*Traffic Sign Recognition Model Training and Testing Process*

We develop a model for traffic sign recognition for our Self-Driving car project. This model is all about training some image of traffic sign to machine. This helps car to detect a traffic sign on motion. For a self-driving car this is very important. It is very import for a car to detect a sign and acts like it. Where car should stop, limit its speed, the carefulness of the road situation etc. will help a car to act like rationally without a driver.

Now I describe the process how we develop this model as well as description of the usage of python module. We have used Python Programming Language to develop our whole project, because of the library facilities of it.

The things we need to start our journey:

1. Datasets: We need a datasets which help us to training our machine as well as test also. Internet has some good datasets for self-driving car. Kagggle.com is a one of the best website for Data Science related datasets and a community of other data scientists or data science student. We download a German Traffic Sign Recognition datasets for our project.
2. Python and Anaconda: Previously we mentioned why we need Python and Anaconda the purpose of it and installation process.
3. VS Code Editor: This a powerful code editor among the other code editors. Lots of extension features, customization for increasing your productivity. So, I have installed the Jupyter notebook extension for writing code.

Starting Code and How I approach to the process:

1. Importing some important libraries: Initally I have imported numpy(for working with numpy array as it is faster than normal list), pandas(for importing from CSV file), matplotlib(for plotting graph visualization), cv2(for real time video capturing), PIL(this module has an object called Image for working with image file) , tensorflow(for model training and testing).
2. In our datasets there are classes of 43 type images. So need to take import the images with their classes. We need a help of OS module to create path of the images. And we need a PIL module Image object to open the image, resize it into a fixed dimension. Because all the image does not belongs to same dimension. We resize to 30x30. After resizing we append the image in data variable and class id store in labels. We loop the process classes \* no of image in each classes. After importing all the image, we have converted to numpy array of “data, labels”. It helps us for making faster calculation. The shape of data is 39209. That means we have total 39,209 images for calculation.
3. Now we have separated the data (20% for testing and 80% for training) on randomly. So we have 31367 images for training our model and rest 7842 images for testing our model.
4. All the images we are going to use in 3D format. The pixel values range (0-255). We can convert to range (0-1) by dividing 255. This is called scaling. This process saves lots of time when we train and test our model. So test and train image are scaled.
5. Now comes the model designing part to training. We need to import keras module from tensorflow package. We use keras.models.Sequential method for fixing the input layes, hidden layer, neurons. We have used Convolutional Neural Network(CNN) for our image classification. [See description about CNN before this article.] We have added filters in Conv2D for image feature detectors. We have addressed the activation method, padding size and input shape. Once input shape declared we do not need further definition of it. Now we have added pooling layer. This helps to reduce the size of image. We have used the max-pooling. It takes the maximum number from our specified matrix. So have Conv2d-Maxpooling2D for 3 times with different filter but activation function, kernel-size are same. Then we have flatten this and added two dense layers. We have used activation function ‘softmax’ for better accuracy at the last layer.
6. Now compilation process starts, here we have specified some parameters. Like we have used sparse\_categorical\_crossentropy for loss. Because the images are already classified into folder basis. Others parameters like for optimizer we used adam and accuracy metrics.
7. We are going to start our model training. Model.Fit() method has included some parameters – x\_train\_scaled as image data set, y\_train as class id, epochs size=10. This process takes some time for finishing 10 epochs. After successful epochs, our model is trained. We have got 99.67% accuracy and 0.0131% loss.
8. We have evaluated this model with x\_test data set. We have got 98.63% accuracy and 0.09 loss.
9. The test data set we have evaluated this is from split from train data set. We have to use the some random test data set for actual evaluation. For this, in our dataset there is some image for testing.
10. This dataset is not formatted. There is a csv file including image path and class id. So we have defined a method to import the images and respective class id.
11. There are 12,631 images in test dataset. In model evaluation we have got 88.16% accuracy.
12. We have addressed the classes with a name. Then defined a method to test the images individually; Is the image can detect sign?