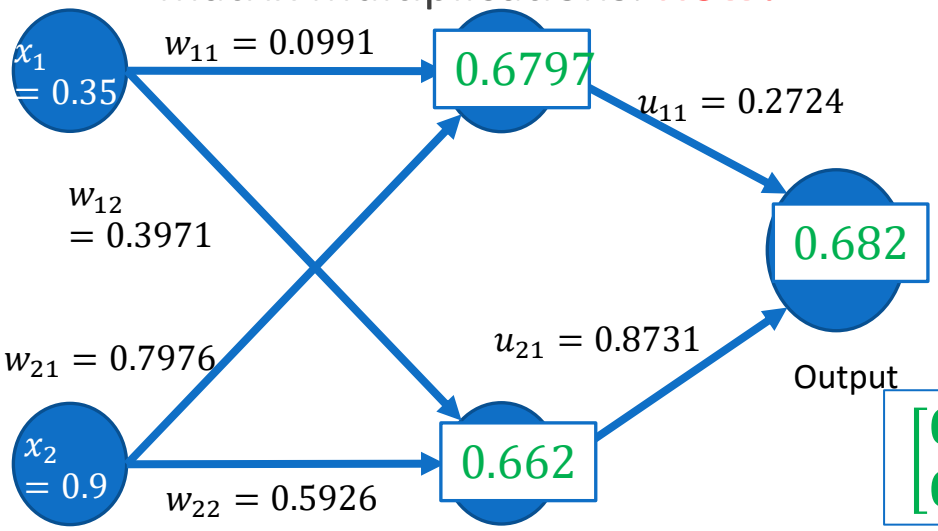
Multilayer Perceptron

RECAP

In class activity…

- The output of the above model in forward pass is nothing but two matrix multiplications. **How?**



$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$h(X) = W^T X$$

$$\begin{bmatrix} w_{11}, w_{21} \\ w_{12}, w_{22} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [\,]$$

$$\begin{bmatrix} 0.0991, 0.7976 \\ 0.3971, 0.5926 \end{bmatrix} \times \begin{bmatrix} 0.35 \\ 0.9 \end{bmatrix} = \begin{bmatrix} 0.7525 \\ 0.6723 \end{bmatrix}$$

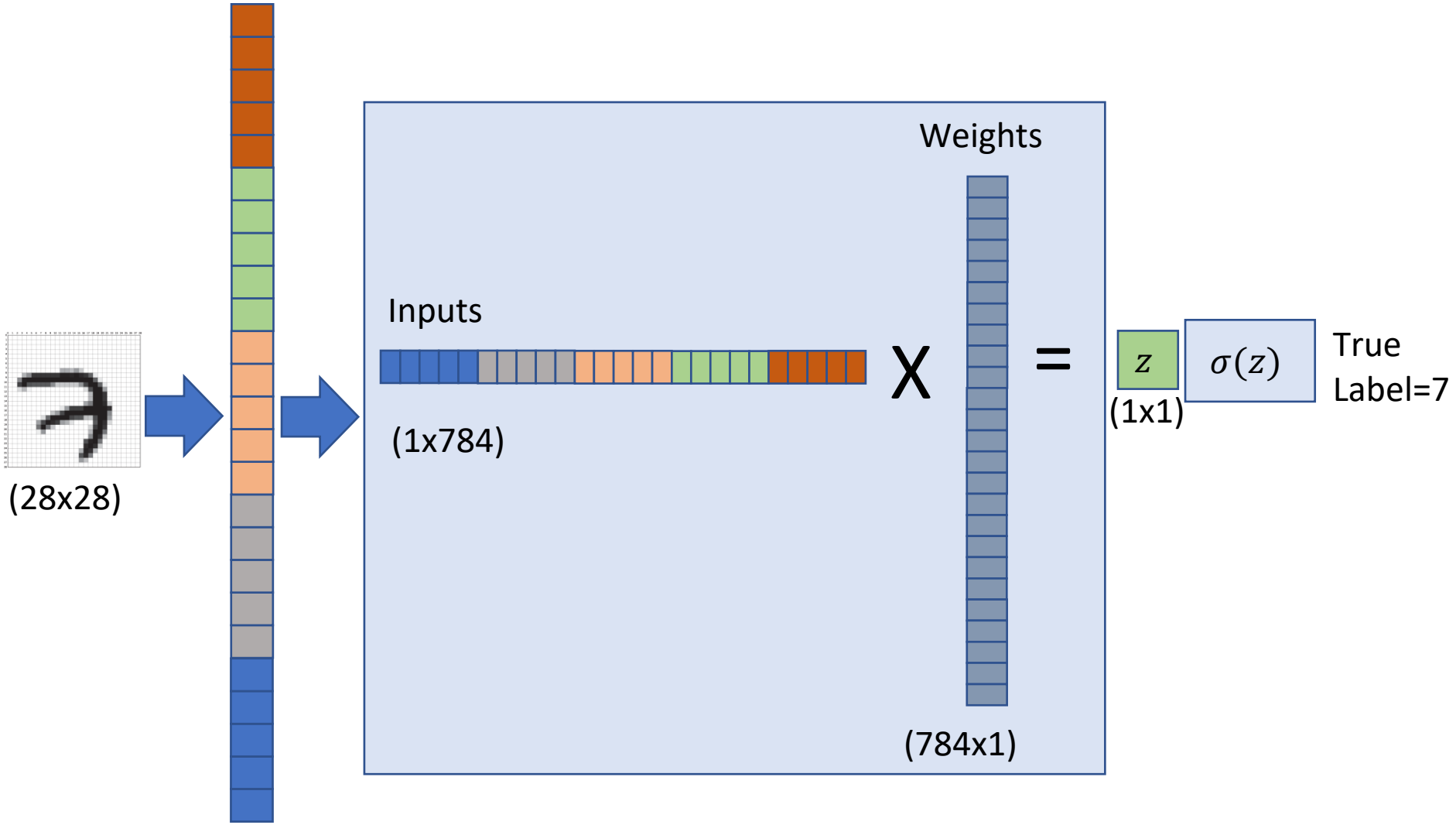
$$\sigma(z_i) = \begin{bmatrix} 0.6797 \\ 0.662 \end{bmatrix}$$

$$\begin{bmatrix} u_{11} \\ u_{21} \end{bmatrix} \times \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = [\,] \qquad h(X) = U^T X \qquad [u_{11}, u_{21}] \times \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = [\,]$$

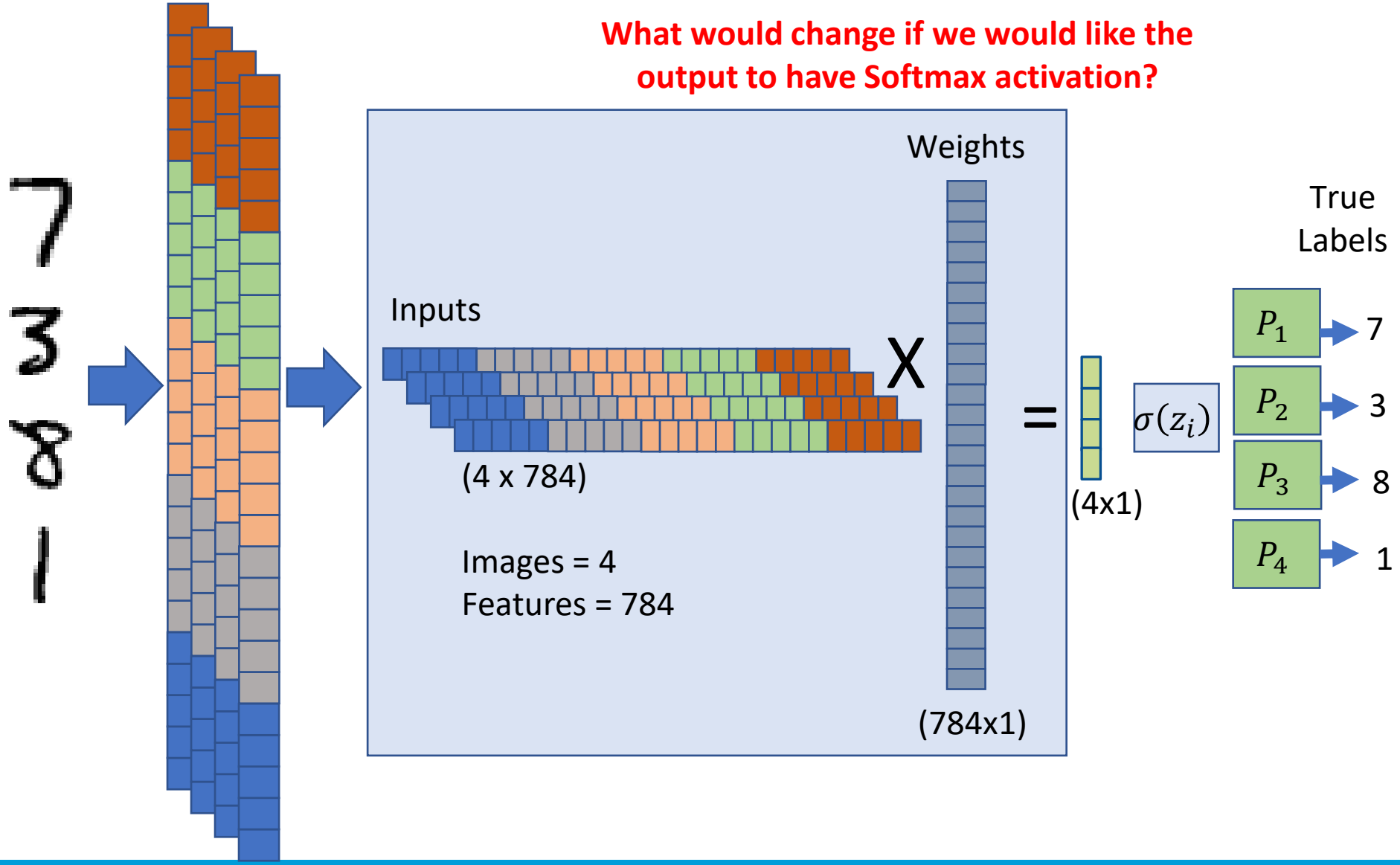
$$[0.2724, 0.8731] \times \begin{bmatrix} 0.6797 \\ 0.662 \end{bmatrix} = [0.7631]$$

$$\sigma(z_i) = [0.682]$$

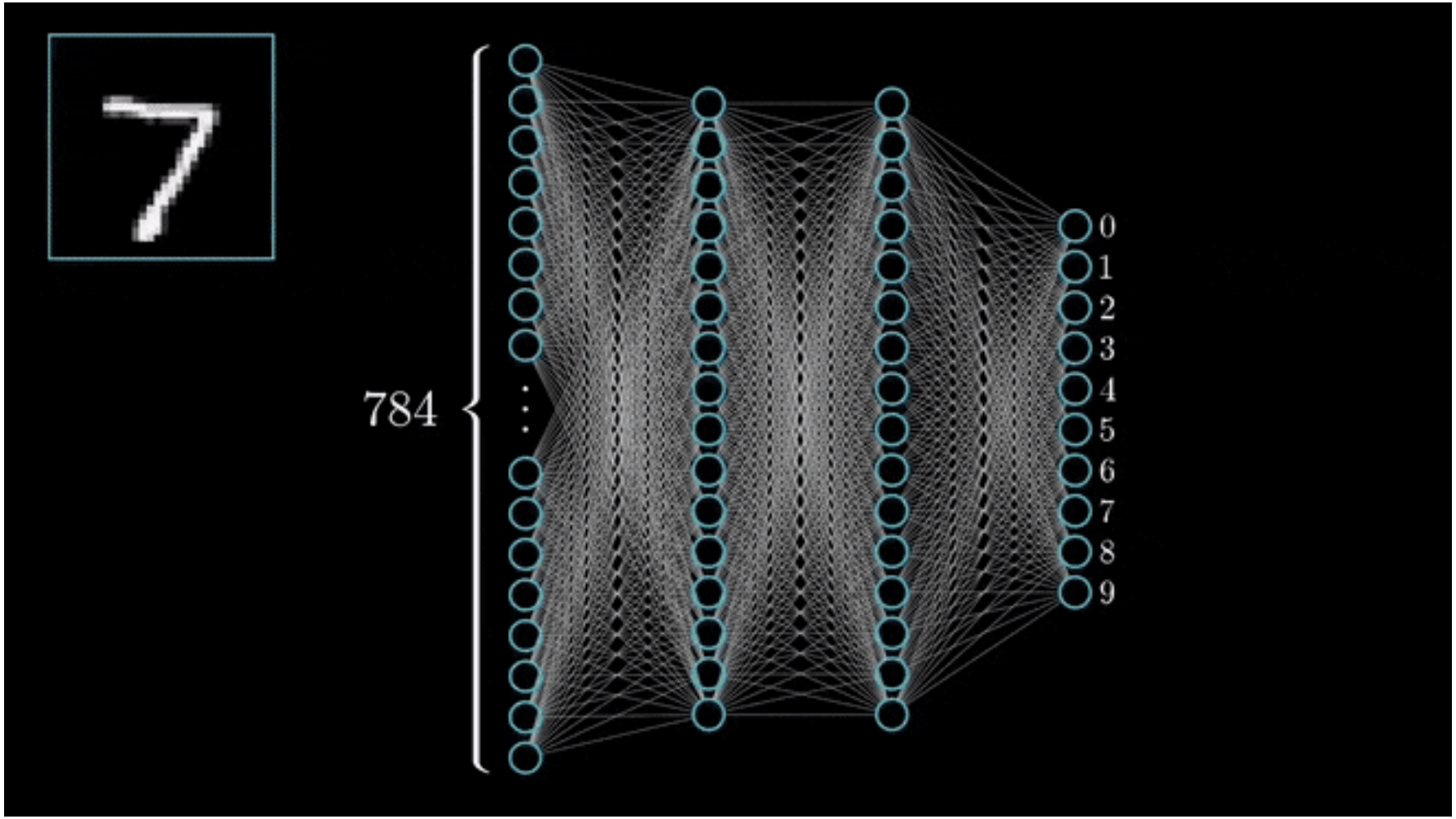
Single Image Prediction



Multiple Image Prediction



MLP Animation



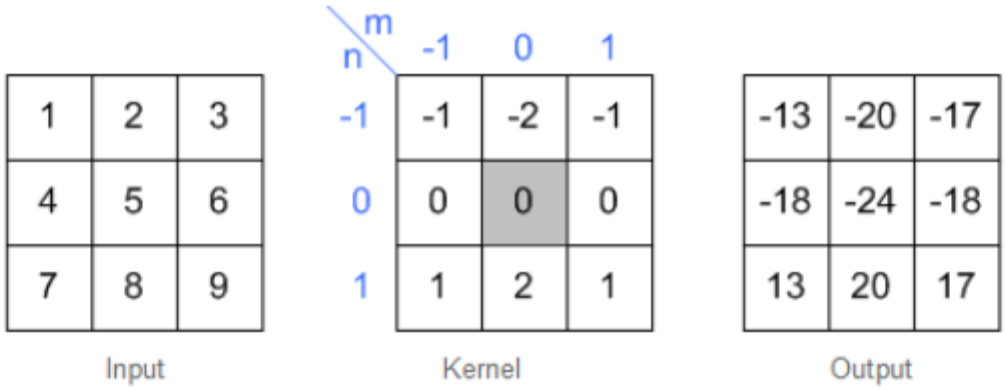
Credit: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

Hidden Layers, Neurons, Polynomial Features

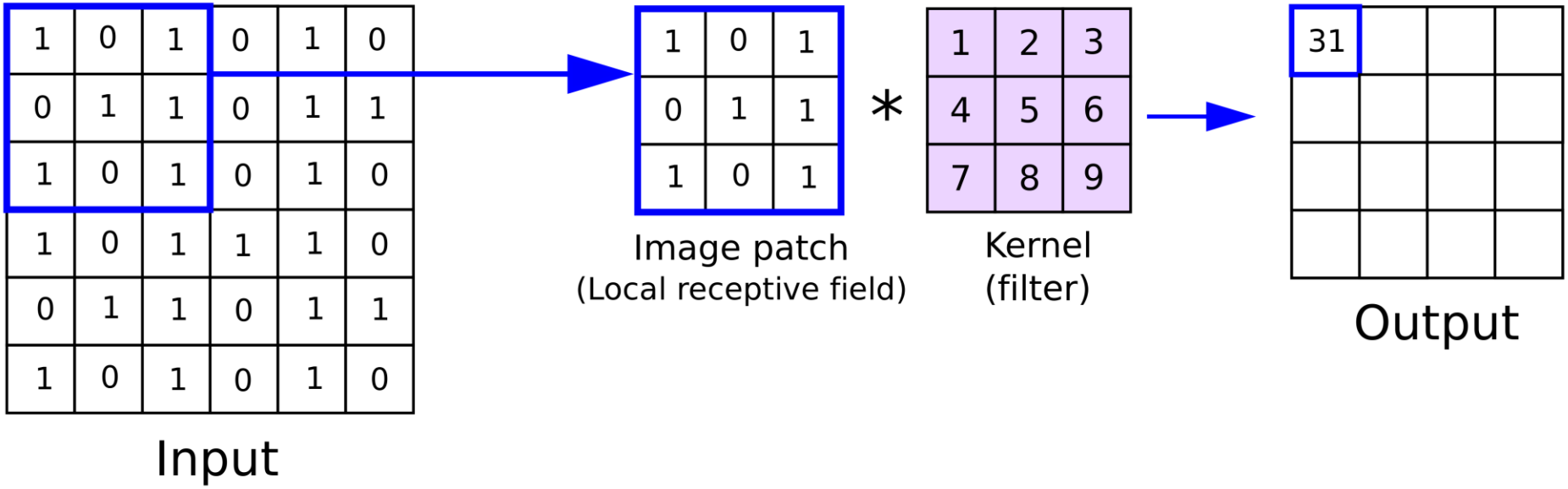
- Demo
  - http://playground.tensorflow.org/

Convolutions and Filters

Filter/Kernel



Convolution



Credit: https://anhreynolds.com/blogs/cnn.html

Convolution

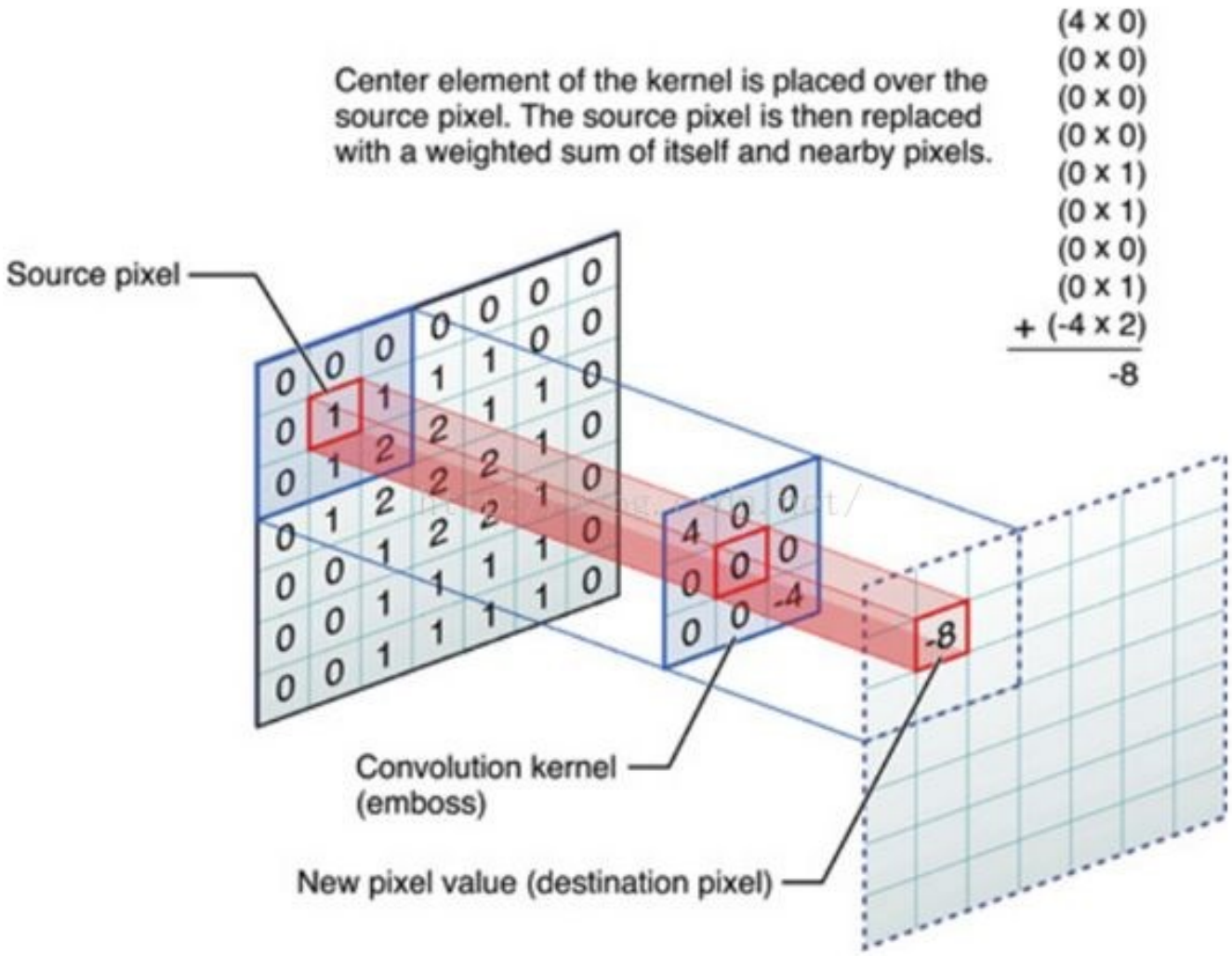
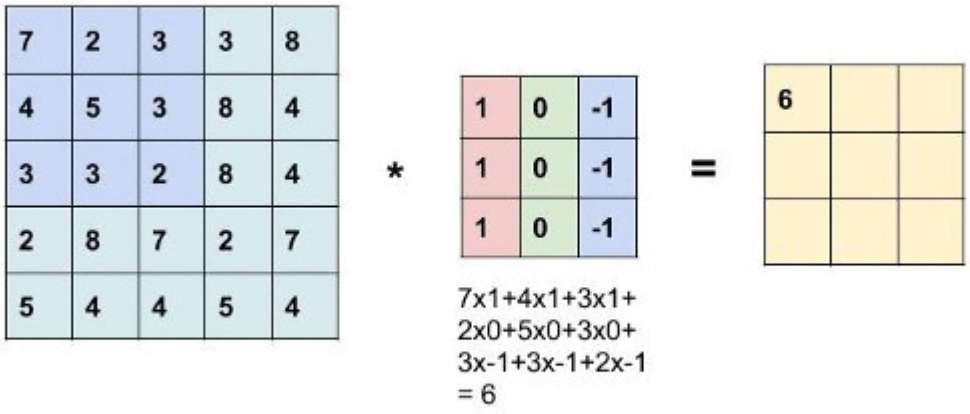


Image Filtering

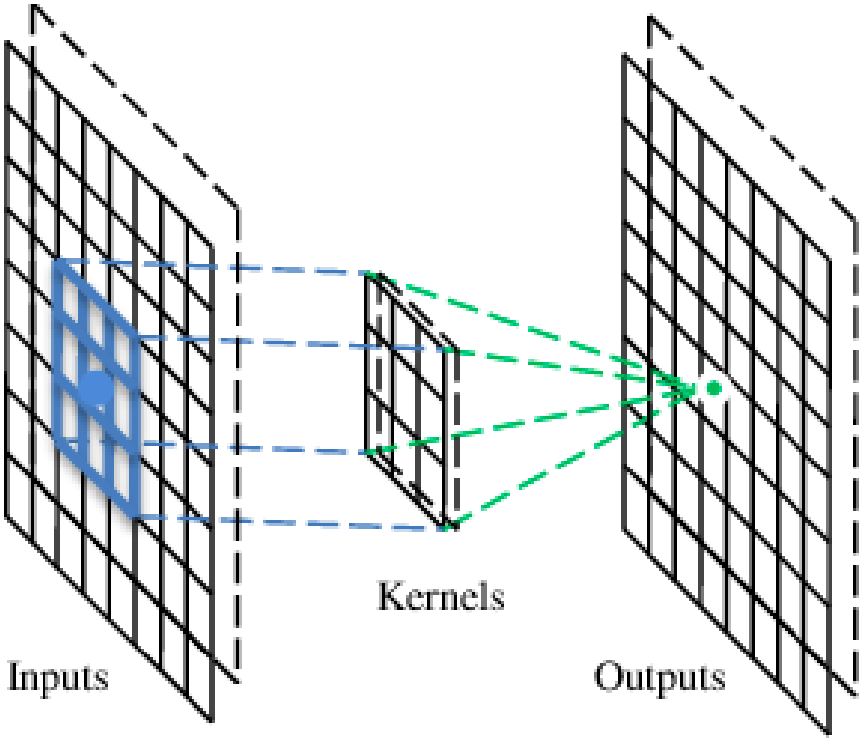
- Modify the pixels in an image base on some function of a local neighborhood of the pixels

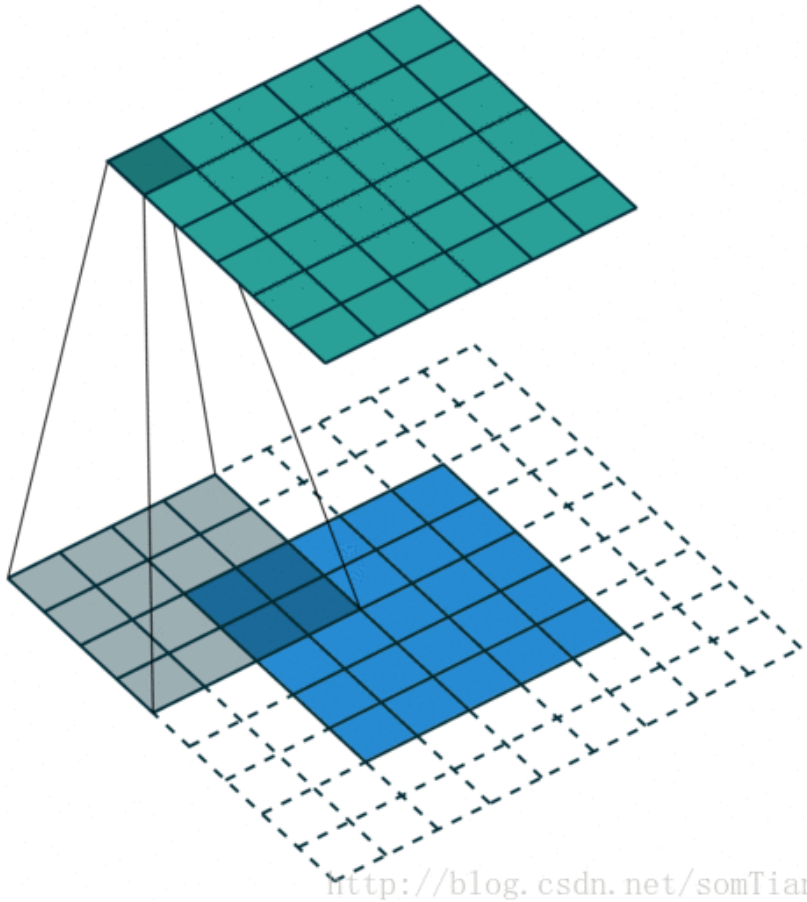


Credit: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

A Convolutional Layer

- A Convolutional layer has a number of filters/kernels, that perform convolution operation to find certain patterns.





Smoothing/Blur Filter

Image X

128	128	128	52	52	52
128	128	128	52	52	52
128	128	128	52	52	52
52	52	52	128	128	128
52	52	52	128	128	128
52	52	52	128	128	128

Filter W

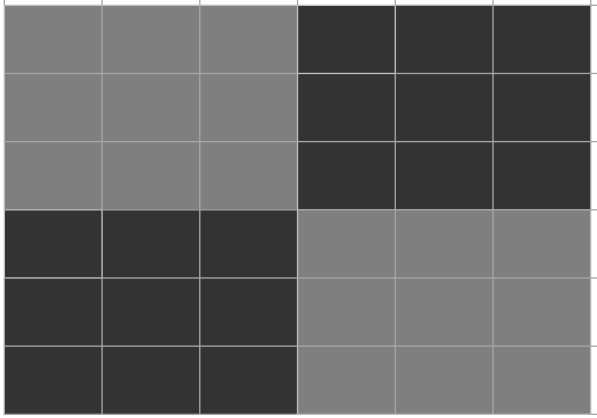
1	1	1
1	1	1
1	1	1

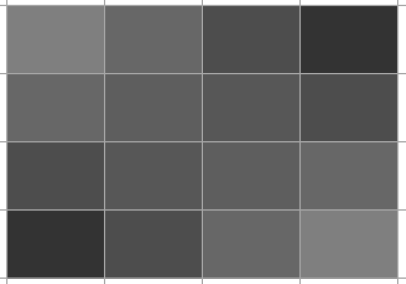
x1/9

Conv Output

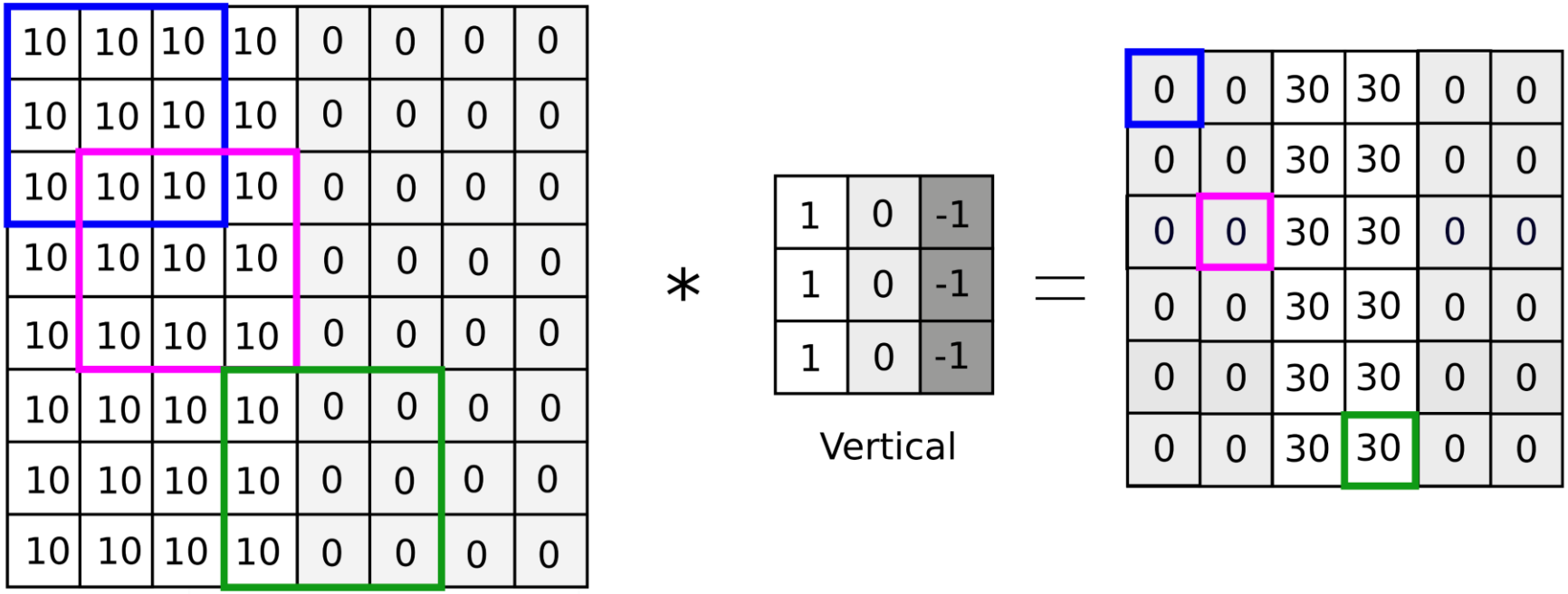
128	103	77	52
103	94	86	77
77	86	94	103
52	77	103	128

Padding on the input with zeros such a way that the spatial dimension of the input is not altered.



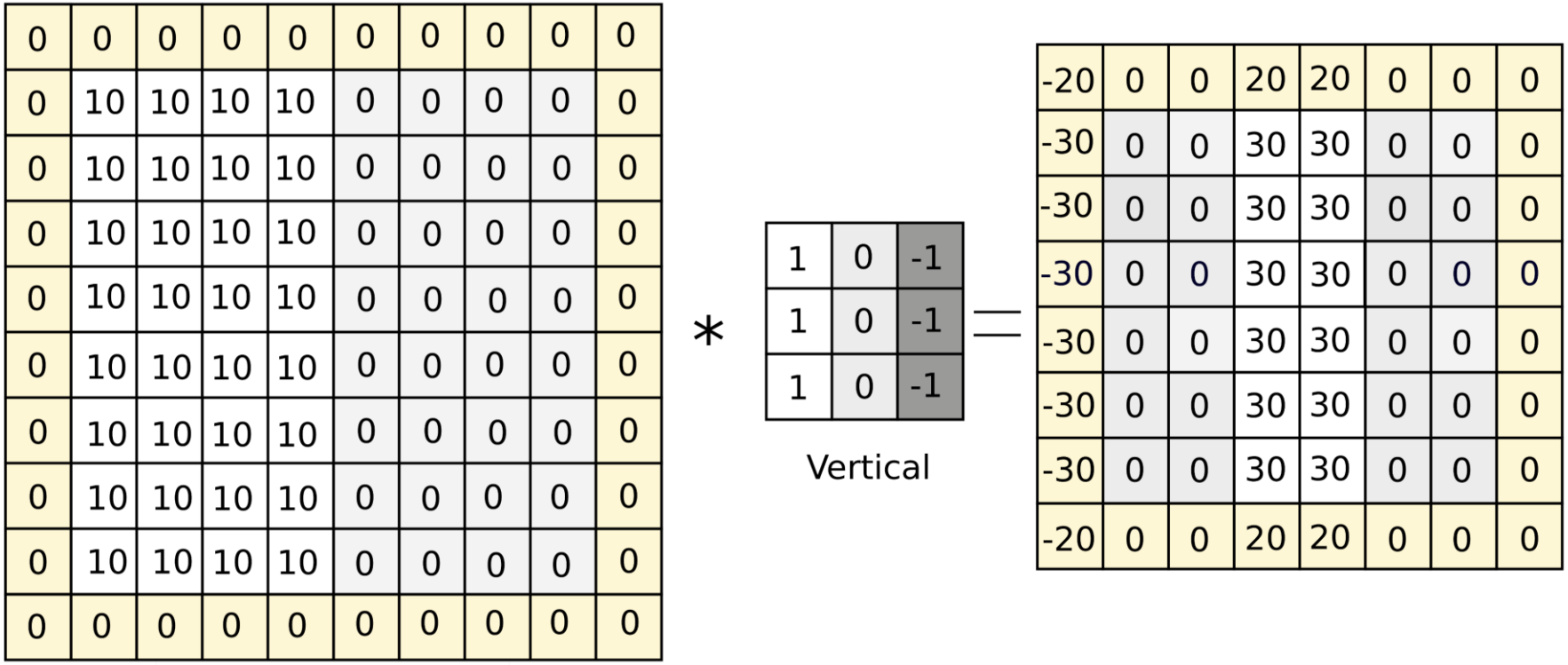


Vertical Edge Detection



Credit: https://anhreynolds.com/blogs/cnn.html

Padding



Credit: https://anhreynolds.com/blogs/cnn.html

Horizontal Edge Detection

- Just change the filter!

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

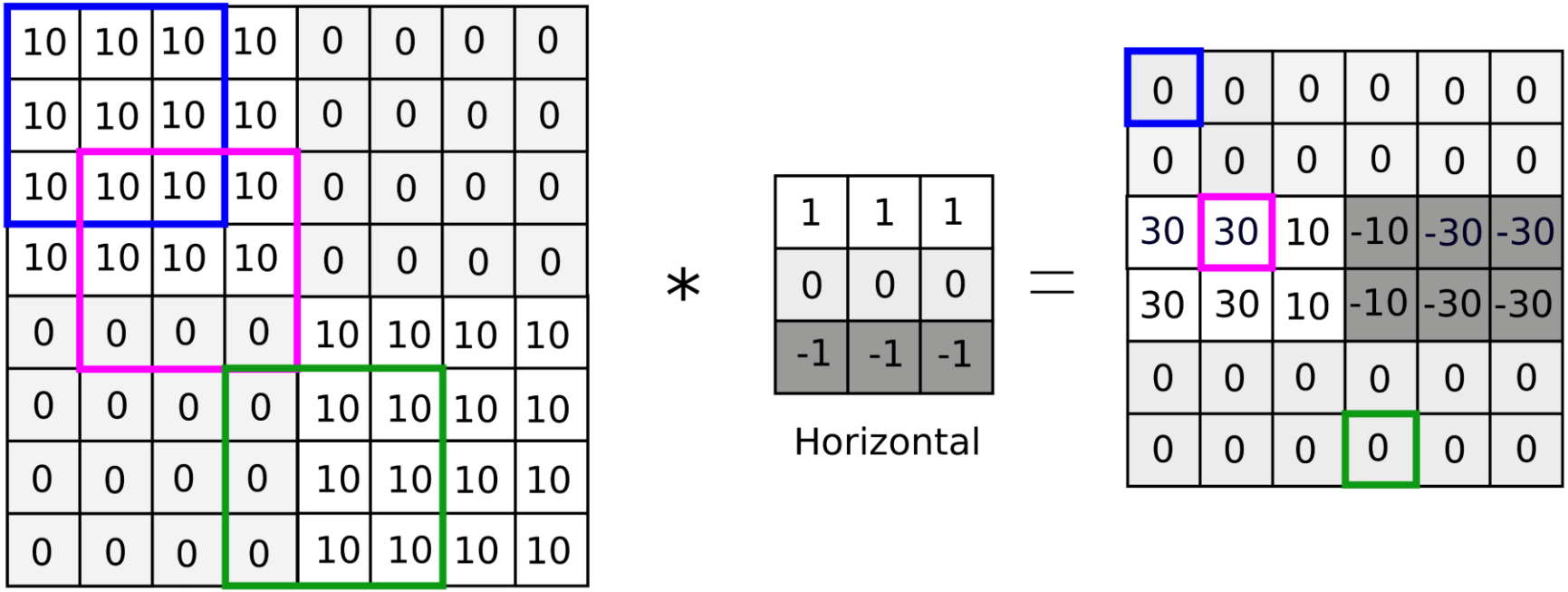
Vertical

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

Horizontal

Credit: https://datahacker.rs/edge-detection-extended/

Horizontal Edge Detection

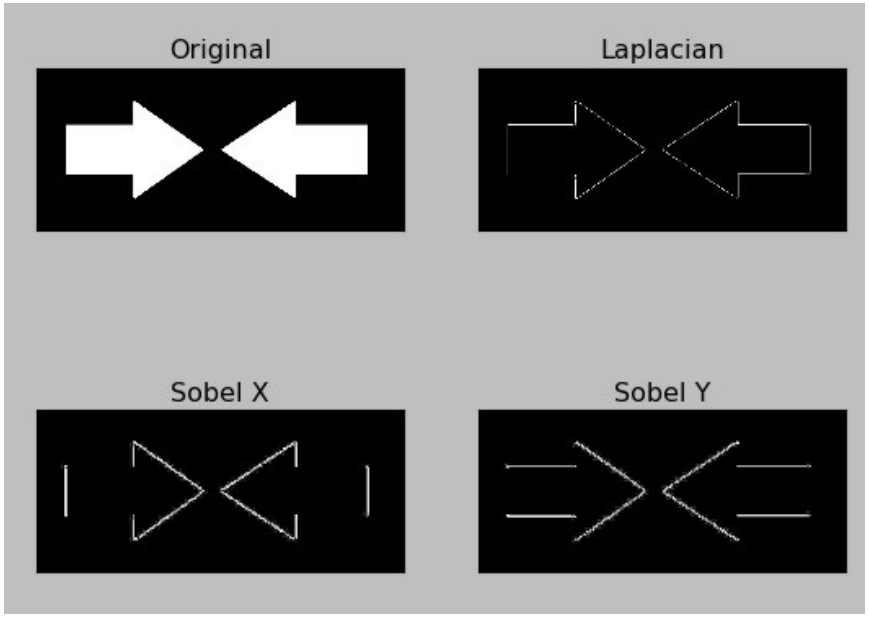


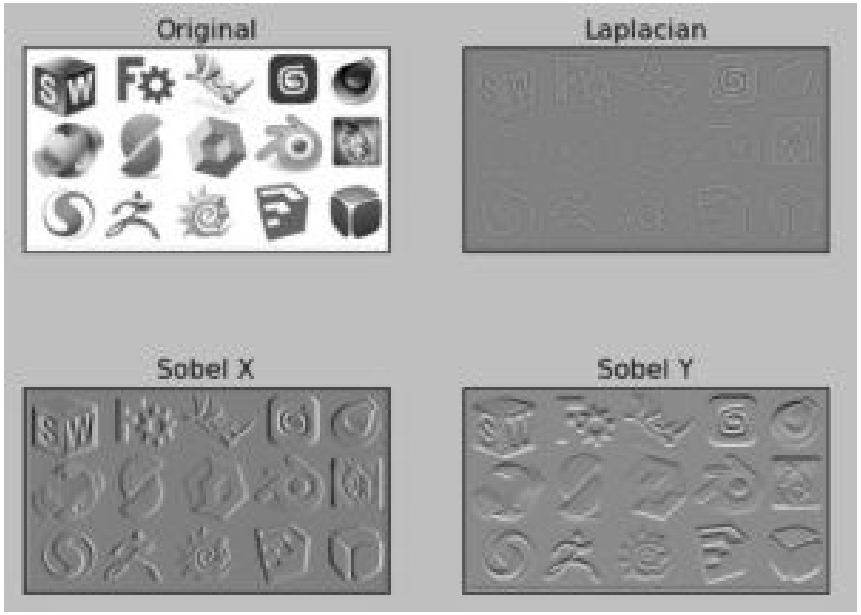
Credit: https://anhreynolds.com/blogs/cnn.html

Vertical edges

Horizontal edges

Credit: https://datahacker.rs/edge-detection/





Credit: https://www.meccanismocomplesso.org/en/opencv-python-edge-detection-and-image-gradient-analysis/

Kernel/Filter As A Weight Matrix

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

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

parameterized filter

Credit: https://anhreynolds.com/blogs/cnn.html

Book Reading

- See reference shared in the slides.