**Traffic Sign Recognition**

**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**](https://review.udacity.com/#!/rubrics/481/view) **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](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic_Sign_Classifier.ipynb)

**Data Set Summary & Exploration**

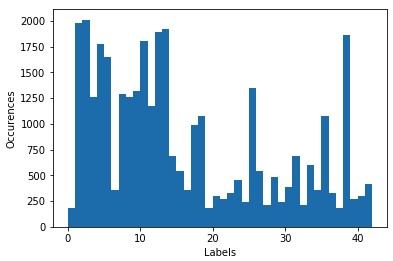
**1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.**

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.

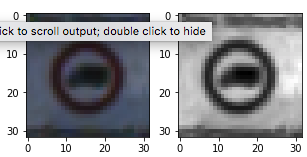


**Design and Test a Model Architecture**

1. **I’ve used**
2. **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 in order to minimize the filter depth and to make the training process a little faster.

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



But then I’ve seen that the accuracy is not matching the requirements and I had to go back to 3 channels solution.

The results where a little better which make sense, than different from other objects the colours of the traffic signs are not arbitrary chosen and cannot vary for the same sign.

First, I’ve normalized the image data as suggested around

zero, but the try to normalize the data from 0 to 1 brought me a

better result. To be honest I do not really understand, why we

have to normalize the data for a ReLU activation function. In

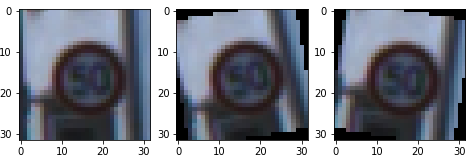
case of the sigmoid function this is clear. But the case with the

low gradient dos not seem to be the reason here.

Cause I still have been on the edge to miss the requirement I

decided to create more training data. I’ve seen that some pictures are slightly turn to one or another side so I’ve decided to rotate the images a little, which brought a much better result.

Here is an example of an original image and an augmented image:



**2. For the Traffic Sign Classification project, I’ve used the provided LeNet model. But to reach the required accuracy I’ve had to make some changes.**

**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 | 32x32x3 RGB image |
| Convolution 5x5 | 1x1 stride, same padding, outputs 28x28x6 |
| RELU |  |
| Max pooling | 2x2 stride, outputs 16x16x64 |
| Convolution 3x3 | etc. |
| Fully connected | etc. |
| Softmax | etc. |
|  |  |
|  |  |

**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 an ....

**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 ?
* validation set accuracy of ?
* test set accuracy of ?

If an iterative approach was chosen:

* What was the first architecture that was tried and why was it chosen?
* What were some problems with the initial architecture?
* 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.
* Which parameters were tuned? How were they adjusted and why?
* 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?

If a well known architecture was chosen:

* What architecture was chosen?
* Why did you believe it would be relevant to the traffic sign application?
* How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

**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:

[](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/examples/placeholder.png) [](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/examples/placeholder.png) [](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/examples/placeholder.png) [](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/examples/placeholder.png) [](https://github.com/udacity/CarND-Traffic-Sign-Classifier-Project/blob/master/examples/placeholder.png)

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:

|  |  |
| --- | --- |
| **Image** | **Prediction** |
| Stop Sign | Stop sign |
| U-turn | U-turn |
| Yield | Yield |
| 100 km/h | Bumpy Road |
| Slippery Road | Slippery Road |

The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares favorably to the accuracy on the test set of ...

**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.

For the first image, the model is relatively sure that this is a stop sign (probability of 0.6), and the image does contain a stop sign. The top five soft max probabilities were

|  |  |
| --- | --- |
| **Probability** | **Prediction** |
| .60 | Stop sign |
| .20 | U-turn |
| .05 | Yield |
| .04 | Bumpy Road |
| .01 | Slippery Road |

For the second image ...