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

Writeup / README

Data Set Summary & Exploration

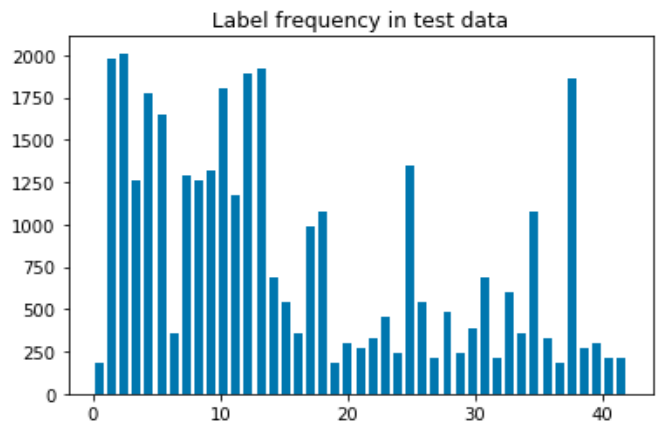
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- 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.

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



#

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 because color information doesn't help us detect important edges and introduces additional complexity to the code

As a last step, I normalized the image data because each image will have similar range (-1, 1) so that the same algorithms can be applied to all of them. Also it will produce better results for image comparison.

2. Describe 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.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Gray image
Convolution 3x3	1x1 stride, same padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 3x3	1x1 stride, same padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	Output 400
Dropout	keep prob 0.9
Fully connected	Output 120
Dropout	keep prob 0.6
RELU	
Fully connected	Output 84
Dropout	keep prob 0.6
RELU	
Fully connected	Output 43

3. Describe how 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.

I found that increasing the number of epoch as well as learning rate and decreasing the number of batch size gave better results.

To train the model, I used: * Batch size : 100 * Number of epoch: 60 * Learning rate : 0.005

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 99% * validation set accuracy of 94.6% * test set accuracy of 92.8%

If an iterative approach was chosen: * What was the first architecture that was tried and why was it chosen? I chose LeNet because it was a good starting point for image detection

- What were some problems with the initial architecture? Initial architecture didn't give good verification 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.

By adding dropouyt layers on Fully connected layers with 0.6 rate and one dropout layer on last convolutional layer with 0.9 rate I found that accuracy improved

- Which parameters were tuned? How were they adjusted and why? batch size, number of epochs, learning rate and dropout rate. At first I set dropout rate at 0.5 but better results were at 0.6 as well as adding a dropout at last convolutional layer.
- What are some of the important design choices and why were they chosen? For example, why might a convolution layer work well with this problem? How might a dropout layer help with creating a successful model? Dropout layer is chosen because then network is less likely to overfit the data.

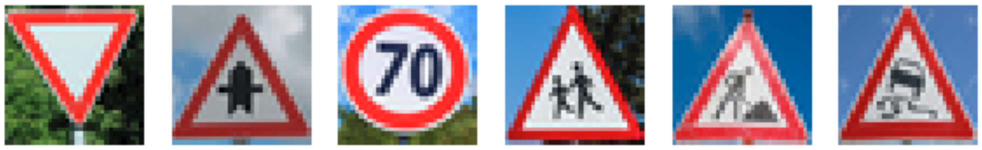
If a well known architecture was chosen: * What architecture was chosen? LeNet

- Why did you believe it would be relevant to the traffic sign application? It worked well on number image recognition
- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well? There is only a small descrepency between accuracy on training, validation and test set results which means that there is small probability for overfit.

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 six German traffic signs that I found on the web:



2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. 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).

Here are the results of the prediction:

Image	Prediction
Yield	Yield
Right-of-way	Right-of-way
70 km/h	70 km/h
Children crossing	Children crossing
Road work	Road work
Slippery road	Slippery road

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

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. 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 model is relatively sure for all images exept for Children crosiing, where prediction is 16%, which is very low. This is suprising since the image doesn't seem to be of lower quality that the rest of the images.

Probability	Prediction
1.0	Yield
1.0	Right-of-way
.88	70 km/h
.16	Children crossing
1.0	Road work
0.77	Slippery road