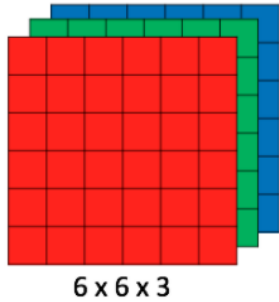


In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

CNN image classifications takes an input image, process it and classify it under certain categories (Eg., Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see  $h \times w \times d$  (  $h$  = Height,  $w$  = Width,  $d$  = Dimension ). Eg., An image of  $6 \times 6 \times 3$  array of matrix of RGB (3 refers to RGB values) and an image of  $4 \times 4 \times 1$  array of matrix of grayscale image.



Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

### Convolution 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 inputs such as image matrix and a filter or kernel

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



Figure 3: Image matrix multiplies kernel or filter matrix

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

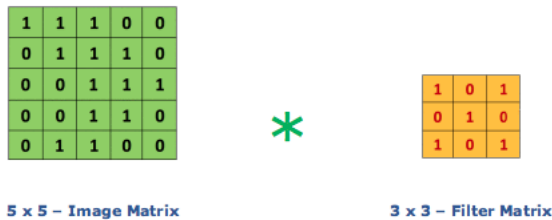


Figure 4: Image matrix multiplies kernel or filter matrix

Then the convolution of  $5 \times 5$  image matrix multiplies with  $3 \times 3$  filter matrix which is called “Feature Map” as output shown in below

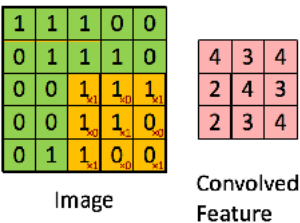


Figure 5:  $3 \times 3$  Output matrix

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

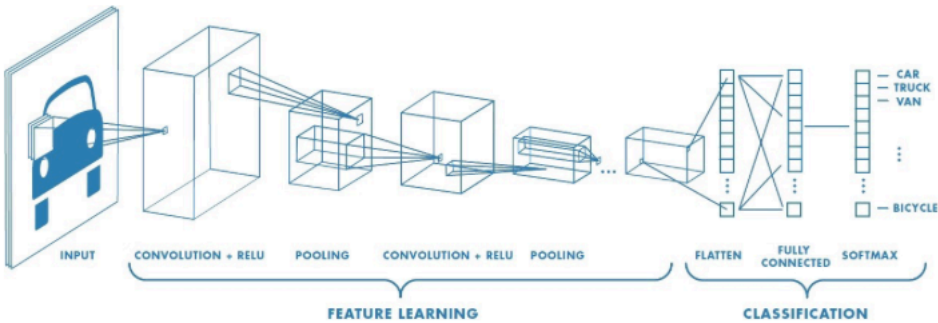


Figure 2 : Neural network with many convolutional layers

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
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 7 : Some common filters

## Strides

Stride 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.

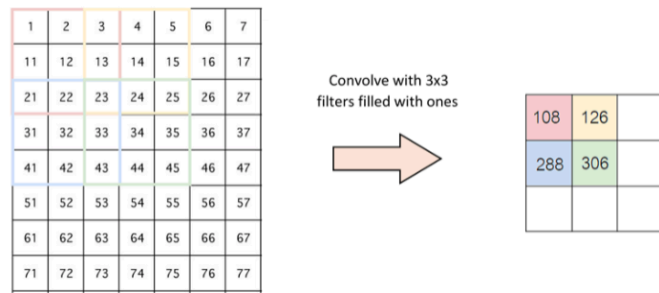


Figure 6 : Stride of 2 pixels

## Padding

Sometimes filter does not fit perfectly fit the input image. We have two options:

- Pad the picture with zeros (zero-padding) so that it fits
- Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

## Non Linearity (ReLU)

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is  $f(x) = \max(0, x)$ .

Why ReLU is important : ReLU's purpose is to introduce non-linearity in our ConvNet. Since, the real world data scientist uses ReLU since performance wise ReLU is better than other two.

There are other non linear functions such as tanh or sigmoid can also be used instead of ReLU. Most of the data scientists uses ReLU since performance wise ReLU is better than other two.

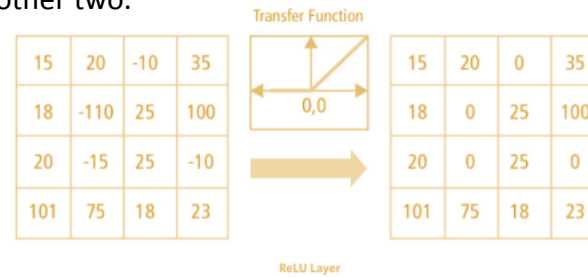


Figure 7 : ReLU operation

## Pooling Layer

Pooling layers section would reduce the number of parameters when the images are too 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.

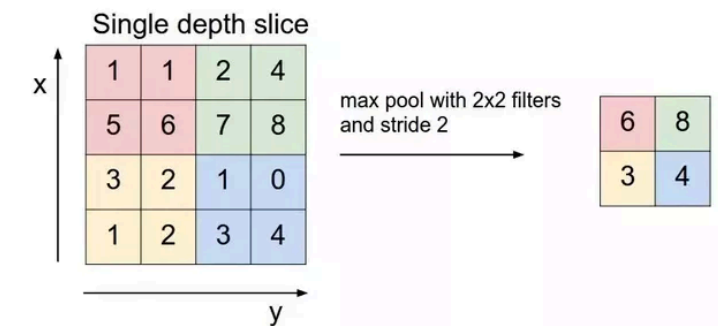


Figure 8 : Max Pooling

## Fully Connected Layer

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

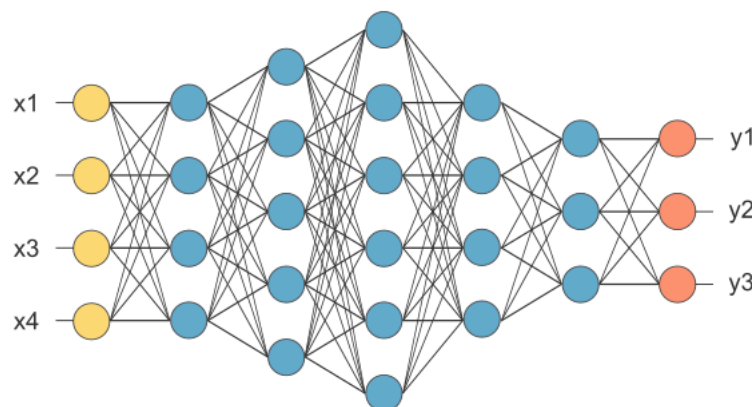


Figure 9 : After pooling layer, flattened as FC layer

In the above diagram, feature map matrix will be converted as vector (x1, x2, x3, ...). 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.,

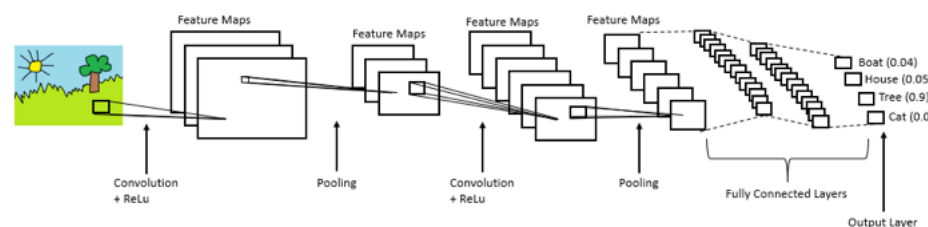


Figure 10 : Complete CNN architecture

## Summary

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer)
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.

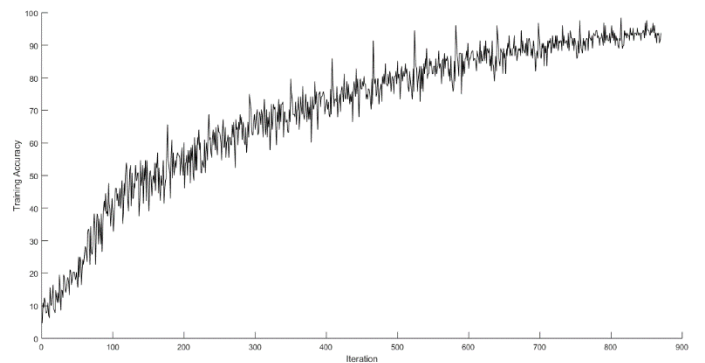
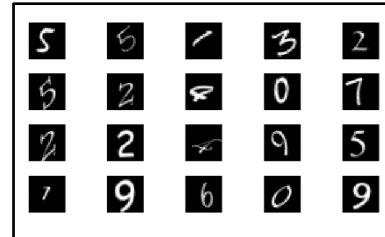
## Convolutional Neural Networks

We use the script CNNDigits.m of the handwritten digits dataset to investigate some CNN architectures with different number of layers, combinations of different kinds of layers, dimensions of the weights, etc.

### CNN Architecture:

```
InputLayer([28 28 1])  
convolution2dLayer(5,20)  
reluLayer  
maxPooling2dLayer(2,'Stride',2)  
fullyConnectedLayer(10)  
softmaxLayer  
classificationLayer();
```

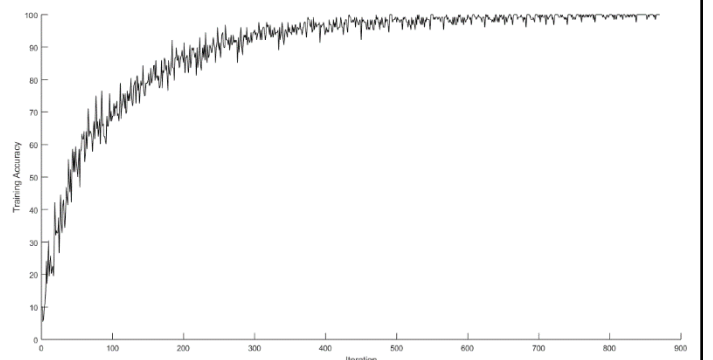
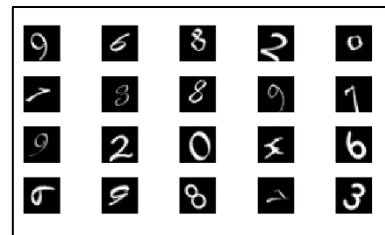
94%  
Accuracy



### CNN Architecture:

```
InputLayer([28 28 1])  
convolution2dLayer(5,12)  
reluLayer  
maxPooling2dLayer(2,'Stride',2)  
fullyConnectedLayer(10)  
softmaxLayer  
classificationLayer();
```

98%  
Accuracy

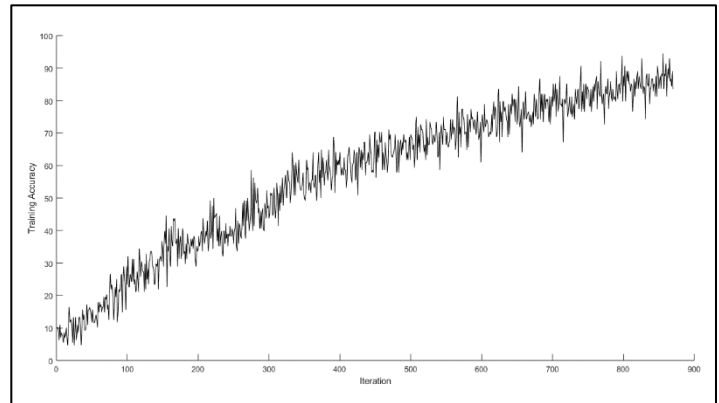
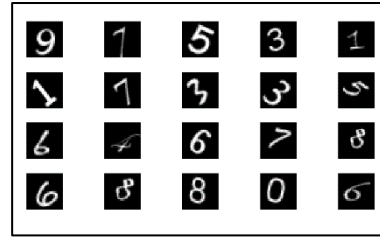


### CNN Architecture:

```
InputLayer([28 28 1])
convolution2dLayer(5,20)
reluLayer
maxPooling2dLayer(2,'Stride',2)
fullyConnectedLayer(10)
fullyConnectedLayer(10)
softmaxLayer
classificationLayer();
```



85%  
Accuracy



### CNN Architecture

```
InputLayer([28 28 1])
convolution2dLayer(5,12)
reluLayer
maxPooling2dLayer(2,'Stride',2)
convolution2dLayer(5,24)
reluLayer
fullyConnectedLayer(10)
softmaxLayer
classificationLayer();
```



85%  
Accuracy

