

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

Convolution

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

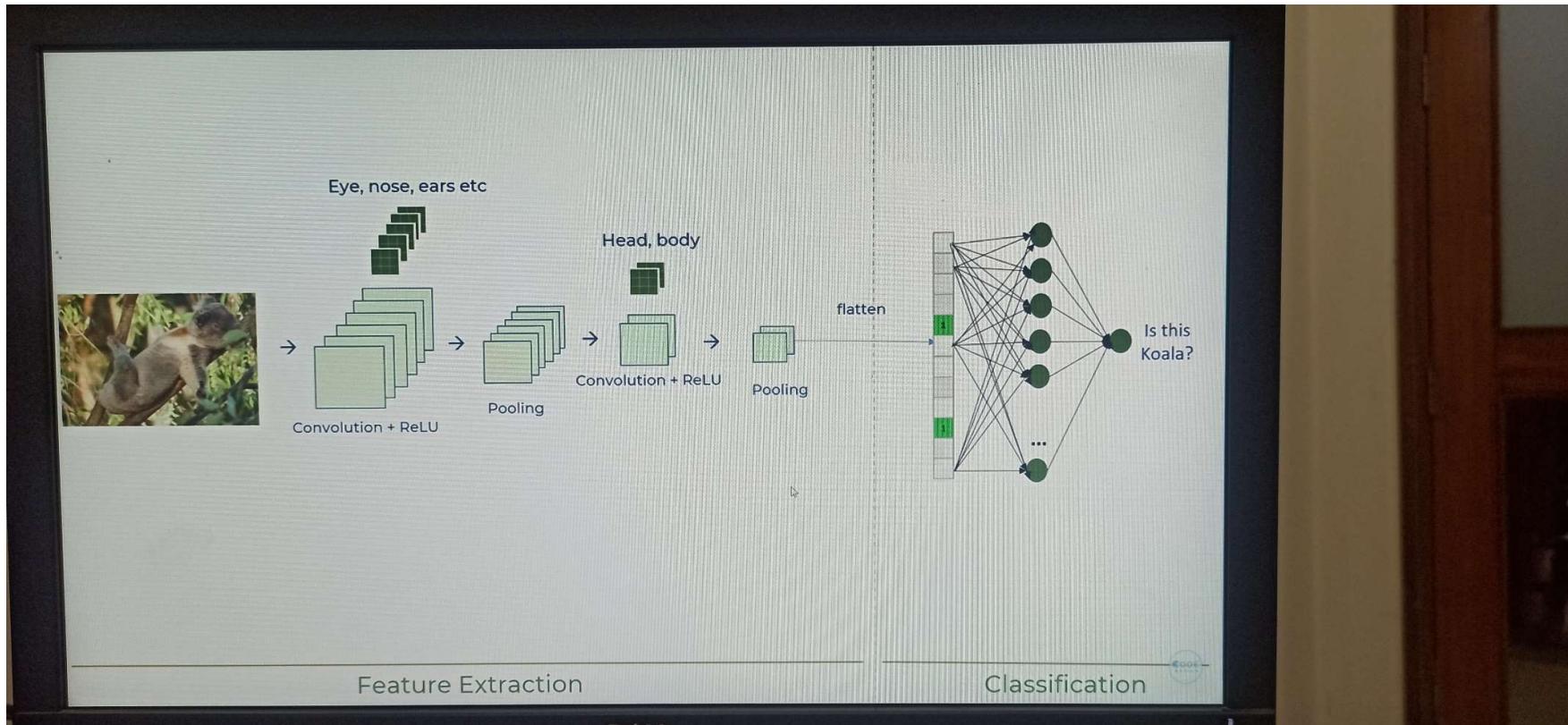
ReLU

- Introduces nonlinearity
- Speeds up training, faster to compute

Pooling

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

DELL



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

Shifted 9 at
different position

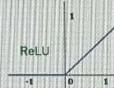
1	1	1	-1	-1
1	-1	1	-1	-1
1	1	1	-1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1
1	-1	-1	-1	-1

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

Shifted 9 at
different position

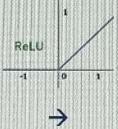
1	1	1	-1	-1
1	-1	1	-1	-1
1	1	1	-1	-1
-1	-1	1	-1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1
1	-1	-1	-1	-1

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

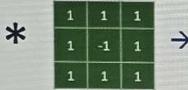
0	1	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

1	1
0.33	0.33
0.33	0.33
0	0

2 by 2 filter with stride = 1

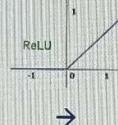
-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

Loopy pattern filter



$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow$$

-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



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

Pooling layer is used to
reduce the size

DELL

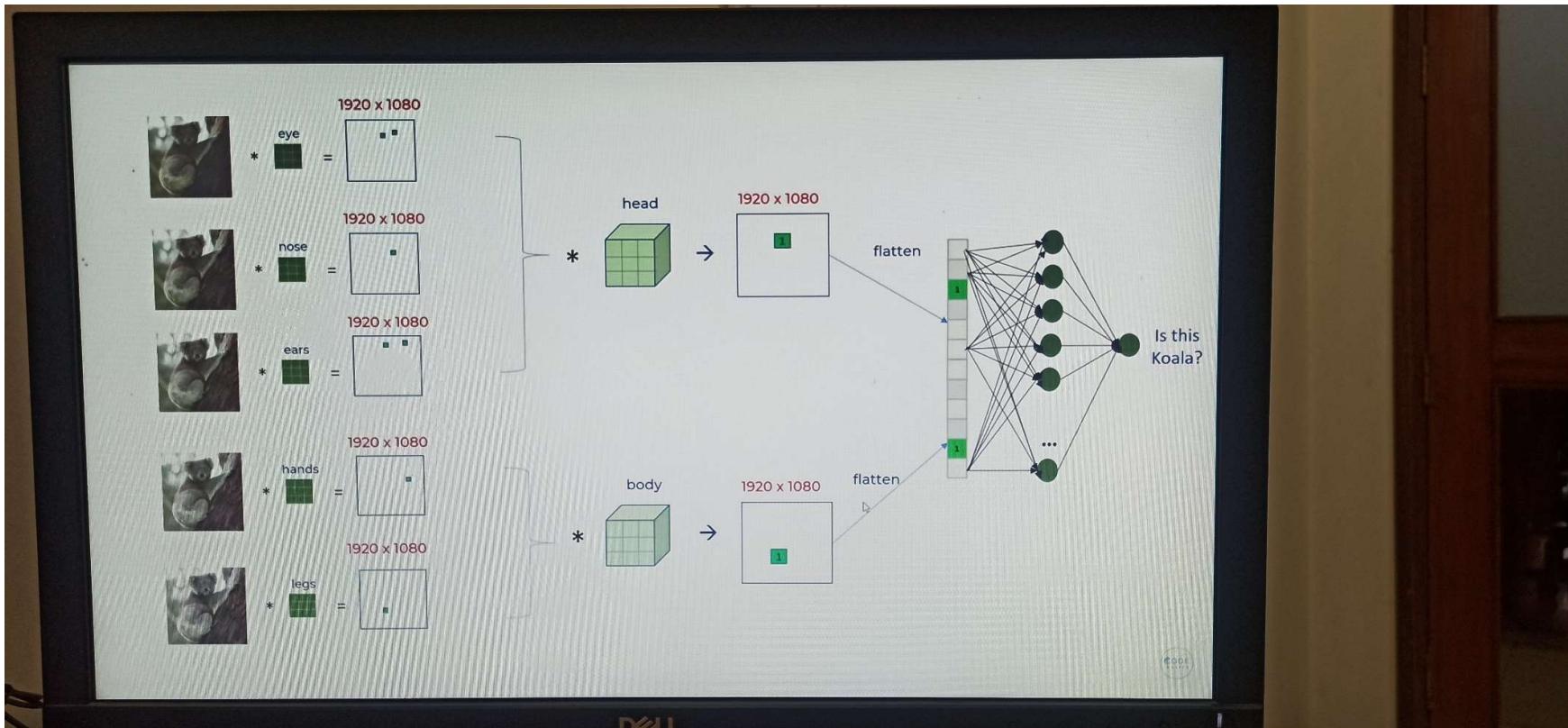




Image size = 1920 x 1080 X 3

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 = $6 \text{ mil} * 4 \text{ mil}$
= 24 million

ReLU helps with making
the model nonlinear



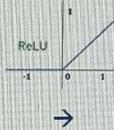
-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

Loopy pattern
filter

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

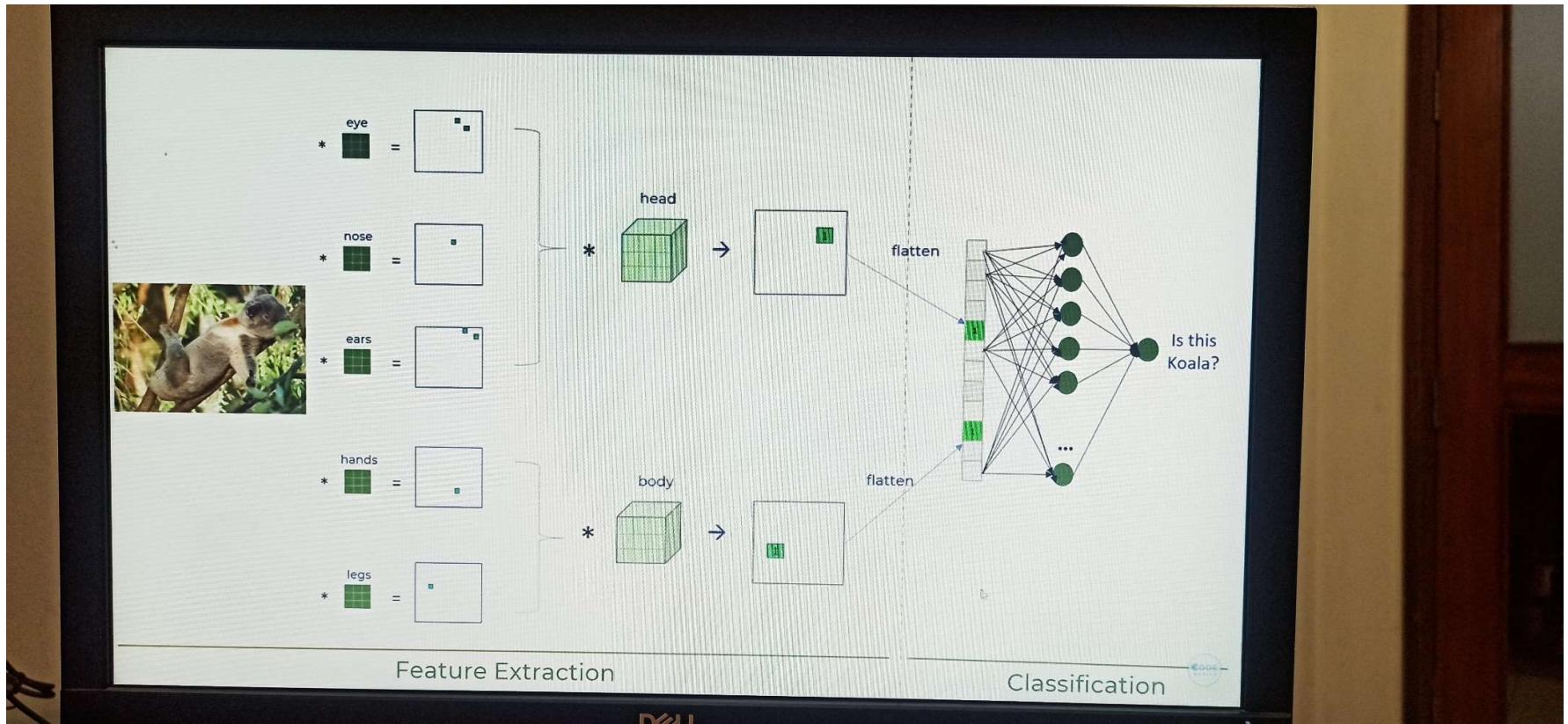
*

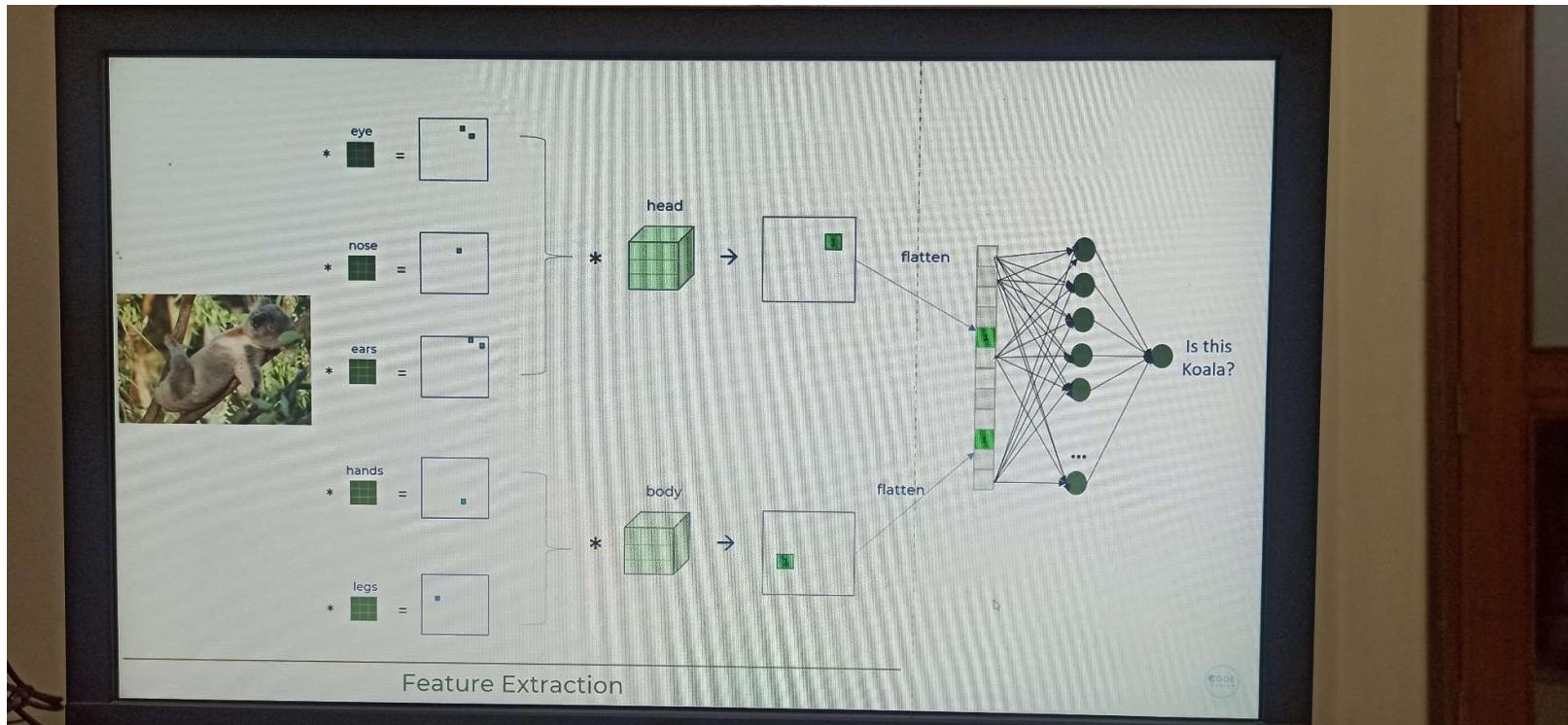
-0.11	1	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

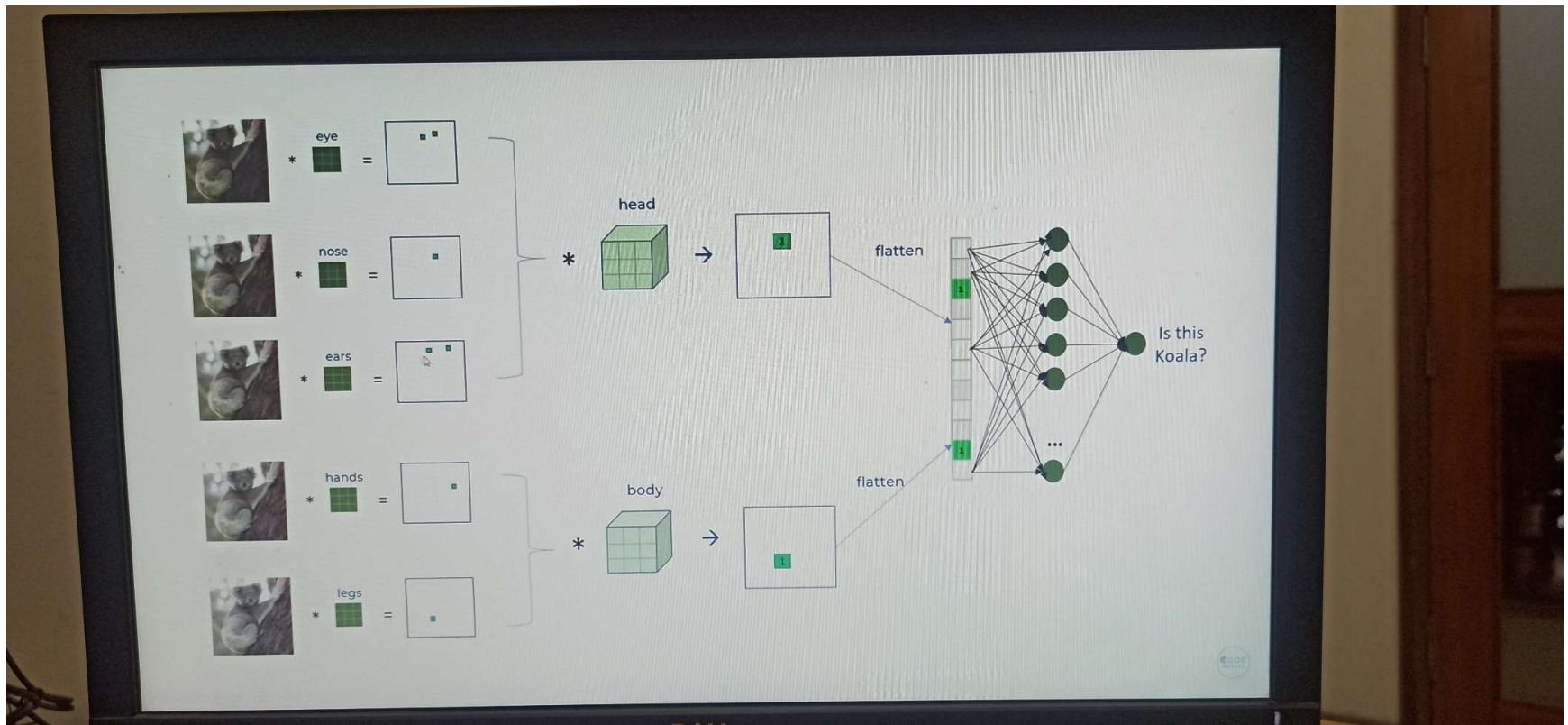


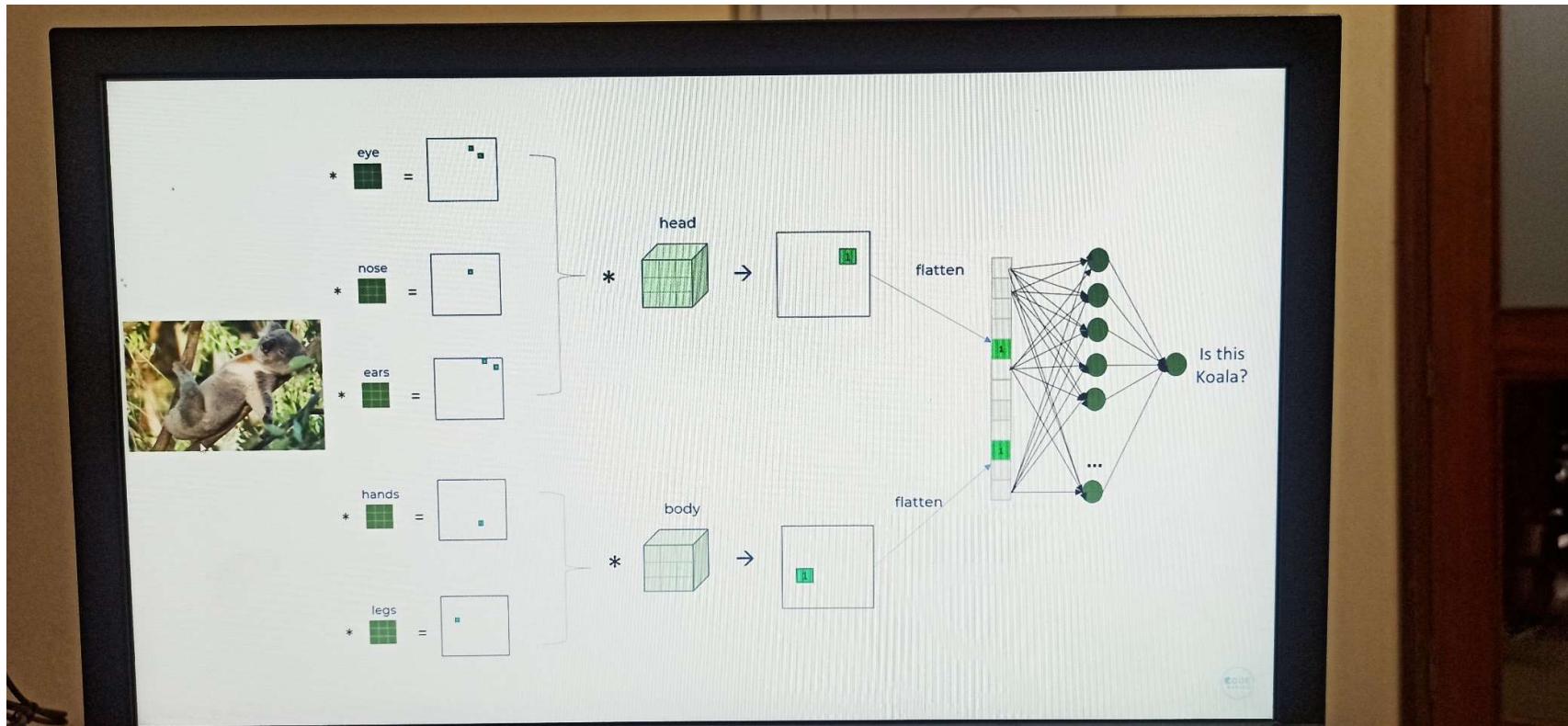
0	1	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

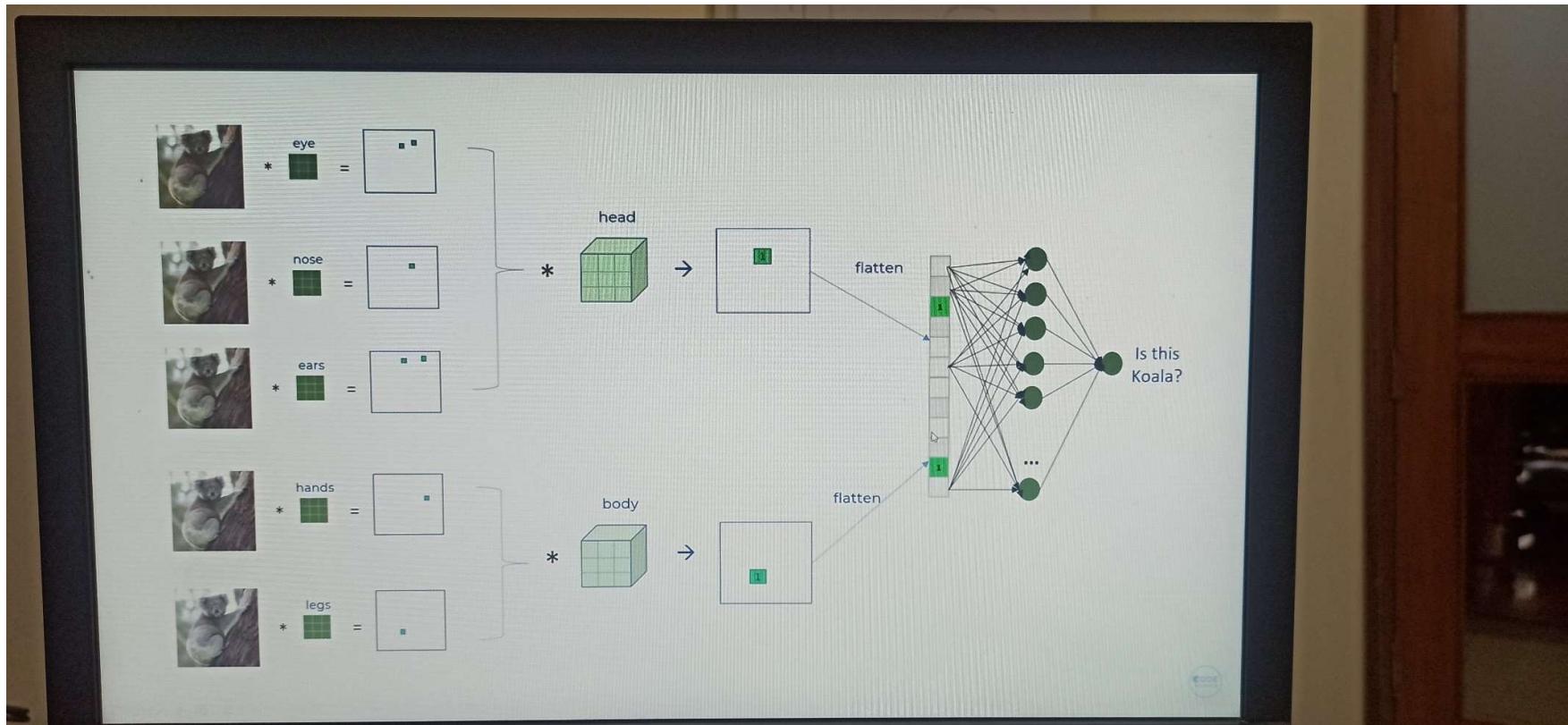
DELL

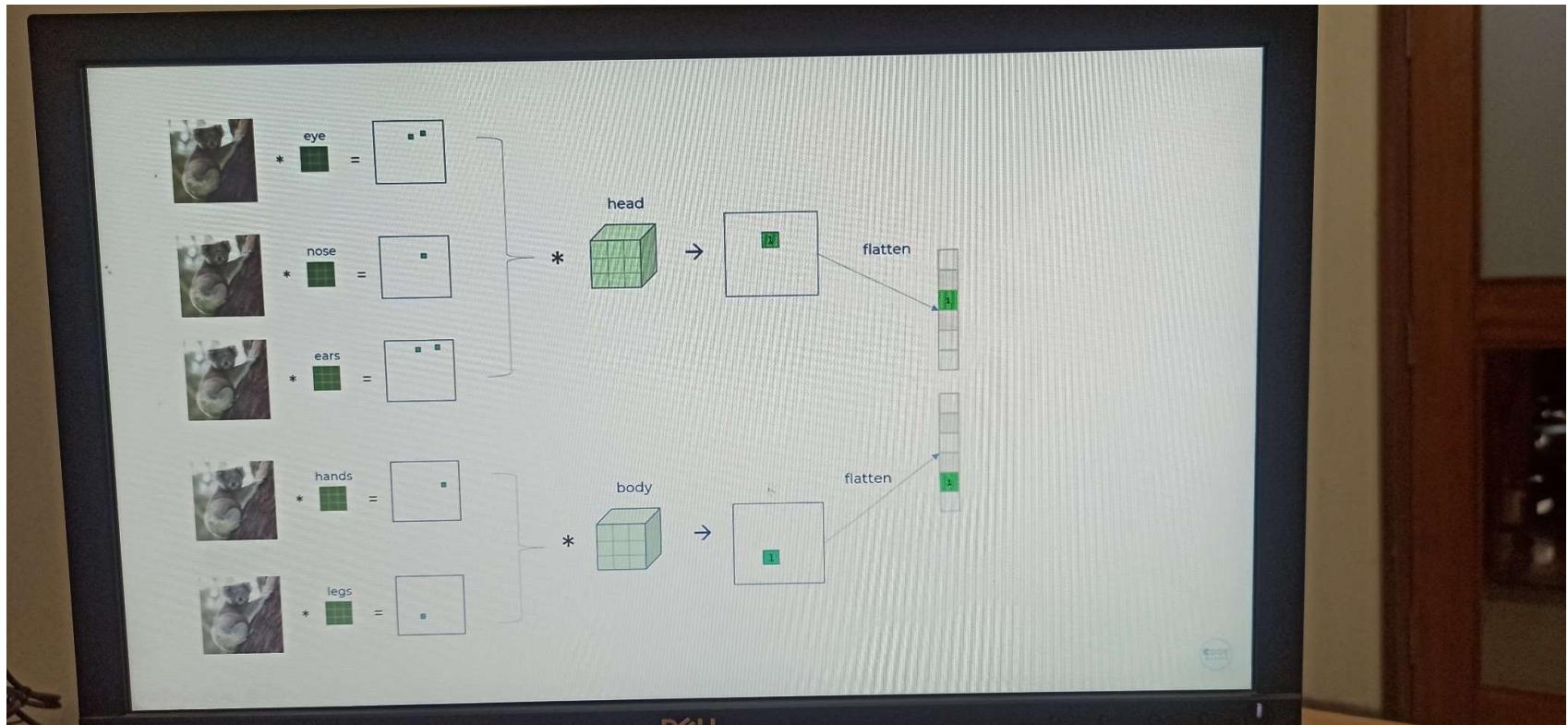


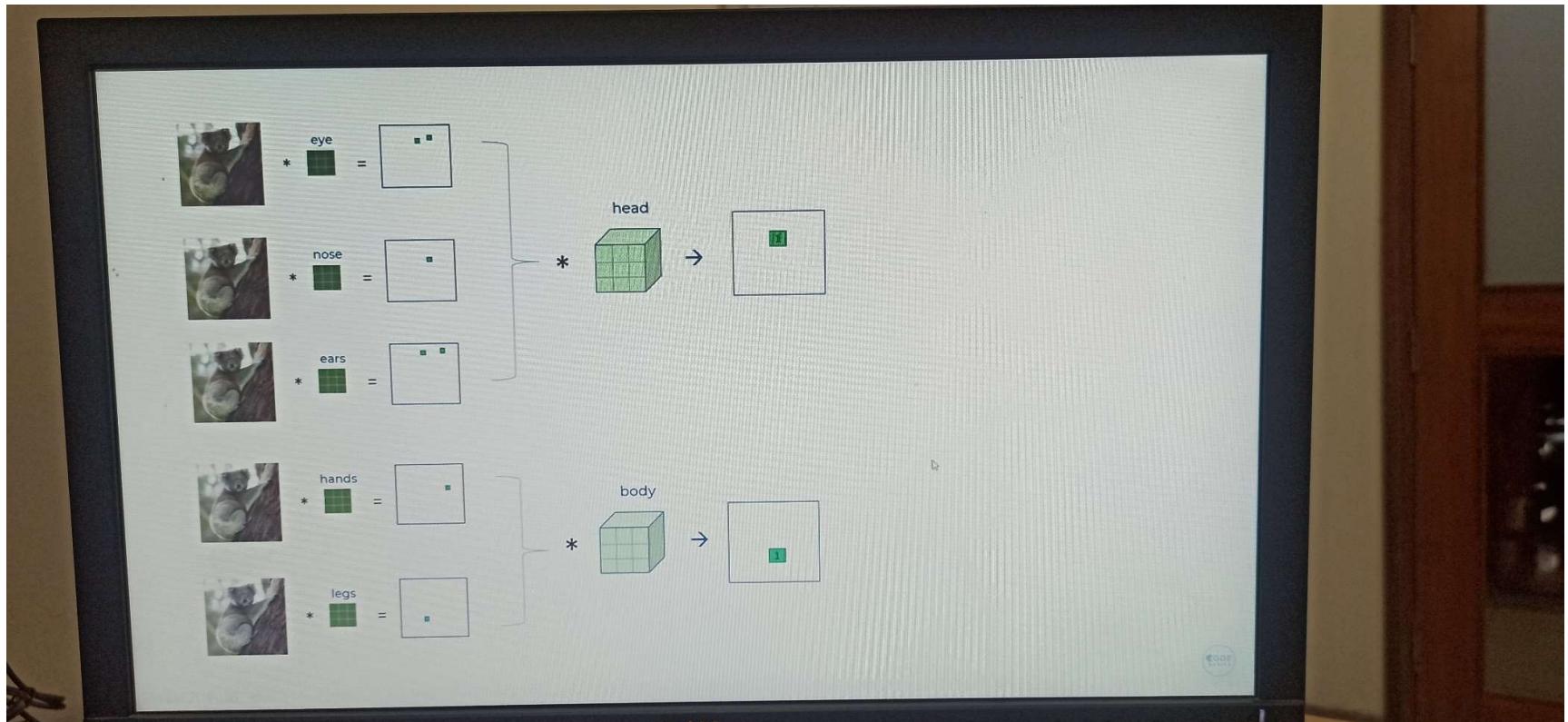


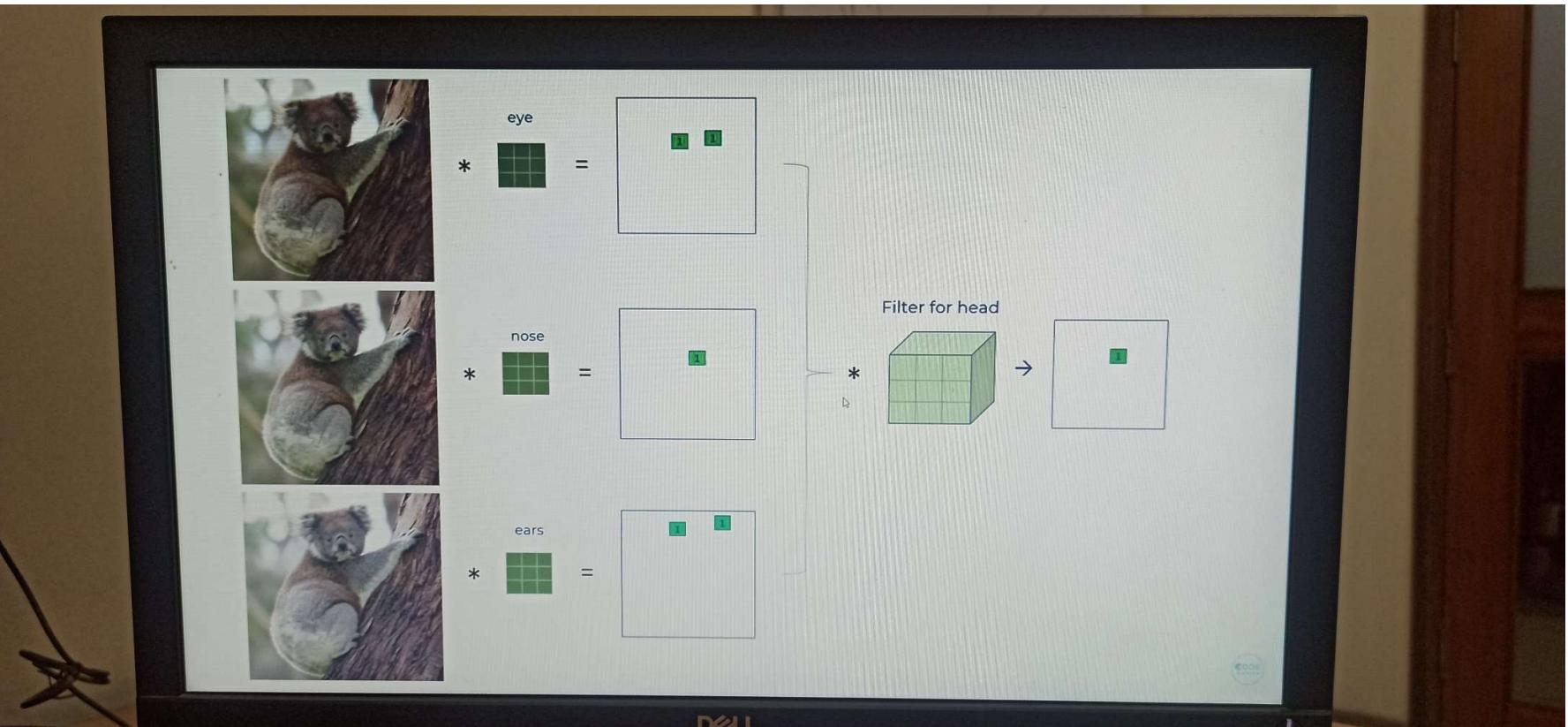


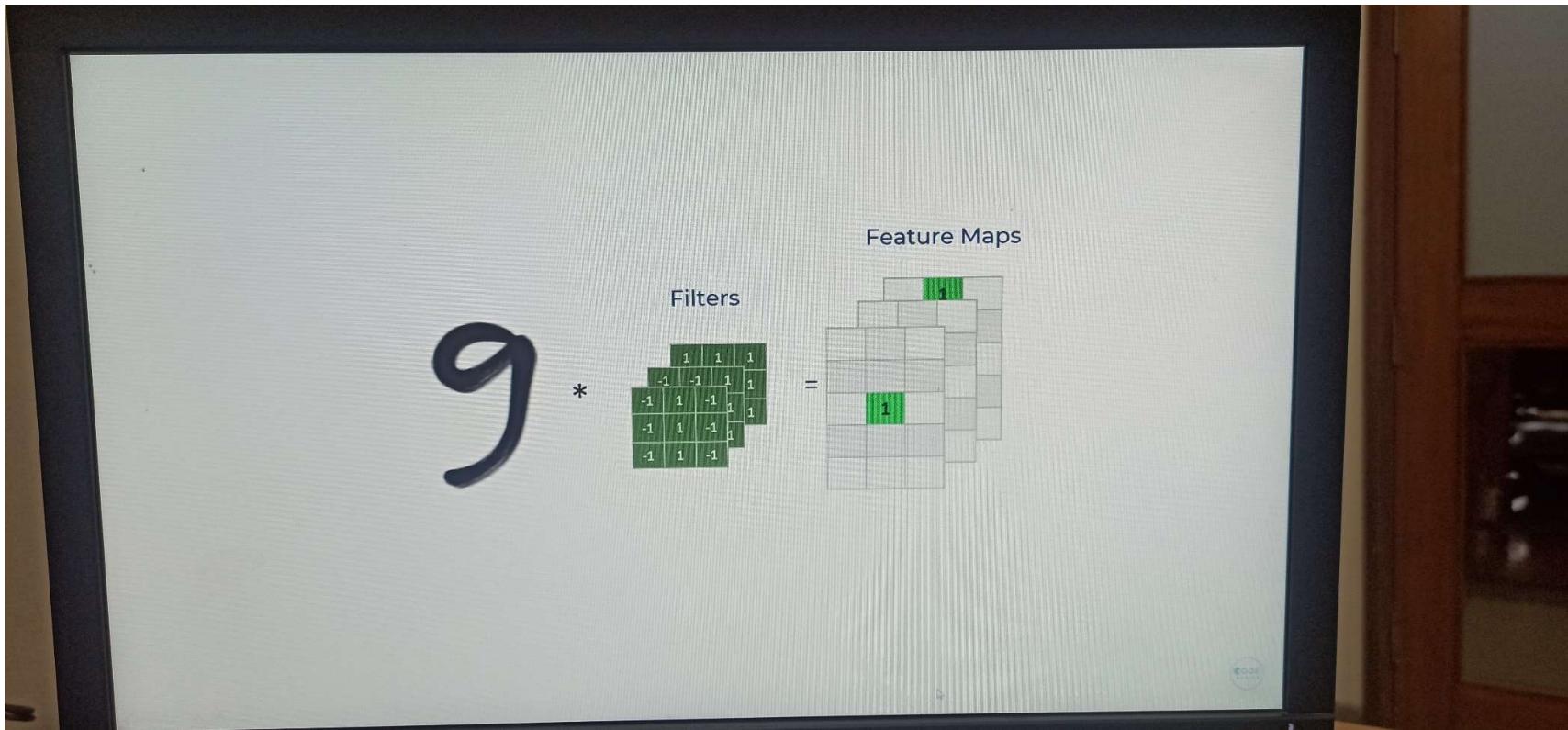












9

Loopy pattern
detector

$$\begin{matrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

*

=

			1	

9

Vertical line
detector

$$\begin{matrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{matrix}$$

*

=

			1	

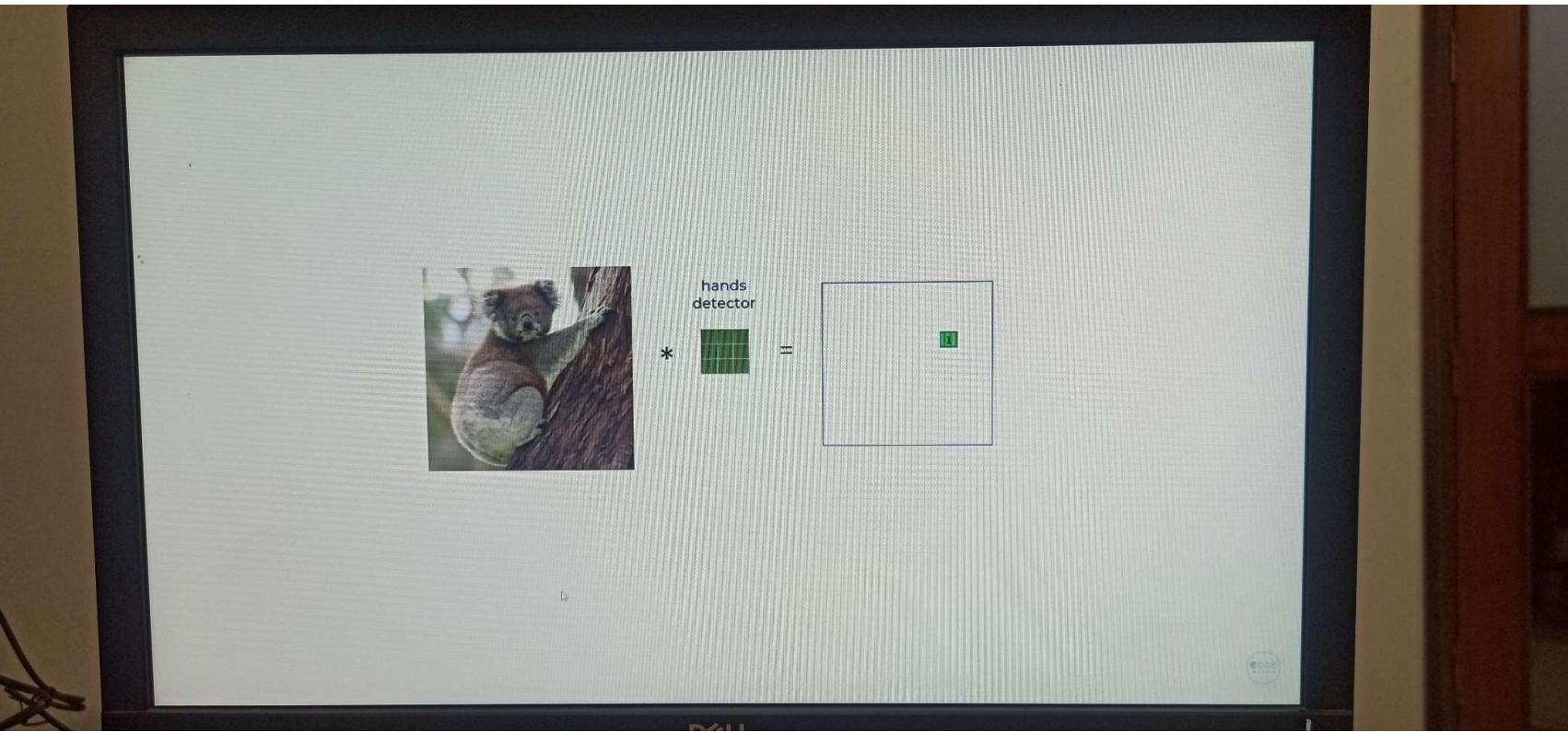
9

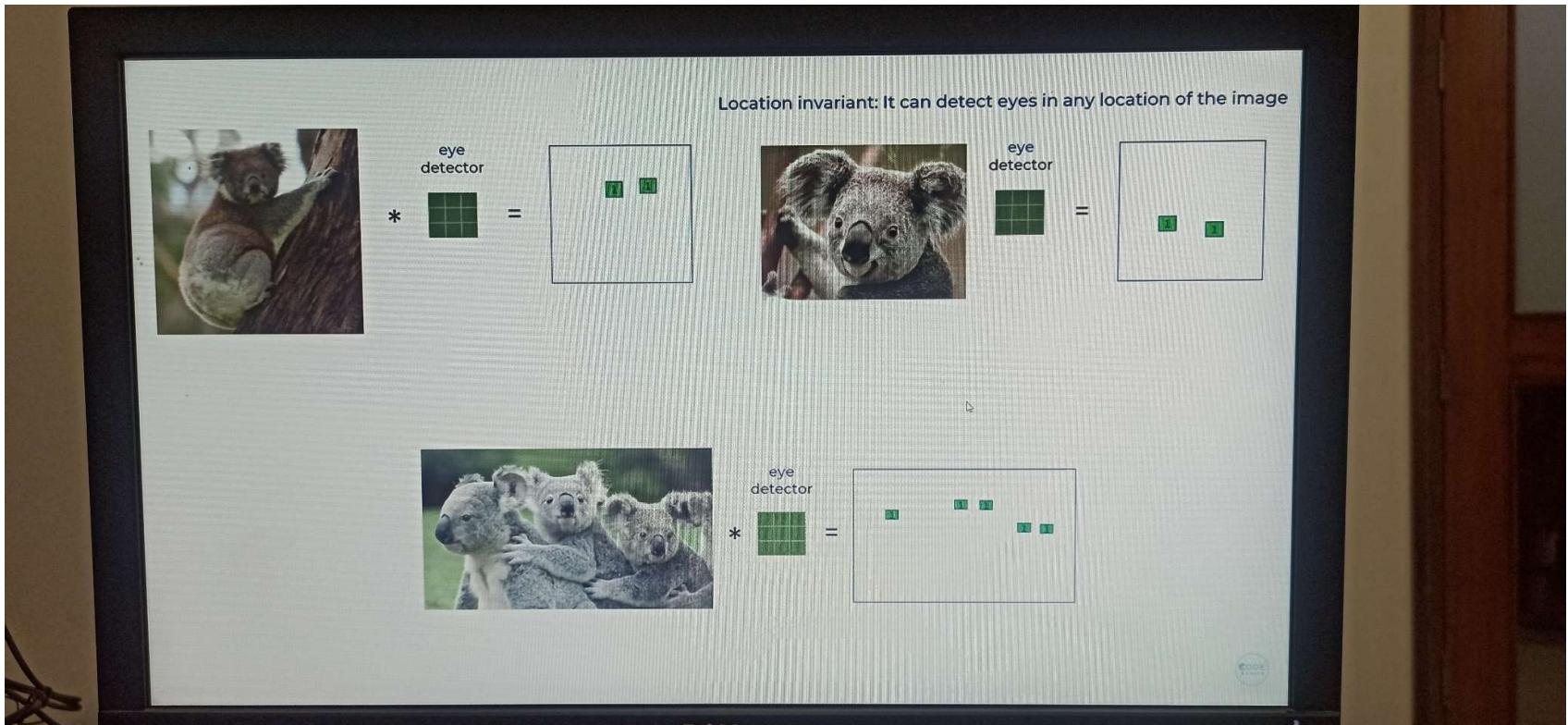
Diagonal line
detector

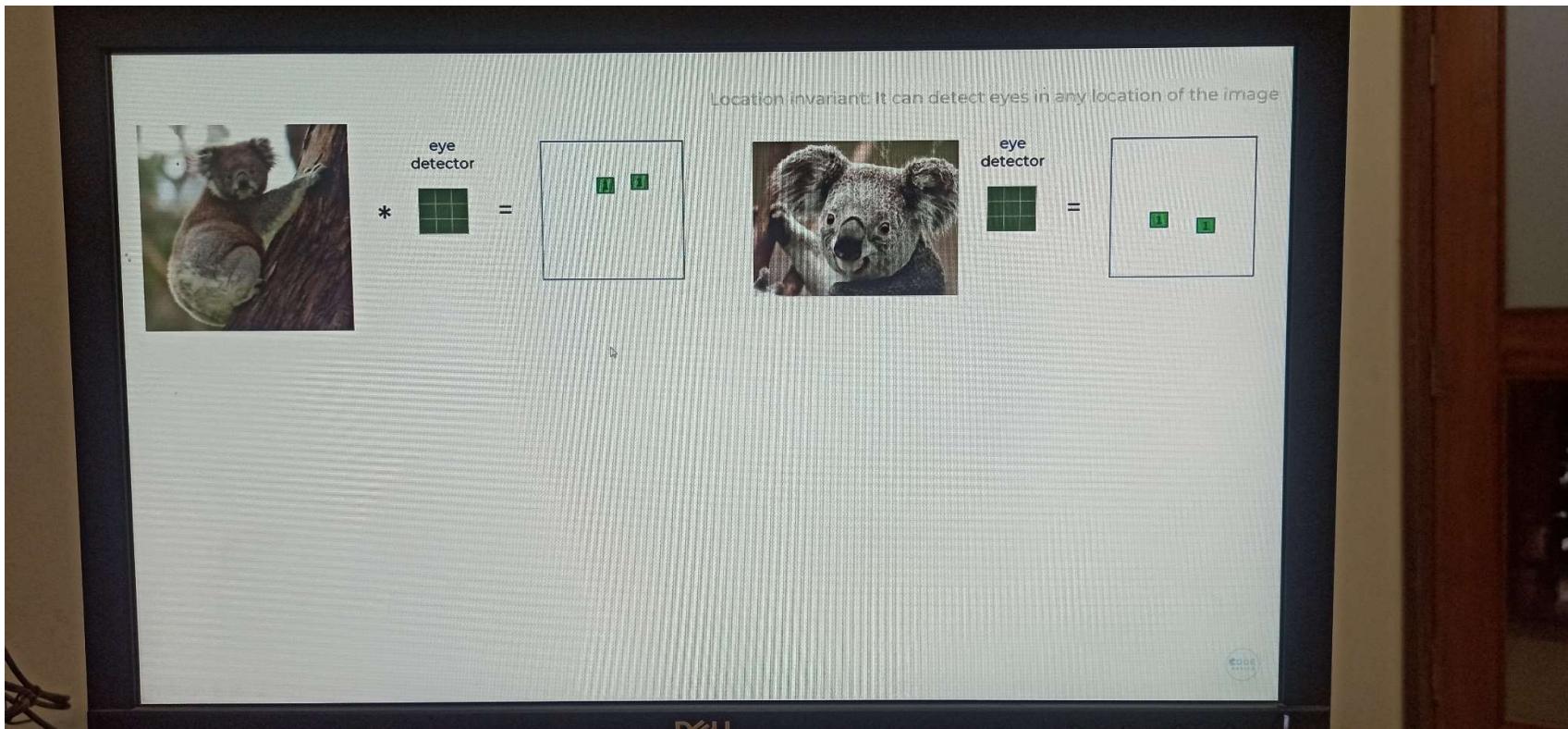
$$\begin{matrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{matrix}$$

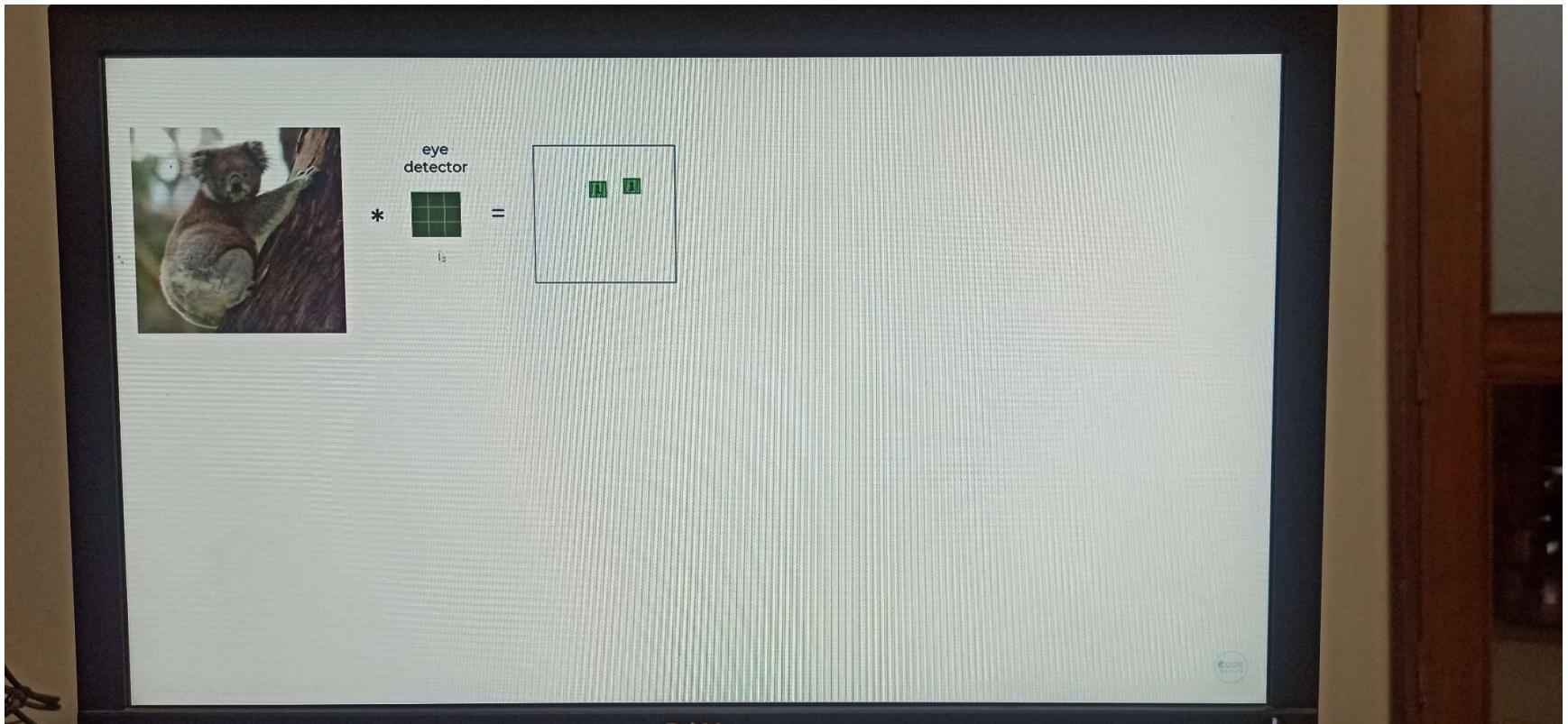
*

1









**Filters are nothing
but the feature
detectors**

DELL

9 * Loopy pattern detector

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

=



6 * Loopy pattern detector

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

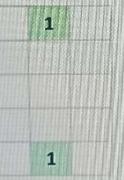
=



8 * Loopy pattern detector

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

=



96 * Loopy pattern detector

$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & -1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

=



CODE

9

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

8

Loopy pattern
detector

1	1	1
1	-1	1
1	1	1

*

1

1

9

Loopy pattern
detector

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

*

=

		1	

6

Loopy pattern
detector

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

*

=

CODE
SCHOOL

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

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

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

-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

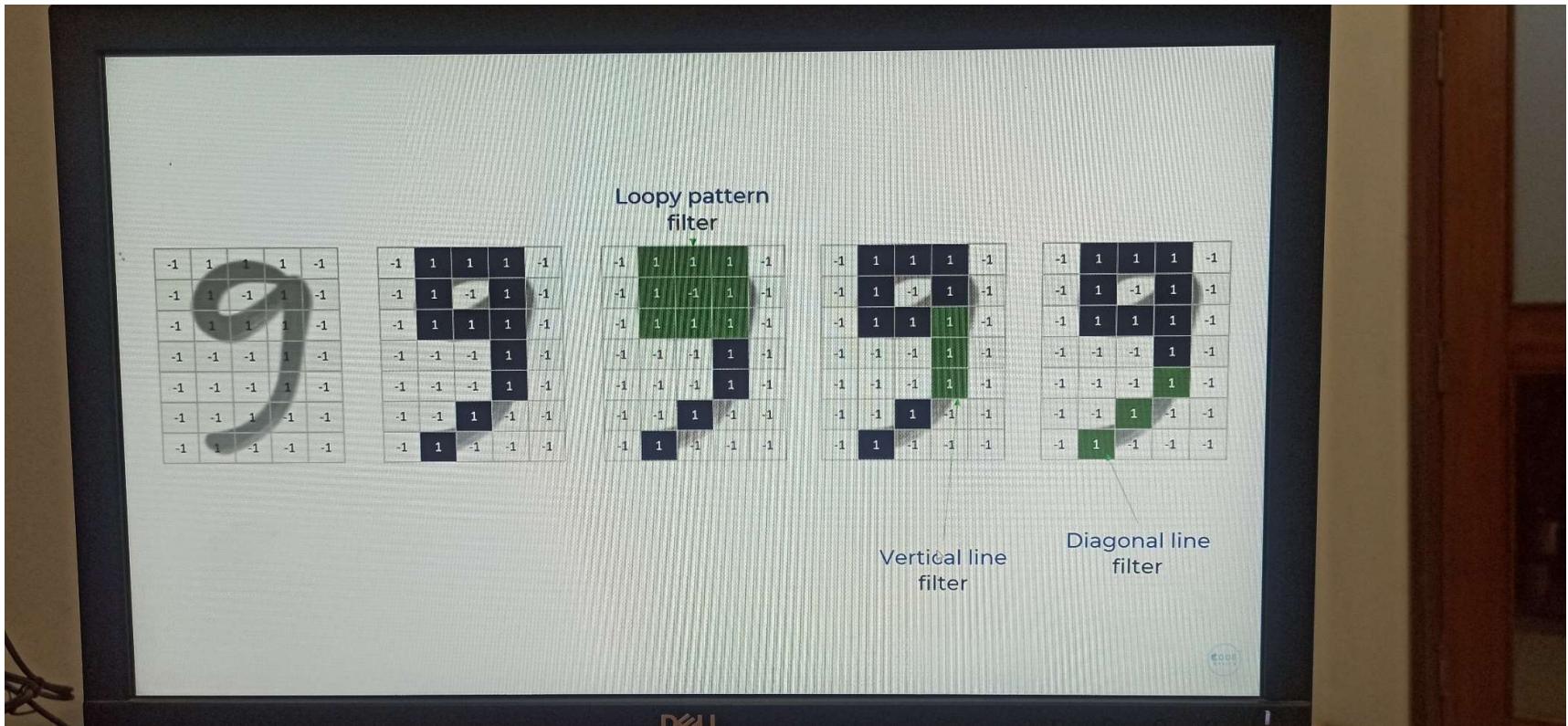
*

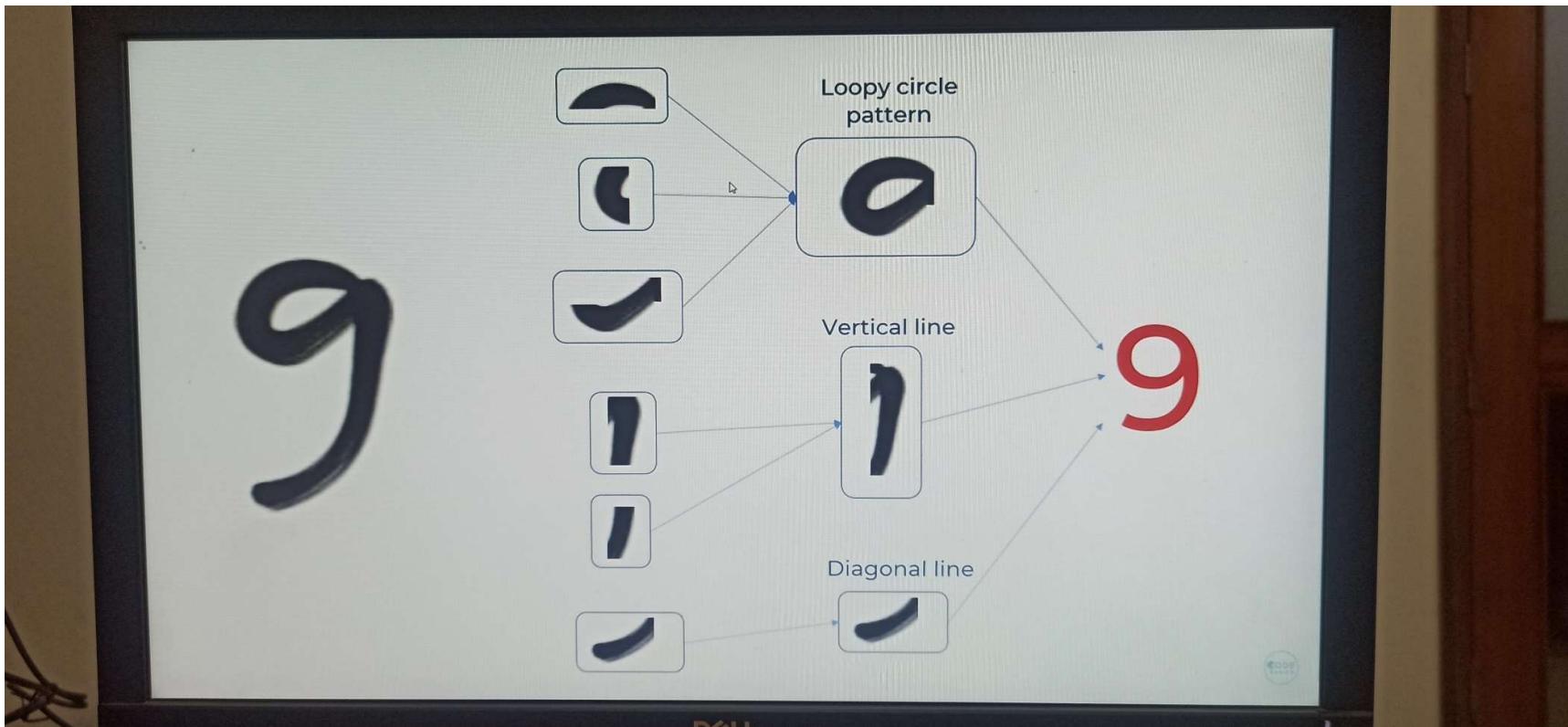
1	1	1
1	-1	1
1	1	1

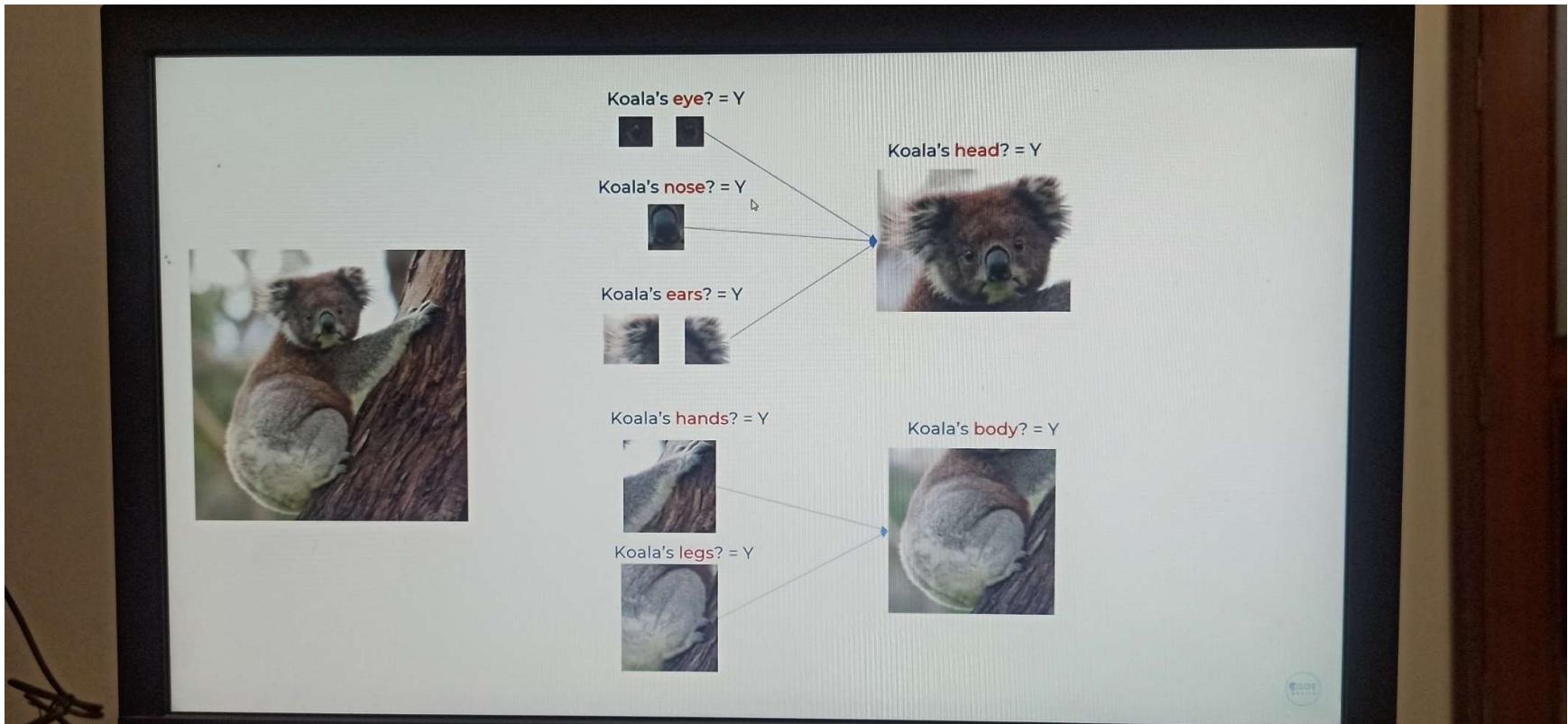
-0.11		

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

1	1	1
1	-1	1
1	1	1







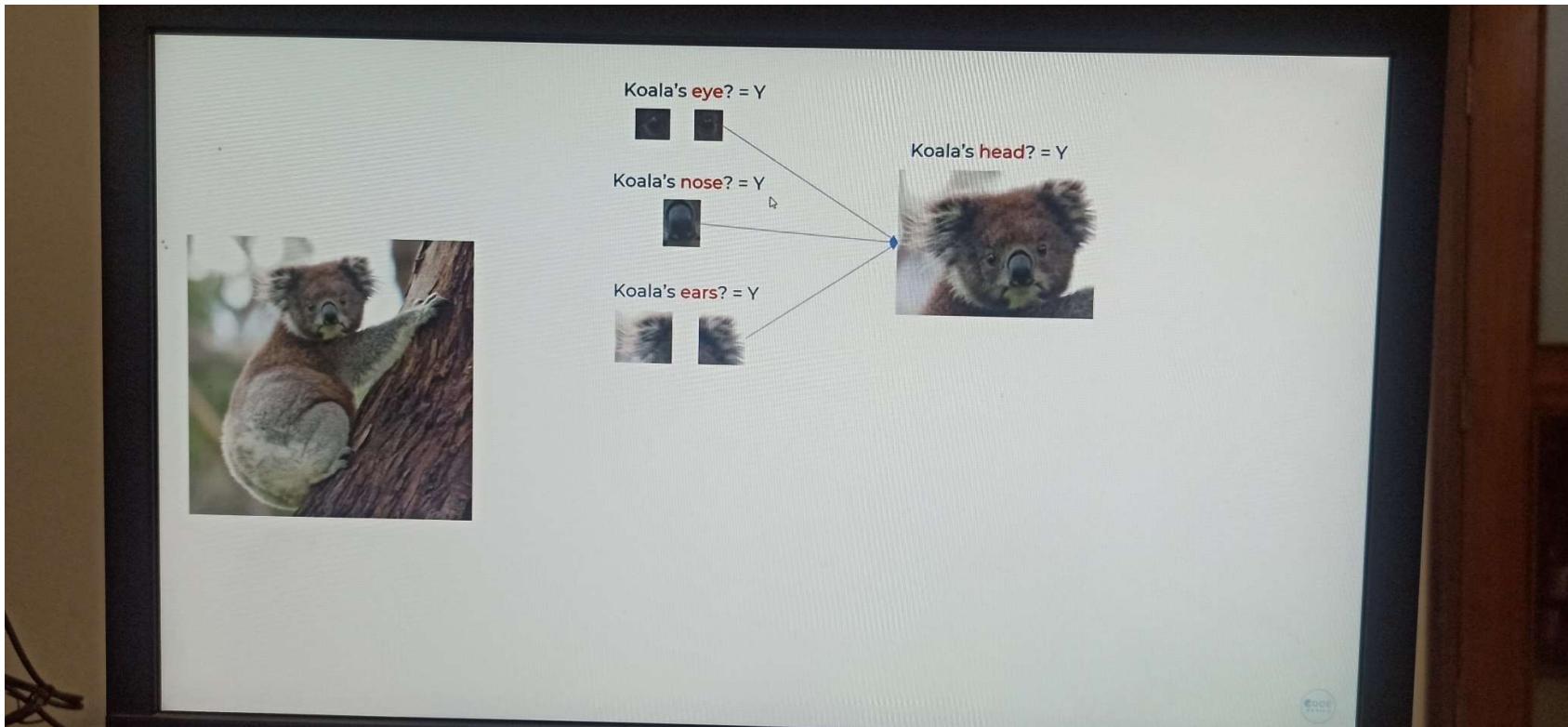


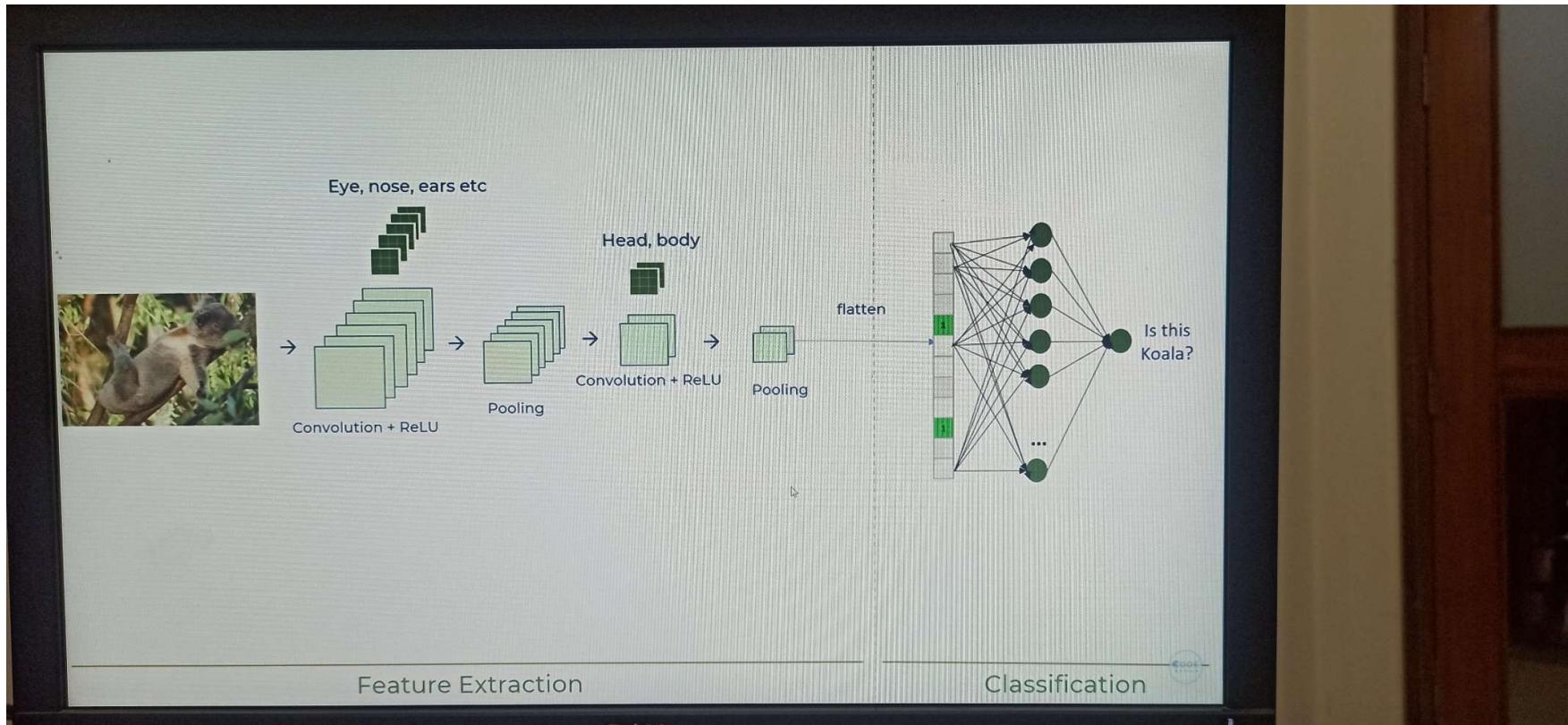


Image size = 1920 x 1080 X 3

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

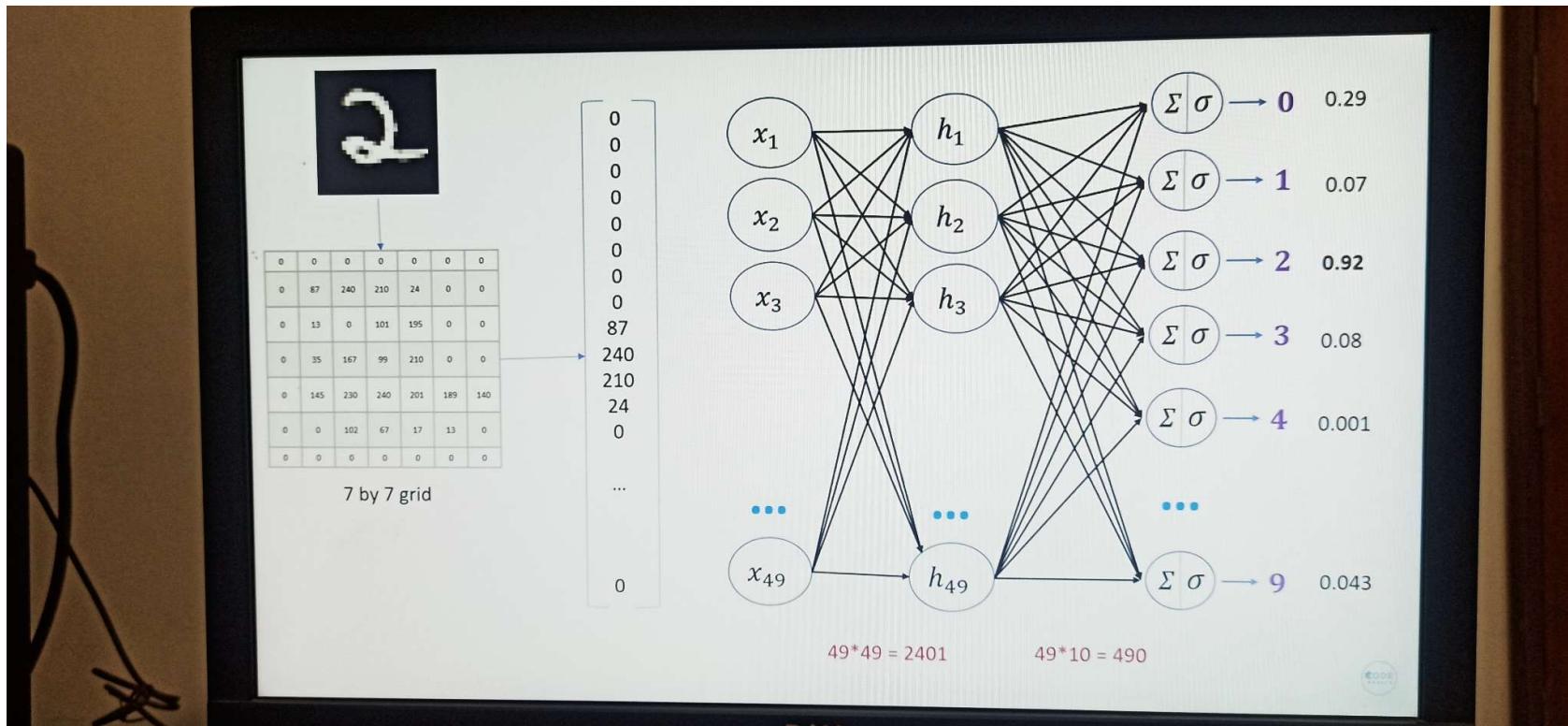
Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = $6 \text{ mil} * 4 \text{ mil}$
= 24 million

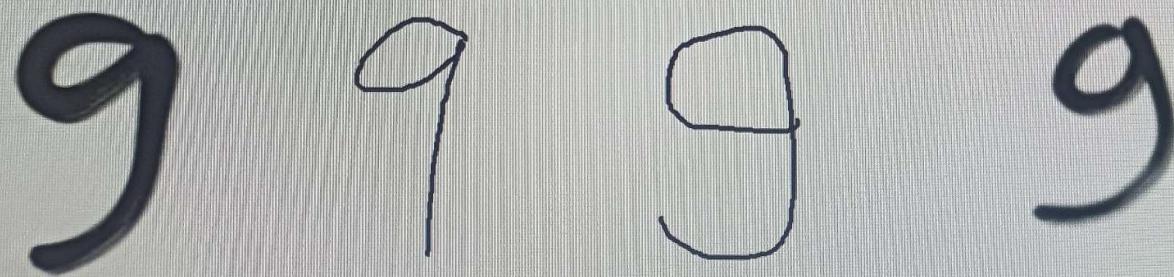


Disadvantages of using ANN for image classification

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



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



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
-1	1	-1	-1	-1

Location shifted

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



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

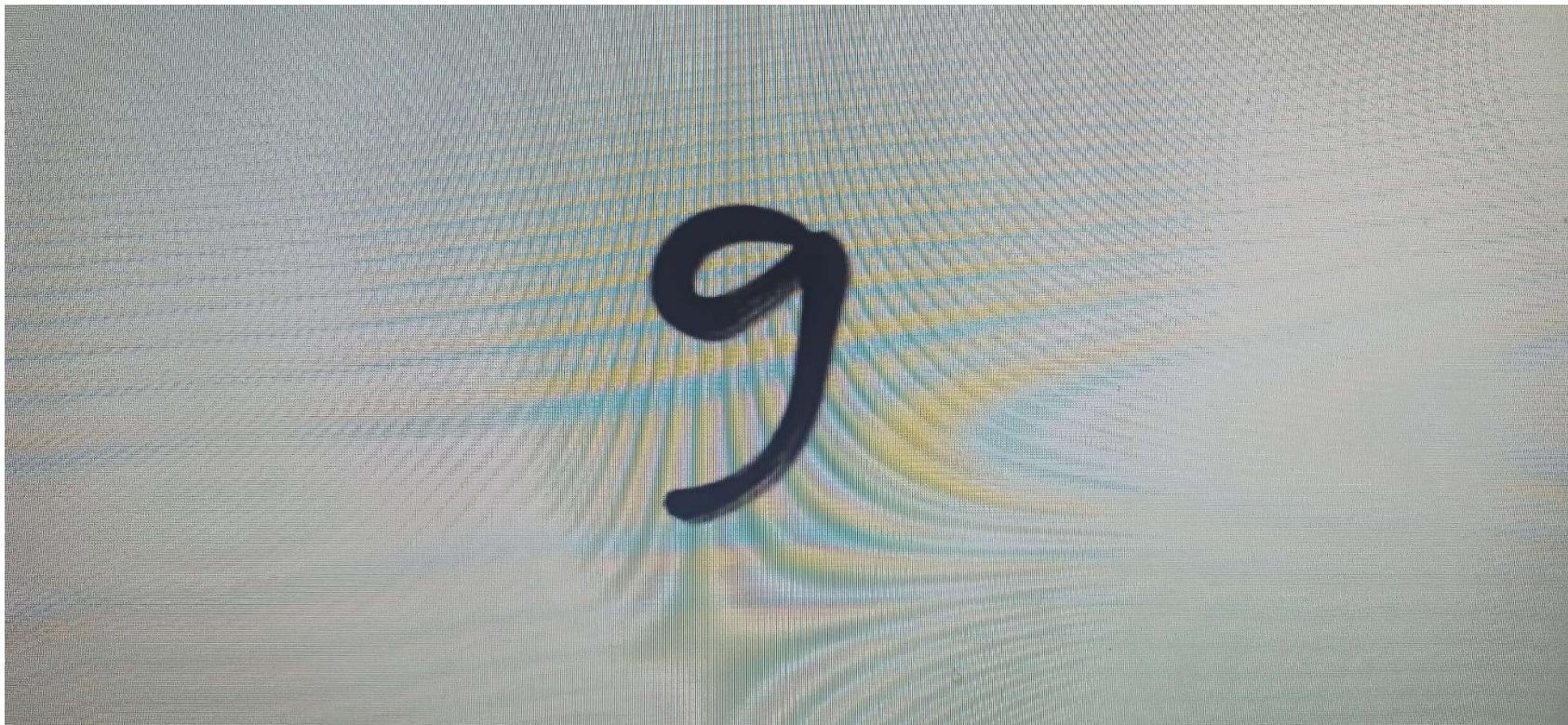


-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	-1	1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

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

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

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



Convolution Neural Network (CNN)