KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

AN AUTONOMOUS INSTITUTION - ACCREDITED BY NAAC WITH 'A' GRADE

Narayanaguda, Hyderabad.

Deep Learning

Introduction to CNN

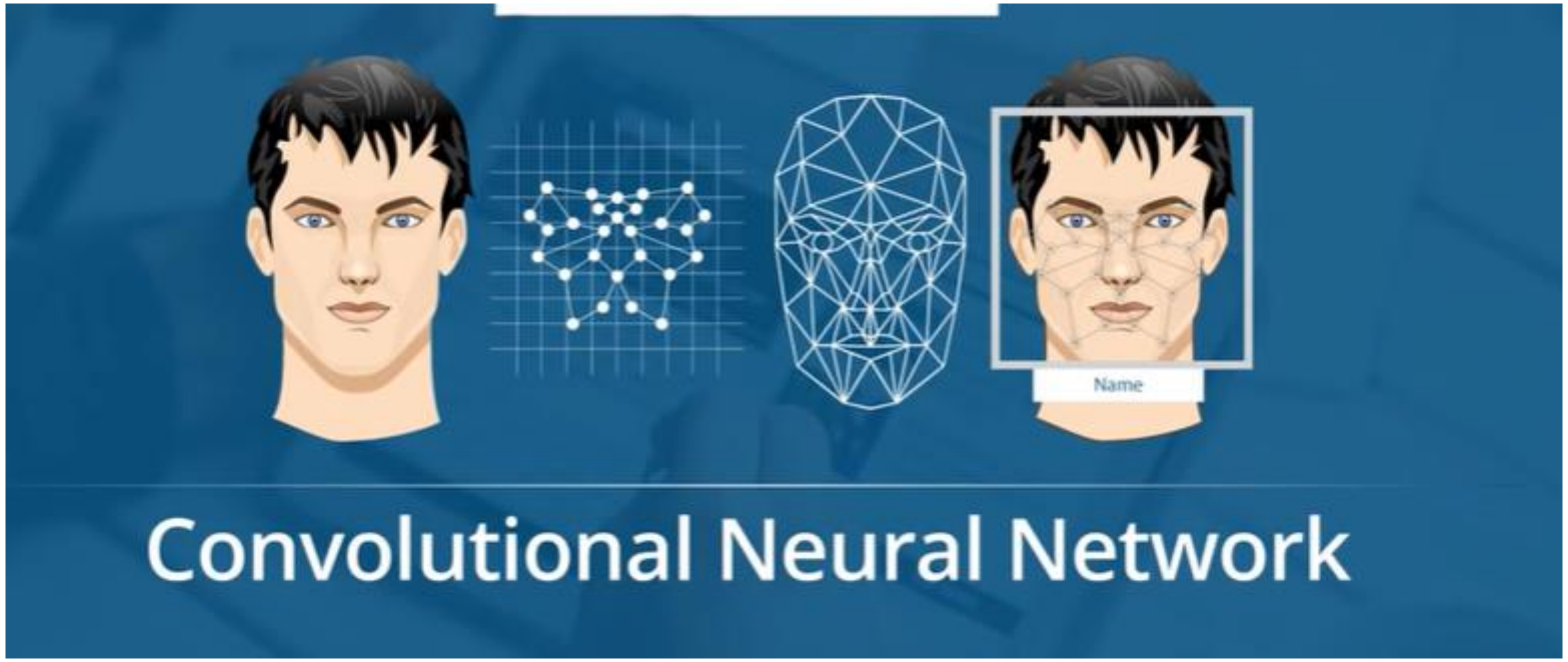
SESSION1

18-11-2024

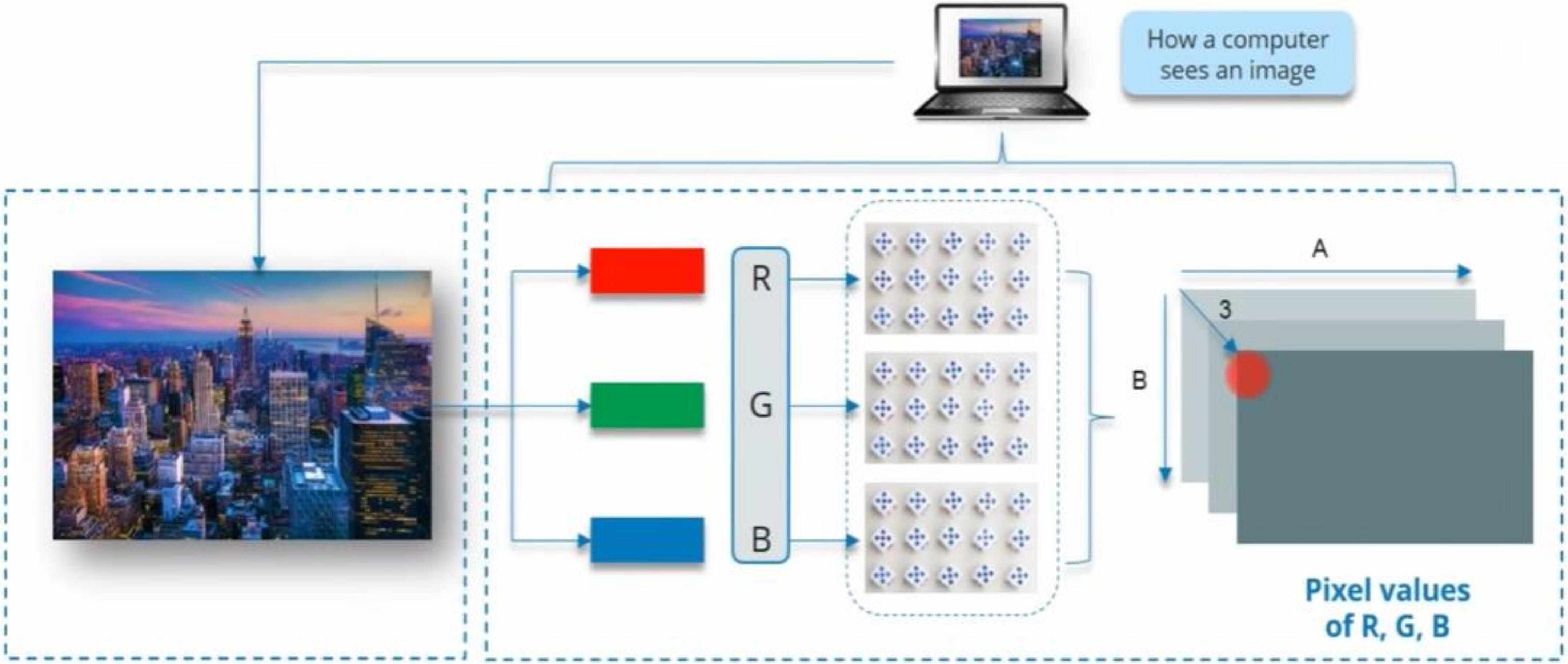
BY

ASHA

INTRODUCTION TO CNN

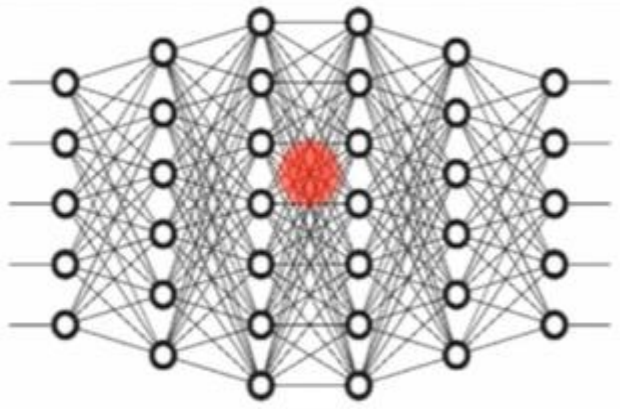


How a Computer Reads an Image



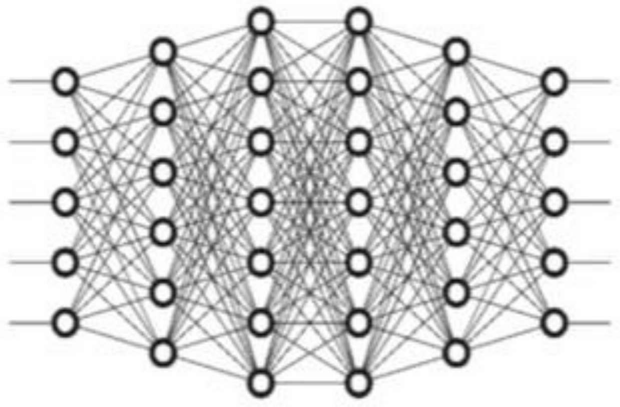
Why Not Fully Connected Networks

Image with 28 x 28 x 3 pixels



Number of weights in the first hidden layer will be 2352

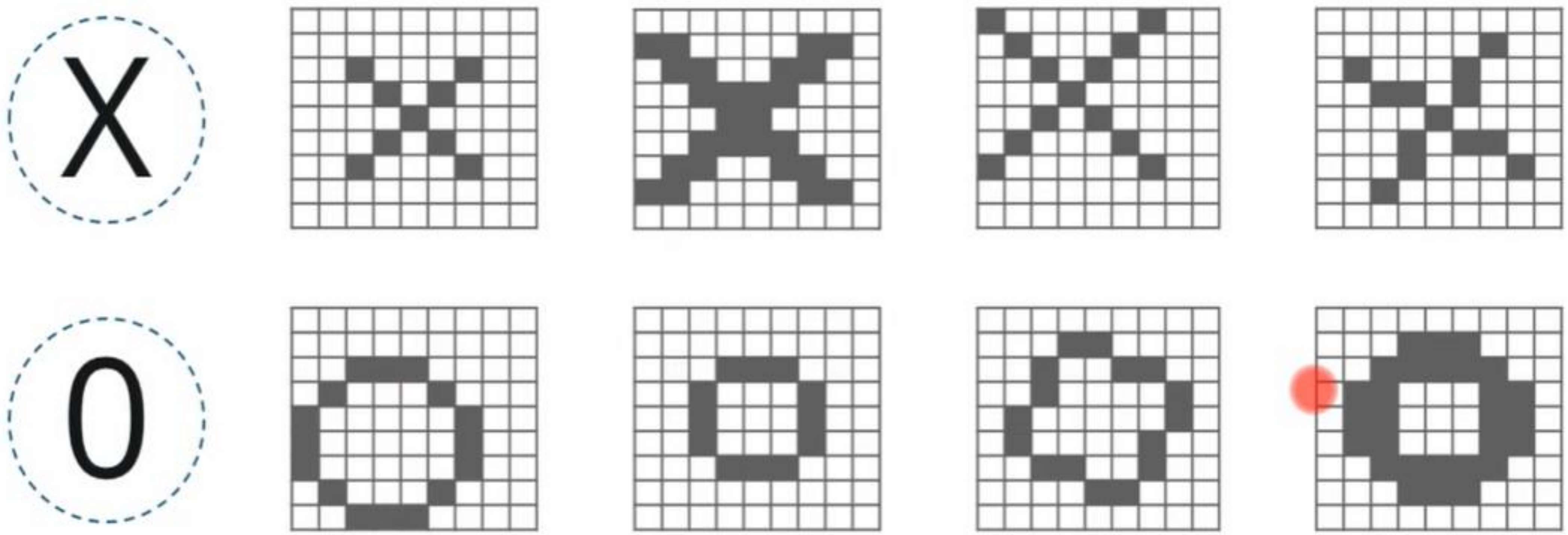
Image with 200 x 200 x 3 pixels



Number of weights in the first hidden layer will be 120,000

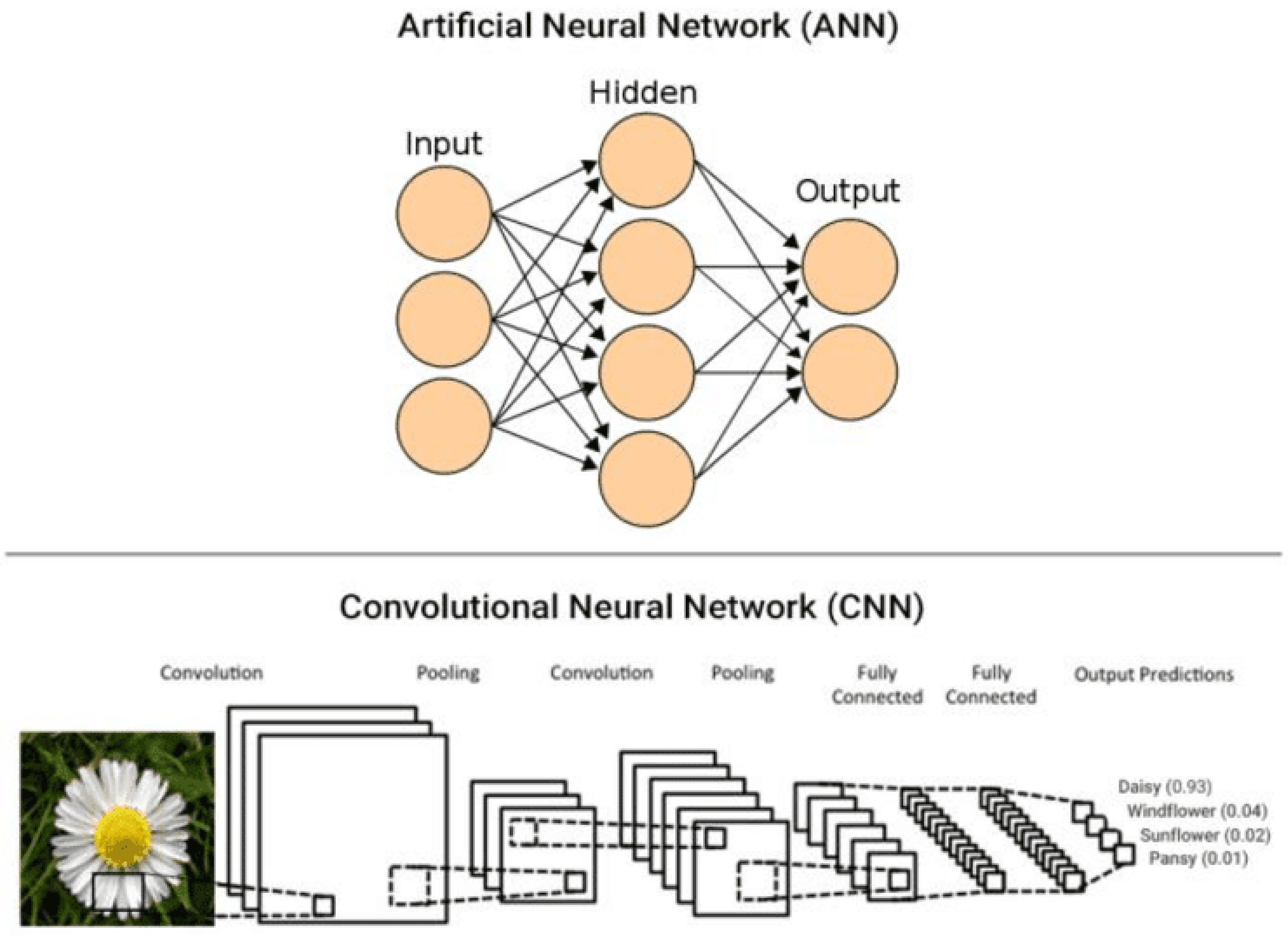
Trickier Case

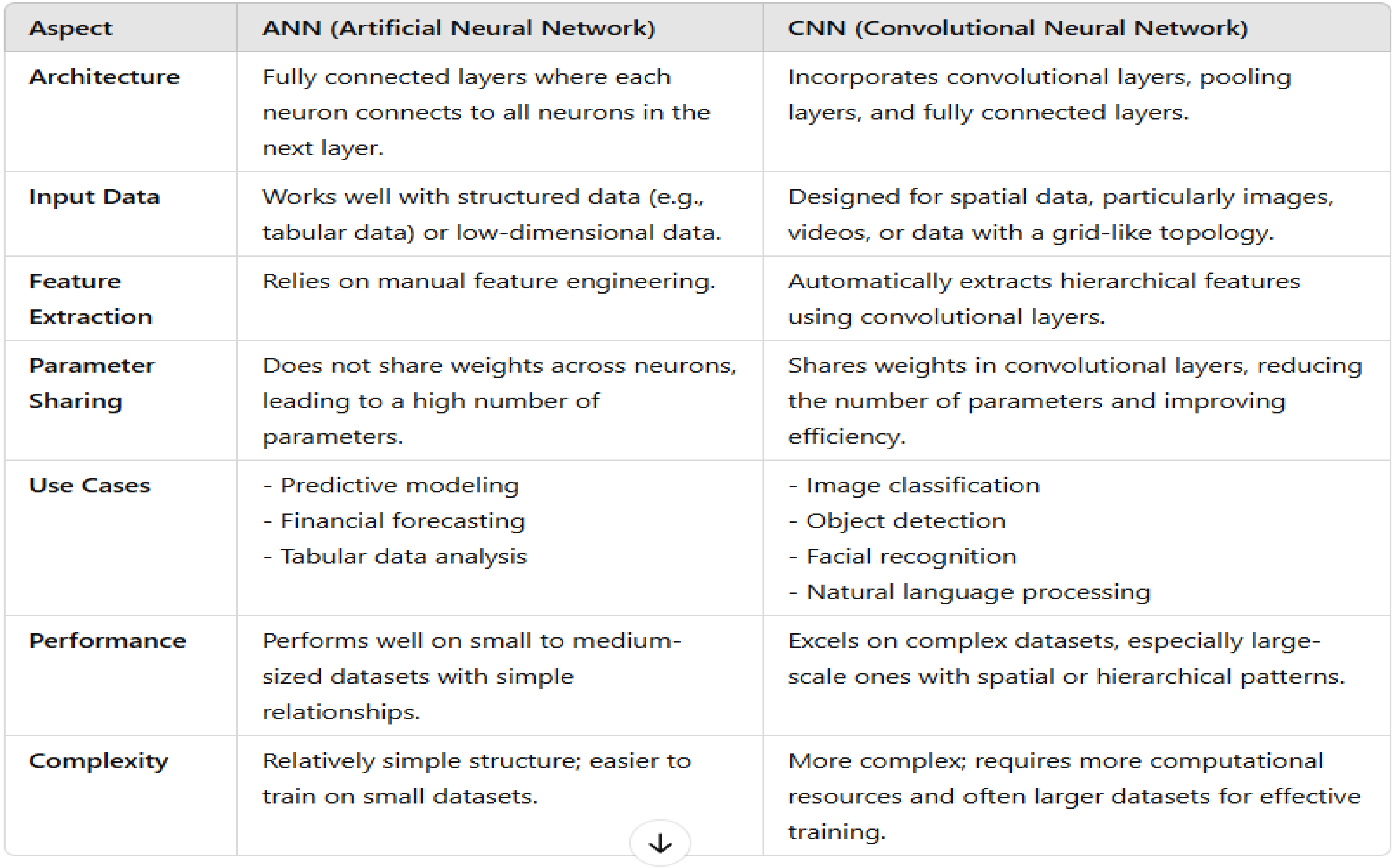
Here, we will have some problems, because X and O images won't always have the same images.
There can be certain deformations. Consider the diagrams shown below:



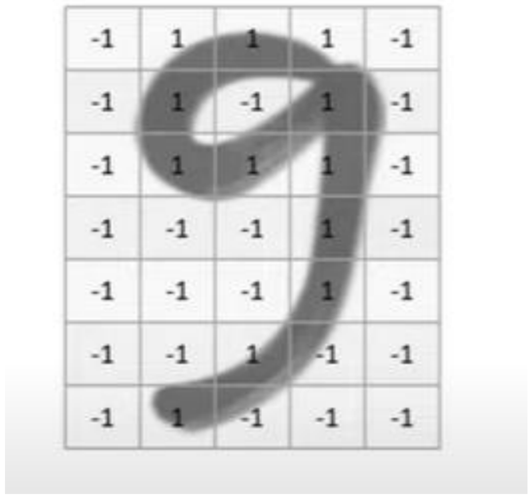
- A **convolutional neural network (CNN),** is a network architecture for deep learning which learns directly from data.
- CNNs are particularly useful for finding patterns in images to recognize objects.
- They can also be quite effective for classifying non-image data such as audio, time series, and signal data.

Convolutional Neural Networks vs. Artificial Neural Networks

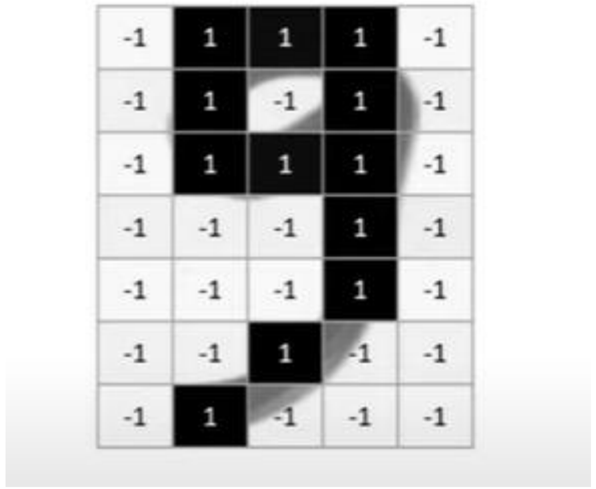




Aspect	ANN (Artificial Neural Network)	CNN (Convolutional Neural Network)
Architecture	Fully connected layers where each neuron connects to all neurons in the next layer.	Incorporates convolutional layers, pooling layers, and fully connected layers.
Input Data	Works well with structured data (e.g., tabular data) or low-dimensional data.	Designed for spatial data, particularly images, videos, or data with a grid-like topology.
Feature Extraction	Relies on manual feature engineering.	Automatically extracts hierarchical features using convolutional layers.
Parameter Sharing	Does not share weights across neurons, leading to a high number of parameters.	Shares weights in convolutional layers, reducing the number of parameters and improving efficiency.
Use Cases	- Predictive modeling - Financial forecasting - Tabular data analysis	- Image classification - Object detection - Facial recognition - Natural language processing
Performance	Performs well on small to medium-sized datasets with simple relationships.	Excels on complex datasets, especially large-scale ones with spatial or hierarchical patterns.
Complexity	Relatively simple structure; easier to train on small datasets.	More complex; requires more computational resources and often larger datasets for effective training.

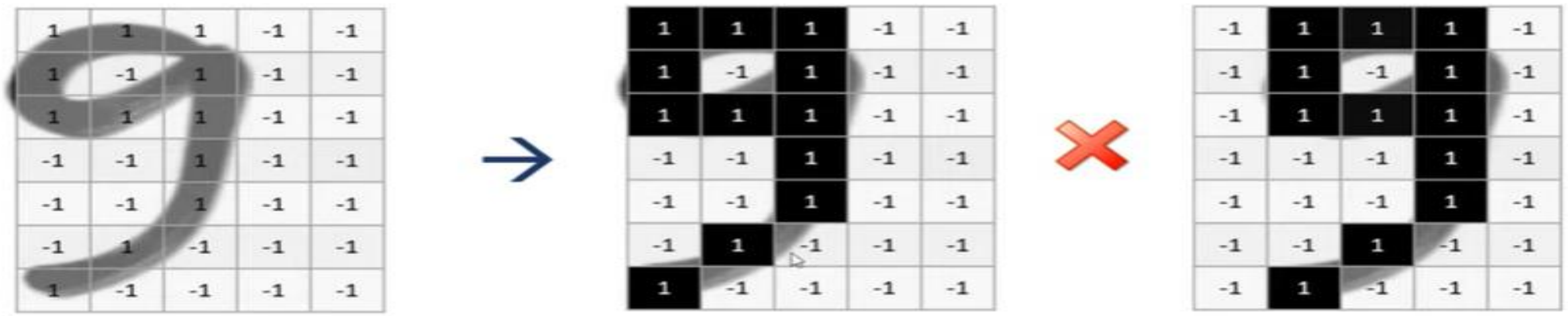


-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1



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

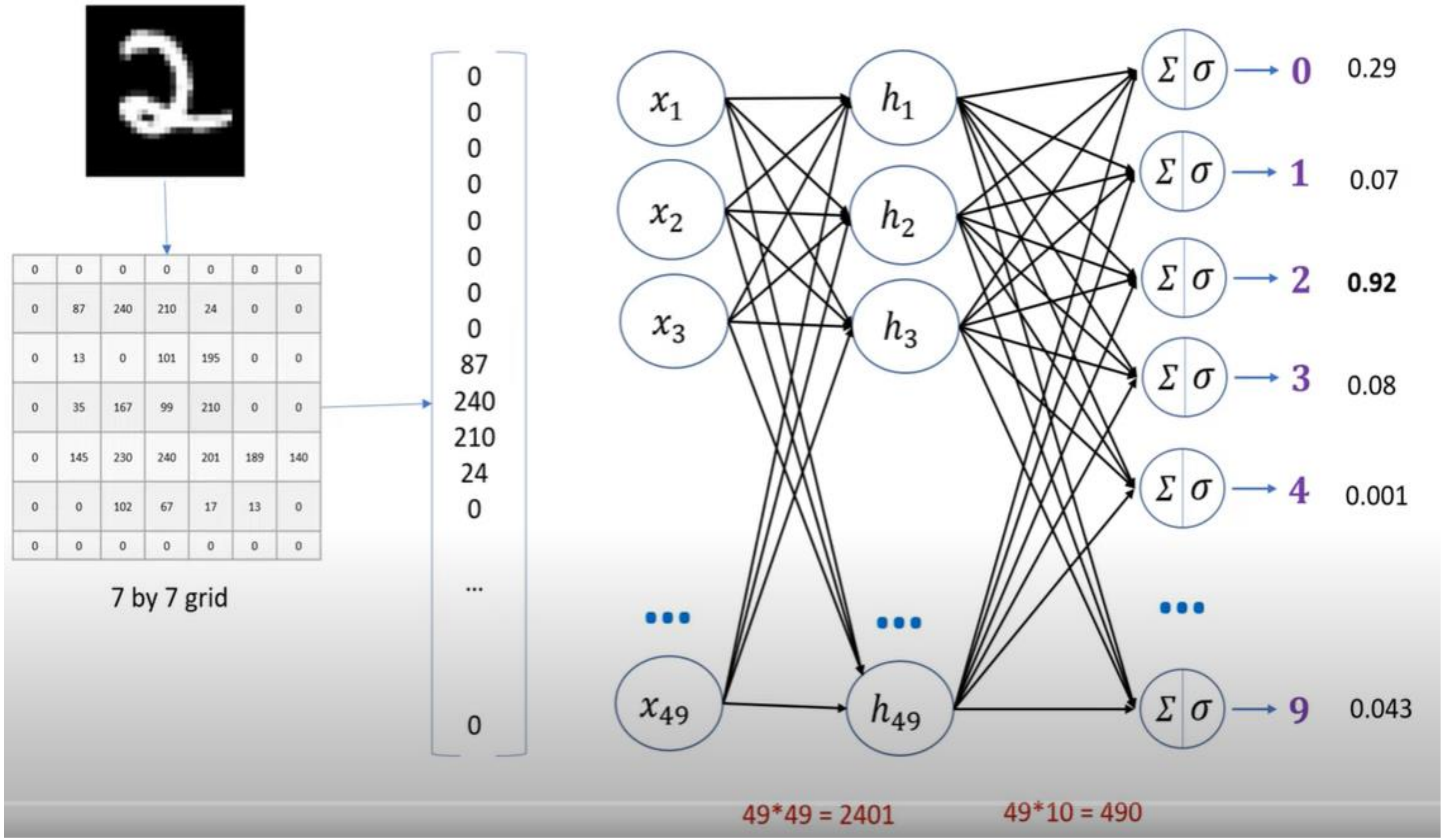
$\rightarrow$

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

✗

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

Image size = 1920 x 1080 X 3

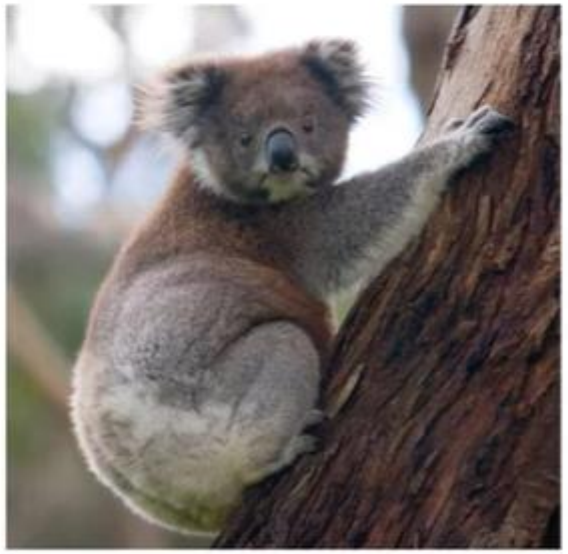
First layer neurons = 1920 x 1080 X 3 ~ 6 million

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

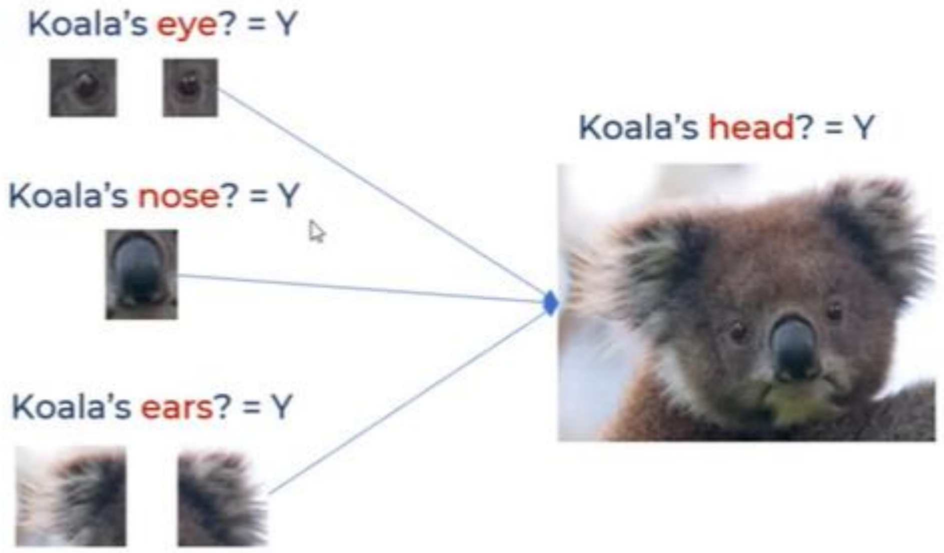
1. Too much computation
2. Treats local pixels same as pixels far apart
3. Sensitive to location of an object in an image

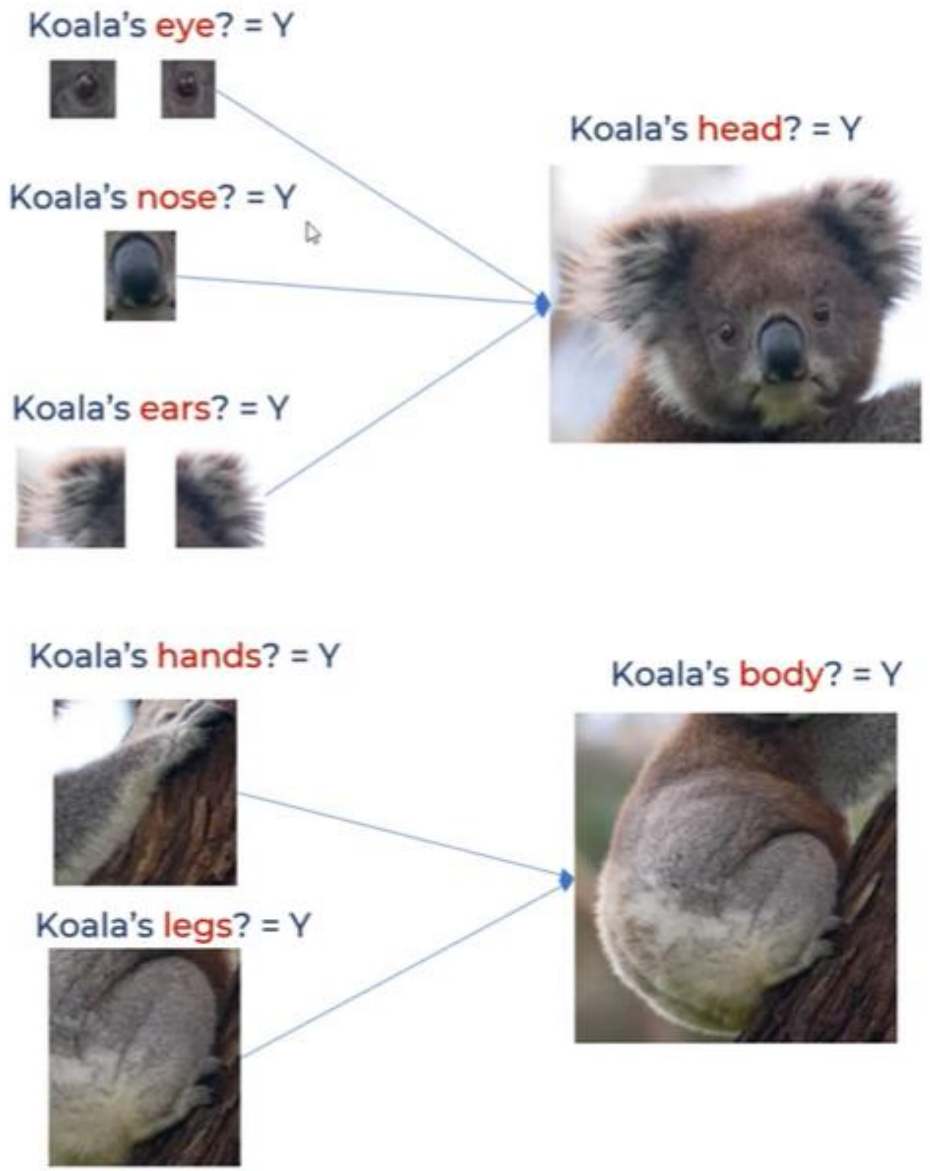


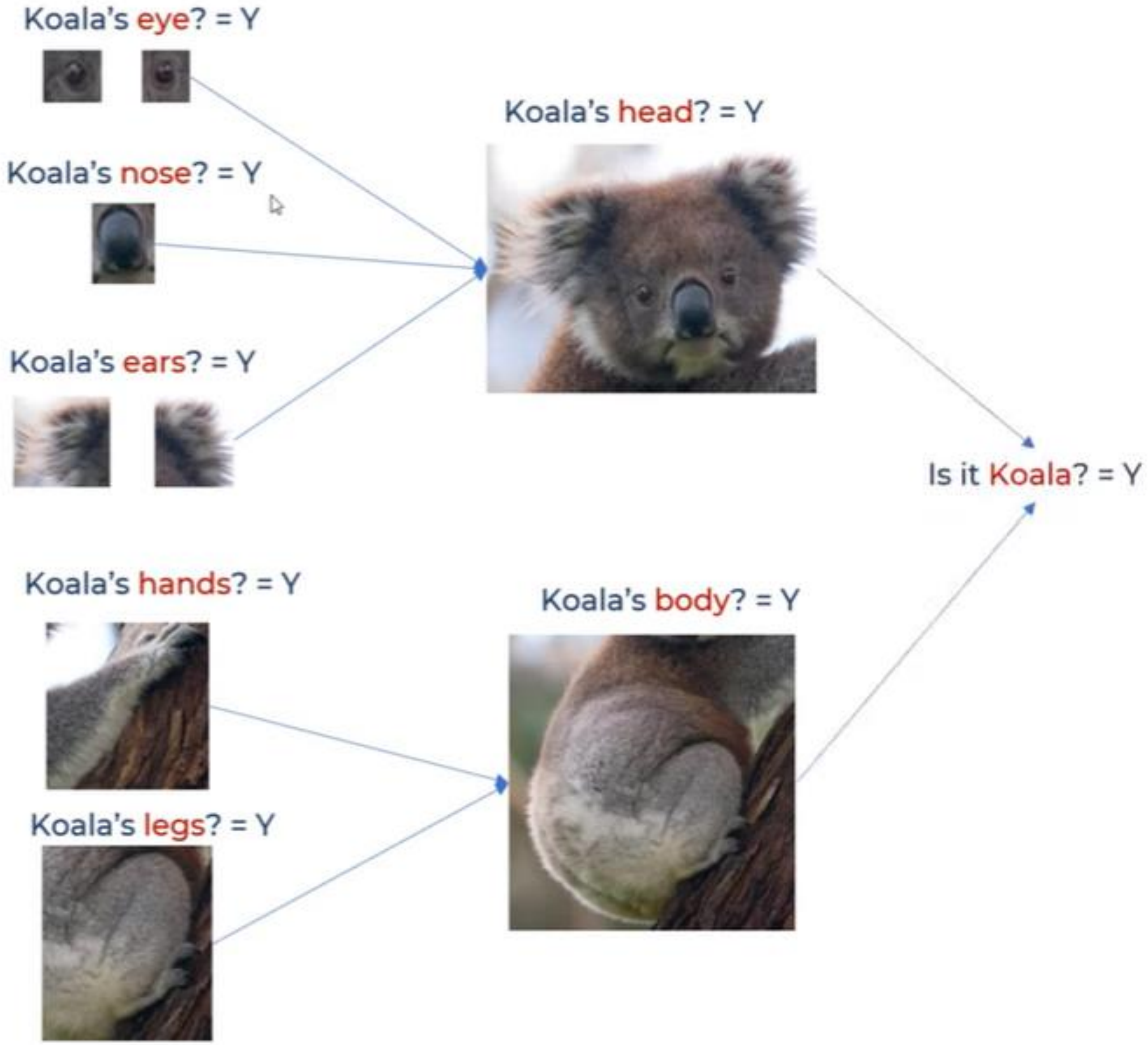
Koala's eye? = Y

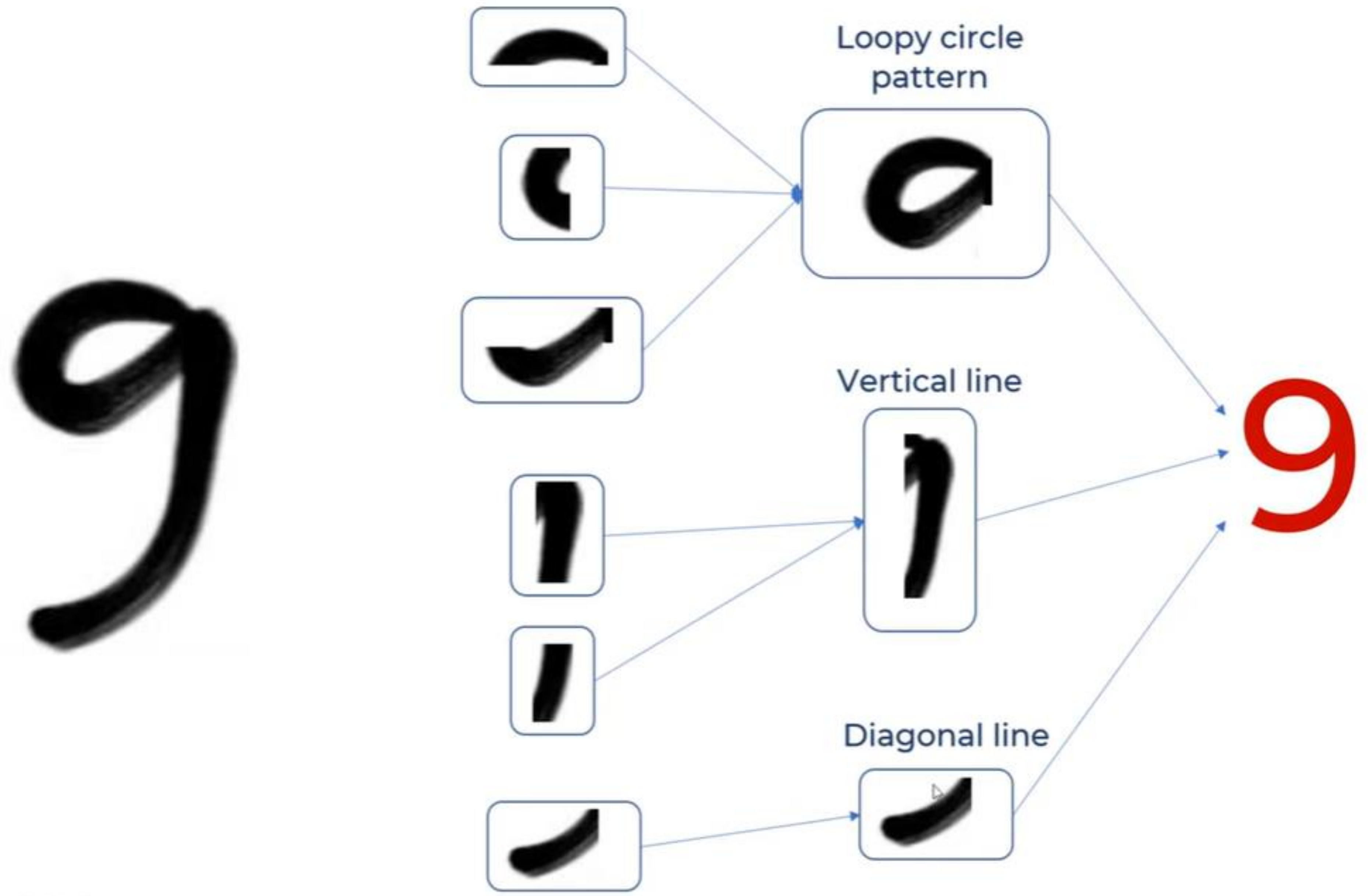
Koala's nose? = Y

Koala's ears? = Y

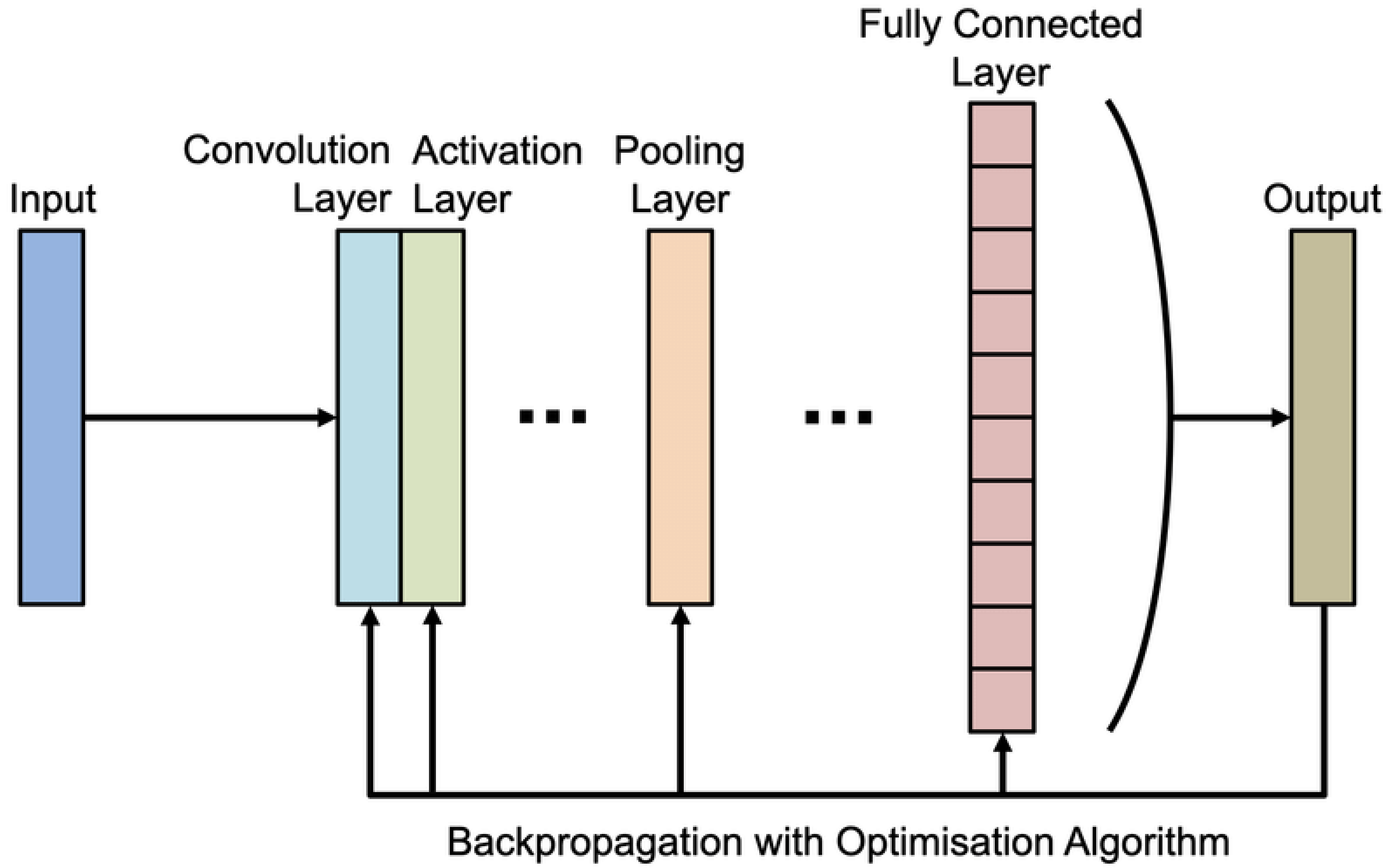

Koala's eye? = Y
Koala's nose? = Y
Koala's ears? = Y
Koala's head? = Y

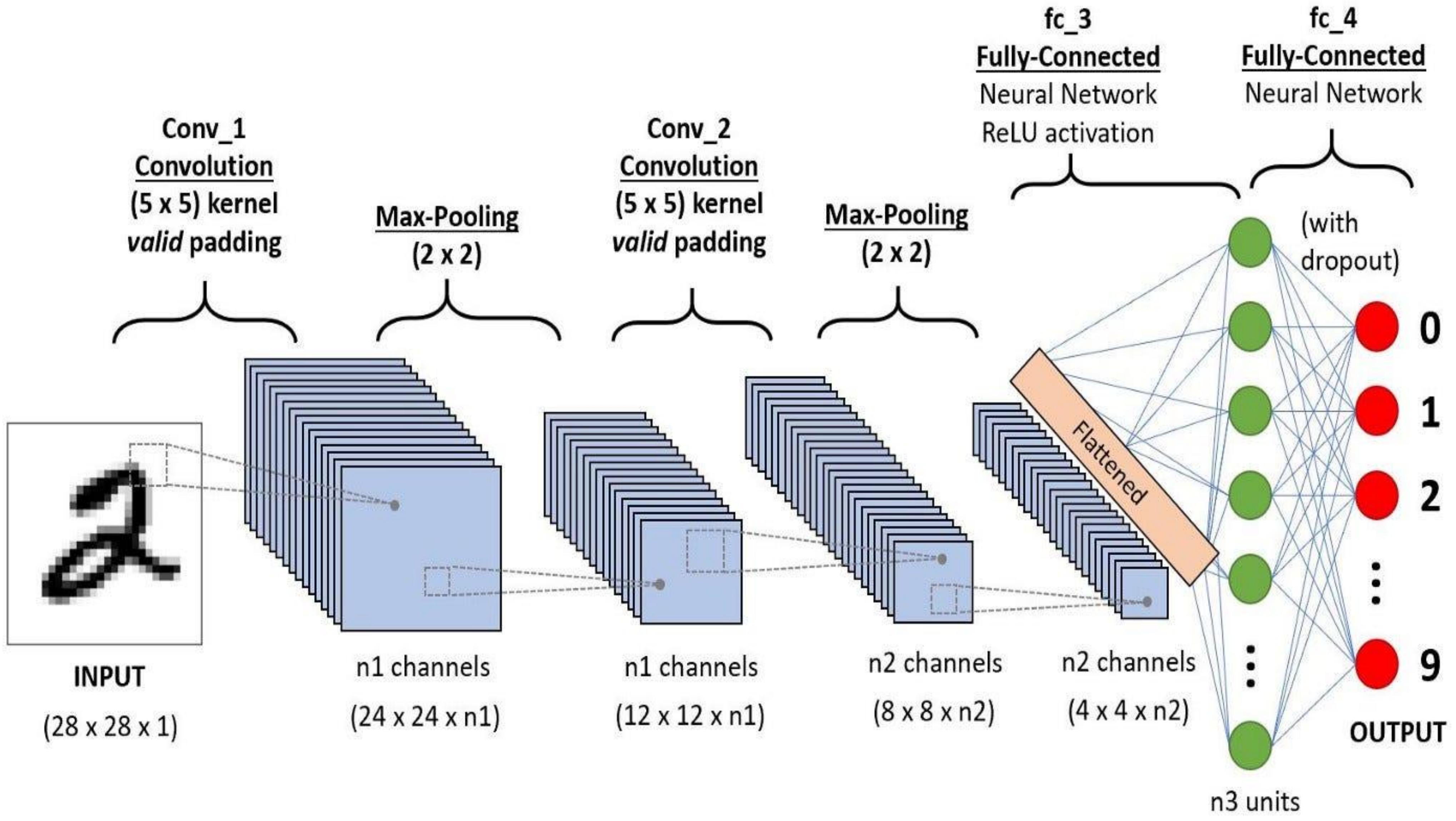

Koala's eye? = Y
Koala's nose? = Y
Koala's ears? = Y
Koala's head? = Y
Koala's hands? = Y
Koala's legs? = Y
Koala's body? = Y


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


Loopy circle pattern
Vertical line
Diagonal line
9

The standard architecture of a CNN is composed of a series of layers, each serving a specific purpose.



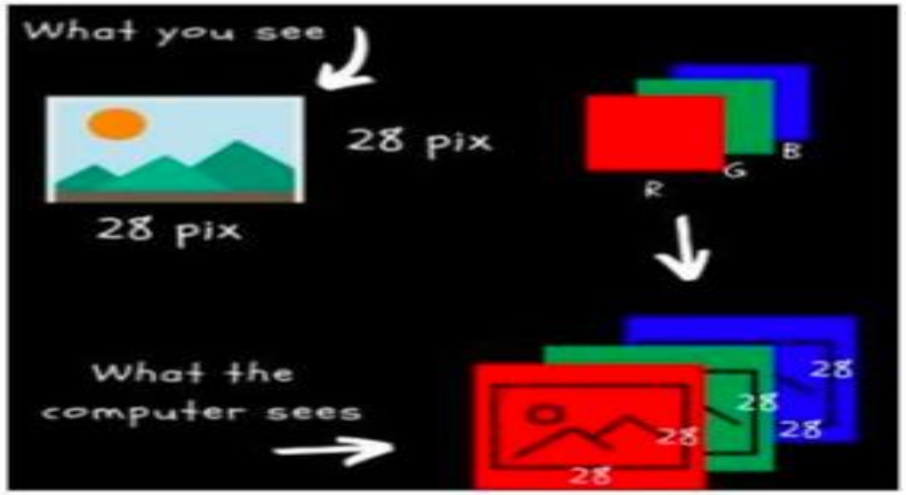

Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)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

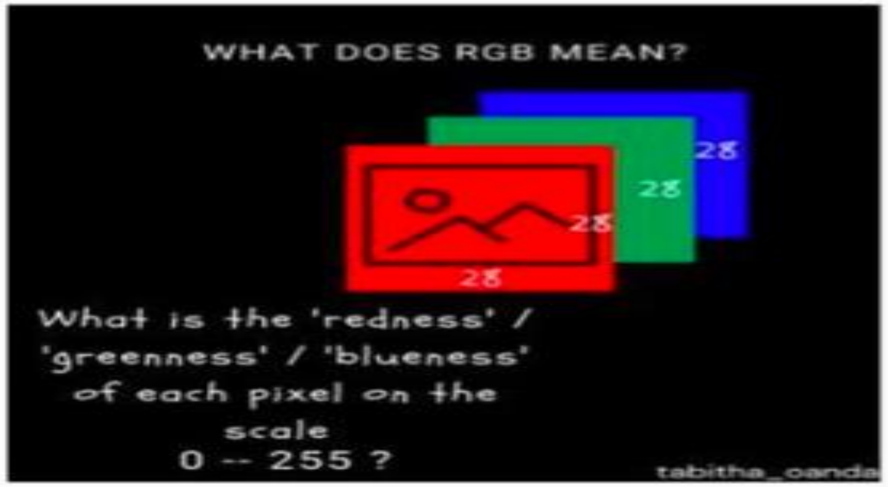
Basic Layers in a CNN Architecture

1. Input Layer
2. Convolutional Layer
3. Activation Layer (e.g., ReLU)
4. Pooling Layer (e.g., Max Pooling)
5. Fully Connected (Dense) Layer
6. Output Layer

1.Input Layer:

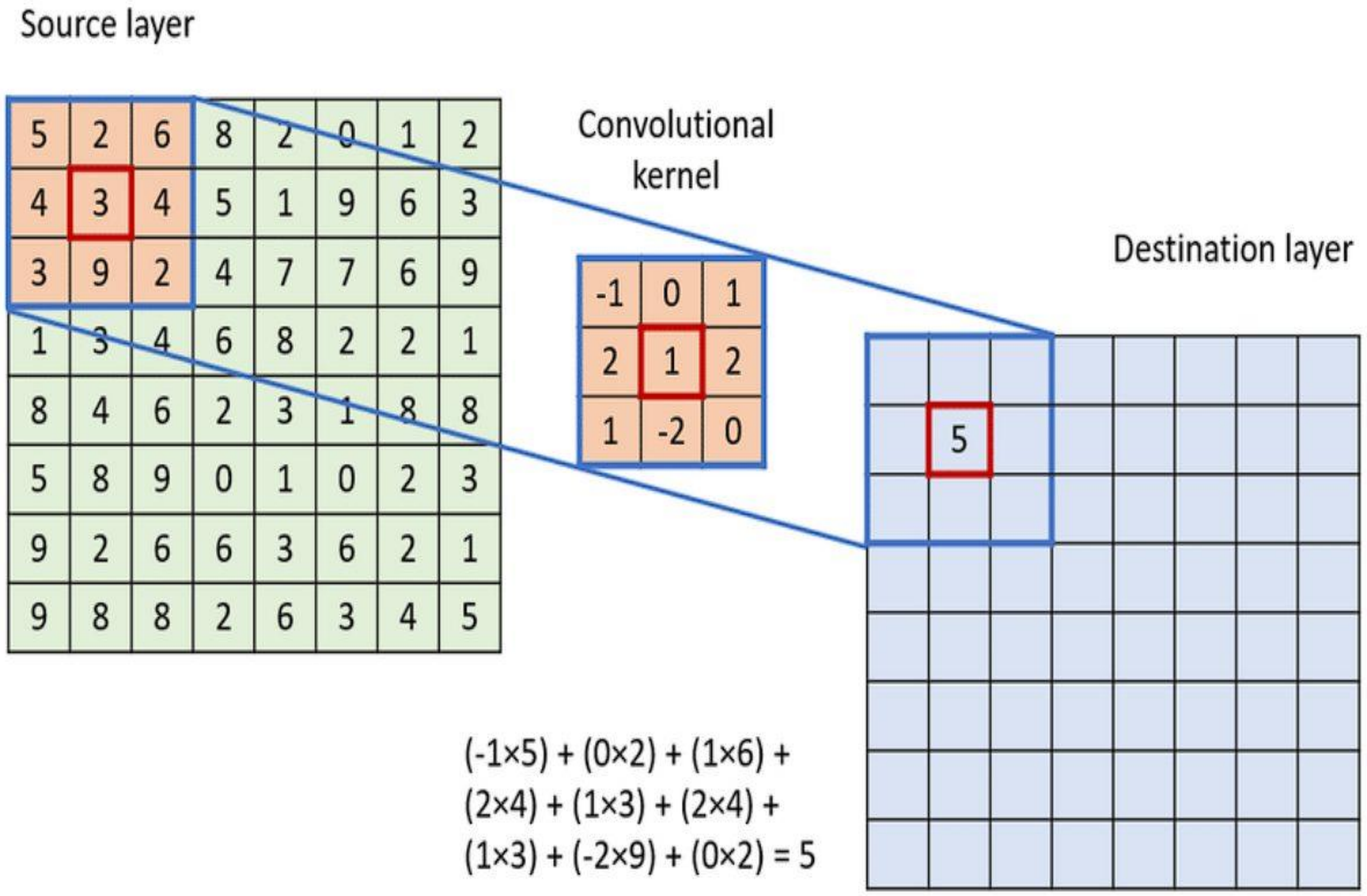
1. This layer holds the raw pixel values of the image, typically in the form of a 3D matrix (height x width x channels).
2. For instance, an RGB image with a size of 28x28 pixels would have an input shape of 28×28×3, where 3 represents the Red, Green, and Blue color channels.




WHAT DOES RGB MEAN?
28
28
28
28
What is the 'redness' / 'greenness' / 'blueness' of each pixel on the scale 0 -- 255 ?
tabitha_oanda

2. Convolutional Layer:

1. The core layer in CNNs, this layer applies a convolutional filters (or kernels) over the input image to detect patterns.
2. Each filter slides (convolves) over the image and computes a "dot product" between the filter and a small section of the input, generating a feature map.
3. Each filter is designed to detect specific patterns like edges, textures, or shapes.



(-1×5) + (0×2) + (1×6) +
(2×4) + (1×3) + (2×4) +
(1×3) + (-2×9) + (0×2) = 5

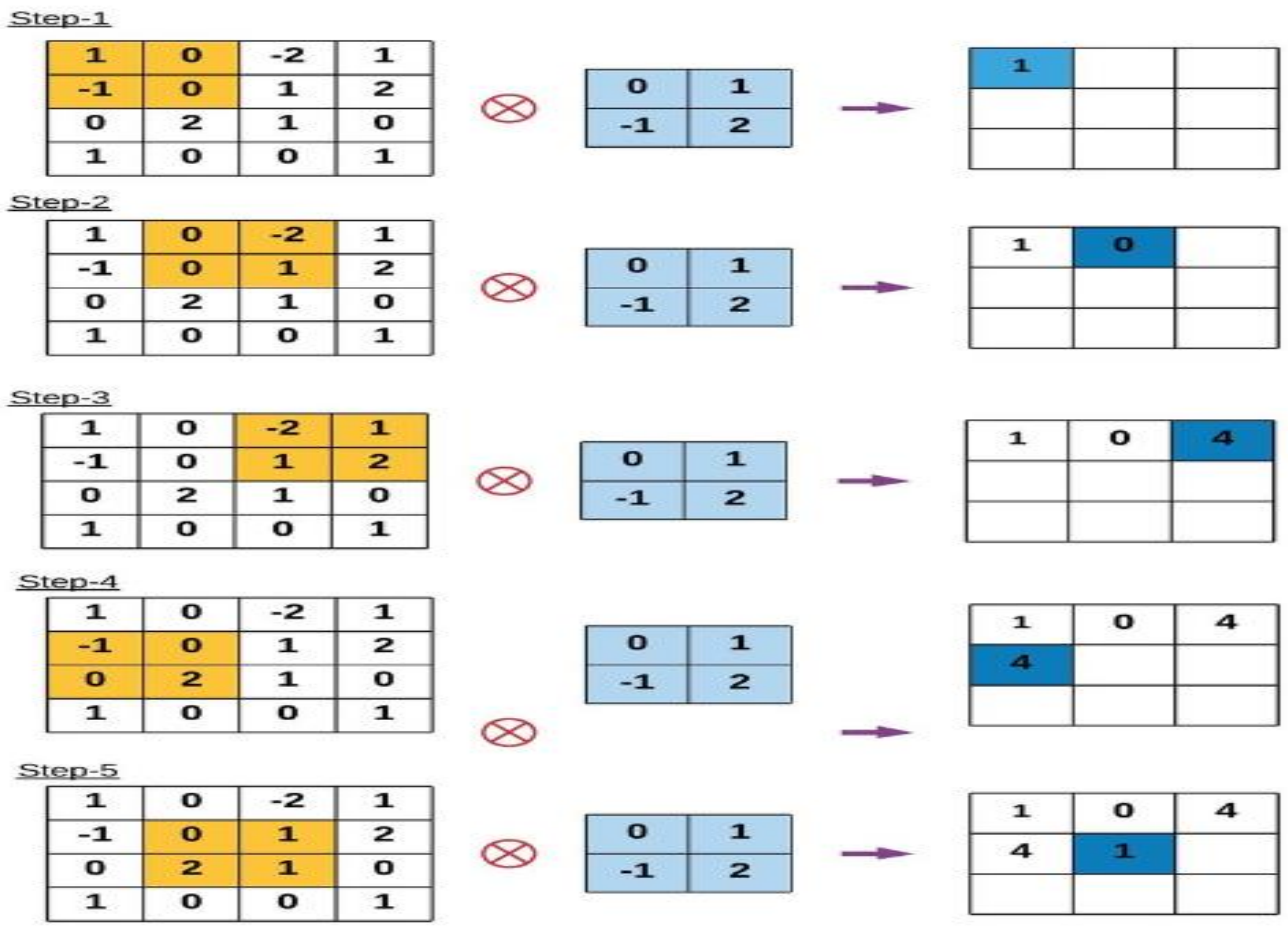
To apply the convolution:

•Overlay the Kernel on the Image: Start from the top-left corner of the image and place the kernel so that its center aligns with the current image pixel.

•Element-wise Multiplication: Multiply each element of the kernel with the corresponding element of the image it covers.

•Summation: Sum up all the products obtained from the element-wise multiplication. This sum forms a single pixel in the output feature map.

•Continue the Process: Slide the kernel over to the next pixel and repeat the process across the entire image.



Step-1

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

Step-2

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

Step-3

$$\begin{bmatrix} 1 & 0 & -2 & 1 \\ -1 & 0 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 4 \\ & & \\ & & \end{bmatrix}$$

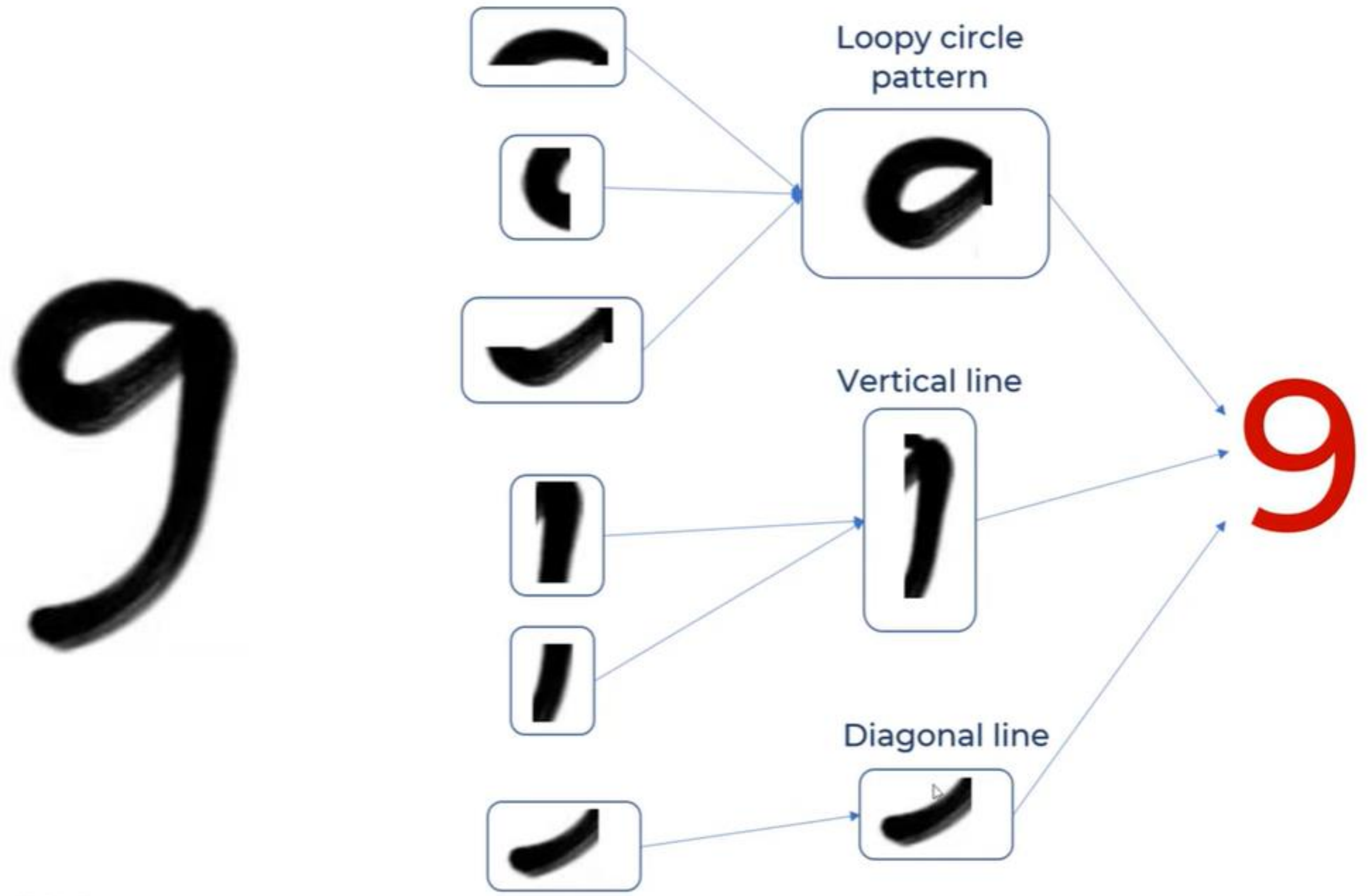
Step-4

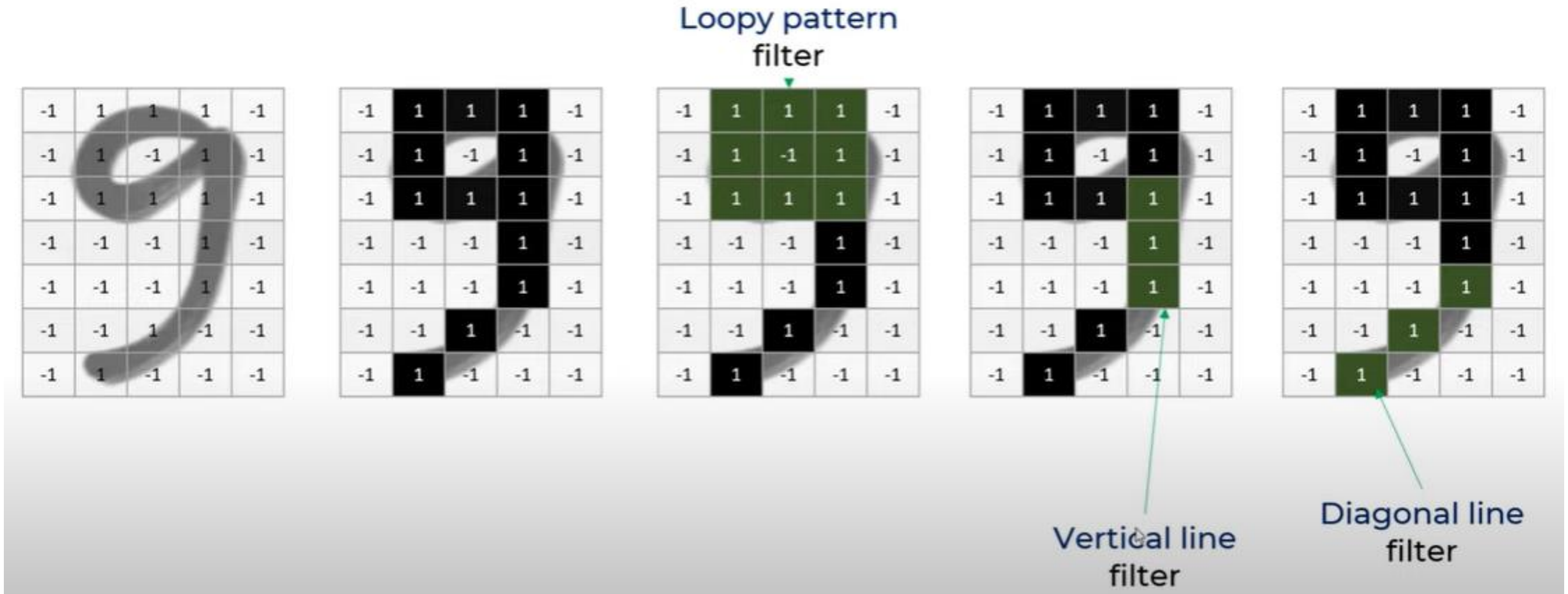
$$\begin{bmatrix} 1 & 0 & -2 & 1 \\ -1 & 0 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 4 \\ 4 & & \\ & & \end{bmatrix}$$

Step-5

$$\begin{bmatrix} 1 & 0 & -2 & 1 \\ -1 & 0 & 1 & 2 \\ 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 4 \\ 4 & 1 & \\ & & \end{bmatrix}$$

1. **Hyperparameters**:
    1. **Filter Size (Kernel Size)**: Usually 3x3 or 5x5.
    2. **Stride**: Determines the step size as the filter slides over the input.
    3. **Padding**: Controls the spatial dimensions of the output (adding "zero-padding" can preserve input size).
2. **Output**: The output of a convolutional layer is a set of **feature maps**, each representing the presence of a specific feature across the spatial dimensions.


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

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

-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		

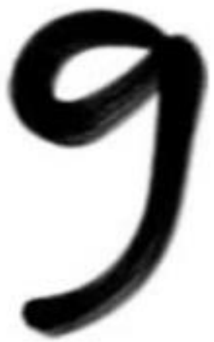
-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-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

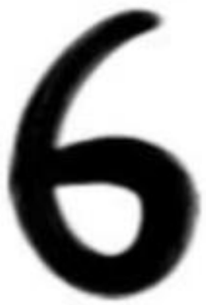
Feature Map



Loopy pattern detector

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

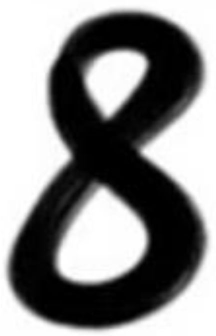
	1	



Loopy pattern detector

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

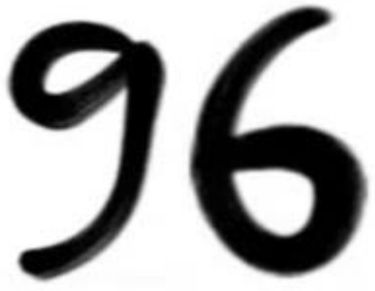
	1	



Loopy pattern detector

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

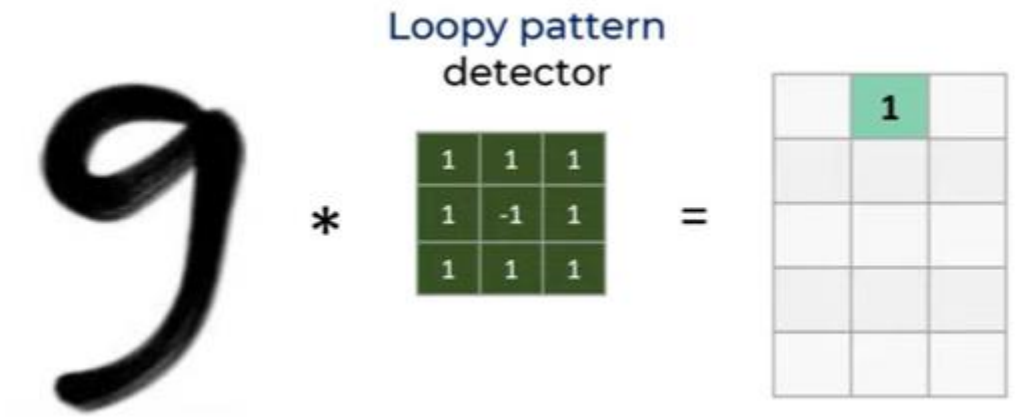
	1	
	1	

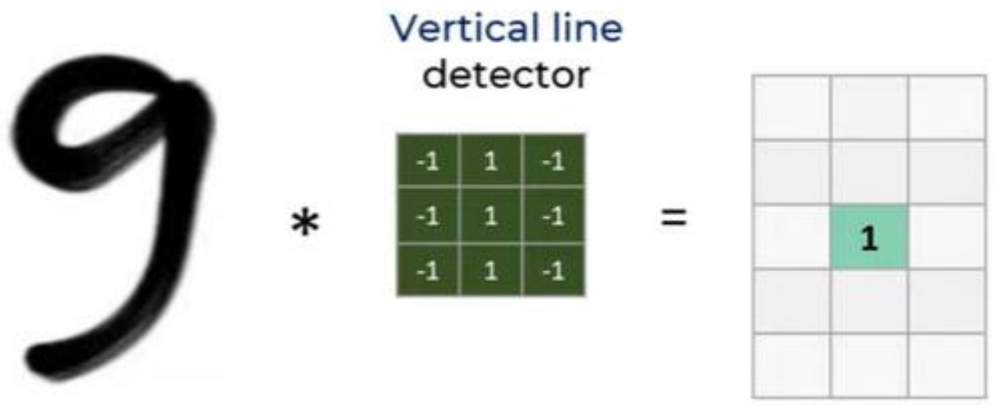


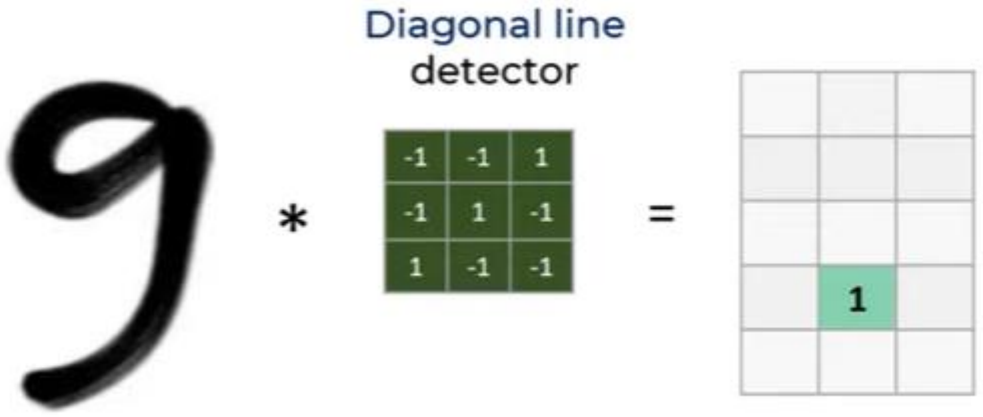
Loopy pattern detector

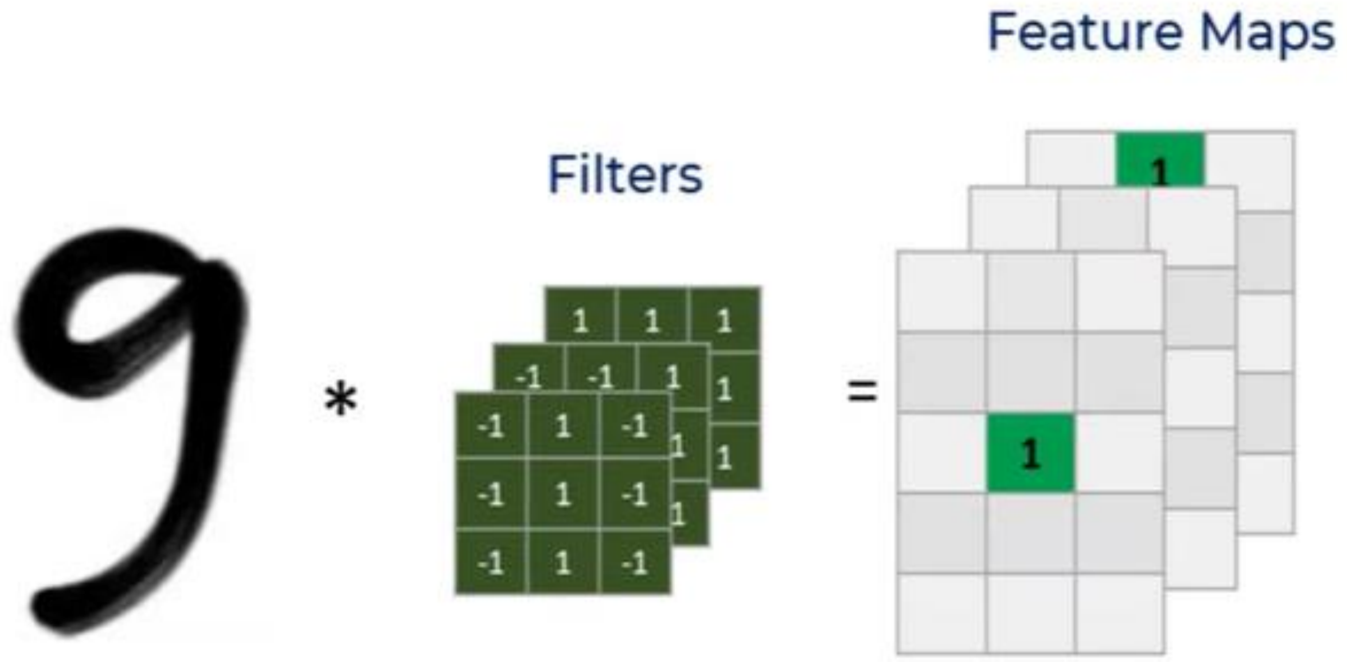
$$* \begin{bmatrix}1&1&1\\1&-1&1\\1&1&1\end{bmatrix} =$$

	1			
			1	


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

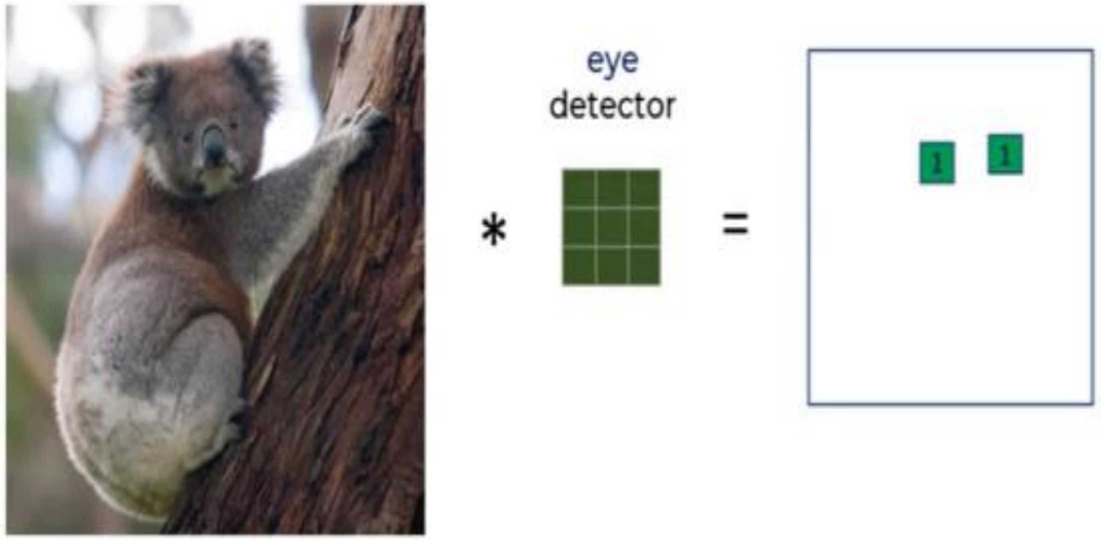

Vertical line detector
-1 1 -1
-1 1 -1
-1 1 -1
*
=
1

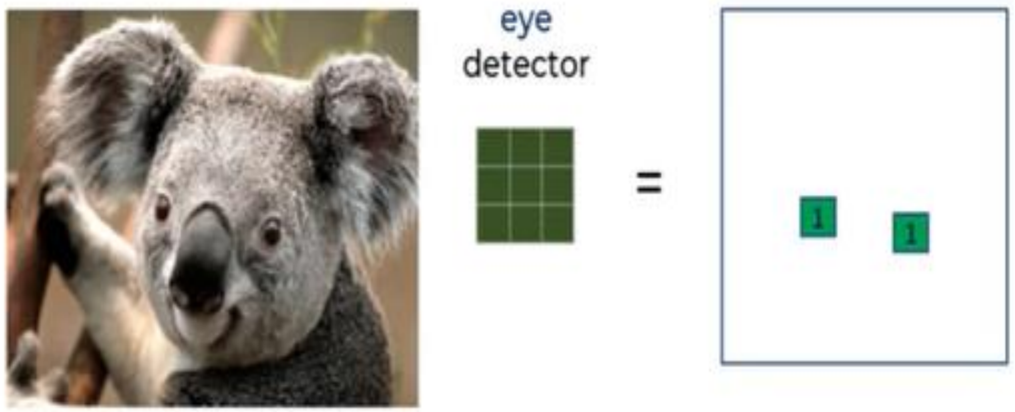

Diagonal line detector
-1 -1 1
-1 1 -1
1 -1 -1
*
=
1


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

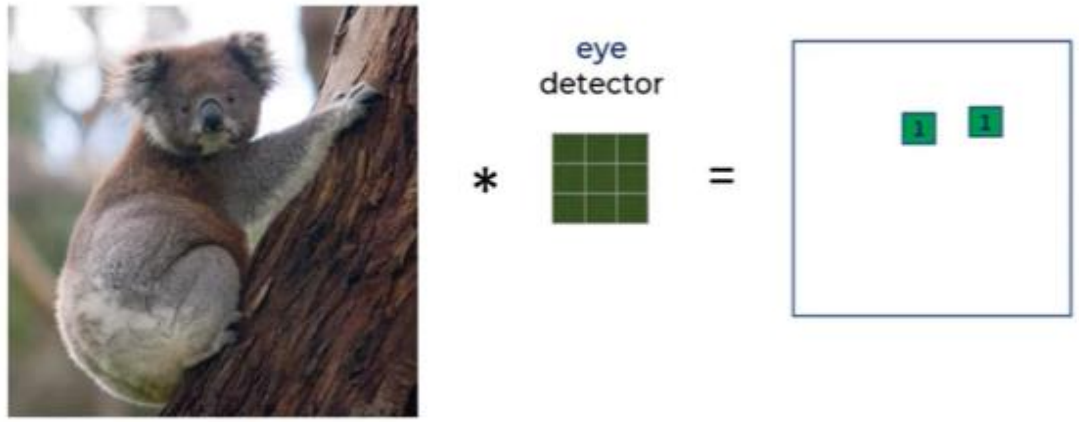
Filters are nothing but the feature detectors

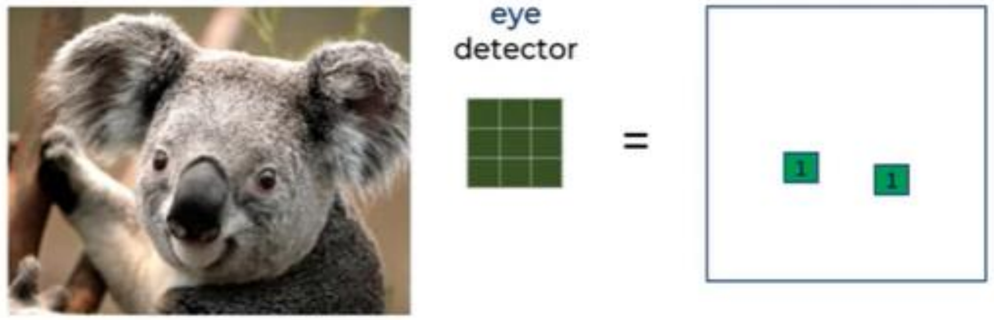
Location invariant: It can detect eyes in any location of the image

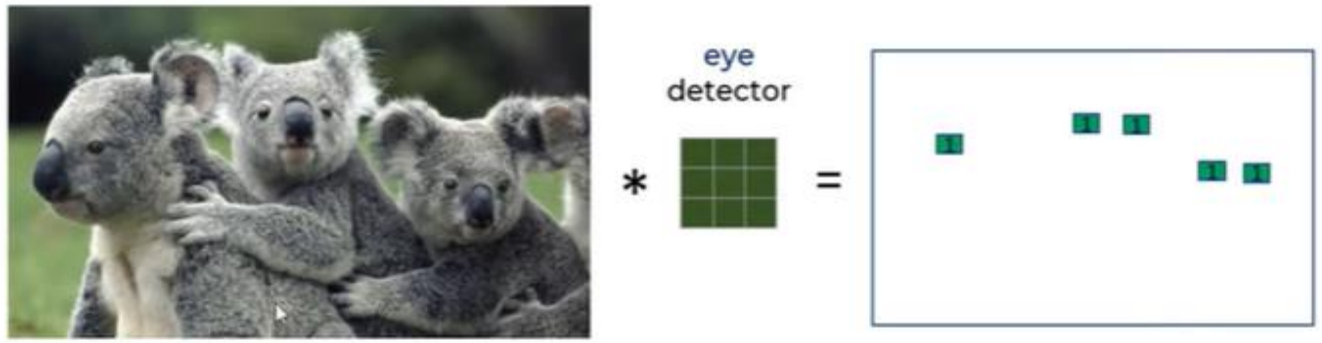


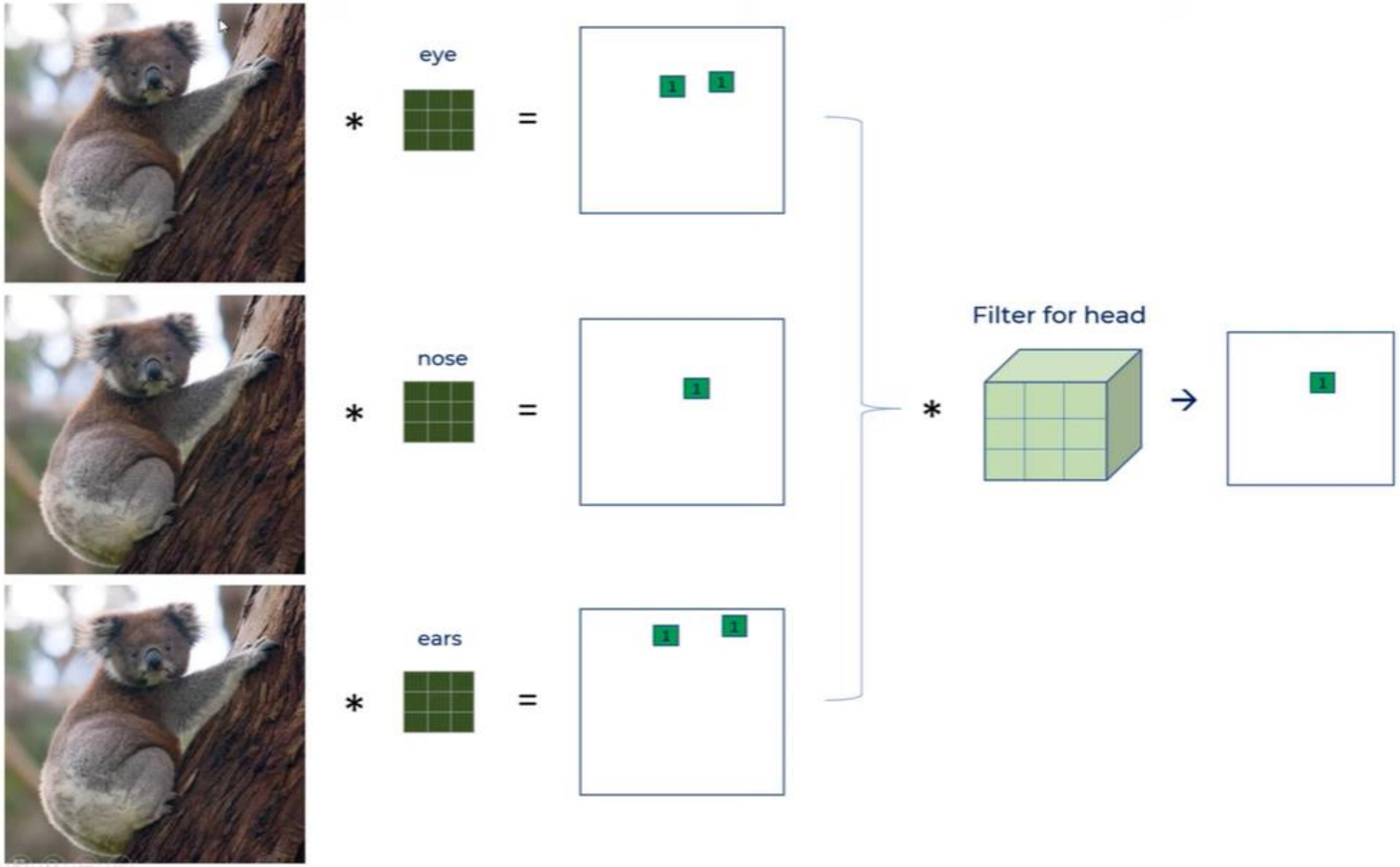


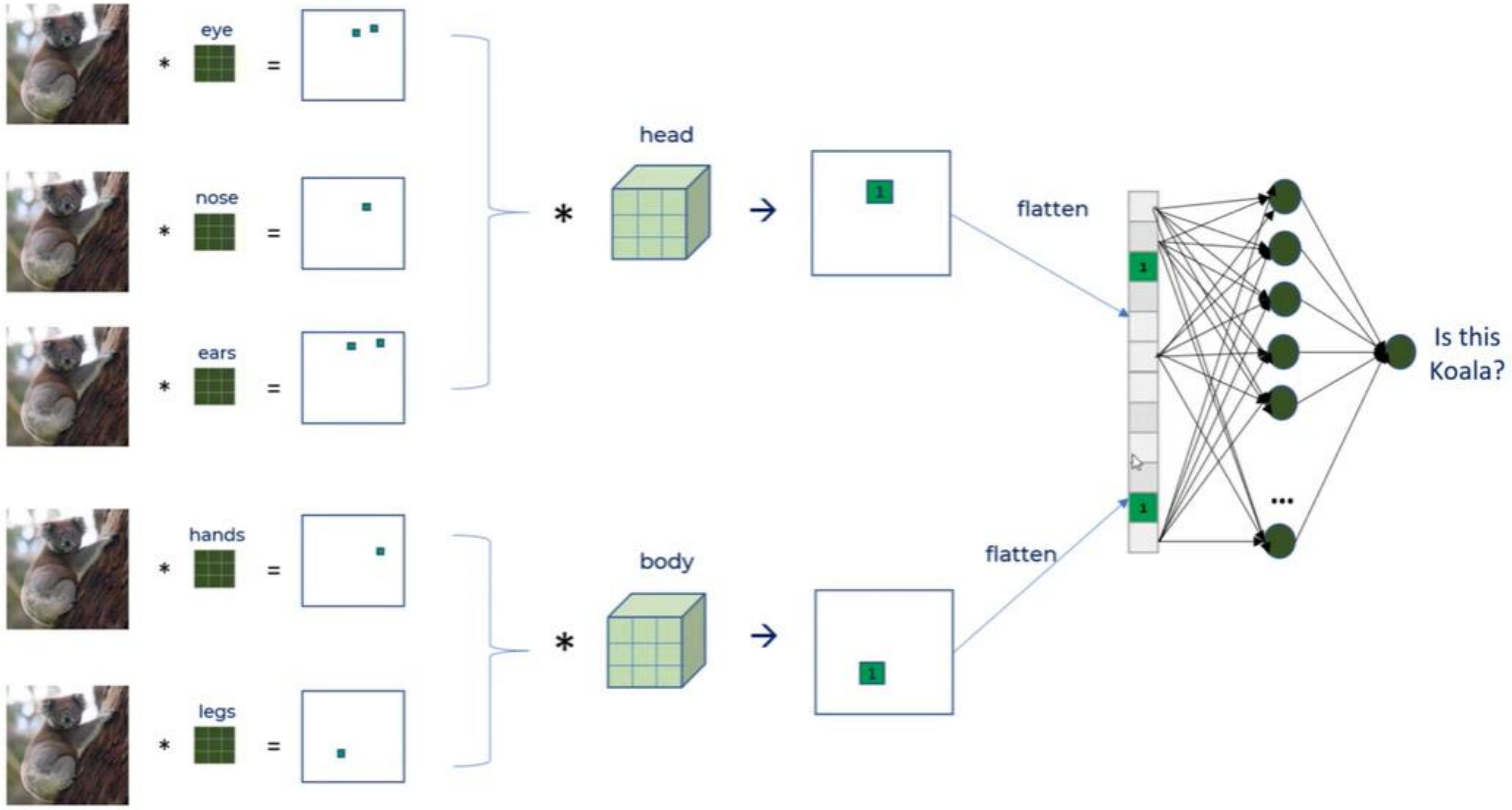
Location invariant: It can detect eyes in any location of the image

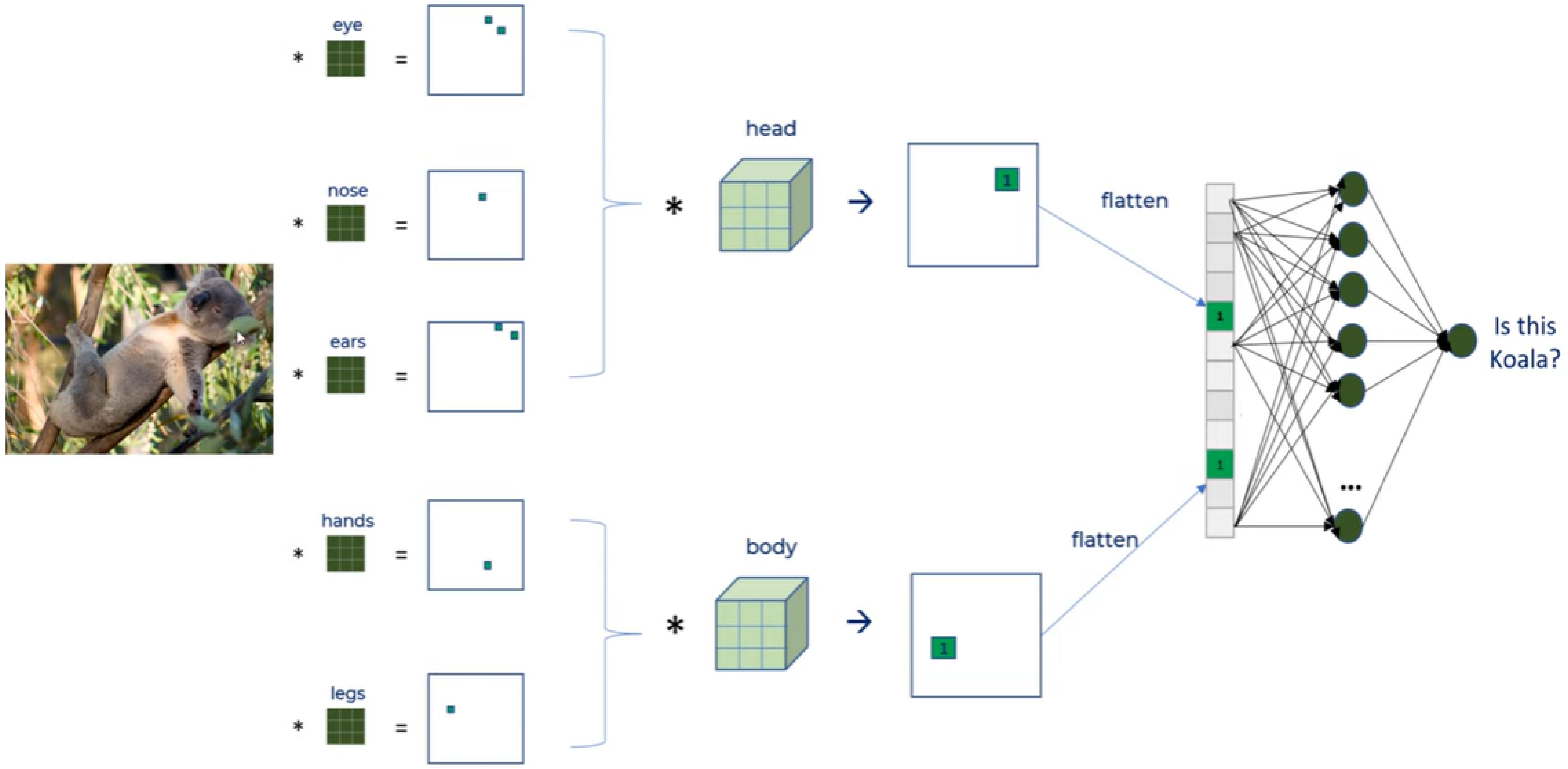




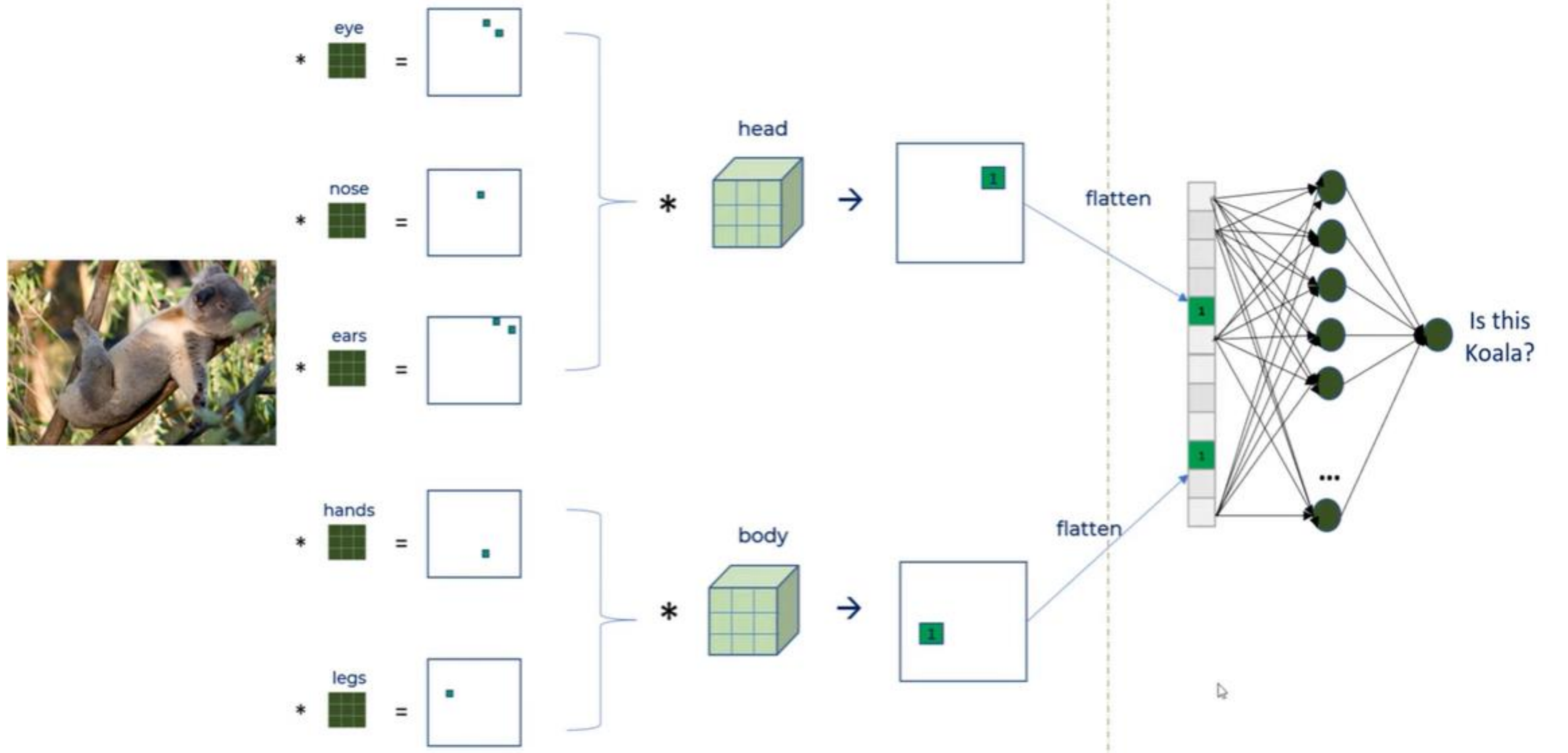



eye
nose
ears
Filter for head
1


eye
nose
ears
hands
legs
*
=
head
body
1
flatten
flatten
...
Is this Koala?



eye
nose
ears
hands
legs
head
body
flatten
flatten
1
1
1
1
...
Is this Koala?

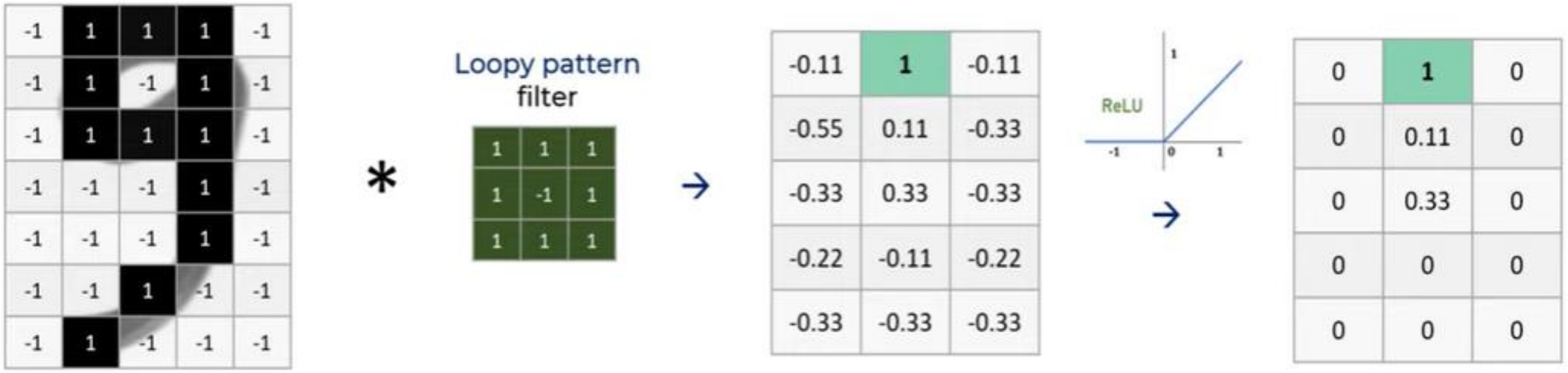


Feature Extraction

Classification

3. Activation Layer (e.g. ReLU):

1. Following the convolutional layer, an activation function is typically applied element-wise to add non-linearity to the network.
2. The **Rectified Linear Unit (ReLU)** is the most common activation function in CNNs. It replaces all negative values with zero, keeping only the positive activations.
3. This layer helps the CNN learn more complex patterns by allowing non-linear combinations of features.



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

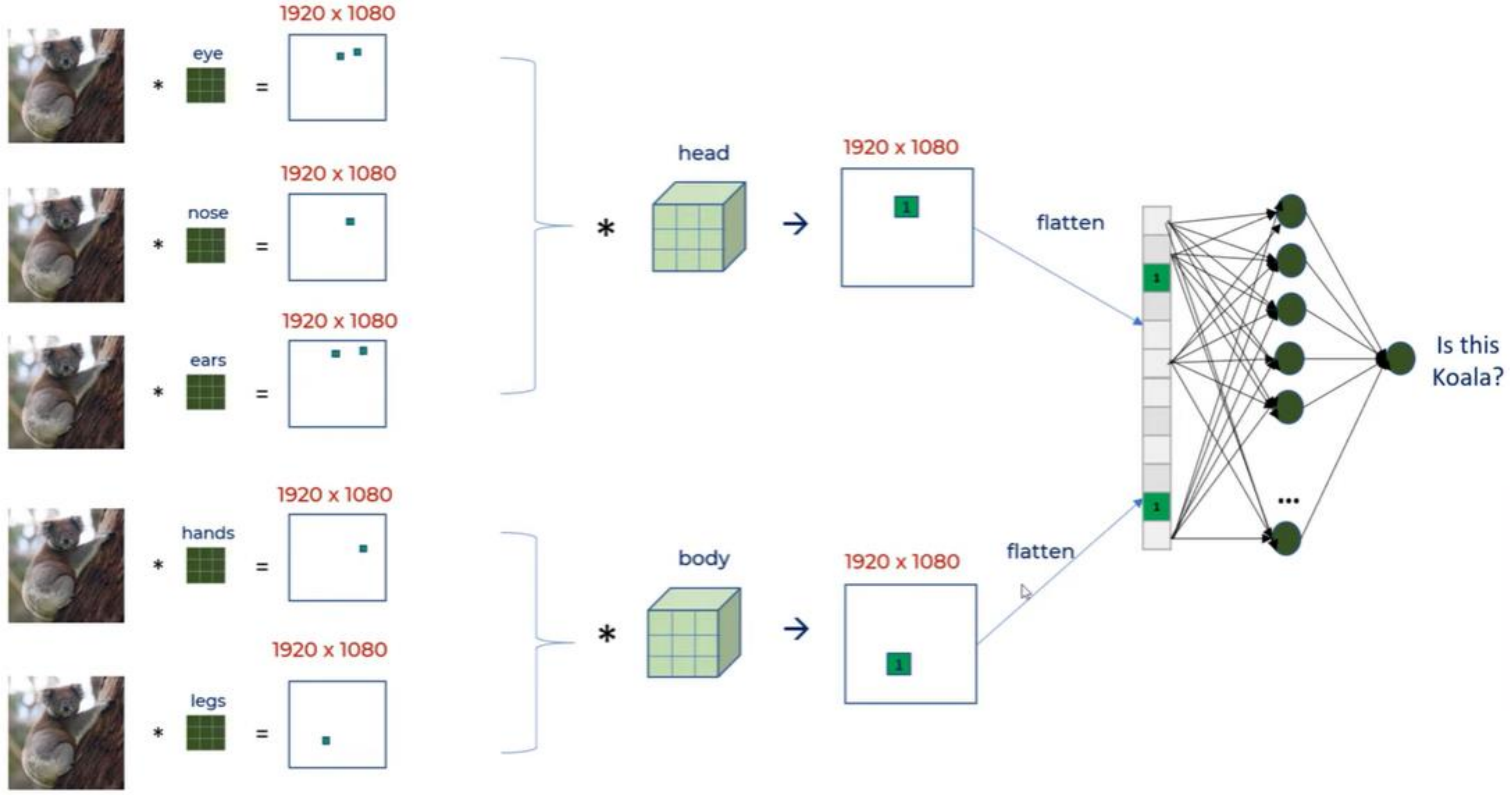
1	1	1
1	-1	1
1	1	1

-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

4. Pooling Layer (e.g., Max Pooling):

1. Pooling layers reduce the spatial dimensions (height and width) of the feature maps, which decreases the number of parameters and computational load.
2. **Max Pooling** is the most common pooling method, where the maximum value within a sliding window (e.g., 2x2) is taken to represent that region.
3. Pooling also adds a degree of translation invariance, making it easier for the network to recognize objects in different positions within the image.
4. **Example**:
   1. For a 2x2 max pooling layer applied to a section like [1324]\begin{bmatrix}1 & 3 \\ 2 & 4\end{bmatrix}[1234], the result is 4.


1920 x 1080
eye
1920 x 1080
nose
1920 x 1080
ears
1920 x 1080
hands
1920 x 1080
legs
head
1920 x 1080
1
flatten
body
1920 x 1080
1
flatten
1
1
...
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

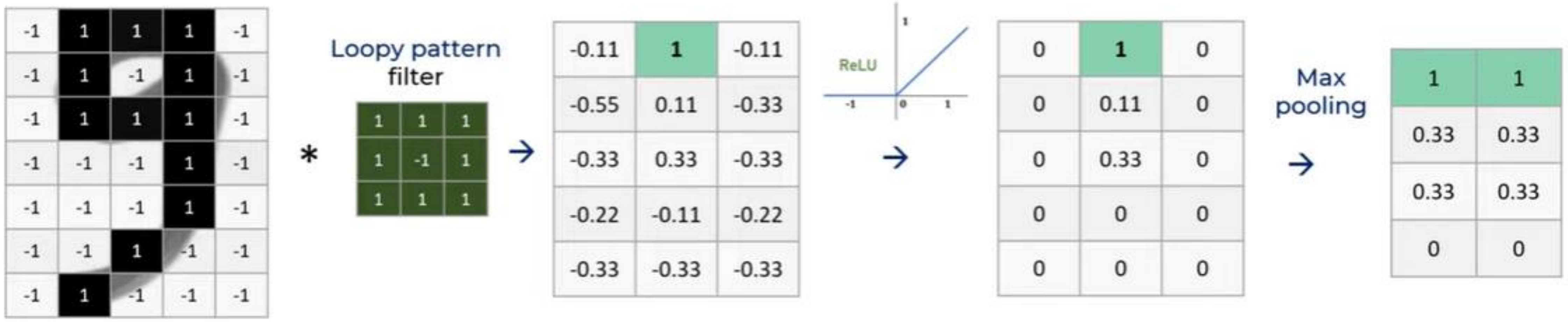
8	**9**
3	2

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



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

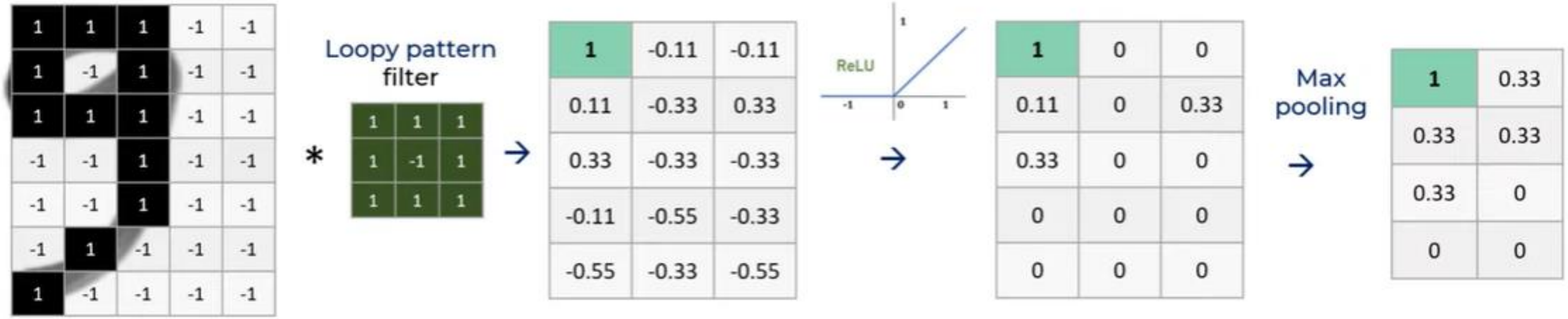
1	1	1
1	-1	1
1	1	1

-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

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

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

ReLU

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

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

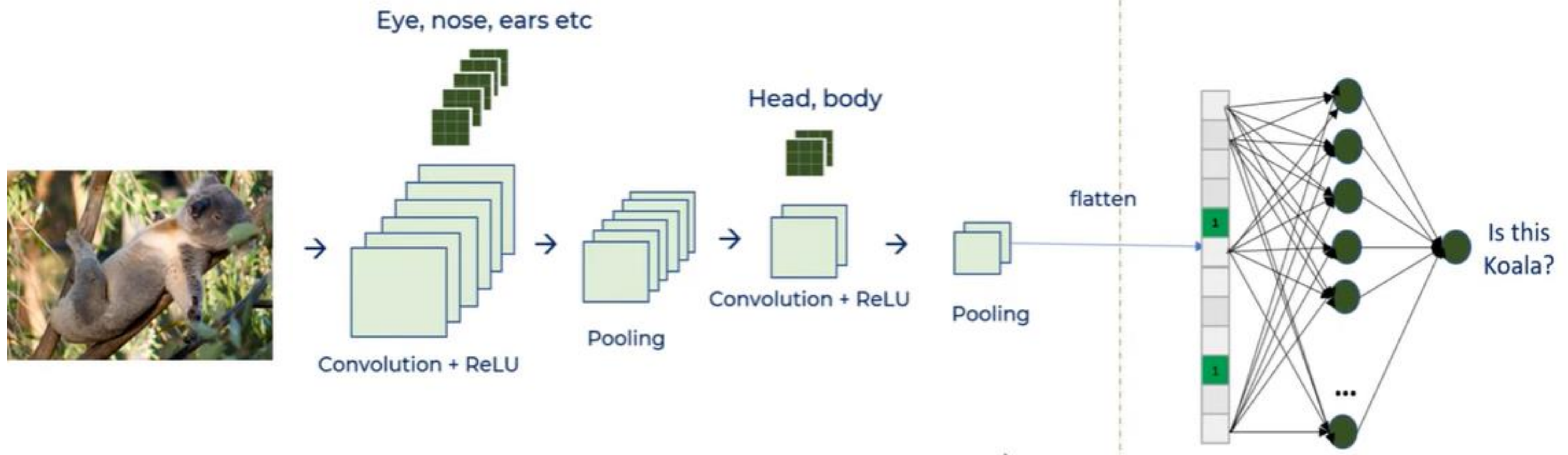
Reduces dimensions & computation

Reduce overfitting as there are less parameters

Model is tolerant towards variations, distortions

5. Fully Connected (Dense) Layer:

1. After several convolutional and pooling layers, the feature maps are flattened into a 1D vector and fed into one or more fully connected (dense) layers.
2. These layers learn complex patterns by combining all features learned in previous layers, enabling the network to make final predictions.
3. The fully connected layers treat each feature equally and produce the final output based on the learned weights.



Feature Extraction

Classification

6. Output Layer:

1. The output layer depends on the task at hand:
    1. **Classification**: A softmax activation function is often used in the output layer for multi-class classification. The softmax outputs a probability distribution over classes, with each value representing the probability of the input belonging to a specific class.
    2. **Regression**: For regression tasks, the output layer usually has a single neuron without activation (or sometimes with linear activation) to predict a continuous value.

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