Traffic Sign Classifier

The goals / steps of this project are the following:

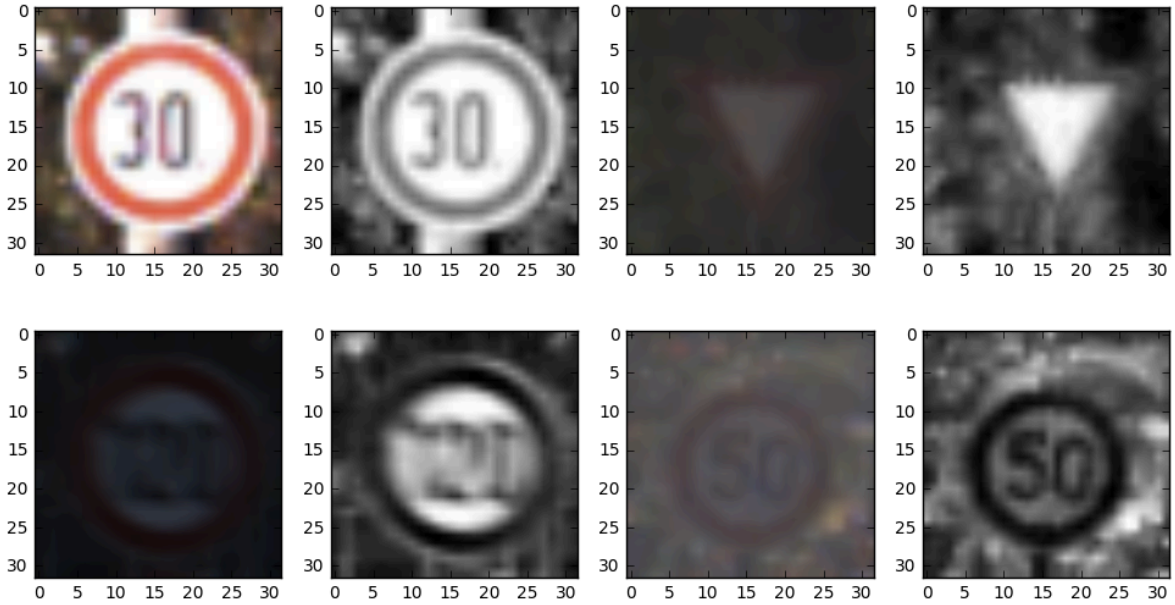
1. Load the data set (see below for links to the project data set)
2. Explore, summarize and visualize the data set
3. Design, train and test a model architecture
4. Use the model to make predictions on new images
5. Analyze the softmax probabilities of the new images
6. Further improvements

1) Load the data set
   Data set is loaded via a pickle file provided by Udacity. This contains a training, validation and label set

2) Data exploration
   Data exploration is done in the cell 2 and 3 of the ipython notebook. I have used python len and shape methods to figure out the size of the data sets. In Cell 3, I have plotted a image of each class to get a better idea how the datat actually looks like.

3) Model training and testing
   Started off with a basic Lenet architecture and was able to get a validation accuracy of 87%. Rather than changing the architecture played around with processing the data. Data processing I did are converting to grey scale, histogram equalization using opencv. This can be seen in the cell [4] of the notebook.

   As can be seen from the below images, the data processing helps bring out information from the pictures that are otherwise difficult to pick up. Especially darker images.



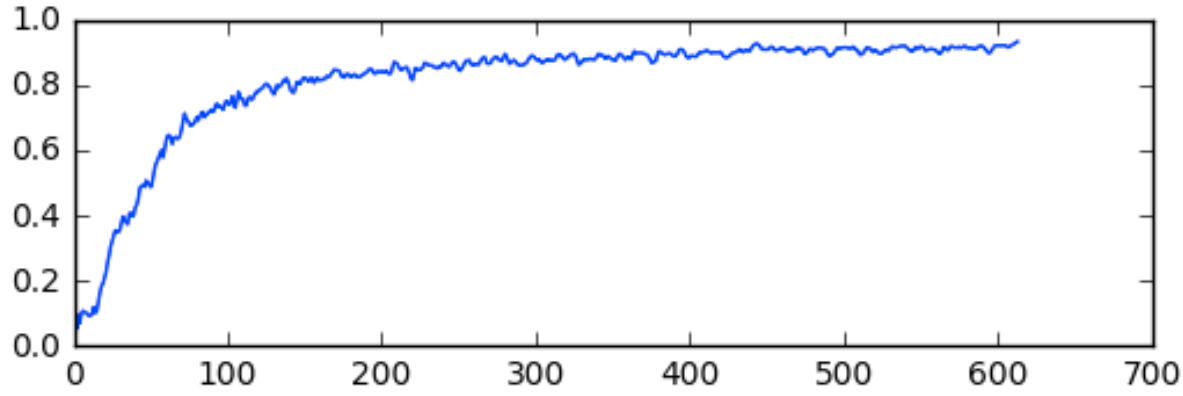
Because I changed the input image dimensions from 32x32x3, I had to make the input stage of Lenet take a 32x32x1 vector as a input. I mainly played around with learning rate, BATCH SIZE,sigma till I hit upon ones that gave me the validation accuracy.

The below parameters gave me the best validation accuracy

EPOCHS 1000
rate 0.008
mu 0.0
sigma 0.1
batch size 256

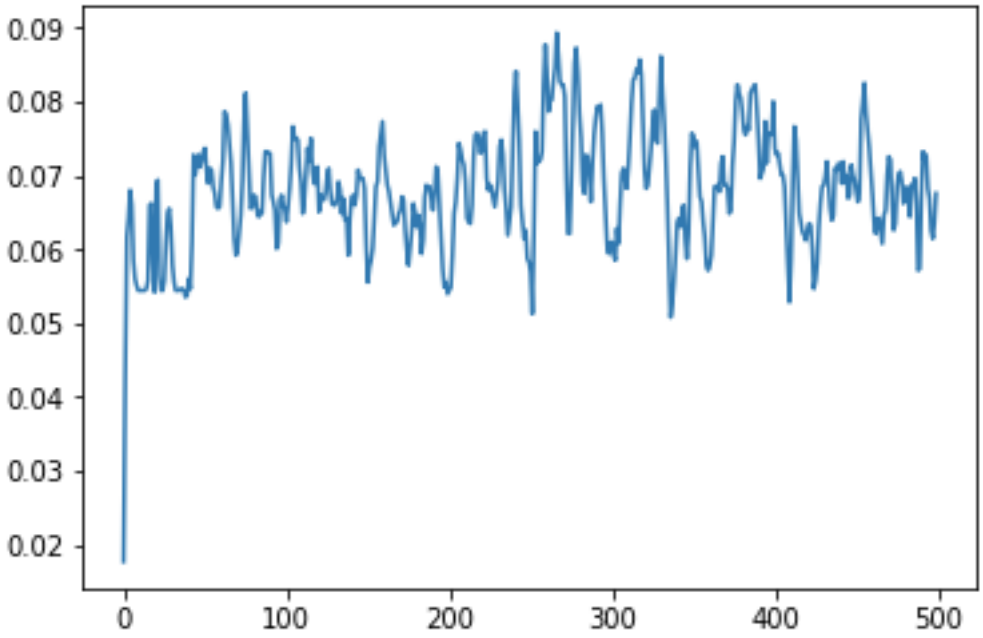
Max validation accuracy achieved **0.932199546485**

In the below image, one can see the accuracy inching towards the target with every EPOCH.



The only change I did to the Lenet architecture was adding a drop out with 30% probability to the fc0 fully connected layer. I found that adding more drop outs did not significantly aid in improvement of validation accuracy.

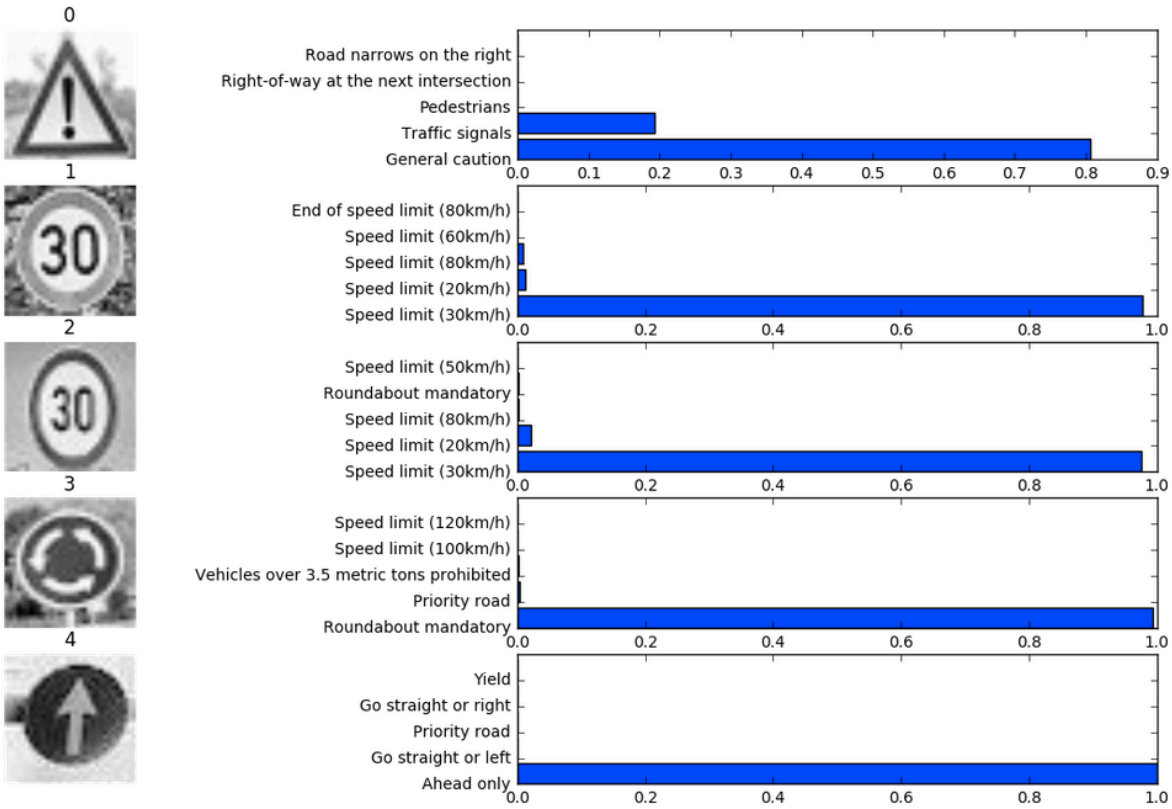
I should also point out that when the network was trained with values that made no sense, like very high values of sigma, the learning was almost zero. The below graph illustrates this



4.0) Use the model to make predictions on new images

I downloaded 5 images off the internet and applied the same pre processing steps that I had used for validation and training sets. Evaluation of these gave a accuracy of 100 %. As can be verified in cell [24]

5.0) Softmax probabilities analysis
The below image illustrates the softmax probabilities of top 5 outputs



The classifier does a good job of getting high probabilities of > 0.8 for the correct class. Interestingly, for the 30 Kmph traffic signs, the softmax probabilities of other speed limit related signs show up in top 5. This signifies that the speed limit related signs are somewhat closer to each other in the feature space

6) Further improvements

The traffic sign classifier was able to obtain a maximum validation accuracy of 94.5 % in my tests. Further improvements have been shown to be possible

a Use a better model? Maybe Lenet is not that great
b Data augmentation. Generating fake data, more data to be tried
c Increase the depth of Lenet (admittedly, it would no longer be Lenet)