IMAGE CLASSIFICATION

Let's take an image size is 1920 x 1080 x 3 (3 is for RGB)

First layer neurons =1920 x 1080 x 3= 6 million apprx.

Hidden layer neurons = would be lets say around 4 million

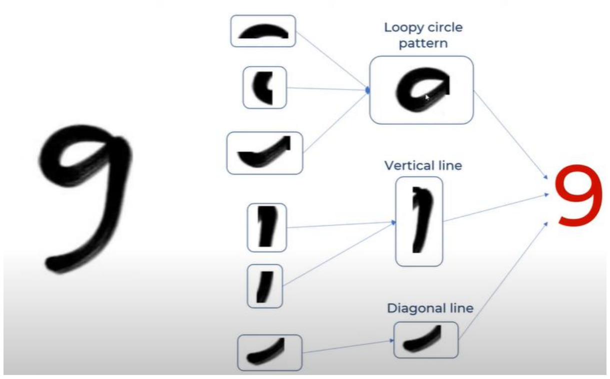
And now if we come to weights , its around 6x4 =24 million weights apprx. , plus there are more than 1 hidden layer.

VERY HIGH COMPUTATION IS THERE

DISADVANTAGES OF USING **ANN** FOR IMAGE CLASSIFICATION:-

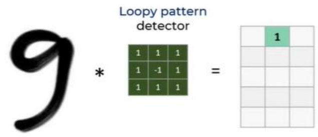
- Too much computation
- Treats local pixels same as far apart ones
- Sensitive to location of an object in an image

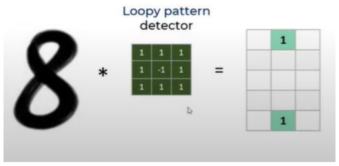
CNN (Convolutional Neural Network)



This how CNN works :

- Divides parts of the image and find it individually by using FILTERS and aggregate them at the end
- For 9 we have three filters :
    - Loopy Pattern Filter
    - Vertical Line Filter
    - Diagonal Line Filter

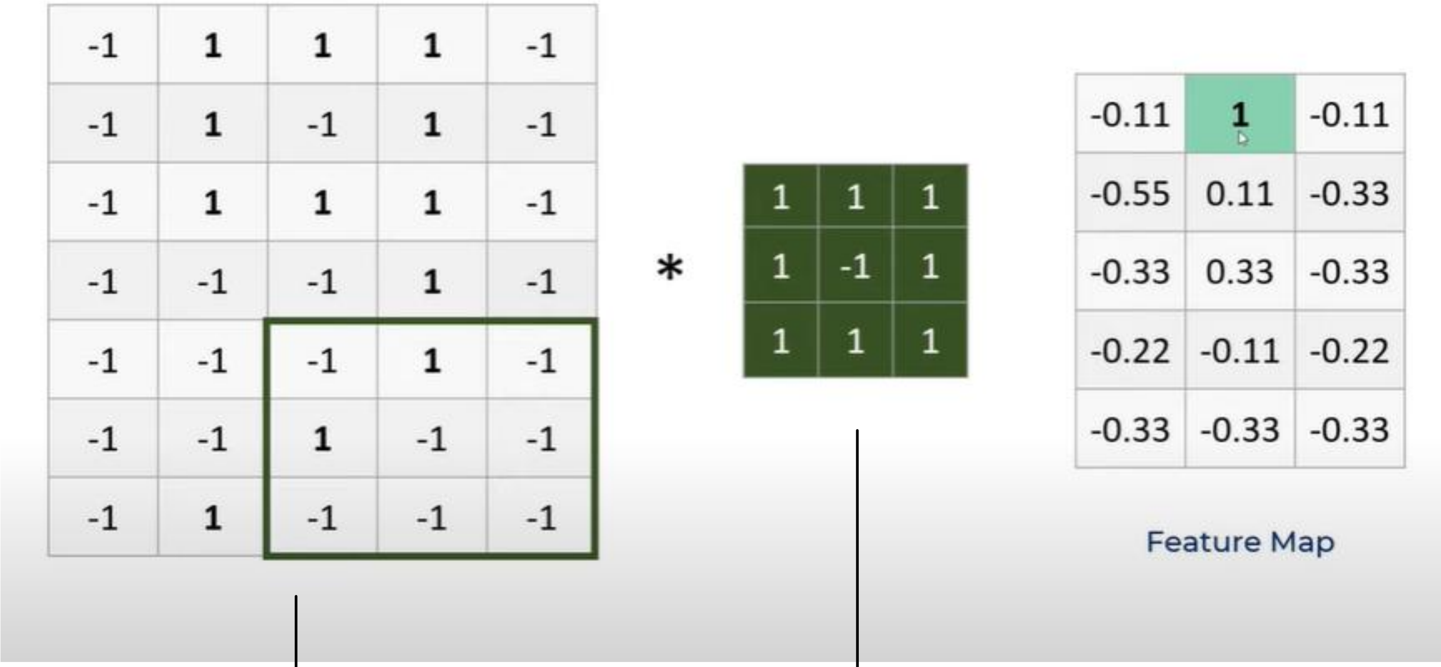




How THIS ‘1’ is calculated?

->Take every 3x3 box (same as filter size) from image grid and take an avg of (multiplication of respective number from both matrices and add them)

-> For below example : ((-1x1)+(1x1)+(1x-1)+(1x1)+(-1x-1)+(1x-1)+(-1x1)+(-1x1)+(-1x1))/9=-0.33



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

*

1	1	1
1	-1	1
1	1	1

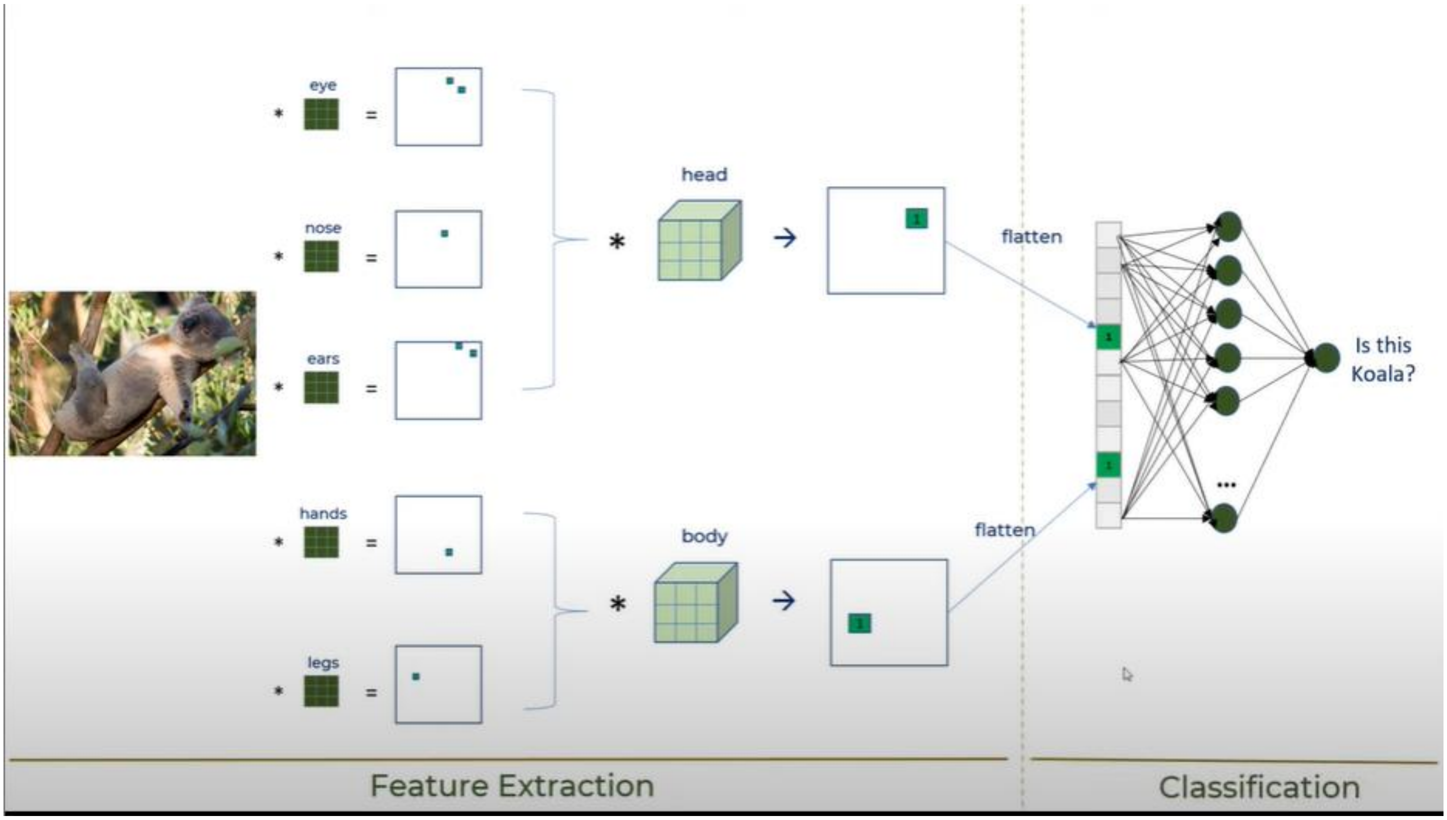
-0.11	1	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

Feature Map

IMAGE GRID

Loopy Filter

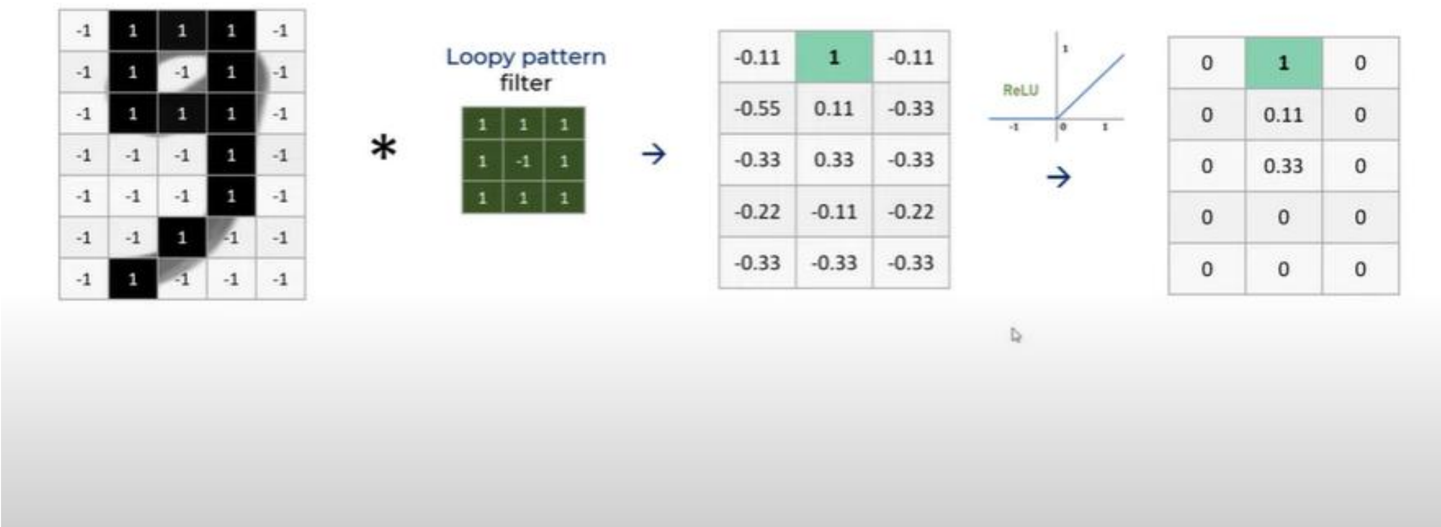
Lets take an example of a kuala image



- All the features of a kuala were divided and for each of them a filter is used. Ex – Nose ,eye ,ears, hands and legs.
- Convolutional operation is applied again and respective features were aggregated to form 3D filter head and body.
- Which is then flattened for further creating a fully dense neural network for CLASSIFICATION.

ReLU Funtion (Rectified Linear Unit)

- It is used to bring the NON LINEARITY in the mode



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

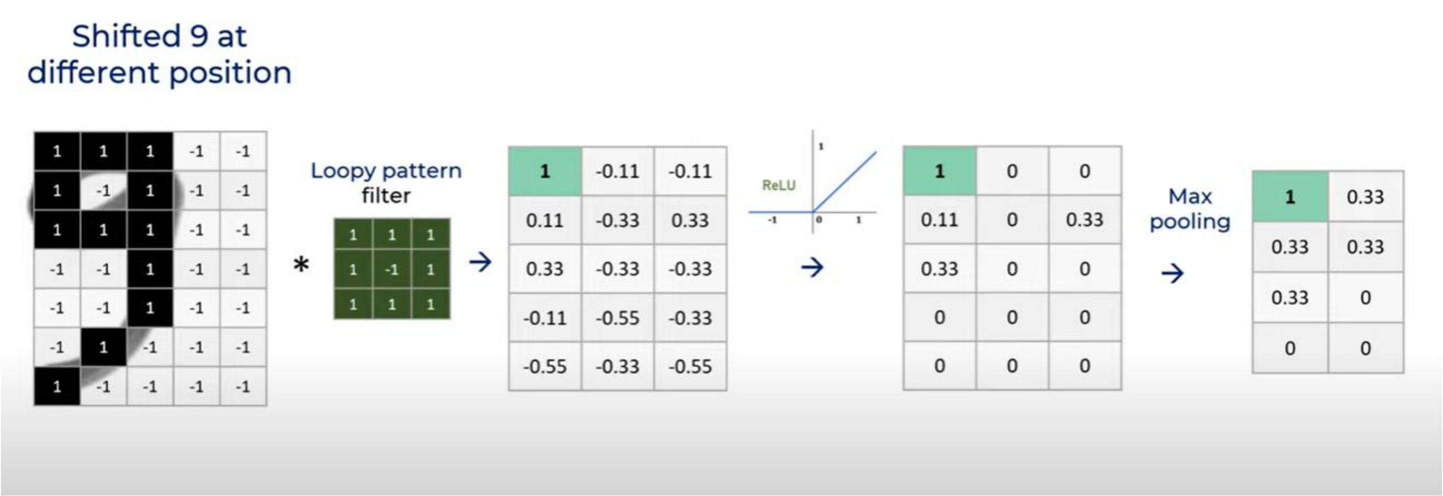
→

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

→ It speeds the training , faster to compute.

What about the HIGH COMPUTATION problem ?

➔ We will use POOLING Layer to reduce the size



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

➔ **Max Pooling** is done here with **2x2 filter with STRIDE = 1**

➔ There is **Average Pooling** also but preferred is MAX Pooling

➔ **SIZE** is **reduced** significantly here from **15 values -> 8 values**

➔ Makes the model **tolerant of small distortions** and **variations**

Problem with CNN

➔ CNN by itself doesn’t take care of **ROTATION** and **SCALE**.

  - You need to have **ROTATED** and **SCALED** samples in the training dataset.
  - If you don’t have such samples in dataset then use **data augmentation methods** to **generate new rotated/scaled samples** from existing training dataset.