Traffic Sign Recognition

Writeup

Data Set Summary & Exploration

1. Basic summary of the data set.

I used the numpy, 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.

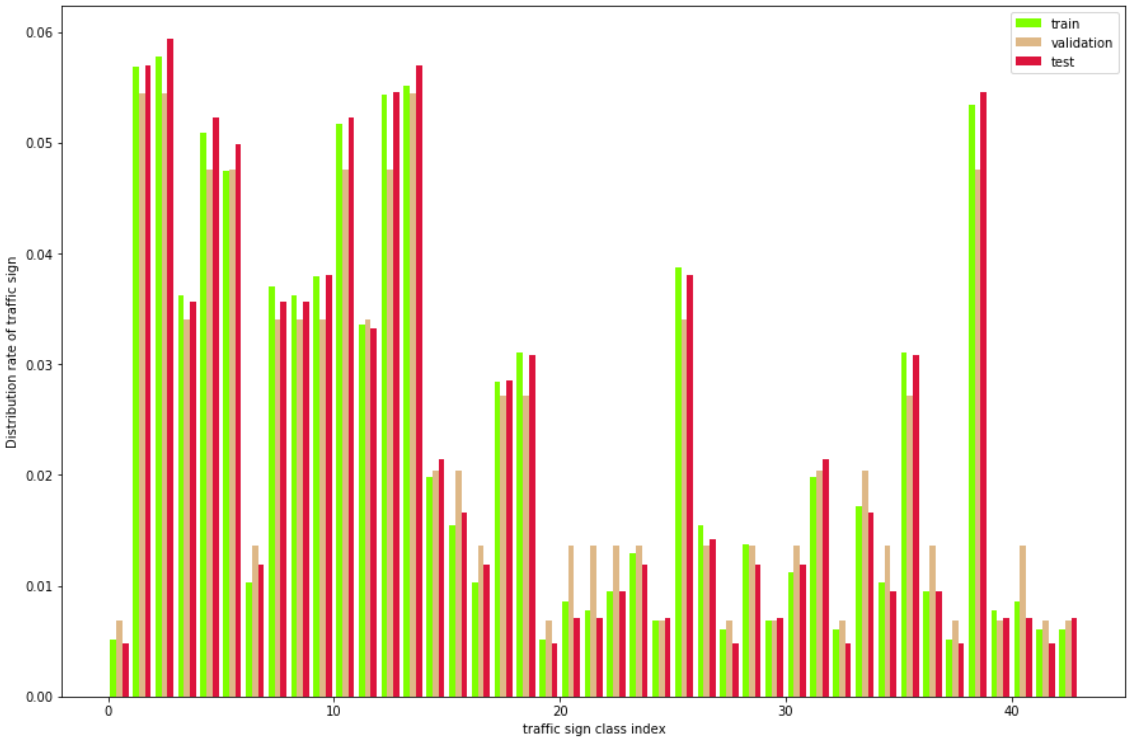
The shape of a traffic sign image is 32x32x3.

The number of unique classes/labels in the data set is 43.

2. An exploratory visualization of the dataset.

An example of traffic sign image:

Here is a visualization of the data set. It is a bar chart showing how the classes are distributed in the train, validation and test sets:



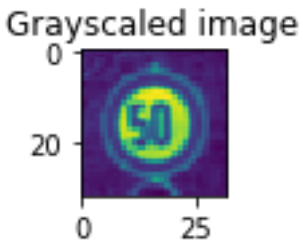
It seems that the distributions of classes in the train, test and validation sets are mostly equal with regard to each particular class. But there are big differences between distribution rates of classes. And that makes sense. Some signs are more common. For example the rate of #38(keep right) is about 8 times higher than the rate of #39(keep left). Most probably, the reason is that the traffic in Germany is right-hand.

Design and Test a Model Architecture

1. preprocessing the image data.

As a first step, I decided to convert the images to grayscale because it's easier for classifier to distinguish edges of signs in grayscale. Additionally, since with grayscale there is only one channel to work with, the computation becomes less complicated.

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



As a last step, I normalized the image data because we want the neural network to perform backpropagating in better conditions.

I decided to generate additional data in order make the classifier more robust and efficient. Also additional data can help prevent overfitting.

To add more data to the data set, I used the following techniques:

- rotating (the angle is random from (-15, 15))
- moving alongside X and Y (the distance to move is a random from (-2,2) pixels)
- zooming (the scale is a random from(0.9,1.1))

Additional data should be altered from the origin data randomly, so the network could learn something new. However, to be helpful, new data should looks like it was gathered in the same manner as the original. I added 3 instances of additional altered data. each instance is just an altered copy of the original data, so the distribution of classes doesn't change. That data makes the network to be more precise (about 1 percent better for test set). More augmented data leads to underfitting and makes training to long, even for gpu.

Here is an example of an original image and an augmented image:

2. Final model.

My final model consisted of the following layers:

Layer	Description

Input	32x32x1 Grayscale image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x50
RELU	
Max pooling	2x2 stride, ouputs 14x14x50
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x100
RELU	
Max pooling	2x2 stride, ouputs 5x5x100
Fully connected	2500x500
Fully connected	500x120
Fully connected	120x43
Softmax	

.

3. Training the model.

To train the model, I used AdamOptimizer. Since, AdamOptimizer control learning rates by using moving averages of the parameter, it doesn't require more hyperparameter tuning like gradient descent. It is useful.

I started with learning rate = 0.001 Epoch = 10 and batch size = 128. Eventually learning rate was increased to 0.003 which didn't harm performance on validation set, but make the network train faster. Once network architecture is determined the batch size was increased, but that didn't significantly change the performance on validation set, so the final batch size is 128. During the train only the best epoch (with respect to accuracy on the validation set) is saved.

4. Approach taken for finding a solution and getting the validation set accuracy to be at least 0.93.

My final model results were:

training set accuracy of 0.964

validation set accuracy of 0.965

test set accuracy of 0.953

These results were calculated just after the training, in the cell "Calculate and report accuracy"

Typical neural netoworks designed for traffic sign recognition like networks mentioned in http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf (the link provided in the initial Jupyter notebook for the project) consist of 2 convolution layers+subsampling and classifier (with one or more layers).

Since LeNet also has 2 convolution layers + classifier and validation accuracy was 0.89 which is not so far from the target 0.93, I started with LeNet. It was designed for number recognition. There was understanding that the convolutions should have more filters in order to deal with traffic signs which seemed to be more complicated to recognize. Step by step it comes to 50-100 (first and second convolutions respectively) instead of 6-16. That improves the rate on validation set by 1.5-2 in average. At that moment there was another obvious flaw - over fitting. To tackle that I used dropout. During experiments with keep_prob rate (from 0.5 to 0.8) it appeared that 0.6 is the best tradeoff. Additionally, L2 regularization was added to network (tf.nn.l2_loss tensor in tensor flow). L2 improves the average result by 1-1.5 percent which is slight improvement. It is worth to mention, that introducing additional train data also improves the network in terms of preventing overfitting. At the end the difference between the validation rate and the train rate is quite small.

Test a Model on New Images

1. Five German traffic signs found on the web.

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

The quality of the new signs is well enough. Although the last one is cut in the top and the third was probably taken from the angle. The rest three should not make difficulties for classifier.

2. The model's predictions on these new traffic signs.

Here are the results of the prediction:

Image	Prediction
Speed limit (30km/h)	Speed limit (30km/h)
Right-of-way at the next intersection	Right-of-way at the next intersection
Pedestrians	Pedestrians
Priority road	Priority road

Roundabout mandatory	Roundabout mandatory

The model performance on new 5 signs is 100% which is better than for the train set (95.3%).

3. Top 5 softmax probabilities for each image along with the sign type of each probability.

The code for making predictions on my final model is located in the cell below the cell “Output Top 5 Softmax Probabilities For Each Image Found on the Web” 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

1. For the first image the model is certain. Interestingly, the rest 4 softmax probabilities belongs to other speed limit signs, which make sence.

Probability	**Prediction**
99.9	1,Speed limit (30km/h)
0	2,Speed limit (50km/h)
0	0,Speed limit (20km/h)
0	4,Speed limit (70km/h)
0	5,Speed limit (80km/h)

2. For the second one the model is also certain, and 3 next softmax probabilities belongs to the signs with the similar shape (triangles)

Probability	**Prediction**
99.9	11,Right-of-way at the next intersection
0	30,Beware of ice/snow
0	21,Double curve
0	27,Pedestrians

0	12,Priority road

3. As it was predicted third one was not detected as certainly as two above. Only 94.3. 4 others go to similar signs with the same shape (triangles)

Probability	Prediction
94.3	27,Pedestrians
2.07	24,Road narrows on the right
1.6	18,General caution
1.2	11,Right-of-way at the next intersection
0.47	21,Double curve

4. This one is predicted quite well. However, if the model is not well trained against additional data, sometimes the second softmax probability(Roundabout mandatory) would appear on the first row. That's interesting, why this two signs can be perceived as they are similar. Probably, the sign roundabout mandatory after converting to 32X32 becomes elongated. Additionally, these two signs in grayscale have some similarities.

Probability	Prediction
99.9	12,Priority road
0	40,Roundabout mandatory
0	38,Keep right
0	17,No entry
0	14,Stop

5. The last one also is not as certaint as 1,2 and 4, and that was predicted above.

Probability	Prediction

94.8	40,Roundabout mandatory
3.6	7,Speed limit (100km/h)
0.8	12,Priority road
0.2	16,Vehicles over 3.5 metric tons prohibited
0.2	17,No entry