Traffic Sign Recognition

Write-up

Build a Traffic Sign Recognition Project

The goals / steps of this project are the following:

- Load the 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

Data Set Summary & Exploration

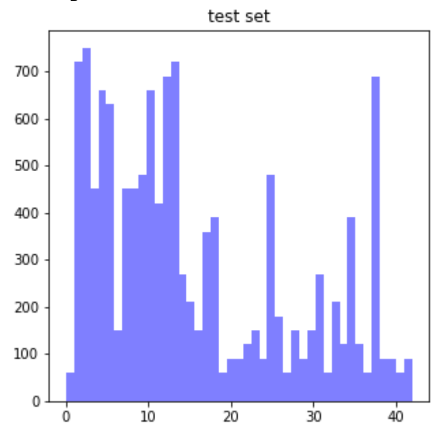
1. Provide a basic summary of the data set. In the code, the analysis should be done using python, numpy and/or pandas methods rather than hardcoding results manually.

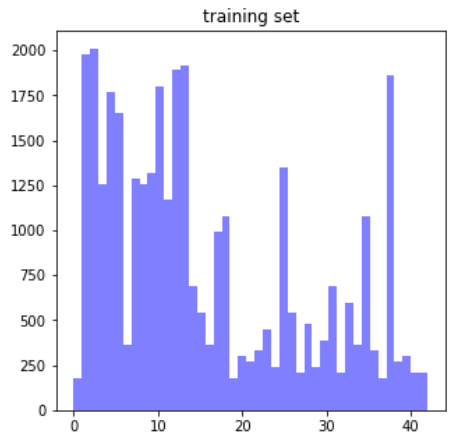
I used the pandas library to calculate summary statistics of the traffic signs data set:

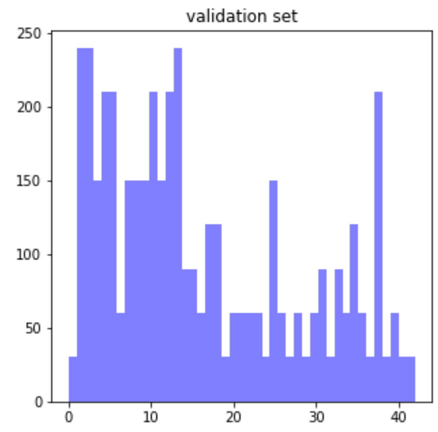
- The size of training set is 34799
- The size of the validation set is 4410
- The size of test set is 12630
- The shape of a traffic sign image is (32,32,3)
- The number of unique classes/labels in the data set is 43

2. Include an exploratory visualization of the dataset.

Here is an exploratory visualization of the data set. It contains 3 histograms showing distribution of classes. Y-axis is number of examples and x-axis is class of images







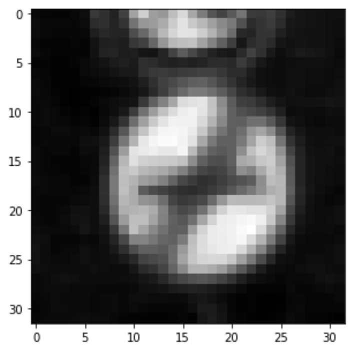
The number of images in training set is greater than validation and test set, but distribution as per classes is quite symmetrical

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.

- As a first step, I decided to convert the images to grayscale because they take lesser number of bits to represent and make our calculations fast. Also color has no usage to identify the features in the image.

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



- Then I normalized the image data to make the scale of all the features similar. This helps in the learning process. To normalize data I subtracted 128 from each pixel value and then divided it by 128.
- To make the data compatible with the lenet function, I added a new axis to the images. Image shape was converted from 32x32 to 32x32x1.

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:

Layer	**Description**
Input	32x32x1 Gray image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
RELU	
Max pooling	2x2 stride, valid padding, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
RELU	
Max pooling	2x2 stride, valid padding, outputs 5x5x16
Fully Connected	Input is flattened, output is n_examples x 120
RELU	
Dropout	35% values dropped to prevent over-fitting
Fully Connected	Input is flattened, output is n_examples x 84
RELU	
Dropout	35% values dropped to prevent over-fitting
Fully Connected	Input is flattened, output is n_examples x 43 (logits)

3. Describe how you trained your model. The discussion can include the type of optimizer, the batch size, number of epochs and any hyper parameters such as learning rate.

To train the model, I used Adam optimizer, batch size of 128, number of epochs equal to 10 and learning rate of 0.001. To prevent over-fitting, I used L2 regularisation technique and added convolution1 and convolution2 weights l2 loss to the operation loss. This will put restriction on conv1 conv2 weight to take high values. Another technique used to get the desired results was dropout.

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:

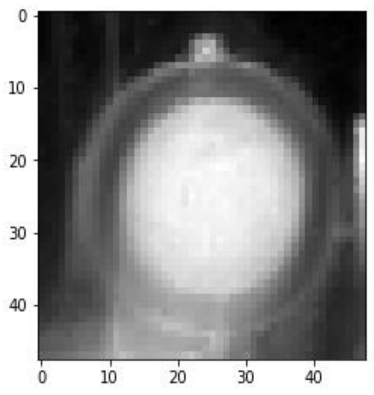
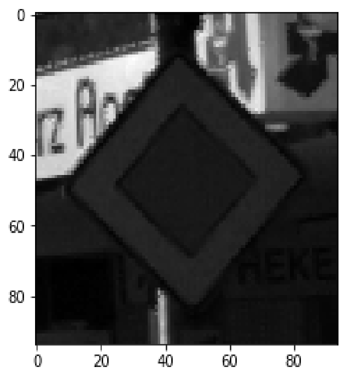
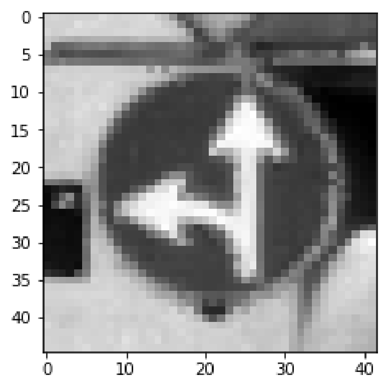
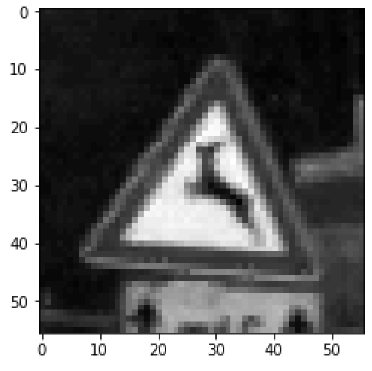
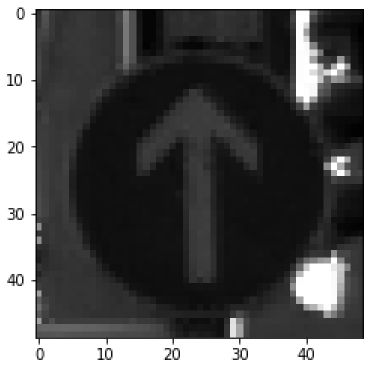
- training set accuracy of **97.4%**
- validation set accuracy of **93.3%**
- test set accuracy of **100%**

To get 93% validation accuracy Lenet architecture was used. But using this architecture only gave 89% validation accuracy. This was because of over-fitting of the model. To prevent over-fitting two methods were used, dropout and l2 regularisation. In dropout 35% of the values obtained after 1st and 2nd full connected layers where dropped. And in L2 regularisation, weights of conv1 and conv2 layers were added to cost function. This put restriction on weights to take very high value. These measures prevented over-fitting and hence I got the desired accuracy.

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 converted images to gray so that they are easy to classify and takes lesser memory. Images are difficult to classify since they are not very clear especially the first image. But my model classified all of them perfectly as can be seen in my jupyter notebook, hence giving 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.

Image	**Prediction**
15(No vehicles)	15
12 (Priority road)	12
37(Go straight of left)	37
31(Wild animals crossing)	31
35 (Ahead only)	35

Model predicted all the images correctly giving accuracy of 100%

Top 5 softmax probabilities of each image

Image 1 top 5 softmax pr	**Image Index**
0.993	15
0	2
0	3
0	12
0	13

Image 2 top 5 softmax pr	**Image Index**
0.956	12
0	13
0	40
0	15
0	9

Image 3 top 5 softmax pr	**Image Index**
0.987	37
0	40
0	39
0	33
0	4

Image 4 top 5 softmax pr	**Image Index**
0.793	31
0.134	21
0	24
0	2
0	29

Image 5 top 5 softmax pr	**Image Index**
0.992	35
0	3
0	12
0	13
0	9