Building Convolutional Neural Networks for Image Classification

Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Narrow and wide convolution

Zero-padding and the feature map sizes for convolutional layers

Calculating feature map dimensions

Batch normalization of input images

Building and training a CNN for image classification

Changing model hyperparameters

Convolutional Neural Networks

Two Kinds of Layers in CNNs

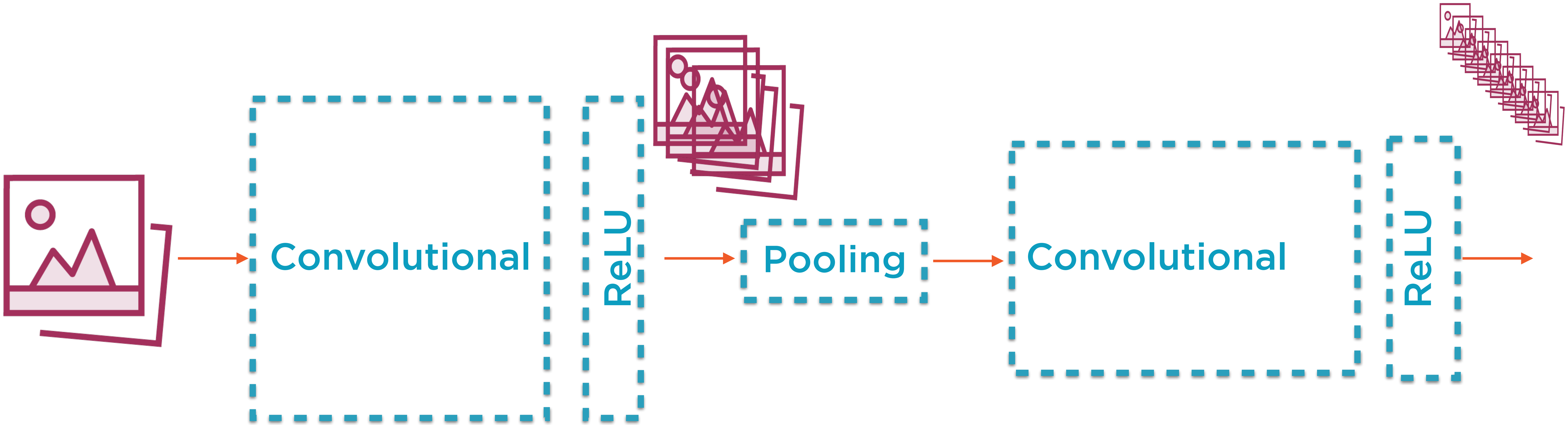
Convolution

Local receptive field

Pooling

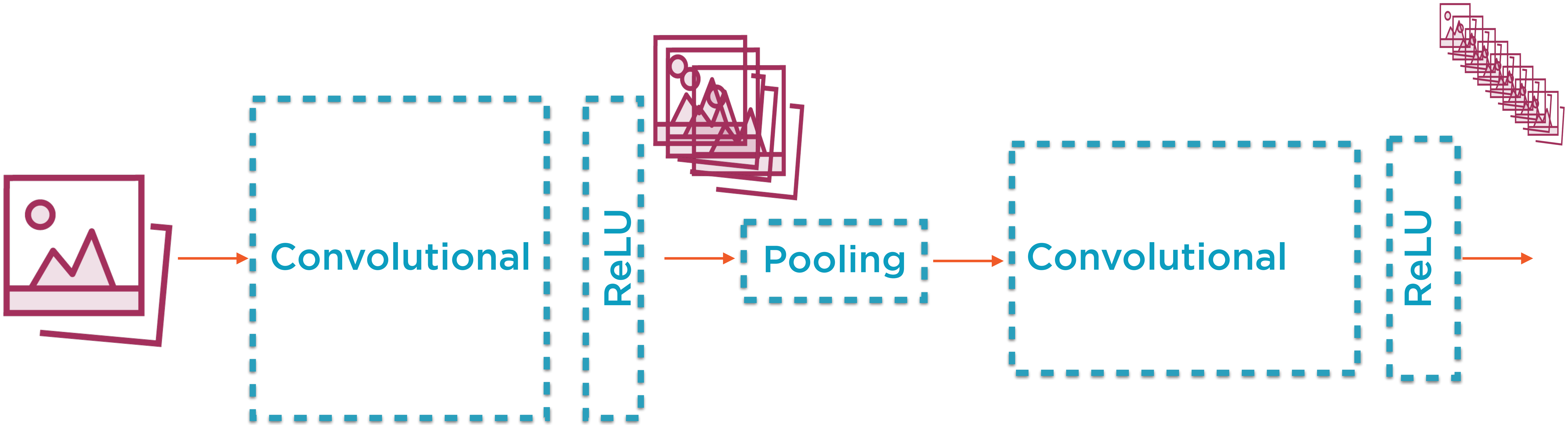
Subsampling of inputs

Typical CNN Architecture



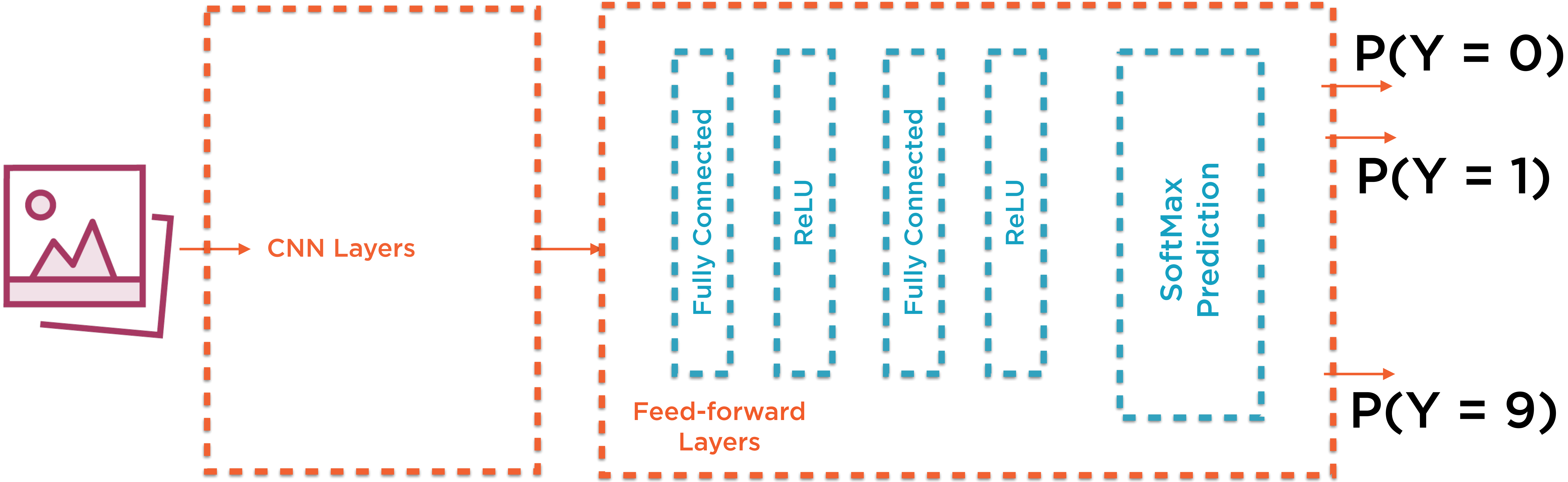
Alternating convolutional and pooling layers

Typical CNN Architecture



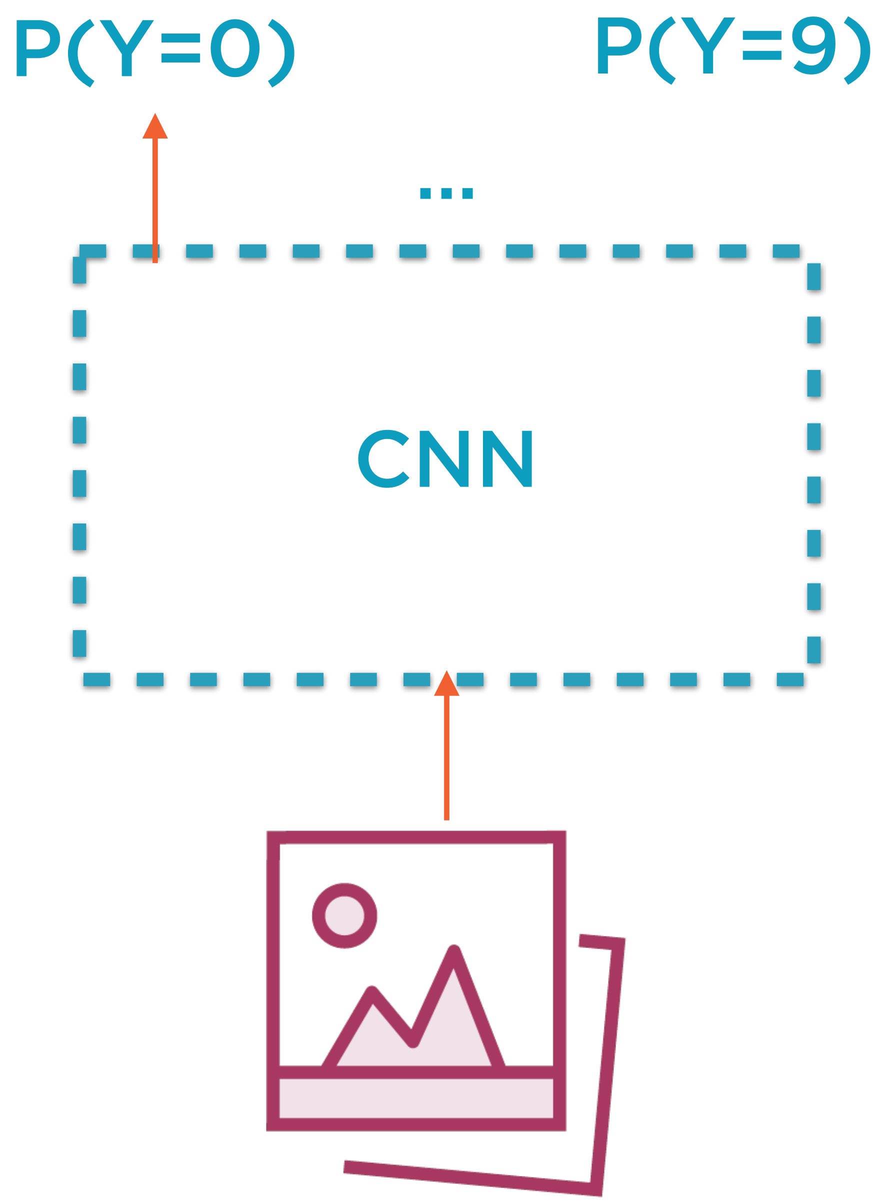
This entire set of layers is then fed into a regular, feed-forward NN

Typical CNN Architecture



This is the output layer, emitting probabilities

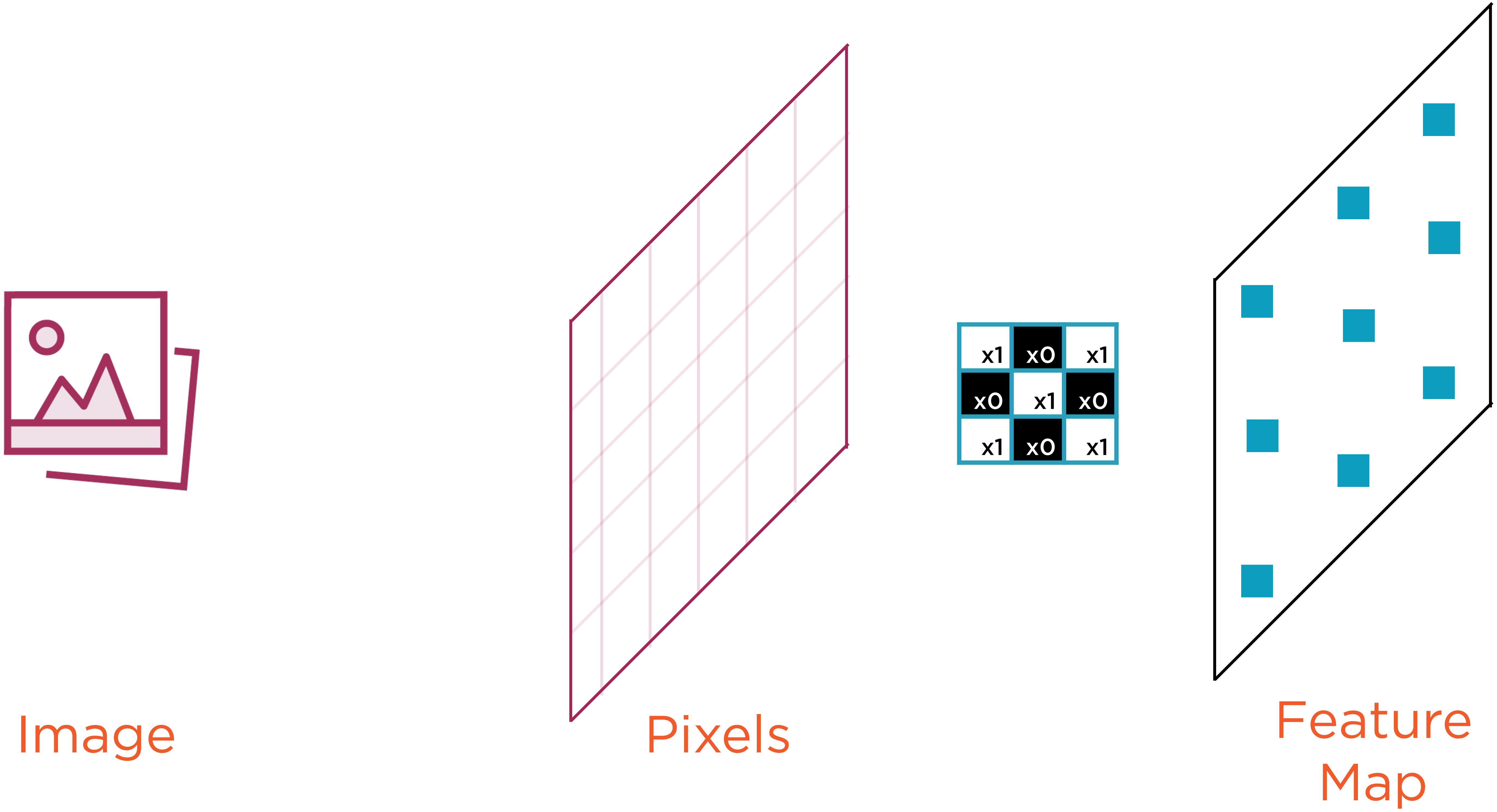
Typical CNN Architecture



Input is an image

Outputs are probabilities

Feature Maps

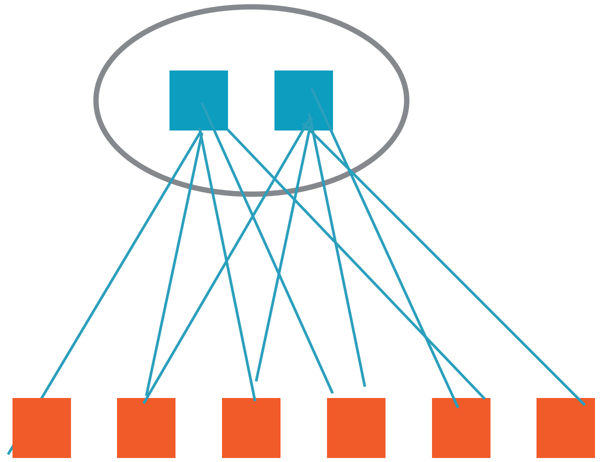


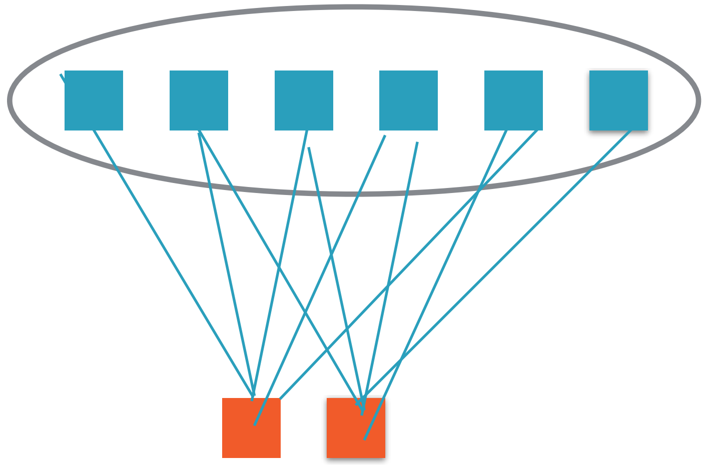
Zero-padding, Stride Size

Narrow vs. Wide Convolution

Input matrix i.e. image

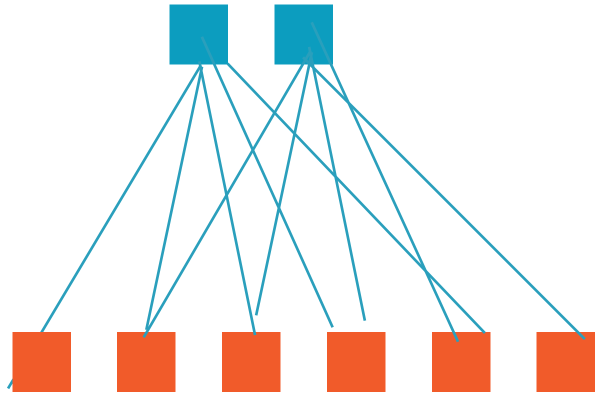
Narrow vs. Wide Convolution





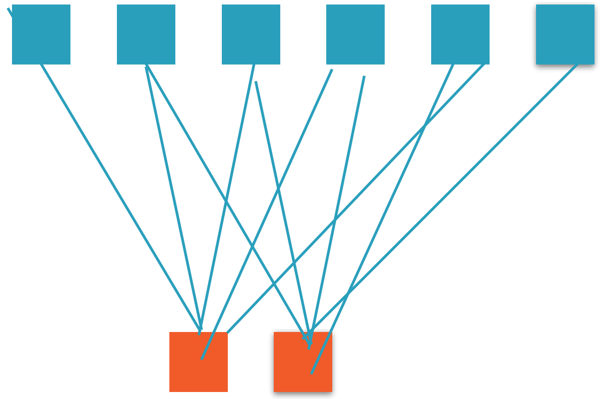
Convolution result

Narrow vs. Wide Convolution



Narrow Convolution

Little zero padding; output narrower than input



Wide Convolution

Lots of zero padding; output wider than input

Without Zero Padding

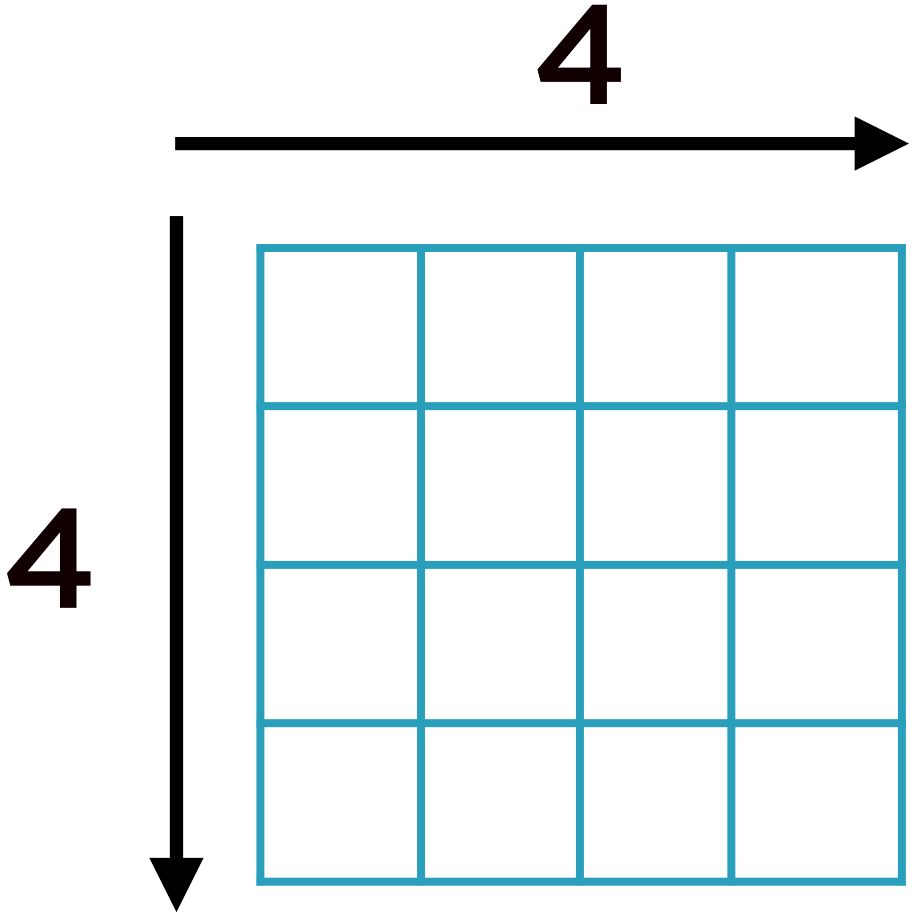
6

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

Matrix

x1	x0	x1
x0	x1	x0
x1	x0	x1



Convolution Result

Zero Padding

10

10

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

Matrix

x1	x0	x1
x0	x1	x0
x1	x0	x1

8

8

Convolution Result

Zero Padding

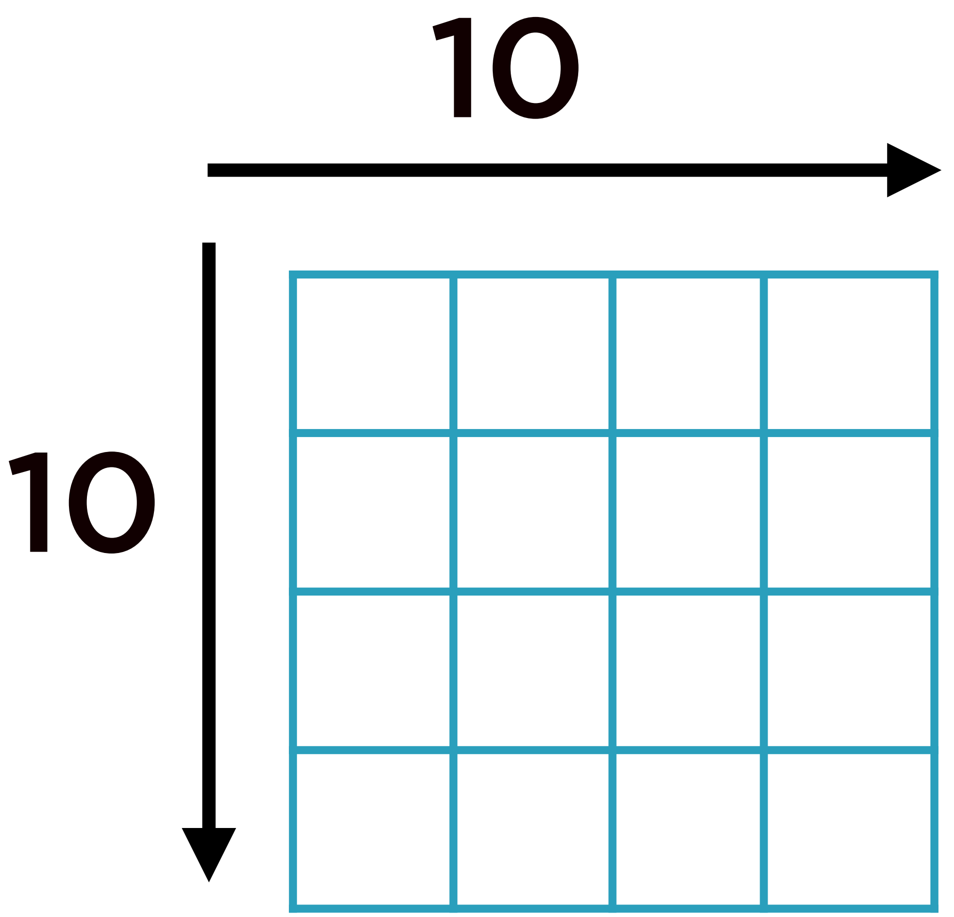
12

12

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

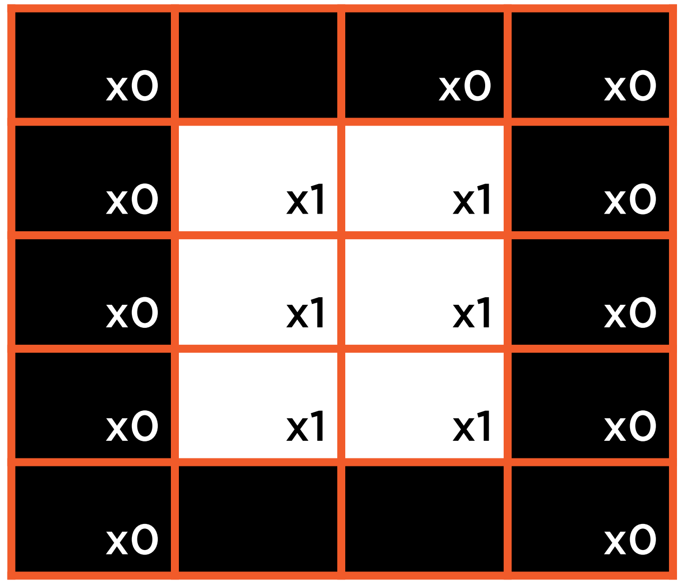
Matrix

x1	x0	x1
x0	x1	x0
x1	x0	x1



Convolution Result

Zero Padding

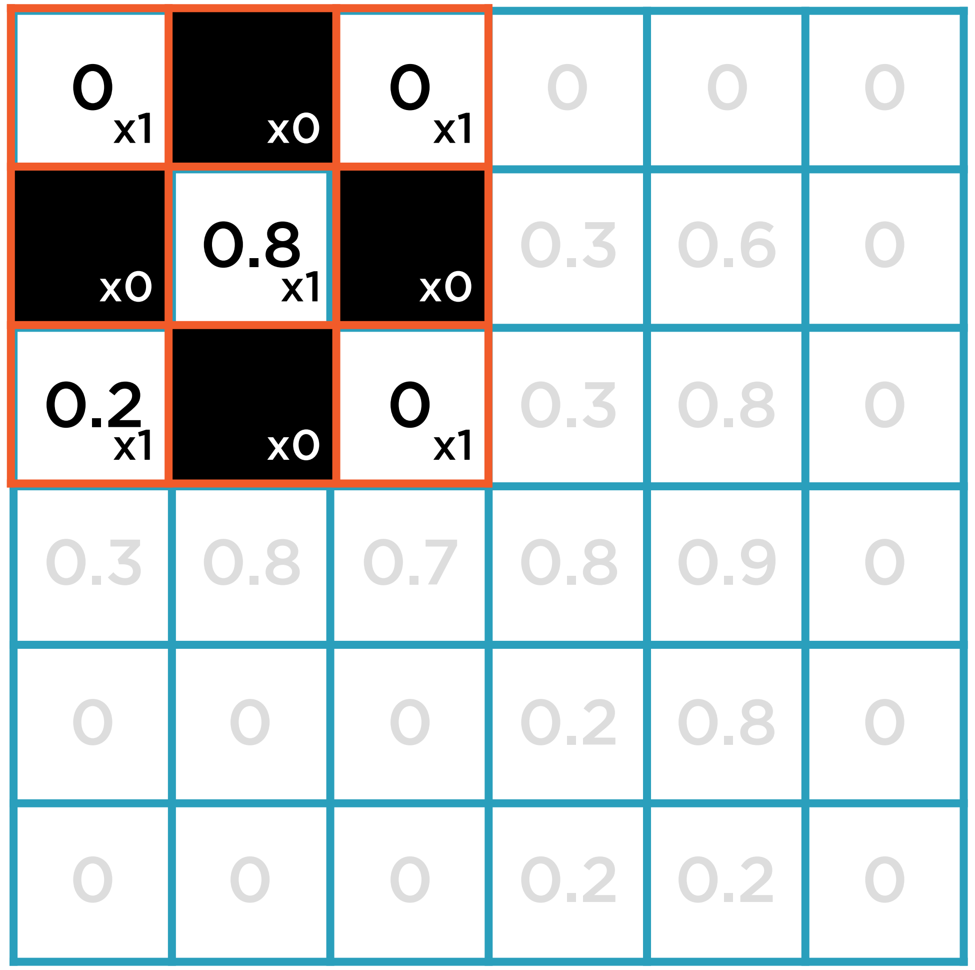


With zero-padding, every element of matrix will be passed into filter

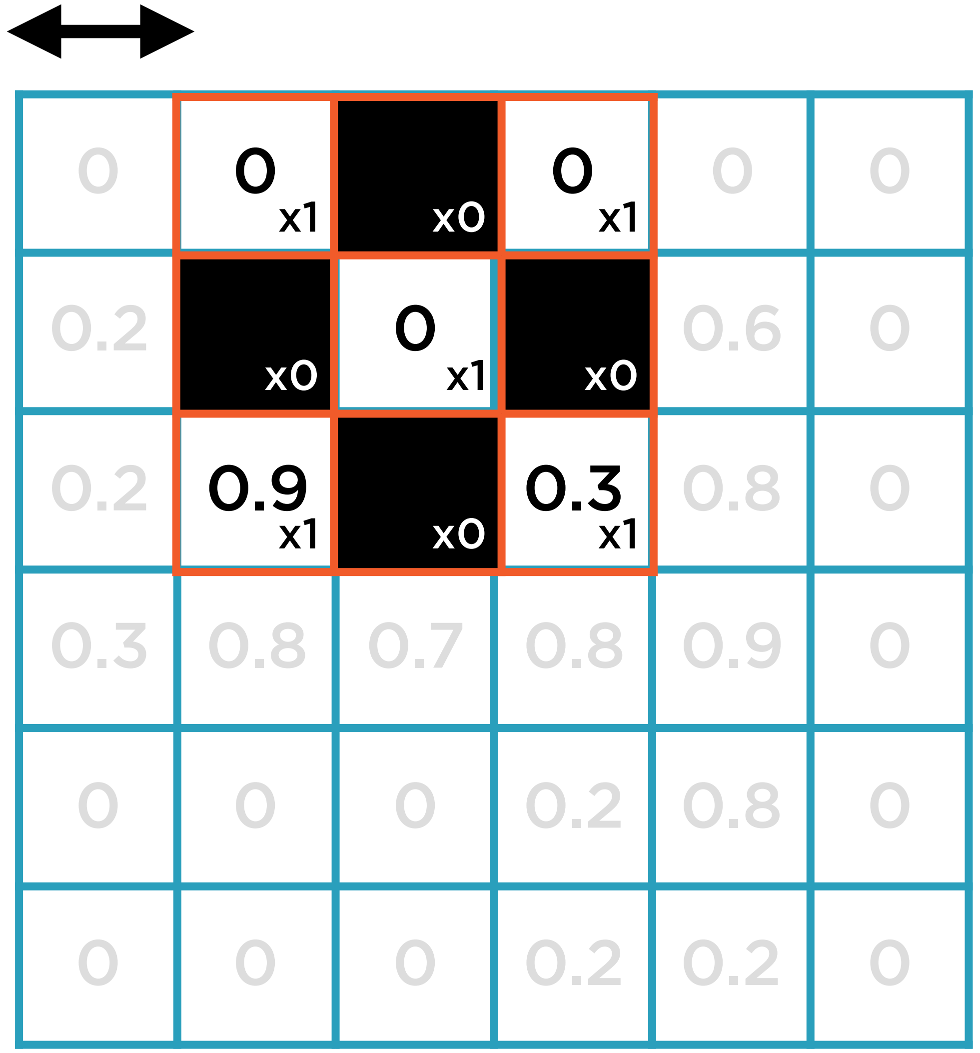
Can decide number of zero columns to pad with

Use to get output larger than input

Stride Size

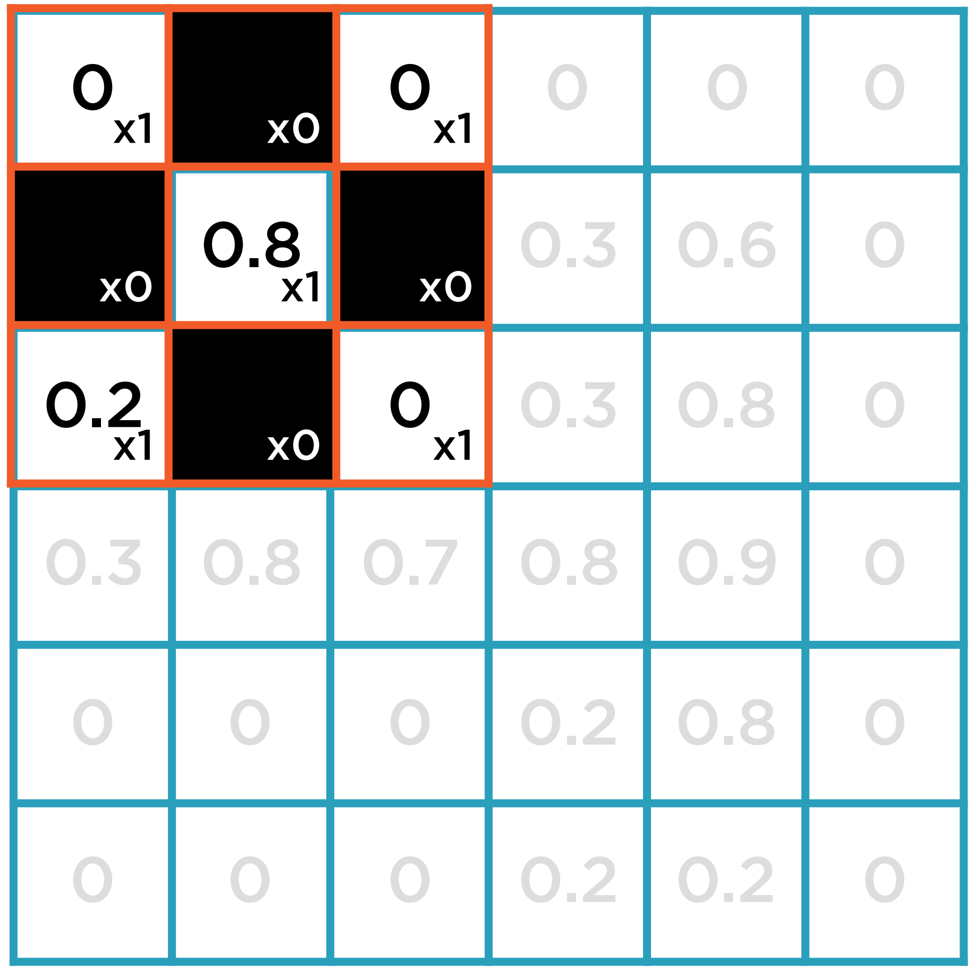

0 x1
x0
0 x1
0 0 0
x0
0.8 x1
x0
0.3 0.6 0
0.2 x1
x0
0 x1
0.3 0.8 0
0.3 0.8 0.7 0.8 0.9 0
0 0 0 0.2 0.8 0
0 0 0 0.2 0.2 0

Stride Size

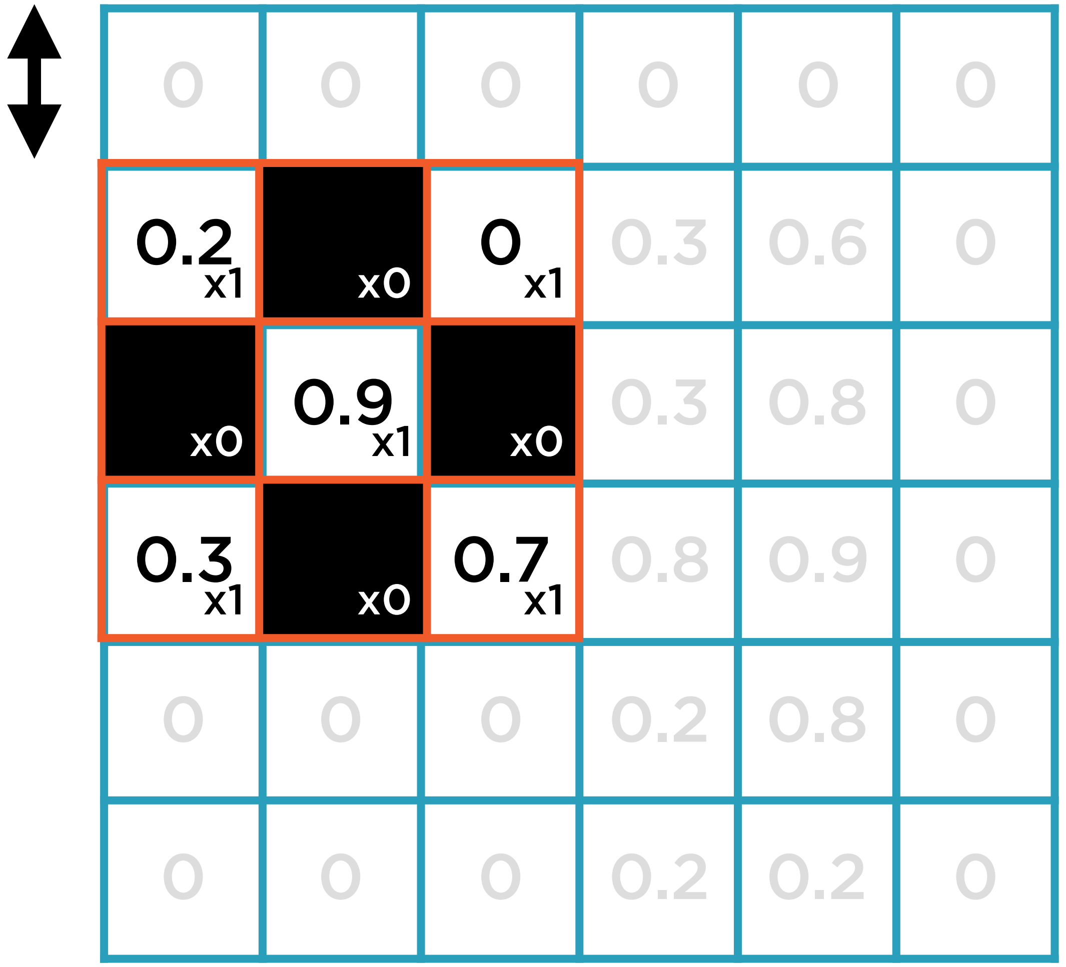


Horizontal stride of 1

Stride Size

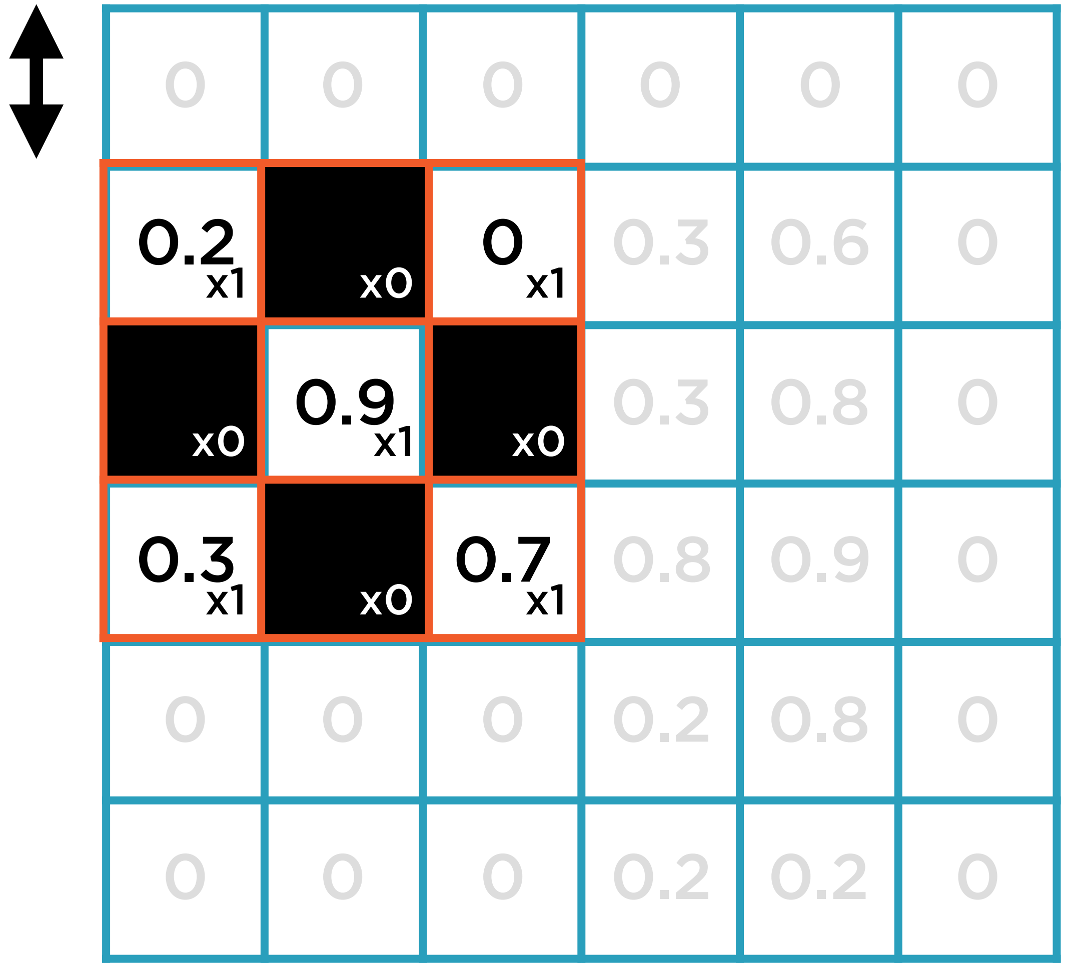

0 x1
x0
0 x1
0
0
0
x0
0.8 x1
x0
0.3
0.6
0
0.2 x1
x0
0 x1
0.3
0.8
0
0.3
0.8
0.7
0.8
0.9
0
0
0
0
0.2
0.8
0
0
0
0
0.2
0.2
0

Stride Size



Vertical stride of 1

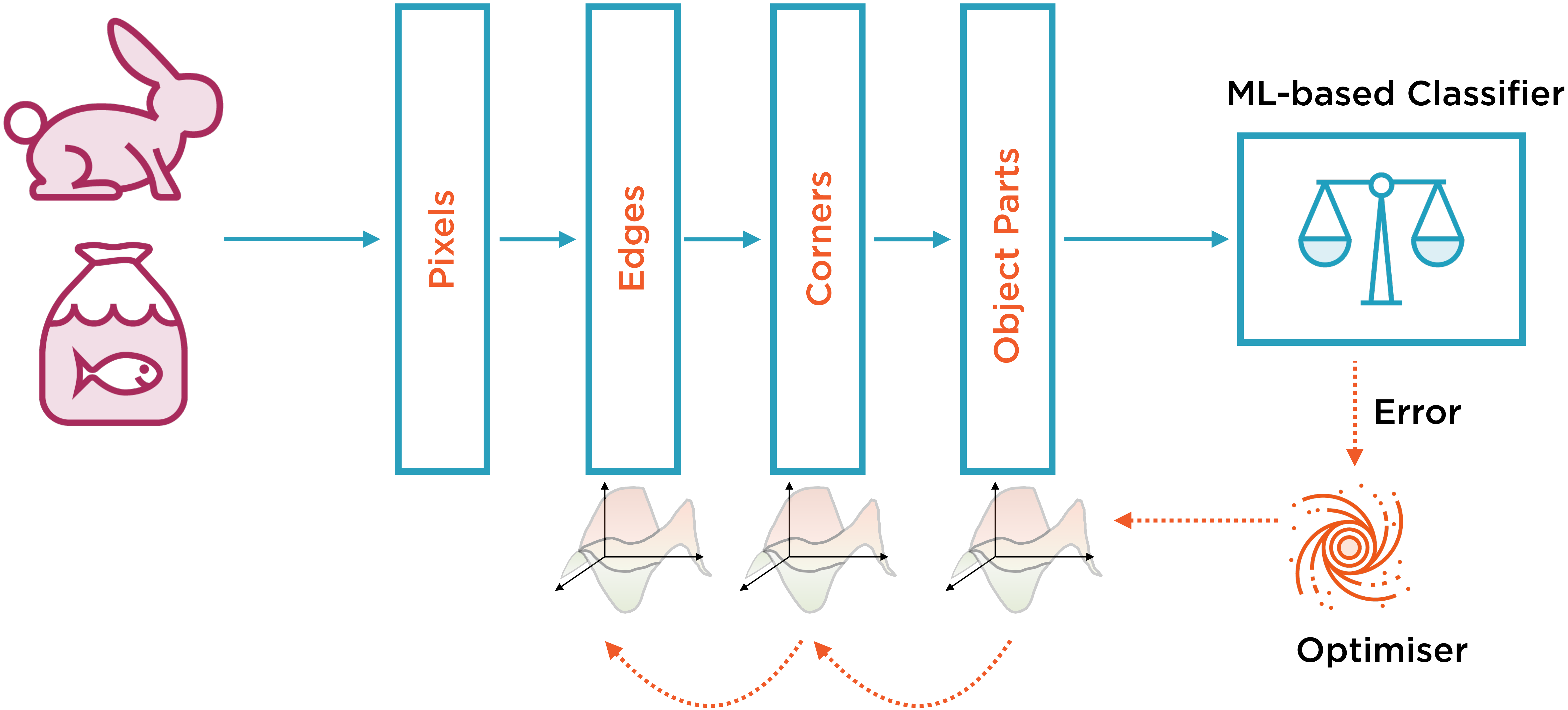
Stride Size



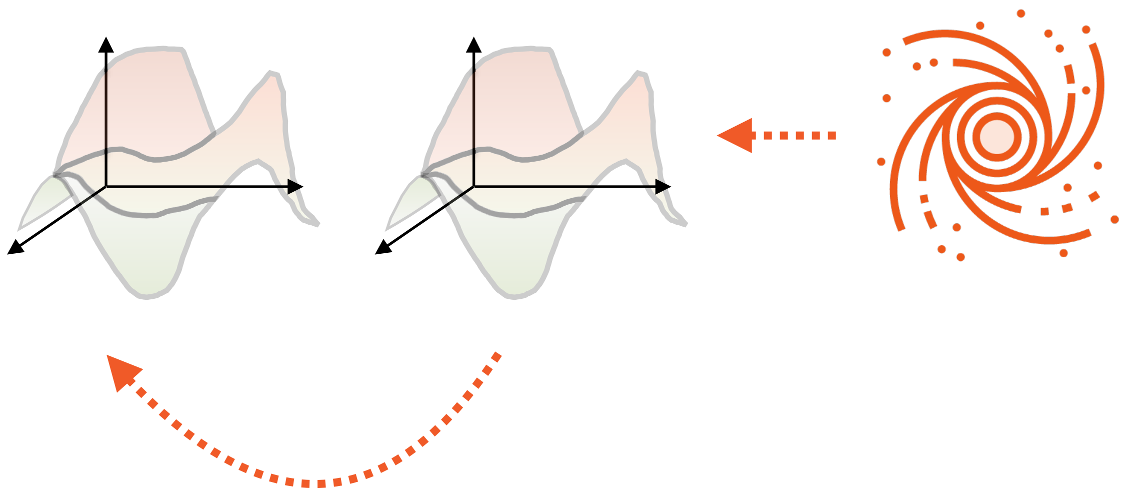
Stride size is an important hyperparameter in CNNs

Batch Normalization

Training via Back Propagation



Vanishing and Exploding Gradients

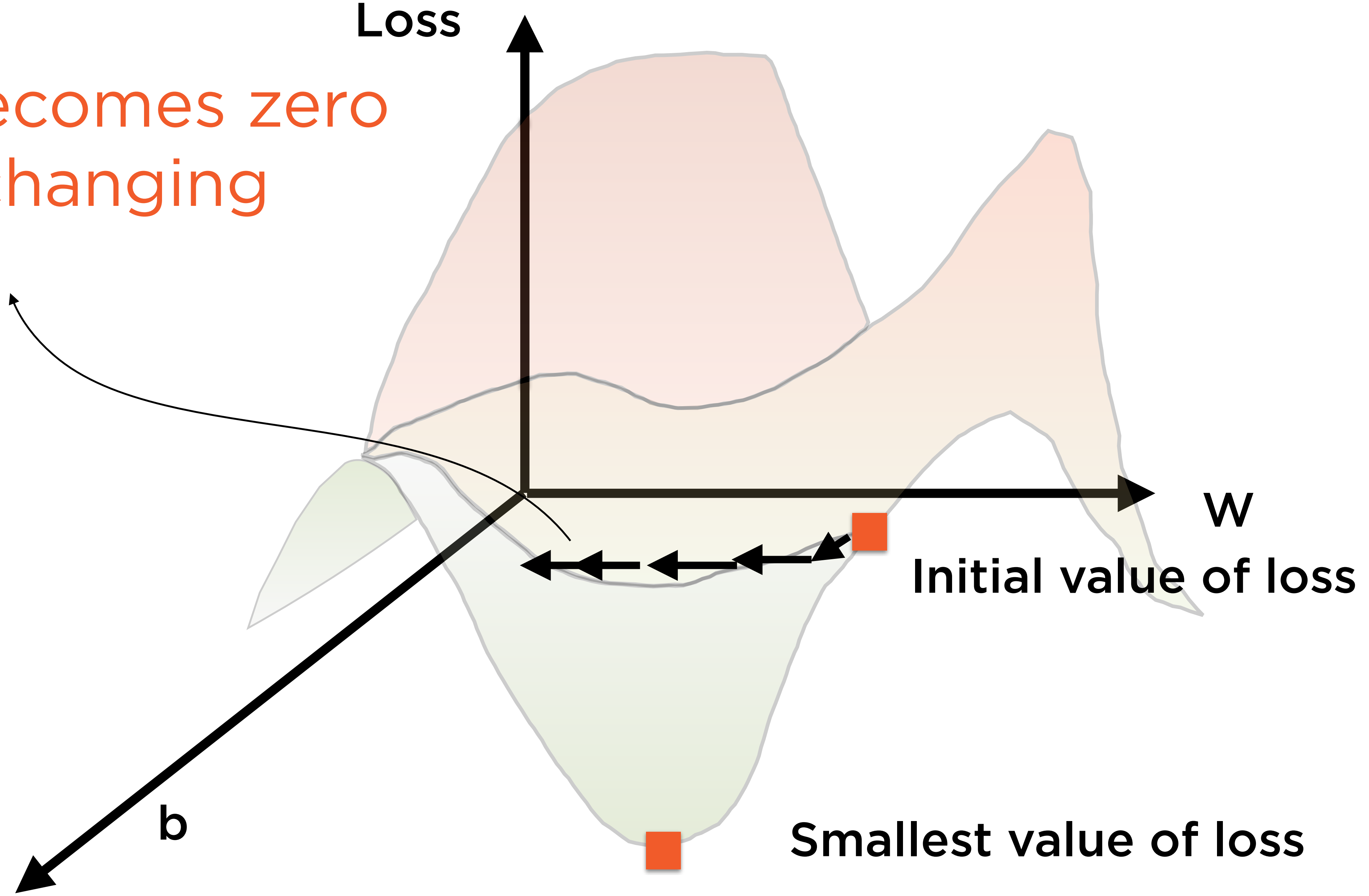


Back propagation fails if

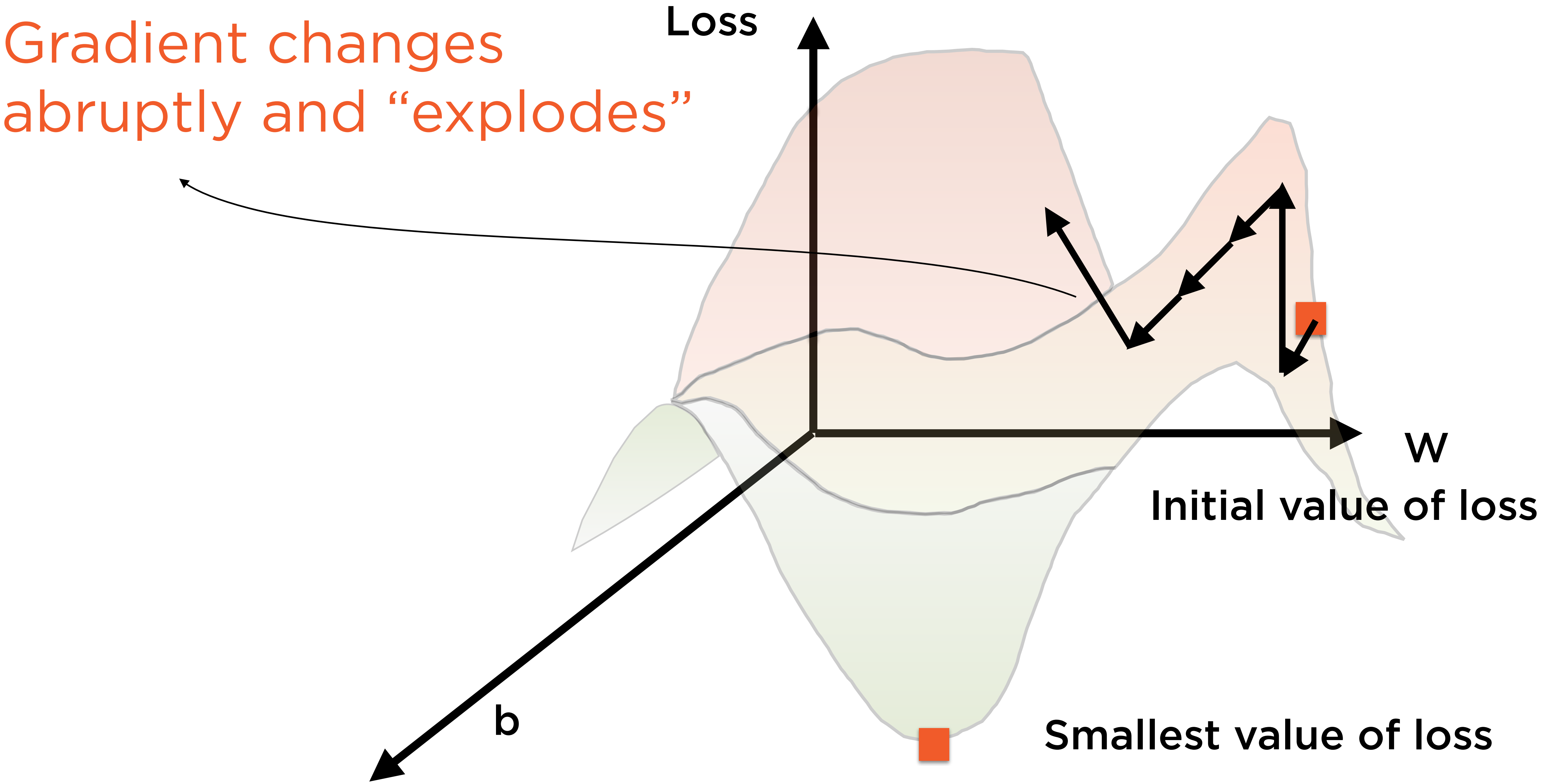
- gradients are vanishing
- gradients are exploding

Vanishing Gradient Problem

Gradient becomes zero and stops changing



Exploding Gradient Problem



Coping with Vanishing/Exploding Gradients

Proper initialization

Non-saturating activation function

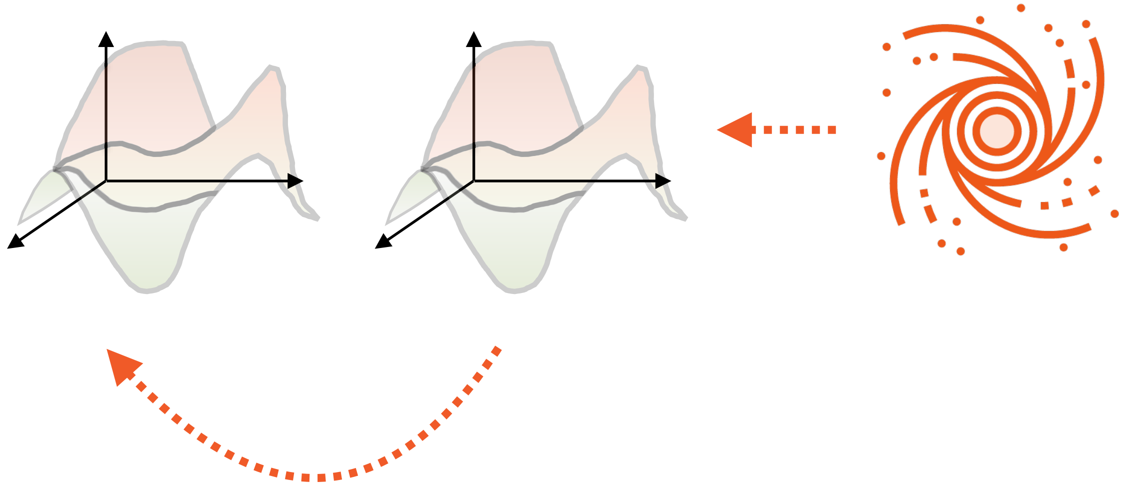
Batch normalization

Gradient clipping

Coping with Vanishing/Exploding Gradients

Proper initialisation	Non-saturating activation function
Batch normalization	Gradient clipping

Batch Normalization

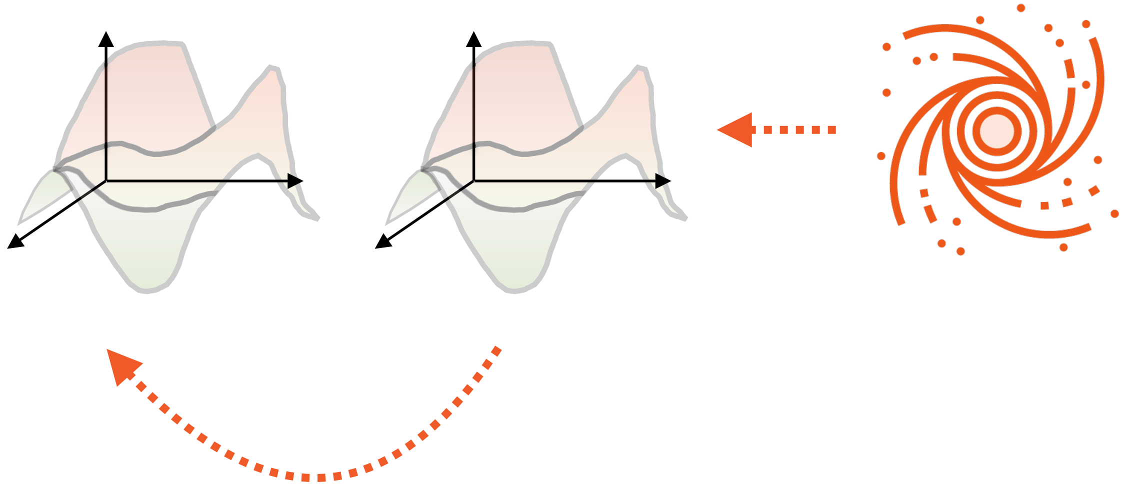


Just before applying activation function

First, “normalize” inputs

Second, “scale and shift” inputs

Batch Normalization



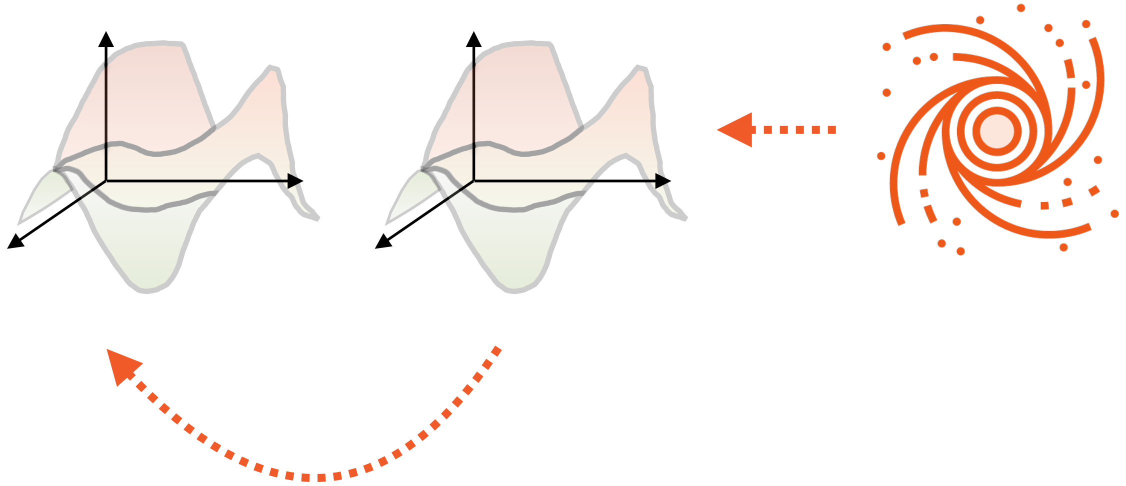
“Normalize” inputs

- subtract mean
- divide by standard deviation

“Scale and shift” inputs

- scale = multiply by constant
- shift = add constant

Batch Normalization



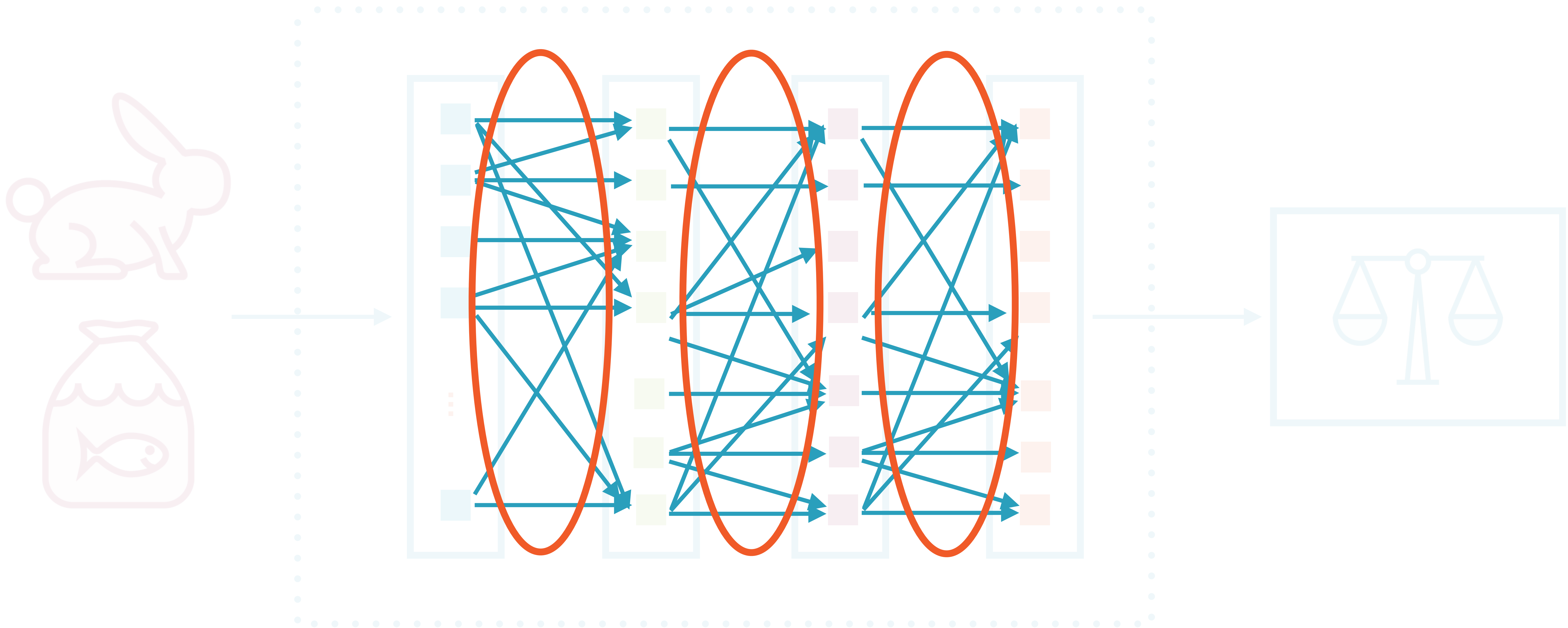
Supported in PyTorch

Many other benefits

- allows much larger learn rate
- reduces overfitting
- speeds convergence of training

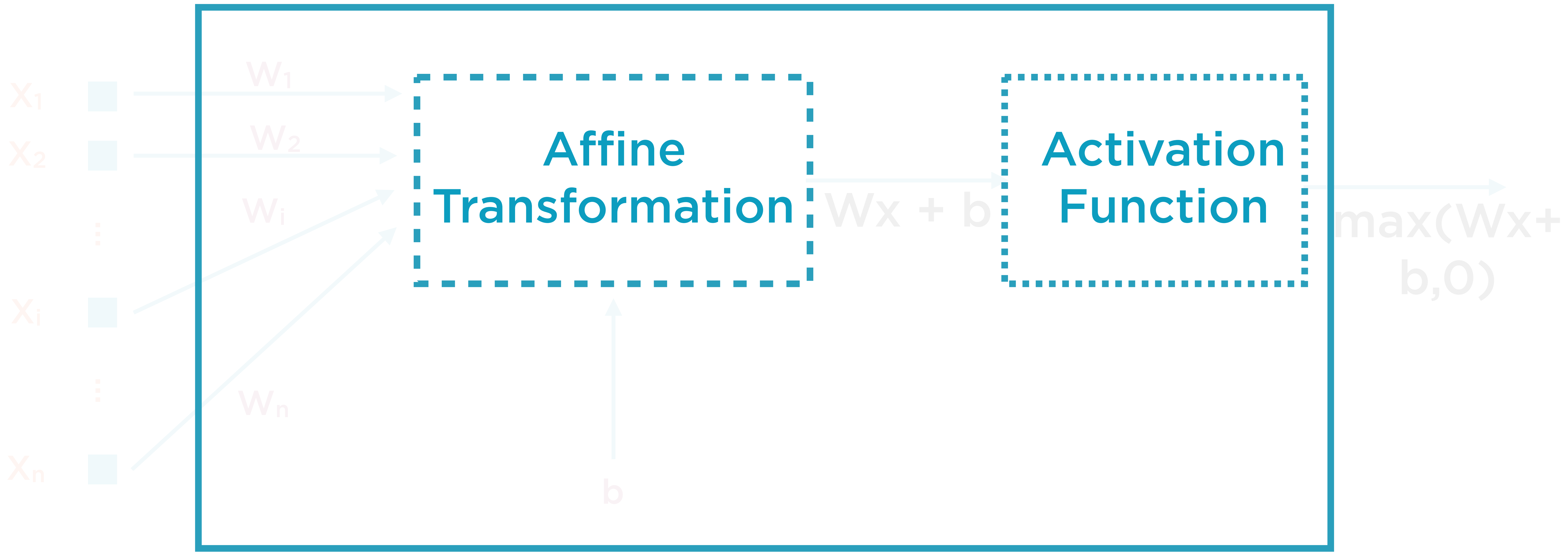
Choice of Activation Function

A Neural Network



Once a neural network is trained all edges have weights which help it make predictions

Operation of a Single Neuron

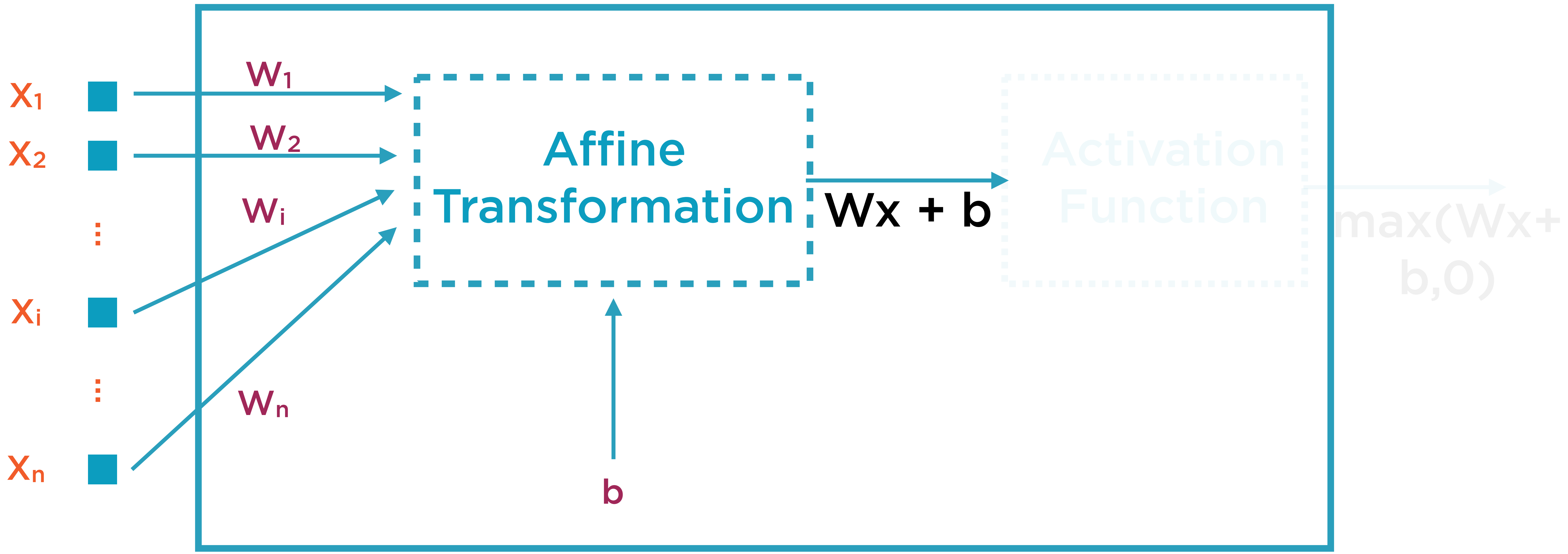


Each neuron only applies two simple functions to its inputs

Operation of a Single Neuron

The affine transformation alone can only learn linear relationships between the inputs and the output

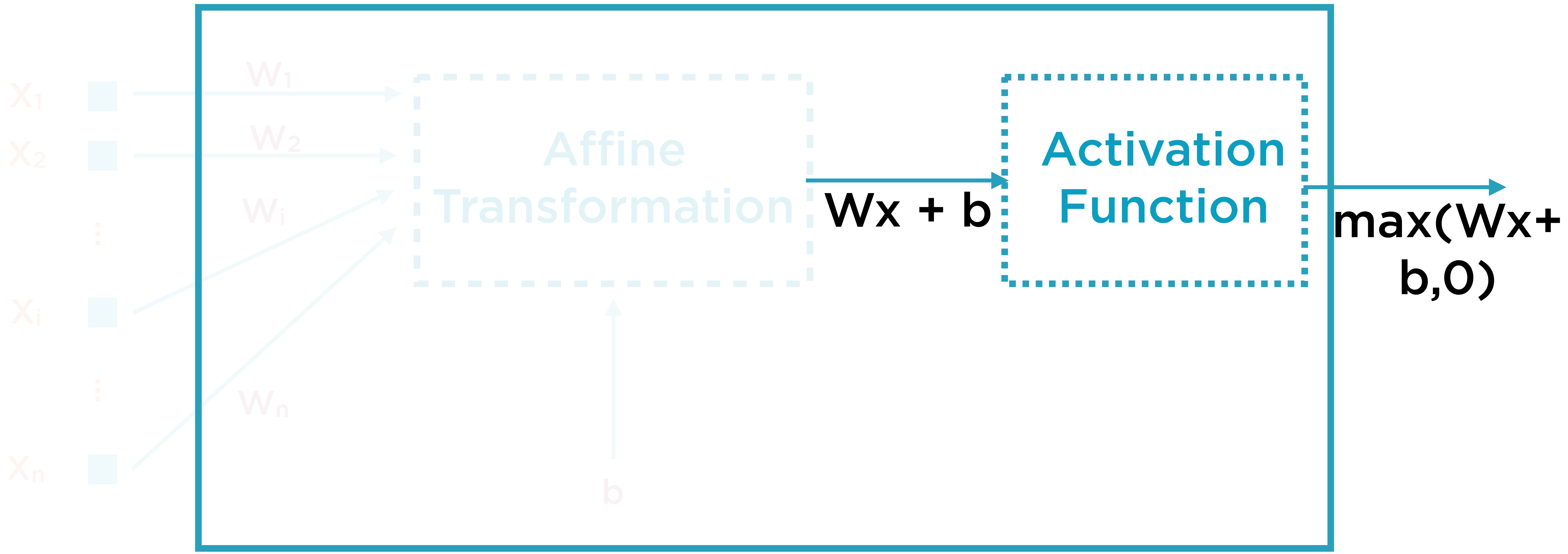
Operation of a Single Neuron



The affine transformation is just a weighted sum with a bias added: $W_1x_1 + W_2x_2 + \ldots + W_nx_n + b$

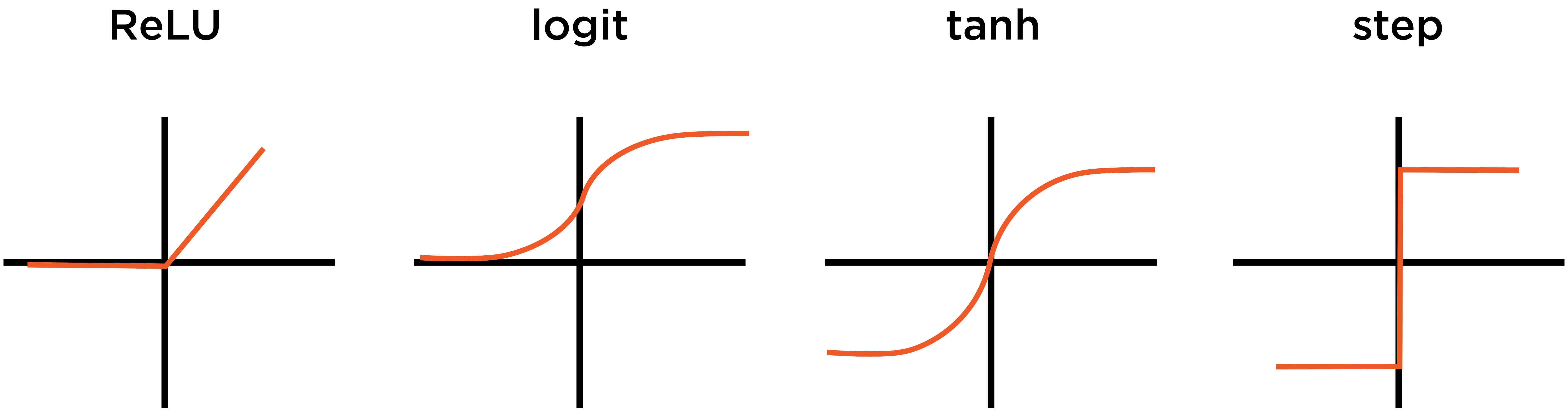
The **weights** and **biases** of individual neurons are determined during the **training** process

Operation of a Single Neuron



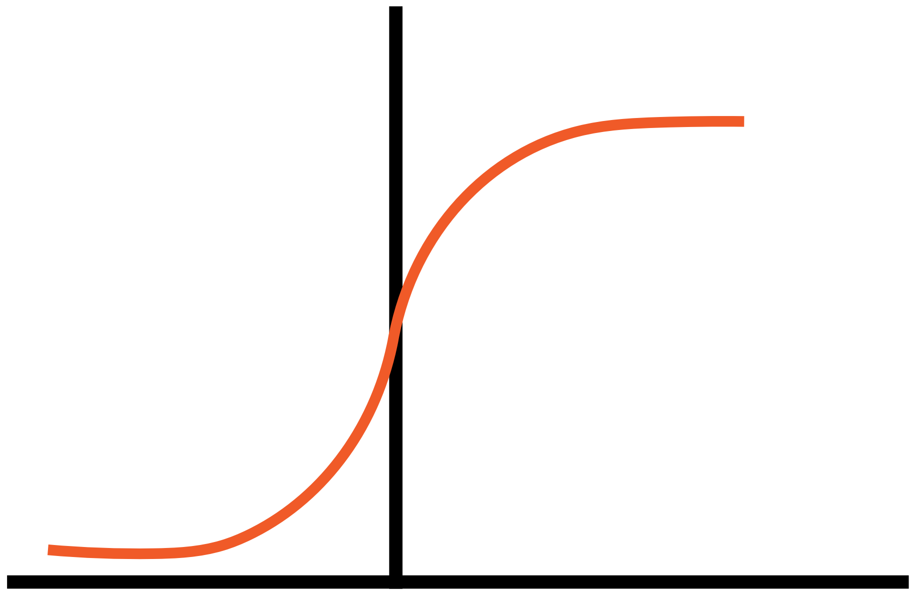
The **combination** of the affine transformation and the activation function can **learn any arbitrary relationship**

Activation Function



Various choices of activation functions exist and drive the design of your neural network

Importance of Activation



The choice of activation function is crucial in determining performance

Feature Map Size Calculations

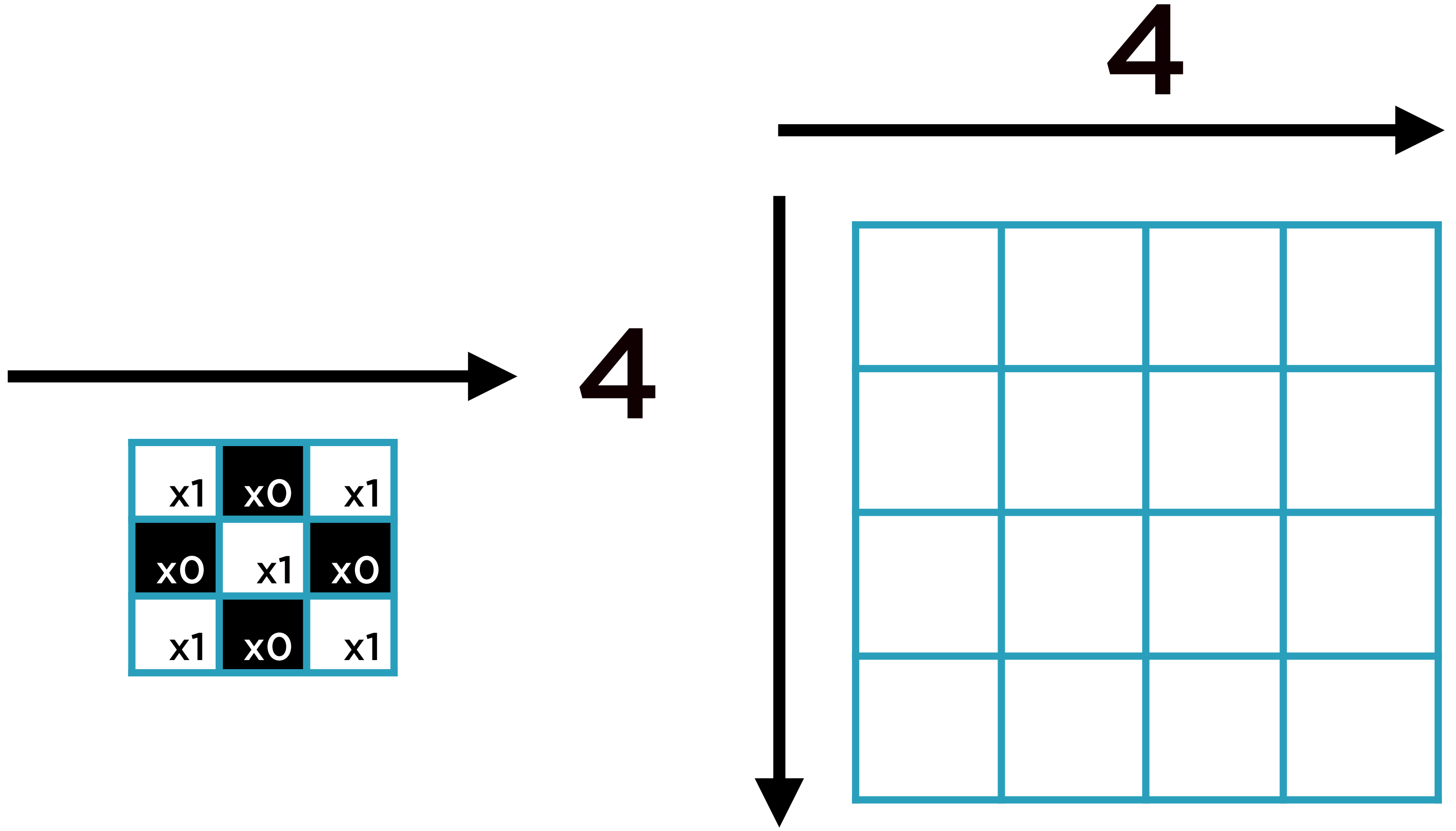
Without Zero Padding

6

6

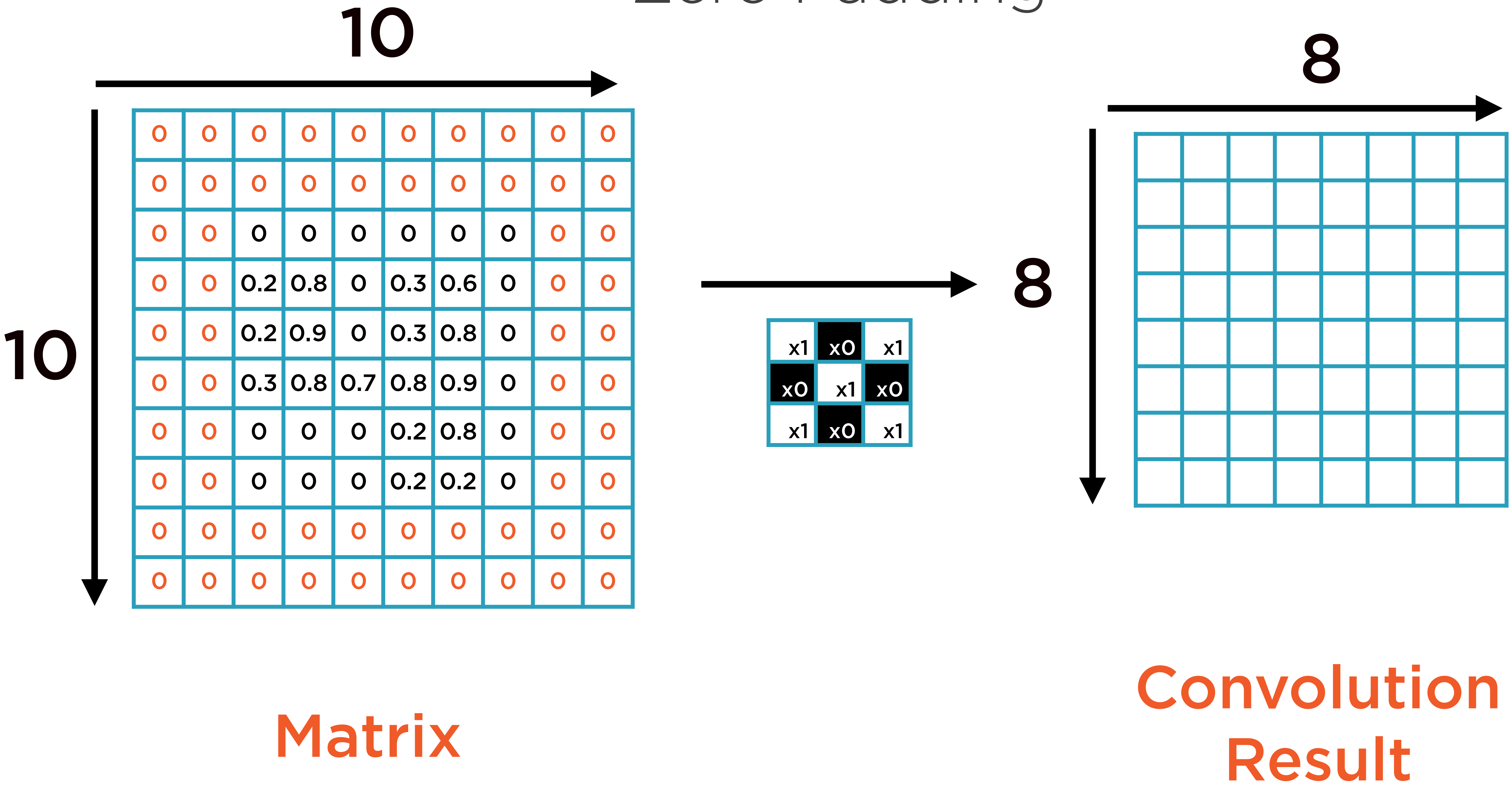
0	0	0	0	0	0
0.2	0.8	0	0.3	0.6	0
0.2	0.9	0	0.3	0.8	0
0.3	0.8	0.7	0.8	0.9	0
0	0	0	0.2	0.8	0
0	0	0	0.2	0.2	0

Matrix



Convolution Result

Zero Padding



$$O = \frac{W-K+2P}{S} + 1$$

Formula for dimension calculations

Handy in getting dimensions of CNN layers right

$$O = \frac{W-K+2P}{S} + 1$$

O = Output dimension

Height/width of output

$$O = \frac{W-K+2P}{S} + 1$$

W = Input dimension

Height/width of input image

$$O = \frac{W-K+2P}{S} + 1$$

K = Kernel size

Height/width of kernel

$$O = \frac{W-K+2P}{S} + 1$$

P = Padding (if any)

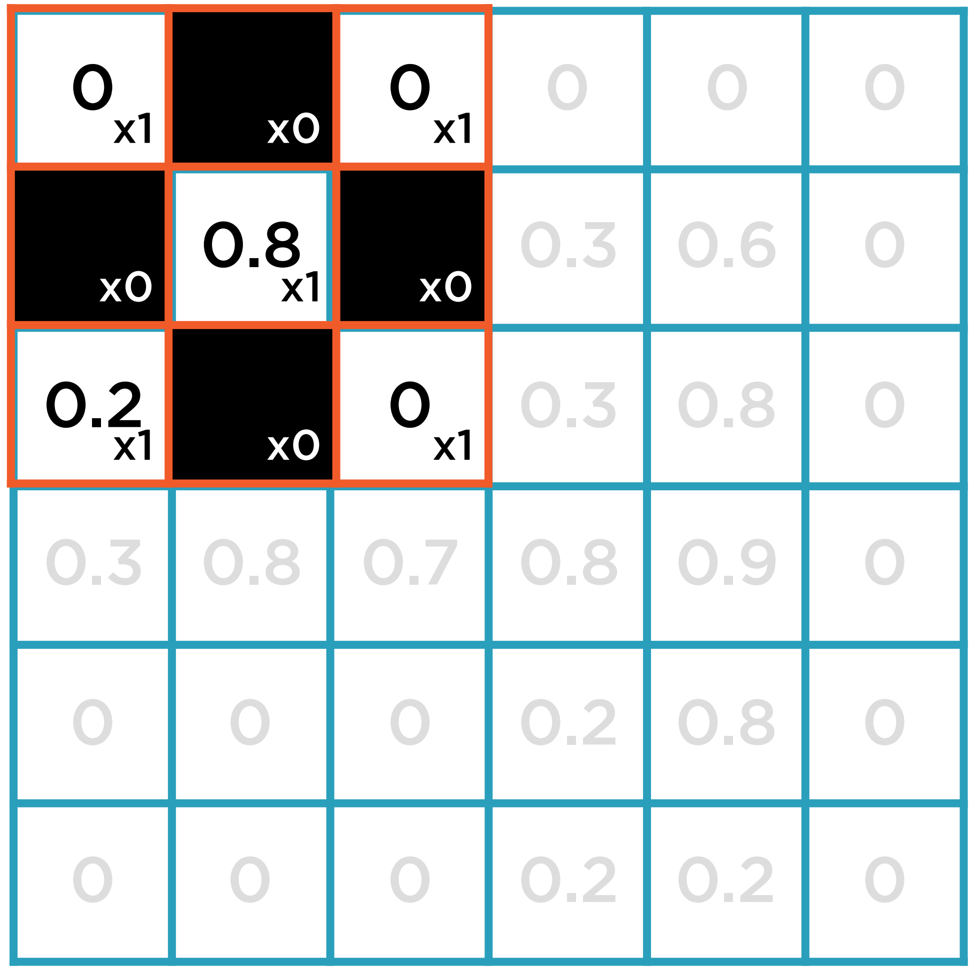
Maybe zero

$$O = \frac{W-K+2P}{S} + 1$$

S = Stride

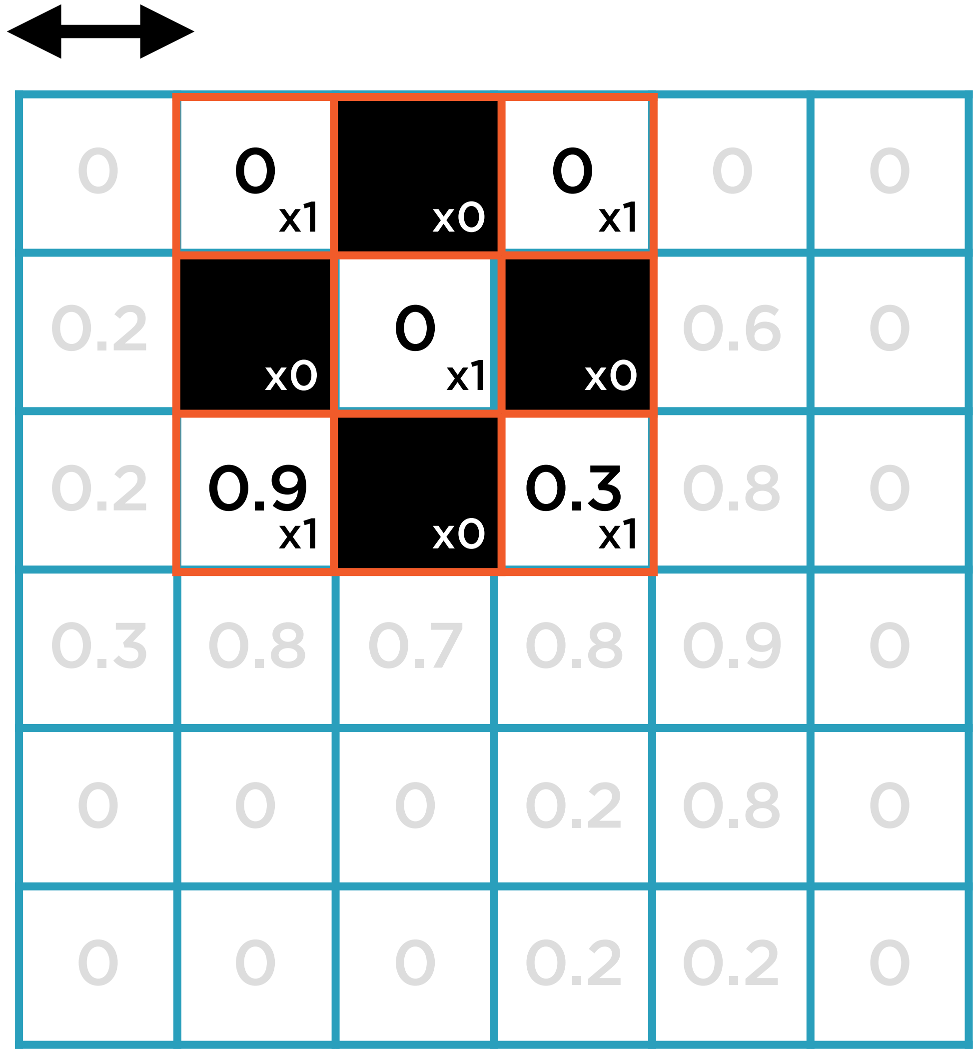
How far the kernel advances in each step

Stride Size



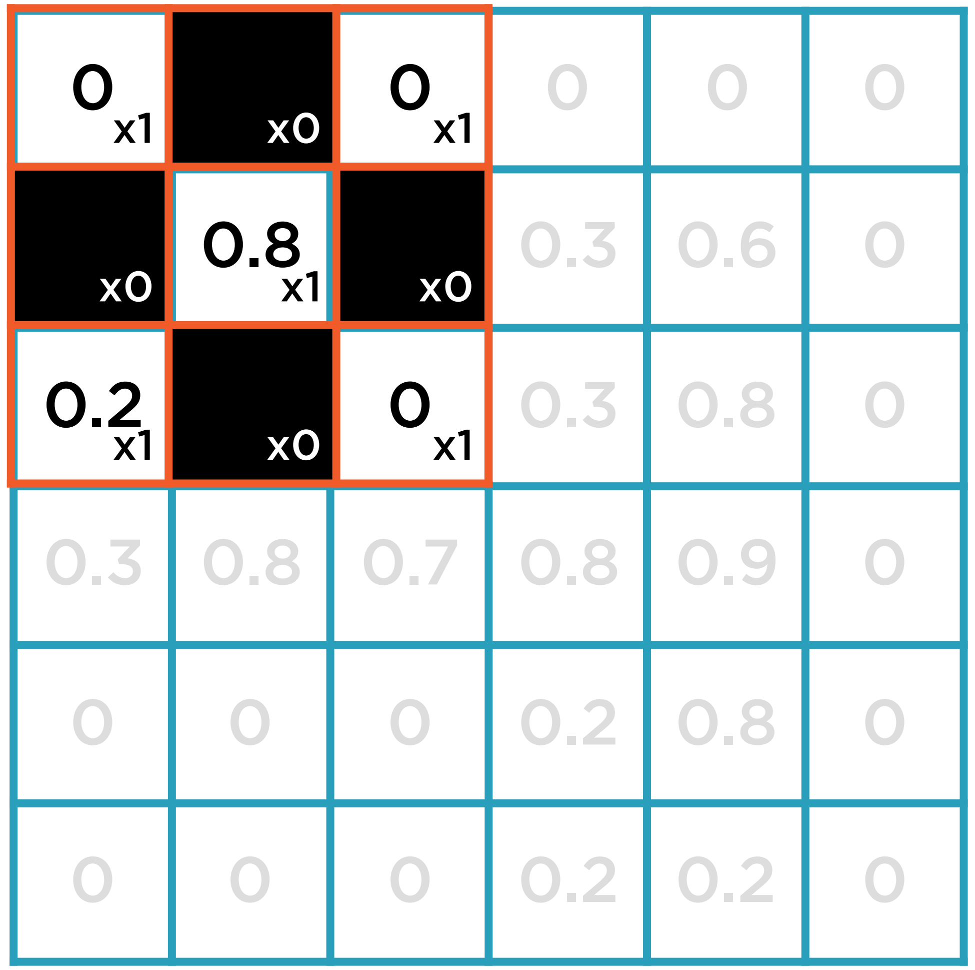
0 x1 | x0 | 0 x1 | 0 | 0 | 0
x0 | 0.8 x1 | x0 | 0.3 | 0.6 | 0
0.2 x1 | x0 | 0 x1 | 0.3 | 0.8 | 0
0.3 | 0.8 | 0.7 | 0.8 | 0.9 | 0
0 | 0 | 0 | 0.2 | 0.8 | 0
0 | 0 | 0 | 0.2 | 0.2 | 0

Stride Size



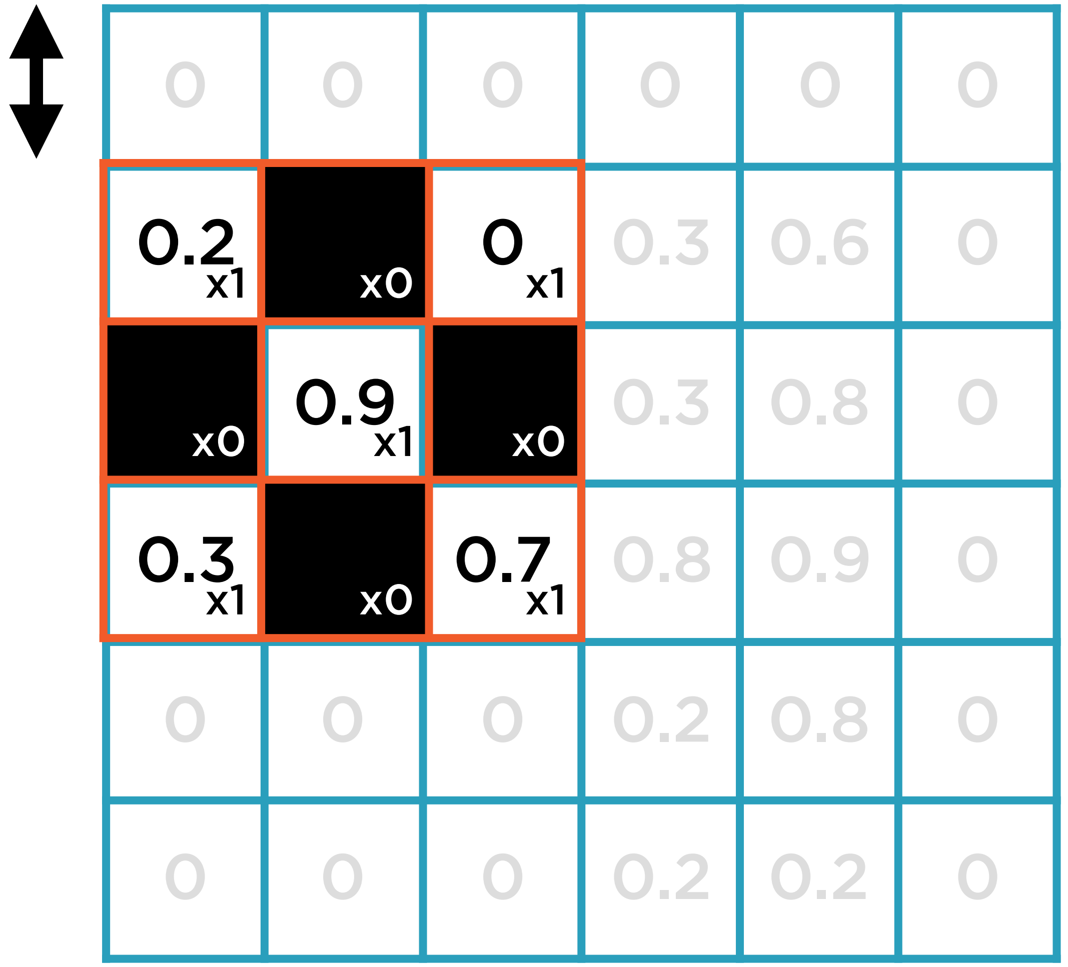
Horizontal stride of 1

Stride Size



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

Stride Size



Vertical stride of 1

$$O = \frac{W-K+2P}{S} + 1$$

Formula for dimension calculations

Handy in getting dimensions of CNN layers right

Demo

Image classification using convolutional neural networks (CNNs)

Hyperparameter tuning

Summary

Narrow and wide convolution

Zero-padding and the feature map sizes for convolutional layers

Calculating feature map dimensions

Batch normalization of input images

Building and training a CNN for image classification

Changing model hyperparameters