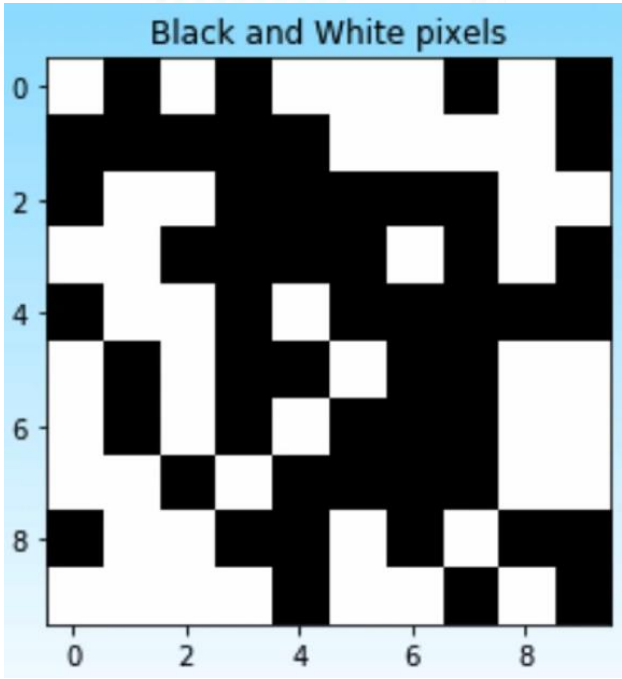
Deep Learning – Convolutional Neural Networks (CNN)

By,
Divesh R. Kubal
Data Scientist, eClerx Services
Center of Excellence – Machine Learning

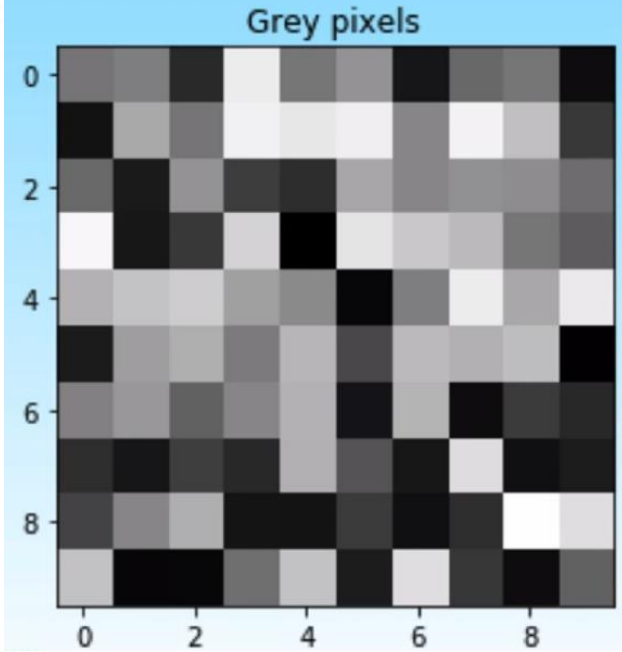
Table of Contents

Image Representation



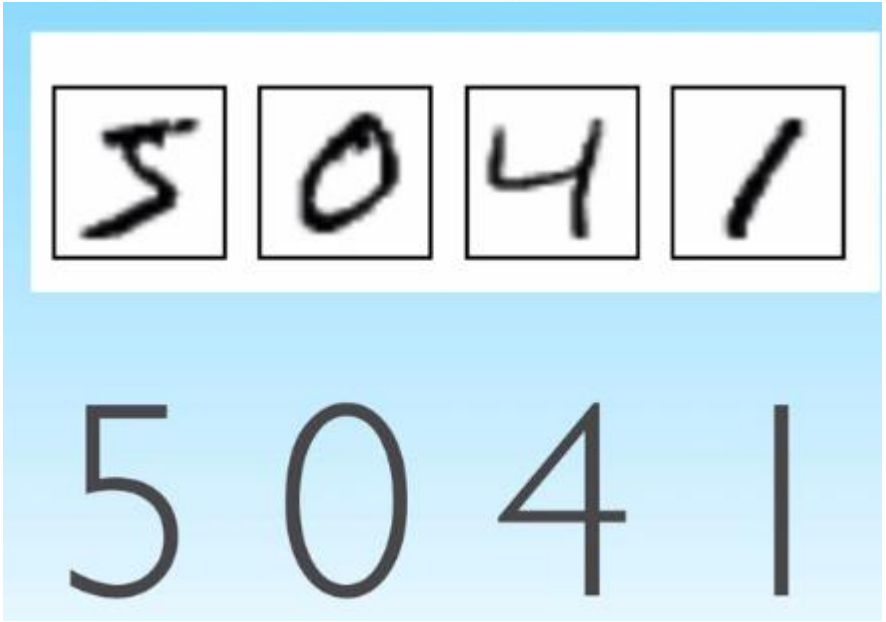

```
array([[1, 0, 1, 0, 1, 1, 1, 0, 1, 0],
       [0, 0, 0, 0, 0, 1, 1, 1, 1, 0],
       [0, 1, 1, 0, 0, 0, 0, 0, 1, 1],
       [1, 1, 0, 0, 0, 0, 1, 0, 1, 0],
       [0, 1, 1, 0, 1, 0, 0, 0, 0, 0],
       [1, 0, 1, 0, 0, 1, 0, 0, 1, 1],
       [1, 0, 1, 0, 1, 0, 0, 0, 1, 1],
       [1, 1, 0, 1, 0, 0, 0, 0, 1, 1],
       [0, 1, 1, 0, 0, 1, 0, 1, 0, 0],
       [1, 1, 1, 1, 0, 1, 1, 0, 1, 0]])
```

Grayscale

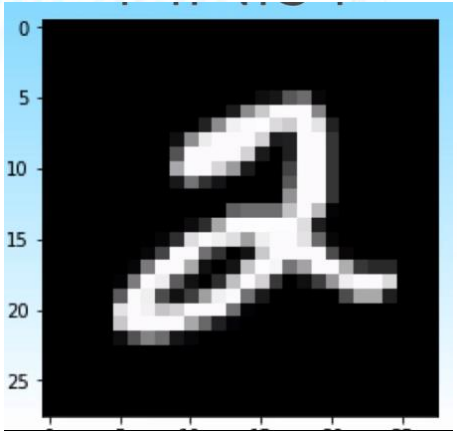


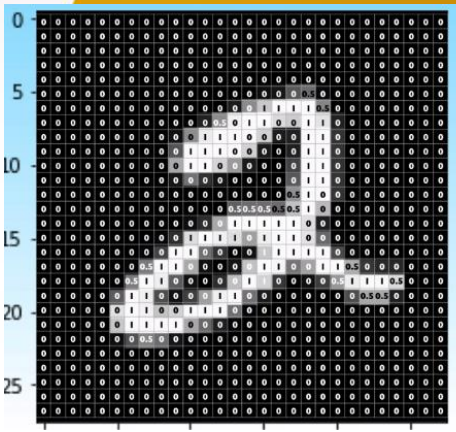

```
array([[127, 136,  48, 234, 128, 155,  24, 114, 128,  12],
       [ 18, 177, 127, 239, 230, 236, 145, 240, 195,  65],
       [116,  29, 155,  70,  51, 174, 144, 153, 149, 120],
       [243,  26,  66, 213,   1, 227, 203, 191, 128, 105],
       [183, 199, 207, 167, 146,   7, 136, 234, 175, 232],
       [ 31, 166, 180, 133, 187,  82, 192, 182, 194,   2],
       [139, 161, 108, 144, 183,  23, 185,  12,  67,  47],
       [ 53,  24,  72,  46, 182,  94,  25, 221,  16,  34],
       [ 76, 144, 181,  22,  21,  67,  16,  53, 250, 222],
       [197,   6,   7, 122, 198,  34, 222,  64,  12, 108]])
```

MNIST Dataset

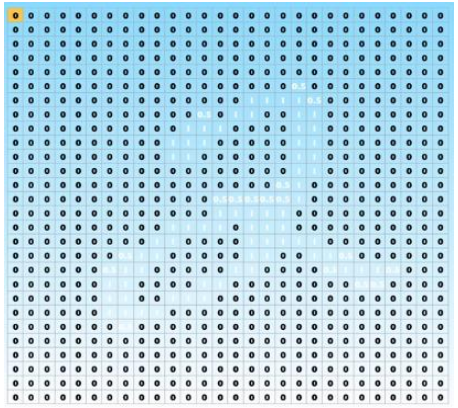


MNIST Sample Image



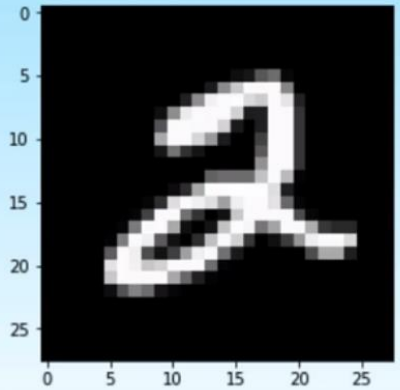


28 x 28 = 784



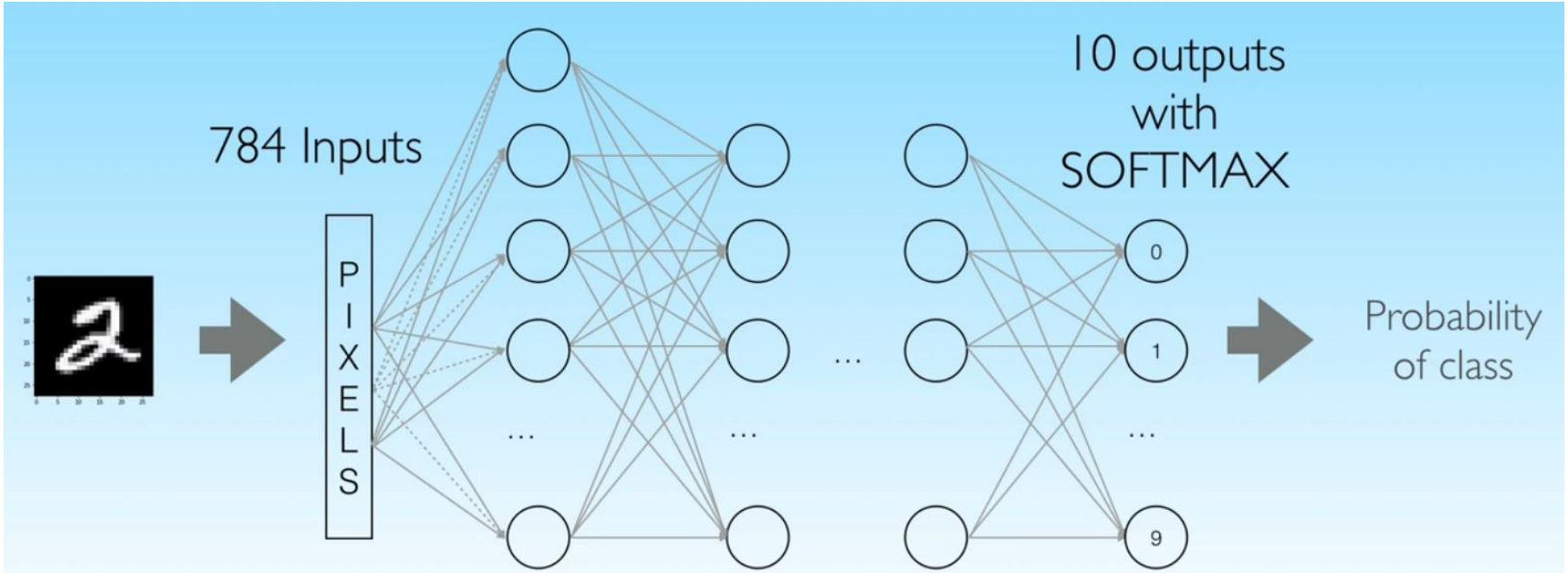
Images unrolled as long vector

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 .5 .5 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

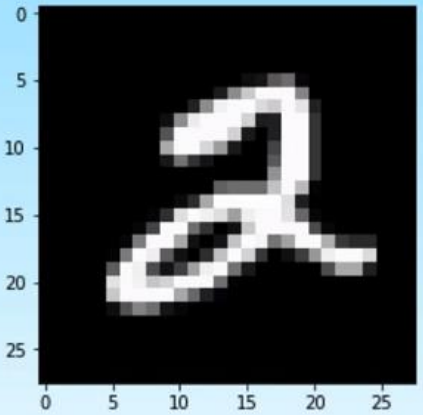


28 x 28 image => 784 input pixels array

Fully Connected Neural Network Architecture

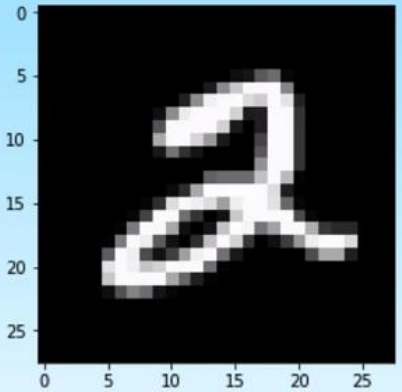


Features



Pixels Vector

Feature Vector



Feature Vector

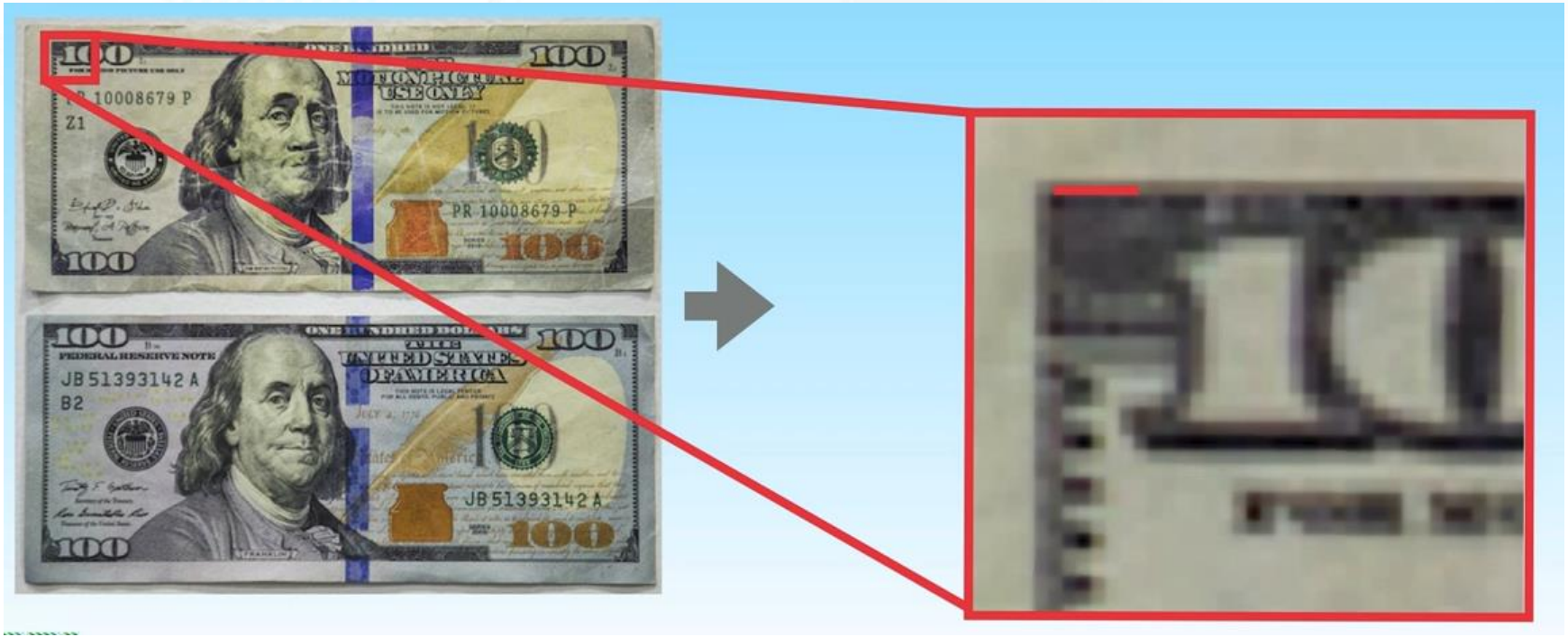
- Fourier coefficients
- Wavelets
- Histogram of Oriented Gradients (HOG)
- Speeded Up Robust Features (SURF)
- Local Binary Patterns (LBP)
- Color histograms
-

Hand Engineering Feature

Feature Vector

- Fourier coefficients
- Wavelets
- Histogram of Oriented Gradients (HOG)
- Speeded Up Robust Features (SURF)
- Local Binary Patterns (LBP)
- Color histograms
-

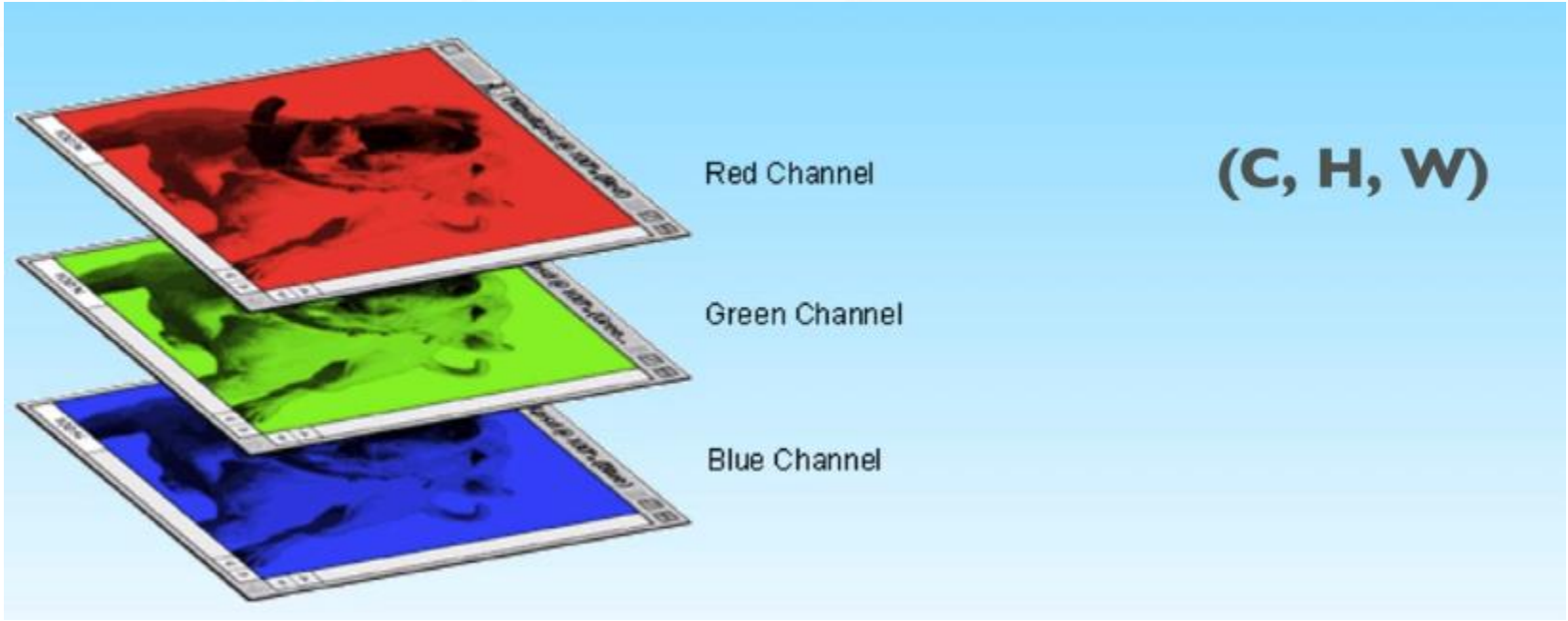
Local Patterns



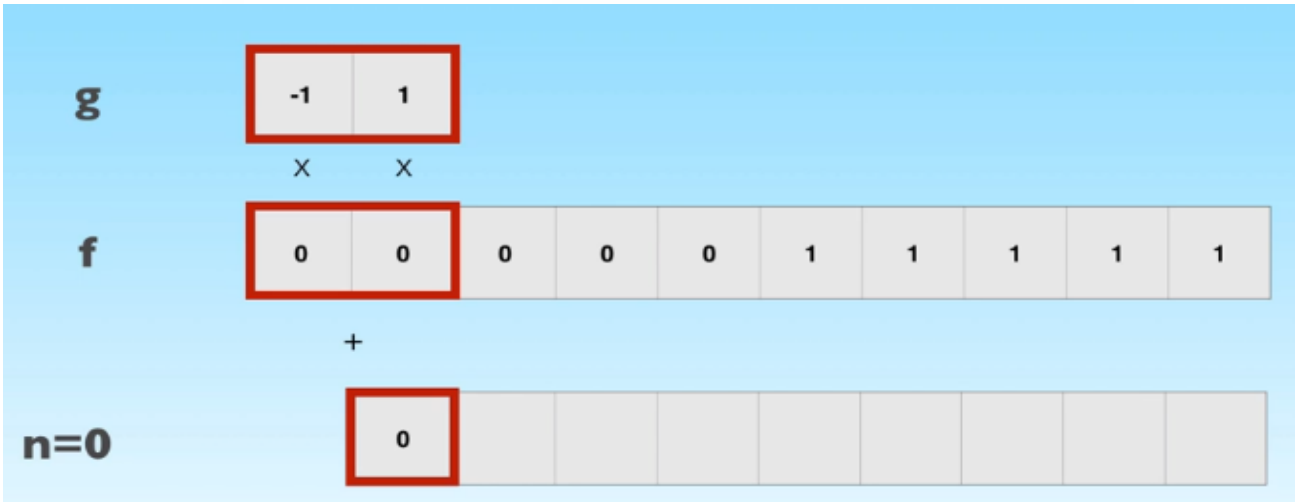
Tensors

Order	Name	Example	Shape
0	Scalar	3	no shape
1	Vector	[4, 5, 0, 3, 1, 4, 5]	(7,)
2	Matrix	[[0, 1, 0], [5, 0, 2]]	(2, 3)
3	Tensor	[[[0, 1, 0, 5], [5, 0, 2, 6]], [[1, 2, 4, 4], [8, 3, 1, 9]]]	(2, 2, 4)

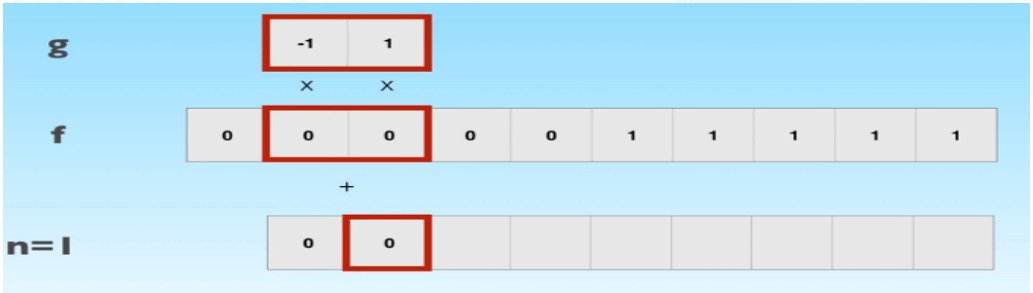
Colored Images Representation

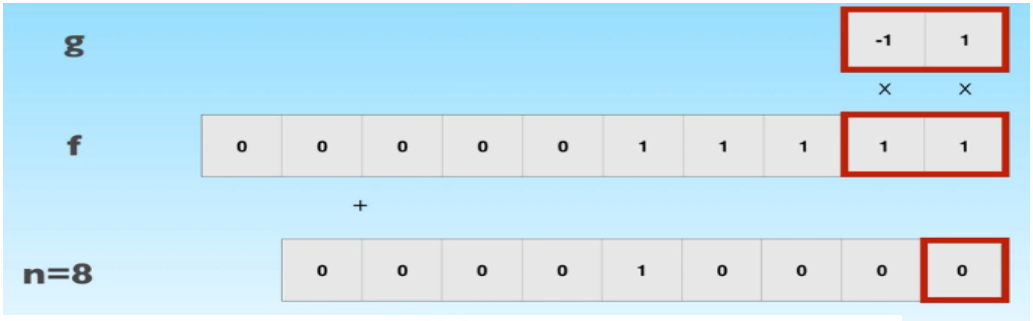


Discrete Convolution



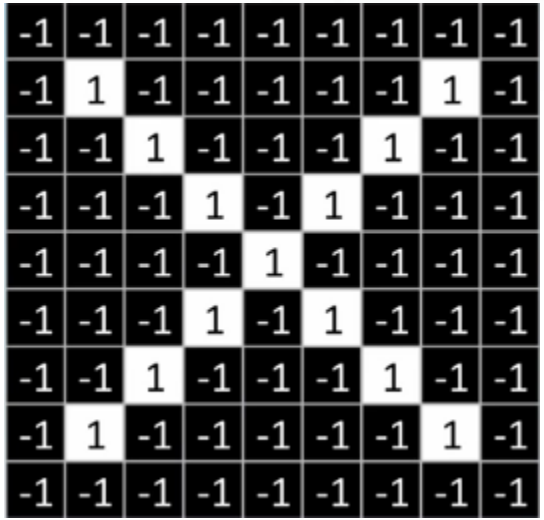
Discrete Convolution



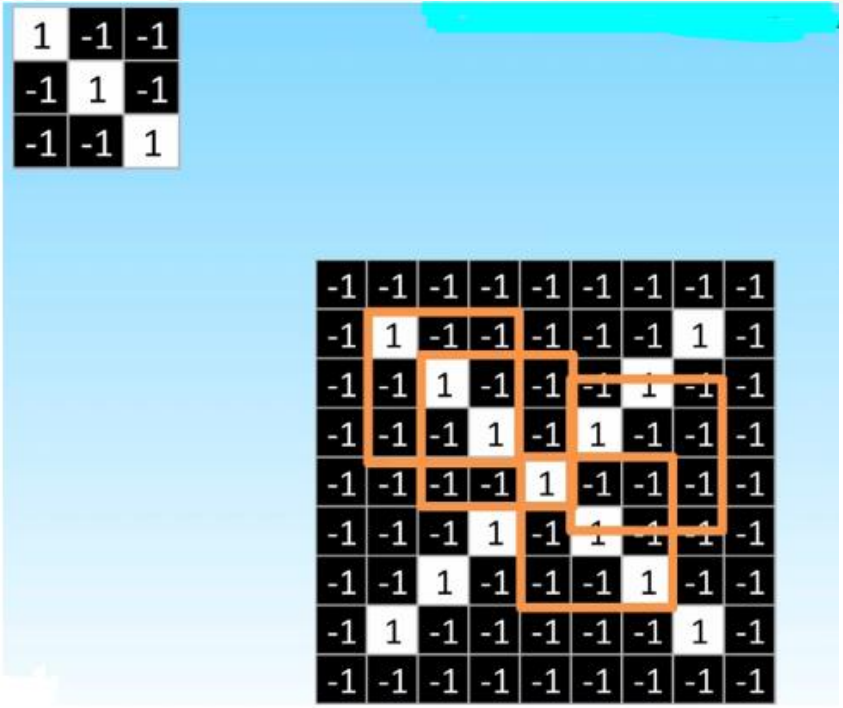


$$(f * g)[n] = \sum_{m=-M}^{M} f[n-m]g[m]$$

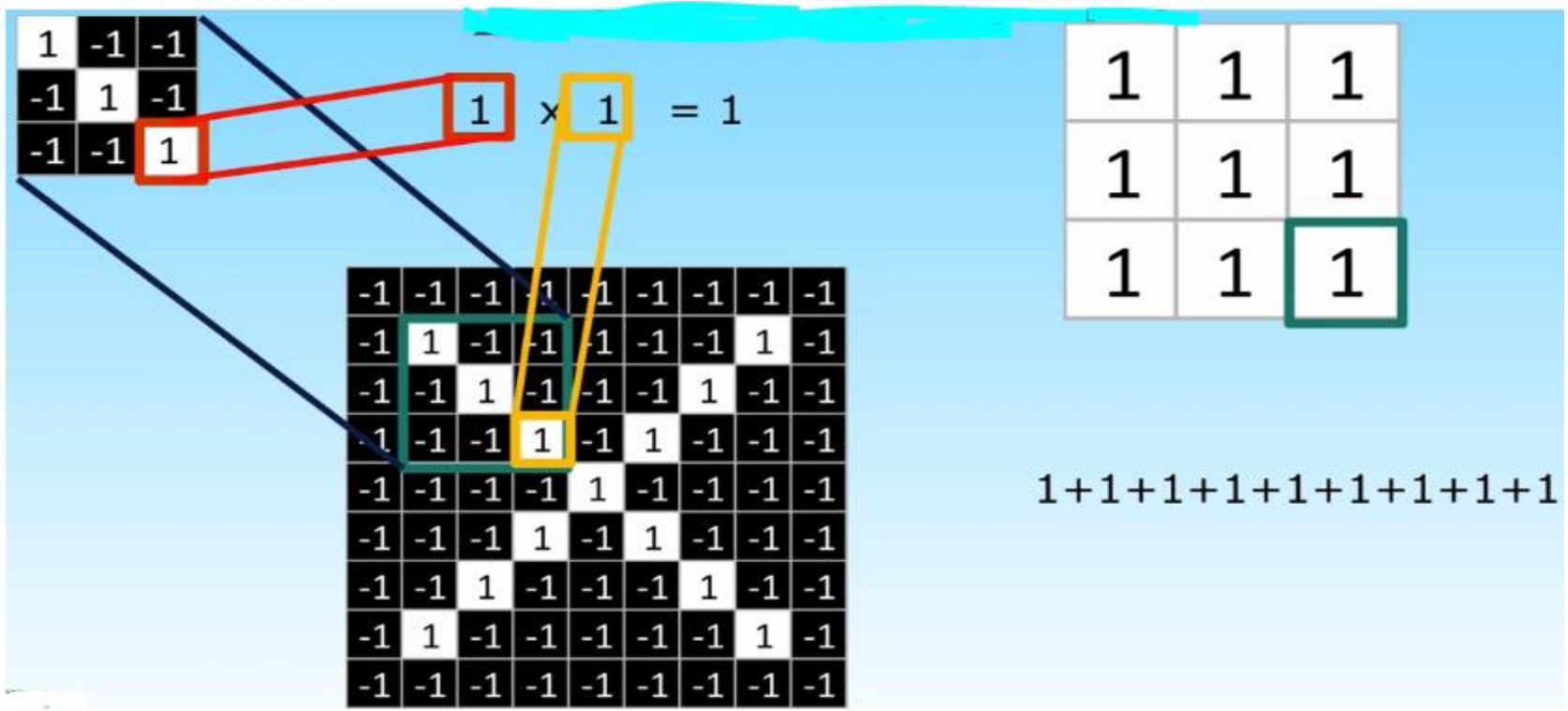
2D Convolution



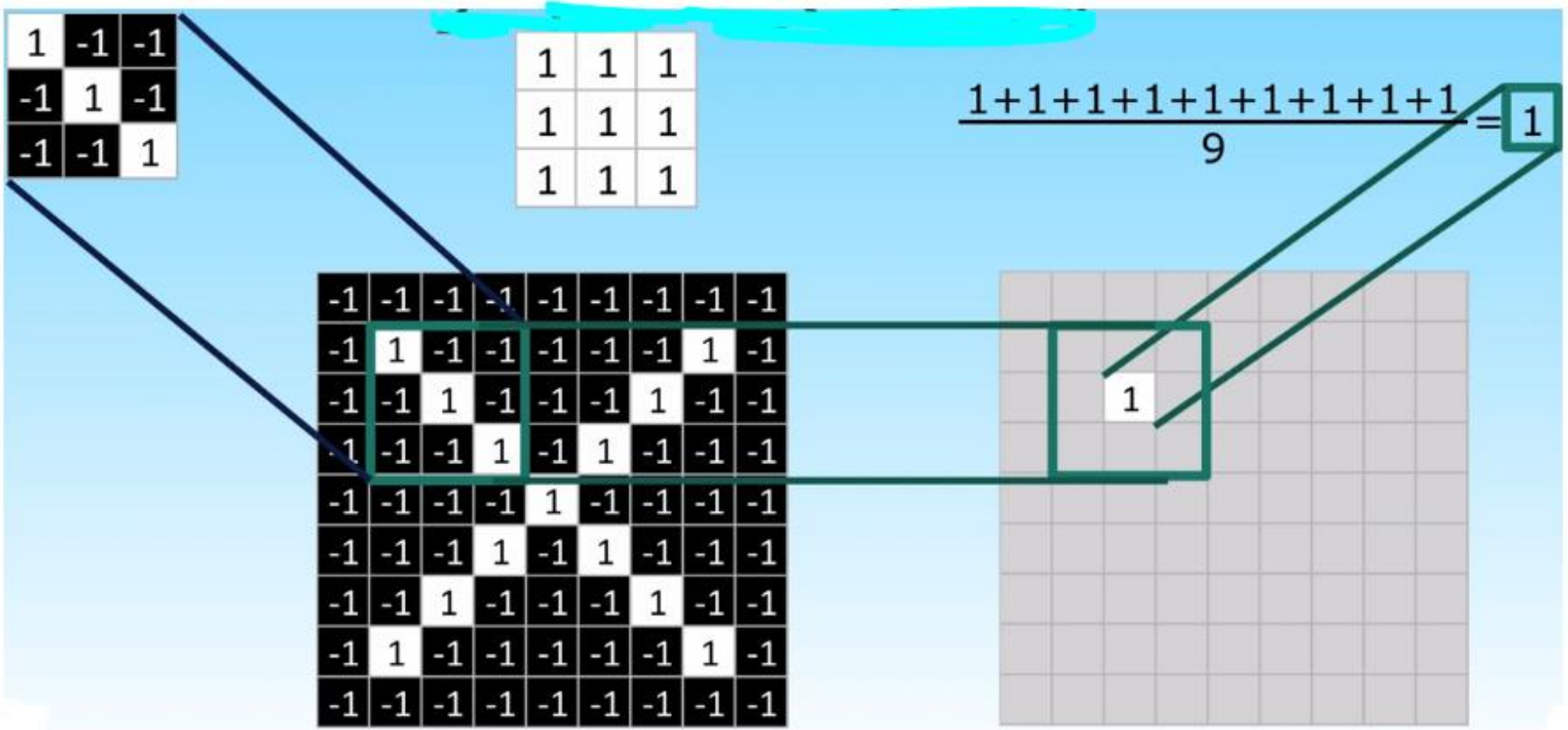
2D Convolution



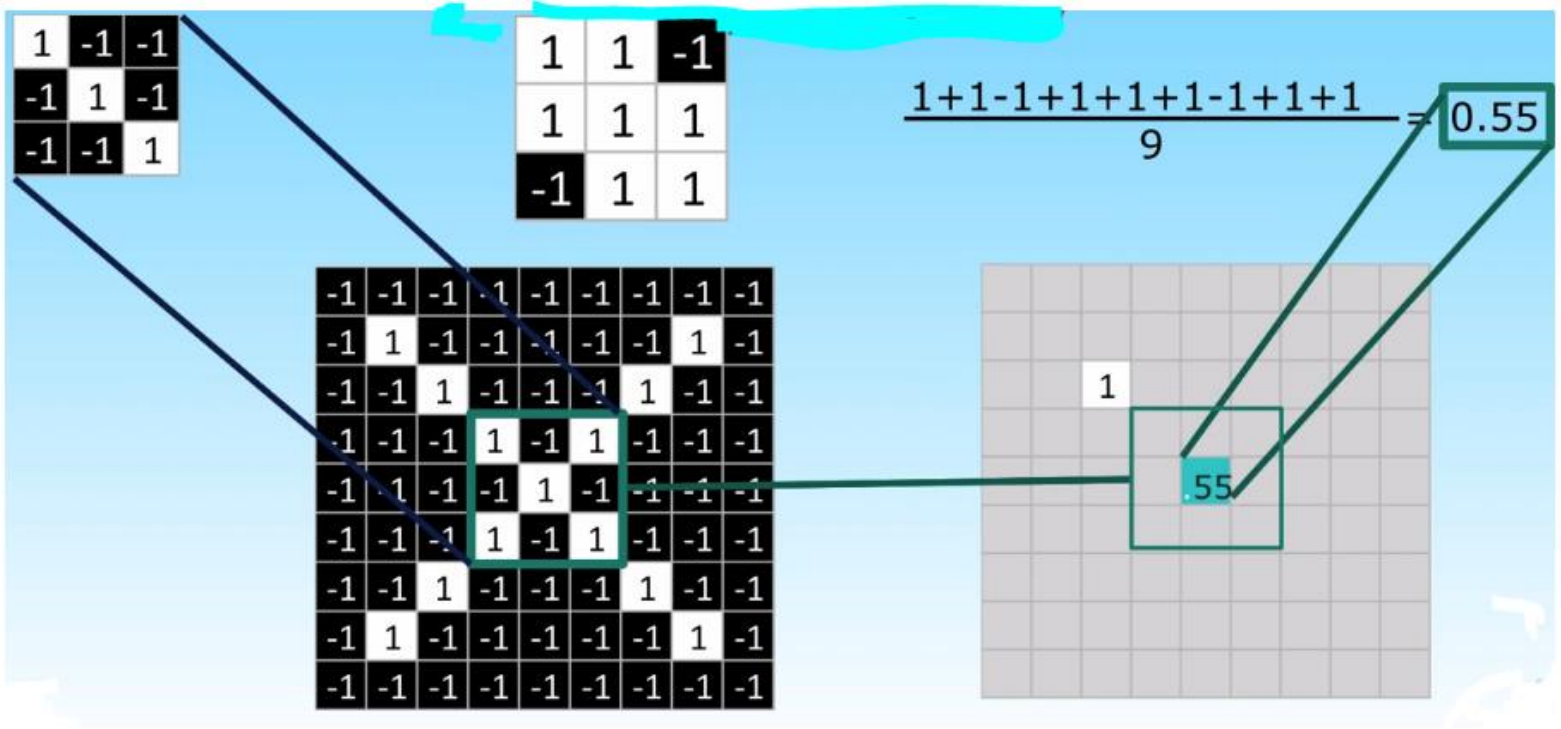
2D Convolution



2D Convolution

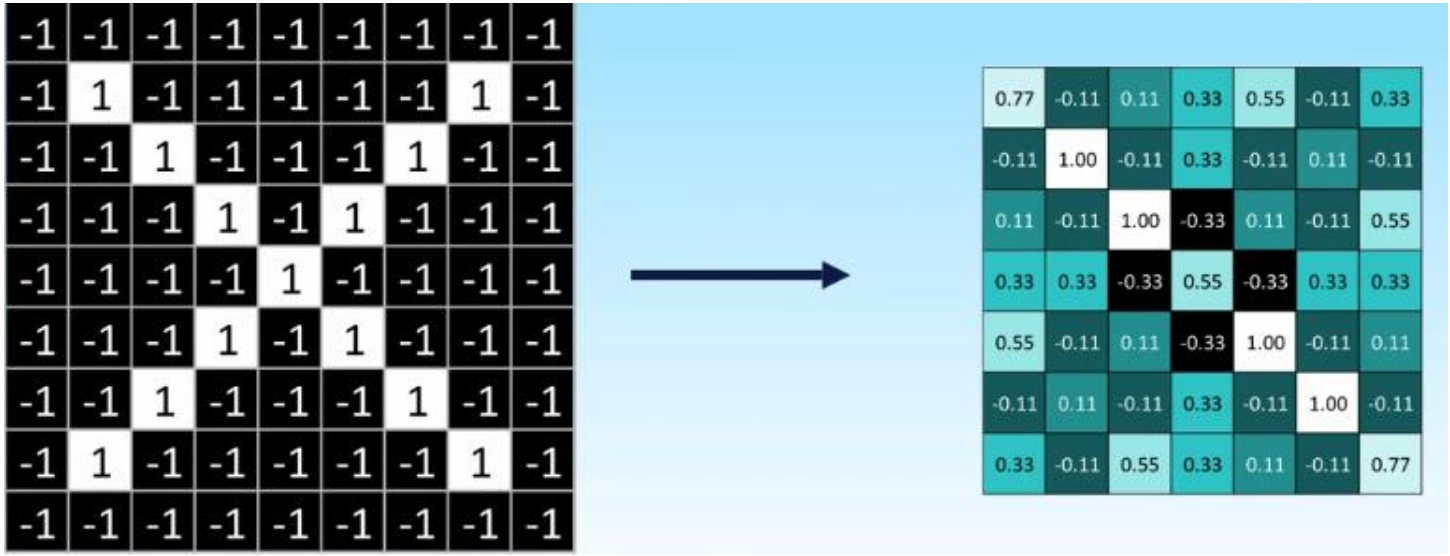


2D Convolution

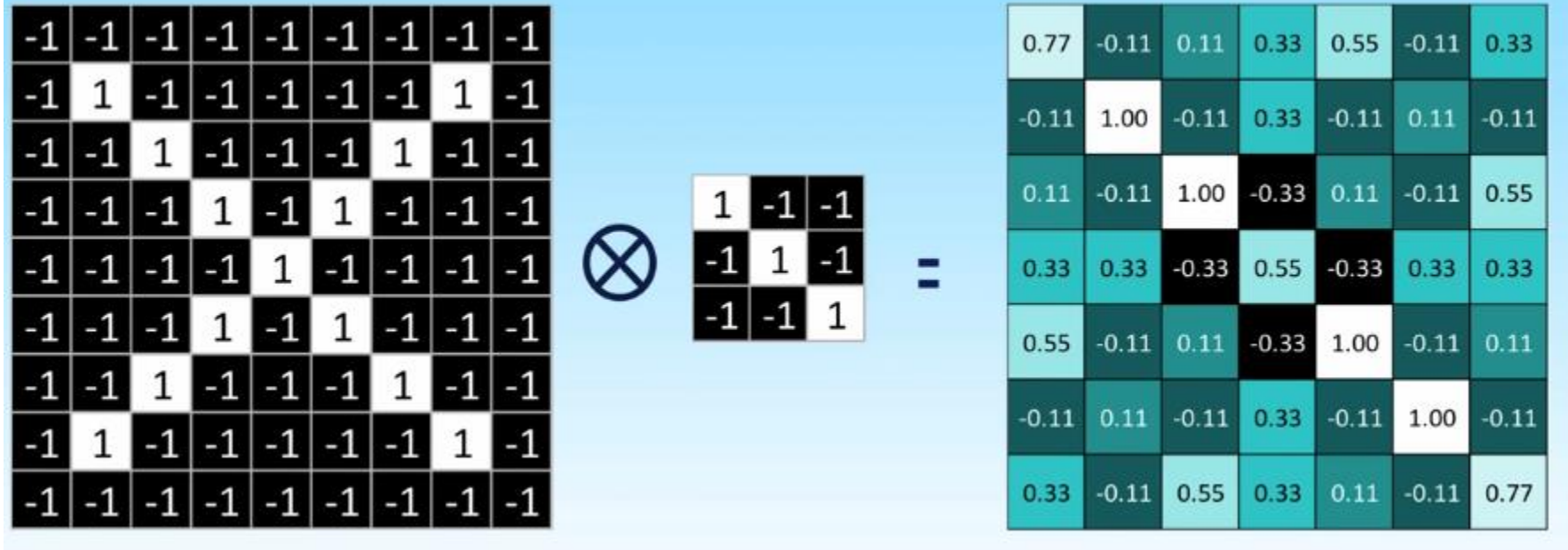


2D Convolution

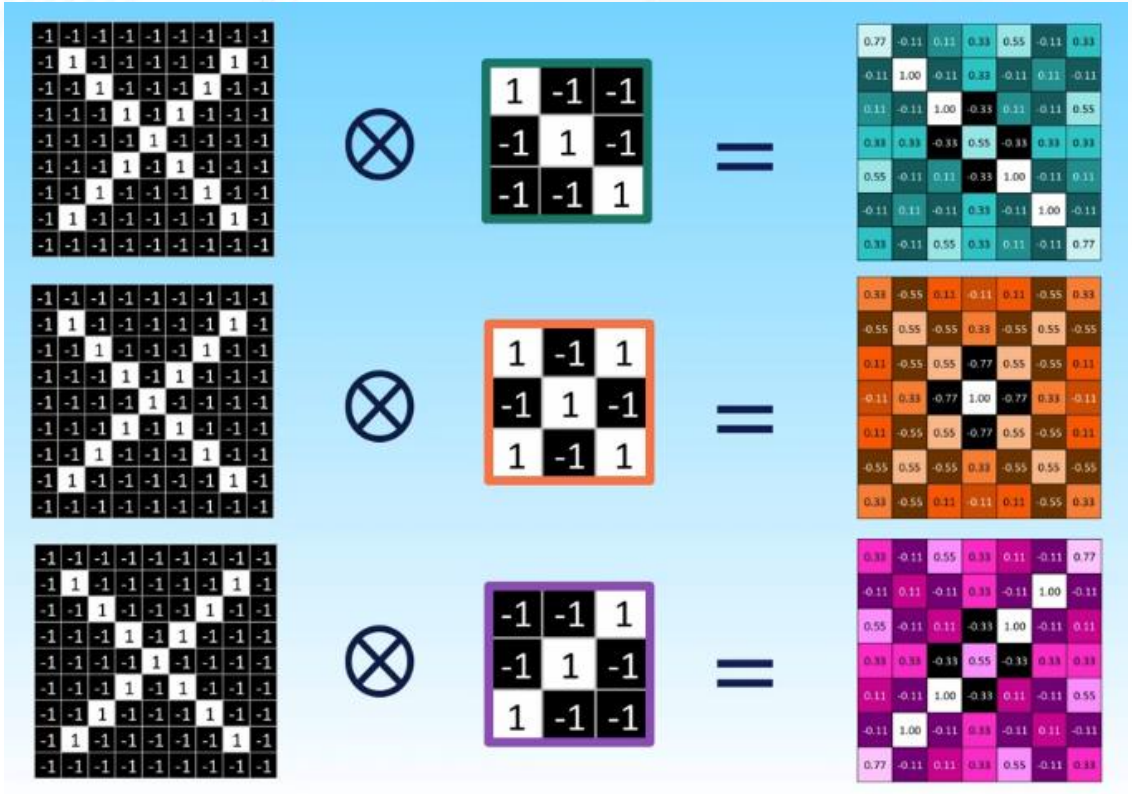
1	-1	-1
-1	1	-1
-1	-1	1



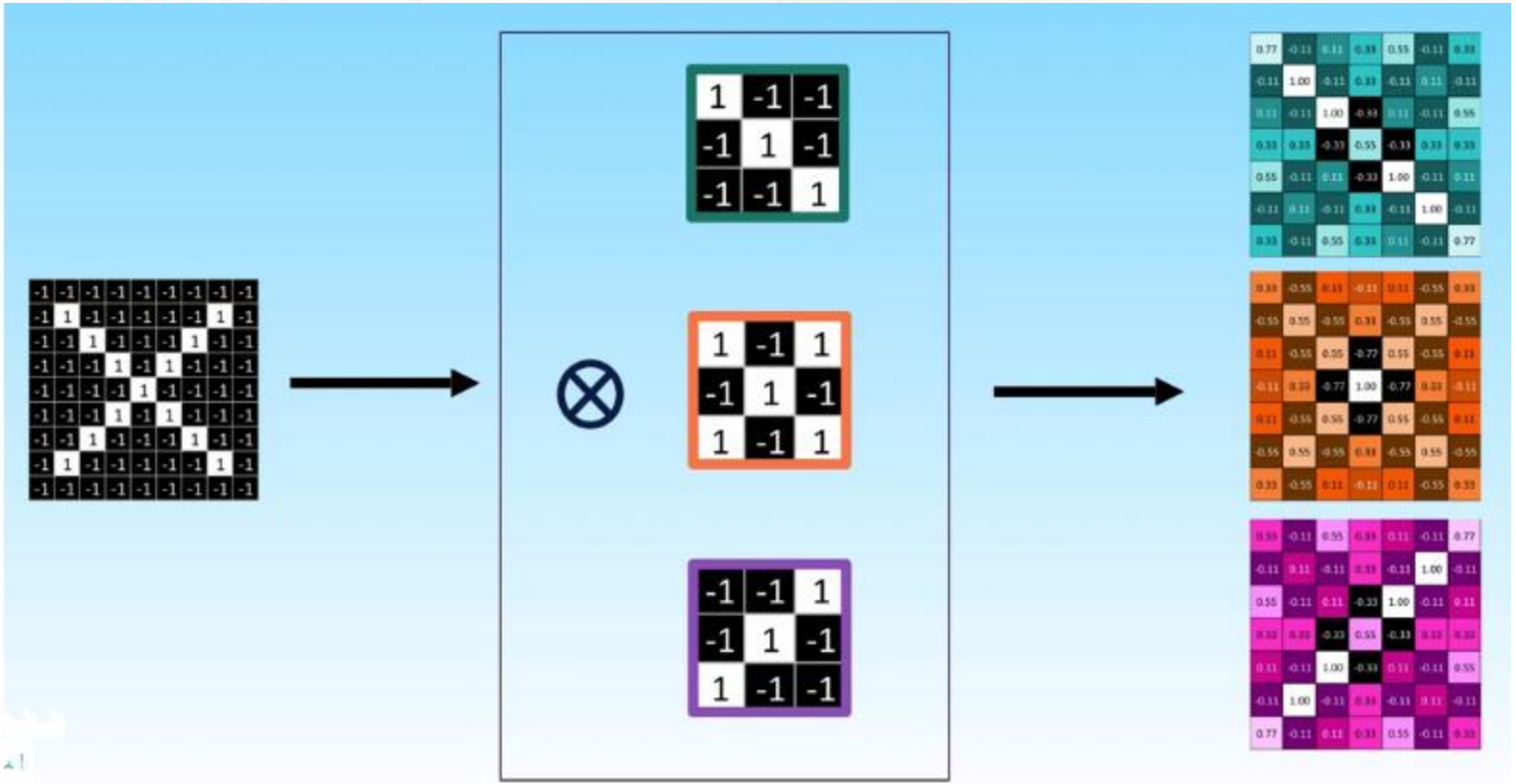
2D Convolution

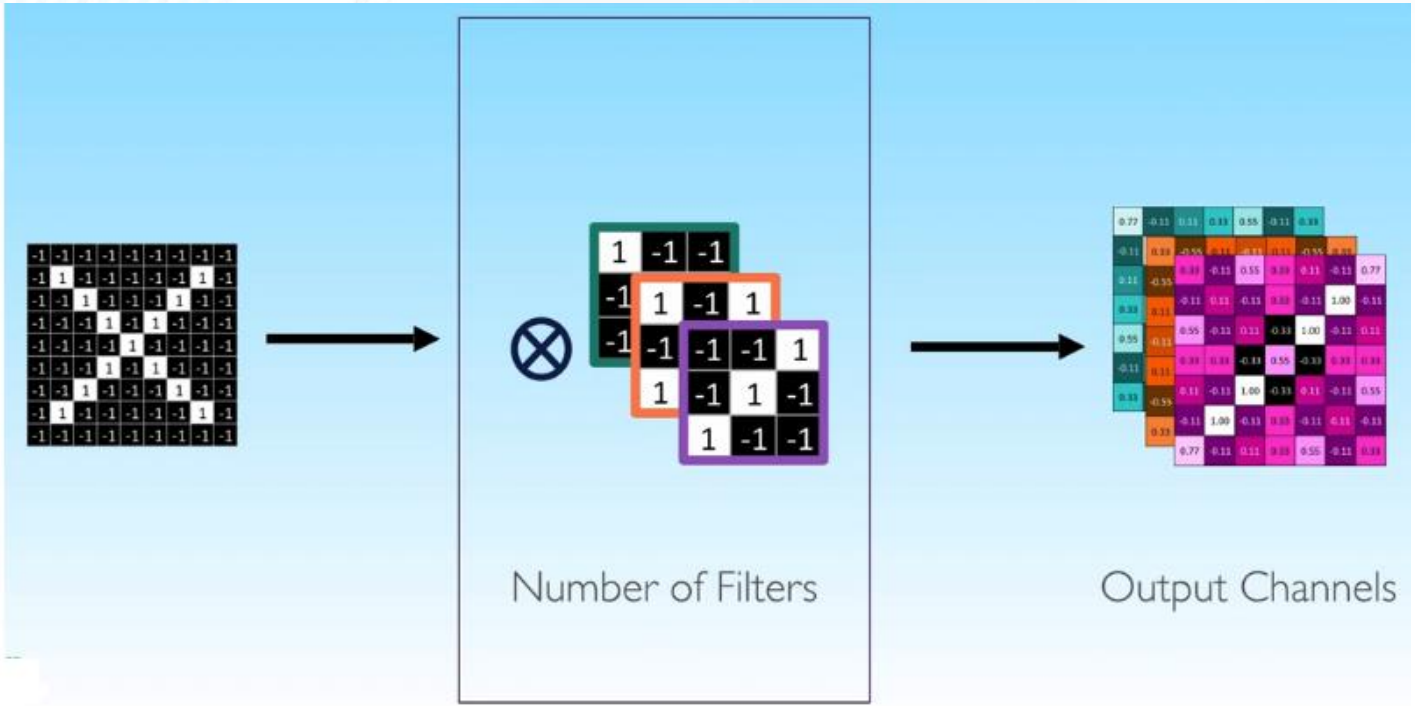


2D Convolution



Convolutional Layer




Number of Filters
Output Channels

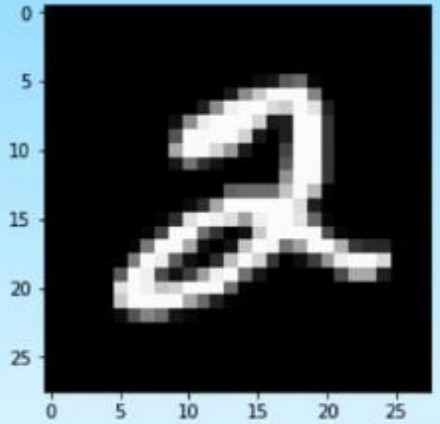
Input Tensor

Input: order 4 tensor

(N, H, W, C)

N: Number of images
H: Height of image
W: Width of image
C: Number of color channels

MNIST training set

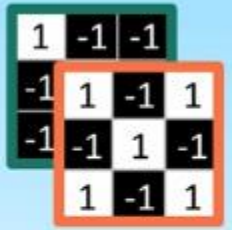


(60000, 28, 28, 1)

Conv Layer Tensor

CONV: order 4 tensor

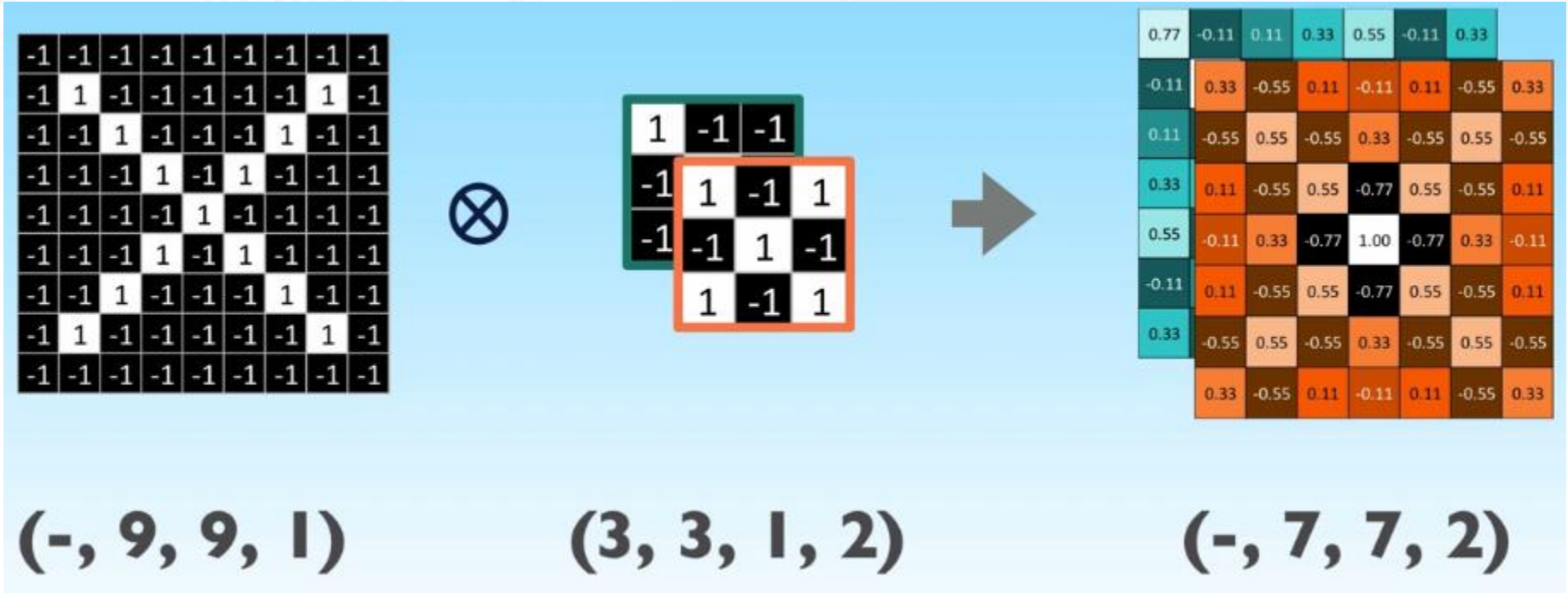
$$(H_f, W_f, C_i, C_o)$$



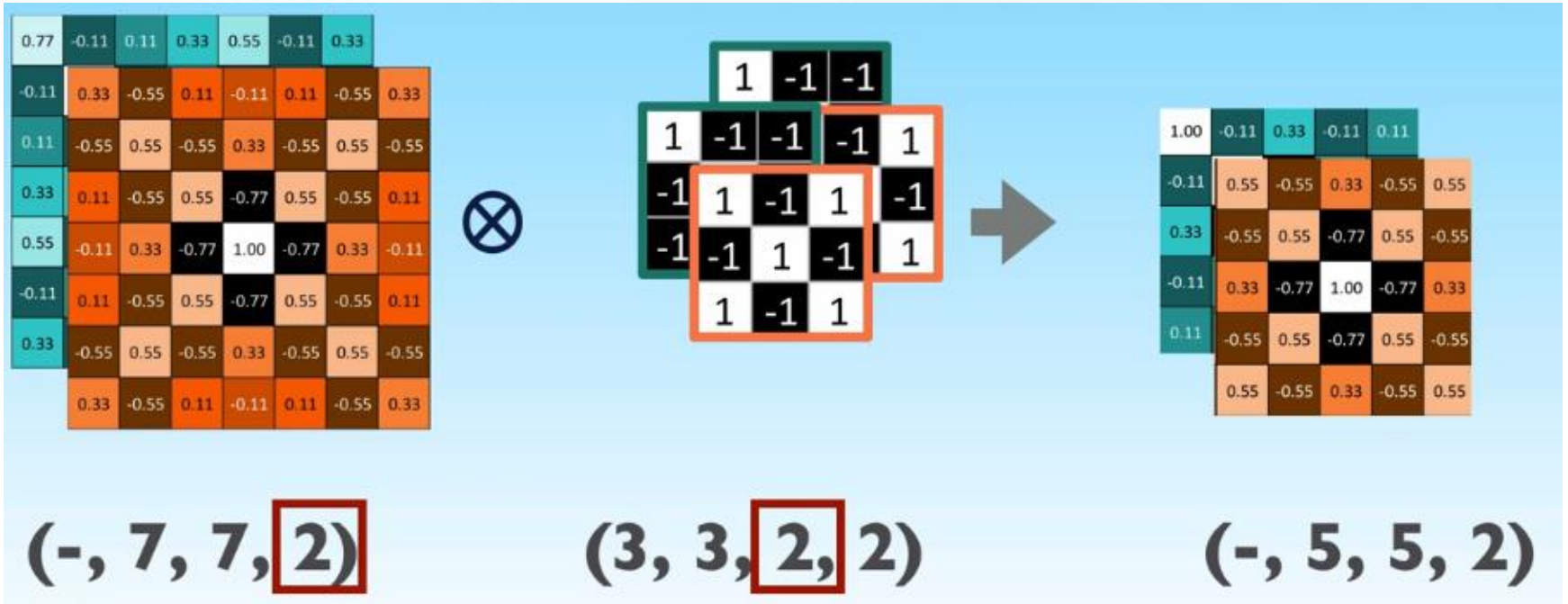
H_f: Height of filter patch
W_f: Width of filter patch
C_i: Channels in input
C_o: Channels in output (# filters)

$$(3, 3, 1, 2)$$

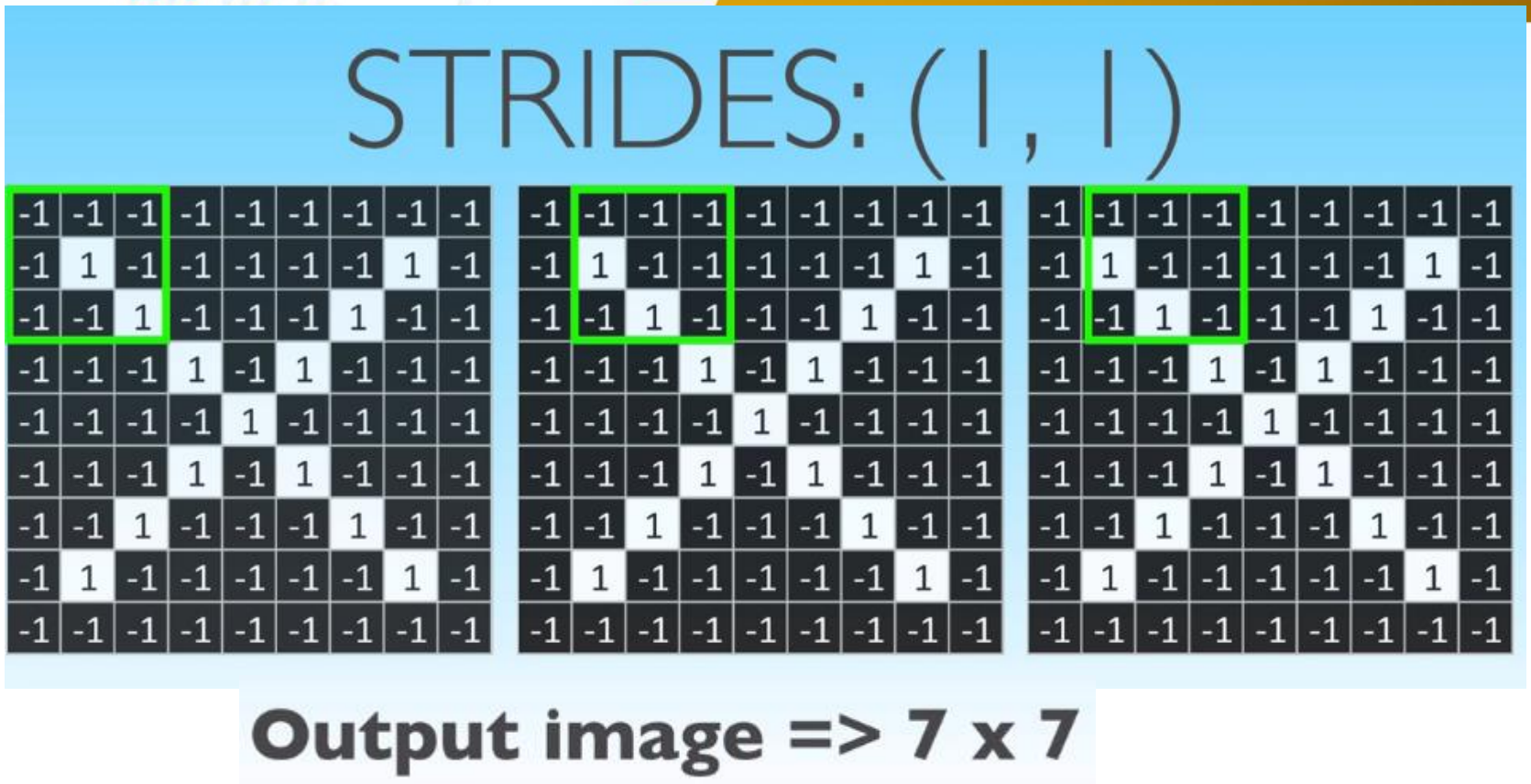
Example 1


-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1 1
-1 -1 1 -1
1 -1 1
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
0.11 -0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
0.55 -0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
-0.11 0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
0.33 -0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
(-, 9, 9, 1)
(3, 3, 1, 2)
(-, 7, 7, 2)

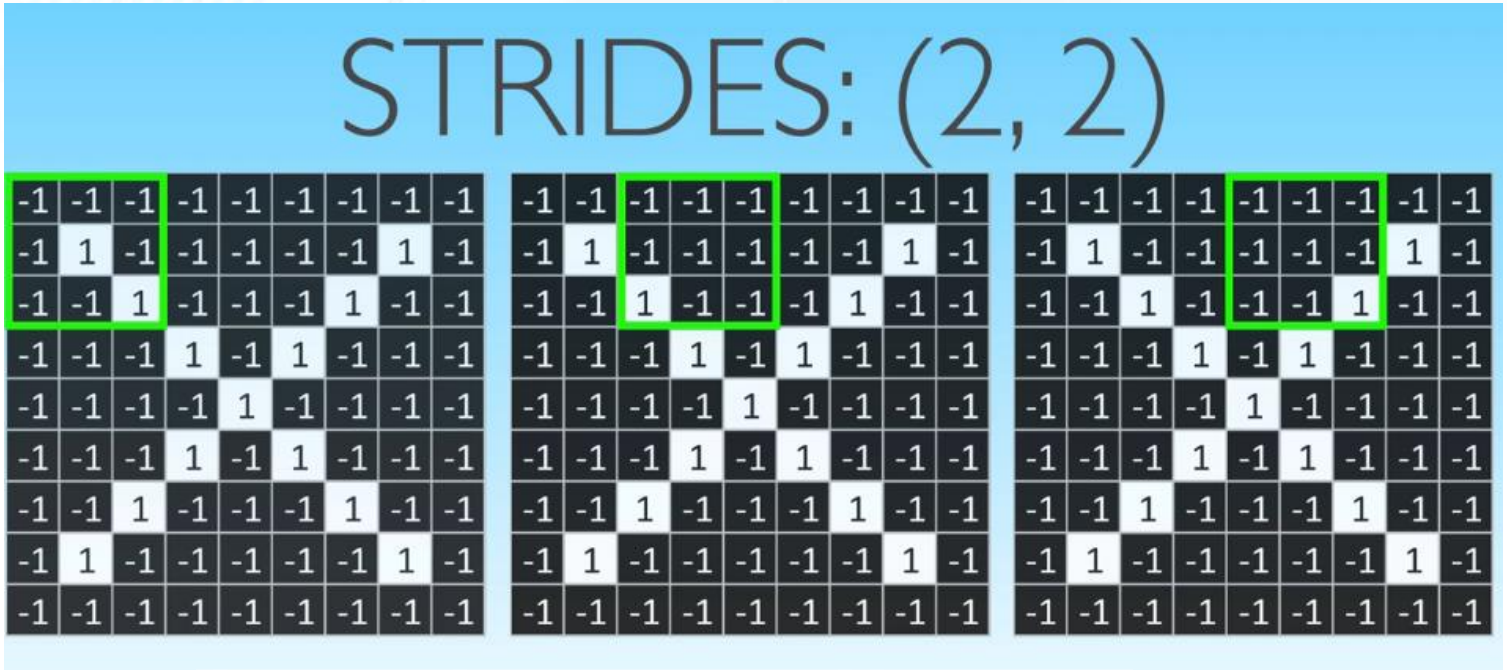
Example 1


0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
0.11 -0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
0.55 -0.11 0.33 -0.77 1.00 -0.77 0.33 -0.11
-0.11 0.11 -0.55 0.55 -0.77 0.55 -0.55 0.11
0.33 -0.55 0.55 -0.55 0.33 -0.55 0.55 -0.55
0.33 -0.55 0.11 -0.11 0.11 -0.55 0.33
1 -1 -1
1 -1 -1 -1 1
-1 1 -1 1 -1
-1 -1 1 -1 1
1 -1 1
1.00 -0.11 0.33 -0.11 0.11
-0.11 0.55 -0.55 0.33 -0.55 0.55
0.33 -0.55 0.55 -0.77 0.55 -0.55
-0.11 0.33 -0.77 1.00 -0.77 0.33
0.11 -0.55 0.55 -0.77 0.55 -0.55
0.55 -0.55 0.33 -0.55 0.55
(-, 7, 7, 2)
(3, 3, 2, 2)
(-, 5, 5, 2)

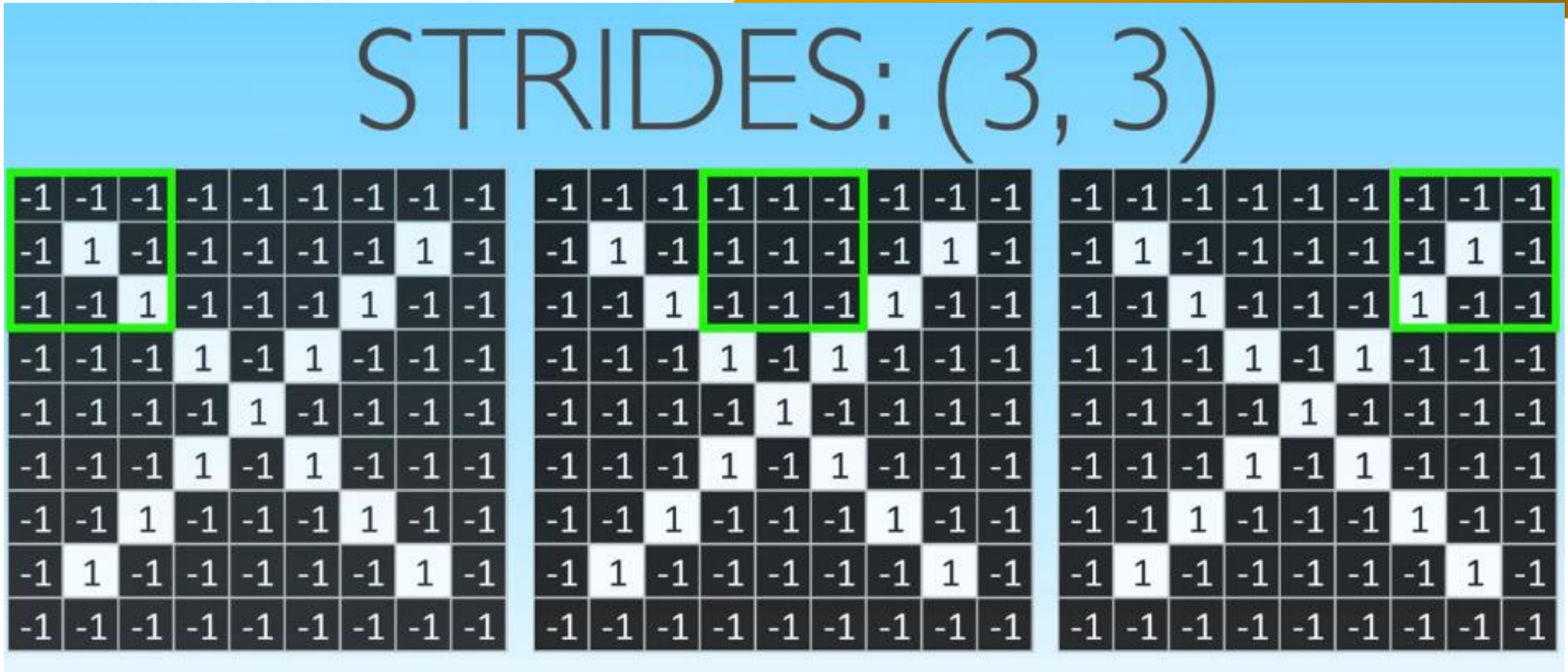
Strides



Strides

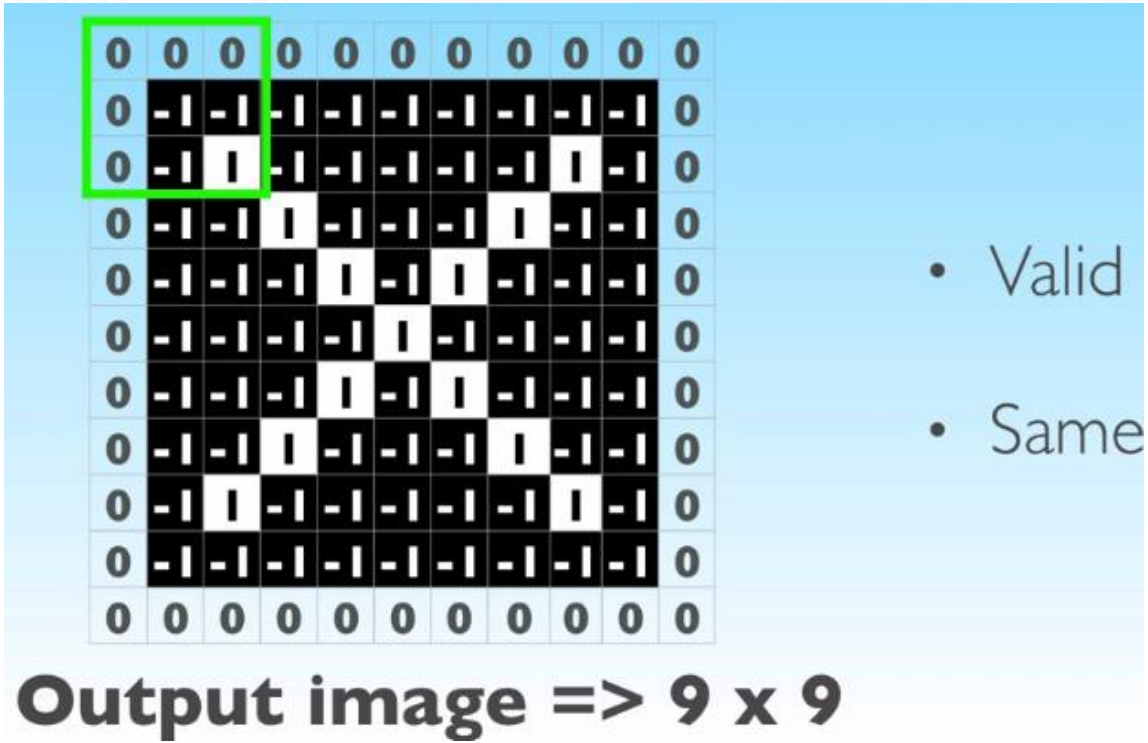


Strides



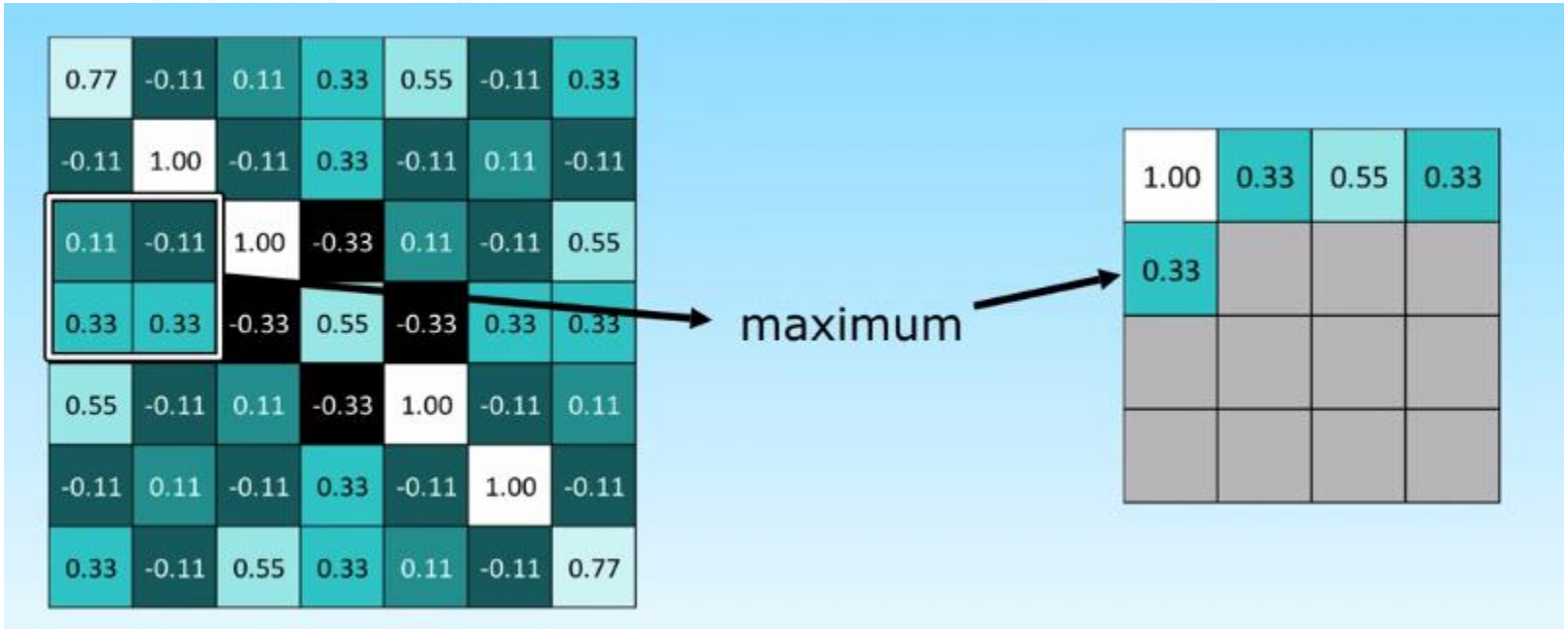
Output image => 3 x 3

Padding Options

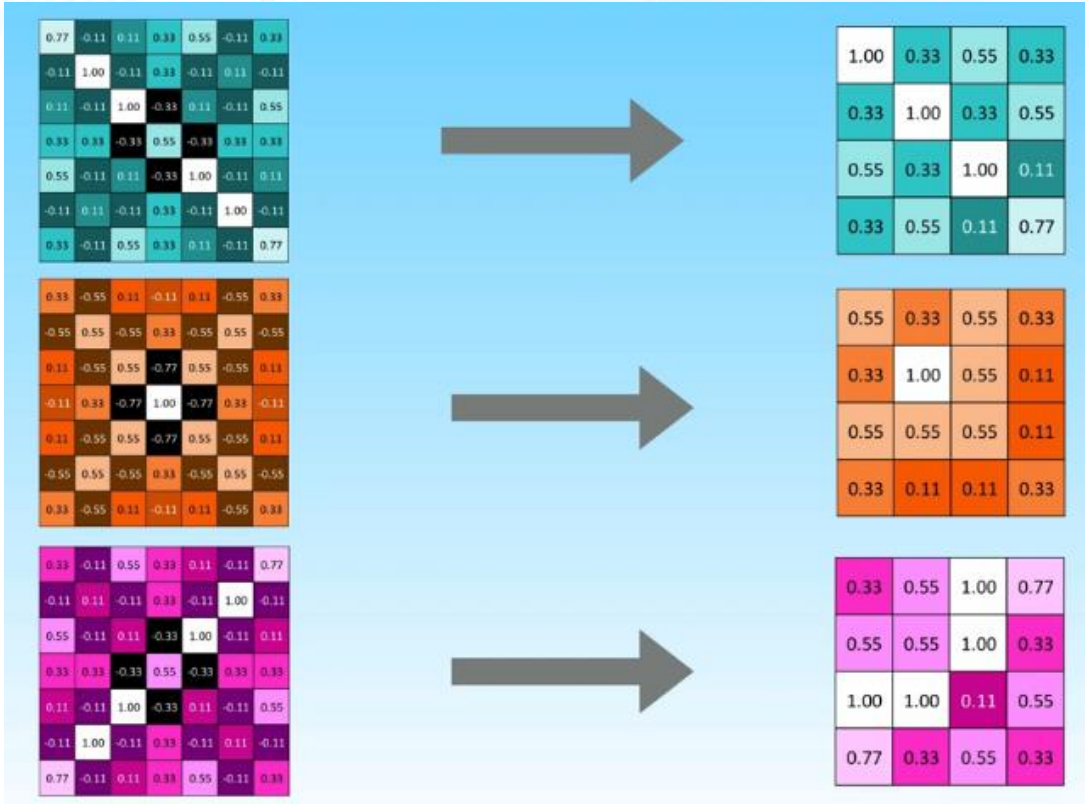


- Valid
- Same

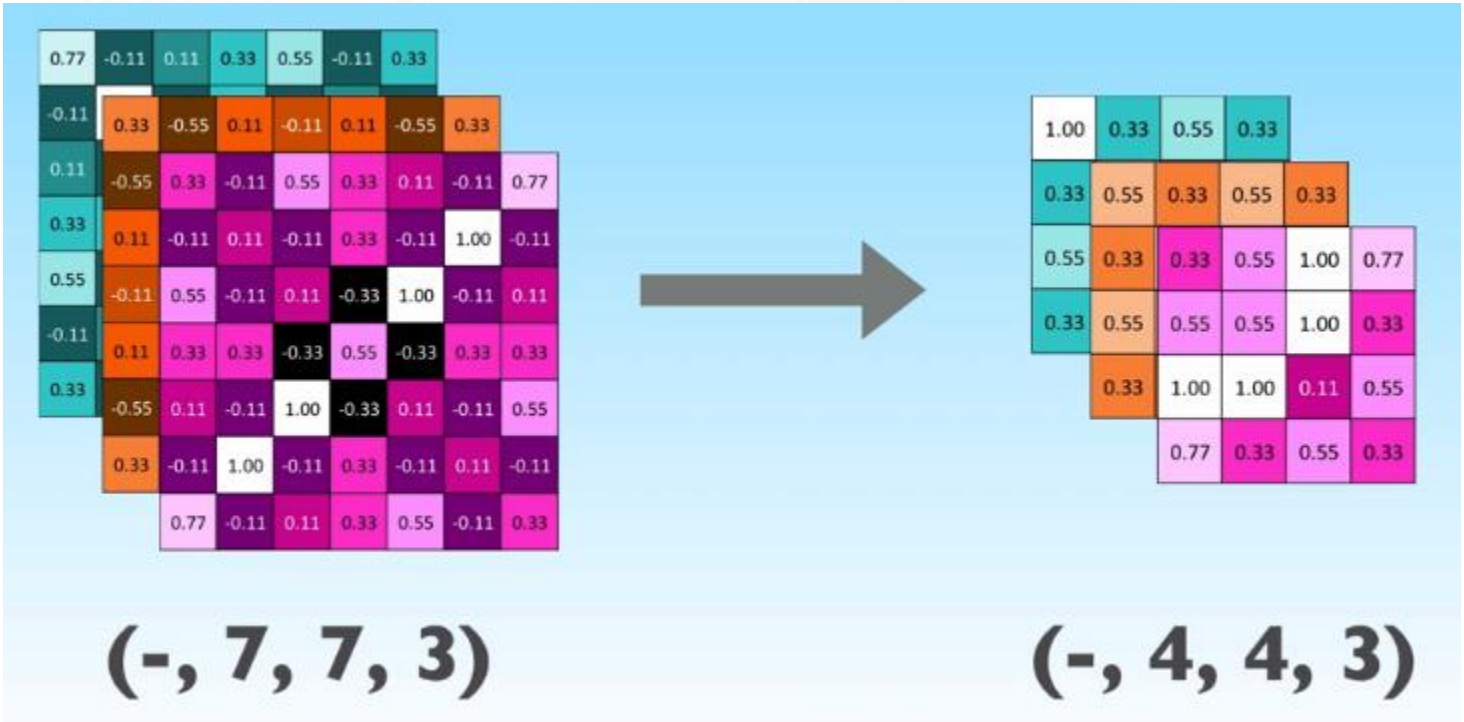
Max Pooling


0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
maximum
1.00 0.33 0.55 0.33
0.33

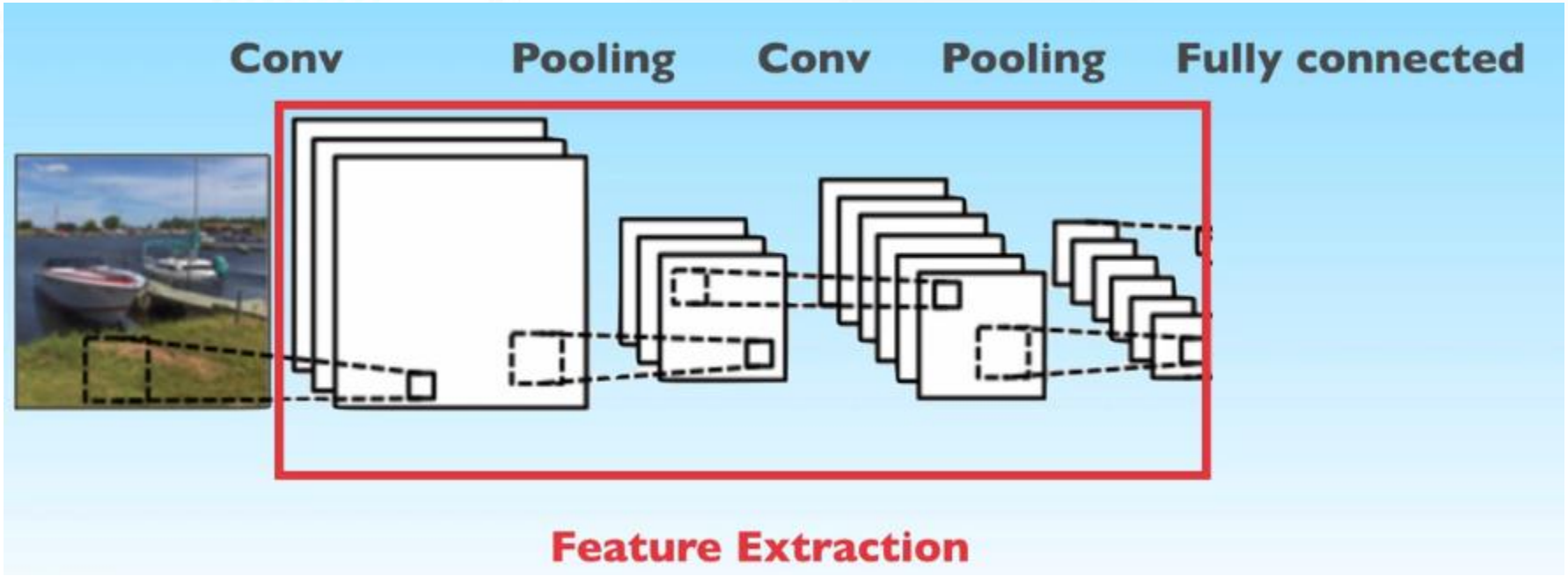
Pooling Layer

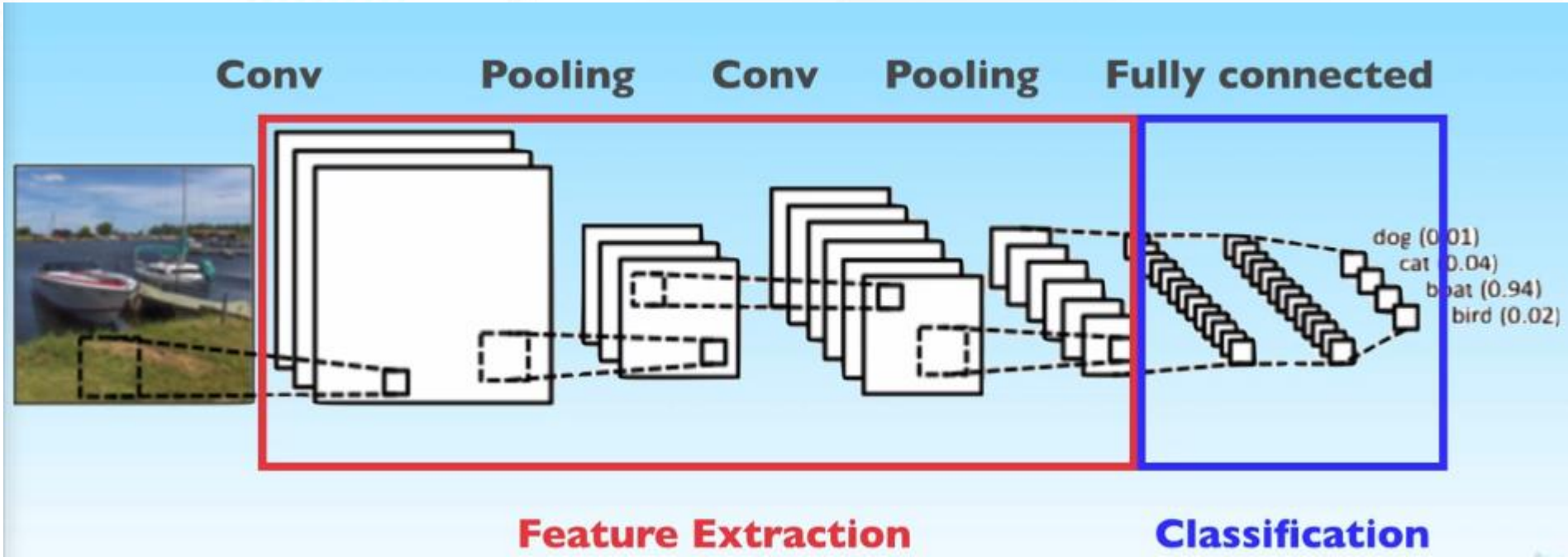


Pooling Layer

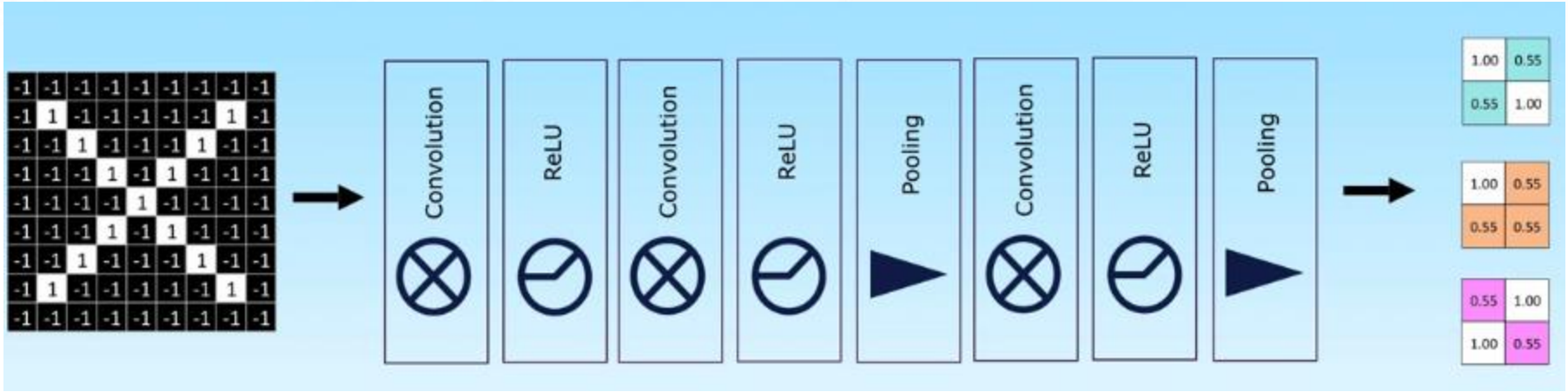


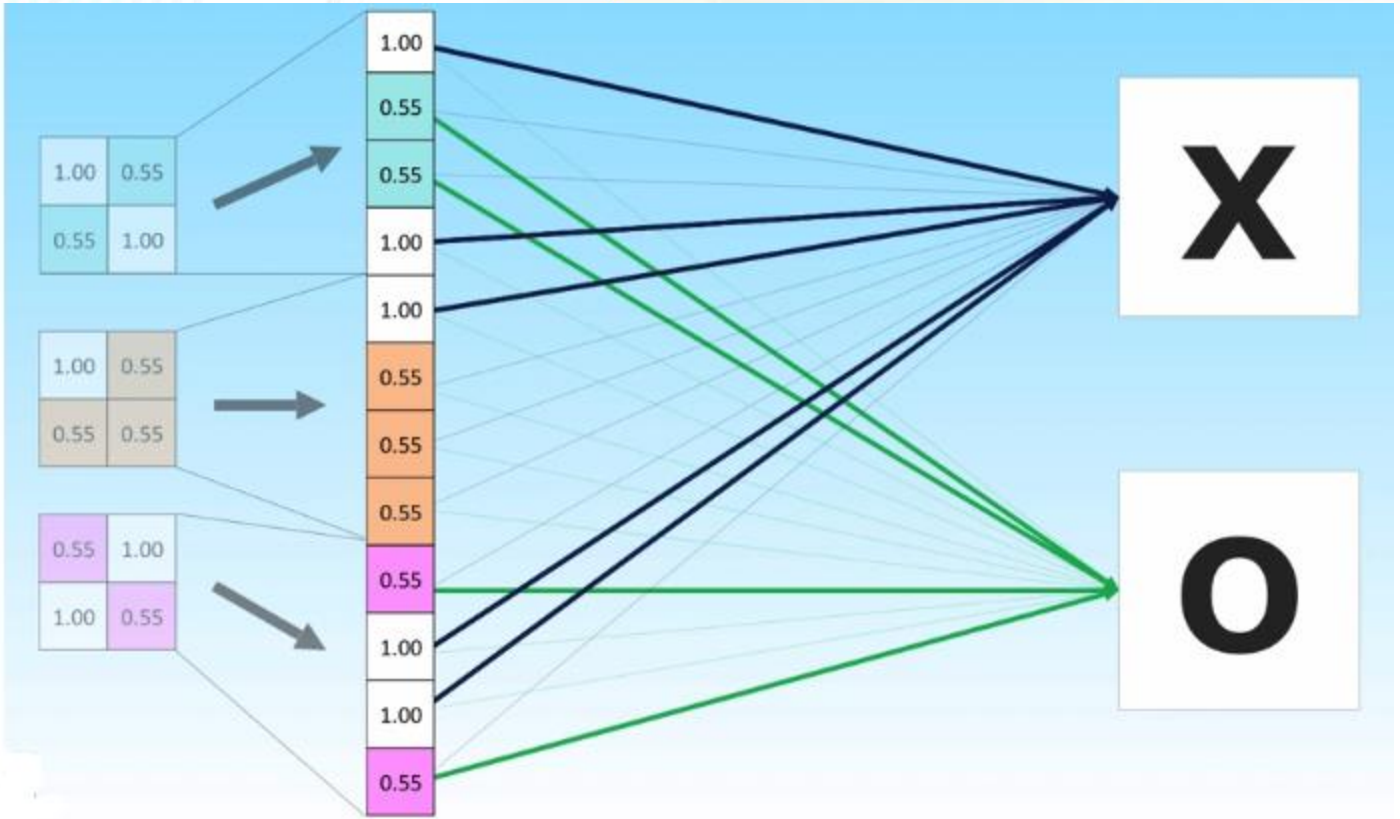
Stacking Layers in CNN



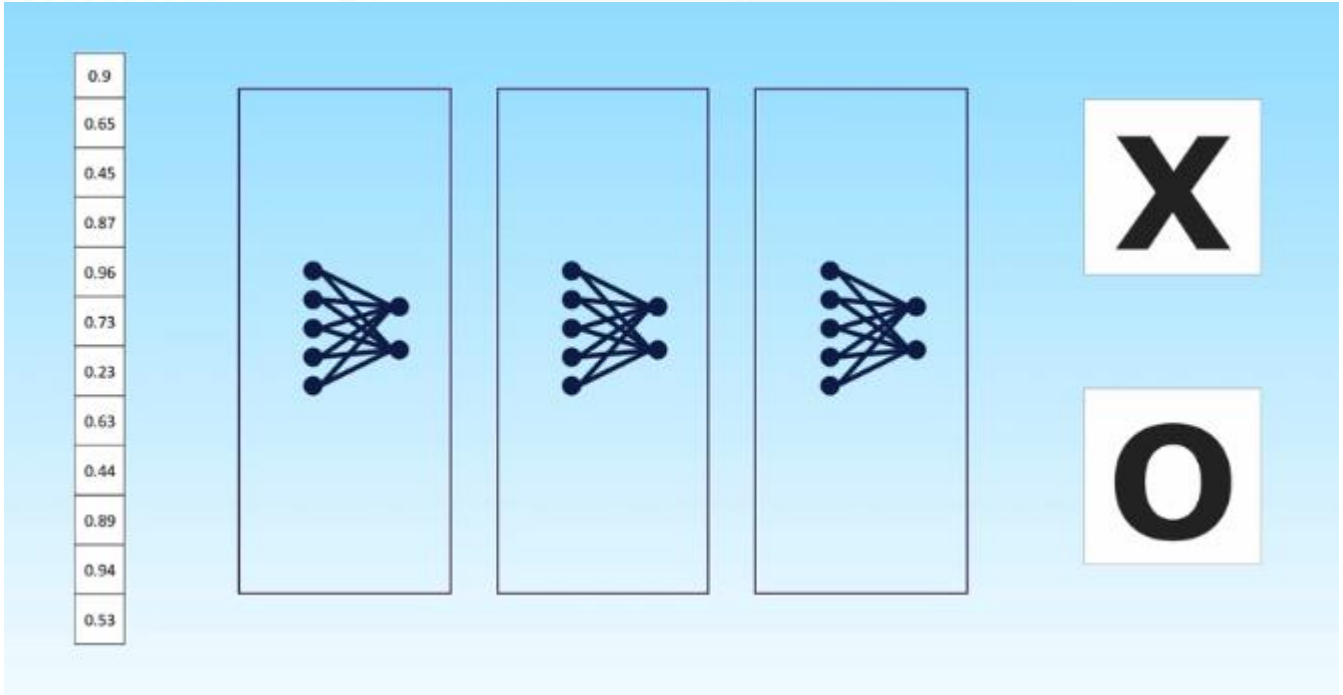

Conv
Pooling
Conv
Pooling
Fully connected
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)
Feature Extraction
Classification

Feature Extraction

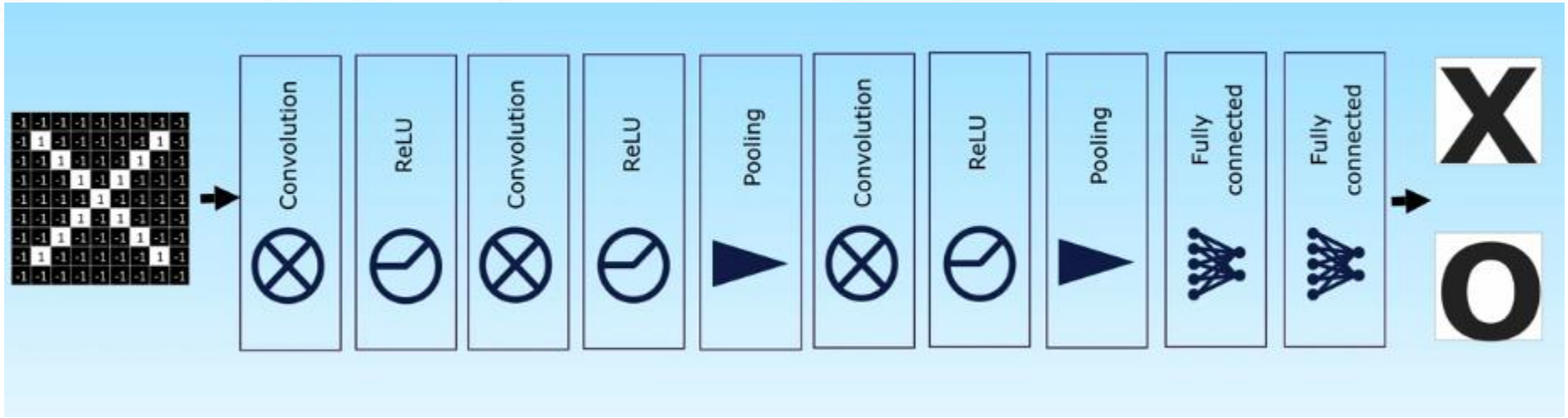



1.00
0.55
0.55
1.00
1.00
0.55
0.55
0.55
0.55
1.00
1.00
0.55
1.00
0.55
0.55
0.55
0.55
0.55
0.55
1.00
1.00
1.00
0.55
X
O

Stacked FC



Getting it all together



Weights In FC

- Input data: Image 10*10 => 100 Numbers
- First Layer: 32 nodes => 100 weights each
- => 3200 weights!!!!

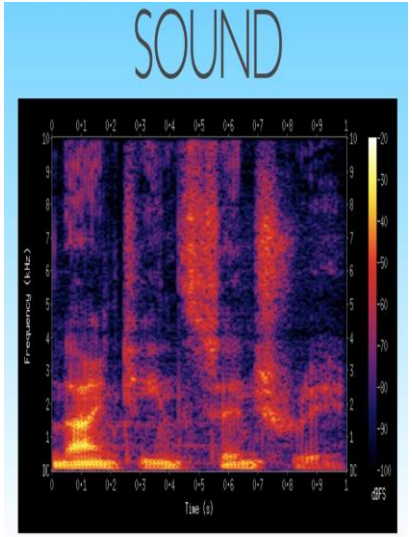
Weights In CNN

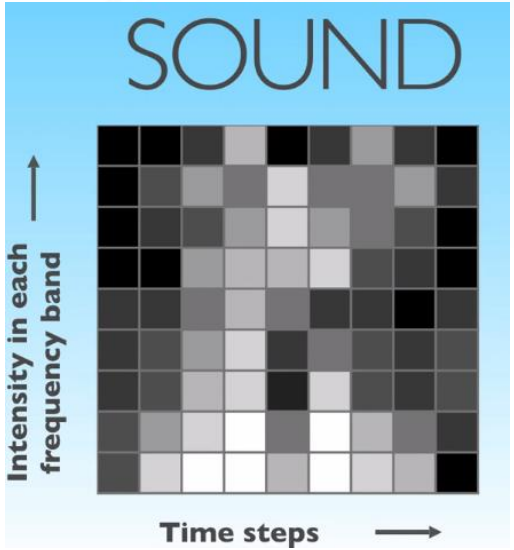
- Input data: Image 10*10 => 100 Numbers
- First Layer: 32 conv, kernel 3*3 => 9 weights each
- => 32*9 = 288 weights!!!

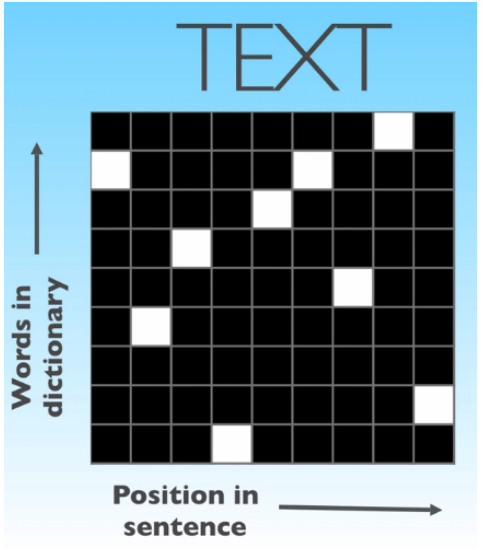
Advantages of CNN

- Weights are shared by Pixels
- Traded parameters with computation
- Number of weights in CNNs < Number of weights in FC

Beyond Images







Biggest Limitation of CNN

- ConvNets capture only local “spatial” patterns in data

User ID, Age, Email ...

A	22	1A	a@a	1	aa	a1.a	123	aa1
B	33	2B	b@b	2	bb	b2.b	234	bb2
C	44	3C	c@c	3	cc	c3.c	345	cc3
D	55	4D	d@d	4	dd	d4.d	456	dd4
E	66	5E	e@e	5	ee	e5.e	567	ee5
F	77	6F	f@f	6	ff	f6.f	678	ff6
G	88	7G	g@g	7	gg	g7.g	789	gg7
H	99	8H	h@h	8	hh	h8.h	890	hh8
I	111	9I	i@i	9	ii	i9.i	901	ii9

- Can you swap rows?
- Can you swap columns?
- Is data still useful?

No Use for ConvNet

Exercise 1

- You've been hired by a shipping company to overhaul the way they route mail , parcels and packages. They want to build an image recognition system capable of recognizing the digits in the zip-code on a package, so that it can be automatically routed to the current location. You are tasked to build the digit recognition system. Luckily, you can rely on the MNIST dataset for the initial training model!
- Build a deep convolutional neural network with at least two convolutional and two pooling layers before the fully connected layer.
- Start from the network we have just built
- Insert a Conv2D layer after the first MaxPool2D, give it 64 filters.
- Insert a MaxPool2D After that one
- Insert an Activation layer
- Retrain the model
- Does performance improve?
- How many parameters does this new model have?
- How long did this model take to train?

Exercise two

- Pleased with your performance with the digits recognition task, your boss decides to challenge you with a harder task. Their online branch allows people to upload images to a website that generates and prints a postcard that us shipped to destination. Your boss would like to know what images people are uploading on the site in order to provide targeted advertising on the same page, so he asks you to build an image recognition system capable of recognizing a few objects. Luckily for you, there's a dataset ready made with a collection of labeled images. This is the Cifar 10 dataset, a very famous dataset that contains images for 10 different categories:
  - Airplane
  - Automobile
  - Bird
  - Cat
  - Deer
  - Dog
  - Frog
  - Horse
  - Ship
  - truck

Exercise 2

- In this exercise we will reach the limit of what you can achieve on your laptop and get ready for the next session on cloud GPUs.
- Here's what you have to do:
- Load the cifar 10 dataset using keras.datasets.cifar10.load_data()
- Display a few images, see how hard/easy it is for you to recognize an object with such low resolution
- Check the shape of X_train, does it need reshape?
- Check the shape of X_train, does it need rescaling?
- Check the shape of y_train, does it need reshape?
- Build the model with the following architecture, and choose the parameters and activation functions for each of the layers:
  - conv2d
  - Conv2d
  - Maxpool
  - Conv2d
  - Conv2d
  - Maxpool
  - Flatten
  - Dense
  - output
- Compile the model and check the number of parameters.
- Attempt to train the model with the optimizer of your choice. How fast does training proceed?
- If training is too slow(as expected) stop the execution and move on the the next session!