Traffic Sign Recognition

Build a Traffic Sign Recognition Project

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

Rubric Points
Here I will consider the [rubric points](https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

The code is provided as HTML as required.

Data Set Summary & Exploration

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

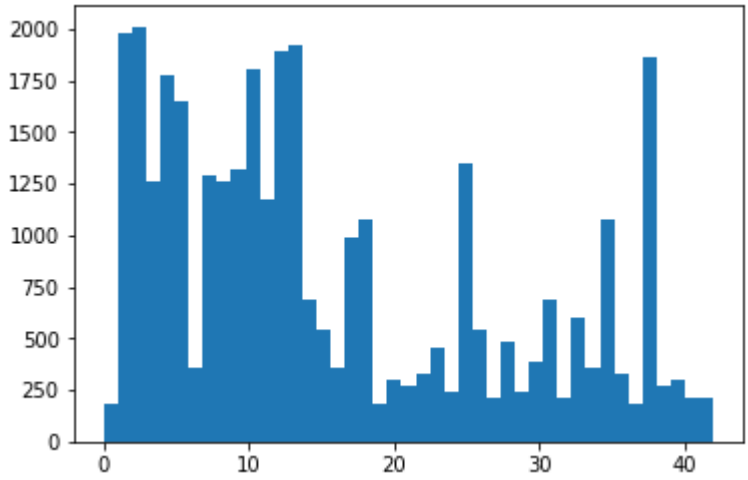
I used the python and numpy library to calculate summary statistics of the traffic signs data set:

```
Number of training examples = 34799
Number of testing examples = 12630
Image data shape = (32, 32, 3)
Number of classes = 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, 4th and 5th code cells of the IPython notebook.

The exploratory visualization of the data set is both unique traffic signs and histrogram. Here is the bar chart showing how the data is distributed based on the 43 unique traffic signs and number of the training examples.



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 6th code cell of the IPython notebook.

I used sklearn shuffle method to shuffle my training features and labels as it's done in LeNet-Lab-Solution-Traffic-Signs due to the good results.

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 6th 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 train_test_split method with 20% for test set and random state 42.

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.

The code for my final model is located in the 7th cell of the ipython notebook.

The net layers and architecture is as follow:

- Layer 1: Convolutional. Input = 32x32x3. Output = 28x28x6.
- Using relu for activation layer
- Pooling. Input = 28x28x6. Output = 14x14x6.
- Layer 2: Convolutional. Output = 10x10x16.
- Using relu for activation layer
- Pooling. Input = 10x10x16. Output = 5x5x16.
- Flatten. Input = 5x5x16. Output = 400.
- Layer 3: Fully Connected. Input = 400. Output = 120.
- Using relu for activation layer
- Layer 4: Fully Connected. Input = 120. Output = 84.
- Using relu for activation layer
- Layer 5: Fully Connected. Input = 84. Output = 43.

My final model is identical to the LeNet example but the output of last 5th layer is set to the number of classes 43.

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 hyper parameters such as learning rate.

The code for training the model is located from 8th- 11th the cell of the ipython notebook.

To train the model, I used 20 Epochs and 128 BATCH size. I tried with different learning rates as 0.001 gave good results and the big number of Epoch is found based on multiple tests with epochs between 10 and 30.

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 12th and 13th cell of the Ipython notebook.

- What was the first architecture that was tried and why was it chosen?
- What were some problems with the initial architecture?

  My first tries was using validation set directly from valid.p and by mistake the one_hot oreder was create on 10. This leads to results under 10%.

- How was the architecture adjusted and why was it adjusted?

  After fixing the order one_hot to the number of unique signs = 43 I had better results but still too poor with validation accuracy around 0.85.

- Which parameters were tuned? How were they adjusted and why?

  I used sklearn train_test_split to adjust train and validation data sets and this increase dramatically my accuracy. After tests with different epochs and learning_rate I found current combination as optimal. The learning rate less than 0.001 gave too slow increase of the accuracy over training process. Higher values have less train accuracy.

- What are some of the important design choices and why were they chosen?

  Accurate one_hot_y matrix with accurate learning rate and validation / train sets are most important for my final results below:

My final model results were:
* training set accuracy of 0.988
* validation set accuracy of 0.962
* test set accuracy of 0.873

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:




    - I found and fill the number corresponding from the signnames.cvs file.
    - I choose higher resolution images to give better quality and chance to be better classified. After that they was reshape to the expected input shape.

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 from 14th to 17th cells of the Ipython notebook.

    - I resize all images to the expected 32x32x3 , after which I fill new pictures labels and features arrays and use them to evaluate the accuracy which is 0.20

    - The accuracy of newly tested images is only 40% where before on the test set was 87% thus it seems the model is overfitting.

3. The top five softmax probabilities of the predications on the captured images are outputted in the 23th cell.

    The model is almost or 100% sure for 4 from 5 top probabilities. Nevertheless for the fifth probability the model is around 55/45 percent between 30 and 25th sign.
    I would say that the model still need to be improved.