###Writeup / README

####1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.

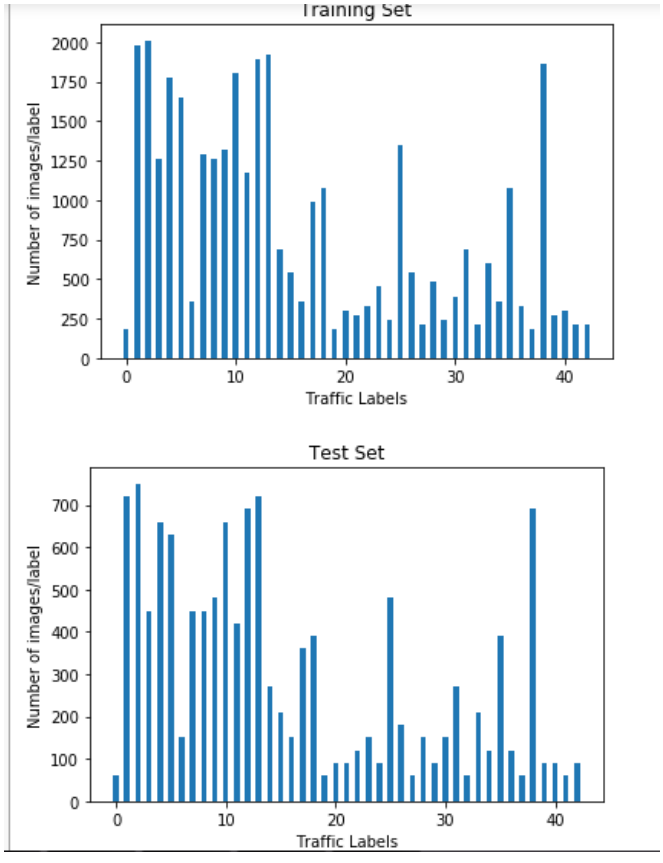
You're reading it! and here is a link to my project code

###Data Set Summary & Exploration

####1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.
I used numpy library to get the below data:

1. Number of training examples = 34799
2. Number of validation examples = 4410
3. Number of testing examples = 12630
4. Image data shape = (32, 32, 3)
5. Number of classes = 43

The bar plot analysis :



###Design and Test a Model Architecture

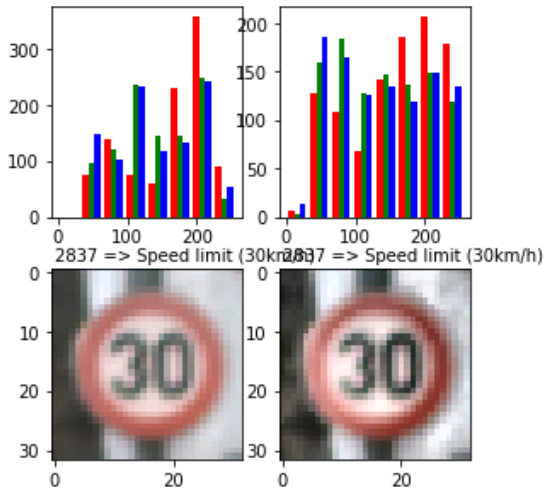
####1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional data for training, describe why you decided to generate additional data, how you generated the data, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

Plotting the data for analysis :

1. **plotHistogram** function shall return the histogram of an image
2. **normalizeImage** shall return the normalized image or histogram equalisation using Clahe
3. **plotRandSymbols** shall return random symbol in the set provided to the function

The purpose of Histogram equalization using Clahe is to increase the contrast of some images which are very dull and difficult to identify. This is depicted as in below picture with and without histogram equalization

```
: #Comparison of histograms after normalization
  plt.figure(figsize=(14,5))
  #random.seed(4000)
  rand = random.randint(0,4000)
  plotHistogram(X_train,y_train, rand , 6, 1);
  plotHistogram(X_train_Norm,y_train, rand, 6, 2);
```



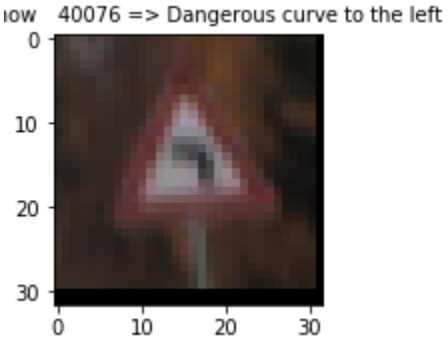
####2. Include an exploratory visualization of the dataset.
Since the dataset has different number of images per classifier, it will be a problem for model training or the efficiency of learning process (weights), the idea of generating additional data set is as follows:

1. Augmentation of data either through adding **Blurred images, Image rotation, image shifting**.

The three methods are included in my jupyter notebook.

A sample dataset before and after data augmentation is as shown below :

→Rotated image



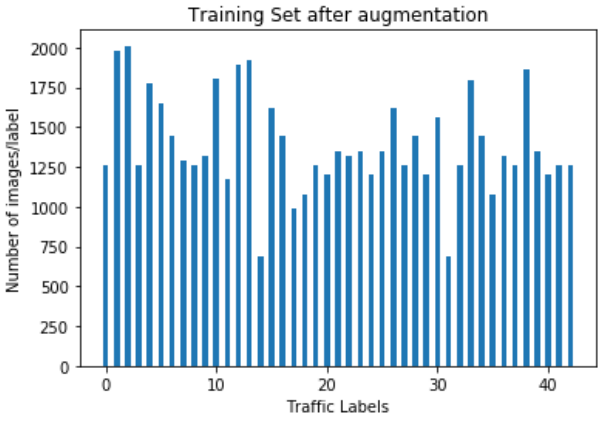
→ Shifted image

The first image is rotated and second image is shifted.

Classes augmented 0,6,15,16,19,20,21,22,23,24,26,27,28,29,30,32,33,34,36,37,39,40,41,42

Expected number of samples taken as 1000 and additional images are generated.

Since data augmentation is done now, training set images are increased from 34799 to 59727



59727

Data is rescaled between not in the range 0..1 but rather -0.5….+0.5, see function **Imgrescale.** This helps model to train quicker and avoids overfitting.

####2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.
Initialisation of weights with mu=0 and sigma=0.1, as in LeNet-Lab
My final model consisted of the following layers:
I followed the LeNet architecture and it is as follows.

1. first convolution layer takes 3 channel inpu, 5x5x3, output depth 12, activation with Relu
2. 2x2 maxpool reduces to 14x14x12
3. Second convolution layer: 7x7x12, output depth 28, activation with Relu
4. 2x2 maxpool reduces to 4x4x28
5. Fully connected 1st layer outputs 240
6. Activation with softsign
7. Fully connected 2nd layer outputs 124
8. Activation with softsign
9. Fully connected 3rd layer outputs the number of classes 43
10. For training, the output layer gets an activation with softsign

####3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyperparameters such as learning rate.
To train the model :

- Optimizer used : Adam Optimizer
- Final batch Size = 512
- Final Epochs : 45 epochs
- Final learning rate = 0.0020

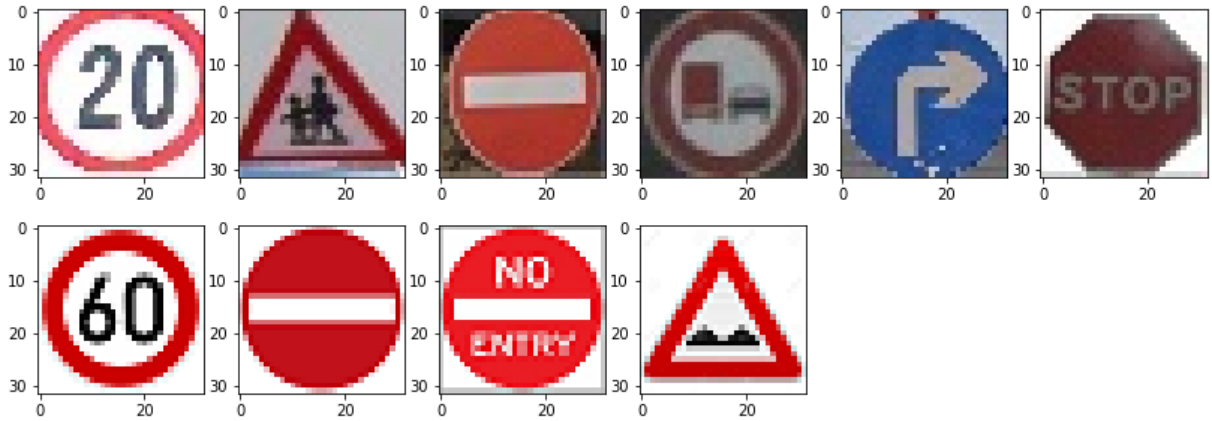
The normalization of -0.5...+0.5,there is a significant improvement of the validation accuracy in the first two epochs (0.738 to 0.820) which is quite interesting, later on this shoot behavior is not seen with the upcoming Epochs. In some of the epochs, the accuracy is fluctuating which is quite strange. The final conclusion was derived by playing with Epochs and Batch size. With the current data provided above, the validation accuracy was found to be quite acceptable (approx 97%)

The test accuracy is 0.938

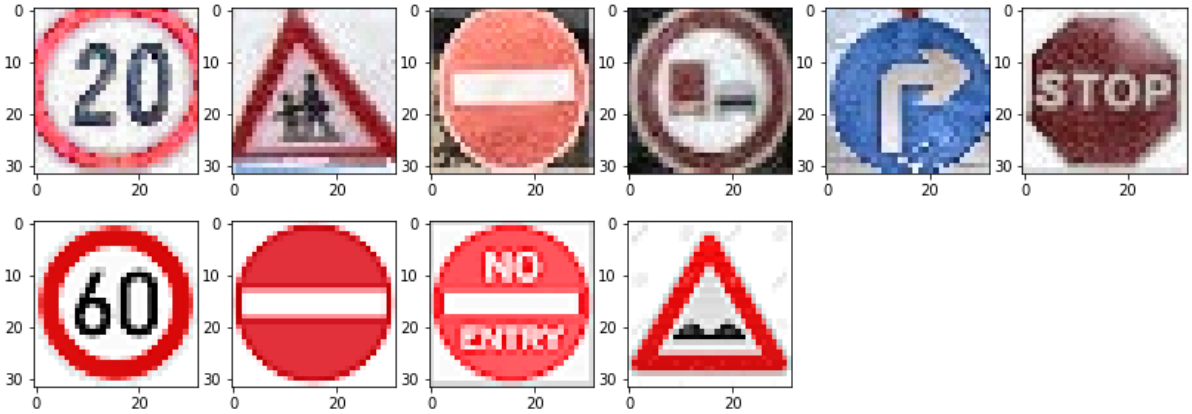
###Test a Model on New Images

####1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

Here are German traffic signs that I found on the web: about 10 images of 32X32X3 in png format.



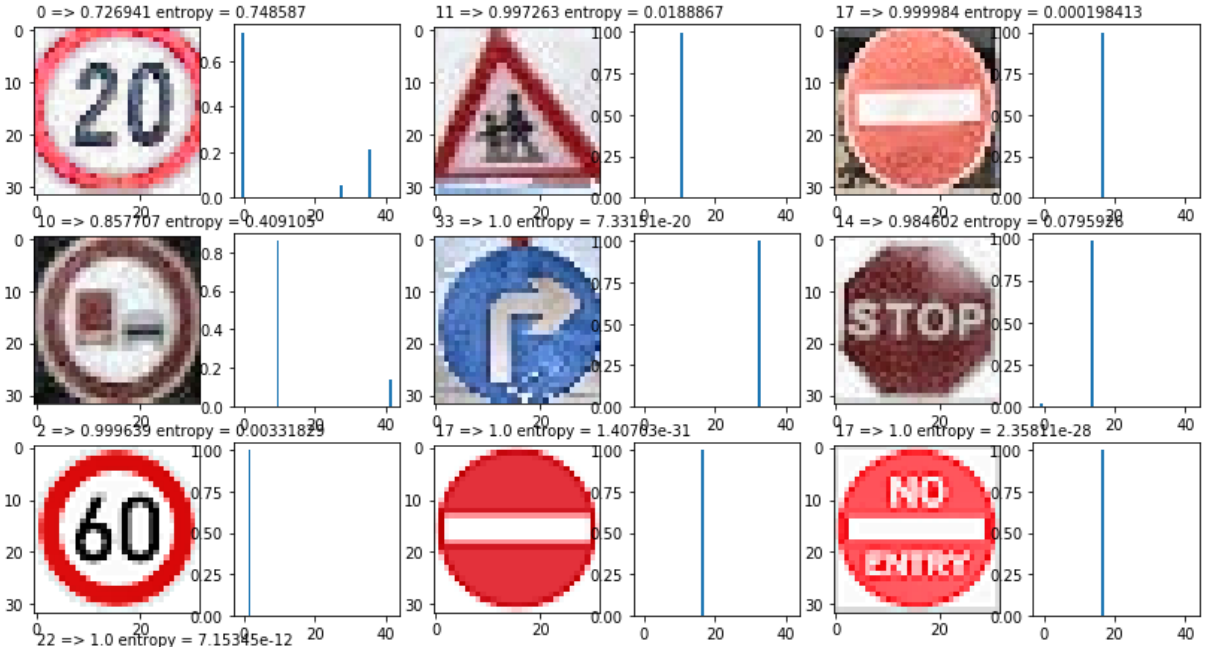
These images are normalized (histogram equalization) and rescaled to -0.5…+0.5



####2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL: Discuss the results in more detail as described in the "Stand Out Suggestions" part of the rubric).

Here are the results of the prediction:

1. 20kmh is detected correctly
2. Children crossing is detected wrongly
3. There are three “No Entry” symbols in the test data. Which are identified perfectly even though in one symbol, there is a writing of it.
4. Stop is also recognized correctly.
5. Unfortunately 60kmh is detected wrongly. This is quite surprising for me because the image is quite clearly indicated.
6. Turn right ahead is also identified correctly.
7. Bumpy road is identified correctly
8. No passing for vehicles over 3.5 metric tons is identified correctly.



In total, according to the number of inputs that I provided for detection, out of 10, 8 are identified correctly which approximates to 80% detection accuracy for new images.

The test accuracy on the test set is 93.8%

Due to the problems that I faced in saving the session, I had to run all the calculations in a single session. Else I should have restored the session and do my calculations. Still I am in the work of solving that error. Sorry for that! But since I am diving into these topics, I am working hard in this area.

(Optional) Visualizing the Neural Network (See Step 4 of the Ipython notebook for more details)

####Improvisation or observations:

1. Converting the images to Greyscale could help train the model quickly
2. Due to data augmentation, the number of training data set is increased from the set provided. This can be reduced again and could be checked for accuracy
3. The training has been carried out few times and the accuracy was approx 98%, this could be due to the fact that everytime the model is trained well from the shuffle function. So, this plays also an important role when it comes to number of epochs, batch size, shuffling the data.