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.

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! I submitted the code and writeup in the zip file.

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 second code cell of the IPython notebook.

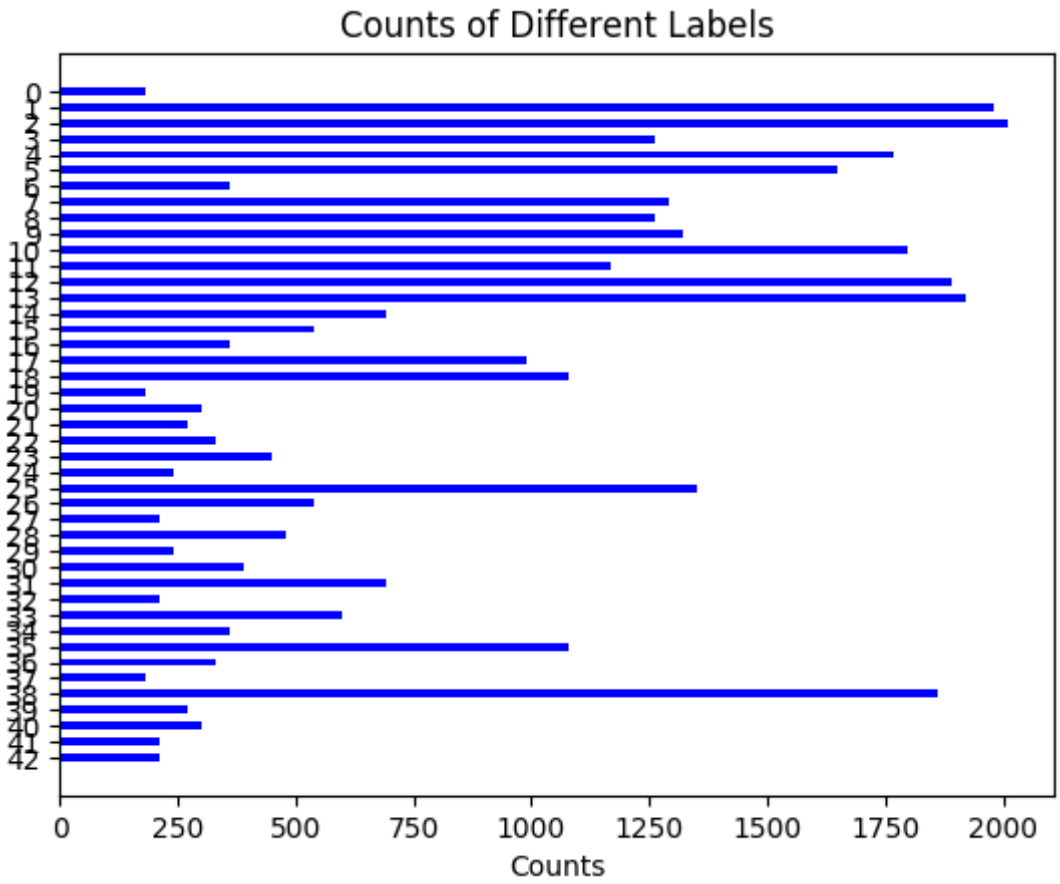
I used the numpy library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799
- The size of test set is 12630
- The shape of a traffic sign image is 32x32x3
- 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 third code cell of the IPython notebook.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed among the different labels.



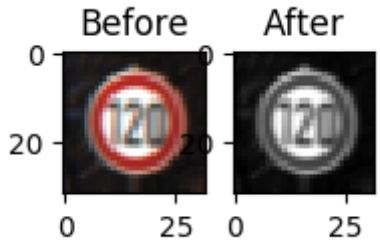
Design and Test a Model Architecture

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 fourth to seventh code cells of the IPython notebook.

As a first step, I decided to convert the images to grayscale because it's easier to detect edges in grayscale.

Here is an example of a traffic sign image before and after grayscaling.



As a last step, I normalized the image data because normalized data can avoid numerical instability and gives the optimizer good condition to work with and search for the solution faster.

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 difference between the original data set and the augmented data set is the following ...

I added the grayscale values as the fourth channel to the original data. So the shape of the input image becomes 32x32x4.

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 eighth cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x4 RGB+grayscale image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	non-linear transformation
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16.
RELU	non-linear transformation
Max pooling	2x2 stride, outputs 5x5x16
RELU	non-linear transformation
Fully connected	400x120
RELU	non-linear transformation
Fully connected	120x84
RELU	non-linear transformation
Fully connected	84x43
Softmax	calculate the probability
Cross Entropy	calculate the distance to label

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 ninth, tenth and eleventh cells of the ipython notebook.

To train the model, I used an AdamOptimizer, an advanced version of gradient descent algorithm, which utilizes momentum mechanism. I used set the learning rate to 0.001, batch size to 128 and number of epochs to 20.

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 ninth cell of the Ipython notebook.

My final model results were:

- training set accuracy of 99.4%
- validation set accuracy of 92.0%
- test set accuracy of 89.9%

I chose the LeNet 5 architecture I learned in the class because it's a proven CNN to classify images. It turns out to perform reasonably well on this project, given the above accuracy it achieved.

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:

The 1st, 2nd and 4th images might be difficult to classify because they have watermark on it. The 4th image (30km/h speed limit sign) might be easily confused with a 50km/h speed limit sign because 3 and 5 usually look quite similar.

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 tenth cell of the Ipython notebook.

Here are the results of the prediction:

Image	Prediction
Pedestrian	Pedestrian
Priority Road	Priority Road
Keep Right	Keep Right
30 km/h	50 km/h
Stop Sign	Stop sign
Road Work	Road Work

The model was able to correctly guess 4 of the 6 traffic signs, which gives an accuracy of 83.3%. It performed worse than on the test data set (89.9% accuracy), which shows the model may be overfitting on the test data. The overfitting problem can be alleviated by augmenting the training data and using drop layer in the network.

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 11th cell of the Ipython notebook.

For the first image, the model is quite sure that this is a pedestrian sign (probability of 0.99), and the image does contain a pedestrian sign. The top five soft max probabilities were

Probability	Prediction
.99	Pedestrian sign
.01	General Caution
.00	Road narrows on the right
.00	Right-of-way at the next intersection
.00	Traffic signals

For the second image, the model is very sure that this is a priority road sign (probability of 1.0), and the image does contain a priority road sign. The top five soft max probabilities were

Probability	Prediction
1.0	Priority road
.00	Right-of-way at the next intersection
.00	End of no passing by vehicles over 3.5 metric tons
.00	End of no passing
.00	No passing for vehicles over 3.5 metric tons

For the third image, the model is very sure that this is a keep right sign (probability of 1.0), and the image does contain a keep right sign. The top five soft max probabilities were

Probability	Prediction
1.0	Keep right
.00	Dangerous curve to the right
.00	Go straight or right
.00	Children crossing
.00	Yield

For the fourth image, the model mistakes it for a 50km speed sign, and gives very low probability that this is a 30km speed sign (probability of 0.02). The top five soft max probabilities were

Probability	Prediction
.98	Speed limit (50km/h)
.02	Speed limit (30km/h)
.00	Wild animals crossing
.00	Bicycles crossing
.00	Speed limit (80km/h)

For the fifth image, the model is very sure that this is a stop sign (probability of 1.0), and the image does contain a keep right sign. The top five soft max probabilities were

Probability	Prediction
1.0	Stop sign
.00	Speed limit (30km/h)
.00	No entry
.00	Yield
.00	Children crossing

For the sixth image, the model is very sure that this is a road work sign (probability of 1.0), and the image does contain a keep right sign. The top five soft max probabilities were

Probability	Prediction
1.0	Road work
.00	Dangerous curve to the right
.00	Road narrows on the right
.00	Bicycles crossing
.00	Traffic signals