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](https://review.udacity.com/#!/rubrics/481/view) individually and describe how I addressed each point in my implementation.

### **Writeup / README**

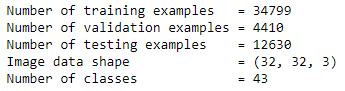
#### **1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. You can use this template as a guide for writing the report. The submission includes the project code.**

You're reading it!

### **Data Set Summary & Exploration**

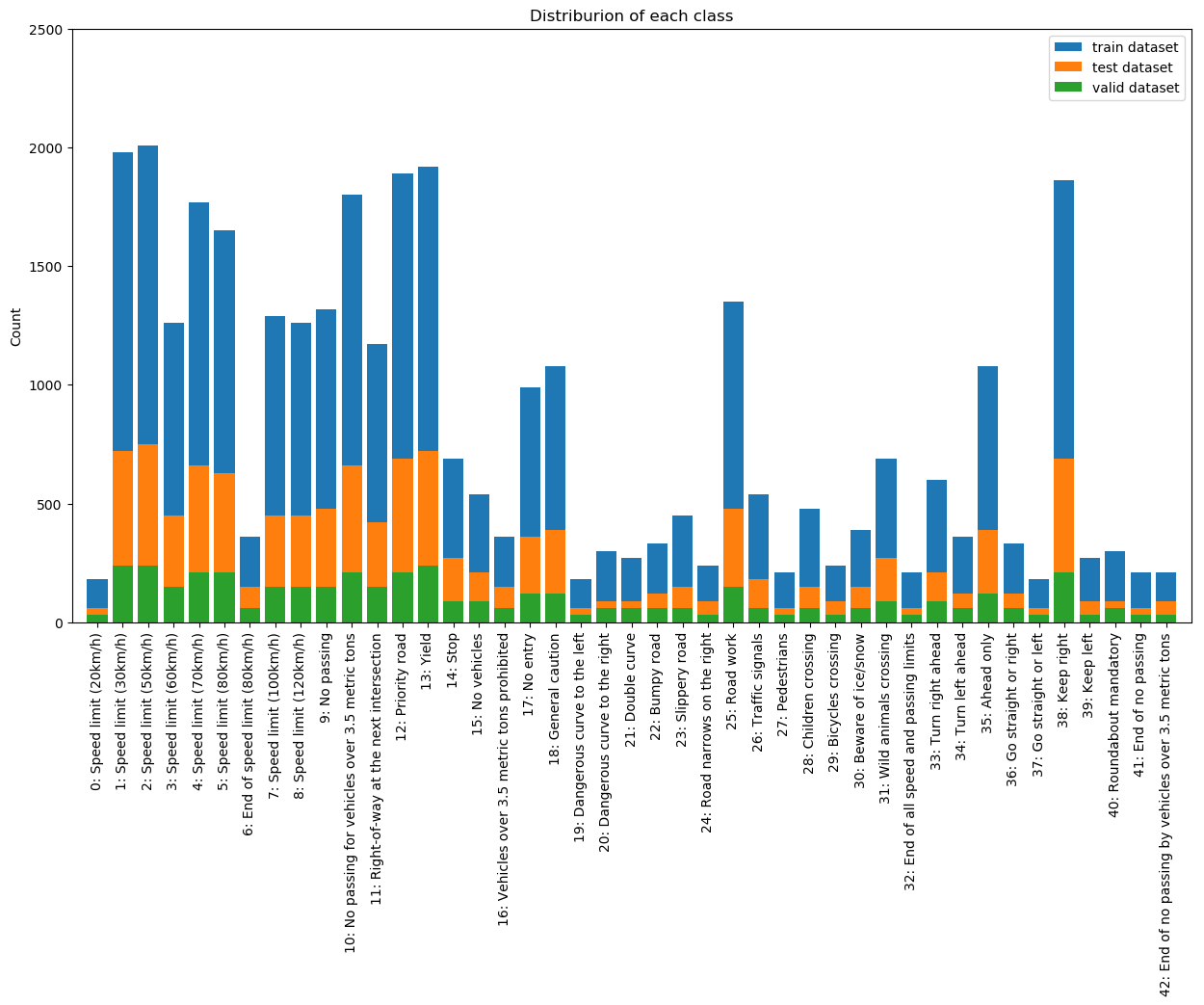
#### **1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.**

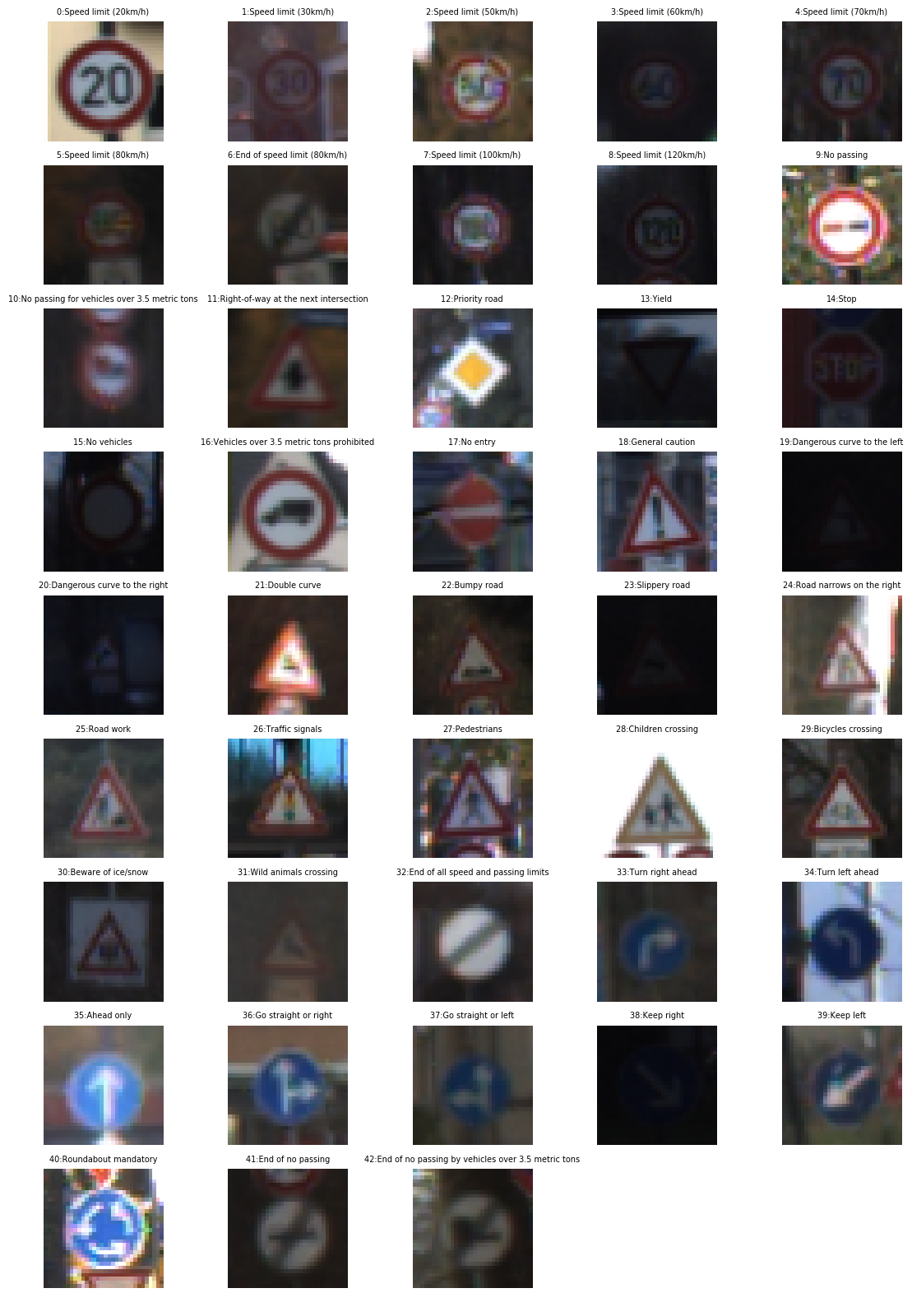
I used the pandas library to calculate summary statistics of the traffic signs data set:



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

Here is an exploratory visualization of the data set. It is a bar chart showing the distribution of each dataset with traffic sign class. It is clear that each dataset has all the classes.



In addition to that. I displayed an image from each class

### **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.)**

At first, I decided to increase the data set by creating a function to rotate the training dataset. The result is increase in performance but at the expenses of increase of training time

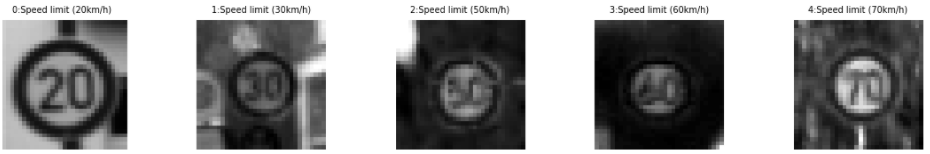
After that, I decided to convert the images to grayscale because it will reduce the number of channels in the image, this will lead to decrease the input size to the model and finally the training time will decrease. In addition, converting to grayscale because the model should detect the traffic sign based on the geometry and shape not by color.

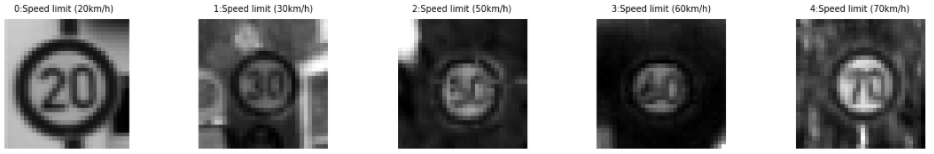
According to the article “[How to Manually Scale Image Pixel Data for Deep Learning](https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/)”; when using the raw image with pixels between 0 to 255, then the model will be very slow during training. That’s why we have to normalize the images’ pixels. I picked the algorithm “Standardize Pixel Values” where centering the pixel values on zero and normalizing the values by the standard deviation.

Finally, I shuffled the training data so that the model will not learn any order in the traffic signs.

While trying to improve the performance, I tried to make the image blur but the performance did not increase, so i decided not to include it in the pipeline.

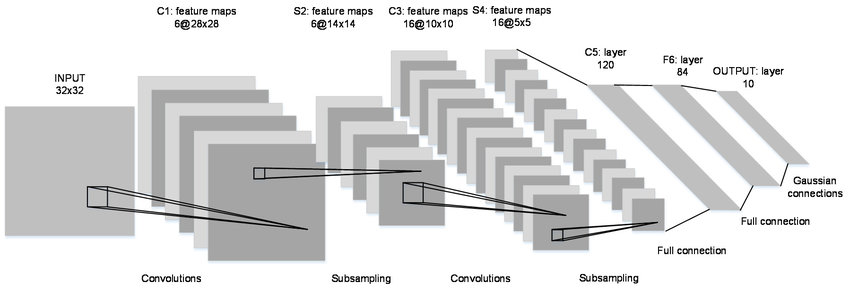






#### **2. Describe what your final model architecture looks like including model type, layers, layer sizes, connectivity, etc.) Consider including a diagram and/or table describing the final model.**

My final model is the LeNet. LeNet is a convolutional neural network structure proposed by Yann LeCun et al. in 1989. In general, LeNet refers to LeNet-5 and is a simple convolutional neural network. Convolutional neural networks are a kind of feed-forward neural network whose artificial neurons can respond to a part of the surrounding cells in the coverage range and perform well in large-scale image processing.



The network is the LeNet model used in our lesson. I modified the input parameter to 32x32x1 because the input is grayscale with one channel. The out layer is modified to 43 output because we have 43 traffic signs in our data set. The model is described as:

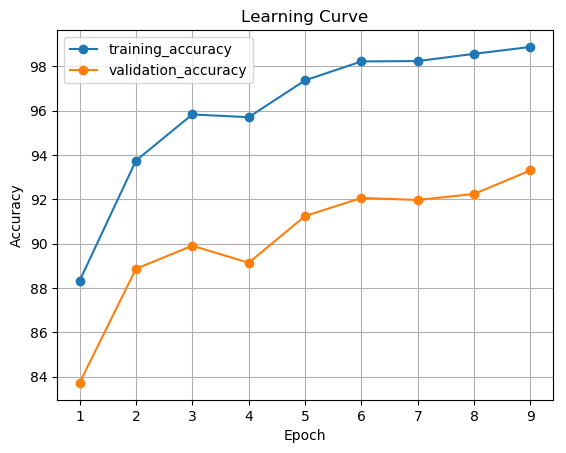
|  |  |
| --- | --- |
| Layer | Description |
| Input | 32x32x1 Gray image |
| Convolution\_1 5x5 | VALID padding, output 28x28x6 |
| RELU |  |
| Max pooling | VALID padding, output 14x14x6. |
| Convolution\_2 5x5 | VALID padding, output 10x10x16 |
| RELU |  |
| Max pooling | VALID padding, output 5x5x16. |
| Flatten | Output 400 |
| Fully connected\_1 | Output 120 |
| RELU |  |
| Fully connected\_2 | Output 84 |
| RELU |  |
| Fully connected\_3 | 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 parameters are:

* Epoch = 30
* Batch\_size = 128
* Learning rate = 0.001
* Optimizer = “AdamOptimizer”

While training the model, I put a stopping condition. The condition was that the model reach 93% validation accuracy.



#### **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.**

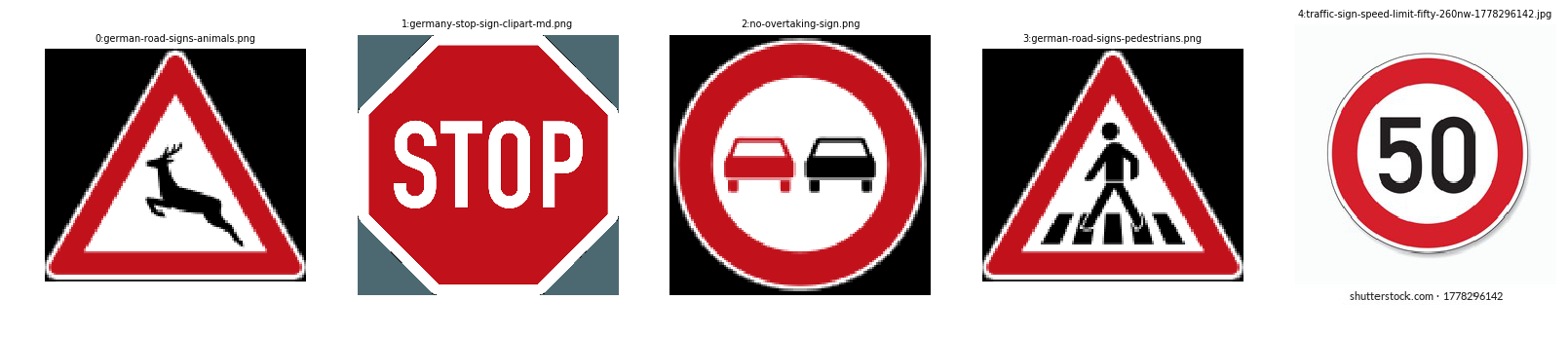
My final model results were:

* Training Accuracy = 98.88%
* Validation Accuracy = 93.31%
* Test Accuracy = 90.69%

### **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:



The main issue while working with them is that they are in png format, so they have a 4th channel and I had to remove it so that is will work correctly with the model

#### **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 |
| Wild animals crossing | Wild animals crossing |
| Stop | Stop |
| No passing | No passing |
| Pedestrians | Pedestrians |
| Speed Limit 50 | Speed Limit 50 |

The model was able to correctly guess 5 of the 5 traffic signs, which gives an accuracy of 100%.

#### **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)**

I made a bar chart to describe the probability of estimating the traffic sign for each picture.

