Deep Learning for NLP

Week 12: CNNs

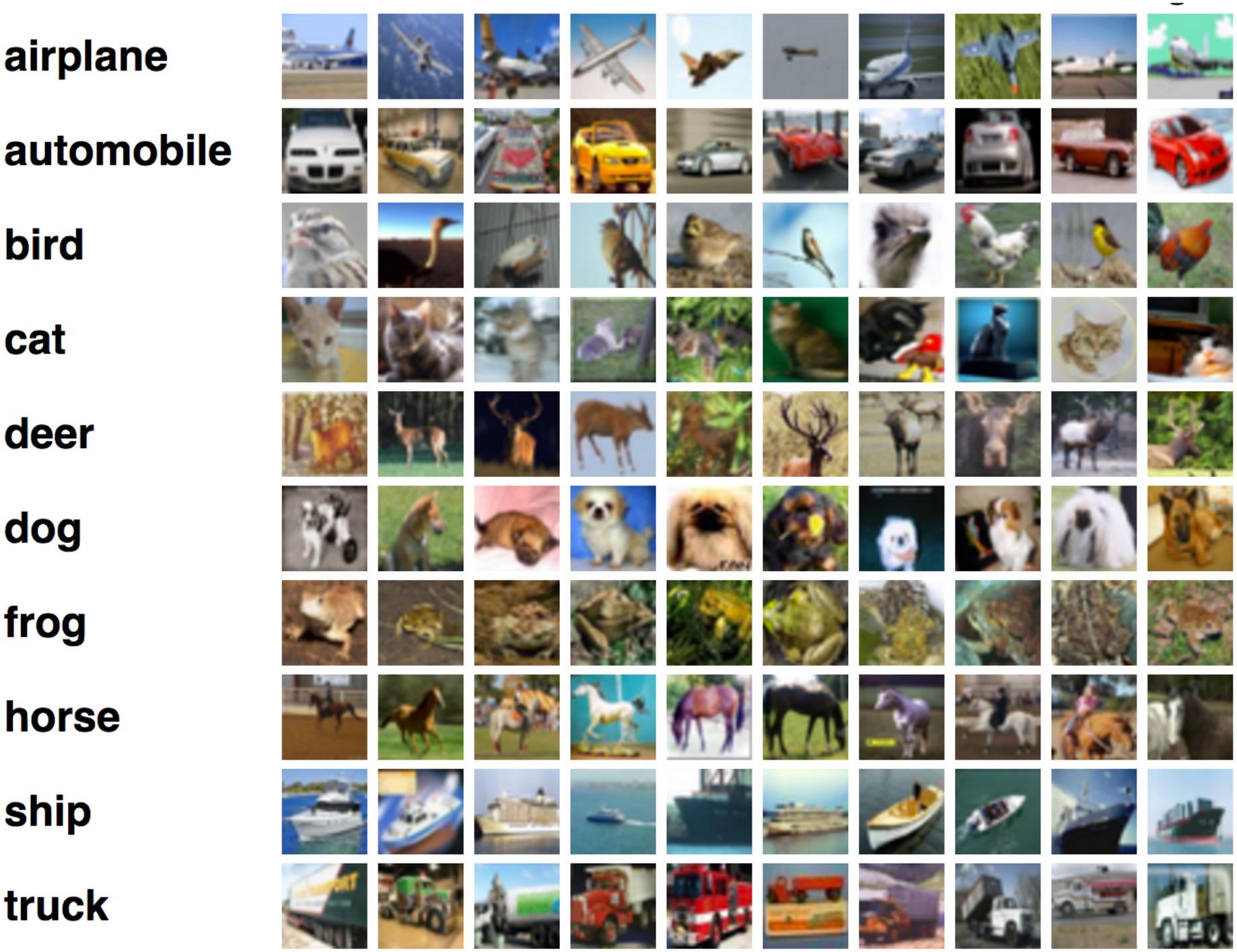
Sharid Loáiciga — July 7th, 2020

Slides by Miikka Silfverberg & Hande Celikkanat

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

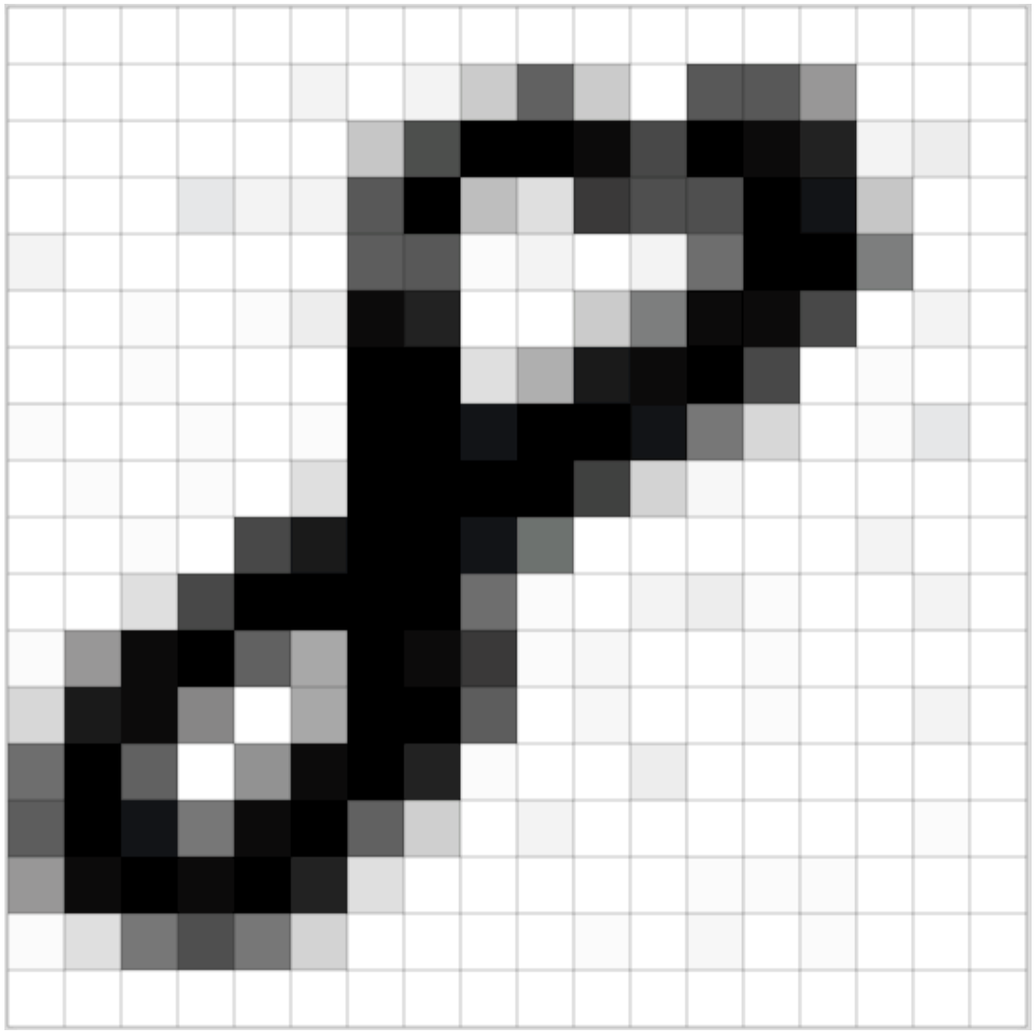
Convolutional Neural Networks

Image Classification



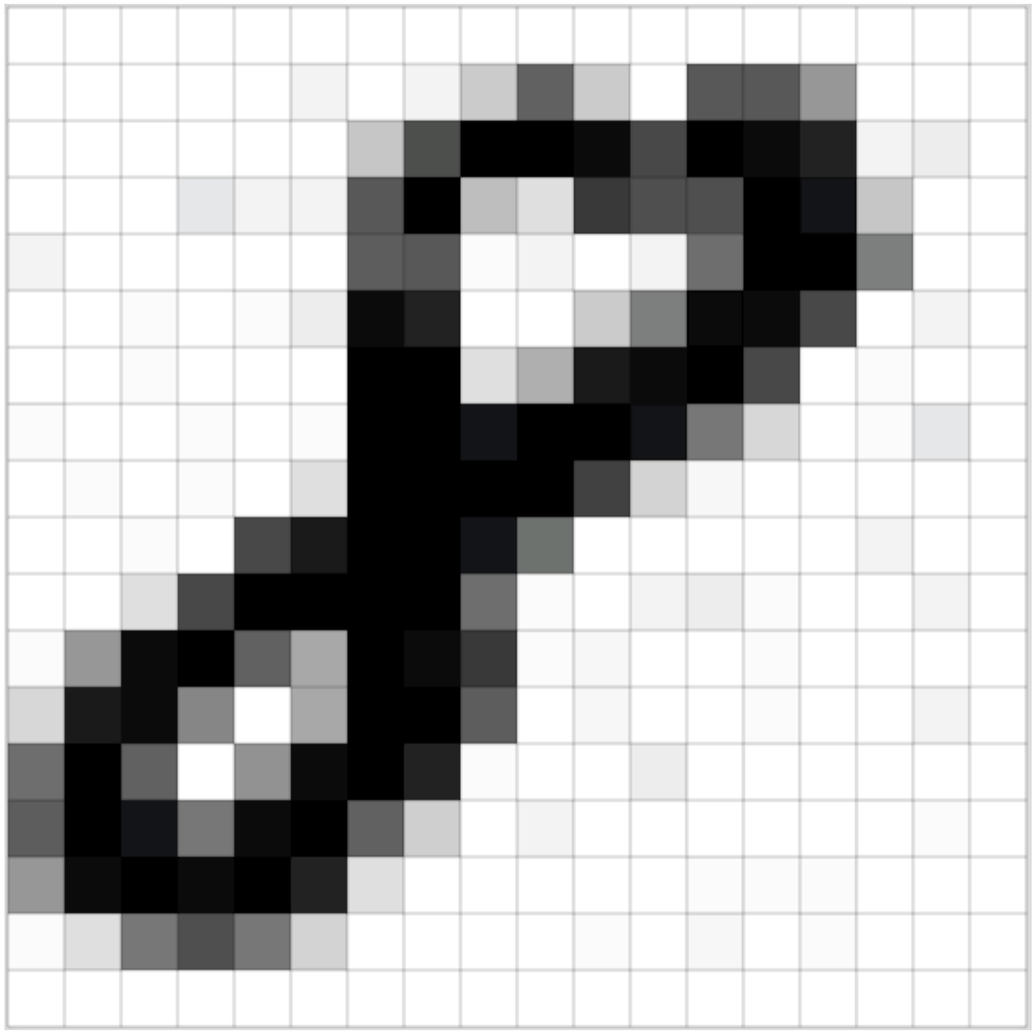
CIFAR-10 dataset

Representing Images



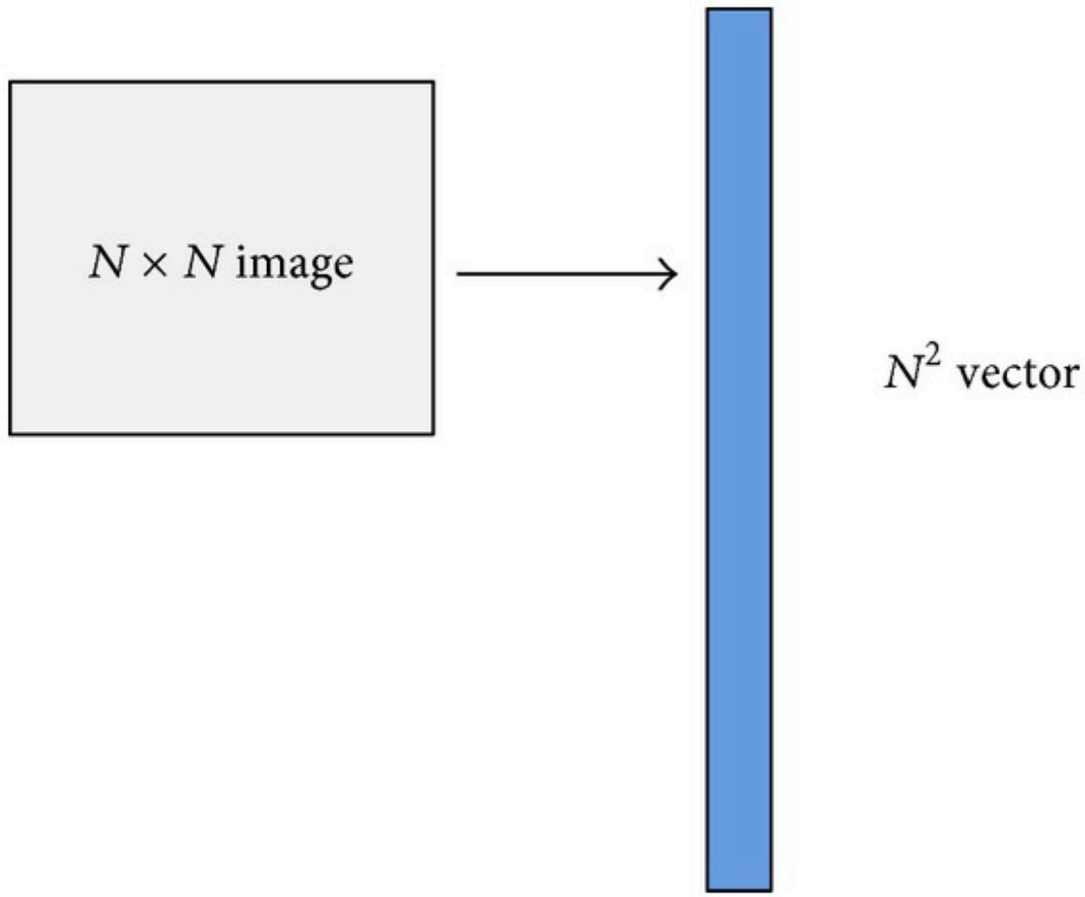
The easiest way to represent images as matrices is to interpret each pixel as a real number value

Representing Images



The easiest way to represent images as matrices is to interpret each pixel as a real number value

Representing Images



To get a vector, just concatenate rows or columns

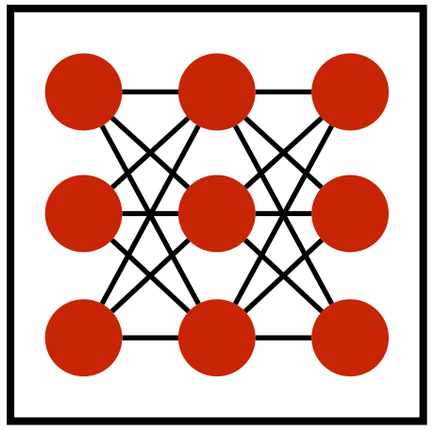
Representing Images

We can split an image into a red, green and blue channel which can be transformed into vectors and concatenated.

Image Classification using Feed-Forward Networks

scale

N^2 vector



cat 0.85

dog 0.12

bird 0.05

car 0.002

house 0.002

…

In this architecture, we feed the image vector directly into a feed-forward NN trained for classification.

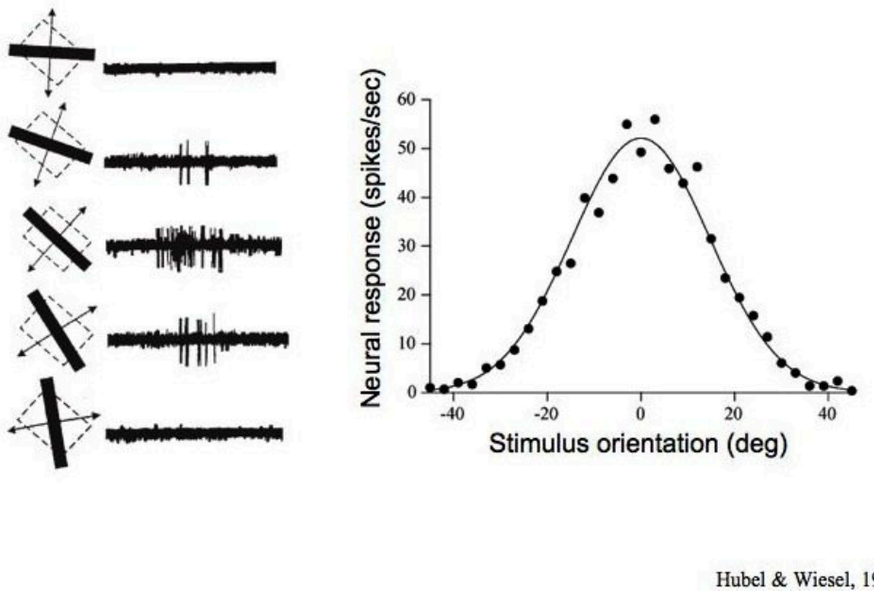
Problems

Problems

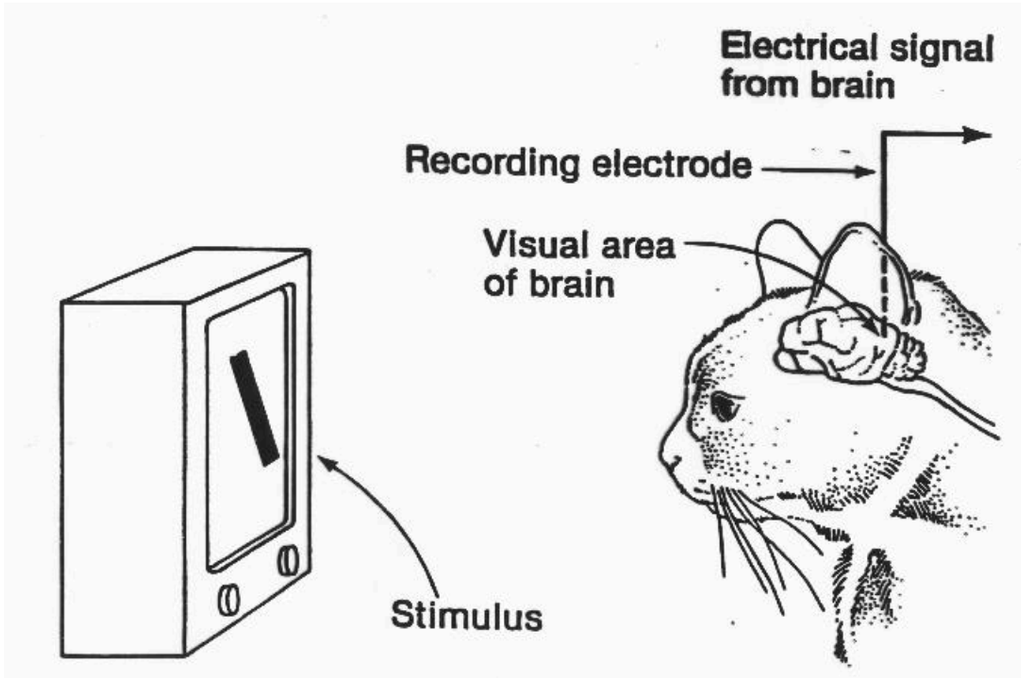
Learning to recognize a circle in the left upper corner doesn't (directly) help in recognizing a circle in the right lower corner.

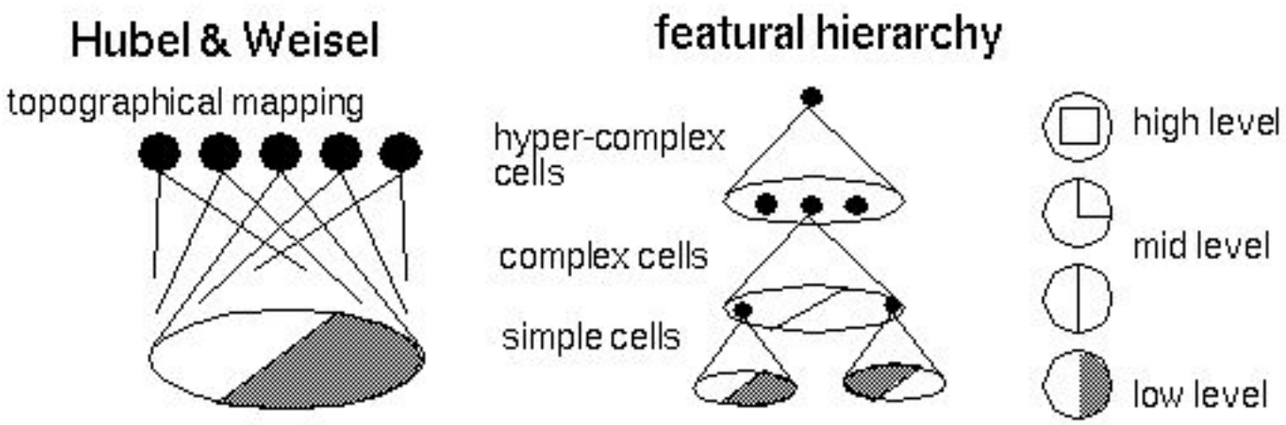
The main problem in the FFNN architecture is that there is no parameter sharing across the spatial dimension.

Biological Inspiration



Hubel & Wiesel, 1968

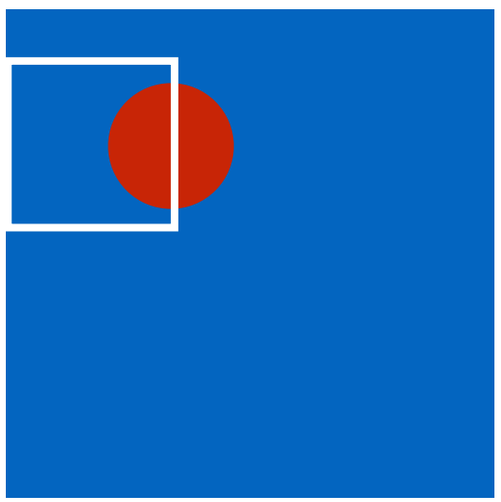


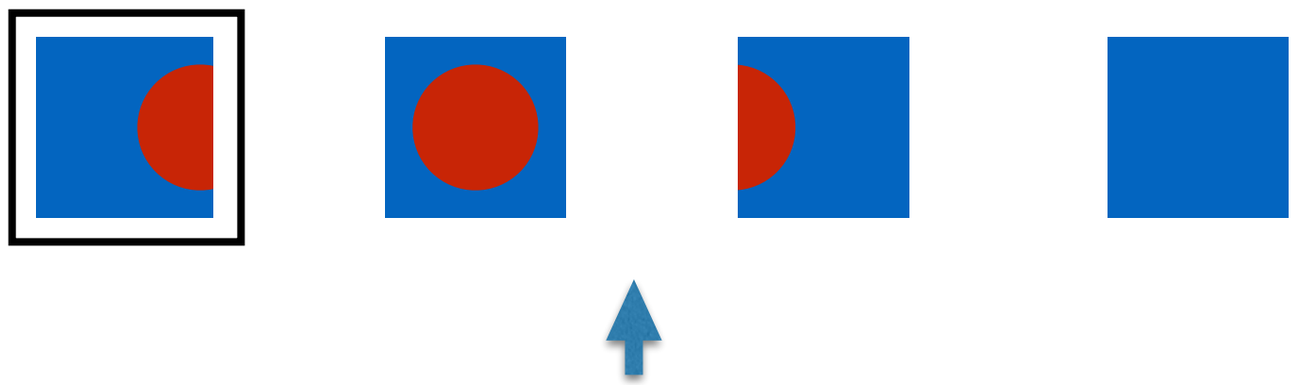


Hubel & Wiesel 1958

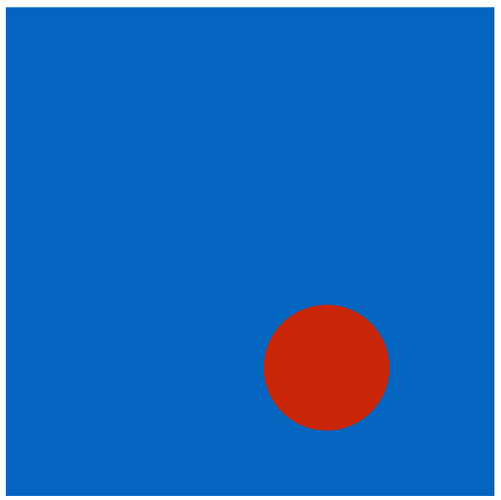
Biological Inspiration

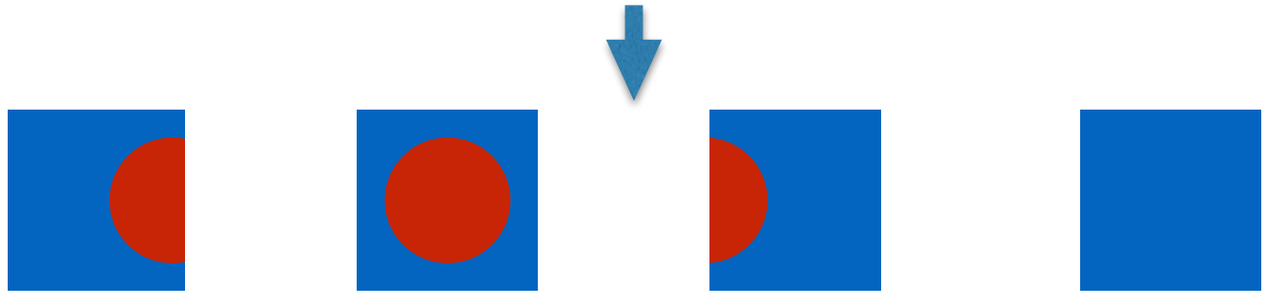
Convolutional filters: Intuition





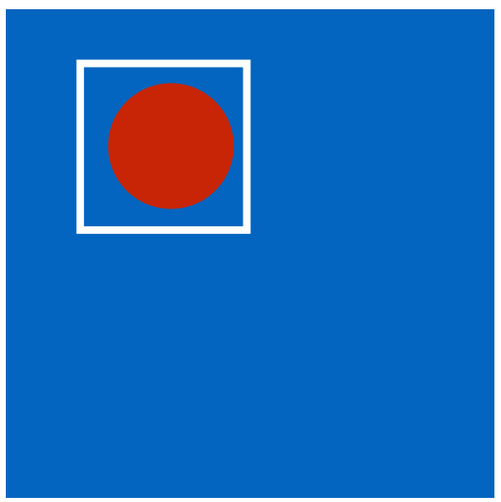
The images are composed of pieces like this

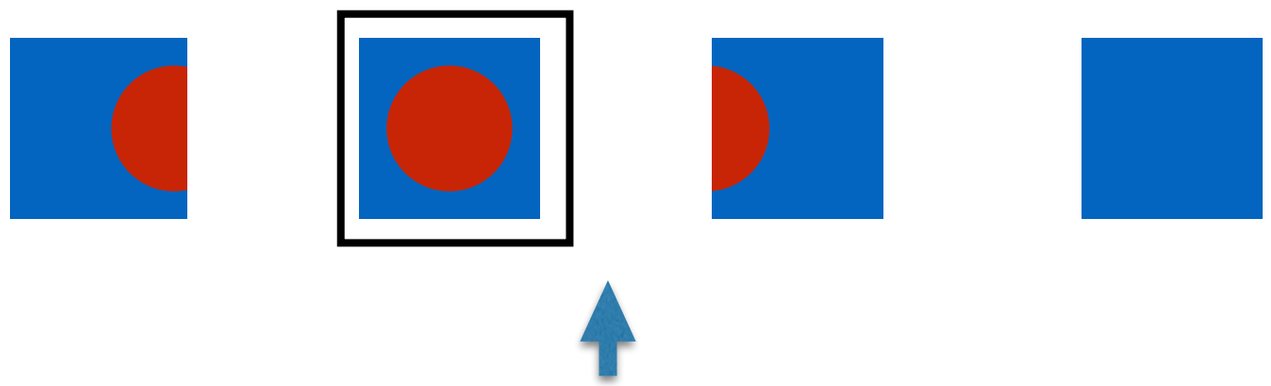




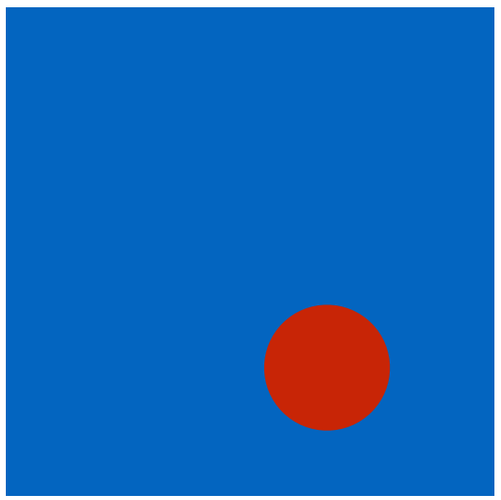
Using convolutional filters, we can look at a small section of the image at a time.

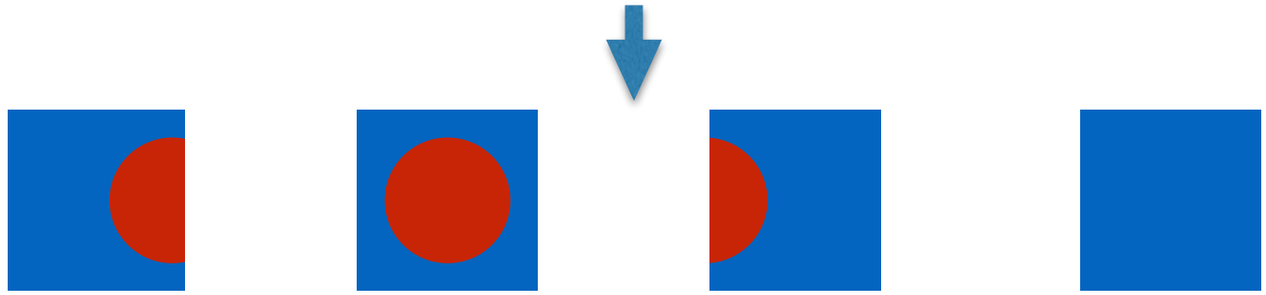
Convolutional filters: Intuition





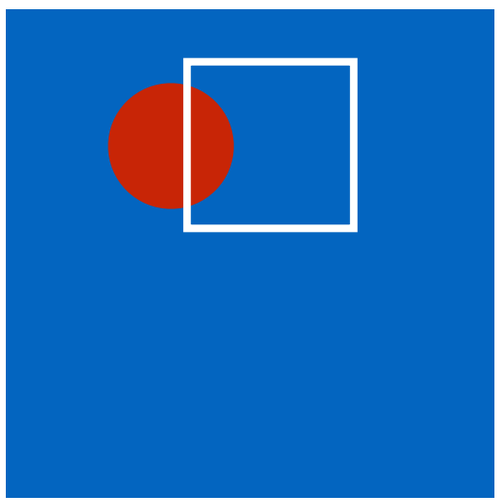
The images are composed of pieces like this

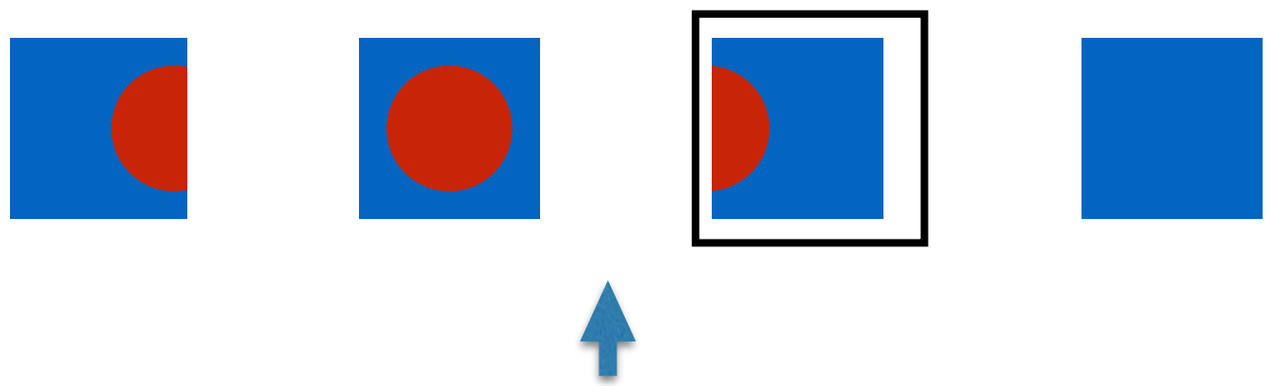




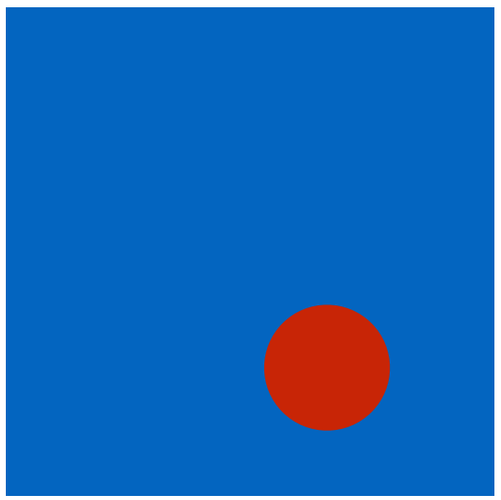
Using convolutional filters, we can look at a small section of the image at a time.

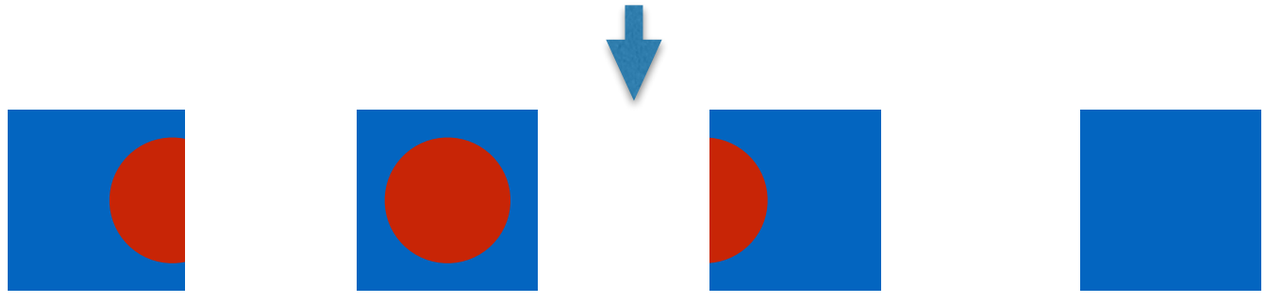
Convolutional filters: Intuition





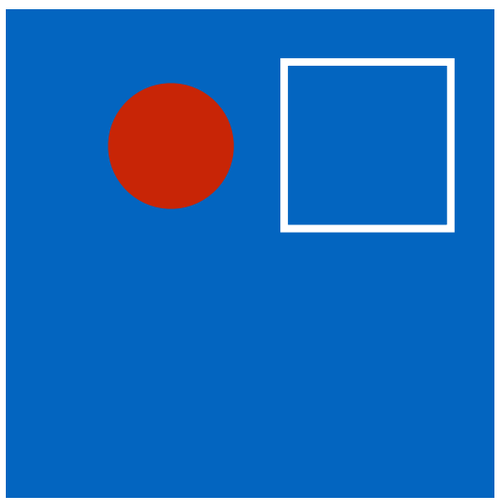
The images are composed of pieces like this

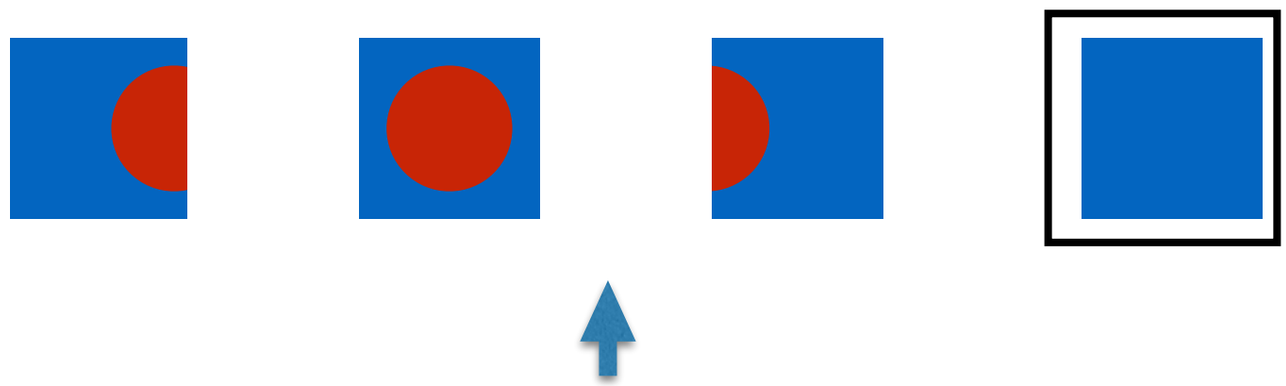




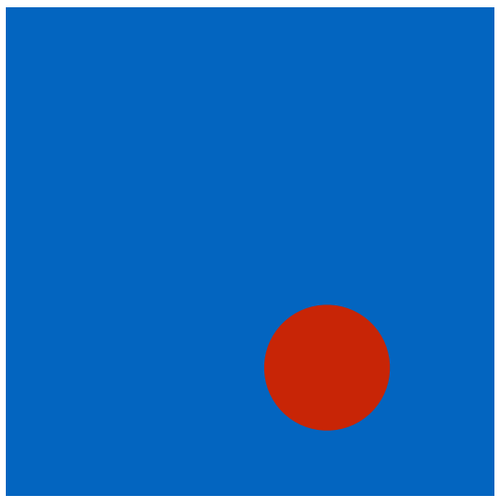
Using convolutional filters, we can look at a small section of the image at a time.

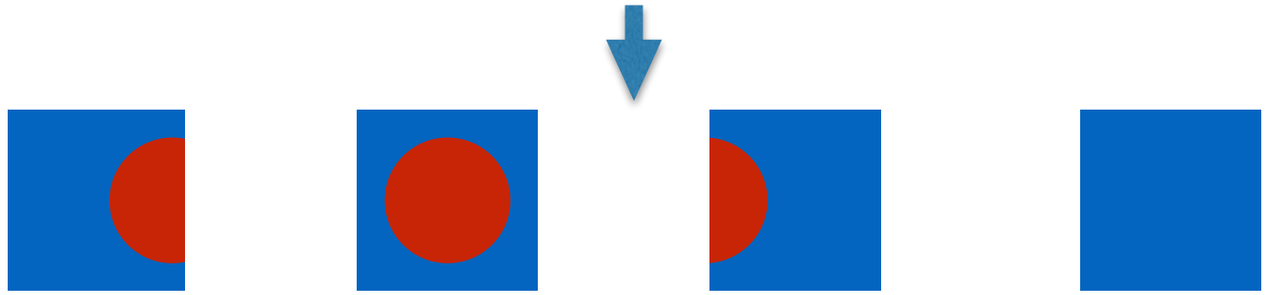
Convolutional filters: Intuition





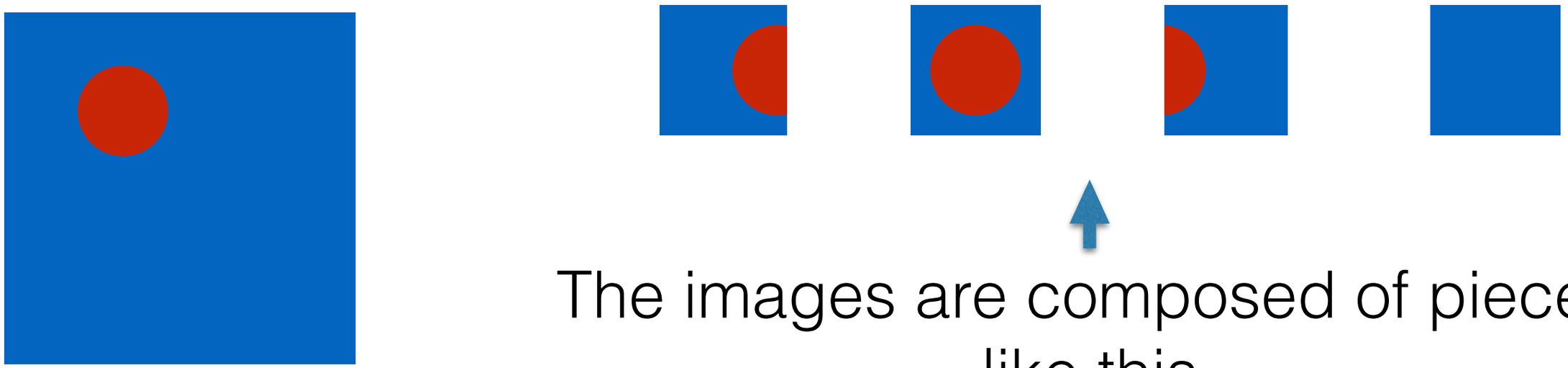
The images are composed of pieces like this



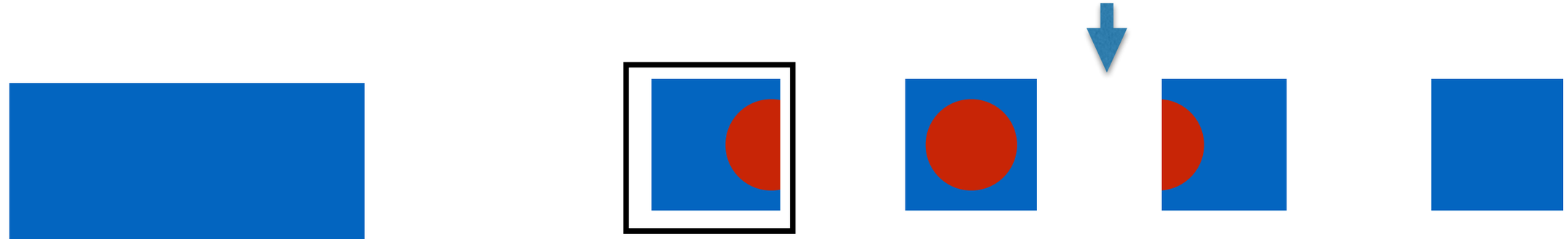


Using convolutional filters, we can look at a small section of the image at a time.

Convolutional filters: Intuition

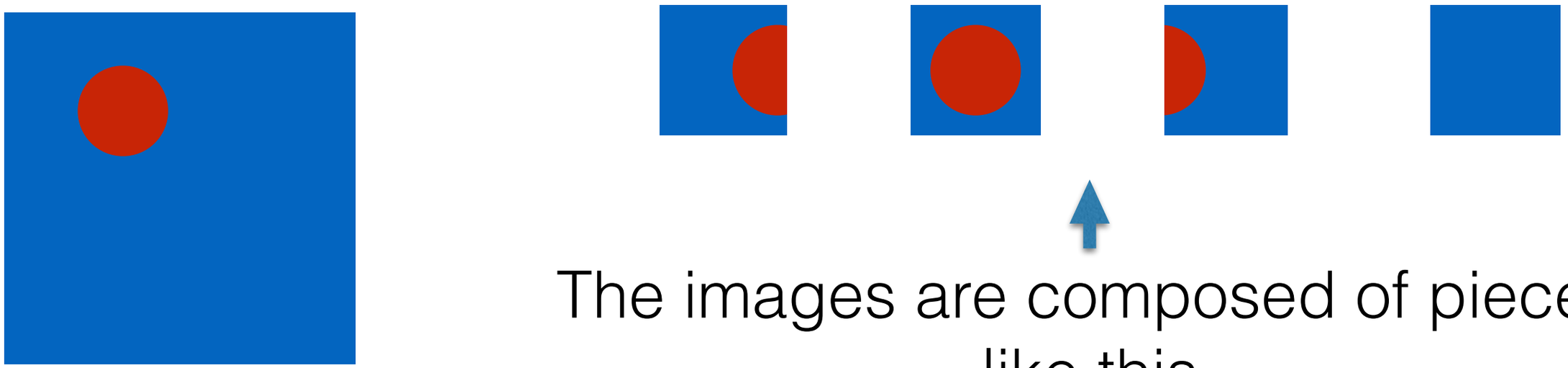


The images are composed of pieces like this

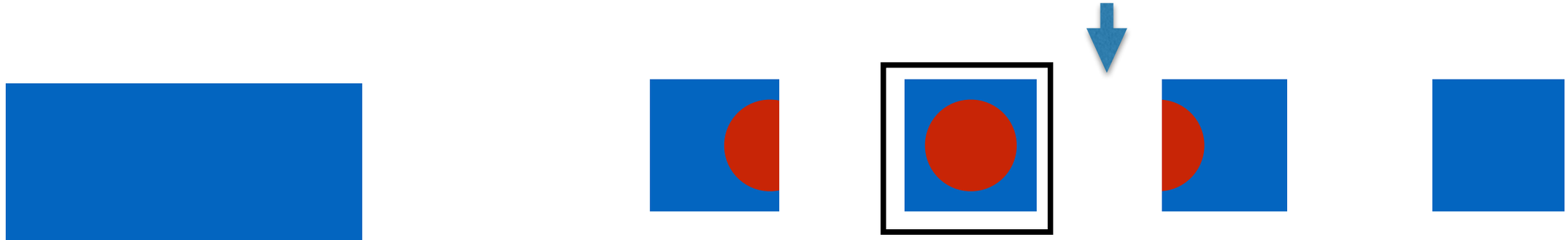


Using convolutional filters, we can look at a small section of the image at a time.

Convolutional filters: Intuition

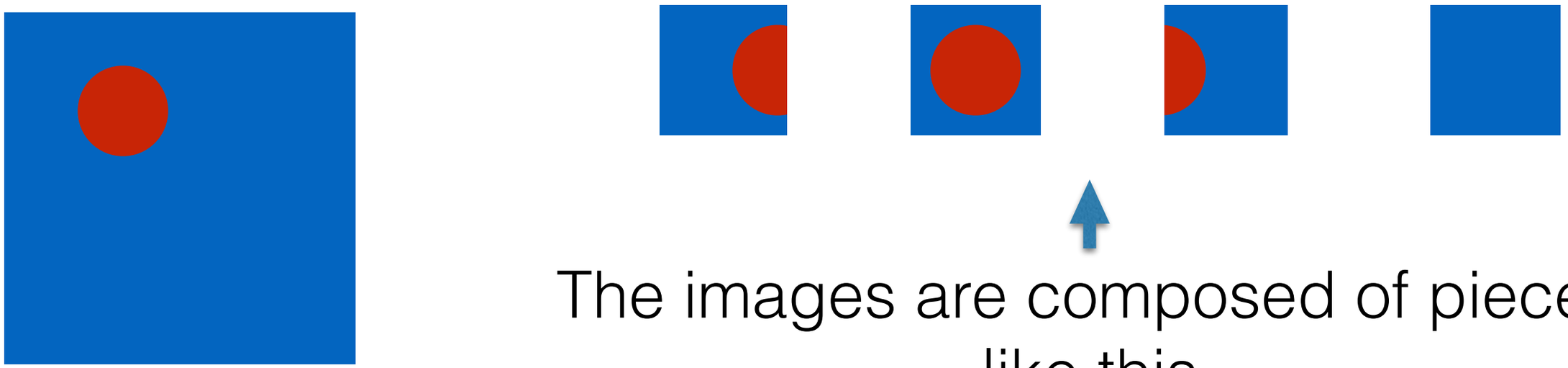


The images are composed of pieces like this

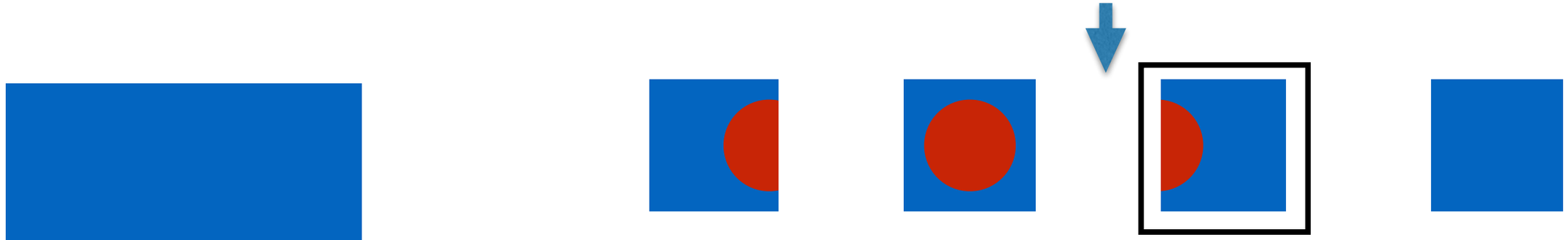


Using convolutional filters, we can look at a small section of the image at a time.

Convolutional filters: Intuition



The images are composed of pieces like this



Using convolutional filters, we can look at a small section of the image at a time.

Convolutional Filter

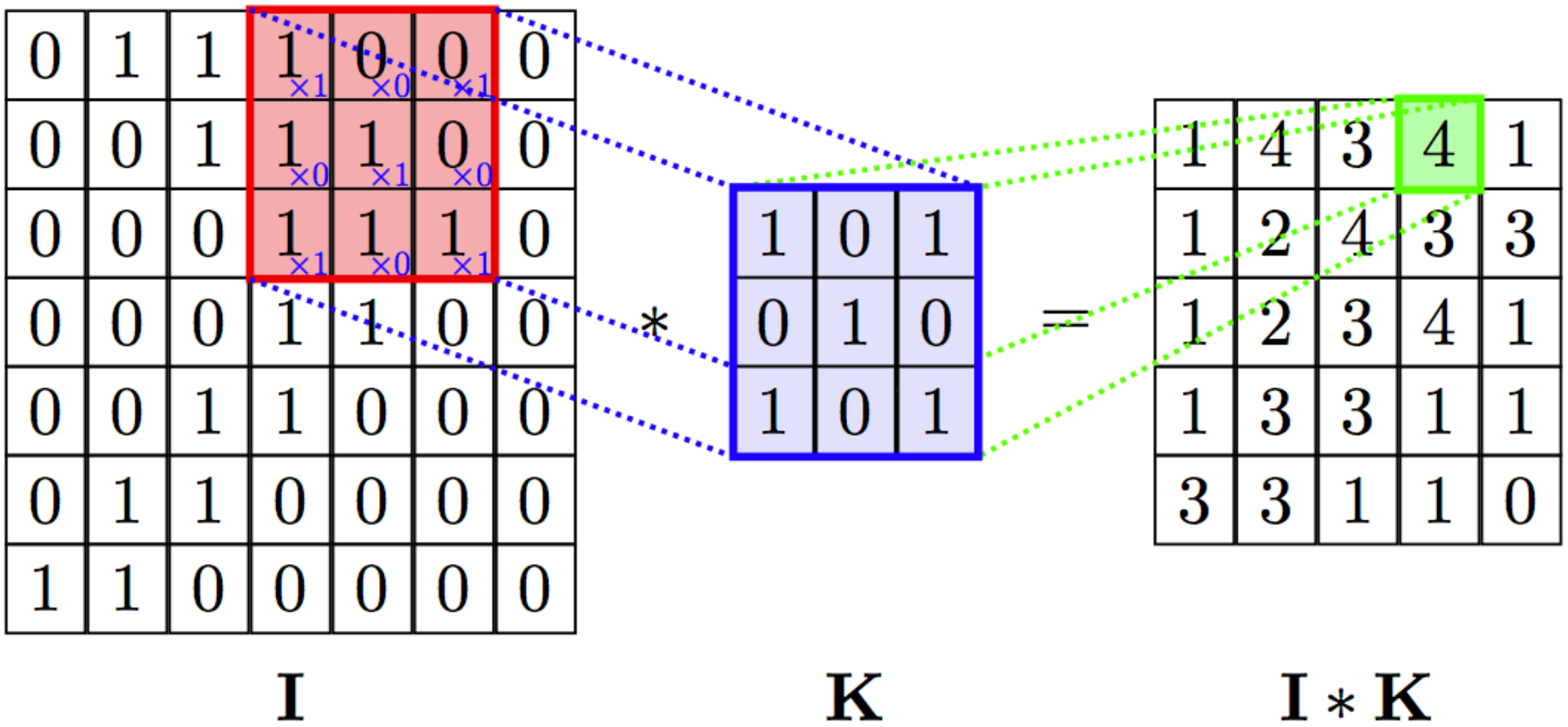


Image K is convolved with filter I. Each element in I * K is a dot product of I and K interpreted as vectors.

Convolutional Filter

Image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Filter:

```
-1  1 -1
-1  1 -1
-1  1 -1
```

Convolutional Filter

Image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Filter:

```
-1  1 -1
-1  1 -1
-1  1 -1
```

```
-1  1 -1       0  0  0
-1  1 -1   ·   0  0  0   =   0
-1  1 -1       0  0  0
```

Convolutional Filter

Image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Filter:

```
-1  1 -1
-1  1 -1
-1  1 -1
```

```
-1  1 -1       0  0  0
-1  1 -1   ·   0  0  0   =   0
-1  1 -1       0  0  0
```

Convolutional Filter

Image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Filter:

```
-1  1 -1
-1  1 -1
-1  1 -1
```

```
-1  1 -1       0  0  0
-1  1 -1   ·   0  0  0   =   0
-1  1 -1       0  0  0

-1  1 -1       0  1  0
-1  1 -1   ·   0  1  0   =   3
-1  1 -1       0  1  0
```

Convolutional Filter

Image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Filter:

```
-1  1 -1
-1  1 -1
-1  1 -1
```

```
-1  1 -1       0  0  0
-1  1 -1   ·   0  0  0   =   0
-1  1 -1       0  0  0

-1  1 -1       0  1  0
-1  1 -1   ·   0  1  0   =   3
-1  1 -1       0  1  0
```

Result:

Filter fires strongly here

```
0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0
0  0 -1  1 -1  0  0  0  0
0  0 -2  2 -2  0  0  0  0
0  0 -3  3 -3  0  0  0  0
0  0 -2  2 -2  0  0  0  0
0  0 -1  1 -1 -1  0 -1  0
0  0  0  0  0 -1  0 -1  0
0  0  0  0  0 -1  0 -1  0
```

Convolutional Filter

Image:

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

Filter:

```
-1  1 -1
-1  1 -1
-1  1 -1
```

```
-1  1 -1       0  0  0
-1  1 -1   ·   0  0  0   =   0
-1  1 -1       0  0  0

-1  1 -1       0  1  0
-1  1 -1   ·   0  1  0   =   3
-1  1 -1       0  1  0
```

Result:

```
0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0
0  0 -1  1 -1  0  0  0  0
0  0 -2  2 -2  0  0  0  0
0  0 -3  3 -3  0  0  0  0
0  0 -2  2 -2  0  0  0  0
0  0 -1  1 -1 -1  0 -1  0
0  0  0  0  0 -1  0 -1  0
0  0  0  0  0 -1  0 -1  0
```

Filter fires strongly here

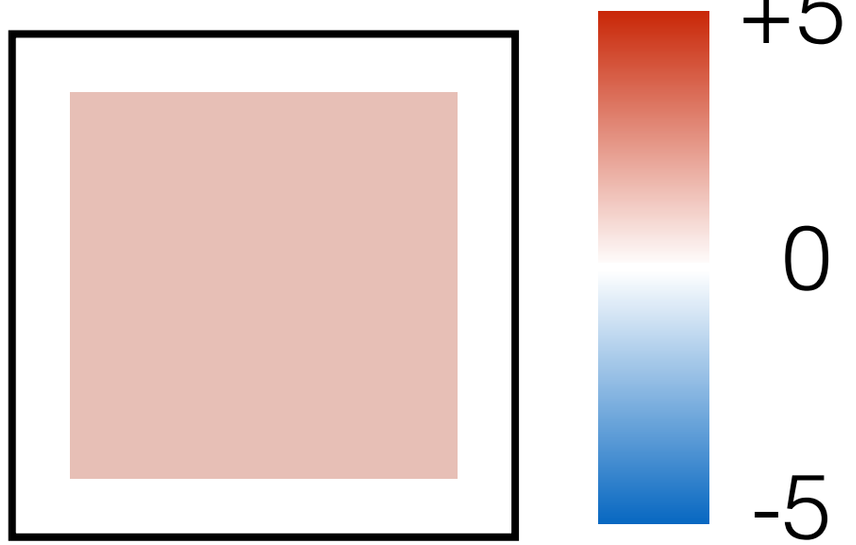
After ReLU:

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

Example: Detecting Edges

Image:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



Applying to monochr. region:

$$\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix} \cdot \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} = 0$$

$$\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix} \cdot \begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} = 0$$

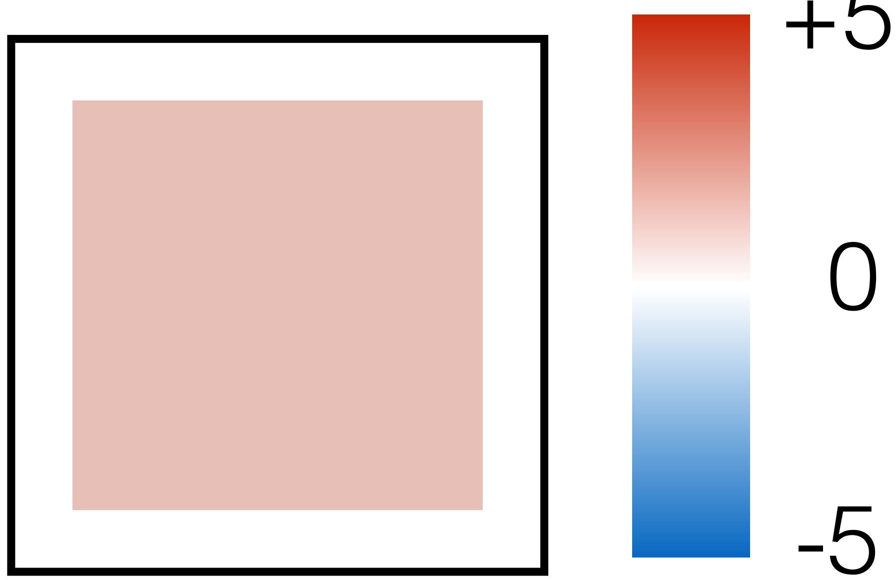
Filter:

```
-1 -1 -1
-1  8 -1
-1 -1 -1
```

Example: Detecting Edges

Image:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
```



Applying to monochr. region:

```
-1 -1 -1     0 0 0
-1  8 -1  ·  0 0 0  =  0
-1 -1 -1     0 0 0

-1 -1 -1     1 1 1
-1  8 -1  ·  1 1 1  =  0
-1 -1 -1     1 1 1
```

Filter:

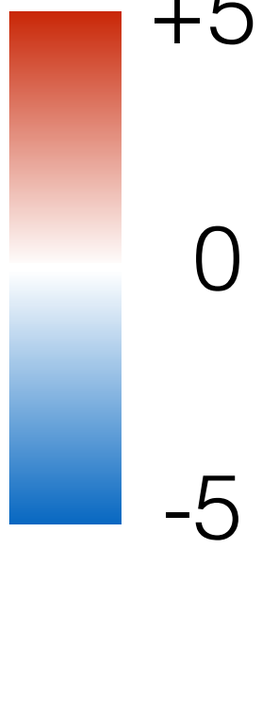
```
-1 -1 -1
-1  8 -1
-1 -1 -1
```

Applying near an edge:

```
-1 -1 -1     0 0 0
-1  8 -1  ·  0 1 1  =  5
-1 -1 -1     0 1 1
```

Example: Detecting Edges

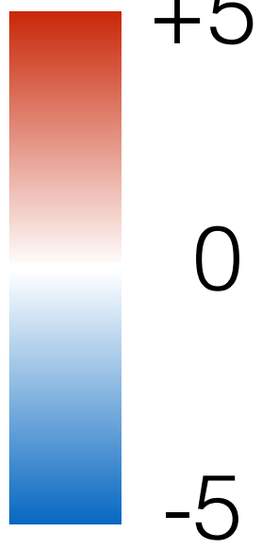
$$\text{conv2d}\left(\begin{matrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{matrix}, \begin{matrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{matrix}\right)$$



Example: Detecting Edges

$$\text{conv2d}\left(\begin{matrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{matrix}, \begin{matrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{matrix}\right) =$$

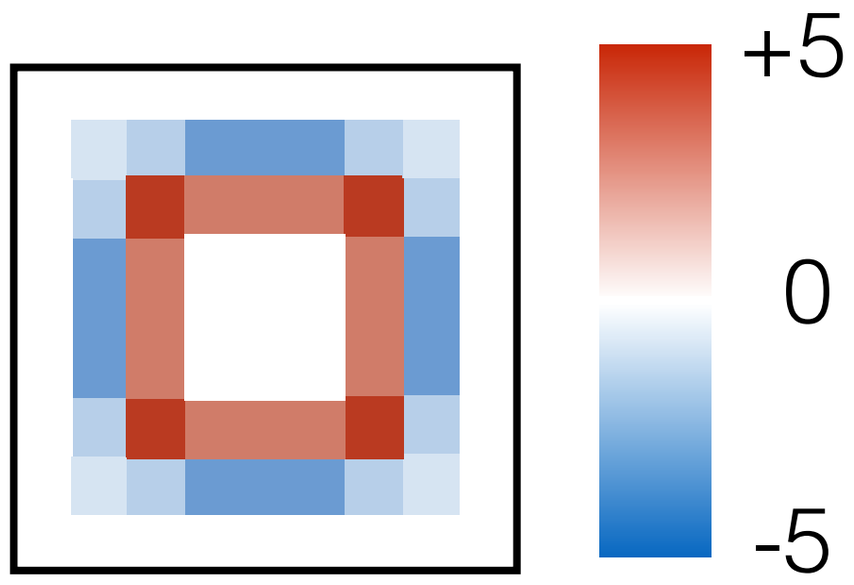
$$\begin{matrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&-1&-2&-3&-3&-3&-3&-3&-2&-1&0&0\\0&0&0&-2&5&3&3&3&3&3&5&-2&0&0\\0&0&0&-3&3&0&0&0&0&0&3&-3&0&0\\0&0&0&-3&3&0&0&0&0&0&3&-3&0&0\\0&0&0&-3&3&0&0&0&0&0&3&-3&0&0\\0&0&0&-3&3&0&0&0&0&0&3&-3&0&0\\0&0&0&-3&3&0&0&0&0&0&3&-3&0&0\\0&0&0&-2&5&3&3&3&3&3&5&-2&0&0\\0&0&0&-1&-2&-3&-3&-3&-3&-3&-2&-1&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{matrix}$$



Example: Detecting Edges

$$\text{conv2d}\left(\begin{matrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{matrix}, \begin{matrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&1&1&1&1&1&1&1&1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{matrix}\right) =$$

$$\begin{matrix}0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&-1&-2&-3&-3&-3&-3&-3&-3&-2&-1&0&0&0\\0&0&0&-2&5&3&3&3&3&3&3&5&-2&0&0&0\\0&0&0&-3&3&0&0&0&0&0&0&3&-3&0&0&0\\0&0&0&-3&3&0&0&0&0&0&0&3&-3&0&0&0\\0&0&0&-3&3&0&0&0&0&0&0&3&-3&0&0&0\\0&0&0&-3&3&0&0&0&0&0&0&3&-3&0&0&0\\0&0&0&-3&3&0&0&0&0&0&0&3&-3&0&0&0\\0&0&0&-3&3&0&0&0&0&0&0&3&-3&0&0&0\\0&0&0&-2&5&3&3&3&3&3&3&5&-2&0&0&0\\0&0&0&-1&-2&-3&-3&-3&-3&-3&-3&-2&-1&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{matrix}$$

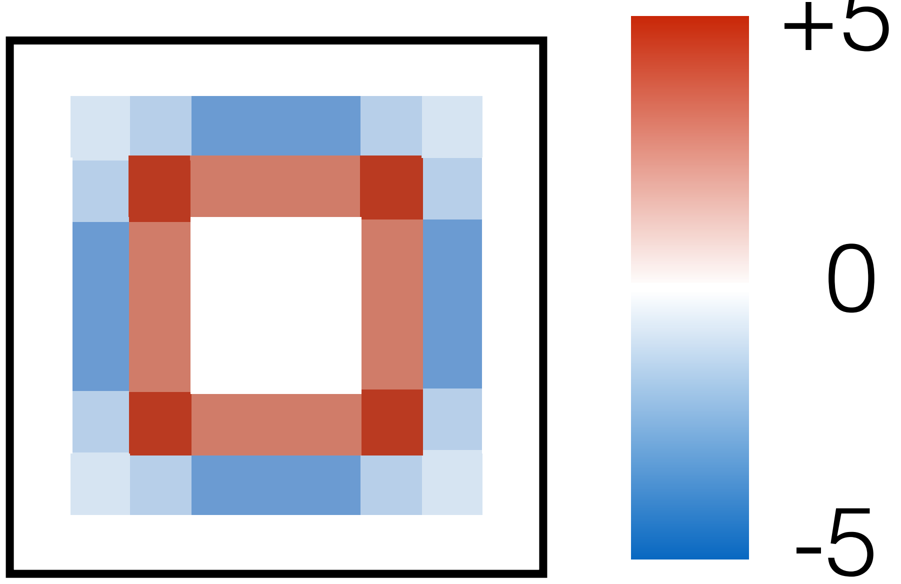


Example: Detecting Edges

conv2d($\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$,

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

)

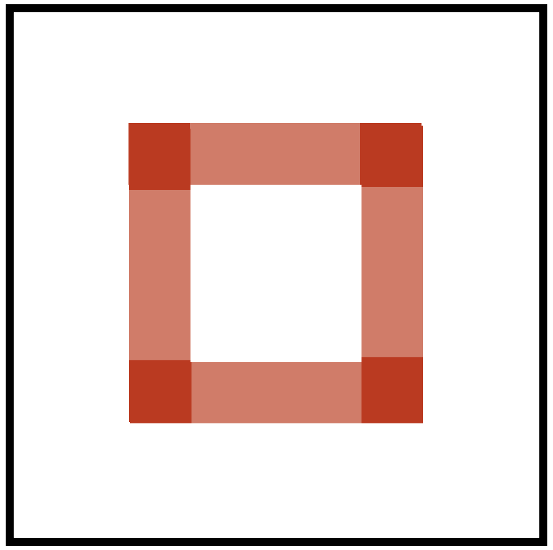


=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	−1	−2	−3	−3	−3	−3	−3	−3	−2	−1	0	0	0
0	0	0	−2	5	3	3	3	3	3	3	5	−2	0	0	0
0	0	0	−3	3	0	0	0	0	0	0	3	−3	0	0	0
0	0	0	−3	3	0	0	0	0	0	0	3	−3	0	0	0
0	0	0	−3	3	0	0	0	0	0	0	3	−3	0	0	0
0	0	0	−3	3	0	0	0	0	0	0	3	−3	0	0	0
0	0	0	−3	3	0	0	0	0	0	0	3	−3	0	0	0
0	0	0	−3	3	0	0	0	0	0	0	3	−3	0	0	0
0	0	0	−2	5	3	3	3	3	3	3	5	−2	0	0	0
0	0	0	−1	−2	−3	−3	−3	−3	−3	−3	−2	−1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Add a ReLU:

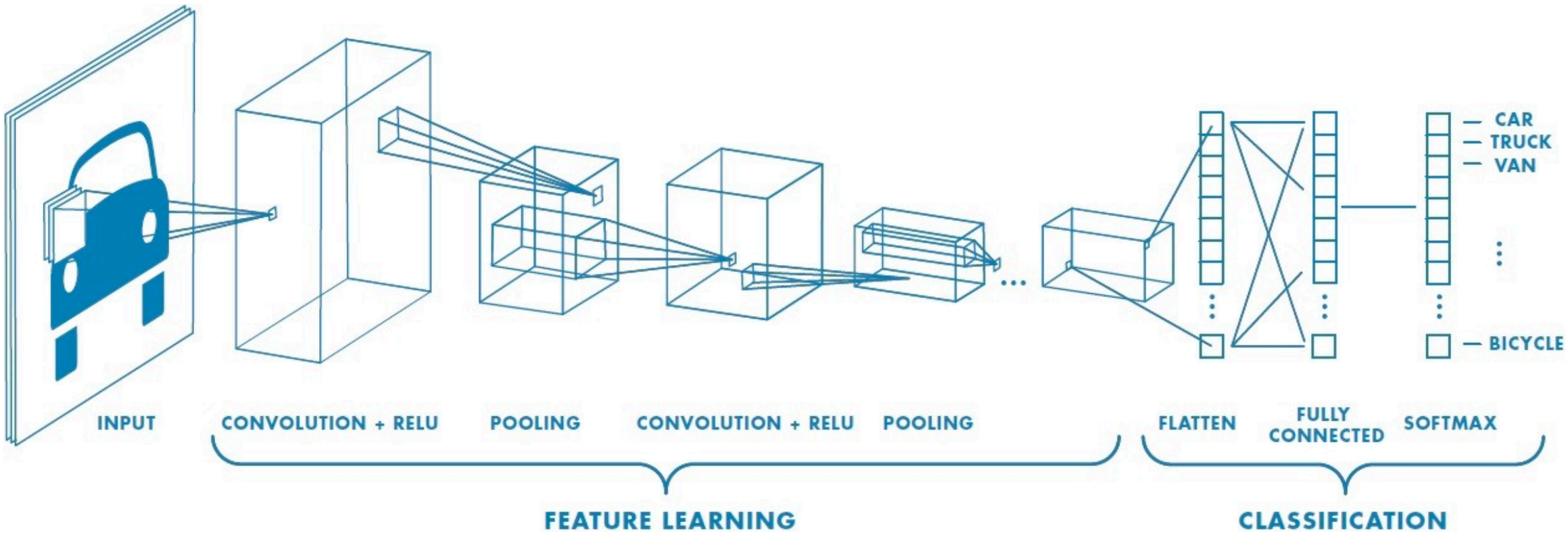
ReLU(conv2d(F, I)) =



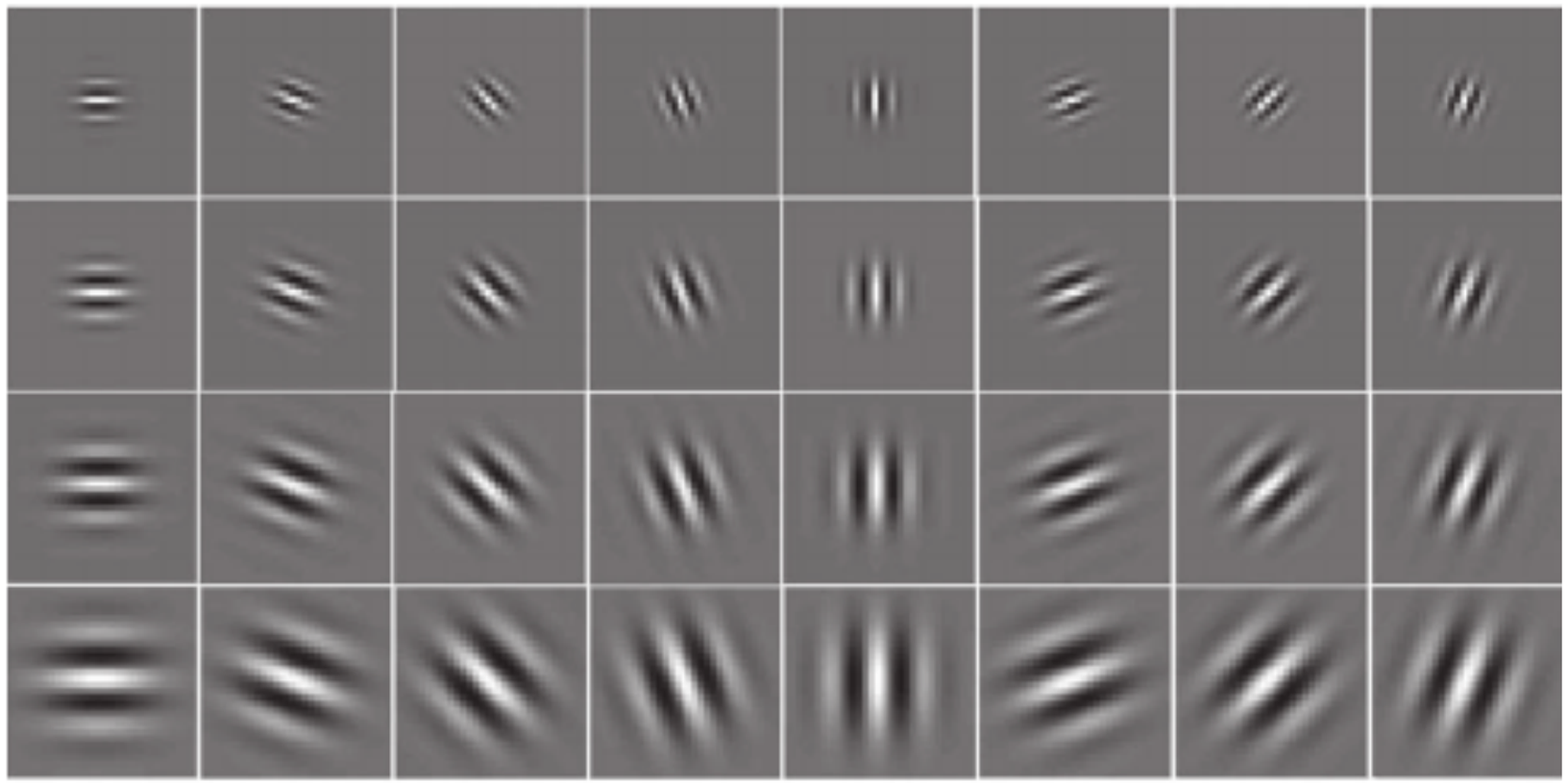
Example: Detecting Edges

Monotonous regions vanish. Only edges between regions are preserved.

General CNN architecture

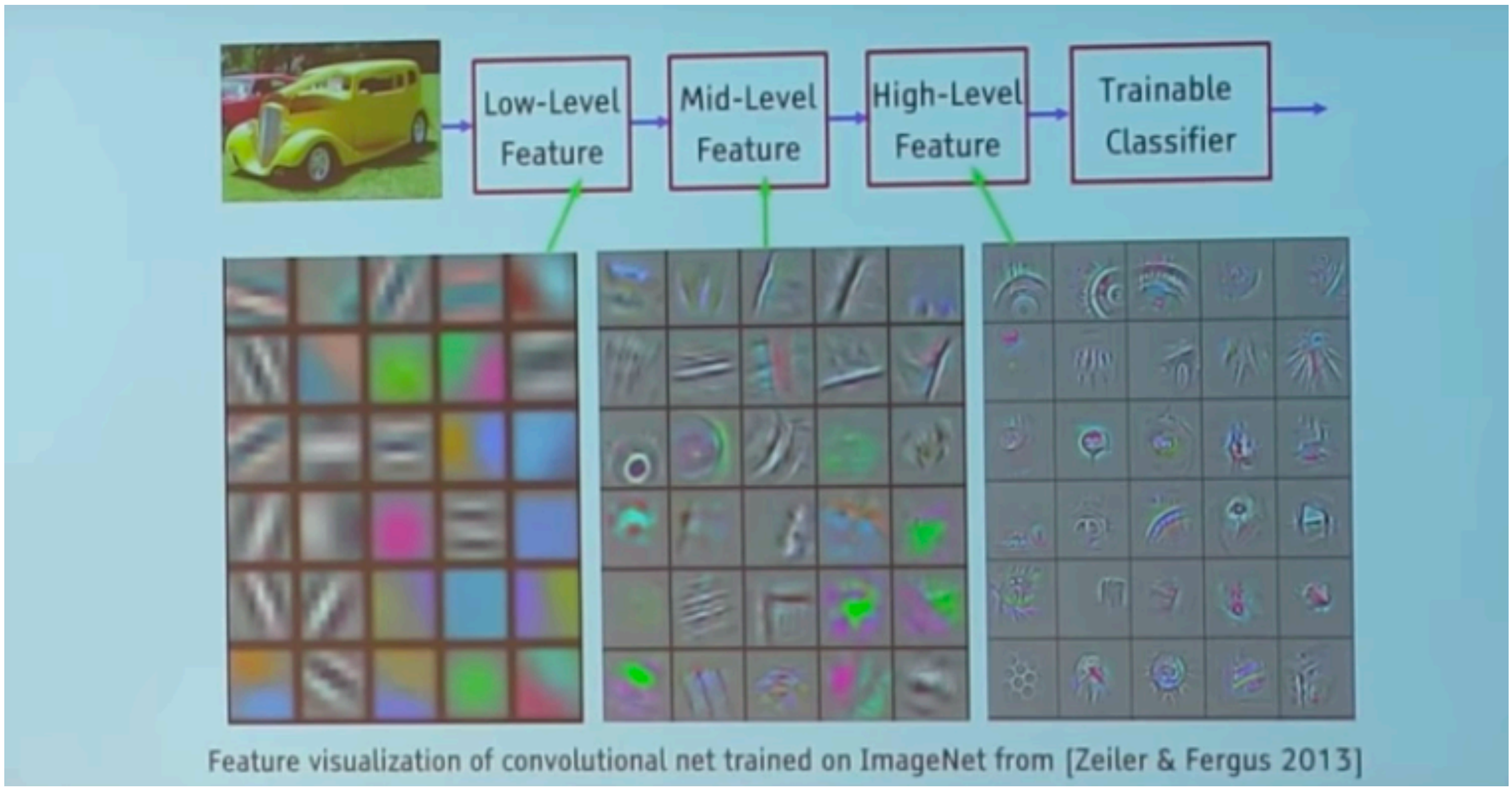


Gabor Filters

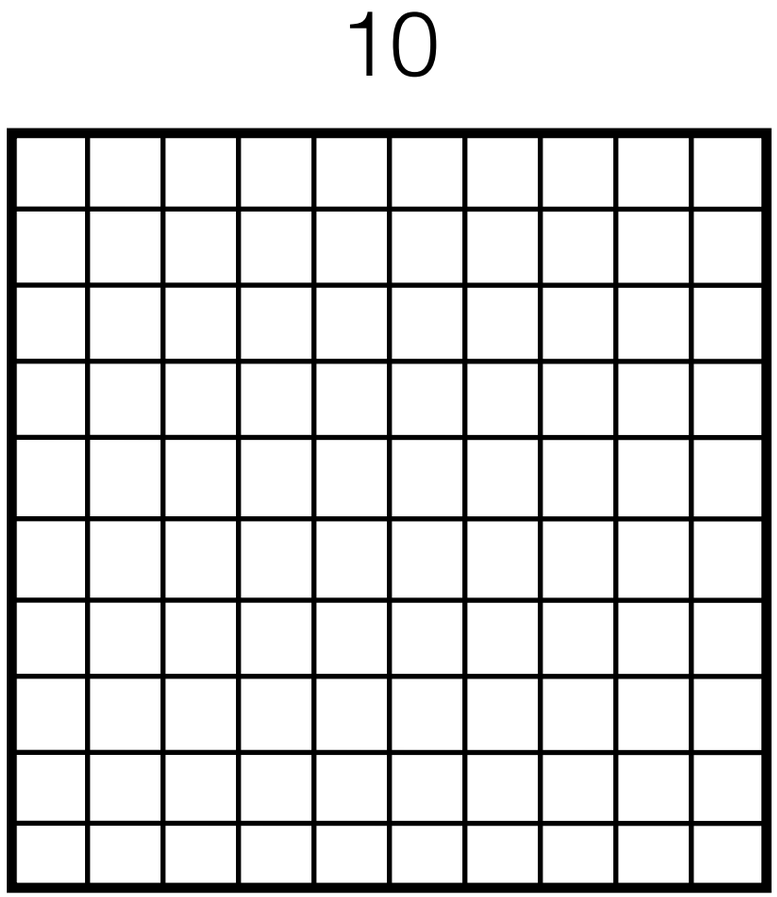


Convolutional neural networks trained for image recognition tend to learn filters which can detect edges.

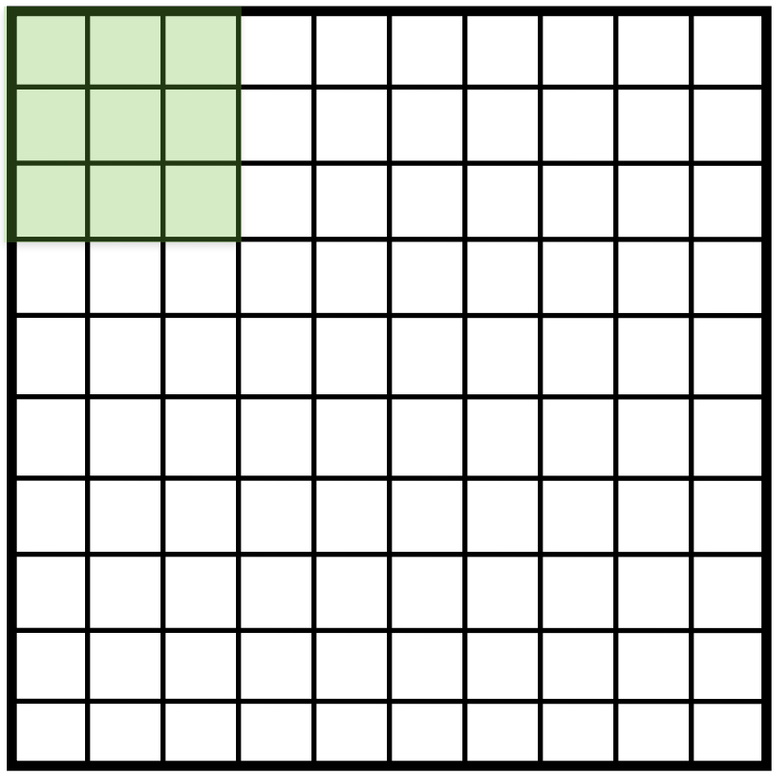
Filter learned by a CNN



Convolution Changes the Dimension



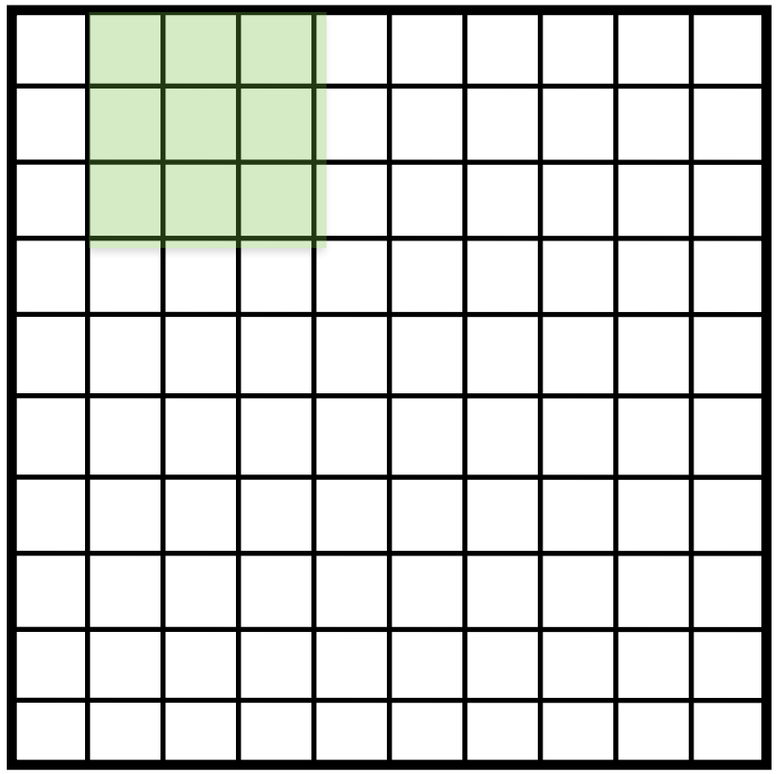
Convolution Changes the Dimension



2.1

Convolution Changes the Dimension

10

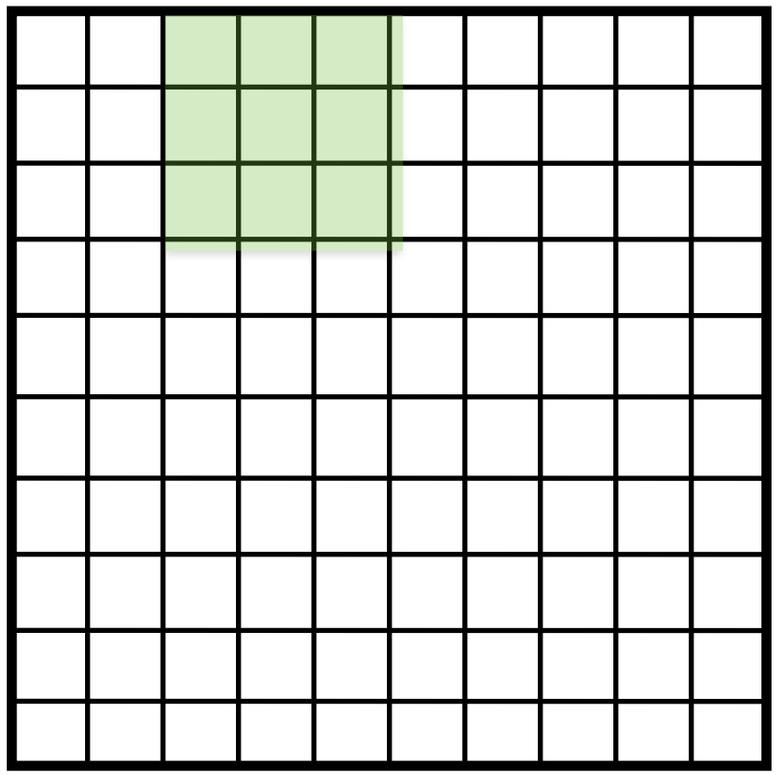


2.1 1.2

10

Convolution Changes the Dimension

10

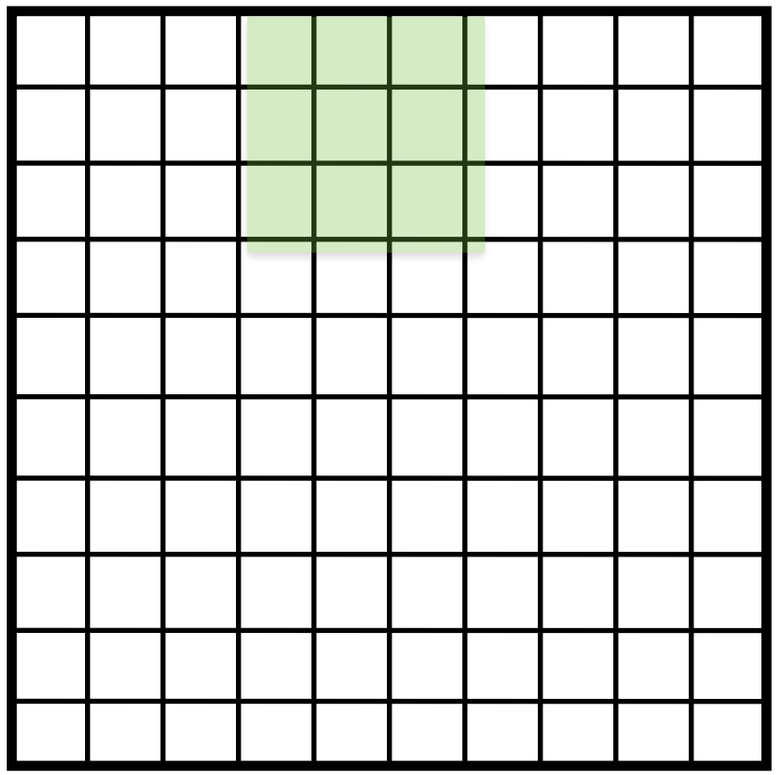


```
2.1   1.2   -2.0
```

10

Convolution Changes the Dimension

10

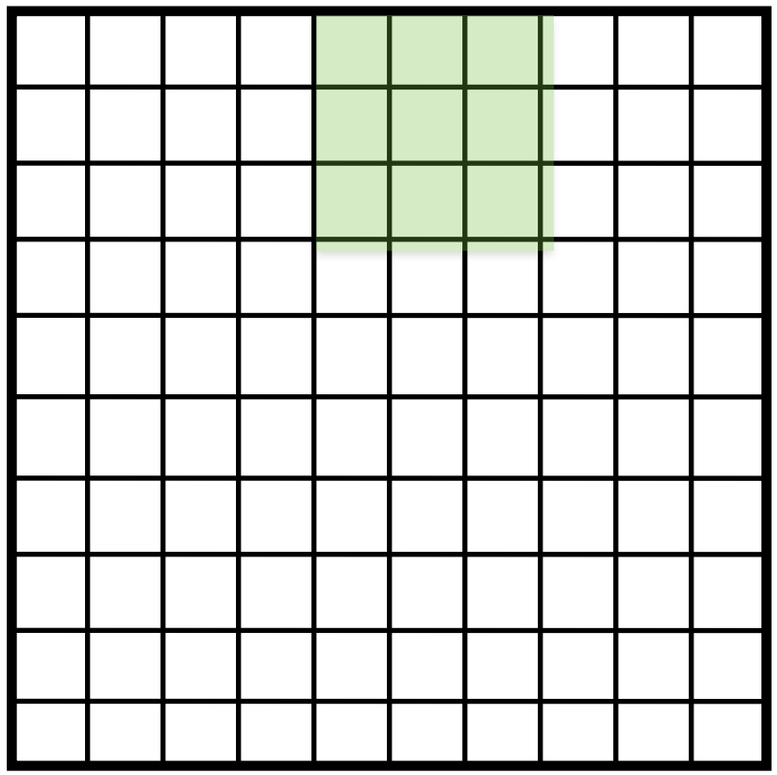


2.1 1.2 -2.0 -2.2

10

Convolution Changes the Dimension

10

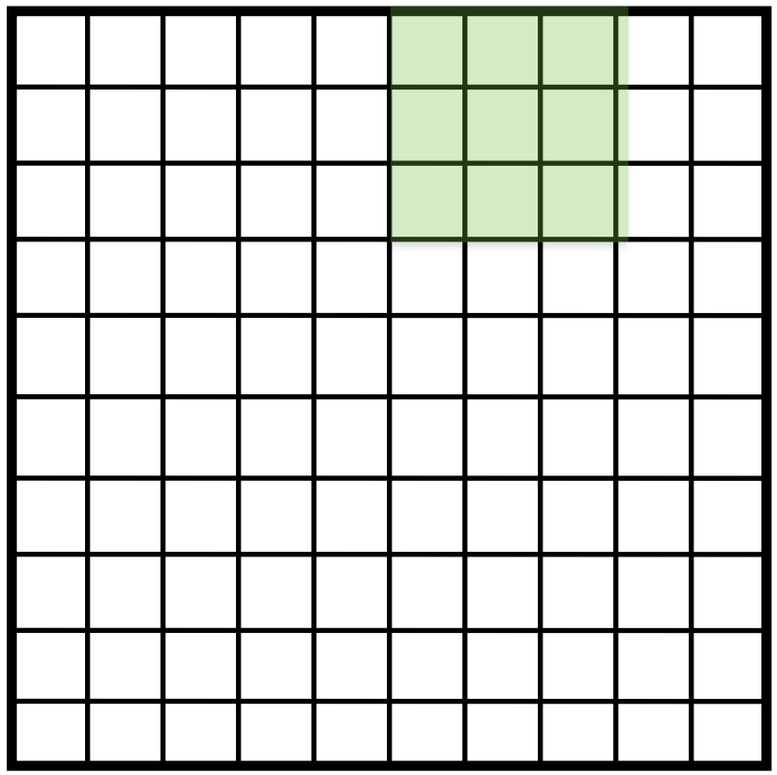


2.1 1.2 -2.0 -2.2 0.7

10

Convolution Changes the Dimension

10

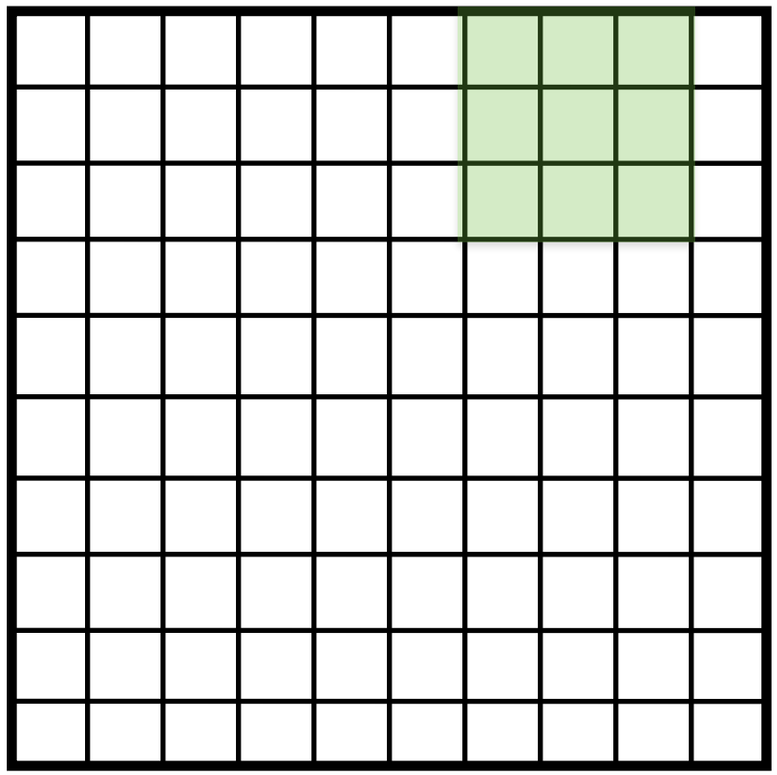


2.1 1.2 -2.0 -2.2 0.7 0.3

10

Convolution Changes the Dimension

10

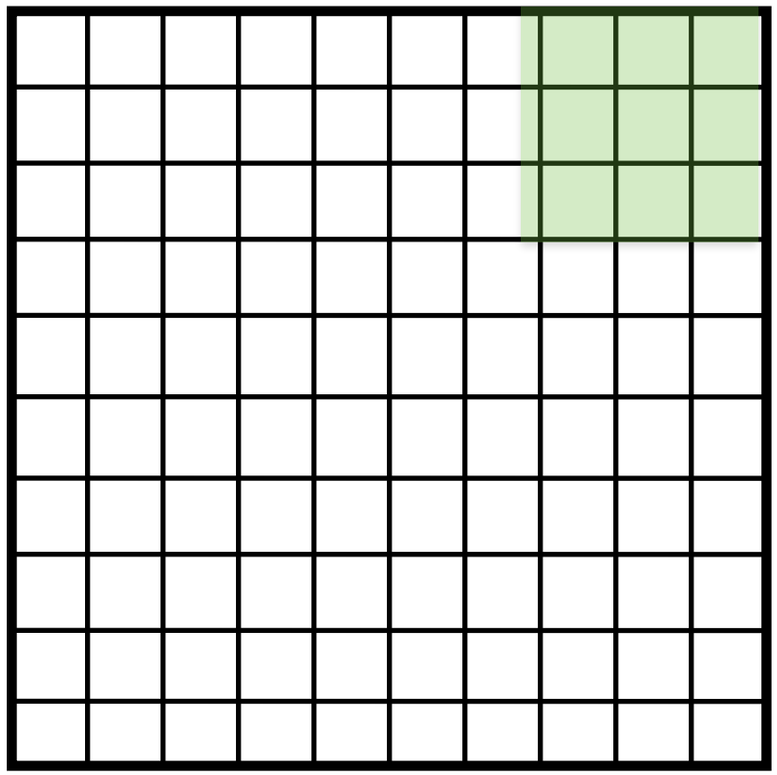


2.1 1.2 -2.0 -2.2 0.7 0.3 -1.0

10

Convolution Changes the Dimension

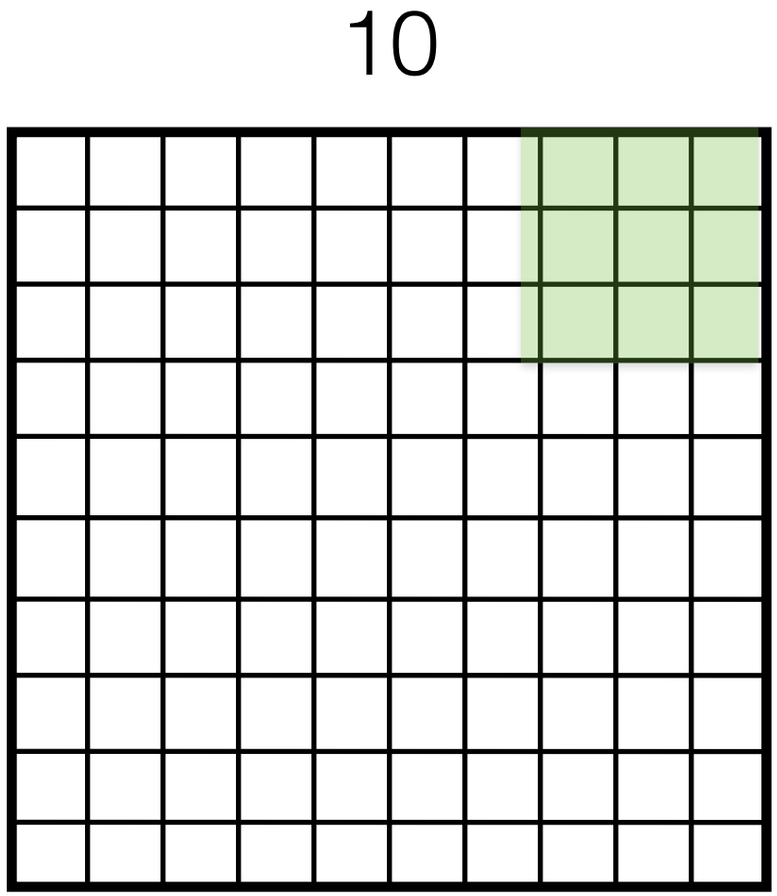
10



2.1 1.2 -2.0 -2.2 0.7 0.3 -1.0 2.5

10

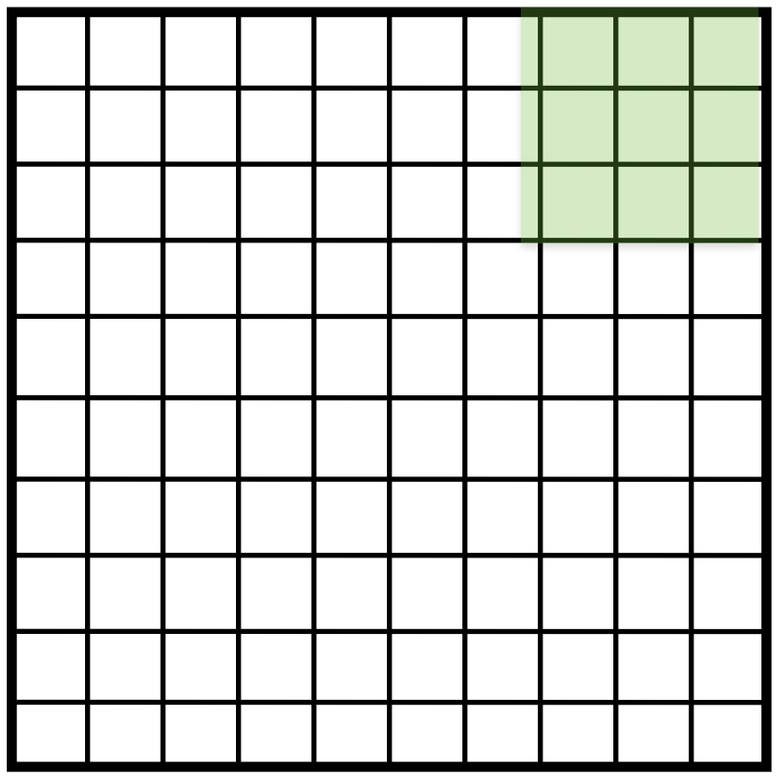
Convolution Changes the Dimension



2.1 1.2 -2.0 -2.2 0.7 0.3 -1.0 2.5

8

Convolution Changes the Dimension

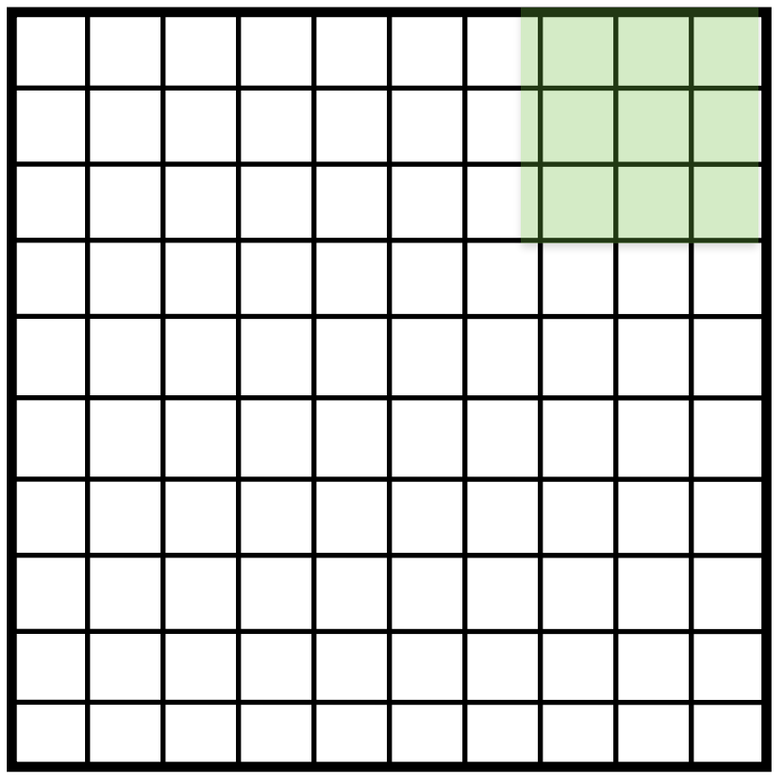


2.1 1.2 -2.0 -2.2 0.7 0.3 -1.0 2.5

8

10x10 -> 8x8

Convolution Changes the Dimension



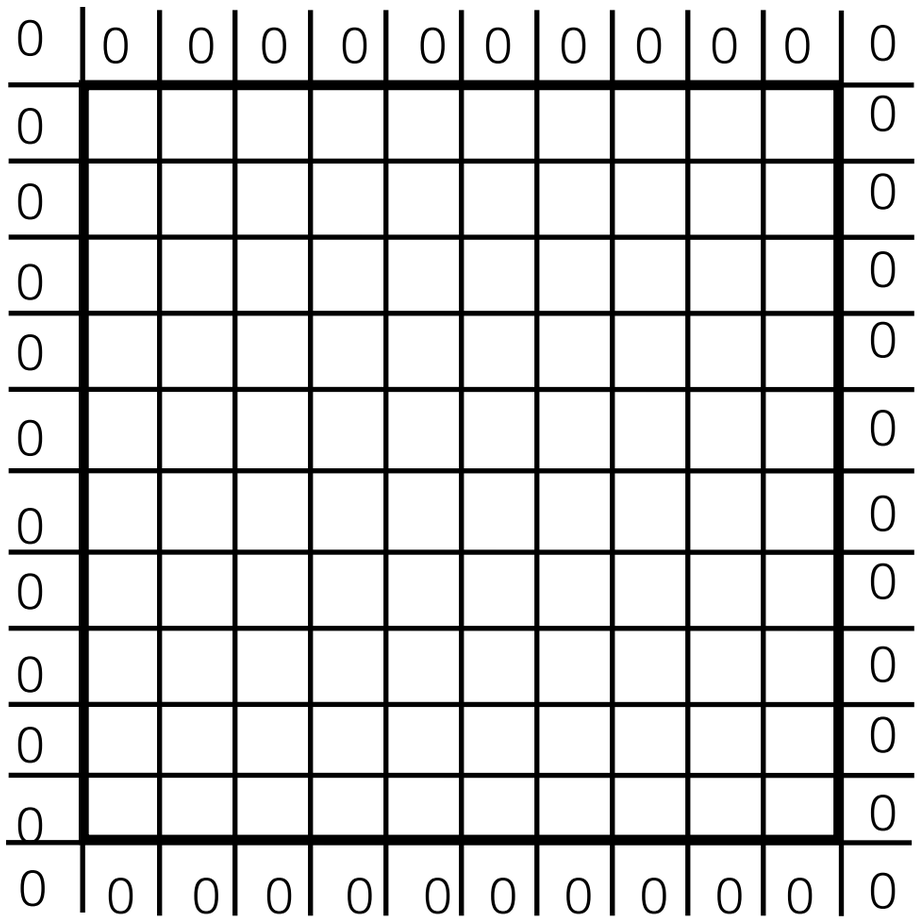
2.1 1.2 -2.0 -2.2 0.7 0.3 -1.0 2.5

8

10x10 -> 8x8

output_dim = input_dim - filter_size + 1

Padding



We add zeros at the boundaries

convolution

10x10 -> 12x12 -> 10x10

padding

Convolutional Filter with Colors

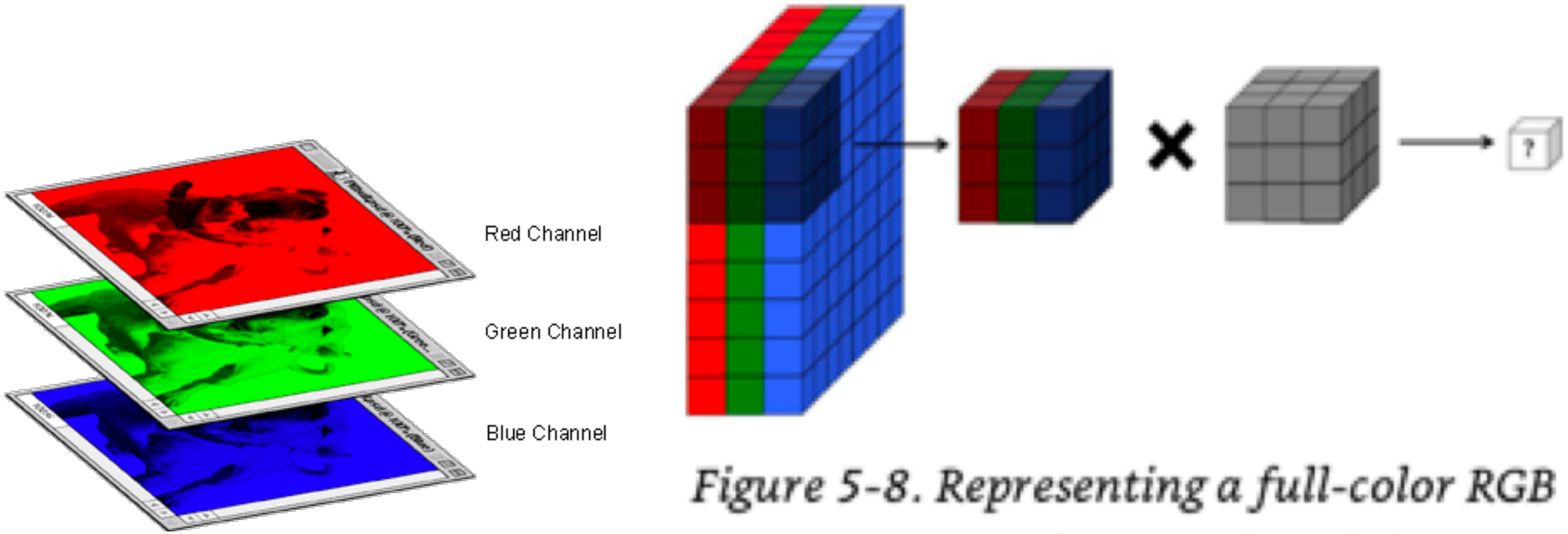
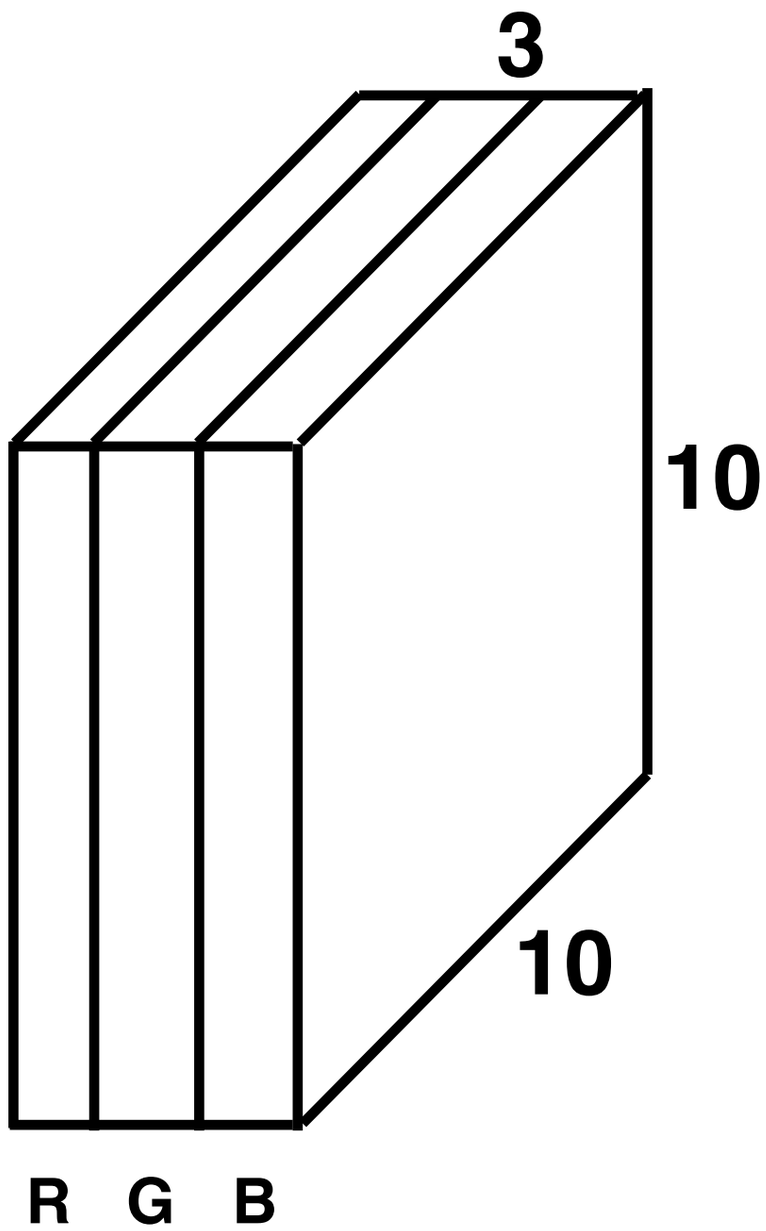


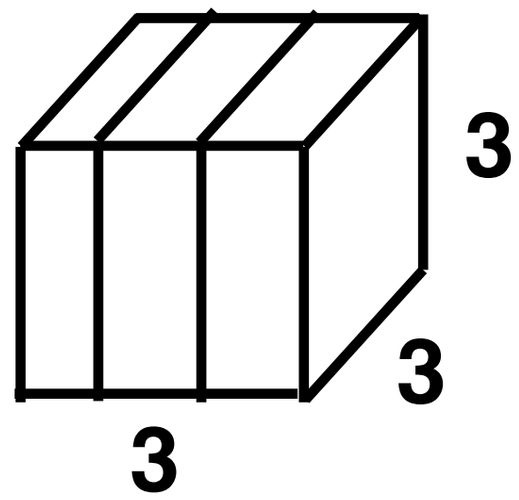
Figure 5-8. Representing a full-color RGB image as a volume and applying a volumetric convolutional filter

Example of Convolutional Layer

Input: 12x12x3



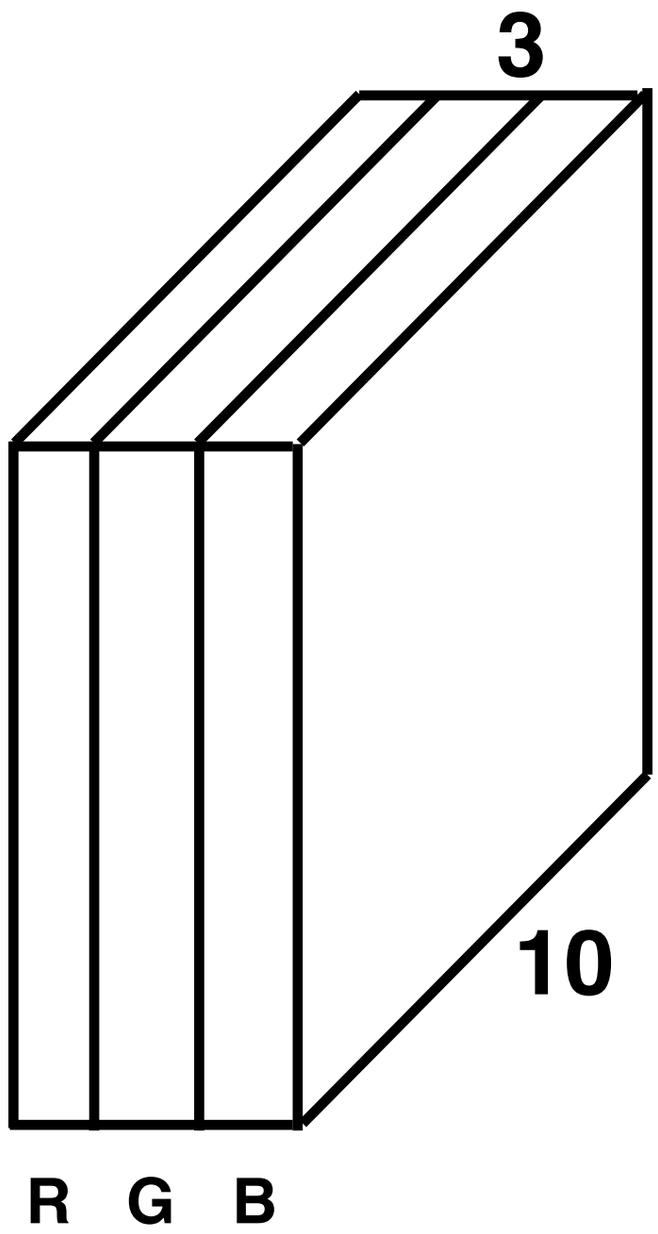
Individual filter: 3x3x3



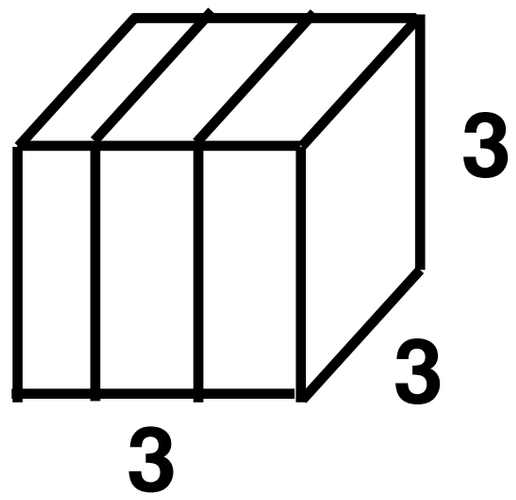
We have 5 filters altogether.

Example of Convolutional Layer

Input: 12x12x3



Individual filter: 3x3x3

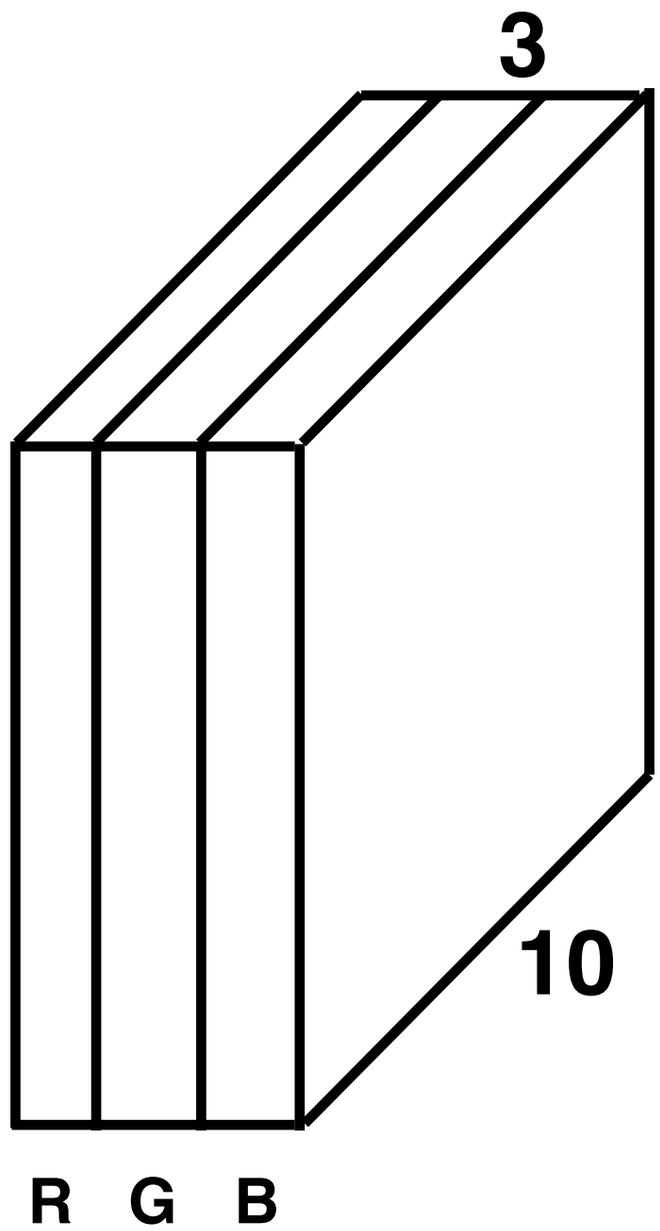


We have 5 filters altogether.

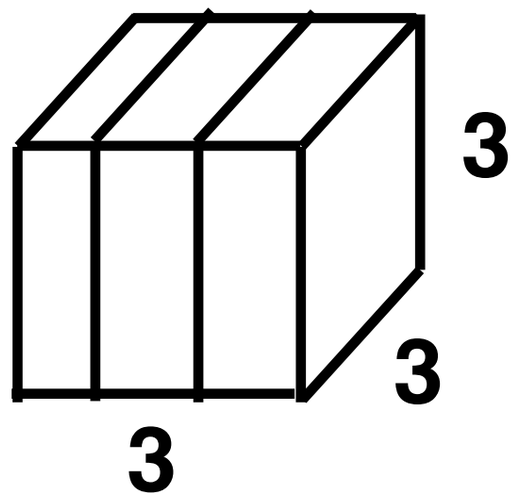
Each filter maps the 10x10x3 input into a 10x10 output because we use padding

Example of Convolutional Layer

Input: 12x12x3



Individual filter: 3x3x3

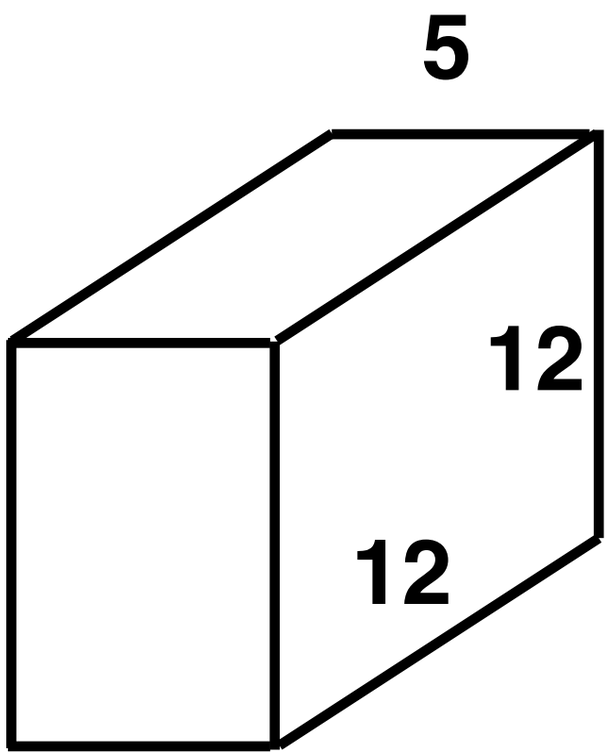


We have 5 filters altogether.

Each filter maps the 10x10x3 input into a 10x10 output because we use padding

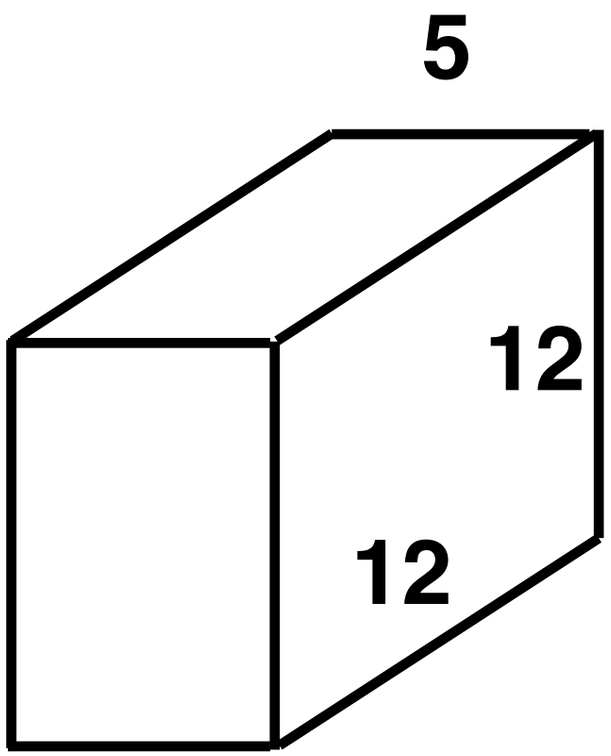
We stack the individual outputs to a 10x10x5 output for the entire layer.

Applying ReLU




```
-0.2  0.7  0.9  0.3  0.7  1.1 -0.5 -0.4  0.4  0.6  0.3  1.3
 0.7  0.5 -0.2  1.3  0.3  0.8  0.3 -0.1  0.7  1.2  1.2  0.1
 0.8 -0.   0.3  0.2 -0.2  0.   1.2  1.3  1.5  0.4 -0.1  0.5
 0.7  0.4  0.6  0.7  0.2  1.0  0.2  1.3  0.9  1.0 -0.3  1.0
-0.2 -0.  -0.3  1.4  1.4  0.1  0.5  1.3  1.1  1.   0.1  0.5
 0.9  1.1  1.3  0.5  1.2  1.1  0.3  0.4 -0.5  0.6  1.1 -0.4
 0.7 -0.1 -0.   0.9 -0.5  0.1 -0.   0.8  0.1  1.2 -0.2 -0.
 0.5  0.7  0.5  1.0  0.1 -0.2  0.8  0.8  0.5 -0.   0.5 -0.4
-0.5  0.3  0.4  0.2 -0.2 -0.4 -0.   1.3 -0.3  1.3  1.4  1.5
-0.4  1.2  0.4 -0.4  0.2  1.4  0.  -0.3  0.6  1.2  1.0  0.1
 1.0  1.2  0.6  1.1 -0.5 -0.1  0.   0.2  1.0  0.8  1.3 -0.0
 0.9  1.1  1.5 -0.1  0.7 -0.2  1.4  1.1  1.2  1.0 -0.1  0.3
```

Applying ReLU

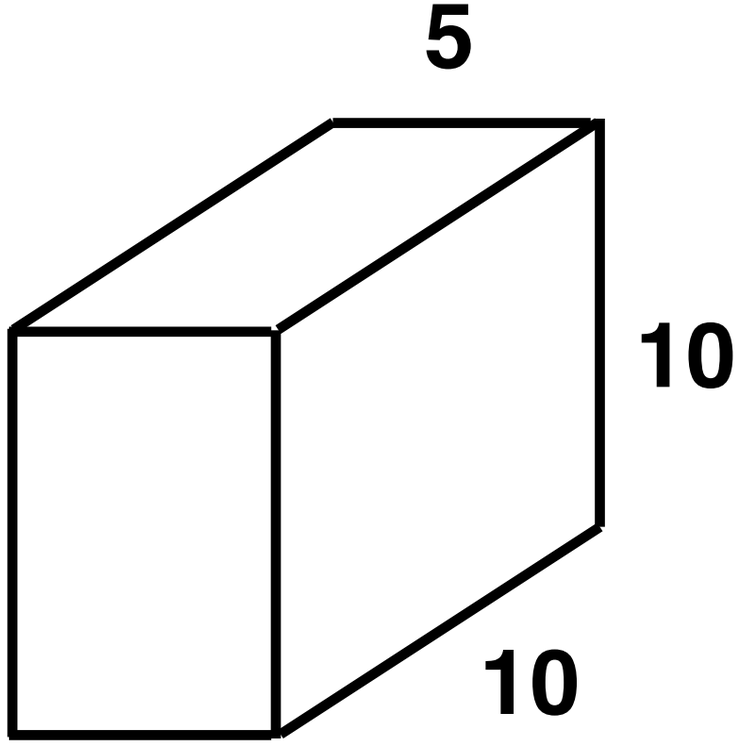



```
-0.2  0.7  0.9  0.3  0.7  1.1 -0.5 -0.4  0.4  0.6  0.3  1.3
 0.7  0.5 -0.2  1.3  0.3  0.8  0.3 -0.1  0.7  1.2  1.2  0.1
 0.8 -0.   0.3  0.2 -0.2  0.   1.2  1.3  1.5  0.4 -0.1  0.5
 0.7  0.4  0.6  0.7  0.2  1.0  0.2  1.3  0.9  1.0 -0.3  1.0
-0.2 -0.  -0.3  1.4  1.4  0.1  0.5  1.3  1.1  1.   0.1  0.5
 0.9  1.1  1.3  0.5  1.2  1.1  0.3  0.4 -0.5  0.6  1.1 -0.4
 0.7 -0.1 -0.   0.9 -0.5  0.1 -0.   0.8  0.1  1.2 -0.2 -0.
 0.5  0.7  0.5  1.0  0.1 -0.2  0.8  0.8  0.5 -0.   0.5 -0.4
-0.5  0.3  0.4  0.2 -0.2 -0.4 -0.   1.3 -0.3  1.3  1.4  1.5
-0.4  1.2  0.4 -0.4  0.2  1.4  0.  -0.3  0.6  1.2  1.0  0.1
 1.0  1.2  0.6  1.1 -0.5 -0.1  0.   0.2  1.0  0.8  1.3 -0.0
 0.9  1.1  1.5 -0.1  0.7 -0.2  1.4  1.1  1.2  1.0 -0.1  0.3
```

```
  0  0.7  0.9  0.3  0.7  1.1    0    0  0.4  0.6  0.3  1.3
0.7  0.5    0  1.3  0.3  0.8  0.3    0  0.7  1.2  1.2  0.1
0.8    0  0.3  0.2    0    0  1.2  1.3  1.5  0.4    0  0.5
0.7  0.4  0.6  0.7  0.2  1.0  0.2  1.3  0.9  1.0    0  1.0
  0    0    0  1.4  1.4  0.1  0.5  1.3  1.1  1.0  0.1  0.5
0.9  1.1  1.3  0.5  1.2  1.1  0.3  0.4    0  0.6  1.1    0
0.7    0    0  0.9    0  0.1    0  0.8  0.1  1.2    0    0
0.5  0.7  0.5  1.   0.1    0  0.8  0.8  0.5    0  0.5    0
  0  0.3  0.4  0.2    0    0    0  1.3    0  1.3  1.4  1.5
  0  1.2  0.4    0  0.2  1.4  0.      0  0.6  1.2  1.   0.1
1.   1.2  0.6  1.1    0    0  0.   0.2  1.   0.8  1.3    0
0.9  1.1  1.5    0  0.7    0  1.4  1.1  1.2  1.      0  0.3
```

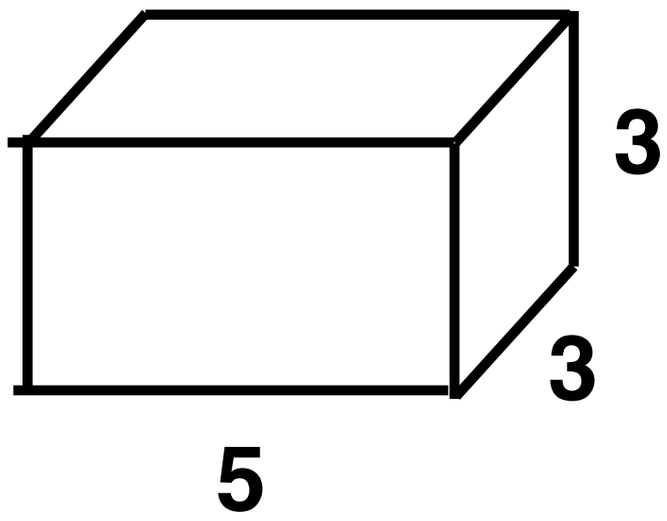
Additional Layers

Input:

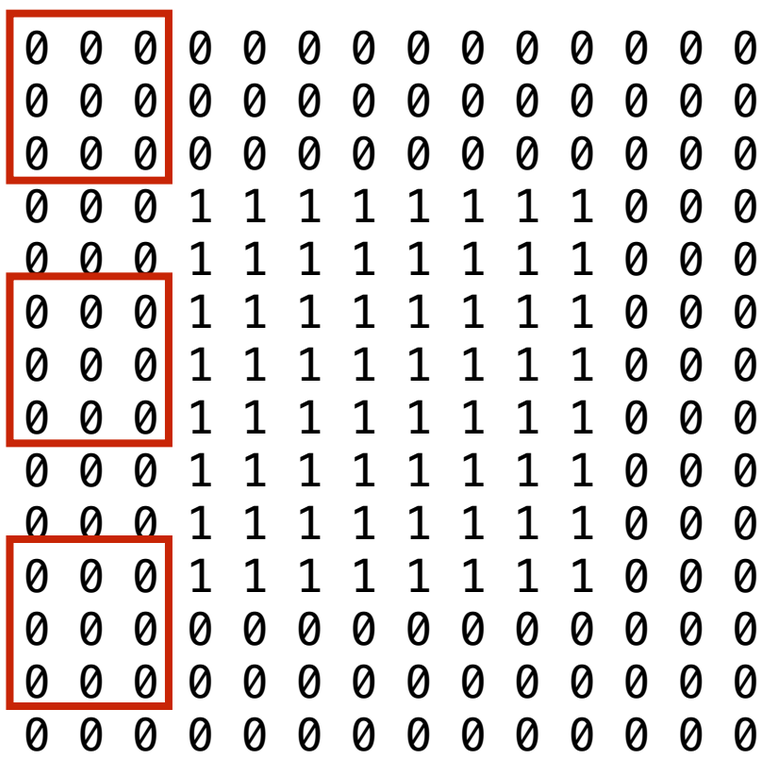


- Since the input has depth 5, we need filters of depth 5.
- The number of filters here again determines the depth of the output.

Filter:



Stride



Stride 1

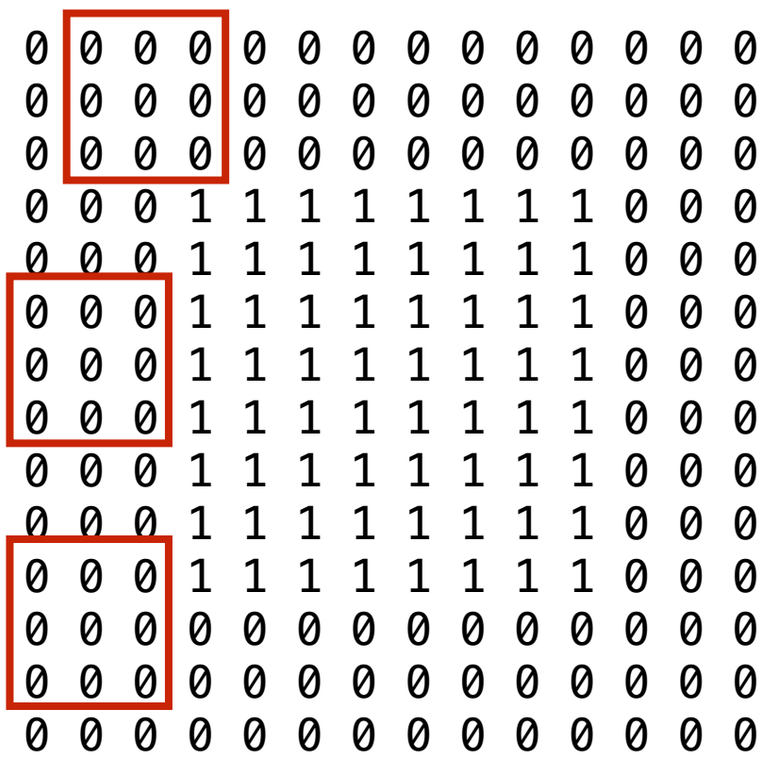
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

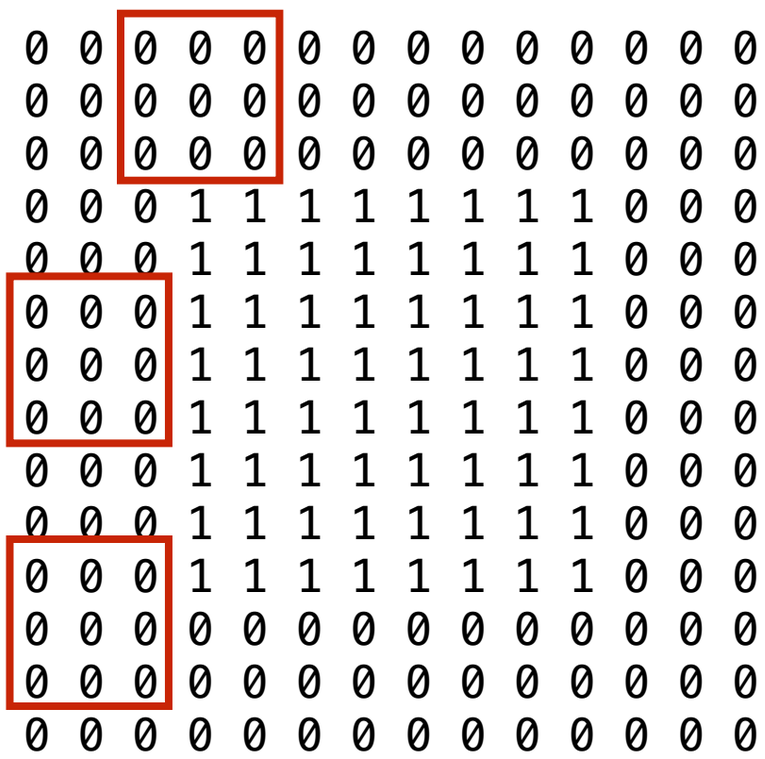
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

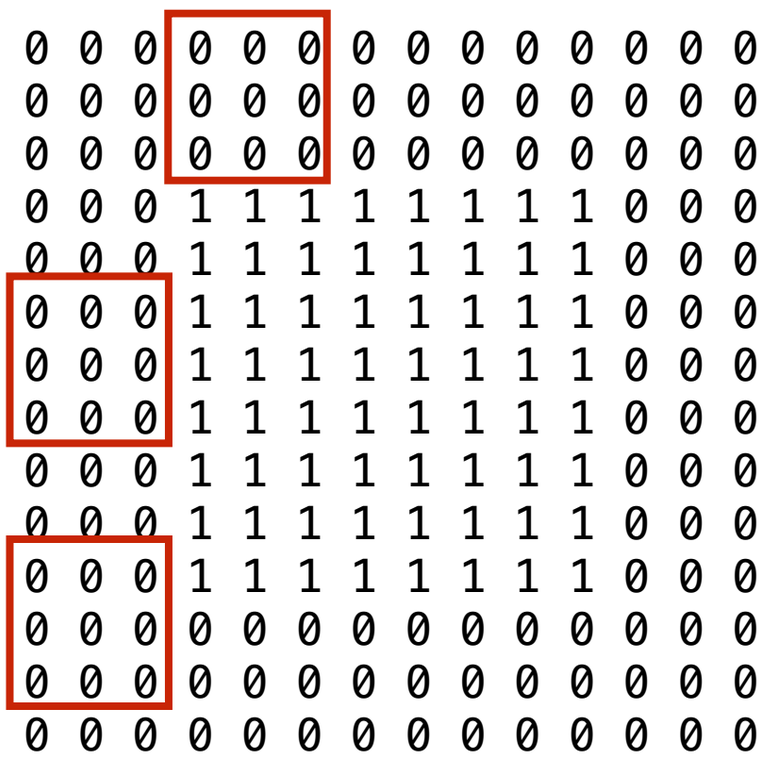
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

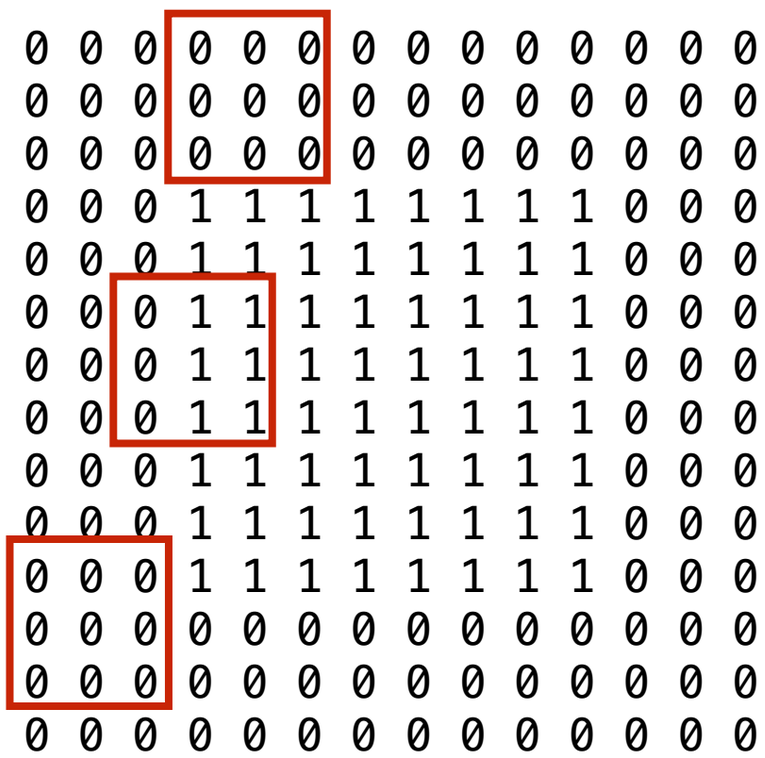
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

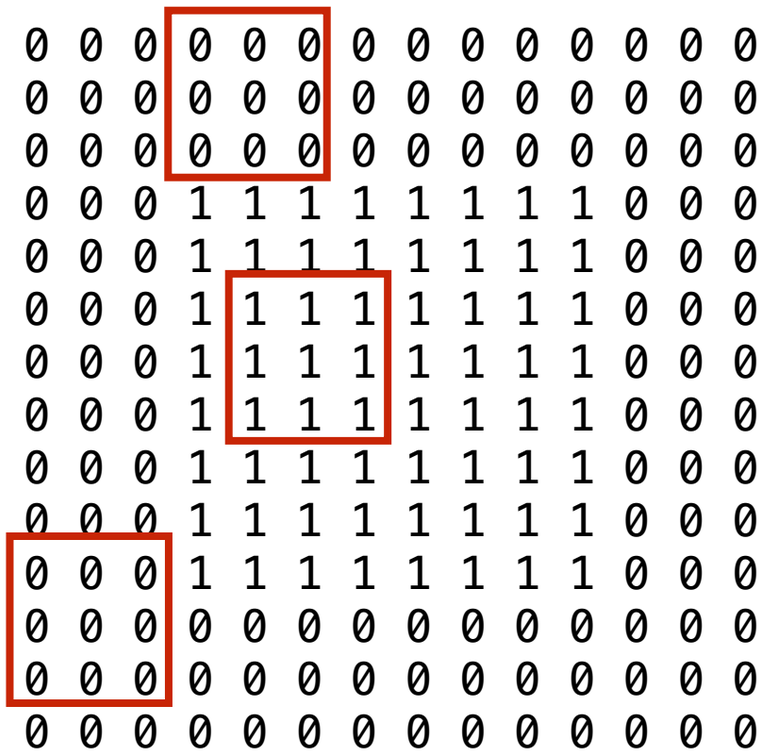
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

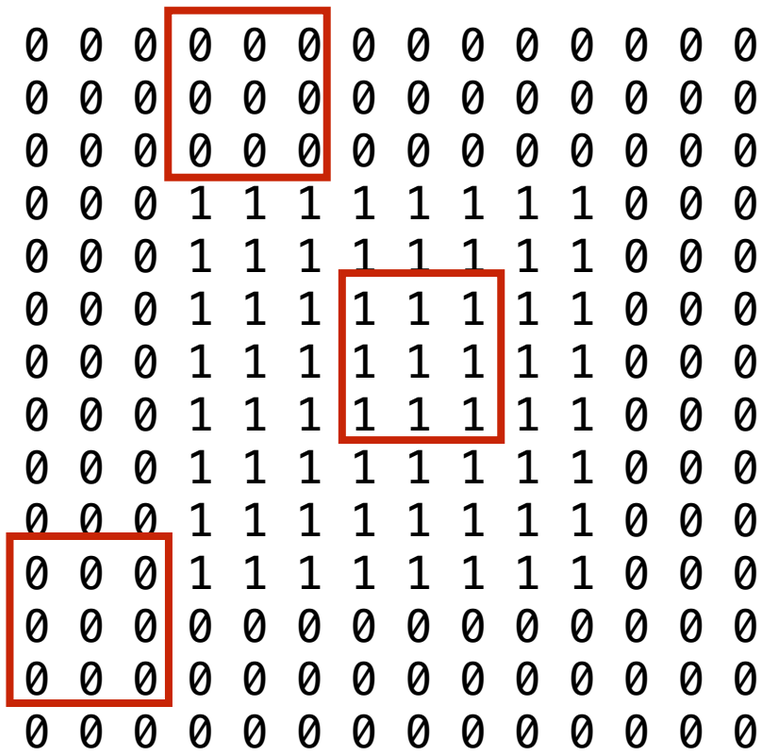
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

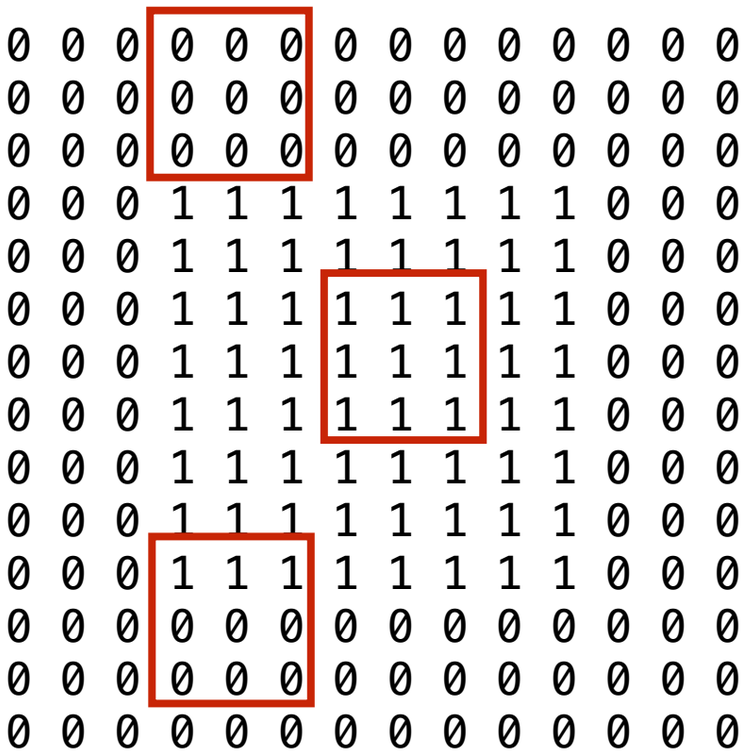
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

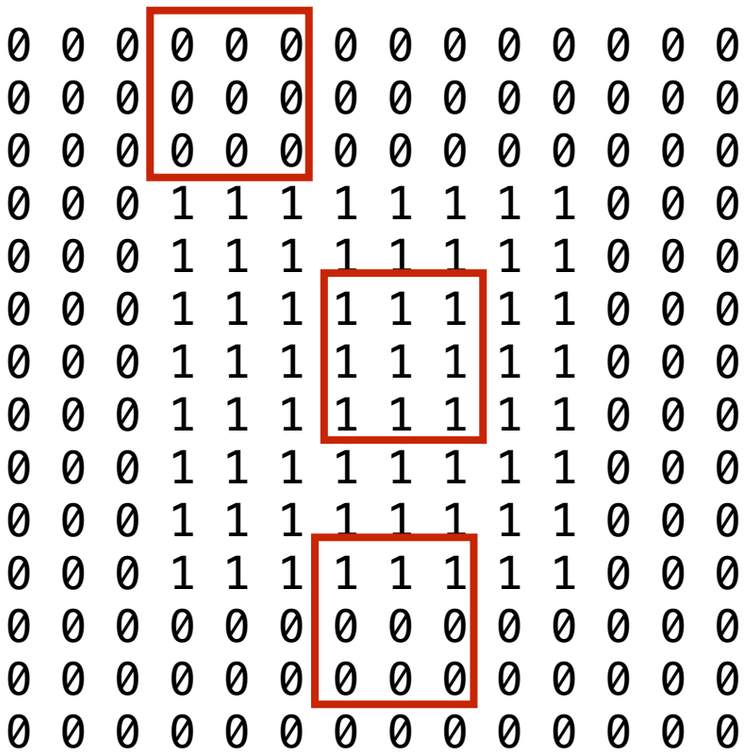
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

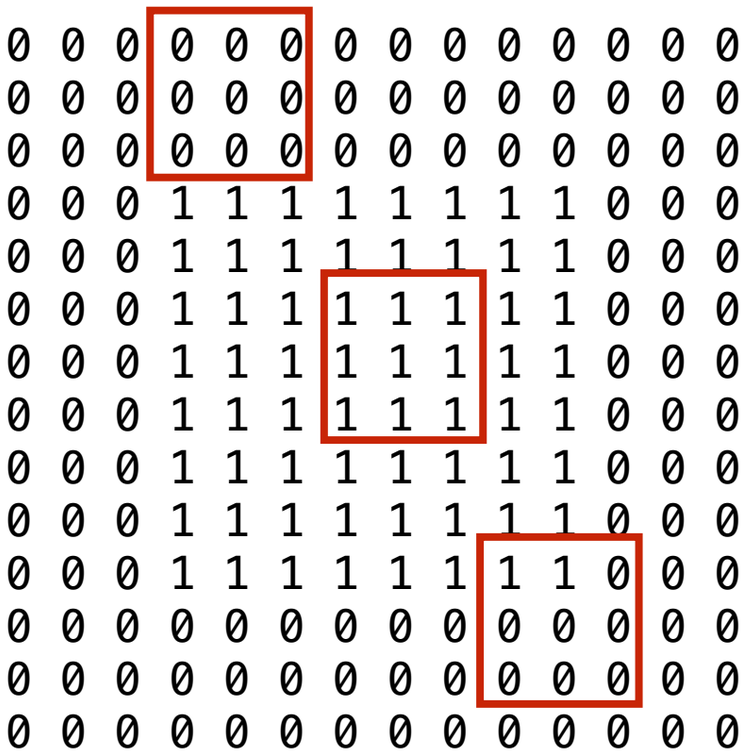
Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Stride



Stride 1

Stride 2

Stride 3

Greater stride can improve efficiency because we compute fewer dot products.

Greater stride also decreases resolution

Fully specifying a Convolutional Layer

1. We need to specify the input dimension,
2. filter size,
3. number of filters,
4. padding size and
5. stride.

Fully specifying a Convolutional Layer

1. Input size: 12x12x3,
2. filter size: 3x3x3,
3. number of filters: 5
4. padding: 1
5. stride: 1

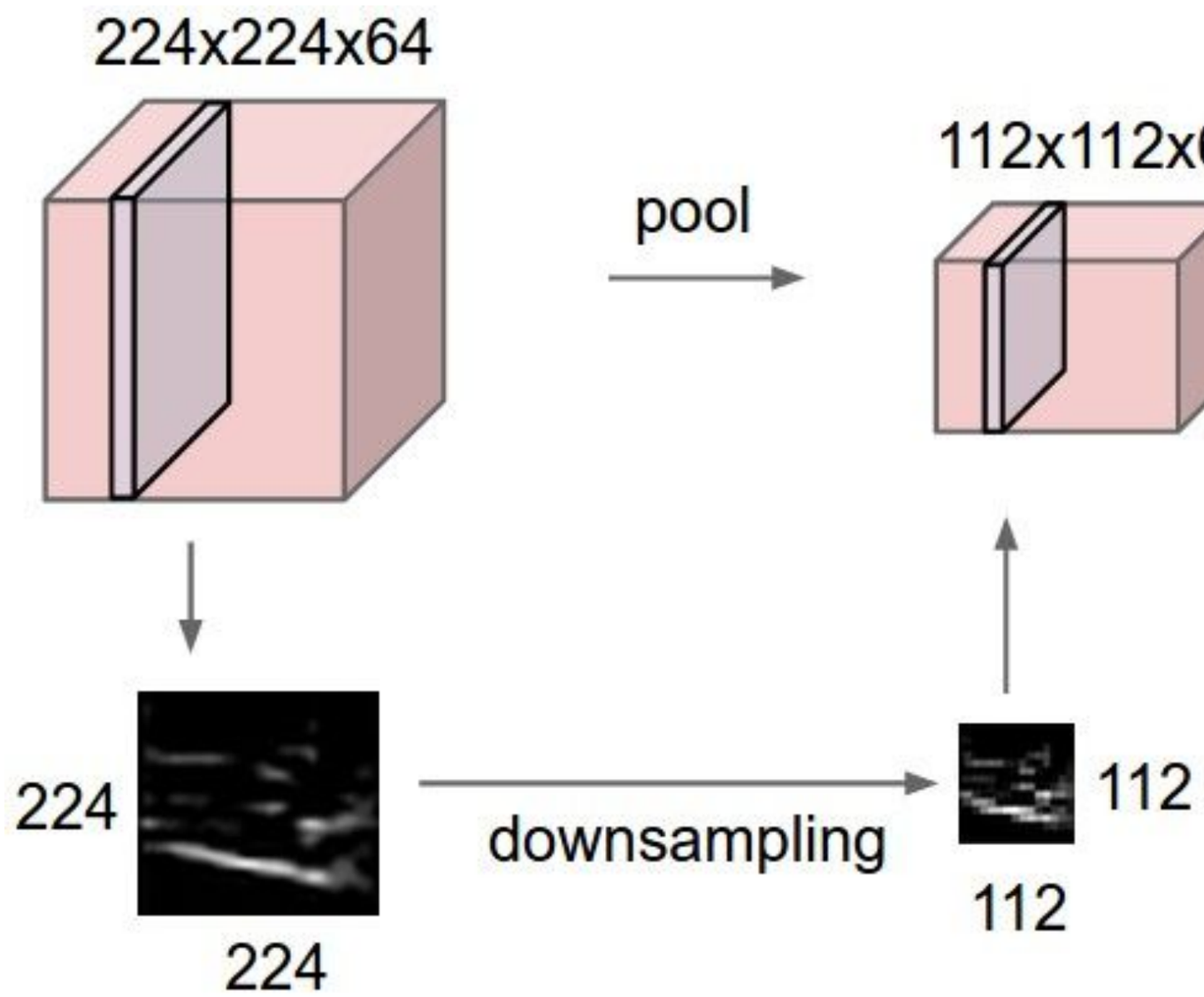
Input:

12x12x3

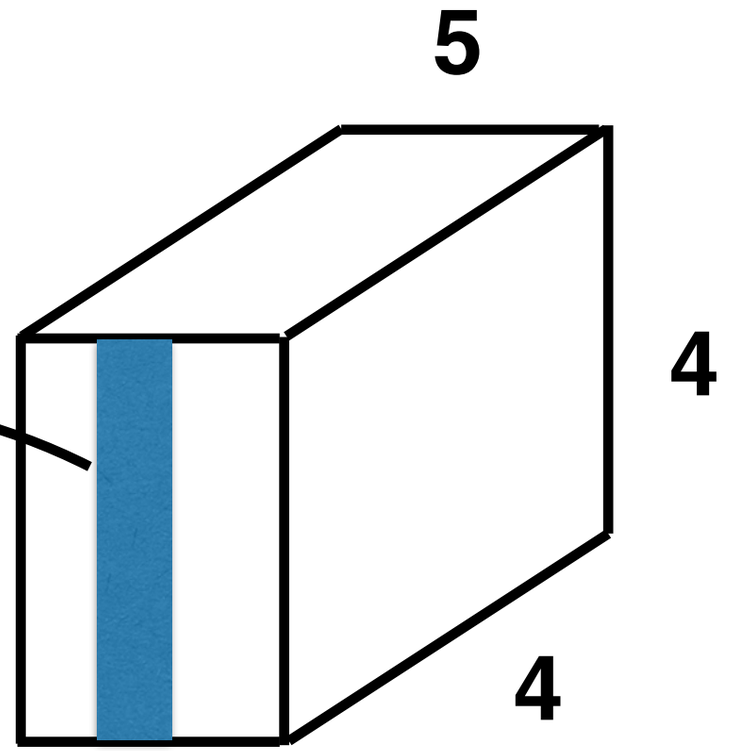
Output:

12x12x5

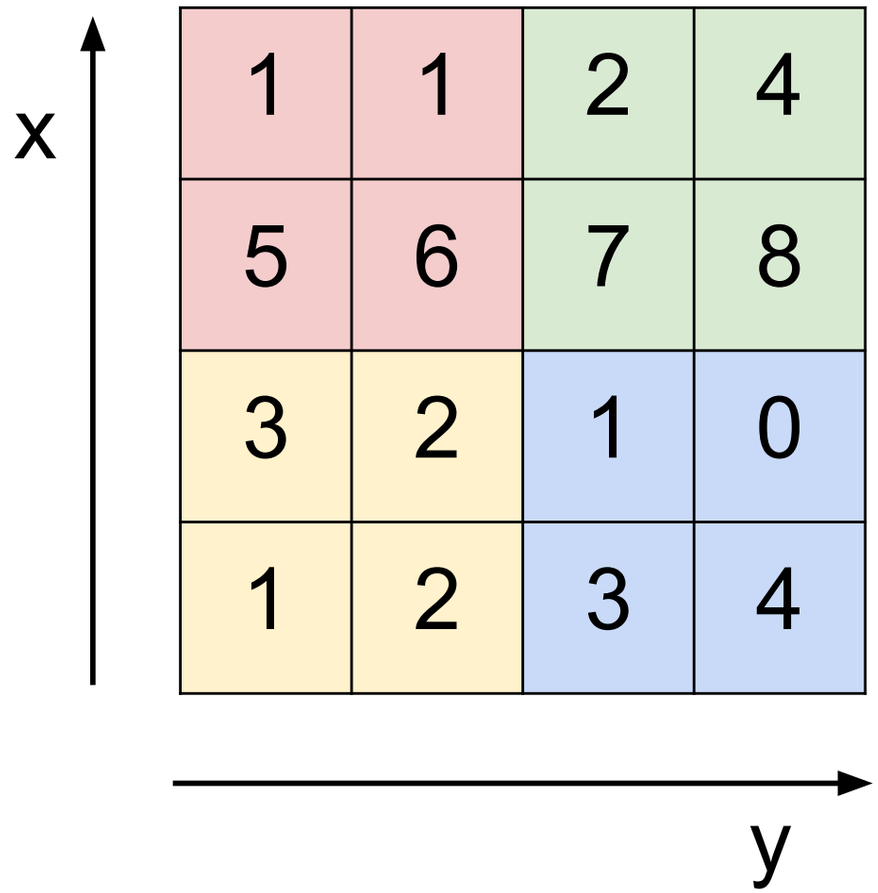
Pooling Layers



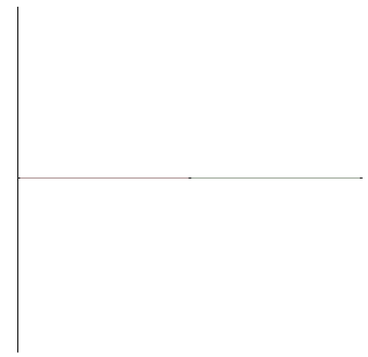
Max Pooling



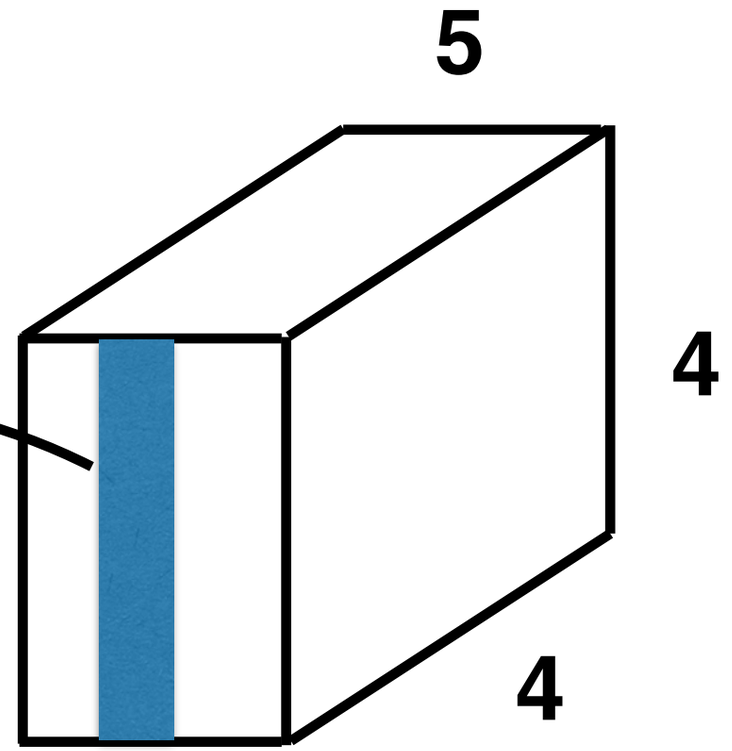
Single depth slice



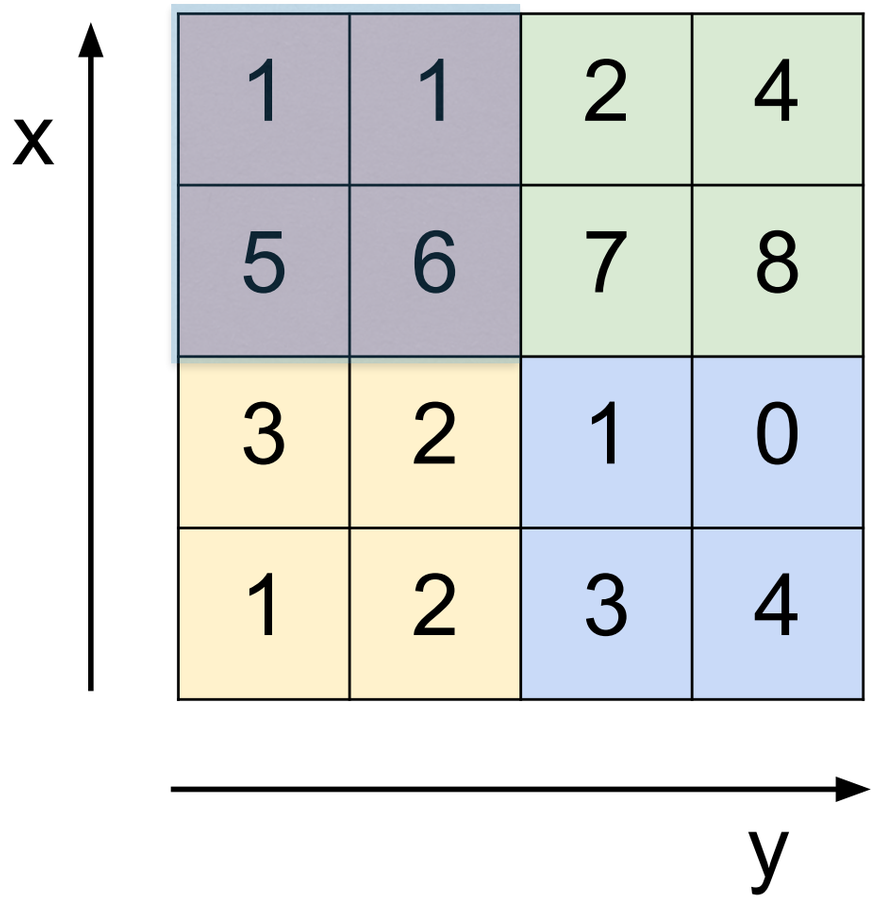
max pool with 2x2 filters
and stride 2



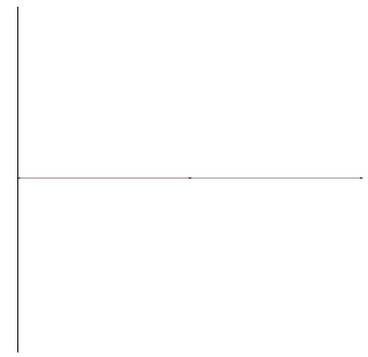
Max Pooling



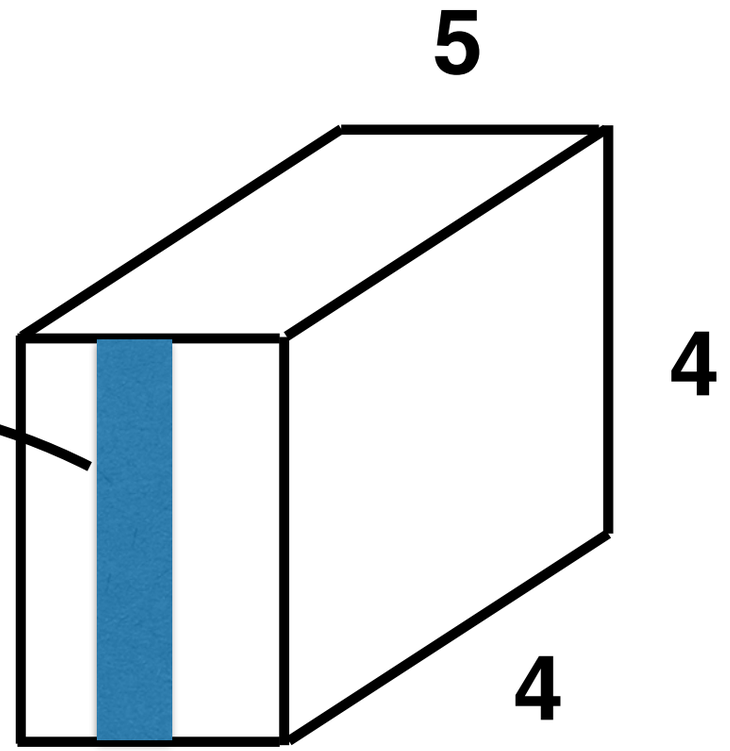
Single depth slice



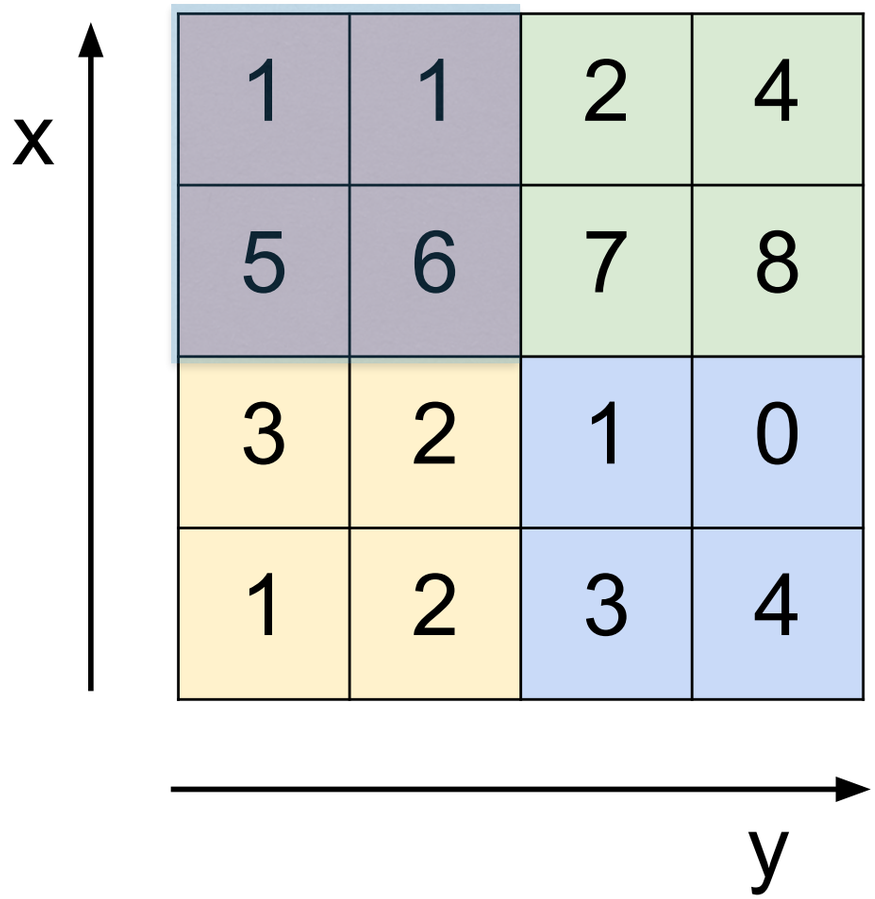
max pool with 2x2 filters
and stride 2



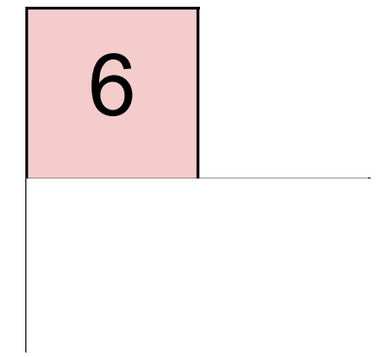
Max Pooling



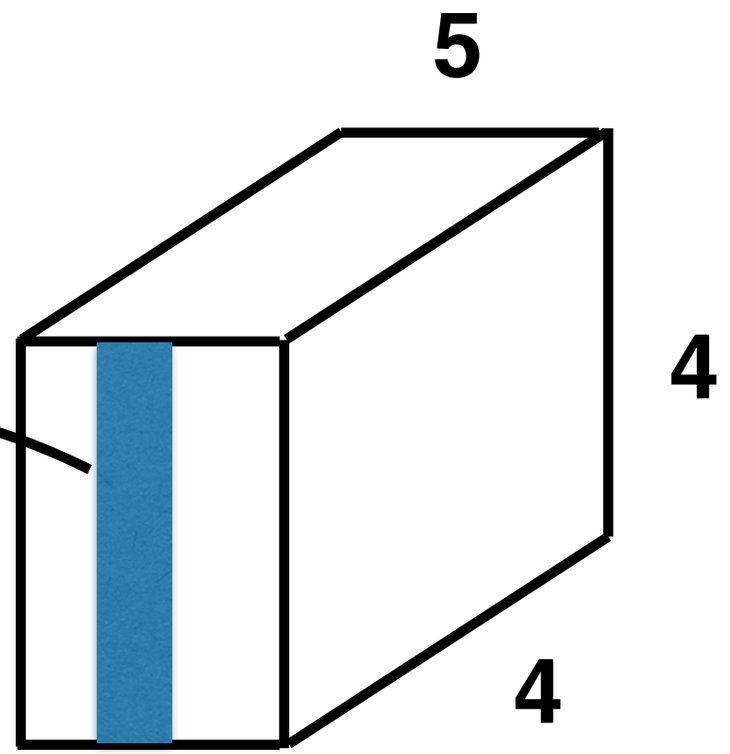
Single depth slice



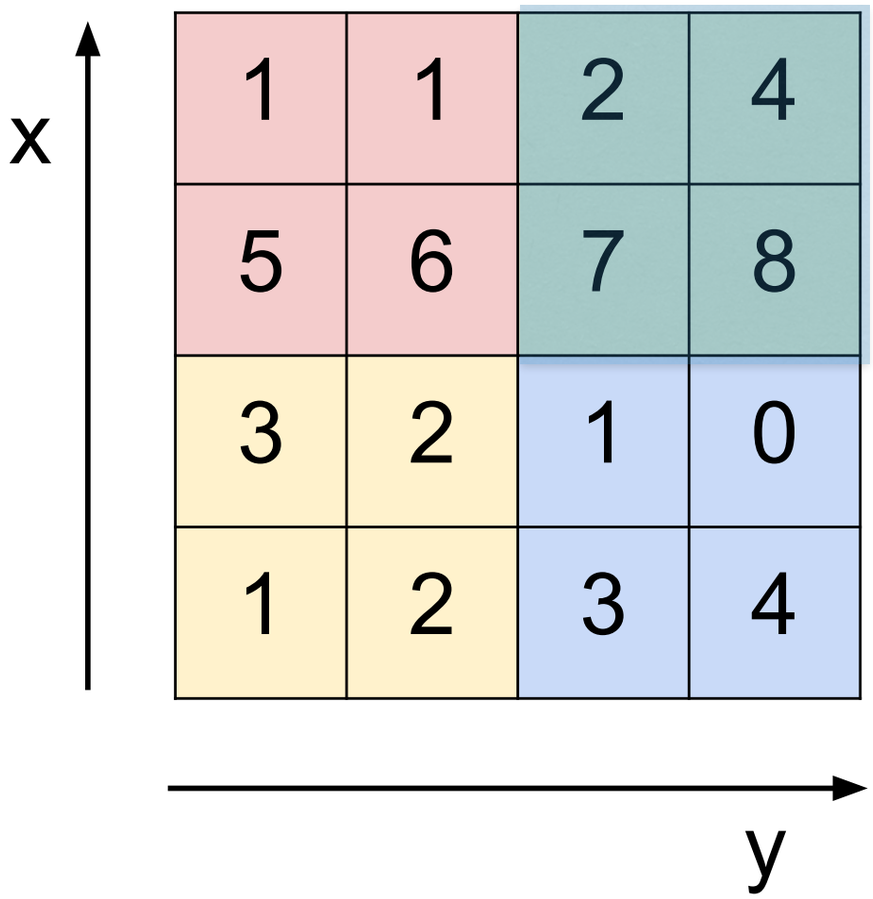
max pool with 2x2 filters
and stride 2



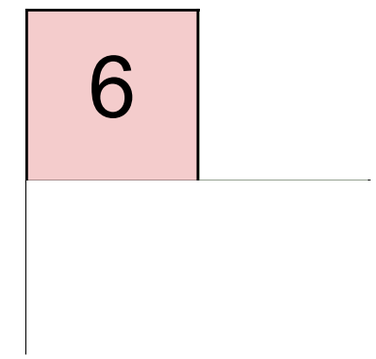
Max Pooling



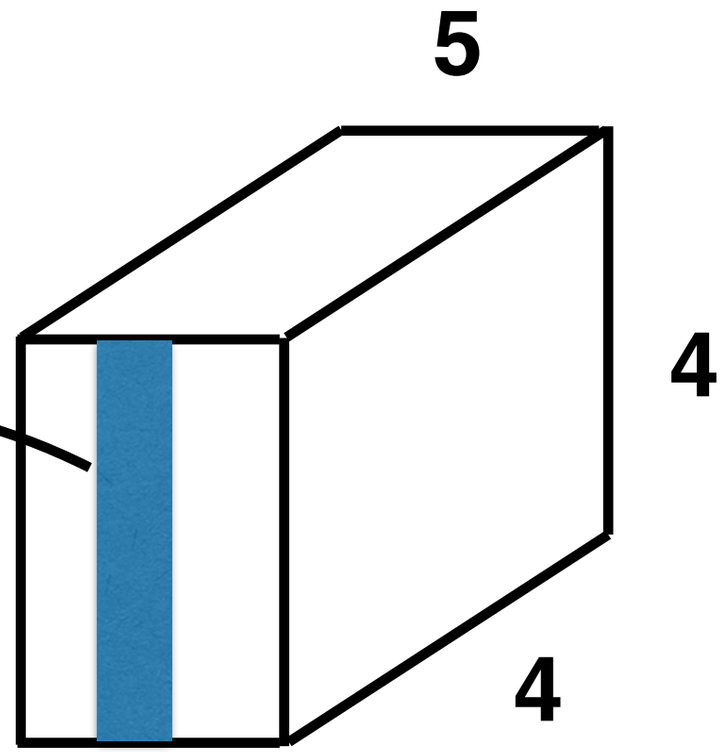
Single depth slice



max pool with 2x2 filters
and stride 2



Max Pooling



Single depth slice

x

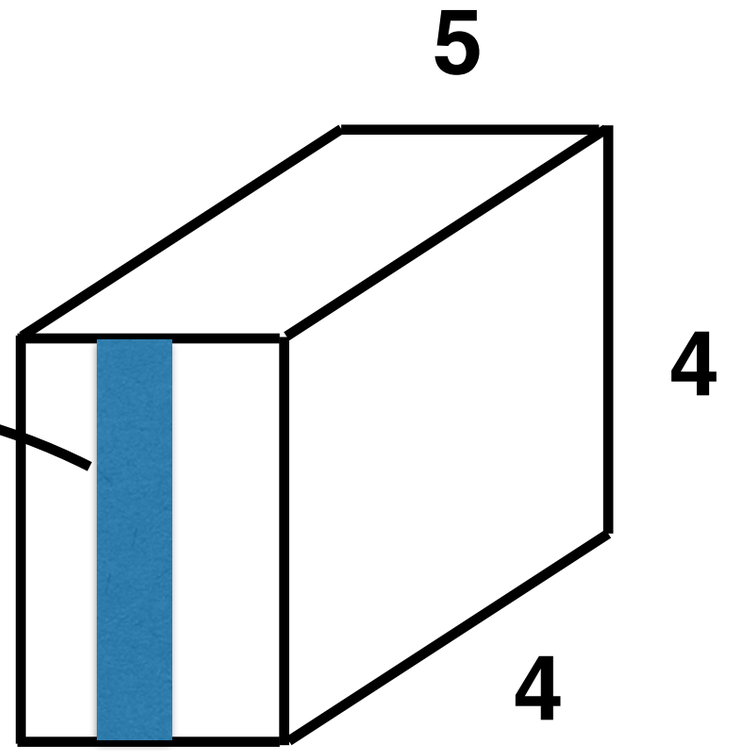
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

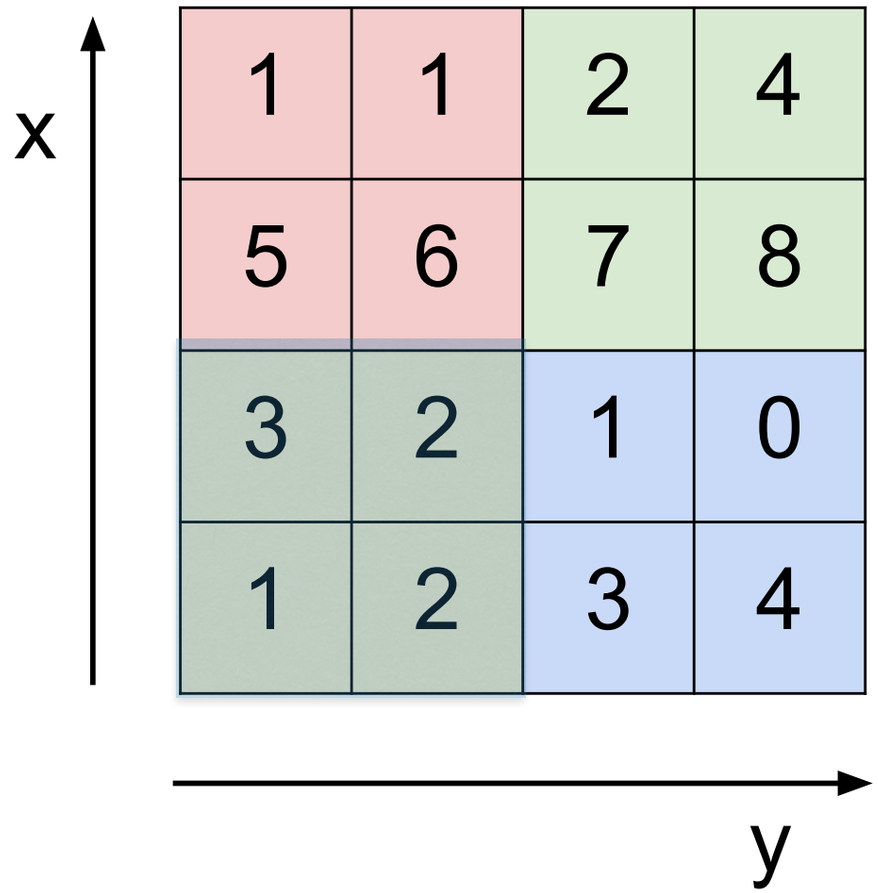
max pool with 2x2 filters and stride 2

6	8

Max Pooling



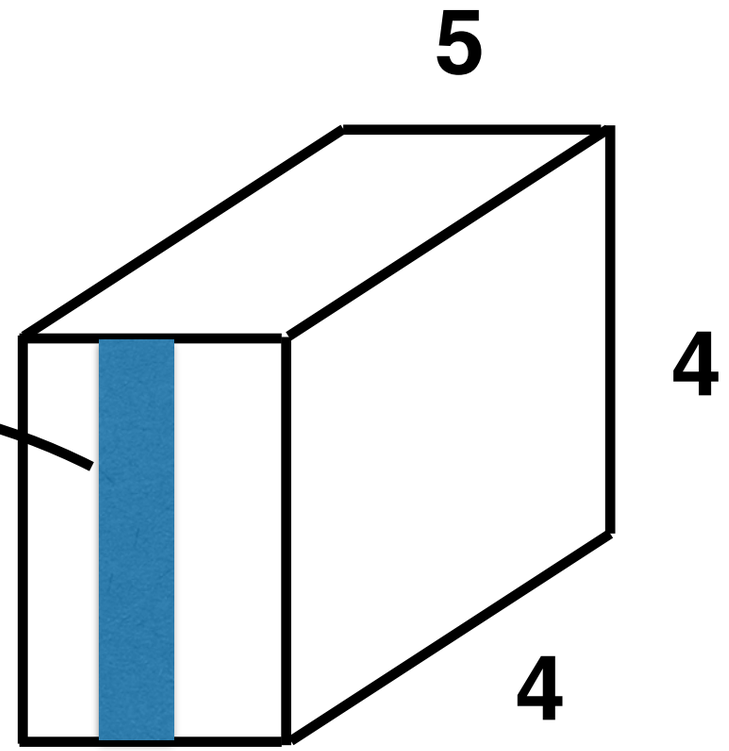
Single depth slice



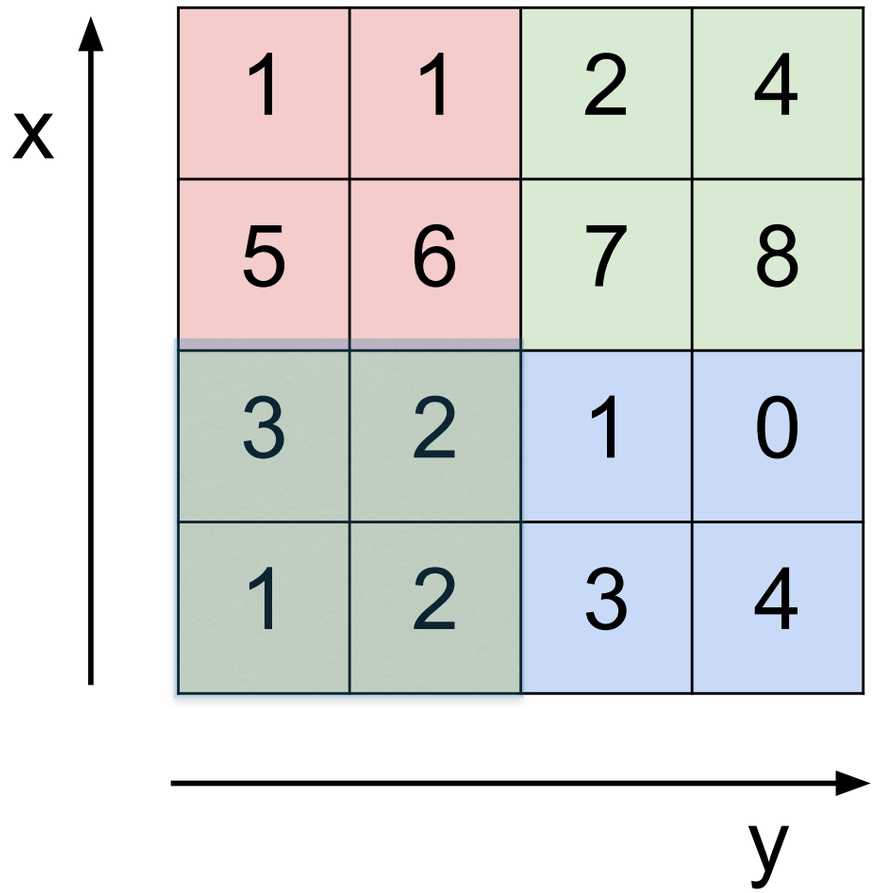
max pool with 2x2 filters
and stride 2

6	8

Max Pooling



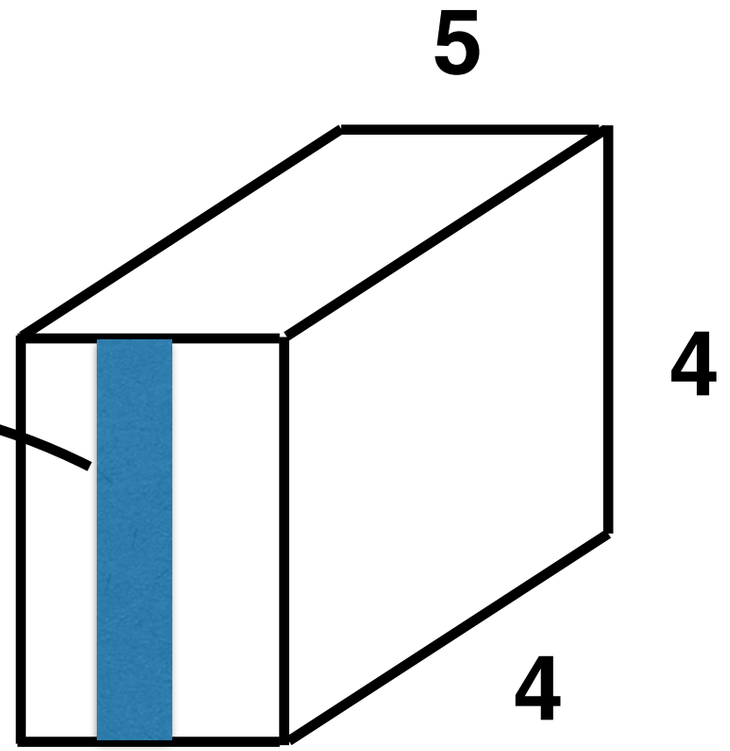
Single depth slice



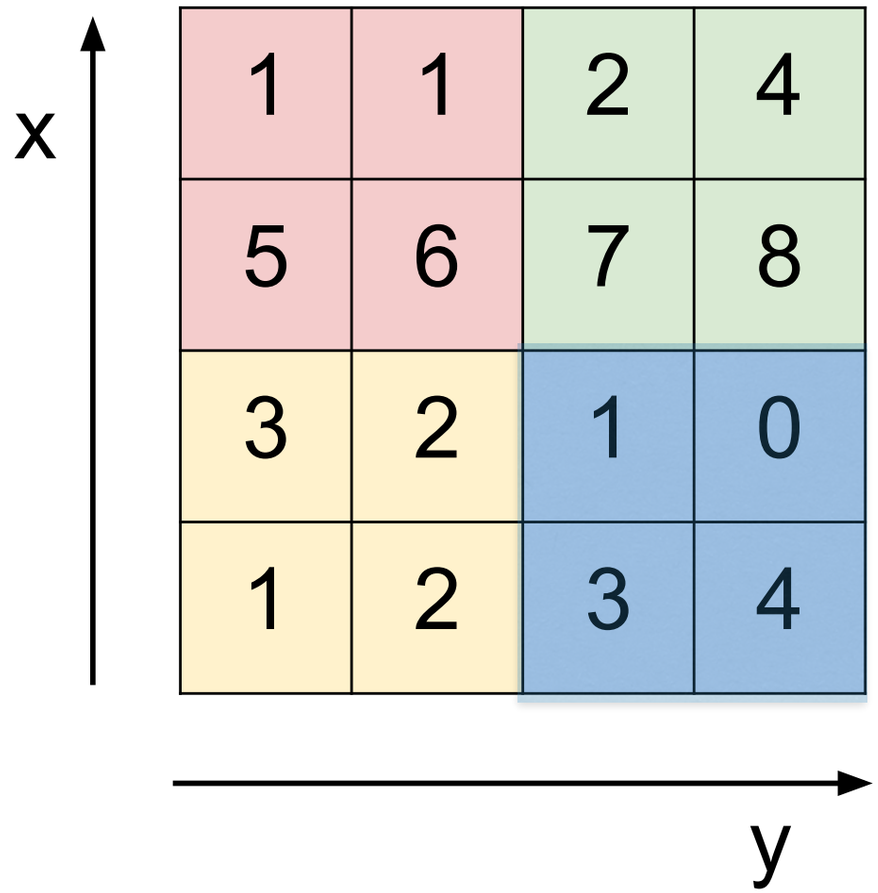
max pool with 2x2 filters
and stride 2

6	8
3	

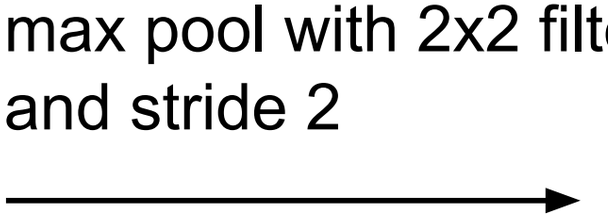
Max Pooling

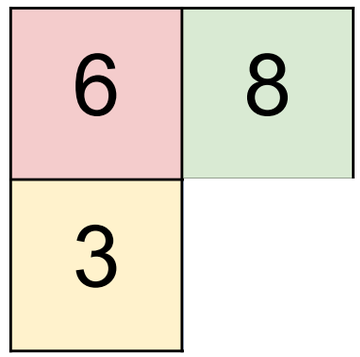


Single depth slice

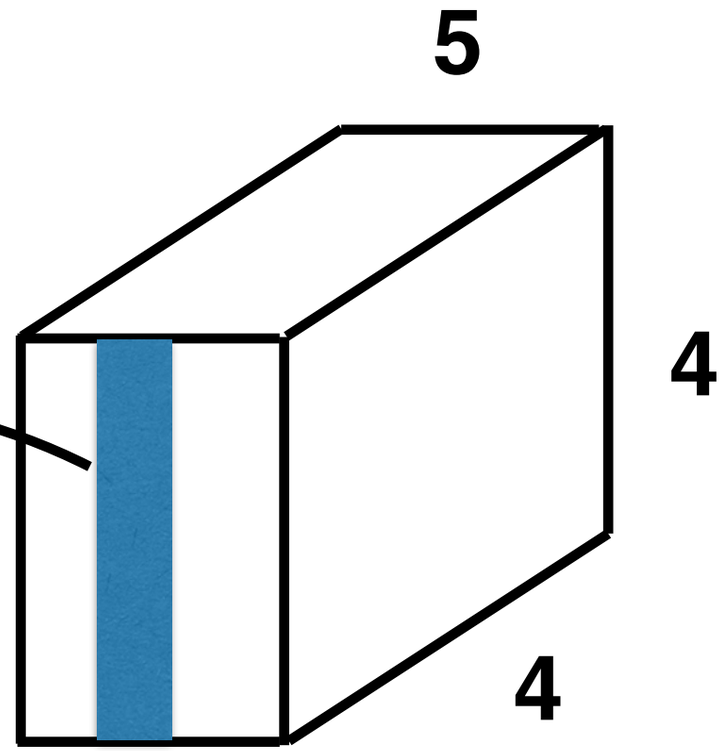


max pool with 2x2 filters
and stride 2

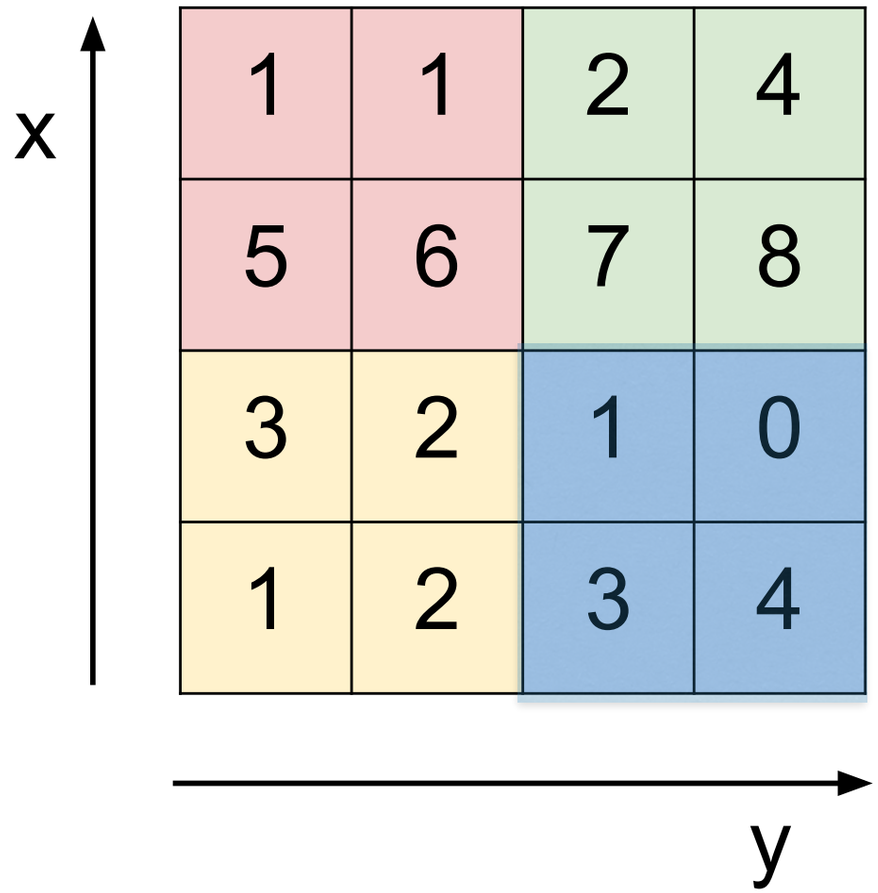




Max Pooling



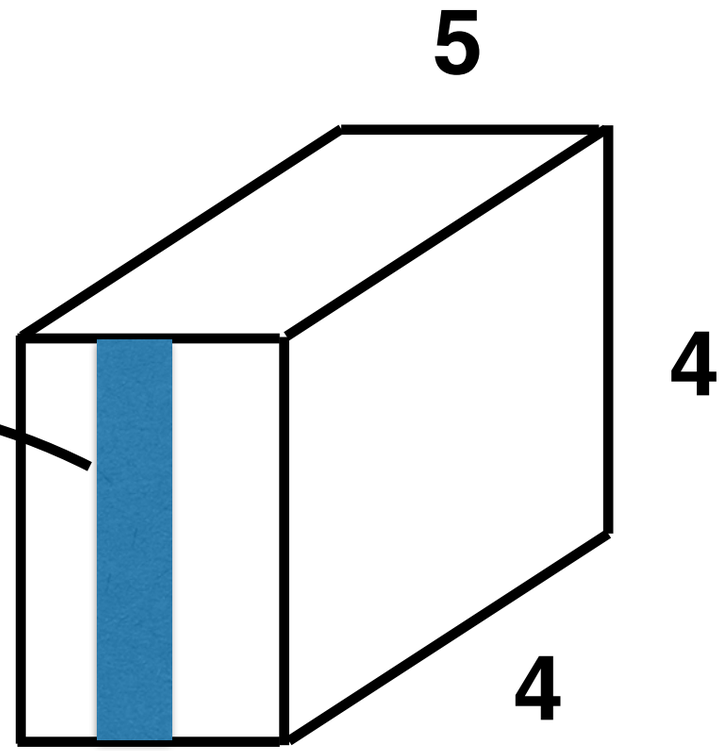
Single depth slice



max pool with 2x2 filters
and stride 2

6	8
3	4

Max Pooling



Single depth slice

x

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

max pool with 2x2 filters
and stride 2

6	8
3	4

Input: 4x4x5 Output: 2x2x5

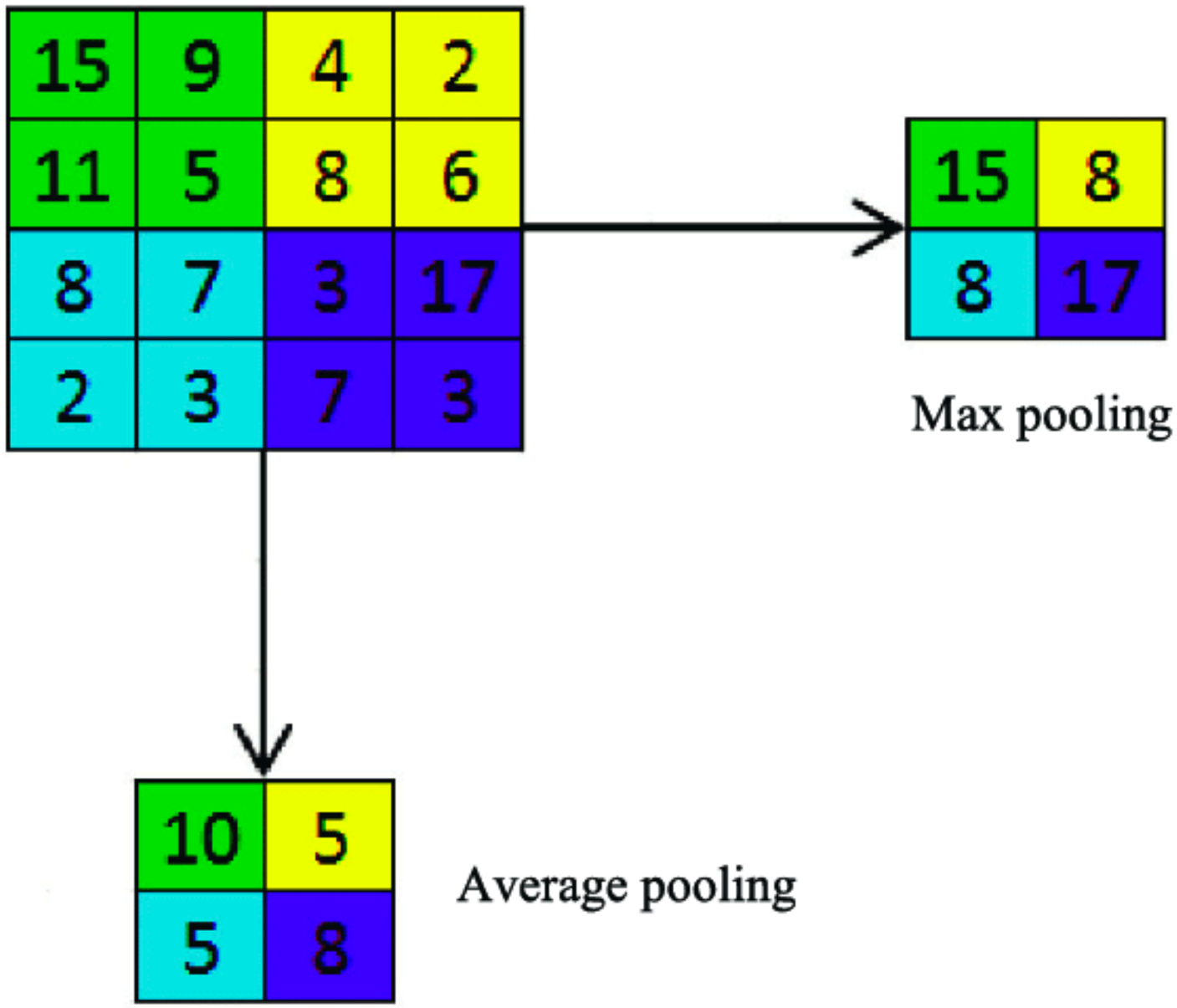
Max Pooling

Idea: We trade some spacial resolution for efficiency.

For image recognition, it is often not necessary to know the exact position of an object.

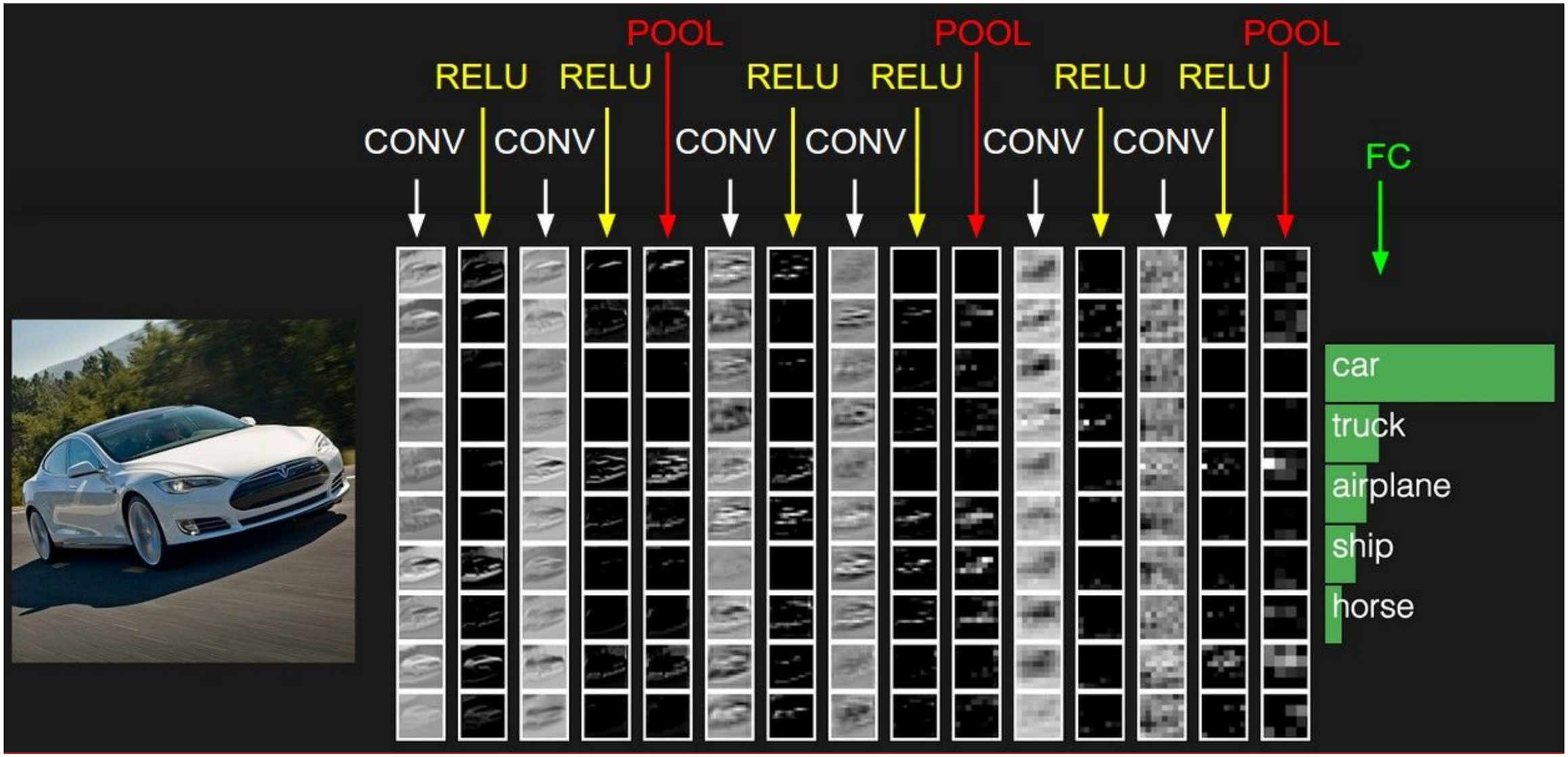
In fact, pooling can improve results by increasing spatial tolerance.

Average Pooling

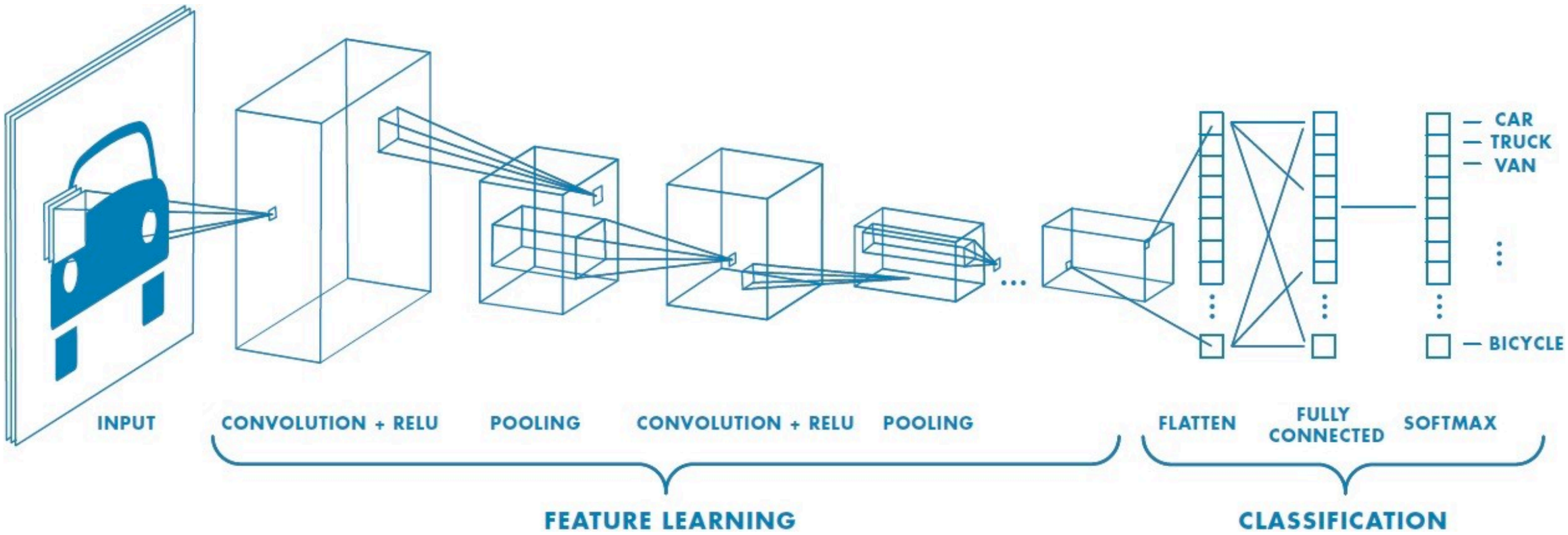


Average pooling seems to not work as well as max pooling.

Overall Architecture



Final Classification



Training a CNN

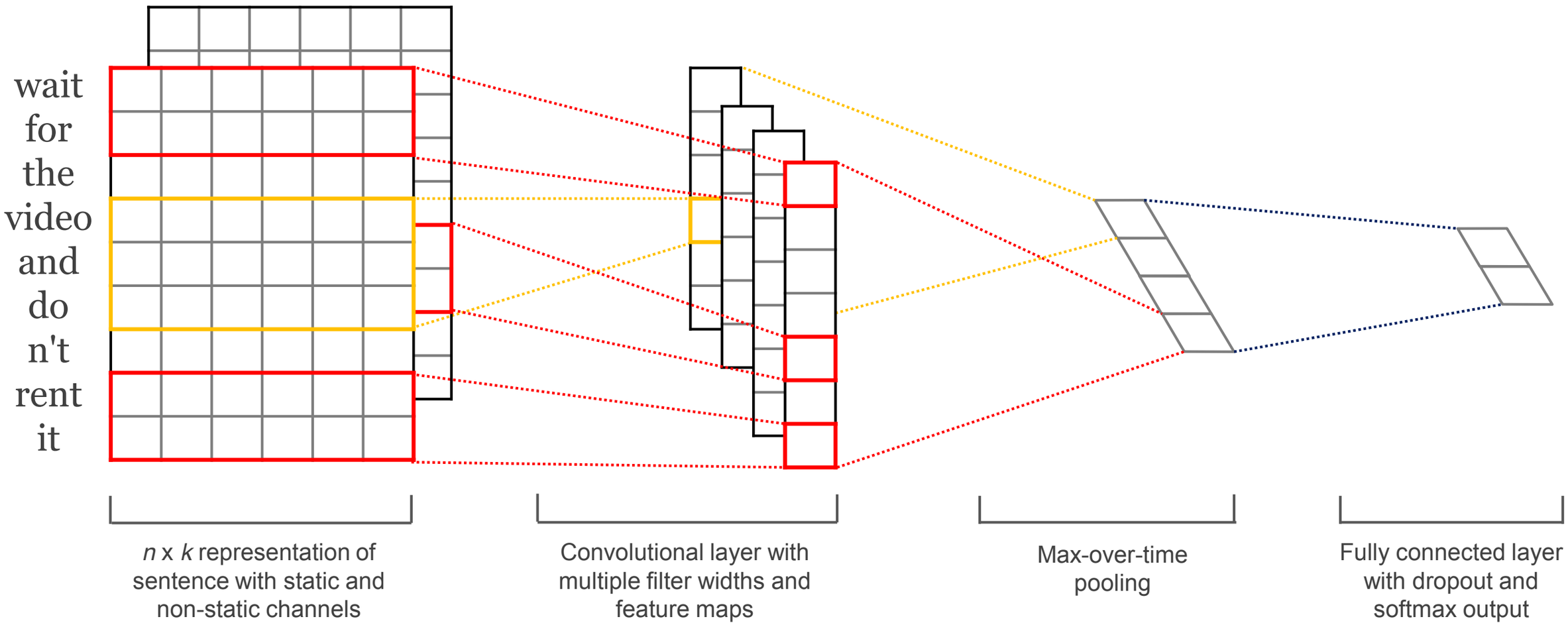
We train CNN's using backprop like any other NN.

Convolutional filters are initialized randomly!

Because of the large number of dot products that needs to be computed, a GPU is typically used.

CNN's also have a fairly large memory footprint because the intermediate results of each convolution operation need to be stored for backprop.

CNNs for NLP



Yoon Kim: Convolutional Neural Networks for Sentence Classification EMNLP 2014