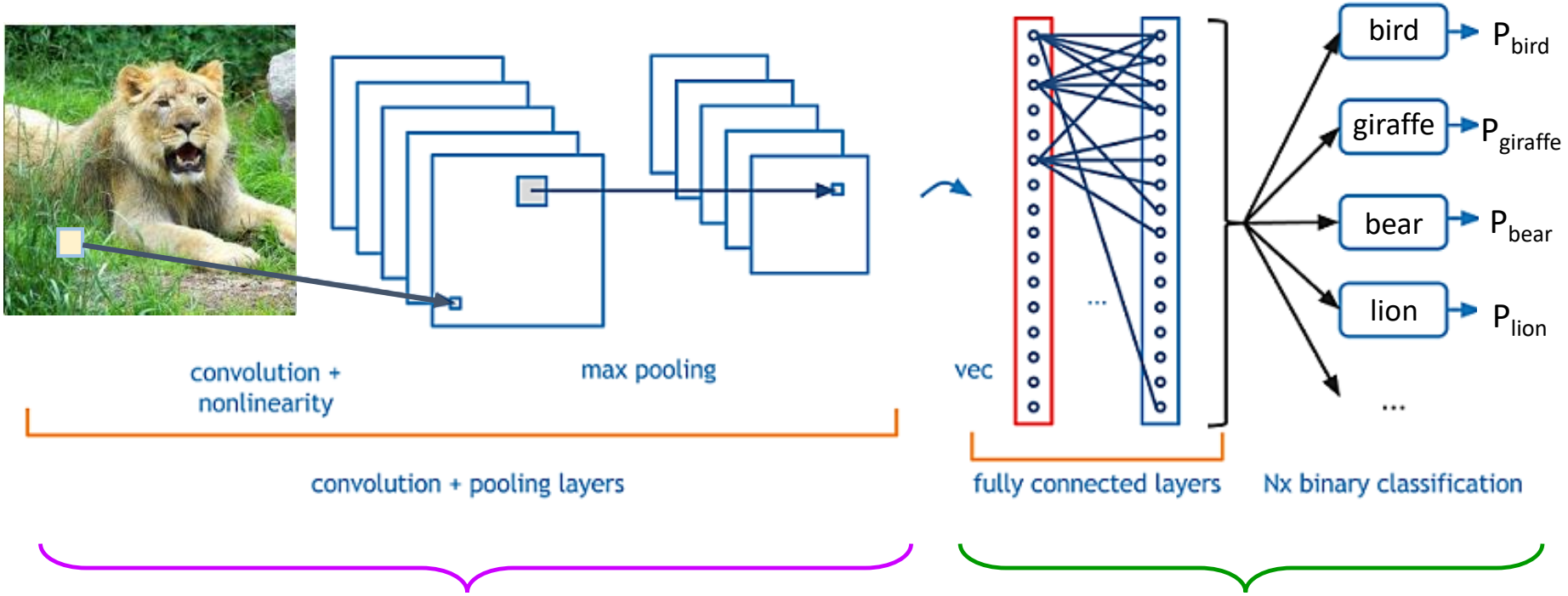
CNN

Convolutional Neural Network



Adaptation after: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

CNN (ConvNet) for deep learning- intro

• CNN – Convolutional Neural Network

- A class of deep neural networks, most commonly applied to analyzing visual imagery.
- Applications in: image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

  - Little pre-processing compared to other image classification algorithms.
  - The network **learns the filters** that in traditional algorithms were hand-engineered.
    - ✓ This independence from prior knowledge and human effort in feature design is a major advantage

CNN (ConvNet) for deep learning - intro

- The network employs a mathematical operation called **convolution**.
- CNN are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.
- CNN have learnable parameter like neural network (weights, biases, etc).

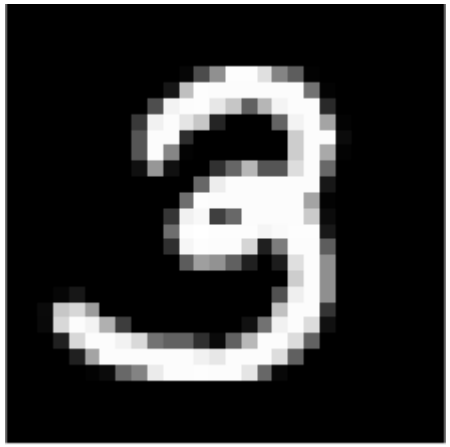
❖ Convolution

- a specialized kind of linear operation
- a mathematical operation on two functions (*f* and *g*) that produces a third function expressing how the shape of one (*f*) is modified by the other (*g*).
- defined as the integral of the product of the two functions after one *(g)* is reversed and shifted.

Why CNN?

If working with simple images, for example MINST data set

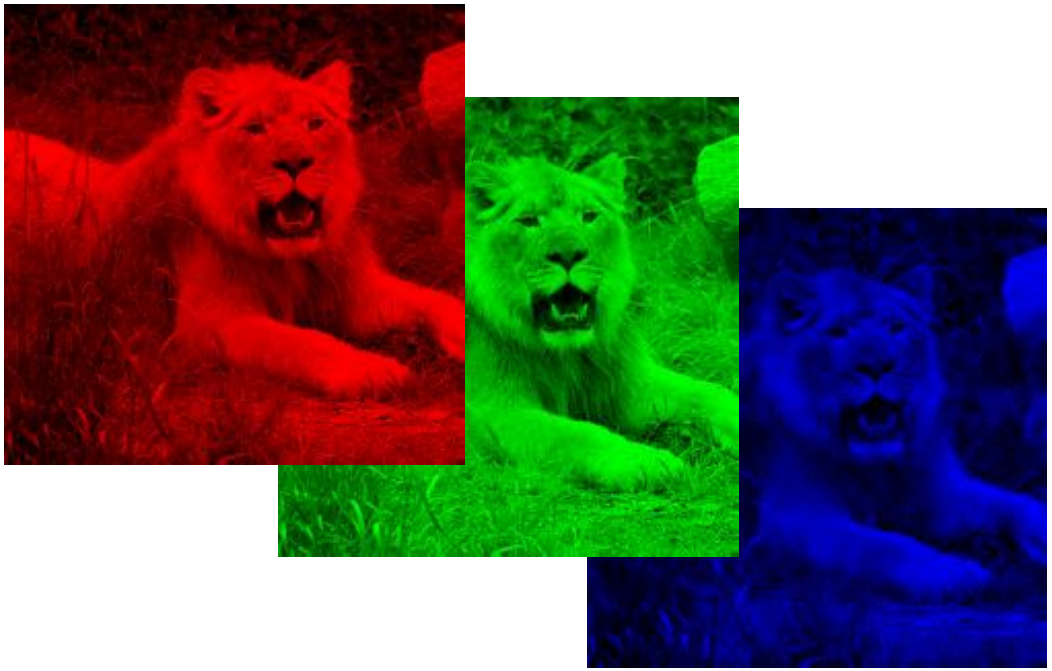
- 28 x 28 x 1 (b&w; 1 channel) = 784 features



The input layer in an ANN must have 784 neurons – can be manageable

If working with “real” images

- 224 x 224 x 3 (3 channels) = 150,528 features

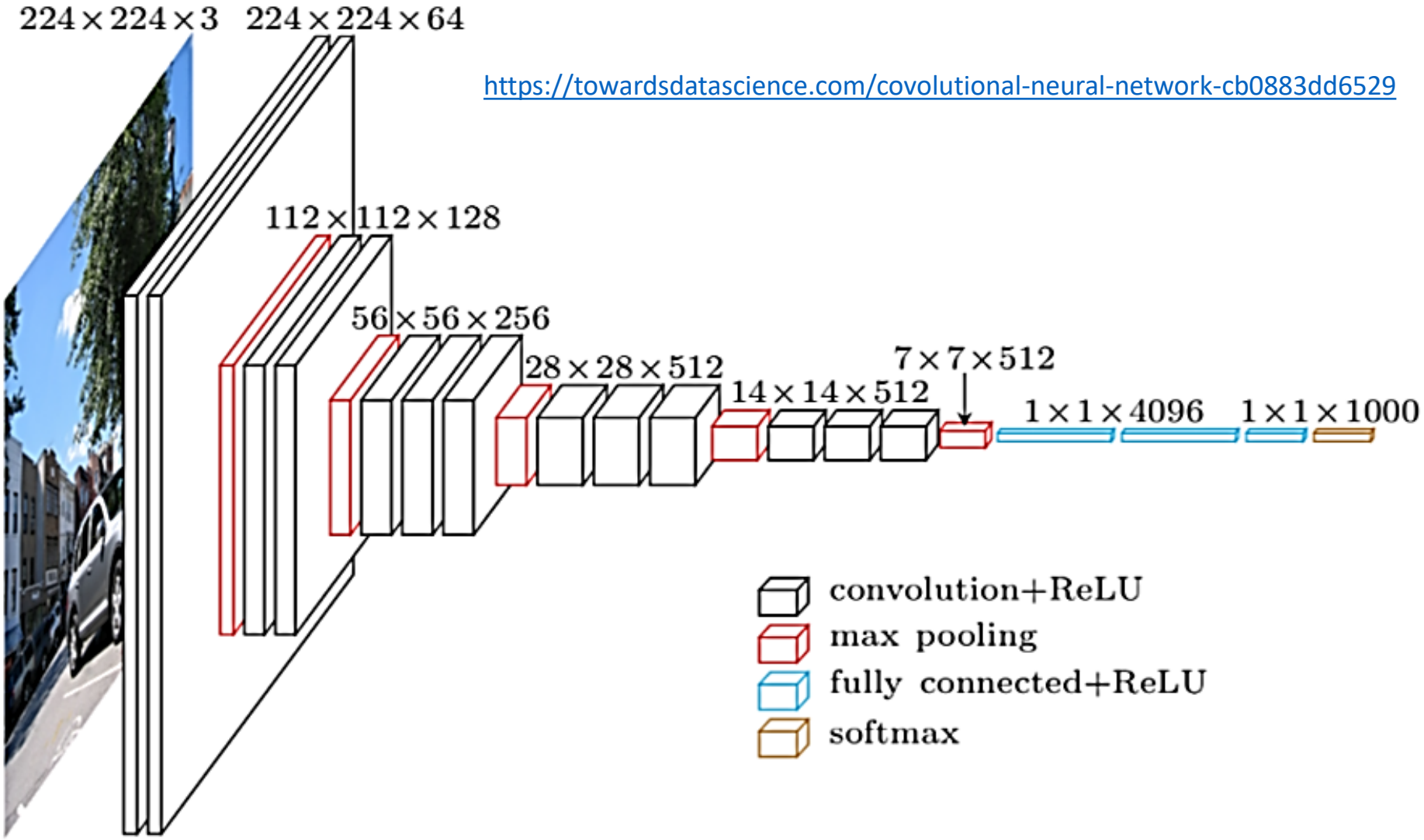


Needs 150,528 neurons!!! **Computationally inefficient!**

Applying convolution the input tensor dimension is finally reduced down to 1 x 1 x 1000

Only 1000 neurons in the 1st layer of the feed forward neural network

Downsampling

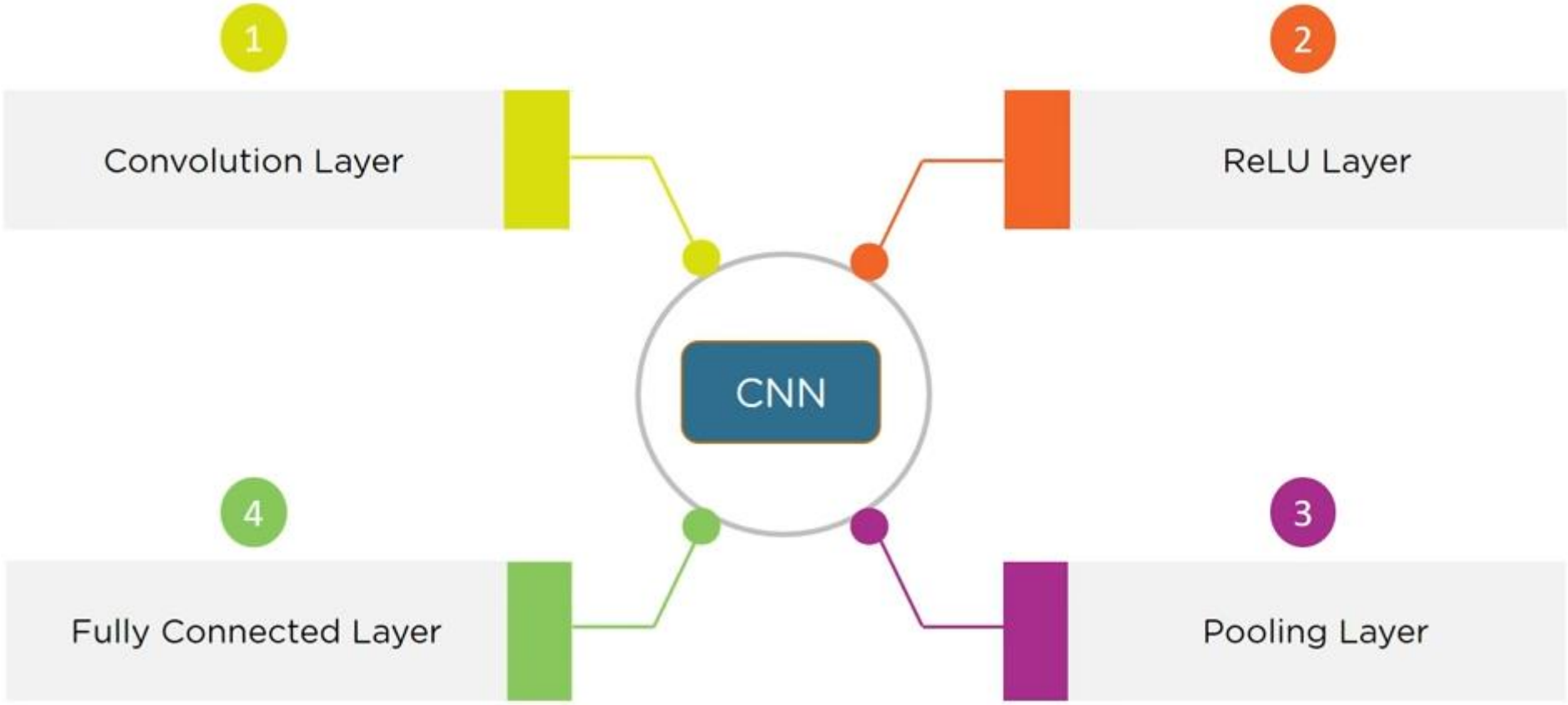


https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers

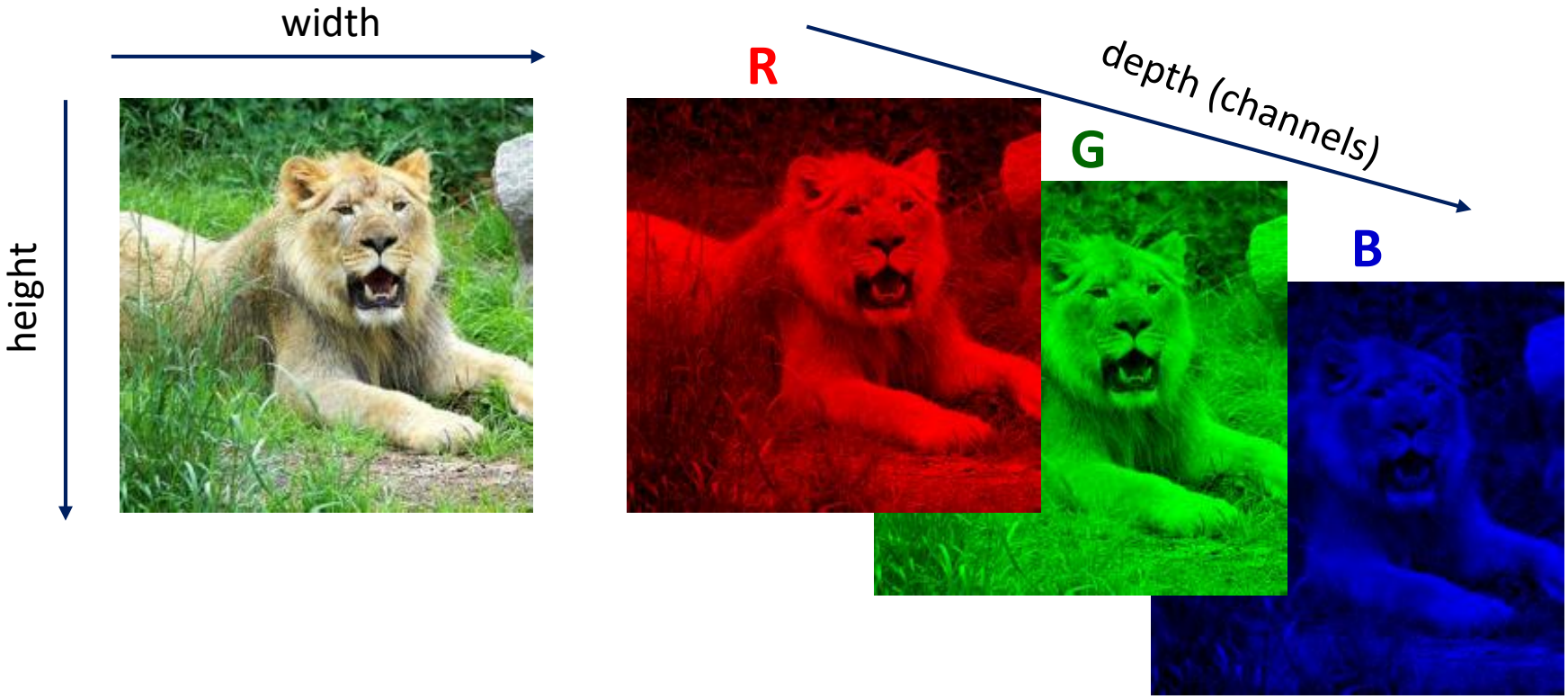
- The hidden layers of a CNN typically consist of
  - convolutional layers + ReLU
  - pooling layers
  - fully connected layers
  - normalization layers

Layers in CNN



https://www.youtube.com/watch?v=Jy9-aGMB_TE

Image representation



3D - tensor

Convolution

Convolution operates on two signals (in 1D) or two images (in 2D): you can think of one as the "input" signal (or image), and the other (called the kernel) as a "filter" on the input image, producing an output image.
Convolution takes two images as input and produces a third as output.

https://www.cs.cornell.edu/courses/cs1114/2013sp/sections/S06_convolution.pdf

Different kind of **information can be extracted**, according with the kernel (filter):

- Edge detection
- Sharpening
- Blurring
- Embossing

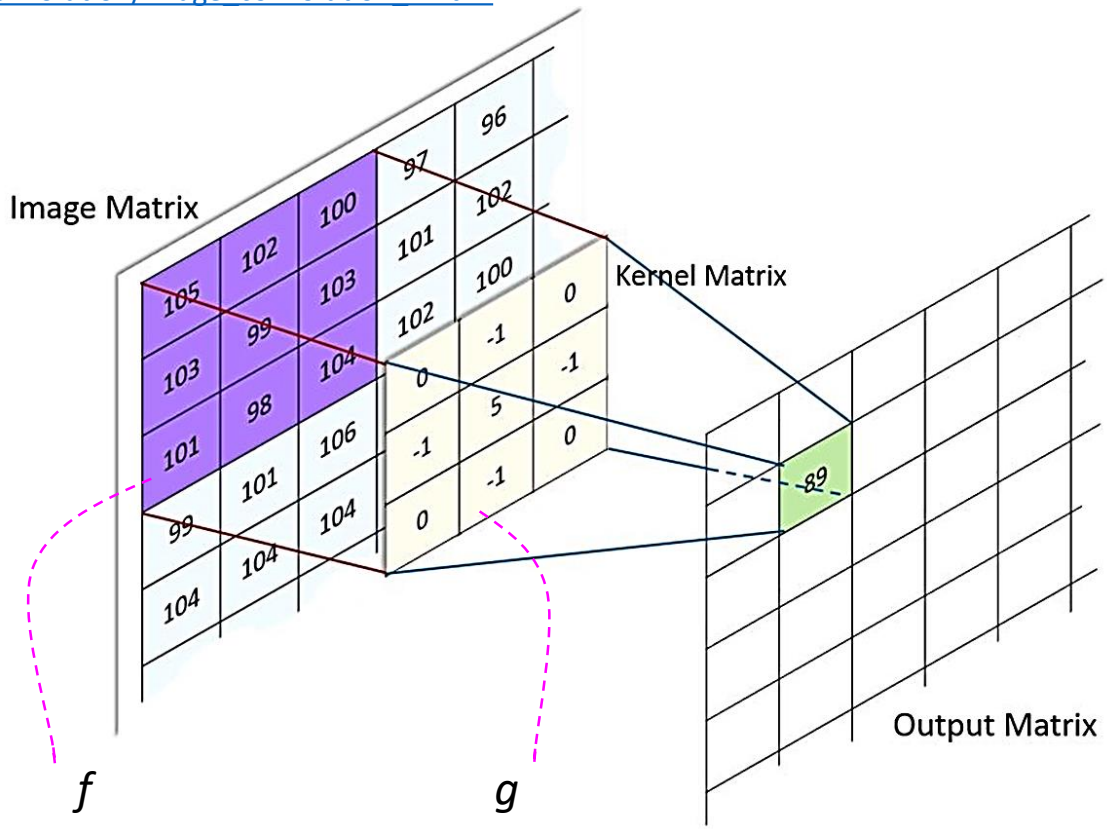
Convolution operation extract the **feature maps**

The values in all filters are training parameters of the CNN; they are learned during the training process

Convolution

Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel

http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html



$$f * g$$

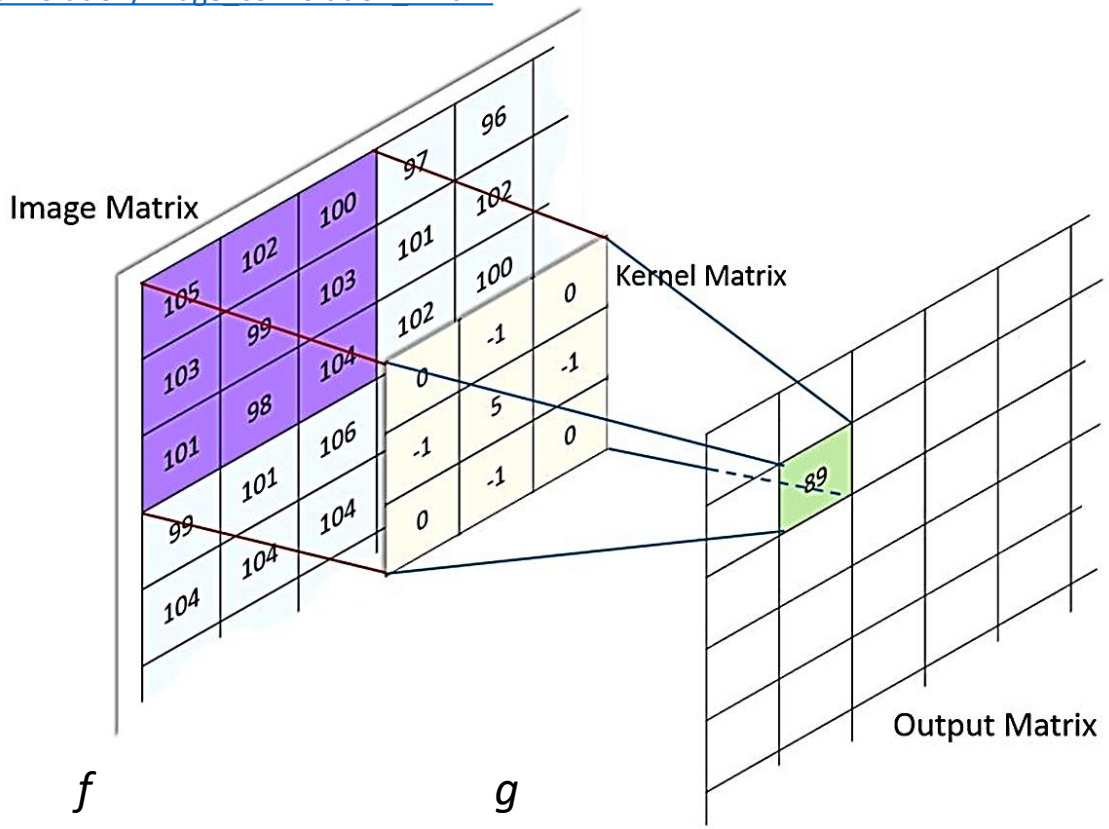
Dot product,
not matrix multiplication

Multiply the arrays element wise and sum the product

Convolution

Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel

http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html



$f * g$:

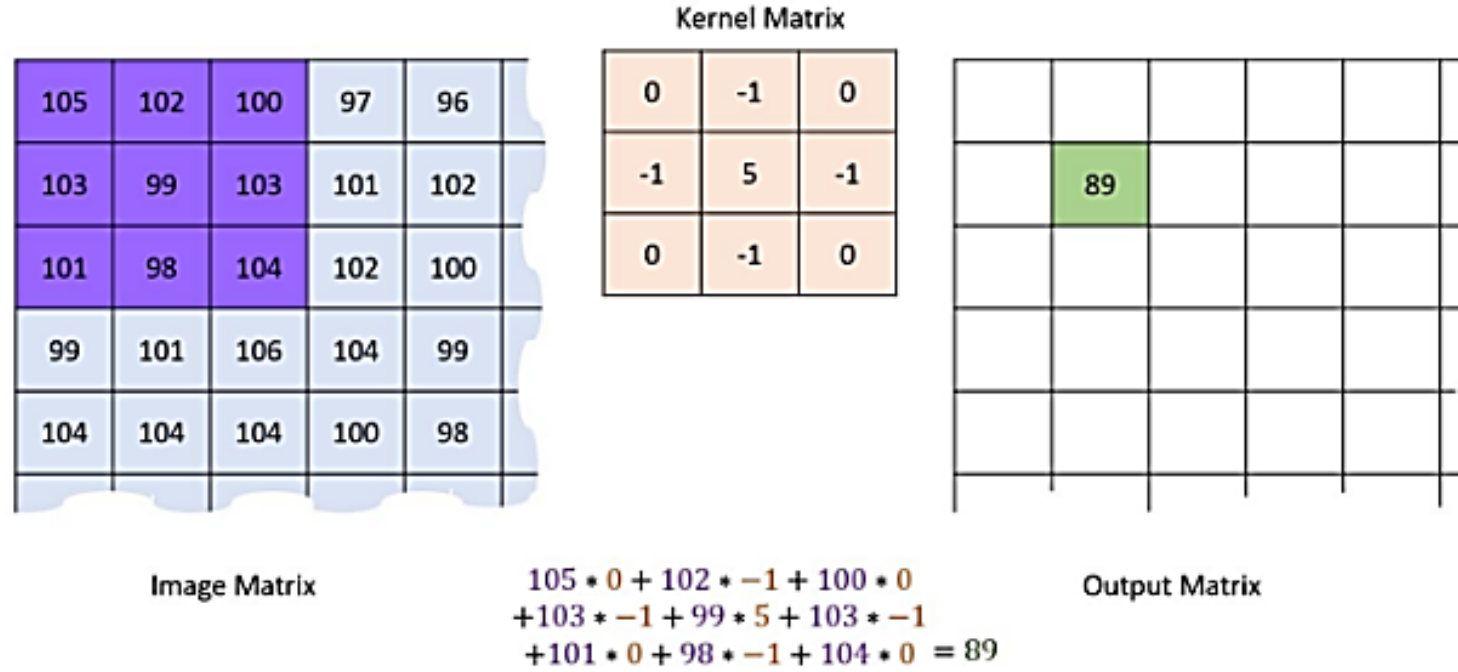
105*0 + 102*(-1) + 100*0 +
103*(-1) + 99*5 + 103*(-1) +
101*0 + 98*(-1) + 104*0 = **89**

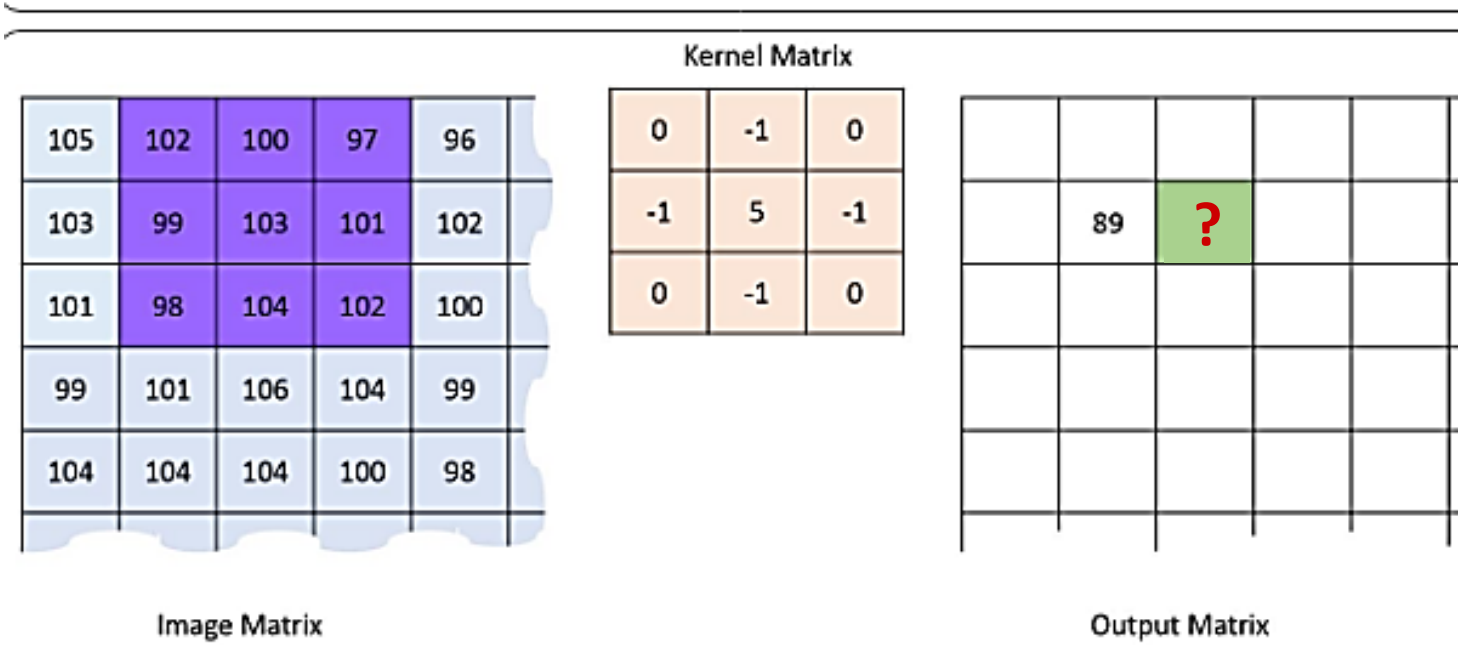
Sharpening filter (kernel)

Sharpening an image increases the contrast between bright and dark regions to bring out features.

The sharpening process is basically the application of a **high pass filter** to an image.

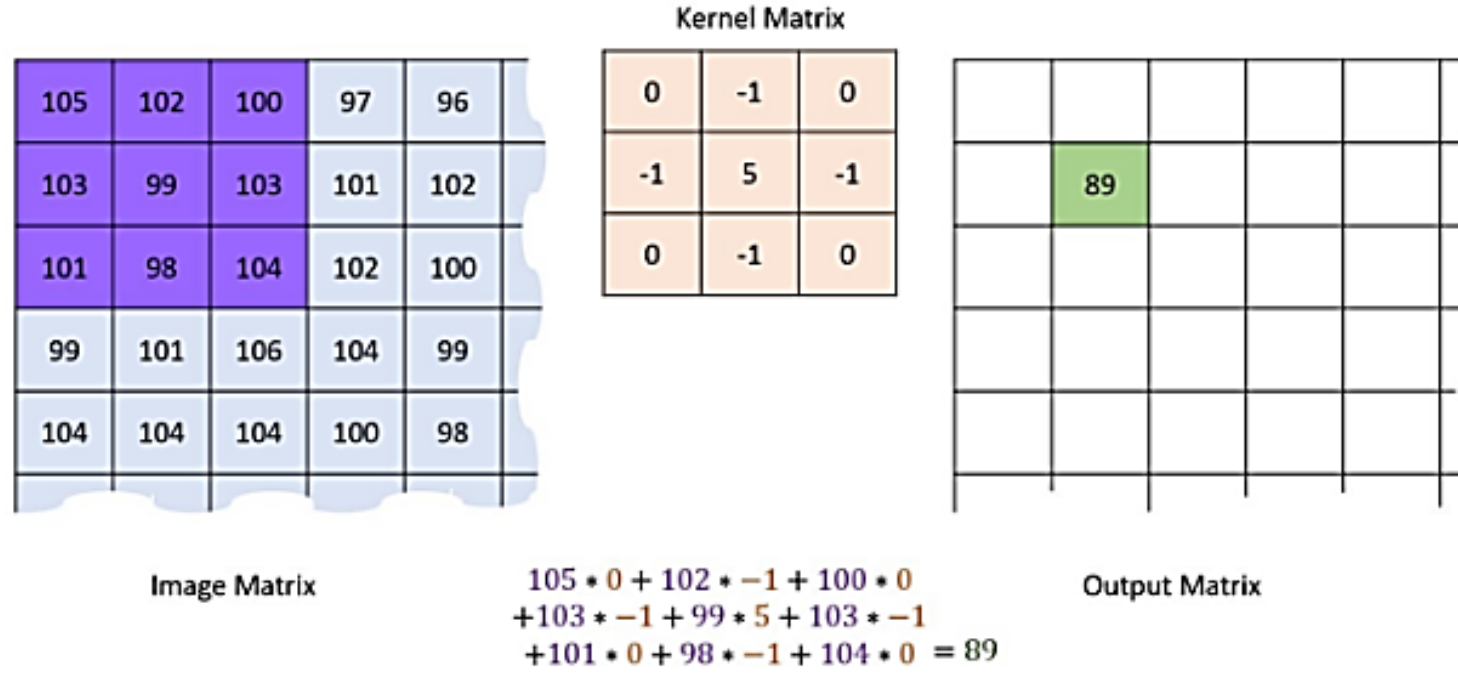
Convolution

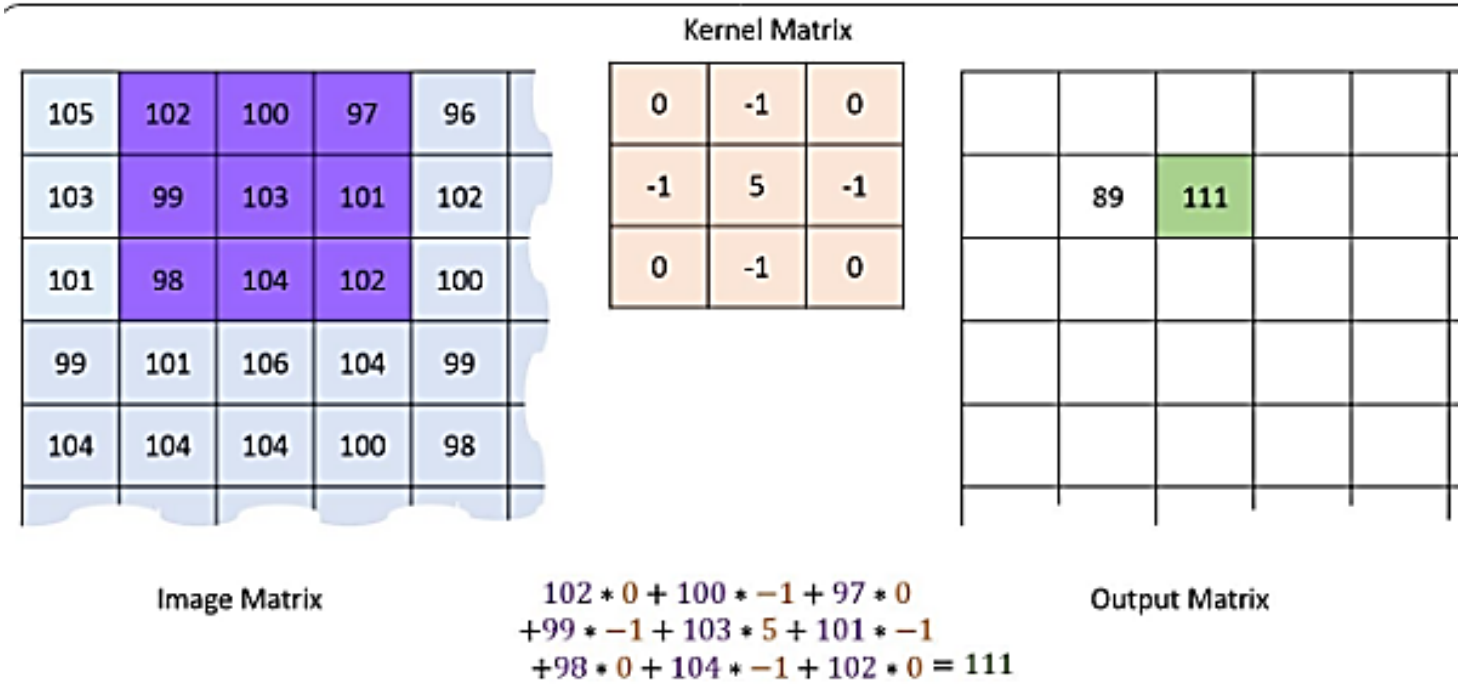




http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Convolution

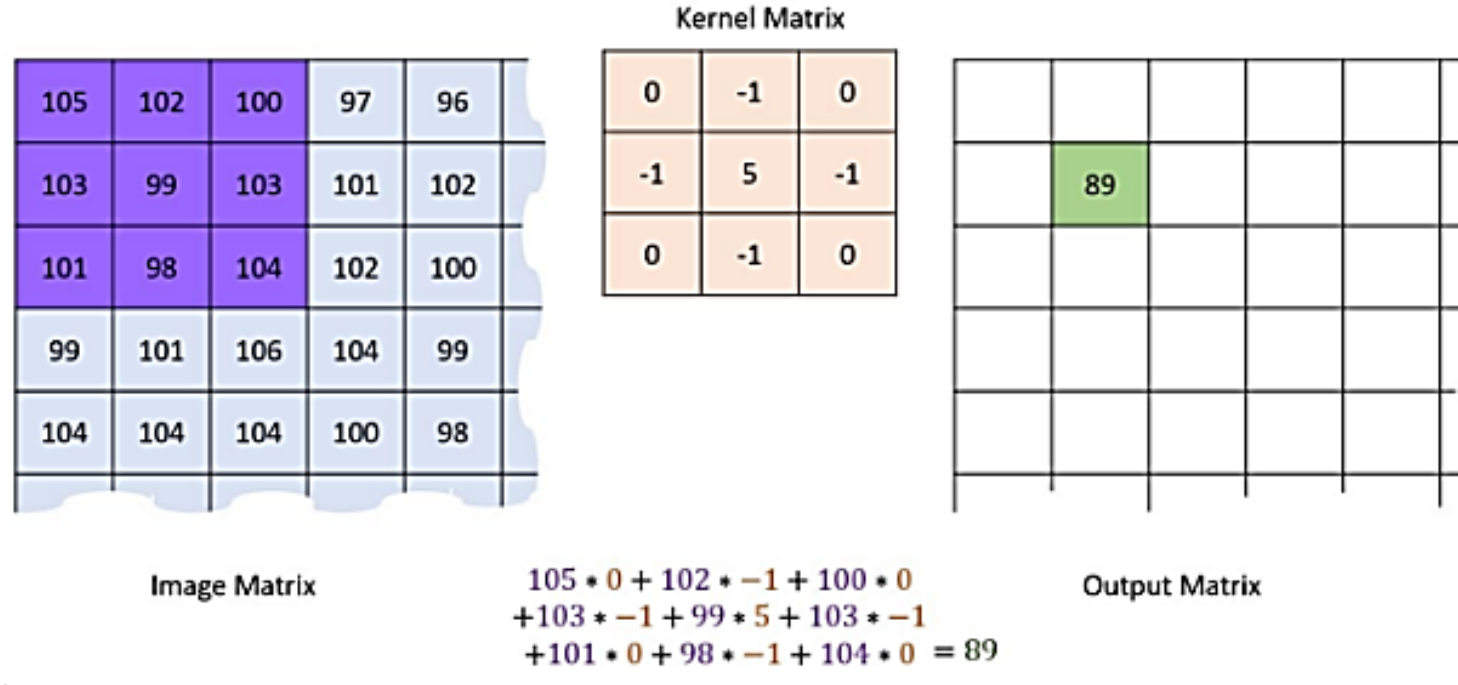


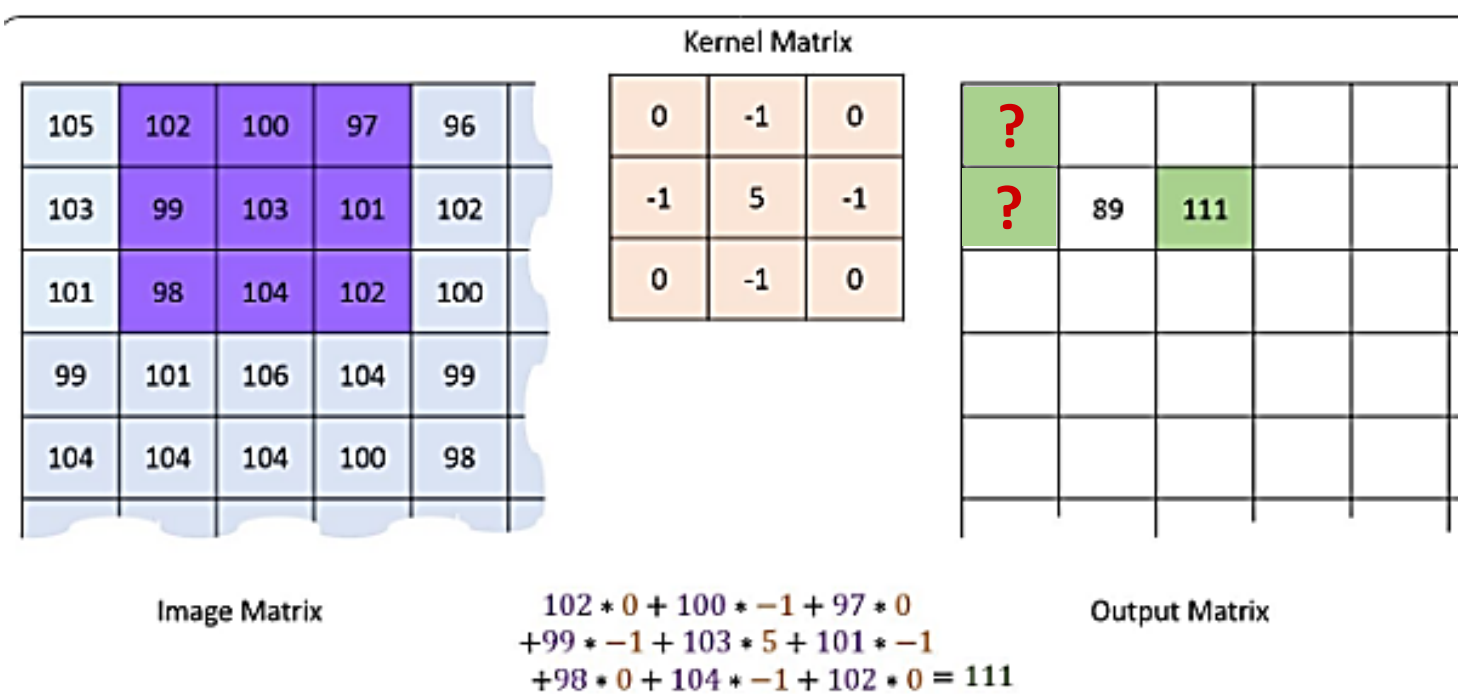


http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Convolution

Pixels on the border of image matrix?





http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

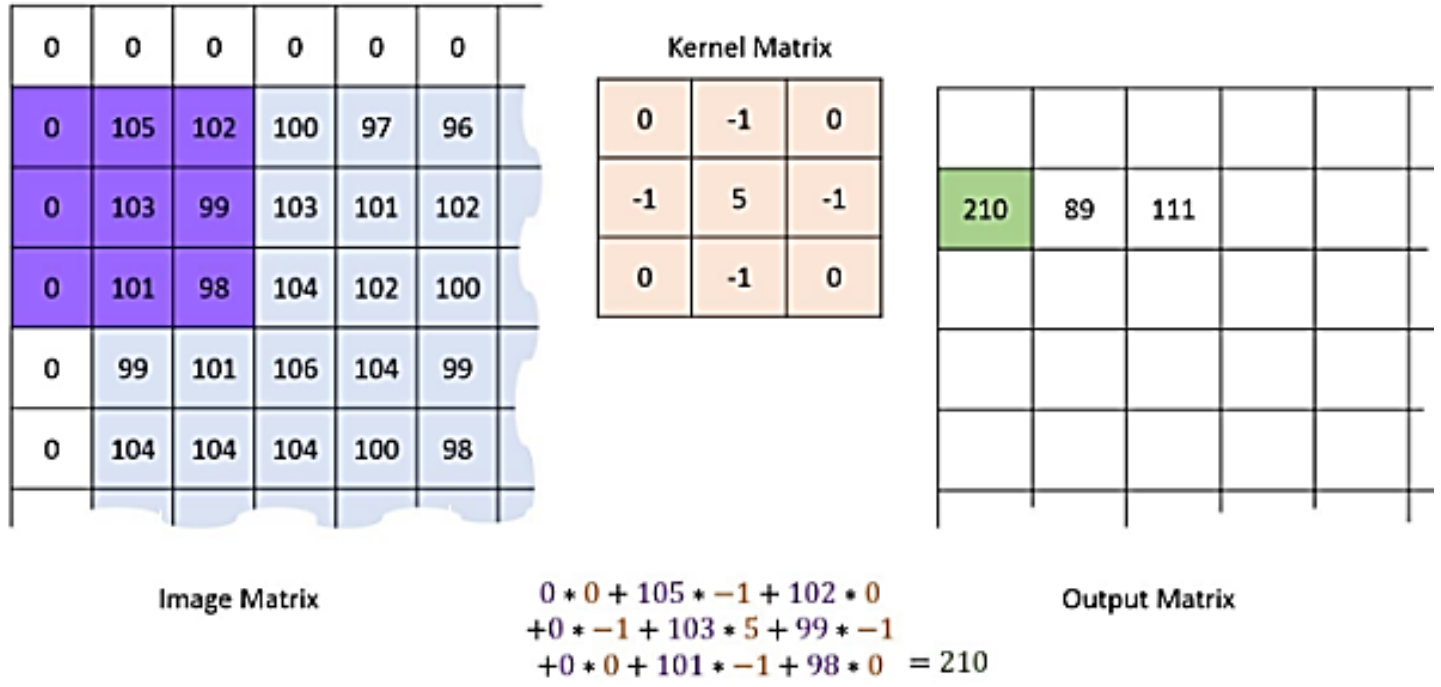
Convolution

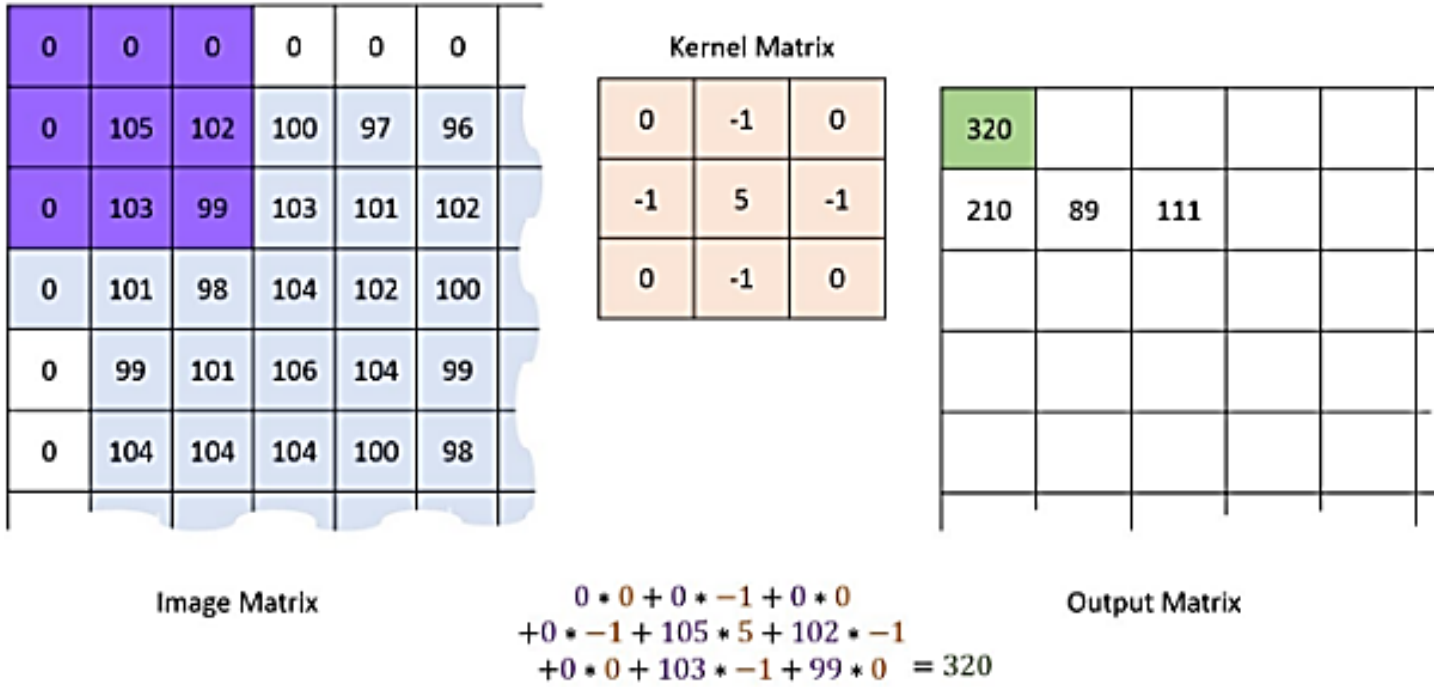
Padding

The process of adding zeros to the input matrix symmetrically to maintain the dimension of output as in input.

(1 pixel padding here)

Padding depends on the dimension of the filter





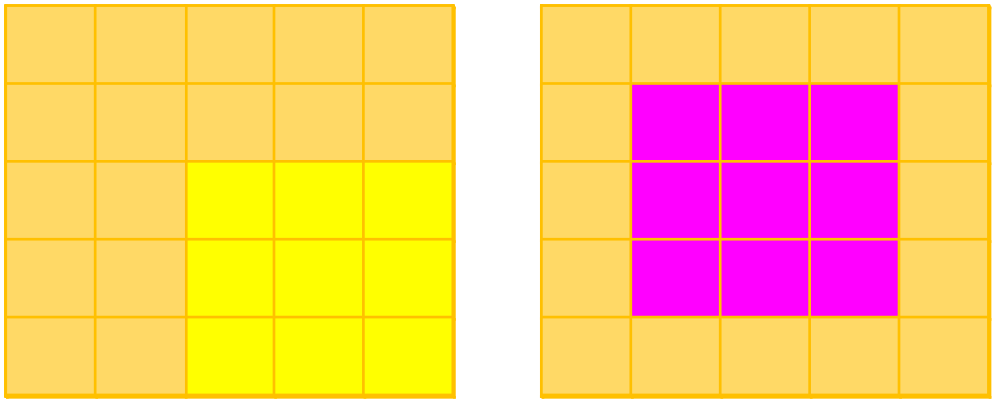
http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Convolution

Stride

Stride denotes how many steps we are moving in each steps in convolution.

Usually it is 1



Original gray image

Sharpening filter

stride = 1

0	-1	0
-1	5	-1
0	-1	0

* =

Sharpen image

Original gray image - histogram equalization

Sharpen image - histogram equalization

Original gray image

Edge detection filter

stride = 1

-1	-1	-1
-1	8	-1
-1	-1	-1

* =

Edge detection image

Original gray image - histogram equalization

Edge detection image - histogram equalization

Original gray image

Edge detection filter

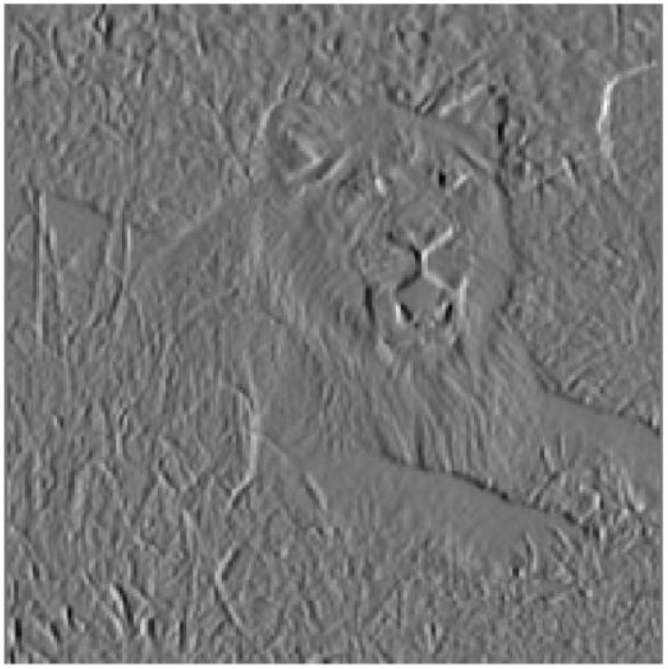
stride = 1

-1	0	+1
-2	0	+2
-1	0	+1

* =

Sobel; horizontal changes (vertical edges)

Edge detection image

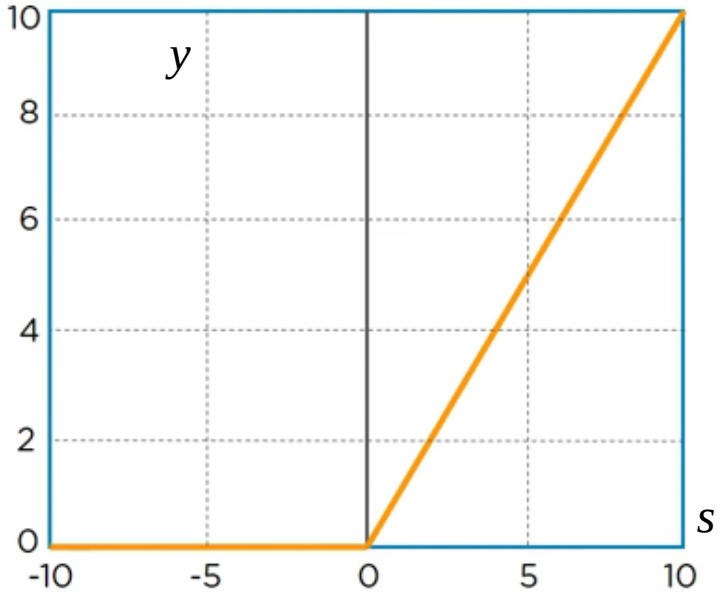


Original gray image - histogram equalization

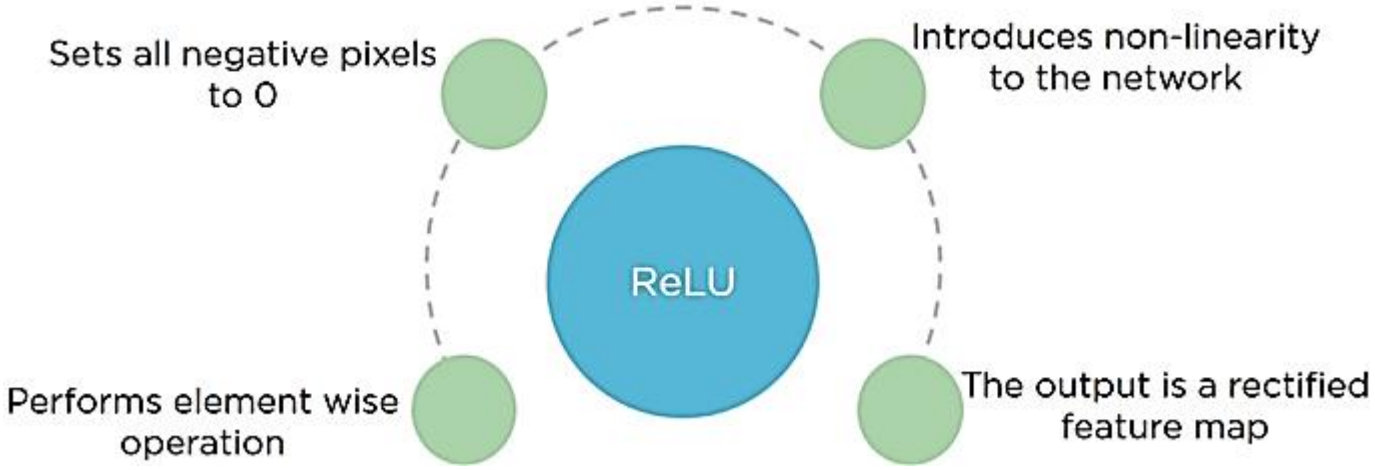
Edge detection image - histogram equalization

ReLU layer

Once the feature map is extracted by the convolution, the next operation is to apply the ReLU activation function



$$y = \max(0, s)$$



https://www.youtube.com/watch?v=Jy9-aGMB_TE

Convolution

Multiple filters

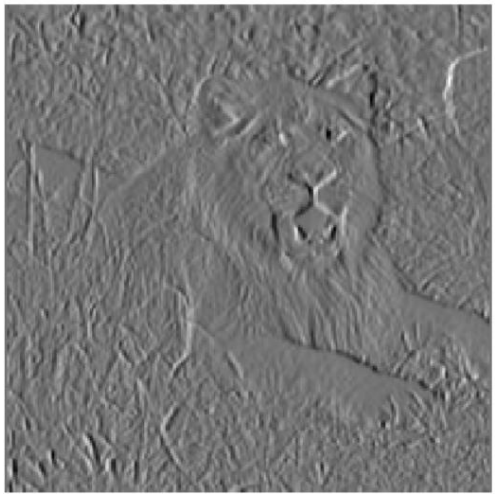
ReLU

Original gray image

Input image

Sharpen image

Edge detection image



Input feature map

Sharpen image + ReLU

Edge detection image + ReLU

Rectified feature map

Input image is scanned in multiple convolution and ReLU layers

Original gray image

First 5 columns and rows of the input image matrix:

[[33.6861 21.462 22.3216 32.8946 29.1734]
[62.8323 38.6861 21.462 15.7388 12.1618]
[53.7622 55.191 50.8323 39.2553 27.7504]
[67.3392 79.3392 74.3372 57.8323 47.2553]
[85.2028 79.8441 50.8421 29.3314 34.8323]]

Sharpen image

First 5 columns and rows of the image sharpen matrix:

[[84.1362 12.3258 36.2923 95.4401 78.5729]
[188.1617 33.9352 -19.8438 -29.4384 -28.3799]
[80.8206 54.1131 63.916   44.8381 15.6216]
[121.9746 116.2651 133.6276 98.9823 78.1111]
[196.6922 98.347   8.774 -31.7601 18.4111]]

Sharpen image + ReLU

First 5 columns and rows of the image sharpen+ReLU matrix:

[[84.1362 12.3258 36.2923 95.4401 78.5729]
[188.1617 33.9352   0.   0.   0.   ]
[80.8206 54.1131 63.916   44.8381 15.6216]
[121.9746 116.2651 133.6276 98.9823 78.1111]
[196.6922 98.347   8.774   0.   18.4111]]

Pooling layer

Once the feature map is rectified by the ReLU activation function, the next operation is to **down-sampling** the images to **reduce the dimensionality** through a **pooling layer**

0	0	14	82
149	32	0	0
28	53	64	44
39	120	133	99

Rectified feature map

4 x 4 = 16

max pooling

2 x 2 filter
stride 2

149	82
120	133

Pooled feature map

2 x 2 = 4

Dimensionality reduction (given by the filter size and stride):

4 to 1; 4 times

Dimensionality reduction for 2x2 filter and stride 1?

Pooling layer

Max pooling – how valuable is a feature in the area of the filter

- best (maximum feature value)

0	0	14	82
149	32	0	0
28	53	64	44
39	120	133	99

Rectified feature map

max pooling

2 x 2 filter
stride 2

149	82
120	133

Pooled feature map

2 x 2 = 4

Pooling layers uses different filters to be able to identify different aspects in an image: edges, corners, body parts (eyes, ear, paw, fur, etc)

Flattening

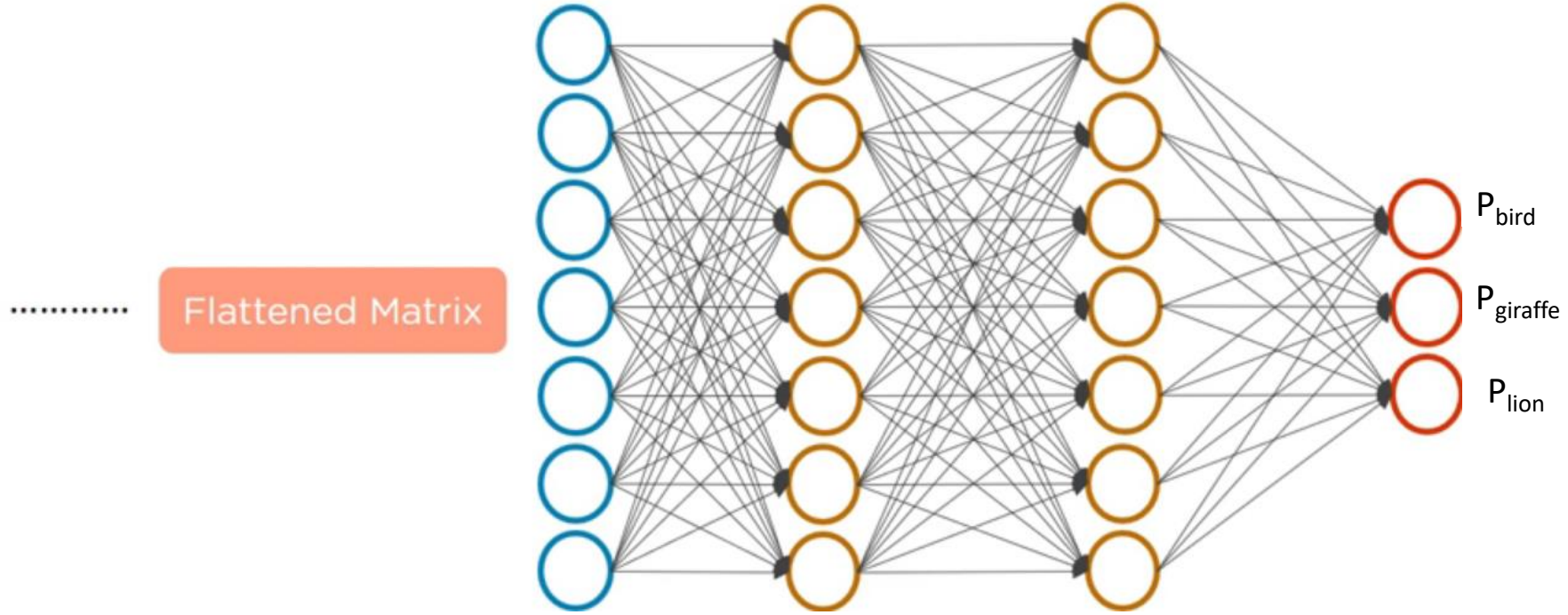
Once the dimensionality is reduced to a manageable size, we have to connect further with the fully connected layer.

We need to convert the 2D dimensional array from pooled featured map to a single vector using the **flattening** operation

Pooled feature map

Fully connected layer

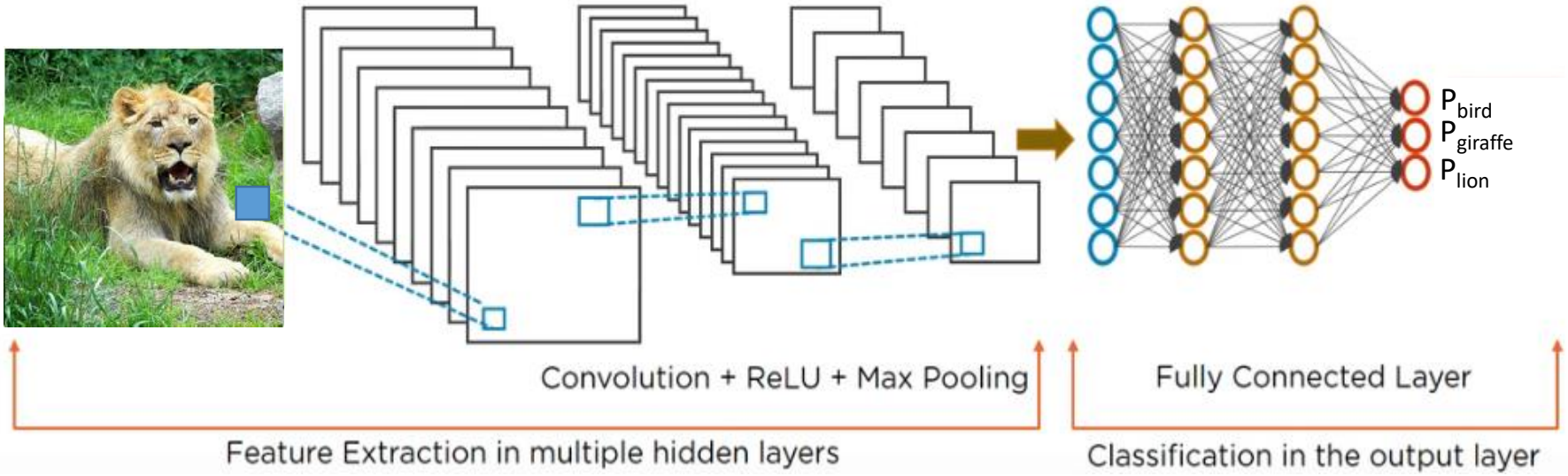
The vector (flattened 2D array) from the pooling layer is fed to the fully connected layer (conventional feed-forward ANN) to classify the image



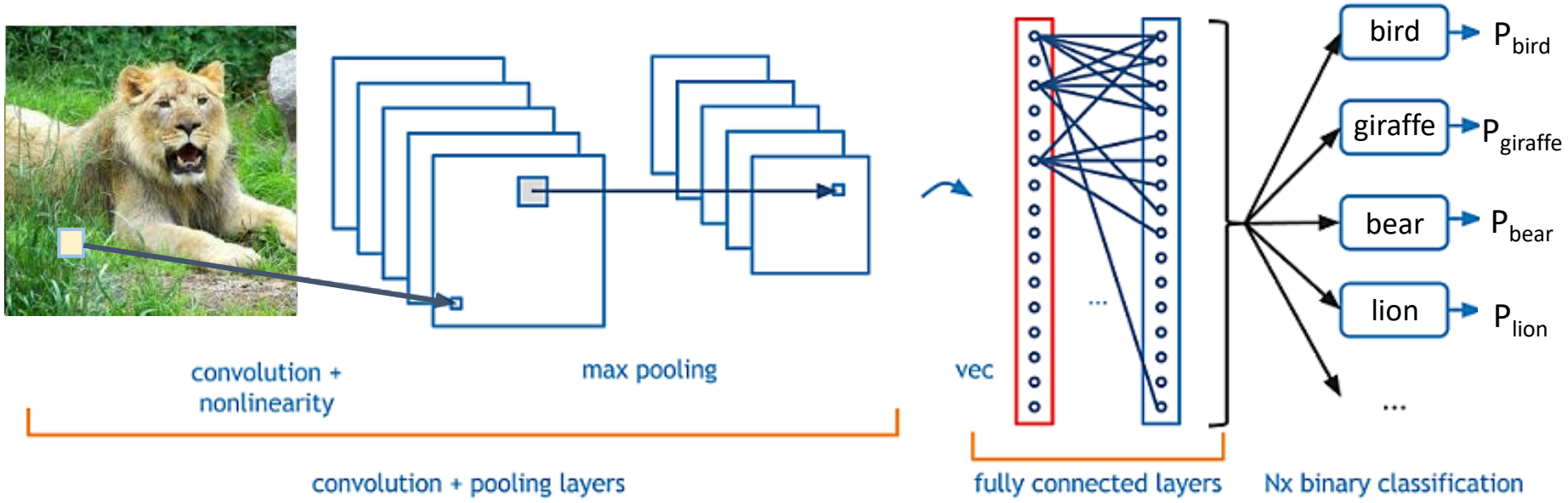
https://www.youtube.com/watch?v=Jy9-aGMB_TE

Fully connected layer

CNN – big picture

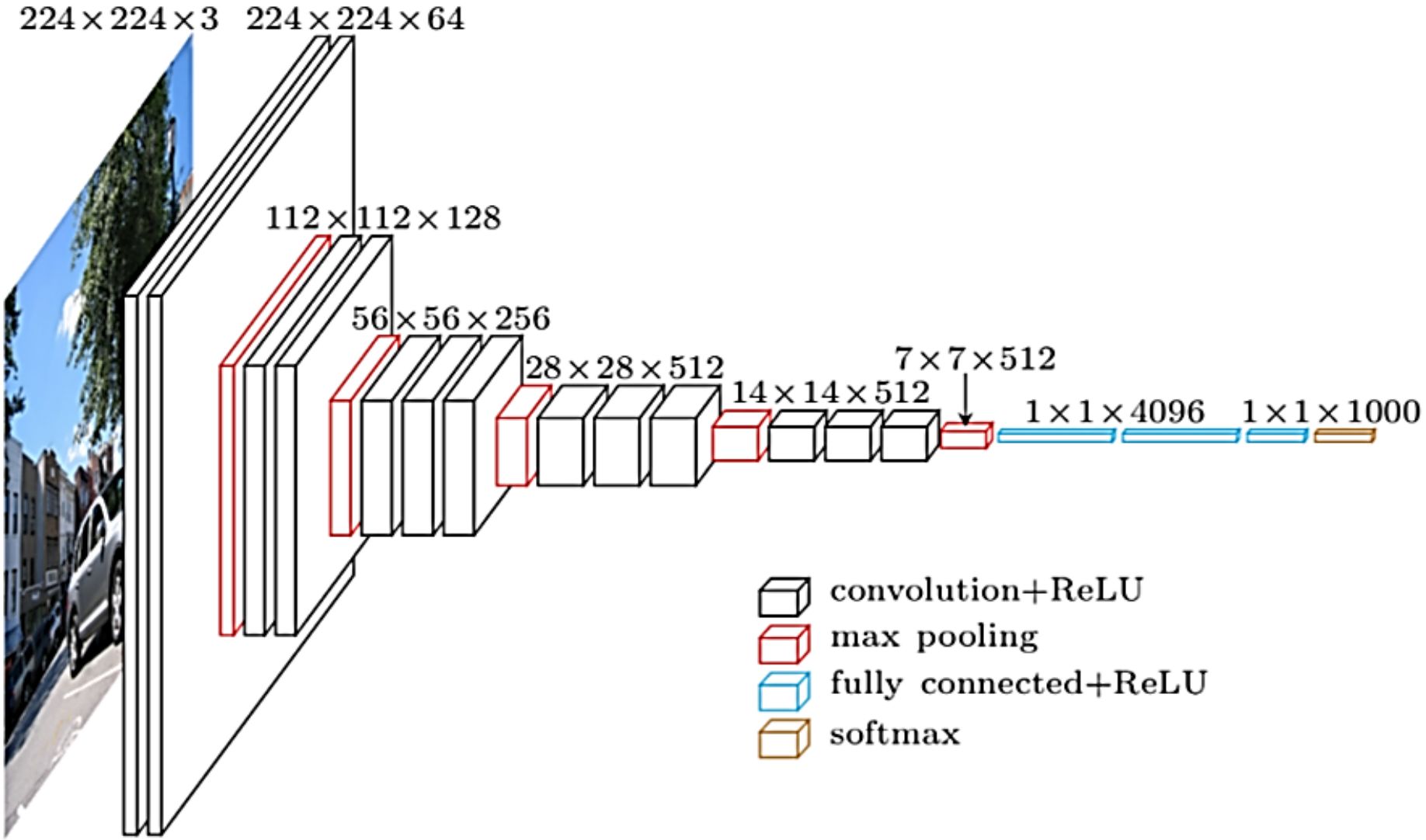


CNN – big picture



Adaptation after: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

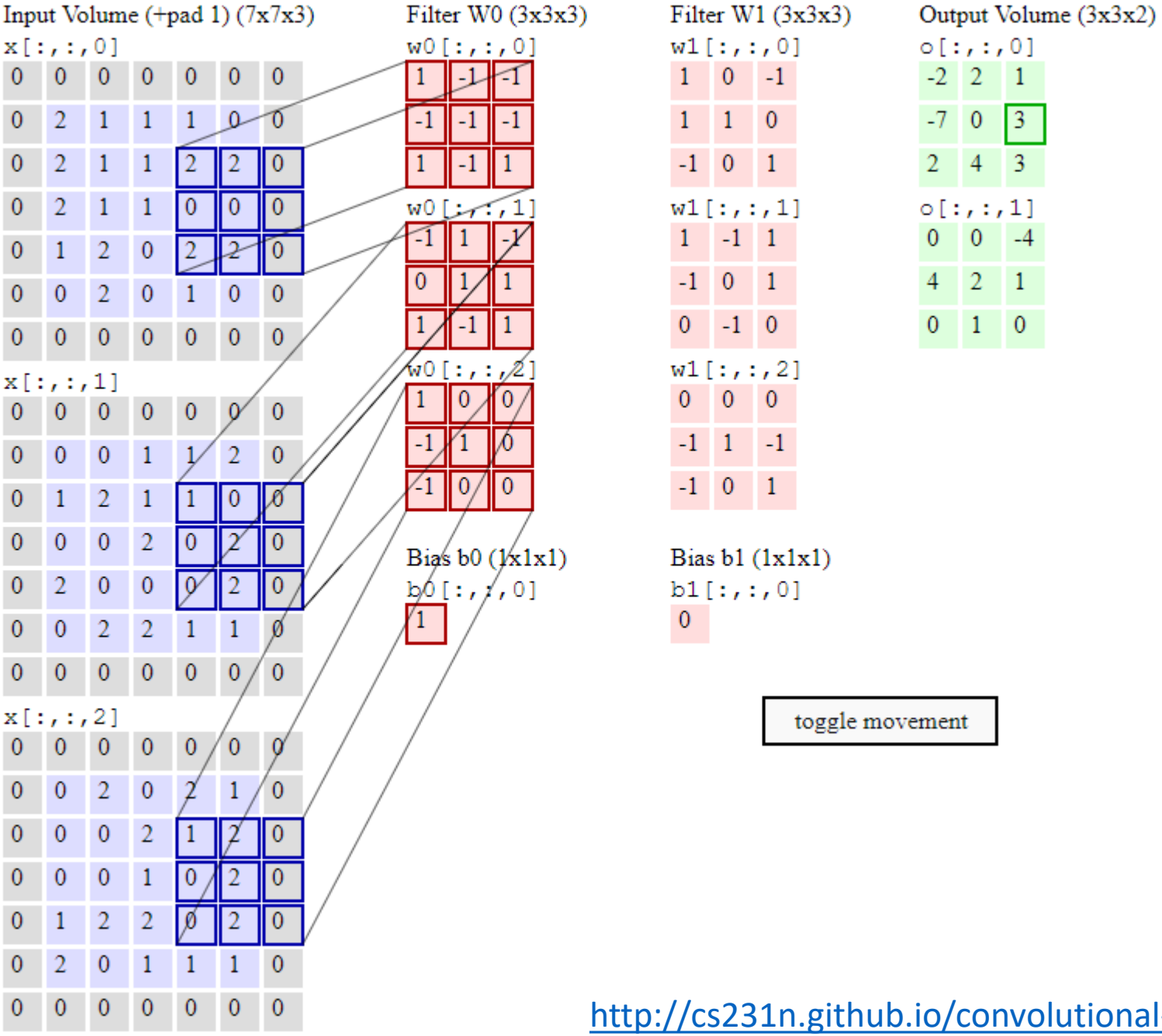
CNN – the big picture



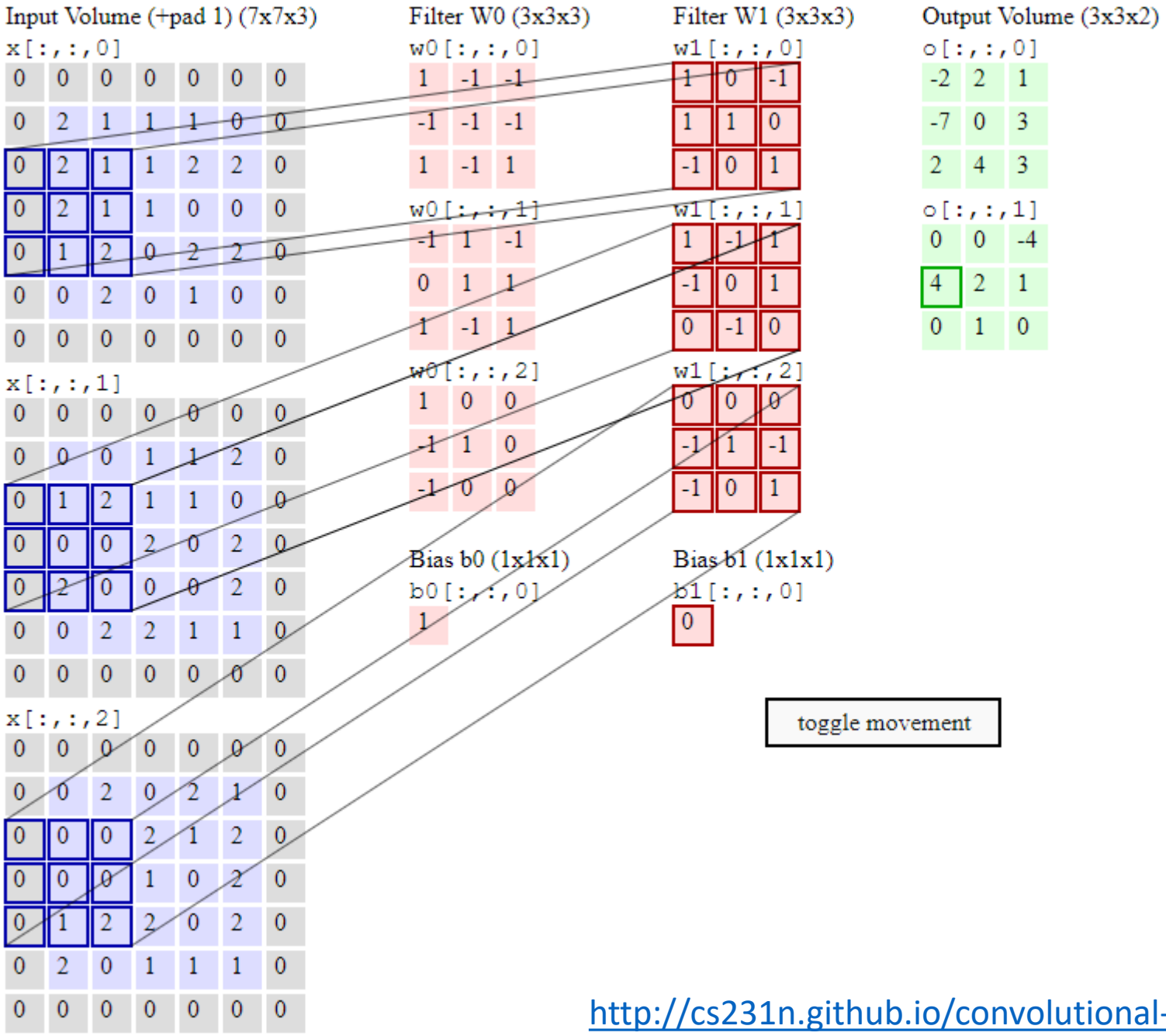
https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Convolution Demo. Below is a running demo of a CONV layer. Since 3D volumes are hard to visualize, all the volumes (the input volume (in blue), the weight volumes (in red), the output volume (in green)) are visualized with each depth slice stacked in rows. The input volume is of size $W_1 = 5, H_1 = 5, D_1 = 3$, and the CONV layer parameters are $K = 2, F = 3, S = 2, P = 1$. That is, we have two filters of size 3×3, and they are applied with a stride of 2. Therefore, the output volume size has spatial size (5 - 3 + 2)/2 + 1 = 3. Moreover, notice that a padding of $P = 1$ is applied to the input volume, making the outer border of the input volume zero. The visualization below iterates over the output activations (green), and shows that each element is computed by elementwise multiplying the highlighted input (blue) with the filter (red), summing it up, and then offsetting the result by the bias.

http://cs231n.github.io/convolutional-networks/



http://cs231n.github.io/convolutional-networks/



http://cs231n.github.io/convolutional-networks/