CNN: Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a specialized type of neural network designed for image and video data. They are particularly effective for tasks like image classification, object detection, and segmentation.

CNN : I/P => Images **Eg: Image Classification**

Object Detection

Input Layer

Images:

Black & White:

Matrix Representation: For a 5x5 black-and-white image

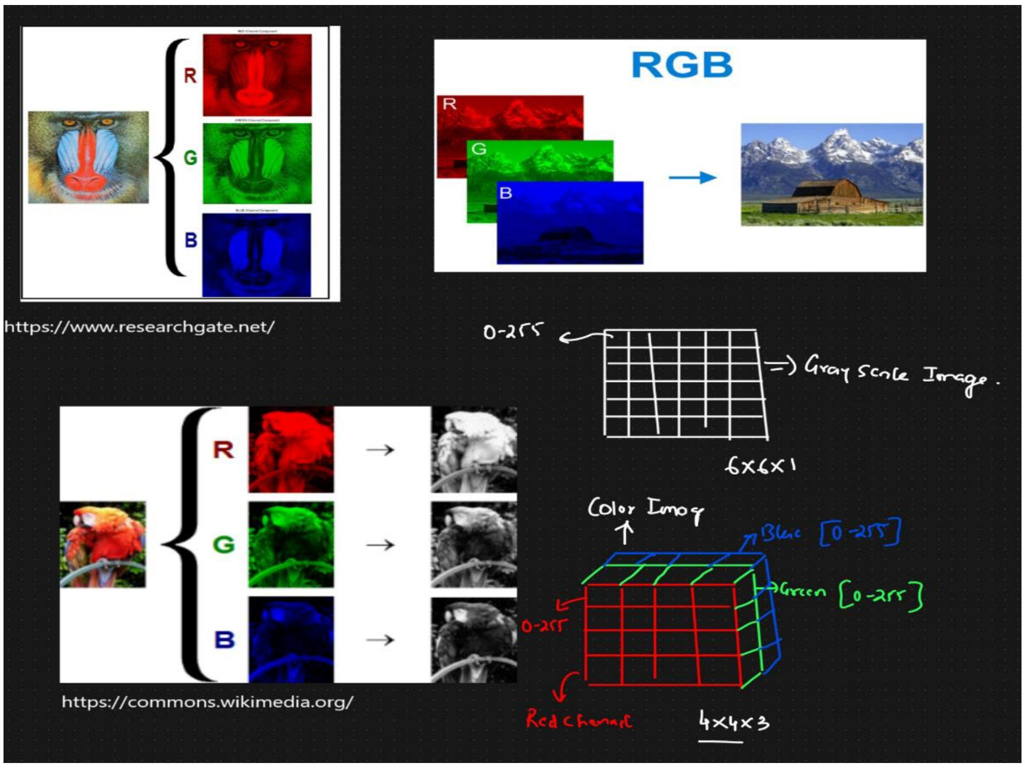
Pixel Values: Range from 0 (black) to 255(white).

Channels: 1 (grayscale).

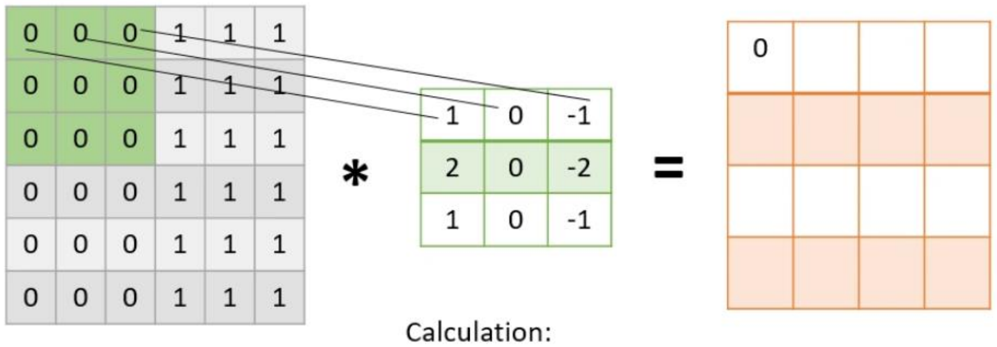
RGB:

Matrix Representation: For a 5x5 RGB image, it is a 3D matrix of size 5×5×3

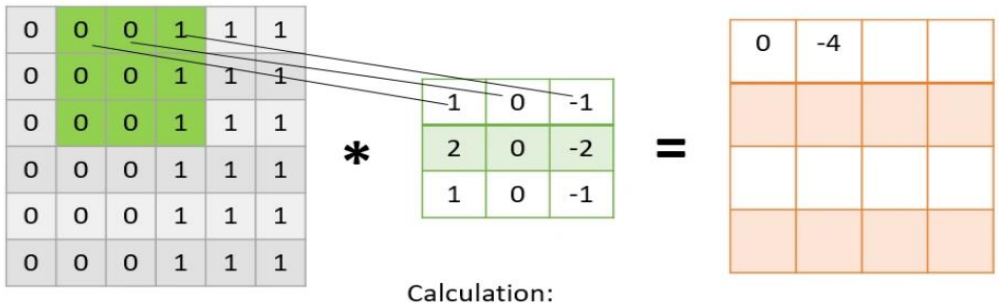
Image= [Red Channel, Green Channel, Blue Channel]



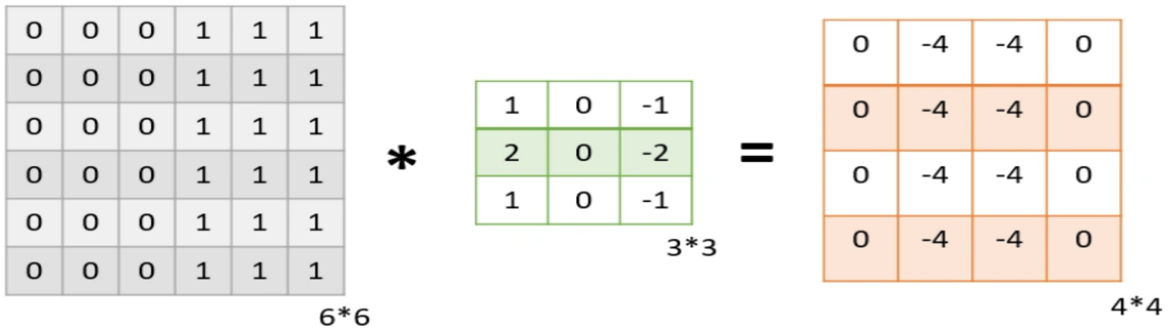
- **Convolutional Layers**: Extract features from the input image by applying filters

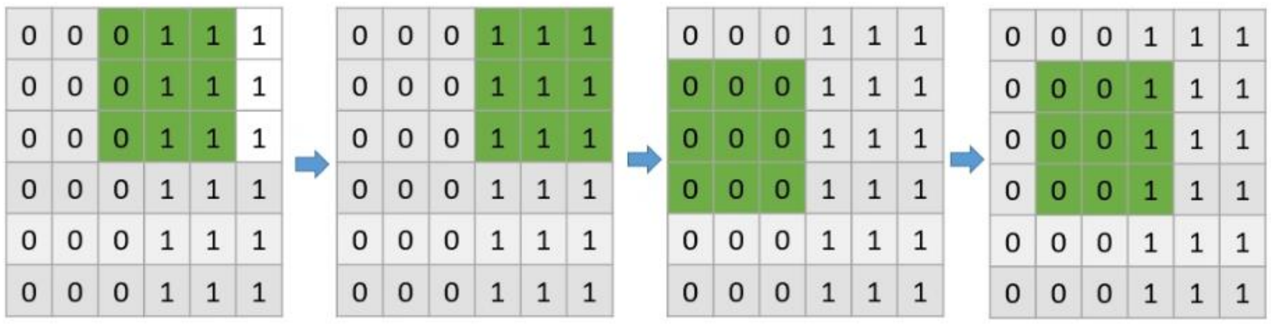


Calculation:
0*1 + 0*0 + 0*-1 +
0*2 + 0*0 + 0*-2 +
0*1 + 0*0 + 0*-1



Calculation:
0*1 + 0*0 + 1*-1 +
0*2 + 0*0 + 1*-2 +
0*1 + 0*0 + 1*-1





.

Activation Layers (ReLU): Introduce non-linearity into the network.

ReLU is the most commonly used activation function in CNNs because it is computationally efficient, simple, and helps introduce non-linearity into the network. Its ability to handle large datasets and avoid the vanishing gradient problem makes it highly effective in training deep neural networks. Variants like Leaky ReLU and ELU are used to overcome some of the limitations of ReLU, such as dead neurons and the lack of negative outputs.

Example:

Let's say you have the following output from a convolutional layer (feature map):

```
[-1,2-3]
[4-5,6]
```

After applying the ReLU activation function:

[0, 2, 0]

```
[4,0,6]
```

In this case, all the negative values are replaced by 0, and the positive values remain unchanged.

Pooling Layers: Reduce the spatial dimensions of feature maps to retain only the most important features

Example of Max Pooling

[1, 3, 2, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]
[13, 14, 15, 16]

With 2×2 max pooling:

- First window: [1,3,5,6] → Max value = 6
- Second window: [2,4,7,8] → Max value = 8
- Third window: [9,10,13,14] → Max value = 14
- Fourth window: [11,12,15,16] → Max value = 16

After pooling, the output feature map is:

```
[6, 8]
[14, 16]
```

Flattening: Flattening converts this h×w×d matrix into a 1D vector of size h·w·d.

h = height of the feature map.

w = width of the feature map.

d = number of channels (depth).

Dimensions: 4×4×3 (height × width × channels).

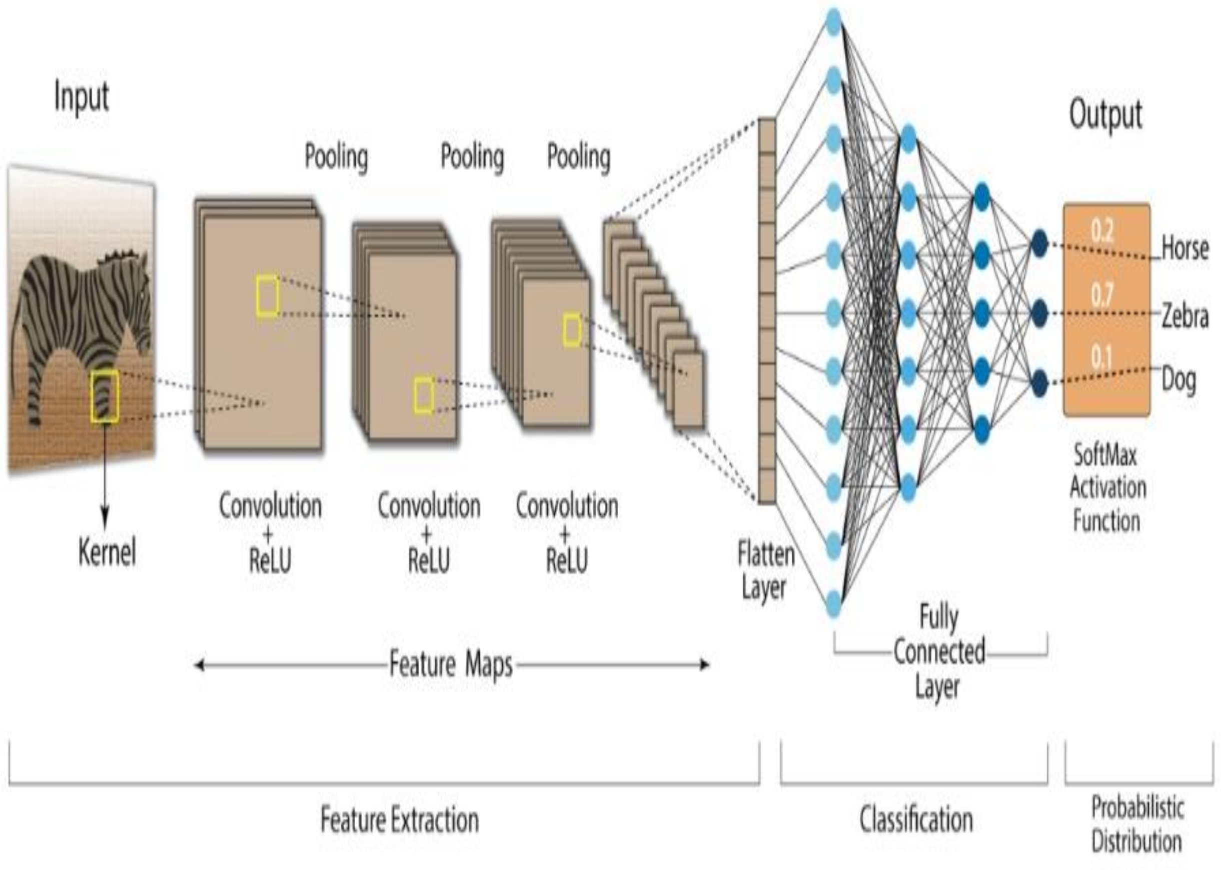
Example: A 3-channel feature map with 4 rows and 4 columns.

Flattened Output:

Converts the feature map into a vector of size 4·4·3= 48

The output is a 1D array with 48 elements.

Fully Connected Layers: Use the extracted features to make final predictions


Input
Pooling
Pooling
Pooling
Output
0.2
Horse
0.7
Zebra
0.1
Dog
SoftMax Activation Function
Kernel
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Flatten Layer
Fully Connected Layer
Feature Maps
Feature Extraction
Classification
Probabilistic Distribution