Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the 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 individually and describe how I addressed each point in my implementation.

---

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! HTML file is attached as a project submission.

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 first and second code cell of the IPython notebook.

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

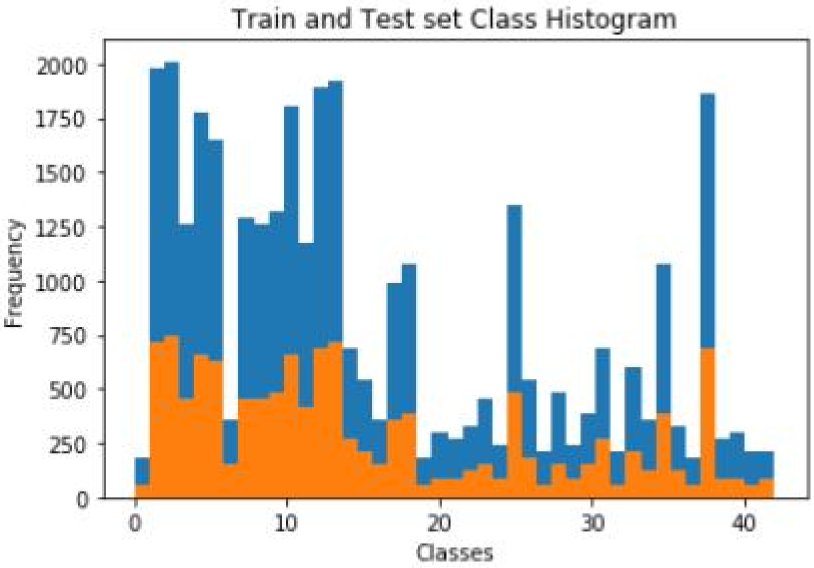
- The size of training set is 34799 samples.
- The size of validation set is 4410 samples.
- The size of test set is 12630 samples.
- The shape of a traffic sign image is (32, 32, 3).

- 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. Here I displayed a histogram of training and test data set by classes.



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 fourth code cell of the IPython notebook.

I decided to shuffle the images and keep them in color (not to change them to grayscale) because color also matters in determining the sign boards.

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)

Data was split into training and validation sets as below:

- The size of training set is 34799 samples.
- The size of validation set is 4410 samples.

The size of test set is 12630 samples.

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 fifth cell of the ipython notebook.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, VALID padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6

Layer	Description
Convolution 5x5	1x1 stride, VALID padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	Input = 5x5x16. Output = 400
Fully connected	Input = 400. Output = 120
RELU	
Fully connected	Input = 120. Output = 84
RELU	
Fully connected	Input = 84. Output = 43
Logits	

Layer	Description
Softmax	
AdamOptimizer	

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 Sixth cell of the ipython notebook.

To train the model, I used below numbers and hyper-parameters:

- EPOCHS = 10
- BATCH_SIZE = 256
- Learning_rate = 0.001
- mu = 0
- sigma = 0.1
- Softmax: To calculate the probabilities and averaged the cross entropy from all the images.
- AdamOptimizer: Used it to minimize the loss function.

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 seventh cell of the Ipython notebook.

My final model results were:

- Validation set accuracy of 85%
- Test set accuracy of 83.8%

I used iterative approach to finalize on the Model:

- I selected first architecture as LeNET with 7*7 filter with 1 layer of Conv & Pooling. I chose LeNet as it’s a simple model to recognize the shapes and formats.
- Initial architecture gave me accuracy below 60% and hence was not suitable.
- But then after changing the LeNET Model with 5*5 filter and adding additional Conv, Pooling and FC layers, accuracy went up to 85%.
- In terms of hyper-parameters, below configuration gave me optimal results:
    - EPOCHS = 10
    - BATCH_SIZE = 256
    - Learning_rate = 0.001
    - mu = 0
    - sigma = 0.1

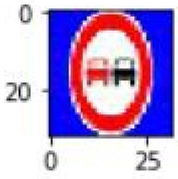
If a well-known architecture was chosen:

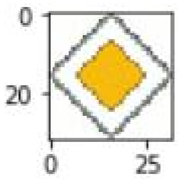
- LeNET-5 architecture was chosen finally.
- This architecture is relevant to the traffic sign application as it’s good to detect the lines, curves and objects made out of them.
- Final model's accuracy of 85% on validation data and 83.8% on test set provided evidence that the model is working well.

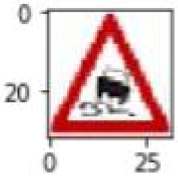
Test a Model on New Images

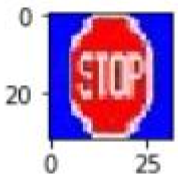
1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

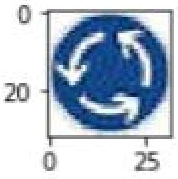
Here are five German traffic signs that I found on the web:











The forth image (Slippery Road) might be difficult to classify because it resembles with other signs like Beware of ice/snow or Children crossing sign.

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 ninth cell of the Ipython notebook.

Below are the results of the prediction:

Image	Prediction
No Passing	No Passing

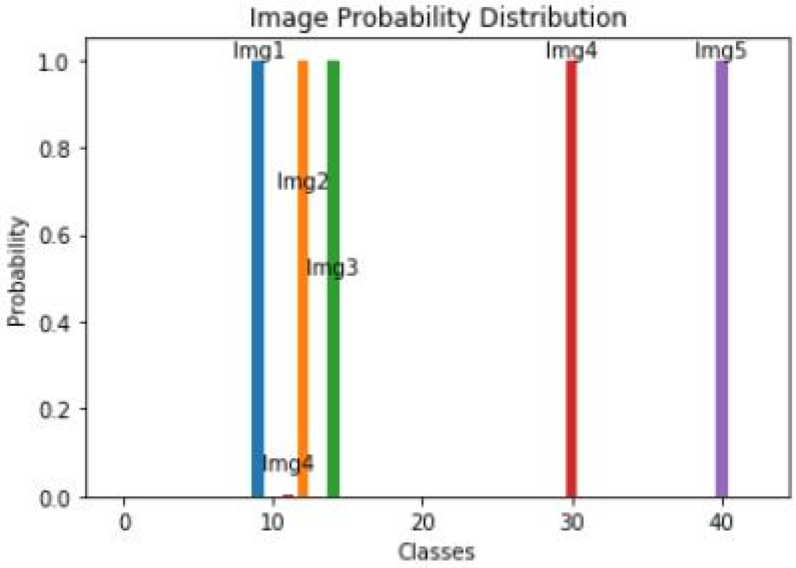
Image	Prediction
Priority Road	Priority Road
Stop	Stop
Slippery Road	Beware of ice/snow
Roundabout Mandatory	Roundabout Mandatory

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares to the accuracy on the test set of 83.8%.

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.

The model is very certain of its predictions. The 4 out of 5 images which are correctly classified have probability close to 1.0. The probabilities plotted against the Classes for all 5 images are shown as below:



The probabilities and predictions are as below for each image:

Image 1: No Passing

Probability	ClassID	Prediction
1.00000000e+00	9	No passing
5.56405223e-17	10	No passing for vehicles over 3.5 metric tons
3.01116924e-24	16	Vehicles over 3.5 metric tons prohibited
1.04058248e-28	20	Dangerous curve to the right
3.96698448e-37	41	End of no passing

Image 2: Priority Road

Probability	ClassID	Prediction
1.00000000e+00	12	Priority road
5.49419927e-16	32	End of all speed and passing limits
4.64003390e-22	10	No passing for vehicles over 3.5 metric tons
4.40065993e-22	41	End of no passing
1.79161533e-24	11	Right-of-way at the next intersection

Image 3: Stop Sign

Probability	ClassID	Prediction
1.00000000e+00	14	Stop
6.49372500e-11	10	No passing for vehicles over 3.5 metric tons
2.20462480e-16	5	Speed limit (80km/h)
2.08331415e-21	8	Speed limit (120km/h)
6.08043888e-22	17	No entry

Image 4: Slippery Road

Probability	ClassID	Prediction
8.99154782e-01	30	Beware of ice/snow
1.00844659e-01	23	Slippery road
5.62119340e-07	28	Children crossing
2.11065405e-08	21	Double curve
1.26723687e-09	11	Right-of-way at the next intersection

Image 5: Roundabout Mandatory

Probability	ClassID	Prediction
9.99931335e-01	40	Roundabout mandatory
6.59195721e-05	11	Right-of-way at the next intersection
2.30511273e-06	36	Go straight or right
2.10882760e-07	39	Keep left
1.93393277e-07	35	Ahead only