Traffic Sign Recognition

Rubric Points

Here I consider the rubric points individually and describe how I addressed each point in my implementation.

Submission files

in the ZIP-file, next to this Writeup, you can find:

- Ipython notebook with code: *Traffic_Sign_Classifier.ipynb*
- HTML output of the code: *Traffic_Sign_Classifier.html*

Data Set Summary & Exploration

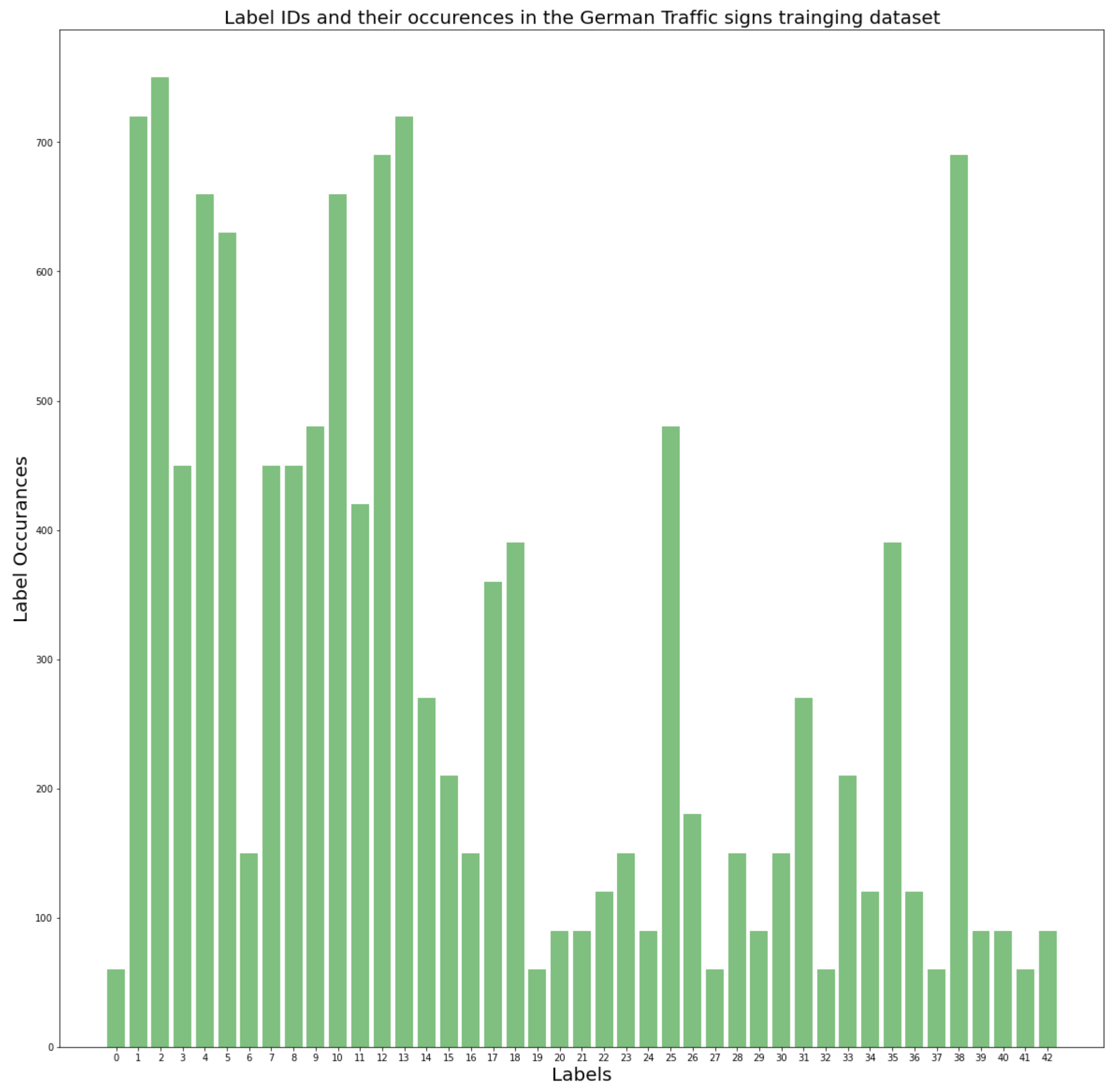
1. Basic summary of the data set.

I used the standard python methods to calculate summary statistics of the traffic signs data set, since amount of labels == amount of images:

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

2. Visualization of the dataset.

Here is an exploratory visualization of the data set. It is a bar chart showing the traffic sings distribution (IDs in the X-Axis) over the whole training dataset (amount of images per each traffic sign in y-Axis)



Decoding of the Traffic Sign IDs can be found in *signnames.csv*.

See visualization of the pre-processed examples of the traffic signs in section Test a Model on New Images

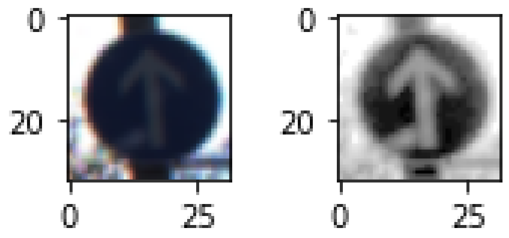
Design and Test a Model Architecture

1. Preprocessing

During preprocessing:

1. I converted the images to grayscale because:
    - it reduces the size of the image by 66%
    - reduces noise in the image
    - abstracts away unimportant details of the image.
2. I have equalized the histogram for contrast adjustment in order to avoid too high/low values without any loss of information.
3. After changing the image to Grayscale, the size is reduced to 32x32. Thus I've extended dimension by the color channel to 32x32x1, since a 2D array required for 2D convolutions by Tensorflow.
4. I applied normalization (histogram stretching with zero as mean) to change the range of pixel intensity values.

Here is an example of a traffic sign image before (left) and after (right) preprocessing:



I didn't generate any additional data though and didn't add more data to the the data set, since with the existing model the recognition rate of 96% has been achieved.

2. Model architecture

My final model consisted of the following layers:

Layer	Description
Input	32x32x3 RGB image
Preprocessing	32x32x3 -> 32x32x1 Grayscale image
Convolution 5x5 filter	1x1 stride, valid padding, outputs 28x28x6
ReLU	activation function
Max pooling	2x2 stride, 2x2 kernel, valid padding, outputs 14x14x6
Convolution 5x5 filter	1x1 stride, valid padding, outputs 10x10x16
ReLU	activation function
Max pooling	2x2 stride, 2x2 kernel, valid padding, outputs 5x5x16
Flattening	Input = 5x5x16. Output 1D array with 400 entries.
Fully connected	With shape change 400 -> 120, mu = 0 and sigma = 0.1
ReLU	activation function
Dropout	Dropout value of 0.5 is used for training and 1.0 for validation/testing
Fully connected	With shape change 120 -> 84, mu = 0 and sigma = 0.1
ReLU	activation function
Dropout	Dropout value of 0.5 is used for training and 1.0 for validation/testing
Output	Fully connected with shape change 84 -> 43, mu = 0 and sigma = 0.1

3. Model Training.

Layer	Description
Softmax cross entropy	Measure how different are the logits from the one-hot encoded ground-truth
Reduce mean	Averaging cross entropy over all training images
Optimizer	Adam optimizer is used, which functions similarly to stochastic gradient descent

For model training I have set following hyperparameters:

- learning rate = 0.001
- number of epochs = 20
- batch size = 128

4. Model Evaluation.

Layer	Description
Equal	Compare label prediction with one-hot encoded expected label
Reduce mean	Averaging individual predictions

My model is based on the LeNet-Lab solution.

This model has been chosen since it is an established and proven-in-use robust skeleton for solving such classification problems. I have tuned it with:

- image preprocessing (grayscale, normalization)
- prevention of underfitting/overfitting by introducing two dropouts after fully connected hidden layers.

The values for hyperparameters, architecture and as a consequence given results were achieved by the understanding of cause-effect relationship of the layers and trial-and-error.

My final model results were:

- validation set accuracy: 96.1% (97% is possible)
- test set accuracy: 93.5%

The evidence for the fact, that the given model performs well is that it continuously to 93.5-96% correctly classifies the traffic signs, despite:

- their density in the image
- light conditions
- different capture angles
- different image quality
- images, never seen by the model before

Test a Model on 9 images of German traffic signs

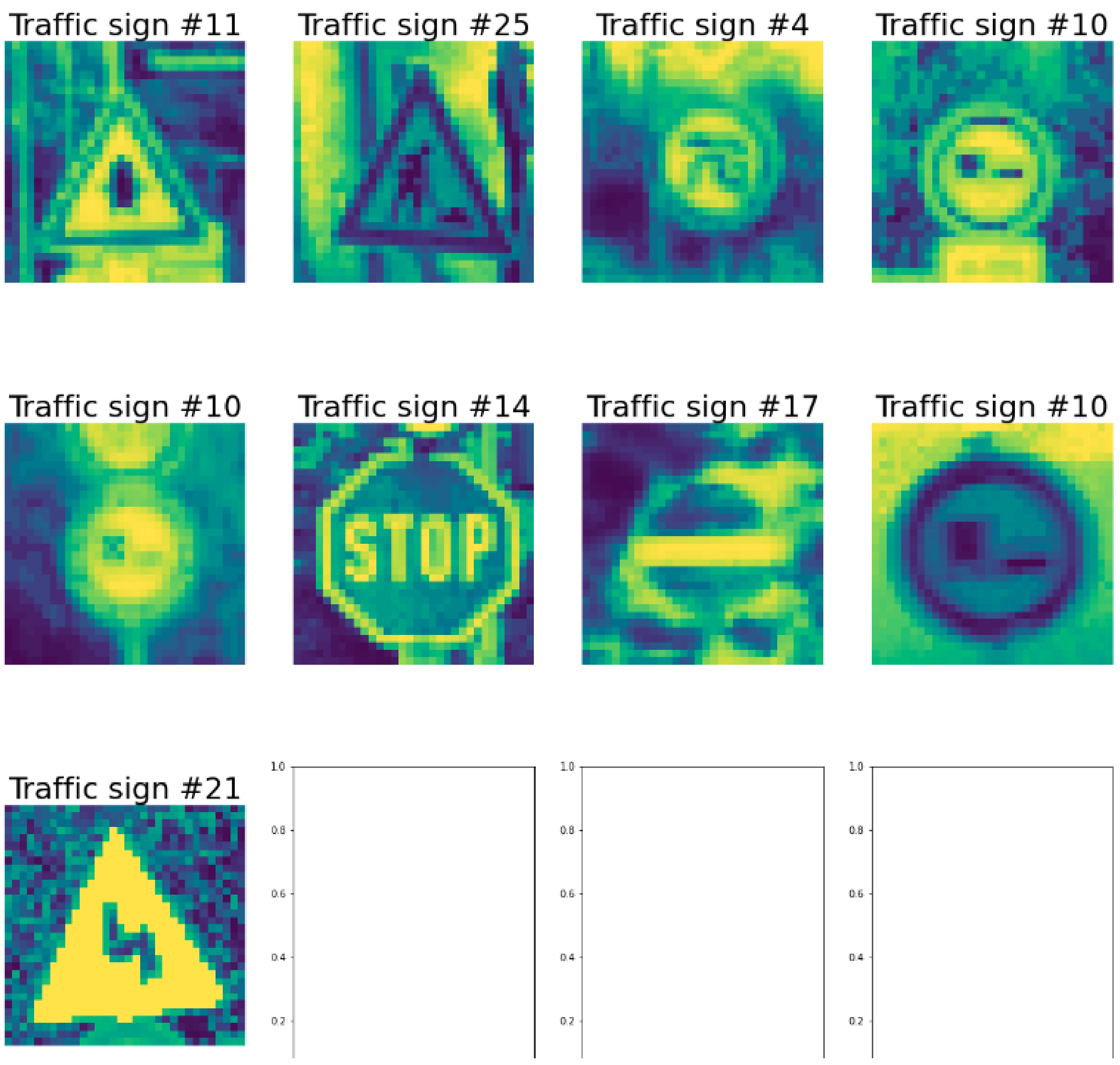
I have used randomly selected 9 German traffic signs that are provided in the test set of the German Traffic Sign Dataset, so no additional split of the training set was required.

I made this decision, because:

- this training set is statistically significant

- images are of the same size
- poor quality of the images
- poor quality of image capturing in terms of angle and position of the traffic signs

Here are the exemplary 9 images, that were randomly selected for testing (after being preprocessed to grayscale, but depicted here in color for better perception by the human eye):



Surprisingly, despite poor quality of the images, all but one have been correctly classified. So the correct prediction rate of these 9 images is 88.889%.

The second image of the "Road work" traffic sign has been wrongly classified as "Right-of-way at the next intersection" with 99.5% probability and with 0.01% probability as "Road work" itself.

Reasons for this are:

- high pixelization of the images, which doesn't allow the model to derive precise shapes
- In the training dataset, there are roughly 700 images for traffic sign #11 ("Right-of-way at the next intersection") and roughly 500 images for traffic sign #25 ("Road work")

Other difficulties for traffic sign recognition:

- background noise (e.g. trees)
- unequal distribution of the traffic signs over the training set
- low light capture conditions
- multiple traffic signs in one image (e.g. traffic sign #10)

Predictions on these 9 traffic signs vs. predictions on the whole test set.

The test results of these 9 randomly selected traffic signs with 88.9% recognition rate are worse recognized, than the overall test set with 93.5% or the validation set with 96.1%. Reasons for this are:

- single step in evaluation of 9 images is 9/10 = 11.1%
- statistically significant pool with 1 badly recognized image (1 out of 6.5% of wrongly classified traffic signs)

Here are the results of the prediction:

Meaning actual	**Meaning predicted**
Right-of-way at the next intersection	Right-of-way at the next intersection
Road work	Right-of-way at the next intersection
Speed limit (70km/h)	Speed limit (70km/h)
No passing for vehicles over 3.5 metric tons	No passing for vehicles over 3.5 metric tons
No passing for vehicles over 3.5 metric tons	No passing for vehicles over 3.5 metric tons
Stop	Stop
No entry	No entry
No passing for vehicles over 3.5 metric tons	No passing for vehicles over 3.5 metric tons
Double curve	Double curve

The model was able to correctly guess 8 of the 9 traffic signs, which gives an accuracy of 88.9%.

Prediction of each of the 9 new images in detail

The code for making predictions on my final model is located in the 16th cell of the Ipython notebook. Here's it once again:

```
    saver.restore(sess, tf.train.latest_checkpoint('.'))

    batch_x, batch_y = X_test[0:IMAGES_AMOUNT], y_test[0:IMAGES_AMOUNT]
    predictions = logits.eval(feed_dict={x: batch_x, y: batch_y, dropout_prob: 
1.0})
    predicted_labels = (np.argmax(predictions, axis=1))
    print("Predicted lables to the above traffic signs:")
    print(predicted_labels)
```

Here are the probabilities of a traffic sign being correctly classified:

Label #	Meaning	Probability
11	Right-of-way at the next intersection	100%
25	Road work	99.5%
4	Speed limit (70km/h)	98.7%
10	No passing for vehicles over 3.5 metric tons	100%
10	No passing for vehicles over 3.5 metric tons	100%
14	Stop	100%
17	No entry	100%
10	No passing for vehicles over 3.5 metric tons	100%
21	Double curve	87.7%

The softmax top-5 probabilities for these images are:

Label #	Meaning	Softmax top-5 probabilities with corresponding labels
11	Right-of-way at the next intersection	Label 11: 100% Label 30: 0% Label 27: 0% Label 21: 0% Label 18: 0%
25	Road work	Label 11: 99.52% Label 18: 0.26% Label 27: 0.18% Label 30: 0.02% Label 25: 0.01%
4	Speed limit (70km/h)	Label 4: 98.57% Label 0: 0.81% Label 1: 0.42% Label 8: 0.15% Label 7: 0.03%
10	No passing for vehicles over 3.5 metric tons	Label 10: 100% Label 19: 0% Label 23: 0% Label 9: 0% Label 16: 0%
10	No passing for vehicles over 3.5 metric tons	Label 10: 100% Label 5: 0% Label 42: 0% Label 3: 0% Label 7: 0%
14	Stop	Label 14: 100% Label 17: 0% Label 25: 0% Label 33: 0% Label 3: 0%
17	No entry	Label 17: 100% Label 14: 0% Label 39: 0% Label 33: 0% Label 37: 0%
10	No passing for vehicles over 3.5 metric tons	Label 10: 100% Label 5: 0% Label 9: 0% Label 7: 0% Label 42: 0%

Label #	Meaning	Softmax top-5 probabilities with corresponding labels
21	Double curve	Label 21: 87.73% Label 11: 7.75% Label 30: 3.06% Label 23: 0.55% Label 19: 0.35%