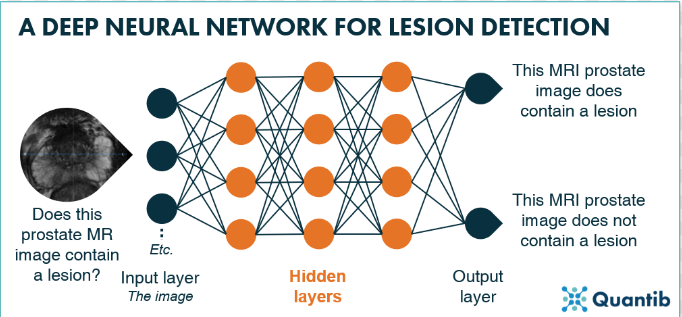
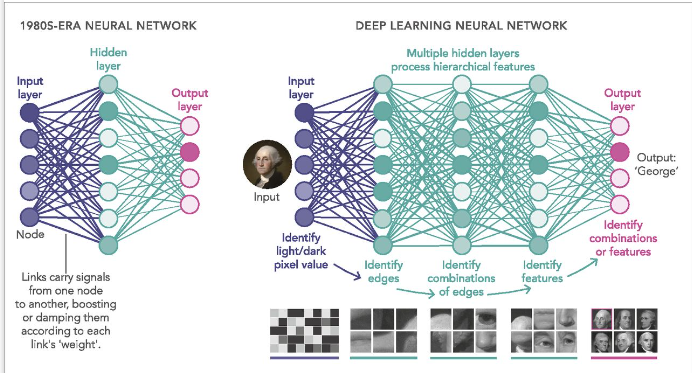
**Convolutional Neural Networks.**

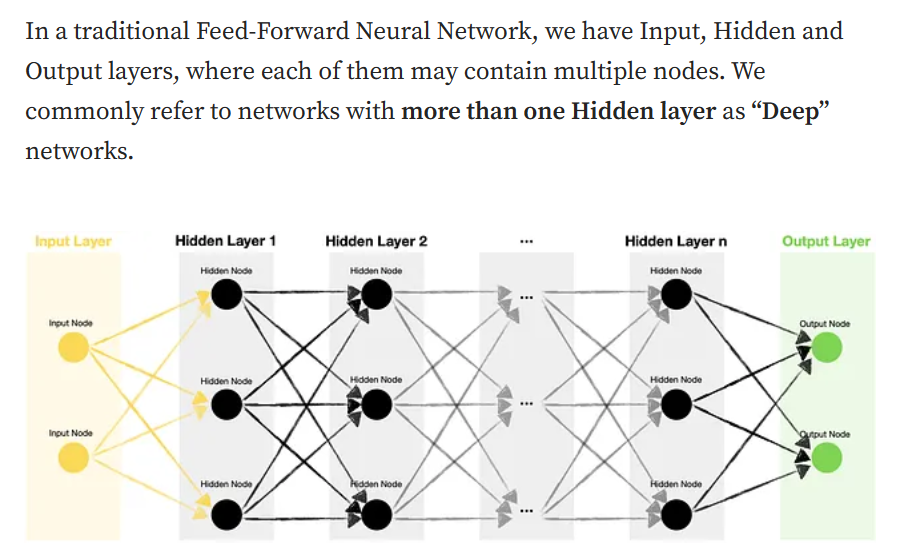
# Convolutional Neural Networks (CNNs) Deep Learning – which has emerged as an effective tool for analyzing big data – uses complex algorithms and artificial neural networks to train machines/computers so that they can learn from experience, classify and recognize data/images just like a human brain does.

****

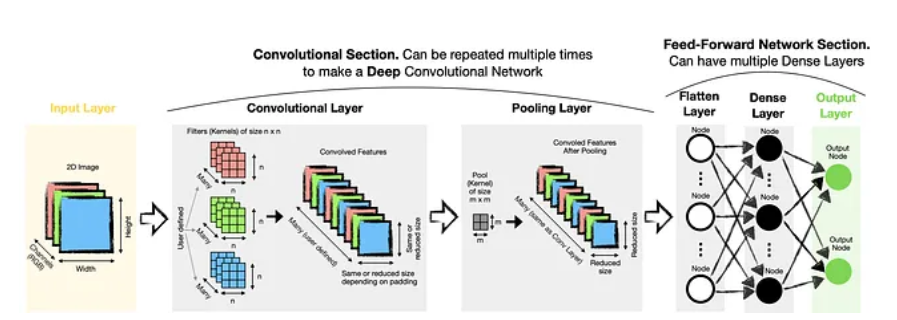
****

Convolutional layer is core building block of CNN, it helps with **feature detection.**

# What is the structure of Convolutional Neural Networks, and how do they work?

****

Meanwhile, Convolutional Neural Networks (CNN) tend to be multi-dimensional and contain some special layers, unsurprisingly called **Convolutional layers**. Moreover, Convolutional layers are often accompanied by **Pooling layers (Max or Average),** which help reduce the size of convolved features.

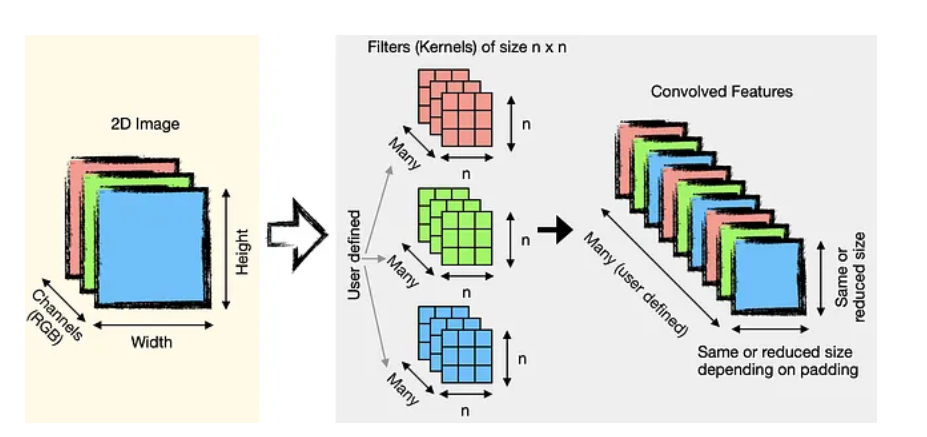
****

## Convolutional layer

It is worth highlighting that we can have Convolutional layers of different dimensions:

* One-dimensional (**Conv1D**) — suitable for text embeddings, time-series or other sequences.
* Two-dimensional (**Conv2D**) — typical choice for images.
* Three-dimensional (**Conv3D**) — can be used for videos, which are essentially just sequences of images, or for 3D images such as MRI scans.

Since I focus on image recognition in this article, let’s take a closer look at how 2D convolution works. 1D and 3D convolutions work in the same way, except they have one fewer or one extra dimension.

****

Note that for a greyscale picture, we would only have one channel. Meanwhile, we would have **three separate channels for a colour picture**, each containing values for its respective colour (Red, Green, Blue).

Convolutional layer is core building block of CNN, it helps with **feature detection.**

Kernel K is a set of learnable filters and is small spatially compared to the image but extends through the full depth of the input image.

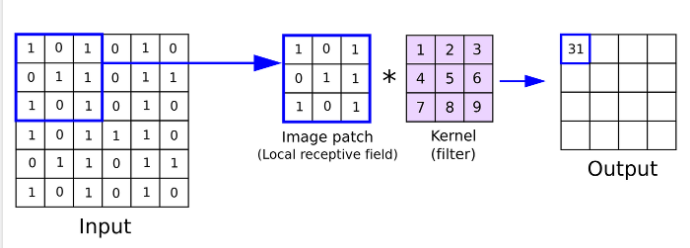
Kernel K, **which is a feature detector** is equivalent of the flashlight on image I, and we are trying to detect features and create multiple **feature maps** to help us identify or classify the image.

Kernel K can be define as a window by which we examine a subset of the image, and subsequently scans the entire image looking through this window creating feature maps.

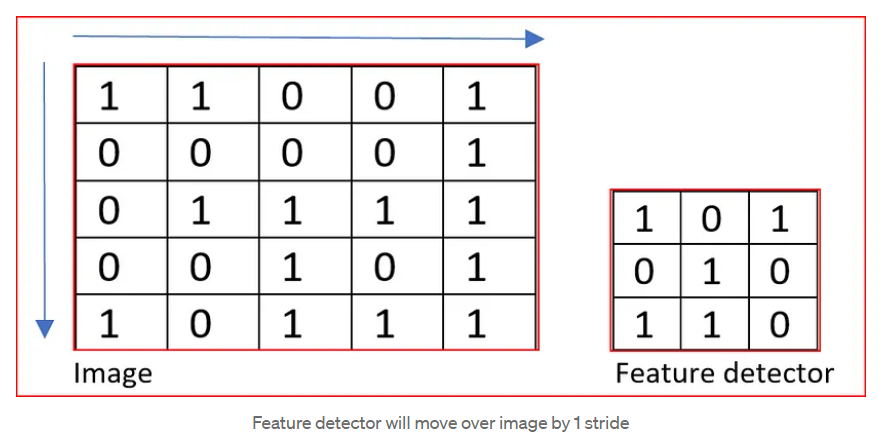
We have multiple feature detector to help with things like edge detection, identifying different shapes, bends or different colors etc.

# Feature maps can help to reduce the dimensionality of the input data, making it easier and faster to process and analyze. By using multiple layers of feature maps, a CNN can capture complex and hierarchical relationships between different features in the input data, leading to more accurate and robust predictions.

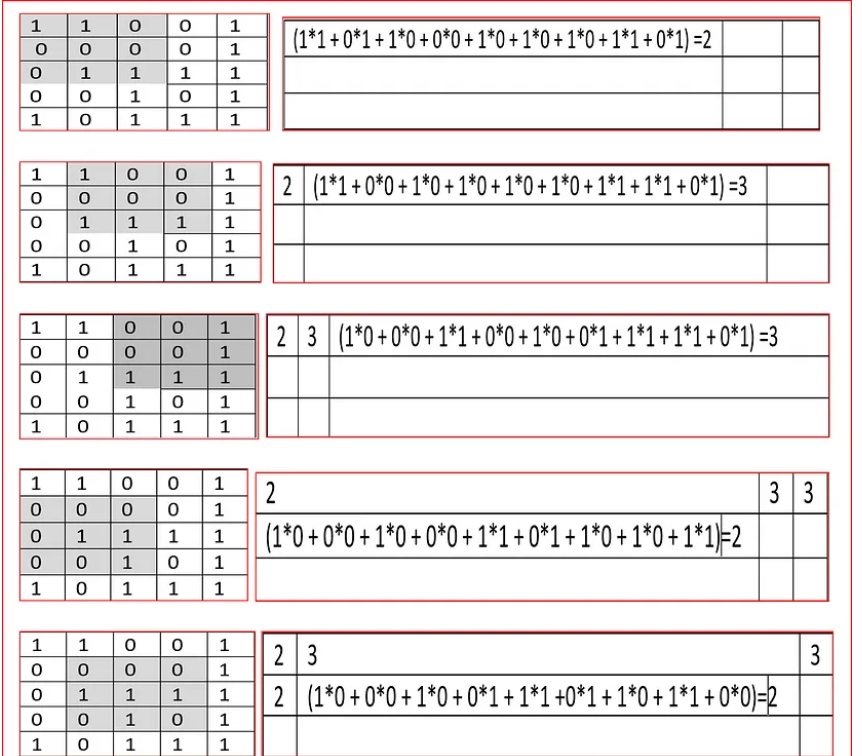
# The conjunction of multiple feature maps it is what creates a convolutional layer.

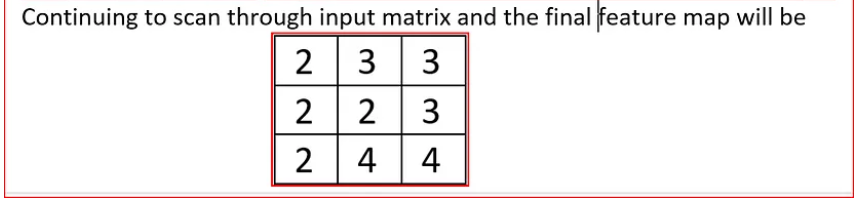
****

In the above image the output is a feature map of smaller dimension than the input created by the kernel, filter window. In put 6\*6. Output 4\*4.

****

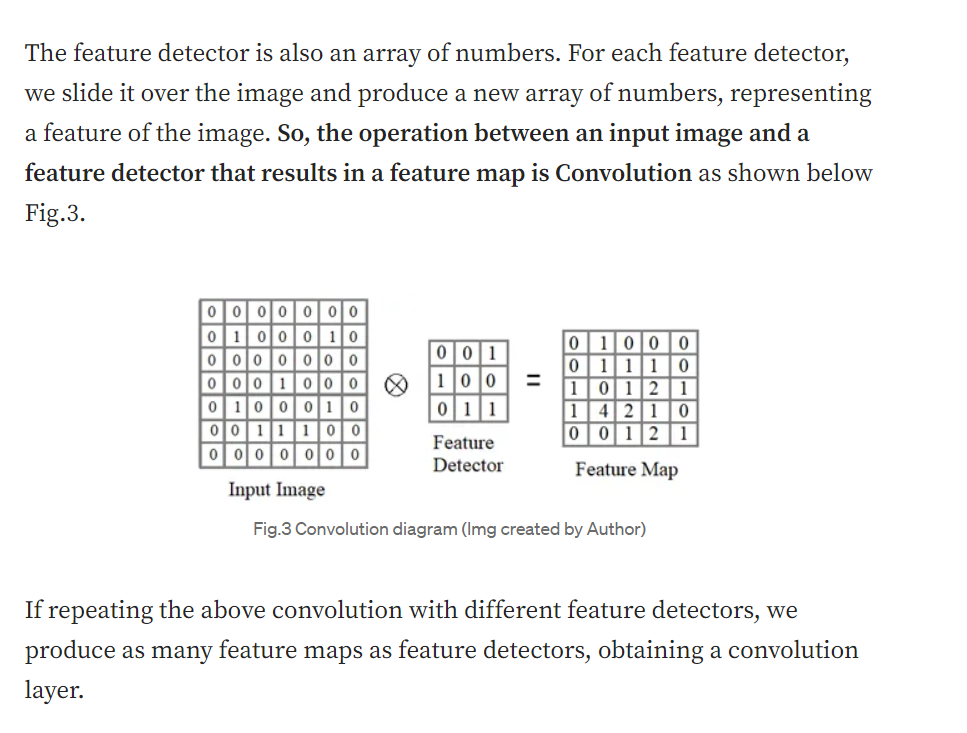
# What will be the dimension of the output matrix or feature map when I apply a feature detector, kernel filter, over an image?

****

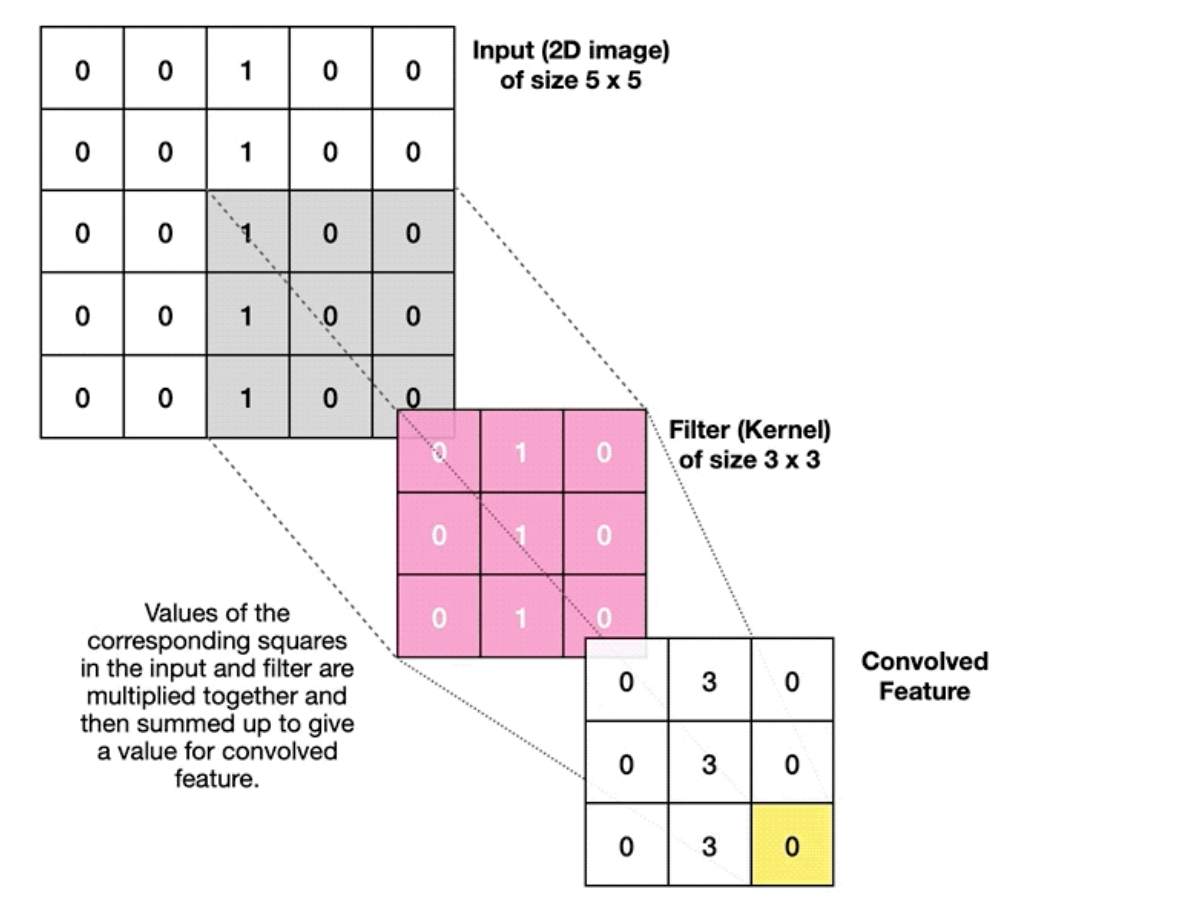
****

We see that 5 by 5 input image is reduced to 3 by 3 feature maps. The depth or channels remain the same as 3(RGB)

We have multiple feature Maps that help with things like edge detection, identifying different shapes, bends or different colors etc.



**We can also specify how many filters we want to have in the Convolutional layer. Having multiple filters lets us extract a broader range of features from the image.**



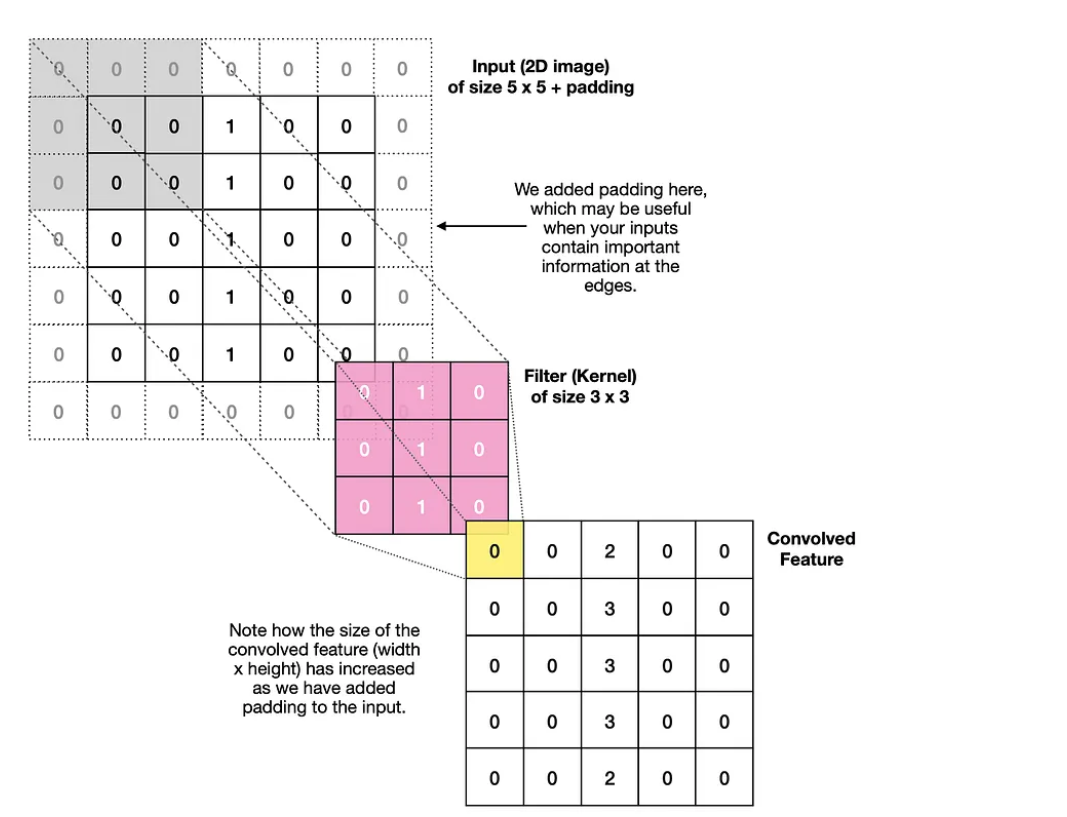
You will note that my **custom filter** has all 1’s down the middle column. This type of filter is designed to **identify vertical lines** in the input image as it gives a strong signal (high values) whenever vertical lines are present.

**It is important to note that we do not need to specify values for different filters manually. The creation of filters is handled automatically during the training of the Convolutional Neural Network. Although, we can tell the algorithm how many filters we want to have.**

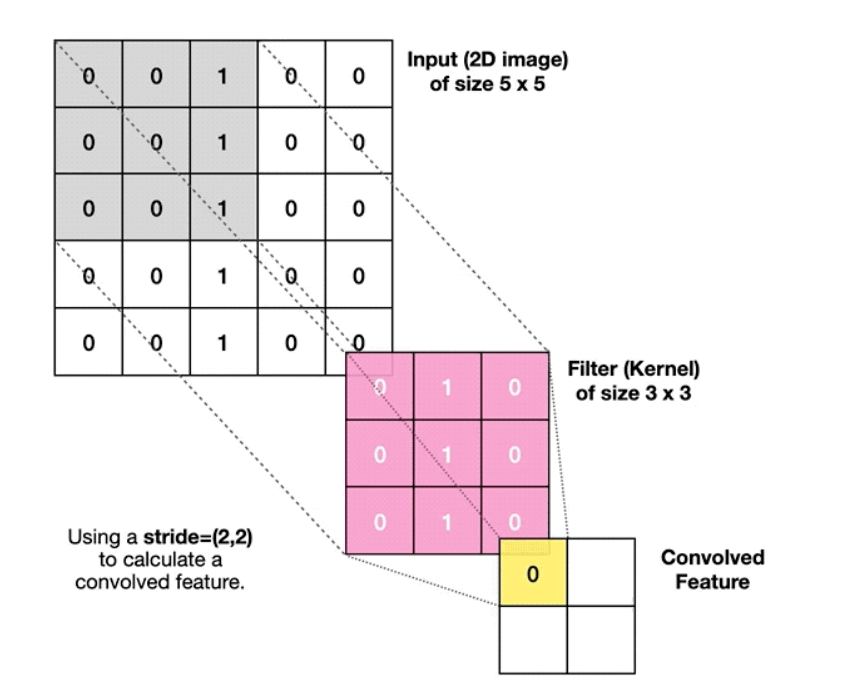
## **Additional options**

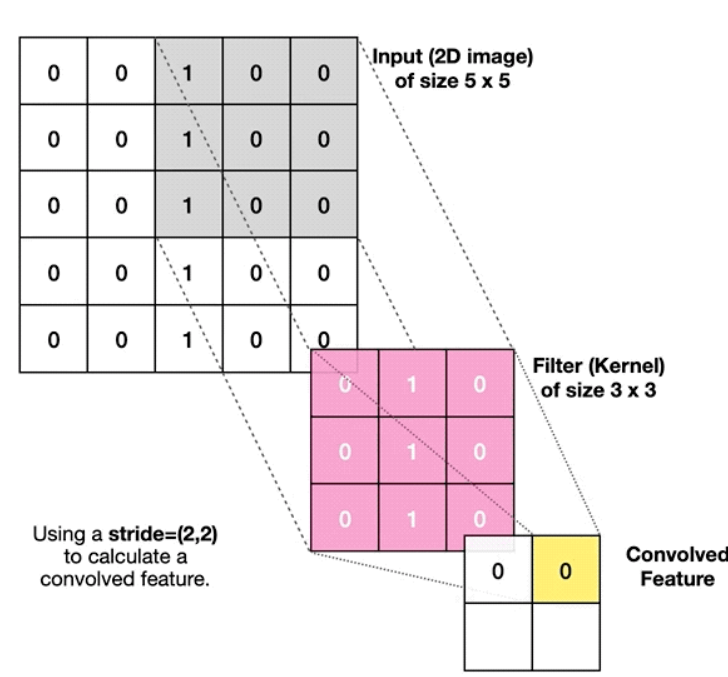
There are a couple more options for us to tweak when setting up a Convolutional layer:

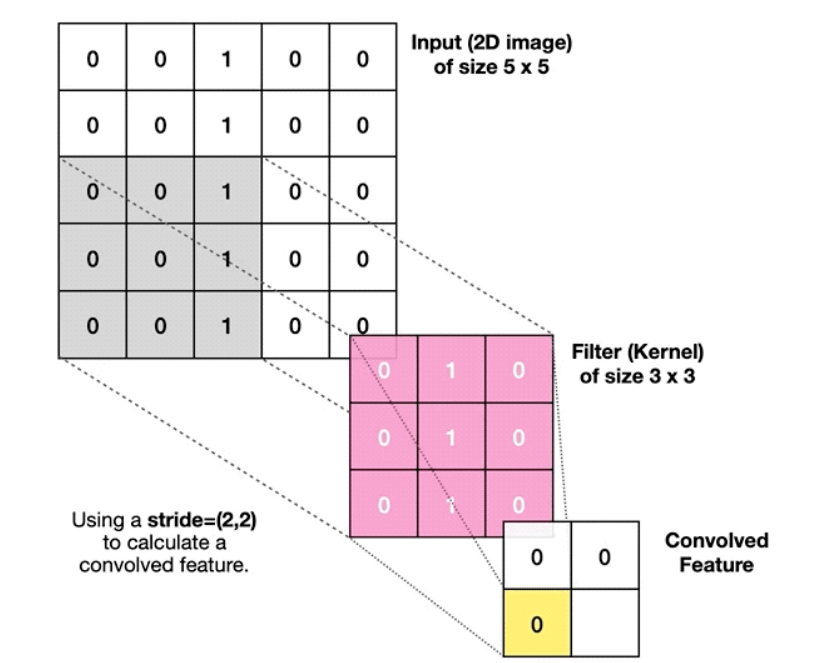
* **Padding**— in some scenarios, we may wish for the output to be the same size as the input. We can achieve that by adding some padding. At the same time, it may make it easier for the model to capture essential features residing at the edges of an image.

**Padding** can help in reducing the loss of information at the borders of the input feature map and can improve the performance of the model. However, it also increases the computational cost of the convolution operation. Overall, padding is an important technique in CNNs that helps in preserving the spatial dimensions of the feature maps and can improve the performance of the model.

* **Stride**— if we have large images, then we may want to use larger strides, i.e., shifting a filter by multiple pixels at a time. While it does help to reduce the size of the output, larger strides may result in some features being missed, like in the example below:

****

****

****

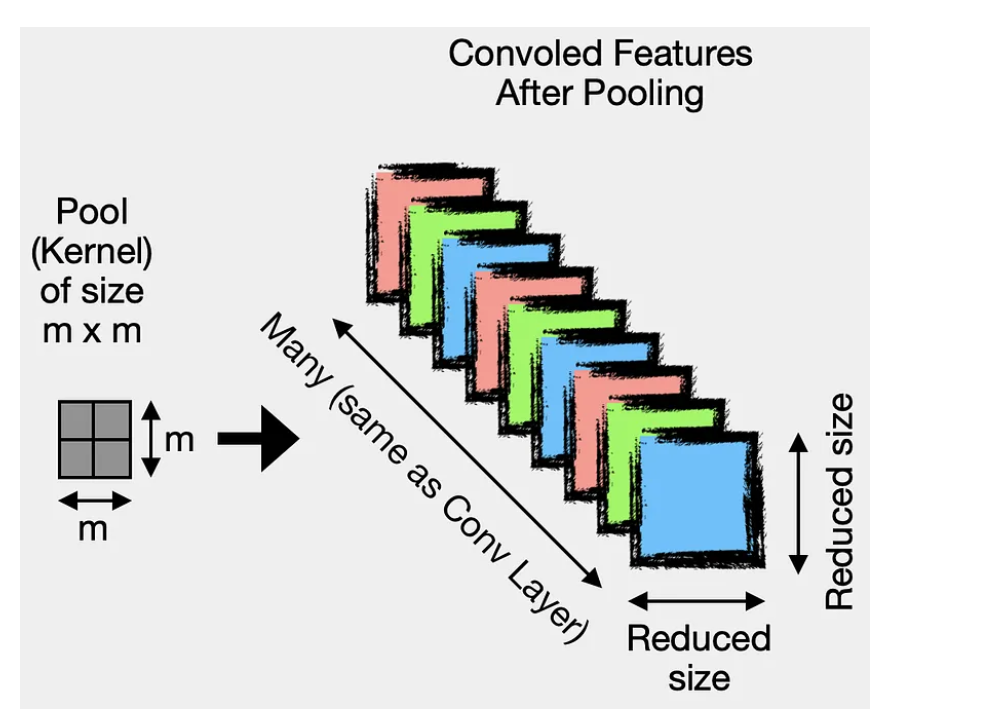
## **Multiple convolutional layers**

It is often beneficial to set up multiple Convolutional layers to improve the network. The benefits arise from subsequent Convolutional layers identifying extra complexity within the image.

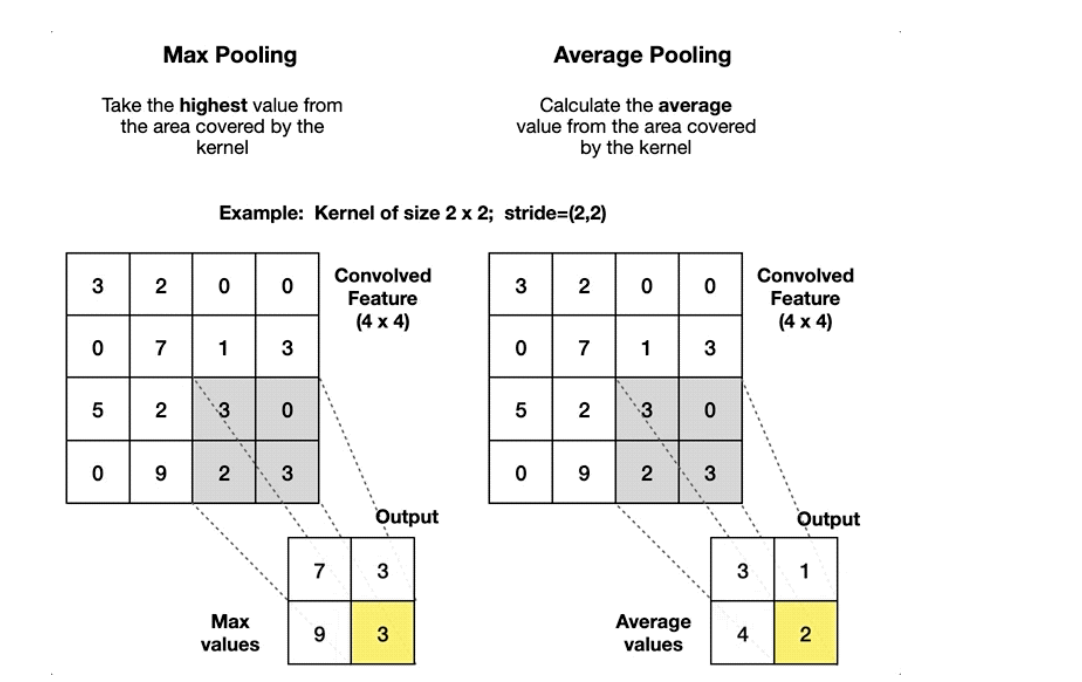
The first layer in a **Deep Convolutional Network (DCN)**tends to find **low-level** features (e.g., vertical, horizontal, diagonal lines…). Meanwhile, the deeper layers can identify **higher-level** characteristics, such as more complex shapes, representing real-world elements like eyes, nose, ears etc.

## **Pooling layer**

It is common to add a Pooling layer following a Convolutional layer. Its purpose is to **reduce the size of Convolved Features** **improving computational efficiency**. Also, it can help to de-noise the data by **keeping the strongest activations**.

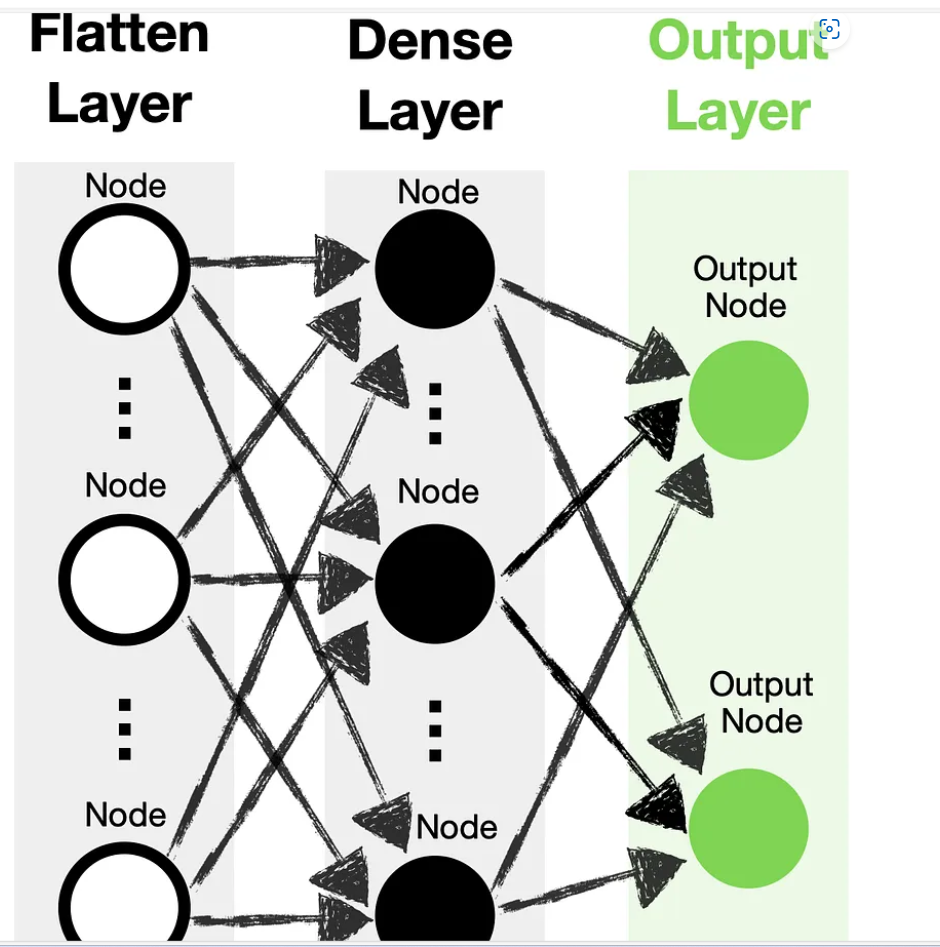


The Convolved Features are Feature Maps.



## **Flatten and Dense Layers**

Once we have finished deriving Convolved Features, we need to flatten them. This enables us to have a **one-dimensional input vector** and utilize a traditional Feed-Froward Network architecture. In the end, we train the network to find the optimum weights and biases, which enables us to classify images correctly.



**Depending on the size and complexity of your data, you may want to have multiple pairs of Convolutional and Pooling layers followed by multiple Dense Layers, making your network “Deep.”**