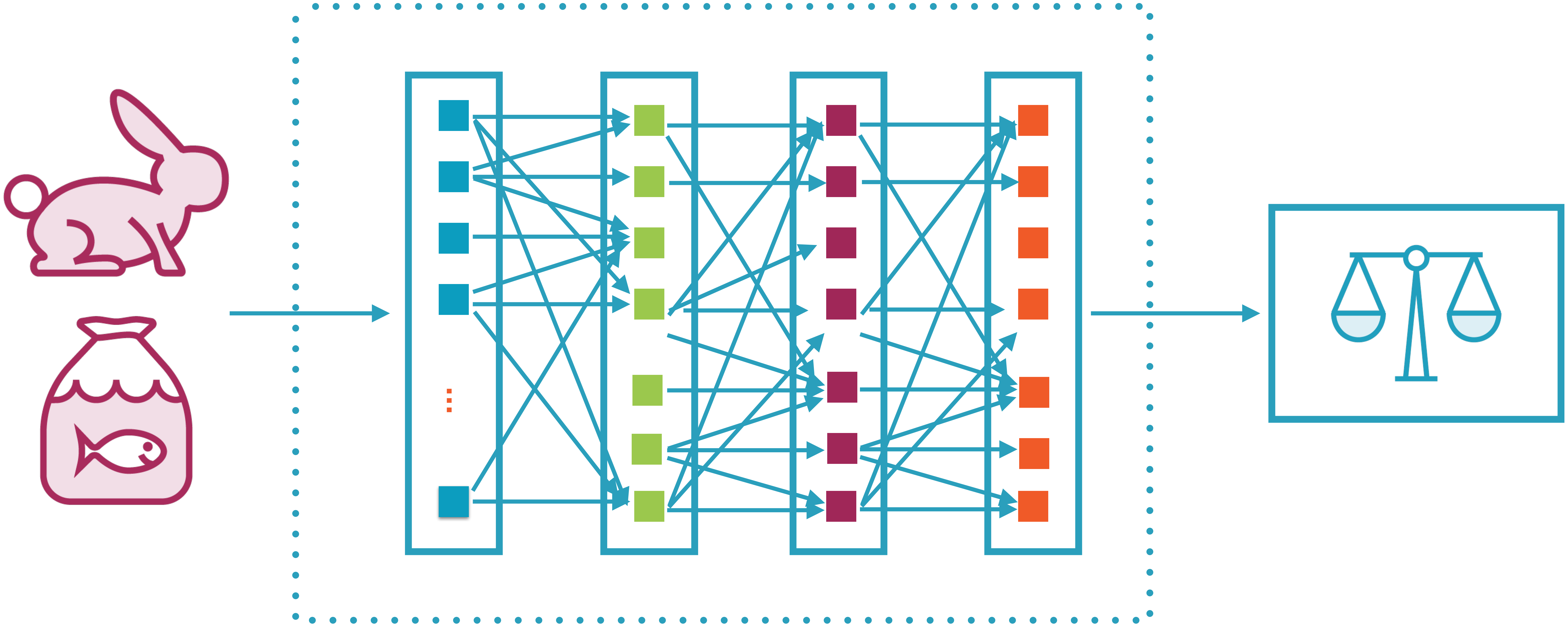


Corpus of Images

Layers in a neural network

ML-based Classifier

Neural Networks for Image Classification



Corpus of Images

Each layer consists of individual interconnected neurons

ML-based Classifier

Parameter Explosion

Consider a 100 x 100 pixel image (10,000 pixels)

If first layer = 10,000 neurons

Interconnections ~ O(10,000 * 10,000)

100 million parameters to train neural network!

Parameter Explosion

Dense, fully connected neural networks can’t cope

Also do not provide feature extraction with location invariance

Convolutional neural networks to the rescue

Introducing Convolutional Neural Networks

Dense neural networks do not consider the spatial aspects of images

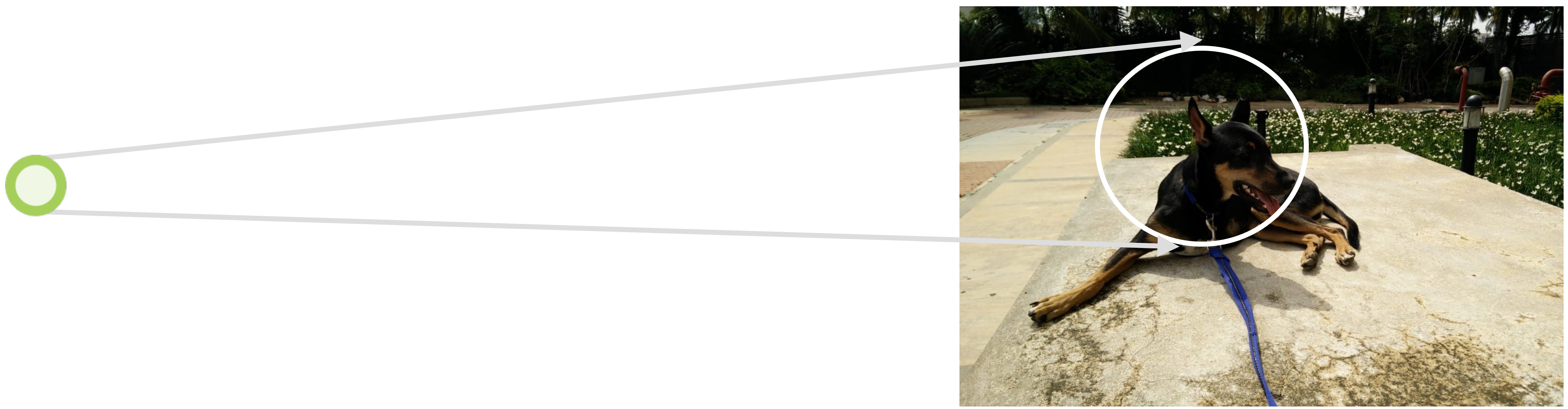
Viewing an Image

All neurons in the eye don't see the entire image

Viewing an Image

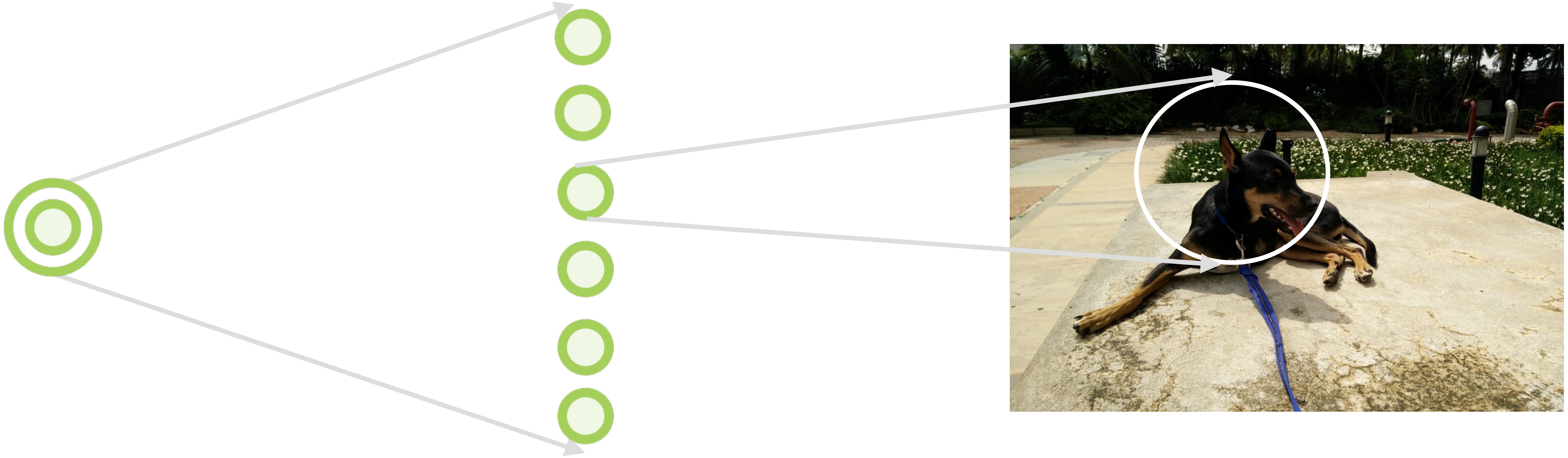
Each neuron has its own local receptive field

Viewing an Image



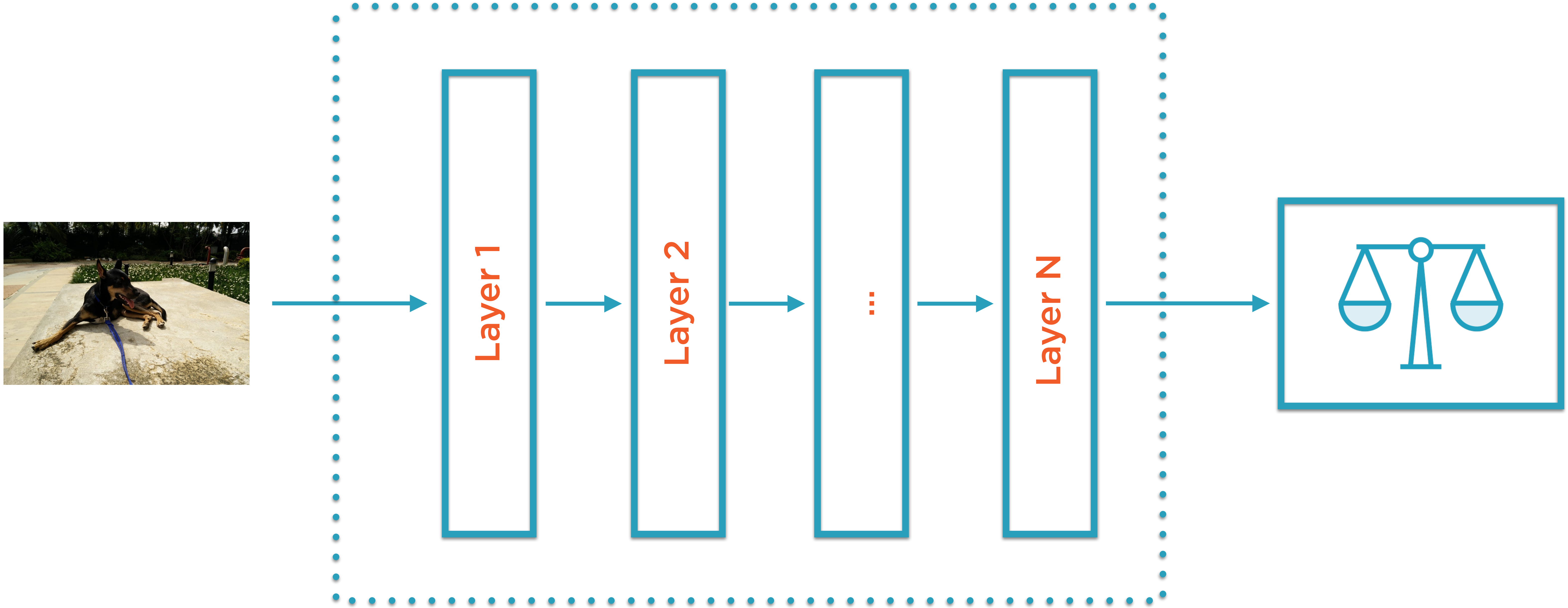
It reacts only to visual stimuli located in its receptive field

Viewing an Image



Some neurons react to more complex patterns that are combinations of lower level patterns

Neural Networks



Sounds like a classic neural network problem

Convolutional neural networks consider the **spatial** aspects of image and **aggregate** information from local fields

Convolutional Neural Networks

Eye perceives visual stimulus in 2D visual field

“Local receptive field”

Eye sends 2D image to visual cortex

Convolutional Neural Networks

Visual cortex adds depth perception

Individual neurons in cortex focus on small field

Convolutional Neural Networks

CNNs perform spectacularly well at many tasks

Particularly at image recognition

Dramatically fewer parameters than DNN with similar performance

Inspirations for CNNs

Two Dimensions

Data comes in expressed in 2D

Local Receptive Fields

Neurons focus on narrow portions

Two Kinds of Layers in CNNs

Convolution

Local receptive field

Pooling

Subsampling of inputs

Convolution

Two Kinds of Layers in CNNs

Convolution

Local receptive field

Pooling

Subsampling of inputs

Convolution

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

Convolution

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

e.g. a matrix of pixels
representing an image

Convolution

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

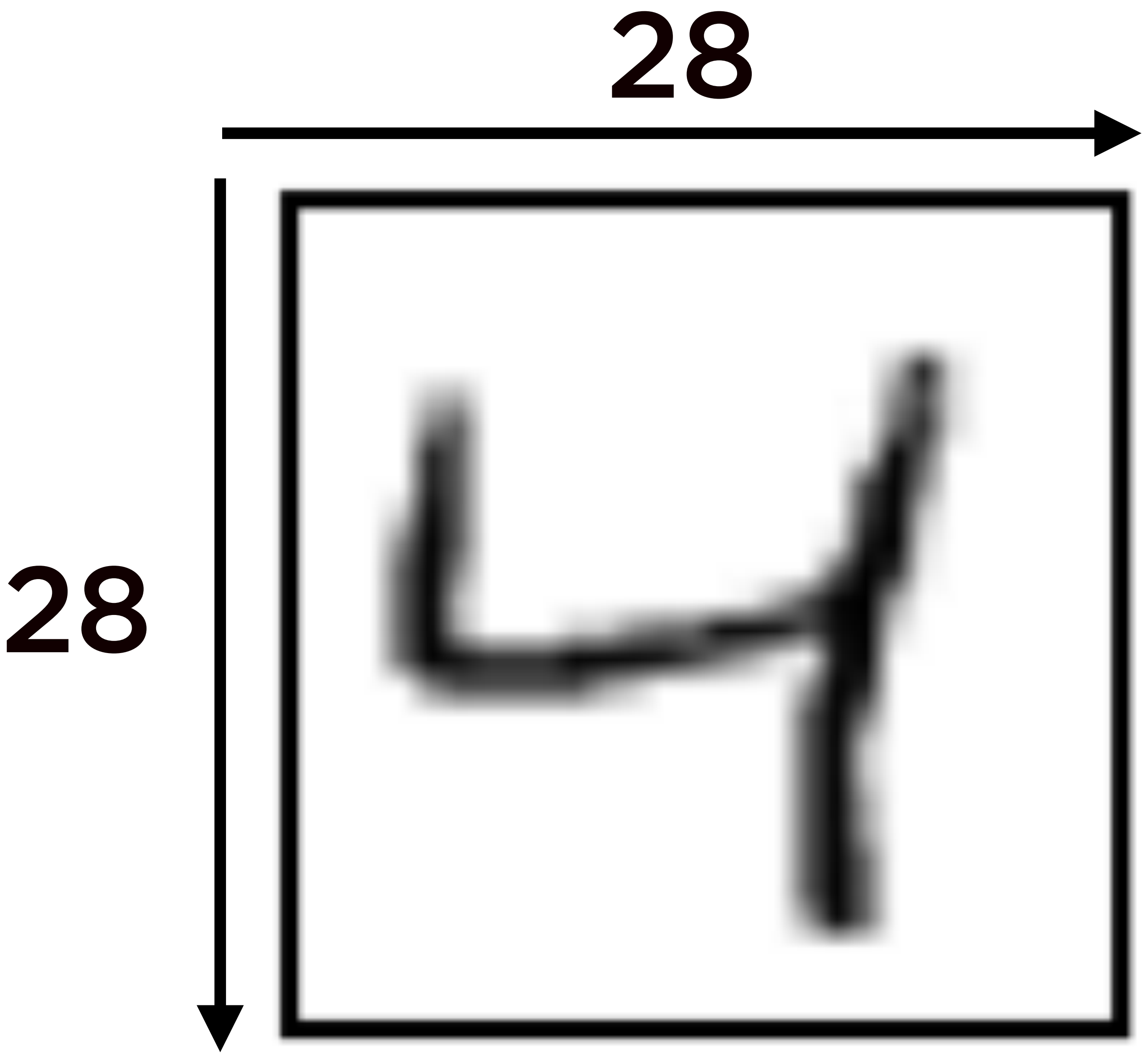
Often called a kernel or filter

Convolution

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

Kernel is applied element-wise
in sliding-window fashion

Representing Images as Matrices



= 784 pixels

Representing Images as Matrices

6

6

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

= 36 pixels

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

3

3

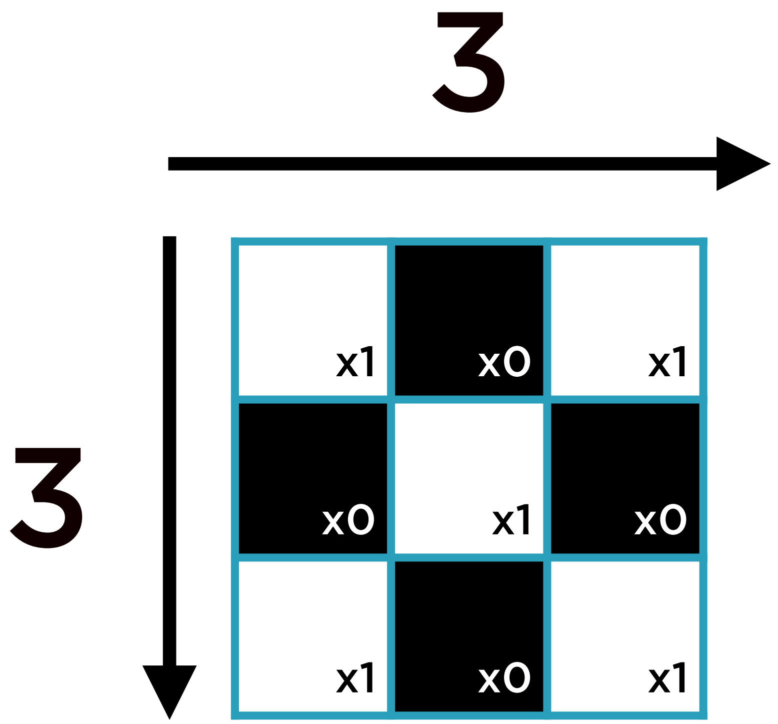
1	0	1
0	1	0
1	0	1

Kernel

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



Kernel

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

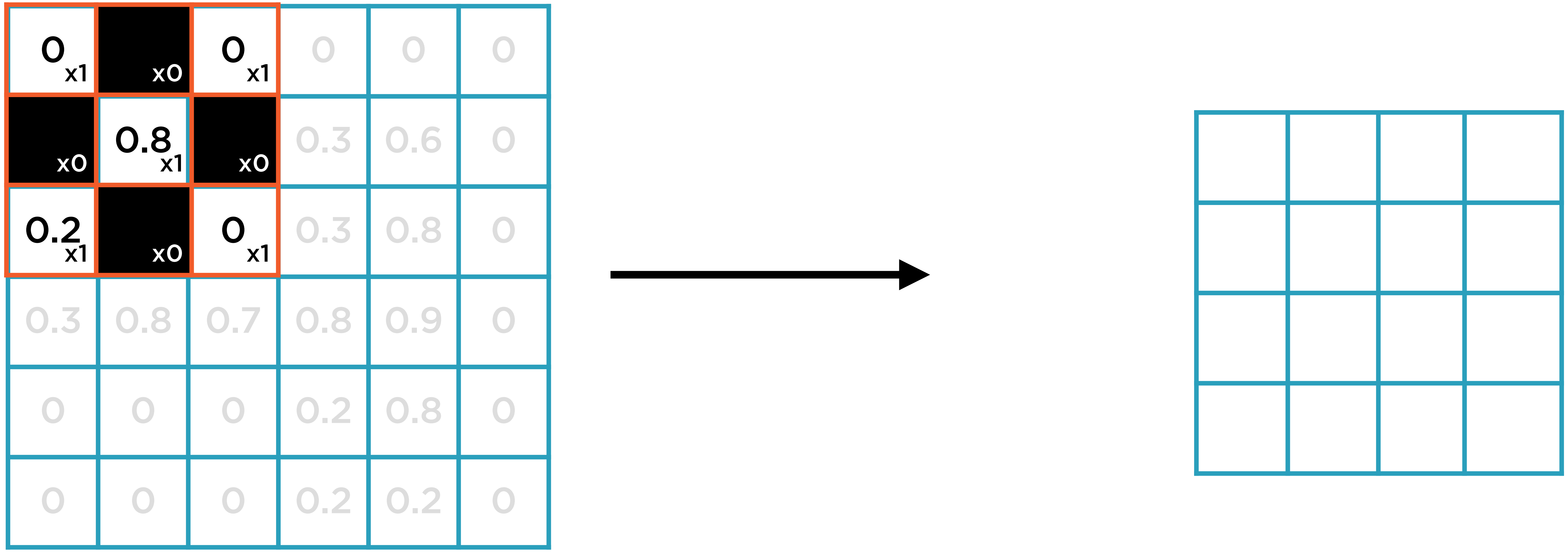
4

4

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

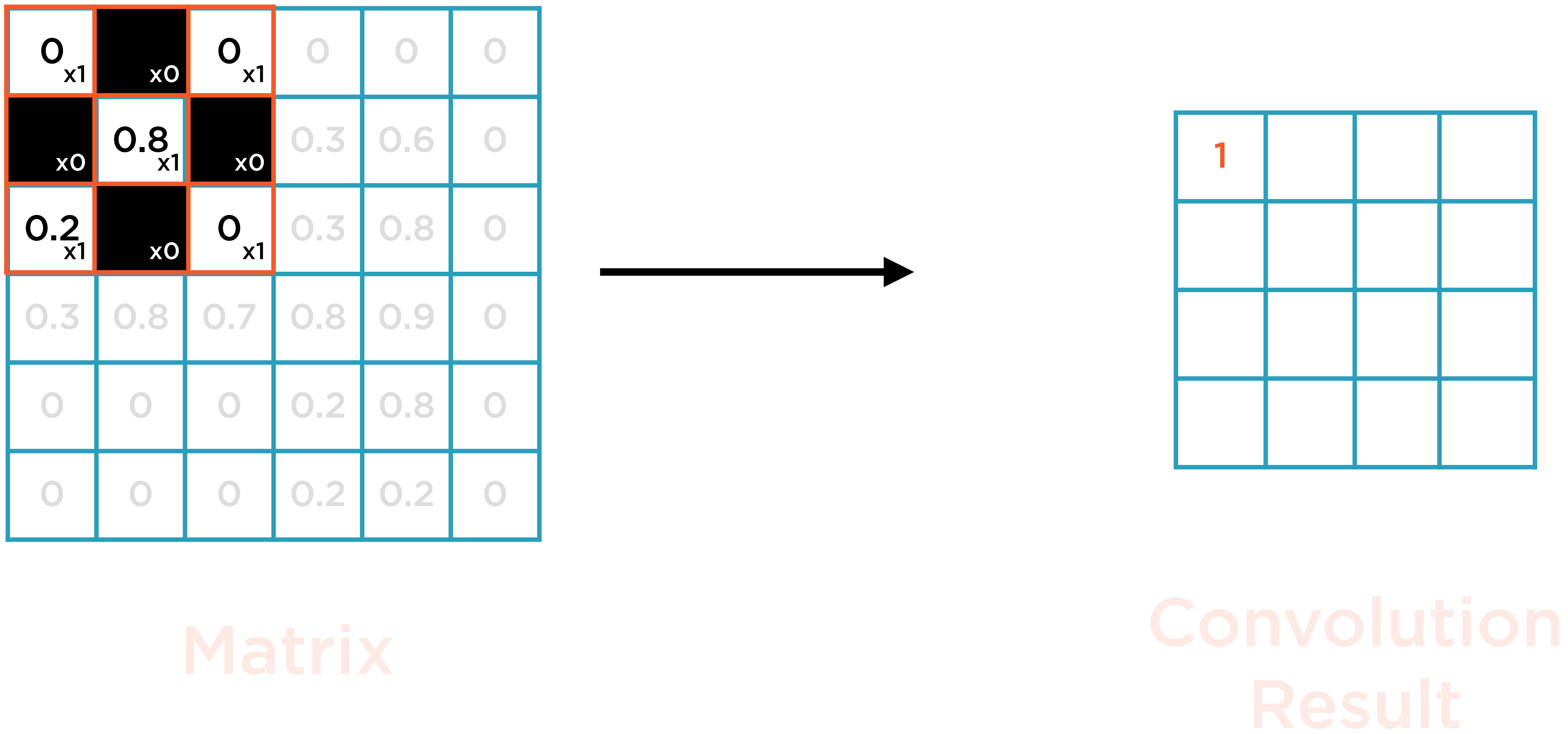
Convolution



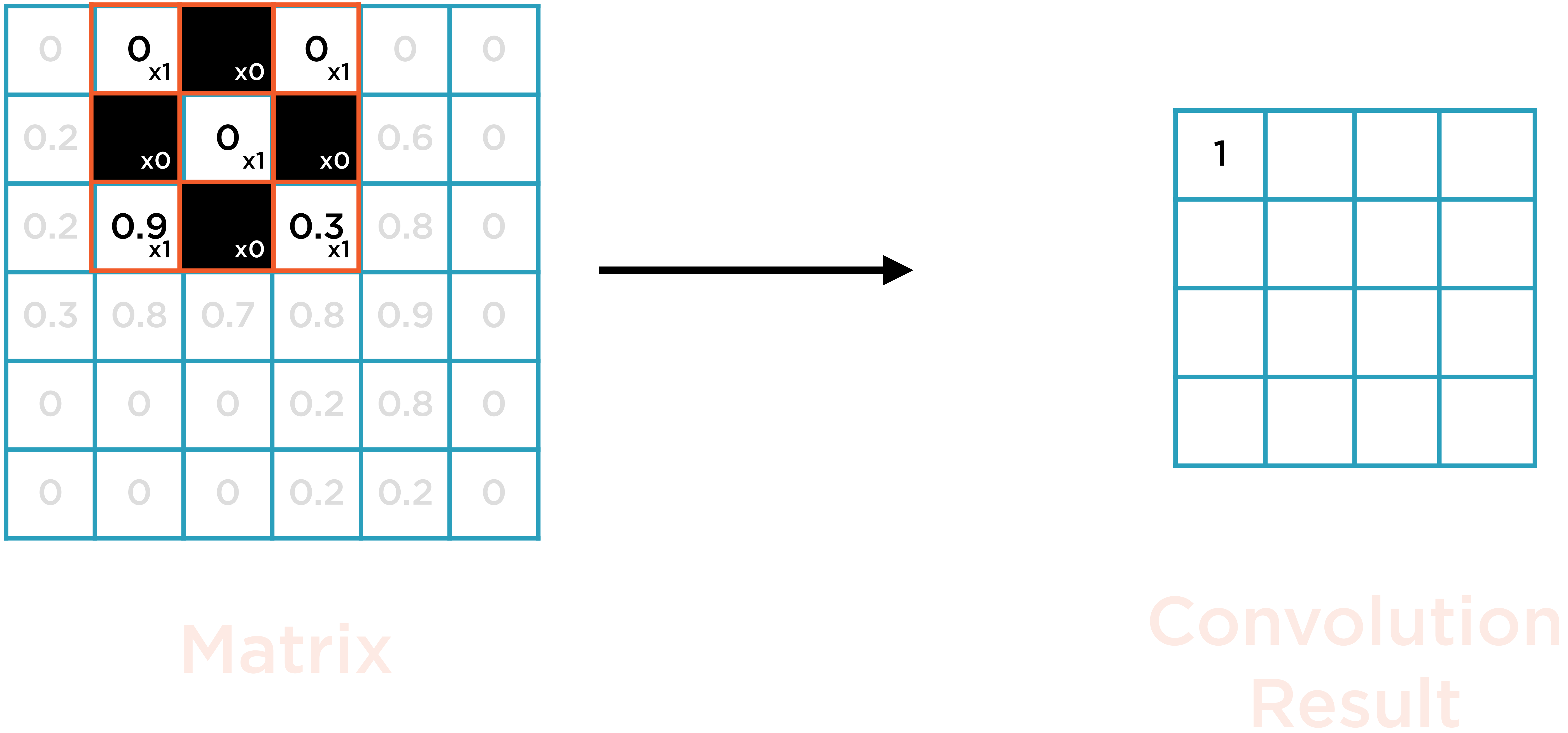
Matrix

Convolution Result

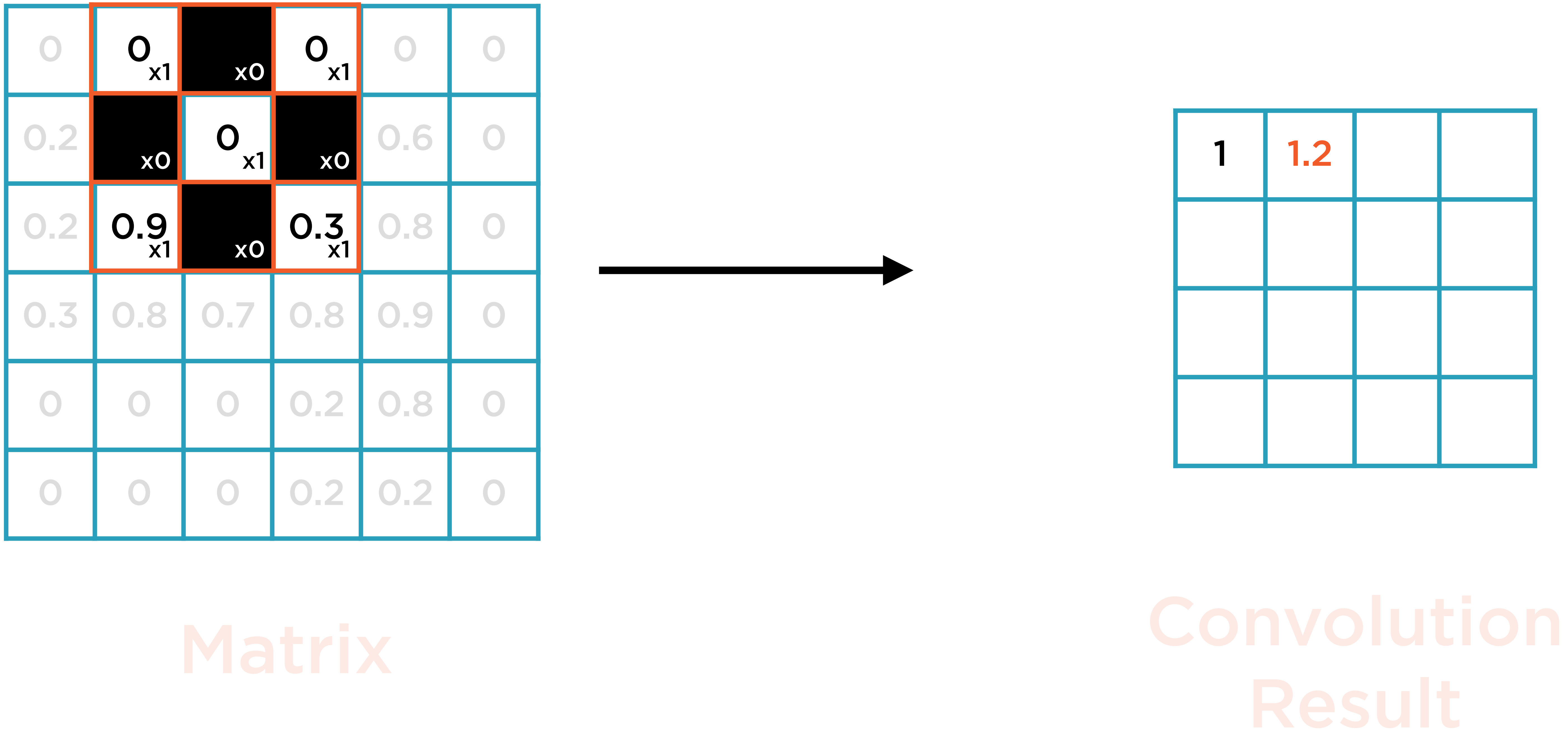
Convolution



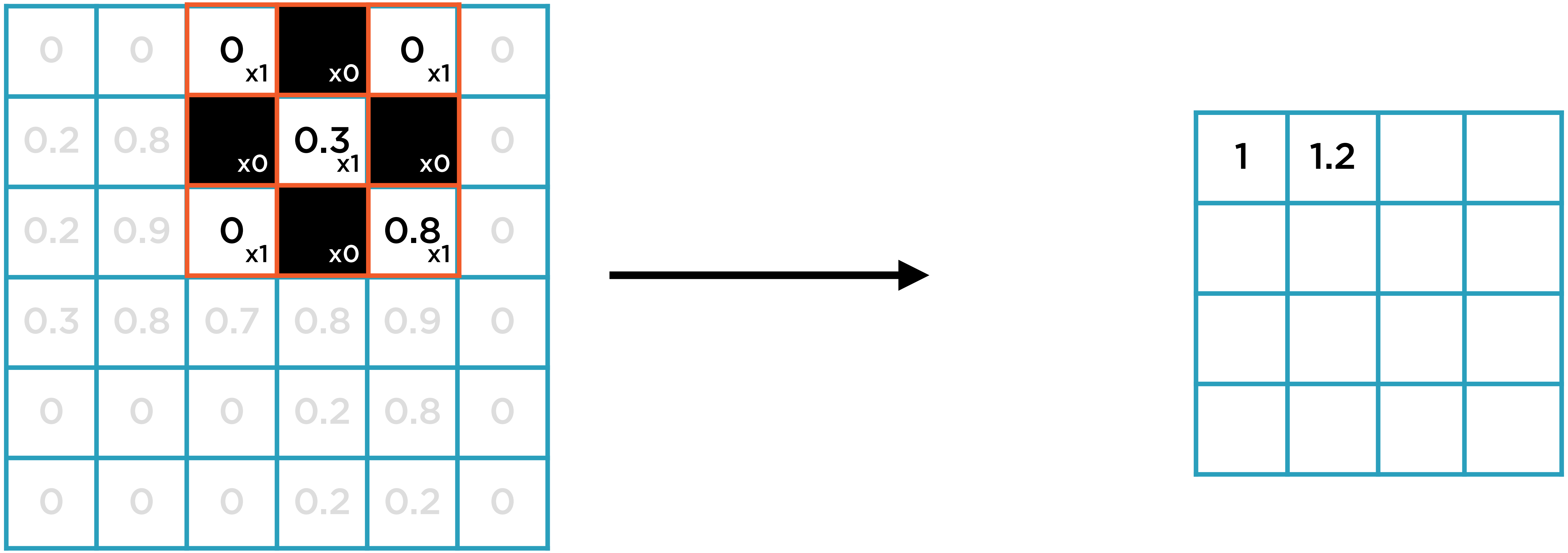
Convolution



Convolution



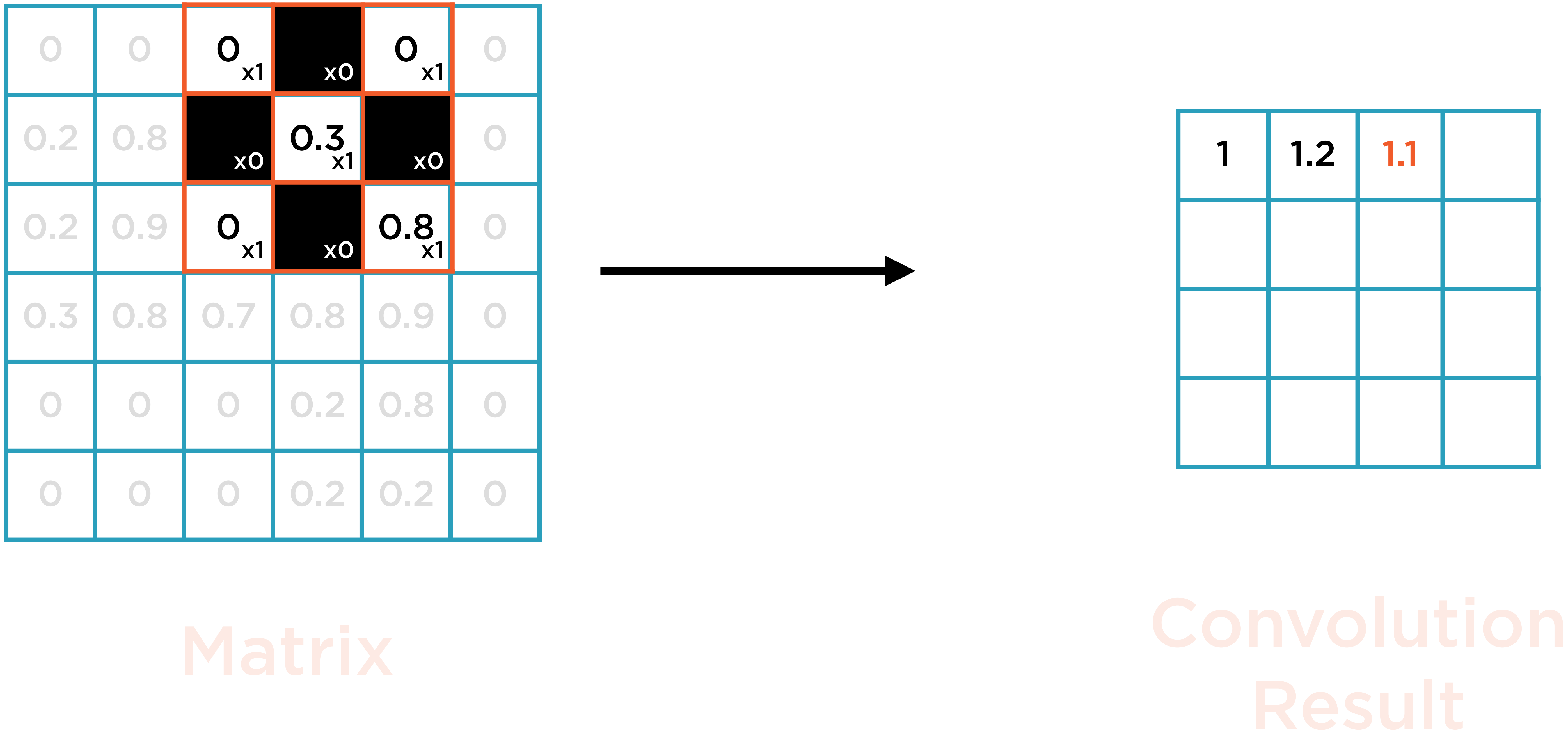
Convolution



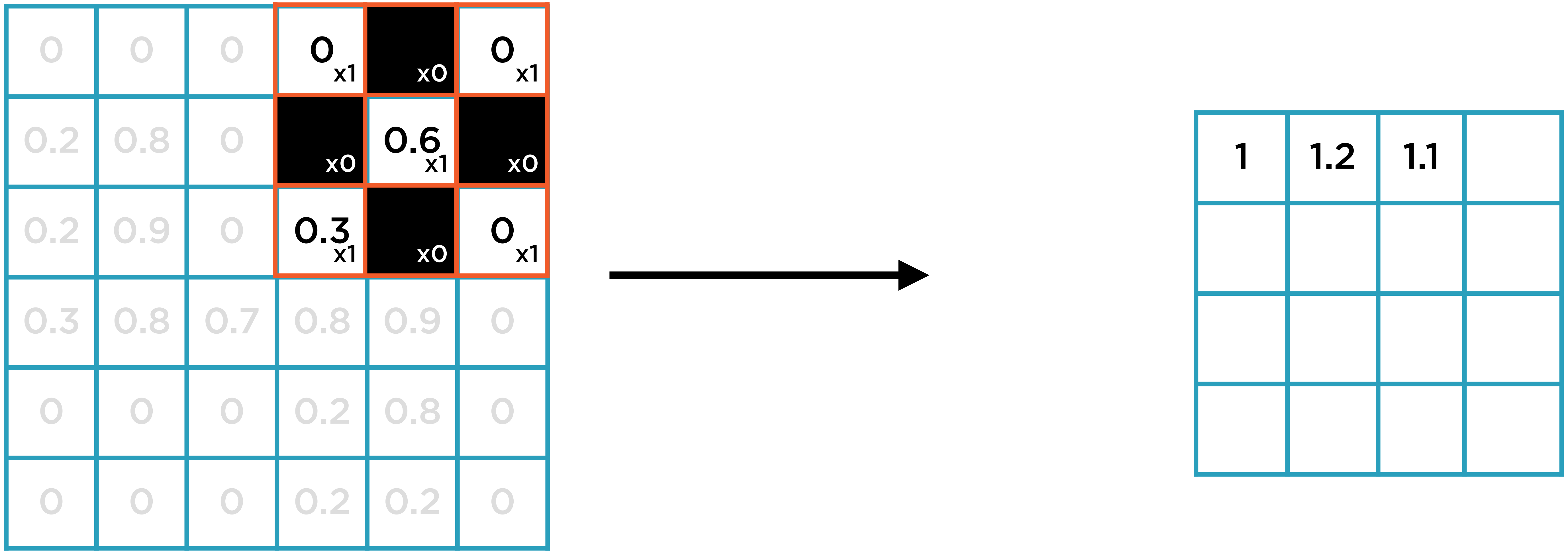
Matrix

Convolution Result

Convolution



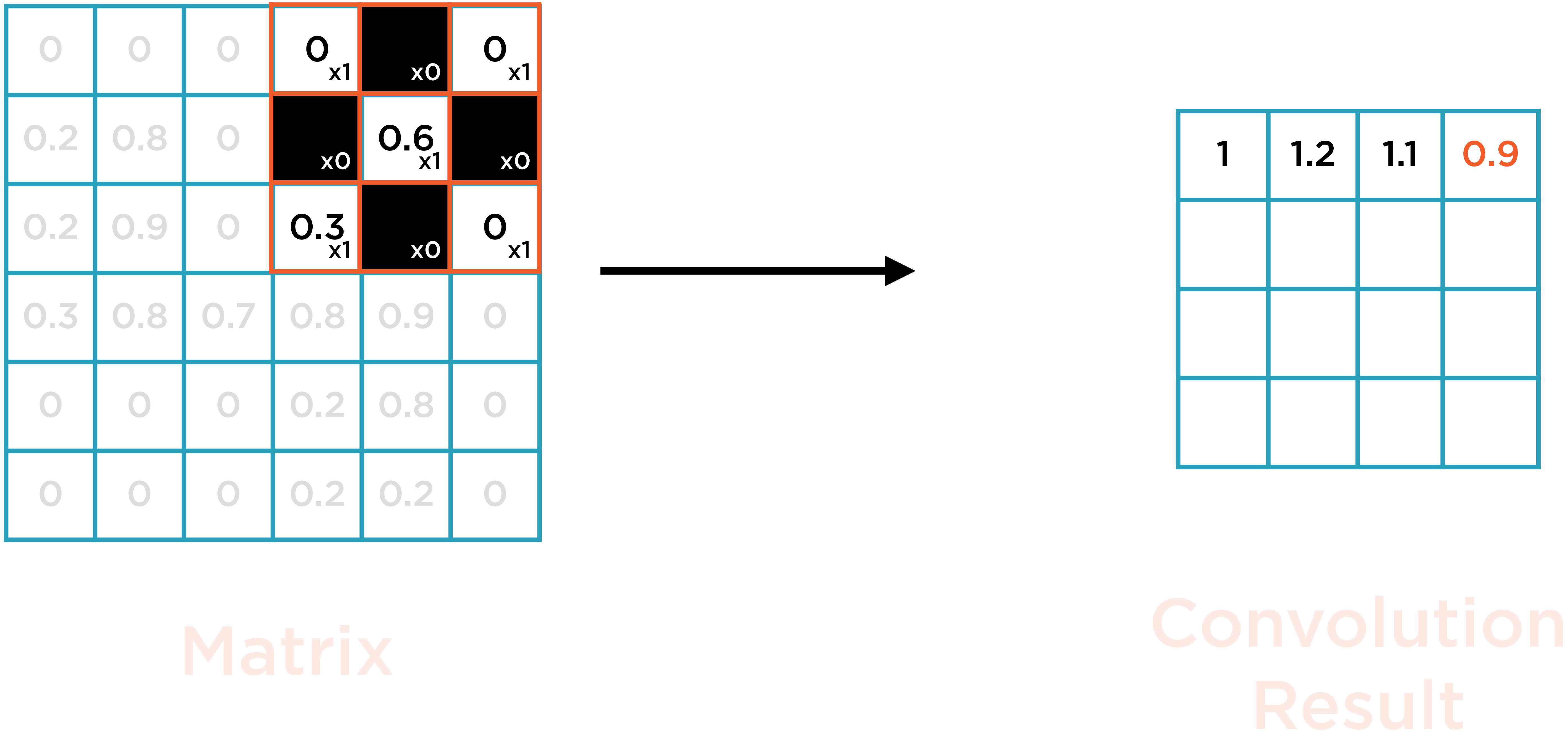
Convolution



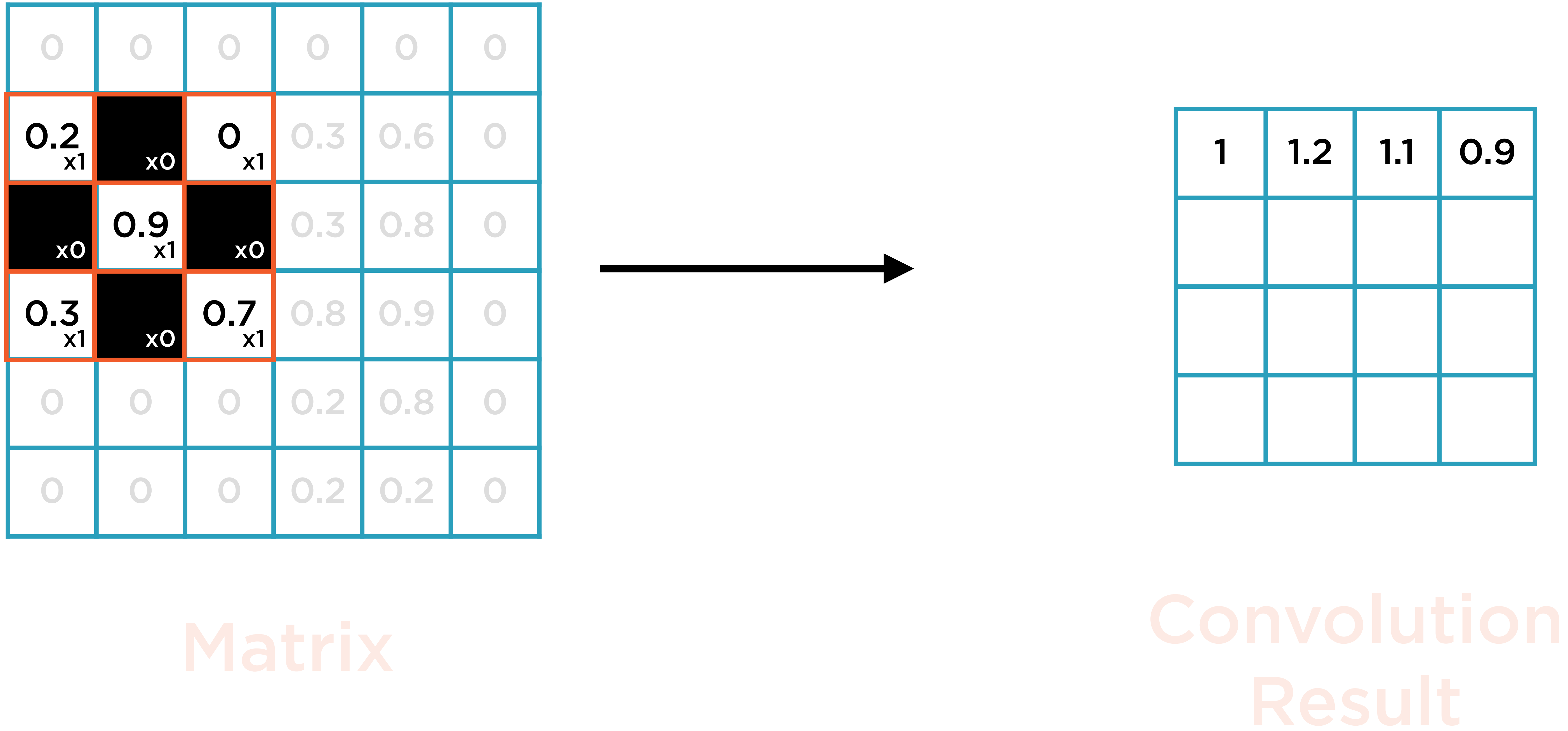
Matrix

Convolution Result

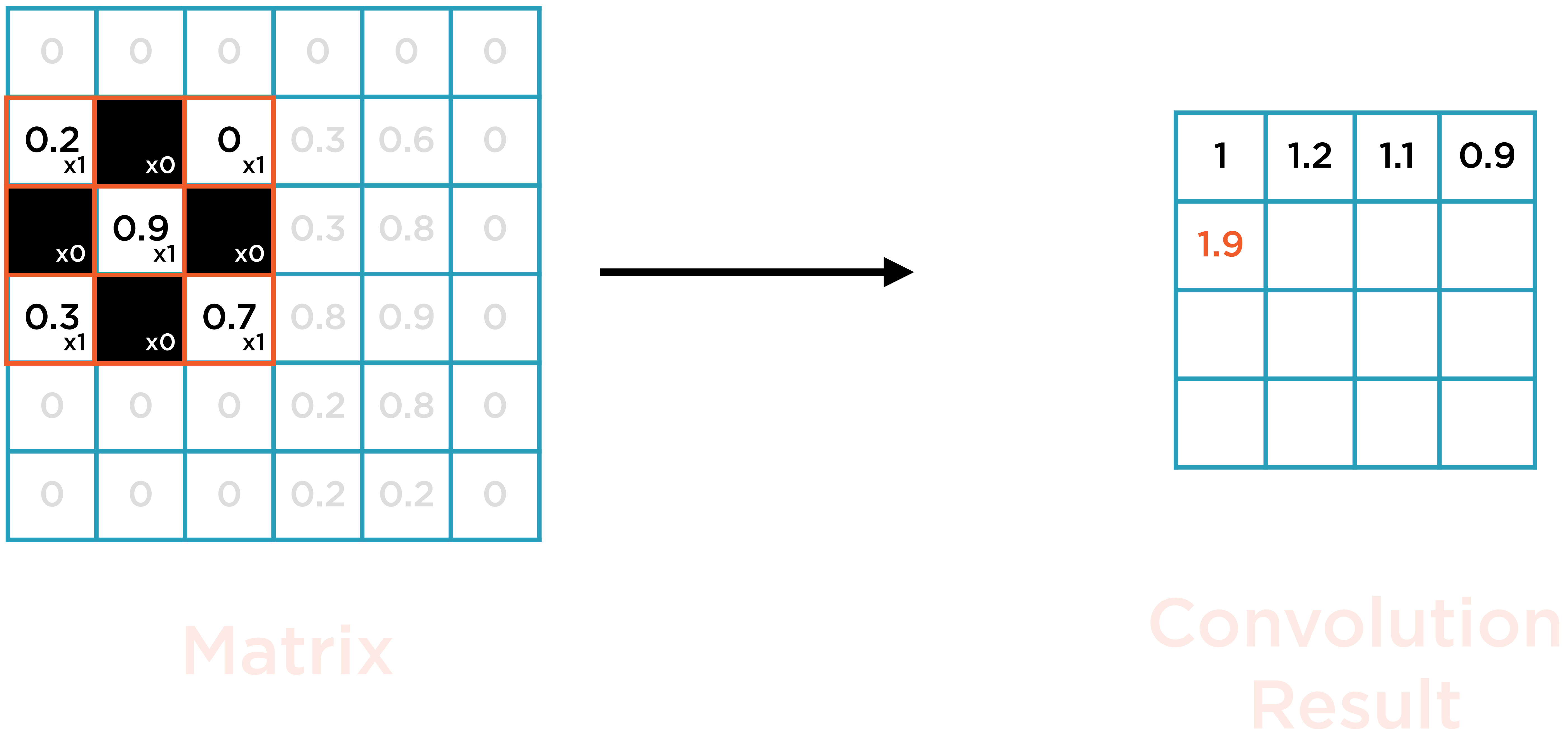
Convolution



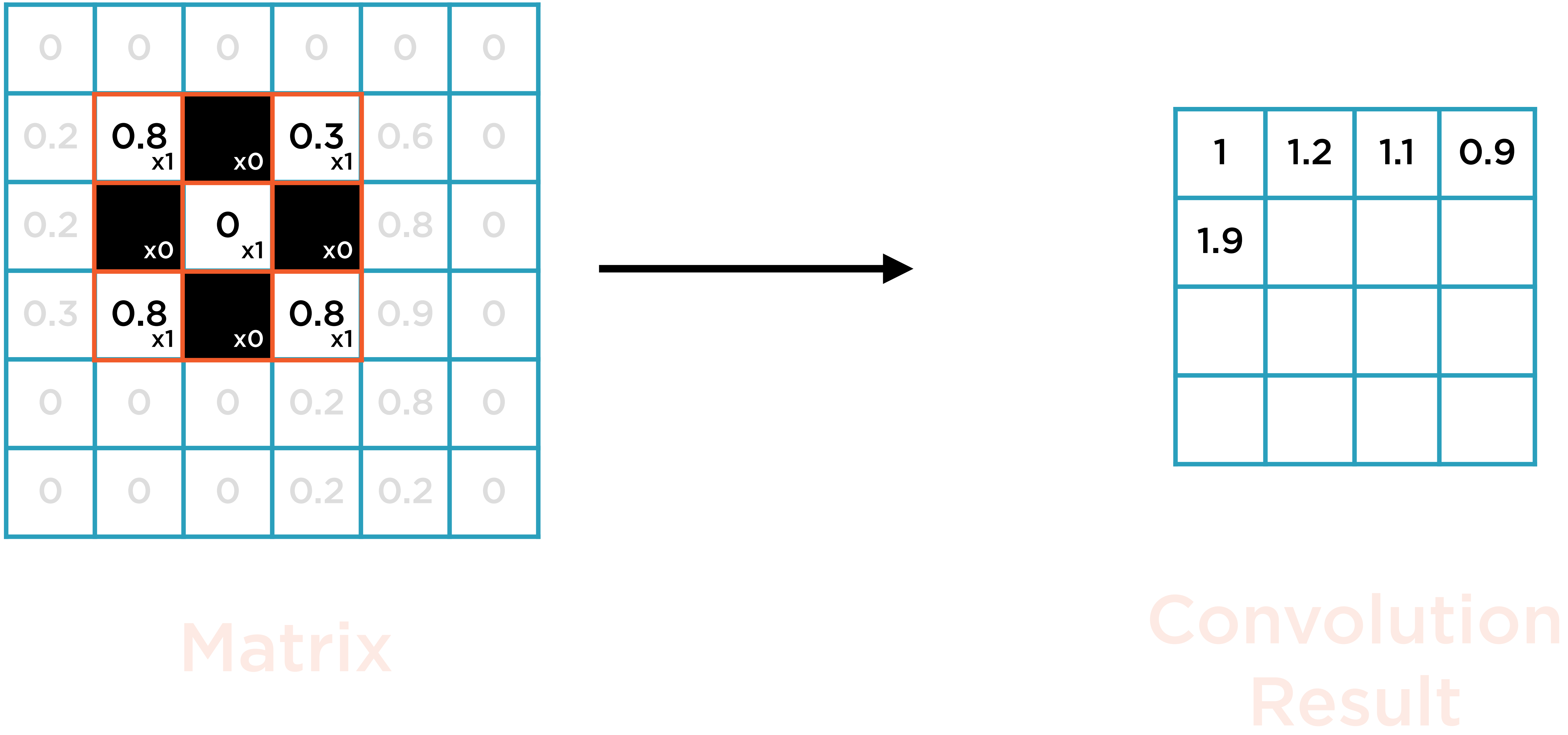
Convolution



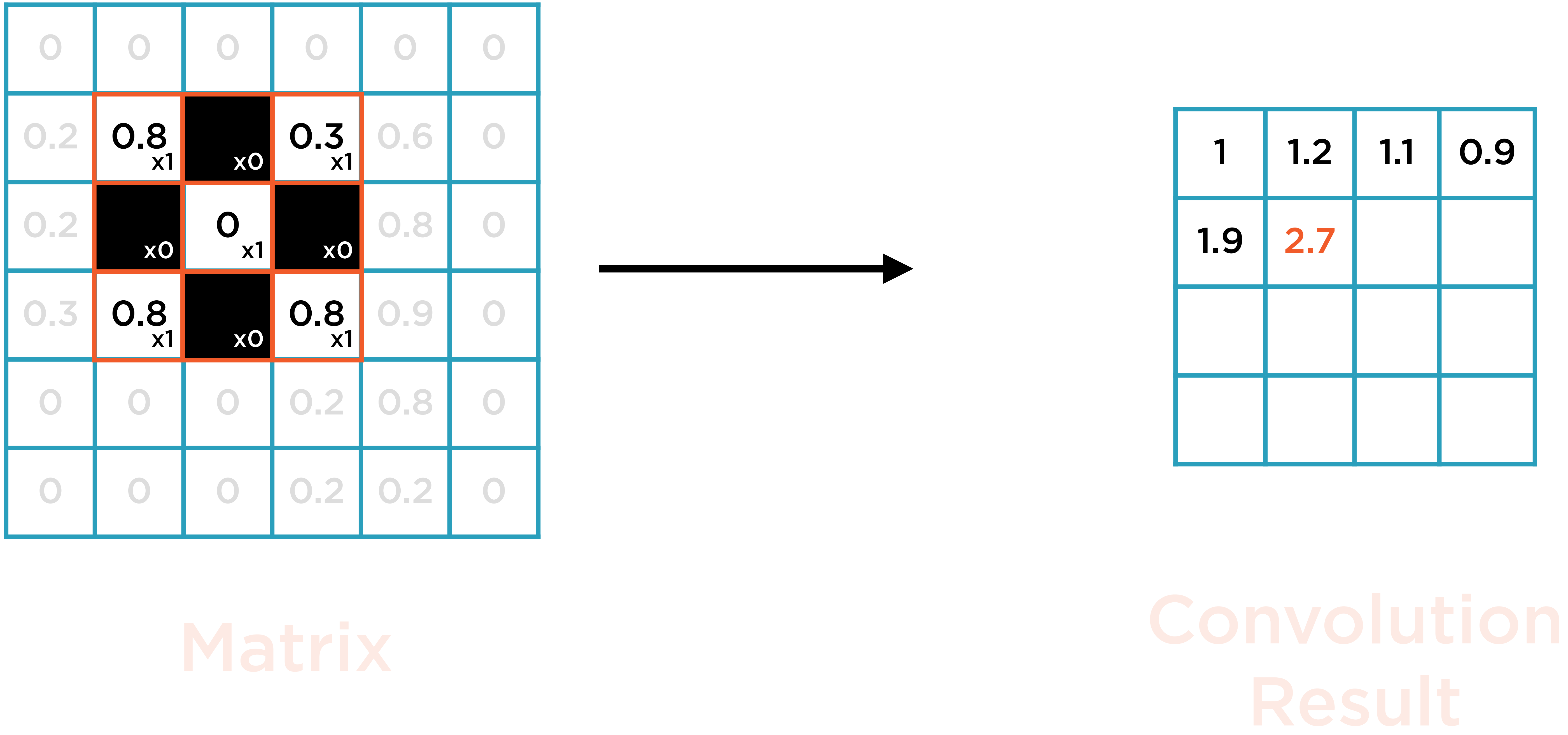
Convolution



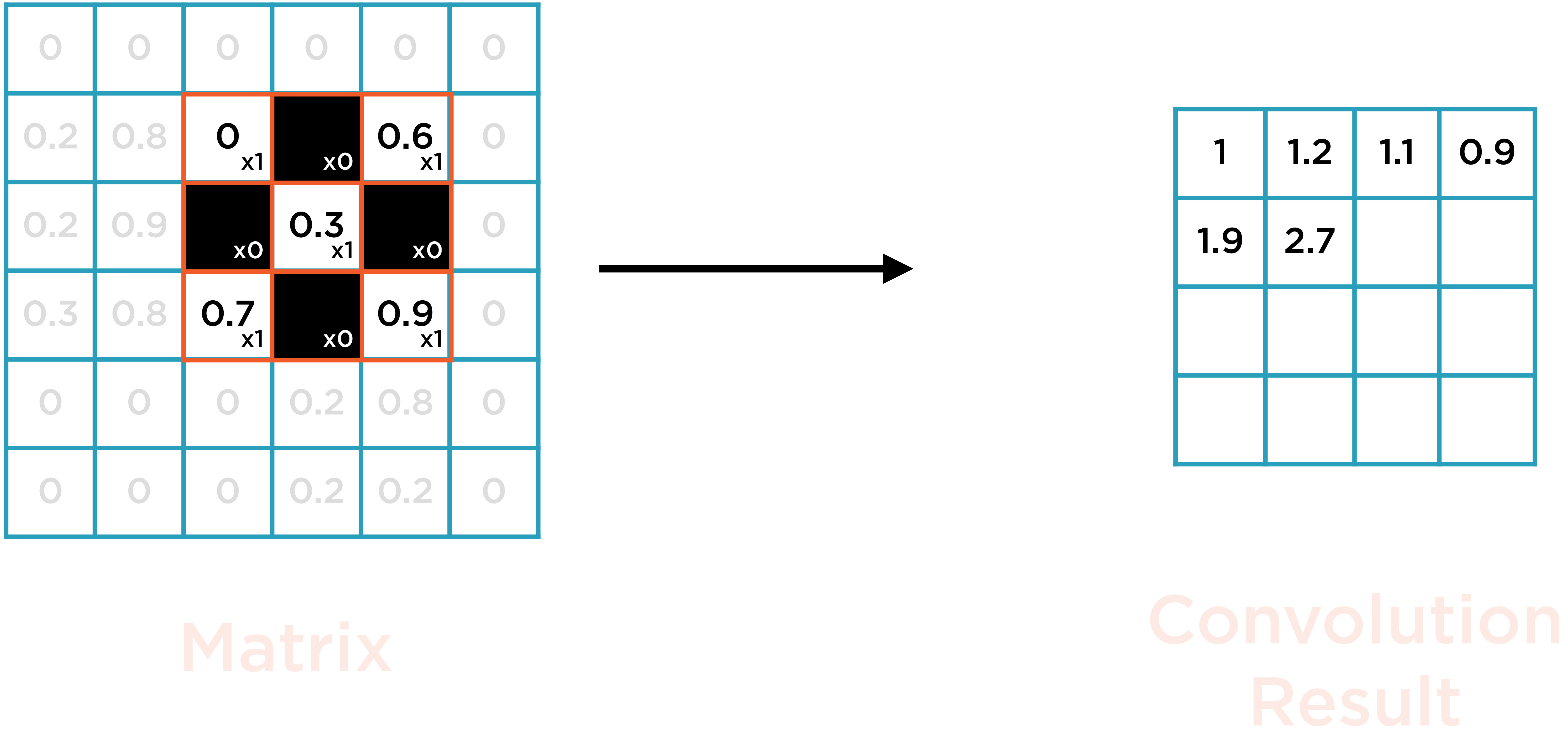
Convolution



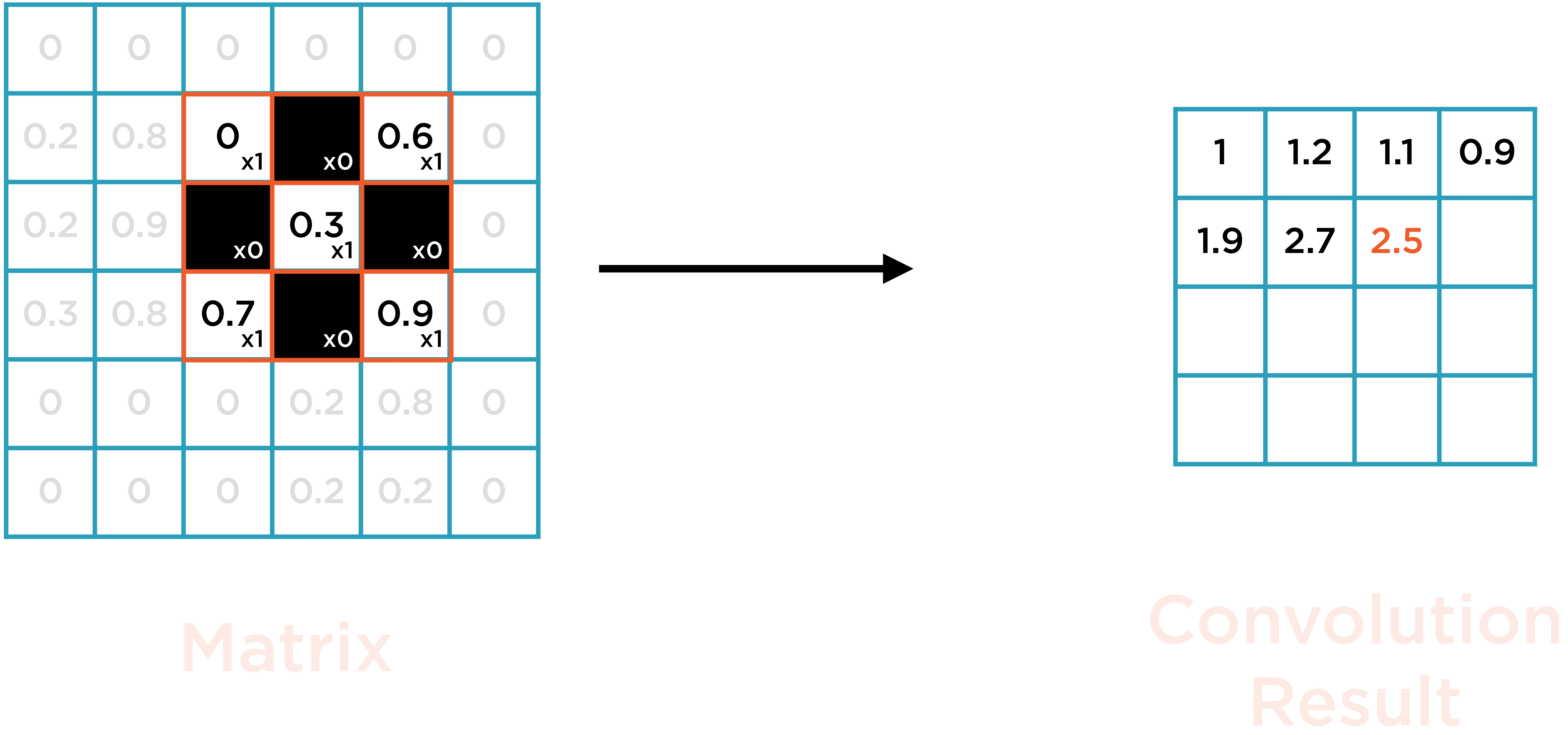
Convolution



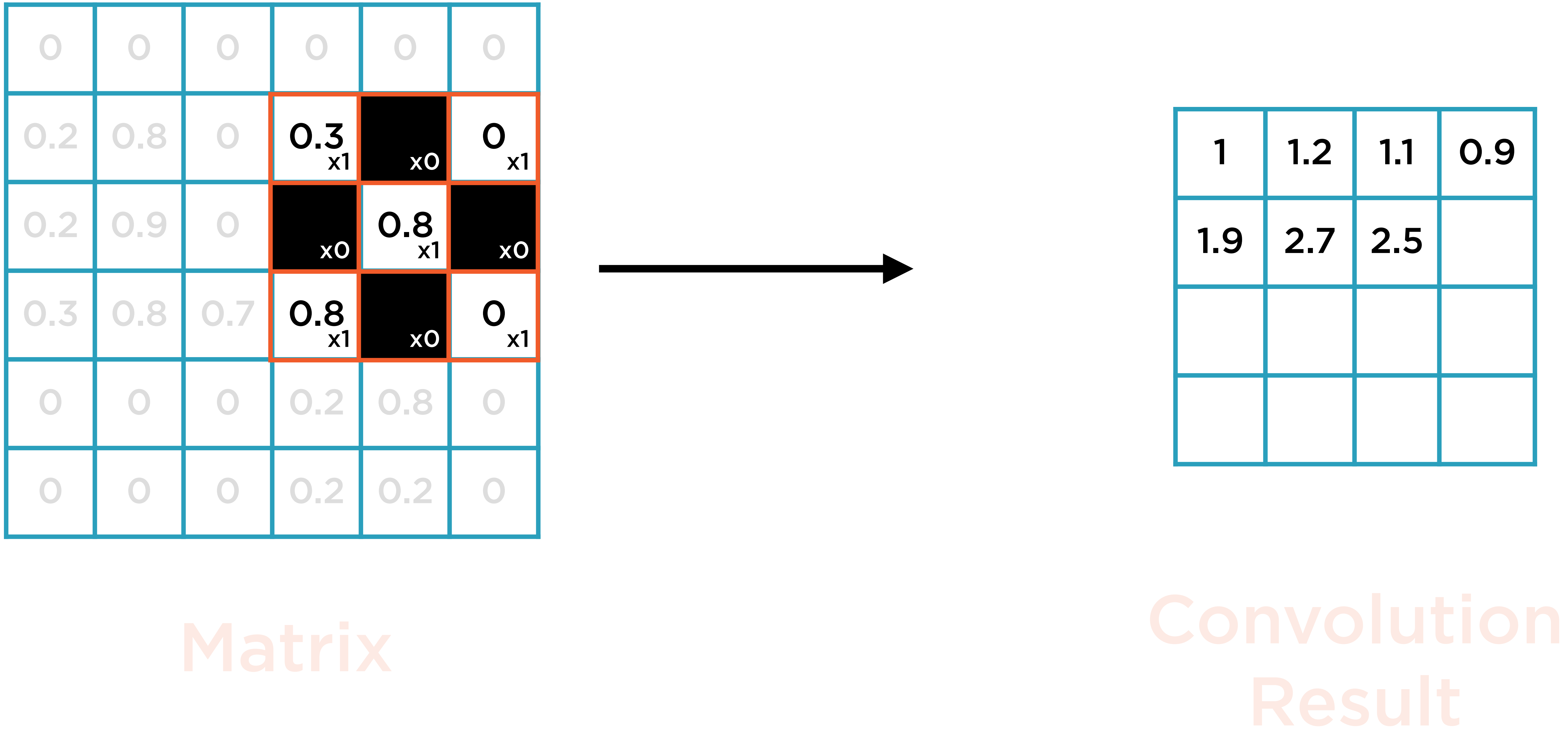
Convolution



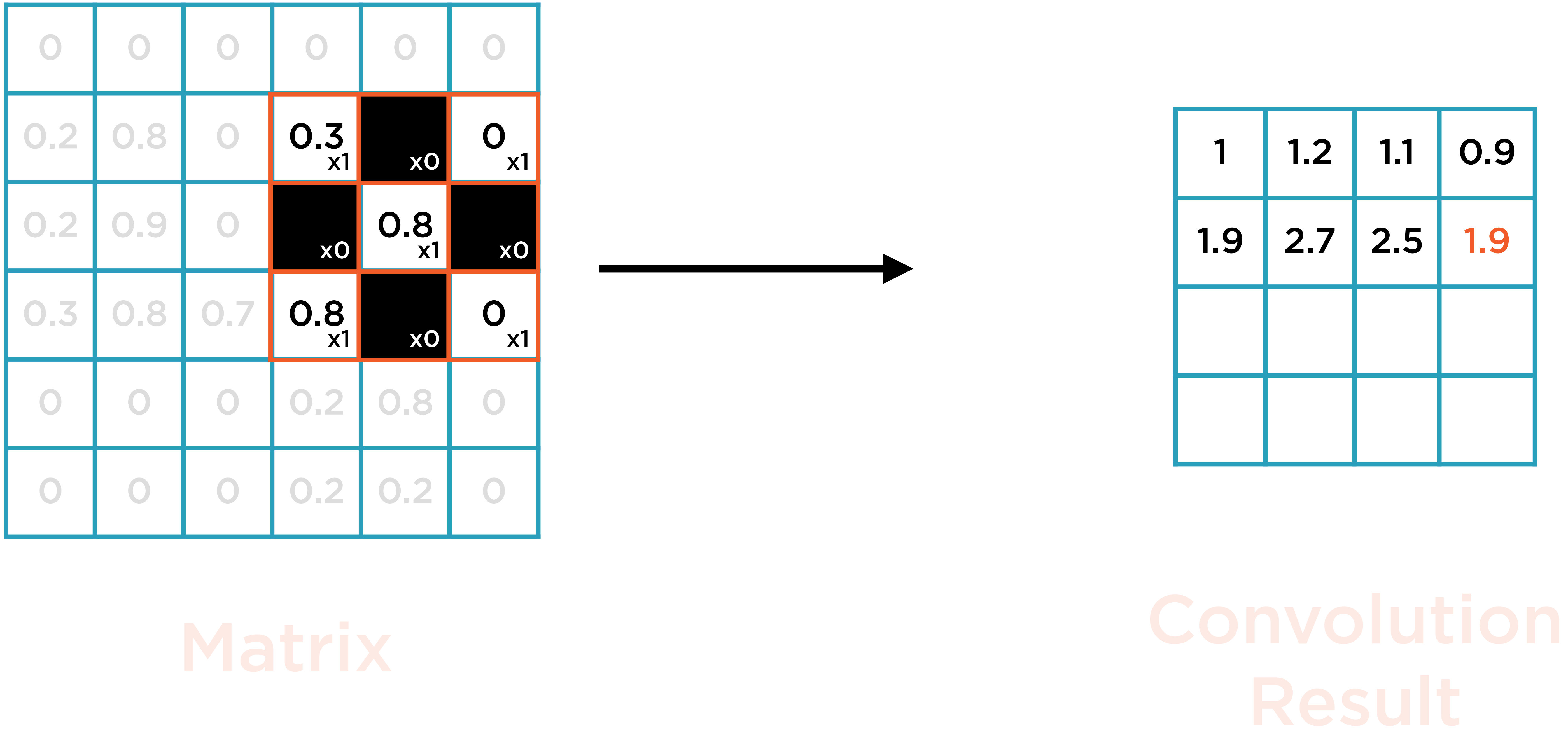
Convolution



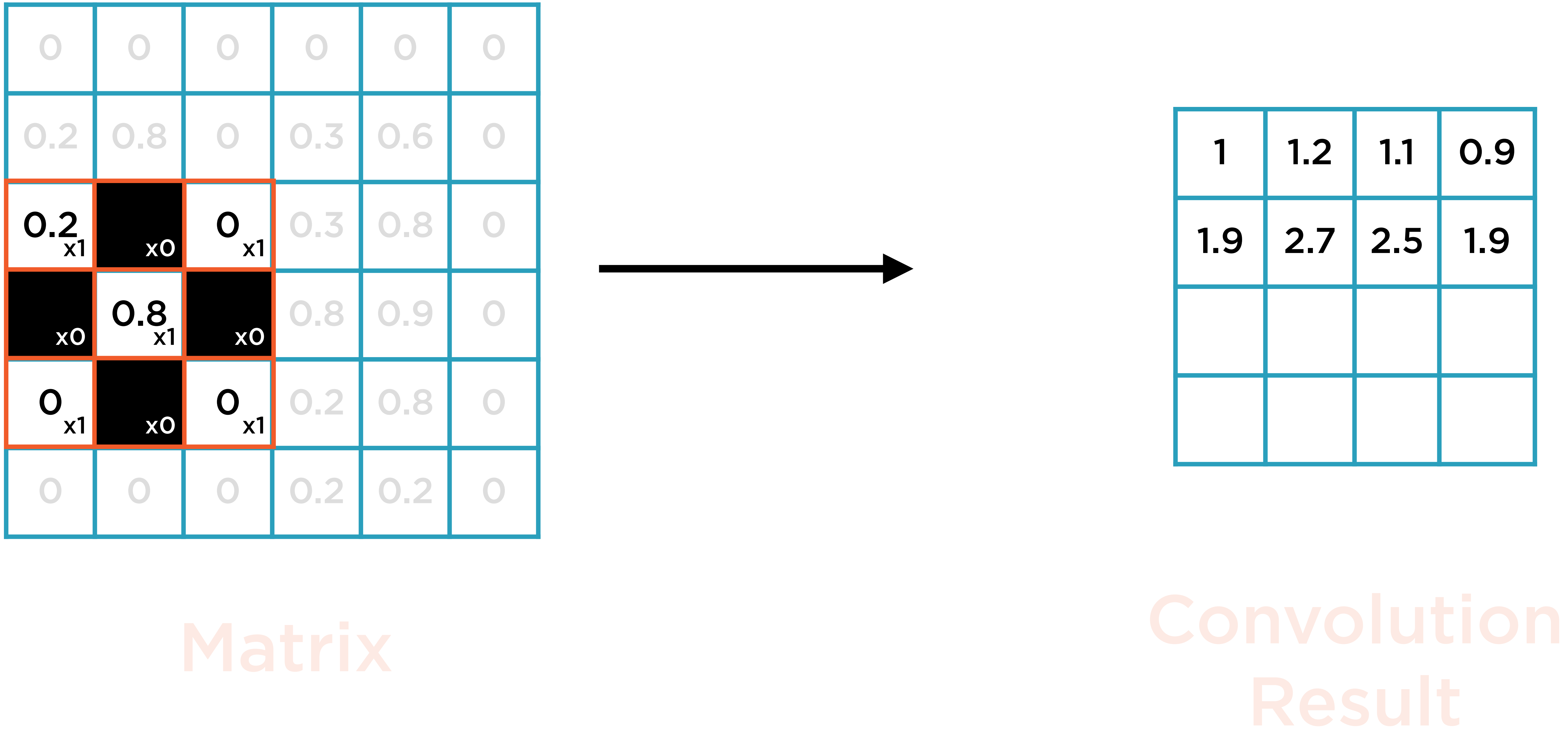
Convolution



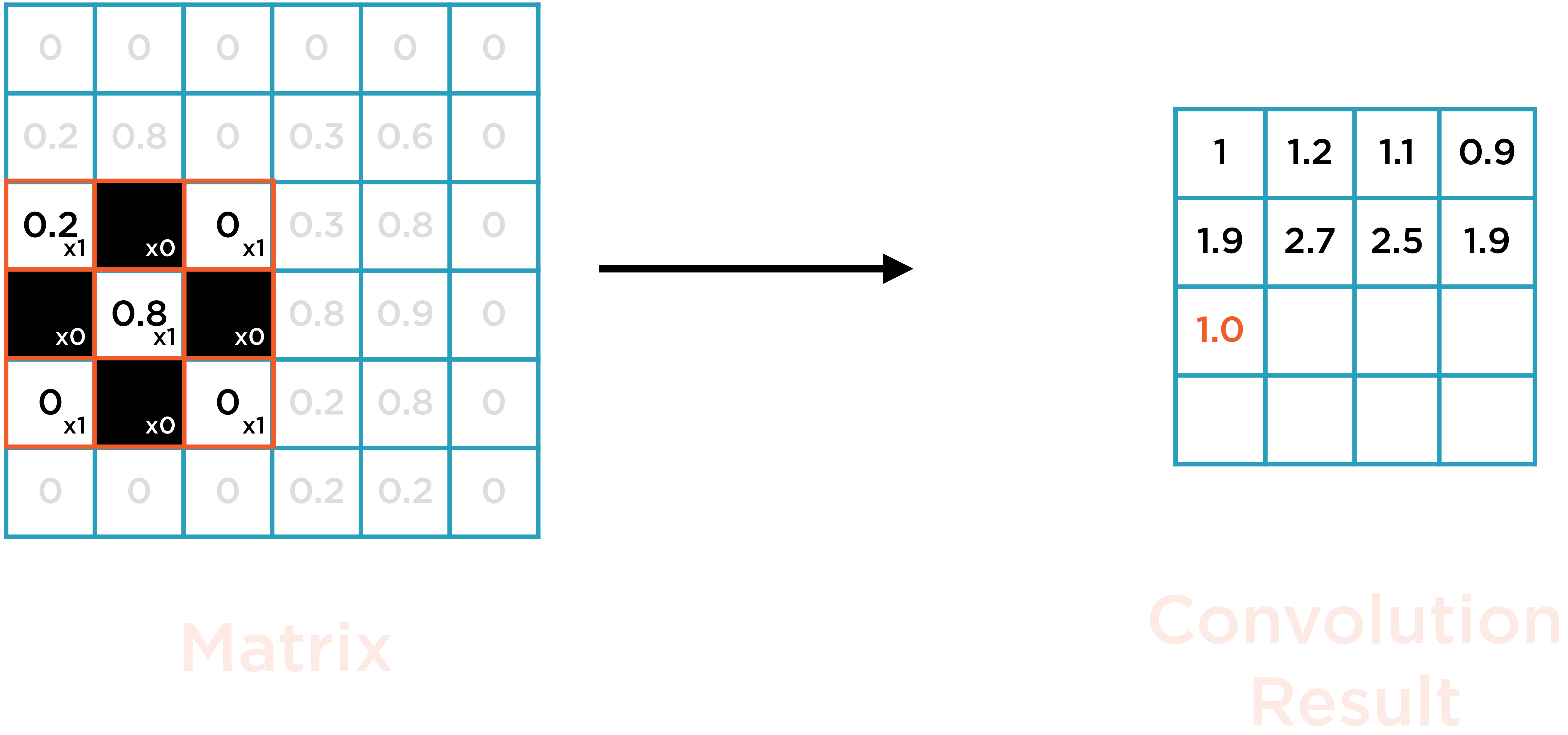
Convolution



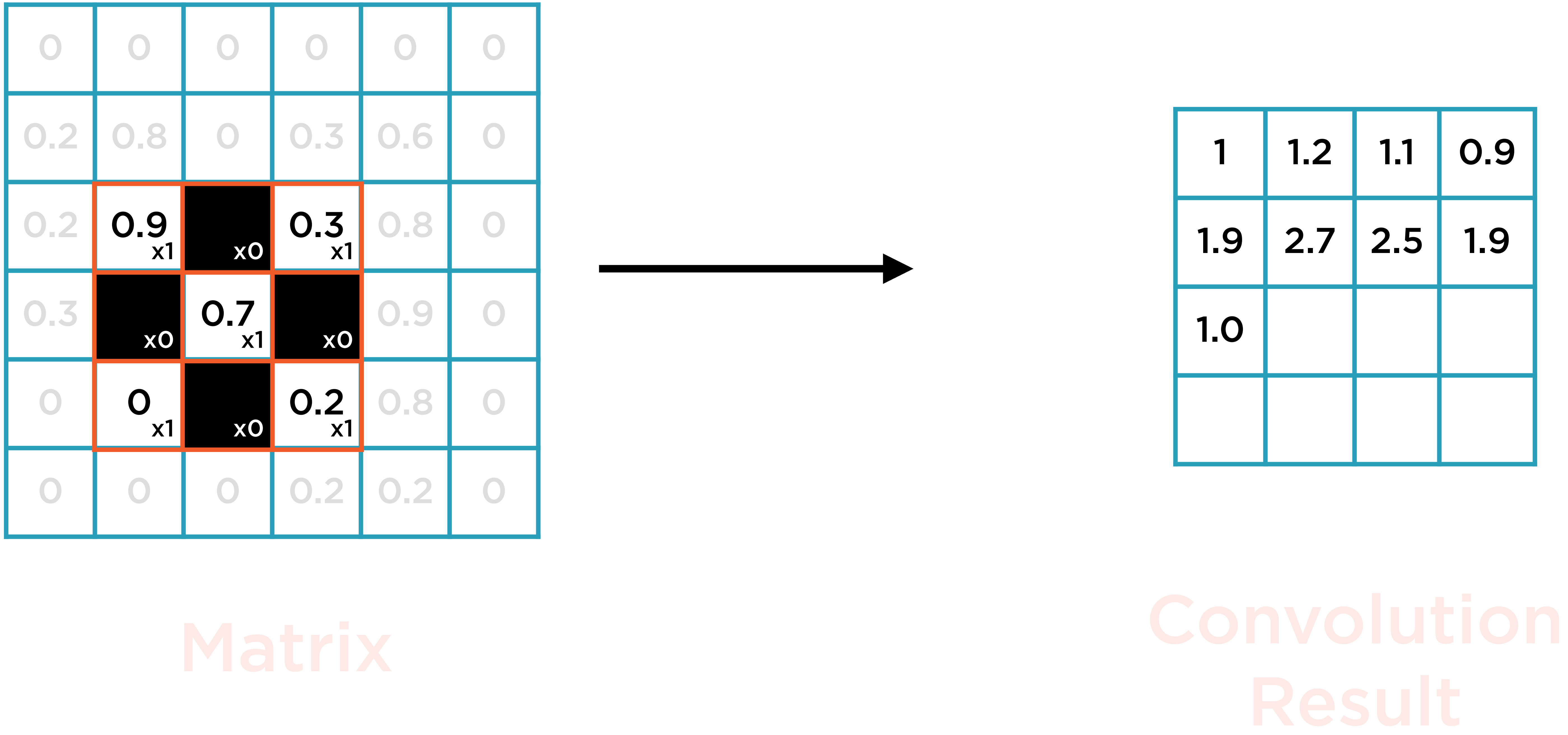
Convolution



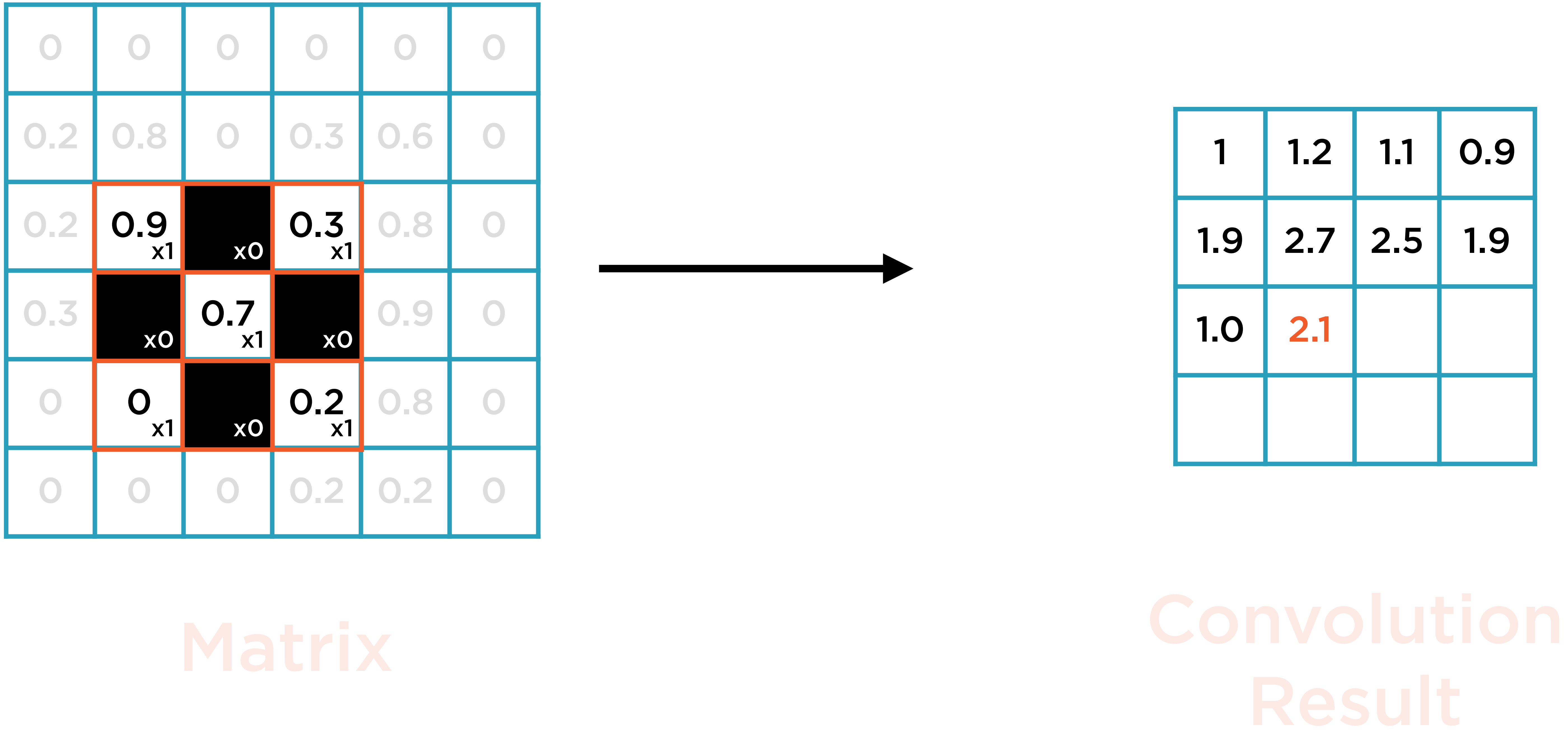
Convolution



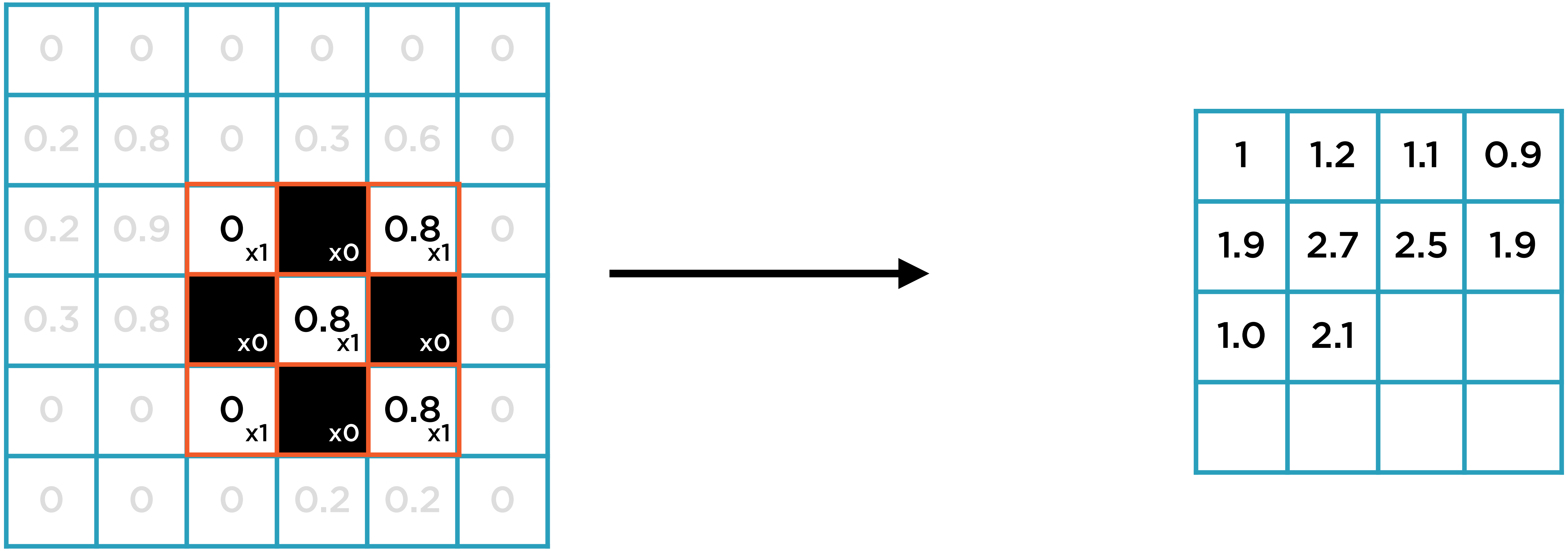
Convolution



Convolution



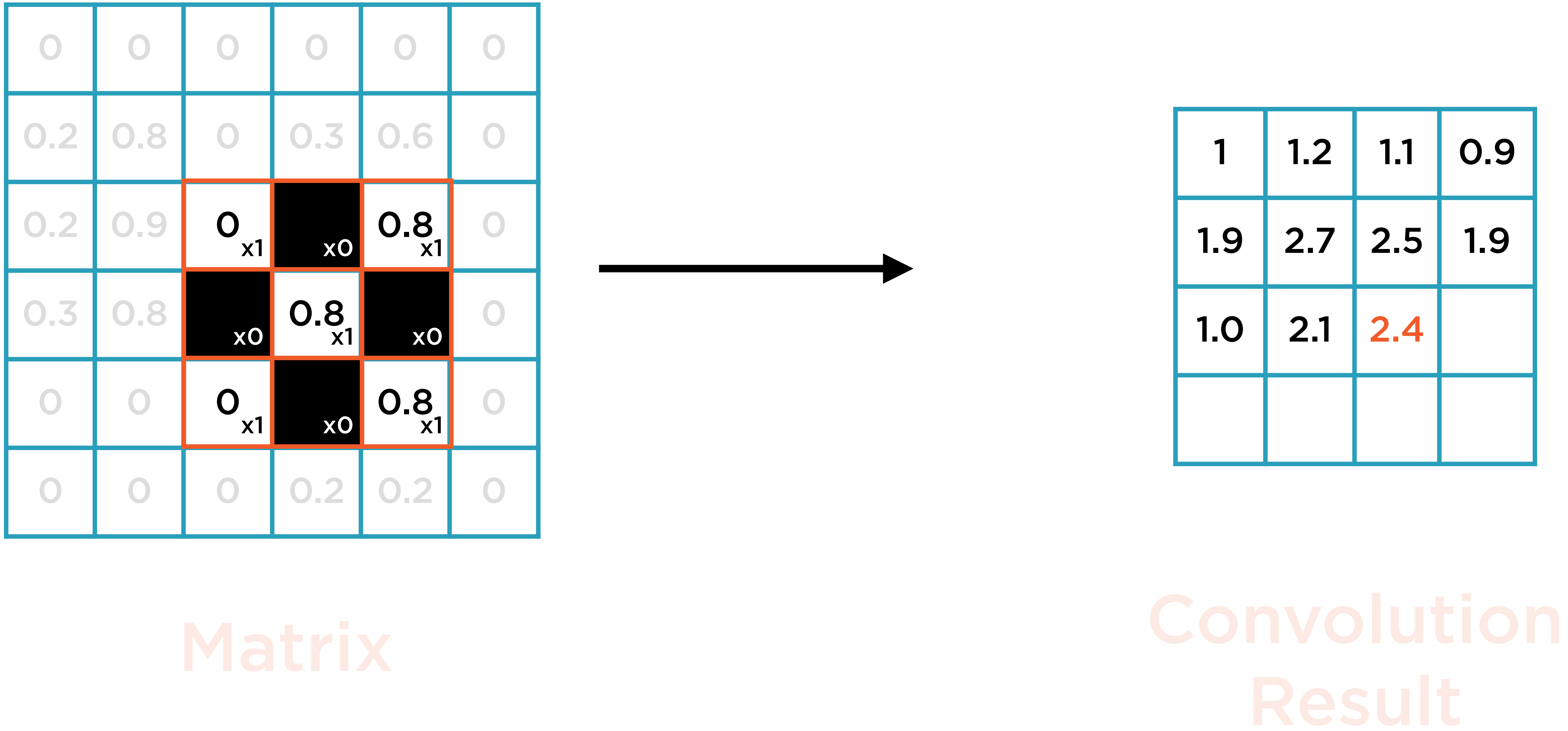
Convolution



Matrix

Convolution Result

Convolution



Convolution

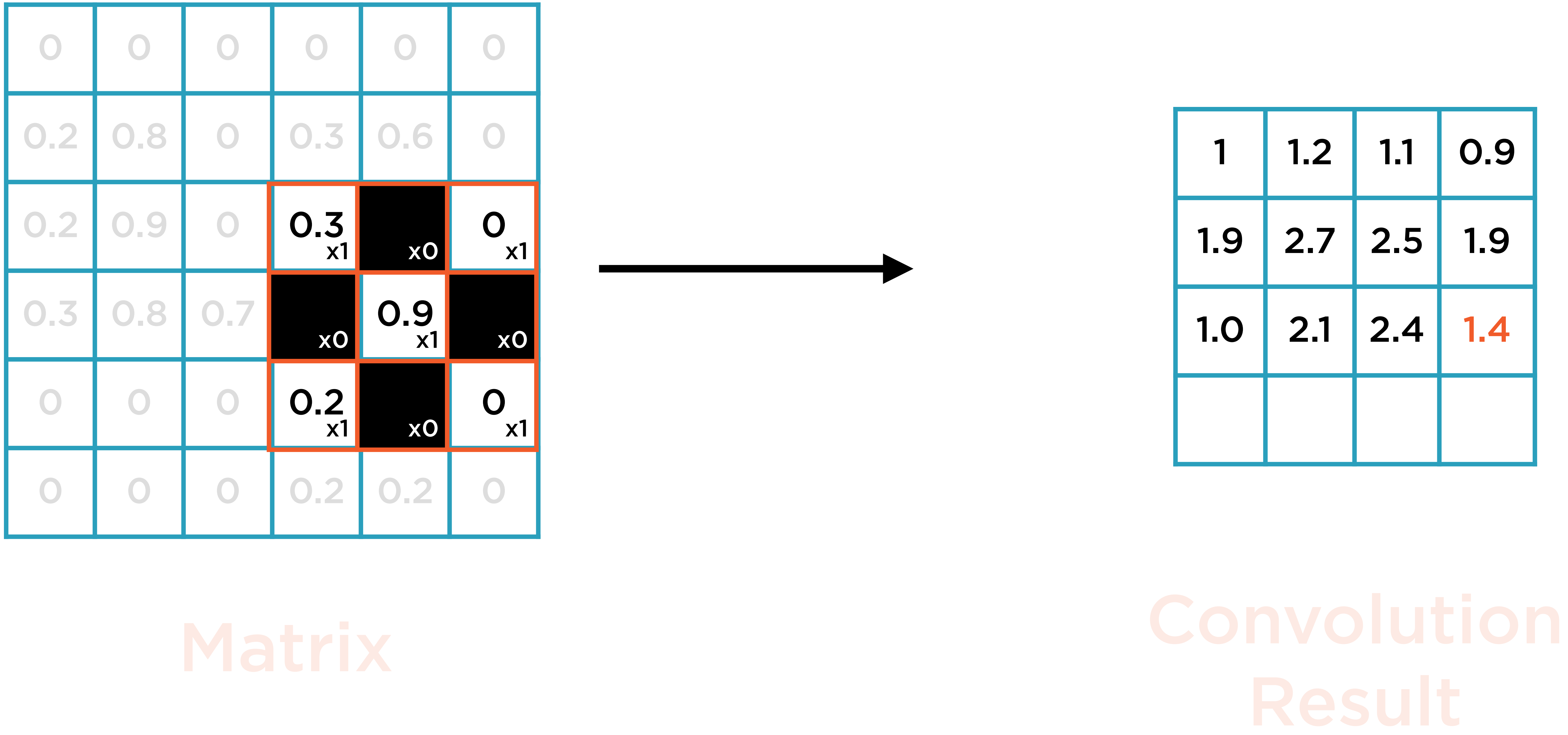
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3 x1	x0	0 x1
0.3	0.8	0.7	x0	0.9 x1	x0
0	0	0	0.2 x1	x0	0 x1
0	0	0	0.2	0.2	0

Matrix

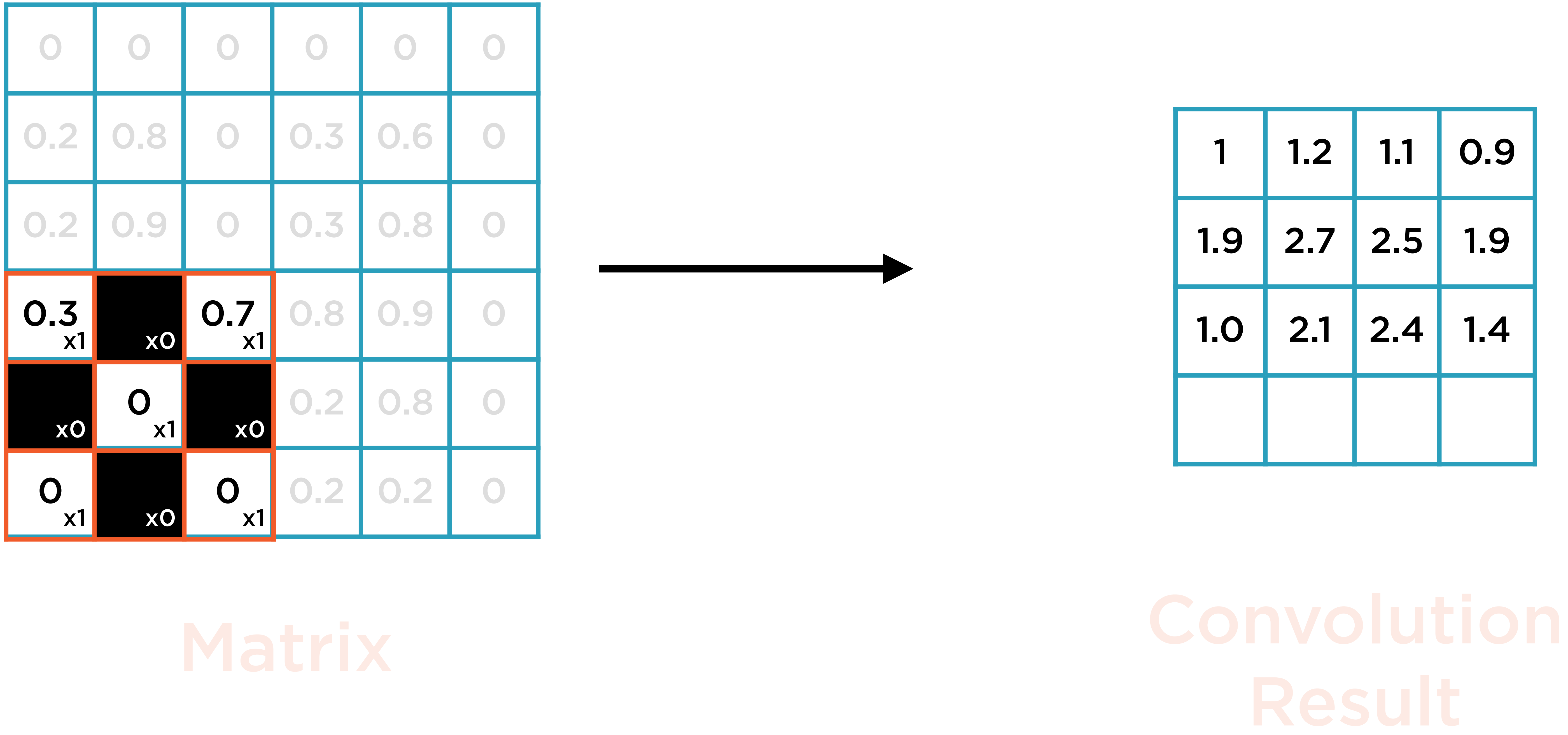
1	1.2	1.1	0.9
1.9	2.7	2.5	1.9
1.0	2.1	2.4	

Convolution Result

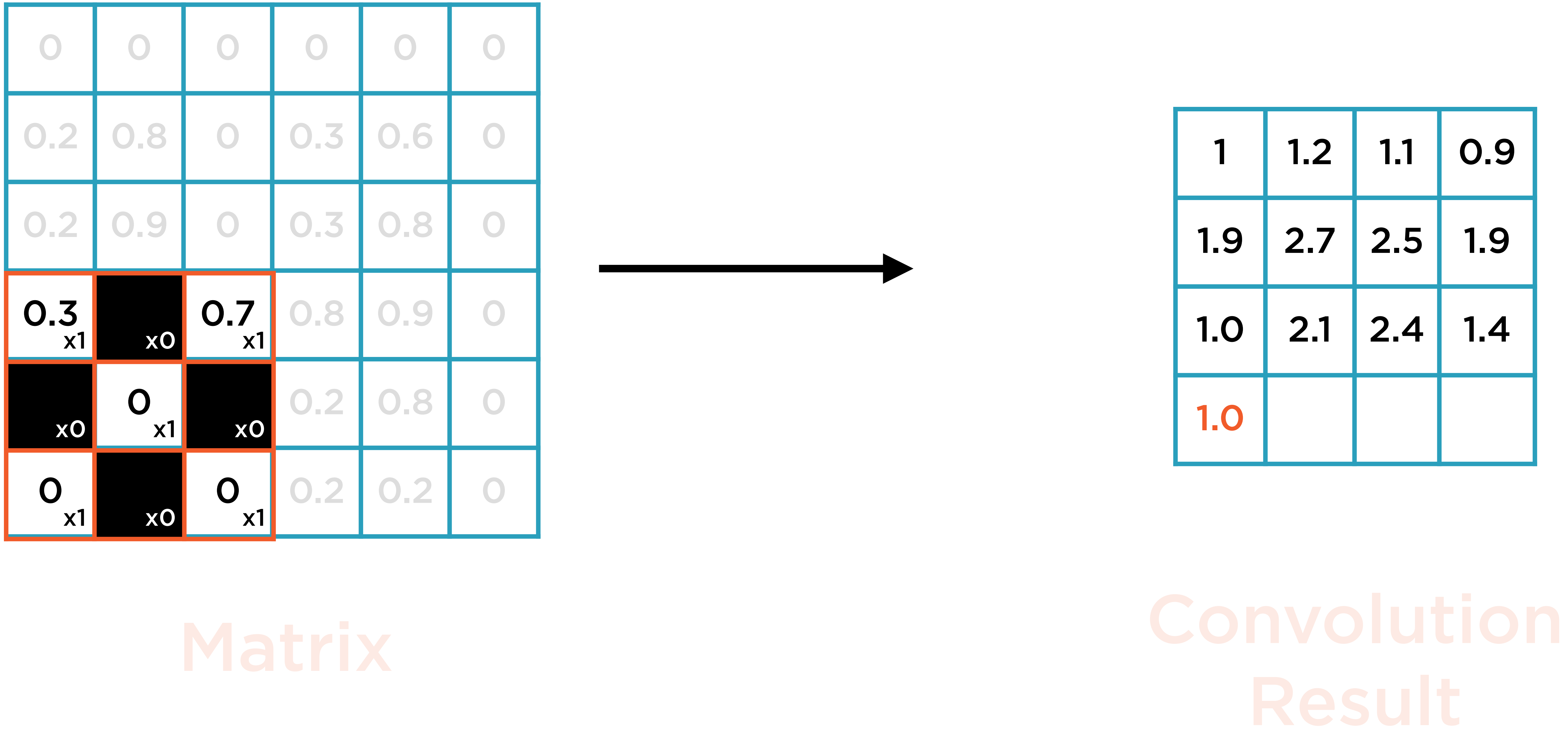
Convolution



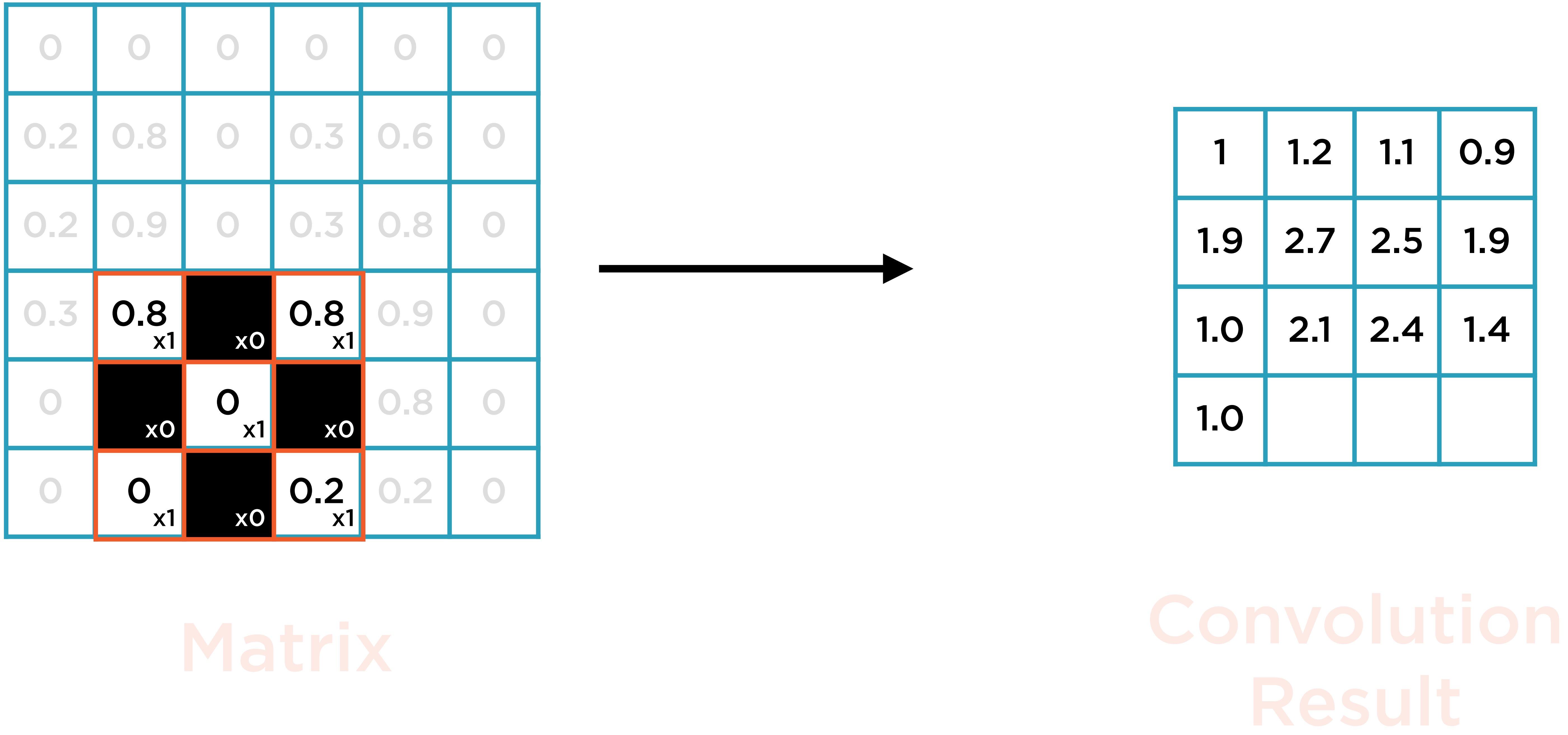
Convolution



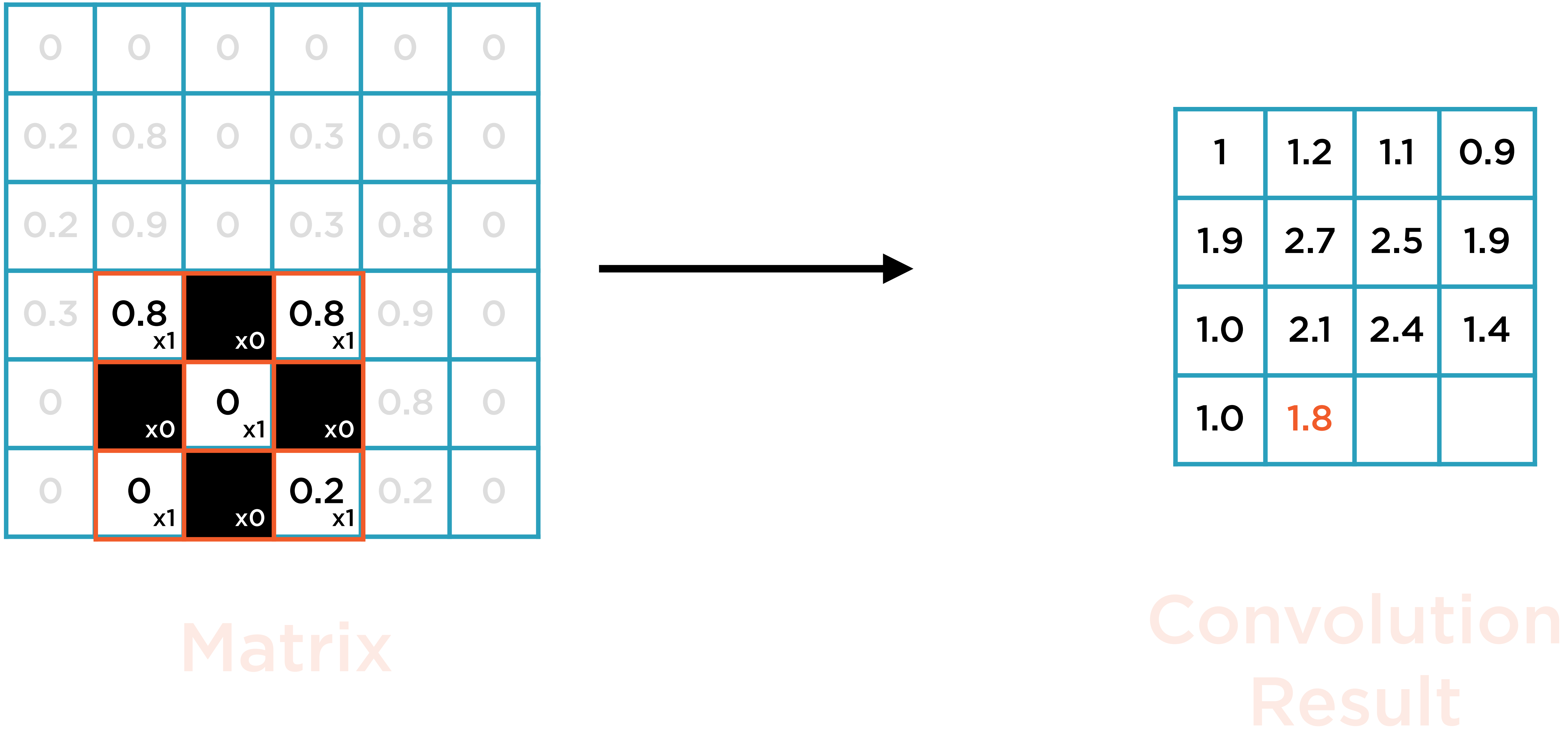
Convolution



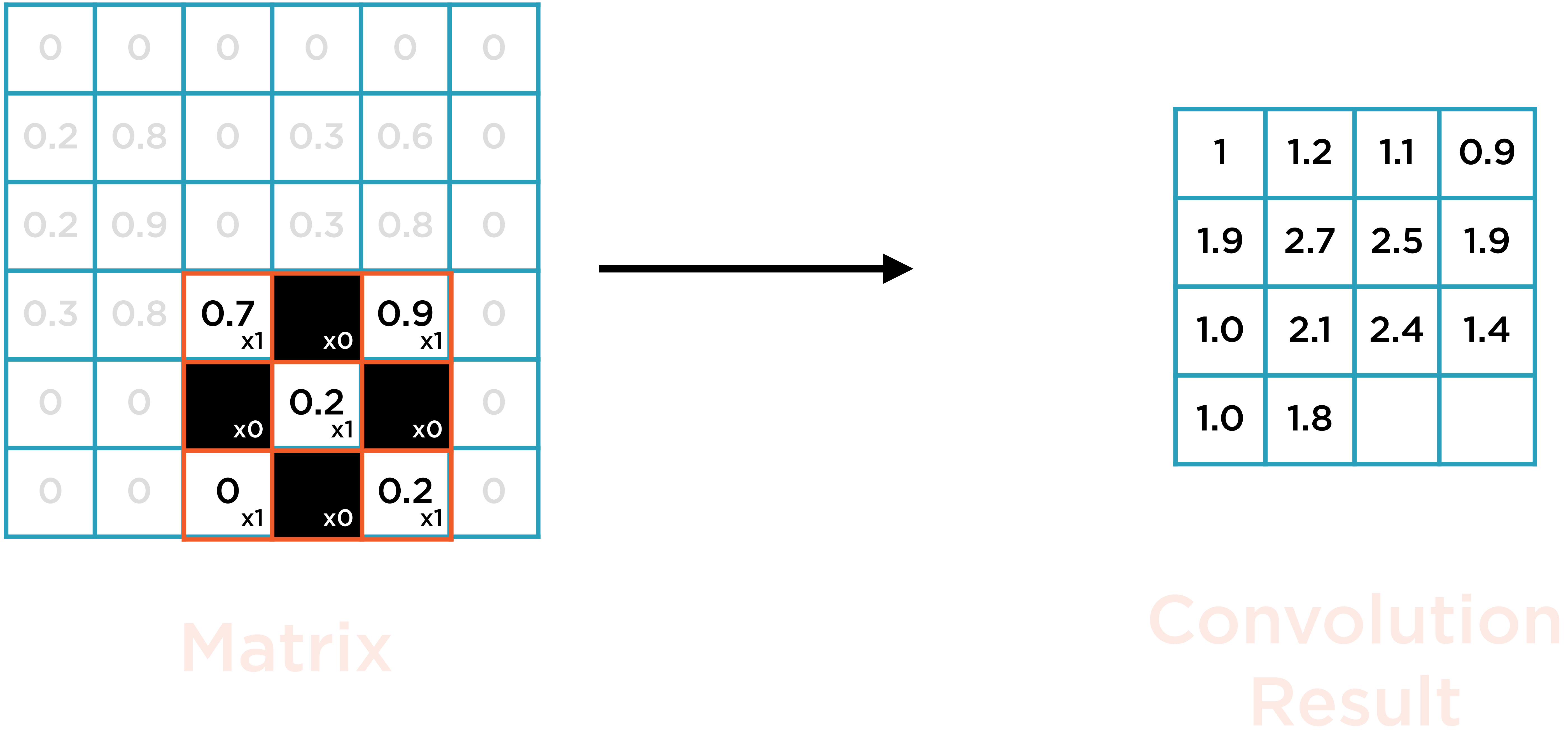
Convolution



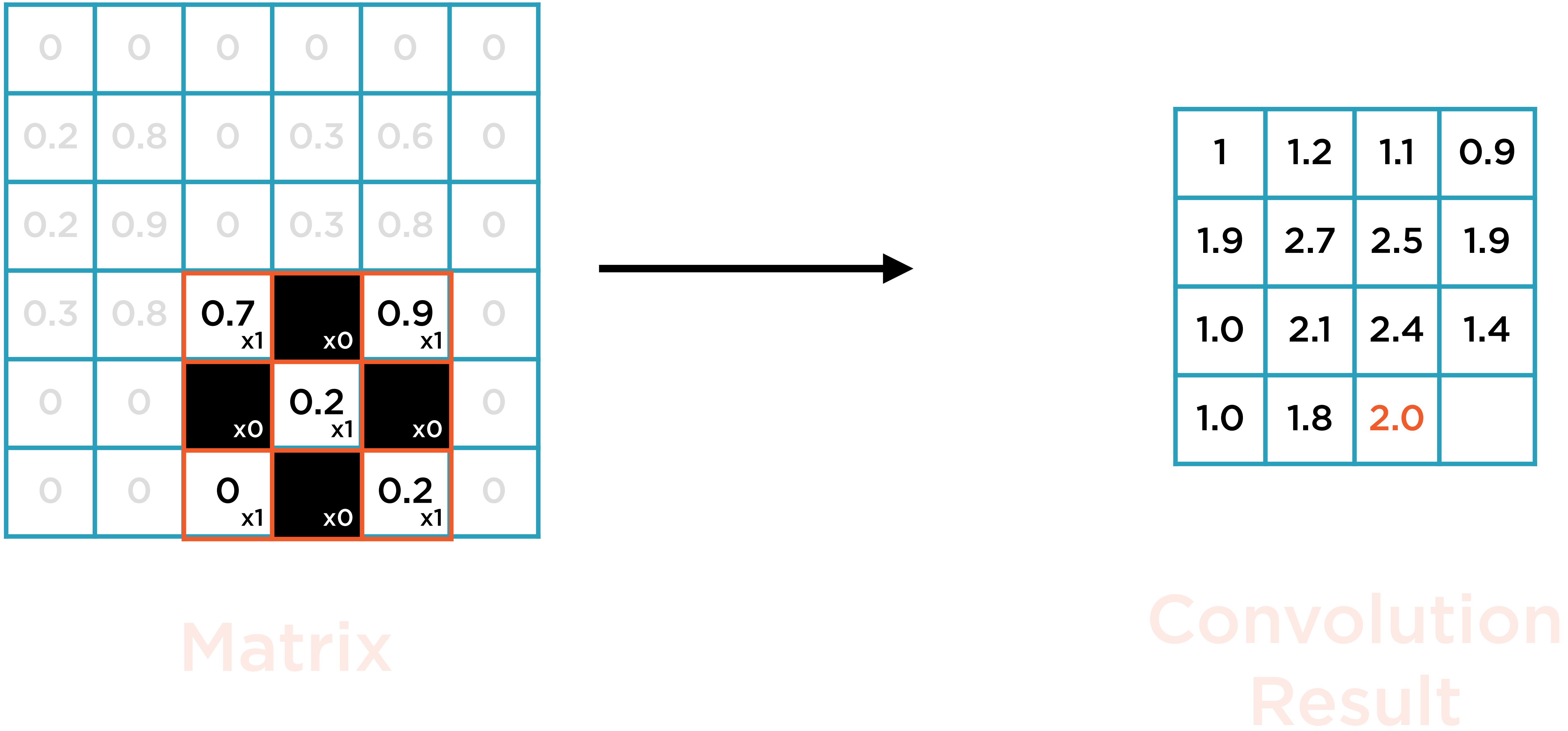
Convolution



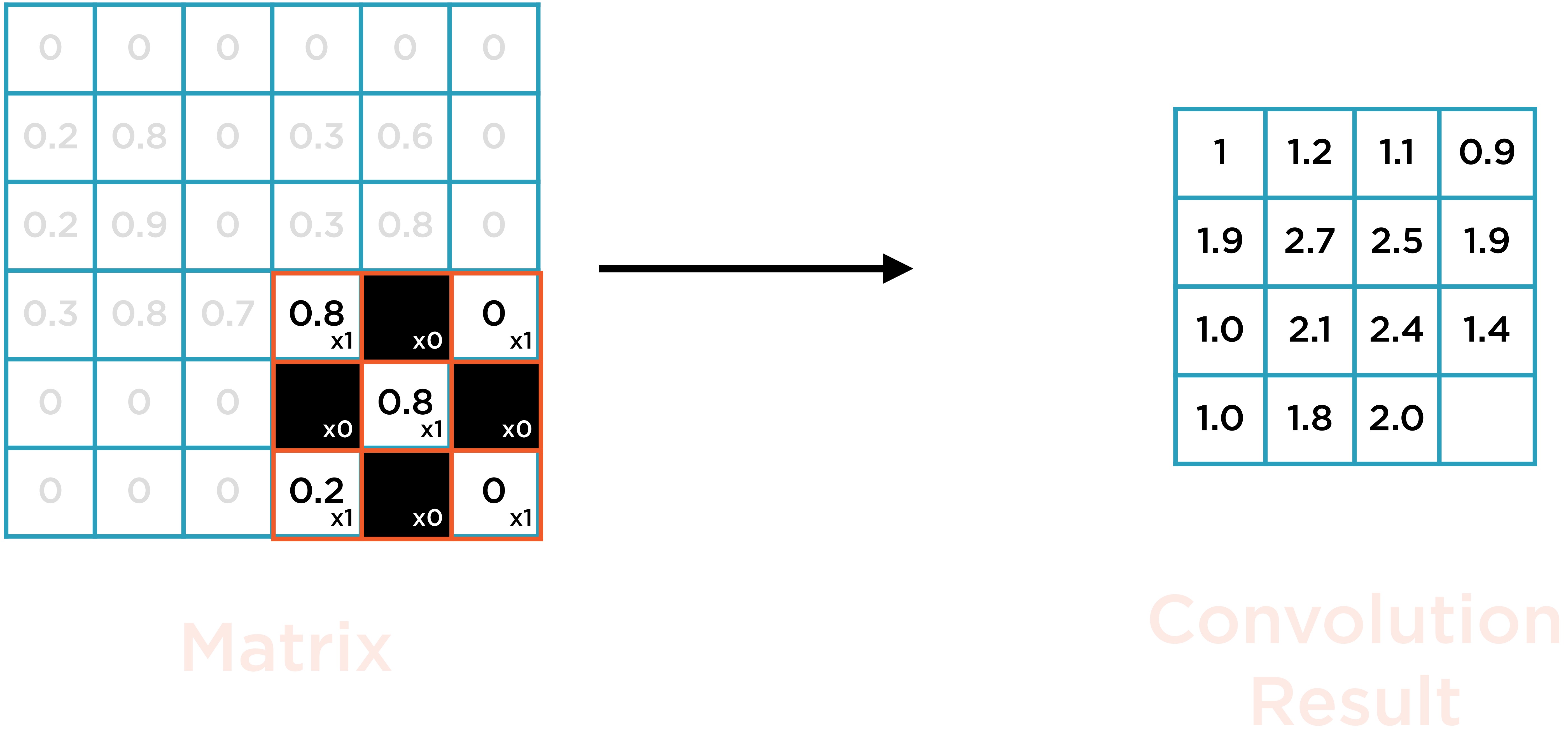
Convolution



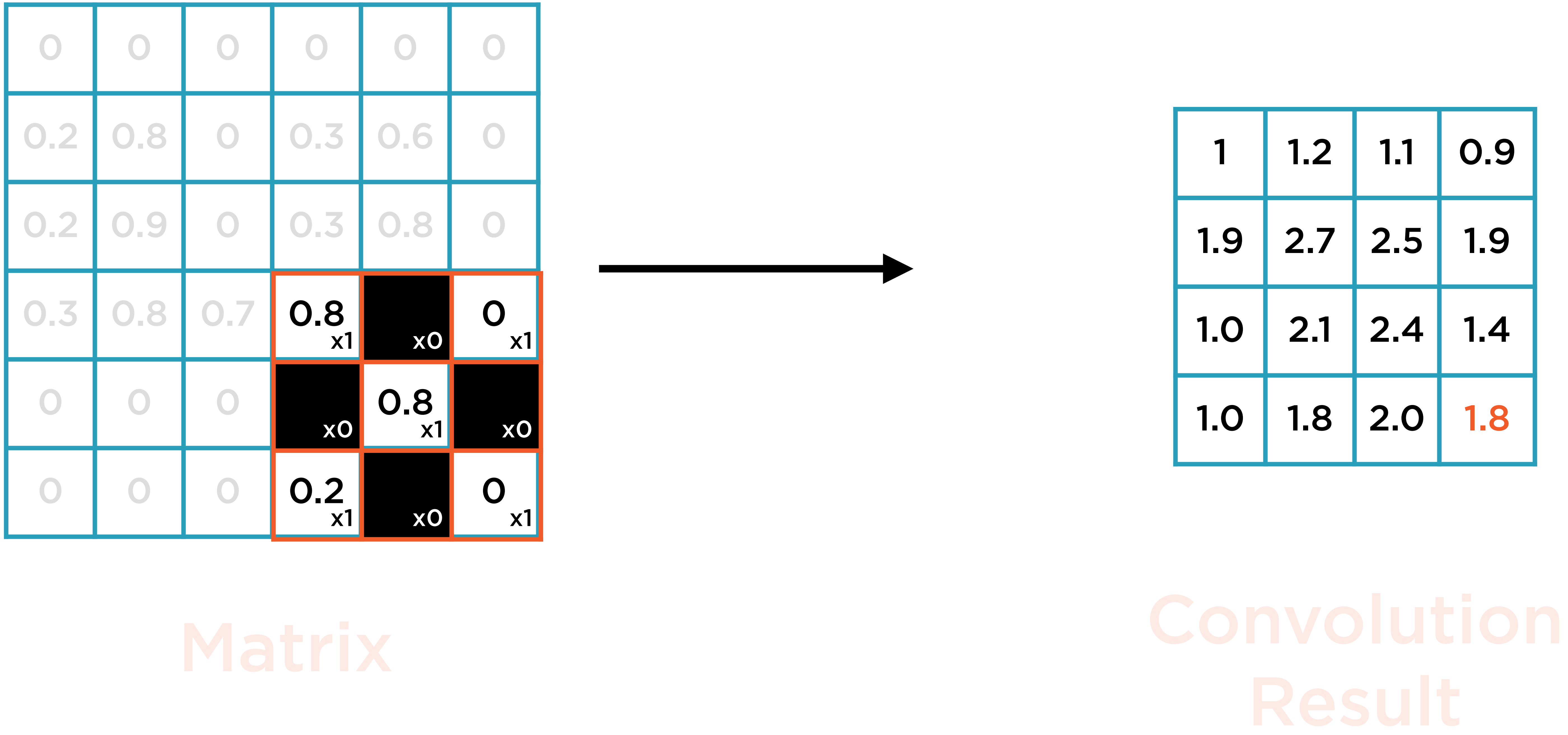
Convolution



Convolution

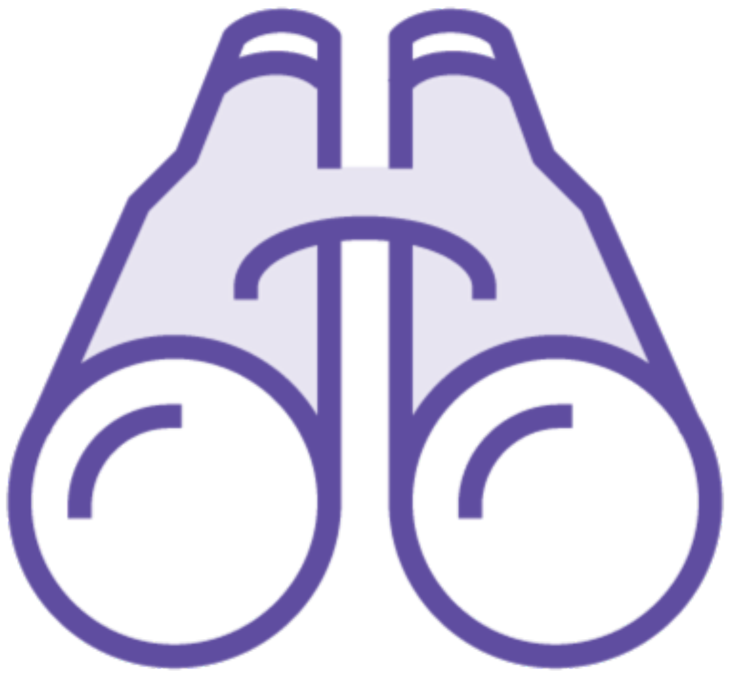


Convolution



Convolutional Layers

Convolutional Layers



Convolution layers - zoom in on specific bits of input

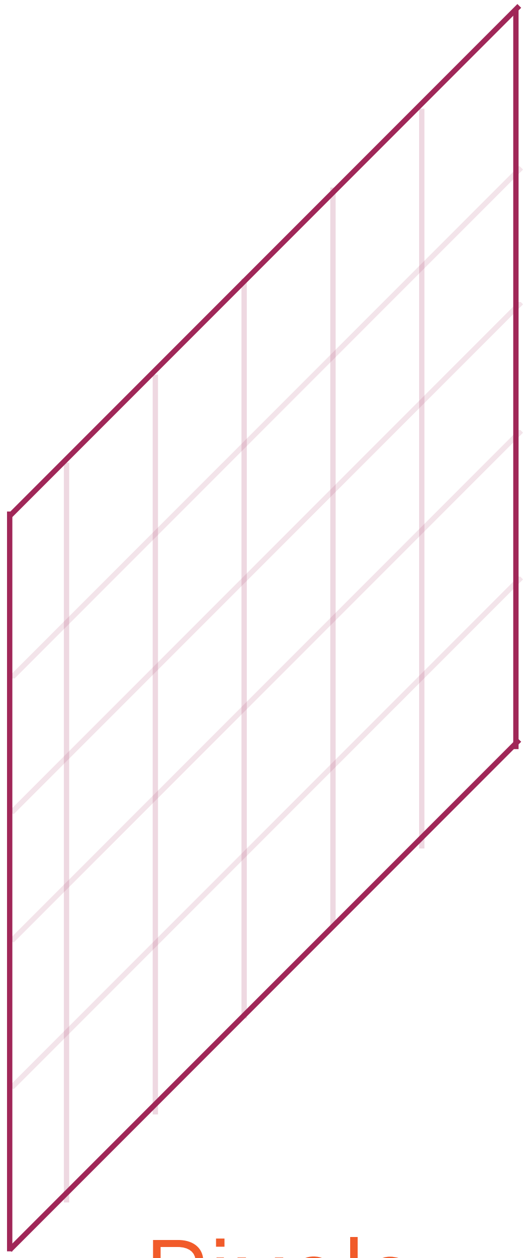
Extract structure and features in the input image

Successive layers aggregate inputs into higher level features

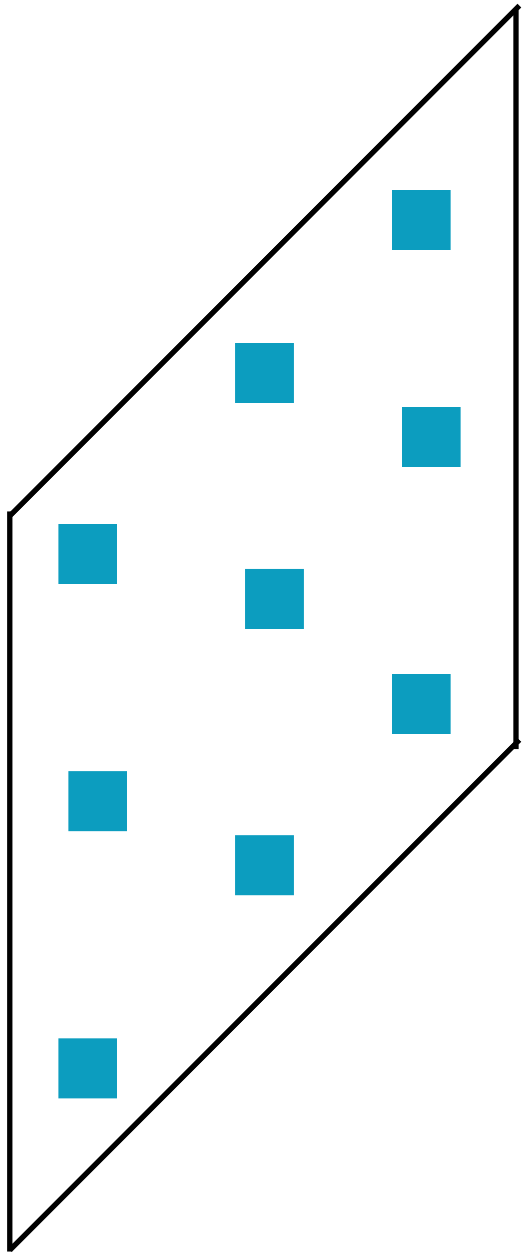
Pixels >> Lines >> Edges >> Object

Feature Maps

Image



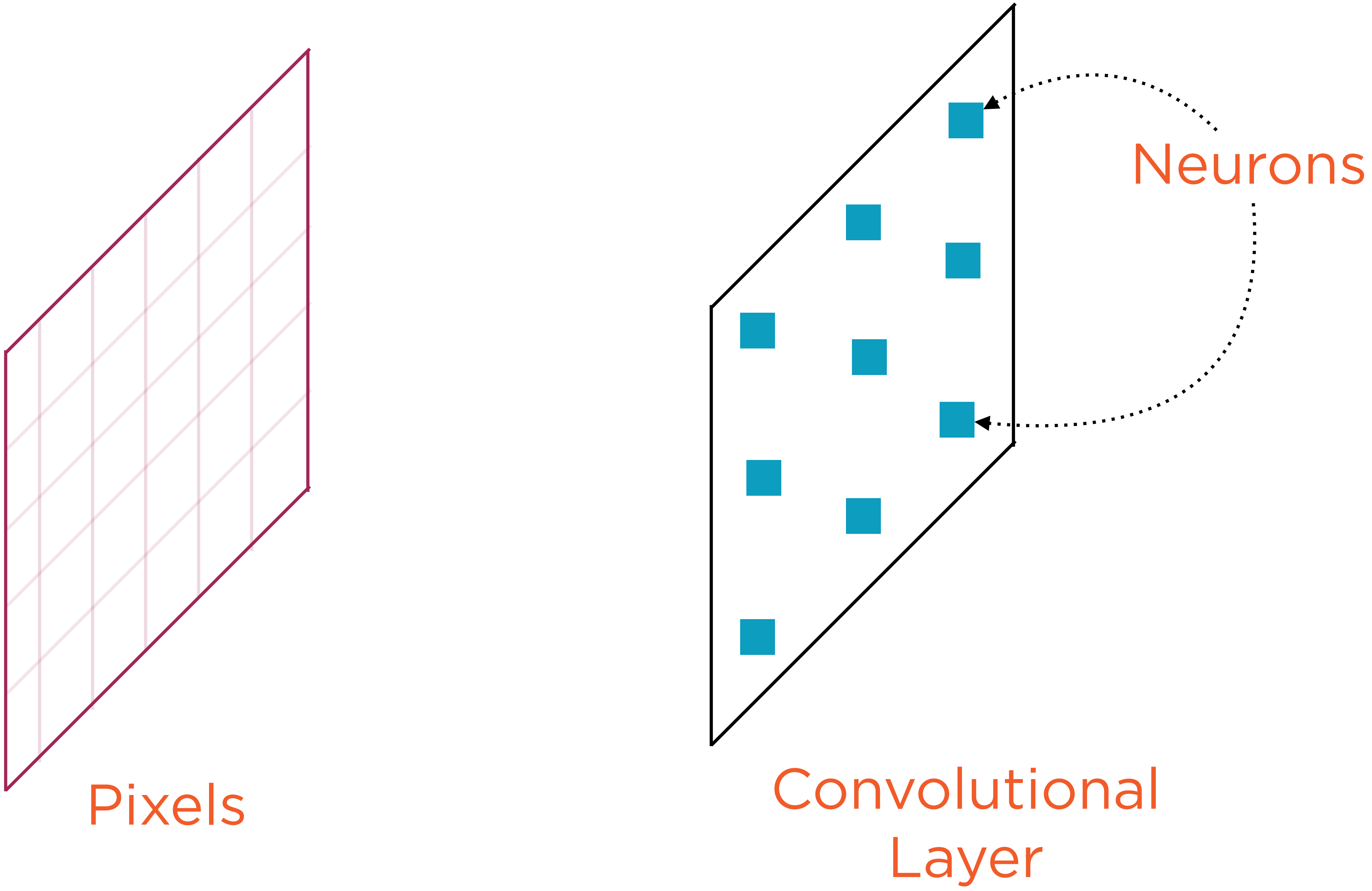
Pixels



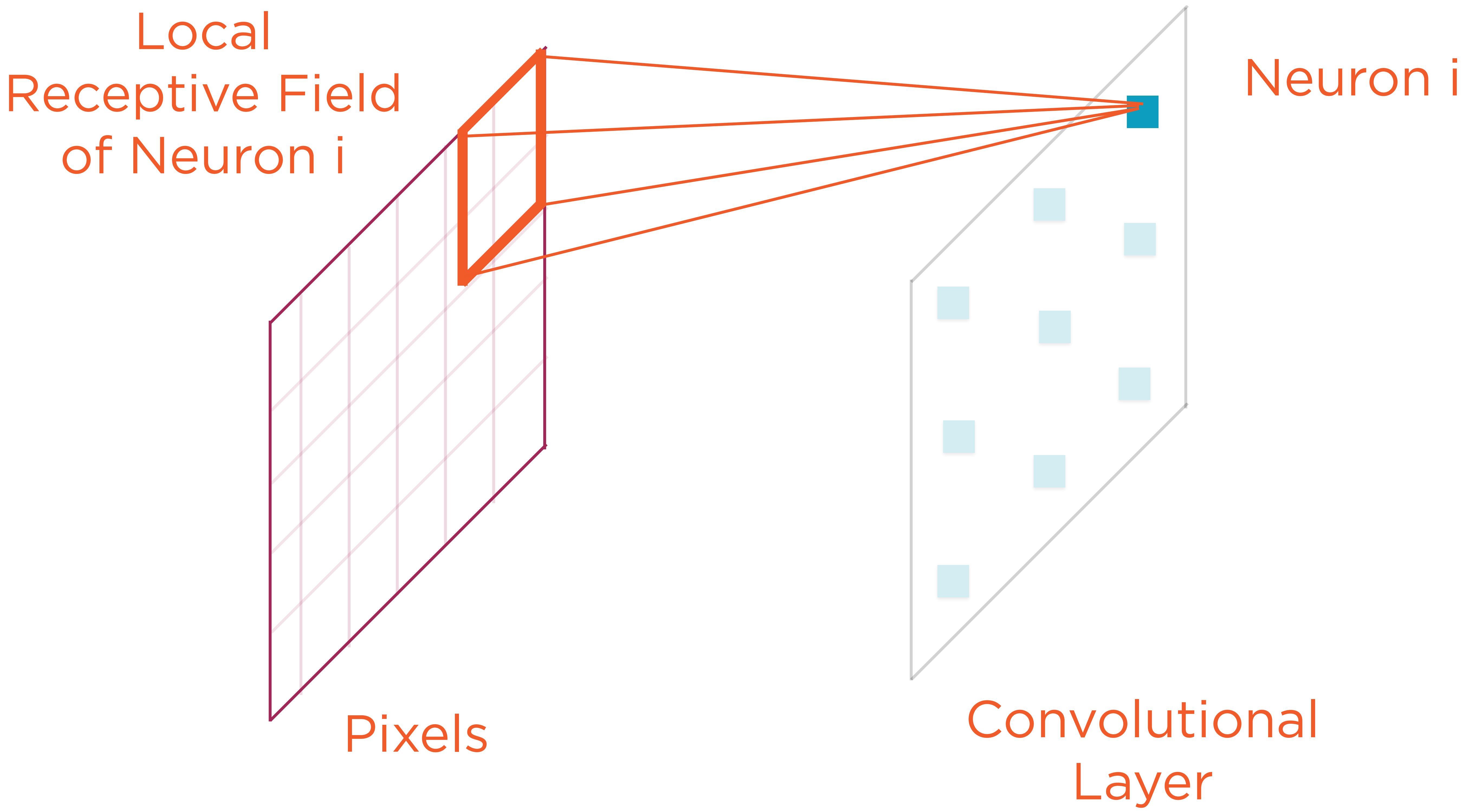
Feature Map

Feature maps are convolutional layers generated by applying a convolutional kernel to the input

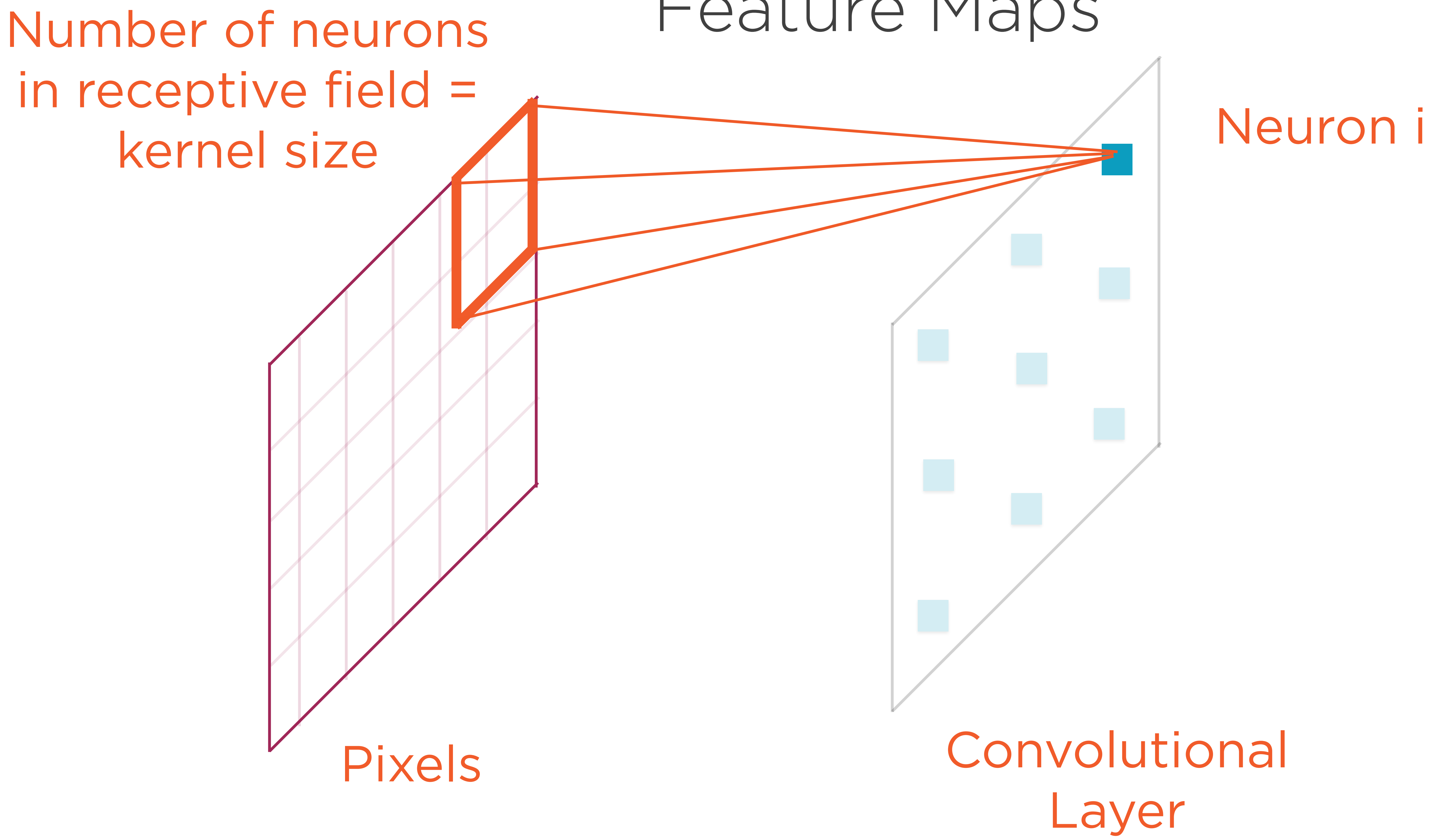
Feature Maps



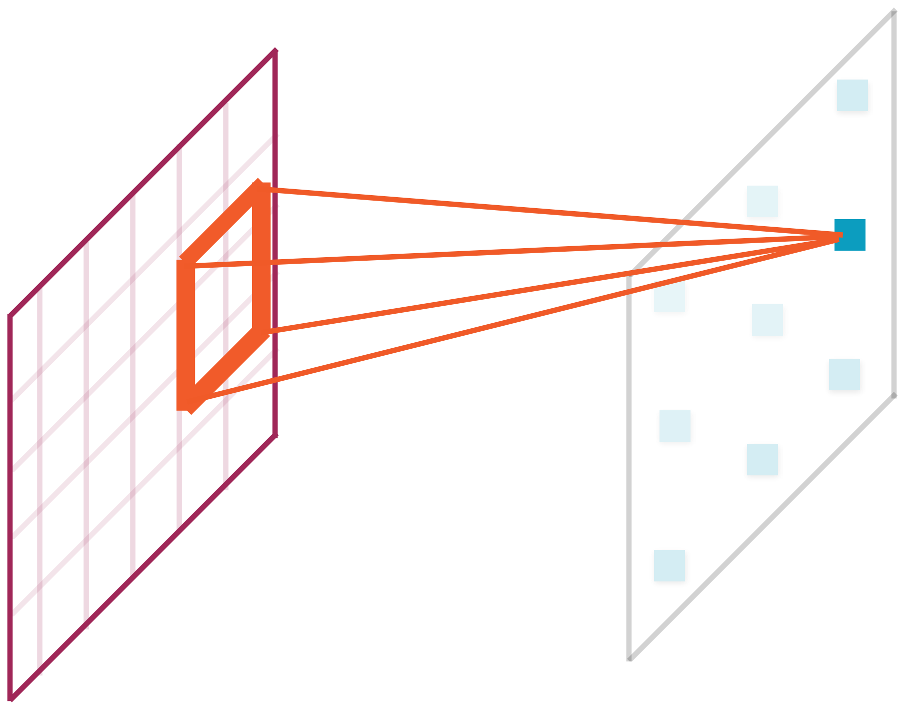
Feature Maps



Feature Maps



Kernel Size

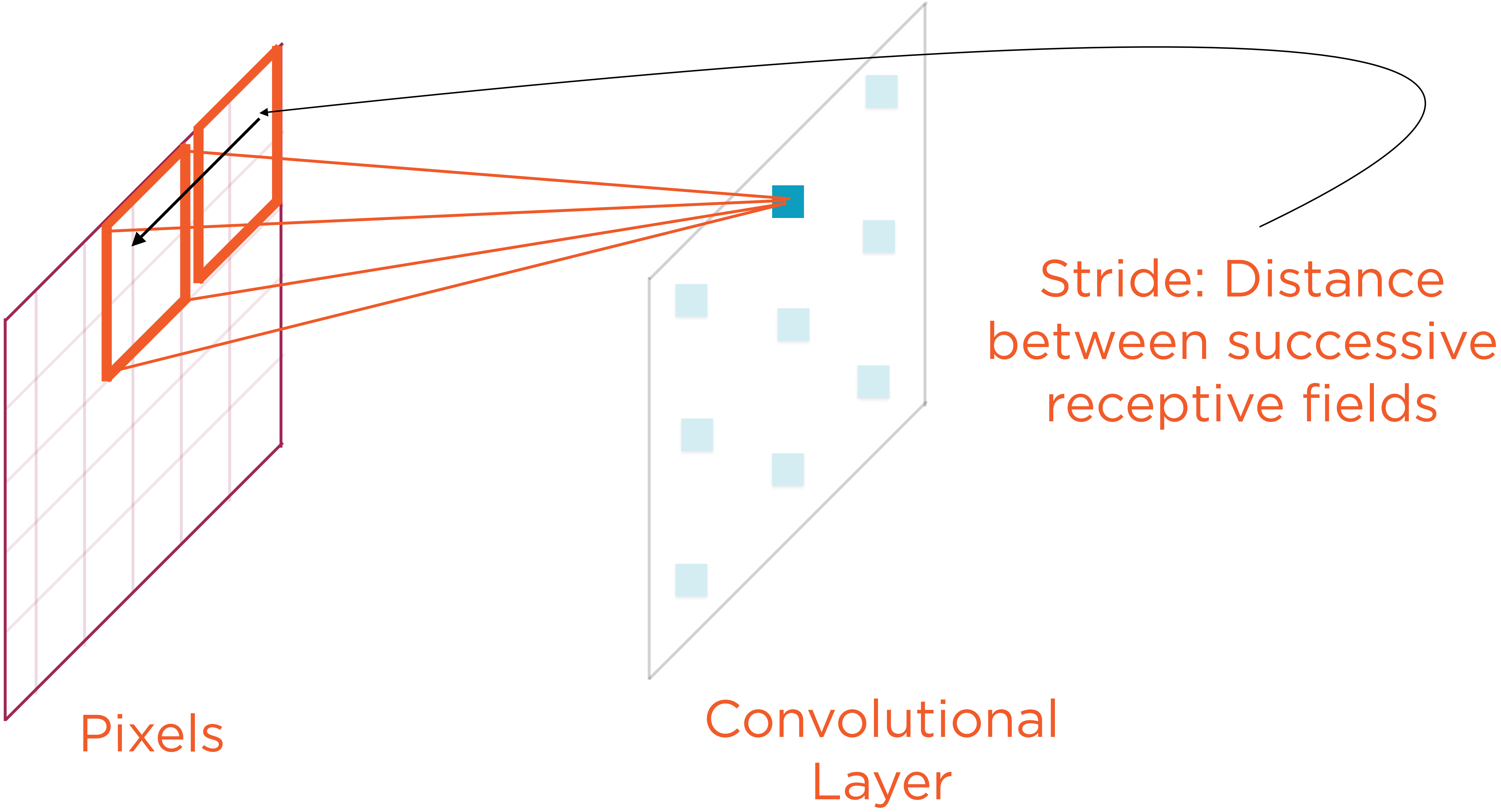


Convolutional kernel size usually expressed in terms of width and height of receptive area

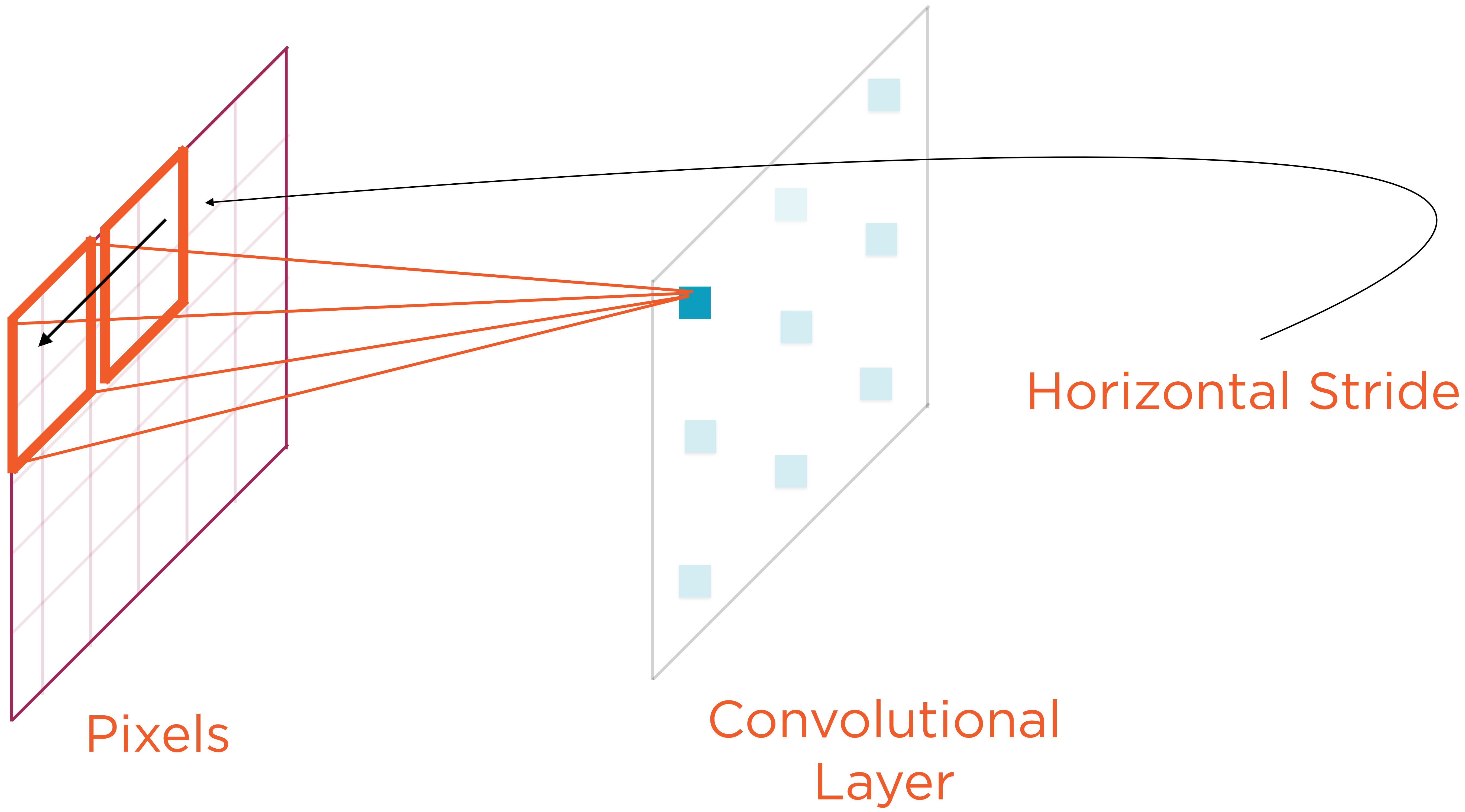
Use small convolutional kernels, more efficient

Stacking two 3x3 kernels is preferable to one 9x9 kernel

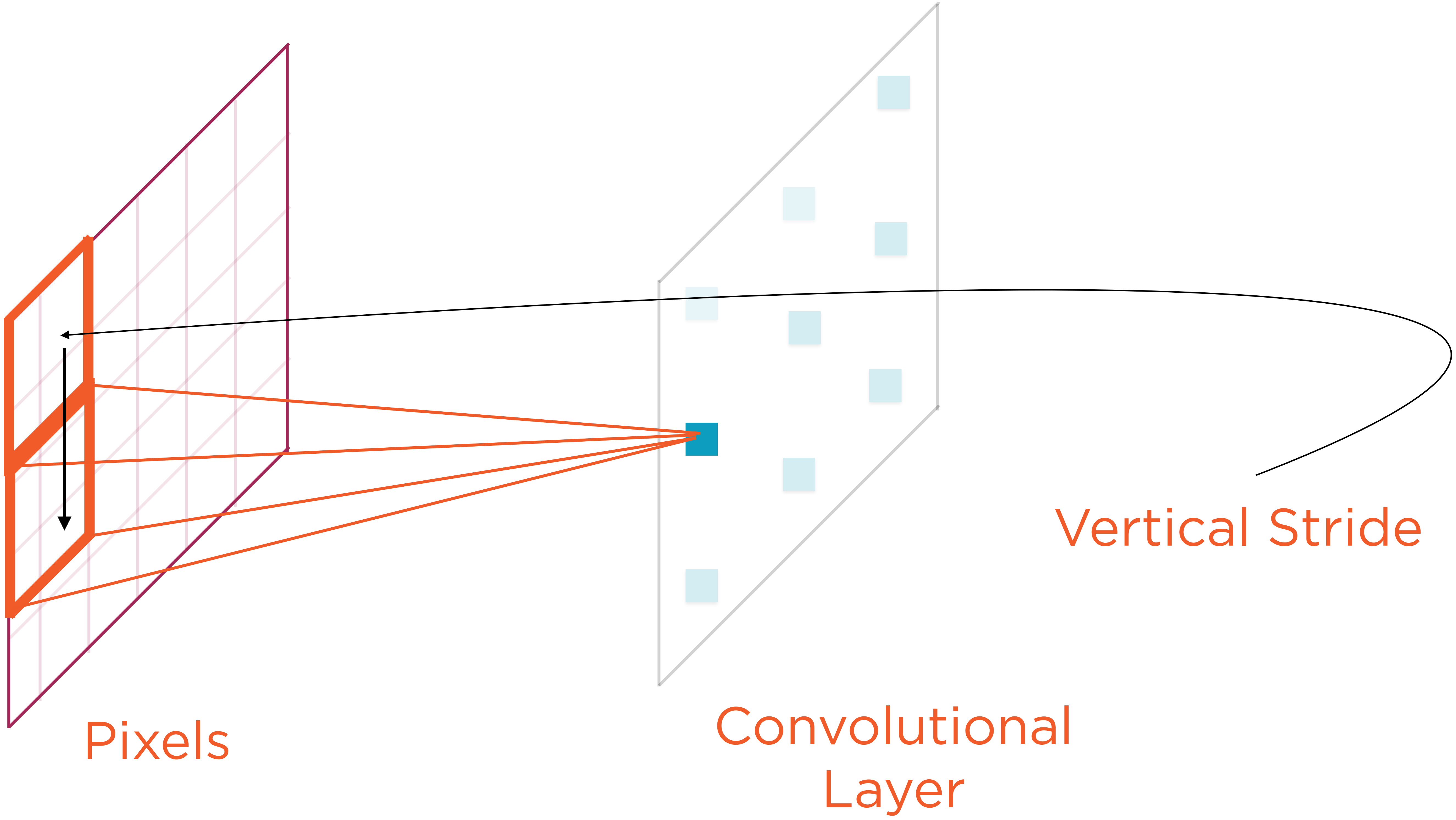
Feature Maps



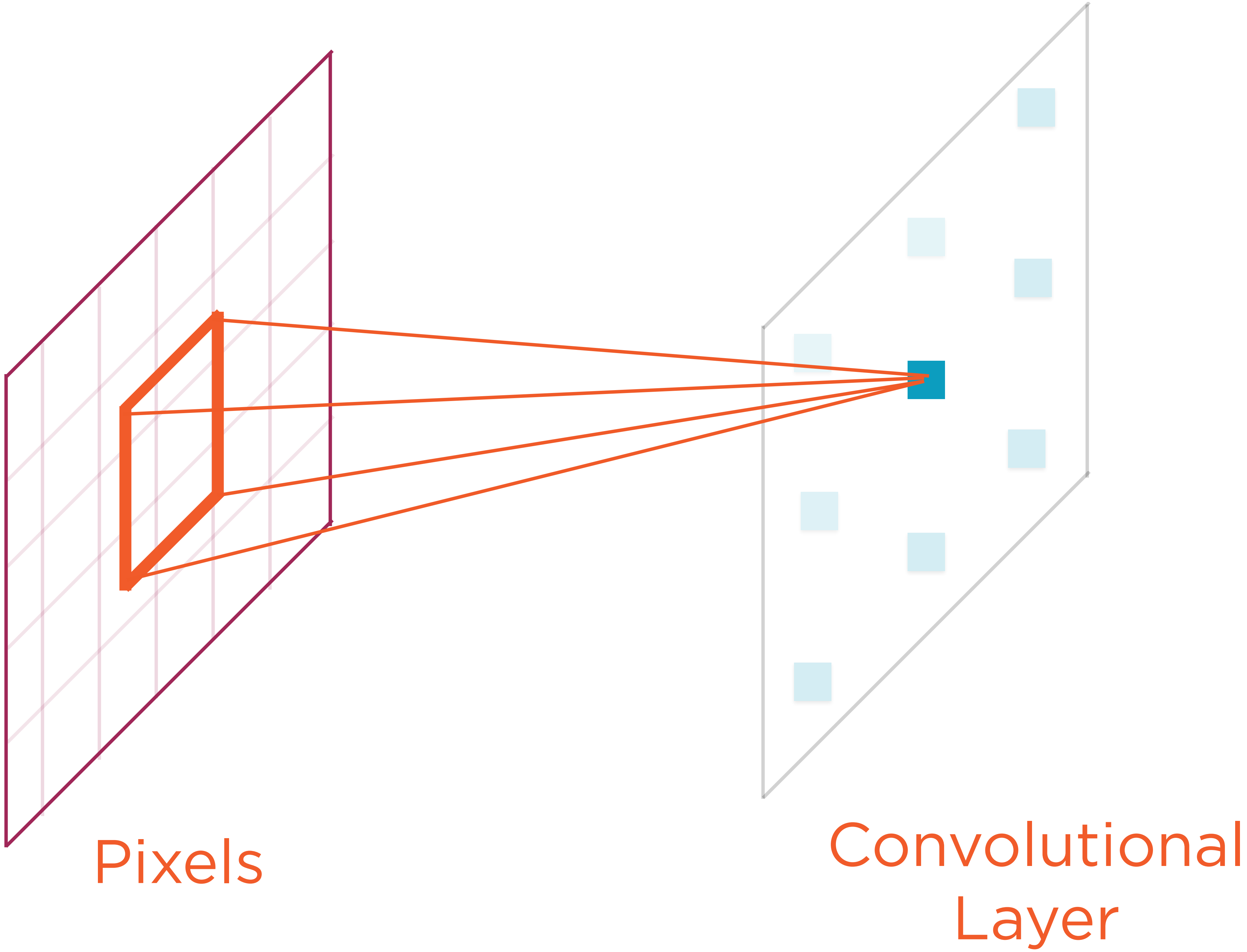
Feature Maps



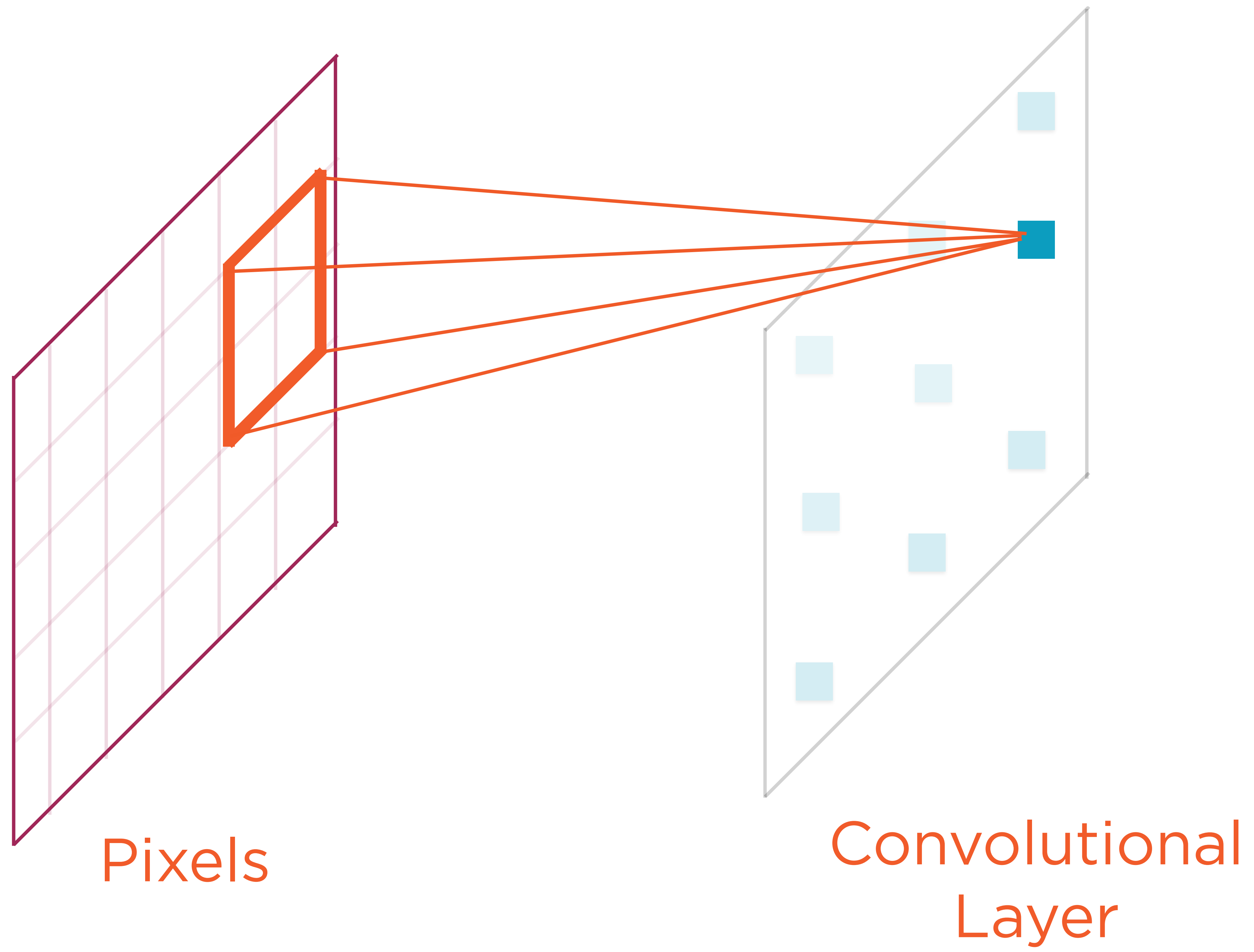
Feature Maps



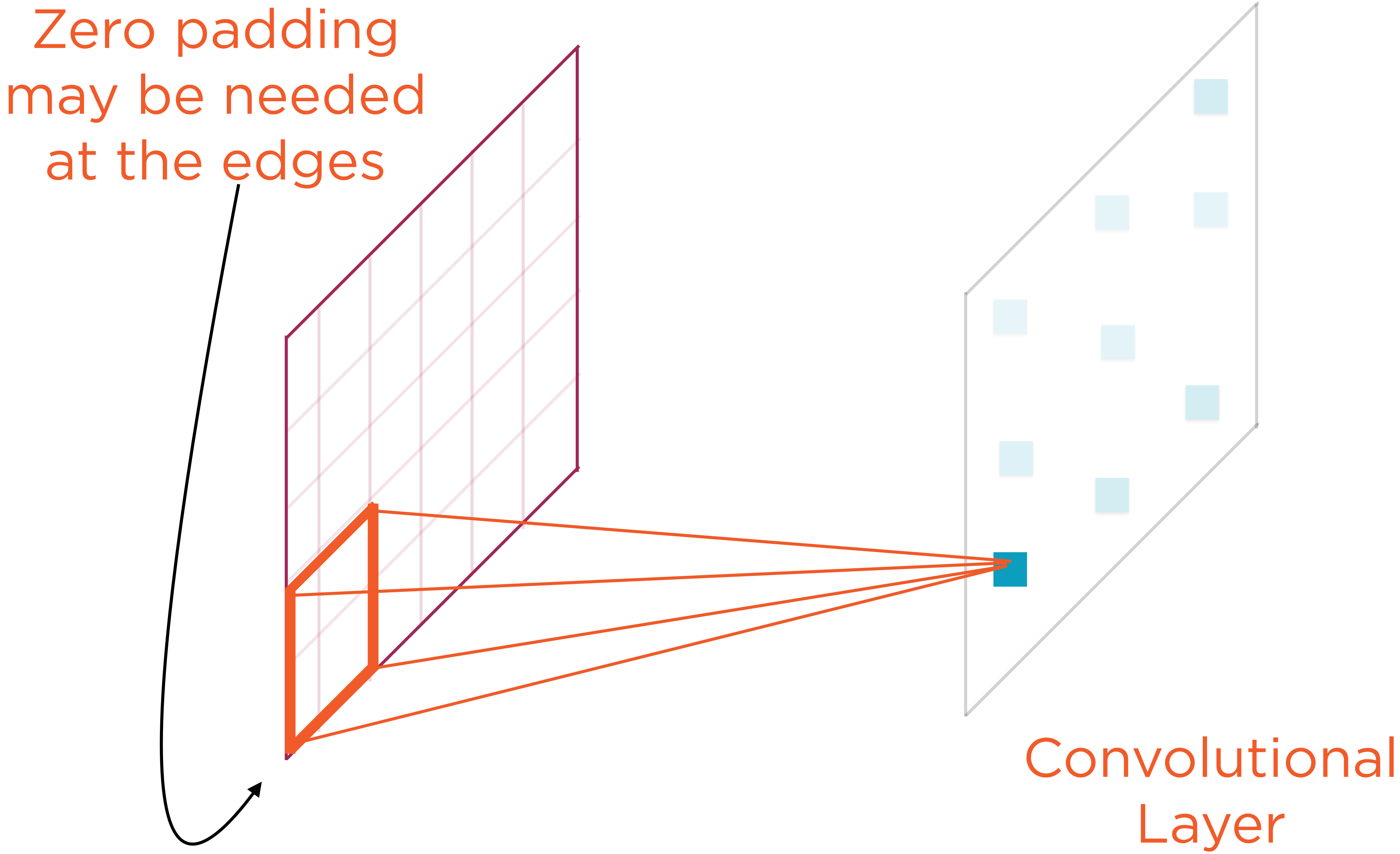
Feature Maps



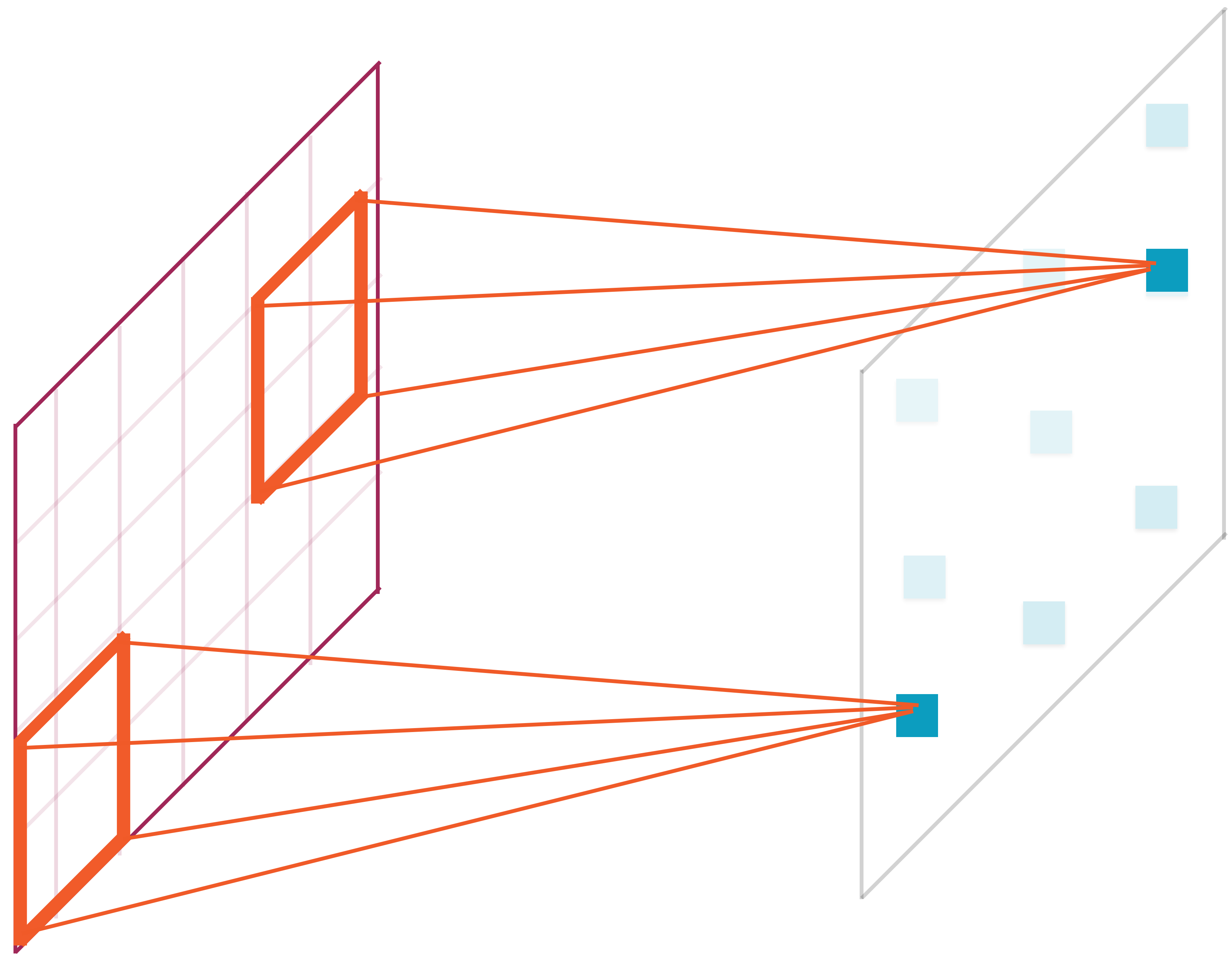
Feature Maps



Feature Maps

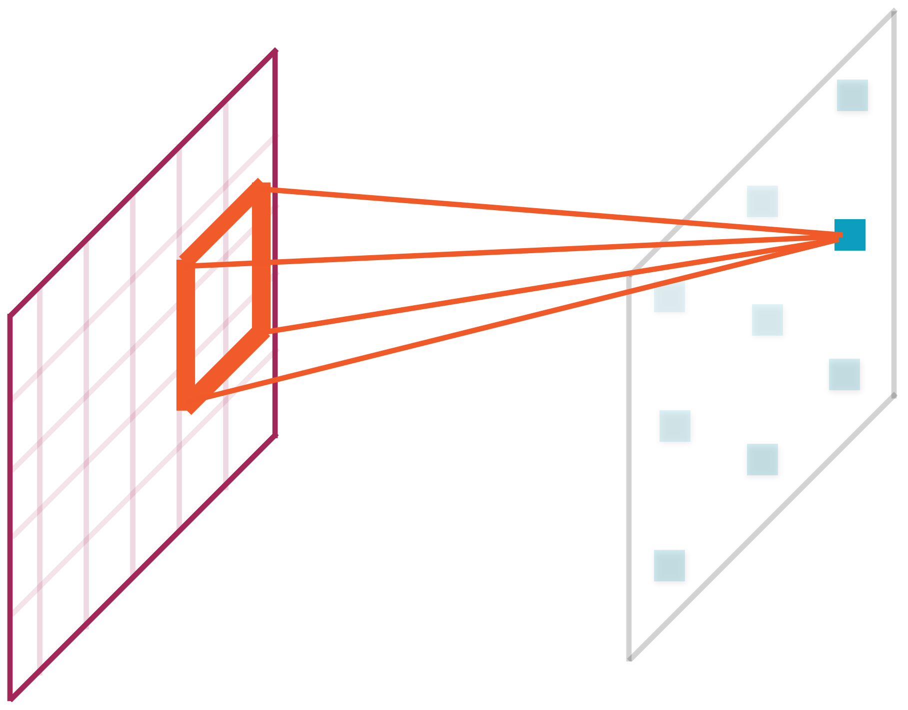


Feature Maps



Sparse, not Dense

Feature Maps

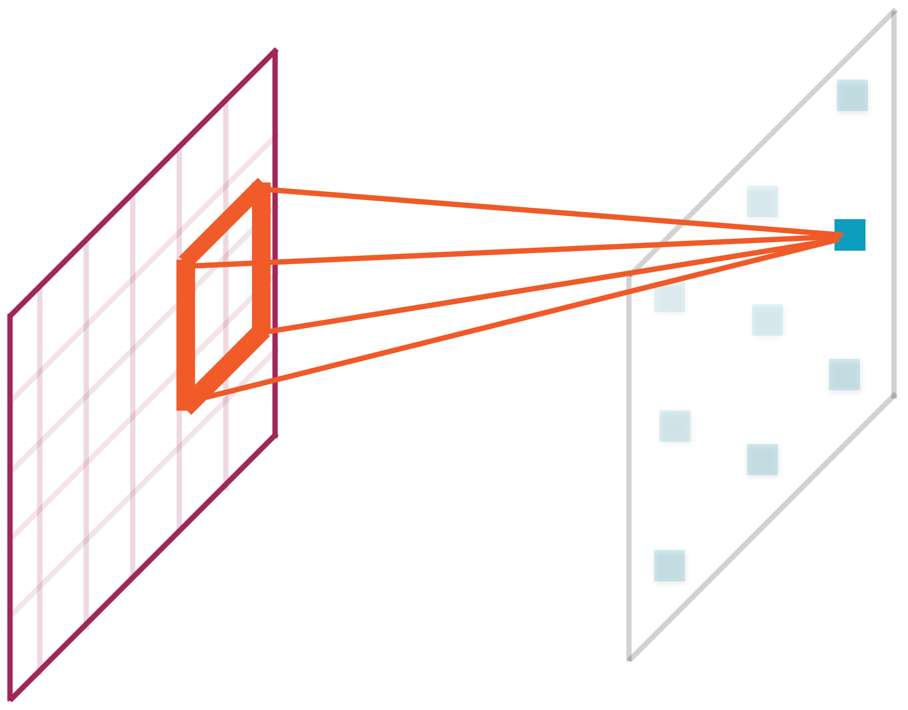


All neurons in a feature map have the same weights and biases

Two big advantages over DNNs

- Dramatically fewer parameters to train
- CNN can recognize feature patterns independent of location

Feature Maps

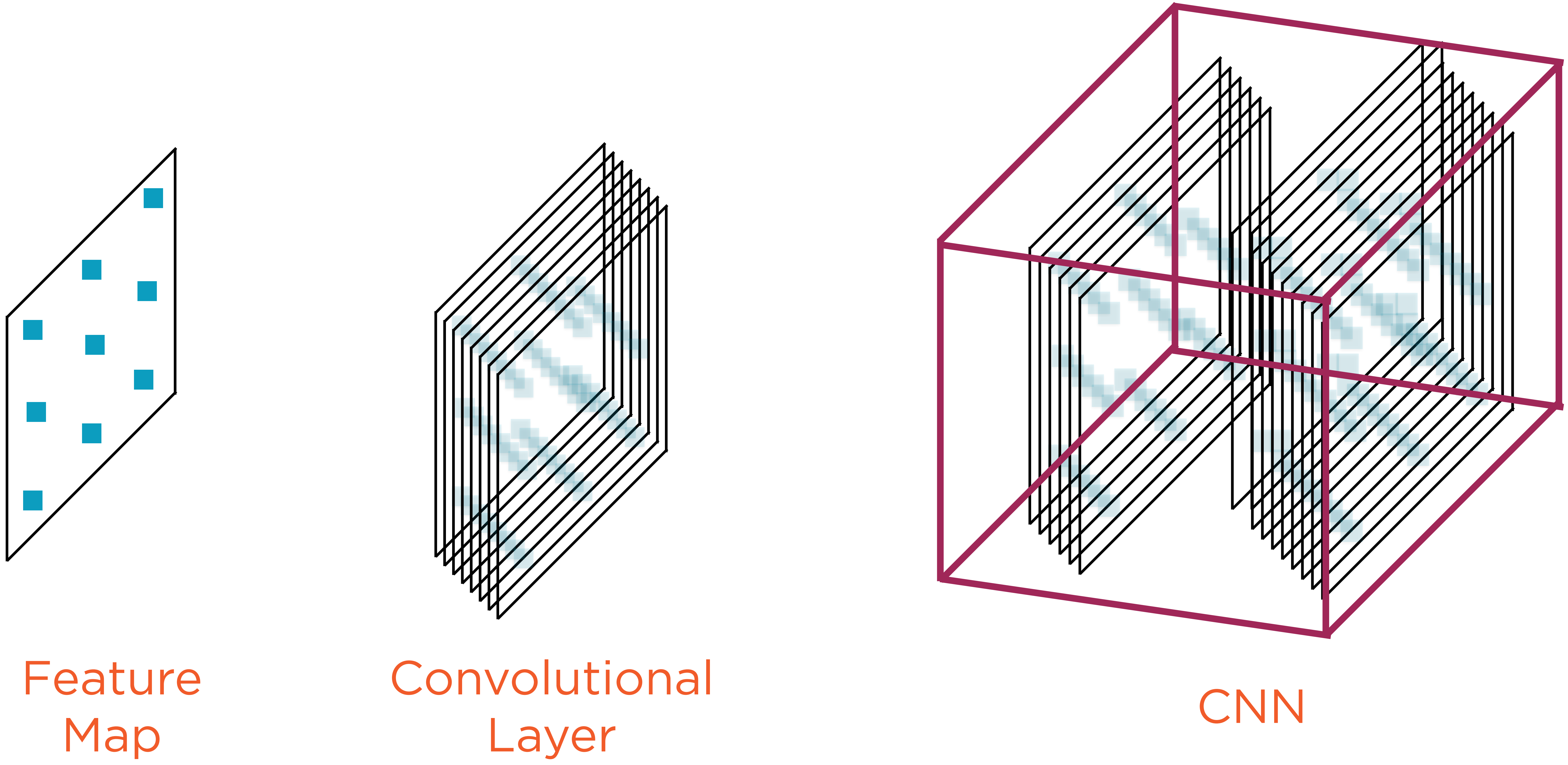


The parameters of all neurons in a feature map are collectively called the filter

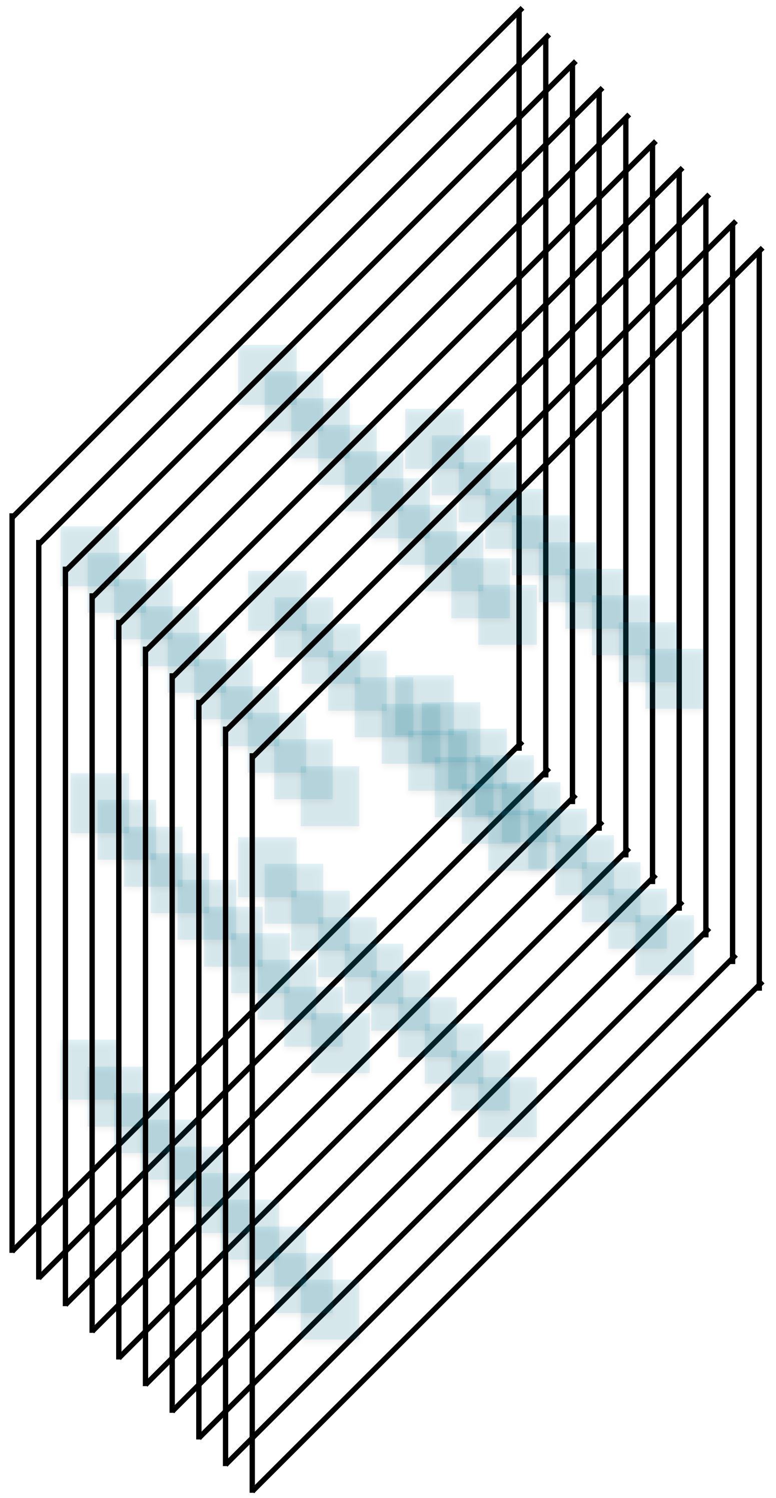
Why filter?

Because weights highlight (filter) specific patterns from the input pixels

CNNs



Convolutional Layer



Each convolutional layer consists of several feature maps of equal sizes

The different feature maps have different parameters

Pooling Layers

Two Kinds of Layers in CNNs

Convolution

Local receptive field

Pooling

Subsampling of inputs

Pooling

4

4

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

Matrix

Max,
2x2 filter,
stride = 2

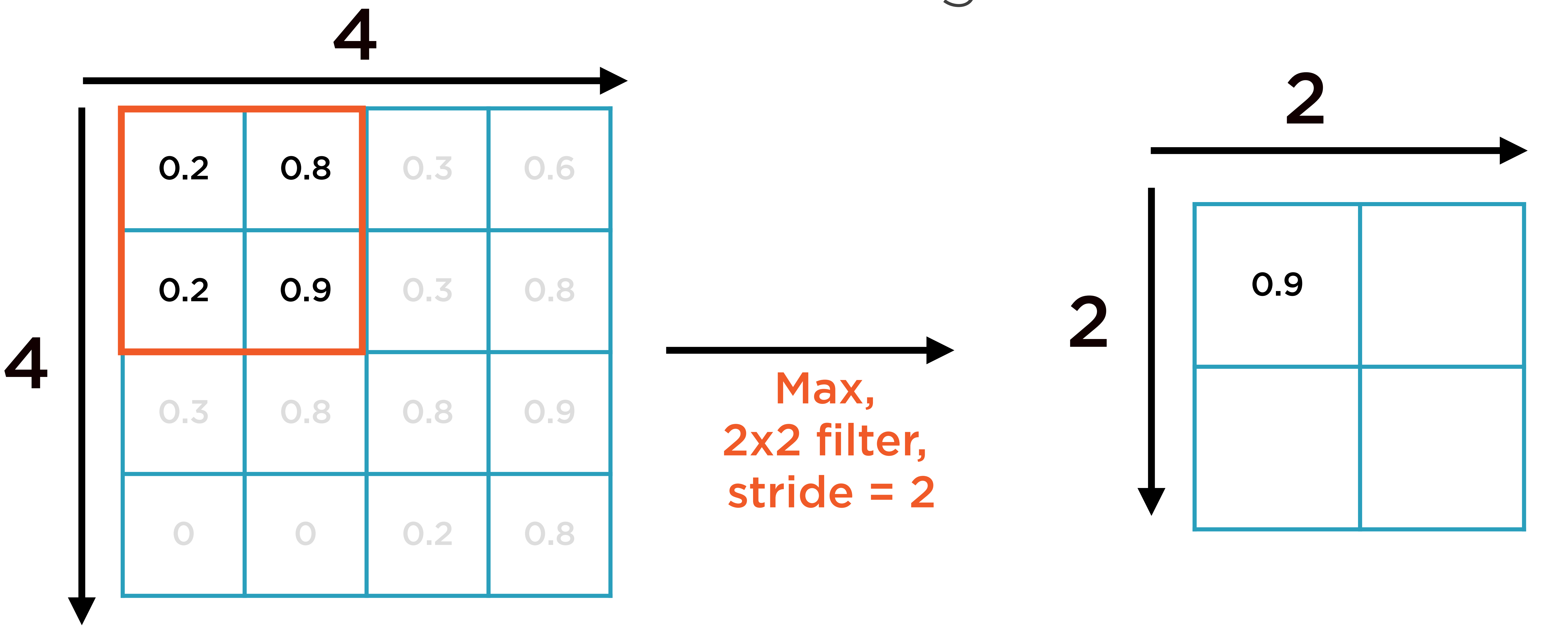
2

2

0.9	0.8
0.8	0.9

Pooling Result

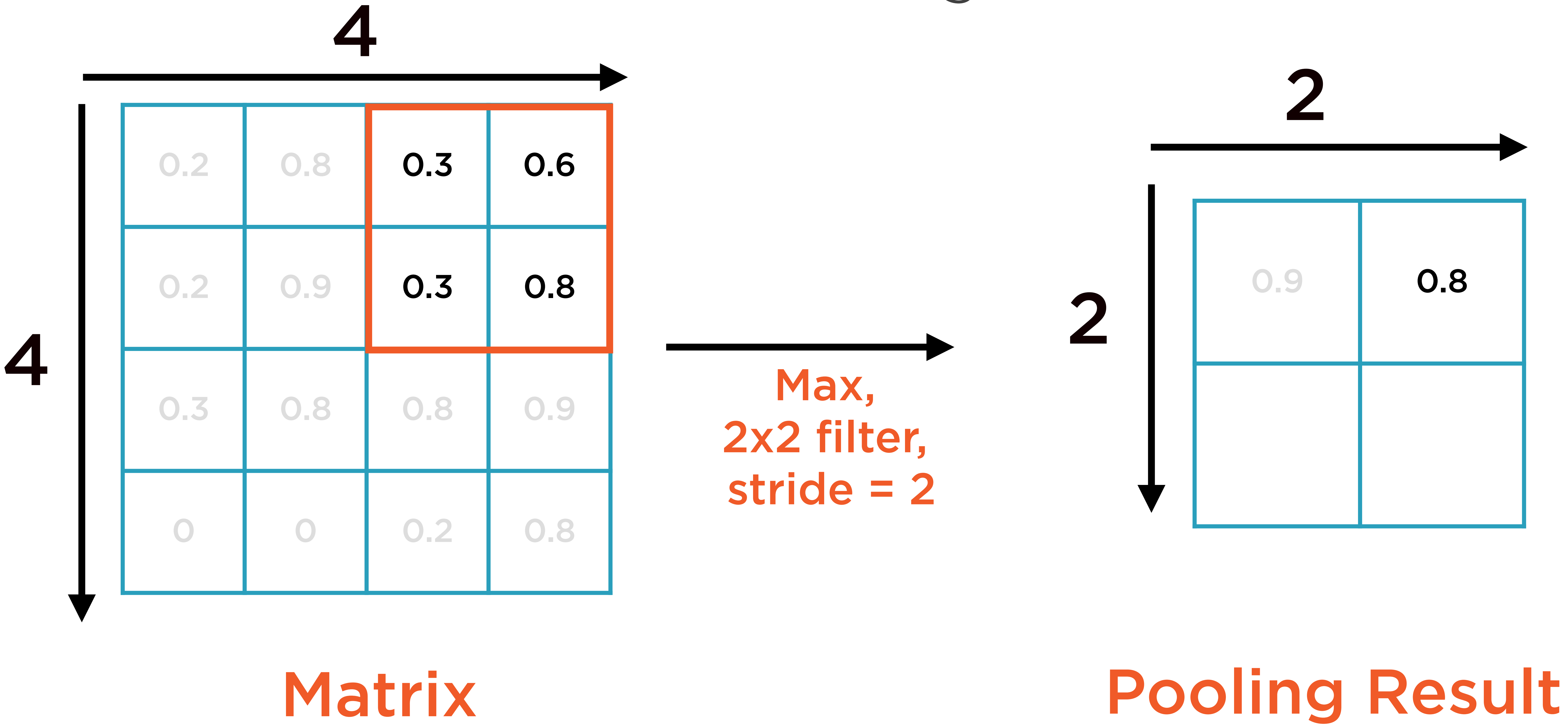
Pooling



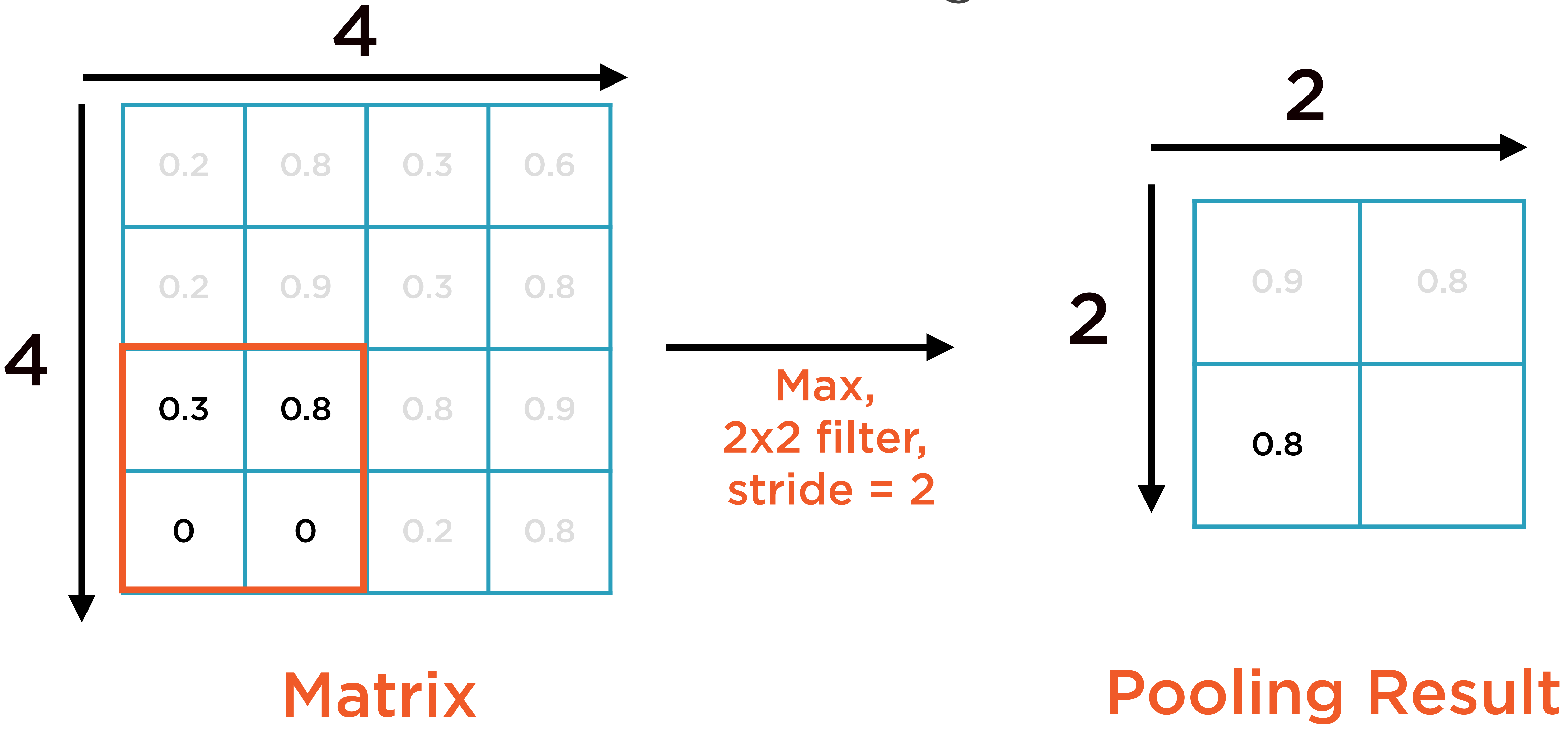
Matrix

Pooling Result

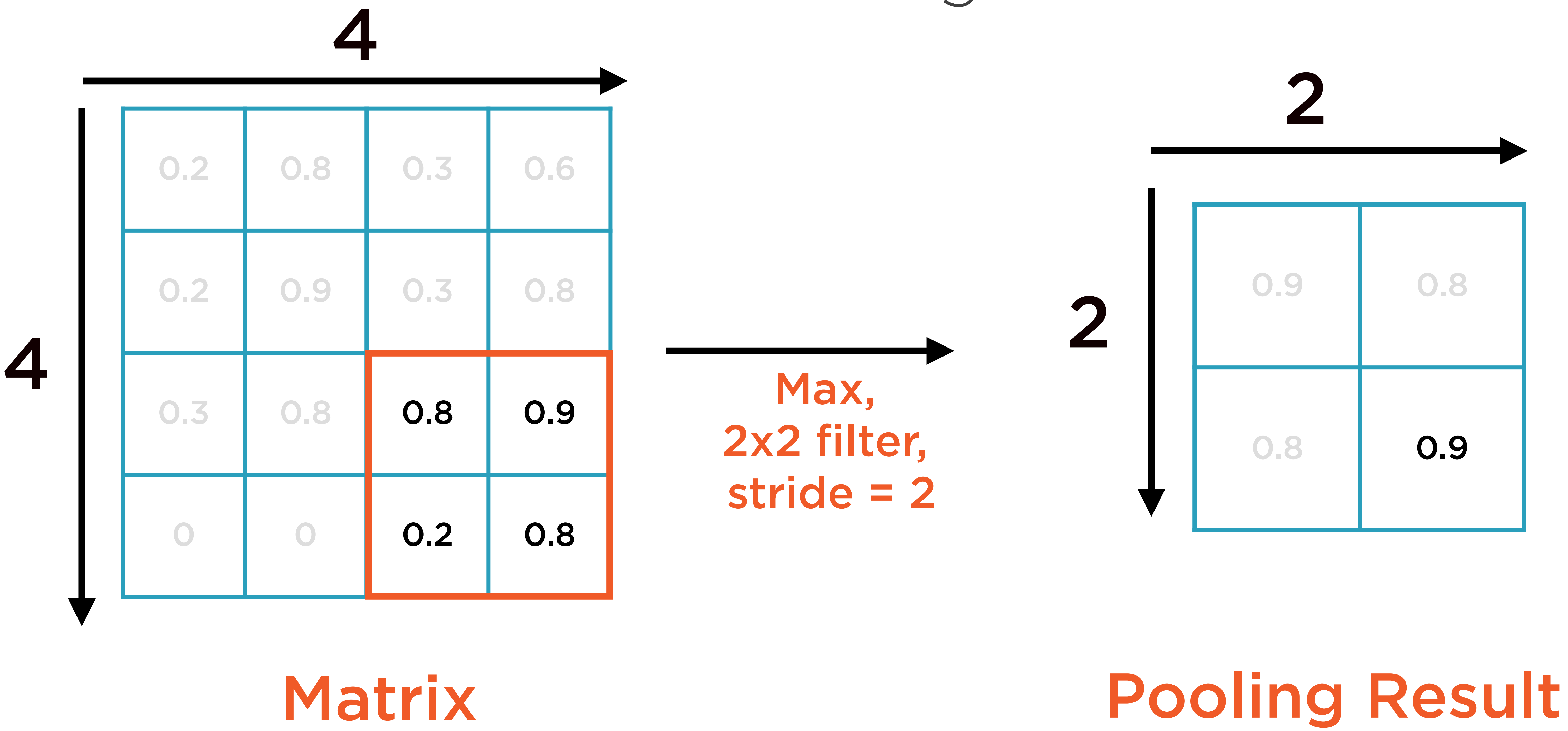
Pooling

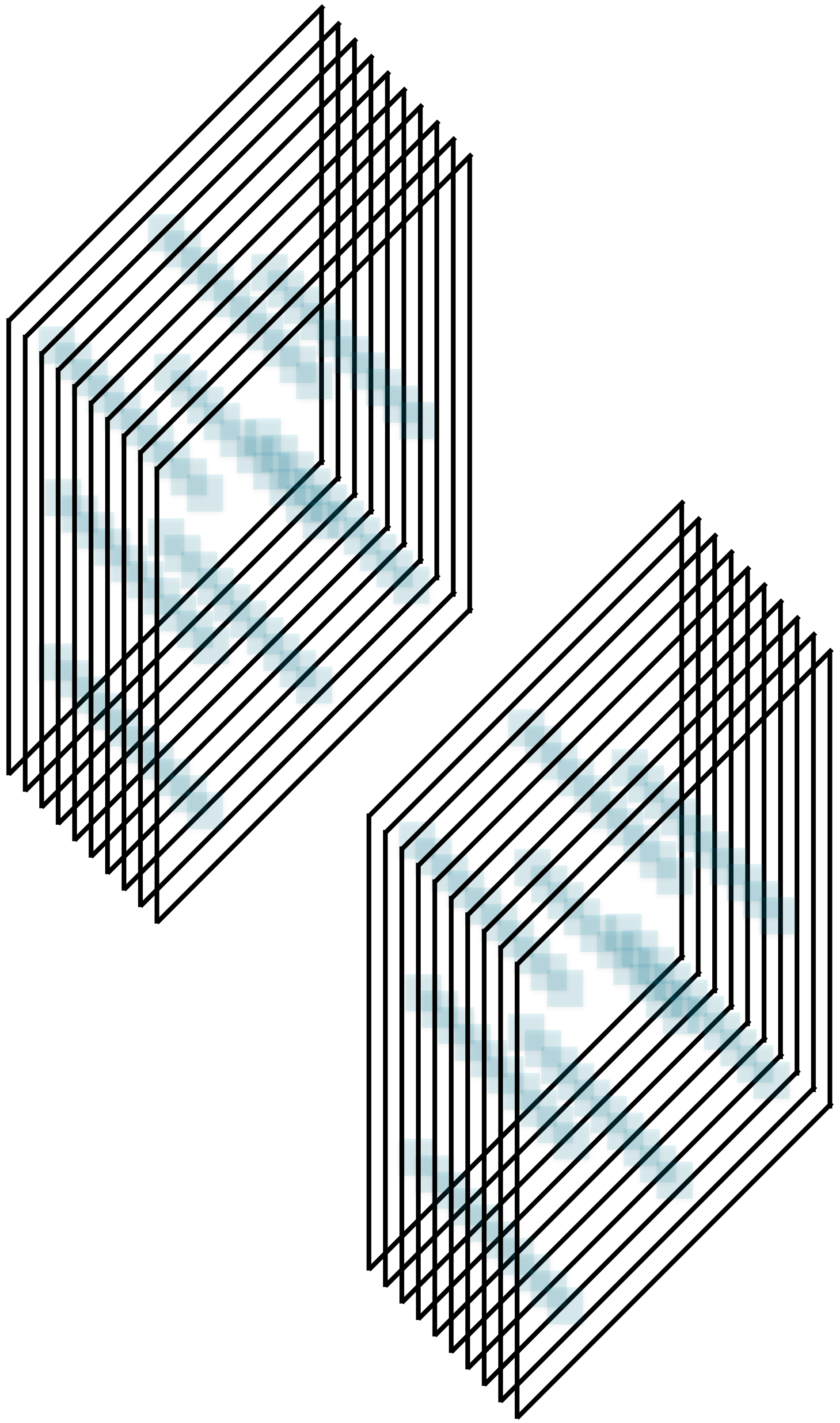

4
4
0.2 0.8 0.3 0.6
0.2 0.9 0.3 0.8
0.3 0.8 0.8 0.9
0 0 0.2 0.8
Max,
2x2 filter,
stride = 2
2
2
0.9 0.8
Matrix
Pooling Result

Pooling


4
4
0.2 0.8 0.3 0.6
0.2 0.9 0.3 0.8
0.3 0.8 0.8 0.9
0 0 0.2 0.8
Max,
2x2 filter,
stride = 2
2
2
0.9 0.8
0.8
Matrix
Pooling Result

Pooling


4
4
0.2
0.8
0.3
0.6
0.2
0.9
0.3
0.8
0.3
0.8
0.8
0.9
0
0
0.2
0.8
Max,
2x2 filter,
stride = 2
2
2
0.9
0.8
0.8
0.9
Matrix
Pooling Result

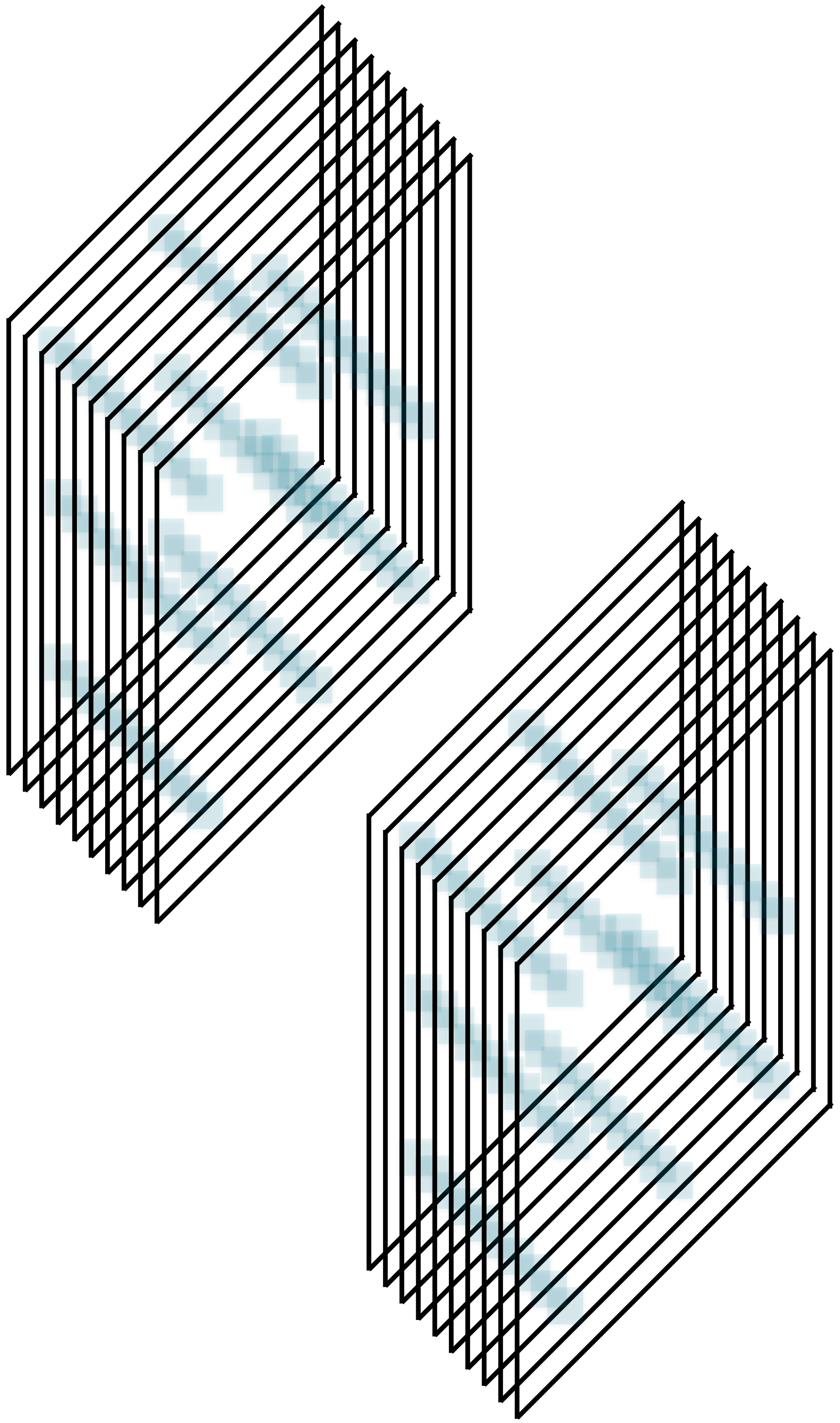


Pooling Layers

Neurons in a pooling layer have no weights or biases

A pooling neuron simply applies some aggregation function to all inputs

Max, sum, average

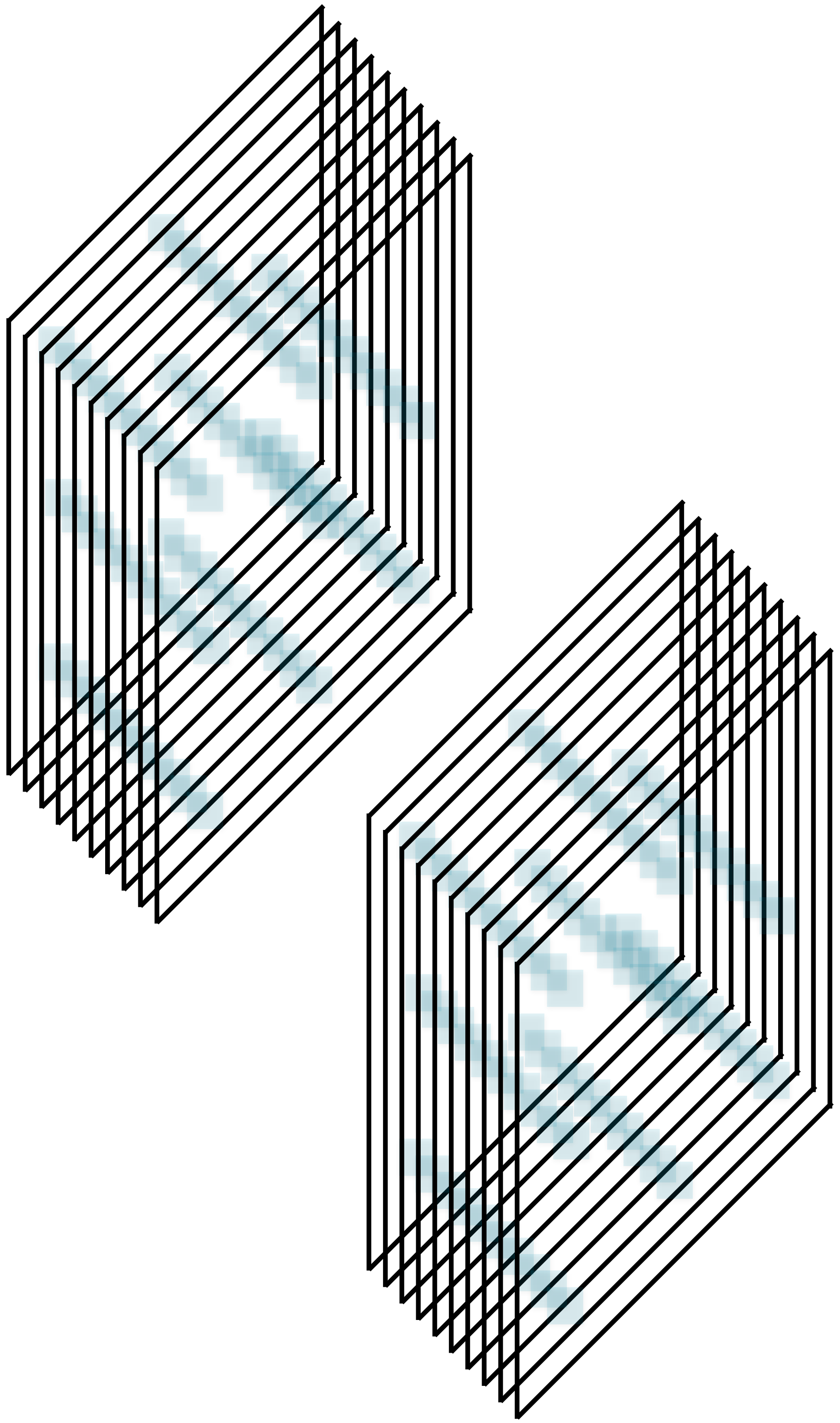


Pooling Layers

Why use them?

- Greatly reduce memory usage during training
- Mitigate overfitting (via subsampling)
- Make NN recognize features independent of location (location invariance)

Pooling Layers



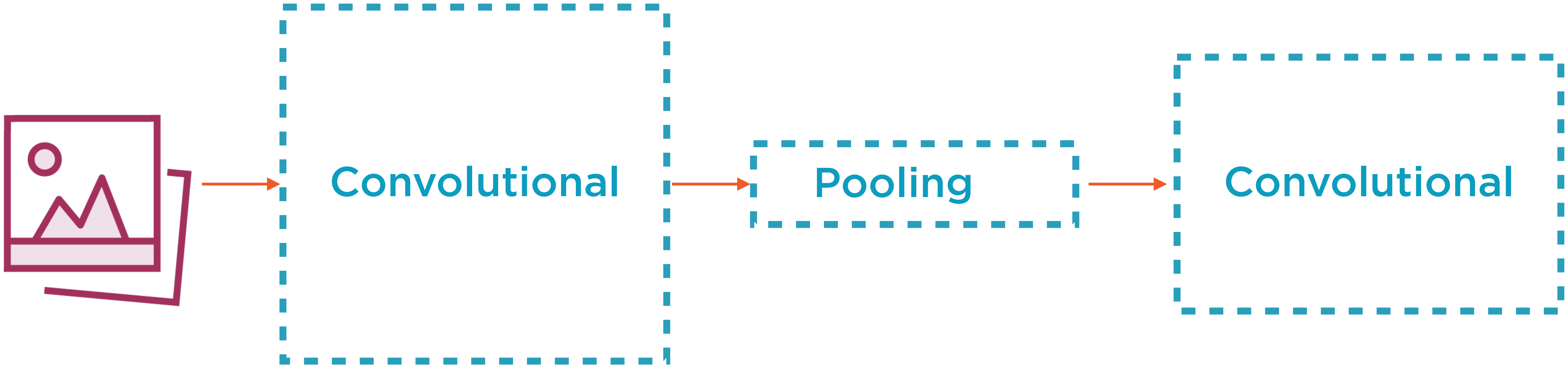
Pooling layers typically act on each channel independently

So, usually, output area < input area but

Output depth = Input depth

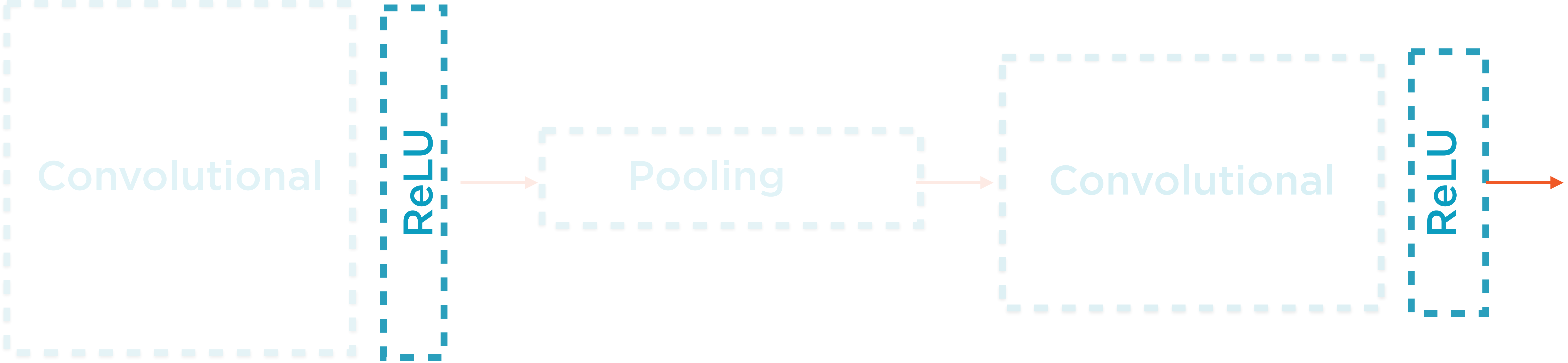
CNN Architectures

Typical CNN Architecture



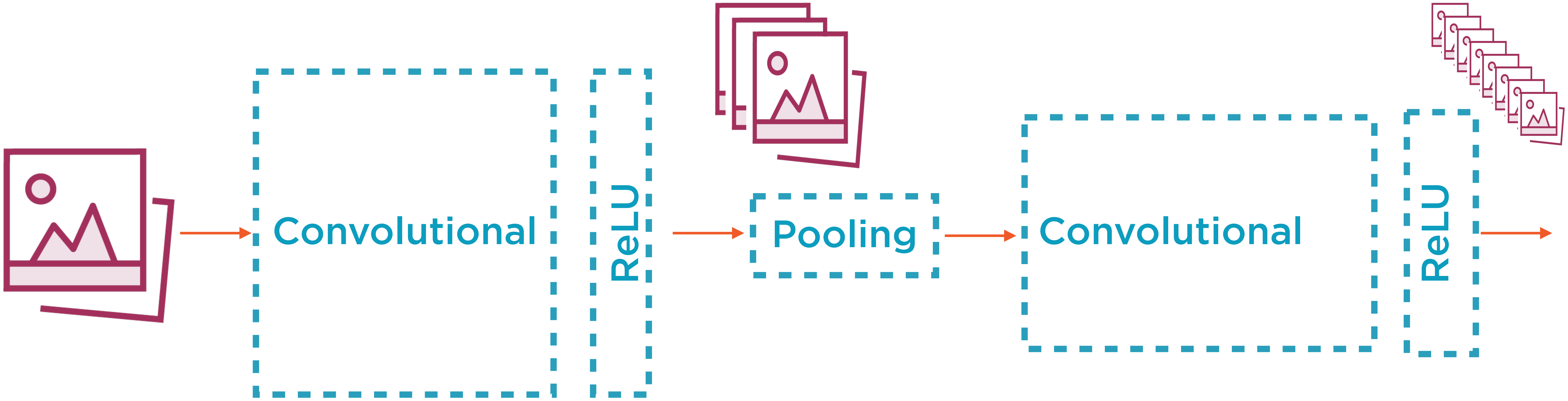
Alternating groups of convolutional and pooling layers

Typical CNN Architecture



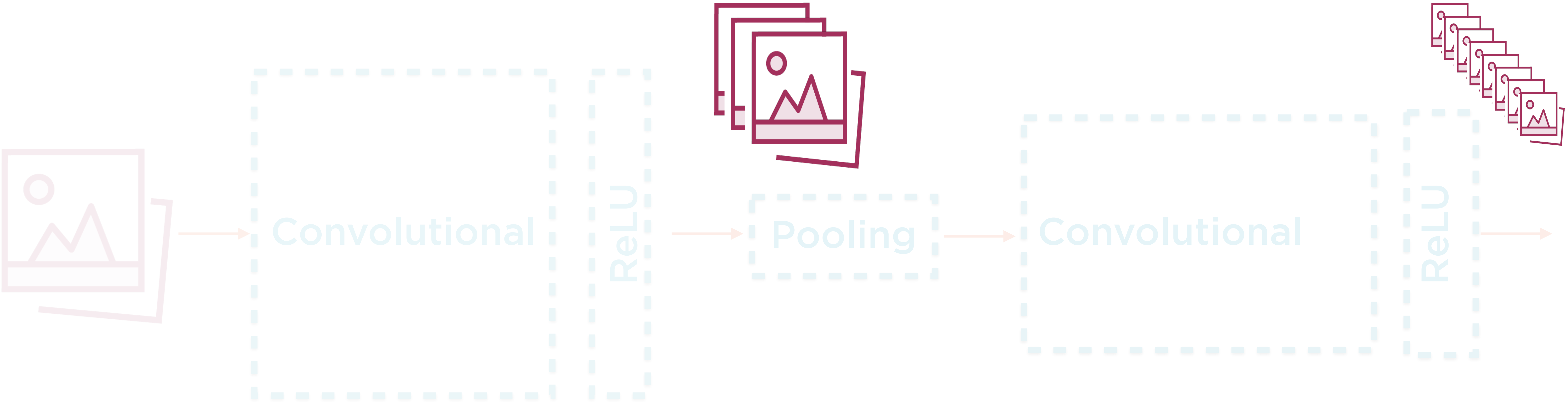
Each group of convolutional layers usually followed by a ReLU layer

Typical CNN Architecture



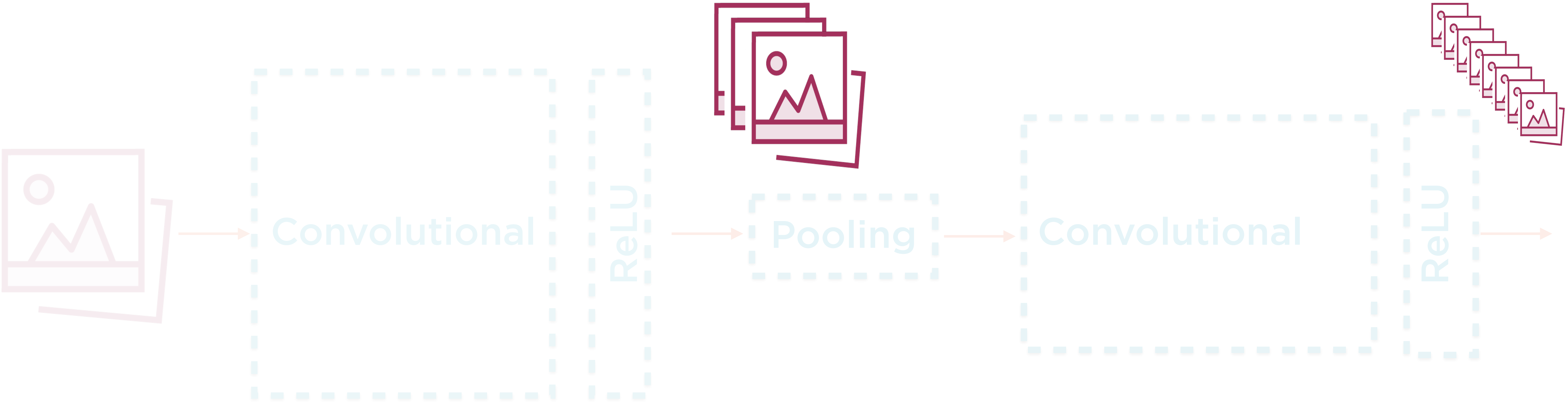
The output of each layer is also an image

Typical CNN Architecture



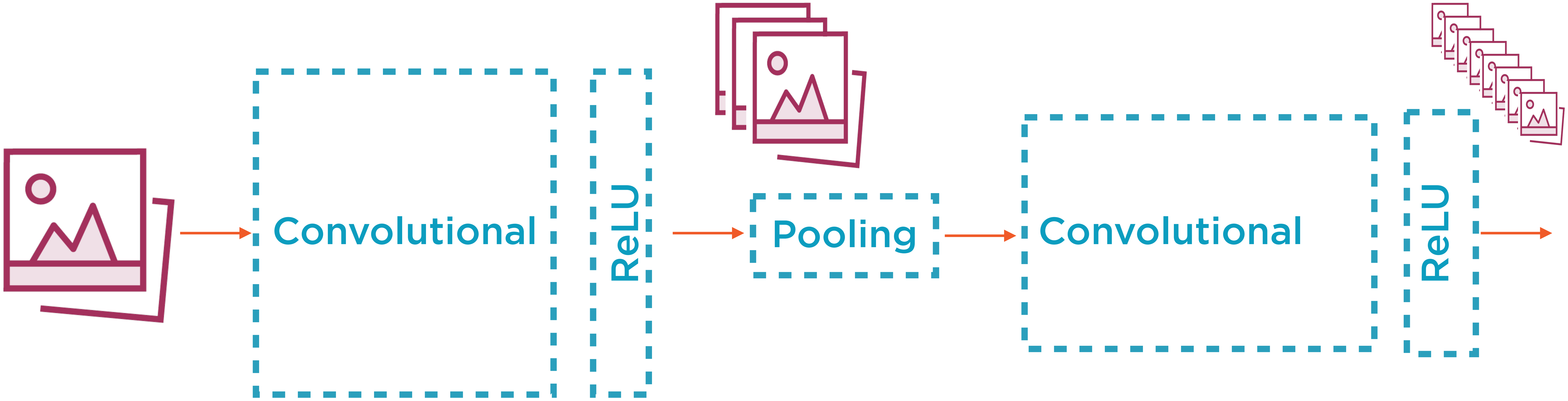
However successive outputs are smaller and smaller (due to pooling layers)

Typical CNN Architecture



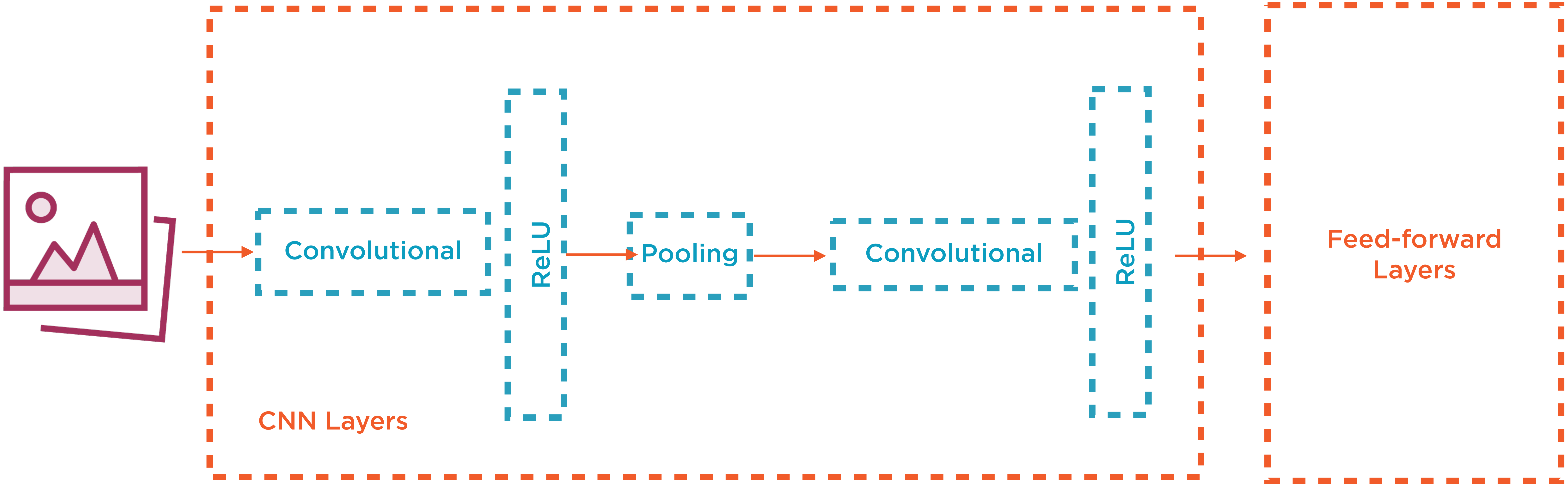
As well as deeper and deeper (due to feature maps in the convolutional layers)

Typical CNN Architecture



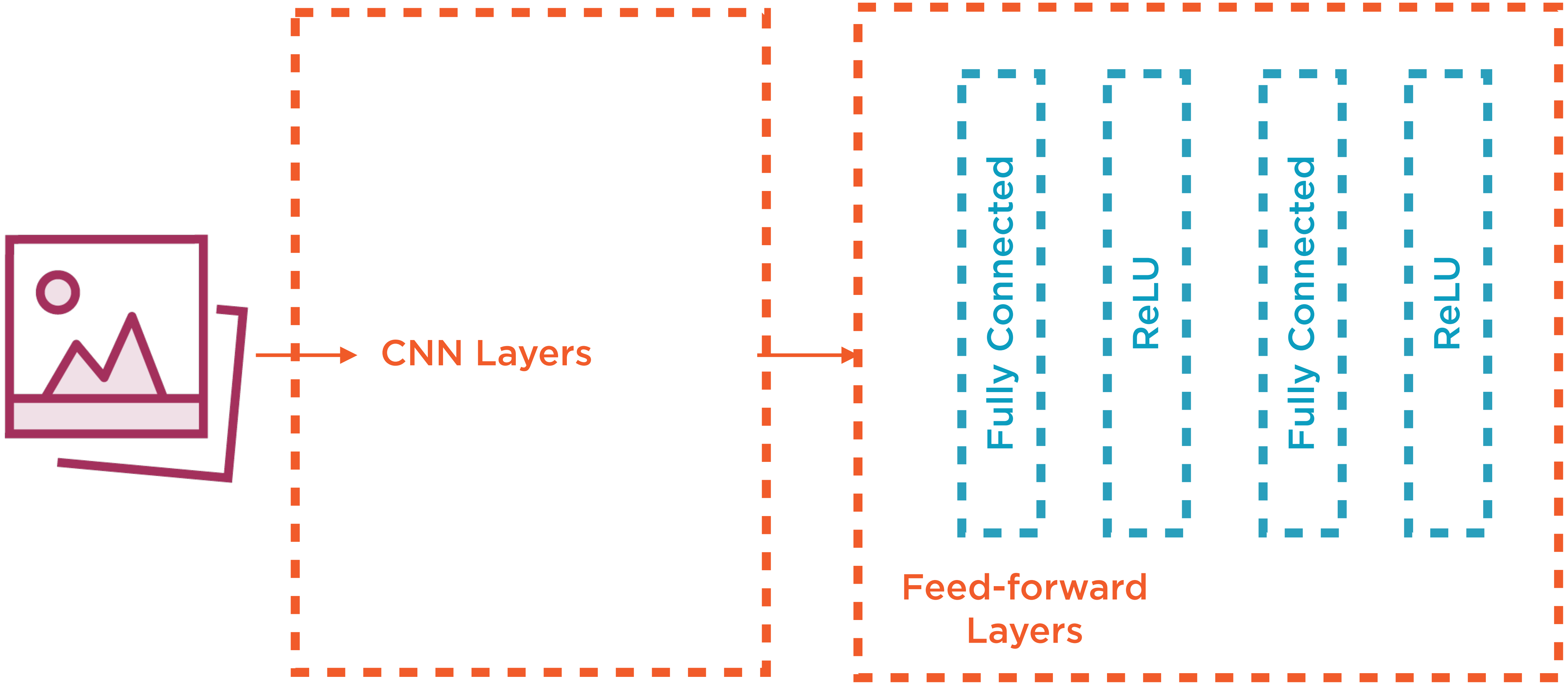
This entire set of layers is then fed into a regular, feed-forward neural network

Typical CNN Architecture



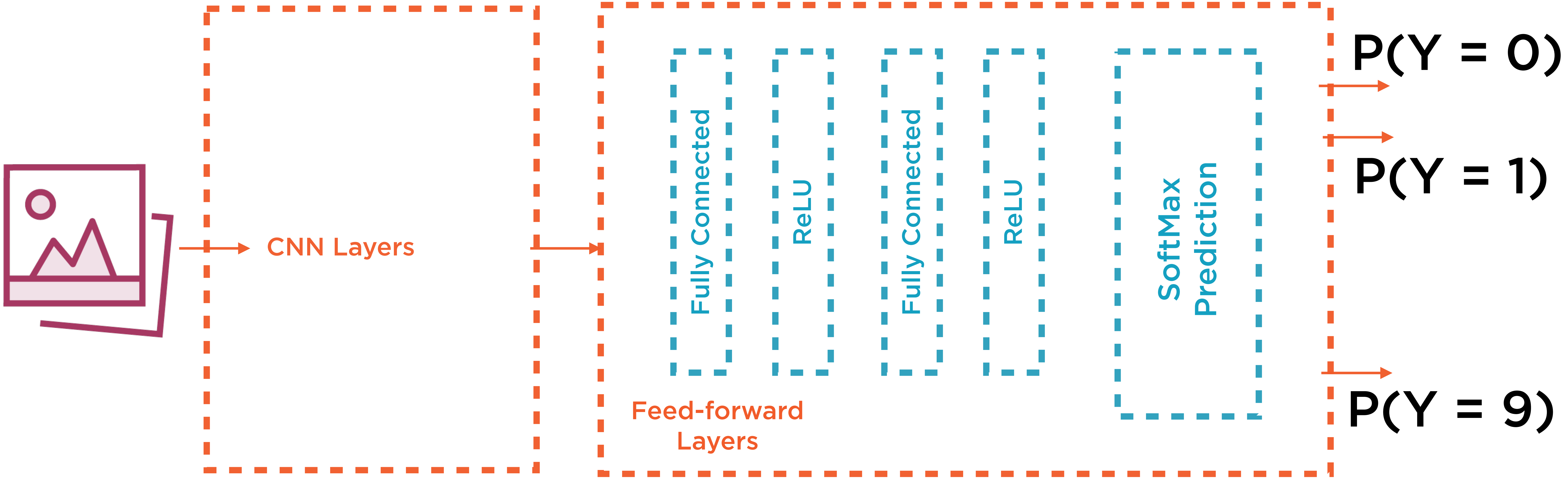
This entire set of layers is then fed into a regular, feed-forward neural network

Typical CNN Architecture



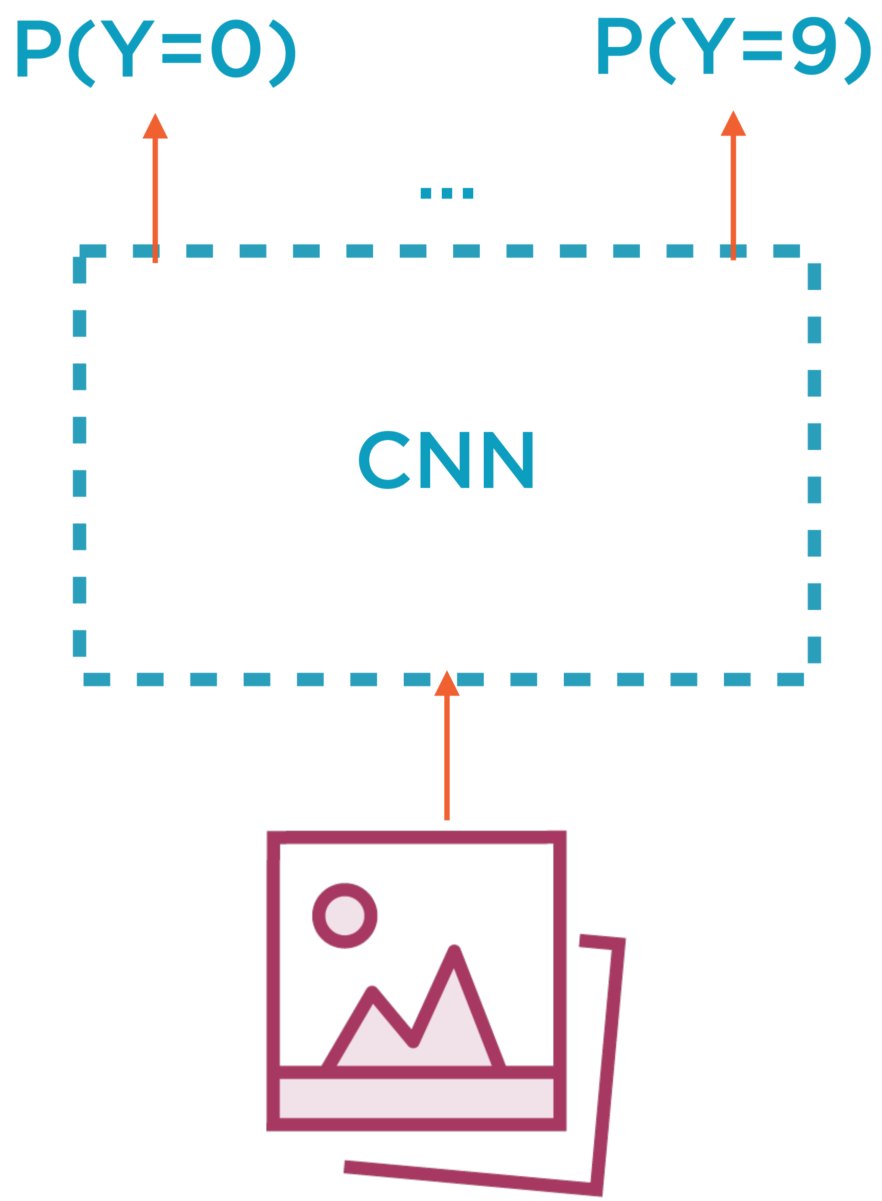
This feed-forward has a few fully connected layers with ReLU activation

Typical CNN Architecture



This is the output layer, emitting probabilities

Typical CNN Architectures



Input is an image

Outputs are probabilities

Demo

Image classification using a convolutional neural network

Summary

Image classification models

Convolutional layers and pooling layers

Convolutional Neural Networks (CNNs) for image classification

Implementing CNNs in tf.keras for image classification

Up Next:

Building Unsupervised Machine Learning Models