**Traffic Sign Recognition**

**Writeup Template**

**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)

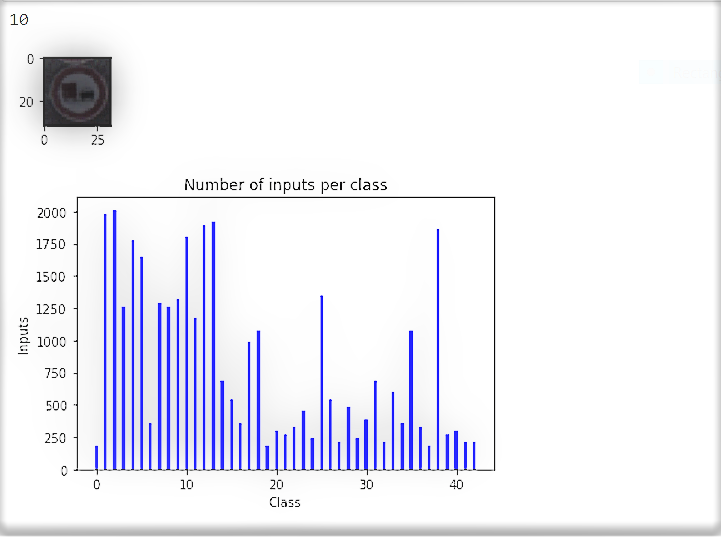
**Data Set Summary & Exploration**

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

The code for this step is contained in the second code cell of the IPython notebook.

* The size of training set is 34799
* The size of test set is 12630
* The size of the validation 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 and identify where the code is in your code file.** The code for this step is contained in the fourth code cell of the IPython notebook. Here is an exploratory visualization of the data set. I created a bar chart and visualized one image.

****

**Design and Test a Model Architecture**

**3. Describe how, and identify where in your code, 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.**

The code for this step is contained in the fifth and sixth code cell of the IPython notebook.

As a first step, I decided to convert the images to grayscale for hopefully making the neural network faster. All images were also normalized to reduce computational complexity.

Also,   TensorFlow's tf.image.rgb\_to\_grayscale() function was taking a long time to process so I decided to use cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY). OpenCV's gray scaling was giving back only height and depth so I used numpy’s newaxis feature to add back the additional depth of 1.

I also normalized the data similar to what was discussed in the lectures.

**4. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (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, identify where in your code, and provide example images of the additional data)**

The training, validation and test data sets were already provided. I am reading the sets and adding them to lists in the first cell.

Number of training examples = 34799

Number of testing examples = 12630

Number of testing examples = 4410

**3. Describe, and identify where in your code, 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.**

The code for my final model is located in the sixteenth cell of the ipython notebook.

I used a convolutional neural network with mostly the same architecture as explained in the lectures with two convolutional layers and three fully connected layers. I added dropout to the second fully connected layer, however, I found that it did not increase accuracy so I decided not to include it.

My final model consisted of the following layers:

| **Layer** | **Description** |
| --- | --- |
| Input | 32x32x1 image |
| Convolution 1 | 1x1 stride, valid padding, outputs 28x28x6 |
| RELU | Activation Layer |
| Max pooling | Output 14x14x6 |
| Convolution 2 | 1x1 stride, valid padding, Output 10x10x6 |
| RELU | Activation Layer |
| Max Pooling | 2x2 stride, Output 5x5x16 |
| Flatten data | Output 400 |
| Fully connected 1 | Input = 400. Output = 120. |
| RELU | Activation Layer |
| Fully connected 2 | Input = 120. Output = 84. |
| RELU | Activation Layer |
| Fully connected 3 | Input = 84. Output = 43. |
|  |  |
|  |  |

**4. Describe how, and identify where in your code, 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.**

The code for training the model is located in cells 17, 18 and 19 of the ipython notebook.

To train the model, I used an Adam Optimizer and the code similar to what was discussed in the lectures. I experimented with various hyper parameters including batch size, Epoch’s. Initially I settled on batch size = 150 and Epochs = 15. However, with these parameters, I was getting an accuracy of .875. After my project was reviewed and returned requesting to improve the accuracy to a value higher than .93, I again experimented with different parameters and finally setting these parameters worked - Batch size = 128 and Epochs = 100. I decided to keep mu = 0 and sigma = 0.1, Learning rate = .001. I changed Learning rate to .005, however, that did not improve validation accuracy so I decided to stick to learning rate = .001. Also, due to gray scaling and adding back a dimension, the inputs to my network needed to be at 32x32x1 instead of 32x32x3.

**5. Describe the approach taken for finding a solution. 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.**

The code for calculating the accuracy of the model is located in cells 20 and 21 of the Ipython notebook.

My final model results were:

* validation set accuracy of 0.94
* test set accuracy of 0.917

Initially when I started training the model, I got a very low accuracy of 0.060. I checked the code again and realized that I had not added the pre-processing for the validation data set. After adding that I was able to get an accuracy of .875 or above.

Also, after my project was reviewed and returned requesting to improve the accuracy to a value higher than .93, I again experimented with different parameters and finally setting these parameters worked - Batch size = 128 and Epochs = 100. The accuracy increased to .94.

I used the CNN with 2 convolutional layers, 3 fully connected layers, a learning rate of .001 and a batch size 128. I experimented with changing the learning rate to .005, however, it did not increase the validation accuracy. I wanted to try this architecture first because it was used in the lectures and the accuracy was good. I added dropout as well, however, I noticed that it did no improve the accuracy so I removed it from the final code. Since I was able to get an accuracy of .94, I was satisfied with the model.

**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:

Downloaded images:



The five images I added were – “speed limit (30km/h)”, stop sign, yield sign, general caution sign and a “no entry” sign. I feel that the “Speed limit (30km/h)” and stop signs will be the most difficult to classify because the images there are letters and words on the signs which may be difficult to verify. Also, it could be difficult to predict the actual speed of the sign since all speed signs are circular. I am hoping that the yield, Caution and do not enter signs will be easier to predict since these are more unique compared to other signs.

**2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. 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).**

The code for making predictions on my final model is located in cells 22 to 26 of the Ipython notebook.

Here are the results of the prediction:

| **Image** | **Prediction** |
| --- | --- |
| Speed limit (30km/h) | Keep right (38) |
| Stop sign | Speed limit (60km/h) (3) |
| Yield sign | Slippery road(23) |
| General caution sign | General caution sign (18) |
| No entry | No entry sign (17) |

The model was able to correctly guess 2 of the 5 traffic signs, which gives an accuracy of 40%. This is much lower than the accuracy of the test 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 and identify where in your code softmax probabilities were outputted. 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 cells 27 to 31 of the Ipython notebook.

**For the first image Speed limit (30km/h) and the top five soft max probabilities were –**

| **Probability** | **Prediction** |
| --- | --- |
| 558.97625732 | 38 ( Keep right ) |
| -233.66937256 | 13 ( Yield ) |
| -613.33380127 | 10 ( No passing for vehicles over 3.5 metric tons ) |
| -1090.90612793 | 2 ( Speed limit (50km/h) ) |
| -1548.56079102 | 14 ( Stop ) |

The model was not able to correctly predict the speed sign of ‘30’. I had assumed that it would be difficult to predict and my assumption was correct.

**For the second image** Stop sign **and the top five soft max probabilities were -**

| **Probability** | **Prediction** |
| --- | --- |
| 7857.65039062 | 3 ( Speed limit (60km/h) ) |
| 3630.94946289 | 35 ( Ahead only ) |
| 3090.96972656 | 13 ( Yield ) |
| 3016.63598633 | 9 ( No passing ) |
| 2097.99194336 | 28 ( Children crossing ) |

In the case of the stop sign (14), it did not get predicted correctly at all. The highest probability was assigned to “Speed limit (60km/h)”.

**For the third image** yield sign **and the top five soft max probabilities were -**

| **Probability** | **Prediction** |
| --- | --- |
| 3327.87963867 | 23 ( Slippery road ) |
| 2838.98608398 | 35 ( Ahead only ) |
| 2553.94824219 | 20 ( Dangerous curve to the right ) |
| 1083.94287109 | 28 ( Children crossing ) |
| 718.98565674 | 36 ( Go straight or right ) |

Again, for this image yield sign **(13)**, it did not get predicted correctly with the highest probabilities going to other signs.

**For the fourth image,** general caution **and the top five soft max probabilities were -**

| **Probability** | **Prediction** |
| --- | --- |
| 17305.828125 | 18 ( General caution ) |
| 2632.43164062 | 38 ( Keep right ) |
| 2484.12841797 | 25 ( Road work ) |
| -427.77050781 | 11 ( Right-of-way at the next intersection ) |
| -546.00085449 | 1 ( Speed limit (30km/h) ) |

This image did get predicted correctly as “General Caution”.

**For the fifth image** no entry, **the top five soft max probabilities were -**

| **Probability** | **Prediction** |
| --- | --- |
| 35403.30078125 | 17 ( No entry ) |
| 10030.38183594 | 9 ( No passing ) |
| 3596.79418945 | 0 ( Speed limit (20km/h) ) |
| 3347.59375 | 40 ( Roundabout mandatory ) |
| 991.40142822 | 37 ( Go straight or left ) |

This image was correctly predicted.