

: Convolution Neural Networks

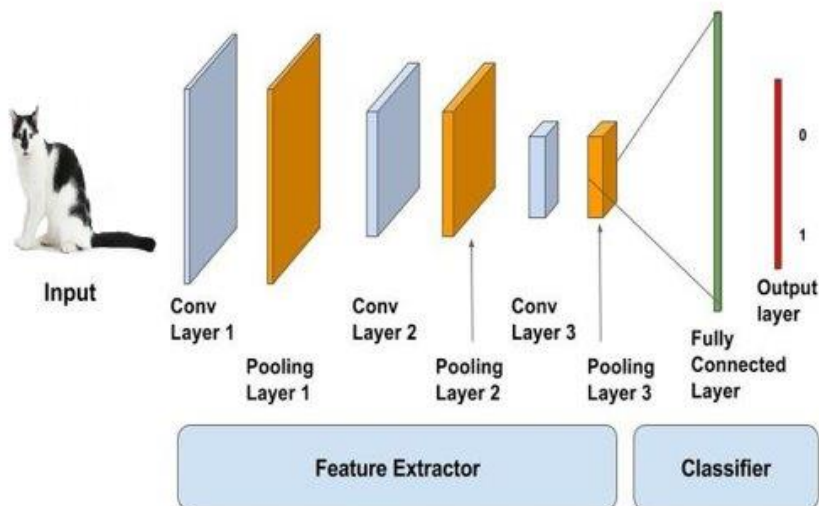
Topics Covered

1. Convolution Neural Networks
2. The Neuroscientific Basis for Convolutional Networks
3. Convolution Process
4. Maxpooling
5. Dropout
6. Maths Behind CNNs
7. Feature Extraction
8. Variants Of The Basic Convolution Function
9. Summary

Convolution Neural Networks

Convolution networks are the types of NN which are useful for data like images, videos, or text. This type of data is highly complex and non-linear. Also, the features within the images such as edges, curves, corners, color contrast, etc., can't be extracted manually. Hence CNN completely takes care of such data.

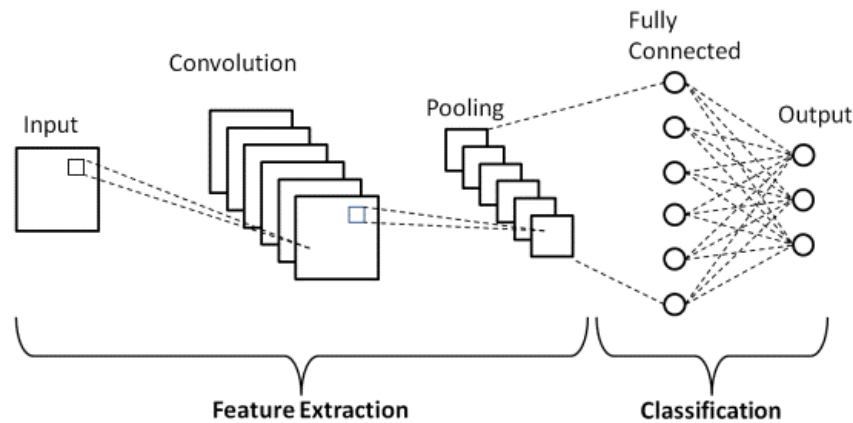
One of the major advantages of CNN is that it performs dimensionality reduction on the data, and it also automatically extracts the unique features.



Basic Architecture

There are two main parts to a CNN architecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.



Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

1. Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size $M \times M$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ($M \times M$).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In **Max Pooling**, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer

3. Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

4. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

5. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

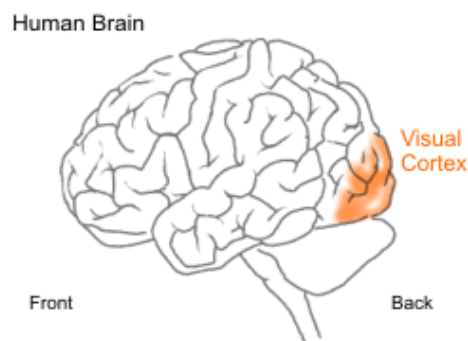
It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for a multi-class classification, generally softmax is used.

The Neuroscientific Basis for Convolutional Networks

The history of convolutional networks begins with neuroscientific experiments long before the relevant computational models were developed.

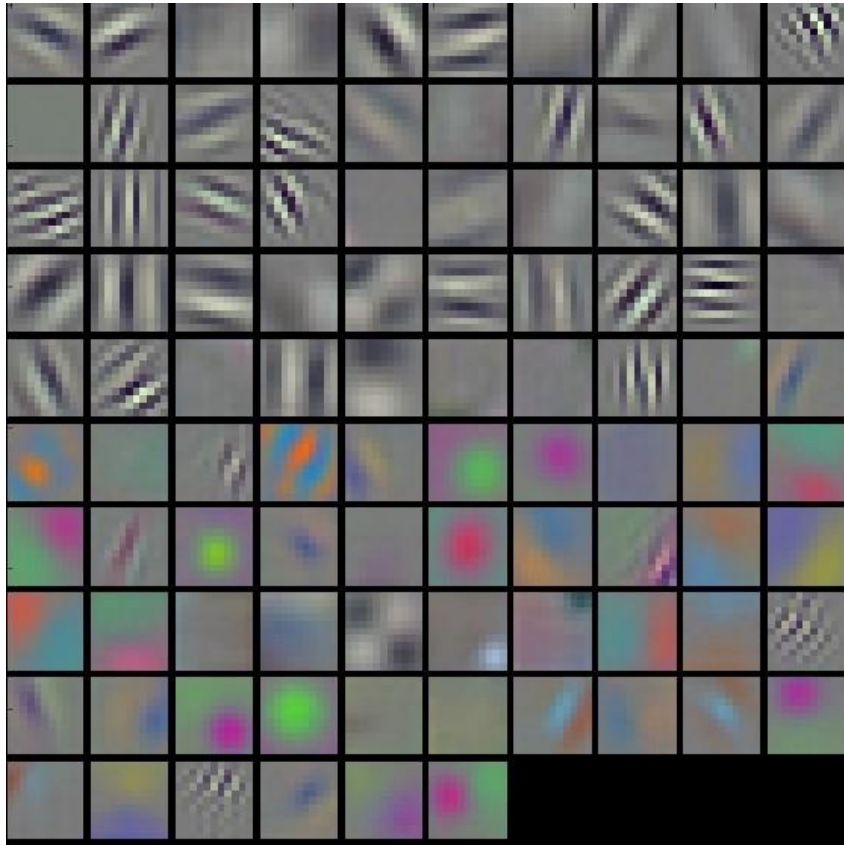
Neurophysiologists David Hubel and Torsten Wiesel observed how neurons in the cat's brain responded to images projected in precise locations on a screen in front of the cat.

“Their great discovery was that neurons in the early visual system responded most strongly to very specific patterns of light, such as precisely oriented bars, but responded hardly at all to other patterns”



The Neurons in the early visual cortex are organized in a hierarchical fashion, where the first cells connected to the cat's retinas are responsible for detecting simple patterns like edges and bars, followed by later layers responding to more complex patterns by combining the earlier neuronal activities.

Convolutional Neural Network may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to detect higher-level features, such as facial shapes in higher layers



Filters in a Convolutional Neural network

The **Visual Cortex** of the brain is a part of the cerebral cortex that processes visual information. **V1** is the first area of the brain that begins to perform significantly advanced processing of visual input.

A convolutional network layer is designed to capture three properties of V1:

1. V1 is arranged in a spatial map. It actually has a two-dimensional structure mirroring the structure of the image in the retina. Convolutional networks capture this property by having their features defined in terms of two dimensional maps.
2. V1 contains many simple cells. A simple cell's activity can be characterized by a linear function of the image in a small, spatially localized receptive field. The detector units of a convolutional network are designed to emulate these properties of simple cells.

3. V1 also contains many complex cells. These cells respond to features that are similar to those detected by simple cells, but complex cells are invariant to small shifts in the position of the feature. This inspires the pooling units of convolutional networks.

There are many differences between convolutional networks and the mammalian vision system. Some of these differences are -

1. The human eye is mostly very low resolution, except for a tiny patch called the fovea. Most convolutional networks receive large full resolution photographs as input.
2. The human visual system is integrated with many other senses, such as hearing, and factors like our moods and thoughts. Convolutional networks so far are purely visual.
3. Even simple brain areas like V1 are heavily impacted by feedback from higher levels. Feedback has been explored extensively in neural network models but has not yet been shown to offer a compelling improvement.





Convolution Process

The name “Convolutional neural network” indicates that the network employs a mathematical operation called Convolution. Convolution is a specialized kind of linear operation. Convnets are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

Convolution between two functions in mathematics produces a third function expressing how the shape of one function is modified by other

Convolution Kernels

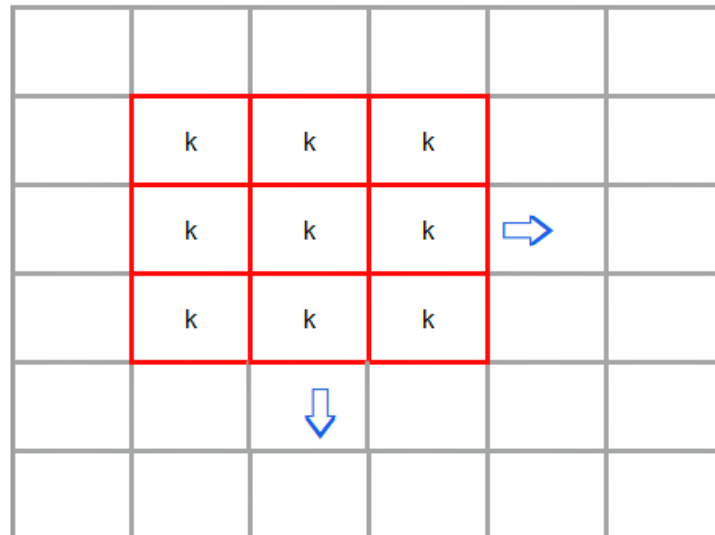
A kernel is a small 2D matrix whose contents are based upon the operations to be performed. A kernel maps on the input image by simple matrix multiplication and addition, the output obtained is of lower dimensions and therefore easier to work with.

<i>Original</i>	<i>Gaussian Blur</i>	<i>Sharpen</i>	<i>Edge Detection</i>
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

Kernel types

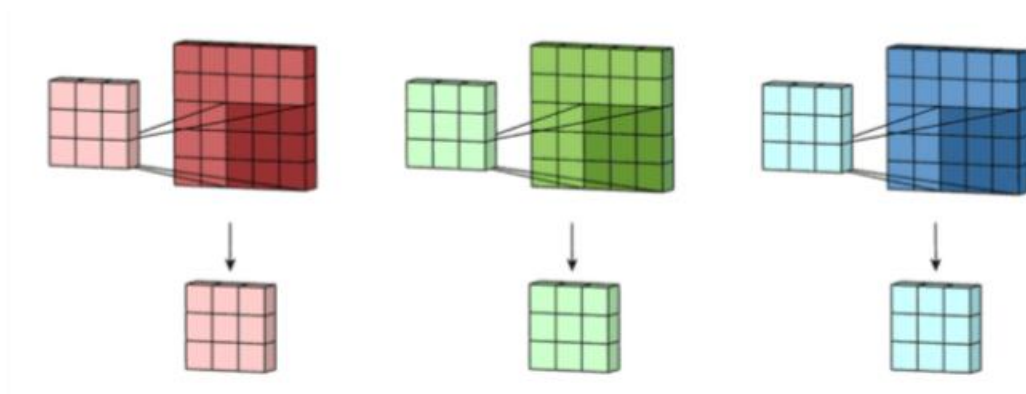
Above is an example of a kernel for applying Gaussian blur(to smoothen the image before processing), Sharpen image(enhance the depth of edges) and edge detection.

The shape of a kernel is heavily dependent on the input shape of the image and architecture of the entire network, mostly the size of kernels is **(MxM)** i.e a square matrix. The movement of a kernel is always from left to right and top to bottom.



Kernel movement

Stride defines by what step does to kernel move, for example stride of 1 makes kernel slide by one row/column at a time and stride of 2 moves kernel by 2 rows/columns.



Multiple kernels aka filters with stride=1

For input images with 3 or more channels such as RGB a **filter** is applied
Filters are one dimension higher than kernels and can be seen as multiple kernels stacked on each other where every kernel is for a particular channel.
Therefore for an RGB image of (32x32) we have a filter of the shape say (5x5x3)
Now let's see how a kernel operates on sample matrix.

Here the input matrix has shape 4x4x1 and the kernel is of size 3x3 since the shape of input is larger than the kernel, we are able to implement a sliding window protocol and apply the kernel over entire input. First entry in the convoluted result is calculated as:

$$45*0 + 12*(-1) + 5*0 + 22*(-1) + 10*5 + 35*(-1) + 88*0 + 26*(-1) + 51*0 = -45$$

Input Matrix

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

Kernel

0	-1	0
-1	5	-1
0	-1	0

⇒

-45	12
22	10

Input Matrix

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

Kernel

0	-1	0
-1	5	-1
0	-1	0

⇒

-45	103
22	10

Input Matrix

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

Kernel

0	-1	0
-1	5	-1
0	-1	0

⇒

-45	103
-96	10

Input Matrix

45	12	5	17
22	10	35	6
88	26	51	19
9	77	42	3

Kernel

0	-1	0
-1	5	-1
0	-1	0

⇒

-45	103
-176	133

Convolution in action

Sliding window protocol:

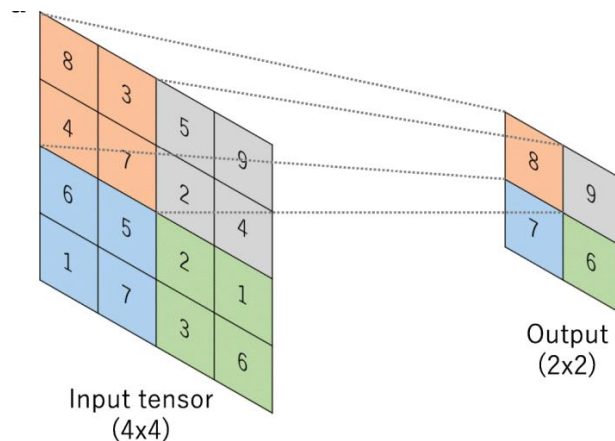
1. The kernel gets into position at the top-left corner of the input matrix.
2. Then it starts moving left to right, calculating the dot product and saving it to a new matrix until it has reached the last column.
3. Next, kernel resets its position at first column but now it slides one row to the bottom. Thus following the fashion left-right and top-bottom.
4. Steps 2 and 3 are repeated till the entire input has been processed.

For a 3D input matrix the movement of the kernel will be from front to back, left to right and top to bottom.

MaxPooling

Max Pooling is a convolution process where the Kernel extracts the maximum value of the area it convolves. Max Pooling simply says to the Convolutional Neural Network that we will carry forward only that information, if that is the largest information available amplitude wise.

Max-pooling on a 4×4 channel using 2×2 kernel and a stride of 2: As we are convolving with a 2×2 Kernel. If we observe the first 2×2 set on which the kernel is focusing the channel have four values 8,3,4,7. Max-Pooling picks the maximum value from that set which is "8".

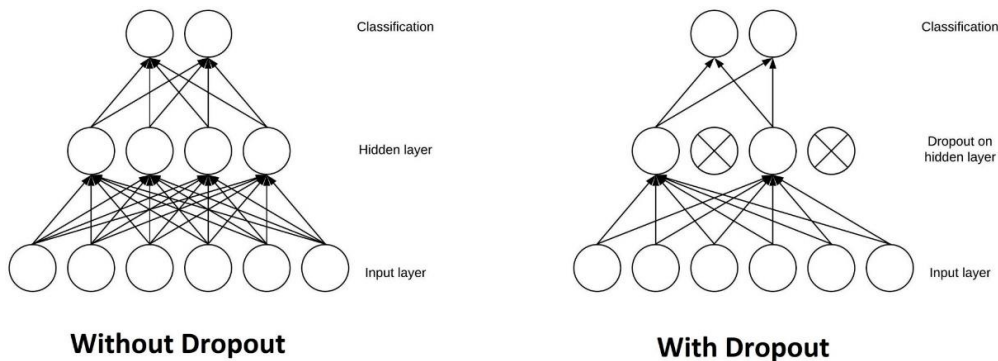


It's not advised to do Max pooling in the initial stages of the Convolutional Neural Network as the Kernels would be at the stage of extracting edges and gradients.

Dropout

Another typical characteristic of CNNs is a Dropout layer. The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. We can apply a Dropout layer to the input vector, in which case it nullifies some of its features; but we can also apply it to a hidden layer, in which case it nullifies some hidden neurons.

Dropout layers are important in training CNNs because they prevent overfitting on the training data. If they aren't present, the first batch of training samples influences the learning in a disproportionately high manner. This, in turn, would prevent the learning of features that appear only in later samples or batches:

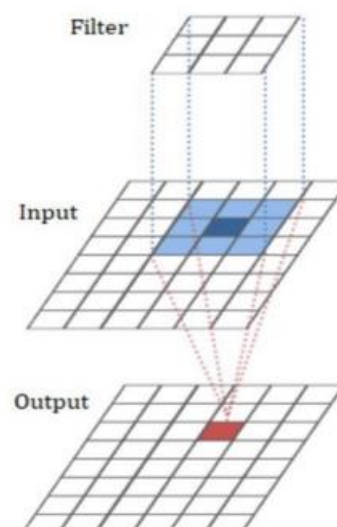
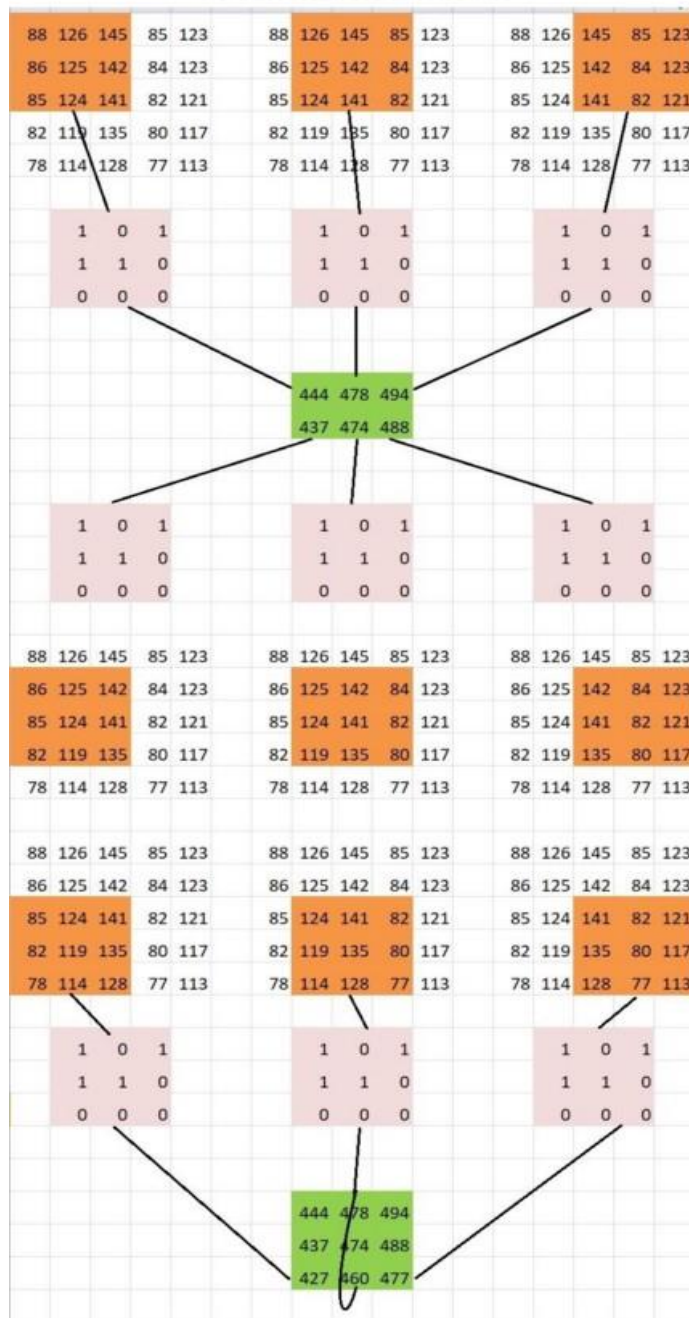


Say we show ten pictures of a circle, in succession, to a CNN during training. The CNN won't learn that straight lines exist; as a consequence, it'll be pretty confused if we later show it a picture of a square. We can prevent these cases by adding Dropout layers to the network's architecture, in order to prevent overfitting.

For simplicity, we took 0 and 1 for filters, usually, these are continuous values.

The filter convolutes with the image to detect patterns and features.

Convolution Process



Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called “**Feature Map**”. We apply the dot product to the scalar value and then move the filter by the stride over the entire image.

Sometimes filter does not fit perfectly fit the input image. Then there is a need to pad the image with zeros as shown below. This is called **padding**

Next, we need to reduce the size of images, if they are too large. **Pooling layers** section would reduce the number of parameters when the images are too large.



As shown in the above image, the padding is applied so that the filter perfectly fits the given image. Adding pooling layer then decrease the size of the image and hence decrease the complexity and computations.

Next Step, is **Normalization**. Usually, an activation function ReLU is used. ReLU stands for Rectified Linear Unit for a non-linear operation.

The output is $f(x) = \max(0, x)$.

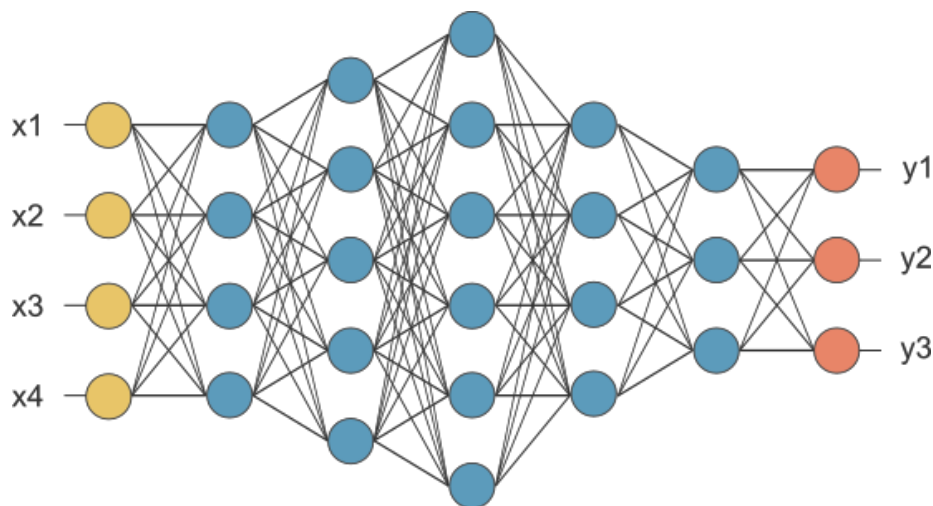
The purpose of ReLU is to add non-linearity to the convolutional network. In usual cases, the real-world data want our network to learn non-linear values.

A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. Here we are assuming that we have negative values since dealing with the real-world data. In case, if there is no negative value, you can skip this part.

$$RELU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

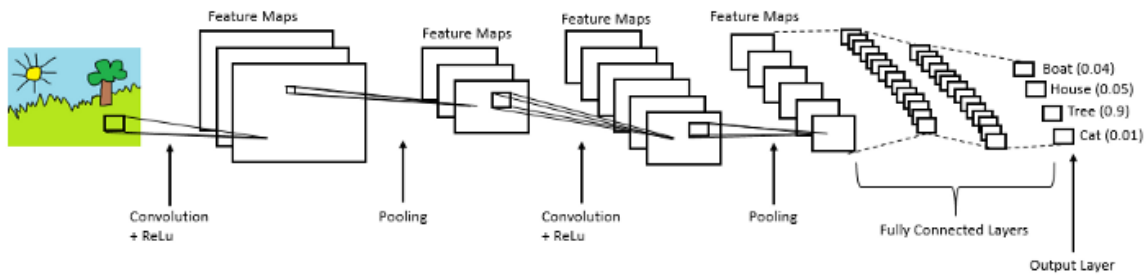
-478	494	0	494
460	-477	460	0

The final step is to flatten our matrix and feed the values to fully connected layer.



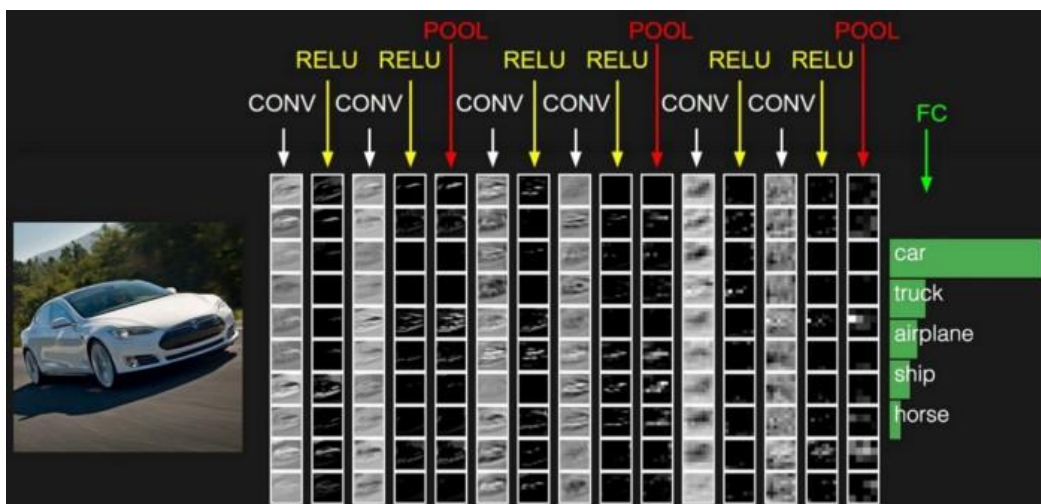
Next, we need to train the model in the same way, we train other neural networks. Using the certain number of epochs and then backpropagate to update weights and calculate the loss.

Overall Structure of CNN



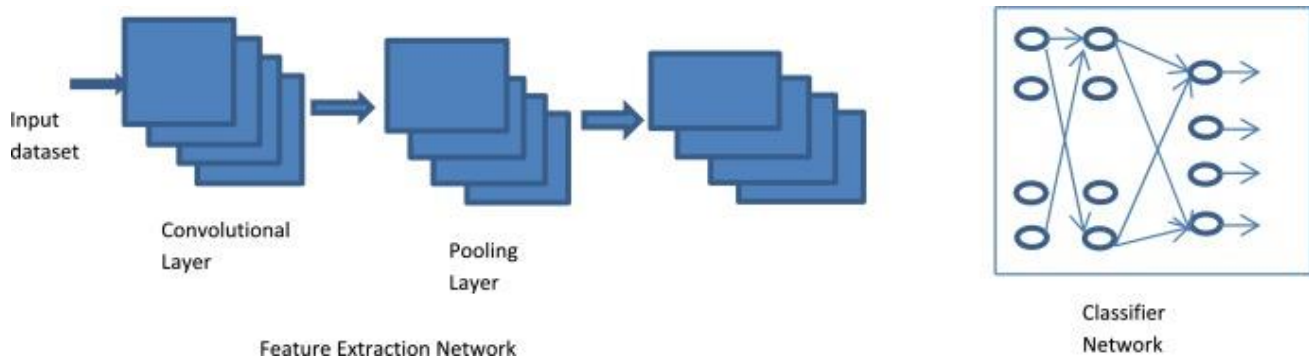
Using the above diagram, we know that there are:

1. Convolution layer where convolution happens.
2. Pooling layer where the pooling process happens
3. Normalization usually with the use of ReLu
4. Fully Connected Layers



Feature Extraction

The true fact is that CNNs provide automatic feature extraction, which is the primary advantage. The specified input data is initially forwarded to a feature extraction network, and then the resultant extracted features are forwarded to a classifier network as shown in Figure. The feature extraction network comprises loads of convolutional and pooling layer pairs. Convolutional layer consists of a collection of digital filters to perform the convolution operation on the input data. The pooling layer is used as a dimensionality reduction layer and decides the threshold. During backpropagation, a number of parameters are required to be adjusted, which in turn minimizes the connections within the neural network architecture.

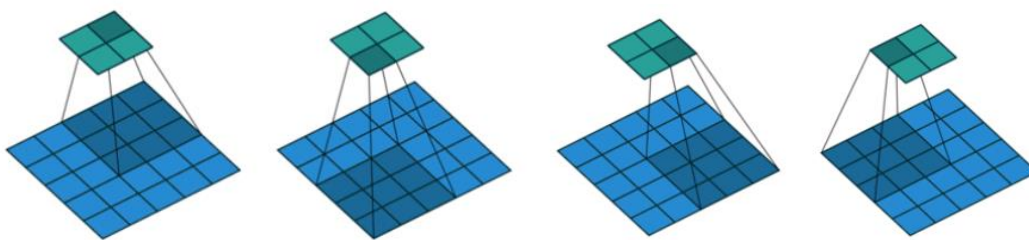


CNN is the model that extracts input image features and another neural network classifies the image features. The input image is used by the feature extraction network. The extracted feature signals are utilized by the neural network for classification. The neural network classification then works on the basis of the image features and produces the output. The neural network for feature extraction includes convolution layer piles and sets of pooling layers. As its name implies, the convolution layer transforms the image using the process of the convolution. It can be described as a series of digital filters. The layer of pooling transforms the neighboring pixels into a single pixel. The pooling layer then decreases the image dimension. As CNN's primary concern is the image, the convolution and pooling layers' procedures are intuitively in a two-dimensional plane. This is one of CNN's distinctions with other neural networks

Variants of the Basic Convolution Function

In practical implementations of the convolution operation, certain modifications are made which deviate from the discrete convolution formula mentioned above:

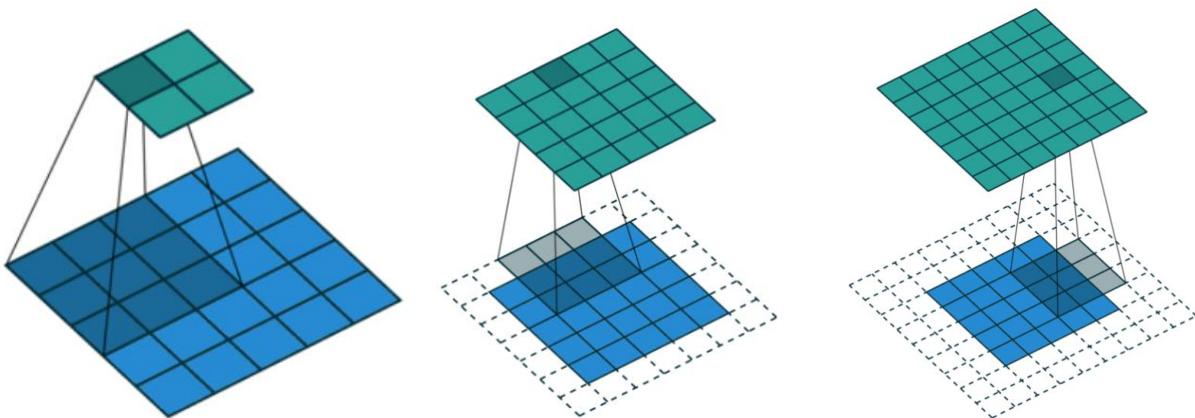
- In general a convolution layer consists of application of several different kernels to the input. This allows the extraction of several different features at all locations in the input. This means that in each layer, a single kernel (filter) isn't applied. Multiple kernels (filters), usually a power of 2, are used as different feature detectors.
- The input is generally not real-valued but instead vector valued (e.g. RGB values at each pixel or the feature values computed by the previous layer at each pixel position). Multi-channel convolutions are commutative only if number of output and input channels is the same.
- In order to allow for calculation of features at a **coarser level** strided convolutions can be used. The effect of strided convolution is the same as that of a convolution followed by a down sampling stage. This can be used to reduce the representation size.



2D convolution 3x3 kernel and stride of 2 units

- Zero padding helps to make output dimensions and kernel size independent. 3 common zero padding strategies are:

1. **valid**: The output is computed only at places where the entire kernel lies inside the input. Essentially, no zero padding is performed. For a kernel of size k in any dimension, the input shape of m in the direction will become $m-k+1$ in the output. This shrinkage restricts architecture depth.
2. **same**: The input is zero padded such that the spatial size of the input and output is same. Essentially, for a dimension where kernel size is k , the input is padded by $k-1$ zeros in that dimension. Since the number of output units connected to border pixels is less than that for centre pixels, it may under-represent border pixels.
3. **full**: The input is padded by enough zeros such that each input pixel is connected to the same number of output units. In terms of test set accuracy, the optimal padding is somewhere between same and valid.



valid(left), **same**(middle) and **full**(right) padding The extreme left one is for stride=2.

- Besides locally-connected layers and tiled convolution, another extension can be to restrict the kernels to operate on certain input channels. One way to implement this is to connect the first m input channels to the first n output channels, the next m input channels to the next n output channels and so on. This method decreases the number of parameters in the model without decreasing the number of output units.
- When max pooling operation is applied to locally connected layer or tiled convolution, the model has the ability to become transformation invariant because adjacent filters have the freedom to

learn a transformed version of the same feature. This is essentially similar to the property leveraged by pooling over channels rather than spatially.

- Bias terms can be used in different ways in the convolution stage. For locally connected layer and tiled convolution, we can use a bias per output unit and kernel respectively. In case of traditional convolution, a single bias term per output channel is used. If the input size is fixed, a bias per output unit may be used to counter the effect of regional image statistics and smaller activations at the boundary due to zero padding.

Summary

- Convolution networks are the types of NN which are useful for data like images, videos or text. This type of data is highly complex and non-linear.
- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
- **Filters are tensor** which keeps track of spatial information and learns to extract features like edge detection, smooth curve, etc of objects in something called a **convolutional layer**.
- The true fact is that CNNs provide automatic feature extraction, which is the primary advantage.
- The feature extraction network comprises loads of convolutional and pooling layer pairs. Convolutional layer consists of a collection of digital filters to perform the convolution operation on the input data.