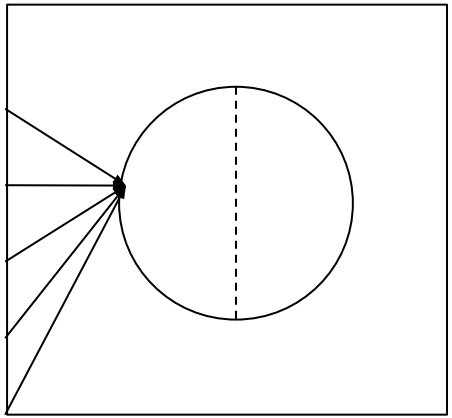
Copyright Notice

deeplearning.ai

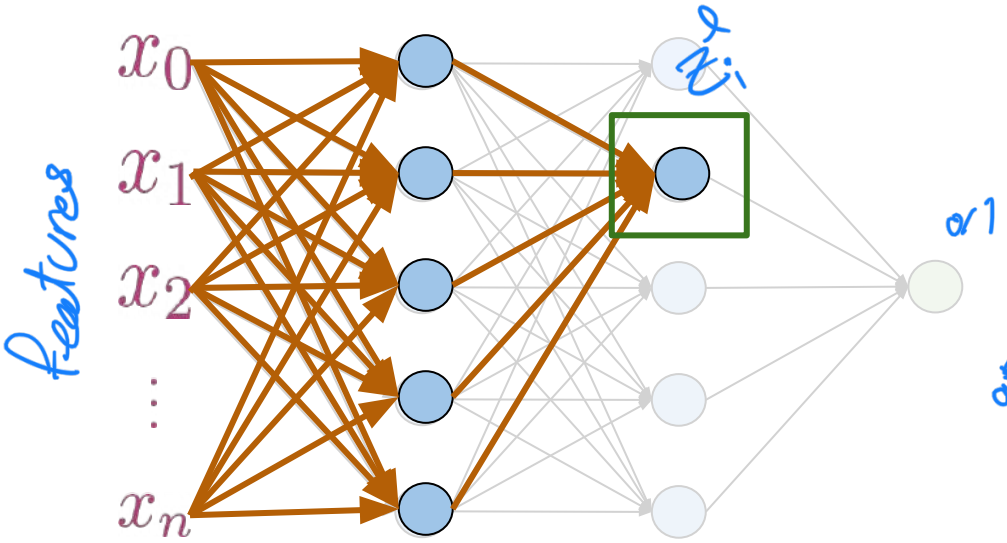
Activations (Basic Properties)

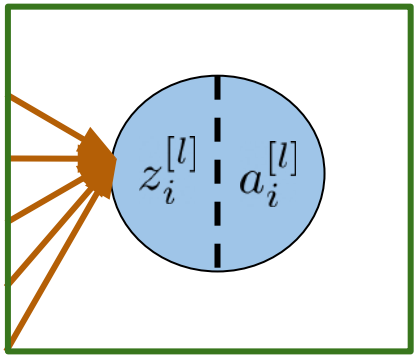
Outline

- What are activations
- Reasoning behind non-linear differential activations



Activations





output layer

prev layer

$$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]}$$

$$a_i^{[l]} = g^{[l]}(z_i^{[l]})$$

actv function

Differentiable non-linear function

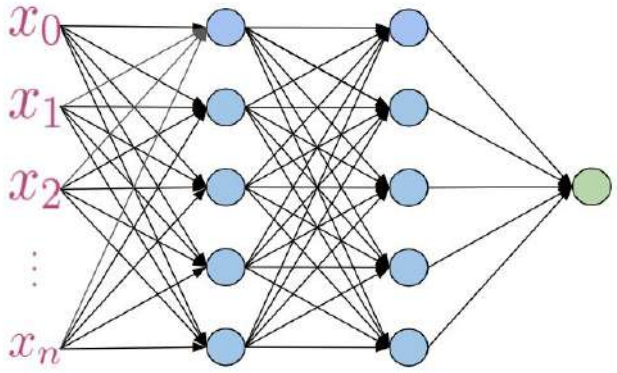
Activations

$$a_i^{[l]} = g^{[l]}(z_i^{[l]})$$

Differentiable non-linear function

1. Differentiable for backpropagation

2. Non-linear to compute complex features, **if not**:



$\equiv$

$$WX + b$$

Linear regression

if linear acts func

Summary

- Activation functions are non-linear and differentiable
- Differentiable for backpropagation
- Non-linear to approximate complex functions

deeplearning.ai

Common Activation Functions

Outline

- Common activations and their structure
  - ReLU
  - Leaky ReLU
  - Sigmoid
  - Tanh

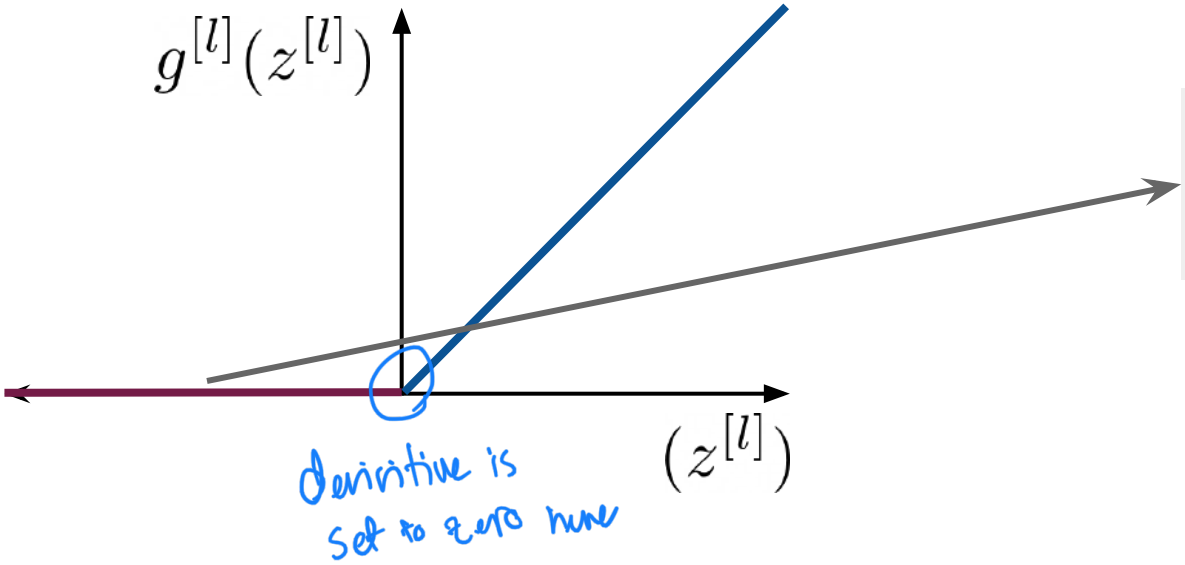
Activations: ReLU

No Neg Value

ReLU

$$g^{[l]}(z^{[l]}) = \max(0, z^{[l]})$$

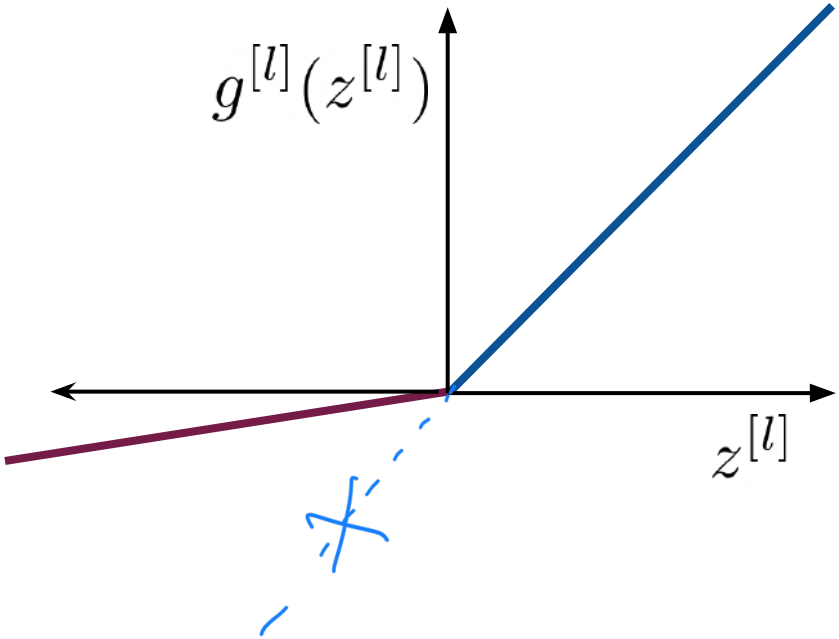
ReLU = Rectified Linear Unit



Dying ReLU problem

Stops learning because of zeros

Activations: Leaky ReLU



leak / slope

$$g^{[l]}(z^{[l]}) = max(az^{[l]}, z^{[l]})$$

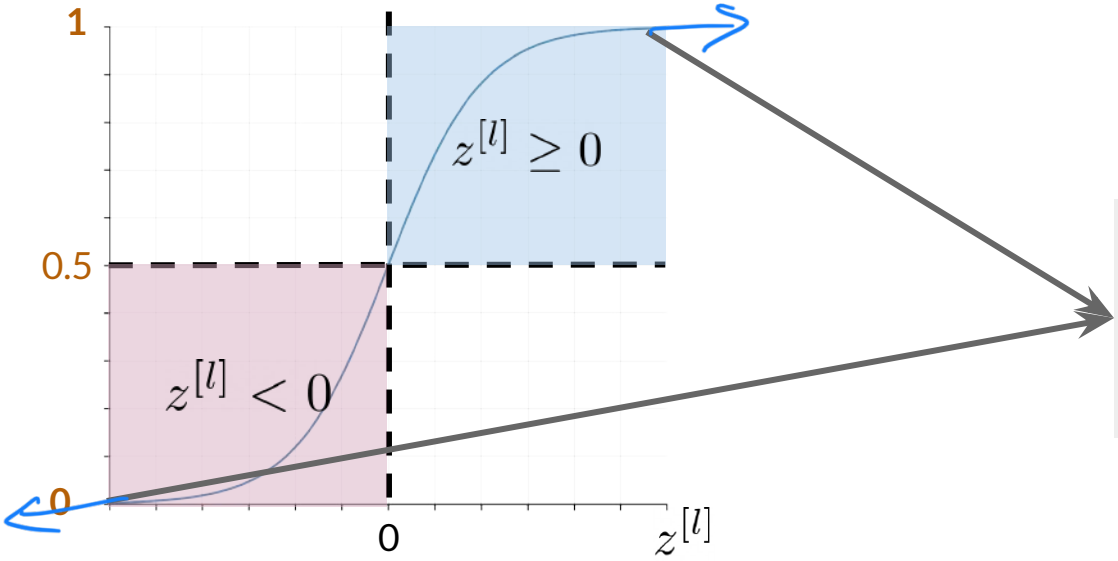
less than 1

like 0.1

Solves the dying ReLU problem

Activations: Sigmoid

$$g^{[l]}(z^{[l]}) = \frac{1}{1 + e^{-z^{[l]}}}$$



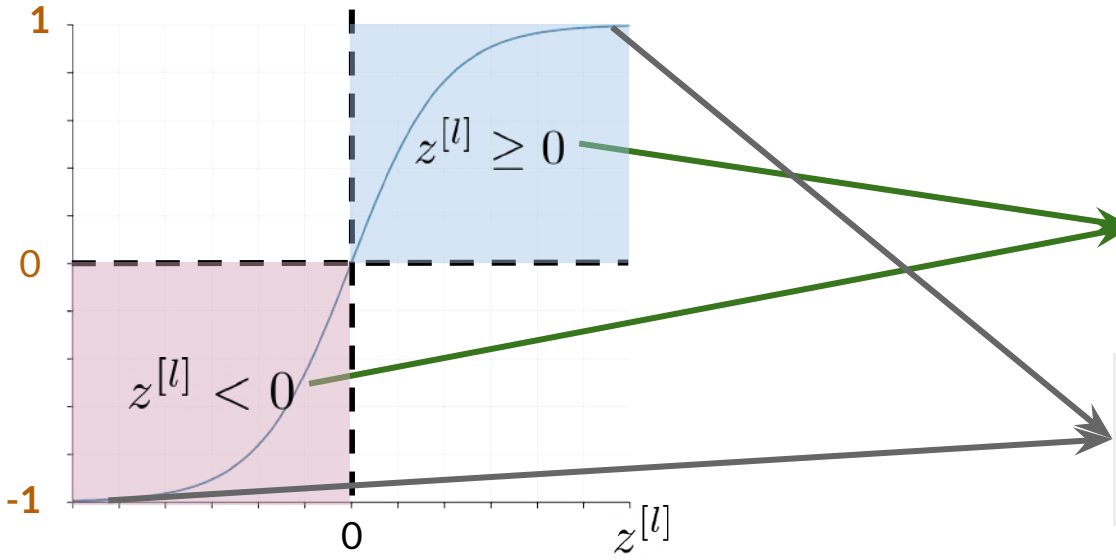
for binary classification prob of 0/1

Values between 0 and 1

Vanishing gradient and saturation problems

Activations: Tanh

$$g^{[l]}(z^{[l]}) = tanh(z^{[l]})$$



Values between -1 and 1

Keeps the sign of the input

Same issues as Sigmoid

Summary

- ReLU activations suffer from dying ReLU
- Leaky ReLU solve the dying ReLu problem
- Sigmoid and Tanh have vanishing gradient and saturation problems

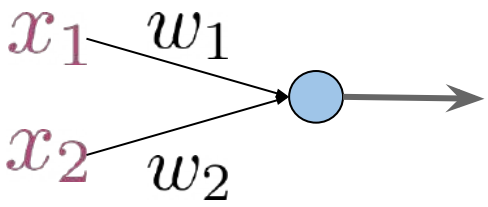
deeplearning.ai

Batch Normalization (Explained)

Outline

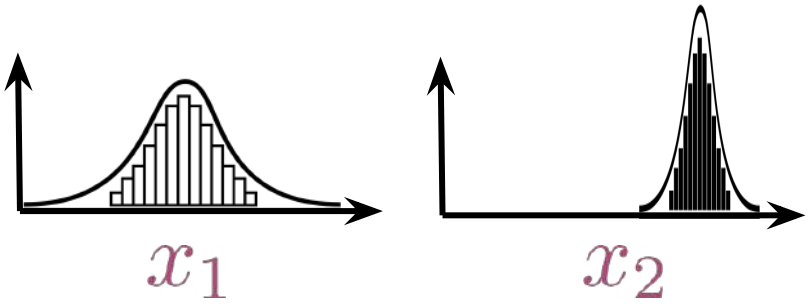
- How normalization helps models
- Internal covariate shift
- Batch normalization → stabilize training

Different Distributions of features

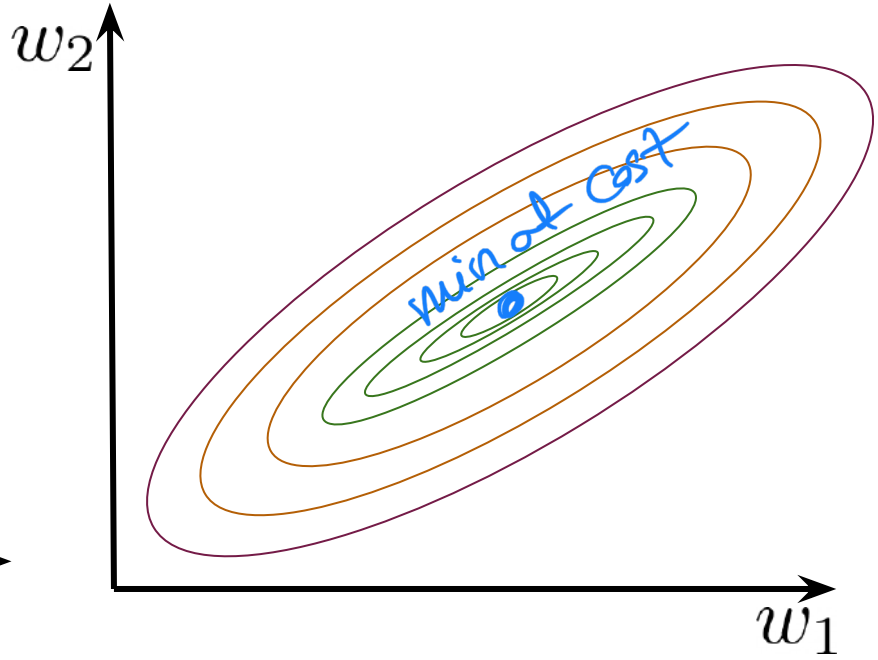


Cat

Not Cat

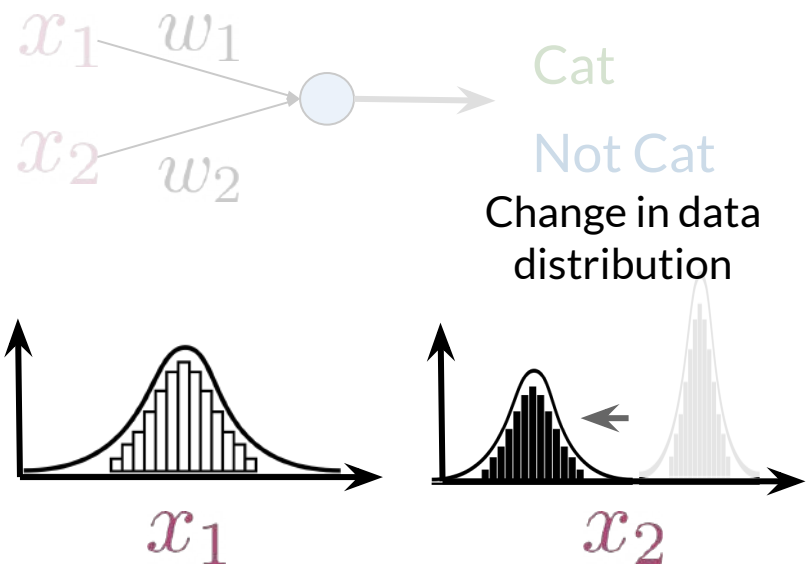


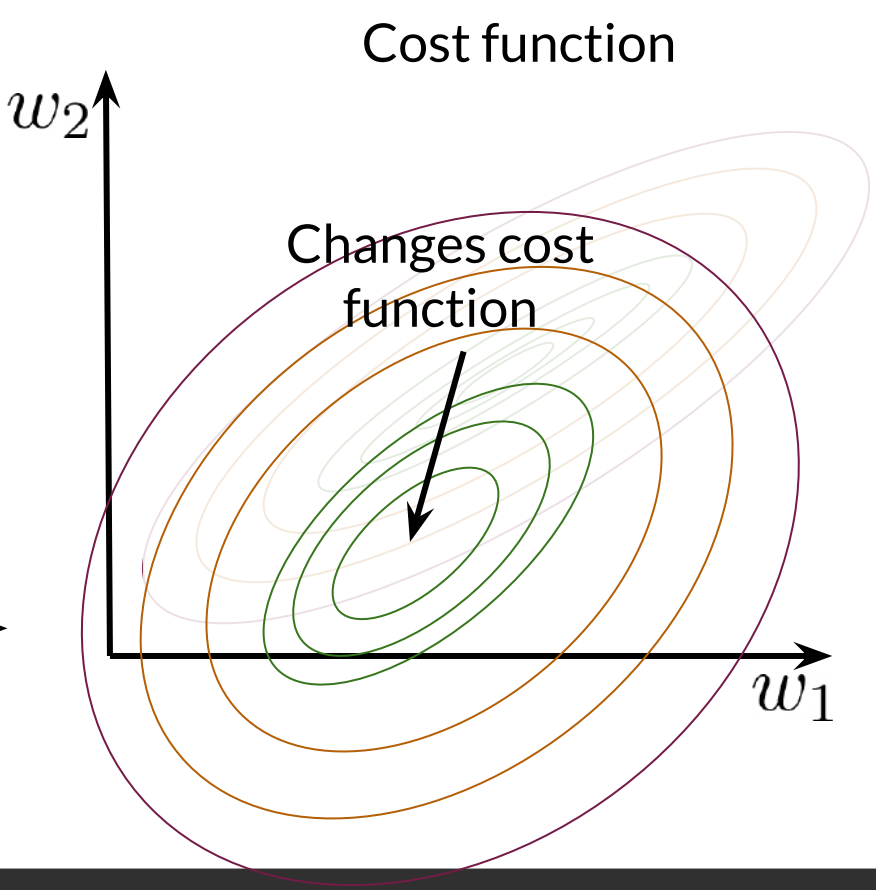
Cost function



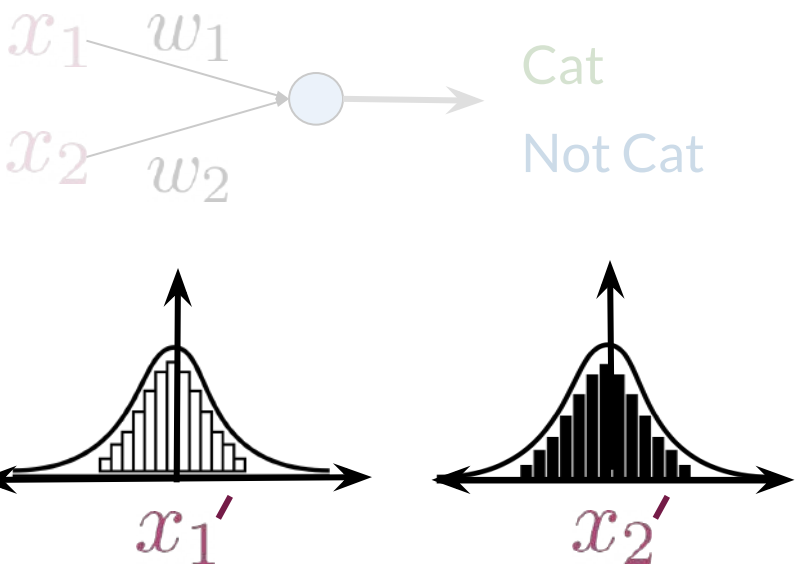
happens between train/test sets

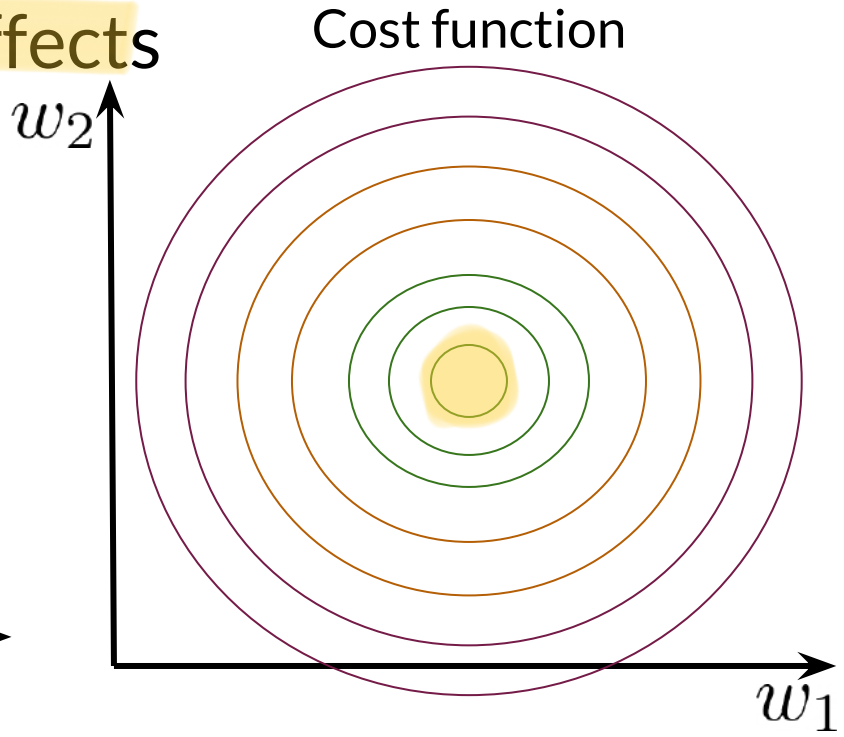
Covariate Shift



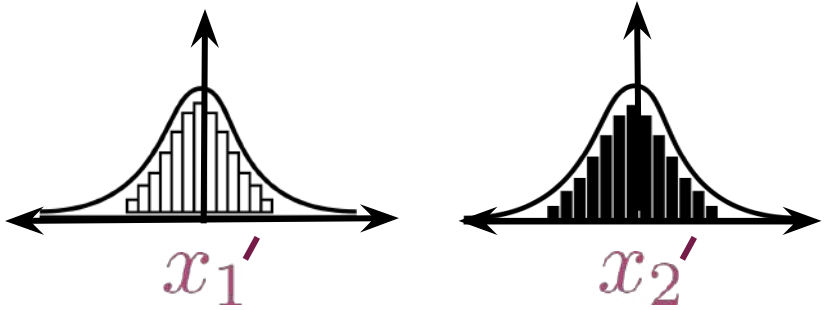


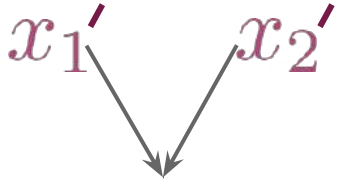
Normalization and Its Effects





Normalization and Its Effects





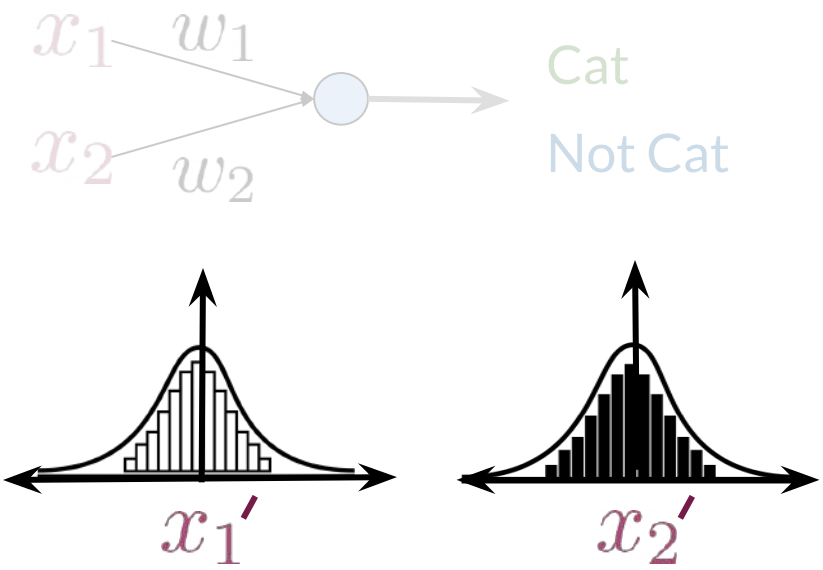
Around mean at 0
and std. at 1

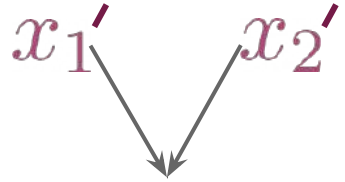
Training data uses
batch stats

Test data uses
training stats

test gets closer
to train data

Normalization and Its Effects





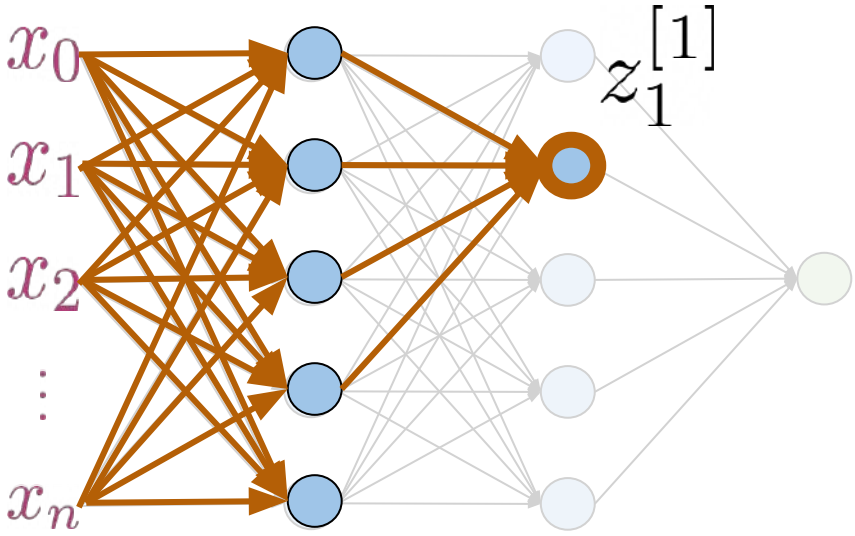
Around mean at 0 and std. at 1

↓

Reduction of covariate shift

Internal Covariate Shift

in hidden layers

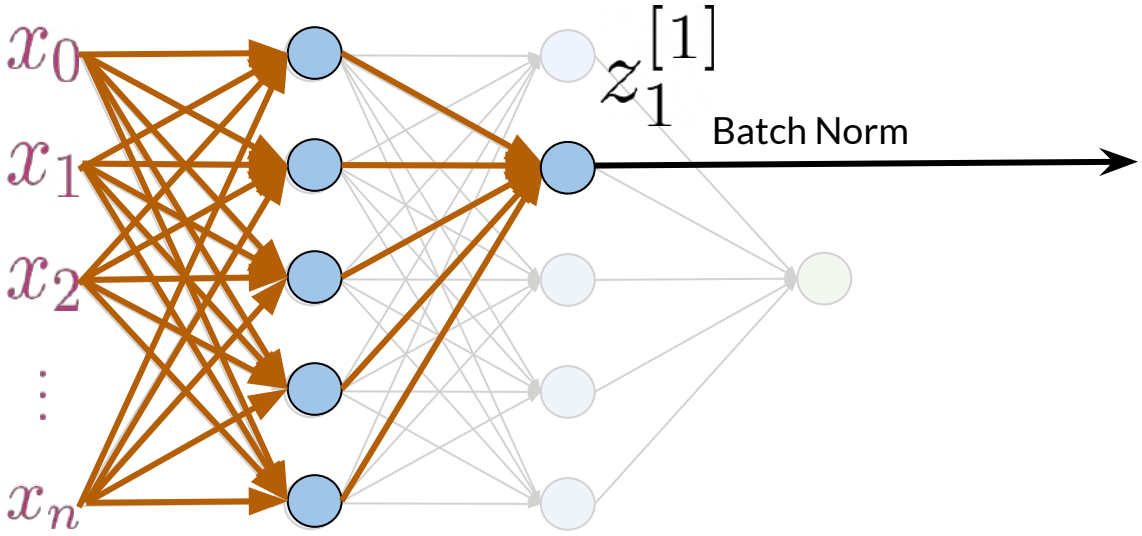


Changes in weights

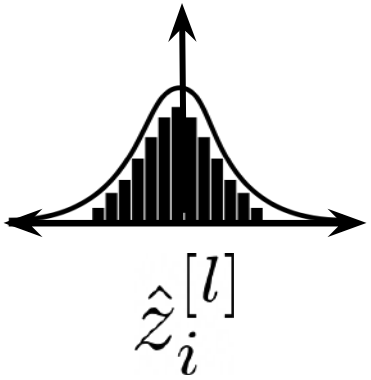
↓

Changes in activation distribution

Batch Normalization

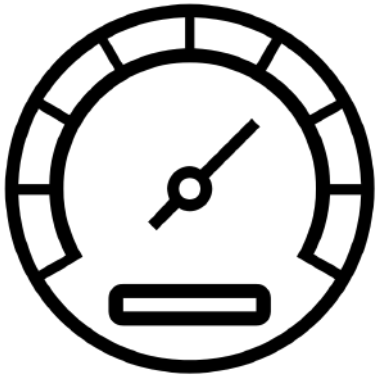


Normalizes the input for each neuron



Summary

- Batch normalization smooths the cost function
- Batch normalization reduces the internal covariate shift
- Batch normalization speeds up learning!

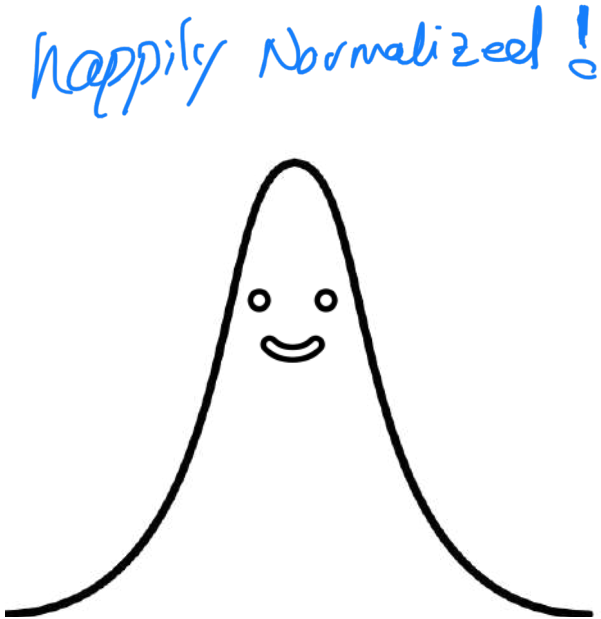


deeplearning.ai

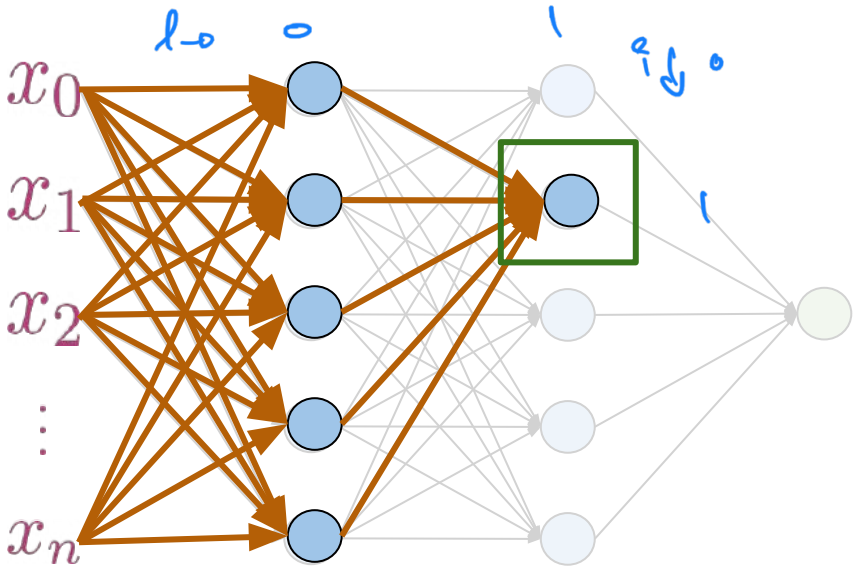
Batch Normalization (Procedure)

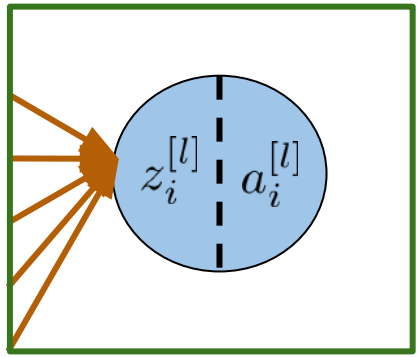
Outline

- Batch norm for training
- Batch norm for testing



Batch Normalization: Training





$$z_i^{[l]} = \sum_{i=0} W_i^{[l]} a_i^{[l-1]}$$

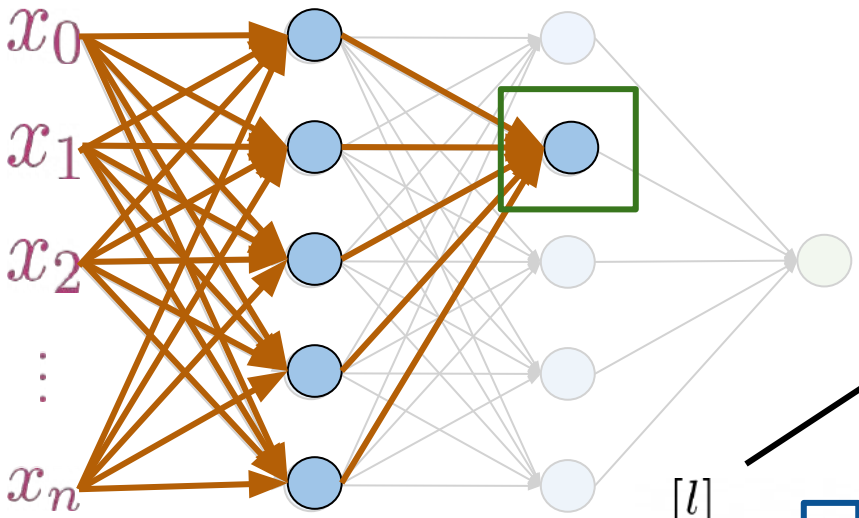
For every example in the batch

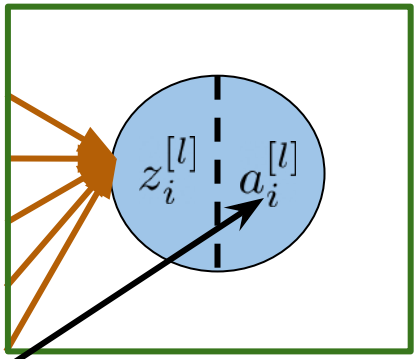
$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}$$

Batch mean of $z_i^{[l]}$

Batch std of $z_i^{[l]}$

Batch Normalization: Training





$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma^2_{z_i^{[l]}} + \epsilon}}$$

$$y_i^{[l]} = \gamma \hat{z}_i^{[l]} + \beta$$

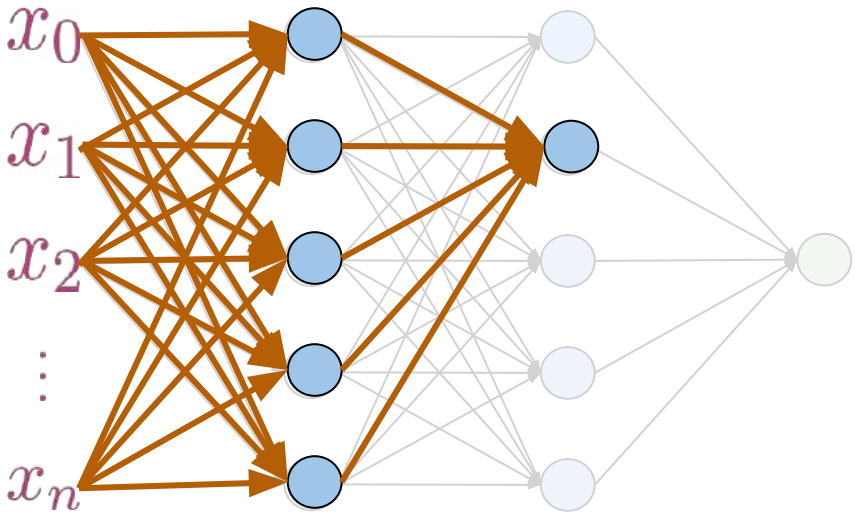
normalized

Shift factor

Scale Factor

Learnable parameters to get the optimal dist.

Batch Normalization: Test



$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mathbf{E}(z_i^{[l]})}{\sqrt{\mathrm{Var}(z_i^{[l]}) + \epsilon}}$$

Running mean and running std from training (entire set)

it's fixed

Frameworks like Tensorflow and Pytorch keep track of them

Summary

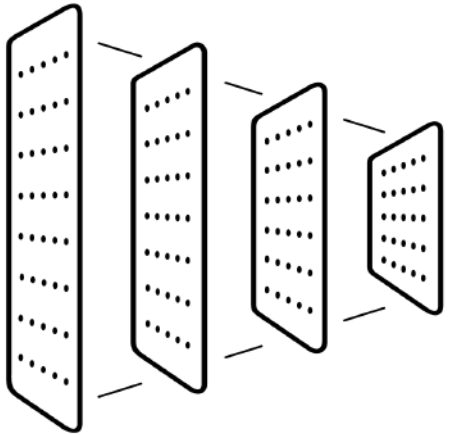
- Batch norm introduces learnable shift and scale factors
- During test, the running statistics from training are used
- Frameworks take care of the whole process

deeplearning.ai

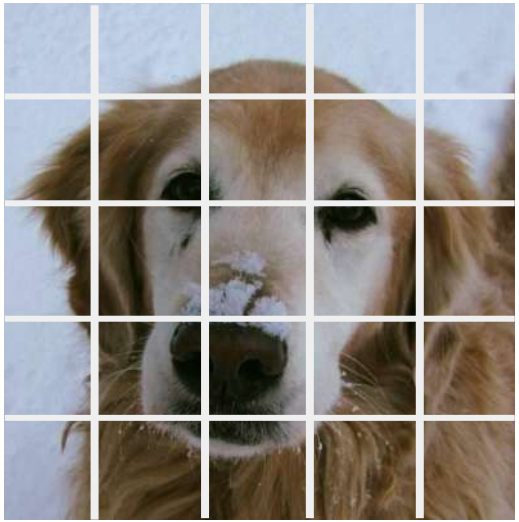
Review of Convolutions

Outline

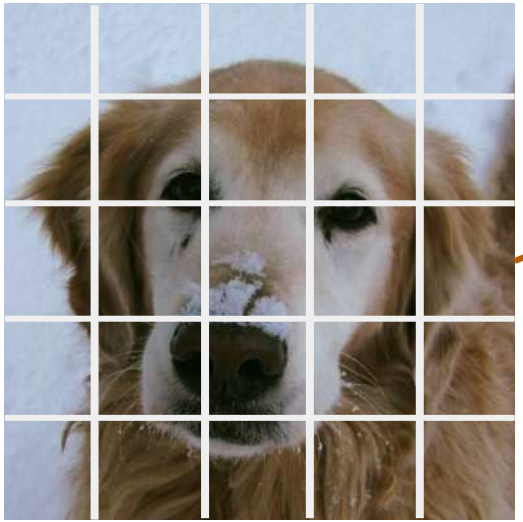
- What convolutions are
- How they work



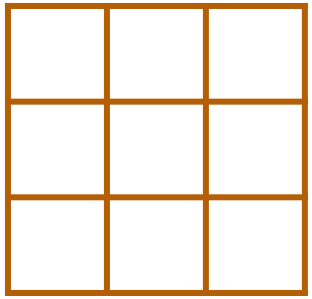
What is a convolution?



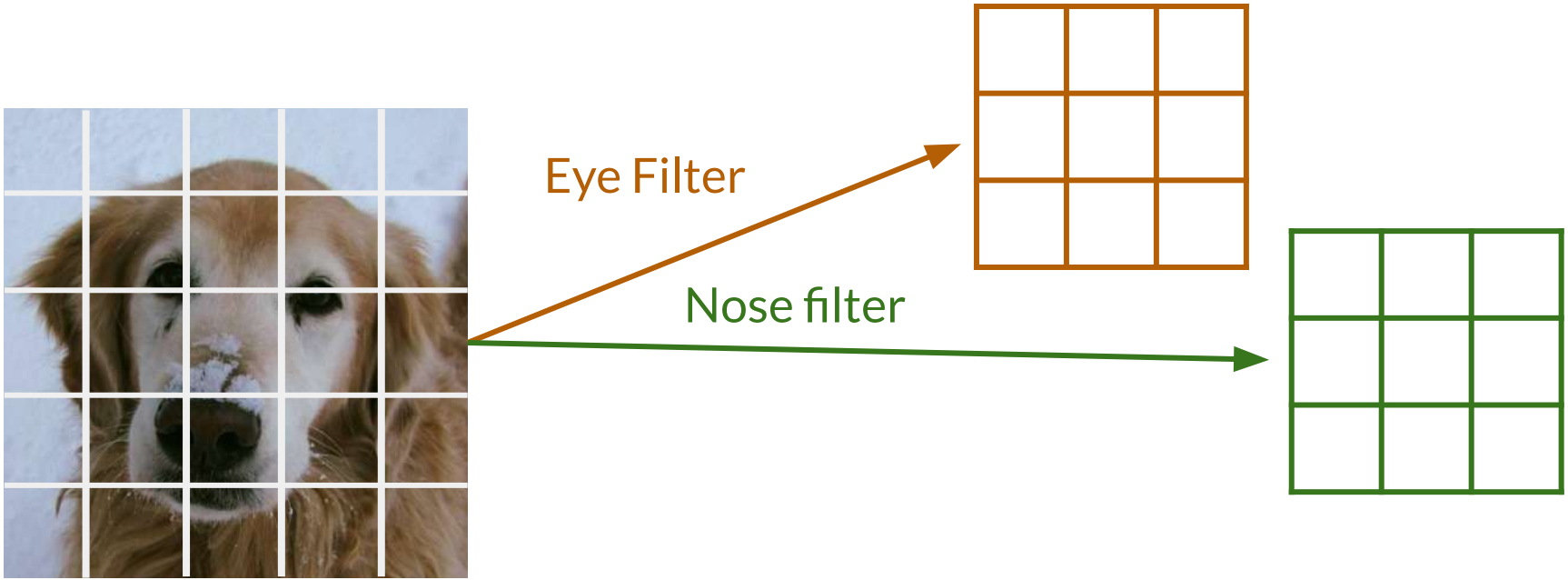
What is a convolution?



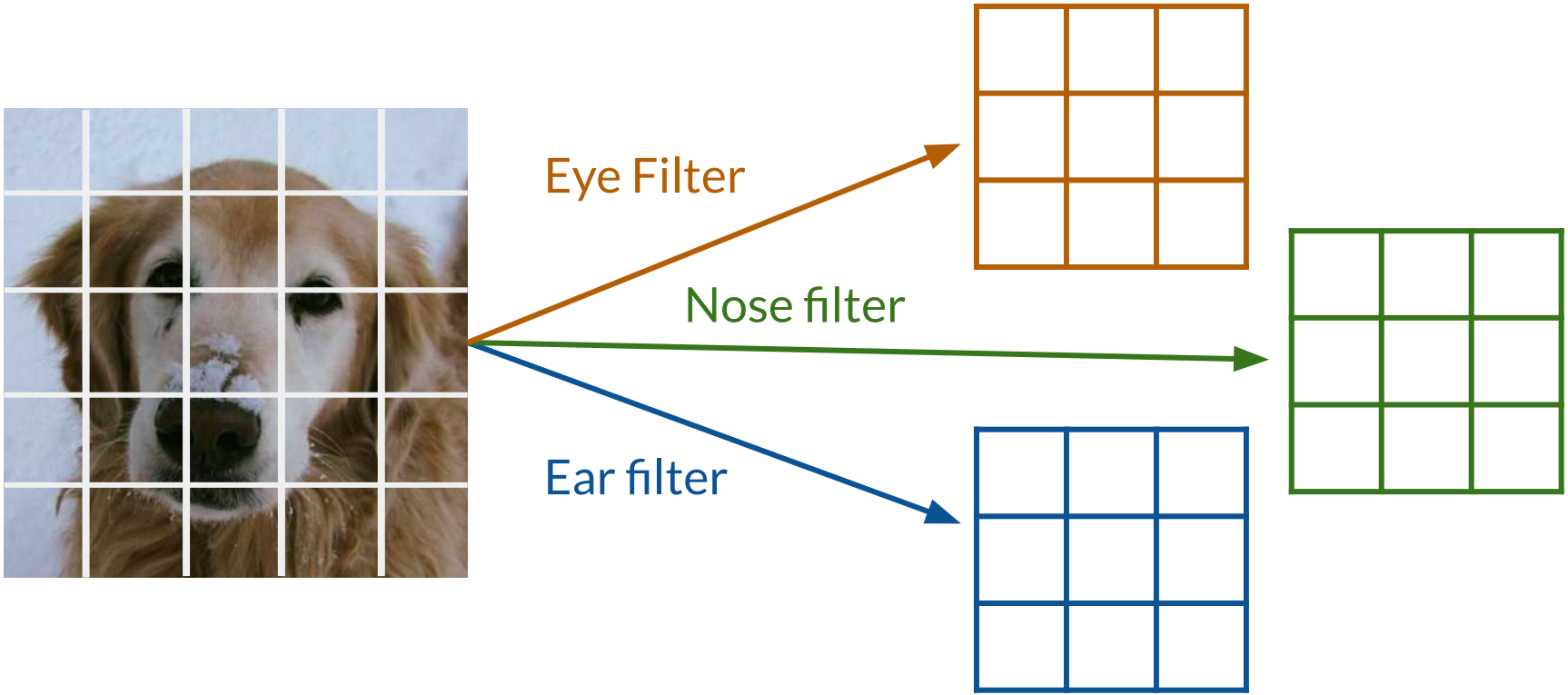
Eye Filter



What is a convolution?

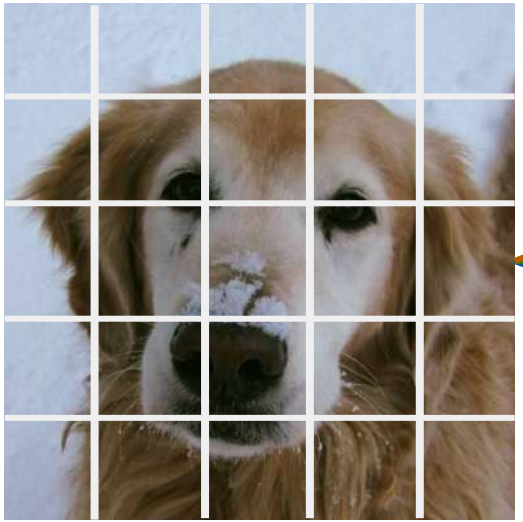


What is a convolution?



What is a convolution?

values learned in training



Eye Filter

5	3	1
10	23	1
1	42	4

Nose filter

2	21	5
23	45	12
3	32	4

Ear filter

32	2	24
25	12	66
3	45	2

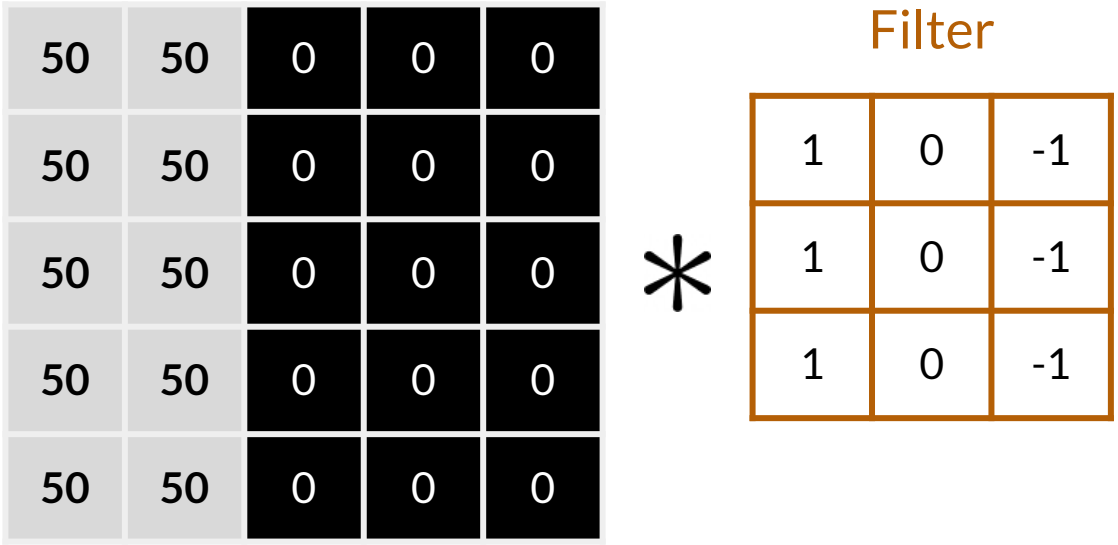
What is a convolution?

50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

0 (black) to 255 (white)

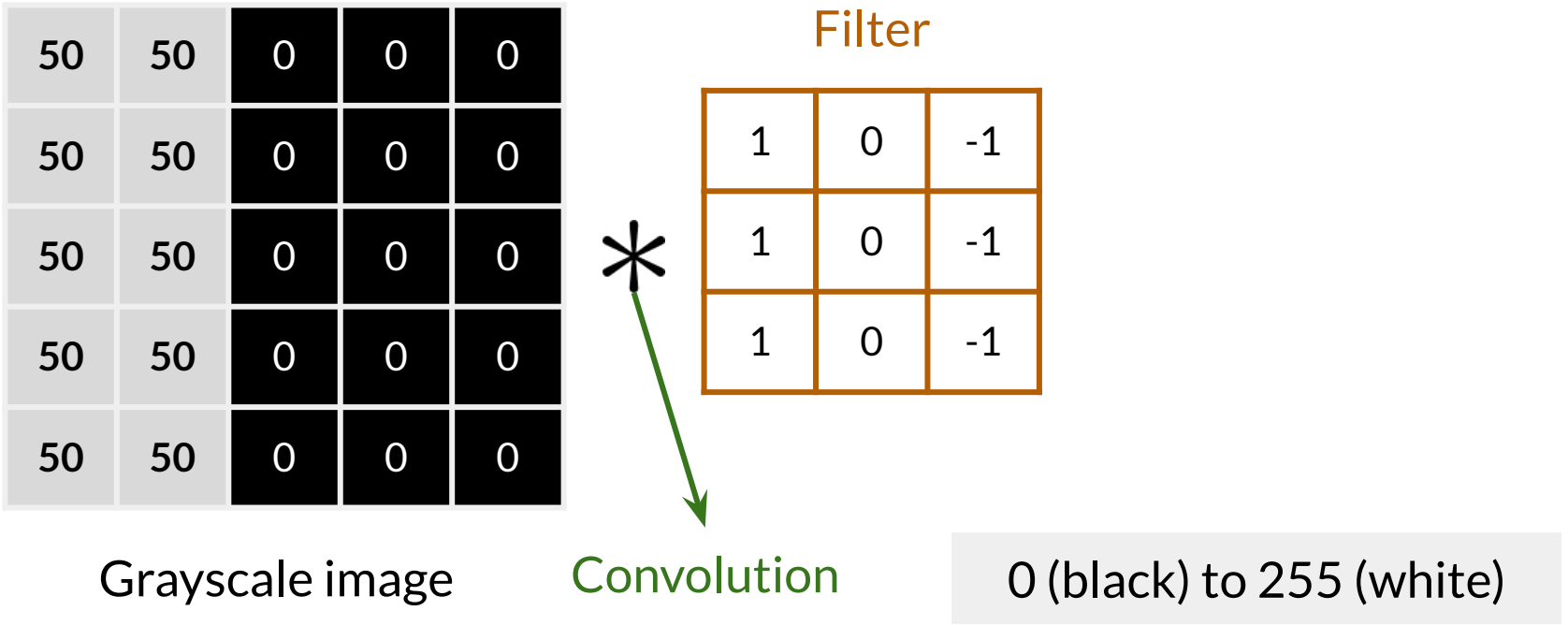
What is a convolution?



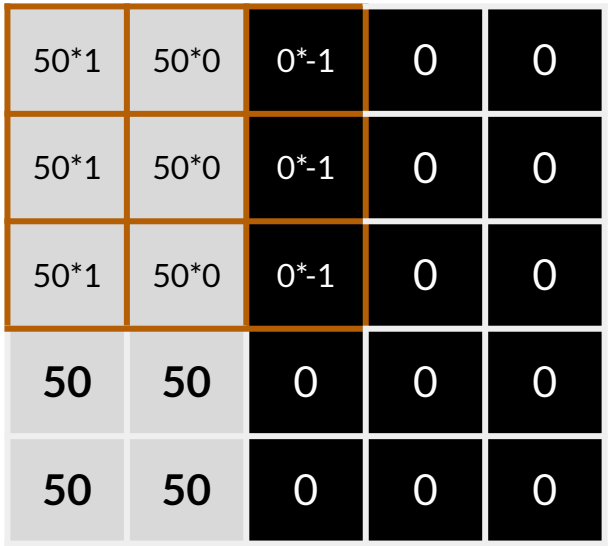
Grayscale image

0 (black) to 255 (white)

What is a convolution?



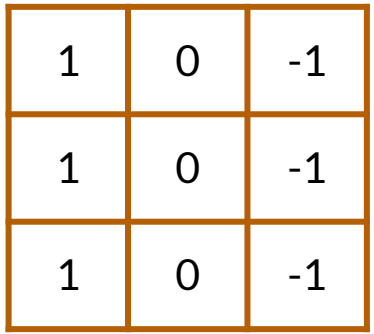
What is a convolution?



Grayscale image

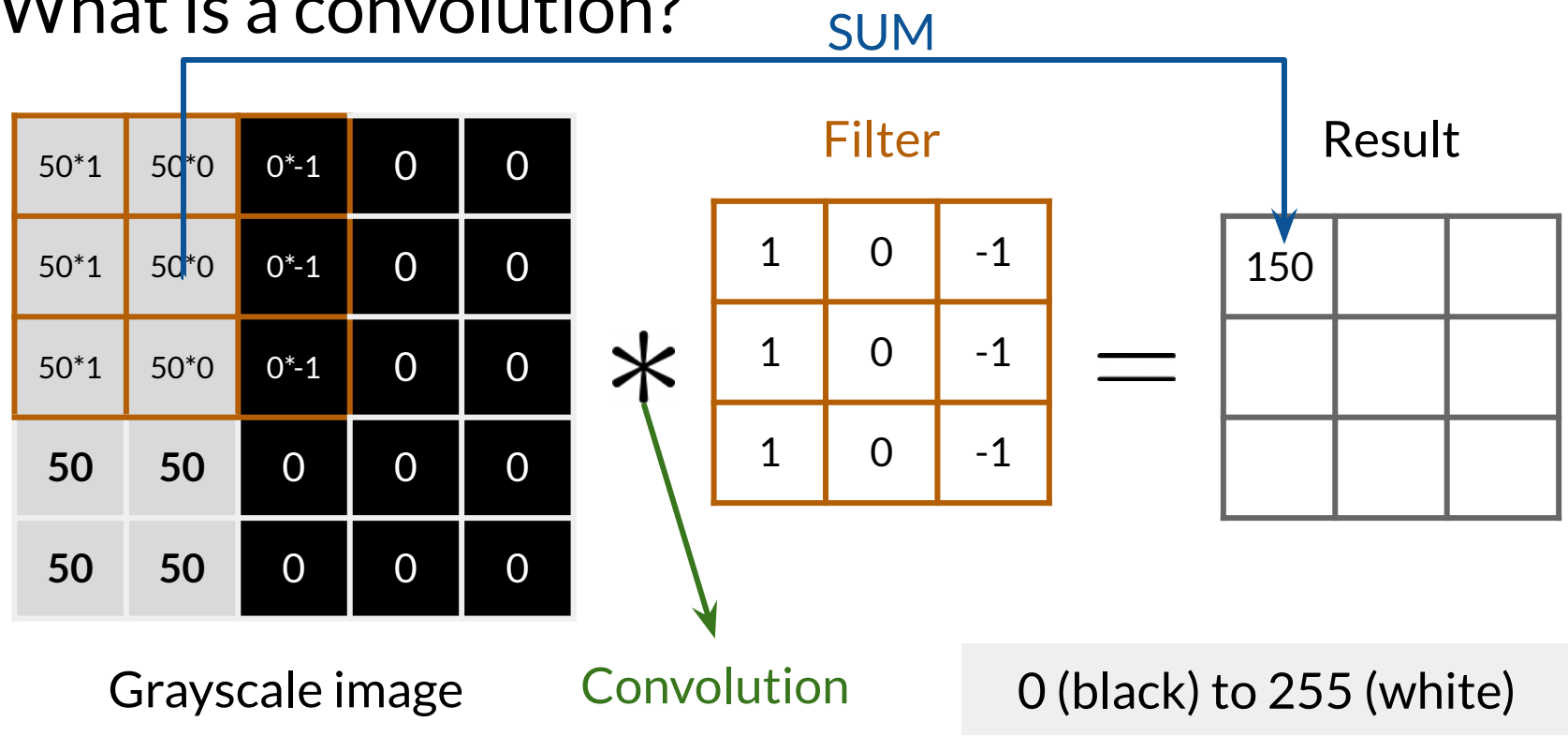
Convolution

Filter

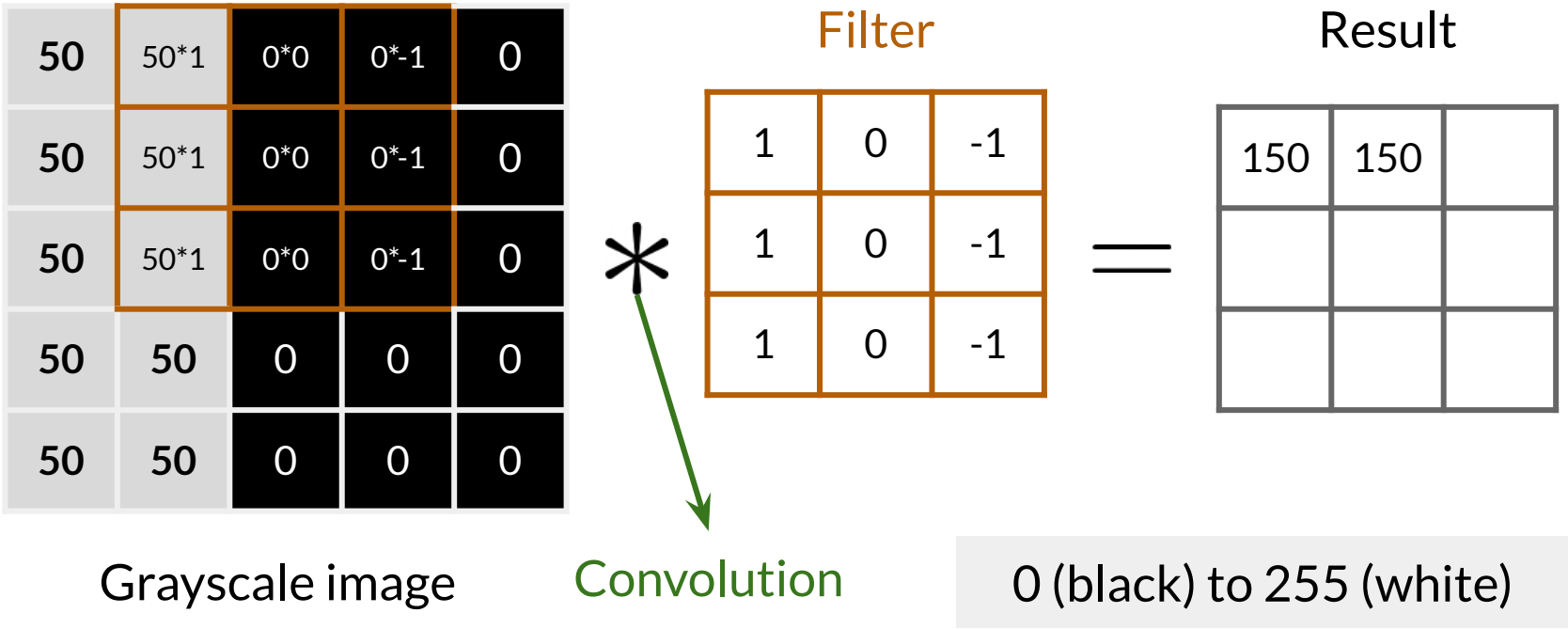


0 (black) to 255 (white)

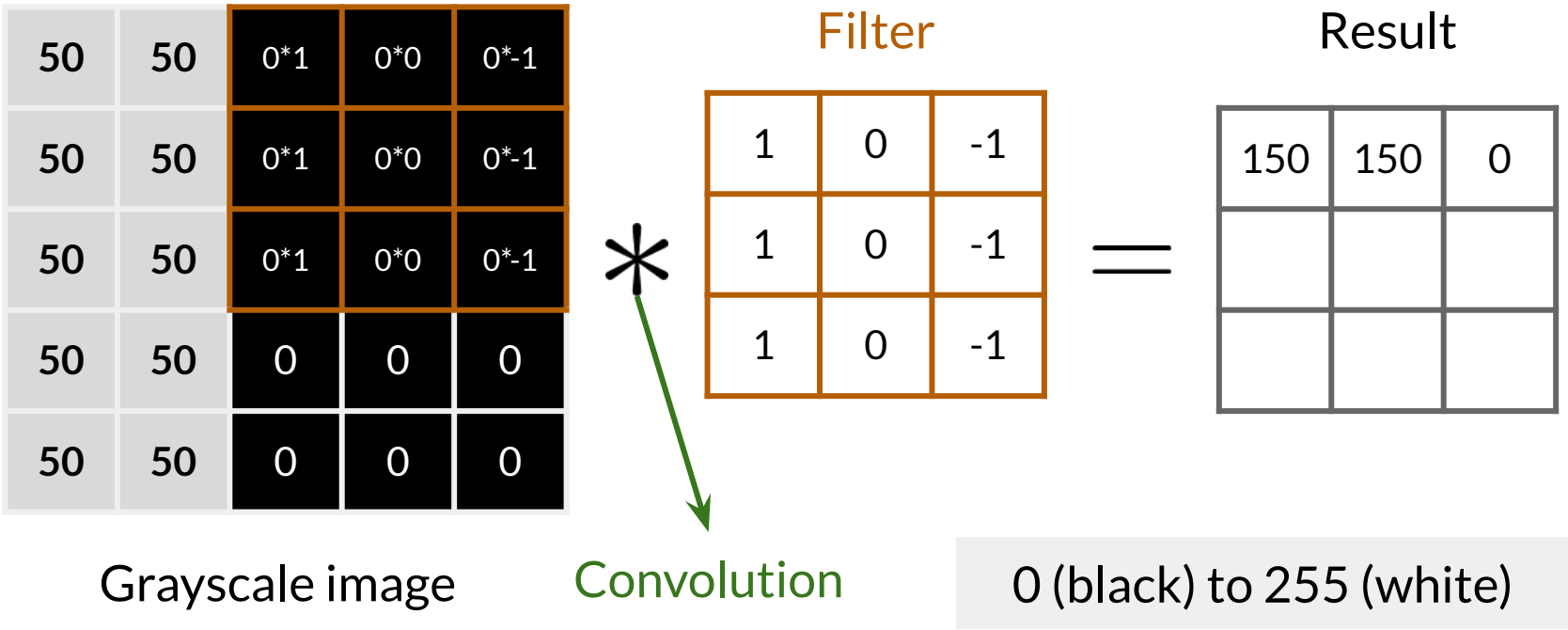
What is a convolution?



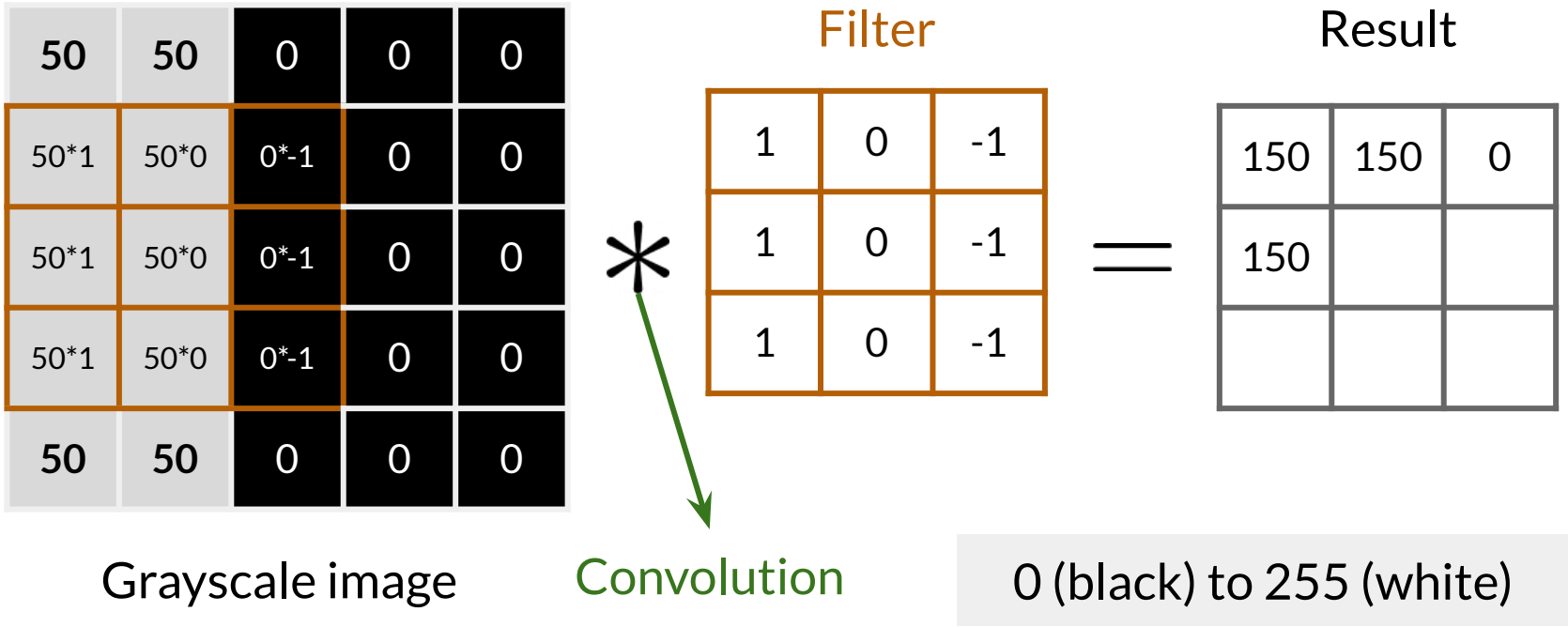
What is a convolution?



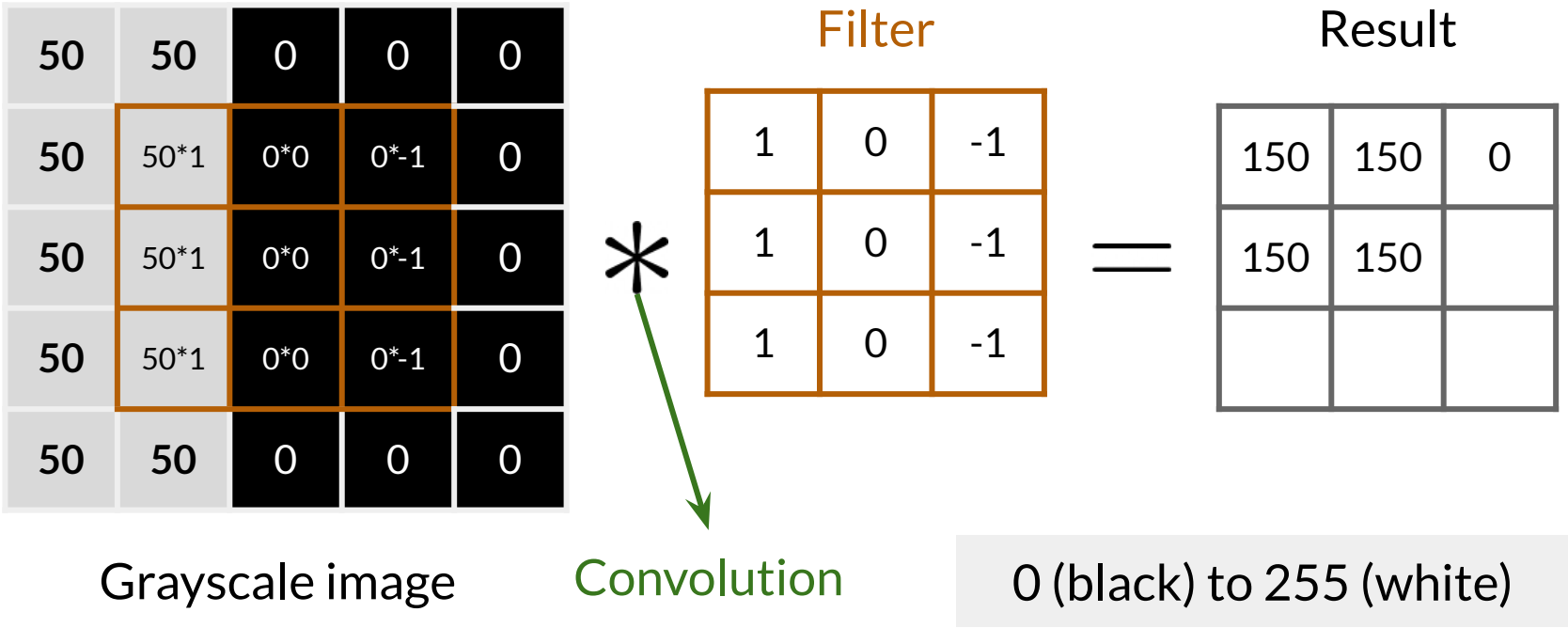
What is a convolution?



What is a convolution?

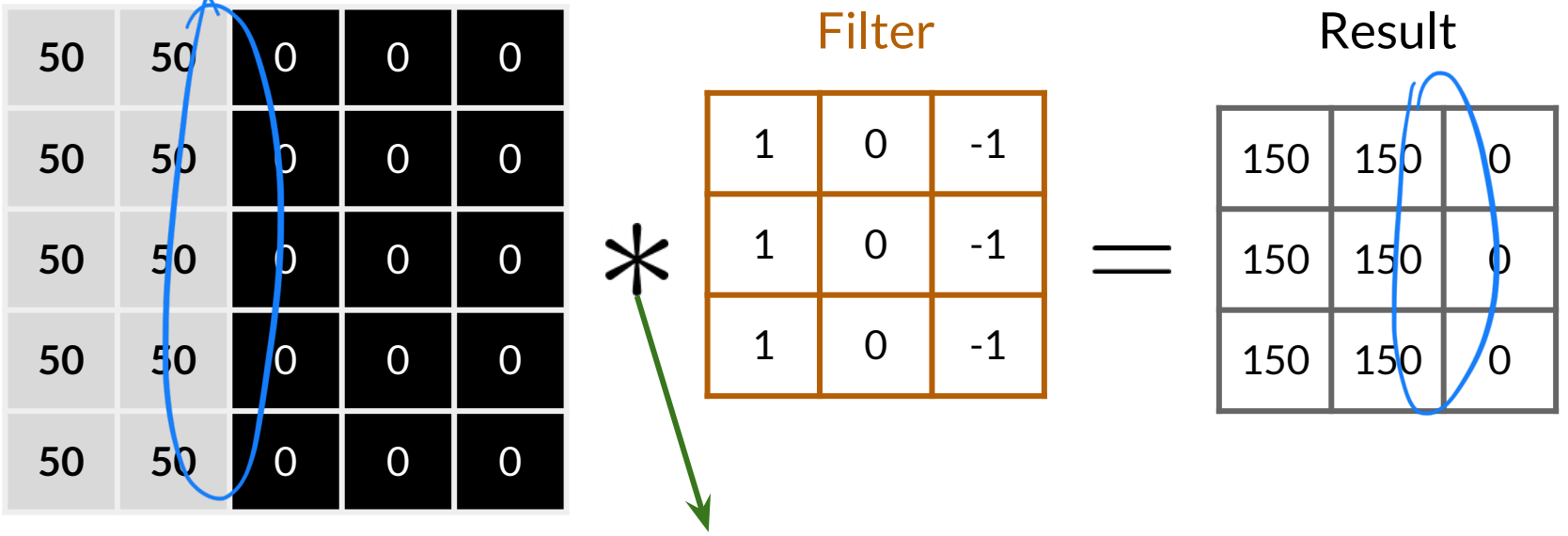


What is a convolution?



What is a convolution?

does vertical line detection



Summary

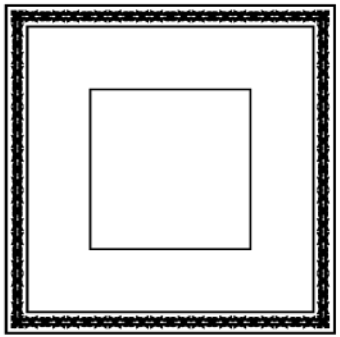
- Convolutions are useful layers for processing images
- They scan the image to detect useful features
- Just element-wise products and sums!

deeplearning.ai

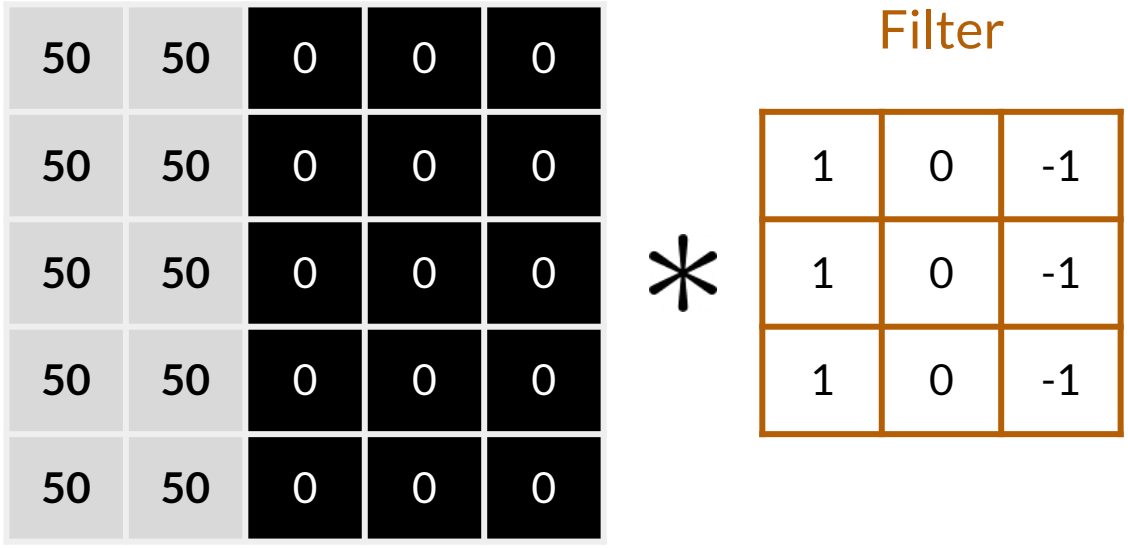
Padding and Stride

Outline

- Padding and stride
- The intuition behind padding

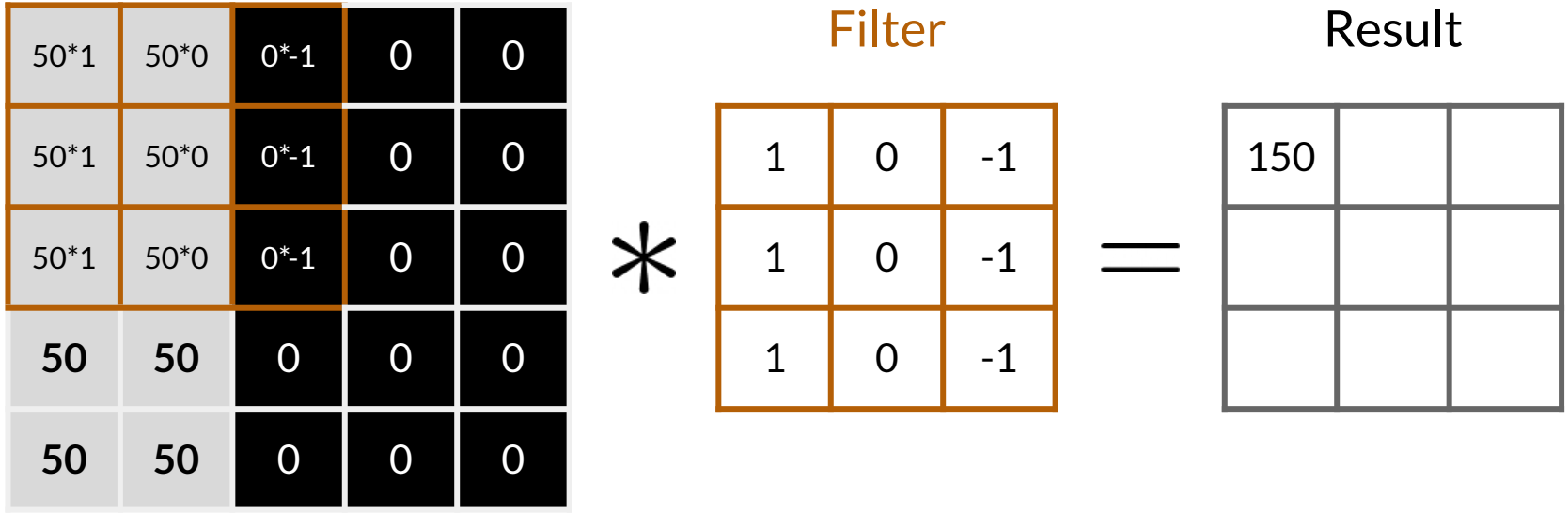


Stride



Grayscale image

Stride



Grayscale image

Stride

→ **1 Pixel** to the right

50	50*1	0*0	0*-1	0
50	50*1	0*0	0*-1	0
50	50*1	0*0	0*-1	0
50	50	0	0	0
50	50	0	0	0

Grayscale image

*

Filter

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

=

Result

150	150	

Stride

→ **1 Pixel** to the right

50	50	0*1	0*0	0*-1
50	50	0*1	0*0	0*-1
50	50	0*1	0*0	0*-1
50	50	0	0	0
50	50	0	0	0

Grayscale image

*

Filter

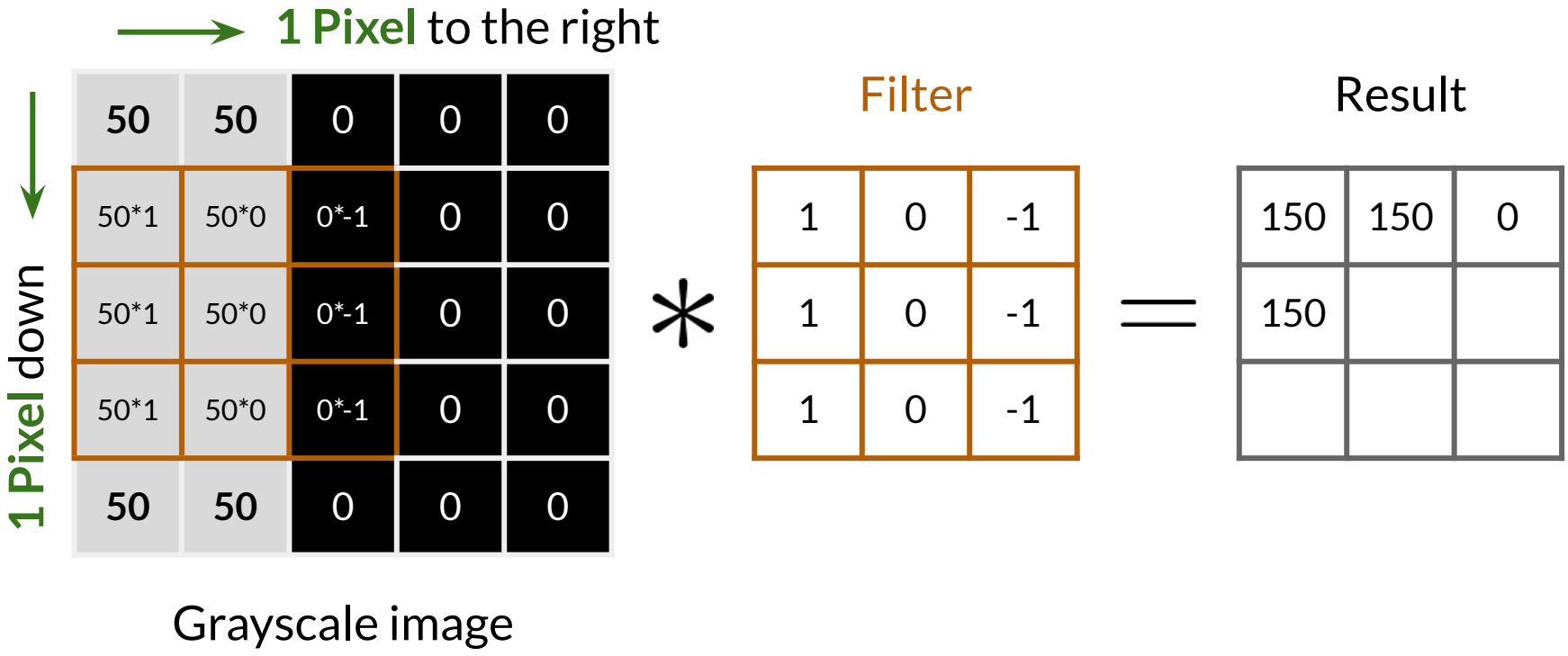
1	0	-1
1	0	-1
1	0	-1

=

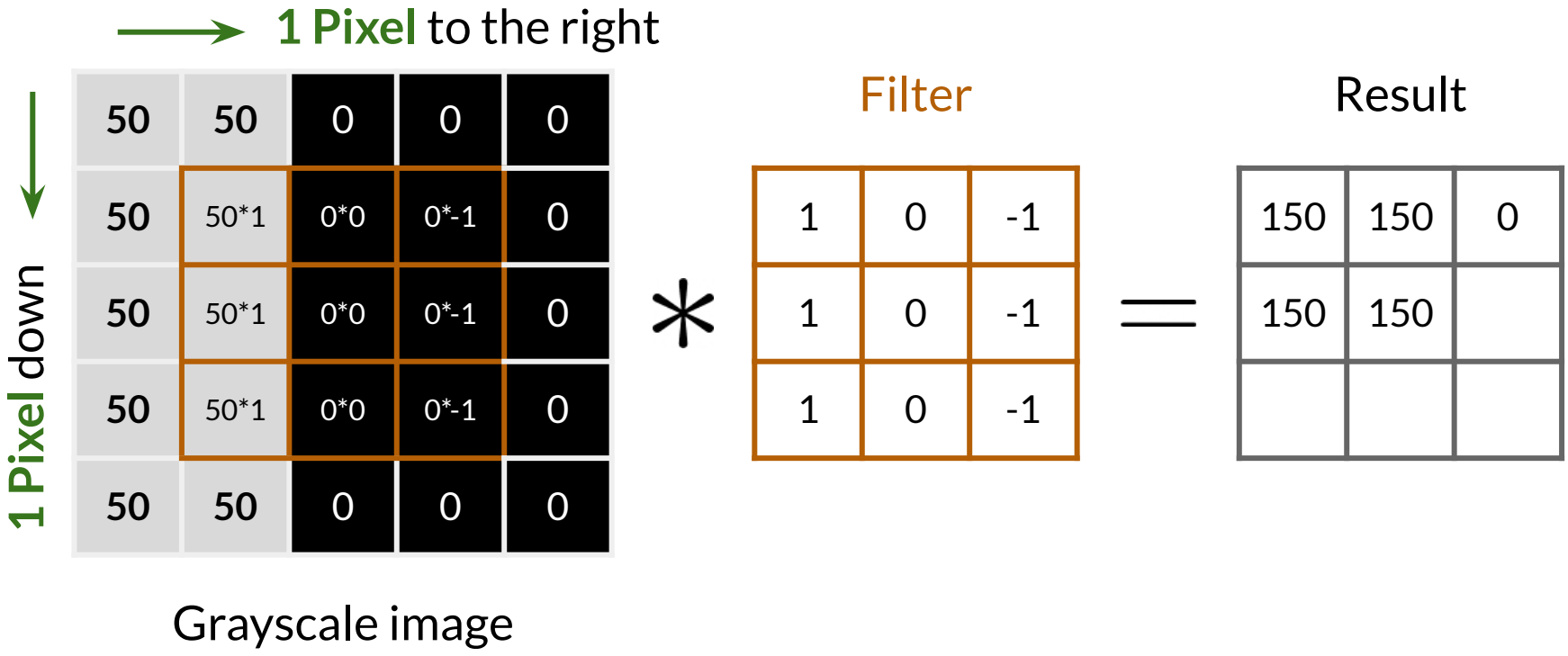
Result

150	150	0

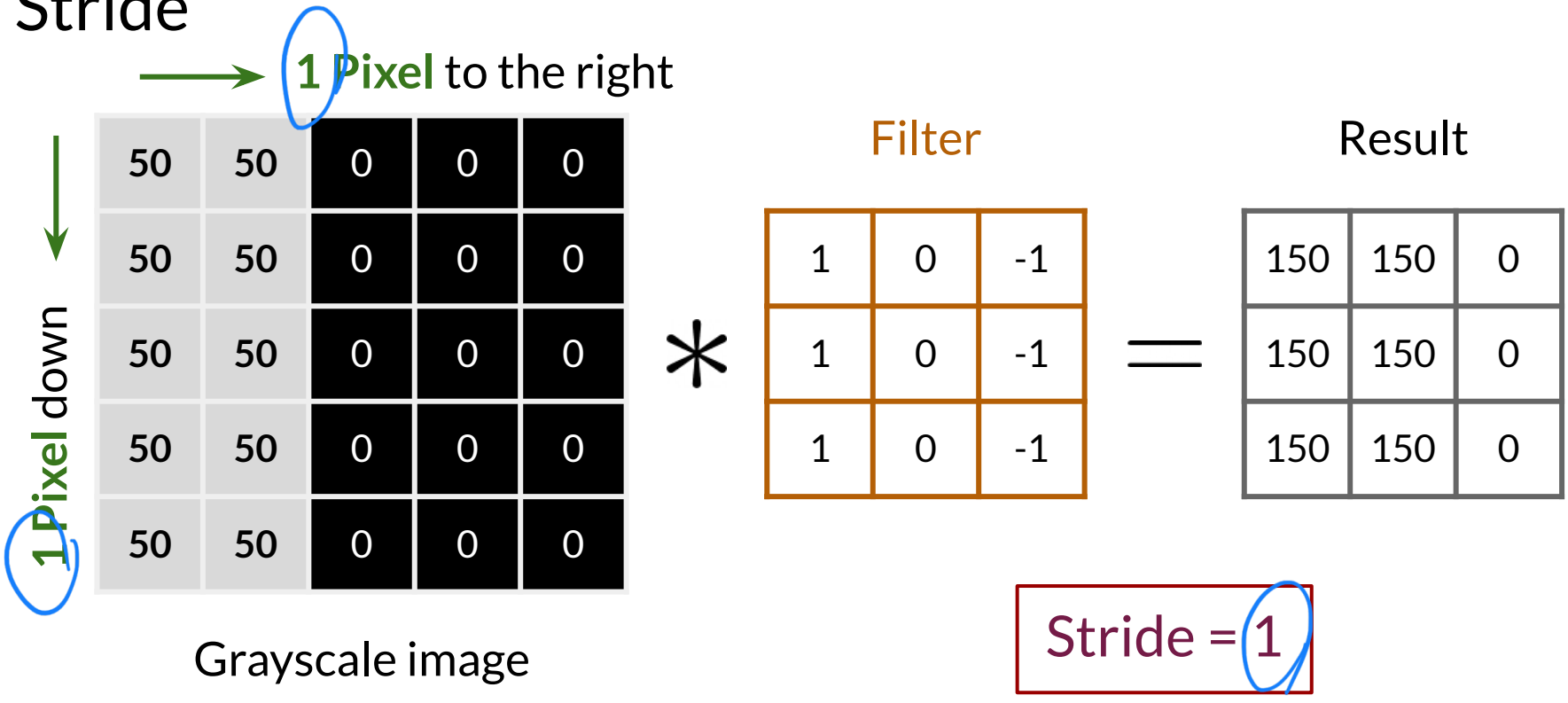
Stride


1 Pixel to the right
1 Pixel down
50 50 0 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50 50 0 0 0
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 150 0
150

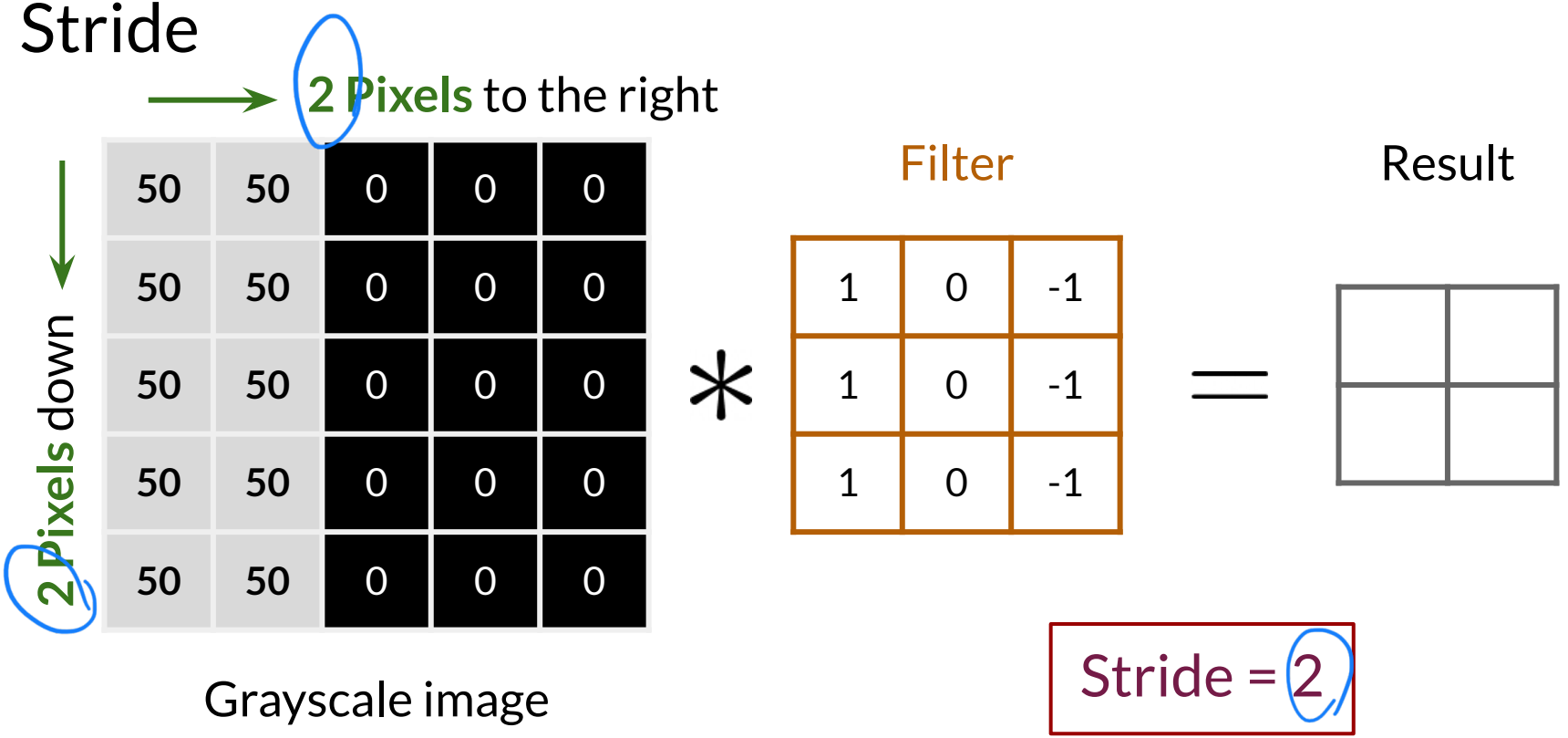
Stride


1 Pixel to the right
1 Pixel down
50 50 0 0 0
50 50*1 0*0 0*-1 0
50 50*1 0*0 0*-1 0
50 50*1 0*0 0*-1 0
50 50 0 0 0
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 150 0
150 150

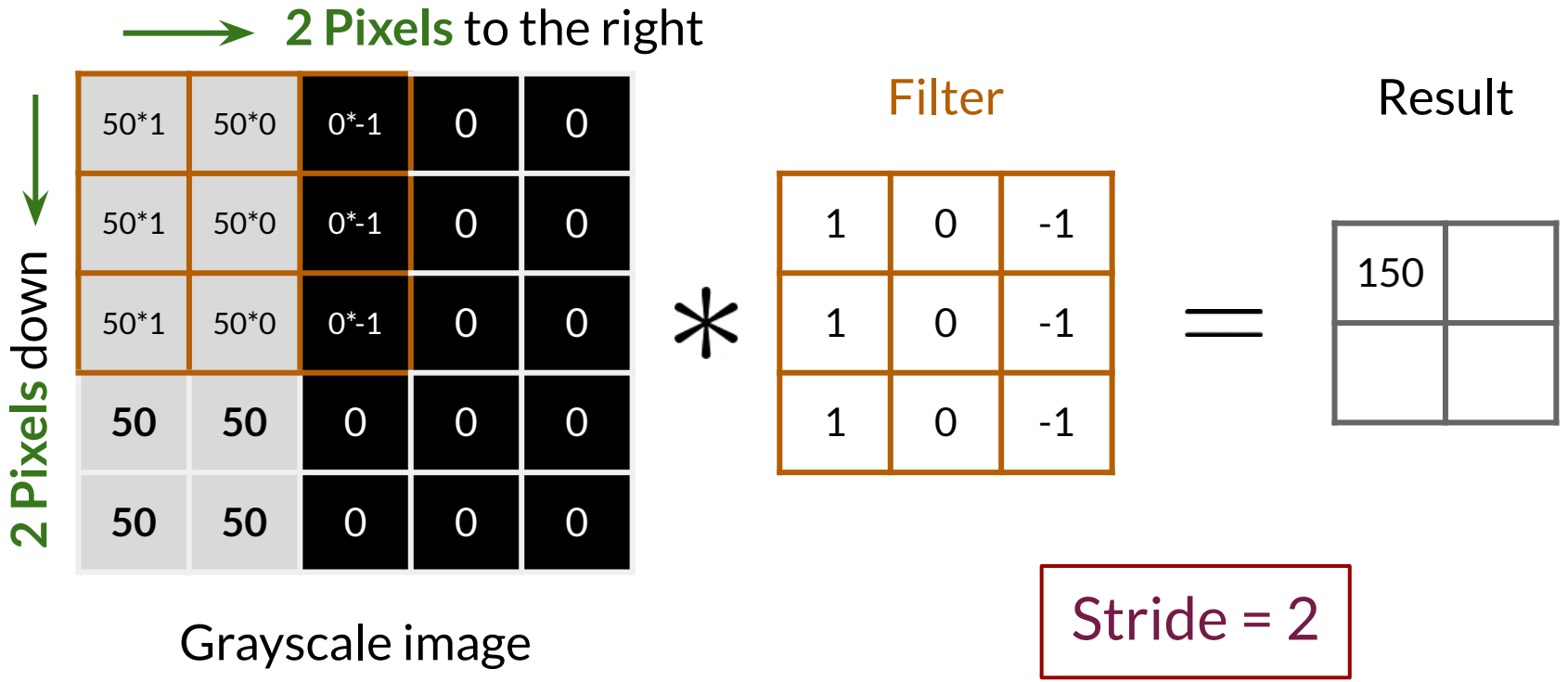
Stride


1 Pixel to the right
1 Pixel down
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
Grayscale image
*
Filter
1 0 -1
1 0 -1
1 0 -1
=
Result
150 150 0
150 150 0
150 150 0
Stride = 1

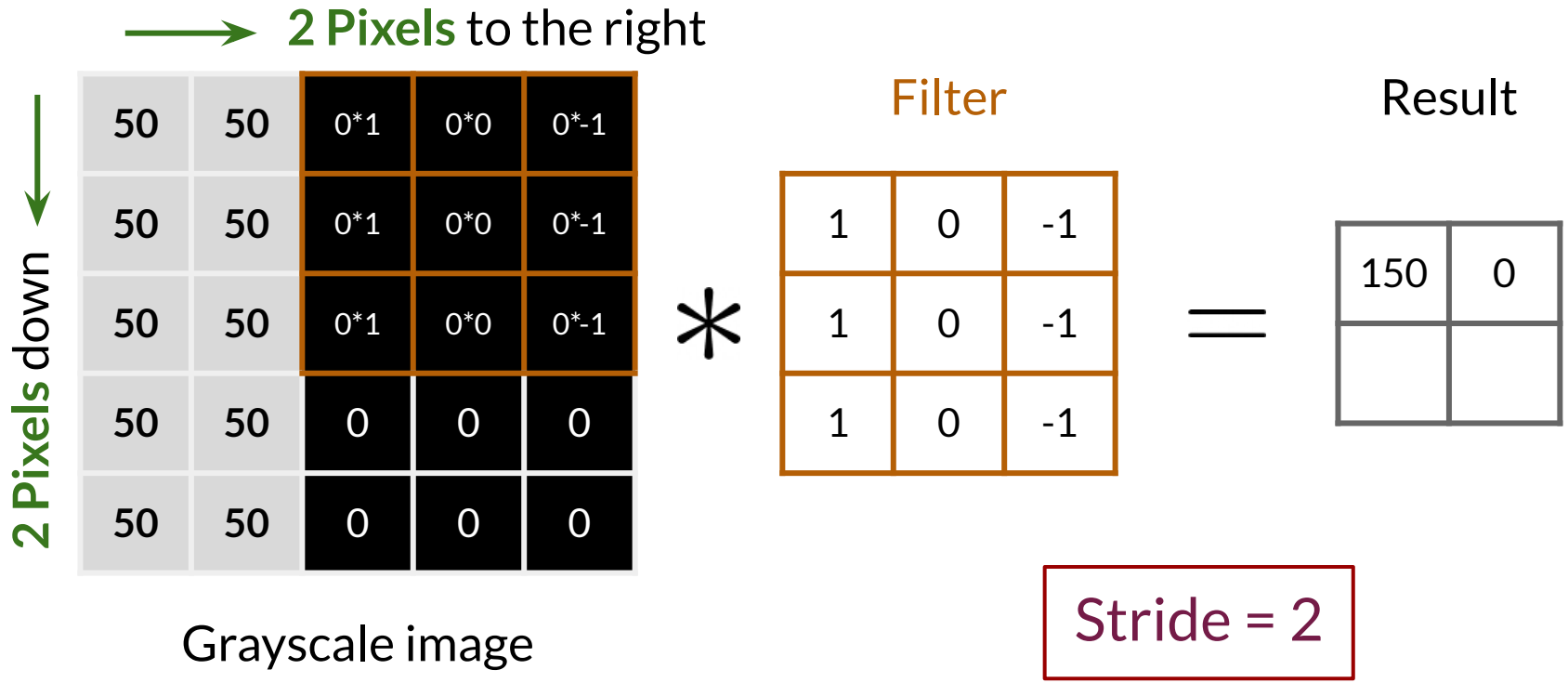
Stride


2 Pixels to the right
2 Pixels down
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
Grayscale image
*
Filter
1 0 -1
1 0 -1
1 0 -1
=
Result
Stride = 2

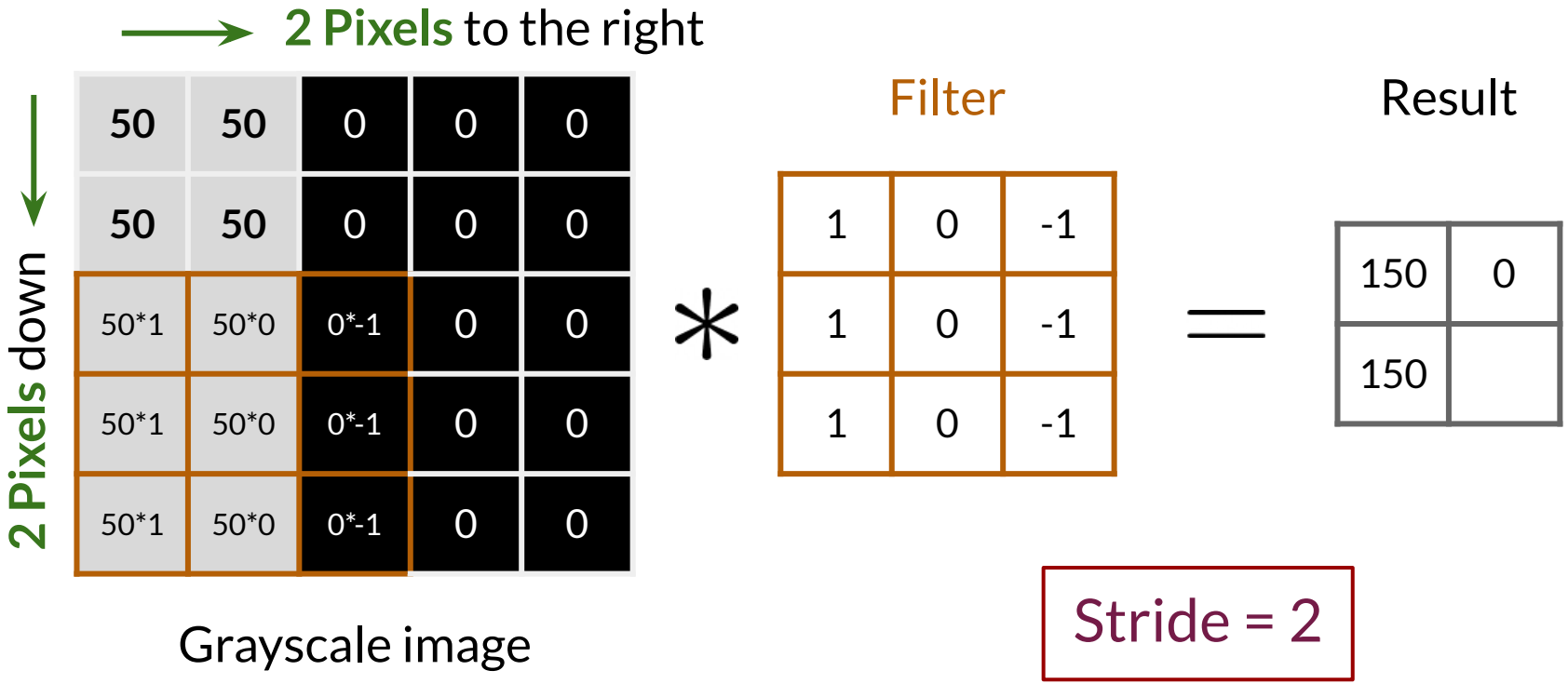
Stride


2 Pixels to the right
2 Pixels down
50*1
50*0
0*-1
0
0
50*1
50*0
0*-1
0
0
50*1
50*0
0*-1
0
0
50
50
0
0
0
50
50
0
0
0
Grayscale image
*
Filter
1
0
-1
1
0
-1
1
0
-1
=
Result
150
Stride = 2

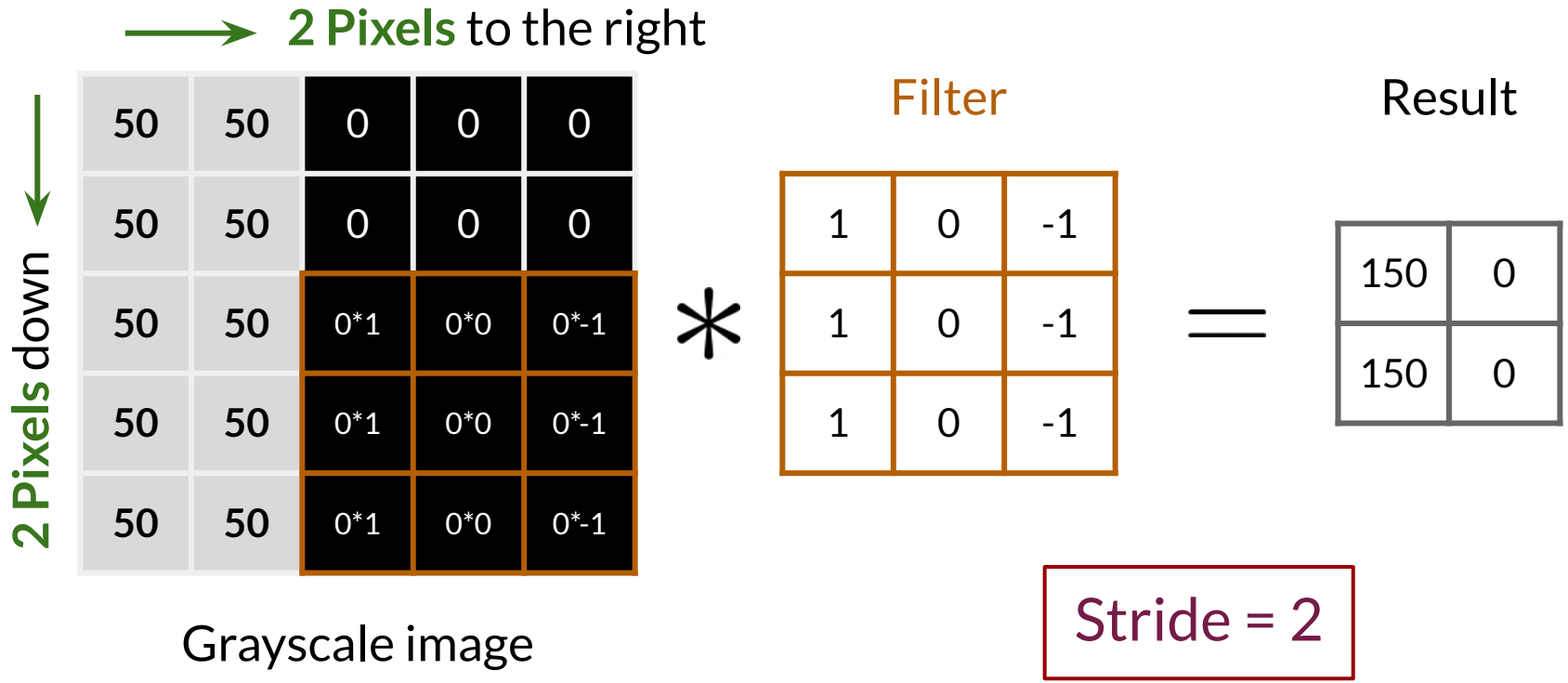
Stride


2 Pixels to the right
2 Pixels down
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
50 50 0 0 0
50 50 0 0 0
Grayscale image
*
Filter
1 0 -1
1 0 -1
1 0 -1
=
Result
150 0
Stride = 2

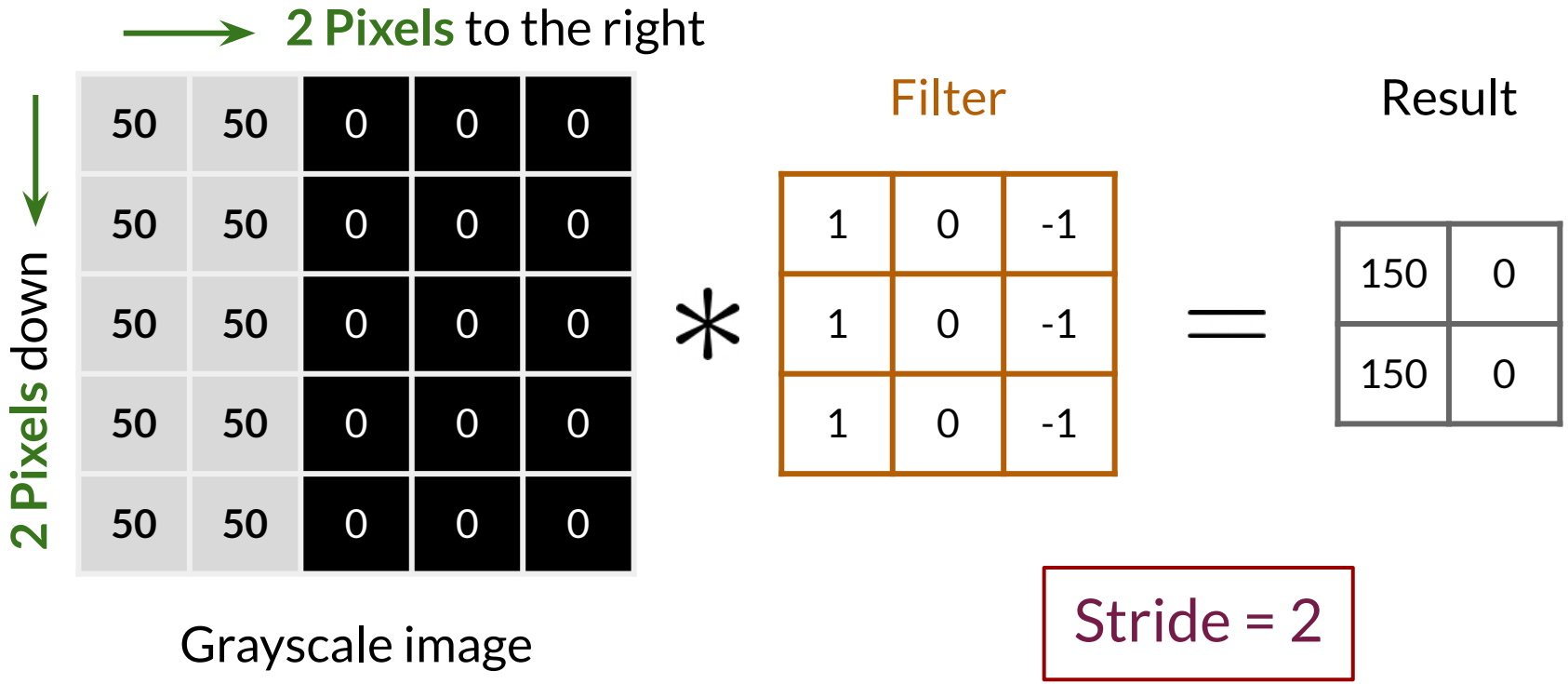
Stride


2 Pixels to the right
2 Pixels down
50 50 0 0 0
50 50 0 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
50*1 50*0 0*-1 0 0
Grayscale image
*
Filter
1 0 -1
1 0 -1
1 0 -1
=
Result
150 0
150
Stride = 2

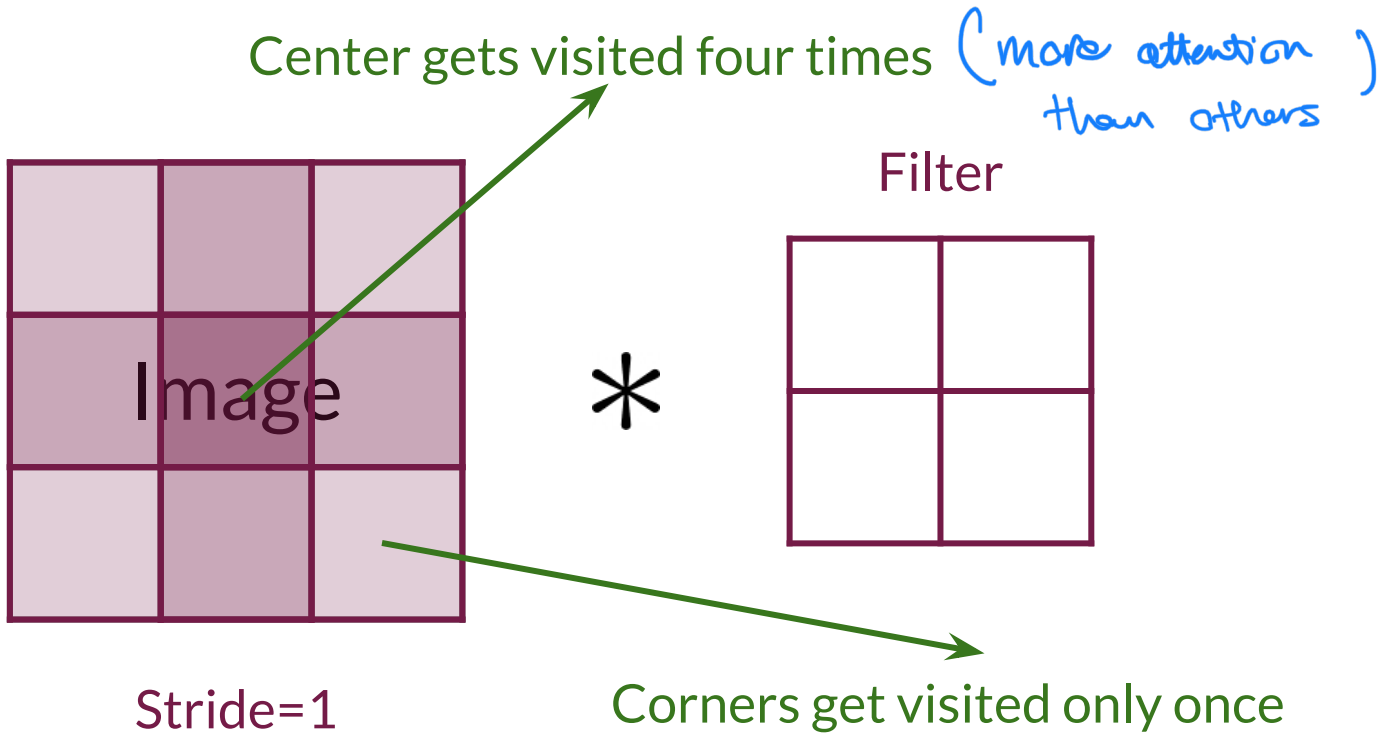
Stride


2 Pixels to the right
2 Pixels down
50 50 0 0 0
50 50 0 0 0
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
50 50 0*1 0*0 0*-1
Grayscale image
Filter
1 0 -1
1 0 -1
1 0 -1
Result
150 0
150 0
Stride = 2

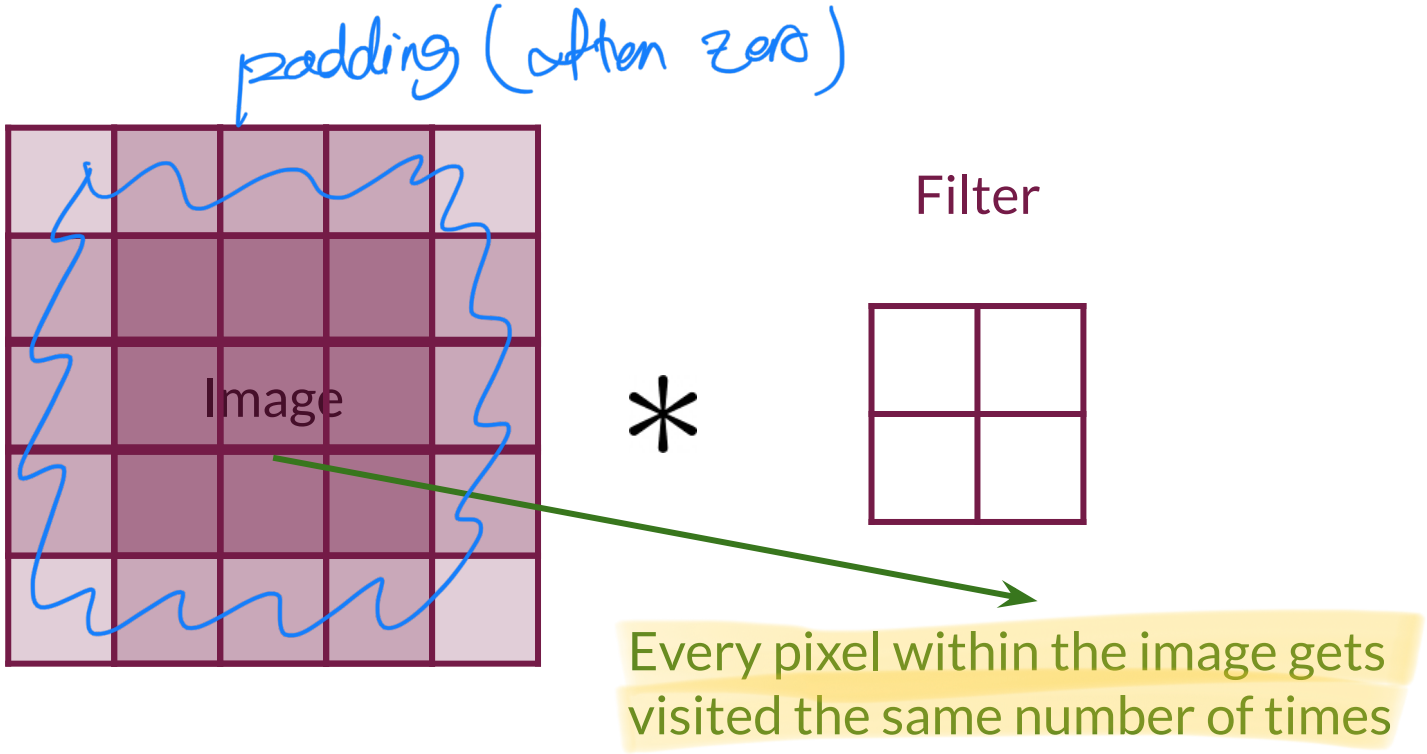
Stride


2 Pixels to the right
2 Pixels down
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
50 50 0 0 0
Grayscale image
*
Filter
1 0 -1
1 0 -1
1 0 -1
=
Result
150 0
150 0
Stride = 2

Padding


Center gets visited four times
(more attention than others)
Filter
Image
*
Stride=1
Corners get visited only once

Padding


padding (often zero)
Image
*
Filter
Every pixel within the image gets visited the same number of times

Summary

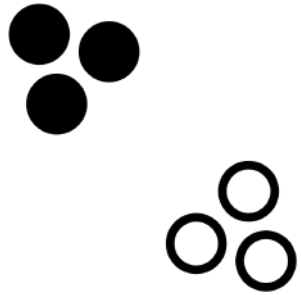
- Stride determines how the filter scans the image
- Padding is like a frame on the image
- Padding gives similar importance to the edges and the center

deeplearning.ai

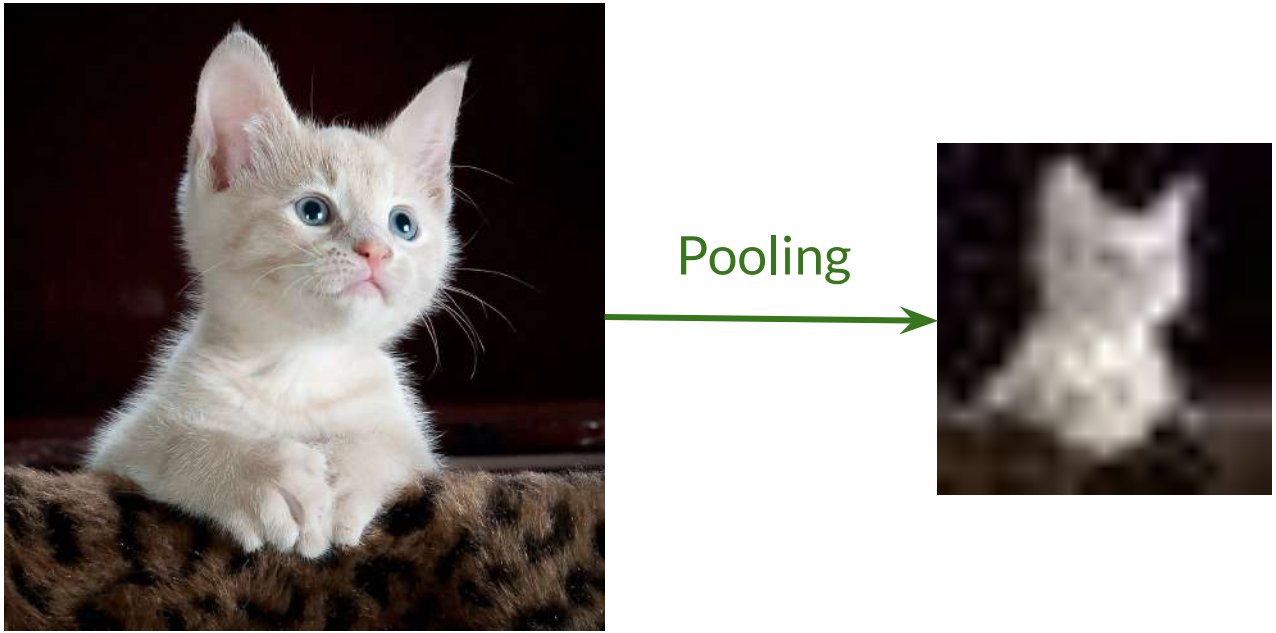
Pooling and Upsampling

Outline

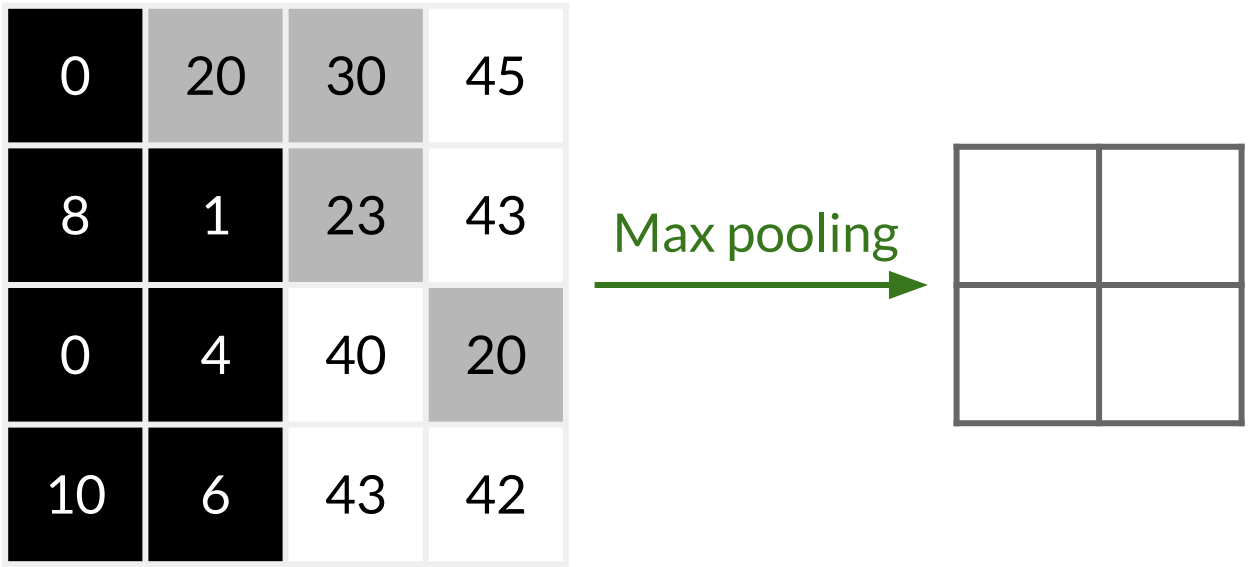
- Pooling
- Upsampling and its relation to pooling



Pooling

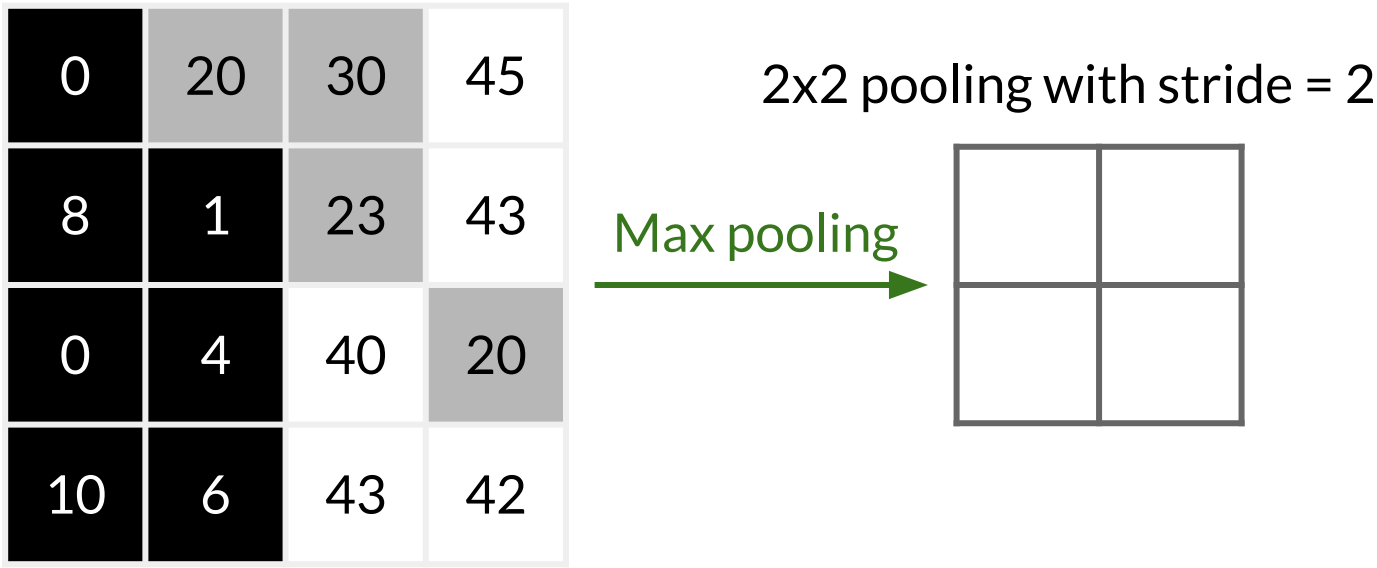


Max Pooling



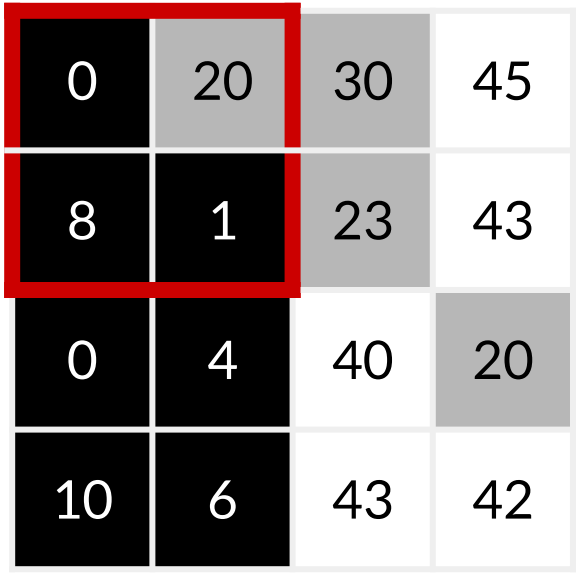
4x4 input to 2x2 output

Max Pooling



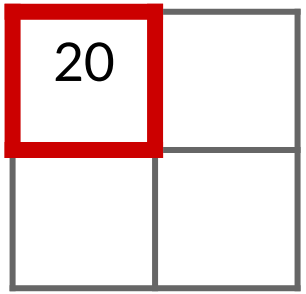
4x4 input to 2x2 output

Max Pooling



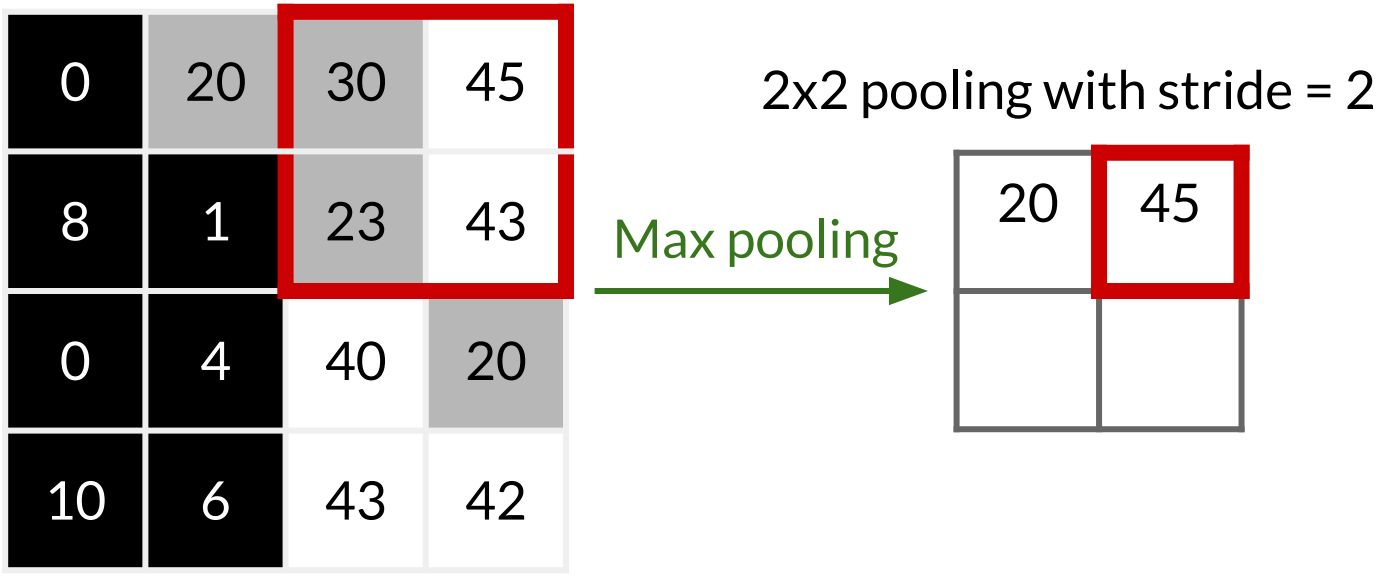
Max pooling

2x2 pooling with stride = 2



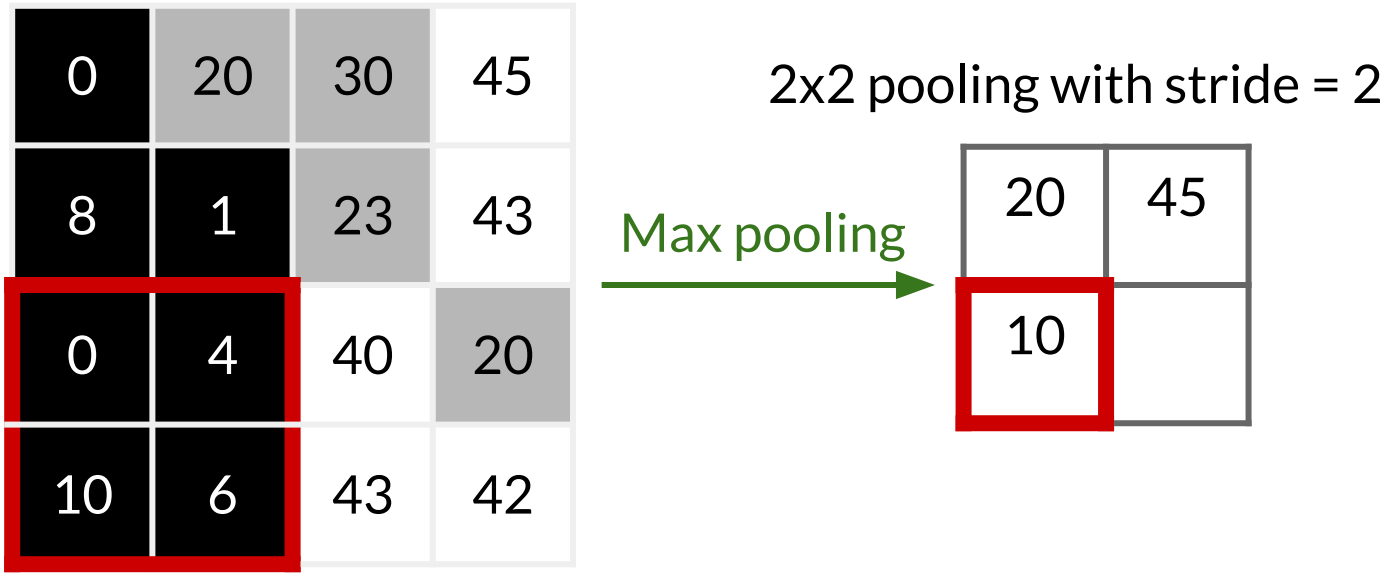
4x4 input to 2x2 output

Max Pooling



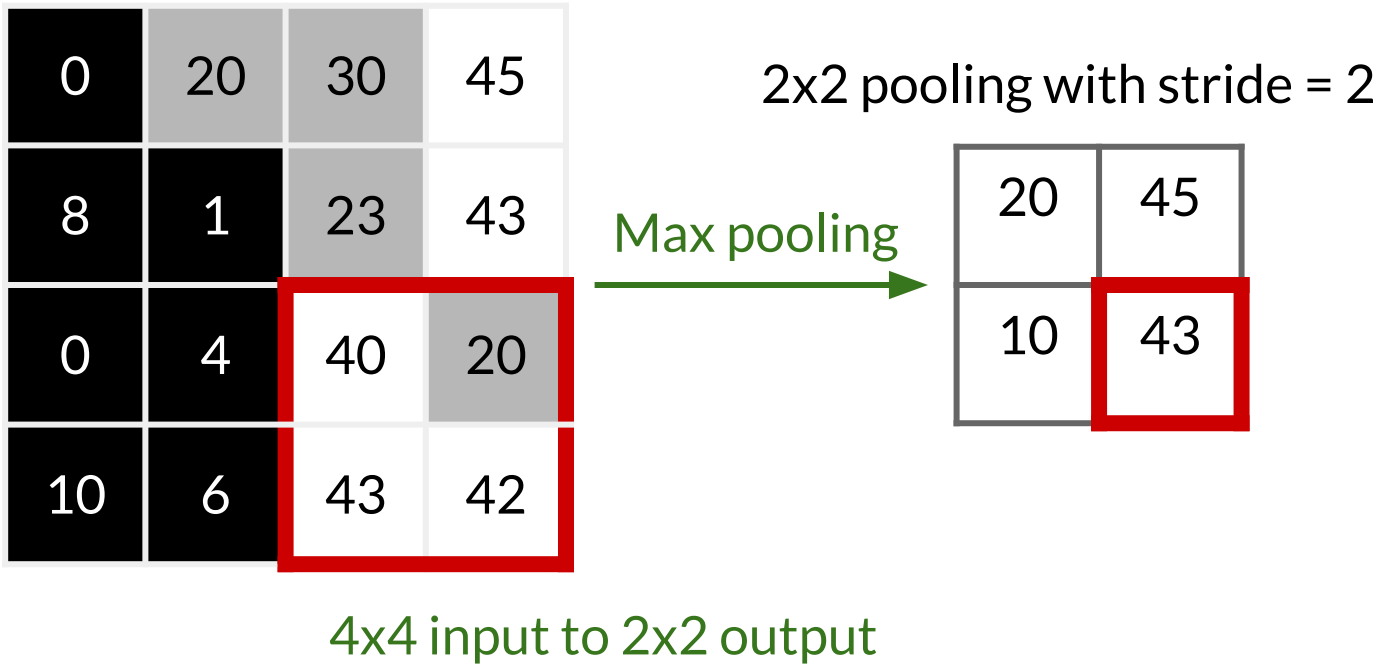
4x4 input to 2x2 output

Max Pooling



4x4 input to 2x2 output

Max Pooling


0
20
30
45
8
1
23
43
0
4
40
20
10
6
43
42
Max pooling
2x2 pooling with stride = 2
20
45
10
43
4x4 input to 2x2 output

Max Pooling

to get best/most important features

0	20	30	45
8	1	23	43
0	4	40	20
10	6	43	42

4x4 input to 2x2 output

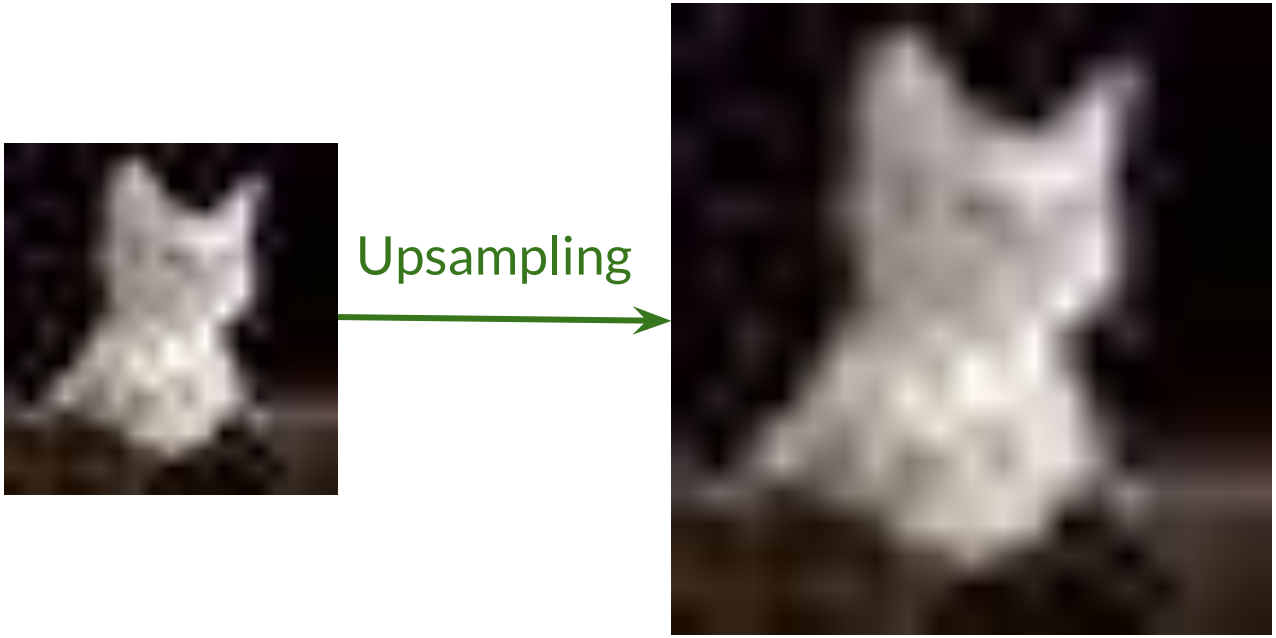
Max pooling

2x2 pooling with stride = 2

20	45
10	43

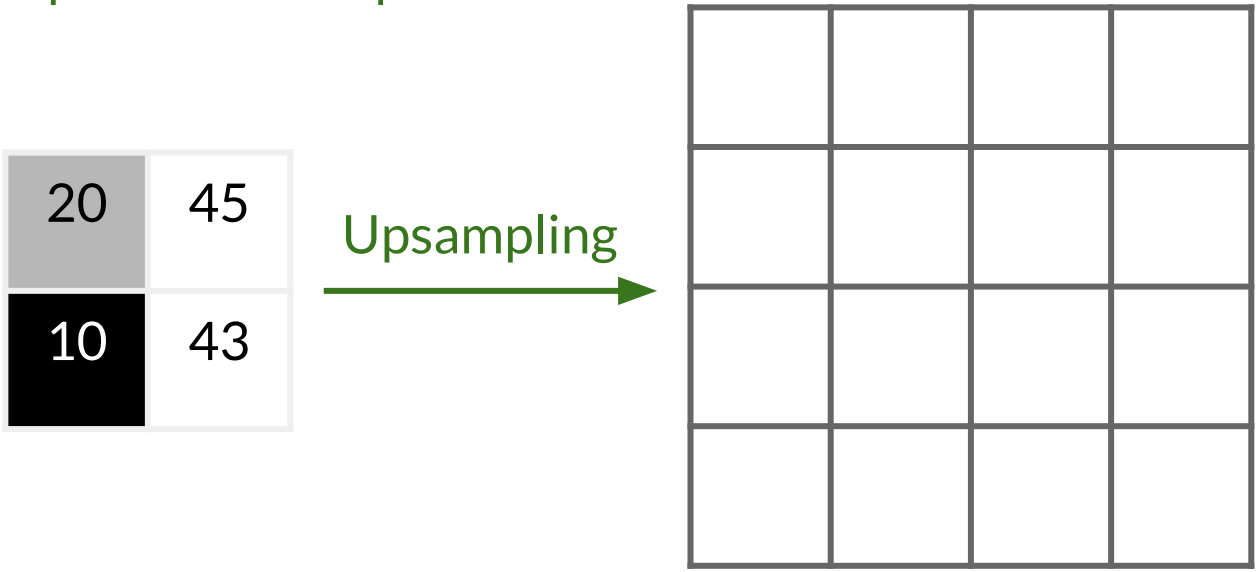
Other types include:
1. Average pooling
2. Min pooling

Upsampling



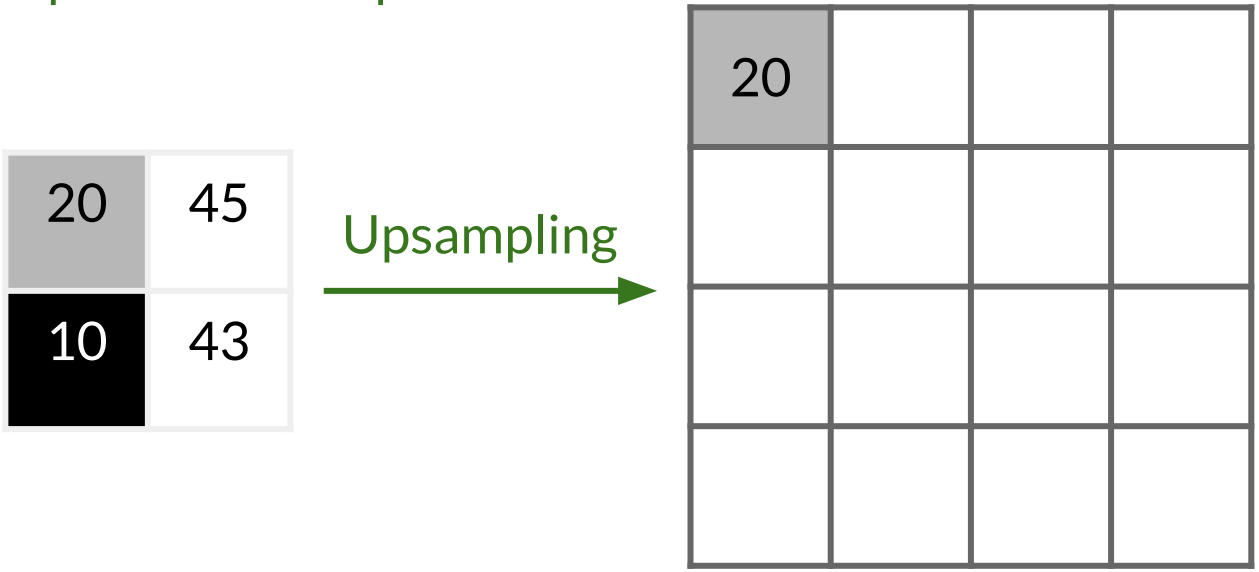
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



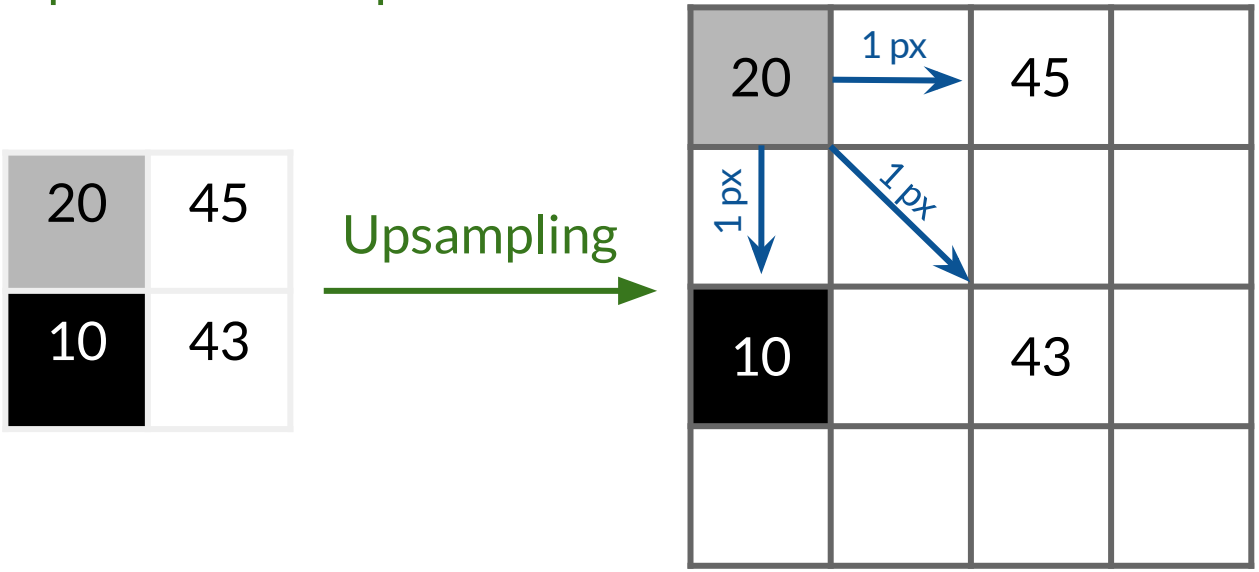
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



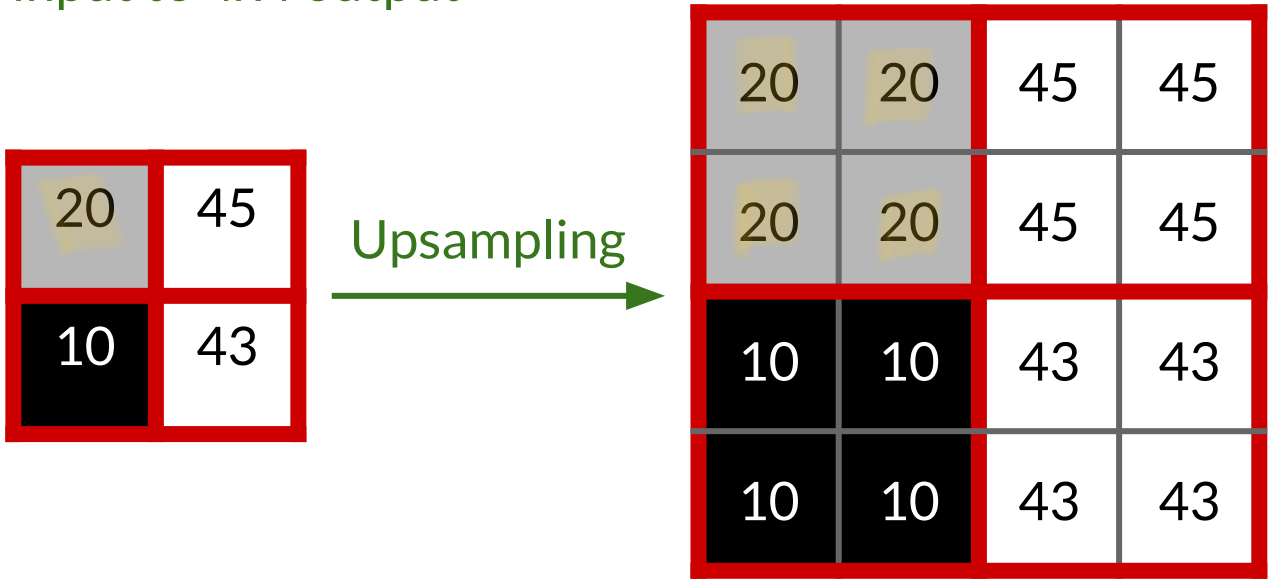
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



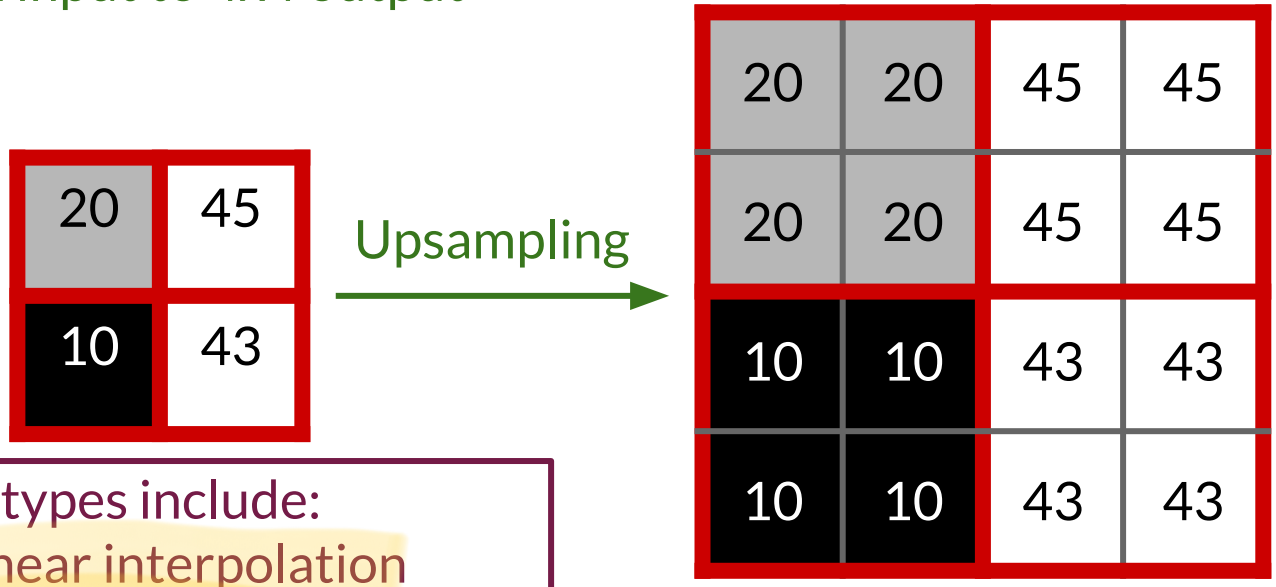
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



Upsampling: Nearest Neighbors

2x2 input to 4x4 output

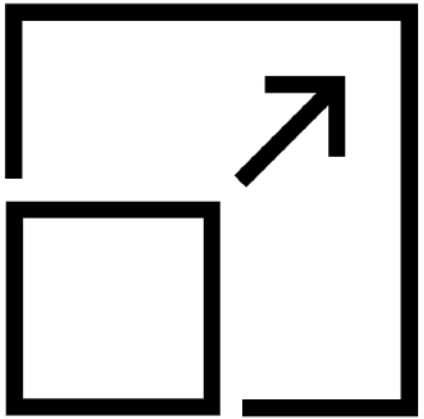


Other types include:

1. Linear interpolation
2. Bi-linear interpolation

Summary

- Pooling reduces the size of the input
- Upsampling increases the size of the input
- No learnable parameters!



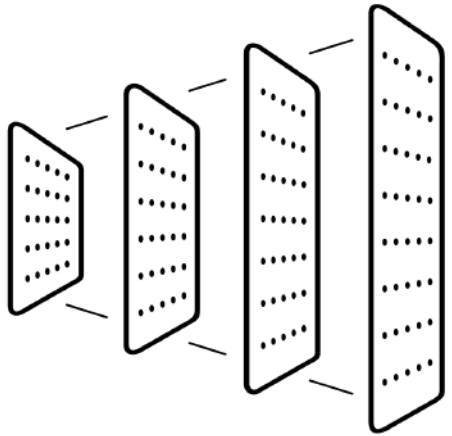
deeplearning.ai

Transposed Convolutions

Outline

- Transposed convolutions as an upsampling technique
- Issues with transposed convolutions

to enlarge input

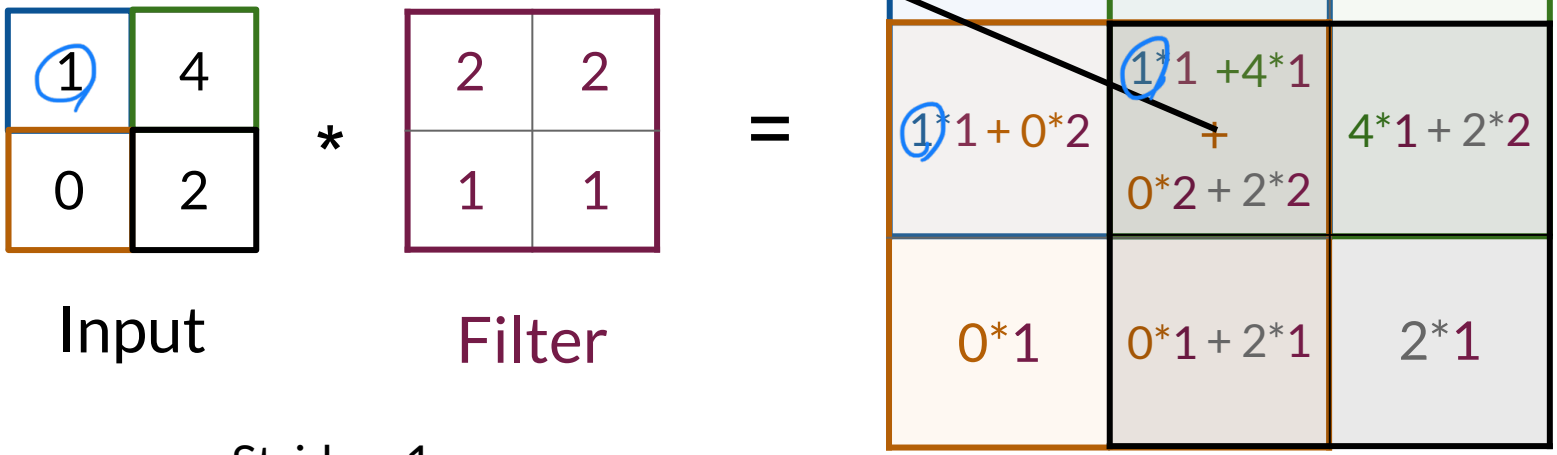


Transposed Convolution

2x2 → 3x3

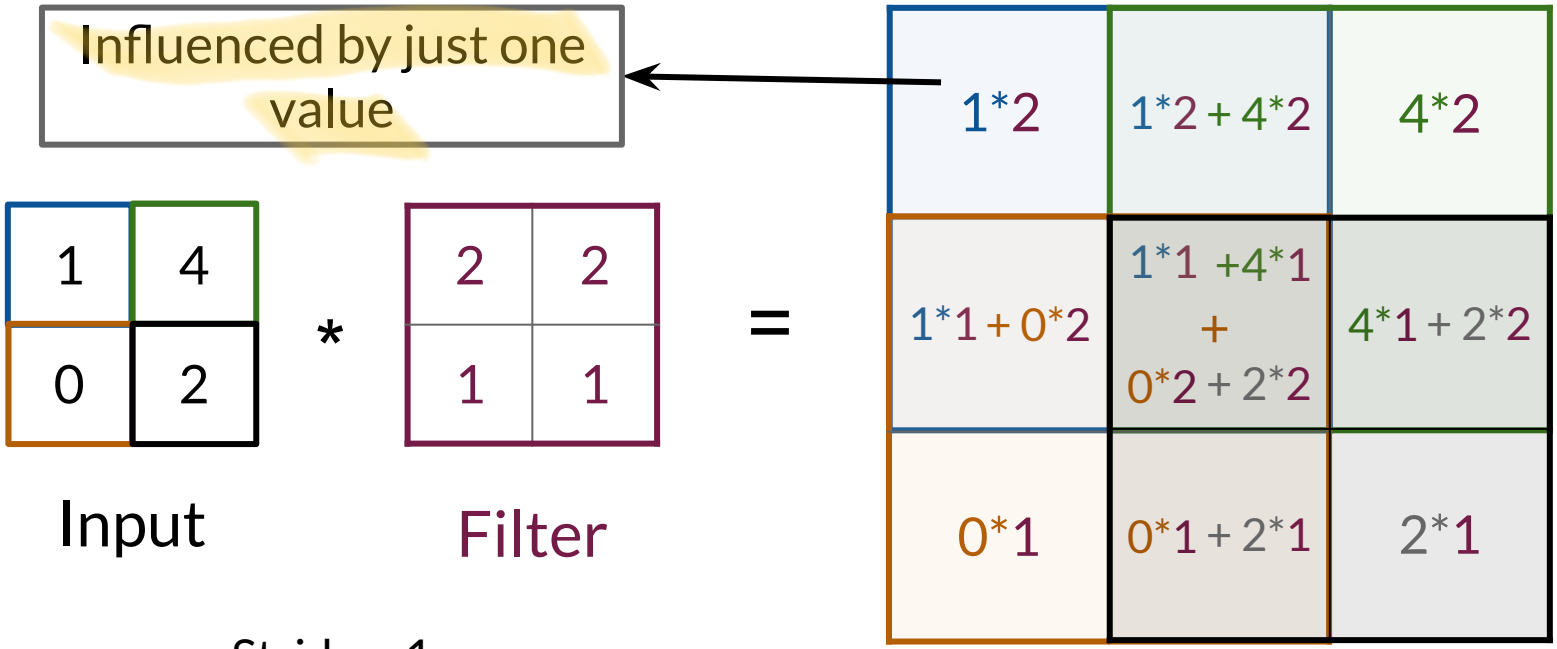
For overlaps

Influence from every value in the input



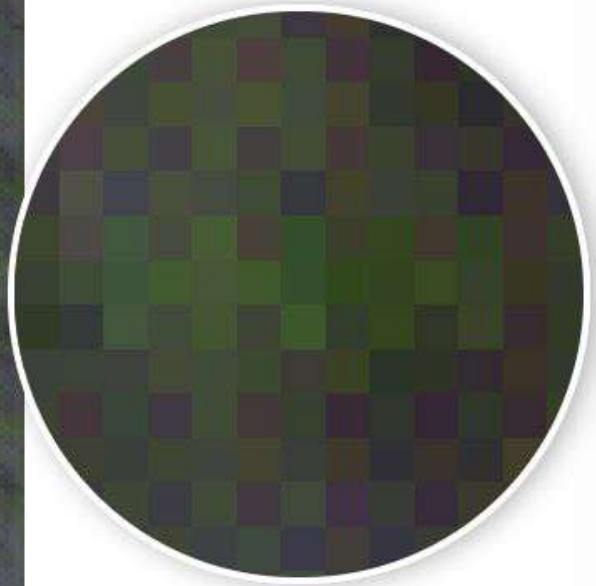
Stride = 1

Transposed Convolution



Stride = 1

The Problems with Transposed Convolution



Checkerboard Pattern

Available from: http://doi.org/10.23915/distill.00003

Summary

- Transposed convolutions upsample
- They have learnable parameters
- Problem: results have a checkerboard pattern