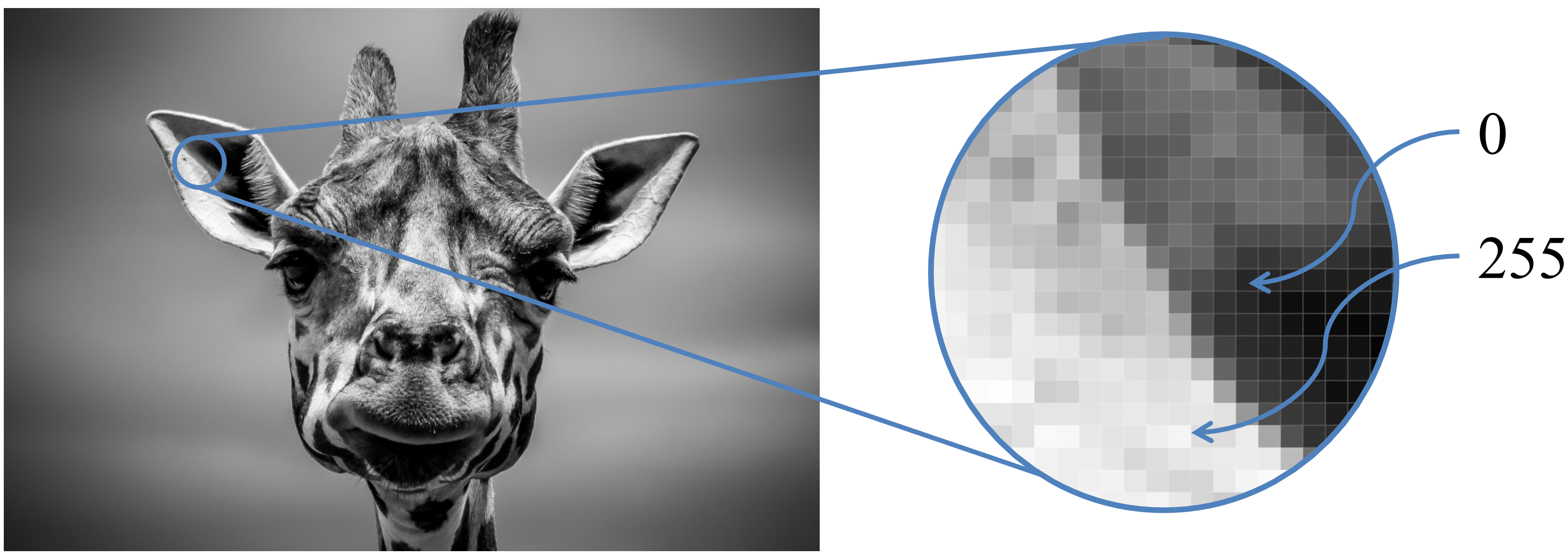
Intro

- Hi! My name is Andrey. This week you will learn how to solve computer vision tasks with neural networks
- You already know about MLP that has lots of hidden layers
- In this video we will introduce a new layer of neurons specifically designed for image input

Digital representation of an image

- Grayscale image is a matrix of pixels (**pic**ture **el**ements)
- Dimensions of this matrix are called image resolution (e.g. 300 x 300)
- Each pixel stores its brightness (or **intensity**) ranging from 0 to 255, 0 intensity corresponds to black color:



- Color images store pixel intensities for 3 channels: red, green and blue

Image as a neural network input

- Normalize input pixels: $x_{norm} = \frac{x}{255} - 0.5$

Image as a neural network input

- Normalize input pixels: $x_{norm} = \frac{x}{255} - 0.5$
- Maybe MLP will work?

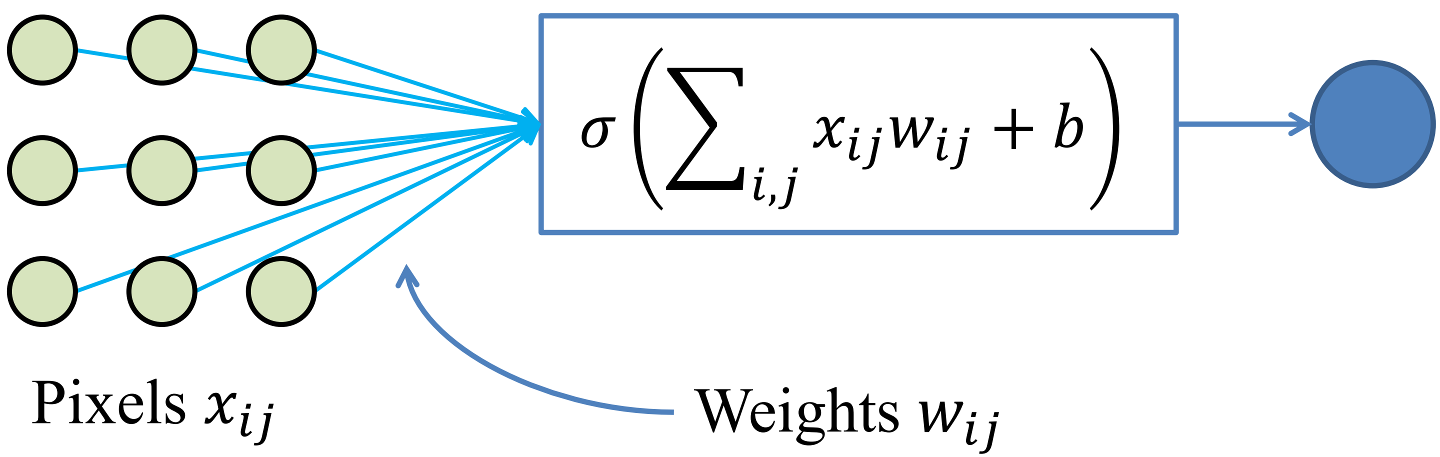
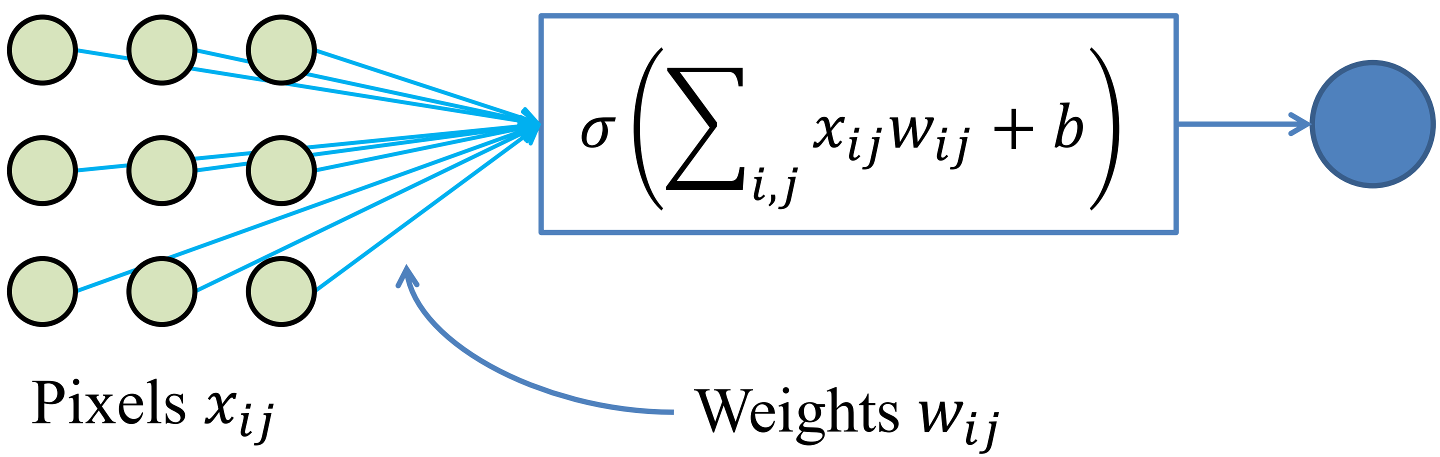


Image as a neural network input

- Normalize input pixels: $x_{norm} = \frac{x}{255} - 0.5$

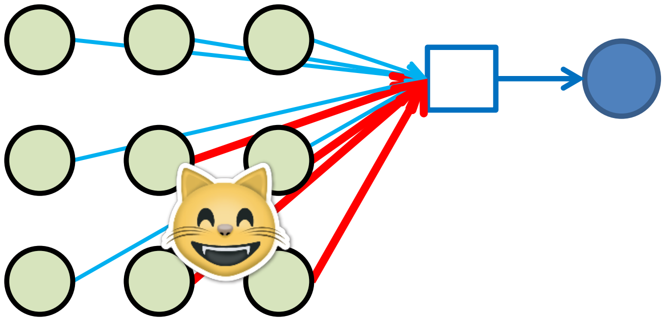
- Maybe MLP will work?



- Actually, no!

Why not MLP?

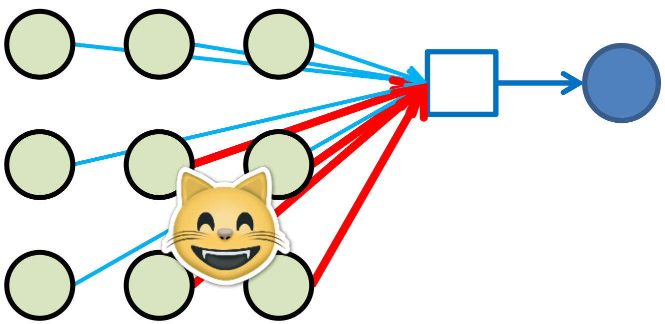
- Let’s say we want to train a “cat detector”



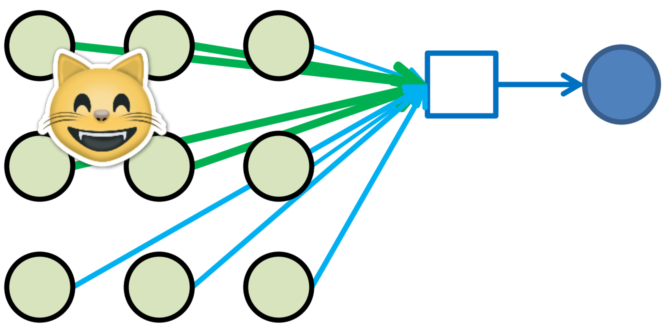
On this training image red weights w_{ij} will change a little bit to better detect a cat

Why not MLP?

- Let's say we want to train a "cat detector"



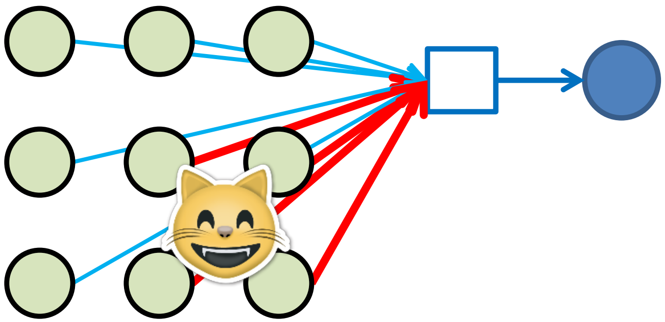
On this training image red weights w_{ij} will change a little bit to better detect a cat



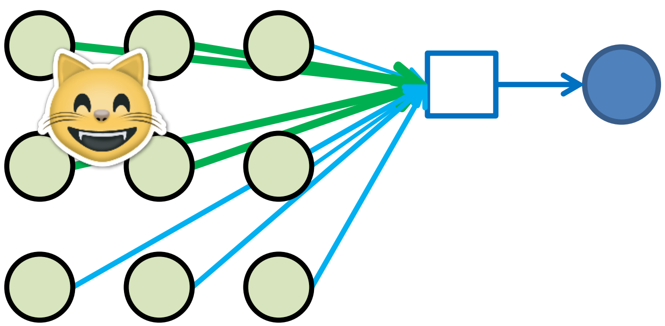
On this training image green weights w_{ij} will change…

Why not MLP?

- Let’s say we want to train a “cat detector”



On this training image red weights w_{ij} will change a little bit to better detect a cat

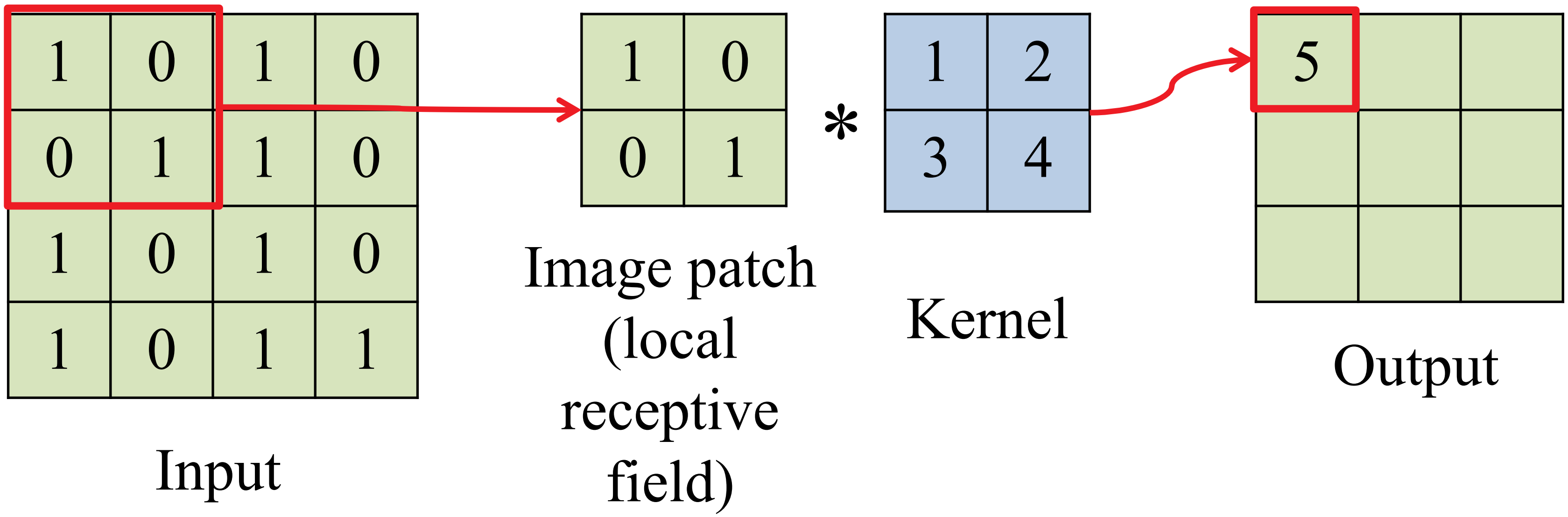


On this training image green weights w_{ij} will change…

- We learn the same “cat features” in different areas and don’t fully utilize the training set!
- What if cats in the test set appear in different places?

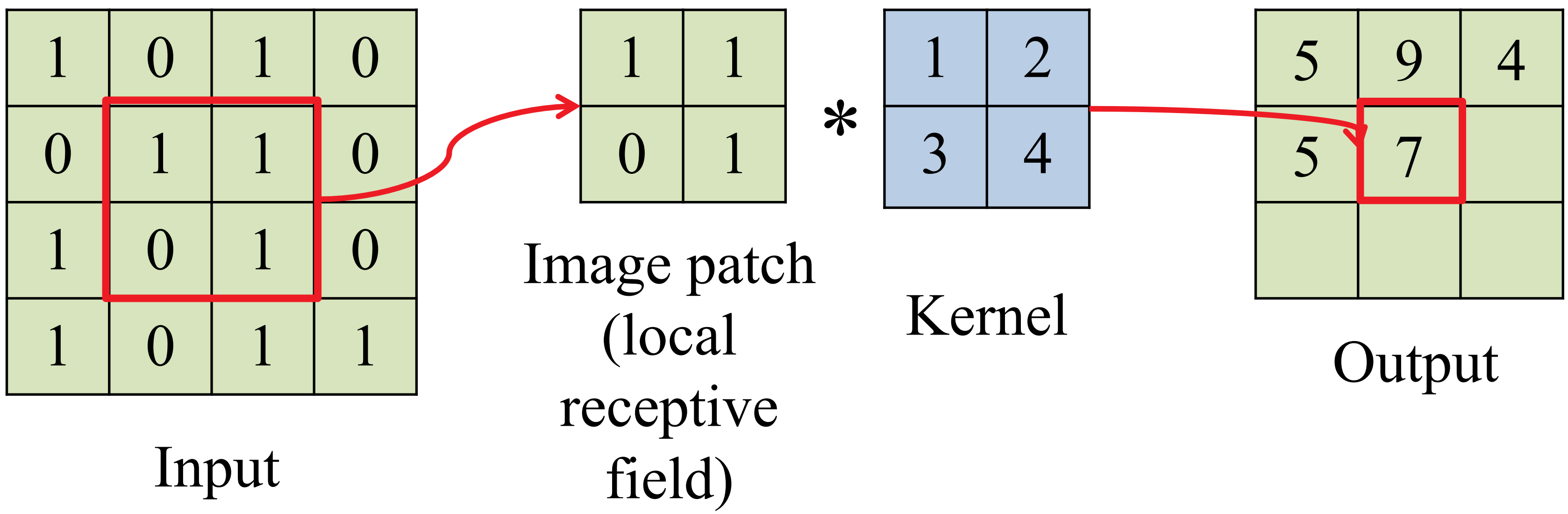
Convolutions will help!

Convolution is a dot product of a **kernel** (or filter) and a patch of an image (**local receptive field**) of the same size



Convolutions will help!

Convolution is a dot product of a **kernel** (or filter) and a patch of an image (**local receptive field**) of the same size



Convolutions have been used for a while

Kernel

$$* \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} =$$

Edge detection

Original image

Sums up to 0 (black color) when the patch is a solid fill

Convolutions have been used for a while

Kernel

Original image

$*$

-1	-1	-1
-1	8	-1
-1	-1	-1

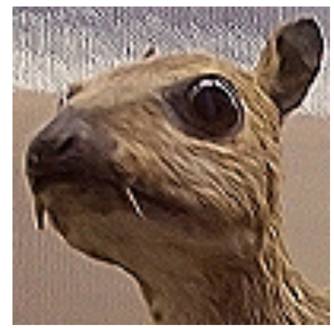
$=$

Edge detection

$*$

0	-1	0
-1	5	-1
0	-1	0

$=$



Sharpening

Doesn’t change an image for solid fills

Adds a little intensity on the edges

Convolutions have been used for a while

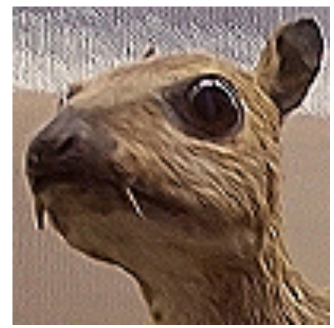
Original image

Kernel

$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

Edge detection

$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

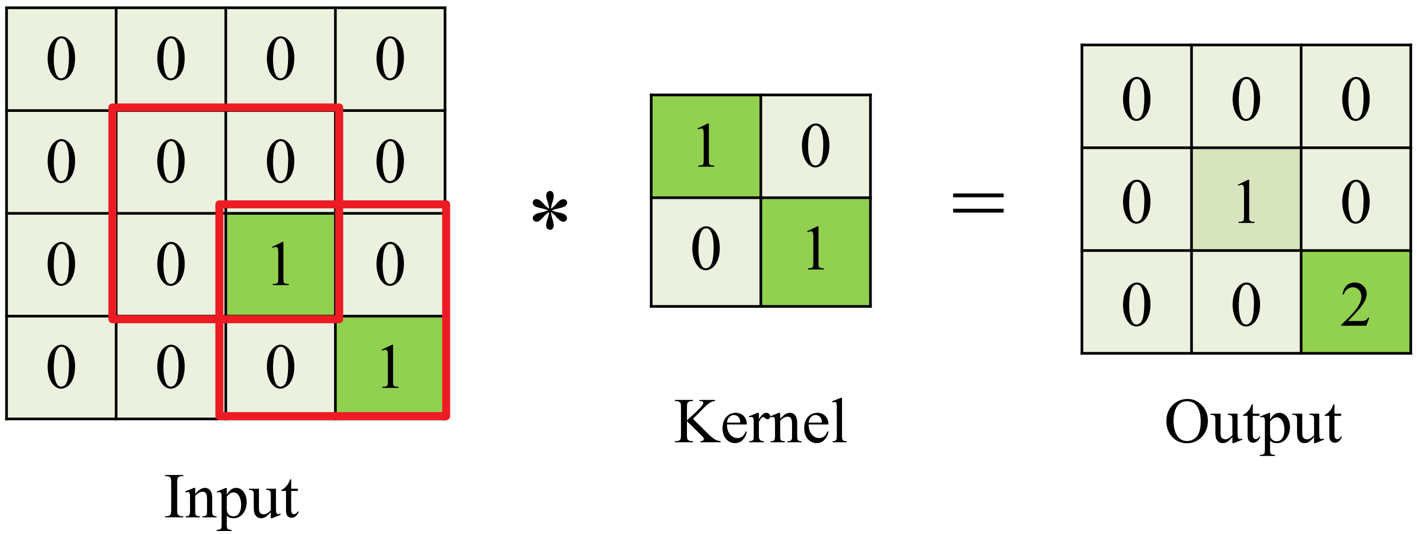


Sharpening

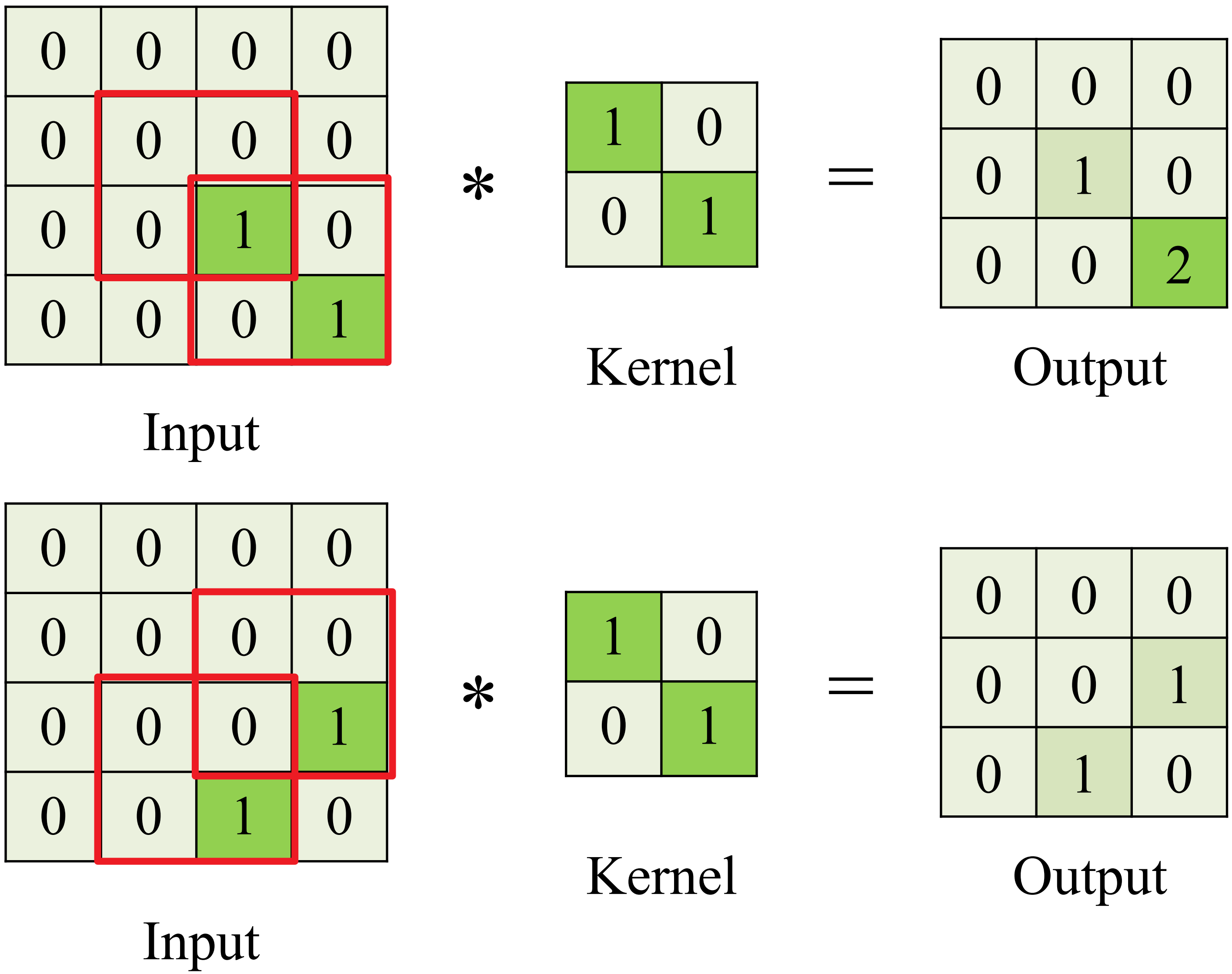
$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Blurring

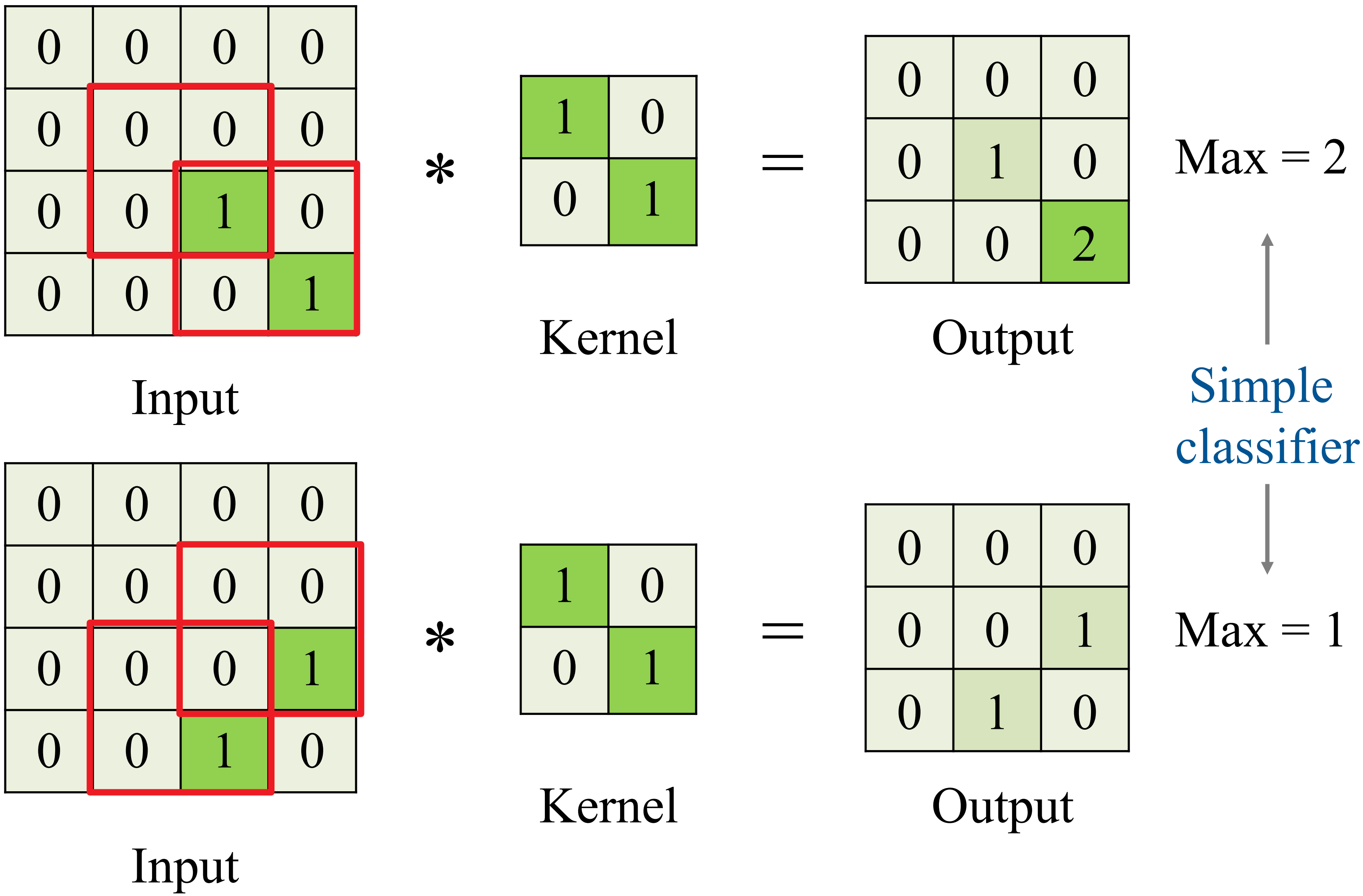
Convolution is similar to correlation



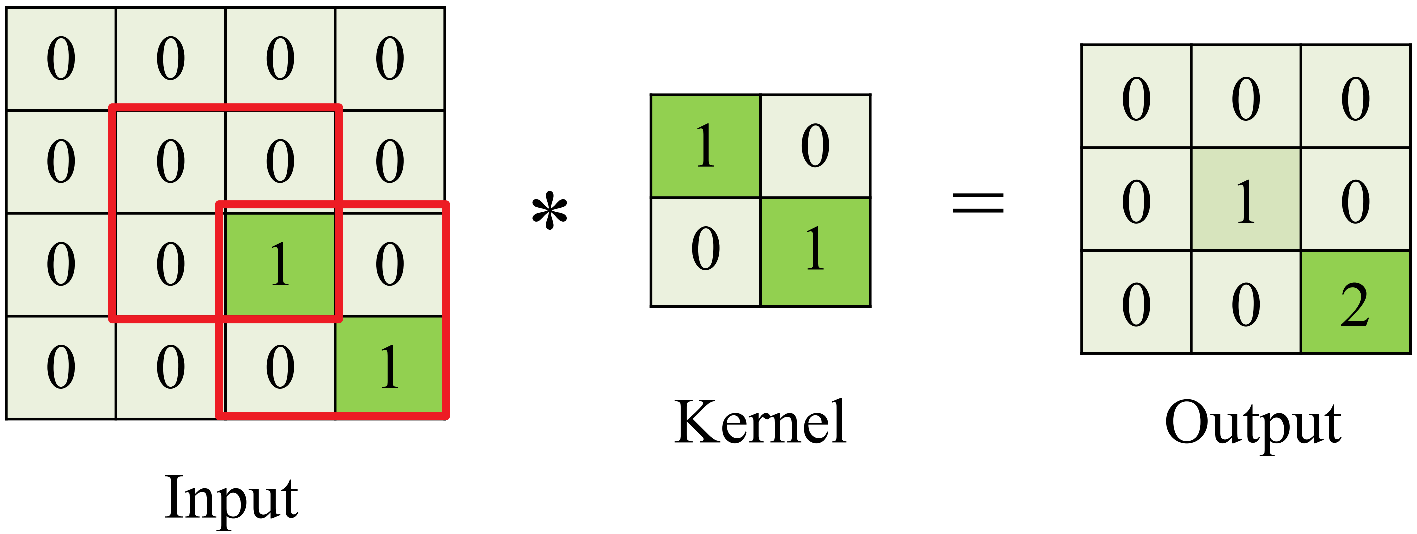
Convolution is similar to correlation



Convolution is similar to correlation



Convolution is translation equivariant

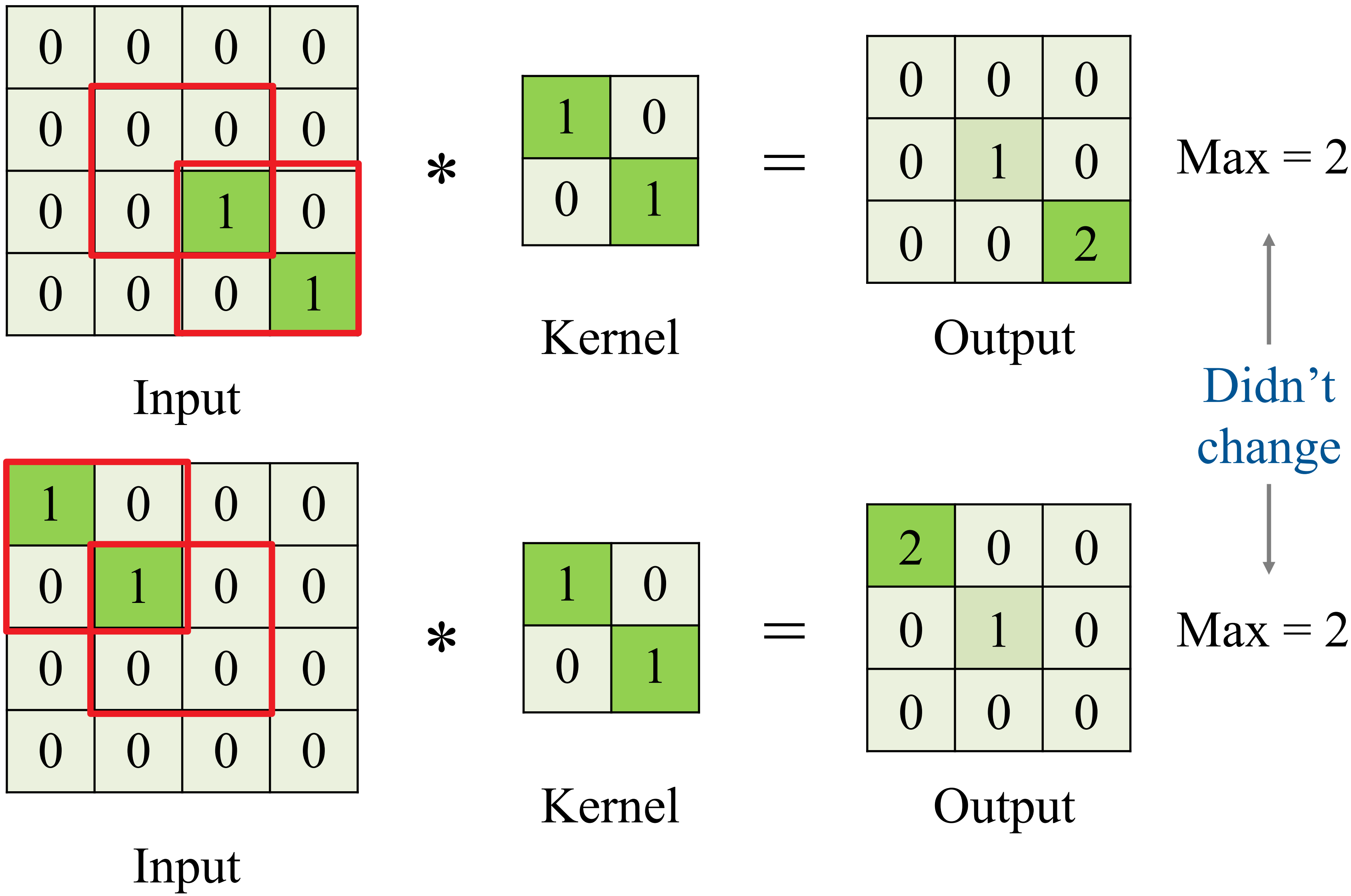


Convolution is translation equivariant

$$\underbrace{\begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix}1&0\\0&1\end{bmatrix}}_{\text{Kernel}} = \underbrace{\begin{bmatrix}0&0&0\\0&1&0\\0&0&2\end{bmatrix}}_{\text{Output}}$$

$$\underbrace{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{bmatrix}}_{\text{Input}} * \underbrace{\begin{bmatrix}1&0\\0&1\end{bmatrix}}_{\text{Kernel}} = \underbrace{\begin{bmatrix}2&0&0\\0&1&0\\0&0&0\end{bmatrix}}_{\text{Output}}$$

Convolution is translation equivariant



Convolutional layer in neural network

Shared bias:
b

Shared kernel:

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

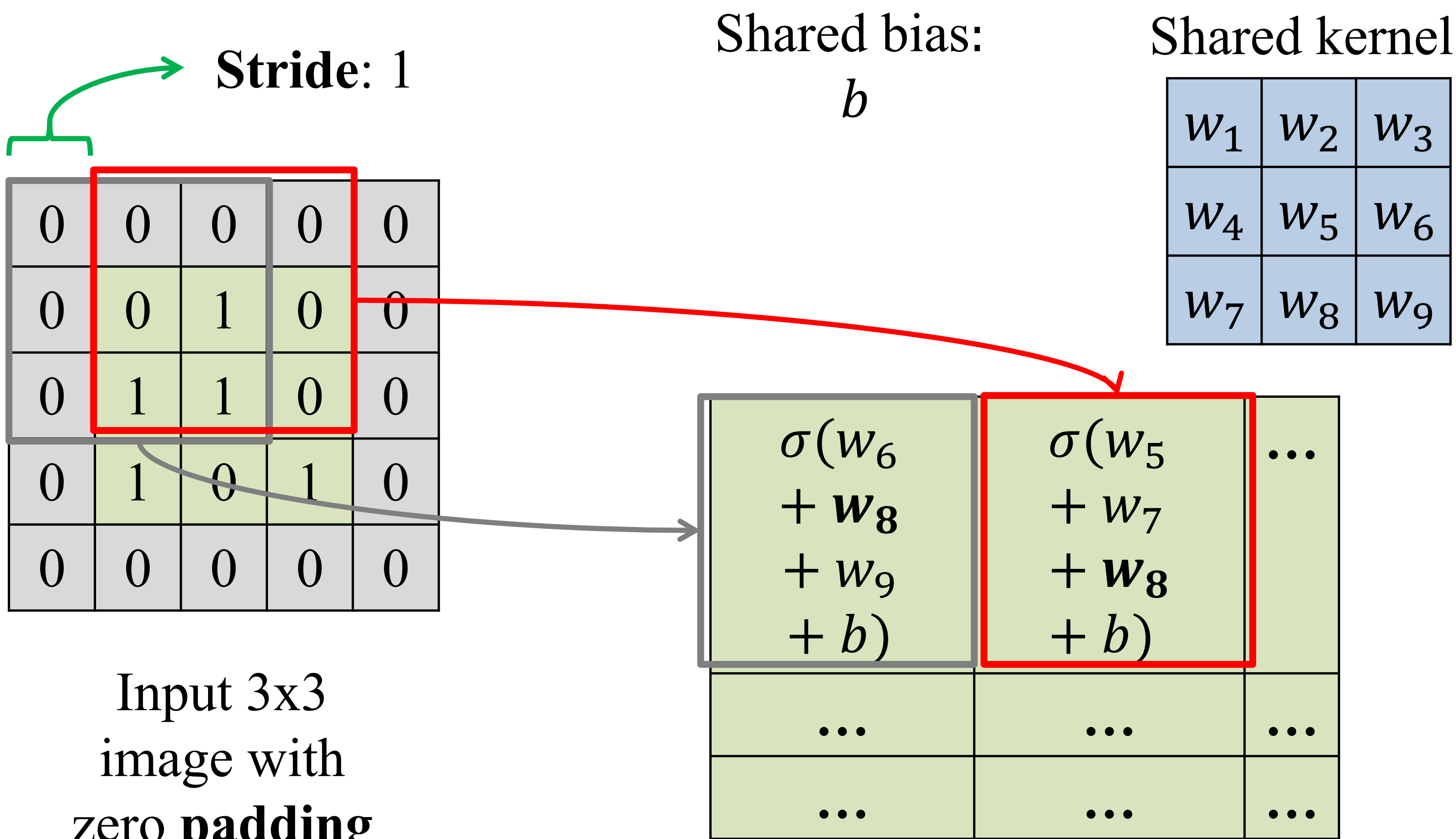
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
0	1	0	1	0
0	0	0	0	0

Input 3x3 image with zero **padding** (grey area)

$\sigma(w_6 + \boldsymbol{w_8} + w_9 + b)$	…	…
…	…	…
…	…	…

9 output neurons (**feature map**) with only 10 parameters

Convolutional layer in neural network

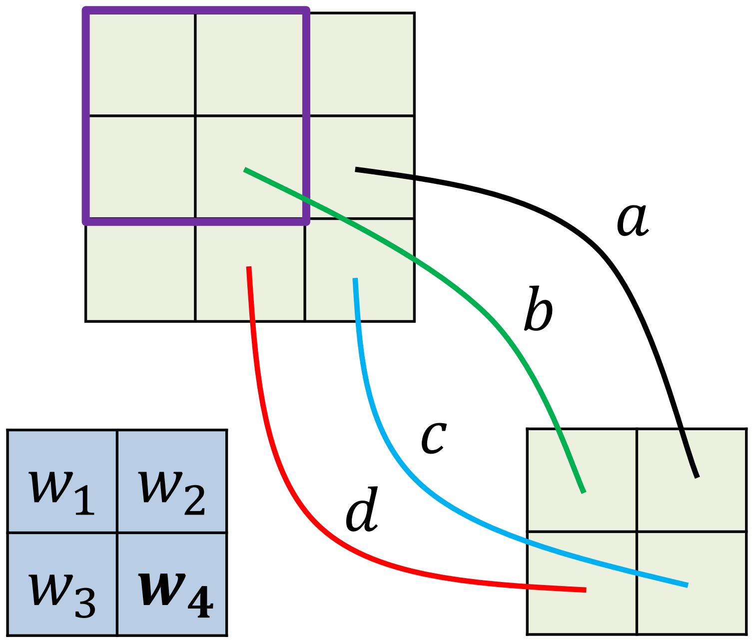


Input 3x3 image with zero **padding** (grey area)

9 output neurons (**feature map**) with only 10 parameters

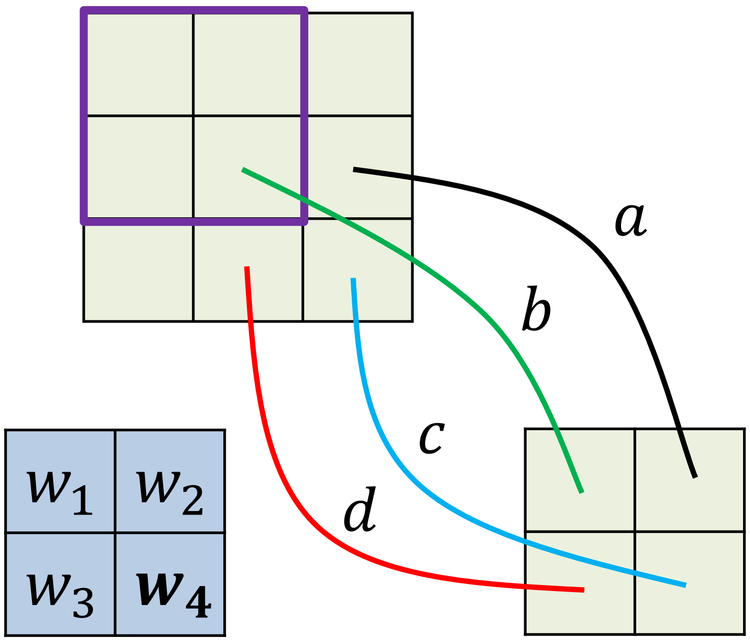
Backpropagation for CNN

Gradients are first calculated as if the kernel weights were not shared:



Backpropagation for CNN

Gradients are first calculated as if the kernel weights were not shared:

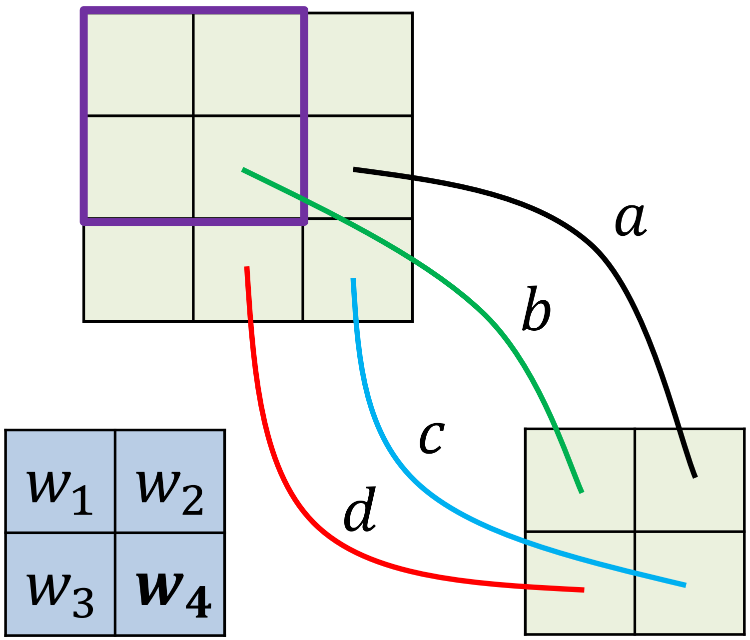


$$a = a - \gamma \frac{\partial L}{\partial a} \qquad b = b - \gamma \frac{\partial L}{\partial b}$$

$$c = c - \gamma \frac{\partial L}{\partial c} \qquad d = d - \gamma \frac{\partial L}{\partial d}$$

Backpropagation for CNN

Gradients are first calculated as if the kernel weights were not shared:



$$a = a - \gamma \frac{\partial L}{\partial a} \qquad b = b - \gamma \frac{\partial L}{\partial b}$$

$$c = c - \gamma \frac{\partial L}{\partial c} \qquad d = d - \gamma \frac{\partial L}{\partial d}$$

$$w_4 = w_4 - \gamma \left(\frac{\partial L}{\partial a} + \frac{\partial L}{\partial b} + \frac{\partial L}{\partial c} + \frac{\partial L}{\partial d} \right)$$

Gradients of the same shared weight are summed up!

Convolutional vs fully connected layer

- In convolutional layer the same kernel is used for every output neuron, this way we share parameters of the network and train a better model;

Convolutional vs fully connected layer

- In convolutional layer the same kernel is used for every output neuron, this way we share parameters of the network and train a better model;
- 300x300 input, 300x300 output, 5x5 kernel – **26** parameters in convolutional layer and $\mathbf{8.1\times10^{9}}$ parameters in fully connected layer (each output is a perceptron);

Convolutional vs fully connected layer

- In convolutional layer the same kernel is used for every output neuron, this way we share parameters of the network and train a better model;
- 300x300 input, 300x300 output, 5x5 kernel – **26** parameters in convolutional layer and $\mathbf{8.1 \times 10^9}$ parameters in fully connected layer (each output is a perceptron);
- Convolutional layer can be viewed as a special case of a fully connected layer when all the weights outside the **local receptive field** of each neuron equal 0 and kernel parameters are shared between neurons.

Summary

- We’ve introduced a convolutional layer which works better than fully connected layer for images: it has fewer parameters and acts the same for every patch of input.
- This layer will be used as a building block for larger neural networks!
- In the next video we will introduce one more layer that we will need to build our first fully working convolutional network!