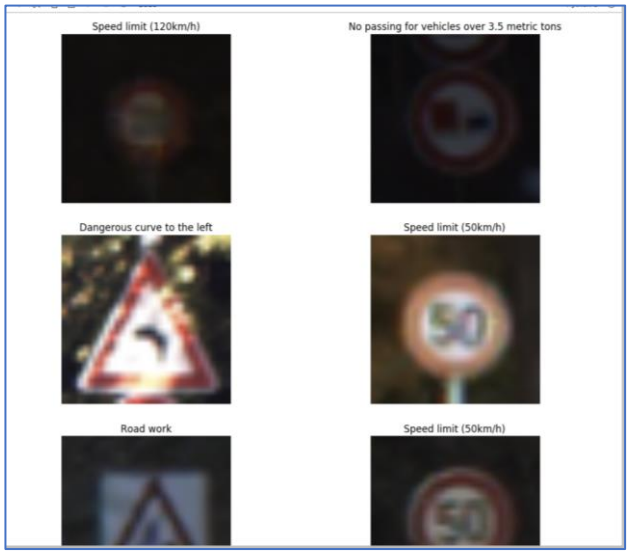
CarND-Project3-Traffic-sign-classifier

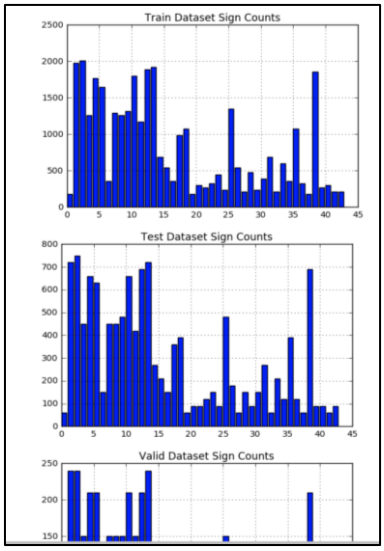
I sincerely thank all friend who have published their hard work in Github on this topic. Their works not only helped me to understand about the project more clearly, but also guided me to consolidate my coding work on this project and complete this document.

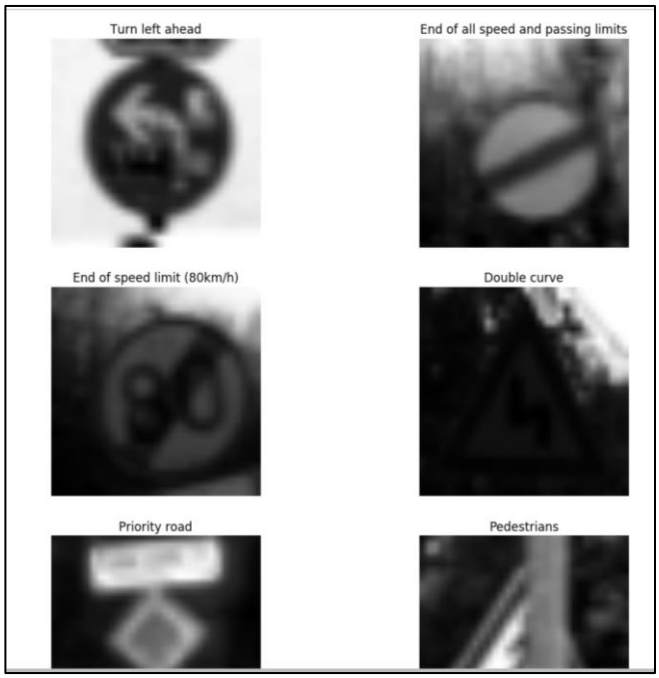
1. In this project, I target to consolidate a code file "test_model.py" that output result information (in the terminal) and images (in the pop-up screen) of various states such as:-
    a. Downloading of Training, Validation and Test datasets
    b. Randomly checking of the data for familiarization with the variance in the images
    c. Pre-processing the data for designing a model
    d. Developing the model
    e. Training and validation of the model
    f. Testing the model with new data

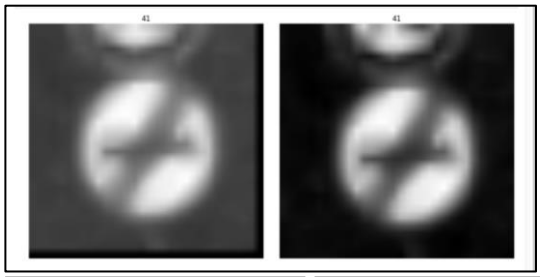
2. Downloading of Training, Validation and Test dataset
    a. The datasets are downloaded from "German Traffic Sign Recognition Benchmark Dataset (GTSRB)". The final images are in ".ppm" format
    b. The same stored in this folder as files "test.p", "valid.p" and "test.p"
    c. Their contents are as follows
        - Number of training example - 34799
        - Number of testing examples - 12630
        - Image data shape - (32, 32, 3)
        - Number of classes - 43

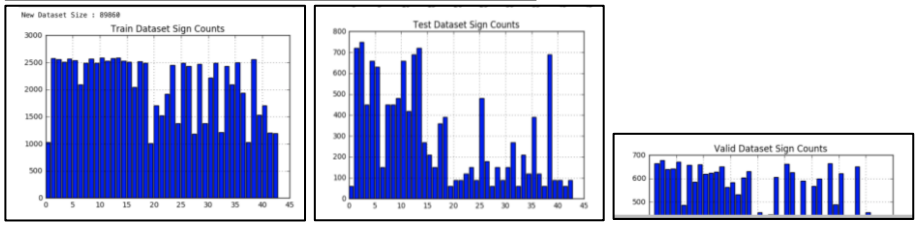
3. Random checking of the data for familiarization with the variance in images
    a. Libraries used for import are matplotlib.pyplot, random and csv.
    b. Also a file called "signnames.csv" is used to provide corresponding labels images in training, valid and test dataset
    c. Finally the pictures are plot on a 4 X 2 style
    d. Randomly selected images look like below. Histogram of images also below
    e. The images reflect various light conditions and also aspect ratio to simulate reality



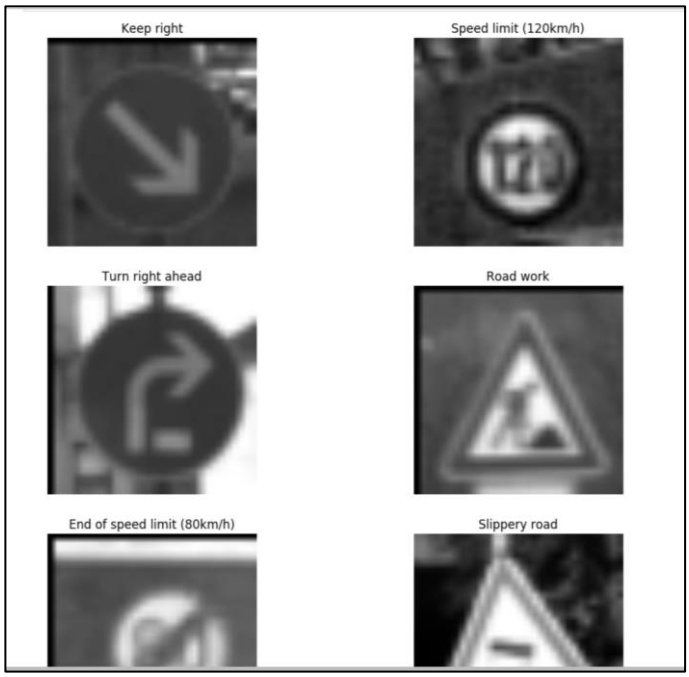


4. Pre-processing the data for designing a model
   a. 3 main actions done here are Grayscaling, Perspective correction and normalization.
   b. For Grayscale, Important libraries used here for import as tensorflow, math & sklearn.utils. After grayscale, there should not be any loss of images. A random sample display of grayscale images as follows
   c. 

   d. After Grayscale, the aspect ratio correction includes perspective, warp and tilt. The resulting random images as follows. The count of the images in the dataset are again checked through histogram to ensure no loss
   e. 

   f. 

   g. Finally the normalization is done to put the important part of the images at center as much as possible. This is mainly required as images are taken from various angles and centricity of important features in the images are always not at the center. A random selection of images after normalization is as follows

h. 

5. Developing the model
   a. Main functions of tensorflow involved here are conv2d, maxpool, reshape, relu and dropout
   b. Other details used here are as follows

c.

Layer	Description
Input	32x32x1 grayscale image
Convolution 5x5	2x2 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	2x2 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Convolution 1x1	2x2 stride, valid padding, outputs 1x1x412
RELU	
Fully connected	input 412, output 122
RELU	
Dropout	50% keep
Fully connected	input 122, output 84
RELU	
Dropout	50% keep
Fully connected	input 84, output 43

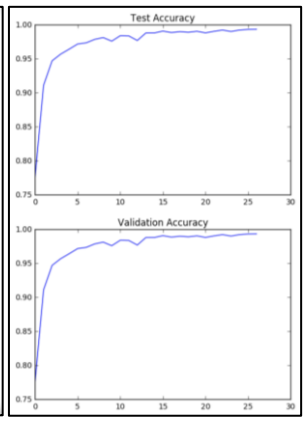
6. For training the model, following premises were used
   a. EPOCHS = 27, BATCH_SIZE = 156 and rate = 0.00097
   b. Functions / steps such as below are used to trained the model
      a. 2 empty array are made to house images (32 x 32 pixcel, RGB) and their Labels
      b. Images flattened to create convolutional layer and Maxpool layers
      c. Logits are defined using input as flattened images, signal classes, Relu, …
      d. Argmax function is used to find the logits of highest value

e. Loss (between the actual & iterative target of logits) is determined using softmax cross entropy
f. The loss is reduced using AdamOptimizer
g. Finally, all the global variables are initialized (set back to original value) for the next round of Epoch. In this way, all 27 round of EPOCHS are completed.

c. The result of the training is as follows:-

d.

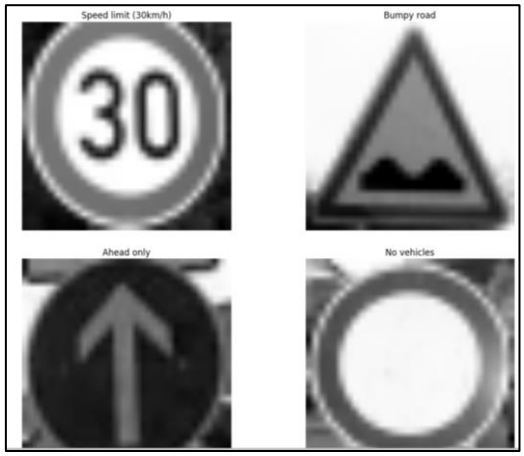
```
Test Accuracy = 1.000
Validation Accuracy = 0.992

EPOCH 26 ...
Test Accuracy = 1.000
Validation Accuracy = 0.993

EPOCH 27 ...
Test Accuracy = 1.000
Validation Accuracy = 0.993

Model saved
```



e. `Finally the accuracies found are :` Train Accuracy = 1.000, Valid A ccuracy = 0.993, Test Accuracy = 0.942

7. Testing of the model with new data
   a. 2 important input here are – signnames.csv and 5 new images downloaded from internet and stored in folder “mysigns”
   b. The output of above looks like below – in normal and grayscale view

c.



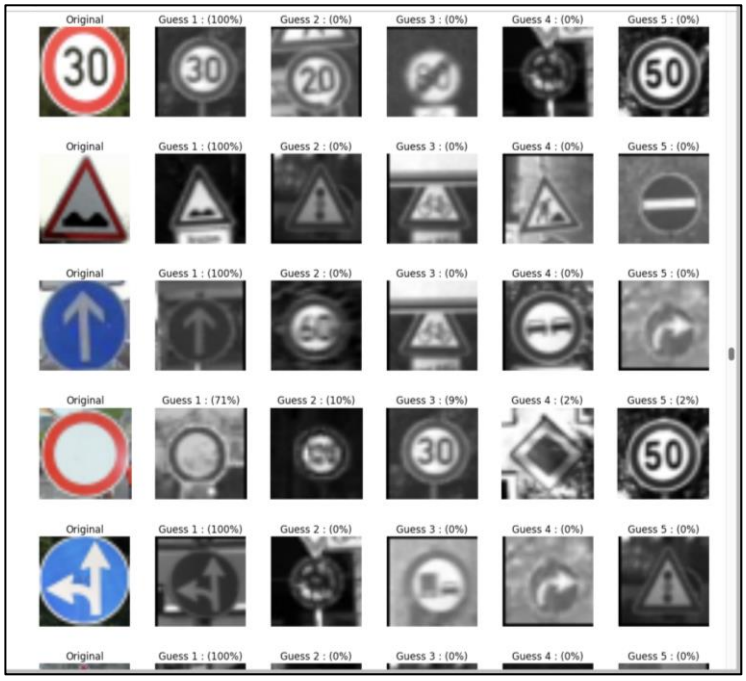
   d. Accuracy of the above new data is checked for 100% (1.00)
   e. Then accuracy of each image is checked with the trained model and following output is obtained

f.

```
Image 1
Image Accuracy = 1.000

Image 2
Image Accuracy = 1.000

Image 3
Image Accuracy = 1.000

Image 4
Image Accuracy = 1.000

Image 5
Image Accuracy = 1.000

Image 6
Image Accuracy = 1.000
```

g. Finally for each of the test image, the model's softmax probability is shown. The output is limited to 5 top probabilities. The result is as follows

h.



8. Conclusion

It was great experience in understanding, digesting and finally trying to write the code for this project.

Based on the little knowledge that I gained from this exercise, I would like to further try the following situations:-

a. Recognizing the traffic sign from pictures where many more items are there like human, car, houses,…
b. Doing the same from video clip