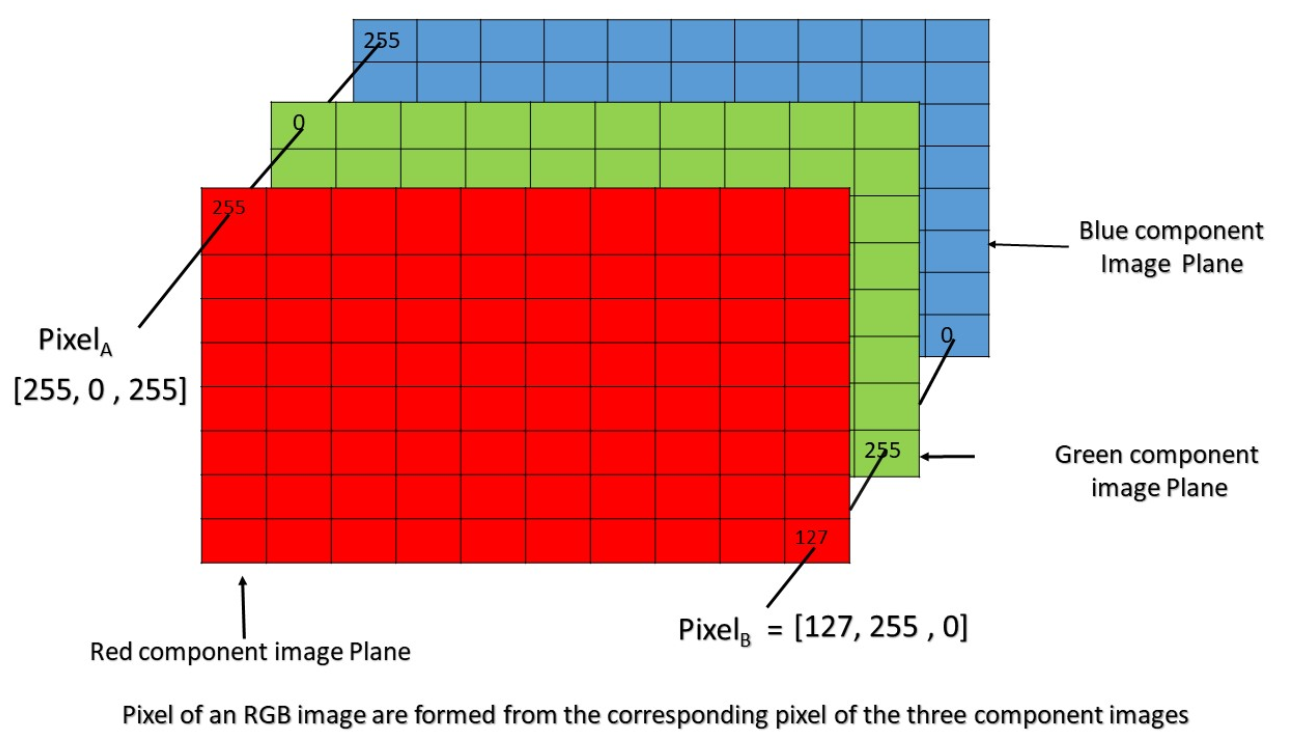
Convolutional Neuron Networks (CNN)

# Introduction

Every image is composed of pixels. Image is represented by a collection of numbers.

For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically, zero is taken to be black, and 255 is taken to be white.

An RGB image is represented with an M-by-N-by3 array where each 3-vector corresponds to the red, green, and blue intensities of each pixel. An RGB image can be viewed as three different images (a red scale image, a green scale image and a blue scale image) stacked on top of each other, and when fed into the red, green and blue inputs of a colour monitor, it produces a colour image on the screen.



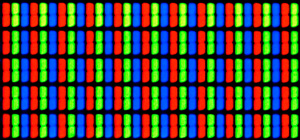


Figure Pixels on LCD monitor extremally enlarged.

How image recognition works using classical Neuron Nets?

The ‘MNIST’ dataset of handwritten digits images is often used in experimentations about image recognition. It contains 60,000 training images and 10,000 testing images. Below is the sample of MNIST dataset.



Figure Sample images from MNIST test dataset

Every image has the same size 28x28 pixels in a grayscale. Values are stored in 2-dimensional array. An array could be something like this:

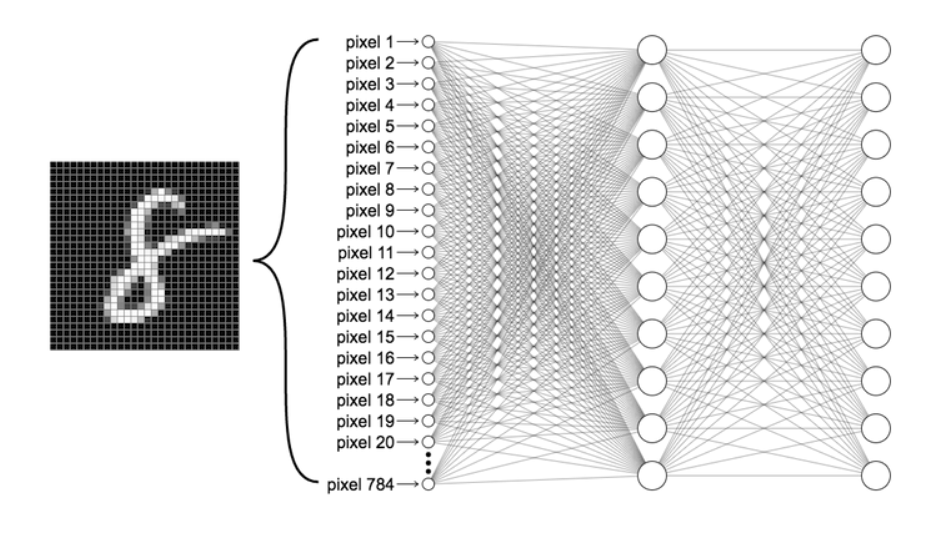
[[0, 225, ..., 180],

[34, 167, ..., 145]

...,

[222, 54, ..., 68]]

Images cannot be simply put into Neuron Nets as 2-dimensional array. A Neuron Net input layer takes input as 1-dimensional array. Those data must be **flattened**, transformed to 1-dimentional array. The image size 28 x 28 pixels after flattening has 784 features in 1-dimensional array. It means, that Neuron Nets must have 784 inputs.

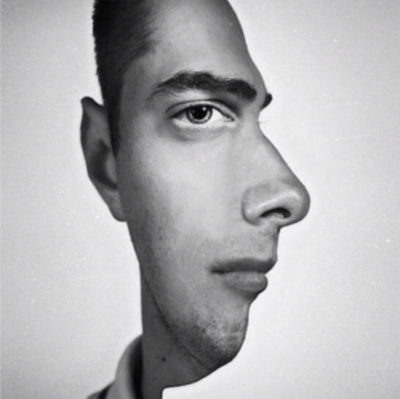


I made an example, how it could be implemented with classic Neuron Nets. It works for those datasets. Somehow it was able to find correct weights and biases.

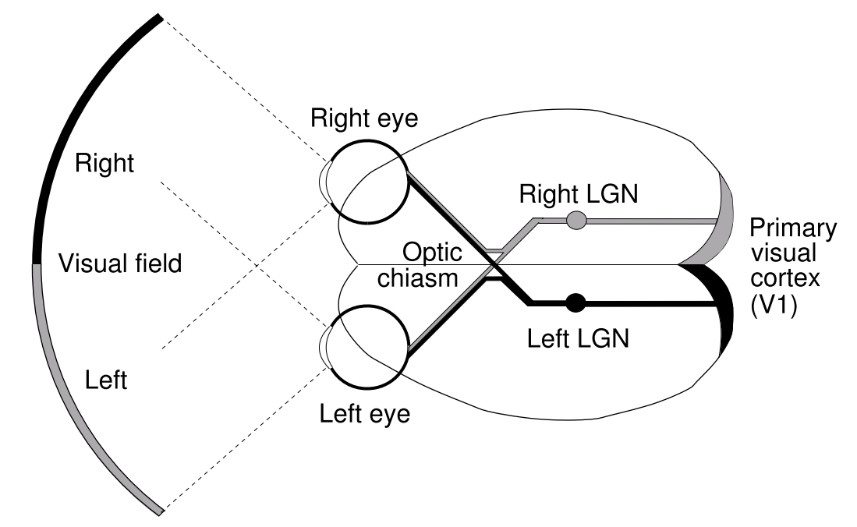
Two reasons why classical NN is not suffice for image recognition:

* Too many features. A small grayscale image of size 28x28 pixels has already 784 features.
* Local features loss. By flattening an image, local features of the image are lost. A local feature of an image is a shape it represents. For a cat image it could be cat’s face, ears, eyes.

# Convolution layer



Biologists discovered that colour and **shape** are processed separately in the early visual cortex and then integrated later. Actually, the newest research proved that some neurons are only activated by either a specific colour or shape, but many other neurons are responsive to a particular colour and shape simultaneously.



Biologists discovered also that optical neuros break down an image to extract a certain shape i.e., curves. lines. They extract those features and in next step they combine it i.e., 4 curves combined into circle. Another layer could extract i.e., lines. Next all of them are combined to create recognizable object.

## Implementation

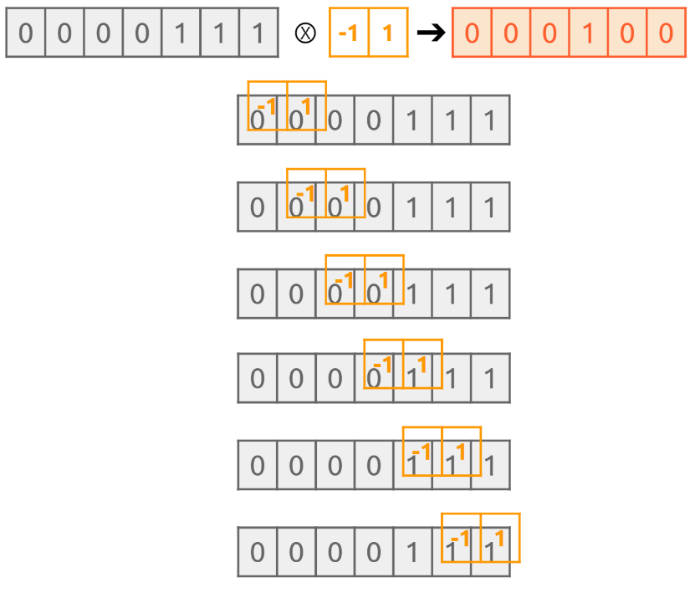
That idea was implemented to Neuron Nets by adding one more layer called **a convolutional layer**. This layer performs an operation called a convolution.

A convolution in mathematics is an operation which takes two functions as input and produces a single function output (much like addition or multiplication of functions).

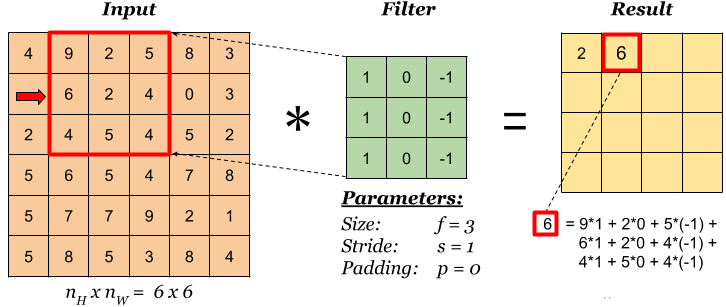
A convolutional layer uses **filters (feature detector)** to extract shapes. A filter is a matrix, usually 3x3 or 5x5. The result of filtering an image is a two-dimensional array which is called a **feature map**. The same concept of filtering is used i.e., in image editing by using different filters.

Below is the example with the filter 1x2.

The product of convolution is ‘shrunk’. The input shape is 1x7, the result shape is 1x6.



Below is the example with the filter 3x3. The input size is 6x6. After applying the filter of size 3x3 with stride=1 and padding=0, the size of the result is 4x4 (down by 1 on each side comparing to the input size).



Convolution parameters:

* **filter size** - usually 3x3 or 5x5
* **stride** - the number of pixels a filter will move each time. In the example above the stride is 1, meaning it moves right (or down) by one pixel. The higher the stride, the greater the image size reduction.
* **padding -** it can be added to the input to prevent the image size reduction. Padding use values 0. In the example above one layer of the zero-padding should be added on each side of the input to prevent the image dimension reduction.

The problem with reducing the image size is that if the image is passed through many convolution layers without padding and with a high stride parameter, the image size shrinks and eventually becomes too small to be useful. On the other hand, it is desirable to lower the number of features that will be passed to a hidden layer.

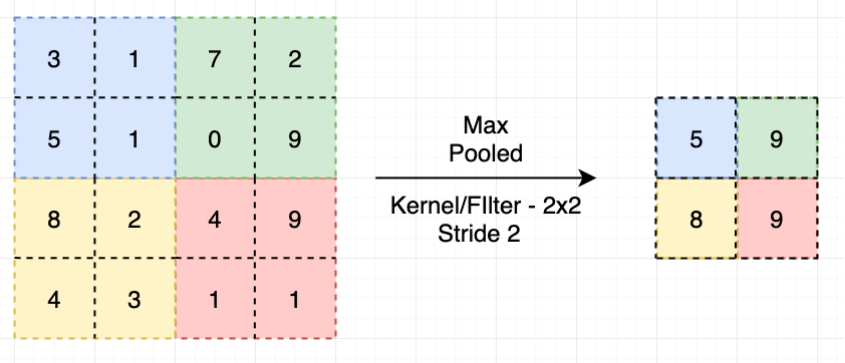
A convolutional layer may use many filters in parallel for a given input to learn multiple features. Each filter uses the same original image. Filters can be handcrafted, but usually an algorithm finds the best set of values for each filter during training process. Values of the filter are like its weights. The network will learn what types of features to extract from the input that are the most useful for classifying images i.e., dogs or cats.

# Max Pooling layer

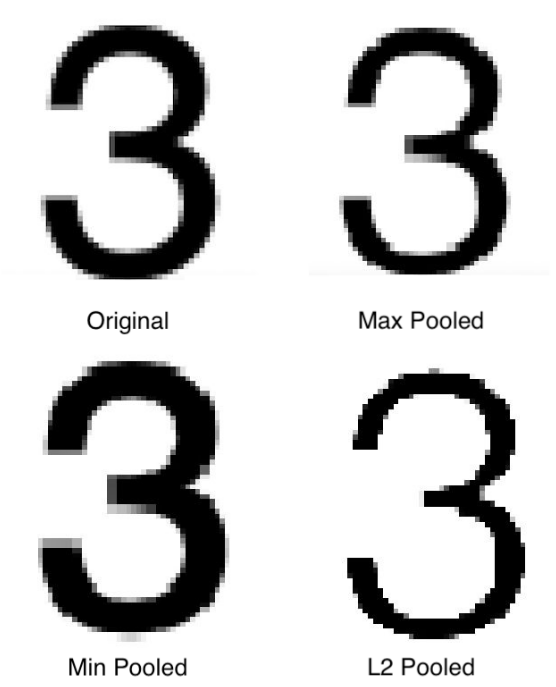
**Max Pooling layer** goes after a convolutional layer. The purpose of Max Pooling layer is to reduce the size of an image. Pooling operation does sub-sampling of the image. The term **subsampling** means an operation performed by a pooling layer.

Max Pooling also uses a filter. The filter is almost always 2×2 pixels with a stride of 2 pixels. The filter returns the maximum value from numbers being filtered.

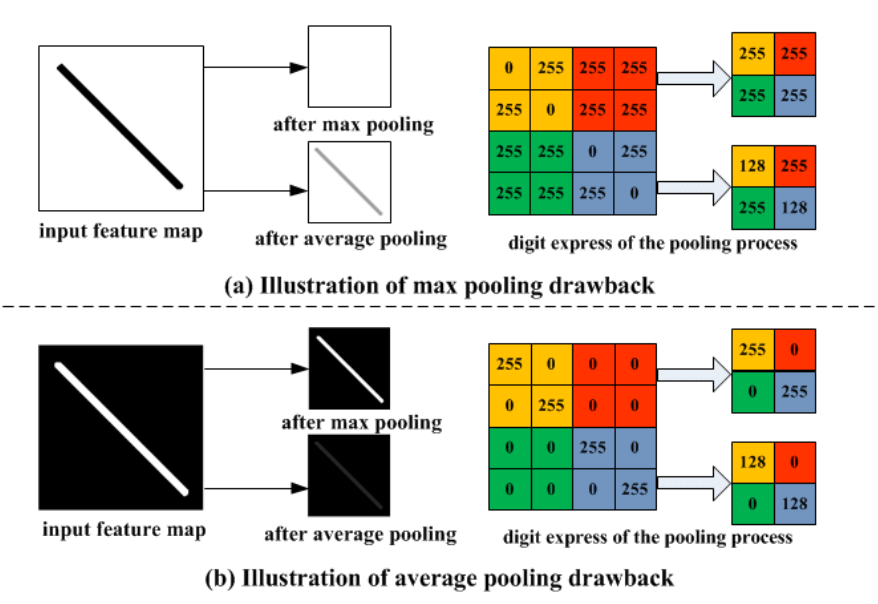
The example below uses the filter 2x2 with the stride=2. The size of the result is 0.5 height and 0.5 width of the input size. 75% reduction of the input size.



Other types of pooling include: Min Pooling, Average Pooling, L2 Pooling (L2 norm, Frobenius norm). The filter in Average Pooling layer returns the average value, etc.

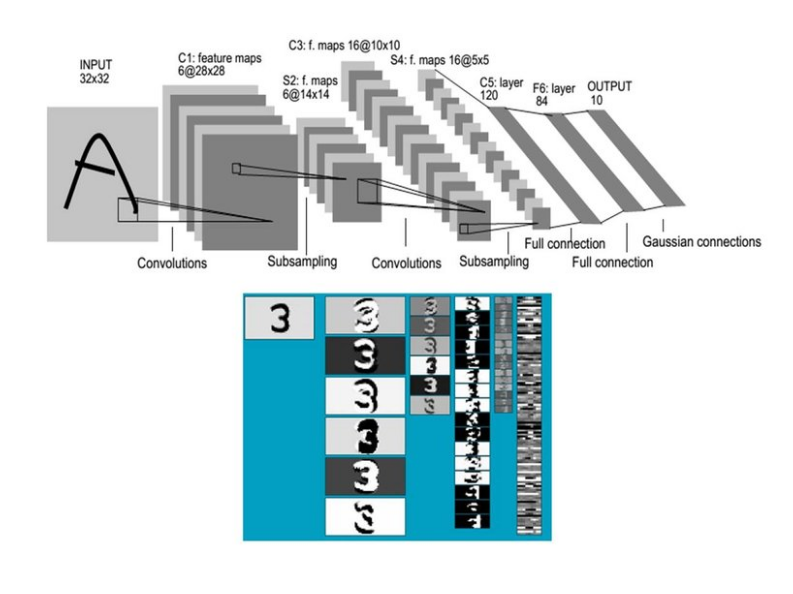
****

Each type of pooling has its drawback:

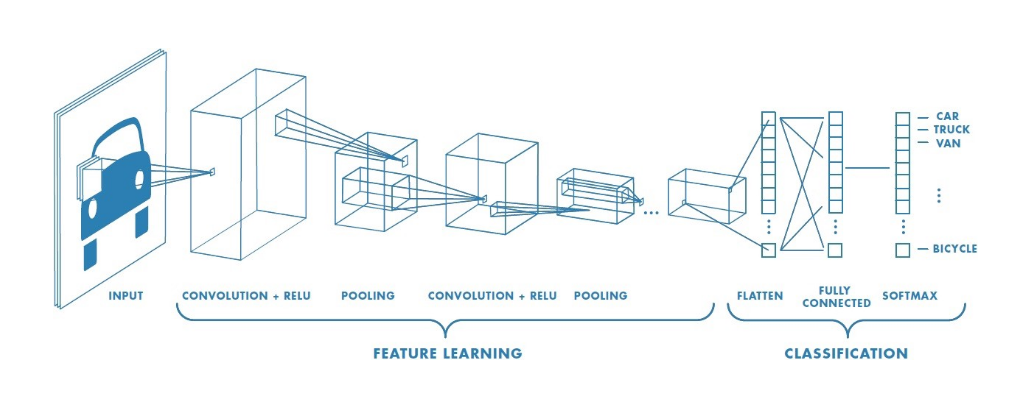


# Multiple Layers

Convolutional layers are not only applied to input data, e.g. raw pixel values, but they can also be applied to the output of other layers (convolution + pooling layer).



# Summary of CNN



The rectified linear activation function (**ReLU**) is the default activation when developing multilayer convolutional neural networks. ReLU overcomes the **vanishing gradient problem**, allowing models to learn faster and perform better.

# Tutorials

First 11 minutes are about convolutional layers: <https://www.youtube.com/watch?app=desktop&v=qPKsVAI_W6M>

Developing CNN from scratch: <https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/>

Note 1: the resolution of images used in above article is very low. Hard to recognize even for human.

Note 2: In the above article pixel values are normalized before training. **Normalization** is used in statistics to eliminate the units of measurement, enabling to compare data more easily from different places. Example: some data are in kilograms while the other is in grams, another one is litres. One way of normalizing values in dataset is rescaling data to have values between 0 and 1. This is usually called feature scaling. It is also known as **Min-Max scaling**. Here is the formula for normalization:

Normalization equation

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.:

* When the value of X is the minimum value in the column, the numerator will be 0, and hence X’ is 0
* When the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X’ is 1

The terms **normalization** and **standardization** are sometimes used interchangeably, but they usually refer to different things. Normalization usually means to scale a variable to have values between 0 and 1, while standardization transforms data to have a mean of zero and a standard deviation of 1.

Standardization is another scaling technique where the values are centred around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation. In this case, the values are not restricted to a particular range. Here is the formula for standardization:

Standardization equation

It is preferred to have normalized data in Machine Learning, because performance is better on normalized data.

Machine learning algorithms like linear regression, logistic regression, neural network, etc. that use gradient descent as an optimization technique require data to be scaled. The value of feature will affect the step size of the gradient descent. The difference in ranges of features will cause different step sizes for each feature. To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, the data must be scaled before feeding it to the model.

# CUDA drivers

**CUDA Nvidia GPU** – By using CUDA, data can be stream to GPU instead of CPU. The CUDA software gives direct access to the GPU. It is developed and implemented in Nvidia graphic cards. Not every Nvidia GPU has CUDA enabled. Here is the list of GPUs that can use it: <https://developer.nvidia.com/cuda-GPUs>

CUDA is usually used for training purposes to speed up the process. Execution of ready model is usually much cheaper than training.