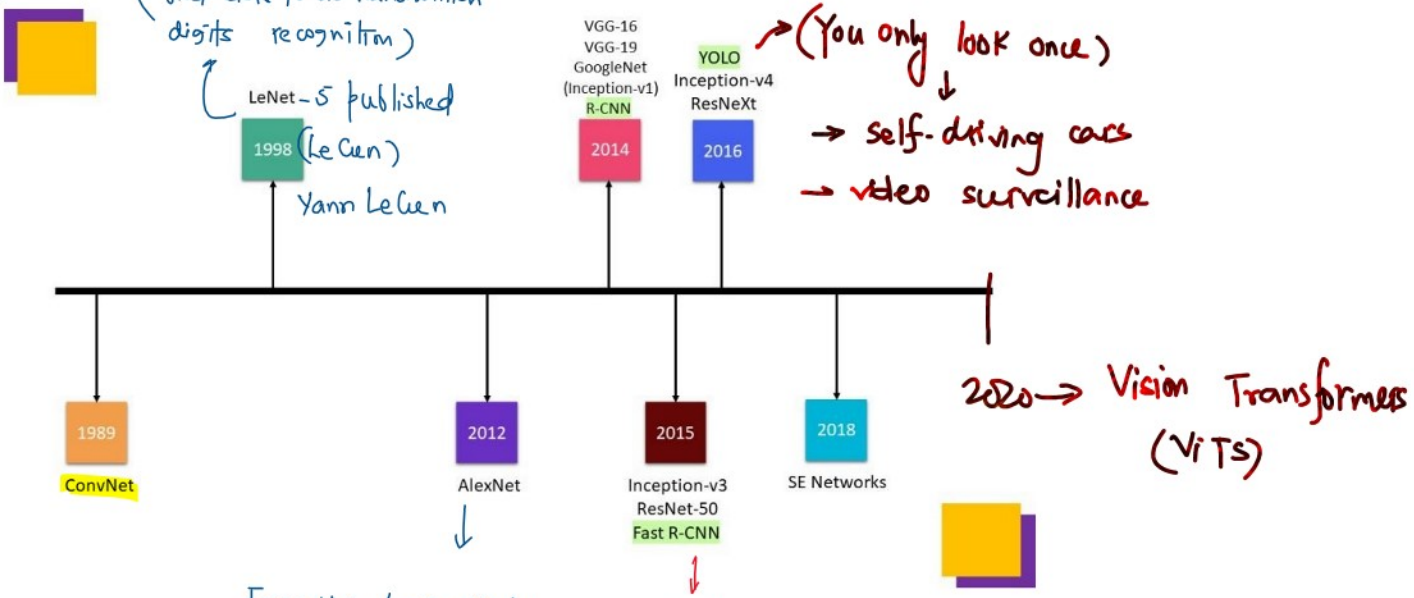
CNN Detailed Discussion

28 September 2024 21:41

CNNs have proven very effective in image recognition and classification.

- Really successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars.

a soccer player is kicking a soccer ball | a street sign on a pole in front of a building | a couple of giraffe standing next to each other



ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

Training (upto 152 layers)

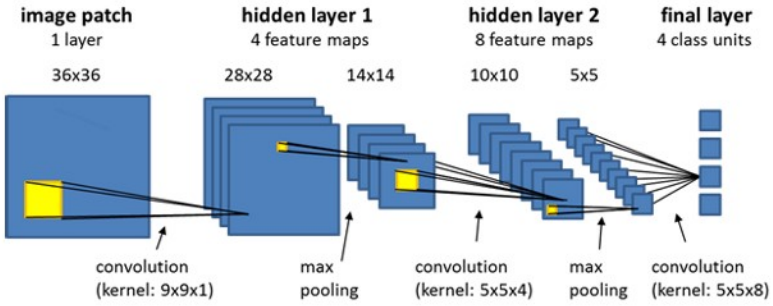
- ReLU activation function
- Dropout
- GPU acceleration

⇓

- GPU acceleration

⇓

(to make deep CNNs feasible for large-scale classification tasks)



① ReLU Activation
② Dropout
③ Batch Normalization
④ Transfer learning

} CNN (recent developments)

Architecture overview

A typical CNN architecture consists of several key layers:

1. Input Layer

- the input layer of a CNN receives the raw image data in the form of a multi-dimensional array (also known as Tensor)

For grayscale images

input Tensor → (H x W x 1)

- H: Height of the image (no. of pixels vertically)
- W: width of the image (no. of pixels horizontally)

* W: width of the image (no. of pixels horizontally)
* 1: single color channel - grayscale.

For color (RGB) images:

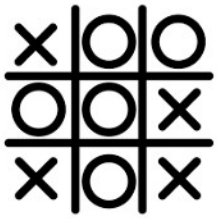
input tensor ⟶ (H x W x 3)

↳ 3 represents the three color channels – red, blue green.

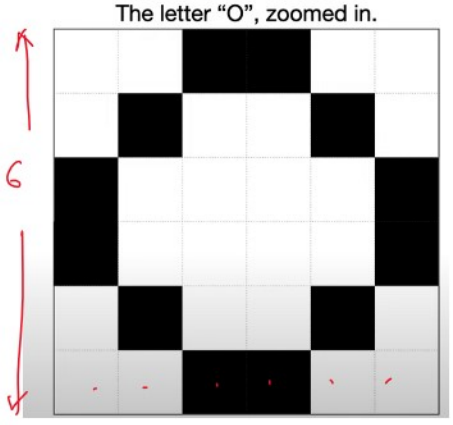
2. Convolutional Layer

- core building block of CNN
- it applies filters (also known as kernels) to extract features like edges, corners, textures etc.
- the filters slide over the input and this process is called convolution and it generates a feature map as output.

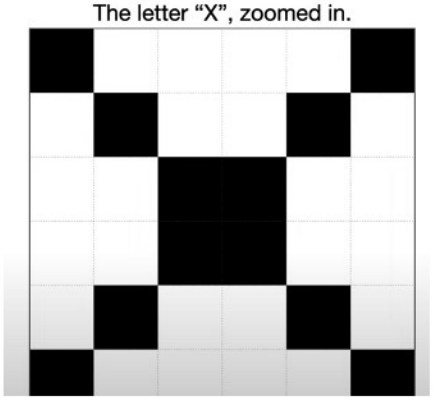
Tic- Tac- Toe

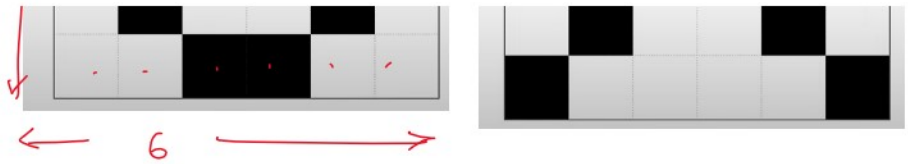


6x6 pixels

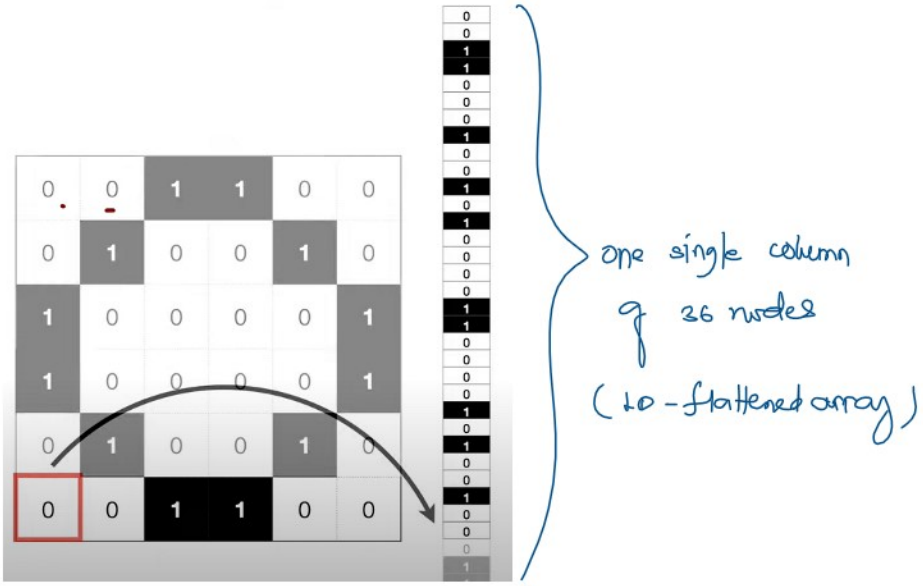


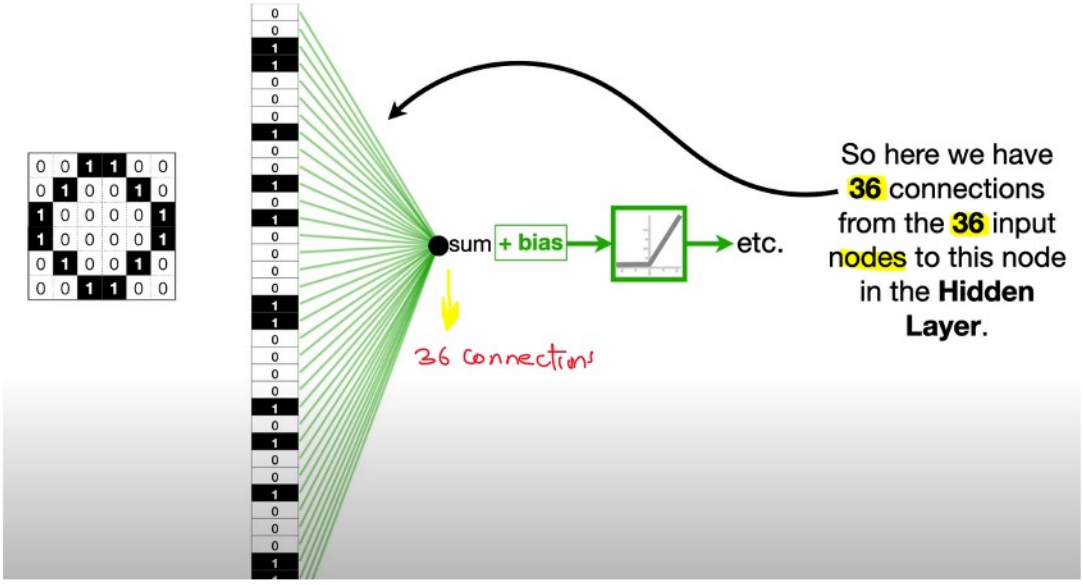
6x6 pixels

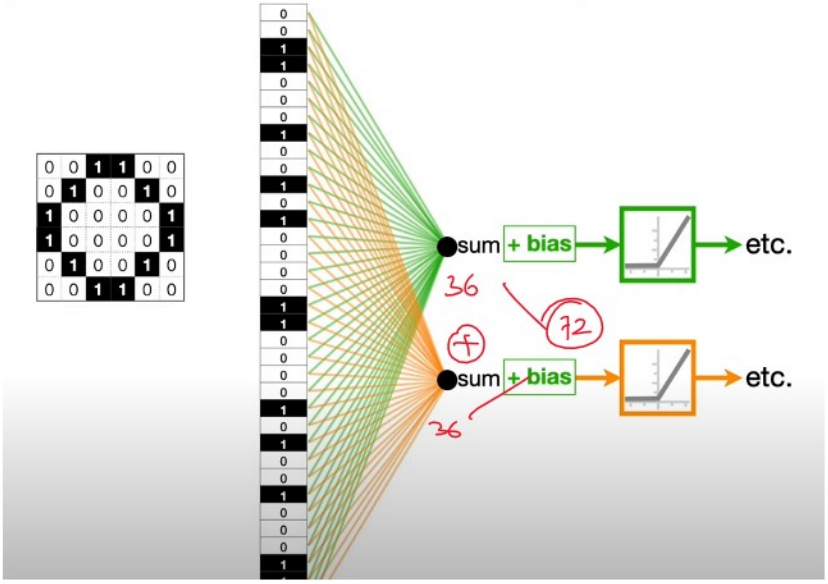




In artificial neural n/w,







1 Mega pixel → 1×10^6 pixels

$\sqrt{10^6} = 1000$ pixels → 1 mega pixel image.

1000 pixels

● → one neuron → 1 input (10^6 connections)

For a neural network having 1×10^6 (input) connections for a standard size image the model needs to estimate 1 Million weights for each neuron which is computationally highly expensive.

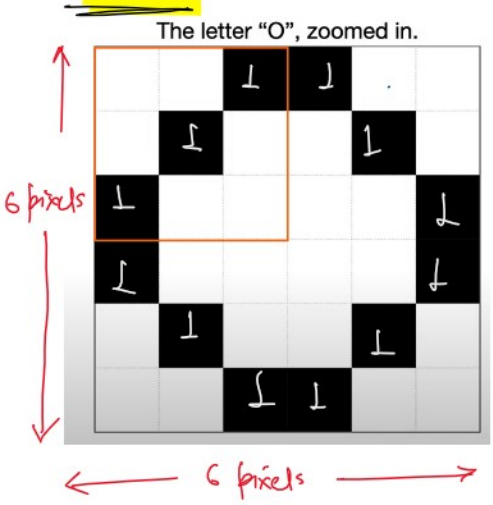
Applying neural networks (vanilla) is not the right approach as it doesn't scale well.

CNN comes and simplify it and make it computationally less heavy.

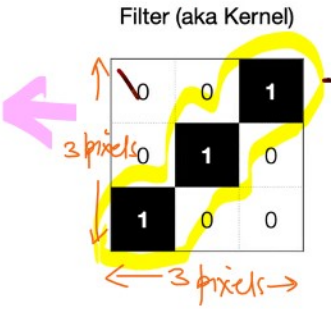
CNNs do three things to make image classification practical:

1) Reduce the no. of input nodes
2) Tolerate / accommodate the marginal shifts in the pixels within the image.
3) Mind the correlations in the complex image.

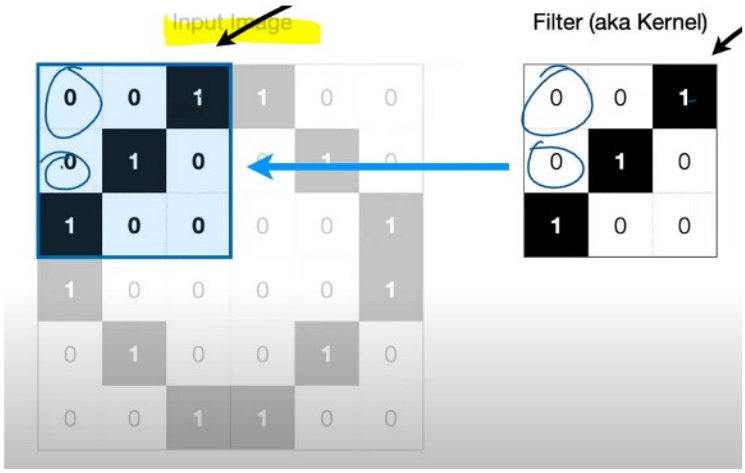
Step #①



CNN applies a filter to the input image.



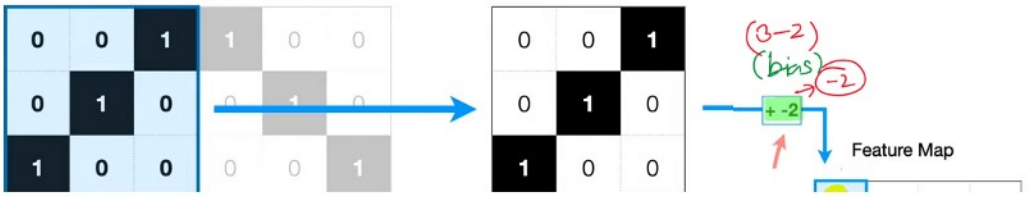
→ on off-diagonal, values are 1 and rest all zero

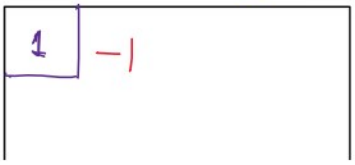


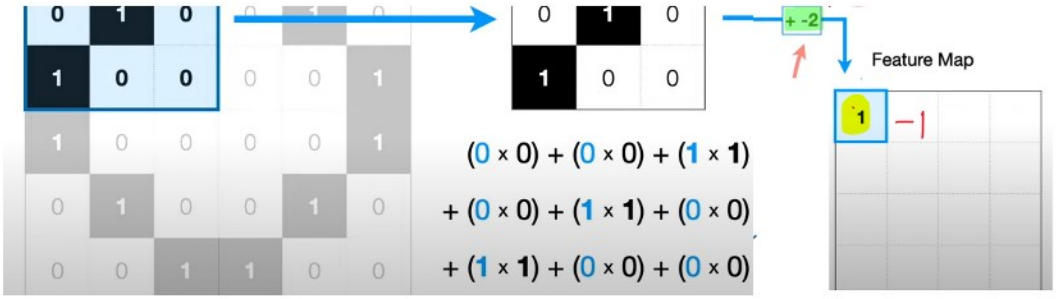
In convolution, dot product between the input and the filter in the overlapped area, and hence we can say that the filter is convolved with the input.

dot product → sum of products

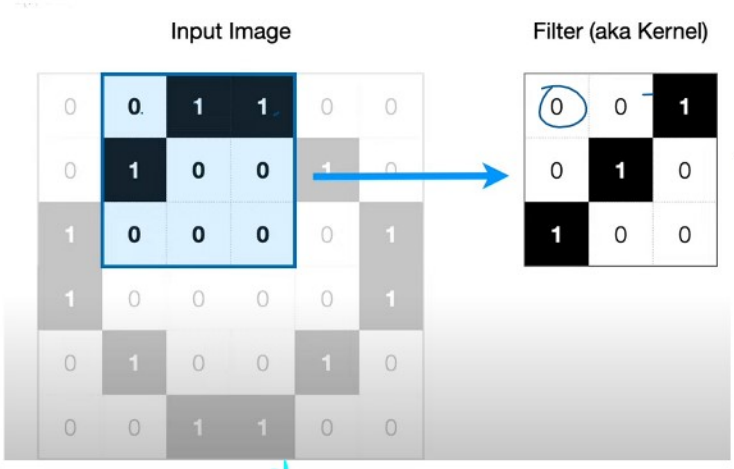
$(0\times0)+(0\times0)+(1\times1)$
$+(0\times0)+(1\times1)+(0\times0)$
$+(1\times1)+(0\times0)+(0\times0)$
$=3$ → bias (−2) → 1







Next shift by a pixel:



(0x0) + (1x0) + (1x1)
+(1x0) + (0x1) + (0x0)
+(0x1) + (0x0) + (0x0) = 1

bias (-2)

-1

(Final Feature Map)

Feature Map

1	-1	-2	-1
-1	-2	-1	-2
-2	-1	-2	-1
-1	-2	-1	1

ReLU

1	0	0	0
0	0	0	0
0	0	0	0
0	0	0	1