

Deep Learning (CS324)

5. Convolutional Neural Networks*

Prof. Jianguo Zhang

SUSTech

*Based on <http://www.deeplearningbook.org> chapter 9 + referenced papers



What's special about images?



HEIGHT

↔

WIDTH



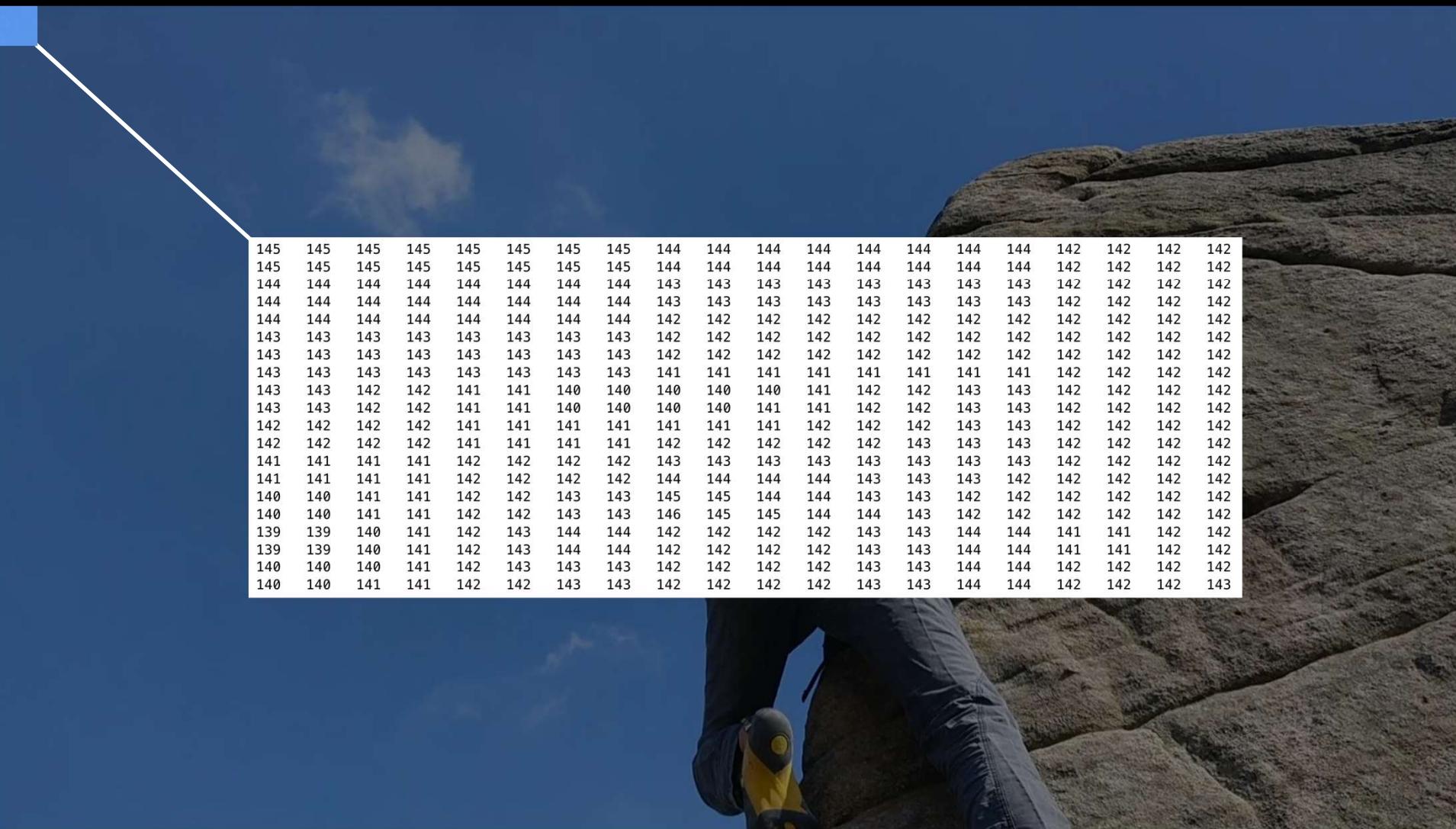
ATIAL STRUCTURE



What does the computer see?



What does the computer see?



A photograph showing a close-up of a person's lower legs and feet. They are wearing dark blue jeans and grey sneakers with yellow accents. The background is a bright blue sky with a few wispy clouds and a large, light-colored rock formation.

145	145	145	145	145	145	145	145	144	144	144	144	144	144	144	144	144	142	142	142	142	
145	145	145	145	145	145	145	145	144	144	144	144	144	144	144	144	144	142	142	142	142	
144	144	144	144	144	144	144	144	143	143	143	143	143	143	143	143	143	142	142	142	142	
144	144	144	144	144	144	144	144	143	143	143	143	143	143	143	143	143	142	142	142	142	
144	144	144	144	144	144	144	144	142	142	142	142	142	142	142	142	142	142	142	142	142	
143	143	143	143	143	143	143	143	142	142	142	142	142	142	142	142	142	142	142	142	142	
143	143	143	143	143	143	143	143	142	142	142	142	142	142	142	142	142	142	142	142	142	
143	143	143	143	143	143	143	143	141	141	141	141	141	141	141	141	141	141	142	142	142	
143	143	143	142	142	141	141	140	140	140	140	140	140	140	140	140	140	142	142	142	142	
143	143	142	142	142	141	141	140	140	140	140	140	140	140	140	140	140	142	142	142	142	
142	142	142	142	142	141	141	141	141	141	141	141	141	141	141	141	141	142	142	142	142	
142	142	142	142	142	141	141	141	141	141	142	142	142	142	142	142	142	142	142	142	142	
141	141	141	141	141	142	142	142	142	143	143	143	143	143	143	143	143	142	142	142	142	
141	141	141	141	141	142	142	142	142	144	144	144	144	144	143	143	143	142	142	142	142	
140	140	141	141	141	142	142	143	143	145	145	145	144	144	144	143	143	142	142	142	142	
140	140	141	141	141	142	142	143	143	146	145	145	144	144	143	142	142	142	142	142	142	
139	139	140	141	142	143	144	144	142	142	142	142	143	143	143	143	144	144	141	141	142	142
139	139	140	141	142	143	144	144	142	142	142	142	143	143	143	144	144	141	141	142	142	142
140	140	140	141	141	142	143	143	142	142	142	142	143	143	143	144	144	142	142	142	142	142
140	140	141	141	141	142	142	143	143	142	142	142	142	143	143	144	144	142	142	142	142	143

First $20 \times 20 = 400$ pixels

1920 (w) x 1080 (h) pixels = 2,073,600 input variables



1920 (w) x 1080 (h) x 3 (RGB) pixels = 6,220,800 input variables



1920 (w) x 1080 (h) x 3 (RGB) pixels = 6,220,800 input variables



What if you feed this to an MLP with 1 hidden layer of 100 units?

1920 (w) x 1080 (h) x 3 (RGB) pixels = 6,220,800 input variables



10 MILLION PARAMETERS

What if you feed this to an MLP with 1 hidden layer of 100 units?

Spot the difference...



Spot the difference...



Spot the difference...



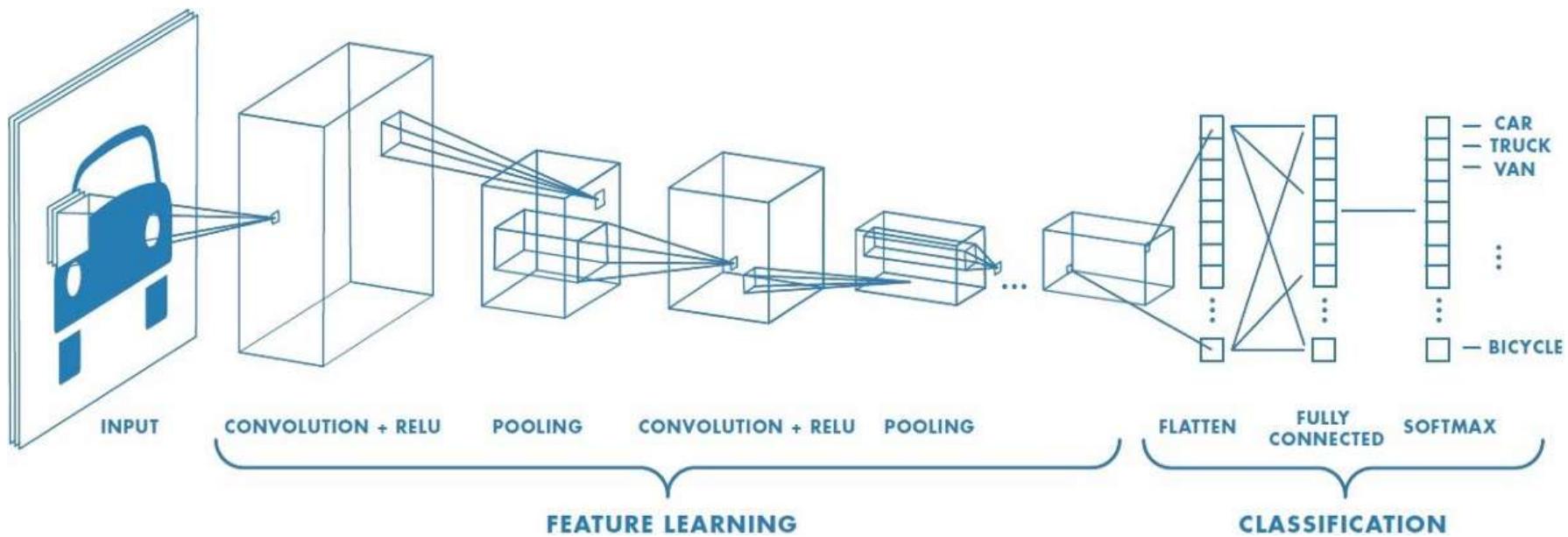
We shifted it of 1 px to the right

Spot the difference...

G CHANGE OF INPUT VECTOR

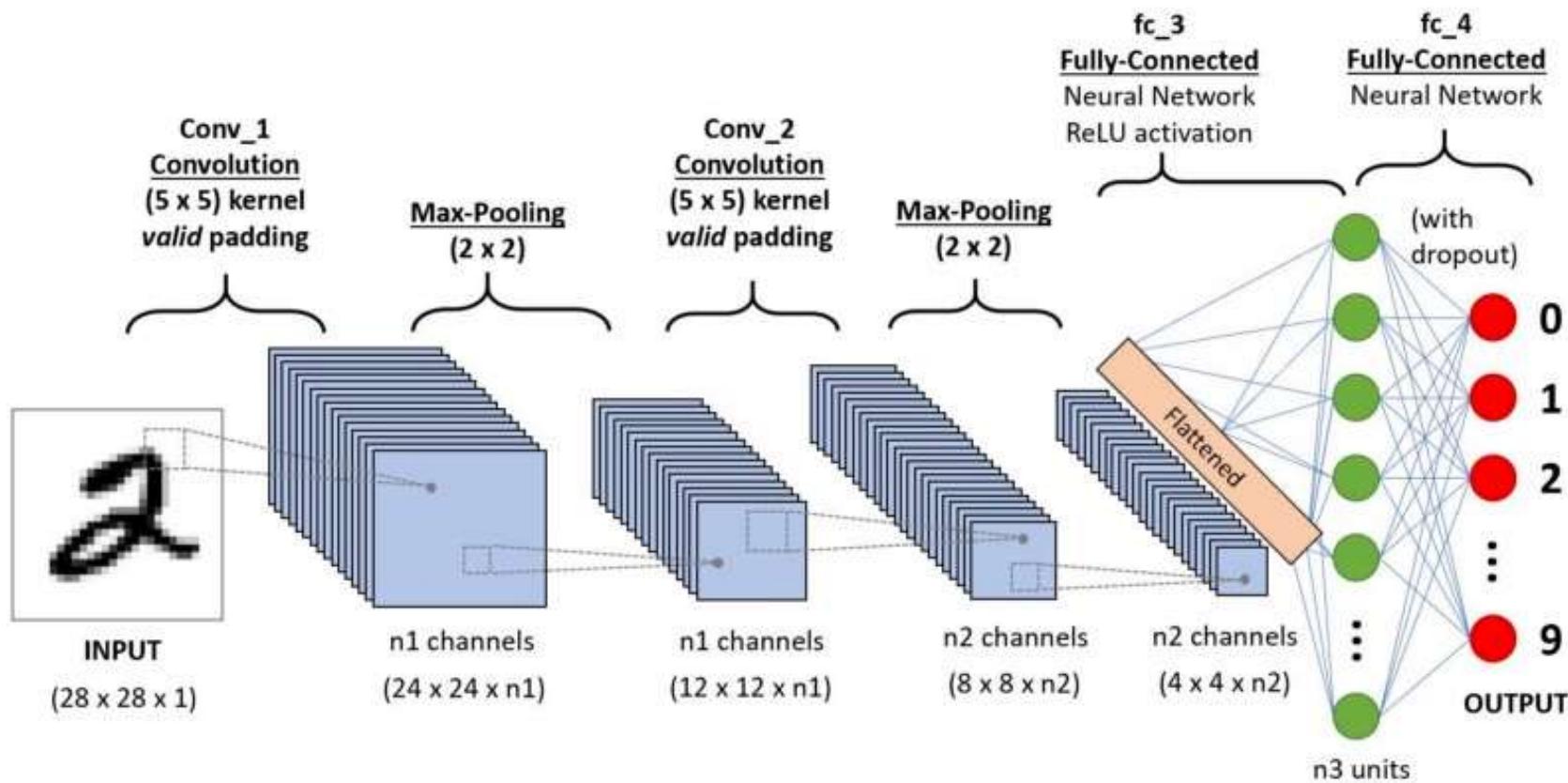
We shifted it of 1 px to the right

Convolutional neural networks



Source: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Convolutional neural networks



Source: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

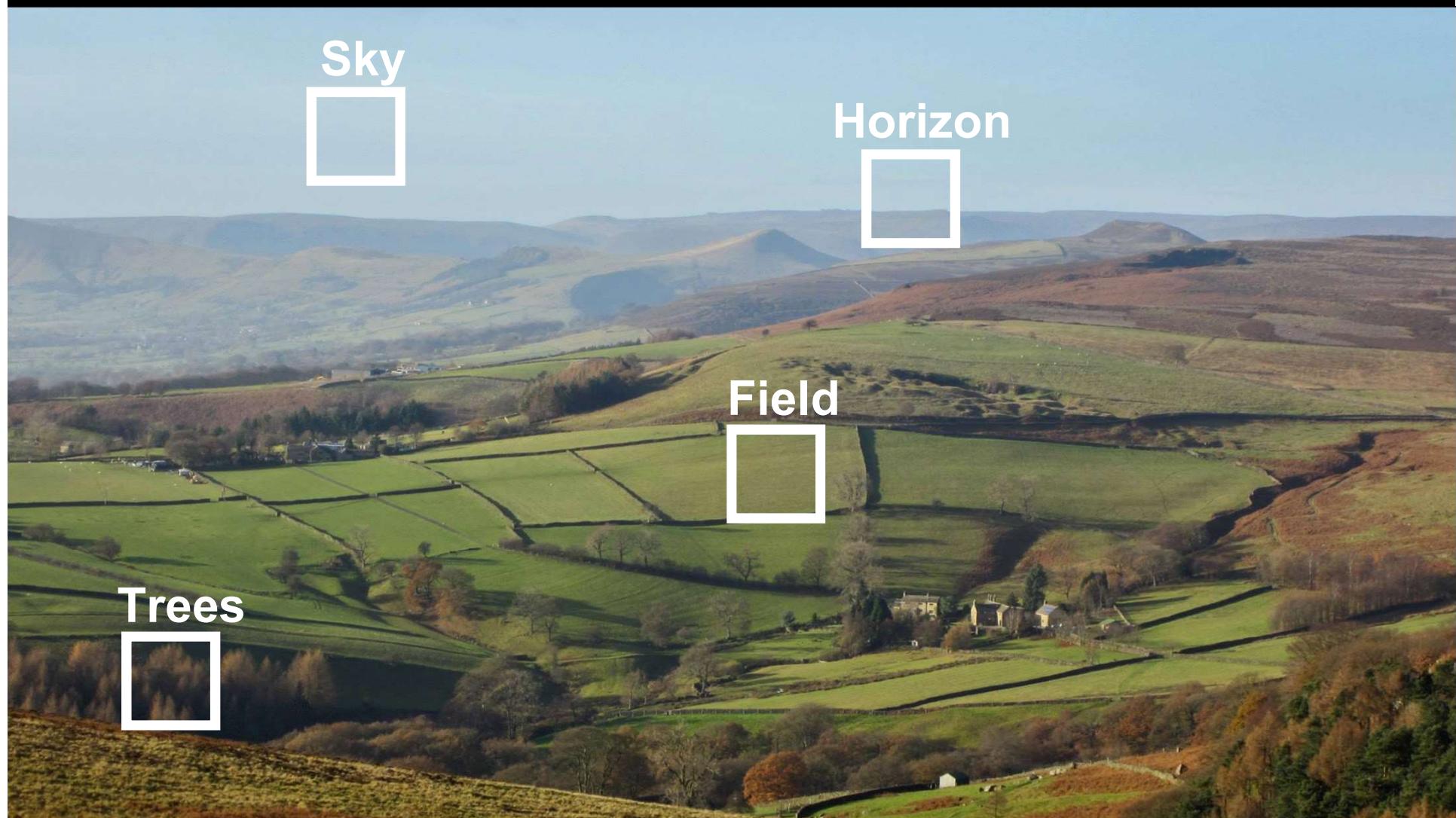
Convolutional neural networks

- Preserve spatial structure by
 - Convolutional filters
- Scale up to process very large inputs (laid out on a grid) using
 - Sparse connections
 - Parameter sharing
- Robust to local variances thanks to
 - Spatial pooling

Neighbouring variables are locally correlated



Neighbouring variables are locally correlated



We say it already here...

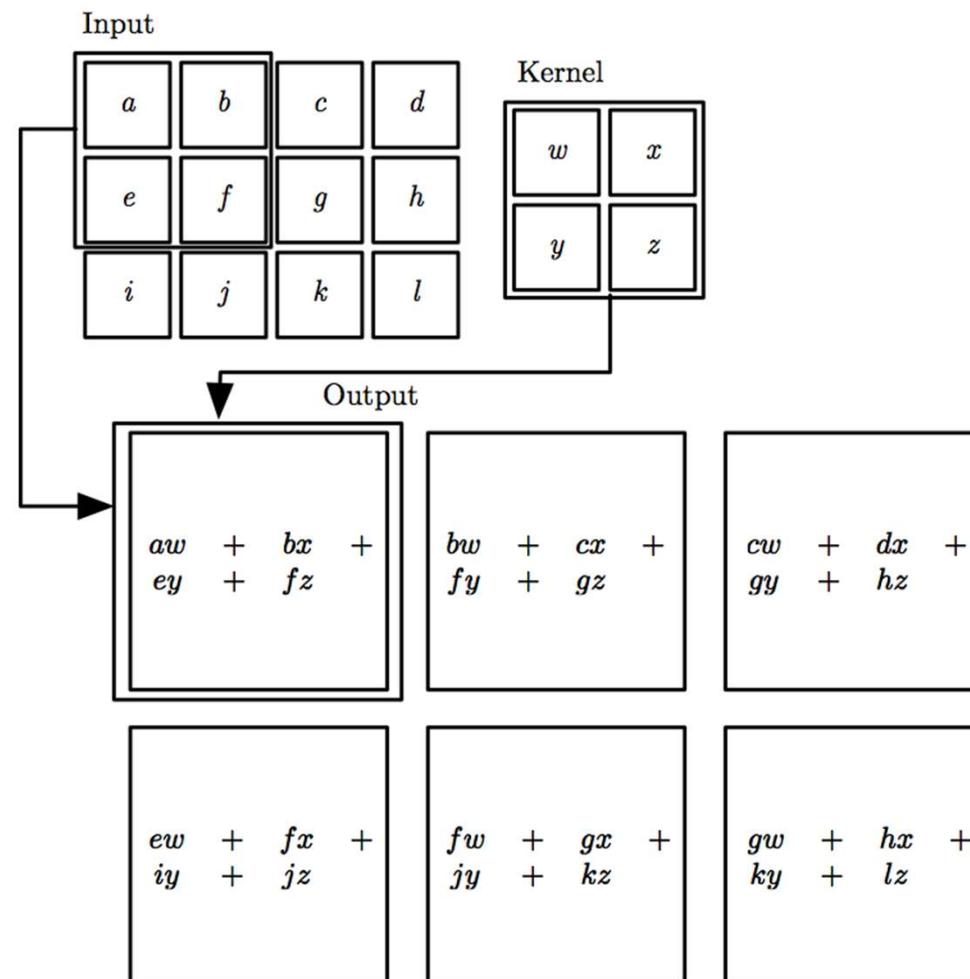
145	145	145	145	145	145	145	145	144	144	144	144	144	144	144	144	144	144	142	142	142	142	
145	145	145	145	145	145	145	145	144	144	144	144	144	144	144	144	144	144	142	142	142	142	
144	144	144	144	144	144	144	144	143	143	143	143	143	143	143	143	143	143	142	142	142	142	
144	144	144	144	144	144	144	144	143	143	143	143	143	143	143	143	143	143	142	142	142	142	
144	144	144	144	144	144	144	144	142	142	142	142	142	142	142	142	142	142	142	142	142	142	
143	143	143	143	143	143	143	143	142	142	142	142	142	142	142	142	142	142	142	142	142	142	
143	143	143	143	143	143	143	143	142	142	142	142	142	142	142	142	142	142	142	142	142	142	
143	143	143	143	143	143	143	143	141	141	141	141	141	141	141	141	141	141	142	142	142	142	
143	143	143	143	143	143	143	143	140	140	140	140	140	140	140	140	140	140	142	142	142	142	
143	143	142	142	141	141	140	140	140	140	140	140	140	140	140	140	140	140	142	142	142	142	
143	143	142	142	142	141	141	140	140	140	140	140	140	140	140	140	140	140	142	142	142	142	
142	142	142	142	142	141	141	141	141	141	141	141	141	141	141	141	141	141	142	142	142	142	
142	142	142	142	142	141	141	141	141	141	142	142	142	142	142	142	142	142	143	142	142	142	
141	141	141	141	141	142	142	142	142	143	143	143	143	143	143	143	143	143	142	142	142	142	
141	141	141	141	141	142	142	142	142	144	144	144	144	144	144	144	144	144	142	142	142	142	
140	140	141	141	141	142	142	143	143	145	145	145	145	145	145	145	145	145	142	142	142	142	
140	140	141	141	141	142	142	143	143	146	146	145	145	144	144	144	144	144	142	142	142	142	
139	139	140	141	142	143	144	144	142	142	142	142	143	143	143	143	143	144	144	141	141	142	142
139	139	140	141	142	143	144	144	142	142	142	142	143	143	143	143	143	144	144	141	141	142	142
140	140	140	141	141	142	143	143	142	142	142	142	143	143	143	143	143	144	144	142	142	142	142
140	140	141	141	141	142	142	143	143	142	142	142	142	143	143	143	143	144	144	142	142	142	143



Key idea of CNNs

- **Replace first layers of matrix multiplications with convolution operations**
- Everything else stays the same
 - Maximum likelihood
 - Back-propagation
 - etc.

2D convolution



2D convolution

1 <small>$\times 1$</small>	1 <small>$\times 0$</small>	1 <small>$\times 1$</small>	0	0
0 <small>$\times 0$</small>	1 <small>$\times 1$</small>	1 <small>$\times 0$</small>	1	0
0 <small>$\times 1$</small>	0 <small>$\times 0$</small>	1 <small>$\times 1$</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

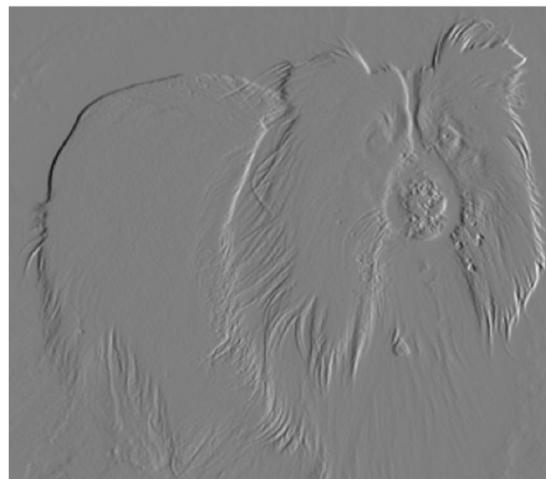
Convolved
Feature

Source: <https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59>

Edge detection by convolution



Input



Output

1	-1
---	----

Kernel

Example: Sobel filter

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A}$$

Source: https://en.wikipedia.org/wiki/Sobel_operator

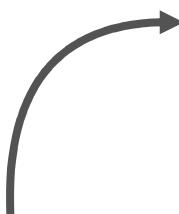
Example: Sobel filter

```
%% MATLAB code  
clear  
img = imread('test_image.jpg');  
img = rgb2gray(img);  
imshow(edge(img, 'sobel'));
```



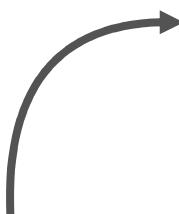
Example: Sobel filter

%% MATLAB code
clear
img = imread('test_image.jpg');
img = rgb2gray(img);
imshow(edge(img, 'sobel'));



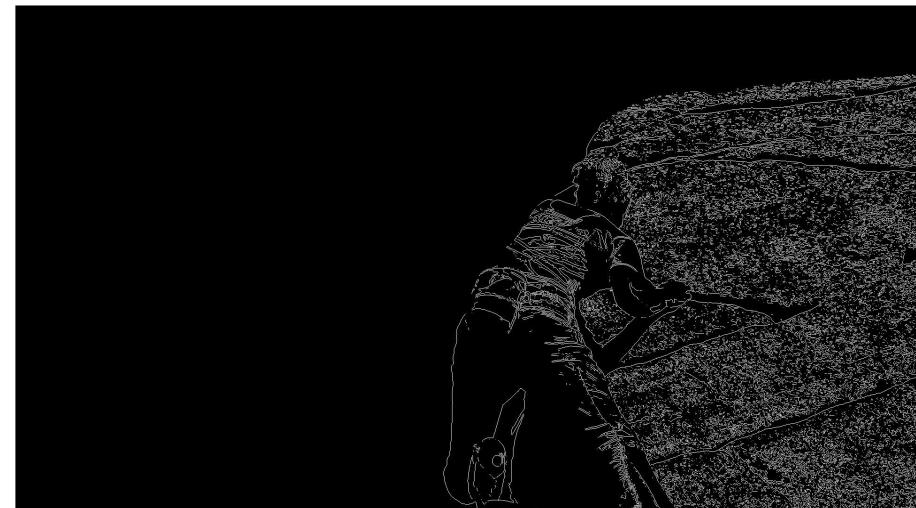
Example: Sobel filter

```
%% MATLAB code  
clear  
img = imread('test_image.jpg');  
img = rgb2gray(img);  
imshow(edge(img, 'sobel'));
```



Example: Sobel filter

```
%% MATLAB code  
clear  
img = imread('test_image.jpg');  
img = rgb2gray(img);  
imshow(edge(img, 'sobel'));
```



Example: Sobel filter

```
%% MATLAB code  
clear  
img = imread('test_image.jpg');  
img = rgb2gray(img);  
imshow(edge(img, 'sobel'));
```



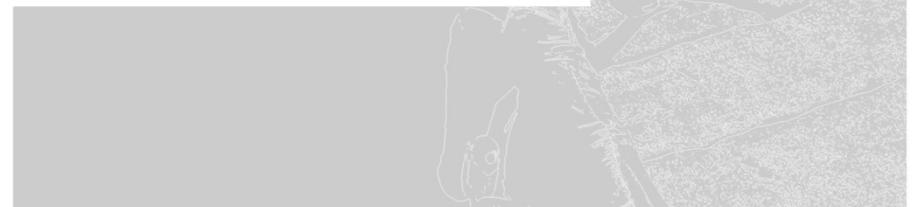
Sobel filter is handcrafted: **can we learn filters from data instead?**

Example: Sobel filter

```
%% MATLAB code
```

```
clear
```

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$



Sobel filter is handcrafted: **can we learn filters from data instead?**

Example: Sobel filter

```
%% MATLAB code
```

```
clear
```

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

$$a_{rc} = \sum_{i=-a}^a \sum_{j=-b}^b x_{r-i,c-j} \cdot w_{ij}$$

Convolution is essentially a dot product, just like linear layers

Example: Sobel filter

```
%% MATLAB code
```

```
clear
```

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

$$\frac{\partial a_{rc}}{\partial w_{ij}} = \sum_{i=-a}^a \sum_{j=-b}^b x_{r-i,c-j}$$

Example: Sobel filter

%% MATLAB code

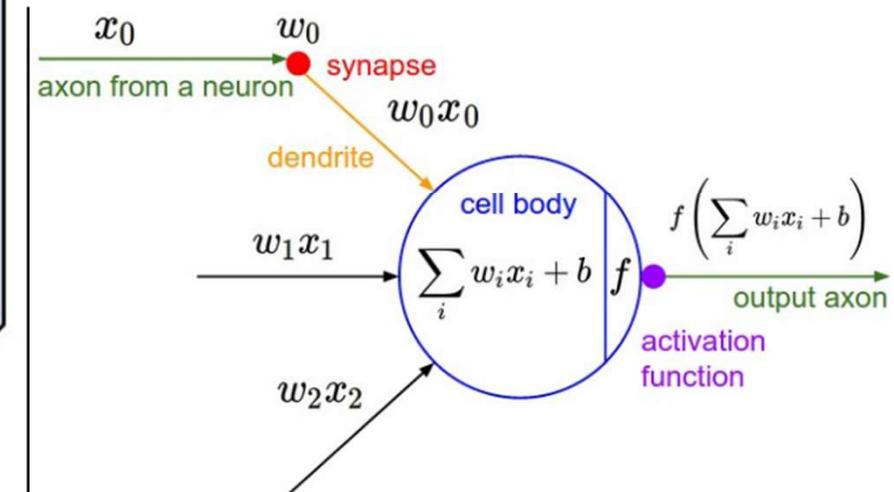
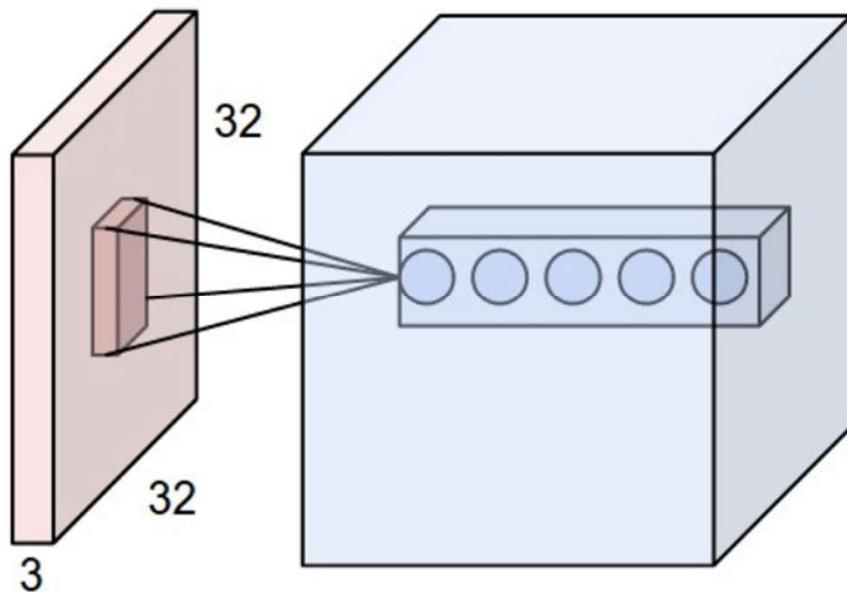
clear

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}$$

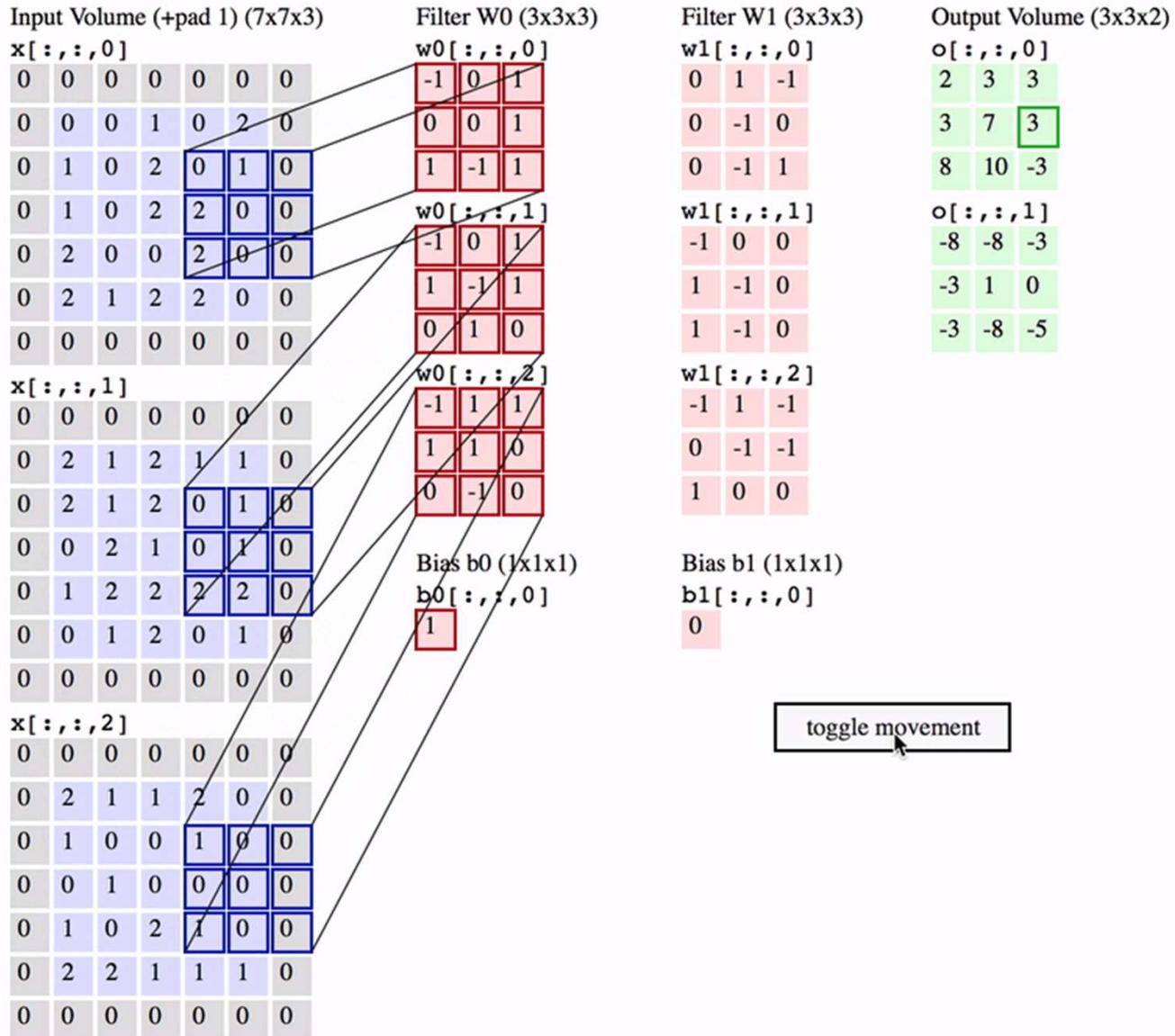
$$\frac{\partial a_{rc}}{\partial w_{ij}} = \sum_{i=-a}^a \sum_{j=-b}^b x_{r-i,c-j}$$

We can learn W

Convolutional layers



Source: <http://cs231n.github.io/convolutional-networks/>



Source: <http://cs231n.github.io/convolutional-networks/>

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

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

Kernel Channel #1

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

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

↓

310

+

↓

-170

+

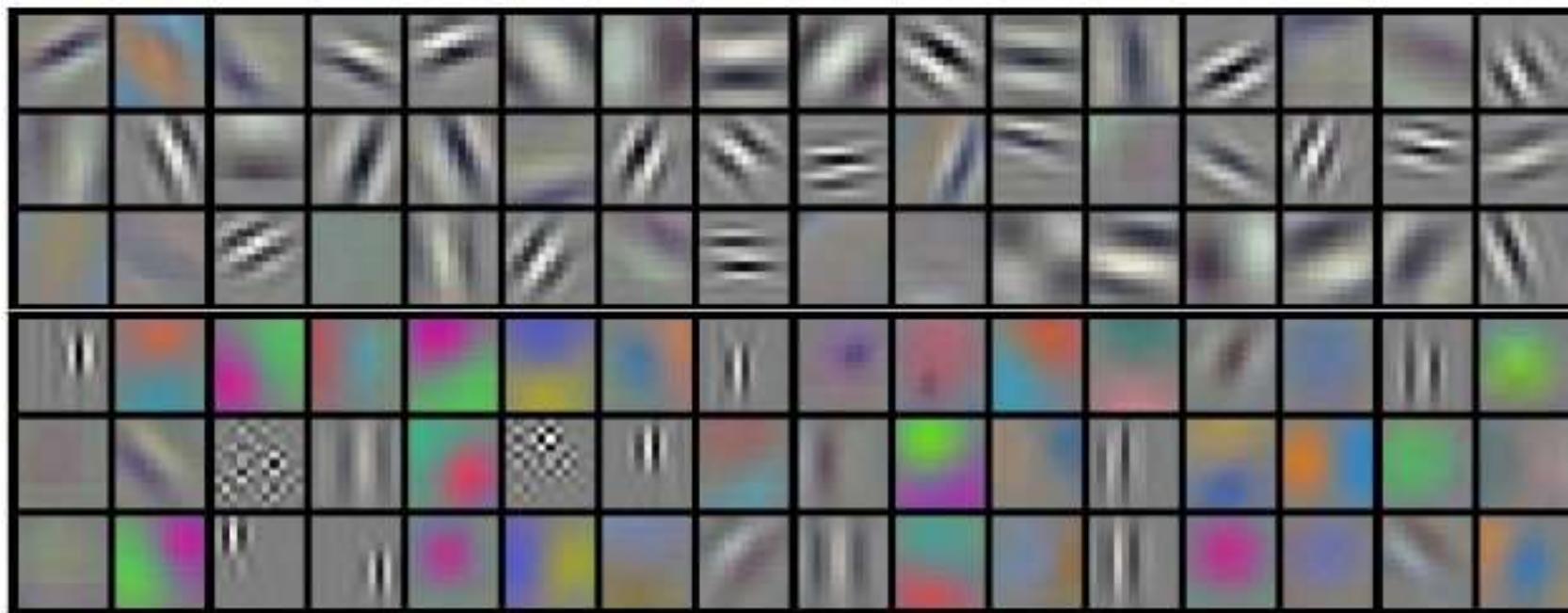
325

 $+ 1 = 466$

↑
Bias = 1

-25	466			...
				...
				...
				...
...

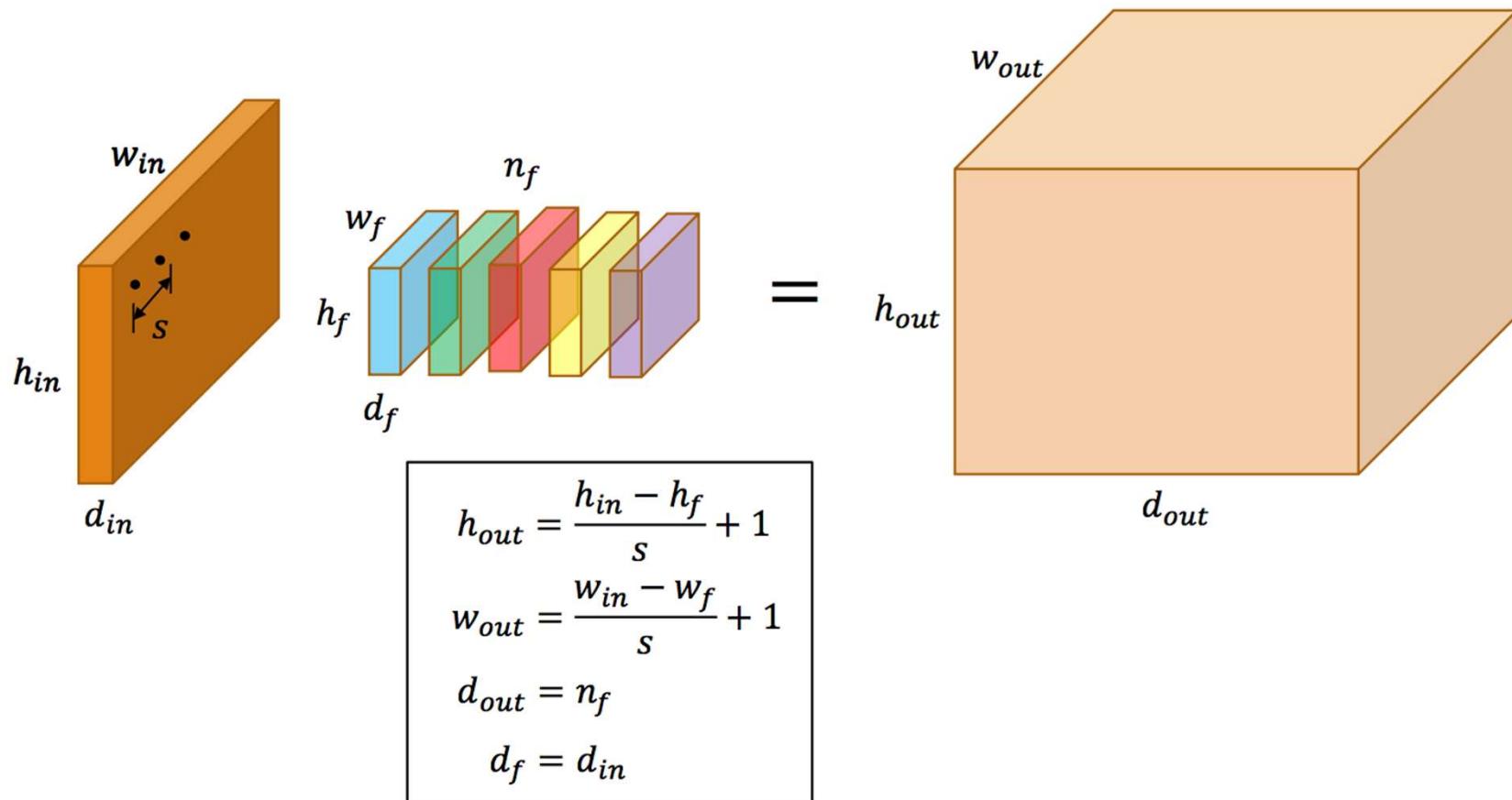
Source: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>



Example filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size [11x11x3], ε

Source: <http://cs231n.github.io/convolutional-networks/>

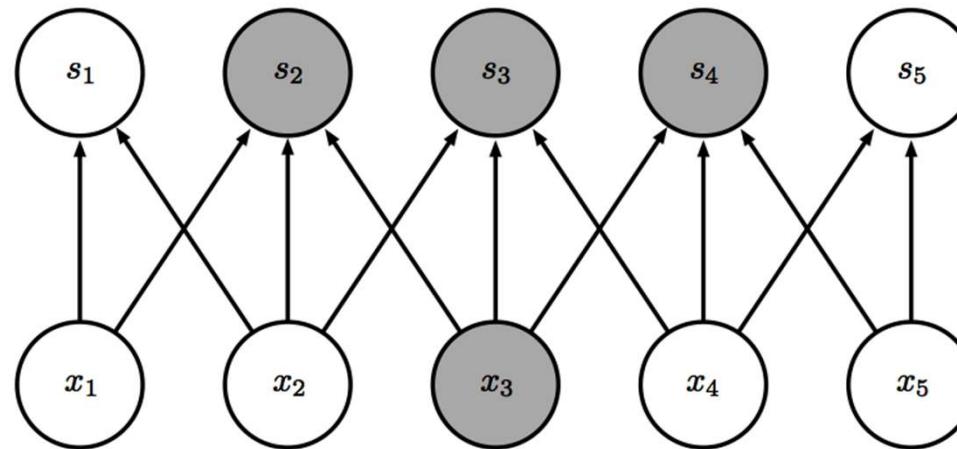
Dimensions after convolution



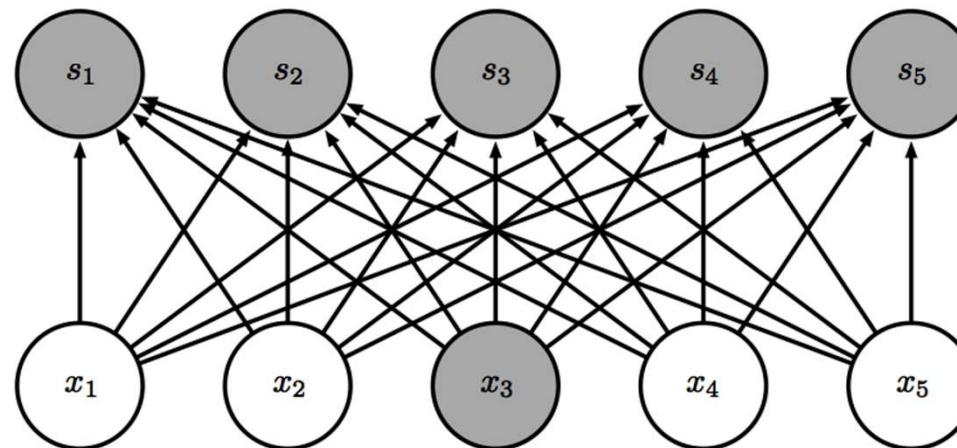
Source Dr. Efstratios Gavves UVA lectures

Sparse connectivity

Sparse
connections
due to small
convolution
kernel

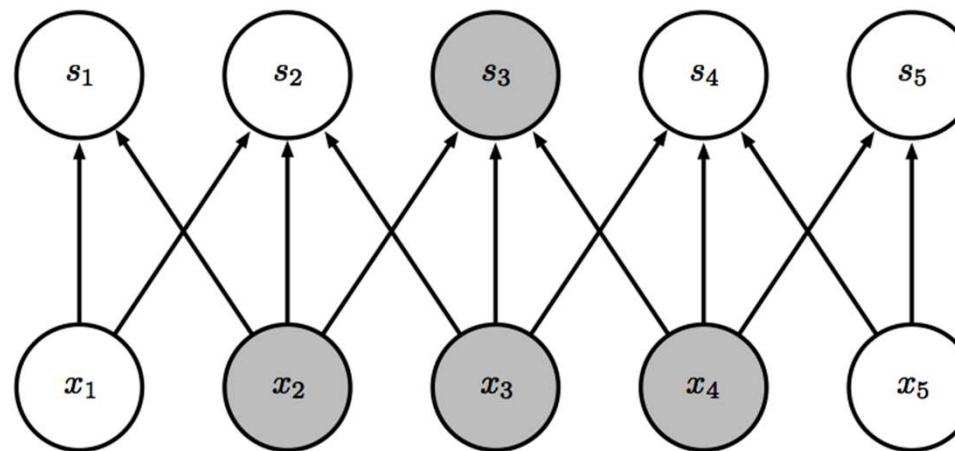


Dense
connections

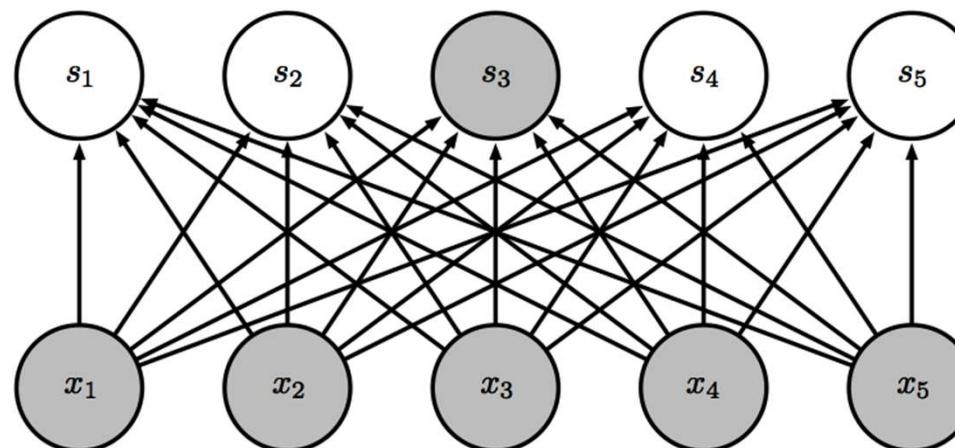


Sparse connectivity

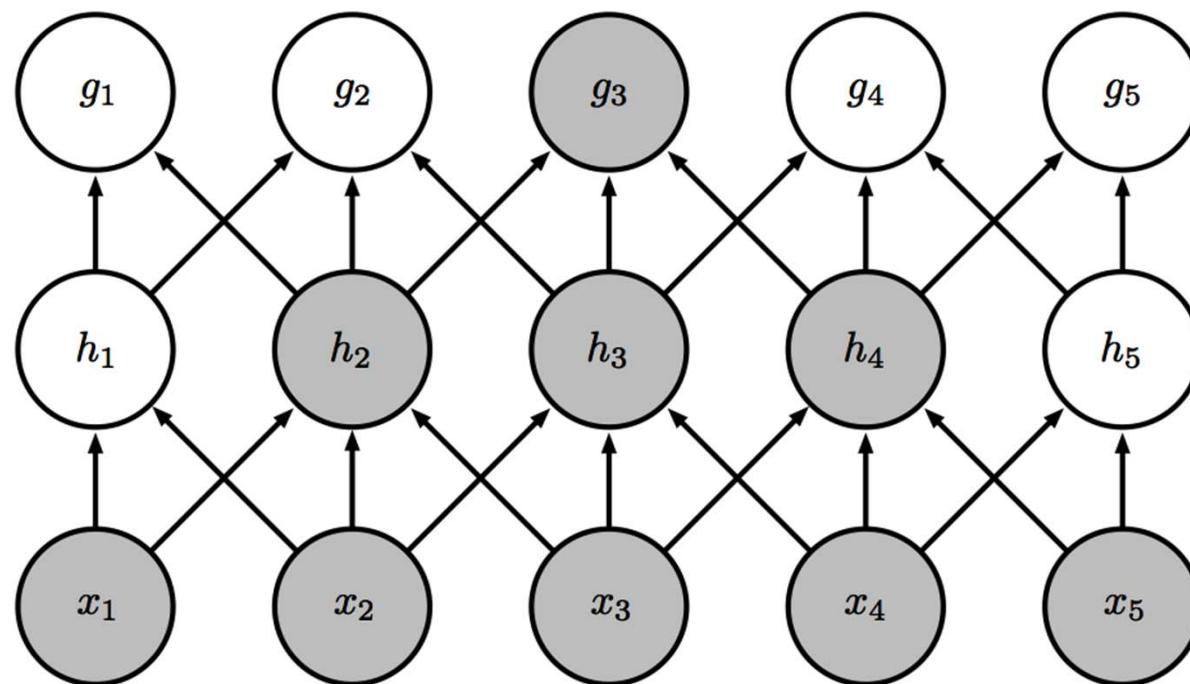
Sparse
connections
due to small
convolution
kernel



Dense
connections

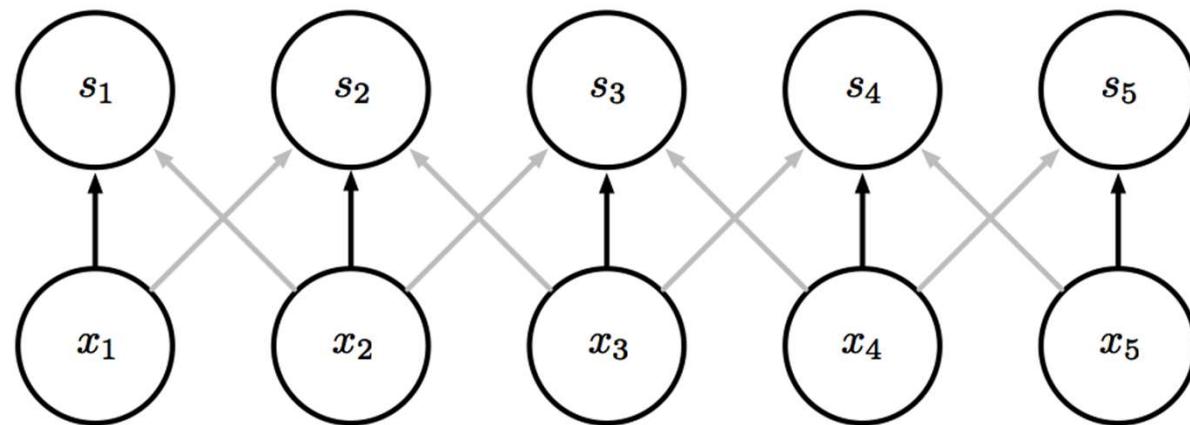


Growing receptive fields

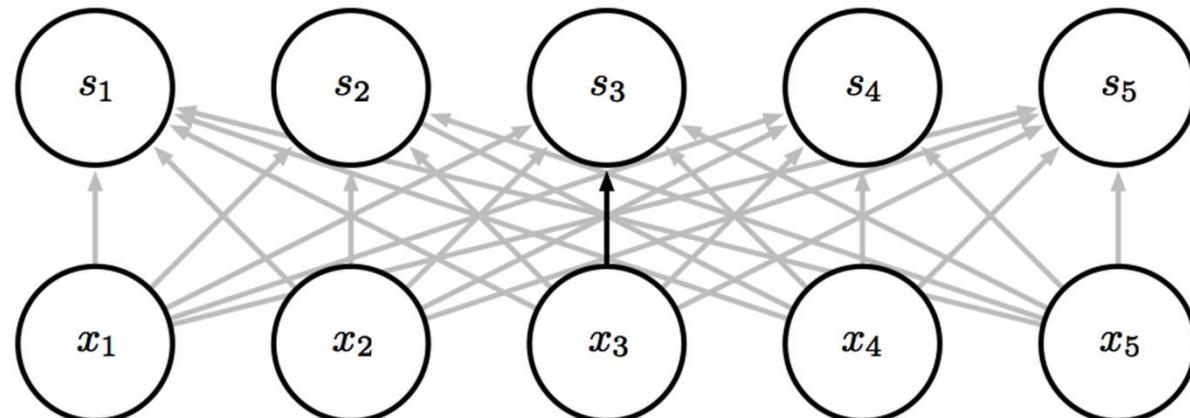


Parameter sharing

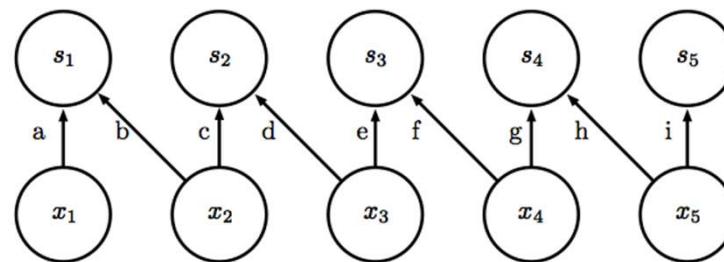
Convolution shares the same parameters across all spatial locations



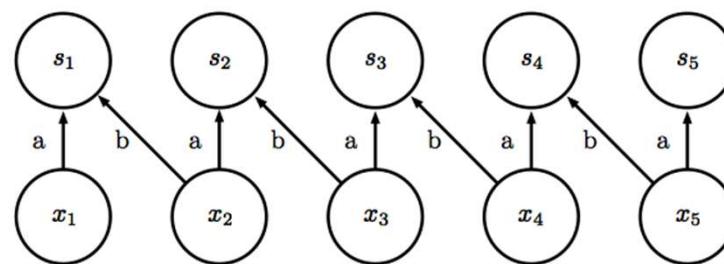
Traditional matrix multiplication does not share any parameters



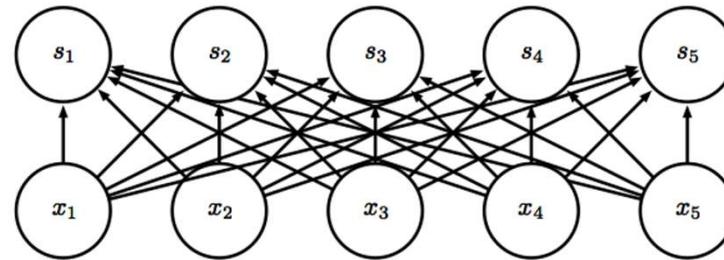
Local connectivity vs convolution



Local connection:
like convolution,
but no sharing

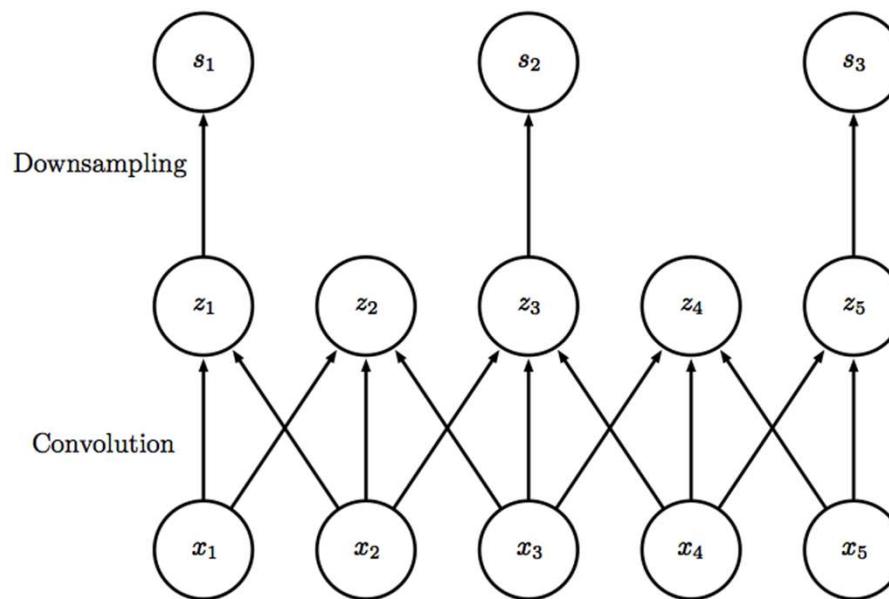
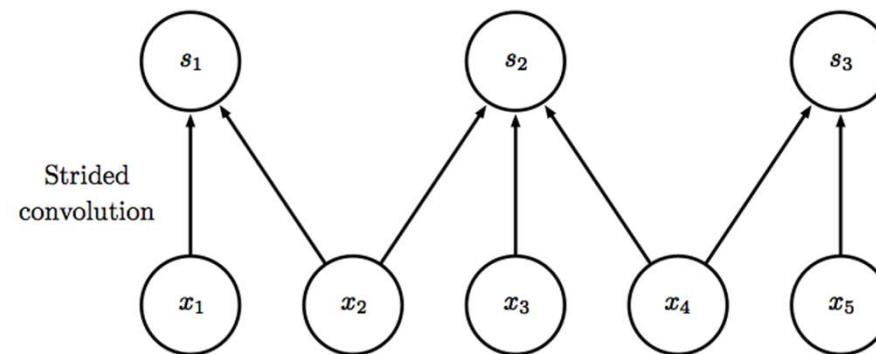


Convolution



Fully connected

Convolution with stride



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

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

Kernel Channel #1



310

+

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

Kernel Channel #2



-170

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



325

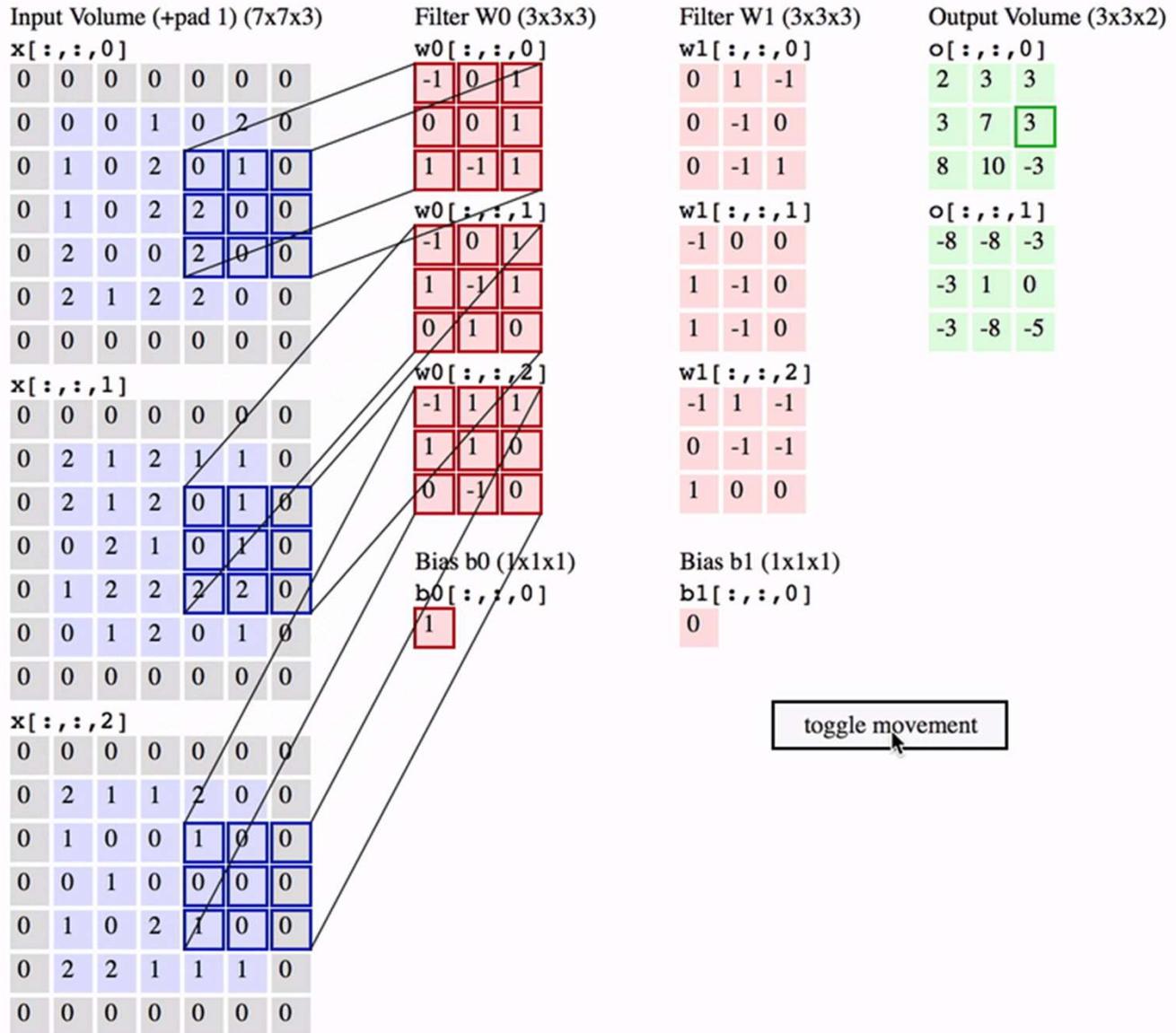
+

Output

-25	466			...
				...
				...
				...
...

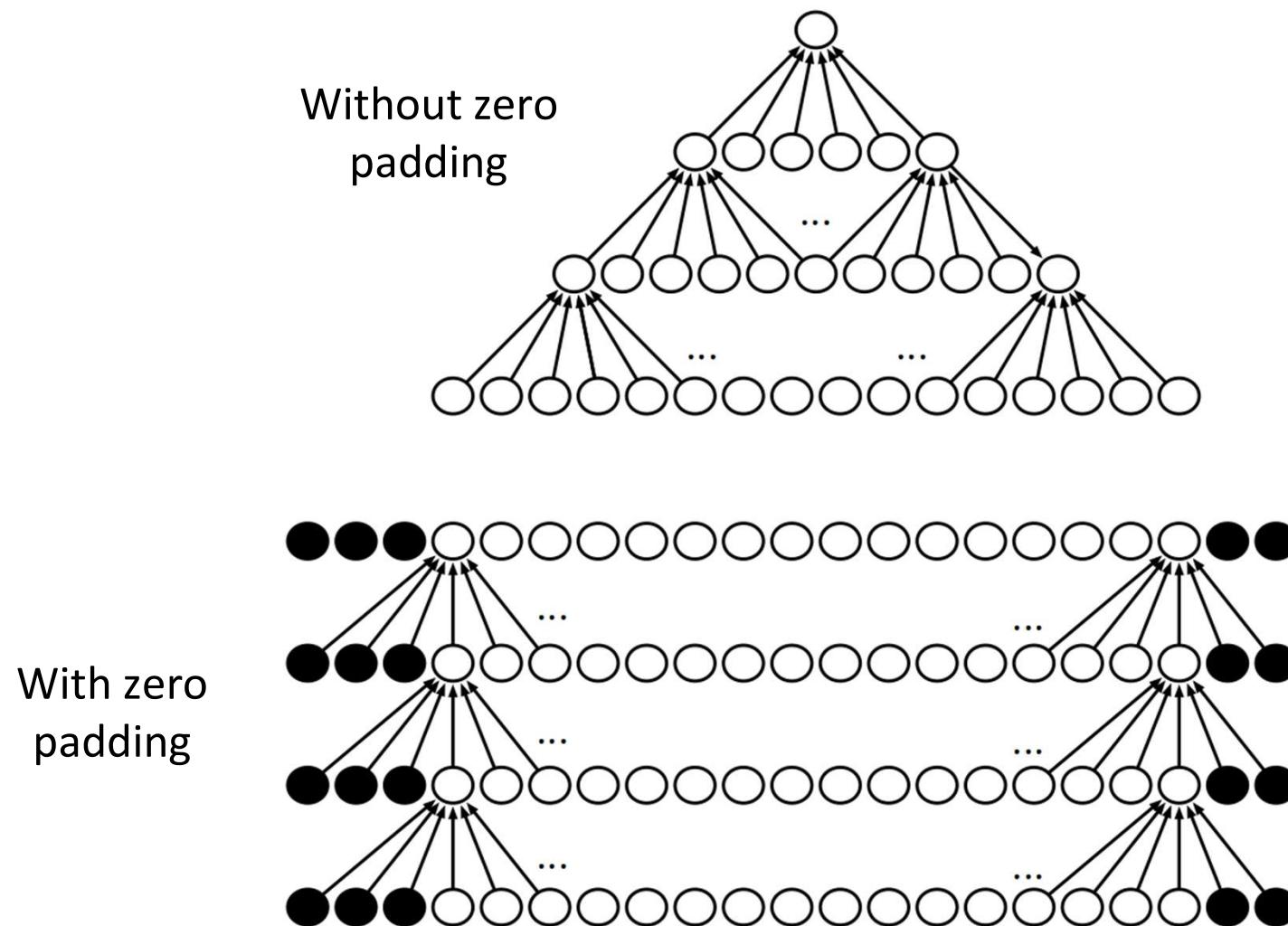
Bias = 1

Source: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>



Source: <http://cs231n.github.io/convolutional-networks/>

Zero padding controls size



Original

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

After 1st conv

4	3	4
2	4	3
2	3	4

After 2nd conv

18

Without padding the image shrinks!

Original

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

*

0	0	1
0	1	1
1	1	1

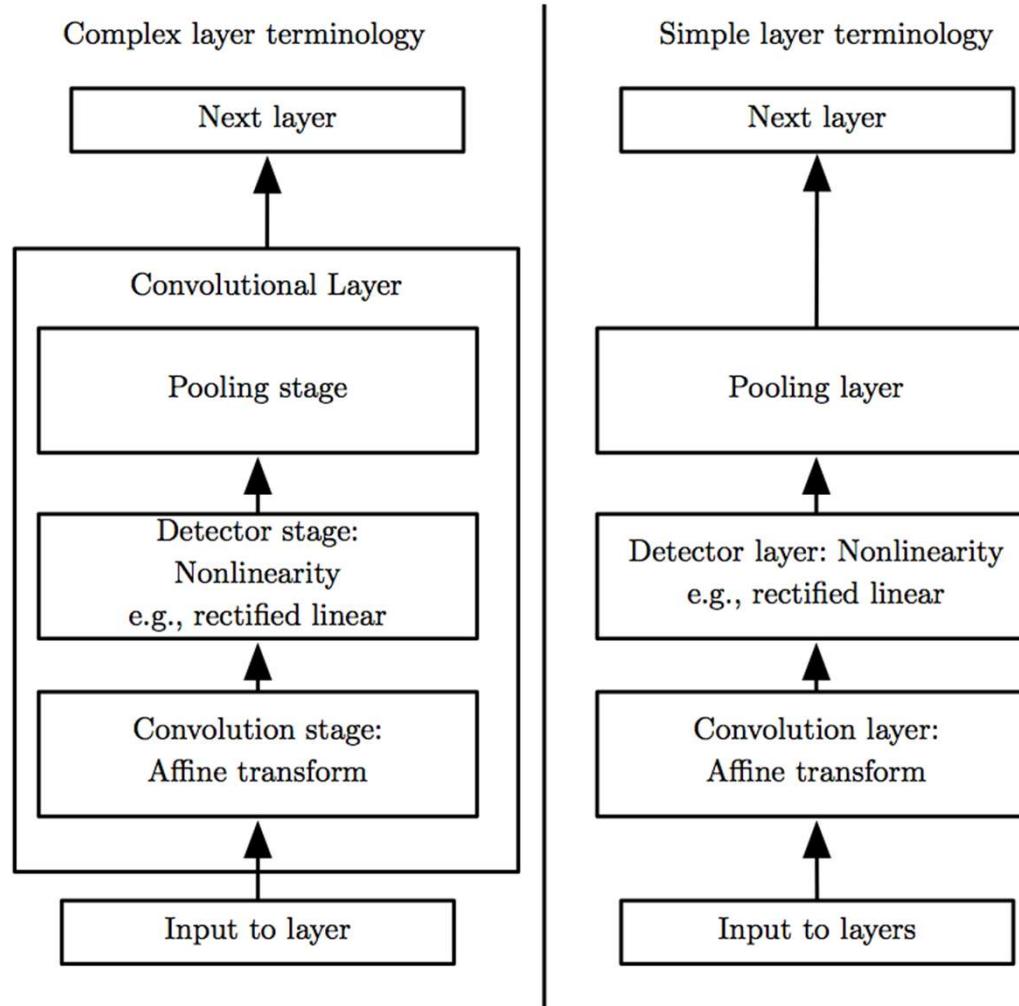
=

After 1st conv

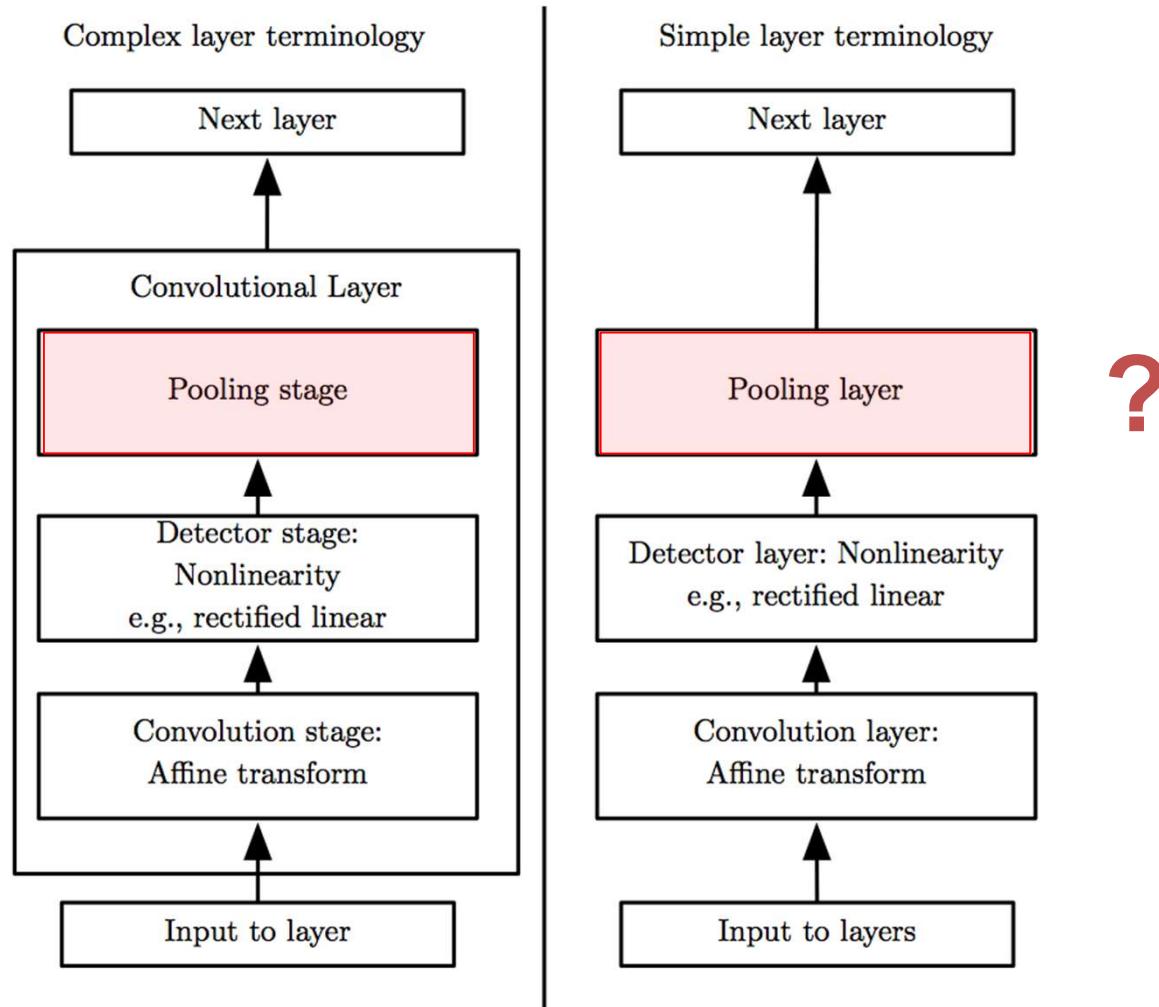
1	1	2	0	0
0	1	1	1	0
0	0	1	2	1
1	0	2	1	0
0	1	1	3	0

With padding the image doesn't shrinks!

CNN components

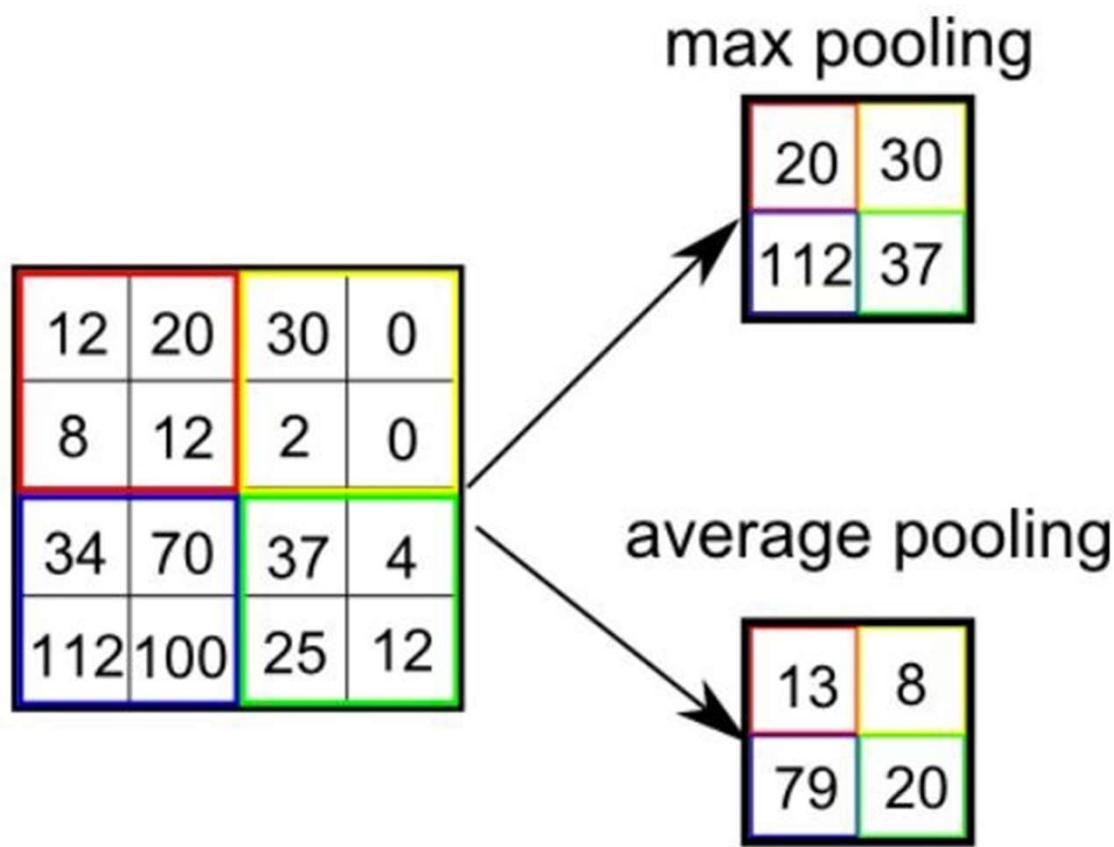


CNN components



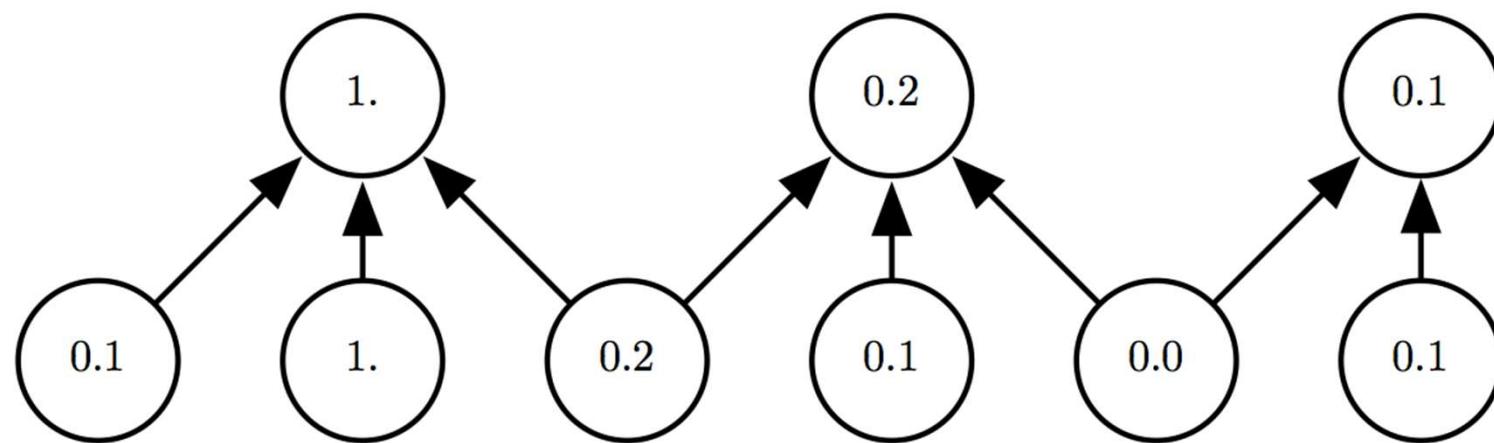
Pooling

- Aggregate multiple values into a single value
 - Reduce size of layer (speed up computation)
 - Keep most important information for next layer
 - Invariance to small transformation

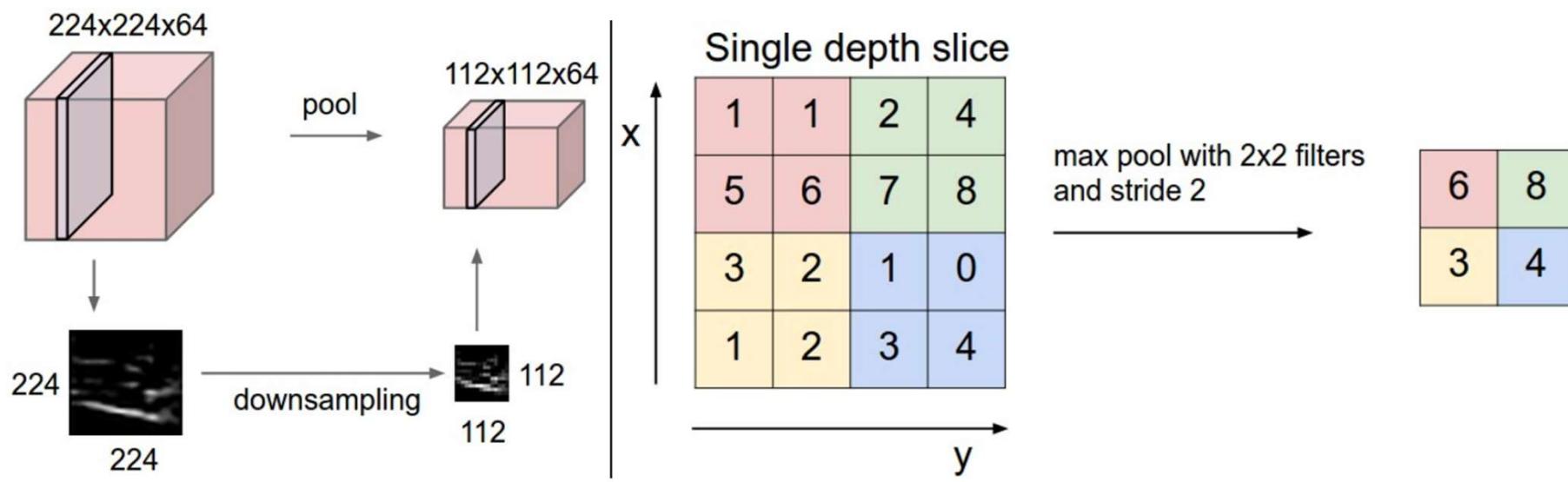


Source: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Downsampling with pooling

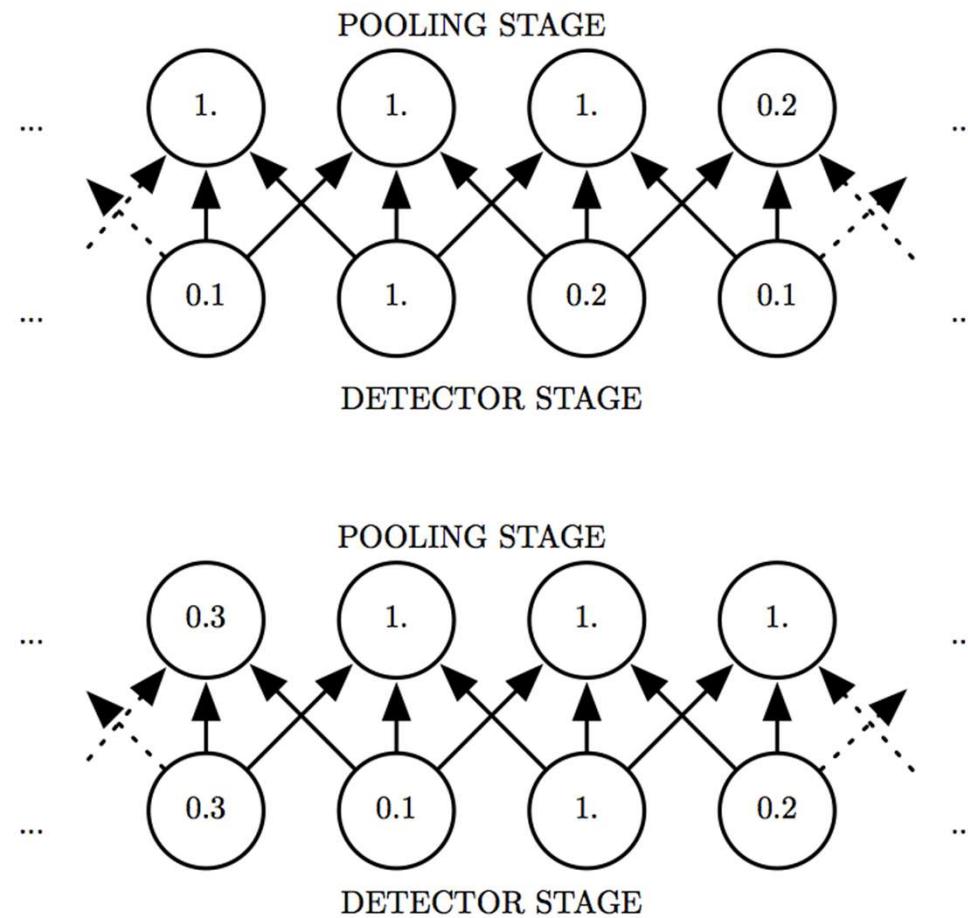


(When stride size > 1)

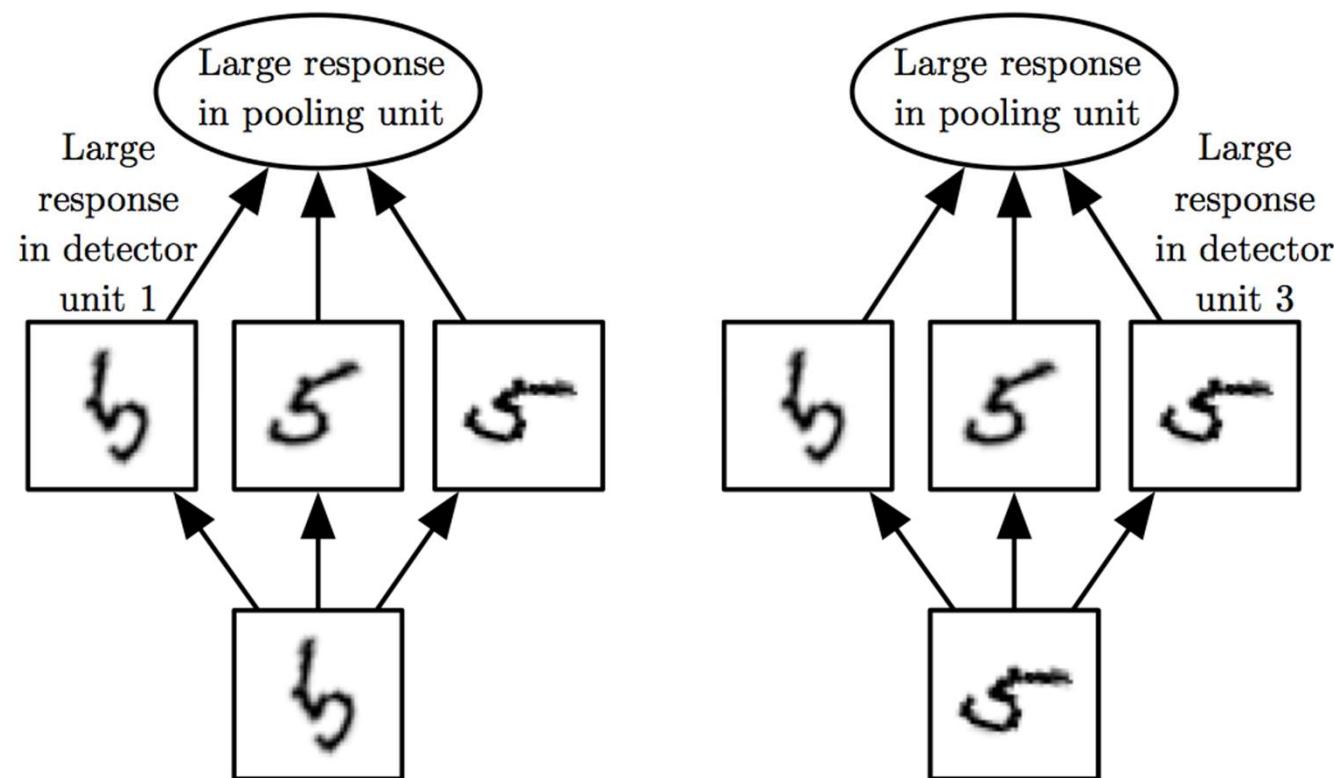


Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Left: In this example, the input volume of size $[224 \times 224 \times 64]$ is pooled with filter size 2, stride 2 into output volume of size $[112 \times 112 \times 64]$. Notice that the volume depth is preserved. Right: The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2×2 square).

Invariance through pooling



Cross-channel pooling



Do we really need pooling?

- Increasing the stride size of the kernel also reduces the size of the image...so why not doing that instead of pooling? Yes you can

Table 2: Model description of the three networks derived from base model C used for evaluating the importance of pooling in case of classification on CIFAR-10 and CIFAR-100. The derived models for base models A and B are built analogously. The higher layers are the same as in Table 1 .

Model		
Strided-CNN-C	ConvPool-CNN-C	All-CNN-C
Input 32×32 RGB image		
3×3 conv. 96 ReLU	3×3 conv. 96 ReLU	3×3 conv. 96 ReLU
3×3 conv. 96 ReLU with stride $r = 2$	3×3 conv. 96 ReLU	3×3 conv. 96 ReLU
	3×3 conv. 96 ReLU	
	3×3 max-pooling stride 2	3×3 conv. 96 ReLU with stride $r = 2$
3×3 conv. 192 ReLU	3×3 conv. 192 ReLU	3×3 conv. 192 ReLU
3×3 conv. 192 ReLU with stride $r = 2$	3×3 conv. 192 ReLU	3×3 conv. 192 ReLU
	3×3 conv. 192 ReLU	3×3 conv. 192 ReLU with stride $r = 2$
	3×3 max-pooling stride 2	

:

See: <https://arxiv.org/pdf/1412.6806.pdf> and

<https://towardsdatascience.com/iclr-2015-striving-for-simplicity-the-all-convolutional-net-with-interactive-code-manual-b4976e206760>

Summary

- The concept of convolutional NN
 - Why it is helpful in image recognition?
- What is convolution
- Convolutional layer and its basic components
- The special characteristics of convolutional layer
 - Sparse connectivity
 - Parameter sharing
 - Pooling: different type of Invariance

Next Week:

- Variants of CNN and the introduction of RNN