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%.

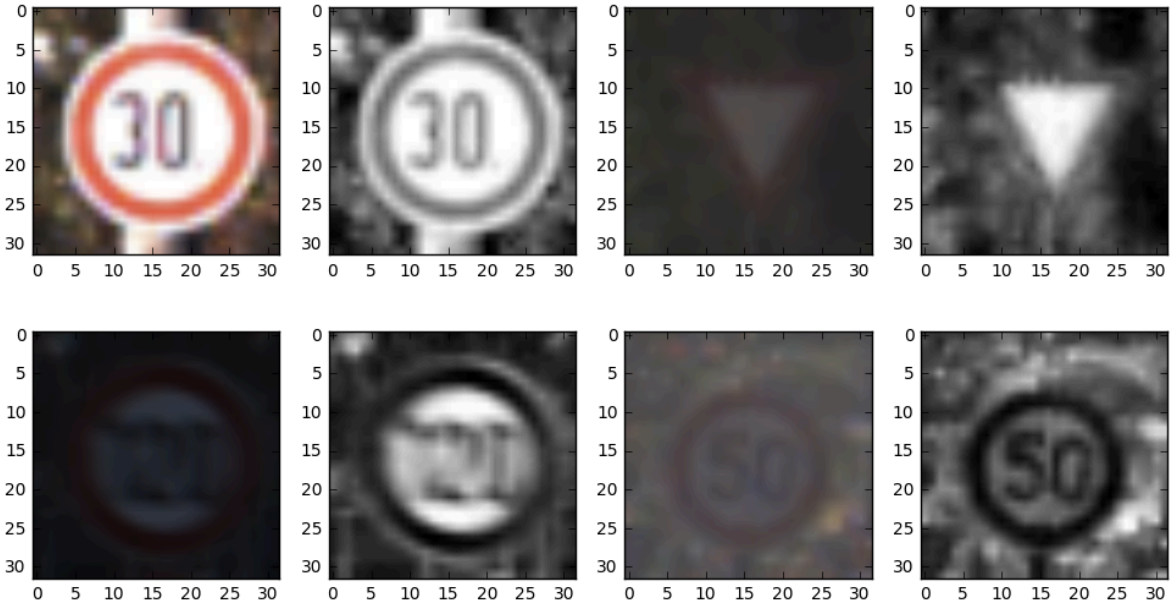
   I added a dropout layer after the fc0 which seemed to improve performance. With that the architecture was

   Layer 1: Convolutional.(5x 5x 1x 6) Input = 32x32x1. Output = 28x28x6.
   Relu activation
   Pooling. Input = 28x28x6. Output = 14x14x6
   Layer 2: Convolutional. (5x5x1x6) Output = 10x10x16
   Relu activation

Pooling. Input =(2x2 pooling and 2x2 kernel size) 10x10x16. Output = 5x5x16
Flatten and Dropout (30% probability) (400 nodes)
Layer 3: Fully Connected. Input = 400. Output = 120
RELU
Layer 5: Fully Connected. Input = 84. Output = 43

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.

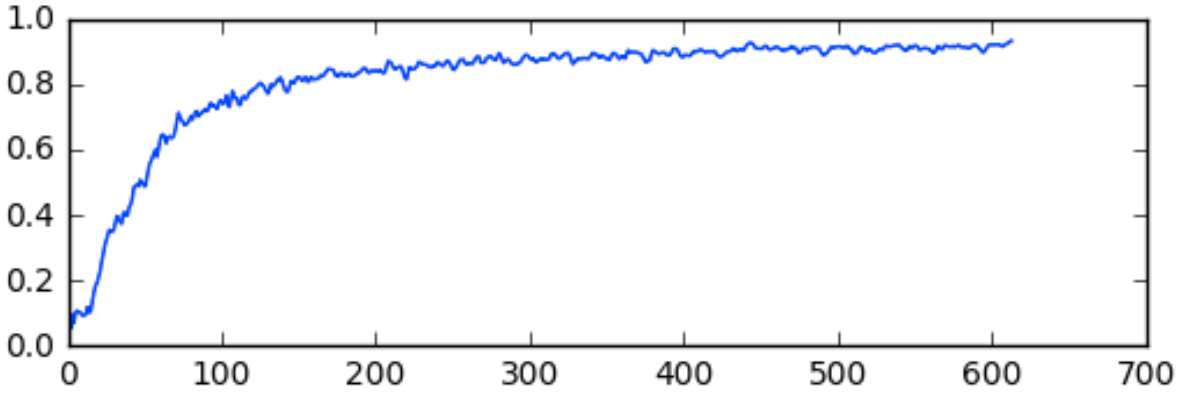


Because I changed the input image dimensions from 32x32x3, I had to make the input stage of Lenet take a 32x32x1 vector as a input. I mainly played around with learning rate, BATCH SIZE,sigma till I hit upon ones that gave me the validation accuracy.

The below parameters gave me the best validation accuracy
EPOCHS 1000
rate 0.008
mu 0.0
sigma 0.1
batch size 256

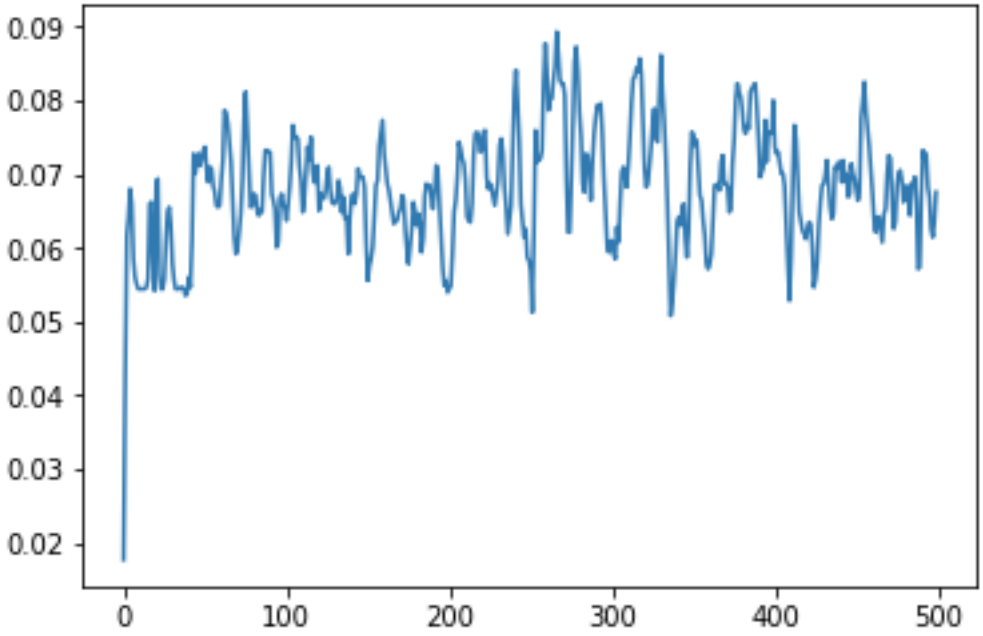
Max validation accuracy achieved **0.9317**
Max testing accuracy achieved is **0.907**

In the below image, one can see the validation 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



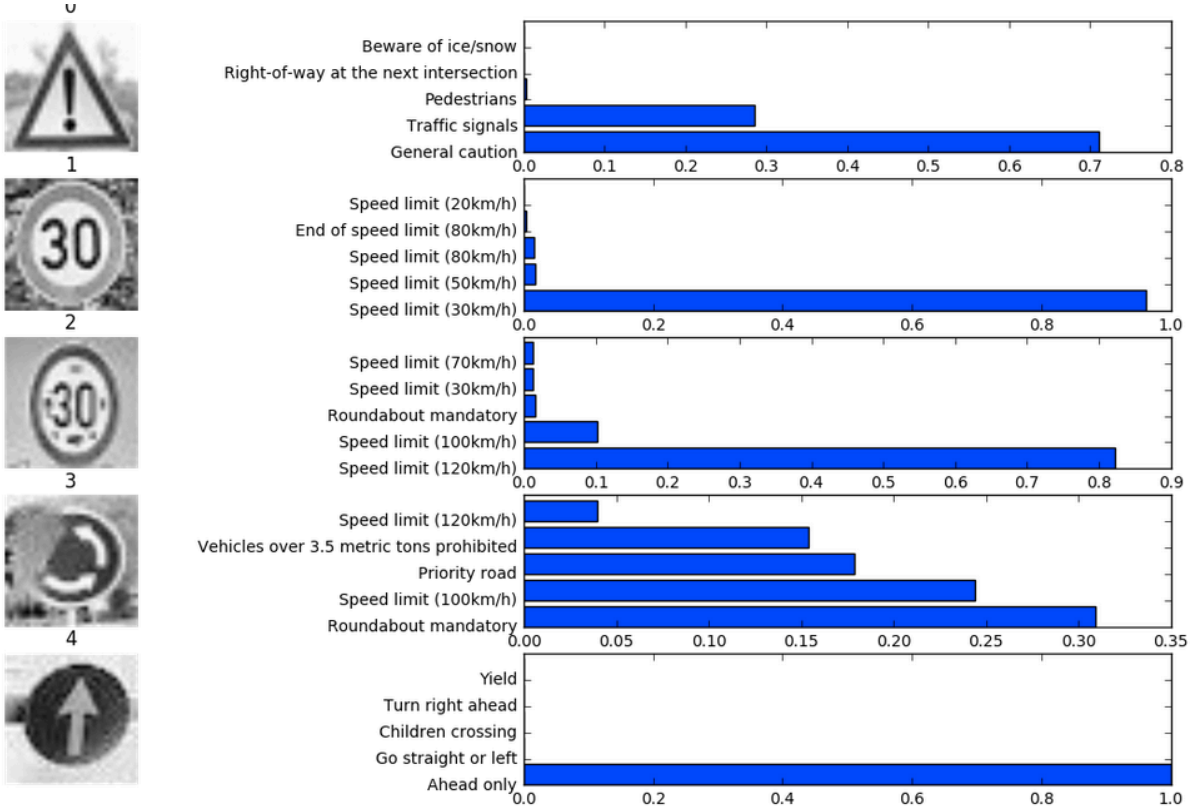
4.0) Use the model to make predictions on new images

I downloaded 5 images off the internet and applied the same pre processing steps that I had used for validation and training sets.

The third image, “30 Kmph speed limit” has some artifacts added to it artificially. So, the classifier should have trouble classifying it.

The 4th image has one of the three round about arrows masked off, so it will be interesting to see what happens.
Over all got a 80% accuracy.

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



The overall accuracy on the 5 random images comes up to 80 %.

The third image, which had some artifacts introduced on it failed classification and from the bar chart , one can see that the classifier thinks it is closer to 120 Kmph, which makes sense. Infact, the module is pretty confident (>80%) that that it’s a 120 Kmph signal.

The roundabout got detected correctly despite having the third arrow being removed. Although, the confidence level is just > 30 %. The model isn’t sure, but, it somehow just managed to classify it correctly

So, other than the edited images, the other three images were classified with a good enough (> 70%) confidence values. So the model is able to generalize well enough and is pretty confident about its predictions. Safe to say that it is fitted well.

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)