Traffic Sign Recognition

Writeup for Project 2

Paul Comitz 7/30/2017

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 (https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

Provide a Writeup / README

You're reading it! and here is a link to my project code (https://github.com/pcomitz/Traffic-Sign-Classifier)

Data Set Summary & Exploration

A basic summary of the data set.

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

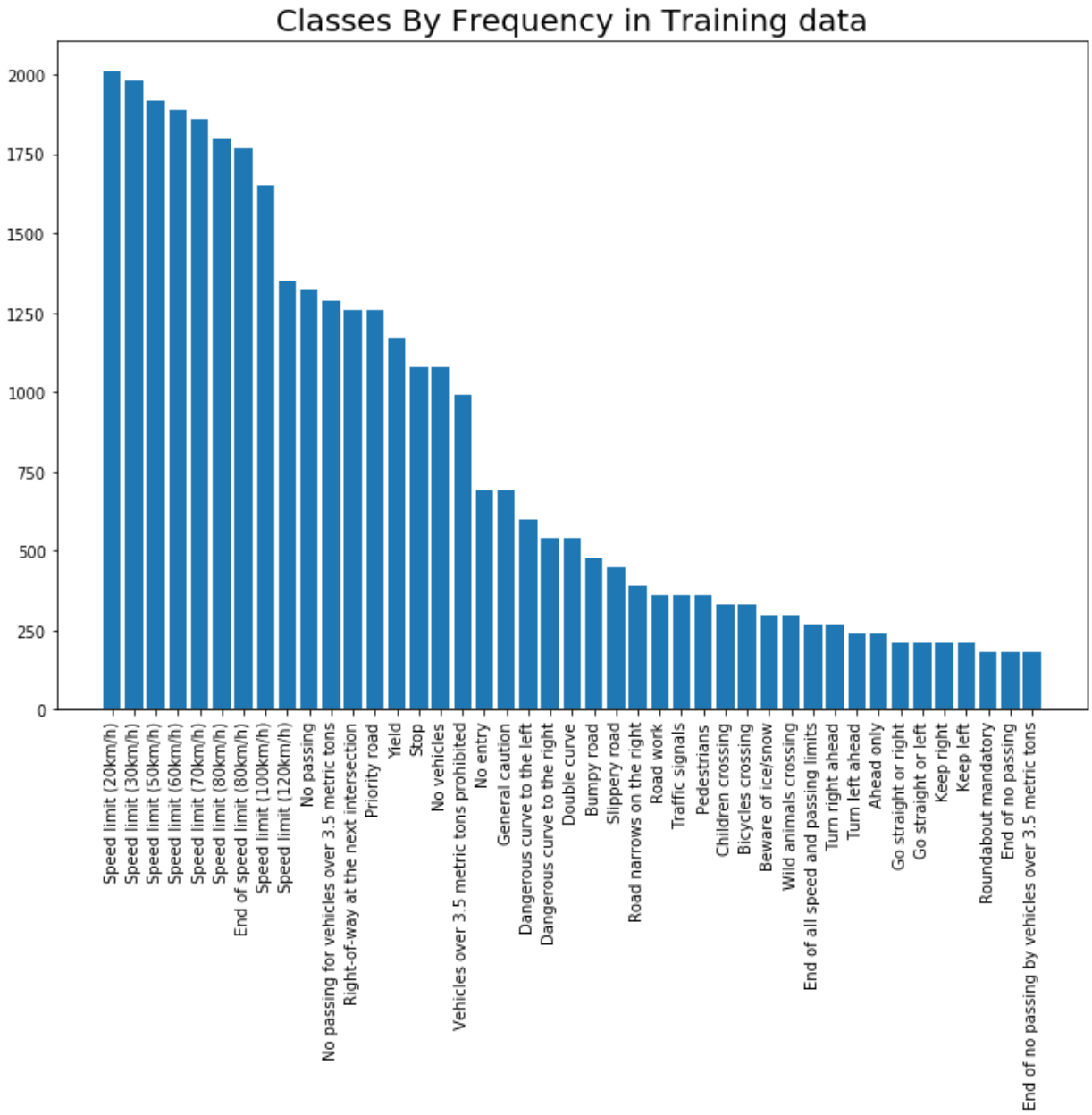
- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32,32,3)
- The number of unique classes/labels in the data set is 43

Include an exploratory visualization of the dataset.

The visulaizations use the matplotlib and pandas libraies. Here is an exploratory visualization of the data set. The first image shows five randomly selected images from the training set. The classes for each image are shown above the image. In this particular random sample, the classes that are shown are:

- Right-of-way, label 11
- No passing for vehcles > 3,5 metric tons, label 10
- Stop Sign, label 14
- Priority Road, label 12
- Speed Limit 30 km/h, label 1
- Yield , label 13
- Speed Limit 120 km/h, label 8

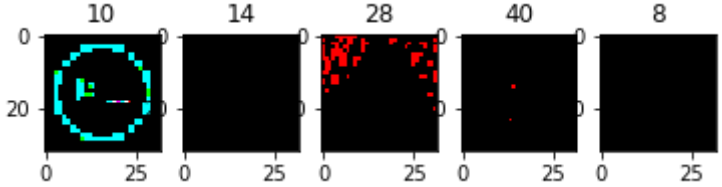
The figure below shows a distribution of the classes in the training dataset. This visualization was shown to me by a mentor. The graphc shows that the speed limit signs are the most frequent.



Design and Test a Model Architecture

Preprocessing the image data.

All of the images were normalized. All images were normalized using (pixel - 128)/ 128. This was done for all three channels for all images including the test set, training set, validiation set, and the new images downloaded form the web. The normalized images from the new images downloaded from the web are shown below. The class label is shown above each normalized image.



Final model architecture

This model follows the LeNet 5 architecture shown in class. This model was chosen because I had never done a Convolutional Neural Network before. I decided it was a good idea to try to use and understand this model before moving on to other architectures. The layers are shown in the table below.

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 normalized RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding,output 10x10x16
RELU	
Max pooling	2x2 stride, outputs 5x5x16
Flatten	
Fully Connected	input = 400, output = 120
RELU	
Fully Connected	input = 120, output = 84
RELU	
Fully Connected	input = 84, output = 43

Model Training

To train the model, I followed most of the techiques and examples shown in class. The following were used:

- Learning rate 0.001
- batch size 128
- number of epochs 30
- Optimizer: AdamOptimizer (https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer). This optimization alogorithm is described here. (https://arxiv.org/abs/1412.6980)

Solution and Accuracy

My final model results were:

- training set accuracy of 0.931
- validation set accuracy of 0.931
- test set accuracy of 0.925

The approach was not very different than that shown in the class examples. As indicated above a well-known approach (LeNet 5) was chosen because this is my first Convolutional Neural Network. A reference and an example are enormously useful. Since the intent in this class is to exercise existing techniques (rather than perform research leading to new techniques), the choice of a well-know architecture follows.

I tried a number of different epochs. I did not do much experimentaion with the bacth size and learning rate. I plan to try this after I submit this assignment.

I started with 10 epochs. I moved to 20 and eventually settled on 30. In many cases the accuracy of the training set would reach 93% at around epoch 15. In some cases the accuracy would decrease on subsequent epochs and return to the 93% rate at around epoch 25. The highest accuracy observed on the training set with 30 epochs was 94.1%

I considered a greyscale implementation. I noted that the normalized images were very difficult to visualize. I also noted that many on the discussion forums referred to the use of greyscale images. I decided against a greyscale implementation because there are cases were (though not necessarily in the German system) the color of the sign conveys important information.

Test a Model on New Images

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

- label = 8

- label = 40

- label = 28

- label = 14

- label = 10

All of the images above were normalized using the identical approach used for the training set, test, set, and validation set. The images I found were 32x 32 jpg images.

I tried several sets and combination of images. In some cases the accuracy was 0. I also had results of 14% and 40% with other image sets from different sources. It is difficult to know all of the characteristic of images collected from the internet. I suspect subtle vartiations in images (intensity, brightness etc), can have a significant effect on the classifier.

The model was able to correctly guess 3 of the 5 traffic signs, which gives an accuracy of 60%. This suggests reasonable performance, but a larger set, perhaps 50 to 100 examples, would provide a better assessment.

Softmax probabilities for each prediction.

Provide the top 5 softmax probabilities for each image along with the sign type of each probability. The probabilities and the predictions are shown in the below. There are also shown in the jupyter notebook.

For the first image, the model is relatively sure that this is a No passing sign (probability of 1), and the image does contain a No passing sign. It is interesting to note that the probabilties for correctly matched images are close to 1 (reported as 1) and the other probabilities are essentially zero.

Probability	Prediction
1.0	No passing over 3.5
1.18e-15	End no passing
2.19e-18	No passing
9.97e -24	Bumpy Road
1.2e -29	Slippery Road

For the remaining images, the softmax probabilities are approximately 1 when there is match, and essentially 0 for the other choices. This was true for 3 (including the image shown above) of tne new images downloaded from the web. Those results are:

- Image 2 Stop Sign
  - Probability 1, prediction Stop sign
- Image 3 Children Crossing
  - Probabilty 1, prediction Children Crossing
- Image 4 Roundabout Mandatory
  - Probability 0.09, prediction Road narrows
- Image 5 Speed Limit 120 km/h
  - Probability 0.09, prediction Road work

In [ ]:
```
1
```