Convolutional Neural Networks(CNN)

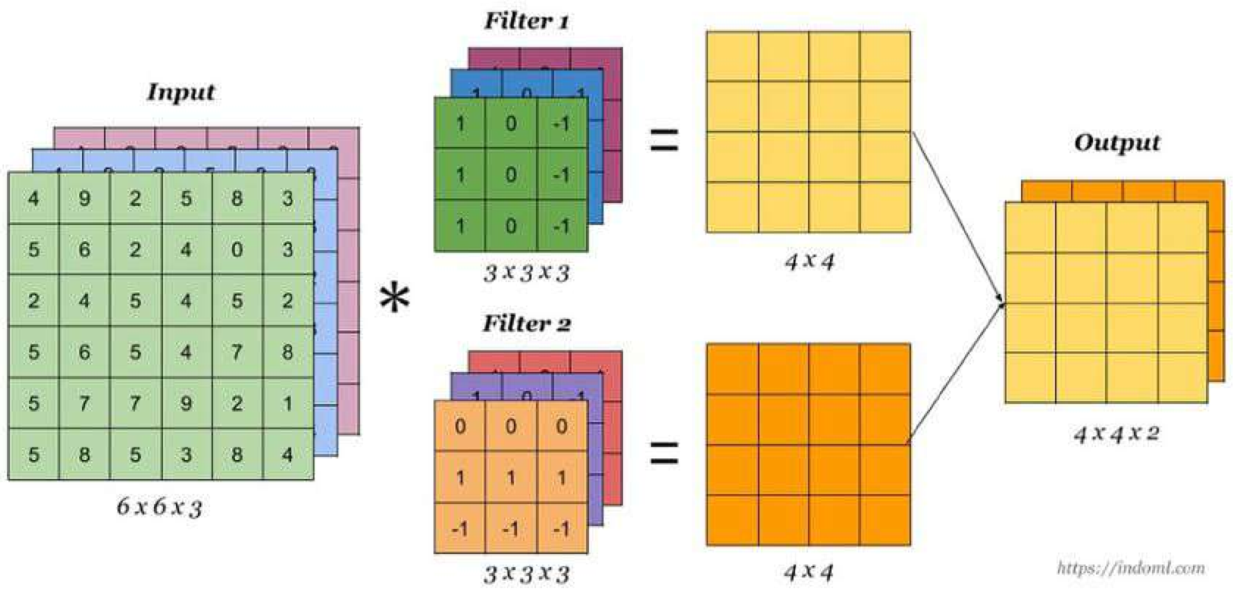
(DOWNLOAD THE FREE PDF)

Deep Learning a subset of Machine Learning which consists of algorithms that are inspired by the functioning of the human brain or the neural networks. These structures are called as Neural Networks. It teaches the computer to do what naturally comes to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Autoencoders, Recurrent Neural Networks (RNN) and Reinforcement Learning. But there has been one particular model that has contributed a lot in the field of computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNets.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analysing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term '**Convolution**" in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

Convolutional Neural Networks (CNN) Introduction



Why Convolutions

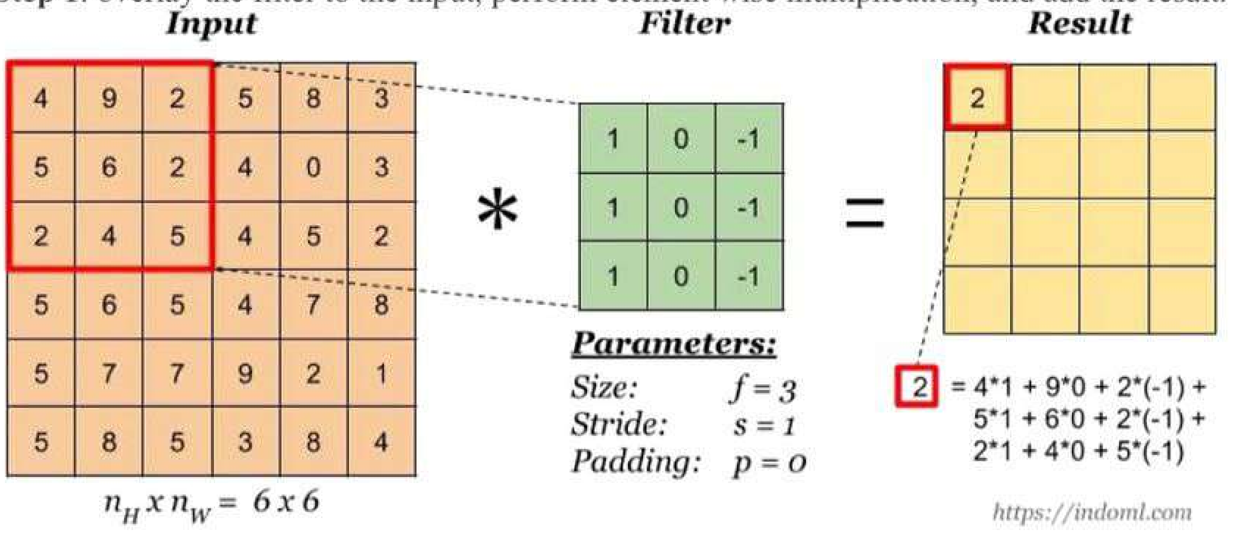
- **Parameter sharing**: a feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

- **Sparsity of connections**: in each layer, each output value depends only on small number of inputs.

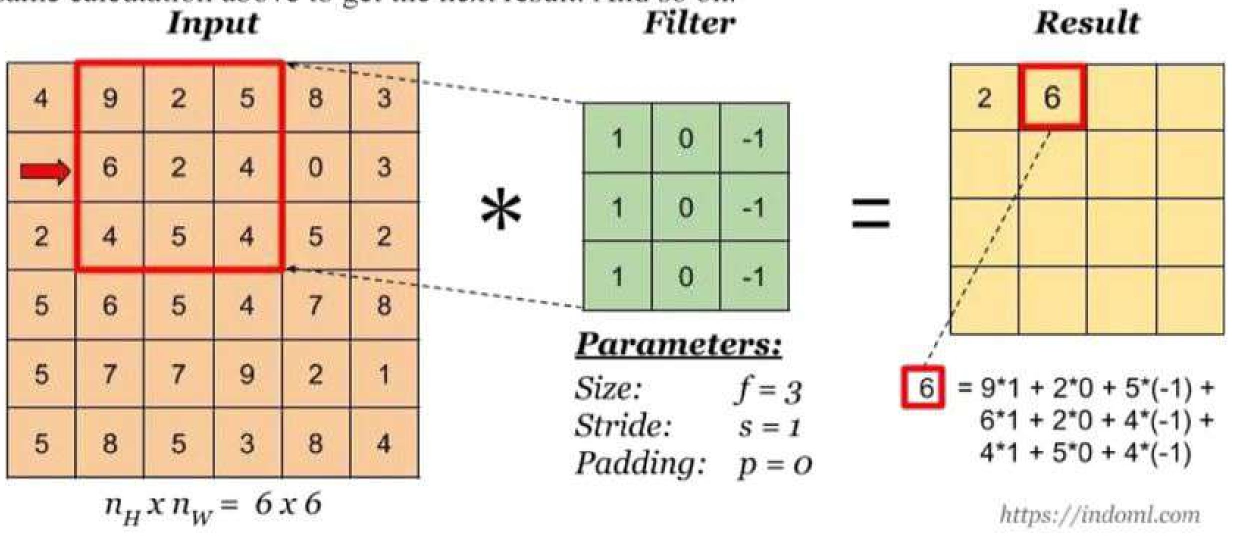
~

Basic Convolution Operation

Step 1: overlay the filter to the input, perform element wise multiplication, and add the result.



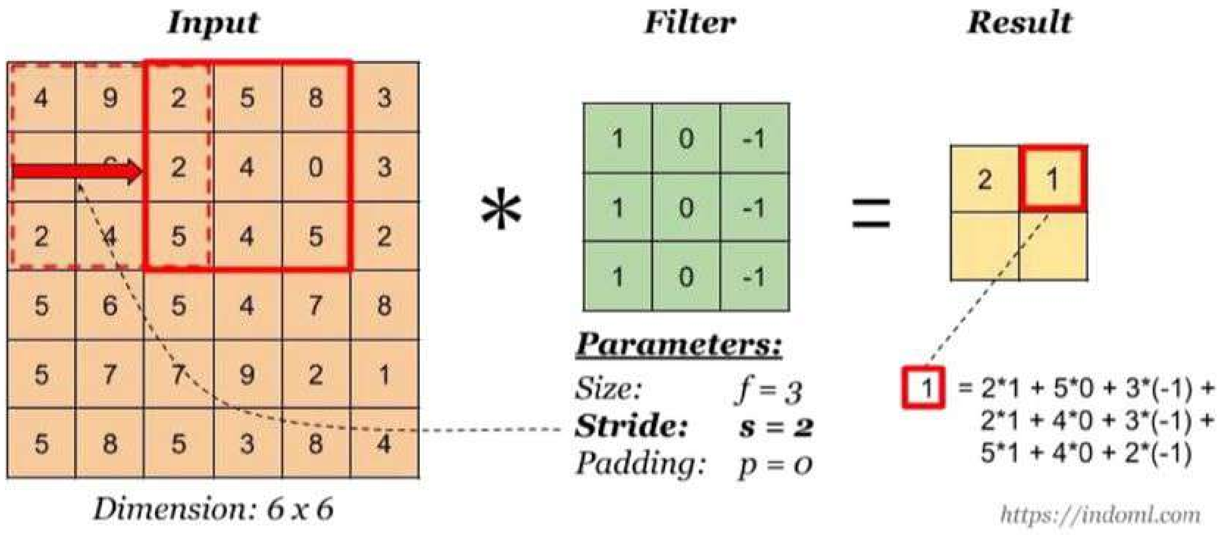
Step 2: move the overlay right one position (or according to the **stride** setting), and do the same calculation above to get the next result. And so on.



The total number of multiplications to calculate the result above is (4 x 4) x (3 x 3) = 144.

Stride

Stride governs how many cells the filter is moved in the input to calculate the next cell in the result.



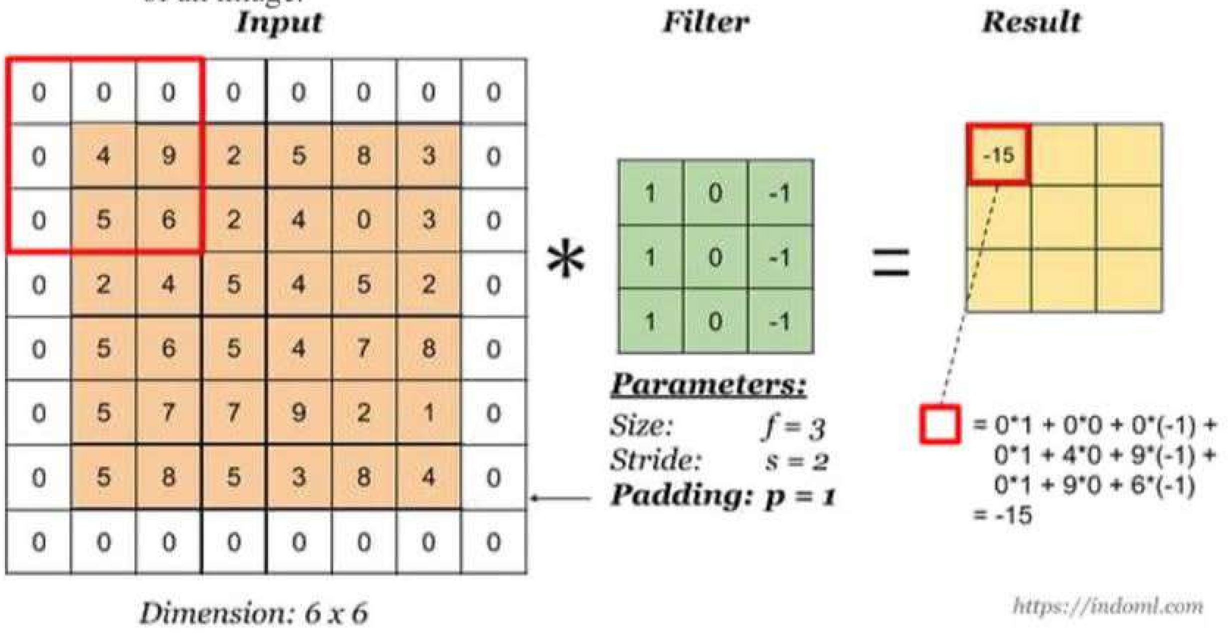
Filter with stride (s) = 2

The total number of multiplications to calculate the result above is (2 x 2) x (3 x 3) = 36.

Padding

Padding has the following benefits:

1. It allows us to use a CONV layer without necessarily shrinking the height and width of the volumes. This is important for building deeper networks, since otherwise the height/width would shrink as we go to deeper layers.
2. It helps us keep more of the information at the border of an image. Without padding, very few values at the next layer would be affected by pixels as the edges of an image.



Notice the dimension of the result has changed due to padding. See the following section on how to calculate output dimension.

Some padding terminologies:

- "**valid**" padding: no padding
- "**same**" padding: padding so that the output dimension is the same as the input

Calculating the Output Dimension

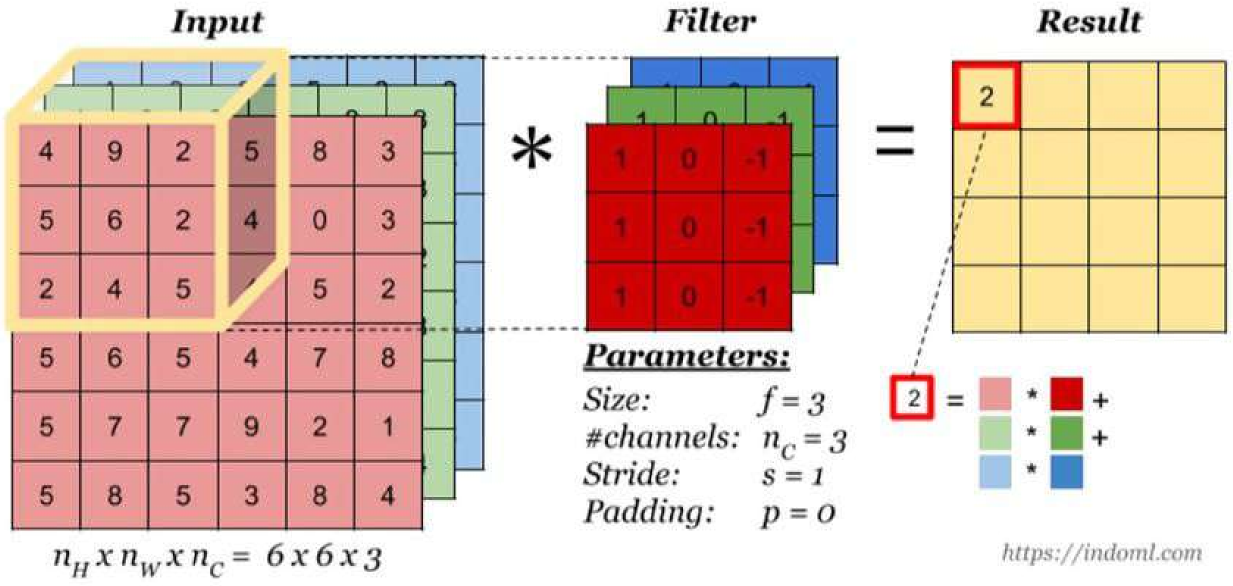
The output dimension is calculated with the following formula:

$$n^{[l]} = \lfloor \frac{n^{[l-1]} + 2p^{[l-1]} - f^{[l]}}{s^{[l]}} + 1 \rfloor$$

where the ⌊ ⌋ symbols denote *math.floor()* operation.

Convolution Operation on Volume

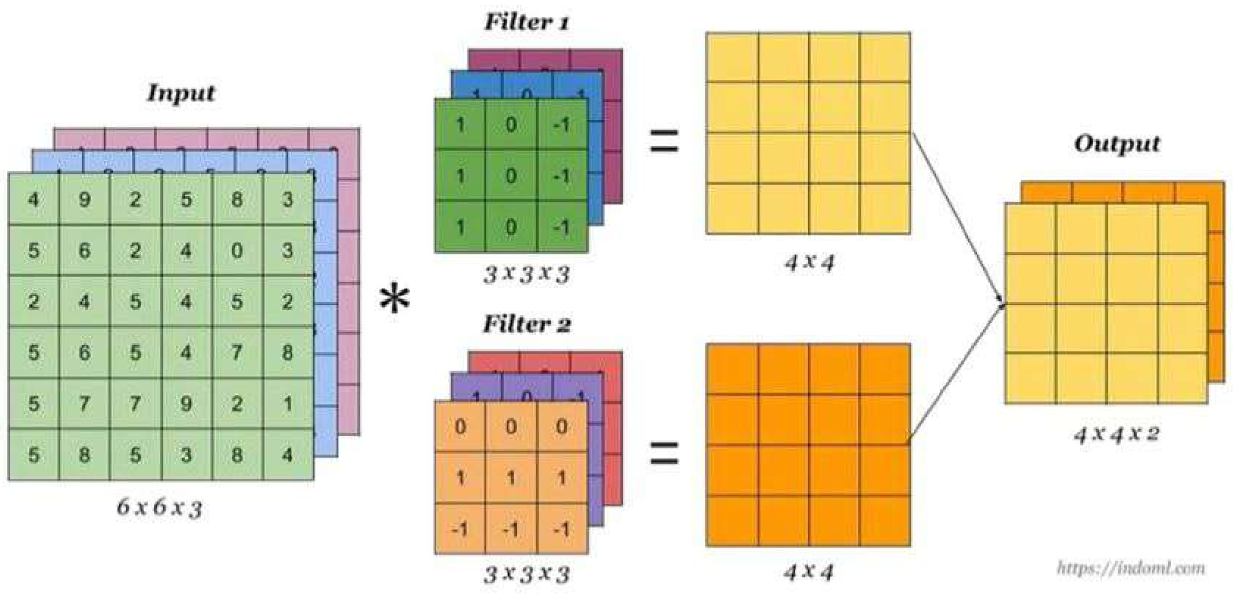
When the input has more than one channels (e.g. an RGB image), the filter should have matching number of channels. To calculate one output cell, perform convolution on each matching channel, then add the result together.



The total number of multiplications to calculate the result is (4 x 4) x (3 x 3 x 3) = 432.

Convolution Operation with Multiple Filters

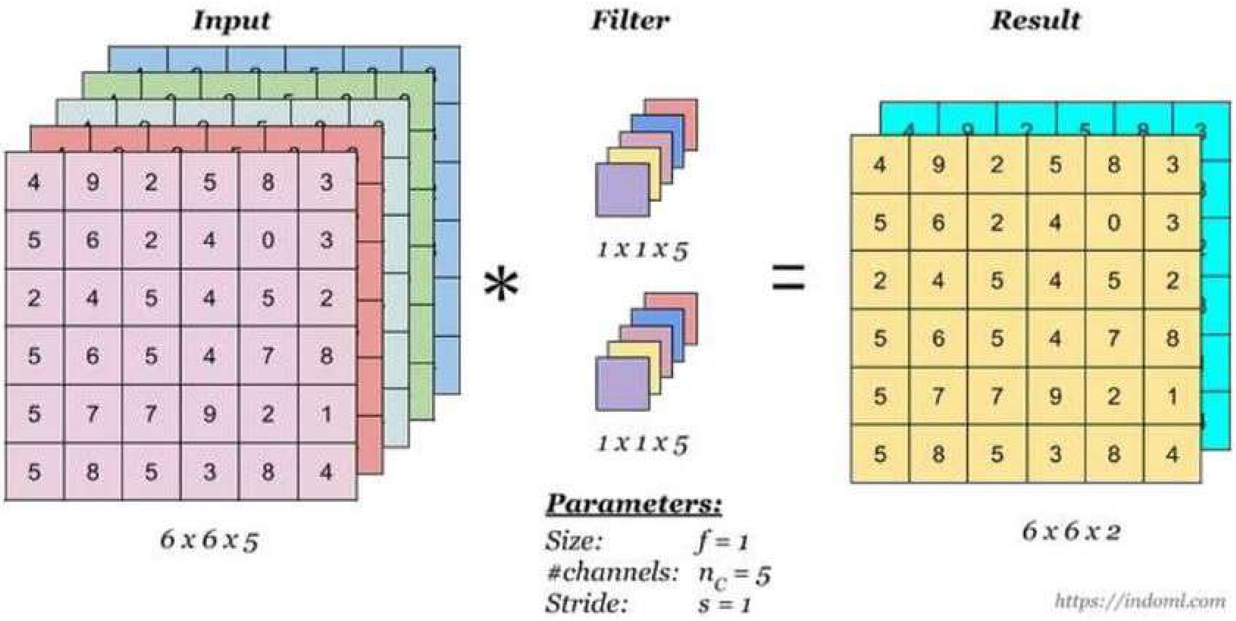
Multiple filters can be used in a convolution layer to detect multiple features. The output of the layer then will have the same number of channels as the number of filters in the layer.



The total number of multiplications to calculate the result is (4 x 4 x 2) x (3 x 3 x 3) = 864.

1 x 1 Convolution

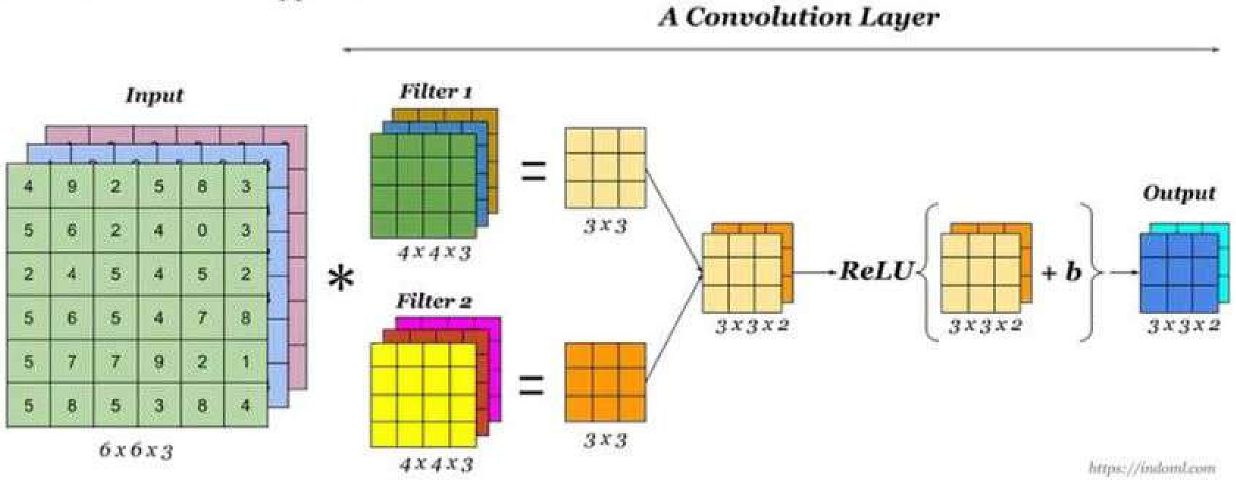
This is convolution with 1 x 1 filter. The effect is to flatten or "merge" channels together, which can save computations later in the network:



~

One Convolution Layer

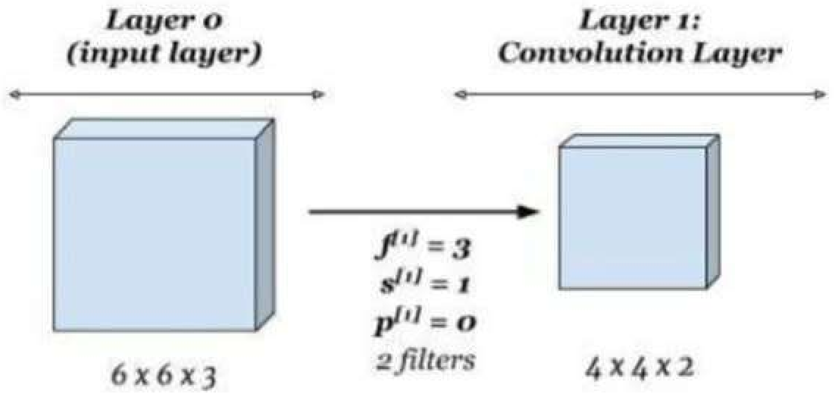
Finally, to make up a convolution layer, a bias (ϵ R) is added and an activation function such as **ReLU** or **tanh** is applied.



~

Shorthand Representation

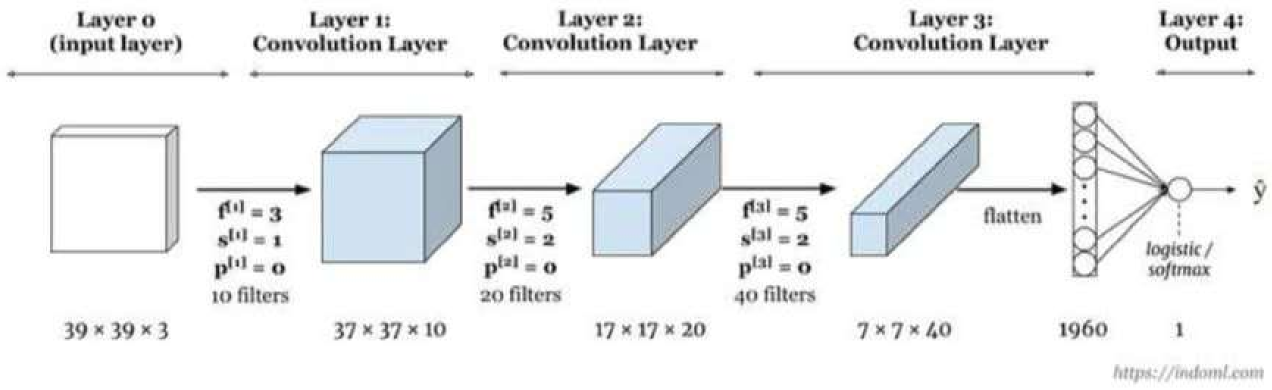
This simpler representation will be used from now on to represent one convolutional layer:



~

Sample Complete Network

This is a sample network with three convolution layers. At the end of the network, the output of the convolution layer is flattened and is connected to a logistic regression or a softmax output layer.

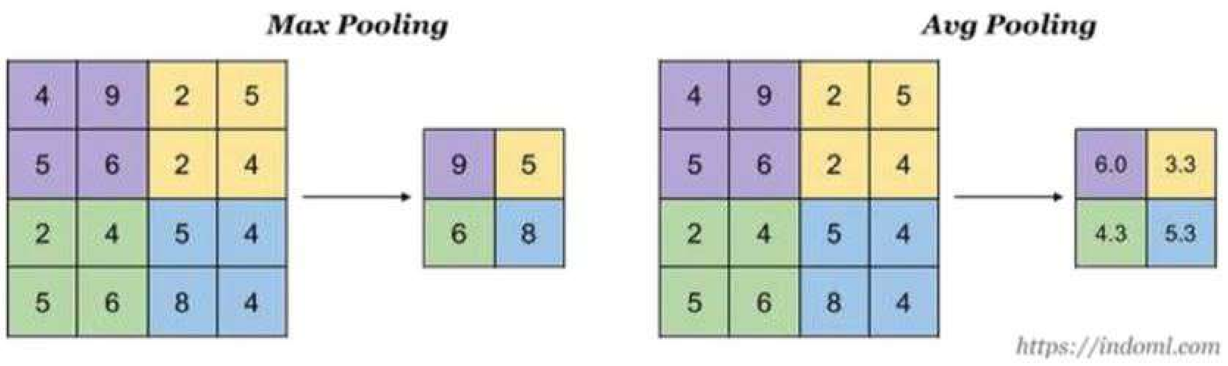


~

Pooling Layer

Pooling layer is used to reduce the size of the representations and to speed up calculations, as well as to make some of the features it detects a bit more robust.

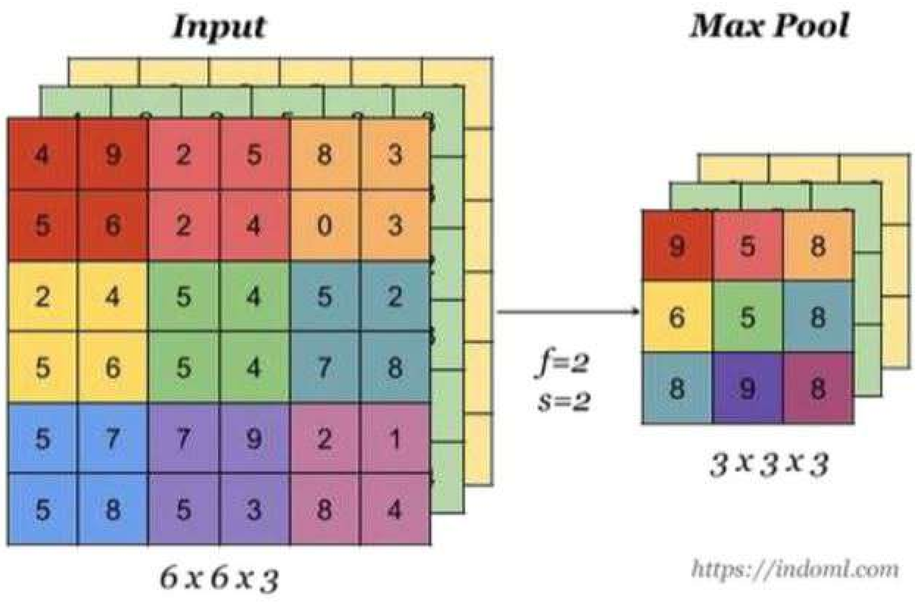
Sample types of pooling are **max pooling** and **avg pooling**, but these days max pooling is more common.



Interesting properties of pooling layer:

- it has hyper-parameters:
  - **size** (*f*)
  - **stride** (*s*)
  - **type** (max or avg)
- but it doesn't have parameter; there's nothing for gradient descent to learn

When done on input with multiple channels, pooling reduces the height and width (nW and nH) but keeps nC unchanged:

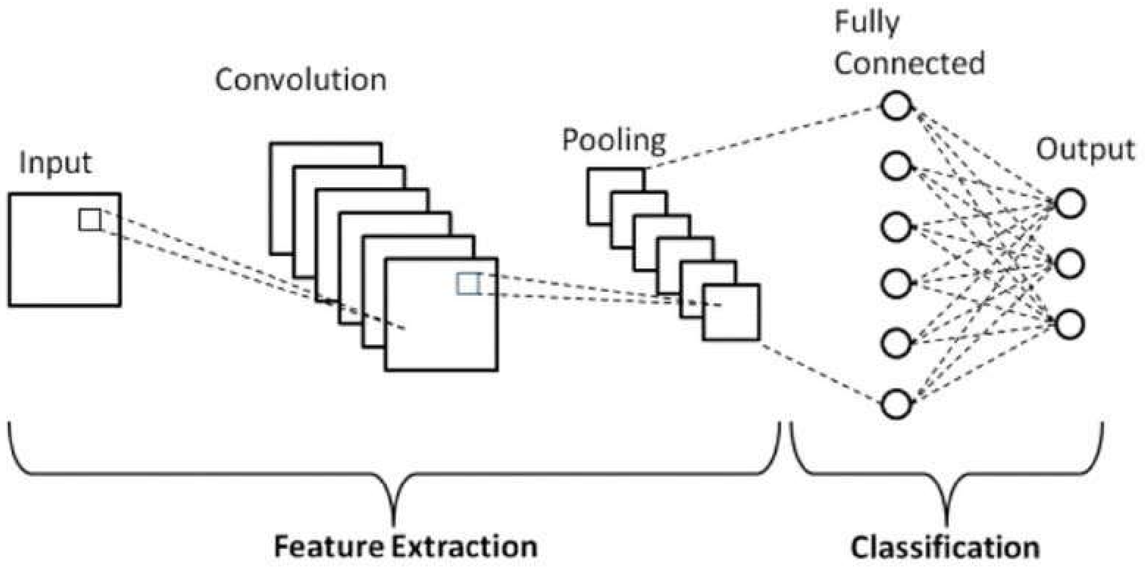


Basic Architecture of CNN

There are two main parts to a CNN architecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction

- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.



Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

1. **Convolutional Layer**

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.
In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

4. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On

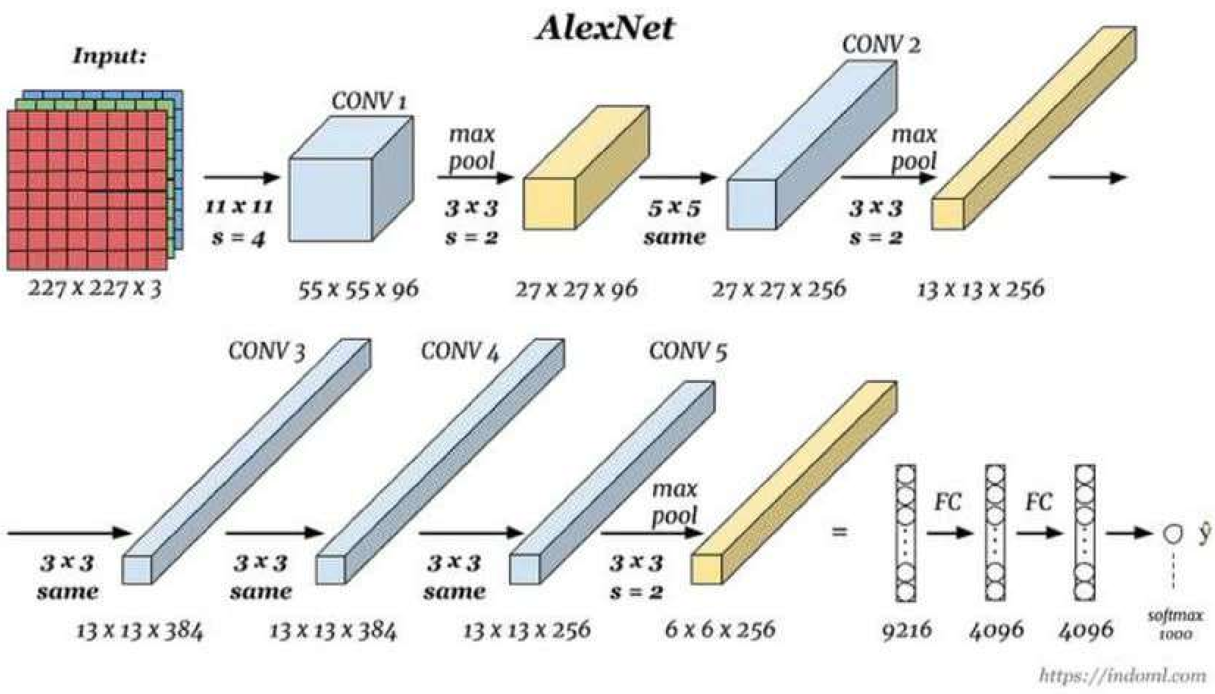


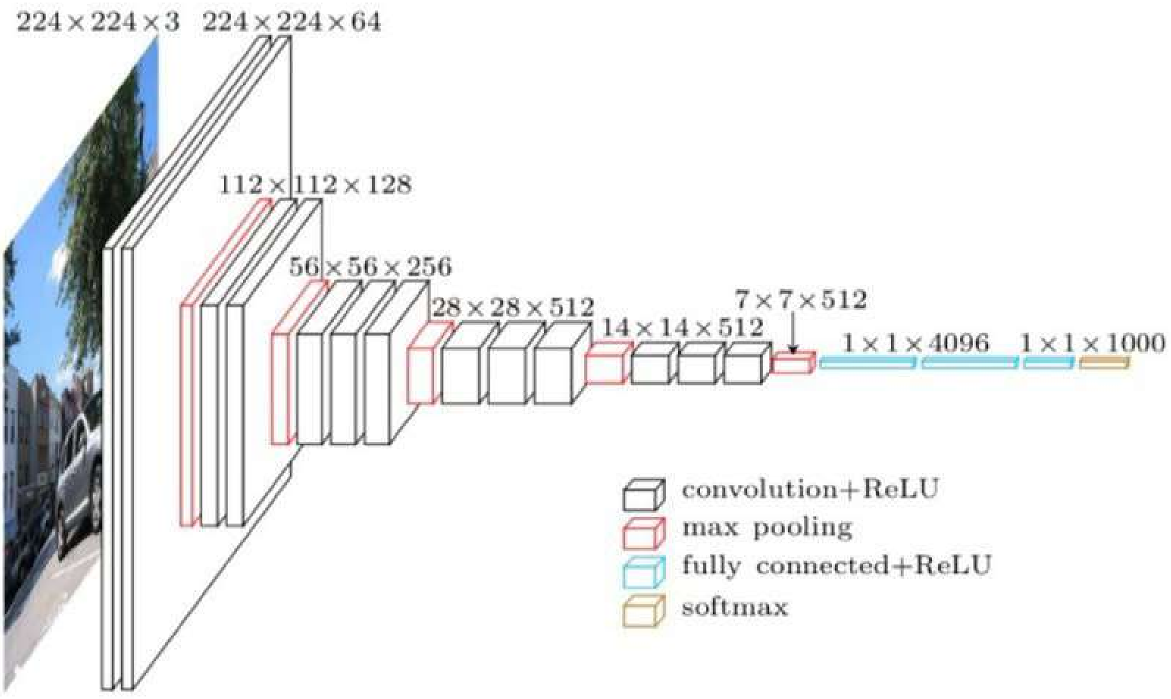
Figure 1: Architecture of AlexNet

- Number of parameters: ~ 60 millions.
- AlexNet and won the most difficult ImageNet challenge for visual object recognition called the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.
- AlexNet achieved state-of-the-art recognition accuracy against all the traditional machine learning and computer vision approaches.
- The architecture of AlexNet is shown in Figure 1. The first convolutional layer performs convolution and max pooling with Local Response Normalization (LRN) where 96 different receptive filters are used that are 11×11 in size.
- The max pooling operations are performed with 3×3 filters with a stride size of 2. The same operations are performed in the second layer with 5×5 filters.
- 3×3 filters are used in the third, fourth, and fifth convolutional layers with 384, 384, and 296 feature maps respectively.
- Two fully connected (FC) layers are used with dropout followed by a Softmax layer at the end. Two networks with similar structure and the same number of feature maps are trained in parallel for this model.

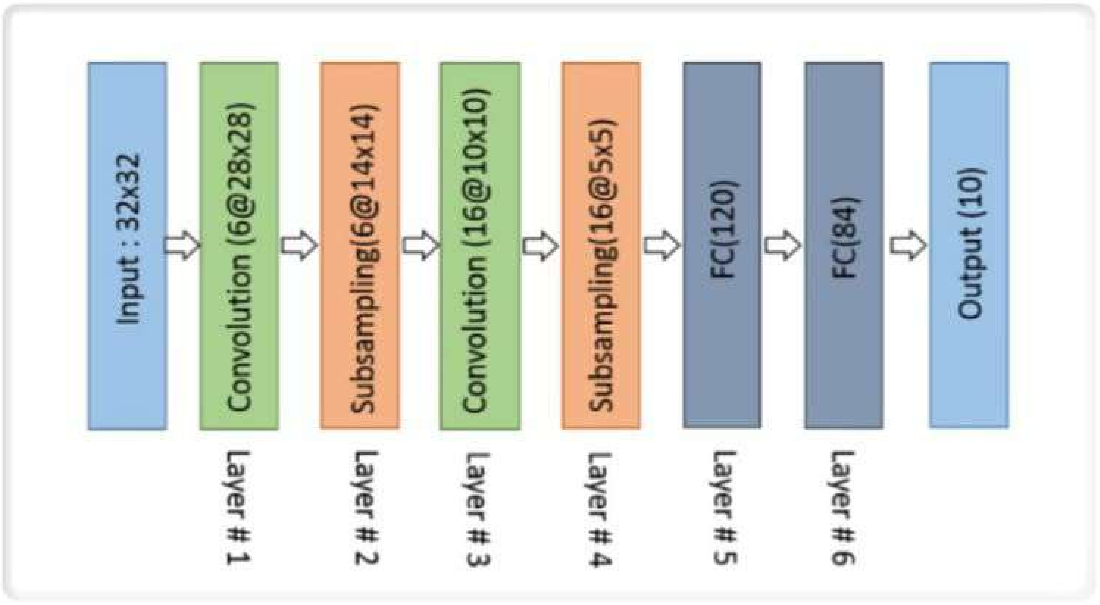
- AlexNet has 3 convolution layers and 2 fully connected layers. When processing the ImageNet dataset. The total number of weights and MACs for the whole network are 61M and 724M respectively.

Classic Network: VGG-16

VGG-16, The number 16 refers to the fact that the network has 16 trainable layers (i.e., layers that have weights.



- Number of parameters: ~ 138 millions.
- The strength is in the simplicity: the dimension is halved and the depth is increased on every step (or stack of layers)



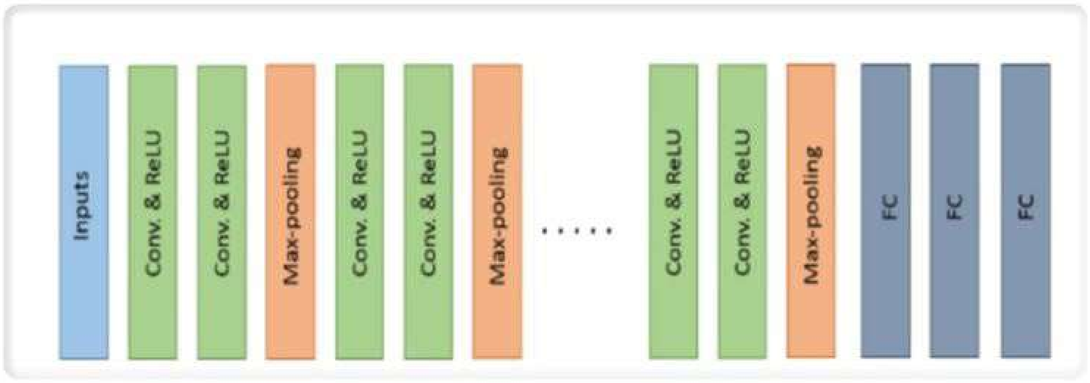


Figure 2: Basic building blocks of VGG

- The VGG architecture consists of two convolutional layers both of which use the ReLU activation function. Following the activation function is a single max pooling layer and several fully connected layers also using a ReLU activation function.
- The final layer of the model is a Softmax layer for classification. In VGG-E the convolution filter size is changed to a 3×3 filter with a stride of 2.
- Three VGG-E models, VGG-11, VGG-16, and VGG-19; were proposed the models had 11,16, and 19 layers respectively. All versions of the VGG-E models ended the same with three fully connected layers.
- However, the number of convolution layers varied VGG-11 contained 8 convolution layers, VGG-16 had 13 convolution layers, and VGG-19 had 16 convolution layers. VGG-19, the most computational expensive model, contained 138M weights and had 15.5M MACs.