#**Traffic Sign Recognition**

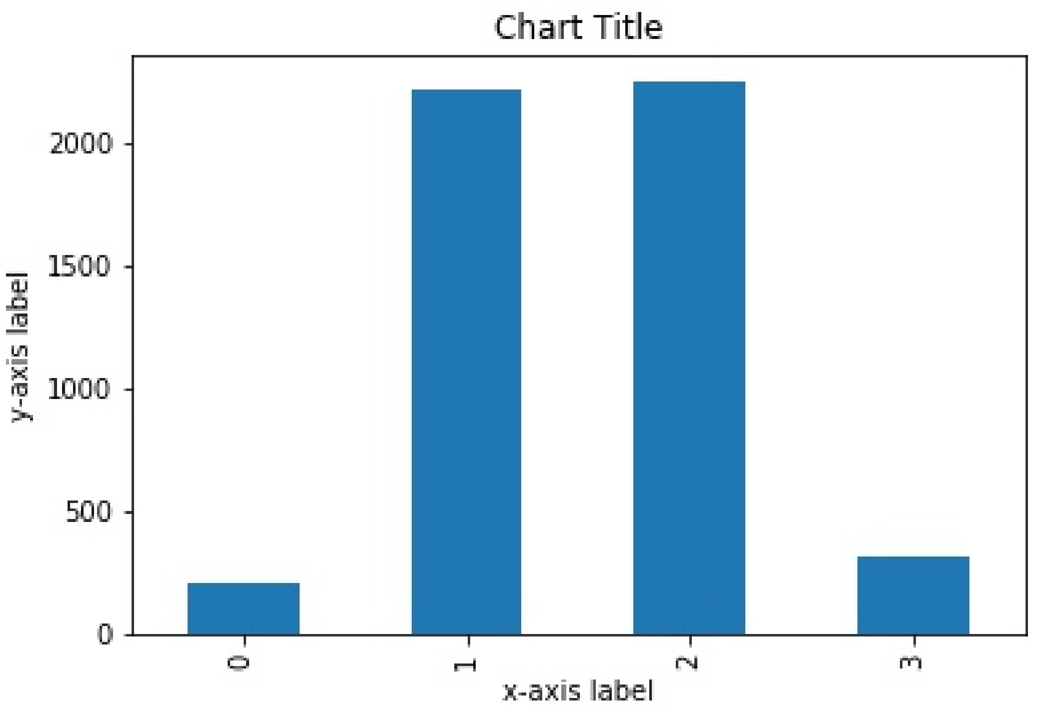
##Writeup Template

###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

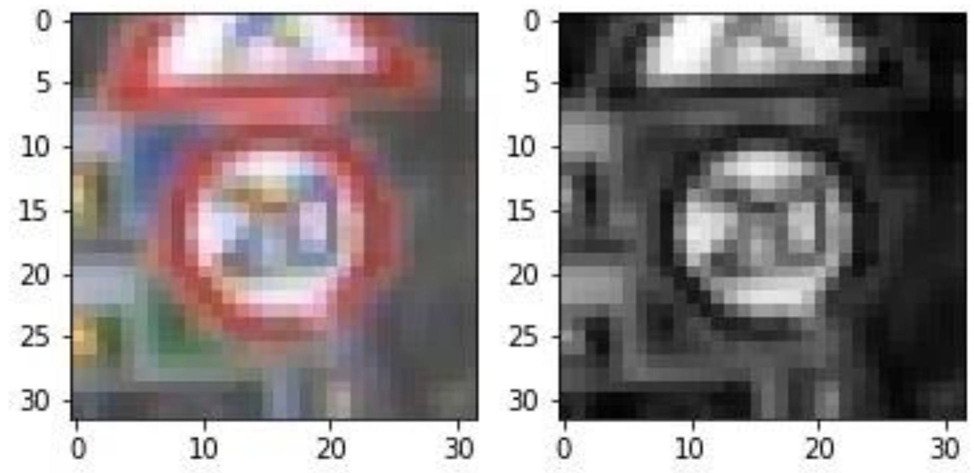
The goals / steps of this project are the following:
* Load the data set (see below for links to the project data set)
* Explore, summarize and visualize the data set
* Design, train and test a model architecture
* Use the model to make predictions on new images
* Analyze the softmax probabilities of the new images
* Summarize the results with a written report

[//]: # (Image References)

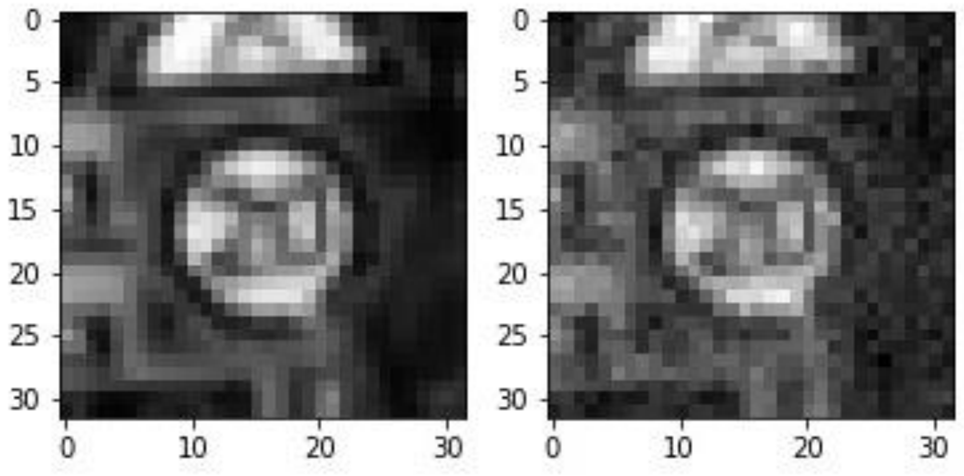


:

./examples/visualization.jpg "Visualization"



:
./examples/grayscale.jpg "Grayscaling"



:
./examples/random_noise.jpg "Random Noise"

: ./examples/placeholder.png "Traffic Sign 1"

: ./examples/placeholder.png "Traffic Sign 2"

: ./examples/placeholder.png "Traffic Sign 3"

: ./examples/placeholder.png "Traffic Sign 4"

: ./examples/placeholder.png "Traffic Sign 5"

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.

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 test set is` 12630

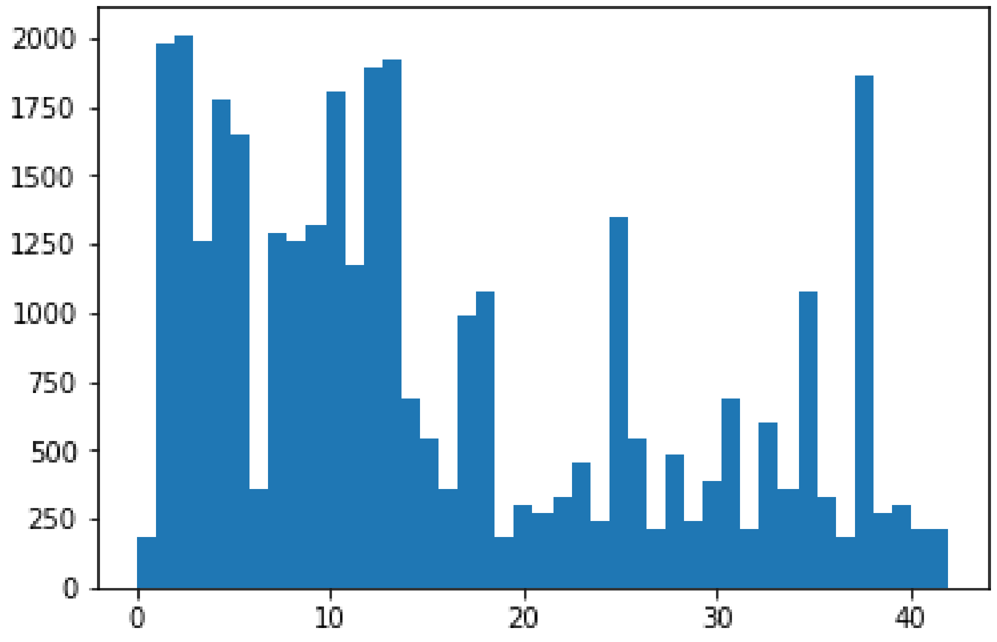
`* The shape of a traffic sign image is` 32, 32, 3

* The number of unique classes/labels in the data set is 43

####2. Include an exploratory visualization of the dataset 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 using bar chart.



###Design and Test a Model Architecture

####1. Describe how, and identify where in your code, you preprocessed the image data. What tecniques 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 code cell of the IPython notebook.

As a first step, I decided to convert the images to grayscale because grayscale will make the image shape (32,32,3) into (32,32,1)

As a last step, I Normalizes the data between 0.1 and 0.9 instead of 0 to 255

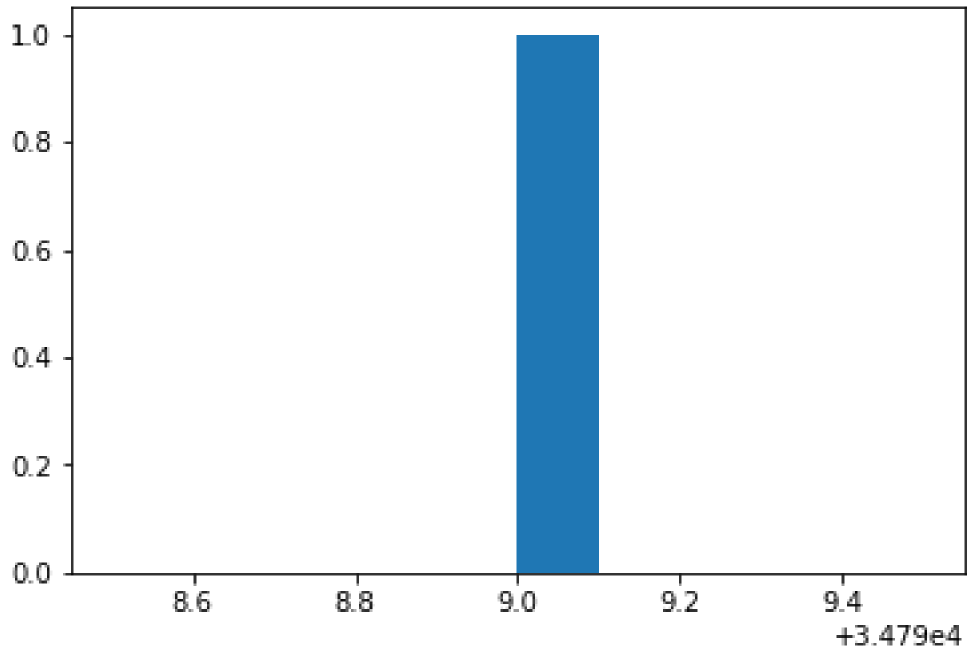
####2. Describe how, and identify where in your code, you set up training, validation and testing data. How much data was in each set? Explain what techniques were used to split the data into these sets. (OPTIONAL: As described in the "Stand Out Suggestions" part of the rubric, if you generated additional 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)

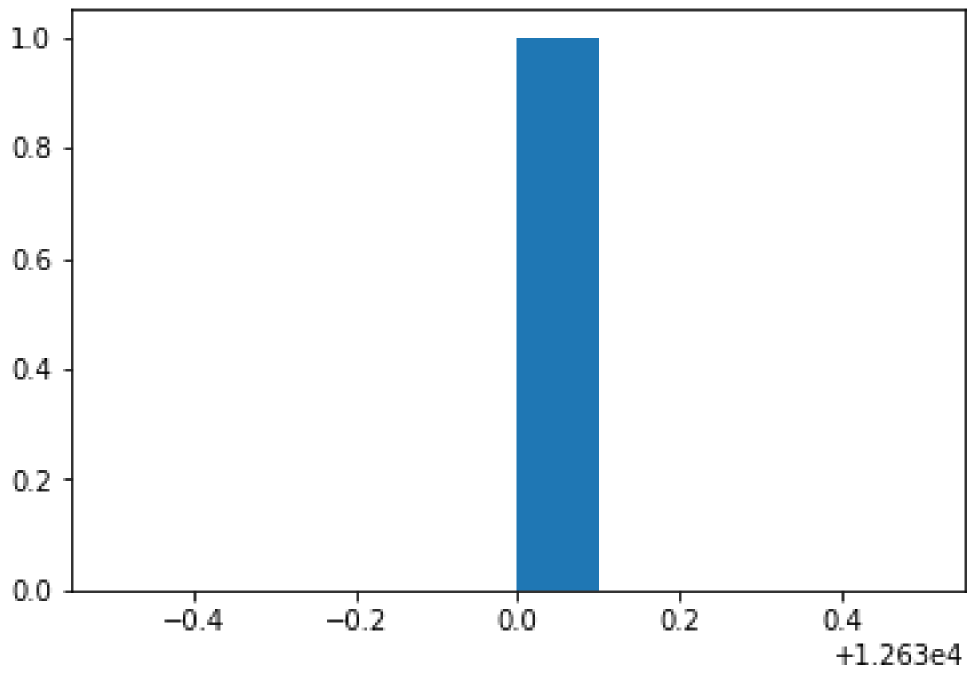
The code for training and validation sets is contained in the first code cell of the IPython notebook.

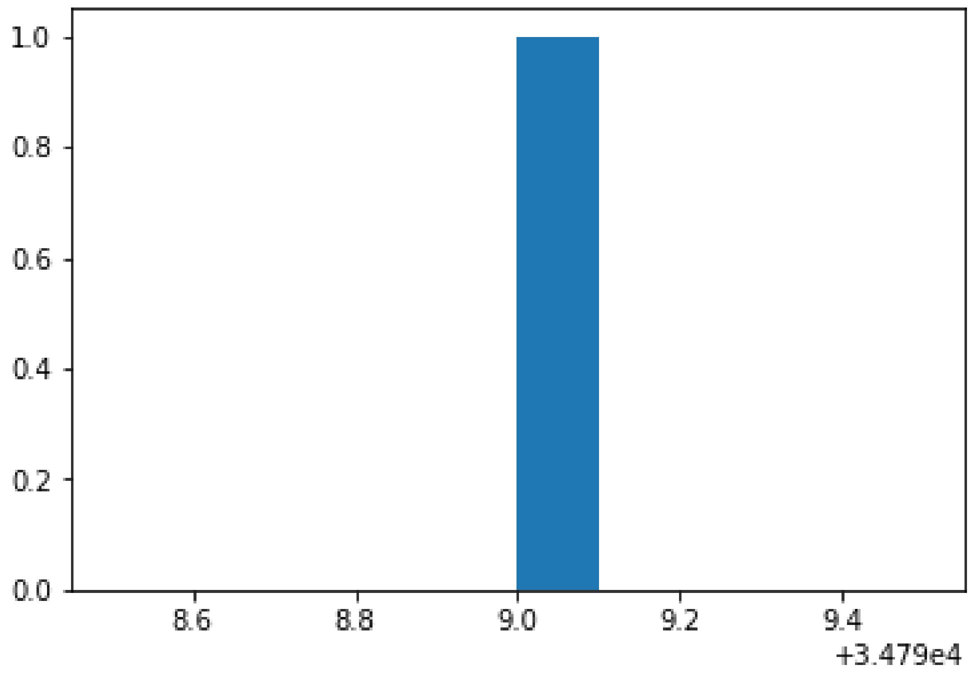
To cross validate my model, I used the training data into a training set and validation set.

```
My final training set had 34799 number of images. My validation set and test s
et had 34799 and 12630 number of images.
```

Here is the visualization of number of images present in the train, test and validation set







####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 tenth cell of the ipython notebook.

Implement the LeNet-5 neural network architecture.

Input

The LeNet architecture accepts a 32x32xC image as input, where C is the number of color channels. Since I converted traffic sign images into grayscale, C is 1 in this case.

Architecture

Layer 1: Convolutional. The output shape should be 28x28x6.

Activation. Your choice of activation function.

Pooling. The output shape should be 14x14x6.

Layer 2: Convolutional. The output shape should be 10x10x16.

Activation. Your choice of activation function.

Pooling. The output shape should be 5x5x16.

Flatten. Flatten the output shape of the final pooling layer such that it's 1D instead of 3D. The easiest way to do is by using `tf.contrib.layers.flatten`, which is already imported for you.

Layer 3: Fully Connected. This should have 120 outputs.

Activation. Your choice of activation function.

Layer 4: Fully Connected. This should have 84 outputs.

Activation. Your choice of activation function.

Layer 5: Fully Connected (Logits). This should have 43 outputs.

Output

Return the result of the 2nd fully connected layer.

####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 the twelveth cell of the ipython notebook.

To train the model, I used rate = 0.009. Then, softmax cross entropy with reduce mean and I used Adam optimizer

####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 the thirteenth cell of the Ipython notebook.

My final model results were:
* training set accuracy of 0.980
* validation set accuracy of 0.980
* test set accuracy of 0.879

If an iterative approach was chosen:

* What was the first architecture that was tried and why was it chosen?
Firstly, I tried with multi layer perceptron model. I chosen this how good it is classying the images

* What were some problems with the initial architecture?
Even though it is simple, With respect to problems with multi layer perceptron model, it did’t give the better results when compared to others.

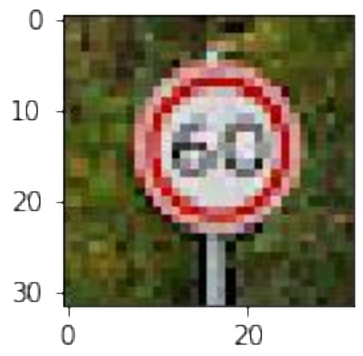
* How was the architecture adjusted and why was it adjusted? Typical adjustments could include choosing a different model architecture, adding or taking away layers (pooling, dropout, convolution, etc), using an activation function or changing the activation function. One common justification for adjusting an architecture would be due to over fitting or under fitting. 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.
Pretty much tried every combination. With respect to pooling, I tried both max pooling as well as average pooling. And regarding dropout, I went with different values like 0.9, 0.8, 0.75,etc.

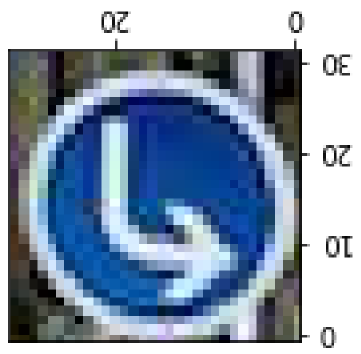
If a well known architecture was chosen:
* What architecture was chosen?
 I used LeNet architecture
* Why did you believe it would be relevant to the traffic sign application?
I believe, it is simple and powerful. Moreover, robust.
* How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
Simple and straight forward, accuracy results was the evidence

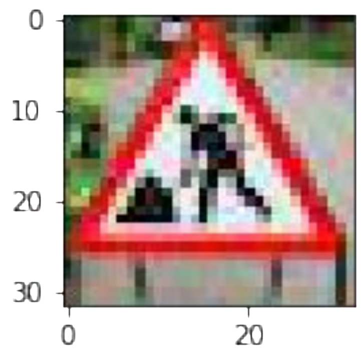
###Test a Model on New Images

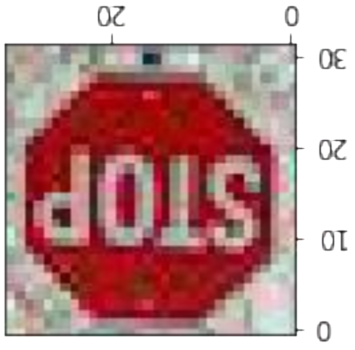
####1. Choose five German traffic signs found on the web and provide them in the report. For each image, discuss what quality or qualities might be difficult to classify.

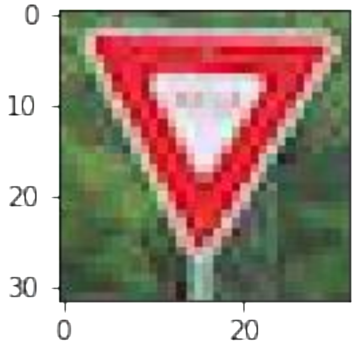
Here are five German traffic signs that I found on the web:



!


!




!


!

I intentionally rotated the second and fourth images.

####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 the twentieth cell of the Ipython notebook.

The model was able to correctly guess 1 of the 5 traffic signs, which gives an accuracy of 20%.

####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 the 22th cell of the Ipython notebook.

The model was able to correctly guess 1 of the 5 traffic signs, which gives an accuracy of 20%.

```
Image 0 probabilities: [ 6.54919815  4.06732702  2.7324872   2.144346    2.0946424 ]
 and predicted classes: [14  4 33  8  5]
Image 1 probabilities: [ 91.5749054   34.92451096  31.86141777  24.24924278  16.62455177]
 and predicted classes: [34 38  3 35 12]
Image 2 probabilities: [ 31.59500504  22.11148834  18.81607819  17.68084908   4.14453983]
 and predicted classes: [27 40 11 18 24]
Image 3 probabilities: [ 42.85917282  22.26390839  18.75226402  11.73657227   5.98850632]
 and predicted classes: [14 17 39 34 38]
Image 4 probabilities: [ 47.61253738  43.01803207  35.40981293  15.07883453  11.26122284]
 and predicted classes: [13 38  2 12 10]
```