Traffic Sign Recognition

The goals / steps of this project are the following:
* Load the data set (see below for links to the project data set)
* Explore, summarize and visualize the data set
* Design, train and test a model architecture
* Use the model to make predictions on new images
* Analyze the softmax probabilities of the new images
* Summarize the results with a written report

Data Set Summary & Exploration

1. The code for this step is contained in the second code cell of the IPython notebook.

I used the python library to calculate summary statistics of the traffic
signs data set:

* The size of training set is 34799
* The size of test set is 12630
* The shape of a traffic sign image is (32, 32, 3)
* The number of unique classes/labels in the data set is 43

2. The code for this step is contained in the third code cell of the IPython notebook.

Here is an exploratory visualization of the data set. I use pyplot to plot random image in the data set. Here is an example:

Design and Test a Model Architecture

1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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.

The code for this step is contained in the 8th code cell of the IPython notebook.

I design the code in the cell with 3 options to test. They are color, grayscale, and Y channel from YUV color space but among the three, I get the best result from the RGB color space.

Then I normalize the data to -0.5 and 0.5 because I get over fitting when I use the actual value of the RGB

2. Since I am provided with the validation (valid.p), I try to use it to see if I get a good validation. Therefore, my final training set had 34799 number of images. My validation set and test set had 4410 and 12630 number of images.

3. The code for my final model is located in the 11th cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 3x3	1x1 stride, VALID padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	1x1 stride, VALID padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Faltten	Output 400
Fully connected	Output 120
RELU	
Dropout	Probability 0.5
Fully connected	Output 84
RELU	
Dropout	Probability 0.5
Softmax	Output 43

4. The code for training the model is located in the 15th cell of the ipython notebook.

To train the model, I used:
Optimizer: AdamOptimizer
Batch size: 512
Number of epochs: 30

Learning rate: 0.001 switch to 0.0005 if the current validation loss greater than the previous one
mu = 0
sigma = 0.1

5. The code for calculating the accuracy of the model is located in the 14th cell of the Ipython notebook.

My final model results were:
* training set accuracy of 97%
* validation set accuracy of 93%
* test set accuracy of 92%

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 five German traffic signs that I found on the web:

2. The code for making predictions on my final model is located in the 19th cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
30km/h Stop sign Turn right ahead 20km/h Wild animals crossing	30km/h Stop sign Turn right ahead Priority road Wild animals crossing

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares favorably to the accuracy on the test set of 43

3. The code for making predictions on my final model is located in the 11th cell of the Ipython notebook.

For the first image, the model is relatively sure that this is a speed limit 30km/h (probability of 0.19), and the image does contain a 30km/h. The top five soft max probabilities were

Probability	Prediction
0.19 0.12 0.09 0.07 0.07	30km/h 20km/h 50km/h 70km/h 80km/h

For the second image, the model is relatively sure that this is a stop sign (probability of 0.06), and the image does contain a stop sign. The top five soft max probabilities were

Probability	Prediction
0.06	Stop sign

0.02	Dangerous curve to the left
0.01	30km/h
0.01	60km/h
0.01	50km/h

For the third image, the model is relatively sure that this is a right turn ahead (probability of 0.16), and the image does contain a right turn ahead. The top five soft max probabilities were

Probability	Prediction
0.16	Turn right ahead
0.07	Keep left
0.05	Roundabout mandatory
0.05	Ahead only
0.03	Yield

For the forth image, the model is relatively sure that this is a priority road (probability of 0.03), but the image does not contain a priority road. The top five soft max probabilities were

Probability	Prediction
0.03	Priority road
0.02	No entry
0.01	Traffic signals
0.006	Stop
0.005	Wild animals crossing

For the forth image, the model is relatively sure that this is a priority road (probability of 0.03), but the image does not contain a priority road. The top five soft max probabilities were

Probability	Prediction
0.03	Priority road
0.02	No entry
0.01	Traffic signals
0.006	Stop
0.005	Wild animals crossing

Summary

It is a good starting point to use LeNet training model architecture for traffic sign Recognition but there are some more places to tune to get a better result for example, I have not augmented the training data set. I am sure, I can get higher accuracy than 93%, if I spend more time to augmented the training data set and fine tune the hyper-parameters. It surprises me that the model predict the 20km/h zone wrong. I am not sure if the sign I got it from the web is the German sign for 20km/h.