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

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! and here is a link to my project code

Data Set Summary & Exploration

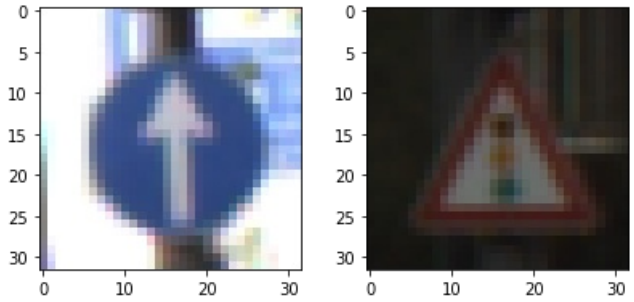
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

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

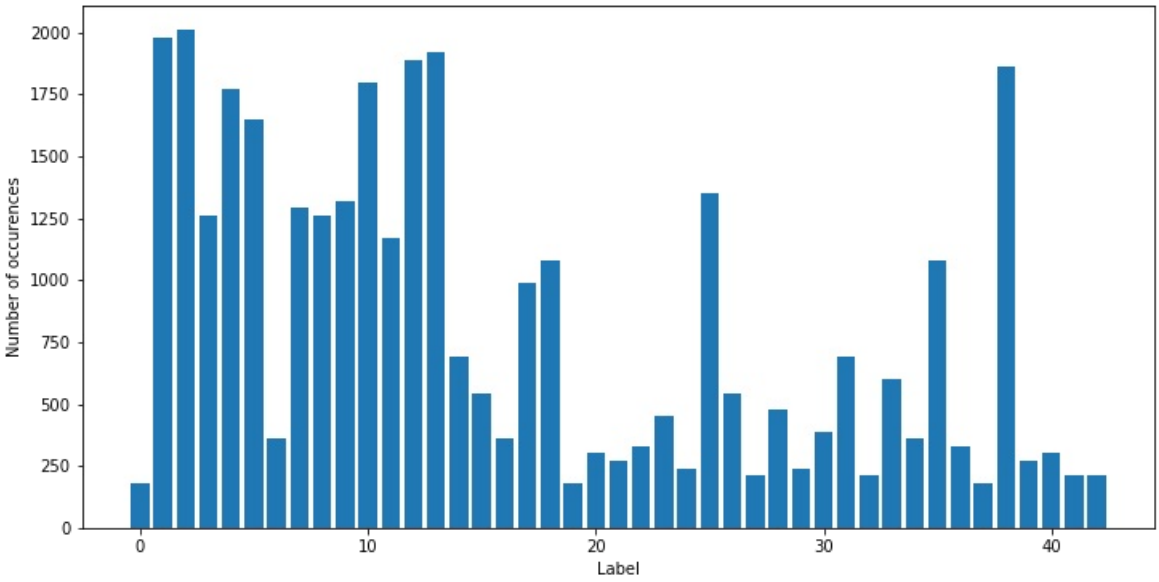
- The size of training set is 34799.
- The size of the validation set is 4410.
- The size of test set is 12630.
- The shape of a traffic sign image is 32x32x3.
- The number of unique classes/labels in the data set is 43.

2. Include an exploratory visualization of the dataset.

The first step in exploring the dataset was to plot some of the images and their corresponding labels in order to get a better idea on the quality of the images.



The next step was to perform an exploratory visualization of the data set. The image below is a bar chart showing the frequancies of the labels that appear in the training set. As it can be seen from the image, the dataset is imbalanced and some examples appear far more frequently than others.

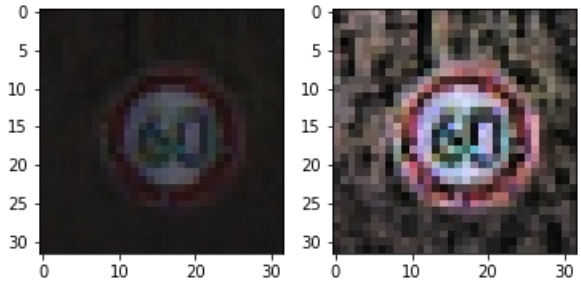


Design and Test a Model Architecture

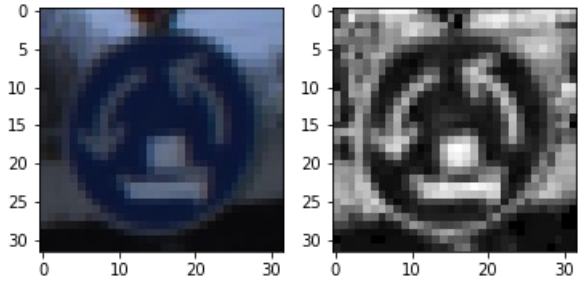
1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described 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 use a technique called Adaptive Histogram Equalization (AHE). The main reason for that decision was that during the initial exploration phase, some images appeared as very dark and this tecnique helped in generating better contrasts. A downside of this approach is that it takes a siginificant amount of time to process all images.

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



The next step was to convert all images to gray scale. Here is an example of an original image and an image that has been through both preprocessing stages.



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 Grayscale image
Convolution 3x3	1x1 stride, same padding, outputs 32x32x32
RELU	
Max pooling	2x2 stride, outputs 16x16x32
Convolution 3x3	1x1 stride, same padding, outputs 16x16x64
RELU	
Max pooling	2x2 stride, outputs 8x8x64
Fully connected	4096 x 256
RELU	
Dropout	0.5
Fully connected	256 x 128
RELU	
Dropout	0.5

Fully connected	128 x 43
Softmax	

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.

To train the model, I used Adam optimizer with a learning rate of 0.0008. The final batch size was 80 and the epochs for training were 15.

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 1.000.
- validation set accuracy of 0.976.
- test set accuracy of 0.961.

The first model tested was the LeNet one with some basic preprocessing steps such as normalization and grayscale. Although the model could beat the 0.93 benchmark, I decided to create a model with a higher number of parameters because the number of classes (43) was a lot bigger than the MNIST one and there was room for improvement based on the results. For that reason the number of channels in the two convolutional layers were changed to 32 and 64 respectively and three fully connected layers were also used. In order to avoid overfiiting, dropout was added after the first two fully connected layers. Max-pooling was also used after the convolutional layers and this effectively reduces the size of the model but it also make it more robust to small translations or distortions.

The next step was to select values for the different hyperparameters, namely the learning rate, the batch size and the number of tarining epochs. For the batch size, I tested values 256, 128, 80 and for the learning rate I tested values 0.001, 0.0009, 0.0008, 0.0006, 0.0001.

At this stage, I decided to re-examine the pre-processing strategy and tested Adaptive Histogram Equalization which also improved the validation score and I decided to replace normalization with this approach.

The final selection parameter was the number of epochs, which were selected in a way that prevents overtraining / overfitting, mainly by looking the validation and training scores and making sure that the validation score is improving and does not deteriorate.

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:

The first image might be difficult to classify because …

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:

Actual Class	Actual label	Predicted label
13	Yield	Yield
11	Right-of-way	Right-of-way in next intersection
18	General caution	General caution
12	Priority road	Priority road
17	No entry	No entry
14	Stop	Stop

The model was able to correctly guess 6 out of the 6 traffic signs, which gives an accuracy of 100%. This compares favorably to the accuracy on the test set of ~96% and it can be related to the fact that all these signs belong to well represented classes in the training set.

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 code for making predictions on my final model is located in the 19th cell of the Ipython notebook. The top 5 softmax probabilities are calculated in the 21st cell.

For all images the models is very certain about its predictions and the probabilities calculated are very close to 1.0. The top five soft max probabilities for all images are given in the tables below.

First image:

Probability	Class	Prediction label
1.0	13	Yield
.00	12	Priority road
.00	35	Ahead only
.00	33	Turn right ahead
.00	39	Keep left

Second image:

Probability	Class	Prediction label
1.0	11	Yield
.00	30	Beware of ice/snow
.00	27	Pedestrians
.00	21	Double curve
.00	40	Roundabout mandatory

Third image:

Probability	Class	Prediction label
1.0	18	General caution
.00	26	Traffic signals
.00	27	Pedestrians
.00	25	Road work
.00	24	Road narrows on the right

Fourth image:

Probability	Class	Prediction label
1.0	12	Priority road
.00	40	Roundabout mandatory
.00	14	Stop

.00	17	No entry
.00	13	Yield

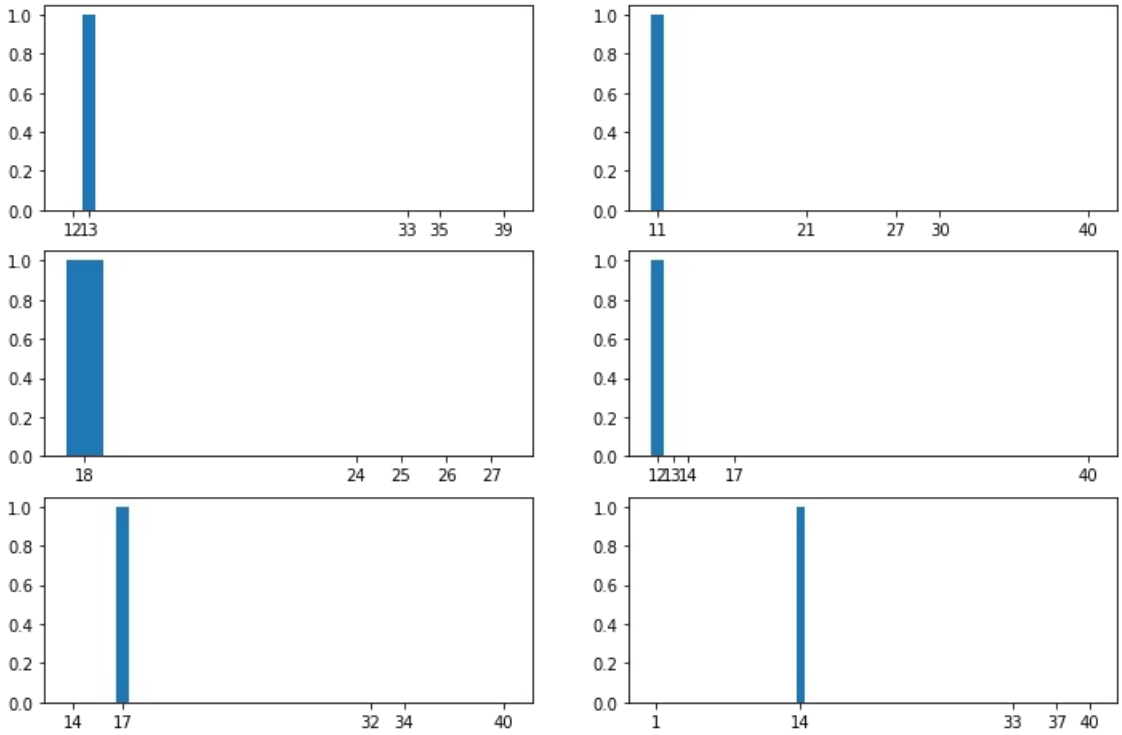
Fifth image:

Probability	Class	Prediction label
1.0	17	No vehicles
.00	14	Stop
.00	40	Roundabout mandatory
.00	34	Turn left ahead
.00	32	End of all speed and passing limits

Sixth image:

Probability	Class	Prediction label
1.0	14	Stop
.00	1	Speed limit (30km/h)
.00	33	Turn right ahead
.00	37	Go straight or left
.00	40	Roundabout mandatory

The same results are also illustrated with the use of bar charts:


1.0
0.8
0.6
0.4
0.2
0.0
12 13 33 35 39
11 21 27 30 40
18 24 25 26 27
12 13 14 17 40
14 17 32 34 40
1 14 33 37 40