

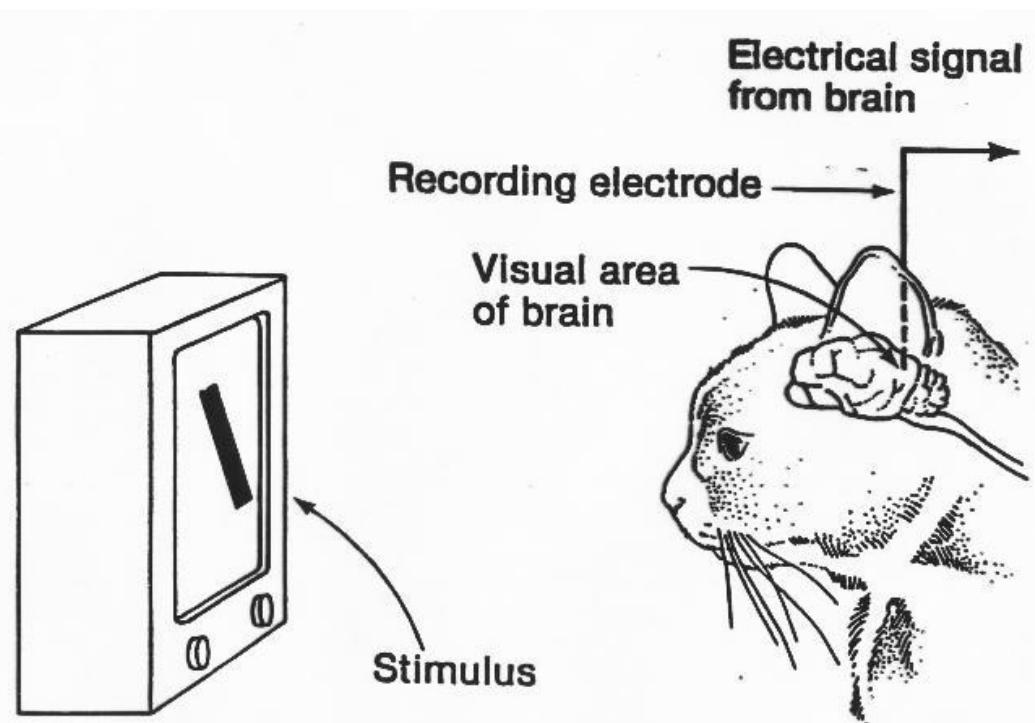
Neural networks

Convolution

The revolution is convolution

But not a new revolution

Hubel and Wiesel, 1959



The experiments showed that certain cells in the visual cortex respond to specific stimuli. Formally, the response is obtained by a convolution operation between the input signal and the cell functionality.

What is convolution?

- It is a linear operation that transform the image content by structural element called *filter* or *kernel*
- Technically, it is a linear function

10	5	3
4	5	1
1	1	7

Local image data

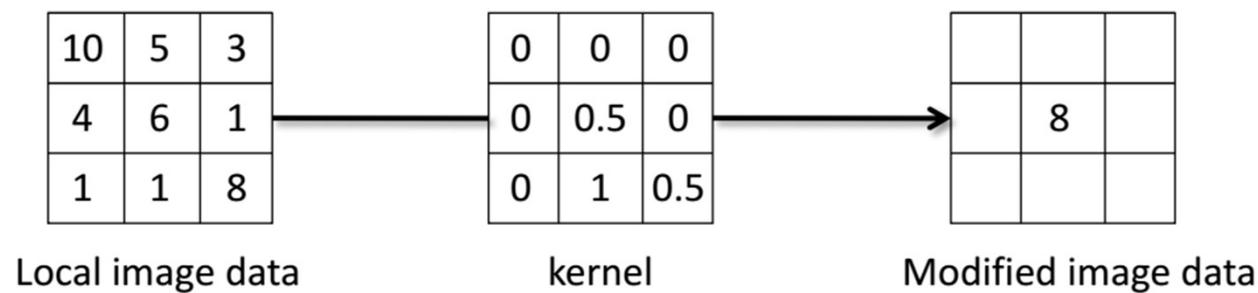


		7

Modified image data

What is convolution?

- In general, a pixel is replaced by a combination of its neighborhood
- The kernel contains the weights to compute that combination



Convolution

- Let F be an image, H (size $2k + 1, 2k + 1$) be a kernel, and G be the output image

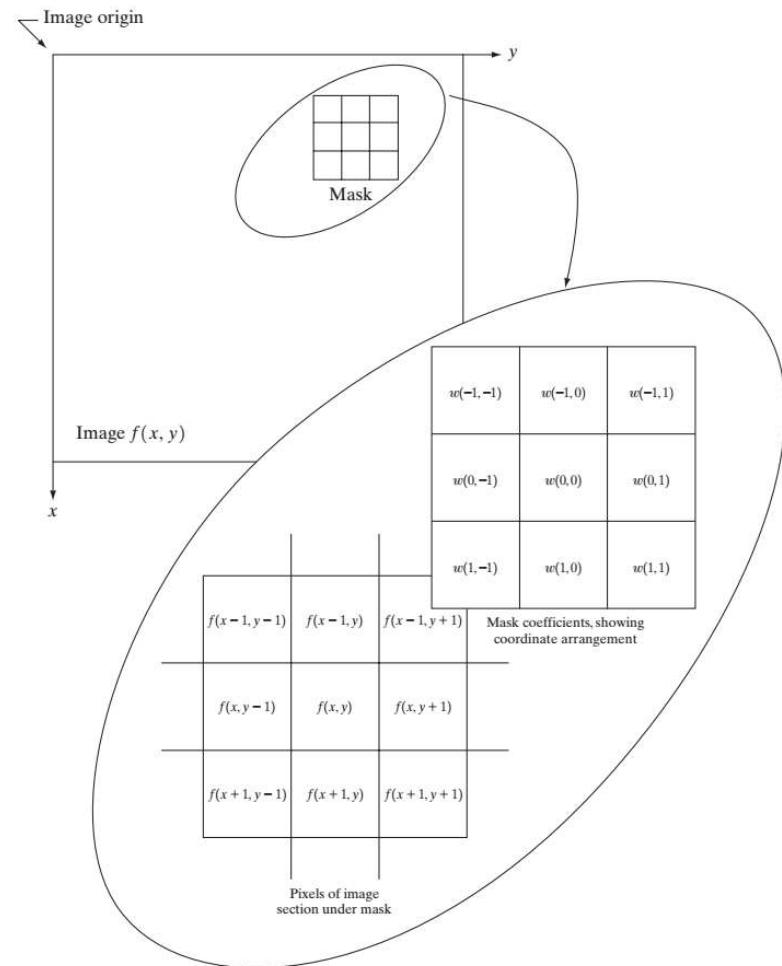
$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v]F[i - u, j - v]$$

- Convolution is associative and commutative

$$G = H * F$$

Convolution

- It is a pixel-wise operation
- The kernel slides in the image while computing the local linear transform.



Convolution: Example

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 3 & 2 & 2 & 1 & 3 \\ \hline 2 & 1 & 3 & 2 & 3 \\ \hline 2 & 3 & 3 & 1 & 2 \\ \hline 3 & 1 & 2 & 2 & 3 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

Convolution: Example

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 3 & 2 & 2 \\ \hline 2 & 1 & 3 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 20 & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

Convolution: Example

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 3 & 2 & 2 & 1 & 3 \\ \hline 2 & 1 & 3 & 2 & 3 \\ \hline 2 & 3 & 3 & 1 & 2 \\ \hline 3 & 1 & 2 & 2 & 3 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 20 & 18 & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

Convolution: Example

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 3 & 2 & 2 & 1 & 3 \\ \hline 2 & 1 & 3 & 2 & 3 \\ \hline 2 & 3 & 3 & 1 & 2 \\ \hline 3 & 1 & 2 & 2 & 3 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 20 & 18 & 19 \\ \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

The diagram illustrates a convolution operation. It shows two input matrices being multiplied by a smaller kernel matrix. The result is a single output value highlighted with a red circle.

The first input matrix (5x5) has values:

1	2	3	2	1
3	2	2	1	3
2	1	3	2	3
2	3	3	1	2
3	1	2	2	3

The second input matrix (3x3) has values:

1	1	1
1	1	1
1	1	1

The kernel matrix (3x3) has values:

1	1	1
1	1	1
1	1	1

The result matrix (3x3) has values:

20	18	19

The value 19 is circled in red, indicating it is the result of the convolution step where the kernel is aligned with the bottom-right 3x3 window of the input matrix.

Convolution: Example

$$\begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 3 & 2 & 2 & 1 & 3 \\ \hline 2 & 1 & 3 & 2 & 3 \\ \hline 2 & 3 & 3 & 1 & 2 \\ \hline 3 & 1 & 2 & 2 & 3 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 20 & 18 & 19 \\ \hline 20 & 18 & 19 \\ \hline 20 & 18 & 21 \\ \hline \end{array}$$

The diagram illustrates a convolution operation. It shows two input matrices (the left one has a blue border around its bottom-right 3x3 submatrix) being multiplied by a smaller kernel matrix (the middle one). The result is an output matrix where the bottom-right element (21) is circled in orange, indicating it is the result of the convolution step.

Convolution: Example

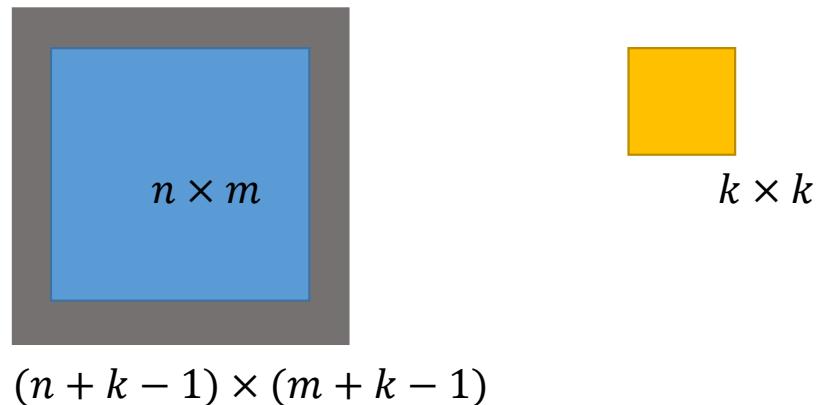
- Have you noted that we loose information?

$$\begin{array}{|c|c|c|} \hline & & \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|c|} \hline 1 & 2 & 3 & 2 & 1 \\ \hline 3 & 2 & 2 & 1 & 3 \\ \hline 2 & 1 & 3 & 2 & 3 \\ \hline 2 & 3 & 3 & 1 & 2 \\ \hline 3 & 1 & 2 & 2 & 3 \\ \hline \end{array} \quad * \quad \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|c|} \hline 20 & 18 & 19 \\ \hline 20 & 18 & 19 \\ \hline 20 & 18 & 21 \\ \hline \end{array}$$

- Input image is 5x5 and output image is 3x3
- We cannot apply the kernel in certain positions (border)

Convolution: padding

- Strategies for the boundary problem:
 - Take the output as it is. It could be not that harmful
 - Padding: fill input image with zeros, so the output is an image equal in size than the original



Convolution response

- Let's see some examples to gain insight about convolution



$$\text{Kernel} = \frac{1}{25}$$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Which operation are we performing?

Average

Convolution response

- Let's see some examples to gain insight about convolution

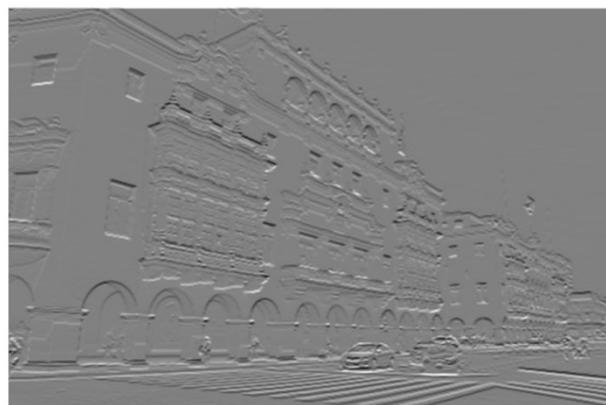


Kernel = average kernel of size 21



Convolution response

- Let's see some examples to gain insight about convolution

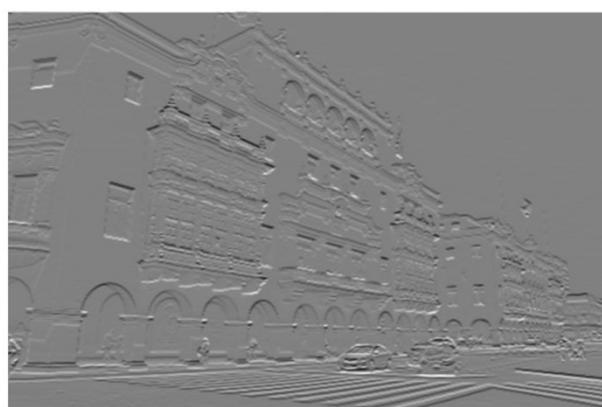


$$\text{Kernel} = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

Sobel, vertical edges

Convolution response

- Let's see some examples to gain insight about convolution



$$\text{Kernel} = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Sobel, horizontal edges

Summary

- Convolution is a linear operation
- It computes the response of an image against a kernel
- The kernel defines the kind of transformation to perform

Neural networks

Convolutional Layers

Images as data

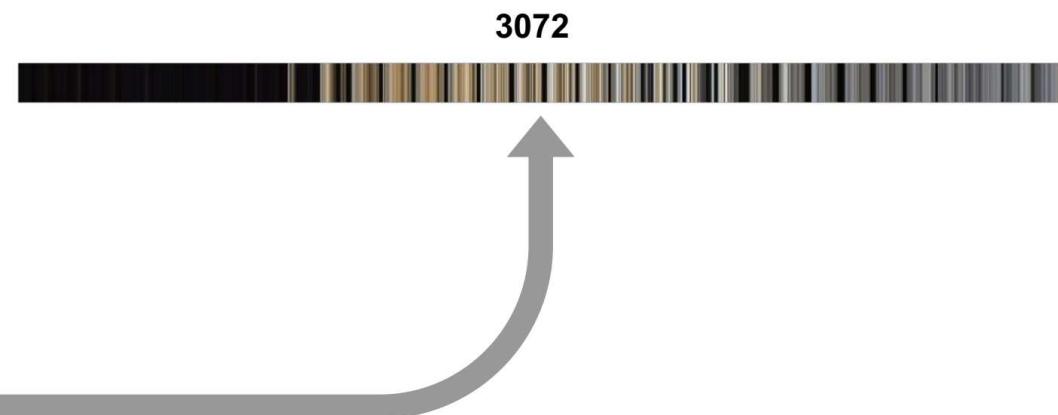
- If we want to implement a MLP for image classification



32x32x3

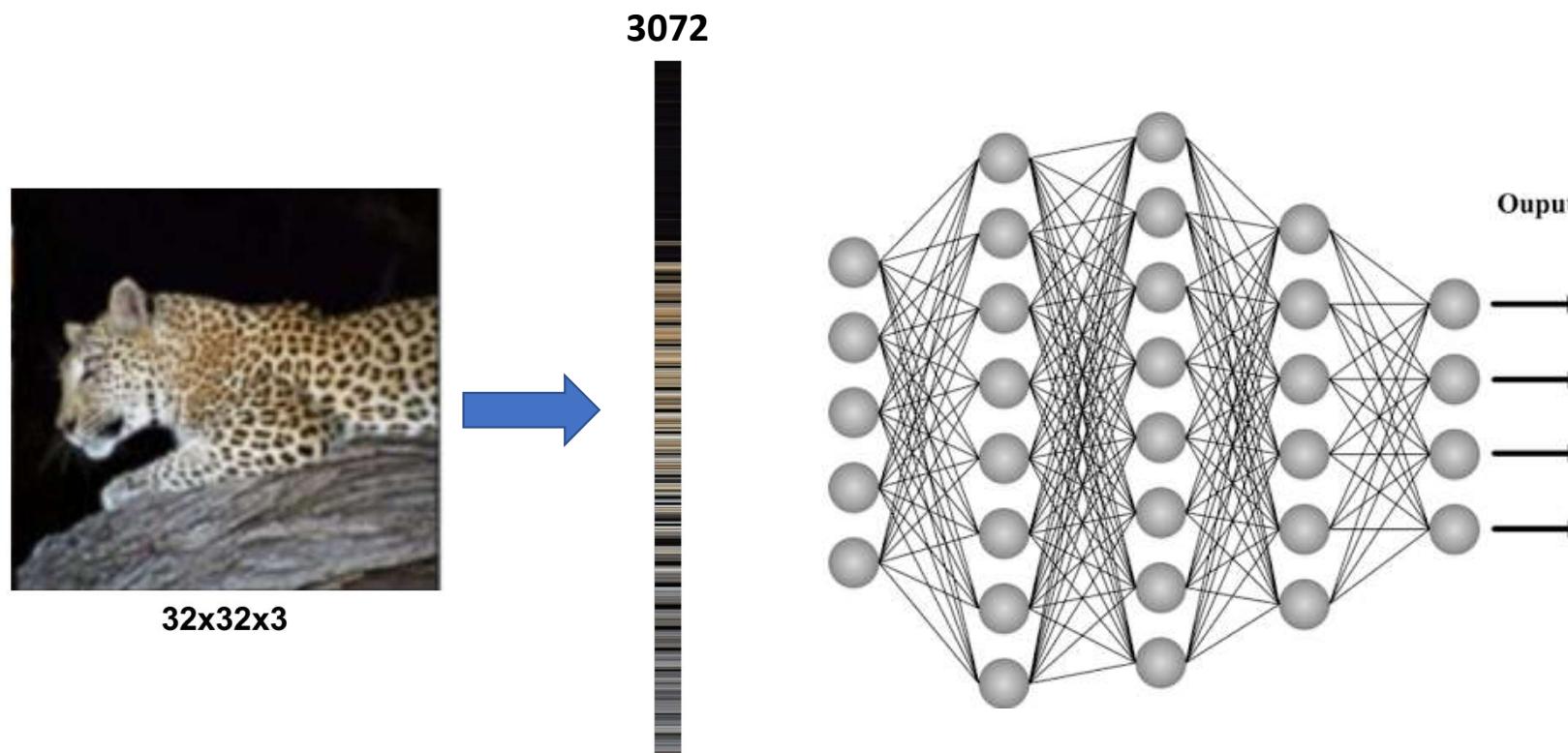
Images as data

- If we want to implement a MLP for image classification



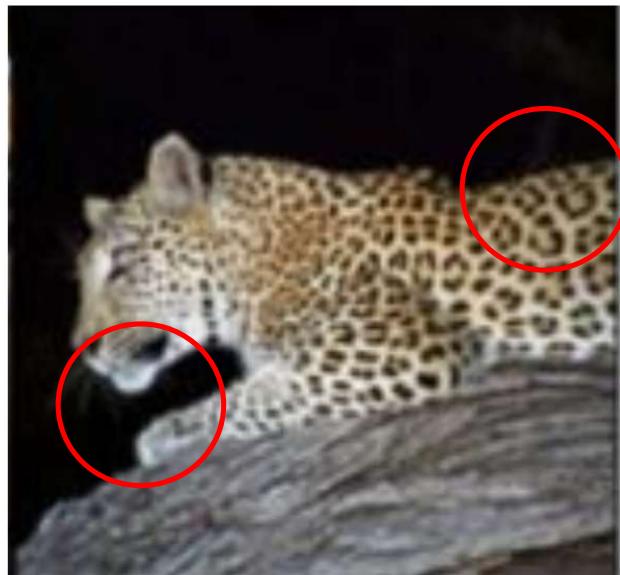
Images as data

- If we want to implement a MLP for image classification



Images as data

- If we want to implement a MLP for image classification
- The amount of parameters is huge
- A MLP tries to find all possible relations in the input data. Is that good?



Visual structure information is local, not global

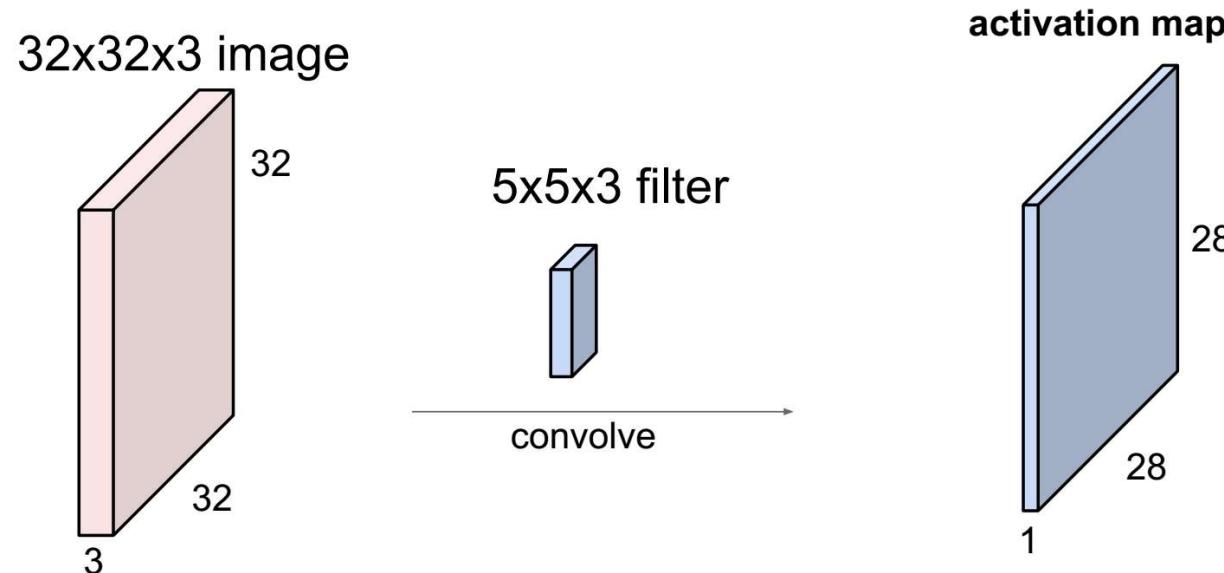
Pixels in far regions probably do not compose interesting content

We know an operation for computing local relationships

Convolution!

Convolutional Layers

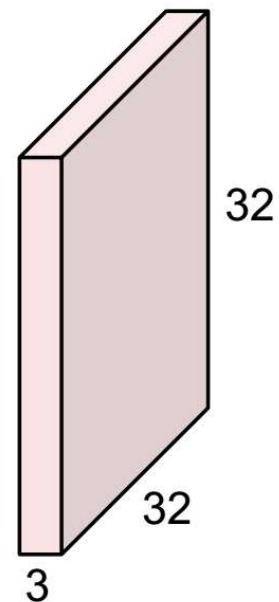
- A convolutional layer is composed of a set of kernels
- Each kernel is convolved with the input and computes an activation map (also known as feature map)



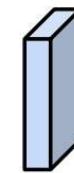
Convolutional Layers

- The depth of the input is always the depth of the kernels

32x32x3 image

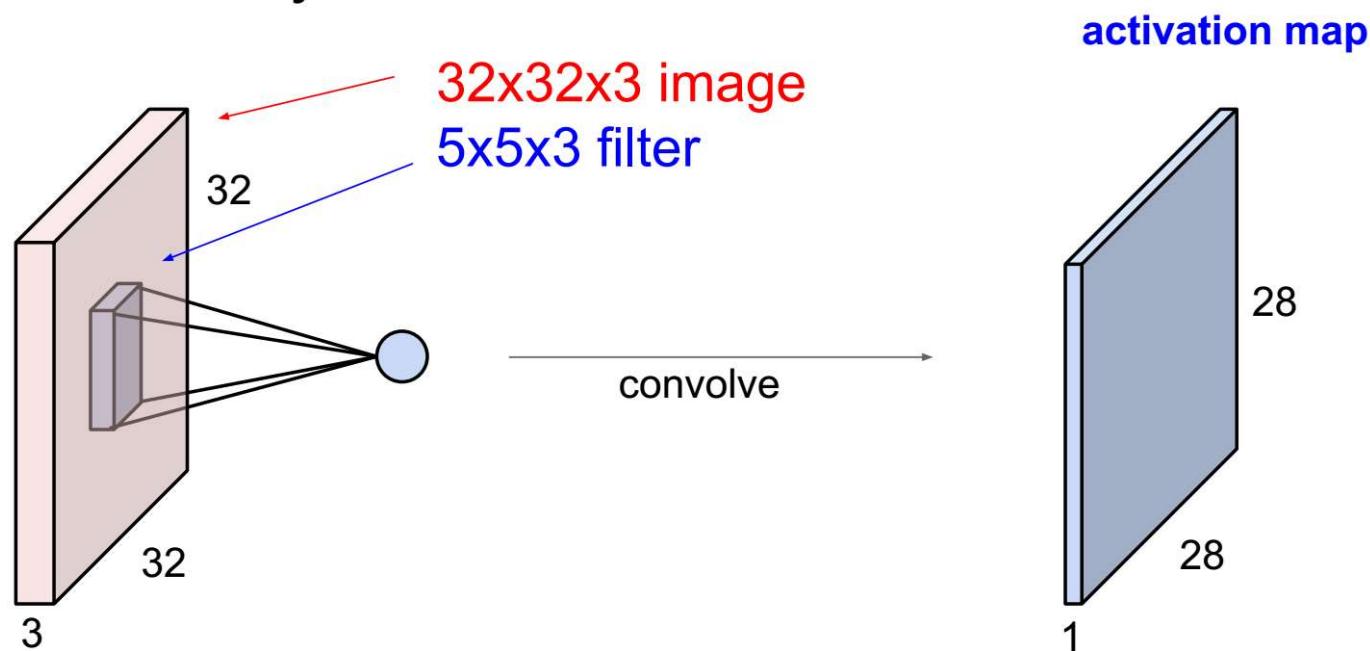


5x5x3 filter



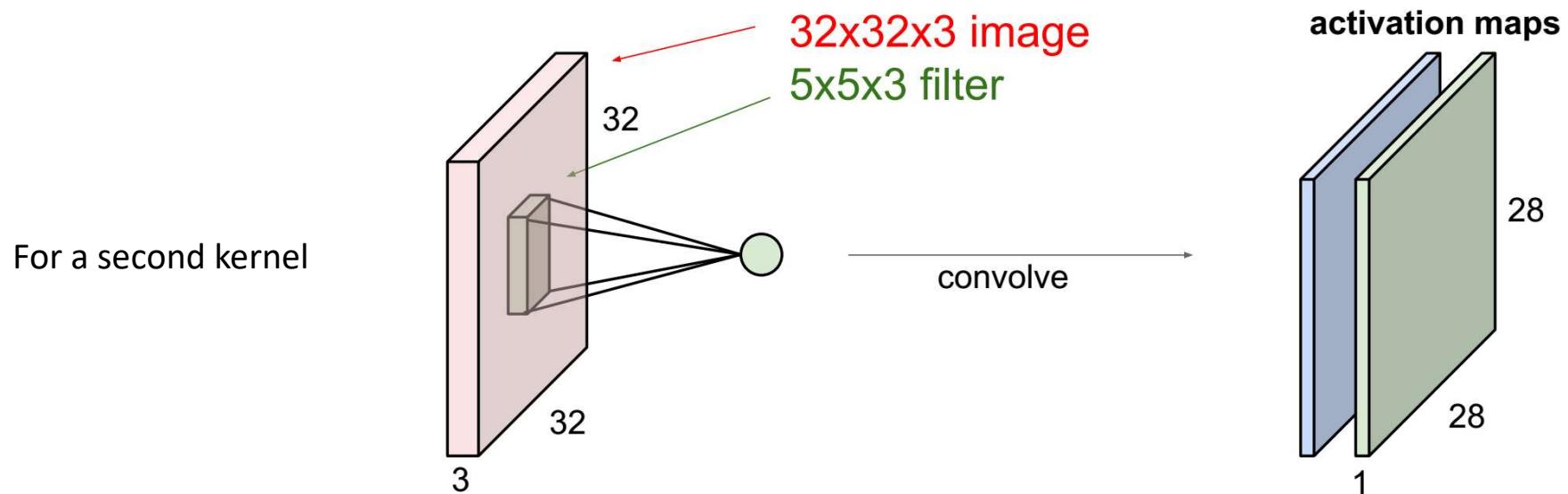
Convolutional Layers

- We can have several kernels in the same layer
- Each kernel generates an activation map



Convolutional Layers

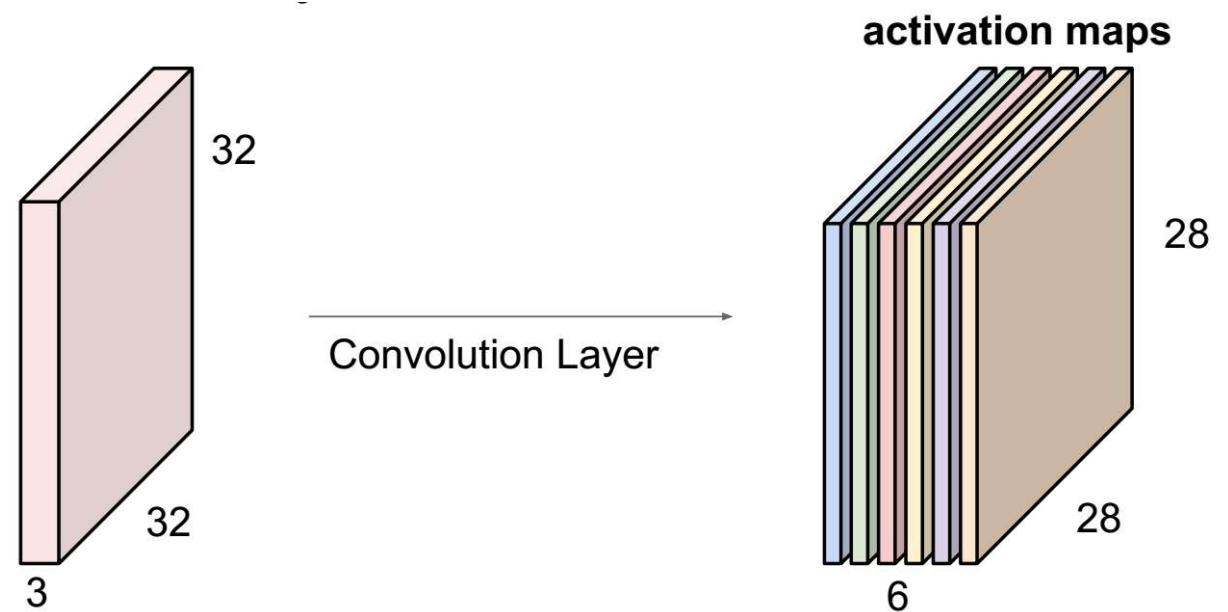
- We can have several kernels in the same layer
- Each kernel generates an activation map



Convolutional Layers

- We can have several kernels in the same layer
- Each kernel generates an activation map

If we use, six kernels, the output activation map will have depth 6



Convolutional Layers

- Parameters
 - In a linear layer, the parameters are weights and the bias, and the forward operation is as follows

$$z_L = X_{L-1} \cdot W_L + b_L$$

- The linear operation is the dot product between the input and the weights
- In the convolutional layer, we have

$$z_L = X_{L-1} * W_L + b_L$$

where the linear operation is the convolution, and the output is a feature map. W_L represents the kernel and the bias is still a single value.

Convolutional Layers

- How much parameters do we have if my convolutional layer has m kernels of dimension $k \times k \times d$?

$$m \times k \times k \times d + m$$

Kernel parameters Biases

Summary

- Convolutional layer is the core of the current progress in computer vision
- Convolution is a linear operation, hence we can include it in the feedforward pass with no problems
- As convolution is linear, backpropagation is straightforward

Neural networks

*Hyperparameters
and Pooling*



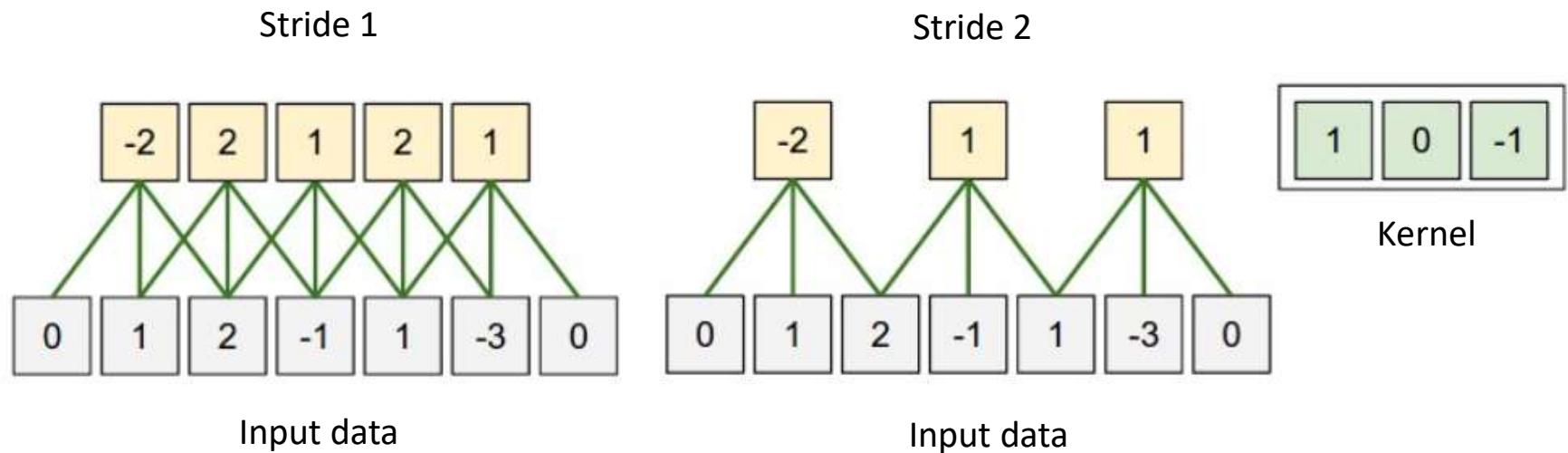
Parameters

- To build a convolutional layer, we need to set
 - K : the number of kernels in the layer
 - F : the spatial extension of the kernel
 - S : the stride (number of pixels to jump during sliding convolution)
 - P : the number f pixels for padding

If your input volumen has size W , the size of the activation map in the convolutional layer is

$$\frac{W - F + 2P}{S} + 1$$

Example - Stride



Complete example

Red channel

Input Volume (+pad 1) (7x7x3)	Filter W0 (3x3x3)
x[:, :, 0]	w0[:, :, 0]
0 0 0 0 0 0 0 0 0 2 1 0 2 0 0 1 1 1 2 2 0 0 2 1 2 1 2 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0	1 1 1 1 1 0 0 -1 -1 1 0 0 1 0 0 1 1 0
x[:, :, 1]	w0[:, :, 1]
0 0 0 0 0 0 0 0 1 2 0 0 1 0 0 1 1 2 1 2 0 0 2 1 0 0 1 0 0 2 1 2 1 0 0 0 1 2 1 2 1 0 0 0 0 0 0 0 0	1 0 0 -1 0 1 0 0 -1
x[:, :, 2]	w0[:, :, 2]
0 0 0 0 0 0 0 0 1 1 0 2 0 0 0 2 2 0 2 1 0 0 2 0 2 0 0 0 0 1 1 0 2 2 0 0 0 0 0 0 0 0	1 -1 1 -1 0 1 0 0 -1

Green channel

Filter W1 (3x3x3)	Output Volume (3x3x2)
w1[:, :, 0]	o[:, :, 0]
-1 0 -1 0 -1 -1 0 1 0	-1 5 2 5 14 10 5 9 4
w1[:, :, 1]	o[:, :, 1]
1 -1 0 0 -1 -1 1 1 1	-1 5 -3 -3 1 -8 -4 -4 -2
w1[:, :, 2]	o[:, :, 2]
0 0 1 -1 0 1 -1 -1 1	0 0 1 -1 0 1 -1 -1 1

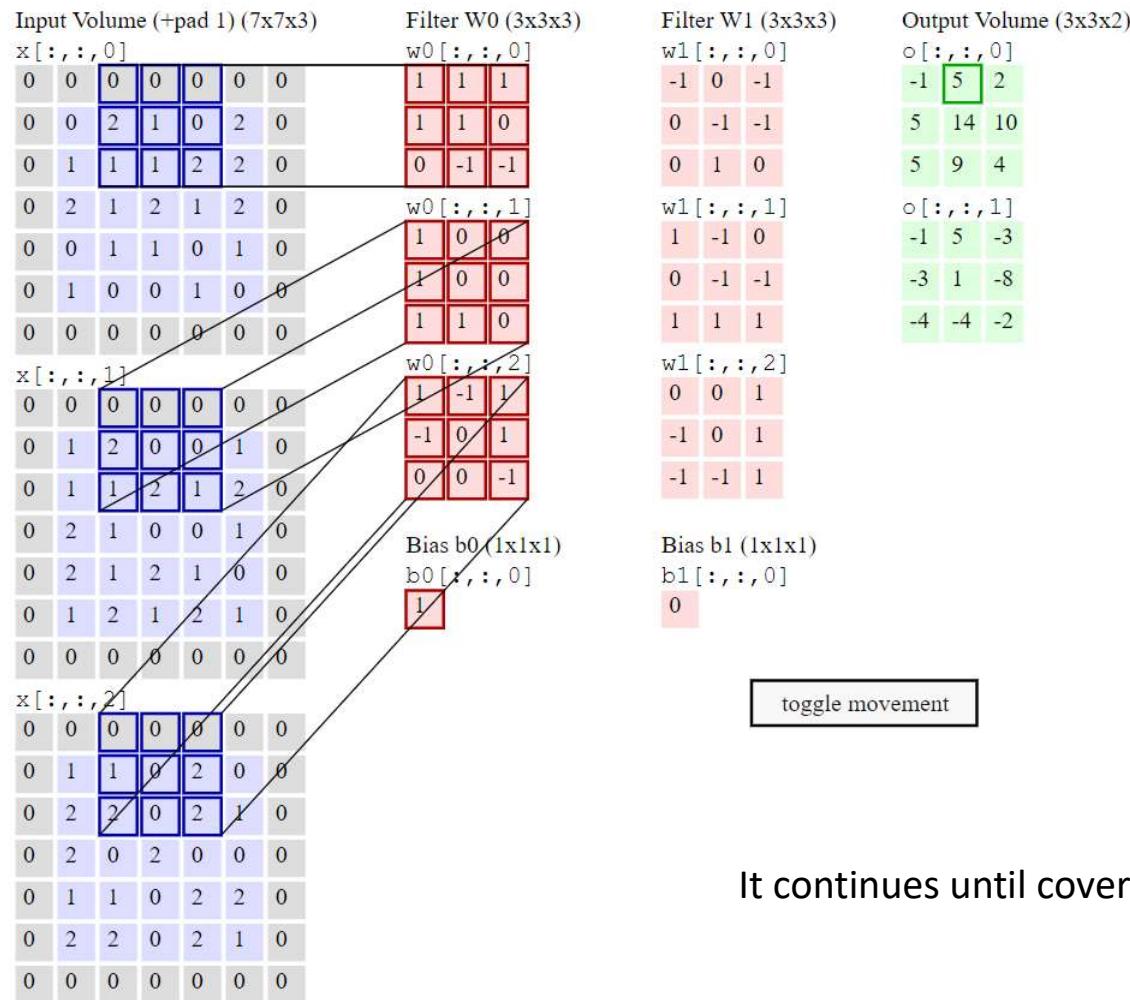
Bias b0 (1x1x1)
b0[:, :, 0]

Bias b1 (1x1x1)
b1[:, :, 0]

Blue channel

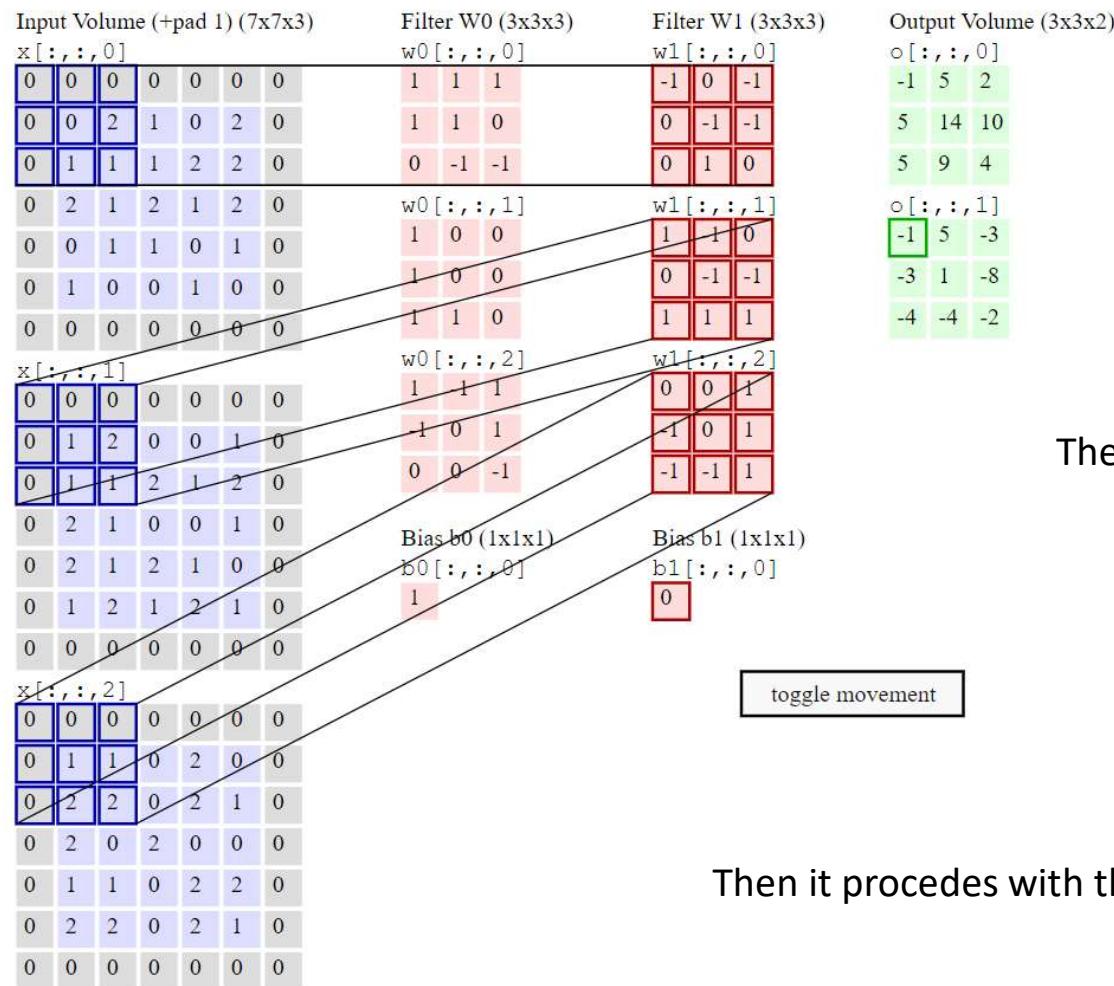
toggle movement

Complete example



It continues until covering the complete image

Complete example

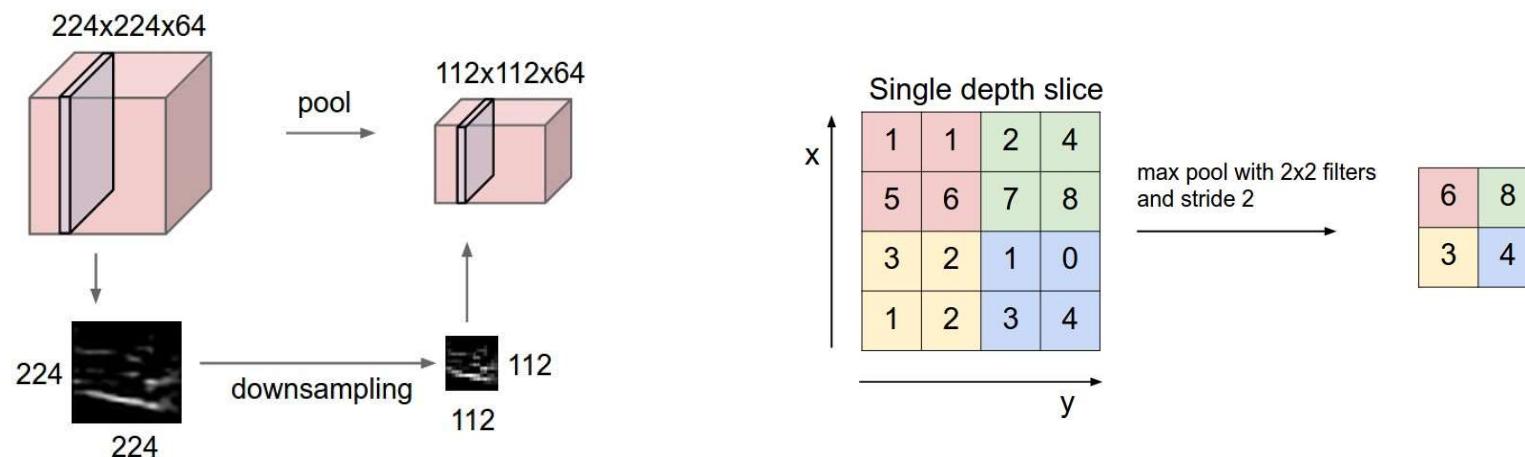


Then it proceeds with the second kernel

The output volumen is 3x3x2

Pooling Layer

- The goal is to reduce the spatial size of the representation
 - Reduces the amount of parameters
 - Reduces the computation
 - Allows us to find multi-scale representation in images
- Max pooling, size 2, stride 2 is the typical choice



Summary

- Hyperparameters for ConvNets
- Volume computation
- Pooling layers

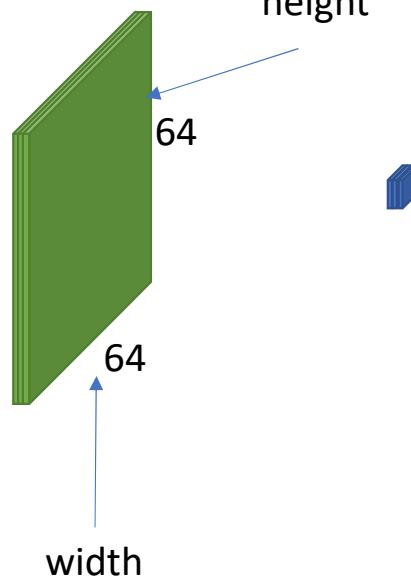
Neural networks

Convolution examples

Input image $64 \times 64 \times 3$ Kernel $5 \times 5 \times 3$

Input depth = kernel depth

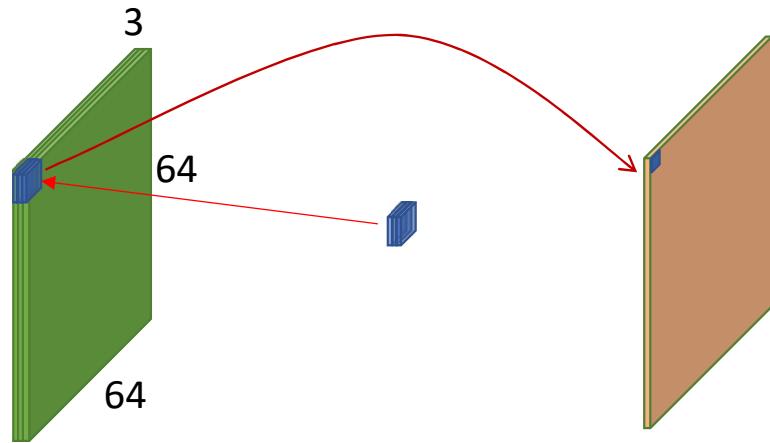
Depth $\rightarrow 3$



Input image 64x64x3

Kernel 5x5x3

Input depth = kernel depth

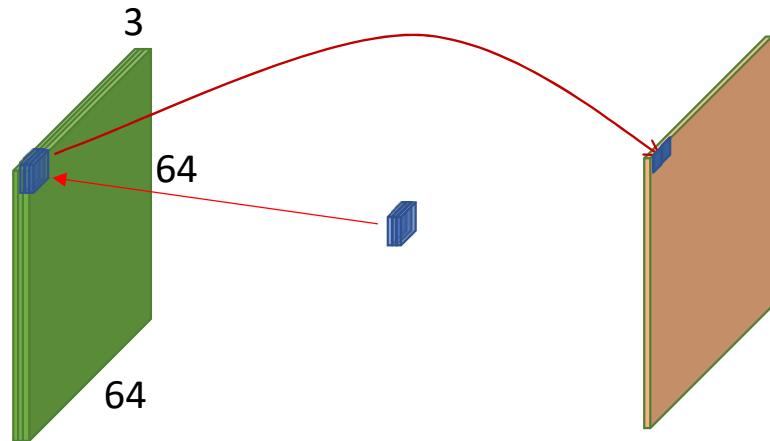


The convolution of the kernel with the local patch
in the image outputs only one value

Input image 64x64x3

Kernel 5x5x3

Input depth = kernel depth

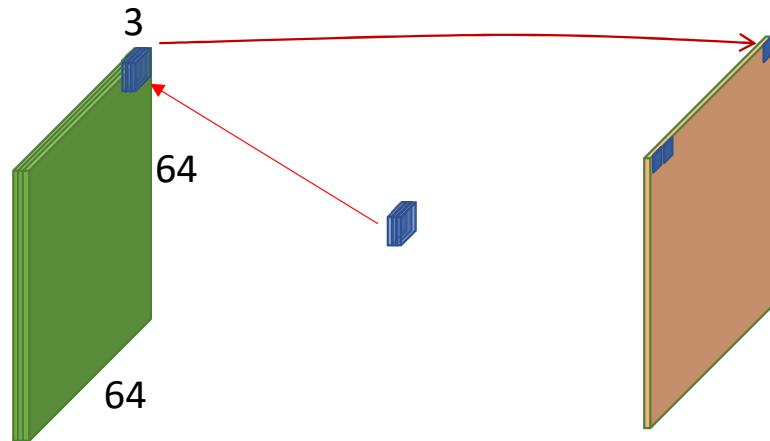


The convolution of the kernel with the local patch
in the image outputs only one value

Input image 64x64x3

Kernel 5x5x3

Input depth = kernel depth

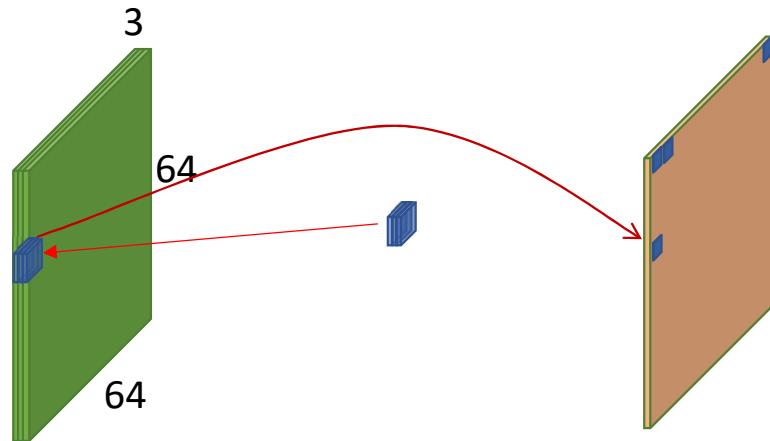


The convolution of the kernel with the local patch
in the image outputs only one value

Input image 64x64x3

Kernel 5x5x3

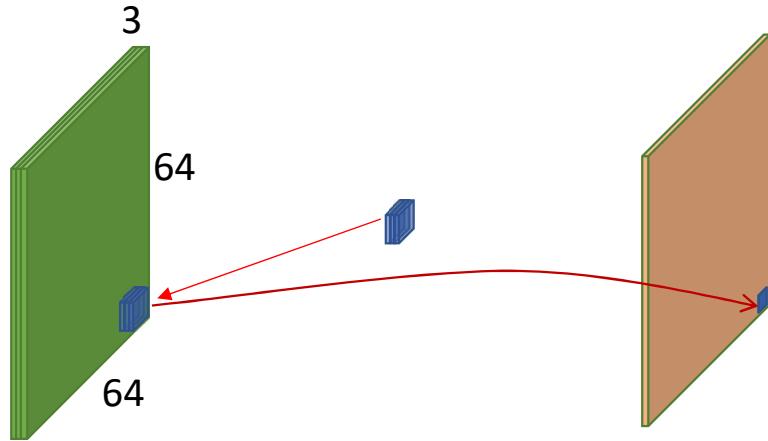
Input depth = kernel depth



The convolution of the kernel with the local patch
in the image outputs only one value

Input image 64x64x3

Kernel 5x5x3



The convolution of the kernel with the local patch in the image outputs only one value

Input depth = kernel depth

The final result of a convolution is a *feature map*

Feature map: 60x60x1 (with no padding)

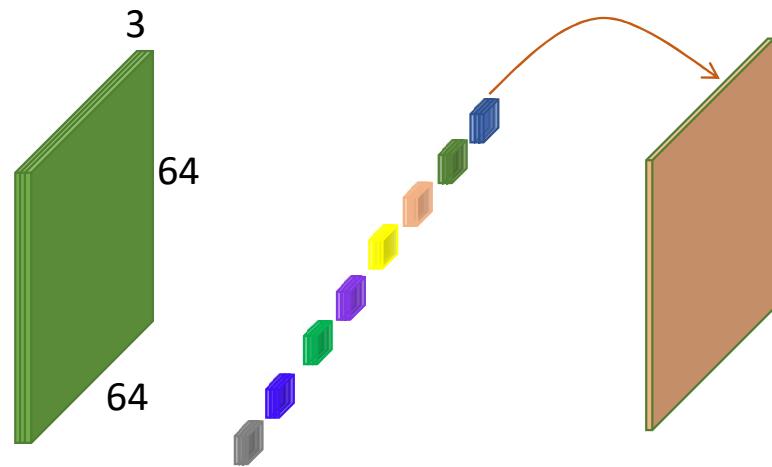
Feature map: 64x64x1 (with padding=2)

One kernel produces one feature map of depth 1

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

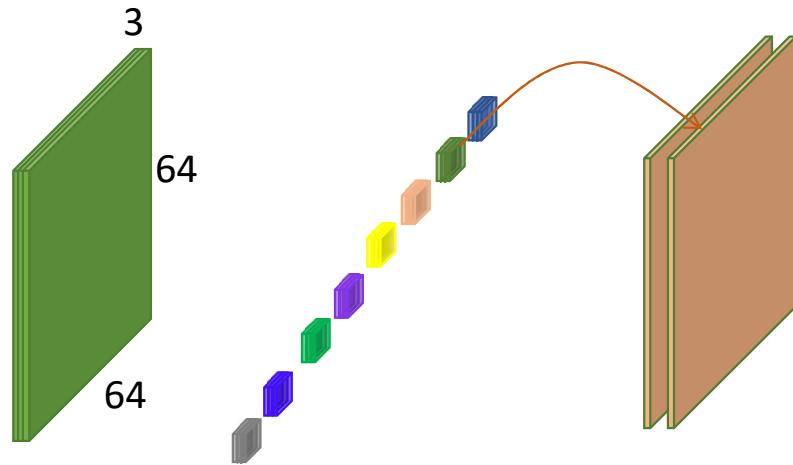


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

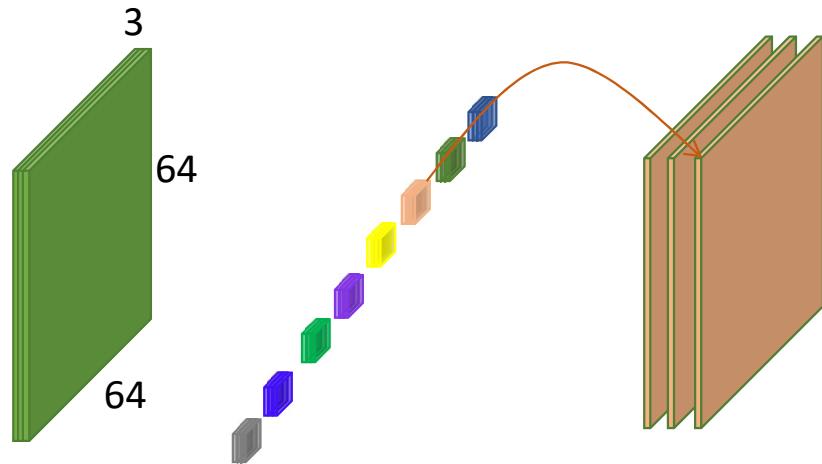


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

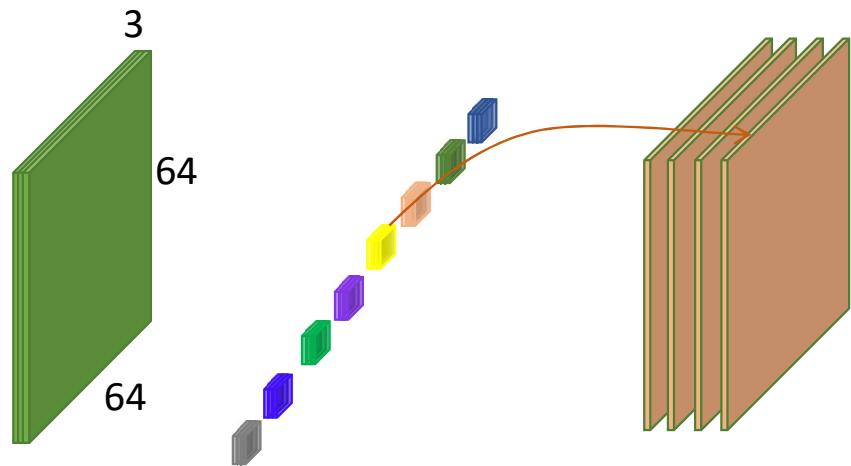


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

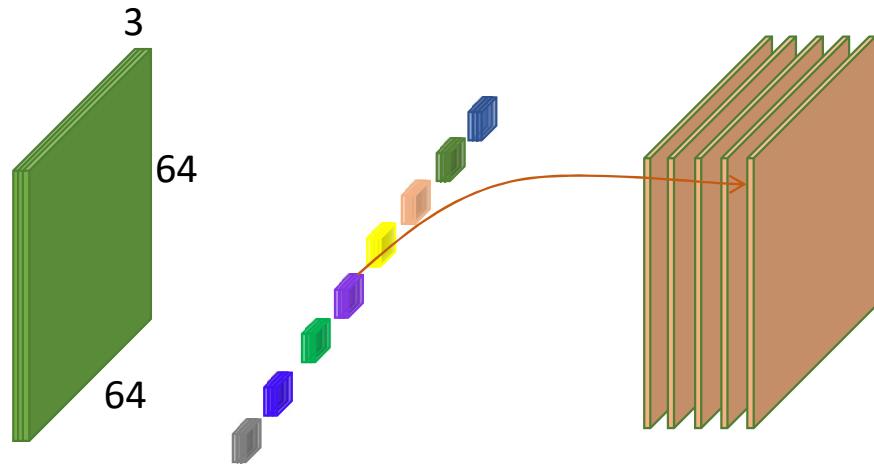


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

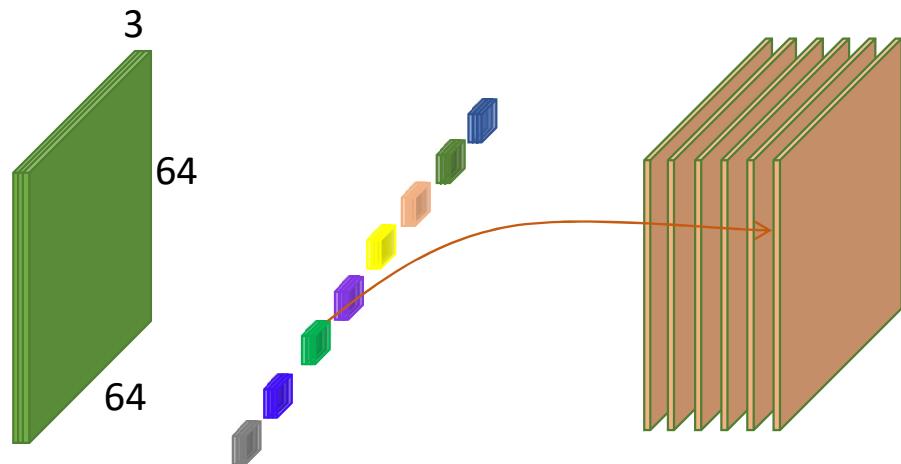


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

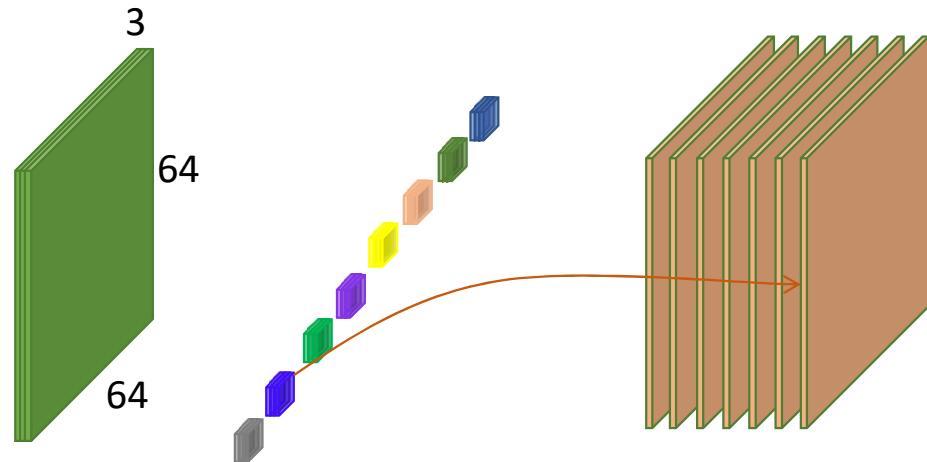


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth

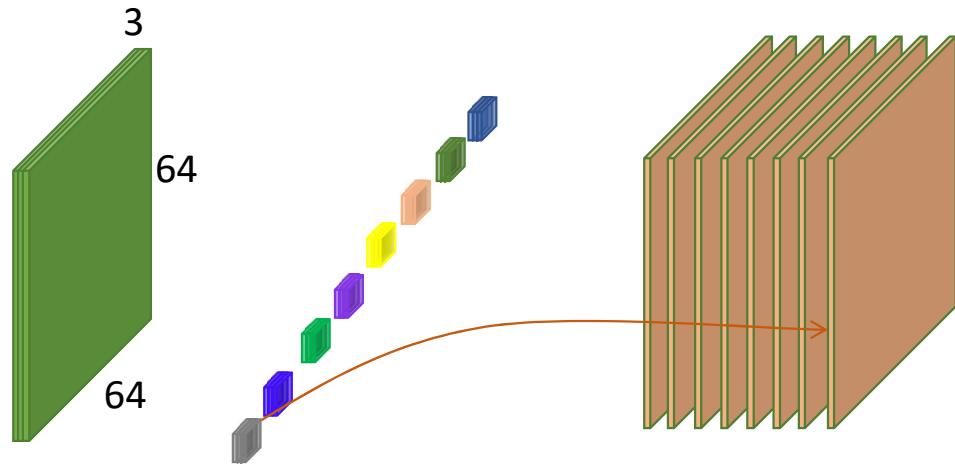


8 kernels of 5x5x3

Input image 64x64x3

kernel 5x5x3

Input depth = kernel depth



Output is a *volume* of 64x64x8

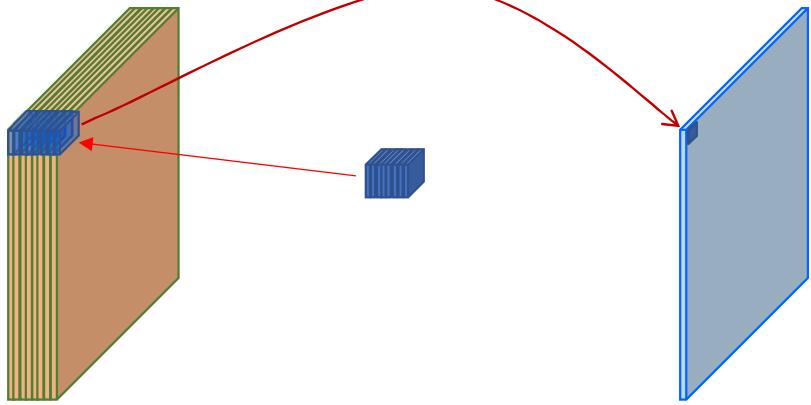
First convolutional layer

- Input: volume of 64x64x3
- Kernels: 8 of 5x5x3 with padding=1
- Output: volumen of 64x64x8

Second convolutional layer

Input: volume $64 \times 64 \times 8$

Kernel: volume $5 \times 5 \times 8$



Second convolutional layer

Input: volume $64 \times 64 \times 8$

Kernel: volume $5 \times 5 \times 8$

