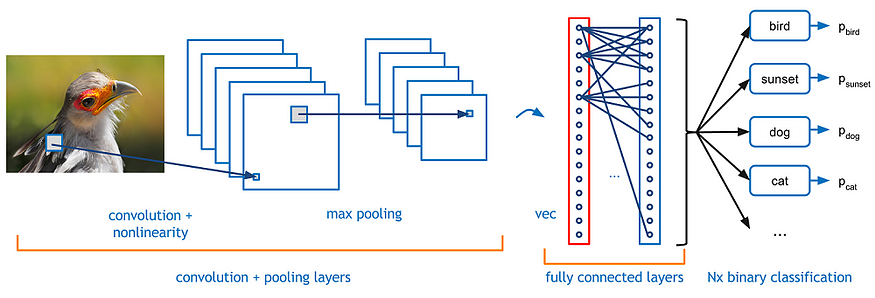
**UNIT-4**

**Convolutional Neural Network**



# 1. What is CNN?

Computer vision is evolving rapidly day-by-day. It’s one of the reason is deep learning. When we talk about computer vision, a term convolutional neural network (abbreviated as CNN) comes in our mind because CNN is heavily used here. Examples of CNN in computer vision are face recognition, image classification etc. It is similar to the basic neural network. CNN also have learnable parameter like neural network i.e, weights, biases etc.

# 2. Why should we use CNN?

## Problem with Feedforward Neural Network

Suppose you are working with MNIST dataset, you know each image in MNIST is 28 x 28 x 1(black & white image contains only 1 channel). Total number of neurons in input layer will 28 x 28 = 784, this can be manageable. What if the size of image is 1000 x 1000 which means you need 10⁶ neurons in input layer. Oh! This seems a huge number of neurons are required for operation. It is computationally ineffective right. So here comes Convolutional Neural Network or CNN. In simple word what CNN does is, it extract the feature of image and convert it into lower dimension without loosing its characteristics. In the following example you can see that initial the size of the image is 224 x 224 x 3. If you proceed without convolution then you need 224 x 224 x 3 = 100, 352 numbers of neurons in input layer but after applying convolution you input tensor dimension is reduced to 1 x 1 x 1000. It means you only need 1000 neurons in first layer of feedforward neural network.

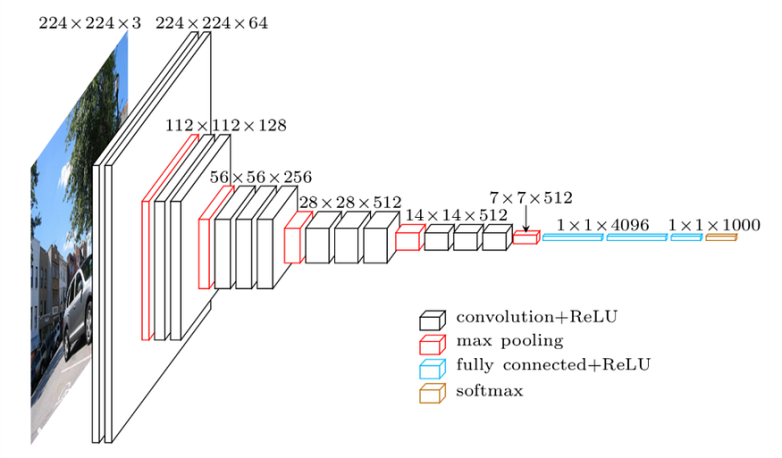


Fig. Down sampling

# 3. Few Definitions

There are few definitions you should know before understanding CNN

## 3.1 Image Representation

Thinking about images, its easy to understand that it has a height and width, so it would make sense to represent the information contained in it with a two dimensional structure (a matrix) until you remember that images have colors, and to add information about the colors, we need another dimension, and that is when Tensors become particularly helpful.

Images are encoded into color channels, the image data is represented into each color intensity in a color channel at a given point, the most common one being RGB, which means Red, Blue and Green. The information contained into an image is the intensity of each channel color into the width and height of the image, just like this

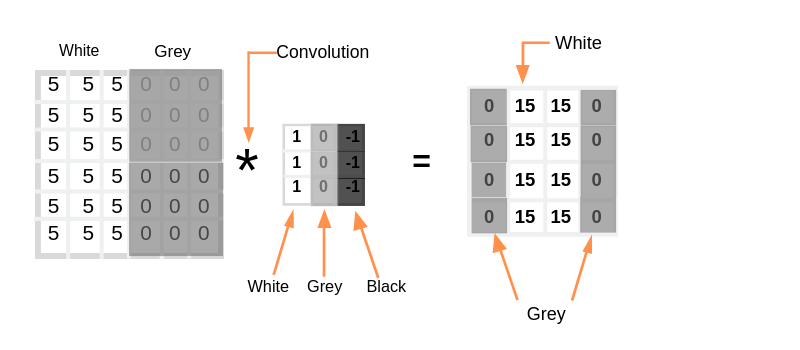


Fig. RGB representation of a image

So the intensity of the red channel at each point with width and height can be represented into a matrix, the same goes for the blue and green channels, so we end up having three matrices, and when these are combined they form a tensor.

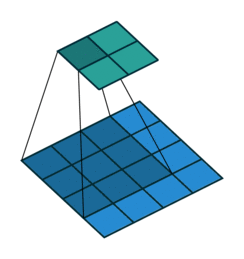
## 3.2 Edge Detection

Every image has vertical and horizontal edges which actually combining to form a image. Convolution operation is used with some filters for detecting edges. Suppose you have gray scale image with dimension 6 x 6 and filter of dimension 3 x 3(say). When 6 x 6 grey scale image convolve with 3 x 3 filter, we get 4 x 4 image. First of all 3 x 3 filter matrix get multiplied with first 3 x 3 size of our grey scale image, then we shift one column right up to end , after that we shift one row and so on.



Convolution operation

The convolution operation can be visualized in the following way. Here our image dimension is 4 x 4 and filter is 3 x 3, hence we are getting output after convolution is 2 x 2.



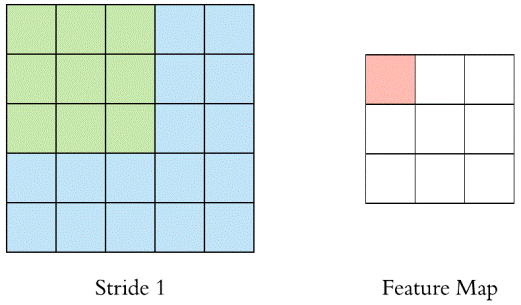
Visualization of convolution

If we have N x N image size and F x F filter size then after convolution result will be

(N x N) \* (F x F) = (N-F+1)x(N-F+1)(Apply this for above case)

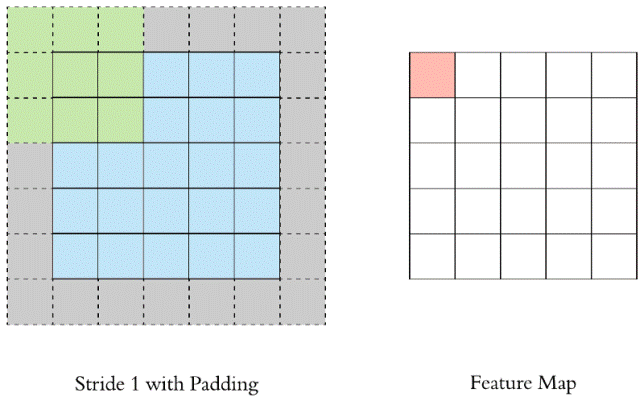
## 3.3 Stride and Padding

Stride denotes how many steps we are moving in each steps in convolution.By default it is one.



Convolution with Stride 1

We can observe that the size of output is smaller that input. To maintain the dimension of output as in input , we use padding. Padding is a process of adding zeros to the input matrix symmetrically. In the following example,the extra grey blocks denote the padding. It is used to make the dimension of output same as input.



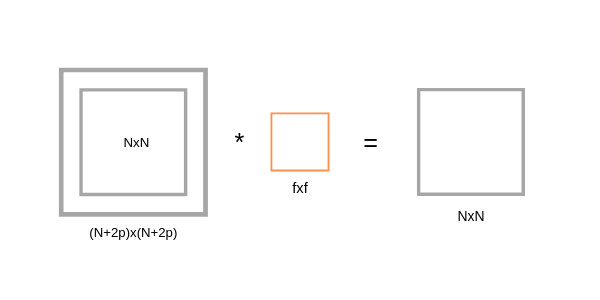
Stride 1 with Padding 1

Let say ‘p’ is the padding

Initially (without padding)

(N x N) \* (F x F) = (N-F+1)x(N-F+1)---(1)

After applying padding



After applying padding

If we apply filter F x F in (N+2p) x (N+2p) input matrix with padding, then we will get output matrix dimension (N+2p-F+1) x (N+2p-F+1). As we know that after applying padding we will get the same dimension as original input dimension (N x N). Hence we have,

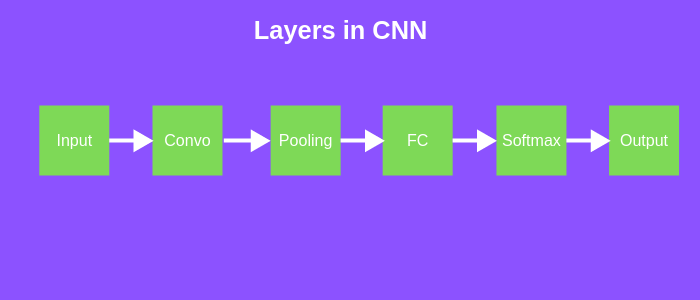
(N+2p-F+1)x(N+2p-F+1) equivalent to NxN  
 N+2p-F+1 = N ---(2)  
 p = (F-1)/2 ---(3)

The equation (3) clearly shows that Padding depends on the dimension of filter.

# 4. Layers in CNN

There are five different layers in CNN

* Input layer
* Convo layer (Convo + ReLU)
* Pooling layer
* Fully connected(FC) layer
* Softmax/logistic layer
* Output layer



**Different layers of CNN**

## 4.1 Input Layer

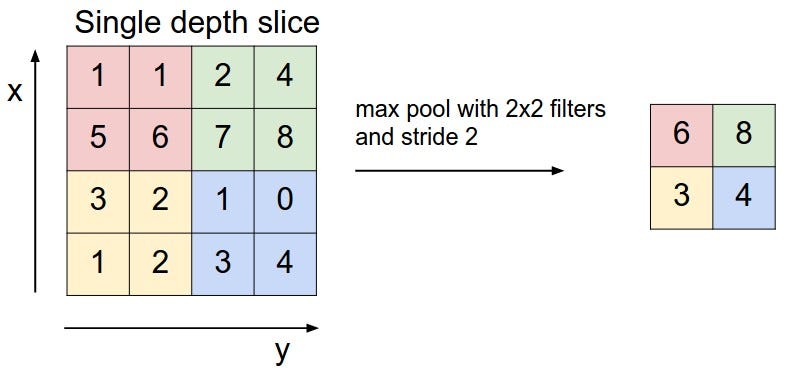
Input layer in CNN should contain image data. Image data is represented by three dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, you need to convert it into 784 x 1 before feeding into input. If you have “m” training examples then dimension of input will be (784, m).

## 4.2. Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field(it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer.

Convo layer also contains ReLU activation to make all negative value to zero.

## 4.3. Pooling Layer



Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layer. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don’t want it. So, the max pooling is only way to reduce the spatial volume of input image. In the above example, we have applied max pooling in single depth slice with Stride of 2. You can observe the 4 x 4 dimension input is reduce to 2 x 2 dimension.

There is no parameter in pooling layer but it has two hyperparameters — Filter(F) and Stride(S).

In general, if we have input dimension W1 x H1 x D1, then

W2 = (W1−F)/S+1

H2 = (H1−F)/S+1

D2 = D1

Where W2, H2 and D2 are the width, height and depth of output.

## 4.4. Fully Connected Layer(FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different category by training.

## 4.5. Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

## 4.6. Output Layer

Output layer contains the label which is in the form of one-hot encoded.

Now you have a good understanding of CNN. Let’s implement a CNN in Keras.

# 5. Keras Implementation

We will use CIFAR-10 dataset to build a CNN image classifier. CIFAR-10 dataset has 10 different labels

* Airplane
* Automobile
* Bird
* Cat
* Deer
* Dog
* Frog
* Horse
* Ship
* Truck

It has 50,000 training data and 10,000 testing image data. Image size in CIFAR-10 is 32 x 32 x 3. It comes with Keras library.

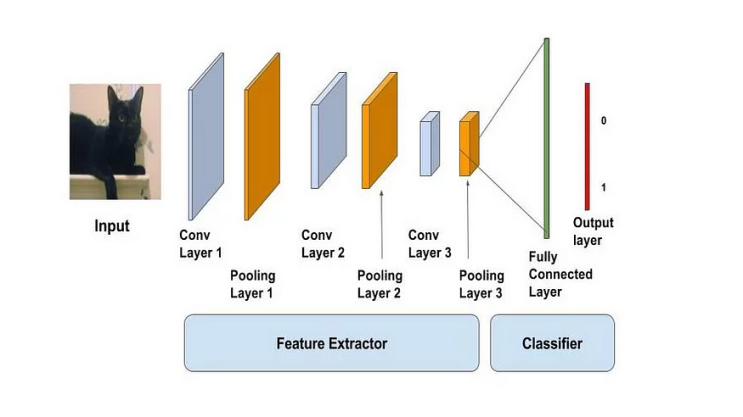


Fig. Model visualization

If you are using google colaboratory, then make sure you are using GPU. To check whether your GPU is on or not. Try following code.

Output:

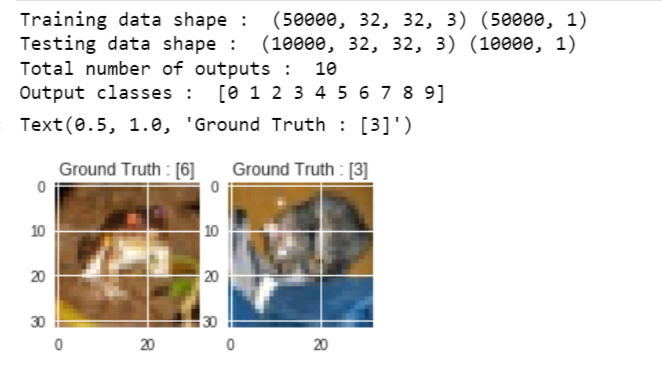
Found GPU at: /device:GPU:0

First of all, import all necessary modules and libraries.

Then load the dataset and split it into train and test sets.

We will print training sample shape, test sample shape and total number of classes present in CIFAR-10. There are 10 classes as we saw earlier. For the sake of example, we will print two example image from training set and test set.

Output:



Find the shape of input image then reshape it into input format for training and testing sets. After that change all datatypes into floats.

Normalize the data between 0–1 by dividing train data and test data with 255 then convert all labels into one-hot vector with to\_catagorical() function.

Display the change for category label using one-hot encoding.

Output:

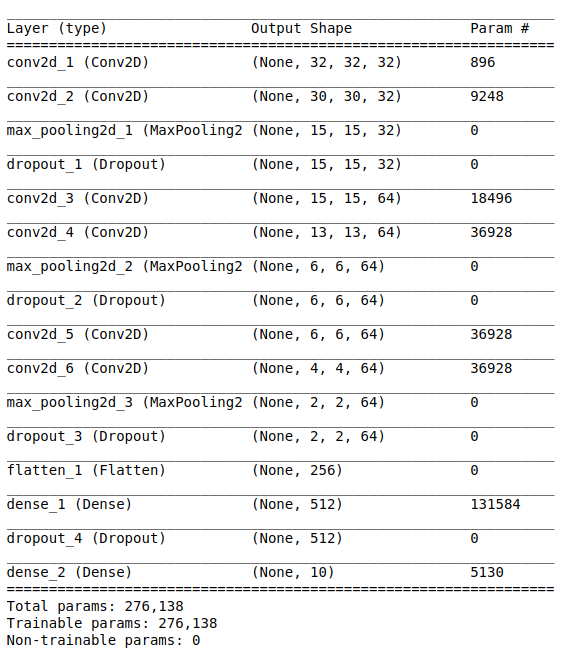
Original label 0 : [6]  
After conversion to categorical ( one-hot ) :   
[0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]

Now create our model. We will add up Convo layers followed by pooling layers. Then we will connect Dense(FC) layer to predict the classes. Input data fed to first Convo layer, output of that Convo layer acts as input for next Convo layer and so on. Finally data is fed to FC layer which try to predict the correct labels.

Initialize all parameters and compile our model with rmsprops optimizer. There are many optimizers for example adam, SGD, GradientDescent, Adagrad, Adadelta and Adamax ,feel free to experiment with it. Here batch is 256 with 50 epochs.

model.summary() is used to see all parameters and shapes in each layers in our models. You can observe that total parameters are 276, 138 and total trainable parameters are 276, 138. Non-trainable parameter is 0.

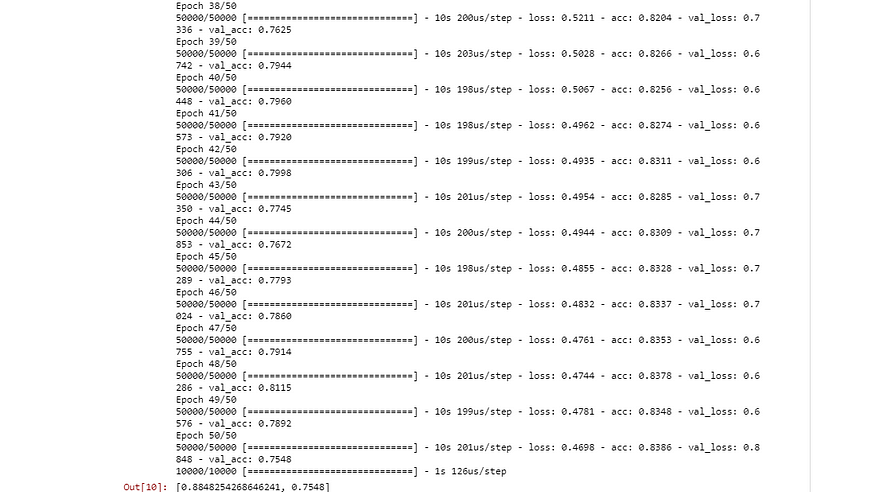
Output:



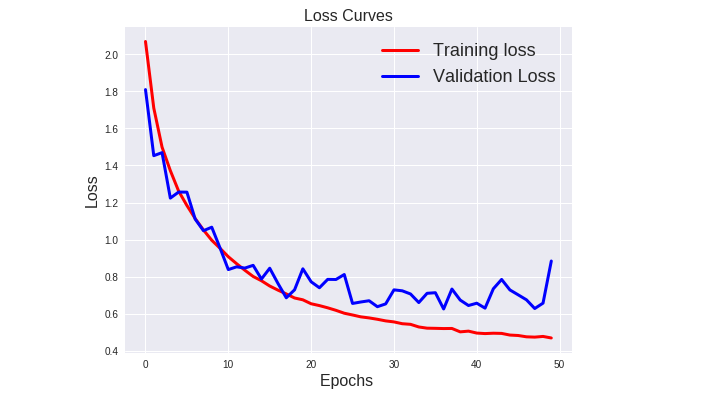
**Model Summary**

After compiling our model, we will train our model by fit() method, then evaluate it.

Output:



After training we got 83.86% accuracy and 75.48% validation accuracy. It is not actually bad at all.



Loss vs Epochs graph Accuracy vs Epochs graph