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**

Now I will be discussing each step of this project in detail

Load the data set and Explore, Summarize and Visualize the Data Set

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.

The code for the summary of the data set is given in the 2nd cell of the Ipython Notebook called Traffic_Sign_Classifier.ipynb.

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

* The size of training set is 34799 samples

* The size of the validation set is 4410 samples

* The size of test set is 12630 samples

* 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.

The code for the visualization of the data set is given in the 3rd cell of the Ipython Notebook called Traffic_Sign_Classifier.ipynb.

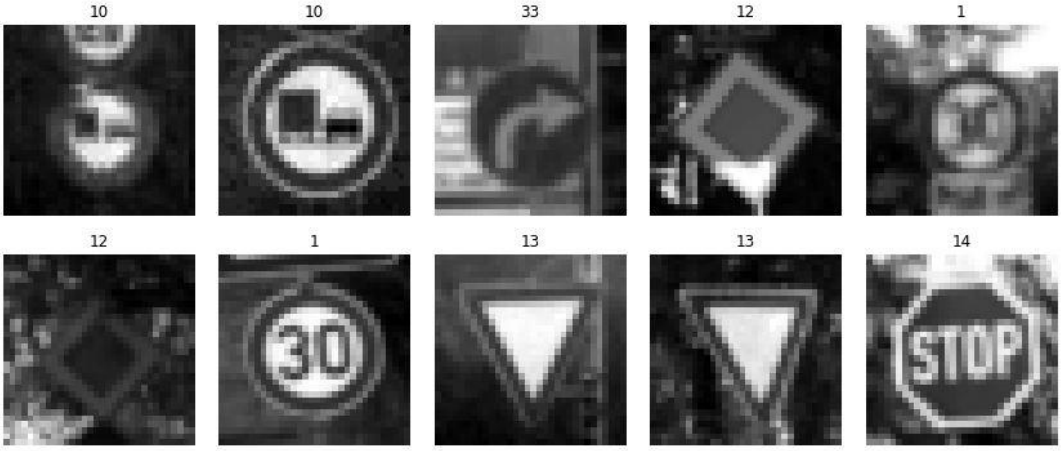
Here is an exploratory visualization of the data set. I have used the matplotlib library to plot these data samples randomly from the data set by using the random library as well.

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.

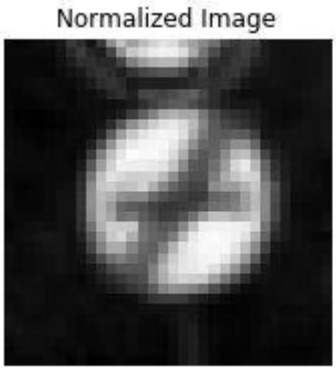
Code cells 4 – 6 contain the conversion and visualization part for the grayscale conversion.

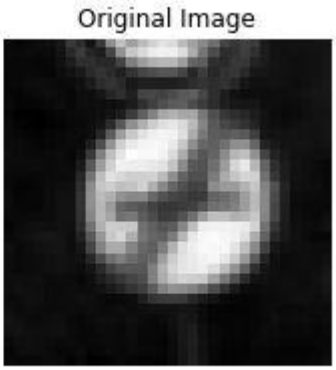
I decided to convert the images to grayscale because the information about the colours is not so relevant here and graysacale images are faster to process also. Some examples of my grayscale images are given below :

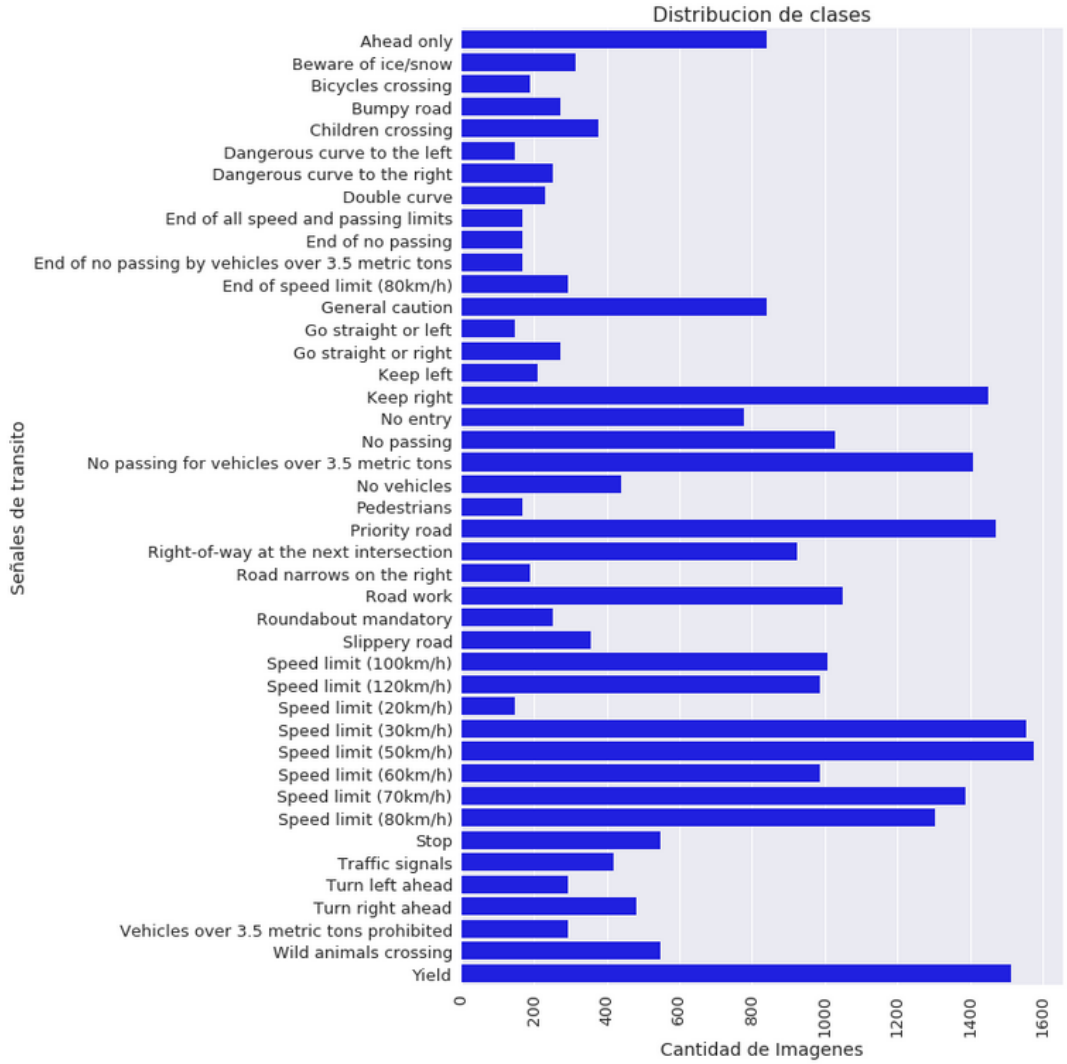


Code cells 7 – 9 is for normalizing the images.

After the grayscaling, I normalized the image data because it helps it helps a lot to speed up the training process and save lots of memory resources as well. Here is a sample of Original and Normalized image :







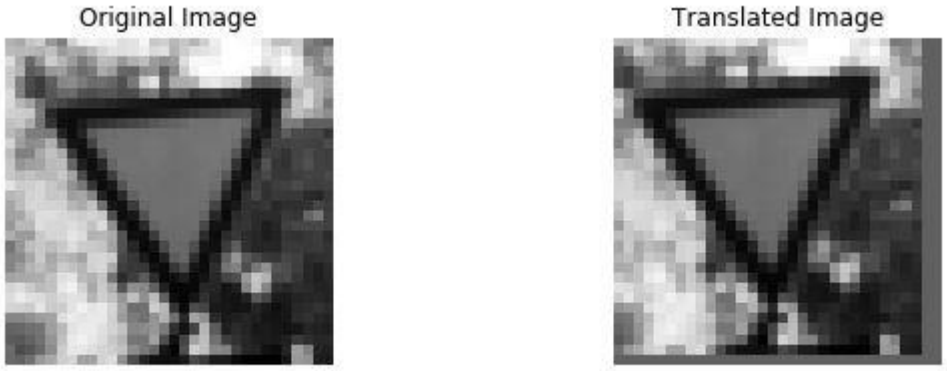
Now, from the above image, we can clearly observe the distribution of the data and we can clearly see that the data is highly imbalanced, which means that some signs have more samples and some signs have less samples, so if we train our neural network with this data, it is going to be much more biased towards the data points which are large in number.

Due to this data imbalance, I decided to increase the number of data point up to 800 samples for the labels which are very low on number with the help of Data Augmentation technique which is very effective for deep learning based tasks and increases the data set size a lot.

To add more data to the data set, I used the techniques such as Translating – moving the image by some pixels in x and y direction, Scaling – increasing the size of the image slightly, Warping the images and Brightness Adjustment – adjusting the brightness of the image

Code cell 10 is for Translation.

An example of a Translated Image is given below :

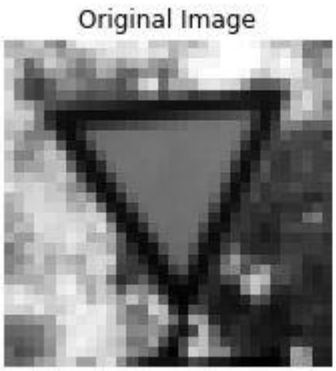


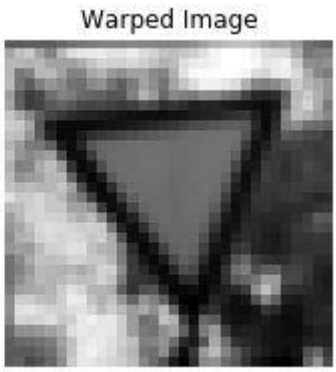
Code cell 11 is for Scaling.

An example of Scaled Image is given below :

Code cell 12 is for Warping the images

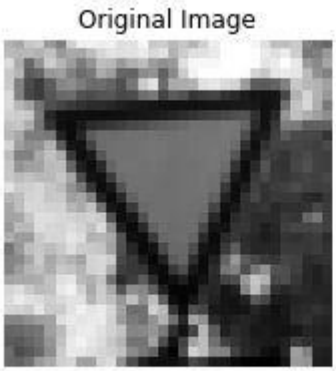
An example of a Warped Image is given below :

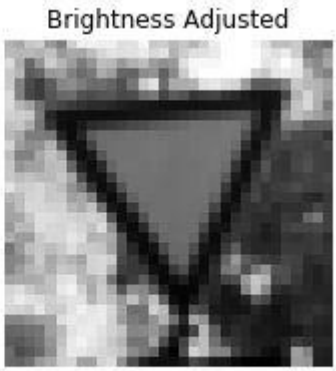




Code cell 13 is for Brightness Adjustment of the images.

An example of Brightness Adjusted image is given below :





In the code cell 14 and 15, I apply the Data Augmentation techniques described above into my training set to generate appropriate samples.

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.

Code cell 21 describes the LeNet model architecture which I have used here to train the traffic signs images.

My final model consisted of the following layers:

Layer	Description
Input	32x32x1 Grayscale image

Convolution 3x3	1x1 stride, Valid padding, Output - 28x28x6
RELU	
Max Pooling	2x2 stride, Output - 14x14x6
Convolution 3x3	1x1 stride, Valid padding, Output – 10x10x16
RELU	
Max Pooling	2x2 stride, Output – 5x5x16
Convolution 3x3	Output – 1x1x400
RELU	
Flatten	Input – 5x5x16, Output - 400
Flatten	Input – 1x1x400, Ouput - 400
Drouput	50%
Fully Connected	Input – 800, 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.

Code cells 23 and 24 are for training and evaluating the model.

To train the model, I used the LeNet model, where I opted for 60 epochs and a batch size of 100. I used the Adam optimizer with a learning rate of 0.0009 to train my classifier as it is much more robust and greatly modified version of the Stochastic Gradient Descent. With this configuration, I was able to get accuracy around 94% in the test set.

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 1.000

* validation set accuracy of 0.990

* test set accuracy of 0.937

If a well known architecture was chosen:

* What architecture was chosen?

I chose the LeNet architecture to classify the traffic signs.

* Why did you believe it would be relevant to the traffic sign application?

I believe this model is relevant for traffic signs classification because this model uses the convolutional neural networks which are a very powerful architecture to capture the essence of the traffic signs and it’s spatial characteristics in detail.

* How does the final model's accuracy on the training, validation and test set provide evidence that the model is working well?

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:

These image might be difficult to classify because these do not have any borders around the edges of the image whereas the images in our training set has borders around the images.

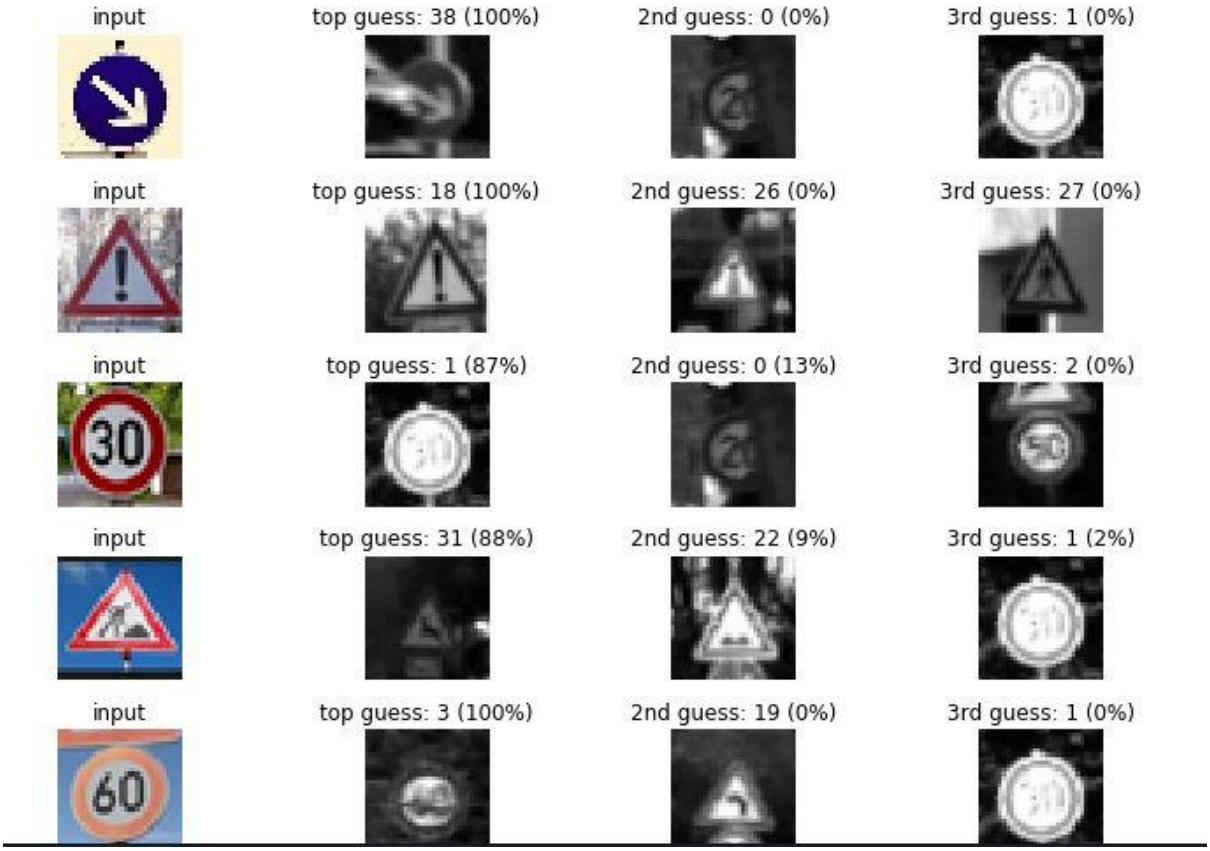
2. Discuss the model's predictions on these new traffic signs and compare the results to predicting on the test set. At a minimum, discuss what the predictions were, the accuracy on these new predictions, and compare the accuracy to the accuracy on the test set (OPTIONAL)

The model was able to correctly guess 7 of the 8 traffic signs, as shown below in the image given in the next point.

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)

The code for making predictions on my final model is located in the 30th cell of the Ipython notebook.

Here is the Output given below :



input	top guess: 12 (100%)	2nd guess: 40 (0%)	3rd guess: 9 (0%)
			
input	top guess: 11 (100%)	2nd guess: 30 (0%)	3rd guess: 21 (0%)
			
input	top guess: 34 (100%)	2nd guess: 38 (0%)	3rd guess: 35 (0%)