Statistical View on Machine Learning

CNN - Convolutional Neural Networks

Marta Kałużna
Szymon Czop

Plan of our presentation

1. Why using CNN is better than normal Neural Networks?
2. CNN overview. Convolution, ReLU and pooling - what do they mean?
3. Building and training the model - Jupyter coding.

Before we begin - some facts about CNN

Before we begin - some facts about CNN

- One of the variants of neural networks.

Before we begin - some facts about CNN

- One of the variants of neural networks.
- Used heavily in the field of Computer Vision.

Before we begin - some facts about CNN

- One of the variants of neural networks.
- Used heavily in the field of Computer Vision.
- Derives its name from the type of hidden layers.

Why using Convolutional Neural Networks is better than normal Neural Networks?

1. Why using CNN is better than normal NN?

- **Reason 1:** Images are Big

1. Why using CNN is better than normal NN?

- **Reason 1:** Images are Big

The nice thing about images: pixels are most useful in the context of their neighbors.

1. Why using CNN is better than normal NN?

- **Reason 1:** Images are Big

The nice thing about images: pixels are most useful in the context of their neighbors.

- **Reason 2:** Positions can change

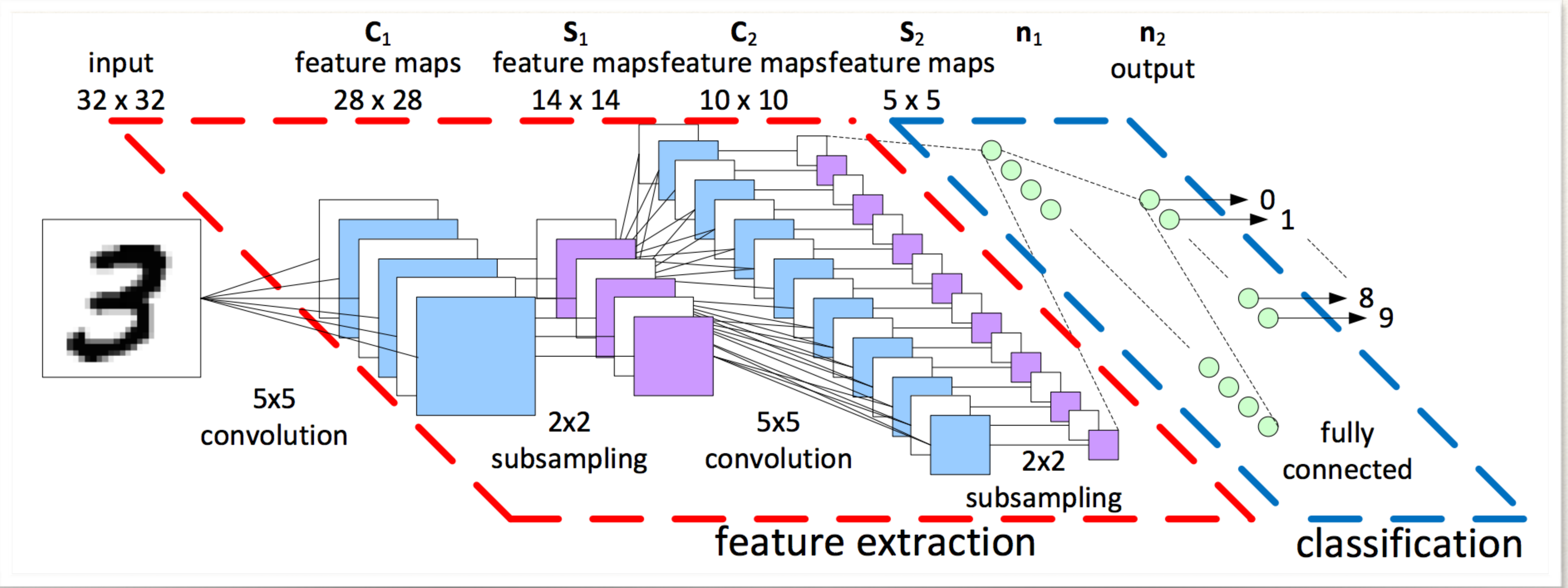
1. Why using CNN is better than normal NN?

- **Reason 1:** Images are Big

The nice thing about images: <u>pixels are most useful in the context of their neighbors</u>.

- **Reason 2:** Positions can change

We want to be able to detect a thing <u>regardless of where it appears in the image.</u>



2. CNN Overview

Convolutional Neural Networks architecture

Differences between CNN and NN

Differences between CNN and NN

- The layers are organized in <u>3 dimensions</u>: width, height, depth.

Differences between CNN and NN

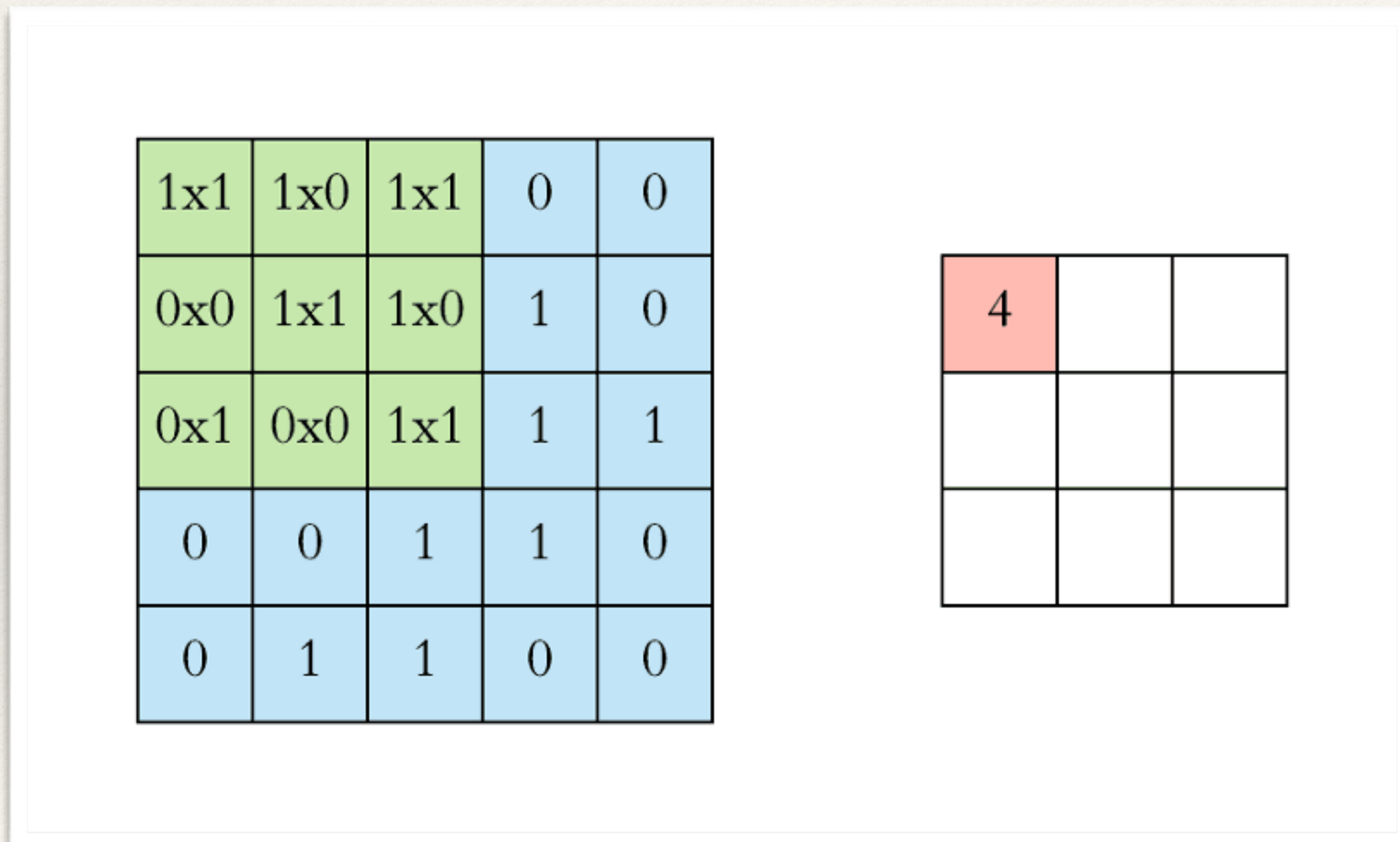
- The layers are organized in <u>3 dimensions</u>: width, height, depth.
- The neurons in one layer <u>do not connect to all the neurons</u> in the next layer (only to a small region of it).

Differences between CNN and NN

- The layers are organized in <u>3 dimensions</u>: width, height, depth.
- The neurons in one layer <u>do not connect to all the neurons</u> in the next layer (only to a small region of it).
- The final output will be reduced to a <u>single vector of probability scores</u> (organized along the depth dimension).

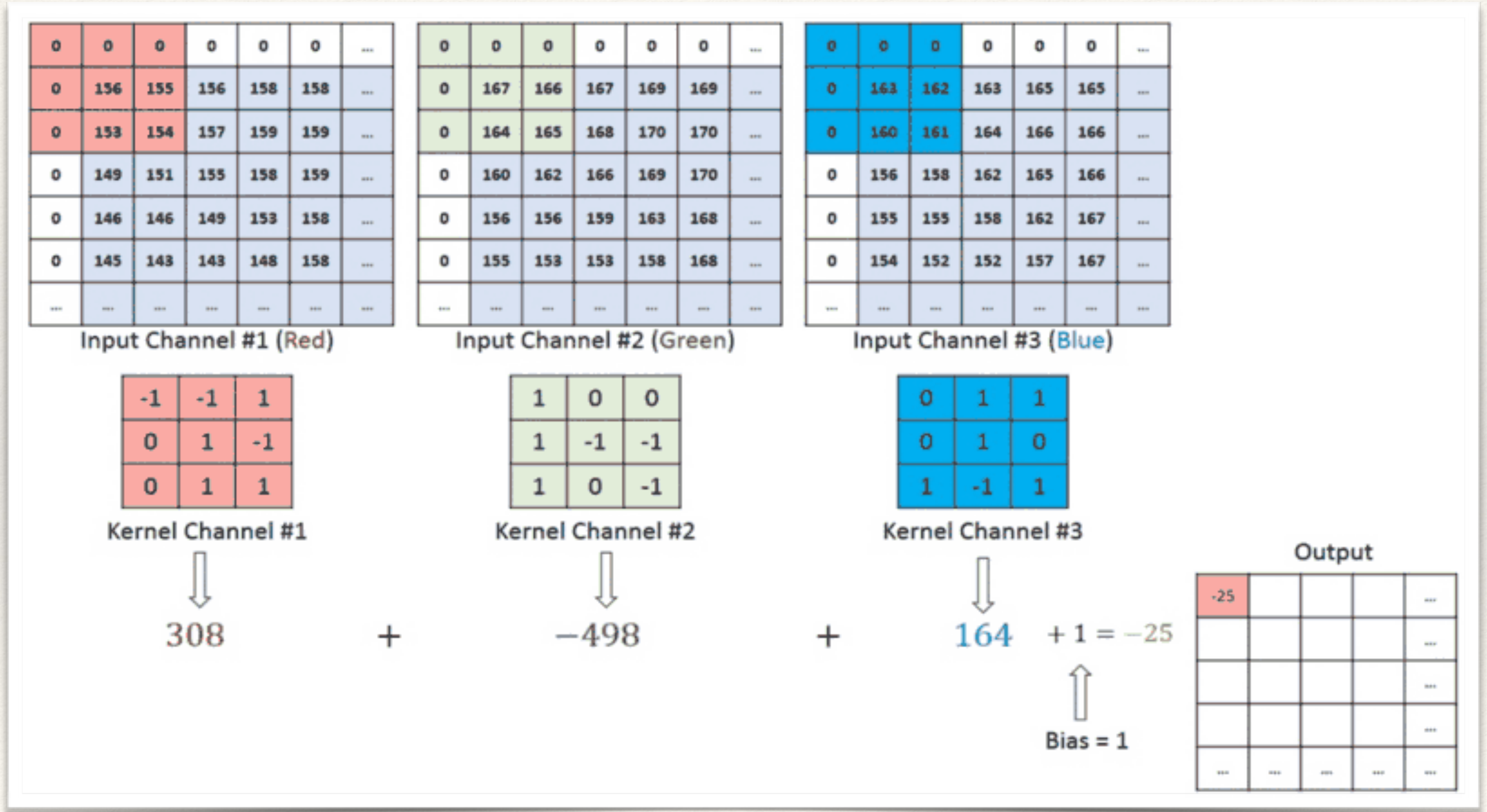
Convolution, padding, pooling

Convolution

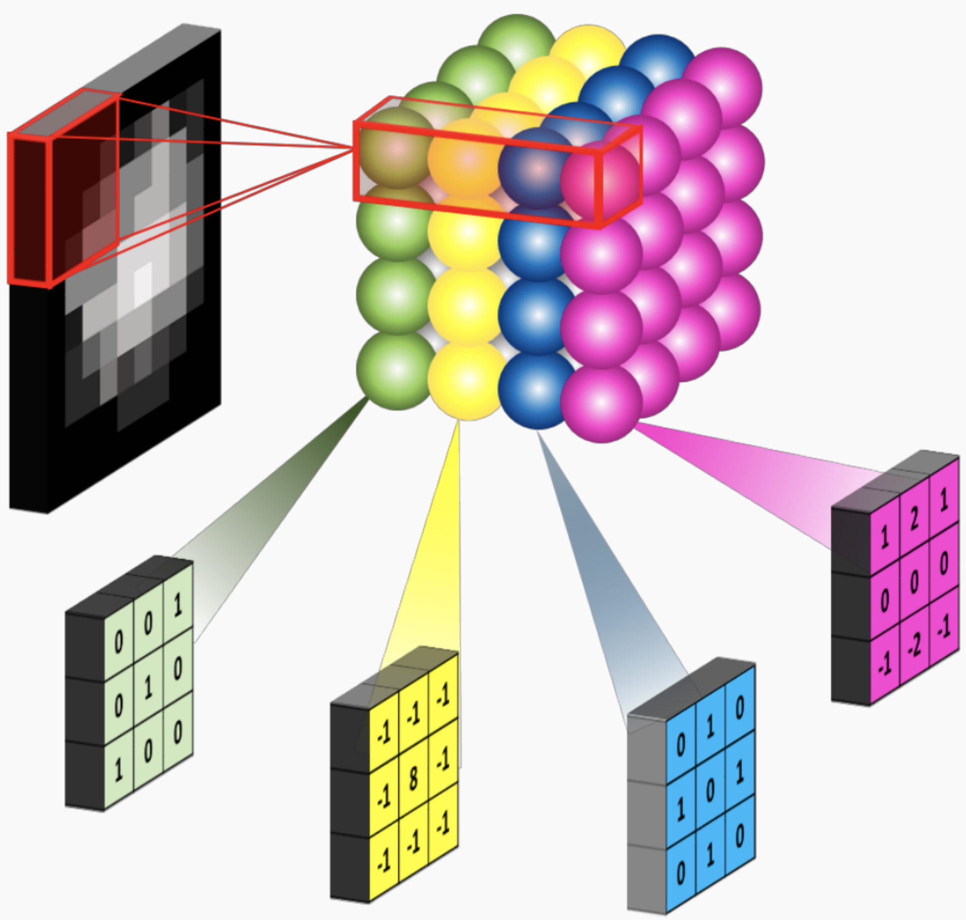


1. Overlaying the filter on top of the image at some location.
2. Performing **element-wise multiplication** between the values in the filter and their corresponding values in the image.
3. Summing up all the element-wise products. This sum is the output value for the **destination pixel** in the output image.
4. Repeating for all locations.

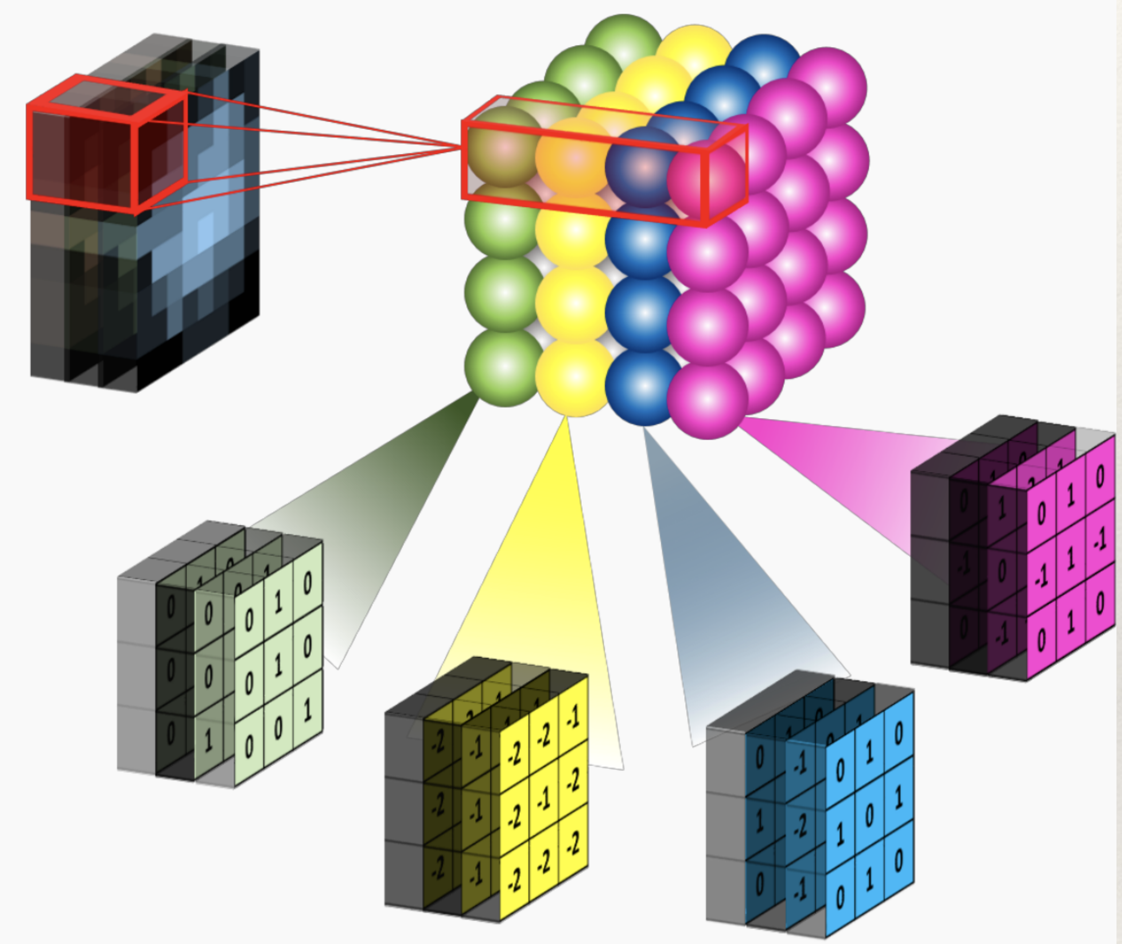
Convolution of images with multiple channels



How do we connect our filters together?

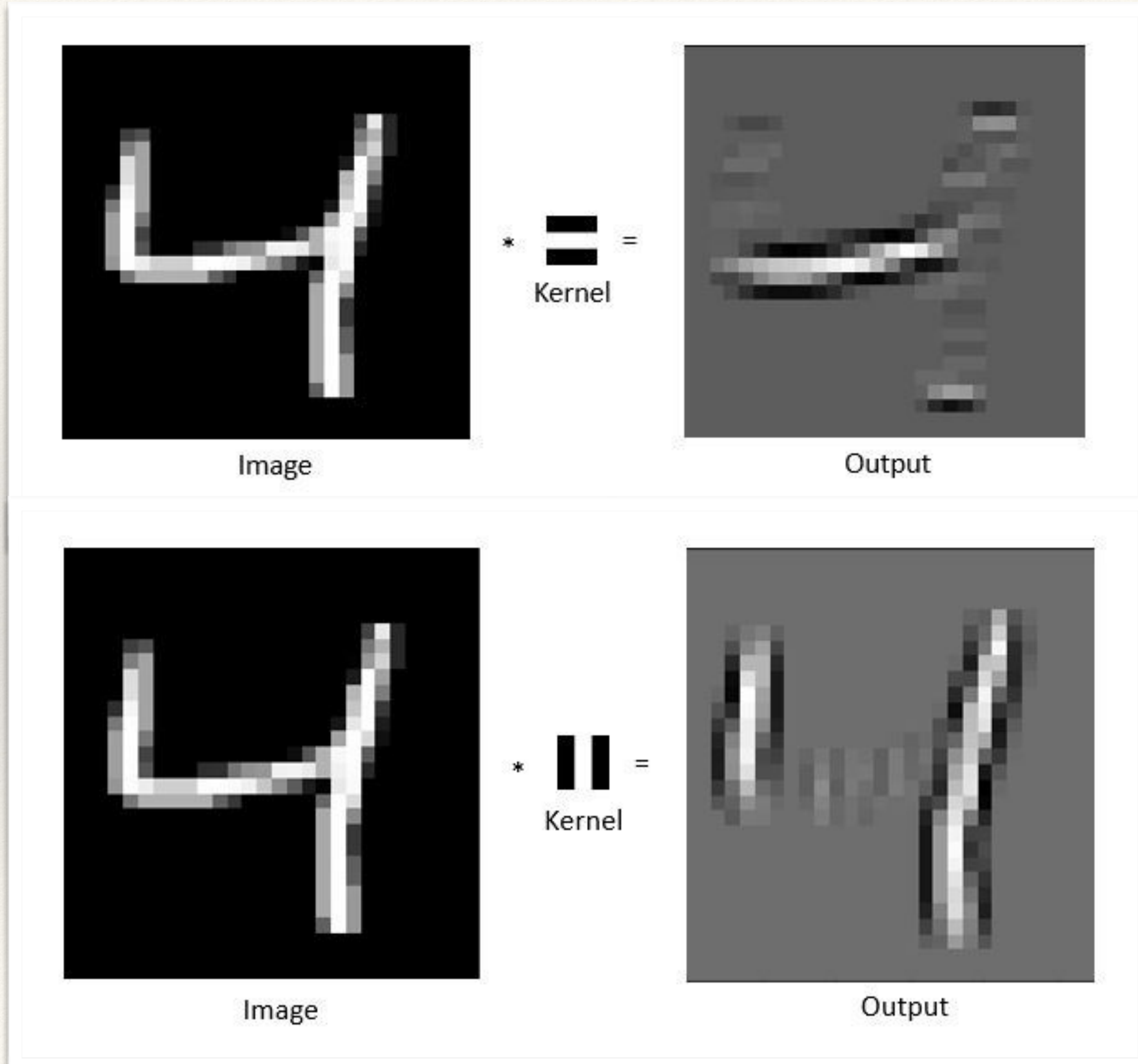


Convolutional layer with four 3x3 filters on a black and white image (just one channel)

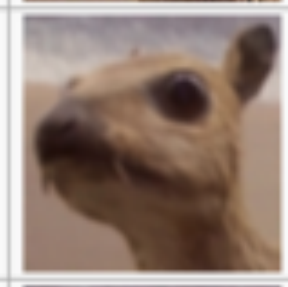


Convolutional layer with four 3x3 filters on an RGB image. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

Horizontal vs vertical filter

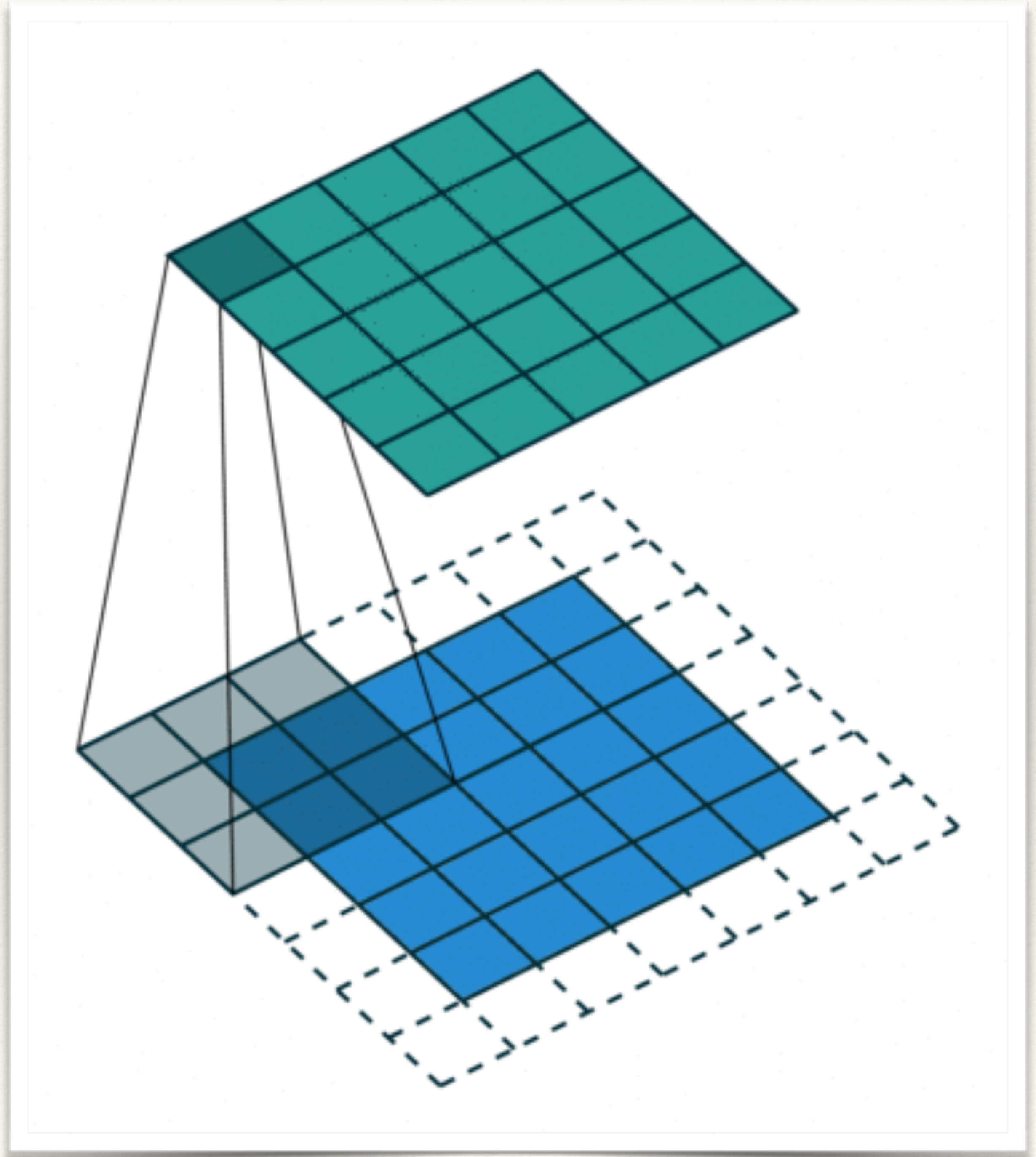


Various convolution image after applying different types of filters

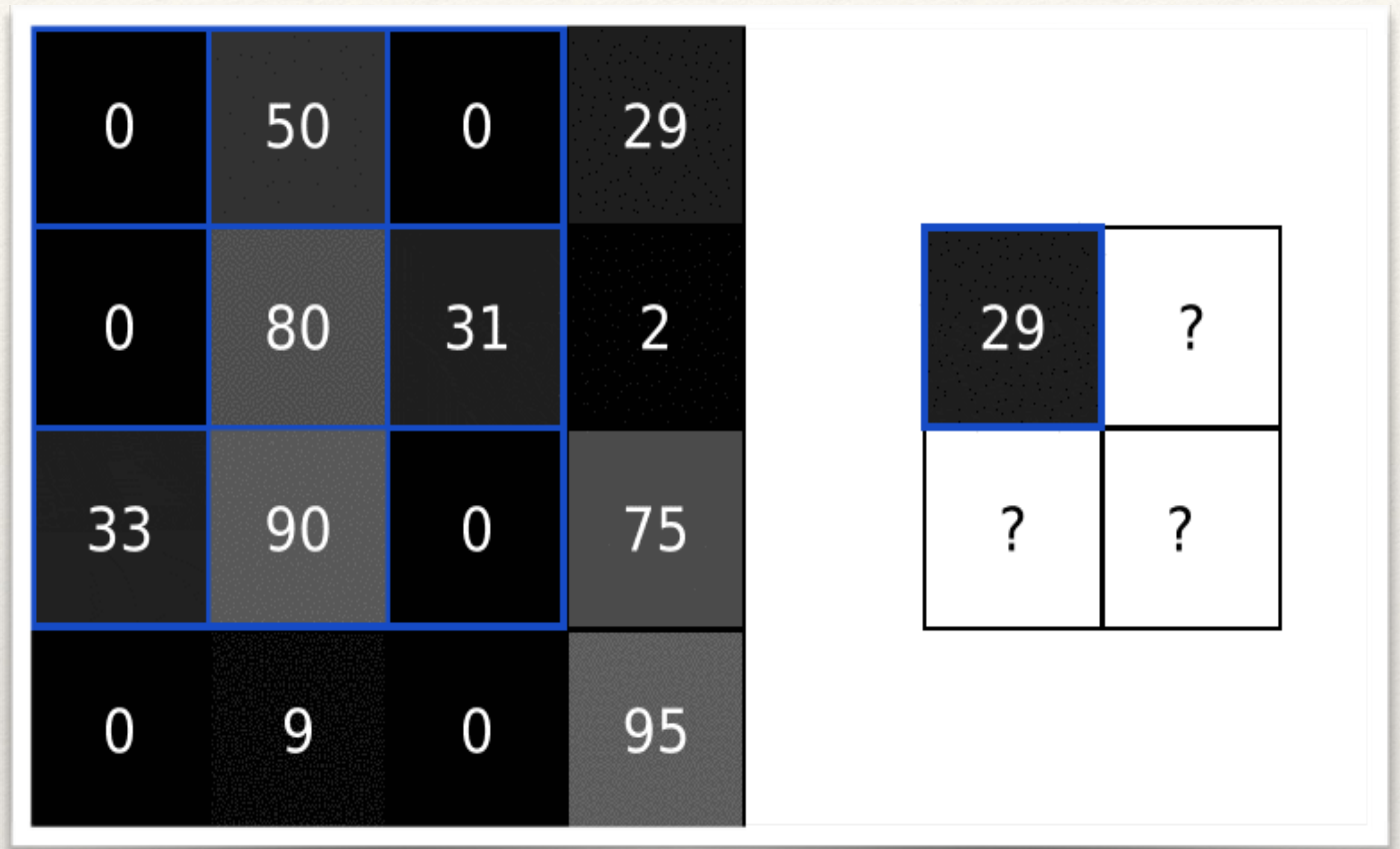
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Padding

- **Same Padding** - the output image has the same dimensions as the input image (to achieve it, we pad the input image with zeros).



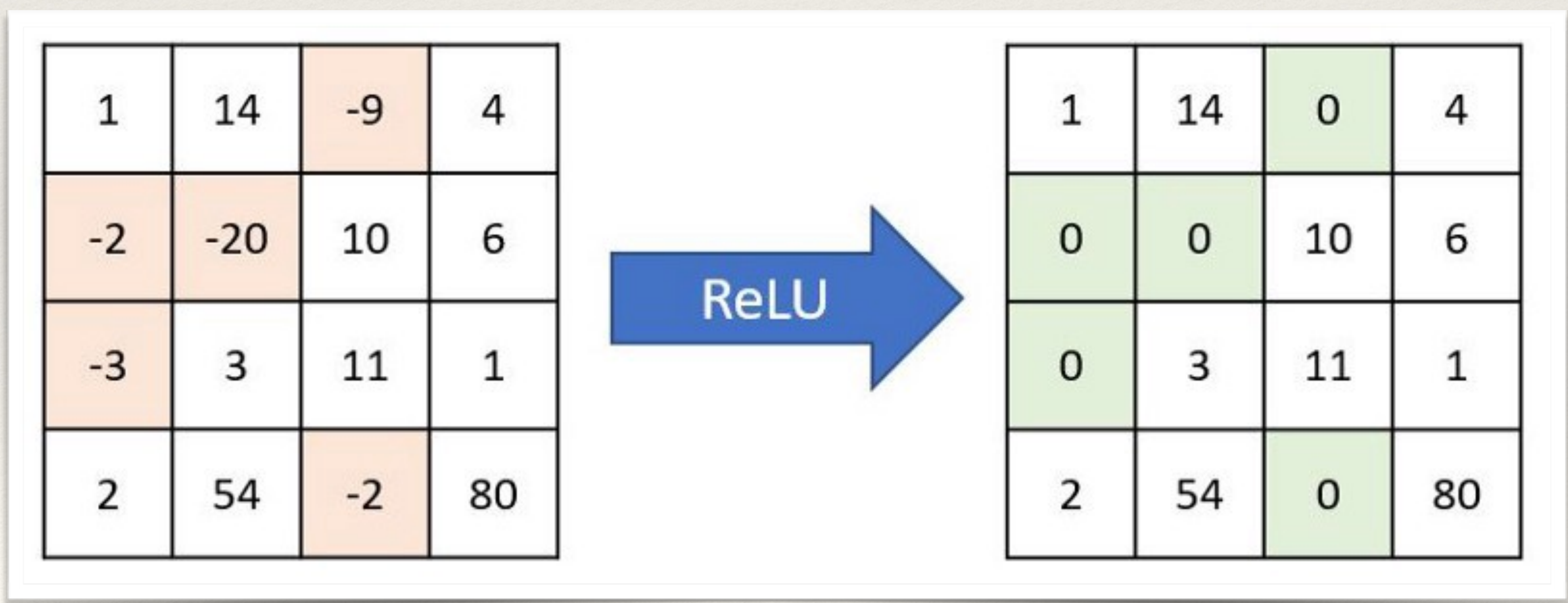
Padding



- **Valid Padding** - the output image is reduced in the dimensionality as compared to the input

Non Linearity (ReLU)

- **ReLU** = Rectified Linear Unit
- The output is $f(x) = \max(0,x)$.
- Converts all of the negative values to 0 and keeps the positive values the same.

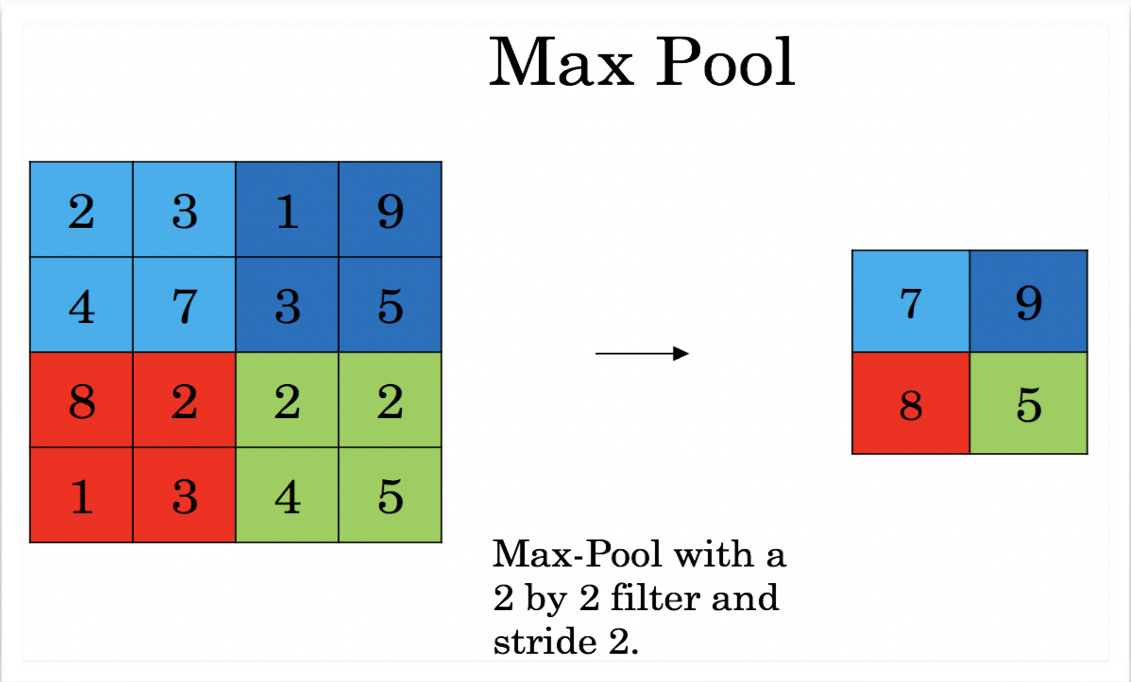


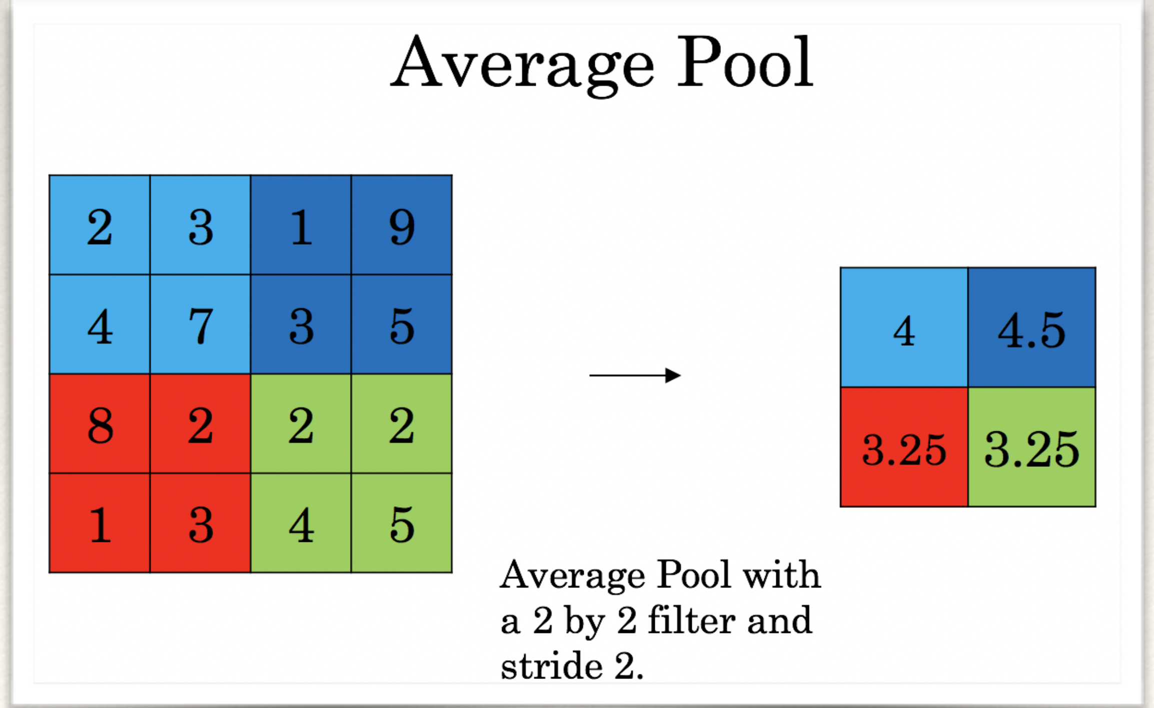
Pooling

- Reduces the number of parameters when the images are too large.
- Why? To decrease the computational power required to process the data through dimensionality reduction.
- Shortens the training time and controls over-fitting.

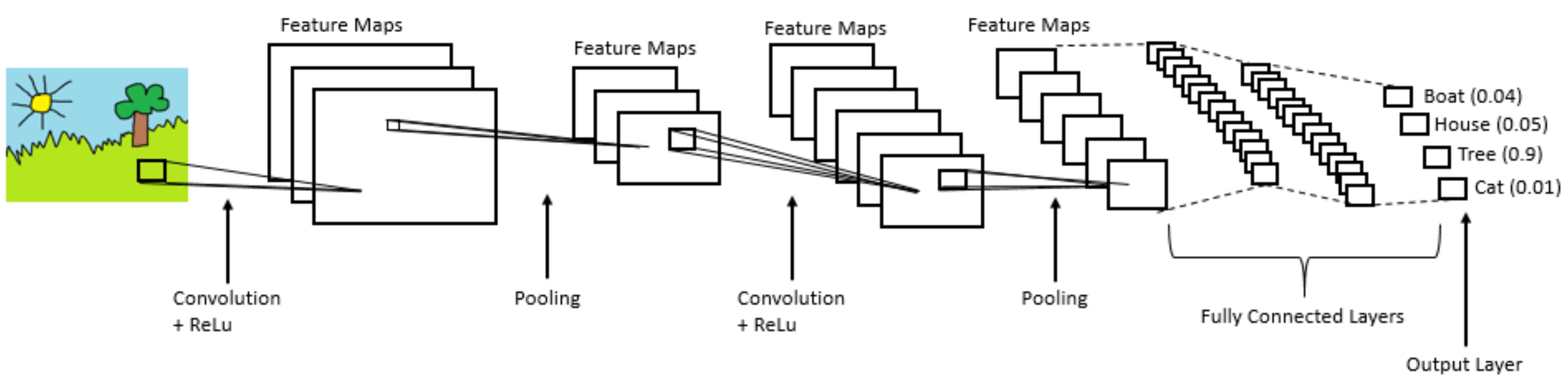
Pooling

- **Max Pooling** - returns the maximum value from the portion of the image covered by the Kernel.
- **Average Pooling** - returns the average of all the values from the portion of the image covered by the Kernel.



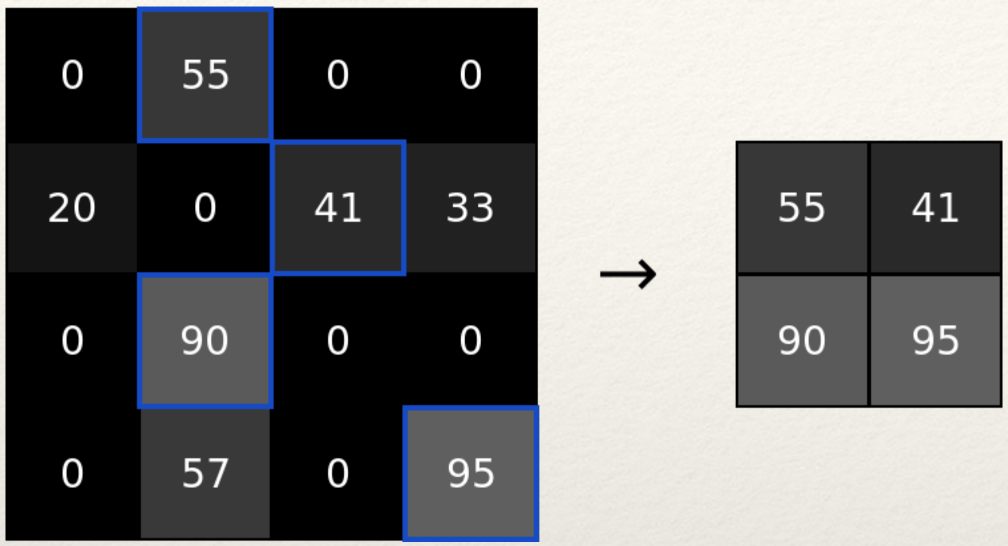


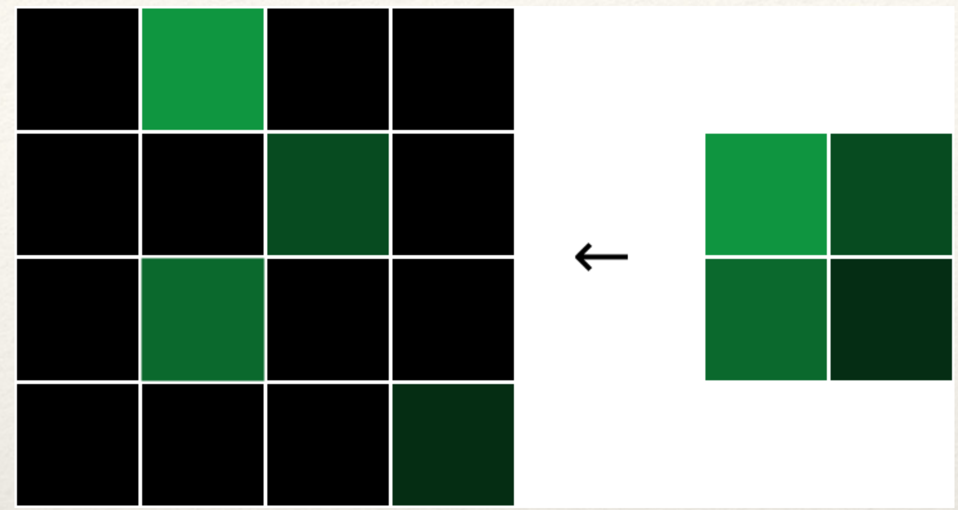
Complete CNN architecture



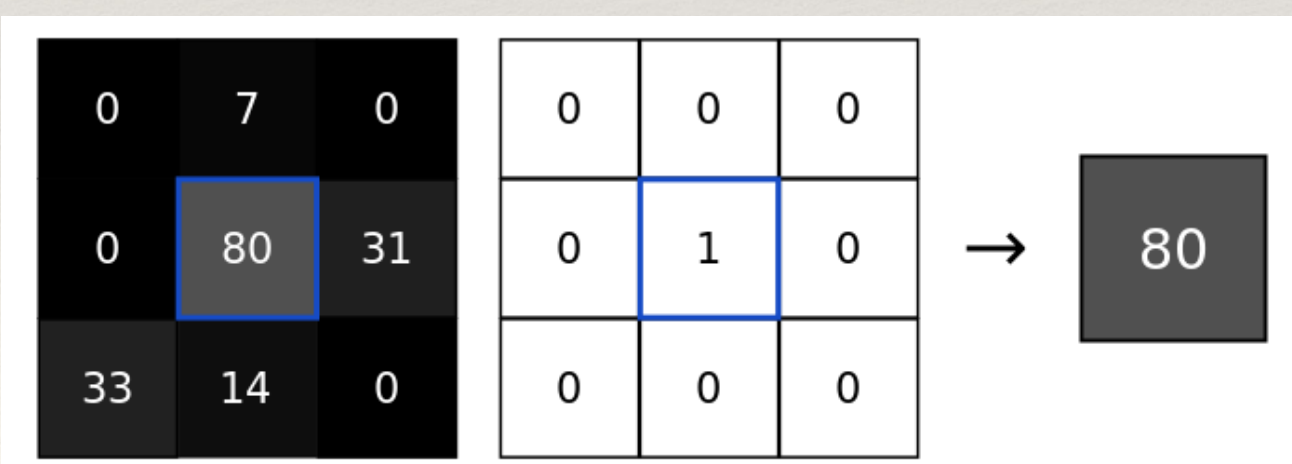
Backpropagation

Max Pooling





Conv layer



$$\begin{aligned}\text{out(i, j)} &= \text{convolve(image, filter)} \\ &= \sum_{x=0}^{3}\sum_{y=0}^{3} \text{image}(i+x, j+y) * \text{filter}(x, y)\end{aligned}$$

$$\frac{\partial \text{out}(i,j)}{\partial \text{filter}(x,y)} = \text{image}(i+x, j+y)$$

We can put it all together to find the loss gradient for specific filter weights:

$$\frac{\partial L}{\partial \text{filter}(x,y)} = \sum_i \sum_j \frac{\partial L}{\partial \text{out}(i,j)} * \frac{\partial \text{out}(i,j)}{\partial \text{filter}(x,y)}$$

Building and training the model

Thank you for your attention!