Traffic Sign Recognition

David Lopez Rubio Writeup

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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

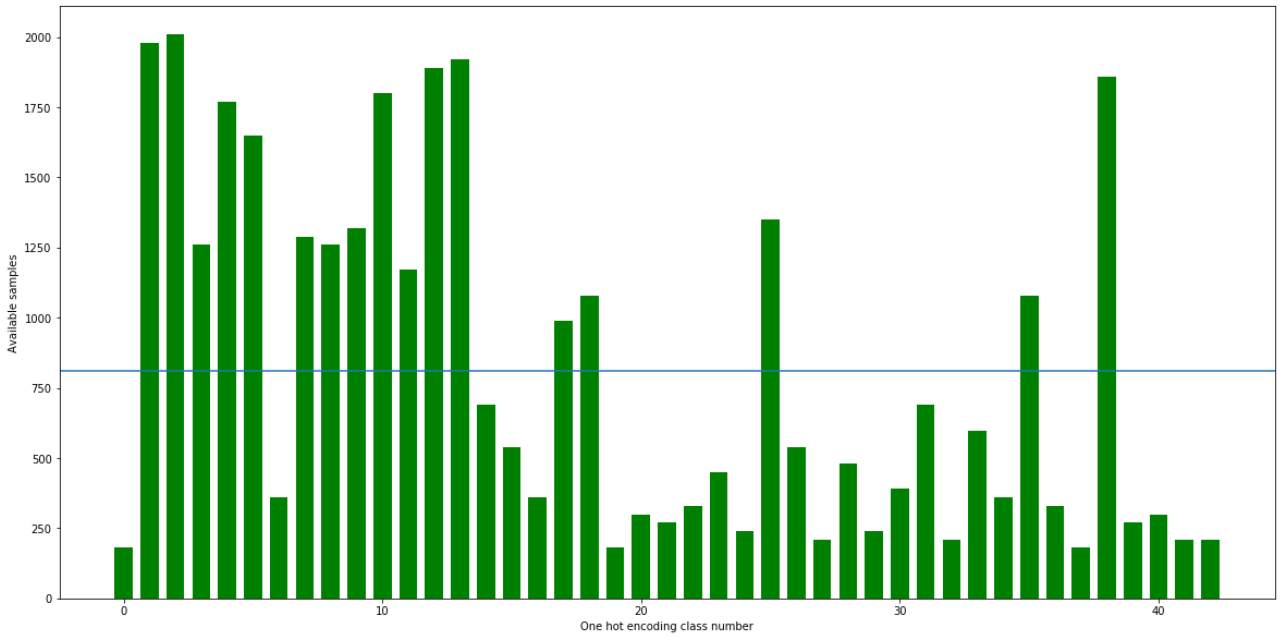
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

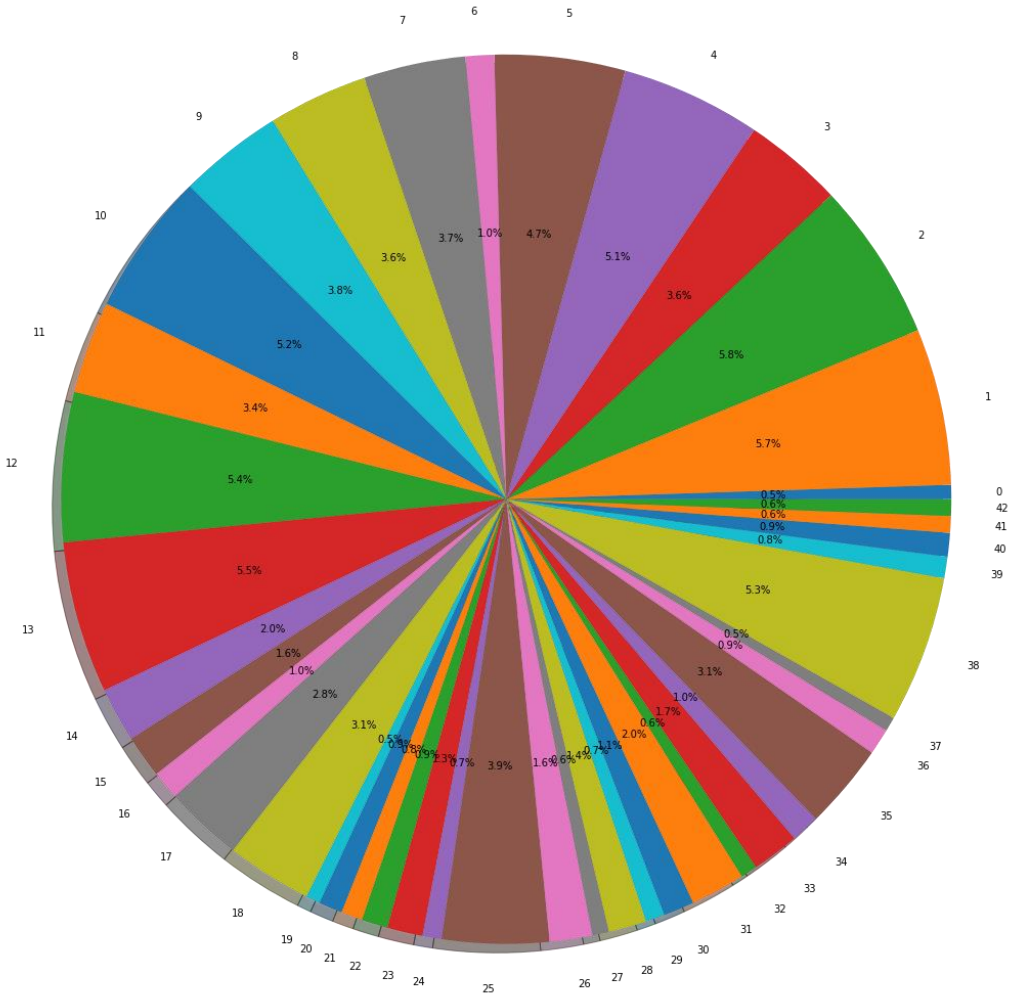
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 4410
- The shape of a traffic sign image is 32x32x3 (32x32 pixel RGB)
- The number of unique classes/labels in the data set is 43 different signs

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. Histogram is samples per label, horizontal line is the average of samples per label.





Design and Test a Model Architecture

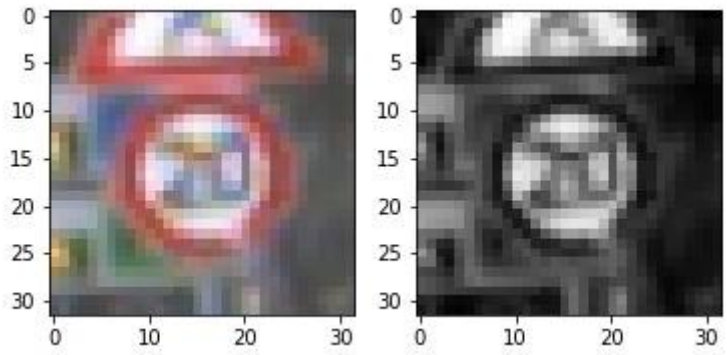
1. Describe how you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc. (OPTIONAL: As described 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 and normalize because is the main recommendation. Also, the suggested paper achieves a good result. The main reason I think is a good idea to start with grayscale, is

that uses tensors with 1/3 of size (only one gray pixels matrix instead of RGB) and then training is lighter and quicker.

After some more research and testing myself with a local sign with yellow background (construction locations use yellow backgrounded signs in Spain) I understand that sometimes using color information can be useful.

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



As a last step, I normalized the image data because ...

To add more data to the the data set, I used the following techniques because ...

My dataset is not augmented, no transformation, no blurr, no rotations, no light changes... Since I am a bit on rush and project specifications require only 92% accuracy on training and preliminary training achieves 96% I did not perform any dataset augmentation at all.

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: (Standard LeNet with adapted shapes)

Layer	Description
Input	32x32x1 grayscale image
Convolution 3x3	1x1 stride, same padding, outputs 32x32x64
RELU	

Layer	Description
Max pooling	Input 28x28 output 14x14
Convolution 3x3	Outputs 10x10x16
RELU	
Pooling	Max pooling output 5x5
Flatten	Output 400
Fully connected	Output 120
RELU	
dropout	
Fully connected	Input 120 output 84
Relu	
dropout	
Fully connected	Input 84 output 43 (43 different signs)

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.

Adam optimizer as suggested on Tensorflow documentation. Batch size is very big, 250 since images in grayscale are small and I have an Nvidia 1080Ti with 12GB of RAM (training networks with color images I have to lower the batch size). I use 38 epoch with a low learning rate, of 0.0003, but I have tested larger LR and with around 20 epoch will suffice affecting around 0.01% in the model performance.

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 0.964
- validation set accuracy of 0.964
- test set accuracy of 0.800

If a well known architecture was chosen:

- What architecture was chosen?

  LeNet architecture was chosen as suggested in Udacity course. LeNet is a network that works with images, is easy to port to similar model requirements, has good performance and is almost plug and play!

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

  Traffic signs are easy to reduce to a small images as LeNet was made for.

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

  Training is enough performance. The validation step gives us a not overfitted model. Also, the fact that works with 80% in a small batch of signals that are a bit different (from Spain), tells us that the network can generalize well enough.

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 traffic signs from Spain, some are from web , some are photographed by me

Note that the child crossing signal is a bit different in Spain, this is the UK one, more similar to German one.

The passing sign is different too, is the construction works used sign which have yellow background and is meant to be temporal. Should be OK for the net to recognize since my model works in grayscale.

The banned pass signal photo is taken near my house, and has another yield signal behind, so it can be a bit difficult for the net to recognize a round shape.

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:

```
INFO:tensorflow:Restoring parameters from .\lenet_traffic_signs
Real world test accuracy = 0.800
```

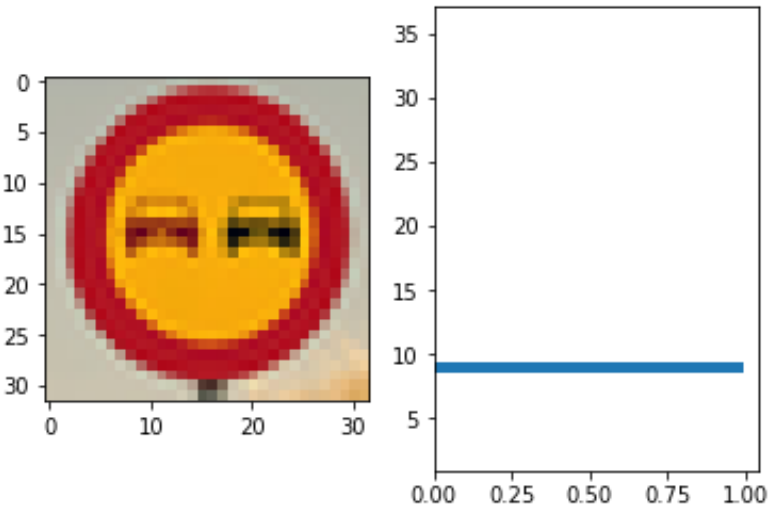
The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%.

Lets see the results in deep.

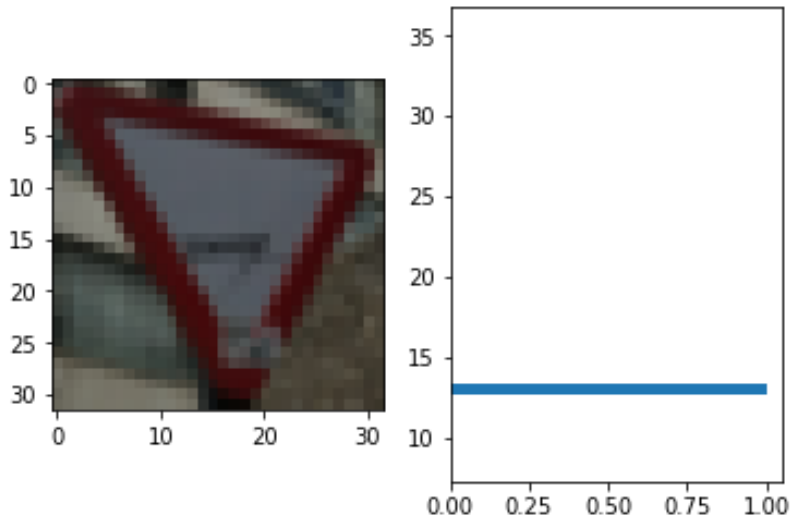
3. Describe how certain the model is when predicting on each of the five new images by looking at the softmax probabilities for each prediction. Provide the top 5 softmax probabilities for each image along with the sign type of each probability. (OPTIONAL: as described in the "Stand Out Suggestions" 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 11th cell of the Ipython notebook.

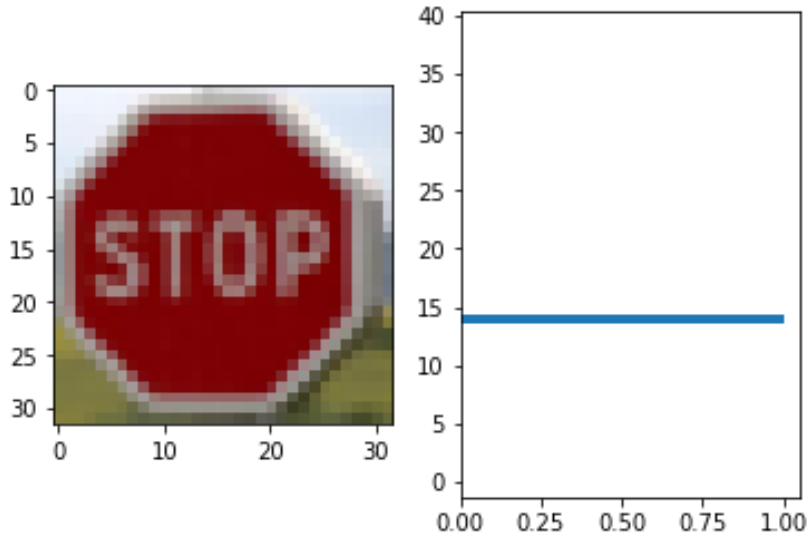
```
Sign:  [9 'No passing']
Predicted:  [9 'No passing']
```




```
Sign:  [13 'Yield']
Predicted:  [13 'Yield']
```

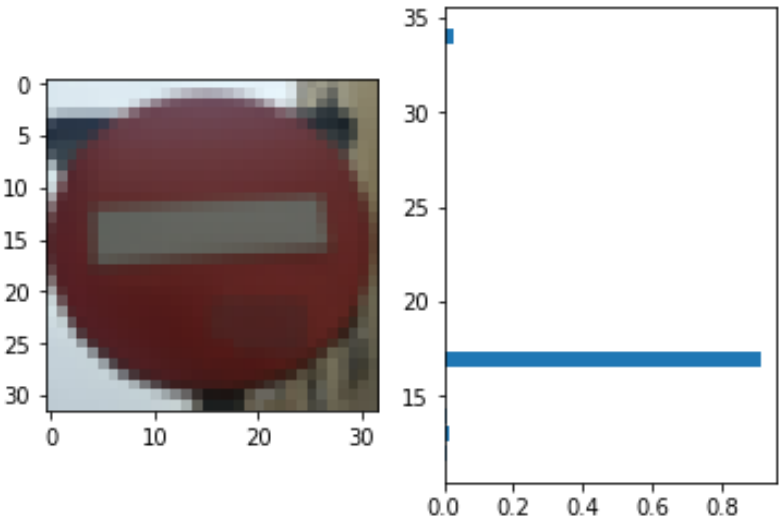


```
Sign:  [14 'Stop']
Predicted:  [14 'Stop']
```

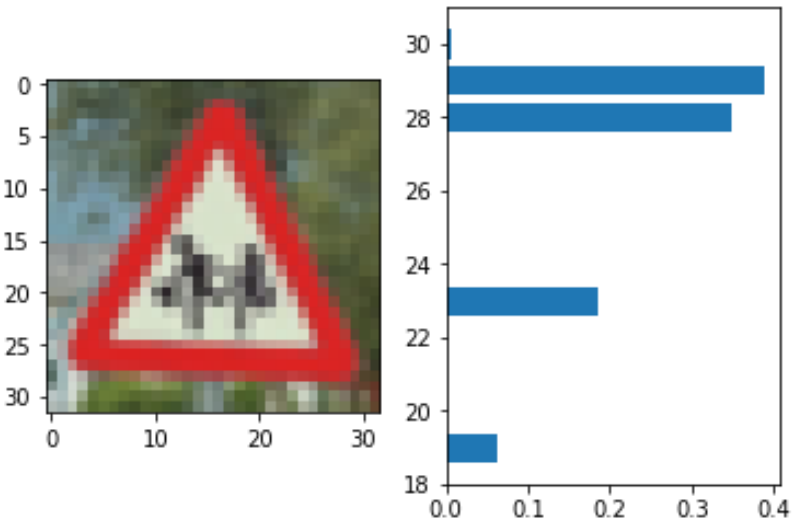


```
Sign:  [17 'No entry']
```

```
Predicted:  [17 'No entry']
```




```
Sign:  [28 'Children crossing']
Predicted:  [29 'Bicycles crossing']
```



For a little probability in softmax function the result is incorrect!! But you have to keep in mind that this signal is not from Germany, is from Spain, and the net has to generalize (the net NEVER seen this sign before), so not a bad work at all.

For the rest of signs, the net seems to do a very good work,gives predictions with almost 99% certain (no softmax probability on other signs)