**CONVOLUTION NEURAL NETWORK**

Before moving into the depth of a convolution neural network, let us first revisit the basics of a neural network. Basics of a neural network include knowledge of its underlying architecture, i.e., the input layer, the hidden layer, and the output layer.

**Input layer**: We fed data to the input layer in the form of features. In the input layer, a number of neurons used are equal to the number of features fed as input to the system.

**Hidden Layer**: Output from the input layer goes into the hidden layer. Here we apply some activation function to get the output from the hidden layer. The number of layers and the number of neurons in each layer can be varied according to the application for which it is being designed. The greater number of layers implies a more detailed feature extraction.

**Output Layer**: Output from the hidden layer goes as input to the output layer. Here the final decision making takes place, and an object is classified under one of the labeled classes.

Processing through each layer in a forward direction, i.e., output calculation contributes towards the feed forward phase. Once the final output is obtained via the output layer, we start proceeding with error calculation part and backpropagating it to the system for error correction via weight updation.

Working of convolution neural network is highly inspired by the way a human brain perceives the world around it, learns from it and then try to classify it under some category or classes by processing any new data presented in future space. Convolution neural network is broadly known for its significant application in the field of image classification. Here in this chapter, we will try to understand the working of a convolution neural network with the help of an example designed for image classification.

Convolution Neural Network comprises of following layers:

1. A. Convolution Layer

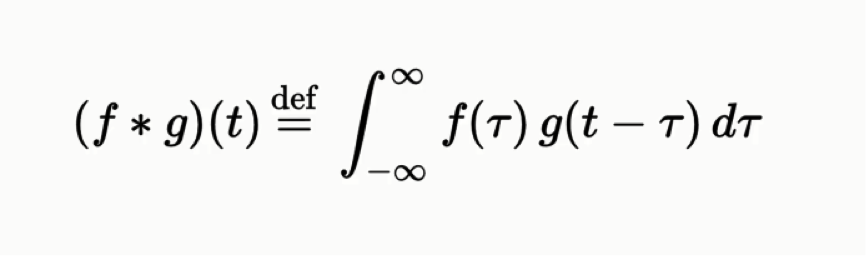
B. ReLU

1. Pooling Layer
2. Fully Connected Layer



**CONVOLUTION LAYER**

Mathematically, convolution is defined as a function derived from two given functions by integration which expresses how the shape of one is modified by the other.

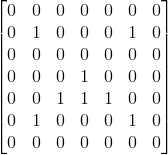


However, in deep learning convolution function is applied using two matrices, one is an input matrix, and another is a feature detector matrix. In the convolution network, the main purpose of convolution layer is to extract features from the input matrix.

**Convolution operation**

Let us consider an example of a sad face as shown below,



Matrix for an input image of sad face Feature Detector

There are three important elements of a convolution layer:

1. **Input matrix**: It contains 0’s and 1’s to describe the input image. 1’s depict the useful information, and 0's depict the blank area or area of no interest.

2.**Feature Detector**: It is also a combination of 0’s and 1’s used to represent a specific feature of interest. We can have a number of feature detectors for a single input image matrix.

3.**Feature map**: Feature map is the output matrix obtained after applying convolution operation over the input matrix and feature detector matrix. It is different from the input matrix and the feature detector in terms of the values it contains. Values stored in a feature map can be any digit, no need to be 1’s and 0’s only.

Main terminologies related to convolution layer are the following:

1. **Depth**: It gives the no of filters we use for the convolution operation.
2. **Stride**: It is defined as the number of pixels by which we move our feature detector matrix over the input matrix. Larger stride implies smaller size of feature map and vice versa.
3. **Zero-Padding**: It is defined as the process of adding extra zeros around the border of our input matrix so that we can apply the filter to border elements of our input matrix.

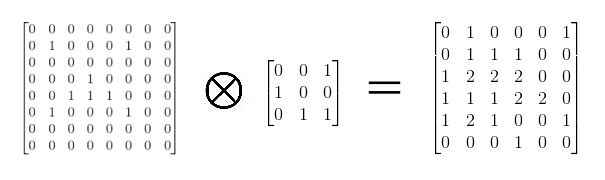
Also, zero padding results in wide convolution and not using zero padding results in narrow convolution.

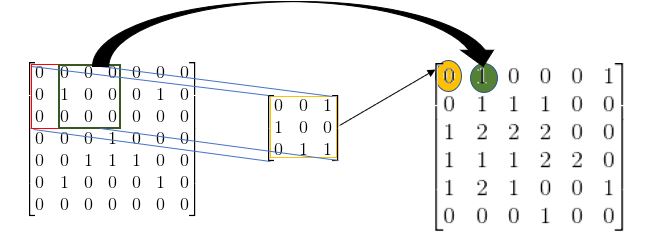
In the mentioned example, we can figure out a sad face using a combination of 1's and 0's in the input matrix.

Working of convolution operation in convolution neural network:

* we place feature detector over the input image matrix beginning from the top-left corner, and then we start counting the number of cells in which the feature detector matches the input image matrix.
* The number of matching cells is then inserted in the top-left cell of the feature map.
* We then move the feature detector one cell to the right and follow the same process. In this way, we cover the input matrix from the top-left cell to the bottom-right cell.

In the example, we assume depth to be one, stride to be one and add zero padding to make the number of rows and columns to be even for simplifying calculation part.





**Rectified Linear Unit (ReLU)**

ReLU is a non-linear operation. Purpose of using ReLU is to introduce non-linearity in our model since most of the real world data would be non-linear. Other than ReLU, list of non-linear functions includes tanh, sigmoid and many more. However, we prefer ReLU over others due to its better performance.

ReLU function is always used after convolution function to bring non-linearity in the feature map.

Mathematically, an output of ReLU operation for an input value is defined as a maximum of zero and that input value, i.e., it replaces all negative values with zero.

**OUTPUT= MAX( ZERO, INPUT )**

**POOLING LAYER**

Pooling or Spatial Pooling is also known as subsampling or downsampling as it reduces the dimensionality of each feature map. However, it retains the essential information.

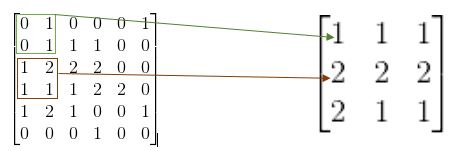
Different types of pooling operation include MAX pooling operation, AVG pooling operation, SUM pooling operation and so on.

Before applying pooling over a feature map, we need to specify window size for which we will apply pooling operation over a feature map and then shift the window without overlapping.

**MAX POOLING**

Let us first consider window size to be (2\*2). Now over an input grid of (2\*2) apply max operation to get the output of this particular input grid considered. Repeat this operation over the entire feature map such as it covers the feature map from the top left corner to bottom right corner. MAX operation is simply the largest element of all from the considered input window.

**OUTPUT = MAX(input window)**

∑

**SUM POOLING**

Working of sum poling is exactly similar to the MAX pooling except for the main operation. Here, the main operation we use is the sum of all the elements in the input window considered.

**OUTPUT = SUM(input window)**



Similarly, other operations are applied.

**Benefits of POOLING:**

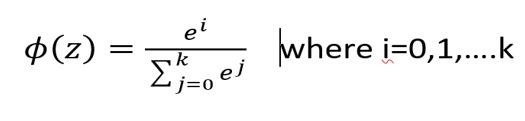
1. Makes the input representation smaller and more manageable.
2. Control Overfitting, i.e. reduces intaking of extra features which are not required.
3. Makes the network invariant to small transformations as we are applying operations such as AVG, SUM, MAX in pooling.
4. Almost scale invariant representation.

**FULLY CONNECTED LAYER**

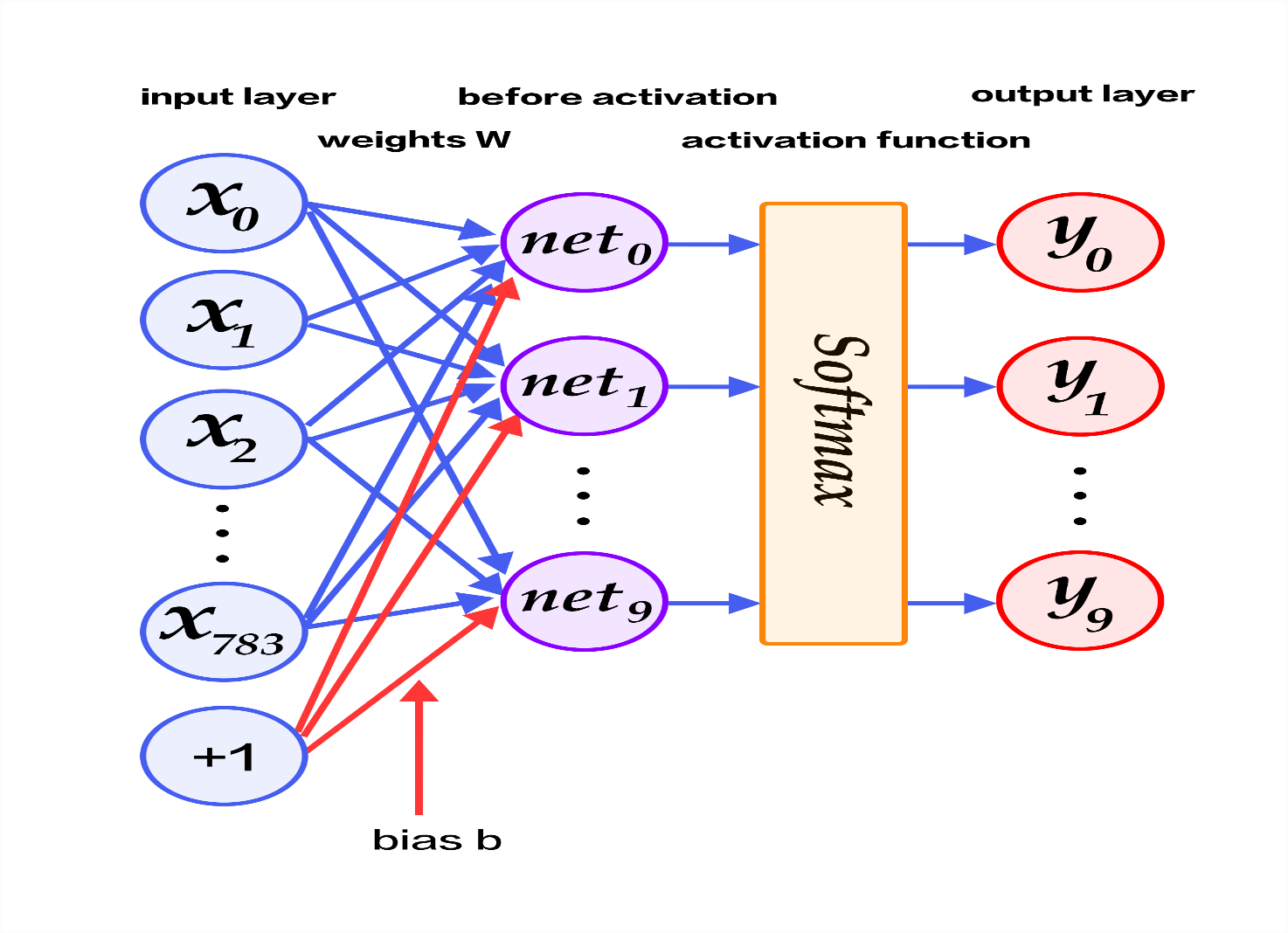
Fully Connected layer acts as a classifier while layers discussed till now, i.e., Convolution Layer and Pooling layer are feature extractors.

The fully connected layer consists of multilayer perceptron and fully connected layers with softmax activation function. Fully Connected means every neuron in the previous layer will be connected with every neuron in the next layer. The fully connected layer is also a cheap way of learning non-linear combinations of the feature.

Softmax activation function helps in ensuring the sum of output probabilities from a fully connected layer to be one.



**SOFTMAX ACTIVATION FUNCTION**



**STEPS FOR CONVOLUTION NEURAL NETWORK**

1. Initialize all the filters and parameters or weights with random values.
2. Perform the following operations to find out output probabilities for each class:
3. Convolution
4. ReLU
5. Pooling
6. Forward Propagation
7. Since, weights considered are random therefor, probabilities calculated for first training are also random.
8. Calculate total error at the output layer,

**E= ∑ ( 0.5 \* (target probability - output probability)^2 )**

1. Use Backpropagation to calculate the gradients of the error for all weights and perform weight updation.

Note that the number of filters, filter size, architecture of the network, etc. remains fixed. Only the values of the filter matrix and connection weights get updated.

**CONVOLUTION NEURAL NETWORK APPLIED TO AUTONOMOUS VEHICLES**

Autonomous vehicle is a big innovation towards saving lives due to road accidents. Since, self driving cars works on machinery therefor it never gets tired and results in most accurate and efficient driving.

Important components of Self Driving car includes scanning camera, sensor, radar laser to perceive the world around and to create its digital map.

One of the important functioning of autonomous vehicle includes object detection. Two major steps involved in object detection are object classification and object localization i.e., detecting the position of the object in the surrounding environment. The algorithm which serves as the basis for object classification is Convolution neural network. To detect the different objects present in the captured image we use different window size and slide over the captured image. Since, sliding the windows of different size increases the computational time therefor, to overcome this problem we use the concept of YOLO i.e., You Only Look Once. In YOLO we split our image into grids and run it through CNN.