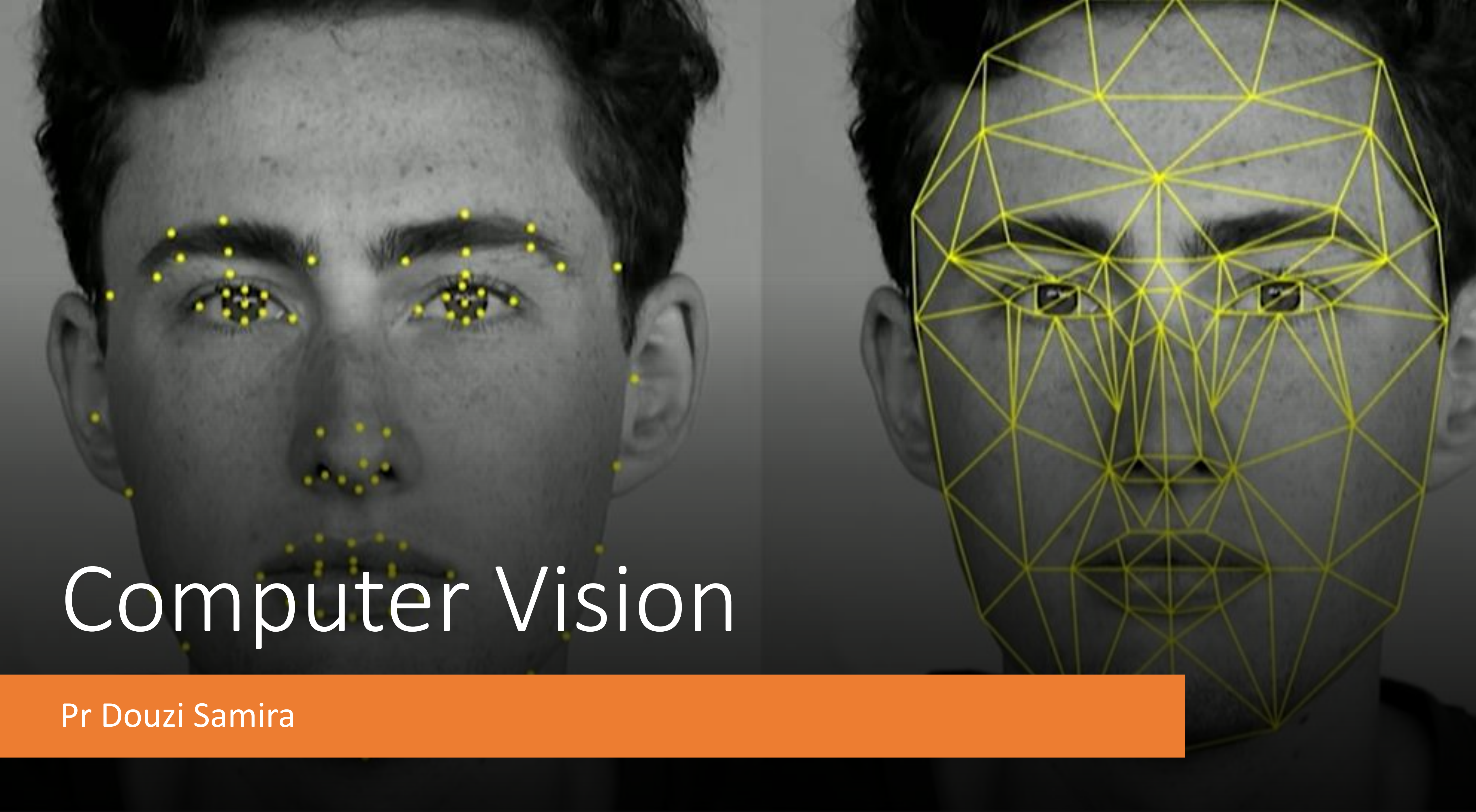
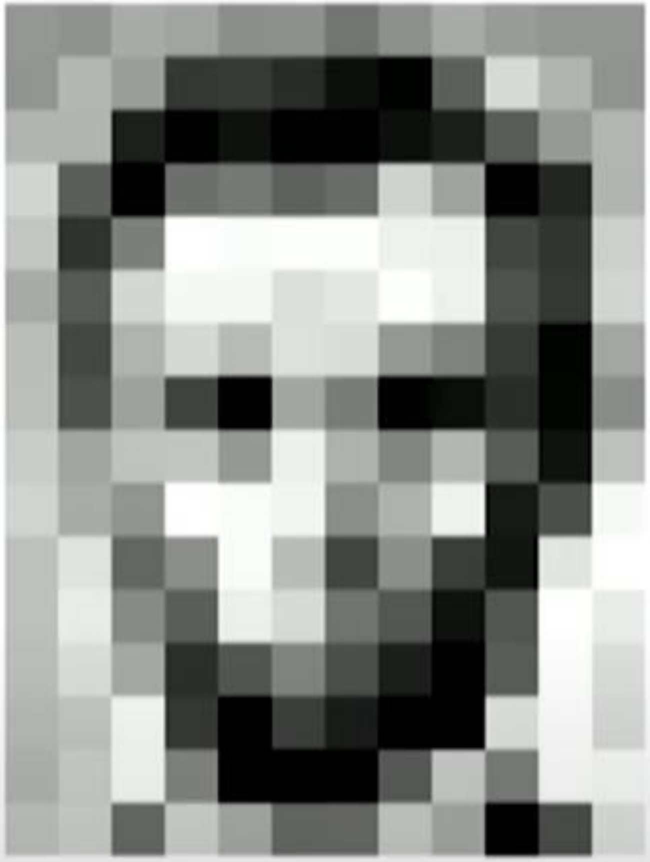


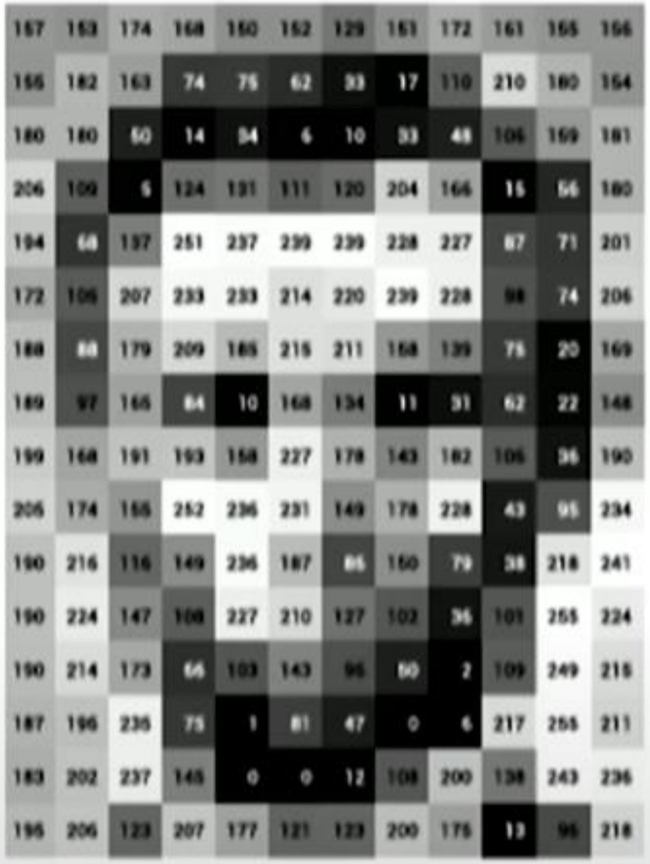
Computer Vision

Pr Douzi Samira

Images are Numbers

Images are Numbers





What the computer sees

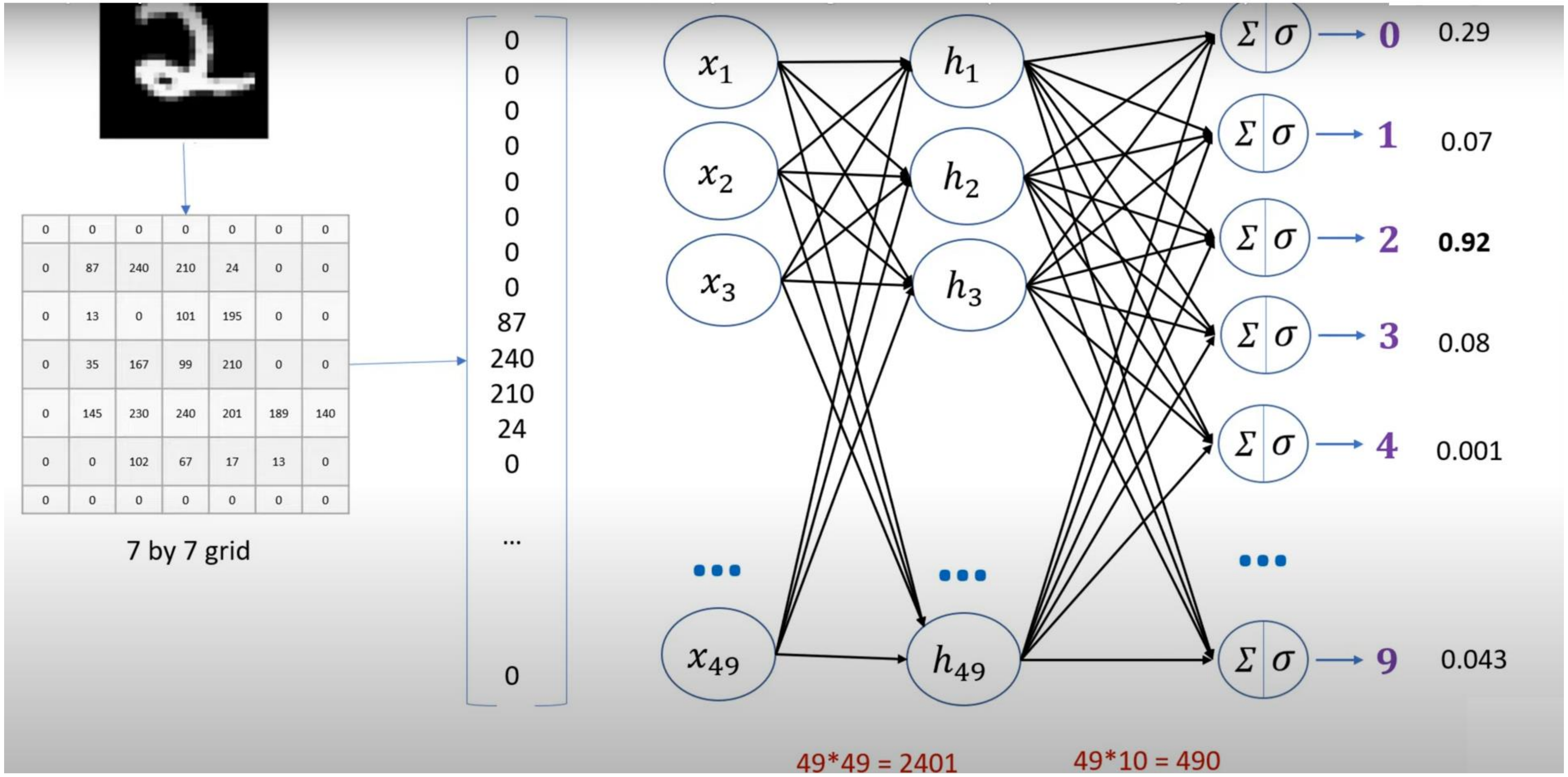
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

An image is just a matrix of numbers [0,255]!
i.e., 1080x1080x3 for an RGB image

Tasks in Computer Vision

- **Regression**: output variable takes continuous value
- **Classification**: output variable takes class label. Can produce probability of belonging to a particular class

Can we use ANN for Images?



Can we use ANN for Images?

Image size = 1920 x 1080 X 3

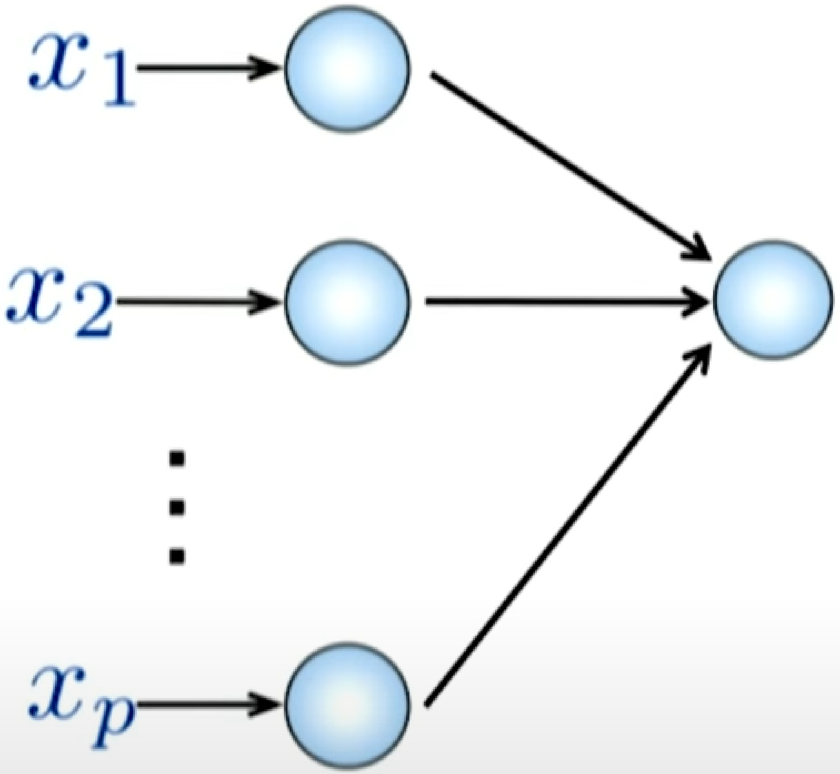
First layer neurons = 1920 x 1080 X 3 ~ 6 million

Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

Fully Connected Neural Network

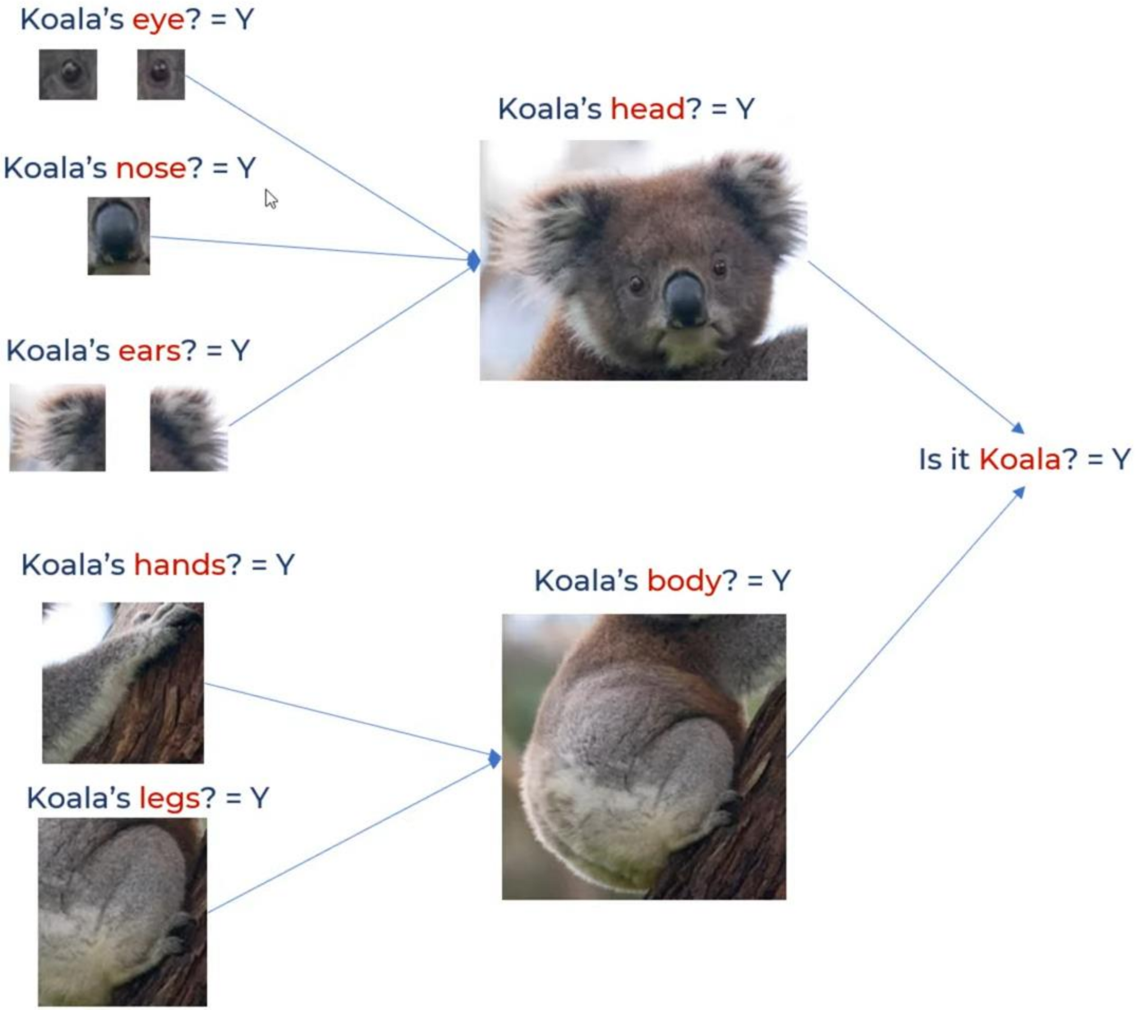
Input:
- 2D image
- Vector of pixel values



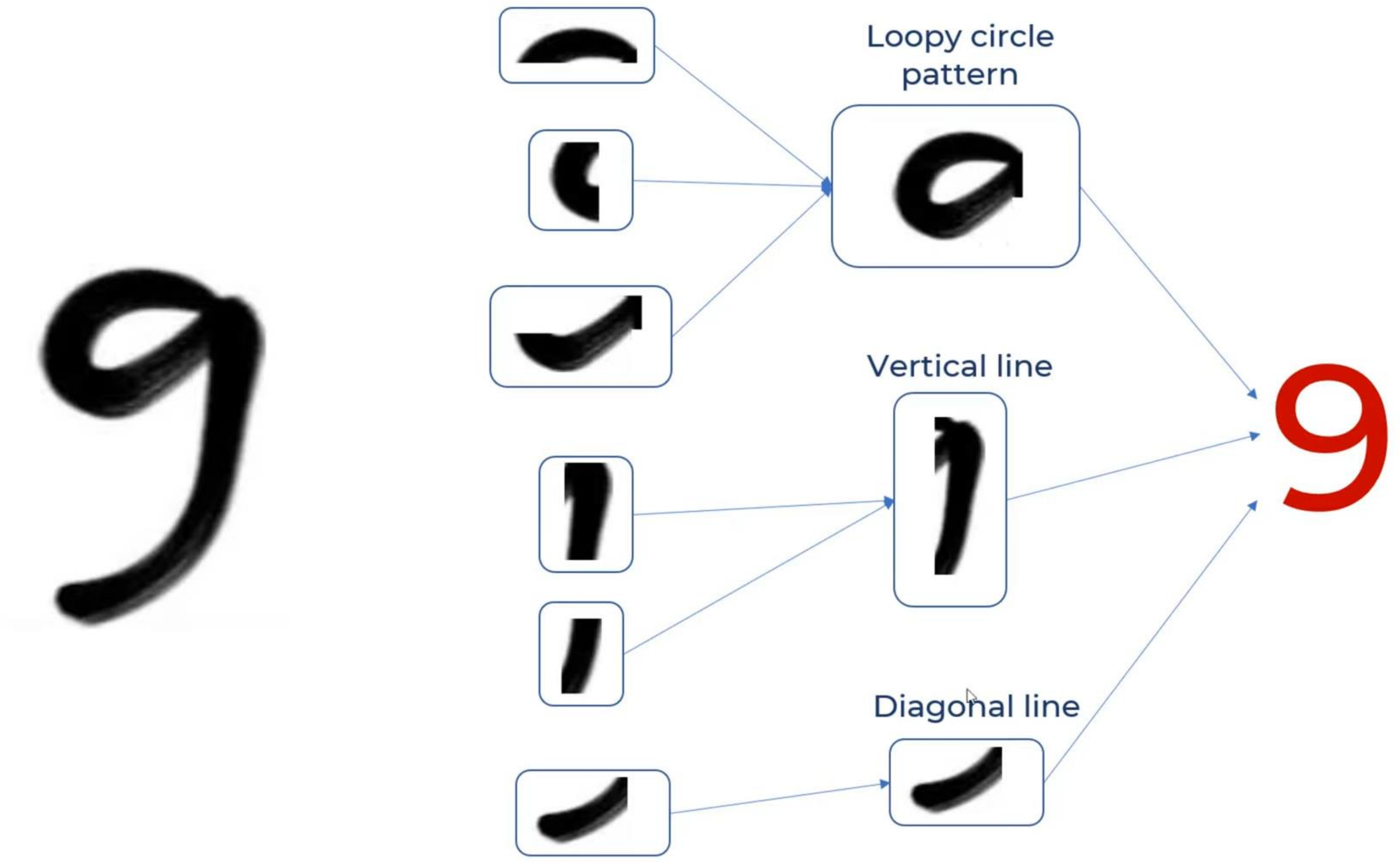
Fully Connected:
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- And many, many parameters!

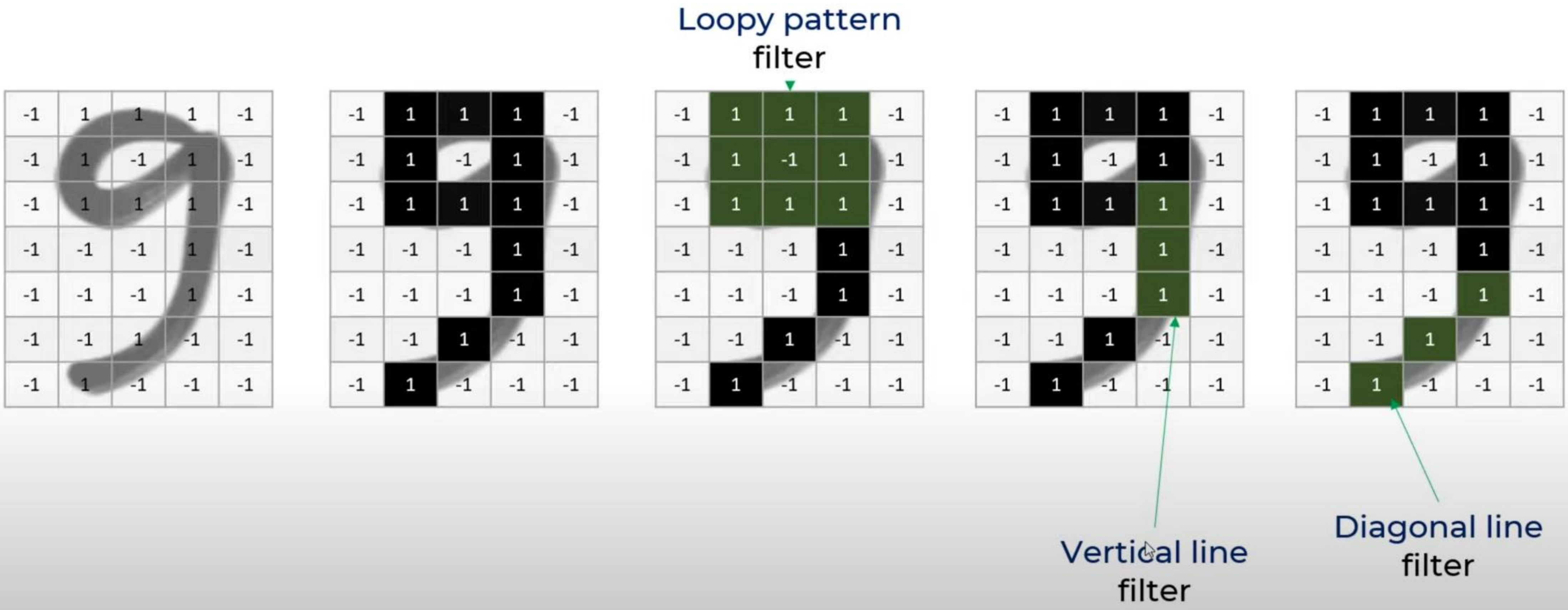
How can we use **spatial structure** in the input to inform the architecture of the network?

Features Detection



Features Detection




Loopy pattern
filter
Vertical line
filter
Diagonal line
filter

Convolution

$$-1+1+1-1-1-1-1+1+1 = -1 \rightarrow -1/9 = -0.11$$

-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

-0.11		

Convolution

-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

-0.11	**1**	

Convolution

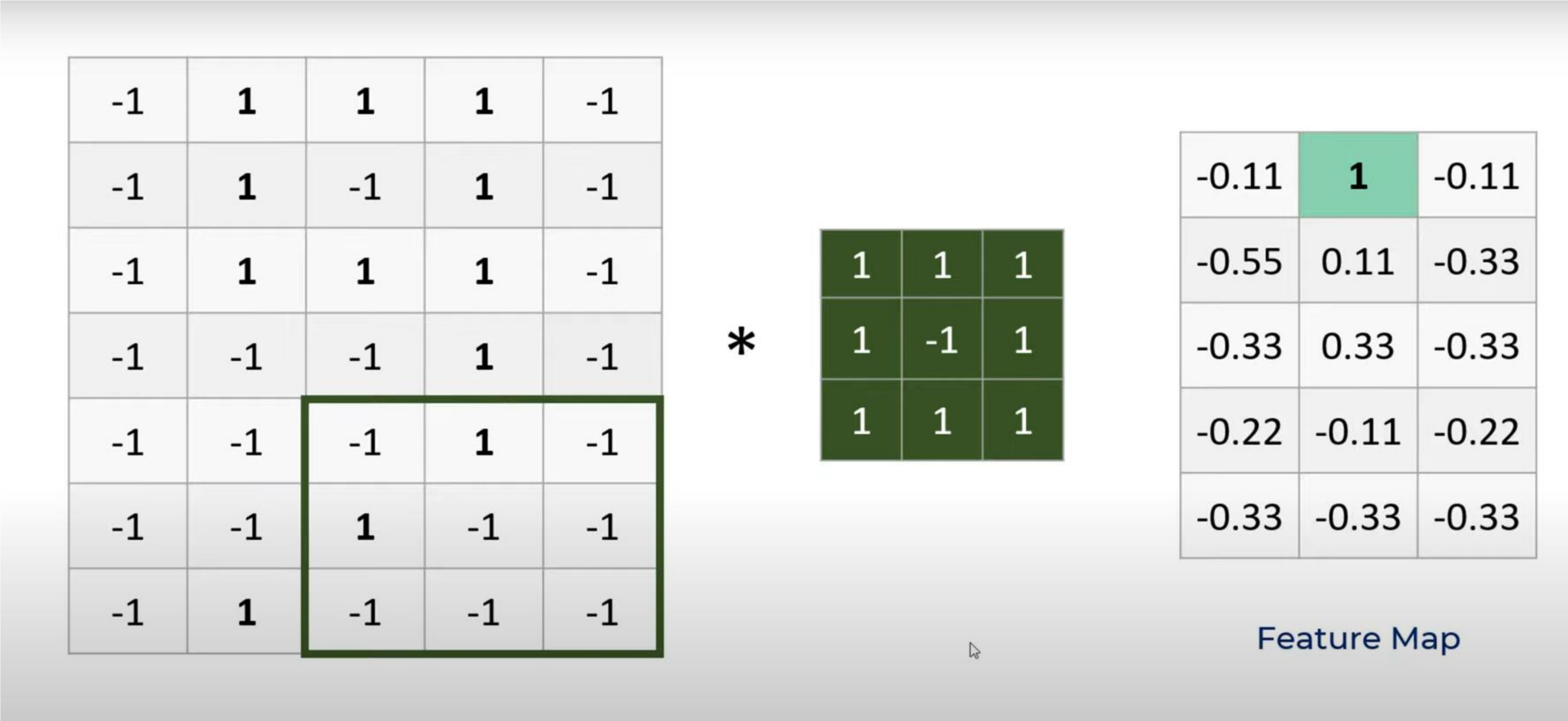
-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

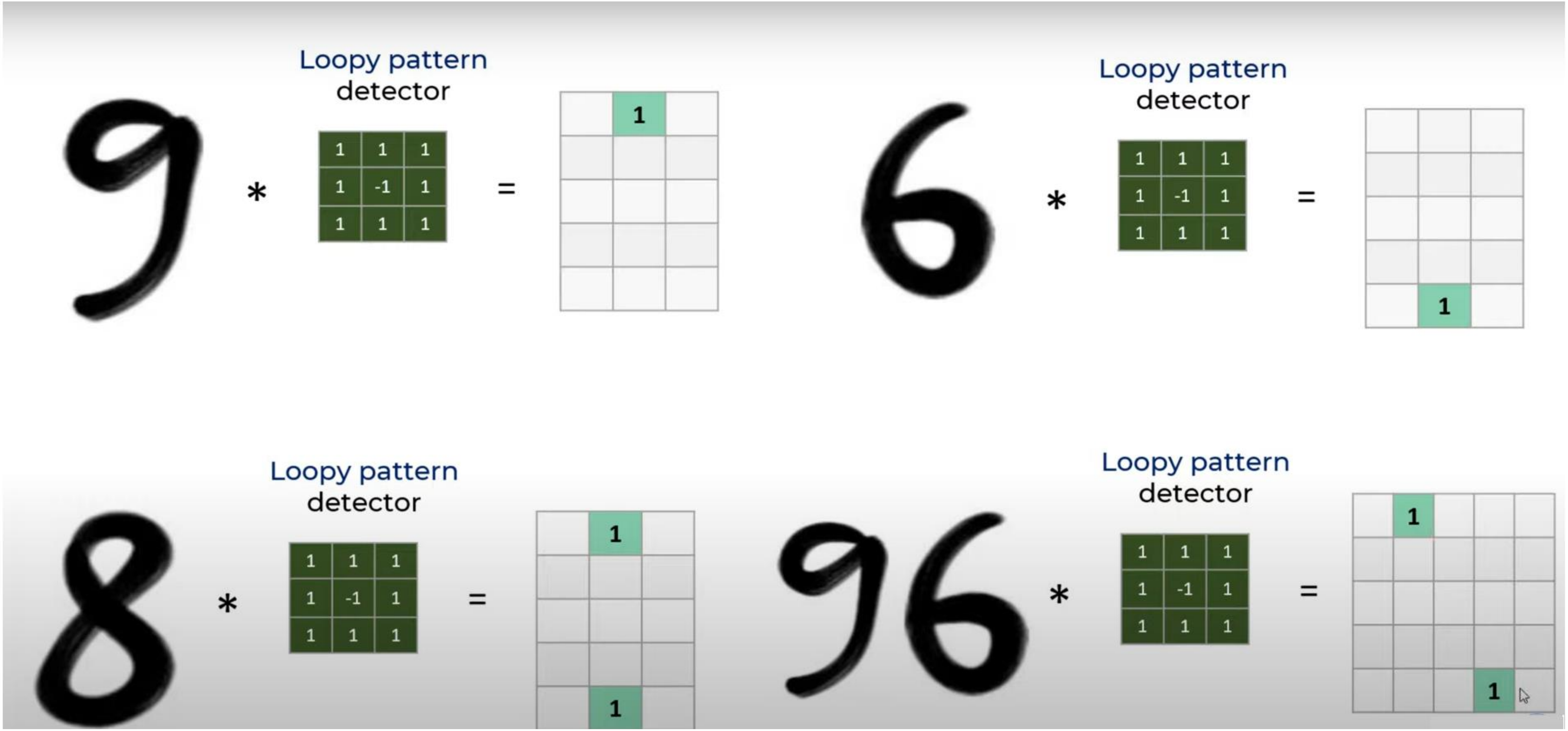
1	1	1
1	-1	1
1	1	1

-0.11	1	-0.11
-0.55	0.11	

Convolution



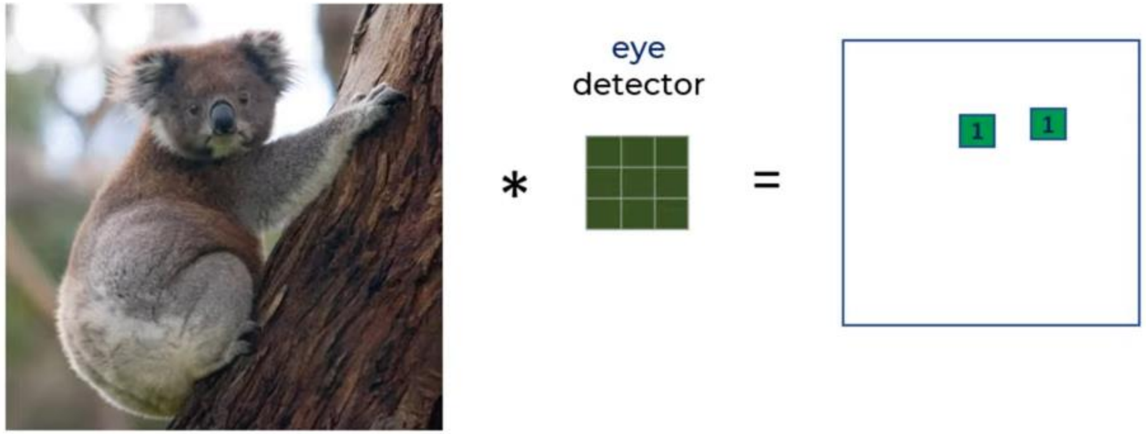
Feature Map

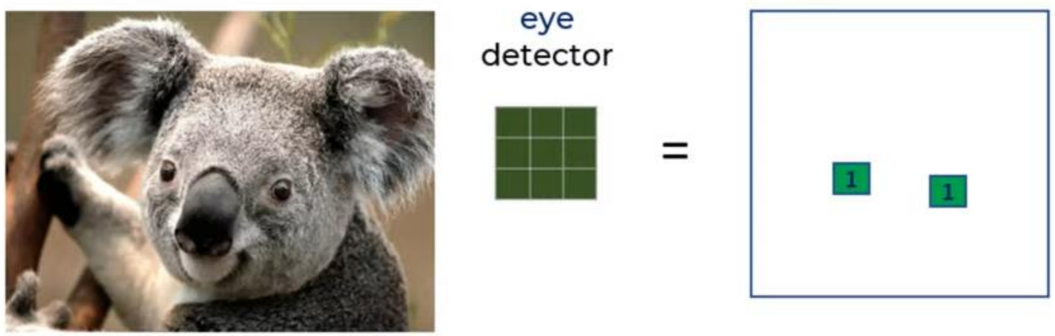


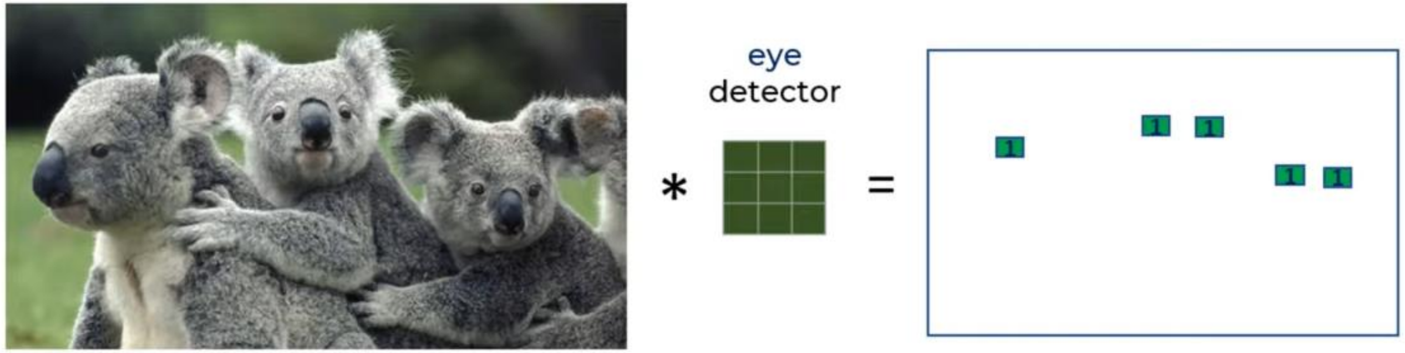
Loopy pattern
detector
*
1 1 1
1 -1 1
1 1 1
=
1
Loopy pattern
detector
*
1 1 1
1 -1 1
1 1 1
=
1
Loopy pattern
detector
*
1 1 1
1 -1 1
1 1 1
=
1
1
Loopy pattern
detector
*
1 1 1
1 -1 1
1 1 1
=
1
1

Feature Map

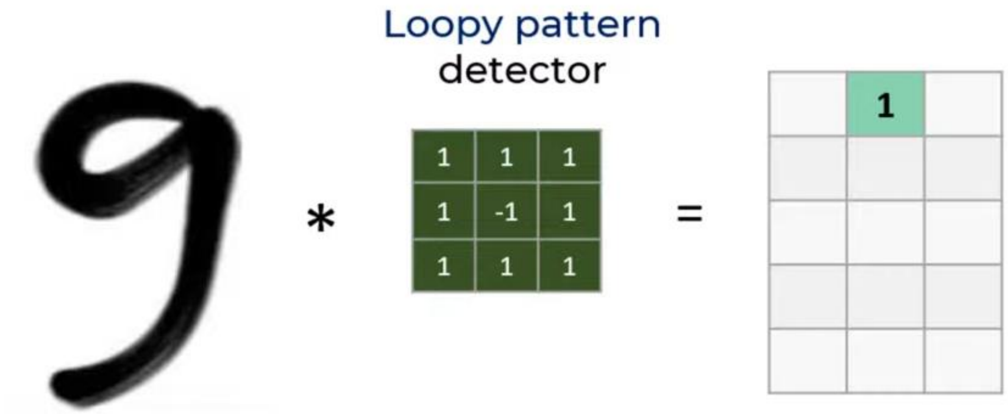
Location invariant: It can detect eyes in any location of the image



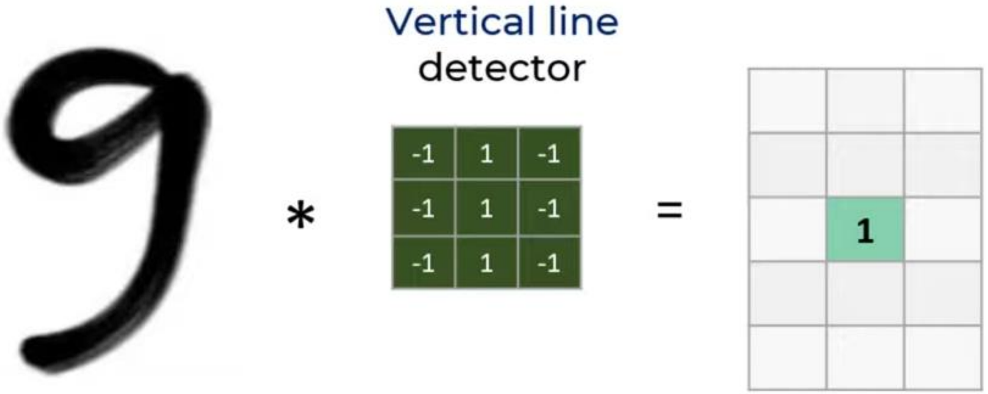




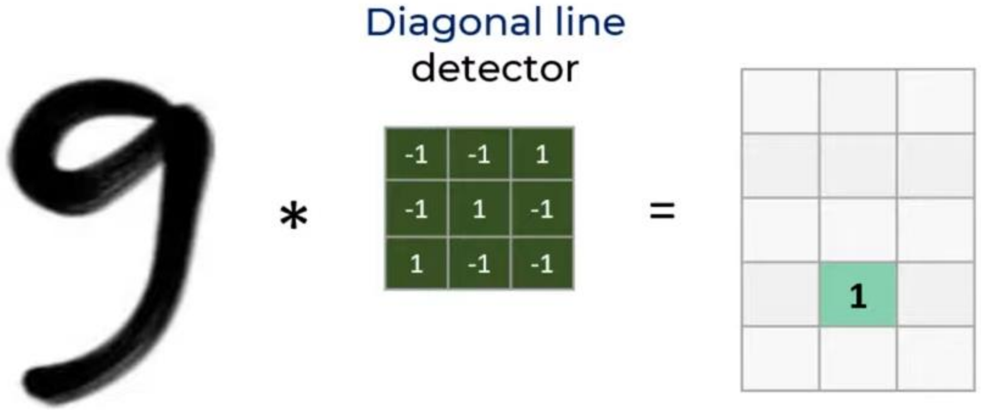
Feature Map



Loopy pattern
detector
1 1 1
1 -1 1
1 1 1
*
=
1

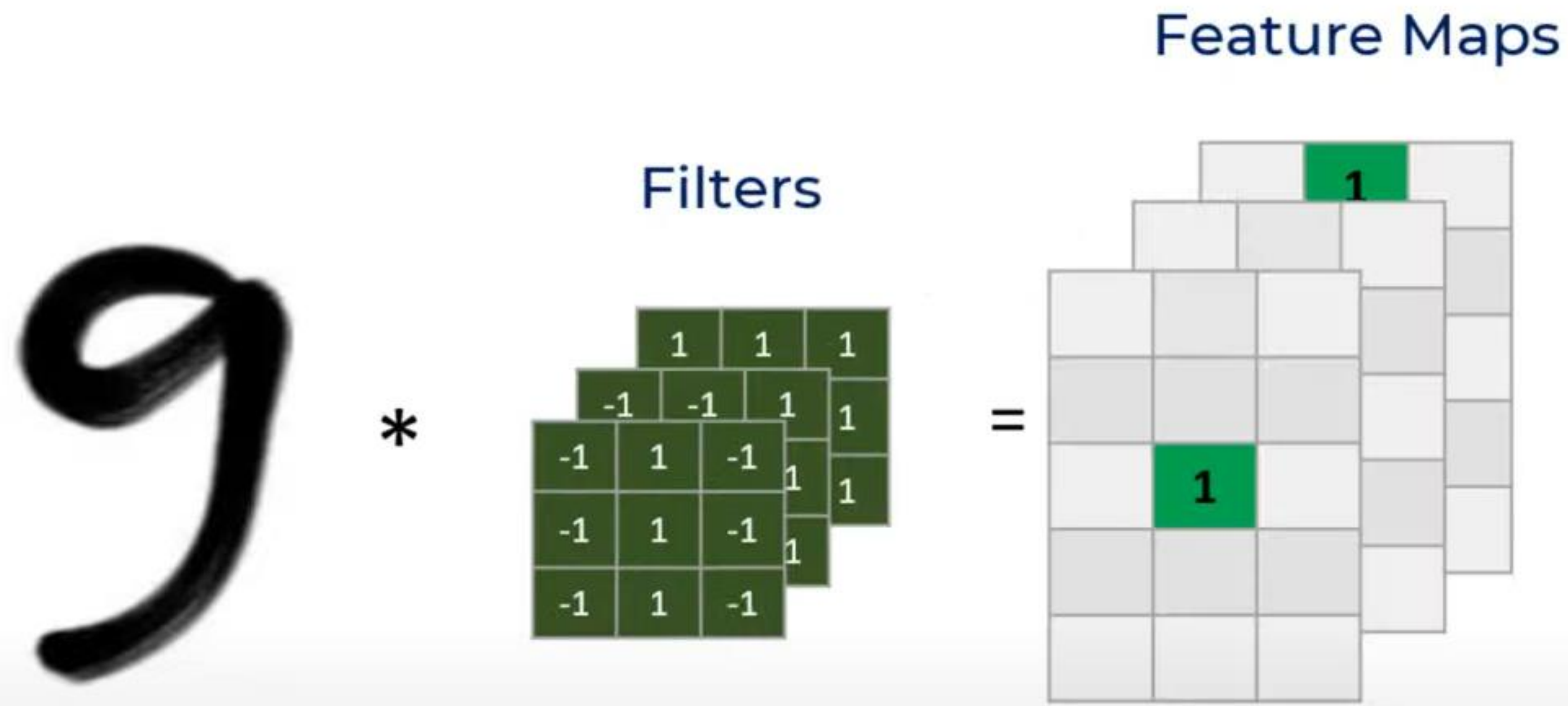


Vertical line
detector
-1 1 -1
-1 1 -1
-1 1 -1
*
=
1

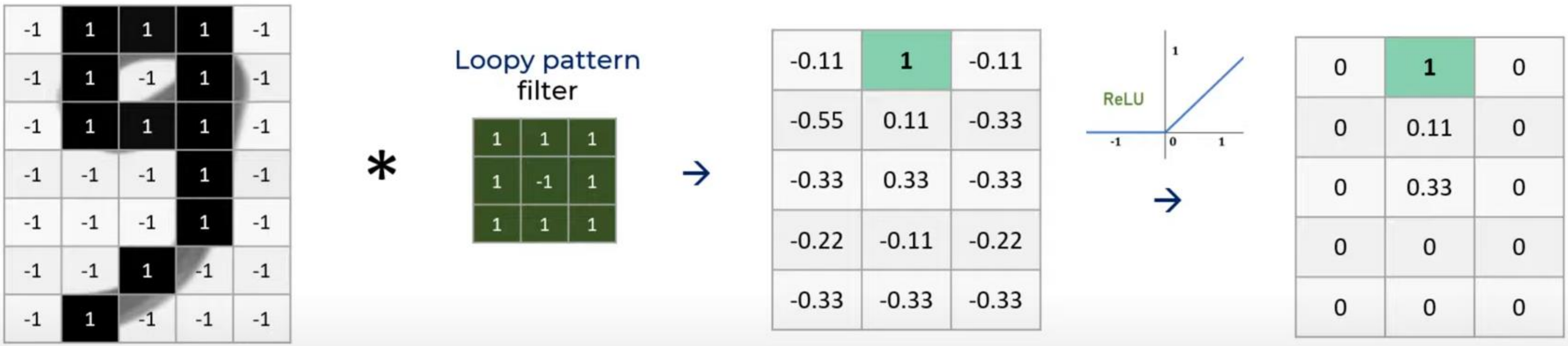


Diagonal line
detector
-1 -1 1
-1 1 -1
1 -1 -1
*
=
1

Feature Map

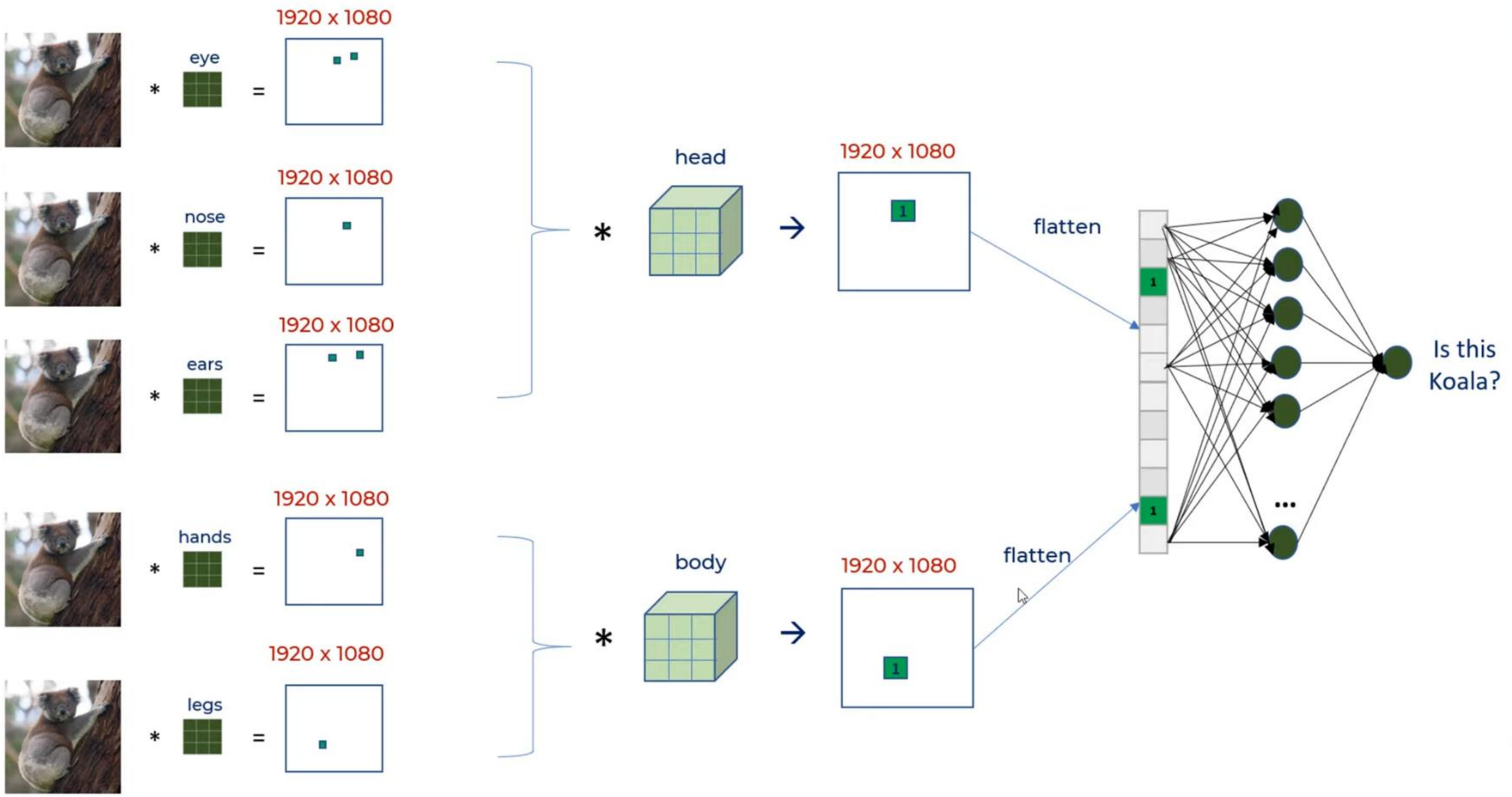


Feature Map

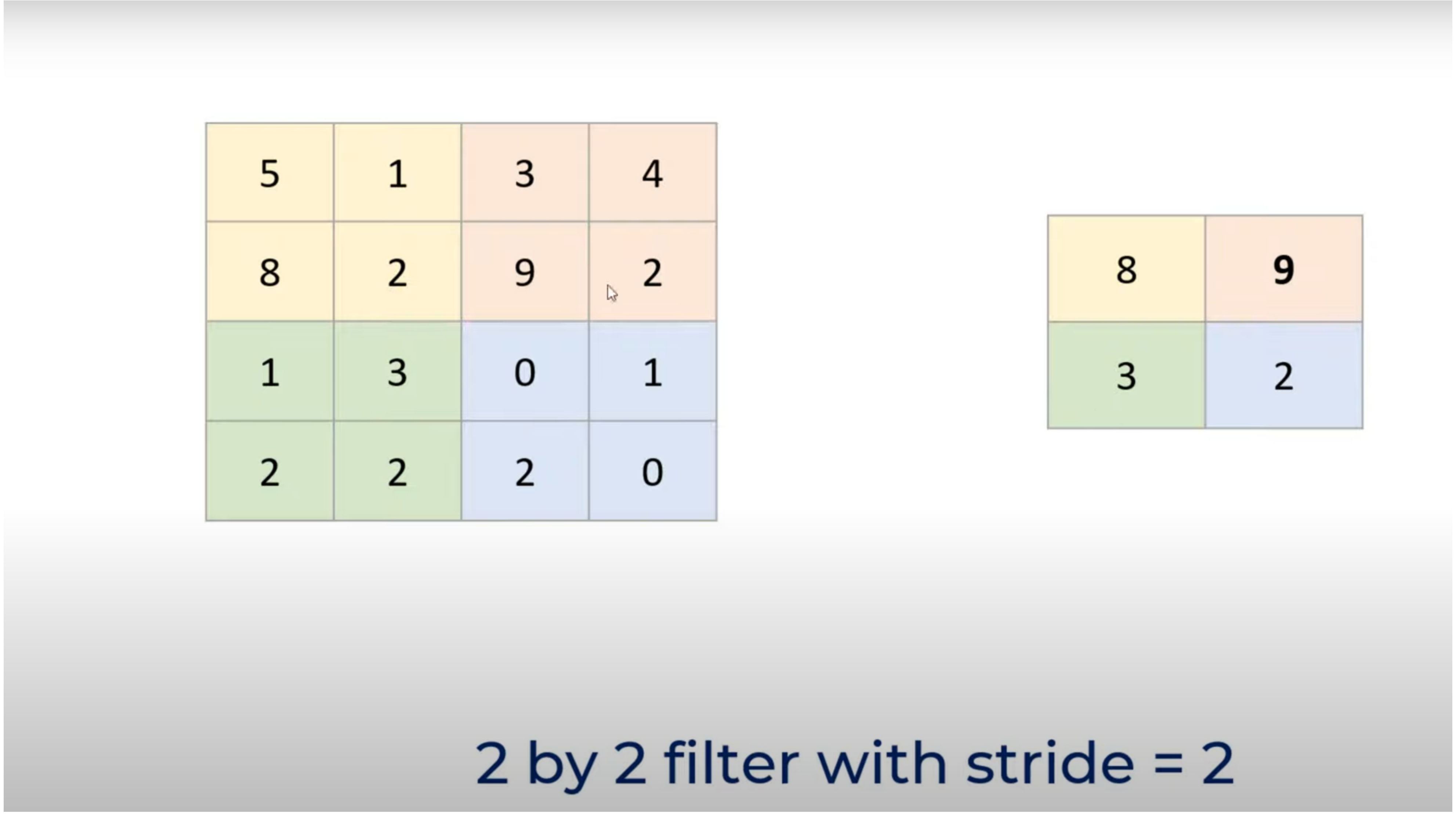


-1 1 1 1 -1
-1 1 -1 1 -1
-1 1 1 1 -1
-1 -1 -1 1 -1
-1 -1 -1 1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
*
Loopy pattern
filter
1 1 1
1 -1 1
1 1 1
-0.11 1 -0.11
-0.55 0.11 -0.33
-0.33 0.33 -0.33
-0.22 -0.11 -0.22
-0.33 -0.33 -0.33
ReLU
0 1 0
0 0.11 0
0 0.33 0
0 0 0
0 0 0

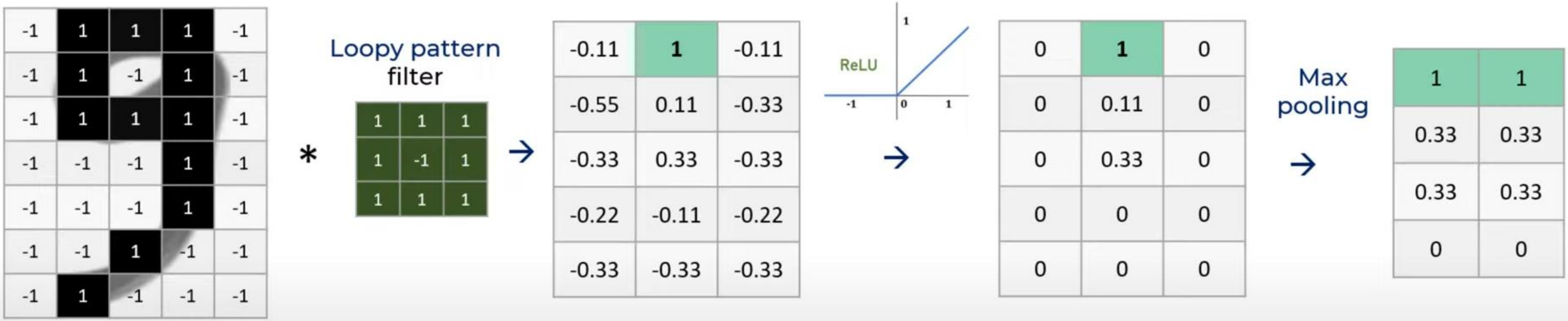
Feature Map and Classification



Reduction by Pooling



Reduction by Max Pooling

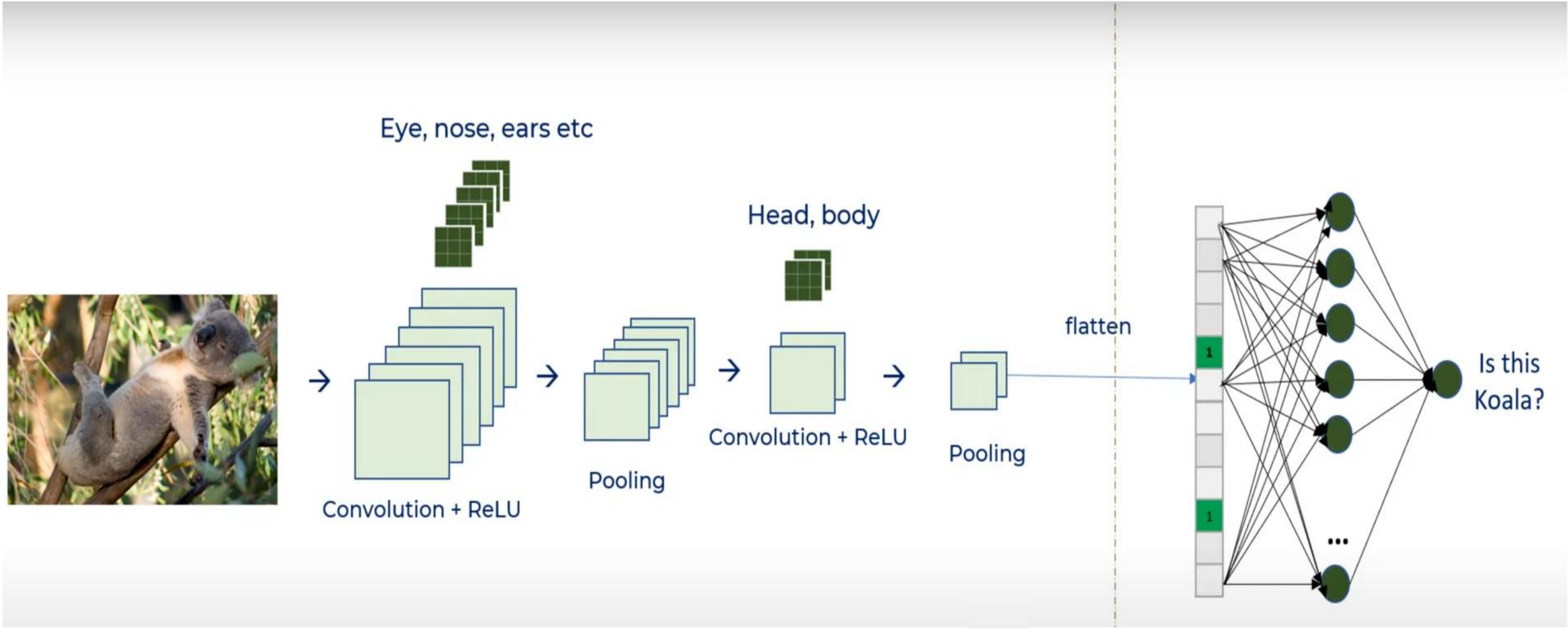


Reduction by Average Pooling

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

4	4.5
2	0.75

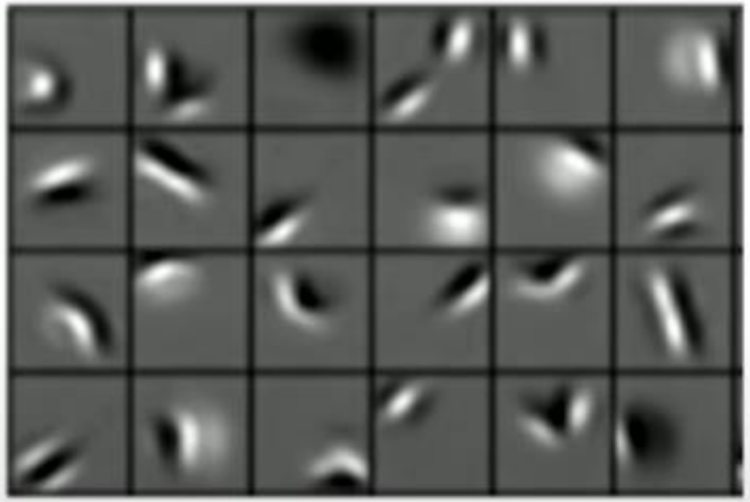
Convolutional layer



Learning Feature Representations

Can we learn a **hierarchy of features** directly from the data instead of hand engineering?

Low level features



Edges, dark spots

Mid level features

Eyes, ears, nose

High level features

Facial structure

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

ReLU

- Introduces nonlinearity
- Speeds up training, faster to compute

Pooling

- Reduces dimensions and computation
- Reduces overfitting
- Makes the model tolerant towards small distortion and variations

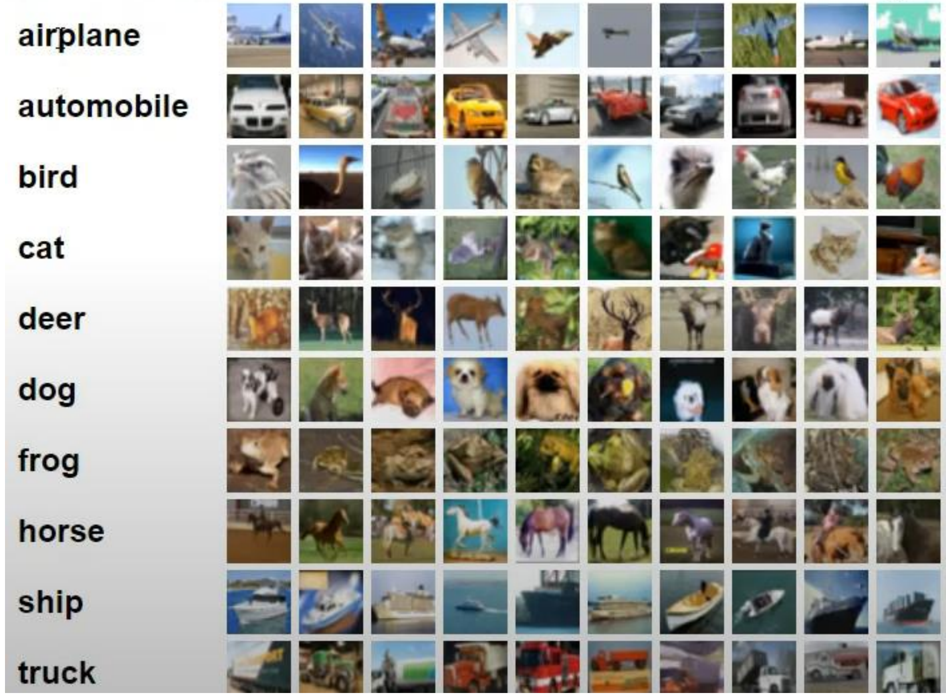
CNN Example

The CIFAR-10 dataset

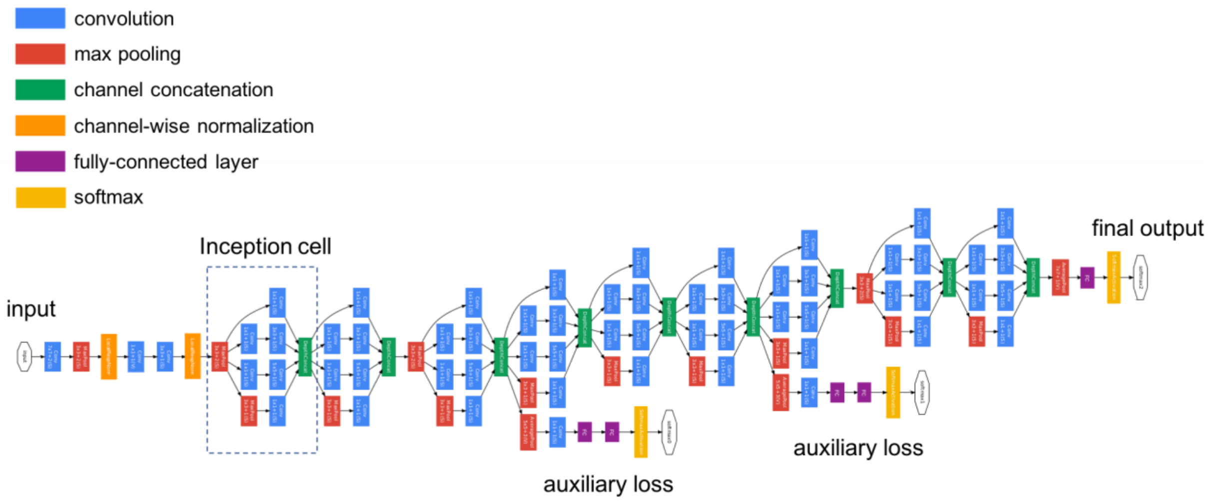
The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

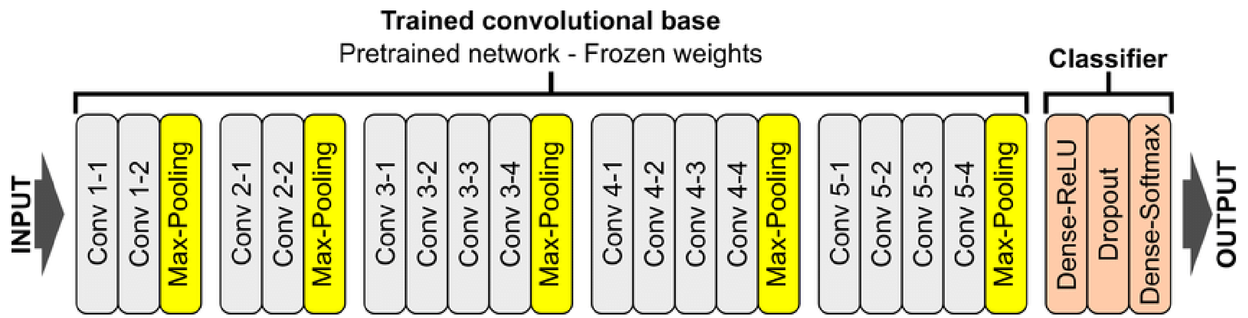
The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



CNN pre-trained





CNN pre-trained