Convolution Operation

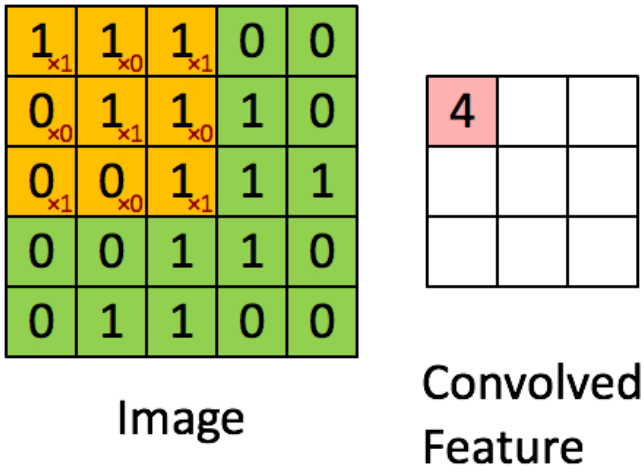
- A **Convolution operation** is a **mathematical way to extract patterns** from data (like images).
- It works by **sliding a small matrix (called a filter or kernel)** over the image, **multiplying values**, and **summing them** to **create a new image** (**called a feature map or activation map**).
- This helps the system **recognize patterns like edges, shapes, or textures**.

🔍 What is Convolution in Simple Words?

- Convolution = **Pattern-matching process**
- We use a **filter** (also called a **kernel**) to look at **small parts of the input** (like small windows).
- The filter slides across the input and does **element-wise multiplication**, then **adds all values** to make a **single number**.
- This creates a new matrix (called a **feature map** or **convolved output**) that highlights certain features in the input.

How It Works

1. **Filter Slides Over Input**:
    - The kernel (e.g., 3×3 matrix) moves across the input image (left-to-right, top-to-bottom).
    - At each position, it performs **element-wise multiplication** between the kernel and the overlapping region of the input, then sums the results.
2. **Output Feature Map**:
    - Each computed value becomes a pixel in the output feature map, highlighting detected features (e.g., edges).



Example (Edge Detection):

- **Input Image (5×5)** and **Filter (3×3)**:

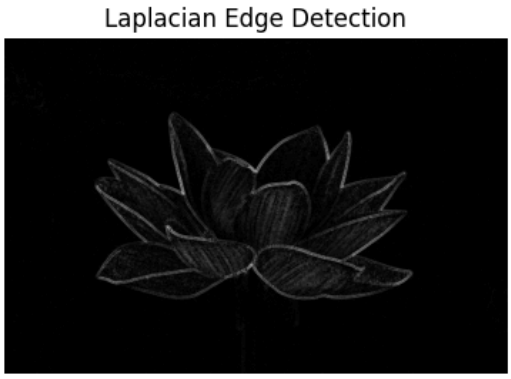
```
Input (I):      Kernel (K):
[1, 1, 1, 0, 0]   [-1, -1, -1]
[0, 1, 1, 1, 0]   [-1,  8, -1]
[0, 0, 1, 1, 1]   [-1, -1, -1]
[0, 0, 1, 1, 0]   (Edge detector)
[0, 1, 1, 0, 0]
```

- **Convolution Output (3×3)**:
  - The filter detects transitions (edges) where pixel values change sharply.

Edge Detection

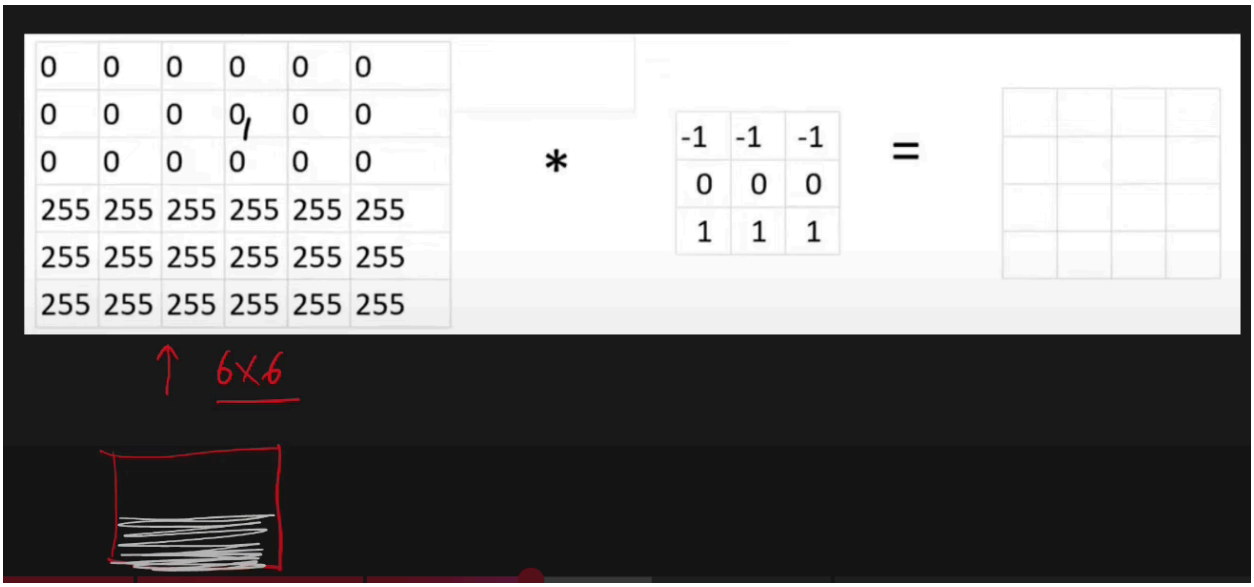
- Edge detection is → **Intensity change**


Original Image


Laplacian Edge Detection

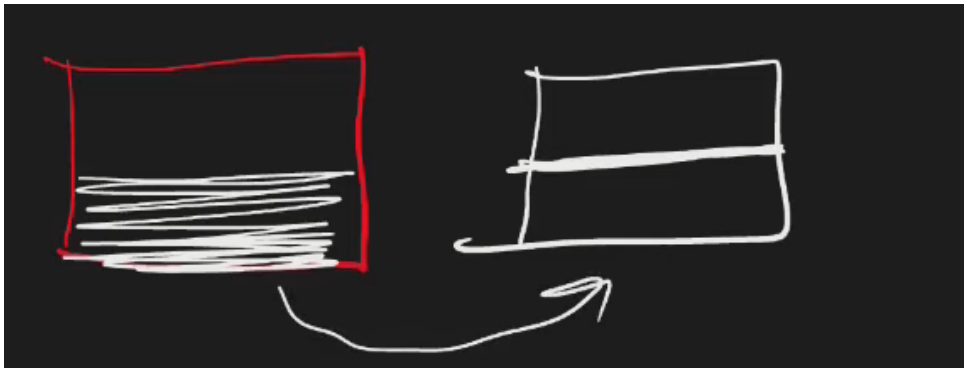

Original Image
Edge Image

How it works?



- 👆Above is 6 × 6 image
- The above part is black and the below one is white

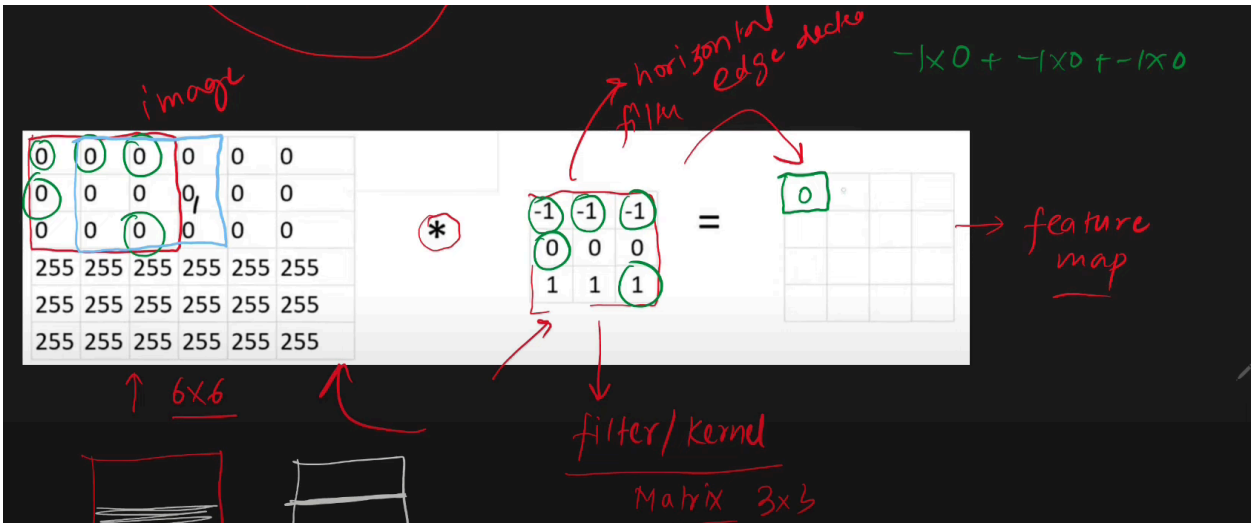
Expected output:



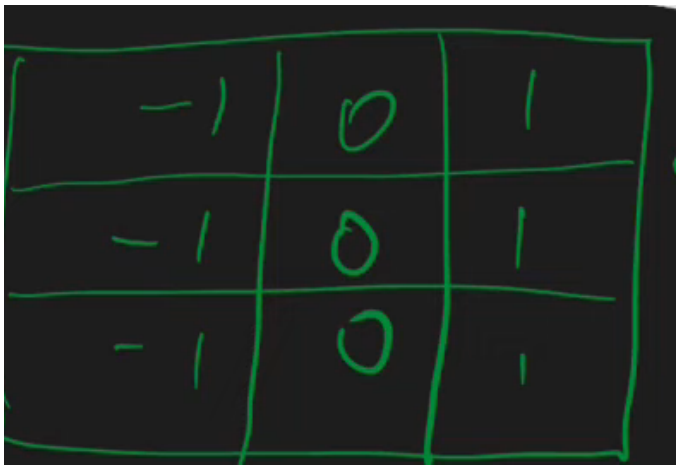
We need a filter (kernel) for this (Generally a 3 × 3 matrix)

1. You put the kernel on the image
2. Multiply the respective numbers
3. Add the sum in the respective cell
4. Slide the filter

5. Repeat



- 👆 Above is the filter for **horizontal** edge detector
- For **vertical** edge detection, the filter is 👇



- Similarly, there are filters for various other edge detectors.

💡 **We don’t need to make the filters in CNN. We just give the shape of the matrix**

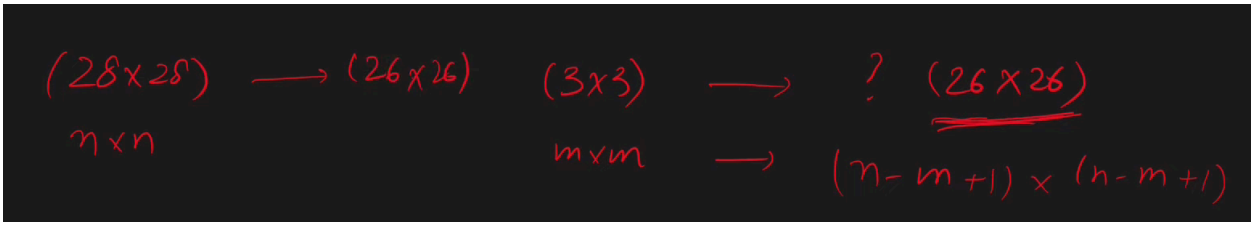
CNN Demo → https://deeplizard.com/resource/pavq7noze2

Image shape → (28 × 28)

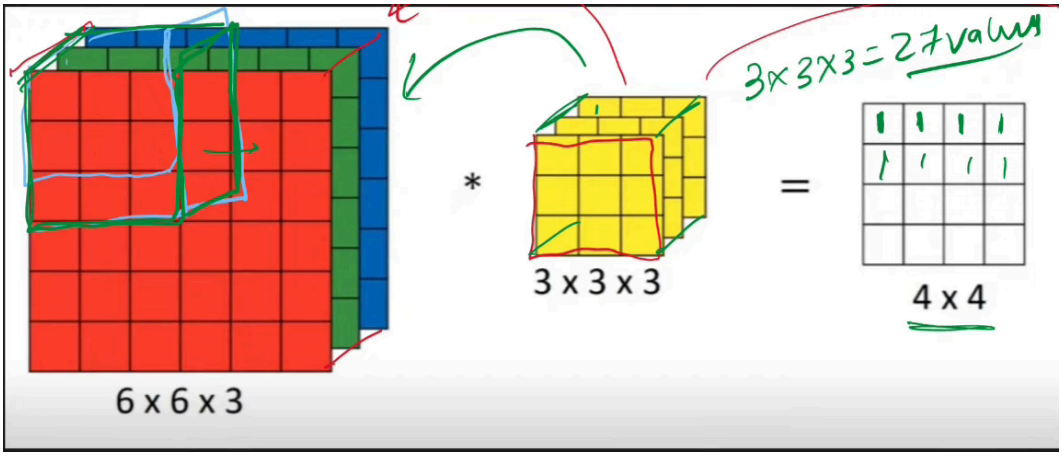
Filter → (3 × 3)

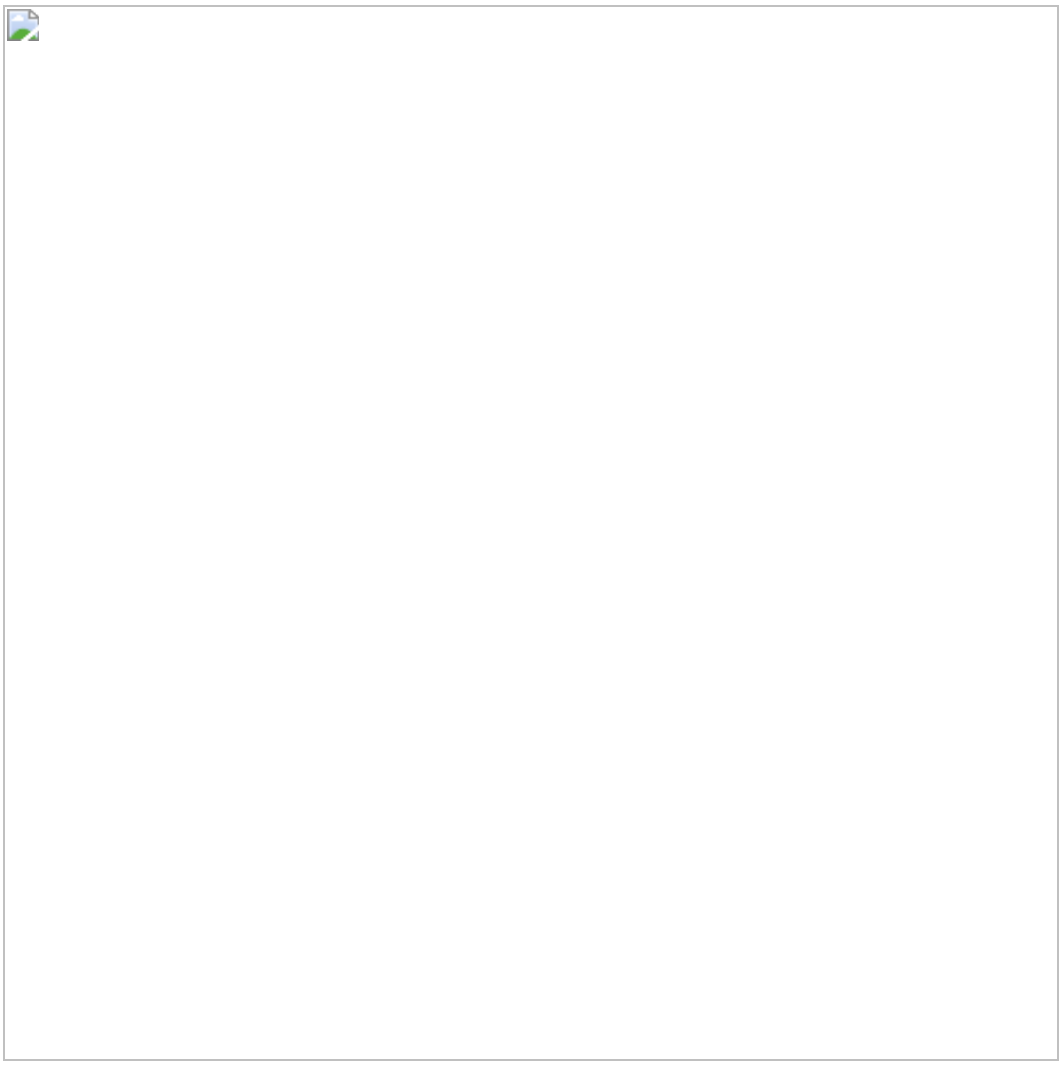
Output→ **26 × 26**

Output shape = (n-m+1) x (n-m+1)

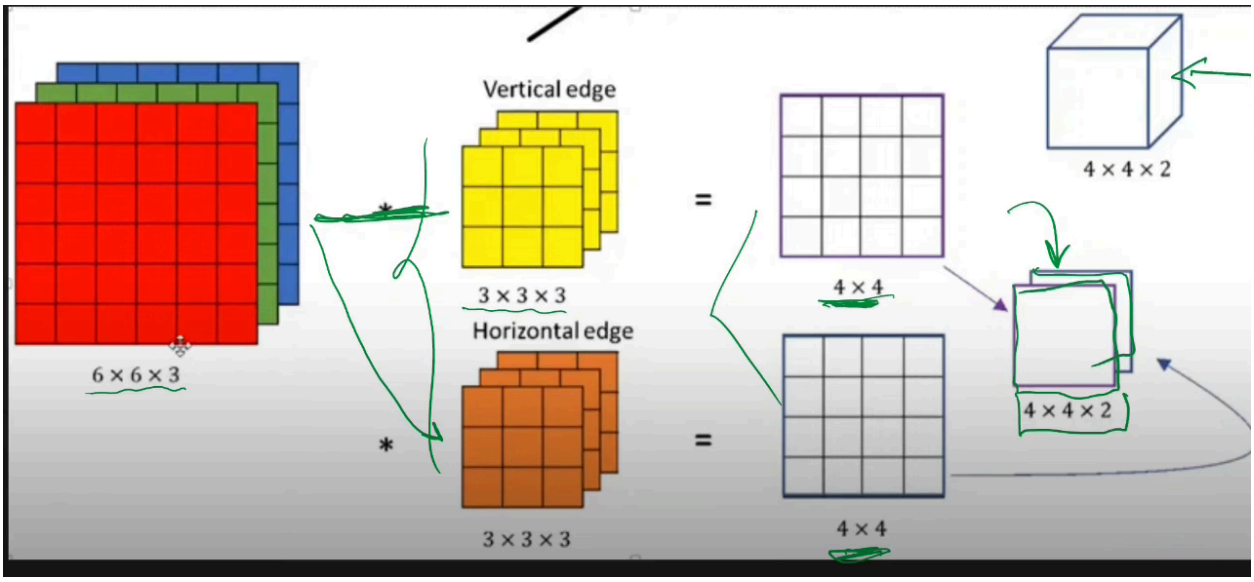


RGB Filtering:





Multiple Filters



- You get multiple feature maps
- The channels can act as filters for subsequent layers

Key Parameters in Convolution

Parameter	Meaning
`kernel_size`	Size of the filter (e.g., 3×3)
`stride`	Number of steps the filter moves at a time (default: 1)
`padding`	Adds zeros around the border to preserve dimensions
`filters`	Number of different filters used (each finds a different pattern)