# 2Traffic 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 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

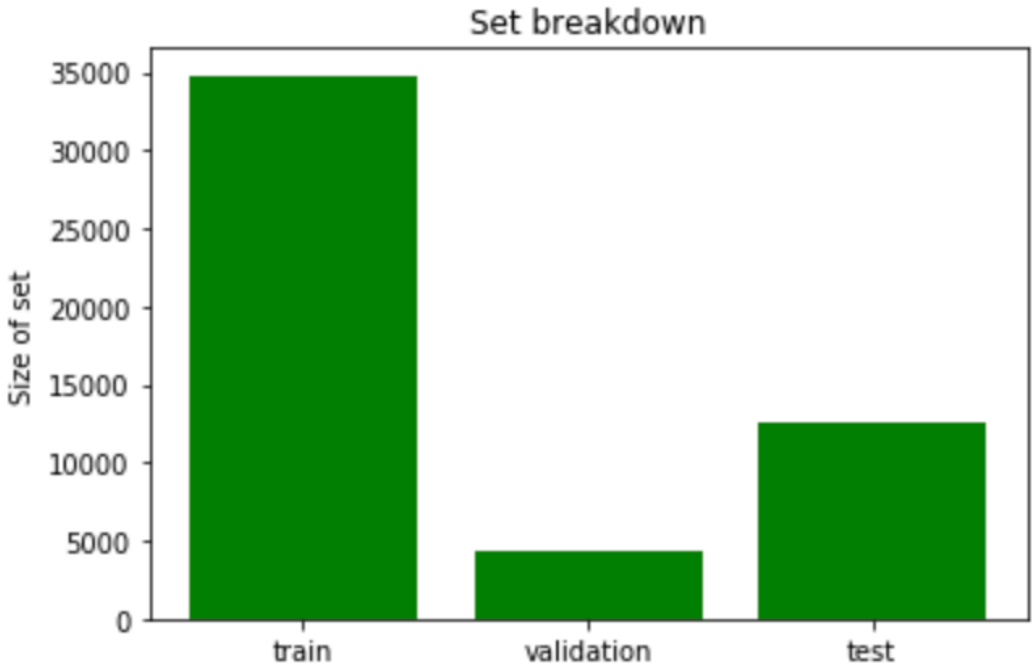
**1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.**

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

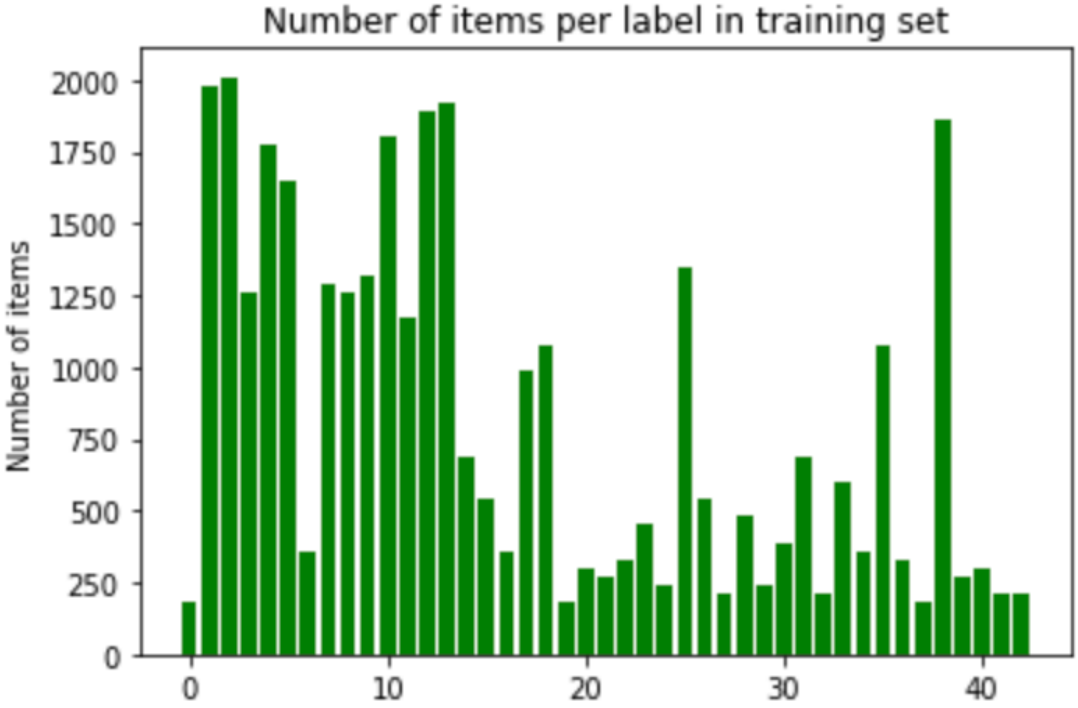
- The size of training set is: 34799
- The size of the validation set is: 12630
- The size of test set: 4410
- The shape of a traffic sign image is: 32,32 with three color channels
- The number of unique classes/labels in the data set is: 43

**2. Include an exploratory visualization of the dataset.**

The breakdown of the different sets seems reasonable.




On the other hand, the distribution within the testing set looks unreasonable. There are ten times more samples per speed limit sign then per keep left/right sign. If we aren't careful, the model will learn the following: when in doubt, guess speed sign. We'll come back to this issue later.

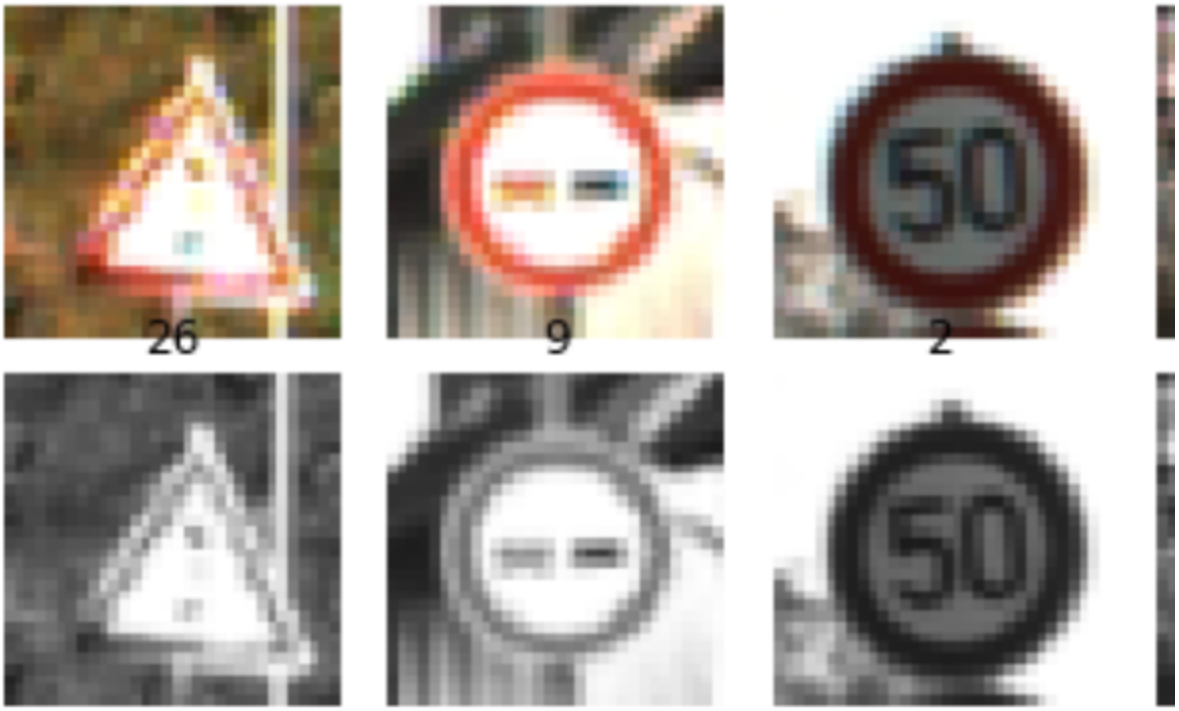



## Design and Test a Model Architecture

**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.)**

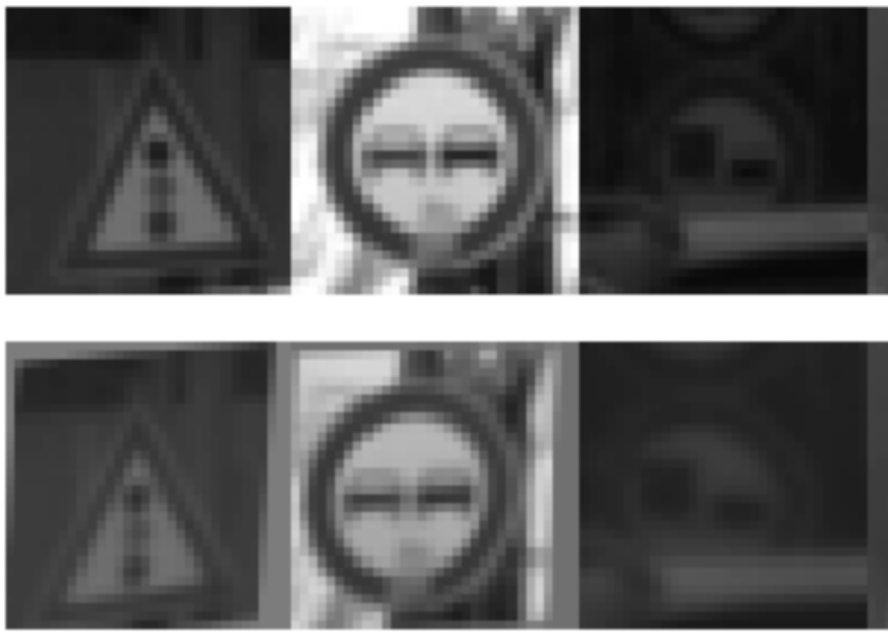
As a first step, I started by converting to grayscale. I believed the color might be useful. But figured I'd start with the exact same architecture as LeCun until I needed to change anything.

Next, I normalized the data so that all pixel values range from -1 to +1 since having a mean of 0 is best practice. I did this with a simple linear mapping from [0,255] to [-1,+1] so I didn't get the mean perfectly to 0. But close enough. See three images with before and after post processing.




Next, I decided that I wanted a more evently distributed training set. So I decided to augment the training set with mutated images. I performed the following mutations: affine scale, affine warp, affine translate and brightness. The goal of these changes: simulates retaking a photo of the same image with a slightly different exposure and different angle.

See three images before and after mutation.



**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, based closely on Le Cun's model:

| Layer | Description |
|---|---|
| Input | 32x32x1 Grayscale image |
| Convolution 5x5x6 | 1x1 stride, outputs 28x28x6 |
| RELU | Standard RELU |
| Max pooling | 2x2 window and 2x2 stride.<br>outputs 16x16x6 |
| Convolution 3x3x16 | 1x1 stride<br>Outputs 10x10x16. |
| RELU | |
| Max Pooling | 2x2 window and 2x2 stride.<br>Outputs 5x5x16 |
| Flatten | |
| Fully Connected | Input = 400. Output = 120 |
| RELU | |
| Dropout | During training, keep_prob=0.5 |
| Fully Connected | Input = 120. Output = 84 |
| RELU | |
| Dropout | During training, keep_prob=0.5 |
| Fully Connected | Input = 84. Output = 43 |

**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 AdamOptimizer with the following:

- Batch size of 128
- Epochs: up to 20
- Learning rate: 0.001

**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 0.994
- validation set accuracy of 0.961
- test set accuracy of 0.938

If an iterative approach was chosen:

- I first tried the LeNet architecture, since it was a simple well proven architecture that was proven capable of recognizing a similarly sized set of classes where each class was only a little simpler then traffic signs
- Initially, my results were terrible. After much debugging I noticed that my architecture had the wrong sized output layer. Once this was fixed, I obtained 0.85 validation accuracy
- Next, I tried pre-processing the images to fix contrast and brightness issues. This didn't help very much. It appeared the model learnt how to compensate for these issues on its own. So I removed this part of the pipeline.
- Next, I decided to address the imbalanced distribution in the training set with data augmentation. This immediately bumped my validation set accuracy to 0.91-0.92.
- Next I increased the number of EPOCHs. This got me to the target accuracy on the validation set.
- Next I tried adding dropout between the fully connected layers. This noticably increased the accuracy further.

## 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.**




Like in the training set, these images contain unobstructed pictures of german traffic signs. However, there are differences between these images and the images in the training set:

- Higher average brightness
- The 50KPH sign has no background
- Three of these images are taken from low angle, unlike most examples in the training set
- The resolution is different then the training set (but don't worry, we'll resize them so this problem goes away)

These differences will like reduce the ability of the model to correctly classify these images.

**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).**

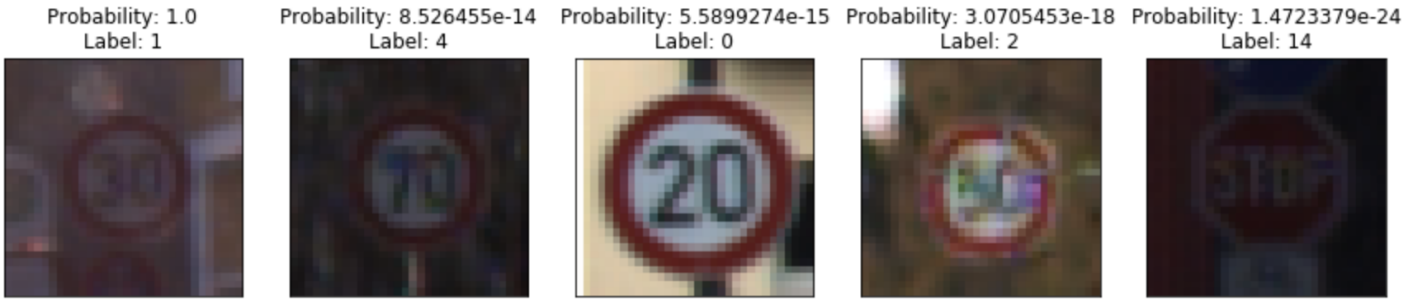
Here are the results of the prediction:

| Image | Prediction |
|---|---|
| 50kph | 50kph |
| Arterial | Stop |
| Yield | Yield |
| Right of way | Right of way |
| Stop | Turn Right |

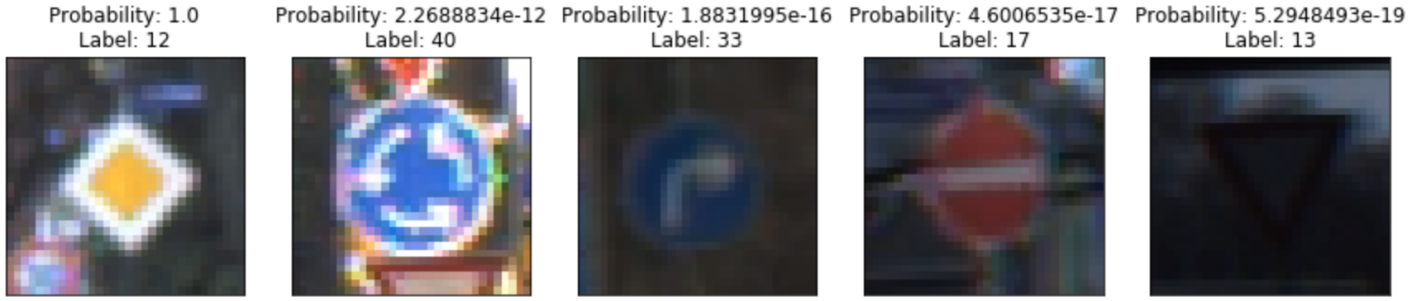
The model was able to correctly guess 4 of the 5 traffic signs, which gives an accuracy of 80%. This compares unfavorably to the accuracy on the test set. It is hard to draw conclusions with so few samples.

**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)**

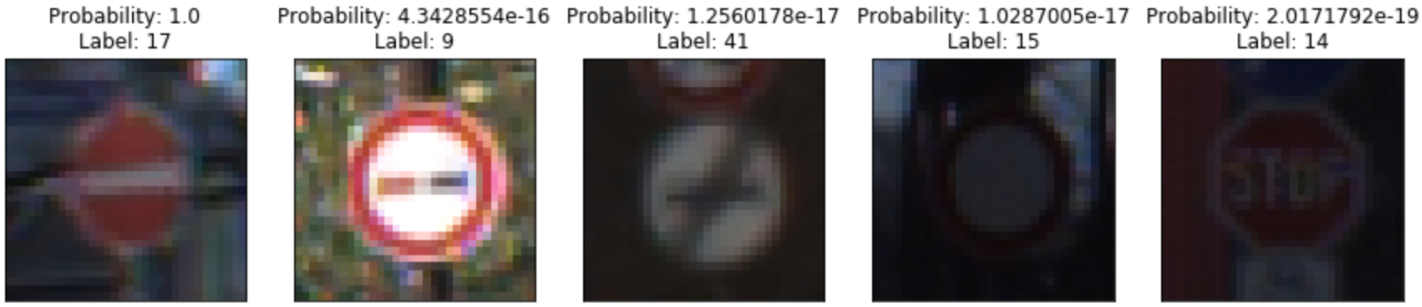
For the first image, the top five probabilities are the following. I've included sample images for the predicted class.



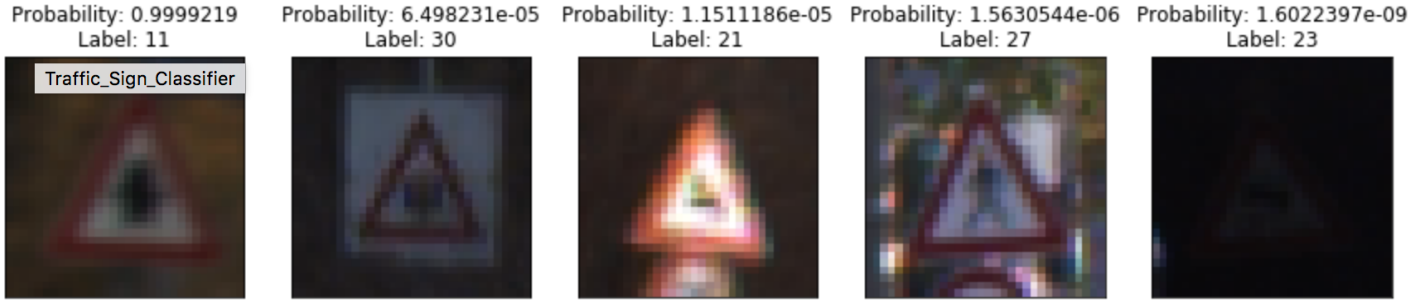

Second image:



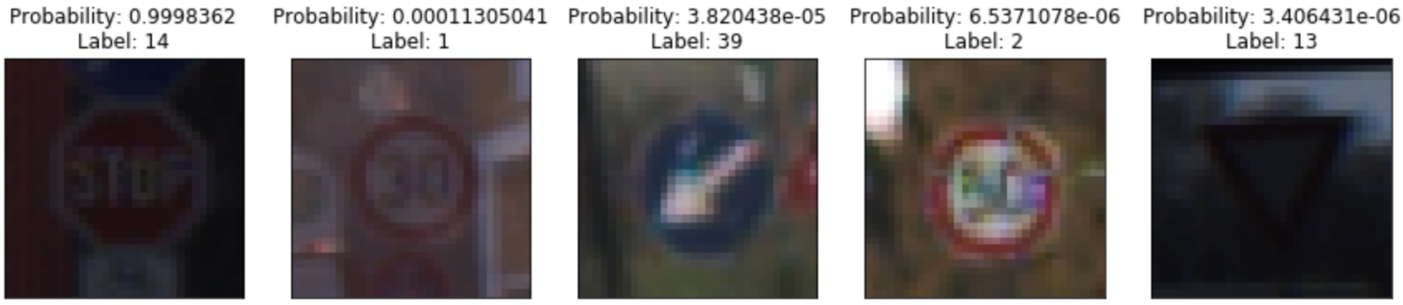

Third image:




Forth image:


Probability: 0.9999219
Label: 11
Traffic_Sign_Classifier
Probability: 6.498231e-05
Label: 30
Probability: 1.1511186e-05
Label: 21
Probability: 1.5630544e-06
Label: 27
Probability: 1.6022397e-09
Label: 23

Fifth image:


Probability: 0.9998362
Label: 14
Probability: 0.00011305041
Label: 1
Probability: 3.820438e-05
Label: 39
Probability: 6.5371078e-06
Label: 2
Probability: 3.406431e-06
Label: 13