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 github repo:
https://github.com/titanum456/Udacity_SelfDriving_TrafficSignClassifier

Data Set Summary & Exploration

1. Provide a basic summary of the data set and identify where in your code the summary was done. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

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

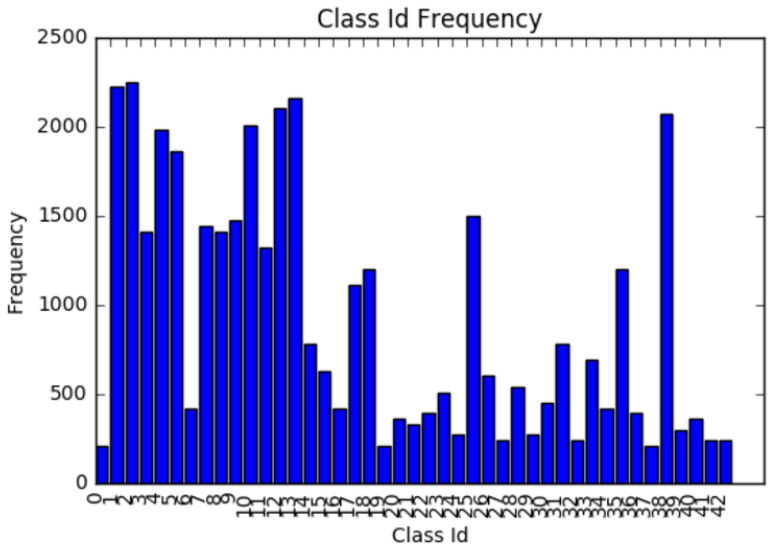
- The size of training set is 39209
- The size of test set is 12630
- The shape of a traffic sign image is (32,32)
- The number of unique classes/labels in the data set is 43.

2. Include an exploratory visualization of the dataset and identify where the code is in your code file.

The code for this step is contained in the 3rd code cell of the IPython notebook.

A good practice of data exploration would be to identify the frequencies of each class in the training set.

Here is an histogram chart of the frequencies of the Class Id. The frequencies are not evenly scattered across each Class Id hence there are 8 Class Id with more than 1500 samples and the majority of the Class Id has less than 1500 samples. Due to this observation, the training set is slightly bias towards the sample with more frequency thus this discrepancies might affect the final accuracy levels.



As well, below is a plot of the sample image in the dataset. Most of the images are blurry and in low brightness. Hence, this too affects the final accuracy of the training model.

Design and Test a Model Architecture

1. Describe how, and identify where in your code, 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.

The code for this step is contained in the 5th code cell of the IPython notebook.

I considered several preprocessing techniques after having a look at the sample image in the training dataset in the previous question.

Initially, I wanted to test out gray scaling techniques but after reading up on several Udacity forum threads, it was advised to not implement this technique as some of the images in the dataset have more meaning when there is color on them like traffic sign of red sign or green sign. So, I dropped that idea.

Finally, I decided to normalize the training datasets. By normalizing, all the images will have equal mean and variances thus this will make the gradient descent optimizer process more easier.

For visualization purpose, here is a sample image after normalization:

2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)

The code for splitting the data into training and validation sets is contained in the 8th code cell of the IPython notebook.

To cross validate my model, I randomly split the training data into a training set and validation set. I did this by using sklearn's train_test_split module. I proportioned 20% of the training data as validation set.

My final training set had 31367 number of images. My validation set and test set had 7842 and 12630 number of images.

3. Describe, and identify where in your code, 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.

As a start, I implemented LeNet architecture with few hyperparameter tuning and added regularization at the fully connected layers except the final layer.

My final modified model consisted of the following layers:

Input (32x32x3) -> Convolutional (28x28x6) -> ReLu -> MaxPool (stride of 2) -> Convolutional (10x10x16) -> ReLu -> MaxPool (stride of 2) -> Flatten (400) -> Fully Connected (120 units) -> ReLu -> Regularization (Dropout)-> Fully Connected (84 units) -> ReLu -> Regularization (Dropout)-> Fully Connected (43 units, output)

The code for my final model is located in the 10^{th} code cell of the ipython notebook.

4. Describe how, and identify where in your code, 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.

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

To train the model, I used the below hyperparameters:

1) Optimizer: AdamOptimizer
2) Batch Size : 100
3) Epochs : 15
4) Learning Rate : 0.001

5. Describe the approach taken for finding a solution. Include in the discussion the results on the training, validation and test sets and where in the code these were calculated. Your approach may have been an iterative process, in which case, outline the steps you took to get to the final solution and why you chose those steps. Perhaps your solution involved an already well known implementation or architecture. In this case, discuss why you think the architecture is suitable for the current problem.

The code for calculating the accuracy of the model is located in the 16th cell of the Ipython notebook.

My final model results were:

- validation set accuracy of 96.9%
- test set accuracy of 91.1%

To reach this final model, I have done several iterative approach in starting with the data preprocessing where I experimented with histogram equalization and then normalization. But, after testing these two methods with the final architecture, the preprocessing step which worked the best was just simple normalization.

As for the network architecture, I started with the LeNet architecture and tweaked the parameters so that is able to accept input images corresponding to the traffic signs like input images are now in color instead of gray input image and as well, 43 output classes.

I used the existing architecture except that I added dropouts at each of the first two fully connected layers and excluding the final output layers. This is avoid overfitting of the data which may causes decrease in validation set accuracy. Then, I reduced the number of epochs as I noticed that when I tested with 40 epochs, the validation accuracy was unstable and the % accuracy was fluctuating badly. Next, I tuned the batch size hyperparameters. When I reduced the batch size from default 128 to 100, the validation and test accuracy increased.

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:

Since this images are German Traffic Signs, there is an instance of these same images in the training dataset. Thus, the model should be able to have good accuracy in predicting the traffic sings for these new images.

However, if the first image is a little blur and looks like orange edges instead of red this this might effect the final accuracy as for preprocessing steps, I have only applied normalization and no other preprocessing techniques.

2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).

The code for making predictions on my final model is located in the 20th code cell of the Ipython notebook.

Here are the results of the prediction:

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. To get the prediction, I used the signnnames.csv file for prediction labels.

Image	Prediction
Yield	Yield
Stop	Stop
General Caution	Priority Road
Speed limit 70km/h	Speed limit 70km/h
No entry	No entry

Comparing the new images accuracy of 80% to the test set accuracy of 91.1%, one possible explanation for the incorrect prediction for one of the image is that the model was trained on less samples of that kind of image in the training dataset.

3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction and identify where in your code softmax probabilities were outputted. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" part of the rubric, visualizations can also be provided such as bar charts)

The code for making predictions on my final model is located in the 22nd code cell of the Ipython notebook.

From the output of the top 5 softmax probabilies below:

```
TopKV2(values=array([[  1.00000000e+00,   7.34688850e-17,   1.45375272e-17,
          9.75838136e-21,   8.96026112e-21],
       [  1.00000000e+00,   1.97818541e-12,   1.12235221e-12,
          1.09503098e-14,   4.41473233e-15],
       [  9.94189620e-01,   3.51918093e-03,   2.27547623e-03,
          1.55400412e-05,   1.05458390e-07],
       [  9.99996066e-01,   3.20267350e-06,   4.92994843e-07,
          1.93227720e-07,   3.49134675e-12],
       [  1.00000000e+00,   1.41653886e-16,   3.77888157e-18,
          4.67415864e-23,   2.20204495e-23]], dtype=float32), indices=array([[1
3,  9, 35, 39,  3],
       [14, 25, 17, 32, 12],
       [12, 17, 14, 32,  9],
       [ 4,  0,  1,  5, 33],
       [17, 14, 12,  1, 15]]))
```

The model were at least 99% confident for its 1st prediction labels. For the image, which was inaccurately predicted, the correct sign was not in the top 5 softmax prediction. Again, the reasons for this situation was explained in the previous question.