Assignment 2

By Krishna Chari

For overall understanding of convolution neural networks, you can refer to,

Andrew-NG-Notes/andrewng-p-4-convolutional-neural-network.md at master · ashishpatel26/Andrew-NG-Notes · GitHub

The following document outlines the different steps you will undertake as part of the assignment. I have added a *For Your understanding* when explaining concepts.

The different steps for the assignment,

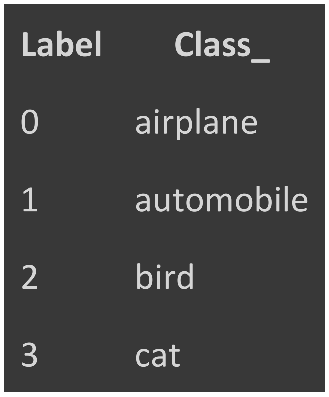
I. Load CIFAR 10 data.

Total of 60000 images, containing 10 classes of images, 6000 images of the same class.

For this assignment, we divide the 60000 into

45000 training images, 5000 validation and 10000 test images.

- **The goal of the assignment is to Classify each of the 10000 test images into one of 10 classes.**
- The classes can be one of,



Label	Class_
0	airplane
1	automobile
2	bird
3	cat

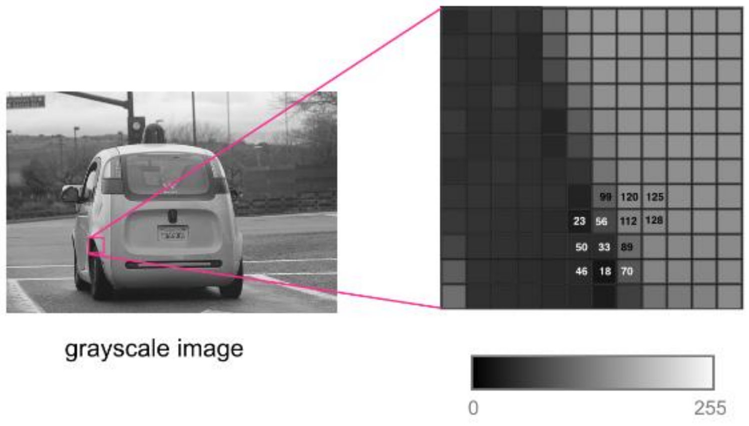
Label	Class_
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

- The training set consists of 50000 images of 32*32*3 representing the height, width and RGB channels.

For Your understanding Images as Grids of Pixels

(From https://cezannec.github.io/Convolutional_Neural_Networks/)

As described in class, an image is seen by a neural network (and by computers) as a grid of numerical values. Below, you'll see a zoomed-in portion of a grayscale image of a car. The image is broken up into a fine grid, and each of the grid cells is called a pixel. For grayscale images, each pixel has a value between 0 and 255, where 0 is black and 255 is white; shades of gray are anywhere in between.



Rules regarding number of channels in the yielded images:

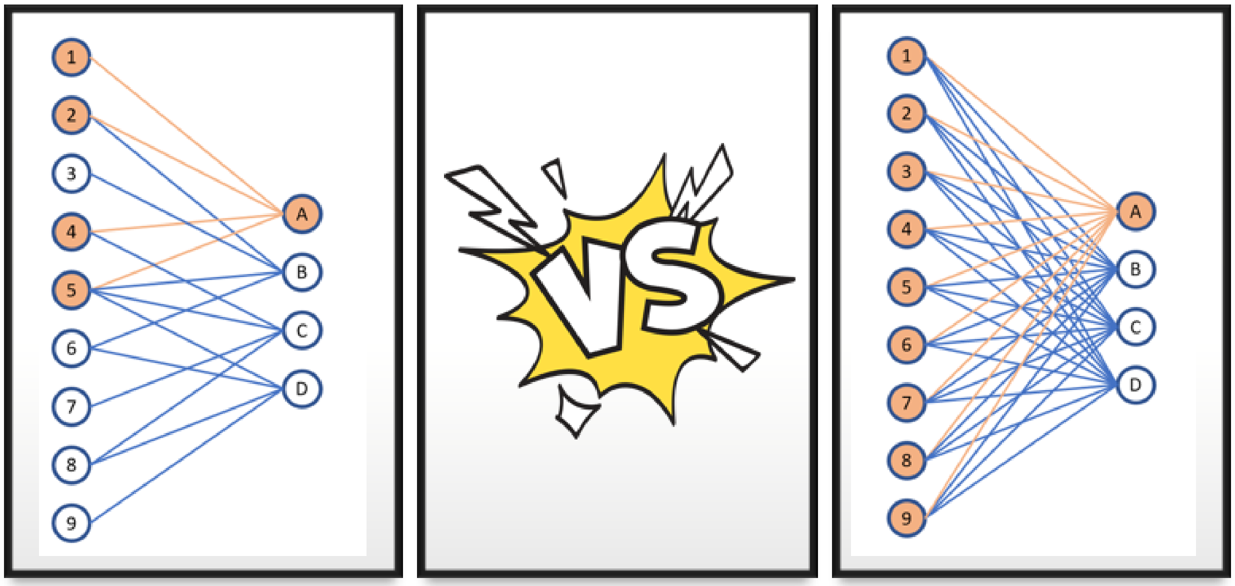
- If color_mode is grayscale, there's 1 channel in the image tensors.
- If color_mode is rgb, there are 3 channels in the image tensors.
- If color_mode is rgba, there are 4 channels in the image tensors.

II. Plot and examine input images and labels.

III. Normalize image so each pixel value is between 0-1 instead of 0-255

This involves first converting the data type from unsigned integers to floats, then dividing the pixel values by the maximum value.

IV. *For Your understanding* **Convolution vs. Dense Layer** (From https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b)



In fully connected layers, the neuron applies a linear transformation to the input vector through a weights' matrix, taking the dot product between the weights matrix W and the input vector x and adding the bias term to this.

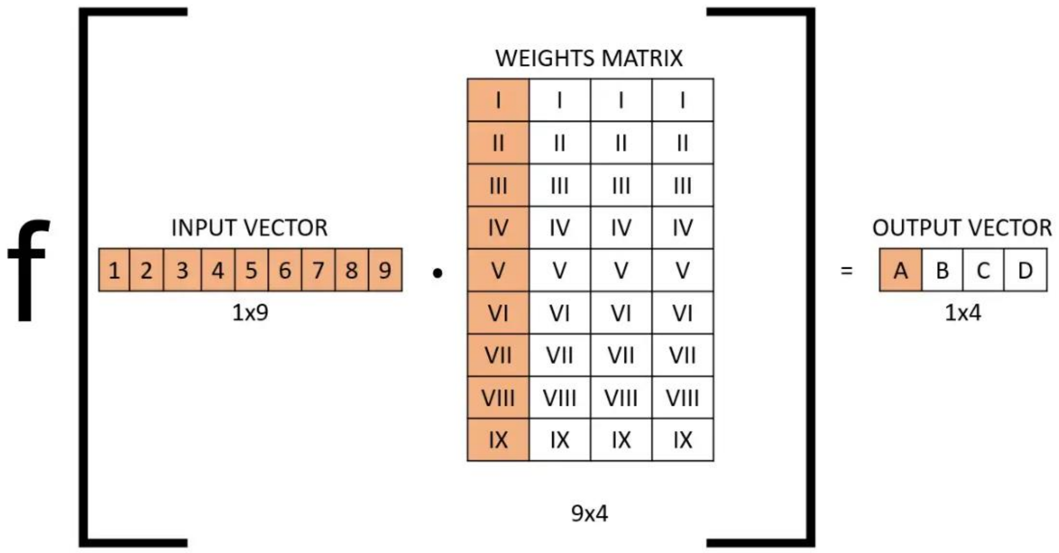


Image by Author

The input is a 1x9 vector, the weights matrix is a 9x4 matrix. By taking the dot product and applying the non-linear transformation with the activation function we get the output vector (1x4).

A convolution is effectively a sliding dot product, where the kernel shifts along the input matrix, and we take the dot product between the two as if they were vectors. Below is the vector form of the convolution shown above. You can see why taking the dot product between the fields in orange outputs a scalar (1x4 • 4x1 = 1x1).

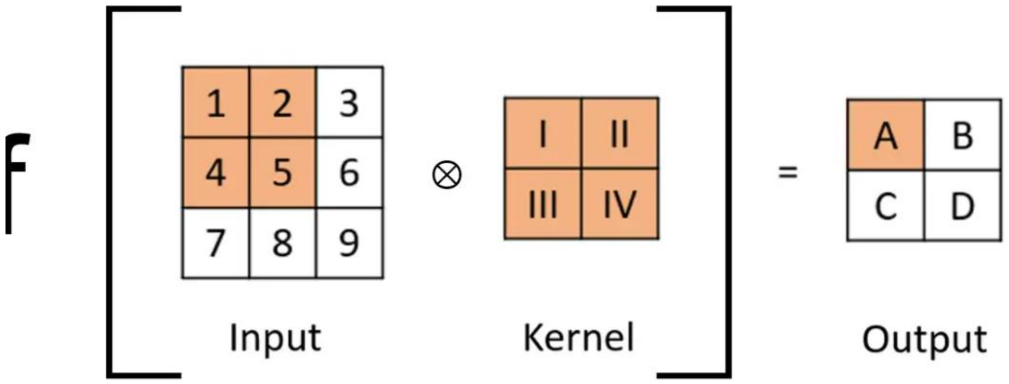


Image by Author

Convolutions are not densely connected, not all input nodes affect all output nodes.

V. *For Your understanding* CNN: Extracts the feature of image and converts it into lower dimension without losing its characteristics.

VI. *For Your understanding* Edge Detection: Apply different filters to detect edges. Refer to. https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123. Every image has vertical and horizontal edges which combine to form an image. Convolution operation is used with some filters for detecting edges.

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

VII. Stride: Stride is the number of steps taken during one convolution, by default it is 1.

VIII. *For Your understanding* Padding:

With convolution,

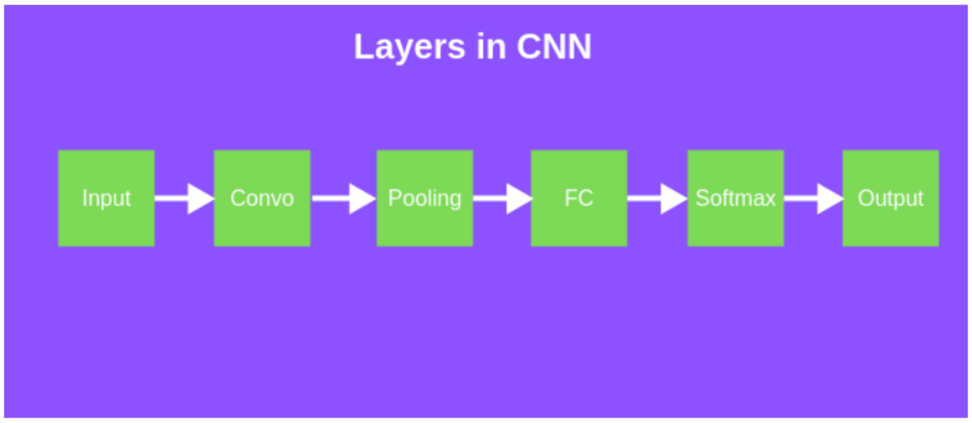
- Every time we apply a convolutional operation, the size of the image shrinks
- Pixels present in the corner of the image are used only a few numbers of times during convolution as compared to the central pixels. Hence, we do not focus too much on the corners since that can lead to information loss
- To overcome these issues, we can pad the image with an additional border, i.e., we add one pixel all around the edges. This means that the input will be an 8 X 8 matrix (instead of a 6 X 6 matrix). Applying convolution of 3 X 3 on it will result in a 6 X 6 matrix which is the original shape of the image.

From
https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

IX. *For Your understanding* Layers in CNN:

- Input Layer
- Convo Layer (Convo + ReLu)
- Pooling
- Fully connected layer
- Softmax/Logistic

- Output

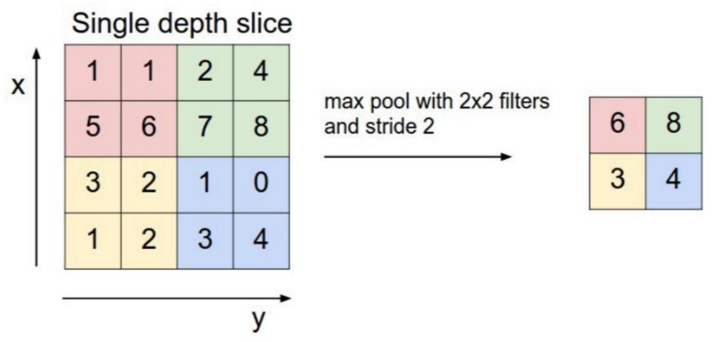


Different layers of CNN

Input Layer is the first layer containing image data. The input layer must be converted to a vector and scaled. A 32 * 32 image will be converted to 784 *1 dimension. Each pixel is then divided by 255 to normalize it. The output Label is one-hot encoded.

Convo Layer is where a filter is applied to an area of the image of the same size as the filter and the dot product is computed. The result value is an integer. Then the receptive area is advanced by the stride length and the filter is applied to the new area. The output will be the input for the next layer. Convo layer also contains ReLU activation to make all negative value to zero.

Pooling Layer: Pooling layer is used to reduce the spatial volume of input image after convolution. Max pooling is only way to reduce the spatial volume of input image.



Source: CS231n Convolutional Neural Network

There are 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.

A 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 into different categories by training.

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.

X. Flattening

To move from the output of a convolutional/pooling layer to a linear layer, you must first flatten your extracted features into a vector. In Keras, this is done by Flatten (),

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

XI. CNN Layers, Summary

CNNs are made of a number of layers: a series of convolutional layers + activation and maxpooling layers, and at least one, final fully connected layer that can produce a set of class scores for a given image. The convolutional layers of a CNN act as feature extractors; they extract shape and color patterns from the pixel values of training images. It’s important to note that the behavior of the convolutional layers, and the features they learn to extract, are defined entirely by the weights that make up the convolutional kernels in the network. A CNN learns to find the best weights during training using a process called backpropagation, which looks at any classification errors that a CNN makes during training, finds which weights in that CNN are responsible for that error, and changes those weights accordingly.

XII. For each experiment, you can add additional convolution layers, add regularization, change weights and so on to arrive at the best model with the lowest bias and accuracy for the test set.