Traffic Sign Recognition

Writeup

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

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.

Data Set Summary & Exploration

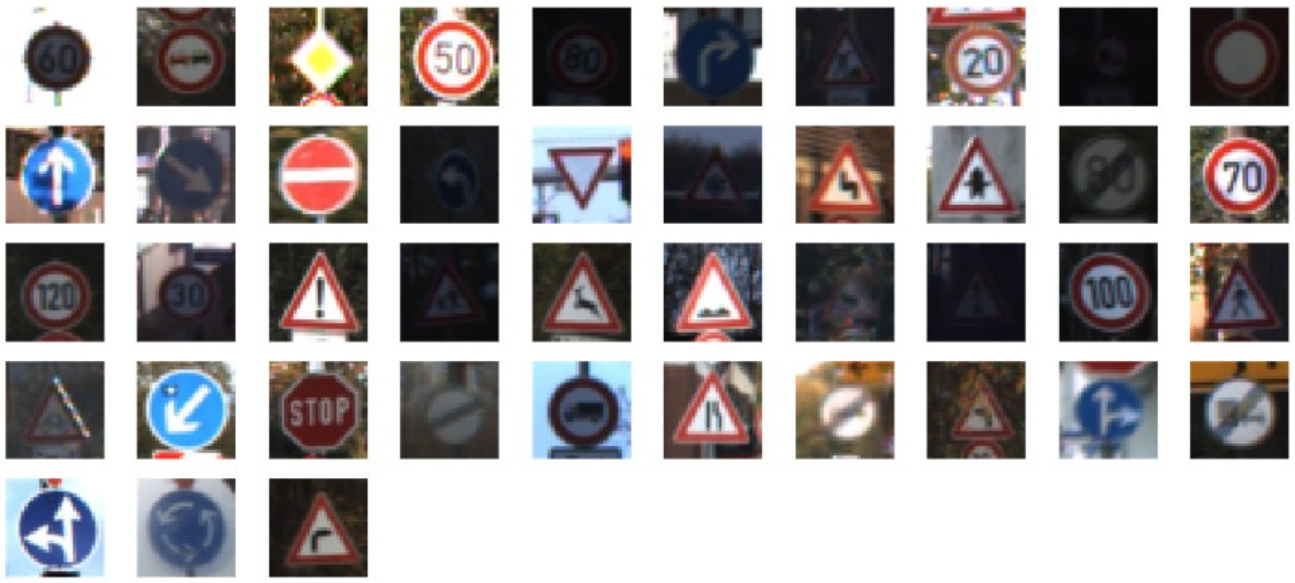
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

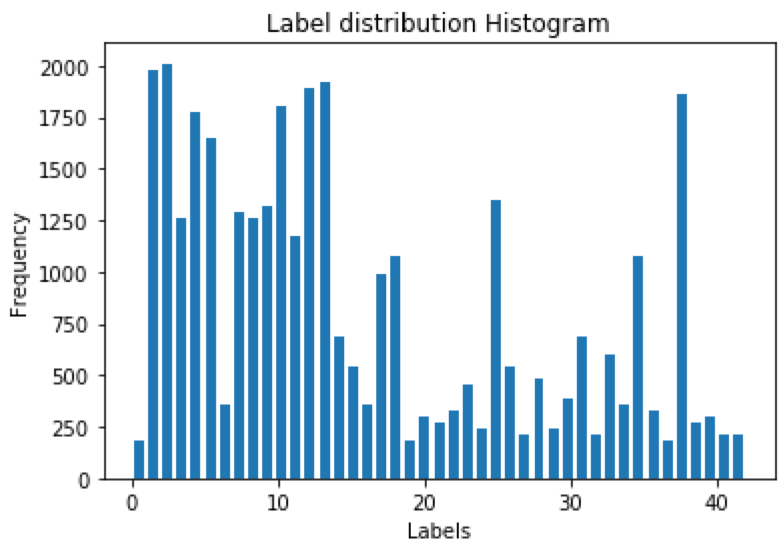
- The size of training set is `34799`.
- The size of the validation set is `12630`.
- The size of test set is `4410`.
- The shape of a traffic sign image is `(32, 32, 3)`.
- The number of unique classes/labels in the data set is 43.

2. Include an exploratory visualization of the dataset.

There are 43 unique traffic signs in the given set. Below is one of each traffic sign classes.



Here is an exploratory visualization of the data set. A histogram is plotted showing frequency of each sample type.



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.)

I did two operations on the image.

Convert to gray scale : This is done as colour is an important factor for classifying the signs and grayscale images as less processor intensive to train.

Normalizing: Normalizing is done to reduce the distribution of data. Wider distribution makes it difficult for the Neural network to learn with a single linear learning rate.

I also attempted to generate augmented data set, but this was so much time consuming on my PC so I gave up.

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.

I started with the LeNet model, and experimented with modifying parameters and layers. I added two dropout layers which seems to give better results that the original LeNet.

My final model consisted of the following layers:

Layer	Description
Convolution	Input = 32x32x1. Output = 28x28x6
	Activation RELU
	Pooling `Input = 28x28x6. Output = 14x14x6.`
Convolution	Output = 10x10x16
	Activation RELU
	Pooling. Input = 10x10x16. Output = 5x5x16.
	Flatten. Input = 5x5x16. Output = 400.
Fully Connected	Input = 400. Output = 120.
	Activation RELU
	Dropout with probability 0.5
Fully Connected	Input = 120. Output = 84.
	Activation RELU
	Dropout with probability 0.5
Fully Connected	Input = 84. Output = 10
	Activation RELU

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 started with the default values from LeNet implementation and modified them for performance. Here are my final values.

Bach size	: 128
Epochs	: 40
Learning rate	: 0.0007
Mean	: 0
Standard deviation	: 0.1
Drop probability	: 0.5

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 0.99
- validation set accuracy of `0.956`
- test set accuracy of `0.935`

If an iterative approach was chosen:

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

The original LeNet architecture was chosen as known working model.

•What were some problems with the initial architecture?

I was not able to use data augmentation, so the initial validation results where less than 0.90. The learning seemed to saturate and go back and forth around the higher 0.80es.

•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.

From a bit of reading and going thorough the classrooms again I decided to use droput layers. Experimented with one dropout layer and then 2. I also adjusted most of the parameters to get a satisfactory result.

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

Learning rate was adjusted after adding dropout layers to avoid overfitting.

Epochs were increased to offset for lower learning rate.

If a well known architecture was chosen:

•What architecture was chosen?

Improved on the original LeNet as it was available to play with.

•Why did you believe it would be relevant to the traffic sign application?

LeNet was already used in previous lab sessions for classifying problems

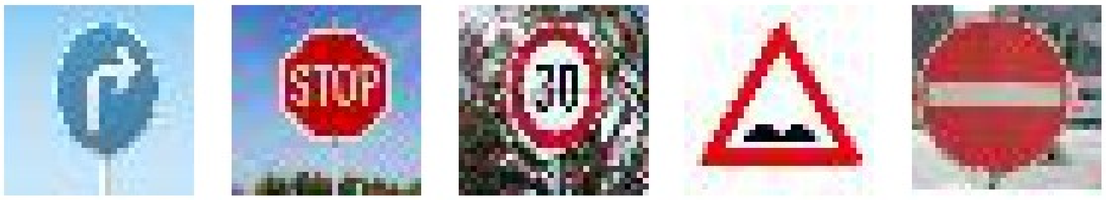
•How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

The test set gave an accuracy of 0.93 which I think is pretty good without data augmentation.

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 first image might be difficult to classify because they have back ground noice.

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).

Please See below

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)

Below is a visualization of the top 5 predictions. Everything other the 30KM speed limit sign is predicted with high probability.

30KM speed limit sign has a tree in the background. This shows that the model is not working well with background images.