What is Convolutional Neural Network

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

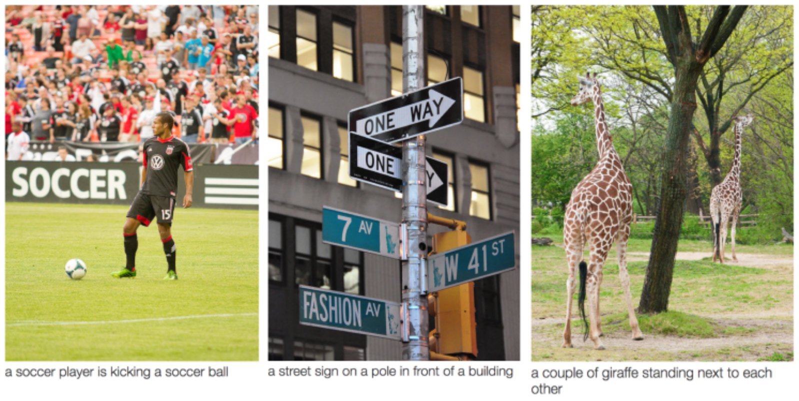

a soccer player is kicking a soccer ball
a street sign on a pole in front of a building
a couple of giraffe standing next to each other

Figure 1

In above figure, a ConvNet is able to recognize scenes and the system is able to suggest relevant captions ("a soccer player is kicking a soccer ball") while below figure shows an example of ConvNets being used for recognizing everyday objects, humans and animals. Lately, ConvNets have been effective in several Natural Language Processing tasks (such as sentence classification) as well.

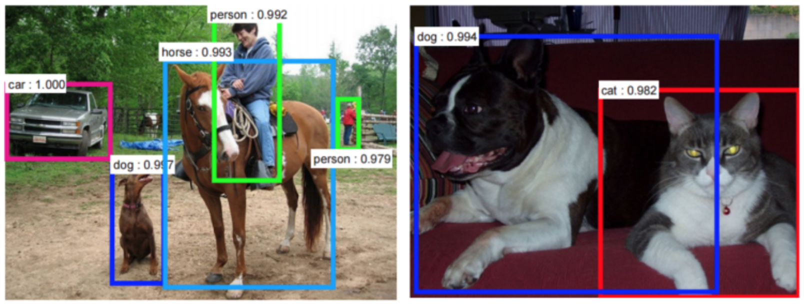


Figure 2

How does a CNN Work?

A convolutional neural network can have tens or hundreds of layers that each learn to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very

simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.
CNNs perform feature identification and classification of images, text, sound, and video.

Feature Learning, Layers, and Classification

Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.

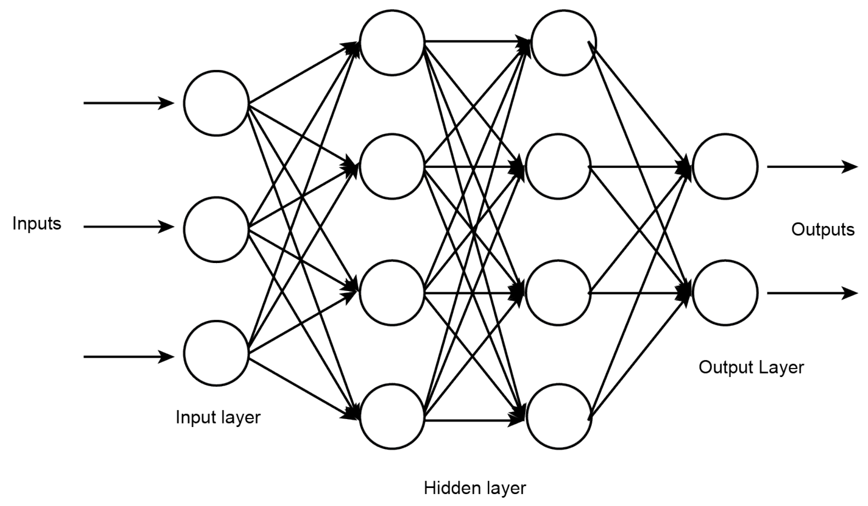


Figure 3

These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are: convolution, activation or ReLU, and pooling.

Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images.

Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as activation, because only the activated features are carried forward into the next layer.

Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

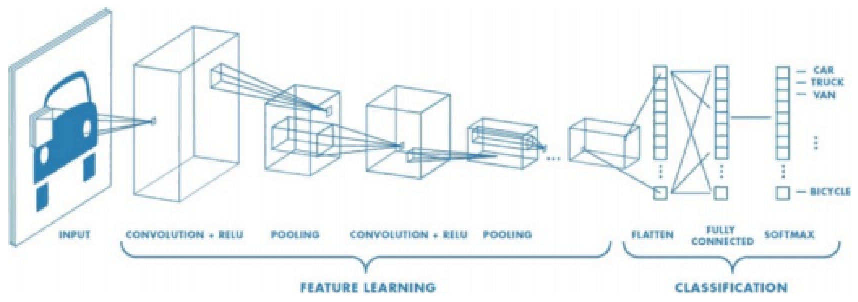


Figure 4

Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer.

Convolutional Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputssuch as image matrix and a filter or kernel.

- An image matrix(volume) of dimension (h * w * d)
- A filter($f_h * f_w * d$)
- outputs a volume dimension $(h - f_h + 1) * (w - f_w + 1) * 1$

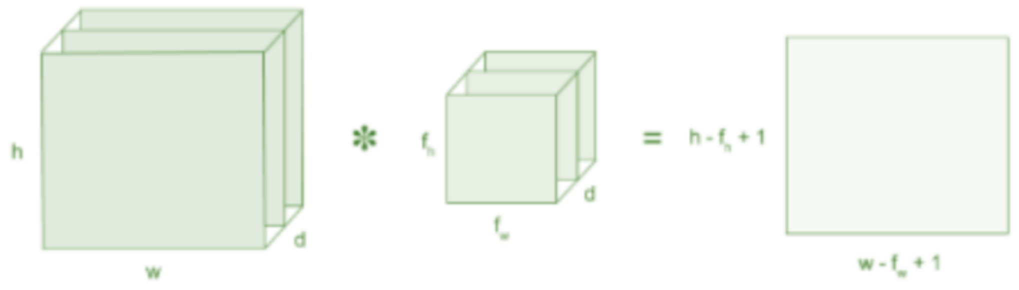


Figure 5

Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

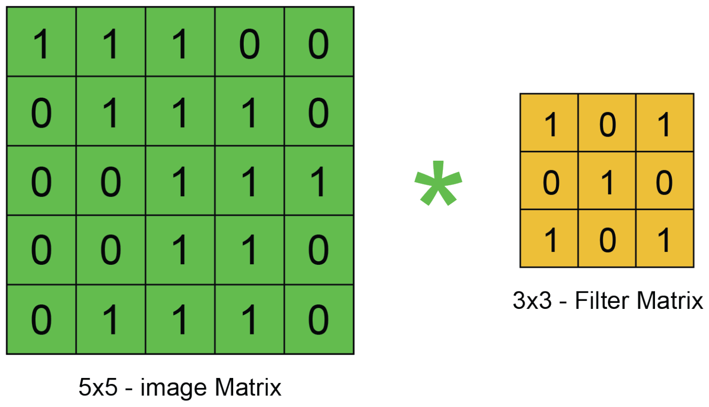


Figure 6

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called "Feature Map" as output shown in below

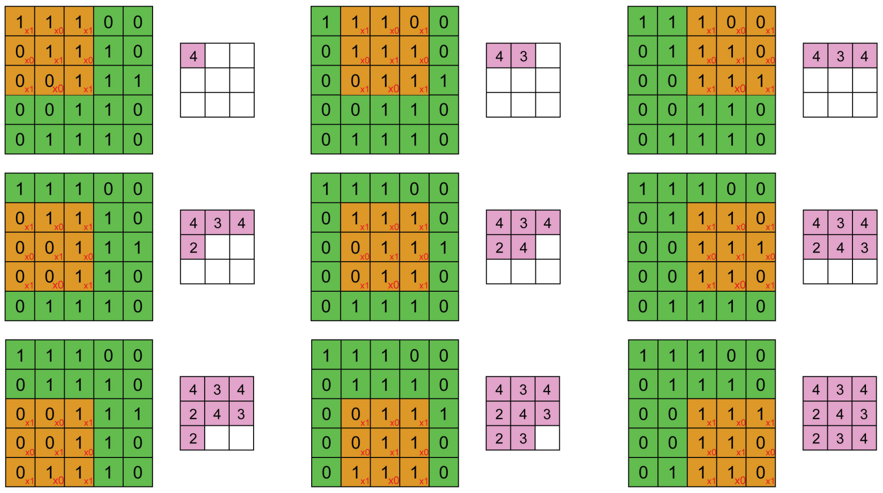


Figure 7

Take a moment to understand how the computation above is being done. We slide the orange matrix over our original image (green) by 1 pixel (also called 'stride') and for every position, we compute element wise multiplication (between the two matrices) and add the multiplication outputs to get the final integer which forms a single element of the output matrix (pink). Note that the 3∗3

matrix “sees” only a part of the input image in each stride.
In CNN terminology, the 3∗3 matrix is called a ‘filter‘ or ‘kernel’ or ‘feature detector’ and the matrix formed by sliding the filter over the image and computing the dot product is called the ‘Convolved Feature’ or ‘Activation Map’ or the ‘Feature Map‘. It is important to note that filters acts as feature detectors from the original input image.

Feature Map

The feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. You can see that if the receptive field is moved one pixel from activation to activation, then the field will overlap with the previous activation by (field width − 1) input values.
The filter matrix will produce different Feature Maps for the same input image. As an example, consider the following input image:

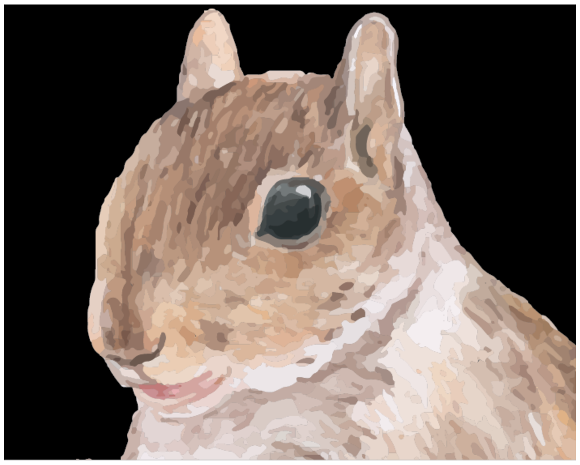
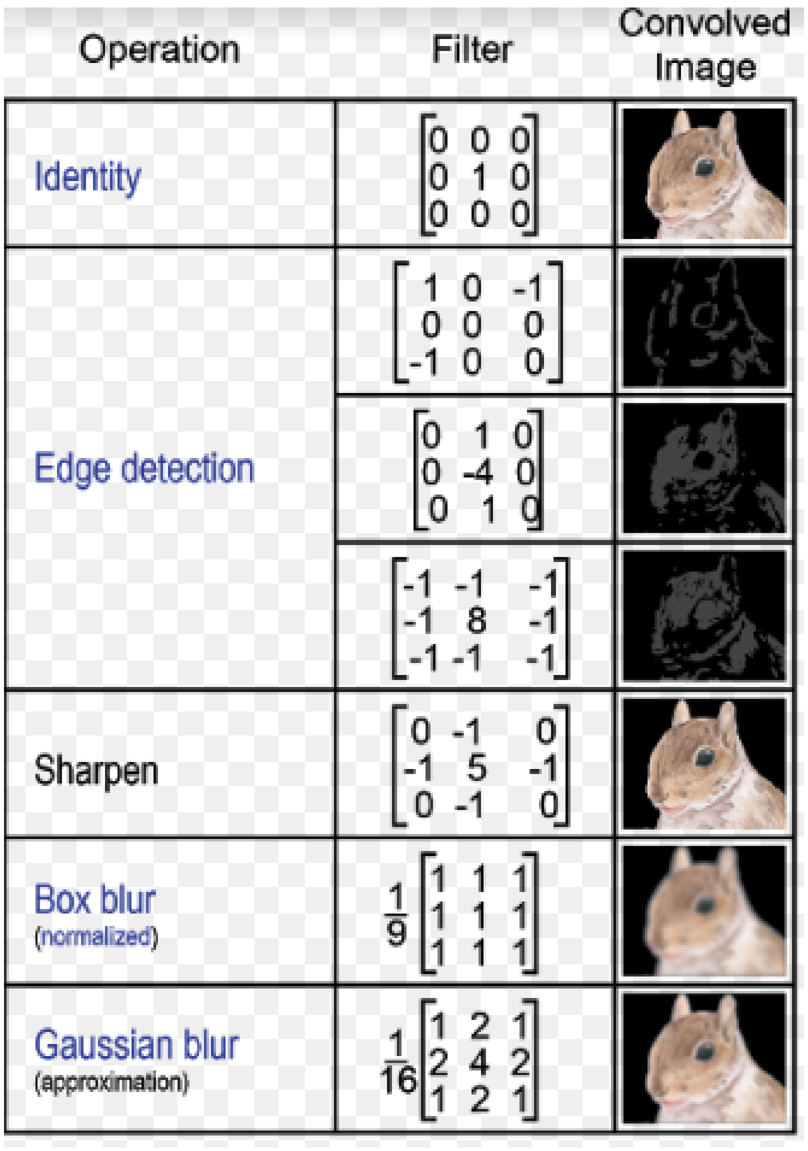


Figure 8

In the table of next page, we can see the effects of convolution of the above image with different filters.



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & -4 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 9

As shown, we can perform operations such as edge detection, blur and sharpen by applying filters. The example shows various convolution image after applying different types of filters (Kernels).

Strides

Strides is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

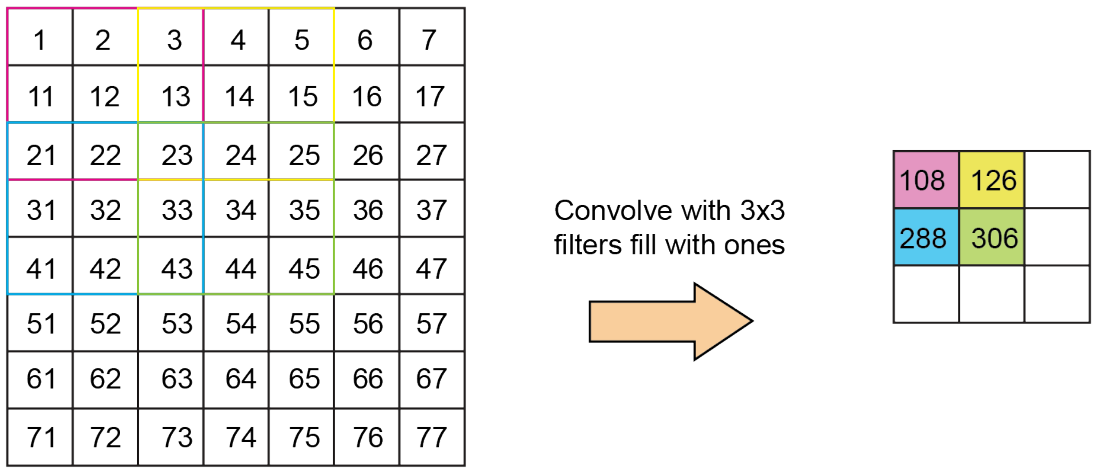


Figure 10

Padding

Padding is to add extra pixels outside the image and zero padding means every pixel value that you add is zero.

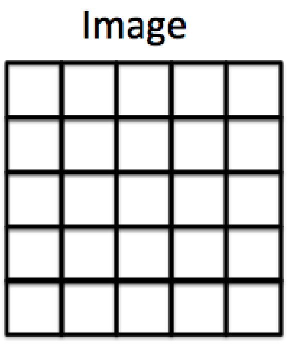


Figure 11

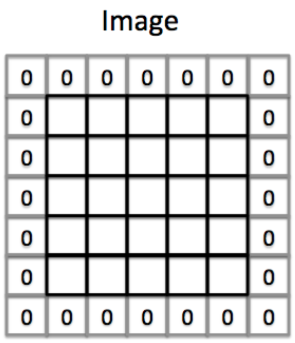


Figure 12

Pooling Layer

Pooling layers section would reduce the number of parameters when the images aretoo large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information.
Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

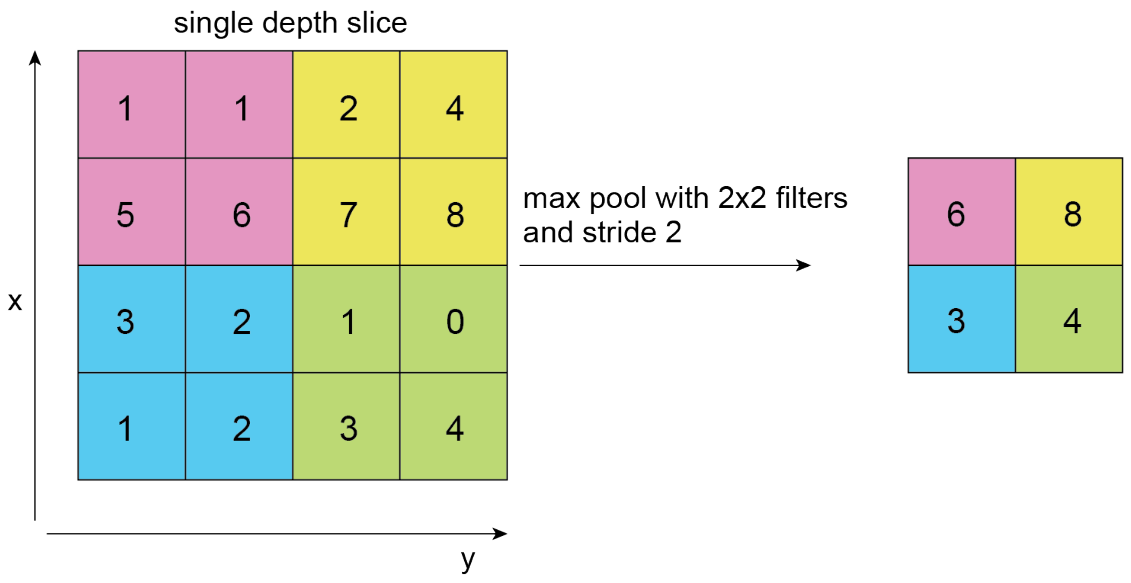


Figure 13

Fully Connected Layer

The layer we call as Fully Connected Layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

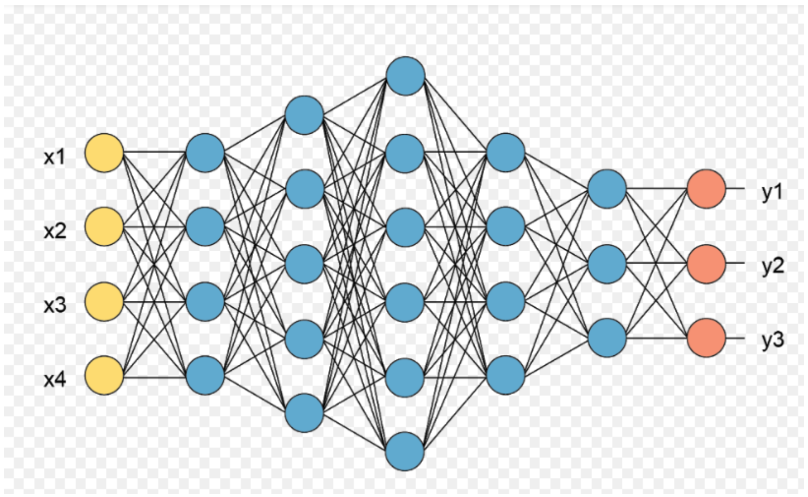


Figure 14

In the above diagram, feature map matrix will be converted as vector $(x_1, x_2, x_3, ...)$. With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as cat, dog, car, truck etc.

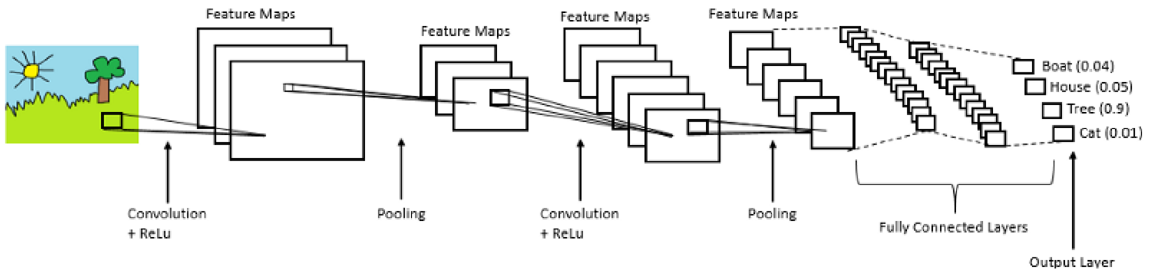


Figure 15

References:

For more details can refer below:
`https://en.wikipedia.org/wiki/Convolutional$_$neural$_$network`
`http://cs231n.github.io/convolutional-networks/`