Traffic Sign Recognition

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

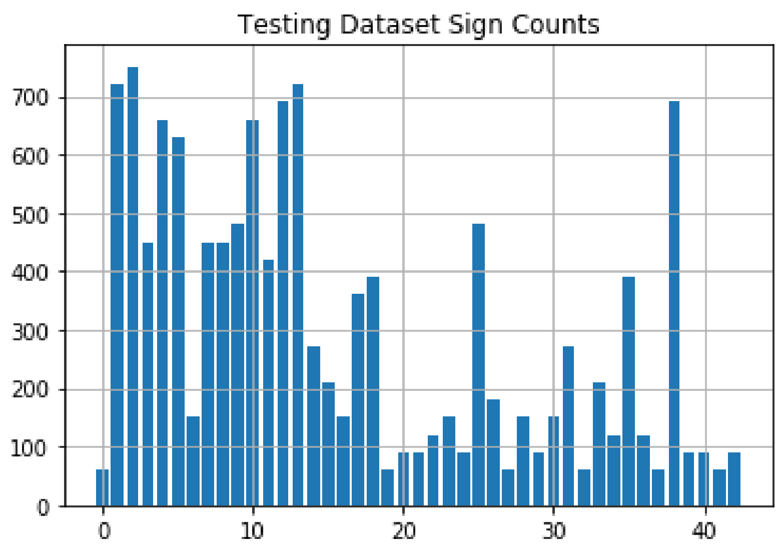
- Load the data set
- Explore, summarize and visualize the data set
- Design, train and test a model architecture
- Use the model to make predictions on new images
- Analyze the softmax probabilities of the new images
- Summarize the results with a written report

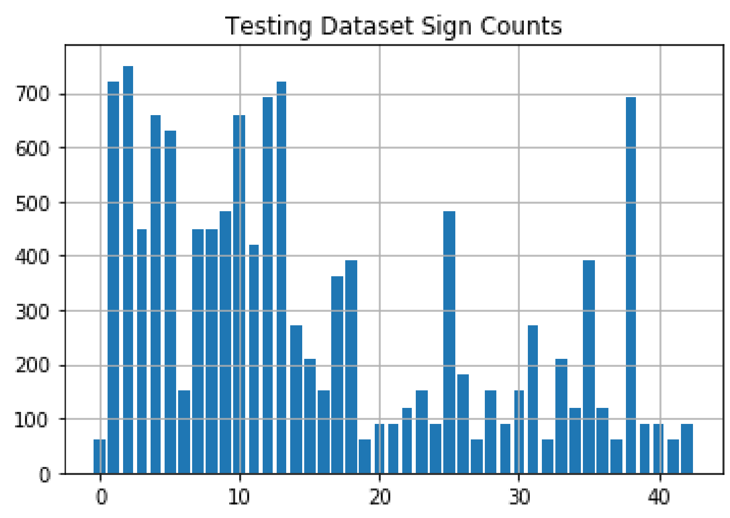
Data Set Summary & Exploration

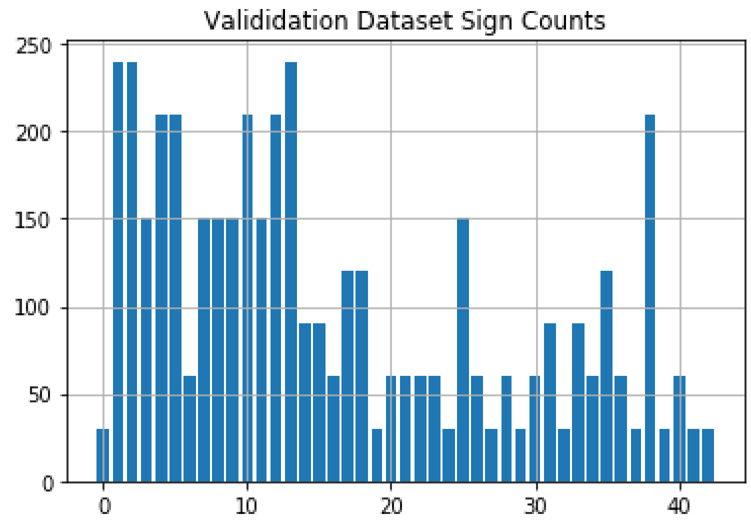
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799 images
- The size of the validation set is 4410 images
- The size of test set is 12630 images
- The shape of a traffic sign image is (32, 32, 3)
- The number of unique classes/labels in the data set is 43

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







Design and Test a Model Architecture

1. Describe how 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. (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, and provide example images of the additional data. Then describe the characteristics of the augmented training set like number of images in the set, number of images for each class, etc.)

As a first step, I decided to convert the images to grayscale so they could be normalized. It is important that the data has mean zero and equal variance.

My final model consisted of the following layers:

Layer	**Description**
Input	32x32x1 RGB image
Convolution 3x3	1x1 stride, same padding, outputs 28x28x6
RELU	Activation layer
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	Output 10x10x16
RELU	Activation layer
Max Pooling	2x2 stride, outputs 5x5x16
Flatten Layer	Output 400
Fully Connected	Output 120
RELU	Activation Layer
Fully Connected	Output 84
RELU	Activation layer
Fully Connected Layer	Output 43

To train the model, I ran 100 EPOCHS with a Batch_size of 156 and included a Dropout of 50%

Utilized Adams Optimizer which worked on a learning ratee of 0.00097

4. Describe the approach taken for finding a solution and getting the validation set accuracy to be at least 0.93. 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.

My final model results were:

- training set accuracy of 94.9%
- validation set accuracy of 96.5%
- test set accuracy of 99.4%

If an iterative approach was chosen:

- What was the first architecture that was tried and why was it chosen?

  Used LeNet as suggested within the lessons

- What were some problems with the initial architecture?

  Preprocessing the data was mandatory to prevent issues with accuracy

- How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to overfitting or underfitting. A high accuracy on the training set but low accuracy on the validation set indicates over fitting; a low accuracy on both sets indicates under fitting.

  Through the introduction of dropout to maintain a high level of accurazcy.

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

  Learning rate was placed at a low level to minimise the loss of the model.

  Batch Size to find a balance between training as much as possible and computational accuracy

  Dropout to prevent overfitting

Test a Model on New Images

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

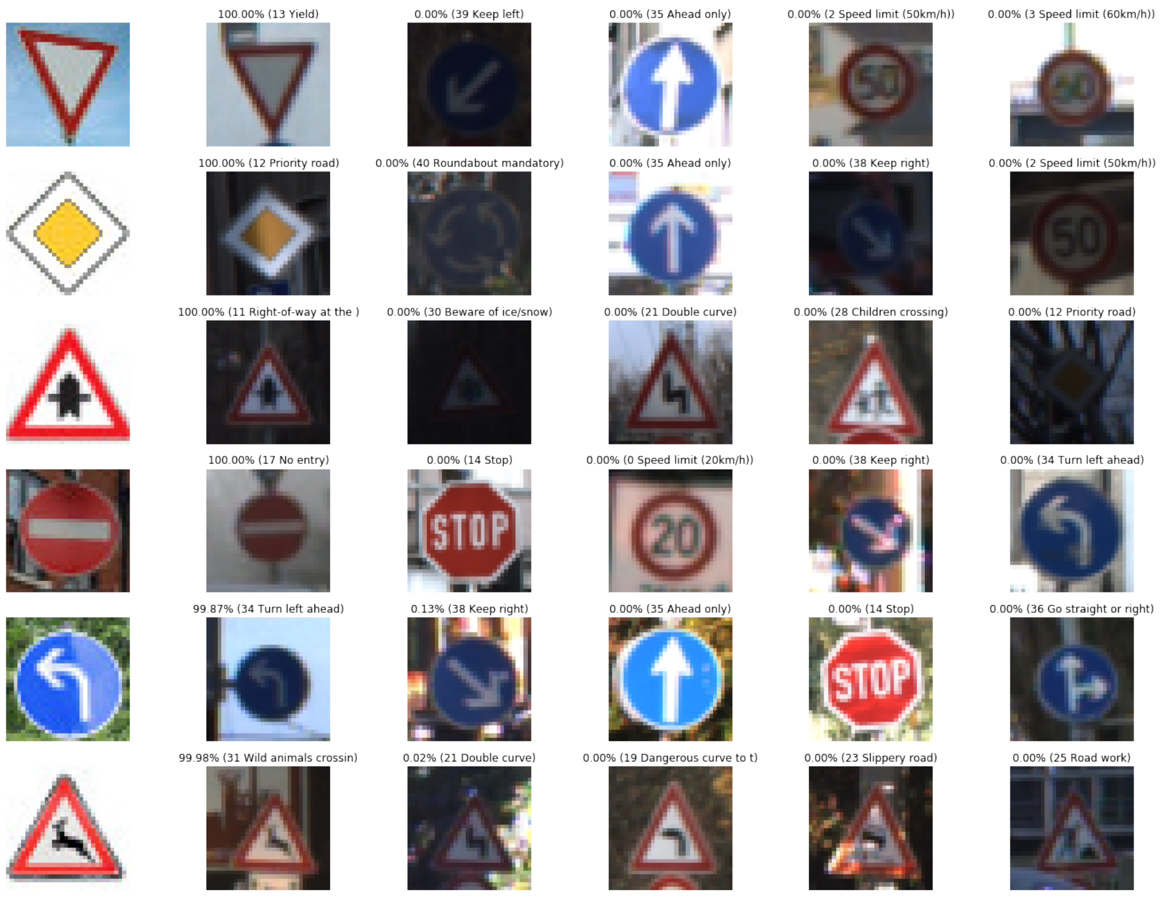
The first image might be difficult to classify because the images were not the size of 32x32x1 so they had to be prepocessed

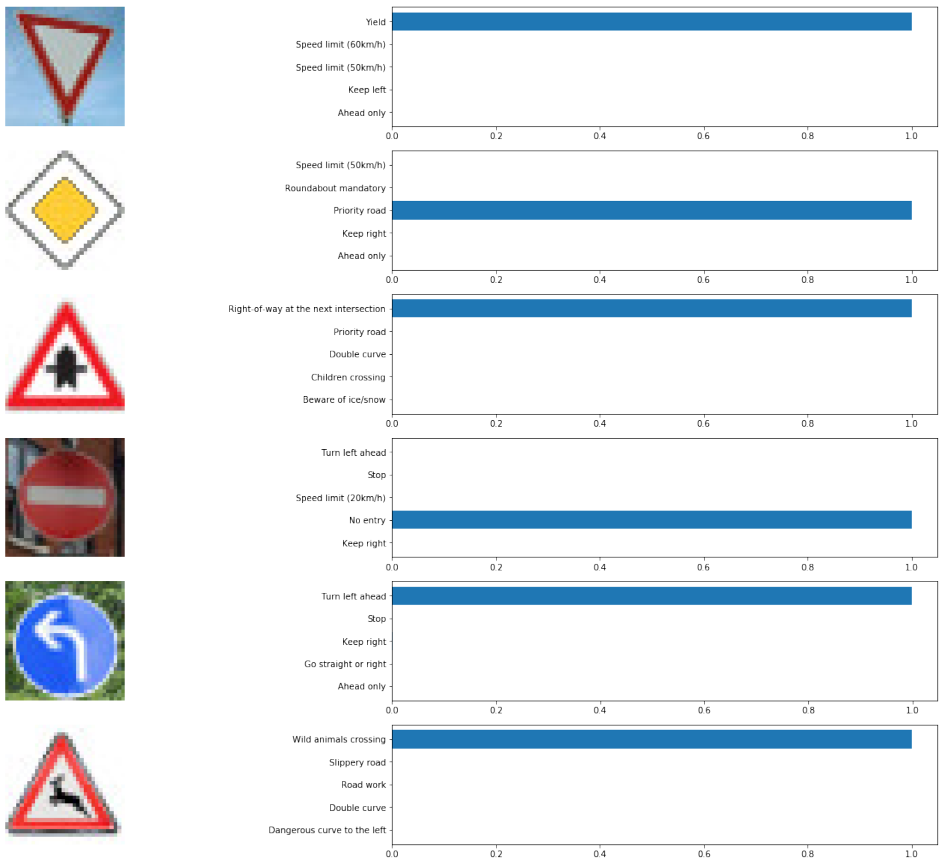
Here are the results of the prediction:

Image	Prediction
Yield	Yield
Priority Road	Priority Road
Right of way	Right of way
No entry	No entry
Turn Left Ahead	Turn Left Ahead
Yield Animal Crossing	Yield Animal Crossing

The model was able to correctly guess 6 of the 6 traffic signs, which gives an accuracy of 100%.

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


100.00% (13 Yield)
0.00% (39 Keep left)
0.00% (35 Ahead only)
0.00% (2 Speed limit (50km/h))
0.00% (3 Speed limit (60km/h))
100.00% (12 Priority road)
0.00% (40 Roundabout mandatory)
0.00% (35 Ahead only)
0.00% (38 Keep right)
0.00% (2 Speed limit (50km/h))
100.00% (11 Right-of-way at the)
0.00% (30 Beware of ice/snow)
0.00% (21 Double curve)
0.00% (28 Children crossing)
0.00% (12 Priority road)
100.00% (17 No entry)
0.00% (14 Stop)
0.00% (0 Speed limit (20km/h))
0.00% (38 Keep right)
0.00% (34 Turn left ahead)
99.87% (34 Turn left ahead)
0.13% (38 Keep right)
0.00% (35 Ahead only)
0.00% (14 Stop)
0.00% (36 Go straight or right)
99.98% (31 Wild animals crossin)
0.02% (21 Double curve)
0.00% (19 Dangerous curve to t)
0.00% (23 Slippery road)
0.00% (25 Road work)


Yield
Speed limit (60km/h)
Speed limit (50km/h)
Keep left
Ahead only
Speed limit (50km/h)
Roundabout mandatory
Priority road
Keep right
Ahead only
Right-of-way at the next intersection
Priority road
Double curve
Children crossing
Beware of ice/snow
Turn left ahead
Stop
Speed limit (20km/h)
No entry
Keep right
Turn left ahead
Stop
Keep right
Go straight or right
Ahead only
Wild animals crossing
Slippery road
Road work
Double curve
Dangerous curve to the left
0.0
0.2
0.4
0.6
0.8
1.0