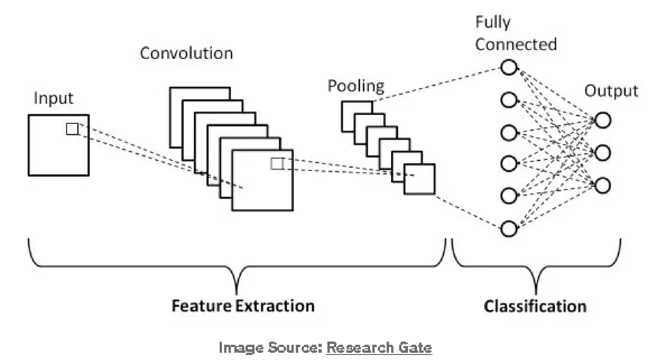
Convolutional Neural Network (CNN)

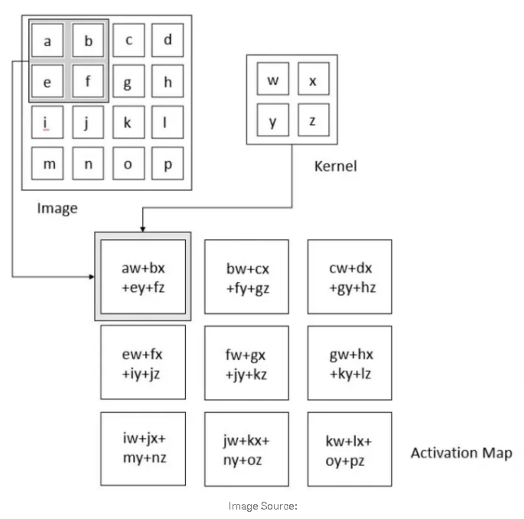
Aim: Understand how the fundamental components of CNN interplay with each other at a layer level for image classification

CNN was developed to reach the two milestones in image processing: i) cutting down the number of parameters learnt by reducing dimensions ii) retaining the spatial information of an image.

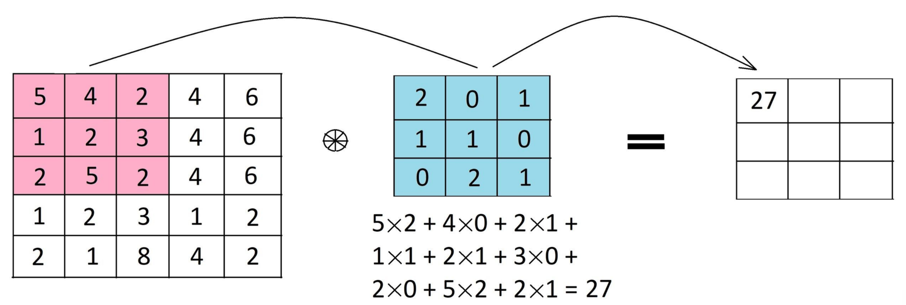
In the architecture of a CNN, there are three fundamental layers of components: convolutional layers, pooling layers, and fully-connected layers.



1. The Convolution Layer



-> The convolution layer plays a critical role in preserving the pixel-spatial relationship. The layer computes a dot product between filter and image-pixel values



-> How this dot-product calculation is performed is through a kernel, a tile-like unit that slides over elements, multiplying each element from one tensor (image pixel) to its corresponding element in itself. The product of this process is an activation map/ a feature map with a matrix of single output values from summing up dot-product calculations.

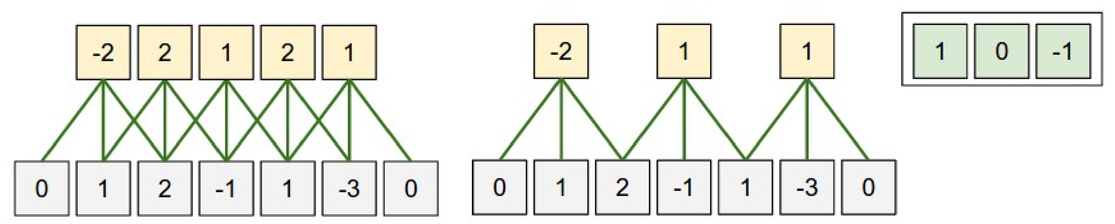
-> There are four important hyperparameters that govern the behaviour of a covolutional layer: i) kernel size ii) stride iii) zero-padding iv) depth

i) The size of a kernel determines how much of an input data will be filtered through an input tensor by a square unit

ii) A filter moving by one pixel at a time is often referred to as stride being 1. This striding movement produces output volumes.

iii) zero-padding involves enveloping the outer edges of an input data with zeros. The size of this zero-padding becomes a parameter

iv) The depth of an output volume is equivalent to the number of filters used and a depth column (fibre) is a set of neurons that rootle through the same region of an input.

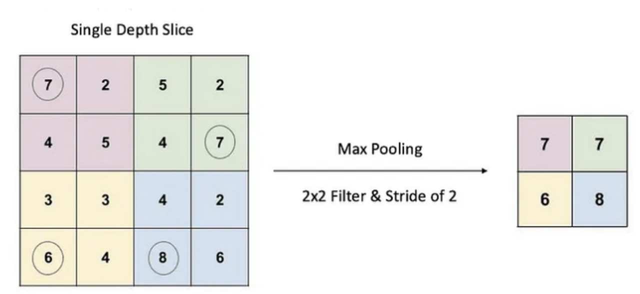


Calculating the spatial size of an output volume: (w-f+2p)/s+1
w=the size of an input volume,
f = the receptive field size of a convolutional layer neurons,
s = the size of a stride, **p**= the size of a zero-padding

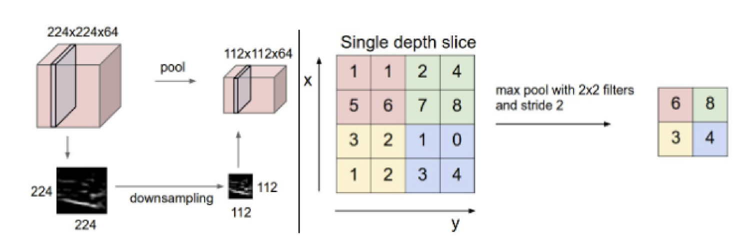
The illustration on the left shows a one-dimensional spatial arrangement with one neuron of a receptive field size of F=3, w=5, and p=1. For left, the neuron striding across the input will have s=1 and the output size will be (5-3+2)/1+1 = 5. As for the right whose s=2, (5-3+2)/2+1=3

2. The Pooling Layer

-> The pooling layer reduces the dimensionality of an input by chopping down the number of parameters.



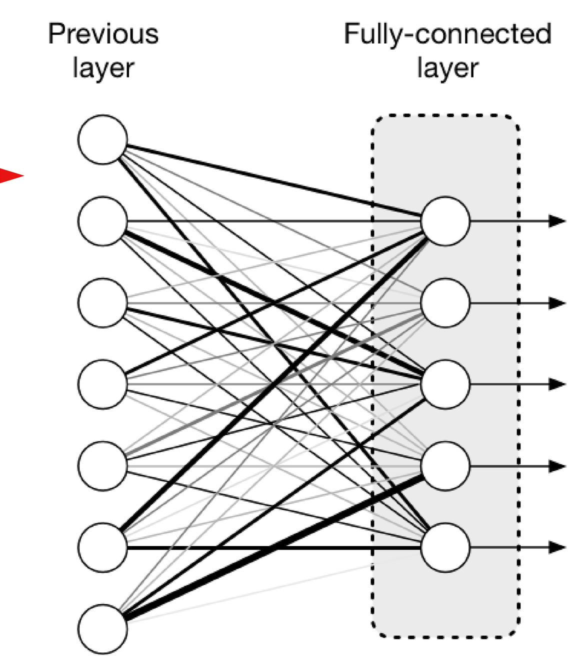
-> Although this pooling operation involves a sliding of a filter across an input, this pooling filter does not possess any weight. However, the filter generates an array of ouput through aggregation.



For instance, the graphic illustration on the left shows an example of 'max pooling', which select an output pixel with the maximum value that would be sent to an output array.

Average pooling is another example of a pooling which calculates the average value of pixels within a receptive field.

3. The Fully-connected Layer



-> The fully-connected layer transforms high-level features contained in a flattened feature map from the preceding pooling layer into a final output. The specific kind of this final output may vary depending on the task required.

An overview will be as such: suppose that there is a simple convolutional net for CIFAR-10 classificiation with the architecture [INPUT-CONV-POOL-FC]

INPUT [32x32x3] holds the raw pixel values. Width=32, height=32, RGB= 3

The CONVlayer will compute the output of neurons by calculting the dot-product between their weights and the corresponding pixels in the kernel. If there are 12 filters, then the volume will be [32x32x12]

** RELU may be performed i.e max(0,x) to solve the vanishing gradient issue that emerges when calculating the linear relationship between input and output by incorporating non-linearity. The result of this RELU layer may be [16x16x12]

The fully-connected (FC) layer will transform / normalise a vector of elements into a probability distribution, ranging between 0 and 1. In the case of image classification, a particular element of the vector producted by this softmax function will correspond to the probability of the input belonging to its corresponding class i.e) the ith element belonging to its ith class

Source:
1) CS231n Convolutional Neural Networks for Visual Recognition.
2) Understanding Convolutional Neural Networks 🧠: A Beginner's Journey into the Architecture 🚀 | by Afaque Umer | CodeX | Medium
3) neural networks - What is the use of softmax function in a CNN? - Artificial Intelligence Stack Exchange