

Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see

<https://creativecommons.org/licenses/by-sa/2.0/legalcode>

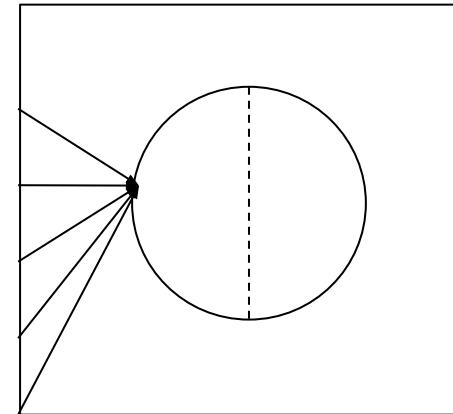


deeplearning.ai

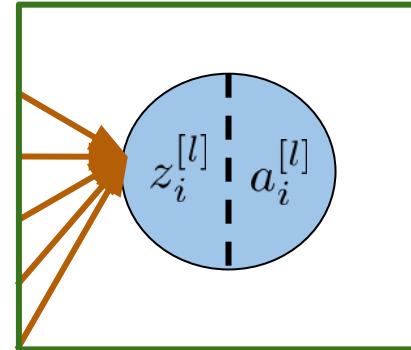
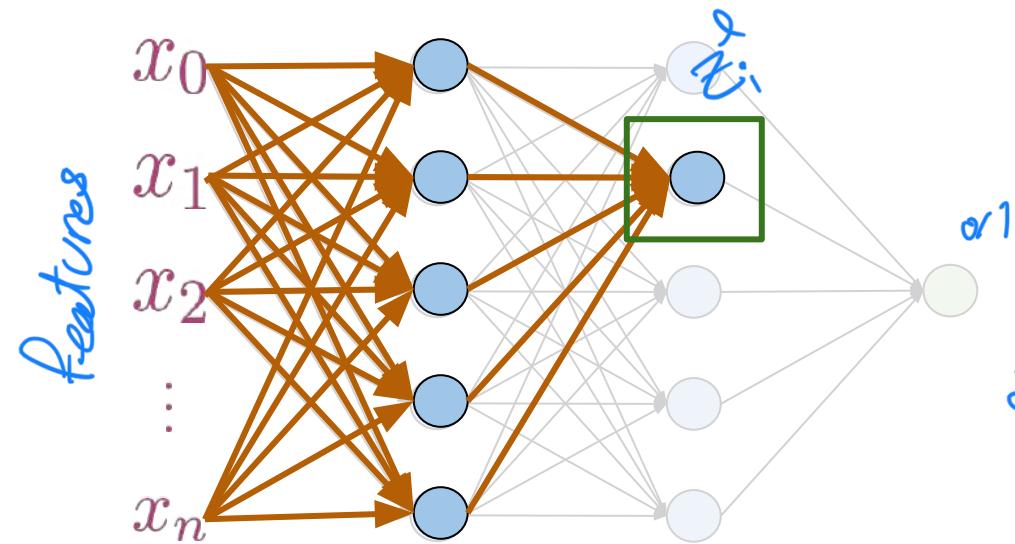
Activations (Basic Properties)

Outline

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



Activations



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

layer
prev layer

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

actv
function

Differentiable
non-linear
function

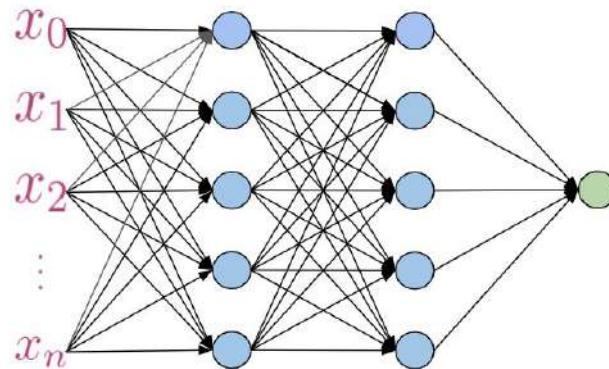
Activations

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

Differentiable
non-linear
function

1. Differentiable for backpropagation

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



≡

$$WX + b$$

Linear
regression

if linear activ func

Summary

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

$f(x)$



deeplearning.ai

Common Activation Functions

Outline

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

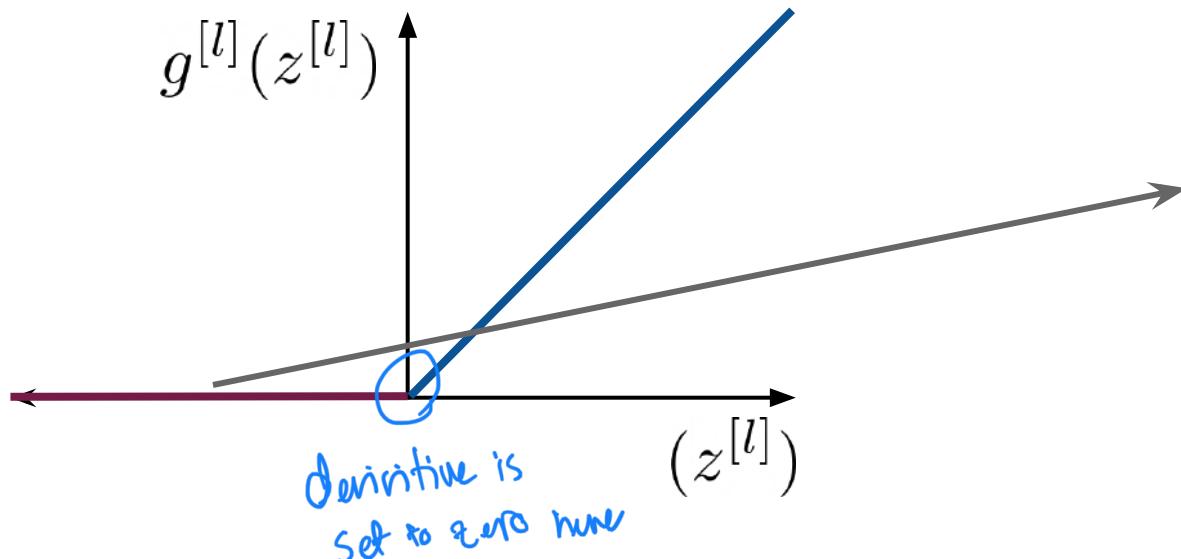


Activations: ReLU

No Neg Value

ReLU = Rectified Linear Unit

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

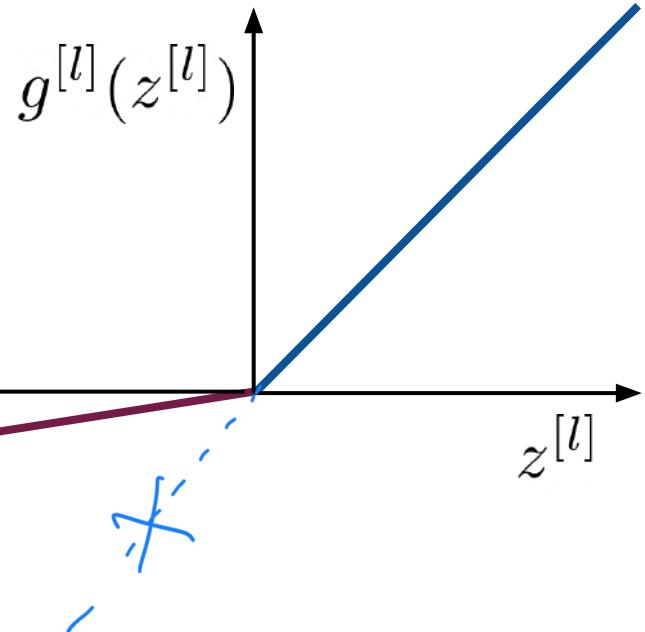


Dying ReLU
problem

Stops learning
because of zeros



Activations: Leaky ReLU



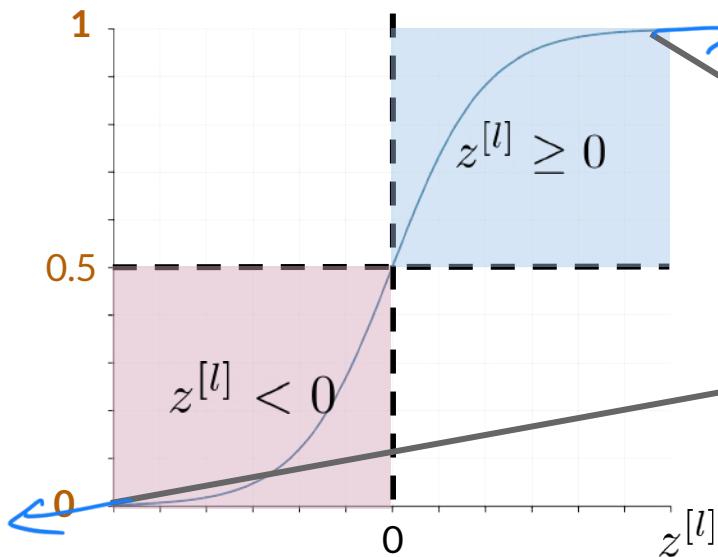
$$g^{[l]}(z^{[l]}) = \max(a z^{[l]}, z^{[l]})$$

leak / slope
less than 1
like orl

Solves the dying
ReLU problem

Activations: Sigmoid

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



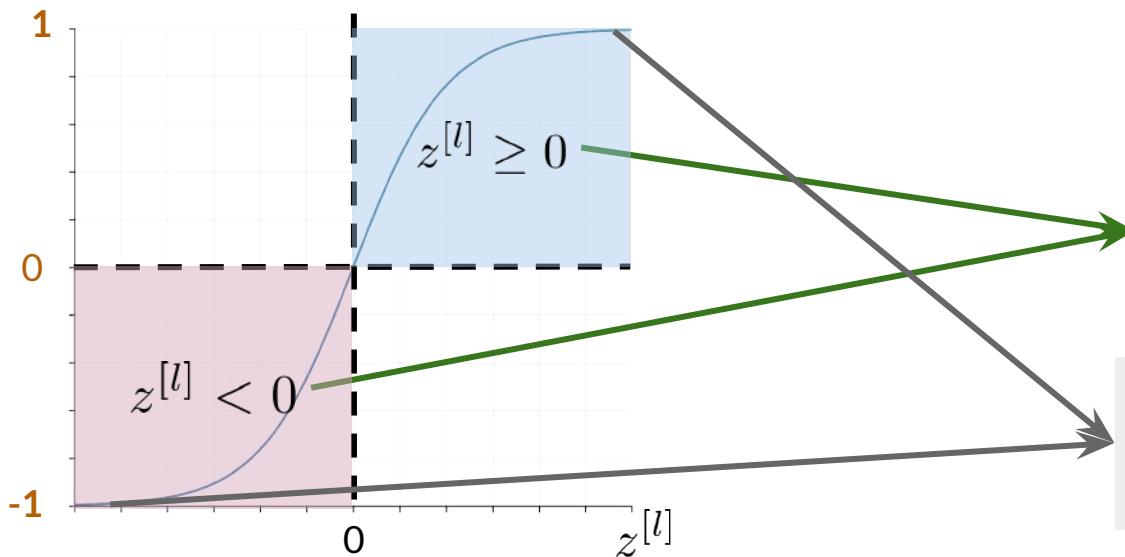
for binary classification
prob at 0/1

Values between 0
and 1

Vanishing gradient
and saturation
problems

Activations: Tanh

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



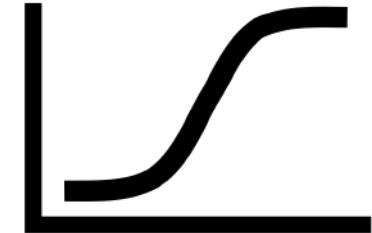
Values between -1 and 1

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

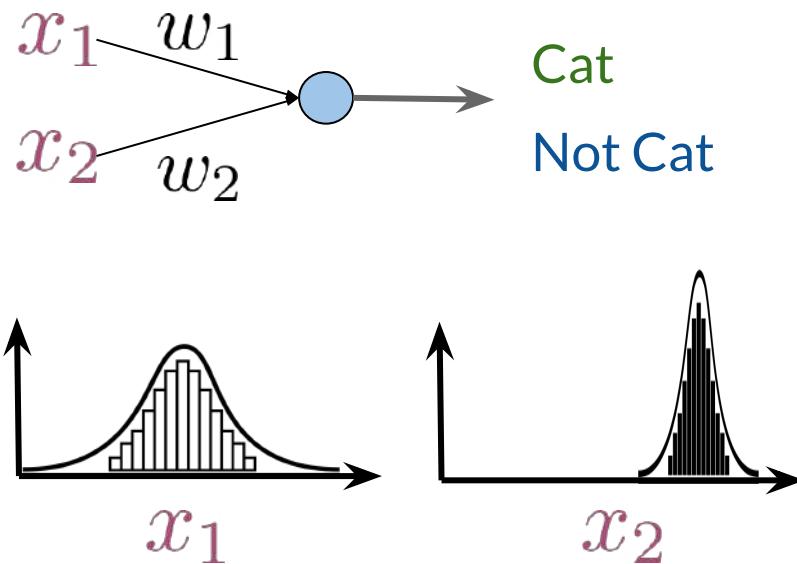
Batch Normalization (Explained)

Outline

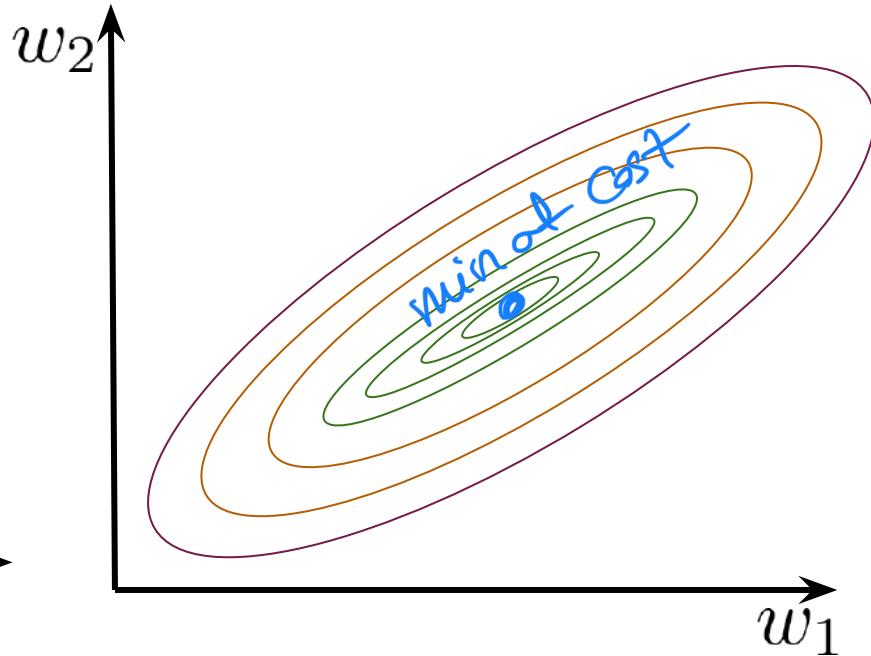
- How normalization helps models
- Internal covariate shift
- Batch normalization → stabilize training



Different Distributions

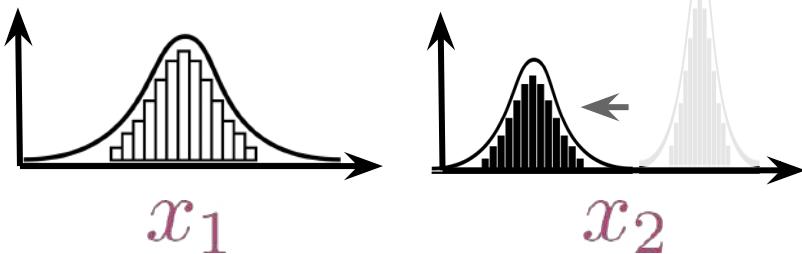
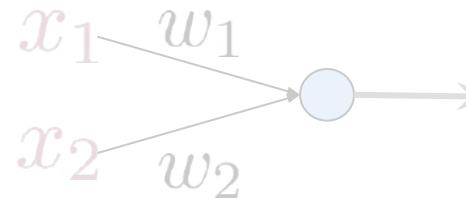


of features Cost function

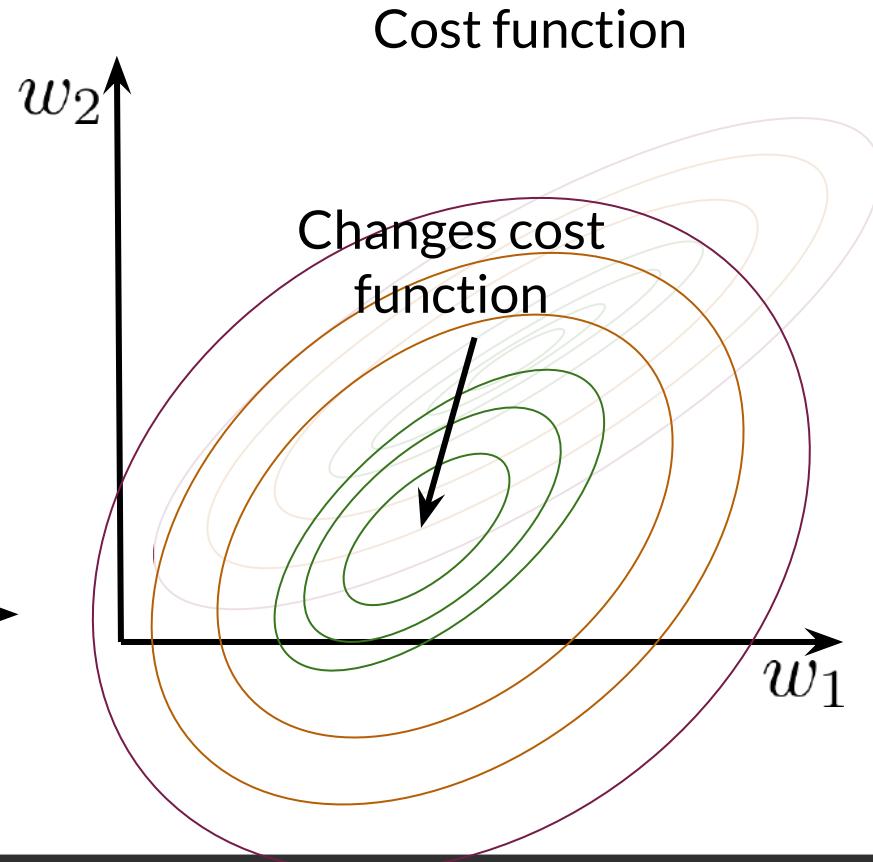


happens between train/ test sets

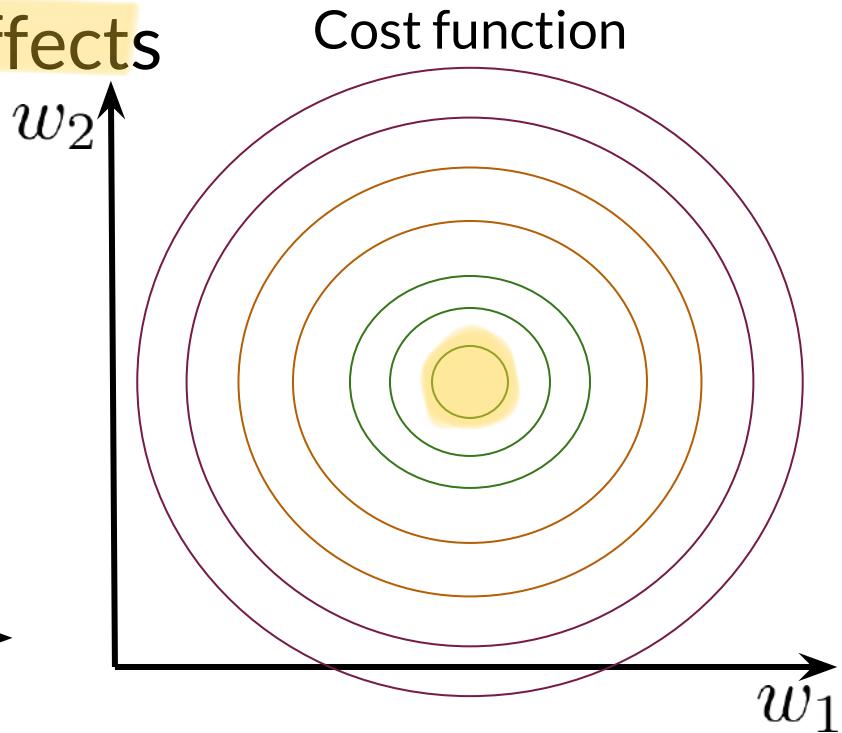
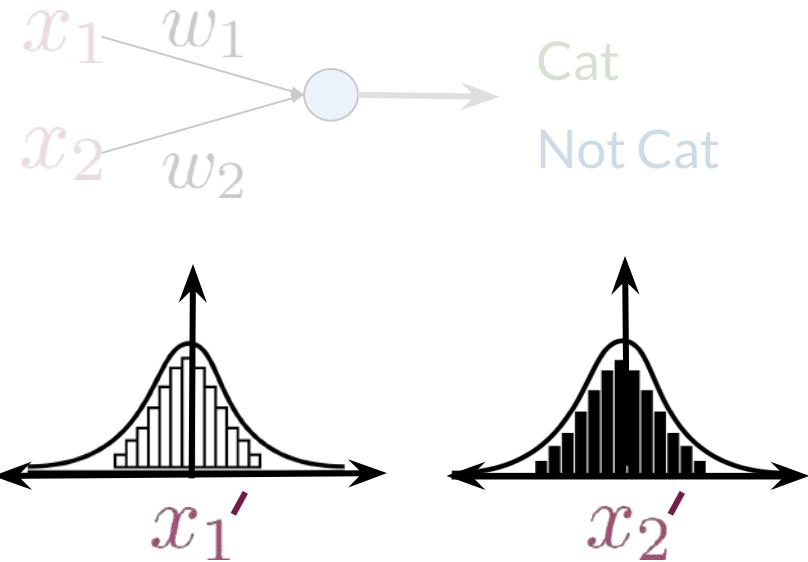
Covariate Shift



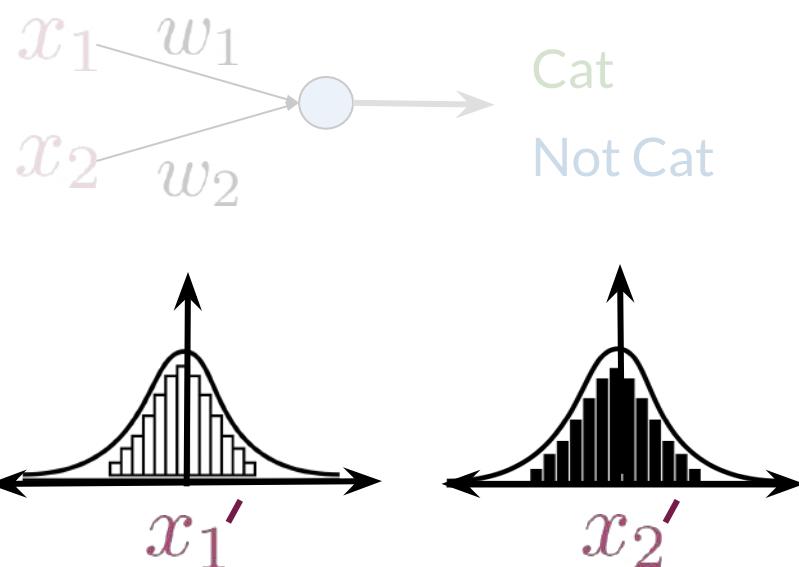
Cat
Not Cat
Change in data distribution



Normalization and Its Effects



Normalization and Its Effects



$$x_1' \quad x_2'$$

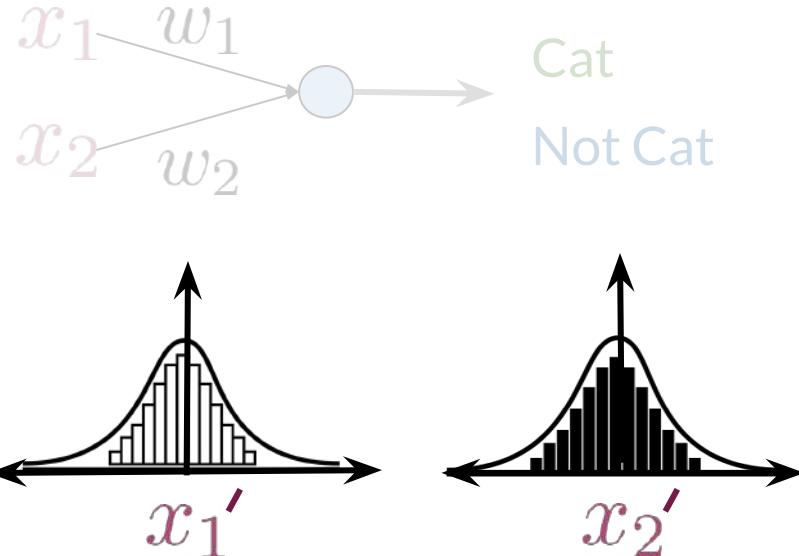
Around mean at 0
and std. at 1

Training data uses
batch stats

Test data uses
training stats

test gets closer
to train data

Normalization and Its Effects



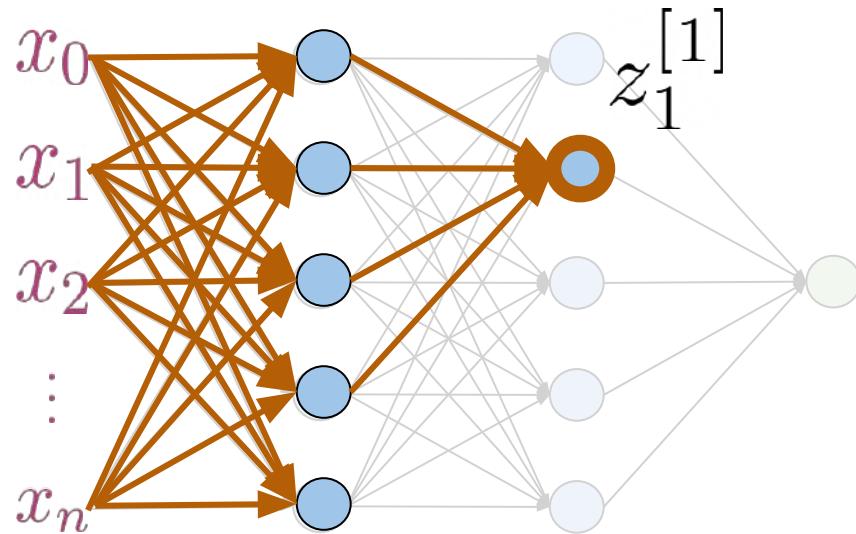
x_1' x_2'

Around mean at 0
and std. at 1



Reduction of
covariate shift

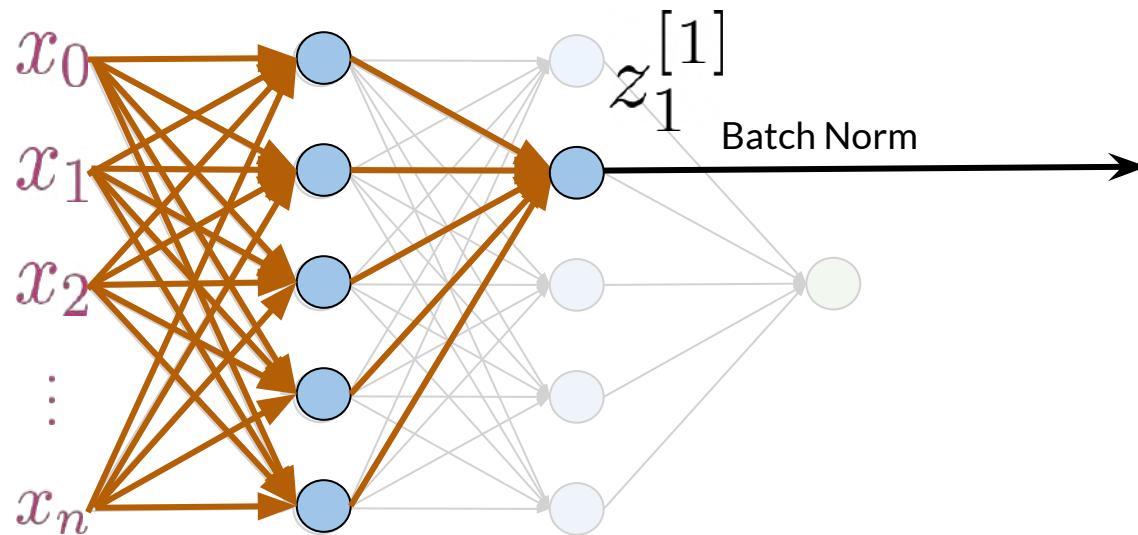
Internal Covariate Shift in hidden layers



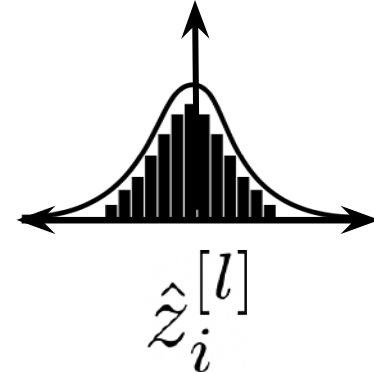
Changes in
weights

Changes in
activation
distribution

Batch Normalization

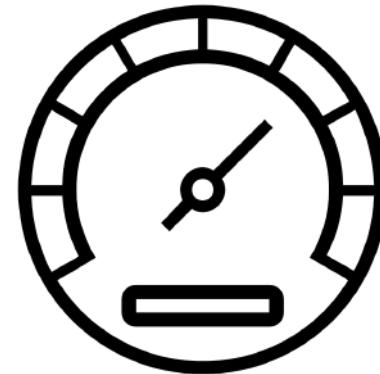


Normalizes the
input for each
neuron



Summary

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





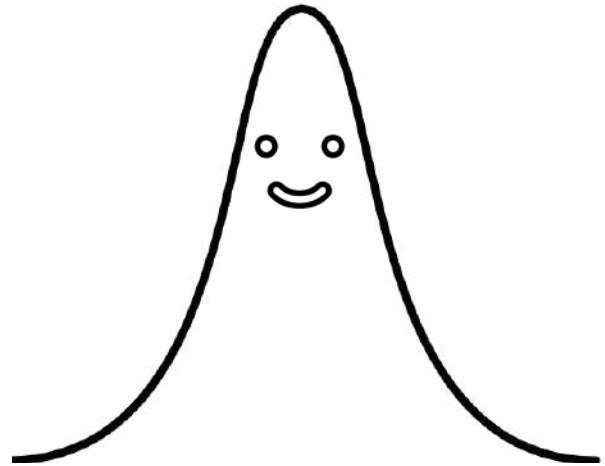
deeplearning.ai

Batch Normalization (Procedure)

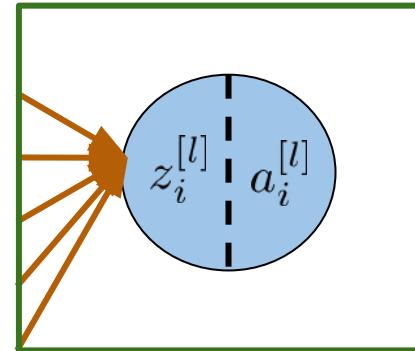
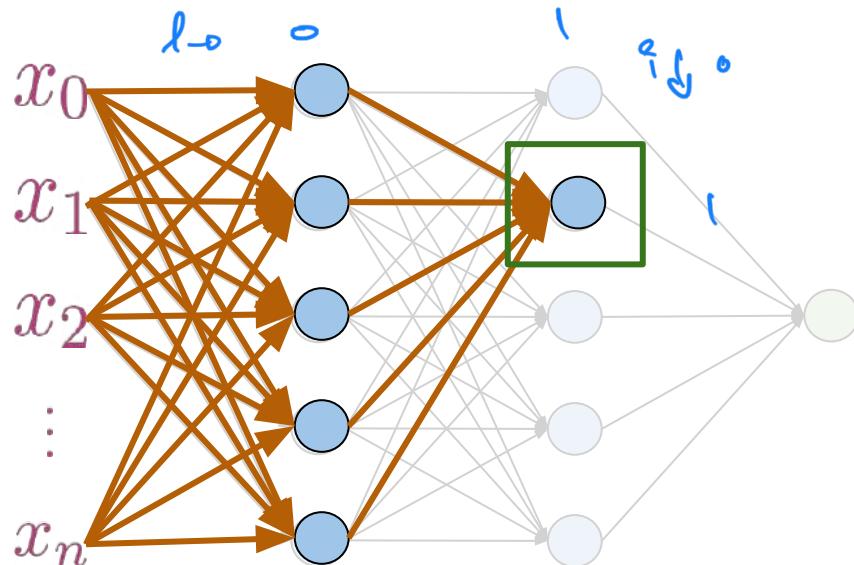
Outline

- Batch norm for training
- Batch norm for testing

happily Normalized !



Batch Normalization: Training



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

For every example in the batch
whole batch

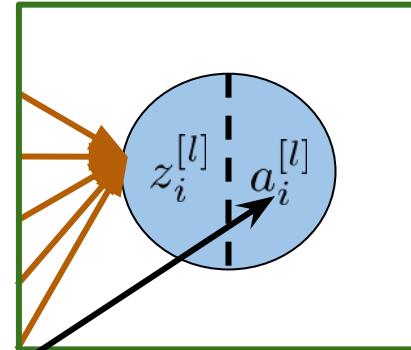
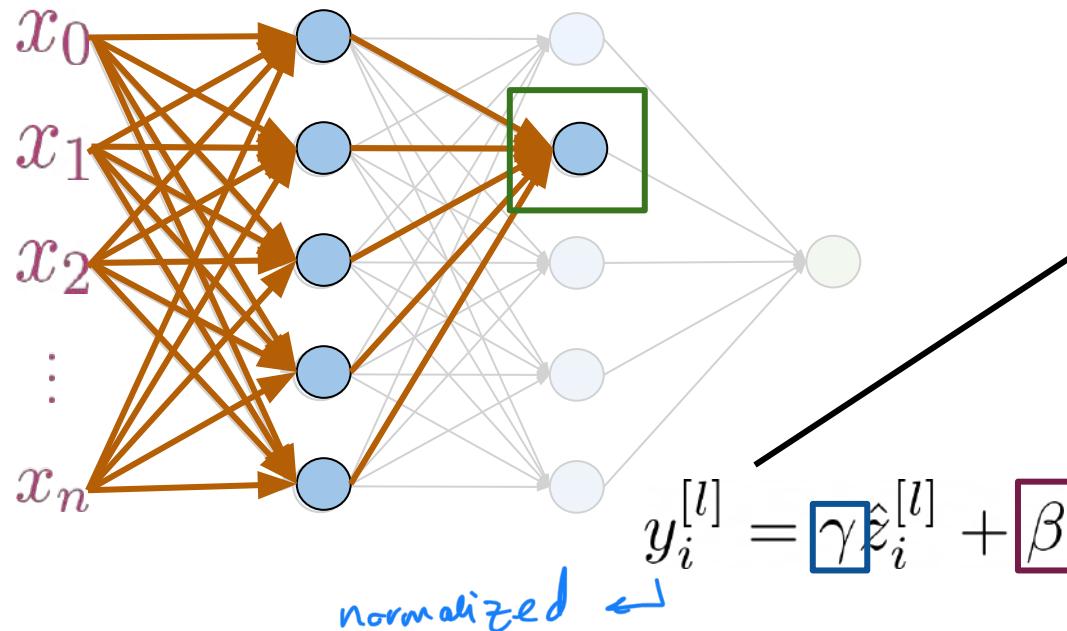
$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma_{z_i^{[l]}}^2 + \epsilon}}$$

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

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

whole batch

Batch Normalization: Training



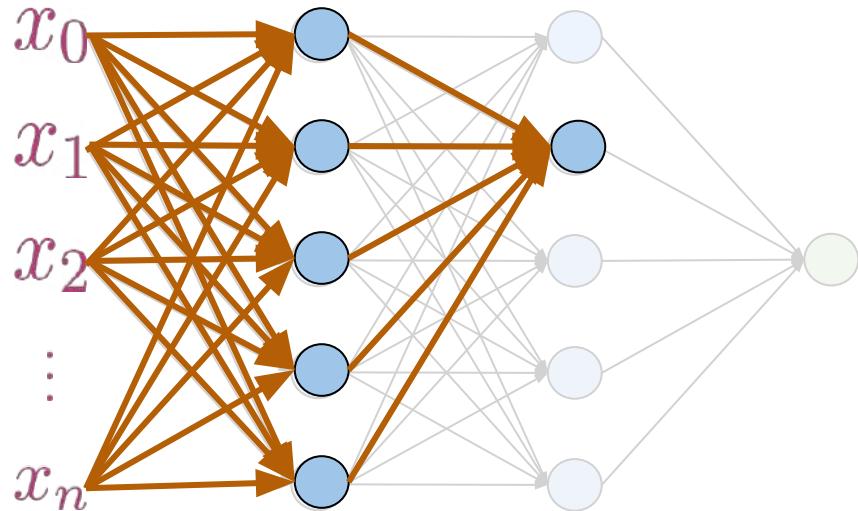
$$\hat{z}_i^{[l]} = \frac{z_i^{[l]} - \mu_{z_i^{[l]}}}{\sqrt{\sigma_{z_i^{[l]}}^2 + \epsilon}}$$

Shift factor

Scale Factor

Learnable
parameters to get
the optimal dist.

Batch Normalization: Test



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

Running mean and running std from training
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



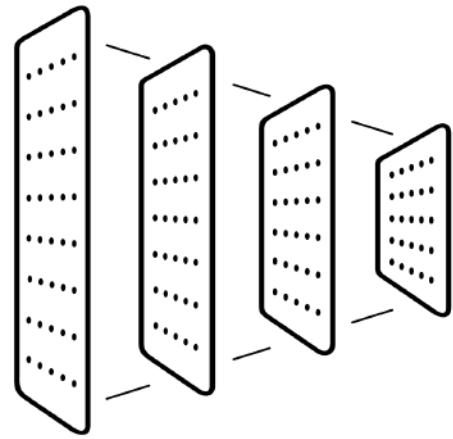


deeplearning.ai

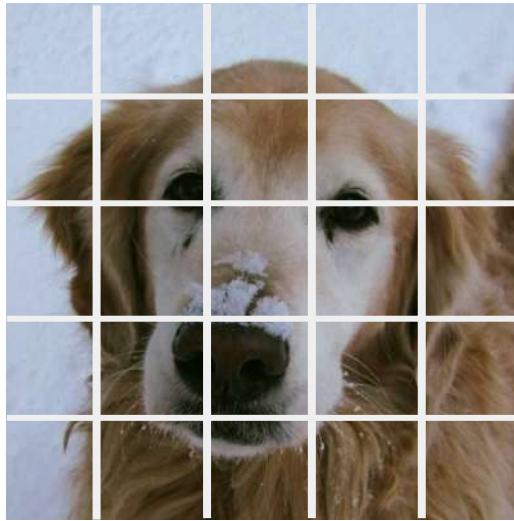
Review of Convolutions

Outline

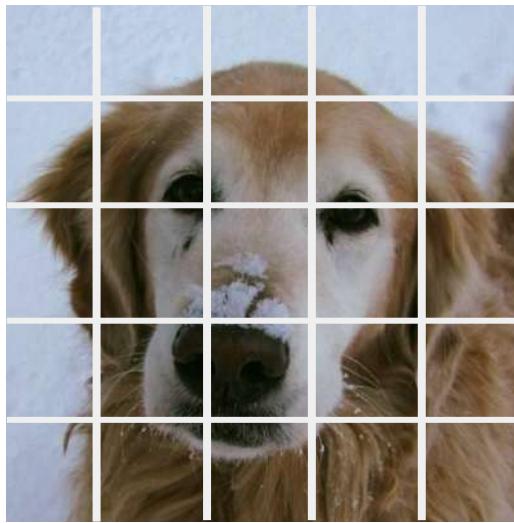
- What convolutions are
- How they work



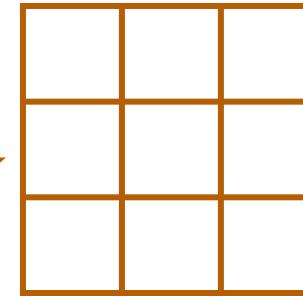
What is a convolution?



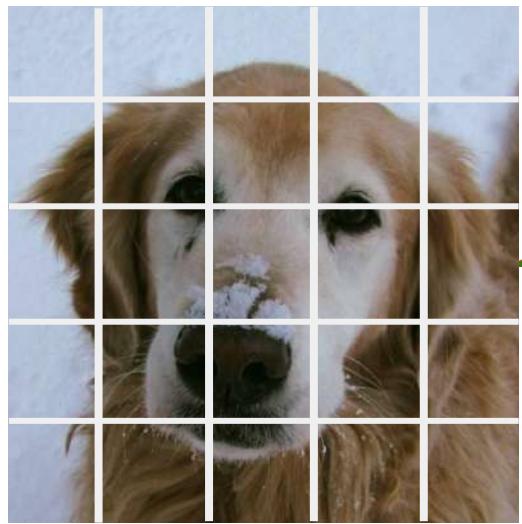
What is a convolution?



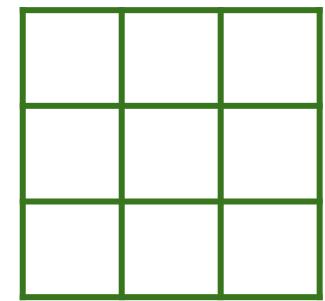
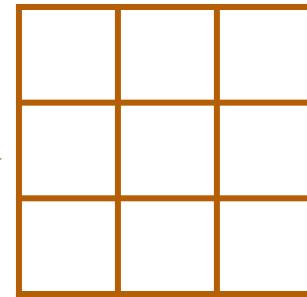
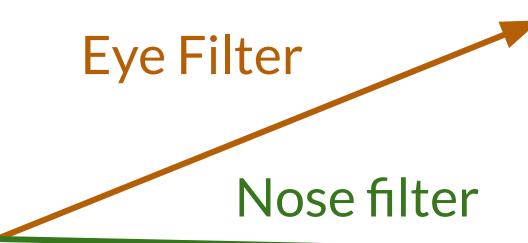
Eye Filter



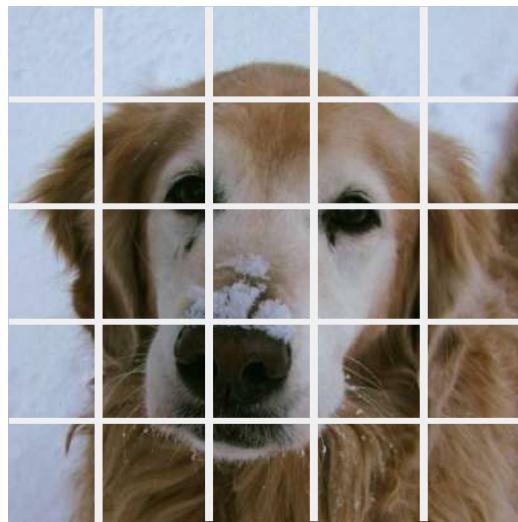
What is a convolution?



Eye Filter



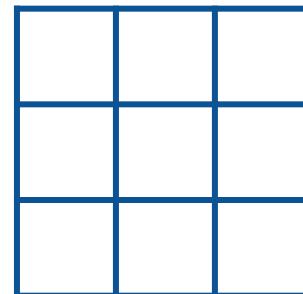
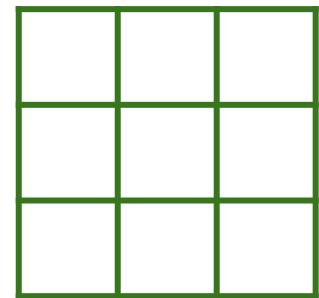
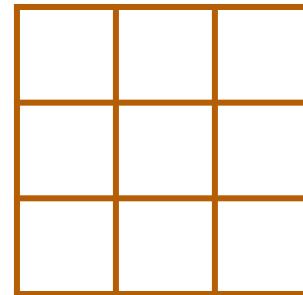
What is a convolution?



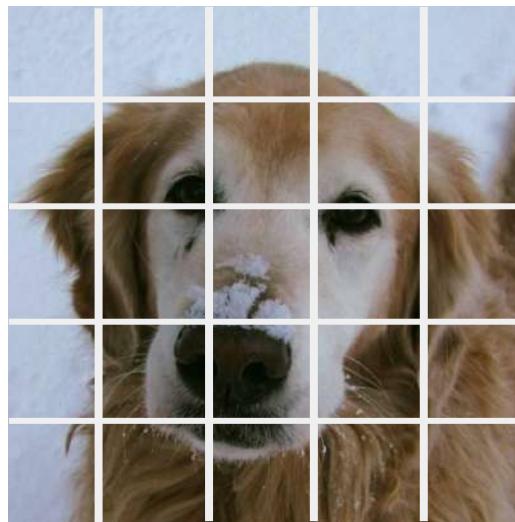
Eye Filter

Nose filter

Ear filter



What is a convolution?



Eye Filter

Nose filter

Ear filter

5	3	1
10	23	1
1	42	4

values learned
in training

32	2	24
25	12	66
3	45	2

2	21	5
23	45	12
3	32	4

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

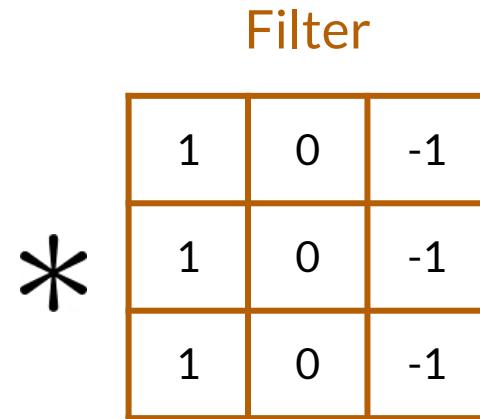
Grayscale image

0 (black) to 255 (white)

What is a convolution?

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

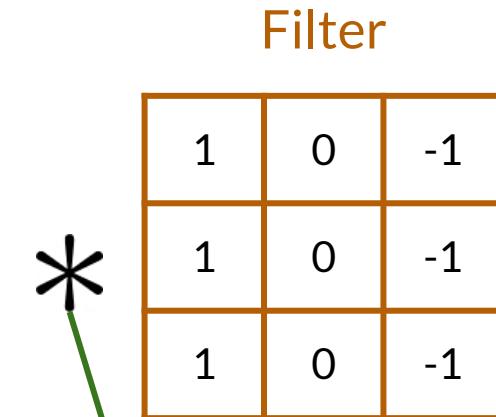


0 (black) to 255 (white)

What is a convolution?

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



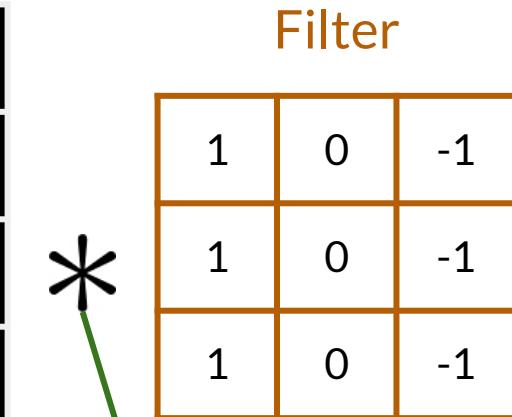
Convolution

0 (black) to 255 (white)

What is a convolution?

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



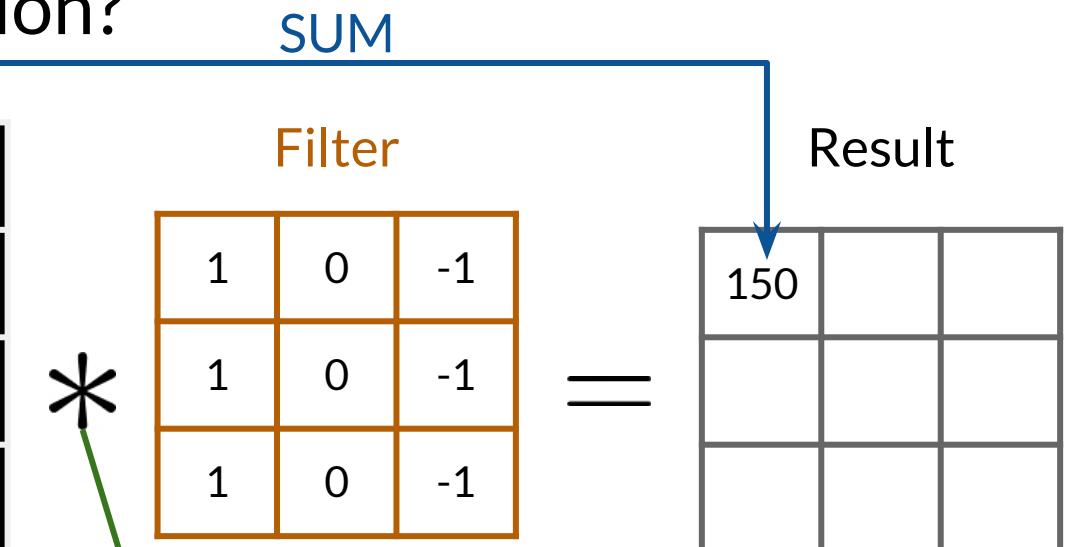
Convolution

0 (black) to 255 (white)

What is a convolution?

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



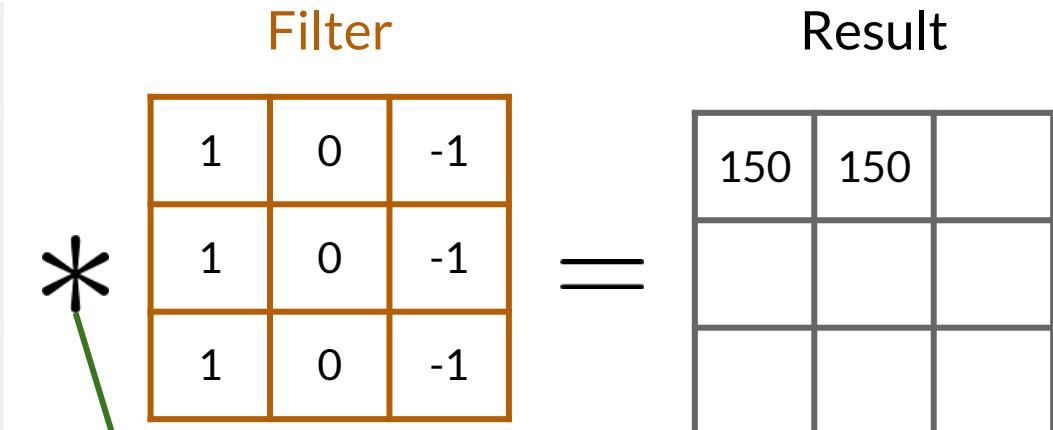
Convolution

0 (black) to 255 (white)

What is a convolution?

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

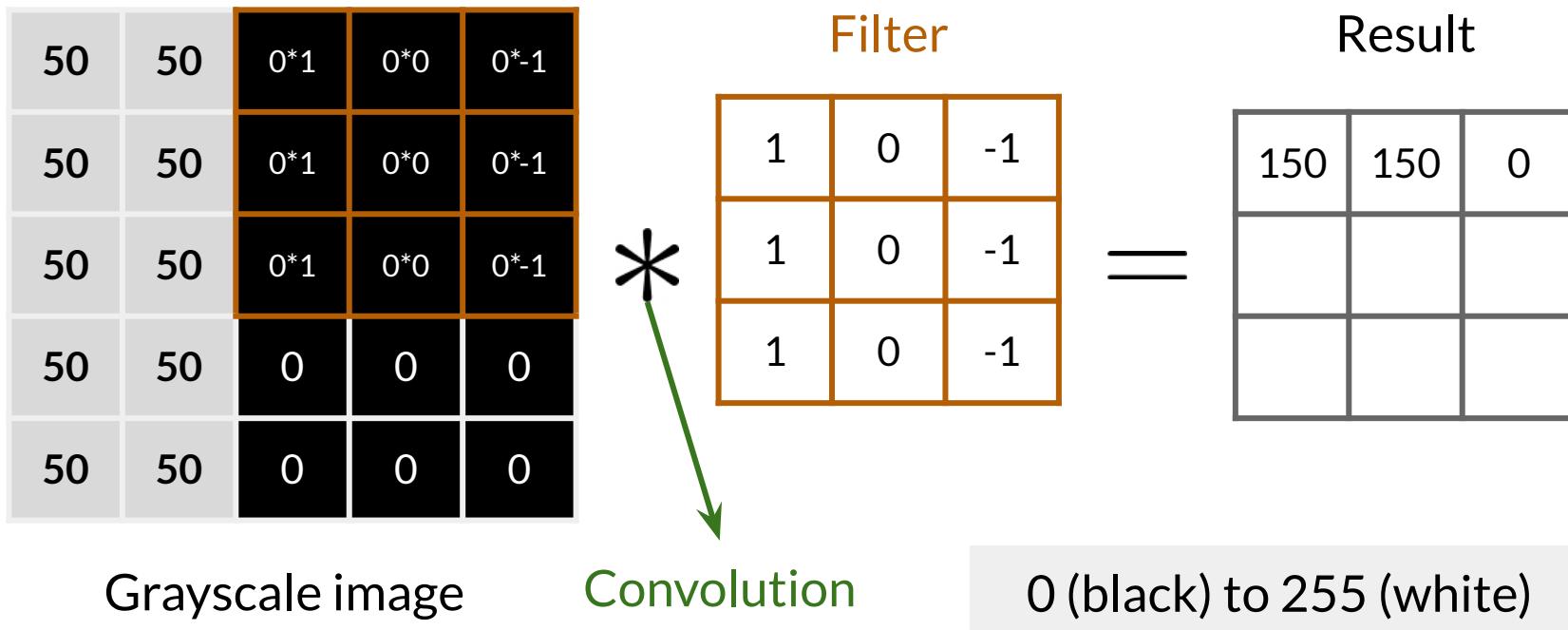
Grayscale image



Convolution

0 (black) to 255 (white)

What is a convolution?



What is a convolution?

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

Convolution

Result

150	150	0
150		

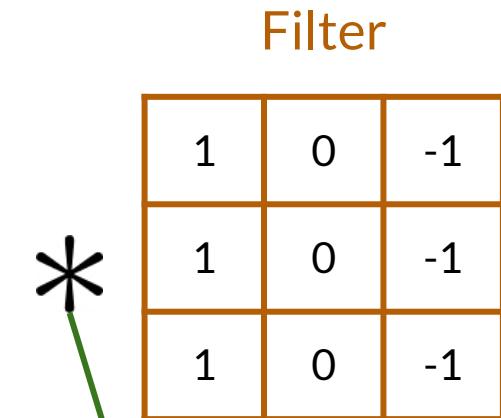
=

0 (black) to 255 (white)

What is a convolution?

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



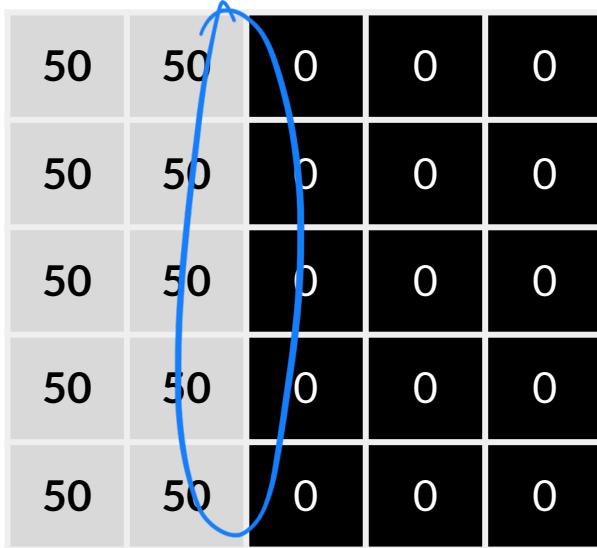
Convolution

Result

$$= \begin{matrix} 150 & 150 & 0 \\ 150 & 150 & \end{matrix}$$

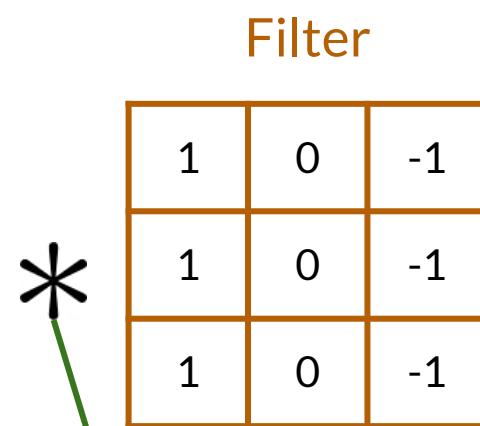
0 (black) to 255 (white)

What is a convolution?



Grayscale image

Convolution

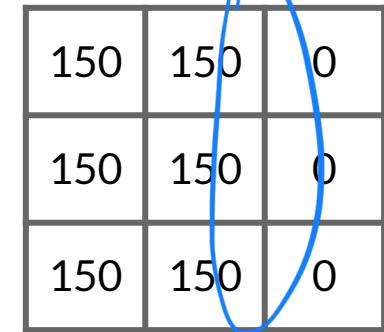


does vertical line detection



Filter

Result



=

0 (black) to 255 (white)

Summary

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



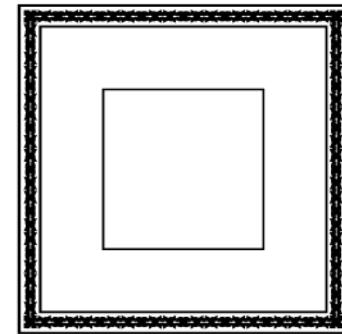


deeplearning.ai

Padding and Stride

Outline

- Padding and stride
- The intuition behind padding



Stride

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

*

Filter

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

Grayscale image

Stride

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

*

Result

=

The diagram illustrates a convolution operation. On the left, a 5x5 grayscale image is shown with its dimensions (50x1, 50x0, etc.) and values (50, 50). A 3x3 filter is applied to the image, indicated by an asterisk (*). The result of the convolution is 150, shown in the first cell of a 3x3 result matrix.

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

150		

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

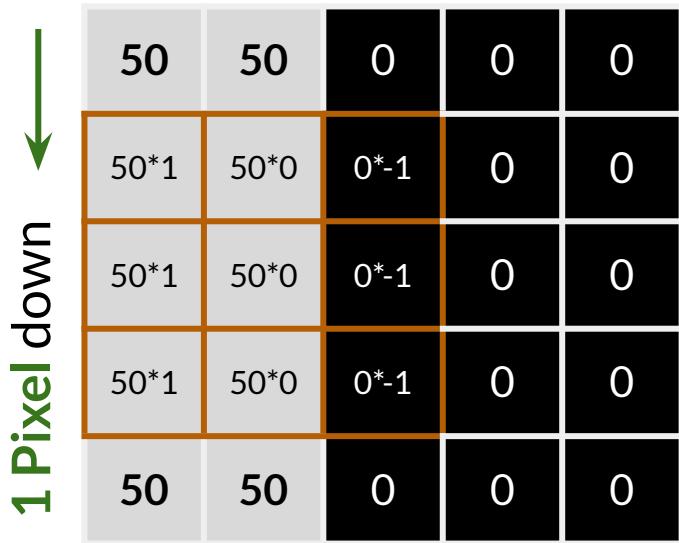
Result

150	150	0



Stride

→ 1 Pixel to the right



*

Filter

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

=

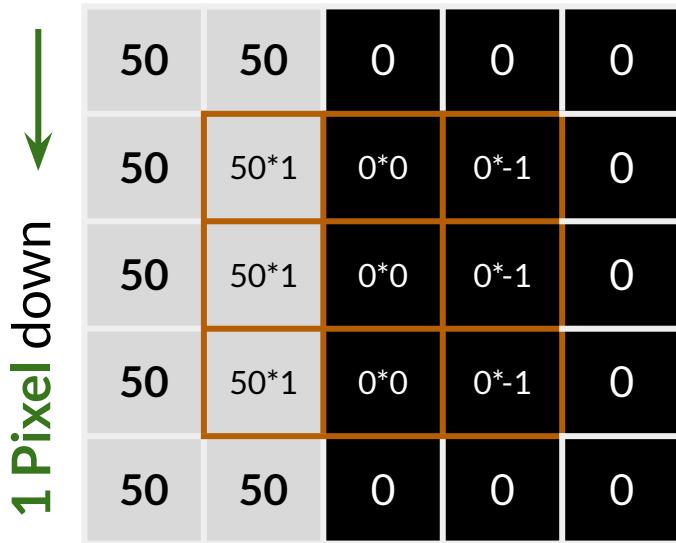
Result

150	150	0
150		

Grayscale image

Stride

→ 1 Pixel to the right



Grayscale image

*

Filter

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

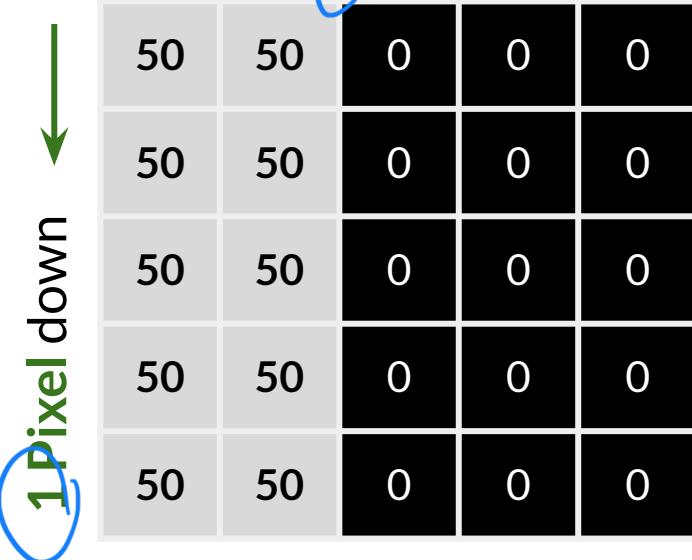
=

Result

150	150	0
150	150	

Stride

→ 1 Pixel to the right



Grayscale image

*

Filter

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

=

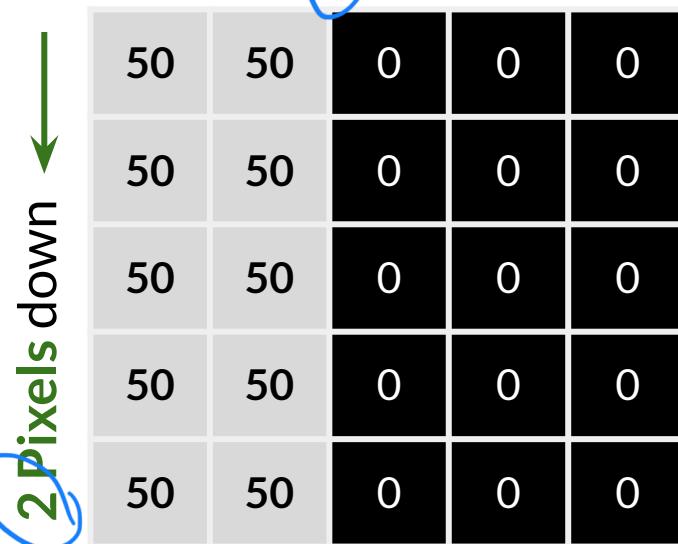
Result

150	150	0
150	150	0
150	150	0

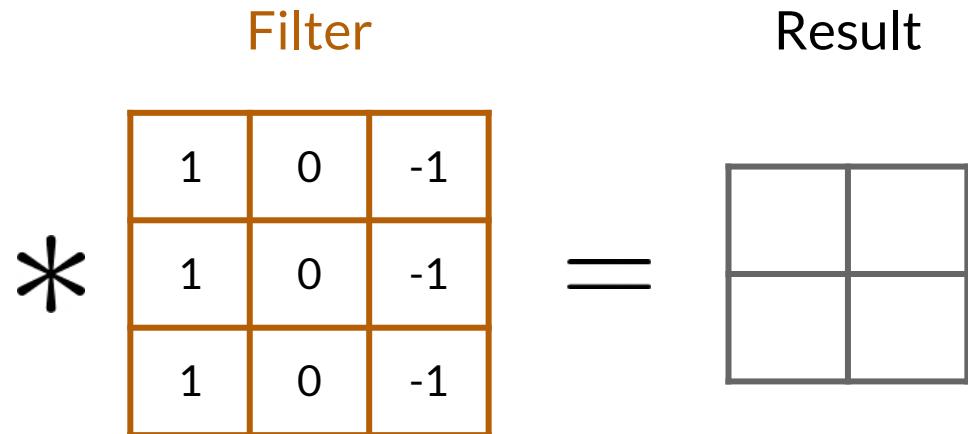
Stride = 1

Stride

→ **2 Pixels** to the right



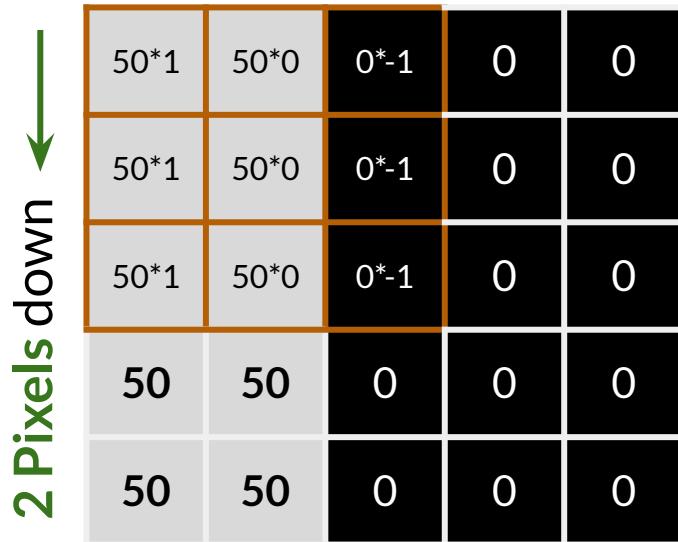
Grayscale image



Stride = 2

Stride

→ 2 Pixels to the right



Filter

*

A convolution operation diagram showing a 3x3 input image being multiplied by a 3x3 filter. The result is a single value of 150. The input image has values: Row 1: 50*1, 50*0, 0*-1, 0, 0. Row 2: 50*1, 50*0, 0*-1, 0, 0. Row 3: 50*1, 50*0, 0*-1, 0, 0. The filter has values: Row 1: 1, 0, -1. Row 2: 1, 0, -1. Row 3: 1, 0, -1. An equals sign follows the multiplication symbol.

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

=

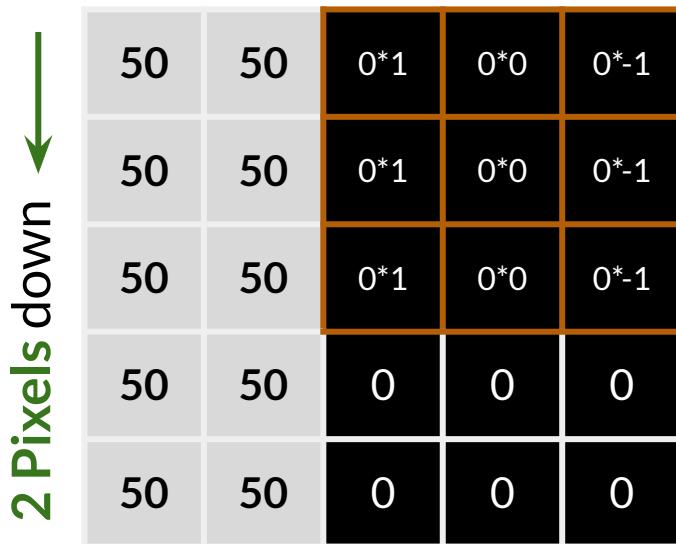
150	

Result

Stride = 2

Stride

→ 2 Pixels to the right



Grayscale image

Filter

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$$

*

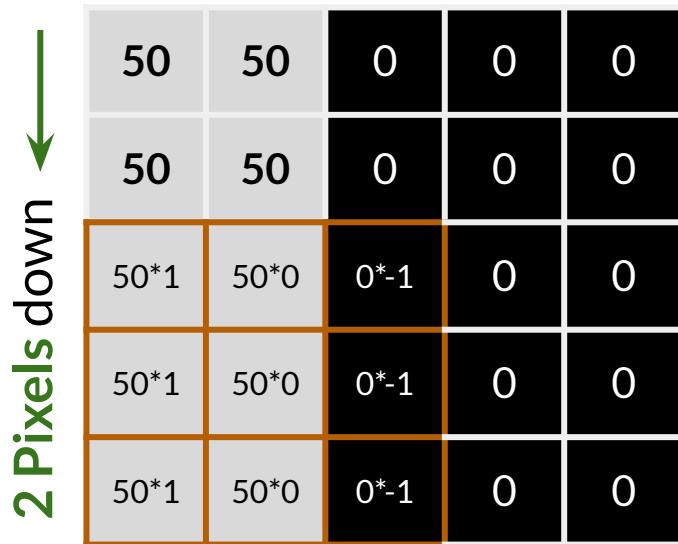
=

150	0

Stride = 2

Stride

→ 2 Pixels to the right



Grayscale image

Filter

*

The convolution operation is shown as $\text{Grayscale image} * \text{Filter} = \text{Result}$. The result is a 2x2 matrix with values 150 and 0. The filter is a 3x3 matrix with values 1, 0, -1 in each row.

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

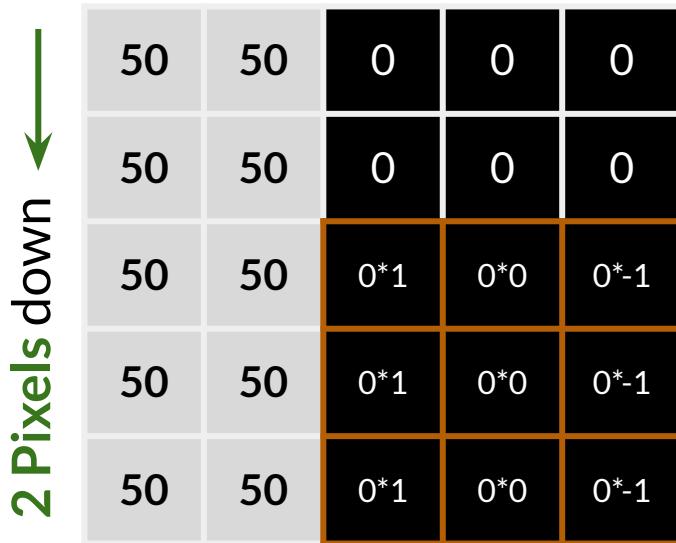
=

150	0
150	

Stride = 2

Stride

→ 2 Pixels to the right



Grayscale image

Filter

*

The diagram shows a convolution operation between a 5x3 input image and a 3x3 filter. The input image has a stride of 2. The result is a 2x2 output matrix with values 150 and 0.

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

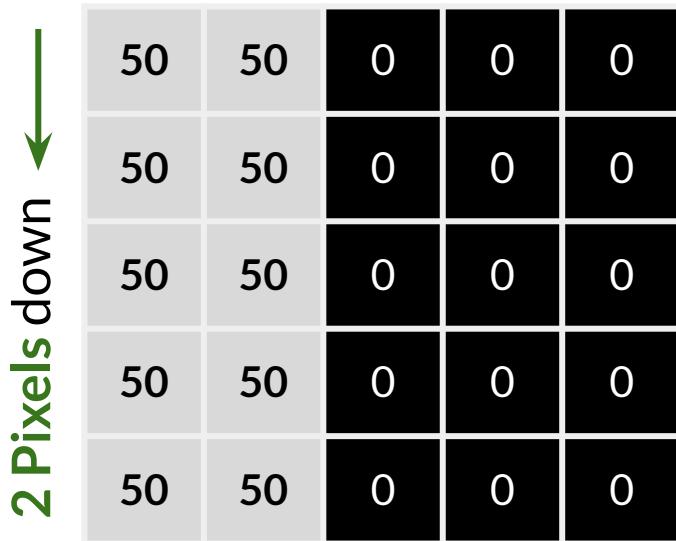
=

150	0
150	0

Stride = 2

Stride

→ 2 Pixels to the right



Grayscale image

Filter

*

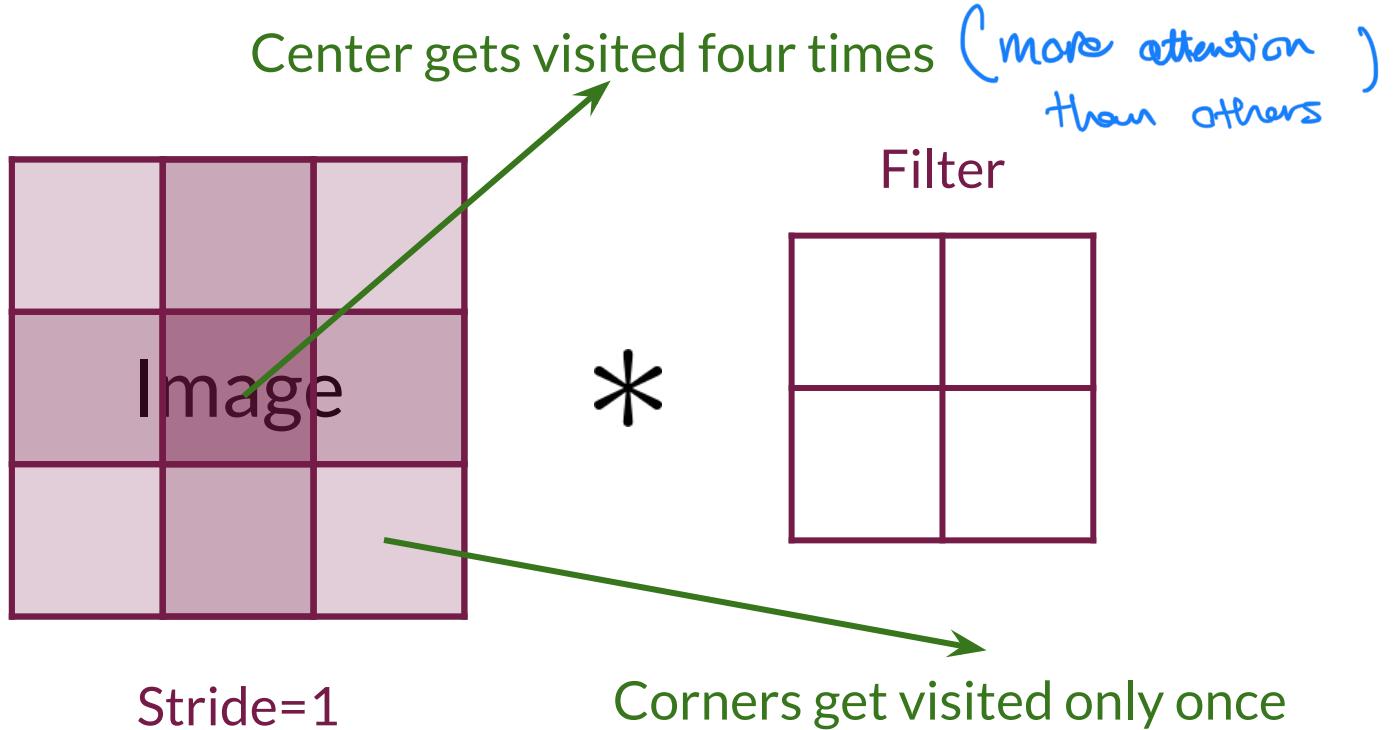
1	0	-1
1	0	-1
1	0	-1

=

150	0
150	0

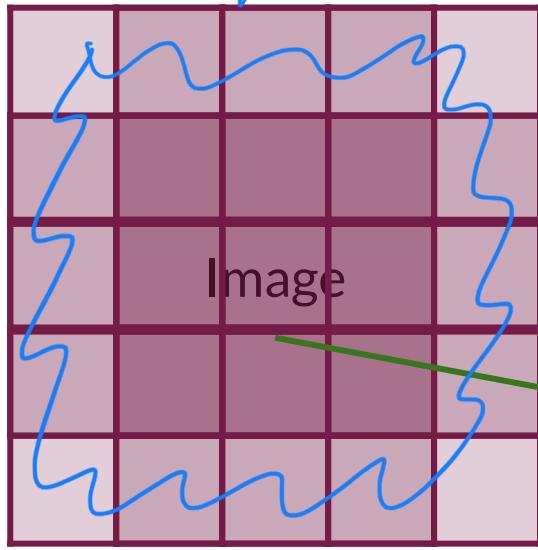
Stride = 2

Padding



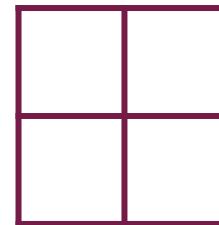
Padding

padding (often zero)



*

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



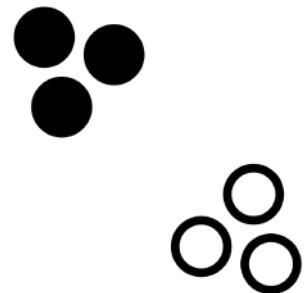


deeplearning.ai

Pooling and Upsampling

Outline

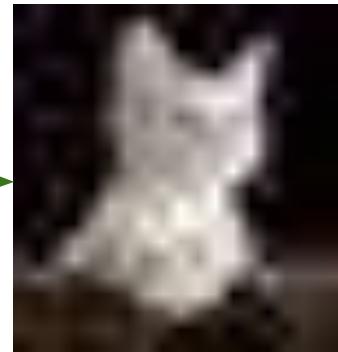
- Pooling
- Upsampling and its relation to pooling



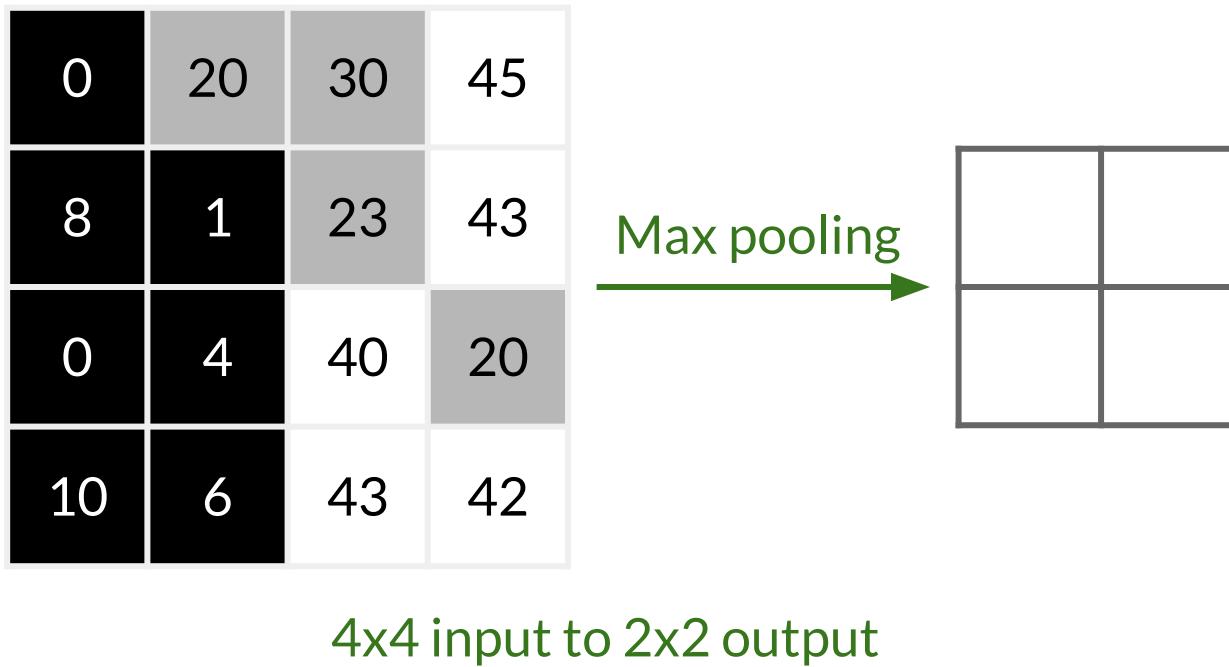
Pooling



Pooling



Max Pooling



Max Pooling

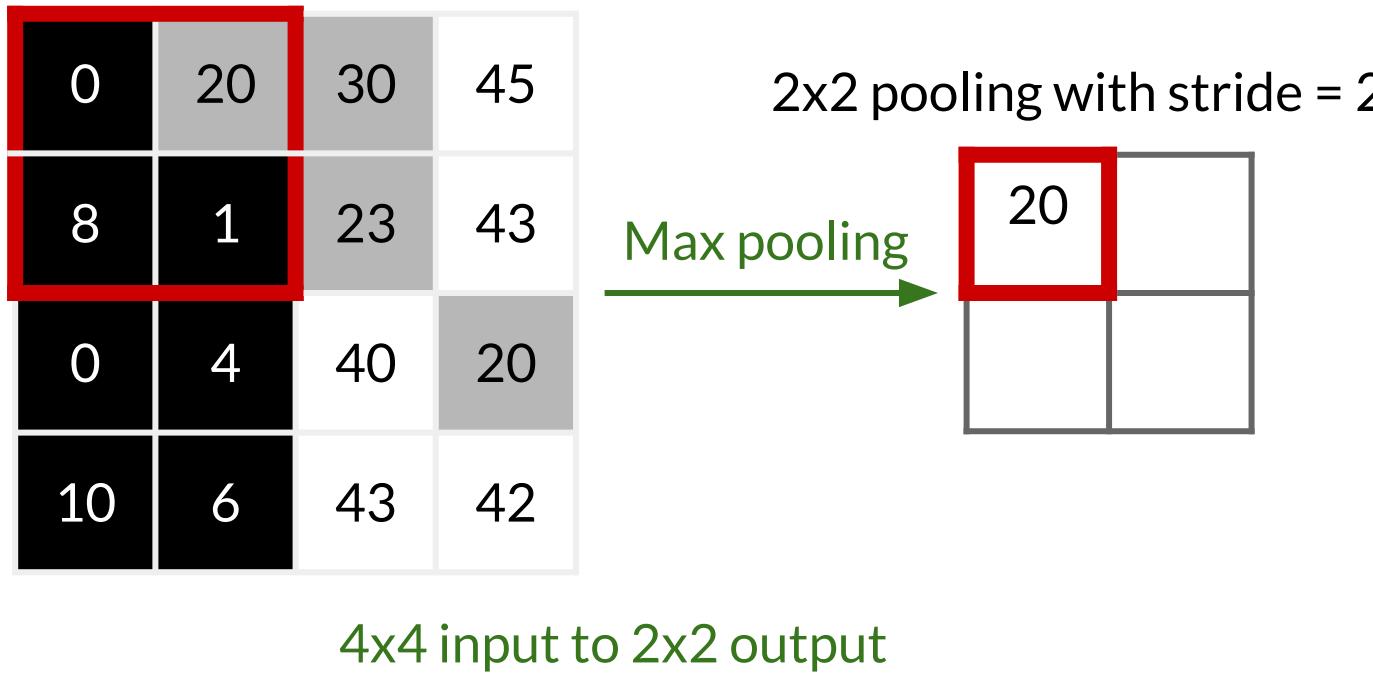
0	20	30	45
8	1	23	43
0	4	40	20
10	6	43	42

2x2 pooling with stride = 2

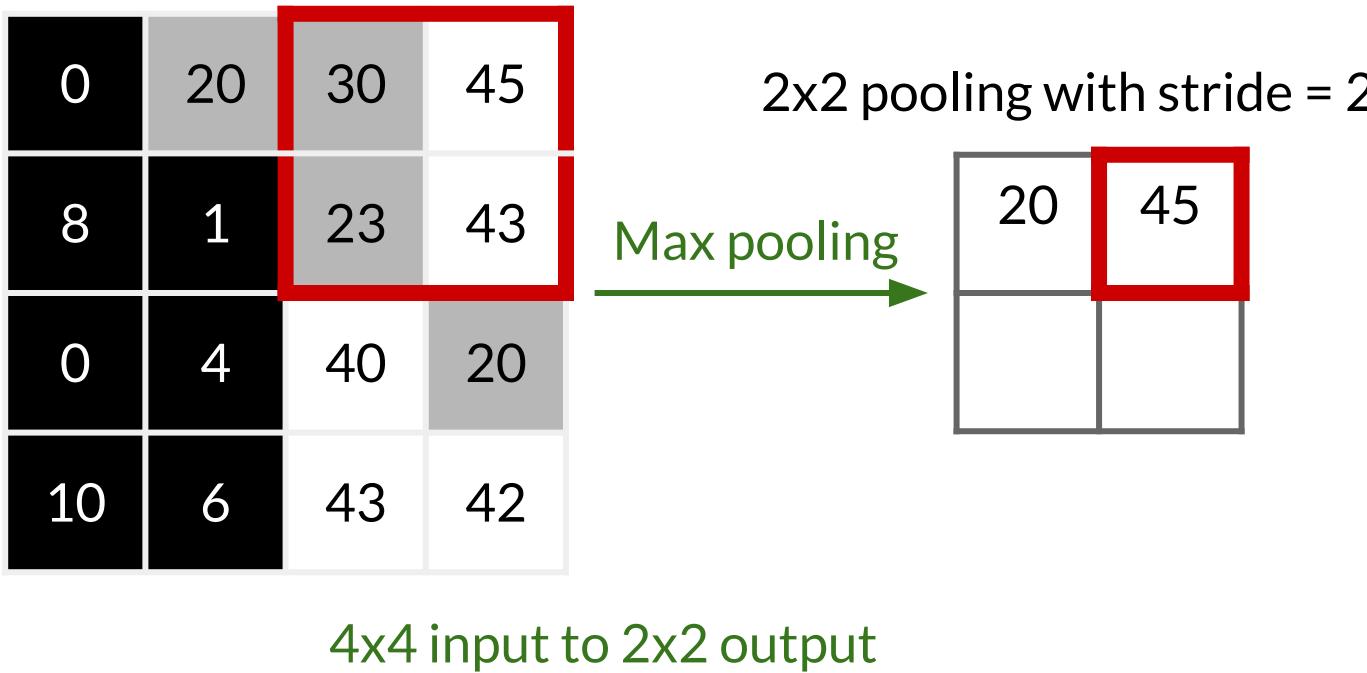
Max pooling

4x4 input to 2x2 output

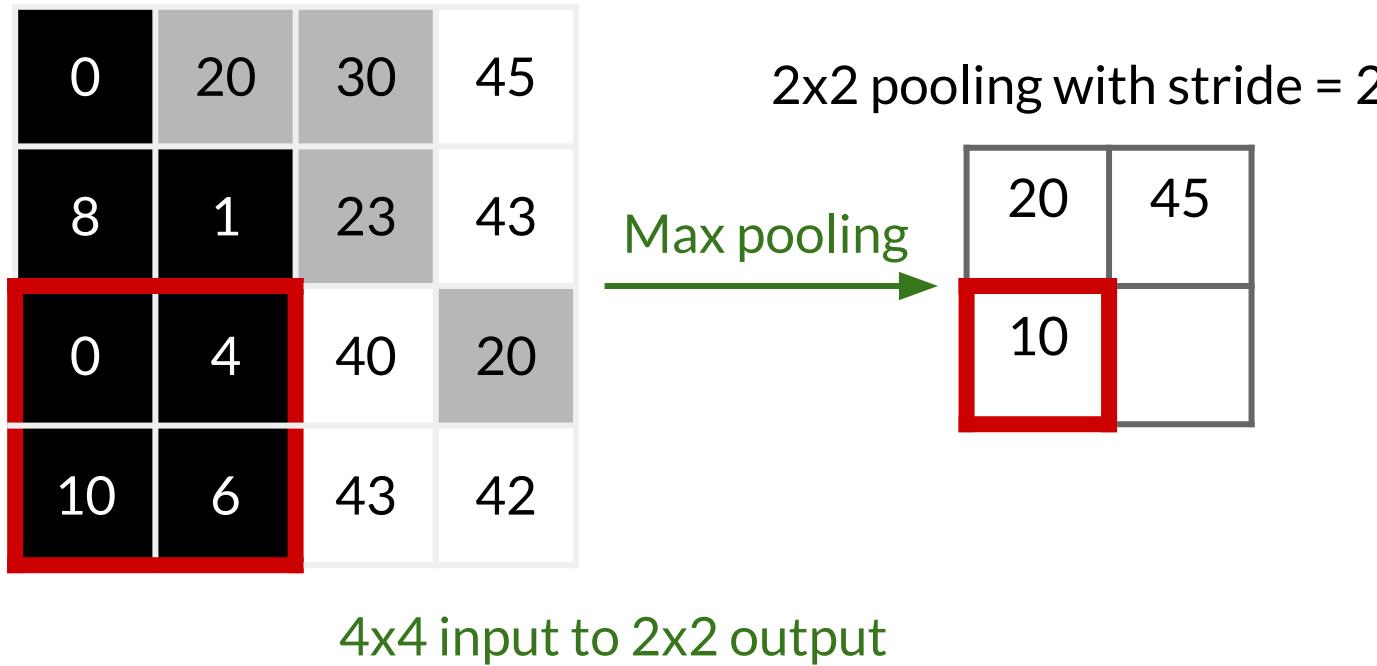
Max Pooling



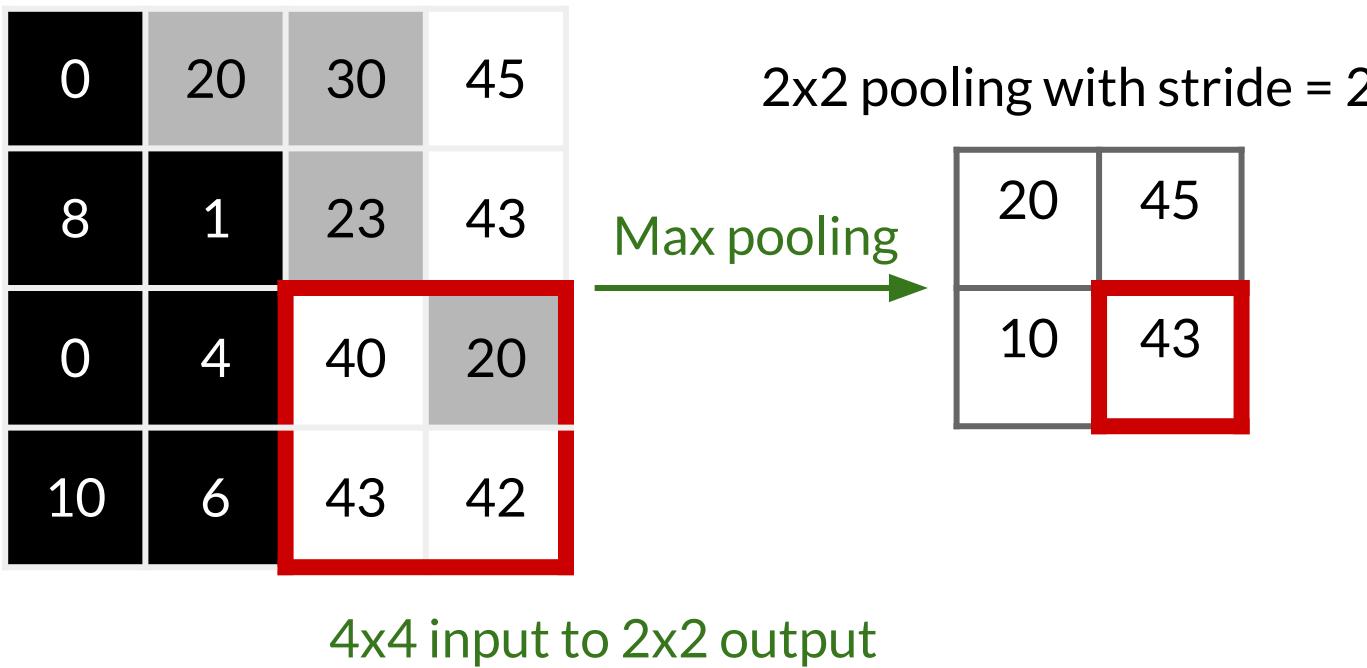
Max Pooling



Max Pooling



Max Pooling



Max Pooling *to get best/most important features*

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

2x2 pooling with stride = 2

Max pooling

20	45
10	43

4x4 input to 2x2 output

Other types include:

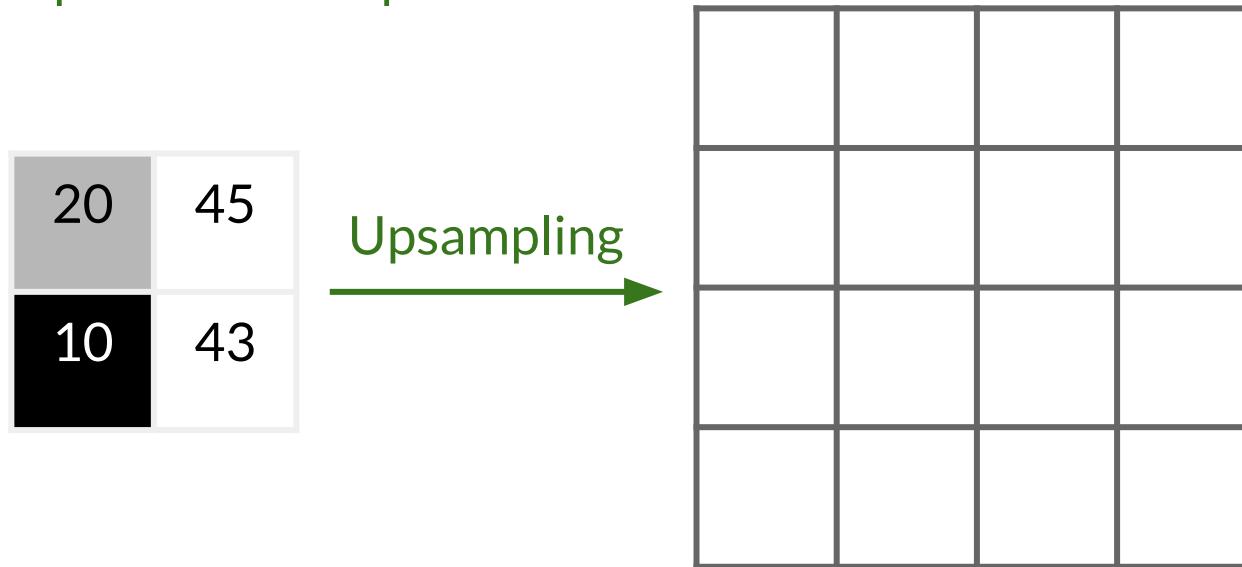
1. Average pooling
2. Min pooling

Upsampling



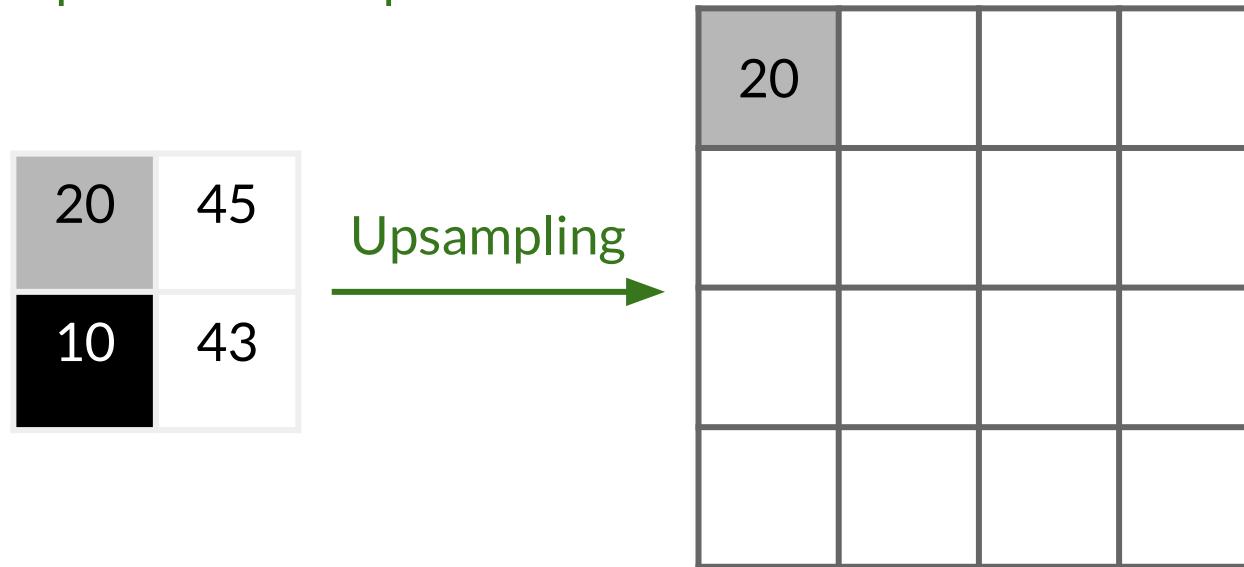
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



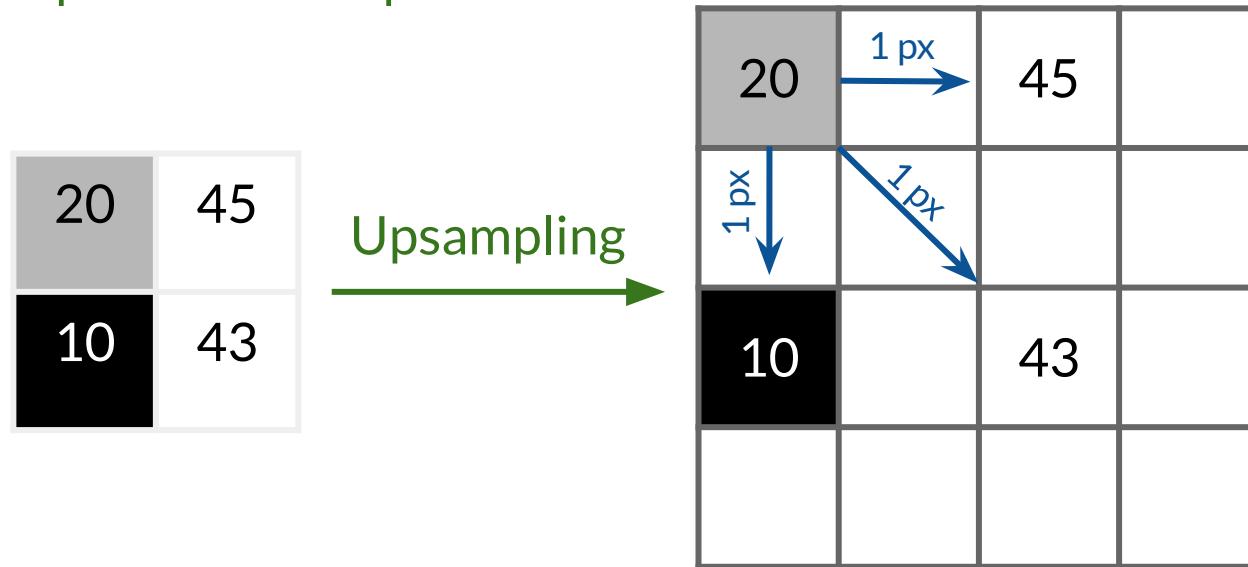
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



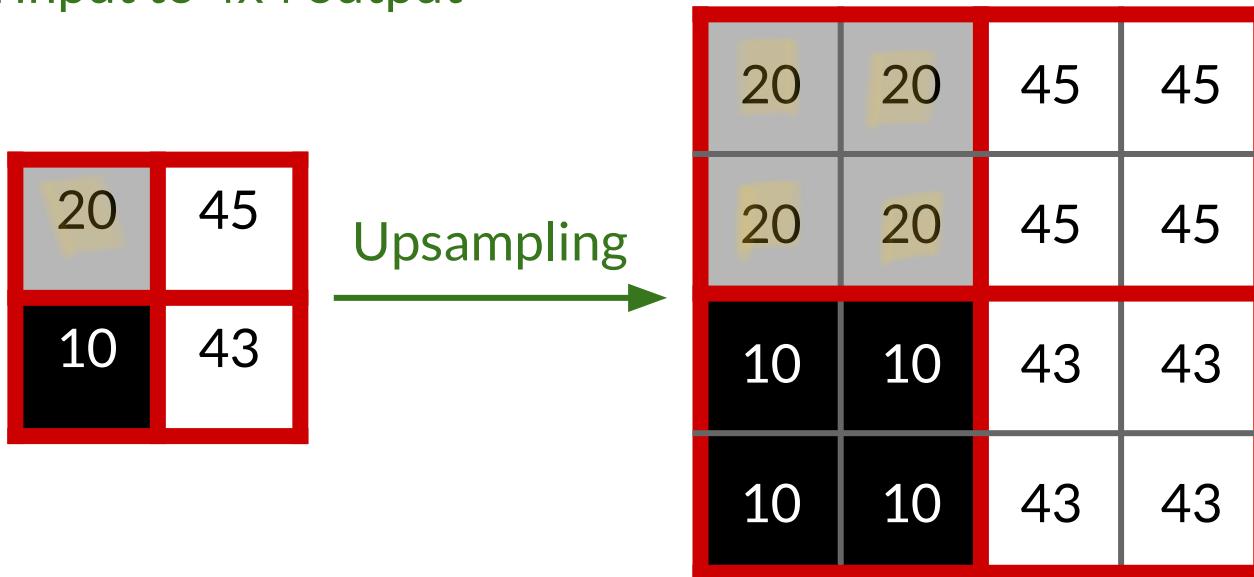
Upsampling: Nearest Neighbors

2x2 input to 4x4 output



Upsampling: Nearest Neighbors

2x2 input to 4x4 output



Upsampling: Nearest Neighbors

2x2 input to 4x4 output

20	45
10	43

Upsampling

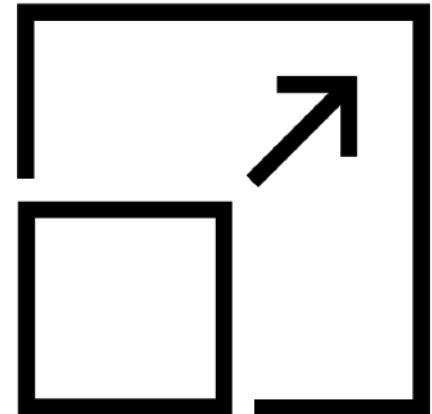
20	20	45	45
20	20	45	45
10	10	43	43
10	10	43	43

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!





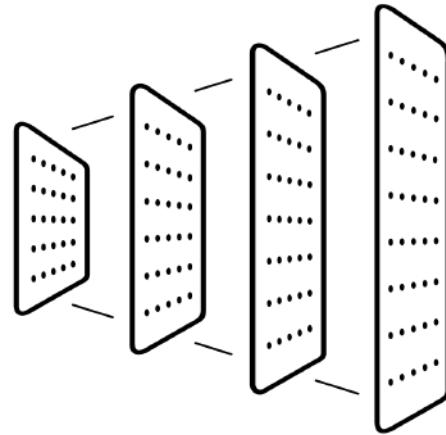
deeplearning.ai

Transposed Convolutions

Outline

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

to enlarge
input



Transposed Convolution

$2 \times 2 \rightarrow 3 \times 3$

Influence from every value in the input

Input Filter

Stride = 1

1×2

$1 \times 2 + 4 \times 2$

4×2

$1 \times 1 + 0 \times 2$

$1 \times 1 + 4 \times 1$

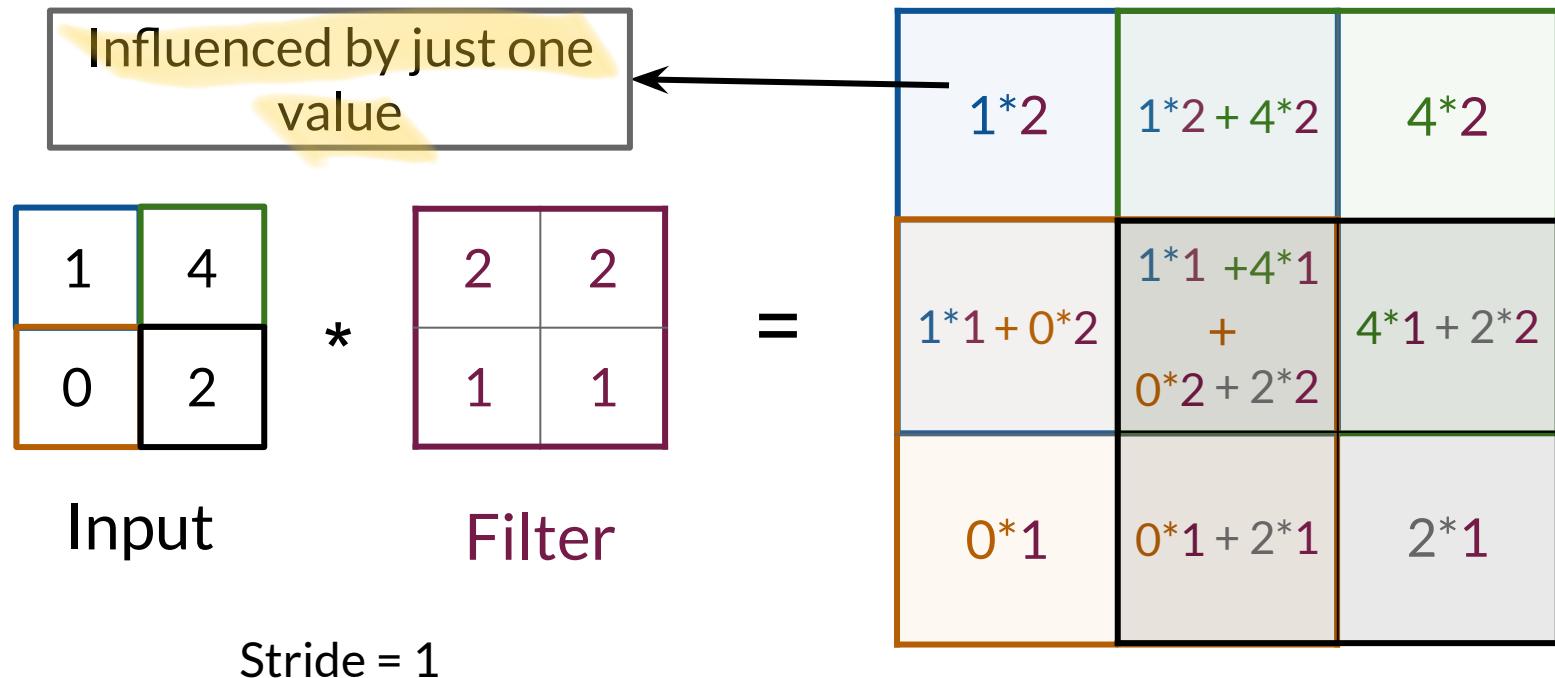
$0 \times 2 + 2 \times 2$

0×1

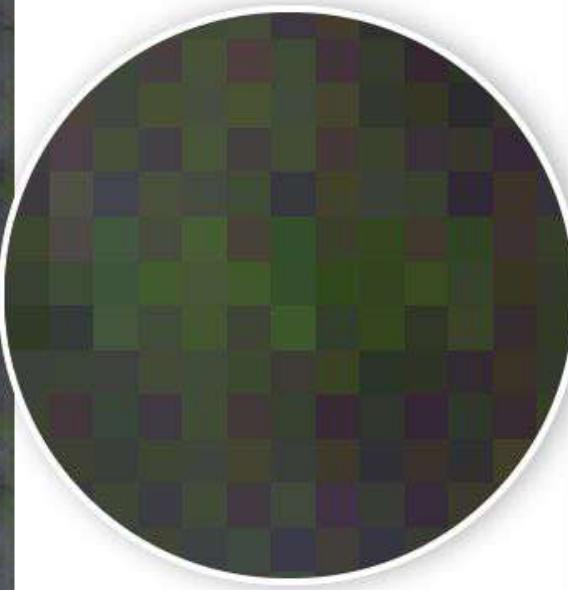
$0 \times 1 + 2 \times 1$

2×1

Transposed Convolution



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

