BATCH : BATCH 150 DATA SCIENCE

LESSON : DEEP LEARNING

DATE : 31.08.2023

SUBJECT : CNN

techproeducation

techproeducation

techproeducation

techproeducation

techproedu

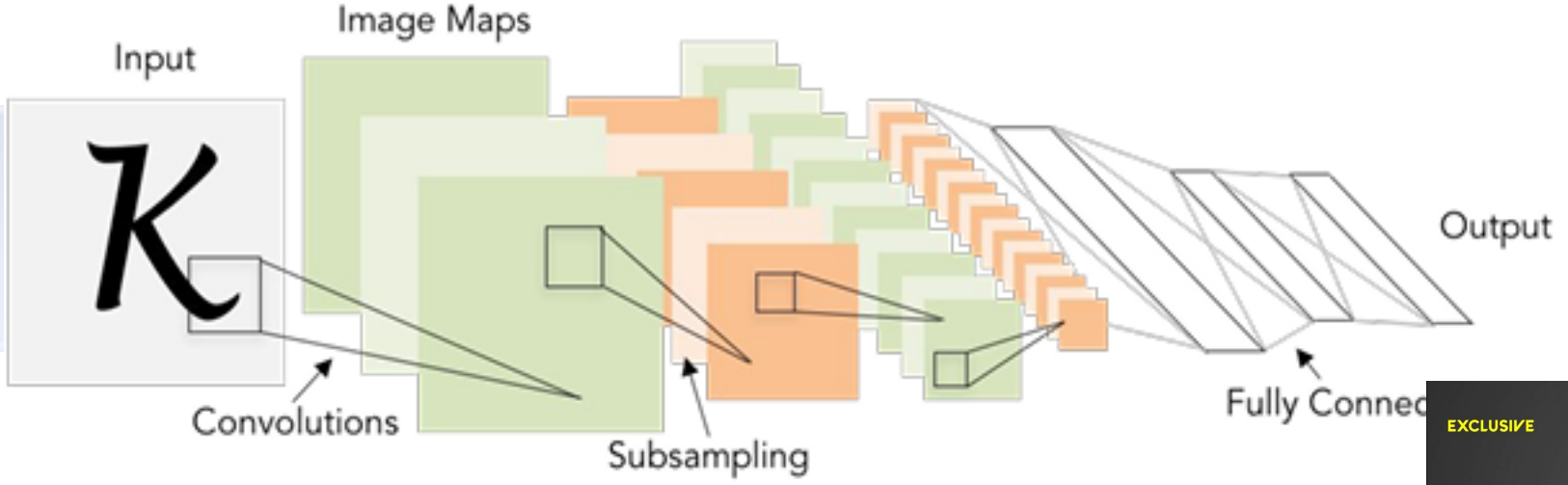
CONVOLUTIONAL NEURAL NETWORKS

CNN (Convulutional Neural Network)

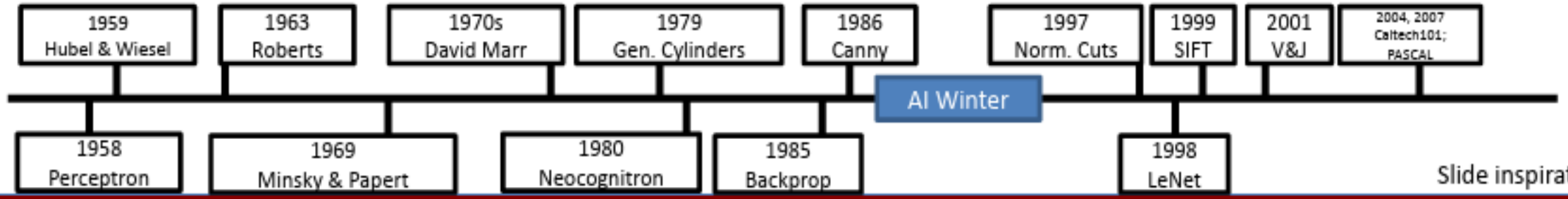
EVRİŞİMLİ SİNİR AĞLARI


EXCLUSIVE
The GodFather of Deep Learning:
Yann LeCun

Convolutional Networks: LeCun et al, 1998

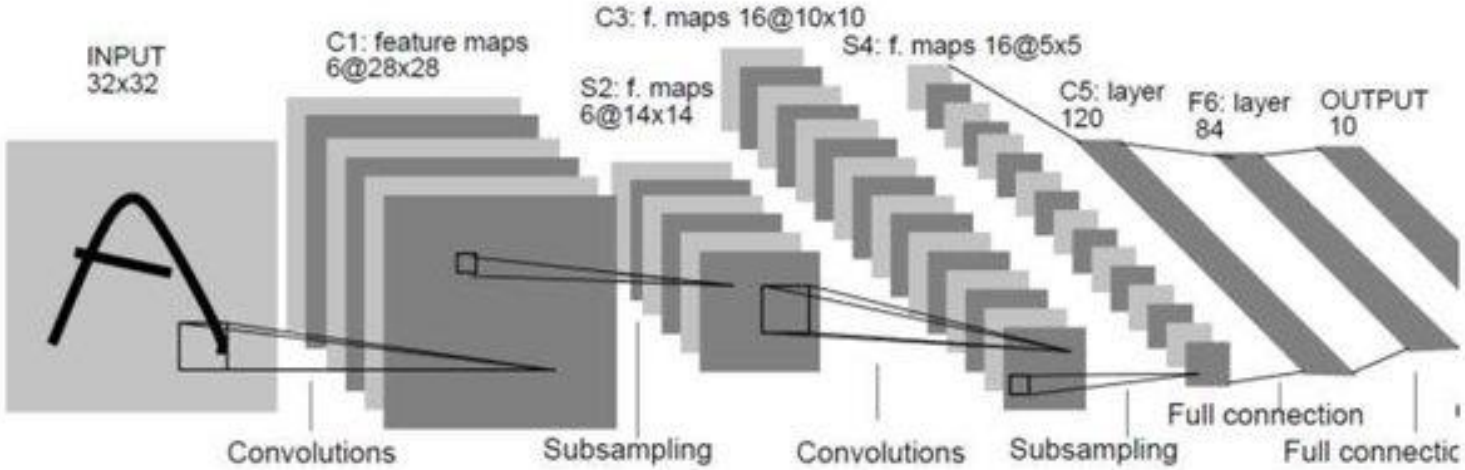


Applied backprop algorithm to a Neocognitron-like architecture
Learned to recognize handwritten digits
Was deployed in a commercial system by NEC, processed handwritten c
Very similar to our modern convolutional networks!



Slide inspiration: Justin Johnson

Lenet-5 (1998)



MNIST: handwritten digits

- 70,000 28x28 pixel images
- Gray scale
- 10 classes

CIFAR-10: simple objects

- 60,000 32x32 pixel images
- RGB
- 10 classes

1989 (Lecun) A convnet is used for an image classification task (zip codes)

- First time backprop is used to automatically learn visual features
- Two convolutional layers, two fully connected layer (16x16 input, 12 FMs each layer, 5x5 filters)
- Stride=2 is used to reduce image dimensions
- Scaled Tanh activation function
- Uniform random weight initialization

1998 (Lecun) LeNet-5 convnet achieves state of the art result on MNIST

- Two convolutional layers, three fully connected layers (32x32 input, 6 and 12 FMs, 5x5 filters)
- Average pooling to reduce image dimensions
- Sparse connectivity between feature maps

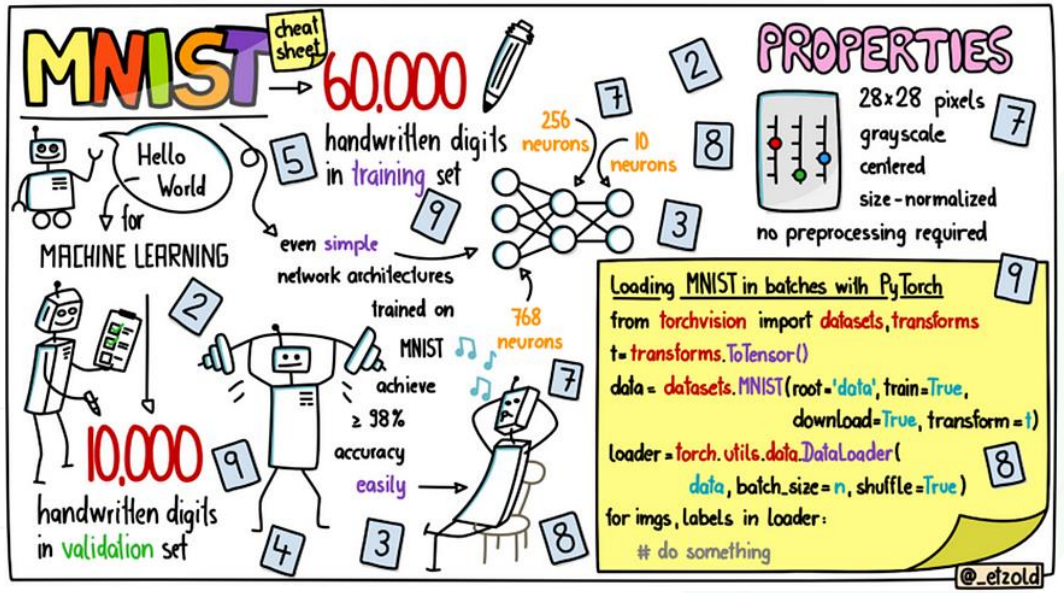
LeCun et al, Gradient-Based Learning Applied to Document Recognition

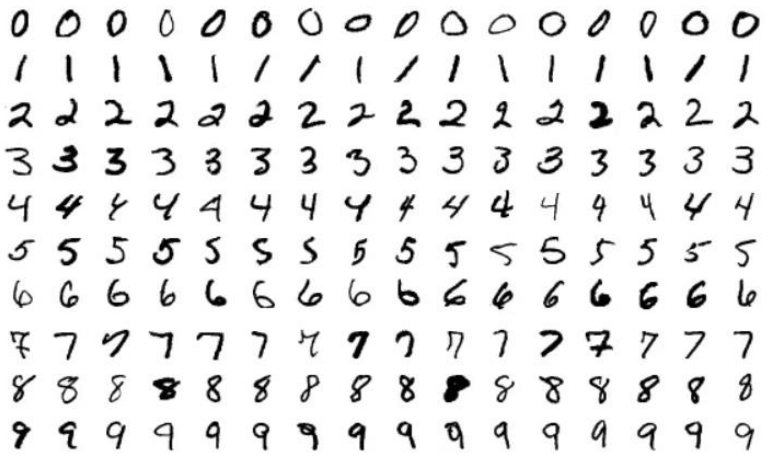
Derin Öğrenme Modellerinin Eğitiminde Kullanılan 5 Önemli Veri Seti

Muslum Yildiz

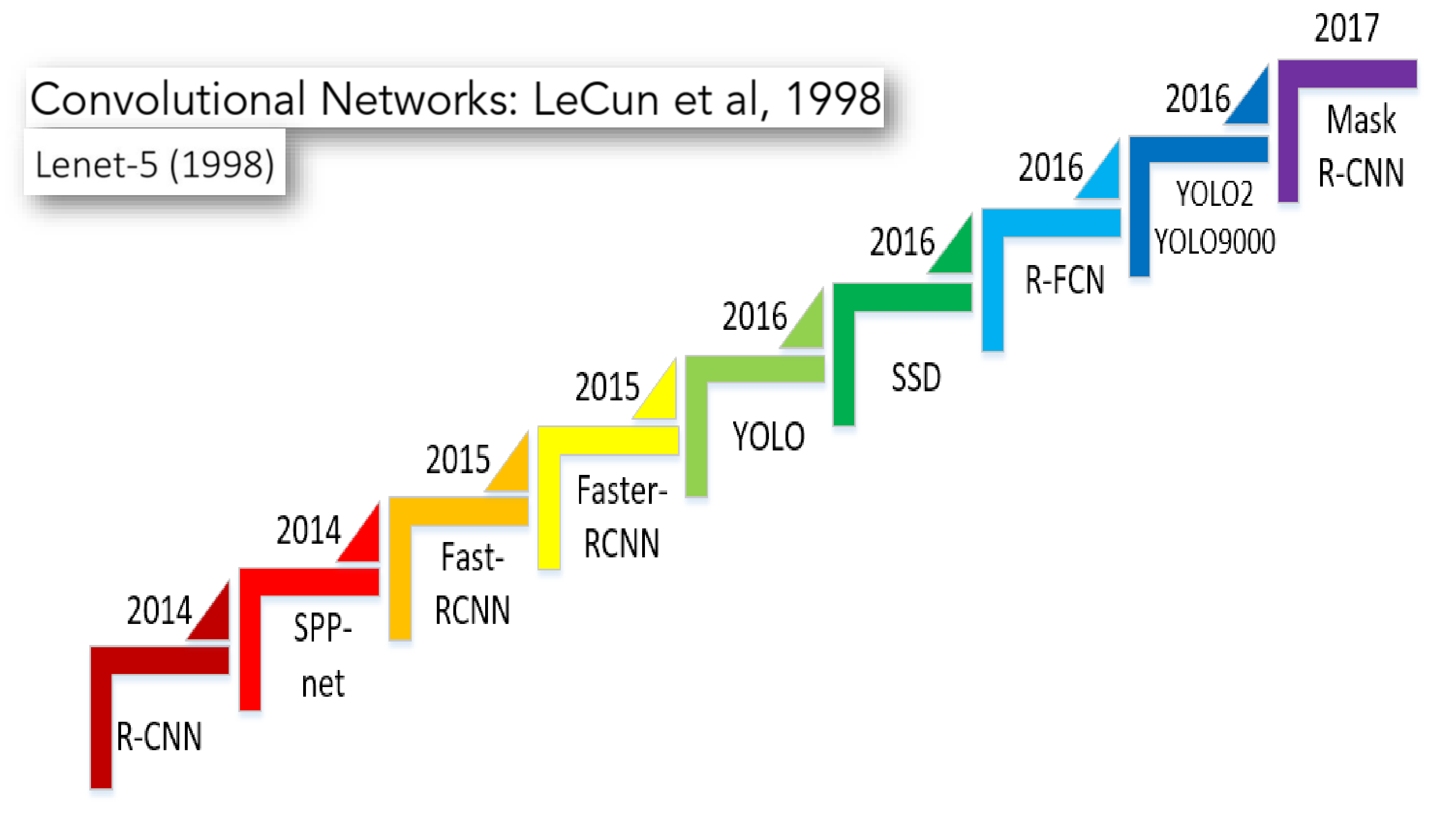
Published in Academy Team · 9 min read · Apr 28

53





Derin öğrenme, yapay zeka alanında son yıllarda büyük bir ivme kazanmıştır ve birçok uygulama alanında çığır açan sonuçlar elde etmiştir. Ancak, bu modellerin doğru bir şekilde eğitilmesi için uygun veri setlerine ihtiyaç duyulmaktadır. Bu nedenle, derin öğrenme modellerinin eğitiminde kullanılan önemli veri setleri incelenmeli ve bu veri setlerinin özellikleri dikkate alınarak modellerin eğitiminde kullanılmalıdır. Bu veri


Convolutional Networks: LeCun et al, 1998
Lenet-5 (1998)
2014
R-CNN
2014
SPP-net
2015
Fast-RCNN
2015
Faster-RCNN
2016
YOLO
2016
SSD
2016
R-FCN
2016
YOLO2
YOLO9000
2017
Mask R-CNN

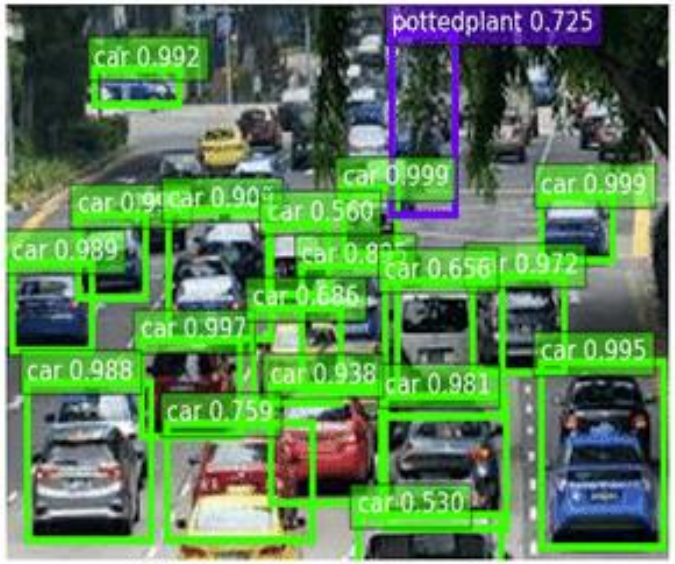
Model	Yıl	Geliştirici	Özellikleri
CNN (ConvNet)	1998	Yann LeCun	Görüntü işleme için evrişimli sinir ağı (CNN)
R-CNN	2014	Ross Girshick	Görüntü sınıflandırma ve nesne tespiti için bölgesel evrişimli sinir ağı (R-CNN)
Fast R-CNN	2015	Ross Girshick	R-CNN'nin hızlandırılmış versiyonu
Faster R-CNN	2015	Shaoqing Ren	Nesne tespiti için bir önceden eğitilmiş modeli kullanarak RPN (Region Proposal Network) kullanan bir R-CNN modeli
YOLO (You Only Look Once)	2016	Joseph Redmon	Nesne tespiti için end-to-end CNN modeli
SSD (Single Shot Detector)	2016	Wei Liu	Hızlı ve hassas nesne tespiti için end-to-end CNN modeli
Mask R-CNN	2017	Kaiming He	R-CNN modelinin genişletilmiş versiyonu, hem nesne tespiti hem de piksel bazlı nesne segmentasyonu için kullanılabilir

CNN

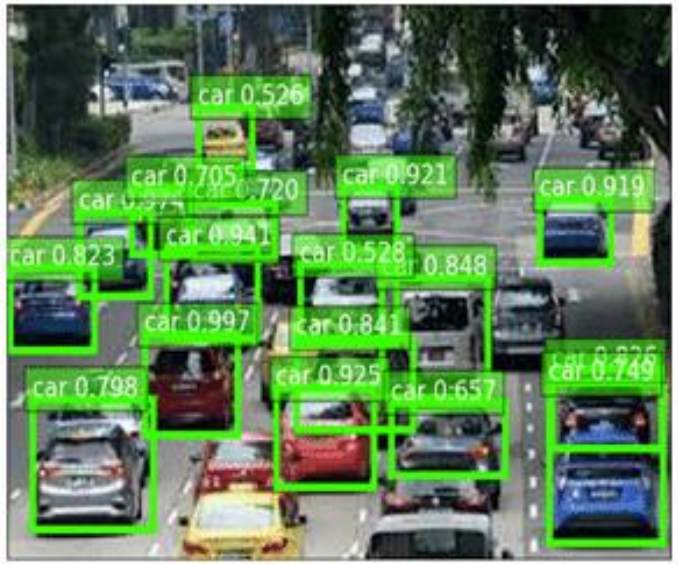
The input image

Faster R-CNN

YOLO

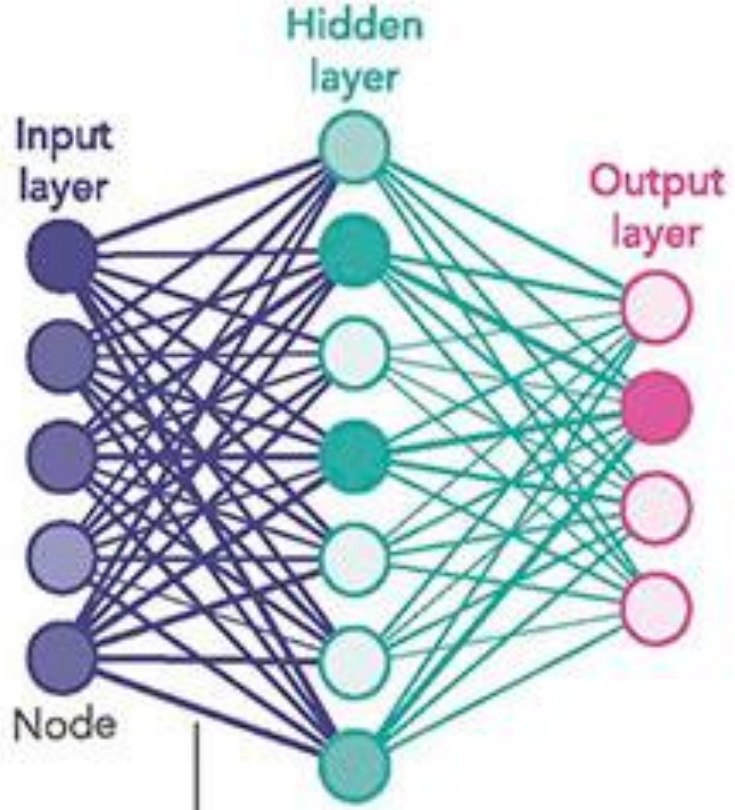


The predicted image by Faster R-CNN



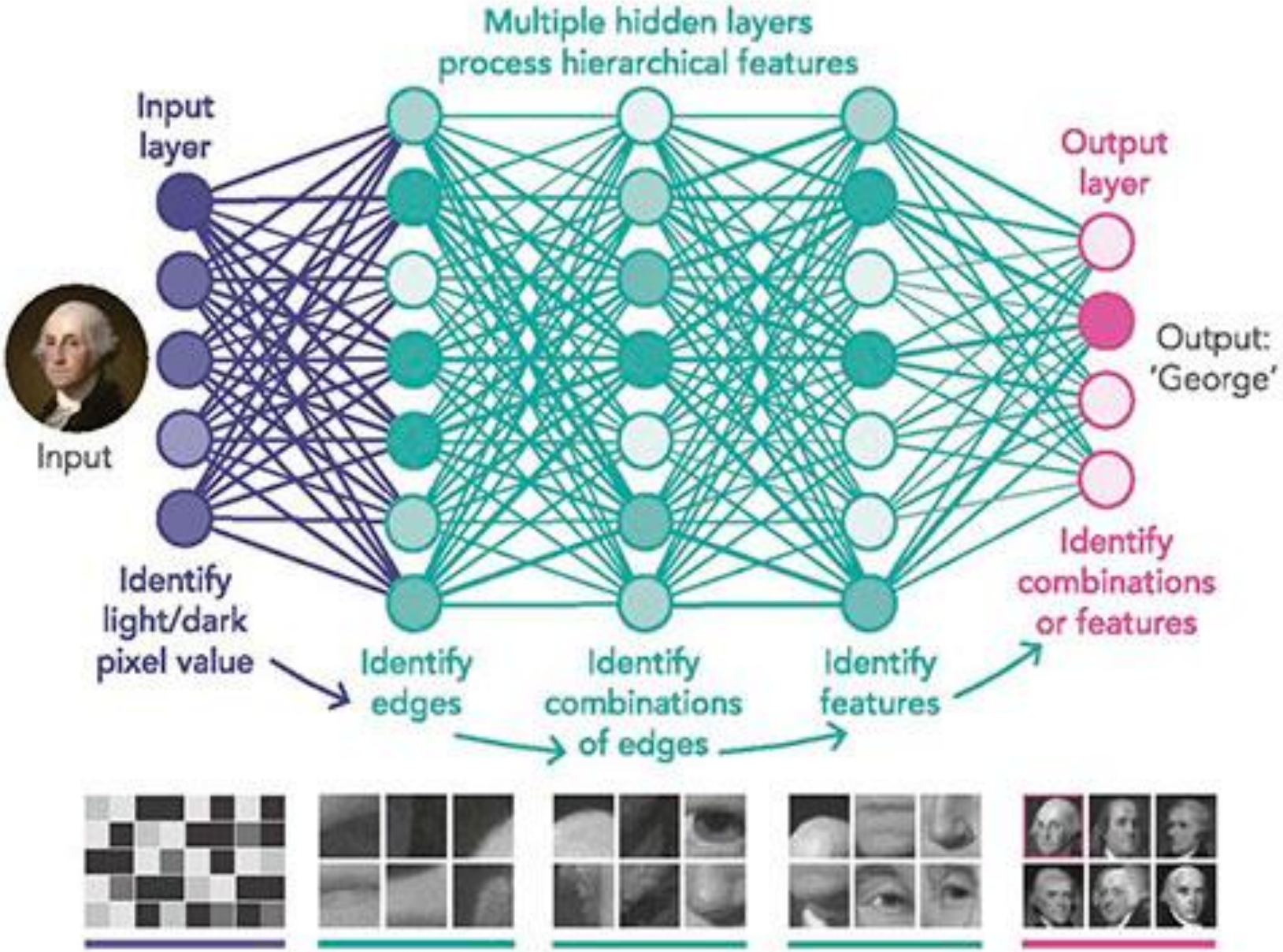
The predicted image by YOLO

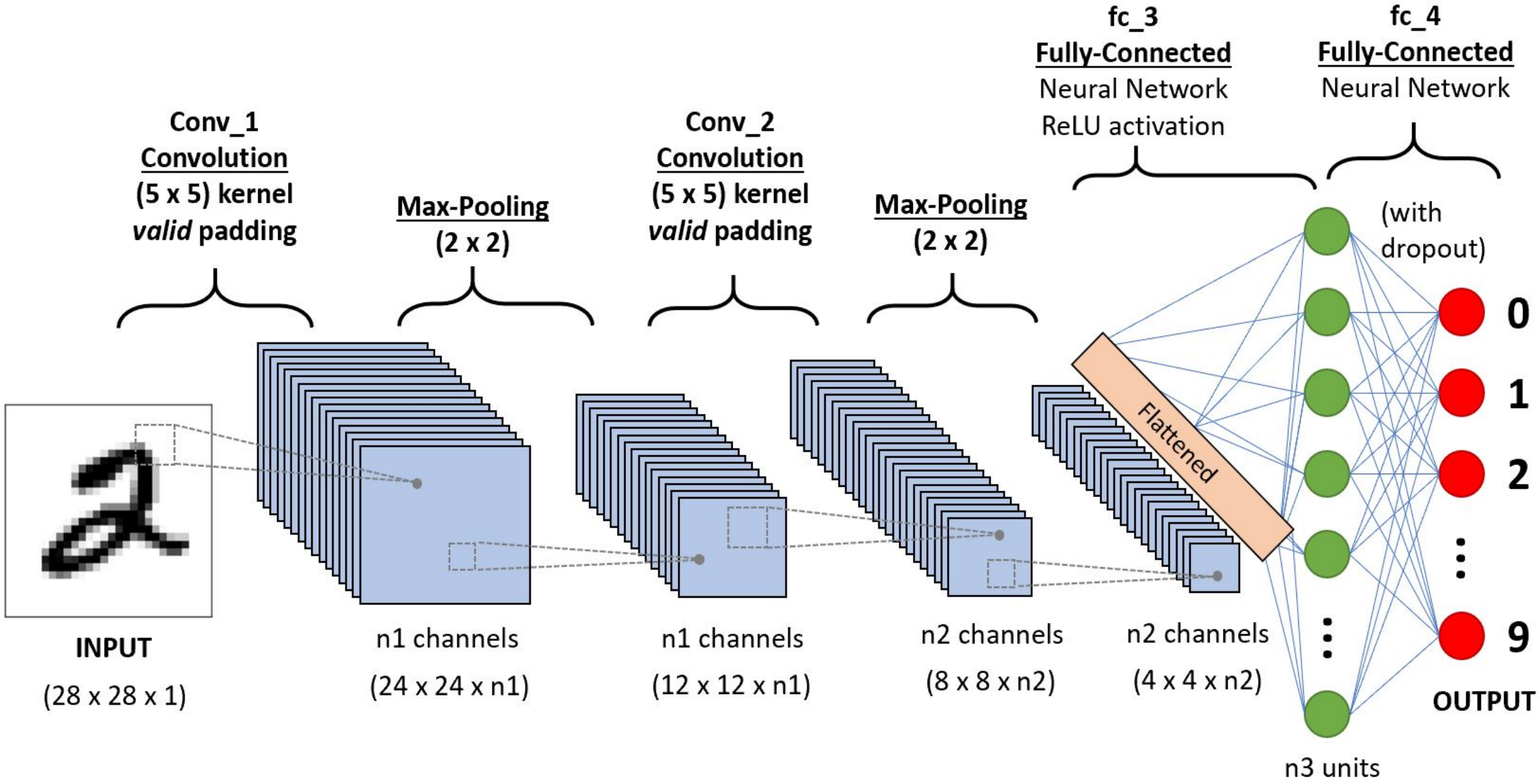
1980S-ERA NEURAL NETWORK



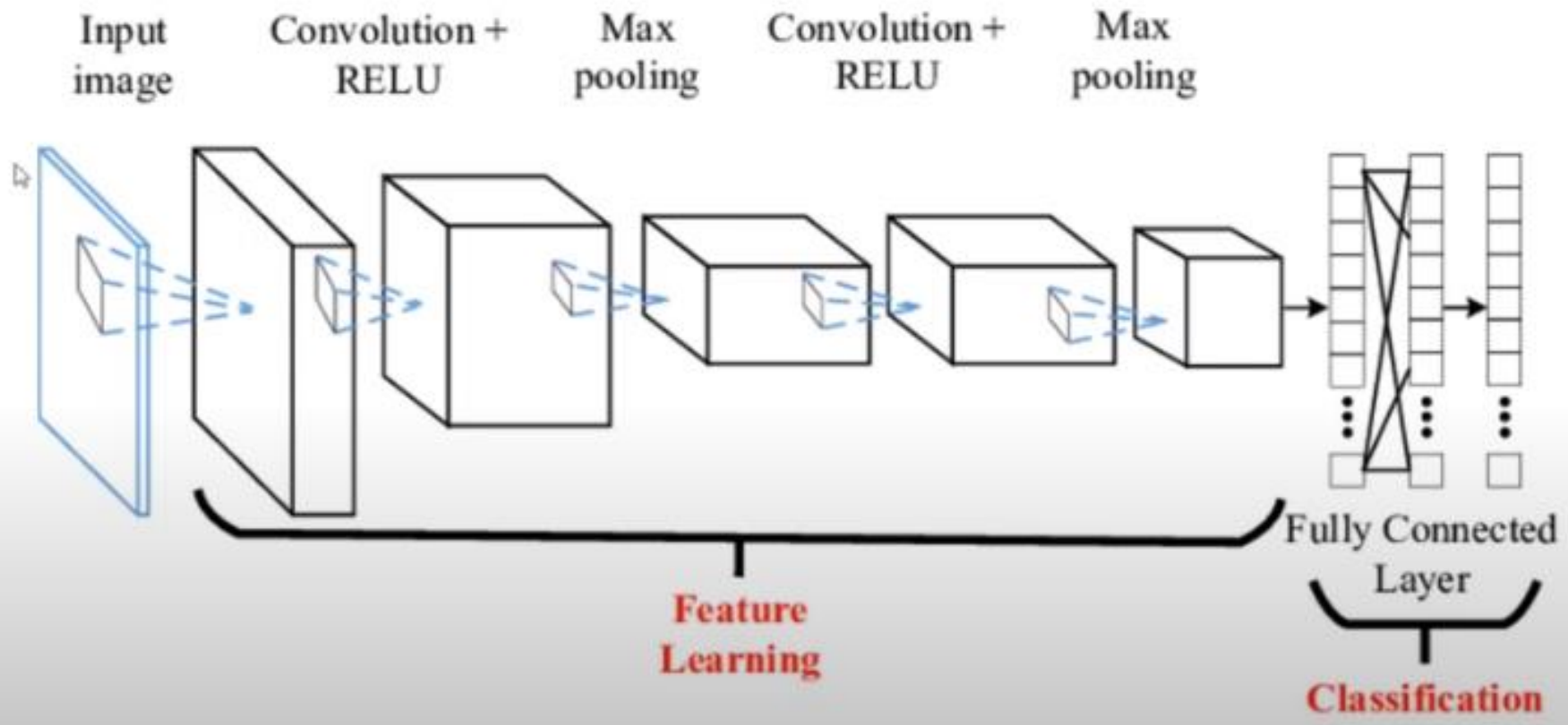
Links carry signals from one node to another, boosting or damping them according to each link's 'weight'.

DEEP LEARNING NEURAL NETWORK

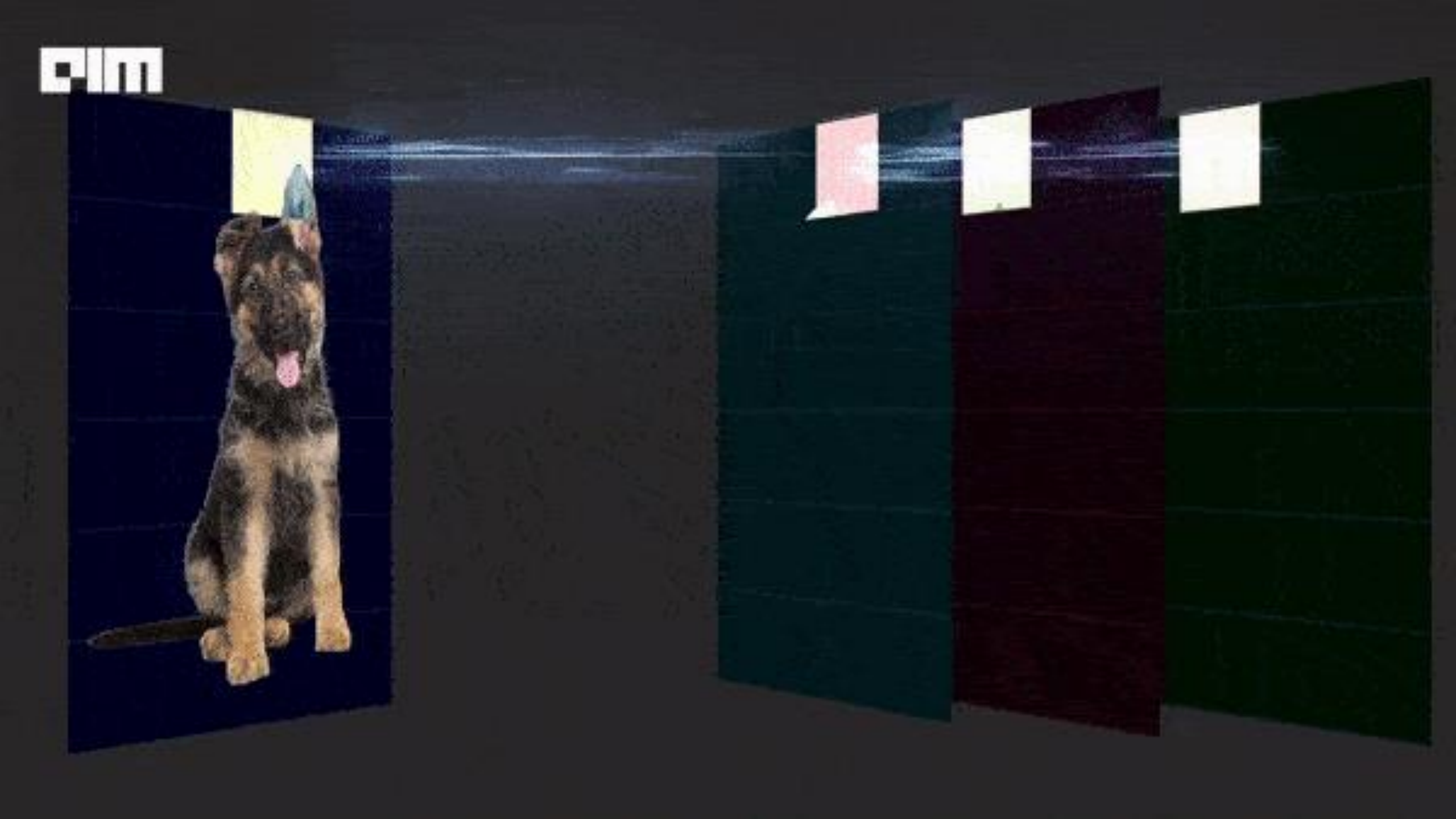



Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

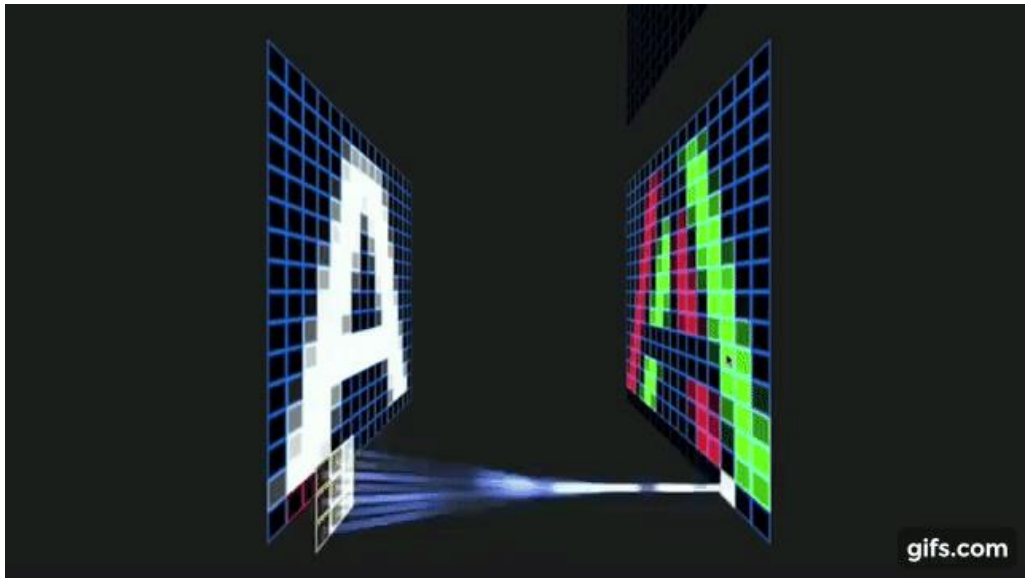
CNN

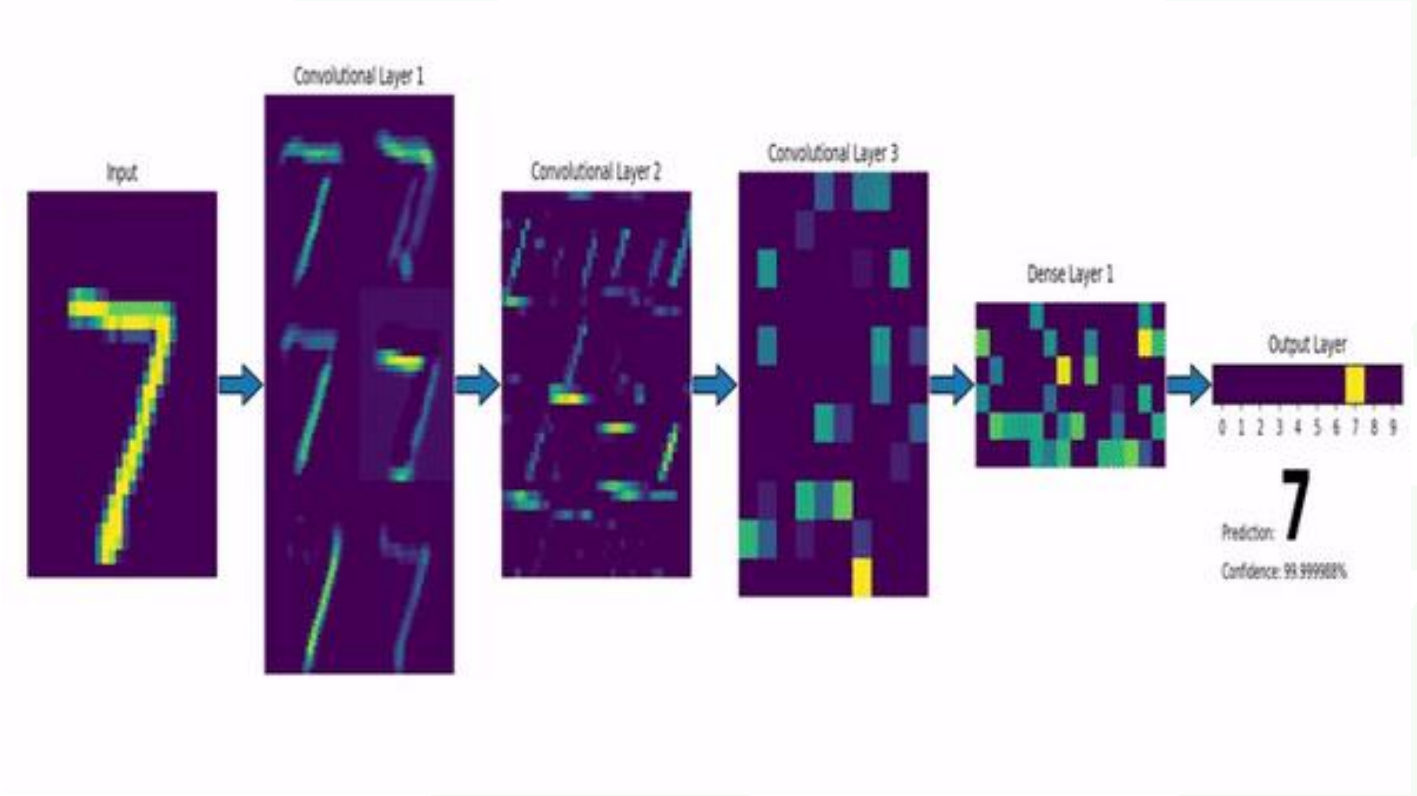

Input image
Convolution + RELU
Max pooling
Convolution + RELU
Max pooling
Fully Connected Layer
Feature Learning
Classification


horse

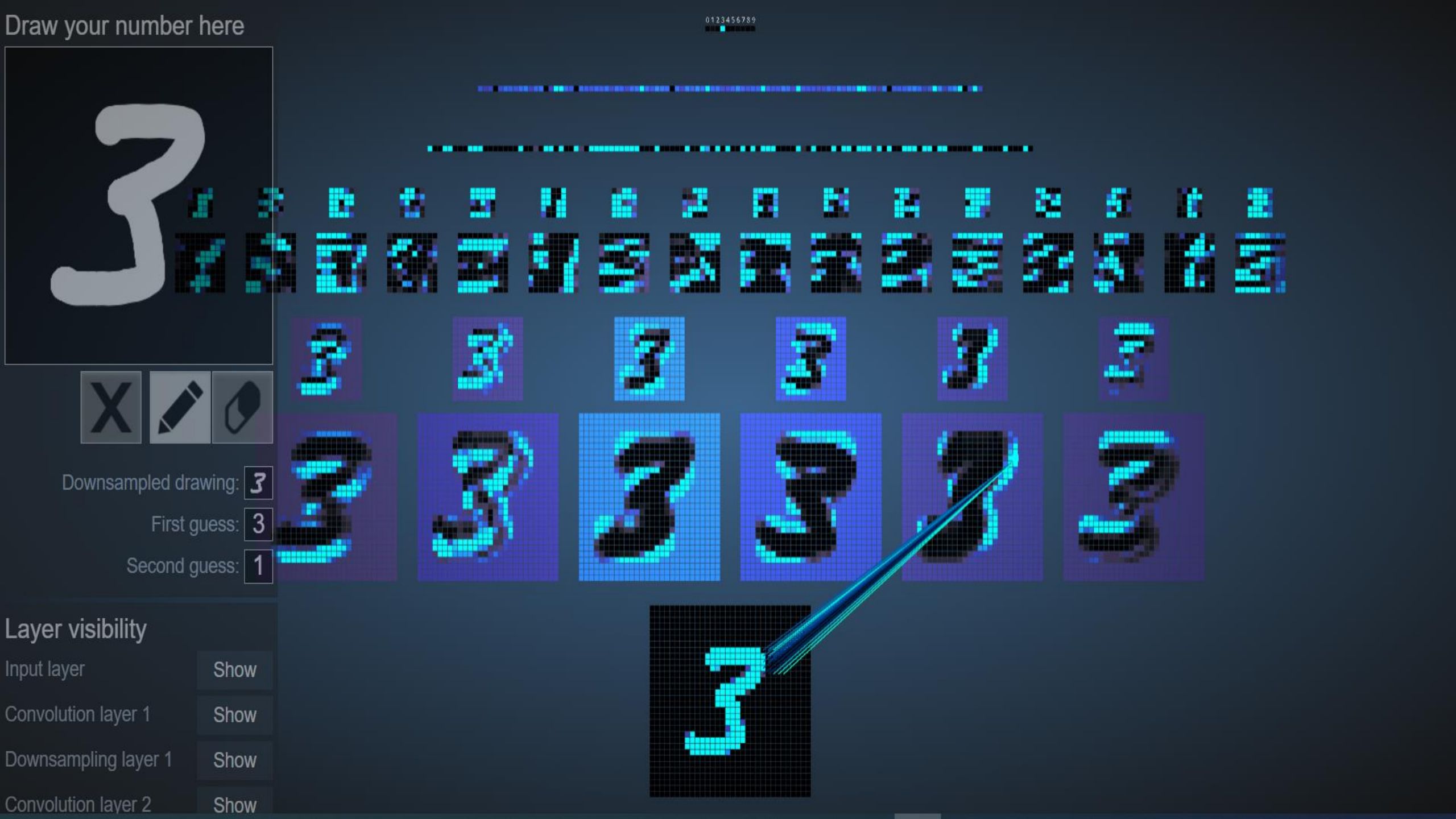


CNN

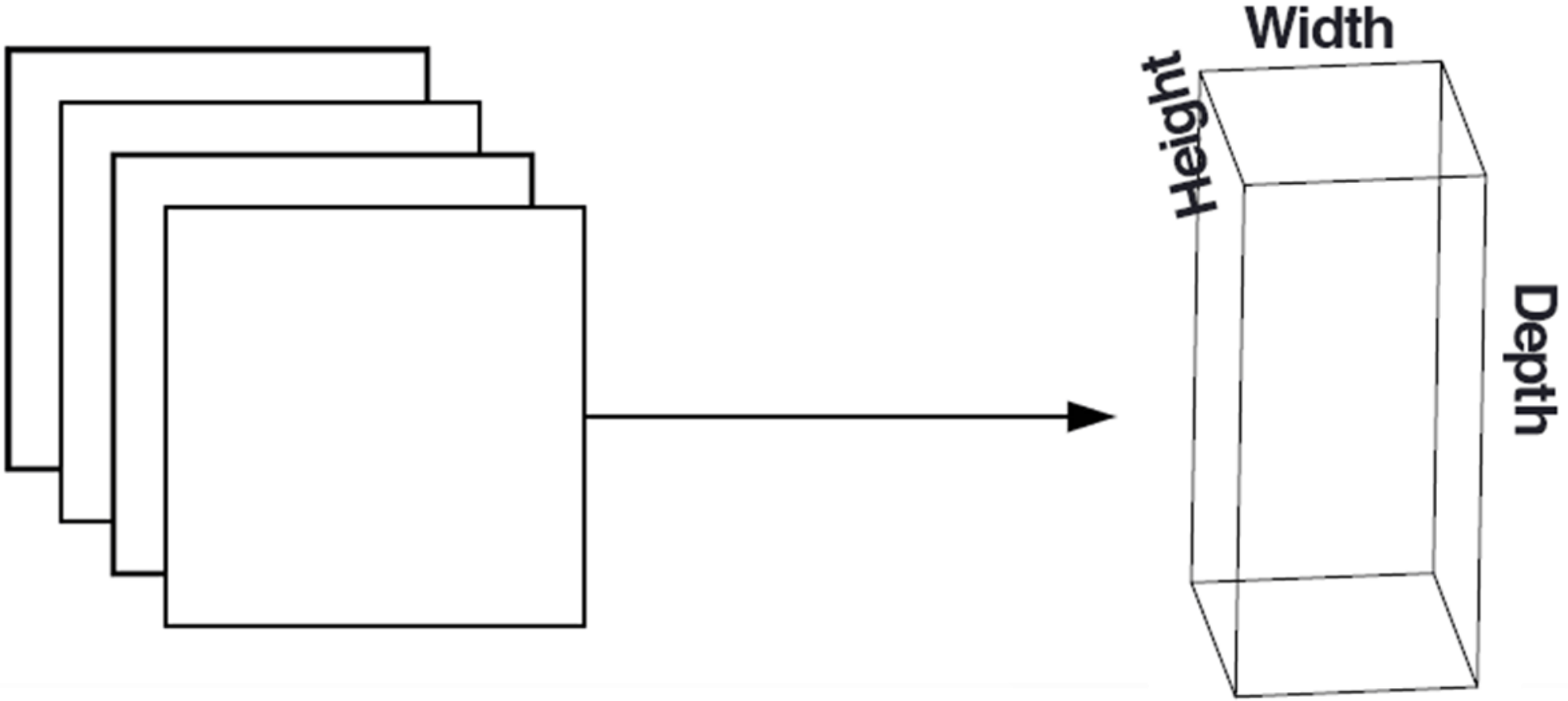

gifs.com


Input
Convolutional Layer 1
Convolutional Layer 2
Convolutional Layer 3
Dense Layer 1
Output Layer
0 1 2 3 4 5 6 7 8 9
Prediction: 7
Confidence: 99.999988%

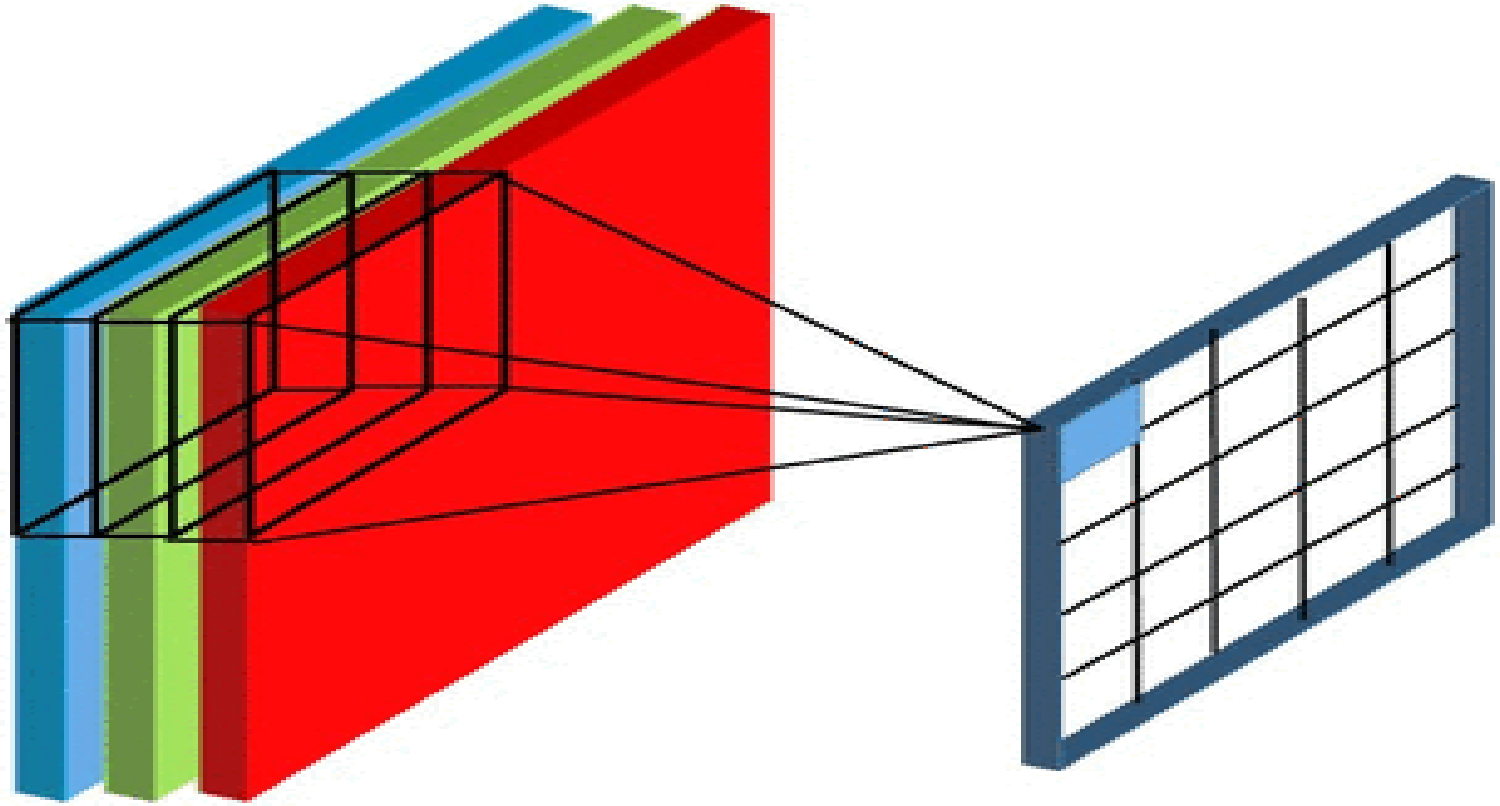
CNN


Draw your number here
0123456789
Downsampled drawing:
First guess: 3
Second guess: 1
Layer visibility
Input layer
Show
Convolution layer 1
Show
Downsampling layer 1
Show
Convolution layer 2
Show

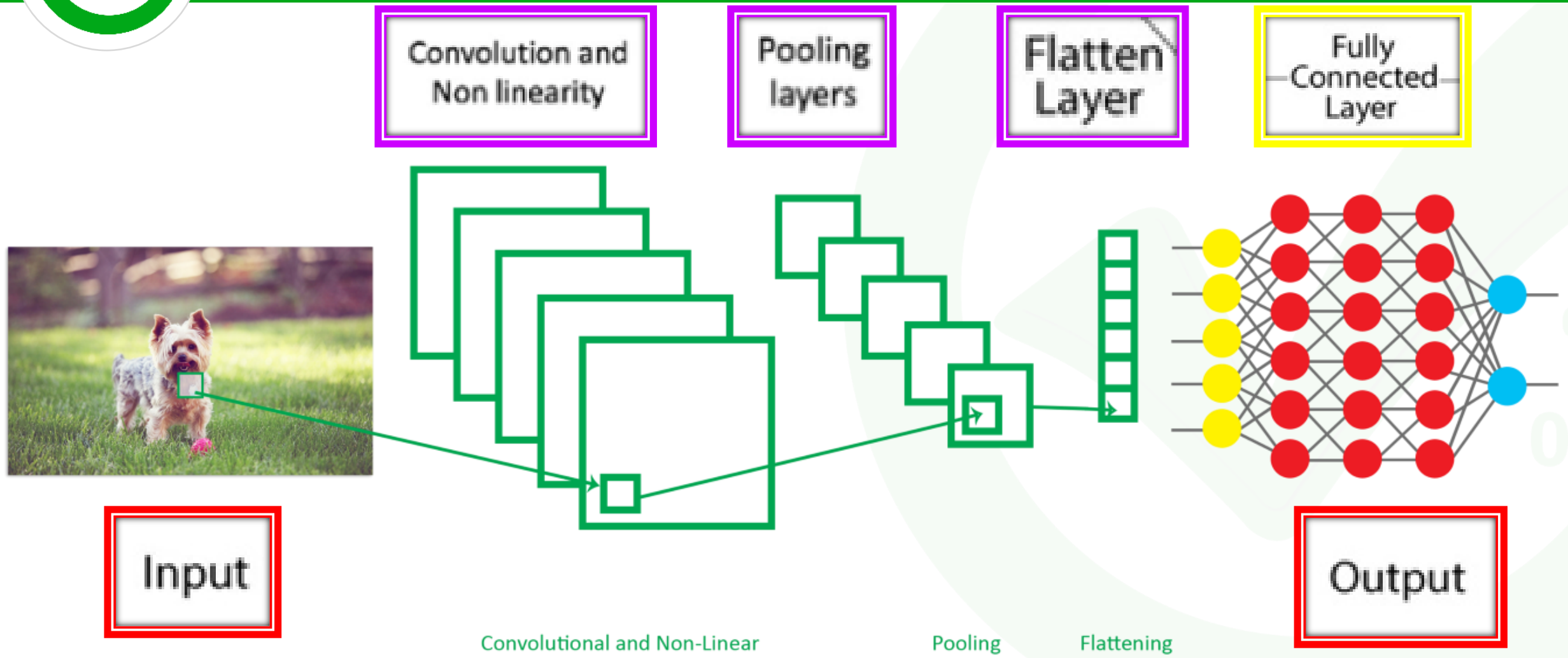
CNN



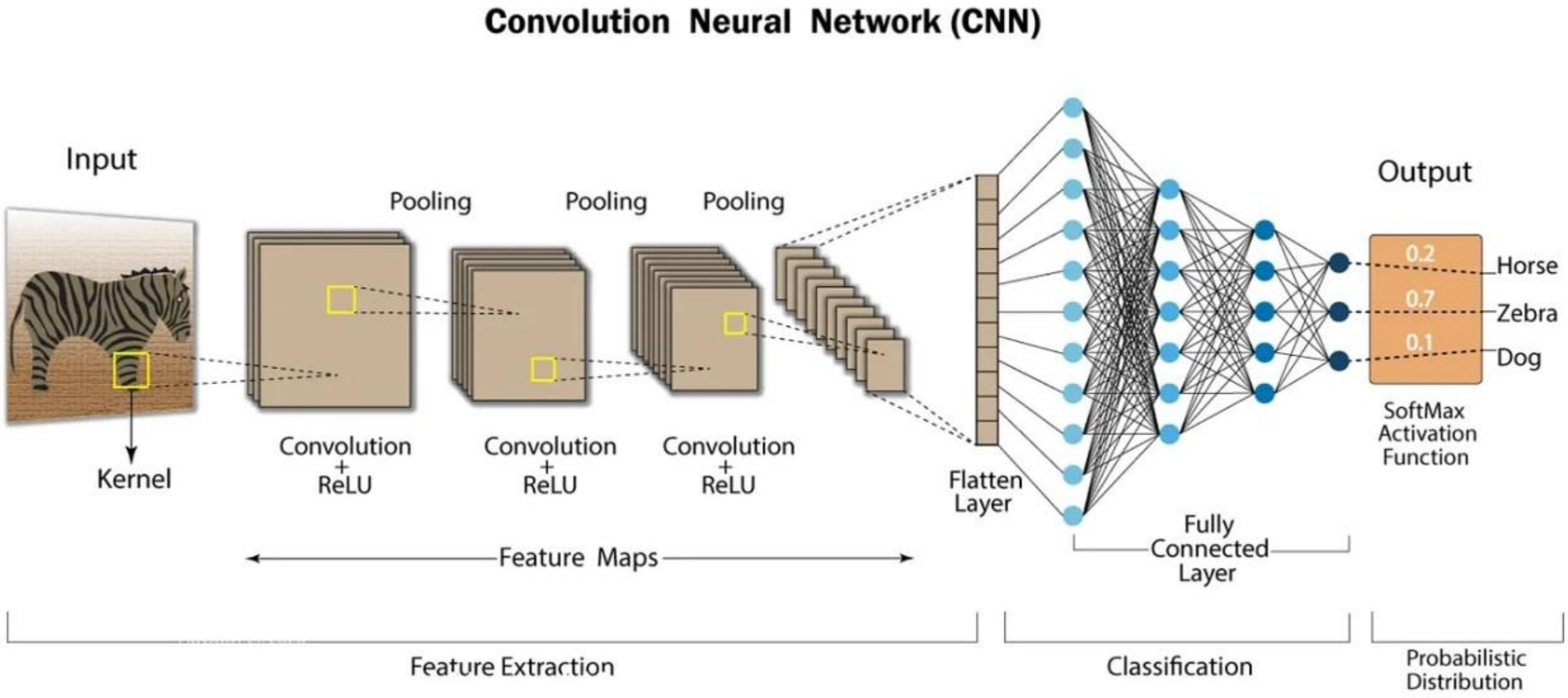
CNN



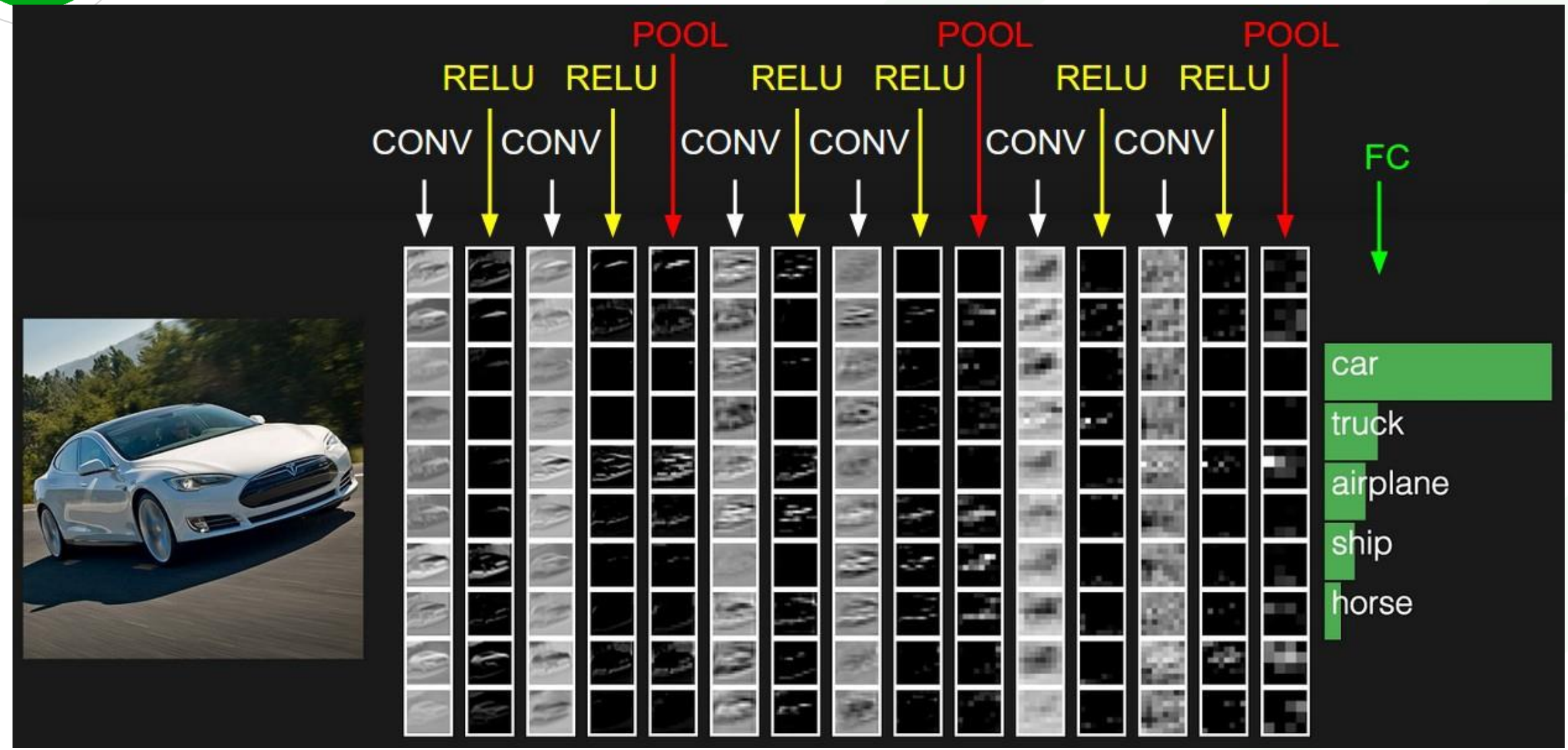
CNN


Convolution and Non linearity
Pooling layers
Flatten Layer
Fully Connected Layer
Input
Output
Convolutional and Non-Linear
Pooling
Flattening

CNN

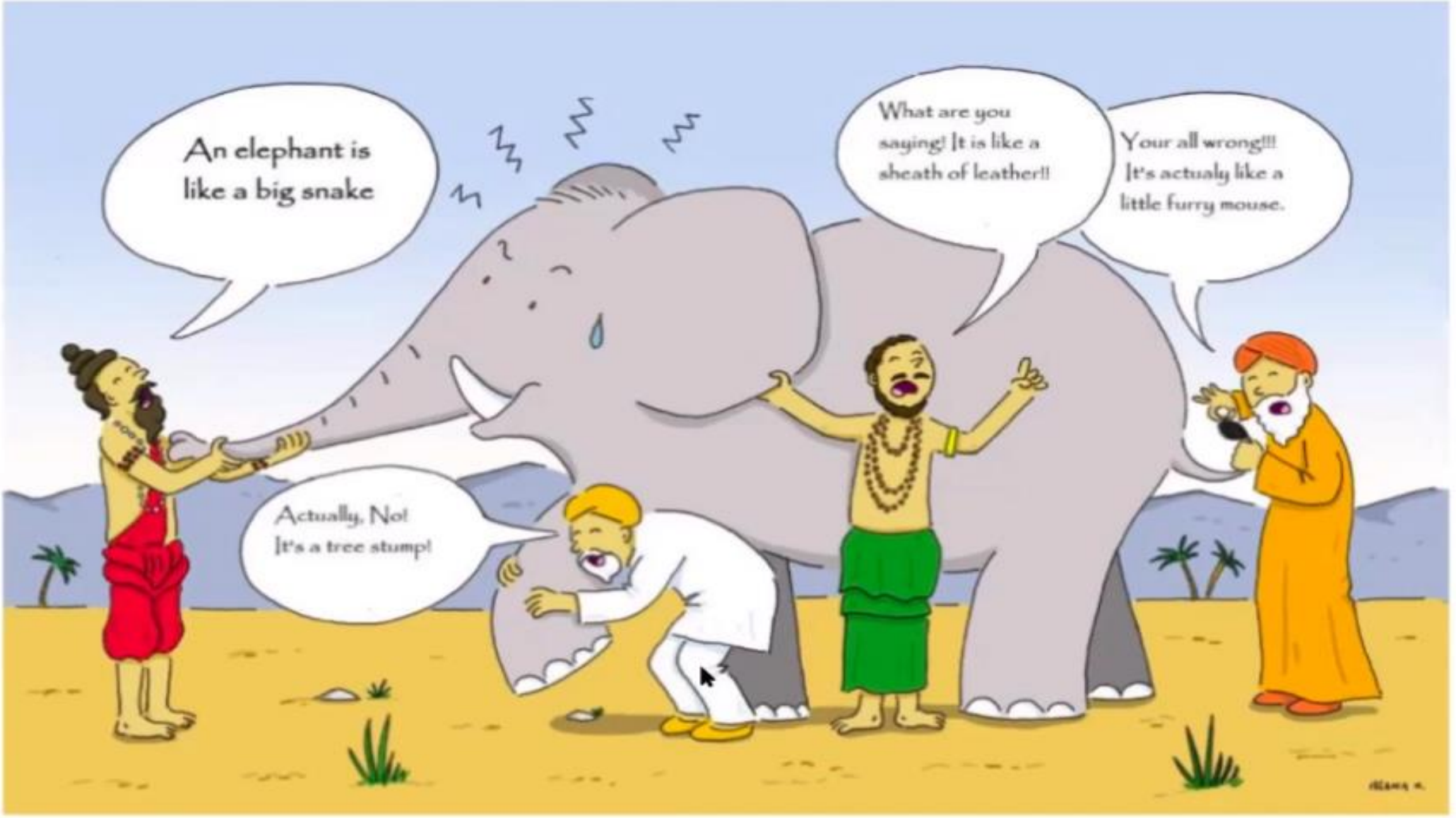

Convolution Neural Network (CNN)
Input
Pooling
Pooling
Pooling
Output
0.2
Horse
0.7
Zebra
0.1
Dog
Kernel
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Flatten Layer
SoftMax Activation Function
Feature Maps
Fully Connected Layer
Feature Extraction
Classification
Probabilistic Distribution

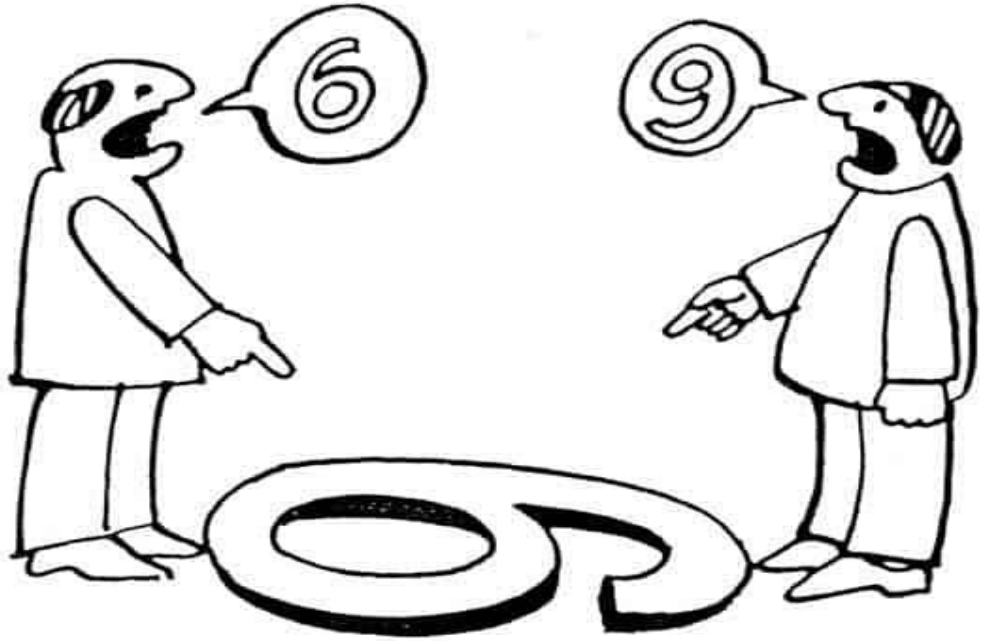
CNN

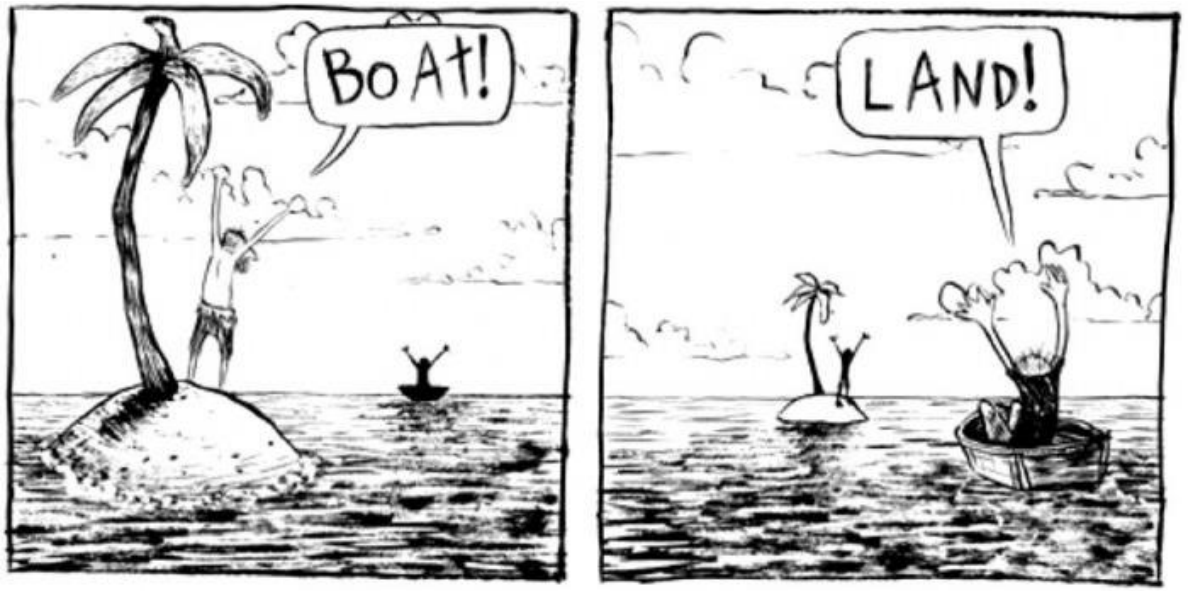

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

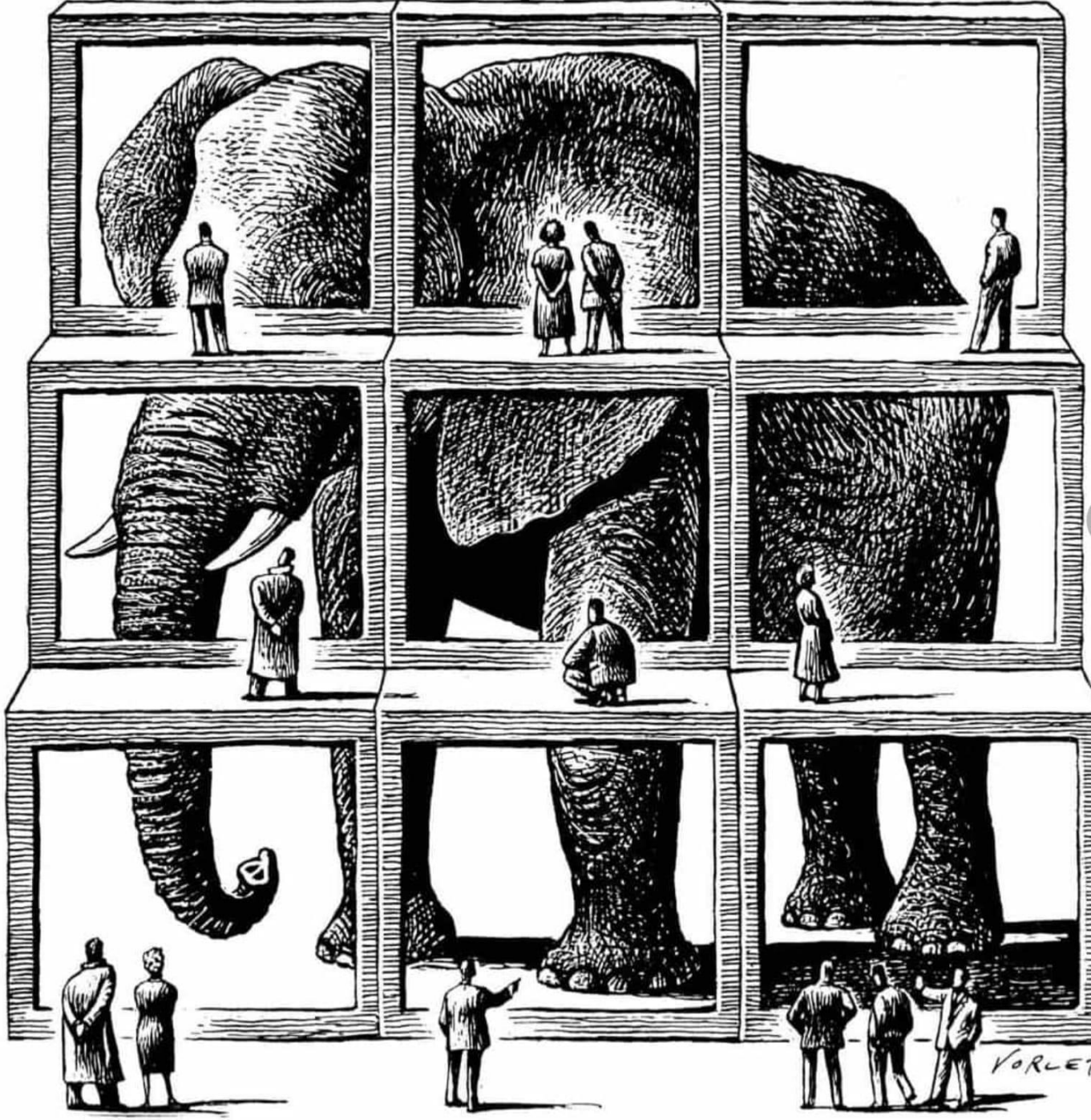
CONVOLUTIONAL LAYER


TIPO EXCELSIUS

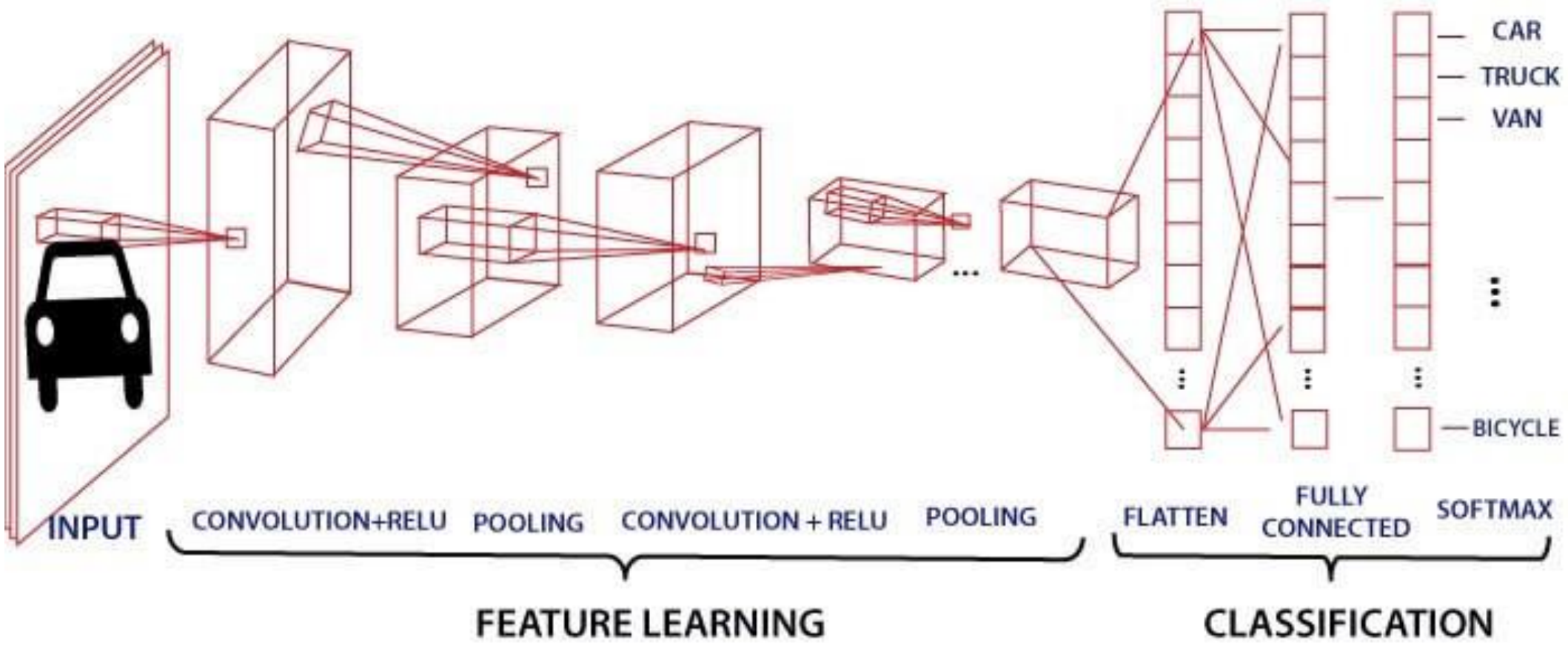

An elephant is like a big snake
Actually, No! It's a tree stump!
What are you saying! It is like a sheath of leather!!
Your all wrong!!! It's actualy like a little furry mouse.


6
9

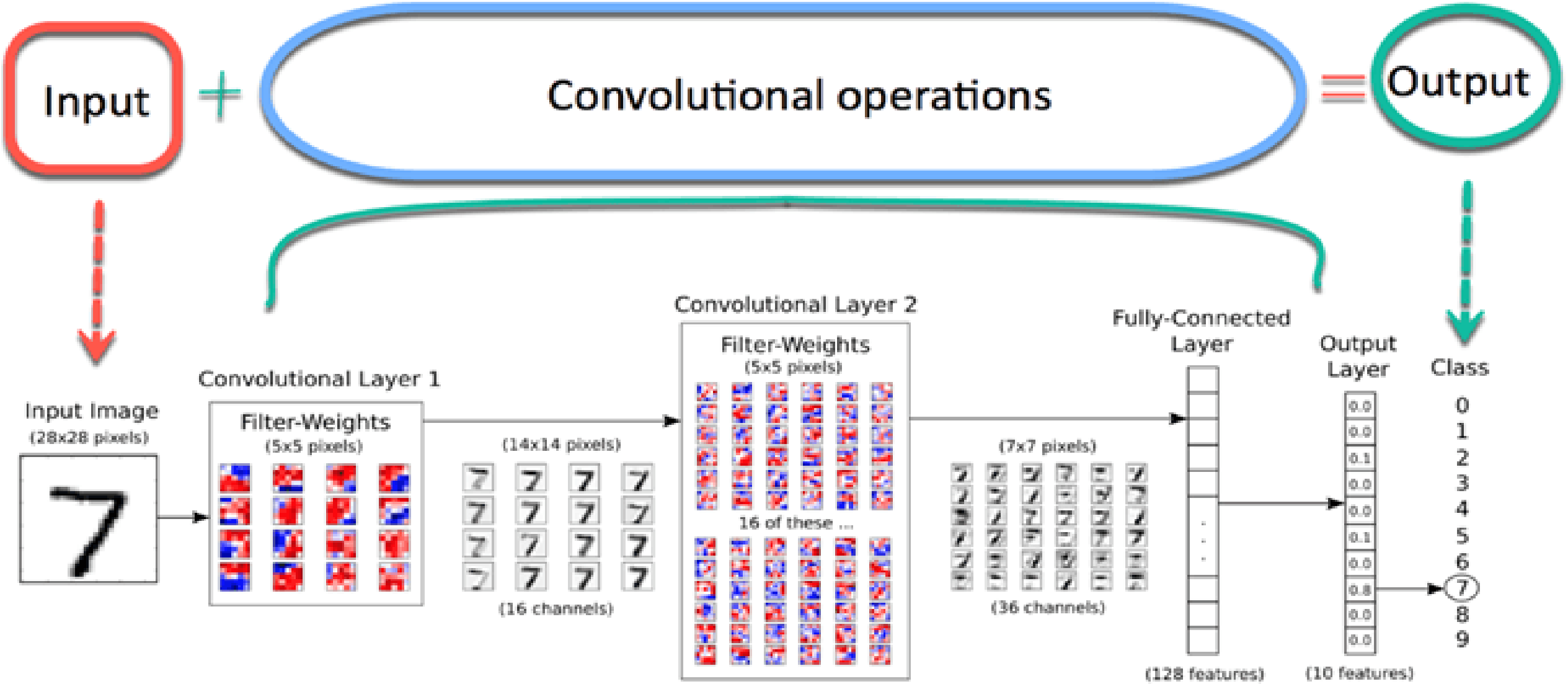

BOAT!
LAND!


VORLET

CNN


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION+RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

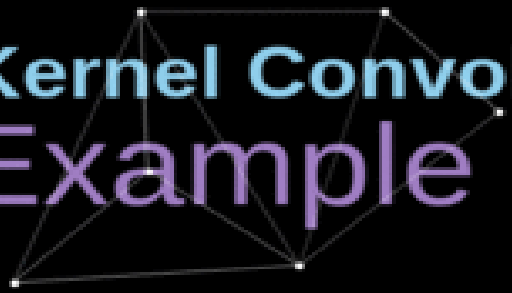
CNN


Input
Convolutional operations
Output
Input Image
(28x28 pixels)
Convolutional Layer 1
Filter-Weights
(5x5 pixels)
(14x14 pixels)
(16 channels)
Convolutional Layer 2
Filter-Weights
(5x5 pixels)
16 of these ...
(7x7 pixels)
(36 channels)
Fully-Connected Layer
(128 features)
Output Layer
(10 features)
Class
0
1
2
3
4
5
6
7
8
9

2D CNN ANIMATION

BASIC ARCHITECTURE

Kernel Convolution Example



Input Image

10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

*

Kernel

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

=

Feature Map

Resim 5x5

7 x1	2 x0	3 x-1	3	8
4 x1	5 x0	3 x-1	8	4
3 x1	3 x0	2 x-1	8	4
2	8	7	2	7
5	4	4	5	4

*

Filtre 3x3

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

=

Öznitelik Haritası

6		

7x1+4x1+3x1+
2x0+5x0+3x0+
3x-1+3x-1+2x-1
= 6

1.Adım

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

*

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

=

6	-9	

2x1+5x1+3x1+
3x0+3x0+2x0+
3x-1+8x-1+8x-1
= -9

2.Adım

Convolutional Layer

1	0	1
0	1	0
1	0	1

FİLTRE

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

Convolutional Layer

1	0	1
0	1	0
1	0	1

FİLTRE

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Convulutional Layer

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

*

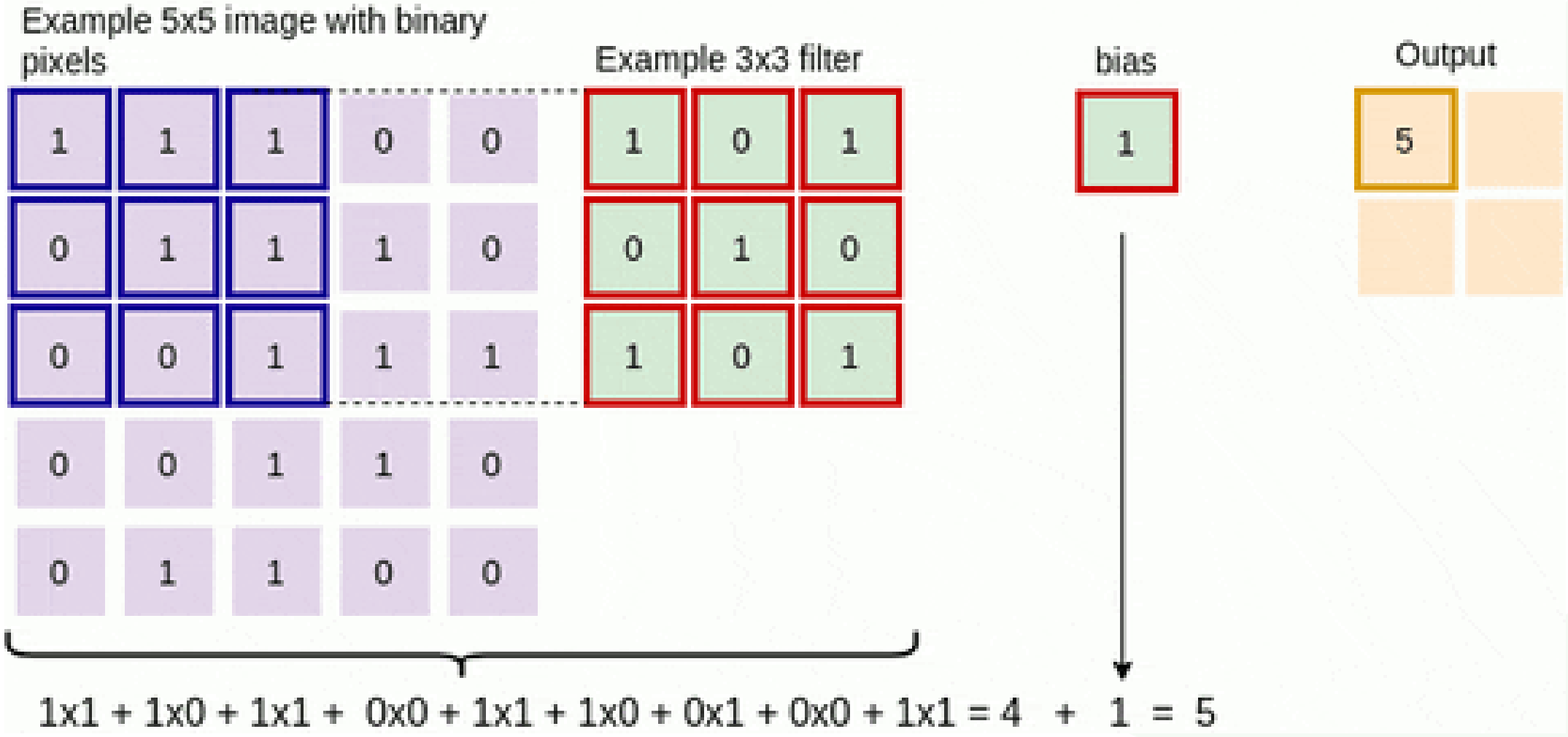
1	0	-1
1	0	-1
1	0	-1

=

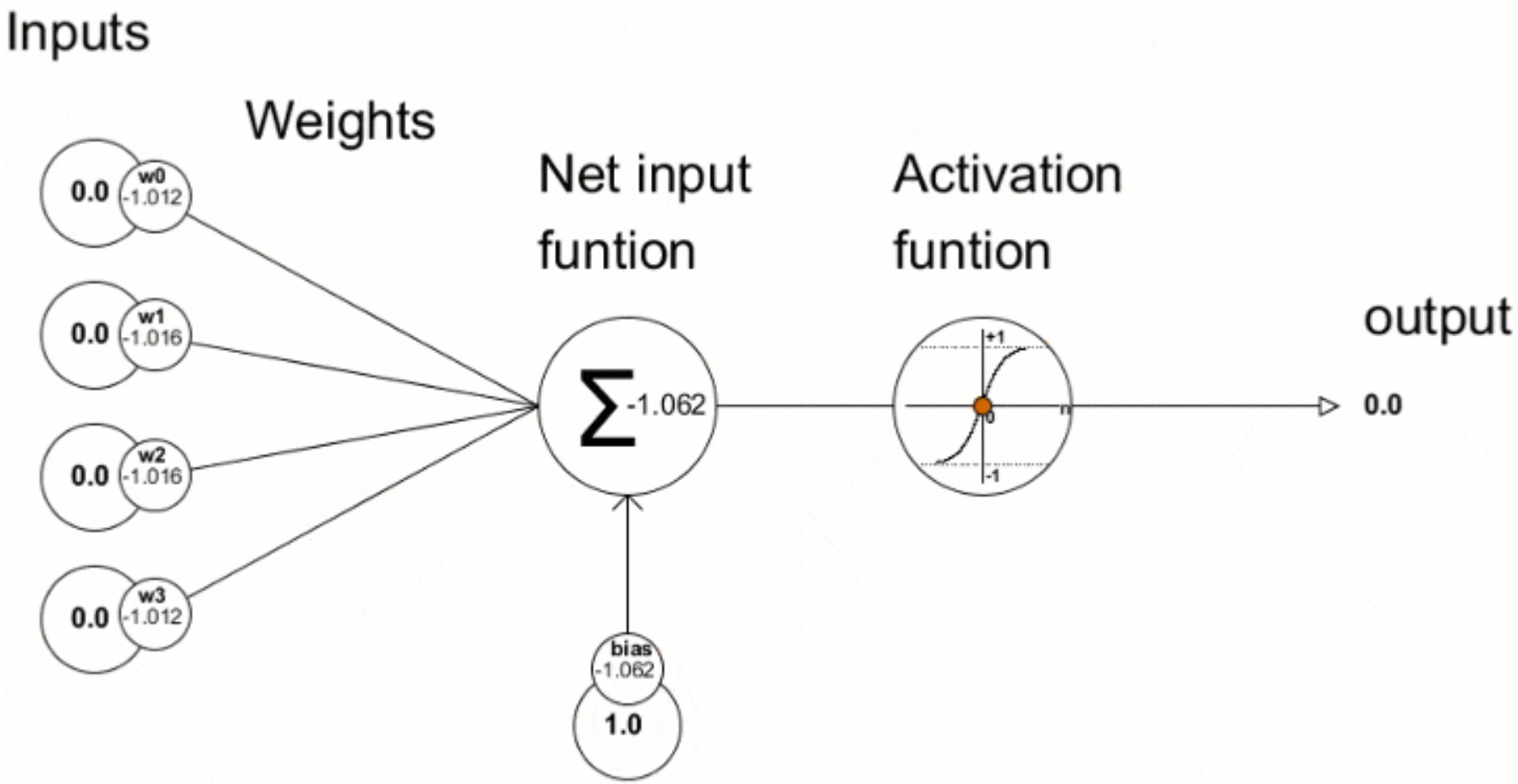
6		

7x1+4x1+3x1+
2x0+5x0+3x0+
3x-1+3x-1+2x-1
= 6

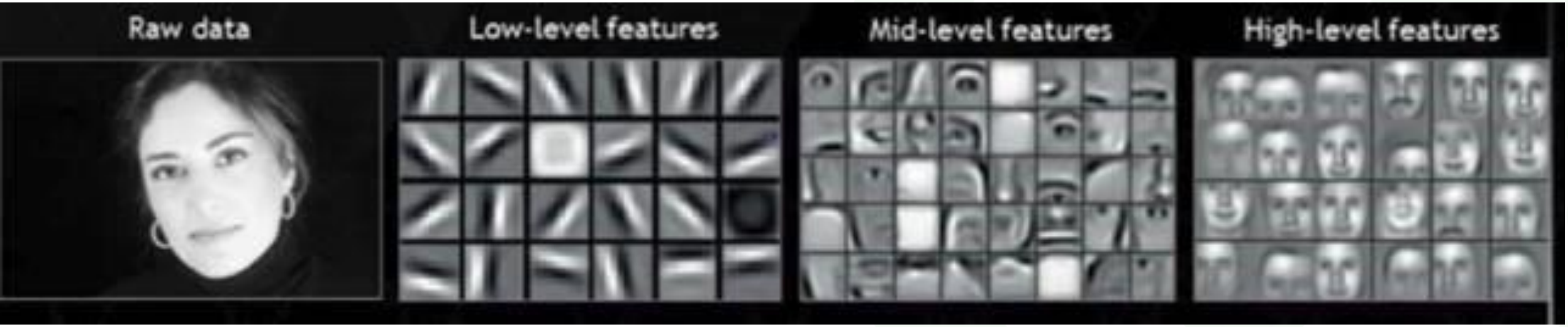
Convulutional Layer



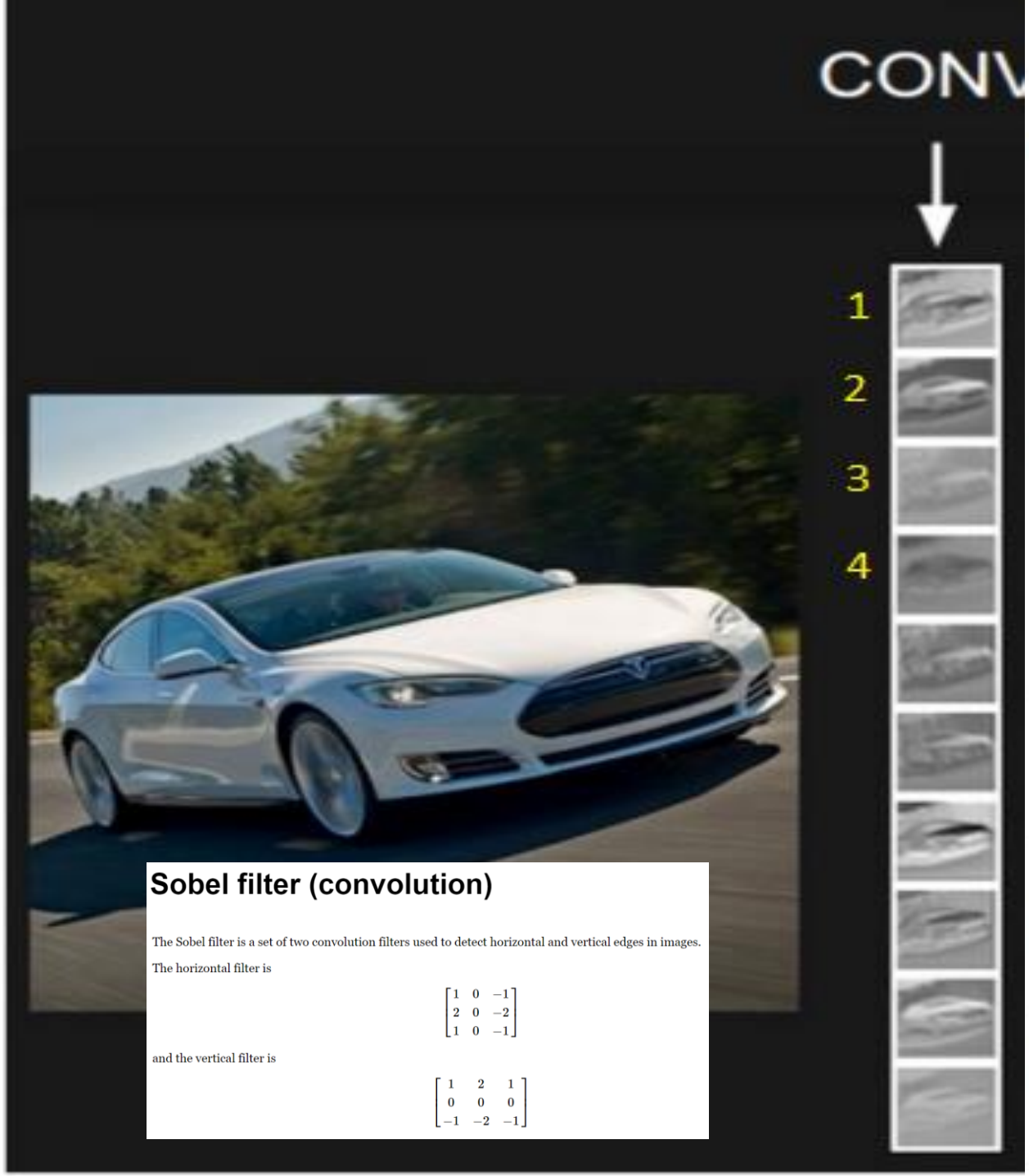
BIAS AND ACTIVATION FUNCTION



CNN


Raw data
Low-level features
Mid-level features
High-level features

Convulutional Layer



Sobel filter (convolution)

The Sobel filter is a set of two convolution filters used to detect horizontal and vertical edges in images.

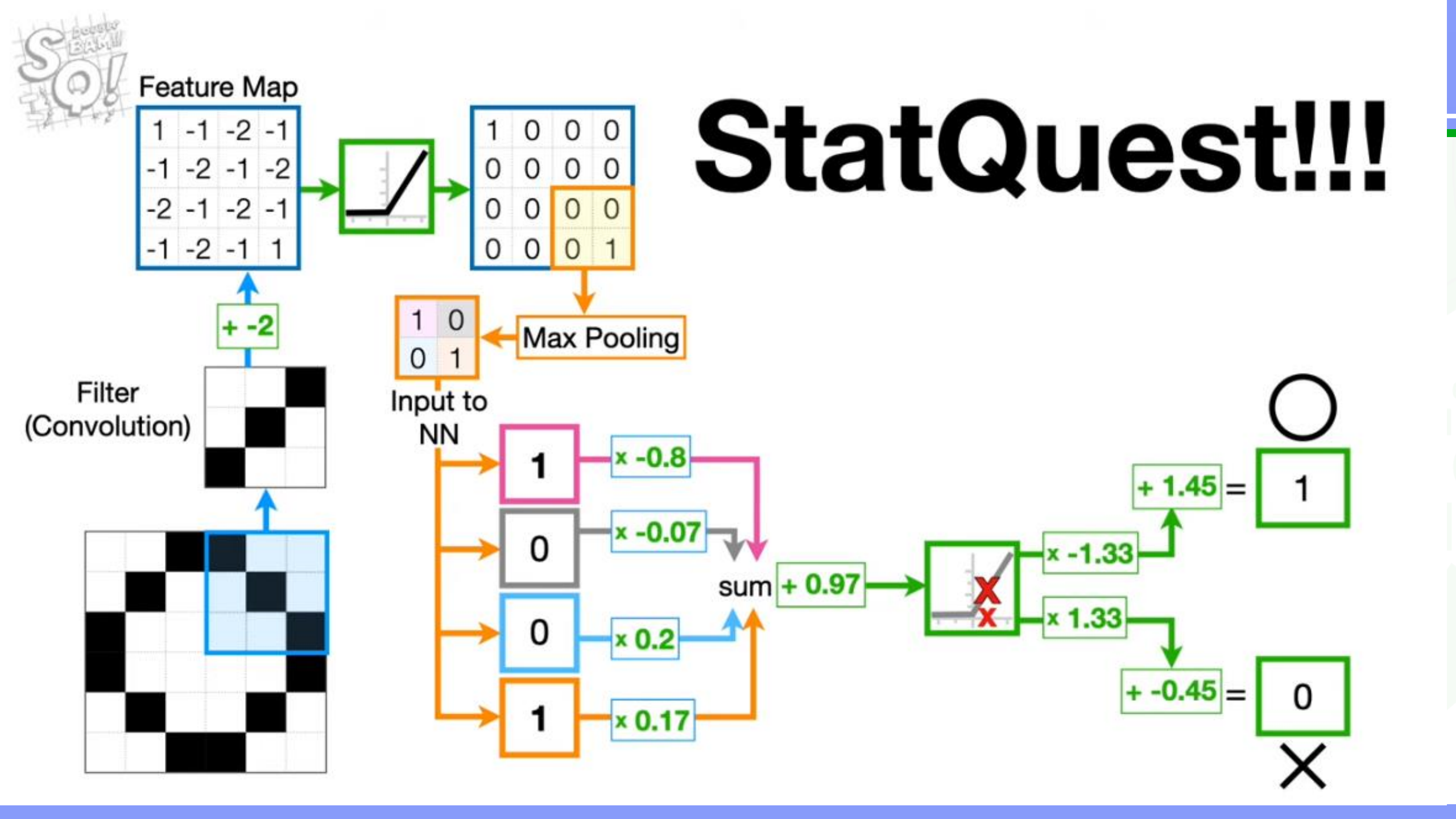
The horizontal filter is

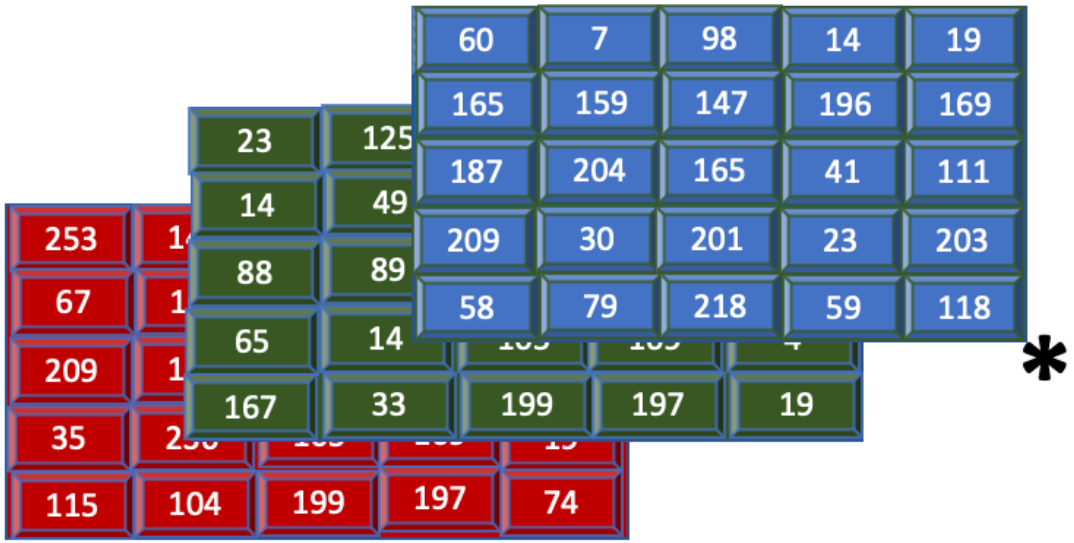
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

and the vertical filter is

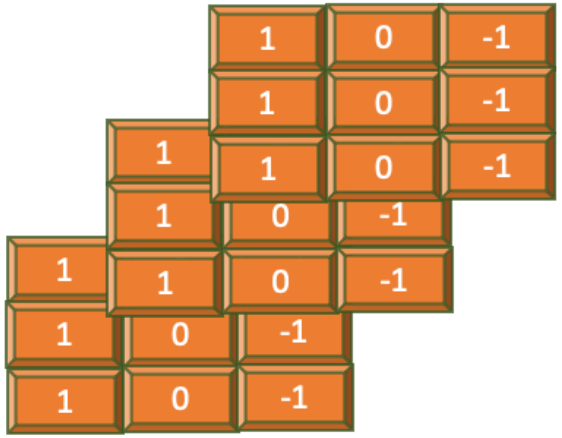
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

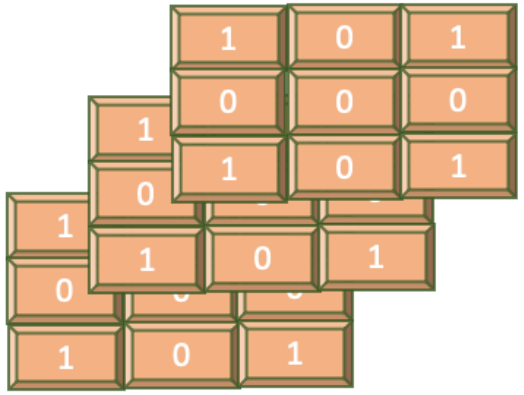
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
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)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

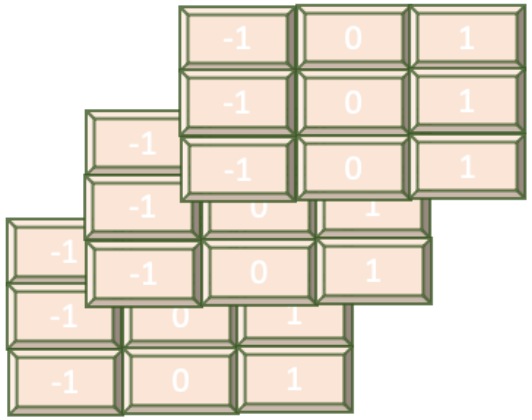

StatQuest!!!
Feature Map
1 -1 -2 -1
-1 -2 -1 -2
-2 -1 -2 -1
-1 -2 -1 1
+ -2
Filter
(Convolution)
1 0 0 0
0 0 0 0
0 0 0 0
0 0 0 1
Max Pooling
1 0
0 1
Input to
NN
1
0
0
1
x -0.8
x -0.07
x 0.2
x 0.17
sum
+ 0.97
x -1.33
x 1.33
+ 1.45 = 1
+ -0.45 = 0


253
67
209
35
115
104
199
197
74
23
14
88
65
167
33
199
197
19
60
7
98
14
19
165
159
147
196
169
187
204
165
41
111
209
30
201
23
203
58
79
218
59
118

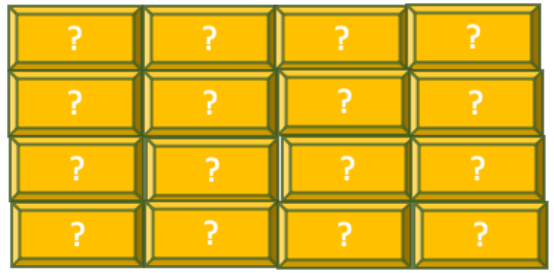
*

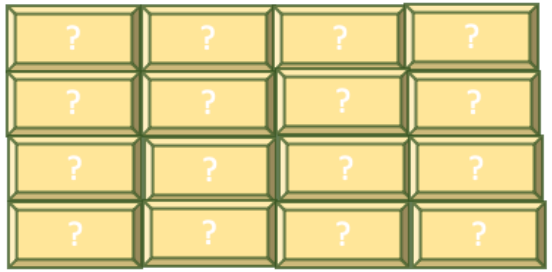

1 0 -1
1 0 -1
1 0 -1
1 0 -1
1 1 0 -1
1 0 -1
1 0 -1

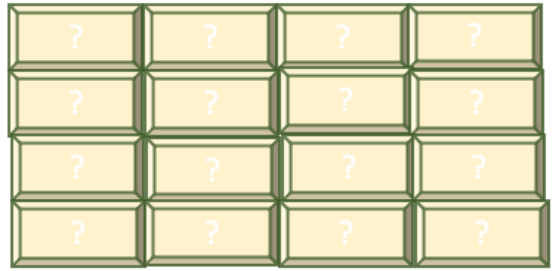

1 0 1
0 0 0
1 0 1
1 0 1
1 0 1
0 1 0
1 0 1

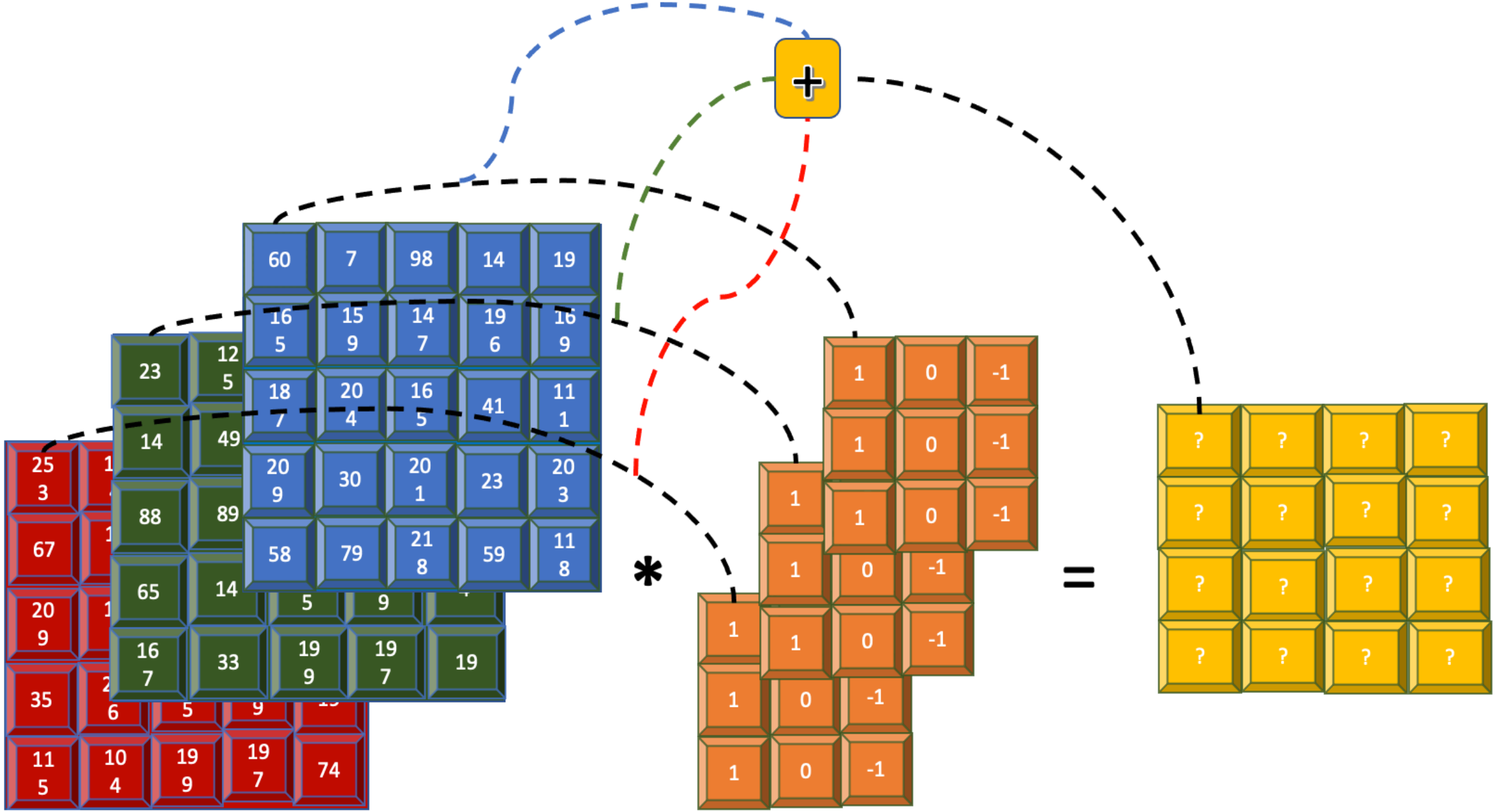

-1 0 1
-1 0 1
-1 0 1
-1 0 1
-1 -1 0 1
-1 0 1
-1 0 1

=

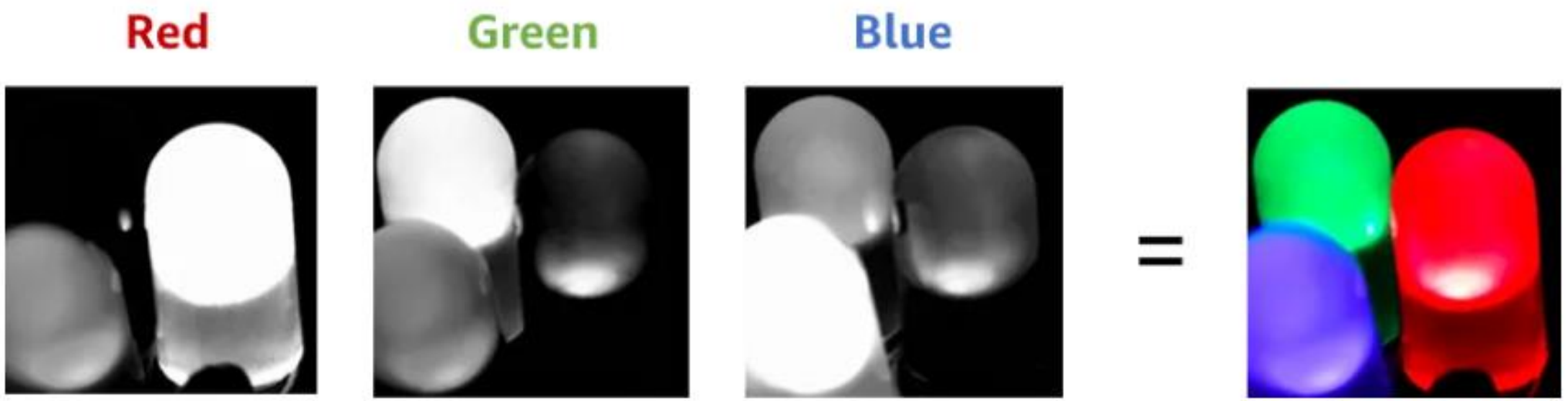

? ? ? ?
? ? ? ?
? ? ? ?
? ? ? ?

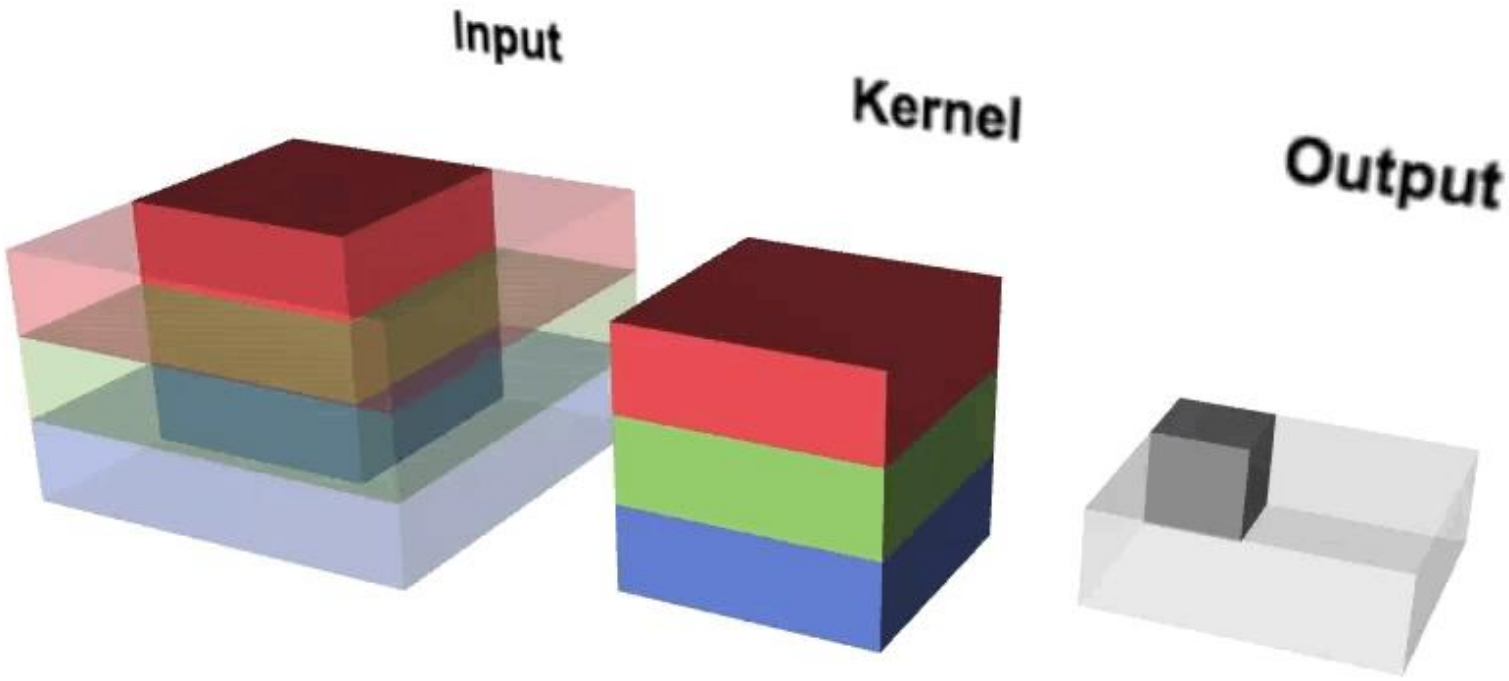

? ? ? ?
? ? ? ?
? ? ? ?
? ? ? ?


? ? ? ?
? ? ? ?
? ? ? ?
? ? ? ?

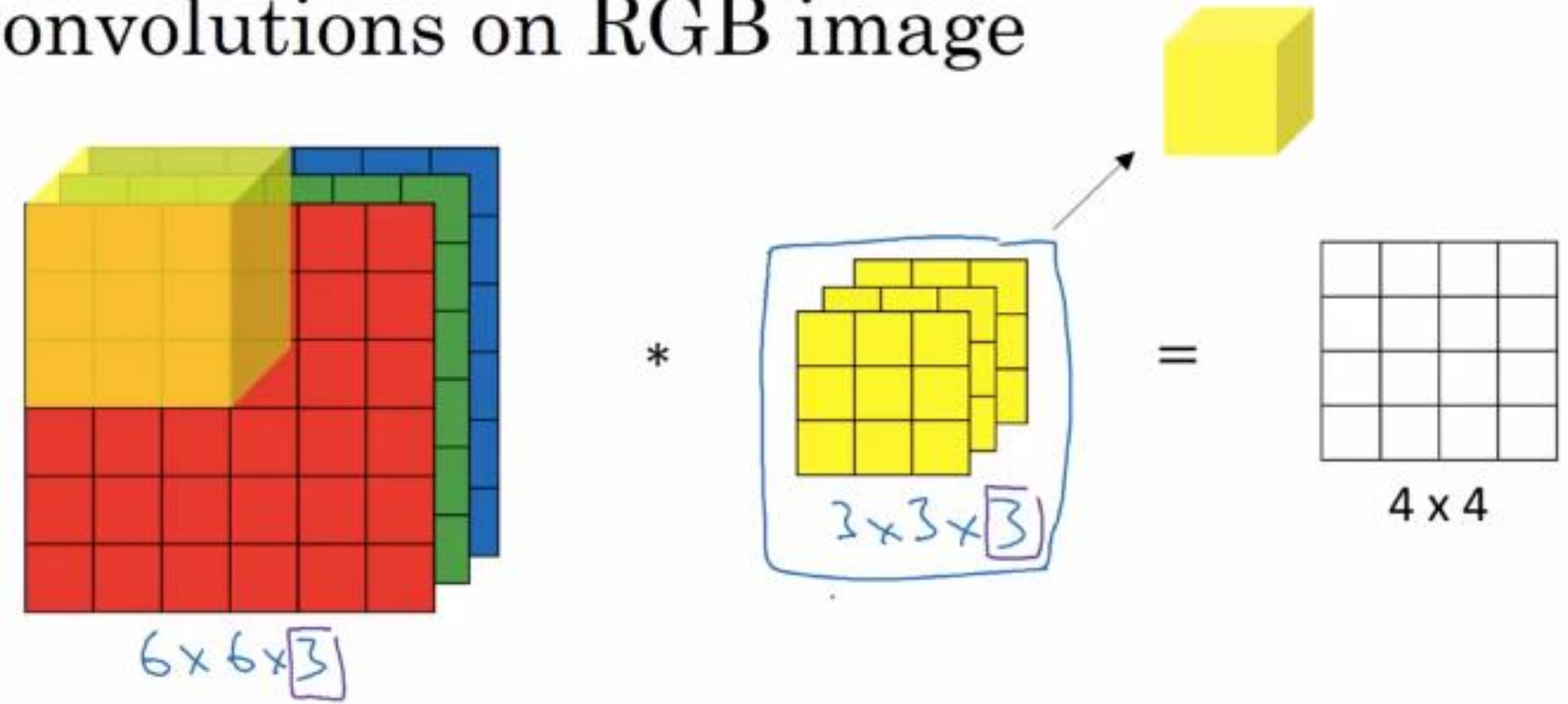

+
60 7 98 14 19
16 5 15 9 14 7 19 6 16 9
23 12 5
18 7 20 4 16 5 41 11 1
14 49
20 9 30 20 1 23 20 3
88 89
58 79 21 8 59 11 8
65 14
16 7 33 19 9 19 7 19
25 3
67
20 9
35
11 5 10 4 19 9 19 7 74
*
1 0 -1
1 0 -1
1 1 0 -1
1 1 0 -1
1 1 0 -1
1 0 -1
1 0 -1
=
? ? ? ?
? ? ? ?
? ? ? ?
? ? ? ?

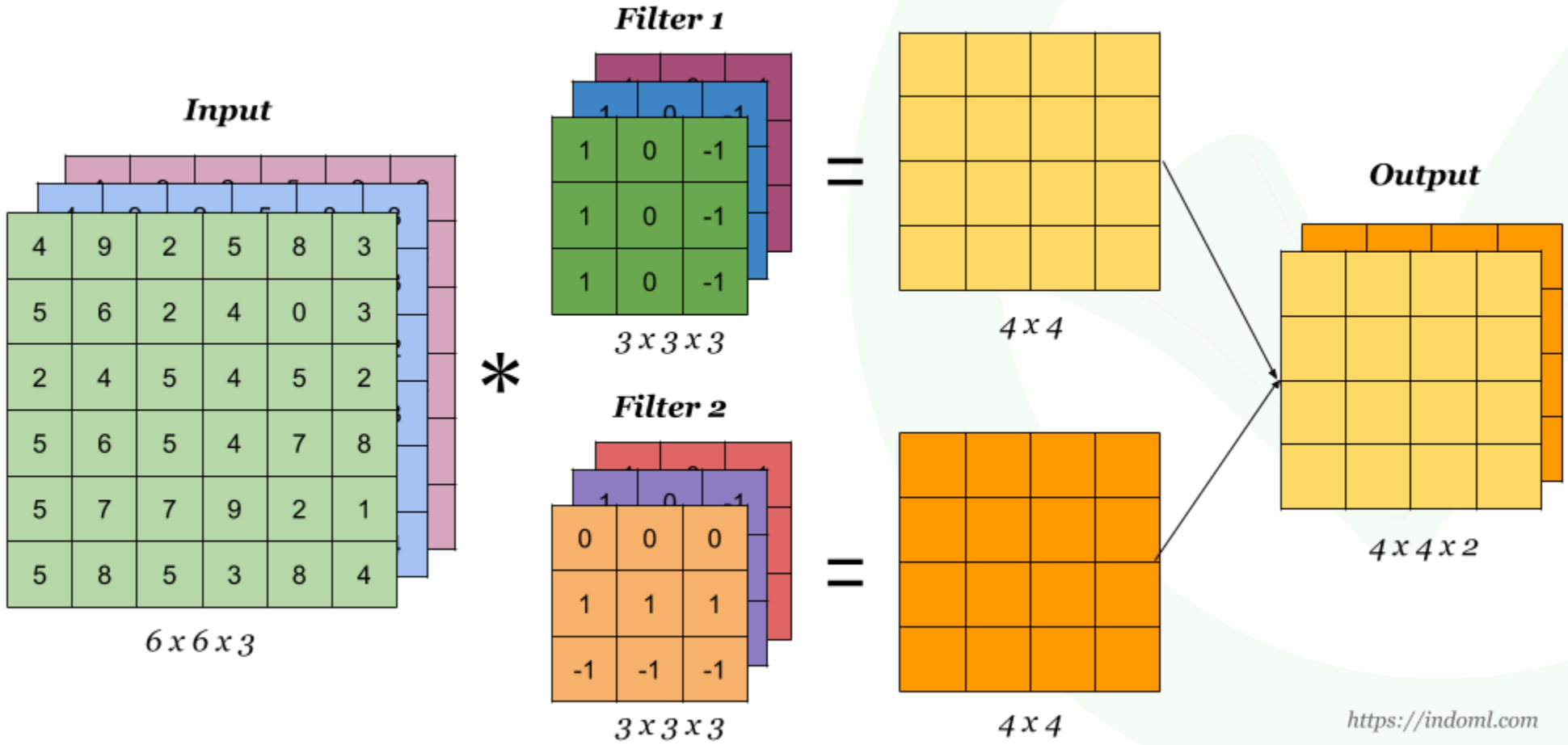
Conv2D with Multiple Input Channels

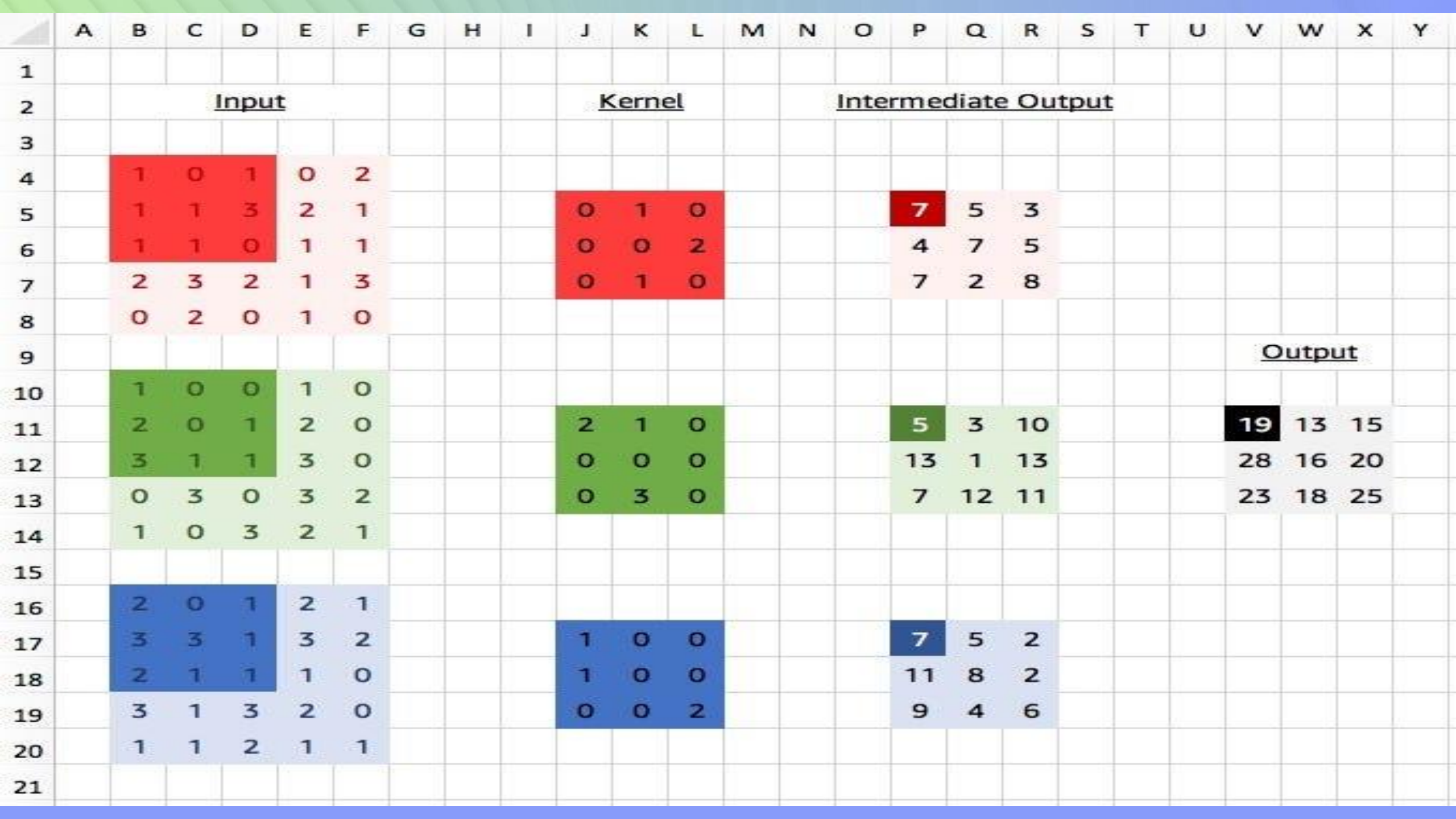



Input
Kernel
Output

Convolutions on RGB image




Input
4 9 2 5 8 3
5 6 2 4 0 3
2 4 5 4 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
6 x 6 x 3
*
Filter 1
1 0 -1
1 0 -1
1 0 -1
3 x 3 x 3
Filter 2
0 0 0
1 1 1
-1 -1 -1
3 x 3 x 3
=
4 x 4
=
4 x 4
Output
4 x 4 x 2
https://indoml.com


Input
Kernel
Intermediate Output
Output
1 0 1 0 2
1 1 3 2 1
1 1 0 1 1
2 3 2 1 3
0 2 0 1 0
1 0 0 1 0
2 0 1 2 0
3 1 1 3 0
0 3 0 3 2
1 0 3 2 1
2 0 1 2 1
3 3 1 3 2
2 1 1 1 0
3 1 3 2 0
1 1 2 1 1
0 1 0
0 0 2
0 1 0
2 1 0
0 0 0
0 3 0
1 0 0
1 0 0
0 0 2
7 5 3
4 7 5
7 2 8
5 3 10
13 1 13
7 12 11
7 5 2
11 8 2
9 4 6
19 13 15
28 16 20
23 18 25

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

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

x[:,:,1]

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

x[:,:,2]

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

Filter W0 (3x3x3)

w0[:,:,0]

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

w0[:,:,1]

0	0	1
0	-1	0
-1	0	1

w0[:,:,2]

-1	0	0
-1	1	1
0	0	0

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

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

w1[:,:,1]

0	-1	0
0	0	0
0	0	1

w1[:,:,2]

-1	1	1
-1	1	0
0	1	0

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

o[:,:,0]

3	1	0
8	-6	-1
1	-5	-4

o[:,:,1]

2	-2	-3
4	0	1
5	0	1

toggle movement

PADDING AND STRIDE

Padding

Padding

Convolutional işleminden sonra boyut sayısı azaldığı için bir miktar veri kaybı söz konusudur. Evrişim işleminden sonra oluşabilecek boyut farkı yönetilebilir. Bunun için giriş matrisine ekstra pixellerle sağlanabilir.

Bu işlem iki farklı yöntemle yapılabilir. Örnek olarak tüm pixeller 0 ile doldurulabilir yada komşu kenarla doldurulabilir.

- Giriş matrisi nxn,
- Filtre (ağırlık) matrisi (fxf)

0	0	0	0	0	0
0	2	1	3	5	0
0	1	1	4	4	0
0	6	2	7	3	0
0	1	6	5	5	0
0	0	0	0	0	0

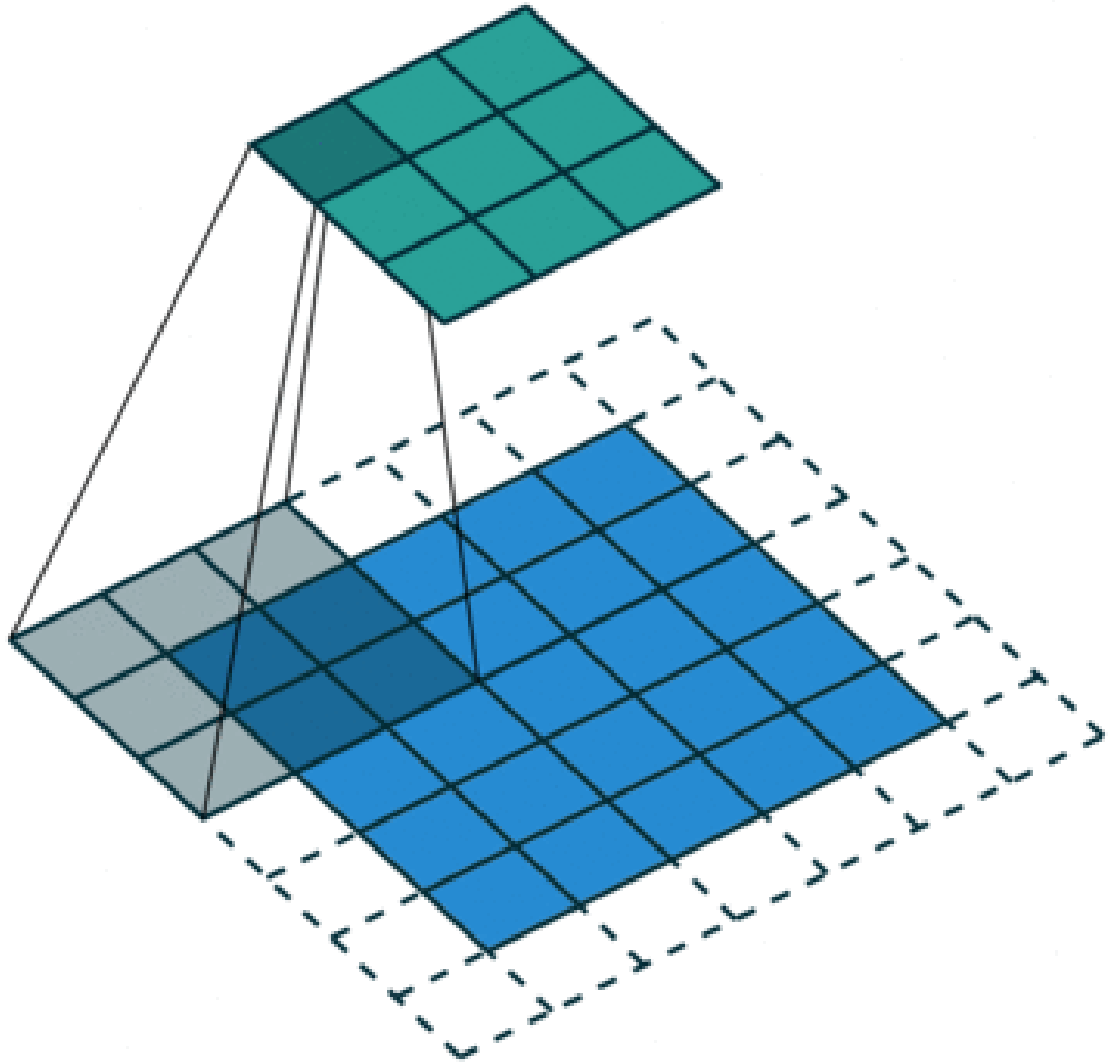
$$p = \frac{(f-1)}{2}$$

2	2	1	3	5	5
2	2	1	3	5	5
1	1	1	4	4	4
6	6	2	7	3	3
1	1	6	5	5	5
1	1	6	5	5	5

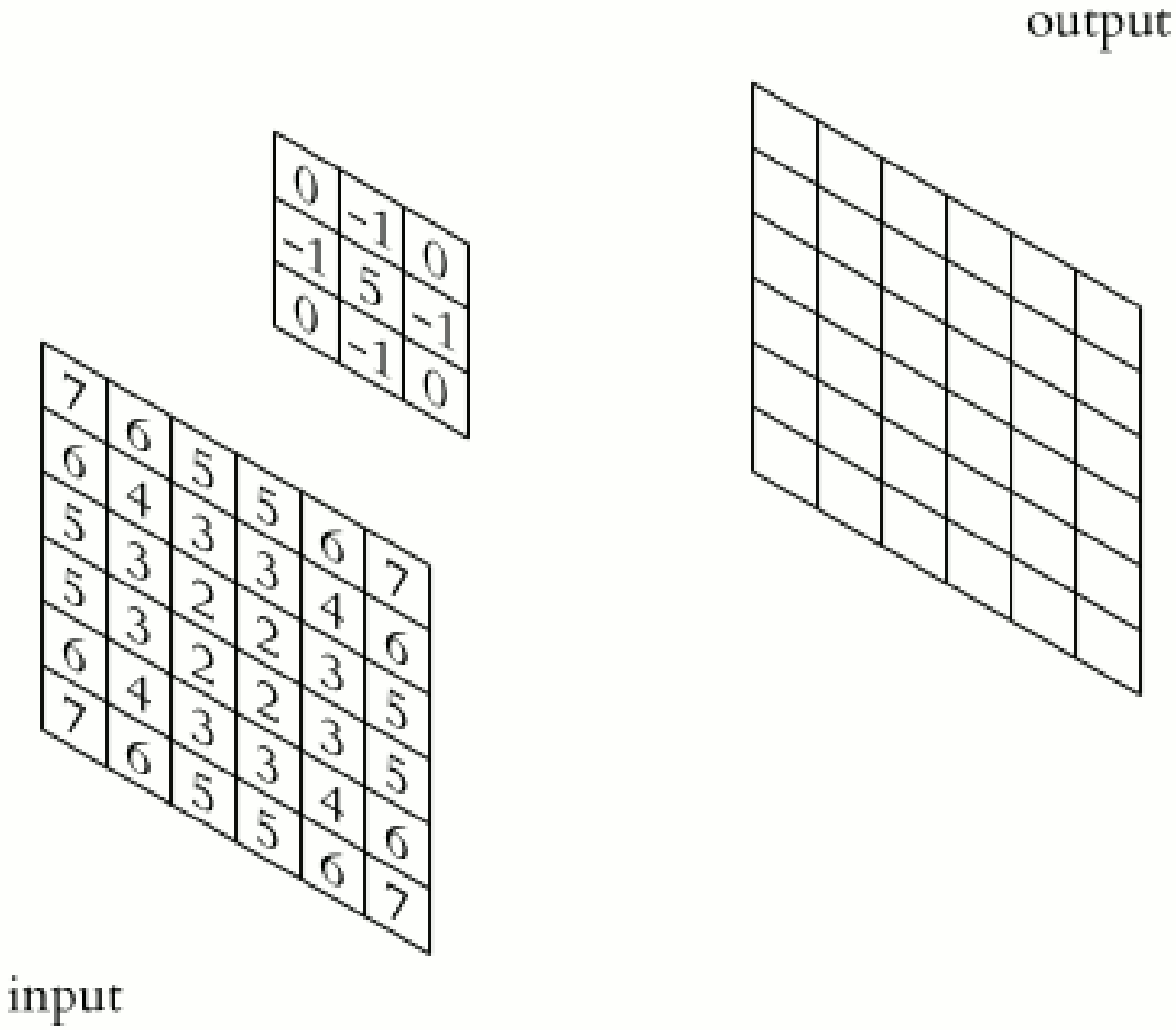
PADDING

Padding works by extending the area of which a convolutional neural network processes an image.

1	6	5
7	10	9
7	10	8



Convulutional Layer



Convulutional Layer

Padding

Input

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

Kernel

0	1
2	3

=

Output

0	3	8	4
9	19	25	10
21	37	43	16
6	7	8	0

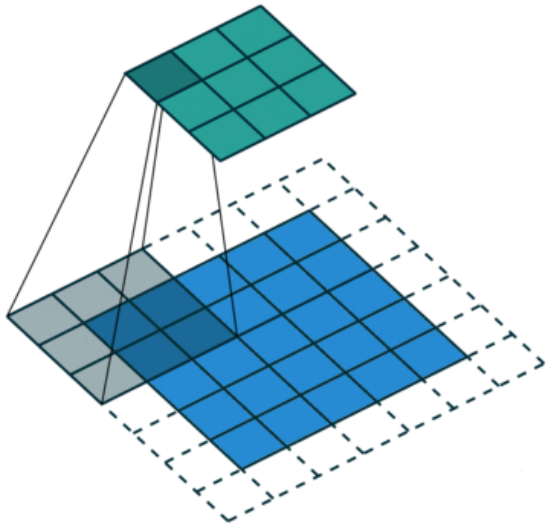
PADDING

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

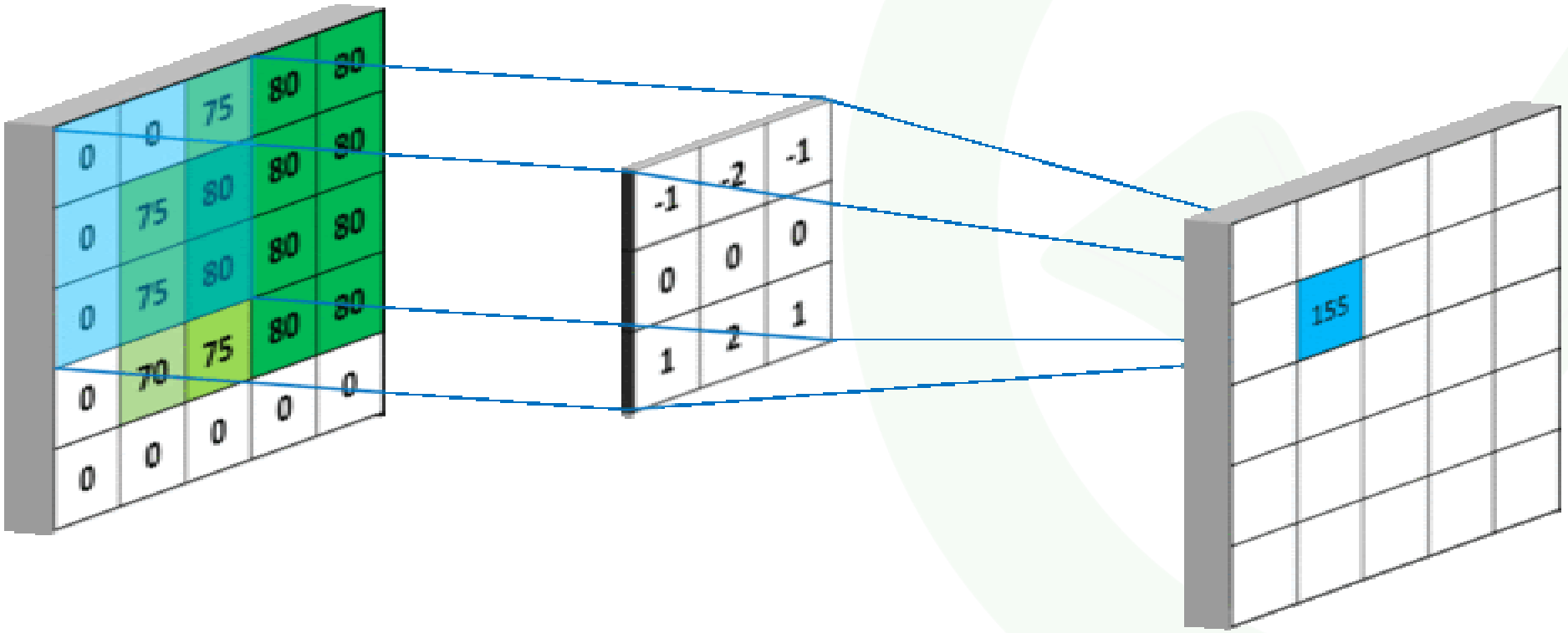
Kernel

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

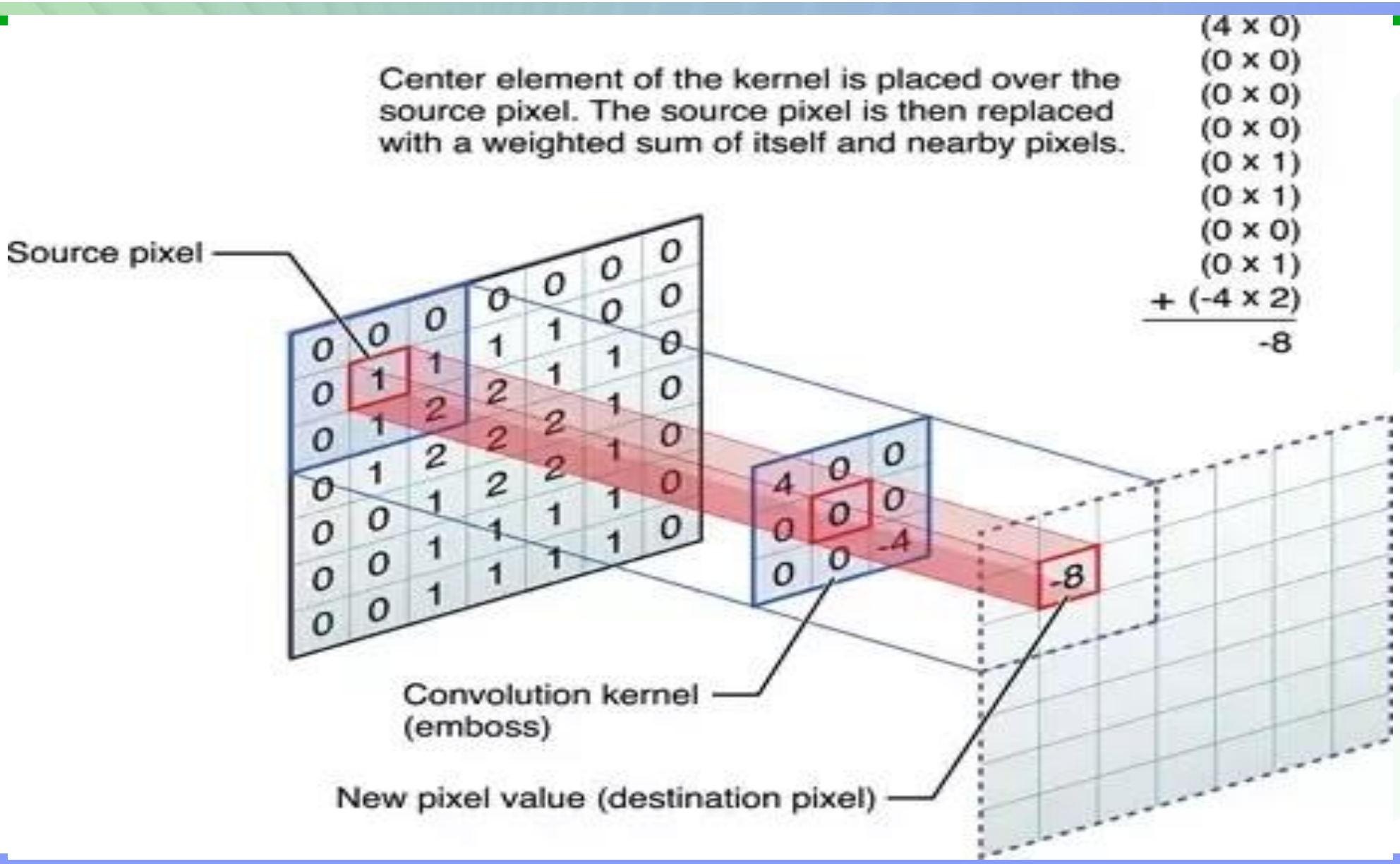
114				



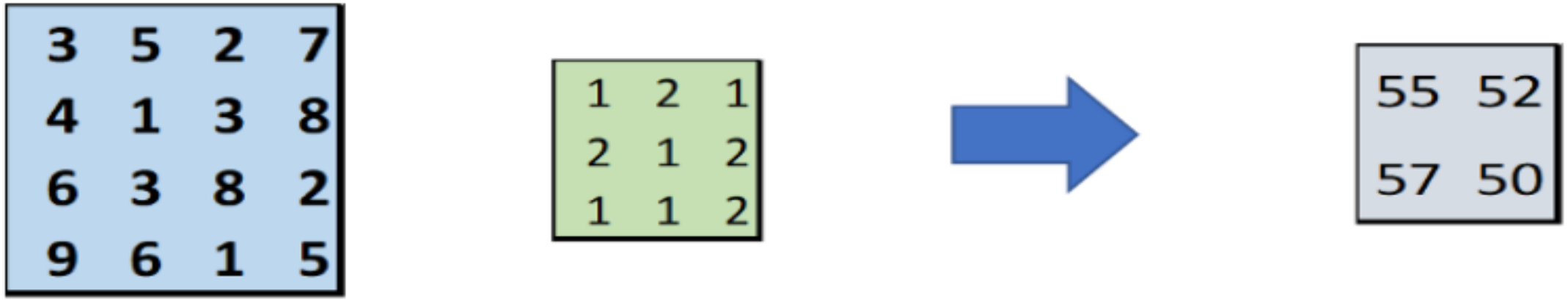
CNN


0 0 75 80 80
0 75 80 80 80
0 75 80 80 80
0 70 75 80 80
0 0 0 0 0
-1 -2 -1
0 0 0
1 2 1
155

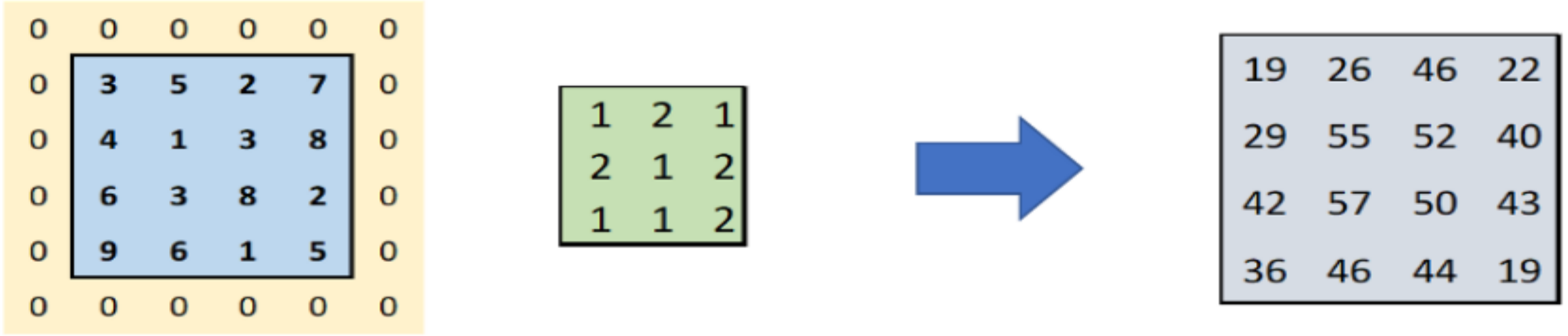
CNN


Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Padding: “valid”



Padding: “same”



STRIDE

Padding = Same

Filter

1	0
0	0.5

Stride X

Input

0	0	0	0	0	0
0	1	0	0.5	0.5	0
0	0	0.5	1	0	0
0	0	1	0.5	1	0
0	1	0.5	0.5	1	0
0	0	0	0	0	0

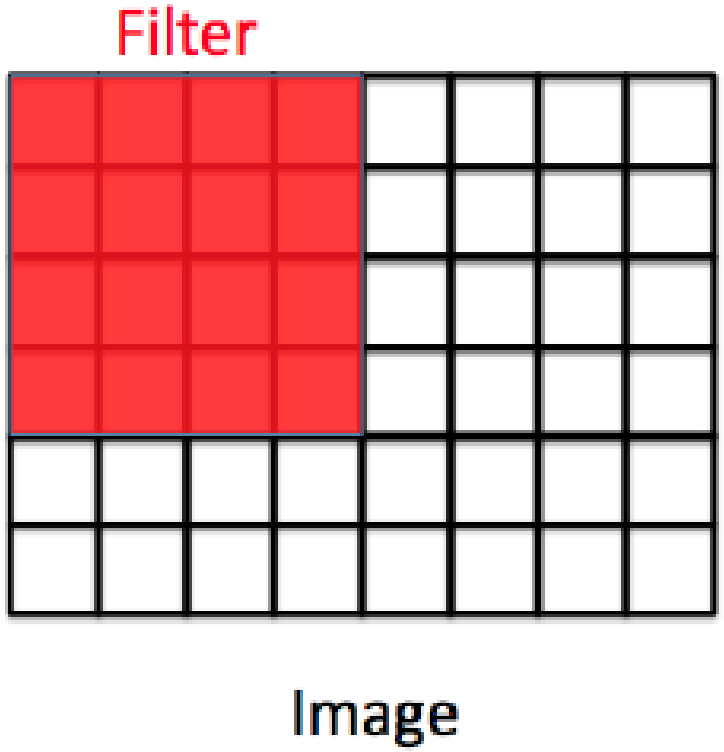
Stride Y

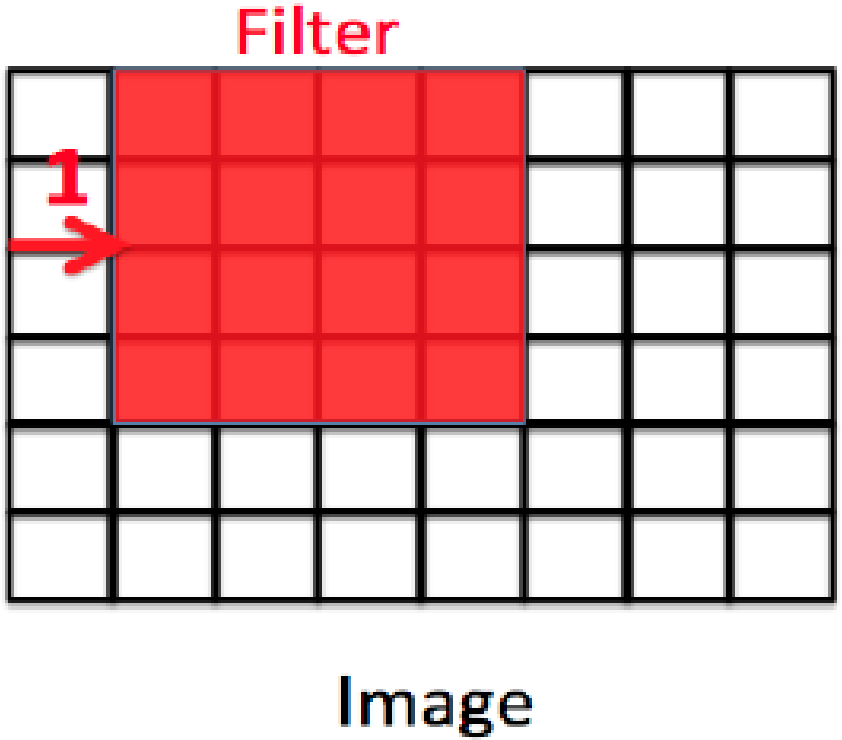
Output

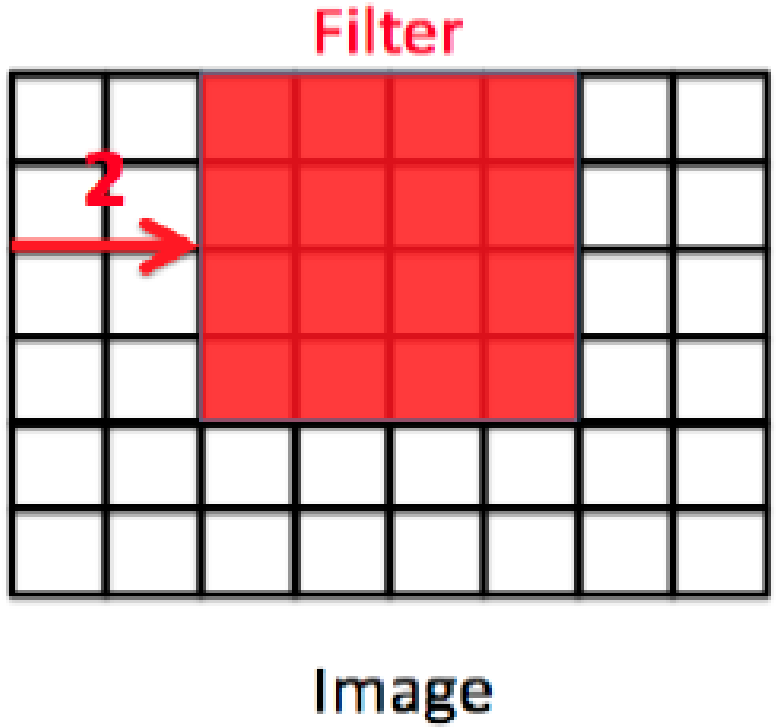
0.5	0	0.25	0.25
0	1.25	0.5	0.5
0	0.5	0.75	1.5
0.5	0.25	1.25	1

outDim = (inpDim)/strideDim

STRIDE







Convulutional Layer

28x28 piksel boyutunda bir resmi 3x3 boyutunda bir filtreyi 0 birim padding ve 1 birim stride ile işlediğinizde, çıkış boyutu kaç olur?

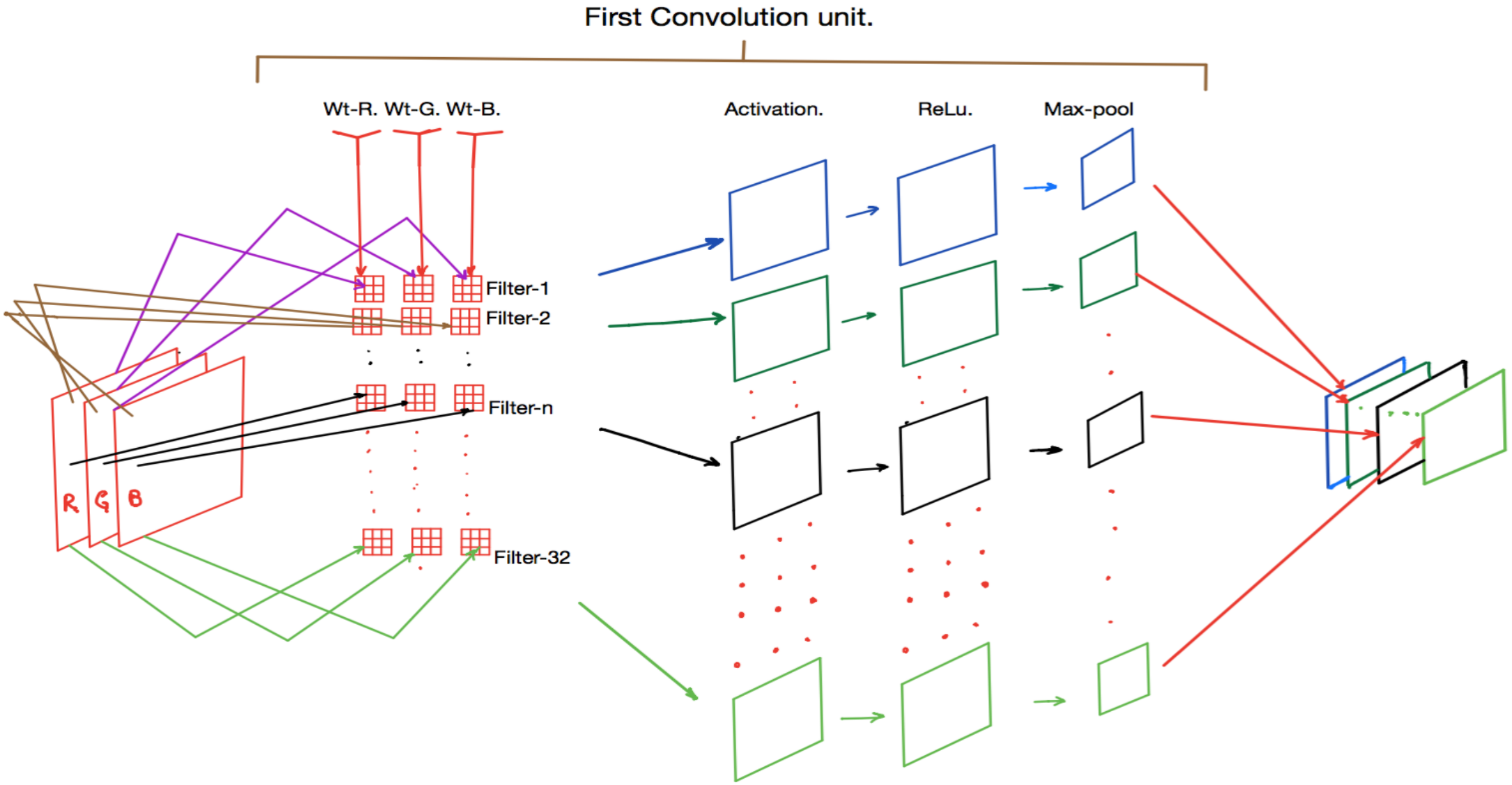
Çıkış boyutu = (Girdi boyutu - filtrenin boyutu) / stride + 1
Çıkış boyutu = (28 - 3) / 1 + 1
Çıkış boyutu = 26

çıkış boyutu 26x26 piksel olur.

NON-LINEARITY

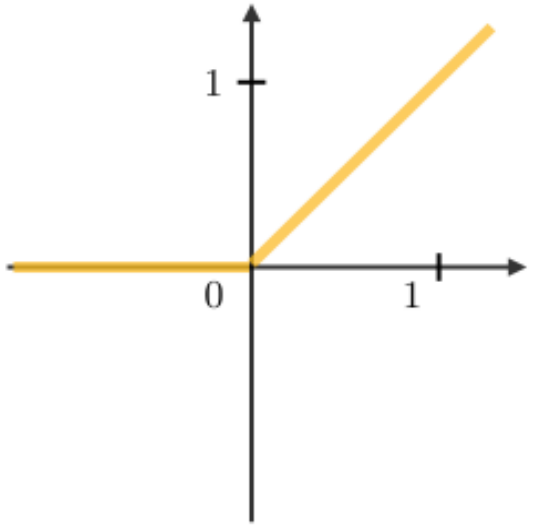
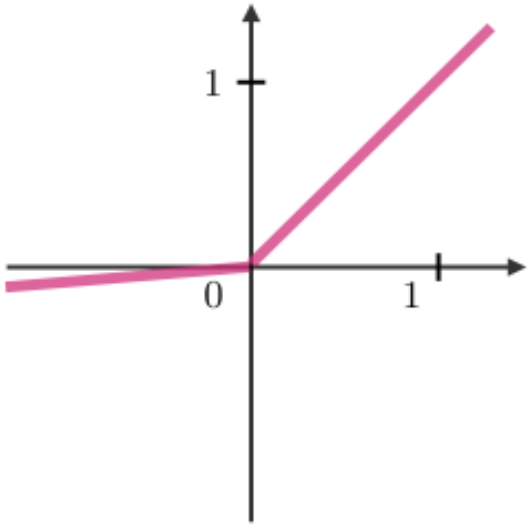
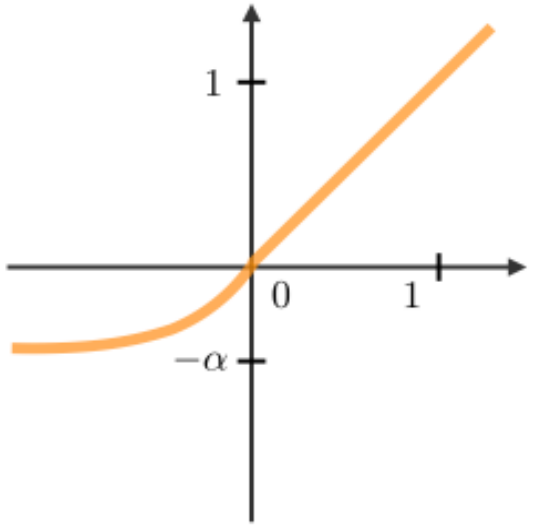
PERSONAL RECOVERY IS NOT A LINEAR PROCESS

Non-linearity



Commonly used activation functions

❒ **Rectified Linear Unit** — The rectified linear unit layer (ReLU) is an activation function g that is used on all elements of the volume. It aims at introducing non-linearities to the network. Its variants are summarized in the table below:

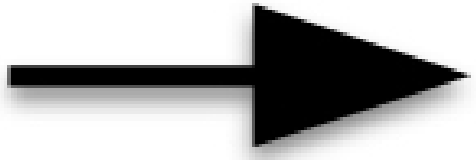
ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ with $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ with $\alpha \ll 1$
		
• Non-linearity complexities biologically interpretable	• Addresses dying ReLU issue for negative values	• Differentiable everywhere

Non-linearity

ReLU Layer

Filter 1 Feature Map

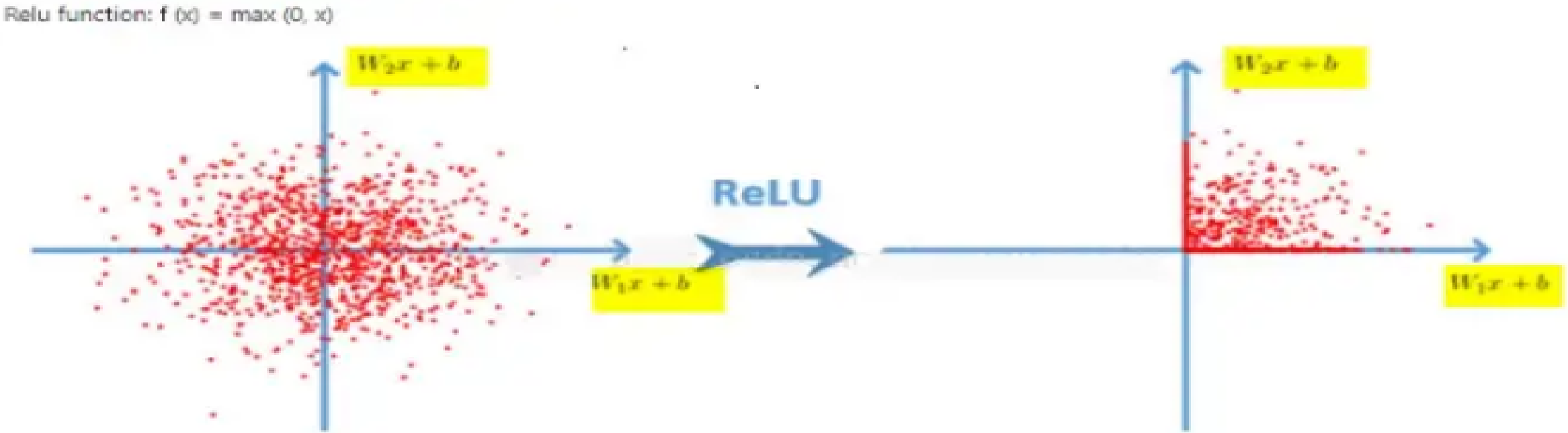
9	3	5	-8
-6	2	-3	1
1	3	4	1
3	-4	5	1



9	3	5	0
0	2	0	1
1	3	4	1
3	0	5	1

1. ReLU

RelU based deep convolutional networks are trained several times faster than tanh and sigmoid- based networks. After RelU, the value of the function has no range like tan h or the sigmoid function so a normalization is usually is done afterwards.



Non-linearity

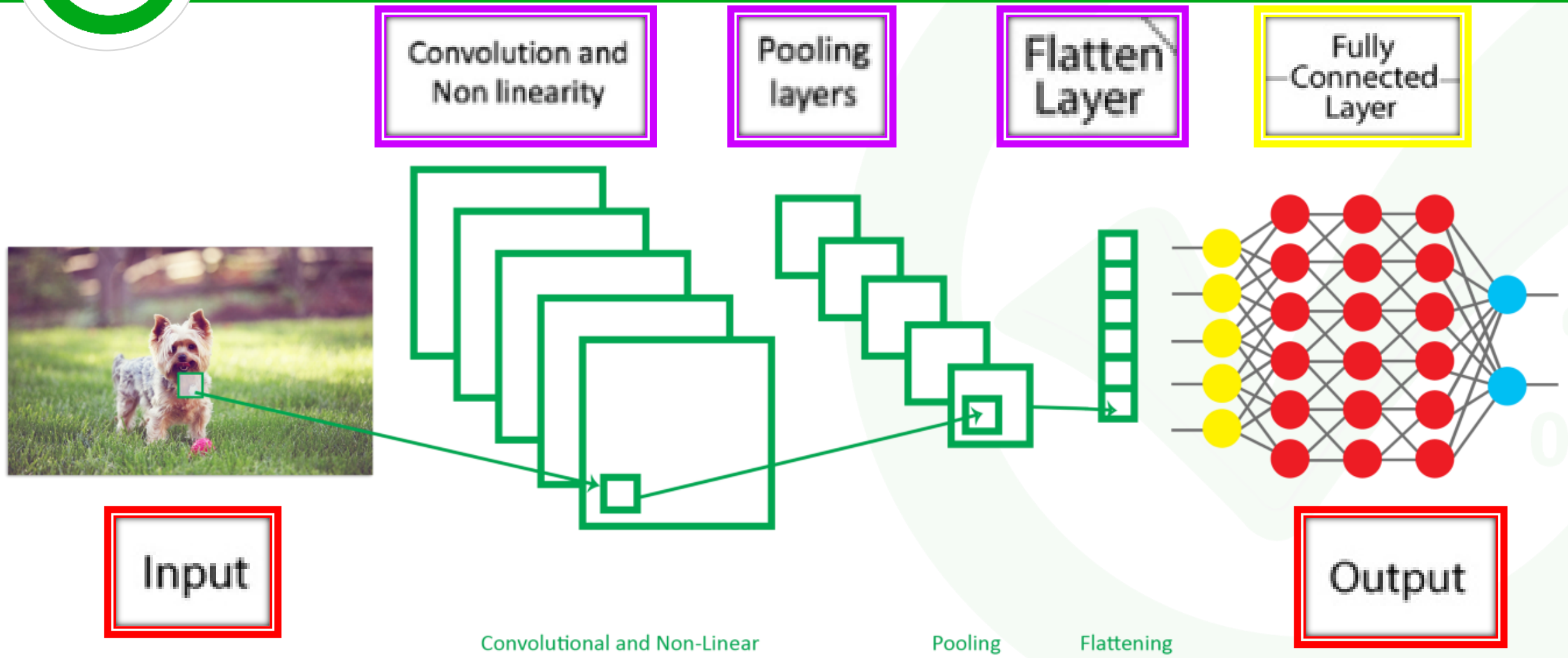
Original Image

Feature Map

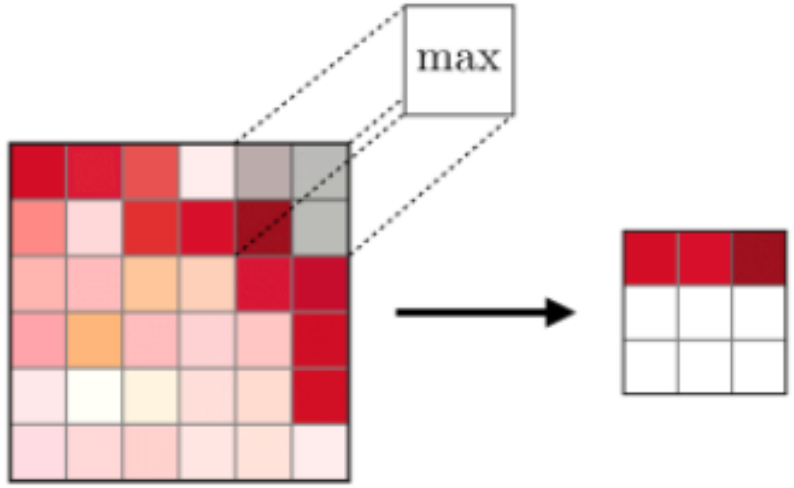
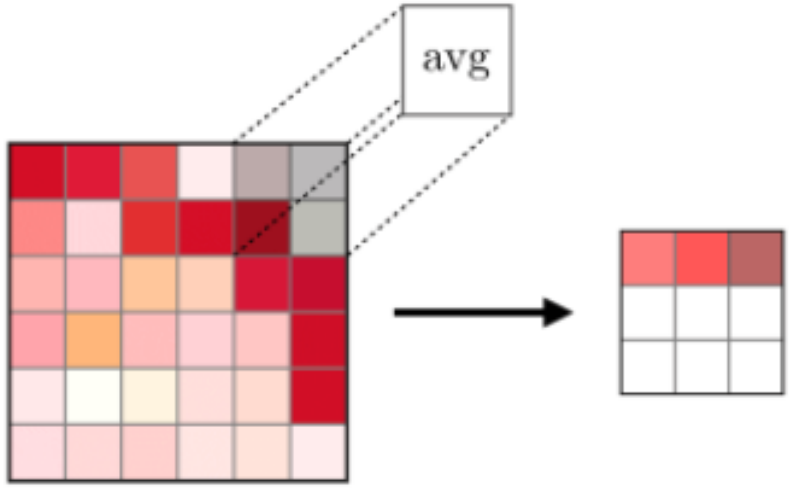
Non-Linear

POOLING LAYER

CNN


Convolution and Non linearity
Pooling layers
Flatten Layer
Fully Connected Layer
Input
Output
Convolutional and Non-Linear
Pooling
Flattening

Pooling Layer

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

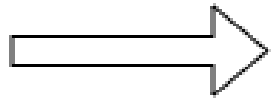
Pooling Layer

Max Pooling

4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4

Max([4, 3, 1, 3]) = 4

4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4



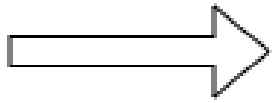
4	8
6	9

Pooling Layer

Average Pooling

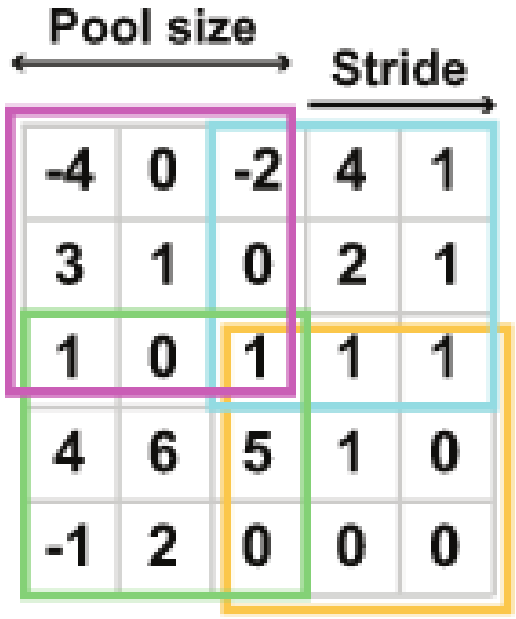
4	3	1	5
1	3	4	8
4	5	4	3
6	5	9	4

Avg([4, 3, 1, 3]) = 2.75



2.8	4.5
5.3	5.0

Pooling Layer



Features

Max Pooling

3	4
6	5

Min Pooling

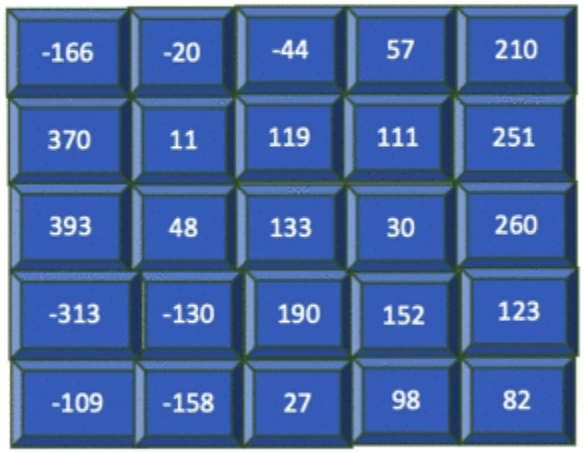
-4	-2
-1	0

Average Pooling

0	1
2	1

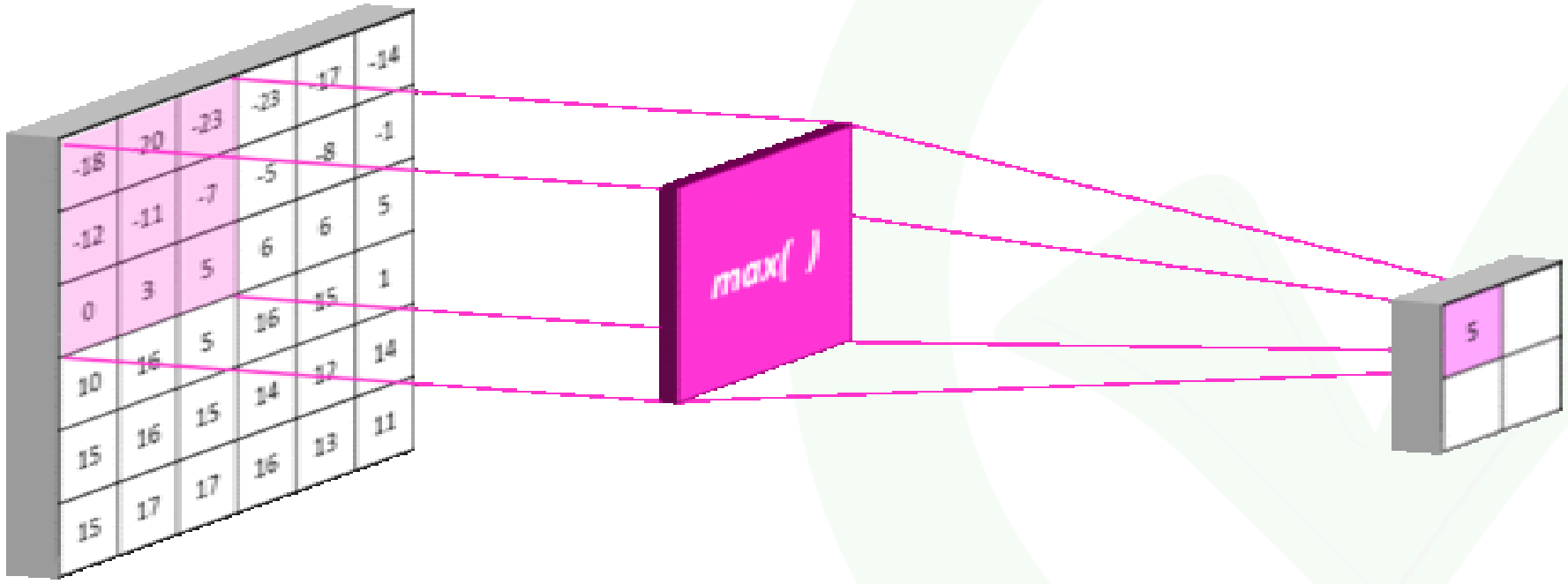
Output

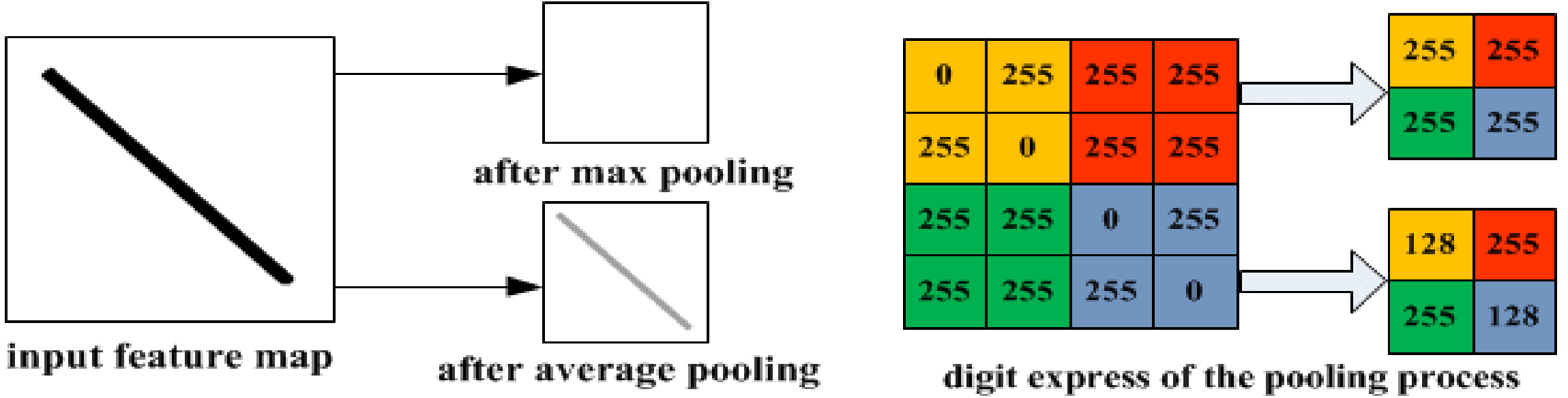
Pooling Layer



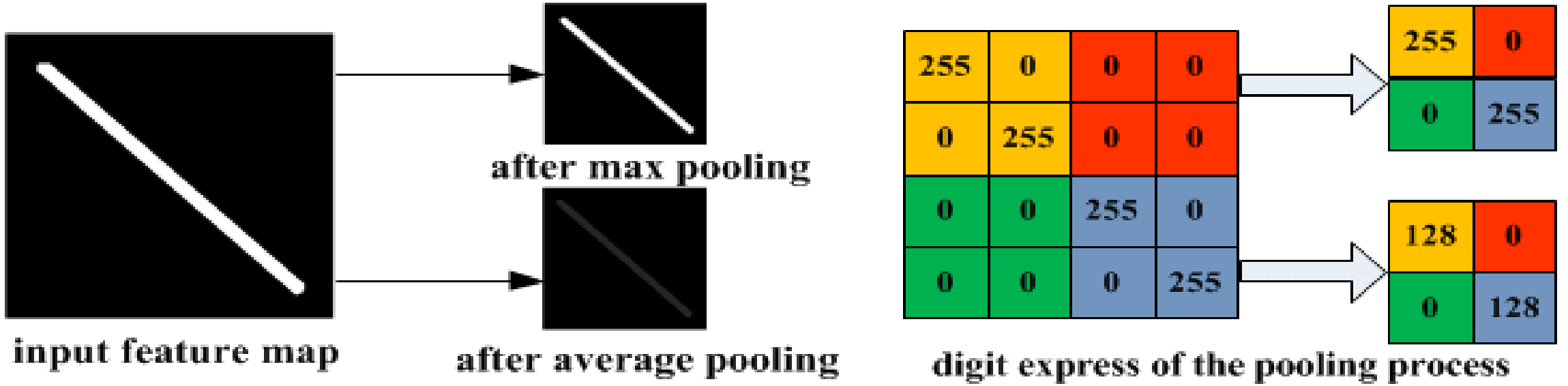
=

Pooling Layer





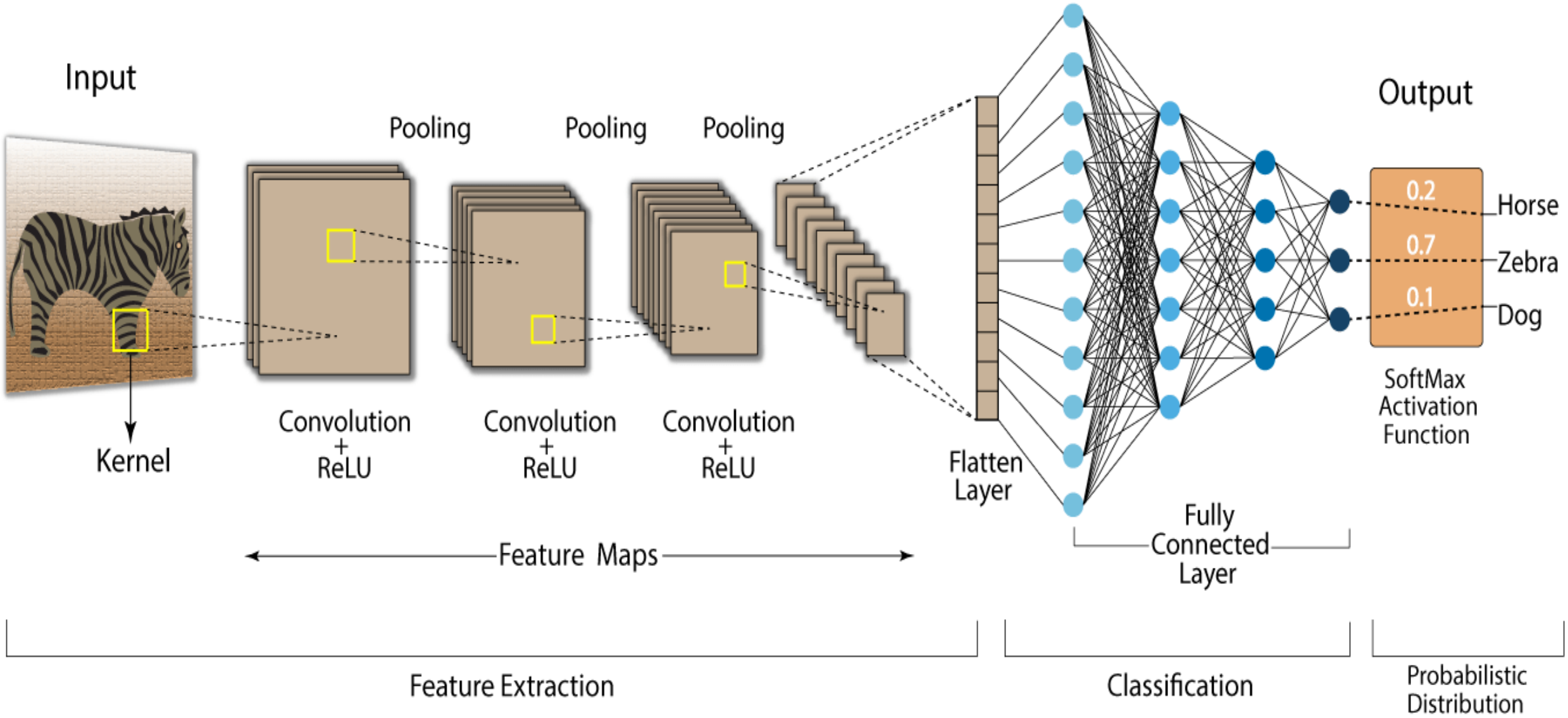
(a) Illustration of max pooling drawback



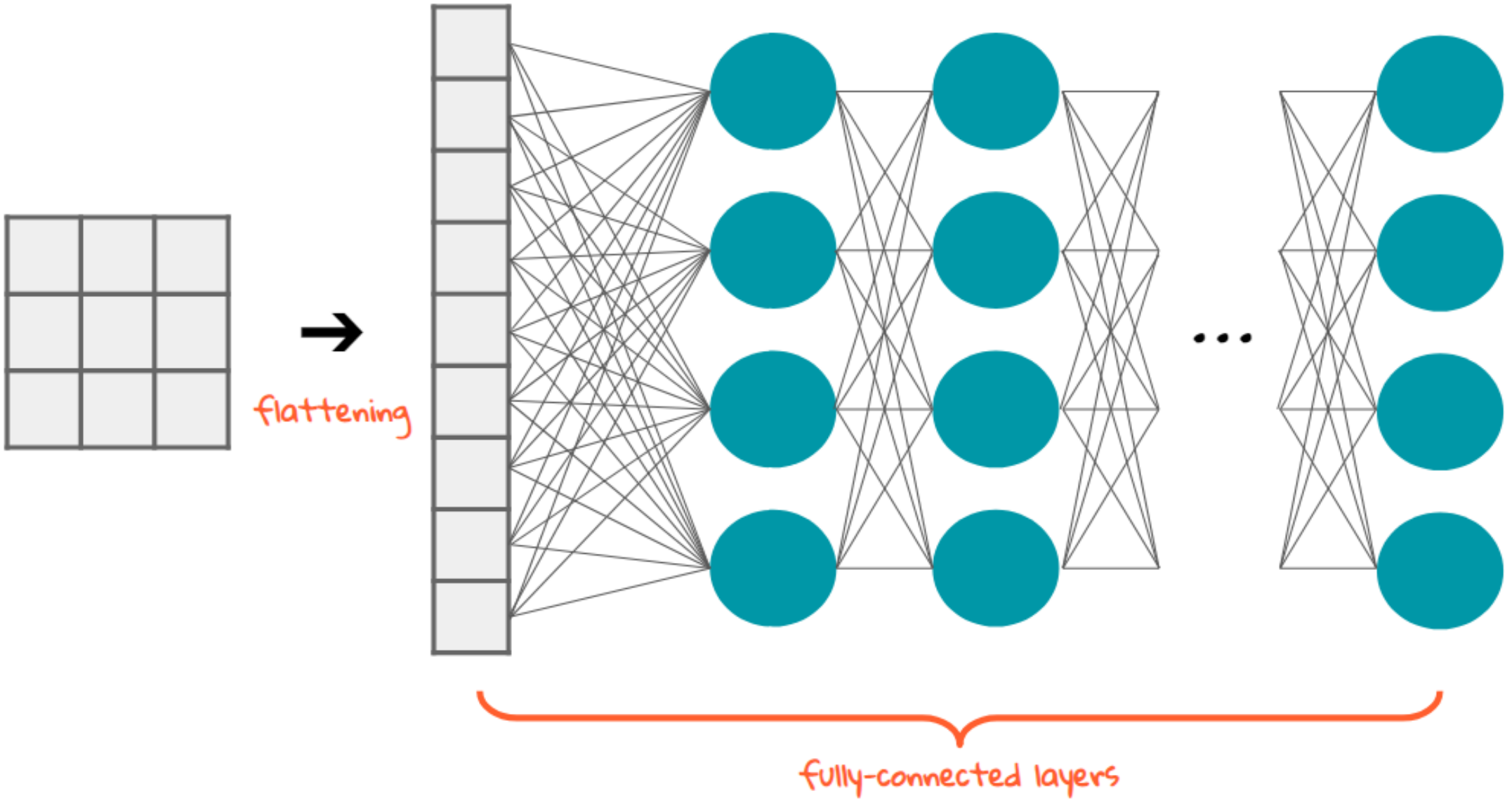
(b) Illustration of average pooling drawback

FLATTENING

Convolution Neural Network (CNN)



Flattening



Flattening

1	1	0
4	2	1
0	2	1

Pooled Feature Map

Flattening

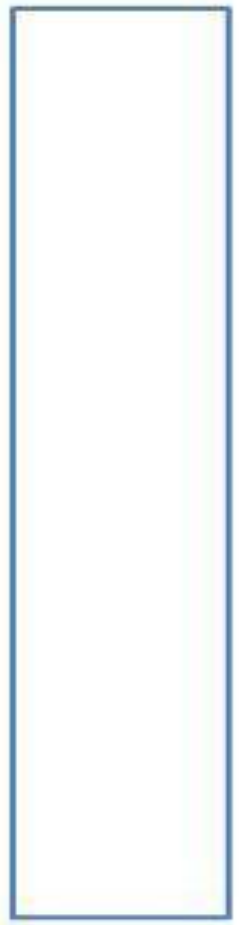
1
1
0
4
2
1
0
2
1

Flattening

11	119	119	251
393	119	119	260
393	190	190	260
-109	190	190	152

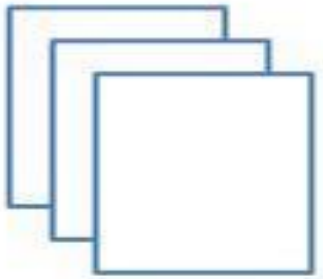
=

Flattening



Input Volume
32x32x1

ReLU Activation Fn.
Volume-28x28x3



Convolution
layer Stride 1

Output Volume
14x14x3

Max Pool
layer Stride 2

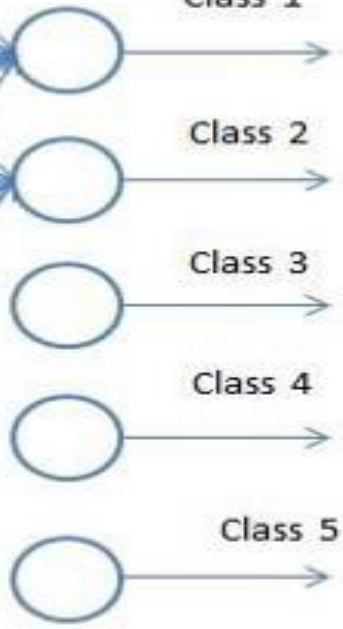
Output Volume
588x1

Flatten layer

Output Volume
20x1

Fully connected
Layer ReLU Activation
Fn.

Output Nodes
5x1

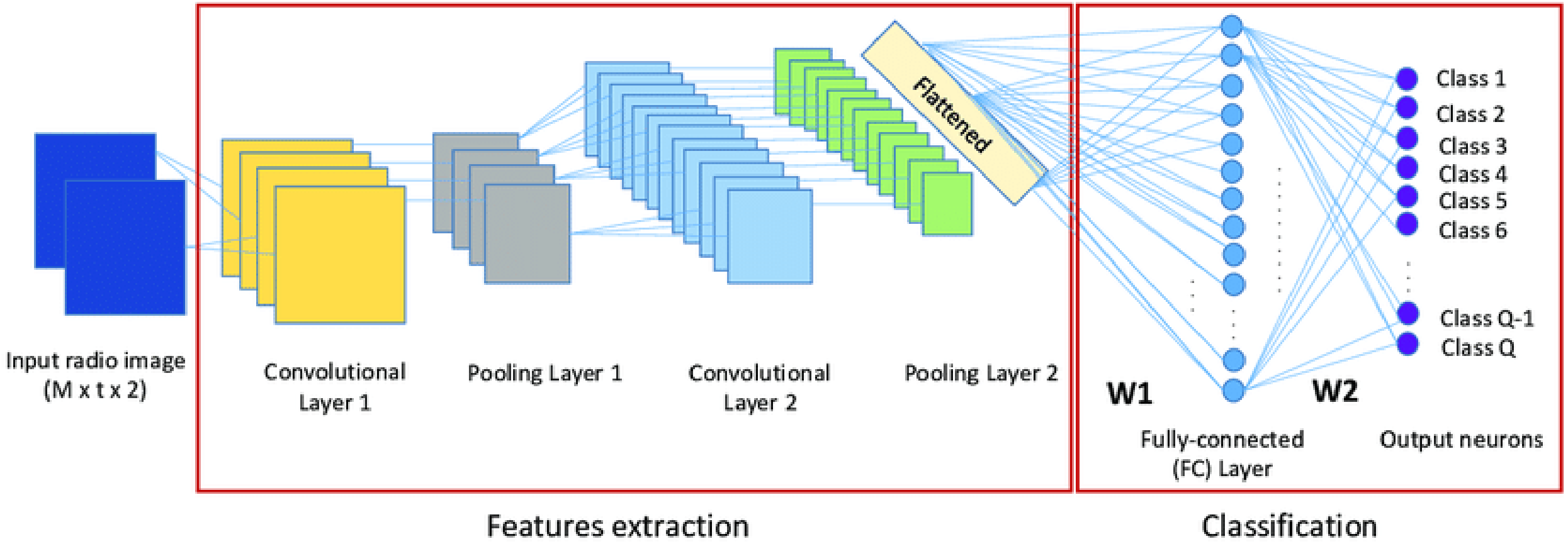


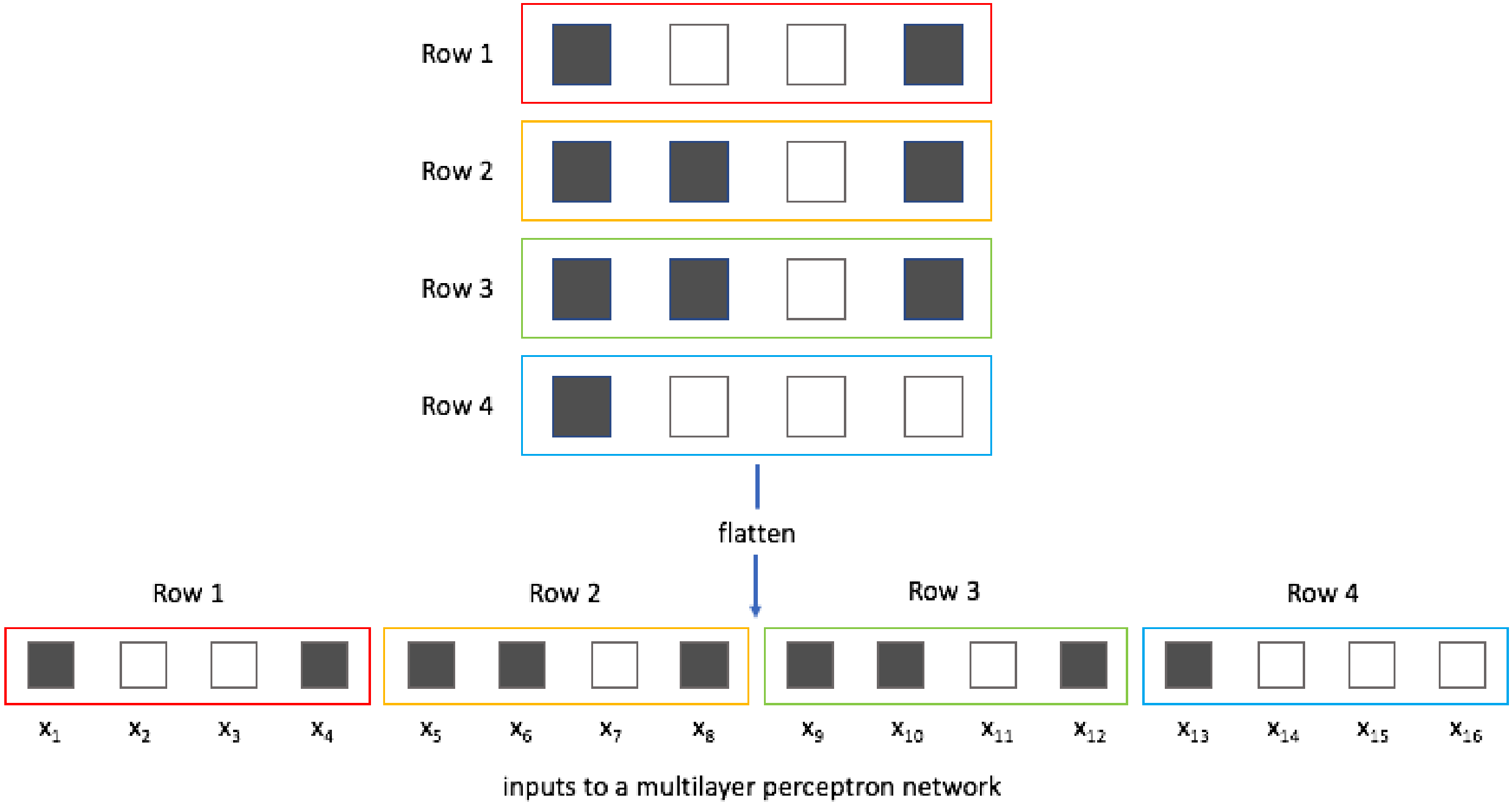
Soft-max Layer

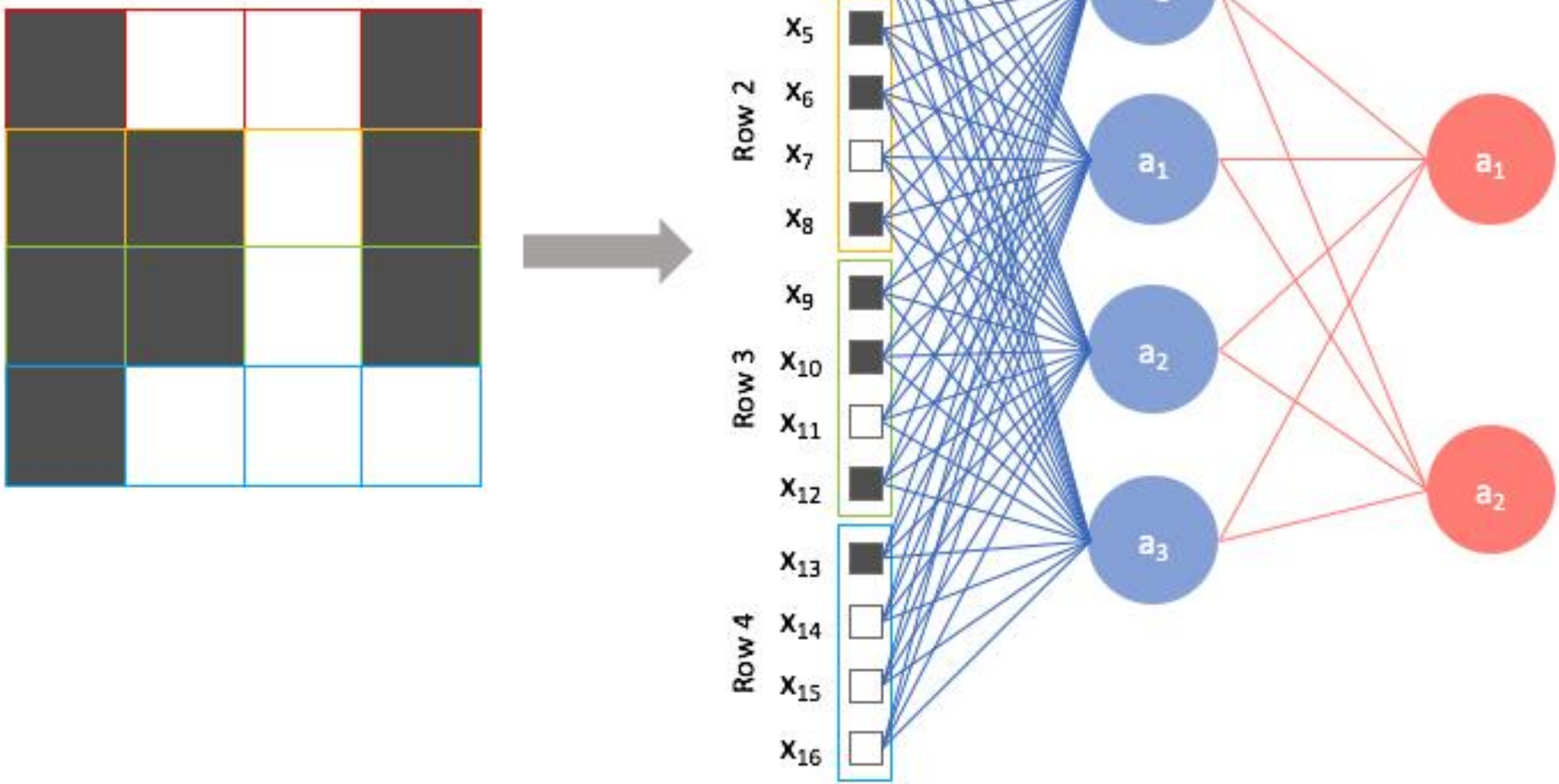
Soft-Max
Activation Fn

FULLY CONNECTED LAYER

Fully-Connected Layer




Row 1
Row 2
Row 3
Row 4
flatten
Row 1
Row 2
Row 3
Row 4
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}
inputs to a multilayer perceptron network


Input layer
Hidden layer
Output layer
Row 1
Row 2
Row 3
Row 4
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x12
x13
x14
x15
x16
a1
a1
a2
a3
a1
a2

The whole CNN

Convolution

Max Pooling

Convolution

Max Pooling

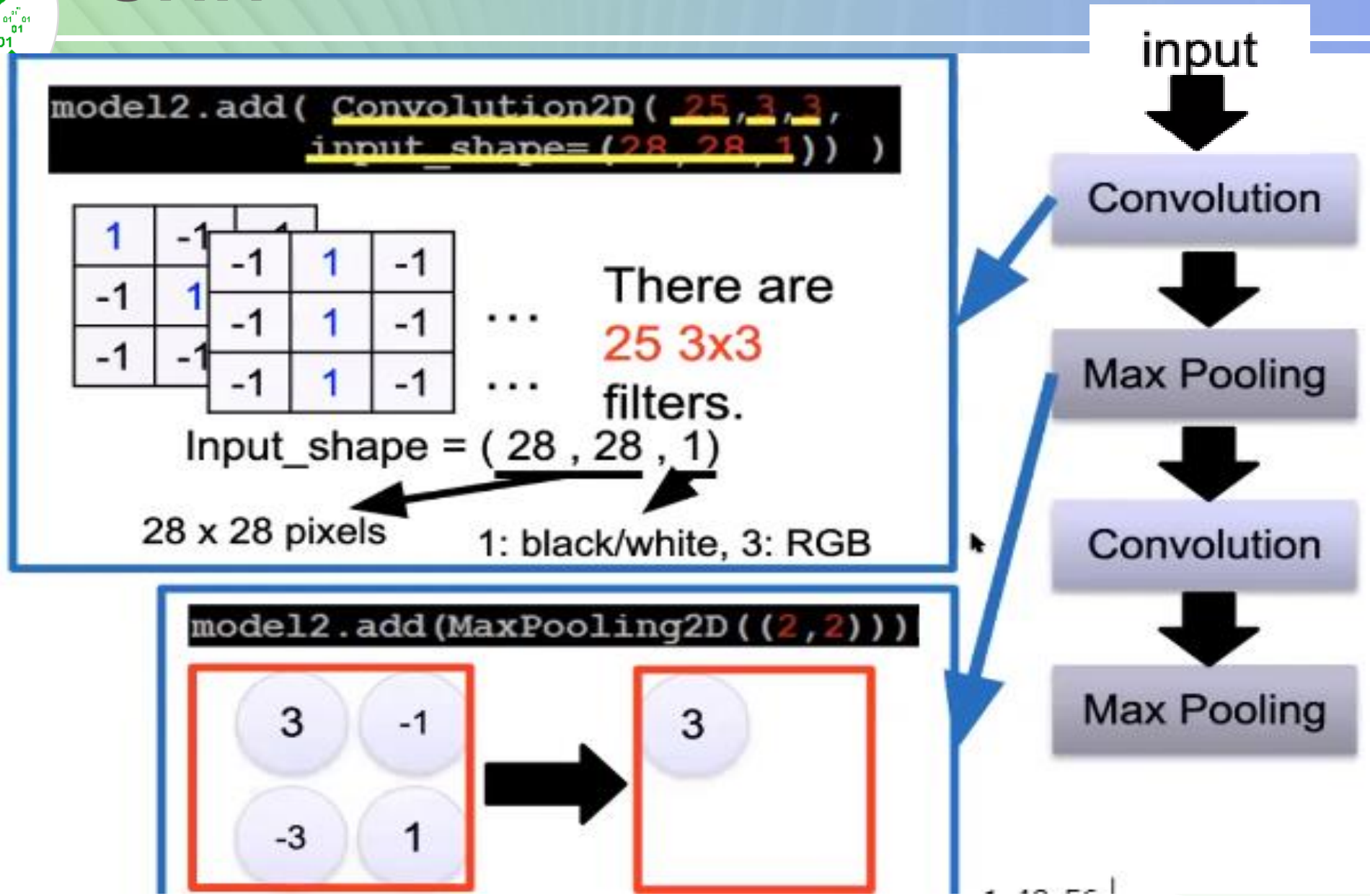
Can repeat many times

Flatten

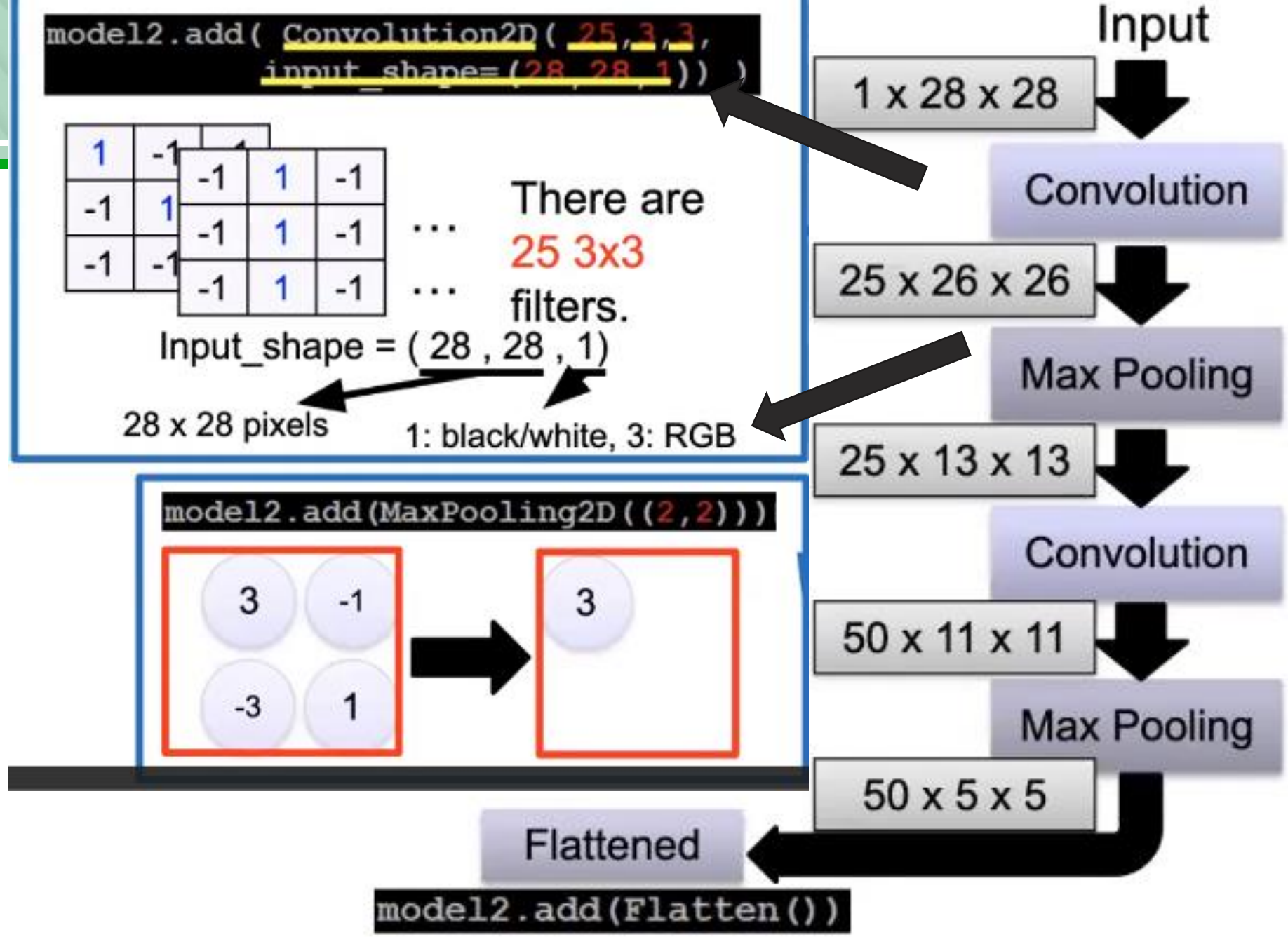
Fully Connected Feedforward network

cat dog ……

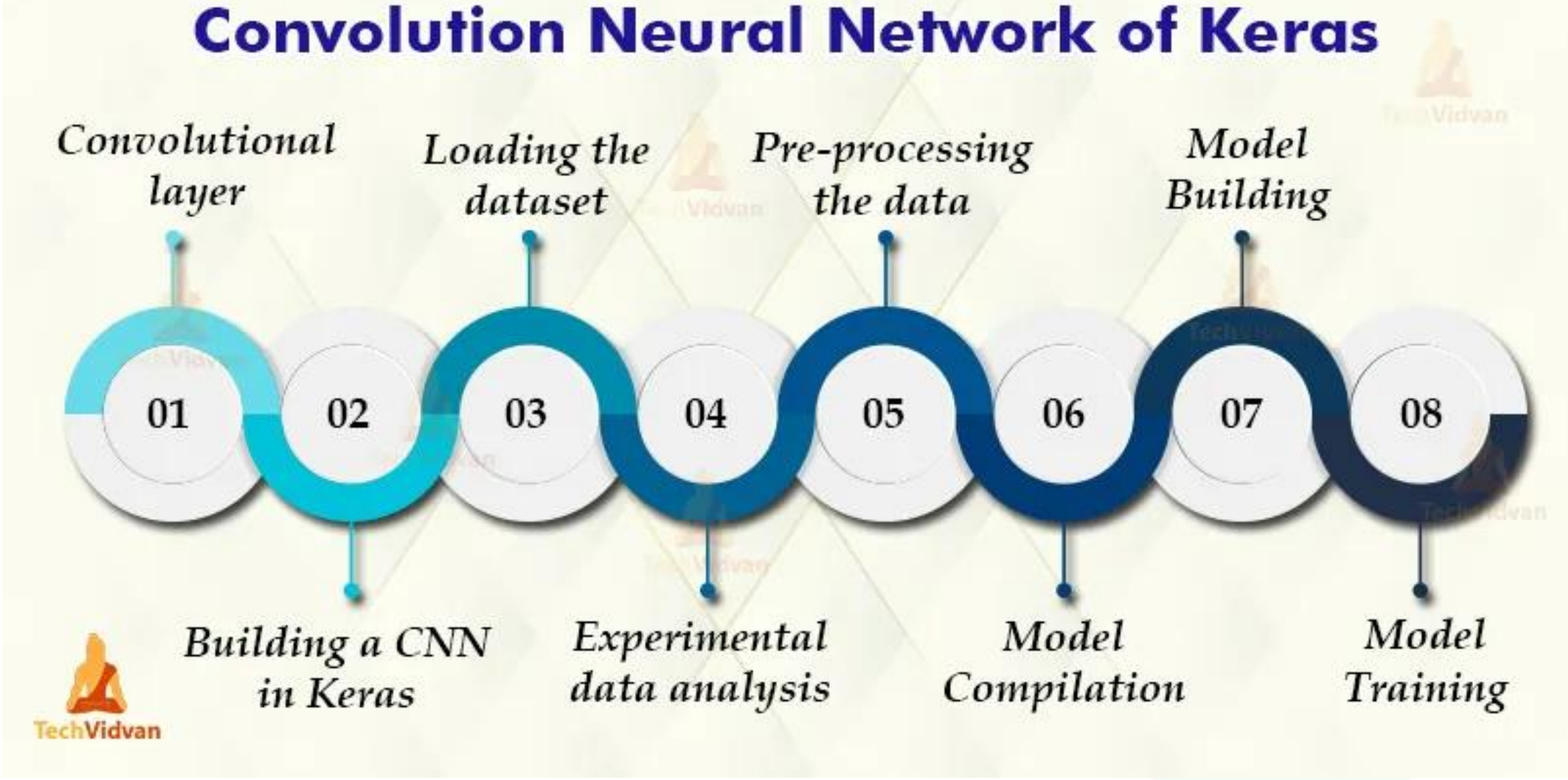
CNN


model2.add(Convolution2D(25,3,3,
input_shape=(28,28,1)))
There are
25 3x3
filters.
Input_shape = (28 , 28 , 1)
28 x 28 pixels
1: black/white, 3: RGB
model2.add(MaxPooling2D((2,2)))
3
-1
-3
1
3
input
Convolution
Max Pooling
Convolution
Max Pooling

CNN


model2.add(Convolution2D(25,3,3,
input_shape=(28,28,1)))
There are
25 3x3
filters.
Input_shape = (28 , 28 , 1)
28 x 28 pixels
1: black/white, 3: RGB
model2.add(MaxPooling2D((2,2)))
3
-1
-3
1
3
Input
1 x 28 x 28
Convolution
25 x 26 x 26
Max Pooling
25 x 13 x 13
Convolution
50 x 11 x 11
Max Pooling
50 x 5 x 5
Flattened
model2.add(Flatten())

CNN


Convolution Neural Network of Keras
Convolutional layer
Loading the dataset
Pre-processing the data
Model Building
01
02
03
04
05
06
07
08
Building a CNN in Keras
Experimental data analysis
Model Compilation
Model Training
TechVidvan

NOTEBOOK SAMPLES WITH CNN

CNN

Veri Önişleme

Eğitim setinin ön işlenmesi

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
training_set = train_datagen.flow_from_directory('dataset/training_set',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')
```

```
Found 8000 images belonging to 2 classes.
```

Burada resimler üzerinde kırma işlemi uygulanarak 64x64 boyutuna indirgenir. Burada kullanılan parametreler resim işleme ile ilgili parametreleri ifade eder. Batch size'ın büyük olması, daha doğru gradyan değerinin hesaplanmasını sağlamaktadır.

CNN

Test setinin ön işlenmesi

```
test_datagen = ImageDataGenerator(rescale = 1./255)
test_set = test_datagen.flow_from_directory('dataset/test_set',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
```

CNN

CNN Mimarisinin oluşturulması

Nesne oluşturulması

```
]:  cnn = tf.keras.models.Sequential()
```

Adım 1 - Convolution

```
]:  cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=[64, 64, 3]))
```

Burada kullanılan Conv2D fonksiyonu asyesinde eldeki datalar 64 x 64 pixel boyutunda ele alınacaktır. buradaki 3 parametresi ise resimler renkli olduğu için 3 adet katman oluşturulacaktır.

CNN

Adım 2 - Pooling

```
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

pooling işlemi için MaxPool yöntemi kullanılacaktır. Pool işleminde her seferinde asım olarak yani striders 2 olarak belirlenmişdir.

Adım 3- ikinci convolutional katman

```
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
```

CNN

Adım 4 - Flattening

```
cnn.add(tf.keras.layers.Flatten())
```

Step 5 - Full Connection

```
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
```

Yapay sinir ağında gizli katmanlar ve neronlar oluşturulur. Aktivasyon fonksiyonu için en sık kullanılan yöntemlerden relu fonksiyonu kullanılmıştır. Kaç katman olacakğı ise units metodu ile 128 olarak belirlenmiştir. Buradaki parametre resimler 64x64 boyutunda olduğu için 128 olarak belirlenmiştir.

Adım 6 - Çıkış Katmanı

```
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

Çıkış katmanı ise sınıflama işlemi yapılacağı için sigmoid olarak belirlenmiştir. Eğer çıktı 1 olarak çıkarsa köpek 0 olarak çıkarsa kedi çıktısı verecektir.

CNN

Derleme işlemi

```
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Son adımda çıkış katmanından optimizer için adam fonksiyonu kullanmıştır. metric olarak accuracy seçilmiştir.

Eğitim işlemi

```
cnn.fit(x = training_set, validation_data = test_set, epochs = 50)
```

```
Epoch 3/50
250/250 [==============================] - 44s 176ms/step - loss: 0.5871 - accuracy: 0.6883 - val_loss: 0.5813 - val_accu
racy: 0.6940
Epoch 4/50
250/250 [==============================] - 43s 173ms/step - loss: 0.5530 - accuracy: 0.7074 - val_loss: 0.5068 - val_accu
racy: 0.7520
Epoch 5/50
250/250 [==============================] - 46s 183ms/step - loss: 0.5105 - accuracy: 0.7457 - val_loss: 0.4880 - val_accu
racy: 0.7700
Epoch 6/50
250/250 [==============================] - 47s 189ms/step - loss: 0.4890 - accuracy: 0.7599 - val_loss: 0.5459 - val_accu
racy: 0.7355
Epoch 7/50
250/250 [==============================] - 43s 173ms/step - loss: 0.4754 - accuracy: 0.7702 - val_loss: 0.4821 - val_accu
```

CNN

Tahmin oluşturmak

```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img('dataset/single_prediction/cat_or_dog_1.jpg', target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
  prediction = 'dog'
else:
  prediction = 'cat'
```

```
print(prediction)
```

```
dog
```

CNN EXPLAINER Learn Convolutional Neural Network (CNN) in your browser!