**Introduction to Convolution Neural Network**

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data. When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network, there are three types of layers:

* **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
* **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.
* **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

**Convolution Neural Network**

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

**CNN architecture**

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

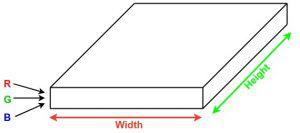


**Simple CNN architecture**

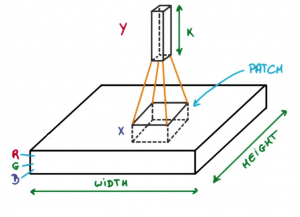
The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

**How Convolutional Layers works?**

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called **Convolution**. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.



**Image source: Deep Learning Udacity**

Now let’s talk about a bit of mathematics that is involved in the whole convolution process.

Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).

For example, if we have to run convolution on an image with dimensions **34x34x3**. The possible size of filters can be **axax3**, where ‘a’ can be anything like 3, 5, or 7 but smaller as compared to the image dimension.

During the forward pass, we slide each filter across the whole input volume step by step where each step is called **stride** (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.

As we slide our filters we’ll get a 2-D output for each filter and we’ll stack them together as a result, we’ll get output volume having a depth equal to the number of filters. The network will learn all the filters.

**Layers used to build ConvNets**

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

**Types of layers:** datasets

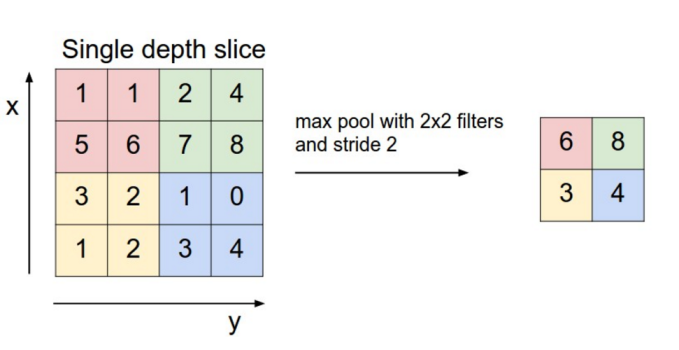
Let’s take an example by running a covnets on of image of dimension 32 x 32 x 3.

**Input Layers:** It’s the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

**Convolutional Layers:**This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we’ll get an output volume of dimension 32 x 32 x 12.

**Activation Layer:**By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are **RELU**: max(0, x),  **Tanh**, **Leaky RELU**, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.

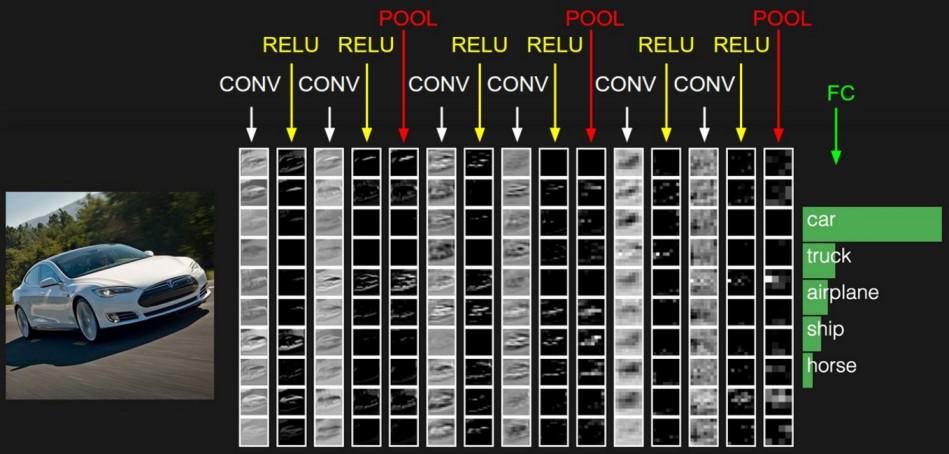
**Pooling layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.



**Image source: cs231n.stanford.edu**

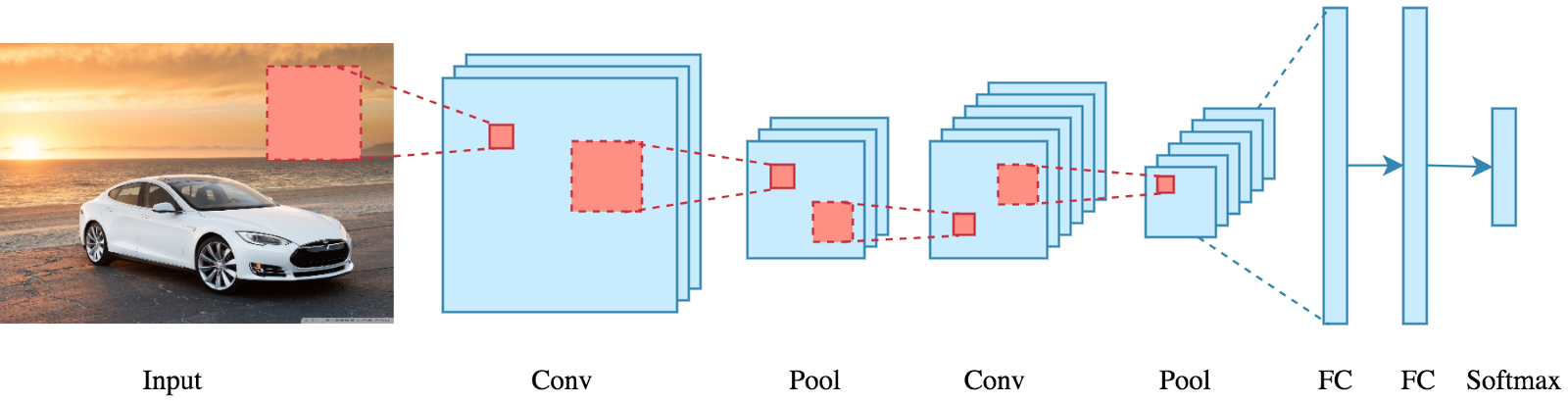
**Flattening:**The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

**Fully Connected Layers:**It takes the input from the previous layer and computes the final classification or regression task.



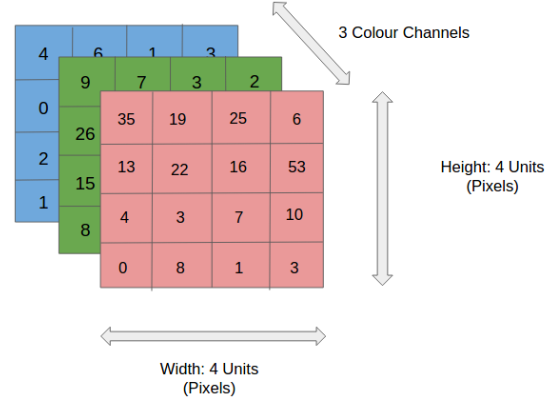
**Image source: cs231n.stanford.edu**

**Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

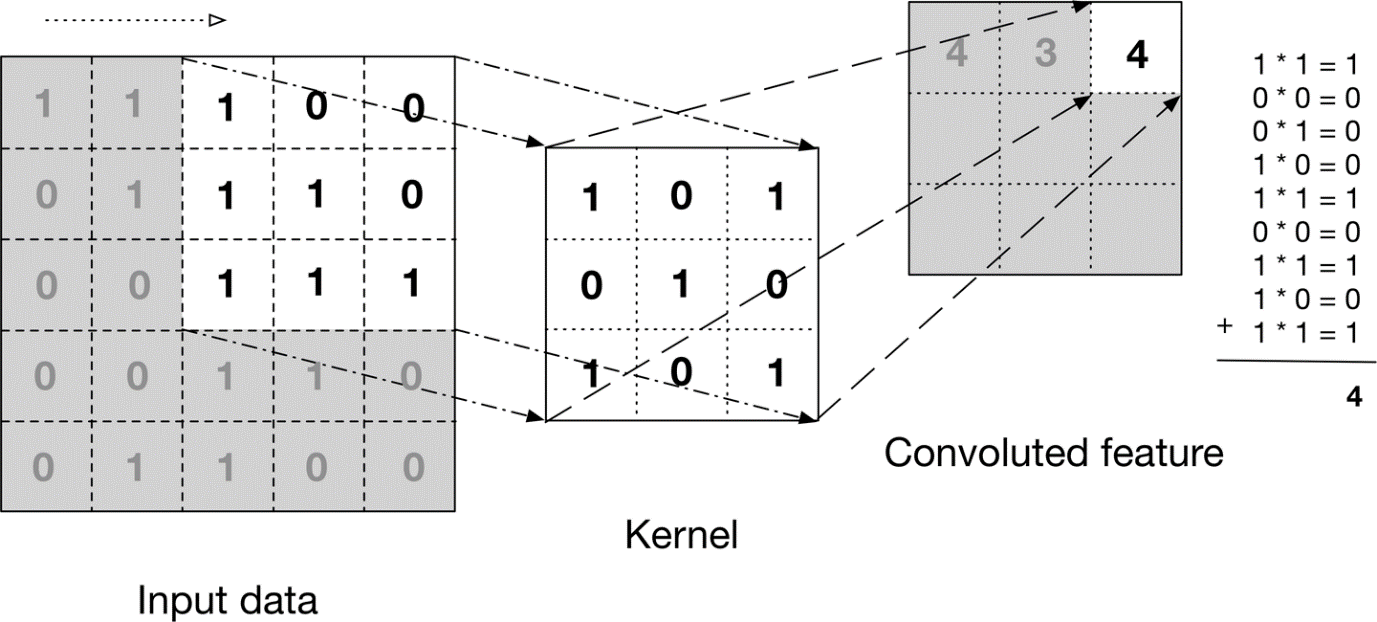


**How Does CNN work?**

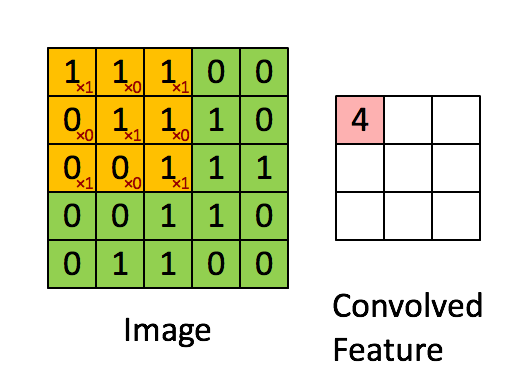
Before we go to the working of Convolutional neural networks (CNN), let’s cover the basics, such as what an image is and how it is represented. An RGB image is nothing but a matrix of pixel values having three planes whereas a grayscale image is the same but it has a single plane. Take a look at this image to understand more.



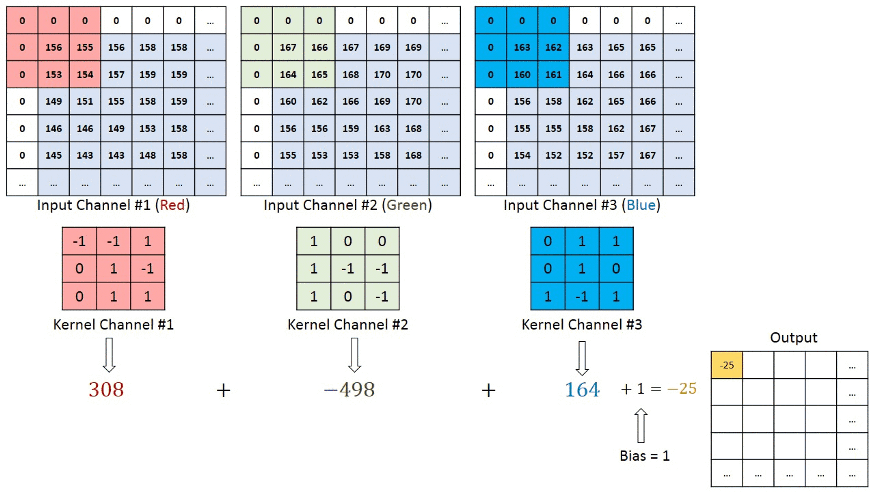
**For simplicity, let’s stick with grayscale images as we try to understand how CNNs work.**



The above image shows what a convolution is. We take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.



**In the case of RGB color, channel take a look at this animation to understand its working**

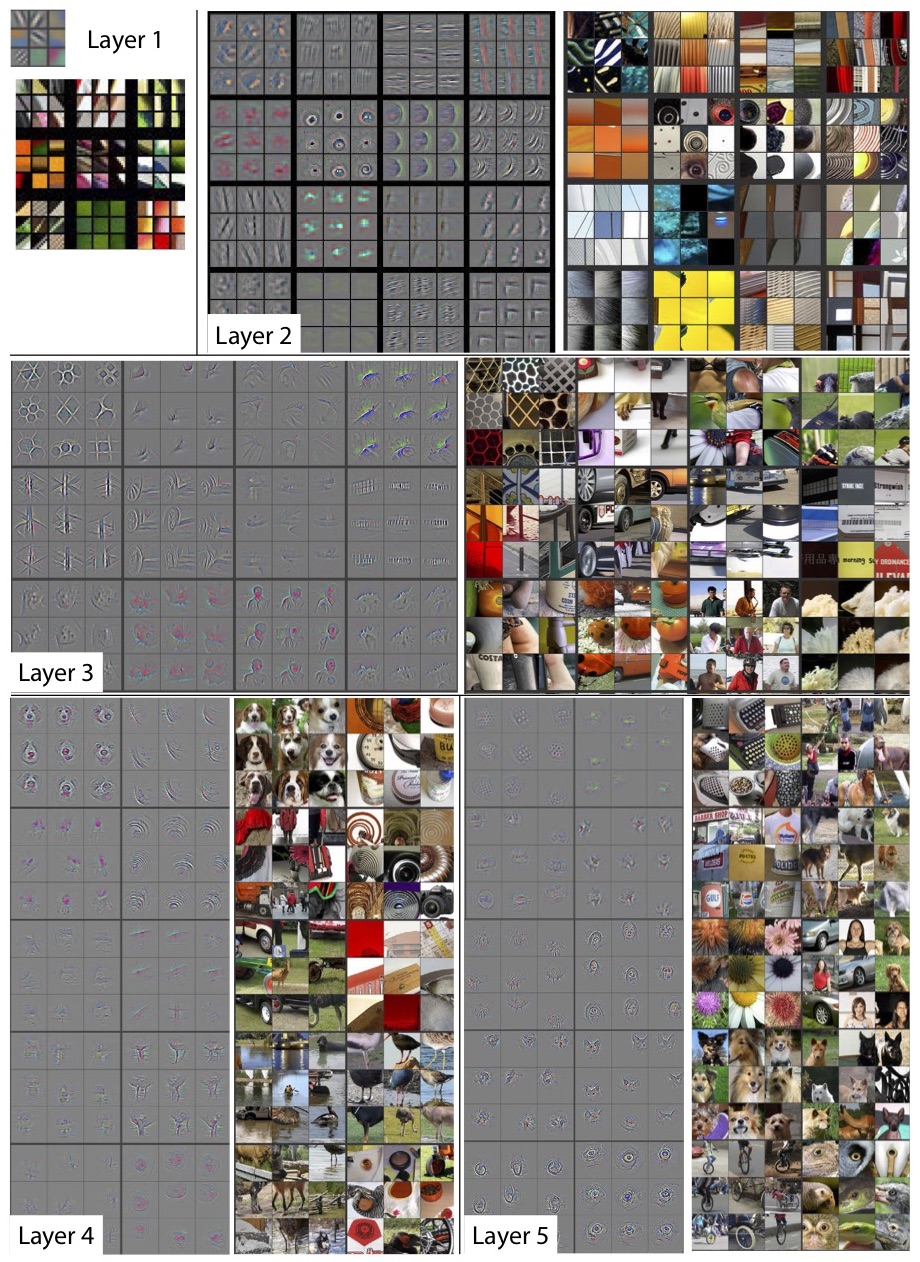


Convolutional neural networks are composed of multiple layers of artificial neurons.

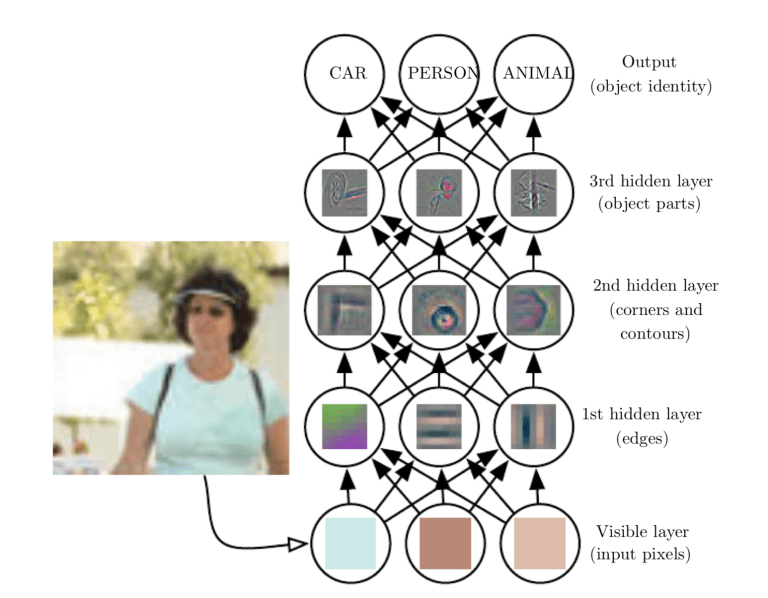
**Artificial Neurons**

Artificial neurons is a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.

The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the network it can identify even more complex features such as objects, faces, etc.



Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a “class.” For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals.



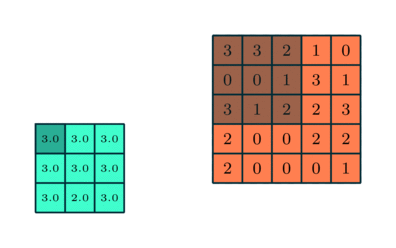
**Background of Convolutional neural networks (CNNs)**

CNN’s were first developed and used around the 1980s. The most that a Convolutional Neural Networks (CNN) could do at that time was recognize handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

In 2012 Alex Krizhevsky realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs

What Is a Pooling Layer?

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to **decrease the computational power required to process the data** by reducing the dimensions. There are two types of pooling average pooling and max pooling. I’ve only had experience with Max Pooling so far I haven’t faced any difficulties.



So what we do in Max Pooling is we find the maximum value of a pixel from a portion of the image covered by the kernel. Max Pooling also performs as a**Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

On the other hand, **Average Pooling**returns the **average of all the values**from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that **Max Pooling performs a lot better than Average Pooling**.



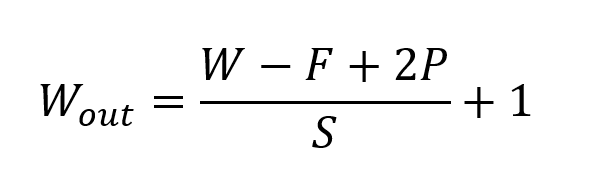
**Limitations of Convolutional neural networks (CNNs)**

Despite the power and resource complexity of CNNs, they provide in-depth results. At the root of it all, it is just recognizing patterns and details that are so minute and inconspicuous that it goes unnoticed to the human eye. But when it comes to **understanding**the contents of an image it fails.



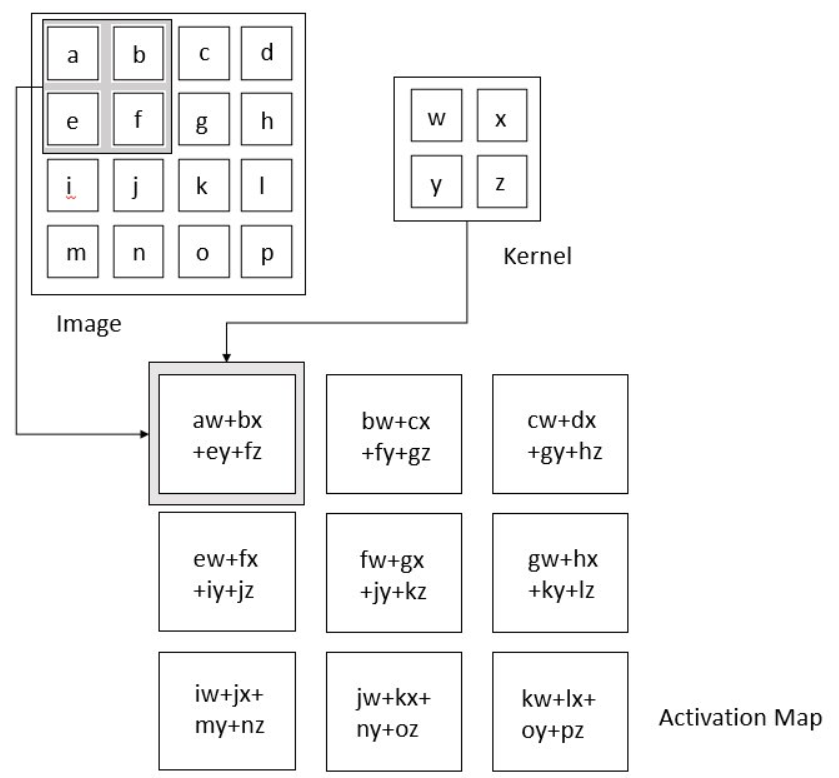
**Convolution Layer**

If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:



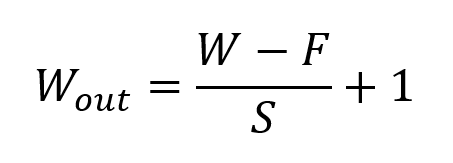
**Formula for Convolution Layer**

This will yield an output volume of size *Wout*x *Wou*t x *Dout*.



**Pooling Layer**

If we have an activation map of size *W* x *W* x *D*, a pooling kernel of spatial size *F*, and stride *S*, then the size of output volume can be determined by the following formula:



**Formula for Padding Layer**

This will yield an output volume of size *Wout*x *Wout* x *D*.

**Designing a Convolutional Neural Network**

Now that we understand the various components, we can build a convolutional neural network. We will be using Fashion-MNIST, which is a dataset of Zalando’s article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The dataset can be downloaded here.

**Our convolutional neural network has architecture as follows:**

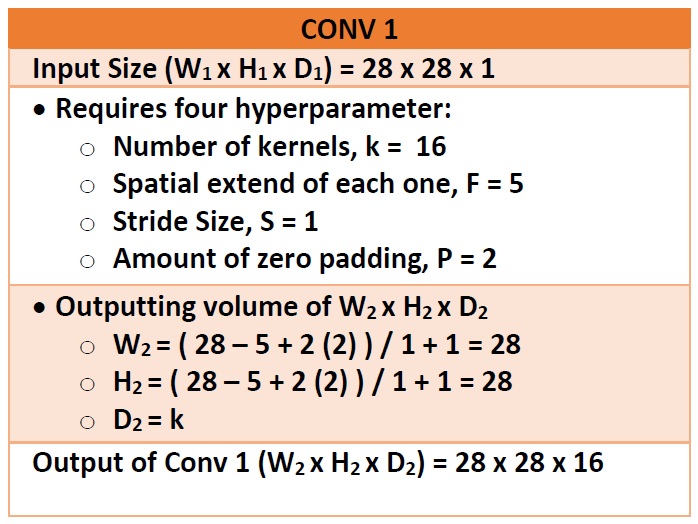
[INPUT]

→[CONV 1] → [BATCH NORM] → [ReLU] → [POOL 1]

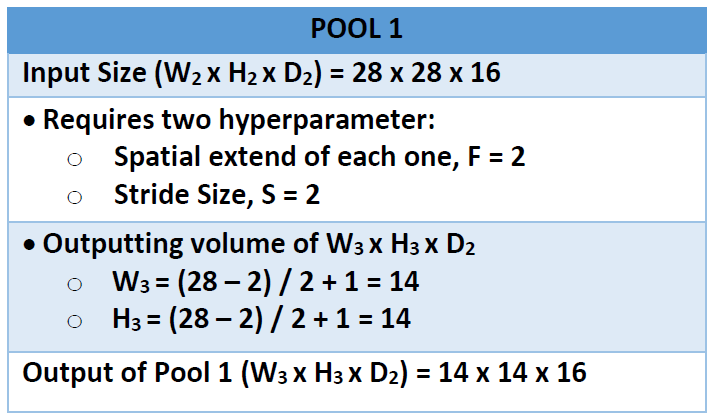
→ [CONV 2] → [BATCH NORM] → [ReLU] → [POOL 2]

→ [FC LAYER] → [RESULT]

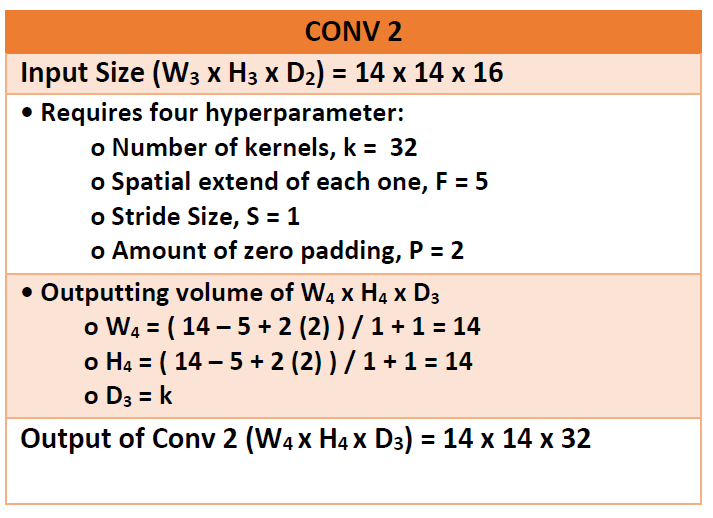
For both conv layers, we will use kernel of spatial size 5 x 5 with stride size 1 and padding of 2. For both pooling layers, we will use max pool operation with kernel size 2, stride 2, and zero padding.



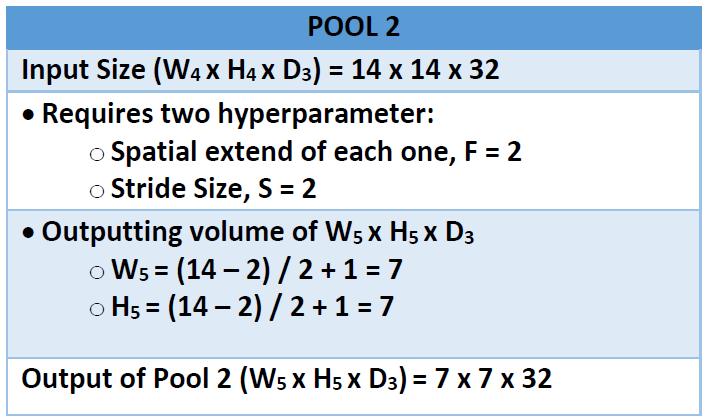
**Calculations for Conv 1 Layer (Image by Author)**



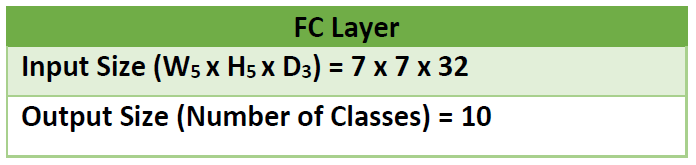
**Calculations for Pool1 Layer (Image by Author)**



**Calculations for Conv 2 Layer (Image by Author)**



**Calculations for Pool2 Layer (Image by Author)**



**Size of Fully Connected Layer (Image by Author)**