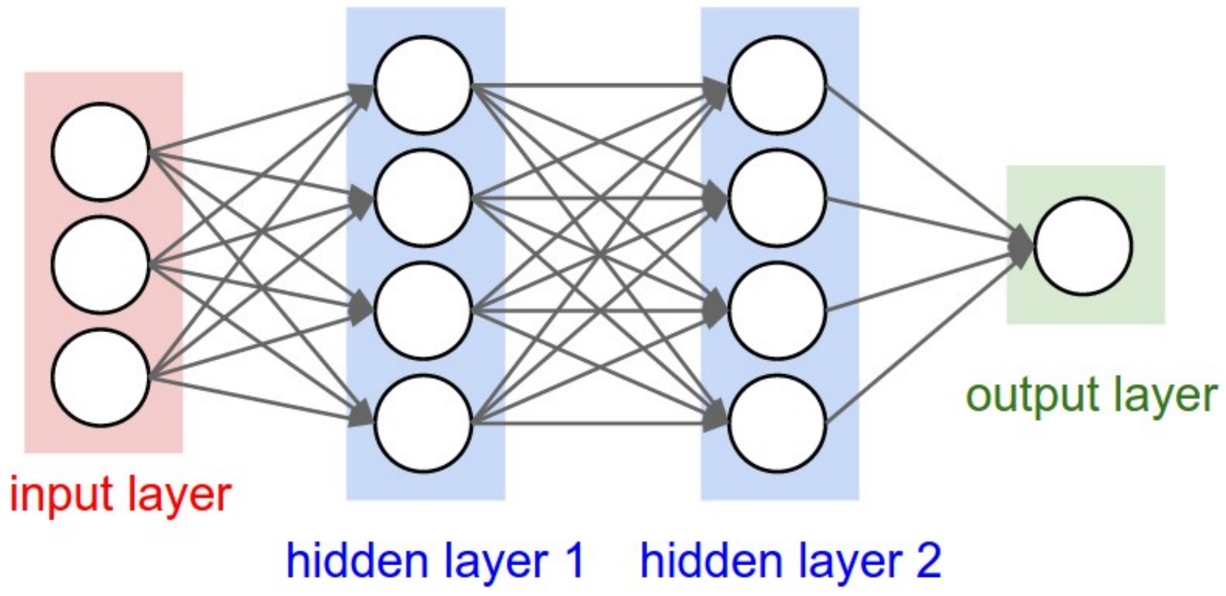
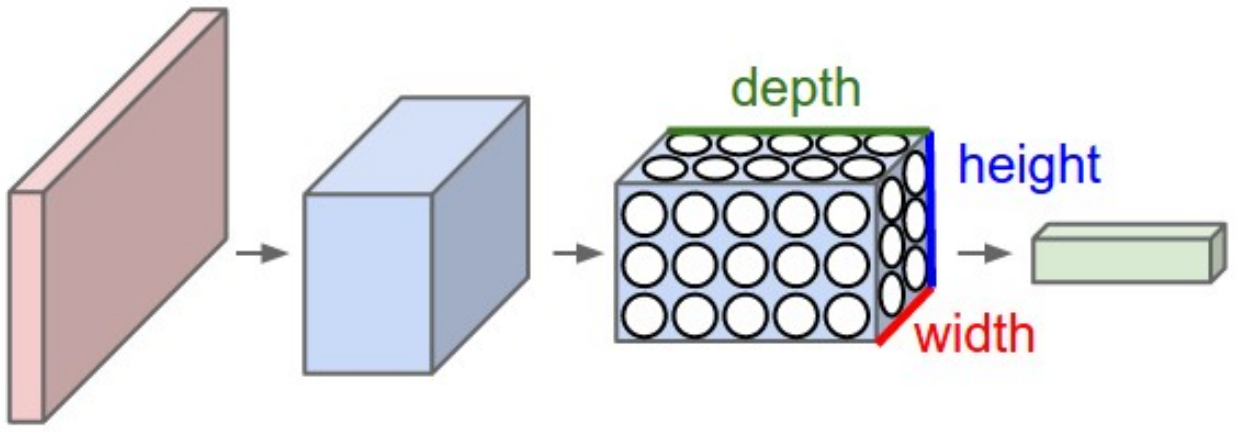
CNN

- CNN (ConvNet) are very similar to ordinary Neural Networks (feed-forward): they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. And have a loss function on the last (fully-connected) layer.
- But, these architectures make an explicit assumption that the inputs are images, which allows them to encode certain properties into the architecture.
- These make the forward function more efficient to implement and vastly reduce the amount of parameters as compared to the feed-forward networks.
- Unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth.

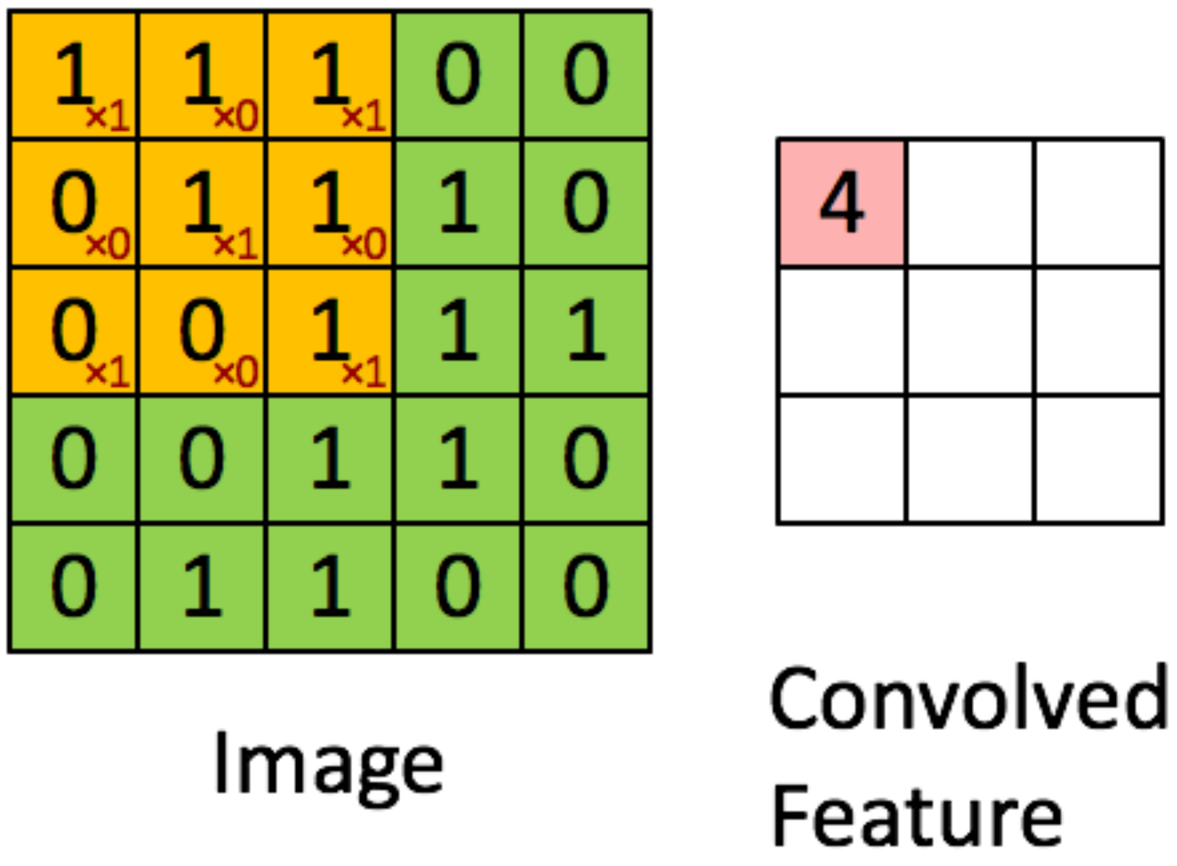


Feed-forward Neural Network



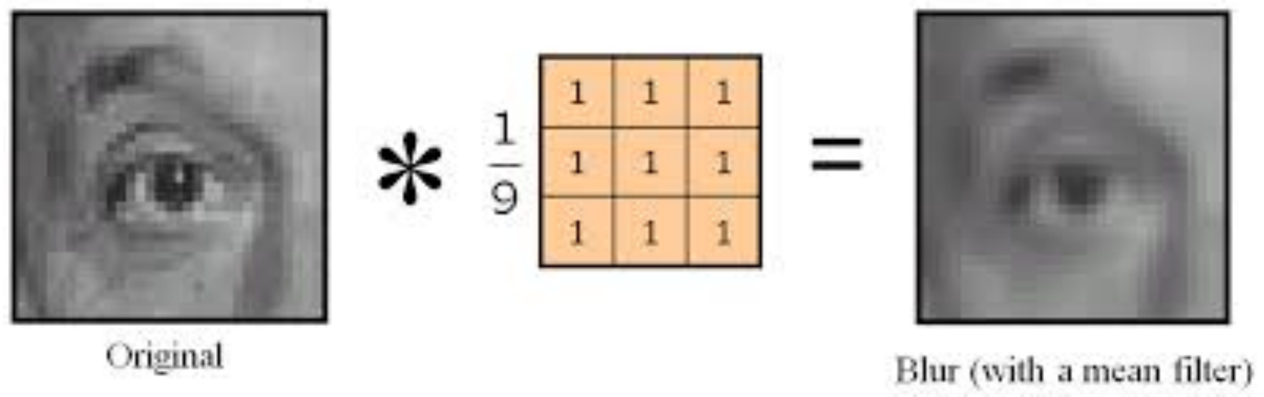
Convolutional Neural Network

- A CNN is a neural network that typically contains several types of layers, one of which is a **convolutional layer**.



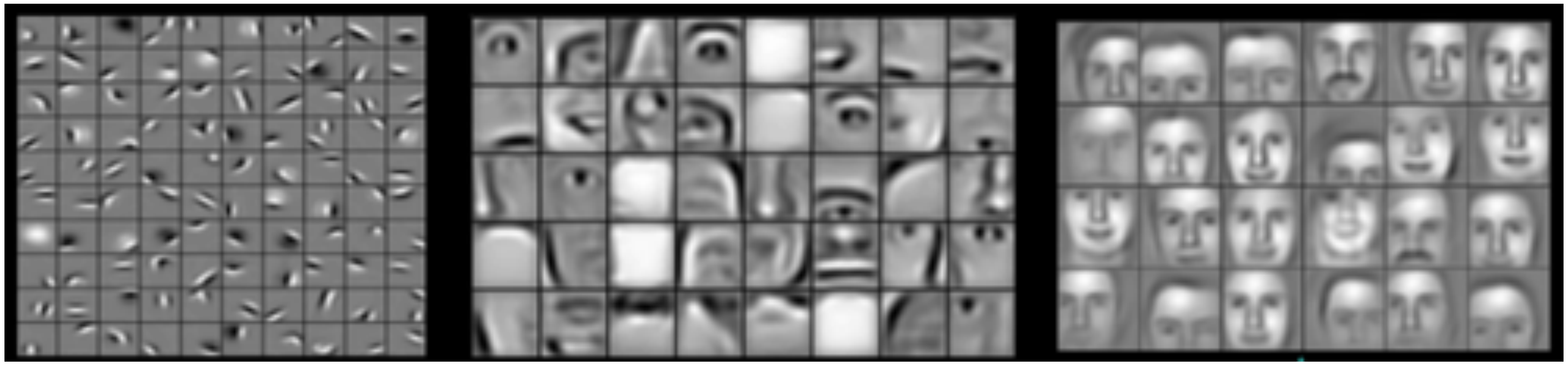
Convolutional Layer

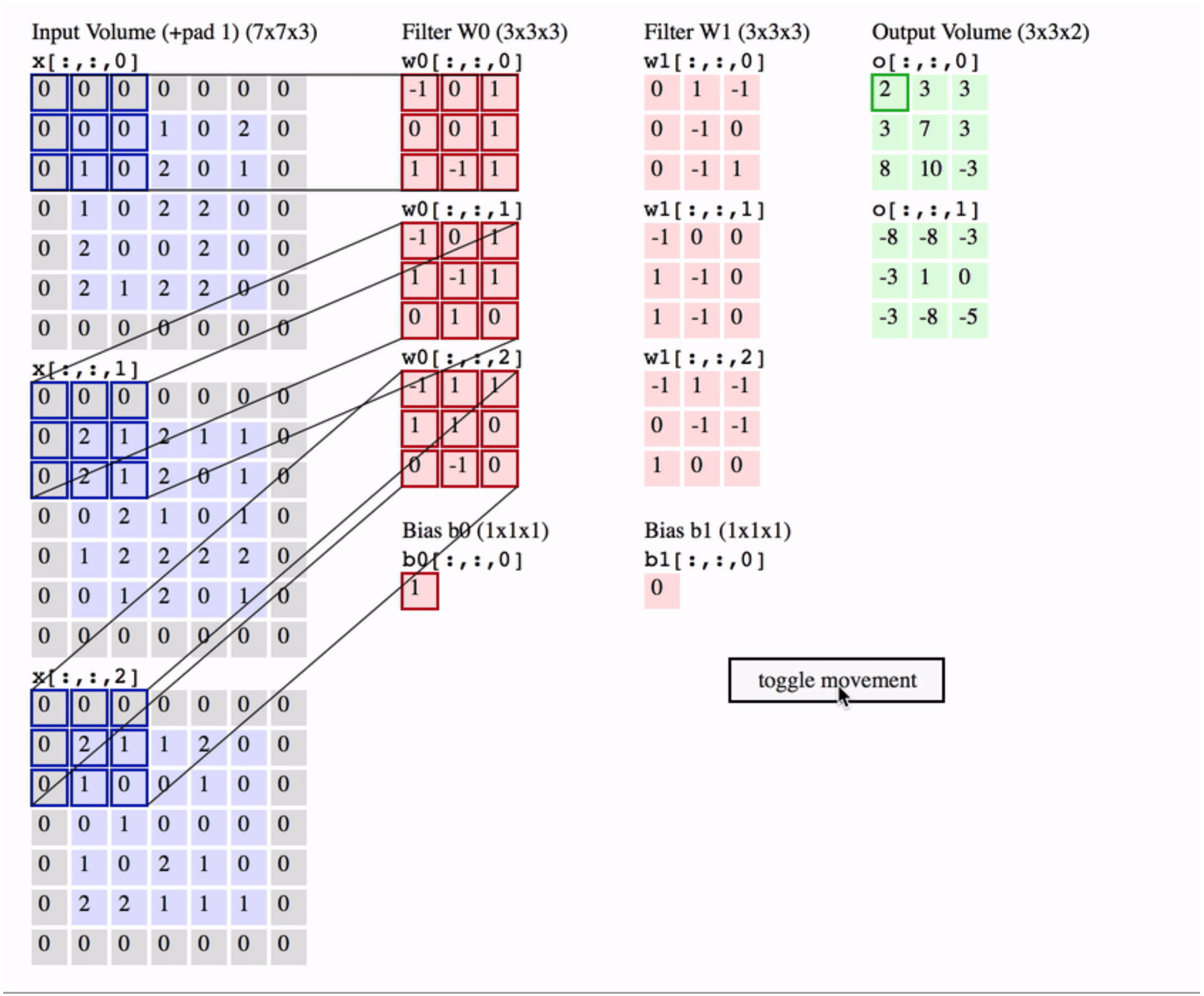
- Convolution is a pretty common operation in image processing tasks.
- It works by convolving / sliding a filter (matrix) throughout the source image and performing element wise dot product to obtain a resultant matrix.
- The filter works as a feature detector and tries to find a pattern in a source image.



A simple filter to blur the source image.

- The “filter” that moves over the image is called a **kernel**. Kernels are typically square and 3x3 is a fairly common kernel size. The distance the window moves each time is called the **stride**.
- Images are sometimes **padded** with zeros around the perimeter when performing convolutions, which dampens the value of the convolutions around the edges of the image (the idea being typically the center of photos matter more).
- The goal of a convolutional layer is **filtering.** As we move over an image we effective check for patterns in that section of the image.
- When training an image, these filter weights change, and so when it is time to evaluate an image, these weights return high values if it thinks it is seeing a pattern it has seen before.
- The combinations of high weights from various filters let the network predict the content of an image.


Features detected by different convolution layers in face detection model



Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	1	0	2	0	1	0
0	1	0	2	2	0	0
0	2	0	0	2	0	0
0	2	1	2	2	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	2	1	2	1	1	0
0	2	1	2	0	1	0
0	0	2	1	0	1	0
0	1	2	2	2	2	0
0	0	1	2	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	2	1	1	2	0	0
0	1	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	2	1	0	0
0	2	2	1	1	1	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	1
0	0	1
1	-1	1

w0[:,:,1]

-1	0	1
1	-1	1
0	1	0

w0[:,:,2]

-1	1	1
1	1	0
0	-1	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

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

w1[:,:,1]

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

w1[:,:,2]

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

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

2	3	3
3	7	3
8	10	-3

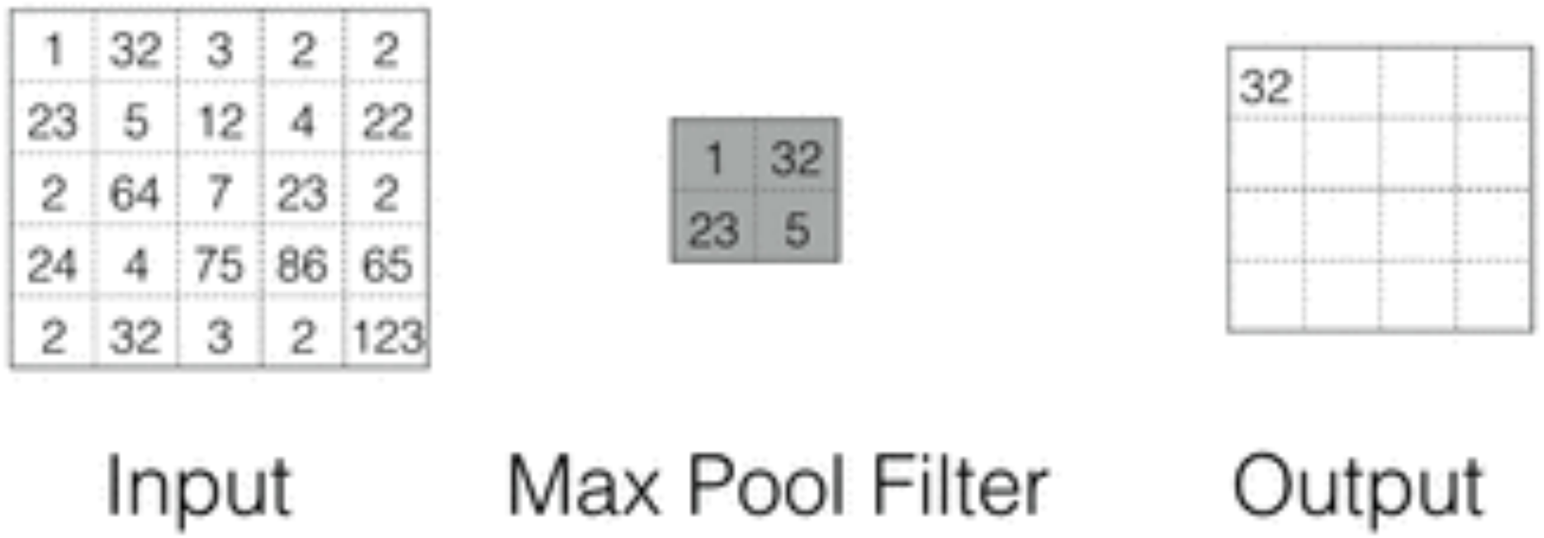
o[:,:,1]

-8	-8	-3
-3	1	0
-3	-8	-5

Convolution layer in action

Pooling Layer

- Pooling works very much like convolution where we take a **kernel** and move the kernel over the image, the only difference is the function that is applied to the kernel and the image window isn’t linear.
- **Max pooling** and **Average pooling** are the most common pooling functions. Max pooling takes the largest value from the window of the image currently covered by the kernel, while average pooling takes the average of all values in the window.

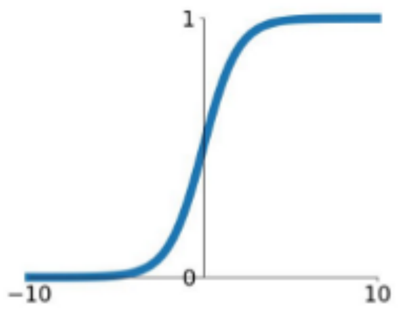


Activation Layer

- Activation layers work exactly as in other neural networks, a value is passed through a function that squashes the value into a range.
- It provides the non-linearity to the model.
- Some commonly used activation functions are given below.

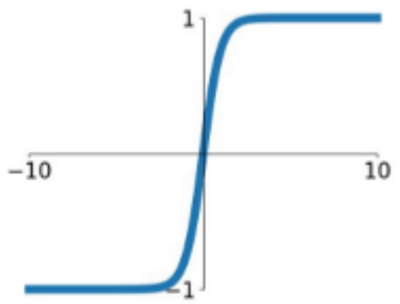
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



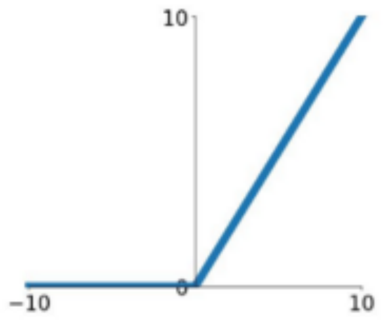
tanh

$\tanh(x)$



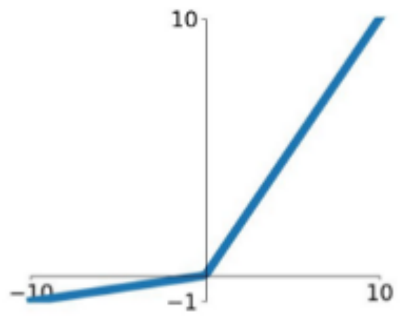
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

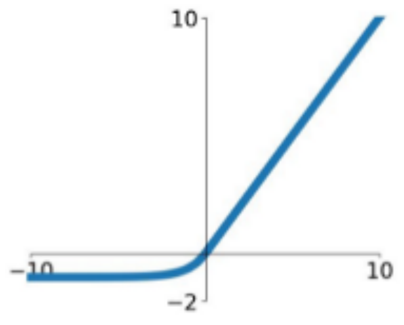


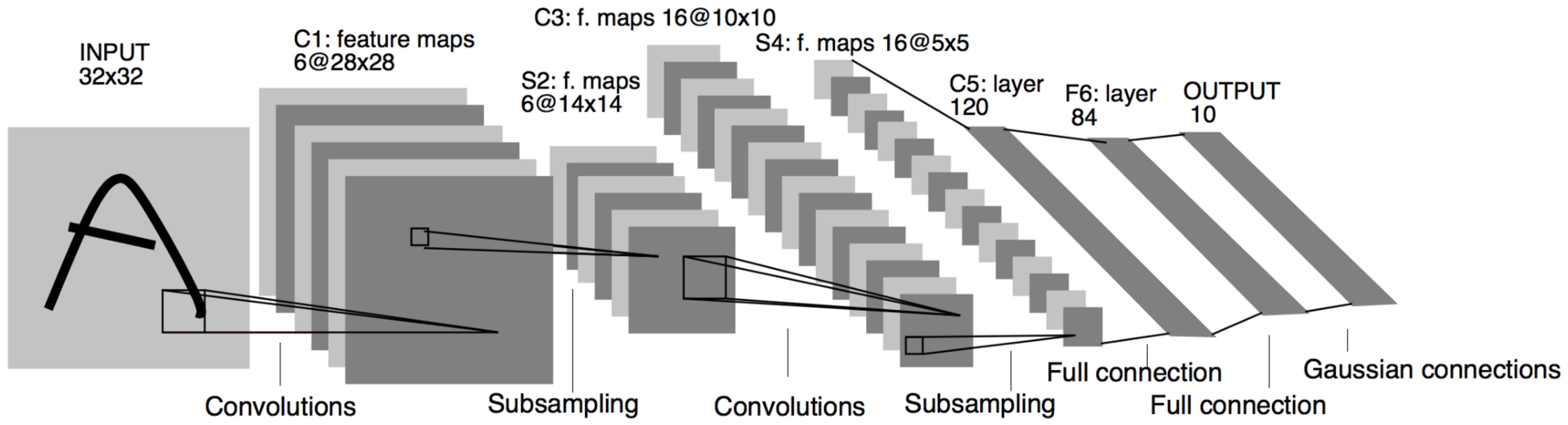
Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





A typical CNN architecture: LeNet

References

1. http://cs231n.github.io/convolutional-networks/
2. https://hackernoon.com/visualizing-parts-of-convolutional-neural-networks-using-keras-and-cats-5cc01b214e59