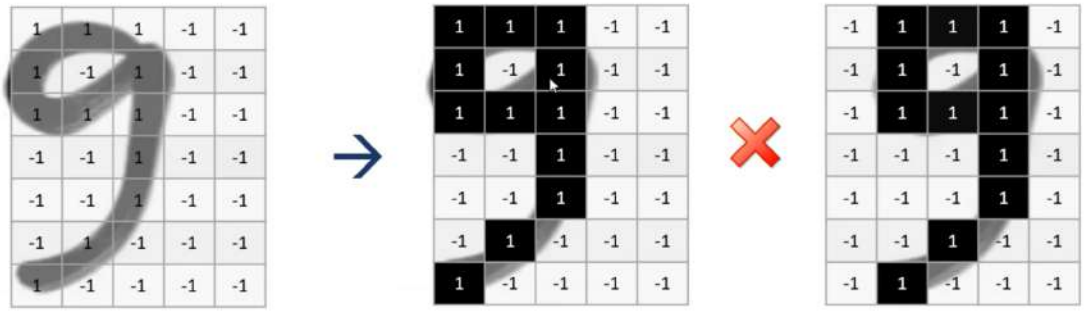
Image credit: https://mathematicaforprediction.files.wordpress.com/2013/08/digitimageswithzenbrush-testset.jpg

-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

Location shifted



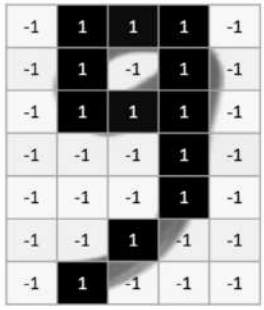
Variation 1

-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

✗



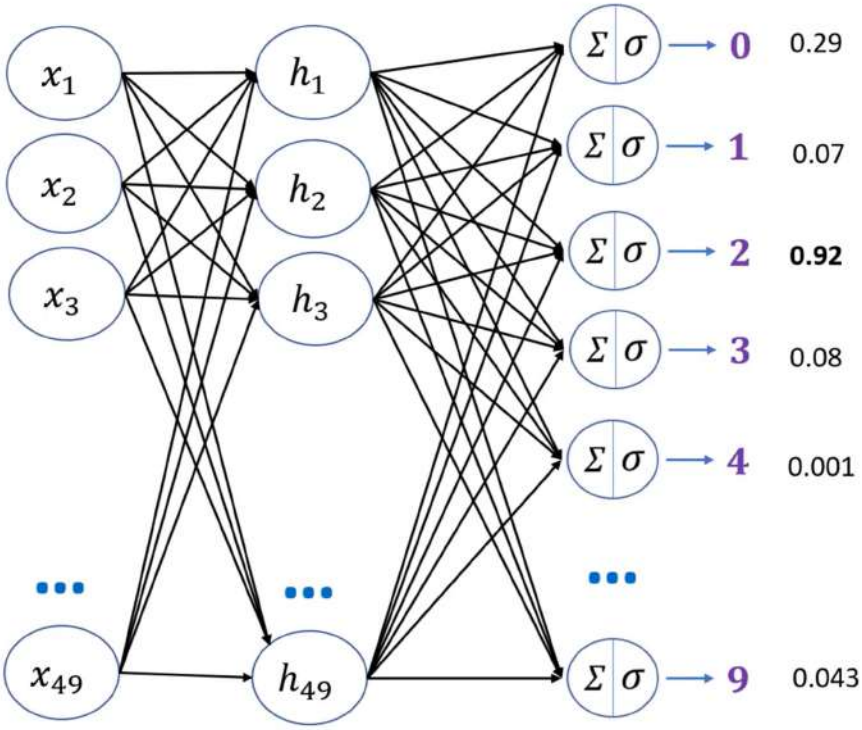
-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

To handle variety in digits we can use simple artificial neural network (ANN)

0	0	0	0	0	0	0
0	87	240	210	24	0	0
0	13	0	101	195	0	0
0	35	167	99	210	0	0
0	145	230	240	201	189	140
0	0	102	67	17	13	0
0	0	0	0	0	0	0

7 by 7 grid

0
0
0
0
0
0
0
0
87
240
210
24
0
...
0



49*49 = 2401 49*10 = 490

Image size = 1920 x 1080 X 3

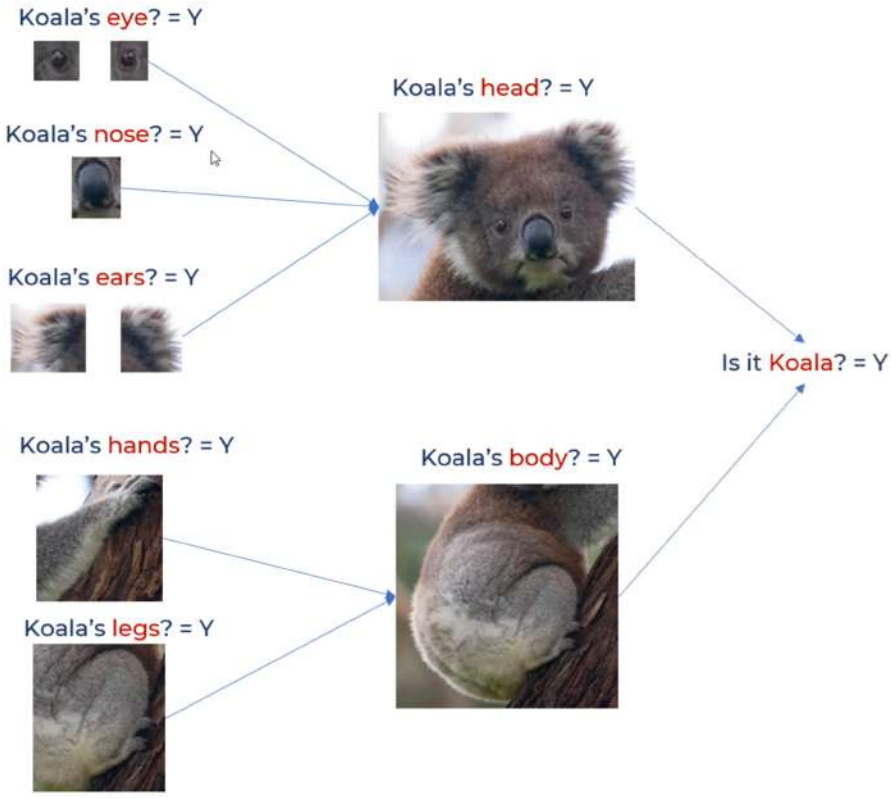
First layer neurons = 1920 x 1080 X 3 ~ 6 million

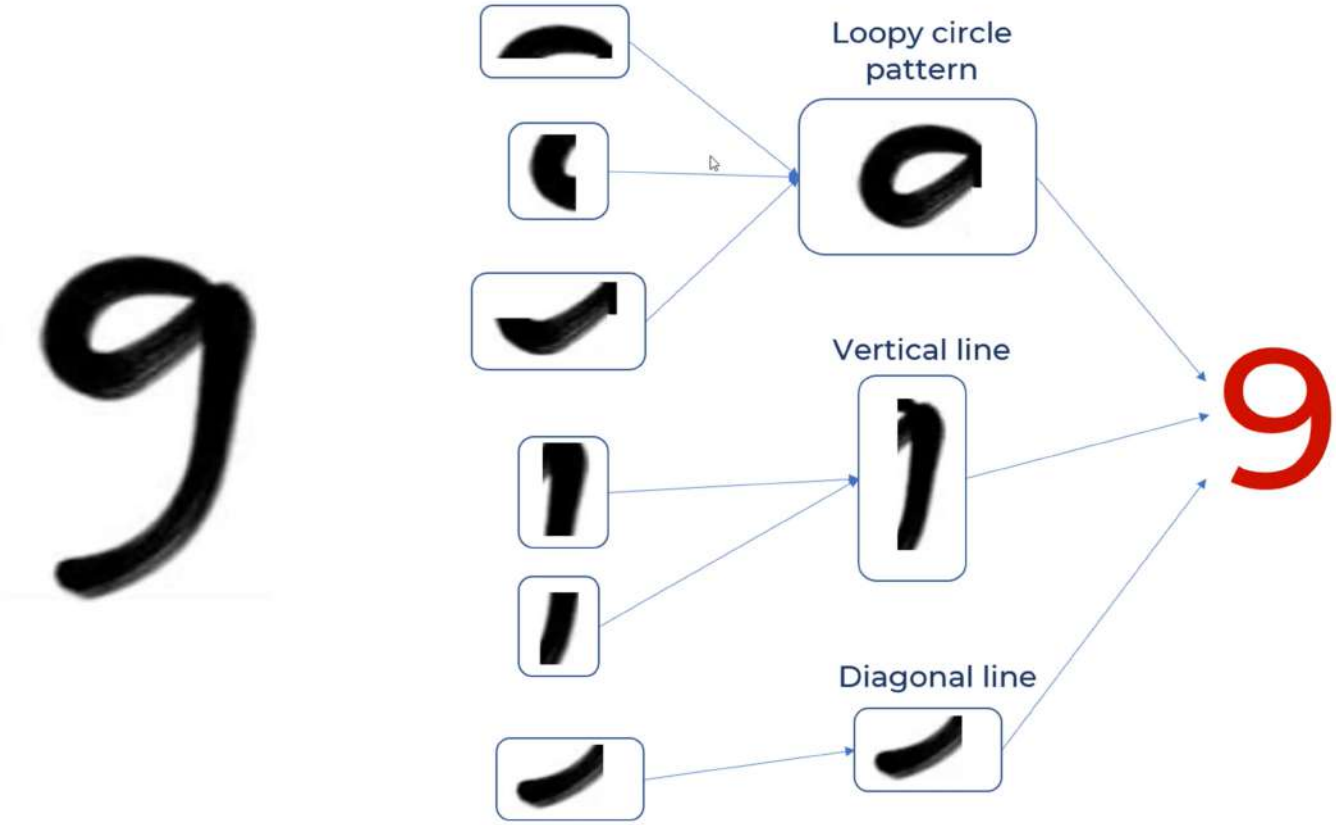
Hidden layer neurons = Let's say you keep it ~ 4 million

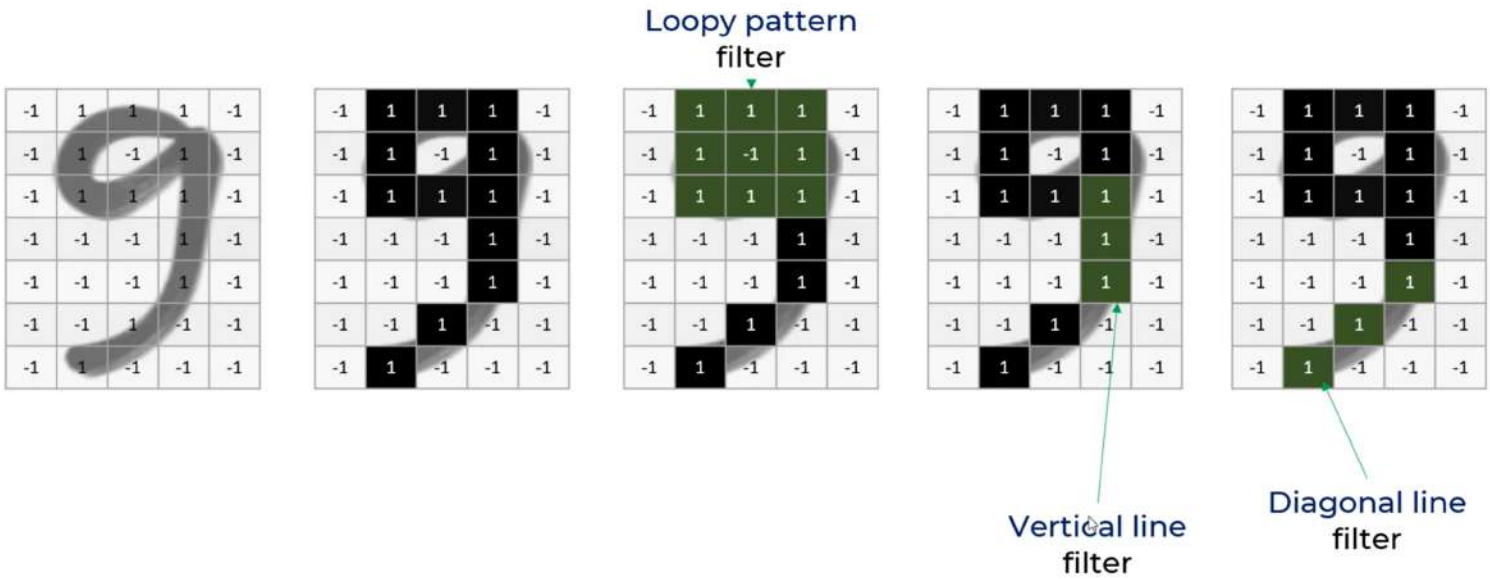
Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

Disadvantages of using ANN for image classification

1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image


Koala's eye? = Y
Koala's nose? = Y
Koala's ears? = Y
Koala's head? = Y
Is it Koala? = Y
Koala's hands? = Y
Koala's legs? = Y
Koala's body? = Y


Loopy circle pattern
Vertical line
Diagonal line
9


Loopy pattern filter
Vertical line filter
Diagonal line filter

-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**	

-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**	

-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**	

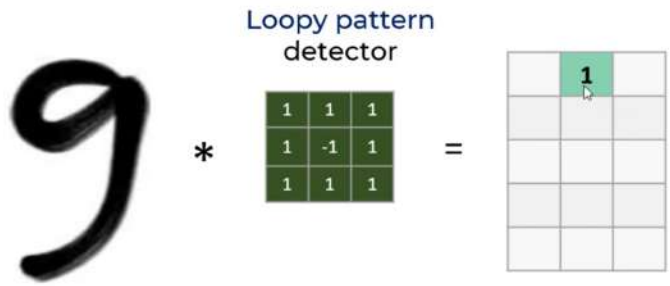
-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

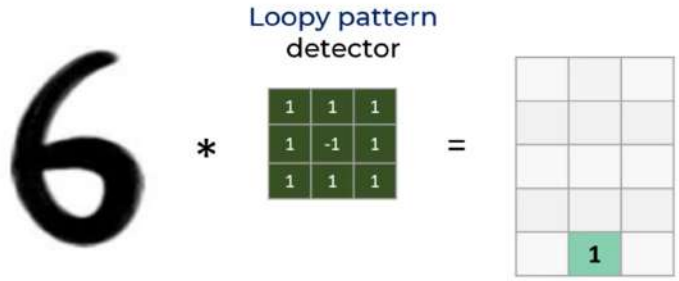
*

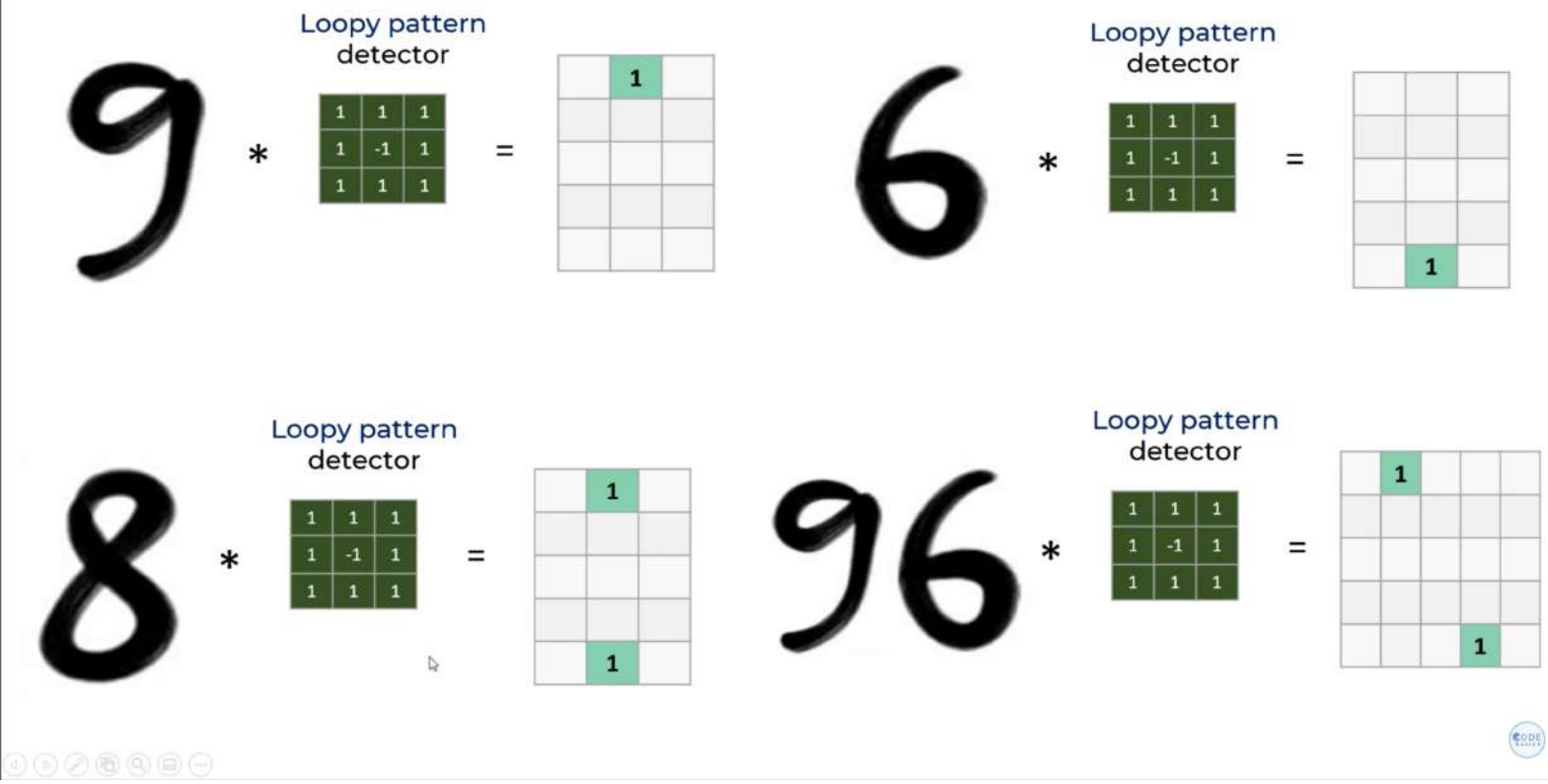
1	1	1
1	-1	1
1	1	1

-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

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

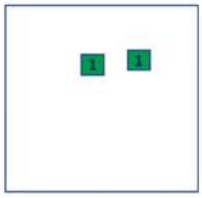

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1
Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1

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

*

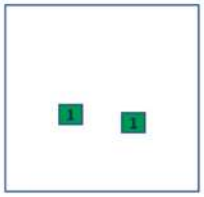
eye
detector

=



eye
detector

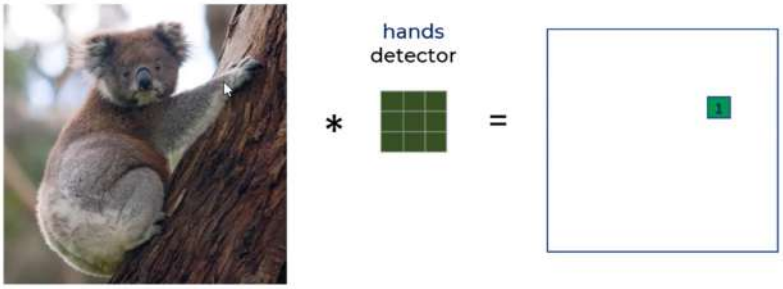
=

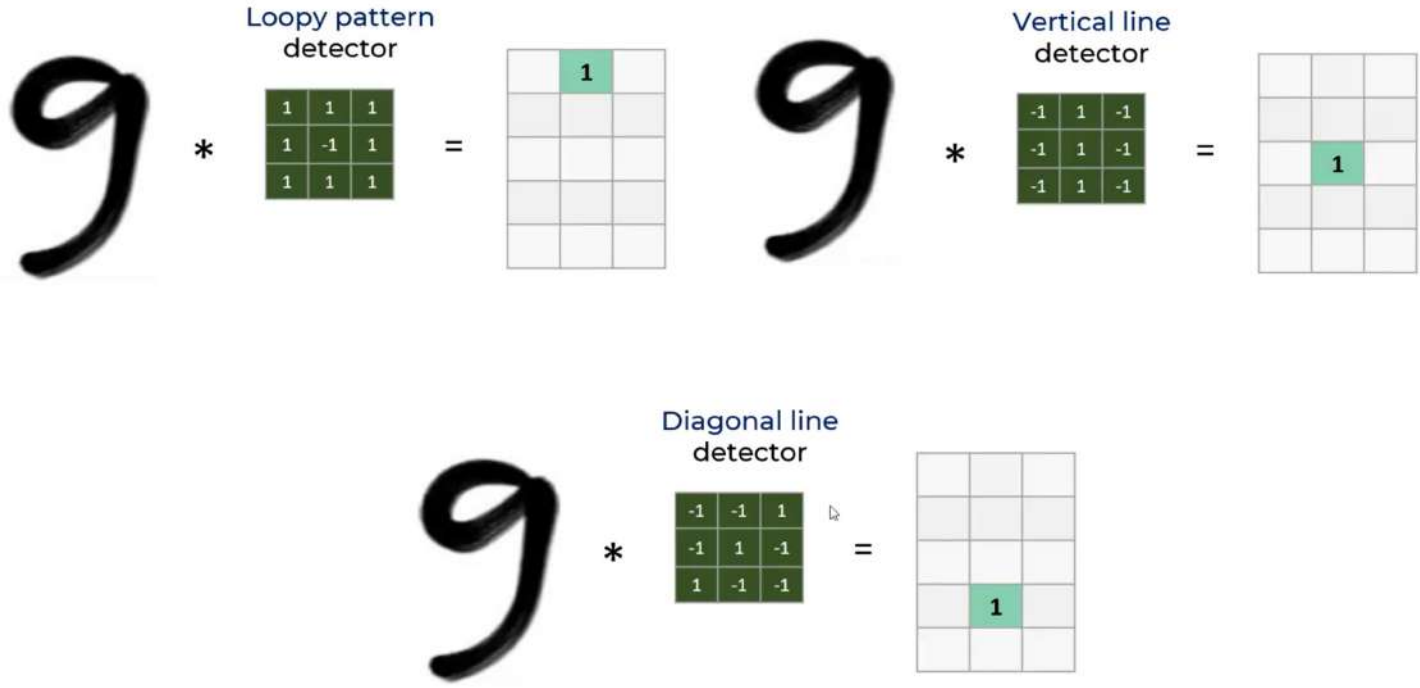


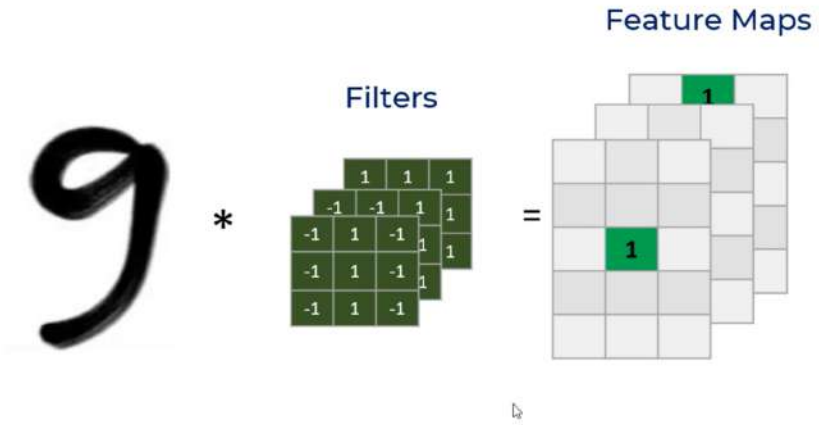
*

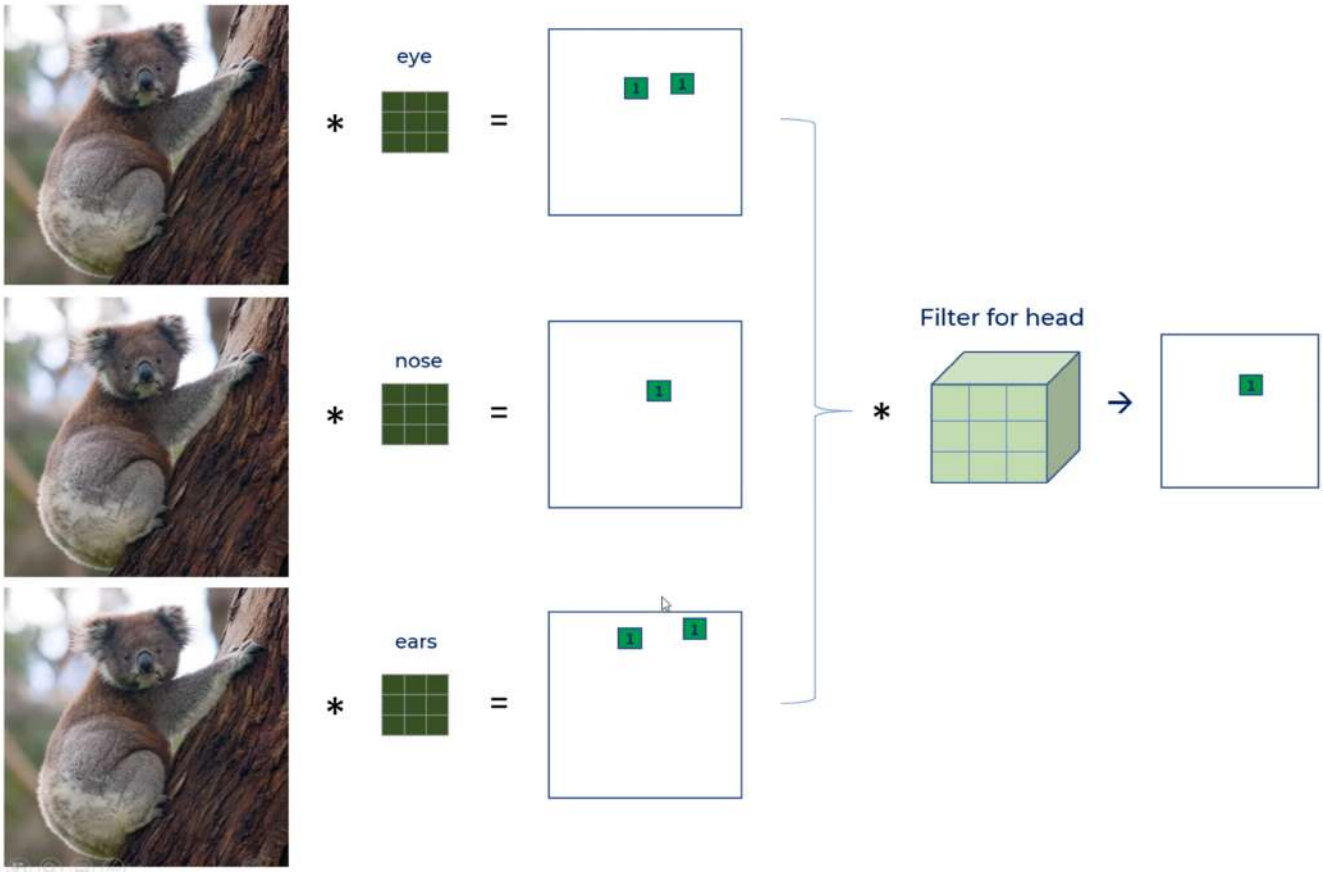
eye
detector

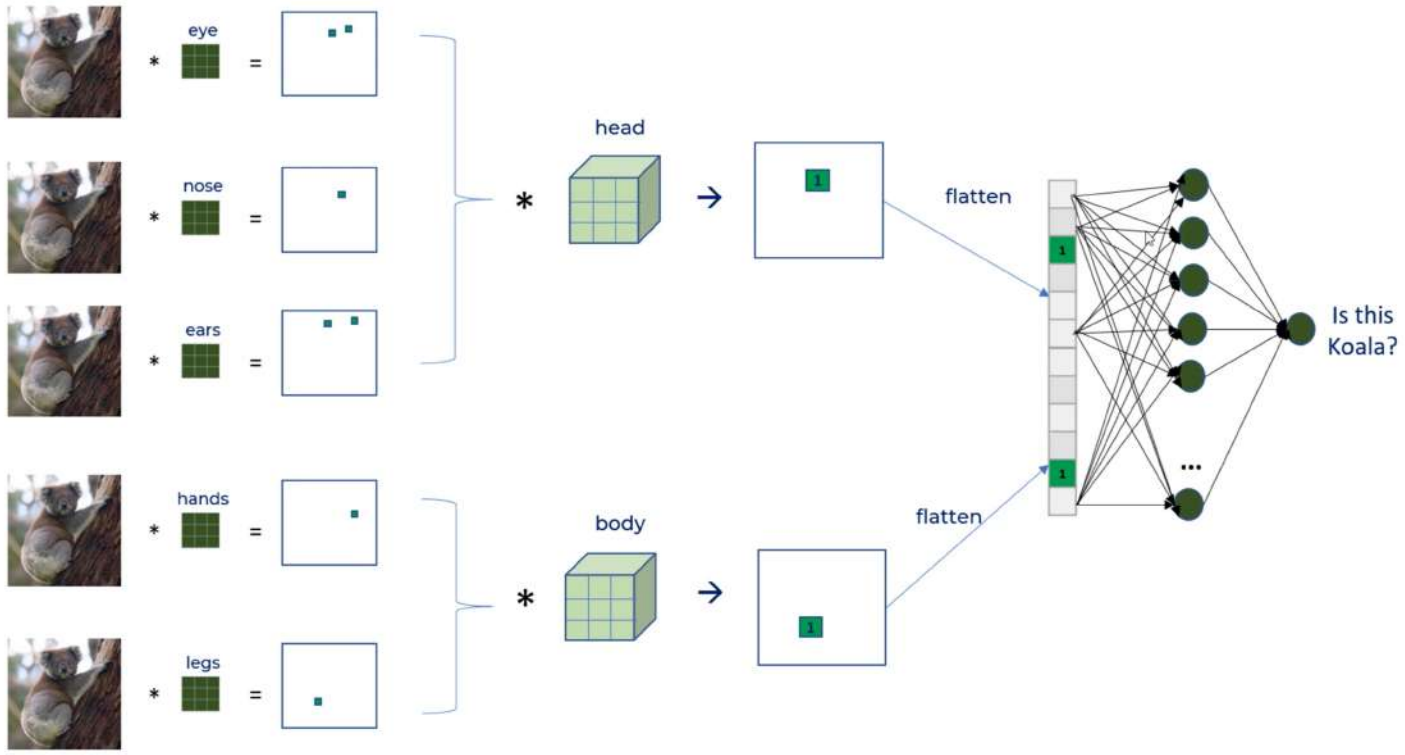
=

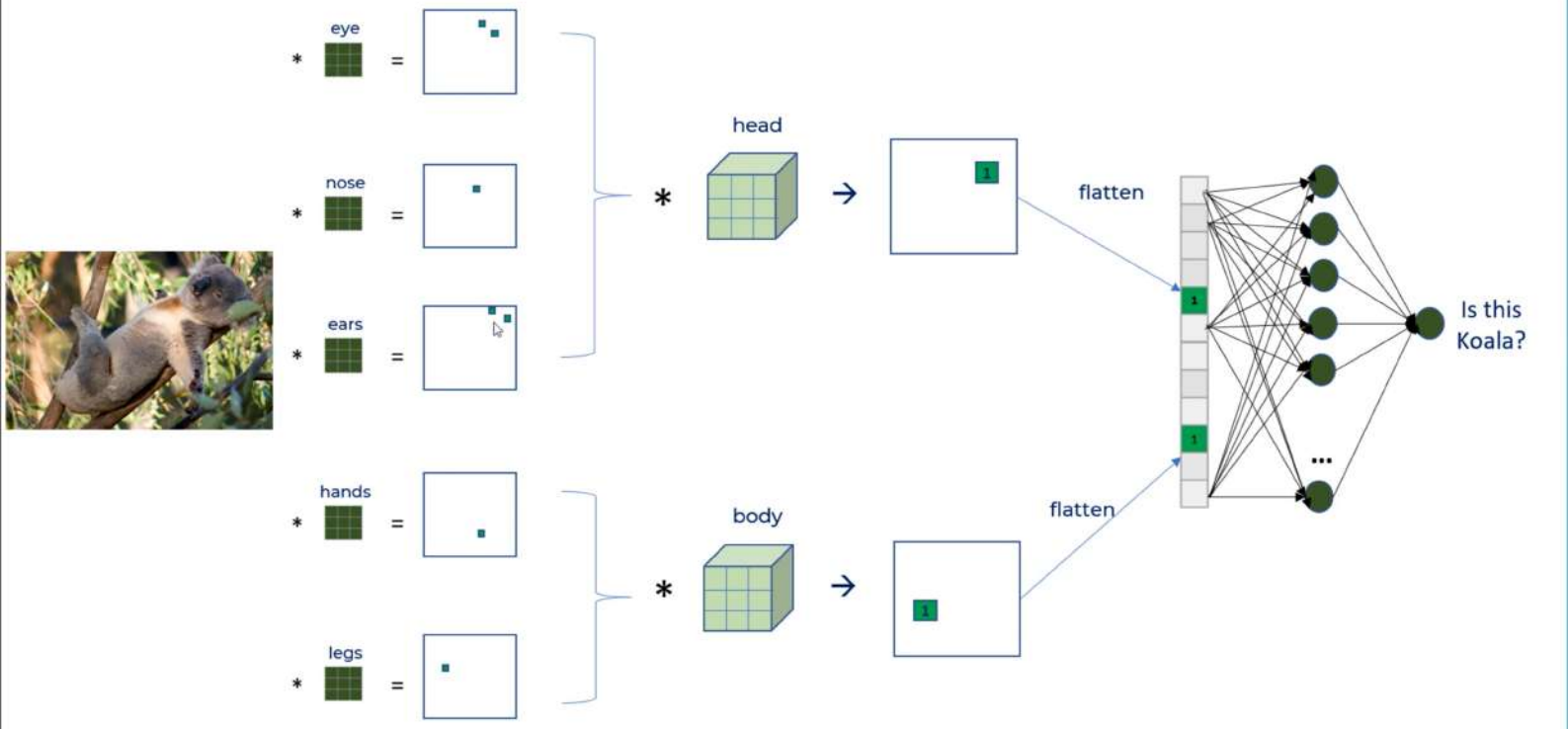

hands
detector
*
=
1

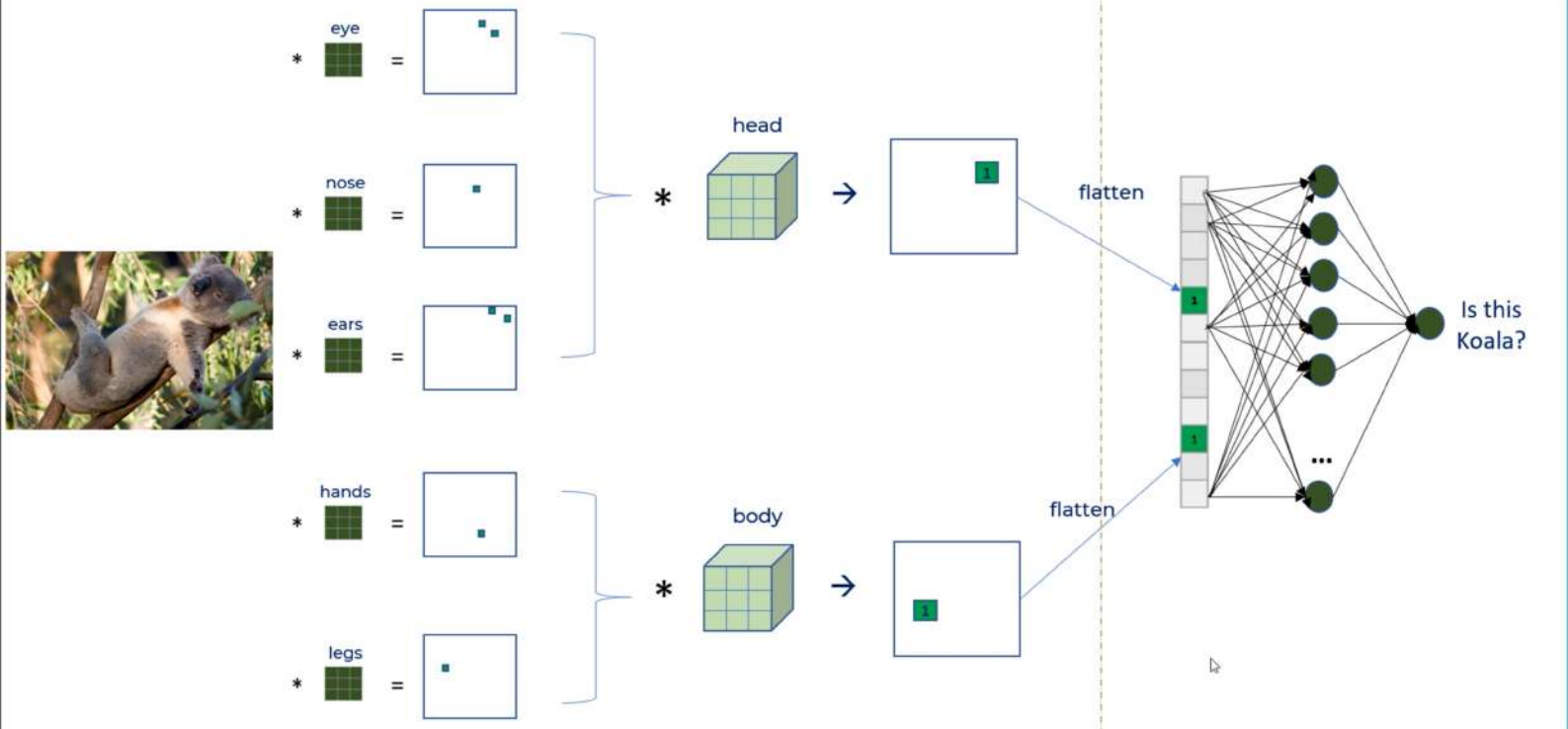

Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
Vertical line detector
-1 1 -1
-1 1 -1
-1 1 -1
1
Diagonal line detector
-1 -1 1
-1 1 -1
1 -1 -1
1


Feature Maps
Filters
*
=
1
1 1 1
-1 -1 1
-1 1 -1
-1 1 -1
-1 1 -1
1
1
1
1


eye
nose
ears
Filter for head


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?


eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?



Feature Extraction

-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

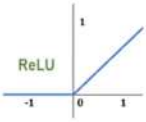
*

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



→

0	1	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

ReLU helps with making the model nonlinear

-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

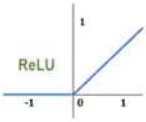
*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

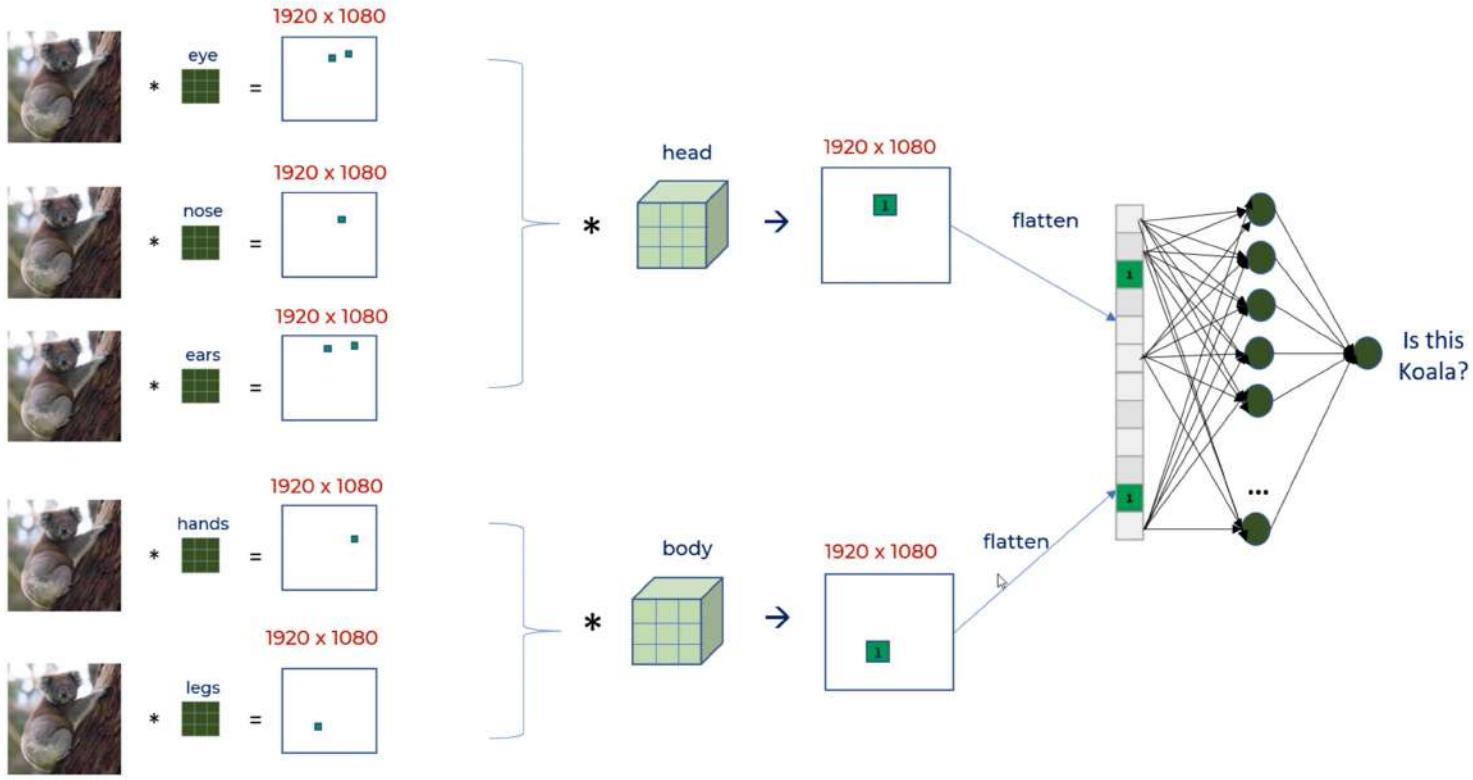
→

-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



→

0	1	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0


1920 x 1080
eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?

Pooling layer is used to reduce the size

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

8	**9**
3	2

2 by 2 filter with stride = 2

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

1	

2 by 2 filter with stride = 1

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

1	**1**
0.33	0.33
0.33	0.33
0	0

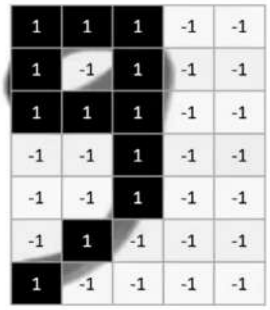
2 by 2 filter with stride = 1

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

1	**1**
0.33	0.33
0.33	0.33
0	0

2 by 2 filter with stride = 1

Shifted 9 at different position



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

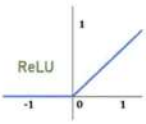
*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

→

1	-0.11	-0.11
0.11	-0.33	0.33
0.33	-0.33	-0.33
-0.11	-0.55	-0.33
-0.55	-0.33	-0.55



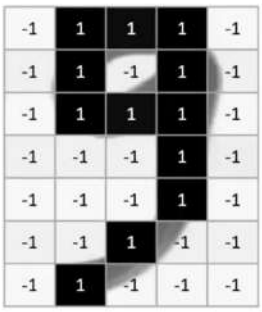
→

1	0	0
0.11	0	0.33
0.33	0	0
0	0	0
0	0	0

Max pooling

→

1	0.33
0.33	0.33
0.33	0
0	0



-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

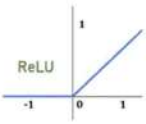
*

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



→

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

Max pooling

→

1	1
0.33	0.33
0.33	0.33
0	0

There is average pooling also...

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

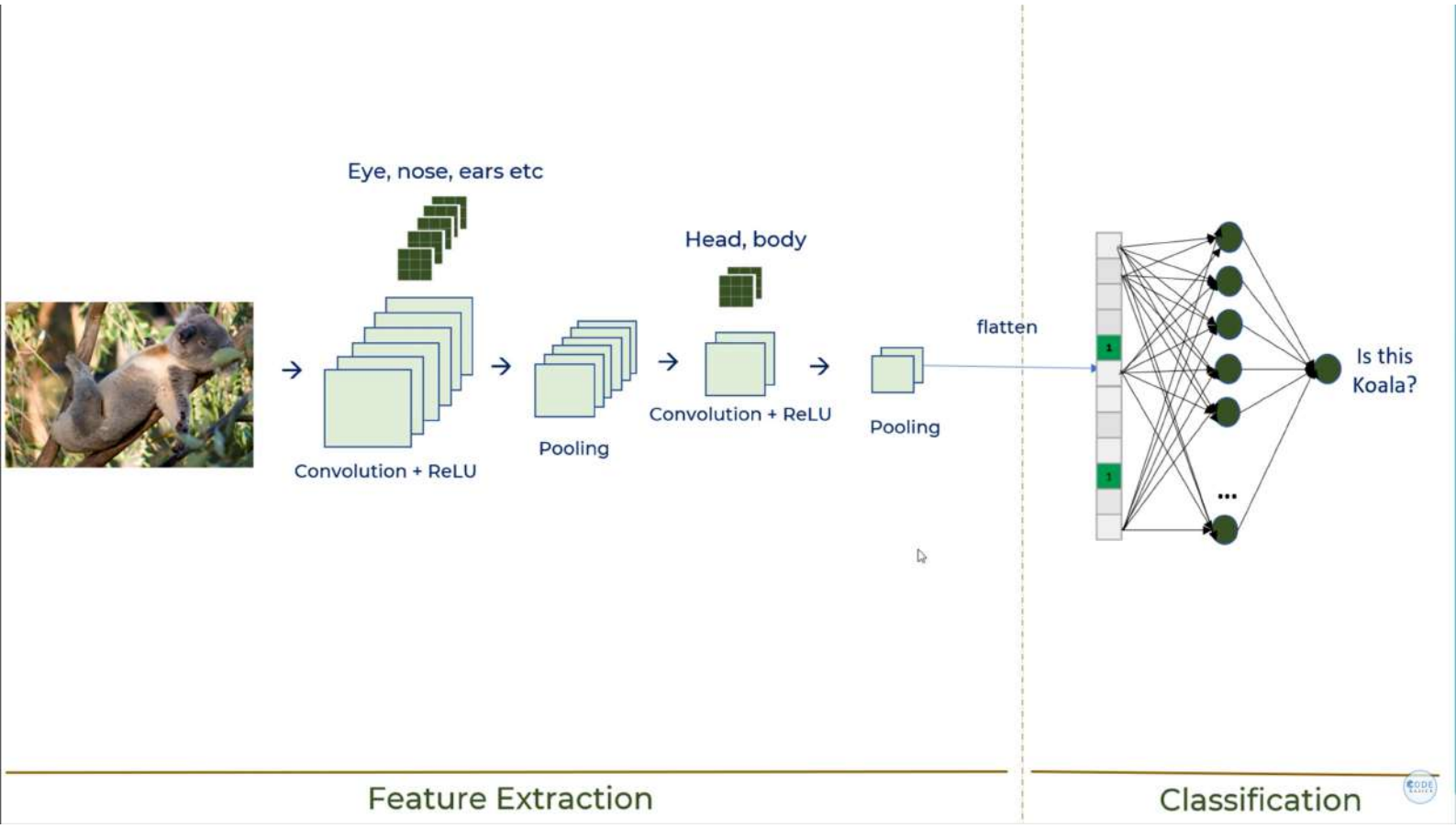
4	4.5
2	0.75

Benefits of pooling

Reduces dimensions & computation

Reduce overfitting as there are less parameters

Model is tolerant towards variations, distortions


Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
1
1
...
Is this Koala?
Feature Extraction
Classification

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

Rotation

Thickness

CNN by itself doesn't take care of rotation and scale

- You need to have rotated, scaled samples in training dataset
- If you don't have such samples than use data augmentation methods to generate new rotated/scaled samples from existing training samples