# Traffic Sign Recognition

**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! The code is inside 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.

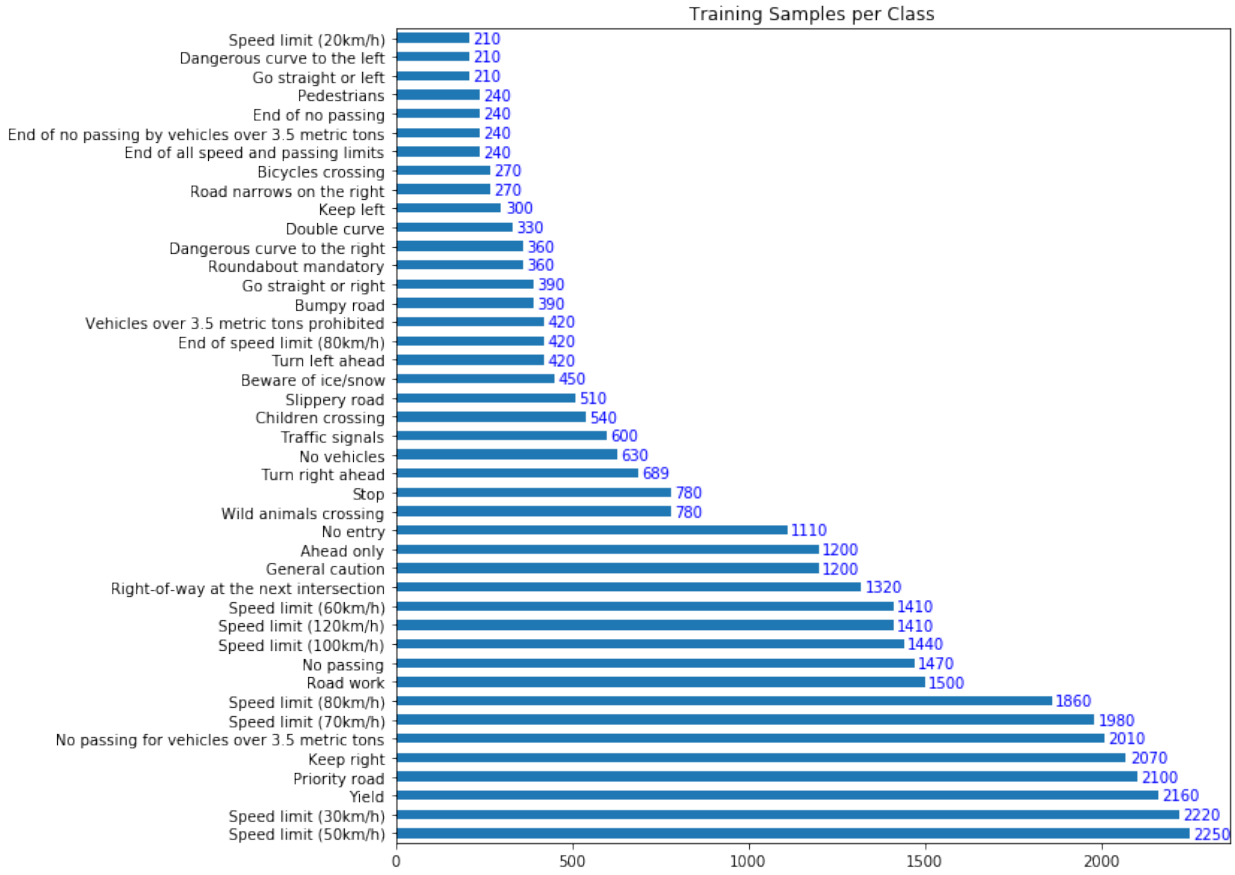
I used the pandas library to calculate summary statistics of the traffic signs data set:

- The size of training set is 39209
- The size of test set is 12630
- The shape of a traffic sign image is 29x30
- The number of unique classes/labels in the data set is 43

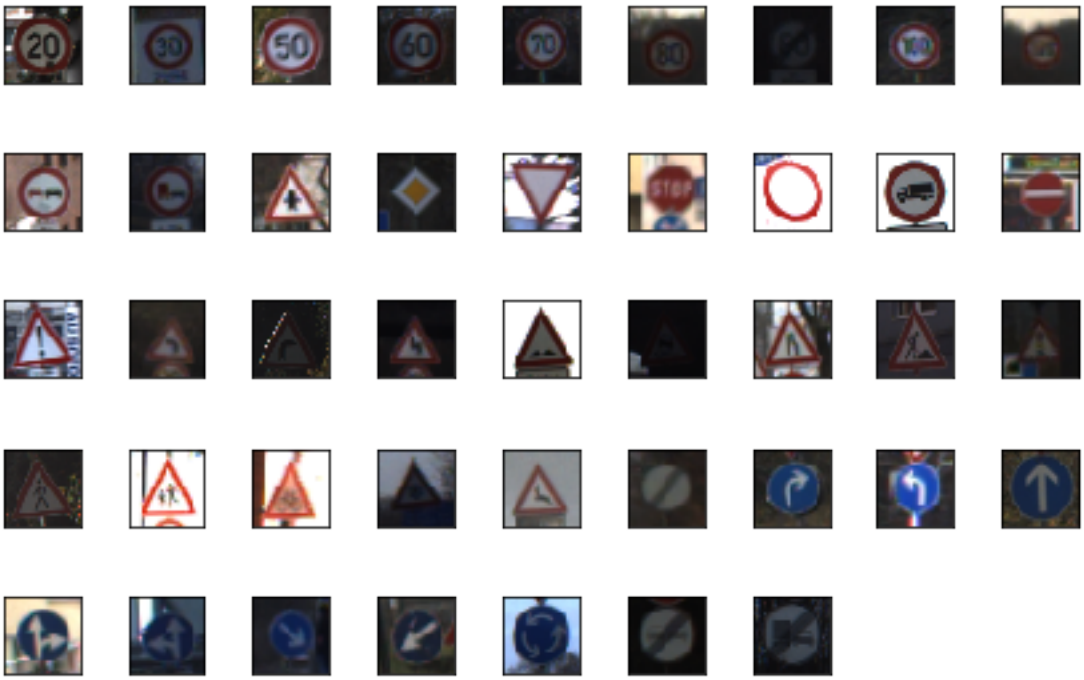
**2. Include an exploratory visualization of the dataset and identify where the code is in your code file.**

The code for this step is contained in the third and fourth code cell of the IPython notebook.

Here is an exploratory visualization of the data set. It is a bar chart showing how the data is distributed across various classes.




In fifth cell the code displays one type of sign for each class as exploration of training dataset



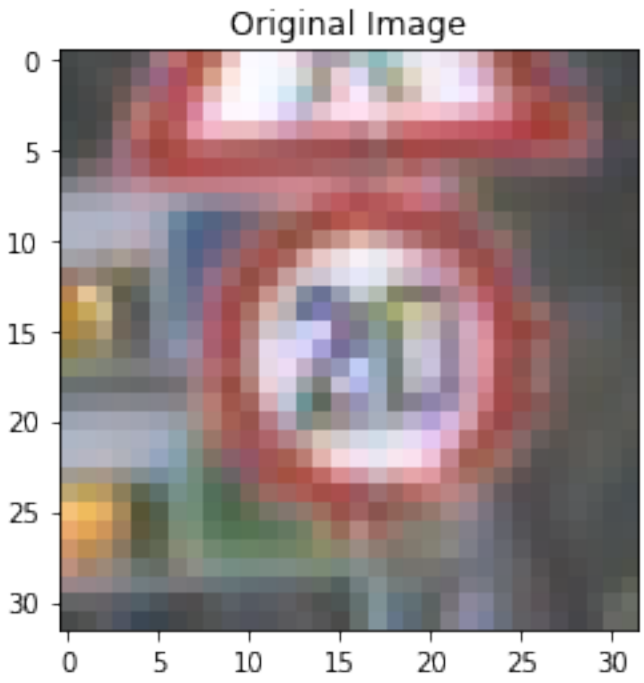
## Design and Test a Model Architecture

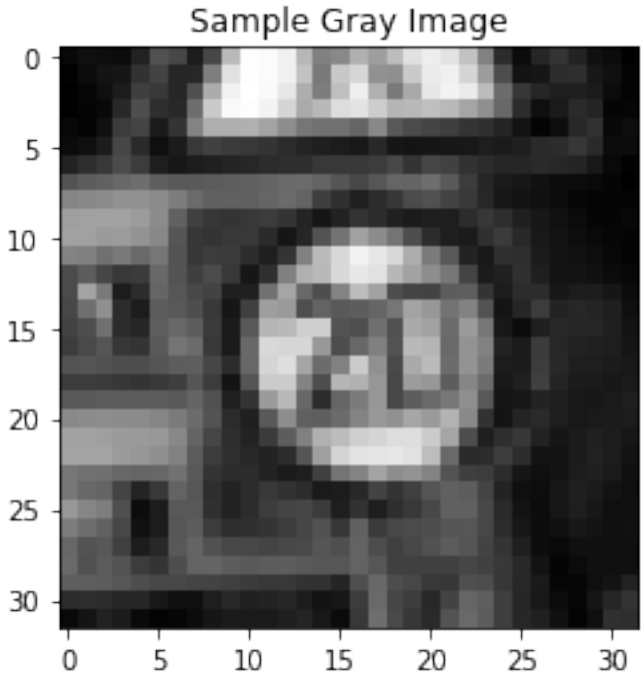
**1. Describe how, and identify where in your code, you preprocessed the image data. What techniques were chosen and why did you choose these techniques? Consider including images showing the output of each preprocessing technique. Pre-processing refers to techniques such as converting to grayscale, normalization, etc.**

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

As a first step, I decided to convert the images to grayscale because there is not much value in color as traffic sign convey the message in text and pictures. This information is available in grayscale also.

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



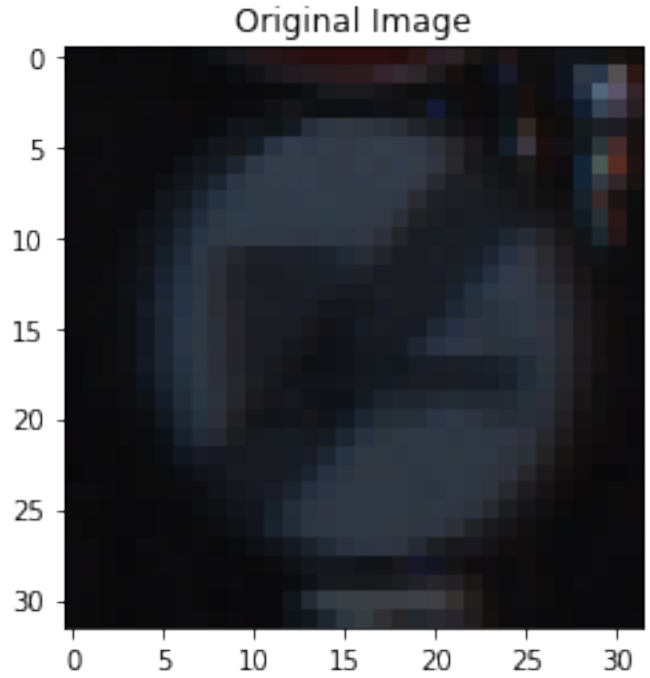

As a last step, I normalized the image data to remove outliers in data. This is needed because we want to use same learning rate

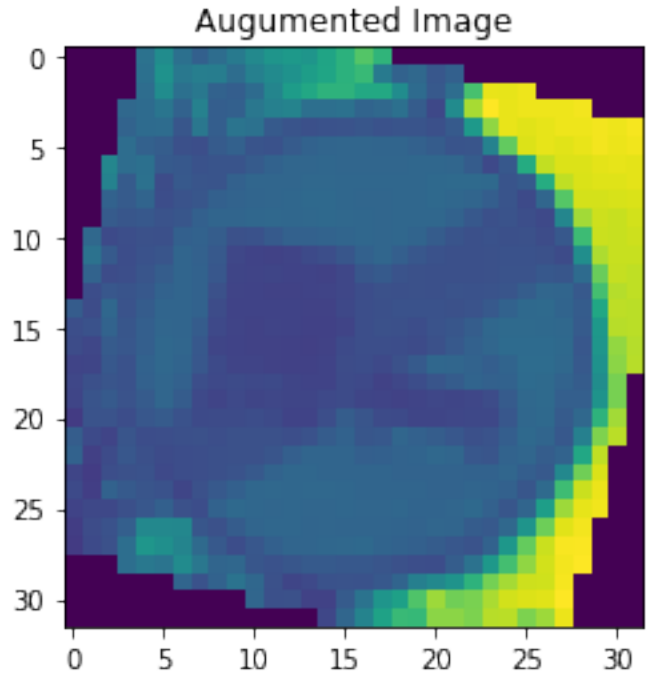
while updating the weights and don't want to under-compensate or overcompensate for outliers in the image data.

**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 augmenting data is added in seventh code cell. This made all classes have 2500 images to make sure network is not biased to any one particular class of images. To add more data I rotated original images.

Here are samples of original and augmented






The code for splitting the data into training and validation sets is contained in the tenth code cell of the IPython notebook.
To cross validate my model, I randomly split the training data into a training set and validation set. I did this by using train_test_split from sklearn.model_selection, Kept 20% of data for validation out of complete training set.

My final training set has 86000 (2500 images_per_class * 43 classes * 0.8). My validation set and test set had 21500 images which is 20% from total training set.

Test is made of 12630 images, same as before, no augmentation is done to test images.

**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 eleventh cell of the ipython notebook.

My final model consisted of the following layers:

**Layer**
**Description**

Input
32x32x1 Grayscale image

Convolution1 - 5x5
1x1 stride, valid padding, outputs 28x28x6

RELU-1

Max pooling-1
2x2 stride, outputs 14x14x6

Convolution-2 5x5
1x1 stride, same padding, outputs 10x10x16

RELU-2

Max pooling-2
2x2 stride, outputs 5x5x16

Fully connected-1
Input =400 and Output 120

Fully connected-2
Input=120 and Output=84

Fully connected-3

Input=84 and 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 the 12th cell of the ipython notebook.

To train the model, I used an Adamoptimizer, batch size of 128, number of epochs=10 and learning rate of 0.001

**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 13th  cell of the Ipython notebook.

My final model results were:

- validation set accuracy = 0.971
- test set accuracy = 0.902
-

If a well known architecture was chosen:

- Used Lenet and found accuracy to be OK after normalizing the images

- Which parameters were tuned? How were they adjusted and why?
  - tuned learning rate to be 0.001. Having bigger learning rate was giving less accuracy
  - Weights has mu=0 and sigma=0.1

- Why did you believe it would be relevant to the traffic sign application?
  - Traffic sign identification in image has same meaning irrespective of where is present in the image. Lenet does a good job of identifying these.

- How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?
  - The accuracy was more than 97% on validation data and 90% on test set.

## 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.**

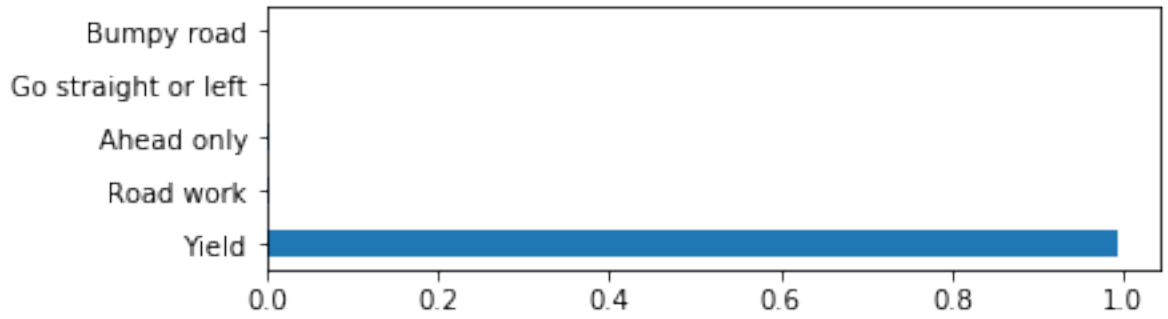
Here are five German traffic signs that I found on the web:



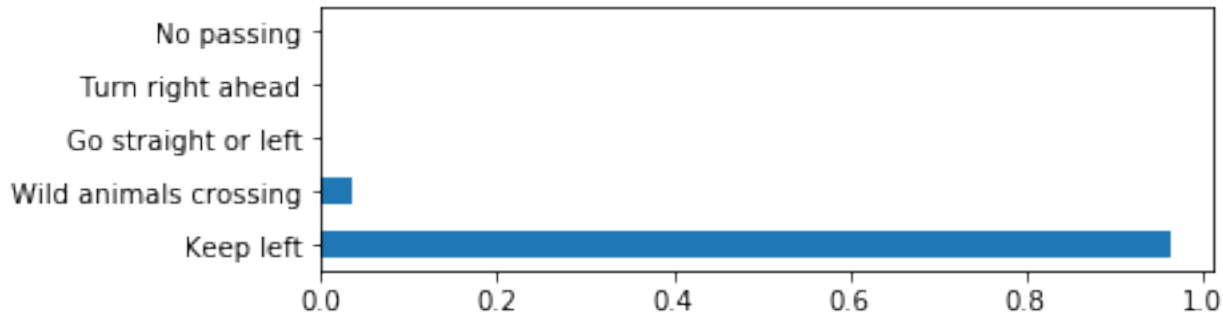
**2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. Identify where in your code predictions were made. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on 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 18th cell of the Ipython notebook.

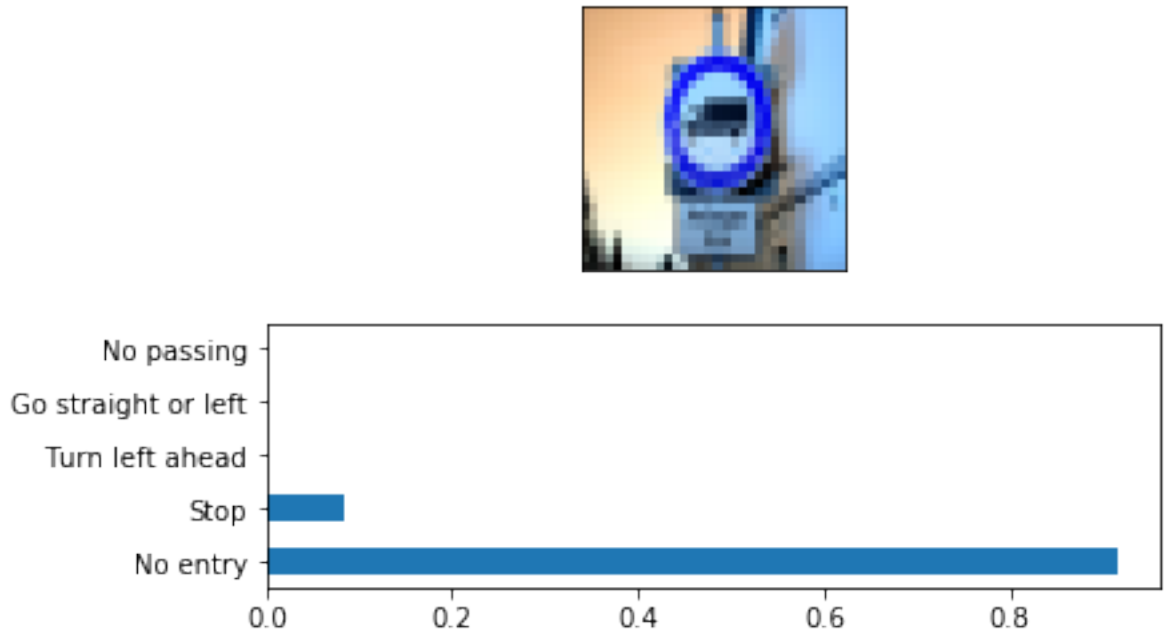
Here are the results of the prediction:



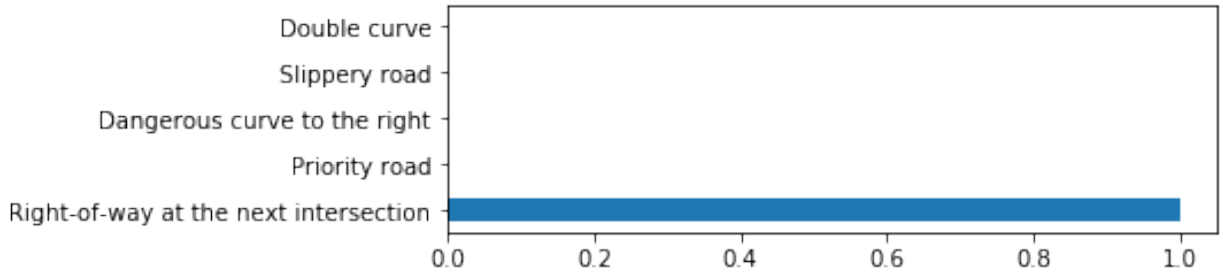

This image is two people walking with cellphone and not paying attention to cars.
This can be classified as pedestrian sign.
Model predicted yield which may or may not be OK.



This sign is for men at work. The model predicted is completely wrong which mean car with this model running may hit people at work on road.




This sign is for trucks. The model predicted is completely wrong which said no entry,  car with this model running may not be able to use this road.




These are multiple sign which includes, road work, speed limit 80 and slippery road. I wasn't expecting model to predict anything useful on this one, it still predicts right of way at next intersection with high probability. It is completely wrong classification.

**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 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

Already covered in question#2 above